SMART WATER FOUNTAIN

IOT_PHASE3

Processing and Data

1. **Hardware Setup:**

   - Acquire IoT devices like ultrasonic sensors or cameras for detecting parking spaces.

   - Set up a Raspberry Pi or Arduino board to act as the IoT device to collect data.

   - Establish a network connection for the IoT device to send data.

2. **Data Collection:**

   - Use the IoT devices to collect data on parking space occupancy.

   - Ultrasonic sensors can detect the presence of a vehicle in a parking space, and cameras can captureimages for further analysis.

3. **Data Processing:**

   - Develop a Python script to process the data from the IoT devices.

   - Analyze the sensor data or images to determine parking space availability.

4. **Data Communication:**

   - Set up a communication protocol to send the parking space data to a central server or a cloud platform.

   - Use technologies like MQTT or HTTP for data transmission.

5. **User Interface:**

   - Create a user interface, possibly a web or mobile app, to display real-time parking space availability.

   - Users can check the app to find vacant parking spots.

6. **Document and Assessment:**

   - Document the project comprehensively, including hardware setup, data processing, and user interface.

   - Explain the Python script and the technologies used in detail.

   - Share the documentation with your instructor or assessors for evaluation.

SENSORS

1. Water Level Sensor: A water level sensor is essential to monitor the water level in the fountain. It can help prevent the water pump from running dry, which can damage the pump. These sensors can be float switches, capacitive sensors, or ultrasonic sensors.

2. Water Quality Sensor: Monitoring the water quality is important for maintaining a healthy and clean water source for the fountain. You can use sensors such as pH sensors, turbidity sensors, or conductivity sensors to check water quality.

3. Temperature Sensor: To monitor the water temperature in your fountain, you can use temperature sensors like DS18B20, DHT22, or DS3231. Maintaining the right water temperature can be crucial for certain fountain setups.

4. Motion Sensor: A motion sensor (PIR sensor) can be used to detect when someone is near the fountain. This can trigger features such as turning on decorative lighting or activating the fountain when someone approaches.

5. Light Sensor: Light sensors can detect ambient light levels, which can be used to control the fountain's lighting. For example, you can have the lights automatically turn on when it gets dark.

6. Ultrasonic Distance Sensor: Ultrasonic distance sensors can help measure the water depth or the distance to the water surface, which is useful for maintaining the water level in the fountain.

7. Water Flow Sensor: If you want to monitor the rate at which water is flowing through your fountain, you can use a water flow sensor. This can be useful for tracking water consumption.

8. Soil Moisture Sensor: If your fountain setup includes plants, you can use soil moisture sensors to monitor the moisture level in the soil. This can help automate the watering of plants.

9. Humidity Sensor: If you have sensitive electronic components in your smart fountain, a humidity sensor can help you monitor the humidity levels to prevent damage due to moisture.

10. Pressure Sensor: Pressure sensors can be used to monitor water pressure in the fountain system. This can be essential for controlling water flow and pressure in more complex fountains.

11. Wireless Connectivity: To make your fountain truly "smart," you may want to include wireless connectivity options such as Wi-Fi or Bluetooth to control and monitor the fountain remotely via a smartphone app or a web interface.

12. Water Purification Sensors: If your fountain includes a water purification system, you may need sensors to monitor the effectiveness of the purification process, such as detecting the presence of contaminants.

Python script for this Project:

```
import RPi.GPIO as GPIO

import time


# GPIO pins for water pump and water level sensor

WATER_PUMP_PIN = 17

WATER_LEVEL_PIN = 18


# Set the GPIO mode to BCM

GPIO.setmode(GPIO.BCM)


# Setup the water pump pin as an output

GPIO.setup(WATER_PUMP_PIN, GPIO.OUT)


# Setup the water level sensor pin as an input

GPIO.setup(WATER_LEVEL_PIN, GPIO.IN)


# Function to turn on the water pump

def turn_on_water_pump():

    GPIO.output(WATER_PUMP_PIN, GPIO.HIGH)

    print("Water pump is ON")


# Function to turn off the water pump

def turn_off_water_pump():

    GPIO.output(WATER_PUMP_PIN, GPIO.LOW)

    print("Water pump is OFF")


# Function to check the water level

def check_water_level():
    if GPIO.input(WATER_LEVEL_PIN) == GPIO.LOW:
        return "Low"
    else:
        return "High"

try:
    while True:
        current_level = check_water_level()
        print(f"Current water level: {current_level}")

        # Check the water level and turn on the pump if it's low
        if current_level == "Low":
            turn_on_water_pump()
        else:
            turn_off_water_pump()

        time.sleep(5)  # Check the water level every 5 seconds

except KeyboardInterrupt:
    GPIO.cleanup()
```