Aquarium Management

Alexandre Correia (53298) & Hugo Rodrigues (53309)

May 2022

1 Introduction

Taking care of an aquarium is a hard task when it comes to responsibility and amount of work. In order to keep fish alive, it is necessary to manage the aquarium's environment properly, i.e. according to the fish's living conditions, specially regarding the characteristics of the water - type (salted or fresh), temperature and PH - as well as the light conditions - darker or lighter.

People need to measure manually those parameters very often to make sure that the fish's living conditions are being satisfied. So, it is useful to have a mechanism that gives some sort of alert when those conditions are not being respected, e.g. if the water's PH is not within the appropriate range for the fish, then the user could be notified with that information, instead of manually measuring it consistently.

In the context of the course of Mobile and Pervasive Computing Systems, we aim to develop a system that not only shares the real-time conditions of aquariums with people, but also allows them to set, in real-time, some of the aquariums' characteristics either locally (same network) or remotely (different networks).

2 Architecture

This project consists of the design and implementation of three components, as shown in Figure 1: a physical simulator, a mobile application and a database.

The physical simulator senses and controls the various hardware components, which are accessible and controllable via a single-board micro-controller like an Arduino or ESP32.

The mobile application is the main way of interaction with the user, allowing him to access and modify some characteristics, and define some alerts for each of the aquariums.

For persistently storing every measured characteristic from each aquarium, there is a database which stores that information. Both the physical simulator and the mobile application can communicate directly with each other and with the database. The physical simulator needs to inform, periodically, the database with the measured values so that the real-time state of the aquarium can be stored, as well as to obtain the remote updates from the database. Also, the mobile application needs to get the real-time conditions of

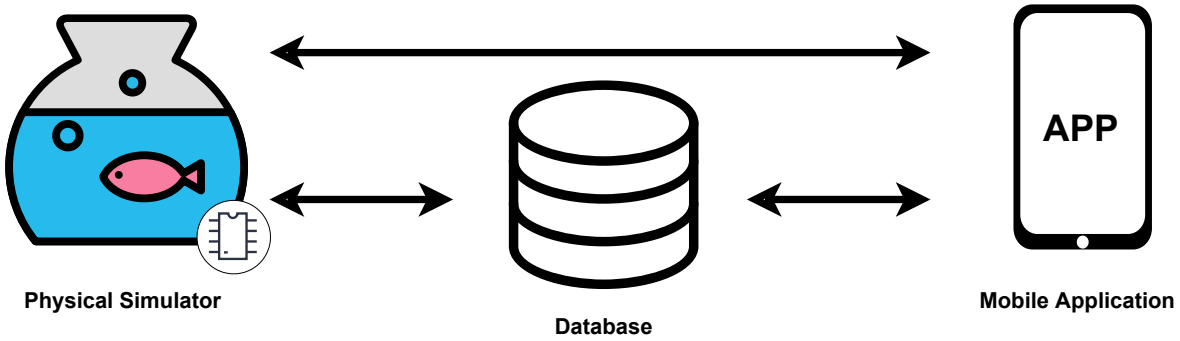

Figure 1: Architecture

the aquarium, periodically, from the database to display them. However, if both of them happen to be in the same network, then they can communicate directly with each other, but the final state of the physical simulator must be stored in the database.

3 Physical Simulator

The physical simulator consists of assembling some hardware components, as displayed in Figure 2:

- Sensors: *2 Potentiometers* and *1 Photoresistor*
  - 1 Potentiometer: for simulating the PH level
  - 1 Potentiometer: for simulating the temperature level
  - 1 Photoresistor: for measuring the ambient light
- Actuators: *4 White LEDs*, *1 Yellow LED* and *1 Red LED*
  - 4 White LEDs: for simulating the internal light with multiple intensities (fixed)
  - 1 Yellow LED: for warning the user that the PH level is not within the correct range (intermittent)
  - 1 Red LED: for warning the user that the temperature level is not within the correct range (intermittent)

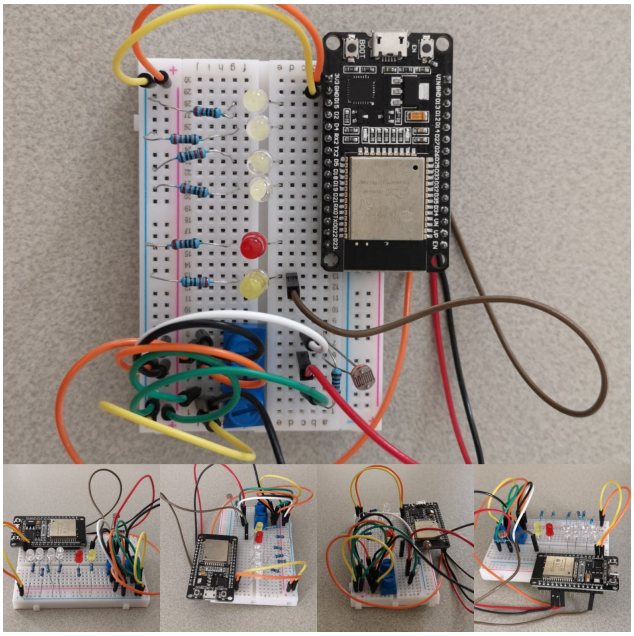

Figure 2: Physical Simulator

4 Mobile Application

The mobile application is composed by three main screens, which are illustrated in Figure 3:

- Aquariums List: displays two lists with the aquariums' names:
  - Local: showing only the local aquariums;

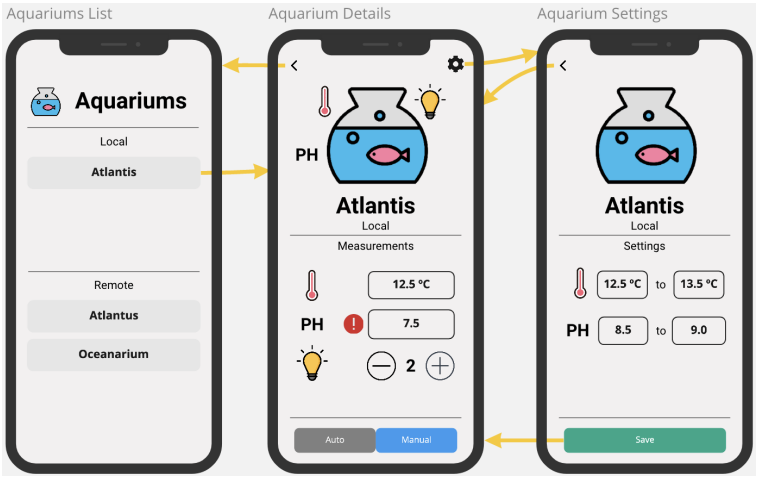

Figure 3: Mobile Application

  - Remote: showing all the remaining aquariums;

- Aquarium Details: displays the environment conditions of the chosen aquarium: temperature, PH and illumination levels, which might be obtained directly from physical simulator (if local) or from the database (if remote). The user can also choose between manually setting the illumination, or letting the system to manage it automatically;

- Aquarium Settings: allows the user to specify the range of values that are allowed for the temperature and PH levels.

When the users open the application, they observe the Aquariums List screen, where they choose the aquarium they intend to control and keep up with.

After selecting an aquarium, they are redirected to the Aquarium Details screen that allows them to observe the environment characteristics of the aquarium selected, as well as controlling the brightness, or simply having it automatic (according to what the photoresistor sensor is capturing).

Finally, when the users tap on the settings button of the aquarium, they can configure the range of values allowed for the temperature and the PH levels of the water.

If the value of any of these parameters (temperature and PH levels) happens to be outside of the specified range, then an alert on the Aquarium Details screen is shown, e.g in Figure 3, we can observe that the measured value for the PH is outside of the defined range of values, so a “danger” icon is shown in the PH measurement section, and the same could happen for the temperature.

5 Checkpoint

5.1 Database

Our database of choice is the Firebase Realtime Database [1] for the following reasons:

- There is no need for advanced querying, sorting or transactions;

- Both the physical simulator and mobile app will be sending a stream of tiny updates;

- Simple data is easy to store.

The database stores data as one large JSON tree, each field being saved as a key/value pair. The aquariums are stored as JSON objects, directly at the root of the JSON tree, where the key corresponds to its id. The remaining data, such as the properties of each aquarium, is kept as a key/value under the aquarium’s JSON, where the key is the name of the property.

```json
{
  "-N2Qy6IYbzSgi--YvBun": {
    "brightness": 1,
    "localIP": "172.20.10.11",
    "manualMode": true,
    "name": "Atlantis",
    "ph": 7,
    "phMax": 8,
    "phMin": 6,
    "ssid": "MEO-A8B113",
    "temperature": 26,
    "temperatureMax": 30,
    "temperatureMin": 20
  }
}
```

Listing 1: Firebase Realtime Database JSON tree with one aquarium stored example

An example of the data stored in our database is shown in Listing 1, consisting of a JSON tree that has one aquarium with the id *-N2Qy6IYbzSgi–YvBun*.

5.2 Physical Simulator

This component is capturing the characteristics of the aquarium - temperature, PH and brightness (when the brightness mode is set to *auto*) - and storing them in the Firebase Realtime Database using the Firebase ESP32 Client [2] library, only when they change over time. Besides, it accesses the database every second in order to keep up with remote changes on the temperature range, PH range and brightness (when the mode is set to *manual*).

In short, the features described above have been implemented (including the warnings of the temperature and PH) only considering remote connections.

Direct/local communications between the Mobile App and this component have not yet been addressed. Although, we are thinking about accessing the *ssid* and *localIP* of the aquariums in the mobile app, in order to check if the device's *ssid* corresponds to the aquarium's one, and if it does, the App can exchange data directly with the ESP32, given its *localIP*. So, we still need to sync these local updates with the remote ones (database), which the former ones will be prioritized against the latter ones.

5.3 Mobile App

From the beginning of the project until the current moment we have been focusing on the development of the physical simulator, as well as the design of the database.

Now that these components are ready, we are now starting to focus on the implementation of the mobile application, using Kotlin, which we will start by addressing the scenario of being remote to the aquarium (fetching from and updating data to the database).

References

[1] *Firebase Realtime Database — Firebase Documentation*. URL: `https://firebase.google.com/docs/database`. (accessed: 19.05.2022).

[2] *Firebase Realtime Database Arduino Library for ESP32*. URL: `https://github.com/mobizt/Firebase-ESP32`. (accessed: 19.05.2022).