PROJECT DEFINITION:

In this project we are going to measure the water consumption and going to measure the contamination level in the water by implementing IOT sensors. And these measured values will be used to know whether we can use the water and how much we can use to reduce the consumption in terms of to save the wastage of water. If the water is not contaminated, we can use it for general purpose but with the amount we needed and if the water was contaminated then we can use it for any other purposes. These are all done to promote the water conservation. These can be implemented anywhere like home, public places, Urban areas, Rural areas, Industry, Agriculture, etc. These data will be visible to anyone who implemented the sensor. If it is in public place, then all the public can see the data of water consumption and how clean it is and if it is in home then it can be seen by who implemented the sensor. This project includes defining objectives, designing the IoT sensor system, developing the data-sharing platform, and integrating them using IoT technology and Python.

A) PROJECT OBJECTIVES:

1. Real-Time Water Consumption Monitoring:

- Implement sensors and meters to provide real-time data on water consumption in the target area.
- Enable users to access and analyse their water usage patterns through a user-friendly interface.

1. Water Consumption Measurement:

- **Objective:** Develop a system to accurately measure and monitor water consumption levels in real-time.
- **Components:**
- IoT-enabled flow meters or water sensors.
- Microcontroller (e.g., Arduino, Raspberry Pi) for data processing.
- Connectivity modules (e.g., Wi-Fi, GSM) for data transmission.
- Cloud platform integration for data storage and analysis.

2. Real-time Data Visualization:

- **Objective:** Implement a user-friendly dashboard to visualize real-time water consumption data.
- **Components:**
- Web or mobile application for users to monitor consumption.
- Integration with IoT platform (e.g., ThingSpeak, Blynk) for data visualization.
- Historical data storage for trend analysis.

3. Water Contamination Detection:

- **Objective:** Integrate sensors to detect and measure water contamination levels.
- **Components:**

- Water quality sensors for detecting contaminants (e.g., pH, turbidity, conductivity).
- Microcontroller for processing sensor data.
- Alerts or notifications for abnormal contamination levels.

4. **Contamination Source Identification:**

- **Objective:** Implement a system to identify potential sources of water contamination.
- **Components:**
- Geospatial sensors or data to track contamination sources.
- Machine learning algorithms for pattern recognition.
- Integration with geographic information systems (GIS) for mapping.

5. **Alerts and Notifications:**

- **Objective:** Provide real-time alerts to users and relevant authorities in case of abnormal water consumption or contamination.
- **Components:**
- Automated alert systems through email, SMS, or mobile app notifications.
- Integration with emergency services or relevant authorities.

6.Data Analytics and Reporting:

- **Objective:** Implement data analytics for trend analysis and generate insightful reports.
- **Components:**
- Integration with analytics tools (e.g., MATLAB, Python) for in-depth analysis.
- Automated reporting features for periodic summaries.

7.Community Engagement and Education:

- **Objective:** Foster community awareness and engagement in water conservation practices.
- **Components:**
- Educational materials integrated into the system.
- Community forums or outreach programs.

1) Configure IOT sensors :

A) Magnetic Flow Sensor – For analysing water consumption.

B) pH Sensor – For analysing water contamination level and quality.

2) Configuring Magnetic Flow Sensor :

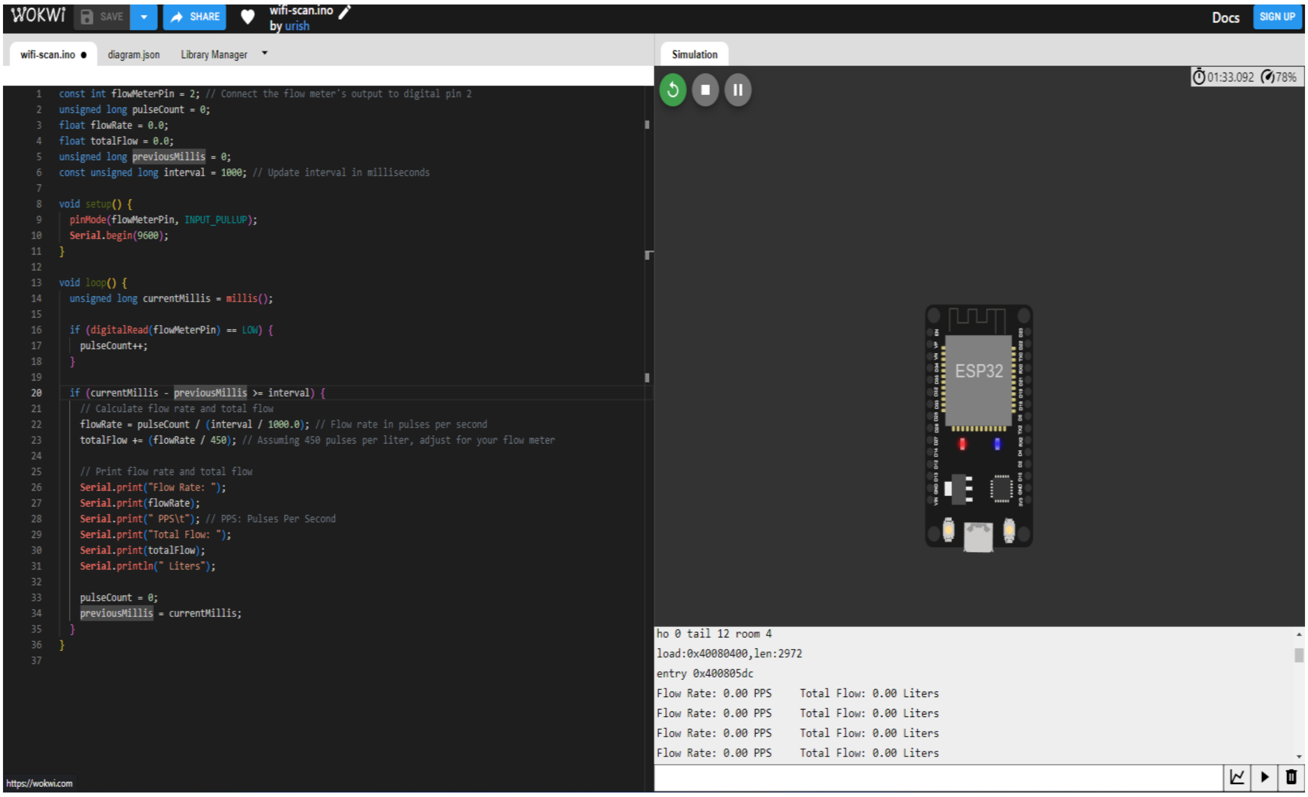
A) Get a Magnetic Flow Sensor.

B) Configure the flow sensor based on our need.

II) Configuring pH Sensor :

A) Get a pH Sensor.

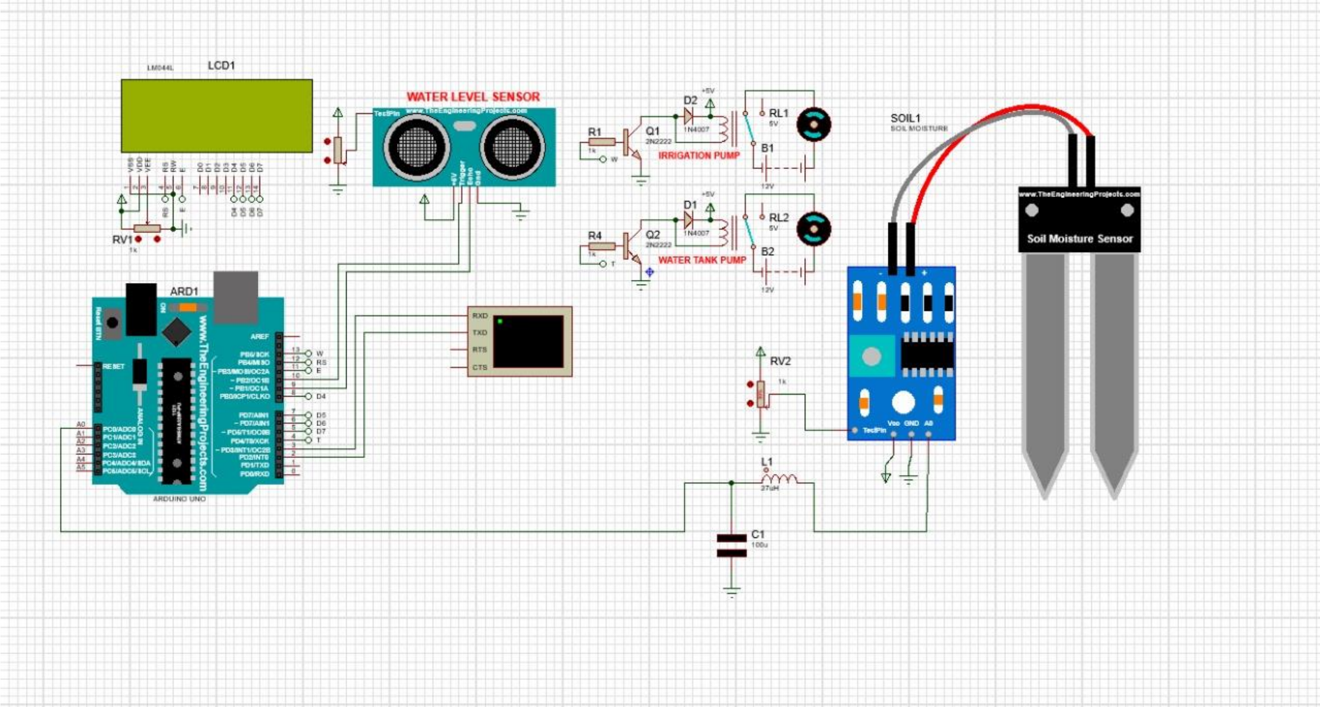
B) Configure the pH sensor based on our need.

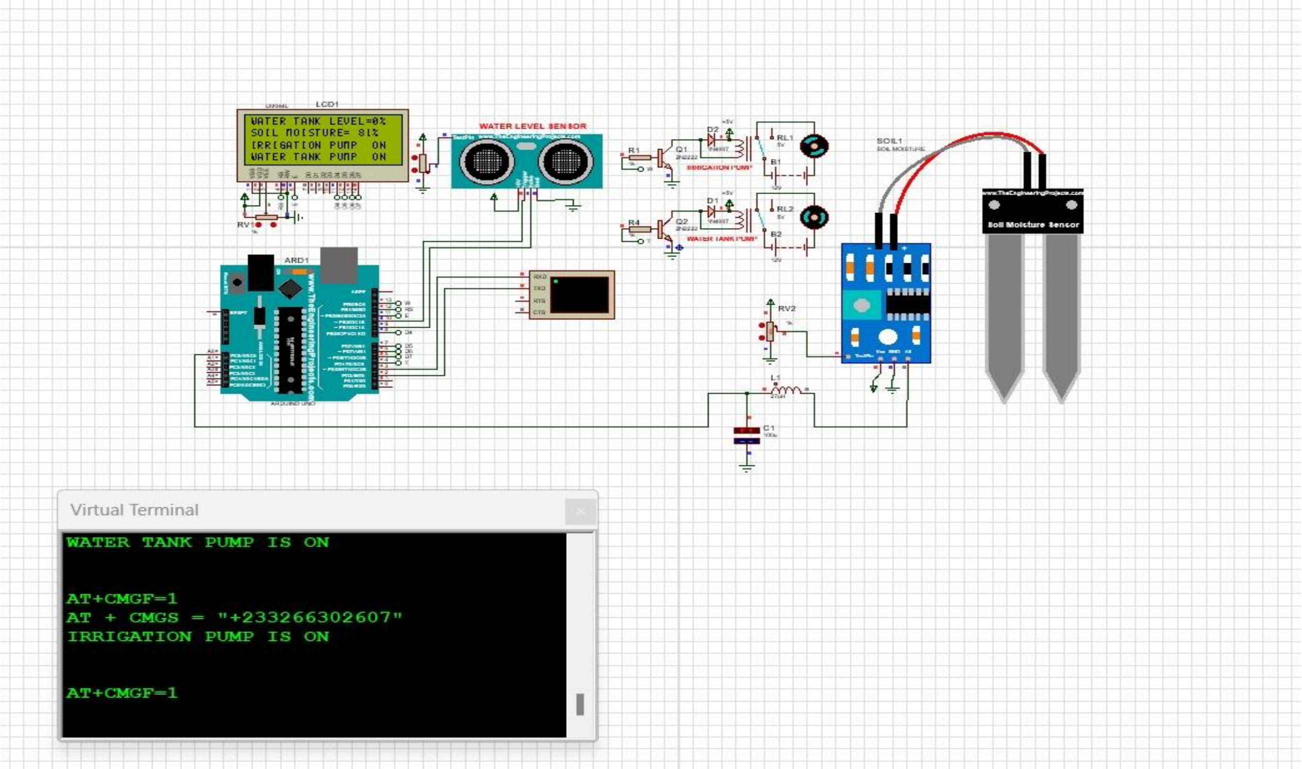


```
const int flowMeterPin = 2; // Connect the flow meter's output to digital pin 2
unsigned long pulseCount = 0;
float flowRate = 0.0;
float totalFlow = 0.0;
unsigned long previousMillis = 0;
const unsigned long interval = 1000; // Update interval in milliseconds

void setup() {
  pinMode(flowMeterPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  unsigned long currentMillis = millis();

  if (digitalRead(flowMeterPin) == LOW) {
    pulseCount++;
  }

  if (currentMillis - previousMillis >= interval) {
    // Calculate flow rate and total flow
    flowRate = pulseCount / (interval / 1000.0); // Flow rate in pulses per second
    totalFlow += (flowRate / 450); // Assuming 450 pulses per liter, adjust for your flow meter

    // Print flow rate and total flow
    Serial.print("Flow Rate: ");
    Serial.print(flowRate);
    Serial.print(" PPS\t"); // PPS: Pulses Per Second
    Serial.print("Total Flow: ");
    Serial.print(totalFlow);
    Serial.println(" Liters");

    pulseCount = 0;
    previousMillis = currentMillis;
  }
}
```

```
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Flow Rate: 0.00 PPS    Total Flow: 0.00 Liters
Flow Rate: 0.00 PPS    Total Flow: 0.00 Liters
Flow Rate: 0.00 PPS    Total Flow: 0.00 Liters
Flow Rate: 0.00 PPS    Total Flow: 0.00 Liters
```

Diagrams:





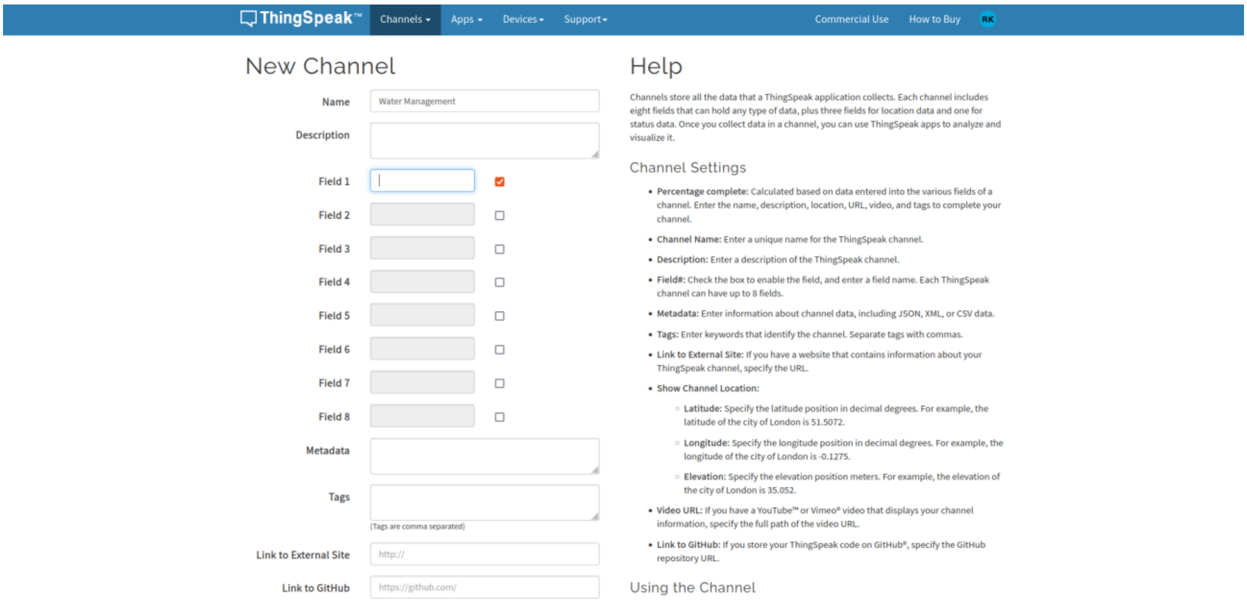
```
WATER TANK LEVEL=0%
SOIL MOISTURE= 81%
IRRIGATION PUMP  ON
WATER TANK PUMP  ON
```

```
WATER TANK PUMP IS ON


AT+CMGF=1
AT + CMGS = "+233266302607"
IRRIGATION PUMP IS ON


AT+CMGF=1
```

Data Sharing Platform:

We used ThingSpeak software.

1) Opening an account and creating a new channel.



New Channel

Name: Water Management

Description:

Field 1 ☑

Field 2 ☐

Field 3 ☐

Field 4 ☐

Field 5 ☐

Field 6 ☐

Field 7 ☐

Field 8 ☐

Metadata:

Tags:

(Tags are comma separated)

Link to External Site: http://

Link to GitHub: https://github.com/

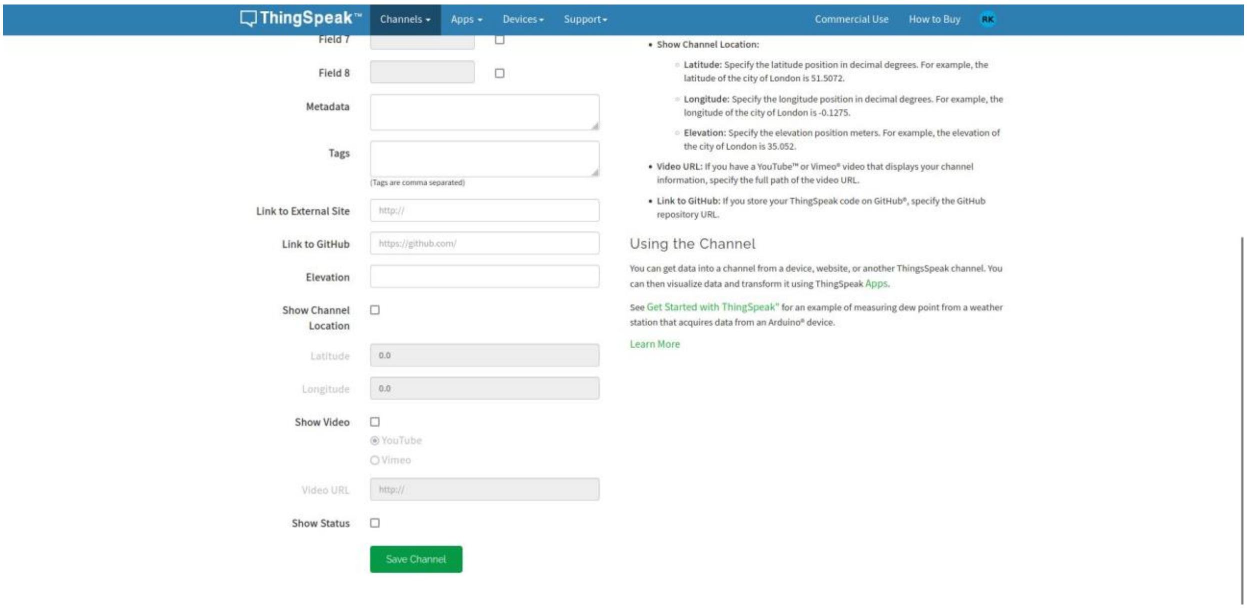
Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
    - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
    - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
    - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- **Video URL:** If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.
- **Link to GitHub:** If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

Using the Channel



Field 7 ☐

Field 8 ☐

Metadata:

Tags:

(Tags are comma separated)

Link to External Site: http://

Link to GitHub: https://github.com/

Elevation:

Show Channel Location ☐

Latitude: 0.0

Longitude: 0.0

Show Video ☐

● YouTube

○ Vimeo

Video URL: http://

Show Status ☐

- **Show Channel Location:**
    - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
    - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
    - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- **Video URL:** If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.
- **Link to GitHub:** If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

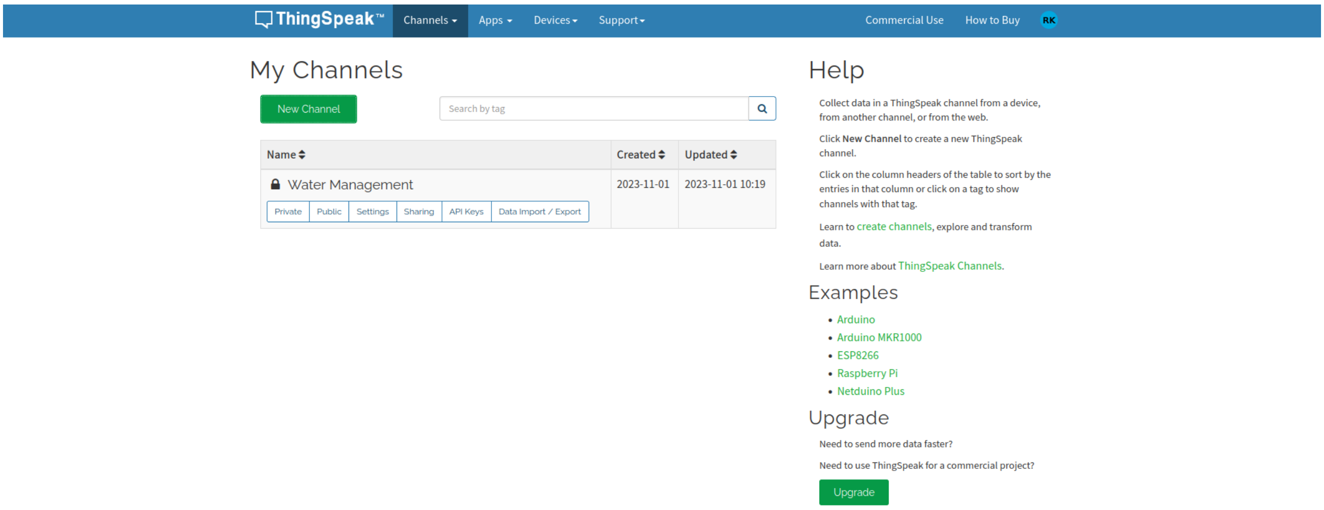
Using the Channel

You can get data into a channel from a device, website, or another ThingsSpeak channel. You can then visualize data and transform it using ThingSpeak Apps.

See Get Started with ThingSpeak" for an example of measuring dew point from a weather station that acquires data from an Arduino® device.

Learn More

2)Created new channel:



My Channels

New Channel

Search by tag

Name	Created	Updated
Water Management Private Public Settings Sharing API Keys Data Import / Export	2023-11-01	2023-11-01 10:19

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

Upgrade

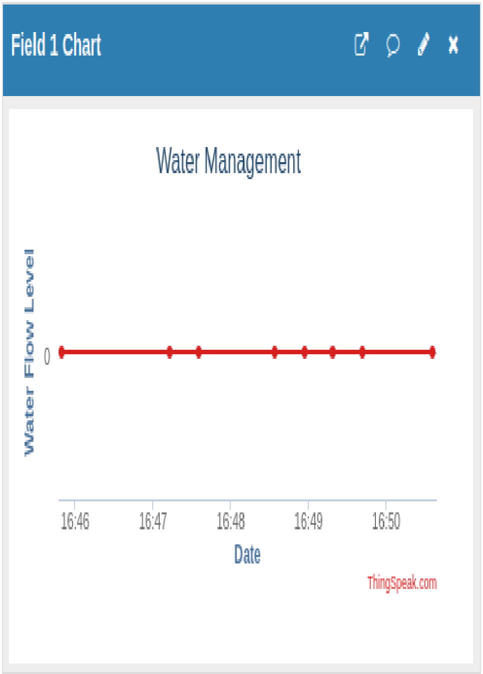
Need to send more data faster?

Need to use ThingSpeak for a commercial project?

Upgrade

3)Visualizing the data:

Entries: 8

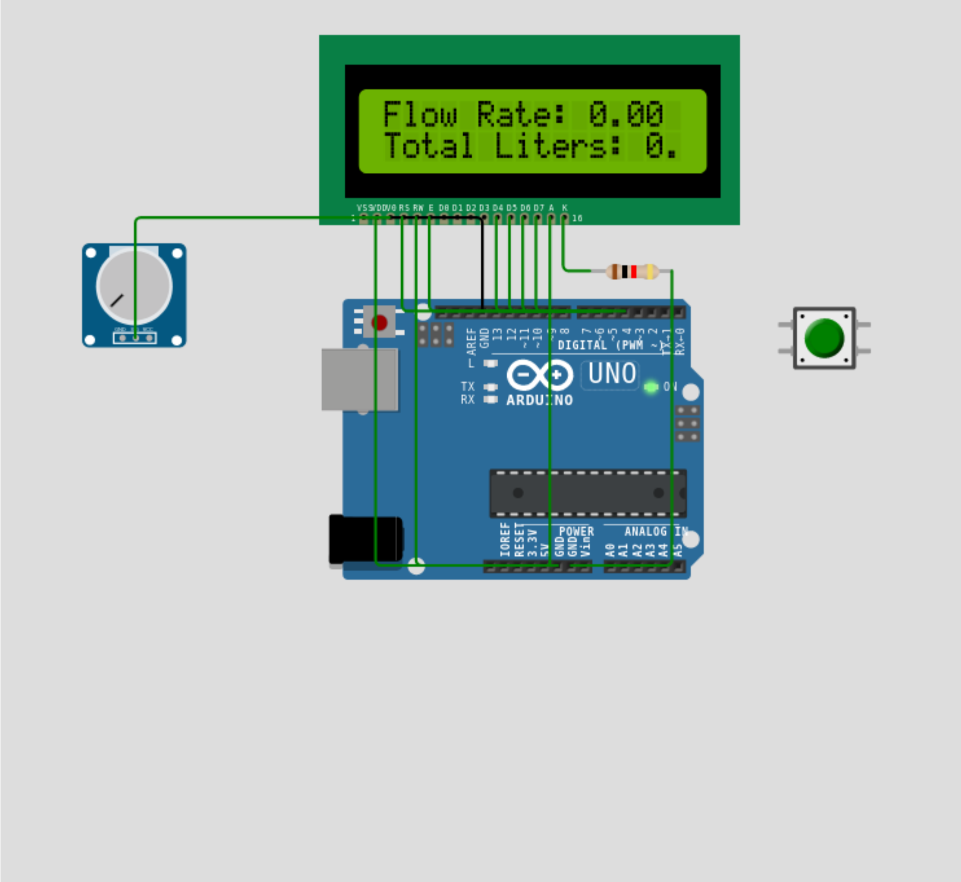


Code used for ThingSpeak in WOKWI:

Program:

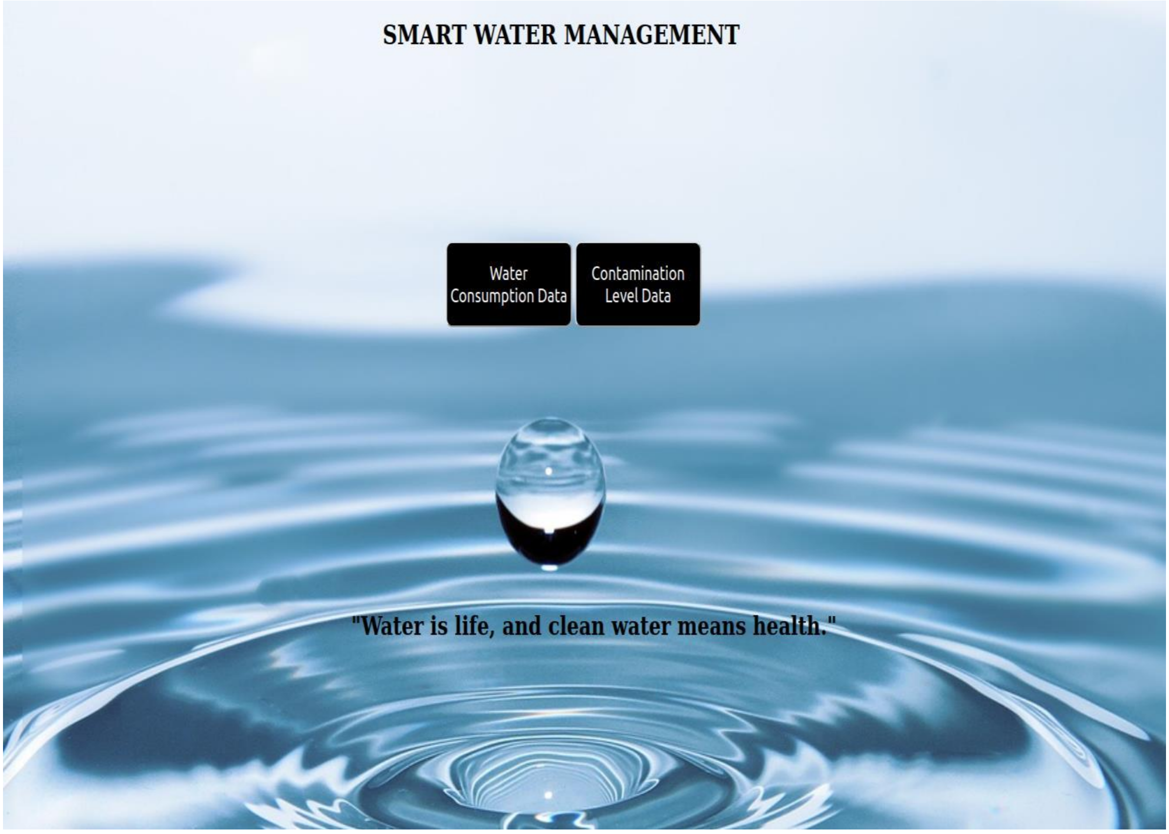
```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <WiFi.h>
#include <NewPing.h>
#include <ThingSpeak.h>
// ThingSpeak settings
const char *ssid = "Wokwi-GUEST";
const char *password = "";
const char *thingSpeakApiKey = "YourAPIkey";
const long channelId = Your Channel ID;
const int trigPin = 10;
const int echoPin = 30;
NewPing sonar(trigPin, echoPin);
WiFiClient client;
void setup() {
Serial.begin(115200);
// Connect to Wi-Fi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.println("Connecting to WiFi...");
}
Serial.println("Connected to WiFi");
ThingSpeak.begin(client);
}
void loop() {
unsigned int distance = sonar.ping_cm();
Serial.print("Distance: ");
Serial.print(distance);
Serial.println(" cm");
updateThingSpeak(distance);
delay(10000); }
void updateThingSpeak(unsigned int distance) {
ThingSpeak.setField(1, static_cast<float>(distance));
int status = ThingSpeak.writeFields(channelId, thingSpeakApiKey);
if (status == 200) {
Serial.println("ThingSpeak update successful!");
} else {
Serial.println("Error updating ThingSpeak. HTTP error code: " + String(status));
}
}
```

Smart Water Management:



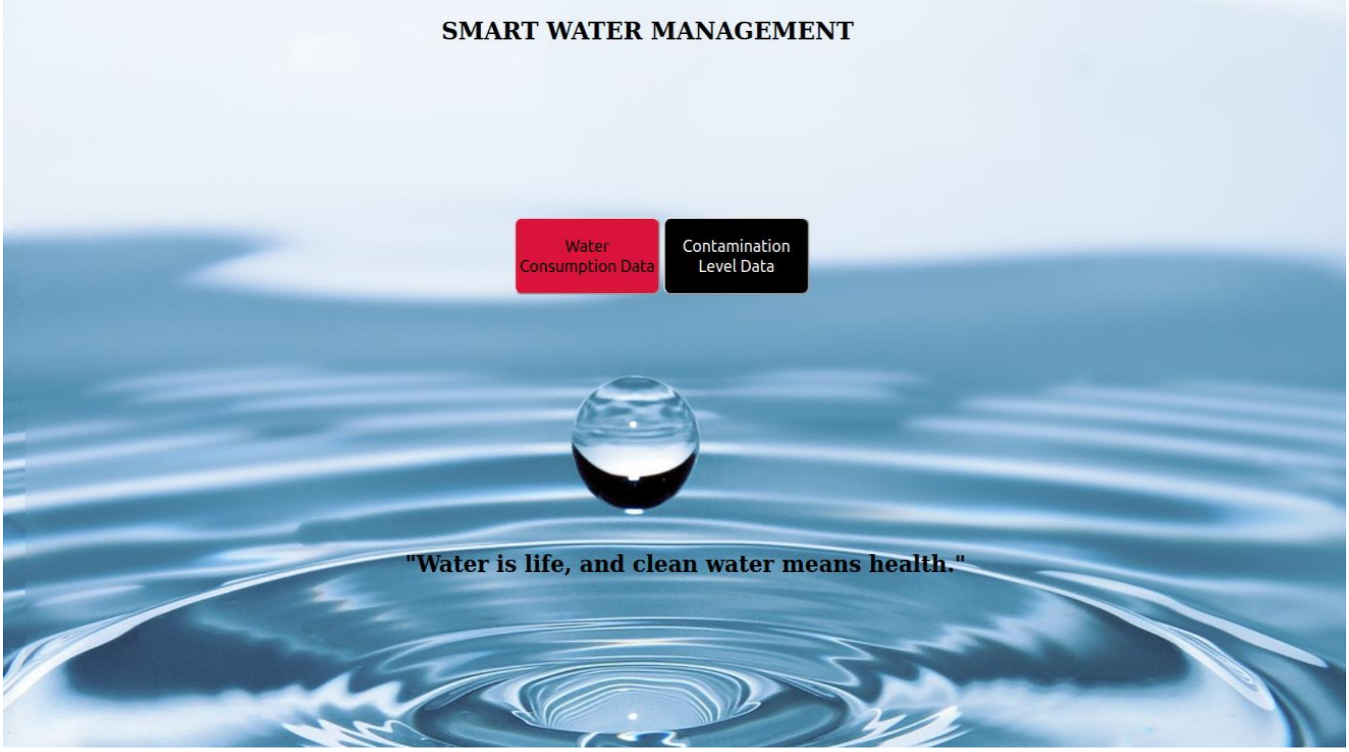
```
Flow Rate: 0.00
Total Liters: 0.
```

To see the real time data:



SMART WATER MANAGEMENT

"Water is life, and clean water means health."



SMART WATER MANAGEMENT

"Water is life, and clean water means health."

Uses of this real time water consumption monitoring system:

1. Awareness and Education:

- **Real-Time Feedback:** Users receive immediate feedback on their water consumption patterns, encouraging awareness of daily usage.
- **Educational Insights:** Visualization of consumption trends helps users understand their impact on water resources, fostering a sense of responsibility.

2. Identifying Anomalies and Leaks:

- **Leak Detection:** The system can quickly identify and alert users about abnormal water usage patterns, indicating potential leaks.
- **Timely Repairs:** Prompt detection allows users to address leaks promptly, preventing water wastage and property damage.

3. Behavioral Changes:

- **Informed Decision-Making:** Users can make informed decisions about water use based on real-time data, encouraging more conscious and responsible behavior.
- **Adjustment of Habits:** Access to consumption data promotes behavioral changes, such as reducing unnecessary water usage or adopting water-efficient practices.

4. Water-Efficient Appliances and Fixtures:

- **Appliance Monitoring:** Integration with smart appliances provides insights into their water usage, promoting the adoption of water-efficient devices.
- **Recommendations for Upgrades:** The system can suggest upgrades to water-efficient appliances and fixtures based on usage patterns.

5. Customized Water Conservation Plans:

- **Personalized Recommendations:** Analyzing individual consumption patterns allows the system to provide personalized water conservation recommendations.
- **Goal Setting:** Users can set water conservation goals and track their progress over time, creating a sense of accomplishment.

6. Community Engagement:

- **Comparative Analysis:** Users can compare their water consumption with neighbors or community averages, fostering healthy competition and community-wide conservation efforts.
- **Community Challenges:** Implementing challenges or initiatives within a community can further encourage water-saving practices.

7. Drought Management:

- **Early Warning System:** Real-time monitoring provides an early warning system for potential water shortages during droughts.
- **Resource Allocation:** Authorities can allocate water resources more efficiently based on real-time demand, minimizing the impact of drought conditions.

*These are all the uses and advantages when we monitor the real time data of water consumption system.

*Because water is the driving force.

*We can save water by seeing the data and it can be useful to the environment.

*By doing this we can save water for future generations.

*So we want to save water to save life.

"Water is life, and clean water means health."

- Audrey Hepburn