Enhanced Knowledge in Sciences and Technology Vol. 2 No. 2 (2022) 266-273

EKST

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/ekst
e-ISSN : 2773-6385

Characteristics of Data Transmission from ESP32 to Arduino IoT Cloud

Muhamad Faiz[1], Afishah Alias[2]*

[1]Muhamad Faiz Zulkefli Aini, Department of Physics and Chemistry,
Faculty of Applied Sciences and Technology,
Universiti Tun Hussein Onn Malaysia, 84600 Pagoh, Johor, MALAYSIA

[2*]Afishah Alias,
Photonics Devices and Sensor Research Center (PDSR),
Department of Physics and Chemistry, Faculty of Applied Sciences and Technology,
Universiti Tun Hussein Onn Malaysia, 84600 Pagoh, Johor, MALAYSIA

*Corresponding Author Designation

DOI: https://doi.org/10.30880/ekst.2022.02.02.028
Received 02 January 2022; Accepted 30 January 2022; Available online 23 November 2022

Abstract: Water storage tanks are used widely around the world to store clean water. However, existing water storage tanks are outdated in terms of innovation. Faulty water tanks usually need to be tended by users manually to be checked. Isolated incidents regarding water tanks may lead to significant problems such as compromised building structural rigidity and leakage of water tanks. Therefore, the project proposed aims to develop a water level monitoring system that utilizes the Internet of Things (IoT). IoT is essentially a medium for things (sensors and devices) to communicate with each other. With the help of IoT, manual supervision can be eliminated through the use of Dashboard. The water level monitoring system comprised of ESP32 will be used to transmit data of water level. Arduino IoT Cloud will be utilized as the dashboard in which users can inspect water levels inside the water tanks without having to manually examine them themselves. The outcome for this project will be a system where users can access the retrieved water level monitoring data in the dashboard. This paper will discuss the delay time and the accuracy of data transmission for the prototype built.

Keywords: Water Level Monitoring, ESP32, Arduino IoT Cloud

1. Introduction

Water has always been an essential need to humans and represents an important role in human society. However, it can be as destructive as the benefits bring. Water causes floods, droughts and landslides [1]. Unpredictable events or accidents also may occur on the tanks that contain water. Industrial and residential usage count for the highest usage of water. Containers to hold water are used in every building and the malfunction of these water tanks is inevitable. Accidents do occur involving water tanks which cost a huge sum of money for the structural damage and appliances damage.

Corresponding author: afishah@uthm.edu.my

Internet of Things (IoT) unlocks the ability to transmit and receive data without human interaction over a network. IoT is defined as a global scale infrastructure for the Information Society, empowering interconnecting things based on existing and evolving, interoperable information and communication technologies reported by The International Telecommunication Union (ITU). Devices connect to other devices through the Internet to communicate and exchange data on a day-to-day basis to achieve specific goals which make our lives easier. Context-aware services are provided from measurement, identification, and process capabilities for communications and information analytics [2]. Systems need to have good reliability and impressive performance to carry out complex tasks. Scaling to larger sizes for the distributed systems is an important factor in the designing phase [3]. Many types of devices utilize IoT in their operations such as microcontroller; Arduino, microprocessor; Raspberry Pi and sensors.

Arduino is classified as a microcontroller. Users need to send instructions to Arduino boards for them to read inputs and convert them into outputs. Inputs such as button pushed or triggered sensors are read and actions like spinning a motor and posting something online are generated. Arduino is created to create a more accessible technology. True to its objective, Arduino has been in thousands of projects ranging from simple to complex projects. Unlike Raspberry Pi which has embedded software to each of its boards, Arduino uses mainly Arduino Integrated Development Environment or Arduino IDE as the medium of input across all of its boards. The main programming language for Arduino is C++. Generic Arduino boards use USB cables to connect to users' computers. Through proper applications, Arduino can be of good use that brings benefits and advantages to our daily life [4][5]. The prototype utilizes ESP32 as the main operating board to process and transmit data.

To build the prototype, a set of testings involving connection to the Wi-Fi, LED circuit and connection between Arduino Uno and ESP32 had to be done. The testings were to ensure the built prototype works as intended and any problems arising can be fixed. Dashboard in which the data will display is created using Arduino IoT Cloud. The circuit assembly and coding program was written to read the input from the sensors and transmit the output to the Dashboard.

2. Components and Methods

The project starts with multiple testings regarding connection (Wi-Fi), LED circuit and connection between Arduino Uno and ESP32. The components used for the projects are; ESP32, LEDs, resistors, jumper wires, micro-USB cable and power bank. ESP32 is a microcontroller, just like Arduino boards but it has Wi-Fi and Bluetooth modules integrated into the board. ESP32 has several variants but the one that is used for this project is DOIT ESP32 DEVKIT V1. The GPIO pins equipped for this board are 30 in total and these GPIO can be modified to be used for various purposes with the right coding program. It has a dual-core and 2.4 GHz Wi-Fi frequency. The board is designed for specifically smart home applications, wearables, automation and cloud-based IoT [6]. Thus, it is a reasonable choice for the project.

2.1 Testings

Testings are done before assembling the prototype. This is mainly for the understanding of the components and should issue arise during prototype assembly, troubleshooting methods are already at hand. The testings are done for Wi-Fi connection, LEDs circuit and boards connection. For ESP32 to create an MQTT connection with Arduino IoT Cloud, it needs to be connected to Wi-Fi. To check if ESP32 is capable of connecting to Wi-Fi, Arduino IDE is used to upload coding programs to the board. Desired Wi-Fi credentials such as SSID and SSID passwords are written in a text document named "WiFi.h" and placed into the Arduino IDE library. The connected Wi-Fi can be observed through the serial monitor.

LED circuit test is done by connecting a simple LED circuit and ESP32. Instead of Arduino IDE, Arduino IoT Cloud is used to configure the connection and variables needed. "random_value"

variable is added to determine if the connection has been made and to match LED according to the value put out by the board. Arduino Uno reads the data from the sensor and transmits the data to the ESP32. The way the coding program functions for Uno is that it will keep on sending data over a serial port with the interval of delay time. Meanwhile, the coding program for ESP32 works to keep on receiving data serially from Uno and establish the MQTT connection. Essentially, this is a simple test to check and establish UART communication between the two microcontrollers.

2.2 Methods

When all the testings are done, the circuit for the prototype is constructed. Figure 1 shows the circuit constructed for the prototype. 3.3V pin on Arduino Uno is connected to the Vin pin of the ESP32 and the GPIO 3 pin of Uno is connected to the GPIO 13 of the ESP32.

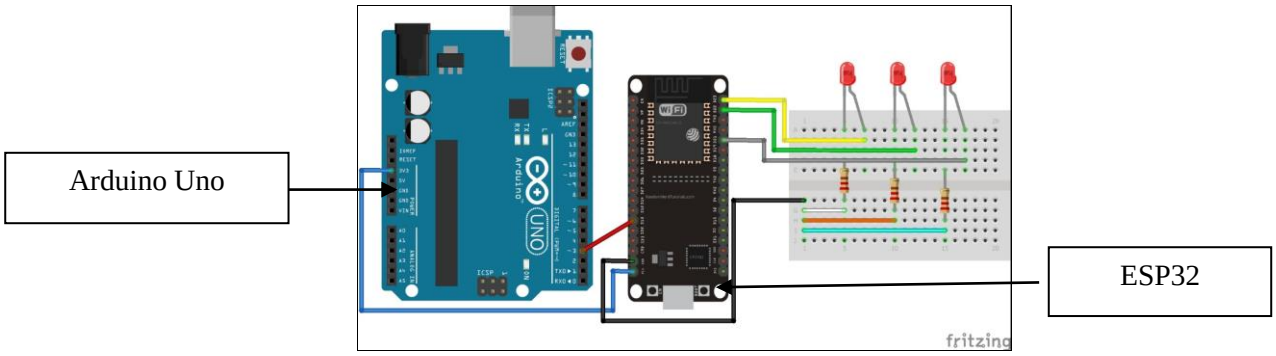

Figure 1: Full circuit layout.

Although ESP32 is the main operating board for the prototype, it is connected to Arduino Uno to read the input from the sensors. Sensors for the water tank communicate directly with Arduino Uno. A3 pin is in the Analog In the section of the board, which functions to convert analogue value to digital representation [7]. ESP32 extracts the data through the A3 pin of the Uno and debug the value to be displayed on the Dashboard. Then, the coding program is written in the Sketch program. A few variables needed to be configured in the Arduino IoT Cloud. All three LEDs used have their variables added as boolean (bool) to function in the manner of a switch. Another variable added is "WaterLevel" in the terms of integer (int).

```
void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  pinMode (23, OUTPUT);
  pinMode (22, OUTPUT);
  pinMode (21, OUTPUT);
   digitalWrite (23, HIGH);
  digitalWrite (22, HIGH);
  digitalWrite (21, HIGH);
  delay(1500);
```

Figure 2: Coding program void setup.

Figure 2 shows the void setup of the coding program. In this part of the coding program, the pinMode function is called to declare the pins for the corresponding LEDs. After the pinMode has been declared, the digitalWrite function is called to determine the output for the stated pins of LEDs.

```
switch (WaterLevel){
  case 1:
  if (WaterLevel == 30){
  digitalWrite (23, HIGH);
  digitalWrite (22, LOW);
  digitalWrite (21, LOW);
  LED1 = true;
  LED2 = LED3 = false;
  }
  else{
  digitalWrite (23, LOW);
  LED1 = false;
  break;
  }
  case 2:
  if (WaterLevel == 50){
  digitalWrite (23, LOW);
  digitalWrite (22, HIGH);
  digitalWrite (21, LOW);
  LED2 = true;
  LED1 = LED3 = false;
  }
```

Figure 3: Switch function in the coding program.

Figure 3 shows the switch function used in the coding program. The switch function works to enable the system to read written code in an organized manner. It allows the system to run the desired outcome based on the variable set for the switch function. For the prototype, the waterLevel variable is set in the switch function. The function is constrained to what is mentioned in the corresponding function itself [8] resulting in the repeated output only coming from the switch function. Therefore, the system runs according to the waterLevel input retrieved from the Arduino Uno. The waterLevel variable is separated into three inputs which are 30%, 50% and 100% water level.

On to the next step, building the Dashboard. Widgets are added suited to each variable in the Arduino IoT Cloud, Dashboard tab. Figure 4 shows the completed Dashboard for the prototype.

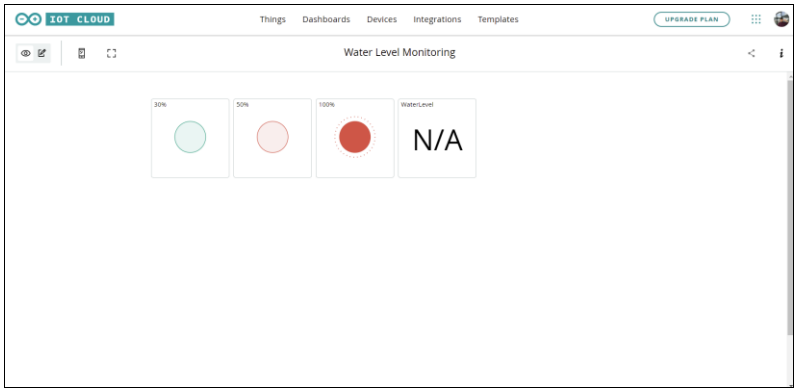

Figure 4: Completed Arduino IoT Cloud Dashboard.

The written Sketch program is uploaded to the board. Uploading the Sketch program will automatically connect the board to the registered Wi-Fi network and establish the MQTT connection between the board and the Arduino IoT Cloud. Figure 5 shows the serial monitor Arduino IoT Cloud which is connected to the mobile hotspot "faiz" and Arduino IoT Cloud.

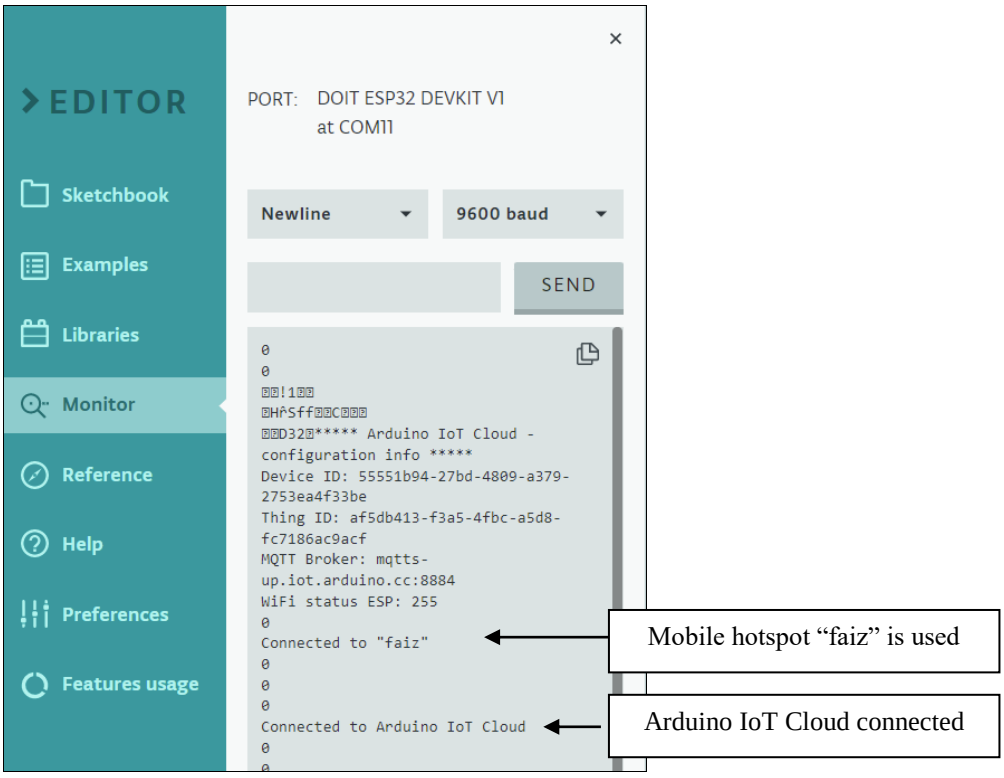

Figure 5: Serial monitor of Arduino IoT Cloud.

3. Results and Discussion

Figure 6 shows the Dashboard viewed from the computer. Figure 7 shows the board connected and in synchronization with Arduino IoT Cloud. ESP32 has successfully established a connection with the Arduino IoT Cloud. The water level data transmitted to the Arduino IoT Cloud is in percentage.

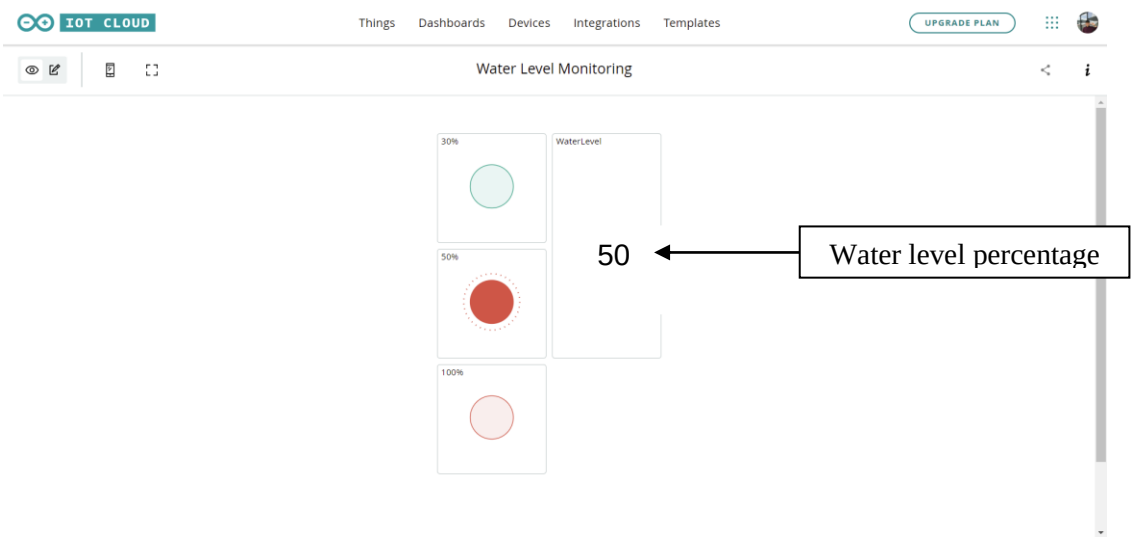

Figure 6: The Dashboard for the prototype.

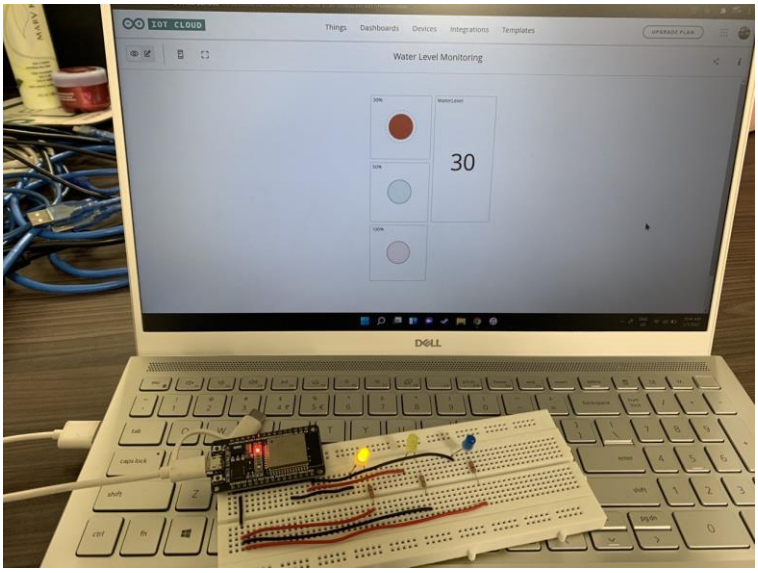

Figure 7: The board in synchronization with the Dashboard.

Figure 7 shows the built circuit able to synchronize with the Dashboard. The overall operation flow of the prototype is sequenced in a very organized manner. Uno transfers the data recorded by the sensors inside the water tank into the ESP32. The ESP32 will then transmit the data to the Dashboard in 5 seconds intervals to avoid any confusion occurring on the written coding program and the connection to the Arduino IoT Cloud. Meanwhile, Table 1 shows the time interval of LEDs change and the time interval of Dashboard update.

Table 1: Delay time between ESP32 and Arduino IoT Cloud.

Time interval of LEDs change (s)	Time interval of Dashboard update (s)
5.00	1.38
5.17	2.43
5.10	1.71
4.89	2.10
5.35	1.99

This was tested by linking the LEDs to the random_value variable. The LED changes according to the changes of the random_value variable. The random_value was set to change every 5 seconds and so do the LEDs. The recorded time interval of the Dashboard update should be given a tolerance of ± 0.5 s for the random error of human reaction. The average time interval of the Dashboard update is 1.92 s.

The time delay may be affected by the mobile network speed. Since the mobile phone utilizes a wireless mobile network that depends on the nearest telecommunication substation, any barrier between the two may disrupt or interrupt the connection and reduce the network speed. Table 2 shows the expected value transmitted from the Arduino Uno, the actual value received by ESP32 and the corresponding LEDs for the value received.

Table 2: Accuracy of data transmitted.

The expected value transmitted by Arduino Uno	The actual value received by ESP32	The corresponding LEDs for the value received
50%	50%	Blue
30%	30%	Red
100%	100%	Green

As shown in Table 2, the actual value received by ESP32 and the expected value transmitted by Arduino Uno is identical. The corresponding LEDs for the value received is also tallying as described in the coding program. This proves the written coding program works in order as intended. The value put out by Arduino Uno via pin A3 can be read by the ESP32 through the "analogRead" function. The function is capable of converting voltage into integer values for specified pins. Thus, values can be read and written from Uno to ESP32 respectively.

The prototype works by utilizing the MQTT protocol which directly transmits the data to the Dashboard. Since the data transmitted by the ESP32 is considered to be minimal, server service and an additional layer of Cloud service are not required. Unlike the prototype, Malche et al. [9] worked on a similar concept of water level monitoring in water sources. The system used different layers for its service and presentation because of the system requirement to process latitude and longitude for the water sources. Moreno et al. [10] displays the perfect example for the use of server service. The prototype recorded water depth, temperature and relative humidity across a river by using mobile nodes. RiverCore server is used to detect the mobile nodes and extract and store the data in real-time.

4. Conclusion

The prototype was designed to facilitate users from industrial and commercial backgrounds to be more aware of the water level in the tank. The point of data transmission for the water level detector is to display the data obtained by sensors on the Dashboard. Like any other communication device, the delay time transmission and receiving data is unavoidable. The average recorded delay time for data transmission of the prototype is 1.92 seconds which is acceptable for the concerned parameter. The working prototype is proven accurate in transmitting data to the Dashboard. It has been tested using the real sensors inside the water tank to rid any ambiguity of any false data shown in the Dashboard.

Acknowledgement

The authors would like to thank the Faculty of Applied Sciences and Technology (FAST), Universiti Tun Hussein Onn Malaysia for its support.

References

[1] David Grey, & Claudia Sadoff. (2006). Water for Growth and Development. https://documents.worldbank.org/en/publication/documentsreports/documentdetail/255681468314995282/water-for-growth-and-development

[2] Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. Computer Communications, 54, 1–31. https://doi.org/10.1016/j.comcom.2014.09.008

[3] Čolaković, A., & Hadžialić, M. (2018). Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. Computer Networks, 144, 17–39. https://doi.org/10.1016/j.comnet.2018.07.017

[4] What is Arduino? (2018, February 5). Arduino. https://www.arduino.cc/en/Guide/Introduction/

[5] Kushner, D. (2011, October 26). Full Page Reload. IEEE Spectrum : Technology, Engineering, and Science News. https://spectrum.ieee.org/geek-life/hands-on/themaking-of-arduino

[6] Babiuch, M., Foltynek, P., & Smutny, P. (2019). Using the ESP32 Microcontroller for Data Processing. 2019 20th International Carpathian Control Conference (ICCC). https://doi.org/10.1109/carpathiancc.2019.8765944

[7] Pan, T., & Zhu, Y. (2017). Getting Started with Arduino. Designing Embedded Systems with Arduino, 3–16. https://doi.org/10.1007/978-981-10-4418-2_1

[8] Eckel, B. (2000). Thinking in C++, Vol. 1: Introduction to Standard C++, 2nd Edition (2nd ed.). Prentice Hall.

[9] Malche, T., & Maheshwary, P. (2017). Internet of Things (IoT) Based Water Level Monitoring System for Smart Village. Advances in Intelligent Systems and Computing, 305–312. https://doi.org/10.1007/978-981-10-2750-5_32

[10] Moreno, C., Aquino, R., Ibarreche, J., Pérez, I., Castellanos, E., Álvarez, E., Rentería, R., Anguiano, L., Edwards, A., Lepper, P., Edwards, R. M., & Clark, B. (2019). RiverCore: IoT Device for River Water Level Monitoring over Cellular Communications. Sensors, 19(1), 127. https://doi.org/10.3390/s19010127