Internet of Things-CSE3009

Smart Door Locking System

Review-3

Submitted By-

Jasshu Garg

19BCT0082

Software Solution

Data/Database includes

- UIDs of different Persons
- Passcode
- Storing Log

Algorithm:

1. The System Validates the UID of the person
2. Only if the UID is valid, the person will be able to enter the Passcode
3. The person will can have maximum 3 attempts to fill the passcode and after that UID Tag needs to be verified again
4. If the person enters the Correct passcode within or less than 3 attempts the Door will be opened.
5. There will be no action If there is an invalid UID, but the Log will be created.

Each Data will be logged into the cloud at Amazon Web Services (AWS)

Mathematical Model (Conditional Statement)

- Inputs:
  - RFIDpassed (0 or 1),
  - Passcode Passed (0 or 1),
- Outputs: Open the door (1), Do not open the door (0)

output = 2 <= (RFIDpassed+PasscodePassed)

Programming Environment

Both Boards are Programmed with Arduino IDE from Windows Machine

Programming Language - Embedded C

Embedded C language is used to develop microcontroller-based applications. Embedded C is an extension to the C programming language including different features such as addressing I/O, fixed-point arithmetic, multiple-memory addressing, etc. In embedded C language, specific compilers are used.

System Requirements:

- Microsoft Windows 7, Windows 8/8.1 and Windows 10 operating system.
- Microsoft .NET Framework 4.0 or higher.
- Intel Pentium / AMD Athlon processor or equivalent running at 1 GHz or more.
- 512 MB RAM (1 GB RAM recommended).
- 10MB free hard drive space or more (only for PROGRAMINO IDE for Arduino).

Compiled/Interpreted

The Arduino code is a working subset of the C/C++ programming language.

The C/C++ programming language is a compiled one, not an interpreted language.

Why this language?

C provides optimized machine instructions for the given input, which increases the performance of the embedded system. Most of the high-level languages rely on libraries, hence they require more memory which is a major challenge in embedded systems.

Packages Required:

Boards in Arduino IDE:

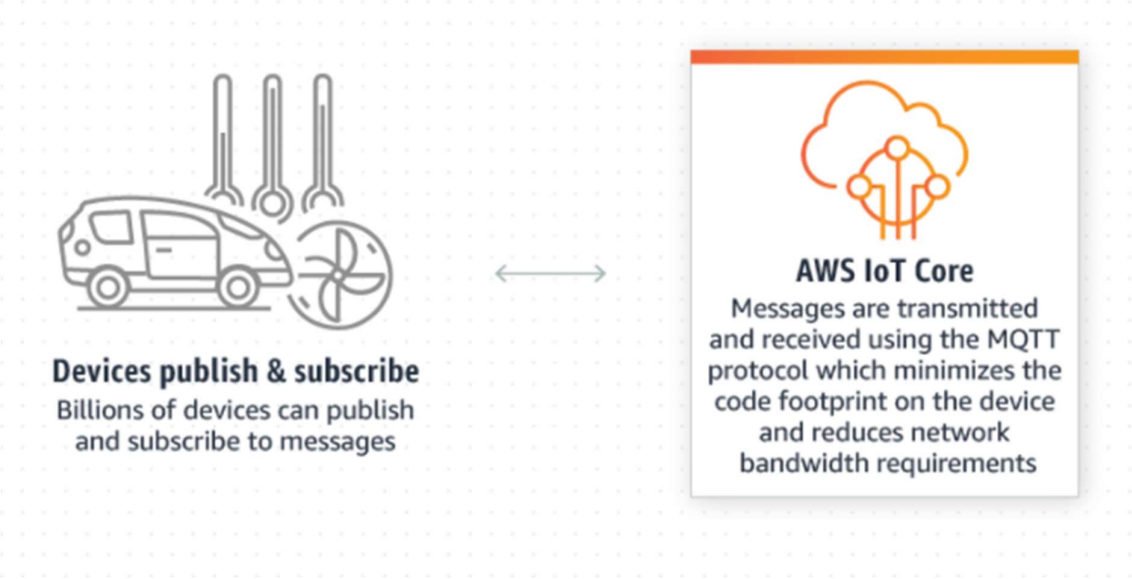
- Arduino UNO
- NodeMCU (Esp8266)

Libraries:

- #include <SPI.h>
- #include <MFRC522.h>
- #include <Keypad.h>
- #include <Servo.h>
- #include "FS.h"
- #include "ESP8266WiFi.h "
- #include "PubSubClient.h "
- #include "NTPClient.h "
- #include "WiFiUdp.h "
- #include <SoftwareSerial.h>

Cloud platform: AWS IoT Core

AWS IoT Core supports device connections that use the MQTT protocol



Analysis:

IoT core provides a JSON file of the MQTT messages tranfered , which can be used as an API for analysing and extracting information like

- Frequency of a person getting entry
- Frequency of Emergency situations
- Time of Emergency situations
- Frequency of incorrect passwords by each individual
- Usual/Unusual Entry Time of individuals

Or

AWS IoT Analytics can also be used to analyse the data collected using MQTT services.

Results:

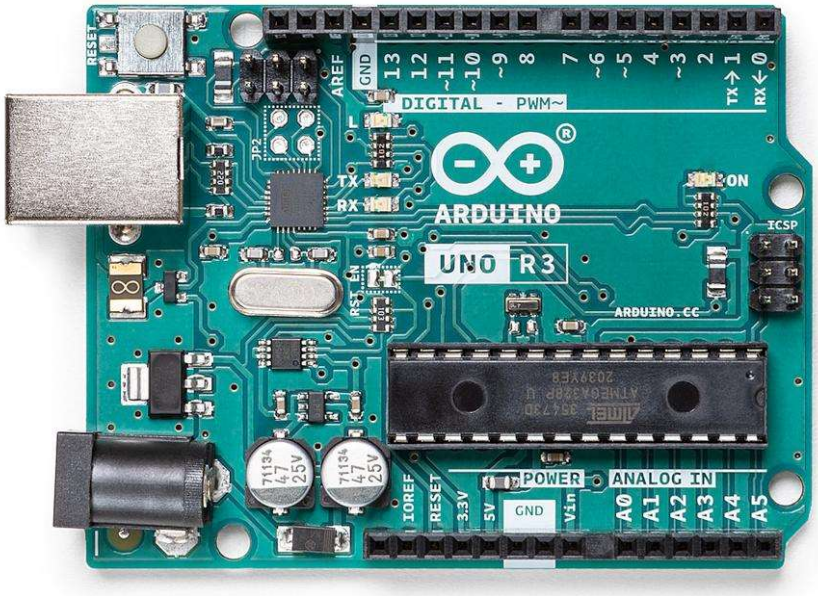
The code written for Arduino UNO and NodeMCU worked well, tested by hardware.

The data is getting published to AWS IoT Core using MQTT service to a Topic.

And AWS IoT Analytics is monitoring the data by subscribing to that IoT core Topic.

Hardware Solution:

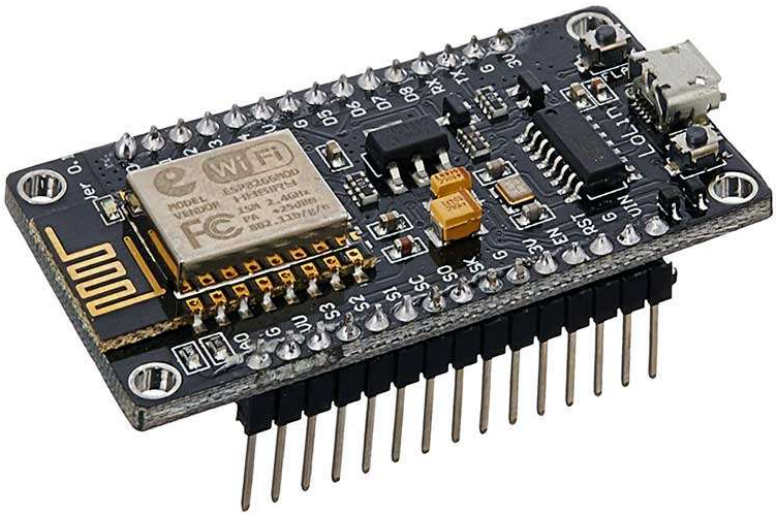
Development Board Chosen for Processing: Arduino Uno



Arduino Uno is a microcontroller board based on the ATmega328P

MICROCONTROLLER	ATmega328P
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	14 (of which 6 provide PWM output)
PWM DIGITAL I/O PINS	6
ANALOG INPUT PINS	6
DC CURRENT PER I/O PIN	20 mA
DC CURRENT FOR 3.3V PIN	50 mA
FLASH MEMORY	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
CLOCK SPEED	16 MHz
LED_BUILTIN	13

NodeMCU (ESP8266) for Internet Connectivity:



NodeMCU ESP8266 Specifications & Features

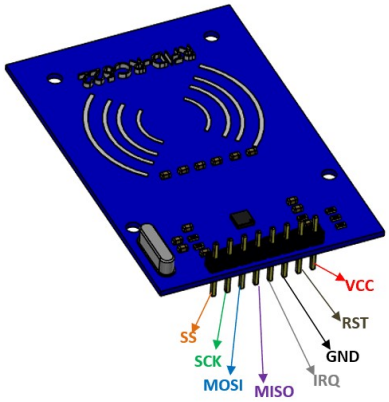
- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna

Processing Element: ATmega328P

Max ADC Resolution (bits)	**10**
Program Memory Size (KB)	**32**
Capture/Compare/PWM (CCP)	**1**
Number of Comparators	**1**
CPU Speed (MIPS/DMIPS)	**20**
Data EEPROM (bytes)	**1024**
DigitalTimerQty_16bit	**1**
Max 8 Bit Digital Timers	**2**
Ethernet	**None**
I2C	**1**
Program Memory Type	**Flash**
ADC Channels	**8**
Low Power	**Yes**
Operating Voltage	**1.8 - 5.5**
outputcomparatorPWM	**6**
Pin Count	**32**

Components:

MFRC522 Module



Principle:

- RFID system is Derived from Automatic Identification and Data Capture (AIDC) technology.
- AIDC is used for object identification, object data collection and mapping of the collected data to computer systems with little or no human intervention.
- RFID uses radio waves to perform AIDC functions.

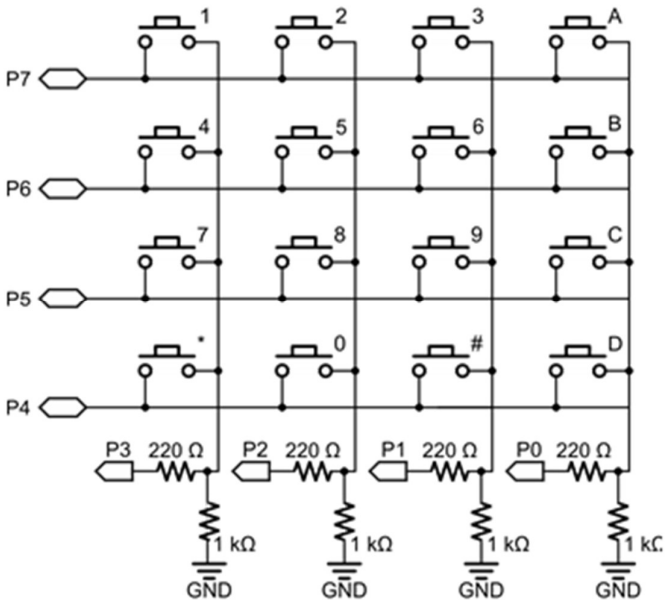
Reader transmits certain amount of energy in form Electromagnetic waves.
These waves reach the tags as a carrier wave.
Initial carrier wave is used for harvesting RF power to electric power
These waves power the Tag and the Tag gives the data back to the reader.
Tag is essentially doing back scatter of the data on the same forwarded RF power wave. impedance of the wave transmitted by the reader antenna is changed by the chip after getting powered by the initial carrier wave from reader antenna RF power

Code:

```
int validrfid()
{
  if (!mfrc522.PICC_IsNewCardPresent())
  {
    return 0;
  }
  if (!mfrc522.PICC_ReadCardSerial())
  {

    return 0;
  }


  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
//      Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
//      Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
```

```
  content.toUpperCase();
  if (content.substring(1) == "D7 58 82 D6")
  {
    Serial.print(content +" Passed the Card Authentication");
    return 1;
  }
  else
  {

    Serial.print(content +" Unrecognized Card");
     content = "";
    delay(3000);
    return 0;
  }
}
```

4x4 matrix keypad



Working principle:

This 4x4 matrix keypad has 16 built-in pushbutton contacts connected to row and column lines. A microcontroller can scan these lines for a button-pressed state. In the keypad library, the Propeller sets all the column lines to input, and all the row lines to input. Then, it picks a row and sets it high. After that, it checks the column lines one at a time. If the column

connection stays low, the button on the row has not been pressed. If it goes high, the microcontroller knows which row (the one it set high), and which column, (the one that was detected high when checked).

Interfacing:

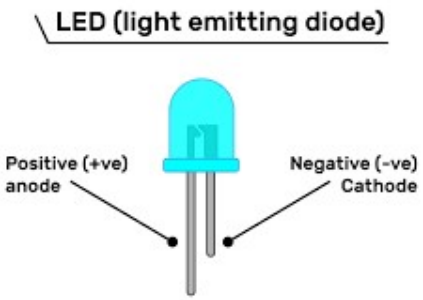
Code:

```
const byte n_rows = 4;
const byte n_cols = 4;
String content;
char keys[n_rows][n_cols] = {
   {'1', '2', '3', 'A'},
   {'4', '5', '6', 'B'},
   {'7', '8', '9', 'C'},
   {'*', '0', '#', 'D'}};

byte rowPins[n_rows] = {14, 3, 2, 8};
byte colPins[n_cols] = {7, 6, 5, 4};

Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins,
n_rows, n_cols);
```

```
char key = myKeypad.getKey();
```

LEDs

Working Principle: A light-emitting diode is a two-lead semiconductor light source. It is a p–n junction diode that emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons.



Interfacing

Relay

Working principle

- Relay works on the principle of electromagnetic induction.
- When the electromagnet is applied with some current it induces a magnetic field around it.
- Above image shows working of the relay .A switch is used to apply DC current to the load.
- In the relay Copper coil and the iron core acts as electromagnet.
- When the coil is applied with DC current it starts attracting the contact as shown. This is called energizing of relay.
- When the supply is removed it retrieves back to the original position. This is called De energizing of relay.

There are also such relays, whose contacts are initially closed and opened when there is supply i.e., exactly to opposite to the above shown relay.

Solid state relays will have sensing element to sense the input voltage and switches the output using opto-coupling.

Interfacing:

Code :

```
// Arduino Relay Control Code

#define relay A0
#define interval 1000
void setup() {
  pinMode(relay, OUTPUT);
}
void loop()
{
   digitalWrite(relay, HIGH);
   delay(interval);
   digitalWrite(relay, LOW);
   delay(interval);
}
```

Solenoid Lock

Working Principle:

The solenoid lock is available in unlocking in the power-on mode type, and locking and keeping in the power-on mode type, which can be used selectively for situations.

The power-on unlocking type enables unlocking only while the solenoid is powered on. A door with this type is locked and not opened in case of power failure or wire disconnection, ensuring excellent safety. This type is used mainly for places requiring crime prevention.

The power-on locking type can lock a door while the solenoid is powered on. If the power is disconnected, the door is unlocked. This type unlocks the door in case of wire disconnection due to a fire or accident, and it is used for emergency exits through which fire-fighting activity or evacuation should preferentially be made rather than safety for crime prevention.

The keeping type performs two operations, locking and unlocking by applying a positive or negative pulse voltage to the solenoid, and keeps the no-power state in each position. This type features energy saving because it is unnecessary to always power the solenoid on. For the continuous rating and the intermittent rating, the continuous rating is designed to be able to feed a rated voltage power continuously for hours without exceeding a specified temperature rise limit, and the intermittent rating is designed to be able to feed a specified voltage only for a specified time duration without exceeding a specified temperature rise limit.

Interfacing:

Code:

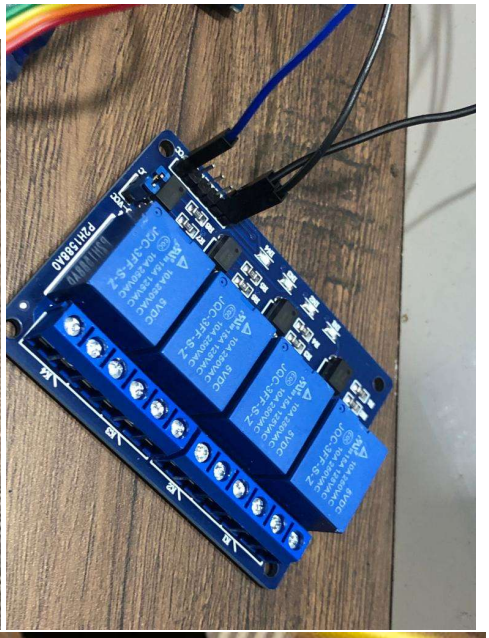
```
void setup ()
  {
  pinMode (12, OUTPUT); // RELAY PIN
  digitalWrite (12, HIGH); // Normally ON Only For Relay Module
  }
  void loop ()
  {
  digitalWrite (12, LOW); // RELAY ON
  delay (1000);
  digitalWrite (12, HIGH); // RELAY OFF
  delay (1000);
  }
```

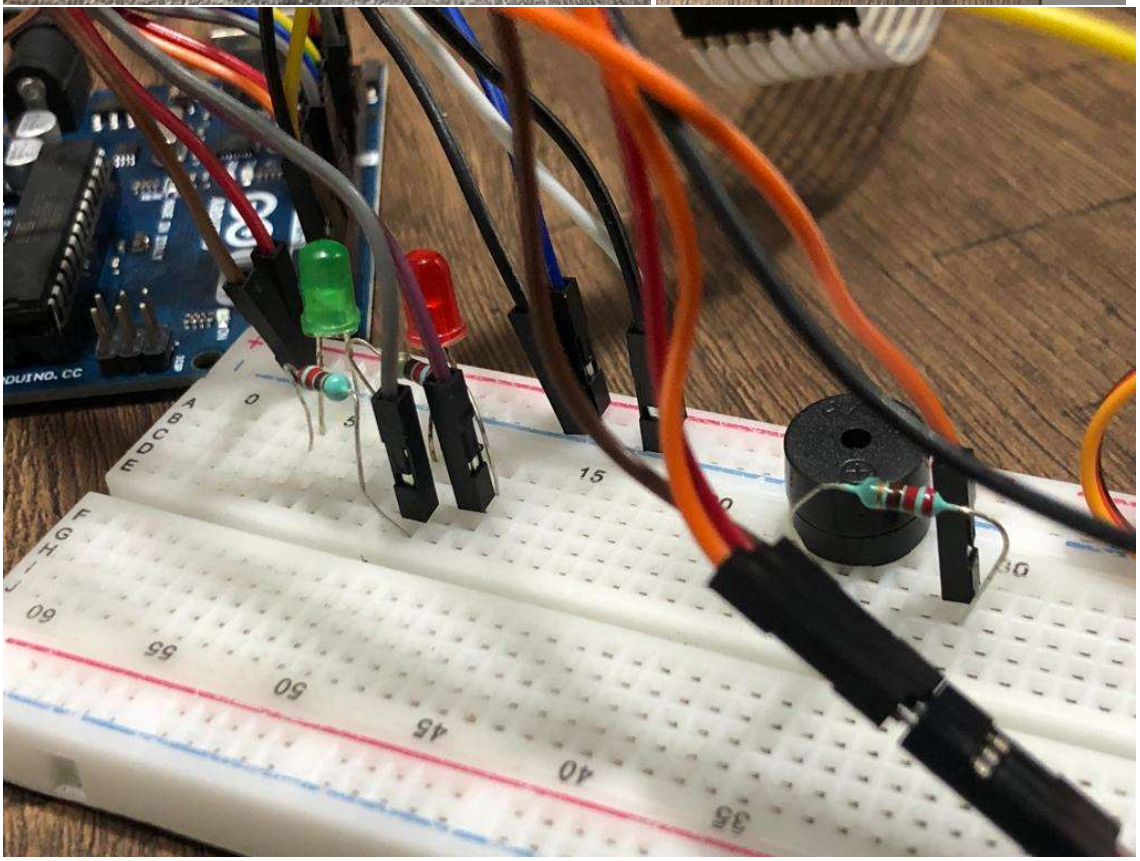
Actual hardware snapshots

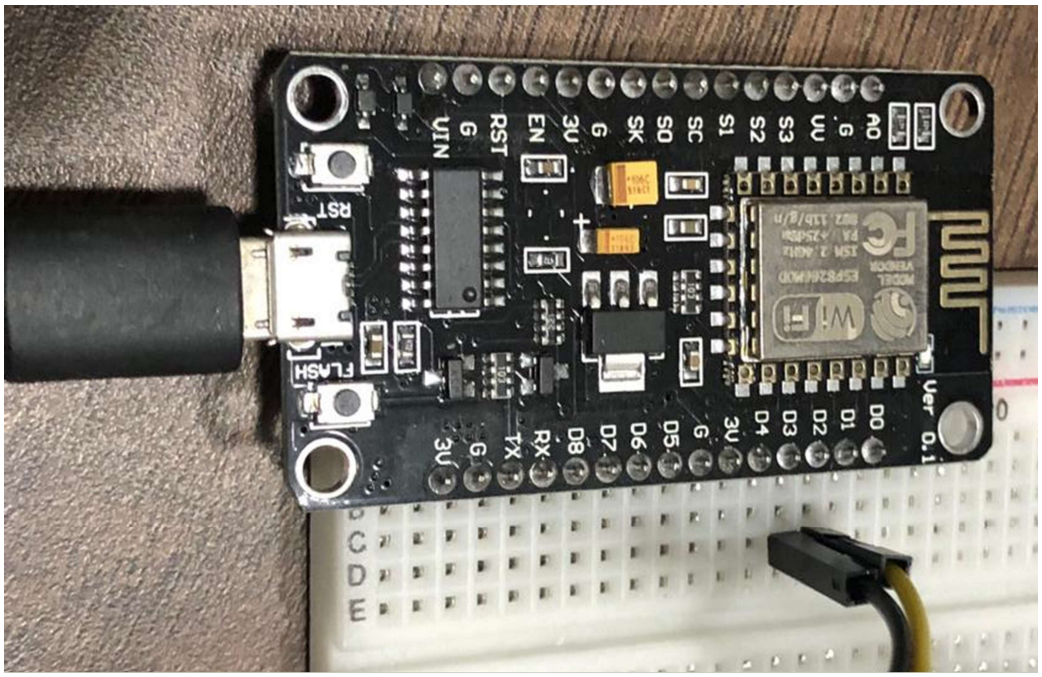
- Arduino Uno
- NodeMCU
- MFRC522
- 4x4 Matrix Keypad
- LEDs and Buzzer
- Servo Motor
- Relay Switch

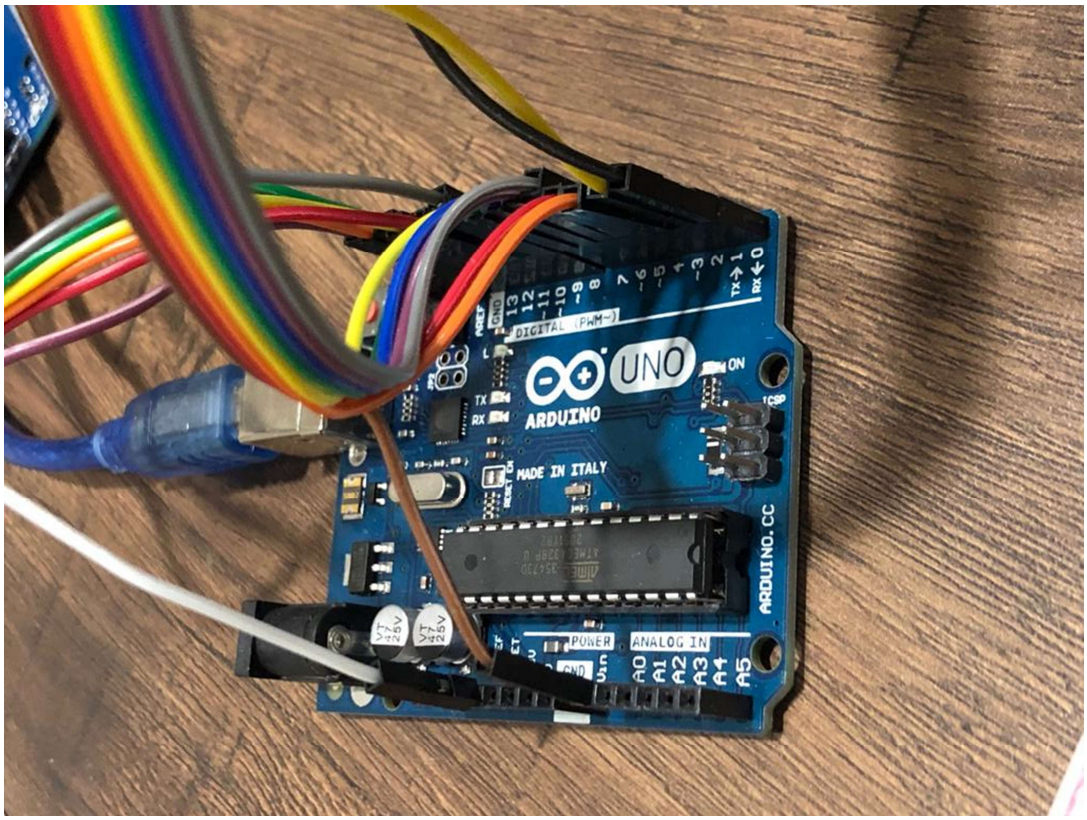
Physical Hardware


TOWER PRO™
MICRO SERVO
SG90

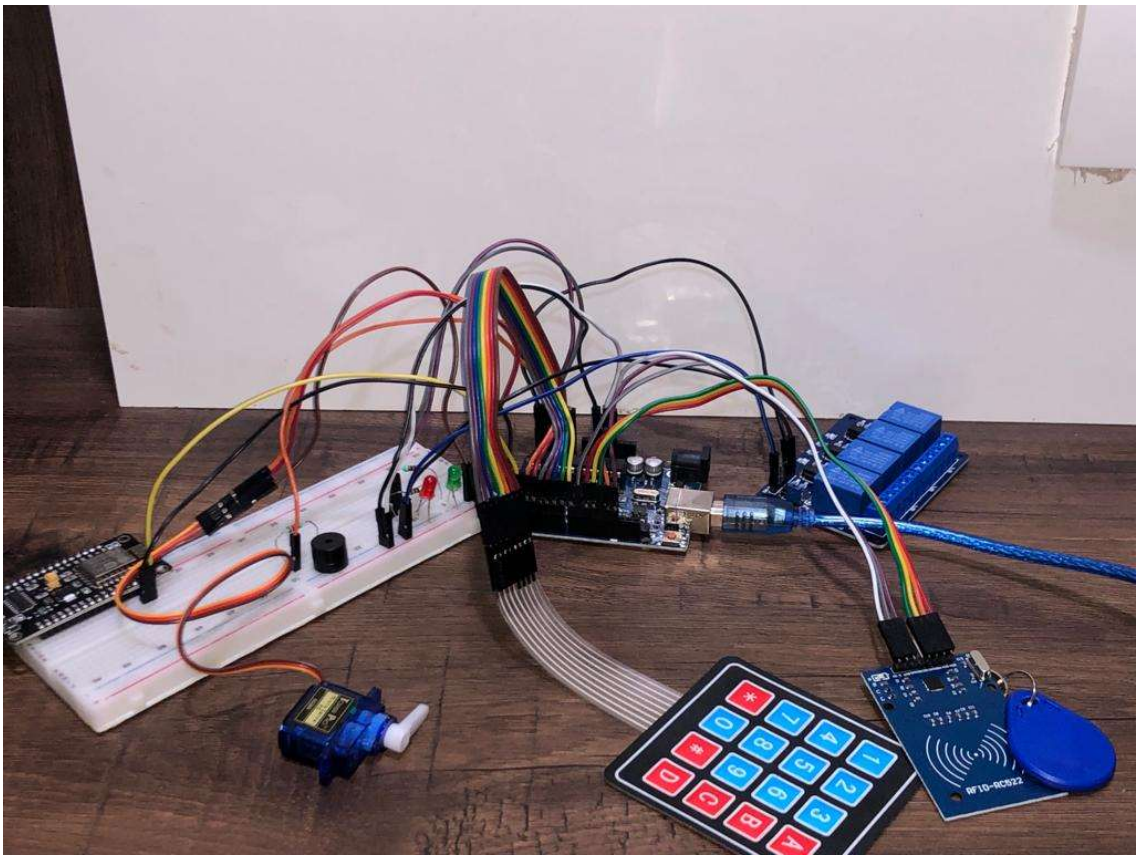





VIN
G
RST
EN
3V
G
SK
SO
SC
S1
S2
S3
VV
G
A0
RST
FLASH
3V
G
TX
RX
D8
D7
D6
D5
G
3V
D4
D3
D2
D1
D0
Ver 0.1
C
D
E


DIGITAL (PWM~)
UNO
ARDUINO
MADE IN ITALY
ON
TX
RX
POWER
ANALOG IN
ARDUINO.CC

Circuit connection snapshots



Pin Configuration

MFRC522	← -- →	Arduino Uno
3.3V	←→	3.3V
RST	←→	Digital pin 9
GND	←→	GND
IRQ	No connection	-
MSO	←→	Digital pin 12
MOSI	←→	Digital pin 11
SCK	←→	Digital pin 10
SDA	←→	Digital pin 9

4x4 matrix keypad	Arduino Uno
Rows	14, 3, 2, 8
Cols	7,6,5,4

NodeMCU	Arduino Uno
D1	Tx
D2	Rx

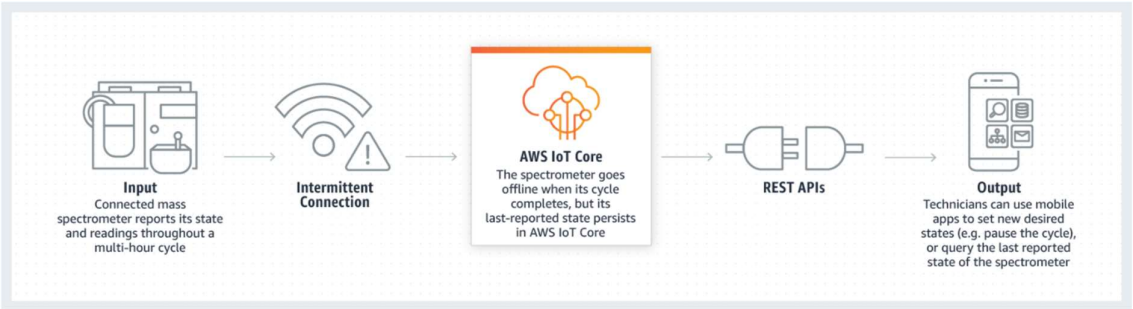
Other connections:

Arduino uno (A1)	Servo Motor
Arduino uno (A2)	Green LED
Arduino uno (A3)	Red LED
Arduino uno (A4)	Speaker/buzzer
Arduino uno (A5)	Relay Switch

Results:

1. The person with invalid RFID is rejected
2. The person with Valid RFID
    a. Was able to get entry when the entered Passcode is correct
        i. Green LED Turns on
    b. Wasn't able to get entry if the Passcode if not correct
        i. Red LED turns on
3. The Person have to validate RFID again if there are more than 3 attempts made for Passcode
    a. Emergency Situation
    b. Red LED blinking
    c. Buzzer Making Noise
4. Every data is sent to Cloud with a message and the UID scanned by RFID Scanner

Cloud Platform: AWS IoT Core



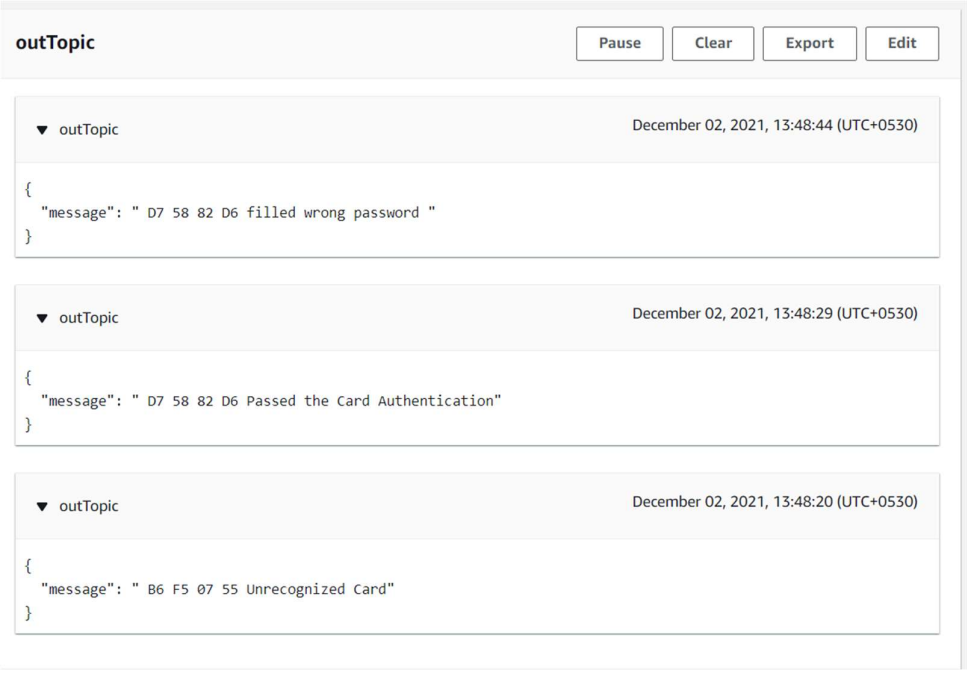
With Device Shadow, store the latest state of a connected device so it can be read or set at any time, making the device appear to your applications as if it were online all the time.

Observations:

1. Someone with trying to enter with Invalid Card

   Some one trying to enter with valid Card (Passed Authentication)

   But filled wrong password.



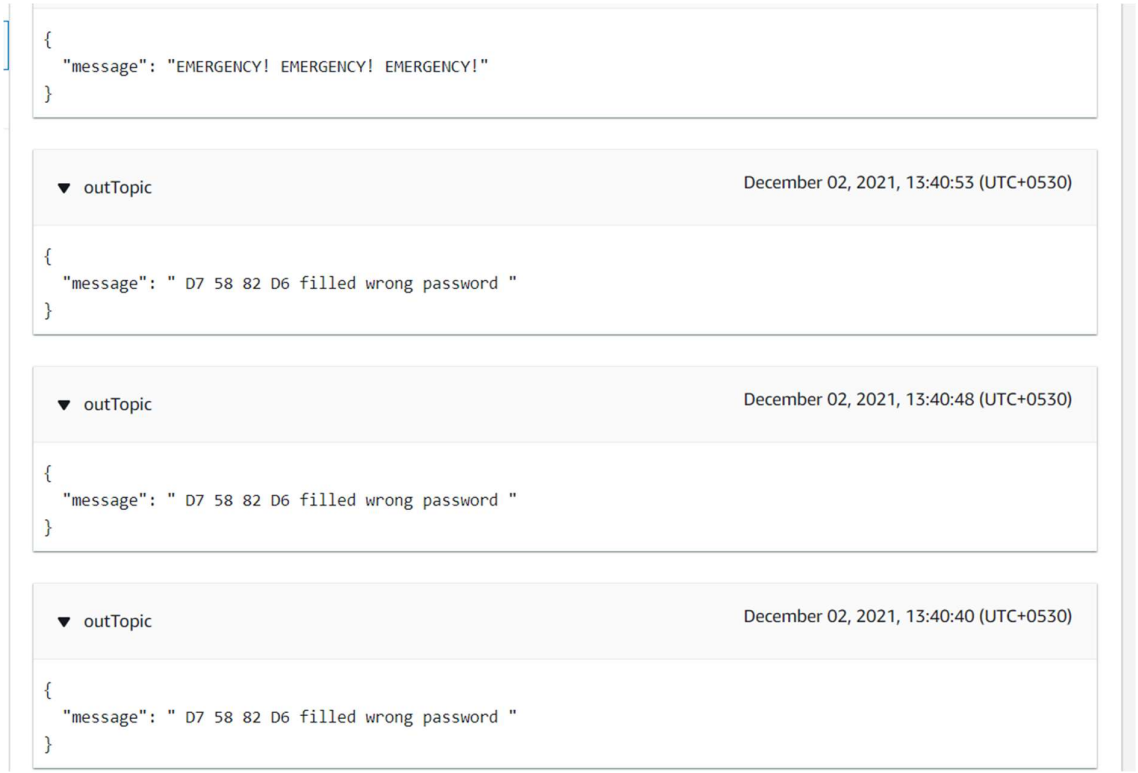
2. Someone with trying to enter with Invalid Card

Some one trying to enter with valid Card (Passed Authentication)

But filled wrong password for only once and

Filled correct passcode.

3. Someone tryiing to enter after filled wrong password 3 times , Emergency message publish



Analysis:

IoT core provides a JSON file of the MQTT messages tranfered , which can be used as an API for analysing and extracting information like

- Frequency of a person getting entry
- Frequency of Emergency situations
- Time of Emergency situations
- Frequency of incorrect passwords by each individual
- Usual/Unusual Entry Time of individuals

Or

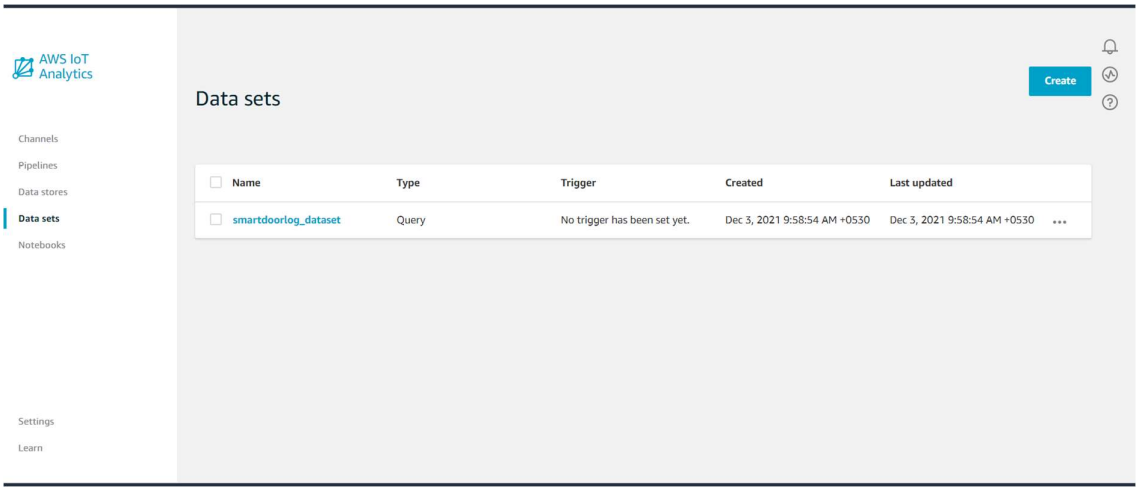
AWS IoT Analytics can also be used to analyse the data collected using MQTT services.

JSON File Created by AWS IoT Core

subscription (3).json

E: > subscription (3).json > [ ] messages > {} 3

```
{
  "topicFilter": "outTopic",
  "qos": 0,
  "messages": [
    {
      "format": "json",
      "topic": "outTopic",
      "timestamp": 1638433147563,
      "payload": {
        "message": "Approximate your card to the reader..."
      }
    },
    {
      "format": "json",
      "topic": "outTopic",
      "timestamp": 1638433145596,
      "payload": {
        "message": " D7 58 82 D6 got entry "
      }
    },
    {
      "format": "json",
      "topic": "outTopic",
      "timestamp": 1638433124973,
      "payload": {
        "message": " D7 58 82 D6 filled wrong password "
      }
    },
    {
      "format": "json",
      "topic": "outTopic",
      "timestamp": 1638433109949,
      "payload": {
        "message": " D7 58 82 D6 Passed the Card Authentication"
      }
    },
    {
      "format": "json",
```



Future Scope:

1. A camera/fingerprint module can be used to increase security
2. Data Generated by AWS IoT Core can be used as an API or can be used to create some specific Web Application or Mobile Application.
3. Other Modules like GPS, GPRS can be included which will extract the Location by out Mobile and unlock the door for us when we are at some specific distance.

Code for Arduino Uno:

```
#include <SPI.h>
#include <MFRC522.h>
#include <Keypad.h>
#include <Servo.h>
Servo myservo;

//green A2
// red A3

const byte n_rows = 4;
const byte n_cols = 4;
String content;
char keys[n_rows][n_cols] = {
   {'1', '2', '3', 'A'},
   {'4', '5', '6', 'B'},
   {'7', '8', '9', 'C'},
   {'*', '0', '#', 'D'}};

byte rowPins[n_rows] = {14, 3, 2, 8};
byte colPins[n_cols] = {7, 6, 5, 4};

Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, n_rows,
n_cols);
String password = "2345";
String mypassword;
int counter = 0;
int attempts = 0;

int max_attempts = 3;
int rfidpasses = 0;
int passcodepassed = 0;
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup()
{
  pinMode(A2,OUTPUT); // green
  pinMode(A3,OUTPUT); //red
```

```
  myservo.attach(A1);
  pinMode(A4, OUTPUT);
digitalWrite(A2,LOW);
digitalWrite(A3,LOW);
   rfidpasses = 0;
myservo.write(0);
    content = "";
    mypassword = "";
    Serial.begin(9600);
    SPI.begin();
    mfrc522.PCD_Init();
    Serial.print("Approximate your card to the reader...");


    for(int i=0;i<5;i++)
    {
      digitalWrite(A3,HIGH);
      digitalWrite(A2,LOW);
      delay(100);
      digitalWrite(A3,LOW);
      digitalWrite(A2,HIGH);
      delay(100);
    }
    digitalWrite(A2,LOW);
    digitalWrite(A3,LOW);

}

void loop()
{
// tone(A4, 2500, 19000);
// delay(10);
// tone(A4, 1500, 15000);
//digitalWrite(A4,LOW);
//digitalWrite(A3,HIGH);
//myservo.write(90);
//delay (500);
//myservo.write(0);
//delay (500);
```

```
  if (rfidpasses == 1)
  {
    func();
  }
  else
  {
    rfidpasses = validrfid();
  }
  delay(500);
}

int validrfid()
{

  if (!mfrc522.PICC_IsNewCardPresent())
  {
    return 0;
  }
  if (!mfrc522.PICC_ReadCardSerial())
  {

    return 0;
  }


  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
//      Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
//      Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }

  content.toUpperCase();
  if (content.substring(1) == "D7 58 82 D6")
  {
    Serial.print(content +" Passed the Card Authentication");
     digitalWrite(A2,HIGH);
```

```
      delay(1000);
      digitalWrite(A2,LOW);
      return 1;
   }
   else
   {

      Serial.print(content +" Unrecognized Card");
       content = "";
       digitalWrite(A3,HIGH);
      delay(1000);
      digitalWrite(A3,LOW);
      setup();
      return 0;
   }
}

void func()
{

   char key = myKeypad.getKey();
   if (key)
   {
     Serial.print(key);
      counter = counter + 1;
      if (attempts > 2)
      {
         Serial.print("EMERGENCY! EMERGENCY! EMERGENCY!");

         for(int i=0;i<20;i++)
         {
           digitalWrite(A3,HIGH);
            tone(A4, 2500, 19000);
            delay(100);
            tone(A4, 1500, 15000);
            digitalWrite(A3,LOW);
            delay(100);

         }
```

```
        attempts = 0;
//        delay(1000);
        setup();
        return;
      }
   }

   if (key == '1')
   {
      mypassword = mypassword + 1;
   }

   if (key == '2')
   {
      mypassword = mypassword + 2;
   }

   if (key == '3')
   {
      mypassword = mypassword + 3;
   }

   if (key == '4')
   {
      mypassword = mypassword + 4;
   }

   if (key == '5')
   {
      mypassword = mypassword + 5;
   }

   if (key == '6')
   {
      mypassword = mypassword + 6;
   }

   if (key == '7')
   {
      mypassword = mypassword + 7;
```

```
  }

  if (key == '8')
  {
    mypassword = mypassword + 8;
  }

  if (key == '9')
  {
    mypassword = mypassword + 9;
  }

  if (key == '0')
  {
    mypassword = mypassword + 0;
  }

  if (key == '*')

  {
//      Serial.println(mypassword);

    if (password == mypassword)

    {

      Serial.print(""+content + " got entry ");
      digitalWrite(A2,HIGH);

      myservo.write(90);
      delay(2000);
      myservo.write(0);


      content = "";
      mypassword = "";
      rfidpasses = 0;
      passcodepassed = 0;
      counter = 0;
      setup();
```

```
      }
      else
      {
         Serial.print( ""+content + " filled wrong password ");
         mypassword = "";
         digitalWrite(A3,HIGH);
         delay(100);
         digitalWrite(A3,LOW);
         attempts = attempts + 1;
         func();
      }
   }
}
```

Code for NodeMCU:

```
#include "FS.h"
#include "ESP8266WiFi.h "
#include "PubSubClient.h "
#include "NTPClient.h "
#include "WiFiUdp.h "
#include <SoftwareSerial.h>
SoftwareSerial mySerial(D1,D2);
//// Update these with values suitable for your network.
//
const char *ssid = "Garg";
const char *password = "1@333_55555";

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org");
//
```

```
const char *AWS_endpoint = "a1d7zgjv8pytti-ats.iot.ap-south-
1.amazonaws.com"; //MQTT broker ip

void callback(char *topic, byte *payload, unsigned int length)
{
   Serial.print("Message arrived [");
   Serial.print(topic);
   Serial.print("] ");
   for (int i = 0; i< length; i++)
   {
      Serial.print((char)payload[i]);
   }
   Serial.println();
}
WiFiClientSecure espClient;
PubSubClient client(AWS_endpoint, 8883, callback, espClient); //set MQTT
port number to 8883 as per //standard
long lastMsg = 0;
char msg[50];
int value = 0;

void setup_wifi()
{

   delay(10);
   // We start by connecting to a WiFi network
   espClient.setBufferSizes(512, 512);
   Serial.println();
   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");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  timeClient.begin();
  while (!timeClient.update())
  {
    timeClient.forceUpdate();
  }

  espClient.setX509Time(timeClient.getEpochTime());
}

void reconnect()
{
  // Loop until we're reconnected
  while (!client.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESPthing"))
    {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("ESP8266_Smart_door_log", "hello world");
      // ... and resubscribe
      client.subscribe("inTopic");
    }
    else
    {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      char buf[256];
      espClient.getLastSSLError(buf, 256);
      Serial.print("WiFiClientSecure SSL error: ");
      Serial.println(buf);

      // Wait 5 seconds before retrying
```

```
        delay(5000);
      }
   }
}

void setup()
{

   Serial.begin(115200);
   mySerial.begin(9600);
   Serial.setDebugOutput(true);
   // initialize digital pin LED_BUILTIN as an output.
   pinMode(LED_BUILTIN, OUTPUT);
   setup_wifi();
   delay(1000);
   if (!SPIFFS.begin())
   {
      Serial.println("Failed to mount file system");
      return;
   }

   Serial.print("Heap: ");
   Serial.println(ESP.getFreeHeap());

   // Load certificate file
   File cert = SPIFFS.open("/cert.der", "r"); //replace cert.crt eith your
uploaded file name
   if (!cert)
   {
      Serial.println("Failed to open cert file");
   }
   else
      Serial.println("Success to open cert file");

   delay(1000);

   if (espClient.loadCertificate(cert))
      Serial.println("cert loaded");
   else
      Serial.println("cert not loaded");
```

```
  // Load private key file
  File private_key = SPIFFS.open("/private.der", "r"); //replace private eith
your uploaded file name
  if (!private_key)
  {
    Serial.println("Failed to open private cert file");
  }
  else
    Serial.println("Success to open private cert file");

  delay(1000);

  if (espClient.loadPrivateKey(private_key))
    Serial.println("private key loaded");
  else
    Serial.println("private key not loaded");

  // Load CA file
  File ca = SPIFFS.open("/ca.der", "r"); //replace ca eith your uploaded file
name
  if (!ca)
  {
    Serial.println("Failed to open ca ");
  }
  else
    Serial.println("Success to open ca");

  delay(1000);

  if (espClient.loadCACert(ca))
    Serial.println("ca loaded");
  else
    Serial.println("ca failed");

  Serial.print("Heap: ");
  Serial.println(ESP.getFreeHeap());
}

void loop()
```

```
{

 String msg1 = mySerial.readStringUntil('\r');
   Serial.println(msg1);
   if (!client.connected()) {
     reconnect();
     }
client.loop();



Serial.print("Publish message: ");

snprintf (msg, 75, "{\"message\": \"%s\"}", msg1.c_str());
Serial.println(msg1);
if(msg1!="")
{
client.publish("outTopic", msg);
}

Serial.print("Heap: ");
Serial.println(ESP.getFreeHeap()); //Low heap can cause problems
// wait for a second

digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage
level)
delay(100); // wait for a second
digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage
LOW
delay(100); // wait for a second
}
```

References:

1. https://components101.com/articles/introduction-rfid-modules-construction-types-and-working
2. https://learn.parallax.com/tutorials/language/propeller-c/propeller-c-simple-devices/read-4x4-matrix-keypad
3. https://www.toppr.com/bytes/principles-of-led/#:~:text=Working%20Principle%3A,in%20the%20form%20of%20photons.
4. https://www.electronicshub.org/what-is-relay-and-how-it-works/
5. https://circuitdigest.com/microcontroller-projects/arduino-relay-control
6. https://www.takigen.com/products/list/14020
7. https://techatronic.com/solenoid-lock-interfacing-with-arduino/
8. https://www.instructables.com/How-to-Interface-RFID-to-Arduino/
9. https://store.arduino.cc/products/arduino-uno-rev3/
10. https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet
11. https://www.microchip.com/en-us/product/ATmega328P?gclid=Cj0KCQiA-qGNBhD3ARIsAO_o7ykK23-Xu7Cxg2fhtTXgPqKHmxekIPB4IclPvtQk3D8d2SapylWoXkcaAiqOEALw_wcB
12. https://aws.amazon.com/iot-core/
13. https://www.youtube.com/watch?v=28FS2qix2u4
14. https://www.youtube.com/watch?v=I8zROK4S2bE
15. https://www.youtube.com/watch?v=vMwlnm8GvIA&t=782s