

Introduction to Embedded System Project Title: Digital Safe System

# Project Description:

1. **Introduction:**

In today’s world, personal belongings and personal information are more important and hence more important to protect than ever. The Digital Safe System project has been developed to implement an electronic locking system with characteristics very similar to those of a real world safe. This system built with an Arduino Uno has password and RFID based access control, which provides more flexibility and more security. Keypad, an OLED display for instructions and status, LED indicators, and a buzzer for audio feedback, are all used by the safe to follow user interactions. The lock’s movement is physically represented by a servo motor, and it also allows authorized users to access quickly, contactless, through an RFID module.

1. **Project Purpose:**

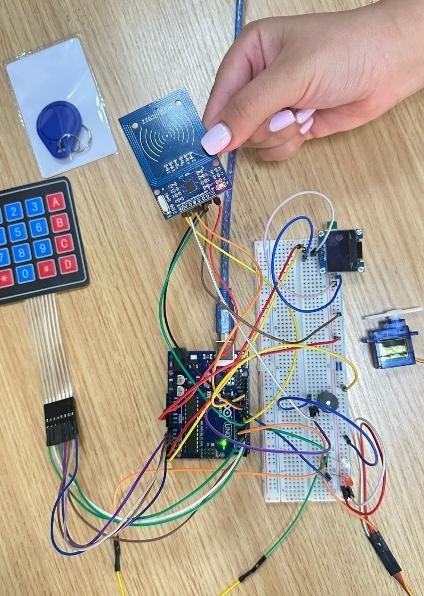
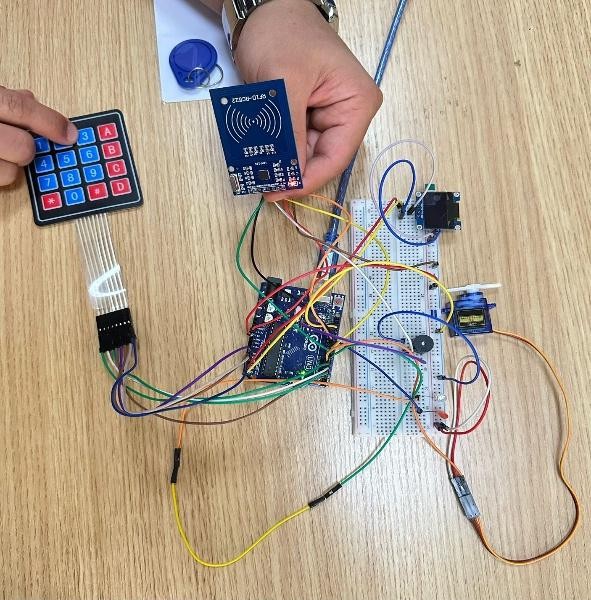
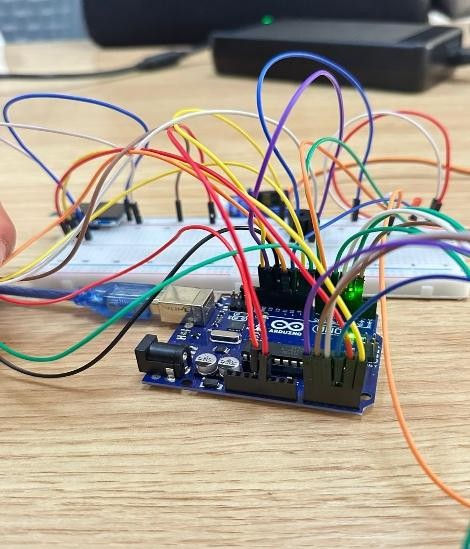
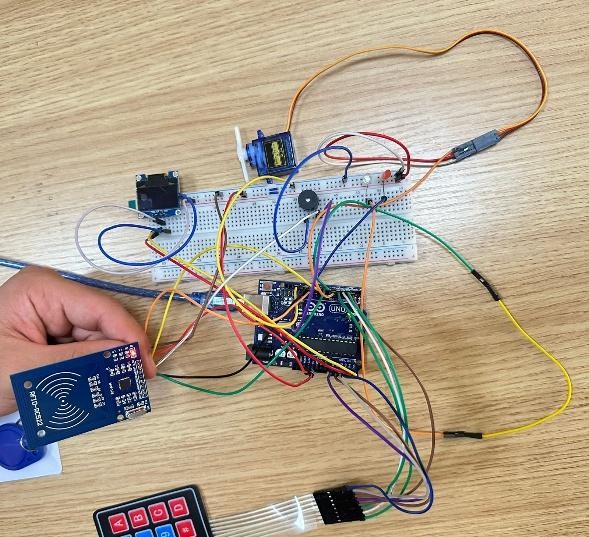
The foremost desired outcome of this project is to design a safe with Arduino based hardware, which has a dual authentication digital locking system. From start, the system is designed in a way by which users can set the custom password on the keypad. Users can unlock the safe by using a correct password, authorized RFID tag or card. Various indicators are provided by the system to signal the safe’s status, such as a simulated servo motor which opens and closes the safe, LED lights for visual cues, an OLED screen which displays text status indicating and updates, and an audible buzzer providing success alert, error, and alert cues. Moreover, the system also offers a master reset option if the user forgets the password. If the safe is opened, then the user must manually re-lock the safe using the special key (such as the # key) to ensure; it’s safe.

# Features

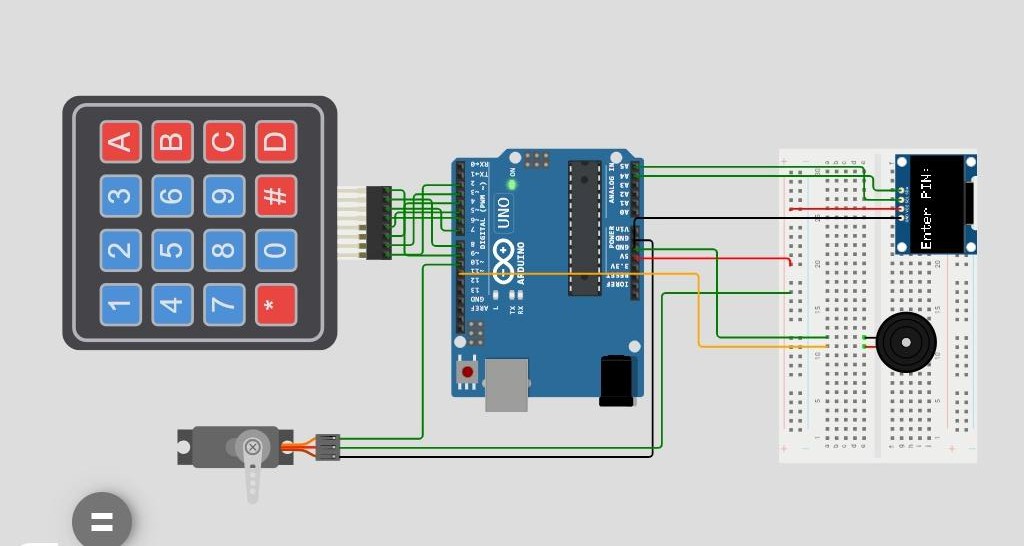
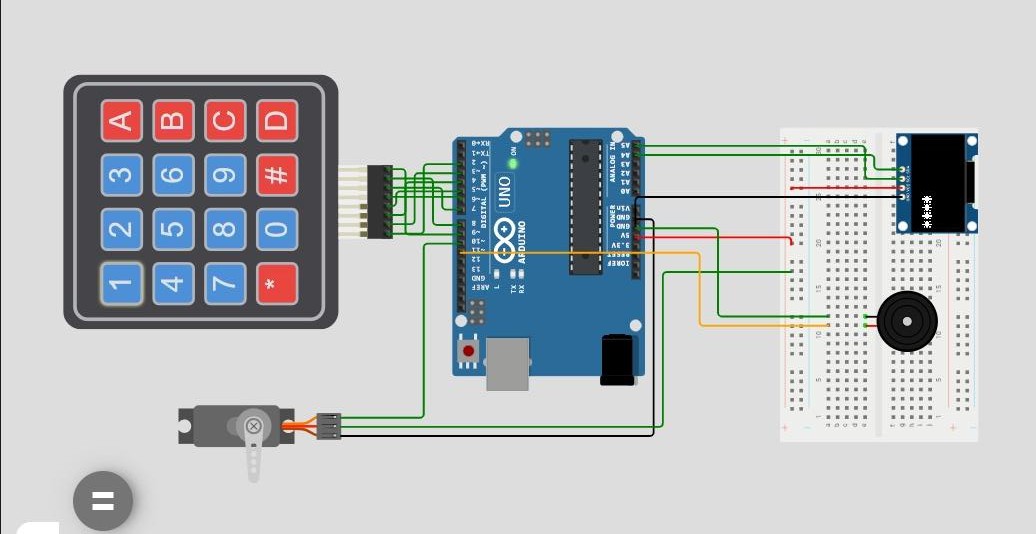
* 1. Password Setup (Keypad + Arduino Uno + OLED Screen)
  2. At the beginning, the user can input and set a custom password using the keypad.
  3. The OLED screen displays real-time prompts or confirmation messages during the setup process.
  4. Unlocking Mechanism (Keypad + RFID + Arduino Uno + Servo Motor)
  5. The safe can be unlocked by the users by either scratching the right password on the keypad or scanning an authorized RFID card.
  6. After authentication, the servo motor acts to move the lock to the open position.
  7. Visual C Audio Indicators (LEDs + Buzzer + Arduino Uno)
  8. The safe’s input transistors are used to wire the buzzers if the safe is successfully opened, a green LED turns on and the buzzer gives a short beep.
  9. A red LED and buzzer sounds an alert if a wrong password or RFID tag is used.
  10. Safe State Indication (Servo Motor + LEDs + OLED Screen)
  11. Servo motor represents the safe’s lock state:
  12. Open: Servo rotates to unlock; LED ON.
  13. Closed: Servo returns to lock position; LED OFF.
  14. The OLED screen shows the current lock status (e.g., “Safe Open”, “Safe Locked”).
  15. Closing the Safe (Keypad + Arduino Uno)
  16. After opening, the user must press the ‘#’ key on the keypad to re-lock the safe.
  17. Master Code Reset (Keypad + OLED + Arduino Uno)
  18. A special master code can be entered to reset the safe in case the user forgets the set password.
  19. The OLED screen confirms when the master code is accepted, and password reset is allowed.

# Schematic Circuit Simulation / Sketch

* **Hardware:**

****

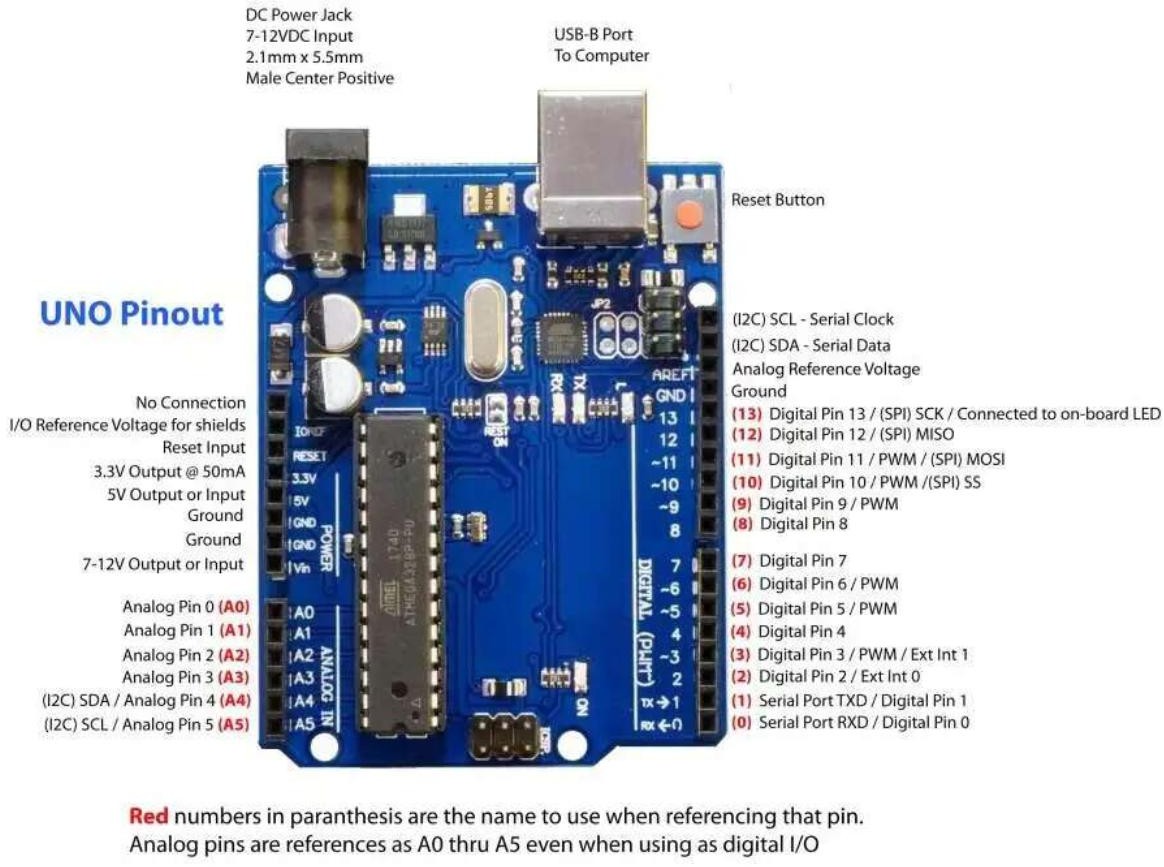
* **Software:**

****

# List of Components

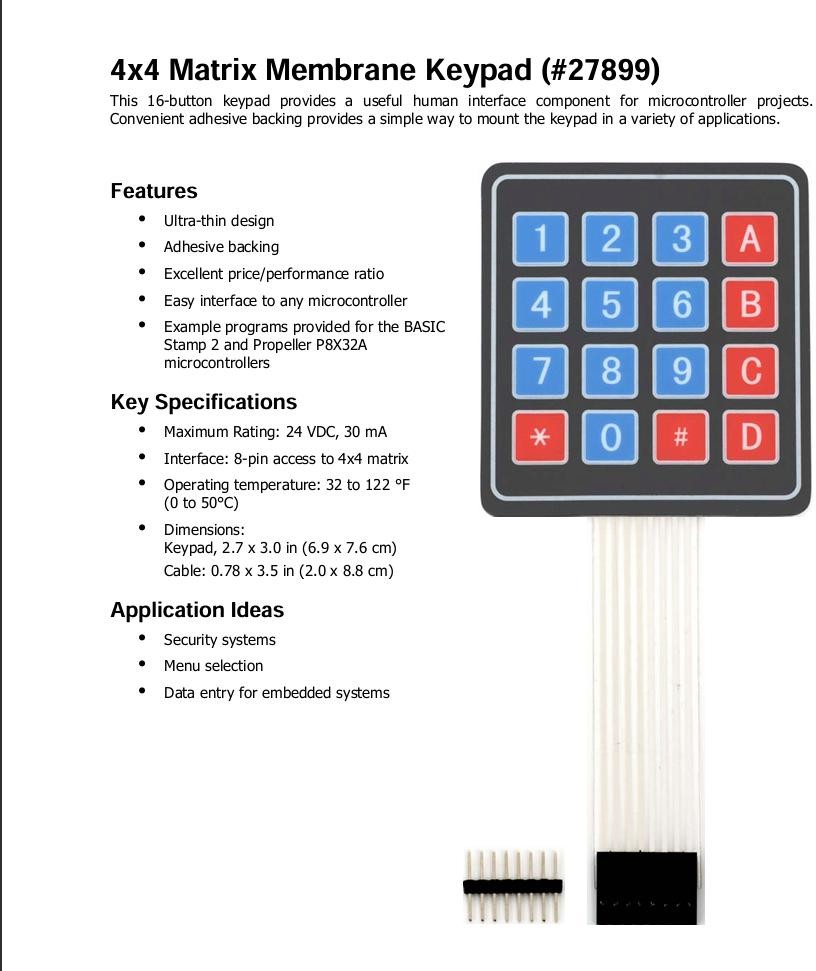
* **Arduino Uno:**

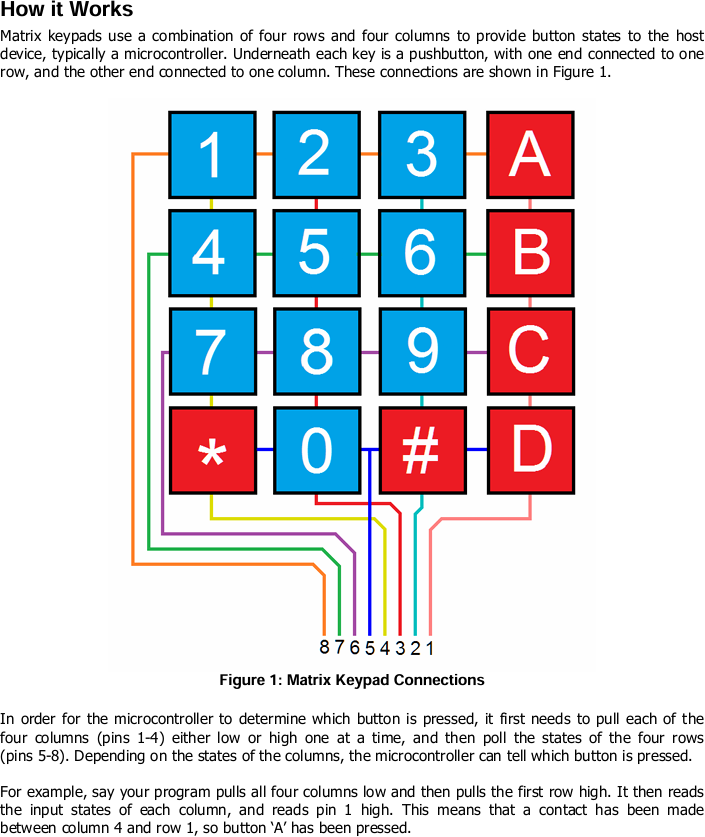
The Arduino Uno R3 is chosen for its Atmega328p microcontroller, it is the core controller of the digital safe system. The keypad and RFID module's input handling as well as servo control to lock and unlock the safe are handled by it and the visual feedback of the OLED screen and LEDs are also managed by it. The buzzer can give audible alerts to the actions of entering a wrong password or when the user successfully enters the password and accesses the device. They include resistors as necessary to keep the current flowing within the circuit as it should be and to protect sensitive devices in the circuit.



* **Keypad Module:**

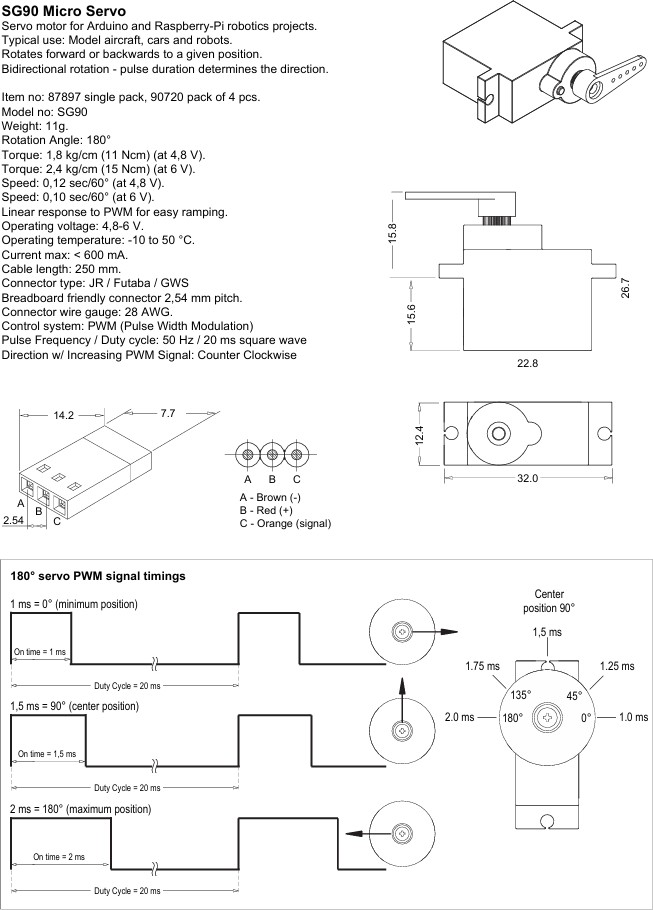
This keypad is used for to input the user such as the password, restoring the system or giving commands. The 16 keys of the 4x4 matrix are read by the Arduino by the columns and the rows and then presented to it for a password entry or a control action, which is possible.





## Servo Motor:

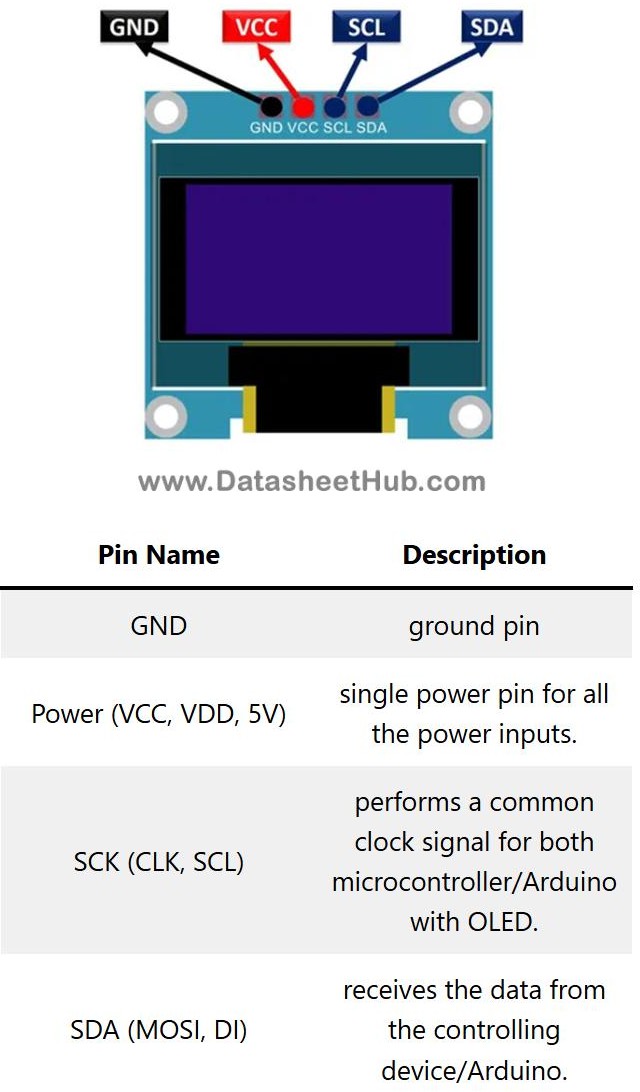
This small servo motor is used to simulate the closing and opening mechanism for the safe. Then it can rotate to one position determined by the Arduino's signal and typically rotates 0° to 180° to represent the locked and unlocked positions.





## OLED Screen:

OLED display uses I2C communication to show system messages such as status updates, prompts, errors and instructions. It enables the users to interact with the system with some sort of visual feedback.



## RFID Reader:

RFID reader is used to scan RFID tags or cards for contactless user authentication. It sends the unique ID of the RFID tag to the Arduino, which then verifies the ID and allows or denies access to the safe.



## RFID Tags/Cards:

These are used in conjunction with the RFID reader to authenticate users. Each tag or card contains a unique ID that is scanned and recognized by the system to allow or deny access.

## Buzzer:

A buzzer is a small speaker capable of producing beeping sounds. It provides sound feedback regarding key actions such as password entry, safe opening or closing, or incorrect attempts with the buzzer, improving user interaction keys, and security.

## LEDs:

LEDs indicate the safe’s status visually. The safe is 'locked' if the red LED is lit, it is unlocked if the green LED is on. They are controlled by the Arduino to indicate the state of the system displayed.

## Resistor:

In the Digital Safe System, LEDs are protected by resistors, which limit current through the LEDs to ensure safe working of the LEDs and Arduino suffers no damage. Moreover, they make the input signals of components such as keypads and RFID modules clean HIGH or LOW signals as well as stabilizing the inputs against floating inputs. In conclusion, resistors are necessary for use safely, for a significant signal reliability and system stability.

# Implementation Steps

The Digital Safe System was executed with the help and support of the following equipment: Arduino Uno for basic functionality, a 4x4 keyboard for digital password input, an RFID module for contactless user certification, a servo motor for the lock mechanism, an OLED screen for user interaction, indicators in the form of LEDs, and a buzzer for notifications. The main components have been put up on the breadboard: OLED (I2C) was used, the servo was connected to a PWM digital pin, the keypad and the RFID module were interfaced via digital pins, and LEDs and buzzer were connected to suitable outputs. The whole system gives the possibility for the user to establish a password for the first time, which will then be saved in the EEPROM. When the opening is selected, the user should either type the approved password into the keypad or place an authorized RFID tag in front of the module. Upon the positive confirmation, the servo swings at an angle to open the safe, the green LED is on, and a message that marks the access of the correct user is also displayed on the OLED screen. On the other hand, locking the safe is realized by pressing the "#" key, thus, the servo returns to the initial/locked place and the red LED is on. If the password or the RFID tag that was used is incorrect, the system immediately responds by beeping and indicating an error. Apart from the password, there is still one more possibility that the reset of the password can be executed. A test was conducted and two of the results confirmed that the system passes both the keypad and the RFID operations, which means that it is easy to use and safe to access.

# Arduino Code

1. #include <Wire.h>
2. #include <Adafruit\_GFX.h>
3. #include <Adafruit\_SSD1306.h>
4. #include <Keypad.h>
5. #include <Servo.h>
6. #define SCREEN\_WIDTH 128
7. #define SCREEN\_HEIGHT 64
8. Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);
9. // Password
10. String password = "1111";
11. String input = "";
12. int maxAttempts = 3;
13. int attemptCount = 0;
14. // Buzzer and Servo
15. const int buzzerPin = 11;
16. const int servoPin = 10;
17. Servo myServo;
18. // Keypad
19. const byte ROWS = 4;
20. const byte COLS = 4;
21. char keys[ROWS][COLS] = {
22. {'1','2','3','A'},
23. {'4','5','6','B'},
24. {'7','8','9','C'},
25. {'\*','0','#','D'}
26. };
27. byte rowPins[ROWS] = {9, 8, 7, 6};
28. byte colPins[COLS] = {5, 4, 3, 2};
29. Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
30. void beep(int duration = 100) {
31. tone(buzzerPin, 350);
32. delay(duration);
33. noTone(buzzerPin);
34. }
35. void alarm() {
36. display.clearDisplay();
37. display.setCursor(0, 10);
38. display.setTextSize(2);
39. display.print("ALARM!");
40. display.display();
41. for (int i = 0; i < 20; i++) {
42. digitalWrite(buzzerPin, HIGH);
43. delay(200);
44. digitalWrite(buzzerPin, LOW);
45. delay(200);
46. }
47. }
48. void setup() {
49. Serial.begin(9600);
50. pinMode(buzzerPin, OUTPUT);
51. digitalWrite(buzzerPin, LOW);
52. myServo.attach(servoPin);
53. myServo.write(0); // Locked
54. if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {
55. Serial.println("OLED failed");
56. while (1);
57. }
58. display.clearDisplay();
59. display.setTextSize(2);
60. display.setTextColor(SSD1306\_WHITE);
61. display.setCursor(0, 10);
62. display.println("Enter PIN:");
63. display.display();
64. }
65. void loop() {
66. char key = keypad.getKey();
67. if (key) {
68. beep();  // click sound
69. if (key == '#') {
70. if (input == password) {
71. display.clearDisplay();
72. display.setCursor(0, 10);
73. display.setTextSize(2);
74. display.println("Access OK");
75. drawOpenLock();
76. delay(1000);
77. display.display();
78. myServo.write(90); // Unlock
79. delay(30000);
80. myServo.write(0); // Lock again
81. input = "";
82. attemptCount = 0;
83. display.clearDisplay();
84. display.setCursor(0, 10);
85. display.println("Enter PIN:");
86. display.display();
87. } else {
88. attemptCount++;
89. display.clearDisplay();
90. display.setCursor(0, 10);
91. display.setTextSize(2);
92. display.println("Wrong PIN");
93. display.display();
94. delay(1500);
95. if (attemptCount >= maxAttempts) {
96. alarm();
97. attemptCount = 0;
98. }
99. input = "";
100. display.clearDisplay();
101. display.setCursor(0, 10);
102. display.println("Enter PIN:");
103. display.display();
104. }
105. } else if (key == '\*') {
106. input = "";
107. display.clearDisplay();
108. display.setCursor(0, 10);
109. display.println("Cleared");
110. display.display();
111. delay(1000);
112. display.clearDisplay();
113. display.setCursor(0, 10);
114. display.println("Enter PIN:");
115. display.display();
116. } else {
117. input += key;
118. display.clearDisplay();
119. display.setCursor(0, 10);
120. display.setTextSize(2);
121. for (int i = 0; i < input.length(); i++) {
122. display.print("\*");
123. }
124. display.display();
125. }
126. }
127. }
128. void drawOpenLock() {
129. display.clearDisplay();
130. // Draw lock body
131. display.fillRect(50, 30, 28, 25, SSD1306\_WHITE);
132. // Draw open shackle
133. display.drawCircle(50, 30, 10, SSD1306\_WHITE);
134. display.drawLine(50, 20, 70, 20, SSD1306\_WHITE);
135. display.drawLine(70, 20, 70, 30, SSD1306\_WHITE);
136. display.display();
137. }