IoT-Based

Smart water management

Objective:

Consider incorporating machine learning algorithms to analyze water consumption patterns and provide conservation suggestion

Components Needed:

1. Arduino or Raspberry Pi: The central control unit that manages the sensors and actuators.
2. Flow Sensors: To measure water consumption accurately.
3. Solonoid Valves: To control the flow of water.
4. Wi-Fi Module (e.g., ESP8266): For connecting the system to the internet.
5. Tinkercad or Tinkercad Circuits: To simulate and design the system virtually.
6. Web-based Dashboard: To monitor and control water usage remotely.
7. Power Supply: Batteries or a power source for the IoT devices.

Steps to Implement:

System Design and Simulation:

Use Tinkercad to design and simulate the smart water management system. Create a virtual representation of all the components.

Flow Sensor Integration:

Connect flow sensors to the central control unit (Arduino or Raspberry Pi). These sensors will measure water usage in real-time.

Solonoid Valve Integration:

Connect solenoid valves to the central control unit. These valves will control the flow of water to different outlets.

IoT Connectivity:

Integrate the Wi-Fi module (e.g., ESP8266) with your control unit to enable internet connectivity.

Data Collection:

Set up data collection from flow sensors. Record water consumption data at regular intervals and send it to the cloud.

Cloud Storage and Analysis:

Use platforms like AWS, Google Cloud, or Azure to store and analyze the data. You can calculate water usage patterns and detect anomalies.

Web-Based Dashboard:

Create a user-friendly web-based dashboard that allows users to monitor water usage, set usage limits, and control the solenoid valves remotely.

Mobile App Integration (Optional):

Develop a mobile app that provides users with real-time updates and control over the water management system from their smartphones.

Alerts and Notifications:

Implement email or SMS alerts for users when water usage exceeds predefined limits or if there are leaks detected.

Energy Efficiency:

Optimize the system for energy efficiency to reduce power consumption.

Testing and Calibration:

Thoroughly test the system to ensure accurate measurement and control of water usage. Calibrate the sensors if necessary.

Documentation and Presentation:

Document your project thoroughly, including the design, implementation, and testing phases. Prepare a presentation to explain the project to your audience.

Sustainability and Environmental Impact:

Consider the environmental benefits of your smart water management system, such as water conservation and reduced energy usage.

Demonstration:

Present your project to your class or instructors and demonstrate how it works, highlighting its benefits for water conservation and efficiency.

This project not only introduces students to IoT, sensor technology, and programming but also promotes sustainability by addressing a critical issue - water management. It allows students to apply their technical skills to real-world problems and showcases the potential of technology in addressing environmental challenges.

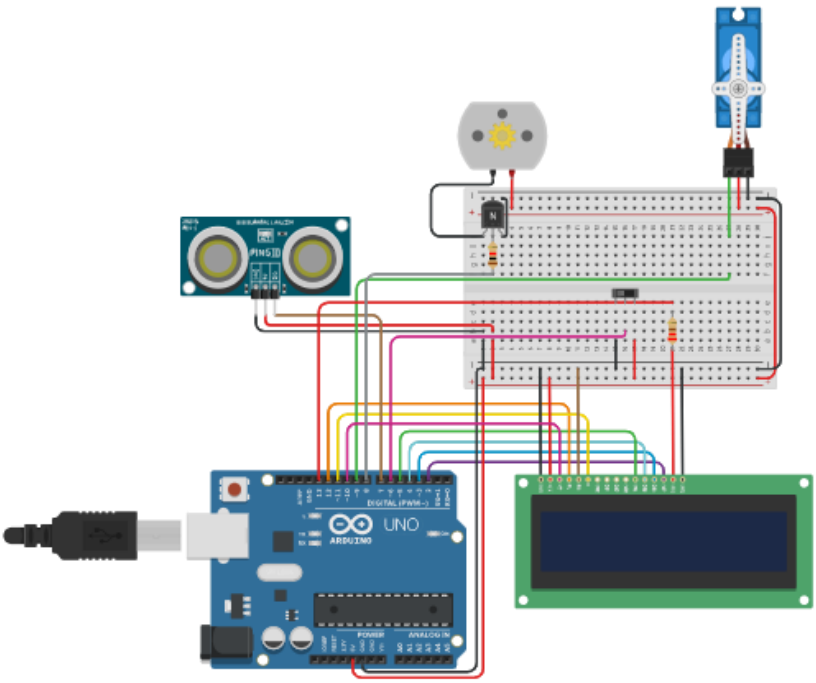
Python code:

Smart water management systems often involve sensors, data collection, and decision-making algorithms. Below is a simplified example of Python code for a basic smart water management system using random data generation to simulate sensor readings: Python import random import time

```
class WaterSensor:
    def __init__(self, location):
        self.location = location self.level = 0

    def generate_reading(self):
        # Simulate water level readings (0-100) self.level =
        random.randint(0, 100)

class WaterManagementSystem:
    def __init__(self): self.sensors =
        [] def add_sensor(self,
```

```
        sensor):
        self.sensors.append(sensor)

    def monitor_water_levels(self): while
        True:
            for sensor in self.sensors: sensor.generate_reading()
                print(f"Location: {sensor.location}, Water Level: {sensor.level}%") if
                sensor.level > 90:
                    self.alert_high_water_level(sensor.location)
            time.sleep(5) # Simulate readings every 5 seconds

    def alert_high_water_level(self, location):
        # Replace this with your alerting mechanism (e.g., sending notifications) print(f"ALERT: High
        water level detected at {location}!")

if __name__ == "__main__": system =
    WaterManagementSystem()

    # Add water sensors with locations sensor1
    = WaterSensor("Kitchen") sensor2 =
    WaterSensor("Bathroom")
    system.add_sensor(sensor1)
    system.add_sensor(sensor2)

    # Start monitoring water levels system.monitor_water_levels()
```

Smart Water Management Circuit diagram:



Smart Water Management 3D Representation: