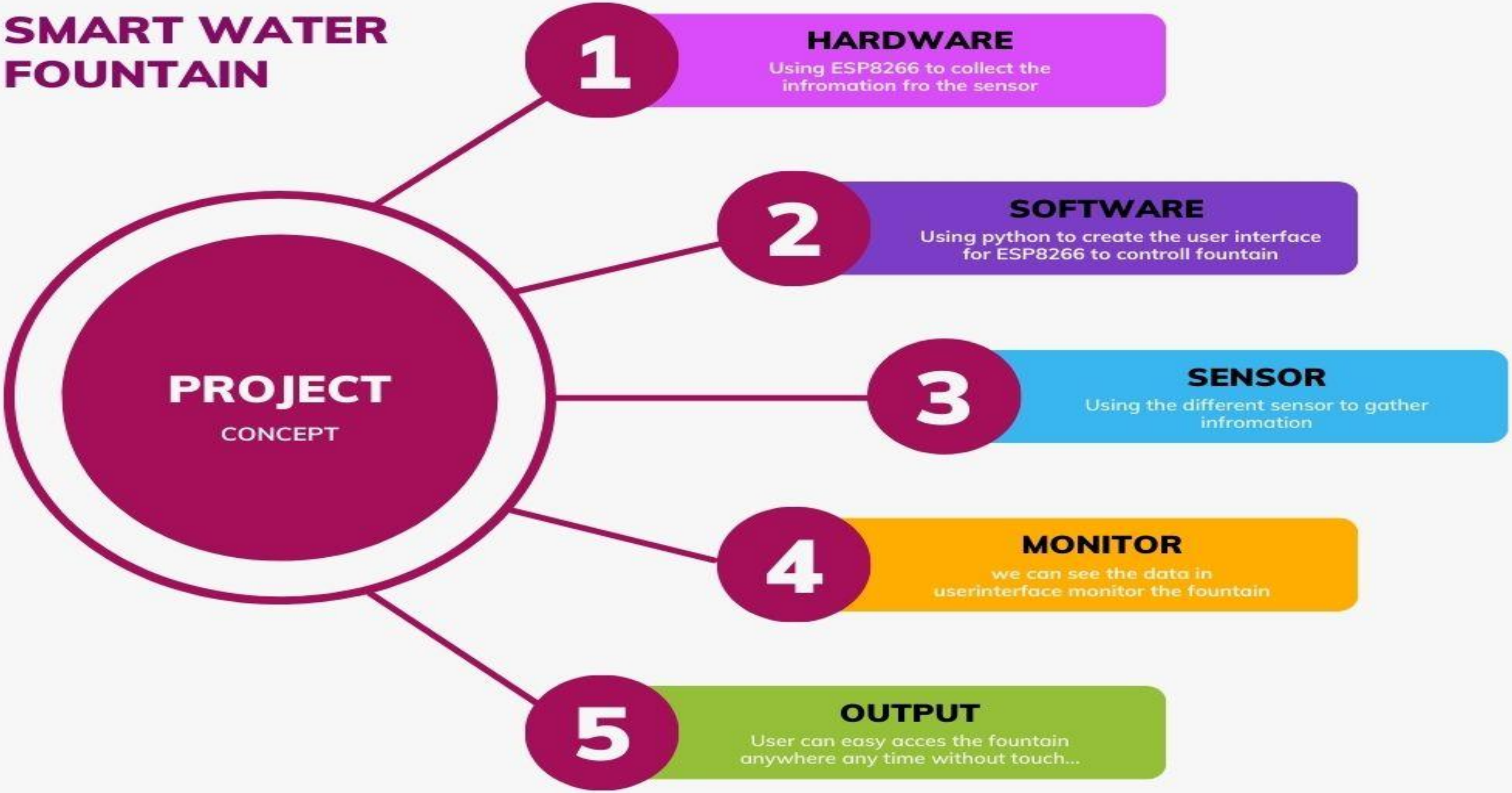
SMART WATER FOUNTAIN

USING (IOT)

SMART WATER FOUNTAIN



PROJECT
CONCEPT

1. **HARDWARE**
Using ESP8266 to collect the infromation fro the sensor

2. **SOFTWARE**
Using python to create the user interface for ESP8266 to controll fountain

3. **SENSOR**
Using the different sensor to gather infromation

4. **MONITOR**
we can see the data in userinterface monitor the fountain

5. **OUTPUT**
User can easy acces the fountain anywhere any time without touch...

Project Description

- The Smart Water Fountain project aims to design and develop an intelligent, eco-friendly water feature for outdoor public spaces, such as parks, gardens, and urban plazas. This innovative fountain incorporates IoT technology to provide not only an aesthetic and relaxing element but also serves as a sustainable water source for visitors. The project leverages various sensors and a user-friendly interface to enhance the overall user experience while focusing on water conservation and environmental impact

- The combination of these components will empower our Smart Water Fountain to offer a sophisticated and sustainable solution for delivering fresh water while enhancing user interaction and promoting water conservation.

NORMAL WATER FOUNTAIN

- The purposes of water fountains are diverse. They serve as sources of potable water, providing refreshment and hydration for people passing by. In public spaces, fountains often become meeting points and gathering spots, enhancing social interactions and community engagement. The visual and auditory aspects of a water fountain, with the gentle splashing of water and the play of light and reflections, create a serene and calming atmosphere, making them a popular choice for adding aesthetic appeal to various environments

- In this water fountain are working base on human control system,if any water lag or any problem in fountain can't slove quick because the fault find is fountain is hard..but SMART WATER FOUNTAIN overcome this problem.

SOFTWARE FOR ESP8266

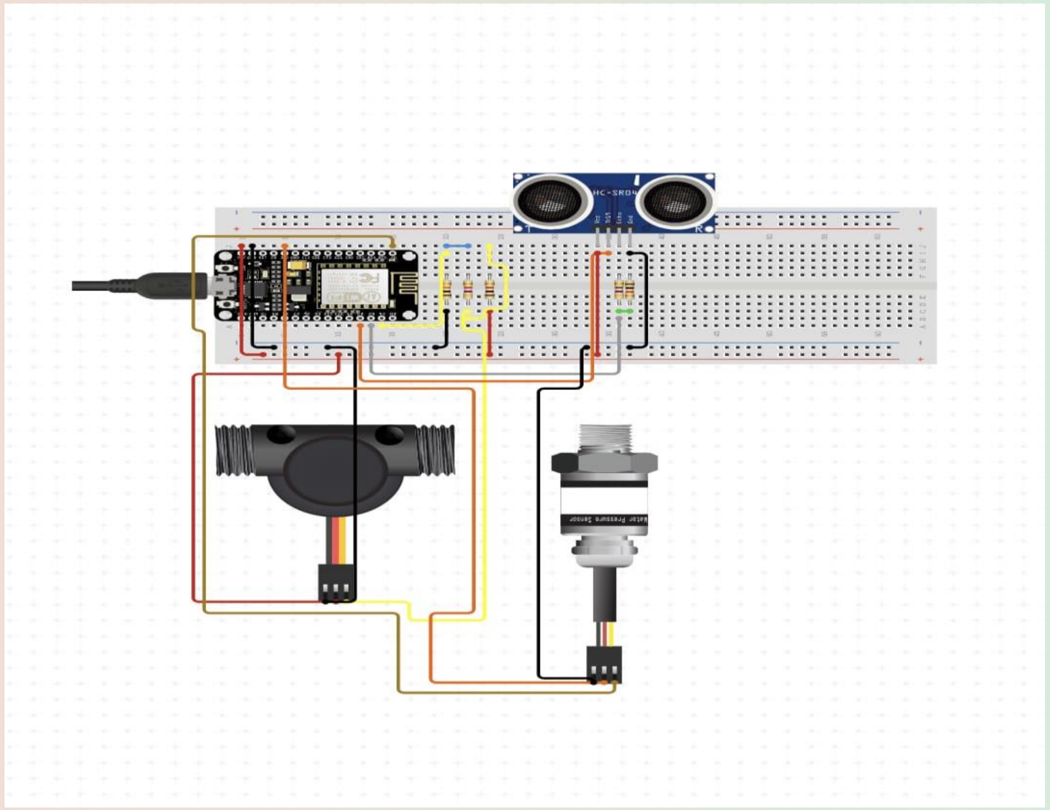
- MicroPython is a compact implementation of the Python programming language that is designed to run on microcontrollers and embedded systems. When combined with hardware like the ESP8266, it provides a versatile platform for IoT (Internet of Things) development. Here's an overview of MicroPython on the ESP8266

- We are using MICROPYTHON software to control the the ESP8266. Its using to collect the all information from the sensor to process with help of micropython.

- This micropython software are bulid the Userinterface platform to monitor the all data in sensor through the microcontroller.

HARDWARE SETUP AND GUI

SMART WATER FOUNTAIN



Coding sensor using MICROPYTHON

```
FLOW_SENSOR_PIN = 4  # Change this to the appropriate GPIO pin

# Initialize the flow sensor
flow_sensor = machine.Pin(FLOW_SENSOR_PIN, machine.IN)

# Constants for flow rate calculation
FLOW_CALIBRATION_FACTOR = 7.5  # Adjust this factor based on your flow sensor

# Variables for flow rate calculation
pulse_count = 0
last_time = time.ticks_ms()

# Main loop to continuously monitor flow rate
while True:
    if flow_sensor.value() == 1:
        pulse_count += 1

    # Calculate flow rate every 1 second (adjust as needed)
    if time.ticks_diff(time.ticks_ms(), last_time) >= 1000:
        flow_rate = pulse_count / FLOW_CALIBRATION_FACTOR
        print("Flow Rate: {:.2f} L/min".format(flow_rate))

        # Reset pulse count and update last time
        pulse_count = 0
        last_time = time.ticks_ms()
```

```
pressure_sensor_pin = machine.Pin(4, machine.Pin.IN)

def read_pressure():
    # Read the analog voltage from the pressure sensor
    analog_value = machine.ADC(pressure_sensor_pin).read()

    # Convert the analog value to pressure (you may need to calibrate this)
    pressure = analog_value  # You should calibrate this based on your sensor's specifications

    return pressure

while True:
    pressure = read_pressure()
    print("Pressure: {} Pa".format(pressure))
    time.sleep(1)
```

```
# Define GPIO pins for the ultrasonic sensor
TRIGGER_PIN = 12  # GPIO 12 for trigger
ECHO_PIN = 13    # GPIO 13 for echo

# Initialize the ultrasonic sensor
trigger = machine.Pin(TRIGGER_PIN, machine.Pin.OUT)
echo = machine.Pin(ECHO_PIN, machine.Pin.IN)

def get_distance():
    # Trigger a pulse to start the measurement
    trigger.value(1)
    time.sleep_us(10)
    trigger.value(0)

    # Wait for the echo pin to go high (start of the pulse)
    while echo.value() == 0:
        pass
    start_time = time.ticks_us()

    # Wait for the echo pin to go low (end of the pulse)
    while echo.value() == 1:
        pass
    end_time = time.ticks_us()

    # Calculate the duration of the pulse and convert it to distance (in cm)
    pulse_duration = time.ticks_diff(end_time, start_time)
    distance_cm = pulse_duration / 58  # Speed of sound is approximately 343 m/s

    return distance_cm

while True:
    distance = get_distance()
    print("Distance: {} cm".format(distance))
    time.sleep(1)
```

MicroPython Server Code

```
import network
import socket
import ure
import ujson
import time


# Set your Wi-Fi credentials
SSID = "Your_WiFi_SSID"
PASSWORD = "Your_WiFi_Password"


# Initialize Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(SSID, PASSWORD)
while not wifi.isconnected():
    pass


# Create a web server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)


# Read sensor data
def read_sensor_data():
    # Replace with code to read sensor data from your sensors
    flow_rate = 0.0
    pressure = 0.0
    water_level = 0.0
    return {
        'flowRate': flow_rate,
        'pressure': pressure,
        'waterLevel': water_level
    }
```

HTML User Interface

```
<!DOCTYPE html>
<html>
<head>
  <title>Smart Water Fountain</title>
</head>
<body>
  <h1>Smart Water Fountain Control</h1>

  <h2>Control Fountain</h2>
  <button id="startBtn">Start Fountain</button>
  <button id="stopBtn">Stop Fountain</button>
  <button id="refillBtn">Refill Water</button>

  <h2>Water Fountain Status</h2>
  <p>Flow Rate: <span id="flowRate">N/A</span> L/min</p>
  <p>Pressure: <span id="pressure">N/A</span> PSI</p>
  <p>Water Level: <span id="waterLevel">N/A</span> cm</p>

  <script>
    document.getElementById("startBtn").onclick = function() {
      // Code to start the fountain in MicroPython
      // Send a request to the ESP8266 to start the fountain
    };

    document.getElementById("stopBtn").onclick = function() {
      // Code to stop the fountain in MicroPython
      // Send a request to the ESP8266 to stop the fountain
    };

    document.getElementById("refillBtn").onclick = function() {
      // Code to trigger water refill in MicroPython
      // Send a request to the ESP8266 to refill the water reservoir
    };

    // Periodically update sensor data using JavaScript (AJAX)
    function updateSensorData() {
      var xhr = new XMLHttpRequest();
      xhr.open("GET", "/sensor_data", true);
```

CODING FOR SMART WATER FOUNTAIN

```
import machine
import time
import network
import urequests
import uhttpd
from machine import ADC
import dht

# Set your Wi-Fi credentials
SSID = "Your_WiFi_SSID"
PASSWORD = "Your_WiFi_Password"

# Initialize Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(SSID, PASSWORD)
while not wifi.isconnected():
    pass

# Initialize Sensors (adjust pin numbers and
sensor settings)
flow_rate_sensor = machine.Pin(12,
machine.IN)  # Adjust the GPIO pin
pressure_sensor = ADC(0)  # ADC0 for pressure
sensor
ultrasonic_trigger = machine.Pin(5, machine.OUT)
ultrasonic_echo = machine.Pin(4, machine.IN)

# Initialize DHT sensor for temperature and
humidity
dht_sensor = dht.DHT22(machine.Pin(14))  #
Adjust the GPIO pin
```

```
# Function to read flow rate sensor (adjust as needed)
def read_flow_rate_sensor():
    return 0.0  # Replace with actual code to read the flow rate sensor

# Function to read pressure sensor (adjust as needed)
def read_pressure_sensor():
    return pressure_sensor.read()

# Function to read ultrasonic sensor (adjust as needed)
def read_ultrasonic_sensor():
    ultrasonic_trigger.value(1)
    time.sleep_us(10)
    ultrasonic_trigger.value(0)
    while not ultrasonic_echo.value():
        pass
    start_time = time.ticks_us()
    while ultrasonic_echo.value():
        pass
    end_time = time.ticks_us()
    duration = end_time - start_time
    # Convert duration to distance (adjust for speed of sound and units)
    distance = duration / 58.0
    return distance

# Function to read temperature and humidity from DHT sensor (adjust as needed)
def read_dht_sensor():
    dht_sensor.measure()
    return dht_sensor.temperature(), dht_sensor.humidity()
```

CREATING A WEBSITE

```
# Create a web page template
html_template = """
<!DOCTYPE html>
<html>
<head>
  <title>Smart Water Fountain</title>
</head>
<body>
  <h1>Smart Water Fountain</h1>
  <p>Flow Rate: {flow_rate} L/min</p>
  <p>Pressure: {pressure} PSI</p>
  <p>Water Level: {water_level} cm</p>
</body>
</html>
"""
```

CREATING A SERVER

```
# Create a web server
server = uhttpd.Server()

def request_handler(request, response):
  # Read sensor data
  flow_rate = read_flow_rate_sensor()
  pressure = read_pressure_sensor()
  water_level = read_ultrasonic_sensor()
  temperature, humidity = read_dht_sensor()

  # Create HTML response with sensor data
  response.write(html_template.format(flow_rate=flow_rate, pressure=pressure, water_level=water_level,
temperature=temperature, humidity=humidity))

# Add the request handler to the server
server.add_handler('/', request_handler)

# Start the server
server.listen(80)

# Main loop
while True:
  pass
  xhr.onload = function() {
        if (xhr.status == 200) {
          var data = JSON.parse(xhr.responseText);
          document.getElementById("flowRate").textContent = data.flowRate;
          document.getElementById("pressure").textContent = data.pressure;
          document.getElementById("waterLevel").textContent = data.waterLevel;
        }
      };

      xhr.send();
    }

    // Update sensor data every 5 seconds (adjust as needed)
    setInterval(updateSensorData, 5000);
  </script>
</body>
</html>
```

```
# Handle client requests
while True:
    conn, addr = s.accept()
    request = conn.recv(1024)
    request = str(request, 'utf-8')

    # Serve the HTML page
    if ure.search("GET / HTTP/1.1", request):
        html_file = open('index.html', 'r')
        conn.send("HTTP/1.1 200 OK\n")
        conn.send("Content-Type: text/html\n\n")
        conn.sendall(html_file.read())
        html_file.close()

    # Provide sensor data
    if ure.search("GET /sensor_data HTTP/1.1", request):
        sensor_data = read_sensor_data()
        conn.send("HTTP/1.1 200 OK\n")
        conn.send("Content-Type: application/json\n\n")
        conn.sendall(ujson.dumps(sensor_data))

    conn.close()
```

CONCLUSION

- The user interface aspect of the project enhances the overall experience, enabling users to adjust water temperature, track water consumption, ad receive real-time feedback, creating a more user-centric and interactive water fountain. The ESP8266's Wi-Fi capabilities further extend the project's reach, allowing for remote monitoring and control, making it a valuable addition to IoT-enabled spaces.

- Smart water fountains not only provide a convenient source of hydration but also contribute to water conservation efforts and sustainability. By monitoring water levels, ensuring water quality, and controlling the flow rate, the project promotes responsible water usage. It also encourages user engagement and awareness, fostering a sense of responsibility towards environmental conservation.

- In this endeavor, MicroPython proves to be a powerful and accessible tool for developing IoT projects on constrained devices like the ESP8266. Its simplicity and ease of use make it an excellent choice for programming the ESP8266 microcontroller while allowing developers to harness the full potential of the smart water fountain concept.