SMART WATER FOUNTAIN

PROBLEM STATEMENT

The project aims to enhance public water fountains by implementing IoT sensors to control water flow and detect malfunctions. The primary objective is to provide real-time information about water fountain status to residents through a public platform. This project includes defining objectives, designing the IoT sensor system, developing the water fountain status platform, and integrating them using IoT technology and Python. This smart water fountain should address the following challenges:

DESIGN THINKING:

Project Objectives:

Design and build a functional and aesthetically pleasing smart water fountain capable of delivering water in controlled patterns and integrate IoT technology into the water fountain system to enable remote monitoring, control, and data collection.

IoT Sensor Design:

Hardware components:

- Water pump and valves
- Sensors (e.g., flow sensors, water level sensors, temperature sensors)
- Microcontroller (e.g., Raspberry Pi, Arduino)
- Power supply

Design the Water Fountain:

- Create a physical design for the water fountain, considering aesthetics and functionality.Determine the layout of components, such as pumps, valves, and sensors.Ensure that the water flow system is safe and leak-proof.
- Connect the water pump, valves, and sensors to the microcontroller.
- Set up the microcontroller to connect to the internet via Wi-Fi or Ethernet.
- Install the necessary libraries or packages for IoT communication on your microcontroller (e.g., MQTT, HTTP).

Real-Time Transit Information Platform:

Design a mobile application Visualize data from the water fountain (e.g., flow rates, water levels).Allow remote control of the fountain's features.Use web frameworks like Flask, Django, or mobile app development tools.

Integration Approach:

python code for monitoring water flow and malfunction:

import RPi.GPIO as GPIO

```
import time
import Adafruit_DHT  # For DHT temperature and humidity sensor
import paho.mqtt.client as mqtt  # MQTT library

# Sensor pins and parameters
FLOW_SENSOR_PIN = 14  # GPIO pin for flow sensor
DHT_SENSOR_PIN = 4    # GPIO pin for DHT sensor
DHT_SENSOR_TYPE = Adafruit_DHT.DHT22  # Type of DHT sensor (DHT11, DHT22, or DHT2302)

# MQTT parameters
MQTT_BROKER_HOST = "your_mqtt_broker_host"
MQTT_BROKER_PORT = 1883
MQTT_TOPIC = "your_mqtt_topic"

# Initialize GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(FLOW_SENSOR_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Initialize MQTT client
client = mqtt.Client()

def read_flow_sensor(channel):
    # Function to read flow sensor data (replace with your sensor reading logic)
      # You may need to install RPi.GPIO and set up interrupt-based reading for precise flow measurement
    flow_rate = 0  # Replace with actual flow rate reading logic
    return flow_rate

def read_dht_sensor():
    # Function to read DHT temperature and humidity sensor
    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR_TYPE, DHT_SENSOR_PIN)
    return humidity, temperature

def control_water_flow(flow_rate):
    # Function to control water flow based on sensor readings
    # Implement your control logic here
    pass

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.subscribe(MQTT_TOPIC)

def on_message(client, userdata, msg):
    print(f"Received message: {msg.payload.decode()}")

# Set MQTT callbacks
client.on_connect = on_connect
client.on_message = on_message

# Connect to MQTT broker
client.connect(MQTT_BROKER_HOST, MQTT_BROKER_PORT, 60)

try:
    while True:
        # Read sensor data
        flow_rate = read_flow_sensor(FLOW_SENSOR_PIN)
        humidity, temperature = read_dht_sensor()

        # Control water flow based on sensor readings
        control_water_flow(flow_rate)
        # Send sensor data to MQTT broker
            client.publish(MQTT_TOPIC, f"Flow Rate: {flow_rate}, Humidity: {humidity}, Temperature: {temperature}")

        time.sleep(10)  # Adjust the interval as needed

except KeyboardInterrupt:
    GPIO.cleanup()
```