THE HOME SECURITY SYSTEM

A Project Report

Submitted by

BARANITHARAN.B	1912104
CIBI.P	1912106
NIYASKHAN.M	1912129

Guided BY:

Dr.A.CHARLES M.E.,Ph.D

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

GOVERNMENT COLLEGE OF ENGINEERING-
BARGUR(AUTONOMOUS)
KRISHNAGIRI-635 104, TAMILNADU
ANNA UNIVERSITY: CHENNAI – 600 025

BONAFIDE CERTIFICATE

Certificated that this project work-I **"THE HOME SECURITY SYSTEM"** is the bonafide work done by the following students under my supervision

BARANITHARAN.B	1912104
CIBI.P	1912106
NIYASKHAN.M	1912129

SIGNATURE

Dr.A.CHARLES M.E.,Ph.D.,(AP/ECE)
Supervisor,
Department of ECE,
Government college of Engineering,
Bargur-635 104

SIGNATURE

Dr.M.KAVITHA M.E.,Ph.D.,
Head of the Department,
Department of ECE,
Government college of Engineering,
Bargur-635 104

Submitted for project work-I viva-voce examination held on …..…..…… at Government College of Engineering, Bargur-635104.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We feel glad to take this opportunity to cordially acknowledge a number of people who provided us a great support during our project.

We would like to express our deep sense of gratitude to our respected Principle Dr.M.NATARAJ, M.E., Ph.D., who has bestowed his kind grace and affection of us in accomplishing this project.

Sincere thanks to Dr.M.KAVITHA M.E., Ph.D., Head of the Department of Electronics and Communication Engineering, Bargur on providing facilities to do the project work and utilize all the facilities in this college.

We would like to thanks our project Coordinator, Prof. R.RANJITH M.E., Ph. D ., Assistant Professor , Department of Electronics and Communication Engineering, who guided us throughout the entire phase of our projects with his esteemed presences . It is his motivation and guidance which made us explore our project.

It is our opportunity to express gratitude and sincere thanks to our guide Dr.A.CHARLES M.E.,Ph.D.,(AP/ECE) Supervisor who gave us the motivation to do our project. We are also thankful to all the faculty members and laboratory assistants for our department for their constructive guidance and encouragement.

Finally, we thank our parents, teaching, non-teaching staff and friends who helped us in completing this project successfully.

TABLE OF CONTENTS

LIST OF FIGURES:

LIST OF TABLES:

ABSTRACT

Aim of the Project is to give the consumers, a 2nd layer of the security system for their home to safeguard their valuable things. This project provides us a two-factor authentication for our Home. In this project we use the ESP8266 microcontroller to get the signal from phone through Wi-Fi and based on that instruction microcontroller decide to give instruction to the Servo Motor. The Servo motor is connected to the door lock. To Open the door, user need to install the app that we designed using MIT app inventor, then they need to connect to internet. Then they need to verify the fingerprint or Enter the password in the app, if the password entered is correct the app will send open() signal to the Microcontroller via Wi-Fi then ESP8266 give open signal to the servo motor to open the door. The door opened or closed information is got by the button mechanism and the information is stored in MIT app inventor CLOUDDB with the time. Vibration Sensor connected to ESP8266 to sense, is someone break the door and send the information to app. User can able to get when door opened information from cloudDB in app.

CHAPTER 1

1.1 INTRODUCTION

Home is the place where we live, kept our valuable things, it has our loved ones and it is the part of us in our daily life. But the security for the home is less now a days. We need a security system for our Home to safe our things. To avoid this, we are introducing a project called Home Security System

This Home Security System is compact box which will be attached to the door and connected to the door lock. If the someone wants to open the door lock to get into the home, they first need to verify their identity by scanning their finger on their phone under the app that we created. Suppose if the phone does not have the finger print sensor, then they need to enter the password to open the door lock. If the identity is matched then the app will send open signal with RFID to the home security system. Then the system checks the RFID and it will open the door lock. This acts as a two-factor authentication for our home.

1.2 PROBLEM OF STATEMENT

Security is the main thing to safeguard our things. In today scenario thief's and theft are increased due to the loss of security. Due to that loss of security some people losses their money, jewellery, and more. So, we need a system which authenticate us and verify our identity to open the door and we need to know when someone hitting or destroying our door. When this happens, we need a system to tell us that some one hitting or destroying the door. And also, we need to make that system at affordable price to reach the all people. To solve this all problem, we come up with the solution called home security system

1.3 AIM OF THE PROJECT

Aim of the Project is to give the consumers, a 2^{nd} layer of the security system for their home to safeguard their valuable things. This project provides us a two-factor authentication for our Home.

1.4 MOTIVATION

Security is necessary in Home. Our goal is to give a dual layer security to house at affordable price. We get inspired on phone's two-factor authentication system. We think why don’t we do project regarding providing two-factor authentication

n on house door. Two-factor authentication means to get access to something, we need to verify our identity in case of phone we get verified either via Phone number or email ID. But here in this project we verify the Reference ID which is the unique ID that will be programmed in Microcontroller and finger print.

First, we have the idea to buy the Fingerprint sensor, then we look at the Price of fingerprint sensor which Rs.1000+. Then we noticed that, most of the people have the phone with fingerprint sensor, then we decide Why don’t we design a app to access the fingerprint sensor in phone and send the signal to microcontroller to open door from the app. This idea will minimize the cost of our product in market. Using this technique, we can safeguard our sweet home

CHAPTER 2

HOME SECURITY SYSTEM

2.1 EXPLANATION

Here you can see in figure 2.1 is a block diagram that NodeMCU is connected to Vibration sensor, Wi-Fi module, Servo motor, Switch and power source. For first time use, user need to register and create password for their product with the unique reference ID allotted to that product. The created password and reference ID will get stored on CloudDB. To open the door, user should turn on the data transfer and click scan button on the app that we created using the MIT app inventor and they need to scan their fingerprint. If the fingerprint is matched it will send the Open() signal along with the reference id to NodeMCU. Then the Arduino checks if the Reference Id matches or not. If it matches then it will open the door lock using the servo motor. Then it wait for if the button is become '0' if it became '0' NodeMCU send door_opened() function to app, then the app will save that information with time in CloudDB. If button become '1' NodeMCU send door_closed() signal to app this also will get saved in cloudDB. We can get this info later whenever we want. Mean While if the strong vibration is produced in the door. NodeMCU decide it as someone destroying the door and send Alert() to the app.

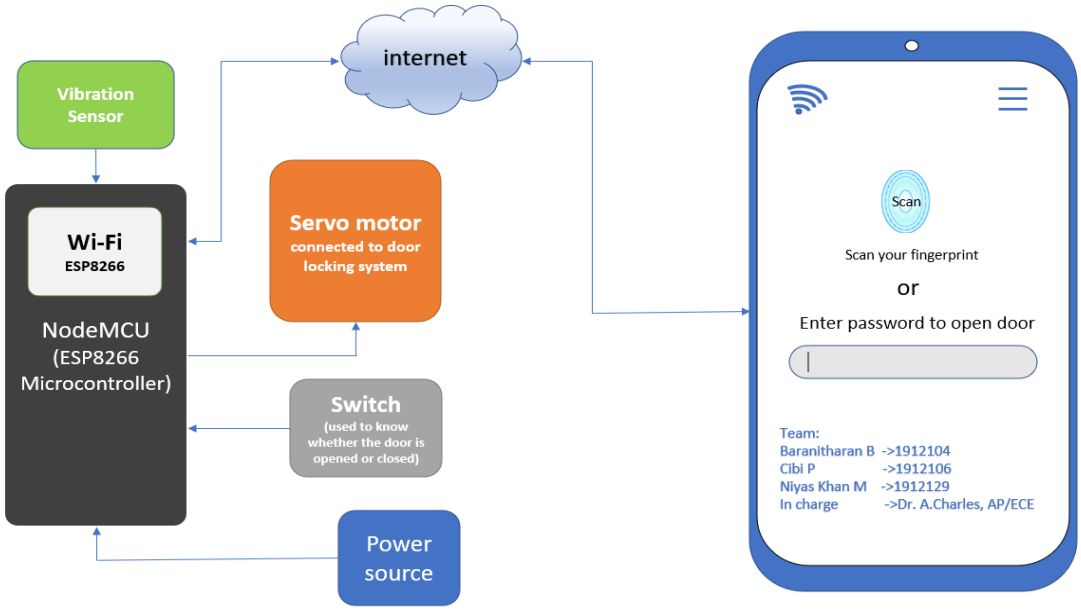


Figure 2.1-Block diagram

Here you can see in figure 2.3 is the circuit diagram used for our Home Security project. Here NodeMCU is the common component which connected to the all-other components. The circuit diagram is quite simple. Here we use the 3digital pin D2, D4andD5. D2 digital pin is connected to servo motor to open the door lock. Digital pin D5 is connected to Switch like mechanism attached to the door as shown in the image, to keep track the status of the door. Using this microcontroller knows whether the door is opened or closed. It will Be 1 if the door is closed and it will be 0 If the door is opened. Digital pin D4 is connected to the Vibration sensor Sw-420 as shown in the image. From which the vibration occurred at the door will be detected. 5v and ground from the Nodemcu are connected to the vibration sensor and servomotor as shown in the figure. 5v is connected to the switch as shown in the figure

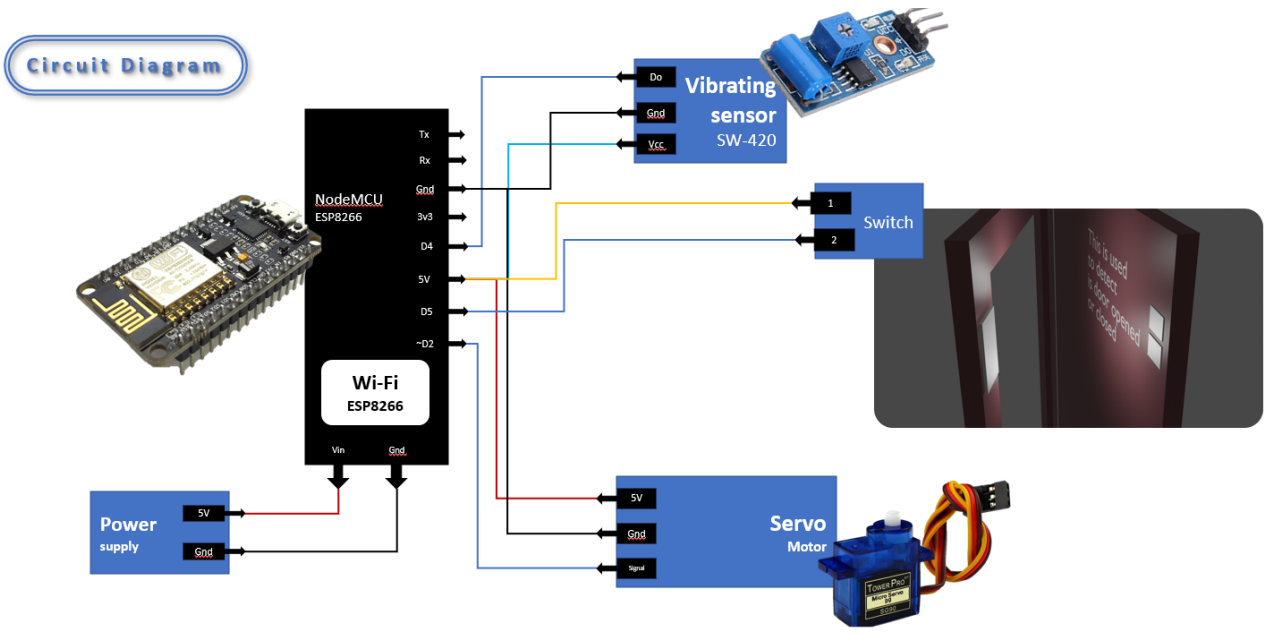


Figure 2.2-Circuit Diagram

2.2 COMPONENTS

S.NO	COMPONENTS	DESCRIPTION
2.2.1	ESP8266 Microcontroller (NodeMCU) + Wi-Fi Module	It is a microcontroller (decision maker in project) and it has WIFI to connect to internet
2.2.2	Switch	To get status of the door
2.2.3	Servo Motor	To open the door lock
2.2.4	Door lock	To lock the door
2.2.5	Vibrating Sensor SW-420	To get the vibration status occurring at the door
2.2.6	ThingSpeak.com	To make the connect between the project with app over the internet
2.2.7	MIT App Inventor	To create the app
2.2.8	Power Source	To power the project

Table 2.1-Components List

2.2.1 NODEMCU (ESP8266 DEVELOPMENT BOARD)

NodeMCU is an open-source Lua based firmware and **development board** specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

Pin Category	**Name**	**Description**
Power	Micro-USB, 3.3V, GND, Vin	Micro-USB: NodeMCU can be powered through the USB port 3.3V: Regulated 3.3V can be supplied to this pin to power the board GND: Ground pins Vin: External Power Supply
Control Pins	EN, RST	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.

UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

Table 2.2-NodeMCU Pin Configuration

NodeMCU ESP8266 Specifications & Features

a. Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
b. Operating Voltage: 3.3V
c. Input Voltage: 7-12V
d. Digital I/O Pins (DIO): 16
e. Analog Input Pins (ADC): 1
f. UARTs: 1
g. SPIs: 1
h. I2Cs: 1
i. Flash Memory: 4 MB
j. SRAM: 64 KB
k. Clock Speed: 80 MHz
l. USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
m. PCB Antenna
n. Small Sized module to fit smartly inside your IoT projects

Brief About NodeMCU ESP8266

The NodeMCU ESP8266 development board comes with the ESP-12E module containing the ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

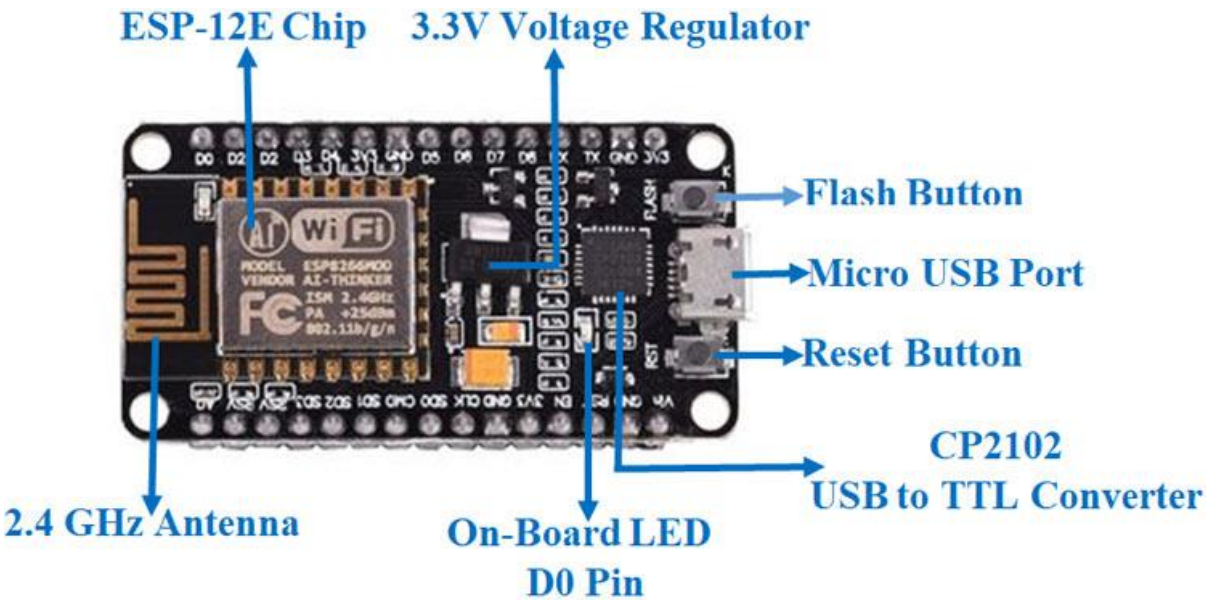


Figure 2.3-NodeMCU Layout

Programming NodeMCU ESP8266 with Arduino IDE

The NodeMCU Development Board can be easily programmed with Arduino IDE since it is easy to use.

Programming NodeMCU with the Arduino IDE will hardly take 5-10 minutes. All you need is the Arduino IDE, a USB cable and the NodeMCU board itself.

Uploading your first program

Once Arduino IDE is installed on the computer, connect the board with the computer using the USB cable. Now open the Arduino IDE and choose the correct board by selecting Tools>Boards>NodeMCU1.0 (ESP-12E Module), and choose the correct Port by selecting Tools>Port. To get it started with the NodeMCU board and blink the built-in LED, load the example code by selecting Files>Examples>Basics>Blink. Once the example code is loaded into your IDE, click on the 'upload' button given on the top bar. Once the upload is finished, you should see the built-in LED of the board blinking.

Applications

i. Prototyping of IoT devices
ii. Low power battery operated applications
iii. Network projects
iv. Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities

2.2.2 SWITCH

It is the switch like mechanism attached to the door to get the status of the door whether it is opened or not. The switch will be HIGH if the door is closed. And it will be LOW if the door is opened.

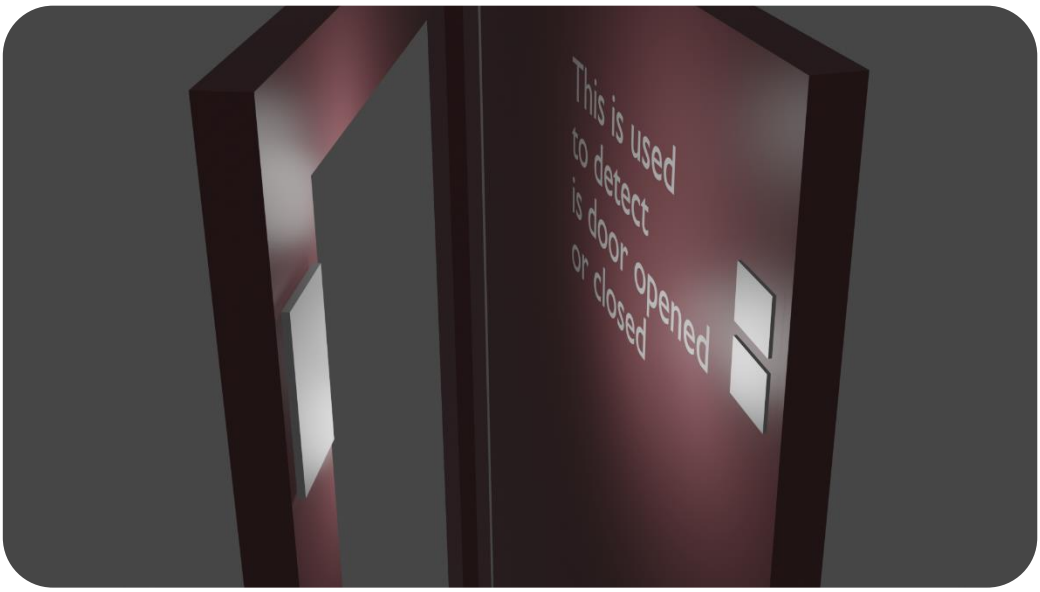


Figure 2.4 Door with Switch like Mechanism

2.2.3 SERVO MOTOR

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor, although the term *servomotor* is often used to refer to a motor suitable for use in a closed-loop control system.

Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing

Figure 2.5 Servo Motor

MECHANISM

A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.

The motor is paired with some type of position encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in

either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

The very simplest servomotors use position-only sensing via a potentiometer and bang-bang control of their motor; the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos used for radio-controlled models.

More sophisticated servomotors use optical rotary encoders to measure the speed of the output shaft and a variable-speed drive to control the motor speed. Both of these enhancements, usually in combination with a PID control algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less overshooting.

2.2..4 DOOR LOCK

Door lock is used to lock the door by the use of the servo motor.

Figure 2.6-Door Lock with Servo Motor

2.2.5 VIBRATION SENSOR

The vibration sensor module based on the vibration sensor SW-420 and Comparator LM393 is used to detect vibrations. The threshold can adjust using an on-board potentiometer. During no vibration, the sensor provides Logic Low and when the vibration is detected, the sensor provides Logic High.

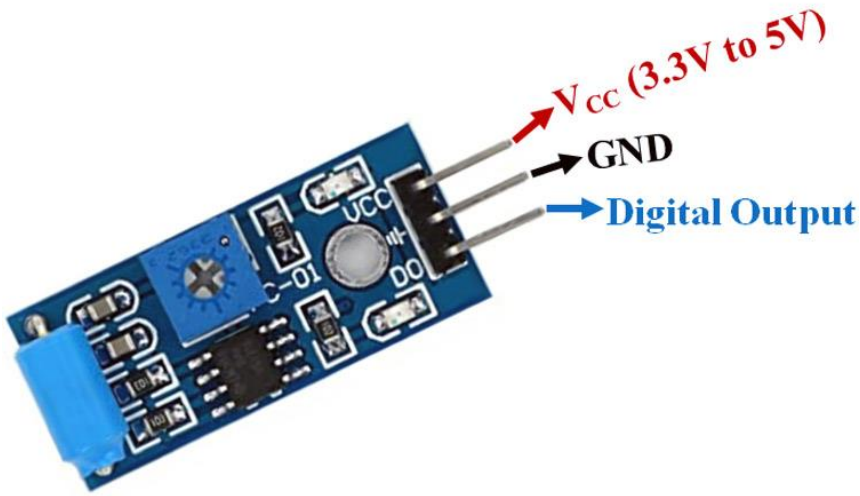


Figure 2.7-Vibration Sensor pin Configuration

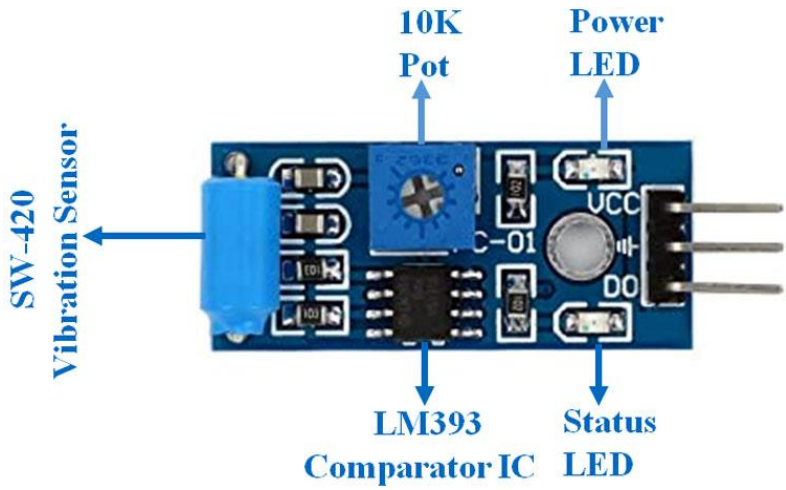


Figure 2.8-Vibration Sensor

VCC	The Vcc pin powers the module, typically with +5V
GND	Power Supply Ground
DO	Digital Out Pin for Digital Output.

Vibration Sensor Module Features & Specifications

i. Operating Voltage: 3.3V to 5V DC
ii. Operating Current: 15mA
iii. Using SW-420 normally closed type vibration sensor
iv. LEDs indicating output and power
v. LM393 based design
vi. Easy to use with Microcontrollers or even with normal Digital/Analog IC
vii. With bolt holes for easy installation
viii. Small, cheap and easily available

Brief about SW-420 Vibration Sensor/Motion Sensor Module

This Vibration Sensor Module consists of an SW-420 Vibration Sensor, resistors, capacitor, potentiometer, comparator LM393 IC, Power, and status LED in an integrated circuit. It is useful for a variety of shocks triggering, theft alarm, smart car, an earthquake alarm, motorcycle alarm, etc.

LM393 IC

LM393 Comparator IC is used as a voltage comparator in this vibration sensor module. Pin 2 of LM393 is connected to Preset (10KΩ Pot) while pin 3 is connected to vibration sensor. The comparator IC will compare the threshold voltage set using the preset (pin2) and the Vibration Sensor pin (pin3).

Preset (Trimmer pot)

Using the onboard preset, you can adjust the threshold (sensitivity) of the digital output.

SW-420 Vibration Switch

Vibration switch recognizes the amplitude of the vibration to which it is exposed. The switch response can be electrical contact closure or contact opening. The electrical contact may be either an electromechanical relay or a solid-state device.

How to Use SW-420 Vibration Sensor Module

Vibration sensor module consists of three pins i.e. VCC, GND, and DO. The Digital out pin is connected to the output pin of the LM393 comparator IC. The Internal Circuit diagram of the Temperature sensor module is given below.

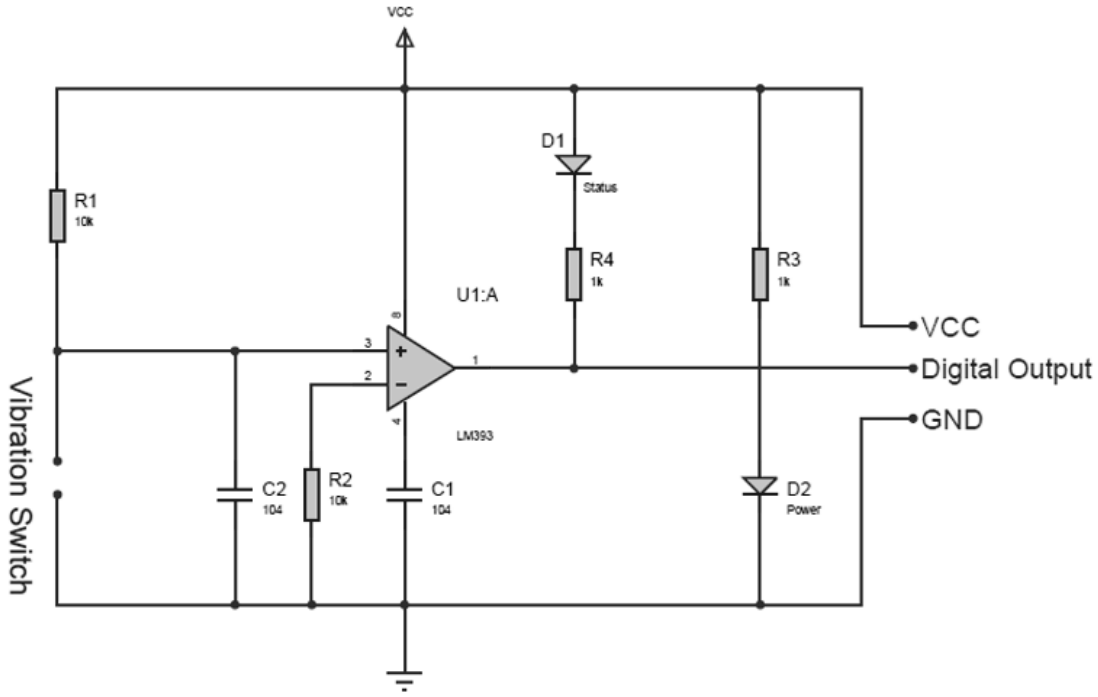


Figure 2.9-Vibration Sensor circuit

Using the Vibration sensor module with the microcontroller is very easy. Connect the Digital Output pin of the module to the Digital pin of Microcontroller. Connect VCC and GND pins to 5V and GND pins of Microcontroller.

Applications of Vibration Sensor Module

- Shocks triggering
- Theft alarm
- Smart car
- Earthquake alarm
- Motorcycle alarm

2.2.6 THINKSPEAK.COM

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.

ThingSpeak Features

1) Collect data in private channels
2) Share data with public channels
3) RESTful and MQTT APIs
4) MATLAB® analytics and visualizations
5) Event scheduling
6) Alerts
7) App integrations

Works With

1) MATLAB® & Simulink®
2) Arduino®
3) Particle devices
4) ESP8266 and ESP32 WIFI Modules
5) Raspberry Pi™
6) LoRaWAN®
7) Things Network

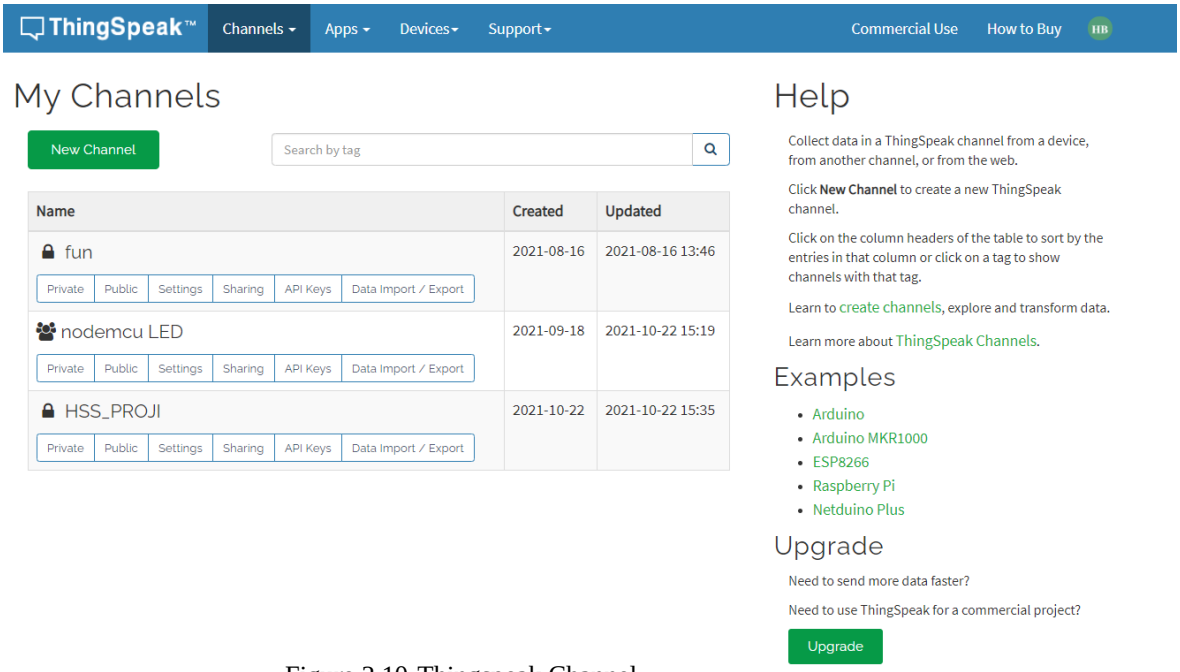


Figure 2.10-Thingspeak Channel

The above image shows the HSS project channel created in thingspeak.com

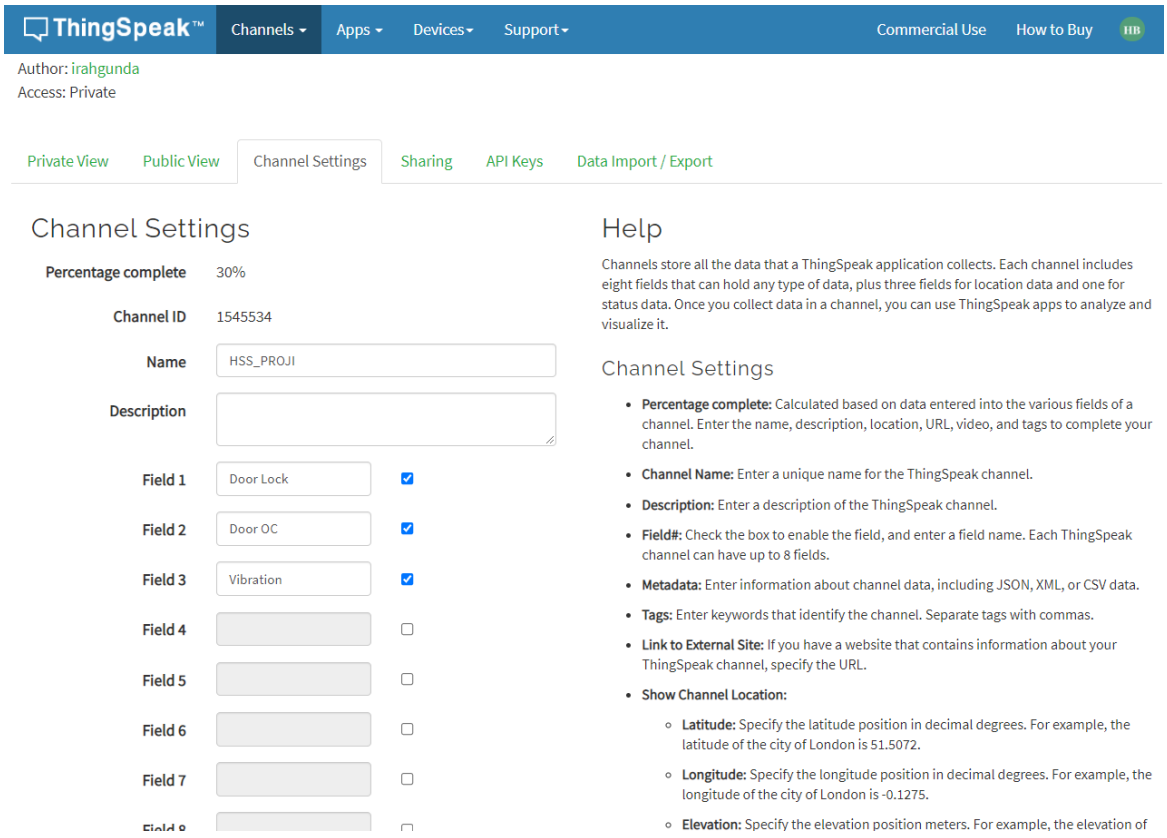


Figure 2.11-Thingspeak channel setting

The above image show channel settings used for our project in thingspeak.com

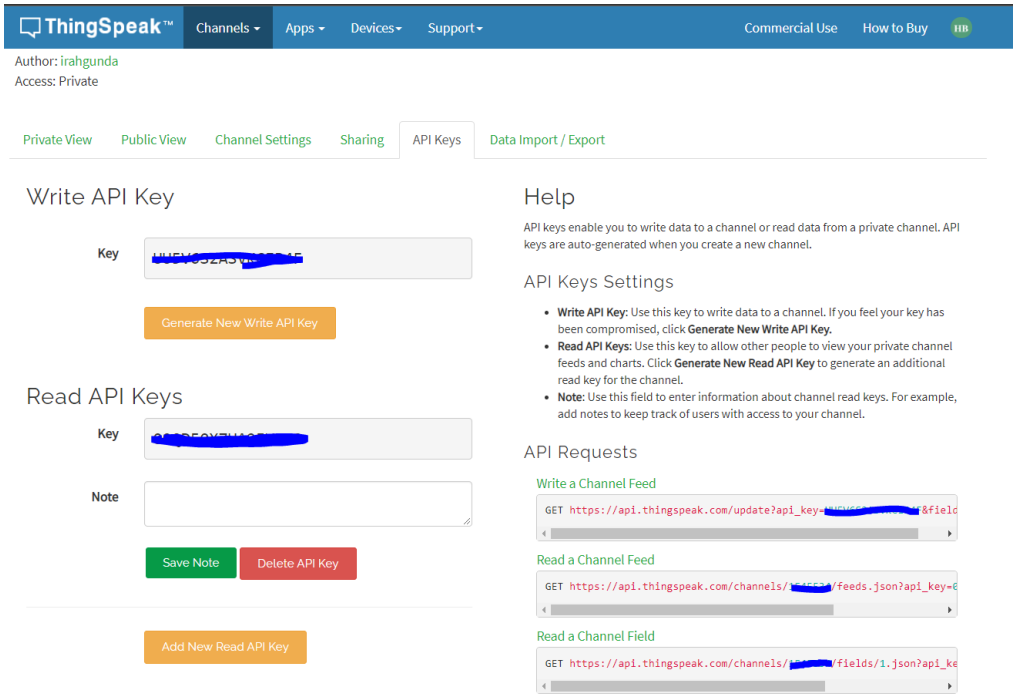


Figure 2.12-Thingspeak API key

The above image shows the API key (API keys are blured due to security reason)

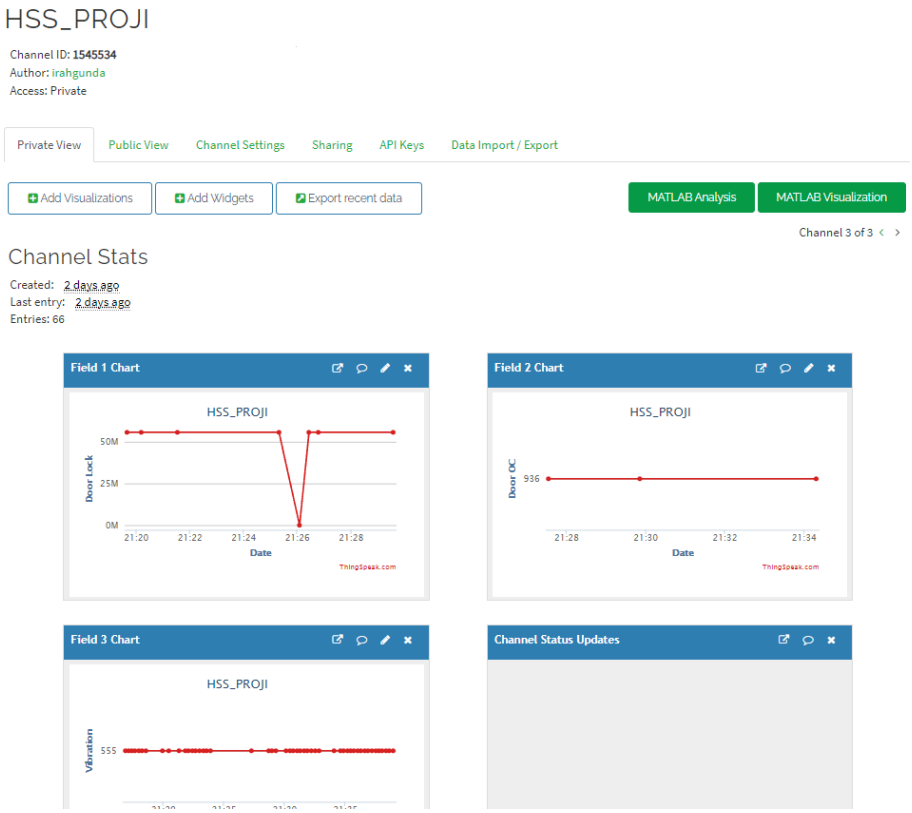


Figure 2.13 Thingspeak Channel Stats

The above image shows the graphical representation of the field value in thingspeak.com

2.2.7 MIT app Inventor

MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It allows newcomers to computer programming to create application software(apps) for two operating systems (OS): Android, and iOS, which, as of 8 July 2019, is in final beta testing. It is free and open-source software released under dual licensing: a Creative Commons Attribution ShareAlike 3.0 Unported license, and an Apache License 2.0 for the source code.

It uses a graphical user interface (GUI) very similar to the programming languages Scratch (programming language) and the StarLogo, which allows users to drag and drop visual objects to create an application that can run on Android devices, while a App-Inventor Companion (The program that allows the app to run and debug on) that works on iOS running devices are still under development. In creating App Inventor, Google drew upon significant prior research in educational computing, and work done within Google on online development environments.[1]

App Inventor and the other projects are based on and informed by constructionist learning theories, which emphasize that programming can be a vehicle for engaging powerful ideas through active learning. As such, it is part of an ongoing movement in computers and education that began with the work of Seymour Papert and the MIT Logo Group in the 1960s, and has also manifested itself with Mitchel Resnick's work on Lego Mindstorms and StarLogo.[1][2]

App Inventor also supports the use of cloud data via an experimental Firebase#Firebase Realtime Database component.

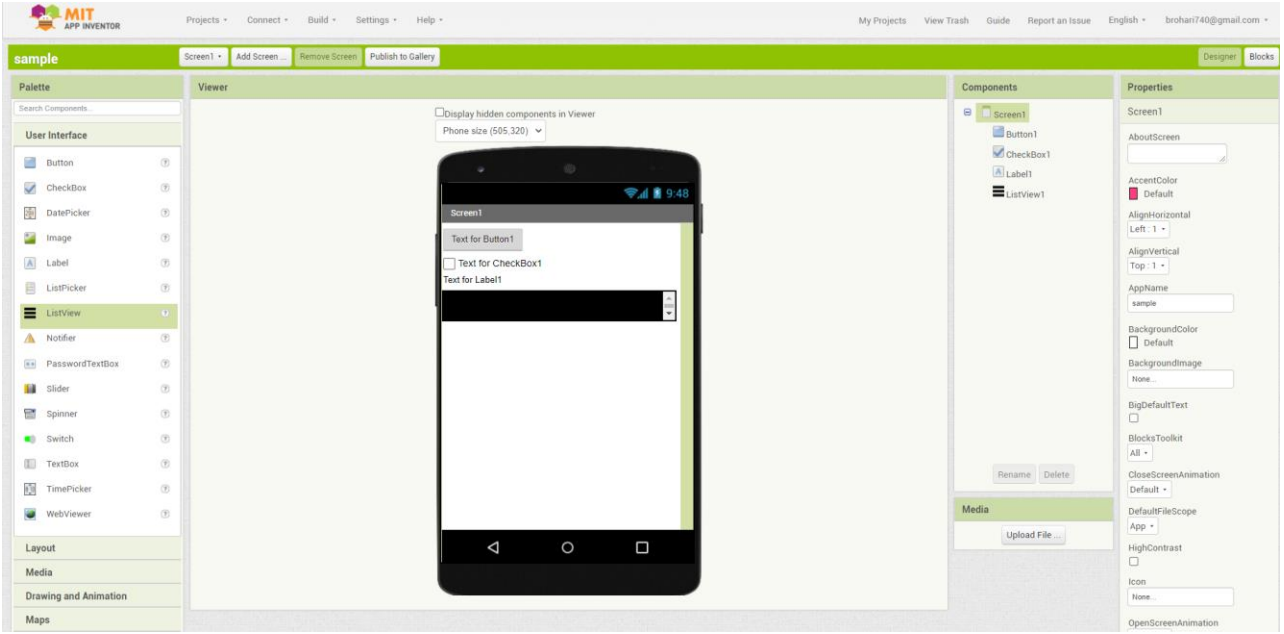


Figure 2.14-MIT App Inventor Design Page

The above image show the MIT APP INVENTOR work space here where the UI is designed for app.

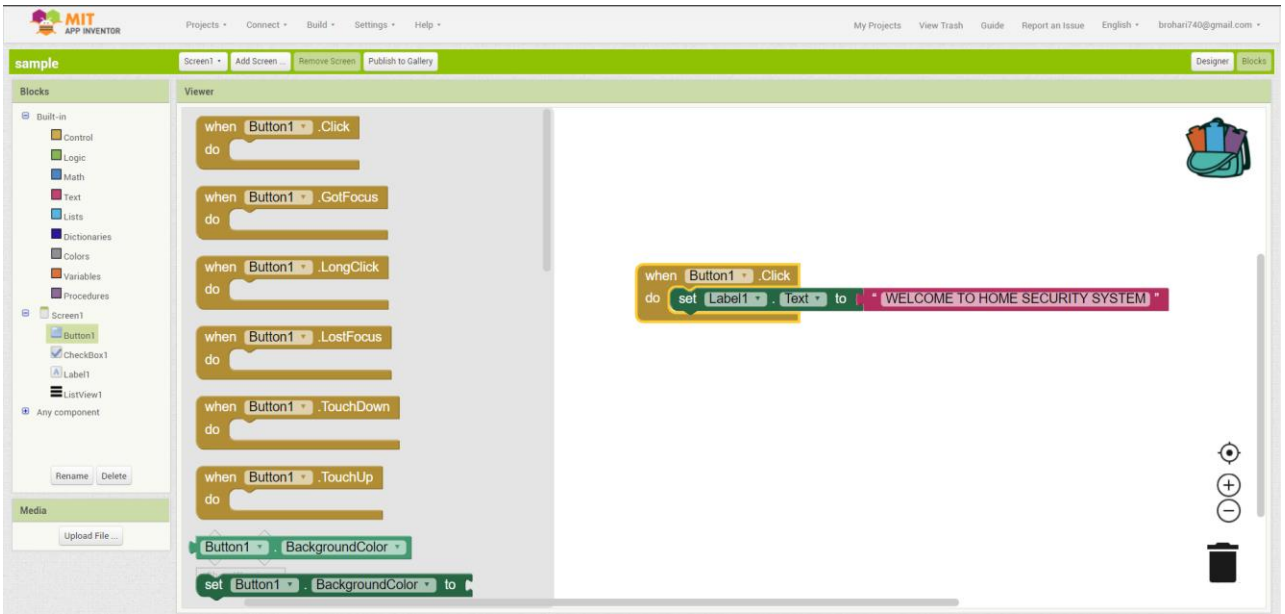


Figure 2.14-MIT App Inventor Block coding

The above image shows the block coding page of the MIT APP INVENTOR. Here where codes can be added to make the working app.

2.3 APPLICATION

1) It is used to secure our home from thief.
2) It provide us the second layer of the security.
3) If someone try to destroy the door, the system will send vibration detected message to the app so the user can the necessary action wherever the user in the internet.
4) From the app we know that the door is closed or open.
5) It can also be used in lockers.

2.4 LIMITIATIONS

1) It is hard to change the ssid and password which is programed inside the NodeMCU.

2) We need to maintain the API key Safe, using the API key other person can able to open the door lock.

3) To receive those notification the app needs to be open on the android phone.

CHAPTER 3

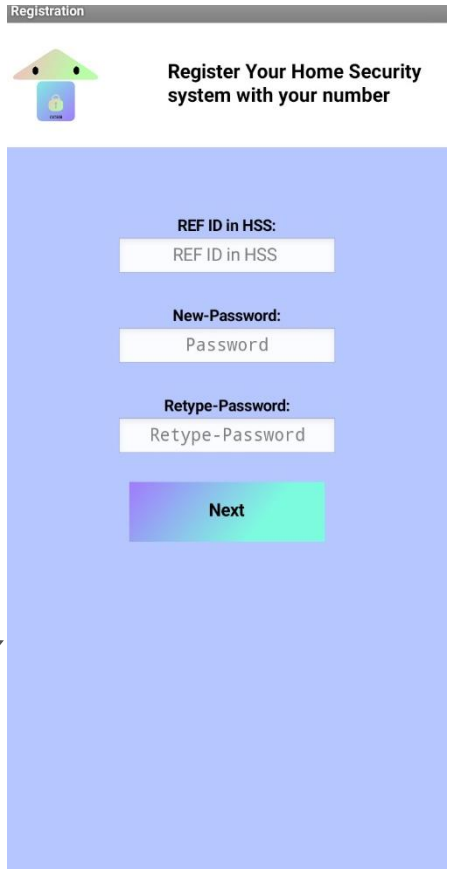
APP DESIGN IN MIT APP INVENTOR

To develop the android app, we use the MIT app inventor. Below listed images are the screen created for our Home Security System project.

It is the Welcome Page With give the Register of Login Option to user. User can Perform the Respective action by Pressing the Respective Button

Welcome Page

Figure 3.1-App Welcome Page



Need to enter the unique RFID and Password to Register

Registration Page

Figure 3.2-App Registration Page

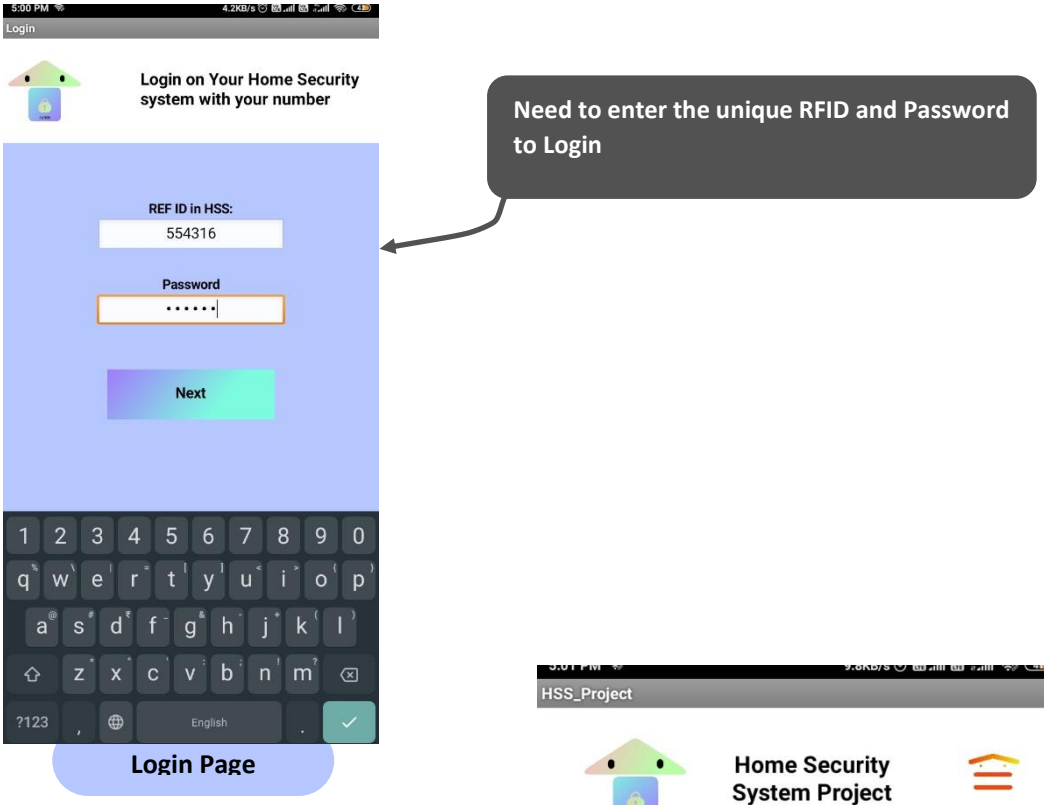


Figure 3.3-App Login Page

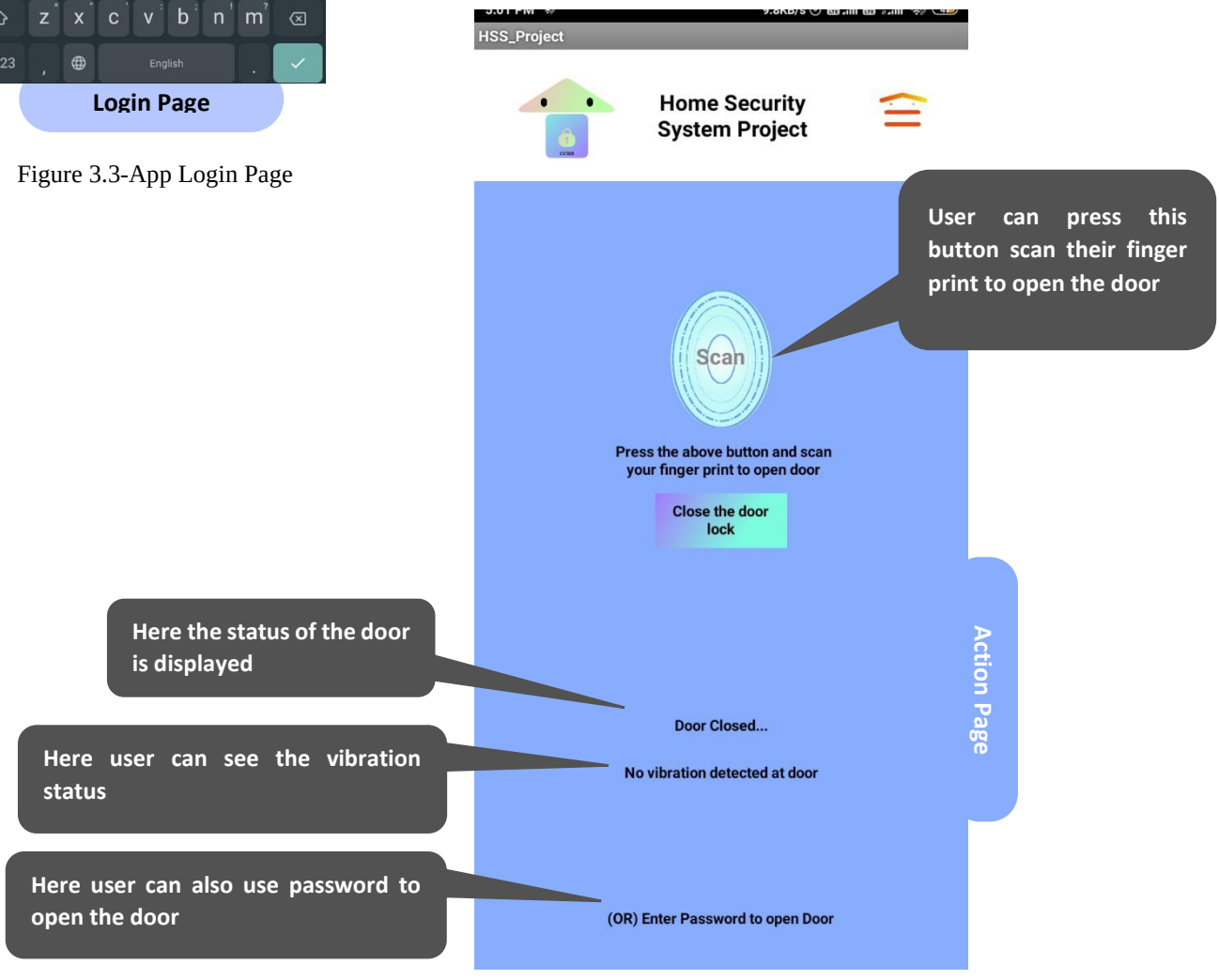


Figure 3.4-App Action Page

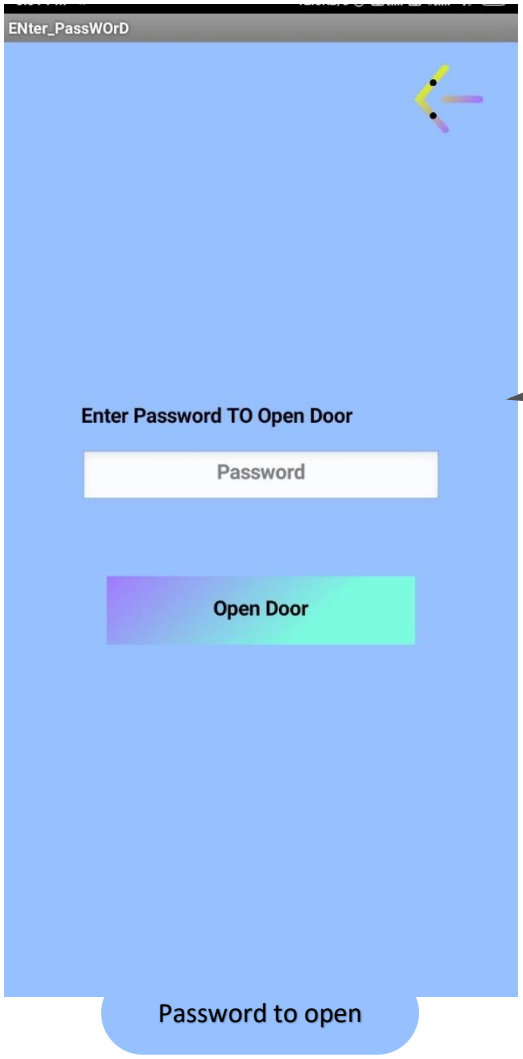


Figure 3.5-App Password Page

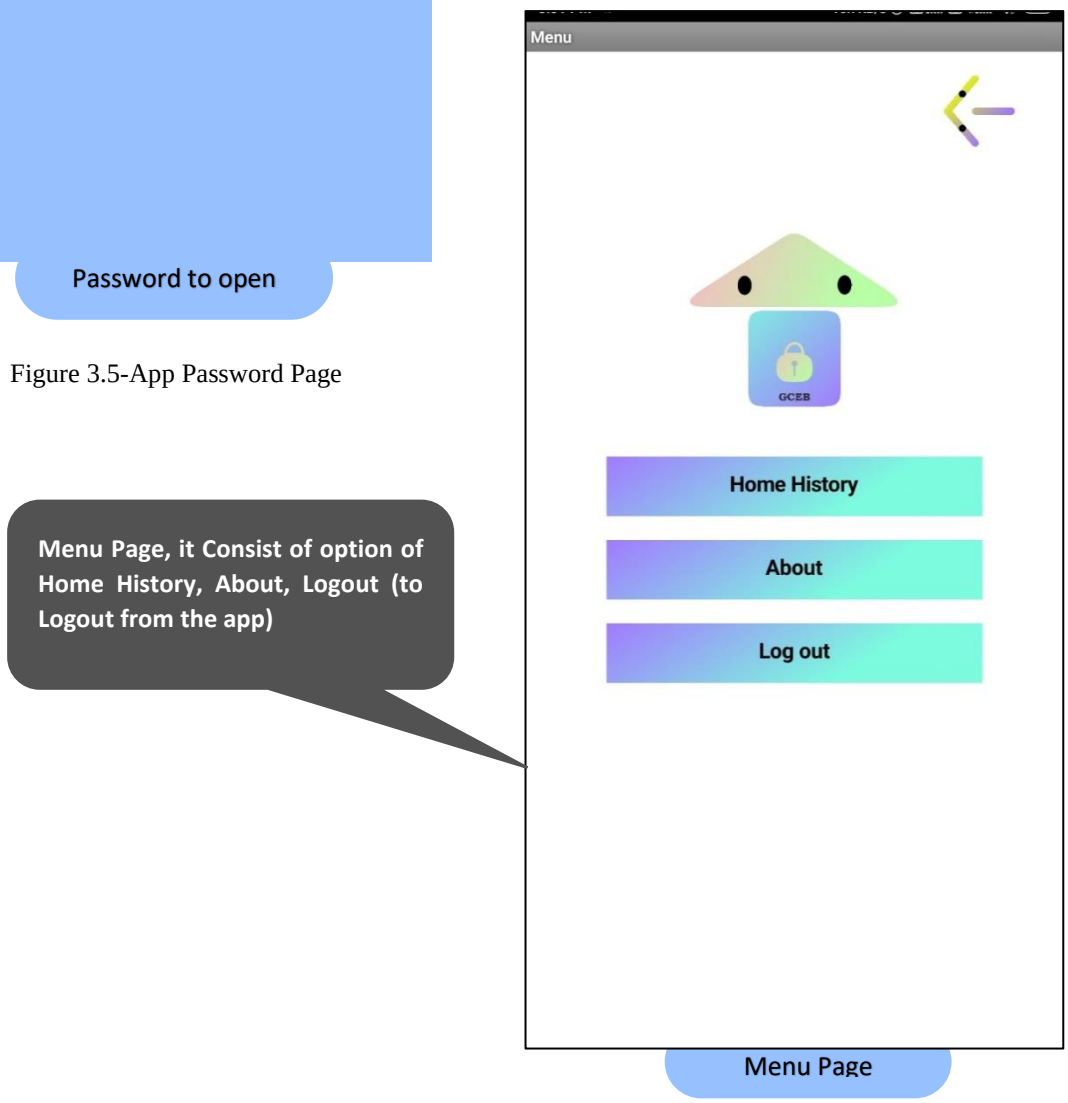


Figure 3.6-App Menu Page

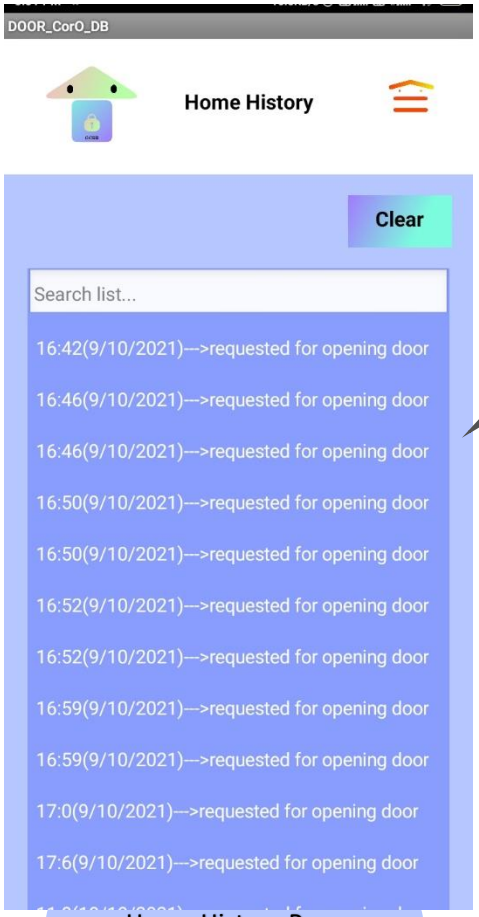


Figure 3.7-App Home History Page

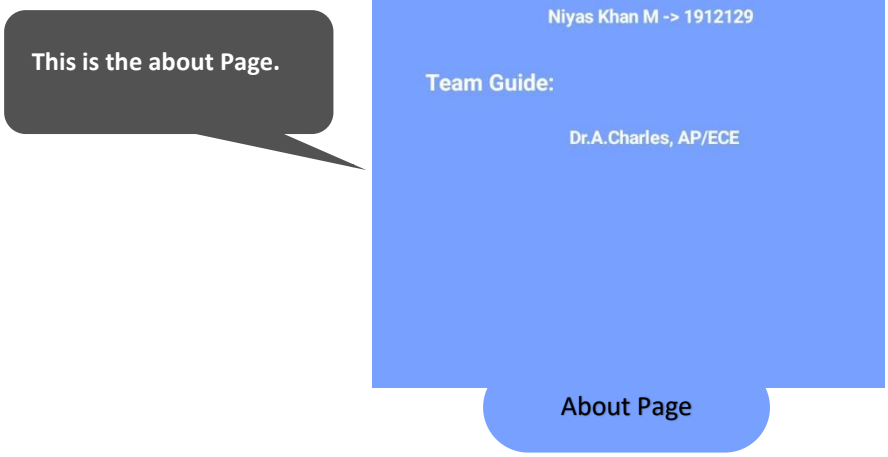


Figure 3.8-App About Page

CHAPTER 4

APP BLOCK CODING IN MIT APP INVENTOR

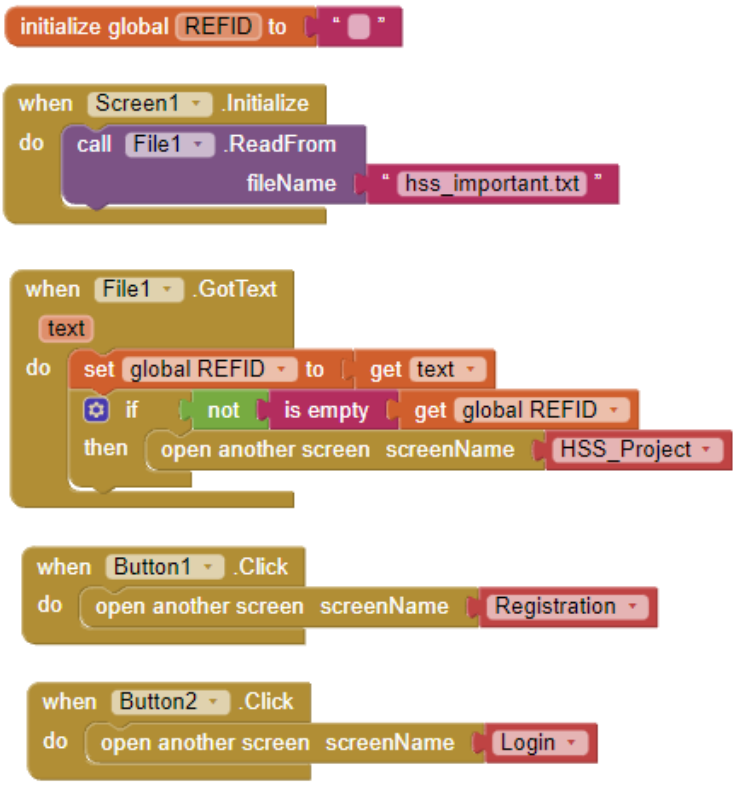


Figure 4.1-App Welcome Screen Block Coding

The above block code is used for welcome page in app. When ever this screen initialize it check for the file hss_important.txt file. If the file is found it will take us to the action page otherwise it urges the user to register or login to go to the action page.

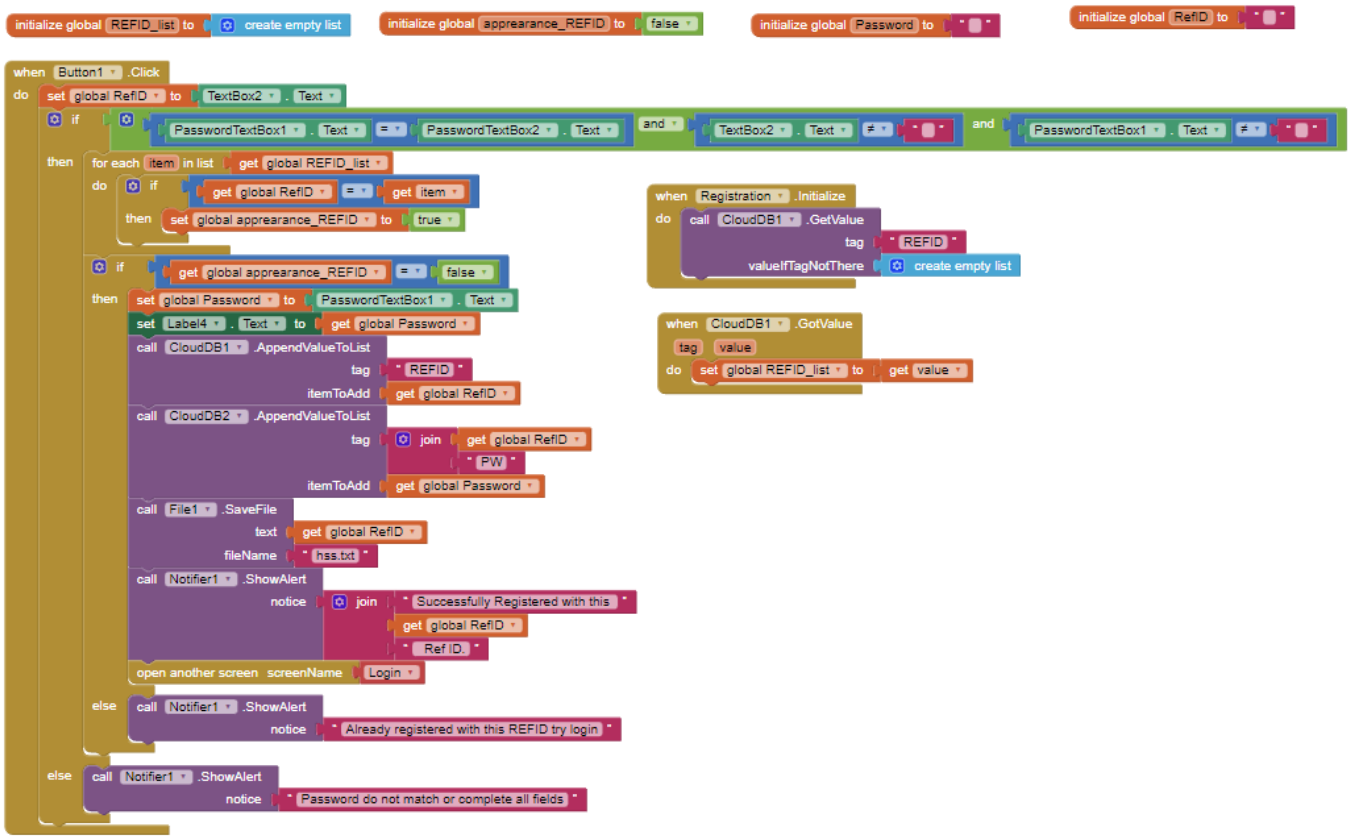


Figure 4.2-App Registration Screen Block Coding

The above block coding is used in Registration page in our app. User can able to register by providing the RFID, Password and confirm the Password by entering the password again and then click register button. It checks for that the Entered RFID is already available or not, if it is available, it stores the RFID and Password. It also Check for whether the New-Password and Confirm Password matches not.

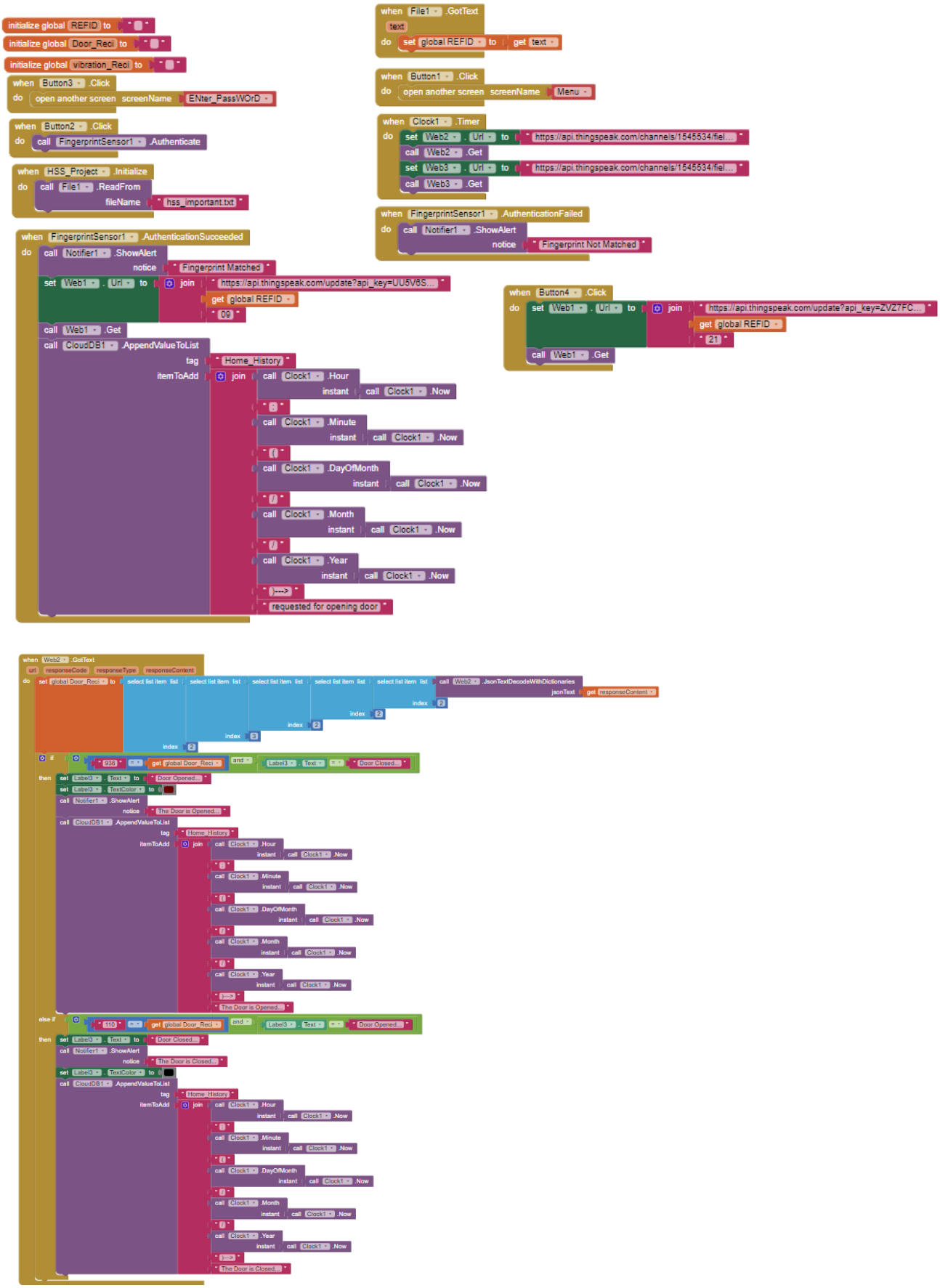


Figure 4.3.a-App Action Page Block Coding

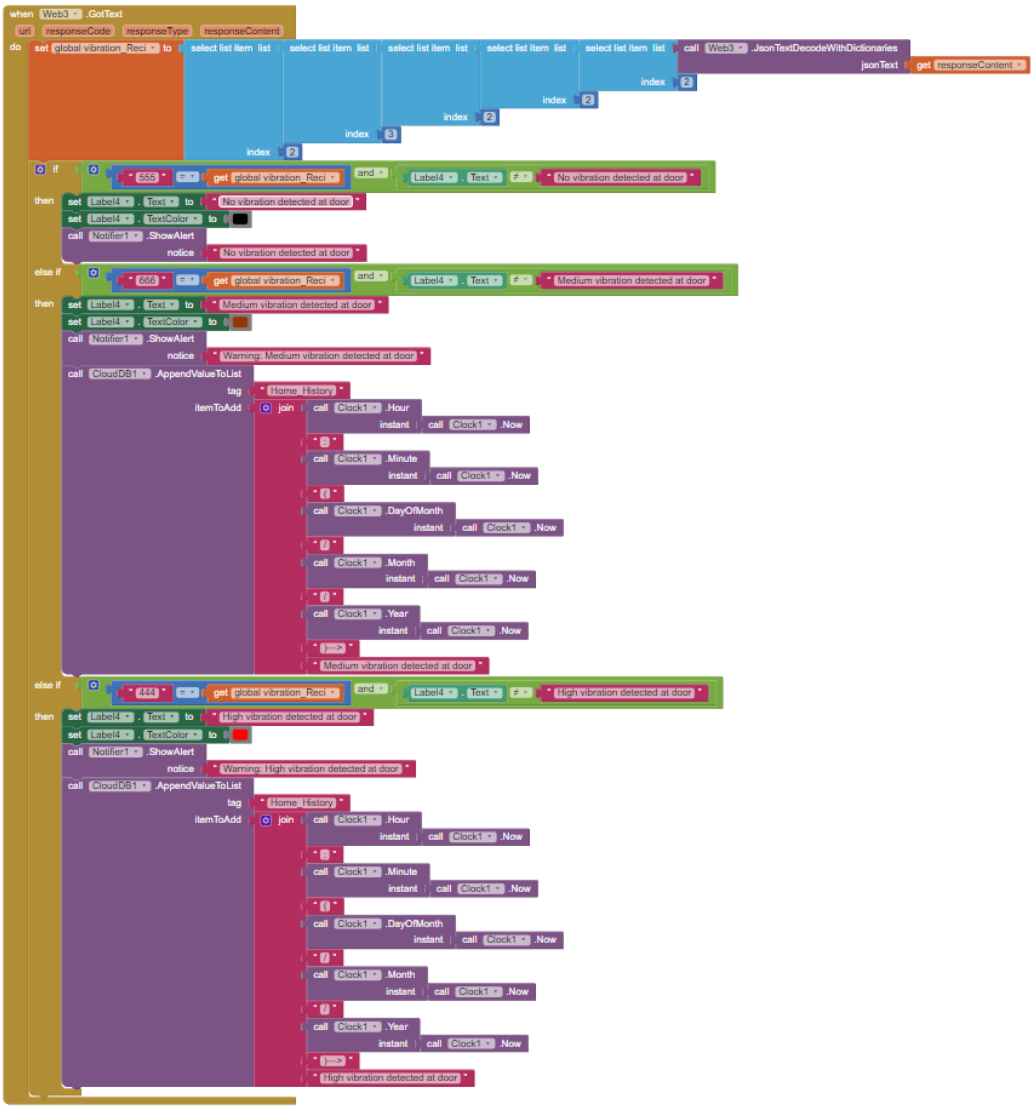


Figure 4.3.b-App Action Page Block Coding

The above Block Coding (in figure 6.3.a & 6.3.b) shows the block coding for the action page in the app. This code includes the checking of the figure print, If the identity is verified it sends the open signal to the Thingspeak then the Thingspeak send the information to the nodeMCU to open the door. This code also get the vibration and door open or close status from the Thingspeak and display it as the label. Those actions are saved in the cloudDB

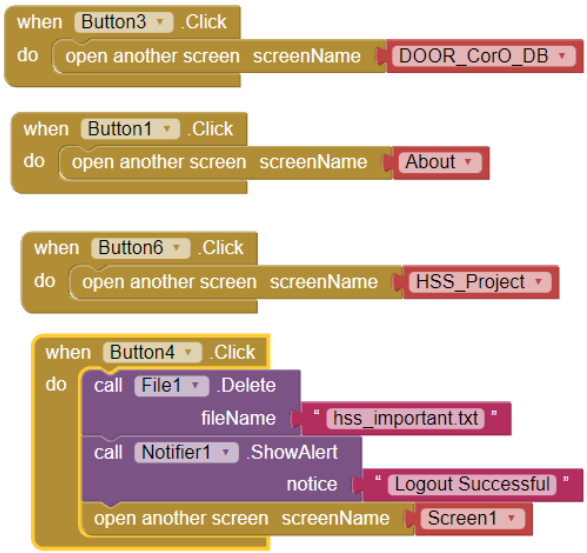


Figure 4.4-App Menu Page Block Coding

The above block code is the code used for the menu page. It will open the respective page listed in that menu, when the user presses the button.

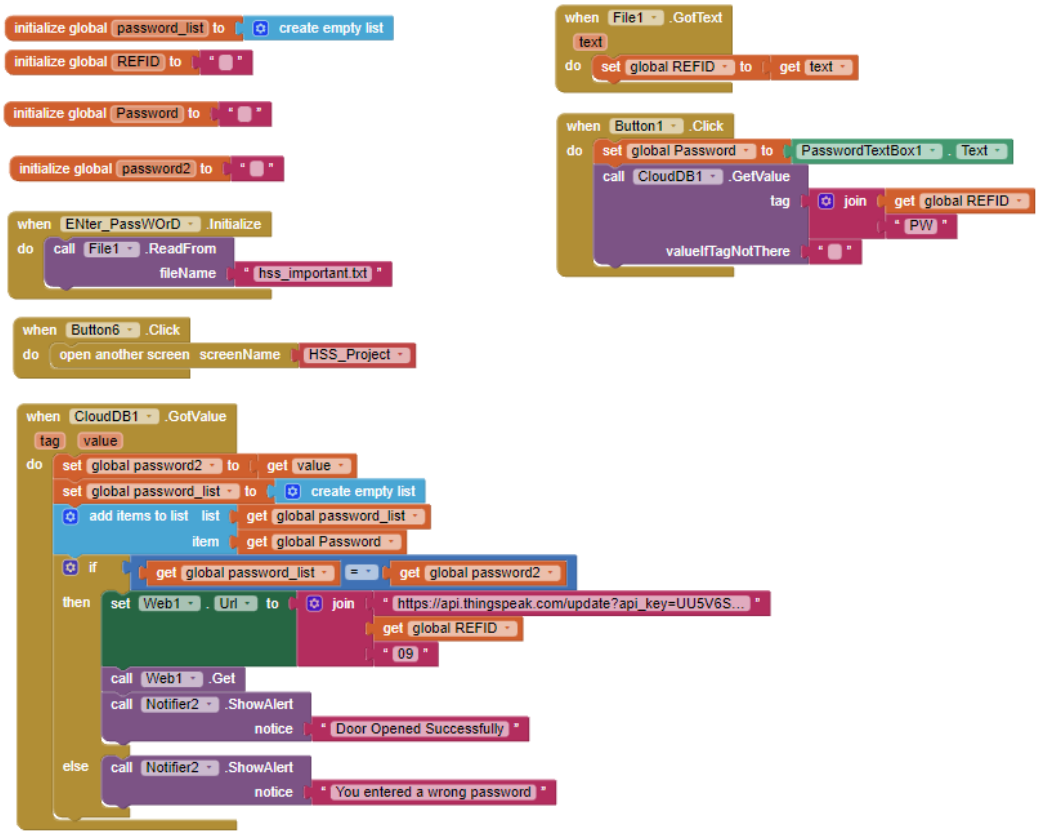


Figure 4.5-App Enter Password Page Block Coding

The above block code in figure 6.5 is the code used for the Enter password page. User need to the Enter the password, If the password is correct, it will send the open door signal to Thingspeak.com.

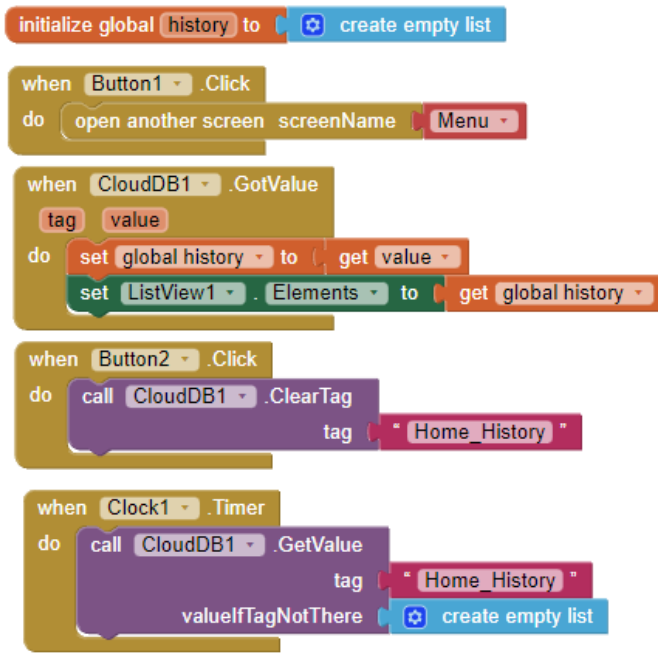


Figure 4.6-App Home History Page Block Coding

The above block code is code used for the Home History Page. It takes the list which is previously stored in the cloudDB with tag Home_History and store it in the separate list. Then it will show those list in the app in the list box.

CHAPTER 5

SOURCE CODE USED IN ARDUINO IDE

5.1 CODE

```
//The Home Security System
//Team Members
// Baranitharan B
// Cibi P
// Niyas Khan M
//Team Guide
// Dr.charles.A(AP/ECE)
#include<ThingSpeak.h>
#include<ESP8266WiFi.h>
#include<ESP8266WebServer.h>
#include<Servo.h>

unsigned long channel_num=1545534;
const char* ssID="Hari_animie";
const char* password="IiiamIiiam";
String REFID="554316";
String input;
String Command;
Servo drservo;
bool checkrefid;
int door = 2; //GPIO2-->D4
int vibration=4;//GPIO3->D2
int door_OC=5;//GPIO-->D3
WiFiClient client;
void setup() {
```

```
Serial.begin(115200);
drservo.attach(door);
drservo.write(90);
pinMode(vibration,INPUT);
pinMode(door_OC,INPUT);
WiFi.begin(ssID,password);
erial.print("connecting");
while (WiFi.status()!=WL_CONNECTED)
{
  delay(500);
  Serial.print(".");
}
Serial.print("Connected, IP address: ");
Serial.print(WiFi.localIP());
ThingSpeak.begin(client);
Serial.println("The Home Security System");
Serial.println("Enter your Command with REFID fist");
}

void loop() {
  // put your main code here, to run repeatedly:
  //if(Serial.available()>0)
  //{
    input=String(ThingSpeak.readLongField(channel_num, 1,"00QD50X7HA9FWB02" ));
    input.trim();
    Serial.println("input=");
    Serial.print(input);
    checkrefid=checkREFID(input);
    if(checkrefid==true)
    {
```

```
   Serial.println("REFID Verified");
   Serial.println("recived command");
   Command=String(REFID+"09");
   //Serial.print(input);
   if(command_Check(input,Command))
   {
     drservo.write(0);
     Serial.println("Door lock opened");
    /* delay(10*1000);
     drservo.write(90);
     int evv1=ThingSpeak.writeField(channel_num,1,55576870,"UU5V6S2ASVKCIB4F");
     if(evv1!=200){
   Serial.println("data upload failed.. on numl");
   //vibbbb();*/
  }
  else if(command_Check(input,"55431621"))
  {
   drservo.write(90);
   Serial.println("Door Lock Clocsed");
   }
   else
   {
     //Serial.println(command);
     //Serial.println(input);
     Serial.println("command not identified");
   }
  }
  else
   Serial.println("REFID not matched");
//}
```

```
 /* if(digitalRead(vibration)==HIGH)
  {
    Serial.println("vibration occured");
  }*/
  vibbbb();
  if(digitalRead(door_OC)==HIGH)
  {
    Serial.println("door is closed");
    int evv=ThingSpeak.writeField(channel_num,2,936,"UU5V6S2ASVKCIB4F");
  }
  else if(digitalRead(door_OC)==LOW)
  {
    Serial.println("door is opened");
    int evv=ThingSpeak.writeField(channel_num,2,110,"UU5V6S2ASVKCIB4F");
  }
}

void vibbbb()
{
  long v_meas=vpulse();
  delay(50);
  Serial.println(v_meas);
  if (v_meas<10000)
  {
    int evv=ThingSpeak.writeField(channel_num,3,555,"UU5V6S2ASVKCIB4F");
    if(evv!=200){
      Serial.println("data upload failed in low vibration..");
      Serial.println(evv);
      //vibbbb();
    }
```

```
   else
     Serial.println("data updated");
  }
  if (v_meas>10000 && v_meas<20000)
  {
   int evv=ThingSpeak.writeField(channel_num,3,666,"UU5V6S2ASVKCIB4F");
   if(evv!=200){
     Serial.println("data upload failed in medium vibration..");
     Serial.println(evv);
    // vibbbb();
   }
   else
     Serial.println("data updated");
  }
  if (v_meas>20000)
  {
   int evv=ThingSpeak.writeField(channel_num,3,444,"UU5V6S2ASVKCIB4F");
   if(evv!=200){
     Serial.println("data upload failed in high vibration..");
         Serial.println(evv);
     //vibbbb();
   }
   else
     Serial.println("data updated");
  }
}

long vpulse()
{
  delay(10);
```

```
  long v_meas=pulseIn(vibration,HIGH);
  return v_meas;
}
bool command_Check(String in1,String in2)
{
  int mismatch =0;
  if(in1.length()==in2.length())
  {
    for(int i=0; i<in2.length(); i++)
    {
      Serial.println(String(i)+"->"+String(in1[i])+"="+String(in2[i]));
      if(in1[i]!=in2[i])
      {
        mismatch++;
      }
    }
  }
/*  else
  {
    //Serial.println(in1.length());
    //Serial.println(in2.length());
    //Serial.println("size not matched");
    return false;
  }*/
  if(mismatch==0)
    return true;
  else
  {
    Serial.println("mismatch="+String(mismatch));
    return false;}
```

```
}
bool checkREFID(String input1)
{
  int mismatch =0;
  for(int i=0; i<5; i++)
  {
    if(REFID[i]!=input1[i])
    {
      mismatch++;
    }
  }
    if(mismatch==0)
      return true;
    else
    {
      Serial.println("mismatch="+String(mismatch));
      return false;}
}
```

5.2 EXPLANATION

Here we use the Arduino IDE to program the NodeMCU microcontroller. The library used in code are listed below.

```
#include<ThingSpeak.h>
#include<ESP8266WiFi.h>
#include<ESP8266WebServer.h>
#include<Servo.h>
```

ThingSpeak.h Library is used to link the nodeMCU to the Thingspeak server. ESP8266WiFi.h is the WiFi library which used to connect the nodeMCU to the internet via the local Hotspot. For to connect to internet we should mention the SSID and Password in our code. Servo.h Library is used to control the servo motor to open and close the door lock.

Then we Define the D2 for Door lock, D4 for Vibration, D5 for Door Open Close status in the code.

Serial.begin(115200); is used to display the result in Serial Monitor to debug the pogram in the bandrate of 115200.

Then using pinMode(pin,INPUT/OUTPUT) is used to define whether the pin is input or output. We define D4 and D5 pin as the INPUT.

WiFi.begin(ssid, password); beigns the wifi connect with the ssid mentioned in the code. ThingSpeak.begin(client); begins the connection to the thighspeak via the wifi client.

ThingSpeak.readLongField(channel name, field number, read api key); is used to get the command updated on thingspeak by the app.

The function checkrefid(string); checks the command and the REFID. IF it matches only it will Proceed for the further action.

If the command get from the Thingspeak first field is “REFID+09” it will open the door lock. If the command get is “REFID+21” nodeMCU will close the door lock.

drservo.write(90); rotates the servo motor to 90degree, lock knob is attached to servo motor as shown in the figure 3.4, so the lock will get opened. similarly drservo.write(0); will close the door lock by rotating the servo motor.

Vibbbb() function check checks for the vibration. If the vibration is measured as more than 10,000 it writes 555 on the thingspeak field 3, in app it will show as “No vibration detected”. Suppose if the vibration is measured in between 10,000 and 20,000 it writes 666 on the thingspeak field 3, in app it will show as “Medium vibration detected”. If the vibration is measured as more than 30,000 it writes 666 on the thingspeak field 3, in app it will shown as “High vibration detected”

ThingSpeak.writeField(channel number, filed number, value, write api key); is used to enter the value in the particular field.

Then it check the digital pin D5 which is where the switch like mechanism attached to door as shown in figure 3.2 is connected. If D5 is HIGH it will write the 936 on the thingspeak in field 2, in app it is shown as “door closed”. If D5 is LOW it will write the 110 on the thingspeak in field 2, in app it is shown as “door opened”. digitalRead(D5); is used to read the voltage at D5. The above process will get looping till the power is supplied to the NodeMCU.

CHAPTER 6

RESULT

The door lock opened when the figure print is matched, the vibration and door open close status is updated in the app successfully as shown in the image below.

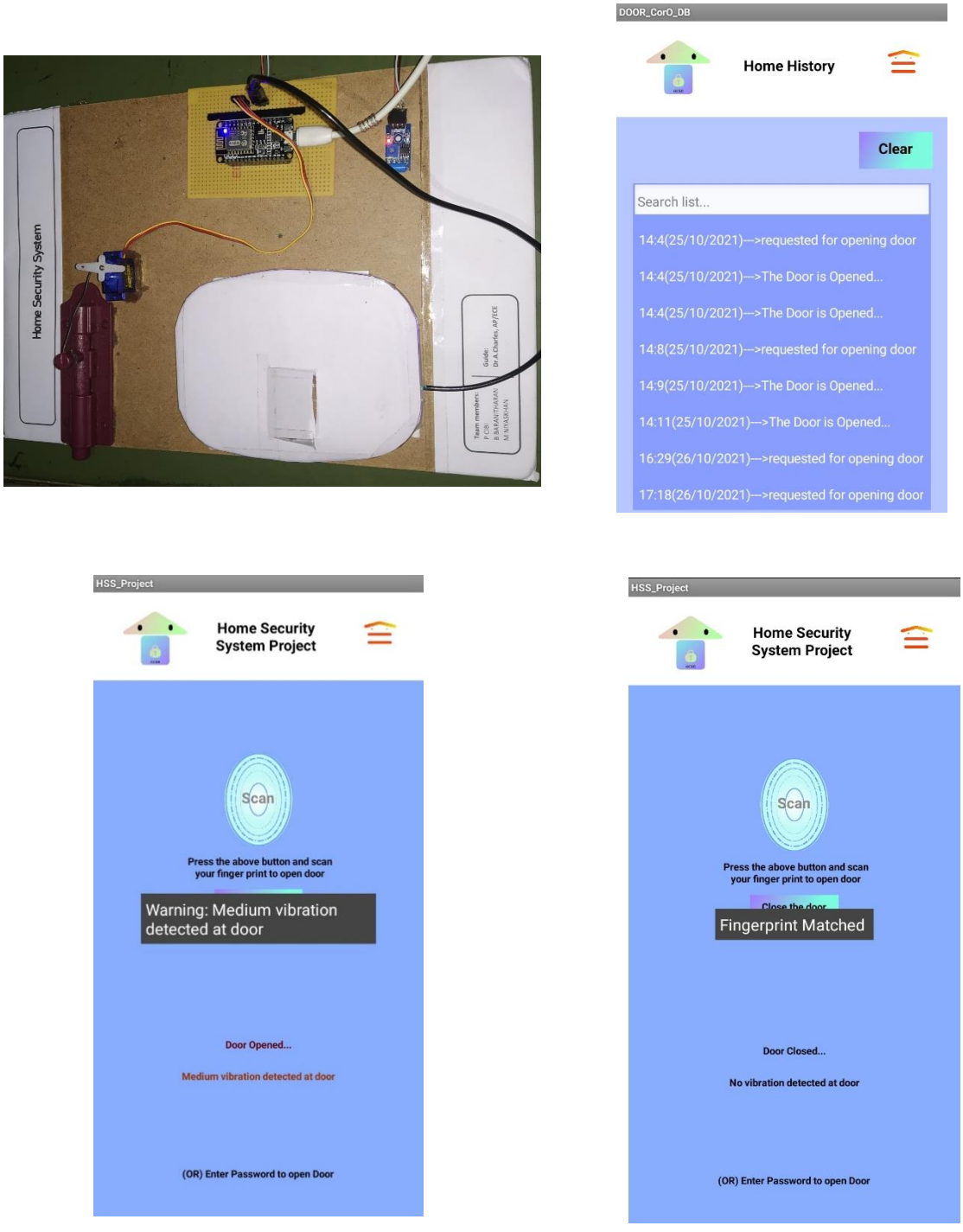


Figure 6.1-Result

CHAPTER 7

FUTURE SCOPE & CONCLUSION

7.1 FUTURE SCOPE

1) We can use the EEPROM in NodeMCU to store the ssid, password, and we can use the ESP8266WebServer.h library to create the local server webpage from that page user can easily change the ssid and password

2) In app we can add change password page.

3) We can add camera and mic to see and hear what happening on the door at any time.

7.2 CONCLUSION

The Home Security System does not allow unauthorised person to open the door and it also measure the vibration at the door when someone tries to destroy the door. It stores when the door is opened and closed in the app which can be viewed later in app. So, it can be used to save our home, valuable things. If someone tries to destroy the door the app will notify us we can take the necessary action such as inform police or to inform neighbours to save our home. So, this provides security to our Home.

REFERENCE

For Arduino IDE: "Arduino Software Release Notes". Arduino Project. Retrieved September 25, 2019.

For Idea: "OTP Based Smart Wireless Locking System Using Arduino" By Ashwini Kumar Sinha – Web Article - February 13, 2019.

For MIT App Inventor: Wolber, David; Abelson, Hal; Spertus, Ellen; Looney, Liz (May 2011), "*App Inventor for Android: Create Your Own Android Apps*", O'Reilly, ISBN 978-1-4493-9748-7

For Thingspeak.com: *An Introduction to ThingSpeak*, 23 November 2014 Published in Web.

For NodeMCU: "NodeMCU - A Perfect Board for IoT". *circuito.io blog*. 2018-11-21. Retrieved 2021-05-27

For Servo motor: "Interfacing Servo Motor With NodeMCU" By CodeChamp – Instructables blog.

For Vibration Sensor: "SW-420 Vibration Sensor Arduino Interface" Article from TheoryCircuits