Trackify System Documentation: Integration of RFID, Facial Recognition, and IoT for Attendance Management

1. reconnaissancefinale.py

Description

This script integrates facial recognition with RFID authentication to create a robust attendance management system. It ensures that attendance records are secure and reliable by combining the use of RFID card scanning and facial recognition using OpenCV.

Purpose

To provide an accurate and secure attendance tracking system for academic institutions that minimizes proxy attendance and enhances accountability.

Key Features

Facial Recognition: Detects and verifies individuals against stored facial profiles in a database.

RFID Authentication: Scans RFID cards to validate user identity.

Dual Validation: Combines RFID and facial recognition for enhanced security.

Attendance Management: Automatically marks attendance in a database after successful verification.

Setup

Hardware Requirements:

Raspberry Pi 4.

RFID reader (MFRC522).

Camera module compatible with Raspberry Pi (e.g., Hikvision).

Software Requirements:

Python 3.x.

Libraries: cv2 (OpenCV), RPi.GPIO, mfrc522.

Step-by-Step Workflow

Initialization:

Import required libraries and set up GPIO pins for the RFID reader and camera.

Load pre-stored user profiles (facial images and RFID UIDs).

RFID Authentication:

Wait for an RFID card to be scanned.

Extract the card's UID and check if it matches a stored UID in the authorized database.

Facial Recognition:

Capture an image using the camera.

Compare the captured image against stored facial profiles using OpenCV.

Attendance Logging:

If both RFID and facial recognition validations are successful, log the user as "present" in the attendance database.

Feedback:

Provide real-time feedback on the success or failure of authentication.

Dependencies

OpenCV: For facial recognition and image processing.

RPi.GPIO: For interfacing with the GPIO pins of the Raspberry Pi.

mfrc522: For RFID card reading and writing.

Error Handling

If RFID UID does not match the authorized list, display an error message and deny access.

If facial recognition fails, prompt the user to try again.

2. rfid-read.py

Description

This script is a simple standalone RFID reader that scans RFID cards, validates their UIDs, and outputs corresponding user roles (e.g., teacher, student).

Purpose

To validate user identities based on RFID card data and provide quick feedback on access rights.

Key Features

Reads RFID card data.

Matches the UID against a preloaded database of authorized users.

Outputs user roles or denial messages based on validation.

Setup

Hardware Requirements:

Raspberry Pi.

RFID reader module (MFRC522).

Software Requirements:

Python 3.x.

Libraries: RPi.GPIO, mfrc522.

Step-by-Step Workflow

Initialization:

Import required libraries and set up GPIO pins for the RFID reader.

Define a list of authorized UIDs and their associated roles.

Card Scanning:

Wait for a card to be placed near the RFID reader.

Extract the UID from the scanned card.

Validation:

Check if the scanned UID matches an entry in the authorized database.

Output the corresponding user role (e.g., "Teacher", "Student") or deny access.

Dependencies

RPi.GPIO: For hardware-level GPIO interfacing.

mfrc522: For reading RFID card data.

Error Handling

If the UID is not found in the database, display a "Denied Access" message.

3. rfidtest.py

Description

This script integrates RFID authentication with ultrasonic sensors and relay control for managing physical access (e.g., door control).

Purpose

To provide a secure physical access control system for restricted areas using a combination of RFID and ultrasonic sensors.

Key Features

Ultrasonic Sensor: Detects the presence of individuals near the door.

RFID Authentication: Verifies users' identities using RFID cards.

Relay Control: Activates a relay to open the door for authorized users.

Event Logging: Logs access events and sensor readings.

Setup

Hardware Requirements:

Raspberry Pi.

RFID reader (MFRC522).

Ultrasonic sensor (e.g., HC-SR04).

Relay module.

Software Requirements:

Python 3.x.

Libraries: RPi.GPIO, mfrc522.

Step-by-Step Workflow

Initialization:

Set up GPIO pins for the ultrasonic sensor, RFID reader, and relay.

Load authorized RFID UIDs.

Proximity Detection:

Use the ultrasonic sensor to detect a person near the door.

Measure the distance and trigger RFID authentication if within range.

RFID Authentication:

Wait for a card to be scanned.

Validate the UID against authorized user data.

Relay Activation:

If the UID is valid, activate the relay to open the door.

Log the event in the system.

Dependencies

RPi.GPIO: For GPIO control.

mfrc522: For RFID operations.

Error Handling

If no one is detected by the ultrasonic sensor, the system waits.

If the UID is invalid, deny access and log the event.

4. rfid-write.py

Description

This script writes user data onto RFID cards. It allows for encoding text data (e.g., user names, roles) onto RFID cards.

Purpose

To encode user-specific information onto RFID cards for use in authentication systems.

Key Features

Prompts the user to input data for encoding.

Writes the input data onto an RFID card.

Setup

Hardware Requirements:

Raspberry Pi.

RFID reader/writer (MFRC522).

Software Requirements:

Python 3.x.

Libraries: RPi.GPIO, mfrc522.

Step-by-Step Workflow

Initialization:

Import required libraries and set up GPIO pins for the RFID writer.

Input Collection:

Prompt the user to input the data to be written (e.g., name, role).

Card Writing:

Wait for an RFID card to be placed near the writer.

Encode the input data onto the card.

Confirmation:

Confirm that the data has been successfully written.

Dependencies

RPi.GPIO: For GPIO interfacing.

mfrc522: For writing data to RFID cards.

Error Handling

If the card is removed before writing is complete, prompt the user to try again.

Conclusion

This documentation provides detailed information about the purpose, features, setup, workflow, and error handling for each script. It can be used for understanding, troubleshooting, or extending the functionality of the system.