**IOT Project Document - CODE**

Contents

[1. Architecture 1](#_Toc93589539)

[2. Hardware 1](#_Toc93589540)

[3. Software 1](#_Toc93589541)

[4. Install locally (for development purposes) 2](#_Toc93589542)

[4.1. Backend 2](#_Toc93589543)

[4.2. Arduino 2](#_Toc93589544)

[4.3. Esp32 2](#_Toc93589545)

[4.4. Frontend 4](#_Toc93589546)

[5. Deploy to the Internet 4](#_Toc93589547)

[5.1. Backend 4](#_Toc93589548)

[5.1.1. Update Esp32’s firmware 4](#_Toc93589549)

[5.1.2. Update frontend 4](#_Toc93589550)

[5.2. Frontend 4](#_Toc93589551)

# Notes

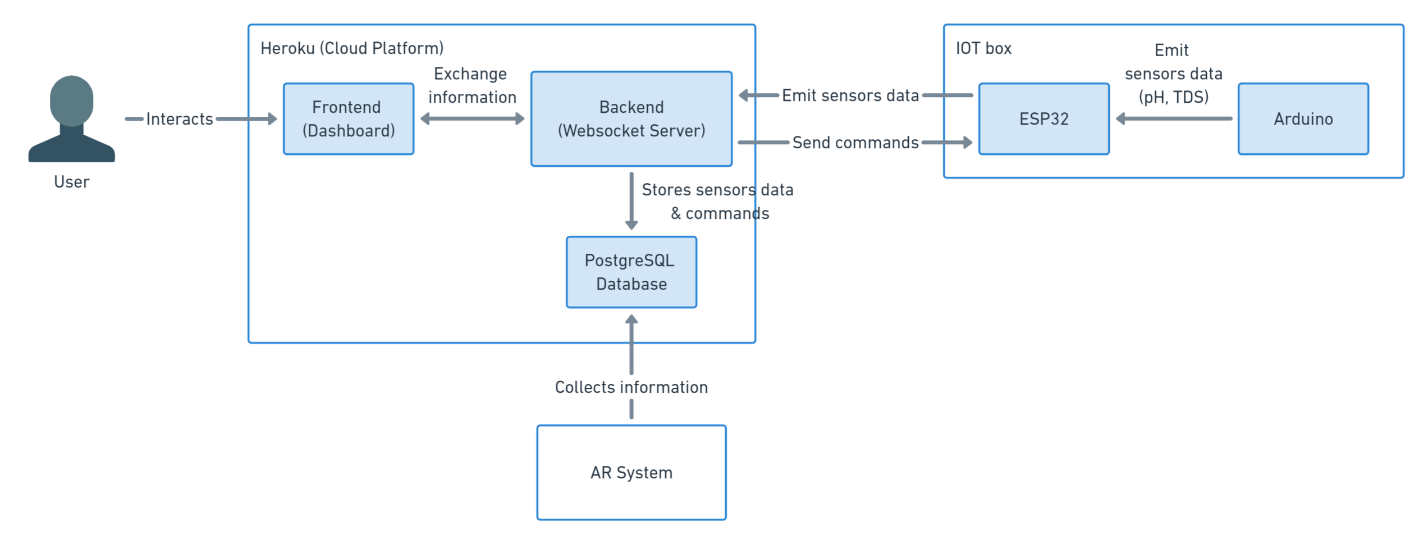
GitHub repository: <https://github.com/thanhlongb/c4s-iot-project/>

Shared to the following email: [manh.nguyentuan@rmit.edu.vn](mailto:manh.nguyentuan@rmit.edu.vn)

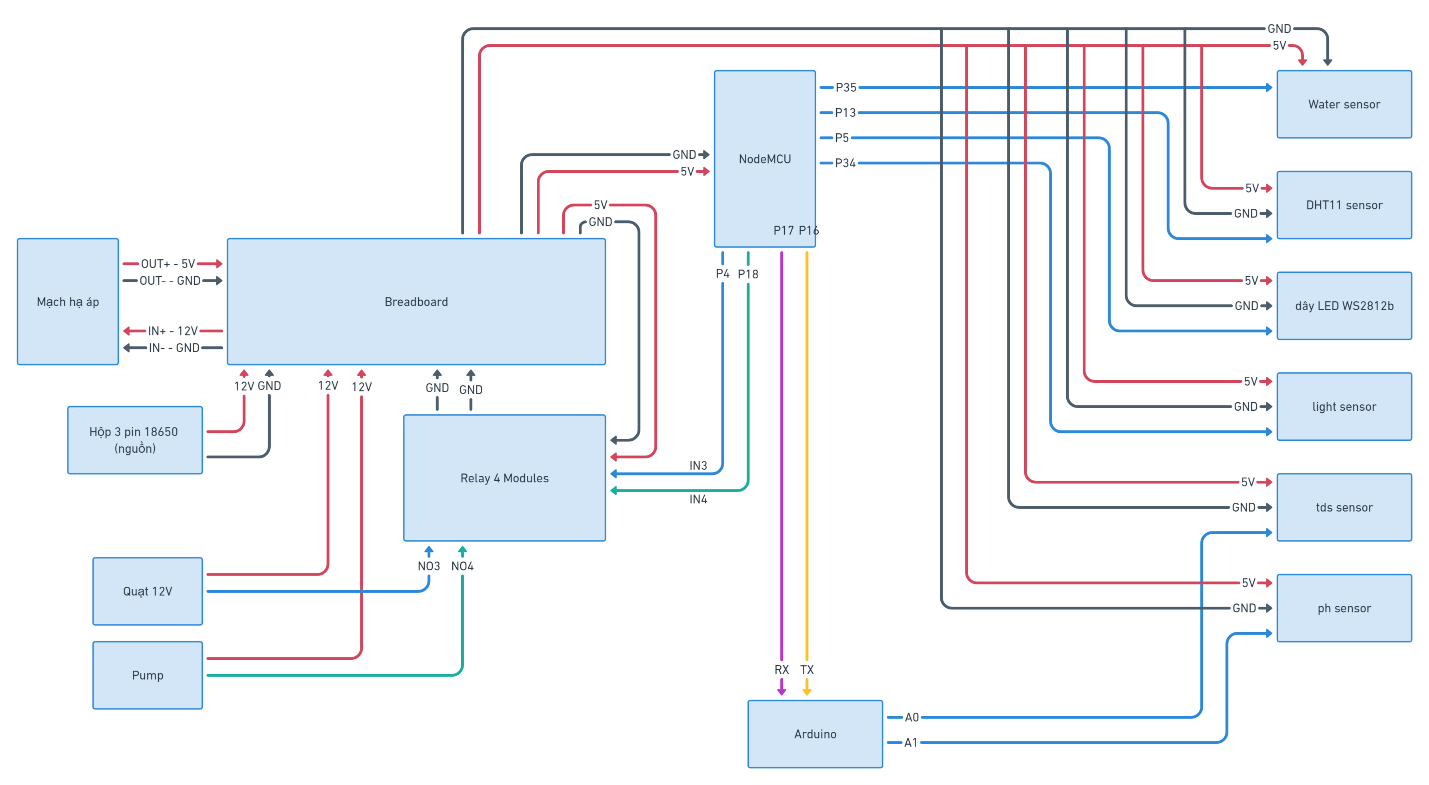
Heroku credentials:

* Username: **dlt.rmit.edu.vn@gmail.com**
* Password: **CODE4School@RMIT**

# Architecture



*Figure 1: System architecture diagram*



*Figure 2: Circuit diagram*

# Hardware

The system consists of three main hardware components:

1. **A laptop or server**: for running the web-socket server, which acts as the middle-man between the Esp32 and the dashboard.
2. **An Esp32 (NodeMCU-32S,** [**datasheet**](https://docs.ai-thinker.com/_media/esp32/docs/nodemcu-32s_product_specification.pdf)**)**: for collecting data from sensors (water level, moisture and humidity, light exposure) and also receiving commands from web dashboard to control other components, specifically: fan, water pump, LED light.
3. **An Arduino UNO**: for collecting data from analog-output sensors such as the TDS and pH sensors since the analog input from the Esp32 give weird results.

Other components are shown in the Figure 1.

# 3. Software

This project’s repository has four main folders:

1. **/frontend**: contains NextJS-based (a framework built on top of ReactJS) application which serve the web dashboard.
2. **/backend**: contains the Nodejs backend code for running a web-socket server.
3. **/esp32**: firmware for the Esp32.
4. **/arduino**: firmware for the Arduino.

# 4. Install locally (for development purposes)

For the system to work, **all three main components must be in the same Local Area Network**, otherwise, they will not be able to communicate with each other. It is advised to setup the system step by step with the below order from top to bottom:

1. Backend
2. Arduino
3. Esp32
4. Frontend

## 4.1. Backend

The backend is a crucial piece in this system. Without it, there will be no data displayed on the web dashboard. Follow these instructions to run the backend:

1. Go to **/backend** folder.
2. Run **npm i**
3. Run **npm start**
4. Done.

## 4.2. Arduino

*\* This section is only FYI since the Arduino already has the latest firmware uploaded to it, there is no need for further re-upload.*

There is nothing much to talk about this one, just upload the firmware from the **/arduino** folder to the component using Arduino IDE (no additional libraries required). Make sure you select the correct COM port corresponding the Arduino itself.

Graphical user interface, text, application

Description automatically generated

*Figure 3: Configurations of Arduino MCU in Arduino IDE.*

## 4.3. Esp32

First off, go through [this document](https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/) on how to add ESP32 boards to Arduino IDE.

For this component, there are quite a few required libraries, please make sure they are installed (refer to [this tutorial](https://docs.arduino.cc/software/ide-v1/tutorials/installing-libraries) on how to install addition libraries for Arduino IDE) in your Arduino IDE:

* FastLED (<https://github.com/FastLED/FastLED>)
* DHT sensor library for ESPx (<https://desire.giesecke.tk/index.php/2018/01/30/esp32-dht11/>)
* ArduinoWebsockets (<https://github.com/gilmaimon/ArduinoWebsockets>)
* ArduinoJson (<https://arduinojson.org/?utm_source=meta&utm_medium=library.properties>)

After having all these libraries installed, follow these steps:

1. Go to **/esp32** folder.
2. Open **remote.cpp** file.
3. Replace the information inside the double quotation marks at line 10 and 11 with your LAN’s Wifi credentials.
4. Grab the device running the backend, open terminal and type **ipconfig** (or **ifconfig** for Macs) and hit enter.
5. Look for the IP of the device that has the backend code running, for example, the IP in the image is **192.168.1.6**.

Text

Description automatically generated

1. Replace the IP into the double quotation mark at line 13 of **remote.cpp**.
2. Make sure the configuration for uploading to esp32 is as follow:

Graphical user interface, application

Description automatically generated

*Figure 4: Configurations of ESP32 MCU in Arduino IDE.*

1. Upload the firmware to the MCU.
2. Open Serial monitor (hotkey: Ctrl + Shift + M) to see the debug messages.
3. Done.

**\* Note**: There is a blue LED light indicator to know if the Esp32’s connection status:

- Periodically (1s) blink: trying to connect a Wi-Fi.

- Light on: connected to a Wi-Fi and trying to connect the backend (WebSocket server).

- Light off: connected to the backend.

## 4.4. Frontend

It’s recommended to run frontend on the same device as the backend for easier setup, or else, you will have to change to URL from at line 92 in **/frontend/pages/index.tsx** to the IP of the backend’s device.

Follow these instructions to run the frontend:

1. Go to **/frontend** folder.
2. Run **npm i**
3. Run **npm run dev**
4. Visit localhost:3000 to see the dashboard.
5. Done.

# 5. Deploy to the Internet

Heroku is the selected platform for deploying the system onto due to its ease-of-use. Please ensure that you have the heroku-cli installed, if not, refer to [this tutorial](https://devcenter.heroku.com/articles/heroku-cli).

It is advised to deploy the system following this order:

1. Backend
2. Frontend

## 5.1. Backend

The backend is simple, follow these steps:

1. Go to the project’s root folder.
2. Create a new Heroku app: **heroku create c4s-iot-backend**
3. Create Heroku Postgres database:
   1. Create a PostgreSQL database instance: **heroku addons:create heroku-postgresql:hobby-dev**
   2. View database credentials: **heroku pg:credentials:url -a c4s-iot-backend**

Text

Description automatically generated

* 1. Using these credentials, you can access the PostgreSQL using database management tools, specifically: DataGrip.
  2. After having access to the database, you need to execute the script from **/bin/database\_schema.sql** to create tables.

1. Add Heroku git remote: **heroku git:remote -a c4s-iot-backend**
2. Deploy the app: **git subtree push --prefix backend heroku master**
3. (optional) View server’s output log (for debugging): **heroku logs –tail**
4. Done.

After you deployed the server, take note of the server’s URL. It will be necessary for putting into the Esp32’s firmware and frontend’s web-socket URL.

### Update Esp32’s firmware

Update the web-socket URL for Esp32’s firmware **at line 13 in remote.cpp** file.



### Update frontend

Update the web-socket URL for frontend **at line 92 in pages/index.tsx** file.



# 5.2. Frontend

Steps for deploying the frontend are similar to the backend:

1. Go to the project’s root folder.
2. Create a new Heroku app: **heroku create c4s-iot-frontend**
3. Add Heroku git remote: **heroku git:remote -a c4s-iot-frontend**
4. Deploy the app: **git subtree push --prefix frontend heroku master**
5. (optional) View server’s output log (for debugging): **heroku logs –tail**
6. Done.

# 6. Limitations

## 6.1. Free Heroku’s PostgreSQL database only allow up to 10,000 rows

([reference document](https://elements.heroku.com/addons/heroku-postgresql#:~:text=Direct%20SQL%20access-,row%20limit%2010%2C000,-Storage%20Capacity%0A%0A%20%20%20%201))

Since we are using a free database, there is a limitation on the total number of rows in the database at 10,000. Knowing that send data from the Esp32 to the backend every second. Hence, in the worst-case scenario, the longest duration the system can run without exceeding the limit is: 10,000 seconds ~ 2 hours and 46 minutes.

**Solution:**

1. Upgrade to a paid subscription.

2. Increase the time where Esp32 send data to the backend.