A

Skill Based Mini Project Report

on

“DIGITAL SAFE”

Design and thinking lab (280406)

In fulfilment of the requirement for the award of the degree

SUBMITTED BY

Pratima Jadon (0901AM211039)
Rudraksh Saraf (0901AM211046)
Sakshi Tandon (0901AM211048)
Siddhant Jain (0901AM211058)
Pragya Bhadoriya(0901AM220D03)

Artificial Intelligence and Machine Learning
IV semester

SUBMITTED TO

Dr. Kritika Bansal

Dr. Dhananjay Bisen

Session: Jan-June 2023
Department of Information Technology

Madhav Institute of Technology and Science, Gwalior

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

DECLARATION

I hereby declare that the mini skill-based project for the course Design and Thinking Lab (280406) is being submitted in the partial fulfilment of the requirement for the award of **Bachelor of Technology** in Artificial Intelligence and Machine Learning**.**

All the information in this document has been obtained and presented in accordance with academic rule and ethical conduct.

Date:

Place: Gwalior **Pratima Jadon(0901AM21103**

Rudraksh Saraf (0901AM211046) **Sakshi Tandon(0901AM211048)**

Siddhant Jain(0901AM211058) **Pragya Bhadoriya(0901AM220D03)**

ACKNOWLEDGEMENT

I am deeply indebted to all those who have significantly contributed towards the successful completion of this project. At the outset, I would like to express my sincere gratitude to our project supervisor Dr. Kritika Bansal for her invaluable guidance, encouragement, and support throughout the project. The insightful feedback and constructive criticism has been instrumental in shaping the project's outcome. I would also like to extend my sincere thanks to our institute and department for providing us with the necessary resources and infrastructure required to complete this project. I would like to acknowledge the contributions of our team members and peers who have collaborated with us throughout the project. Their dedication, teamwork, and expertise have been vital in achieving our project goals. Finally, I would like to appreciate the efforts of the open-source community and researchers. Thank you all again for your outstanding support and encouragement throughout the project execution.

Abstract

The project aims to design a digital safe using Arduino, which is an open-source platform for building electronics projects. The safe will be equipped with a keypad and an LCD screen to display messages and input values. The safe will use a combination of numbers entered via the keypad to unlock the door. The Arduino will control a servo motor that will lock and unlock the door, and a buzzer will provide feedback to the user. The code for the safe will be written in the Arduino programming language, and the safe will be powered by a 9V battery. The project will demonstrate the use of microcontrollers in building practical applications and will be a useful tool for securingpersonal belongings.

A digital safe is a secure storage container that is controlled by an Arduino microcontroller board. The safe has a keypad or a fingerprint scanner that allows authorised users to open it while prohibiting unauthorised access.

The project involves programming the Arduino to receive input from the keypad and control the motor, as well as implementing security measures such as password encryption and tamper detection. The safe's components, including the keypad, motor, and Arduino board, are assembled into a sturdy enclosure to prevent unauthorized access.

The digital safe provides several advantages over traditional safes,such as ease of use, customizable security features, and lower cost. Additionally, the Arduino's open-source nature allows for customization and expansion of the project, such as the addition of sensors or remote access capabilities.

Overall, the digital safe using Arduino is a practical and innovative project that demonstrates the potential of microcontrollers and their applications in creating secure and efficient systems.

Table of content

Introduction –

A digital safe is a secure storage container that uses an Arduino microcontroller board to control access to the safe. The safe is equipped with a keypad or a fingerprint scanner to allow authorized users to openthe safe, while preventing unauthorized access.

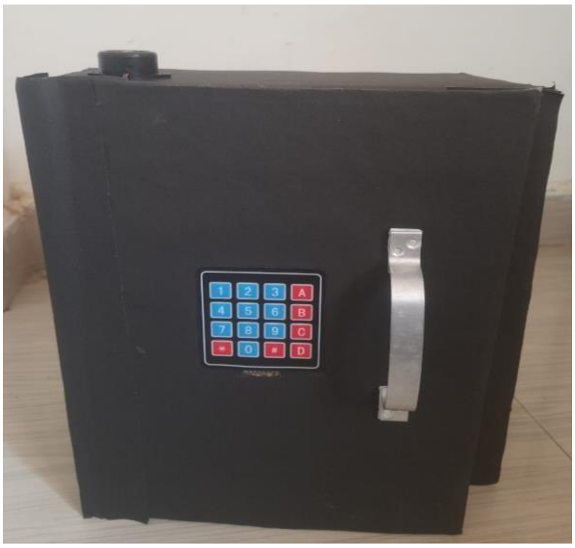


Fig.1 Digital Safe

Here is a general overview of how to create a digital safe using Arduino:

1. Build the Safe: Build the physical structure of the safe using sturdymaterials such as metal or wood. The safe should be large enoughto hold the items you want to protect, but small enough to be easily hidden.
2. Install a Keypad or Fingerprint Scanner: You can use a standard4x4 keypad or a fingerprint scanner that is compatible with Arduino.
3. Wire the Components: Connect the keypad or fingerprint scanner to the Arduino microcontroller board. Also, connect a small motor or solenoid that will act as the lock of the safe.
4. Program the Arduino: Write a program that will allow theArduino to control access to the safe.
5. Test the Safe: Test the safe to make sure that it is working as intended. Try different inputs and combinations to ensure that thesafe only unlocks when authorized.

OBJECTIVE OF DIGITAL SAFE-

The objective of a digital safe is to provide a secure and convenient way to store valuable items, such as cash, jewellery, important documents, and other personal items. Unlike traditional safes that use a physical lock, a digital safe typically uses an electronic locking mechanism that requires a combination code or a key card to access the contents.

Digital safes are designed to provide a high level of security against theft and unauthorized access. They often have features such as a tamper-proof design, an alarm system, and an automatic lockout function that activates when an incorrect combination code is entered multiple times.

In addition to security, digital safes also offer the convenience of quick and easy access to the contents. Users can typically change the combination code or key card as needed, and some digital safes even offer features such as biometric scanning for added security and ease of use.

Overall, the objective of a digital safe is to provide a reliable and secure storage solution for valuable items, while also offering convenience and ease of use for the user.

Micro Project

Component Required In Digital Safe:

1) **Arduino UNO** -

Arduino Uno is a popular microcontroller board based on the ATmega328P microcontroller.It is a simple yet powerful board that can be used for a wide range of projects, from controlling simple LED lights to building advanced robots and complex systems.

Pin description for Arduino Uno:

- Arduino Uno has a total of 20 pins: 14 digital pins and 6 analog pins.
- The digital pins are labelled from 0 to 13 and can be used for both input and output.
- The analog pins are labelled A0 to A5 and can be used for reading analog signals from sensors and other devices.
- Each digital pin can provide up to 40mA of current, and has an internal pull-up resistor that can be activated using code.
- Pins 0 and 1 are used for serial communication, and are also connected to the built-in USB-to-serial converter on the board.
- Pin 2 and 3 are used for interrupts, which allow the board to respond to external events quickly and efficiently.
- Pins 4, 5, 6, 7, 8, 9, 10, and 11 support PWM, which allows analog signals to be generated using digital signals.
- The 5V and 3.3V pins provide regulated voltage to power external devices.
- The GND pins provide a ground connection for external devices.
- The reset pin is used to reset the board and start the bootloader for uploading new code.
- The ICSP header can be used for programming the board using an external programmer.

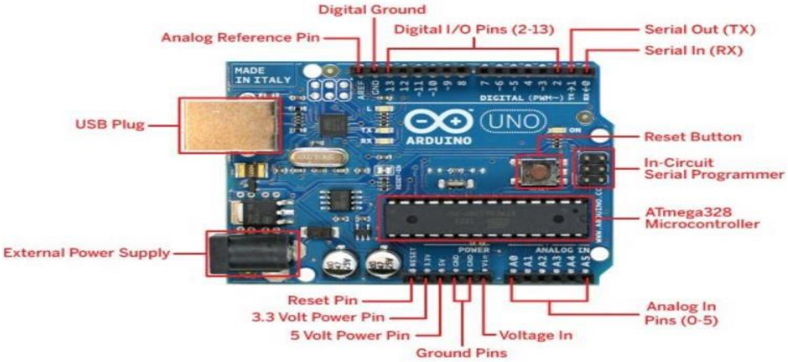


Fig.2 Arduino uno Microcontroller

2)Passive Infrared Sensor(PIR)

PIR (Passive Infrared) Sensor is a type of motion sensor that detects movement of objects by sensing changes in the infrared radiation emitted by them. It works by detecting the movement of heat-emitting objects within its field of view.

PIR sensors have a pyroelectric sensor, which generates a voltage when exposed to heat. The voltage produced by the sensor changes as the amount of heat changes.

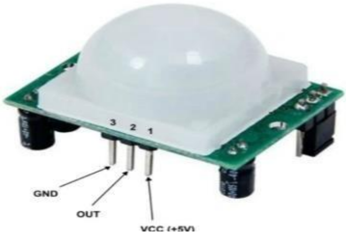
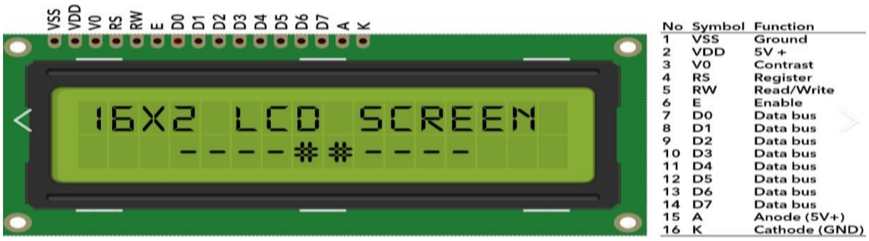


Fig.2 PIR Sensor

3)Liquid Crystal Display(LCD)



No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

Fig.3 LCD(Liquid Crystal Display)

Liquid Crystal Display(LCD) is a type of display that uses liquid crystals to produce images and text.

The liquid crystal display screen works on the principle of blocking light rather than emitting light. LCDs require a backlight as they do not emit light them.

- Pin 1 (VSS): Ground pin, connects to the ground of the microcontroller.
- Pin 2 (VDD): Power pin, connects to the power supply of the microcontroller (+5V for most applications).
- Pin 3 (VO): Contrast pin, controls the contrast of the display. Connected to a potentiometer to adjust the contrast.
- Pin 4 (RS): Register select pin, select between command and data mode. Low for command mode, high for data mode.
- Pin 5 (RW): Read/write pin, selects between read and write mode. Low for write mode, high for read mode (not commonly used).
- Pin 6 (E): Enable pin, enables the module to latch the data present on the data lines (pins 7 to 14).
- Pin 7 to 14 (D0 to D7): Data pins, used to send data and commands to the LCD module.
- Pin 15 (A): Anode pin of the LED backlight. Connected to a current-limiting resistor and then to a positive supply voltage.

- Pin 16 (K): Cathode pin of the LED backlight. Connected to ground.

4) BUZZER

A buzzer is an electronic component that produces sound when it is activated. It is commonly used as an audible indicator or alarm in electronic circuits and devices. Here are some key features and specifications of buzzers:

- Buzzers can be either passive or active. Passive buzzers require an external oscillator or AC signal to generate a sound, while active buzzers have a built-in oscillator and can generate sound by applying DC voltage.
- Buzzers are available in various sizes and shapes, ranging from small surface mount devices to large horn-type buzzers.

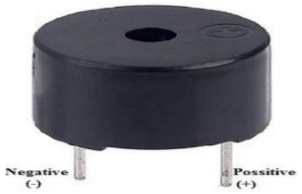


Fig.4 Buzzer

5)KEYPAD

A keypad is a user input device that allows users to enter data, such as numbers, letters, or symbols, by pressing keys arranged in a matrix pattern. It is commonly used in electronic circuits and devices that require user input, such as calculators, security systems, and digital locks.

Buzzers can be used in various applications, such as alarm systems, timers, doorbells, toys, and musical instruments.

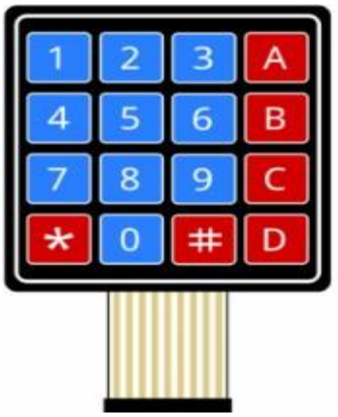


Fig.5 Keypad

6.Servo Motor

In a digital safe using an Arduino, a servo motor is often used to lock and unlock the safe. A servo motor is a type of motor that can be controlled to move to a specific angle. In this case, the servo motor is used to move a locking mechanism to either lock or unlock the safe.The servo motor is connected to a specific pin on the Arduino board and can be controlled using the Servo library. The position of the servo motor can be set using the write () function, which takes an angle between 0 and 180 degrees.

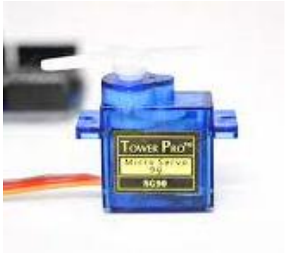


Fig.6 Servo motor

To lock the safe, the servo motor is typically moved to a position of 180 degrees, which is defined as SERVO_LOCK_POS in the code. To unlock the safe, the servo motor is moved to a position of 90 degrees, which is defined as SERVO_UNLOCK_POS in the code.

Overall, the servo motor is a crucial component of the digital safe, as it provides a reliable and secure way to lock and unlock the safe using an electronic mechanism.

6) Breadboard

A breadboard is a board with a grid of holes where electronic components can be placed and connected together without the need for soldering. The breadboard typically has a set of power rails running along the top and bottom, which can be used to provide power to the various components.

In the case of the digital safe, the breadboard is used to connect the keypad, LCD display, servo motor, PIR motion sensor, and buzzer to the Arduino board. The rows and columns of the breadboard are typically used to create circuits that connect the various components together.

For example, the pins of the keypad are connected to specific rows and columns on the breadboard, and these connections are then connected to specific pins on the Arduino board. Similarly, the pins of the servo motor, PIR motion sensor, and buzzer are connected to the breadboard and then connected to specific pins on the Arduino board. Overall, the breadboard provides a flexible and convenient way to connect the various components of the digital safe together, making it easier to assemble and modify the safe as needed.

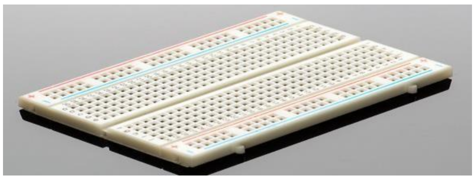


Fig.7 Breadboard

Macro Project:

LCD and Keypad implementation in Digital Safe using Arduino

Here are the steps for connecting an Arduino board with a keypad and outputting the results on the serial monitor for a digital safe:

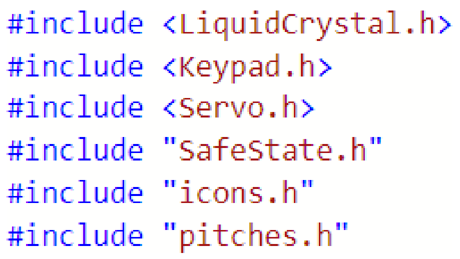
1. Gather the necessary materials:

- Arduino board (e.g. Uno)
- Keypad module
- Jumper wires
- USB cable for Arduino board
- Computer with Arduino IDE software installed

2. Connect the keypad to the Arduino board:

- Connect the keypad's VCC pin to the Arduino's 5V pin
- Connect the keypad's GND pin to the Arduino's GND pin
- Connect the keypad's OUT pin to any digital pin on the Arduino board (e.g. pin 2)

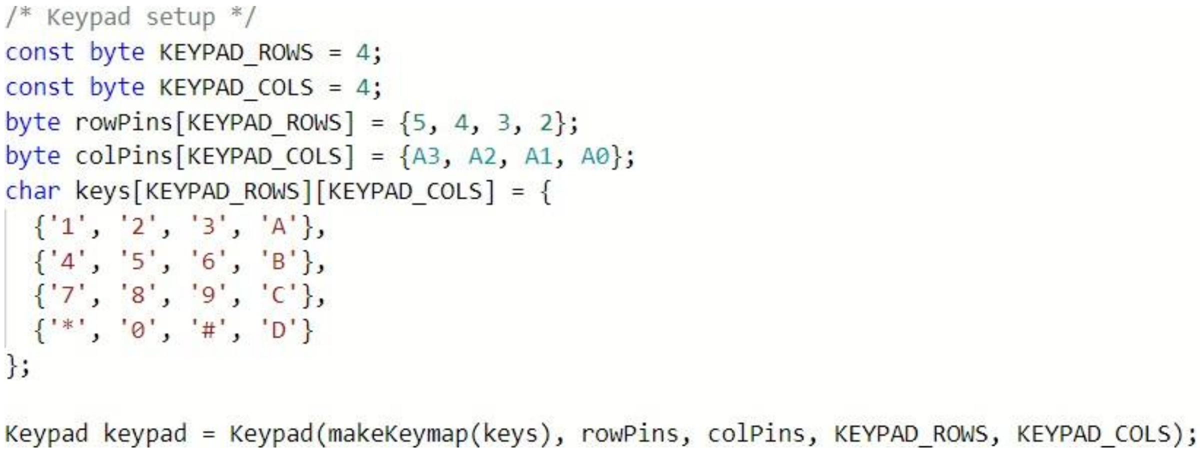
3. Open the Arduino IDE software on your computer and create a new sketch.
4. In the Arduino IDE, add the Keypad library by going to Sketch > Include Library > Manage Libraries, then search for "Keypad" and install the library.



```
#include <LiquidCrystal.h>
#include <Keypad.h>
#include <Servo.h>
#include "SafeState.h"
#include "icons.h"
#include "pitches.h"
```

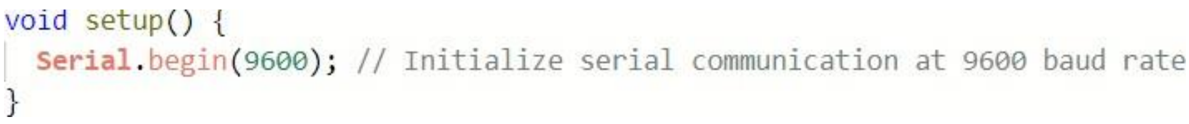
5. Define the keypad layout and pins in the sketch

This code defines a 4x4 keypad layout with four rows and four columns, and assigns the keypad pins to specific Arduino pins. You can modify the layout and pins to match your specific keypad.



```
/* Keypad setup */
const byte KEYPAD_ROWS = 4;
const byte KEYPAD_COLS = 4;
byte rowPins[KEYPAD_ROWS] = {5, 4, 3, 2};
byte colPins[KEYPAD_COLS] = {A3, A2, A1, A0};
char keys[KEYPAD_ROWS][KEYPAD_COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, KEYPAD_ROWS, KEYPAD_COLS);
```

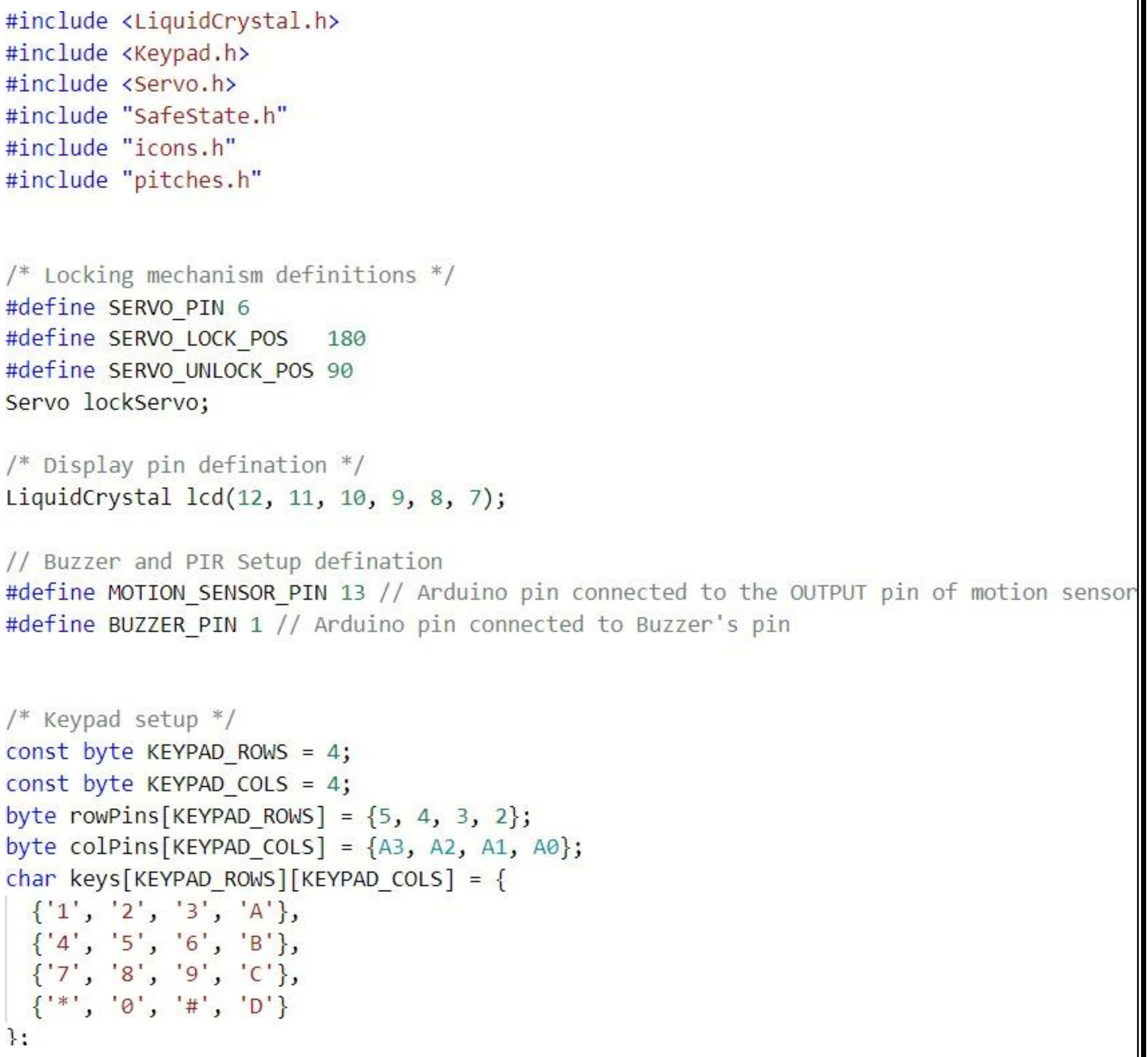
6. In the **setup ()** function, initialize the serial monitor:



```
void setup() {
  Serial.begin(9600); // Initialize serial communication at 9600 baud rate
}
```

7. In the **loop ()** function, use the function from the Keypad library to read the keypad input and output it to the serial monitor.
8. Upload the sketch to the Arduino board by clicking the "Upload" button in the Arduino IDE.
9. Open the serial monitor by clicking the "Serial Monitor" button in the Arduino IDE or by pressing Ctrl+Shift+M. The serial monitor should display any keypad input as soon as a key is pressed.

Program-



```
#include <LiquidCrystal.h>
#include <Keypad.h>
#include <Servo.h>
#include "SafeState.h"
#include "icons.h"
#include "pitches.h"


/* Locking mechanism definitions */
#define SERVO_PIN 6
#define SERVO_LOCK_POS   180
#define SERVO_UNLOCK_POS 90
Servo lockServo;

/* Display pin defination */
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);

// Buzzer and PIR Setup defination
#define MOTION_SENSOR_PIN 13 // Arduino pin connected to the OUTPUT pin of motion sensor
#define BUZZER_PIN 1 // Arduino pin connected to Buzzer's pin


/* Keypad setup */
const byte KEYPAD_ROWS = 4;
const byte KEYPAD_COLS = 4;
byte rowPins[KEYPAD_ROWS] = {5, 4, 3, 2};
byte colPins[KEYPAD_COLS] = {A3, A2, A1, A0};
char keys[KEYPAD_ROWS][KEYPAD_COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
```

This code uses the LiquidCrystal library to control an LCD screen and the Keypad library to read input from a 4x4 keypad. This code defines the pin connections and layout for a digital safe using an Arduino.

It also sets up a servo motor connected to pin 6 for locking and unlocking the safe. The servo can be moved to two positions: SERVO_LOCK_POS (180) to lock the safe and SERVO_UNLOCK_POS (90) to unlock it.

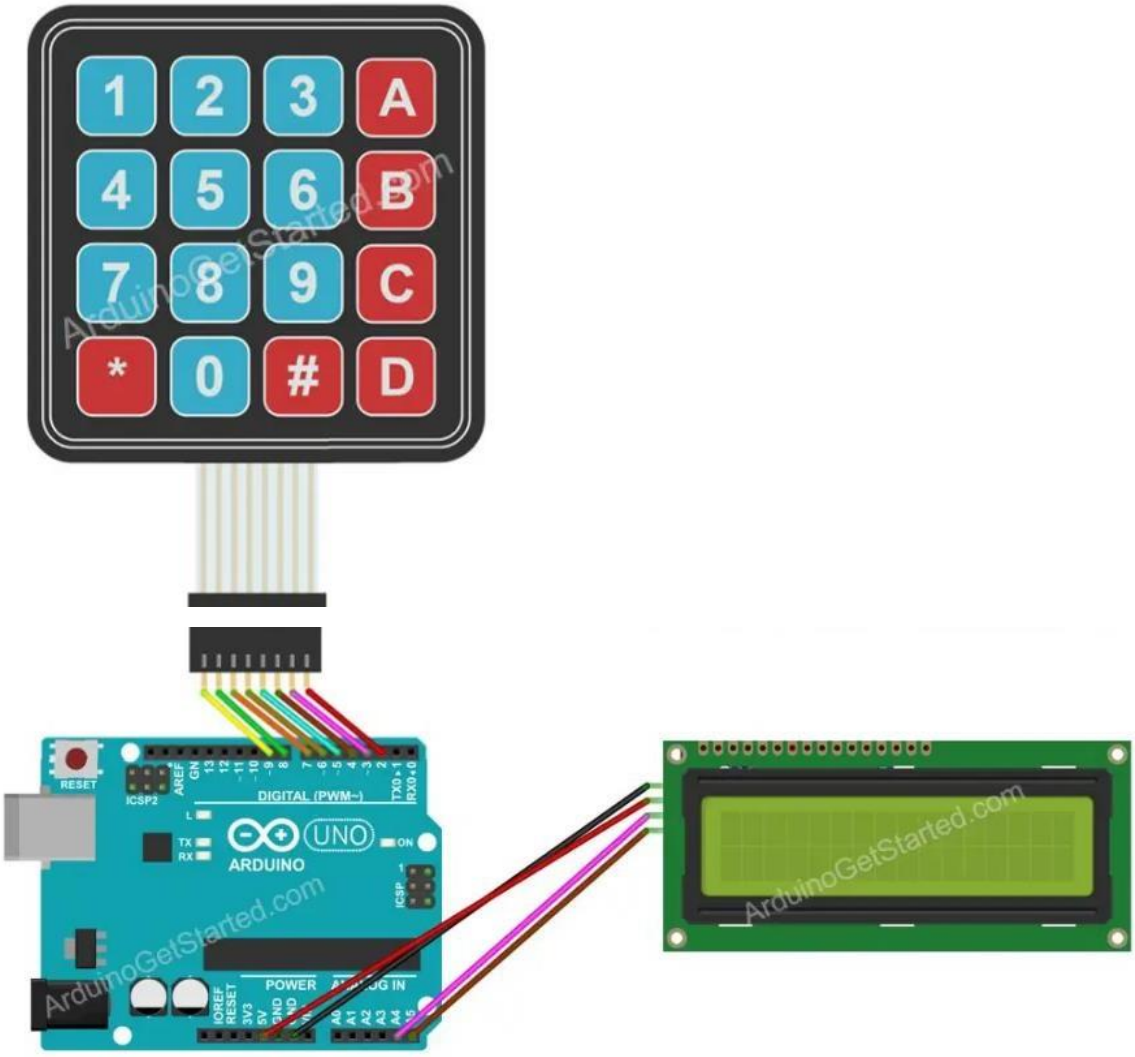


Fig.8 Implementation of LCD and Keypad

Additionally, the code defines the pins for a PIR motion sensor and a buzzer that can be used to trigger an alarm when the safe is tampered with.

The KEYPAD_ROWS and KEYPAD_COLS variables define the number of rows and columns on the keypad, respectively. The rowPins and colPins arrays specify the pins used for the keypad rows and columns, respectively. The keys array defines the layout of the keypad, with each character representing a key on the keypad. For example, '1' represents the top-left key on the keypad, and 'A' represents the top-right key.

Mini Project

Here are the steps for connecting LCD, buzzer, keypad, PIR sensor, servo motor, and Arduino for a digital safe:

1. LCD: Connect the LCD display to the Arduino board by connecting the VSS pin to GND, VDD to +5V, and V0 to a potentiometer. Connect the RS pin to digital pin 12, RW to GND, and E to digital pin 11. Connect the data pins D4-D7 to digital pins 5-8 respectively.
2. Buzzer: Connect the positive leg of the buzzer to digital pin 9, and the negative leg to GND.
3. Keypad: Connect the keypad to the Arduino board by connecting the rows to digital pins 2-5, and the columns to digital pins 6-9.
4. PIR sensor: Connect the PIR sensor to the Arduino board by connecting the VCC pin to +5V, GND to GND, and the signal pin to digital pin 10.
5. Servo motor: Connect the servo motor to the Arduino board by connecting the red wire to +5V, the black wire to GND, and the signal wire to digital pin 3.
6. Power: Connect the power supply to the Arduino board, ensuring that the voltage and current are appropriate for the components being used.

Once all the components are connected, you can program the Arduino to control the digital safe. The programming code will depend on the specific functionality and security features of the safe, such as the required combination for unlocking, the actions to take upon detecting motion with the PIR sensor, and the movements of the servo motor to lock and unlock the safe. The LCD and buzzer can be used to provide feedback to the user about the status of the safe and any errors that occur during operation.

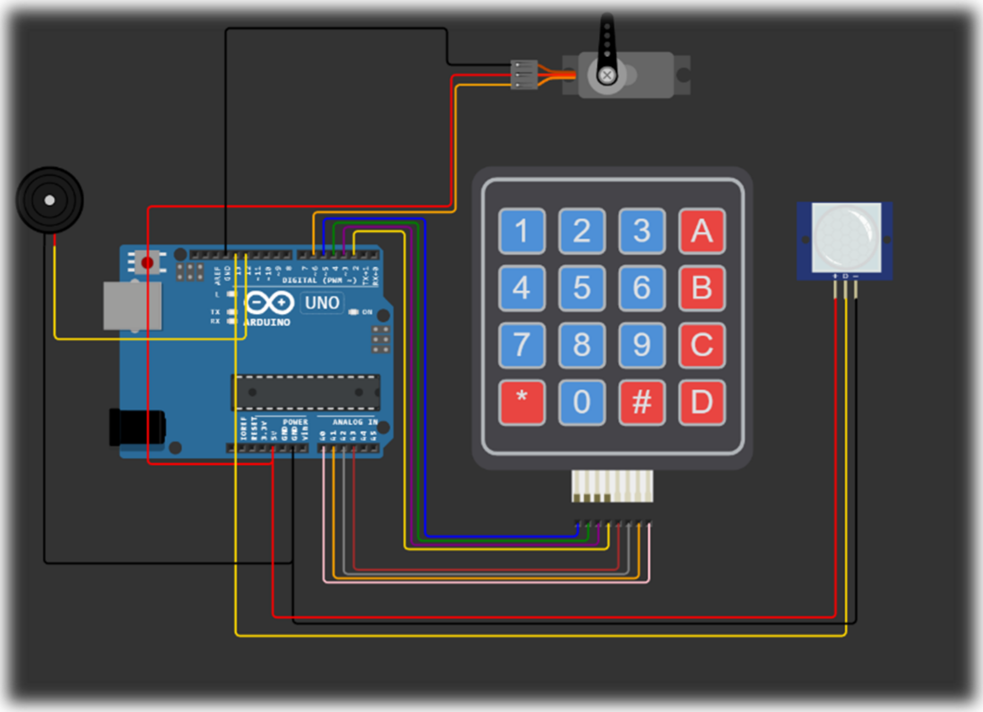
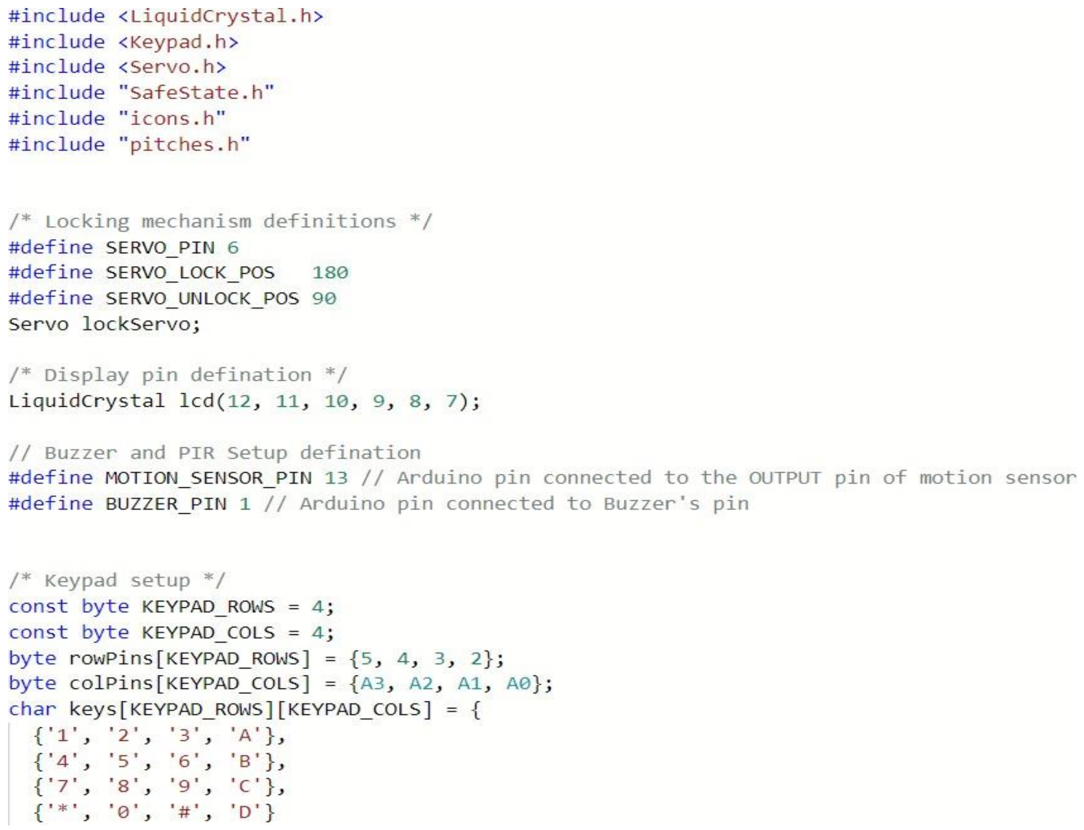
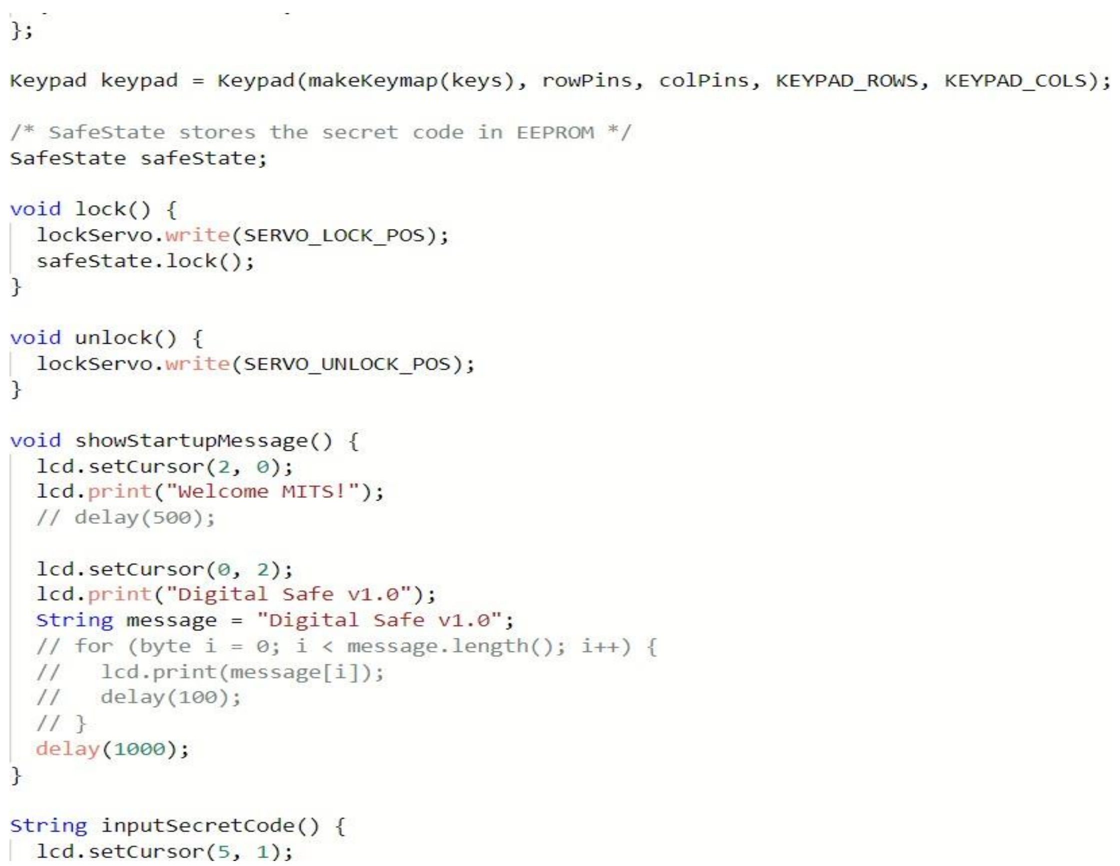


Fig.9 Digital Safe

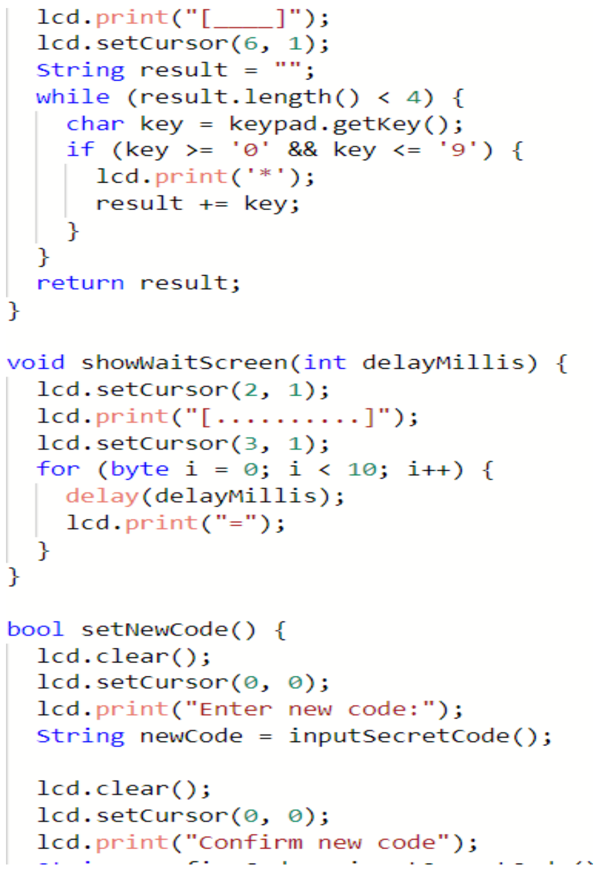
Program-



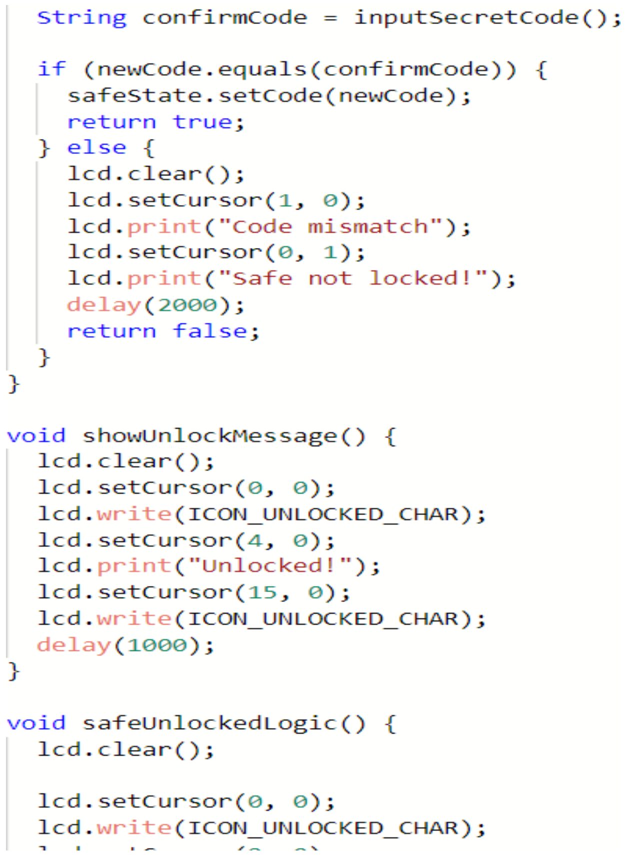
```
#include <LiquidCrystal.h>
#include <Keypad.h>
#include <Servo.h>
#include "SafeState.h"
#include "icons.h"
#include "pitches.h"


/* Locking mechanism definitions */
#define SERVO_PIN 6
#define SERVO_LOCK_POS   180
#define SERVO_UNLOCK_POS 90
Servo lockServo;

/* Display pin defination */
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);

// Buzzer and PIR Setup defination
#define MOTION_SENSOR_PIN 13 // Arduino pin connected to the OUTPUT pin of motion sensor
#define BUZZER_PIN 1 // Arduino pin connected to Buzzer's pin


/* Keypad setup */
const byte KEYPAD_ROWS = 4;
const byte KEYPAD_COLS = 4;
byte rowPins[KEYPAD_ROWS] = {5, 4, 3, 2};
byte colPins[KEYPAD_COLS] = {A3, A2, A1, A0};
char keys[KEYPAD_ROWS][KEYPAD_COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
```



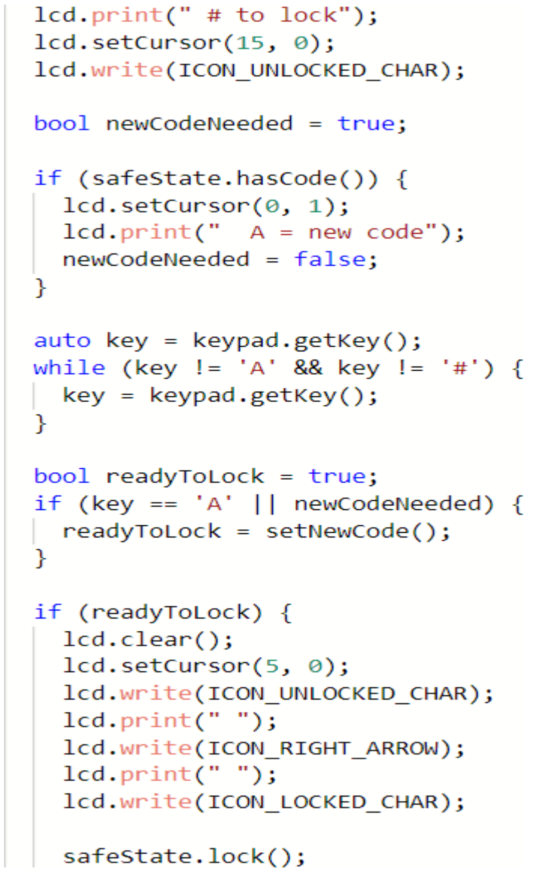
```
};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, KEYPAD_ROWS, KEYPAD_COLS);

/* SafeState stores the secret code in EEPROM */
SafeState safeState;

void lock() {
  lockServo.write(SERVO_LOCK_POS);
  safeState.lock();
}

void unlock() {
  lockServo.write(SERVO_UNLOCK_POS);
}

void showStartupMessage() {
  lcd.setCursor(2, 0);
  lcd.print("Welcome MITS!");
  // delay(500);

  lcd.setCursor(0, 2);
  lcd.print("Digital Safe v1.0");
  String message = "Digital Safe v1.0";
  // for (byte i = 0; i < message.length(); i++) {
  //   lcd.print(message[i]);
  //   delay(100);
  // }
  delay(1000);
}

String inputSecretCode() {
  lcd.setCursor(5, 1);
```



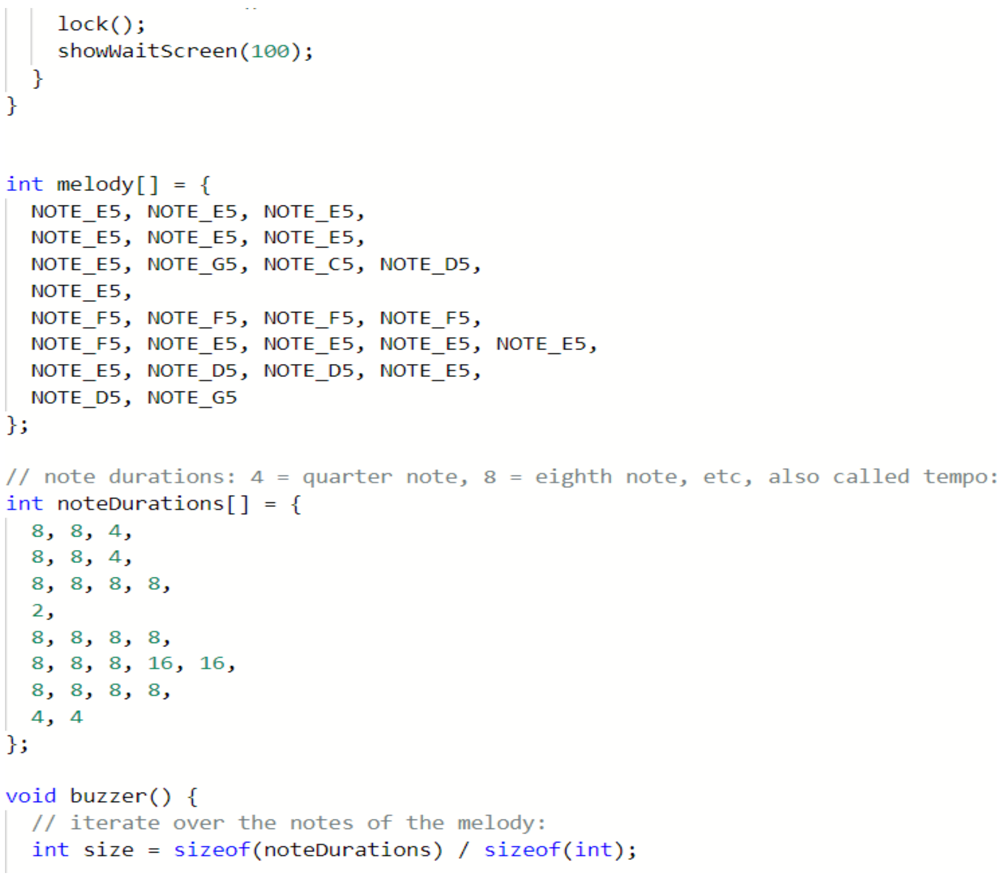
```
  lcd.print("[____]");
  lcd.setCursor(6, 1);
  String result = "";
  while (result.length() < 4) {
    char key = keypad.getKey();
    if (key >= '0' && key <= '9') {
      lcd.print('*');
      result += key;
    }
  }
  return result;
}

void showWaitScreen(int delayMillis) {
  lcd.setCursor(2, 1);
  lcd.print("[..........]");
  lcd.setCursor(3, 1);
  for (byte i = 0; i < 10; i++) {
    delay(delayMillis);
    lcd.print("=");
  }
}

bool setNewCode() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Enter new code:");
  String newCode = inputSecretCode();

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Confirm new code");
```



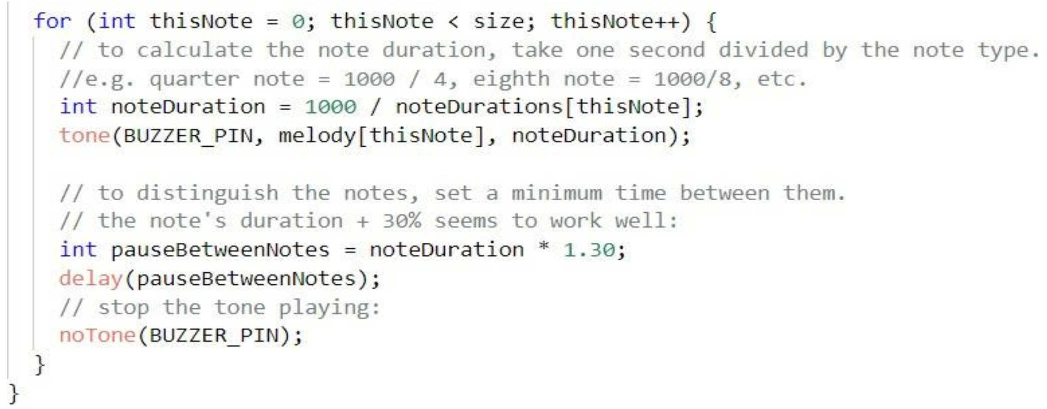
```
  String confirmCode = inputSecretCode();

  if (newCode.equals(confirmCode)) {
    safeState.setCode(newCode);
    return true;
  } else {
    lcd.clear();
    lcd.setCursor(1, 0);
    lcd.print("Code mismatch");
    lcd.setCursor(0, 1);
    lcd.print("Safe not locked!");
    delay(2000);
    return false;
  }
}

void showUnlockMessage() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.write(ICON_UNLOCKED_CHAR);
  lcd.setCursor(4, 0);
  lcd.print("Unlocked!");
  lcd.setCursor(15, 0);
  lcd.write(ICON_UNLOCKED_CHAR);
  delay(1000);
}

void safeUnlockedLogic() {
  lcd.clear();

  lcd.setCursor(0, 0);
  lcd.write(ICON_UNLOCKED_CHAR);
  lcd.setCursor(2, 0);
```



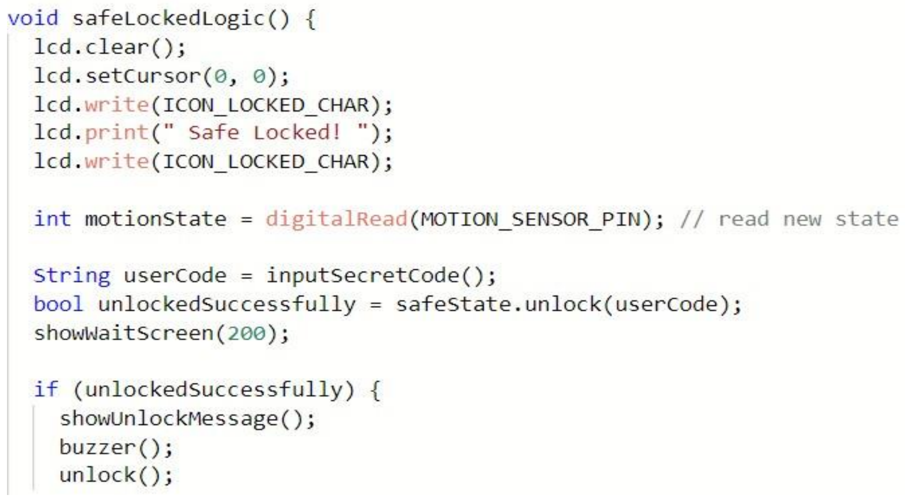
```
  lcd.print(" # to lock");
  lcd.setCursor(15, 0);
  lcd.write(ICON_UNLOCKED_CHAR);

  bool newCodeNeeded = true;

  if (safeState.hasCode()) {
    lcd.setCursor(0, 1);
    lcd.print("  A = new code");
    newCodeNeeded = false;
  }

  auto key = keypad.getKey();
  while (key != 'A' && key != '#') {
    key = keypad.getKey();
  }

  bool readyToLock = true;
  if (key == 'A' || newCodeNeeded) {
    readyToLock = setNewCode();
  }

  if (readyToLock) {
    lcd.clear();
    lcd.setCursor(5, 0);
    lcd.write(ICON_UNLOCKED_CHAR);
    lcd.print(" ");
    lcd.write(ICON_RIGHT_ARROW);
    lcd.print(" ");
    lcd.write(ICON_LOCKED_CHAR);

    safeState.lock();
```



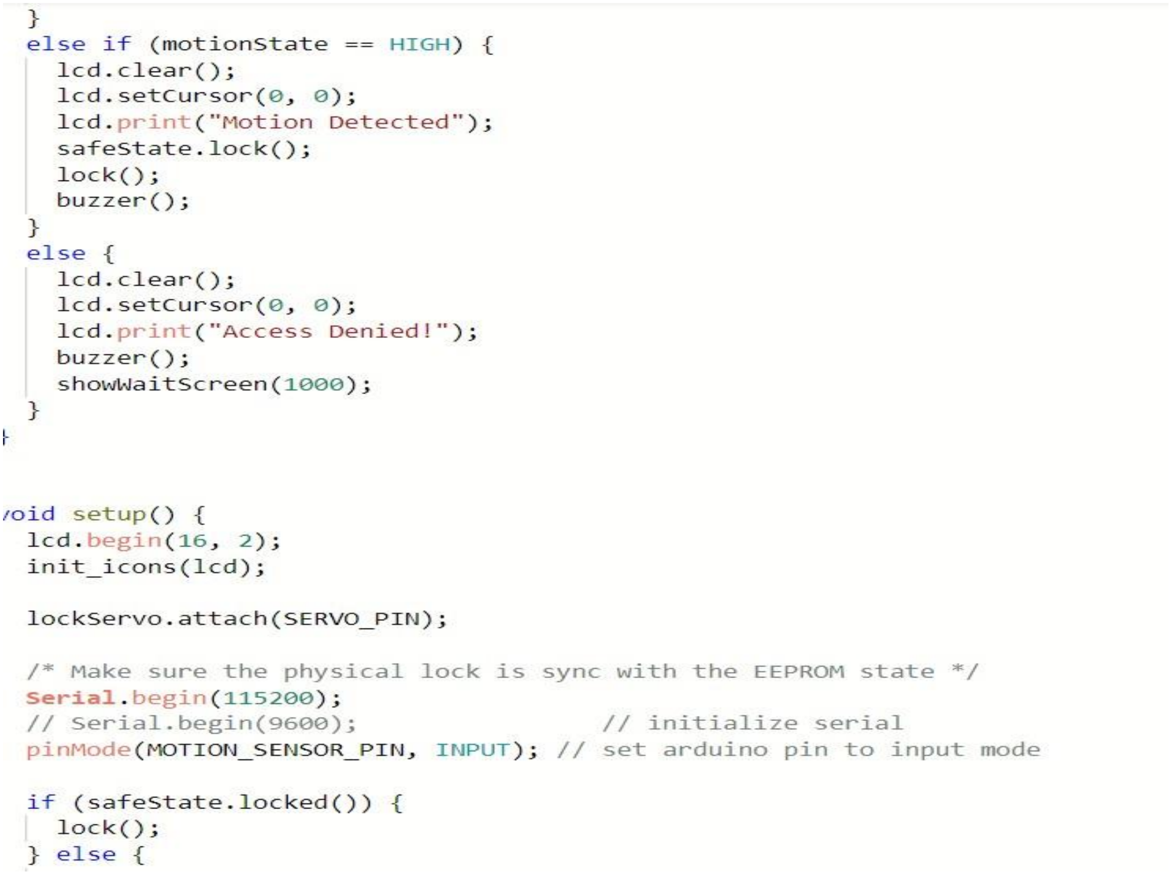
```
    lock();
    showWaitScreen(100);
  }
}


int melody[] = {
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_G5, NOTE_C5, NOTE_D5,
  NOTE_E5,
  NOTE_F5, NOTE_F5, NOTE_F5, NOTE_F5,
  NOTE_F5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_D5, NOTE_D5, NOTE_E5,
  NOTE_D5, NOTE_G5
};

// note durations: 4 = quarter note, 8 = eighth note, etc, also called tempo:
int noteDurations[] = {
  8, 8, 4,
  8, 8, 4,
  8, 8, 8, 8,
  2,
  8, 8, 8, 8,
  8, 8, 8, 16, 16,
  8, 8, 8, 8,
  4, 4
};

void buzzer() {
  // iterate over the notes of the melody:
  int size = sizeof(noteDurations) / sizeof(int);
```



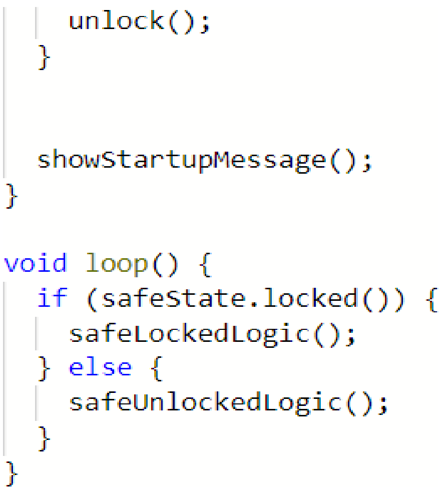
```
  for (int thisNote = 0; thisNote < size; thisNote++) {
    // to calculate the note duration, take one second divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(BUZZER_PIN, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(BUZZER_PIN);
  }
}
```



```
void safeLockedLogic() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.write(ICON_LOCKED_CHAR);
  lcd.print(" Safe Locked! ");
  lcd.write(ICON_LOCKED_CHAR);

  int motionState = digitalRead(MOTION_SENSOR_PIN); // read new state

  String userCode = inputSecretCode();
  bool unlockedSuccessfully = safeState.unlock(userCode);
  showWaitScreen(200);

  if (unlockedSuccessfully) {
    showUnlockMessage();
    buzzer();
    unlock();
```



```
  }
  else if (motionState == HIGH) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Motion Detected");
    safeState.lock();
    lock();
    buzzer();
  }
  else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Access Denied!");
    buzzer();
    showWaitScreen(1000);
  }
}


/oid setup() {
  lcd.begin(16, 2);
  init_icons(lcd);

  lockServo.attach(SERVO_PIN);

  /* Make sure the physical lock is sync with the EEPROM state */
  Serial.begin(115200);
  // Serial.begin(9600);                    // initialize serial
  pinMode(MOTION_SENSOR_PIN, INPUT); // set arduino pin to input mode

  if (safeState.locked()) {
    lock();
  } else {
```



```
    unlock();
  }


  showStartupMessage();
}

void loop() {
  if (safeState.locked()) {
    safeLockedLogic();
  } else {
    safeUnlockedLogic();
  }
}
```

Application of Digital Safe-

Digital safes, also known as electronic safes, have become increasingly popular for personal and commercial use due to their convenience and security features. Here are some common applications of digital safes:

1. Home security: Digital safes are often used to secure valuable items such as jewellery, cash, and important documents in the home. They can be easily installed in a closet, on a shelf, or even bolted to the floor for added security.
2. Business security: Many businesses use digital safes to secure cash, receipts, and other sensitive documents. These safes can be programmed to require multiple access codes or to trigger an alarm if accessed without authorization.
3. Hotel rooms: Digital safes are commonly found in hotel rooms as a secure place for guests to store their valuables. They are usually programmed with a unique access code for each guest and can be reset after each use.
4. Gun safes: Digital safes are often used to secure firearms in the home or at shooting ranges. They can be programmed to require a fingerprint or other biometric data for added security.
5. Personal lockers: Digital safes are also commonly used in personal lockers at gyms, schools, and other public places. They provide a secure place for individuals to store their personal belongings while they are away.

Overall, digital safes offer a convenient and secure way to protect valuable items and sensitive information from theft, fire, and other types of damage.

Conclusion

A successful digital safe project should be able to provide a secure and convenient solution for storing valuable items or personal belongings

In conclusion, the digital safe project using Arduino Uno is an effective and reliable solution for securing valuable items. With the use of an electronic lock mechanism, the safe can only be opened with the correct password input. The Arduino Uno microcontroller serves as the brains of the operation, providing the necessary logic to control the lock and user interface.

The project involves several important components, including the keypad, LCD screen, and servo motor. By combining these components with the Arduino Uno board, it's possible to create a robust and user-friendly safe system.

Overall, the project is a great way to explore the capabilities of Arduino Uno and gain practical experience in electronics and programming. It can also be easily customized and extended to suit specific requirements, making it a versatile solution for a range of security applications.

A successful digital safe project should be able to provide a secure and convenient solution for storing valuable items or personal belongings.

Future Scope

Digital safes using Arduino Uno have a bright future due to the increasing demand for secure storage of valuable items and documents. With the advancement in technology, people are looking for innovative solutions that can provide enhanced security features.

Here are some potential future applications of digital safes using Arduino Uno:

1. **Home and Office Security**: Digital safes can be used for home and office security, where individuals can keep their valuables and important documents safe. With the integration of Arduino Uno, users can add more security features such as biometric authentication, remote access control, and real-time alerts.
2. **Smart Lockers**: Arduino Uno-based digital safes can be used as smart lockers in various applications such as airports, railway stations, gyms, and shopping malls. Smart lockers can provide secure storage for personal belongings while eliminating the need for traditional lock and key mechanisms.
3. **Banking and Financial Institutions**: Digital safes can be used in the banking and financial industry to provide secure storage for cash, documents, and other valuables. With the integration of Arduino Uno, banks can add more security features such as facial recognition, fingerprint authentication, and two-factor authentication.
4. **Healthcare Industry**: In the healthcare industry, digital safes can be used to store sensitive patient data and medical records. With the integration of Arduino Uno, hospitals can add more security features such as real-time monitoring and remote access control.
5. **Military and Defense**: Digital safes using Arduino Uno can be used in the military and defense industry to store sensitive data and confidential information. With the integration of advanced encryption and authentication technologies, digital safes can provide robust security against cyber attacks and data breaches.

Bibliography

1. Arduino. (2021). Arduino Uno. Retrieved from https://www.arduino.cc/en/Guide/ArduinoUno
2. Patel, S. (2020). How to Make a Digital Safe with Arduino. Arduino Project Hub. Retrieved from https://create.arduino.cc/projecthub/shubham-patel/how-to-make-a-digital-safe-with-arduino-2b10d6
3. Singh, A. (2019). Development of a Digital Safe Using Arduino Microcontroller. International Journal of Innovative Technology and Exploring Engineering, 8(11), 678-682. doi: 10.35940/ijitee.L1137.1181119
4. TutorialsPoint. (2021). Arduino - Password Protected Digital Safe. Retrieved from https://www.tutorialspoint.com/arduino/arduino_password_protected_digital_safe.htm
5. Udemy. (2021). Arduino Programming and Hardware Fundamentals with Hackster. Retrieved from https://www.udemy.com/course/arduino-programming-and-hardware-fundamentals-with-hackster/
6. Arduino Forum. (2021). Digital Safe with Servo and Keypad. Retrieved from https://forum.arduino.cc/t/digital-safe-with-servo-and-keypad/546604