Phase 3 : DEVELOPMENT PART 1

PROJECT TITLE	**IOT SMART WATER SYSTEM**
NAME	**V.KIRUBAGARAN**
TEAM ID	**5254**
TEAM NAME	**PROJ_204182_TEAM_2**
COLLEGE CODE - NAME	**9238-MANGAYARKARASI COLLEGE OF ENGINEERING PARAVAI, MADURAI**
GITHUB REPOSITORY LINK	**https://github.com/kirubagaran-v/IBM-NAANMUDHALVAN.git**

Abstract:

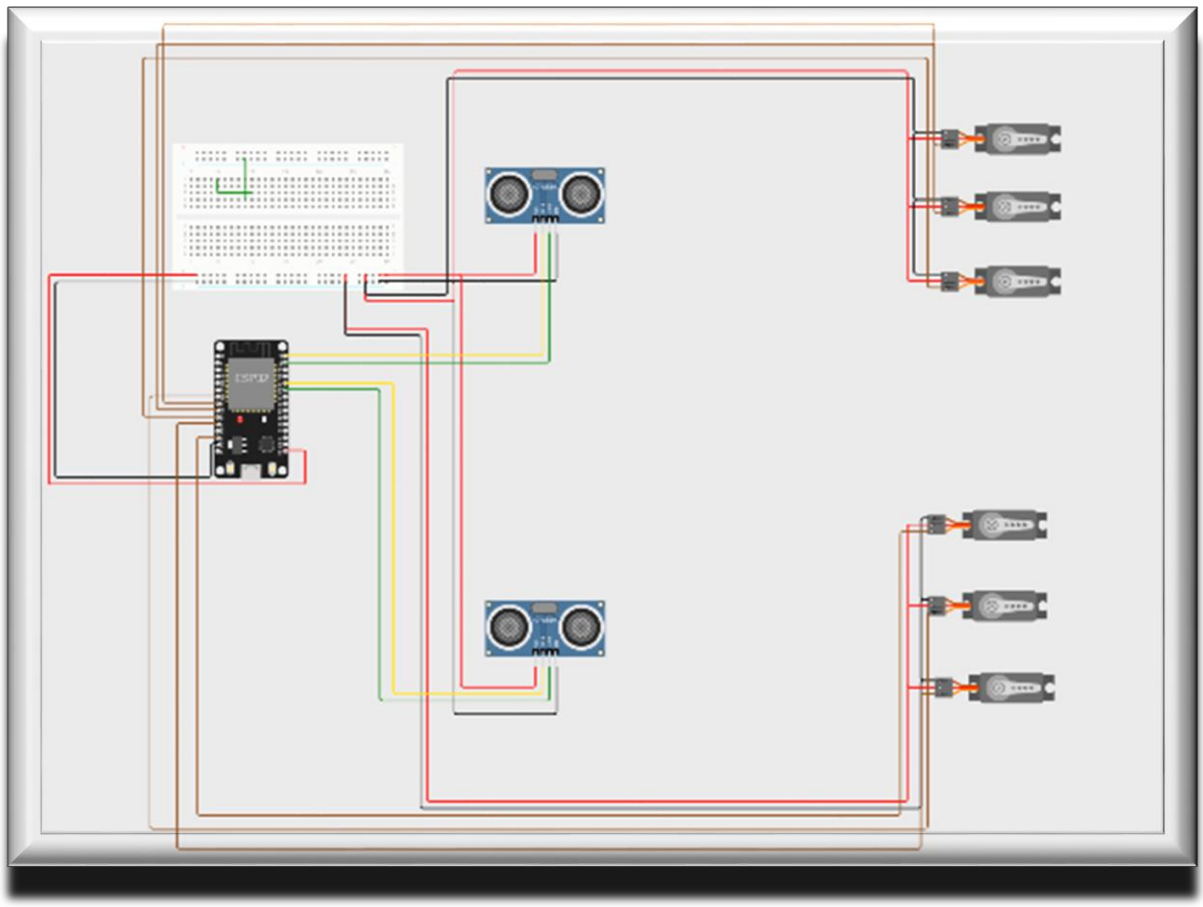
Water is essential for living things. So, we have to know about how to manage the water usage . where the water is being wasted ? find the places and we have to prevent this situation by **SMART WATER SYSTEM**. And we ensure the quality of water that is also important . so, monitor the pH value of the water by this plan .

1.Project Definition

1.1. Project Overview

Smart Water Management is the activity of planning, developing, distributing and managing the use of water resources using an array of IoT technologies which are designed to increase transparency, and make more reasonable and sustainable usage of these water resources.

Hardware setup :



Hardware Components：

1. ESP32 :

   The ESP32 serves the main control and runs the python program. It provide the necessary GPIO pins for connecting sensors

2. Ultrasonic sensor :

   Ultrasonic sensor (HC-SR04) are used to measure the water level in the tank and You have configured two ultrasonic sensor in the project.

3. Servo motors :

   Servo motors are act as valves in this project. Water can be distributed by this Servo motors. There are 6 servo motors used. Each ultrasonic sensor have three servo motors.

4. Bread board :

   A breadboard (sometimes called a plugblock) is used for building temporary circuits. It is useful to designers because it allows components to be removed and replaced easily. It is useful to the person who wants to build a circuit to

demonstrate its action, then to reuse the components in another circuit.

5.Jumper wires :

Jumper wires are used to establish connections between the ESP32's GPIO pins and the sensors.

Hardware Connections:

Here's how the hardware components should be connected based on your code:

Ultrasonic Sensors (HC-SR04):

Each ultrasonic sensor (e.g., HC-SR04) requires four connections:

- VCC (Voltage): Connect to a 5V pin on the ESP32 for power.
- GND (Ground): Connect to a ground (GND) pin on the ESP32.
- Trig (Trigger): Connect to the GPIO pins defined in trig_pins (pins 21 and 23 in your code).
- Echo: Connect to the GPIO pins defined in echo_pins (pins 19 and 22 in your code).

Servo motors :

There are 6 servo motors in this project. Three set of servo motors connected in series . so,There are two sets in this project.

- servo motor 1 : connect to the GPIO pins – 33
- servo motor 2 : connect to the GPIO pins - 25
- servo motor 3: connect to the GPIO pins - 26
- servo motor 4 : connect to the GPIO pins - 27
- servo motor 5 : connect to the GPIO pins - 14
- servo motor 6 : connect to the GPIO pins – 12

Wi-Fi Module:

You should ensure that your ESP32 is connected to a Wi-Fi network, either using• built-in Wi-Fi or an external Wi-Fi module. The program relies on this network connection to send data to Firebase.

Power Supply:

The ESP32 and sensors should be powered appropriately. The ESP32 can be powered• through a USB power supply, and sensors may need a separate 5V supply. Ensure that all components share a common ground .

Program:

Micro Python Coding:

```
import machine
import time
import urequests
import network
import ujson

# Define your Wi-Fi credentials
wifi_ssid = 'Wokwi-GUEST'
wifi_password = ''

# Connect to Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)

# Wait for Wi-Fi connection
while not wifi.isconnected():
    pass

# Define GPIO pins for ultrasonic sensors
trig_pins = [21, 23]  # Example pins, use the appropriate pins for your setup
echo_pins = [19, 22]  # Example pins, use the appropriate pins for your setup

# Define GPIO pins for servo motors (6 servos)
servo_pins = [13, 14, 25, 26, 27, 33]  # Example pins, use the appropriate pins for
your setup
servos = [machine.PWM(machine.Pin(pin), freq=50, duty=0) for pin in
servo_pins]

# Firebase Realtime Database URL and secret
firebase_url = 'https://iot-smart-water-manageme-6f2dc-default-rtdb.asia-
southeast1.firebasedatabase.app'
firebase_secret = 'PuZIQmVtENsSNLybJ4wDwEHzXUZiiKxsCgh7j6SS'

# Function to measure distance using ultrasonic sensor
def measure_distance(trig_pin, echo_pin):
    trig = machine.Pin(trig_pin, machine.Pin.OUT)
    echo = machine.Pin(echo_pin, machine.Pin.IN)

    # Ensure the trigger pin is low
```

```
    trig.off()
    time.sleep_us(2)

    # Generate a 10us pulse on the trigger pin
    trig.on()
    time.sleep_us(10)
    trig.off()

    # Measure the duration of the pulse on the echo pin
    while echo.value() == 0:
        pulse_start = time.ticks_us()
    while echo.value() == 1:
        pulse_end = time.ticks_us()

    # Calculate the duration of the pulse
    pulse_duration = time.ticks_diff(pulse_end, pulse_start)

    # Calculate the distance based on the speed of sound
    distance = (pulse_duration / 2) / 29.1  # Speed of sound in air is approximately
343 m/s

    return distance


# Function to calculate the water level percentage
def calculate_water_level_percentage(current_distance, min_distance,
max_distance):
    if current_distance < min_distance:
        return 0
    elif current_distance > max_distance:
        return 100
    else:
        return ((current_distance - min_distance) / (max_distance - min_distance)) *
100
# Define the minimum and maximum distances your sensor can detect
min_distance = 2  # Example minimum distance (in cm)
max_distance = 400  # Example maximum distance (in cm)

# Function to send water level to Firebase
def send_water_level_to_firebase(water_level_percentage, sensor_number):
```

```
    data = {'WaterLevel': water_level_percentage}
    url = f'{firebase_url}/sensor_{sensor_number}.json?auth={firebase_secret}'

    try:
        response = urequests.patch(url, json=data)
        if response.status_code == 200:
            print(f"Data for Sensor {sensor_number} sent to Firebase")
        else:
            print(f"Failed to send data to Firebase. Status code:
{response.status_code}")
    except Exception as e:
        print(f"Error sending data to Firebase: {str(e)}")

while True:
    # Measure water level using ultrasonic sensors
    for sensor_number in range(len(trig_pins)):
        ultrasonic_distance = measure_distance(trig_pins[sensor_number],
echo_pins[sensor_number])

        # Calculate and print water level percentage
        water_level_percentage =
calculate_water_level_percentage(ultrasonic_distance, min_distance,
max_distance)
        print(f"Water Level (Sensor {sensor_number + 1}):
{water_level_percentage:.2f}%")

        # Send data to Firebase
        send_water_level_to_firebase(water_level_percentage, sensor_number)

        # Control the servo valve based on the water level percentage
        if water_level_percentage < 50:
            # Adjust servo control logic based on the water level percentage
            servos[sensor_number].duty(512)  # Set servo duty cycle to a value to
control it
        else:
            servos[sensor_number].duty(0)  # Set servo duty cycle to 0 to stop it
    send_water_level_to_firebase(water_level_percentage, sensor_number)
    time.sleep(1)  # Adjust the sleep duration as needed
```

This is our Python program which is used to design for a smart water consumption monitoring system using ultrasonic and other sensors to monitor the water consumption and send this information to a Firebase Real-time Database .

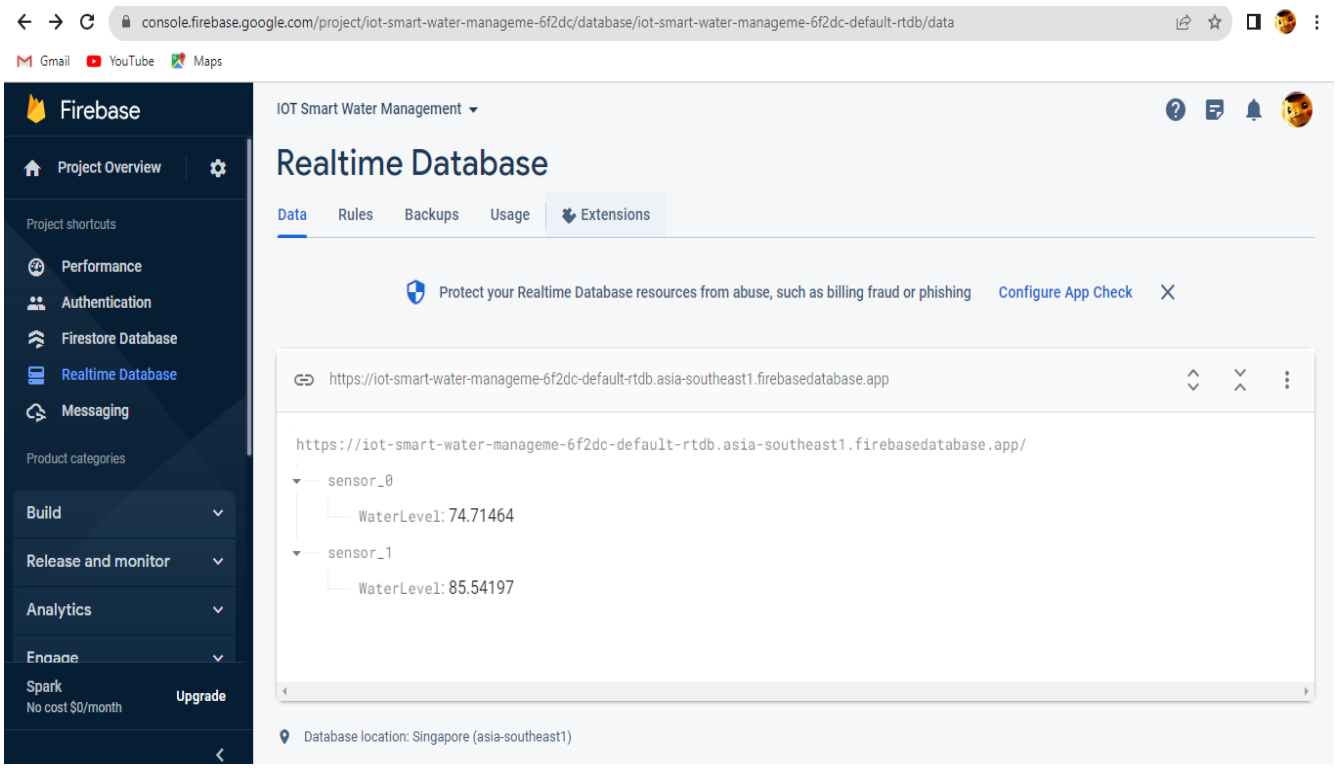
1. **Import Libraries**:
    - **machine**: Provides access to the hardware components of the microcontroller.
    - **time**: Allows you to work with time delays.
    - **urequests**: Enables HTTP requests to be made to send data to Firebase.
    - **network**: Used for connecting to Wi-Fi.
2. **Wi-Fi Setup**:
    - You define the Wi-Fi credentials (SSID and password) and connect to the network.
3. **Pin Definitions**:
    - You define GPIO pins for ultrasonic sensors (trig and echo), as well as for servo motors.
4. **Firebase Credentials**:
    - You provide the URL of your Firebase Realtime Database and the secret for authentication.
5. **Functions**:
    - **measure_distance(trig_pin, echo_pin)**: Measures the distance using an ultrasonic sensor. This function triggers the sensor and calculates the distance based on the time taken for the echo pulse to return.
    - **calculate_water_level_percentage(current_distance, min_distance, max_distance)**: Calculates the water level

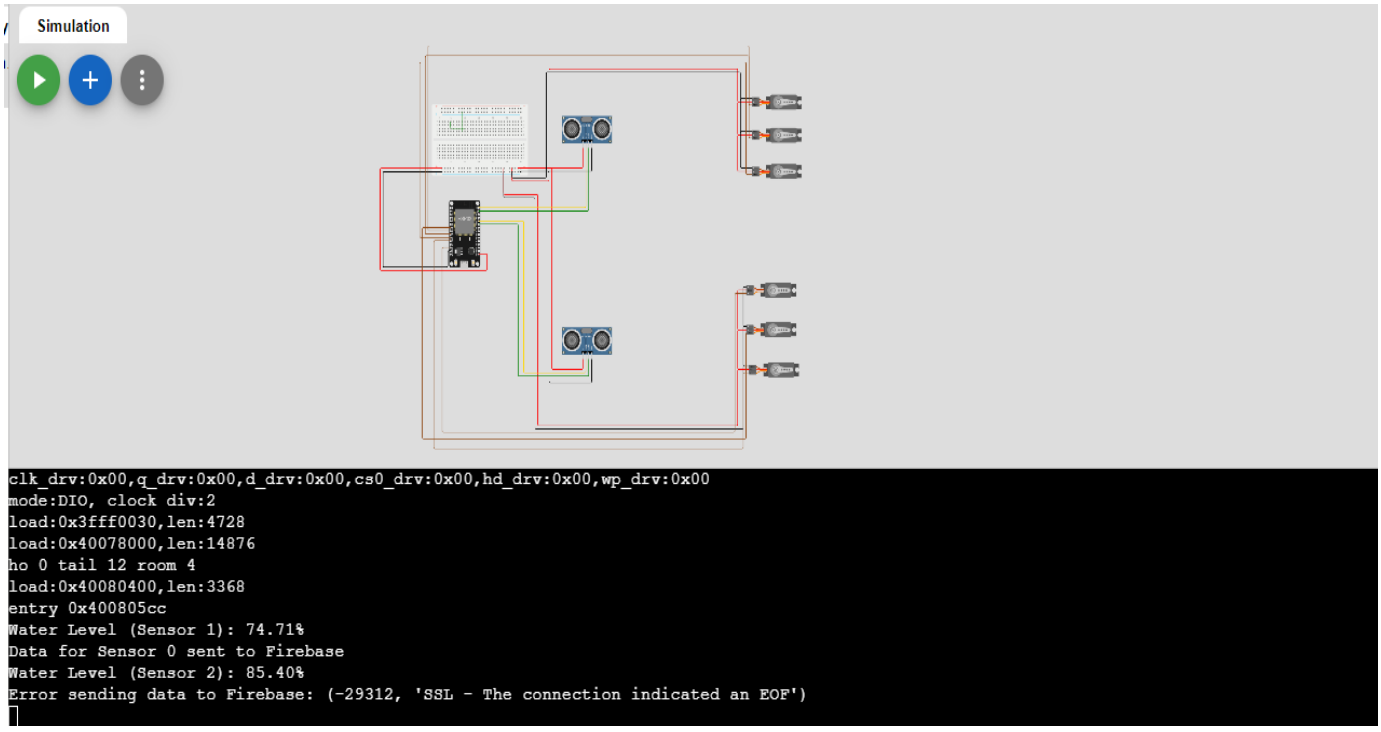
percentage based on the current distance, minimum distance, and maximum distance.
- **send_water_level_to_firebase(water_level_percentage, sensor_number)**: Sends the water level percentage to Firebase for a specific sensor. It uses a PATCH request to update the data in Firebase.

6. **Main Loop**:
   - It's an infinite loop (**while True**) where the following steps occur repeatedly:
     - For each sensor, it measures the water level, calculates the percentage, sends the data to Firebase, and controls the servo valve based on the water level.
     - The loop also includes a sleep period of 1 second to control the data transmission frequency.

Firebase Database:

Real time Database: Firebase Realtime Database is a cloud-hosted NoSQL database provided by Firebase,• a mobile and web application development platform that is now part of Google's cloud offerings. Firebase Realtime Database is designed for real-time data synchronization and is commonly used in applications where you need to store, retrieve, and synchronize data across various clients and platforms.


console.firebase.google.com/project/iot-smart-water-manageme-6f2dc/database/iot-smart-water-manageme-6f2dc-default-rtdb/data
Gmail
YouTube
Maps
Firebase
IOT Smart Water Management
Project Overview
Project shortcuts
Performance
Authentication
Firestore Database
Realtime Database
Messaging
Product categories
Build
Release and monitor
Analytics
Engage
Spark
No cost $0/month
Upgrade
Realtime Database
Data
Rules
Backups
Usage
Extensions
Protect your Realtime Database resources from abuse, such as billing fraud or phishing
Configure App Check
https://iot-smart-water-manageme-6f2dc-default-rtdb.asia-southeast1.firebasedatabase.app
https://iot-smart-water-manageme-6f2dc-default-rtdb.asia-southeast1.firebasedatabase.app/
sensor_0
WaterLevel: 74.71464
sensor_1
WaterLevel: 85.54197
Database location: Singapore (asia-southeast1)


Simulation
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14876
ho 0 tail 12 room 4
load:0x40080400,len:3368
entry 0x400805cc
Water Level (Sensor 1): 74.71%
Data for Sensor 0 sent to Firebase
Water Level (Sensor 2): 85.40%
Error sending data to Firebase: (-29312, 'SSL - The connection indicated an EOF')

CONCLUSION & FUTURE SCOPE

In this paper, a prototype water monitoring system using IOT is presented. For this some sensors are used. The collected data from the all the sensors are used for analysis purpose for better solution of water problems. The data is sends to the cloud server via Wi-Fi module ESP8266. So this application will be the best challenger in real time monitoring & control system and use to solve all the water related problems.