Smart Door Lock System

STUDENT'S NAMES

Kolli Sreeram Sai
Suchit Reddi
Akshay Veeragandham
Koti Eswar

Group No.14

—

EED379: Internet of Things

—

Dr. Rohit Singh

TABLE OF CONTENTS

ABSTRACT

This project presents the development of an IoT-based Smart Door Security System leveraging the MQTT protocol for secure and efficient communication. Central to the system is a Raspberry Pi, which serves as the control unit for managing access and intrusion detection. Upon an incorrect password attempt, the system activates a camera module to capture a photo of the individual, which is then promptly sent to a subscribing device for verification. This mechanism ensures real-time alerts and enhances the security measures of the premises. The project encapsulates the integration of hardware and software components, programming the Raspberry Pi to handle MQTT messaging, and deploying the system in a real-world environment. The outcome is a scalable, robust door security system that contributes to the safety and smart management of access control in the context of IoT.

COMPONENTS

1) Raspberry Pi

At the heart of the Smart Door Security System lies the Raspberry Pi, functioning as the central control unit. This versatile single-board computer is responsible for managing access control, processing keypad input, and activating the camera module. The GPIO pins of the Raspberry Pi are utilized to interface with the system's peripherals, creating a seamless integration between hardware and software.

2) Keypad

The keypad serves as the primary user interface for entering access codes into the system. Interfacing with the Raspberry Pi, the keypad provides a secure and user-friendly means of input. Its integration is crucial for enabling convenient access control, allowing users to input their access codes with ease.

3) Camera Module

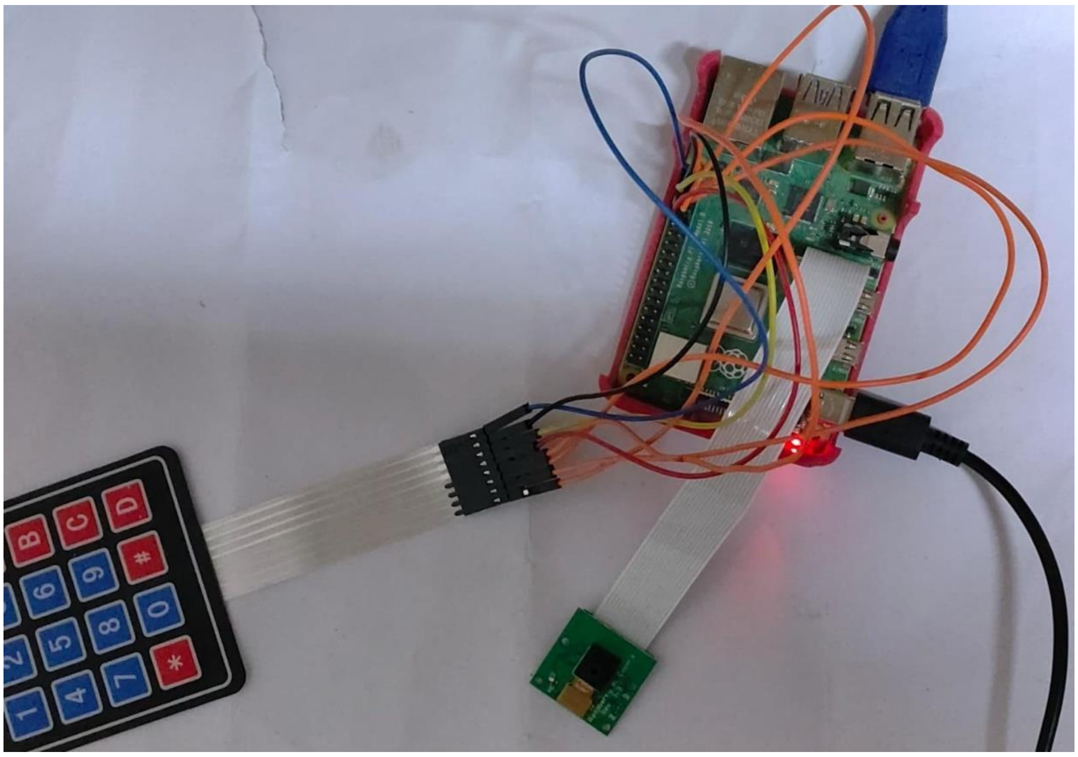
Enhancing the security measures of the Smart Door Security System, the camera module is activated upon incorrect password attempts. Capturing images in real-time, the camera module visually verifies individuals seeking access. This visual data not only adds an extra layer of authentication but also facilitates real-time monitoring and alerts in the event of unauthorized access.

4) Mqtt protocol

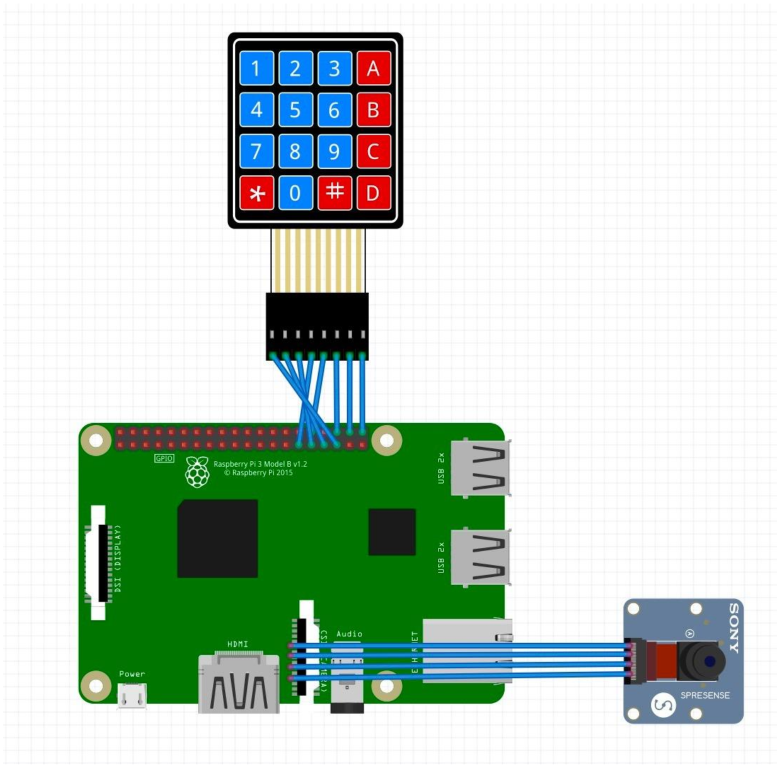
The MQTT (Message Queuing Telemetry Transport) protocol acts as the communication backbone of the system. This lightweight and efficient protocol enable secure and real-time data exchange between devices. In the context of the Smart Door Security System, Mosquitto Mqtt is utilized for transmitting images captured by the camera module. This ensures that subscribing devices receive prompt alerts and visual verification in response to security events.

mosquitto

WORKING CIRCUIT



SCHEMATIC



WORKING

The Smart Door Security System operates seamlessly through a combination of hardware components and software functionalities, ensuring efficient access control and real-time monitoring.

Upon initialization, the Raspberry Pi, functioning as the central control unit, establishes communication with the connected peripherals: the keypad and the camera module. The keypad serves as the user interface, allowing individuals to input access codes securely. The GPIO pins on the Raspberry Pi facilitate this interaction, providing a responsive and reliable means of capturing user input.

In the event of an incorrect password attempt, the system is designed to trigger the camera module. The camera captures a photo of the individual seeking access, creating a visual record for verification purposes. This image is then transmitted using the MQTT protocol, ensuring secure and efficient communication between the Raspberry Pi and subscribing devices.

To handle the messaging and transmission, two separate Python scripts, `send.py` and `receive.py`, are employed. The `send.py` script is responsible for publishing the captured image to the MQTT topic, while the `receive.py` script, running on a subscribing device, is configured to receive and process incoming images.

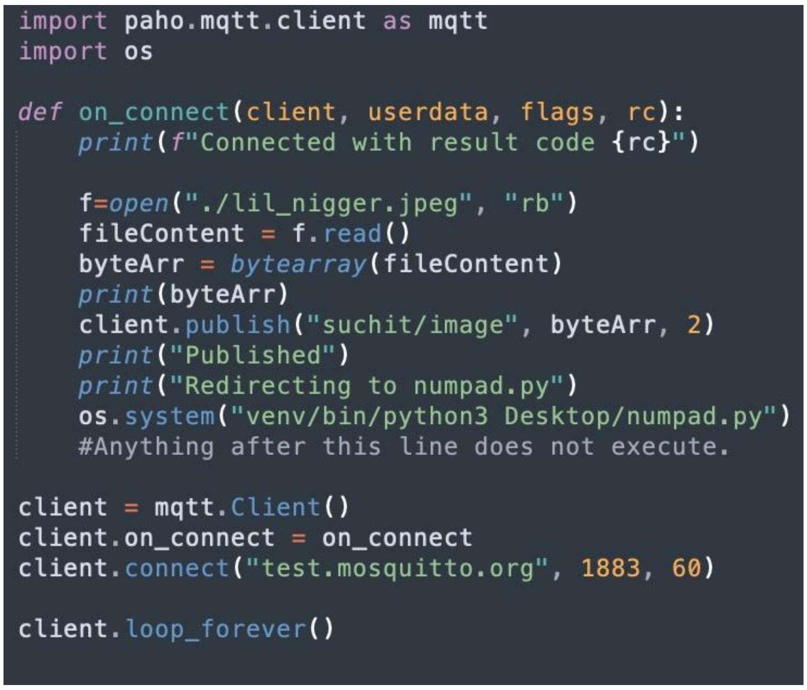
Additionally, the system incorporates a script named `numpad.py` that manages user input through the keypad. This script ensures that correct access codes are accepted, and incorrect attempts trigger the camera module for visual verification. The captured image is then resized to meet MQTT constraints before being transmitted for further validation.

The collaborative operation of these components results in a robust Smart Door Security System that enhances access control, captures visual evidence upon security events, and provides real-time alerts to subscribing devices. The integration of hardware and software ensures a seamless and responsive user experience, contributing to the overall security and smart management of access control in the context of the Internet of Things (IoT).

CODE

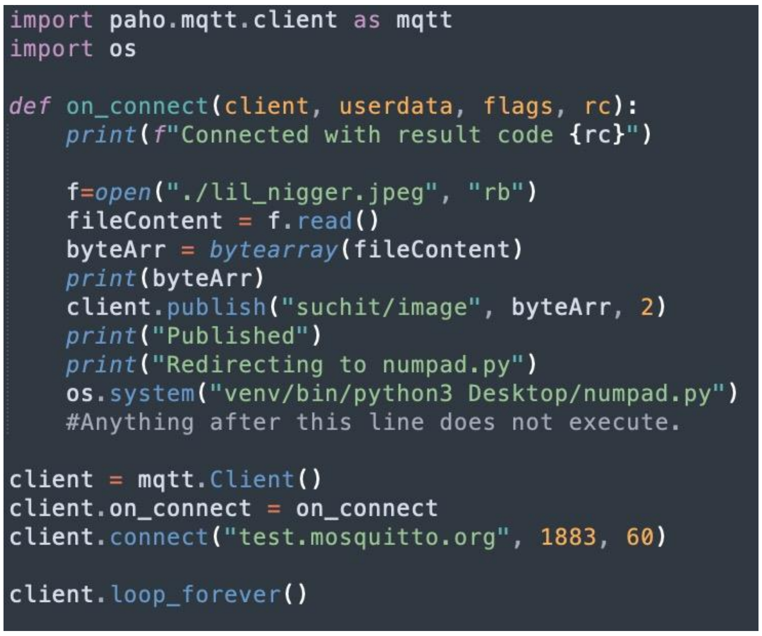
The implementation of the Smart Door Security System involves a series of Python scripts designed to orchestrate the communication between the Raspberry Pi, keypad, camera module, and subscribing devices through the MQTT protocol. Below, the key aspects of the code are outlined:

Send.py



```python
import paho.mqtt.client as mqtt
import os

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")

    f=open("./lil_nigger.jpeg", "rb")
    fileContent = f.read()
    byteArr = bytearray(fileContent)
    print(byteArr)
    client.publish("suchit/image", byteArr, 2)
    print("Published")
    print("Redirecting to numpad.py")
    os.system("venv/bin/python3 Desktop/numpad.py")
    #Anything after this line does not execute.

client = mqtt.Client()
client.on_connect = on_connect
client.connect("test.mosquitto.org", 1883, 60)

client.loop_forever()
```

Receive.py



```
import paho.mqtt.client as mqtt
import os

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")

    f=open("./lil_nigger.jpeg", "rb")
    fileContent = f.read()
    byteArr = bytearray(fileContent)
    print(byteArr)
    client.publish("suchit/image", byteArr, 2)
    print("Published")
    print("Redirecting to numpad.py")
    os.system("venv/bin/python3 Desktop/numpad.py")
    #Anything after this line does not execute.

client = mqtt.Client()
client.on_connect = on_connect
client.connect("test.mosquitto.org", 1883, 60)

client.loop_forever()
```

BENEFITS

The Smart Door Security System presents several advantages that contribute to enhanced security, access control, and real-time monitoring in the IoT context:

1. **Authentication Mechanism:**
   Combining keypad authentication with visual verification through the camera module ensures a robust authentication mechanism, significantly enhancing the system's security.
2. **Real-time Alerts:**
   The system provides real-time alerts upon incorrect password attempts, enabling immediate response and intervention in the event of potential security breaches.
3. **User-Friendly Interface:**
   The keypad serves as a user-friendly interface for entering access codes, making the system convenient for authorized users while maintaining a high level of security.
4. **Scalability:**
   The modular design of the system and the use of MQTT for communication facilitate scalability, allowing for easy integration with additional devices or expansion of security features.
5. **Integration with IoT Ecosystem:**
   Leveraging the MQTT protocol, the system seamlessly integrates with other IoT devices, contributing to a comprehensive IoT ecosystem for smart home or office environments.

CHALLENGES

1. **Image Size Constraint:**
   The limitation on image size for MQTT transmission poses a challenge. Resizing captured images is necessary to meet this constraint, potentially impacting the quality of visual verification.
2. **Limited Password Change Accessibility:**
   The challenge lies in the restricted ability to change the keypad password, which currently necessitates code modification. This limitation may hinder user-friendly adaptation and requires a more accessible and secure password change mechanism, ideally through a dedicated interface or remote configuration, to accommodate evolving security needs.

FUTURE ITERATIONS

1. **Image Compression Enhancement:**
   - Future iterations could explore more advanced image compression techniques to maintain visual quality while adhering to MQTT size constraints. This optimization would improve the efficiency of image transmission.
2. **Code Optimization for Efficiency:**
   - Continuous refinement of the codebase can lead to increased system efficiency and reduced resource utilization. This ongoing optimization ensures the system operates seamlessly, especially in resource-constrained IoT environments.
3. **Biometric Integration Exploration:**
   - Consideration for integrating biometric authentication methods, such as facial recognition or fingerprints, can add an extra layer of security and user convenience, potentially enhancing the overall authentication process.
4. **User-Friendly Password Management:**
   - Future iterations may focus on creating a more user-friendly mechanism for changing keypad passwords, possibly through a dedicated interface or remote configuration. This improvement aims to enhance accessibility and streamline security updates.
5. **Integration with Smart Home Systems:**

- Exploring integration possibilities with existing smart home ecosystems, such as voice assistants or mobile applications, could broaden the system's compatibility and user interface options.

CONCLUSION

In conclusion, the Smart Door Security System presents a commendable solution for secure access control within the realm of the Internet of Things. While facing challenges such as image size constraints and limited password change accessibility, the system excels in providing real-time alerts and maintaining a user-friendly interface. Looking ahead, future iterations could focus on image compression refinement, code optimization, biometric integration, improved password management, and integration with broader smart home ecosystems. As it currently stands, the system stands as a testament to enhanced security and intelligent access control in the dynamic landscape of IoT.

REFERENCES

1. **Paho MQTT Library Documentation:**
   Eclipse Paho - MQTT for Python.
   (https://www.eclipse.org/paho/clients/python/docs/)
2. **Libcamera Documentation:**
   Libcamera - A camera stack for embedded systems.
   (https://libcamera.org/docs/)
3. **Imagemagick Documentation:**
   (https://imagemagick.org/script/index.php)
4. **Smart Door Security System - David Maciejak's Blog:**
   Blog Post: "Files Over MQTT - Solution in Python."
   (https://davidmac.pro/posts/2021-07-21-files-over-mqtt/)