Project Title: IoT Smart Water Fountains

Project Title : Smart Water Fountains

Name	: J.Jasmine Nesa Thirsha
Reg No.	: 950421104021
Department	: Computer Science And Engineering
College Name	: Dr. G.U. Pope College of Engineering
College Code	: 9504
Course Name	: IoT (Internet of Things)

Project Overview:

The IoT Water Fountain project is a modern and innovative approach to water fountain management and control. It leverages Internet of Things (IoT) technology to create an automated and smart water fountain system. This system not only offers aesthetic appeal but also provides real-time monitoring, control, and conservation of water resources.

Circuit Diagram for IoT Smart Water Fountain:

```
  |     Water Level       |
  |      Sensor           |

+-----------+ |           | +-----------+
| Micro   | |           | | Water   |
|Controller | |            | |   Pump    |
| (e.g.,    | |            | |           |
| Raspberry | |            | |           |
| Pi, Arduino| |           | |           |
+-----------+ |            | +-----------+
```

Components and Connections:

Water Level Sensors:

Connect the water level sensor to one of the microcontroller’s GPIO pins to read water level data.

Microcontroller (e.g., Raspberry Pi or Arduino):

- Connect the microcontroller to the water level sensor.
- Connect the microcontroller to the water pump to control its operation.
- Ensure proper power connections for the microcontroller.

Water Pump:

- Connect the water pump to the microcontroller to control its operation.
- Ensure that the power supply for the water pump is adequate.

Coding and programming:

```
import RPi.GPIO as GPIO
Import time
Import paho.mqtt.client as mqtt

# GPIO setup
GPIO.setmode(GPIO.BCM)
Water_pump_pin = 17
GPIO.setup(water_pump_pin, GPIO.OUT)
```

```
# MQTT setup
Mqtt_broker = “your_broker_address”
Mqtt_topic = “water_fountain/control”
Client = mqtt.Client()

# Callback for MQTT message received
Def on_message(client, userdata, message):
    If message.payload.decode() == “on”:
GPIO.output(water_pump_pin, GPIO.HIGH)
    Else:
        GPIO.output(water_pump_pin, GPIO.LOW)

# Connect to MQTT broker
Client.on_message = on_message
Client.connect(mqtt_broker)
Client.subscribe(mqtt_topic)
Client.loop_start()

Try:
    While True:
        # Read water level and temperature sensors
        Water_level = …  # Read water level from sensor
        Temperature = …  # Read temperature from sensor

        # Your logic for controlling the water pump based on sensor readings
        If water_level < threshold:
        # Water level is too low, turn off the pump
        GPIO.output(water_pump_pin, GPIO.LOW)


        # Publish sensor data to the MQTT broker
        Client.publish("water_fountain/data", f"Water Level: {water_level}, Temperature:
{temperature}")


        # Add any other functionality or logic here
       Time.sleep(5)  # Adjust the interval as needed
       Except KeyboardInterrupt:
       GPIO.cleanup()
```

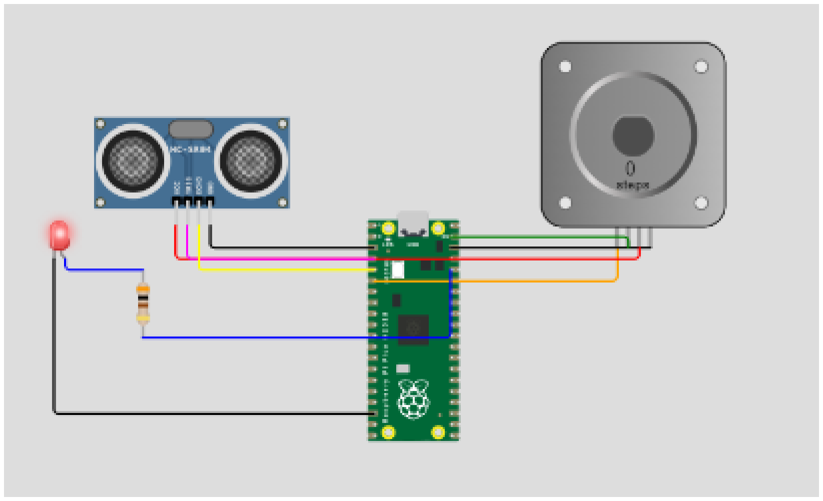
Data transmission:

Data transmission in an IoT smart water fountain project involves sending data from the fountain's sensors to a central server or cloud platform and receiving control commands or updates from a remote user interface (e.g., a mobile app or web interface).

Coding:

```
Import paho.mqtt.client as mqtt
# MQTT setup
Mqtt_broker = "your_broker_address"
Mqtt_topic_sensor = "water_fountain/data"
Mqtt_topic_control = "water_fountain/control"


Def on_connect(client, userdata, flags, rc):
    Print("Connected with result code " + str(rc))
    Client.subscribe(mqtt_topic_control)


Def on_message(client, userdata, message):
    Payload = message.payload.decode()
    # Handle control commands (e.g., pump control) based on payload


Client = mqtt.Client()
Client.on_connect = on_connect
Client.on_message = on_message


Client.connect(mqtt_broker, 1883, 60)
Try:
    While True:
        # Read sensor data and publish it
        Sensor_data = “Water level: 50%, Temperature: 25°C”  # Replace with actual data
        Client.publish(mqtt_topic_sensor, sensor_data)
Except KeyboardInterrupt:
    Client.disconnect()
```

Diagram:



Appendix:

IoT Smart Water Fountain

In this appendix, we provide additional information, technical specifications, and references related to the IoT smart water fountain.

Coding:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

Const char *ssid = “YourWiFiSSID”;
Const char *password = “YourWiFiPassword”;

Int pumpPin = D1; // Pin connected to the water pump relay
Bool pumpState = false;

ESP8266WebServer server(80);

Void setup()
{   pinMode(pumpPin,
OUTPUT);
digitalWrite(pumpPin,
LOW);   // 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”);

 // Define server routes
 Server.on(“/”, HTTP_GET, handleRoot);
```

```
  Server.on("/control", HTTP_POST, handleControl);

  Server.begin();
}

Void loop() {
  Server.handleClient();
}

Void handleRoot() {
  String html = "<html><body>";
  Html += "<h1>IoT Smart Water Fountain</h1>";
  Html += "<p>Pump is " + String(pumpState ? "ON" : "OFF") + "</p>";
  Html += "<form method='post' action='/control'>";
  Html += "<input type='submit' value='Toggle Pump'>";
  Html += "</form></body></html>";

  Server.send(200, "text/html", html);
}

Void handleControl() {   pumpState =
 !pumpState;   digitalWrite(pumpPin,
pumpState ? HIGH : LOW);
server.send(200, "text/plain", "Pump state
toggled");
}
```

Conclusion:

The IoT smart water fountain is a practical, efficient, and environmentallyfriendly solution for managing water features. It offers convenience, remote control, and real-time monitoring while promoting water conservation and aesthetic appeal.