**IOT-enabled Environmental Monitoring in Parks system**

**PHASE-3**

INTRODUCTION:-

* Water fountains have been a major tourist attraction these days which freeze the attention of tourists with their variety of lights, designs, and heights. And as we all know music holds a major part in our day-to-day lives.
* And hence our idea is to combine the beautiful water fountain with music which makes an extraordinary tourist attraction when constructively set with a range of frequencies that enables us to operate through various electronic devices.
* The project aims to enhance public water fountains by implementing IoT sensors to control water flow and detect malfunctions.

THINGS USED IN THE PROJECT:-

COMPONENTS USED IN THE PROJECT:-

1. Microcontroller:

**ESP8266 or ESP32:** These are low-cost Wi-Fi modules with integrated microcontrollers. They are commonly used for IoT projects due to their Wi-Fi capabilities.

**2. Sensors:**

**Water Level Sensor:** To measure the water level in the fountain.

**Motion Sensor (PIR Sensor):** To detect the presence of people or animals near the fountain.

**Temperature and Humidity Sensor:** To monitor the environment around the fountain.

**3. Actuators:**

**Water Pump:** To control the flow of water in the fountain.

**LEDs:** For decorative lighting or indicating the fountain's status.

**4. Communication:**

**Wi-Fi Module:** Allows the fountain to connect to the internet.

**MQTT Protocol:** A lightweight messaging protocol for small sensors and mobile devices optimized for high-latency or unreliable networks.

**5. Power Supply:**

**Power Source:** Depending on the location of the fountain, you might use batteries, solar power, or a stable electrical source.

**6. IoT Platform:**

**Cloud Service (e.g., AWS IoT, Google Cloud IoT, Azure IoT):** A cloud platform to store data from the fountain and manage device communication.

**IoT Development Board:** Some development boards come with built-in support for IoT platforms, making it easier to connect your devices to the cloud.

**7. User Interface:**

**Mobile App/Web App:** Allows users to remotely control and monitor the fountain.

**Push Notifications:** Sends alerts or notifications to users based on fountain events (e.g., low water level).

**8. Security:**

**Encryption and Authentication:** Ensures secure communication between the fountain and the IoT platform.

**9. Data Storage and Analysis:**

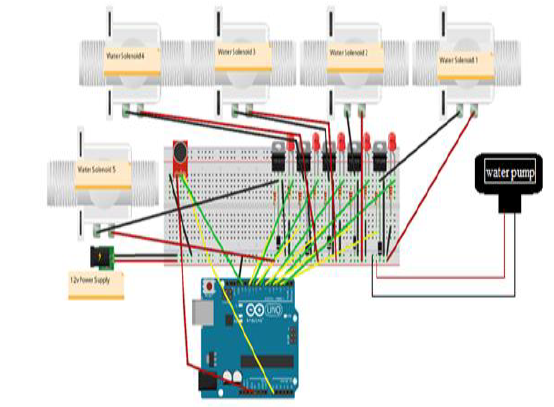
**Database:** For storing historical data from the fountain (e.g., water usage patterns, user interactions).

**Analytics Tools:** To gain insights from the collected data.

**10. Additional Components:**

**Real-Time Clock (RTC) Module:** Maintains accurate time for scheduling events even when the microcontroller is powered off.

**LCD Display:** Provides real-time feedback or information about the fountain's status.

**SCHEMATIC DIAGRAM:-**

**PYTHON CODE:-**

**import RPi.GPIO as GPIO  
import time  
  
# GPIO pins for water level sensor and pump  
WATER\_LEVEL\_PIN = 17  
PUMP\_PIN = 18  
  
# Setup GPIO  
GPIO.setmode(GPIO.BCM)  
GPIO.setup(WATER\_LEVEL\_PIN, GPIO.IN)  
GPIO.setup(PUMP\_PIN, GPIO.OUT)  
  
def is\_water\_available():  
 """Check if water is available based on the water level sensor."""  
 return GPIO.input(WATER\_LEVEL\_PIN) == GPIO.LOW  
  
def turn\_on\_pump():  
 """Turn on the water pump."""  
 GPIO.output(PUMP\_PIN, GPIO.HIGH)  
 print("Pump turned on")  
  
def turn\_off\_pump():  
 """Turn off the water pump."""  
 GPIO.output(PUMP\_PIN, GPIO.LOW)  
 print("Pump turned off")  
  
try:  
 while True:  
 if is\_water\_available():  
 turn\_on\_pump()  
 time.sleep(5) # Run the pump for 5 seconds (adjust as needed)  
 turn\_off\_pump()  
 else:  
 print("No water available. Waiting for refill...")  
 time.sleep(2) # Wait for 2 seconds before checking again  
  
except KeyboardInterrupt:  
 # Cleanup GPIO on keyboard interrupt  
 GPIO.cleanup()**

**OUTPUT:-**

