**Smart water fountains**

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.

**Components You’ll Need :**

**Water Pump:** To create the fountain effect.

**Microcontroller:** A popular choice is the Arduino, but you can also use Raspberry Pi, ESP8266, or ESP32**.**

**Relay or Motor Driver:** To control the water pump.

**Wi-Fi Module:** Such as an ESP8266 or ESP32 for IoT connectivity.

**Power Supply:** To provide power to the components

**Mobile App or Web Interface:** To control the fountain remotely.

**Step-by-Step Guide:**

**Set Up the Hardware:**

* Connect the water pump to the relay or motor driver. The microcontroller should control this component.
* Connect the water level sensor to the microcontroller. Ensure it can accurately detect water levels.

**Program the Microcontroller:**

* Write the code for your microcontroller to control the water pump and read data from the water level sensor. You’ll need a suitable development environment for your chosen microcontroller.
* Create a program that allows you to turn the fountain on and off. You can also program it to turn on or off based on the water level (e.g., turn off when the water level is too low to prevent the pump from running dry).

**Add IoT Connectivity:**

* Integrate the Wi-Fi module with your microcontroller. Many libraries are available to help you connect to the internet.
* Implement code to connect to your Wi-Fi network and send/receive data over the internet.

**Create a Control Interface:**

* Develop a simple mobile app or web interface that allows you to control the fountain remotely. You can use platforms like Blynk, Thinger.io, or even build a custom app or web page.
* Ensure this interface can send commands to your fountain (e.g., turn it on, turn it off).

**Test and Debug:**

* Test the system to make sure it works as expected. Ensure the water pump turns on and off correctly based on your control interface.
* Verify that the water level sensor is providing accurate readings.

**Enclosure and Safety:**

* House the components in a suitable enclosure to protect them from water damage.
* Ensure all electrical connections are safe and sealed.

**Finalize and Deploy:**

* Make any necessary adjustments based on testing and user feedback.
* Deploy your smart fountain in a suitable location.

**Python Code:**

import RPi.GPIO as GPIO

import time

# Set the GPIO mode to BCM

GPIO.setmode(GPIO.BCM)

# Define GPIO pins for the water pump and water level sensor

water\_pump\_pin = 17

water\_level\_sensor\_pin = 18

# Set up the GPIO pins

GPIO.setup(water\_pump\_pin, GPIO.OUT)

GPIO.setup(water\_level\_sensor\_pin, GPIO.IN)

# Function to turn on the water pump

def turn\_on\_pump():

GPIO.output(water\_pump\_pin, GPIO.HIGH)

# Function to turn off the water pump

def turn\_off\_pump():

GPIO.output(water\_pump\_pin, GPIO.LOW)

try:

while True:

# Check the water level

if GPIO.input(water\_level\_sensor\_pin) == GPIO.LOW:

print("Water level is low. Turning on the pump.")

turn\_on\_pump()

else:

print("Water level is sufficient. Turning off the pump.")

turn\_off\_pump()

time.sleep(1) # Check water level every second

except KeyboardInterrupt:

# Clean up and exit on Ctrl+C

GPIO.cleanup()

**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.