Electronics Workshop I

Final Evaluation

Automatic Door Lock

Team 11

2024112001 Kshitij Patkar

2024112005 Harry Jain

TA: S S Ananya Varma Ganapathiraju

Background

Need for Smart Security Solutions

- Rising demand for secure and automated access control systems in homes, offices, and industrial spaces.

- Traditional lock-and-key mechanisms are prone to theft, loss, and unauthorized duplication.

Background

Incorporating Technology for Usability

- Integration of user-friendly interfaces (keypad and Bluetooth app) for easy operation.

- Password-based systems eliminate the need for carrying physical keys.

Background

Key Features of Modern Locking Systems

- Customizable passwords for better control over access.
- Lockout mechanisms to prevent brute force attacks.
- Wireless control for convenience and remote access.

Background

Use Cases

- Residential doors where user control is essential.
- Office spaces requiring shared access but enhanced security.
- Industrial lockers with restricted access for staff only.

Problem Statement

Objective

Develop a secure and interactive door locking system for modern access control.

Key Features Addressed

- Password-based entry with real-time feedback via an LCD display.
- Lockout mechanism to prevent unauthorised access after repeated failures.
- Password customisation and secure management options.
- Integration with a Bluetooth-enabled app for remote control and enhanced convenience.

Proposed Design

Mapping of Requirements to Components:

- **Secure Password Input:**
  4x3 membrane keypad for user input.

- **Feedback and Display:**
  LCD module with I2C for clear, real-time interaction.

Proposed Design

Mapping of Requirements to Components:

- **Actuation Mechanism:**
  Solenoid lock controlled via a relay module for reliable door operation.

- **Wireless Control:**
  Bluetooth module integrated with an app for remote access.

Proposed Design

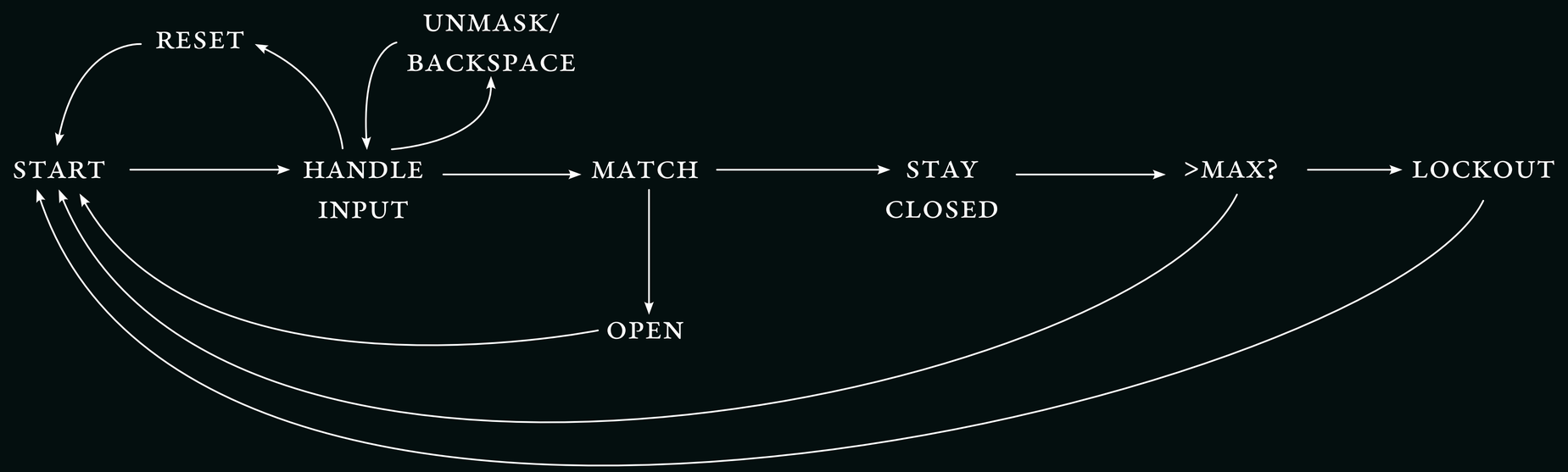
Design Choices

Initially considered a servo-motor-based system,
but opted for a solenoid lock due to:

- Independence from physical variables like friction.
- More reliable and compact design for locking/ unlocking mechanisms.

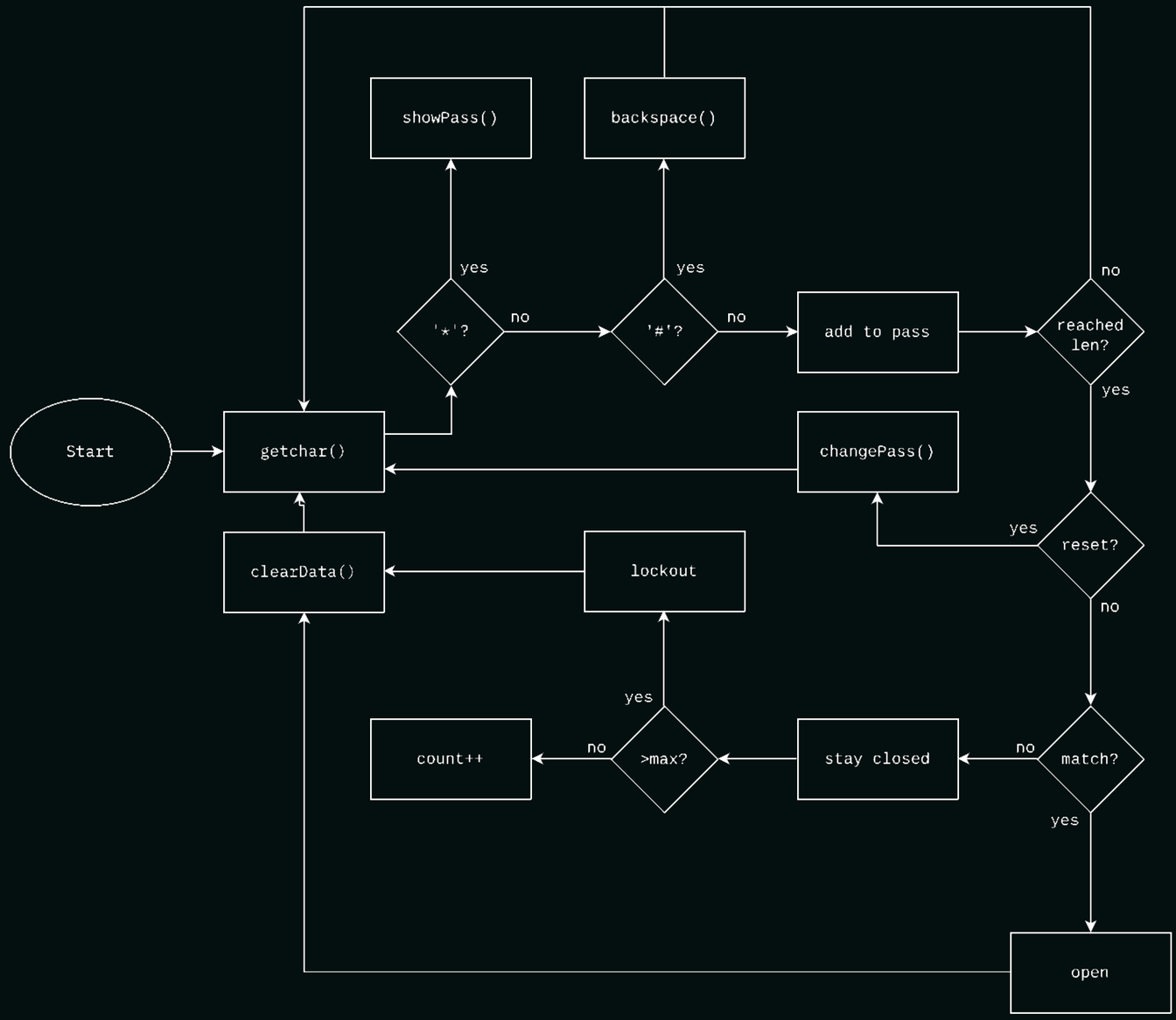
Proposed Design

Process Diagram



Proposed Design

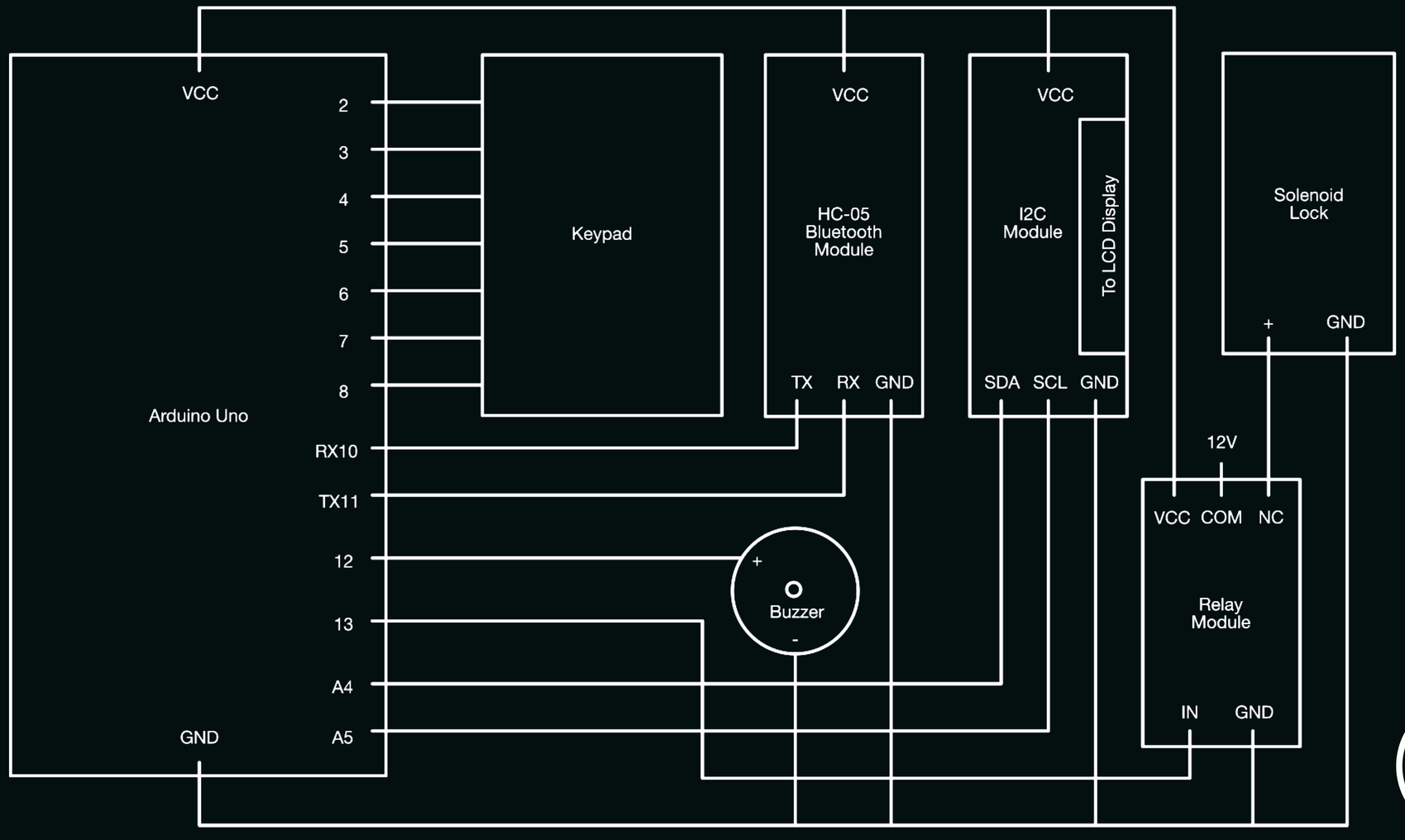
Block Diagram



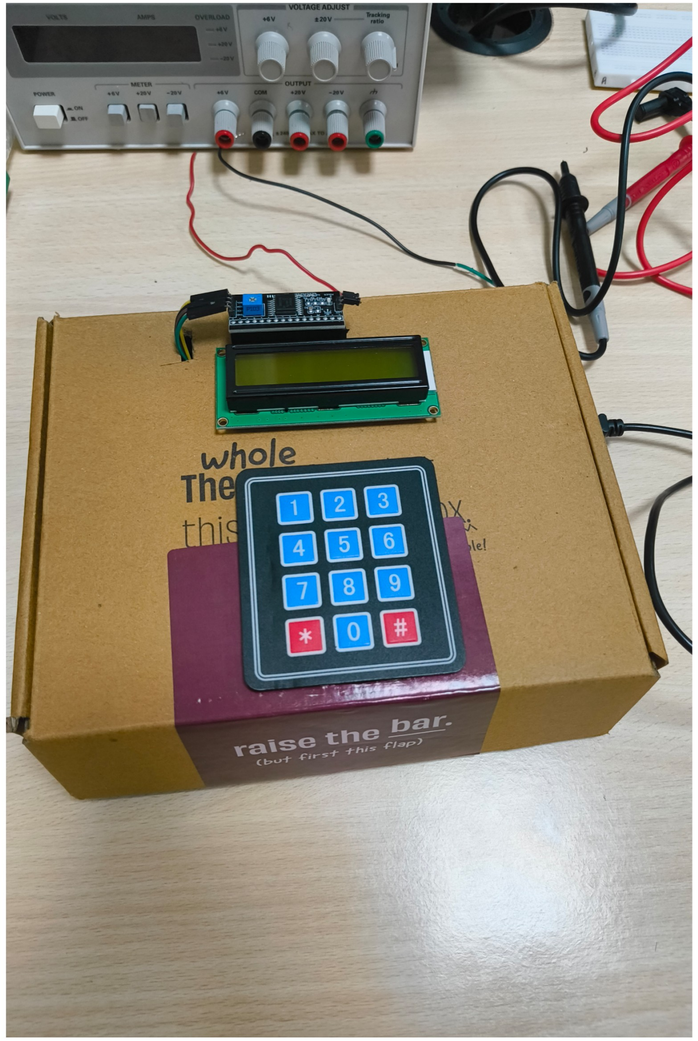
xi

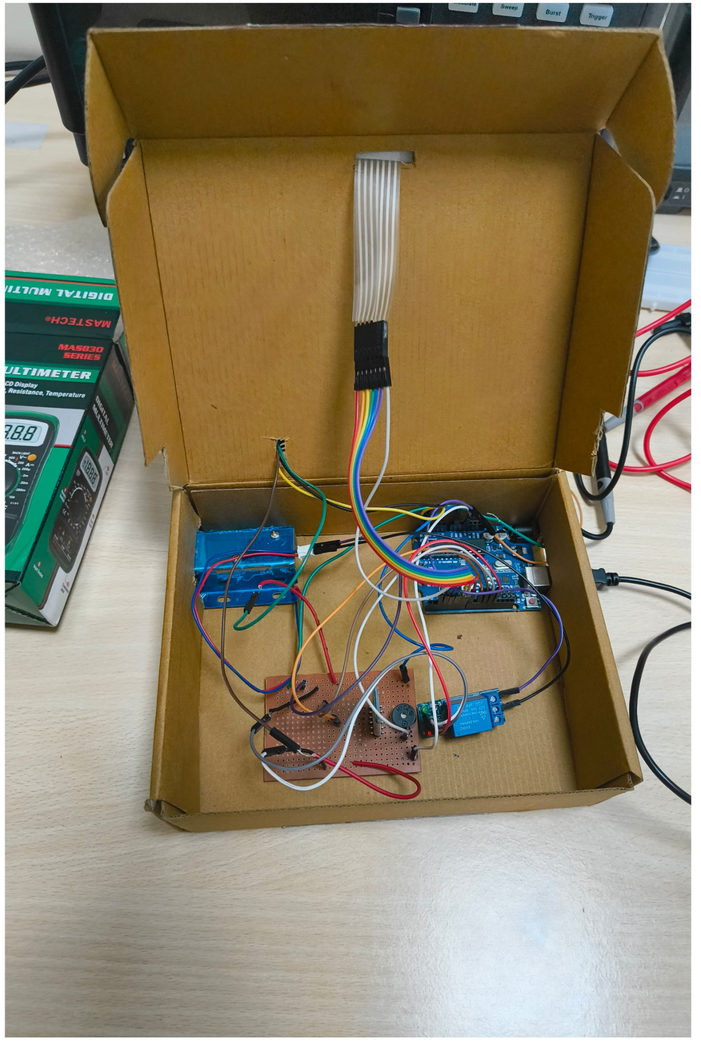
Proposed Design

Circuit Diagram



xii


VOLTAGE ADJUST
raise the bar.
(but first this flap)


DIGITAL MULTI
MASTECH
MAS830
SERIES
ULTIMETER

```
// Include Arduino Wire library for I2C
#include <Wire.h>
// Include LCD display library for I2C
#include <LiquidCrystal_I2C.h>
// Include Keypad library
#include <Keypad.h>
// Include Serial library
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11);  // 10 - Rx, 11 - Tx

// Length of password + 1 for null character
#define Password_Length 9

// Character to hold password input
char Data[Password_Length];

// Password
char Master[Password_Length] = "12345678";

// Pin connected to lock relay input
int lockOutput = 13;

// Pin connected to buzzer
int buzzerPin = 12;

// counter for incorrect attempts, max att
int countinc = 0;
#define MAXINC 3

// Counter for character entries
byte data_count = 0;

// Character to hold key input
char btKey;
char keyKey;
char customKey;

// Constants for row and column sizes
const byte ROWS = 4;
const byte COLS = 3;

// Array to represent keys on keypad
char hexaKeys[ROWS][COLS] = {
  { '1', '2', '3' },
  { '4', '5', '6' },
  { '7', '8', '9' },
  { '*', '0', '#' }
};

// Connections to Arduino
byte rowPins[ROWS] = { 8, 7, 6, 5 };
byte colPins[COLS] = { 4, 3, 2 };

// Create keypad object
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

// Create LCD object
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  // Setup LCD with backlight and initialize
  lcd.backlight();
  lcd.init();

  // Set lockOutput as an OUTPUT pin
  pinMode(lockOutput, OUTPUT);
  pinMode(buzzerPin, OUTPUT);

  //Serial
  mySerial.begin(9600);
  Serial.begin(115200);
}

void loop() {

  // Initialize LCD and print
  lcd.setCursor(0, 0);
  lcd.print("Enter Password:");

  // Look for keypress
  getChar();
  if (customKey) {

    // if *, show password
    if (customKey == '*') {
      showPass();
      goto shown;
    }

    // if #, backspace
    if (customKey == '#') {
      backspace();
      goto shown;
    }

    // Enter keypress into array and increment counter

    Data[data_count] = customKey;
    lcd.setCursor(data_count, 1);
    lcd.print('*');
    data_count++;
  }


  // See if we have reached the password length
  if (data_count == Password_Length - 1) {
    delay(500);
    lcd.clear();

    if (!strcmp("00000000", Data)) {
      changepass();
      goto endofloop;
    }

if (!strcmp(Data, Master)) {
      // Password is correct
      lcd.print("Correct");
      // Turn on relay for 5 seconds
      digitalWrite(buzzerPin, HIGH);
      delay(500);
      digitalWrite(buzzerPin, LOW);
      digitalWrite(lockOutput, HIGH);
      delay(5000);
      digitalWrite(lockOutput, LOW);
      digitalWrite(buzzerPin, HIGH);
      delay(100);
      digitalWrite(buzzerPin, LOW);

    } else {
      // Password is incorrect
      lcd.print("Incorrect");
      digitalWrite(buzzerPin, HIGH);
      delay(100);
      digitalWrite(buzzerPin, LOW);
      delay(50);
      digitalWrite(buzzerPin, HIGH);
      delay(100);
      digitalWrite(buzzerPin, LOW);

      delay(1000);
      countinc++;
      if (countinc == MAXINC) {
        lcd.clear();
        lcd.print("Too many in-");
        lcd.setCursor(0, 1);
        lcd.print("correct attempts.");
        for (int i = 0; i < 3; i++) {
          digitalWrite(buzzerPin, HIGH);
          delay(100);
          digitalWrite(buzzerPin, LOW);
          delay(50);
          digitalWrite(buzzerPin, HIGH);
          delay(100);
          digitalWrite(buzzerPin, LOW);
          delay(100);
        }

        delay(1000);
        lcd.clear();
        lcd.print("Try again later");
        // delay(7000);
        digitalWrite(buzzerPin, HIGH);
        delay(1500);
        digitalWrite(buzzerPin, LOW);
        delay(3500);
        countinc = 0;
      }
    }

endofloop:
    NULL;

    // Clear data and LCD display
    lcd.clear();
    clearData();
  }
```

```
shown:
  NULL;
}

void clearData() {
  // Go through array and clear data
  // put inside bracket to be cool
  while (data_count != 0) {
    Data[data_count] = 0;
    data_count--;
  }
  return;
}

void showPass() {
  // Clear screen, set cursor to start
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Check Password:");
  lcd.setCursor(0, 1);
  // print every char in password
  for (int i = 0; i < data_count; i++) {
    lcd.print(Data[i]);
  }
  // wait for one second
  delay(1500);
  // clear screen, set cursor to start
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Enter Password:");
  lcd.setCursor(0, 1);
  // print stars
  for (int i = 0; i < data_count; i++) {
    lcd.print('*');
  }
}

void backspace() {

  if (data_count) {
    data_count--;
    lcd.clear();
    lcd.print("Enter Password:");
    lcd.setCursor(0, 1);
    for (int i = 0; i < data_count; i++) {
      lcd.print('*');
    }
  }
}

void changepass() {
  clearData();
  lcd.clear();
  lcd.print("Reset Password?");
  lcd.setCursor(0, 1);
  lcd.print("Yes (1), No (0)");
  //delay(1000);
```

```
do {
  getChar();
} while (!customKey);
if (customKey == '1') {
  lcd.clear();
  lcd.print("Old Password:");
  for (int i = 0; i < 8; i++) {
    do {
      getChar();
    } while (!customKey);
    if (customKey) {

      // if *, show password
      if (customKey == '*') {
        showPass();
        i = i - 1;
        continue;
      }

      // if #, backspace

      if (customKey == '#') {
        lcd.setCursor(0, 0);
        lcd.print("Incorrect input");
        delay(500);
        lcd.setCursor(0, 0);
        lcd.print("Old Password:  ");
        i = i - 1;
        continue;
      }


      // Enter keypress into array and increment counter

      Data[data_count] = customKey;
      lcd.setCursor(data_count, 1);
      lcd.print('*');
      data_count++;
    }
  }
  if (!strcmp(Data, Master)) {
    // Password is correct
    lcd.clear();
    lcd.print("Correct");
    delay(1000);
    lcd.clear();
    lcd.print("Enter new");
    lcd.setCursor(0, 1);
    lcd.print("Password:");
    delay(1000);
    lcd.clear();
    lcd.print("Password:");

    for (int i = 0; i < 8; ++i) {

      do {
        getChar();
      } while (!customKey);
```

```
if (customKey) {

        // if *, show password
        if (customKey == '*') {
          lcd.setCursor(0, 0);
          lcd.print("Incorrect input");
          delay(500);
          lcd.setCursor(0, 0);
          lcd.print("Password:      ");
          i = i - 1;
          continue;
        }

        // if #, backspace
        if (customKey == '#') {
          lcd.setCursor(0, 0);
          lcd.print("Incorrect input");
          delay(500);
          lcd.setCursor(0, 0);
          lcd.print("Password:      ");
          i = i - 1;
          continue;
        }
        lcd.setCursor(i, 1);
        lcd.print('*');
        Master[i] = customKey;
      }
    }
    lcd.clear();
    lcd.print("Password Reset");
    lcd.setCursor(0, 1);
    lcd.print("Complete");
    delay(500);
    digitalWrite(buzzerPin, HIGH);
    delay(500);
    digitalWrite(buzzerPin, LOW);
    delay(100);
    digitalWrite(buzzerPin, HIGH);
    delay(500);
    digitalWrite(buzzerPin, LOW);
    delay(100);
    digitalWrite(buzzerPin, HIGH);
    delay(500);
    digitalWrite(buzzerPin, LOW);


    delay(1000);
    lcd.clear();
  } else {
    lcd.clear();
    lcd.print("Incorrect");
    lcd.setCursor(0, 1);
    lcd.print("Password");
    digitalWrite(buzzerPin, HIGH);
    delay(100);
    digitalWrite(buzzerPin, LOW);

    delay(1000);
  }
  return;
} else
  return;
}
```

```
void getChar() {
  btKey = 0;
  if (mySerial.available()) {
    btKey = mySerial.read();
  }
  keyKey = customKeypad.getKey();
  customKey = 0;
  if (btKey || keyKey) {
    customKey = keyKey ? keyKey : btKey;
  }
  if (customKey) {
    digitalWrite(buzzerPin, HIGH);
    delay(100);
    digitalWrite(buzzerPin, LOW);
  }
}
```

Demonstration

https://iiitaphyd-my.sharepoint.com/:v:/g/personal/harry_jain_research_iiit_ac_in/EdGUeO9bXKJGoRh8uWHGxhgBBLm3FiaC0rY1k3uBpZZm1g?e=NOyhcC

Key Performance Indicators

Security

- Lockout mechanism after three incorrect attempts.
- Password masking and real-time feedback to prevent tampering.
- Robustness of wireless communication to prevent interception.

Key Performance Indicators

Hardware Performance

- **Response Time:**
  Quick actuation of the solenoid lock upon correct password entry.

- **Reliability:**
  Consistent operation of the lock mechanism over repeated use.

Key Performance Indicators

Design Efficiency

- **Hardware Complexity:**
  Compact and straightforward design reduces wiring and assembly errors. No reliance on moving parts ensures durability.

- **Power Consumption:**
  Efficient power use with minimal drain during standby.

Key Performance Indicators

User Experience

- **Ease of Operation:**
  Intuitive interface with visual feedback on LCD.
  Clear error messages and lockout notifications.

- **Customisability:**
  Simple password change and wireless control via app.

Key Performance Indicators

Practical Deployment

- **Scalability:**
  Can be easily adapted for multiple doors or enhanced with additional features.

- **Durability:**
  Minimal maintenance due to sturdy components like the solenoid lock.

Cost of the Solution

Arduino Uno R3	420
4x3 Membrane Keypad	45
16x2 LCD with I2C Module	325
Solenoid Lock	310
Relay Module	30
Buzzer	5
Bluetooth Module	230
Misc. (PCB, wires, etc.)	15
	1380

Cost of the Solution

Comparison

- **with the Servo-motor Solution:**
  Could be marginally cheaper but higher dependence on mechanical components, prone to wear and tear, adding maintenance costs.

- **with Commercial Solutions:**
  Cost ranges between INR 5000–10,000 or more depending on features. Often require proprietary apps and lack customizability at the hardware level.

Contributions

- Collaboratively designed and implemented the system architecture, including hardware and software components.
- Jointly developed the keypad, Bluetooth app, and LCD interface for user interaction.
- Worked together on integrating the solenoid lock, relay, and buzzer for seamless operation.
- Shared responsibilities for testing, debugging, and project documentation.

Unimplemented Ideas

Dual-Tone Buzzer Sounds

- Planned to use distinct buzzer tones for each keypress, like an actual keypad.

- Encountered limitations with the Arduino's `tone()` function, leading to unsatisfactory sound quality

Unimplemented Ideas

Camera Integration

- Planned to add a camera for capturing images during incorrect password attempts, enabling remote monitoring.

- Could not implement due to unavailability of a camera module in the lab.

Unimplemented Ideas

Web Server Control

- Considered a web server for Wi-Fi-based remote control and monitoring.
- Implementation was limited by network firewall constraints and restrictions on using laptops during the presentation.

Insights

Technical Skills

- Gained hands-on experience in circuit design, Arduino programming, and Bluetooth integration.

- Learned to interface various components like keypads, LCDs, solenoid locks, and buzzers.

Insights

Problem-Solving

- Overcame challenges in component integration and debugging hardware-software interactions.

- Learned to adapt plans and implement creative solutions under resource constraints.

Insights

System Design Principles

- Understood the importance of modularity for future scalability.
- Balanced security, usability, and cost-efficiency in the design process.

XXX

Insights

Collaboration and Communication:

- Enhanced teamwork and time management while working together on all aspects of the project.

- Developed clear documentation and presentation skills to effectively convey ideas.

References

Arduino Official Documentation

Keypad Library for Arduino

https://www.arduino.cc/reference/en/libraries/keypad/

Arduino Official Documentation

Interfacing HC-05 Bluetooth Module with Arduino

https://www.arduino.cc/en/Guide/ArduinoBT

References

Arduino Official Documentation

I2C Interface for LCDs

https://projecthub.arduino.cc/arduino_uno_guy/i2c-liquid-crystal-displays-5eb615

GEYA

What is a Relay Module and What Does It Do?

https://www.geya.net/what-is-a-relay-module-and-what-does-it-do/

References

MIT App Inventor Documentation

Getting Started with App Inventor

https://appinventor.mit.edu/explore/get-started