**IBM NAAN MUDHALVAN**

**SKILL UP**

**PROJECT TITLE: SMART WATER SYSTEM**

**COLLEGE: PERI INSTITUTE OF TECHNOLOGY**

**DEPT: ELECTRONICS AND COMMUNICATION ENGINEERING**

**DOMAIN: INTERNET OF THINGS (IOT)**

**Submitted By**

**Bhavan S D (411521106012)**

* 1. **Abstract:**
* Water is one of the essential parts of life. Water pollution is one of the big problems to the world. In order to ensure the safe supply of the drinking and useful water for different purposes like agricultural, the water should be monitored. This document presents a design of a low cost system for real time monitoring of the water quality and quantity of water in IOT (internet of things). The system having of several sensors is used to measuring physical of the water. The parameters flow sensor of the water can be measured. The measured values from the sensors can be processed by the controller. The Arduino model can be used as a controller. Finally, the sensor data can be shown on internet using WI-FI system. A cloud server was configured as data saving and analysis. This data can be used in future research and development.
  1. **Introduction:**
* Currently drinking water is very prized for all the humans. In recent times water levels are very low and water in the lakes are going down. So its too important to find the solution for water monitoring & control system. IoT is a solution. In recent days, development in computing and electronics technologies have triggered Internet of Things technology . This paper present a low cost water monitoring system, which is a solution for the water wastage and water quality. Microcontrollers and sensors are used for that system. Ultrasonic Sensor is used to measuring water level. The other parameters like pH, TDS, and Turbidity of the water can be calculated using different corresponding sensors. This system use the flow sensor which can measure the water flow and if the necessary quantity of water flow through the pipe then water flow can be stopped automatically. The calculated values from the sensors can be processed by the Microcontrollers and uploaded to the internet through the Wi-Fi module (ESP 8266).
  1. **Project Definition:**
* The project involves implementing IoT sensors to monitor water consumption in public places such as parks and gardens. It employs advanced technology to monitor and provide immediate insights into water usage. This empowers individuals and industries to make informed decisions, leading to more efficient and sustainable water practices.
  1. **Objective:**
* The objective of this project is to promote water conservation by providing real-time water consumption data to the public. This empowers individuals, communities, and industries to make informed decisions and take proactive measures to reduce water wastage and improve overall efficiency in water usage. By leveraging technology and data transparency, the project aims to foster a culture of responsible water management for a more sustainable future.
  1. **IoT Sensor Design:**
* In this project many components such as sensors, modules , power sources and so on are used
* **Sensors:**
* Flow Sensor
* Ultrasonic sensor
* **Connectivity:**
* Wi-fi module
* **Power Source:**
* Battery / Solar
* **Data Processing:**
* Data processing will send raw data to a central processing unit..

PHASE 3

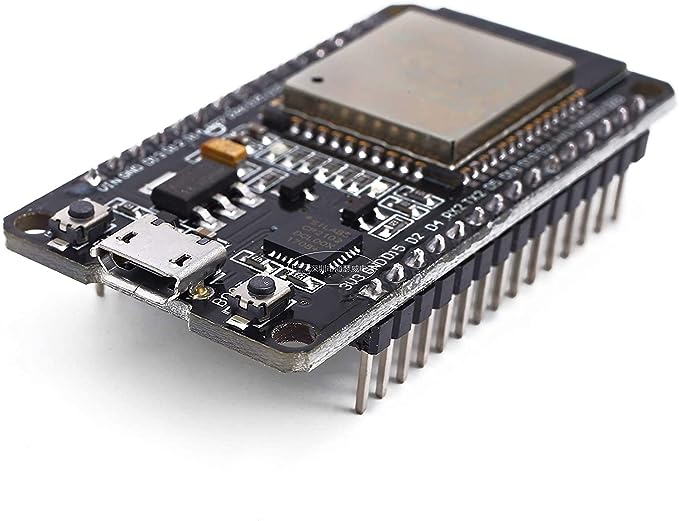
DEVELOPMENT-1

**Components:**

* **Smart Water Meters:**
* Install smart water meters at key water outlets in parks and gardens. These meters should be capable of measuring water flow rates and sending data to a central processor.
* **ESP32 Processor:**
* Use ESP32 microcontrollers as the central processing units. These microcontrollers are equipped with built-in Wi-Fi capabilities, making them suitable for collecting and transmitting data to the internet.
* **IoT Sensors:**
* Attach IoT sensors to the smart water meters to monitor water usage in real-time. These sensors can capture data like water flow rate, temperature, and humidity, and transmit it to the ESP32.
* **Internet Connectivity:**
* Set up a Wi-Fi network in the parks and gardens to ensure seamless data transmission from the ESP32 processors to the internet.
* **Data Storage and Processing:**
* Utilize cloud-based or on-premises servers for storing and processing the data from the IoT sensors. You may use platforms like AWS, Azure, or Google Cloud for this purpose.
* **Public Platform:**
* Develop a public-facing web application or mobile app that displays real-time water consumption data. This platform should be user-friendly and accessible to the public.
* **User Awareness Campaign:**
* Promote the public platform to encourage awareness about water conservation. Educate the public about the importance of water conservation and provide tips on how to reduce water consumption.

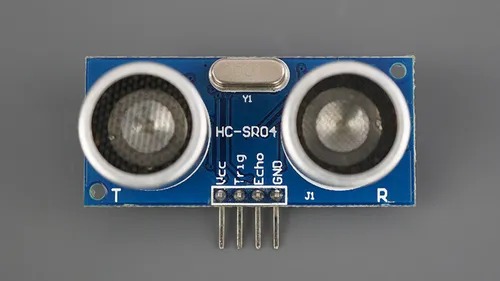
Above mentioned are general components the actual components I am going to use are

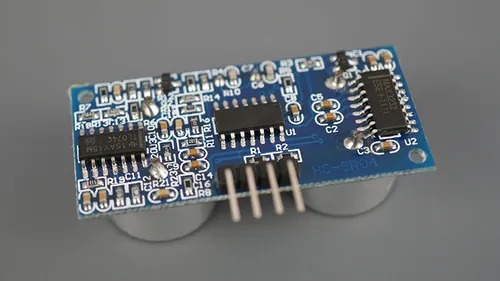
**ESP32 Processor**



The ESP32 is a dual-core microcontroller developed by Espressif Systems, ideal for IoT applications. It features built-in Wi-Fi and Bluetooth support, making it easy to connect to the internet and communicate with other devices. With low-power modes and a wide range of GPIO pins, it suits both battery-operated and energy-efficient devices. The ESP32 offers various interfaces, including ADC for sensor data, UART, SPI, and I2C for communication with other peripherals. It boasts security features like secure boot and flash encryption. A rich development ecosystem, with a strong community, simplifies the development process. The ESP32 supports real-time operating systems and OTA updates, providing flexibility for complex multitasking applications. It's cost-effective, making it a popular choice for a variety of IoT and embedded systems projects.

**Ultrasonic sensor hc-sr04**

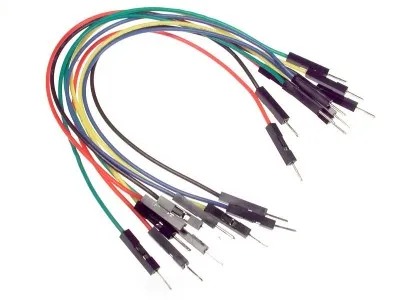


The HC-SR04 is a commonly used ultrasonic sensor module designed for distance measurement. It employs ultrasonic technology to calculate distances, with a typical range of 2 cm to 4 meters and an accuracy of about 3 mm. The module consists of a transmitter and receiver, triggered by a short pulse to measure distances based on the time it takes for the ultrasonic wave to bounce back. The HC-SR04 is easily interfaced with microcontrollers like Arduino or Raspberry Pi, requiring just four pins for operation. It's widely employed in applications such as obstacle avoidance in robotics, liquid level detection, and parking assistance systems. The sensor offers non-contact distance measurement, is cost-effective, and popular among DIY enthusiasts and hobbyists. However, it may have limitations in measuring very small or irregularly shaped objects and can be affected by environmental conditions like temperature and humidity.

9v battery



Jumper wire



USB WIRE FOR PROGRAM DUMPING PROCESS IN CHIP





Step by step process

### **Step 1: Gather Hardware and Set Up Smart Meters**

* Acquire the necessary hardware, including ESP32 microcontrollers, smart water meters, and IoT sensors.
* Install the smart water meters at the water outlets in the parks and gardens.

### **Step 2: Configure the ESP32**

* Program the ESP32 microcontrollers to interface with the smart water meters and IoT sensors. Use the Arduino IDE, PlatformIO, or your preferred development environment.
* Integrate the Blynk library into your ESP32 code to enable communication with the Blynk platform. Make sure to use the unique Blynk Auth Token for your project.

### **Step 3: Set Up IoT Sensors**

* Attach IoT sensors to the smart water meters to monitor relevant data, such as water flow rates, temperature, and humidity. Configure these sensors to provide data to the ESP32.

### **Step 4: Establish Internet Connectivity**

* Set up a Wi-Fi network in the parks and gardens to ensure the ESP32 processors can connect to the internet.

### **Step 5: Develop the Blynk Interface**

* Create a Blynk project in the Blynk app builder that will serve as the public platform for displaying water consumption data.
* Design a user-friendly interface within the Blynk app, using widgets to visualize data, such as gauges, graphs, and labels.

### **Step 6: Send Data to Blynk**

* Modify your ESP32 code to periodically collect data from the sensors and smart meters.
* Use the Blynk library to send this data to the Blynk project you created. Ensure the data is properly formatted and transmitted using the Blynk Auth Token.

### **Step 7: Data Storage and Processing**

* Set up a cloud-based or on-premises server to receive, process, and store the data from the ESP32s. You can use platforms like AWS, Azure, or Google Cloud for this purpose.

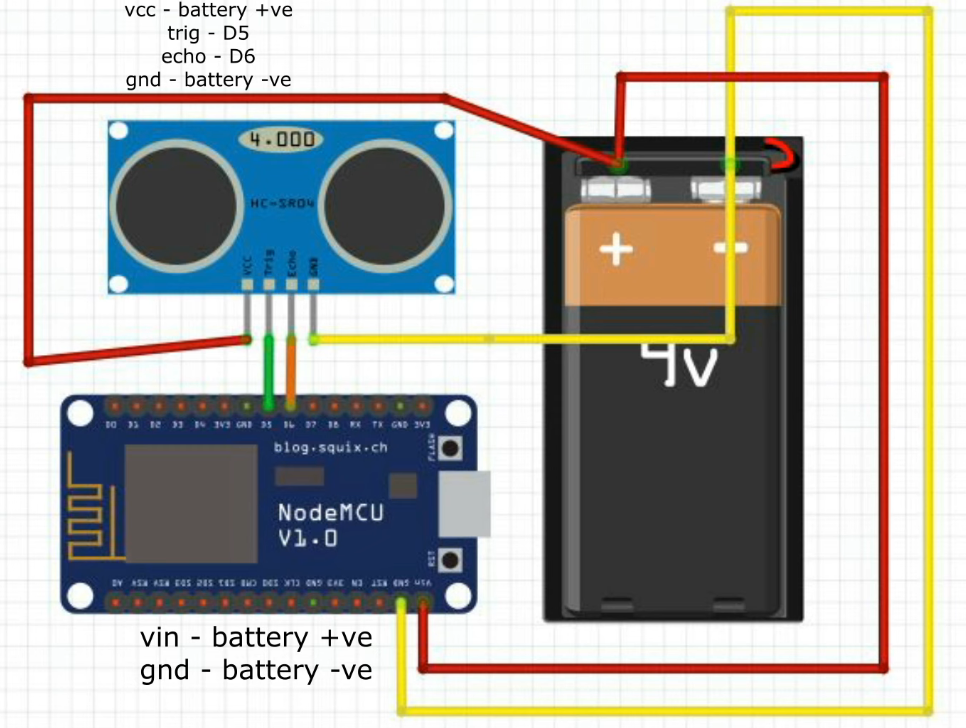
### **Step 8: User Awareness and Engagement**

* Launch an awareness campaign to promote the public platform. Encourage users to access the Blynk app to view real-time water consumption data and learn about water conservation efforts.
* Configure notifications and alerts within Blynk to engage users when unusual water consumption patterns are detected.

### **Step 9: Monitor and Maintain**

* Continuously monitor the system's performance and the accuracy of the data. Make necessary adjustments and conduct regular maintenance.
* Update the Blynk project interface and the ESP32 code as needed to improve the system's functionality.

Circuit



Creating a Blynk interface



**Create a Blynk Account:**

* If you haven't already, start by creating an account on the Blynk platform.
* **Create a Blynk Project:**
* After logging in, create a new project within your Blynk account.
* **Select Hardware Model:**
* In your Blynk project settings, choose the hardware model you'll be using. For an ESP32, you can select "ESP32 Dev Board" or a similar option.
* **Generate Blynk Auth Token:**
* Each project will have a unique authentication token. Click on the "Device Info" button to generate a Blynk Auth Token for your project. You'll need this token to connect your hardware to the Blynk project.
* **Add Widgets:**
* Customize your project by adding widgets. Blynk offers a variety of widgets to choose from, including buttons, sliders, graphs, displays, and more. Each widget serves a specific purpose, such as displaying data or controlling devices.
* **Configure Widgets:**
* Configure each widget according to your project's requirements. This includes setting the widget type, pin, data range, labels, and more. Widgets are linked to your hardware's pins, so they should match your ESP32 pin configuration.
* **Design the User Interface:**
* Use the Blynk app builder to design the user interface. Drag and drop widgets onto the canvas, arrange them as desired, and set their properties.
* **Set Data Frequency and Widgets Behavior:**
* Define how often data should be updated for widgets that display information. Widgets like graphs can be configured to display historical data. Buttons and sliders can control hardware devices.
* **Write Code for Hardware:**
* In your ESP32 or hardware's code, incorporate the Blynk library. Use your Blynk Auth Token to establish a connection between your hardware and the Blynk project. Configure your code to send data to the specified pins that correspond to the widgets in your Blynk project.
* **Test Your Project:**
* Upload the code to your hardware and monitor the data and control functions through the Blynk app. Ensure that the data is being received and that the widgets function as expected.
* **Fine-Tune and Iterate:**
* Refine your Blynk interface as needed. You can modify the widget settings or layout to better suit your project's requirements. Iterate and test until the interface aligns with your project's goals.
* **Public Access :**
* If your project is intended for public access, you can share your Blynk project by making it accessible with a QR code or a unique link. Users can download the Blynk app and access your project by scanning the QR code or clicking the link.
* **Secure Your Project (if needed):**
* If your project deals with sensitive data, ensure data security and access control by setting up appropriate security measures within Blynk.

After creating this Blynk interface we are going to spread it to public use and make sure they are aware of it

Program

#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

// Wi-Fi and Blynk credentials

char ssid[] = "YourSSID";

char pass[] = "YourPassword";

char auth[] = "YourAuthToken";

// Pin configurations

#define TRIG\_PIN 14 // GPIO pin for the trigger pin of the ultrasonic sensor

#define ECHO\_PIN 12 // GPIO pin for the echo pin of the ultrasonic sensor

void setup() {

Serial.begin(115200);

Blynk.begin(auth, ssid, pass);

pinMode(TRIG\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

}

void loop() {

Blynk.run();

// Ultrasonic sensor reading

long duration, distance;

digitalWrite(TRIG\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIG\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG\_PIN, LOW);

duration = pulseIn(ECHO\_PIN, HIGH);

distance = (duration / 2) / 29.1; // Convert to centimeters

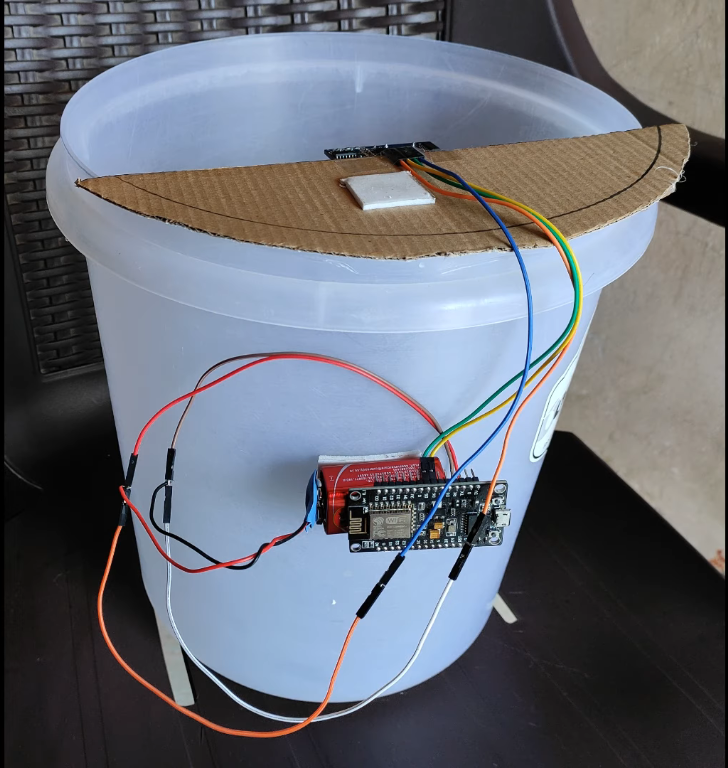
// Send the distance data to Blynk

Blynk.virtualWrite(V1, distance);

delay(1000); // Update frequency

}

Future work in development 2



Conclusion

The implementation of IoT sensors, smart meters, and the Blynk platform for monitoring water consumption in public places provides a valuable solution for water conservation. By utilizing the ESP32 processor, data from smart meters is efficiently collected, transmitted, and made accessible to the public. This project fosters awareness, accountability, and responsible water usage, contributing to sustainable environmental practices and community engagement in preserving a vital resource.