NAME : TECHNOLOGY NAME : INTERNET OF THINGS

PROJECT SMART WATERFOUNTAINS

NAME : BALAGAJARAJ P

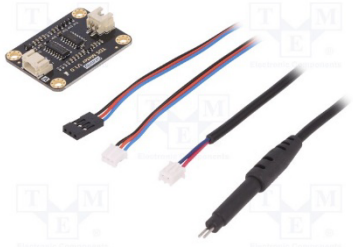
OBJECTIVES:

objective of the "Smart Water Fountain" project is to create an innovative and efficient water fountain system that utilizes IoT (Internet of Things) technology to monitor and manage the fountain's operation. The project aims to provide real-time data on water quality, consumption, and system health, all accessible through a mobile app. It will also enable remote control and automation of the water fountain.

IoT Sensor Setup:

The IoT sensor setup for the Smart Water Fountain includes various sensors and devices to collect and transmit data. These sensors can include

Water Quality Sensor:



Measures parameters like pH levels, turbidity, and temperature to ensure water is safe and clean for consumption.

Flow Sensor:

Monitors water flow to track consumption

and detect leaks or abnormal usage.

Water Level Sensor:

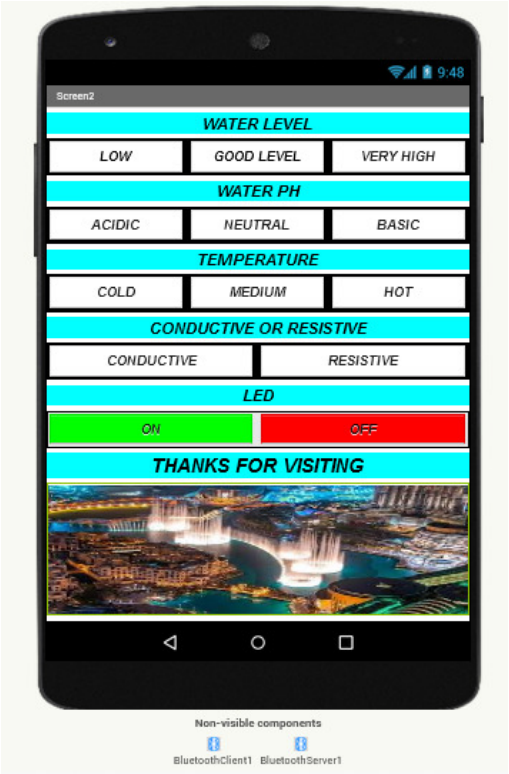
Measures the water level to prevent overflows and ensure consistent water supply.

Mobile App Development:

The mobile app for the Smart Water Fountain is a key component of the project, allowing users to interact with the fountain remotely. The app involves the following aspects:

User Interface (UI): Create an intuitive and user-friendly interface to control and monitor the water fountain.

Mobile Platform: Develop the app for iOS and Android platforms to reach a wider user base.



Features:

//Real-time monitoring of water quality and quantity.

//Notification system for alerts or abnormal conditions.

//Historical data visualization.

// User authentication and security features.

Code Implementation:

The code for the Smart Water Fountain project involves several components:

Sensor Data Collection: Write code to interface with and read data from each sensor connected to the Raspberry Pi.

Data Processing: Analyze and process the sensor data to detect anomalies, calculate water quality, and monitor consumption.

Communication: Implement code for secure communication with the mobile app and cloud server for data transmission.

Automation: Develop logic for controlling the fountain's operation, such as turning it on and off based on user commands or sensor data.

User Interface: Code the user interface for the mobile app, ensuring it connects to the Raspberry Pi for control and data retrieval.

Error Handling and Logging: Implement error-handling mechanisms and logging to troubleshoot and maintain the system effectively.

To display real-time water fountain data, including water flow rate and malfunction alerts by python

```
import time


def initialize_flow_sensor()
```

```
    pass

def read_flow_rate(sensor):

    flow_rate = sensor.read()
    return flow_rate

def control_fountain(flow_rate)
    pass

if __name__ == '__main__':
    flow_sensor = initialize_flow_sensor()

    try:
        while True:
            flow_rate = read_flow_rate(flow_sensor)
            control_fountain(flow_rate)
            time.sleep(1)  # Adjust the monitoring interval as needed

    except KeyboardInterrupt:
  pass
```

Python Code For Smart Water Fountain :

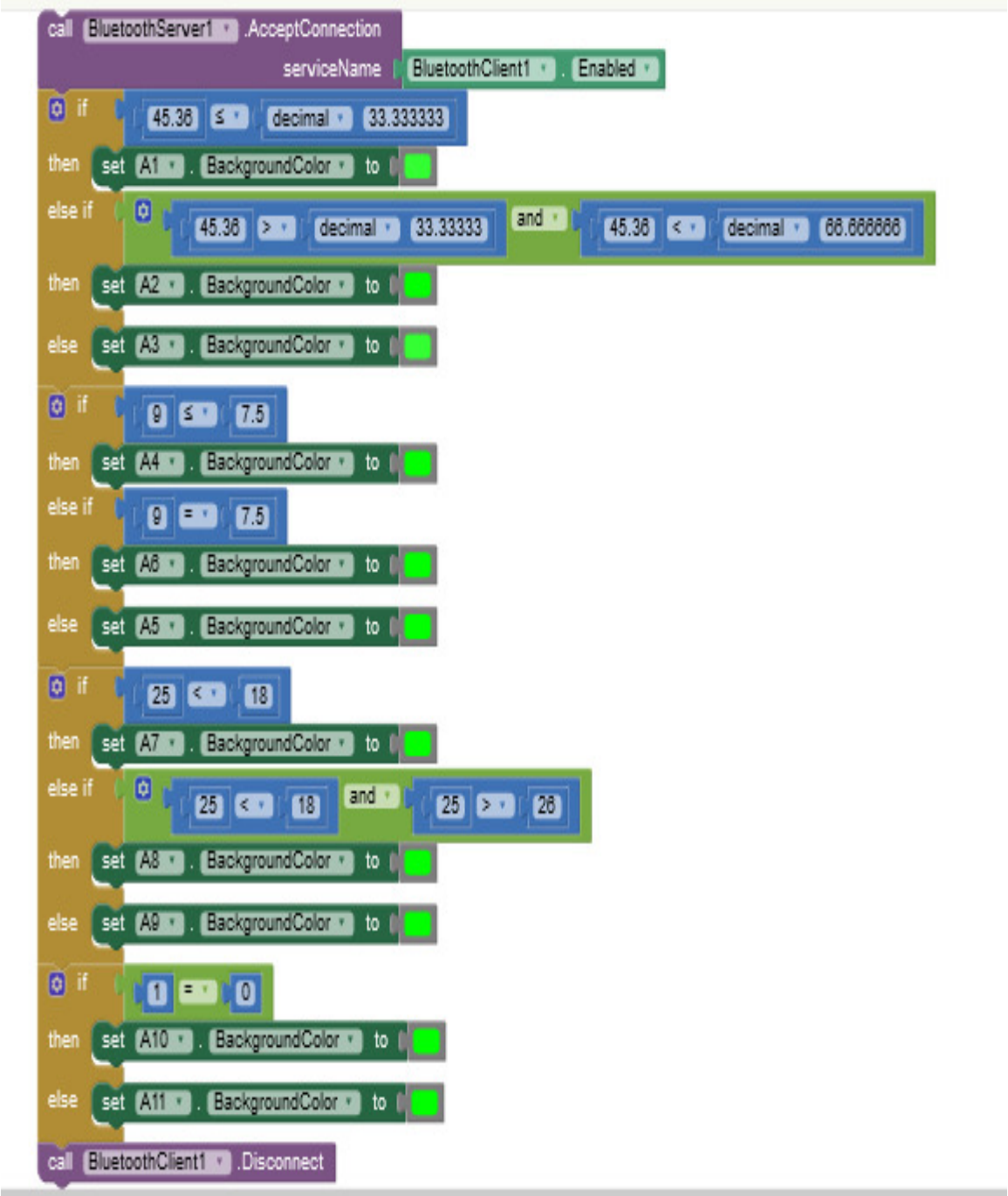
```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.button import Button
```

```
import socket

class SmartWaterFountainApp(App):

    def build(self):
        self.layout = BoxLayout(orientation='vertical')

        # Create a button to send commands to the Raspberry Pi
        self.control_button = Button(text='Start Fountain')
        self.control_button.bind(on_press=self.send_command)
        self.layout.add_widget(self.control_button)

        # Create a label for feedback
        self.feedback_label = Label(text='')
        self.layout.add_widget(self.feedback_label)

        return self.layout

    def send_command(self, instance):
        # Establish a connection to the Raspberry Pi server
        host = 'raspberry_pi_ip'  # Replace with the Raspberry Pi's IP address
        port = 12345
        command = 'START_FOUNTAIN'  # Replace with your command
```

```
        client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        client_socket.connect((host, port))

        # Send the command to the Raspberry Pi
        client_socket.send(command.encode('utf-8'))
        client_socket.close()

        self.feedback_label.text = f'Sent command: {command}'

if __name__ == '__main__':
    SmartWaterFountainApp().run()
Import RPi.GPIO as GPIO
Import time
From AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

# Set up AWS IoT
myMQTTClient = AWSIoTMQTTClient(“myClient”)
myMQTTClient.configureEndpoint(“your-iot-endpoint.amazonaws.com”,
8883)
myMQTTClient.configureCredentials(“root-CA.crt”, “private-key.pem”,
“cert.pem”)

# Set up GPIO pin for your water level sensor
Water_level_pin = 17
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(water_level_pin, GPIO.IN)

# Function to publish data to AWS IoT
Def publish_data():
   Water_level = GPIO.input(water_level_pin)
   Message = {
      “water_level”: water_level
   }
   myMQTTClient.publish(“water-fountain-status”, json.dumps(message),
1)

# Connect to AWS IoT
myMQTTClient.connect()

# Main loop to continuously send data
Try:
   While True:
      Publish_data()
      Time.sleep(10)  # Send data every 10 seconds

Except KeyboardInterrupt:
   Pass

# Clean up GPIO and disconnect from AWS IoT
GPIO.cleanup()
```

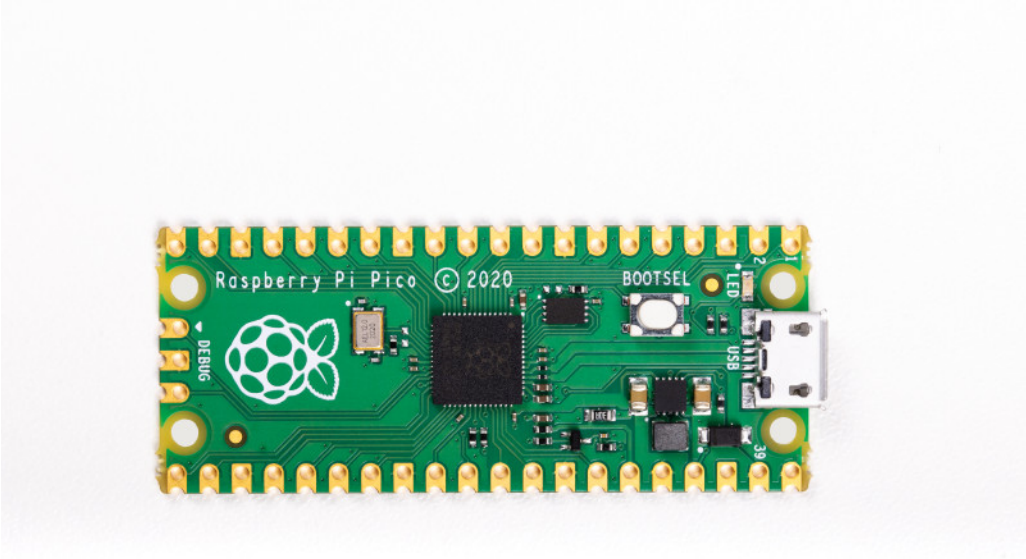
myMQTTClient.disconnect()



ensure you have Kivy and its dependencies installed on your

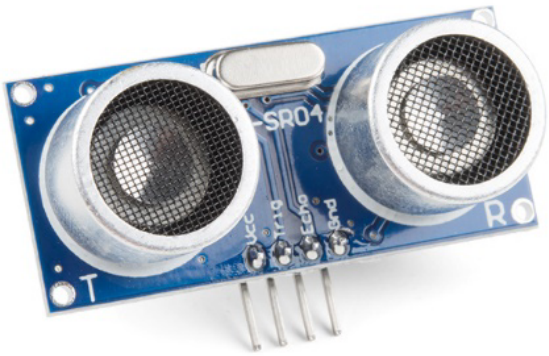
Raspberry Pi:

Hardware used for smart watrer fountain:



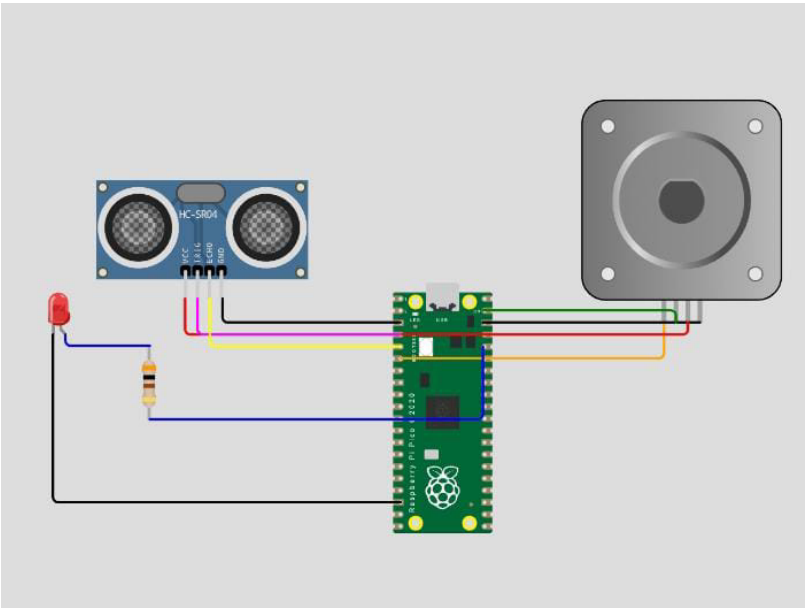
RASPBERRY PI

Raspberry Pi (e.g., Raspberry Pi 4 or 3) or other microcontrollers can serve as the brain of the system, controlling various components and running the necessary software.



HC-SR04

SIMULATION DIAGRAM:



Public Awarenences:

Water Quality Control: The system can monitor and maintain water quality, ensuring that the water dispensed from the fountain is safe and clean. This can encourage the public to use the fountain more confidently, reducing the need for single-use plastic water bottles.

Usage Data: Tracking water consumption through the system provides valuable data on usage patterns. This data can be used to optimize the operation of the water fountain, adjusting factors such as flow rate or hours of operation to minimize water wastage.

Remote Control and Automation: Real-time status monitoring allows for remote control and automation. For example, the fountain can be turned off during non-peak hours or adjusted based on weather conditions to prevent unnecessary water use.

Alerts and Notifications: The system can send alerts and notifications to users or maintenance personnel in case of issues or emergencies, such as low water levels, high usage, or water quality concerns. This immediate feedback can help address problems promptly.

Public Education: Displaying real-time water status information on the mobile app or dedicated screens near the fountain can educate the public about water conservation. When users can see the direct impact of their choices, they may be more motivated to use water responsibly.

Historical Data Analysis: The system can collect historical data on water consumption and fountain usage. This data can be used for analysis and reporting, helping organizations and local authorities make informed decisions about water resource management and infrastructure improvements.

THANK YOU