IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 8, August 2016

'Aquarius'- Smart IOT Technology for Water Level Monitoring System

Prof. A. M. Jagtap[1], Bhaldar Saniya Sikandar[1], Shinde Sharmila Shivaji[1], Khalate Vrushali Pramod[1], Nirmal Kalyani Sarangdhar[1]

Computer Science and Engineering, Rajarambapu Institute of Technology, Islampur, India[1]

Abstract: In mega cities of the world, Water deficiency is one of the crucial problem leads to wastage during transmission. We aim to resolve this issue by using Internet of Things. Monitoring system plays an important role in every household, commercial complex and even industries. The system used ultrasonic sensor to detect the level of water in multiple tanks, switch on or off the pump accordingly and display the status on android device. The water level is monitored and its data is sent through notification to the intended user's android device.

Keywords: Ultrasonic sensor, tanks, remote monitoring, IoT, Raspberry Pi.

I. INTRODUCTION

We need modern methods to protect and preserve as much water. The wastage of water through storage tanks not only waste water but also waste electrical and mechanical energy required to utilize the pumps.
This problem motivated us to design an IoT enabled architecture that can help anyone with an Internet connection to administer water storage level easily without even a click of a button. Looking at the increasing popularity of the Android-based devices and its ability to offer various features we decided to target this platform. Alongside, we have also targeted the generic web platform for those who do not own an Android device.
The web platform is mobile friendly which will make it easy to run on any device available today ranging from old phones to the current generation flagship devices such as iPhones and Windows 10 Phones. This product may reduce cost associated with water usage in large societies or industries.

We get motivation from Modi's Smart City[15] vision which concerns to an urban development vision to integrate multiple Information and Communication Technology (ICT) and Internet of Things (IoT) solutions.

II. BASIC CONCEPTS

Some of the techniques for water level monitoring system are described given below:

A. Water-Level Sensor[4]**:**
The water-level sensors perform a two way communication with the Raspberry Pi where it sends back the raw sensor data when the Raspberry Pi demands for it.
This sensor works on the SONAR principle where it emits an ultrasonic sound wave and receives it back.
When the voltage on the "TRIG" pin is high it emits the sound wave and on receiving it back the voltage on the ECHO pin is set to "HIGH".

B. Raspberry Pi[7]:
The Raspberry Pi houses a Node.js script which performs a two-way communication with the water-level sensor and the MQTT broker respectively.
It receives a request from the Android or Web client via the MQTT broker to get the water-level. On getting the request it sends a "HIGH" signal to the sensor's TRIG pin. The sensor then emits an ultrasonic sound wave which sets the ECHO pin to HIGH after receiving it. This "HIGH" signal is received back by the Raspberry Pi.
The Pi then calculates the distance based on the time elapsed between the sending and receiving of the sound wave. It then communicates with the MQTT broker and sends it the calculated distance.

C. MQTT Broker[9]:
The MQTT broker is a central server which runs on Amazon Web Service's EC2 instance. The broker acts as a bridge that connects all the modules.
It is the centralized MQTT server that hosts topics through with the communication takes place. For the broker all the connected devices are clients (website, Android app, and Raspberry Pi).
The website and/or Android app requests the Raspberry Pi to send the water-level which is received by the broker first. It then locates all the devices subscribed to that topic, finds Raspberry Pi in the list, and forwards it the request.
Similarly, when the Raspberry Pi calculates the level it publishes the level data to a particular topic which is received by the broker first.

The broker then looks for all the devices subscribed to the particular topic, finds website and/or Android app in the list, and forwards the level-data.

D. Node.js server[10]:
The Node.js server is the web-hosting for our website. It is a server for the website but acts as a client for the MQTT

broker. This server is running on a different Amazon Web Service's EC2 instance.
The website is built using Node.js web-technology for two purposes – speed and the ability to communicate with the broker.
The Node.js server constantly demands the Raspberry Pi to send the level, and on receiving the level-data it is mapped to the web user interface.

E. Android MQTT Client[13]:
The Android counterpart acts similar to the web and acts as a client for the broker.
The application uses "Paho MQTT library" as a dedicated service which continuously demands the Raspberry Pi for the water level.

F. Website:
The level-data received from the Raspberry Pi is then mapped to a user-interface element on the website.
This element is a virtual imitation of a reservoir that animates as the water-level in the actual reservoir changes.

G. Android application:
The Android app consists of two parts – the water reservoir animated user interface, and the Android MQTT client.
On receiving the water-level data from the Raspberry Pi, the Android app calculates the level percentage and maps it to the animated user interface.
Additionally, the Android app which runs Android MQTT client as a service is also responsible to push level-change notification to the user if his/her phone is screen-locked.

III. IMPLEMENTATIONS

1) Need to find water level in the tank. So, we use ultrasonic sensor.

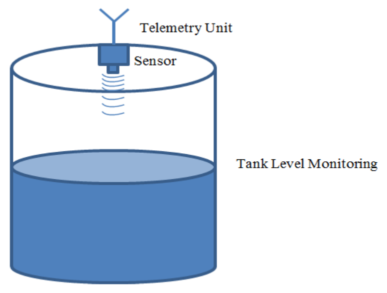

Fig.1 Using Ultrasonic Sensors

2) There are 3 conditions:
- Water at lowest level (20%)
- Water at midlevel (50%)
- Water at highest level (100%)

3) Output of ultrasonic sensor is in cm. We convert it into percentage and store it on a cloud using MQTT technology.

4) This MQTT technology is used via Raspberry Pi. Usually, the language of Raspberry Pi is python. But, we use node.js for better live result.

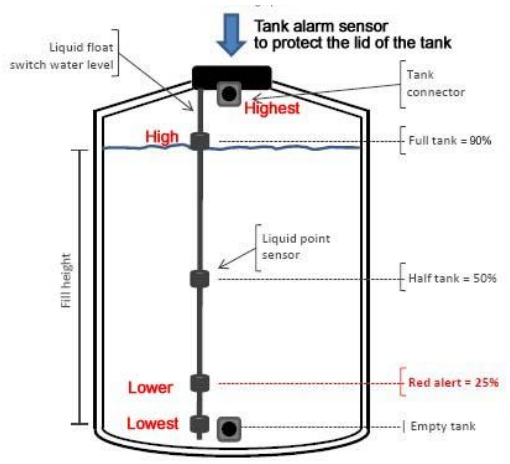

Fig.2 Levels of Tank

5) We retrieve data from cloud and display it on LCD screen using smart phones and websites.
6) We apply this methodology for multiple tanks using different sensor to each tank and using only one Raspberry Pi and only one database on cloud.
7) We use different techniques to analysis cloud data. From this, we get time required to fill tank.

IV. PROPOSED WATER MONITORING NETWORK

Our proposed system guarantees to accumulate a good amount of usable water every day. This monitoring and controlling system uses daily life device like laptop or mobile phone.

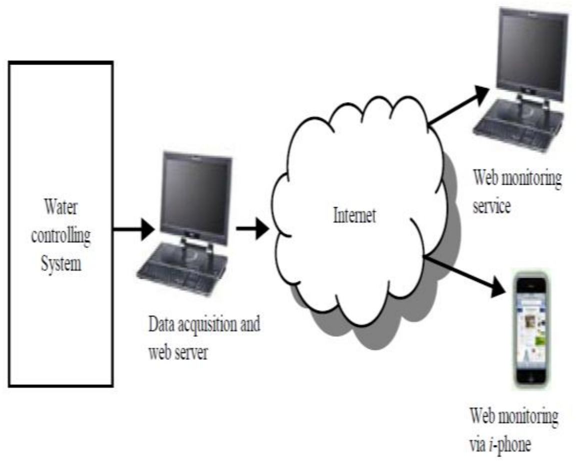

Fig.3 Water Monitoring Network

Due to the fact of controlling remotely we introduced a useful wireless automated controlling system. This newly proposed web based monitoring and controlling network can work with the existing water controlling system successfully.

V. SOFTWARE SPECIFIC REQUIREMENTS

It consists of all the functional and quality requirements of the system. It also describes the system and its features in detail.

1. External Interface Requirement:
This section gives a detailed description of all inputs into and outputs from the system. It also describes the hardware, software and how they interact and provides basic prototype of the user interface.

i) User Interface:
User interface for the system is available on the Android app and web portal. Both of these are designed with minimalistic design keeping in mind the ability to use the CPU and the GPU as less as possible.

The Android app uses a dramatic and immersive representation of water tank and shows a beautiful wave like animation.
The user interface for the Android app is designed using manipulations of XML and Java. The laws of physics are used for the accurate representation of waves.
The web portal uses material design using "Materialized CSS" as well as some bootstrap for responsive design.

ii) Hardware Interfaces:
The hardware is an integral part of our project. The following are the hardware components that we have used in our project.

- Raspberry Pi
- Ultrasonic Sensor
- Jumper Wires
- Android device

iii) Software Interfaces:
The input from the hardware is systematically processed and delivered using the following technologies.

- Node.js
- Paho MQTT Client
- Materialized CSS

iv) Communication Interface:
A client-broker architecture based protocol for machine to machine communication. MQTT is a very powerful yet lightweight protocol with a packet header of 2 bytes.
The MQTT client first connects to MQTT broker. In the case of this project the broker is located at AWS instance with a DNS broker running at port 1883.
After being connected the client subscribes to a particular topic. When data is published on that topic the subscribed clients are able to read the payload of the MQTT packet and interpret the information.
This project will have a network server that is web-based. The server exists to retrieve information from the database and process on data.
The HTTP server will use a MQTT connection to get the data. Furthermore, whenever a user opens the Android app from their phone, a pull protocol will be used to retrieve and sync the latest transaction updates from the server.

2. Functional Requirements:
The functional requirements for the system are divided into following categories:

i) Getting input data:
The controller will use water level sensor for sensing water level. Raspberry Pi will fetch sensor output from controller using its GPIO pins.

ii) Processing data:
Inputted data from sensor will be packed at regular intervals and sent to respective MQTT clients. Web server will put the data in respected datasets for the particular device. Web server will formulate and represent the data in graphical format using HTML and XML in Android.

iii) Displaying the results:
The android app on authenticated user's smartphone will display the result obtained from web server.
Also the results can be seen at web portal maintained centrally.
The end user will have real time status of all the water storage water level. Also the user can command raspberry pi hardware to take further actions such as starting/stopping pumps etc.

3. Performance Requirements:
For the complete satisfaction of a user or customer it is necessary to achieve certain performance targets. Following are some of the requirements we used to evaluate the performance of the system.

i) Real-time ability:
Since this project is based on Internet of Things, it is only useful if the data can be manipulated and viewed in real-time. Despite being a real-time the system it is also event driven. Real-time abilities allow users to take actions in time which is one of the objectives of this project. If it was not a real-time system, users might not be able to shut down the pumps and save water.
This real-time ability is achieved with the help of MQTT and web sockets.

ii) Workload:
The system is designed with client/broker architecture and is beneficial in handling heavy load. Since it is topic based, we can service as many users. The MQTT broker is located in an EC2 instance provided by AWS which is well-known for handling heavy workload and flexibility in providing resources.

iii) CPU Utilization:
The initial attempts of creating this project were not up to the mark due to limited processing abilities of the Raspberry Pi. This problem was solved by using the MQTT protocol.

The only function of the Raspberry Pi now is to send raw data from the sensor to the MQTT broker. It significantly reduced the CPU utilization and increased the speed of the entire system.
All the calculations now are done on the Android device which does not bottlenecks the Raspberry Pi and boosts up the entire system.

VI. FUTURE SCOPE

The current system uses Raspberry Pi 2, which is found to be weak compared to latest Raspberry Pi 3 and Raspberry Pi Zero. The Raspberry Pi 3 has built in Wi-Fi module which was absent in the earlier versions.
Currently the Raspberry Pi and distance sensor are not enclosed by a waterproof casing, we intend to fabricate it with a well-designed waterproof casing to encapsulate the complete module and make it more durable.
We can also apply machine learning algorithms on the data gathered by the sensors and estimate time at which the tank will overflow or when it has extremely low water reserve level. This system will help to create intelligent and automated systems which can reduce effort and economic drags.
We can provide facility of auto-off when tank is reached to maximum level.
Also we can calculate how much time is required for reaching tank to its highest level by using data analysis.

Current system cost for water level monitoring system:
Raspberry Pi 2 with RAM 1GB Rs. 2899
Minimizing costs:
Raspberry Pi 0 Rs.550
Ultrasonic Sensor Rs.80
Jumper wires Rs. 125
AWS Server System Rs.854
Bracket coverage for Raspberry pi Rs.5

Total minimum cost for our product is Rs.1614

VII. CONCLUSION

Many entities and problem solvers have tackled the problem of water wastage in their own way. We have made a serious attempt at solving this problem and saving one of the most precious commodity this world will ever have- “water”. We focused on the problem of water wastage through overflow of water reservoirs or tanks.

This project has successfully addressed this problem and will most definitely help control it. The application uses Internet of Things and Android platform to monitor water level changes and also allows user to administer the water tank from different corners of the world. It has automated the entire process thus reducing energy wastage and manual labour.

The centralized architecture of MQTT worked sublimely and proved to be useful in carrying out most of the functionalities of the system. Android is one of the most widely used platforms for deploying applications. Various features of raspberry pi and its ability to utilize sensor data and support for development environment proved beneficial.

The system is going to help home owners and organizations to cut back economic losses and provide a reliable way to manage water reservoirs without any hassle. Future developments can improve the system and make it more scalable.

REFERENCES

[1] Meena Singh ; TCS Innovation Labs., Bangalore, India ; M. A. Rajan ; V. L. Shivraj ; P. Balamuralidhar, “Secure MQTT for Internet of Things (IoT)”, Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference.
[2] Dinesh Thangavel ; Fac. of Eng., Nat. Univ. of Singapore, Singapore, Singapore ; Xiaoping Ma ; Alvin Valera ; Hwee-Xian Tan, “Performance evaluation of MQTT and CoAP via a common middleware”, Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference.
[3] Seong-Min Kim ; Dept. of Multimedia Eng., Hanbat Nat. Univ., Daejeon, South Korea ; Hoan-Suk Choi ; Woo-Seop Rhee, “IoT home gateway for auto-configuration and management of MQTT devices”, Wireless Sensors (ICWiSe), 2015 IEEE Conference.
[4] Luc Moreau. Sump pump water level. instructables.com: http://instructables.com/id/Sump-pump-water-level-The-software
[5] AWS - Amazon Web Services: http://aws.amazon.com
[6] Ultrasonic distance sensor: https://docs.google.com/document/d/1YyZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit
[7] Raspberry Pi: http://raspberrypi.org/about
[8] XMPP: https://xmpp.org/
[9] MQTT: http://mqtt.org
[10] Note.js: http://notejs.org
[11] IoT: http://en.wikipedia.org/wiki/Internet of Things
[12] Mqtt.js: https://www.npmjs.com/package/mqtt
[13] Paho MQTT Library: https://eclipse.org/paho/clients/android/
[14] Raspbian OS: https://www.raspbian.org/RaspbianImages
[15] https://en.wikipedia.org/wiki/Smart_city