The Vault

By:
Andrew Devadhason, ad7068
Devika Kodi, dak9250
Shagun Majotra, sm11304
Vineela Reddy, vp2504

New York University

Tandon School of Engineering

Department of Mechanical and Aerospace Engineering

Advanced Mechatronics, ROB-GY 6103

Prof. Vikram Kapila

INDEX

ACKNOWLEDGEMENT

We would like to express our heartfelt appreciation to everyone who contributed to the realization of this project. We extend our gratitude to Professor Vikarm Kapila for their invaluable guidance and encouragement throughout the journey. We also thank our team members for their unwavering commitment and collaborative spirit, which played a significant role in overcoming challenges and achieving our goals. Additionally, we acknowledge the support received from our institution and the resources provided that facilitated the implementation of our ideas.

ABSTRACT

This project presents the development of a secure safe system utilizing a dual-layered authentication approach for enhanced security. The system integrates RFID technology and facial recognition to verify user credentials before granting access to the safe's contents. Upon user interaction, RFID validation is initiated, followed by facial recognition authentication. Successful validation results in the unlocking of the safe, while failure restricts access. The hardware setup includes an Arduino for RFID interfacing and servo control and a Raspberry Pi 4 for facial recognition using a camera module. The custom-designed and 3D-printed safe provides secure housing for the servo-based lock mechanism. Through the synergistic combination of these methods, this project offers a comprehensive solution to safeguard physical assets effectively against unauthorized access and impersonation scenarios.

[1] INTRODUCTION

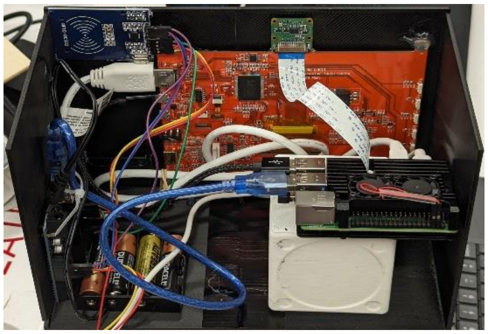
In our project, we have developed a safe with top-tier security features. We've combined RFID technology and facial recognition for a two-step verification process, ensuring only authorized users can access the safe's contents. As a first step, authorized users tap their RFID cards to confirm their identity. A specific buzzer indicates successful validation, while a different sound signifies failure. Once RFID validation is successful, users undergo facial recognition. This adds an extra layer of security, with the Raspberry Pi and a camera recognizing the user's face based on stored data. If facial recognition is successful, the safe unlocks using servo motors. If not, access is denied.

The hardware counterpart includes an Arduino for interfacing with the RFID reader and servos, and a Raspberry Pi 4 for facial recognition using its camera module. The safe itself is custom-designed and 3D printed to house the servo lock securely. With this system, we provide security against unauthorized access, ensuring the safekeeping of valuable assets.

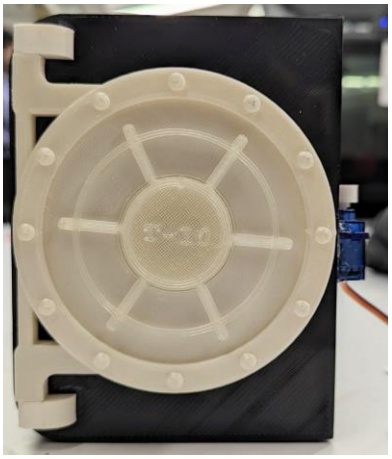
[2] METHOD

[2.1] DESIGN APPROACH

a) **Component Housing:** A component housing has been designed to organize and store all the electronic components required for the safe system. The housing serves as a centralized unit to securely contain the Raspberry Pi, Arduino Uno, servo motor, and buzzer. Additionally, the screen is mounted on the wall with its display facing outward, providing visual feedback to users. Adjacent to the screen, the RFID module is installed inside the wall on the right side, ensuring seamless integration with the overall design. Furthermore, the camera is strategically positioned on top of the screen to facilitate facial recognition. This design configuration optimizes space utilization while maintaining a streamlined and aesthetically pleasing appearance for a safe system



b) **The Vault:** In addition to the component housing, our design features a demo vault-like structure adjacent to it, creating a tangible representation of the safe system's functionality. This vault mimics the appearance of a real safe, with a lock mechanism controlled by the servo motor inside the housing. The servo motor's operation is synchronized with the authentication process, ensuring that the vault opens and closes accordingly based on the authentication status. This interactive demonstration provides users with a visual and tactile understanding of how the safe system operates.



[2.2] CIRCUIT AND ELECTRONICS

1. **Raspberry Pi 4:**

   a. **Raspberry Pi 4 board:** The Raspberry Pi 4 is a compact single-board computer with a quad-core 64-bit processor running at 1.5 GHz. It comes with options for 2GB, 4GB, or 8GB of RAM and features built-in Wi-Fi, Bluetooth, HDMI ports for up to 4K display, USB ports, Gigabit Ethernet, GPIO pins, and support for micro-SD storage. It runs various operating systems and is widely used for diverse projects due to its versatility, affordability, and extensive community support. In this project, the Raspberry Pi camera module captures high-definition images of the user's face. These images are then processed by the Raspberry Pi to perform facial recognition, verifying the user's identity. This module's integration enhances the security of the safe by providing an additional layer of authentication.

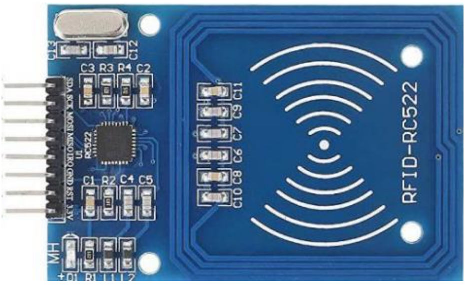
b. **Raspberry Pi camera module:** The Raspberry Pi camera V3 module is a compact camera designed for use with Raspberry Pi computers. It offers high-definition imaging, connects via a ribbon cable, and is compatible with all Raspberry Pi models. With software support for various applications, it's ideal for photography, video recording, and computer vision projects. In this project, the Raspberry Pi camera module captures the user's face for facial recognition. Its high-definition imaging ensures clear capture, enabling reliable authentication. The camera is used to take an image of the person and the images are analyzed to verify the user's identity. This adds an extra layer of security to the safe system, enhancing its effectiveness against unauthorized access.

c. **7" Screen:** The screen underneath the camera serves multiple functions in this project. Firstly, it displays "Access Denied" when the RFID card is not recognized during the initial authentication stage, providing immediate feedback to the user. Secondly, upon successful RFID card recognition, the screen activates the camera for facial recognition through a preview so that the user can align their face to the camera based on the preview. This transition between RFID authentication and facial recognition enhances user experience by guiding them through the authentication process. Overall, the screen plays a crucial role in providing real-time feedback and facilitating user interaction with the safe system.

2. Arduino:

a. **Arduino Uno:** The Arduino Uno, a cornerstone of this project, serves as the primary control unit, seamlessly orchestrating the authentication process and unlocking mechanism of the safe. In our project, the Arduino Uno is like the brain of the safe. It's in charge of checking if the user's credentials are correct, first with the RFID reader, and then unlocking the safe if everything's okay. It's like the controller that makes sure only the right people can get into the safe, keeping everything secure. It then controls servo motors to unlock the safe upon successful authentication.

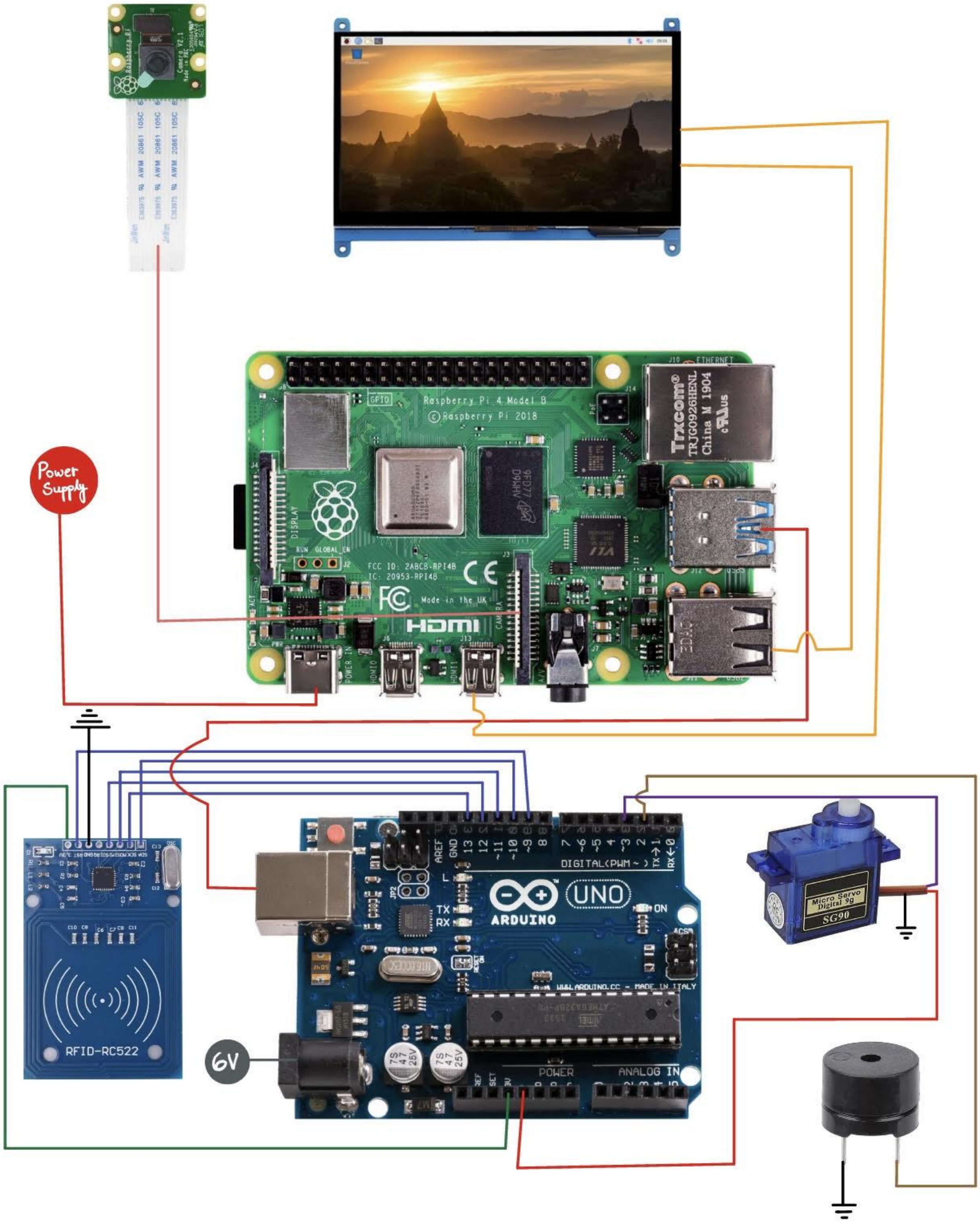
b. **RC522 RFID reader module:** The RC522 RFID reader module is a compact device used to read RFID (Radio Frequency Identification) tags or cards. It operates on the principle of electromagnetic induction to communicate with RFID tags within its proximity. In this project, the RC522 RFID reader module plays a crucial role in the initial stage of user authentication for accessing the safe. It interacts with RFID cards, validating user credentials by reading the unique identification data stored on them. This process serves as the first layer of security, ensuring that only authorized users can proceed with further authentication steps to unlock the safe.



c. **Servo Motor:** A servo motor is a precise rotary actuator used for controlling angular position. In this project, servo motors are utilized to physically unlock the safe door upon successful authentication. Signals from the Arduino Uno trigger the servo motors, ensuring secure access to the safe's contents.

d. **Piezo Buzzer for auditory alerts:** A Piezo Buzzer is an electronic component that produces sound when an electric current is applied to it. In this project, the Piezo Buzzer is used to provide auditory alerts during the authentication process. It emits distinct sounds to signify successful or failed authentication attempts, enhancing user feedback and system usability.

[2.3] CIRCUIT DIAGRAM



[3] OPERATION

INITIALIZATION:

- When activated, the system initializes all components, such as the RFID reader, Raspberry Pi 4, Arduino Uno, servo motors, and facial recognition camera module, ensuring they are ready for operation.

INITIAL VALIDATION:

- Initially, users interact with the system by tapping their RFID credentials on the reader, initiating the authentication process.
- The system verifies these credentials against stored data and promptly provides feedback through auditory cues, such as specific sounds, further informing the user of the validation status using the piezo buzzer.
- If access is granted, it proceeds to facial recognition and if denied, it signals "access denied".

SECONDARY AUTHENTICATION:

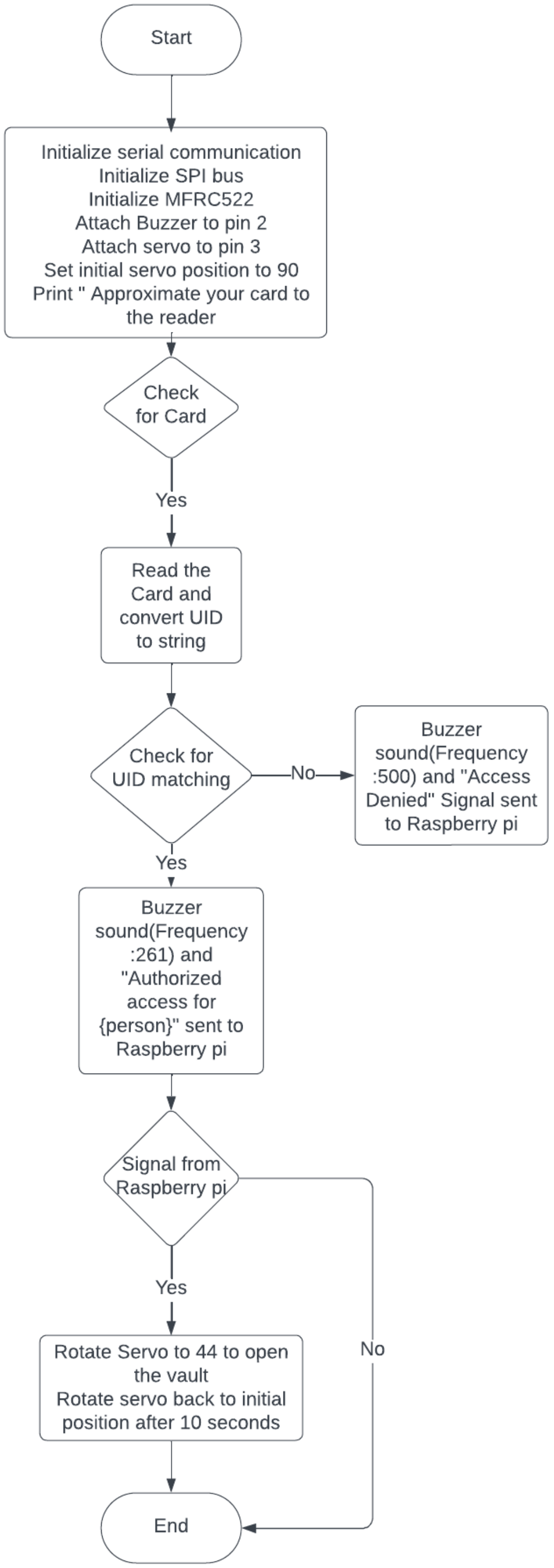
- Following successful RFID validation, users undergo facial recognition for an additional layer of security, further confirming their identity.
- The camera module captures the user's face, and the captured image is processed by Raspberry Pi to search for pre-trained data. It also compares the image associated with RFID and the captured image. If the captured image matches with the pre-trained data as well as with the image associated with RFID, then access is granted. Otherwise, access will be denied.
- If access is granted, it proceeds by prompting “Access granted” and if denied, it prompts "Access denied".

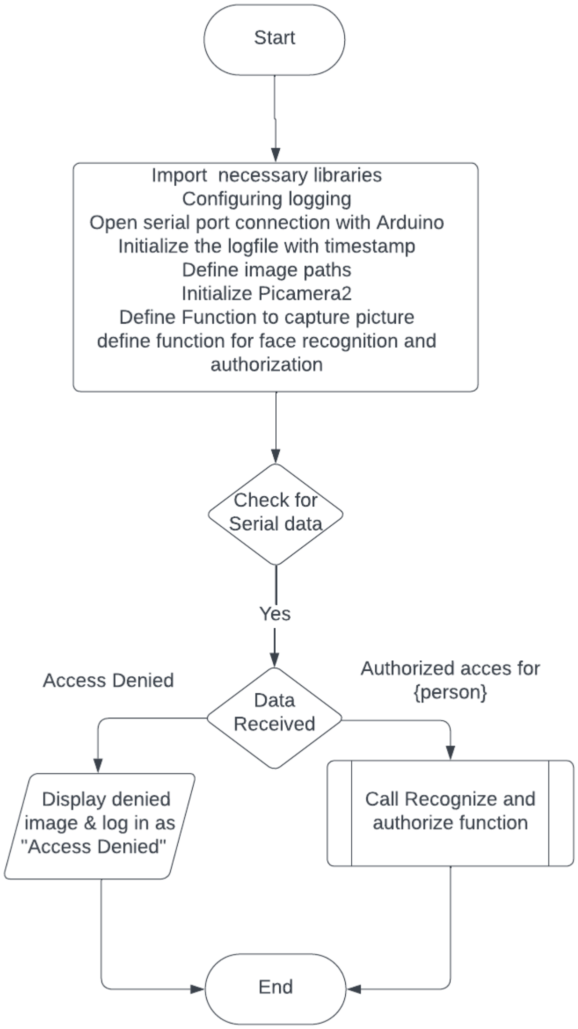
UNLOCKING MECHANISM:

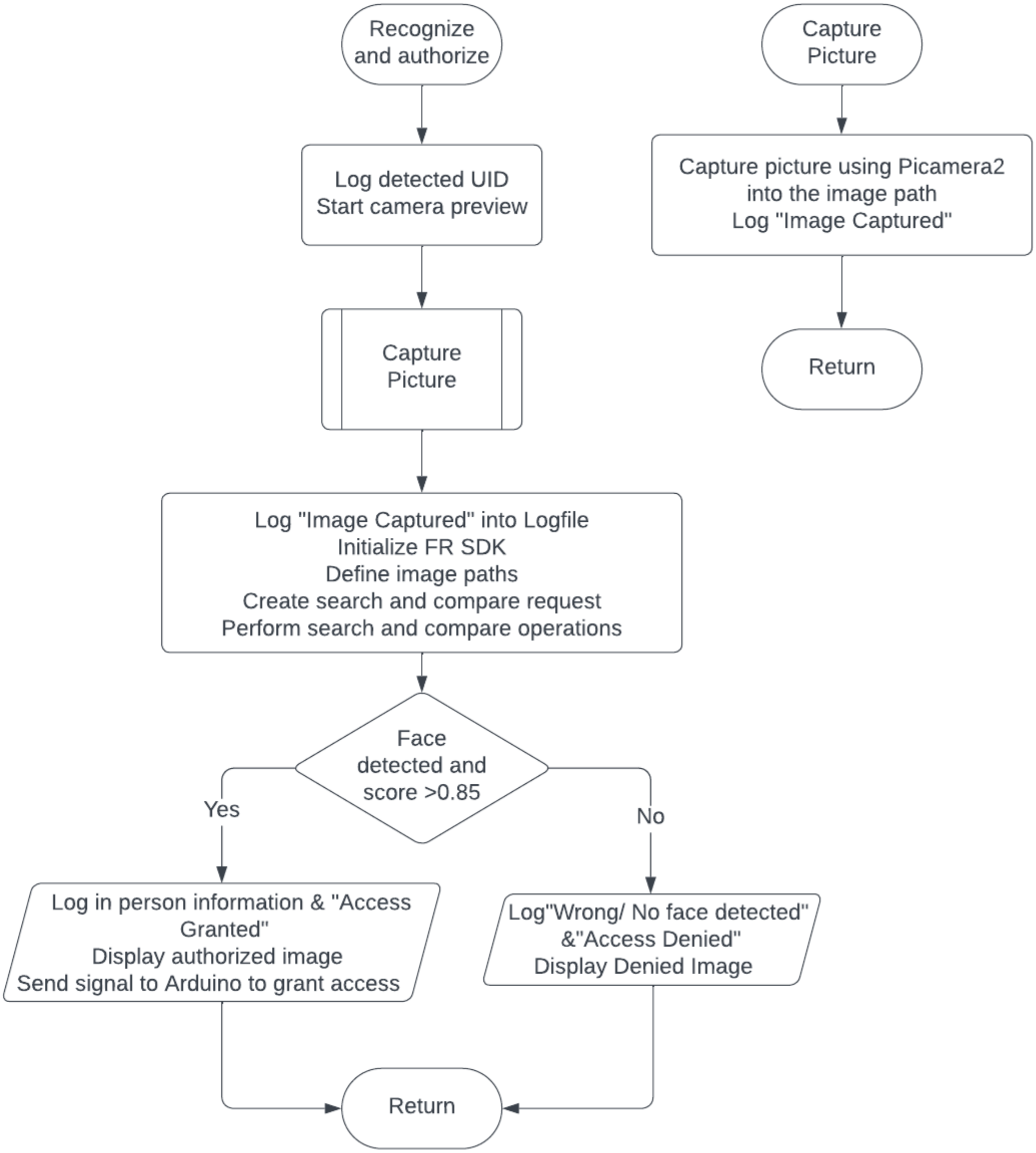
- If both RFID and facial authentication are successful, indicating the user's authorization, the Raspberry Pi sends signals to the Arduino to rotate the servo motor, instructing to unlock the safe's door.

LOGGING DATA: This feature is used to log the data along with the timestamps whenever the code runs. It stores the name of the person associated with the card, whenever the card is tapped. If the face is detected correctly, then it stores the person’s information and message “access granted”. If a wrong or no face is detected or a wrong card is tapped, then it logs the message “access denied”.

[4] FLOWCHART:

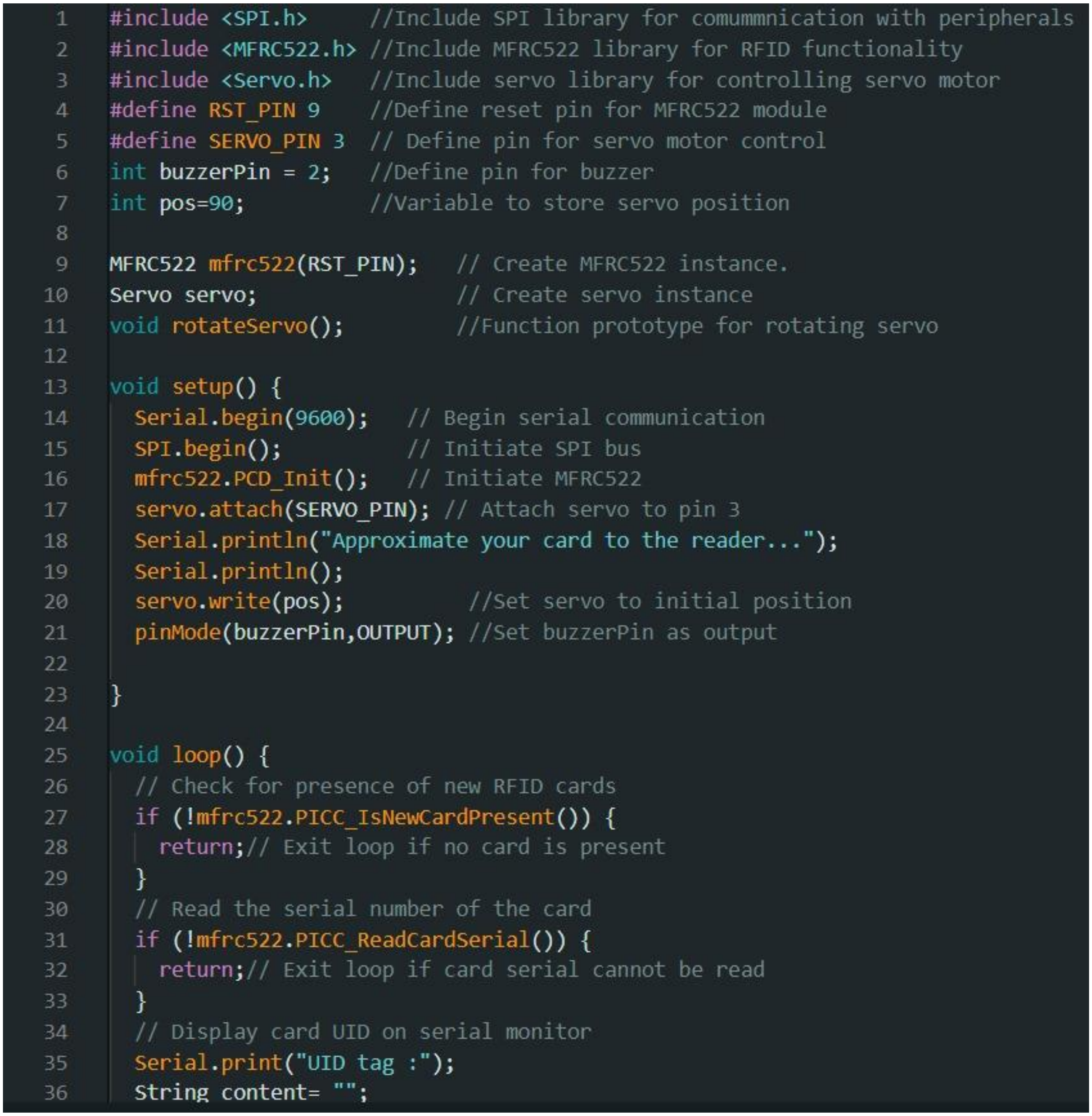

Start
Initialize serial communication
Initialize SPI bus
Initialize MFRC522
Attach Buzzer to pin 2
Attach servo to pin 3
Set initial servo position to 90
Print " Approximate your card to the reader
Check for Card
Yes
Read the Card and convert UID to string
Check for UID matching
No
Buzzer sound(Frequency :500) and "Access Denied" Signal sent to Raspberry pi
Yes
Buzzer sound(Frequency :261) and "Authorized access for {person}" sent to Raspberry pi
Signal from Raspberry pi
Yes
No
Rotate Servo to 44 to open the vault
Rotate servo back to initial position after 10 seconds
End


Start
Import necessary libraries
Configuring logging
Open serial port connection with Arduino
Initialize the logfile with timestamp
Define image paths
Initialize Picamera2
Define Function to capture picture
define function for face recognition and authorization
Check for Serial data
Yes
Access Denied
Data Received
Authorized acces for {person}
Display denied image & log in as "Access Denied"
Call Recognize and authorize function
End

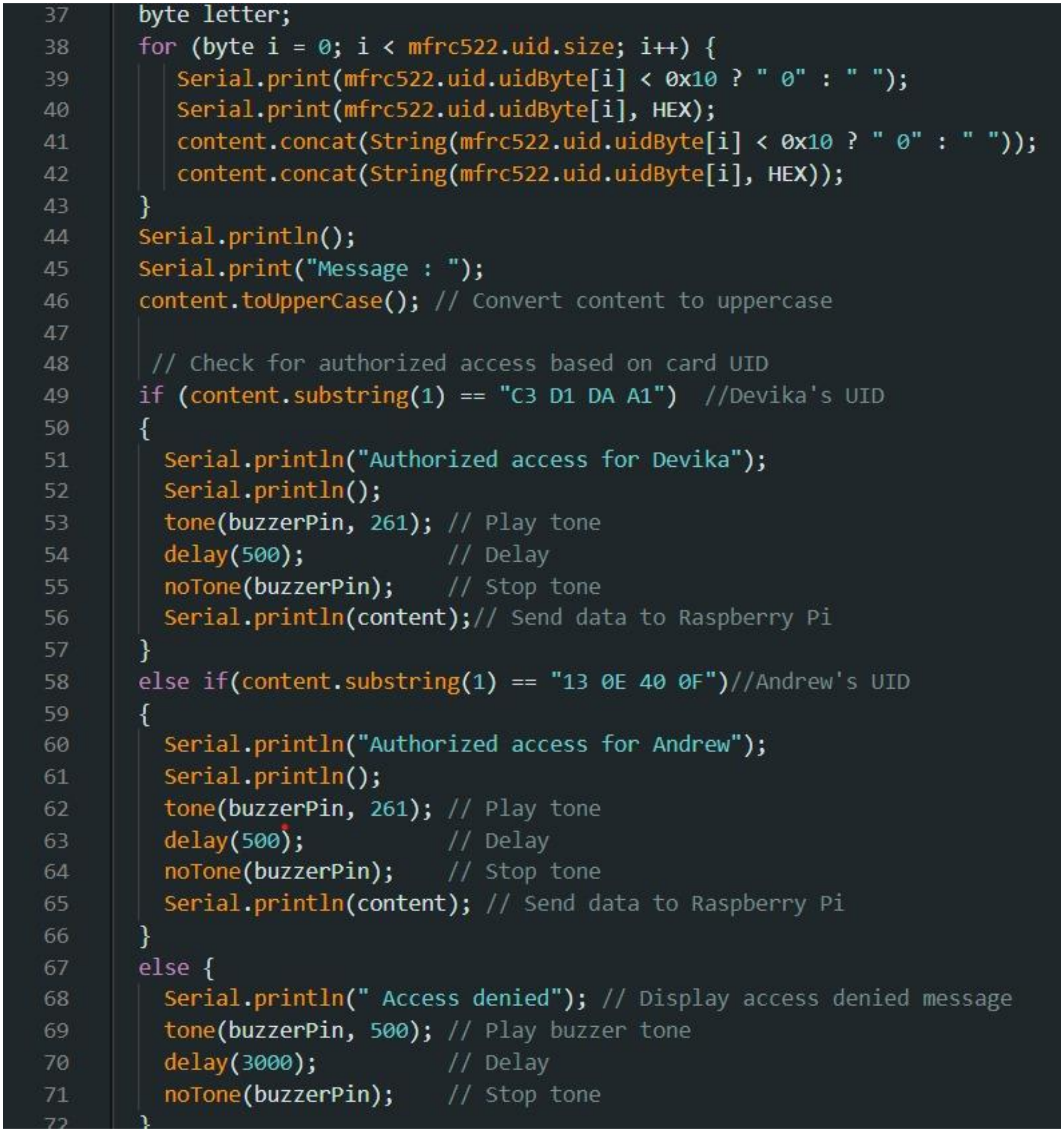

Recognize and authorize
Log detected UID
Start camera preview
Capture Picture
Log "Image Captured" into Logfile
Initialize FR SDK
Define image paths
Create search and compare request
Perform search and compare operations
Face detected and score >0.85
Yes
No
Log in person information & "Access Granted"
Display authorized image
Send signal to Arduino to grant access
Log"Wrong/ No face detected" &"Access Denied"
Display Denied Image
Return
Capture Picture
Capture picture using Picamera2 into the image path
Log "Image Captured"
Return

[5] CODE

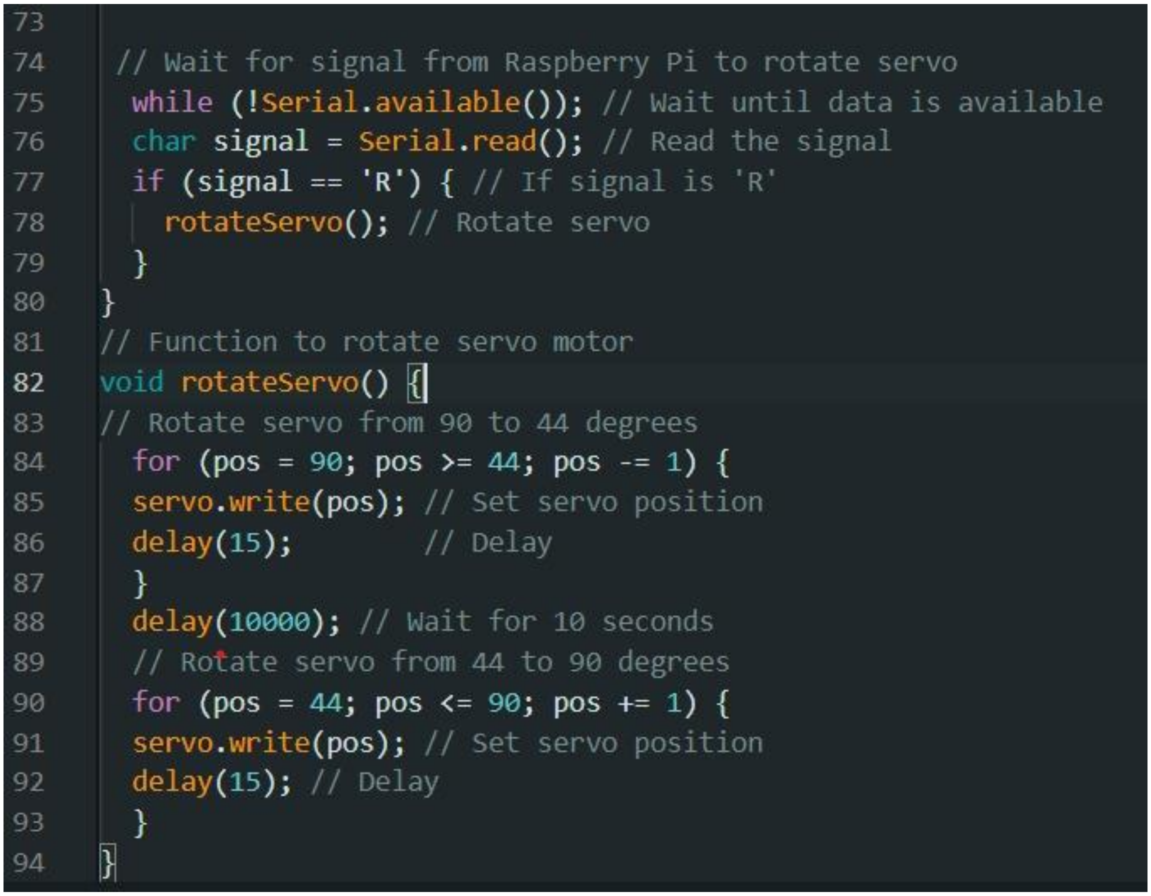
ARDUINO CODE:



```
#include <SPI.h>     //Include SPI library for comummnication with peripherals
#include <MFRC522.h> //Include MFRC522 library for RFID functionality
#include <Servo.h>   //Include servo library for controlling servo motor
#define RST_PIN 9    //Define reset pin for MFRC522 module
#define SERVO_PIN 3  // Define pin for servo motor control
int buzzerPin = 2;   //Define pin for buzzer
int pos=90;          //Variable to store servo position

MFRC522 mfrc522(RST_PIN);   // Create MFRC522 instance.
Servo servo;                // Create servo instance
void rotateServo();         //Function prototype for rotating servo

void setup() {
  Serial.begin(9600);   // Begin serial communication
  SPI.begin();          // Initiate SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522
  servo.attach(SERVO_PIN); // Attach servo to pin 3
  Serial.println("Approximate your card to the reader...");
  Serial.println();
  servo.write(pos);          //Set servo to initial position
  pinMode(buzzerPin,OUTPUT); //Set buzzerPin as output

}

void loop() {
  // Check for presence of new RFID cards
  if (!mfrc522.PICC_IsNewCardPresent()) {
    return;// Exit loop if no card is present
  }
  // Read the serial number of the card
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;// Exit loop if card serial cannot be read
  }
  // Display card UID on serial monitor
  Serial.print("UID tag :");
  String content= "";
```

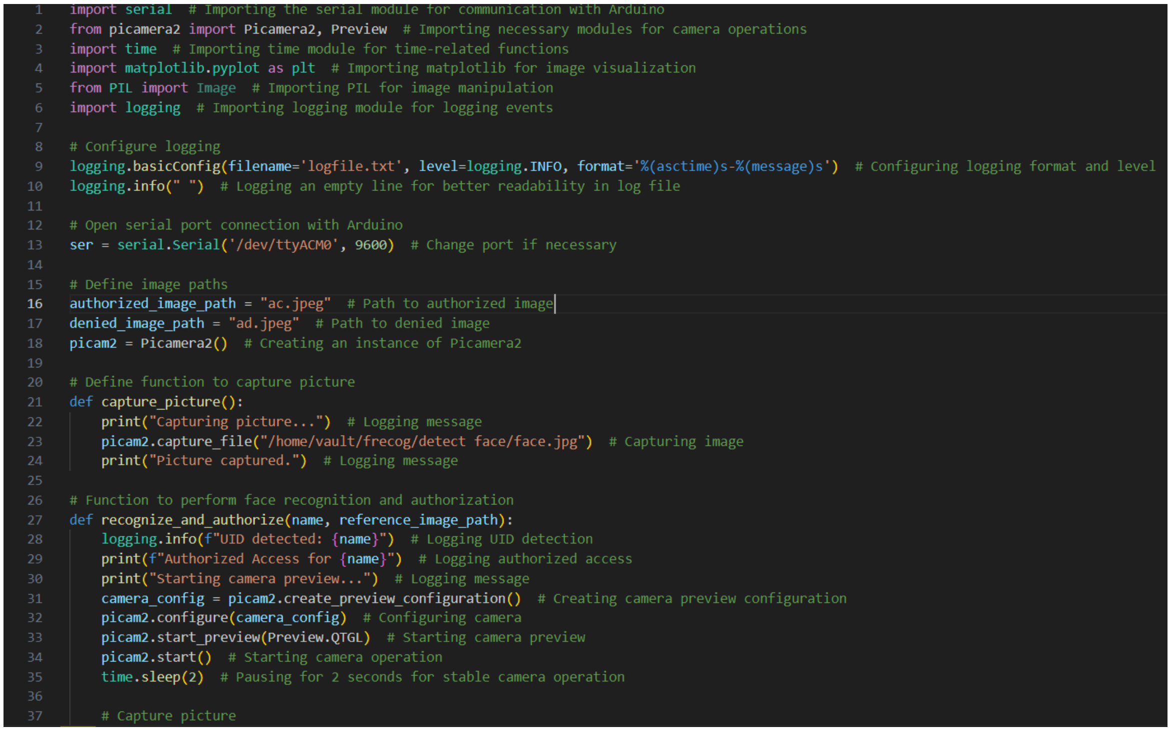


```
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++) {
     Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
     Serial.print(mfrc522.uid.uidByte[i], HEX);
     content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase(); // Convert content to uppercase

   // Check for authorized access based on card UID
  if (content.substring(1) == "C3 D1 DA A1")  //Devika's UID
  {
    Serial.println("Authorized access for Devika");
    Serial.println();
    tone(buzzerPin, 261); // Play tone
    delay(500);           // Delay
    noTone(buzzerPin);    // Stop tone
    Serial.println(content);// Send data to Raspberry Pi
  }
  else if(content.substring(1) == "13 0E 40 0F")//Andrew's UID
  {
    Serial.println("Authorized access for Andrew");
    Serial.println();
    tone(buzzerPin, 261); // Play tone
    delay(500);           // Delay
    noTone(buzzerPin);    // Stop tone
    Serial.println(content); // Send data to Raspberry Pi
  }
  else {
    Serial.println(" Access denied"); // Display access denied message
    tone(buzzerPin, 500); // Play buzzer tone
    delay(3000);          // Delay
    noTone(buzzerPin);    // Stop tone
  }
```

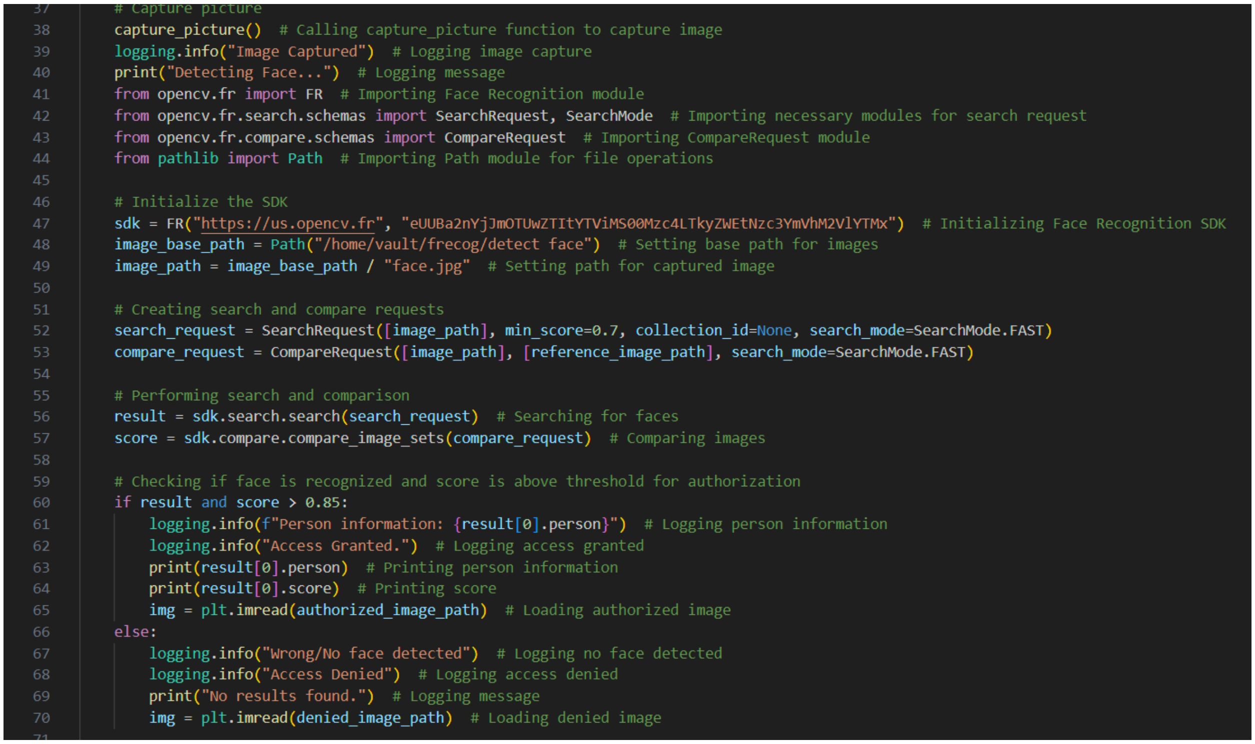


```
 // Wait for signal from Raspberry Pi to rotate servo
  while (!Serial.available()); // Wait until data is available
  char signal = Serial.read(); // Read the signal
  if (signal == 'R') { // If signal is 'R'
    rotateServo(); // Rotate servo
  }
}
// Function to rotate servo motor
void rotateServo() {
// Rotate servo from 90 to 44 degrees
  for (pos = 90; pos >= 44; pos -= 1) {
  servo.write(pos); // Set servo position
  delay(15);         // Delay
  }
  delay(10000); // Wait for 10 seconds
  // Rotate servo from 44 to 90 degrees
  for (pos = 44; pos <= 90; pos += 1) {
  servo.write(pos); // Set servo position
  delay(15); // Delay
  }
}
```

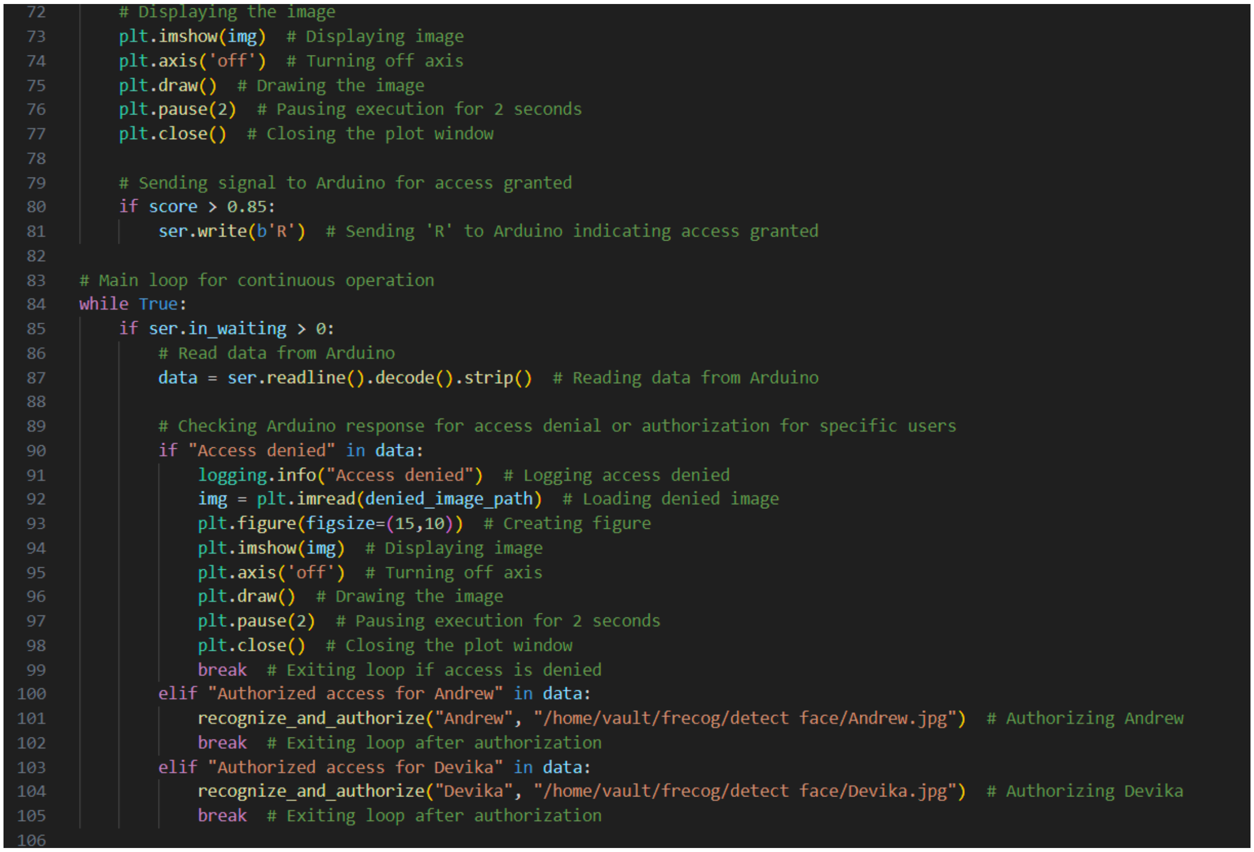
RASPBERRY PI CODE:



```python
import serial  # Importing the serial module for communication with Arduino
from picamera2 import Picamera2, Preview  # Importing necessary modules for camera operations
import time  # Importing time module for time-related functions
import matplotlib.pyplot as plt  # Importing matplotlib for image visualization
from PIL import Image  # Importing PIL for image manipulation
import logging  # Importing logging module for logging events

# Configure logging
logging.basicConfig(filename='logfile.txt', level=logging.INFO, format='%(asctime)s-%(message)s')  # Configuring logging format and level
logging.info(" ")  # Logging an empty line for better readability in log file

# Open serial port connection with Arduino
ser = serial.Serial('/dev/ttyACM0', 9600)  # Change port if necessary

# Define image paths
authorized_image_path = "ac.jpeg"  # Path to authorized image
denied_image_path = "ad.jpeg"  # Path to denied image
picam2 = Picamera2()  # Creating an instance of Picamera2

# Define function to capture picture
def capture_picture():
    print("Capturing picture...")  # Logging message
    picam2.capture_file("/home/vault/frecog/detect face/face.jpg")  # Capturing image
    print("Picture captured.")  # Logging message

# Function to perform face recognition and authorization
def recognize_and_authorize(name, reference_image_path):
    logging.info(f"UID detected: {name}")  # Logging UID detection
    print(f"Authorized Access for {name}")  # Logging authorized access
    print("Starting camera preview...")  # Logging message
    camera_config = picam2.create_preview_configuration()  # Creating camera preview configuration
    picam2.configure(camera_config)  # Configuring camera
    picam2.start_preview(Preview.QTGL)  # Starting camera preview
    picam2.start()  # Starting camera operation
    time.sleep(2)  # Pausing for 2 seconds for stable camera operation

    # Capture picture
```



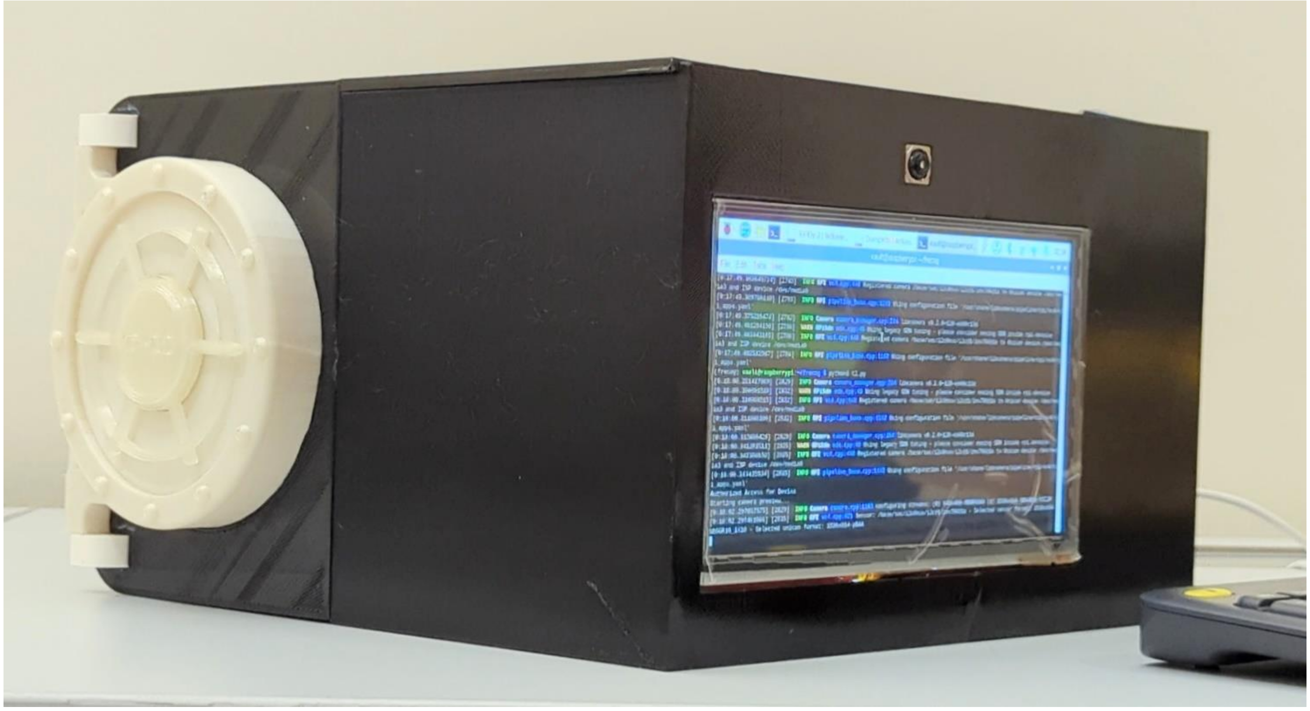
```
    # Capture picture
    capture_picture()  # Calling capture_picture function to capture image
    logging.info("Image Captured")  # Logging image capture
    print("Detecting Face...")  # Logging message
    from opencv.fr import FR  # Importing Face Recognition module
    from opencv.fr.search.schemas import SearchRequest, SearchMode  # Importing necessary modules for search request
    from opencv.fr.compare.schemas import CompareRequest  # Importing CompareRequest module
    from pathlib import Path  # Importing Path module for file operations

    # Initialize the SDK
    sdk = FR("https://us.opencv.fr", "eUUBa2nYjJmOTUwZTItYTViMS00Mzc4LTkyZWEtNzc3YmVhM2VlYTMx")  # Initializing Face Recognition SDK
    image_base_path = Path("/home/vault/frecog/detect face")  # Setting base path for images
    image_path = image_base_path / "face.jpg"  # Setting path for captured image

    # Creating search and compare requests
    search_request = SearchRequest([image_path], min_score=0.7, collection_id=None, search_mode=SearchMode.FAST)
    compare_request = CompareRequest([image_path], [reference_image_path], search_mode=SearchMode.FAST)

    # Performing search and comparison
    result = sdk.search.search(search_request)  # Searching for faces
    score = sdk.compare.compare_image_sets(compare_request)  # Comparing images

    # Checking if face is recognized and score is above threshold for authorization
    if result and score > 0.85:
        logging.info(f"Person information: {result[0].person}")  # Logging person information
        logging.info("Access Granted.")  # Logging access granted
        print(result[0].person)  # Printing person information
        print(result[0].score)  # Printing score
        img = plt.imread(authorized_image_path)  # Loading authorized image
    else:
        logging.info("Wrong/No face detected")  # Logging no face detected
        logging.info("Access Denied")  # Logging access denied
        print("No results found.")  # Logging message
        img = plt.imread(denied_image_path)  # Loading denied image
```



```
    # Displaying the image
    plt.imshow(img)  # Displaying image
    plt.axis('off')  # Turning off axis
    plt.draw()  # Drawing the image
    plt.pause(2)  # Pausing execution for 2 seconds
    plt.close()  # Closing the plot window

    # Sending signal to Arduino for access granted
    if score > 0.85:
        ser.write(b'R')  # Sending 'R' to Arduino indicating access granted

# Main loop for continuous operation
while True:
    if ser.in_waiting > 0:
        # Read data from Arduino
        data = ser.readline().decode().strip()  # Reading data from Arduino

        # Checking Arduino response for access denial or authorization for specific users
        if "Access denied" in data:
            logging.info("Access denied")  # Logging access denied
            img = plt.imread(denied_image_path)  # Loading denied image
            plt.figure(figsize=(15,10))  # Creating figure
            plt.imshow(img)  # Displaying image
            plt.axis('off')  # Turning off axis
            plt.draw()  # Drawing the image
            plt.pause(2)  # Pausing execution for 2 seconds
            plt.close()  # Closing the plot window
            break  # Exiting loop if access is denied
        elif "Authorized access for Andrew" in data:
            recognize_and_authorize("Andrew", "/home/vault/frecog/detect face/Andrew.jpg")  # Authorizing Andrew
            break  # Exiting loop after authorization
        elif "Authorized access for Devika" in data:
            recognize_and_authorize("Devika", "/home/vault/frecog/detect face/Devika.jpg")  # Authorizing Devika
            break  # Exiting loop after authorization
```

[6] RESULTS

- The project effectively integrates RFID technology, facial recognition, and servo-controlled locking mechanisms to create a secure safe system.
- Authentication methods, including RFID validation and facial recognition, ensure that only authorized users can access the safe.
- User-friendly interfaces such as the RFID reader and visual feedback display enhance the ease of interaction with the safe system.
- A demo vault-like structure provides a tangible demonstration of the safe system's functionality, enhancing user understanding and engagement.
- The organized component housing ensures efficient installation and operation of electronic components.
- This project demonstrates a practical application of modern technology in safeguarding physical assets effectively, showcasing the potential for real-world security solutions.



[7] CONCLUSION

In conclusion, our project successfully achieves its objective of creating a secure and user-friendly safe system by integrating advanced technologies such as RFID, facial recognition, and servo-controlled locking mechanisms. Through meticulous design and implementation, we have developed an authentication process that ensures only authorized users can access the safe. The inclusion of user-friendly interfaces and a tangible demonstration further enhances user engagement and comprehension. Additionally, the organized component housing demonstrates our attention to detail in optimizing space utilization. Overall, this project serves as a practical example of how modern technology can be utilized to safeguard physical assets effectively, showcasing its potential for real-world security applications.

[8] REFERENCE

[1] https://docs.opencv.fr/python/

[2] https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf

[3] github.com/raspberrypi/picamera2

[4] https://docs.opencv.org/

[5] https://github.com/miguelbalboa/rfid

[6] Lecture Notes