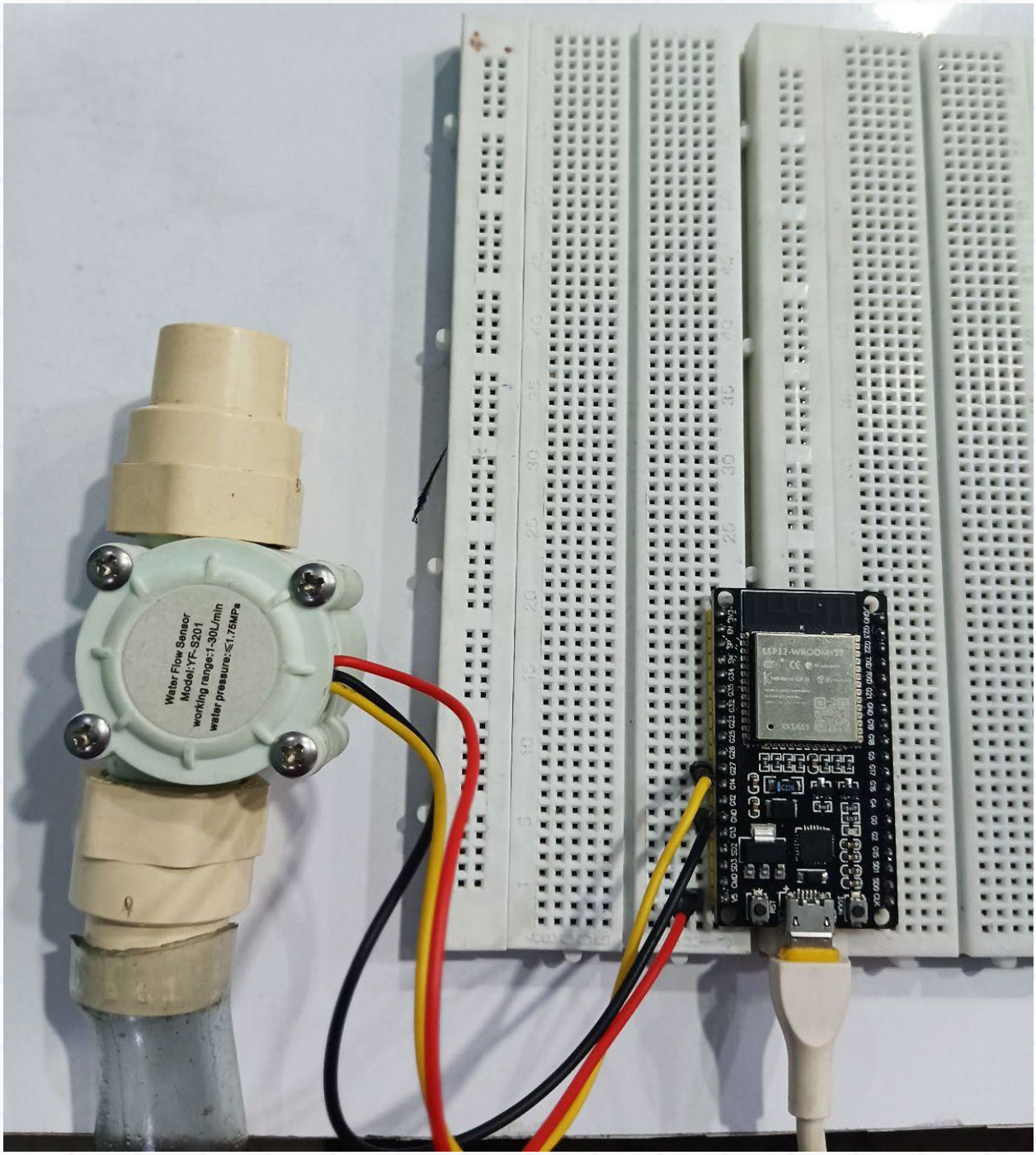
# Build a Water Flow Meter




IoT Water Meter

# Index

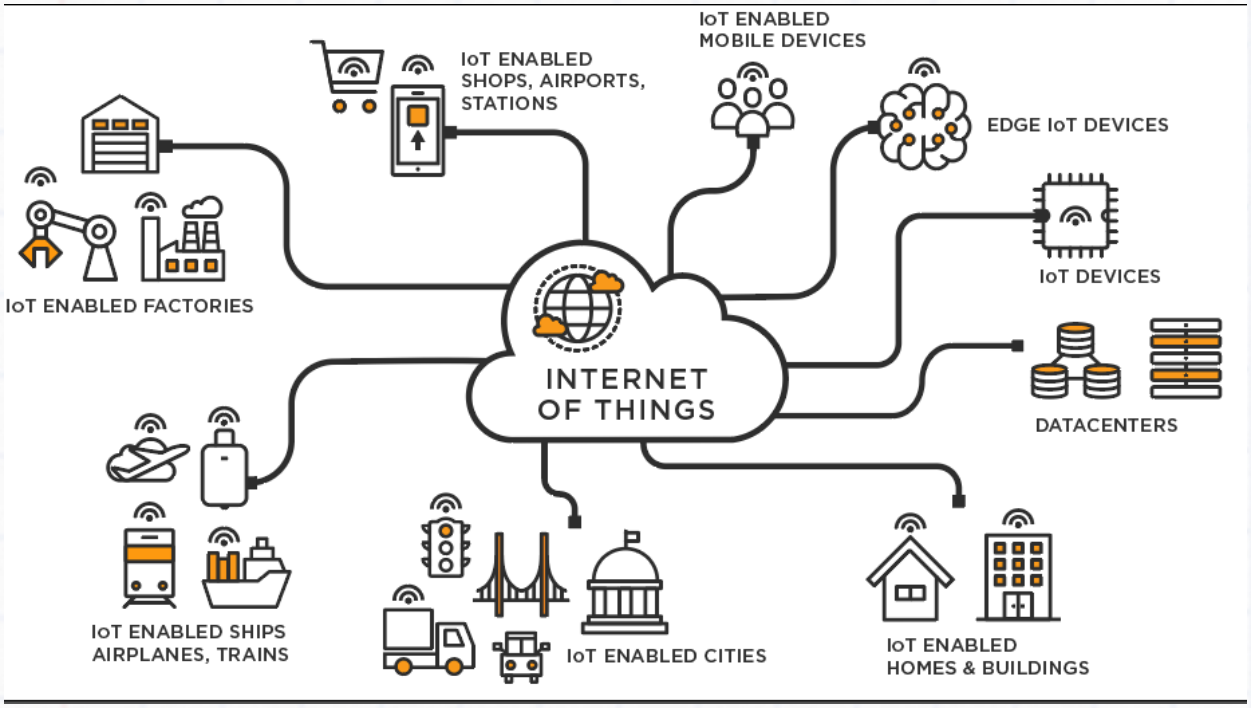
# Aim

Build an IoT (Internet of Things) Water Flow Meter that measures water flow rate and consumption in real-time and track this data remotely on a web IoT platform.

# Concept

### What is IoT?

IoT is a system of interconnected computing devices that use internet protocols to communicate and transfer data. This allows remote devices to communicate with each other without the need for human involvement




### How this project works

In this project, a flow sensor measures the flow rate of any fluid passing through. This value is sent to the ESP32 board using an External interrupt mechanism - which constantly measures the flow of the fluid.
Through the code written to the ESP32, we calculate the values of the Flow rate and Water accumulated.

The ESP32 Wifi module then sends the data to a Ubidots cloud server using an API. The water flow data can then be viewed in real-time on the Ubidots web dashboard.

## Components

1. ESP32 module
2. YF-S201 Flow sensor
3. Breadboard
4. Jumper wires
5. 1 USB cable - 1 for powering the Esp32 from the laptop to the Esp32.
6. PVC tube

Esp 32 Module

YF-S201 Flow sensor

PVC tube

Jumper Wires

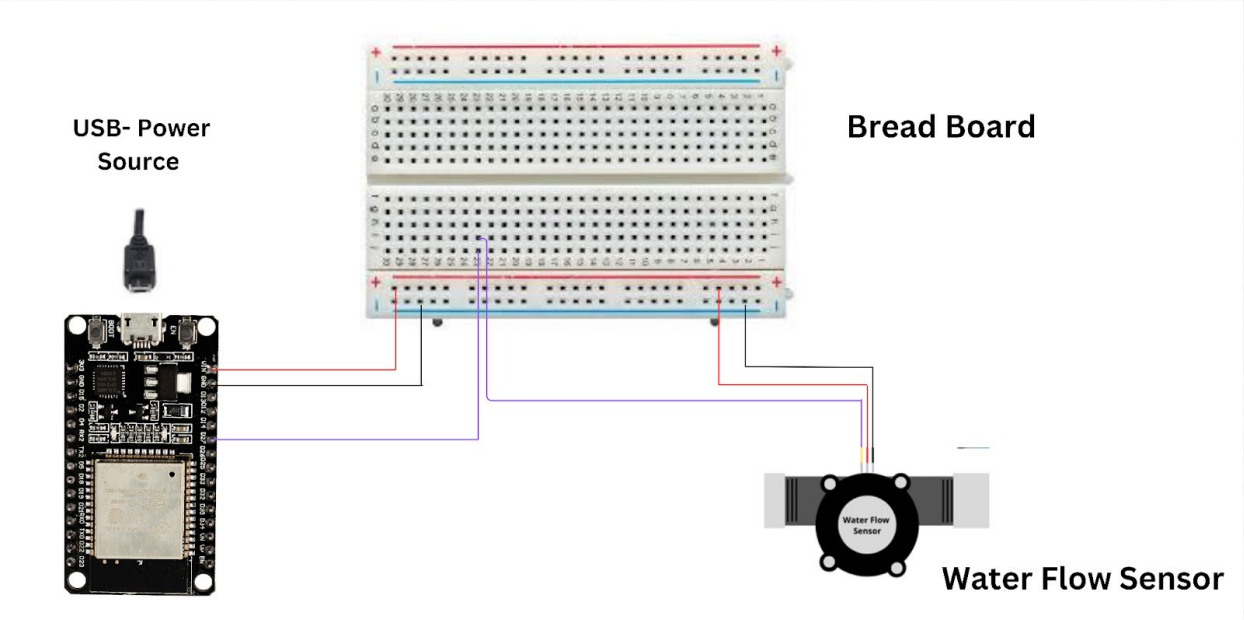
USB

Breadboard

## Connections

**Safety tip**: Always ensure that the connections to the components are correct and completed before connecting the power supply to the Esp32.

Circuit Diagram




Follow the detailed steps in the following pages to complete the circuit.

**NOTE**: Before starting the connections, verify that all the jumper wires are working using a multimeter. Also ensure that the connections are strong, or else the setup may not work.

**Detailed Connection Steps**

Take 3 female-to-male jumper wires and connect them to the pins (5V, GND, D27) of the ESP 32 Board. Connect the other ends as per the below connections:
The YF-S201 Flow sensor has 3 wires RED, BLACK, and YELLOW

1. RED to 5v on ESP32
2. BLACK to GND on ESP32
3. YELLOW to D27 on ESP32

You are now ready to work on the software for the project.

# Software

Launching the IDE for our project

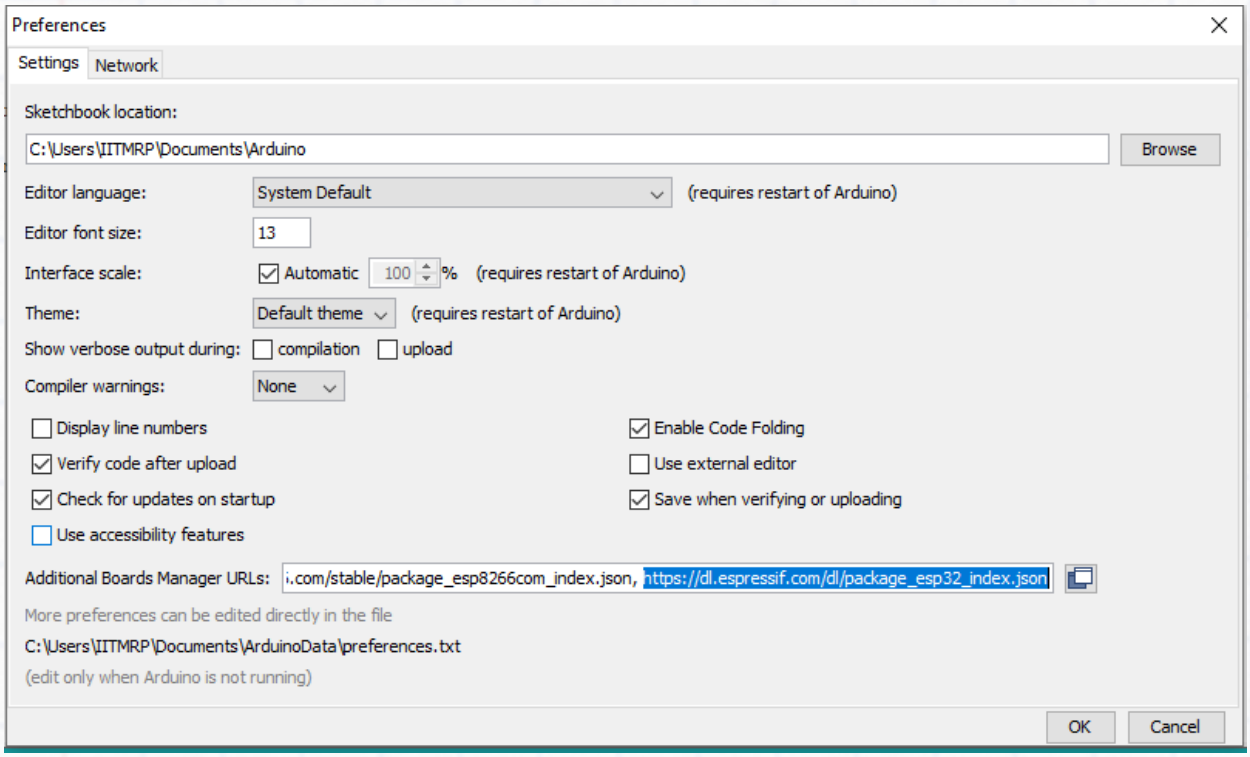
**1. Install the Arduino IDE**

If you haven't already installed the Arduino IDE, download and install it from the official Arduino website.

**2. Install the ESP32 Board in Arduino IDE**

1. Open the Arduino IDE.
2. Go to File > Preferences.
3. In the "Additional Board Manager URLs" field, add the following URL:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json




4.Go to Tools > Board > Boards Manager.

5. Search for ESP32 and install the ESP32 package. This will take some time to install.

**3. Install Required Libraries**

1.Go to Sketch > Include Library > Manage Libraries.

2. Inside the extracted **ESP32_Flow_Meter** folder, you will find 2 folders named **esp32-mqtt-main** & **pubsubclient-master**. Copy them and paste them inside the 'Libraries' folder located in Documents > Arduino > Libraries.

**4. Connect Your ESP32 Board**

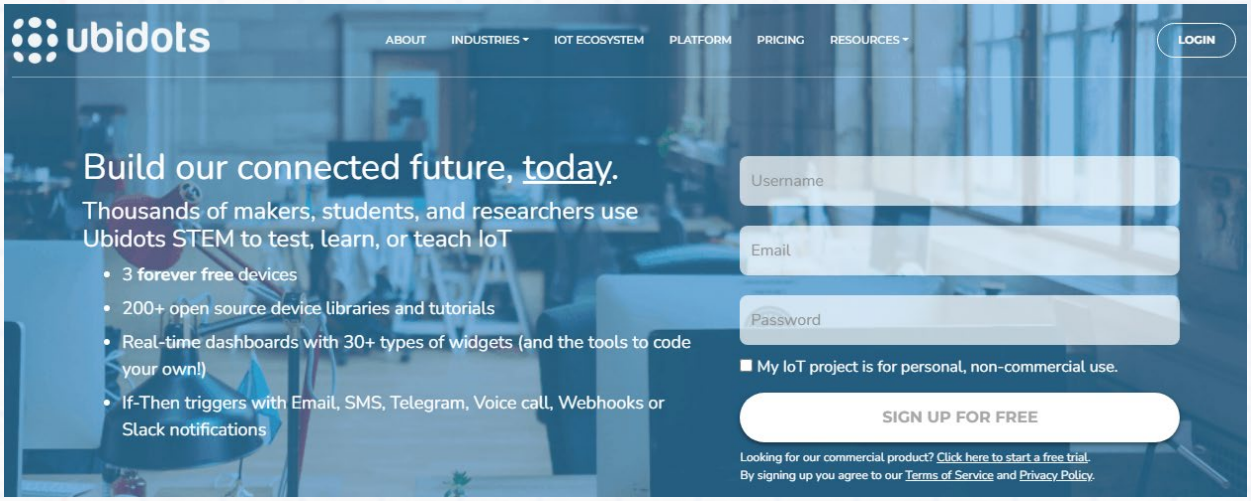
1. Connect your ESP32 board to your computer using a USB cable.
2. Go to Tools > Board > ESP32 Dev Module.
3. Go to Tools > Port and select the COM port to which your ESP32 is connected.

**5. Prepare the Hardware**

Connect your YF-S201 Flow sensor to the ESP32. All Three wires in the flow sensor are connected to the Esp32.

**6. Ubidots Setup**

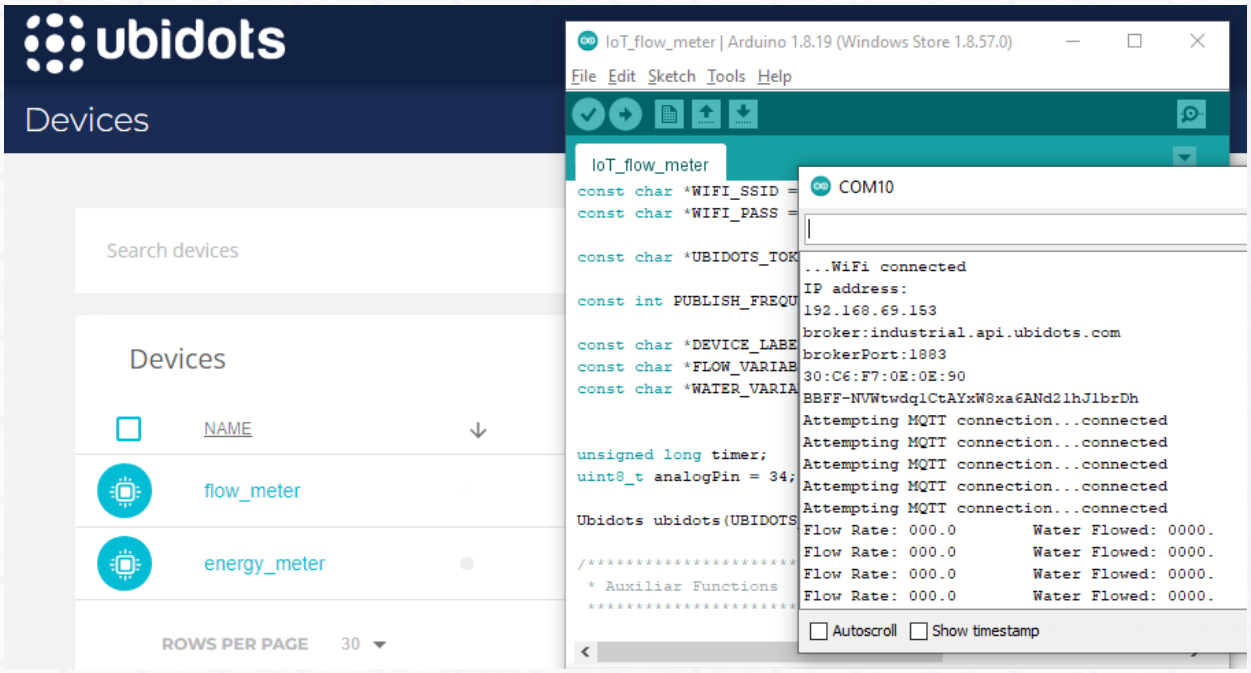
- For monitoring the water flow readings, we will be using the Ubidots IoT Platform. Go to Ubidots.com/stem/ and sign up for free.
- On the Ubidots website, click on your profile icon and select API credentials.
- Copy the 'Default token' on the top right part of the page and paste it into the UBIDOTS_TOKEN variable, which is immediately below the Wifi variables in the Arduino code.



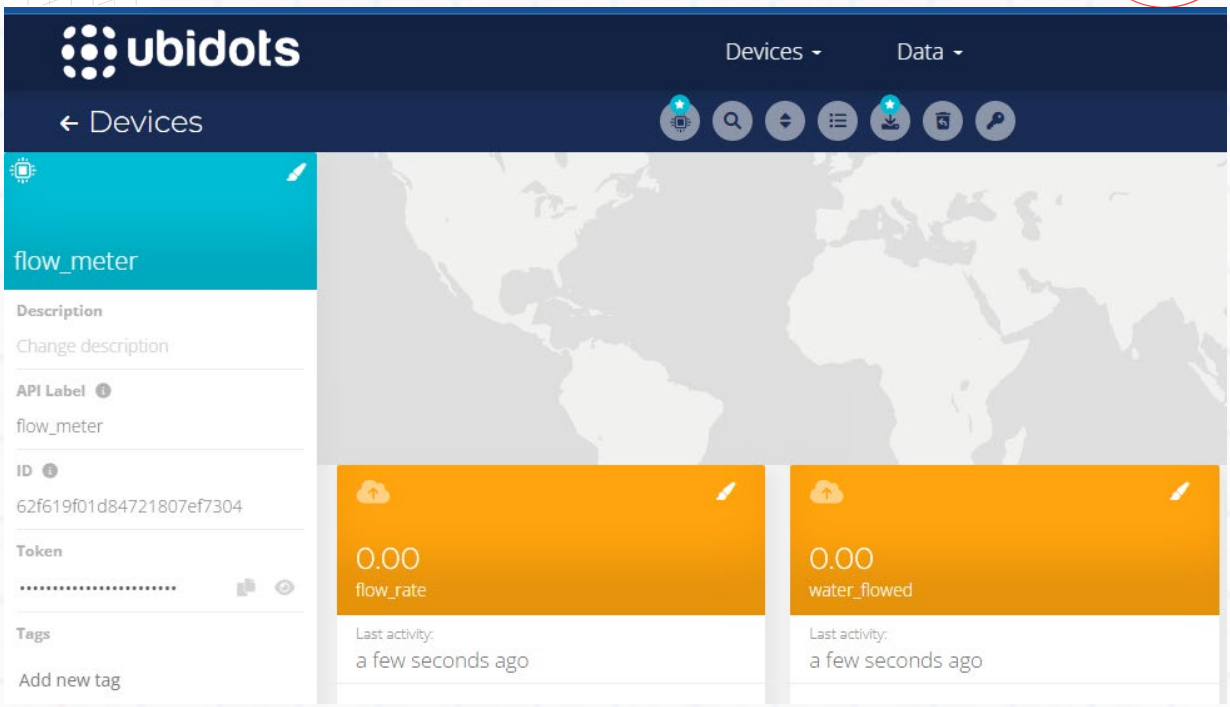

**7. Upload the Code**

- Copy the provided code into a new sketch in the Arduino IDE.
- Fill in your wifi connection's name and password within the empty quotation marks next to the WIFI_SSID and WIFI_PASS variables. Ensure the Wifi network has no firewalls, else the ESP32 won't connect to it.

- Click the ‘Verify’ button on the Arduino IDE - shown by a tick symbol. Once you receive the “Done compiling” message, click the ‘Upload’ button, shown as a right arrow near the ‘Verify’ button. Wait until the “Done uploading” message appears.
- After uploading the code, Click the magnifying glass icon on the top-right of the Arduino IDE. It will open the Serial Monitor. If the connection was successful.




- Now go to the Devices page of the Ubidots website. You will find a device named flowmeter containing variables flowrate and water flowed.




## Code

“Open the .ino file which is attached in the folder and write a code by the below instruction.”

**Explanation:**

**Includes and Constants**:

- #include "**UbidotsEsp32Mqtt.h**": Includes the Ubidots MQTT library for ESP32.

Defines various constants like **LED_BUILTIN**, **SENSOR**, **Wi-Fi credentials**, **Ubidots token**, and **labels**.

- Defines timing variables and flow measurement variables.
- **IRAM_ATTR pulseCounter():** Interrupt service routine to increment **pulseCount**.

**Ubidots Setup:**

- Creates an instance of the **Ubidots** class using the Ubidots token.

- Defines the **callback** function to handle incoming MQTT messages.

```
#include "UbidotsEsp32Mqtt.h"  // Include the Ubidots ESP32 MQTT library

/****************************************
 * Define Constants
 ****************************************/
#define LED_BUILTIN 2  // Define the built-in LED pin for ESP32
#define SENSOR 27  // Define the sensor pin

long currentMillis = 0;  // Variable to store current time
long previousMillis = 0;  // Variable to store previous time
int interval = 1000;  // Interval for measuring flow rate (1 second)
boolean ledState = LOW;  // Variable to store LED state
float calibrationFactor = 4.5;  // Calibration factor for the flow sensor
volatile byte pulseCount;  // Variable to store pulse count (volatile as it's modified in interrupt)
byte pulse1Sec = 0;  // Pulse count in one second
float flowRate;  // Flow rate variable
unsigned int flowMilliLitres;  // Flow in millilitres
unsigned long totalMilliLitres;  // Total flow in millilitres

void IRAM_ATTR pulseCounter()
{
  pulseCount++;  // Increment pulse count (interrupt service routine)
}

const char *WIFI_SSID = "";  // Wi-Fi SSID
const char *WIFI_PASS = "";  // Wi-Fi password
const char *UBIDOTS_TOKEN = "";  // Ubidots TOKEN

const int PUBLISH_FREQUENCY = 100; // Frequency to publish data in milliseconds

const char *DEVICE_LABEL = "Flow_meter";  // Device label for Ubidots
const char *FLOW_VARIABLE_LABEL = "Flow_rate";  // Flow rate variable label for Ubidots
const char *WATER_VARIABLE_LABEL = "Water_flowed";  // Water flowed variable label for Ubidots

unsigned long timer;
uint8_t analogPin = 34; // Pin used to read data from GPIO34 ADC_CH6

Ubidots ubidots(UBIDOTS_TOKEN);  // Create an instance of the Ubidots class
```

Copy this code and paste it on the below Auxiliary function comment line

```
void callback(char *topic, byte *payload, unsigned int length)
{
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  {
    Serial.print((char)payload[i]);  // Print each character of the payload
  }
  Serial.println();
}
```

**Global Variables for Publishing:**

- Defines **flow** and **water** variables for publishing to Ubidots.

**Setup Function:**

- **void setup()**: Initializes serial communication.
- **Serial.begin(9600)** (line 50): Starts serial communication at 9600 baud rate.
- **ubidots.connectToWifi(WIFI_SSID, WIFI_PASS)**: Connects to Wi-Fi.
- **ubidots.setCallback(callback)** : Sets the MQTT callback function.
- **ubidots.setup()** : Initializes the Ubidots client.
- **ubidots.reconnect()** : Reconnects to Ubidots.
- Initializes the timer with the current time.
- Sets the built-in LED and sensor pins.
- Initializes the pulse count and flow measurement variables.
- Attaches an interrupt to the sensor pin to count pulses.

Copy the code and paste it on the below main function comment line.

```
float flow = 0.0;  // Flow rate variable for publishing
float water = 0.0;  // Total water flowed variable for publishing

void setup()
{
  Serial.begin(9600);  // Start serial communication at 9600 baud rate
```

```
 // ubidots.setDebug(true);  // Uncomment to enable debug messages
 ubidots.connectToWifi(WIFI_SSID, WIFI_PASS);  // Connect to Wi-Fi
 ubidots.setCallback(callback);  // Set the MQTT callback function
 ubidots.setup();  // Initialize the Ubidots client
 ubidots.reconnect();  // Reconnect to Ubidots

 timer = millis();  // Initialize the timer with the current time
 pinMode(LED_BUILTIN, OUTPUT);  // Set the built-in LED pin as output
 pinMode(SENSOR, INPUT_PULLUP);  // Set the sensor pin as input with internal pull-up

 pulseCount = 0;  // Initialize pulse count
 flowRate = 0.0;  // Initialize flow rate
 flowMilliLitres = 0;  // Initialize flow in millilitres
 totalMilliLitres = 0;  // Initialize total flow in millilitres
 previousMillis = 0;  // Initialize previous time

 attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);  // Attach interrupt to the
sensor pin
}
```

**Loop Function:**

- **void loop()**: Main loop function.
- **if (!ubidots.connected()) { ubidots.reconnect(); }** : Reconnects to Ubidots if disconnected.
- c**urrentMillis = millis()** : Gets the current time.
- **if (currentMillis - previousMillis > interval)** : Checks if the interval has passed to calculate the flow rate.
- **pulse1Sec = pulseCount; pulseCount = 0;** Copies and resets the pulse count.
- Calculates the flow rate and converts it to millilitres.
- Updates the total flow and prints the flow rate and total flow.
- Updates the **flow** and **water** variables for publishing.
- Publishes the flow rate and total flow to Ubidots at the specified frequency.
- **ubidots.loop()**: Handles Ubidots client tasks.

Copy this code and paste it on inside the void loop

```
 if (!ubidots.connected())
 {
```

```
  ubidots.reconnect();  // Reconnect to Ubidots if disconnected
}

currentMillis = millis();  // Get the current time
if (currentMillis - previousMillis > interval)
{
  pulse1Sec = pulseCount;  // Copy the pulse count
  pulseCount = 0;  // Reset the pulse count

  // Calculate flow rate
  flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) / calibrationFactor;
  previousMillis = millis();  // Update previous time

  // Convert to millilitres
  flowMilliLitres = (flowRate / 60) * 1000;

  // Add to total
  totalMilliLitres += flowMilliLitres;

  // Print flow rate and total
  Serial.print("Flow rate: ");
  Serial.print(int(flowRate));  // Print the integer part of the flow rate
  Serial.print(" L/min");
  Serial.print("\t");  // Print tab space

  Serial.print("Output Liquid Quantity: ");
  Serial.print(totalMilliLitres);  // Print total millilitres
  Serial.print(" mL / ");
  Serial.print(totalMilliLitres / 1000);  // Print total litres
  Serial.println(" L");

  // Update values for publishing
  flow = flowRate;
  water = totalMilliLitres;

  // Publish values
  if ((millis() - timer) > (unsigned long)PUBLISH_FREQUENCY)
  {
    ubidots.add(FLOW_VARIABLE_LABEL, flow);  // Add flow rate to Ubidots payload
    ubidots.publish(DEVICE_LABEL);  // Publish to Ubidots
    delay(100);  // Short delay
```

```
    ubidots.add(WATER_VARIABLE_LABEL, water);  // Add water flowed to Ubidots payload
    ubidots.publish(DEVICE_LABEL);  // Publish to Ubidots
    delay(100);  // Short delay

    timer = millis();  // Reset timer
  }
}

ubidots.loop();  // Handle Ubidots client tasks

delay(10);  // Short delay
Serial.flush();  // Flush the serial buffer
delay(10);  // Short delay
```

**Now verify the code on Arduino**

**Then upload the code into the ESP32.**

## Tasks

The Flow Rate value in the Serial Monitor should read zero. If the Water Flowed value isn't initially zero, press the reset button on the ESP32. Now slowly blow into the water meter pipe and check whether the flowrate and water flowed variables change in the Serial Monitor. Go to the Ubidots Dashboard page - the meter gauges should reflect the change in values.

## Real-world Application

You have now learnt how to build an IoT based Water Flow Meter and are ready to apply it in real-world scenarios! This Water Meter can be used in several applications. Identify locations in your college or home where it can be installed - this could be a hand wash area, a distribution pipe of a water tank, or even a drinking water dispenser. One very simple and practical use-case is in an RO water purifier. Have you ever wondered how an RO water purifier works? Using Reverse Osmosis technology, it concentrates impurities and sediments in one portion of the water - which is wasted - leaving the purified portion for drinking. But do you have any idea how much water is wasted? Taking permission from your college authorities, locate an RO water purifier in

your college and install the Water Meter in the pipe that drains out the wastewater. It must be installed in such a way that the electronics are protected from water, heat or physical damage. Make sure that there is access to Wi-Fi in that area. Track the wastage quantities over different periods of time - a day, a week, a month… - and note these values. Additionally, you can build one more Water Meter or use the same to measure the amount of purified water consumed form the same RO system.

You will be surprised to know the amount of water wasted to purify just 1L of water! Make a video presenting the installation of the water meter and the results/insights you got about the RO system. Share it with the Build Club Community on the Discord Server! Also share your own unique applications Real-world Application of the IoT Water Meter!