FPT UNIVERSITY

IOT102 PROJECT

SMART DOOR LOCK SYSTEM USING ARDUINO

Group member:

1. Lưu Kim Hoàng SE171343
2. Lê Chí Hải SE171358
3. Nguyễn Võ Anh Kiệt SE171347
4. Nguyễn Minh Nhật SE171348

Lecturer and Researcher: Lê Thế Dũng, Ph.D.

I. PROJECT DESCRIPTIONS

Chức năng: mở khóa cửa tự động bằng cách sử dụng thẻ từ hoặc nhập mật khẩu từ bàn phím. Khi nhập đúng hoặc sai hay đưa thẻ đúng hoặc sai sẽ có âm thanh riêng biệt được phát ra.

Yêu cầu:
+ Sử dụng Arduino, keypad, RFID, buzzer(chuông)

+ Mật khẩu có độ dài tùy ý

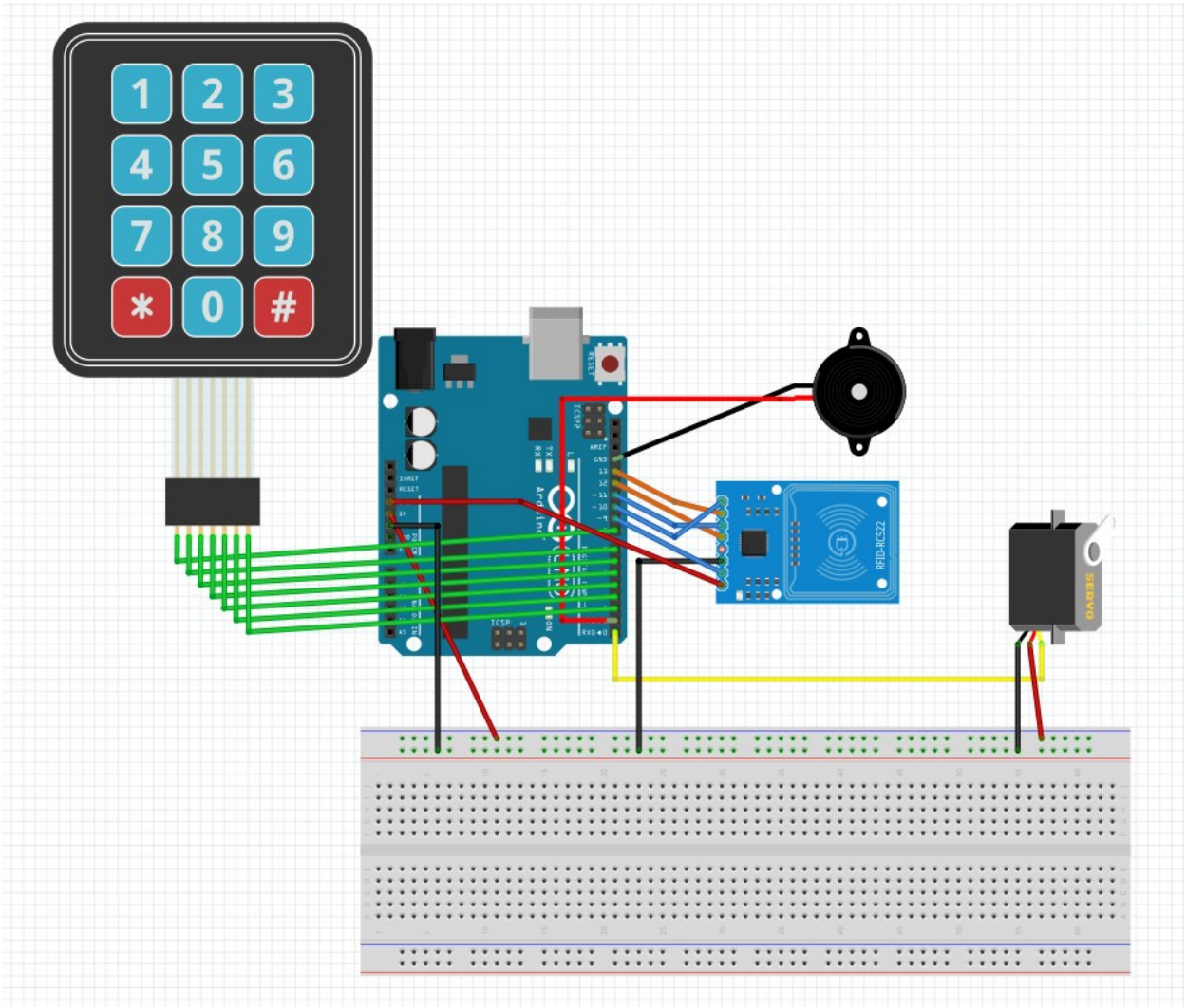
+ Đăng ký/xóa thẻ RFID

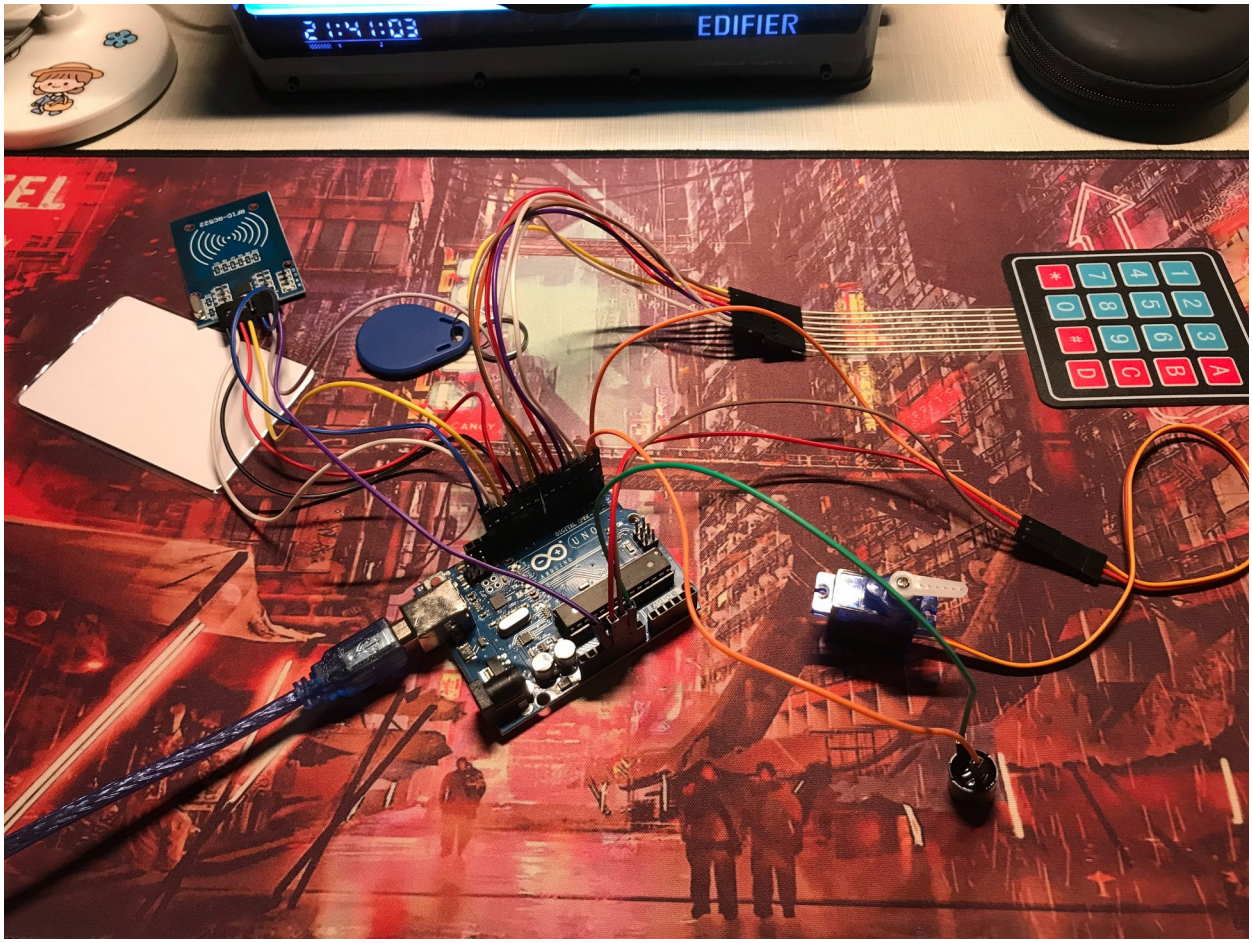
+ Đúng mật khẩu/thẻ RFID -> mở cửa và tự động đóng sau thời gian xác định trước

II. REQUIRED HARWARE

- Arduino Uno
- Keypad
- RFID
- Buzzer
- Servo
- Dây nối

III. CIRCUIT

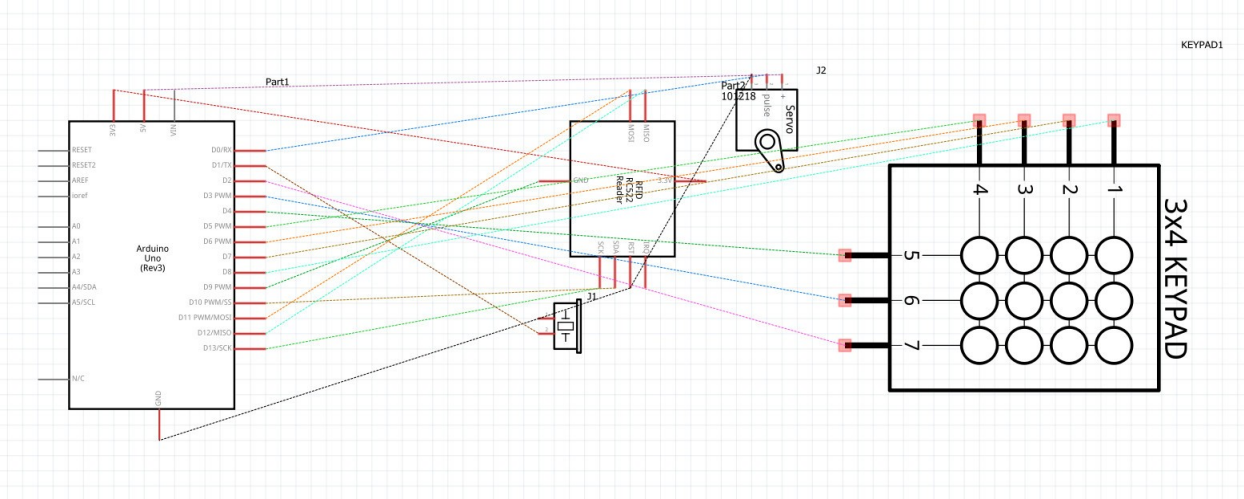

1 2 3
4 5 6
7 8 9
* 0 #
RFID-RC522
SERVO



1.RFID	Dây để Arduino Uno
SDA	số 10
SCK	số 13
MOSI	số 11
SÚP MISO	số 12
IRQ	không dùng
GND	GND
RST	số 9

3.3V	3.3V
2.SERVO	
VCC	5.5V
GND	GND
Chân điều khiển(Signal)	Số 0
3.Chuông (buzzer)	
GND	GND
Chân tín hiệu	Số 1
4.Bàn phím 3x4	
R1, R2, R3, R4	Số 8,7,6,5
C1, C2, C3	Số 4,3,2

III. SCHEMATIC



IV. CODE

```
#include <Servo.h>
#include <Keypad.h>
#include <EEPROM.h>
#include <MFRC522.h>

#define SS_PIN 10  //SS pin of RFID module
#define RST_PIN 9  //RST pin of RFID module

bool e;
int input_funct;
int add_funct;
int delete_funct;
int frequency = 500;  // initialize the starting frequency
int increment = 10;   // set the frequency increment
const int buzzerPin = 1;
// Define a string array to hold the RFID card data
String rfid_cards[10];
int num_cards = 0;

char password_nhap[] = { '0', '0', '0', '0', '0' };
char password[] = "12345";
char add_rfid[] = "*****";
char delete_rfid[] = "00000";

// Define keypad pin configuration and password
const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  { '1', '2', '3' },
  { '4', '5', '6' },
  { '7', '8', '9' },
```

```
  { '*', '0', '#' }
};
// connect the pins from right to left to pin 2, 3, 4, 5, 6, 7, 8
byte rowPins[ROWS] = { 8, 7, 6, 5 };
byte colPins[COLS] = { 4, 3, 2 };

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// Define servo pin configuration
Servo myservo;
int servoPin = 0;

// Define variables for card ID
String cardID = "";

// Define variables for EEPROM memory addresses
int address = 0;
int num_cards_address = 100;

// Define RFID module instance
MFRC522 rfid(SS_PIN, RST_PIN);

void setup() {
  // Initialize RFID module
  SPI.begin();
  rfid.PCD_Init();

  // Attach servo to its pin and set initial position
  myservo.attach(servoPin);
  myservo.write(0);
}

void loop() {
  // Wait for a card to be detected
  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
    // Get the ID of the detected card
    cardID = "";
    for (int i = 0; i < rfid.uid.size; i++) {
      cardID += String(rfid.uid.uidByte[i], HEX);
      cardID.toUpperCase();
      cardID += " ";
    }

    // Check if the detected card is in the list of allowed cards
    bool card_found = false;
    for (int i = 0; i < num_cards; i++) {
      if (cardID == rfid_cards[i]) {
        card_found = true;
        break;
      }
    }
```

```
    // If the card is allowed, unlock the door
    if (card_found) {
      setLocked(true);
    } else {
      setLocked(false);
    }

    // Wait for the card to be removed before detecting another card
    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
  }

  //enter from keypad
  char keyPress = keypad.getKey();
  if (keyPress)  // neu co phim nhan
  {
    delay(100);
    e = true;
    if (keyPress != '#') {
      //read from keypad
      tone(buzzerPin, 1000, 500);  // Generate a 1 kHz tone on the buzzer pin
      for (int i = 4; i > 0; i--) {
        password_nhap[i] = password_nhap[i - 1];
      }
      password_nhap[0] = keyPress;
    }

    if (keyPress == '#')  //when press # do the function
    {
      tone(buzzerPin, 1000, 500);  // Generate a 1 kHz tone on the buzzer pin

      if (e == true) {
        //kiem tra password voi so nhap
        for (int i = 0; i < 5; i++) {
          //password
          if (password_nhap[i] == password[4 - i]) {  //enter password and open
door
            input_funct++;
          }
          if (password_nhap[i] == add_rfid[4 - i]) {
            add_funct++;
          }
          if (password_nhap[i] == delete_rfid[4 - i]) {
            delete_funct++;
          }
        }

        //neu nhap dung
        if (input_funct == 5) {
          e = false;
```

```
          setLocked(true);  //open door
          input_funct = 0;
        } else if (add_funct == 5) {
          e = false;
          addCard();  //do add card
          add_funct = 0;
        } else if (delete_funct == 5) {
          e = false;
          deleteCard();  //do delete card
          delete_funct = 0;
        }
        //neu nhap sai
        else {
          e = false;
          setLocked(false);
          input_funct = 0;
          add_funct = 0;
          delete_funct = 0;
        }
      }
    }
  }
  noTone(buzzerPin);  // turn off the buzzer
}

void addCard() {
  // Check if there is space to add a new card
  if (num_cards < 10) {
    // Wait for a card to be detected
    while (!rfid.PICC_IsNewCardPresent()) {
      delay(10);
    }

    // Get the ID of the detected card
    cardID = "";
    if (rfid.PICC_ReadCardSerial()) {
      for (int i = 0; i < rfid.uid.size; i++) {
        cardID += String(rfid.uid.uidByte[i], HEX);
        cardID.toUpperCase();
        cardID += " ";
      }

      // Check if the card is already in the list
      bool card_found = false;
      for (int i = 0; i < num_cards; i++) {
        if (cardID == rfid_cards[i]) {
          card_found = true;
          break;
        }
      }
```

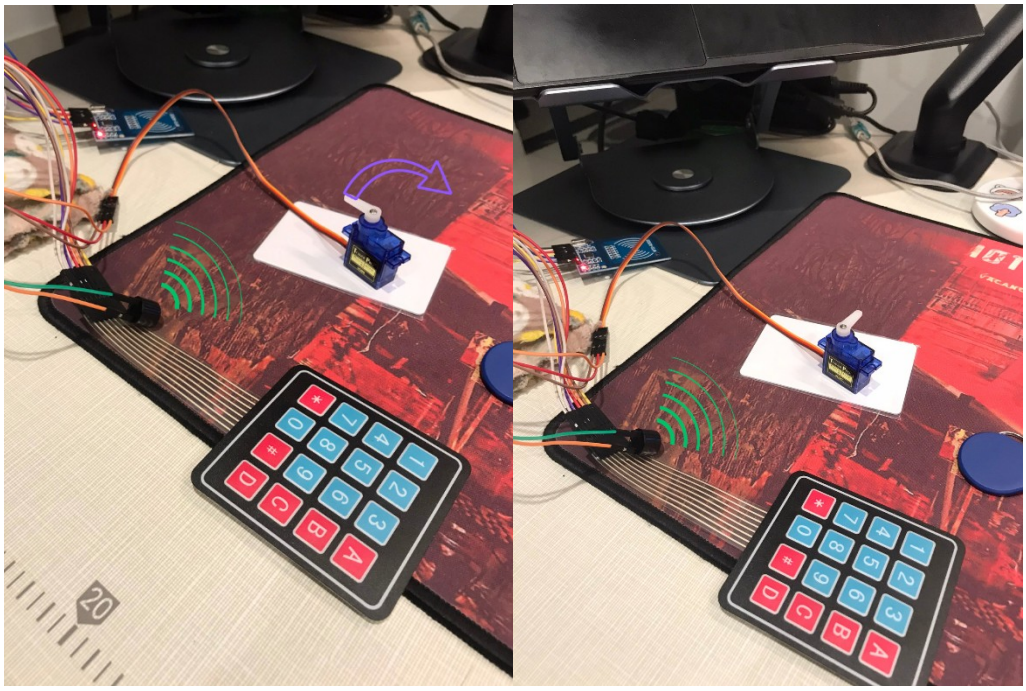
```
      // If the card is not already in the list, add it
      if (!card_found) {
        rfid_cards[num_cards] = cardID;
        num_cards++;
        EEPROM.write(num_cards_address, num_cards);
        address = (num_cards - 1) * 20 + 1;
        for (int i = 0; i < 16; i++) {
          EEPROM.write(address + i, cardID.charAt(i));
        }
        successSound();
      } else {
        failSound();
      }
      // Wait for the card to be removed before detecting another card
      rfid.PICC_HaltA();
      rfid.PCD_StopCrypto1();
    } else {
      failSound();
    }
  } else {
    failSound();
  }
}

void deleteCard() {
  // Wait for a card to be detected
  while (!rfid.PICC_IsNewCardPresent()) {
    delay(10);
  }

  // Get the ID of the detected card
  cardID = "";
  if (rfid.PICC_ReadCardSerial()) {
    for (int i = 0; i < rfid.uid.size; i++) {
      cardID += String(rfid.uid.uidByte[i], HEX);
      cardID.toUpperCase();
      cardID += " ";
    }

    // Check if the card is in the list
    int card_index = -1;
    for (int i = 0; i < num_cards; i++) {
      if (cardID == rfid_cards[i]) {
        card_index = i;
        break;
      }
    }

    // If the card is in the list, remove it
    if (card_index >= 0) {
      for (int i = card_index; i < num_cards - 1; i++) {
```

```
        rfid_cards[i] = rfid_cards[i + 1];
      }
      num_cards--;
      EEPROM.write(num_cards_address, num_cards);
      for (int i = card_index; i < num_cards; i++) {
        address = i * 20 + 1;
        for (int j = 0; j < 16; j++) {
          EEPROM.write(address + j, rfid_cards[i].charAt(j));
        }
      }
      successSound();
    } else {
      failSound();
    }
    // Wait for the card to be removed before detecting another card
    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
  } else {

    failSound();
  }
}

void setLocked(int locked) {
  if (locked) {
    myservo.write(90);  //mo cua
    successSound();
    delay(1000);
    for (int i = 5; i > 0; i--) {
      delay(1000);
    }
    myservo.write(0);
    //DONG CUA
    for (int i = 0; i < 5; i++)  //gan mang led ve 00000
    {
      password_nhap[i] = '0';
    }
  } else {
    for (int i = 0; i < 5; i++)  //gan mang led ve 00000
    {
      password_nhap[i] = '0';
    }
    delay(1000);
    failSound();
  }
}

void failSound() {
  for (int i = 0; i < 3; i++) {                                    //
repeat the loop 3 times
```

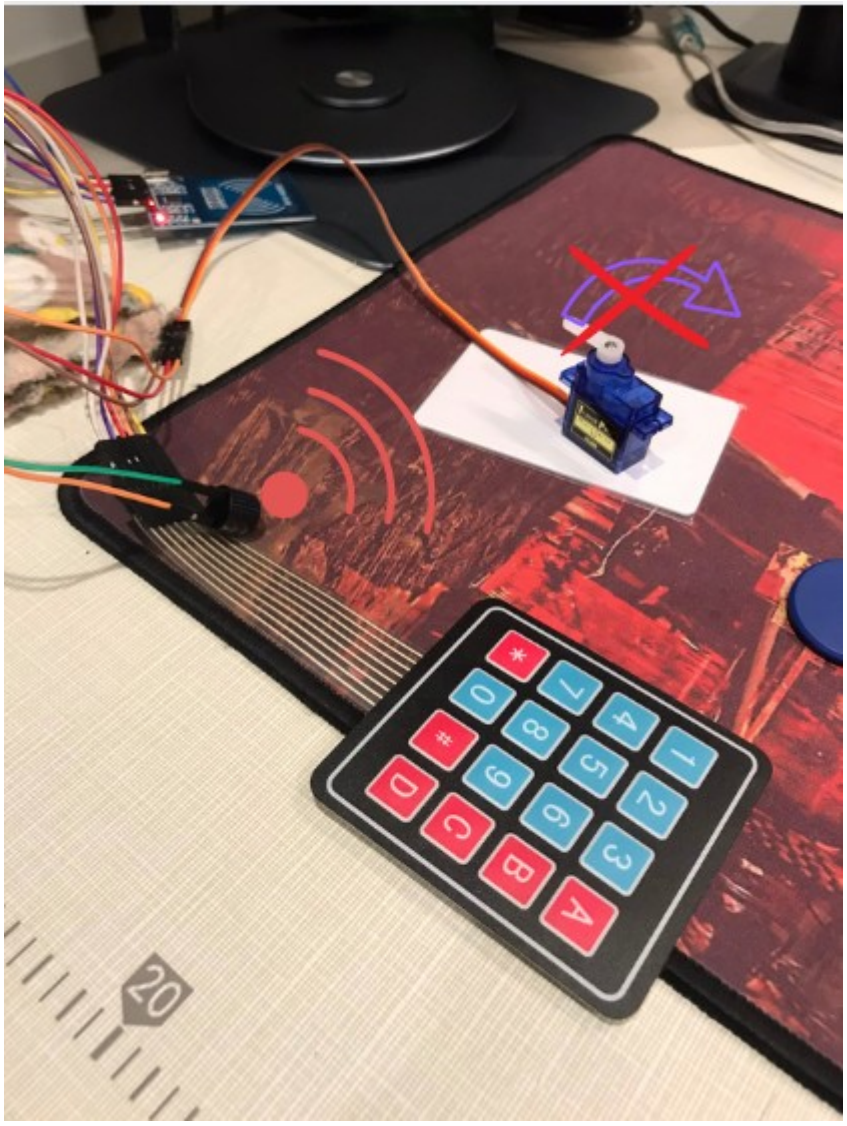
```
    for (frequency = 500; frequency <= 1500; frequency += increment) {  //
increase the frequency
      tone(buzzerPin, frequency, 10);                                   //
generate the tone
      delay(5);                                                         // wait
for a short duration
    }
    for (frequency = 1500; frequency >= 500; frequency -= increment) {  //
decrease the frequency
      tone(buzzerPin, frequency, 10);                                   //
generate the tone
      delay(5);                                                         // wait
for a short duration
    }
  }
  noTone(buzzerPin);  // turn off the buzzer
  delay(1000);        // wait for a longer duration before starting again
}
void successSound() {
  tone(buzzerPin, 1500, 200);  // Generate a 1.5 kHz tone for 200 ms
  delay(200);                  // Wait for 200 ms before generating the next
tone
  tone(buzzerPin, 2000, 200);  // Generate a 2 kHz tone for 200 ms
  delay(200);                  // Wait for 200 ms before generating the next
tone
  noTone(buzzerPin);           // Turn off the buzzer
  delay(1000);                 // Wait for 1 second before playing the sound
again
}
```

V. DEMONSTRATING RESULTS

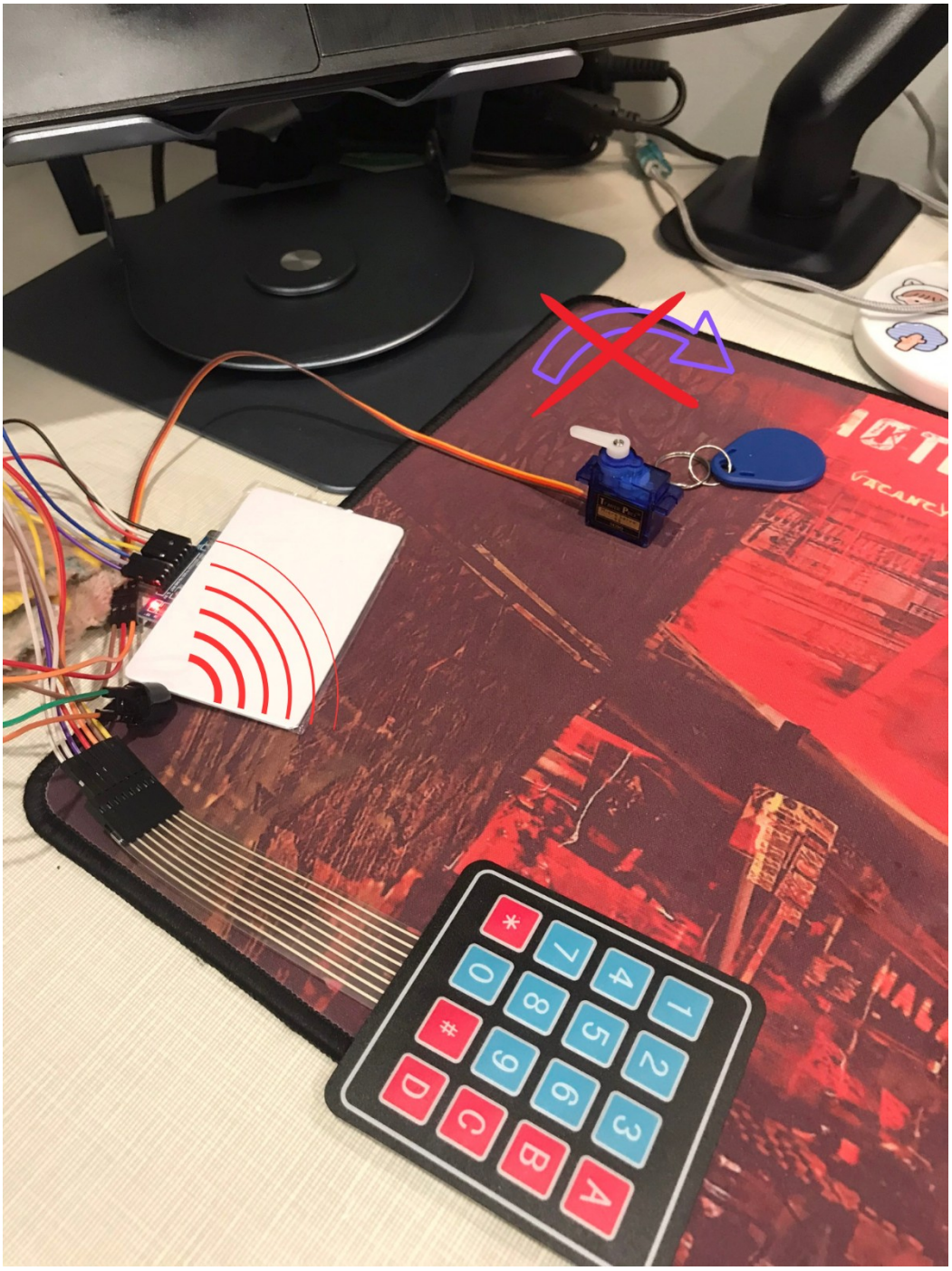
- Khi nhập mật khẩu là 1,2,3,4,5 và ấn # chuông sẽ kêu lên và Servo sẽ quay



- Khi nhập sai mật khẩu Servo sẽ không quay và chuông sẽ kêu âm thanh báo động

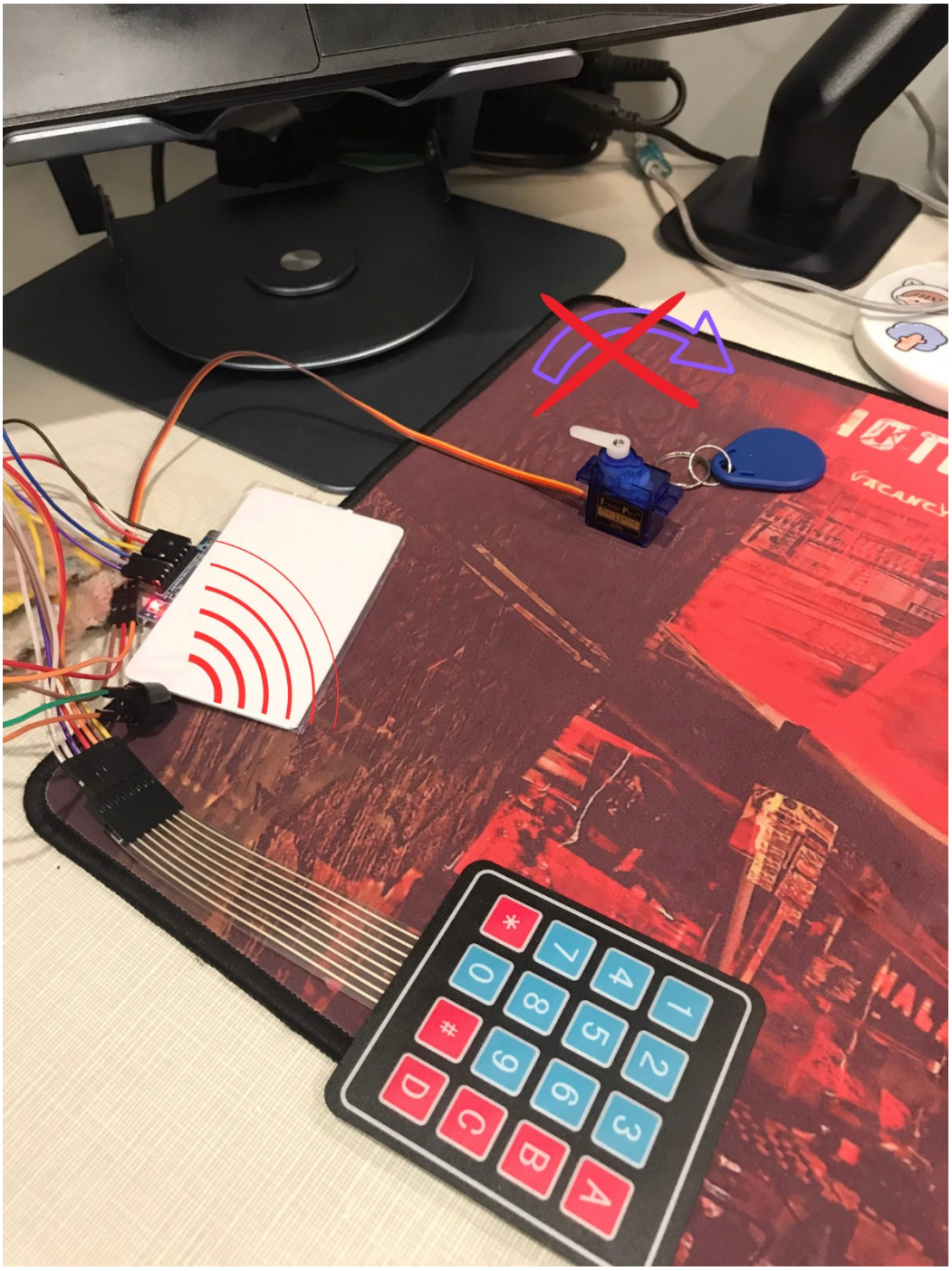


- Khi đưa thẻ chưa được đăng ký vào RFID thì chuông sẽ kêu lên âm thanh cảnh báo và Servo sẽ không quay



- Khi thẻ được ký thì chuông sẽ kêu lên âm thanh thành công và Servo quay
- Ấn * 5 lần và ấn # sau đó âm thanh báo đăng ký kêu lên và đưa thẻ cần đăng ký vào. Tiếng chuông sẽ báo thành công khi thẻ được đăng ký và ngược lại.

- Khi ấn số “0” 5 lần và ấn dấu * thì tiếng chuông báo hiệu chức năng xóa thẻ sẽ kêu lên và chúng ta đưa thẻ cần xóa vào, khi xóa thẻ thành công tiếng chuông báo sẽ kêu lên. Thẻ được xóa sẽ không sử dụng được nữa và sẽ kêu lên âm thanh cảnh báo khi ta đưa thẻ đã xóa vào RFID.


*
7
4
1
0
8
5
2
#
9
6
3
D
C
B
A