MTE 301 FINAL PROJECT
Automated Bike Lock system

Osman Asif - 501235477

Mohammed Dar - 501246407

Hamzah Faisal - 501237784

Muntadher Alzuaibel - 501231793

Professor - Robin Chabbra

Executive Summary **2**
Introduction **3**
Background 4
Project description 5
System/Product description 6
Team Decomposition & work completed **7**
Hardware development - Hamzah & Mohammed 7
SolidWorks design - Mohammed 8
Software development - Hamzah & Mohammed & Osman & Muntadher 8
Report - Osman & Muntadher 9
Outline Of the Report 9
Problem Statement **10**
Software Solution Design **11**
Phases development 11
Interconnected Architecture 12
Pseudo codes 14
Testing & debugging **16**
Conclusion **17**
Appendix **18**

Executive Summary

The automated Bike lock system is a modern and secure solution to the issue of bike theft, providing an effective means of safety for your bike, particularly in an urban environment where it is an ongoing issue where traditional locking systems are ineffective and lead to stolen bikes. The project focuses on replacing the traditional system through an innovative approach, consisting of a motorized locking mechanism, a keypad for unlocking and locking to ensure maximum security as well as a secondary opening system through facial recognition. The design ensures the enhanced security and safety of the bike and easy accessibility and use when locking and unlocking. The system operates using two main functions, a facial scanner and a keypad interface, allowing the user to unlock and lock their bicycle by either entering a password or setting a facial recognition. The dual option provides the user with flexibility and versatility in which option is best suited for them, and ensures reliability if one of the systems fails. Furthermore, the locking mechanism is powered by a motorized system and the system as a whole is controlled by C++ software to perform secure operations as well as python. Initially the design proposed for advanced features such as the RFID-scanner based unlocking, GPS tracking, and a mobile app interface. After extensive research, discussions and implementation issues, the features were to be excluded from the final design, the RFID scanner proved to be lackluster when in comparison to the facial recognition and was thus, left out due to security issues, as well as a more complex solution available. Moreover, the lack of time and complexity led to the the removal of the other two systems, rather then branching out to various systems, keeping it to a minimal allowed for more focus on effort on fewer systems, by focusing on a design that implements less systems, it allowed us to create a more enhanced, sustainable system that provides the most secure outcome.

Introduction

The increasing need for reliable and secure bike locking systems has highlighted the limitations of traditional methods, which often fail to meet modern security demands. To address this issue, this project focuses on the development of an Automated Bike Lock System that combines advanced technology with a user-friendly interface. The system is designed to enhance bicycle security while providing a convenient and reliable solution for users. It features a motorized locking mechanism and incorporates two unlocking methods: a keypad interface for password entry and a facial recognition system. These dual options offer flexibility for users and ensure dependable operation even if one method encounters an issue.

The system's design and operation are entirely programmed using C++, which integrates the motorized locking mechanism with the facial recognition and keypad modules. While initial concepts for the system included additional features such as RFID unlocking, GPS tracking, and a mobile app interface, these were omitted to prioritize the refinement of the core functionalities. This streamlined approach allowed the project to focus on developing a robust and practical system that aligns with the objectives of enhancing security, reliability, and ease of use.

The project involved several stages, including the development of hardware components, the integration of software functionalities, and rigorous testing to ensure smooth operation.

Background

The history of the bike lock system shows a continuous effort to enhance bicycle security. Since the 19th century, when bicycles were starting to get popular, basic padlock systems and heavy iron chains were used as security. In the 1970s, Micheal Zane, the creator of Kryptonite, invented a U-lock system, which made it almost impossible for criminals to break the lock to steal the bicycle (Kryptonite, 2024). Today, numerous types of locking systems provide a varying level of protection and security. U-locks are still popular due to their durability and reliability, whereas chain locks are flexible but can be heavy to carry around. Cable locks, although portable and lightweight, are more prone to being cut, whereas folding locks provide a balanced solution where connected hardened steel plates are utilized (Scharm & Cortez, 2024). In recent years, improvements in technology has led to the development of smart locks that include integrated bluetooth, Wi-Fi, GPS tracking, and alarm systems that trigger when interfered with, allowing for increased protection, security and convenience for bicycle owners. These advances highlight the continued evolution of bicycle lock systems, which combine old methods with modern technology to meet and address security concerns.

Kryptonite. (2024). *History*. @kryptonite. https://www.kryptonitelock.com/en/company-history/company-history.html

Scharm, A., & Cortez, K. (2024). *The 10 best bike locks of 2024 - bike lock reviews*. The 10 Best Bike Locks for Securing Your Ride Anywhere. https://www.bicycling.com/bikes-gear/a20017112/best-bike-locks/

Project description

The goal of this project is to create a secure and user-friendly bike lock system that enhances bicycle security while maintaining ease of use. The design integrates advanced features, including a motorized locking mechanism and two unlocking methods: a keypad for passcode entry and a facial recognition system. These features work together to provide both reliability and flexibility, ensuring that the system remains functional even if one method is unavailable.

The project's main focus is the use of C++ to develop the necessary algorithms and implement the core functionality of the locking system. Throughout the development process, project goals were refined and adapted to align with practical constraints and feedback, allowing for the creation of a robust and effective design. Every aspect of the system is programmed and controlled using C++, which serves as the foundation for integrating hardware components and ensuring smooth system operation.

This approach prioritizes simplicity, reliability, and security, resulting in a modern solution tailored to address the challenges of traditional bike locks while providing an innovative alternative for everyday users.

System/Product description

The Automated Bike Lock System is a security solution powered by Arduino, offering two methods for unlocking: keypad input and facial recognition technology. This system integrates a motorized locking mechanism controlled by a servo motor with advanced facial recognition to provide enhanced security and ease of use.

The keypad interface allows users to unlock the bike by entering a four-digit passcode. Once the entered passcode matches the stored code in the Arduino's memory, a signal is sent to the servo motor, which rotates 90 degrees clockwise to unlock the bike. To relock the system, the user can press the '0' key on the keypad, prompting the servo to return to its original position. This provides a straightforward and dependable solution for securing the bike.

The facial recognition feature utilizes MediaPipe's Face Mesh technology to recognize a pre-set user profile stored in the system. During the setup process, the user's facial landmarks are recorded and saved as a fixed profile in a *face_profiles.pkl* file. During operation, the system processes live webcam footage to extract facial landmarks, comparing them to the stored profile. If the detected face matches the saved profile with an accuracy above 70%, a "TRIGGER" command is sent to the Arduino through serial communication, which activates the servo motor to unlock the bike.

This dual unlocking system provides users with flexibility by allowing them to choose between keypad entry and facial recognition. The integration of C++ for Arduino-based locking control and Python for facial recognition ensures seamless communication between hardware and

software. By combining security and user-friendly operation, this system offers a reliable and modern solution to address bicycle theft.

Team Decomposition & work completed

The development of the automated lock system has involved significant processes in a variety of areas, including hardware development, software development, SolidWorks design, and documentation.

Hardware development

- Motorized lock
    - A motor-driven mechanism was designed to secure and release the lock, (open and close the lock) based on the user commands.
- Keypad module
    - A digital keypad was integrated to allow user input to open the lock and set their own private key
- Camera module
    - Installed to capture real-time facial images for authentication using the Haar Cascade algorithm
- Integration of the systems
    - Laptop runs both the arduino which includes the keypad lock as well as the facial recognition program.

- Essentially like a switch case where either the user is inputting a key on the keypad or using facial recognition and one of the two systems will be working at a time.

SolidWorks design

- Design
  - Housing for motorized lock, keypad using SolidWorks and then 3D printing it (found in the appendix)
  - Created derailed assembly diagrams to guide hardware integration

Software development

- Keypad lock
  - Developed a password authentication system for the keypad which handles the opening and closing as well as the input from the user
  - Combining with the motor to work seamlessly to lock and unlock based off input
- facial recognition
  - Algorithm breaks down input image into regions and computes features by comparing pixel intensities within regions
  - Trained to recognize complex features of face
  - If a match is found based off the data the user sets up, it unlocks motor
  - Integration of the systems

Report

- Documented project objectives, problem statement, and design considerations
- Detailed software architecture, hardware integration and results
- Including pseudo code, pictures and explanations
- Collaborated as a team to ensure all aspects are done effectively

Outline Of the Report

This report provides a comprehensive explanation of the Automated Bike Lock System project. It begins with an introduction to the project's background and objectives, giving context to the need for a secure and modern bike locking solution. The report then details the system's design, including the integration of hardware components and the implementation of software functionalities.

Moreover, the report outlines the development processes, including hardware assembly, software programming, and system integration. Challenges faced during these stages are discussed alongside the solutions that were implemented to ensure the system's functionality and reliability. The testing and debugging phases are also presented, showcasing the steps taken to refine and validate the system's performance.

The report concludes with a reflection on the project's outcomes, summarizing the achievements and offering recommendations for future improvements to enhance the system's capabilities further.

Problem Statement

Bike theft in the Toronto area has increased by a staggering 411% in 2020. The countless easy methods used in breaking through a key-based locking system have deemed this system incapable of keeping your belongings secure. These traditional lock systems can easily be picked and require the owner to insert the key which may be seen as an inconvenience. These old-school lock systems lack innovation and technology that can boost the potential of this idea. This project aims to change that. To bring innovation and technology to the world of lock systems by incorporating features that guarantee the user safety and peace of mind.

Software Solution Design

The challenges in developing the automated system include ensuring the dual layer security system that is reliable and provides seamless integration between each other as well as the hardware components. The proposed solutions:

1. **Dual authentication**
    a. Facial recognition
    b. Keypad system
2. **Motorized locking system**
    a. Servo motor used for precise locking and unlocking
3. **System reliability**
    a. Error handling ensuring smooth operation

Phases development

1. **Requirement Specification**
    a. Define facial recognition and keypad functionalities
    b. Specify what hardware is needed, servo motor, camera, keypad, etc
2. **Design & Planning**
    a. Develop algorithms for facial recognition and password authentication (2 systems)
    b. Establish a communication between the systems using serial commands
    c. Modular software architecture for easy integration
3. **Implementation**
    a. Write python code for facial recognition

b. Develop arduino code to handle keypad input & motor control

c. Integrate the components using serial communication

4. **Testing & debugging**

    a. Validate facial recognition accuracy under various conditions

        i. Challenges regarding mapping of facial recognition as well as mapping errors

    b. Test password authentication and motor operations

        i. The password would not input correctly, the keypad was not receiving input

            1. The wires were not connected properly

        ii. The motor would not move when the key was entered

            1. The code did not correctly give an operation when the input was given

5. **Integration**

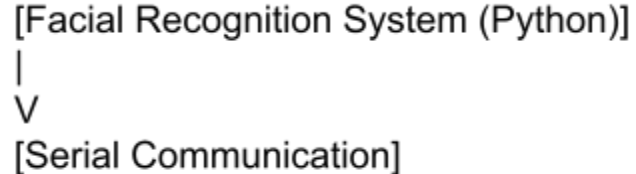
    a. Combine facial recognition & keypad system to control motorized lock

    b. Implement error handling

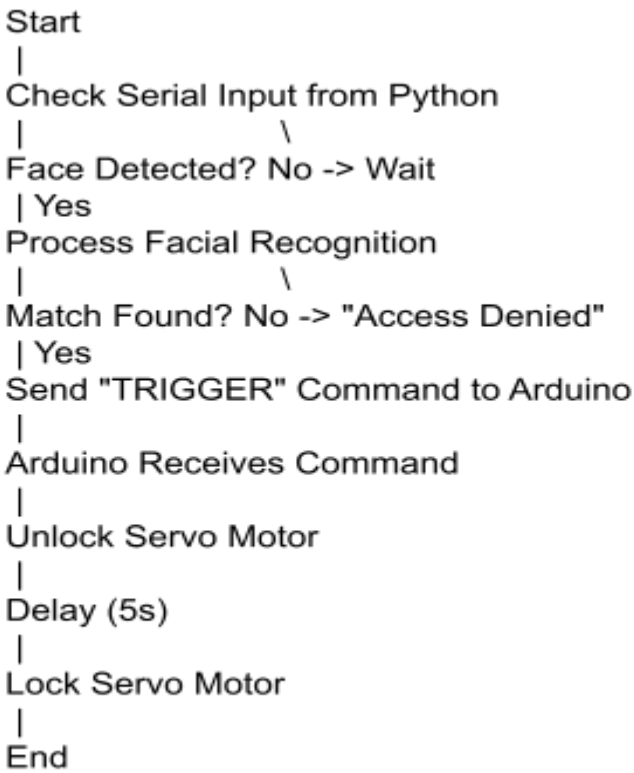
Interconnected Architecture

1. Facial recognition module through python that detects and authenticates user faces using haar cascade algorithm and sens and unlock signal to the arduino
2. Keypad module using arduino C++ and allows users to input a password for unlocking
3. Motorized system allows for the two systems to be opened through a motor

Block diagram



Flowchart



Pseudo codes

1. **Facial recognition**

Initialize serial communication with arduino

Load stored face profiles

While webcam is active:

    Capture and process video frame

    Detect face landmarks using MediaPipe

    Compare landmarks with stored profiles

    IF match> 90% accuracy:

        Send “TRIGGER” command to arduino

        Display “access granted”

    ELSE

        Display”access denied”

End loop

2. **Keypad authentication**

Initialize servo motor connection & keypad connection

Set servo to locked position

LOOP:

    If serial input is “TRIGGER”

    Unlock servo motor rotating from 0 to 90 (clockwise)

    Wait 5 seconds

    Lock servo motor

Otherwise:

Check keypad input

Enter password

IF password matches

    Display “access granted”

    Unlock servo motor rotating from 0 to 90 (clockwise)

    Wait 5 seconds

    Lock motor

ELSE

    Display “access denied”

Clear keypad input after process

End loop

Testing & debugging

During the testing stages of this assignment we ran into multiple errors and bumps in the road in terms of our code, wiring and hardware. Firstly, the code was sending an incorrect signal to the servo motor and turning it in the opposite direction of where we wanted it to go, to resolve this issue, we changed the lockServo.write( ); and added a zero in the brackets in order to send a

clockwise signal to the servo motor. This small change helped fix the issue of incorrect rotation for the locking mechanism. Additionally, while developing the mechanical components for the system we ran into many 3D printing problems which led us to create multiple designs for our project. These issues involved printing errors, incorrect fitment, and non-moveable parts. Using trial and error we ended with a final working product. Lastly, when wiring the system to integrate the servo and the keypad there were difficulties in the wire positioning, arrangement, and overall current issues. To overcome these issues, we watched youtube tutorials, trial and tested using our own skills and created many test cases which eventually led us to find the perfect wiring situation for our design. Overall these challenges during the testing stage helped us enhance our skills in both programming and engineering design.

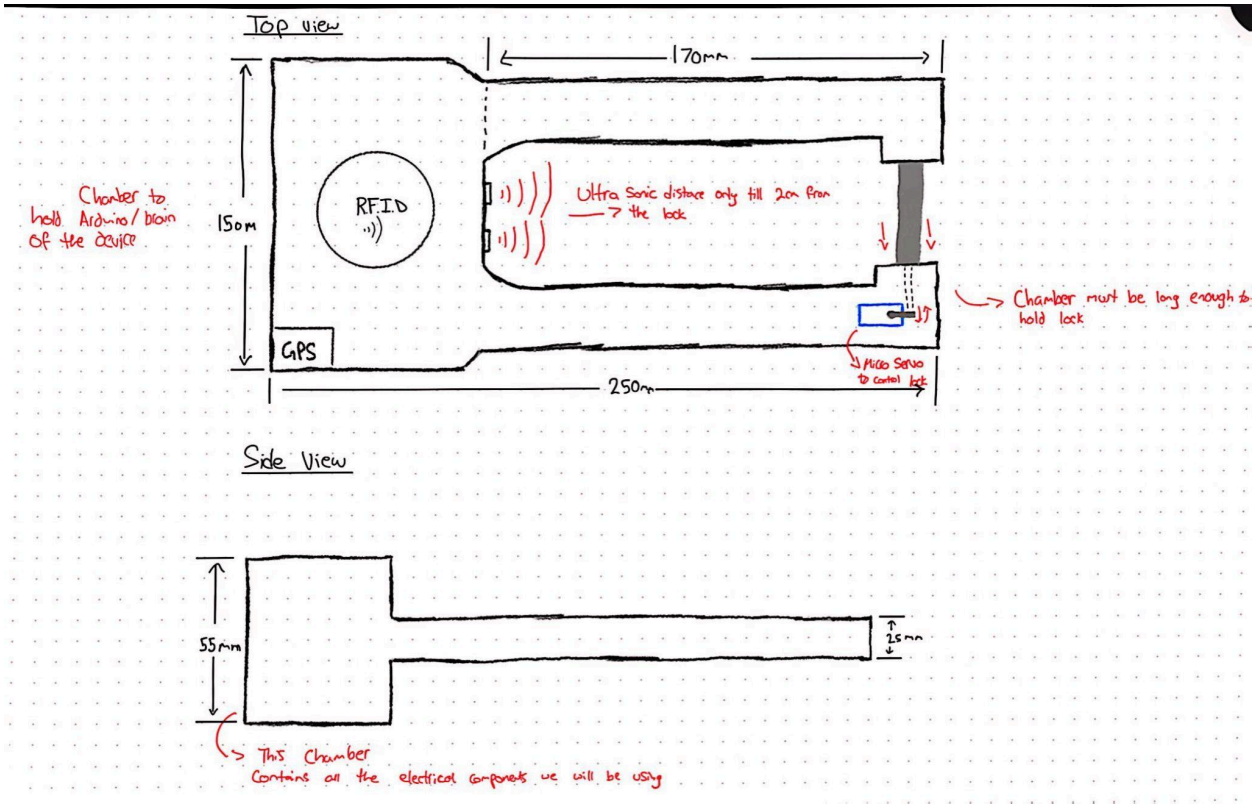
Conclusion

The automated bike lock system has successfully addressed the challenges of modern bicycle security by integrating a dual-layer authentication mechanism with a motorized locking system. Moreover, by using facial recognition as well as a keypad system it provides versatility and reliability and user convenience. The project demonstrates our team's ability to navigate technical challenges such as ensuring facial recognition is accurate and precise to the profiles that are set. The keypad was ensured to resolve the input issues mentioned were all resolved through rigorous

debugging and iterative improvements. While the project achieved its primary objective of creating and establishing a clear two-system design, certain systems were excluded from the final design such as the proposed features of RFID, GPS tracking and a mobile app interface. The decision to exclude these systems resulted in a more refined, higher quality product that effectively integrates the two systems together. For future improvements enhanced recognition algorithms through the use of a raspberry pi would allow for a higher quality and more functional system. Furthermore, incorporating machine learning based facial recognition would exceed all expectations of the design and make an adaptable product. Furthermore, adding multi-user support by creating multiple accounts for passwords and facial recognition would allow for multiple users to use the same product. In conclusion the bike lock system effectively combines the modern technology of automation with practical designs to address the limitations and safety concerns with traditional bike locks, the dual authentication system creates a diverse solution to this problem.

Appendix

Figure 1. Product Sketch:



Product:

Figure 2. design

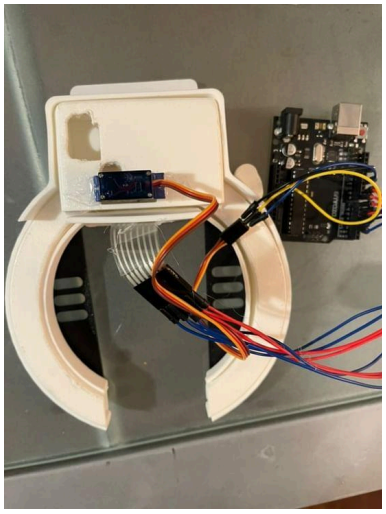
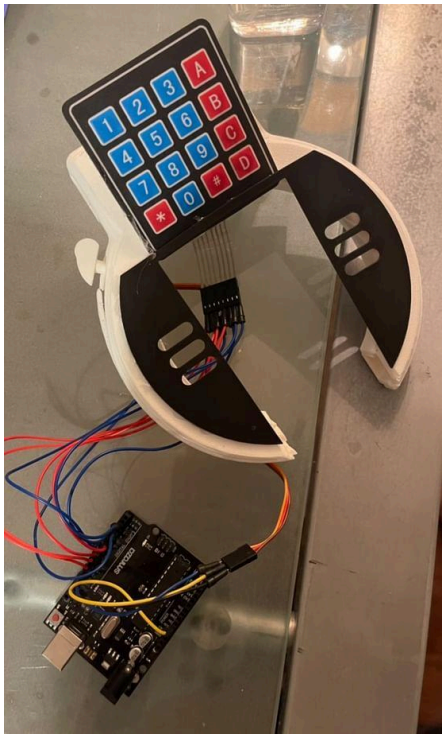


Figure 3. Design

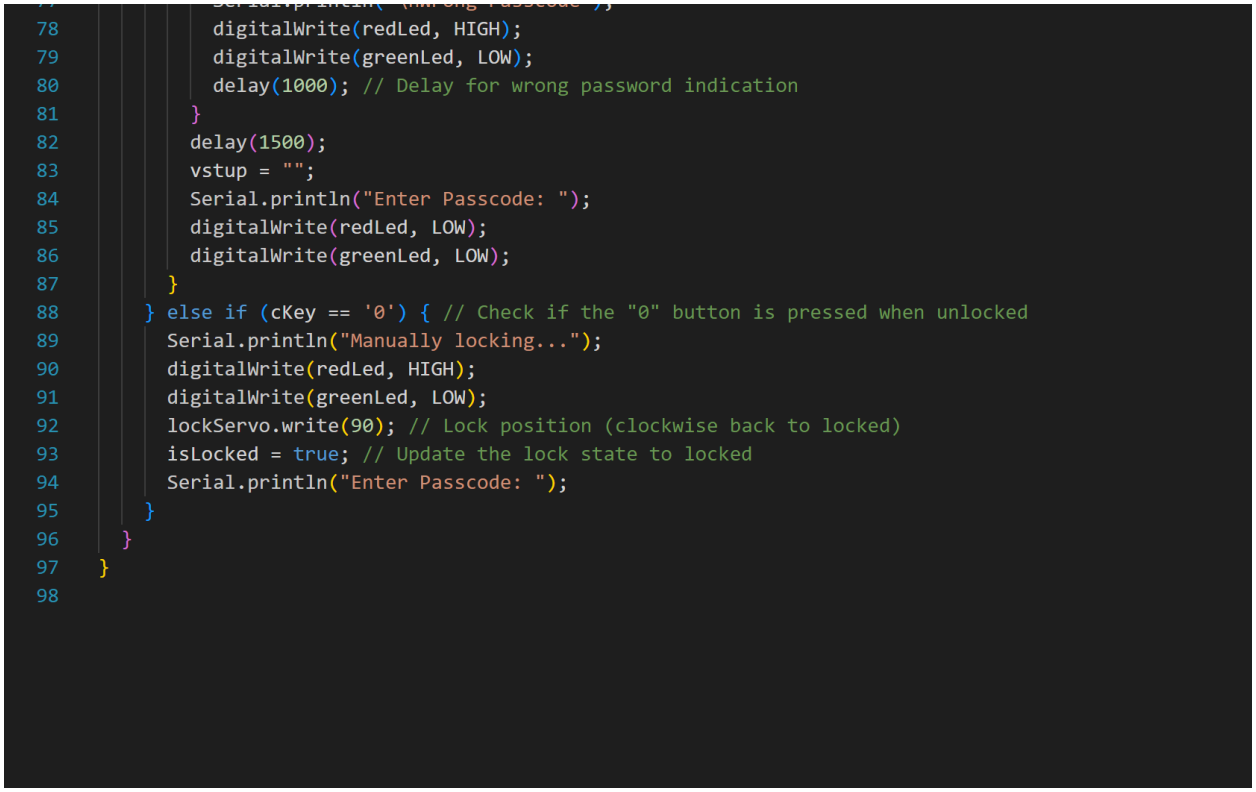


Code:

Figure 4. Code for arduino lock

```ino
#include <Keypad.h>
#include <Servo.h> // servo library

const int row = 4;
const int col = 4;
int redLed = 12;
int greenLed = 13;
int servoPin = 11;    // Servo control pin
int signalPin = 12;   // Signal pin from Raspberry Pi

char numPad[row][col] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPin[row] = {9, 8, 7, 6};
byte colPin[col] = {5, 4, 3, 2};

String password = "4567";
String vstup = "";

bool isLocked = true; // Track whether the servo is locked

Keypad cKeypad = Keypad(makeKeymap(numPad), rowPin, colPin, row, col);
Servo lockServo;

void setup()
{
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(signalPin, INPUT); // Signal from Raspberry Pi
  lockServo.attach(servoPin);
  lockServo.write(90); // Set initial servo position to locked (90°)
  Serial.begin(9600);
  Serial.print("Enter Passcode: ");
}
```

```
void loop()
{
  // Check for signal from Raspberry Pi
  if (digitalRead(signalPin) == HIGH) {
    Serial.println("Signal received from Raspberry Pi. Unlocking...");
    digitalWrite(redLed, LOW);
    digitalWrite(greenLed, HIGH);
    lockServo.write(0); // Unlock position (clockwise)
    delay(5000); // Keep unlocked for 5 seconds
    lockServo.write(90); // Lock position (clockwise back to locked)
    digitalWrite(greenLed, LOW);
    isLocked = true; // Reset lock state
    Serial.println("Enter Passcode: ");
  }

  // Process input from keypad
  char cKey = cKeypad.getKey();

  if (cKey != NO_KEY) {
    if (isLocked) {
      if (vstup.length() < 4) {
        Serial.print("*");
        vstup = vstup + cKey;
      }

      if (vstup.length() == 4) {
        delay(1500); // Wait for 1.5 seconds
        if (password == vstup) {
          Serial.println("\nCorrect Passcode. Unlocking...");
          digitalWrite(redLed, LOW);
          digitalWrite(greenLed, HIGH);
          lockServo.write(0); // Unlock position (clockwise)
          isLocked = false; // Update the lock state to unlocked
          delay(5000); // Keep unlocked for 5 seconds
          lockServo.write(90); // Lock position (clockwise back to locked)
          isLocked = true; // Reset lock state
        } else {
          Serial.println("\nWrong Passcode");
```



```
            digitalWrite(redLed, HIGH);
            digitalWrite(greenLed, LOW);
            delay(1000); // Delay for wrong password indication
          }
          delay(1500);
          vstup = "";
          Serial.println("Enter Passcode: ");
          digitalWrite(redLed, LOW);
          digitalWrite(greenLed, LOW);
        }
      } else if (cKey == '0') { // Check if the "0" button is pressed when unlocked
        Serial.println("Manually locking...");
        digitalWrite(redLed, HIGH);
        digitalWrite(greenLed, LOW);
        lockServo.write(90); // Lock position (clockwise back to locked)
        isLocked = true; // Update the lock state to locked
        Serial.println("Enter Passcode: ");
      }
    }
  }

```

Figure 5. Code for facial recognition

```
import cv2
import mediapipe as mp
import numpy as np
import pickle
import serial
import time

# Initialize Serial Communication with Arduino
arduino = serial.Serial(port='/dev/tty.usbmodem1101', baudrate=9600, timeout=1)  # Replace with your Arduino's port
time.sleep(2)  # Wait for the connection to establish

# Initialize MediaPipe Face Mesh
mp_face_mesh = mp.solutions.face_mesh
mp_drawing = mp.solutions.drawing_utils

# File to store face profiles
PROFILE_FILE = "/Users/owner/Desktop/FaceRecognitionProject/face_profiles.pkl"

# Load or initialize profiles
try:
    with open(PROFILE_FILE, "rb") as f:
        face_profiles = pickle.load(f)
except FileNotFoundError:
    face_profiles = {}

# Function to save a new face profile
def save_profile(name, landmarks):
    face_profiles[name] = landmarks
    with open(PROFILE_FILE, "wb") as f:
        pickle.dump(face_profiles, f)
    print(f"Profile saved for {name}!")

# Function to recognize a face and calculate accuracy
def recognize_profile(landmarks):
    closest_name = None
    closest_distance = float('inf')
    closest_accuracy = 0

    for name, saved_landmarks in face_profiles.items():
        # Calculate Euclidean distance
        distance = np.linalg.norm(np.array(saved_landmarks) - np.array(landmarks))
        accuracy = max(0, 1 - distance) * 100  # Convert to percentage
        if distance < closest_distance:  # Find the closest match
            closest_name = name
            closest_distance = distance
            closest_accuracy = accuracy

    return closest_name, closest_accuracy

# Function to send a command to Arduino
def send_to_arduino(command):
    print(f"Sending to Arduino: {command}")
    arduino.write(f"{command}\n".encode())

# Webcam feed
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Could not access webcam.")
    exit()

# Use MediaPipe Face Mesh
with mp_face_mesh.FaceMesh(max_num_faces=1, refine_landmarks=True, min_detection_confidence=0.5) as face_mesh:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Convert frame to RGB
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = face_mesh.process(rgb_frame)
```

```
        # Convert frame to RGB
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = face_mesh.process(rgb_frame)

        if results.multi_face_landmarks:
            for face_landmarks in results.multi_face_landmarks:
                # Draw face mesh
                mp_drawing.draw_landmarks(
                    frame,
                    face_landmarks,
                    mp_face_mesh.FACEMESH_TESSELATION,
                )

                # Extract landmarks
                landmarks = [(lm.x, lm.y, lm.z) for lm in face_landmarks.landmark]

                # Recognize profile and calculate accuracy
                recognized_name, accuracy = recognize_profile(landmarks)

                if recognized_name:
                    cv2.putText(frame, f"{recognized_name} ({accuracy:.2f}%)", (50, 50),
                                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
                    if accuracy > 90:
                        send_to_arduino("TRIGGER")  # Send trigger signal to Arduino
                else:
                    cv2.putText(frame, "Unknown Face", (50, 50),
                                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        # Display the frame
        cv2.imshow("Face Recognition", frame)

        # Check for user input
        key = cv2.waitKey(1) & 0xFF
        if key == ord('q'):  # Quit
            break
        elif key == ord('n'):  # Save new profile
            name = input("Enter name for new profile: ")
            save_profile(name, landmarks)

cap.release()
cv2.destroyAllWindows()
arduino.close()
```

Figure 6. Integration of systems code

```
const int outputPin = 13;  // Use pin 13 to trigger the circuit (e.g., LED or relay)

void setup() {
  Serial.begin(9600);       // Start serial communication
  pinMode(outputPin, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');  // Read incoming command
    if (command == "TRIGGER") {
      digitalWrite(outputPin, HIGH);  // Turn on the pin
      delay(1000);                    // Keep it on for 1 second
      digitalWrite(outputPin, LOW);   // Turn off the pin
    }
  }
}
```