SMART WATER MANAGEMENT SYSTEM

FINAL PROJECT

PHASE-V

. Name : MOHAMED AHIL A

Nm.no: au813821106069

Mail.id:mohamedahil.a2004@gmail.com

OUTLINE:

Introduction

Project goals and Abjective

Stakeholder analysis

System overview

Sensor selection and .. justification

Hardware and Software . . . . ……………… requirements

Block Diagram

Circuit Diagram

Algorithm

Data collection plan

Code implementation

Result and analysis

Web interface integration

Server setup

User interaction

Applications

Conclusion

INTRODUCTION:

The Water System Management Project is a comprehensive undertaking that leverages IoT and advanced sensor technologies for efficient water resource management. This document outlines the key phases of the project, focusing on the implementation of cutting-edge sensors, data analysis, and the integration of a user-friendly web interface. It highlights the project's objectives, methodologies, and the significance of its contributions to sustainable water management practices.

GOALS:

Implement a robust water system management solution utilizing IoT technologies.

Develop a comprehensive sensor network for real-time monitoring of water parameters.

Create an efficient data processing pipeline to analyze and interpret sensor data accurately.Integrate a user-friendly web interface for accessing and visualizing real-time water system data.Optimize water resource management practices for improved sustainability and conservation efforts.

OBJECTIVE:

Monitor water levels in tanks and reservoirs using ultrasonic sensors.

Analyze water quality through simulated turbidity and pH sensors.

Track water flow rates with simulated water flow sensors.

Develop a filtering process to ensure the purity of water for various applications.

Create a user interface to display data and provide insights for informed decision-making.

Stakeholder Analysis:

Primary Stakeholders:

1.Water Utility Management

2.Environmental Regulatory Authorities

3.End Users (Consumers)

Secondary Stakeholders:

1.Local Communities

2.Technology Providers

3.Research Institutions

Tertiary Stakeholders:

1.Government Agencies

2.Non-Governmental Organizations (NGOs)

3.Academic Institutions

System Overview:

The water system management project aims to integrate IoT technologies for efficient monitoring and management of water resources. Through the use of sensors, the system collects data on water levels, turbidity, pH levels, and flow rates. This data is processed and analyzed to ensure the quality of water for various purposes, including household usage and drinking water supply. The system provides real-time insights for prompt decision-making, enabling effective water resource management and conservation efforts.

SENSOR SELECTION AND JUSTIFICATION:

Ultrasonic Sensor:

Used to measure the distance of the water level in the tank, ensuring timely monitoring and preventing overflow.

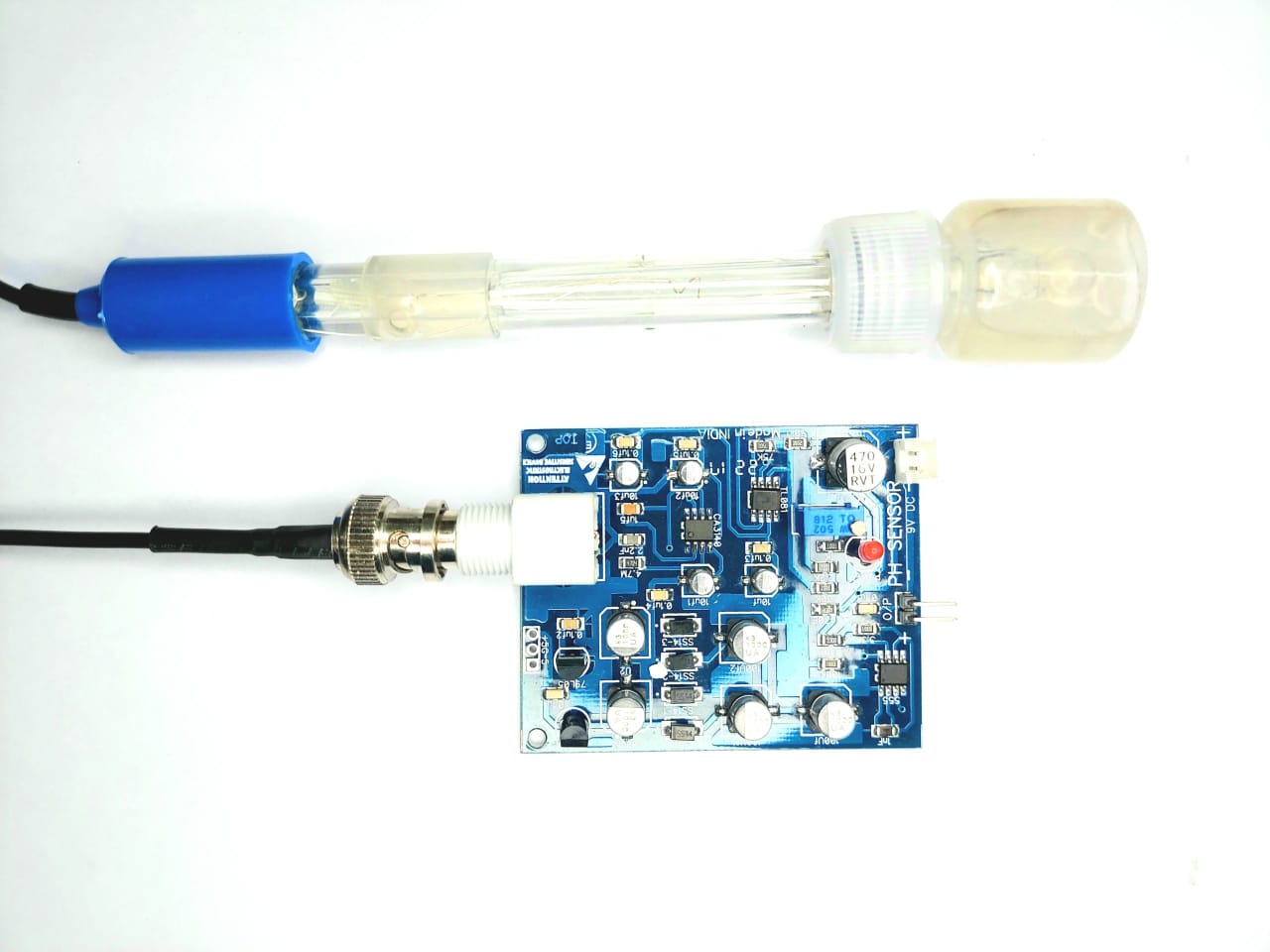


Turbidity Sensor:

Monitors the water's clarity, detecting any impurities or contaminants that might compromise its usability. This is crucial for ensuring water quality.



pH Sensor: Measures the acidity or alkalinity of the water, enabling the detection of any deviations from the acceptable pH range, which could indicate water contamination



Water Flow Sensor:

Tracks the rate of water flow, assisting in monitoring water consumption and flow patterns, aiding efficient usage management

BLOCK DIAGRAM:

USER INTERFACE

USER CONTROL

MICRO CONTROLLER

ALERTS

SENSORS

SOLE

WATER TANK

SOLENOID VALVE

CONTROL LOGIC

BLOCK DIAGRAM DESCRIPTION:

USER CONTROL:

This component represents the user's interface for controlling and monitoring the smart water management system. Users can interact with the system, set preferences, and view data.

USER INTERFACE:

The User Interface component provides a visual interface for users to interact with the system. It displays system status, sensor data, and allows users to control various parameters.

MICROCONTROLLER:

The Microcontroller, such as an Arduino, serves as the central processing unit of the system. It processes data from sensors, implements control logic, and communicates with the User Control and Alerts components.

ALERTS:

Alerts are used to inform users of the system's status. This can include visual or audible alerts to indicate events like water level, system malfunctions, or any other critical information.

SENSORS:

Sensors include various types, such as Ultrasonic, Flow, Turbidity, and pH sensors. They collect essential data about water levels, flow rates, water clarity (turbidity), and pH levels in the system.

CONTROL LOGIC:

The Control Logic component processes the data from the sensors and makes decisions based on predefined rules and thresholds. It determines when to activate the Solenoid Valve to control water flow and may make adjustments based on the sensor data.

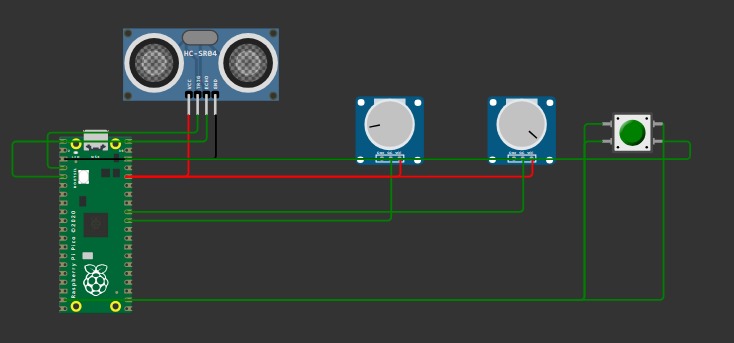
SOLENOID VALVE:

The Solenoid Valve is responsible for controlling the flow of water. It can open or close to regulate the flow of water in response to control signals from the Control Logic.

WATER TANK:

The Water Tank serves as a reservoir for storing and distributing water. It receives and stores rainwater or filtered water, and the Solenoid Valve controls the flow into and out of the tank.

CIRCUIT DIAGRAM:



ALGORITHM:

Step 1: Initialization

Initialize sensors, actuators, and system parameters.

Set control thresholds and calibration.

Step 2: Data Acquisition

Continuously collect sensor data:

Water level (Ultrasonic sensor)

Flow rate (Flow sensor)

Turbidity (Turbidity sensor)

pH level (pH sensor)

Step 3: Control Logic

Check and react to sensor data:

If water level is too high, close the solenoid valve.

If turbidity is high, initiate filtration.

If pH is out of range, adjust it.

If flow rate is excessive, stop water distribution.

Step 4: User Interface and Alerts

Provide a user interface for monitoring.

Generate alerts for abnormal conditions.

Allow user control and adjustments.

Step 5: Loop

Repeat steps 2 to 4 in a continuous loop.

Step 6: End

Implement a system shutdown or exit condition.

DATA COLLECTION PLAN:

Ultrasonic Sensor: Collect distance data every second to monitor water levels and prevent overflow.

Turbidity Sensor: Gather turbidity level readings every minute to ensure consistent water quality.

pH Sensor: Record pH level data every 5 minutes to detect changes in water composition.

Water Flow Sensor: Track water flow rate every 10 seconds for efficient usage management and leak detection.

CODING(MICROPYTHON) USING RASPBERRY PI PICO:

from machine import Pin, ADC

import utime

trigger = Pin(2, Pin.OUT)

echo = Pin(3, Pin.IN)

distance = 0

potentiometer\_pin = 26

turbidity\_adc = ADC(Pin(potentiometer\_pin))

ph\_potentiometer\_pin = 27

ph\_adc = ADC(Pin(ph\_potentiometer\_pin))

water\_flow\_pin = 14

water\_flow\_sensor = Pin(water\_flow\_pin, Pin.IN, Pin.PULL\_UP)

water\_flow\_count = 0

def perform\_filtering\_process():

print("Performing filtering process...")

def ultra():

global distance

trigger.low()

utime.sleep\_us(2)

trigger.high()

utime.sleep\_us(5)

trigger.low()

while echo.value() == 0:

signaloff = utime.ticks\_us()

while echo.value() == 1:

signalon = utime.ticks\_us()

timepassed = signalon - signaloff

distance = (timepassed \* 0.0343) / 2

while True:

ultra()

utime.sleep(1)

if 70 < distance < 90:

print("Water tank is going to be full")

if distance < 100 and not (70 < distance < 90):

print("Water tank is full")

break

print("the distance of water is:",distance)

turbidity\_value = turbidity\_adc.read\_u16()

print("Turbidity Level:", turbidity\_value)

if turbidity\_value > 50000:

print("Water is dirty")

while turbidity\_value > 50000:

perform\_filtering\_process()

turbidity\_value = turbidity\_adc.read\_u16()

print("Turbidity Level:", turbidity\_value)

utime.sleep(1)

print("Water is now pure")

else:

print("Water is pure")

simulated\_ph\_value = ph\_adc.read\_u16()

print("Simulated pH Level:", simulated\_ph\_value)

if simulated\_ph\_value > 30000:

print("Water pH is outside the acceptable range")

else:

print("Water pH is within the acceptable range")

if not water\_flow\_sensor.value():

water\_flow\_count += 1

print("Water flow count:", water\_flow\_count)

if simulated\_ph\_value > 30000:

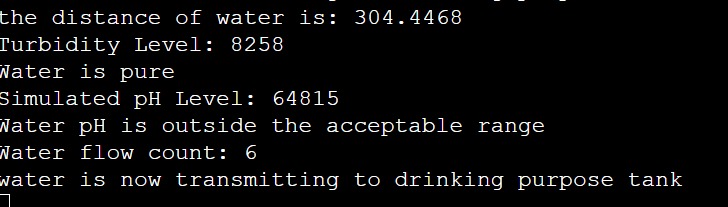
print("water is now transmitting to drinking purpose tank")

else:

print("water is now transmitting to domestic purpose tank")

utime.sleep(1)

RESULT:



WEB INTERFACE INTEGRATION:

DESCRIPTION:

1.Displays simulated sensor data in a web browser using JavaScript.

2.Simulates the behavior of various sensor readings, such as distance, turbidity, pH, and water flow.

3.Updates the displayed data in real-time using the setInterval function to mimic sensor data changes

WEB INTERFACE CODING:

<!DOCTYPE html>

<html>

<head>

<title>Data Display</title>

<script>

function displayData() {

// Simulated variables

let distance = 80;

let turbidity\_value = 60000; // Simulated turbidity value

let ph\_value = 20000; // Simulated pH value

let water\_flow\_count = 0;

let water\_flow\_pressed = false;

// Simulate function

function ultra() {

// Simulate the ultrasonic sensor

}

function perform\_filtering\_process() {

// Simulated filtering process

console.log("Performing filtering process...");

}

// Simulate data reading loop

setInterval(function () {

ultra();

if (70 < distance && distance < 90) {

console.log("Water tank is going to be full");

}

if (distance < 100 && !(70 < distance && distance < 90)) {

console.log("Water tank is full");

}

if (turbidity\_value > 50000) {

console.log("Water is dirty");

while (turbidity\_value > 50000) {

perform\_filtering\_process();

turbidity\_value -= 1000; // Simulate filtering process

console.log("Turbidity Level:", turbidity\_value);

}

console.log("Water is now pure");

} else {

console.log("Water is pure");

if (ph\_value > 30000) {

console.log("Water pH is outside the acceptable range");

perform\_filtering\_process(); // Simulate filtering process for pH adjustment

} else {

console.log("Water pH is within the acceptable range");

}

if (!water\_flow\_pressed) {

water\_flow\_count++;

water\_flow\_pressed = true;

console.log("Water flow count:", water\_flow\_count);

} else {

water\_flow\_pressed = false;

}

}

}, 1000);

// Update the HTML element with the fetched data

document.getElementById('data-container').innerHTML = 'Data displayed here';

}

</script>

</head>

<body onload="displayData()">

<h1>Data Display</h1>

<div id="data-container"></div>

</body>

</html>

CODE TO FETCH DATA FROM THE WOWKI SIMULATION:

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Data Display</title>**

**<script>**

**function displayData() {**

**function fetchData() {**

**fetch('/data')**

**.then(response => response.json())**

**.then(data => {**

**document.getElementById('data-container').innerHTML = JSON.stringify(data);**

**})**

**.catch(error => {**

**console.error('Error fetching data:', error);**

**});**

**}**

**// Fetch data every 3 seconds**

**setInterval(fetchData, 3000);**

**}**

**</script>**

**</head>**

**<body onload="displayData()">**

**<h1>Data Display</h1>**

**<div id="data-container"></div>**

**</body>**

**</html>**

**Fetch Data from Wokwi Simulation:**

**1.Retrieves data from a Wokwi simulation using the fetch API in JavaScript.**

**2.Periodically fetches data every 3 seconds to ensure up-to-date information is displayed in the browser.**

**3.Handles any errors that might occur during the data retrieval process to maintain smooth operation.**

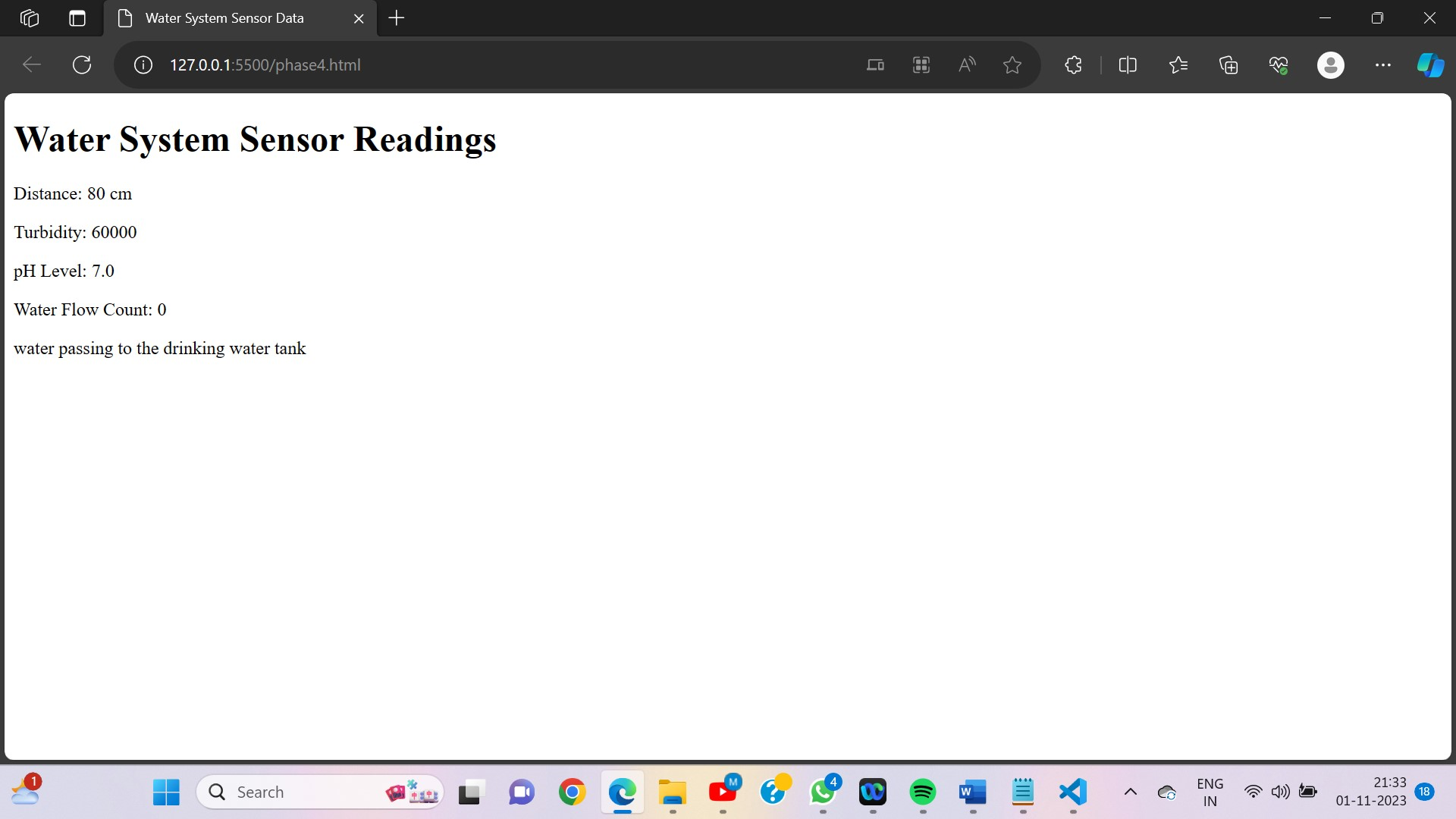
SERVER SETUP:

**Sets up a basic HTTP server in Python using the http.server and socketserver modules.**

**Serves the specified HTML files at the designated port to allow access through a web browser.**

**Continuously runs the server to ensure the HTML files are accessible to clients and updates are reflected in real-time.**

**OUTPUT FROM THE BROWSER:**



**USER INTERACTION:**

**Web Interface: Provide real-time data display of water system sensor readings on a user-friendly web page.**

**Notifications: Alert users about water conditions, such as dirty water or pH imbalances, to ensure prompt action.**

**Manual Override: Allow manual control over water flow and purification processes through the web interface for user convenience.**

APPLICATIONS:

1)Residential water savings through rainwater reuse.

2)Agricultural irrigation efficiency.

3)Industrial water resource sustainability.

4)Environmental conservation and research.

5)Smart city water infrastructure.

conclusion:

The water system management project successfully deploys advanced sensor technologies for efficient water resource monitoring and control, supporting informed decision-making for sustainable water management.