SMART WATER MANAGEMENT

PHASE 3 SUBMISSION

Building a Smart Water Management project using IoT devices:

Hardware Setup:

Choose the necessary IoT devices, such as water sensors,microcontrollers (e.g., Raspberry Pi, Arduino), and communicationmodules (e.g., Wi-Fi, LoRa).Connect the water sensors to themicrocontrollers.

Network Connectivity:

Ensure your IoT devices can connect to theinternet. You may need Wi-Fi or other connectivity options.

Data Collection:

Write Python code on the IoT devices to collect data from the water sensors.Use appropriate libraries orAPIs to read sensor data.

Data Processing:

Develop Python scripts to process the collected data. This could include data filtering, calibration, and error handling.

Data Transmission:

Establish a method for sending the processed data to a central server or cloud platform. MQTT, HTTP, or other protocols may be used.

Cloud Platform Setup:

Set up a cloud platform like AWS, Azure, or Google Cloud to receive and store the data.Create databases to store the water sensor data.

Data Storage:

Write Python scripts on the cloud platform to receive and store the incoming data.

Data Analysis and Visualization:

Develop Python scripts to analyze and visualize the data. You can use libraries like Pandas, Matplotlib, or Plotly.

Alerts and Notifications:

Implement alerting mechanisms in your Python code to notify relevant parties if certain water conditions are met.

User Interface:

Create a user interface for monitoring the water data. You can build a web application or a mobile app using Python frameworks like Flask or Django.

Security:

Implement security measures to protect data transmission and storage, including encryption and access control.

Testing and Calibration:

Thoroughly test the system, calibrate sensors,and ensure that it functions as expected.

Maintenance and Monitoring:

Set up monitoring for your IoT devices and the cloud platform to detect and resolve issues promptly.

Documentation:

Document your project thoroughly, including hardware configurations, software code, and setup instructions.

Scaling and Optimization:

Consider how to scale your system if needed and optimize the code and infrastructure for efficiency.

Building a python script for smart water management:

```
import time
import random
import requests
# Replace with the actual sensor reading logic
def read_water_sensor():
# Simulating a water consumption value
return random.uniform(0.1, 2.0) # Liters per minute
# Replace with the actual API endpoint to send data
API_ENDPOINT = "https://yourserver.com/api/waterconsumption"
# Main loop to collect and send data
while True:
try:
# Read water consumption data from the sensor
water_consumption = read_water_sensor()
# Create a data payload to send to the server
data = {
"location": "Park 1", # Replace with the actual location
"water_consumption": water_consumption
}
# Send data to the central server
response = requests.post(API_ENDPOINT, json=data)
if response.status_code == 200:
```

```
print(f"Data sent successfully: {data}")

else:

print(f"Failed to send data: {response.status_code}")

# Adjust the time interval based on your data collection requirements

time.sleep(60) # Collect data every minute

except Exception as e:

print(f"An error occurred: {str(e)}")
```

This script simulates water consumption data and sends it to a server. Here's what it does:

It defines a function read_water_sensor to simulate reading water consumption from a sensor. In a real-world scenario, you would replace this with code to read data from actual sensors.

It sets the API_ENDPOINT to the URL where you want to send the data. You should replace this with the actual endpoint of your central server.

In the main loop, it reads water consumption data, creates a data payload with the location and water consumption value, and sends this data to the central server using a POST request.

The script includes error handling to handle exceptions that may occur during data collection or transmission.

It repeats this process at a specified time interval (e.g., every minute). We would use appropriate libraries and communication protocols to interface with IoT device's sensors and ensure secure data transmission.