GATE SECURITY SYSTEM

A Mini Project Report Submitted to the APJ Abdul Kalam Technological University in partial fulfillment of requirements for the award of degree

Bachelor of Technology

in

Electronics and Communication Engineering

by

SANA C P (NSS22EC099)
SARIGA R (NSS2EC104)
SHARAN KRISHNAN M (NSS22EC109)
SUHAIR K (NSS22EC118)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
NSS COLLEGE OF ENGINEERING PALAKKAD
KERALA
APRIL 2025

DECLARATION

We hereby declare that the project report **GATE SECURITY SYSTEM**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us the under supervision of Prasad R Menon, Assistant Professor, Department of Electronics and Communication Engineering, NSS College of Engineering, Palakkad.

This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Palakkad
2-04-2025

SANA C P
SARIGA R
SHARAN KRISHNAN M
SUHAIR K

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NSS COLLEGE OF ENGINEERING

2024-2025

CERTIFICATE

This is to certify that the report entitled **GATE SECURITY SYSTEM** submitted by **SANA C P(NSS22EC099),SARIGA R(NSS22EC104) ,SHARAN KRISHNAN M(NSS22EC109)** and **SUHAIR K(NSS22EC118)** to the APJ Abdul Kalam Technological University in partial fulfillment of the B.Tech. degree in Electronics and Communication Engineering is a bonafide record of the mini-project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Asst. Prof. Prasad R Menon	**Dr. Binu G S**	**Dr. Sreeleja N Unnithan**
(Project Guide)	(Project Coordinator)	(Head of Department)
Assistant Professor	Professor	Associate Professor
Dept. of ECE	Dept. of ECE	Dept. of ECE
NSS College of Engineering	NSS College of Engineering	NSS College of Engineering
Palakkad	Palakkad	Palakkad

Contents

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project guide Prasad R Menon , Assistant Professor, Department of Electronics and Communication Engineering, NSS College of Engineering, Palakkad for the constant support, mentorship and co-operation.
We extend our sincere gratitude to our project coordinator Dr.Binu G S, Professor, Department of Electronics and Communication Engineering, NSS College of Engineering, Palakkad for the unwavering support.
We express our sincere thanks to Head of the Department, Department of Electronics and Communication Engineering, NSS College of Engineering, Palakkad for providing us with all the necessary facilities and support.
We sincerely appreciate to Principal, NSS College of Engineering, Palakkad, for his encouragement and inspiration.
Last, but not the least, we take pleasant privilege in expressing our heartfull thanks to our teachers, friends and all well-wishers who were of precious help in completing the miniproject.

SANA C P
SARIGA R
SHARAN KRISHNAN M
SUHAIR K

ABSTRACT

Security is a crucial aspect of modern residential and commercial premises. This project presents a Gate Security System designed using Raspberry Pi 4B to provide secure and efficient access control.The system integrates face recognition, keypad-based authentication, and remote access via the Blynk app to enhance security.

When a person approaches the gate, they can choose between multiple authentication methods. Face recognition is triggered by pressing a designated push button, allowing the system to verify the individual against a pre-stored dataset. If the face is recognized, the servo motor operates to open the gate. If unrecognized, the gate remains closed. Alternatively, authorized users can enter a PIN code via a keypad. A valid PIN grants access, while an incorrect entry denies it. Visitors who are complete strangers can use another push button to ring a bell, notifying the owner. The owner can then monitor the live video feed via the Blynk app and decide whether to allow access.

For enhanced security, the system includes an alert mechanism using a buzzer: a short beep when the gate opens and a longer beep when it closes. The entire setup leverages components including a Pi Camera Module, push buttons, power supply, and breadboard, ensuring a robust and effective security solution. This smart Gate Security System provides an automated, multi-authentication access mechanism while enabling remote monitoring, making it ideal for homes, offices, and gated communities.

List of Figures

Chapter 1

INTRODUCTION

The Gate Security System is designed to enhance access control through multiple authentication modes: Face Recognition, Keypad Authentication, and Visitor Notification. Upon approaching the gate, an individual can initiate face recognition via a push button. If their identity matches the stored dataset, access is granted. In case of unrecognized faces, access can be obtained by entering a predefined PIN code on the keypad. Unauthorized individuals can trigger a visitor notification system by pressing a dedicated push button, which activates a doorbell and alerts the owner via the Blynk app with live video streaming for remote monitoring and decision-making. To enhance security, a short buzzer sound is generated when the gate opens, while a long buzzer sound indicates gate closure, ensuring a reliable and automated access management system.
The following steps outline the development of the Gate Security System:

- Literature Survey: Research existing security systems and technologies related to automated gate control and IoT-based security.
- Data Collection: Gather face recognition datasets and authentication methods for secure access control.
- System Algorithm Development: Develop an algorithm for face recognition, keypad authentication, and remote access control using IoT integration.
- Validation and Testing: Test and validate the gate security system under different conditions to ensure accuracy and reliability.
- Deployment: Implement the Gate Security System prototype, evaluate its effectiveness, and refine its components and algorithms for improved performance.

Chapter 2

LITERATURE REVIEW

R. Yu and M. Zhang [1] on Smart Home Security Analysis System Based on The Internet of Things explores IoT-driven security solutions, emphasizing real-time intrusion detection, automated alerts, and data encryption for smart home environments. The research outlines how smart surveillance cameras, motion sensors, and alarm systems work together to create a multi-layered security approach, providing proactive protection against security breaches. This directly relates to our Gate Security System, which integrates a 4x4 keypad, a buzzer, and an LCD display to enhance authentication and alert mechanisms. The study also highlights the importance of mobile-based control and remote monitoring, which aligns with our use of the Blynk app for remote gate operation, live video feed access, and intrusion alerts. Additionally, the paper discusses energy-efficient IoT security implementations, which resonate with our project's use of low-power components like Raspberry Pi 4B and ESP32, ensuring an optimized cost-effective and power-efficient security solution.

Furthermore Y. Xiao, Y. Jia, C. Liu, A. Alrawais, M. Rekik, and Z. Shan [2] on HomeShield: A Credential-Less Authentication Framework for Smart Home Systems explores an innovative authentication framework that eliminates traditional credentials like passwords and PINs, instead utilizing biometric authentication and device-based verification for enhanced security in smart home systems. The paper highlights how IoT-based security frameworks can improve access control, remote monitoring, and automated authentication, ensuring secure and seamless entry management. The HomeShield system leverages secure communication protocols, encrypted data transfer, and multi-layer authentication mechanisms to prevent unauthorized access. These aspects align closely with our Gate Security System, which also incorporates facial recognition via a Pi Camera for credential-less authentication. Similar to the HomeShield framework, our project integrates real-time monitoring and remote access control through the Blynk app, allowing homeowners to receive alerts, grant or deny access, and monitor entry logs from anywhere. Furthermore, the paper discusses low-latency authentication, which is reflected in our system through raspberry pi's real-time facial recognition processing and automated gate operation using a servo motor, ensuring quick and secure access management.

The insights from these studies [1-2] emphasize IoT-based authentication, real-time security management, and remote access control, which form the foundation of our Gate Security System. By integrating biometric authentication, automated gate operation, and remote access monitoring, our project enhances residential and institutional security, offering a seamless, intelligent, and efficient entry control solution.

2.1 BLOCK DIAGRAM

The block diagram represents the overall architecture and working of the Gate Security System. The system is designed to allow authorized users to access the gate using either face recognition or a keypad-based PIN code while providing a notification mechanism for visitors.The block diagram of the system is given below as figure 2.1.1

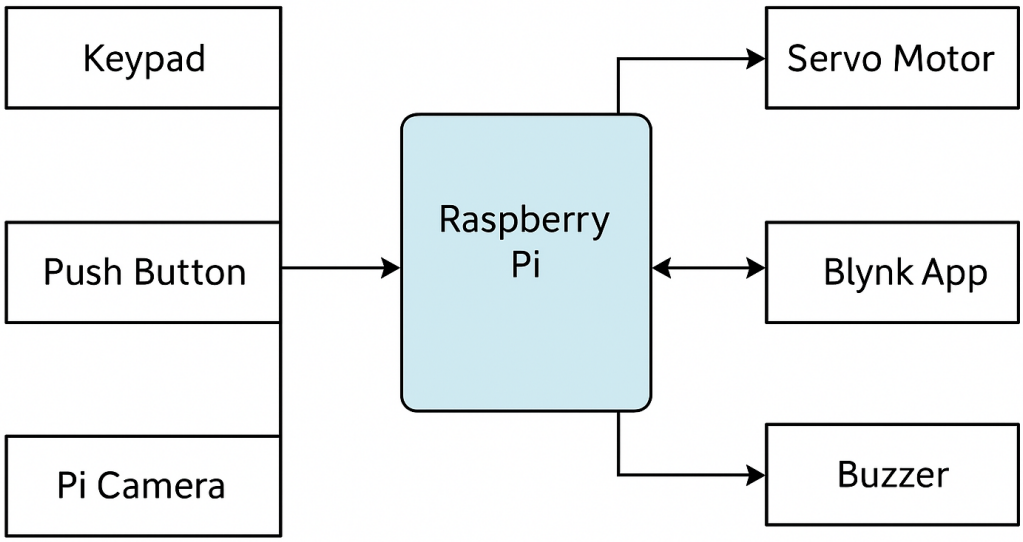

Figure 2.1.1: Block Diagram Of Gate Security system

2.2 THE PROPOSED WORKING MODEL

The proposed Gate Security System is an advanced access control solution that integrates facial recognition, keypad authentication, and remote monitoring to enhance security and convenience. The system is powered by a Raspberry Pi 4B, which acts as the central controller, processing inputs from various sensors and modules. When an individual arrives at the gate, they have three authentication options. The primary method is facial recognition, where pressing a push button activates the Pi Camera Module, capturing the visitor's image and comparing it against a pre-stored dataset. If the face is recognized, the servo motor is triggered, opening the gate and granting access. If the face is not recognized, the person can opt for the keypad authentication method, where they must enter a PIN code. The Raspberry Pi verifies the entered code, and if valid, it signals the servo motor to open the gate. If an incorrect code is entered, access is denied, ensuring unauthorized individuals cannot bypass the system. For visitors who are not registered, an alternative push button is available to ring a bell, alerting the owner. Simultaneously, a notification is sent via the Blynk App, allowing the owner to remotely monitor the visitor through a live video stream and decide whether to grant access manually. To enhance security feedback, a buzzer system is integrated, emitting a short beep whenever the gate opens and a long beep when it closes. The system operates efficiently with a stable power supply, and all components, including the breadboard, connections, and push buttons, are designed for seamless operation. This automated gate security solution ensures a robust, multi-layered authentication system, integrating real-time monitoring, user verification, and remote access control, making it highly suitable for both residential and commercial applications.

Chapter 3

COMPONENTS USED

3.1 RASPBERRY PI

The Raspberry Pi 4B is a compact and powerful single-board computer that serves as the central controller in the Gate Security System. It processes inputs from the Pi Camera Module, Keypad, and Push Buttons, enabling face recognition, PIN verification, and remote access control. The Pi Camera Module captures images for authentication, and if the face matches stored data, the servo motor opens the gate. If unrecognized, access can be granted by entering a valid PIN via the keypad; otherwise, the system denies entry. For unknown visitors, a push button triggers a notification to the owner via the Blynk app, allowing remote monitoring through live video streaming. Additionally, the buzzer provides short and long alarms during gate operations, ensuring security and alerting residents. With IoT integration, the Raspberry Pi 4B efficiently automates and enhances gate security.The pin diagram of Raspberry Pi 4B is given below as figure 3.2.1 .

3.2 Pi CAMERA MODULE

The Pi Camera Module is a high-resolution camera designed for Raspberry Pi boards, enabling image capture and video recording. It connects via the CSI (Camera Serial Interface) port on the Raspberry Pi and offers superior image processing capabilities. The module supports various resolutions, frame rates, and features like autofocus and low-light enhancement, making it ideal for applications such as surveillance, face recognition, and IoT-based monitoring. In this Gate Security System, the Pi Camera plays a crucial role in face recognition by capturing real-time images of visitors. When a person approaches the gate, the camera takes a snapshot and sends it to the Raspberry Pi for image processing and comparison against a stored dataset. If the face

matches an authorized entry, the system triggers the gate to open. Otherwise, an alert is sent to the owner via the Blynk app, allowing remote monitoring through live video streaming. This ensures enhanced security by providing both automated access control and manual intervention capabilities.The figure of Pi camera module is given below as figure 3.2.2 .

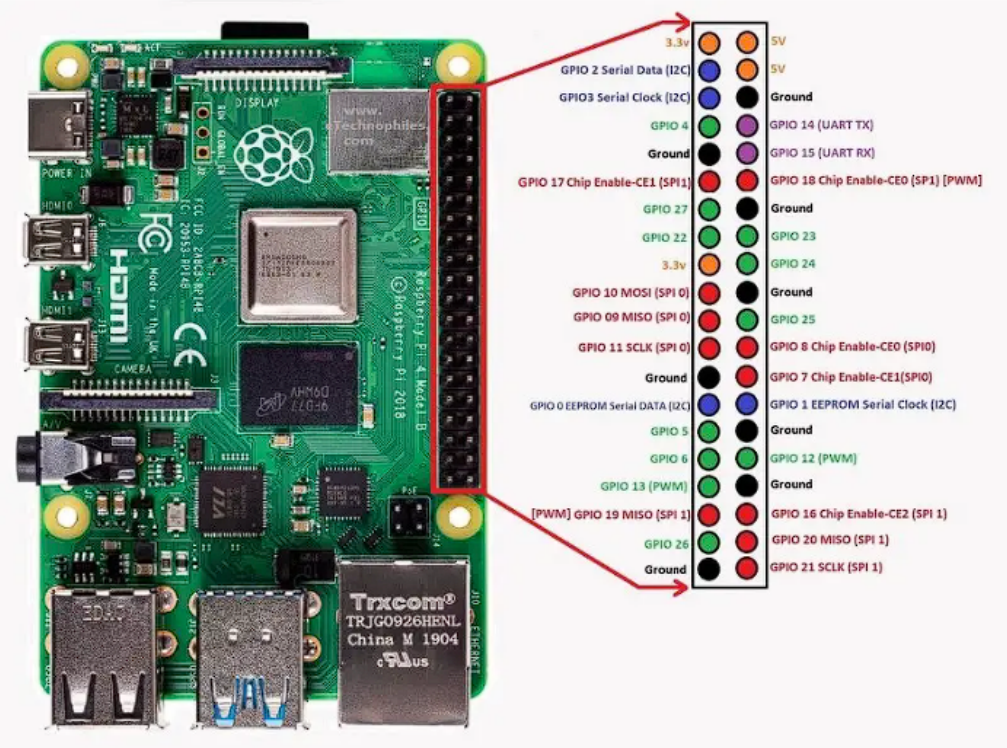

Figure 3.2.1: Pin Diagram Of Raspberry Pi

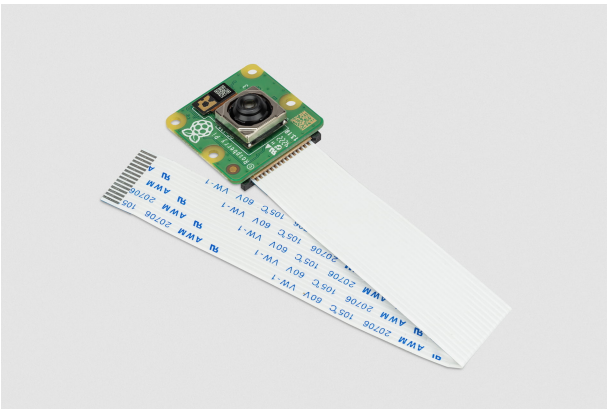

Figure 3.2.2: Pi Camera Module

3.3 SERVO MOTOR

A servo motor is a rotary actuator designed for precise control of angular position, speed, and acceleration. It consists of a DC motor, gear system, position sensor, and control circuitry, allowing it to rotate to a specific angle based on input signals. Servo motors are widely used in automation, robotics, and security applications due to their accuracy and reliability. In this Gate Security System, the servo motor is responsible for opening and closing the gate. When the system verifies an authorized entry—either through face recognition or correct PIN entry via the keypad—the Raspberry Pi sends a control signal to the servo motor, instructing it to rotate to the required angle and unlock the gate. Similarly, when the gate needs to close, the servo motor returns to its original position. The motor's controlled movement ensures smooth operation and precise gate positioning, enhancing the overall efficiency and security of the system.The figure of servo motor is given below as figure 3.3.1 .

Figure 3.3.1: Servo Motor

3.4 KEYPAD

A 4x4 keypad is a matrix-type input device consisting of 16 keys arranged in four rows and four columns. It is commonly used for user input in embedded systems, security applications, and automation projects. The keys are connected in a grid format, minimizing the number of GPIO pins required for interfacing with microcontrollers or microprocessors. In this Gate Security System, the 4x4 keypad is used as an alternative method for gate access. If a person's face is not recognized, they can manually enter a predefined PIN code using the keypad. The Raspberry Pi reads the input, verifies the code, and if correct, triggers the servo motor to open the gate. If an incorrect code is entered, access is denied, ensuring security and controlled entry.The figure

of keypad is given below as figure 3.4.1 .

Figure 3.4.1: Keypad 4x4

3.5 7805 VOLTAGE REGULATOR IC

The 7805 IC is a fixed linear voltage regulator that provides a stable 5V DC output from a higher voltage source, typically 7V to 35V. It is widely used in electronics to ensure a steady and regulated power supply to components requiring 5V operation.In this Gate Security System, the 7805 IC is used to convert higher voltage (e.g., 9V or 12V) to a stable 5V, which is essential for powering components like the Raspberry Pi, LCD display, servo motor, and other peripherals. The input voltage is applied to the VIN pin, and the regulated 5V output is taken from the VOUT pin, while the GND pin is connected to the circuit ground. To enhance its performance, capacitors are typically connected at the input and output to filter voltage fluctuations, ensuring smooth operation of the security system. The 7805 IC is a crucial component for maintaining reliable power delivery to sensitive electronic modules.The figure of voltage regulator is given below as figure 3.5.1 .

Figure 3.5.1: 7805 Voltage Regulator IC

3.6 BUZZER

A buzzer is an electromechanical or piezoelectric sound component used to generate an audible alarm or notification. It operates by converting electrical energy into sound waves through rapid mechanical vibrations or piezoelectric oscillations.In this Gate Security System, the buzzer is used to provide audible feedback for different events. It produces a short beep when the gate opens and a long beep when the gate closes. It is also activated when an unauthorized access attempt is detected or when a visitor presses the notification button. The buzzer is typically controlled by a microcontroller or Raspberry Pi, which sends a signal to trigger it at the appropriate times, enhancing security and user awareness.The figure of buzer is given below as figure 3.6.1 .

Figure 3.6.1: Buzzer

3.7 BLYNK APP

Blynk is an Internet of Things (IoT) platform that allows remote monitoring and control of embedded systems through a mobile application. It provides a user-friendly interface to visualize data, receive alerts, and interact with connected hardware over Wi-Fi, Bluetooth, or cellular networks.In this Gate Security System, the Blynk app is used for real-time monitoring and remote access control. When an unrecognized person presses the notification button, the owner receives an alert via the app. The live video feed from the Pi Camera Module is streamed through the app, enabling the owner to see the visitor. Based on this information, the owner can remotely control the gate, deciding whether to allow or deny entry. This integration enhances security by providing instant notifications, remote surveillance, and control capabilities.The logo of blynk is given below as figure 3.7.1 .

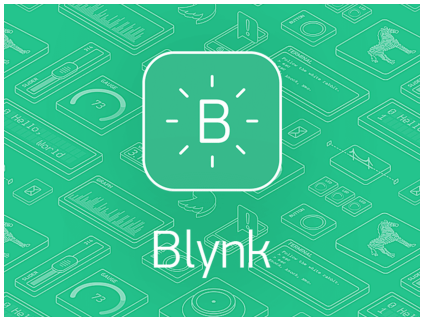

Figure 3.7.1: Logo Of Blynk Platform

3.8 PUSH BUTTON

A push button is a simple electrical switch that is used to complete or interrupt a circuit when pressed. It consists of a spring-loaded mechanism that returns to its default position when released, ensuring momentary or latching operation. In the Gate Security System, push buttons are used for multiple functions, such as triggering face recognition, activating a bell for visitor notification, and initiating manual gate access control. When a recognized person presses the face recognition button, the Raspberry Pi verifies the identity and opens the gate if authenticated. If an unknown visitor presses the notification button, the owner receives an alert via the Blynk app. The system relies on these push buttons to provide secure and user-friendly access control, making it an essential component of the project.The figure of push button is given below as figure 3.8.1 .

Figure 3.8.1: Push Button

3.9 LCD DISPLAY

The LCD (Liquid Crystal Display) used in this project is typically a 16x2 alphanumeric display, meaning it can show 16 characters per line across 2 rows. It operates using Hitachi HD44780 controller or compatible drivers, making it easy to interface with microcontrollers or microprocessors like the Raspberry Pi.In this Gate Security System, the LCD display is used to provide real-time feedback to users. It displays status messages such as ”Enter PIN,” ”Face Recognized,” ”Access Granted,” or ”Access Denied” to guide visitors through the authentication process. This enhances the user experience by providing clear and immediate responses to their actions.The pin diagram of LCD display is given below as figure 3.9.1 .

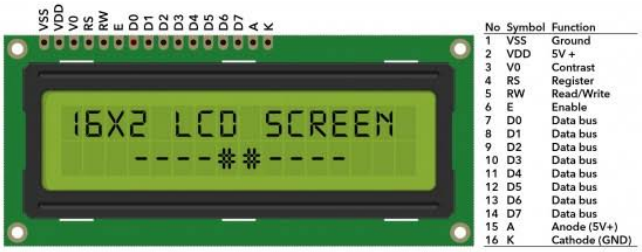

Figure 3.9.1: LCD Display

Chapter 4

WORKING

4.1 CIRCUIT DIAGRAM

The circuit shown is a Raspberry Pi-based Gate Security System integrating multiple components for authentication and access control. A 4x4 keypad is used for entering a passcode, which is processed by the Raspberry Pi to verify access. An LCD display provides real-time status updates, such as passcode entry and authentication results. A servo motor acts as the gate mechanism, rotating to open or close based on authentication success. A buzzer serves as an alert system, triggering an alarm for incorrect passcode attempts or unauthorized access. Additionally, push buttons are included for manual control, possibly for resetting the system or emergency operations. The Raspberry Pi handles the logic, processing inputs from the keypad and buttons, controlling the servo motor, and displaying relevant messages on the LCD. The entire system is designed to enhance security by allowing controlled access through both keypad authentication and remote monitoring via Raspberry Pi.The circuit daigram of the system is given below as figure 4.1.1 .

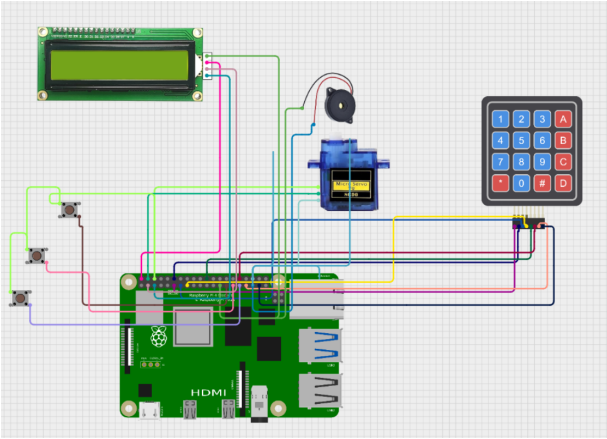

Figure 4.1.1: Circuit Diagram Of Gate Security System

4.2 WORKING OF THE CIRCUIT

- Person Approaches the Gate: When a person comes in front of the gate, they have multiple options to gain access.
- Face Recognition Access (Push Button 1):The visitor presses a push button to initiate face recognition.The Pi Camera Module captures the person's image and compares it with the pre-stored dataset in the Raspberry Pi 4B.If the face is recognized, the servo motor is triggered to open the gate.If the face is not recognized, access is denied, and the person must use an alternate method.
- Keypad Access (4x4 Keypad): If face recognition fails or is not preferred, the visitor can enter a PIN code on the 4x4 keypad.The Raspberry Pi verifies the entered PIN against stored values.If the PIN is correct, the servo motor opens the gate.If the PIN is incorrect, access is denied, and an alert can be triggered.
- Visitor Notification (Push Button 2 - Bell System): If a completely unknown visitor arrives, they can press another push button to ring a buzzer (bell).The Blynk app on the owner's smartphone receives a notification.The owner can view the live video feed from the Pi Camera Module via the Blynk app.Based on the live feed, the owner can manually decide whether to open the gate remotely.
- Gate Operation and Alerts: Whenever the gate opens, a short buzzer alarm is sounded.When the gate closes, a long buzzer alarm is activated.The LCD display can be used to show system messages, such as "Gate Opening ,Welcome!" or "Invalid PIN.Try again"
- Power Supply and Safety: The entire system is powered by a regulated 5V power supply using the 7805 voltage regulator IC.The system ensures security by preventing unauthorized access while providing convenient entry options for verified users.

Chapter 5

RESULTS

5.1 HARDWARE IMPLEMENTATION

The hardware implementation of the system is shown in figure 5.1.1 and in figure 5.1.2 .The gate is closed in normal scenario which is shown in figure 5.1.3 .And gate opens when push button is pressed and entered correct pincode through keypad which is shown in figure 5.1.4 below.The facial recognition output of person stored in dataset and stranger is shown in figure 5.1.5 .The live video streaming from Pi camera is available in the owners mobile phone in Blynk app and it is shown in figure 5.1.6 .

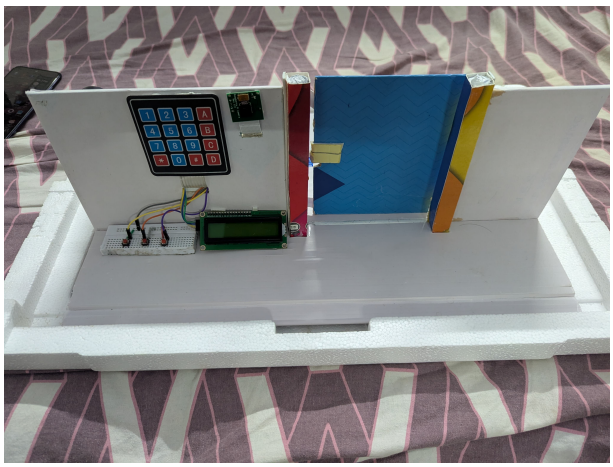

Figure 5.1.1: Hardware setup of system.

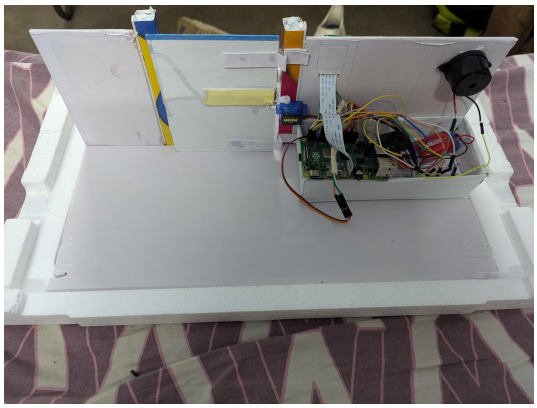

Figure 5.1.2: Hardware setup of system.

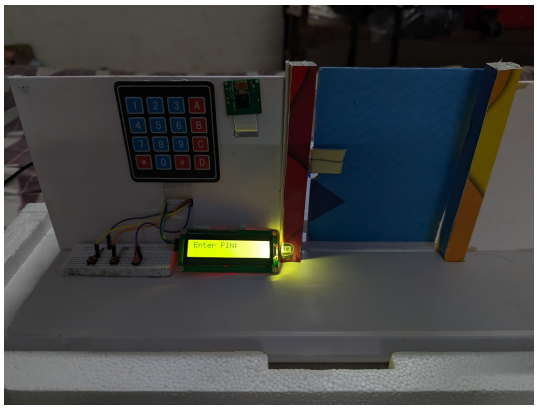

Figure 5.1.3: Gate closed in normal scenario.

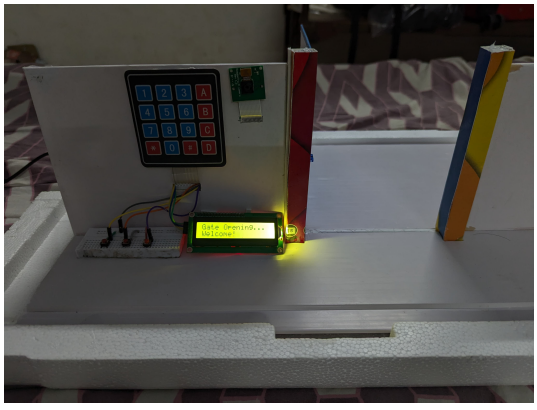

Figure 5.1.4: Gate opens when correct pincode is entered in keypad.

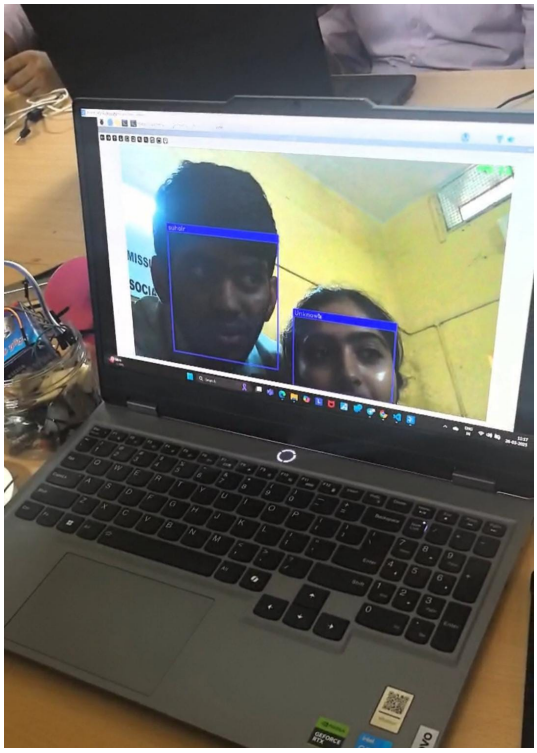

Figure 5.1.5: Output Of Face Recognition

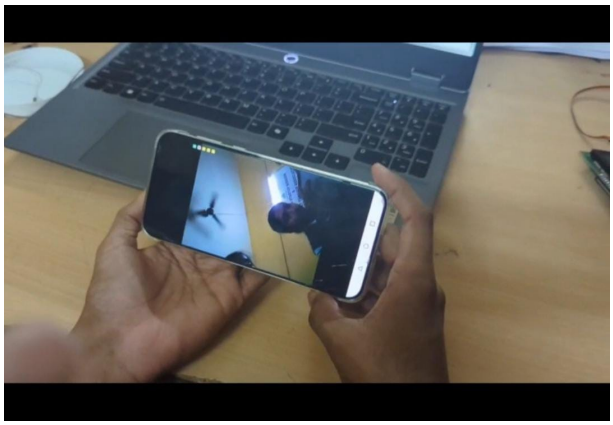

Figure 5.1.6: Output Of Live Video Streaming In Blynk App

Chapter 6

CONCLUSIONS

The Gate Security System is an efficient and reliable access control solution that enhances security through multiple authentication methods. By integrating face recognition, keypad access, and remote monitoring via the Blynk app, the system ensures only authorized individuals can enter. The use of a Raspberry Pi 4B as the central controller enables real-time decision-making, while components such as the Pi Camera Module, servo motor, LCD display, and buzzer work together to provide a seamless user experience. The system not only automates gate access but also notifies the owner of unknown visitors, offering a balance between security and convenience. With its versatile authentication modes, real-time alerts, and controlled gate operations, this project demonstrates a smart and scalable security solution suitable for residential and commercial applications.

Chapter 7

REFERENCES

1. R. Yu and M. Zhang, ”Smart Home Security Analysis System Based on The Internet of Things,” 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE 2021), Qingdao, China, 2021, pp.

2. Y. Xiao, Y. Jia, C. Liu, A. Alrawais, M. Rekik, and Z. Shan, ”HomeShield: A Credential-Less Authentication Framework for Smart Home Systems,” IEEE Internet of Things Journal, vol. 7, no. 9, pp. 7903-7914, Sept. 2020.

APPENDIX A

MAIN CODE

```
import RPi.GPIO as GPIO
import time
import subprocess
import threading
# Define GPIO pins for switches
SWITCH_1 = 0 # Runs facial_detection.py
SWITCH_2 = 5 # Runs pin_lock.py
SWITCH_3 = 6 # Runs calling_bell.py
# GPIO setup
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(SWITCH_1, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(SWITCH_2, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(SWITCH_3, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# Function to run a script based on the switch pressed
def run_script(script_name):
 print(f"Running␣{script_name}...")
 subprocess.run(["python3", script_name])
# Function to start Blynk Stream Control
def start_blynk_stream():
 print("Starting␣Blynk␣Stream␣Control...")
 return subprocess.Popen(["python3", "blynk_stream_cntrl.py"])
blynk_process = start_blynk_stream()
try:
 print("Monitoring␣switch␣states...")
 while True:
 if GPIO.input(SWITCH_1) == GPIO.LOW:
 print("Stopping␣Blynk␣Stream␣for␣Facial␣Recognition...")
 blynk_process.terminate() # Stop Blynk Stream
 blynk_process.wait() # Ensure process has fully stopped

 run_script("facial_detection.py") # Run facial recognition

 print("Restarting␣Blynk␣Stream...")
 blynk_process = start_blynk_stream() # Restart Blynk Stream

 while GPIO.input(SWITCH_1) == GPIO.LOW:
 time.sleep(0.2)

 if GPIO.input(SWITCH_2) == GPIO.LOW:
 run_script("pin_lock.py")
 while GPIO.input(SWITCH_2) == GPIO.LOW:
```

```
 time.sleep(0.2)
 if GPIO.input(SWITCH_3) == GPIO.LOW:
 run_script("calling_bell.py")
 while GPIO.input(SWITCH_3) == GPIO.LOW:
 time.sleep(0.2)
except KeyboardInterrupt:
 print("\nExiting...")
finally:
 GPIO.cleanup()
 if blynk_process:
 blynk_process.terminate()
 blynk_process.wait()
```

BLYNK STREAMING CODE

```
import io
import logging
import socketserver
import threading
import time
import subprocess
from http import server
from threading import Condition
import cv2
from picamera2 import Picamera2
import BlynkLib
from BlynkTimer import BlynkTimer
BLYNK_AUTH_TOKEN = 'TnHAv7NLnYw8rDG_AkGSMB8qNA9SZVzh'
PAGE = """
<html>
<head>
<title>Raspberry Pi - Surveillance Camera</title>
</head>
<body>
<center><h1>Raspberry Pi - Surveillance Camera</h1></center>
<center><img src="stream.mjpg" width="640" height="480"></center>
</body>
</html>
"""
class StreamingOutput(object):
 def __init__(self):
 self.frame = None
 self.condition = Condition()
 def update_frame(self, frame):
 with self.condition:
 self.frame = frame
 self.condition.notify_all()
class StreamingHandler(server.BaseHTTPRequestHandler):
```

```
 def do_GET(self):
 if self.path == '/':
 self.send_response(301)
 self.send_header('Location', '/index.html')
 self.end_headers()
 elif self.path == '/index.html':
 content = PAGE.encode('utf-8')
 self.send_response(200)
 self.send_header('Content-Type', 'text/html')
 self.send_header('Content-Length', len(content))
 self.end_headers()
 self.wfile.write(content)
 elif self.path == '/stream.mjpg':
 self.send_response(200)
 self.send_header('Age', 0)
 self.send_header('Cache-Control', 'no-cache, private')
 self.send_header('Pragma', 'no-cache')
 self.send_header('Content-Type', 'multipart/x-mixed-replace;
boundary=FRAME')
 self.end_headers()
 try:
 while True:
 with output.condition:
 output.condition.wait()
 frame = output.frame
 self.wfile.write(b'--FRAME\r\n')
 self.send_header('Content-Type', 'image/jpeg')
 self.send_header('Content-Length', len(frame))
 self.end_headers()
 self.wfile.write(frame)
 self.wfile.write(b'\r\n')
 except Exception as e:
 logging.warning(
 'Removed streaming client %s: %s',
 self.client_address, str(e))
 else:
 self.send_error(404)
 self.end_headers()
class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
 allow_reuse_address = True
 daemon_threads = True
# Initialize camera and streaming
picam2 = Picamera2()
picam2.configure(picam2.create_video_configuration(main={"format": "
    XRGB8888", "size":
(640, 480)}))
picam2.start()
```

```
output = StreamingOutput()
def capture_loop():
 while True:
 frame = picam2.capture_array()
 ret, jpeg = cv2.imencode('.jpg', frame)
 output.update_frame(jpeg.tobytes())
 time.sleep(0.05) # ~20fps
def run_video_stream():
 capture_thread = threading.Thread(target=capture_loop, daemon=True)
 capture_thread.start()
 address = ('', 8000)
 server = StreamingServer(address, StreamingHandler)
 print("Streaming on http://<raspberrypi_ip>:8000")
 server.serve_forever()
# Blynk setup
blynk = BlynkLib.Blynk
def run_servo_buzzer():
 try:
 subprocess.run(["python3", "servo_buzzer.py"], check=True)
 print("Executed servo_buzzer.py successfully.")
 except subprocess.CalledProcessError as e:
 print(f"Error executing servo_buzzer.py: {e}")
@blynk.on("V0")
def v0_write_handler(value):
 if int(value[0]) != 0:
 print("Switch ON - Running servo_buzzer.py")
 run_servo_buzzer()
 else:
 print("Switch OFF - No action")
@blynk.on("connected")
def blynk_connected():
 print("Raspberry Pi Connected to New Blynk")
def run_blynk():
 while True:
 blynk.run()
# Run both functionalities in separate threads
threading.Thread(target=run_video_stream, daemon=True).start()
threading.Thread(target=run_blynk, daemon=True).start()
# Keep the main thread running
while True:
 time.sleep(1)
```

FACIAL RECOGNITION CODE

```
import face_recognition
import cv2
import numpy as np
import subprocess
```

```
from picamera2 import Picamera2
import time
import pickle
import os
import signal
# Load pre-trained face encodings
print("[INFO] Loading encodings...")
with open("encodings.pickle", "rb") as f:
 data = pickle.loads(f.read())
known_face_encodings = data["encodings"]
known_face_names = data["names"]
# Function to check and initialize the camera safely
def initialize_camera():
 try:
 picam2 = Picamera2()
 picam2.configure(picam2.create_preview_configuration(main={"format": '
    XRGB8888',
"size": (1920, 1080)}))
 picam2.start()
 return picam2
 except Exception as e:
 print(f"[ERROR] Failed to initialize camera: {e}")
 exit(1)
# Initialize the camera
picam2 = initialize_camera()
cv_scaler = 4 # Scale factor for performance
face_locations = []
face_encodings = []
face_names = []
# Function to process frame
def process_frame(frame):
 global face_locations, face_encodings, face_names
 resized_frame = cv2.resize(frame, (0, 0), fx=(1/cv_scaler), fy=(1/
    cv_scaler))
 rgb_resized_frame = cv2.cvtColor(resized_frame, cv2.COLOR_BGR2RGB)
 face_locations = face_recognition.face_locations(rgb_resized_frame)
 face_encodings = face_recognition.face_encodings(rgb_resized_frame,
    face_locations,
model='large')

 face_names = []
 for face_encoding in face_encodings:
 matches = face_recognition.compare_faces(known_face_encodings,
    face_encoding)
 name = "Unknown"
 face_distances = face_recognition.face_distance(known_face_encodings,
 face_encoding)
```

```
 best_match_index = np.argmin(face_distances)
 if matches[best_match_index]:
 name = known_face_names[best_match_index]
 trigger_gate() # Open gate if face is recognized
 terminate_recognition() # Terminate only recognition
 face_names.append(name)

 return frame
# Function to trigger servo and buzzer
def trigger_gate():
 print("[INFO] Recognized face detected, opening gate...")
 subprocess.run(["python3", "servo_buzzer.py"]) # Run the gate control
     script
# Function to terminate face recognition process
def terminate_recognition():
 print("[INFO] Face recognition process terminating...")
 os.kill(os.getpid(), signal.SIGTERM)
# Main loop
try:
 while True:
 frame = picam2.capture_array()
 processed_frame = process_frame(frame)

 # Display results
 for (top, right, bottom, left), name in zip(face_locations, face_names):
 top *= cv_scaler
 right *= cv_scaler
 bottom *= cv_scaler
 left *= cv_scaler
 cv2.rectangle(frame, (left, top), (right, bottom), (244, 42, 3), 3)
 cv2.rectangle(frame, (left - 3, top - 35), (right + 3, top), (244, 42, 3),
cv2.FILLED)
 font = cv2.FONT_HERSHEY_DUPLEX
 cv2.putText(frame, name, (left + 6, top - 6), font, 1.0, (255, 255, 255),
     1)

 cv2.imshow('Video', frame)
 if cv2.waitKey(1) == ord("q"):
 break
except KeyboardInterrupt:
 print("[INFO] Exiting...")
finally:
 print("[INFO] Releasing camera and closing windows...")
 cv2.destroyAllWindows()
 picam2.stop()
```

PIN LOCK CODE

```
import subprocess
from gpiozero import DigitalOutputDevice, Button
import smbus
import time
import RPi.GPIO as GPIO
import os
import signal
# ? I2C LCD Display Setup
I2C_ADDR = 0x27 # Update if needed
LCD_WIDTH = 16
LCD_CHR = 1
LCD_CMD = 0
LCD_LINE_1 = 0x80
LCD_LINE_2 = 0xC0
ENABLE = 0b00000100
E_DELAY = 0.0005
# ? GPIO Setup
GPIO.setmode(GPIO.BCM)
# ? Configure Keypad Pins
rows_pins = [17, 18, 27, 22]
cols_pins = [25, 12, 13, 26]
keys = [["1", "2", "3", "A"],
 ["4", "5", "6", "B"],
 ["7", "8", "9", "C"],
 ["*", "0", "#", "D"]]
# ? Initialize LCD Display
class LCD:
 def __init__(self):
 self.bus = smbus.SMBus(1)
 self.lcd_init()
 def lcd_init(self):
 self.lcd_byte(0x33, LCD_CMD)
 self.lcd_byte(0x32, LCD_CMD)
 self.lcd_byte(0x28, LCD_CMD)
 self.lcd_byte(0x0C, LCD_CMD)
 self.lcd_byte(0x06, LCD_CMD)
 self.lcd_byte(0x01, LCD_CMD)
 time.sleep(E_DELAY)
 def lcd_byte(self, bits, mode):
 BACKLIGHT = 0x08
 high_bits = mode | (bits & 0xF0) | ENABLE
 low_bits = mode | ((bits << 4) & 0xF0) | ENABLE
 self.bus.write_byte(I2C_ADDR, high_bits)
 self.lcd_toggle_enable(high_bits)
 self.bus.write_byte(I2C_ADDR, low_bits)
 self.lcd_toggle_enable(low_bits)
```

```
 def lcd_toggle_enable(self, bits):
 time.sleep(E_DELAY)
 self.bus.write_byte(I2C_ADDR, (bits & ~ENABLE) | 0x08)
 time.sleep(E_DELAY)
 def lcd_display_string(self, message, line):
 message = message.ljust(LCD_WIDTH, "␣")
 if line == 1:
 self.lcd_byte(LCD_LINE_1, LCD_CMD)
 elif line == 2:
 self.lcd_byte(LCD_LINE_2, LCD_CMD)
 for char in message:
 self.lcd_byte(ord(char), LCD_CHR)
 def lcd_clear(self):
 self.lcd_byte(0x01, LCD_CMD)
# ? Initialize LCD
lcd = LCD()
# ? Initialize Keypad
rows = [DigitalOutputDevice(pin) for pin in rows_pins]
cols = [Button(pin, pull_up=False) for pin in cols_pins]
# ? Set the Correct PIN
CORRECT_PIN = "5234"
# ? Function to Read Keypad Input
def read_keypad():
 entered_pin = ""
 lcd.lcd_display_string("Enter␣PIN:", 1)
 lcd.lcd_display_string("␣", 2) # Clear 2nd line
 while len(entered_pin) < 4:
 for row_index, row in enumerate(rows):
 row.on()
 for col_index, col in enumerate(cols):
 if col.is_pressed:
 key = keys[row_index][col_index]
 entered_pin += key
 lcd.lcd_display_string("*" * len(entered_pin), 2)
 time.sleep(0.3) # Debounce
 row.off()
 return entered_pin
# ? Function to Trigger Gate Opening Script
def trigger_gate():
 lcd.lcd_display_string("Gate␣Opening...", 1)
 lcd.lcd_display_string("Welcome!", 2)
 subprocess.run(["python3", "servo_buzzer.py"]) # Run the script
 time.sleep(2)
 lcd.lcd_clear()
 terminate_script()
# ? Function to Terminate key_disp.py
def terminate_script():
```

```
 lcd.lcd_display_string("System Off", 1)
 time.sleep(1)
 GPIO.cleanup()
 os.kill(os.getpid(), signal.SIGTERM)
# ? Main Security System Loop
try:
 while True:
 entered_pin = read_keypad()
 if entered_pin == CORRECT_PIN:
 trigger_gate()
 else:
 lcd.lcd_display_string("Incorrect PIN", 1)
 lcd.lcd_display_string("Try Again", 2)
 time.sleep(2)
 lcd.lcd_clear()
except KeyboardInterrupt:
 terminate_script()
```

SERVO AND BUZZER CODE

```
import pigpio
import time
servo_pin = 19 # GPIO 19 for Servo
buzzer_pin = 16 # GPIO 16 for Buzzer
pi = pigpio.pi()
def set_angle(angle):
 pulse_width = 500 + (angle * 2000 // 180) # Convert angle to pulse width
 pi.set_servo_pulsewidth(servo_pin, pulse_width)
 time.sleep(0.5)
def buzz(duration):
 """Turn the buzzer ON for a specified duration."""
 pi.write(buzzer_pin, 1) # Buzzer ON
 time.sleep(duration)
 pi.write(buzzer_pin, 0) # Buzzer OFF
# Set GPIO 16 as output
pi.set_mode(buzzer_pin, pigpio.OUTPUT)
try:
 # Open gate with short buzzer sound
 buzz(0.2) # Short beep
 set_angle(90)
 print("Gate opened")
 time.sleep(10) # Keep gate open for 10 seconds
 # Close gate with long buzzer sound
 buzz(1.0) # Long beep
 set_angle(0)
 print("Gate closed")
except KeyboardInterrupt:
 pi.set_servo_pulsewidth(servo_pin, 0) # Stop sending signal
```

```
 pi.write(buzzer_pin, 0) # Turn off buzzer
 pi.stop()
```

CALLING BELL CODE

```
import RPi.GPIO as GPIO
import time
# Define GPIO pin for the buzzer
BUZZER = 16
# GPIO setup
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUZZER, GPIO.OUT)
try:
 print("Buzzer is sounding...")
 GPIO.output(BUZZER, GPIO.HIGH) # Turn on buzzer
 time.sleep(1) # Buzzer sound duration (adjust as needed)
 GPIO.output(BUZZER, GPIO.LOW) # Turn off buzzer
 print("Buzzer stopped. Exiting program.")
except KeyboardInterrupt:
 print("\nExiting...")
finally:
 GPIO.output(BUZZER, GPIO.LOW) # Ensure buzzer is off before exit
 GPIO.cleanup()
```