SMART WATER MANAGEMENT

. 24 SYSTEM

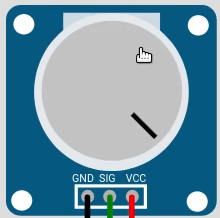
PHASE:III

. Name : MOHAMED AHIL A

Nm.no: au813821106069

Mail.id:mohamedahil.a2004@gmail.com

POTENTIOMETER:



POTENTIOMETER AS A TURBIDITY SENSOR

I)Mimic Turbidity Level: Turn the potentiometer knob to generate different voltage values, simulating varying turbidity levels.

II)Adjust Thresholds: Calibrate the potentiometer to correspond with the turbidity levels you want to represent. Ensure that the threshold values in the code match the desired turbidity ranges.

III)Analog-to-Digital Conversion: The potentiometer's analog output is converted to a digital value using the ADC, which determines the turbidity level in the script.

By adjusting the potentiometer, you can replicate the behavior of a turbidity sensor in your simulation

POTENTIOMETER AS A Ph SENSOR

I)Emulate pH Levels: Adjust the potentiometer's resistance to simulate various pH levels, mimicking the changes in pH with the knob's rotation.

II)Calibration: Map the potentiometer's resistance to the specific pH values you want to represent, ensuring that the analog signals closely match the pH levels in your application.

III)Analog-to-Digital Conversion: Convert the potentiometer's analog output to digital values using an ADC, enabling the microcontroller to interpret and respond to the simulated pH levels within the system.

manipulate the potentiometer and interpret its analog signals, you can replicate the basic behavior of a pH sensor

.

PUSH BUTTON:

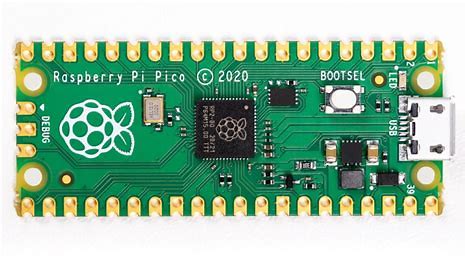


I)Functional Imitation: Configure the push button to mimic the operation of a water flow sensor, where pressing the button simulates the detection of water flow, and releasing the button indicates the cessation of water flow.

II)Counting Mechanism: Use a counter variable to keep track of the number of times the button is pressed, thereby creating a virtual count of the occurrences of water flow.

III)Interface with Microcontroller: Connect one end of the push button to the GPIO pin and the other end to the ground, enabling the microcontroller to monitor changes in the button's state, effectively interpreting them as events of water flow.

RASPBERRY PI PICO:



Raspberry Pi is a small-sized, affordable, single-board computer that offers versatile functionality. With its GPIO pins, it allows for easy interaction with external hardware, making it popular for various projects. It's known for its accessibility, flexibility, and capability to support diverse applications, ranging from learning to program to building advanced DIY projects and even serving as a basic desktop computer.

BLOCK DIAGRAM:

USER INTERFACE

USER CONTROL

MICRO CONTROLLER

ALERTS

SENSORS

SOLE

WATER TANK

DRINKING PURPOSE

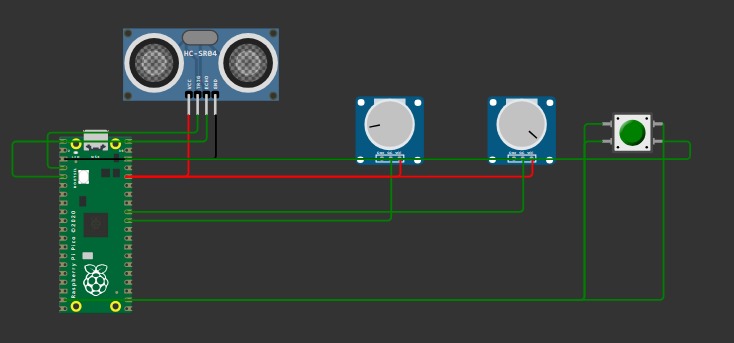
WATER TANK

HOUSEHOLD PURPOSE

SOLENOID VALVE

CONTROL LOGIC

CIRCUIT DIAGRAM:



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)

OUTPUT:

