Technial Report IoT

Omar Elwaliely Seifeldin Elshabshiri

April 22, 2024

1 Introduction

The following documents provides a overview of what was done in the project and the results

1.1 Objectives

The objective of this project was to simulate sending and receiving data from sensors over multiple nodes. This allowed for a real-life example, of real-time monitoring as an application of IoT. To execute this task we performed both a simulation of the system using UDP on Cooja and a physical representation using two ESP-32's and BLE. Then it remained to analyze the system.

2 Modules Used

The following section highlights the modules used to create the system.

2.1 Hardware

The hardware consisted of two of the following of each of these modules: ESP-32, LDR, MQ 135, DHT-11, a buzzer, and LEDs. The LDR is used to gather light readings, the MQ 135 used to gather gas readings, the DHT-11 is used to gather temperature and humidity readings, and the buzzer and LEDs are used area actuators that provide feedback on the system. One ESP-32 acts as a client simply sending its values and one acts as a server aggregating and sending back average readings to the client.

The BLE server uses a specific service UUID that is advertised. Within the service we have 4 characteristics each with a different UUID. There are 3 characteristics that have the notify property, these characteristics notify a client with temperature, humidity and gas readings. The fourth characteristic uses a write property that is written 0 to if there is no emergency situation and 1 if there is an emergency situation. The client reads the value from the characteristic to decide whether or not to light the emergency pin.

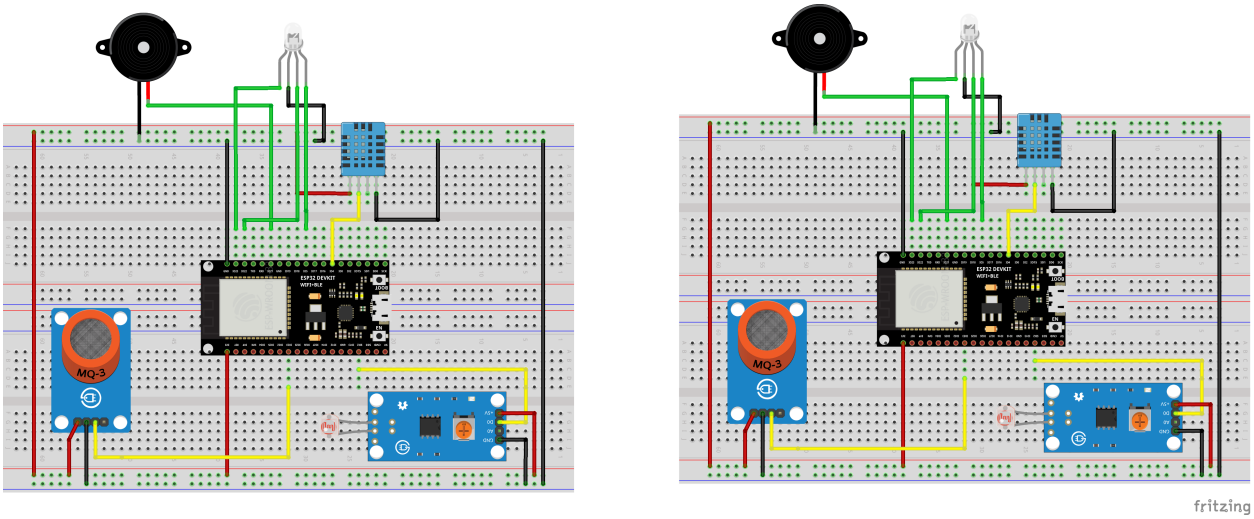

Figure 1: A schematic of the system done on Fritzing, Note that resistance were not added in this schematic and the pins may differ from our physical implementation.

The BLE client is what allows the scanning of BLE advertisers, in this case one of them is out BLE server. The client then checks if the service UUID matches one of the services advertised, if so connects to it. Upon connection the client searches for 3 the characteristics it needs to read from, which are the remote BLE servers temperature, humidity, and gas. Additionally it also searches for the emergency characteristic in the BLE server and writes 1 if there is an emergency situation and 0 if there isn't. This is decided on an average taken by the client on the BLE server's gas reading and it's own gas reading.

The temperature, gas, and humidity readings of the BLE server are also aggregated at the client, to allow the client to communicate the server's and it's own readings to the client and upload them to a real time operating system hosted in the cloud.

2.2 Simulation

The simulation uses Wismote to allow for a larger size. The motes use udp-client.c and udp-server.c as a baseline. Within the file there is also a separate process thread that reads from the sensors. The sensors originally are in file with all sky-sensors, which are all sky mote specific, hence they were edited to use random number generation as values for the sensor. The motes also print LED outputs, buzzer outputs, and actuate red, green, and blue lights.

2.3 Web Server

The webserver uses wifi.h as a baseline library to interact and create a webserver. We use Firebase for hosting and deploying the web interface and for a real time database that is written to from the BLE client ESP32; furthermore we had experience with the technology. A UI was created to show the server which was created with React.

3 Problems Faced

The following section summarizes some of the problems we faced and learned from throughout the project

3.1 Hardware

There were some problems mainly related to the hardware components that are unreliable, for example the DHT11 is very slow and unpredictable. Another major problem was that the client code was too large to fit on the ESP32, to solve this we change the partition scheme of the ESP32 and set it to Minimal SPIFFS (1.9MB with OTA/190KB SPIFFS).

3.2 Simulation

Within the Cooja simulation a multitude of problems were faced. The first issue was the lack of a BLE module. This makes sense as BLE is a link layer protocol; hence it becomes difficult to conceptually see what happens to the data. Cooja uses radio signals to transmit; hence we used it and used UDP as and sent strings as an abstraction that we can manage. The next issue faced was the difficulty of navigation; however this was solved by simply experimenting and reading the header files of different c files. Another issue faced was the small size of sky motes which is 48 kb; hence a Wismote was used which can go to 256 kb. An issue also occured in which adding files would not allow for compilation which was fixed by adding them to the relevant Makefiles. The next issue faced was the inaccurate readings: readings would remain at 65535. To fix this random numbers were generated in the sensors. The final issue was the lack of sound from buzzers and lack of emission of extra LEDs besides the RGB, so prints were simply used to monitor them.

4 Findings

4.1 Hardware

We found that the ESP32 is a good interface to simulate BLE communication between two devices. The amount of libraries on arduino IDE for ESP32 also helped simplify the process for sending and receiving data. The ESP32 also has good interfacing with Firebase because it allows us to quickly and efficiently communicate with a real time database hosted on Firebase.

4.2 Simulation

The simulation exposed that using a simulator is highly beneficial as it allows for scalability. Using the simulator allowed for creating a proof of concept before real implementation. This provided a level of scalability as in our implementation two clients were used to stress test the system. This would be expensive in terms of both time and hardware to perform in real-life without having a reason to do so. The simulation also exposed potential issues: the use of multiple clients causes some contention; additionally, the server may not send always send the readings to all clients. This is expected as Internet of Things emphasizes the prioritization of saving energy over

throughput. Further analysis also showed that this potential issue is not problematic in the long run as it is expected that eventually the client will send its data and receive data from the server given a relatively small amount of time. In fact, incrementing the time between reading from sensors eliminates this problem.

5 Conclusion

In conclusion, this project provided a controlled environment to experiment with and learn more about Cooja, ESP32, and BLE. It also provided a strong starting point and motivation to use these technologies in the future.