**SMART WATER FOUNTAIN**

The specific objectives of the project may vary depending on the goals of the municipality, organization, or individuals responsible for implementing the smart water fountain. Ultimately, the aim is to create a technologically advanced and environmentally responsible water feature that enhances public spaces, conserves resources, and provides an enjoyable experience for all.

**HARDWARE COMPONENTS:**

1.Arduino or compatible microcontroller.

2.Water pump and tubing.

3.Water level sensor (to monitor water levels).

4.Relay module (to control the water pump).

5.Power source.

6.Enclosure to house the components.

7.Optional: a camera for monitoring.

**SOFTWARE COMPONENTS:**

1.Arduino IDE for programming the microcontroller.

2.HTML, CSS, and JavaScript for the web interface.

3.Node.js and Express.js for the web server.

4.WebSocket or AJAX for real-time communication.

5.Database (e.g., MySQL or MongoDB) to store data.

6.Optional: mobile app development using frameworks like React Native.

**TO CREATE THE SMART WATER FOUNTAIN STEPS:**

1. **HARDWARE SETUP:**

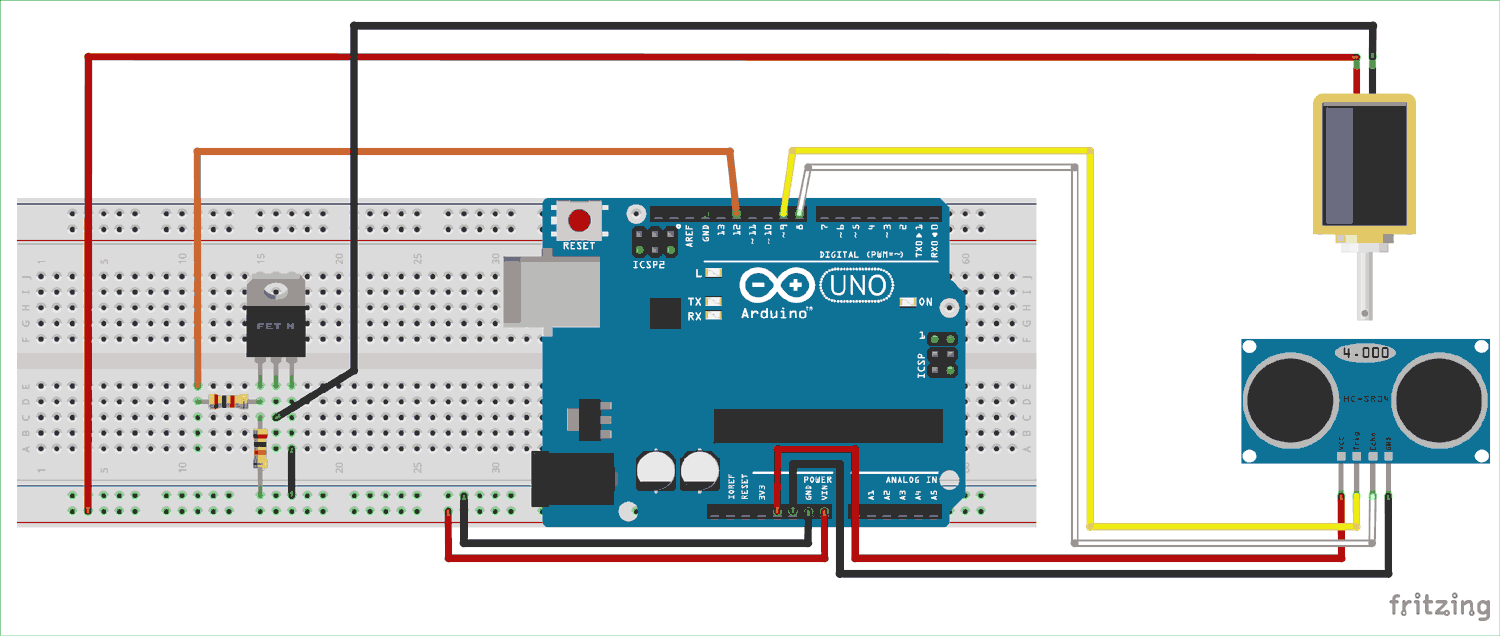
I)Connect the relay module to the Arduino. water pump, water level sensor, and

: II)Power the components appropriately.

1. **ARDUINO PROGRAMMING**

I)Write Arduino code to control the water pump and read the water level sensor.

II)Implement a simple communication protocol for the Arduino to receive commands from the web server.



**3.WEB DEVELOPMENT:**

I)Create a web interface using HTML, CSS, and JavaScript to control the water fountain.

II)Use AJAX or WebSockets for real-time interaction with the Arduino.

**4.WEB SERVER:**

I)Set up a Node.js server using Express.js.

II)Create routes for the web interface and communication with the Arduino.

**5.DATABASE INTEGRATION:**

I)Store data such as water usage and user preferences in a database.

**6.REAL-TIME CONTROL:**

I)Implement real-time updates on the web interface to display water levels and status.

**7.MOBILE APP(OPTIONAL):**

I)Develop a mobile app to control the water fountain remotely.

**8.SECURITY**:

I)Implement security measures to prevent unauthorized access to the system.

**9.TESTING AND DEBUGGING:**

I)Thoroughly test the system to ensure it works as expected.

**10.DEPLOYMENT:**

I)Deploy the web server and database on a hosting platform.

II)Ensure the system is accessible remotely.

**11.MONITORING AND MAINTENANCE:**

I)Set up monitoring for the system's health.

II)Regularly maintain and update the software and hardware.

**ARDUINO FIRMWARE(in C/C++ using Arduino IDE):**

**Using this code, we can control the water pump**

#include <ESP8266WiFi.h>

#include <ESP8266mDNS.h>

#include <ArduinoOTA.h>

const char\* ssid = "your\_SSID";

const char\* password = "your\_PASSWORD";

void setup() {

Serial.begin(115200);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

while (WiFi.waitForConnectResult() != WL\_CONNECTED) {

Serial.println("Connection Failed! Rebooting...");

delay(5000);

ESP.restart();

}

// Initialize your pins and sensors here

}

void loop() {

// Read sensors and control the water pump

// Implement communication with the web server here

}

**PYTHON CODE(WEB SERVER AND WEB INTERFACE):**

**Using this code,we can create the web server and web interface**

python

Copy code

from flask import Flask, render\_template, request, jsonify

import serial # for Arduino communication

app = Flask(\_\_name)

# Set up serial communication with Arduino

ser = serial.Serial('/dev/ttyUSB0', 9600) # Replace with the correct COM port

@app.route("/")

def index():

return render\_template("index.html")

@app.route("/control", methods=["POST"])

def control():

action = request.form["action"]

if action == "turn\_on":

ser.write(b'1') # Send command to Arduino to turn on the pump

elif action == "turn\_off":

ser.write(b'0') # Send command to Arduino to turn off the pump

return jsonify(status="success")

if \_name\_ == "\_main\_":

app.run(debug=True)

**CODE FOR WEB INTERFACE(HTML-index.html):**

**Using this code, we can control the water fountain**

<!DOCTYPE html>

<html>

<head>

<title>Smart Water Fountain</title>

</head>

<body>

<h1>Smart Water Fountain</h1>

<button id="turn-on">Turn On</button>

<button id="turn-off">Turn Off</button>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script>

$(document).ready(function() {

$("#turn-on").click(function() {

$.post("/control", { action: "turn\_on" }, function(data) { console.log("Turned on the fountain");

});

});

$("#turn-off").click(function() {

$.post("/control", { action: "turn\_off" }, function(data) { console.log("Turned off the fountain");

});

});

});

</script>

</body>

</html>

This code sets up a basic web server and web interface to control the water fountain. Please note that this is a complex project that combines both hardware and software development. Depending on your requirements, it may take some time to complete. Always prioritize safety when working with water and electricity.

**CONCLUSION:**

This simple smart fountain project will give you a hands-on experience with IoT and basic hardware integration. As you become more comfortable with these concepts, you can consider adding more features, such as scheduling, remote monitoring, or integration with other IoT devices.