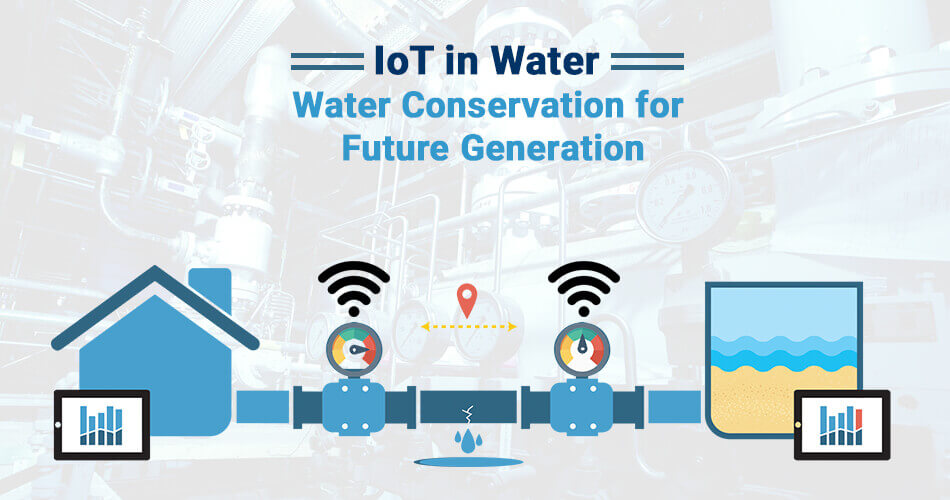
**DOCUMENTATION**

**SMART WATER MANAGEMENT**



TEAM DETAILS

|  |  |
| --- | --- |
| MENTOR | MRS.M.Maheswari |
| LEADER | B.Gayathri |
| MEMBERS | P.S.Anne  N.Boomika  A.Danis Swetha  S.Induja |
| PROBLEM DESCRIPTION | Describing the project objectives,component description,simulation steps and web application development. |

**IOT (INTERNET OF THINGS)**



IoT technology can be described as the ‘Internet of Things,’ based on the  concept that all the desired devices can be connected inside a specific  network for sharing data and information without any manual  intervention. We can segregate the devices used for this purpose based on  their capability to send, receive, and gather data in the network. IoT’s  salient properties include its secure nature, usability on smart devices,  ability to connect to any network, and facilitation of faster connectivity

OBJECTIVES

* water management gives a greater understanding of the water system, including flaw detection, preservation, and water management.

* A comprehensive database of regions with water losses or unlawful connections can be built with the introduction of smart water system technology by public service corporations.

* Smart water grids can save costs by conserving water and energy while improving the quality of service to consumers. Wireless data transfer allows consumers to assess their water use to reduce water costs in other circumstances.

    ABSTRACT

Water is a precious resource that can be intelligently managed. Effective water usage demands computerized home water supply management in a culture where water tanks, motors, and pumps are ubiquitous. The issue is providing a constant, high-quality, low-cost water supply. This study introduces a smart water management (IoT-SWM) system that may be used in structures that do not have access to a constant water supply but instead have water stored in enormous tanks underneath. The GSM module collects water use data from each home in a community and transmits it to the cloud, where it is analyzed. A smart water grid is a hybrid application that uses an inspection mode to identify leaks and measure the resulting height differences to keep track of the tank’s water level. The system automatically deactivates the affected section after detecting any water shortage or malfunction in the system mechanism, such as broken valves, pumps, or pipes. It sends an emergency signal to building managers. It monitors essential water quality elements regularly, and if they fall below acceptable levels, it sends warning signals to the building management, who can take action. Over an extended period, the system monitored and recorded all water quality metrics. The system restarts when the water pump has been reconnected and sends an emergency alert.

COMPONENTS USING IN SMART WATER MANAGEMENT

      1.RASPBERRY PI

      2.DHT-22

      3.ULTRASONIC DISTANCE SENSOR

      4.BUZZER REFERENCE

      5.LED

      6.RESISTOR

1.RASPBERRY PI

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

2.DHT-22

The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed).

3.ULTRASONIC DISTANCE SENSOR

Ultrasonic sensors can measure distance and detect the presence of an object without making physical contact. They do so by producing and monitoring an ultrasonic echo. Depending on the sensor and object properties, the effective range in air is between a few centimeters up to several meters.

4.BUZZER REFERENCE

  Buzzer Reference is  used to fetch and send signals.

The buzzer can operate in two modes: "smooth" (the default) and "accurate".

5.LED

   A light-emitting diode (LED) is a semiconductor device that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor.White light is obtained by using multiple semiconductors or a layer oflight-emitting phosphor on the semiconductor device.

7.RESISTOR

A resistor is an electrical component that limits or regulates the flow of electrical current in an electronic circuit. Resistors can also be used to provide a specific voltage for an active device such as a transistor.

All other factors being equal, in a direct-current (DC) circuit, the current through a resistor is inversely proportional to its resistance, and directly proportional to the voltage across it. This is the well-known Ohm's Law. In alternating-current (AC) circuits, this rule also applies as long as the resistor does not contain inductance or capacitance.

SIMULATION  STEPS:

Step1: Access Wokwi

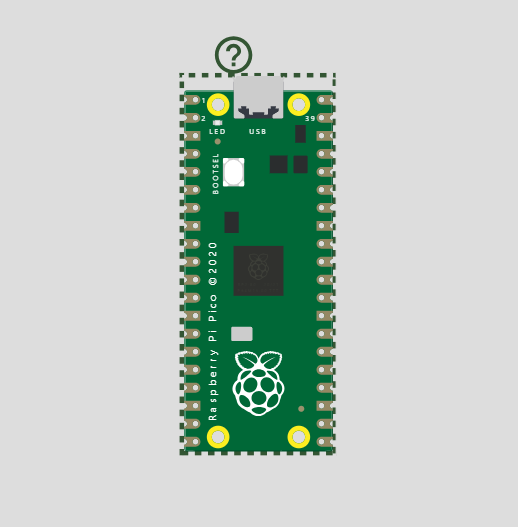
⦁ Go to the websites(⦁ [https://wokwi.com](https://wokwi.com/))

Step2: Create a project

⦁ Click on the new project

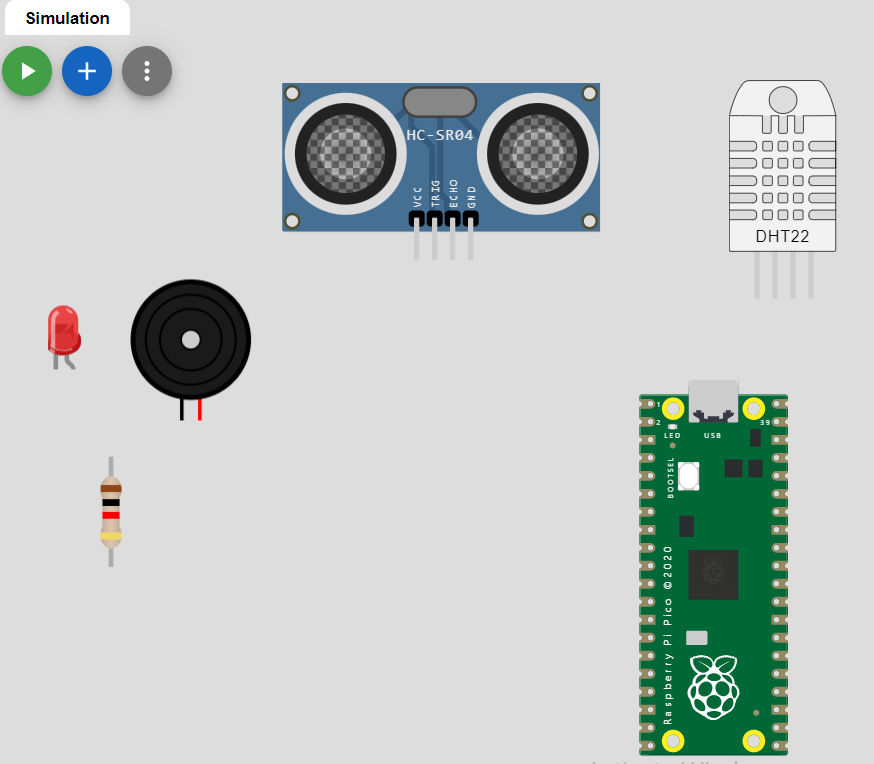
Step3:Add component

⦁ Search for Raspberry pi in the search panel.



Step 4: Add components

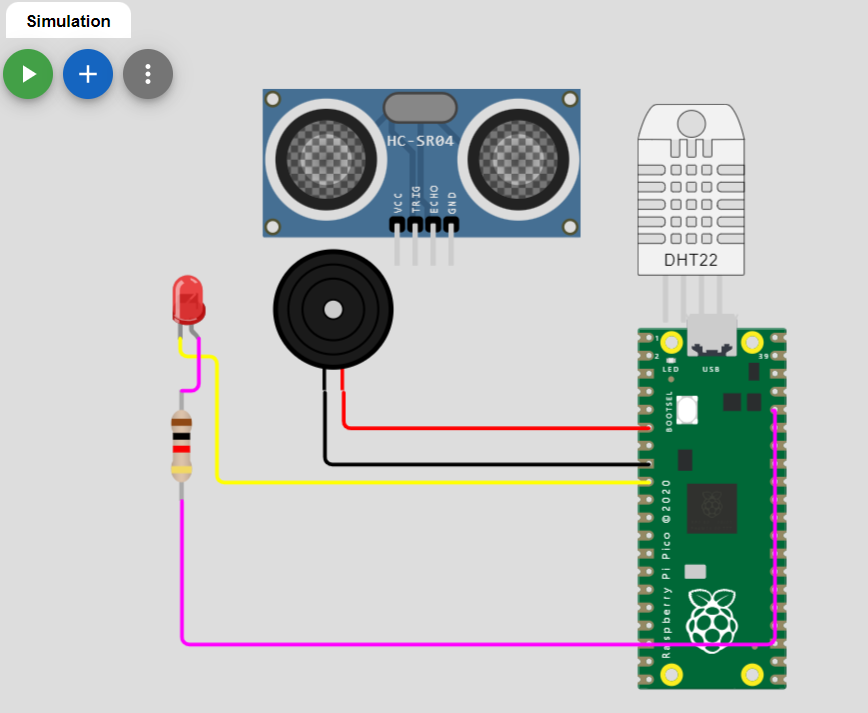
* Search for DHT22 sensor
* Search for Ultrasonic distance sensor
* Search for buzzer reference
* Search for LED
* Search for resistor

https://res-1.cdn.office.net/officeonline/we/s/hA3596C17DAD9A003_resources/1033/progress.gif 

Step5:Connections from Raspberry pi to buzzer reference and LED

* pico:3V3 to led1:A through r1:2
* pico:GP6 to led1:C
* pico:GND.2 to bz1:1

pico:GP4 to bz1:2

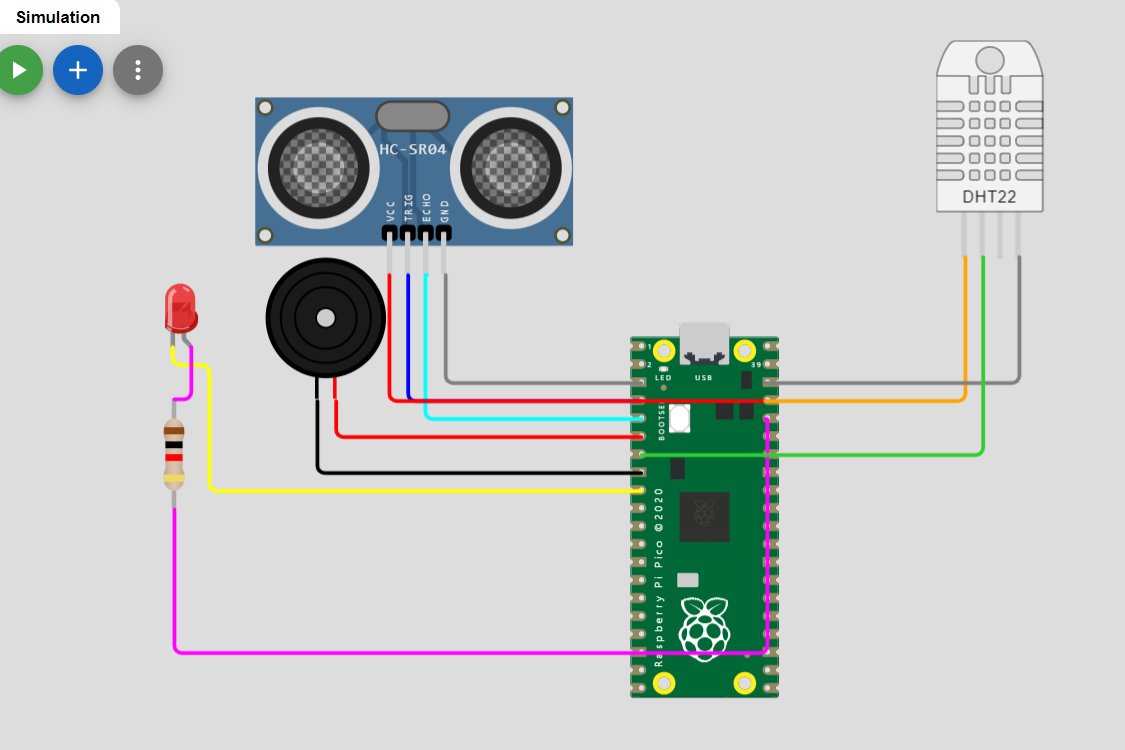
https://res-1.cdn.office.net/officeonline/we/s/hA3596C17DAD9A003_resources/1033/progress.gif 

Step6:Connections form raspberry pi to DHT22 and HC-SR04

pico:GP3 to ultrasonic1:ECHO

pico:3V3\_EN to ultrasonic1:VCC

* pico:GP2 to ultrasonic1:TRIG
* pico:GND.1 to ultrasonic1:GND
* pico:GP5 to dht1:SDA
* pico:3V3\_EN to dht1:VCC
* pico:GND.8 to dht1:GND

https://res-1.cdn.office.net/officeonline/we/s/hA3596C17DAD9A003_resources/1033/progress.gif 

Step7:Source code for the above simulation.

import time

import machine

Import dht

# Define GPIO pins

TRIG\_PIN = machine.Pin(2, machine.Pin.OUT)

ECHO\_PIN = machine.Pin(3, machine.Pin.IN)

BUZZER\_PIN = machine.Pin(4, machine.Pin.OUT)

DHT\_PIN = machine.Pin(5)

LED\_PIN = machine.Pin(6, machine.Pin.OUT)

def distance\_measurement():

    # Trigger ultrasonic sensor

    TRIG\_PIN.on()

    time.sleep\_us(10)

    TRIG\_PIN.off()

    # Wait for echo to be HIGH (start time)

    while not ECHO\_PIN.value():

        pass

    pulse\_start = time.ticks\_us()

    # Wait for echo to be LOW (end time)

    while ECHO\_PIN.value():

        pass

    pulse\_end = time.ticks\_us()

    # Calculate distance

    pulse\_duration = time.ticks\_diff(pulse\_end, pulse\_start)

    distance = pulse\_duration / 58  # Speed of sound (343 m/s) divided by 2

    return distance

def read\_dht\_sensor():

    d = dht.DHT22(DHT\_PIN)

    d.measure()

    return d.temperature(), d.humidity()

buzz\_start\_time = None  # To track when the buzzer started

while True:

    dist = distance\_measurement()

    temp, humidity = read\_dht\_sensor()

    # Check if the distance is less than a threshold (e.g., 50 cm)

    if dist < 50:

        # Turn on the buzzer and LED

        BUZZER\_PIN.on()

        LED\_PIN.on()

        status = "Flooding Detected"

        buzz\_start\_time = time.ticks\_ms()

    elif buzz\_start\_time is not None and time.ticks\_diff(time.ticks\_ms(), buzz\_start\_time) >= 60000:  # 1 minute

        # Turn off the buzzer and LED after 1 minute

        BUZZER\_PIN.off()

        LED\_PIN.off()

        status = "No Flooding Detected"

    else:

        status = "No Flooding Detected"

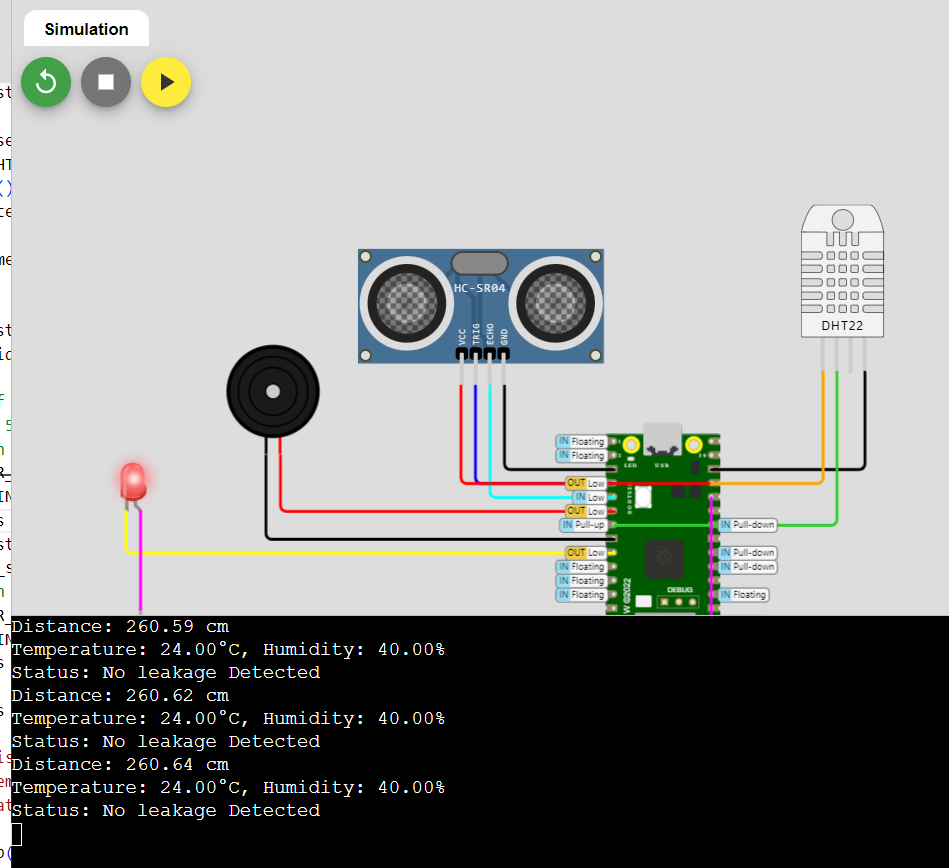
    print(f"Distance: {dist:.2f} cm")

    print(f"Temperature: {temp:.2f}°C, Humidity: {humidity:.2f}%")

    print("Status:", status)

    time.sleep(2)

Step8: Result of the above simulation

https://res-1.cdn.office.net/officeonline/we/s/hA3596C17DAD9A003_resources/1033/progress.gif 

In the above simulation , the humidity and temperature are calculated using the DHT22 and the distance between the maximum and the current present capacity is calculated using the ultrasonic distance sensor.

Thus , the chances of leakage is determined beforehand.

In this case , there is no leakage.

WEB APPLICATION:

/\*Water level monitoring system with the New Blynk app

[https://srituhobby.com](https://srituhobby.com/)

\*/

//Include the library files

#include <LiquidCrystal\_I2C.h>

#define BLYNK\_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

//Initialize the LCD display

LiquidCrystal\_I2C lcd(0x27, 16, 2);

char auth[] = "5\_Q6VTDFVGzauc0BBna2KVUNpn9RjMKL";//Enter your Auth token

char ssid[] = "";//Enter your WIFI name

char pass[] = "";//Enter your WIFI password

BlynkTimer timer;

// Define the component pins

#define trig D7

#define echo D8

#define LED1 D0

#define LED2 D3

#define LED3 D4

#define LED4 D5

#define LED5 D6

#define relay 3

//Enter your tank max value(CM)

int MaxLevel = 20;

int Level1 = (MaxLevel \* 75) / 100;

int Level2 = (MaxLevel \* 65) / 100;

int Level3 = (MaxLevel \* 55) / 100;

int Level4 = (MaxLevel \* 45) / 100;

int Level5 = (MaxLevel \* 35) / 100;

void setup() {

  Serial.begin(9600);

  lcd.init();

  lcd.backlight();

  pinMode(trig, OUTPUT);

  pinMode(echo, INPUT);

  pinMode(LED1, OUTPUT);

  pinMode(LED2, OUTPUT);

  pinMode(LED3, OUTPUT);

  pinMode(LED4, OUTPUT);

  pinMode(LED5, OUTPUT);

  pinMode(relay, OUTPUT);

  digitalWrite(relay, HIGH);

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

  lcd.setCursor(0, 0);

  lcd.print("Water level");

  lcd.setCursor(4, 1);

  lcd.print("Monitoring");

  delay(4000);

  lcd.clear();

  //Call the functions

  timer.setInterval(100L, ultrasonic);

}

//Get the ultrasonic sensor values

void ultrasonic() {

  digitalWrite(trig, LOW);

  delayMicroseconds(4);

  digitalWrite(trig, HIGH);

  delayMicroseconds(10);

  digitalWrite(trig, LOW);

  long t = pulseIn(echo, HIGH);

  int distance = t / 29 / 2;

  int blynkDistance = (distance - MaxLevel) \* -1;

  if (distance <= MaxLevel) {

    Blynk.virtualWrite(V0, blynkDistance);

  } else {

    Blynk.virtualWrite(V0, 0);

  }

  lcd.setCursor(0, 0);

  lcd.print("WLevel:");

  if (Level1 <= distance) {

    lcd.setCursor(8, 0);

    lcd.print("Very Low");

    digitalWrite(LED1, HIGH);

    digitalWrite(LED2, LOW);

    digitalWrite(LED3, LOW);

    digitalWrite(LED4, LOW);

    digitalWrite(LED5, LOW);

  } else if (Level2 <= distance && Level1 > distance) {

    lcd.setCursor(8, 0);

    lcd.print("Low");

    lcd.print("      ");

    digitalWrite(LED1, HIGH);

    digitalWrite(LED2, HIGH);

    digitalWrite(LED3, LOW);

    digitalWrite(LED4, LOW);

    digitalWrite(LED5, LOW);

  } else if (Level3 <= distance && Level2 > distance) {

    lcd.setCursor(8, 0);

    lcd.print("Medium");

    lcd.print("      ");

    digitalWrite(LED1, HIGH);

    digitalWrite(LED2, HIGH);

    digitalWrite(LED3, HIGH);

    digitalWrite(LED4, LOW);

    digitalWrite(LED5, LOW);

  } else if (Level4 <= distance && Level3 > distance) {

    lcd.setCursor(8, 0);

    lcd.print("High");

    lcd.print("      ");

    digitalWrite(LED1, HIGH);

    digitalWrite(LED2, HIGH);

    digitalWrite(LED3, HIGH);

    digitalWrite(LED4, HIGH);

    digitalWrite(LED5, LOW);

  } else if (Level5 >= distance) {

    lcd.setCursor(8, 0);

    lcd.print("Full");

    lcd.print("      ");

    digitalWrite(LED1, HIGH);

    digitalWrite(LED2, HIGH);

    digitalWrite(LED3, HIGH);

    digitalWrite(LED4, HIGH);

    digitalWrite(LED5, HIGH);

  }

}

//Get the button value

BLYNK\_WRITE(V1) {

  bool Relay = param.asInt();

  if (Relay == 1) {

    digitalWrite(relay, LOW);

    lcd.setCursor(0, 1);

    lcd.print("Motor is ON ");

  } else {

    digitalWrite(relay, HIGH);

    lcd.setCursor(0, 1);

    lcd.print("Motor is OFF");

  }

}

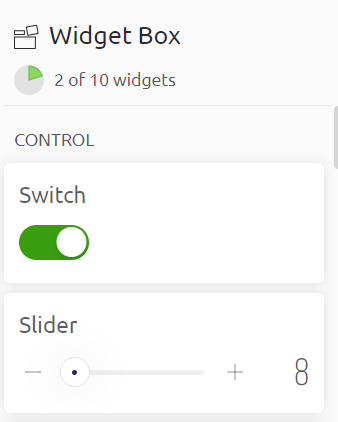
void loop() {

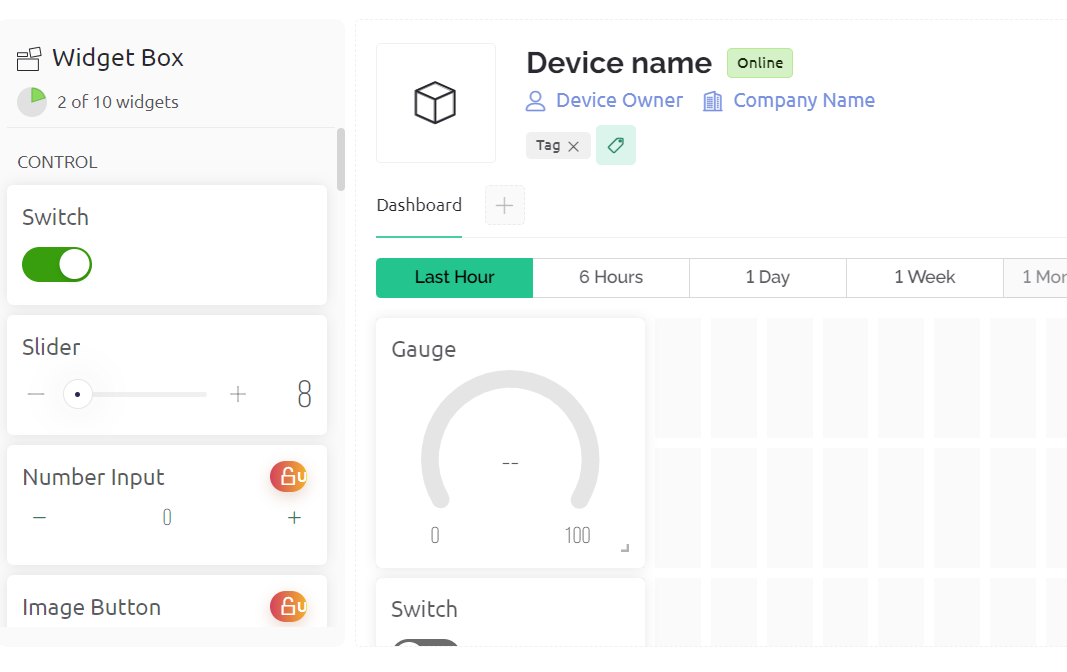
  Blynk.run();//Run the Blynk library

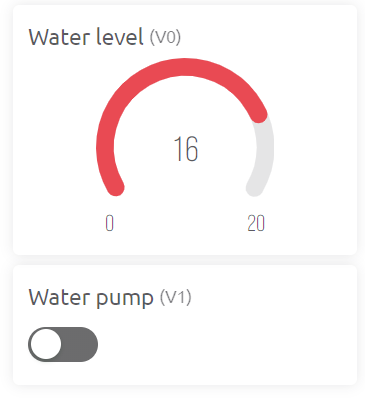
  timer.run();//Run the Blynk timer

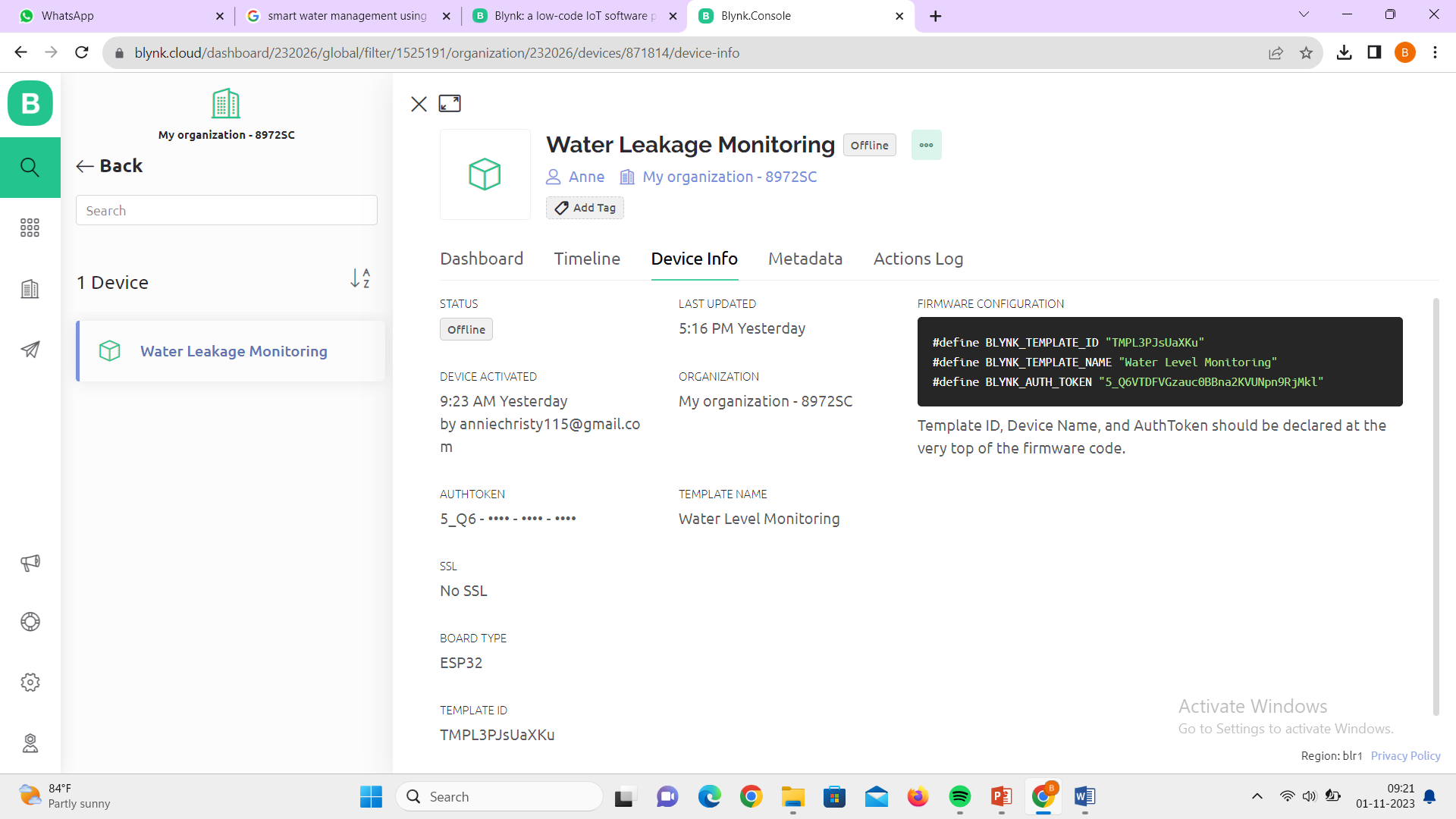
}

OUTPUT:









**THANK YOU**