**Pseudocode**

***Arduino:***

**Step 1:**

**+ Add Libraries to work with Fingerprint and DHT22 Sensor.**

**+ Assigned value to Digital\_Port\_Pins and Variables.**

SoftwareSerial mySerial(12, 13);

Adafruit\_Fingerprint finger = Adafruit\_Fingerprint(&mySerial);

#define SENSOR\_PIN 2

#define BUZZ\_PIN 7

**Step 2:**

**void setup()**

**{**

Serial.begin(9600); // Set the baud rate to match the **ESP32**

finger.begin(57600); // set the data rate for the **Fingerprint** sensor serial port

Sensor.begin(); // set the data rate for the **DHT22** sensor serial port  
  
 **+ Assigned the Input and Output mode to all Digital\_Pins.**

pinMode(BUZZ\_PIN, OUTPUT); // or INPUT

**+ And setup the initial state [High or Low] for some Digital\_Pins.**

digitalWrite(RELAY\_PIN\_BULB, LOW); // or HIGH

**+ Setup the LCD screen, setting the cursor and print out default Characters.** 

1. lcd.init(); 3. lcd.begin(16, 2); 5. lcd.print("Temp = ");

2. lcd.backlight(); 4. lcd.setCursor(0, 0);   
  
**}**

**Step 3:**

**void loop()**

**{**

getFingerprintID(); // Perform fingerprint authentication

handleFailed\_Attempts(); // Handle failed attempts and lockout period

Sensor\_Lightbulb(); // Control the lightbulb based on sensor input

Button(); // Check if the button is pressed for manual unlocking

handle\_Buzzer(); // Handle buzzer activation and deactivation

handle\_ESP32(); // Handle commands sent from the Blynk app of the ESP32

handle\_LCD(); // Handle value and function to run LCD

**}**

**+ getFingerprintID()**

**{**

**p** = finger.fingerSearch();

**if (p == FINGERPRINT\_OK)**

+ LED = Purple + Unlock the door + failed\_Attempts = 0

+ Deactivate the Buzzer **immediately** and send character ‘**3**’ to **ESP32** through

UART connection.

**else if (p == FINGERPRINT\_NOTFOUND)**

+ LED = Red + Door lock does nothing + failed\_Attempts++

+ The time(s) of first failed\_Attempts (=1) is recorded.

**}**

**+ handleFailed\_Attempts()**

**{**

**if (failed\_Attempts >= 5 && Timer starts from the first fail till now < 1 Minute)** 

+ failed\_Attempts = 0

+ Activate the Buzzer and send character ‘**1**’ to **ESP32** through UART connection.

+ The time(s) that activated the Buzzer is recorded.

**if (Timer starts from the first fail till now >= 1 Minute)**

+ failed\_Attempts = 0

+ The time(s) of first failed\_Attempts (=1) is clear (reset to 0).

**}  
+ Button()**

**{**

**if (BUTTON\_PIN == HIGH)** 

+ failed\_Attempts = 0 + Unlock the door

+ Deactivate the Buzzer **immediately** and send character ‘**3**’ to **ESP32** through

UART connection.

**}**

**+ Sensor\_Lightbulb()**

**{**

**if (sensorValue == HIGH)** 

+ Turn on the LightBulb.

**else**

+ Turn off the LightBulb.

**}**

**+ handle\_Buzzer()**

**{**

**if (buzzer is Activated && Timer(s) that activated the Buzzer until now** **>= 6s)**

+ **Deactivate** the Buzzer after **6s** running. It always works and counts the time whenever

the **Buzzer** is **activated** if there is no function that **immediately** **deactivates** the **Buzzer**

during the working time interval (**6s**).

**}**

**+ handle\_ESP32()**

**{**

**if (Serial.available())**

+ char command = **Serial.read()**; // read command from **Blynk** App of

// **ESP32** through UART Connection.

+ **if (command == '1')** >> Activate the Buzzer.

+ **else if (command == '3')** >> Deactivate the Buzzer **immediately**.

+ **else if (command == '5')** >> Unlock the door >> failed\_Attempts = 0 >>

>> send character ‘**7**’ to **ESP32** through UART connection.

**}**

**+ handle\_LCD()**

**{**

// Read temperature and humidity values from the sensor

**float temperature = Sensor.readTemperature();**

**float humidity = Sensor.readHumidity();**

+ **if (isnan(temperature) || isnan(humidity))** >> Print “**Error**” into LCD

+ **else if** >> Print value of **temperature and humidity** into LCD

+ **if (temperature > 35.0 || humidity < 20.0)**

>> Warning LED turn on >> Fan turn on.

+ **else** >> Warning LED turn off >> Fan turn off.

**}**

***ESP32:***

**Step 1:**

**+ Add Libraries to work with Blynk + Wifi module and Mutex -** (not important at all).

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

**+ Add Blynk Token Device to connect with App.**  
 #define BLYNK\_TEMPLATE\_ID "………."

#define BLYNK\_TEMPLATE\_NAME "…….…"

#define BLYNK\_AUTH\_TOKEN "………."

------------------------------------------------------

char auth[] = BLYNK\_AUTH\_TOKEN;

char ssid[] = "…….Wifi…..";

char pass[] = "……pass-wifi…...";

**+ Assigned value to Digital\_Port\_Pins and Variables.**

#define button1\_pin 26

#define led1\_pin 16

**Step 2:**

**+ Change the virtual pins according to the rooms. (Having 5 rooms from V0 to V4)**

#define buzz\_vpin V0 #define button1\_vpin V2

#define lock\_vpin V1 #define button2\_vpin V3

**+ This function is called every time the device is connected to Blynk.Cloud which requests the latest state from the server.**

**BLYNK\_CONNECTED() {**

Blynk.syncVirtual(buzz\_vpin); Blynk.syncVirtual(button1\_vpin);

Blynk.syncVirtual(lock\_vpin); Blynk.syncVirtual(button2\_vpin);  
 **}**

**+ Blynk write signal 1 or 0 (On or Off) from the App to “Device State” of ESP32.**

**BLYNK\_WRITE(buzz\_vpin)**

**{**

+ buzz\_state = **param.asInt();** // (= 1 or 0)

+ **if (buzz\_state == 1)** >> send character ‘**1**’ to **Arduino UNO** through UART

connection.

+ **else** >> Send character ‘**3**’ to **Arduino UNO** through UART connection.

**}**

**BLYNK\_WRITE(lock\_vpin)**

**{**

+ lock\_state = **param.asInt();** // (= 1 or 0)

+ **if (lock\_state == 1)** >> send character ‘**5**’ to **Arduino UNO** through UART

connection.

+ **else** >> do nothing here.

**}**

**BLYNK\_WRITE(button1\_vpin / button2\_vpin / button3\_vpin)**

**{**

+ led1\_state **/** led2\_state **/** relay\_state = **param.asInt();** // (= 1 or 0)

+ **digitalWrite**(led1\_pin **/** led2\_pin **/** relay\_pin**,** led1\_state **/** led2\_state **/**

relay\_state);

>> Turn on and off 2 LEDs and FAN follow the state from the Blynk App.

**}**

**Step 3:**

**void setup()**

**{**

Serial.begin(9600); // Set the baud rate to match the **Arduino UNO**

**+ Assigned the Input and Output mode to all Digital\_Pins.**

pinMode(button1\_pin, INPUT); pinMode(led1\_pin, OUTPUT);

**+ And setup the initial state [High or Low] for some Digital\_Pins.**

digitalWrite(led1\_pin, LOW);

**+ Setup the connection to Blynk.cloud App through the WiFi module.**

Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);  
**}**

**Step 4:**

**void loop()**

**{**

**Blynk**.run();

listen\_push\_buttons();

read\_serial\_commands();

**}**

**+ listen\_push\_buttons()**

**{**

+ **if (button1\_pin == HIGH)** >> led1\_state = **!**led1\_state; >> change state of 

LED1 in the Blynk app >> Turn on and off the LED1 following the state.

+ **if (button2\_pin == HIGH)** >> led2\_state = **!**led2\_state; >> change state of

LED2 in the Blynk app >> Turn on and off the LED2 following the state.

+ **if (button3\_pin == HIGH)** >> relay\_state = **!**relay\_state; >> change state of

Relay\_Fan in the Blynk app >> Turn on and off the Relay\_Fan following the state.

**}**

**+ read\_serial\_commands()**

**{**

**if (Serial.available())**

+ char command = **Serial.read ()**; // read command from **Arduino UNO**

+ **if (command == '1')** >> Assign the **Activate state** (=1) of the **Buzzer** to

Blynk App.

+ **else if (command == '3')** >> Assign the **Deactivate state** (=0) of the

**Buzzer** to Blynk App.

+ **else if (command == '7')** >> Assign the **Deactivate state** (=0) of the

**Lock** to Blynk App.

**}**