PAPER • **OPEN ACCESS**

Water monitoring system design for data collection at specific intervals via cloud application

To cite this article: A N Abdul Muta'Ali *et al* 2021 *IOP Conf. Ser.: Mater. Sci. Eng.* **1078** 012002

Water monitoring system design for data collection at specific intervals via cloud application

A N Abdul Muta'Ali[1,2], N Sazali[1,*], S A C Ghani[1] and J Walter[2]

[1] Faculty of Mechanical & Automotive Engineering Technology, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia
[2] Hochschule Karlsruhe-Technik und Wirtschaft, Karlsruhe University of Applied Sciences, 76012 Karlsruhe, Germany

*Corresponding author's email: azlianie@ump.edu.my

Abstract. With the growing issue of water pollution and resources, the need for a remote water monitoring system has never been more crucial. The implementation of remote monitoring systems has a growing interest in the current technology development. This project aims to design a water monitoring system via cloud application. Based on previous research and papers written on the topic, a few system designs have been theorized but not implemented. In this project, the data is transmitted through the combination of LoRa and Wi-Fi connection. The system is also powered by rechargeable battery and solar panels. The communication using long range achieved a distance test of 898 m. The full system has the capability to be deployed in a distance and have the data collected uploaded to a cloud. A web server accessed through the local IP address serves the user interface that provides current data of the water parameters.

Keywords: Monitoring System; Cloud Application; Distance Test; System Design.

1. Introduction

Water is arguably the most important resource on Earth. With the current environmental situation, water management has never been more crucial. There are 148 countries that share 267 international river basins [1,2]. 60% of the water from this flow around the world. As urban population increases, some cities have resorted to drawing fresh water for distant watersheds due to the depletion and pollution of local surface and groundwater sources [3]. Monitoring water quality allows for data collection needed to safeguard the environment against contamination from point sources. Water quality is a description of the water conditions, the chemical, physical and biological characteristics, with respect to its suitability of its purpose. Water quality is usually sampled and analyzed in laboratories. Since the late 20th century, systems were developed to collect real-time data of the water source. In 2009, with the increasing attention to the controversial drilling technique for natural gas production near Susquehanna river basin, Nexsens Technology came up with a solution of monitoring the water quality and made it available to the public to keep tabs on their local rivers [4]. Over 40 remote monitoring stations were installed as water monitoring became more crucial as day passed. Remote monitoring refers to the specification that helps monitor network operational activities using remote devices known as probes or monitors. In the early 21st century, water remote monitoring began to be deployed to monitor parameters [5-8]. Remote monitoring systems are possible through various methods of data transmission such as satellite, cellular, radio and even telephone modem. Each of these methods carries their own pros and cons, as documented by predecessors before. The methods vary as to meet different needs. Water monitoring for large water bodies has always been one that requires a lot of time and effort. Real time data collecting from the water source is necessary to ensure its quality. The data collection requires professional personnel to go out to the field and collect the data during that time. The data collected covers only as of that time. There is no continuous data collection done. This makes it difficult to detect contamination in the water as there is no way to update on the current water quality. Several studies

have demonstrated remotely sensed data can be used to map water quality parameters that meet the application needs. However, the use of remotely sensed data on an operational basis for monitoring water quality is limited to areas that are within the range of cellular coverage. The environment that is off grid is generally located in areas that are far from civilization and sometimes inaccessible, providing many challenges to maintain wireless networks [9]. It also makes it difficult to return and collect data after several months. One method implemented to retrieve the data is using satellite applications.

Most deployed systems store the data collected internally. The data would then need to be obtained every few weeks or months where personnel would have to physically return to the deployed site [10]. This would not be an issue for deployment within a close and safe area. However, it becomes a problem when the system is deployed further. The journey itself would add up to the cost of the deployment. Water quality monitoring for isolated and remote areas can be quite an expensive task. In the very least it would require personnel to visit the site and manually monitor the parameters using probes. The visits would add up to become a costly operation. There is no real time data collection method available for an affordable price. Available products in the market are designed for professional uses thus carry a heavy price tag in addition to the charge of setting it up. The objective of this project is to design an efficient and cost-effective water monitoring system via cloud. The system is to be able to collect data at specific time intervals, link the microcontrollers in use for long range communication while allowing for the microcontrollers to connect to a cloud system.

Table 1. Components specifications.

Materials	Specification
Microcontroller	● SX1276 LoRa ESP32 ● OLED Screen ● Semtech SX1276 radio
pH sensor	● Module Power 5V ● Measuring range 0-14 pH ● Accuracy ± 0.1 pH (25 °C) ● Response time ≤ 1 min
Temperature sensor	● DS18B20 Digital Temperature Sensor ● Power supply 3- 5 V ● Accuracy: ± 0.5 °C
TDS Sensor	● Input voltage 3.3-5V ● Waterproof probe ● TDS range 0-1000ppm ● Accuracy ± 10 F.S. (25 °C)
Power supply	● Mini Solar Panel 5V ● Rechargeable Li-ion Batteries 3.6V ● TP4056 Lithium Battery charger

2. Materials and methods

2.1. System flowchart and components

The system consists of two ESP32 modules, one to collect data and send it to the other, where the data is then uploaded to the cloud server. The ESP32 connected to the sensors is powered by a rechargeable battery, charged by solar panels. This allows it to be deployed for a longer period. As shown in Figure 1, the general view of how the system works. The two ESP modules are connected using long range

connectivity (LoRa). This allows for data to be transmitted over a large range. The second ESP functions as a gateway to connect to the Internet. The data collected is sent through IFTTT (If This Then That). IFTTT is a free web-based service that creates chains of simple conditional statements, called applets. This applet is a result of triggers and actions to send the reading obtained to the Google Sheet assigned. From the gateway, the data is uploaded over to Google Sheets with the help of IFTTT. The user interface is created in a web browser, accessible through the IP address obtained from the receiver ESP. This allows for the data to be accessed immediately from the gateway. Components specifications are as listed in Table 1 whilst flowchart can be referred in Figure 1.

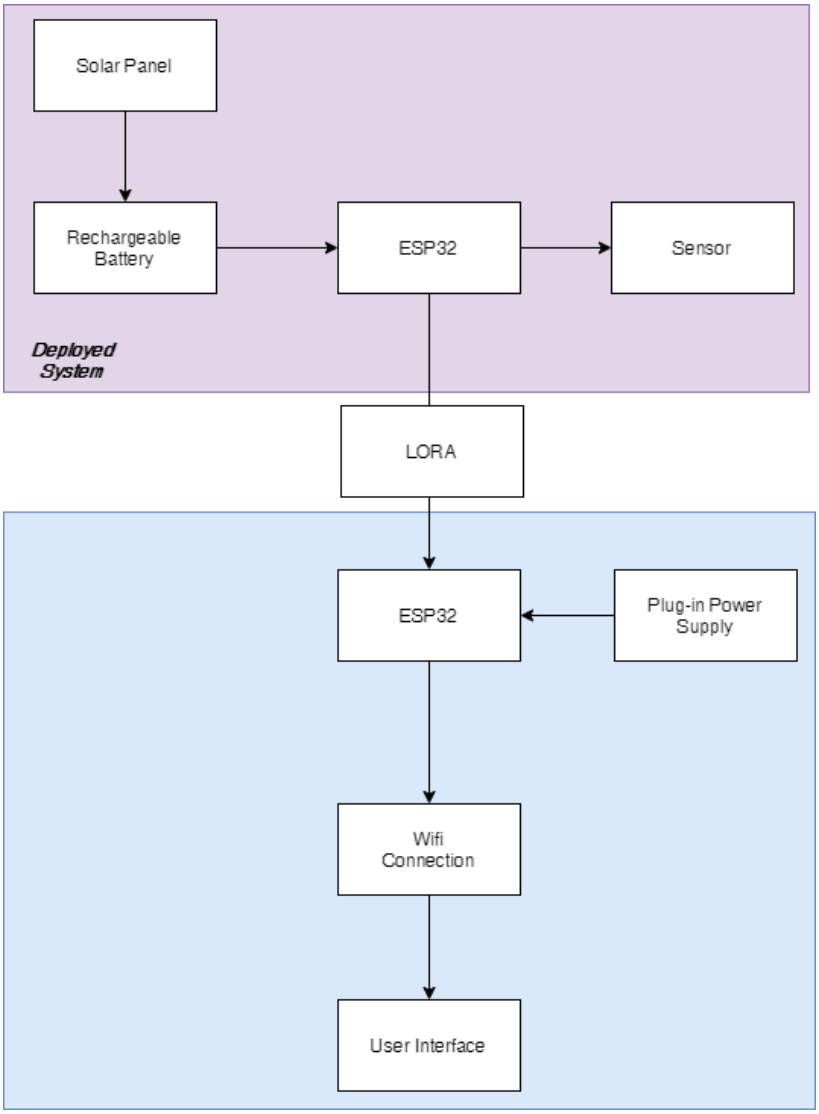

Figure 1. Block diagram of the system.

2.2. Pin allocation plan

The connection of the sensor to the ESP32 is planned as shown in the figure below. The TDS, temperature and pH sensors are connected to GPIO 36, 38 and 34 respectively. All the sensors are powered through the ESP board. With the use of the SX1276 LoRa ESP32 microcontroller, the radio and OLED module are built in. No further additional module needs to be added. As shown in the Figure 2 below, the sensor modules are connected to the 5V supply from the ESP32. The same goes for all the

grounding. There is a 4.7 kΩ pull-up resistor between the Vcc and Data pin of the temperature sensor. This would ensure a more reliable communication.

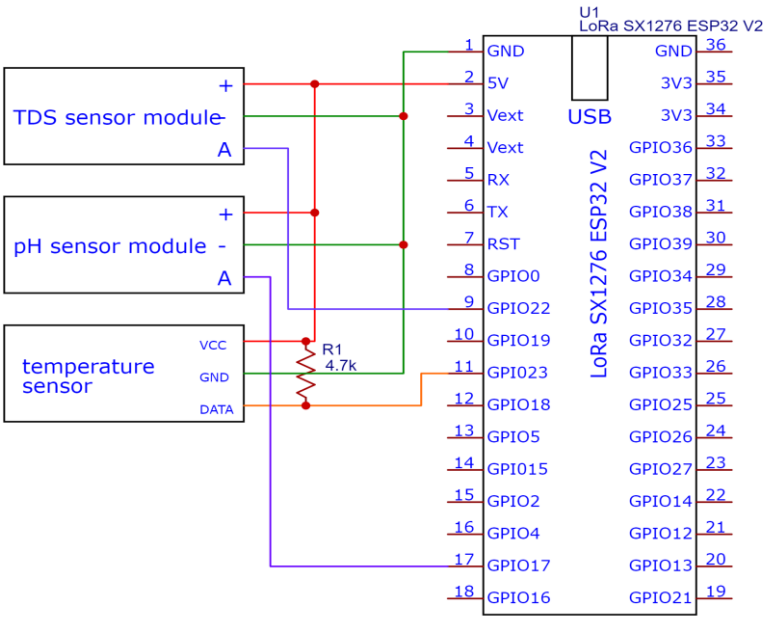

Figure 2. Schematic diagram of ESP32.

2.3. Power source

As the aim of the project is to have the monitoring system be deployed, it would be useful for the system to have its own renewable source of energy. Powering it with solar panels is an alternative method that is put to test in this assembly. The solar panels used have an output voltage of 5V. Connected directly to the solar panels is a TP4056 battery charger module. This module functions as the battery charger. It prevents the battery from overcharging and reverse polarity connection. The battery is a li-ion 18650 battery. The battery outputs 3.6V when it is fully charged. As the ESP32 works with 3.3V, a low dropout voltage regulator circuit needs to be implemented. For this, an MCP1700-3302E is chosen as the low-dropout regulator (LDO). To smooth out the voltage peaks, capacitors are connected in parallel to the LDO. For this, a 100nF and 100uF capacitors are used. The output voltage will then power the ESP32 by connection through the 3.3V pin.

3. Results and discussion

3.1. Sender code

The coding for the Sender requires it to be synced with the Receiver, contains the necessary libraries for the sensors and the ability to transmit the data obtained. For the two ESPs to be able to communicate with each other, a sync word is set. The sync word allows the Receiver to only read the data from a specific source instead of any random radio transmission. The sensors each have their own sensor libraries and settings. As each sensor reading is separate, it is compiled into a single string value, LoRaMessage, which is sent through the LoRa connection. The string consists of a reading ID, the temperature reading, pH value and TDS reading. In between each of the data, a symbol character is added to accommodate to the Receiver's coding. A snippet of the code is shown below in Figure 3.

```
/*------------------Send Data-----------------------------*/
void sendData(){
  LoRaMessage = String(readingID) + "/" + String(temperature) + "&" + String(phValue) + "#" +
String(tdsValue);
 LoRa.beginPacket();
 LoRa.setTxPower(14,RF_PACONFIG_PASELECT_PABOOST);
 LoRa.print(LoRaMessage);
 LoRa.endPacket();
 readingID++;
}
```

Figure 3. Code to string data for lora transmission.

3.2. Receiver code

There are a few additional libraries required for the Receiver compared to the Sender. The ESPAsyncWebServer library provides the web server to request data from the Receiver. The Wi-Fi library is also necessary for the Receiver. The wifi connection needs to be initialized before connection to the server. The SSID and password for the router is inputted in the coding directly as shown in Figure 4. The data received are sent to the IFTTT server with the 'makeIFTTTRequest' function.

```
/*==============set up wifi connection============================*/
const char* ssid     ="bridge";
const char* password  ="password";
/*====================IFTTT server connection=====================*/
const char* resource  ="/trigger/ESP32_water/with/key/c8vajtrYEGq1_Ejhhy4Yt_";
const char* server1   ="maker.ifttt.com";
/*============Create AsyncWebServer on port80====================*/
AsyncWebServer server(80);
```

Figure 4. Setup for the Receiver code.

For the web server, the HTML code is written in the same file. It is also possible to have the HTML be called from a different file however as the coding is not too complicated, it is unnecessary. The readings are connected to the web server with the coding shown in Figure 5.

```
//Route for root / web page
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/index.html", String(), false, processor);
});
server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/style.css", "text/css");
});
server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/index.html", temperature.c_str(), false, processor);
});
server.on("/ph", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/index.html", ph.c_str(), false, processor);
});
server.on("/tds", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/index.html", tds.c_str(), false, processor);
});
```

Figure 5. Sending the data collected to the web server.

3.3. User interface

The web interface of the system should provide a current overview of the condition of the water. Shown on the interface is the latest reading of the temperature, pH and TDS. Along with that, users can also download the data collected in the xls file for further analysis from the Google Sheets. As shown in Figure 6, the data are tabulated automatically through the IFTTT setting. Tabs for each parameter where the graph of the reading collected is shown in Figure 7.

	A	B	C	D
1	Date	temperature	pH value	tds
2	01-01-2020	22.68	6.87615933	51.0
3	02-01-2020	22.68	6.85577404	53.0
4	03-01-2020	22.68	6.49871328	59.0
5	04-01-2020	22.69	6.18471519	64.0
6	05-01-2020	22.69	6.21029986	76.0
7	06-01-2020	22.69	6.77514099	78.0
8	07-01-2020	22.69	6.12274251	80.0
9	08-01-2020	22.69	6.82312351	80.0
10	09-01-2020	22.69	6.12659479	90.0
11	10-01-2020	22.69	6.67174834	93.0

Figure 6. Sample of data obtained from the Google Sheets.

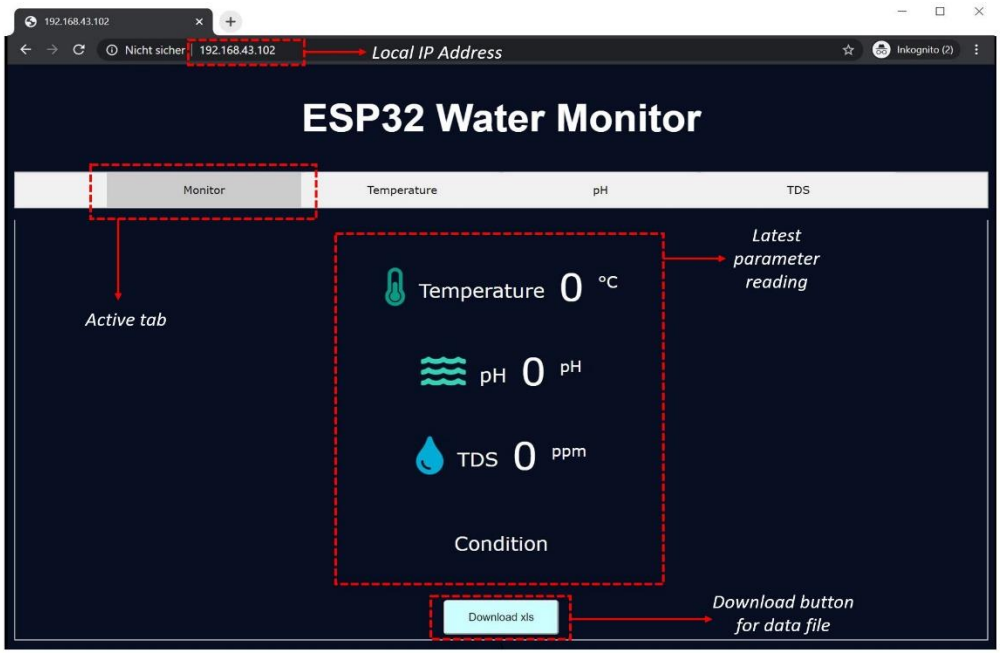

Figure 7. Parameter tab view.

3.4. Distance test

The distance capability of the LoRa connection was done multiple times. The Sender is set up at a fixed location as the Receiver is carried further away from it. The location of the two microcontrollers are then mapped onto Google Maps based on their coordinate to measure the distance. The initial distance test on the ESP had a result of approximately 300m. The sender was located near a window in a building on the fourth floor. Then, the second test was done away from any building interference. The receiver could collect data at 645.18 m. Both ESPs were at ground level. The final test was done to confirm the result from the first and second test. The location test was done around the same location to recreate the same result. The distance observed was approximately 200 m further than that of the predecessor with 896.66 m as shown in Figure 8.

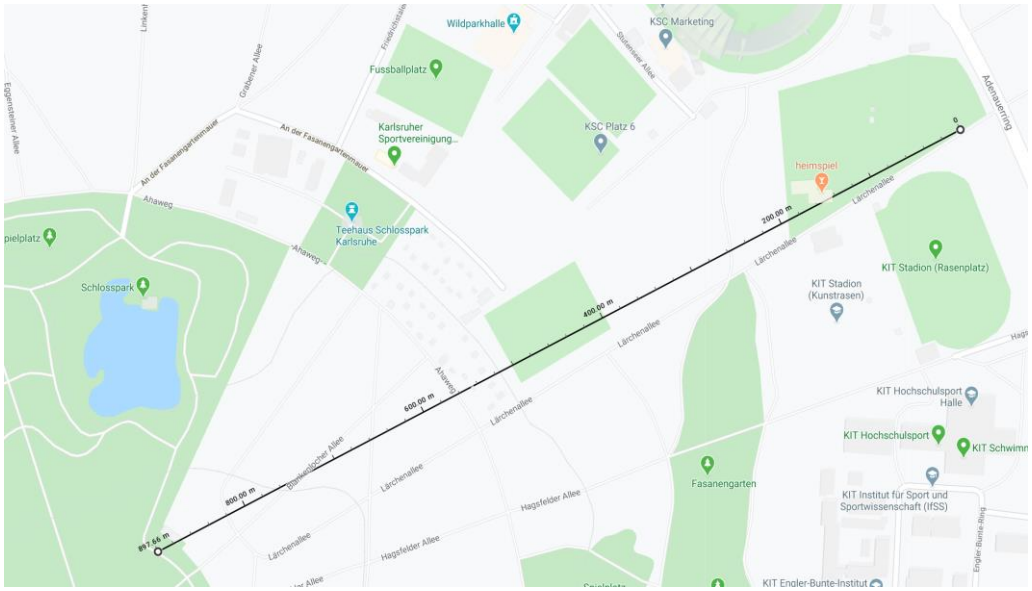

Figure 8. Distance test recorded using Google Map.

4. Conclusion

As a conclusion, a monitoring system that can collect data at set time intervals was successfully created. Any sudden or progressive changes thanks to the water parameters are traceable thanks to the system. Data collected are uploaded directly to Google Sheets via Wi-Fi connection. The system allow for easy data analysis in the long run as the data set is accessible. A web interface is also created to monitor the latest data collected from the system and download the data directly. The connection between the microcontrollers are achieved through LoRa connection for data transmission. The data transfer presented a different way of connection to traditional water monitoring system that are in the market. It is still a relatively new method in its implementation and technologies are ever so growing. There is bound to be new upgrades that the LoRa could offer in the future.

Acknowledgments

The authors would like to acknowledge the financial support from Universiti Malaysia Pahang and Hochschule Karlsruhe-Technik und Wirtschaft (HsKA) under grant number RDU192703 and UIC191514.

References

[1] Masundire H M 2008 *Achieving Sustainable Development and Promoting Developent Cooperation – Dialogues at the ECOSOC* (New York: United Nations)
[2] Biswas A K and Tortajada C 2019 *International Journal of Water Resources Development* **35** 913-916
[3] Jha M K, Sah R K, Rashmitha M S and Sinha R 2018 *International Conference on Inventive Research in Computing Applications*
[4] NexSens Technology 2019 Accessed at https://www.nexsens.com/blog/water-monitoring-buoy- systems.htm
[5] Antonini K, Langer M, Farid A and Walter U 2017 *Acta Astronautica* **140** 10-17
[6] Wang X, Ma L and Yang H 2011 *Proceedia Engineering* **15** 2680-2684
[7] Kim Y, Schmid T, Charbiwala Z M, Friedman J and Srivastava M B 2008 *Proceedings of the 6th ACM Conference on Embedded Network Sensor System* 309-322
[8] Jiang P, Xia H, He Z and Wang Z 2009 *Sensors* **9** 6411-6434
[9] Toledano-Ayala M, Herrera-Ruiz G, Soto-Zarazua G M, Rivas-Araiza E A, Bazan Trujillo R D and Porras-Trejo R E 2011 *Sensors* **11** 71414-7161
[10] Schreiber M E, Schwartz B F, Orndorff W, Doctor D H, Eagle S D and Gerst J D 2015 *Advances in Watershed Science and Assessment* 205-231