Smart Water Management using IoT

1.Objective:

The objective of this project is to create a Smart Water Management system that leverages Internet of Things (IoT) technology to monitor and manage water resources efficiently. The project aims to reduce water wastage, prevent leaks, and ensure the sustainable use of water in urban and rural environments.

IoT Device Setup:

The project involves the deployment of various IoT devices for data collection, control, and communication. Here are the key components of the IoT device setup:

Water Quality Sensors:

These sensors are installed in water sources (e.g., reservoirs, rivers) to monitor water quality parameters such as pH, turbidity, and contamination levels.

Flow Sensors:

Flow sensors are placed in water distribution pipelines to measure water flow rates and detect any irregularities or leaks.

Water Level Sensors:

These sensors are deployed in water storage tanks and reservoirs to monitor water levels and trigger refill requests when necessary.

IoT Gateway:

A central IoT gateway collects data from all the sensors, preprocesses the data, and sends it to a cloud-based platform for analysis and control.

2.Platform Development:

The project's platform development involves creating a cloud-based system for data analysis, visualization, and control. The platform consists of the following components:

Cloud Data Storage:

Data collected from the IoT devices is stored in a cloud database for real-time and historical analysis. Services like AWS, Azure, or Google Cloud can be used for this purpose.

Data Analysis and Prediction:

Machine learning models are developed to analyze the data and predict water quality, consumption trends, and leak detection. These models help in making informed decisions.

Control System:

The platform can send commands back to the IoT devices to control water flow, shut off supply in case of emergencies, or trigger maintenance alerts.

Dashboard:

User-friendly dashboards are created for water authorities, environmental agencies, and consumers to access real-time information about water quality, consumption, and alerts.

3.Code Implementation:

The code for this project will be written in various programming languages, depending on the component:

IoT Device Code:

Each IoT device has its own code to read sensor data and transmit it to the IoT gateway. This code may be written in languages like C, Python, or platforms like Arduino.

IoT Gateway Code:

The gateway has code to aggregate data, perform preprocessing, and securely transmit it to the cloud. Communication protocols like MQTT or HTTP may be used.

Cloud Data Storage Code:

Setting up and managing databases can be done using cloud providers' services and APIs.

Data Analysis and Prediction Code:

Machine learning models are implemented using Python, along with libraries like TensorFlow, scikit-learn, or specific water quality analysis tools.

Dashboard Code:

Web-based dashboards can be developed using HTML, CSS, and JavaScript, along with frameworks like React or Angular.

4.Explanation in Detail:

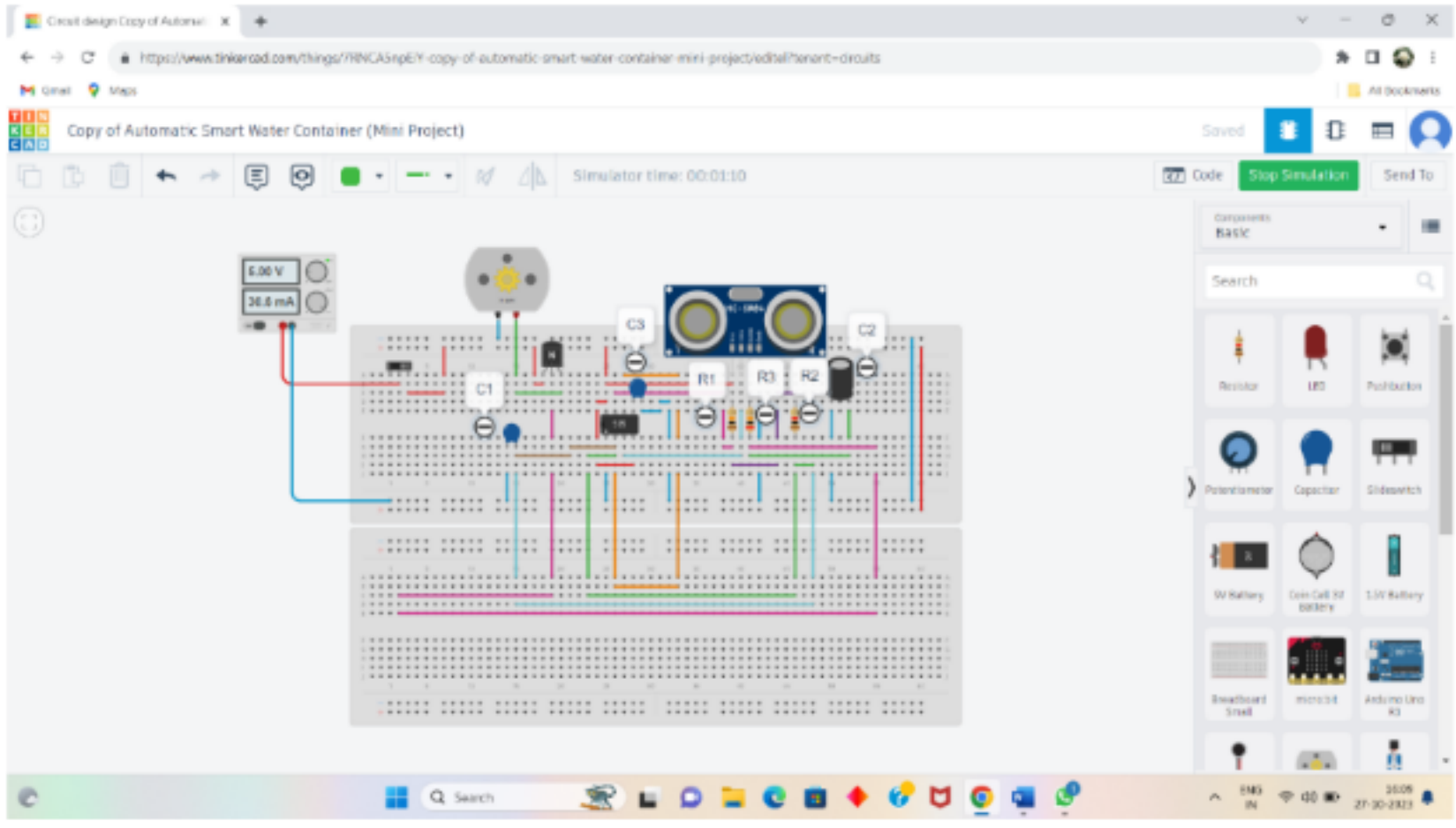
The project monitors water quality, flow rates, and levels to ensure efficient water management. Machine learning models help in early detection of water quality issues and leaks, allowing for preventive actions. Dashboards provide easy access to real-time data and alerts for decision-makers, enabling them to take timely actions to conserve water resources, prevent contamination, and ensure sustainable water usage.

This project's success contributes to better water quality, reduced water wastage, and more efficient water distribution, benefiting both the environment and the community.

URL for Tinkercad Project of Smart water Foundation:

https://www.tinkercad.com/things/7RNCA5npEiY-copy-of-automatic-smart-water-container-mini-project/editel?tenant=circuits

Screenshot of Smart water water Foundation Circuit(Tinkercad):



Code for Running above Circuit:

```
# Import the necessary library

import tinkercad

import requests  # Add the requests library for HTTP requests

# Initialize the Tinkercad API
```

```
tc = tinkercad.Tinkercad(username="Kanimozhi01", password="Kanimozhi")
# Find the simulation you want to run (use the correct ID)
simulation_id = "2303774"
# Get the simulation
simulation = tc.get_simulation(simulation_id)
# Define the ThingSpeak API parameters
thingspeak_api_key = "G5YN4PEKH3VEQ0JA"
thingspeak_url = f"https://api.thingspeak.com/update?api_key={thingspeak_api_key}"
# Define the code to run in the Arduino
arduino_code = """
#include <Ultrasonic.h>
Ultrasonic ultrasonic(2, 3);  // Trigger (pin 2), Echo (pin 3)
void setup() {
  Serial.begin(9600);
}
void loop() {
  float distance = ultrasonic.Ranging(CM);
  Serial.println(distance);
  // Send data to the computer (Python script)
  Serial.print("D:");
  Serial.println(distance);


  delay(1000);
}
"""
# Upload and run the code in the simulation
simulation.run_code(arduino_code)
# Monitor the water level and send data to ThingSpeak
while True:
    data = simulation.get_serial_data()
    if data and data.startswith("D:"):
```

```
distance = float(data[2:])
print(f"Water level: {distance} cm")
# Send data to ThingSpeak
try:
    response = requests.get(f"{thingspeak_url}&field1={distance}")
    if response.status_code == 200:
        print("Data sent to ThingSpeak successfully.")
    else:
        print("Failed to send data to ThingSpeak.")
except Exception as e:
    print("Error sending data to ThingSpeak:", str(e))
```

Thingspeak Channel stats of Smart Water foundation:

Platform UI code for Smart Water Foundation:

```
<!DOCTYPE html>
<html>
<head>
    <title>Smart Water Foundation</title>
    <style>
        /* Style for the water level display */
        #water-level {
            font-size: 24px;
```

```
        font-weight: bold;
      }
      /* Style for the submit button */
      #submit-button {
        padding: 10px 20px;
        font-size: 18px;
      }
    </style>
</head>
<body>
    <h1>Smart Water Foundation</h1>
    <p>Water Level: <span id="water-level">Loading...</span> cm</p>
    <button id="submit-button" onclick="sendDataToServer()">Submit Data</button>
    <script>
      // Function to update water level data from the server
      function updateWaterLevel() {
    // You can use AJAX or fetch to get data from your server
    // Replace the URL with the actual endpoint that provides water level data
    fetch('/getWaterLevelData')
      .then(response => response.json())
      .then(data => {
        document.getElementById('water-level').textContent = data.waterLevel + " cm";
      })
      .catch(error => {
        console.error('Error fetching water level data:', error);
      });
}
// Function to send data to the server (e.g., to trigger data collection)
function sendDataToServer() {
    // You can use AJAX or fetch to send data to your server
    // Replace the URL with the actual endpoint that handles data submission
```

```
        fetch('/submitData', { method: 'POST' })
            .then(response => {
                if (response.status === 200) {
                    console.log('Data submitted successfully');
                } else {
                    console.error('Data submission failed with status:', response.status);
                }
            })
            .catch(error => {
                console.error('Error submitting data:', error);
            });
    }
    // Update water level initially and then at regular intervals
    updateWaterLevel();
    setInterval(updateWaterLevel, 10000);  // Update every 10 seconds
</script>
</body>
</html>
```

Output for above Program: