

الجمهورية الجزائرية الديمقراطية الشعبية  
وزارة التعليم العالي والبحث العلمي

UNIVERSITÉ BADJI MOKHTAR - ANNABA  
BADJI MOKHTAR – ANNABA UNIVERSITY



جامعة باجي مختار – عنابة

Faculté : Sciences de L'ingénierie

Département : Electronique

Domaine : Sciences et Techniques

Filière : Electronique

Spécialité : Instrumentation

Mémoire

Présenté en vue de l'obtention du Diplôme de Master

Thème:

**Design And Development Of Smart contactless  
Thermometer using face recognition**

Présenté par : *Zehir Hatem*

Encadrant : *Bekaik Mounir*

MCB

Université d'Annaba

**Jury de Soutenance :**

Bousbia Salah Mounir	Prof	Université d'Annaba	Président
Bekaik Mounir	MCB	Université d'Annaba	Encadrant
Ait Izem Tarek	MCB	Université d'Annaba	Examineur

Année Universitaire : 2020/2021

## Abstract

This project presented for obtaining the master's degree in Instrumentation consists of producing a smart thermometer based on Raspberry Pi 4b.

Living with the Novel Coronavirus is becoming the new way of life in countries all over the world. However, to stop the virus from spreading, people who have the Coronavirus must be separated from the rest of the population. Our work is an attempt to make a smart contactless thermometer based on the MLX90614 sensor and the Raspberry Pi.

**Keywords:** Facial Recognition, Raspberry Pi, Python, Database

## Résumé

Ce projet présenté pour l'obtention du diplôme de master en Instrumentation consiste à réaliser un thermomètre intelligent à base de Raspberry Pi 4b.

Vivre avec le nouveau coronavirus devient le nouveau mode de vie dans les pays du monde entier. Cependant, pour empêcher le virus de se propager, les personnes atteintes du coronavirus doivent être séparées du reste de la population. Notre travail est une tentative de créer un thermomètre sans contact intelligent basé sur le capteur MLX90614 et le Raspberry Pi.

**Mots-clés :** Reconnaissance Faciale, Raspberry Pi, Python, Base de données

## ملخص

قدم هذا المشروع للحصول على درجة الماجستير في تخصص الأدوات وهو قائم على انجاز مقياس حرارة ذكي اعتمادا على الـ راسبيري باي b4.

أصبح التعايش مع فيروس كورونا المستجد طريقة جديدة للحياة في البلدان في جميع أنحاء العالم. ومع ذلك، لمنع انتشار الفيروس، يجب فصل الأشخاص المصابين بفيروس كورونا عن باقي السكان. عملنا هو محاولة لصنع مقياس حرارة ذكي بدون تلامس يعتمد على مستشعر MLX90614 و Raspberry Pi.

**الكلمات المفتاحية :** التعرف على الوجه، راسبيري باي، بايثون، قاعدة بيانات

## Dedication

*This thesis is dedicated to my loving parents for their patience, support, and sacrifices from the moment I was born, throughout my childhood, and into adulthood.*

*My dear mother, Thank you for your suggestions, sacrifices, assistance, and encouragement; this is your achievement, not mine.*

*My father, who can be proud of the outcome of many years of sacrifices and hardships to assist me in moving forward in life. the education and unwavering support that you provide.*

*To all my friends*

*To all my teachers*

## Acknowledgments

*First and foremost, praise be to Allah, who has given us the wonderful faculty of reasoning.*

*My gratitude and thanks go out to my parents, who have been nothing but supportive and loving, as well as the rest of my family.*

*I'd like to express my sincere thanks to Dr. BEKAIK Mounir, my supervisor. To have patiently supported and trusted me throughout my project.*

*I also want to express my thanks to all of the teachers in the electronics department who have provided us with a wealth of information.*

*I also want to thank each one of my colleagues who has always been supportive of my efforts. Finally, I want to express our gratitude to everyone who helped me complete this project, whether from near or far.*

*Annaba, 2021  
ZEHIR Hatem*

## List of tables

<i>Table 1: The fast diagram of the fever detector</i> .....	6
--	---

## List of Figures

<i>Figure I.1: the horned beast diagram of the fever detector</i> .....	4
<i>Figure I.2: the Octopus diagram of the fever detector</i> .....	4
<i>Figure I.3: The use case diagram of the fever detector</i> .....	10
<i>Figure I.4: Sequence diagram of the fever detector</i> .....	11
<i>Figure II.1: The Raspberry Pi 4b board</i> .....	13
<i>Figure II.2: GPIO connector's pin layout on Raspberry Pi 4b</i> .....	14
<i>Figure II.3: Enable VNC server</i> .....	15
<i>Figure II.4: MLX90614 IR temperature sensor</i> .....	16
<i>Figure II.5: Raspberry Pi Camera Module v1.3</i> .....	17
<i>Figure II.6: How to split one gradient vector's magnitude if its degrees is between two degrees bins</i> .....	18
<i>Figure II.7: The overall architecture of CNN includes an input layer, multiple alternating convolutions, and max-pooling layers, one fully connected layer, and one classification layer</i> .....	19
<i>Figure II.8: Image matrix multiplies kernel or filter matrix</i> .....	19
<i>Figure II.9: 4 x 4 Output matrix</i> .....	20
<i>Figure II.10: Two images to compare</i> .....	20
<i>Figure II.11: The 2 images with coded pixels</i> .....	20
<i>Figure II.12: pixel by pixel comparison</i> .....	21
<i>Figure II.13: some pieces of the image are identical between the 2 images</i> .....	21
<i>Figure II.14: features that we want the machine to find</i> .....	21
<i>Figure II.15: We will start by looking for this feature on this image</i> .....	21
<i>Figure II.16: feature dragging</i> .....	22
<i>Figure II.17: Multiply the pixel values of the feature with the pixel values of the piece of the image to find</i> .	22
<i>Figure II.18: add the results</i> .....	22
<i>Figure II.19: a neural network with one input and one output</i> .....	23
<i>Figure II.20: tanh</i> .....	24
<i>Figure II.21: ReLU function</i> .....	24
<i>Figure II.22: Leaky ReLU function</i> .....	24
<i>Figure II.23: Sigmoid function</i> .....	24
<i>Figure II.24: Types of pooling max, average, and sum pooling</i> .....	25
<i>Figure II.25: Libraries used in the program</i> .....	28
<i>Figure II.26: Initialization</i> .....	29
<i>Figure II.27: The database creation code</i> .....	30
<i>Figure II.28: Load Images</i> .....	30
<i>Figure II.29: Encoding and identifying the faces</i> .....	31
<i>Figure II.30: facial similarity</i> .....	32
<i>Figure II.31: Checking the temperature</i> .....	33
<i>Figure II.32: QR code</i> .....	34
<i>Figure II.33: Project circuit</i> .....	34
<i>Figure III.1: Facial recognition and temperature check</i> .....	33
<i>Figure III.2: a red rectangle will be drawn around the face if the temperature is higher than 38°C</i> .....	34

*Figure III.3: employees table in the database* ..... 35  
*Figure III.4: RedZone table in the database* ..... 35  
*Figure III.5: The green LED will turn on if the temperature is lower than 38°C*..... 36  
*Figure III.6: The red LED will turn on if the temperature is higher than 38°C* ..... 36  
*Figure III.7: The QR Code can be read by the Raspberry Pi camera module* ..... 37  
*Figure III.8: If a negative test is provided, the person will be deleted from the RedZone table*..... 38  
*Figure III.9: If the temperature is lower than 38 °C the engine will say “Normal Temperature”* ..... 38  
*Figure III.10: If the temperature is higher than 38 °C the engine will say “High Temperature”* ..... 39

## List of abbreviations

SARS-COV2: Severe acute respiratory syndrome coronavirus 2

IR: Infrared

FDS: Functional Design Specification

FR: Functional Requirements

CR: Constraint Requirements

SQL: Structured Query Language

LED: light-emitting diode

QR code: Quick Response code

FAST: Function Analysis System Technique

AFNOR: association of French Normalization

ISO: International Organization for Standardization

UML: Unified Modeling Language

GPIO: General-purpose input/output

CNN: Convolutional neural network

HOG: Histogram of oriented gradients

USB: Universal Serial Bus

LPDDR: Low-Power Double Data Rate

SDRAM: Synchronous dynamic random-access memory

FPS: frames per second

HEVC: High-Efficiency Video Coding

LCD: Liquid-crystal display

SDA: Serial Data Line

SCL: Serial Clock Line

VNC: Virtual Network Computing

LAN: Local area network

Wi-Fi: Wireless Fidelity

ADC: Analog-to-digital converter

PWM: Pulse-width modulation



SMBus: System Management Bus

VDD: Voltage Drain Drain

GND: Ground

SVM: Support vector machine

RGB: Red, Green, Blue

CSI: Camera Serial Interface

# Content

<b>ABSTRACT</b> .....	<b>I</b>
<b>RESUME</b> .....	<b>I</b>
<b>ملخص</b> .....	<b>I</b>
<b>DEDICATION</b> .....	<b>II</b>
<b>ACKNOWLEDGMENTS</b> .....	<b>III</b>
<b>LIST OF TABLES</b> .....	<b>IV</b>
<b>LIST OF FIGURES</b> .....	<b>V</b>
<b>LIST OF ABBREVIATIONS</b> .....	<b>VII</b>
<b>CONTENT</b> .....	<b>1</b>
<b>GENERAL INTRODUCTION</b> .....	<b>4</b>
<b>I. THE FUNCTIONAL DESIGN OF THE FEVER DETECTOR</b> .....	<b>3</b>
I.1 INTRODUCTION.....	3
I.2 WHAT IS FUNCTIONAL DESIGN .....	3
I.2.1 Functional Design Specification .....	3
I.3 THE HORNED BEAST DIAGRAM .....	3
I.3.1 The Horned Beast diagram of the fever detector .....	3
I.4 THE OCTOPUS DIAGRAM.....	4
I.4.1 The Octopus diagram of the fever detector.....	4
I.5 THE FAST DIAGRAM .....	5
I.5.1 What are the benefits of the FAST diagram.....	5
I.5.2 The interrogative technique of the FAST diagram .....	5
I.5.3 The fast diagram of the fever detector .....	6
I.6 THE INTERNATIONAL STANDARDS .....	8
I.6.1 ISO 13485: the Quality Management System (“QMS”) for the design and manufacture of Medical Devices .....	8
I.6.1.1 What are the requirements of ISO 13485?.....	8
I.6.2 ISO 14971: the Risk Management for Medical Devices .....	9
I.6.2.1 What is the role of ISO 14971?.....	9
I.7 USE CASE DIAGRAM.....	9
I.7.1 What is the Unified Modeling Language (UML)? .....	10
I.7.2 The use case diagram of the fever detector .....	10
I.8 SEQUENCE DIAGRAM .....	10
I.8.1 Sequence diagram of the fever detector .....	11
I.9 CONCLUSION.....	11
<b>II. CONCEPTION OF THE FEVER DETECTOR</b> .....	<b>13</b>
II.1 INTRODUCTION .....	13
II.2 THE RASPBERRY PI 4B .....	13
II.2.1 General Purpose Input Output (GPIO) .....	14
II.2.2 VNC server.....	14
II.2.2.1 Configure the VNC server .....	14
II.2.2.1.1 Install VNC server.....	14

II.2.2.1.2	Enable VNC server .....	15
II.3	TEMPERATURE SENSOR (MLX90614).....	15
II.3.1	Characteristics.....	16
II.3.2	Pinout Configuration.....	16
II.3.3	Working Principle.....	16
II.4	RASPBERRY PI CAMERA MODULE V1.3.....	16
II.4.1	Characteristics.....	17
II.5	POPULAR FACE RECOGNITION ALGORITHMS.....	17
II.5.1	Histogram of oriented gradients (HOG).....	17
II.5.1.1	Definition.....	17
II.5.1.2	How HOG works.....	17
II.5.2	HaarCascade Classifiers.....	18
II.5.3	Convolution Neural Network (CNN).....	18
II.5.3.1	Definition.....	18
II.5.3.2	Convolution Neural Network architecture.....	18
II.5.3.3	Convolution layer.....	19
II.5.3.4	What is Weight.....	23
II.5.3.5	Activation functions.....	23
II.5.3.6	RELU Layer.....	24
II.5.3.7	Pooling layer.....	24
II.5.3.8	Fully connected layer.....	25
II.5.3.9	Cost function.....	25
II.5.3.9.1	Cost function requirements.....	25
II.5.3.9.2	List of cost functions used in Neural Networks.....	26
II.5.3.10	Back propagation.....	26
II.5.3.11	Gradient Descent.....	26
II.5.3.12	Stochastic gradient Descent.....	26
II.6	WHAT IS THE BEST FACE RECOGNITION ALGORITHM.....	26
II.7	PROGRAMMING.....	26
II.7.1	Libraries.....	26
II.7.2	Initialization.....	28
II.7.3	Create an SQL database.....	29
II.7.3.1	What is SQL.....	29
II.7.3.2	What is SQLite.....	29
II.7.3.3	What is the difference between SQL and SQLite?.....	29
II.7.3.4	The database creation code.....	30
II.7.4	Face Recognition.....	30
II.7.4.1	Load Images.....	30
II.7.4.2	Encoding and identifying the faces.....	31
II.7.4.3	Similarity.....	31
II.7.5	Checking the temperature.....	32
II.7.6	QR code.....	33
II.8	PROJECT CIRCUIT.....	34
II.9	CONCLUSION.....	35
<b>III.</b>	<b>SIMULATION AND FINAL RESULTS.....</b>	<b>33</b>
III.1	INTRODUCTION.....	33
III.2	SIMULATION OF RESULTS.....	33
III.2.1	Facial recognition and temperature check.....	33
III.2.2	Database.....	34
III.2.3	LEDs activation.....	35
III.2.4	QR Code.....	37

III.2.5	Speech synthesis .....	38
III.3	CONCLUSION .....	39
	<b>GENERAL CONCLUSION</b> .....	<b>40</b>
	<b>BIBLIOGRAPHY</b> .....	<b>41</b>

## GENERAL INTRODUCTION

Today, we are going through one of the biggest health crises in history due to the spread of the COVID-19<sup>[1][2]</sup> and the increase in the number of Coronavirus cases.

The COVID-19 is caused by the SARS-COV2 virus, this novel respiratory disease has affected most people's way of life around the globe.

This virus spread quickly to most of the countries in just a few weeks after it was first reported from Wuhan, China, on 31 December 2019, until April 5<sup>th</sup>, 2021 the total number of identified cases was 132,211,006 while taking the lives of 2,869,484 peoples.

Common symptoms of COVID-19 include high body temperature (above 38°C), dry cough, tiredness, in addition to other symptoms such as aches and pains headache, loss of taste or smell, and difficulty breathing or shortness of breath.

To reduce the coronavirus cases and stop the spread of the virus, many safety procedures were taken by governments, such as obligatory mask-wearing, social distancing, and regularly test citizens. Most countries are also making temperature checkups and mask mandatory for schools, offices, and other workspaces.

Unfortunately, in some countries (especially third world countries), there is not enough test for all people, which makes it impossible to identify all of the positive cases and protect the citizens.

In this pandemic, companies must continue to produce and provide their services, but without neglecting the coronavirus health protection regulations. During virus outbreaks like COVID-19 companies around the globe must play a major role to stop the spread of the disease, especially in detecting employees or customers that may have been infected by the virus.

Companies today must be integrated into the governmental health protection protocol developed by governments and the world health organization.

Companies must ensure the continuity of their services but also plan for detecting cases of infected employees or customers. They must also prepare for the psychological effects of the pandemic.

The companies must implement a dedicated response with health care services and protect their employees during their work.

Today, temperature checkups are done manually using IR contactless thermometer, which is inefficient and impractical especially in places with large crowds.

To solve all of these problems, we have designed a small and portable device<sup>[14][20][22]</sup> that automates the process of temperature checkup by using a contactless IR temperature sensor and facial recognition<sup>[4][17][21][23][27]</sup>.

Our solution involves building a device capable of detecting a person in an office, company, or home when he has a high body temperature. This can serve as an early warning system we can make a plan of action ahead of time. This product will serve as a way of protecting the company, the employees, and the customer.

The device is designed to recognize the faces and to determine whether the person has a high body temperature or not. Using the collected data, we can decide whether the concerned person can be allowed inside a public place or not.

For this, we will use the Raspberry Pi 4b<sup>[19][26]</sup> because of its low cost, vast peripheral support (it comes with 26 GPIO Pins), small size, and huge processing power. We will also use the open-source programming language Python because of its vast libraries support.

The use of this system is not limited to companies, offices, schools, buildings but can also be used in high-risk areas like hospitals. It can also be used in places with large crowds such as train stations, stadiums, and airports.

Our work is organized as follow: the first chapter is dedicated to the functional design<sup>[7][10]</sup> of the system, in this section, we will draw the following diagrams: The Horned Beast diagram, the Octopus Diagram<sup>[12][24]</sup>, the FAST Diagram<sup>[6][13]</sup>, and by using StarUML<sup>[30]</sup> we will create the Sequence diagram and the use case diagram<sup>[25]</sup>. In the second chapter, we will focus on the hardware and techniques used in the project such as the Raspberry Pi 4b, the MLX90614 sensor<sup>[3][16]</sup>, face recognition algorithms, QR code<sup>[5][8][9]</sup>, and the SQL database. Finally, in the third chapter, we will discuss the final results of the project.

# **CHAPTER 1**

## **THE FUNCTIONAL DESIGN OF THE FEVER DETECTOR**

# **I. THE FUNCTIONAL DESIGN OF THE FEVER DETECTOR**

## **I.1 Introduction**

Functional design is a design technique that looks at the system from a functional standpoint. The design focuses on separating high-level functions from lower-level functions, which can then be decomposed and synthesized. The development process is organized into a series of step-by-step functional refinements.

## **I.2 What is functional design**

In the development of a project, the functional design phase focuses on isolating the high-level functions that can be decomposed into and synthesized from lower-level functions.

The functional design focus on function rather than aesthetics, it cares about objectives rather than components. It can also refer to the use of a complete requirements document to lead the development and testing of a product.

### **I.2.1 Functional Design Specification**

A Functional Design Specification, or FDS, is a document that explains how a process or control system will work.

There are no highly technical details in the Functional Design Specification. Instead, it explains how the proposed system will work, how people will interact with it, and what to expect in various operational scenarios.

## **I.3 The Horned Beast diagram**

The horned beast diagram is used in functional design and project management, before drawing a horned beast diagram; we must answer the following three key questions:

- Whom will the project serve?
- What does it affect or interact with?
- For what goal?

### **I.3.1 The Horned Beast diagram of the fever detector**

Let's now answer these questions to create the horned beast diagram of our fever detector:

- Whom will the project serve? It is useful for companies.
- What does it affect or interact with? It is part of instrumentation.
- For what goal? To detect if a person has a fever or not.



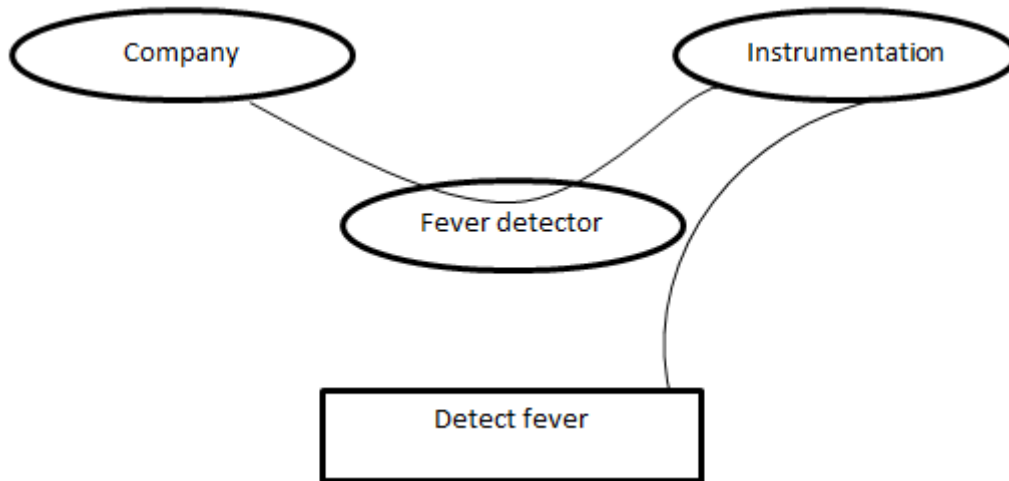


Figure I.1: the horned beast diagram of the fever detector

After creating the horned beast diagram (see figure I.1) and answering the three questions, we can simply tell the requirements of our project with a simple sentence: “The fever detector is an instrumentation tool useful for the company, it allows detecting if a person has a fever or not”

#### I.4 The Octopus Diagram

The octopus diagram has the purpose of illustrating the association between the system and the different elements of its environment. It is applied after analyzing the needs of the customer; the diagram allows the listing of the different functional requirements of systems, as well as the constraint requirements.

- **Functional Requirements (FR):** it represents the interactions of the system with its environment.
- **Constraint Requirements (CR):** it refers to the adoption or the action of the system, in other words, the product has to adopt or act on an element.

##### I.4.1 The Octopus diagram of the fever detector

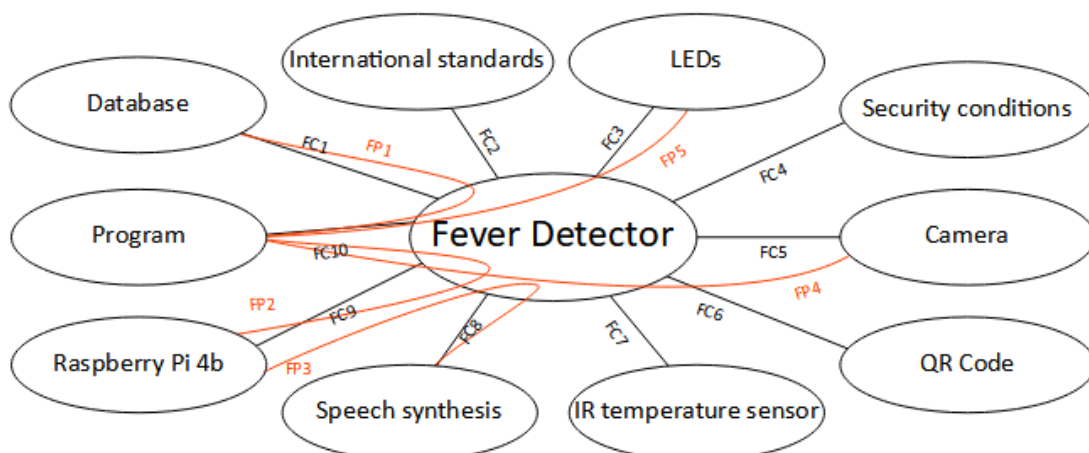


Figure I.2: the Octopus diagram of the fever detector

We can now easily announce the deferent functions of the fever detector:

- FR1: setting a facial recognition program
- FR2: use SQL to create a database
- FR3: check the temperature of every person
- FC1: create a database using SQL
- FC2: respect the international standards
  - ✓ ISO 13485
  - ✓ ISO 14971
- FC3: turn ON/OFF LEDs green/red
- FC4: define security conditions
- FC5: detect faces with a camera
- FC6: use QR code to remove healthy persons from the database
- FC7: check the temperature
- FC8: turn acquired information into a vocal message
- FC9: give orders and receive sensor information
- FC10: artificial intelligence-based program

## **I.5 The FAST Diagram**

The Function Analysis System Technique (FAST) is a technique to highlight the relationships between the different functions of a product or service using a graphical representation.

The association of French Normalization (AFNOR) defines the FAST diagram as one of the methods of functional design.

The Function Analysis System Technique helps us solve the problem objectively and identifying its scope by showing the logical relationships between functions. The FAST diagram enables engineers to verify if, and illustrate how, a proposed solution achieves the needs of the project, and to identify missing, duplicated, or unnecessary functions.

### **I.5.1 What are the benefits of the FAST diagram**

The Function Analysis System Technique is a creative way of thought that encourages more communication between team members. It helps them to:

- Create a shared understanding of the system
- Determine missing and unnecessary functions
- Simplify the given problem
- Comprehend and systematize the relationships between functions
- Build better communication between team members
- Encourage creativity

### **I.5.2 The interrogative technique of the FAST diagram**

The Function Analysis System Technique relies on an interrogative approach, before completing a FAST diagram; we must answer the following three key questions:

- **HOW** can this function be achieved?
- **WHY** do we do this function?
- **WHEN** this function occurs, what other functions should we do?

The answers to these questions are neither unique nor exclusive, they can be singular, multiple, or optional. The FAST functions must be described with a verb in the infinitive form.

### I.5.3 The fast diagram of the fever detector

After following all of the steps mentioned earlier, we can create the FAST diagram of the fever detector

**Table 1:** The fast diagram of the fever detector

Function name	Criteria	Technologic solution
FR1: setting a facial recognition program	FR11: Initialize the face_recognition and OpenCV libraries FR12: Encode known faces FR13: Recognize faces in two steps FR131: detects the presence and location of a face in an image, but does not identify it. FR132: extract 128-d feature vectors (called “embeddings”) that quantify each face in an image	Python SQL MLX90614 LEDs Raspberry Pi 4b Camera
FR2: use SQL to create a database	FR21: create an SQL database FR22: store every person temperature in a database table FR23: if the temperature is high store the information in another database table “RedZone” FR24: is a person with a high temperature provides a negative covid-19 test, he will be removed from the “RedZone” table	
FR3: check the temperature of every person	FR31: Recognize the person standing next to the camera FR32: Check his temperature FR33: Store the information	

<p>FC1: create a database using SQL</p>	<p>FC11: create an SQL database using python  FC12: create two tables  FC121: create an “employees” table  FC122: create “RedZone” table</p>	
<p>FC2: respect the international standards</p>	<p>FC21: respect the international standards (ISO 13485 and ISO 14971)  FC211: respect the Medical standard ISO 13485  FC212: respect the Application of risk management to medical devices ISO 14971</p>	
<p>FC3: turn ON/OFF LEDs green/red</p>	<p>FC31: normal temperature: turn on the green LED and turn off the red LED  FC32: high temperature: turn on the red LED and turn off the green LED</p>	
<p>FC4: define security conditions</p>	<p>FC41: define security conditions of the working environment</p>	
<p>FC5: detect faces with a camera</p>	<p>FC51: recognize faces with a camera in two steps  FC511: detect the faces  FC522: recognize the faces</p>	
<p>FC6: use QR code to remove healthy persons from the database</p>	<p>FC61: read a QR code  FC611: if the covid-19 test is negative remove the person from the “RedZone” table  FC612: if the covid-19 test is positive leave the entry in the “RedZone” table</p>	
<p>FC7: check the temperature</p>	<p>FC71: check the temperature using an IR temperature sensor  FC711: send the reading to a database  FC712: visualize the reading on a screen</p>	

<p>FC8: turn acquired information into a vocal message</p>	<p>FC81: turn acquired information into a vocal message  FC811: if the temperature is below 38°C: “normal temperature”  FC811: if the temperature is above 38°C: “high temperature”</p>	
<p>FC9: give orders and receive sensor information</p>	<p>FC91: the raspberry Pi 4b gives orders and receive sensor information  FC911: receive sensor readings  FC912: turn ON/OFF LEDs green/red  FC913: store information in the database  FC914: communicate with the camera</p>	
<p>FC10: artificial intelligence-based program</p>	<p>FC101: artificial intelligence-based program  FC1011: get sensor readings  FC1012: recognize faces  FC1013: store information on the database</p>	

## I.6 The international standards

The international standards are technical norms developed by intergovernmental organizations such as the World Health Organization Guidelines; these organizations are publicly funded and freely available for worldwide use and consideration.

They may be used either by direct application or by modifying a standard to suit a current condition or function. In our product, we must respect two International standards: ISO 13485 and ISO 14971.

### I.6.1 ISO 13485: the Quality Management System (“QMS”) for the design and manufacture of Medical Devices

ISO 13485 is the most used international standard for quality management in the medical devices industry. It is an effective solution to demonstrate a commitment to the safety and quality of medical devices.

The current version of ISO 13485 is ISO 13485:2016 from march 2016.

#### I.6.1.1 What are the requirements of ISO 13485?

The ISO 13485:2016 requirements apply to all organizations except where explicitly stated. It specifies requirements to manufacture medical devices that meet customer needs.

This international standard is split into eight sections; the last five sections are the most important:

- Section 1, section 2 and section 3: introduction
- Section 4: Quality Management System
- Section 5: Management Responsibility
- Section 6: Resource Management
- Section 7: Product Realization
- Section 8: Measurement, Analysis, and Improvement

### **I.6.2 ISO 14971: the Risk Management for Medical Devices**

ISO 14971 is the international standard addressed to risk management for the medical devices industry. It determines the best practices at all stages in a device's life cycle from the design phase to the production and maintenance phases.

We often don't have enough data to precisely define risks which makes risk management a tricky task. The risk and risk management for medical devices are defined as:

- **Risk:** it is the probability of occurrence of harm and the dangerousness of that harm.
- **Risk management:** it is the application of management policies, procedures, and practices to the tasks of risk analysis, control, and monitoring in a systematic manner.

#### ***I.6.2.1 What is the role of ISO 14971?***

The international standard ISO 14971 defines a process to identify all risks associated with medical devices throughout the life cycle of the product, its goal is to analyze, evaluate, control, and monitor the risks associated with each life-cycle phase.

The latest revision of ISO 14971 is ISO 14971:2019

This international standard is split into ten sections; with the first three being introductory. Here is what the main sections are about:

- Section 4: general requirements for risk management system
- Section 5: risk analysis
- Section 6: risk evaluation
- Section 7: risk control
- Section 8: evaluation of overall residual risk
- Section 9: risk management review
- Section 10: production and post-production activities

### **I.7 Use case diagram**

A use case diagram can summarize the details of the system's users (also known as actors) and their interactions with the system in the Unified Modeling Language (UML). We need a set of specialized symbols and connectors to construct one. A good use case diagram can assist the team in discussing and representing the following:

- Interactions between the system or application and people, organizations, or external systems.
- Goals that the system or application assists those entities (also referred to as actors) in achieving.
- The system's capabilities.

### I.7.1 What is the Unified Modeling Language (UML)?

UML (Unified Modeling Language) is a standardized modeling language that consists of an integrated set of diagrams that was created to assist system and software developers in specifying, visualizing, constructing, and documenting software system artifacts, as well as business modeling and other non-software systems. It expresses software project design primarily through graphical notations. The UML aids project teams in communicating, exploring potential designs, and validating the software's architectural design.

### I.7.2 The use case diagram of the fever detector

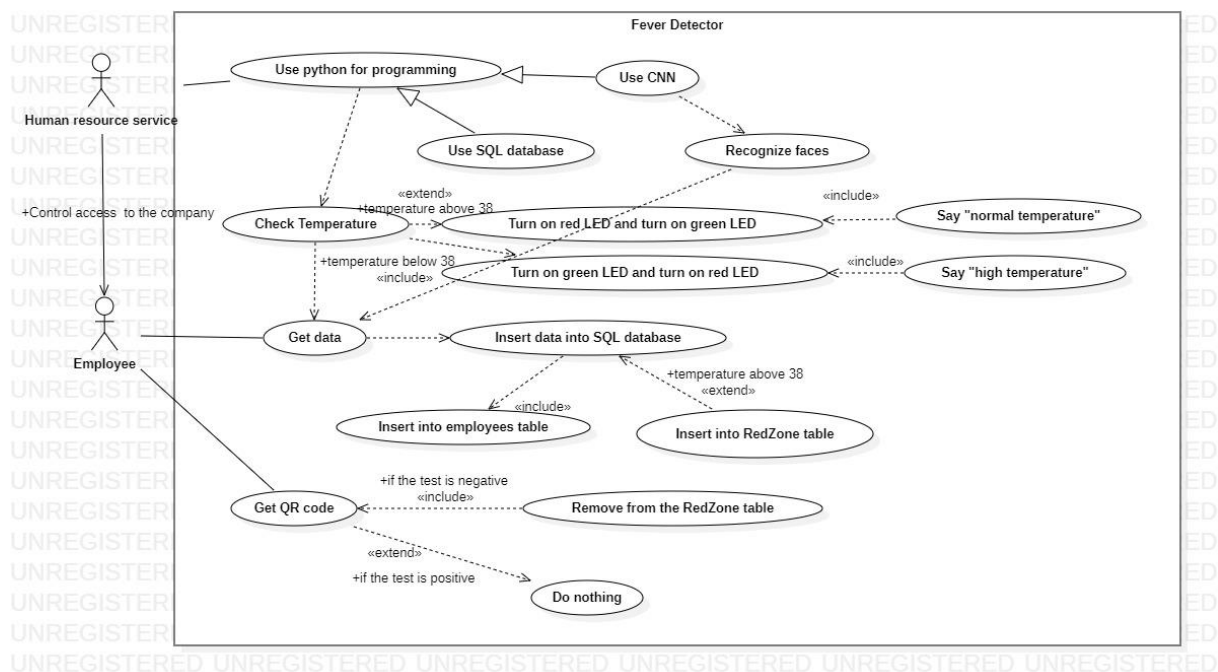


Figure I.3: The use case diagram of the fever detector

## I.8 Sequence diagram

A sequence diagram simply depicts the order in which objects interact or the order in which these interactions occur. A sequence diagram can also be referred to as an event diagram or an event scenario. Sequence diagrams show how and in what order the components of a system work together. Businessmen and software developers frequently use these diagrams to document and understand requirements for new and existing systems.

## I.8.1 Sequence diagram of the fever detector

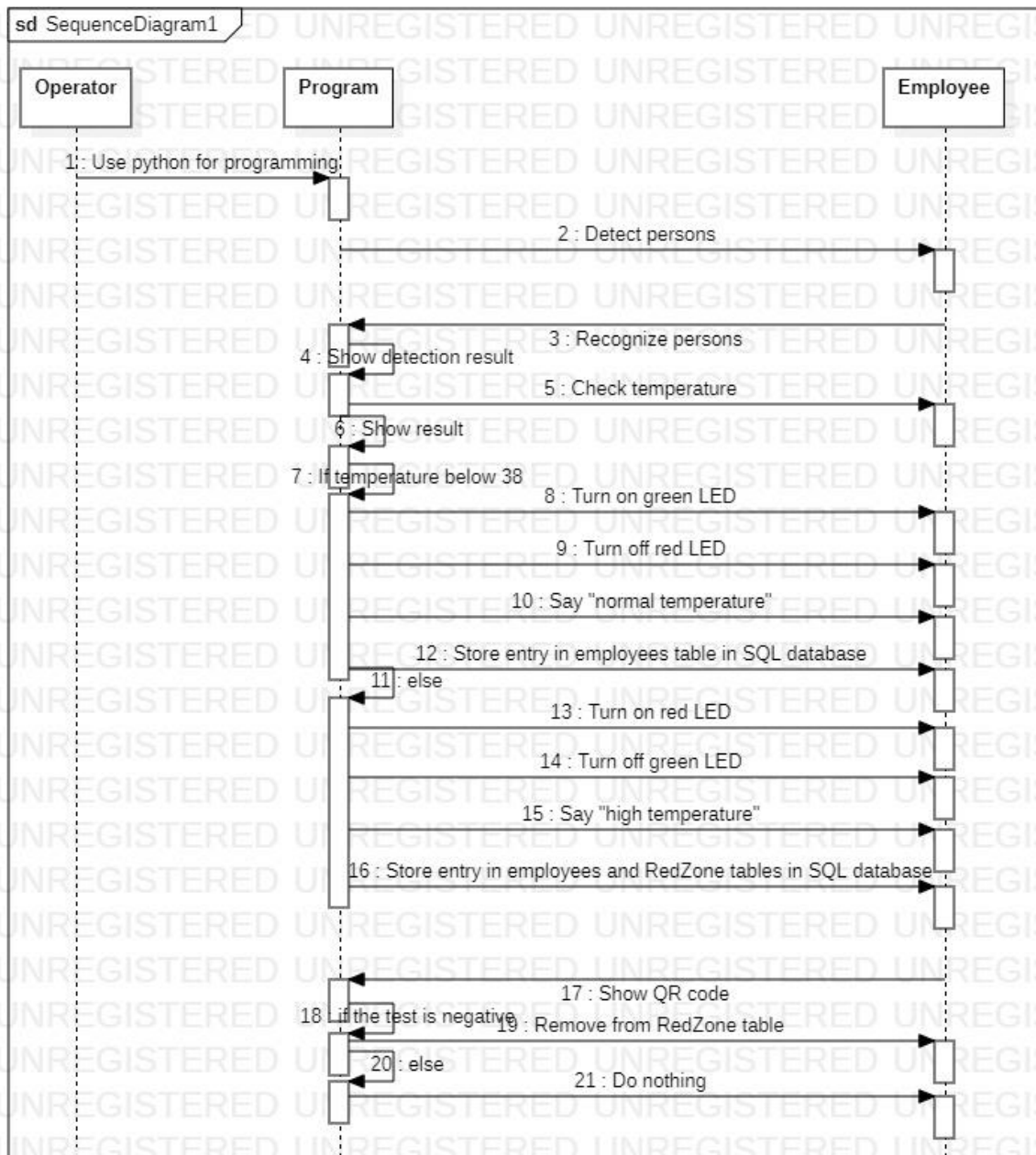


Figure I.4: Sequence diagram of the fever detector

## I.9 Conclusion

The horned beast diagram, the octopus diagram, the fast diagram, use case diagram, and sequence diagram are all tools and methods that can be used in functional analysis to illuminate the functionalities of the product and their relationships with surrounding elements.



## **CHAPTER 2**

### **CONCEPTION OF THE FEVER DETECTOR**

## II. CONCEPTION OF THE FEVER DETECTOR

### II.1 Introduction

This chapter is divided into three parts:

The first part explains the hardware used in our project: a definition of the Raspberry Pi 4b and its general-purpose input-output (GPIO), The MLX90614 IR temperature sensor and its characteristics, and the Raspberry Pi camera module.

The second part is a study about the popular algorithms used in face recognition: Convolution Neural Network (CNN), Histogram of oriented gradients (HOG), and HaarCascade Classifiers.

The third part presents a closer look at the program.

### II.2 The Raspberry Pi 4b

The Raspberry Pi 4b was released in June 2019. It is a 64-Bit, 1.5 GHz quad core, with 2 GB, 4 GB or 8 GB of LPDDR4 SDRAM. It has also full-throughput Gigabit Ethernet, dual-band 802.11ac wireless networking, Bluetooth 5.0, two USB 3.0 and two USB 2.0 ports, dual monitor support at resolutions up to 4K 60FPS, VideoCore VI graphics supporting OpenGL ES 3.x, 4Kp60 hardware decode of HEVC video and complete compatibility with earlier Raspberry Pi products.

Power consumption is 3.4 watts at idle, that number jumps to 7.6 watts under full stress. Figure II.1 shows a metal cap on the CPU chip that helps to dissipate heat without requiring a heat sink.

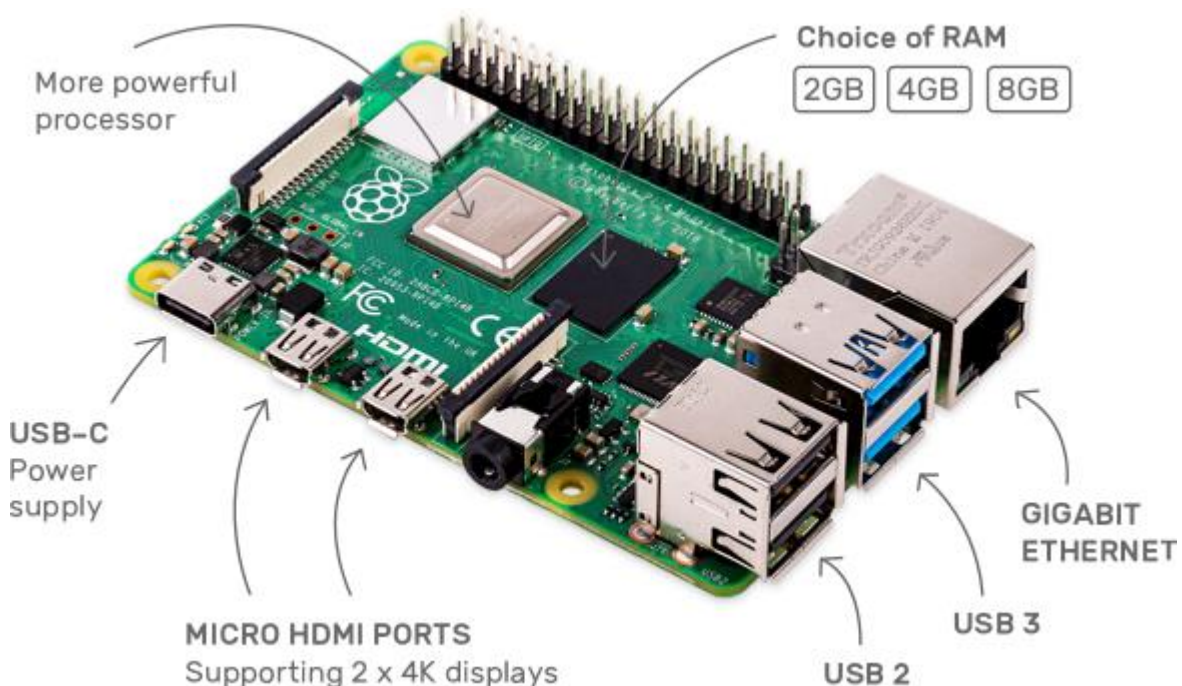


Figure II.1: The Raspberry Pi 4b board

## II.2.1 General Purpose Input Output (GPIO)

On the side of the Raspberry Pi, there are two rows of pins. The GPIO connector is the name for these pins. The Raspberry Pi's GPIO connector allows users to connect electronic hardware to it. All of the pins labeled GPIO can be used as general-purpose input/output pins. It means they can be used as input or output pins.

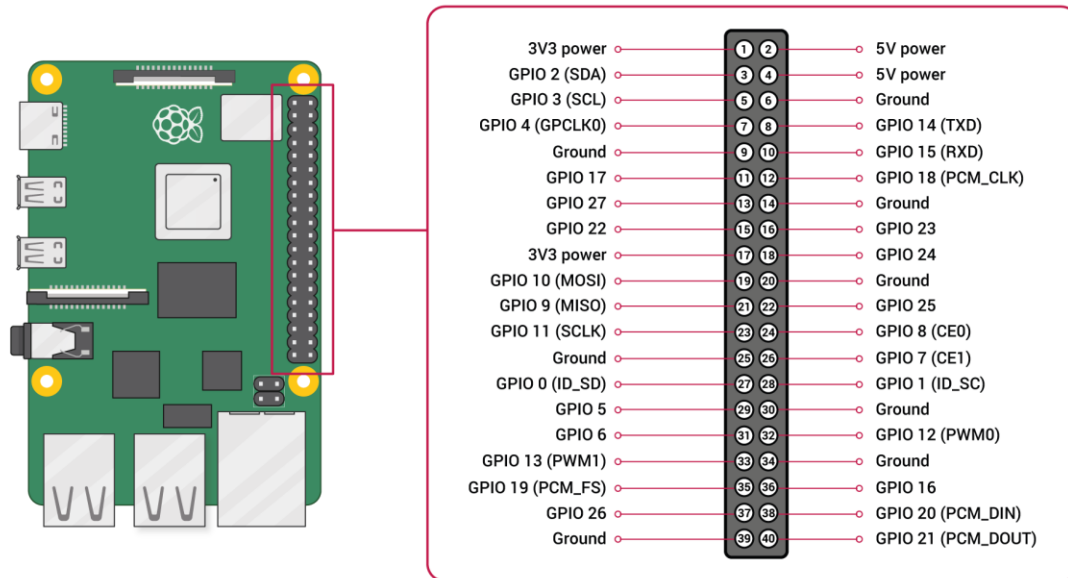


Figure II.2: GPIO connector's pin layout on Raspberry Pi 4b

The pins are numbered from the top left corner, with odd numbers on the left and even numbers on the right. Some of the pins on the GPIO connector have additional labels after the pin name. They are distinguishing marks for distinguishing characteristics. GPIO 2 and GPIO 3 are labeled SDA and SCL, respectively. The data and clock lines for a serial bus are represented by these pins. I2C is a serial bus that is commonly used to communicate with peripherals such as temperature sensors and LCDs.

## II.2.2 VNC server

This tool enables remote access to the Raspberry Pi's graphical desktop. All we need now is a VNC client on our accessing Smartphone (or computer) once the Raspberry Pi's VNC server is configured. A keyboard, mouse, and HDMI display device connected to the Raspberry Pi are no longer required. To put it another way, we could run the Pi without a display.

### II.2.2.1 Configure the VNC server

To begin, we must ensure that our Raspberry Pi 4b is connected to the Internet using a LAN or WiFi network. To set up a VNC server on a Raspberry Pi we need to follow these steps:

#### II.2.2.1.1 Install VNC server

To set up the VNC server on the Raspberry Pi 4b, we type the following commands into the Pi Terminal:

```
$sudo apt-get update
```

```
$sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer
```

### II.2.2.1.2 Enable VNC server

We can enable the VNC server by typing the following command on the terminal:

```
$ sudo raspi-config
```

A menu (Figure II.3) should appear now, from there we could enable the VNC server.

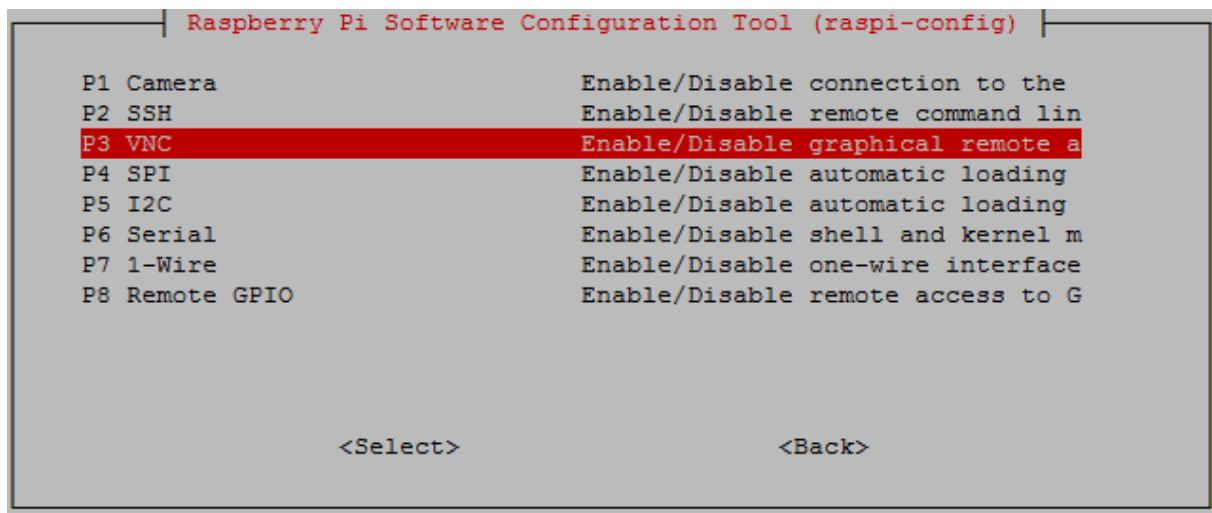


Figure II.3: Enable VNC server

## II.3 Temperature sensor (MLX90614)

The MLX90614 is an infrared thermometer that measures temperature without touching the object. It can measure two different temperatures. Temperatures of the ambient (environmental) and object (desired). The signal conditioning ASIC and the IR-sensitive thermopile detector chip are both housed in the same TO-39 can. A low noise amplifier, 17-bit ADC, and powerful DSP unit are all built into the MLX90614 thermometer, resulting in high accuracy and resolution.

The thermometer is factory calibrated with a digital SMBus output that allows full access to the measured temperature across the entire temperature range(s) with a 0.02°C resolution.

The user can configure the digital output to be pulse width modulation (PWM). The 10-bit PWM is set up to continuously transmit the measured temperature in the range of -20 to 120°C, with an output resolution of 0.14°C, as standard. It's compact and easy to connect to the microcontroller.



Figure II.4: MLX90614 IR temperature sensor

### II.3.1 Characteristics

- Low cost and small size
- It is simple to integrate
- Temperature ranges from  $-40$  to  $125^{\circ}\text{C}$  for the sensor and  $-70$  to  $380^{\circ}\text{C}$  for the object are factory calibrated
- Voltage Range:  $3.6\text{V}$  to  $5\text{V}$
- Supply Current:  $1.5\text{mA}$
- Field of View:  $80^{\circ}$
- Measurement distance:  $2\text{cm}$ - $5\text{cm}$

### II.3.2 Pinout Configuration

- VDD: It can be used to power the sensor, which is usually done with  $5\text{V}$
- GND: Metal can also serve as a ground
- SDA: It is an I2C communication serial data pin
- SCL: It is an I2C Communication Serial Clock Pin

### II.3.3 Working Principle

As previously stated, the MLX90614 sensor can measure an object's temperature without making physical contact with it. This is made possible by the Stefan-Boltzmann Law, which states that all objects and living beings emit infrared energy, the intensity of which is directly proportional to the temperature of the object or living being. As a result, the MLX90614 sensor calculates an object's temperature by measuring the amount of IR energy it emits.

## II.4 Raspberry Pi Camera Module v1.3

The Raspberry Pi camera has a resolution of 5 megapixels and is compatible with the Raspberry Pi 2B, 3, 3B, 3B+, and 4 models. It's also compatible with the Zero and Zero W models (with the correct ribbon cable). A Python library for the picamera allows users to control the camera, take photos, and record video. Users can also control the white balance, shutter, and read each RGB pixel directly from the camera using the library.

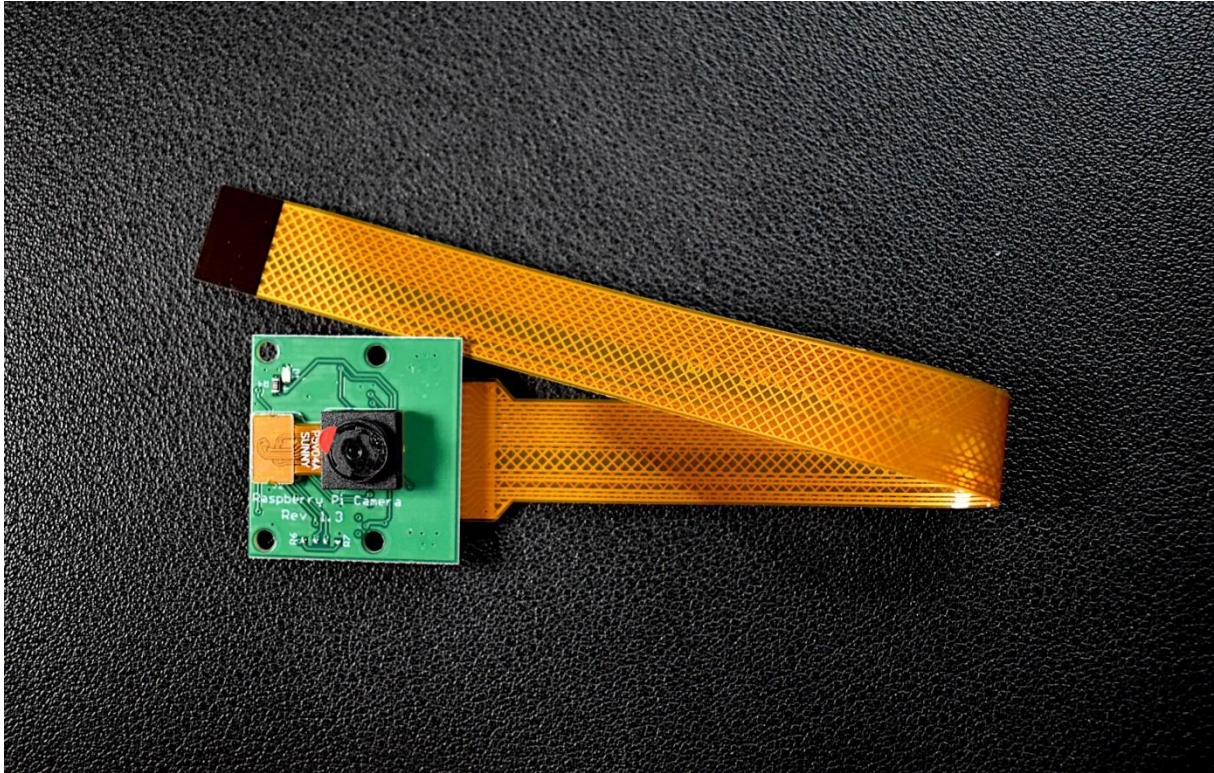


Figure II.5: Raspberry Pi Camera Module v1.3

#### II.4.1 Characteristics

- 5MP resolution
- Max video resolution: 1080p @ 30fps
- Camera Module PCB dimensions: 25mm x 24mm (9mm thickness)

### II.5 Popular face recognition algorithms

#### II.5.1 Histogram of oriented gradients (HOG)

##### II.5.1.1 Definition

HOG, or Histogram of Oriented Gradients, is a feature descriptor for extracting features from image data. It is widely used in object detection tasks in computer vision.

The Histogram of Oriented Gradients method is primarily used to detect faces and classify images. From autonomous vehicles to surveillance techniques to smarter advertising, this field has a wide range of applications.

##### II.5.1.2 How HOG works

1. Resize and color normalize the image as part of the preprocessing.
2. Calculate each pixel's gradient vector, as well as its magnitude and direction.
3. Make a grid of 8x8 pixel cells out of the image. The magnitude values of these 64 cells are binned and added to 9 buckets of unsigned direction in each cell (no sign, so 0-180 degree rather than 0-360 degree; this is a practical choice based on empirical experiments). If the direction of a pixel's gradient vector lies between two buckets for

better robustness, the magnitude of the pixel's gradient vector is proportionally split between the two.

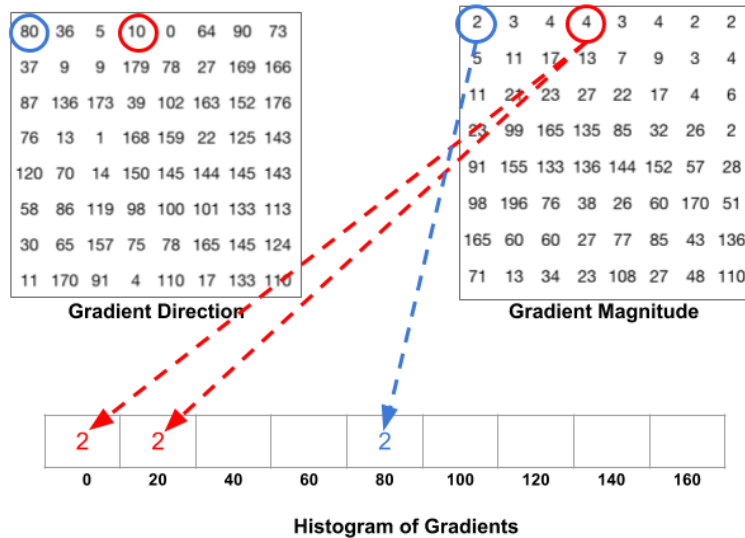


Figure II.6: How to split one gradient vector's magnitude if its degrees is between two degrees bins

- Then, across the image, we slide a 2x2 cell (thus 16x16 pixel) block. 4 histograms of 4 cells are concatenated into a one-dimensional vector of 36 values in each block region, and then normalized to have unit weight. The concatenation of all the block vectors yields the final HOG feature vector. For learning object recognition tasks, it can be fed into a classifier like SVM.

## II.5.2 HaarCascade Classifiers

Paul Viola and Michael Jones proposed an effective object detection method using Haar feature-based cascade classifiers in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It's a machine-learning approach in which a cascade function is trained using a large number of positive and negative images. After that, it's used to find objects in other images.

## II.5.3 Convolution Neural Network (CNN)

### II.5.3.1 Definition

A convolutional neural network (CNN) is a type of feed-forward artificial neural network used in deep learning. It is a supervised multilayer perceptron of a particular type.

Convolutional neural networks (CNNs) are a type of deep model inspired by how the human brain processes information. Each neuron in the visual cortex of the brain has a receptive field that captures data from a specific local neighborhood in visual space. They're made to recognize multi-dimensional data with a high degree of invariance to distortion and shift scaling.

### II.5.3.2 Convolution Neural Network architecture

One input layer, multiple hidden layer types, and one output layer make up the CNN architecture. Convolution is handled by the first type of hidden layer, while local averaging,

subsampling, and resolution reduction are handled by the second type. The third hidden layers act as a traditional multi-layer perceptron classifier.

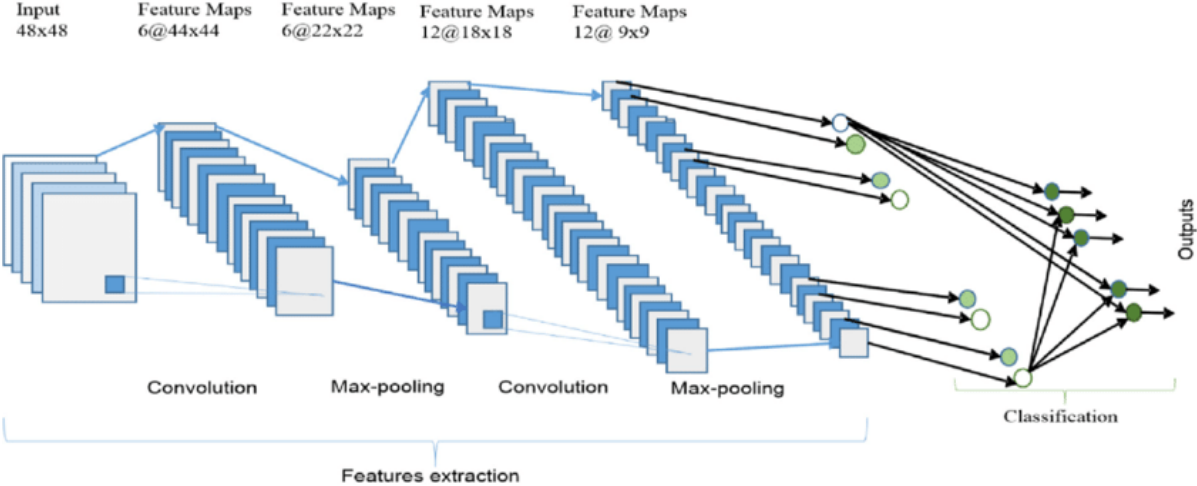


Figure II.7: The overall architecture of CNN includes an input layer, multiple alternating convolutions, and max-pooling layers, one fully connected layer, and one classification layer

**II.5.3.3 Convolution layer**

The first layer to extract features from an input image is convolution. By learning image features with small squares of input data, convolution preserves the relationship between pixels. It's a mathematical operation with two inputs: an image matrix and a filter or kernel.

$$\begin{matrix} h \\ \text{Image Matrix} \\ w \end{matrix} * \begin{matrix} f_h \\ \text{Filter} \\ f_w \\ d \end{matrix} = \begin{matrix} h - f_h + 1 \\ \text{Output Matrix} \\ w - f_w + 1 \end{matrix}$$

Figure II.8: Image matrix multiplies kernel or filter matrix

- An image matrix (volume) of dimension  $(h \times w \times d)$
- A filter  $(f_h \times f_w \times d)$
- Outputs a volume dimension  $(h - f_h + 1) \times (w - f_w + 1) \times 1$

Consider a 6 x 6 image matrix and filter matrix 3 x 3, Then the convolution of 6 x 6 image matrix multiplies with 3 x 3 filter matrix which is called “Feature Map” as output shown in below:



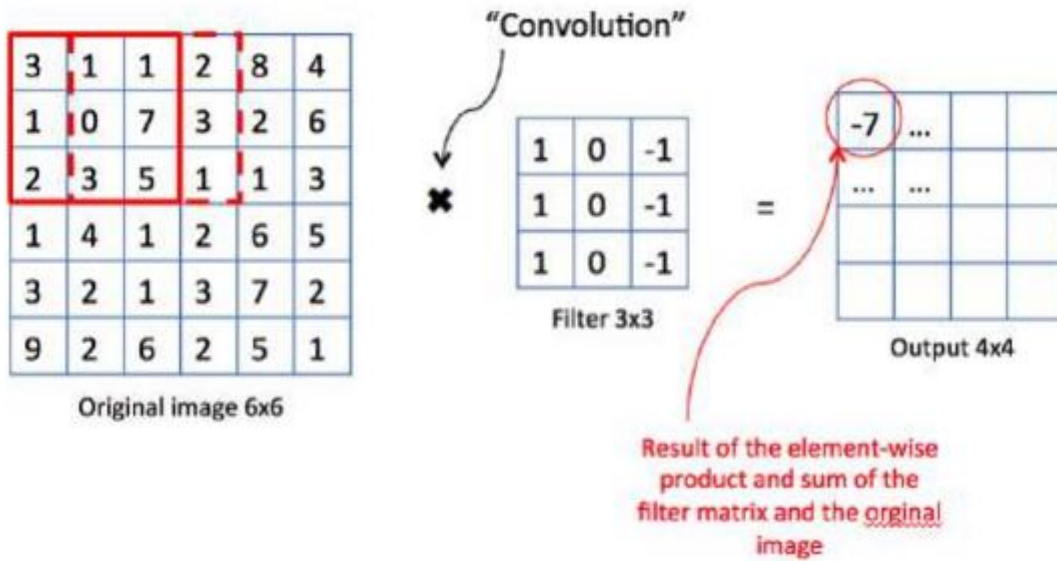


Figure II.9: 4 x 4 Output matrix

For example, if we are going to compare these 2 images:

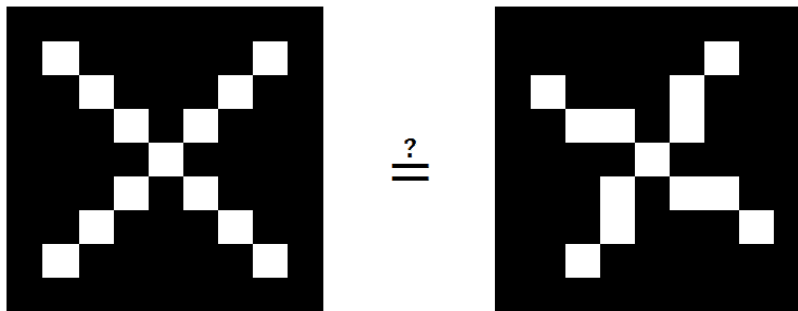


Figure II.10: Two images to compare

These 2 images consist of 81 pixels ( $9 \times 9$ ). By coding the color of the pixels by a value between -1 (black pixel) and +1 (white pixel), we get the following images:

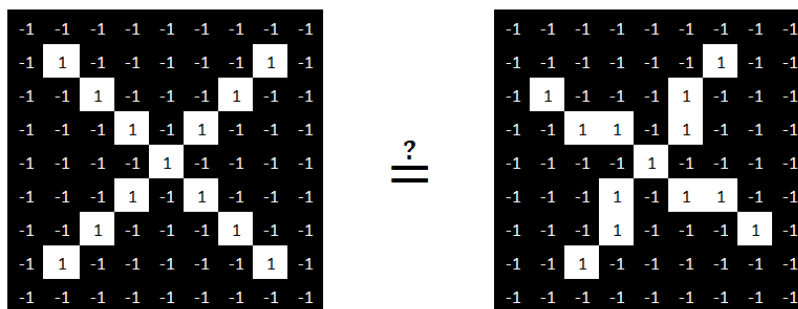


Figure II.11: The 2 images with coded pixels

By comparing pixel by pixel, we see that there are different pixels between the 2 images:

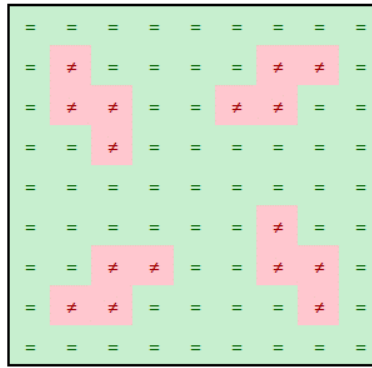


Figure II.12: pixel by pixel comparison

However, some pieces of the image are identical between the 2 images:

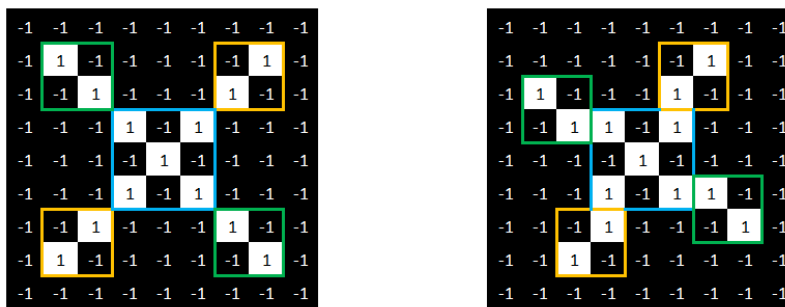


Figure II.13: some pieces of the image are identical between the 2 images

These pieces are 3 in number, these pieces are called features. Now consider features of size  $3 \times 3$ , we will see how the machine can find them by means of very simple calculations.

Here are the new features that we want the machine to find:

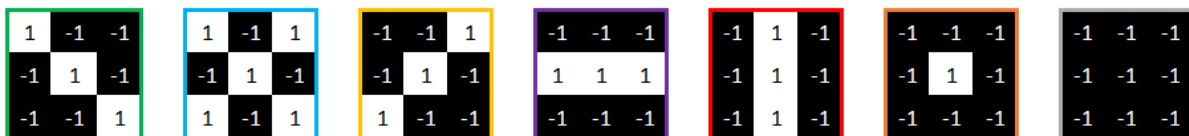


Figure II.14: features that we want the machine to find

We will start by looking for this feature on this image:

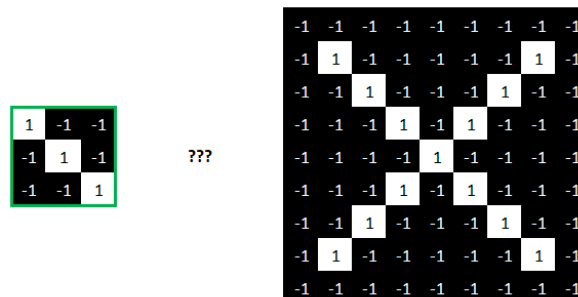


Figure II.15: We will start by looking for this feature on this image

To do this, we will compare this feature by "dragging" it onto the image as follows:

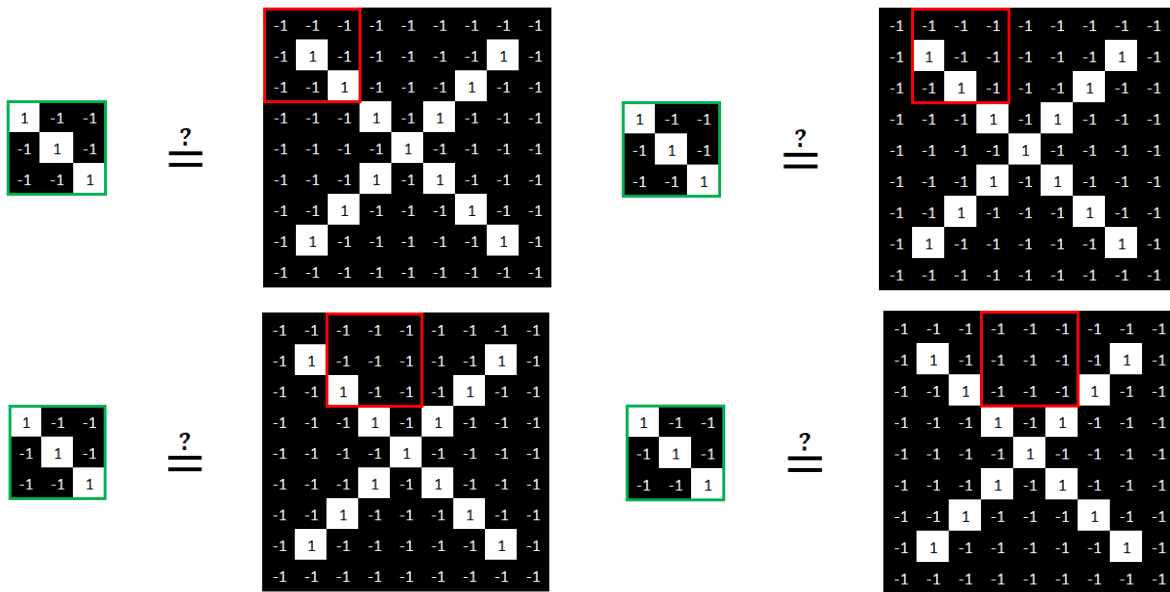


Figure II.16: feature dragging

Concretely, to compare this feature with part of the image, we will:

1) Multiply the 9 pixel values of the feature with the 9 pixel values of the piece of the image to find.

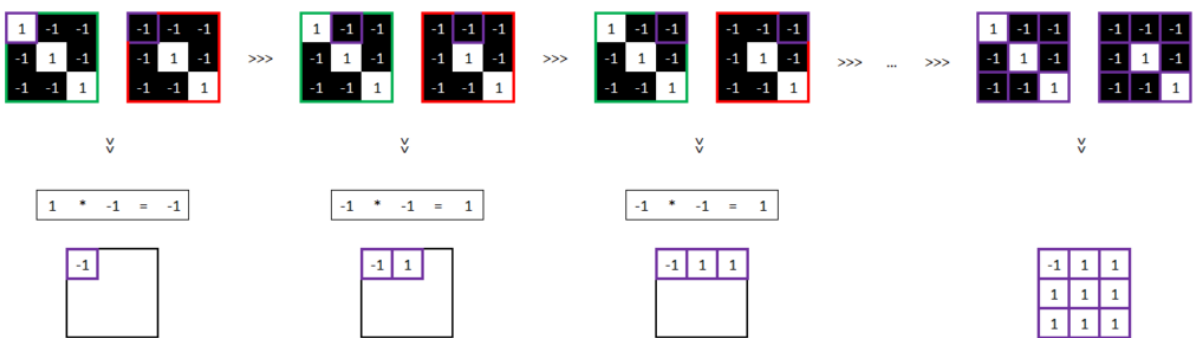


Figure II.17: Multiply the pixel values of the feature with the pixel values of the piece of the image to find

2) Add these 9 results.

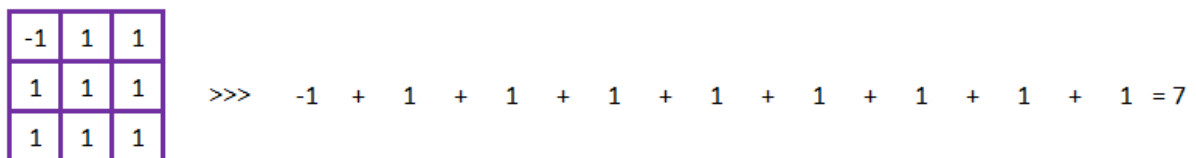


Figure II.18: add the results

3) Divide by the number of pixels (here 9):  $7 / 9 = 0.78$

4) Conclude (if the result is equal to 1, then the feature has been identified in the image): 0.78 is different from 1, so the characteristic was not found in this part of the image.

### II.5.3.4 What is Weight

Within a neural network, the weight parameter transforms input data within the network's hidden layers. A neural network is made up of a series of nodes, also known as neurons. A set of inputs, a weight, and a bias value are all contained within each node. When an input is fed into a node, it is multiplied by a weight value, and the result is either observed or passed on to the next layer of the neural network. The weights of a neural network are frequently stored in the network's hidden layers.

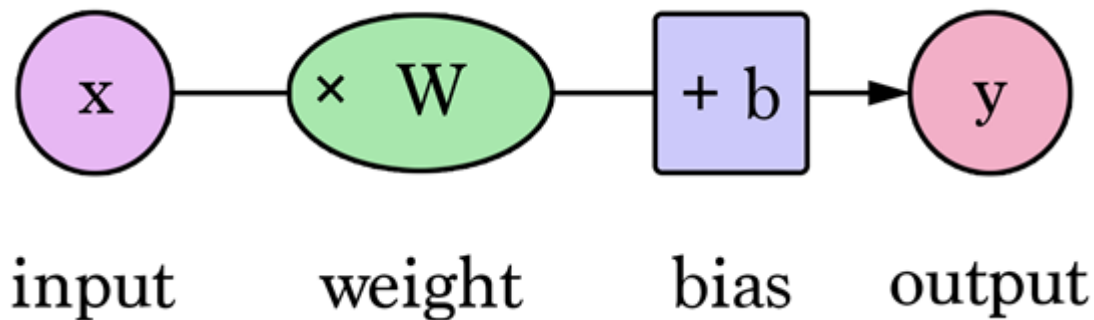


Figure II.19: a neural network with one input and one output

To understand how weights work, it's helpful to imagine a theoretical neural network. An input layer in a neural network receives input signals and passes them on to the next layer.

The neural network then has a series of hidden layers that apply transformations to the data input. The weights are applied to the nodes of the hidden layers' nodes. For example, before passing the data to the next layer, a single node may multiply the input data by an assigned weight value, then add a bias. The output layer is also known as the final layer of the neural network. The output layer frequently tunes the hidden layers' inputs to produce the desired numbers within a given range.

### II.5.3.5 Activation functions

The activation function in a neural network is responsible for converting the node's summed weighted input into the node's activation or output for that input.

Activation functions that are commonly used include:

- Sigmoid function:  $\frac{1}{1 + e^{-x}}$
- Tanh function:  $\tanh(x)$
- Leaky ReLU function:  $\max(0.1x, x)$
- ReLU function:  $\max(0, x)$
- Maxout function:  $\max(w_1^T x + b_1, w_2^T x + b_2)$

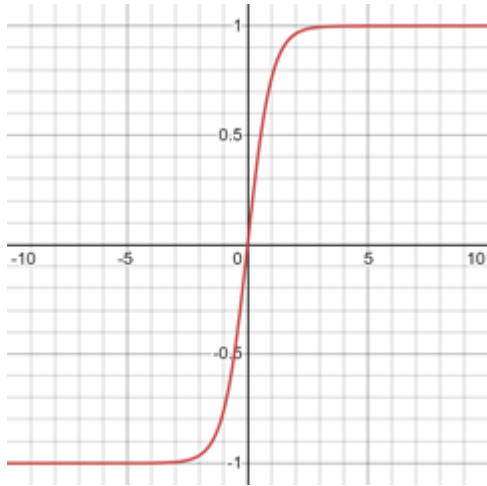


Figure II.20: tanh

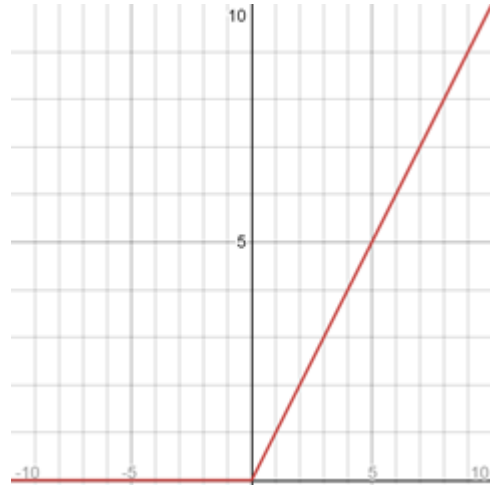


Figure II.21: ReLU function

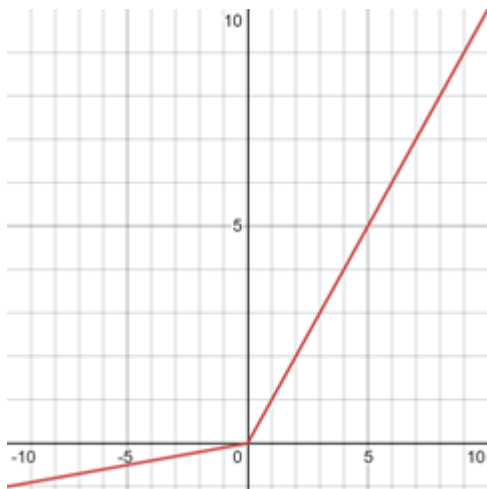


Figure II.22: Leaky ReLU function

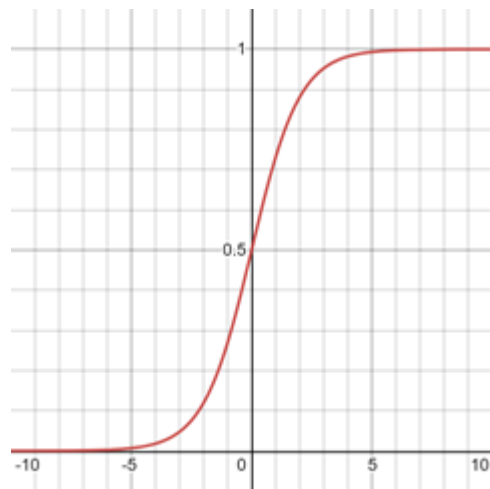


Figure II.23: Sigmoid function

### II.5.3.6 RELU Layer

ReLU stands for Rectified Linear Unit for a non-linear operation. In deep learning models, the Rectified Linear Unit is the most commonly used activation function. If the function receives any negative input, it returns 0; however, if the function receives any positive value  $x$ , it returns that value. As a result, it can be written,  $R(z)=\max(0,z)$  Graphically, it looks like this:

$$y = \text{activation}(\sum(\text{weight} * x) + \text{bias}) \quad (1)$$

### II.5.3.7 Pooling layer

The downsampling operation will be performed by the pooling layer along the spatial dimensions (width, height). Pooling is a simple operation that involves replacing a pixel square (typically  $2*2$  or  $3*3$ ) with a single value. There are three types of pooling to consider:

1. Average pooling: take the average value of the pixels in the selected square.
2. Max pooling: take the max value of the pixels in the selected square
3. Sum pooling: take the sum value of the pixels in the selected square

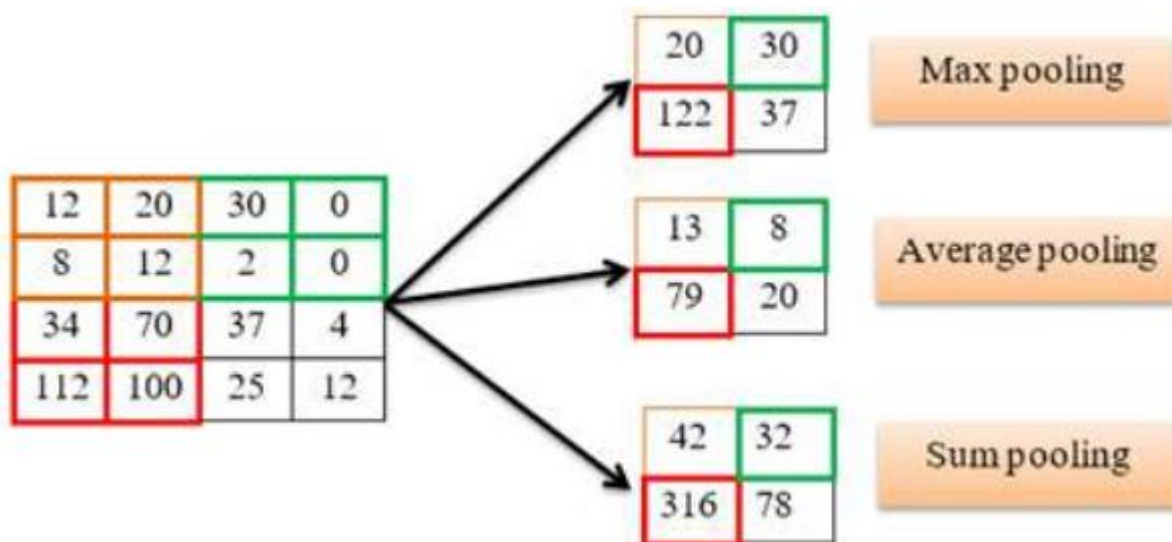


Figure II.24: Types of pooling max, average, and sum pooling

### II.5.3.8 Fully connected layer

The class scores will be computed by the fully connected layer (FC), resulting in a volume of size  $[1 \times 1 \times N]$ . Each of the  $N$  numbers represents a class score, for example, among the  $N$  categories.

### II.5.3.9 Cost function

Any function that outputs a scalar that quantifies the error of your neural network's performance qualifies as a cost function.

It's a function that determines how well a Machine Learning model performs for a given set of data. The Cost Function calculates the difference between predicted and expected values and presents it as a single real number. Cost Functions can be created in a variety of ways depending on the problem. The goal of a Cost Function is to be one of two things:

- **Minimized:** The returned value is commonly referred to as a cost, loss, or error. The goal is to find model parameter values for which the Cost Function returns the smallest number possible.
- **Maximized:** The value it generates is then referred to as a reward. The goal is to find model parameter values with the largest possible return number.

#### II.5.3.9.1 Cost function requirements

A cost function must satisfy two properties in order to be used in back-propagation:

- The cost function  $C$  must be able to be written as an average:  $j = \frac{1}{n} \sum_x C_x$
- The cost function  $C$  must not be dependent on any activation values of a neural network besides the output values.

### II.5.3.9.2 List of cost functions used in Neural Networks

- Regression:  $j = \frac{1}{2} \sum_j (\hat{y}_j - y_j^{pred})^2$
- Classification:  $j = -\sum_j (\hat{y}_j \log(y_j) + (1 - \hat{y}_j) \log(1 - y_j))$
- Exponential Cost:  $j = \tau \exp[\frac{1}{\tau} \sum_j (\hat{y}_j - y_j^{pred})^2]$
- Itakura-Saito Distance:  $j = \sum_j (\frac{\hat{y}_j}{y_j^{pred}} - \log \frac{\hat{y}_j}{y_j^{pred}} - 1)$
- Kullback-Leibler Divergence:  $j = \sum_j \hat{y}_j \log \frac{\hat{y}_j}{y_j^{pred}}$
- Generalized Kullback-Leibler Divergence:  $j = \sum_j \hat{y}_j \log \frac{\hat{y}_j}{y_j^{pred}} - \sum_j \hat{y}_j - \sum_j y_j^{pred}$

### II.5.3.10 Back propagation

The back propagation algorithm will calculate the partial derivatives of the cost once it has been measured. The filters in the different layers of a CNN are the most interesting parameters.

### II.5.3.11 Gradient Descent

Gradient Descent is used to adjust the weights by finding the minimum of the cost function (which is used to calculate the output error).

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (2)$$

### II.5.3.12 Stochastic gradient Descent

The stochastic gradient descent algorithm however has been shown to be faster, more reliable, and less prone to reach bad local minima than standard gradient descent.

A stochastic gradient descent algorithm:

$$w_{t+1} = w_t \Delta j(z, w_t) \quad (3)$$

## II.6 What is the best face recognition algorithm

The HoG algorithm appears to be the fastest, followed by the Haar Cascade classifier and CNNs. However, CNNs are the most accurate algorithm in Dlib. HoG performs admirably but has difficulty recognizing small faces. Overall, HaarCascade Classifiers are comparable to HoG.

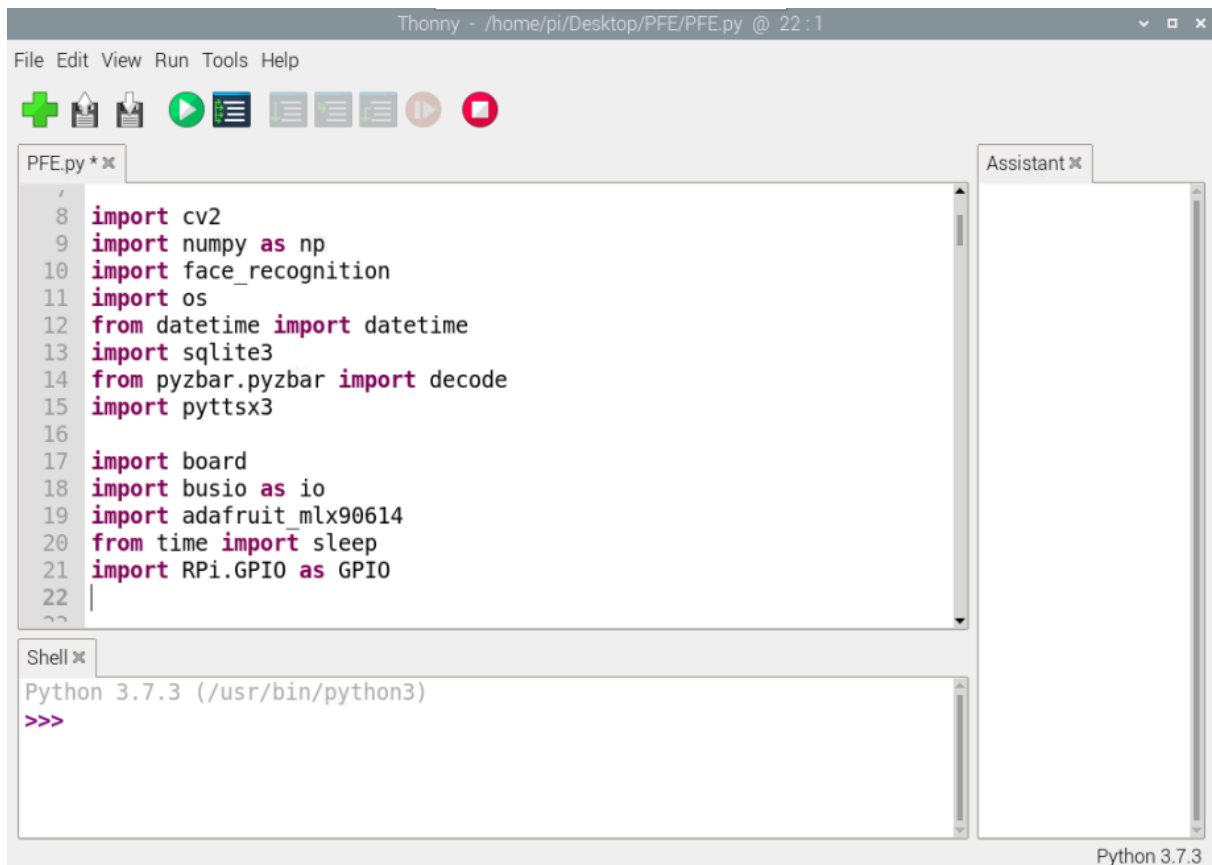
## II.7 Programming

### II.7.1 Libraries

In our project we used the following python libraries:

- **OpenCV:** it is a set of Python bindings for solving computer vision problems.
- **NumPy:** it is a Python library that allows us to work with arrays. It also has functions for working with matrices, Fourier transforms, and linear algebra.
- **face\_recognition:** it is built with dlib's cutting-edge face recognition technology. Deep learning was used to create this library. On the Labeled Faces in the Wild benchmark, the model has a 99.38% accuracy rate.
- **os:** In Python, the OS module has functions for interacting with the operating system.
- **datetime:** Date and time are not data types in Python; however, a module called datetime can be imported to work with both the date and the time.
- **sqlite3:** SQLite is a C library that provides a lightweight disk-based database that does not require a separate server process and can be accessed using a nonstandard SQL query language variant.
- **pyzbar:** It's a Python library that uses the zbar library to read one-dimensional barcodes and QR codes from a variety of sources, including video streams, image files, and raw intensity sensors.
- **pyttsx3:** it is a Python-based text-to-speech conversion library. It works offline, unlike other libraries.
- **board:** Common container for board base pin names.
- **busio:** Classes in the busio module support a variety of serial protocols.
- **adafruit\_mlx90614:** Python module for the MLX90614 IR object temperature sensor.
- **time:** The Python time module offers a variety of methods for representing time in code.
- **RPi.GPIO:** It's a Python module for controlling the Raspberry Pi's GPIO interface.





The screenshot shows the Thonny Python IDE interface. The main editor window displays a Python script named 'PFE.py' with the following code:

```
8 import cv2
9 import numpy as np
10 import face_recognition
11 import os
12 from datetime import datetime
13 import sqlite3
14 from pyzbar.pyzbar import decode
15 import pyttsx3
16
17 import board
18 import busio as io
19 import adafruit_mlx90614
20 from time import sleep
21 import RPi.GPIO as GPIO
22 |
```

Below the editor is a shell window titled 'Shell' showing the Python 3.7.3 prompt 'Python 3.7.3 (/usr/bin/python3)' and the prompt '>>>>'. To the right of the editor is an 'Assistant' panel which is currently empty. The bottom right corner of the IDE window displays 'Python 3.7.3'.

Figure II.25: Libraries used in the program

## II.7.2 Initialization

After importing the different python libraries that we need in our program we initialize the different functions:

- Initialize the text to speech engine
- Set the GPIOs pins used
- Configure I2C

The screenshot shows the Thonny Python IDE interface. The title bar reads 'Thonny - /home/pi/Desktop/PFE/PFE.py @ 31 : 1'. The menu bar includes 'File Edit View Run Tools Help'. Below the menu bar is a toolbar with icons for file operations and execution. The main editor window displays a Python script named 'PFE.py' with the following code:

```
23
24 #text to speech
25 engine = pyttsx3.init()
26 engine.setProperty('voice', 'english_rp+f4')
27 engine.setProperty('rate', 150)
28 engine.say("HELLO")
29
30 GPIO.setwarnings(False)
31 GPIO.setmode(GPIO.BOARD)
32 GPIO.setup(20, GPIO.OUT, initial = GPIO.LOW)
33 GPIO.setup(21, GPIO.OUT, initial = GPIO.LOW)
34
35 i2c = io.I2C(board.SCL, board.SDA, frequency=100000)
36 mlx = adafruit_mlx90614.MLX90614(i2c)
37
38
```

Below the editor is a 'Shell' window showing the Python 3.7.3 prompt: 'Python 3.7.3 (/usr/bin/python3) >>>'. To the right of the editor is an 'Assistant' window, which is currently empty. The bottom right corner of the IDE shows 'Python 3.7.3'.

Figure II.26: Initialization

## II.7.3 Create an SQL database

### II.7.3.1 What is SQL

SQL stands for Structured Query Language, and it is a computer language used to store, manipulate, and retrieve data from a relational database. The Relational Database System (RDBMS) standard language is SQL.

### II.7.3.2 What is SQLite

SQLite is a relational database management system implemented as a software library. In terms of setup, database administration, and required resources, the lite in SQLite stands for "lightweight."

The following are some of SQLite's notable features: transactional, self-contained, serverless, zero-configuration

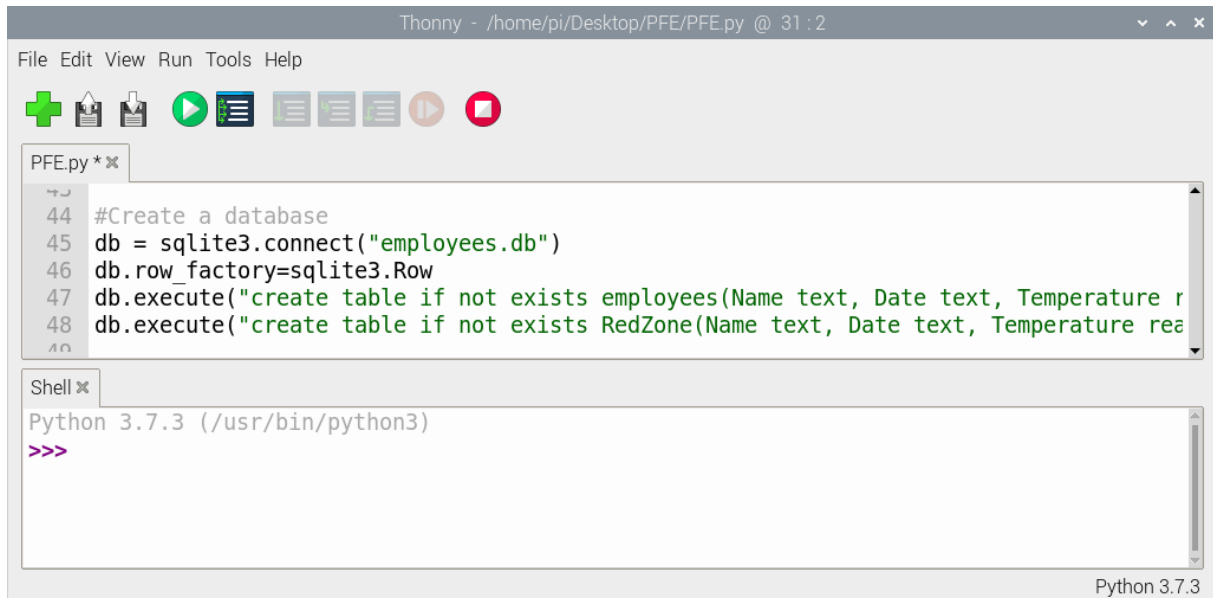
### II.7.3.3 What is the difference between SQL and SQLite?

SQLite is a database system, but it's not a full-fledged database; rather, it's a lightweight database designed for storing small amounts of data locally. It's primarily used for desktop and mobile applications that don't require access to the internet. SQLite uses files and all SQL requests open a file at each call, SQLite is light on the system, and thus is suited more to database and mobile applications, whereas a full fledged RDBMS would require a dedicated service running as long as the application is running to maintain integrity, and they are suitable for large amounts of data. SQLite uses files and all SQL requests open a file at each

call, SQLite is light on the system, and thus is suited more to database and mobile applications.

### II.7.3.4 The database creation code

To create our SQL database we execute the following code



The screenshot shows the Thonny IDE interface. The main editor window displays the following Python code in PFE.py:

```
44 #Create a database
45 db = sqlite3.connect("employees.db")
46 db.row_factory=sqlite3.Row
47 db.execute("create table if not exists employees(Name text, Date text, Temperature r
48 db.execute("create table if not exists RedZone(Name text, Date text, Temperature rea
```

Below the editor is a Shell window with the following text:

```
Python 3.7.3 (/usr/bin/python3)
>>>
```

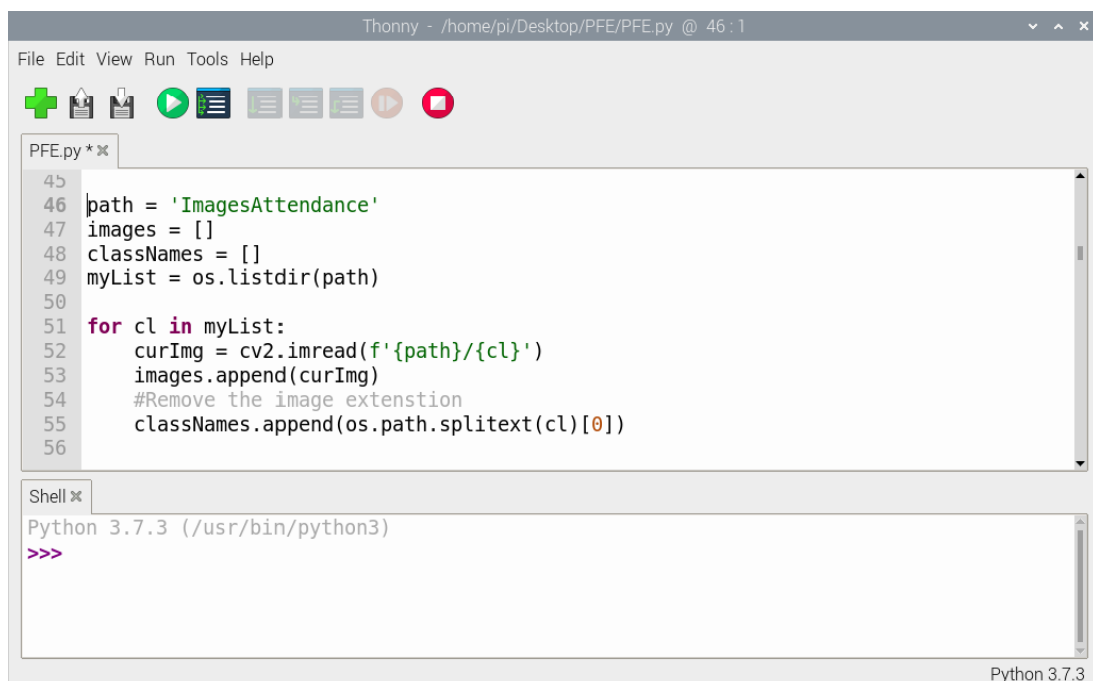
The status bar at the bottom right indicates Python 3.7.3.

Figure II.27: The database creation code

## II.7.4 Face Recognition

### II.7.4.1 Load Images

Now we load the pictures by running this code:



The screenshot shows the Thonny IDE interface. The main editor window displays the following Python code in PFE.py:

```
45
46 path = 'ImagesAttendance'
47 images = []
48 classNames = []
49 myList = os.listdir(path)
50
51 for cl in myList:
52     curImg = cv2.imread(f'{path}/{cl}')
53     images.append(curImg)
54     #Remove the image extension
55     classNames.append(os.path.splitext(cl)[0])
56
```

Below the editor is a Shell window with the following text:

```
Python 3.7.3 (/usr/bin/python3)
>>>
```

The status bar at the bottom right indicates Python 3.7.3.

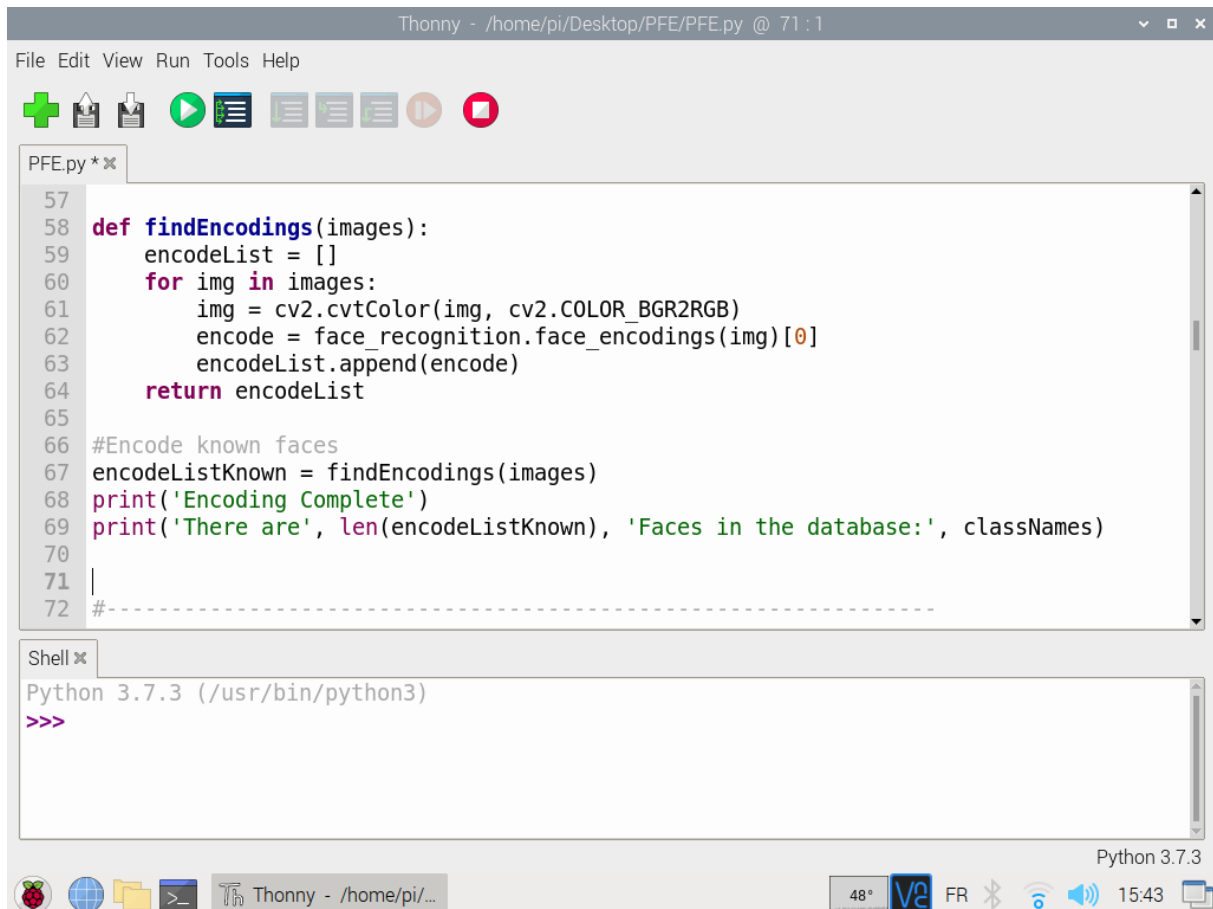
Figure II.28: Load Images

### II.7.4.2 Encoding and identifying the faces

For us, it's a face. But it's just an array of RGB values for our algorithm, which matches a pattern it's learned from the data samples we gave it.

The algorithm takes note of certain important measurements on the face, such as the color, size, and slant of the eyes, the gap between the brows, and so on, for face recognition. All of these factors combine to form the face encoding — the information extracted from the image — that is used to identify a specific face.

Let's take a look at the encodings code:



The screenshot shows the Thonny Python IDE interface. The main window displays a Python script named 'PFE.py' with the following code:

```
57
58 def findEncodings(images):
59     encodeList = []
60     for img in images:
61         img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
62         encode = face_recognition.face_encodings(img)[0]
63         encodeList.append(encode)
64     return encodeList
65
66 #Encode known faces
67 encodeListKnown = findEncodings(images)
68 print('Encoding Complete')
69 print('There are', len(encodeListKnown), 'Faces in the database:', classNames)
70
71 |
72 #-----
```

Below the script editor is a shell window titled 'Shell' with the following content:

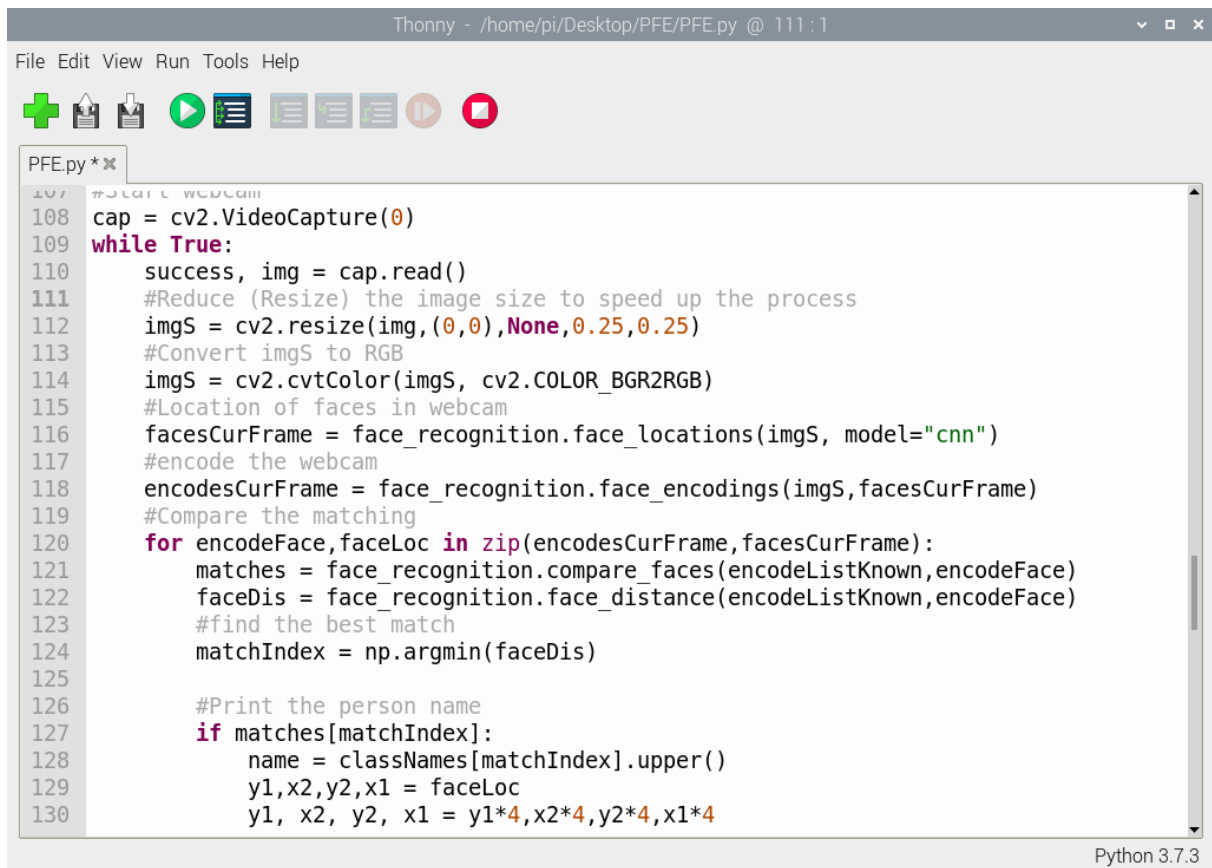
```
Python 3.7.3 (/usr/bin/python3)
>>>
```

The IDE's status bar at the bottom shows 'Python 3.7.3' and system icons including a Raspberry Pi logo, a globe, a folder, a terminal icon, the text 'Thonny - /home/pi/...', a temperature indicator '48°', a VR logo, 'FR', a Bluetooth icon, a Wi-Fi icon, a speaker icon, and the time '15:43'.

Figure II.29: Encoding and identifying the faces

### II.7.4.3 Similarity

Let's move on to the next step, which is determining facial similarity. We'll need to read images from the Raspberry Pi camera for this.



```
107 #Start webcam
108 cap = cv2.VideoCapture(0)
109 while True:
110     success, img = cap.read()
111     #Reduce (Resize) the image size to speed up the process
112     imgS = cv2.resize(img,(0,0),None,0.25,0.25)
113     #Convert imgS to RGB
114     imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
115     #Location of faces in webcam
116     facesCurFrame = face_recognition.face_locations(imgS, model="cnn")
117     #encode the webcam
118     encodesCurFrame = face_recognition.face_encodings(imgS,facesCurFrame)
119     #Compare the matching
120     for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame):
121         matches = face_recognition.compare_faces(encodeListKnown,encodeFace)
122         faceDis = face_recognition.face_distance(encodeListKnown,encodeFace)
123         #find the best match
124         matchIndex = np.argmin(faceDis)
125
126         #Print the person name
127         if matches[matchIndex]:
128             name = classNames[matchIndex].upper()
129             y1,x2,y2,x1 = faceLoc
130             y1, x2, y2, x1 = y1*4,x2*4,y2*4,x1*4
```

Python 3.7.3

Figure II.30: facial similarity

Now, It's no longer difficult to spot the similarity. For this, the face recognition module provides a simple API.

```
matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
```

This method compares each component of the two faces and tells us whether the component in question varies within the tolerance limits.

### II.7.5 Checking the temperature

After recognizing the person we will now check his temperature, this is done by the following part of the program:

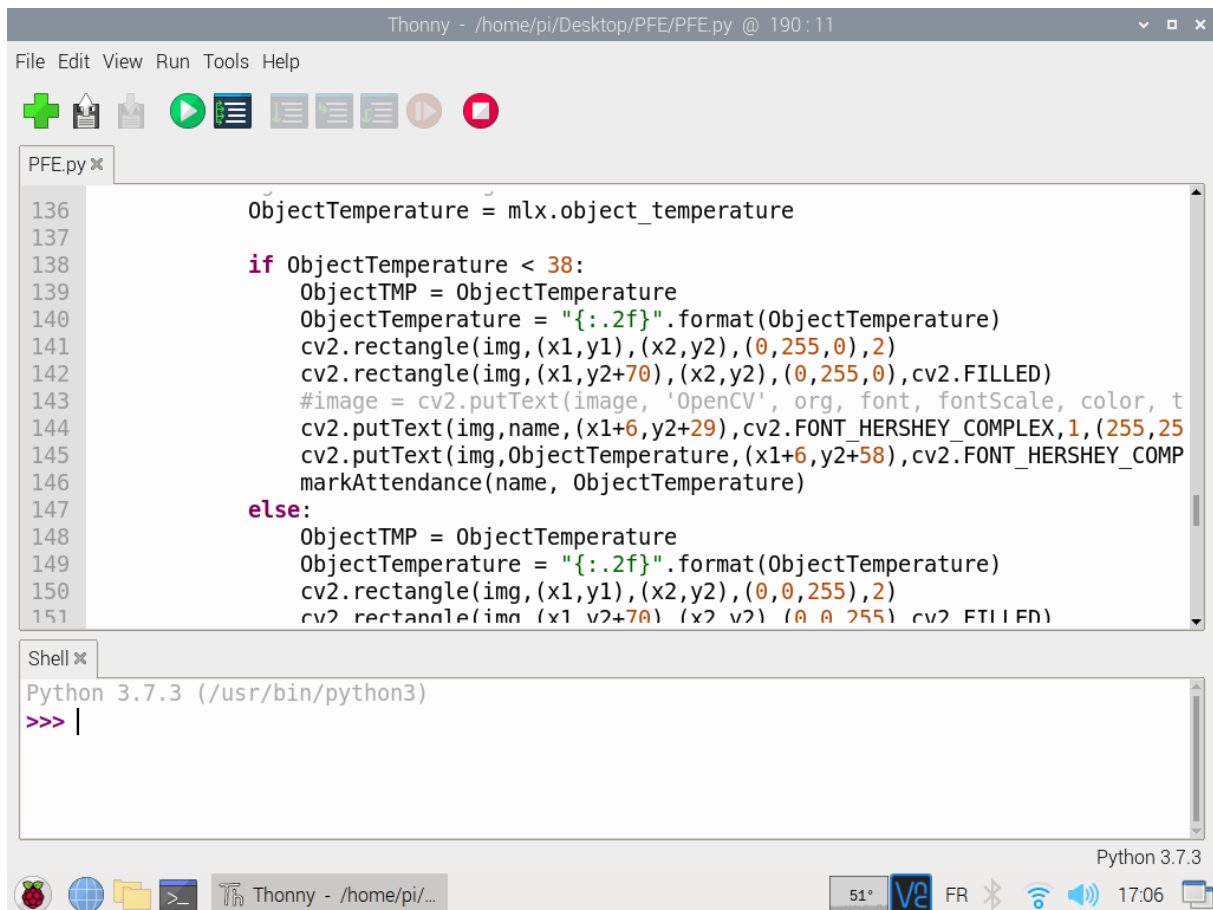


Figure II.31: Checking the temperature

If the temperature is normal (below 38°C) the following actions will be performed:

- Draw a green rectangle around the face
- Turn ON the green LED
- Turn OFF the red LED
- Say “normal temperature”
- Store the entry in the employees table in the SQL database

But, if the temperature is high (above 38°C):

- Draw a red rectangle around the face
- Turn ON the red LED
- Turn OFF the green LED
- Say “high temperature”
- Store the entry in employees and RedZone tables in the SQL database

### II.7.6 QR code

A person can be removed from the RedZone table if he provides a QR code of a negative COVID-19 test. the QR code must contain the following pieces of information:

- Person name
- Test result (positive or negative)

```
Thonny - /home/pi/Desktop/PFE.0/PFE - Copie.py @ 178 : 17
File Edit View Run Tools Help
+ [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
PFE - Copie.py * * *
154
155 barcodes = pyzbar.decode(img)
156 for barcode in decode(img):
157     #Convert the binary data into string
158     myDATA = barcode.data.decode('utf-8')
159     print(myDATA.splitlines())
160     QR = myDATA.splitlines()
161     pts = np.array([barcode.polygon], np.int32)
162     pts = pts.reshape((-1, 1, 2))
163     cv2.polylines(img, [pts], True, (255, 0, 255), 5)
164
165     if QR[1] == "NEGATIVE":
166         cursor19 = db.execute("select * from RedZone")
167         cursor19.execute("SELECT rowid FROM RedZone WHERE name = ? and name = ?", (QR[0], QR[0]))
168         COVIDtest = cursor19.fetchall()
169         if len(COVIDtest)==0:
170             pass
171         else:
172             print("YOU'RE NO LONGER IN THE DATABASE")
173             cursor19.execute("DELETE FROM RedZone WHERE name = ? and name = ?", (QR[0], QR[0]))
174             db.commit()
175
176
177
Python 3.7.3
```

Figure II.32: QR code

## II.8 Project circuit

Our project circuit consists of the following elements:

- Raspberry Pi 4
- MLX90614 IR temperature sensor
- 2 LEDs
- 2 Resistors
- CSI Camera module

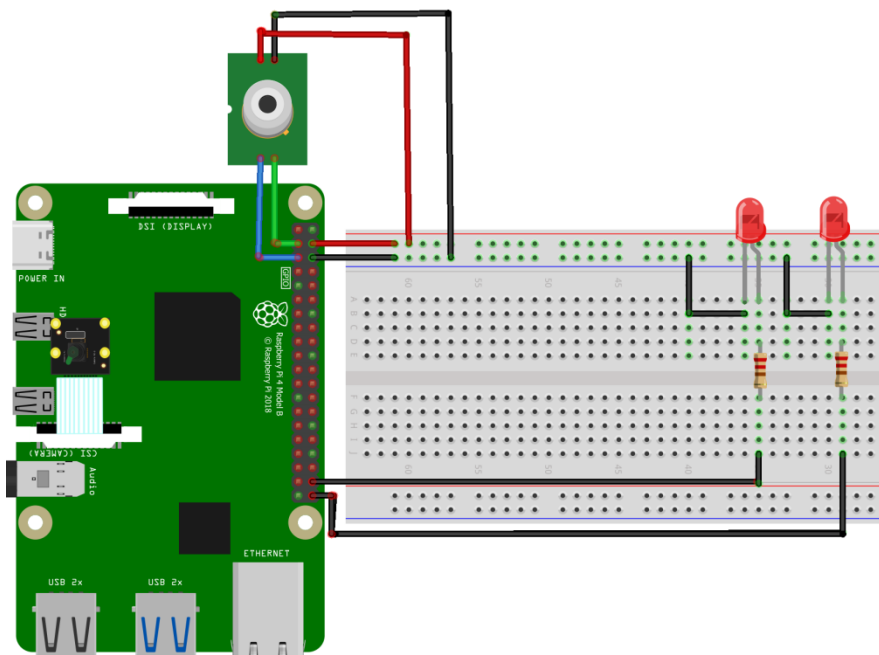


Figure II.33: Project circuit

## **II.9 Conclusion**

This chapter provided a brief overview of the hardware used in the development of our fever detector, as well as a discussion of popular face recognition algorithms and a detailed description of our program.



**CHAPTER 3**  
**SIMULATION AND FINAL RESULTS**

## III. SIMULATION AND FINAL RESULTS

### III.1 Introduction

After familiarizing ourselves with the Raspberry Pi hardware, MLX90614, Camera module, and facial recognition algorithms in the previous chapters, we devote this chapter to the steps to configure our Raspberry Pi and get the project up and running. First, we will start by installing the Raspbian operating system on an SD card, we will connect the Raspberry power supply and peripherals.

### III.2 Simulation of results

#### III.2.1 Facial recognition and temperature check

We will now test the script used to classify the real-time images and recognize faces and check the temperature. We will load the image and preprocess it for classification after importing the necessary packages and checking the temperature:

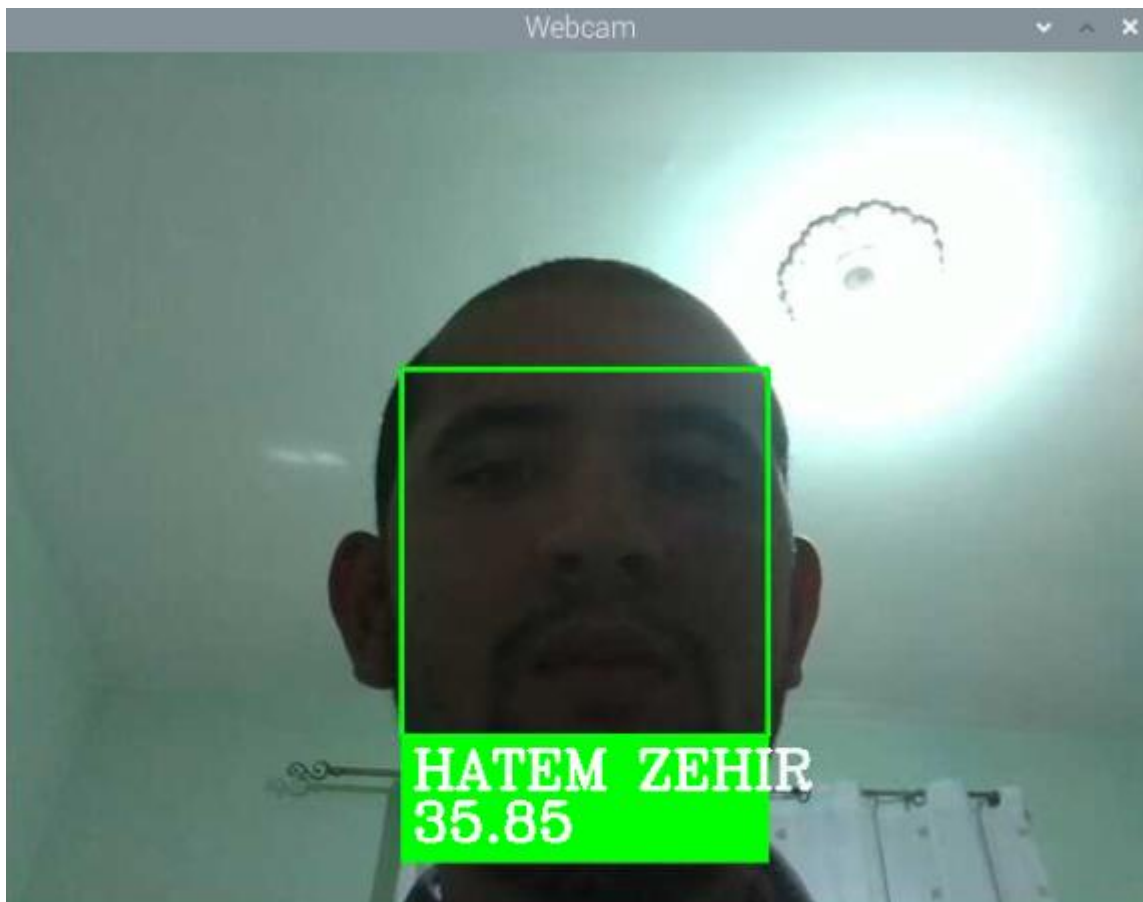


Figure III.1: Facial recognition and temperature check



Figure III.2: a red rectangle will be drawn around the face if the temperature is higher than 38°C

### III.2.2 Database

After checking the temperature and recognizing the faces we can store the data in an SQL database as follow:

- If the temperature is lower than 38°C the entry will be stored in the “employees” table in the database.
- If the temperature is higher than 38°C the entry will be stored in both “RedZone” and “employees” tables.

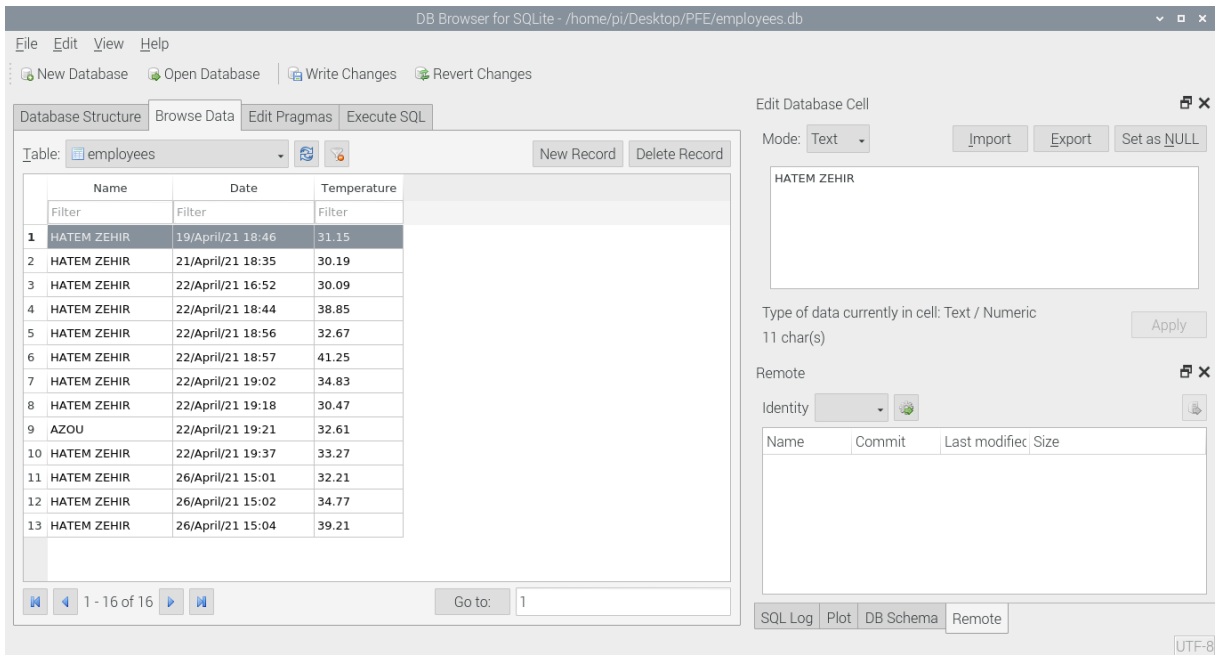


Figure III.3: employees table in the database

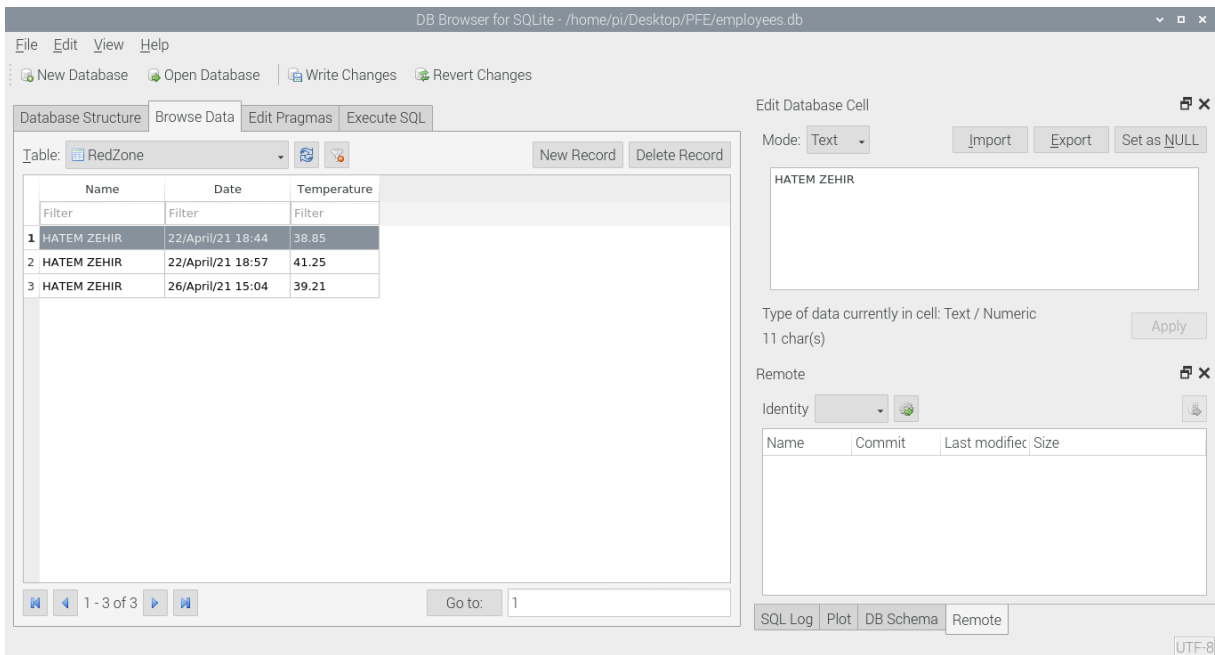


Figure III.4: RedZone table in the database

### III.2.3 LEDs activation

If we checked a person that has a normal temperature the green LED will turn on:

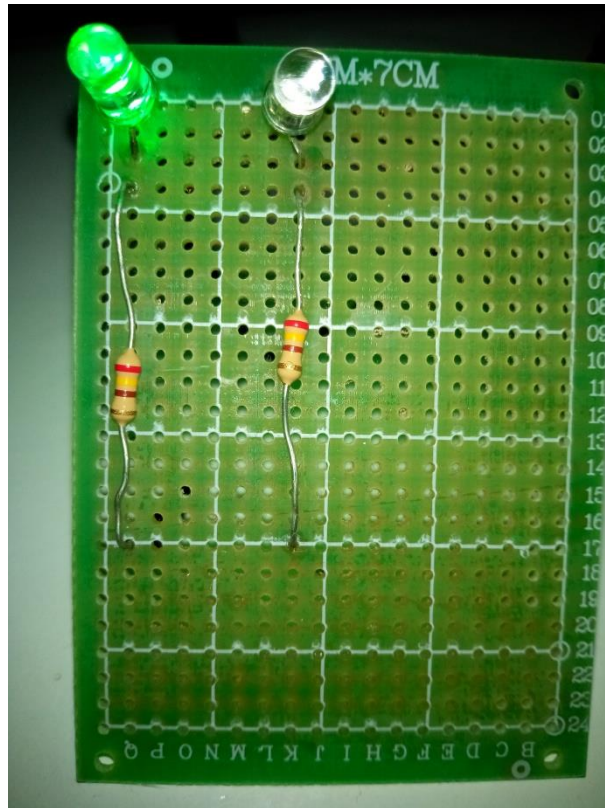


Figure III.5: The green LED will turn on if the temperature is lower than 38°C

If the person has a high temperature (higher than 38°C) the red LED will turn on:

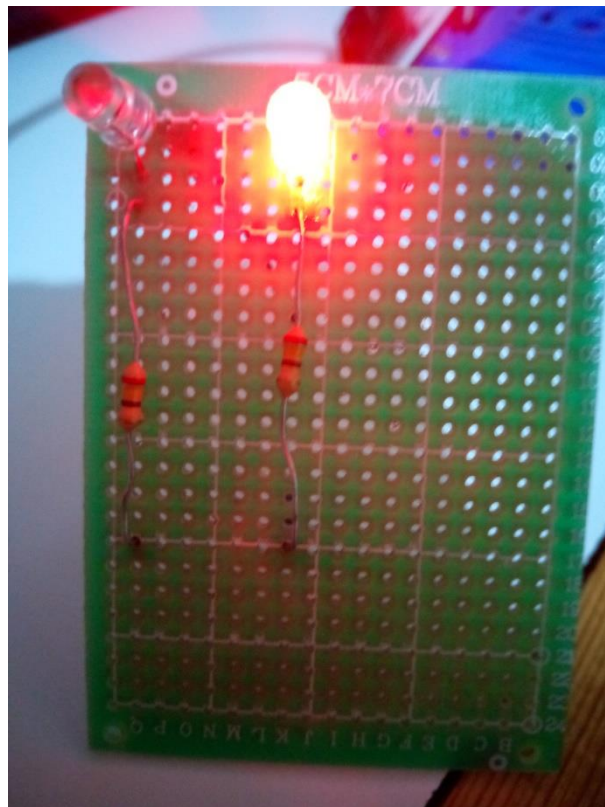


Figure III.6: The red LED will turn on if the temperature is higher than 38°C

### III.2.4 QR Code

QR stands for “quick response”, which means people get quick access to the information embedded in the code when it is scanned.

The concept behind a QR code is to generate an image that can be scanned and converted into something more meaningful by any modern Smartphone or Camera (with a QR code reader application).

In general, a QR code functions similarly to a barcode at the store. It's a machine-readable image that can be read quickly with a camera. A QR code is made up of a series of black squares and dots that mark various pieces of data. When it is scanned, it converts the information into an easily comprehensible form.

In our project, if an individual gives a QR code with a negative COVID-19 test, he may be excluded from the RedZone table. The following details must be used in the QR code:

- Person name
- Test result (positive or negative)

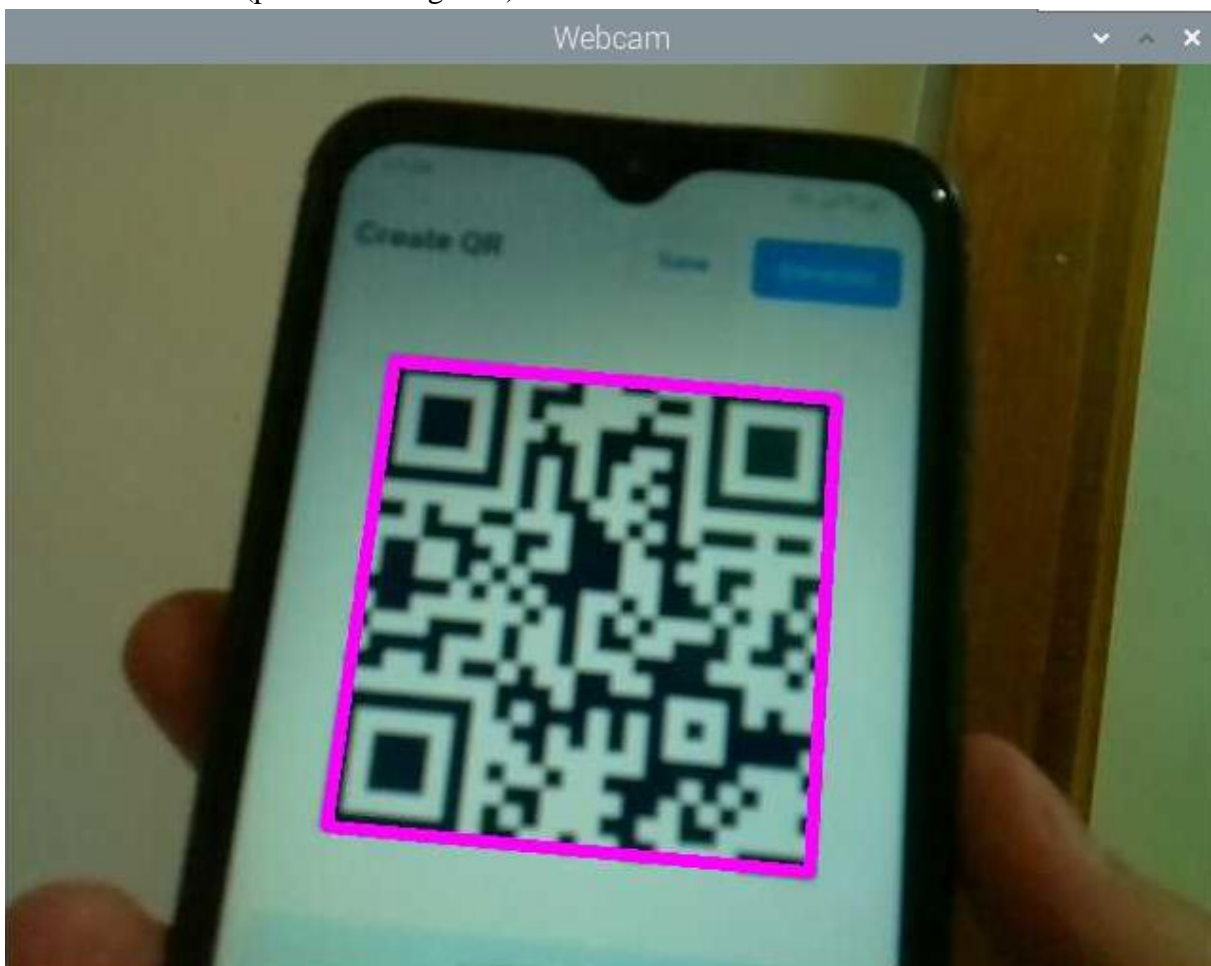


Figure III.7: The QR Code can be read by the Raspberry Pi camera module

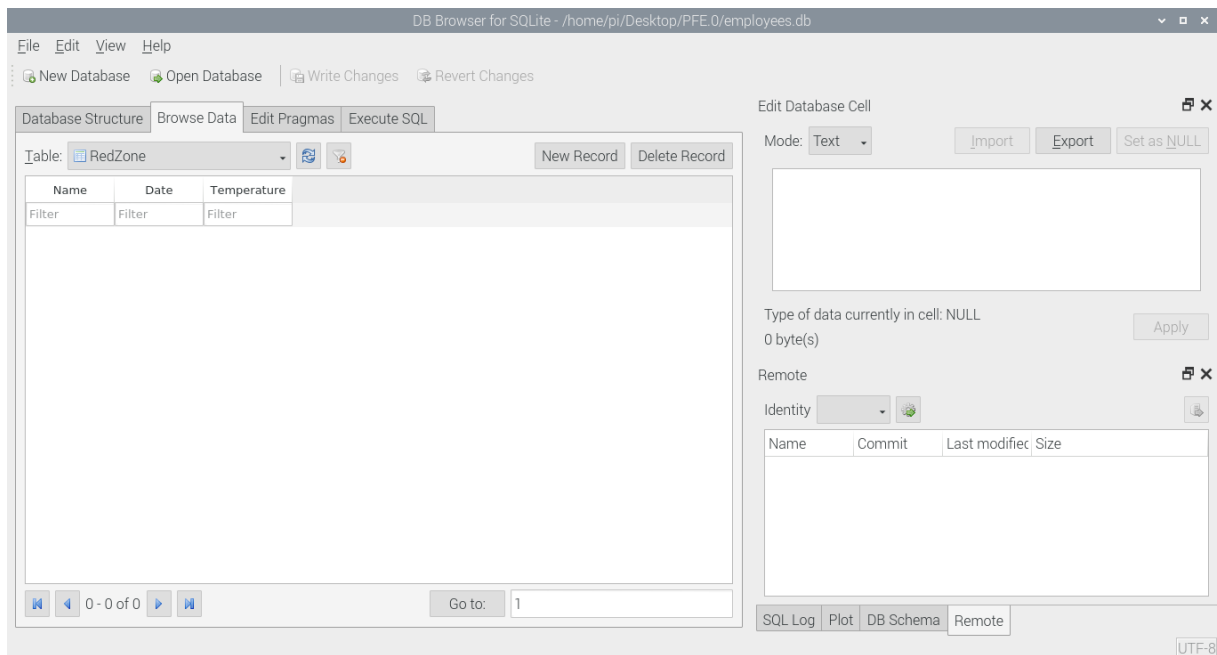


Figure III.8: If a negative test is provided, the person will be deleted from the RedZone table

### III.2.5 Speech synthesis

The computer-generated emulation of human speech is known as speech synthesis. It's used to convert textual content into aural information in situations where it's more practical.

After checking a person's temperature, a Speech synthesis sequence will be initialized. The following actions will be performed:

If the temperature is lower than 38 °C:

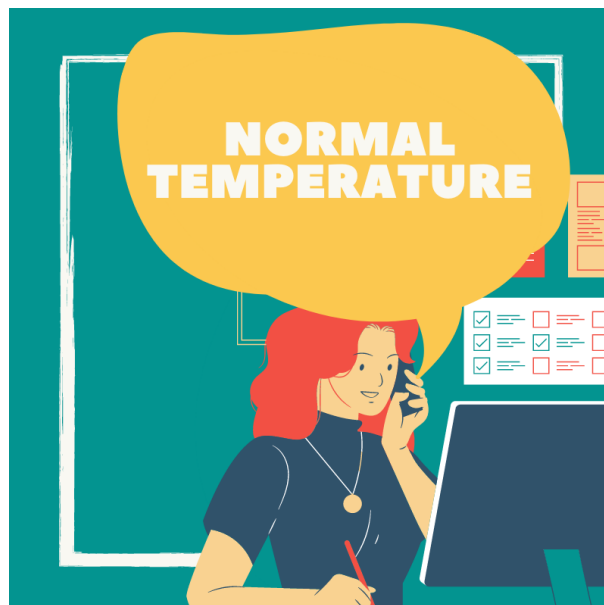


Figure III.9: If the temperature is lower than 38 °C the engine will say “Normal Temperature”

If the temperature is higher than 38 °C:



Figure III.10: If the temperature is higher than 38 °C the engine will say “High Temperature”

### III.3 Conclusion

In this chapter, we have discussed the simulation results of our project called fever detector. We presented the implementation of the face recognition algorithm, temperature checking, LEDs activation, data logging in the database, QR code reading, and finally the Speech synthesis.



## GENERAL CONCLUSION

In this project, we have proposed a smart thermometer system that uses the Digital Non-Contact Infrared Thermometer MLX90614 and facial recognition techniques with the Raspberry Pi 4b. For this, we use a VNC client to display the measurements and control the Raspberry Pi (we can use a smartphone, a tablet, or a computer) and an SQLite database for data gathering.

By using this device to measure the temperature of people and detect fever, we can help to fight the COVID-19 pandemic that is currently affecting the world.

The functionalities of our fever detector system have been successfully tested; the facial recognition, the temperature measurement using the MLX90614 IR sensor, save the measurements using an SQLite database, the speech synthesis, and the QR code.

The thesis was divided into three chapters (The functional design of the fever detector, The conception, and the simulation of results). The first chapter was focused on The Horned Beast diagram, the Octopus Diagram, and the FAST Diagram. In the second chapter, we have discussed the hardware and the tools needed and used to create our project. In the third and last chapter, we have discussed the results of our project, as well as a walk through the execution of our face recognition system, temperature checking, the speech synthesis, the SQLite database, and the results obtained from reading the QR code and removing a person from the database.

A suggestion for future projects is to use the MLX90640 thermal camera (Instead of the non-contact thermometer MLX90614) and a custom 3D printed case, this makes large-scale deployments simple for businesses of all sizes.

We hope that this thesis will aid the reader in grasping the fundamental concepts of facial recognition and Non-Contact Infrared Thermometers systems and providing a foundation of knowledge from which to develop future, more advanced systems.

## BIBLIOGRAPHY

1. Coronavirus disease (COVID-19). (n.d.). World Health Organization. <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/coronavirus-disease-covid-19>
2. Coronavirus outbreak: the role of companies in preparedness and responses. (2020, April 1). ScienceDirect. <https://linkinghub.elsevier.com/retrieve/pii/S2468266720300517>
3. Digital plug & play infrared thermometer in a TO-can. (n.d.). Melexis. <https://www.melexis.com/en/product/MLX90614/Digital-Plug-Play-Infrared-Thermometer-TO-Can>
4. Face Detection using Haar Cascades — OpenCV-Python Tutorials 1 documentation. (n.d.). OpenCV. [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_objdetect/py\\_face\\_detection/py\\_face\\_detection.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html)
5. Foong, N. W. (2020, November 9). Barcodes and QR Codes Decoder in Python - Towards Data Science. Medium. <https://towardsdatascience.com/barcodes-and-qr-codes-decoder-in-python-59615c5f2b23#:~:text=Based%20on%20the%20official%20documentation,files%20and%20raw%20intensity%20sensors>
6. Function Analysis system Technique (FAST) - Canadian Society of Value Analysis. (n.d.). Value Analysis Canada. <https://www.valueanalysis.ca/fast.php>
7. Functional Design: Definition, Process & Example. (n.d.). Study.Com. <https://study.com/academy/lesson/functional-design-definition-process-example.html>
8. Guide to QR Codes for Print & How They Work. (n.d.). CREATIVE MARKETING. <https://www.fastprint.co.uk/blog/quick-response-codes-what-are-they-and-how-do-they-work.html>
9. Guta, M. (2021, January 4). How To Create a QR Code in 5 Simple Steps. Small Business Trends. <https://smallbiztrends.com/2016/05/create-a-qr-code.html>
10. HOSSEIN MOKHTARIAN, ERIC COATANE'A, & HENRI PARIS. (2017, May 25). Function modeling combined with physics-based reasoning for assessing design options and supporting innovative ideation. ResearchGate. [https://www.researchgate.net/publication/317100248\\_Function\\_modeling\\_combined\\_with\\_physics-based\\_reasoning\\_for\\_assessing\\_design\\_options\\_and\\_supporting\\_innovative\\_ideation](https://www.researchgate.net/publication/317100248_Function_modeling_combined_with_physics-based_reasoning_for_assessing_design_options_and_supporting_innovative_ideation)
11. ISO 13485: What is it? Who needs Certification and Why? (2020, July 14). ISO 13485 Store. <https://13485store.com/medical-device-standards/what-is-iso-13485/>
12. Jeff Rude. (n.d.). The Function Wheel. Strategic Value Solutions. <http://www.svs-inc.com/wp-content/uploads/2012/04/Function-Wheel.pdf>
13. John Borza. (2011). FAST Diagrams: The Foundation for Creating Effective Function Models. The Altshuler Institute. [https://www.aitriz.org/documents/TRIZCON/Proceedings/2011-06\\_FAST-Diagrams-The-Foundation-for-Creating-Effective-Function-Models.pdf](https://www.aitriz.org/documents/TRIZCON/Proceedings/2011-06_FAST-Diagrams-The-Foundation-for-Creating-Effective-Function-Models.pdf)
14. Lima Moreira. (2020, June 18). How to make an IR thermometer for COVID-19. Hackster.Io. <https://www.hackster.io/moreiranextpcb/how-to-make-an-ir-thermometer-for-covid-19-047c17>
15. Matrix, S. O. A. (2020, September 29). ISO 14971:2019 - Basics of Medical Device Risk Management. Oriel STAT A MATRIX Blog. <https://www.orielstat.com/blog/iso-14971-risk-management-basics/>
16. MLX90614 Non-Contact IR Temperature Sensor Pinout, Datasheet, Equivalents & Specs. (2020, March 31). Components101. <https://components101.com/sensors/melexis-mlx90614-contact-less-ir-temperature-sensor>
17. Mohamed Zeglache. (2019, June). facial recognition using convolutional neural networks. [http://archives.univ-biskra.dz/bitstream/123456789/13215/1/zeglache\\_mohamed.pdf](http://archives.univ-biskra.dz/bitstream/123456789/13215/1/zeglache_mohamed.pdf)
18. P. (2020, February 5). Understanding of Convolutional Neural Network (CNN) — Deep Learning. Medium. <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>
19. Piltch, A. (2020, September 2). Raspberry Pi 4: Review, Buying Guide and How to Use. Tom's Hardware. <https://www.tomshardware.com/reviews/raspberry-pi-4#:~:text=With%20a%20more%20power%2Dhungry,more%20than%20the%203%20B%2B>
20. Saksham Bhutani. (2020, September 23). TouchFree v2: Contactless Temperature and Mask Checkup. Hackster.Io. <https://www.hackster.io/sakshambhutani2001/touchfree-v2-contactless-temperature-and-mask-checkup-d01dc8>
21. Singh, A. (2020, May 10). Feature Engineering for Images: A Valuable Introduction to the HOG Feature Descriptor. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/>

22. SMART Temperature Monitoring for Schools. (2019, November 15). Arduino Project Hub. <https://create.arduino.cc/projecthub/179110/smart-temperature-monitoring-for-schools-ac3f6c>
23. Solegaonkar, V. (2019, December 17). Introduction to Face Recognition - Towards Data Science. Medium. <https://towardsdatascience.com/face-recognition-25f7421a2268>
24. The Octopus diagram. (2016, August 27). lycée des métiers - léonard de vinci. <http://sti2d.lyceevinciblanquefort.fr/doku.php?id=etlv1:term:specifications#:~:text=The%20Octopus%20diagram,analysis%20determines%20the%20functional%20requirements>
25. UML Use Case Diagram Tutorial. (n.d.). Lucidchart. <https://www.lucidchart.com/pages/uml-use-case-diagram>
26. Upton, E. (2019, June 26). 4 on sale now from \$35. Raspberry Pi. <https://www.raspberrypi.org/blog/raspberry-pi-4-on-sale-now-from-35/>
27. Weng, L. (2017, October 29). Object Detection for Dummies Part 1: Gradient Vector, HOG, and SS. Lil'Log. <https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html>
28. What is ISO 13485? Easy-to-understand explanation. (n.d.). 13485Academy. <https://advisera.com/13485academy/what-is-iso-13485/>
29. What is ISO 14971:2019 Risk? (2020, July 15). ISO 13485 Store. <https://13485store.com/medical-device-standards/what-is-iso-14971/>
30. What is Unified Modeling Language (UML)? (n.d.). Visual Paradigm. <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/>