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 record.

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 see 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 follow the state.

+ if (button2_pin == HIGH) >> led2_state = !led2_state; >> change state of LED2 in the Blynk app >> turn on and off the LED2 follow 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 follow 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.

}