Raunak Kumar Singh
2020BCS0124

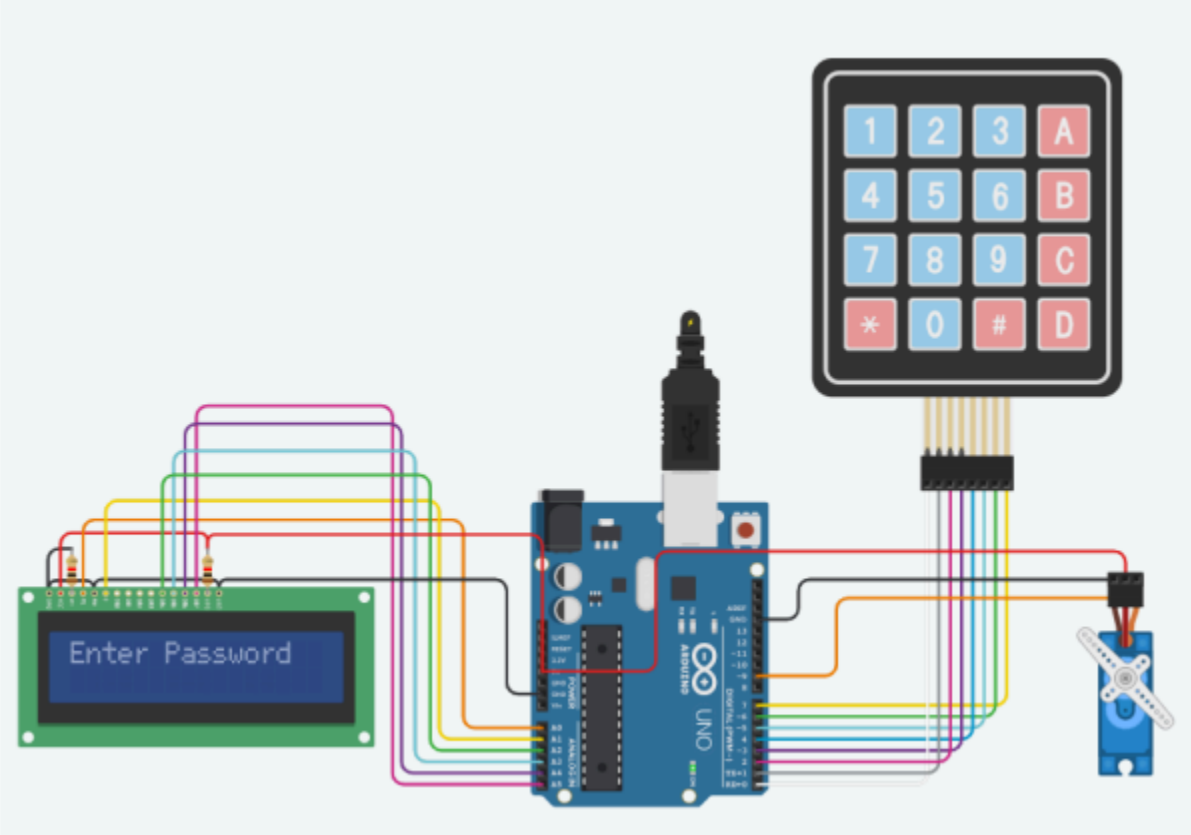
Lab assisment 2

ICS 423 Internet of Things

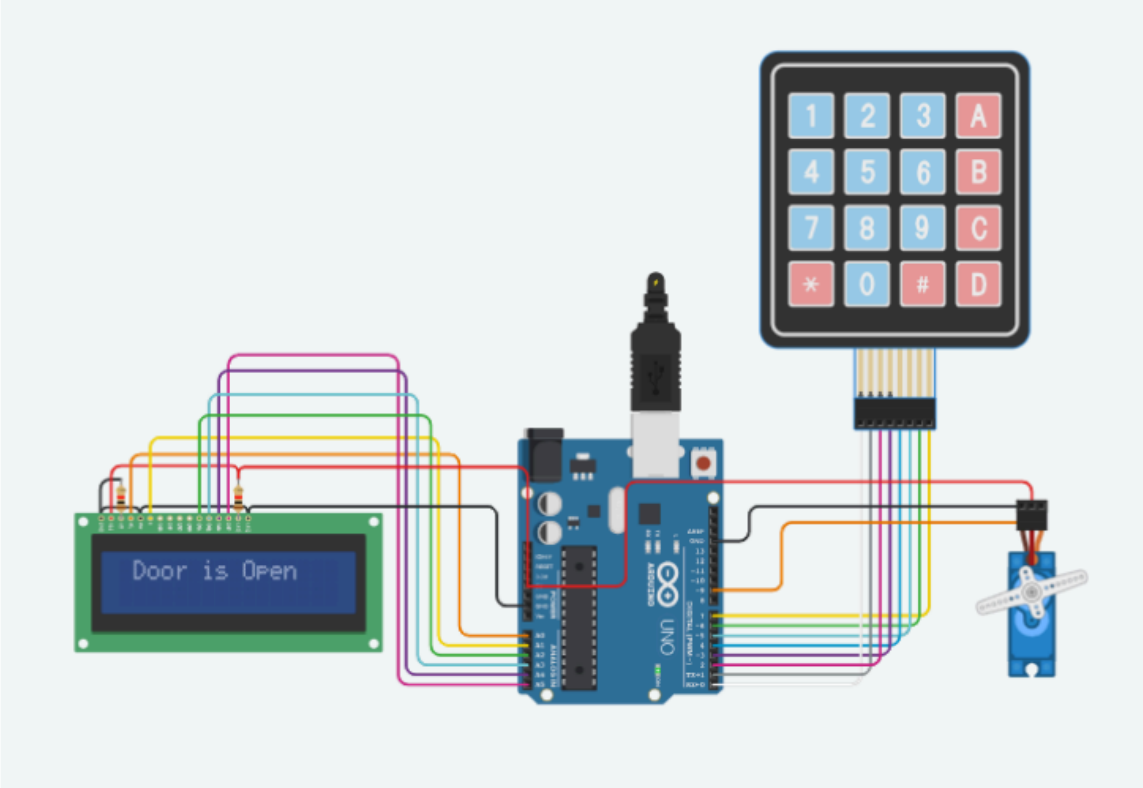
1) Design a Keypad Door Lock with an Arduino Tinkercad

Explanation : In this project we use a keypad to act as a keypad for the door lock and a servo which acts as a door. When the correct password is entered the door will open and close after a few seconds, and it will remain closed when the password entered on the keypad is incorrect. We use a 16x2 LCD as a display for the keypad door lock.

Hardware Required : 4x4 Keypad Arduino Jumper wires Resistors Power source 16x2 LCD Micro Servo



Screenshots : The door is locked initially. When the wrong password is entered the door remains closed.
When the password is correct (0100) the door unlocks



Code :

```
#include <Keypad.h>

#include <LiquidCrystal.h>

#include <Servo.h>

#define Password_Length 5

Servo myservo;

LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);

int pos = 0;
```

```
char Data[Password_Length];
char Master[Password_Length] = "0100";
byte data_count = 0, master_count = 0;
bool Pass_is_good;
bool door = false;
char customKey;
/* preparing keypad */
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {0, 1, 2, 3};
byte colPins[COLS] = {4, 5, 6, 7};
Keypad customKeypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);
void setup()
{
myservo.attach(9, 2000, 2400);
ServoClose();
lcd.begin(16, 2);
lcd.print("Protected Door");
loading("Loading");
lcd.clear();
}
```

```
void loop()
{
if (door == true)
{
customKey = customKeypad.getKey();
if (customKey == '#')
{
lcd.clear();
ServoClose();
lcd.print("Door is closed");
delay(3000);
door = false;
}
}
else
Open();
}
void loading (char msg[]) {
lcd.setCursor(0, 1);
lcd.print(msg);
for (int i = 0; i < 9; i++) {
delay(1000);
lcd.print(".");
}
}
void clearData()
{
```

```
while (data_count != 0)
{
Data[data_count--] = 0;
}
return;
}
void ServoClose()
{
for (pos = 90; pos >= 0; pos -= 10) {
myservo.write(pos);
}
}
void ServoOpen()
{
for (pos = 0; pos <= 90; pos += 10) {
myservo.write(pos);
}
}
void Open()
{
lcd.setCursor(0, 0);
lcd.print("Enter Password");
customKey = customKeypad.getKey();
if (customKey)
{
Data[data_count] = customKey;
lcd.setCursor(data_count, 1);
```

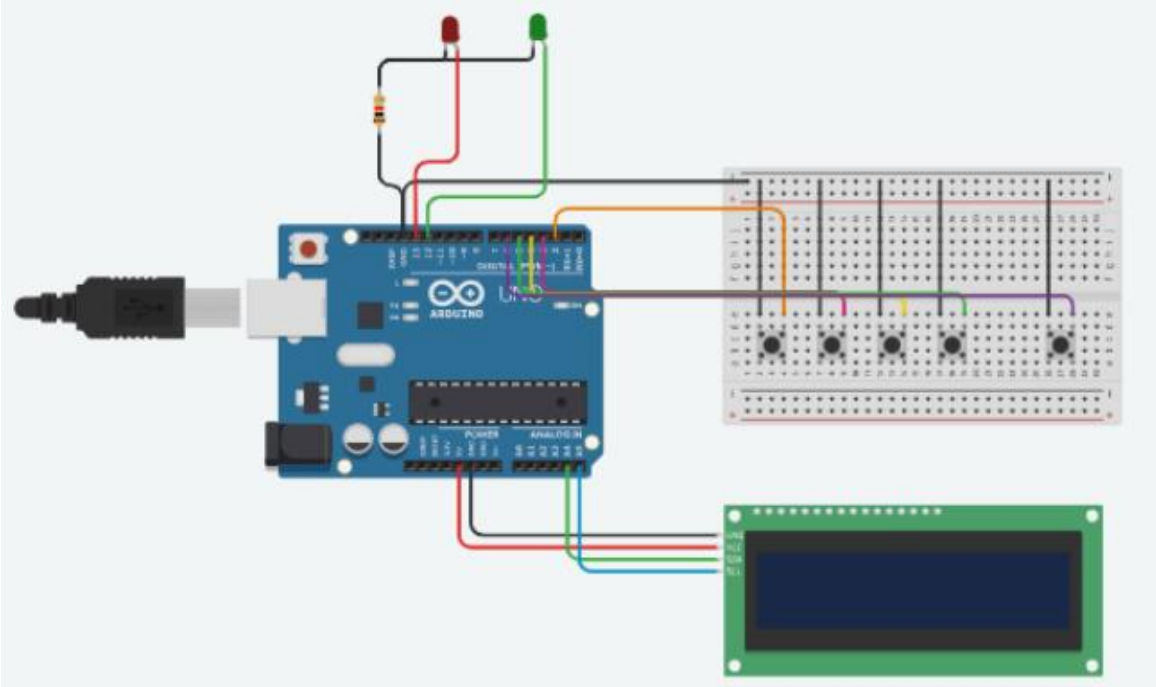
```
lcd.print(Data[data_count]);
data_count++;
}
if (data_count == Password_Length - 1)
{
if (!strcmp(Data, Master))
{
lcd.clear();
ServoOpen();
lcd.print(" Door is Open ");
door = true;
delay(5000);
loading("Waiting");
lcd.clear();
lcd.print(" Time is up! ");
delay(1000);
ServoClose();
door = false;
}
else
{
lcd.clear();
lcd.print(" Wrong Password ");
door = false;
}
delay(1000);
lcd.clear();
```

```
clearData();
}
}
```

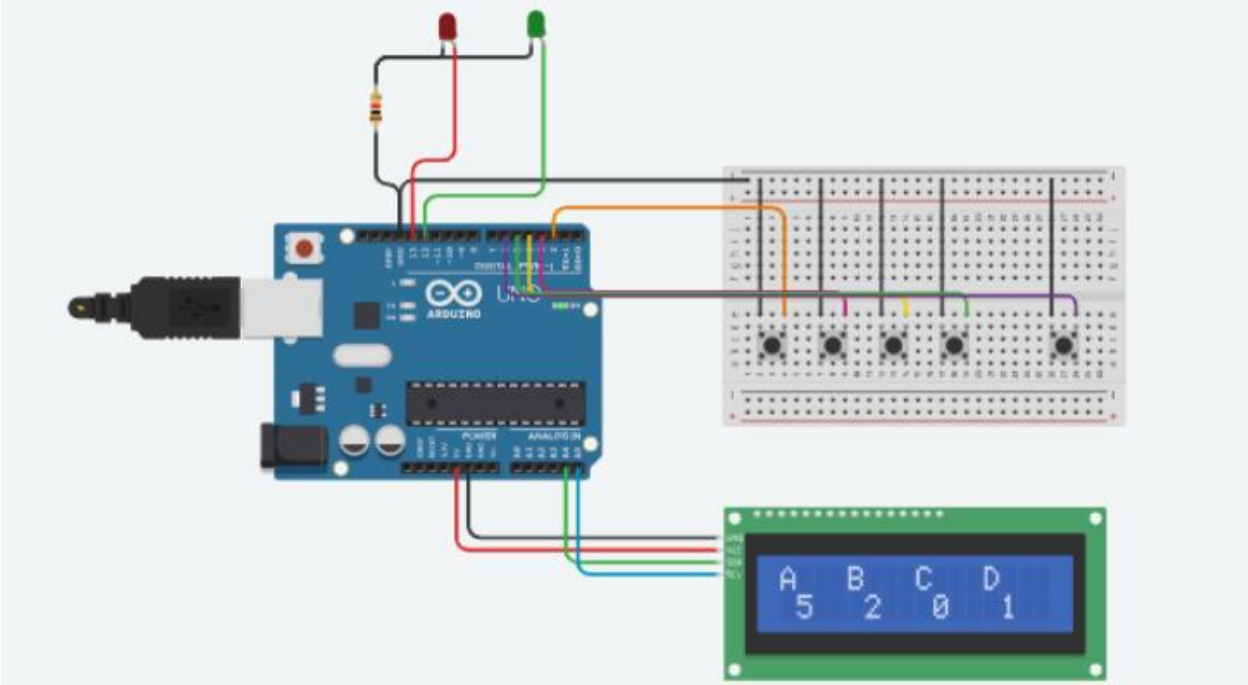
Result : The Tinkercad design successfully simulates a basic Keypad Door Lock system using an Arduino Uno, keypad, and servo motor. Functionality Verification: Through simulation, you can validate the core functionalities of the system: - Keypad responsiveness: Pressing virtual buttons on the Tinkercad keypad simulates user input. - Password recognition: The code checks for a predefined password entered on the keypad. - Servo motor control: Upon entering the correct code, the servo motor rotates to a designated angle, mimicking unlocking a door. Keypad Door Lock with Arduino Tinkercad design demonstrates a functional and potentially more secure door lock system in the real world

Electronic Voting Machine with an Arduino Tinkercad

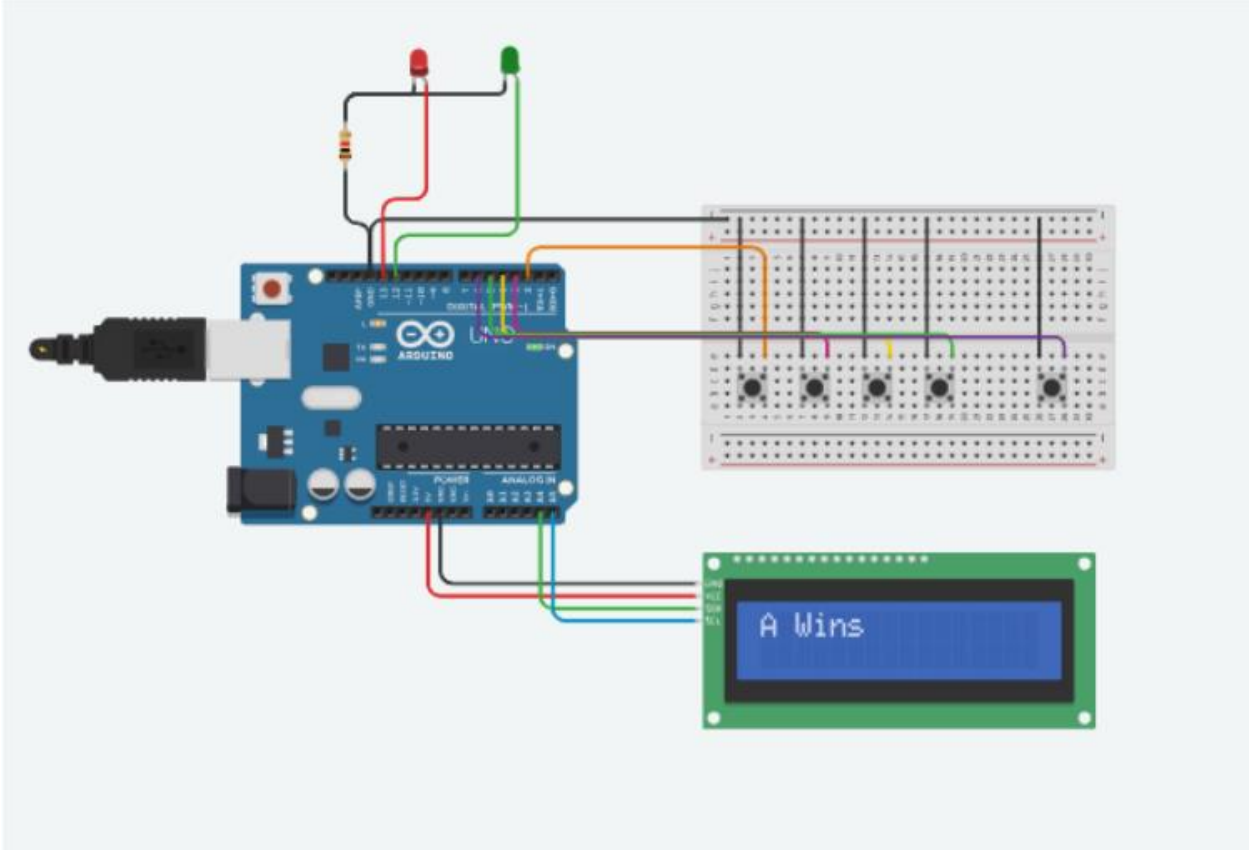
Explanation : We create an Electronic Voting machine where each of the voter can push a button corresponding to their choice, and when a voter presses a button, the green led will lit up signaling the vote has been counted and the same is shown on the LCD display with the vote increased. Finally, when the Voting process has been concluded, pressing the final button shows the results of the votes. Hardware Required : Arduino Uno Pushbuttons (for voting) LEDs (to indicate voting status) Resistors (for LEDs) Breadboard Jumper wires Setup :



Screenshots : Voting



Results



Code :

```
#include <Adafruit_LiquidCrystal.h>

Adafruit_LiquidCrystal lcd_1(0);

#define sw1 2 // Button 1

#define sw2 3 // Button 2

#define sw3 4 // Button 3

#define sw4 5 // Button 4

#define sw5 6 // Button 5 for result

int vote1=0;

int vote2=0;

int vote3=0;
```

```
int vote4=0;
void setup()
{
pinMode(sw1,INPUT);
pinMode(sw2,INPUT);
pinMode(sw3,INPUT);
pinMode(sw4,INPUT);
pinMode(sw5,INPUT);
pinMode(13,OUTPUT);// Red LED
pinMode(12,OUTPUT);// Green LED
lcd_1.begin(16, 2);
lcd_1.setCursor(0,0);
lcd_1.print(" EVM ");
lcd_1.setCursor(0,1);
lcd_1.print("Circuit Init");
delay(1000);
digitalWrite(sw1, HIGH);
digitalWrite(sw2, HIGH);
digitalWrite(sw3, HIGH);
digitalWrite(sw4, HIGH);
digitalWrite(sw5, HIGH);
lcd_1.clear();
lcd_1.setCursor(0,0);
lcd_1.print("A");
lcd_1.setCursor(4,0);
lcd_1.print("B");
lcd_1.setCursor(8,0);
```

```
lcd_1.print("C");
lcd_1.setCursor(12,0);
lcd_1.print("D");
}
void loop()
{
lcd_1.setCursor(0,0);
lcd_1.print("A");
lcd_1.setCursor(1,1);
lcd_1.print(vote1);
lcd_1.setCursor(4,0);
lcd_1.print("B");
lcd_1.setCursor(5,1);
lcd_1.print(vote2);
lcd_1.setCursor(8,0);
lcd_1.print("C");
lcd_1.setCursor(9,1);
lcd_1.print(vote3);
lcd_1.setCursor(12,0);
lcd_1.print("D");
lcd_1.setCursor(13,1);
lcd_1.print(vote4);
if(digitalRead(sw1)==0)
{
vote1++;
digitalWrite(12,HIGH);
delay(50);
```

```
while(digitalRead(sw1)==0);
digitalWrite(12,LOW);
delay(1000);
}
if(digitalRead(sw2)==0)
{
vote2++;
digitalWrite(12,HIGH);
delay(500);
while(digitalRead(sw2)==0);
digitalWrite(12,LOW);
delay(1000);
}
if(digitalRead(sw3)==0)
{
vote3++;
digitalWrite(12,HIGH);
delay(500);
while(digitalRead(sw3)==0);
digitalWrite(12,LOW);
delay(1000);
}
if(digitalRead(sw4)==0)
{
vote4++;
digitalWrite(12,HIGH);
delay(500);
```

```
while(digitalRead(sw4)==0);
digitalWrite(12,LOW);
delay(1000 );
}
if(digitalRead(sw5)==0)
{
digitalWrite(13,HIGH);
int vote=vote1+vote2+vote3+vote4;
if(vote)
{
if((vote1 > vote2 && vote1 > vote3 && vote1 > vote4))
{
lcd_1.clear();
lcd_1.print("A Wins");
delay(5000);
lcd_1.clear();
}
else if((vote2 > vote1 && vote2 > vote3 && vote2 > vote4))
{
lcd_1.clear();
lcd_1.print("B Wins");
delay(5000);
lcd_1.clear();
}
else if((vote3 > vote1 && vote3 > vote2 && vote3 > vote4))
{
lcd_1.clear();
```

```
lcd_1.print("C Wins");
delay(5000);
lcd_1.clear();
}
else if(vote4 > vote1 && vote4 > vote2 && vote4 > vote3)
{
lcd_1.setCursor(0,0);
lcd_1.clear();
lcd_1.print("D Wins");
delay(5000);
lcd_1.clear();
}
else
{
lcd_1.clear();
lcd_1.print(" Tie Up Or ");
lcd_1.setCursor(0,1);
lcd_1.print(" No Result ");
delay(5000);
lcd_1.clear();
}
}
else
{
lcd_1.clear();
lcd_1.setCursor(0,0);
lcd_1.print(" No Voting.... ");
```

```
delay(5000);

lcd_1.clear();

}

vote1=0;vote2=0;vote3=0;vote4=0,vote=0;

lcd_1.clear();

digitalWrite(12,LOW);

digitalWrite(13,LOW);

}

}
```

Result : The Electronic Voting Machine (EVM) simulation using Arduino in Tinkercad was successfully implemented. The system allowed users to cast their votes by pressing corresponding pushbuttons associated with different voting options. LEDs were used to indicate the voting status, lighting up to signify that a vote has been cast for a particular option. Upon pressing a pushbutton, the Arduino detected the input and incremented the respective vote count for the corresponding option. The system provided a simple and intuitive interface for voters to participate in the simulated voting process