PALESTINE POLYTECHNIC UNIVERSITY

College of Information Technology and Computer Engineering

Smart Mirror

Dalah AL-Hashlamoon 201002

Morjana Abu Hussein 217034

Supervisor:

Wael Al Takrouri

Hebron- Palestine

June-2025

Table Of Contents

List of Figures

List of Tables

List of Acronyms

AI	**Artificial Intelligence**
API	**Application Programming Interface**
CNNs	**Convolutional Neural Networks**
DNNs	**Deep Neural Networks**
eMMC	**Embedded Multimedia Card**
FoV	**Field of View**
GPIO	**General Purpose Input Output**
HDMI	**High-Definition Multimedia Interface**
I2C	**Inter-Integrated Circuit**
I2S	**Inter-IC Sound Bus**
LDR	**Light Dependent Resistor**
LPDDR4	**Low-Power Double Data Rate 4**
LED	**Light Emitting Diode**
ML	**Machine Learning**
MOSI	**Master Out Slave In**
MISO	**Master In Slave Out**
NLP	**Natural Language Processing**
OS	**Operating System**
PWM	**Pulse-Width Modulation**

QR	**Quick Response**
RGB	**Red, Green, Blue**
RH	**Relative Humidity**
SPI	**Serial Peripheral Interface**
USB	**Universal Serial Bus**
UART	**Universal Asynchronous Serial Port**
VGA	**Video Graphics Array**
YOLO	**You Only Look Once**

Acknowledgement

We would like to begin by expressing our gratitude to Allah, the Almighty, who provided us with the strength and determination to accomplish this remarkable achievement. The success we have achieved would not have been possible without His mercy and support, and we pray for his blessings on this effort and guidance for what lies ahead.

We also extend our profound appreciation to the project supervisor, Eng.Wael Al Takrouri for his guidance and continuous support, which significantly contributed to the success of this project.

Furthermore, we thank Palestine Polytechnic University (PPU) for providing the necessary resources and a conducive environment that played a crucial role in achieving this accomplishment.

Thanks to our families and friends who have always stood by us, providing unwavering encouragement throughout this academic journey.

Together, as a team, we have reached this significant milestone, and we are grateful to Allah and every individual in this team for their unique efforts and contributions.

Thank you, from the bottom of our hearts.

Abstract

Traditional mirrors are no longer sufficient to meet the growing demand for convenience and connectivity in modern homes. The integration of advanced technologies into daily routines has transformed everyday objects into multifunctional smart devices, offering enhanced utility and user experience.

This project proposes the development of a Smart Mirror—a modernized mirror integrated with intelligent functionalities to assist users in their daily activities. The smart mirror system incorporates various features such as real-time weather updates, calendar synchronization, voice interaction, and personalized notifications. It also integrates a mobile application to allow seamless connectivity, enabling users to share photos via QR code. Additionally, the smart mirror supports environmental sensors for temperature, humidity, and brightness adjustments, along with a unique outfit color matching feature to provide users with suggestions on suitable outfit combinations, enhancing convenience and user interaction with dynamic displays and intuitive control.

Keywords: Smart Mirror, Home Automation, Voice Interaction, Machine Learning and AI, Outfit Color Matching.

الملخص

لم تعد المرايا التقليدية كافية لتلبية الطلب المتزايد على الراحة والاتصال في المنازل الحديثة. أدى دمج التقنيات المتقدمة في الروتين اليومي إلى تحويل الأشياء اليومية إلى أجهزة ذكية متعددة الوظائف توفر تجربة استخدام محسنة وفائدة أكبر.

يقترح هذا المشروع تطوير مرآة ذكية—وهي مرآة حديثة مدمجة بميزات ذكية لمساعدة المستخدمين في أنشطتهم اليومية. يتضمن نظام المرآة الذكية ميزات متنوعة مثل تحديثات الطقس في الوقت الفعلي، مزامنة التقويم، التفاعل الصوتي، والإشعارات المخصصة. كما يتكامل مع تطبيق الهاتف لتوفير اتصال سلس، مما يتيح للمستخدمين مشاركة الصور عبر رمز الاستجابة السريعة . بالإضافة إلى ذلك، تدعم المرآة الذكية مستشعرات بيئية لقياس درجة الحرارة، الرطوبة، وضبط السطوع، إلى جانب ميزة مطابقة ألوان الملابس لتقديم اقتراحات حول تنسيق الملابس المناسبة، مما يعزز تفاعل المستخدم مع شاشات العرض الديناميكية والتحكم التلقائي.

الكلمات المفتاحية: مرآة ذكية، أتمتة المنزل، إنترنت الأشياء، التفاعل الصوتي ،تعلم اله وذكاء اصطناعي، مطابقة ألوان الملابس.

Chapter 1

Introduction

1.1 Preface

With the rise of smart technology, everyday items are becoming more useful and interactive. This project focuses on turning a regular mirror into a smart mirror that displays useful information like time, weather, and calendar events while you get ready. It also supports voice commands, allowing you to check your schedule or get reminders without using your hands. A key feature is the outfit assistant, which suggests matching colors to help you choose what to wear more easily. The goal is to make daily routines simpler, faster, and user-friendly technology.

1.2 Problem Statement

In today's fast-paced world, people are always looking for ways to manage their time, access information quickly, and organize their daily routines without added complexity. Traditional home mirrors, while essential, lack the technology features needed to integrate useful information and real-time updates into everyday life.

1.3 Project Aims and Objectives

In this project, we propose a system that aims to provide the following features:

1) Develop a smart mirror that functions as an interactive personal assistant, providing real-time information such as time, date, weather, and calendar events.
    a) Set up the hardware components, like a Raspberry Pi, an LCD screen, and various sensors, to display useful information.
    b) Program the system to show real-time updates like the time, date, weather, and calendar events.
    c) Make sure the mirror connects to the internet to fetch and sync all the data.
2) Enable voice-activated control to ensure a hands-free user experience, allowing users to interact with the mirror using natural language commands to check their schedule or receive reminders.

a) Install a microphone and set up software that understands spoken commands.
b) Use voice assistants like custom-built tools to let users control the mirror hands-free.
c) Ensure the system responds quickly to make it feel natural and easy to use.

3) Implement a virtual outfit assistant, and color recommendations.
   a) Use the mirror's camera to detect the person's clothes and analyze the colors.
   b) Integrate an AI system to suggest matching outfit colors.
4) Design and integrate a user interface that enables the smart mirror to interact and take commands by voice commands and also display it on the mirror so that it is more time-efficient.
   a) Build a display that shows all the important information clearly and looks good on the mirror.
   b) Make the interface work seamlessly with voice commands while also showing visual feedback.
   c) Focus on making the system quick and enjoyable for users.

1.4 Project Requirements

This section will display the list of the functional and non-functional requirements of the system:

1.4.1 Functional Requirements

The following is a list of functional requirements:

1. Display real-time information such as time, date, weather, and personalized data like calendar events.
2. Support voice interaction through Google Speech Recognition API and includes statements like (e.g check outfit, take picture, what is the time) .
3. Provide outfit color recommendations using AI models.
4. Taking photos can be shared to smartphones via QR code.
5. Adjust the LED lighting based on surrounding light levels to improve visibility and save energy.

1.4.2 Non Functional Requirements

The following is a list of non-functional requirements:

1. Performance:

- The system must respond to voice commands within 2-5 seconds.
- Display updates (e.g., weather, calendar events) should be a fast response with minimal latency.

2. Energy Efficiency:

- The system becomes idle until a person comes closer to the mirror.

3. Availability:

- The system must be available almost all the time, minimizing downtime and errors in connecting the wifi when fetching data (such as weather updates and calendar events).
- The voice recognition system must work accurately in a noisy environment, with most speech recognition accuracy.

1.5 System Description

The system is designed to be both functional and visually appealing, ensuring that it fits easily into the home environment. The system consists of a traditional mirror outfitted with a hidden display screen, a microphone, and speakers. The display unit, embedded within the mirror, will show information such as time, date, weather, and calendar events directly on the mirror. This layout makes it easy for users to view information while getting ready, without the need for separate devices.

The system interface will be voice-controlled through an integrated microphone, enabling hands-free interactions. Users will be able to receive personalized recommendations, access calendar reminders. Additionally, the mirror will include a virtual outfit assistant that suggests outfit colors .

The expected dimensions of the mirror will align with standard wall-mounted mirrors, with a vertical layout to ensure easy visibility of displayed content (see Figure 1.1). This design allows it to fit comfortably in spaces such as bedrooms or entryways, blending technology into the decor without being intrusive.

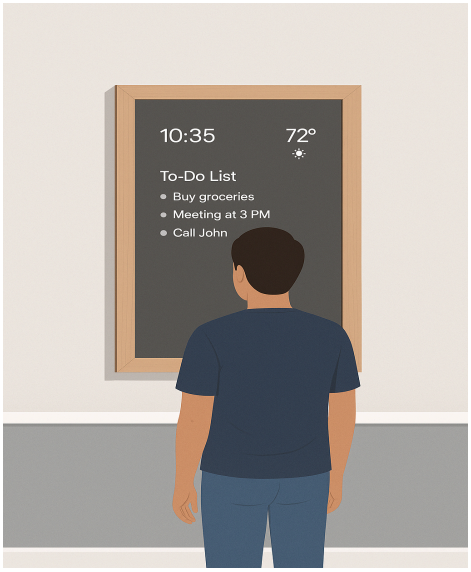


Figure 1.1: Expected layout of the smart mirror.

1.6 Project Limitations and Constraints

Here are some of the system limitations and constraints:

1. User Position (Height and Distance) : The system works best when the user is standing at an average height and at a comfortable distance (about 1-3 meters).
2. Wi-Fi Requirement : The system needs a stable Wi-Fi connection to show information like weather and calendar updates and to use voice assistant features.

1.7 Project Schedule

The tasks of the system implementation and operation are distributed along the first and the second semester summarized in Table 1.1.

Table 1.1: Project Schedule Timetable.

	First Semester			**Second Semester**			
Week	1 - 2	3 - 7	8 - 12	1 - 5	6 - 9	10 - 14	15
Selection of project idea							
Collecting the data and system analysis							
System design							
System implementation							
Unit and system testing							
Results							
Documentation							

1.8 Report outline

This report is organized as follows:

Chapter 1 provides a brief introduction to the system including the problem statement, the system requirements and system description. Chapter 2 discusses the most related keys theoretical basis and a discussion of the literature review. Chapter 3 outlines the project's design, including both hardware and software components. It discusses design choices, the conceptual background of the software, and presents a schematic diagram. Chapter 4 introduces the implementation. Chapter 5 describes the results. Chapter 6 discusses the conclusion and future work.

Chapter 2

Background

2.1 Preface

This chapter introduces the theoretical concepts that are important to our project. Following this, we'll dive into a literature review, comparing our project with what's been done before.

2.2 Theoretical Background

This section presents the key theoretical principles behind our project's development.

2.2.1 Voice Recognition

Voice recognition allows users to control the smart mirror through spoken commands, offering a hands-free and user-friendly experience. Using Google Speech Recognition API, the system converts speech into text, processes audio segments to recognize patterns, and operates accurately even in noisy settings. It relies on cloud-based processing and machine learning to improve performance and responsiveness over time.

2.2.2 Computer Vision

Computer vision is a field of artificial intelligence that enables machines to interpret and understand visual information from the world, such as images and videos.[2] By using algorithms and models, it allows computers to analyze visual data, recognize patterns, and make decisions based on that input. In this project, computer vision enables the system to detect and recognize the user by analyzing their appearance through the camera.

2.2.3 Machine Learning

Machine learning is a branch of artificial intelligence (AI) that focuses on developing algorithms and models that enable systems to learn from data and improve their performance over time without explicit programming for each task. In the context of the system, It plays an important

role in enhancing their functionality and user experience[3]. Using techniques such as neural networks, decision trees, and deep learning, it also enables the system to perform advanced tasks such as facial recognition, voice interaction, and personalized recommendations. For instance, facial recognition algorithms allow the system to identify users and deliver customized content, such as notifications or outfit suggestions.

2.2.5 Cloud Services

Cloud services refer to a range of computing resources such as storage, processing power, and software applications delivered over the internet[4]. These services allow devices and systems to access, process, and store data without relying solely on local resources. In the context of our system, cloud services are vital for enabling dynamic and personalized features. They provide real-time access to data such as weather forecasts, calendar events and news updates. By connecting to cloud platforms, our system can deliver up-to-date information and synchronize with user accounts for personalized content. It also enhances voice interaction capabilities by integrating with platforms like Google Speech Recognition API[5].

2.3 Literature review

Our system has become an exciting area of innovation, combining traditional mirror functions with digital interactivity enabled by advancements in AI and human-computer interaction. These mirrors provide users with important information, such as time, weather, and personal notifications, through a seamless digital interface.

2.3.1 Smart Mirror (Open Source Project)

This project involves transforming a standard two-way mirror into a smart, interactive display powered by a Raspberry Pi. It provides users with live updates such as weather forecasts, news, and upcoming events through a modular and customizable digital interface. The system supports voice interaction and allows developers to add new features using plugins. Despite its versatility and community support, setting up the hardware and configuring the software may be challenging for users with limited technical experience[6].

2.3.2 BYECOLD Smart Mirror (Amazon Product)

The BYECOLD Smart Mirror is a commercially available smart bathroom mirror designed to enhance daily routines with integrated technology. It features a 40"x24" frameless design equipped with LED lighting, anti-fog functionality, and a touch sensor interface. The mirror connects to Wi-Fi, allowing it to display real-time weather updates, time, and date information. Additionally, it includes built-in Bluetooth speakers for audio playback. While users appreciate its practical features and sleek design, some have reported challenges with the initial setup, particularly regarding Wi-Fi connectivity and app integration. Overall, the BYECOLD Smart Mirror offers a blend of functionality and modern aesthetics, making it a notable example of smart mirror technology available to consumers[7].

Table 2.1: comparison between current work and related works

Feature/Item	Open Source Smart Mirror	BYECOLD Smart Mirror (Amazon)	Our Project
Design	Uses a two-way mirror with Raspberry Pi and screen behind it.	Frameless LED smart mirror with built-in touch and Bluetooth.	Interactive mirror with AI module and voice assistant features.
Methodology	Modular system with customizable plugins and voice assistant integration.	Built-in smart functions including weather, time, and music via Wi-Fi.	Uses custom AI for outfit detection and cloud services.
Connection Type	Requires setup with Raspberry Pi and optionally connects to Wi-Fi.	Connects to Wi-Fi and Bluetooth for app sync and music.	Requires setup with Raspberry Pi and necessary connects to Wi-Fi.

2.4 Summary

This chapter outlines the core technologies behind our smart mirror, including voice recognition, computer vision, machine learning, and cloud services. It also reviews existing smart mirror projects, highlighting how our system offers more advanced features like AI-based outfit detection and voice interaction. A comparison table shows key differences in design, functionality, and connectivity.

Chapter 3

System Design

3.1 Preface

This chapter provides an overview of the essential hardware and software components intended for our project.

3.2 System components and Design alternatives

3.2.1 Design alternatives

In this section, we will discuss some design alternatives for displaying our system features such as time and weather, and provide an overview of the system architecture distributed into input, output, and control units. As for alternatives to our system, we have two options:

First Design Option: Smartwatch

Smartwatch is a portable device capable of displaying time, weather updates, and notifications directly on the user's wrist [8]. While convenient and always accessible, its small screen size limits the amount of information displayed, making it less effective for multitasking. Additionally, smartwatches are designed for personal use and do not provide a shared solution suitable for a household. As shown in Figure 3.1 below.

Figure 3.1: Smart Watch.

Second Design Option: Echo Show

Echo Show combines a touchscreen with voice assistant capabilities, allowing users to view weather updates, notifications, and IoT controls[9]. Although they are compact and easy to set up, they lack the reflective dual-purpose functionality of a smart mirror and may not seamlessly blend into the home environment. Below, Figure3.2 shows the system.

Figure 3.2: Echo Show. [10]

After looking into the available options, the smart mirror clearly stands out as the most practical and stylish choice. Unlike smartwatches or Echo Show devices, it combines functionality with design, hands-free interactive experience. It works both as a regular mirror and as a smart display that delivers useful information.

What makes it even more appealing is that it can be used by everyone in the household, so there's no need for each person to have their own device. Its large screen makes content easier to see than on a smartwatch, and unlike the Echo Show, it still looks like a familiar household item rather than a tech gadget.

3.2.2 Hardware Components

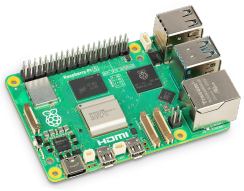
In this section, we provide clarity on the hardware components used in the project and give our reasoning for selecting them. We also present a detailed comparison with other components.

3.2.2.1 Processing Unit

The processing unit is the brain of our system. It receives and processes sensor data, executes algorithms, and controls system components. When comparing and evaluating three choices

Raspberry Pi 4, Jetson Nano, and BeagleBone Black we have studied the possible options, compared them, and chosen the most suitable for our system. The summary of comparison between three options summarized in Table 3.1.

Table 3.1: Comparison between Processing Unit options.

Feature / Item	**Raspberry Pi 4[11]**	**Jetson Nano[12]**	**BeagleBone Black[13]**
Processor	Quad-core ARM Cor-tex A72, 1.5 GHz	Quad-core ARM Cortex-A57, 1.43 GHz	ARM Cortex-A8, 1 GHz
RAM	2GB, 4GB, or 8GB LPDDR4 RAM	4GB LPDDR4	512MB DDR3
Storage	MicroSD card slot	MicroSD card slot + 16GB eMMC storage	4GB eMMC + MicroSD card slot
Processing Power	Low	High; optimized for AI/ML workloads	Medium; limited for basic applications
Connectivity	Ethernet,Wi-Fi, Bluetooth	Ethernet, no built-in Wi-Fi or Bluetooth	Ethernet, no built-in Wi-Fi or Bluetooth
I/O Ports	GPIO, I2C, I2S, SPI , UART, PWM	GPIO, I2C, SPI, UART	GPIO, I2C,SPI, UART
Cost	$35 – $75	$35 – $75	$68-$73
Images			

The Raspberry Pi 4 is ideal for our system due to its strong performance, flexibility, and availability. With a quad-core processor, up to 8GB RAM, built-in WiFi and Bluetooth, it supports advanced tasks like voice recognition and AI. It also allows high-resolution display output and easy sensor integration, making it perfect for an interactive smart mirror.[14]

3.2.2.2 Camera Module

A camera is a hardware component integrated into the system used to input image streams to facial detection alghorithm . It captures images of the user, which are processed by algorithms to provide personalized interactions such as clothing suggestions[16].

We compared three options: Raspberry Pi Camera Module 3, Logitech USB Webcam, and Intel RealSense D435. The comparison is presented in Table 3.2.

Table 3.2: Comparison between Camera module options.

Feature /Item	**Raspberry Pi Camera Module 3[17]**	**Silver Line USB Webcam[18]**	**Intel RealSense D435[19]**
Resolution	12 MP	1080p	1920x1080 (RGB) with depth sensing
Autofocus	Yes	Yes	No
Integration with Pi	Seamless (designed for Raspberry Pi)	Easy (via USB)	Moderate (requires drivers and setup)
Depth Sensing	No	No	Yes (depth and RGB data)
FoV	66°(standard),102° (wide-angle version)	78°	87° horizontal, 58° vertical
Cost	$25 – $38.50	$16.02 – $66	$314.00 – $318.99
Image			

The USB Silver Line webcam is a practical choice for our smart mirror due to its easy USB integration, 1080p resolution, and built-in autofocus. It supports reliable video streaming and image capture for outfit recognition and user interaction, and its compact, plug-and-play design makes setup simple without extra hardware or drivers.

3.2.2.3 Brightness Level Sensor

A brightness sensor measures the ambient light level in the environment to adjust the RGB Strip's brightness automatically. This ensures optimal visibility of the displayed information while reducing eye strain and conserving power[21][22]. By detecting the surrounding light intensity, the mirror can low-light conditions and brighten in well light environments, enhancing the user experience. Three options for the brightness sensor are considered in Table 3.3.

Table 3.3: Comparison between Brightness Sensors.

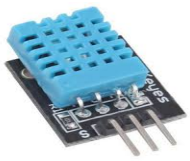
Feature/Item	**TSL2561 (Digital Light Sensor)[23]**	**BH1750 (Ambient Light Sensor)[24]**	**Photoresistor[25]**
Type	Digital sensor with I2C communication	Digital sensor with I2C communication	Analog sensor
Light Sensitivity Range	0.1 to 40,000 lux	1 to 65,535 lux	Limited range, dependent on configuration
Integration with Pi	Seamless with Raspberry Pi (I2C support)	Seamless with Raspberry Pi (I2C support)	Requires ADC for Raspberry Pi
Cost	(~$5-$10)	(~$5-$10)	(~$1-$3)
Image			

The photoresistor is a simple, low-cost solution for detecting ambient light in our smart mirror. It changes resistance based on light levels, allowing basic lighting adjustments. Although less accurate than digital sensors, it's easy to use and ideal when high precision isn't needed. With an ADC, it integrates well with the Raspberry Pi, making it a practical choice for brightness detection.

3.2.2.4 Temperature and Humidity Sensor

This sensor measures the ambient temperature and relative humidity in the room, providing real-time environmental data[26]. This can be displayed on the system interface to enhance user experience by offering useful insights like potential weather impacts on indoor conditions, we have compared between the three sensors in Table 3.4.

Table 3.4: Temperature and Humidity Sensors comparison.

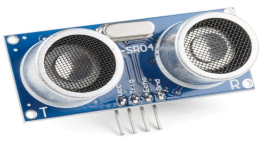
Feature/Item	DHT22[27]	DHT11[28]	BME280[29]
Accuracy (Temp)	±0.5°C	±2°C	±1°C
Accuracy (Humidity)	±2% RH	±5% RH	±3% RH
Integration with Pi	Easy (1-wire protocol)	Easy (1-wire protocol)	Seamless (I2C or SPI)
Cost	$10	$5	$20
Best For	Accurate, wide range, reliable readings	Basic projects with limited precision	High precision, multiple environmental data
Response Time	2 seconds	5 seconds	1 second
Image			

As we notice, The DHT22 and BME280 are both excellent sensors for measuring temperature and humidity. But for our project, The DHT22 sensor is a more suitable choice; It is cost-effective, accurate, and straightforward to integrate with a Raspberry Pi 4, while the BME20 sensor for bigger projects.

3.2.2.5 Motion Detection Sensor

A motion detection sensor is a device used to identify the movement of motion in its area[30]. We compared it with the most efficient option: ultrasonic sensors. The comparison is presented in Table 3.5.

Table 3.5: Comparison between Motion Detection Sensor options.

Feature	**PIR(Passive Infrared Sensor)[31]**	**HC-SR04 Ultrasonic sensor [32]**
Technology	Detect infrared radiation from warm objects (like humans)	Emits ultrasonic waves and measures the echo time
Range	Around 3–6 meters	Maximum range 1-4 meters
FoV	Wide (120°)	Narrow (15°)
Environmental Impact	Works well in darkness or light	May be affected by temperature and humidity
Images		

For our system, the best choice is the PIR sensor. It detects infrared radiation from warm objects (like humans), enabling functionalities such as activating the voice assistant and turning on the mirror display automatically when motion is detected.

3.2.2.5 Microphone

The microphone is a key component of the system, enabling voice-based interaction and hands-free operation [33]. It facilitates the processing of voice commands, integration with virtual assistants, and execution of tasks such as information retrieval or media playback. Table 3.6 provides a comparison of various microphones.

Table 3.6: Microphone Types.

Feature/Item	USB Microphone[34]	Shure MV88+ Bluetooth Microphone[35]
Connection	USB	Bluetooth
Setup	Plug-and-play	Wireless, requires Bluetooth pairing
Cost	~$10-$20	~$50-$70

We chose the USB microphone for its easy setup, high audio quality, and reliable voice detection with noise cancellation. Its plug-and-play design allows smooth integration and consistent performance for voice commands, while its compatibility with virtual assistant platforms adds versatility.

Figure 3.3: USB Microphone[36]

3.2.2.6 Speaker

A speaker in our system is used for audio output, enabling features like voice assistant responses, notifications, music playback, and multimedia integration[37]. The choice of speaker impacts sound quality, volume, and user experience. The ideal speaker for our system should be compact, offer clear audio, and integrate seamlessly with the system. In Table 3.5, the comparison between speaker types is shown in Table 3.7 below.

Table 3.7: Speaker Types.

Feature / Item	USB Speaker[38]	Bluetooth Speaker[39]
Audio Quality	High with stable connection	High but may vary with distance
Setup	Plug-and-play	Requires pairing and setup
Connection Type	USB	Bluetooth
Price	~$15-$50	~$20-$100

For our system, a USB speaker is a compact, affordable, and plug-and-play audio solution ideal for our system. It connects directly to the Raspberry Pi via a USB port, eliminating the need for additional audio output interfaces. It provides clear audio for voice assistants, notifications, and multimedia playback while being easy to integrate into the system. As shown in Figure 3.4 below.

Figure 3.4: USB Speaker[40]

3.2.2.7 LCD Display

An LCD monitor is used in the system to display key information through the smart mirror, allowing easy user interaction. A 24-inch HP LCD was chosen for its clear display, suitable size, and good balance of price and performance. Connected to the Raspberry Pi via a VGA to HDMI adapter, it provides reliable, high-quality visuals and fits well behind the two-way glass for seamless integration.

Figure 3.5: HP Monitor[42]

3.2.2.8 RGB Light Strip

RGB light strips enhance the smart mirror by providing customizable, colorful lighting for both function and style. They improve display readability and add ambient edge lighting for a modern look. Paired with a brightness sensor, the system can auto-adjust light levels based on surroundings, ensuring comfort, visibility, and energy efficiency.

Figure 3.6: RGB Light Strip.[44]

3.2.2.9 Analog to Digital Converter ICU

In our system, we needed an ADC to read analog data from components like a photoresistor (light sensor) and to help control features like an RGB LED strip. Two common ADCs are the MCP3008 and ADS1115 compared in the following Table.

Feature/Item	MCP3008	ADS1115
Resolution	10-bit	16-bit
Input Channels	8	4
Interface	SPI	I2C
Cost	3 $	5 $

We chose the MCP3008 because it provides 8 input channels, perfect for reading sensors like the photoresistor. It uses SPI, which works faster with Raspberry Pi, and its 10-bit resolution is accurate enough to control features like the RGB LED strip based on ambient light. It's also cheaper and easier to use for our setup.

3.2.2.10 Power Supply

A reliable power supply is essential for ensuring the smooth operation of a Raspberry Pi, especially in projects like a system where stability is crucial. The Raspberry Pi v 4 requires a 5V, 3A USB-C power adapter to deliver consistent power to its components and connected peripherals, such as sensors, displays, and cameras. Using an official or high-quality third-party power supply prevents issues like undervoltage, which can lead to system instability or performance degradation. Ensuring the power supply has sufficient wattage and reliability is vital to maintaining the efficiency and longevity of the system[45]. As shown in Figure 3.7.

Figure 3.7: USB-C power adapter. [46]

3.2.3 Software Components

There are several other options for smart mirror system frameworks that can be considered for the project, including:

3.2.3.1 Operating System

For the smart mirror project, three operating system options were evaluated: Raspbian (Raspberry Pi OS), Arch Linux, and Ubuntu. Each has unique strengths and is suited to different project requirements. The comparison is in Table 3.8 mentioned below.

Table 3.8: The differences between Ubuntu, and Arch Linux

Characteristic	Ubuntu[47]	Raspbian[48]	Arch Linux[49]
Inter-platform Operability	Cross-platform (supports many devices)	Cross-platform (optimized for Raspberry Pi)	Cross-platform (requires manual configuration)
Language	Python, C++, and other common languages	Pre-installed support for Python, C++, and more	Python, C++, and others (custom installation)
Supported Systems	Windows Subsystem, Linux, macOS, Raspberry Pi	Raspberry Pi only	Cross-platform (various devices)
Tools	Pre-installed tools for development and multimedia	Pre-installed tools for development	Minimal; requires manual installation
Support High-End Sensors and Actuators	Yes	Yes	Yes (requires additional configuration)
High-End Capabilities	High (supports advanced applications like AI)	Moderate	Very High (can be optimized for specific use cases)
Modularity and Active Community	Extensive and user-friendly	Extensive and beginner-friendly	Niche but highly technical and experienced

We chose Raspbian OS for its lightweight, stable performance and built-in tools optimized for Raspberry Pi. It simplifies smart mirror development and has strong community support. While both C++ and Python are available, Python was selected for its ease of use and rich libraries, making it ideal for fast development and prototyping.

3.2.3.2 MagicMirror Framework

MagicMirror serves as the basis of the system software. It provides a modular and open-source platform for creating customizable smart mirror applications. Developers can integrate existing modules or create custom ones to display information like time, weather,and news[50].

3.2.3.3 Node.js

Node.js is a runtime environment used to run the MagicMirror framework. It handles server-side operations, enabling seamless communication between the user interface and backend functionalities.[51]

3.2.3.4 Visual Studio Code (VS Code)

A popular code editor used for developing and debugging the smart mirror's software. It supports multiple programming languages, including JavaScript and Python, and offers extensions for easier development, such as Git integration and syntax highlighting.[52]

3.2.3.5 Python

Python is used in our system to integrate sensors and peripherals like cameras, using libraries such as OpenCV for facial recognition and Flask for web functionality. It offers powerful tools like NumPy and Pandas for data analysis, and TensorFlow for machine learning. Python's clean syntax makes the code easy to read and maintain, which helps reduce errors and speed up development. Its wide compatibility and strong community support make it a reliable and efficient choice for building smart mirror applications.

3.2.3.6 Algorithms

1. MobileNet-SSD

The MobileNet-SSD (Single Shot MultiBox Detector) is a lightweight deep learning model used to detect objects in real-time. In our system, this algorithm is responsible for detecting a person in the camera. It processes video frames and identifies various objects, our system looking for the "person" class. Once a person is detected, the model draws a bounding box around them and selects two regions within that box: the upper body (for the shirt) and the lower body (for the pants). This step allows the system to isolate the relevant parts of the outfit for analysis.[54].

2. K-Means Clustering

K-Means is an unsupervised machine learning algorithm used to find clusters in data. In our project, it’s applied to the pixels of the shirt and pants regions to identify the dominant color in each. The pixels are grouped into k = 4 color clusters, and the one with the highest number of pixels is considered the dominant color. This method helps reduce noise from lighting or patterns in the clothing and provides a clean, representative color for each clothing item[55].

3. RGB to HSV Conversion Algorithm

In our system, after detecting the dominant color in RGB format, the system converts it to HSV (Hue, Saturation, Value) because this color space is more close to how humans see color. In HSV, Hue represents the type of color (like red or blue), Saturation indicates how pure or faded the color is, and Value describes brightness. This makes it easier to distinguish similar colors, such as dark red vs. pink, which can be hard to separate in RGB. By using HSV, the system can classify colors more accurately and match them to known color names[56].

3.3 Conceptual System Description

The block diagram illustrates the architecture of the proposed smart mirror system, with the Raspberry Pi 4 serving as the central control unit. It connects to multiple input devices including a microphone for voice commands, a camera for facial and motion detection, a PIR sensor to

detect user presence, temperature and humidity sensors to monitor the room environment, and a brightness sensor to adjust lighting conditions. Based on the input data, the Raspberry Pi processes and coordinates responses through various output devices. These include a speaker for audio feedback, an RGB LED strip that adapts to ambient light, and an LCD display that presents real-time information such as time, weather, and notifications. Additionally, the system is linked to cloud services, enabling synchronization of user data and remote updates.

The general block diagram in Figure 3.8 below, illustrates the interconnected components of the system.

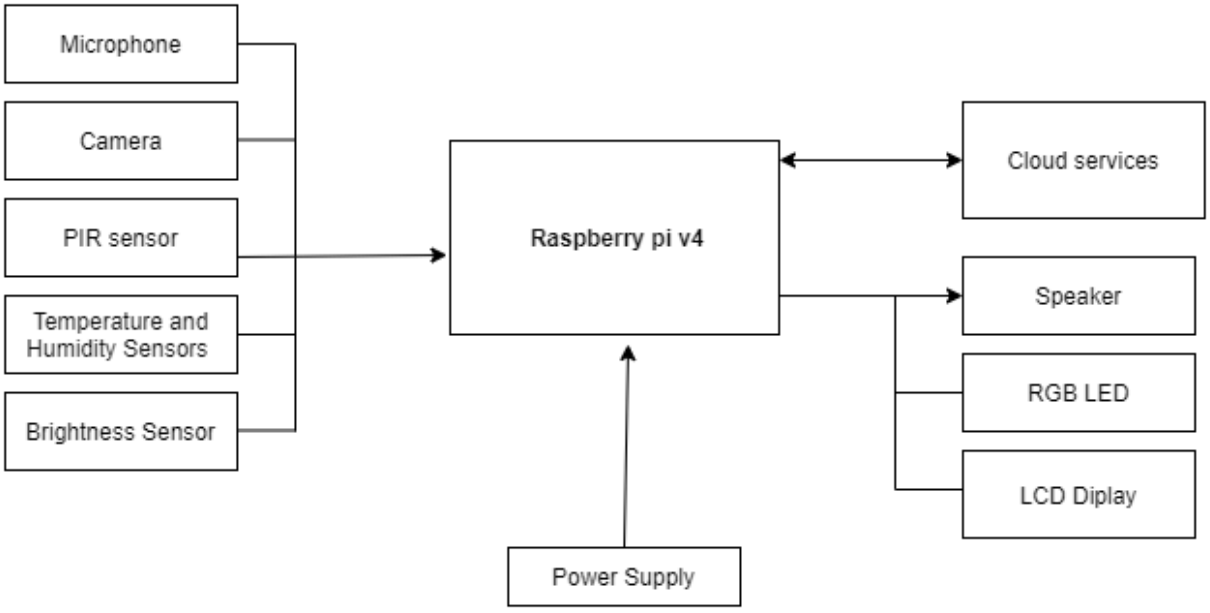


Figure 3.8 : General Block Diagram

3.4 Algorithms and methodologies

In this section, we will discuss the sequence diagram and pseudocode used to describe the system's processes. These methodologies help to outline the step-by-step interactions and the logical flow of the tasks within the system.

3.4.1 Pseudo-code

The algorithm outlines the operation of the system. It initializes components, connects to Wi-Fi, and syncs real-time data like weather and notifications. Using an PIR sensor and face detection, it detects users, activates the display, and processes voice commands to check outfits, update weather, or take photos. When no user is detected, the display turns off to save energy. As shown in the following Figure 3.9, 3.10, 3.11.

```
BEGIN
    INITIALIZE logging system
    DEFINE constants: WAKE_WORD, WEATHER_CITY, WEATHER_API_KEY

    FUNCTION speak(text):
        Convert text to speech using gTTS
        Play the audio output
        Log spoken message

    FUNCTION update_status(status):
        Write assistant status to a status file

    FUNCTION listen():
        Use microphone to capture user audio
        Convert audio to text using Google Speech Recognition
        RETURN text in lowercase

    FUNCTION wait_for_face():
        INITIATE camera and face detector
        LOOP:
            CAPTURE frame
            DETECT faces in frame
            IF WAKE_WORD is detected in speech OR face detected AND motion sensor active:
                TURN ON display
                GREET user
                RETURN True
            IF 'q' is pressed:
                BREAK loop
        CLEAN UP resources
        RETURN False

    FUNCTION wait_for_wake_word():
        LOOP:
            LISTEN for user input
            IF WAKE_WORD in input
                RESPOND
                RETURN True
```

Figure 3.9: listen, speak, and wake functions

```
FUNCTION analyze_outfit():
       LOAD MobileNet-SSD model
       OPEN camera and capture frames for 10 seconds
       DETECT person in frame
       IDENTIFY shirt and pants region
       ANALYZE dominant colors in both regions
       CLASSIFY colors into names (e.g., red, blue)
       PROVIDE outfit combination feedback based on matching rules

   FUNCTION get_weather(city):
       CALL OpenWeatherMap API
       RETURN formatted weather string

   FUNCTION search_wikipedia(query):
       RETURN 2-sentence summary from Wikipedia
       HANDLE disambiguation or errors

   FUNCTION get_general_answer(query):
       RETURN basic hardcoded responses for common phrases

   FUNCTION handle_user_input(user_input):
       IF greeting:
           RESPOND with greeting
       ELSE IF time/date requested:
           RESPOND with current time or date
       ELSE IF weather requested:
           PARSE city from input
           SPEAK weather info
       ELSE IF take picture command:
           CALL external script to take picture
       ELSE IF Wikipedia search command:
           EXTRACT query
           SPEAK Wikipedia result
       ELSE IF outfit analysis command:
           CALL analyze_outfit()
       ELSE IF temperature command:
           SPEAK room temperature from sensor
       ELSE IF exit command:
           SPEAK goodbye
           RETURN False
       ELSE:
           GET general response or fallback message
       RETURN True
```

Figure 3.10: outfit, weather, and command function.

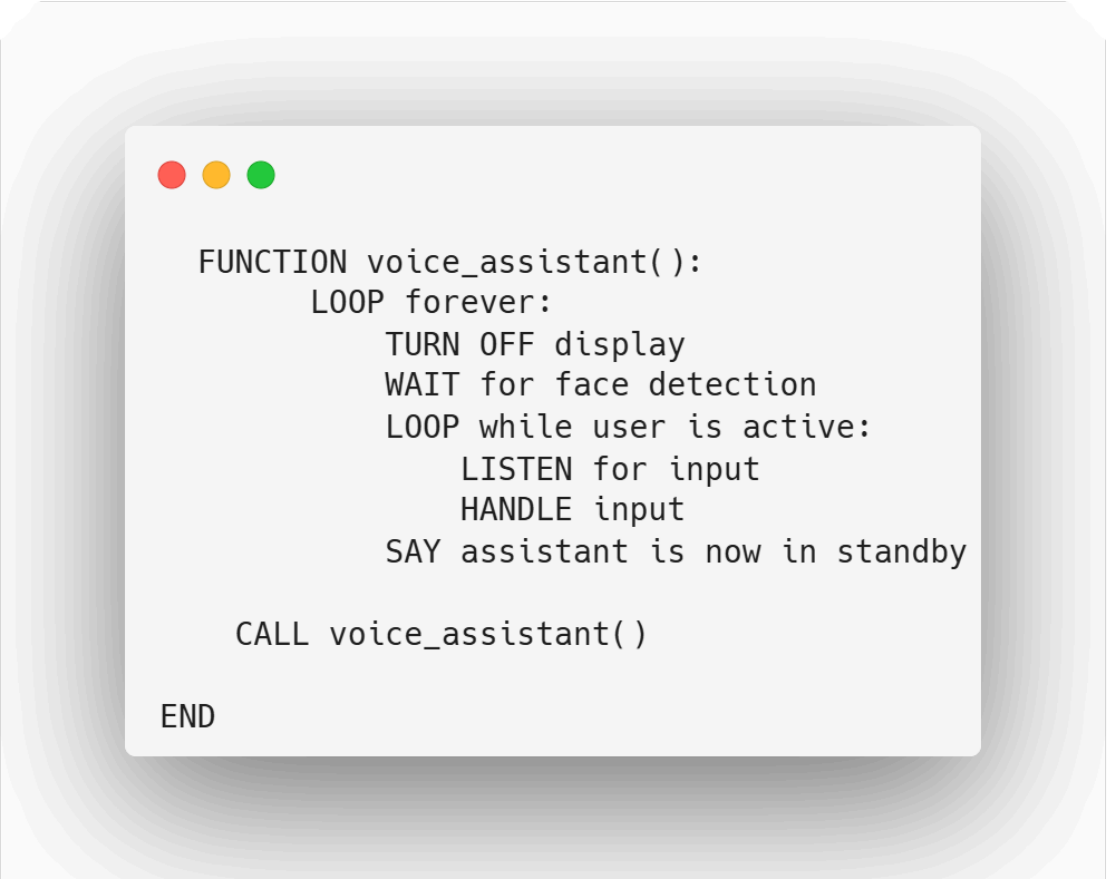


Figure 3.11: The main function of our system

This code defines the logic for a smart voice assistant system that integrates face detection, speech recognition, outfit analysis, and various assistant functionalities. It initializes key constants and sets up helper functions like speak() for text-to-speech, listen() for voice input, and update_status() to record assistant activity. The system can wake up either through a wake word or by detecting a face and motion as shown in Figure 3.9 above. Once activated, it listens for user commands and handles them using handle_user_input() as shown in Figure 3.10 above, responding to requests such as telling the time, checking the weather via an API, searching Wikipedia, analyzing outfit colors using a pre-trained MobileNet-SSD model, and interacting with sensors or scripts (like taking pictures or reporting room temperature). The assistant continuously loops between standby and active modes, managing display power and user interaction until explicitly exited as shown in Figure 3.11 above.

3.4.2 Sequence Diagram

The sequence diagram illustrates the interactions between the system components, showcasing how tasks are executed to achieve the system's objectives. The diagram highlights how data flows between the user, Raspberry Pi, APIs, and display to deliver real-time information and functionality.As shown in Figure 3.12 below.

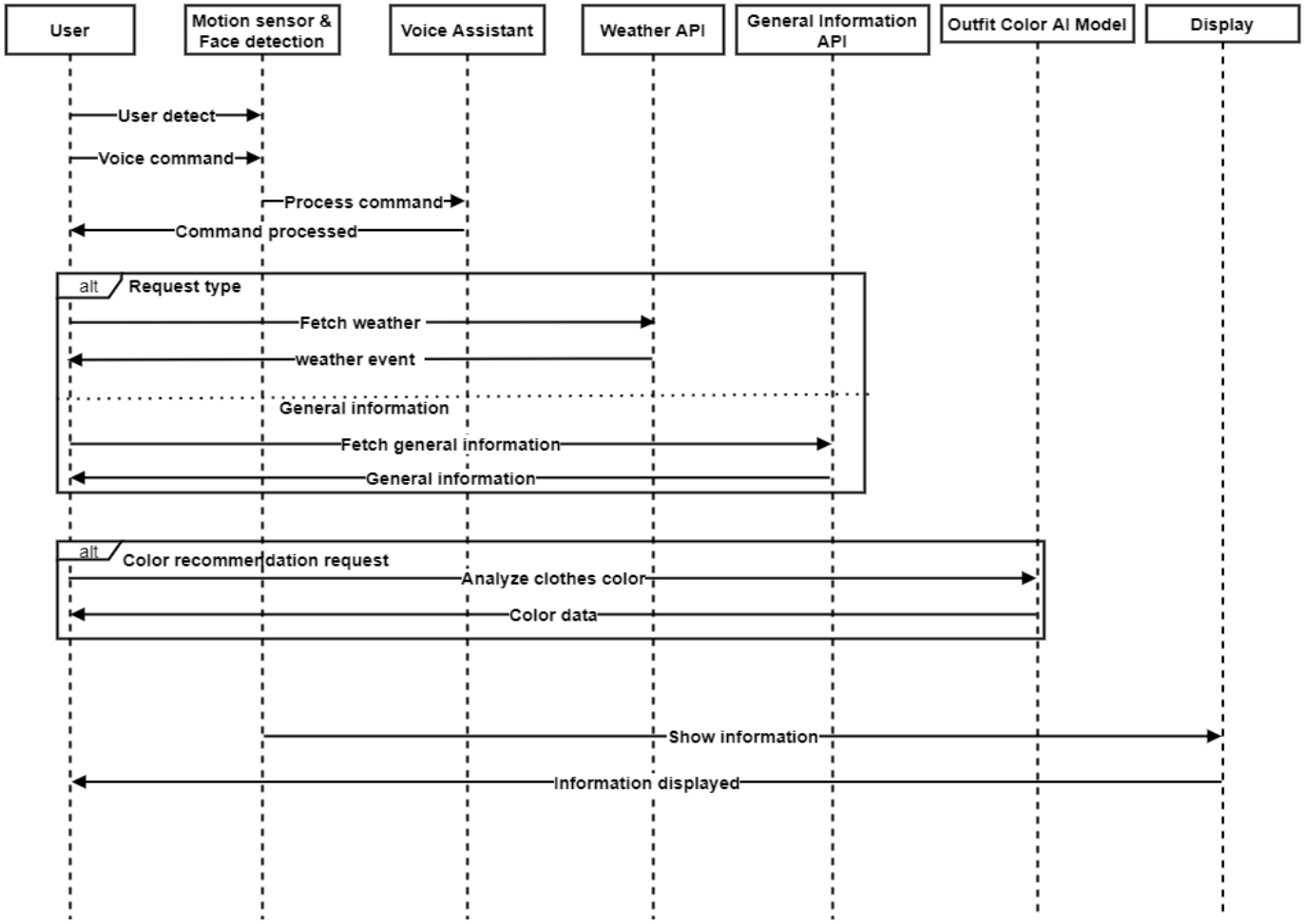


Figure 3.12: Sequence diagram of the system.

3.5 Schematic Diagram

The schematic diagram in Figure 3.13 shows the full hardware setup of the Raspberry Pi-based smart mirror system. The Raspberry Pi is powered by a 5V USB-C adapter and connects to an HDMI display. It also connects to USB devices like a camera for capturing images, a microphone for voice input, and a speaker for sound output. A DHT22 sensor is connected to read temperature and humidity, using a 10kΩ resistor to keep the signal stable. A motion sensor (PIR) is also included to detect movement and activate the mirror when someone is nearby.

To detect light levels, a photoresistor (LDR) is connected with a 10kΩ resistor and sends its analog signal to an MCP3008 chip, which turns it into a digital signal that the Raspberry Pi can understand. The MCP3008 uses specific GPIO pins and works through SPI communication. A Micro SD card module is also added to the same SPI lines and powered safely with 3.3V.

An RGB LED strip is connected to three Raspberry Pi pins that control red, green, and blue colors using PWM to adjust brightness smoothly. All devices share the correct power supply (3.3V or 5V) and a common ground, ensuring everything runs safely and reliably. This setup allows the smart mirror to sense its environment, detect movement, adjust lighting, and respond with sound and visuals.

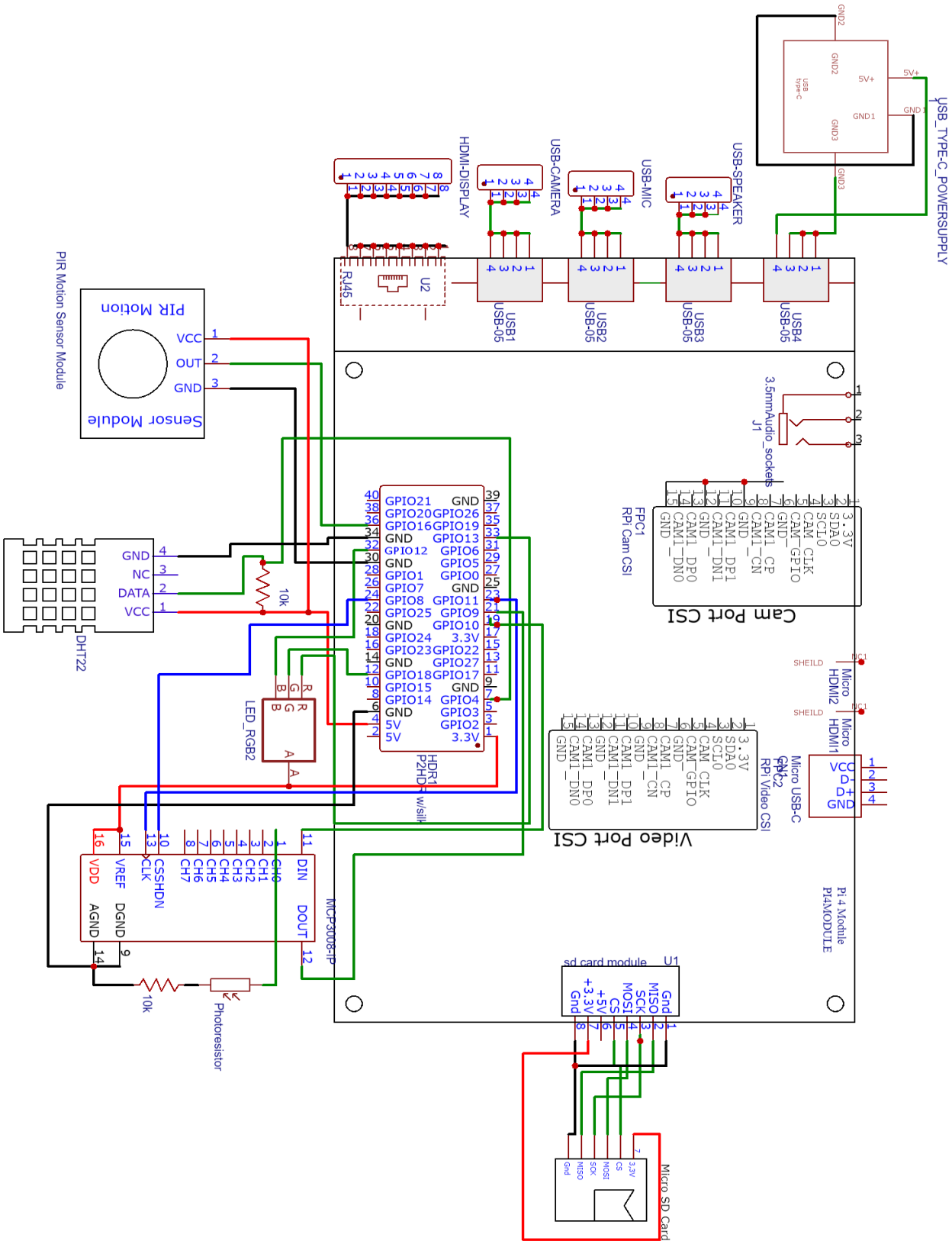


Figure 3.13 : Schematic diagram

3.6 Summary

In this chapter, we have discussed the system hardware and software components with their alternatives. The conceptual description of the system and the general flow of the system with all necessary diagrams are presented, too.

Chapter 4

Implementation

4.1 Preface

This chapter provides an overview of the software and hardware implementation, testing and validation, issues and challenges related to the implementation.

4.2 Hardware Implementation

This section describes the hardware components used in the project and their respective function Alities.

The first step was connecting electronic parts and components. As shown in Figure 4.1 below, The camera was connected to the Raspberry Pi 4 via a USB interface to enable stable real-time image transfer, Image Capture, Outfit Color Analysis, and Face Detection.

Figure 4.1: Webcam camera connected to Raspberry Pi 4.

Then we connected the brightness sensor (photoresistor) to the Raspberry Pi 4 along with the RGB LED strips, and 10K resistor to adjust lighting based on ambient conditions, as shown in Figure 4.2.

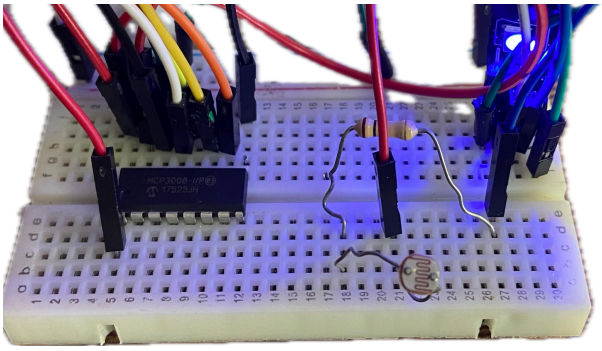


Figure 4.2: Photoresistor circuit

then connecting the DHT22 temperature and humidity sensor to the Raspberry Pi 4, we connect the DHT22 data pin with GPIO 4 and 10K resistor as shown in Figure 4.3.

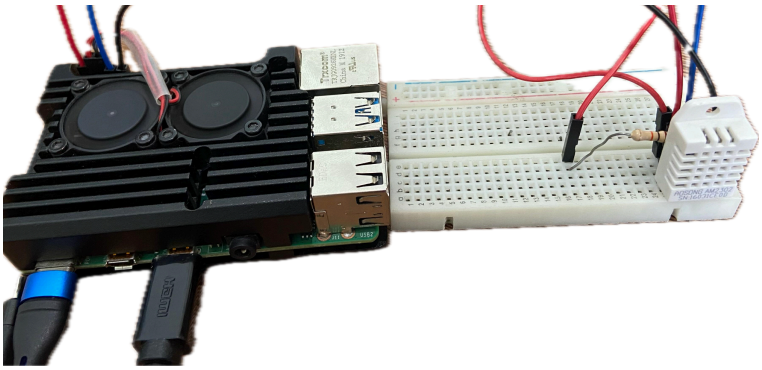


Figure 4.3: Temperature and Humidity Sensor circuit.

And connecting the PIR motion detection sensor pins (VCC, GND, data) with VCC, GND and GPIO 17, respectivly to the Raspberry Pi 4 as shown in Figure 4.4.

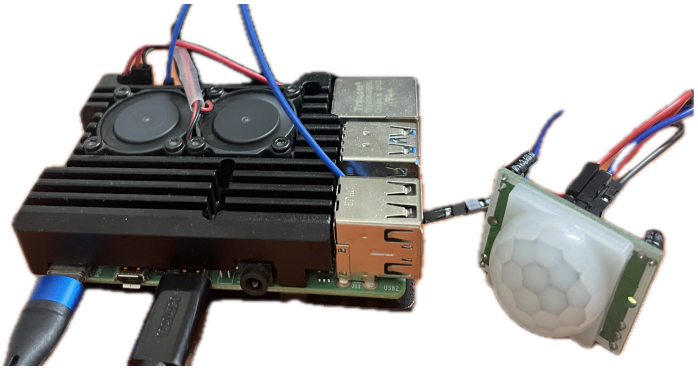


Figure 4.4: PIR Sensor with Raspberry Pi4

Additionally, we connect USB microphone and Speaker via USB port in Raspberry pi v4 as shown in Figure 4.5.

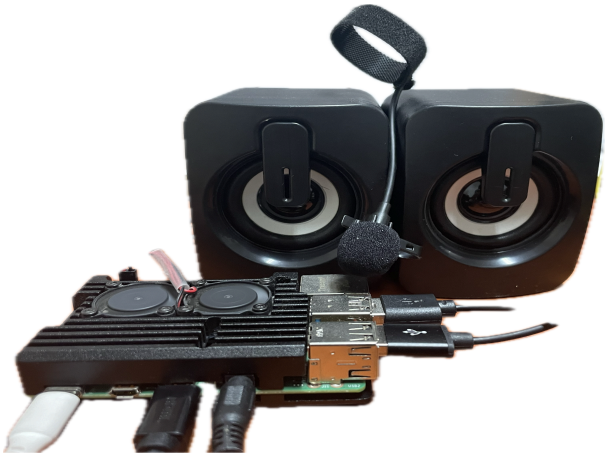


Figure 4.5: Microphone and speaker connected to Raspberry Pi 4.

The next step involved placing two layers of two-way mirror glass over the LCD display to achieve the reflective surface needed for the smart mirror. This setup allows the screen to remain visible behind the glass while also functioning as a regular mirror when the display is off. As shown in Figures 4.6 below.

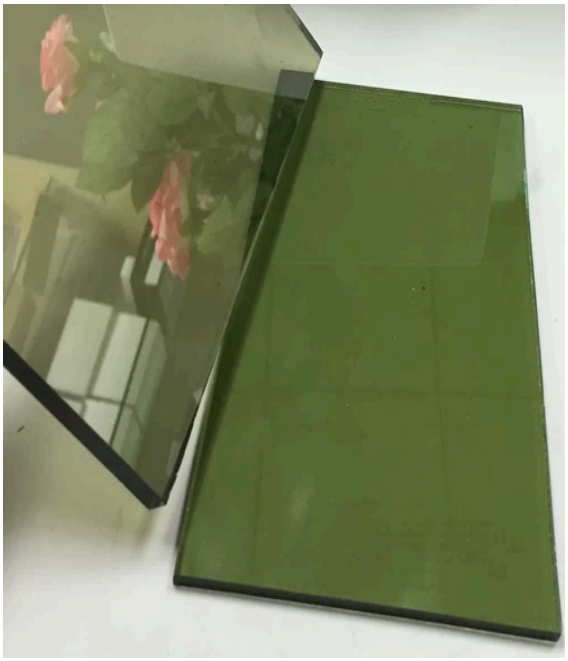


Figure 4.6: Two way mirror glass.

Next, a wooden frame was added around the LCD display to secure the structure and enhance the mirror's overall appearance, as shown in the figure 4.7 below the full shape.

Figure 4.7: The smart mirror with the wooden frame and two way glass

4.3 Software Implementation

4.3.1 Raspbian Operating System

We used Raspbian OS to ensure stability and compatibility with the Magic Mirror and Raspberry Pi 4.

4.3.1.1 Installing Raspbian operating system

We set up the Raspbian OS in our microSD card through a flash microSD card using Raspberry pi 4 Imager and then inserted the MicroSD card into our Raspberry Pi 4[57].

4.3.1.2 Installing MagicMirror

To install MagicMirror, you first need to download and install the required Node.js version (22.15.0) along with the npm version (10.9.2). Next, install Git, then clone the MagicMirror repository using the command git clone https://github.com/MagicMirrorOrg/MagicMirror. After cloning, navigate into the repository directory using cd MagicMirror, and install the application by running npm run install-mm. Once installation is complete, make a copy of the sample configuration file with cp config/config.js.sample config/config.js, and finally, start the application by executing npm run start.

The config.js file in MagicMirror is the main configuration file that controls what modules appear on the mirror and how they behave. It defines settings like language, time format, layout, and most importantly, the list of modules to load. Each module has its own configuration block where you can specify options such as position on the screen or update intervals. MagicMirror comes with default modules like clock, calendar, and weather, but we can also add custom modules we built below in Figure 4.8 a part of the code explains the modules used in the Magic mirror [58].

```
{
    module: "MMM-DHT22-Py",
    position: "bottom_center",
    config: {
      updateInterval: 60000
    }
},{
    module: "MMM-AssistantFeedback",
    position: "top_center"
},{
    module: 'MMM-Globe',
    position: 'middle_center',
    config: {
        style: 'fullBand',
        imageSize: 450,
        ownImagePath:'',
        updateInterval: 10*60*1000
    }
},
{
    module: "weather",
    position: "top_right",
    config: {
        weatherProvider: "openmeteo",
        type: "current",
        lat: 31.7683,
        lon: 35.2137
    }
},
```

Figure 4.8: The config.js file.

4.3.4 The Voice assistant

Here is a simple and clear procedure of how voice assistant work:

1. **Wake-Up and Activation**:

   The assistant stays in standby mode until it detects either motion and a face using the camera and PIR sensor, or hears the wake word "marvin".

2. **Speech Recognition**:

   Once activated, it uses the **Google Speech Recognition API** to listen and convert the user's voice into text.

3. **Command Handling**:

   Based on the recognized command, the assistant performs tasks such as:

   - Speaking the current time or date.
   - Getting room temperature (using DHT22 sensor and MagicMirror module).
   - Checking weather updates via an online API.
   - Searching for information using **Wikipedia**.
   - Analyzing outfit colors using the camera and **MobileNet-SSD model**.
   - Taking a picture using a separate script.

4. **Text-to-Speech Response**:

   The assistant replies using **Google Text-to-Speech (gTTS)**, and plays the audio response to the user.

5. **Visual Feedback and Display Control**:

   It simulates turning the screen on or off using a black full-screen window (using Pygame) when idle or listening.

6. **Logging and Error Handling**:

   All activities are logged to a file, and errors are caught and reported through logs or voice responses.

```python
def voice_assistant():

    # speak("Marvin voice assistant is ready. Say 'Marvin' to activate me.")
    try:
        fake_screen_off()
        while True:
            if wait_for_face():
                active = True
                while active:
                    user_input = listen()
                    if user_input:
                        active = handle_user_input(user_input)
                    else:
                        speak("I didn't catch that. Could you repeat?")
                speak("Marvin is now in standby mode.")
                fake_screen_off()
    except KeyboardInterrupt:
        speak("Voice assistant shutting down. Goodbye!")
    except Exception as e:
        logger.error(f"Unexpected error: {str(e)}")
        speak("I encountered an unexpected error and need to restart.")

# Start the assistant
voice_assistant()
```

Figure 4.9: The main function of the voice assistant.

```python
def handle_user_input(user_input):
    if "hello" in user_input or "hi" in user_input:
        speak("Hello! How can I help you today?")
    elif "how are you" in user_input:
        speak("I'm doing great, thank you for asking! How can I assist you?")
    elif "time" in user_input:
        now = datetime.datetime.now().strftime("%I:%M %p")
        speak(f"The time is {now}")
    elif "date" in user_input:
        today = datetime.datetime.now().strftime("%A, %B %d, %Y")
        speak(f"Today is {today}")
    elif "weather" in user_input:
        city = WEATHER_CITY
        words = user_input.split()
        for i, word in enumerate(words):
            if word == "in" and i < len(words) - 1:
                city = words[i + 1].capitalize()
        speak(get_weather(city))
```

Figure 4.10: The Voice assistant commands function.

4.3.5 The Outfit Color Ai Model

The **Outfit Color Detection Model** was developed in Python using OpenCV, NumPy, and scikit-learn within Visual Studio Code, as shown in **Figure 4.4 below**. This model is essential in the smart mirror system, detecting dominant top and bottom clothing colors and evaluating if the combination is fashionable.

To detect people in real time, it uses the **MobileNet-SSD** framework, integrated with deploy.prototxt (network architecture) and mobilenet_iter_73000.caffemodel (pretrained weights). Once a person is detected, the **region of interest (ROI)** is cropped and divided into upper and lower parts using custom coordinates. Each region is resized, blurred to reduce noise, and passed through a color extraction pipeline.

The **KMeans algorithm** identifies the dominant RGB color in each region, which is then converted to HSV for more accurate color naming. A classification function compares HSV values against defined thresholds to label colors like red, blue, green, black, white, etc. These color names are then used by a **suggestion engine**.

Fashion Suggestion Logic

The model uses a built-in color-matching function to check if the detected top and bottom colors harmonize based on fashion rules. If they match well, the user gets positive feedback; if not, better combinations are suggested.

Visual Output

Using OpenCV, the system displays the live camera feed with detected persons highlighted. It overlays color swatches, names of detected clothing colors, and personalized fashion advice, making the experience interactive and informative.

Here in Figure 4.11 some scripts for the model code, and the full code is on outfit color analyzer.py on github : https://github.com/dalabilal/MobileNet-SSD/tree/dalal .

```python
def detect_dominant_color_cv(image, k=4):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    pixels = image.reshape((-1, 3))
    kmeans = KMeans(n_clusters=k, n_init='auto',
    random_state=42)
    labels = kmeans.fit_predict(pixels)
    counts = np.bincount(labels)
    dominant_index = np.argmax(counts)
    dominant_color = kmeans.cluster_centers_[dominant_index]
    return dominant_color.astype(int)

def rgb_to_hsv_cv(rgb):
    r, g, b = [x / 255.0 for x in rgb]
    mx = max(r, g, b)
    mn = min(r, g, b)
    diff = mx - mn

    if diff == 0:
        h = 0
    elif mx == r:
        h = (60 * ((g - b) / diff) + 360) % 360
    elif mx == g:
        h = (60 * ((b - r) / diff) + 120) % 360
    else:
        h = (60 * ((r - g) / diff) + 240) % 360

    s = 0 if mx == 0 else diff / mx
    v = mx
    return h, s, v
```

Figure 4.11 : The outfit color model code.

4.3.5 Image Capture

The system captures an image using the webcam, uploads it to a temporary public URL via Ngrok, and generates a QR code for secure sharing. The QR code is displayed for 30 seconds, after which both the image and code are deleted, making the link invalid. A voice prompt guides the user through the process. Part of the code is shown in Figure 4.12 below.

```python
def start_server(port=8000):
    httpd.serve_forever()

def get_local_ip():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        s.connect(("8.8.8.8", 80))
        return s.getsockname()[0]
    finally:
        s.close()

def generate_qr(url, qr_filename):
    qr = qrcode.make(url)
    qr.save(qr_filename)
    print(f"📸 QR Code saved as {qr_filename}")
    img = cv2.imread(qr_filename)
    speak("Scan the QR code. You have 30 seconds.")
    cv2.imshow("Scan this QR Code", img)
    print(f"⏳ QR code will be visible for {display_duration} seconds...")
    cv2.waitKey(display_duration * 1000)
    cv2.destroyAllWindows()
```

Figure 4.12: Part of image capture code.

4.4 Implementation challenges & Issues

- An issue occurred during certain libraries not functioning properly on the Raspberry Pi This was resolved by updating the operating system and ensuring all dependencies were installed with compatible versions.
- The photoresistor sensor outputs analog values, while the RGB LED requires digital control. To handle this difference, we integrated an MCP3008 ADC chip to convert the analog signals into a format the Raspberry Pi can process digitally.
- To display the DHT22 sensor reading on the magic mirror there was a problem during the loading of the reading because there is an exception so we solved it by using the try-except block to handle read errors.
- We faced performance lag on the Raspberry Pi 4 when handling tasks like running the outfit analyzer, due to its limited processing power.

4.5 Summary

This chapter outlines the full implementation of the smart mirror system, covering both hardware and software aspects. Hardware components such as sensors, camera, microphone, and a two-way mirror were integrated with a Raspberry Pi 4 to create an interactive mirror setup. On the software side, Raspbian OS and the MagicMirror platform were installed to manage display modules, while a custom voice assistant and an AI-powered outfit color detection model were developed using Python, OpenCV, and scikit-learn. The system also faced and resolved several technical challenges, ensuring smooth operation and user interaction.

Chapter 5

Testing & Results

5.1 Preface

This chapter includes hardware and software testing, analysis, and discussion about the result and recommendations based on the result.

5.2 Validation and testing

In this section, we describe our hardware and software systems.

5.2.1 Hardware Testing

We successfully tested the smart mirror's ability to display real-time updates like time, date, weather, calendar events, and news update, using a magic mirror interface built on Raspberry Pi. The display updates in real-time via internet synchronization. As shown in Figure 5.1 below.

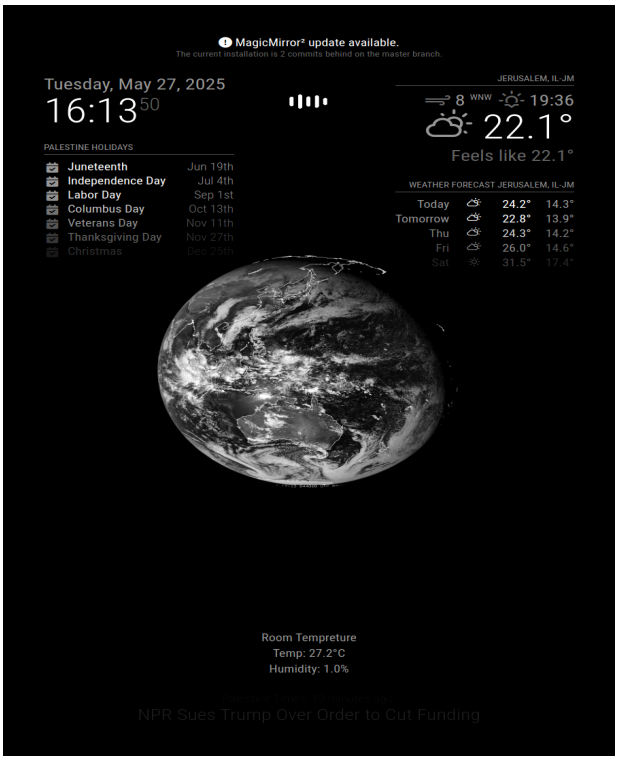


Figure 5.1: Magic Mirror interface

We conducted a combined test of the photoresistor sensor and the RGB light strip, as they work in tandem. The photoresistor detects the intensity of surrounding light, and the system adjusts the RGB light strip's brightness accordingly. Additionally, the RGB light strip was tested for its ability to display a full spectrum of colors, As shown in Figure 5.2, and Figure 5.3.

Figure 5.2 : RGB Led Strips in dark

Figure 5.3 : RGB Led Strips in light

We tested the Silver line USB Webcam for Image Capture, outfit Color Analysis, and Face Detection, as shown in Figure 5.4 , the camera is on the top of the mirror.

Figure 5.4 : USB Webcam placement on the mirror.

The face detection test, as shown in Figures 5.5, was performed by having a person stand in front of the smart mirror. The camera successfully detected his face using the face detection model.

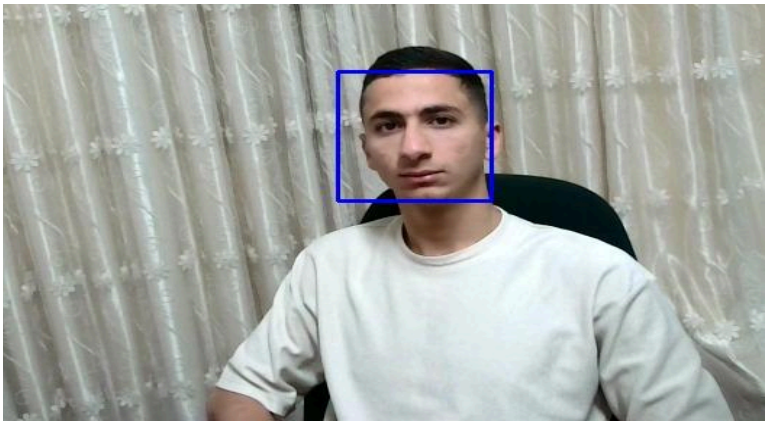


Figure 5.5 : Face detection testing on a person.

The outfit color detection model was tested using images of a person in various clothing combinations. It accurately identified the dominant colors of the upper and lower garments, such as red and white or black and white, with correct body region segmentation. as shown in the Figures.

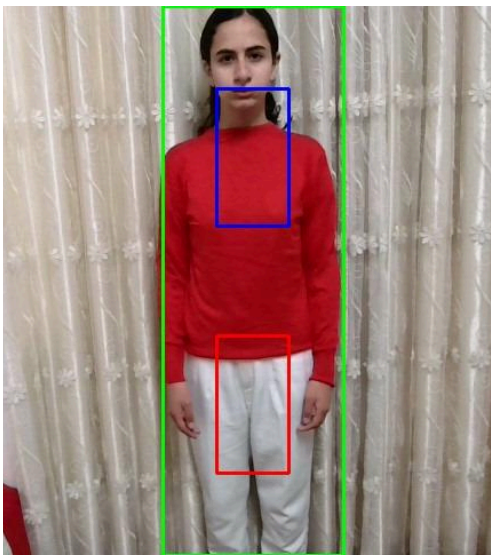


Figure 5.6: Outfit colors red and white.

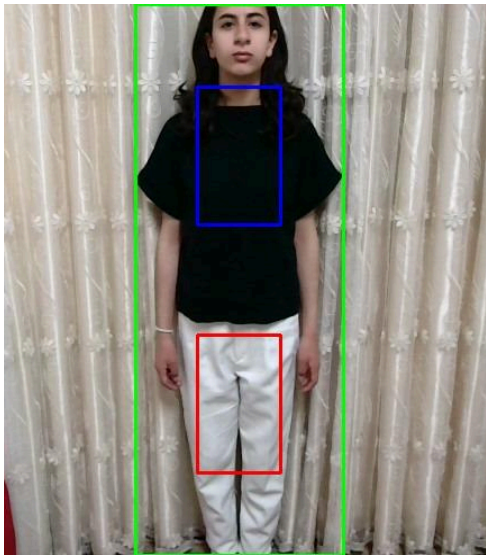


Figure 5.7: Outfit colors black and white.

First thing, the person said to the mirror "Take picture”, and it worked successfully as shown in Figure 5.8.

Figure 5.8: An image taken by the camera in Raspberry Pi.

Then the mirror provides QR code for the picture, as shown in Figure 5.9.

Figure 5.9: QR code for the taken picture.

Finally, it opens the HTTPS page to see the picture, and it expires in 30 seconds for security, as shown in Figure 5.10.

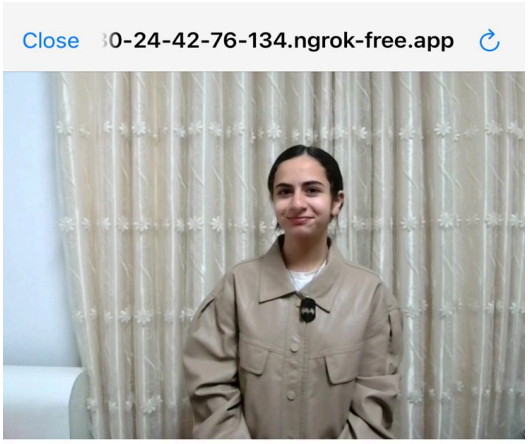


Figure 5.10: The Taken Picture on HTTPS.

The PIR sensor and face detection system were successfully tested. When motion and a face were detected, the mirror display turned on automatically, confirming accurate user presence detection, **as shown in Figure 5.11 and Figure 5.12**.

```
            time.sleep(1)

    except KeyboardInterrupt:
        print("Quitting...")
        GPIO.cleanup()

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  POR
No motion.
No motion.
No motion.
  Motion Detected!
  Motion Detected!
No motion.
No motion.
No motion.
No motion.
  Motion Detected!
  Motion Detected!
No motion.
No motion.
No motion.
No motion.
  Motion Detected!
  Motion Detected!
  Motion Detected!
  Motion Detected!
```

Figure 5.11: The PIR sensor test on VS code.

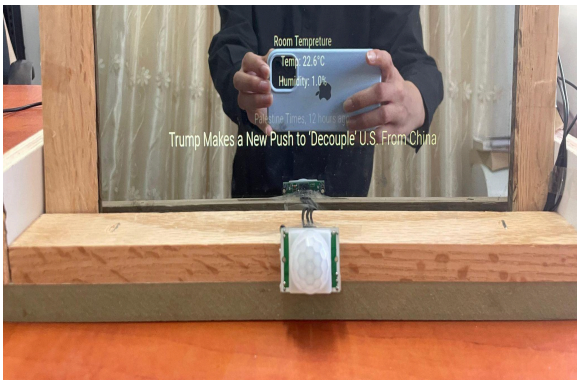


Figure 5.12: The PIR sensor on the mirror

A test was conducted to verify the integration of the microphone, camera, and speaker with the voice assistant and AI model. The system successfully captured voice commands, used the camera to take outfit images, analyzed them with the AI model, and provided feedback through the speaker. as shown in the Figures.

Figure 5.13: The microphone is placed in the smart mirror.

Figure 5.14: The speakers are placed in the back of the smart mirror.

Also we confirmed the testing by showing the listening and speaking signs as shown in Figure 5.15 and Figure 5.16 below.

Figure 5.15: The speaking status on the magic mirror.

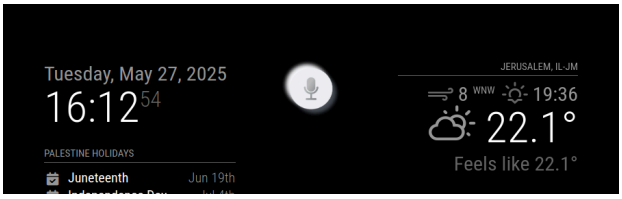


Figure 5.16: The listening status on the magic mirror.

The DHT22 sensor was tested to verify its ability to detect room temperature and humidity and display the readings on the smart mirror interface. The sensor performed accurately during all tests, with values correctly appearing in real time on the mirror screen. As shown in Figure 5.17.

Figure 5.17 : The Temperature and Humidity sensor readings on the magic mirror

5.2.2 Software Testing

During software testing, we ensured that all core features of the smart mirror system functioned reliably on the Raspberry Pi 4 without depending on external apps. Tests verified smooth interaction between local modules, including real-time display of time, date, weather, temperature, and humidity. The voice assistant was tested for its ability to receive commands, trigger actions like brightness control, and interact with the locally integrated AI model.

The AI model successfully analyzed outfit images captured by the mirror's camera and delivered recommendations via the speaker. Bluetooth functionality was also confirmed for transferring images to the AI model when needed. Additional tests showed effective coordination between sensors and the display, ensuring real-time data updates and stable performance in a standalone setup.

As shown in Table 5.1 and Table 5.2, some tests focused on the voice assistant and outfit color mode. A 5-second delay was required between commands to allow the voice assistant to complete processing.

Voice Command	**Input**	**Output**	**Numbe r of try**	**Pass/Fail**
Wake word	Marvin	Yes, how can I assist you?	10	8/10 the wifi was not stable and the second time we was too far
Hello	Hello	Hello! How can I help you today?	5	5/5 pass
How are you	How are you	I'm doing great, thank you for asking! How can I assist you?	5	3/5 we talk slow and it listen just “how”
Room temp	Room temperature	The room temp is … And the room humidity is ….	10	8/10 it takes just the word “room”.
Outdoor weather	weather	The weather is ….	5	3/5 unstable wifi connection
Time	Time	The time is 2:11 pm	7	5/7 it didn't catch the time word in both
date	date	Today is Tuesday, May 27, 2025	7	4/7 unstable wifi connection
look up	palestine	Palestine is an occupied state by israeli oocupation	5	3/5 it take the word “look”
Check my outfit	outfit	I'll analyze your outfit now	10	8/10 we didnt say outfit in the right way
goodbye	exit", "stop", "goodbye", "bye	Goodbye! Have a great day!	5	4/5

Table 5.1 : Voice assistant testing.

Colors	**Model reply**	**Pass/Fail**
Red and white.	Your shirt is red, and your pants is white. That's a great combination!	Pass

Black and white.	Your shirt is black, and your pants is white. This outfit is suitable for a business interview or meeting.	Pass
Blue and pink	Your shirt is blue, and your pants are pink. You can try blue with white, gray, khaki	Pass
Black and blue	Your shirt is black, and your pants are blue. You can try black with red, blue, white.	Pass
Orange and red	Your shirt is orange, and your pants are red. You can try orange with white, black.	Pass
Pink and blue	Your shirt is pink, and your pants are blue. You can try pink with gray, white, black.	Pass

Table 5.2: Outfit color model testing.

5.3 Result and Discussion

The project aimed to build a smart mirror that displays real-time data (time, weather, temperature, humidity) and performs outfit analysis using AI—all without needing a mobile or web app. Built around a Raspberry Pi 4, the system integrates sensors, a camera, RGB LEDs, and a voice assistant. It automatically activates upon detecting user presence, processes voice commands locally, and provides outfit feedback via the speaker. Environmental data is shown live on the mirror. The project successfully created a fully standalone, intelligent, and interactive mirror system.

Figure 5.18: Final view of the system

5.4 Summary

Chapter 5 presents the testing, validation, and analysis of the smart mirror system. Both hardware and software components were thoroughly tested to confirm proper integration and functionality.

Chapter 6

Conclusion and future work

6.1 Preface

The chapter introduces a summary of the project and future work.

6.2 Conclusion

The Smart Mirror System integrates computer vision, artificial intelligence, and voice interaction to offer a personalized, intelligent, and interactive user experience. Developed using Python and implemented in Visual Studio Code, the system utilizes multiple APIs and cloud-based services to deliver functionalities such as time, weather updates, voice-based queries, and outfit color analysis.

The mirror employs a camera for real-time image capture and runs an outfit color detection model that segments the top and bottom clothing regions, extracts dominant colors using clustering algorithms, and provides fashion recommendations based on color harmony. Simultaneously, a voice assistant—built using Python and executed via an online API—enables users to interact naturally with the mirror. All modules are integrated into a single user-friendly interface, enhancing daily routines such as checking the weather or choosing matching outfits.

This project demonstrates the powerful potential of combining AI, IoT, and UX design to create assistive technologies that are both functional and engaging. By delivering real-time insights and hands-free interaction, the smart mirror serves as a digital lifestyle companion in modern smart homes.

6.3 Future Work

1. **Offline Capability for Voice Assistant and Color Detection:**
   Implement on-device models and local voice recognition engines to allow full offline operation, increasing system robustness and data privacy.

2. **Facial Recognition and User Profiles:**
   Add user recognition features to tailor the mirror's responses and recommendations to individual preferences, schedules, or fashion styles.

3. **Emotion Detection and Adaptive UI:**
   Integrate facial emotion analysis to adjust responses, background themes, or motivational quotes based on the user's mood.

4. **Outfit Suggestions Based on Occasion and Weather:**
   Enhance the fashion recommendation model to factor in events (e.g., meetings, dates) and real-time weather data for smarter, context-aware clothing advice.

5. **Mobile App Synchronization:**
   Develop a companion mobile application for remote control, notification access, and personal dashboard synchronization.

6. **Modular Hardware Expansion:**
   Introduce modules such as fitness tracking sensors, ambient light sensors, or voice biometric security for expanded use cases and smarter interactions.

6.2 References

[1] Rabiner, L. R., & Juang, B. H. (1993). Fundamentals of Speech Recognition. Prentice Hall.

[2] Wikipedia, "Computer Vision," Wikipedia: The Free Encyclopedia, [Online]. Available: https://en.wikipedia.org/wiki/Computer_vision. [Accessed: Jan. 5, 2025].

[3] Wikipedia, "Machine Learning," Wikipedia: The Free Encyclopedia, [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning. [Accessed: Jan. 5, 2025].

[4] Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing (Special Publication 800-145). National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf [Accessed: Jan. 5, 2025].

[5] Google Cloud. (n.d.). Speech-to-Text: Automatic Speech Recognition. Retrieved from https://cloud.google.com/speech-to-text [Accessed Nov 29.2024]

[6] Smart Mirror Documentation. (n.d.). Smart Mirror: A voice-controlled life automation hub. Retrieved from https://docs.smart-mirror.io/ . [Accessed: Jan. 5, 2025]

[7] BYECOLD. (n.d.). Smart bathroom mirror with Bluetooth, Wi-Fi, LED lighting, anti-fog, and touch sensor (40” x 24”) [Amazon product listing]. Amazon. https://www.amazon.com/dp/B0CLLMY62M

[8] Smartwatch. (n.d.). In Wikipedia. Retrieved June 2, 2025, from https://en.wikipedia.org/wiki/Smartwatch

[Accessed: Jan. 5, 2025].

[9] Amazon, "Echo Show," https://www.amazon.com/echo-show. [Accessed: Jan. 5, 2025].

[10] Echo Show," Adapted from Amazon, "Echo Show," https://www.amazon.com/echo-show. [Accessed: Jan 3, 202].

[11] Raspberry Pi Foundation, "Raspberry Pi 4 Model B," https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. [Accessed: Dec 29, 2024]

[12] NVIDIA, "Jetson Nano Developer Kit," https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Accessed: Dec 29, 2024]

[Accessed: Jan. 5, 2025].

[13] BeagleBoard.org, "BeagleBone Black," https://beagleboard.org/black. [Accessed: Dec 29, 2024]

[14] Raspberry Pi 4," Adapted from Raspberry Pi Foundation, "Raspberry Pi 4 Model B," https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. [Accessed: Dec 29, 2024].

[15] Raspberry Pi 4," Adapted from Raspberry Pi Foundation, "Raspberry Pi 4 Model B," https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. [Accessed: Dec 29, 2024]

[16] Raspberry Pi Foundation, "Raspberry Pi Camera Module 3," https://www.raspberrypi.org/products/camera-module-3/.[Accessed: Dec 30, 2024]

[17] Raspberry Pi Foundation, "Raspberry Pi Camera Module 3," https://www.raspberrypi.org/products/camera-module-3/.[Accessed: Dec 30, 2024]

[18] Silver Line SLWC500 Full HD Widescreen Webcam. (n.d.). Future Store. Retrieved June 2, 2025, from https://futuremobile.co.il/product/silver-line-slwc500-full-hd-widescreen-webcam-with-auto-focus-sensor-and-built-in-microphone/

[19] Intel, "Intel RealSense Depth Camera D435," https://www.intelrealsense.com/depth-camera-d435/. [Accessed: Dec 30, 2024]

[20] Silver Line SLWC500 Full HD Widescreen Webcam. (n.d.). Future Store. Retrieved June 2, 2025, from https://futuremobile.co.il/product/silver-line-slwc500-full-hd-widescreen-webcam-with-auto-focus-sensor-and-built-in-microphone/

[21] Tutorials-RaspberryPi.com. (n.d.). How to use Photoresistors (Brightness / Light Sensor) with a Raspberry Pi. Retrieved June 2, 2025, from https://tutorials-raspberrypi.com/photoresistor-brightness-light-sensor-with-raspberry-pi/[Accessed: Dec 29, 2024]

[22] Tutorials-RaspberryPi.com. (n.d.). How to use Photoresistors (Brightness / Light Sensor) with a Raspberry Pi. Retrieved June 2, 2025, from https://tutorials-raspberrypi.com/photoresistor-brightness-light-sensor-with-raspberry-pi/[Accessed: Dec 29, 2024]

[23] Zhang, Li, et al., "Ambient Light Sensors for Dynamic Brightness Adjustment in Displays," IEEE Sensors Journal, 2019.

[24] Siepert, B. (2025, March 8). Adafruit BH1750 Ambient Light Sensor. Adafruit Learning System. Retrieved June 2, 2025, from https://learn.adafruit.com/adafruit-bh1750-ambient-light-sensor/overview .

[25] Tutorials-RaspberryPi.com. (n.d.). How to use Photoresistors (Brightness / Light Sensor) with a Raspberry Pi. Retrieved June 2, 2025, from https://tutorials-raspberrypi.com/photoresistor-brightness-light-sensor-with-raspberry-pi/[Accessed: Dec 29, 2024]

[26] Anusha, M., et al., "Performance Analysis of IoT Environmental Monitoring Using DHT Sensors," *IEEE Xplore*, 2021.

[27] M. Anusha, et al., *"Performance Analysis of IoT Environmental Monitoring Using DHT Sensors,"* IEEE Xplore, 2021. Available at: https://ieeexplore.ieee.org/document/9269861.[Accessed: Dec 31, 2024]

[28] S. Kumar, et al., "IoT-Based Smart Agriculture System Using DHT11 Sensors," Springer, 2020. DOI: 10.1007/978-3-030-41281-7.[Accessed: Dec 31, 2024]

[29] M. S. S. Perera, S. S. Samarasinghe, and A. Pasqual, "Temperature and Humidity Measurement Using an Off-the-Shelf MEMS Sensor: Beyond Arduino's Playground," IEEE Xplore, 2021. Available at: https://ieeexplore.ieee.org/document/9557262.[Accessed: Dec 31, 2024]

[30] Wikipedia contributors. (n.d.). Motion detector. Wikipedia. Retrieved June 2, 2025, from https://en.wikipedia.org/wiki/Motion_detector

[31] Solink. (2025, June 2). *What Is PIR on Security Cameras & Why Does It Matter?* Solink. Retrieved from https://solink.com/resources/industry-insights/what-is-pir-on-security-camera/

[32] MaxBotix Inc. (n.d.). *What Is an Ultrasonic Sensor?* Retrieved June 2, 2025, from https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work

[33] "Voice user interface," *Wikipedia*, https://en.wikipedia.org/wiki/Voice_user_interface

[34] All about the USB Microphone," *Mic & Mod*, https://www.micandmod.com/all-about-the-usb-microphone/

[35] Shure. (n.d.). MV88+ Stereo USB Microphone. Retrieved from https://www.shure.com/en-US/products/microphones/mv88plusstereo

[36]Shein. (n.d.). Product Page. Retrieved June 2, 2025, from http://api-shein.shein.com/h5/sharejump/appjump?link=lW3ndrFbrfH_8&localcountry=IL&url_from=GM74382549335

[37]Wikipedia contributors. (2025, May 27). *Smart speaker*. Wikipedia. Retrieved June 2, 2025, from https://en.wikipedia.org/wiki/Smart_speaker

[38]Shein. (n.d.). Product Page. Retrieved June 2, 2025, from http://api-shein.shein.com/h5/sharejump/appjump?link=lFJyqWcnuRL_8&localcountry=IL&url_from=GM74382549335

[39] Digital Gadgetry. (n.d.). Advancements in Bluetooth Speaker Technology: A Comprehensive Overview. Retrieved June 2, 2025, from https://digitalgadgetry.com/bluetooth-speaker-technology-advancements/

[40] Shein. (n.d.). *Product Page*. Retrieved June 2, 2025, from http://api-shein.shein.com/h5/sharejump/appjump?link=lFJyqWcnuRL_8&localcountry=IL&url_from=GM74382549335

[41] HP 24f 24-inch Display - Specifications. (n.d.). HP Support. Retrieved June 2, 2025, from https://support.hp.com/sg-en/document/ish_9848287-9848420-16

[42] HP. (n.d.). *HP P22vb G5 FHD Monitor*. Retrieved June 2, 2025, from https://www.hp.com/in-en/shop/hp-p22vb-g5-fhd-monitor-8l1l6a7.html

[43] LEDSupply. (n.d.). *RGB lighting: Guide to the top 5 RGB LED strips & lights*. LEDSupply Blog. https://www.ledsupply.com/blog/rgb-lighting-guide-to-the-top-5-rgb-led-strips-lights/

[44]Amazon.com.au. (n.d.). *RGB LED Strip WS2812B Individually Addressable Flexible Black PCB Dream Colour Strip IP65 Waterproof DIY Project with Control (DC 5V) for TV, Party, Christmas*. https://www.amazon.com.au/dp/B08Y8Q2GKR [Accessed: Jan 3, 2025]

[45] Raspberry Pi Foundation. (n.d.). Raspberry Pi 4 Model B specifications. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ [Accessed: Jan 3, 2025]

[46] Raspberry Pi Foundation. (n.d.). Power supply requirements for Raspberry Pi 4. Retrieved from https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/[Accessed: Dec 29, 2024]

[47] Wikipedia contributors. (n.d.). *Ubuntu*. Wikipedia. Retrieved June 2, 2025, from https://en.wikipedia.org/wiki/Ubuntu [Accessed: Jan 3, 2025]

[48] Raspberry Pi Foundation. (n.d.). *Raspberry Pi OS (formerly Raspbian) Documentation*. Retrieved from https://www.raspberrypi.org/documentation/ [Accessed: Jan 3, 2025]

[49]Arch Linux. (n.d.). *Arch Linux Wiki*. Retrieved from https://wiki.archlinux.org. [Accessed: Jan 3, 2025]

[50] Teeuw, M., & Contributors. (n.d.). MagicMirror²: The open source modular smart mirror platform. Retrieved June 2, 2025, from https://magicmirror.builders/

[51] Node.js Foundation. (2021). *Node.js Documentation*. Retrieved from https://nodejs.org/en/docs/ [Accessed: Jan 3, 2025]

[52] Microsoft Corporation. (2015). *Visual Studio Code: Documentation*. Retrieved from https://code.visualstudio.com/docs [Accessed: Jan 3, 2025]

[53]Python Software Foundation. (n.d.). *Welcome to Python.org*. https://www.python.org/ [Accessed: Jan 3, 2025]

[54]Roy, Sujan. n.d. *Real-Time Object Detection with MobileNet and SSD*. GitHub. Accessed June 2, 2025. https://github.com/Sujan-Roy/Real-Time-Object-detection-with-MobileNet-and-SSD.

[Accessed: May 3, 2025]

[55] Wikipedia contributors. n.d. "K-means Clustering." Wikipedia. [Accessed May 2, 2025]. https://en.wikipedia.org/wiki/K-means_clustering.

[56] "HSV color space." Wikipedia, Wikimedia Foundation. https://en.wikipedia.org/wiki/HSL_and_HSV [Accessed may 2, 2025]

[57]*Getting Started with Your Raspberry Pi*. Raspberry Pi Foundation, https://www.raspberrypi.com/documentation/computers/getting-started.html. [Accessed 2 May 2025].

[58] *Getting Started: Installation*. MagicMirror² Documentation, https://docs.magicmirror.builders/getting-started/installation.html. [Accessed: May 3, 2025]