Procedure:

1. Setting up the Hardware:

a. Gather all the required components mentioned, including the ESP32DEVKIT, sensors (Piezo, Gas Sensor, Ultrasonic Distance Sensor, PIR Sensor), resistors (R1, R2, R4, R5), microcontroller (Positional Micro Servo), DC motor, transistors (T1, T2), photoresistor, light bulbs, and a 3.3V battery.

b. Connect the components according to the circuit diagram provided with the project documentation. Ensure proper wiring and connections.

Installing Blynk.io:

a. Visit the Blynk.io website and create an account if you don't have one.

b. Download and install the Blynk app on your mobile device from the respective app store.

Creating the Blynk Project:

a. Open the Blynk app and create a new project.

b. Choose the ESP32 as the hardware model in the Blynk project settings.

c. Generate an authorization code for your project.

Setting up the ESP32 :

a. Connect the ESP32 Module to your computer using a USB cable.

b. Install the necessary drivers and Arduino IDE software if not already done.

c. Open the Arduino IDE and select the appropriate board and port.

Assembling the Hardware Components:

a. Connect the components to the ESP32 as follows:

- Connect the Piezo (PIEZO1) to a digital pin on the ESP32 for generating sound alerts.
- Connect the Gas Sensor (GAS1) to an analog pin on the ESP32 for detecting gas levels.
- Connect the Ultrasonic Distance Sensor (PING1) to the ESP32 for measuring water tank levels.
- Connect the PIR Sensor (PIR1) to a digital pin for motion detection.
- Connect the Positional Micro Servo (SERVO1) to a digital pin for controlling automatic doors.
- Connect LED’s for smart lighting control simulation.

- Connect a 3.3V battery to power the system.

Writing and Uploading the ESP32 Code:

a. Write the ESP32 code to control and integrate all the components as per the project requirements.

b. Ensure that the code includes functionalities for automatic water level monitoring, smart lighting control, automatic doors, smoke detection, and data visualization on UI dashboards.

c. Upload the code to the ESP32 using the Arduino IDE.

Ensure that your mobile device is connected to the same Wi-Fi network as the ESP32 Module.

b. Open the Blynk app and tap on the "+" button to add a new widget.

c. Select the desired widget type for each automation feature (e.g., buttons, sliders, graphs).

d. Configure each widget to correspond to the appropriate digital or analog pins on the ESP32 .

Testing and Calibration:

a. Ensure all hardware connections are secure and functioning correctly.

b. Test each component's functionality individually and in conjunction with others to ensure proper automation features.

c. Calibrate sensors and adjust thresholds as needed to achieve accurate readings and desired automation behavior

Testing and Operation:

a. Power the ESP32 Module using the 9V battery.

b. Verify that the ESP32 Module is connected to the Blynk cloud by checking the LED status on the board.

c. Interact with the Blynk app widgets to control and monitor the home automation features.

d. Test each automation feature, such as checking water tank levels, controlling lighting, detecting motion, and monitoring gas levels.

Data Visualization and Analysis:

a. Explore the Blynk app's UI dashboard to visualize data insights from the sensors.

b. Utilize the provided widgets, such as graphs and charts, to represent the sensor data in a visually appealing manner.

ESP32 CODE:

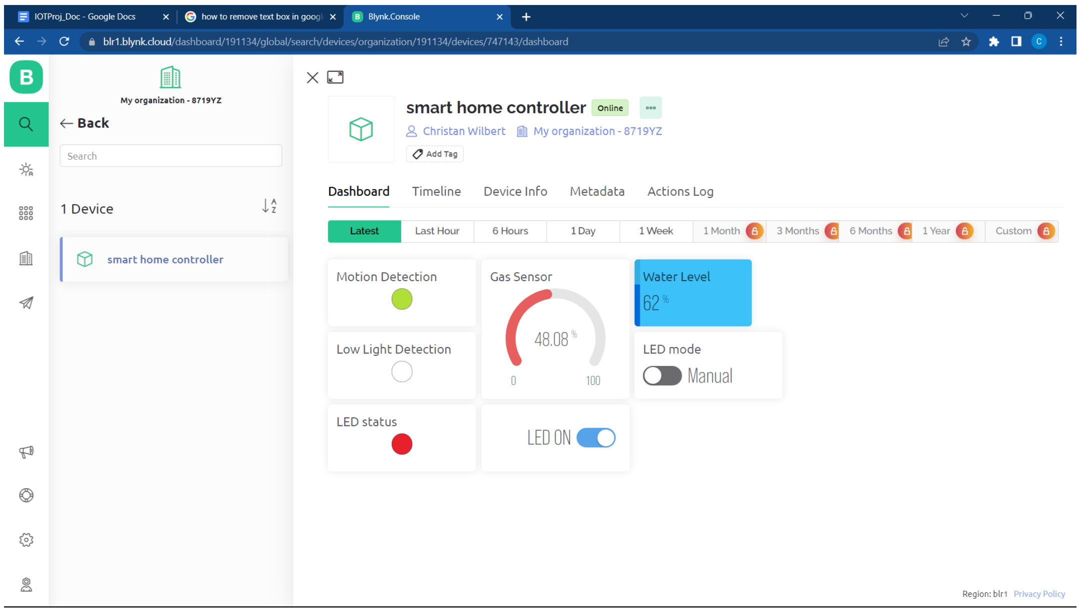
```
/* Fill-in information from Blynk Device Info here */

#define BLYNK_TEMPLATE_ID           "TMPL3bMRtL3e5"

#define BLYNK_TEMPLATE_NAME         "Quickstart Template"

#define                                         BLYNK_AUTH_TOKEN
"3-G9wH7mwsgpQkqPJ_wER4SVwo9kieXa"

#define BLYNK_PRINT Serial

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

#include <ESP32Servo.h>

int pos = 0;

// Your WiFi credentials.

char ssid[] = "Wifi_SSID";

char pass[] = "wifi_password";

BlynkTimer timer;

const int gasPin=34;

const int trigPin = 32;

const int echoPin = 33;

const int motionPin = 25;

const int lightDetection = 26;

const int servoPin=18;

const int LedPin=19;

const int Buzzer=14;
```

```
long duration,containerHeight=24.0;

int distance;

int motion,warm_up;

int ledMode=1;

int ledControl=0;

int LDR_Reading=0;

Servo myservo;

// This function is called every time the Virtual Pin 0 state
changes

void updateLED(){

  if((ledMode&&LDR_Reading)||(!ledMode&&ledControl)){

    digitalWrite(LedPin,HIGH);

    Blynk.virtualWrite(V9,1);

  }

  else{digitalWrite(LedPin,LOW);Blynk.virtualWrite(V9,0);}

}

BLYNK_WRITE(V8)

{

  // Set incoming value from pin V8 to a variable

  ledMode = param.asInt();

  updateLED();

}

BLYNK_WRITE(V7)

{

  // Set incoming value from pin V7 to a variable

  ledControl = param.asInt();

  updateLED();

}
```

```
// This function is called every time the device is connected
to the Blynk.Cloud

BLYNK_CONNECTED()

{

  Serial.println("Connected to blynk");

}

int readUltrasonicDistance()

{

  digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in
microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the distance

distance = duration * 0.034 / 2;

return distance;

}

// This function is called every 1 second

void myTimerEvent()

{

  float h = analogRead(gasPin);

  if (isnan(h))

  {

     Serial.println("Failed to read from MQ-5 sensor!");

     return;

  }
  float gasValue=(h/4095)*100;
                                 int                water_level=
((containerHeight-readUltrasonicDistance())*100.0)/containerHei
ght;


  Blynk.virtualWrite(V3, gasValue);

  Blynk.virtualWrite(V4,water_level);

  LDR_Reading = digitalRead(lightDetection);

  Blynk.virtualWrite(V6,LDR_Reading);

  updateLED();

  if(gasValue>60){tone(Buzzer,900);}else{noTone(Buzzer);}

}

//coding the servo based on motion sensor values

void motionDetect(){

  motion = digitalRead(motionPin);

  int x=myservo.read();

  if( motion == LOW )

  {

   Serial.print("No object in sight\n\n");

   if(x>0){

     for(int i=x;i>0;i=i-2){

    myservo.write(i);

    delay(50);

  }}

   Blynk.virtualWrite(V5,0);

}else

  {

    Serial.print("Object detected\n\n");

    Blynk.virtualWrite(V5,1);

    if(x<90){
     for(int i=x;i<=90;i=i+5){
    myservo.write(i);
    delay(50);
      }
    }
  }
}
void setup()
{
  Serial.begin(115200);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  // You can also specify server:
    //Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass, "blynk.cloud",
80);
            //Blynk.begin(BLYNK_AUTH_TOKEN,      ssid,      pass,
IPAddress(192,168,1,100), 8080);

  // Setup a function to be called every second
  timer.setInterval(1000L, myTimerEvent);
  //function called every 5 seconds
  timer.setInterval(5000L,motionDetect);
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT);
pinMode(motionPin,INPUT);
pinMode(lightDetection,INPUT);
pinMode(servoPin,OUTPUT);
pinMode(LedPin,OUTPUT);
pinMode(Buzzer,OUTPUT);
myservo.attach(servoPin);
myservo.write(0);
```

```
}
void loop()
{
  Blynk.run();
  timer.run();
}
```

Results and Discussion:


smart home controller Online

Christan Wilbert My organization - 8719YZ

Motion Detection ●

Gas Sensor 48.08 % (0 – 100)

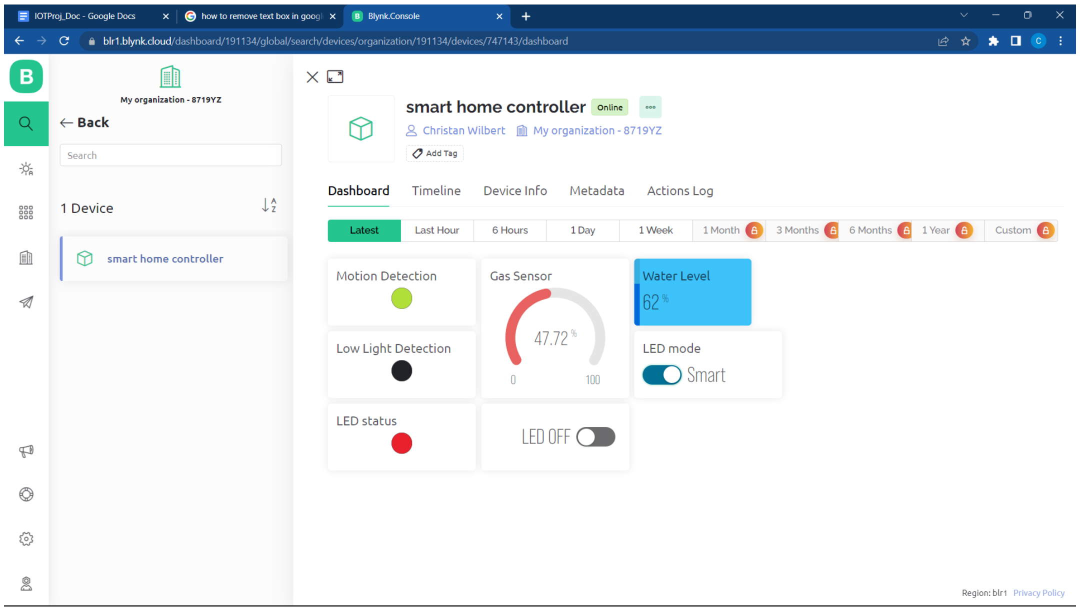
Water Level 62 %

Low Light Detection ○

LED mode Manual

LED status ●

LED ON



smart home controller

Motion Detection ●

Gas Sensor 47.72 %

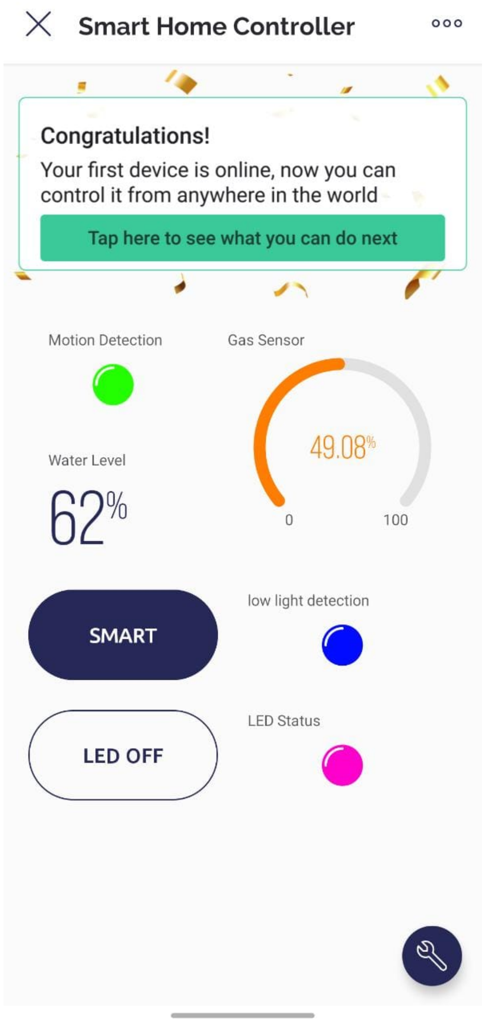
Water Level 62 %

Low Light Detection ●

LED mode: Smart

LED status ●

LED OFF



Smart Home Controller

Congratulations!
Your first device is online, now you can control it from anywhere in the world

Motion Detection ●

Gas Sensor 49.08%

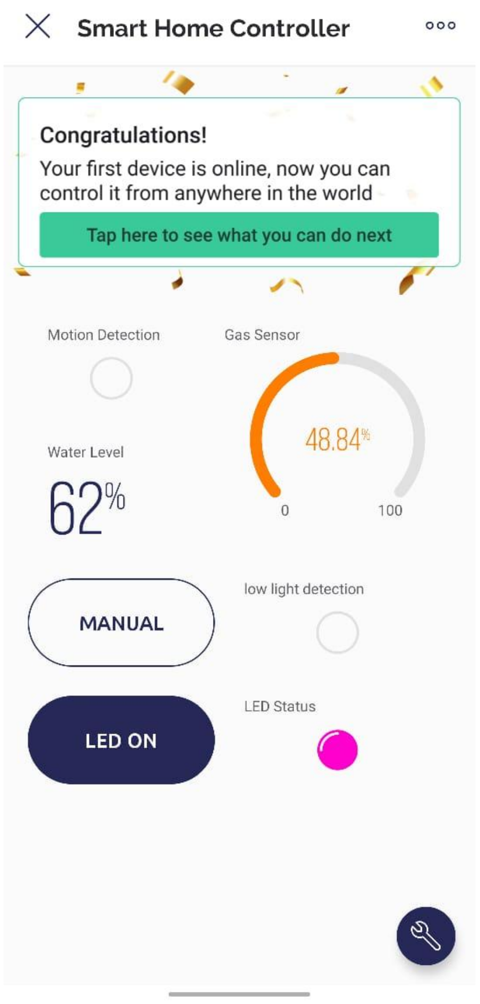
Water Level 62%

SMART

low light detection ●

LED OFF

LED Status ●



Smart Home Controller

Congratulations!
Your first device is online, now you can control it from anywhere in the world

Motion Detection ○

Gas Sensor 48.84%

Water Level 62%

MANUAL

low light detection ○

LED ON

LED Status ●