IOT WATER CONSUMPTION MONITORING SYSTEM :

To Start building the IOT water consumption monitoring system :

DEFINE REQUIREMENTS :

Determine what data you want to collect

(e.g., water usage, flow rate, temperature).

SELECT HARDWARE :

Choose IoT hardware components such as sensors, microcontrollers, and communication modules

(e.g., Arduino, Raspberry Pi, or specialized IoT boards).

Select appropriate water flow sensors and any other necessary sensors for temperature or pressure.

CONNECT SENSORS :

Connect the chosen sensors to your IoT hardware.

DATA TRANSMISSION :

Establish a communication protocol (e.g., Wi-Fi, Bluetooth, LoRa, or cellular) to transmit data from the sensors to a central system.

Ensure a reliable internet connection is available, especially for remote monitoring.

CENTRAL HUB :

Set up a central hub (e.g., a Raspberry Pi or cloud server) to receive and process data from the sensors.

Develop software to handle data reception and storage.

DATA STORAGE :

Choose a database system to store the collected data

(e.g., MySQL, MongoDB, or cloud-based solutions like AWS IoT Core or Azure IoT Hub).

DATA ANALYSIS :

Implement data analysis algorithms to interpret the sensor data, calculate water consumption, and detect anomalies.

USER INTERFACE :

Create a web or mobile interface to display water consumption data to users.

Include features like real-time monitoring, historical data visualization, and alerts for unusual consumption patterns.

SECURITY :

Implement security measures to protect the system from unauthorized access and data breaches.

POWER MANAGEMENT :

Consider power requirements and implement efficient power management for IoT devices, especially if they run on batteries.

TESTING CALIBRATION :

Test the system to ensure accurate data collection and analysis.

Calibrate the sensors as necessary for accurate measurements.

DEPLOYMENT :

Install the IoT sensors and central hub in the target location

(e.g., homes, factories, or agricultural sites).

MAINTENANCE AND UPDATES :

Regularly maintain and update the system to ensure its continued functionality and security.

DATA STORAGE AND ANALYSIS :

Continuously collect, store, and analyze data to monitor water consumption trends and make informed decisions.

USER EDUCATION AND ENGAGEMENT :

Educate users on how to interpret and utilize the data for water conservation and efficiency.

This is a complex project, and you may need to adapt these steps based on your specific requirements and resources. It's important to consider scalability, data privacy, and regulations related to water monitoring in your region as well.

CONFIGURATION OF IOT SENSORS LIKE FLOW METERS TO MEASURE WATER CONSUMPTION IN PUBLIC PLACES :

To configure IoT sensors like flow meters to measure water consumption in public places, you'll need to follow these steps:

SELECT THE RIGHT SENSORS :

Choose flow meters and IoT sensors suitable for measuring water consumption. Ensure they are durable and designed for outdoor use.

CONNECTIVITY :

Set up a network for the sensors. You can use Wi-Fi,

Lora WAN, or cellular connections, depending on the location and coverage.

SENSOR PLACEMENT :

Install the sensors at strategic points in public places like parks, plazas, or water fountains to capture accurate data.

POWER SUPPLY :

Ensure a stable power supply for the sensors. This might include battery packs, solar panels, or nearby electrical outlets.

DATA COLLECTION :

Configure the sensors to collect data at regular intervals. The frequency can vary depending on the need for real-time monitoring.

DATA TRANSMISSION :

Set up a system for transmitting the data from the sensors to a central server or cloud platform. This can be done using APIs or IoT protocols.

DATA STORAGE :

Store the collected data securely. Consider using cloud storage for scalability and accessibility.

DATA ANALYSIS :

Implement data analysis tools to interpret the information. This can include identifying trends, anomalies, and potential water wastage.

VISUALIZATION :

Create a dashboard or user interface to display water consumption data in an understandable format. This can be web-based or mobile app-based.

ALERTS AND NOTIFICATIONS :

Configure the system to send alerts or notifications in case of unusual water consumption, leaks, or other issues.

MAINTENANCE AND CALIBRATION :

Regularly maintain and calibrate the sensors to ensure accuracy and reliability.

COMPLIANCE AND PRIVACY :

Ensure that your project complies with data privacy regulations and obtain any necessary permissions.

SCALABILITY :

Plan for scalability if you intend to expand the system to more public places.

USER ACCESS :

Define who can access the data and how it will be used. Make the data available to relevant authorities and stakeholders.

FEEDBACK LOOP :

Implement mechanisms for taking action based on the data collected, such as water conservation initiatives.

Remember to work with a team experienced in IoT and sensor deployment to ensure a successful implementation for measuring water consumption in public places.

DEVELOPING A PYTHON SCRIPT ON THE IOT SENSORS TO SEND REAL-TIME WATER CONSUMPTION DATA TO THE DATA - SHARING PLATFORM :

The key components of this project are as follows:-

1. **Web Application**.  The web application is created using PHP, HTML & JavaScript as frontend and MySQL as backend.

2. **Webserver**. To run the web application, you will need a webserver. You can host the web application locally on a webserver on your LAN or you can host it on a webserver provided by any hosting provider accessible through a public IP address.

For Local host. You can use widely popular XAMPP to install a webserver  on a PC. The package installs Apache Webserver, PHP and MySQL database server on your PC.

For Remote host. You need to buy a web hosting plan from any of the hosting provider like GoDaddy, Speed host etc.

3. **Raspberry Pi**.  Raspberry Pi has a built in WIFI which makes it easier to connect to internet through home WIFI router. Also, it has GPIO pins for external hardware / sensor interface. So basically, we can program it to measure a sensor’s data and send it to a remote / local server through WIFI ethernet network. It is not necessary to use a Raspberry Pi for this project, you can use other modern microcontrollers such as ESP32 which are much cheaper and have built in WIFI. I chose Raspberry Pi as it supports Python and has good community support.

4. **Ultrasonic Sensor**. HCSR-04 ultrasonic sensor is used for measuring distance. It works on the basic principle of sound wave reflection from an object and time taken by the echo to calculate the distance. The sound wave emitted by this sensor are reflected back by water surface just as good as they would reflect back from a wall. So, the idea is to mount this sensor on top of the water tank to get the sound wave reflection from the surface of water. This sensor is controlled by a Python script running on the Raspberry Pi.

Installing the code in your webserver

Sample.py

#Project: Smart Water Tank

#Created by: Jitesh Saini

import random, os

def measure\_distance():

d=random.randrange(5, 100)

return d

def sendData\_to\_remoteServer(dist):

url\_remote="http://192.168.1.2/water-tank/insert\_data.php?dist=" + str(dist)

cmd="curl -s " + url\_remote

result=os.popen(cmd).read()

print (cmd)

def main():

dist = measure\_distance()

sendData\_to\_remoteServer(dist)

if \_\_name\_\_ == '\_\_main\_\_':

main()

sensor.py

**#you can use the setup\_cron.sh bash script to install a cron job to automatically execute this file every minute.**

import RPi.GPIO as GPIO

import time,os

import datetime

TRIG = 6

ECHO = 5

ALARM = 23

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG,GPIO.OUT)

GPIO.setup(ECHO,GPIO.IN)

GPIO.output(TRIG, False)

GPIO.setup(ALARM,GPIO.OUT)

GPIO.output(ALARM, True)

print ("Waiting For Sensor To Settle")

time.sleep(1) #settling time

def get\_distance():

dist\_add = 0

for x in range(20):

try:

GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)

while GPIO.input(ECHO)==0:

pulse\_start = time.time()

while GPIO.input(ECHO)==1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150

distance = round(distance, 3)

print (x, "distance: ", distance)

dist\_add = dist\_add + distance

#print "dist\_add: ", dist\_add

time.sleep(.1) # 100ms interval between readings

except Exception as e:

pass

avg\_dist=dist\_add/20

dist=round(avg\_dist,3)

#print ("dist: ", dist)

return dist

def sendData\_to\_remoteServer(dist):

#replace 192.168.1.2 with the IP address of your webserver

url\_remote="http://192.168.1.2/water-tank/insert\_data.php?dist=" + str(dist)

cmd="curl -s " + url\_remote

result=os.popen(cmd).read()

print (cmd)

def low\_level\_warning(dist):

tank\_height=114 #set your tank height here

level=tank\_height-dist

if(level<40):

print("level low : ", level)

GPIO.output(ALARM, False)

else:

GPIO.output(ALARM, True)

print("level ok")

def main():

distance=get\_distance()

print ("distance: ", distance)

sendData\_to\_remoteServer(distance)

low\_level\_warning(distance)

print ("---------------------")

if \_\_name\_\_ == '\_\_main\_\_':

main()

setup.sh

#!/bin/bash

# specify the file name that you want to run through cron automatically every one minute

FILE\_NAME="sensor.py"

crontab -u $USER -l| grep $PWD/$FILE\_NAME > /dev/null

if [ $? -eq 0 ]

then

echo "$FILE\_NAME is already running"

else

crontab -l > mycron

echo "\* \* \* \* \* sudo python $PWD/$FILE\_NAME &" >> mycron

crontab mycron

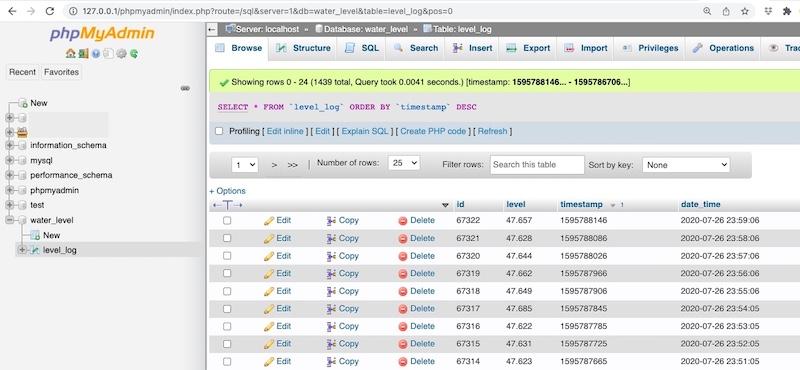
rm mycron

echo "cron task added. Now $PWD/$FILE\_NAME file will run every minute"

fi

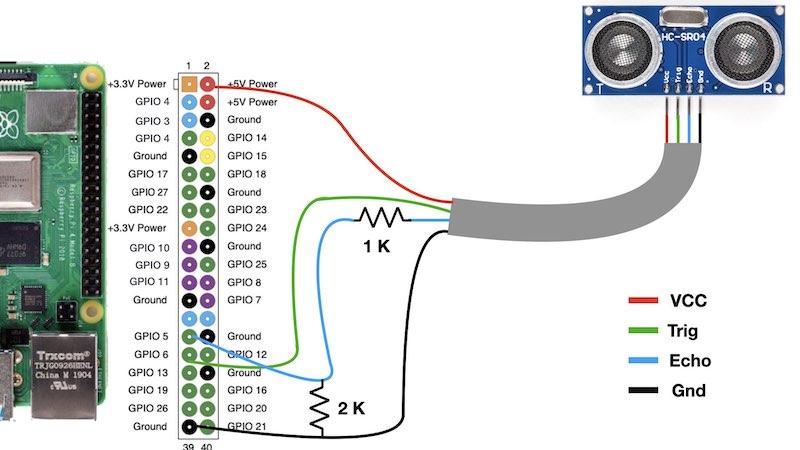
Extract the zip file and place the ‘water-tank’ folder in the public directory of your webserver. It is ‘htdocs’ if you have used the XAMPP. it is also named as ‘www’ if you use different installer. In my macbook this is where the folder is located:-

Inside the ‘water-tank’ folder you should find a file called ‘water\_level.sql’. This contains the sample database.  Using the PhpMyAdmin interface create a database called ‘water\_level’ and import this sql file. Post successful import, you should see the table named ‘level\_log’ is created as shown in picture below.

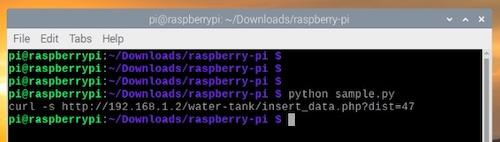


This table is used for storing the readings arriving from the remote sensor. Do not forget to provide the database connection parameters in the beginning of 'util.php' file. Also, enter the physical dimensions of your tank as shown below. They are required for the purpose of calculating volume of the liquid present in the tank.

Now, open a browser and type ‘127.0.0.1/water-tank’ in address bar. Replace ‘127.0.0.1’ with the IP address of the webserver if you are accessing the web application from a some other PC connected to the same LAN.

Raspberry Pi and HCSR-04 Hardware Connection

The 'sample.py' is a minimalistic file that can be used to test communication with the Web Application. This file generates a random value and sends it to the server. Open the terminal and run this file. You will notice that it makes a web request to the server as shown below.

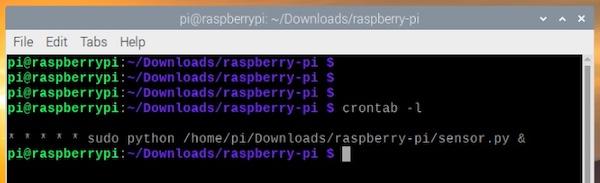


Now, connect the ultrasonic sensor to Raspberry Pi as per connection diagram and run the main file 'sensor.py'. It contains the code to generate distance data through ultrasonic sensor and sends it to the server.

Finally, set up a cron task to run the 'sensor.py' automatically every one minute. You can do it using the 'crontab -e' command and create the entry manually. Otherwise, simply run the 'setup\_cron.sh' bash file. It ensures that the cron task in created correctly.



You can check the cron entry by using 'crontab -l' command as shown below



Thats it, you have successfully implemented this project.

CONCLUSION :

For all those below topics we made a presentation above with suitable code and example. Reference from: https://helloworld.co.in/article/smart-water-tank

1. To built the IoT water consumption monitoring system.

2. To Configure IoT sensors like flow meters to measure water consumption in public places.

3. To Develop a Python script on the IoT sensors to send real-time water consumption data to the data-sharing platform .