# Project Report for EE381A

**Section D Table-6**

## Department of Electrical Engineering

## 14th April, 2023

**PENTA RAMTEJ – 200683**

**MOHAN KRISHNA – 200590**

**NIHARIKA KHATRI – 200632**

# Project Title – AUTOMATED DOOR LOCK SYSTEM

Traditional lock systems have been around for centuries and have been the primary means of securing homes, buildings, and valuables. However, traditional lock systems also have several problems that can compromise security and create inconveniences for users. Here are some of the most common problems of traditional lock systems:

1)Lost or Stolen Keys: Traditional lock systems require physical keys that can be easily lost or stolen. If a key falls into the wrong hands, it can be used to gain access to the property, compromising security.

2)Key Duplication: Traditional keys can be easily duplicated, making it difficult to control who has access to the property. This can create security vulnerabilities, particularly in commercial or rental properties where keys may be passed around or lost.

3)Theft: usually theft happens from night 12:00am to morning 6:00am. So, it is important to lock all the doors during that time.

4)Inconvenience: Traditional lock systems require physical keys to be carried around, which can be inconvenient for users. Keys can be heavy and cumbersome, and users may need to carry multiple keys for different locks.

Overall, traditional lock systems have several problems that can compromise security and create inconveniences for users. Automated door lock systems offer a solution to many of these problems by providing keyless access control, remote monitoring capabilities, and improved security features.

There are various existing solutions for the above problems. Here are some solutions:

1) Make sure we close all doors by ourselves manually without any mistake in terms of locking all the doors properly and also this should be done on time.

2) Hire some security guards to take care of closing all the doors at proper time. But this may lead to heavy costs on security.

These are just a few solutions of the many existing problems for traditional lock systems.

## Solution Approach

We will develop an automatic door lock system that locks the door from 12:00am night to 6:00am morning. If we want to open the lock in between the time we can do so by typing the password on the number pad. So, we are making sure to lock the doors automatically at odd times like 12:00am to 6:00am and also allowing the user to unlock it during this time if the user wants to open the door (the door would unlock by itself after some specific amount of time, say 20 seconds). Time can be manually fed once when the power supply is connected from wi-fi module into Arduino program and the lock will act accordingly. Current time will be displayed on the lcd screen.

Moreover, this door lock can be used as a traditional number lock during normal hours, i.e. 6 am to midnight. We can lock/unlock according to our wish.

<u>Resources Required:</u>

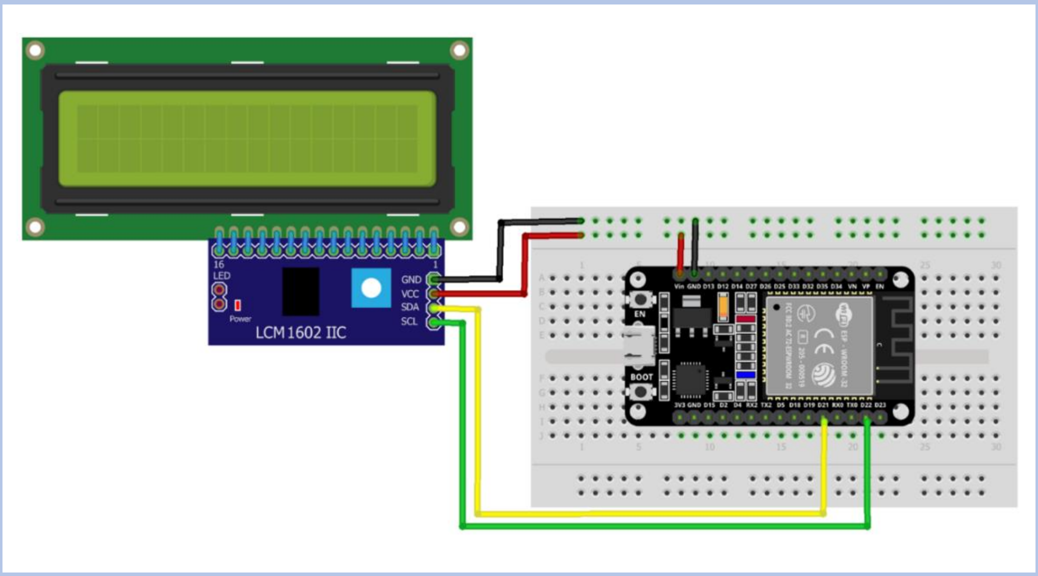
i. Arduino Uno
ii. Breadboard
iii. Jumper wires
iv. Connecting wires
v. Wi-fi module esp32
vi. Number keypad
vii. Solenoid lock
viii. LCD 16x2 display
ix. I2c module
x. Relay

<u>Software Used:</u>

i. Arduino IDE

## Implementation Details

1) Note that the time is projected onto the LCD display from ESP32 module. We require to connect LCD display with an I2c. The circuit diagram is as follows:



The code is as follows:

```
#include <WiFi.h>
#include "time.h"
#include <LiquidCrystal_I2C.h>

// set the LCD number of columns and rows
int lcdColumns = 16;
int lcdRows = 2;

// set LCD address, number of columns and rows
// if you don't know your display address, run an I2C scanner sketch
LiquidCrystal_I2C lcd(0x3F, lcdColumns, lcdRows);

const char* ssid = "Redmi note 9";
const char* password = "sairam1607";
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 19801;
const int daylightOffset_sec = 0;


void setup() {
  // Initialize The LCD. Parameters: [ Columns, Rows ]

  // Clears The LCD Display
  lcd.init();
  lcd.backlight();
  Serial.begin(9600);

  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected.");

  // Init and get the time
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  printLocalTime();

  //disconnect WiFi as it's no longer needed
  WiFi.disconnect(true);
  WiFi.mode(WIFI_OFF);
}
void loop() {

  printLocalTime();
}
```

```
void printLocalTime() {

  lcd.setCursor(3, 0);

  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
  }
    lcd.print(&timeinfo,"%H:%M:%S");
    delay(1000);
    lcd.clear();
     lcd.setCursor(3, 1);
     lcd.print(&timeinfo,"%D");
  // Display The First Message In Home Position (0, 0)
 Serial.println(&timeinfo,"%H:%M:%S");

}
```

Note that we require to include the required libraries, and time.h library is imported from the following github link: https://github.com/PaulStoffregen/Time

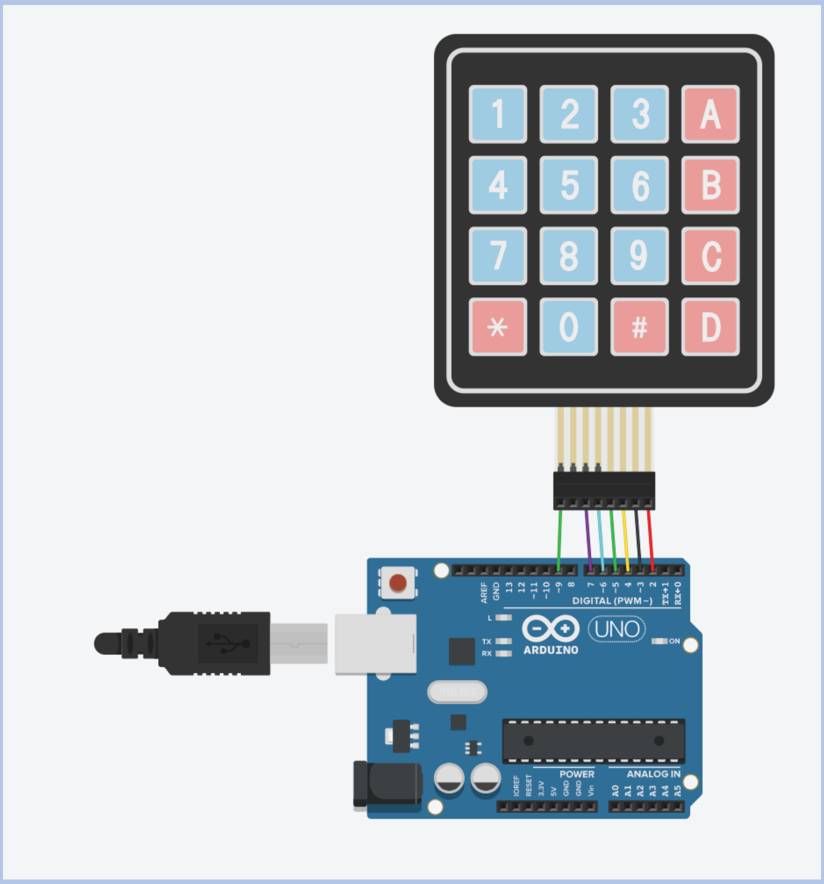
ssid and password have to be changed at respective places, according to that of the wifi from which we are taking in Internet time. Note that internet connectivity is required at the start of the program, after which the internet could be disconnected (once the time is fed into ESP32).

2) From the LCD display, again at the starting of Arduino program, we need to manually change the current date and time, using the setTime() function: setTime(hours, minutes, seconds, days, months, years). The following code does two works: a) It automatically closes the door lock at a specified time and opens it. B) It provides the door lock/unlock feature using Keypad.

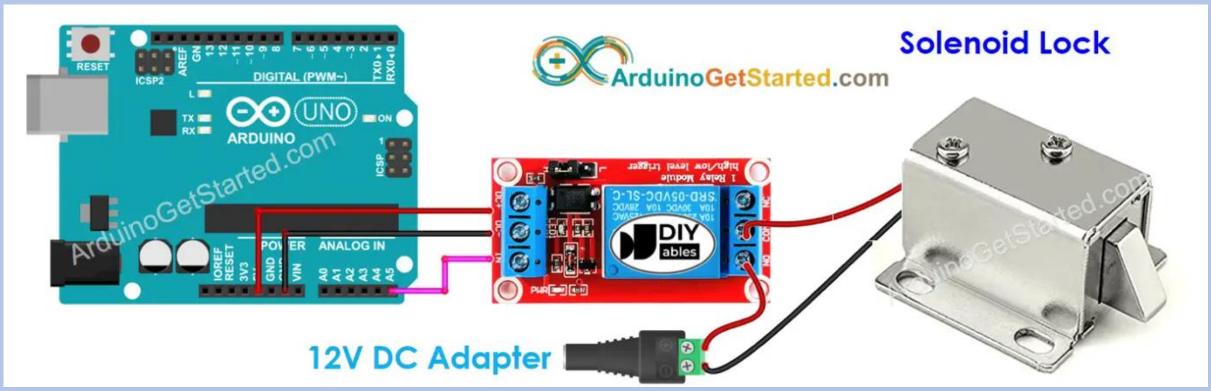
```
#include <TimeLib.h>
#include <Keypad.h>
int flagopen = 0;
int flagclose = 0;  //changes
// constants won't change
const int RELAY_PIN = A5;  // the Arduino pin, which connects to the IN
pin of relay
const int ROW_NUM = 4;     // four rows
const int COLUMN_NUM = 4;  // four columns


char keys[ROW_NUM][COLUMN_NUM] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};
byte pin_rows[ROW_NUM] = { 9, 8, 7, 6 };      //connect to the row
pinouts of the keypad
byte pin_column[COLUMN_NUM] = { 5, 4, 3, 2 };  //connect to the
column pinouts of the keypad
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column,
ROW_NUM, COLUMN_NUM);
// the setup function runs once when you press reset or power the
board
const String password_1 = "1234ABC";  // change your password here
const String lock = "567";           // change your password here

String input_password;

void setup() {
  // initialize digital pin 3 as an output.

  Serial.begin(9600);
  input_password.reserve(32);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, HIGH);
  setTime(14, 5, 55, 13, 4, 2023);  //change
}
// the loop function runs over and over again forever
void loop() {
  //changes
  if (hour() <6 && hour() >= 0) {
    digitalWrite(RELAY_PIN, HIGH);  // lock the door
    char key = keypad.getKey();
    if (key) {
      Serial.println(key);

      if (key == '*') {
        input_password = "";  // reset the input password
      } else if (key == '#') {
        if (input_password == password_1) {
          Serial.println("The password is correct, unlocking the door for 20
seconds");
          digitalWrite(RELAY_PIN, LOW);  // unlock the door for 20 seconds
          delay(20000);
          digitalWrite(RELAY_PIN, HIGH);  // lock the door
        } else if (input_password == lock) {
          Serial.println("Locking the door:");
          digitalWrite(RELAY_PIN, HIGH);
        }
      }
    }
    } else if (hour() == 6 && minute() ==0 && second() == 01) {
      digitalWrite(RELAY_PIN, LOW);  // unlock the door
      delay(1000);
    }

    else {
      char key = keypad.getKey();

      if (key) {
        Serial.println(key);
        if (key == '*') {
          input_password = "";  // reset the input password
        } else if (key == '#') {
          if (input_password == password_1) {
            Serial.println("The password is correct, unlocking the door");
            digitalWrite(RELAY_PIN, LOW);  // unlock the door for 20 seconds
          }

          else if (input_password == lock) {
            Serial.println("Locking the door:");
            digitalWrite(RELAY_PIN, HIGH);
          }

          else {
            Serial.println("The password is incorrect, try again");
          }

          input_password = "";  // reset the input password
        } else {
          input_password += key;  // append new character to input
password string
        }
      }
    }
}
```

The relay is used because generally the signal swing of an Arduino is just 5 volts, but the solenoid lock requires that of 12 volts. Hence, we use a power supply of 12 volts along with a relay, as shown in the diagram, to amplify the signal. The pin connection is to be noted.

The circuit diagram for connecting keypad to Arduino uno Is as follows:



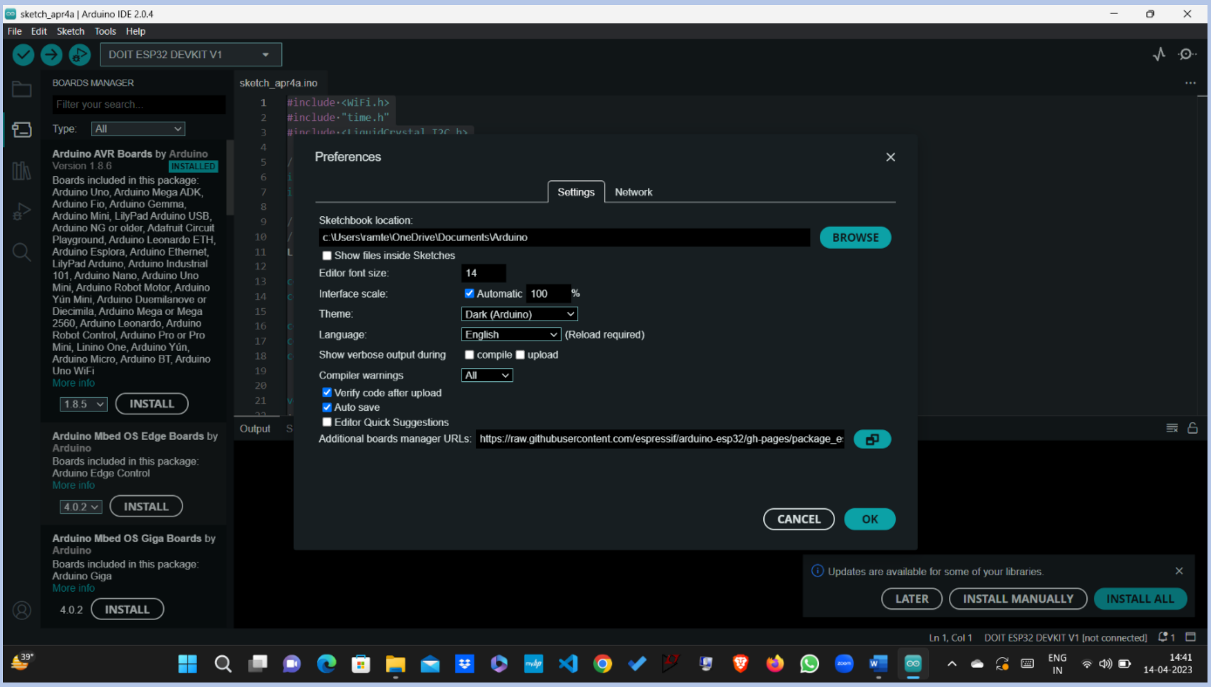
The circuit diagram for connecting relay/door lock to Arduino is (credits: ArduinoGetStarted.com):



3) To configure Arduino IDE:
    a) For Arduino UNO:  Select the board Arduino UNO from the boards manager>>Arduino AVR boards. Upload the code by clicking the upload button.

b) For ESP32:

Choose the correct board for ESP32 (in our case it was DOIT ESP32 DEVKIT V1). If the board is not loaded, then upload using the following link in preferences (CTRL+comma): https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json



Arduino AVR Boards by Arduino
Version 1.8.6 INSTALLED
Boards included in this package: Arduino Uno, Arduino Mega ADK, Arduino Fio, Arduino Gemma, Arduino Mini, LilyPad Arduino USB, Arduino NG or older, Adafruit Circuit Playground, Arduino Leonardo ETH, Arduino Esplora, Arduino Ethernet, LilyPad Arduino, Arduino Industrial 101, Arduino Nano, Arduino Uno Mini, Arduino Robot Motor, Arduino Yún Mini, Arduino Duemilanove or Diecimila, Arduino Mega or Mega 2560, Arduino Leonardo, Arduino Robot Control, Arduino Pro or Pro Mini, Linino One, Arduino Yún, Arduino Micro, Arduino BT, Arduino Uno WiFi
More info
1.8.5 INSTALL

Arduino Mbed OS Edge Boards by Arduino
Boards included in this package: Arduino Edge Control
More info
4.0.2 INSTALL

Arduino Mbed OS Giga Boards by Arduino
Boards included in this package: Arduino Giga
More info
4.0.2 INSTALL

```
#include <WiFi.h>
#include "time.h"
#include <LiquidCrystal_I2C.h>
```

Sketchbook location: c:\Users\ramte\OneDrive\Documents\Arduino
[ ] Show files inside Sketches
Editor font size: 14
Interface scale: [✓] Automatic 100 %
Theme: Dark (Arduino)
Language: English (Reload required)
Show verbose output during [ ] compile [ ] upload
Compiler warnings: All
[✓] Verify code after upload
[✓] Auto save
[ ] Editor Quick Suggestions
Additional boards manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_e

Updates are available for some of your libraries.

After choosing the board, download the Zip files from the github links LiquidCrystal_I2C.h and time.h and add the libraries from Sketch>>Include Library>>Add .zip Library .After writing the code click on upload. While uploading the code, after “Connecting…” appears on Output, long press Boot Button of ESP32 till code starts to get written in ESP32. As soon as code gets uploaded, click on reset button of ESP32. The ESP32 will start behaving according to the code uploaded. If the code doesn’t get uploaded then it might be because of no required drivers. Download the drivers for esp32 from the following link https://www.silabs.com/documents/public/software/CP210x_Windows_Drivers.zip

**Results:**

The door lock is in unlocked mode at a certain period of time and is in locked mode in time other than that period and the door lock can be put in opposite mode by entering the respective password using the number pad. Time and Date is also displayed on the lcd display. The demonstration was shown in lab.

## Discussions

**Scope of Improvement:**

Note that we have to manually feed in time into Arduino, from lcd display. This is because while serially communicating from ESP32 with Arduino, we have to connect rx and tx pins of esp32 and Arduino. Because of which, the Arduino actually ignores the data that is given by the keypad. So, we have to choose between attaching a keypad or communicating using ESP32.

Note that this can be improved if we have either of the following:

a) RTC
b) Arduino with GPS enabled
c) Arduino with Wifi/Ethernet

Some sample programs are present in the following link:
https://github.com/PaulStoffregen/Time/tree/master/examples/TimeSerial

We can hence take in time without using ESP32, and display (with minor modifications) on LCD using Arduino and i2c module.

## Major Challenges

Our major challenge was to include as many functionalities as possible using the equipment available to us in lab/limited pins on Arduino uno. Moreover, one setback was when we accidentally connected 24 volts across the relay, which actually damaged the solenoid lock.

One unsolved problem was serially communicating using esp32 with Arduino to transfer time automatically. It took us a lot of time to realise the fallacy in the process, after which we thought we would change the methodology, and manually feed in time (since whenever we need to start the program/door lock, we need to feed in code from Arduino Ide into Arduino Uno and ESP32. So anyways we need computer-assistance just at the starting, or whenever there is a power failure. Hence, adding this small inconvenience of feeding in time manually was considered as a feasible alternative).

Finding out the perfect libraries for our purpose, and setting up boards in Arduino (for esp32) was also an obstacle. Lastly, debugging and modifying the code also took time.

## Precautions

1) The wires need to be connected meticulously, and they should not be loose (especially those of the keypad).
2) The power supply should be connected properly.
3) There should be proper internet connection using wifi.
4) Program should be loaded into Arduino and ESP32 properly.
5) The voltage across the door lock should be limited to 12 volts, else it could get damaged.
6) The knob on i2c controlling the contrast should be adjusted properly, so that the time display on LCD is visible properly. In case it is not so, then the backlight could be switched off.
7) The ports should be connected properly, taking care of the respective port numbers and circuit diagram.
8) Polarities of the door lock and relay should be taken care of.
9) In case the solenoid lock overheats, it should be disconnected immediately else the circuit could burn/lock could get permanently damaged.