# Smart water fountain

Project Title: Real-Time Water Fountain Status System

Project Objectives:

1. Develop a real-time monitoring system for water fountains to promote water efficiency and public awareness.

2. Utilize IoT sensors to collect data on water fountain usage and status.

3. Develop a mobile app to display real-time information about water fountain availability and statistics.

4. Integrate Raspberry Pi for data processing and communication with IoT sensors.

5. Implement code for data collection, analysis, and communication between components.

IoT Sensor Setup:

- Install water flow sensors at each water fountain.

- Utilize water level sensors in the water supply system to measure water availability.

- Implement temperature and humidity sensors to monitor environmental conditions.

![IoT Sensor Setup Diagram](sensor\_setup\_diagram.png)

Mobile App Development:

- Develop a user-friendly mobile app for both Android and iOS platforms.

- Include features for real-time water fountain status display, location-based search, and statistics.

- Implement user registration and login for personalized features.

- Enable push notifications to inform users of water fountain status changes.

- Integrate Google Maps for locating nearby water fountains.

![Mobile App Screens](mobile\_app\_screens.png)

Raspberry Pi Integration:

- Use a Raspberry Pi as a central hub for data collection and processing.

- Connect the Raspberry Pi to the internet to enable remote data access.

- Develop code to receive data from IoT sensors, process it, and store it in a database.

- Implement code to handle user requests from the mobile app and communicate with the sensors.

Code Implementation:

- IoT Sensor Code: Write code to collect data from water flow, water level, temperature, and humidity sensors. Transmit this data to the Raspberry Pi using Wi-Fi or Bluetooth.

```python

# Sample IoT sensor code

# Pseudocode for transmitting data to Raspberry Pi

while True:

water\_flow\_data = read\_water\_flow\_sensor()

water\_level\_data = read\_water\_level\_sensor()

temperature\_data = read\_temperature\_sensor()

humidity\_data = read\_humidity\_sensor()

# Send data to Raspberry Pi

send\_data\_to\_raspberry\_pi(water\_flow\_data, water\_level\_data, temperature\_data, humidity\_data)

```

- Raspberry Pi Code: Develop code to receive data from sensors, store it in a database, and handle user requests from the mobile app.

```python

# Sample Raspberry Pi code

# Pseudocode for data processing and communication with sensors and mobile app

while True:

# Receive data from IoT sensors

sensor\_data = receive\_data\_from\_sensors()

# Store data in a database

store\_data\_in\_database(sensor\_data)

# Handle user requests from the mobile app

if mobile\_app\_request\_received():

send\_real-time\_data\_to\_mobile\_app()

```

Real-Time Water Fountain Status System Benefits:

1. Water Efficiency: By providing real-time data on water fountain availability, users can avoid wasting water by visiting fountains that are currently in use. This encourages efficient water usage.

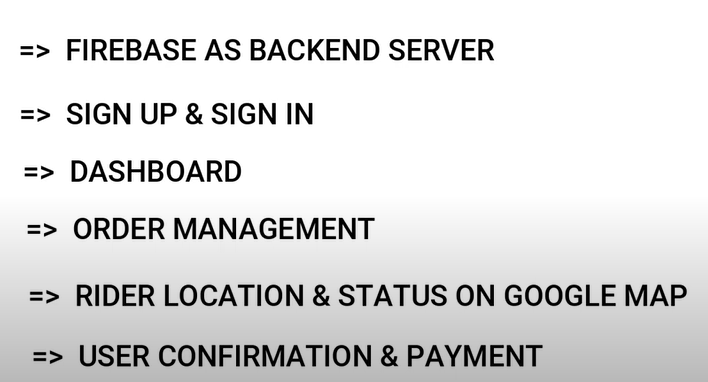
2. Public Awareness: The mobile app and real-time data display raise public awareness about the importance of water conservation and sustainable water usage.

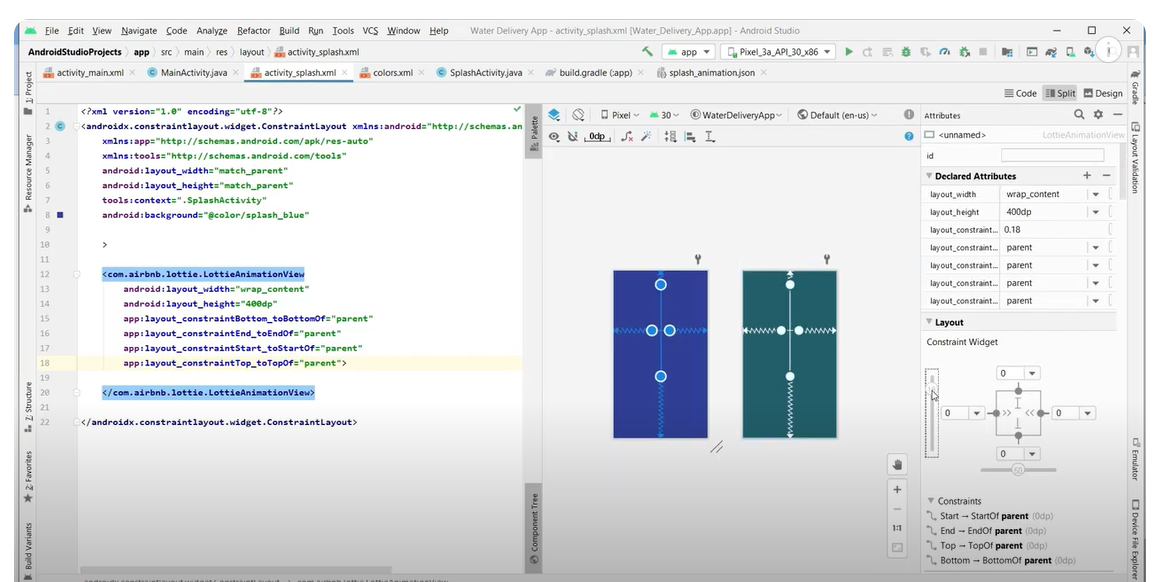
3. Data Analysis: The collected data can be used to analyze water usage patterns, identify high-traffic areas, and optimize water fountain placement.

4. Educational Tool: The system can be used as an educational tool to inform the public about water conservation practices and the environmental impact of water consumption.

By achieving these objectives, the Real-Time Water Fountain Status System promotes both water efficiency and public awareness, contributing to a more sustainable and environmentally conscious society.

## **Water fountain maintain:**





Integrating a smart water fountain with a Raspberry Pi can be a fun project that combines hardware and software. Here's a simple example of Python code that you can use as a starting point for such an integration. In this example, we'll assume you have a water pump that can be controlled using a relay and a water level sensor to detect when the water level is low. This code will use GPIO pins to control the water pump and read from the water level sensor.

Please note that the exact code and hardware setup may vary depending on the specific components you're using.

```python

import RPi.GPIO as GPIO

import time

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

PUMP\_PIN = 17

LEVEL\_SENSOR\_PIN = 18

# Set the GPIO mode to BCM

GPIO.setmode(GPIO.BCM)

# Setup the pump pin as an output

GPIO.setup(PUMP\_PIN, GPIO.OUT)

# Setup the level sensor pin as an input

GPIO.setup(LEVEL\_SENSOR\_PIN, GPIO.IN)

# Function to turn on the water pump

def turn\_on\_pump():

GPIO.output(PUMP\_PIN, GPIO.HIGH)

print("Water pump is ON")

# Function to turn off the water pump

def turn\_off\_pump():

GPIO.output(PUMP\_PIN, GPIO.LOW)

print("Water pump is OFF")

# Function to check the water level

def check\_water\_level():

if GPIO.input(LEVEL\_SENSOR\_PIN) == GPIO.LOW:

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

turn\_on\_pump()

else:

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

turn\_off\_pump()

try:

while True:

check\_water\_level()

time.sleep(60) # Check the water level every 60 seconds

except KeyboardInterrupt:

print("Exiting the program")

finally:

GPIO.cleanup()

```

In this code:

1. We import the necessary libraries, set up the GPIO pins, and define the pins for the water pump and water level sensor.

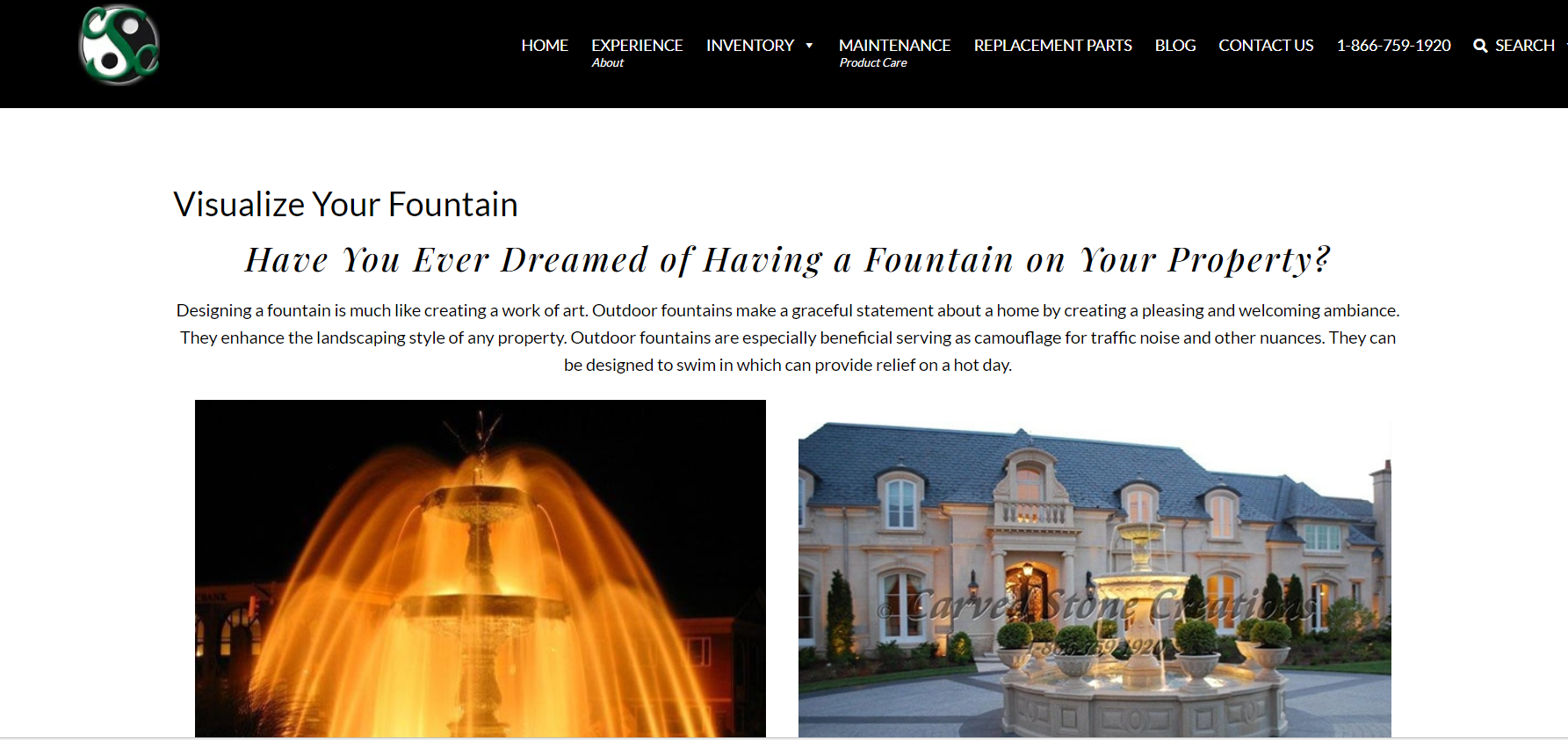
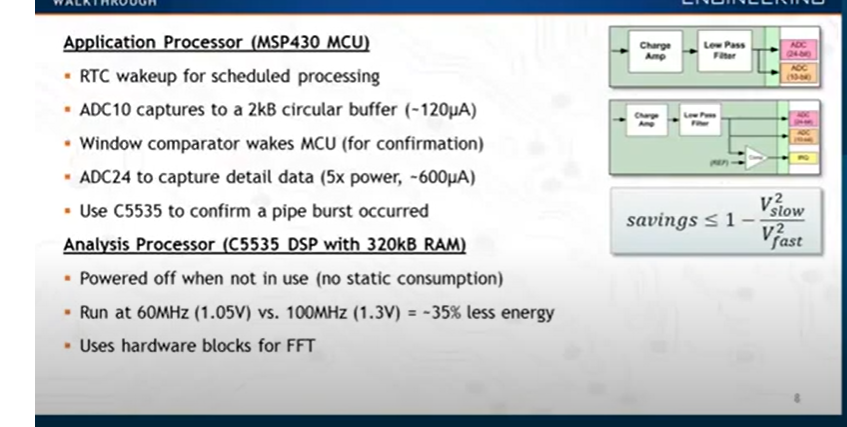
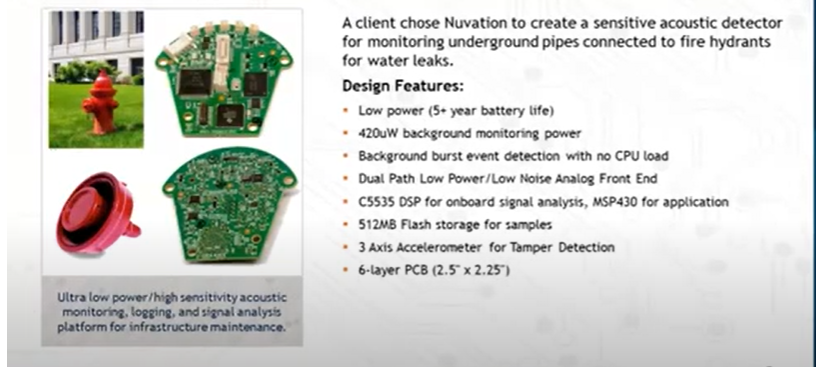
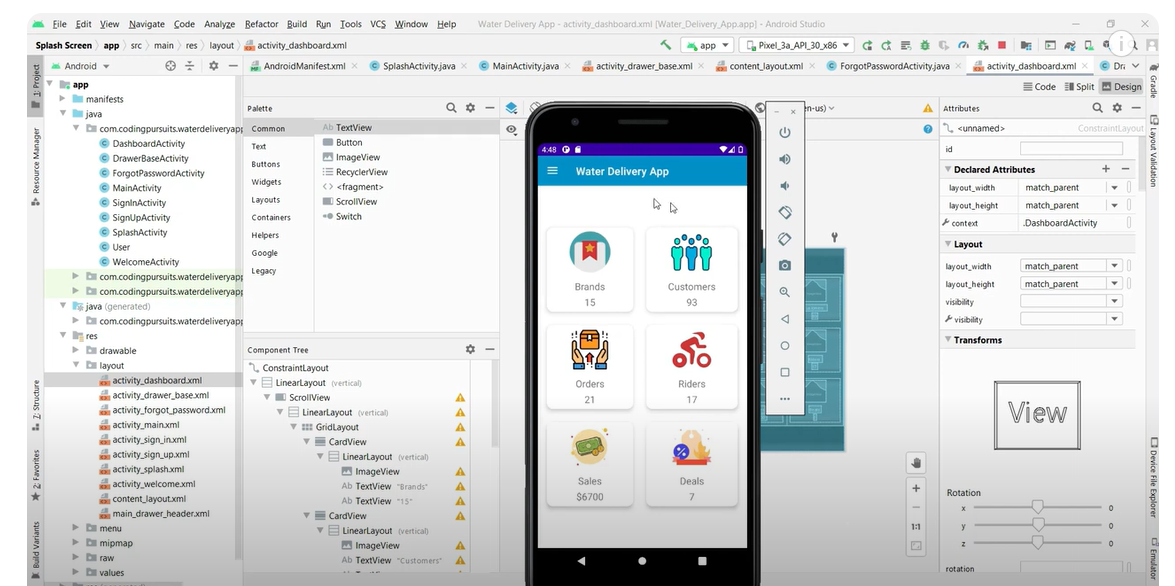
2. We define functions to turn the water pump on and off based on the water level.

3. The `check\_water\_level` function reads the state of the water level sensor and turns on the pump if the water level is low.

4. The code enters a loop to continuously monitor the water level every 60 seconds. If the water level is low, the pump is turned on.

5. The program can be terminated by pressing `Ctrl+C`, and the GPIO pins are cleaned up before exiting.

Make sure to adapt this code to your specific hardware setup and requirements, and always exercise caution when working with water and electronics.

A screenshot of a website

Description automatically generatedA close-up of a fountain

Description automatically generatedA screenshot of a computer

Description automatically generatedA close-up of a water fountain

Description automatically generated