IoT based Water Management System using MQTT protocol.

Kundan Gaikwad
Department of Instrumentation and Control Engineering
Govt. College of Engineering and Research,Awasari(K.)
Pune(Mah.), India
kundan.g21099@gmail.com

***Abstract*—Now a days, increasing environmental pollution, world population, water scarcity, etc. are some major issues faced by the society. We need solutions, which would be implemented from a individual person to large industries. Definition of water management is difficult, but as long as the wastage of water is avoided, everyone is contributing to solving water scarcity.**

The purpose of this research is to monitor the usage of water, avoid overflow, and ultimately find the way to save water. The ultrasonic sensor is used to measure the level of the tank, the flow sensor is used to measure the flow rate of water and total consumed volume. Level, flow rate, and volume data will be sent by Node MCU to the mobile app using Message Queuing Telemetry Transport protocol. Also, the solenoid valve which works as a tap from the mobile app can be controlled. Message Queuing Telemetry Transport (MQTT) protocol is a lightweight, publish/subscribe-based protocol and works on the Transmission Control Protocol (TCP) layer which is perfect for constrained devices like Node MCU.

***Index Terms*—MQTT (Message Queuing Telemetry Transport), ESP8266, Node MCU, SQLite Database, QoS (Quality of Service), Mosquitto broker.**

I. Introduction

According to Inernet World Statistics, estimated number of internet users is 326,72,33,742 in world. Almost 42.7% of world population is using internet [11]. Increased use of internet resulting internet of things (IoT) to grow. Internet of Things (IoT) or Machine-to-Machine (M2M) communications concept is expected to be one of the future networking solutions. The number of IoT devices are forecasted to grow rapidly, up to 50 billion devices in the year 2020. Gartner says that 6.4 Billion connected objects will be in use in 2016, 30% more than in 2015. That number will grow up to 20 billion by 2020. IoT represents the next evolution of the Internet and extends its ability to gather, analyze, and distribute data that can be turned into information and knowledge. Furthermore, IoT has made internet sensory (temperature, pressure, vibration, light, moisture, stress) thereby achieving improved monitoring, analyzing, and tracking systems [1]. For the devices to be able to communicate, a standard protocol is required. As of now, there are several existing candidates to become the standard protocol for IoT. Among them, widely adopted protocols in the loT and M2M [2] fields are MQTT, CoAP [3], and LWM2M [4]. However, in recent years, one protocol has been gaining traction in the community. A discussion has been initiated to analyze Message Queuing Telemetry Transport (MQTT) protocol [5]. Above protocols i.e. MQTT, CoAP, LWM2M works on TCP/IP protocol suite, this is the most useful set of protocols when establishing IoT networks.

The method implemented in this paper helps the user to know-how much water has been used on daily basis to analyze and to limit unwanted usage. This work implements IoT based water management solution using the MQTT protocol. Sensors are used to measure data like level, volume consumed, and flow rate of water. The publishing client is Node MCU which publishes these data on specific MQTT topics to the mosquito broker. A mobile app is subscribing client which subscribes to the same topic and receives messages sent by the Node MCU from the mosquitto broker. In the case of controlling valve mobile app work as a publisher client and publishes messages to Node MCU which acts as subscribing client. Received messages have been stored on the SQLite database and displayed in the form of a graph on the mobile app. Storing data helps the user to analyze how much water has been consumed and to implement methods to save the water.

II. Messsage Queuing Telemetry Transport

This paper will present the MQTT protocol as a representative M2M protocol. MQTT was originally created to gather data from multiple devices and then transport it to the single IT infrastructure. It works on the TCP/IP protocol layer, created as the lightweight publish/subscribe based protocol for small code footprints and designed to work on low bandwidth and low power with a message delivery guarantee [6]. MQTT architecture has shown in fig.(1). Where clients send and receive messages. Messages received by the broker from publisher. However, categorizing the messages and sending to the right subscriber is done by broker that is MQTT server. Discrimination of messages is done on the basis of MQTT topic and client ID. Each client has own unique ID.

A. MQTT Client

MQTT clients are of two types, the one who produce and send data (Publisher client) and one who receives the data (Subscriber client). In both cases, the client must connect to the MQTT broker. After establishing the successful connection with the broker, client must declare whether he is a publisher

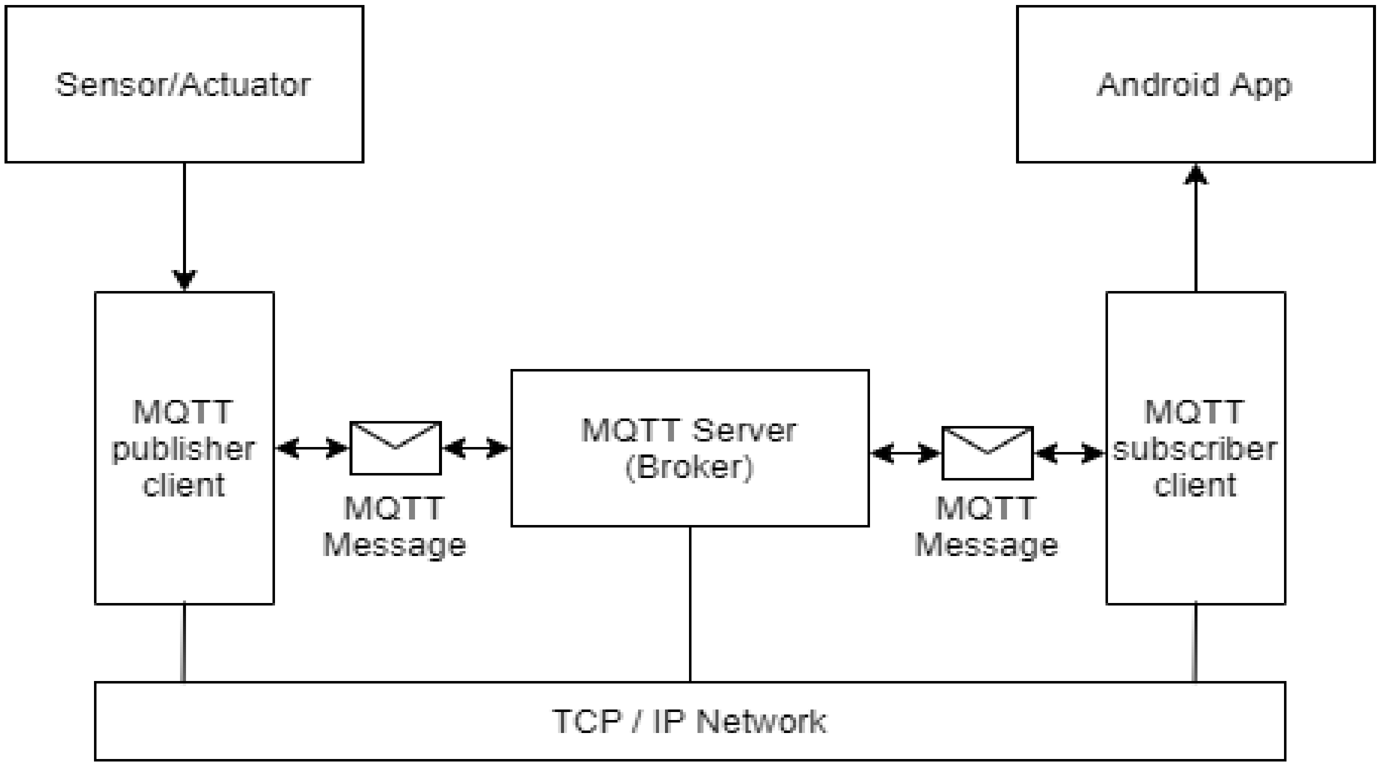


Fig. 1. Message Queuing Telemetry Transport

or subscriber. To send and receive the data, topic string has been used. A publisher sends messages to the particular topic string to the broker, and to receive those messages subscribers must subscribe to the same topic. (Ex of topic- ”esp/temperature” , ”esp123/humidity”, etc.)

B. MQTT broker

MQTT broker could be self-built or hosted by a third party or running on the cloud. The publisher does not carry any addresses or identity of the subscriber as well as the subscriber is unaware of the publisher’s information. When publisher sends messages to broker on a certain topic, broker check whether any client has subscribed to the same topic or not, if yes, then broker sends all messages to subscriber client. This architecture of the MQTT protocol helps to easily scale from a single device to a thousand. (Ex. of broker - Mosquitto, HiveMQ, Solace, MQTT cloud, etc.)

C. MQTT connection

- Step 1 - Client connects to the broker by sending connect packet. The information included in connect packets are as follows-
  - ClientID - A system generated or manually given a unique ID to identify the client.
  - Clean session flag - If a clean session has been set to false, then the broker does not store the subscription. If a clean session is true, then broker stores the all subscriptions with QoS 1 and 2.

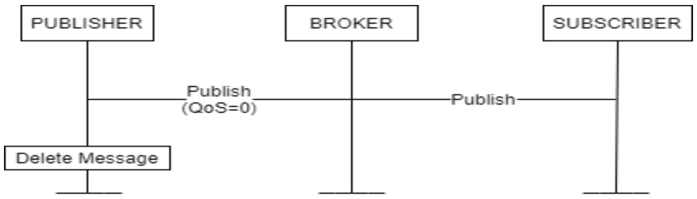


Fig. 2. QoS=0

  - Quality of Service (QoS)- There are three levels of QoS.
    - ∗ Fig(2), at QoS=0, the publisher sends messages at most once but has not been informed about whether the message is received at its destination or not. There is a possibility of losing the message in this process. That’s why this level is also known as ”fire and forget”.
    - ∗ At QoS=1, in fig.(3) the publisher sends the message at least once with a packet identifier to the broker. The broker sends a message to the subscriber and at the same time reply back with the Publish Acknowledgement (PUBACK) packet to the publisher. Publisher stores and sends the same message again if the PUBACK packet has not been received from broker.
    - ∗ At QoS=2, in fig.(4), this level is also known as ”Assured delivery”. It guarantees that each message will be delivered to the intended recip-

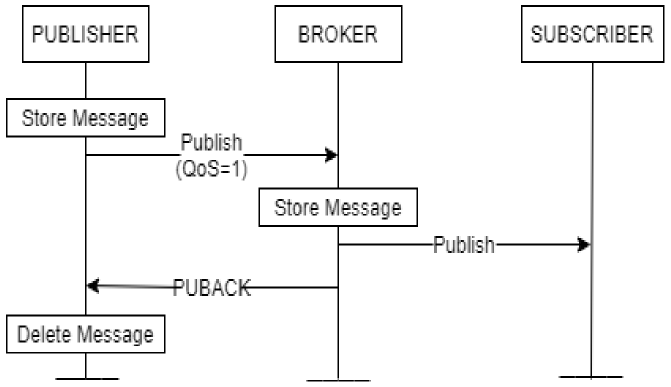


Fig. 3. QoS=1

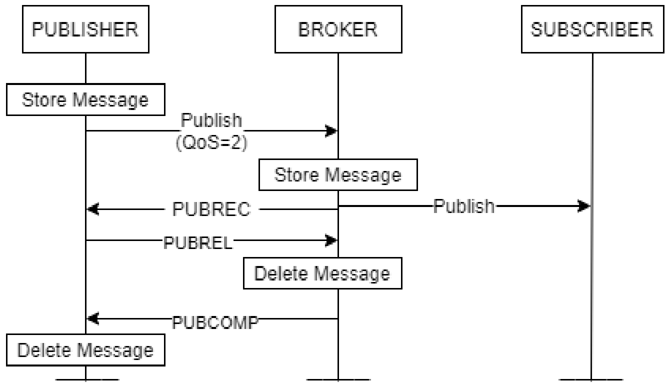


Fig. 4. QoS=2

ient. Broker sends Publish Received (PUBREC) message to publisher client to inform that message has been received. Then after acquiring PUBREC message from broker publisher client send another message i.e. Publish Released (PUBREL), indicating that MQTT message has been released. The moment when broker receives PUBREL message it replies with the Publish Complete (PUBCOMP) message to the publisher client to tell message has been received by the subscriber client.

- Step 2- Broker sends Connection Acknowledgement (CONNACK) packet to publisher client. This packet contains connect acknowledge flag, this flag returns a code that tells whether the connection was successfully established or not.
  List of return code-
  - 0 - Connection accepted
  - 1 - Connection refused, unacceptable protocol version
  - 2 - Connection refused, identifier rejected
  - 3 - Connection refused, server unavailable
  - 4 - Connection refused, bad user name or password
  - 5 - Connection refused, not authorized

III. Objectives of Research

IoT is being used in almost every area : education, agriculture, healthcare, military, tourism, supply chain management, industry and much more [12]. In most of the IoT based system sensors, actuators and especially constrained devices are playing major role. Constrained devices works on less bandwidth, low power, low memory. While considering these limitation of constrained devices, this research must meet following objectives

- To find energy efficient solution.
- To make system less complicated and affordable to everyone.
- Monitoring the system from anywhere.
- To deal with internet connectivity issues in isolated area.
- To find best method of implementation with constrained devices while considering above aspects

IV. Proposed System

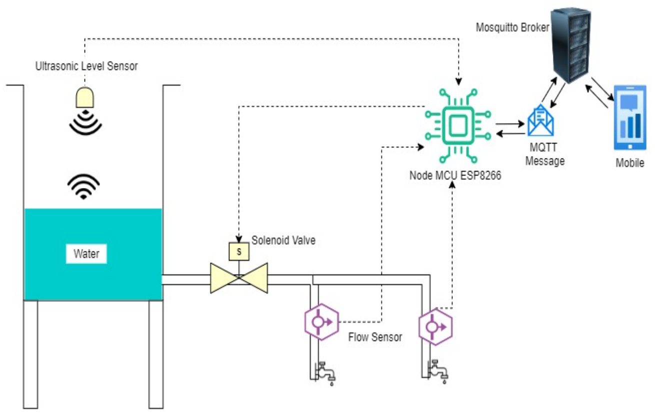


Fig. 5. Proposed System

Fig(6) shows algorithm of node MCU and android app. In case of node MCU, first it get connected to the internet through WiFi then according to program it calculates volume, flow rate, level, and immediately sends messages to the mosquitto broker on the topic 'esp/volume', 'esp/flow', 'esp/level' respectively. Simultaneously it subscribes to the topic 'esp/valve' to receive messages to control valves. If node MCU failed to connect to the mosquitto broker it returns a code. On the other side, in android app, first it will connects to the MQTT mosquitto broker then subscribes to the same topic used in node MCU. So that app will receive messages sent by node MCU from broker. Incoming messages from node MCU has been stored on SQLite database and displayed in a graph instantaneously. At the same time when valve button is clicked by the user, app sends 'ON' and 'OFF' messages to the broker on same topic as node MCU.

To elaborate the concept behind water management system, two flow meter, and two solenoid valve has been used in this system, as shown in fig(7). Consider the jar of 20 L capacity as

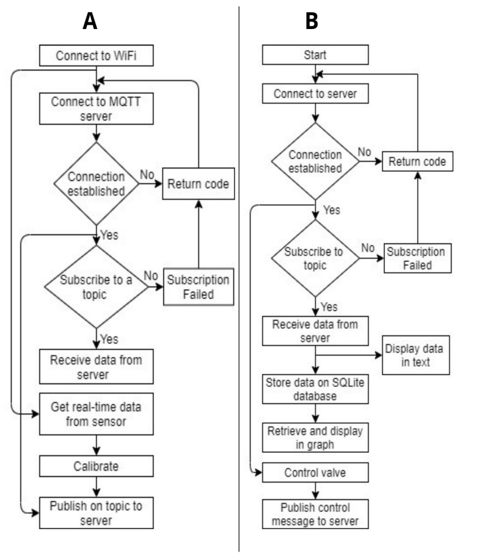


Fig. 6. A- Node MCU and B- Android app algorithm

a tank. Ultrasonic sensor has placed at the top of jar. Schematic diagram of proposed system is showed in fig(5).

- Flow rate and Volume calculation - Theoretically flow rate has been calculated by the following method- If the cross-section area of a pipeline is constant then the average velocity of the water will be our flow rate.

$$Q = V * A \tag{1}$$

  - Q is flow rate/total flow of water through pipe (L/min).
  - V is average velocity.
  - A is cross sectional area of pipe.

  To calculate using flow sensor following methods has been used.

$$PulseFrequency = 7.5 * Q \tag{2}$$

$$VolumeinLitre = \frac{Q * TimeElapsed}{60} \tag{3}$$

$$VolumeinLitre = \frac{[PulseFrequency/7.5] * TimeElapsed}{60} \tag{4}$$

$$VolumeinLitre = \frac{pulses}{7.5 * 60} \tag{5}$$

  - Note= Time Elapsed is in second and Pulses Frequency is in pulses /seconds.

- Level calculation - This is done by using following formula-

$$Distance = \frac{T}{2} * speedofsound \tag{6}$$

  - T= time required to receive reflected ultrasonic waves in seconds.
  - T has divided by 2 because ultrasonic waves has travelled back and forth.
  - speed of sound = 340 m/sec.

  If we consider sound speed as 340 m/sec, that corresponds to 29.412 microseconds /centimeters. To calculate the distance in centimeter-

$$Distance = \frac{(T/2)}{29} \tag{7}$$

  - Consider T is in microseconds in equation (8)

  But above formula calculates distance between water surface and ultrasonic sensor. In order to calculate level of the water, following formula has been used-

$$Level = h - Distance \tag{8}$$

  - h= total height of the water tank .In this system the height of jar is 30 centimetre.
  - Level= water level in centimetre
  - Distance is in centimeter from equation (8)

All of the above calculation has been done in the Node MCU. The calibration factor has been calculated using the trial and error method. Dividing the flow rate by 4.5 and multiplying volume by 3 gives us the exact accurate volume. Node MCU acts as publisher client and sends calculated sensor data on a particular topic by using the PubSub client library in it. However, each calculated value needs a separate topic to publish to the mosquitto public broker. At the same time, the mobile app acts as the subscriber client and subscribes to the same topic to receive the data from the mosquitto broker. Received data will be stored on SQLite database and displayed in the form of a graph in the app. App publishes messages onA, offA, onB, and offB on the respective topic valve, meanwhile, Node MCU acts as a subscriber to receive messages to control valves A and B.

V. Hardware description

A lot of sensors are available in the market but some of them are cheap but ineffective. Aim of this paper is to design a cheap and effective system. The block diagram (i.e. fig 5) shows various blocks. Each block consists of a different module. The components used in the system are Node MCU, HCSR04 ultrasonic sensor, YFS-201 flow sensor, 12V DC solenoid valve, relay.

A. Node MCU ESP8266

Node MCU is a development board and also an open-source IoT platform. This development board is based on ESP-12E. It runs on the ESP8266 WiFi SoC (System On Chip) included firmware from espressif systems. The clock frequency of Node MCU is 80 MHz to 160 MHz adjustable clock frequency. It has 128 RAM and 4 MB flash memory to store data and programs. This board is cheaper than raspberry Pi and other micro-controllers, hence makes the system more affordable. Node MCU can connect to the internet so that we can access the water system from anywhere.

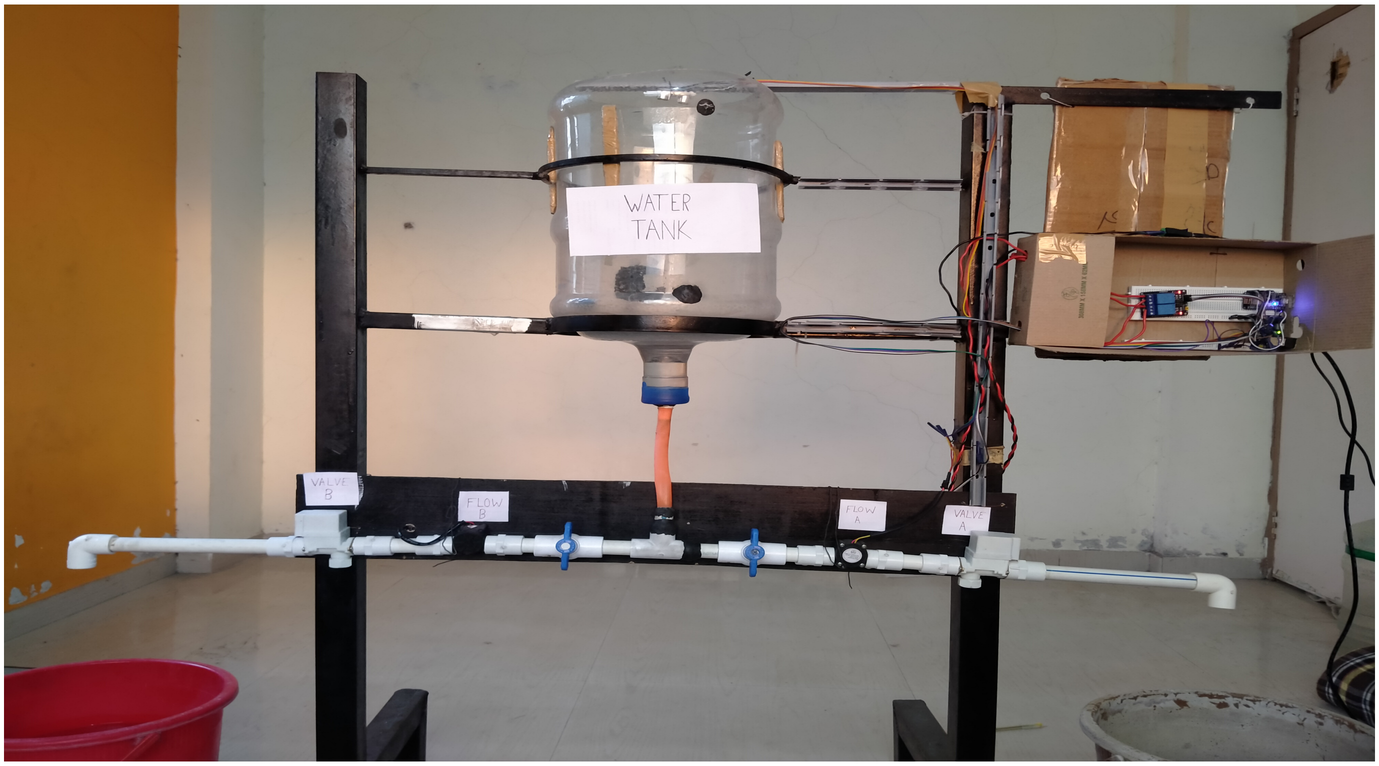


Fig. 7. Water Management System

B. *HCSR04 Ultrasonic Sensor*

HCSR04 Ultrasonic sensor's range is 2 cm to 400 cm. This sensor includes transmitter and receiver to transmit and receive the ultrasonic waves respectively. There are four pins in this sensor

- VCC pin (5V Supply)
- Trigger Pulse Input pin (Transmit the ultrasonic waves when HIGH)
- Echo Pulse Output pin (Receives the reflected ultrasonic waves and sends pulses to the Node MCU)
- GND pin (0V Ground)

As the name suggests, this sensor works on ultrasonic waves. The transmitter transmits the ultrasonic waves which reflect towards the receiver. The time difference between transmitting and receiving the signal will be calculated in the Node MCU through an algorithm. Eventually based on time difference distance will be measured which is nothing but the level of water. The maximum level of water will be the setpoint, stored in Node MCU. If the level exceeds the maximum limit controller will close the valve,

C. *YF-S201 Flow sensor*

This is a Hall effect type sensor. The operating voltage range is 5 V to 18 V with a working flow rate is 1 to 30 liter per minute. A hall effect sensor is used to measure the water flow. When water flows through the flow meter, the inner rotor rolls, its speed changes with different rates of flow. The corresponding pulse signal will be generated by the hall effect sensor as a output. A pulse signal will be sent to the controller. With the help of the algorithm, the received pulse signal will be converted to water flow rate.

D. *Solenoid Valve*

Normally closed, diaphragm type 12 volt DC solenoid valve is used in this system. The operating pressure range is 0.02 MPa to 0.8 MPa.

VI. Software

A. *Arduino IDE*

Arduino IDE (Integrated Development Environment) is a software used to program Arduino compatible boards in C or C++ language. Node MCU has been programmed by using this software

B. *Android Studio*

Android studio has been used to create the android mobile app.

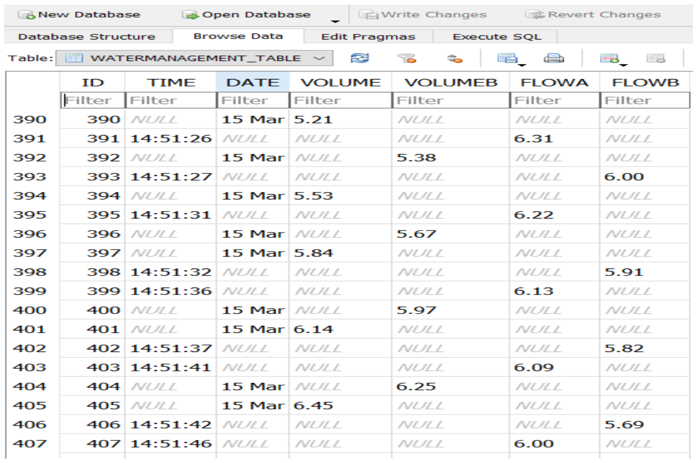
C. *SQLite Database*

D. *Mosquitto MQTT Broker*

VII. Required Libraries

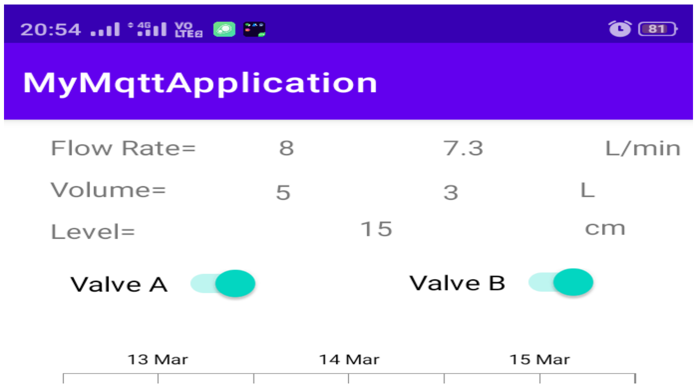
- PubSubClient library - This library allows to send and receive MQTT messages. It supports MQTT 3.1.1 and MQTT 3.1 versions. It is compatibles with Arduino boards, intel Galileo/Edison, ESP8266, ESP32, etc.

- Eclipse Paho Android Service - This library is used to implement MQTT protocol in android app.
- (Note- Use latest version of libraries.)


New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: WATERMANAGEMENT_TABLE

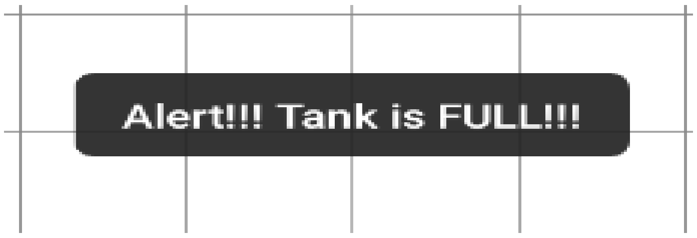
	ID	TIME	DATE	VOLUME	VOLUMEB	FLOWA	FLOWB
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
390	390	NULL	15 Mar	5.21	NULL	NULL	NULL
391	391	14:51:26	NULL	NULL	NULL	6.31	NULL
392	392	NULL	15 Mar	NULL	5.38	NULL	NULL
393	393	14:51:27	NULL	NULL	NULL	NULL	6.00
394	394	NULL	15 Mar	5.53	NULL	NULL	NULL
395	395	14:51:31	NULL	NULL	NULL	6.22	NULL
396	396	NULL	15 Mar	NULL	5.67	NULL	NULL
397	397	NULL	15 Mar	5.84	NULL	NULL	NULL
398	398	14:51:32	NULL	NULL	NULL	NULL	5.91
399	399	14:51:36	NULL	NULL	NULL	6.13	NULL
400	400	NULL	15 Mar	NULL	5.97	NULL	NULL
401	401	NULL	15 Mar	6.14	NULL	NULL	NULL
402	402	14:51:37	NULL	NULL	NULL	NULL	5.82
403	403	14:51:41	NULL	NULL	NULL	6.09	NULL
404	404	NULL	15 Mar	NULL	6.25	NULL	NULL
405	405	NULL	15 Mar	6.45	NULL	NULL	NULL
406	406	14:51:42	NULL	NULL	NULL	NULL	5.69
407	407	14:51:46	NULL	NULL	NULL	6.00	NULL

Fig. 8. SQLite Database



Flow Rate=	8	7.3	L/min
Volume=	5	3	L
Level=		15	cm

Fig. 9. Mobile App Screenshot a



Alert!!! Tank is FULL!!!

Fig. 10. Mobile App Screenshot d

VIII. Results

Fig.(9) is the screenshot of the android app where you can see the real-time data at the top in text format. The parameters flow rate A, volume A, valve A corresponds to the flow meter and valves placed at the right side and flow rate B, volume B, and valve B corresponds to the left side in fig.(7). Fig.(11) shows total volume consumed on 12, 13, and 14 March 2021. Fig.(12) shows flow rates A and B with respect to time. So

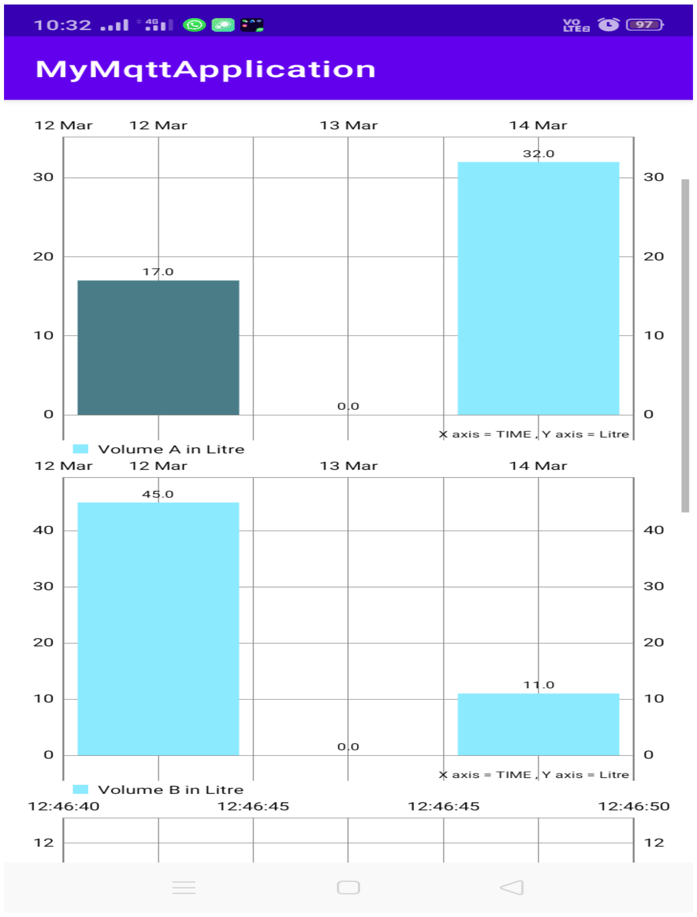


Fig. 11. Mobile App Screenshot b

that users can see at what time the water has been consumed in a day. Storage of data accomplished with the help of SQLite database. Storing data helps the user to understand how much water has been consumed in a week, month, and even in a year. So that he can understand his water consumption and create a solid plan to save water. Fig.(8) shows MQTT messages stored on the SQLite database. Also, the disconnect and delete data button have been made available on the app to disconnect from the mosquitto server and to delete stored data from the database respectively. However, after disconnection, still, the app will be able to receive the missed messages from the publisher client because we are using QoS = 2 levels while subscribing to the MQTT topic. Due to this fact user can see data whenever he wants, there is no need to stay online. If the level of water tank exceeds certain maximum limit, then alert message will be published by the Node MCU on respective MQTT topic. When app receives the alert message it notifies user with the notification shown in fig.(10) with siren.

This system can be implemented from individual person to large societies, industries, etc. A constrained device i.e. Node MCU is at the center of this system. Hence, objectives like less energy consumption, easy to implement, less expensive are already achieved. On the other side MQTT protocol plays an important role to establish reliable communication

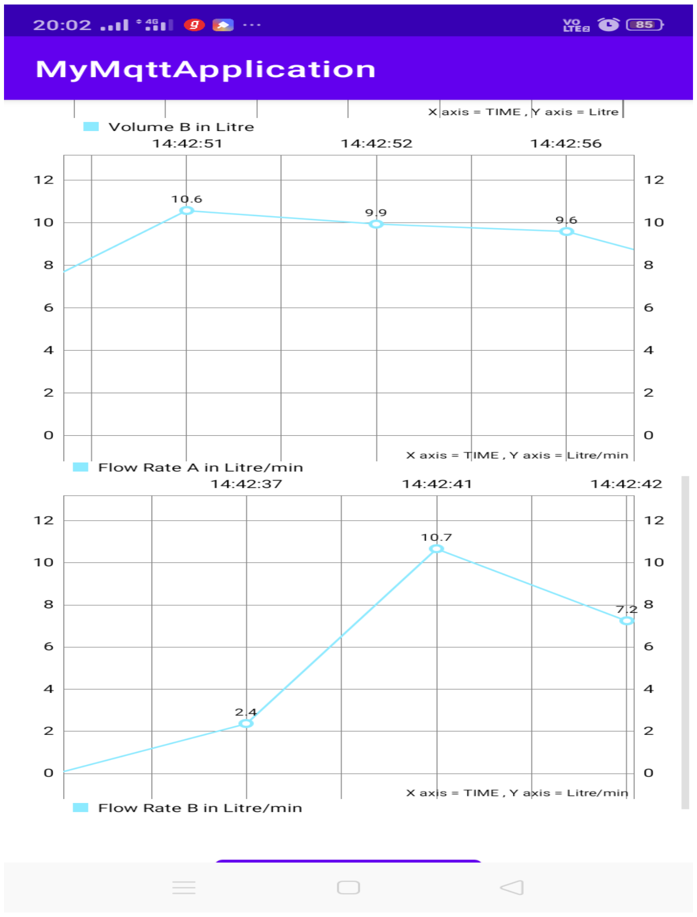


Fig. 12. Mobile App Screenshot c

on constrained devices. Also helps to access system from anywhere and in case of internet failure MQTT provides features like retained message, last will testament. If client disconnects suddenly, broker stores messages and sends when client is online. Hence, all objectives discussed earlier have been met by this research.

IX. Conclusion

The aim of the research was to develop a system to save the water from a smaller scale to a large scale while considering aspects like easy to use, affordability, fast response, and accessibility in isolated areas. So, that everyone can keep a record of the water consumption in order to use water with awareness. To achieve the combination of the given objective a constrained device (i.e. Node MCU) with MQTT protocol implementation had been selected. The Research concludes that the MQTT protocol is the best IoT protocol to meet all of the necessary requirements of this project.

X. Future Scope

Malicious activities on the internet are increasing day by day. As MQTT is designed as a lightweight protocol for low-power applications, it does not provide any strong security option. But there are several types of research that have been conducted to secure MQTT based on SSL/TLS, AES-GCM [7] [8], OAuth-1.0a [9], AES-256 with SHA-256 [8], RSA, etc. But securing MQTT while achieving the objectives of this project is a new challenge. In the case of security implementation in constrained devices payload encryption is one of the good options as constrained devices does not support SSL/TLS layer [10].

XI. Acknowledgement

I want to express my heartfelt thanks to Dr. N. S. Nehe, Dhanashree Raskar, Deepali Bamratkar, Abhishek Kene, and Kohinoor Kotkar for supporting me in this research.

References

[1] D. Evans, ”The Internet of things: how the next evolution of the Internet is changing everything,” Cisco Internet Business Solution Group White Paper, April 2011.
[2] ”Machine to Machine (M2M) Communication Study Report,” IEEE C 80216 100002r7, May, 2010. (accessed on Feb 21, 2015).
[3] Internet Engineering Task Force (IETF), ”The Constrained Application Protocol (CoAP),” Available: http://tools.ietf.orglhtmllrfc7252, 2014.
[4] OMA Lightweight Machine to Machine Protocol vl,O ”http://technical.openmobilealliance.orgiTechnicallreleasej)rogram/ligh tweightM2Mvl0.asp” (accessed on Mar 10,2015).
[5] A. Banks, and R. Gupta. ”MQTT Version 3.1. 1.” OASIS Standard, April 2014.
[6] V. Gazis et al., “A survey of technologies for the internet of things,” in International Wireless Communications and Mobile Computing Conference, August 2015, pp. 1090-1095.
[7] Diego Salas Ugalde. ”Security analysis for MQTT in Internet of Things”, DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE,STOCKHOLM, SWEDEN 2018.
[8] Jameel Ahamed, Md. Zahid, Mohd Omar & Khaleel Ahmad,”AES and MQTT based security system in the internet of things”, Journal of Discrete Mathematical Sciences and Cryptography, 22:8, 1589-1598,2019.
[9] Aimaschana Niruntasukrat, Chavee Issariyapat, Panita Pongpaibool, Koonlachat Meesublak, Pramrudee Aiumsupucgul, Anun Panya, ”Authorization Mechanism for MQTT-based Internet of Things”,IEEE ICC2016-Workshops: W07-Workshop on Convergent Internet of Things.
[10] HiveMQ Enterprise MQTT Broker, [Online]Available: https://www.hivemq.com/mqtt-security-fundamentals/
[11] Al-Rousan, Thamer. (2017). The Future of the Internet of Things. International Journal of Computers, Communications & Control (IJCCC). 4. 10.15242/IJCCIE.AE0417133.
[12] Tirupathi, Megana & Student, Engineering & Professor, Assoc & Krishnamohan, Revu. (2017). Applications of IoT: A Study. 10.13140/RG.2.2.27960.60169.