Development of Distributed Applications

Design Document

Version 1.0

Sergio Piermario Placanica | 916702

Academic Year 2018-2019

Contents

1 Introduction

1.1 Purpose

Clean, accessible water is critical to human health, a healthy environment, poverty reduction, a sustainable economy, and peace and security. Developing countries are most affected by poor water quality. Up to 80% of illnesses in the developing world are linked to inadequate water and sanitation. In many countries, pollution or rising sea levels are contaminating trusted water sources. It is then critical to monitoring levels and evolution of sea and fresh water pollution. Problem is that full coverage is often inadequate and difficult to obtain because of costly instruments and dimension of the target area.
SeaUrchin is proposed as a possible solution to the problem. The idea is to track pools of water status in real-time by using "swarms" of easy to build and very cheap devices based on esp8266 microcontroller.

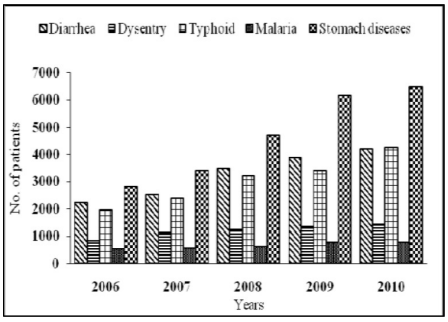


Figure 1: Five years registered data of the patients suffered from water-borne diseases (from a rural community in Lahore, Pakistan

2 Architecture

A great number of water floating devices interconnected that send a real time stream of data to a server using a "Wifi boe" as router. The data is analyzed and different tasks can be accomplished (for example an alert message is dispatched to authorities if sensors indicate an hazardous environment).

3 Technologies

The floating devices will be build using an **ESP8266** microcontroller as core module, the choice was dictated by the fact that it is smaller than an arduino UNO board and has a built in Wifi module, it is also very cheap (see Bibliography for reference).

3.1 Sensors

The goal is to achieve an economically sustainable solution and a wide area coverage, thus the focus should be on producing a large number of cheap devices and accuracy on the single device is not a priority.

3.1.1 List of sensors

Turbidity sensor (see reference) : Turbidity is a measure of the cloudiness of water. Turbidity has indicated the degree at which the water loses its transparency. It is considered as a good measure of the quality of water. Turbidity blocks out the light needed by submerged aquatic vegetation. It also can raise surface water temperatures above normal because suspended particles near the surface facilitate the absorption of heat from sunlight.

Temperature sensor DS18B20 : Water Temperature indicates how water is hot or cold.

is digital type which gives accurate reading.

Water flow sensor (optional): Flow sensor is used to measure the flow of water through the flow sensor. This sensor basically consists of a plastic valve body, a rotor and a Hall Effect sensor. The pinwheel rotor rotates when water / liquid flows through the valve and its speed will be directly proportional to the flow rate.

4 Market analysis

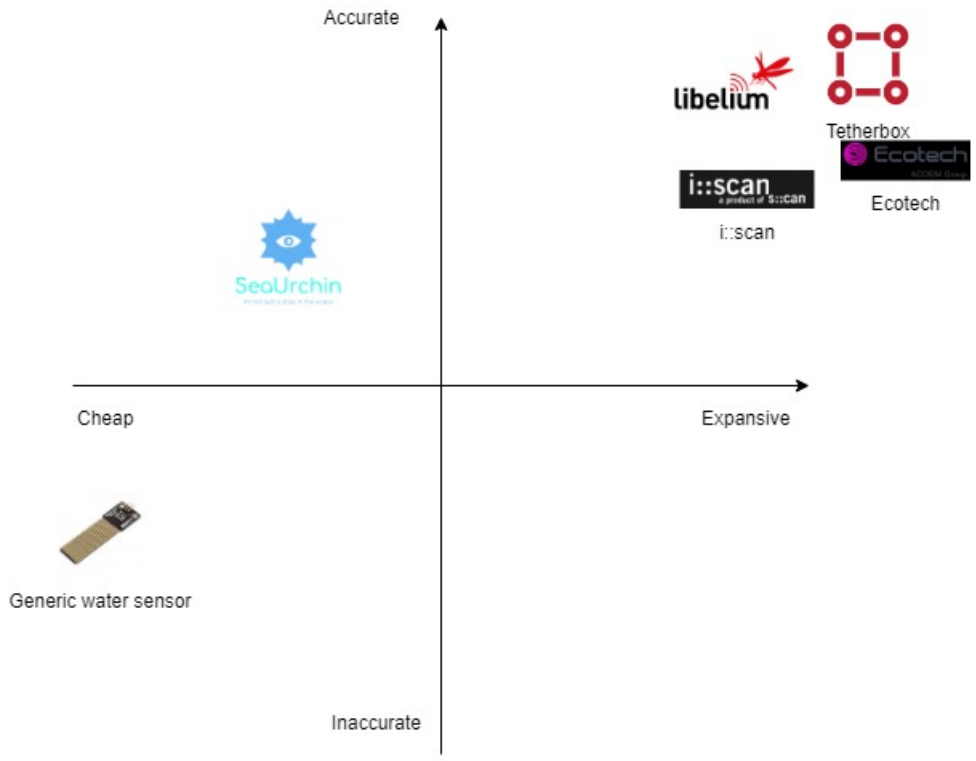


Figure 2: Market Analysis for SeaUrchin, on the X axis costs of the solution, on the Y axis Accuracy, we must keep in mind that expansive solution are not scalable

A rapid market analysis reveal that the majority of the available solutions are out of the scope of SeaUrchin, we want to offer a reliable and fast way to monitor large body of water without a big economic investment.

5 Development

5.1 What was developed

Device Server: device server handles communication between herds and database.

MQTT server: mqtt server is used to manage devices, operation like restart and sleep are handled by this server.

Clients server: clients server offers the API used by SeaUrchin's clients to access the database

mySQL database : the database used to store SeaUrchin data.

nodeMCU v1.0 (ESP8266) code: the code of the sensorArray unit developed in c++ over the nodeMCU 1.0 platform (see reference).

5.2 Documentation

API documentation: the API documentation was generated using POSTMAN, which was used also to test the API correct behaviour.

Design Document: this document offer an overview of the architecture developed.

Code comments: code is commented when necessary, expecially the nodeMCU code.

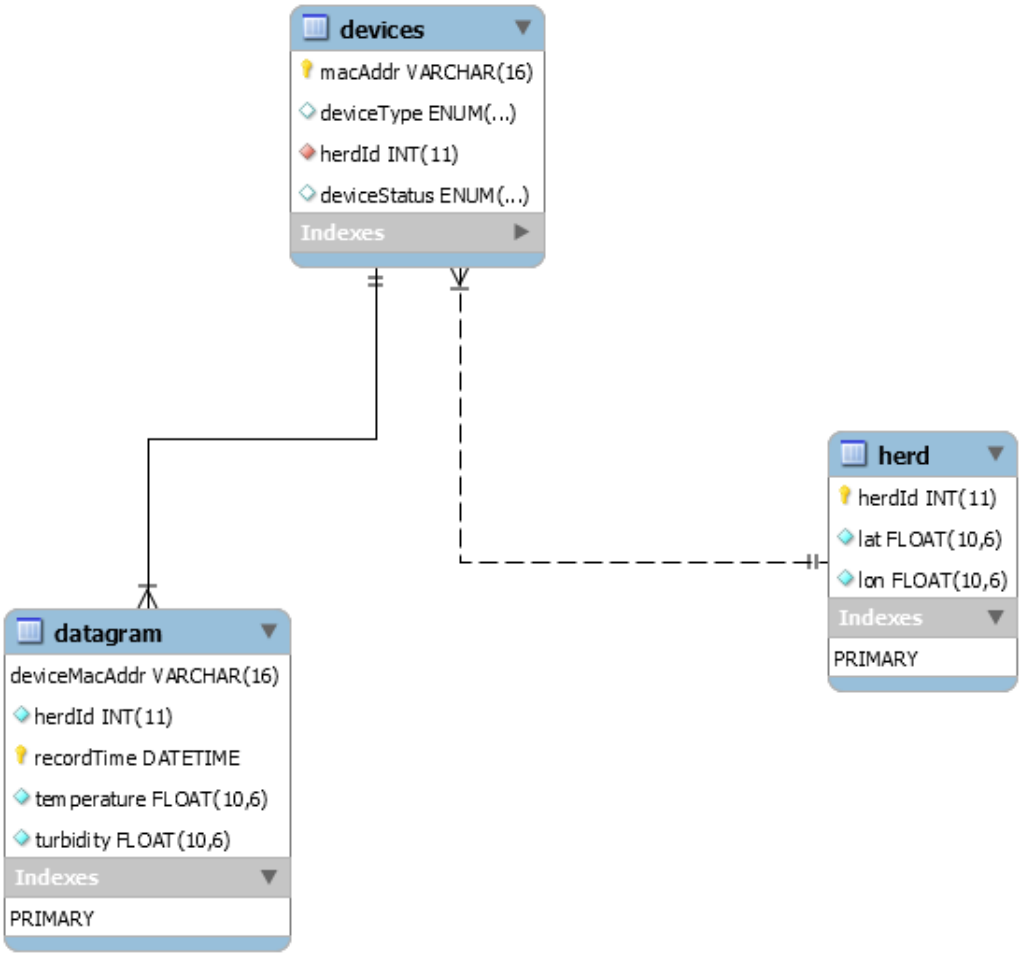
5.3 Architecture

What follow is a short overview of the developed architecture.

5.3.1 Database

The developed database is pretty straigthforward, devices are associated to a herd, an herd is composed of many devices of 2 types : sensorArray and buoyRepeater.
the former device gain data from the water while the latter is a WiFi bridge to the devices Server.
datagram are gained at regular interval and stored inside the database.

Figure 3: DB architecture of SeaUrchin



5.4 developed components architecture

Datagram are sent and received as JSON via HTTP protocol. MQTT is used only to control the devices.

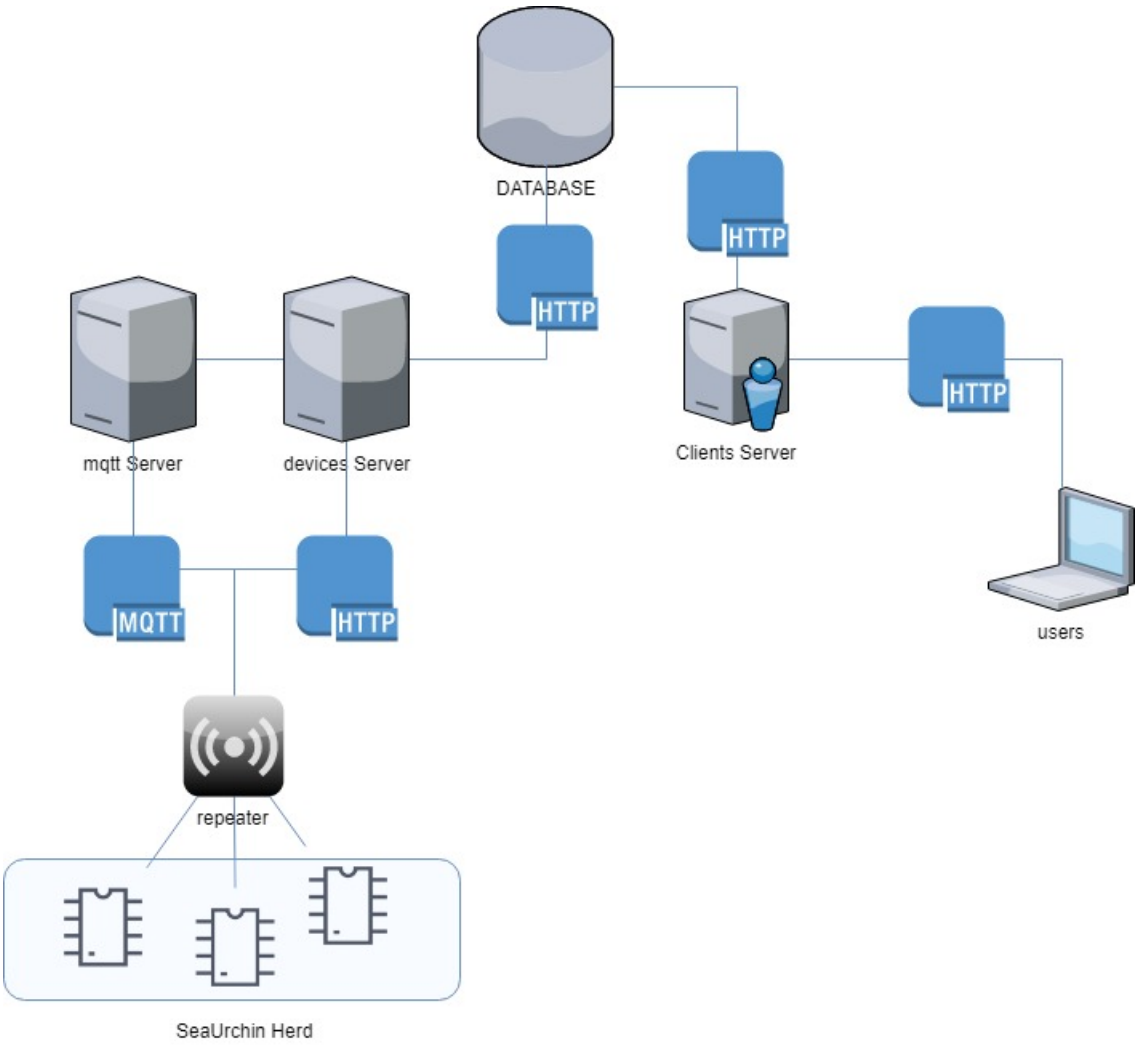


Figure 4: the architecture for seaUrchin system

5.5 API

What follow is a brief description of the developed API

5.5.1 Devices Server

PUT /devices/:macaddr/register used by a device to register itself to the server, 'macaddr' is the MAC address of the device, the body of the put request contain application/json with the type of sensor and the password of the herd.

PUT /devices/:macaddr/datagram used by a device to send a datagram to the server, 'macaddr' is the MAC address of the device, the body of the put request contain application/json with the registered temperature and turbidity of water.

5.5.2 Client Server

GET /herds/:herdId/datagrams used by clients to get all datagrams of 'herdId' as application/json, the datagrams are returned order by device and timestamp.

GET /herds/:herdId/datagrams/?macAddr=*fromDate=*toDate=*status=* used by client to get all datagrams of 'herdId' with the requested filter , the filter are all optional (see full POSTMAN documentation). The datagrams are returned as application/json ordered by device and timestamp.

5.6 MQTT Server

The MQTT server developed is used to control the devices, it can make a device sleep for a set amount of time or reset a device. A device which is in the sleeping state will not send datagrams to the server, however it will be listening on the "devicesbus" MQTT channel, and if it receive a restart signal, it will proceed to restart. The MQTT server is deployed in conjuction to the devices server, only the administrator of the herds could operate it.

5.7 NodeMCU diagram

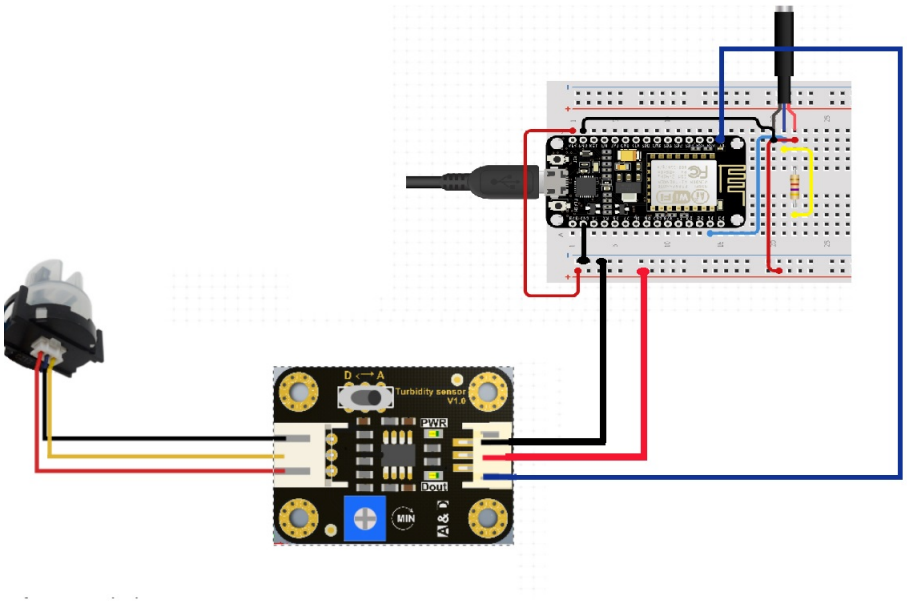


Figure 5: This is the schematic for the connection of the nodeMCU to the sensors

5.7.1 List of components:

NodeMCU x1

ds18b20 temperature sensor.....x1

4k7 ohm resistor.....x1

dfrobot turbidity sensor adapter...x1

generic turbidity sensor.....x1

5.7.2 Connections:

ds18b20 VDD to NodeMCU VDD.

ds18b20 GND to NodeMCU GND.

ds18b20 sensing pin to NodeMCU D0 pin in series with 4k7 ohm resistor.

dfrobot adapter VDD to NodeMCU VDD.

dfrobot adapter GND to NodeMCU GND.

dfrobot sensing pin to NodeMCU A0 pin.

turbidity sensor VDD to dfrobot VDD.

turbidity sensor GND to dfrobot GND.

turbidity sensor sensing pin to dfrobot sensing pin

5.8 how to simulate the system

It is possible to simulate seaUrchin architecture using only an esp8266/nodeMCU microcontroller, a computer and a WiFi network.
Inside the NodeMCU code there is a function called "simulateReading(int LO,int HI)" that return 2 float reading inside LO HI range and can be use to send pseudo random datagram to the server. The code inside "sendDatagram()" function is configured to work with this function.
Inside the server code there's a MySQL script that generate the DB and generate a test herd and a test device.

5.8.1 Steps to simulate are:

1. Change the servers addresses inside the servers code.
2. Change the servers addresses inside the nodeMCU code
3. Run Mysql server on the computer and launch the MySql script to init the database.
4. Connect the NodeMCU to the computer and upload the code.
5. Launch the servers.
6. press the reset button on the NodeMCU
7. monitor servers stdout and NodeMCU serial monitors

5.8.2 Expected behaviour and explanation

Inside the NodeMCU code there's a "herdId" and a "serverpassword" that are the same of the launched devices server, the controller will try to connect to the network and if succeded it will then try to register itself on the device server using the password and on the MQTT server. The server will then add the device to the herd.
At this point if everything went well the device will start sending datagrams to the server with the simulated sensors reading.
At this point the system is working and it is possible to use the APIs provided to query for the registered data or control the microcontroller via MQTT.

5.9 Used technologies

Intellij IDE: used to develop servers code.

Visual Studio Code editor with PlafotrmIO: used to develop NodeMCU 1.0 code, PIO is a framework that enable VSCode to compile and upload code for microcontrollers.

POSTMAN: a tool for testing and documenting API .

MySql Server: the database was realized using a MySql db.

MySql Workbench: to develop and test the database.

draw.io: an online tool used to draw diagrams.

Vert.x 3.7.1: Eclipse Vert.x is an event driven and non blocking scalable framework.

Maven 4.0.0: a notorious wrapper for java projects.

cirtuit.io: a web app to design microcontroller schematics

5.10 libraries used

5.10.1 ESP8266

- Arduino.h
- ESP8266WiFi.h
- PubSubClient.h
- ESP8266HTTPClient.h
- ArduinoJson.h
- OneWire.h (for the temperature sensor)
- DallasTemperature.h (for the temperature sensor)
- EEPROM.h (for storing the turbidity sensor calibration)

5.10.2 Servers

- vertx-core
- vertx-web
- vertx-mysql-postgresql-client
- slf4j-api
- commons-text
- vertx-web-api-contract
- vertx-mqtt

5.11 Conclusion

the developed system is quiet true to the specification.
The cost of the whole sensor array was bigger then it was thought but still contained, the only problem that the deployment of the system could face is the calibration of the turbidity sensor, indeed the sensor need to be precisely calibrated for the environment it will deployed in to produce meaningful data.
newline

6 Bibliography

www.ncbi.nlm.nih.gov - Waterborne diseases in developing countries
(https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3469006/)

www.espressif.com - ESP8266 overview
(https://www.espressif.com/en/products/hardware/esp8266ex/overview)

https://www.dfrobot.com - DfRobot Turbidity sensor (https://tinyurl.com/yxoghzvv)

https://www.electan.com - Water temperature sensor (https://tinyurl.com/y2w3v8aj)