Abstract

This project aims to create a smart doorbell system that seamlessly combines hardware and software for improved home security and user convenience. The system features a camera, the camera module captures live video footage. This video feed will be accessible through a web interface built using Flask, allowing users to monitor their doorstep remotely. Whenever a person is detected or the doorbell is pressed the camera lets the owner know if the person is unknown or known. Furthermore, it captures the picture of the person if the person is unknown. This project highlights the integration of IoT technology with real-time monitoring and communication systems.

Chapter1

Introduction

The Smart Doorbell System is designed to enhance home security and convenience by integrating IoT components such as a camera module, and a Open CV python libraries. Smart doorbells play a vital role in modern home automation by providing real-time monitoring and secure access control. This project combines hardware for visitor detection and live video streaming with a Flask-based web interface that enables users to remotely monitor their doorstep. This system demonstrates the effective integration of IoT technology for smart home applications.

1.1 Objectives :

1. Develop a smart doorbell system that integrates visitor detection, video streaming, and secure access control.
2. Implement a camera module to stream real-time video to a web-based interface.
3. Create a Flask-based web interface for users to monitor the live feed and manage access permissions.

1.2 Problem Statement:

1. Traditional doorbell systems do not provide real-time video streaming for identifying visitors, especially when the user is away.
2. There is no mechanism to notify homeowners instantly about visitor activity, leading to delayed responses.
3. Existing systems lack advanced features like motion detection and secure access control for authorized individuals.
4. Limited integration of IoT components, such as cameras, sensors, and web interfaces, restricts the functionality of conventional doorbell systems.
5. There is a need for a cost-effective and user-friendly smart doorbell solution that enhances security and convenience for modern homes.

Chapter 2

Applications

The Smart Doorbell System using a camera has a variety of real-world applications, such as:

1. **Home Security:** Enhances security by providing real-time video surveillance and notifications of visitor activity.
2. **Remote Monitoring:** Allows homeowners to monitor their doorsteps from anywhere, ensuring peace of mind while away.
3. **Access Control:** Enables secure access for authorized individuals
4. **Smart Communities:** Facilitates advanced security solutions in residential complexes by integrating with centralized monitoring systems.
5. **Elderly and Disabled Assistance:** Offers convenience for individuals with mobility challenges by allowing remote door access.
6. **Educational Projects:** Serves as an example of integrating IoT components, video streaming, and automation for real-world problem-solving.

The system described offers several benefits, including enhanced home security with real-time video surveillance and notifications. It also provides remote monitoring, allowing homeowners to oversee their doorstep activity from anywhere. Additionally, it facilitates secure access control for authorized individuals and supports smart community integrations for improved residential security. This system is also helpful for individuals with mobility challenges, offering remote door access, and serves as a valuable educational tool for integrating IoT, video streaming, and automation in real-world applications.

Chapter 3

Components Used

3.1 Hardware Components:

1. **Raspberry Pi:** Serves as the main controller for processing sensor inputs, streaming video, and handling notifications.

The Raspberry Pi 4 is a compact, versatile single-board computer designed for a wide range of applications, from educational projects to industrial and IoT systems. Below are its key features:

- **Processor**: It is powered by a Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC, clocked at 1.5 GHz, providing significantly improved performance compared to its predecessors.
- **Memory Options**: Available with 2 GB, 4 GB, 8 GB, or 1 GB of LPDDR4 SDRAM, catering to varying performance needs.
- **Connectivity**: Includes dual-band 802.11ac Wi-Fi, Bluetooth 5.0, Gigabit Ethernet, and two USB 3.0 ports alongside two USB 2.0 ports.
- **Display Support**: Features dual micro-HDMI ports, supporting up to 4K resolution at 60 Hz, making it ideal for multimedia and graphical projects.
- **Storage**: Offers a microSD card slot for primary storage and supports booting from USB-attached drives.
- **Expansion**: Equipped with a 40-pin GPIO header for connecting hardware components and peripherals.
- **Power Input**: Powered via a USB-C connector, it requires a 5V/3A power supply.

The Raspberry Pi 4 is widely used for home automation, retro gaming, coding education, media centers, and robotics projects, thanks to its flexibility, affordability, and rich ecosystem of accessories and software support. Let me know if you'd like more details about its capabilities!

2. **Camera Module:** Captures real-time video footage for remote monitoring via the web interface. Aweb camera (webcam) is a digital video camera commonly used for live video streaming, video conferencing, and surveillance. It connects to devices via USB or wirelessly and often includes a built-in microphone for audio capture. Modern webcams support resolutions from HD (720p) to 4K, with adjustable fields of view for different applications. Webcams are widely used in various fields, including communication, security, and IoT projects.

3. **Buzzer:** Emits a sound when the doorbell is pressed or unauthorized access is attempted. A

buzzer is an electronic component that generates sound when powered, often used for signaling and alerts in various devices. It can be classified into active and passive types: active buzzers have an internal oscillator, producing sound with just a DC voltage input, while passive buzzers require an external signal, like a square wave, to produce a tone.

4. **LED Indicator:** Signals system status, such as visitor detection or successful access authorization.

5. **Jumper Wires:** Connects all components for seamless integration.
6. **Power Supply:** Provides power to the Raspberry Pi and other components.

3.2 Software Components:

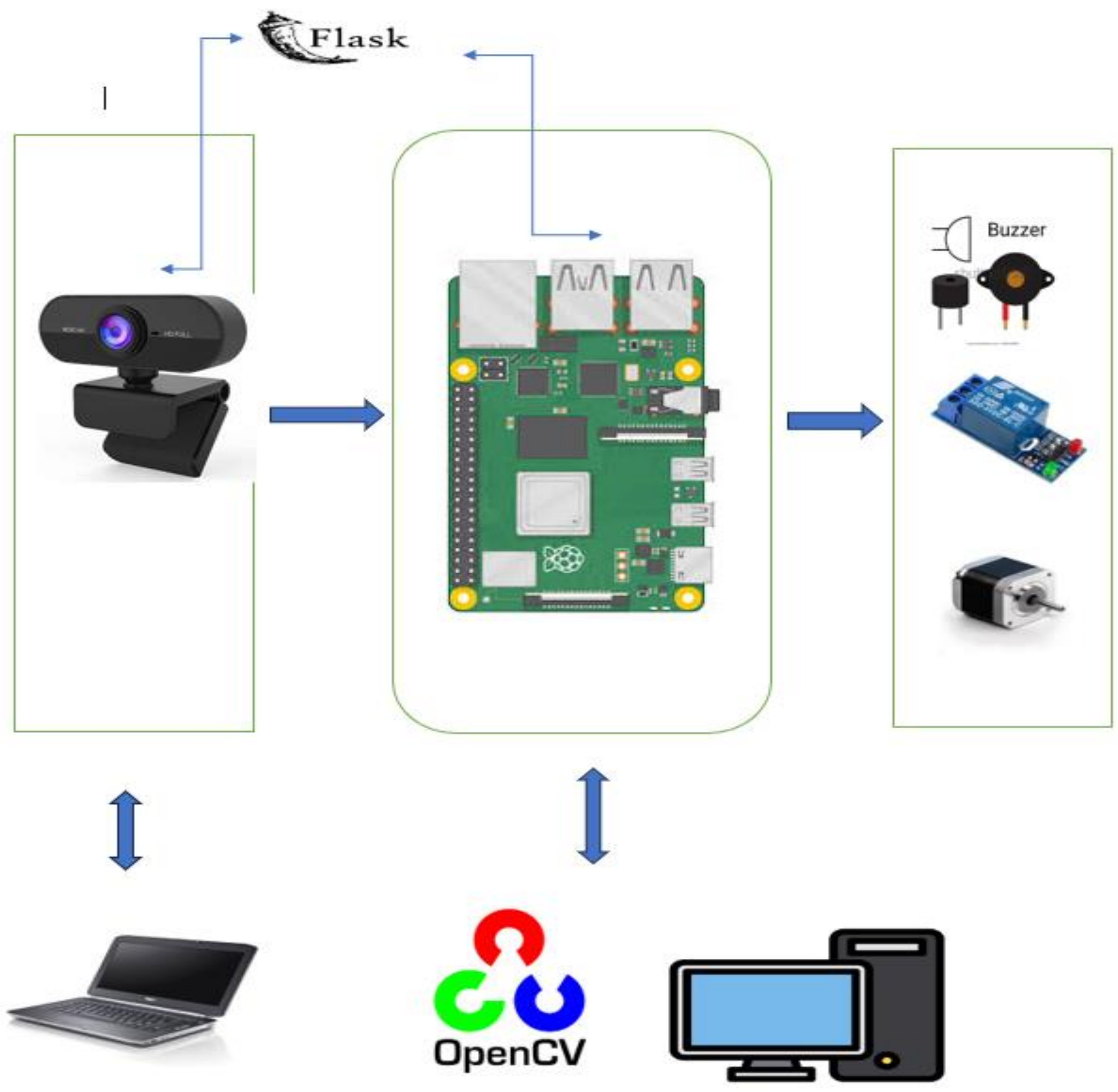
1. **Python:** The primary programming language used to control hardware components and implement system logic.
2. **Flask:** A web framework used to develop the interface for streaming live video and managing notifications.
3. **OpenCV:** Library for processing video streams and integrating camera functionalities.

OpenCV (Open Source Computer Vision Library) is a powerful, open-source library used for real-time computer vision and image processing. It provides a wide range of functions for image manipulation, such as filtering, edge detection, and geometric transformations like scaling and rotation. OpenCV also supports complex tasks like face detection, object recognition, and motion analysis, making it popular for applications in security, robotics, and healthcare. The library is highly efficient and can process images and video in real-time, which is ideal for use cases such as surveillance systems and autonomous vehicles.OpenCV supports multiple programming languages, including C++, Python, and Java, allowing developers to work with their preferred tools. It also includes machine learning algorithms for classification and regression tasks, offering further versatility in various domains like AI and automation.

4. **Raspbian OS:** The operating system for Raspberry Pi to run the application. Raspberry Pi OS (formerly Raspbian) is the official operating system for Raspberry Pi devices, based on Debian Linux. It is optimized for the Raspberry Pi hardware, ensuring smooth performance and full compatibility with the Pi’s components. Raspberry Pi OS comes pre-installed with a wide variety of software, including a desktop environment, programming tools like Python, and utilities for hardware interfacing, making it ideal for both beginners and advanced users. The OS supports various programming languages, including Python, Scratch, and Java, and offers robust community support and documentation.It provides a lightweight and user-friendly interface while allowing access to the Linux terminal for more advanced tasks. Raspberry Pi OS is continually updated with security patches, new features, and improved hardware support, making it a reliable choice for Raspberry Pi projects. It also has versions tailored to specific needs, such as a minimal installation for headless systems or the full desktop version for a graphical user interface.

Chapter 4

Flow Chart



The diagram outlines the flow of a face recognition-based system, likely for security or access control. It begins with requesting face recognition, followed by determining whether the face is recognized as known or unknown. If a known face is detected, the system activates a relay, allowing access or triggering a specific action. If the face is unrecognized, a buzzer is triggered, and the system captures an image of the unknown individual, then notifies that the recognition is invalid. The system may then reset and conclude the process.

The flowchart represents the operational process of a smart doorbell system using face recognition.

1. **Start**: The system begins by requesting face recognition when an event (e.g., button press) occurs.
2. **Face Recognition Check**: It determines if the detected face matches an authorized one.
3. **Known Face**: If the face is recognized as authorized, the relay is activated to unlock the door.
4. **System Reset**: After unlocking, the system resets, preparing for the next interaction.
5. **Unknown Face**: If the face is not recognized, a buzzer is activated as an alert.

6. **Capture Unknown Face**: The system saves an image of the unrecognized face for future reference or logging.
7. **Invalid Notification**: The system notifies that the face is invalid, completing the alert process before resetting or awaiting further inputs.

Conclusion

The Smart Doorbell System was successfully developed to enhance home security and convenience. The system effectively utilized a camera module to provided real-time video streaming accessible through a Flask-based web interface and detects the visitors. The authorized gets the access and if unknown is detected then the buzzer turns on. The seamless integration of hardware and software components demonstrated the potential of IoT technology in building smart home solutions, offering a reliable and user-friendly system for modern households.

Future Work

The following improvements can be implemented in future versions of the Smart Doorbell System:

1. **Mobile Application Integration:** Develop an Android or iOS app for real-time visitor monitoring and system management.
2. **Cloud Storage:** Enable video and access log storage on the cloud for future reference and security audits.
3. **Facial Recognition:** Integrate AI-based facial recognition to identify and differentiate between known and unknown visitors.
4. **Voice Communication:** Add two-way audio functionality to enable communication between the user and the visitor.
5. **Battery Backup:** Include a backup power system to ensure continuous operation during power outages.
6. **Multi-Door Support:** Extend the system to manage multiple doors or entry points in a building.
7. **Advanced Notifications:** Implement SMS and email alerts for additional communication options.
8. **Integration with Smart Home Systems:** Allow compatibility with platforms like Alexa, Google Assistant, or Apple HomeKit for enhanced automation.

Appendix

A. Hardware Connections:*

- *Camera Module:* Connected to Raspberry Pi Camera Interface (CSI) port.

- *Button:* Connected to Raspberry Pi GPIO Pin 17.

- *Speaker/Buzzer:* Connected to GPIO Pin 18.

- *LED:* Connected to GPIO Pin 23 (to indicate doorbell status).

- *Raspberry Pi:* Used for processing and handling inputs/outputs, including the camera, GPIOs, and networking.

B. Code Snippets:

1. Initialize GPIO and Camera:

```
python
import RPi.GPIO as GPIO
from picamera import PiCamera

def init_gpio():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(button_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.setup(led_pin, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(buzzer_pin, GPIO.OUT, initial=GPIO.LOW)
    camera = PiCamera()
    camera.resolution = (1024, 768)
    return camera
```

2. Capture Image when Unknown person detected.

```
 python
import time
import datetime
```

```
    if save_unknown_face(frame):
  """Save the frame with unknown face as an image file."""
  timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
  filename = os.path.join(UNKNOWN_FACES_DIR, f"unknown_face_{timestamp}.jpg")
  cv2.imwrite(filename, frame)
```

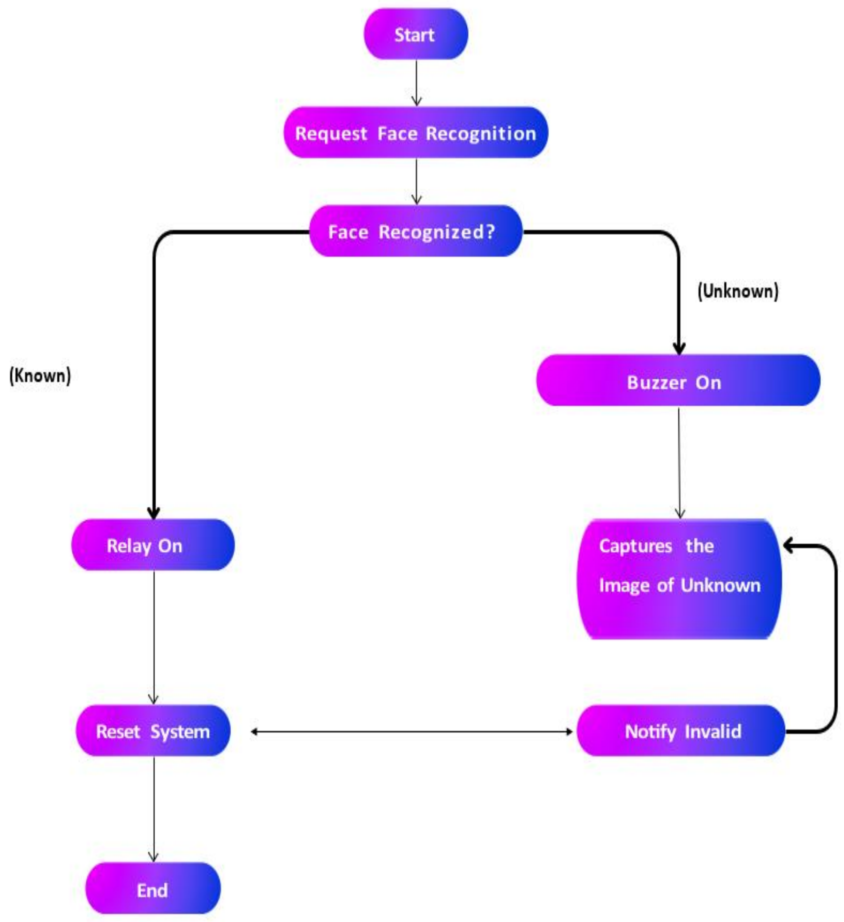
3. *Sound Alert (Buzzer) on Button Press:*

```
python
def activate_buzzer():
    GPIO.output(buzzer_pin, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(buzzer_pin, GPIO.LOW)
```

4. If Known person detected on relay:

```
try:
 while True:
 GPIO.input(SWITCH1) == GPIO.HIGH:
print("Face unlock initiated...")
     response = send_face_unlock_request()
    if response.get("status") == "SUCCESS":
     print("KNOWN PERSON ")
      trigger_relay()
      attempts = 0
```

Psuedo Code


Start
Request Face Recognition
Face Recognized?
(Known)
(Unknown)
Buzzer On
Relay On
Captures the Image of Unknown
Reset System
Notify Invalid
End

References

[1] Enhanced Smart Doorbell System Based on Face Recognition, IEEE Conference Publication, 2016. Available: https://ieeexplore.ieee.org/document/7505106.

[2] Smart Home Security Using IoT and Face Recognition, IEEE Conference Publication, 2019. Available: https://ieeexplore.ieee.org/document/8697695.

[3] An Iris+Voice Recognition System for a Smart Doorbell, IEEE Conference Publication, 2019. Available: https://ieeexplore.ieee.org/document/8760187.