# Remote Monitoring of Dams

Group CHJ

Kaiyang Han
*Electrical and Computer Engineering*
*University of Florida*
Gainesville, U.S.
khan2@ufl.edu

Zhengyuan Jiang
*Electrical and Computer Engineering*
*University of Florida*
Gainesville, U.S.
z.jiang@ufl.edu

Mert Canatan
*Electrical and Computer Engineering*
*University of Florida*
Gainesville, U.S.
mert.canatan@ufl.edu

***Abstract*—This paper proposes a solution for remote monitoring of dams using Internet of Things (IoT). The proposed solution has multiple end nodes and a coordinator node connected with the cloud. The data is collected from end nodes and send it to cloud using the coordinator node. In each end node there are three different sensors namely water level, water turbidity, and water speed which are connected to Raspberry Pi that communicates with the coordinator. From the coordinator data is sent to cloud and the analysis is done on cloud. Unwanted situations (e.g., flooding) are sent back as a notification to be acted on. Depending on the situation, further action is taken such as releasing the dam.**

***Index Terms*—Dam monitoring, Internet of Things (IoT), remote monitoring, Raspberry Pi, WSN.**

## I. Introduction

Dams are the major sources of water supply to cities, they also play a vital role in flood control and can assist river navigation. Most of the dams are built to serve more than one purpose and have numerous benefits. Dams are usually monitored through traditional surveillance techniques. Management of water resources through dams becomes complex as the number of users benefit from dams are abundant. This situation gets much complex with the fact that the available resources are limited with high possibilities of droughts and floods. Manual dam monitoring is a tedious and long term process which has to be improved step by step. Therefore, a new system for dam monitoring and management should be established which can provide water level, water turbidity and water speed in real time and can allow us to come to quick conclusions regarding the safety operations of the dams.

Internet of Things (IoT) can be defined as a network of interconnected devices. It comprises a set of sensors, communication network as well as software enabled electronic devices that enables end users to acquire data from time to time through the communication channel and allows for data interchange between users and the connected devices either locally or on the cloud [1].

Our proposed system can be used to automatize the control of dams without human interference. This can also be used to gather information on the level of water throughout the country and can be used to route water based on the requirements. We can get information on the water availability in a particular region and route the water to that area if there's scarcity. This helps a lot in irrigation. Keeping a check on the safety of dam from time to time is one of the important measure to ensure the safety of dams. Use of Wireless Sensor Network (WSN) with software for dam safety management helps in improving the functionality of dams. All the sensors in the clusters on the dam such as water level sensor, water turbidity sensor and water speed sensor can be used to sense water level, water turbidity, and water speed in centimeters, percentage, and Liters per Hour respectively. In case of floods the routing of flood water can be done more efficiently considering the level of water across different dams. Surveillance of areas near the dams can be done using cameras which transmit live footage to the base station and will be helpful in identifying the presence of people near the dams and can help in ensuring safety while releasing water during flash floods.

Internet of Things focusses on making the ecosystem of sensors more and more intelligent by establishing a connection to the internet. Collecting the data regarding the sensors enables us to generate more reliable equipment which in turn improves the reliability of the dams. Integration of Internet of Things with big data, cloud computing and Wireless Sensor Network will enhance the operation capability to dams to a greater extent. The entire processing of data will be done on the cloud which will ensure that the data retrieval and issuing of commands can be made faster with more reliability and remotely.

Design methodology of this project is as follows:

- **Purpose**: Ensure the water safety of dams against any natural or anthropogenic threats and to develop an effective dam monitoring system.
- **Behavior**: Proposed model is an information system based on the existing systems with the utilization of sensors and IoT, which proposes a novel idea of collecting and sharing real-time information about dams to a cloud system through wireless communication (i.e., WiFi, MQTT).
- **System Management**: The system provides remote monitoring (e.g., through cloud based app) and control functions such as releasing the dam.
- **Data Analysis**: The system transmits data to a secure database to allow experts to analyze data away from the dam area.
- **Application Deployment**: The app notifies the users, and the coordinator node when there is an unwanted situation

(e.g., flood).

- **Security**: The system has a basic user authentication capability.

The architecture of our proposed solution is in Fig. 1.

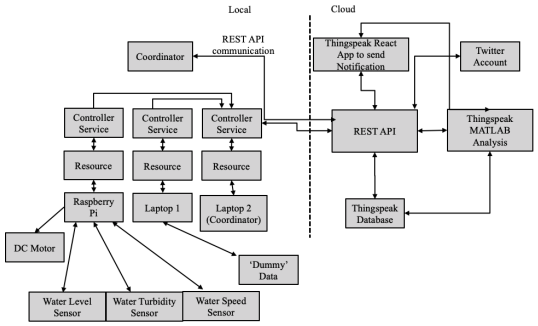


Fig. 1. Architecture of our proposed solution (IoT Level-5 System)

The organization of this paper is as follows: Description of hardware experiments which includes detailed explanation of the three main parts namely sensors, communication, and cloud/app and the conclusion.

## II. Hardware Experiment Description

To gain knowledge about dam monitoring and water management, we researched the field [2], [3], [4], and [5]. After reviewing the literature, we decided on using three property. These are water level, water turbidity, and water speed. To get these data, we used three types of sensors, Cusimax water sensor (Fig. 2), KS0414 Keyestudio Turbidity Sensor V1.0 (Fig. 3) and YF-S201 water flow sensor (Fig. 4). And since all the collected data are analog signals, we used an MCP-3008 A/D convert chip to convert the data and transmit them to Raspberry pi for further processing (Fig. 5).


Fig. 2. Cusimax Water Sensor

The reference voltage of MCP-3008 is 3.3V, which means the highest value we could collect from the sensors is 3.3V [6].

Fig. 3. KS0414 Keyestudio Turbidity Sensor V1.0

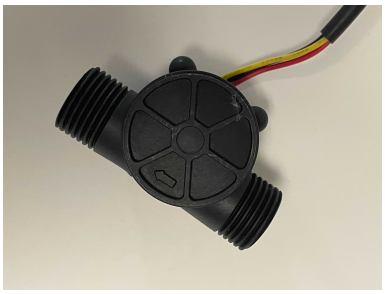

Fig. 4. YF-S201 Water Flow Sensor


Fig. 5. MCP-3008 A/D Conventer

Cusimax water sensor is used to detect water level. It's approach is using the copper line on its surface, when the surface is dry, there is no link between each line, and when it's wet, the route between lines are connected, so that it could pass the current. The voltage is 0 V when it is in a dry condition, and when it is merged, the voltage is 0.53V. The total length of this sensor is 4 centimeters, and we could calculate the transfer function based on this property.

$$h = (d * 4/0.53) + s.\ h. \tag{1}$$

where h is the height, d is the data and s.h. is the set height. Height's unit is centimeter.

The keyestudio turbidity sensor detects water quality by measuring level of turbidity. The principle is to convert the current signal itself into the voltage output through the circuit. Its detection range is 0%-3.5% (0-4550NTU), with an error range of ±05%F*S. When using, measure the voltage value of sensor's Signal end; then work out the water's turbidity by simple calculation formula. This turbidity sensor have both analog and digital signal output modes. The module has a slide switch. When slide the switch to A end, connect the signal end to analog port, can read the analog value to calculate the output voltage so as to get the turbidity degree of water. If slide to D end, connect signal end to digital port, can detect the water whether is turbidity by outputting HIGH or LOW level[7]. If the water is clear, no dust, or other impurity in it, the voltage is 0 V and if the water is completely turbid, the voltage is 0.6 V, this one is tested by putting an opaque object between the sensor, for example, a piece of paper. The sensor tests the percentage of impurity in the water, the range of this sensor is from 0 % to 3.5 %. The transfer function from voltage to turbidity level is

$$t = (0.60 - d) * 3.5/0.60 \tag{2}$$

where t is the turbidity, and d is the data. Turbidity's unit is percentage.

Water flow sensor can detect the water flow speed. To calculate the water speed, the sensor needs multiple detection. There is a wheel inside the sensor, when water flows through it, the wheel will rotate and send high voltage. Therefore, we have to first transfer the voltage to the frequency. In this case, the sampling time is 0.1 seconds and the total time is 1 second, which means there are 10 data. We calculate the non-zero value in these 10 data and divide by the total time that will give out the frequency. Then according to the sensor manual book [8], the equation from frequency to speed is

$$s = 60 * f/7.5 \tag{3}$$

where s is the speed, and f is the frequency. Speed's unit is Liter/Hour.

One of the aims of our project is to release the dam automatically when flood is detected. To meet this condition, we used a motor to control the dam. The motor was a 5 V DC motor, and the driver was L298N [9]. It received three inputs from the Raspberry Pi and give out two outputs to the motor. It controlled the power and the direction of rotation of the motor by these inputs.

To better illustrate our project, we built a physical model, and it is fully functional (Fig. 6).

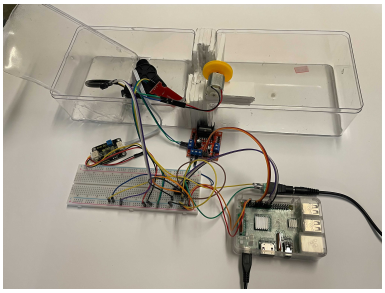

Fig. 6. Physical Model of our system

To collect data, we used Raspberry Pi with Raspbian system (Fig. 7).

Fig. 7. Raspberry Pi with Raspbian system

Then sent the data to the coordinator through MQTT protocol. At this point, we also sent a "dummy" data variables from another laptop to emulat the second end node situation which makes our system IoT-Level 5. At the coordinator, data is decoded and sent to the Thingspeak cloud to be stored, categorized, and analyzed for later. For MQTT communication we selected Python programming language for implementation. Python is a widely used, interpreted, high-level programming, and general purpose programming language. Python's design philosophy emphasizes code readability and concise syntax (especially using space indentation to divide code blocks, rather than using braces or keywords). Python allows developers to express ideas with less code. Whether it is a small or large program, the language tries to make the structure of the program clear [10].

MQTT is a lightweight IoT messaging protocol based on the publish/subscribe model. It can provide real-time and reliable messaging services for networked devices with very little code and bandwidth. It is widely used in IoT, mobile internet, smart hardware, Internet of Vehicles, electric power and other industries [11]. We selected MQTT messaging protocol because our system is a low power system and the messaging can be done efficiently using lightweight protocol.

Next, the paper will mainly introduce how to use the paho-mqtt client library in a Python project to realize the functions of connecting, subscribing, unsubscribing, sending and receiving messages between the client and the MQTT server. To different MQTT libraries, we selected the paho-mqtt. Paho-mqtt is currently the most used MQTT client library in Python. It provides support for MQTT v3.1 and v.3.1.1 for client classes on Python 2.7 or 3.x. It also provides some helper functions to make it very easy to publish messages to the MQTT server.

The project used the free public MQTT server provided by EMQ X, which is based on EMQ X's MQTT IoT cloud platform. Server access information is as follows:

1) Broker: broker.emqx.io
2) TCP Port: 1883
3) Websocket Port: 8083

The coding part required the MQTT connection function. We wrote the connection callback function on_connect, which will be called after the client is connected. In this function, you can judge whether the client is connected successfully or not according to terminal console (rc). Usually at the same time we will create an MQTT client, which will connect to broker.emqx.io. First define a while loop statement. In the loop, we will set the MQTT client publish function to be called every second to send messages to the /python/mqtt topic. Write the message callback function on_message, which will be called after the client receives a message from MQTT Broker. In this function, we will print out the name of the subscribed topic and the content of the received message.

At this point, we have completed using the paho-mqtt client to connect to the public MQTT server, and realized the connection between the test client and the MQTT server, message publishing and subscription.

Unlike high-level languages such as C++ or Java, Python is more suitable for device-side business logic implementation. With Python, we can reduce the logic complexity of the code and reduce the cost of interaction with the device. We believe that Python will have wider applications in the field of Internet of Things.

On the cloud part we used Thingspeak which is an IoT cloud platform supported by MATLAB. Three different type of data from the coordinator node is uploaded to cloud and then the uploaded data is stored in three different time series Figures 8, 9, and 10. After data storage, MATLAB Analysis tool on Thingspeak is used to analyze the data. In the data analysis part we first read the data from using the Channel ID and then processed the data on MATLAB (processing here means getting the average value of the each sensor values to use as a threshold later). After we processed the data, we wrote the results on different fields as numeric value. We used TimeControl app on Thingspeak to collect and act on data in a regular interval which is 30 seconds in our case. Then, in React app on Thingspeak we set condition for three different scenarios for water level, water turbidity, and water speed. If there was an unwanted situation, the ThingTweet app in Fig. 11 sent notification on Twitter to user and put in another column as a "Boolean" value in data storage page. The latter part was done regarding only the water level sensor. We put those in "Boolean" format to read easily from the coordinator node to pass it to Raspberry Pi. At Raspberry Pi, we activated the motor to release water. We also wrote a code to prohibit continuous open/close, in other words, if the dam is already open, it will ignore the next open command.

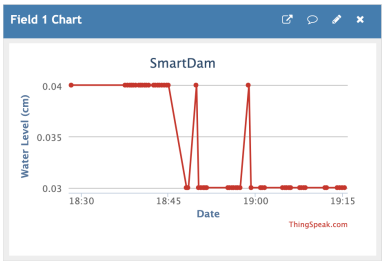


Fig. 8. Water Level Sensor vs. Time

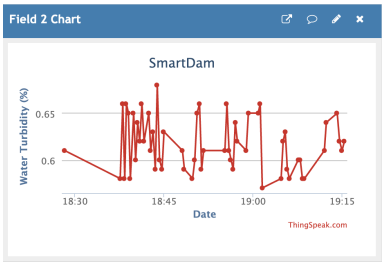


Fig. 9. Water Turbidity Sensor vs. Time

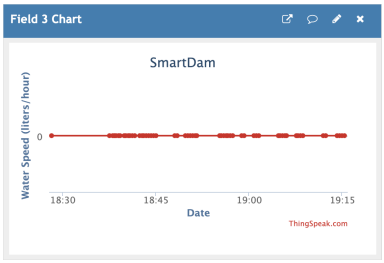


Fig. 10. Water Speed Sensor vs. Time

Comparing with our proposal, we simplified the decision procedure. We were planning to use a machine learning algorithm to detect the flood, but there was a problem: we did not have enough data to train the network to make decisions. To complete the whole processing procedure, we use threshold value to detect the event, that is a simplified decision approach for machine learning algorithms.

Our proof of concept shows that the method is practical and could be applied to solve real world problems. To expand our project to the big picture, we thought the sensors should be more durable for the working condition is extremely arduous and more sensors could be applied to detect other parameters, like water temperature, dam pressure, dam completeness, etc. The decision method could also be improved from threshold


Fig. 11. Notification to User

decision to machine learning algorithm. Other functions like manual control, broadcasting alarm could also be taken into consideration.

## III. Conclusion

For the selection of our equipment, we discuss their special feature according to our project goal. The reasons why we select the water level sensor, the KS0414 Keyestudio turbidity sensor, and the water speed sensor are based on the following reasons. The water level sensor helps us to measure the level of the water which is important detecting flood. The keyestudio turbidity sensor gains the level of turbidity by converting the received signal into output voltage. Besides, the keyestudio turbidity sensor's specification fits into our environment emulation. For example, the operating voltage is DC 5V. It is easy to use in our department. The water speed sensor helps us to measure the speed of the water.

As for the requirement for energy consumption, the operating current is about 11mA. Even if we consider the extreme environment situation, the sensor still covers the detection temperature. The storage temperature is from -10°C to 80°C. The rest list selection of sensors is based on the same standard, such as YF-S201 and MCP-3008 etc.

For the difficulties part, we have to mention the covid-19 makes us communicate through zoom, which makes the communication and equipment debugging harder than before. For the detailed technology difficulties, especially transmit layer, we have trouble in selecting which type of transmit protocol. Because none of us has used MQTT before, we search a lot of information efficiently, increase scalability, reduce network bandwidth consumption and is very well-suited for remote sensing and control. Finally, for the coding part, we search the Internet for an achieved model and learn from it.

Apart from MQTT communication part, the cloud platform selection is another challenge for us. Our first choice is the Amazon Web Service, we study it and learn how to use it. However, it has too much function for us to use it. We have intense debate and finally we turn to select Thingspeak with the advice for our instructor Dr. McNair. To ensure the combination accuracy, the code for the MQTT communication also modified to fit with each other and work together.

The whole project requires our teammates to work together and great communication. It will not work successfully without any teammate. We would also want to say thanks to our instructor Dr. McNair, who is very patient and knowledgeable to give us many useful and practical suggestions. For the duty on each teammate in detail. The sensor part and hardware installation is done by the Kaiyang Han. Zhengyuan Jiang is in charge of the computer communication part and uses his own laptop as the coordinator achieving the MQTT from Raspberry Pi to the Thingspeak cloud platform. Mert Canatan finished all the parts for the cloud and data management including notification to user.

## References

[1] Siddula, S.S.; Babu, P.; Jain, P.C. Water level monitoring and management of dams using IoT. In Proceedings of the 2018 3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), Bhimtal, India, 23–24 February 2018; pp. 1–5.
[2] Niteen Mohod, "Usability of Internet of Things [IoT] For Dam Safety and Water Management", International Journal of Research in Advent Technology, Vol.5, No.1, January 2017.
[3] S. S. Siddula, P. C. Jain and M. D. Upadhayay, "Real Time Monitoring and Controlling of Water Level in Dams using IoT," 2018 IEEE 8th International Advance Computing Conference (IACC), Greater Noida, India, 2018, pp. 14-19, doi: 10.1109/IADCC.2018.8692099.
[4] Siddula, S.S.; Babu, P.; Jain, P.C. Real Time Monitoring and Controlling of Water Level in Dams using IoT. In Proceedings of the 2018 3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), Bhimtal, India, 23–24 February 2018; pp. 1–5.
[5] Martac, R.; Milivojevic, N.; Milivojevic, V.; Cirovic, V.; Barac, D. Using internet of things in monitoring and management of dams in Serbia. Facta Univ. Ser. Electron. Energ. 2016, 29, 419–435.
[6] https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf.
[7] https://wiki.keyestudio.com/KS0414$_{K}eyestudio_{T}urbidity_{S}ensor_{V}$1.0.
[8] https://www.hobbytronics.co.uk/datasheets/sensors/YF-S201.pdf.
[9] https://www.sparkfun.com/datasheets/Robotics/L298$_{H}{}_{B}ridge.pdf$.
[10] https://zh.wikipedia.org/wiki/Python.
[11] https://mqtt.org/.