

Comprehensive air quality monitoring and notification system

CM3040 Physical Computing and and Internet of Things

Enrique Condes Brena

Abstract

Air pollution is an everyday problem that affects billions of people worldwide causing countless deaths every year. Using a comprehensive domestic system to monitor internal and external air quality and alerts about unhealthy conditions, users can quickly and painlessly get information that allows them to take proper actions of preventive nature to minimize undesirable situations.

Contents

List of Figures	1
List of Tables	1
1 Introduction	2
2 Major components	2
2.1 Main node	2
2.2 Sensor node	2
3 Implementation	3
4 Analysis	3
5 Conclusion	4
6 Appendices	5
6.1 Development log	5
6.2 Code compilation	5
6.3 Deployment	6
6.4 Bill of materials	7
6.4.1 Main node	7
6.4.2 Sensor node	7
6.5 Circuitry	8
6.6 Images	9

List of Figures

1 Development version control system log	5
2 Main node code compilation	5
3 Sensor node code compilation	6
4 Main node code deployment	6
5 Sensor node code deployment	6
6 Main node connection diagram	8
7 Sense node connection diagram	9
8 Led matrix information diagram	9
9 Dashboard sketch	10
10 Dashboard sketch	10
11 Main node dashboard	11
12 Sensor node dashboard	11

List of Tables

1 Main node bill of materials	7
2 Sensor node bill of materials	7

1 Introduction

This project creates a modular IoT system for monitoring and generating a comprehensive air quality analysis combining big scale (regional/metropolitan air quality) and micro scale perspectives(domestic spaces). Combining these sources, the system displays easy to interpret notifications that allow the user to make informed decisions regarding his health and safety when performing daily activities.

2 Major components

Air pollution is an everyday problem that affects billions of people worldwide causing countless health problems on a global scale. Information and a proactive stance are within reach, but a comprehensive yet simple to interpret system does not exist. The system built to tackle this problem is comprised of two different types of nodes. Each type of node focuses on detecting and measuring different but complimentary air pollutants to generate a detailed picture of the conditions. If required, both types of nodes can operate independently or offline only loosing some functionality. The two node types are referred to as **main node** and **sensor node**. Every complete system requires one **main node** and an hypothetically infinite number of **sensor nodes**. For this first revision of the system, the number of **sensor nodes** is limited to three units.

2.1 Main node

This node acts as the most visible face of the system towards the user. Because of this visibility, it was conceived as a device with a fixed position, plugged into an external power supply. This allows assigning the most power-intensive functions of the system to it without compromising its functionality. The three power-intensive tasks it performs are measuring local Particulate Matter concentrations, fetching and processing information from all the network sensors, and permanently displaying the status of the whole system for the user to see. Besides that, it configures the network connectivity for all the **sensor nodes**, fetches air quality measurements from remote sources, usually consisting of official environmental agencies, and measures temperature and humidity. In case of a network failure, this kind of node is the most severely affected since it loses the ability to fetch information from **sensor nodes** and external data services providing metropolitan air quality readings.

2.2 Sensor node

This type of nodes is in charge of monitoring the levels of CO_2 and Volatile Organic Compounds, two types of pollutants that quickly produce adverse reactions when present on high concentrations. Besides that, it also measures and reports the temperature and humidity of its surrounding environment. Because of the danger implied by high concentrations of CO_2 , each sensor node also displays area notifications using an RGB led. It is conceived as a small, battery operated node, that can be placed anywhere and whose presence will most likely be forgotten except when an alert is displayed. This type of node is not affected by operating independently or offline because of the system architecture. It can still perform local air quality readings and display alerts if necessary.

3 Implementation

As shown on image 1, an agile methodology was used for developing the system. Each of the implemented features was developed and tested before starting the development of the next. Features related to the general operation of a device were developed before features providing specific functionality. All features related to communications between the different devices were the last to be developed since they rely on "board-specific" functionality. For some cases, a sandbox folder was used to develop and optimize some functionality before integrating it into the system.

Code was split into several, function specific, .ino files to simplify development, debugging, and maintenance which left the main files (**main.ino** and **sense.ino**) only with the **setup** and **loop** functions alongside with objects creation and global variables. Functions related to data gathering and processing were wrapped into .h and .cpp files. This encapsulates the low-level functionality of configuring and interacting with the physical sensors, the use of other sensors with similar functionality would not require major rewrites to the project code, but only the writing of a new wrapper using the specific sensor API.

For developing the dashboard for each of the devices, the templating function included on the **ESPAsyncWebServer** library was used <https://github.com/me-no-dev/ESPAsyncWebServer#template-processing>. Locations that get replaced use placeholders enclosed by the % symbol. Since this library only provides limited functionality in terms of replacing the placeholders with static values, templating requiring loops and logic was implemented by generating on-the-fly HTML structures within the templating file(**templating.ino**).

Because of this, instead of hardcoding the source code for the web pages into the sketch, it is stored into a file system created on the esp8266 memory. Using this technique allows easier development of the code as the files can be executed on a computer for debugging purposes before being uploaded to the microcontroller. This is accomplished by using **ESP8266 LittleFS Filesystem Uploader**, an additional tool that can be installed on the Arduino IDE. Instructions for installing it can be found at <https://randomnerdtutorials.com/install-esp8266-nodemcu-littlefs-arduino/>. To optimized memory usage and network transfer speed, the code is minified before programming using several external tools. The required tools are *html-minifier*, *cssnano*, and *uglifyjs*, and the complete process for setting it up can be found here <https://codeburst.io/how-to-minify-your-static-site-d90565c9aa21>. For the same reason, network credentials are also stored into a JSON file on the filesystem instead of being defined within the sketch code.

To guarantee the existence of coherent memory states between the sketch and the filesystem along with automatizing the minification of the filesystem, I created a script to handle it. The script compiles the sketch, uploads it to the board, minifies the filesystem, and flashes it to the board. This can be performed directly into a plugged device or remotely using OTA updates.

To conform with current scientific standards regarding the acceptable pollutant's levels, how to calculate them, and how they are displayed, I used the EPA's assistance document on the subject <https://www.airnow.gov/sites/default/files/2020-05/aqi-technical-assistance-document-sept2018.pdf>. This information is used for categorizing the level of danger and displaying it properly on the led matrix and the dashboards.

4 Analysis

One obvious problem with the current system is the fact that it trusts blindly on the data it receives. The DHT11 sensor used on both types of devices is known for providing unreliable readings, $\pm 2^\circ C$ and $\pm 5\%$ relative humidity, according to the datasheet, which can be confirmed by comparing the data generated by two sensors located one besides the other. This problematic

could be reduced by using more accurate sensors, but this would probably cause an increase to the system's price. Since temperature and humidity readings are not part of the most crucial functionality of the system, they do not hinder significantly the main functionality of the whole system, so it is considered an acceptable compromise. In the case of the CCS811 sensor, incoherences between sensor readings of CO_2 and Volatile Organic Compounds are acceptable since they can be regarded as regions with irregular concentrations caused by activity or air flow.

One functionality that could provide some added value to the dashboard would be the ability of displaying temperature using Fahrenheit. Although implementing this functionality is simple in terms of low-level implementation with the sensors, the real challenge is in guaranteeing coherent and uniform temperature units along the whole system. Implementing this would require something analogous to MQTT which would be excessive for something so trivial as broadcasting which temperature unit to use.

There is a clear and clashing mismatch between the notification paradigms used by both node types. While the **main node** will be displaying notifications all of the time, the **sensor node** will only show an alert in case of a dangerous or unhealthy situation. This creates a situation of "providing good and bad news equally" versus "only providing bad news" in which both have their pros and cons. An approach in which all situations are notified provides certainty that the message is properly transmitted and that if there is no message, something is wrong with the system, but can be considered inefficient in terms of waste of resources. The opposite approach is more efficient energetically speaking but has the big inconvenience that a lack of notification could produce a "confirmation bias" on which the user interprets a lack of alert as the equivalent of a safe condition which could not always be true, e.g. when the device's battery is exhausted.

The previously mentioned approach of only notifying adverse conditions could be used for creating a simplified version of the system on which the led matrix and Particulate Matter are eliminated and all of the system nodes become enhanced **sensor nodes**. These modified nodes could perform some of the functions of a **main node** like fetching metropolitan air quality data and operate independently or in a mesh network configuration while maintaining a small form factor.

Although it is effective at conveying information for an experienced user, the current method for displaying the state of the system is too simplified and does not follows a logic that is obvious for a first time user, specially information regarding temperature and humidity as shown in image 8. There are several approaches that could be adopted for tackling this problem that include using a face plate for the device and clear explanations on the system manual, but those could be considered palliatives. Rethinking the way data is displayed by involving end-users into a user-centered design strategy would be the best solution.

Even though it performs its task correctly, the current methods for adding and removing new sensor nodes need rethinking. The adding method does not check the identity of the device asking for the credentials so they could be shared with malicious devices. Besides that, the data is transmitted as clear text which is insecure. For removing a sensor node, the main device sends a request to the node for it to erase the network credentials and waits for confirmation. Once the confirmation arrives, it removes the device from the list of registered devices. Following this logic, if a node is broken or lost, it will never acknowledge the credentials erasure; therefore it will never be removed from the list on the main device.

5 Conclusion

In its current state, the system can be considered mature although there are still certain areas that are problematic and require additional work before it can be considered a finished product.

Also, certain aspects can still be explored in order to enhance the functionality and general experience or give a new direction to the project.

6 Appendices

6.1 Development log

383c3ca	- (HEAD -> master) Documentation (36 minutes ago) <Enrique Condes>	
975700a	- Treat values of CO2 == 0 as sensor errors (3 hours ago) <Enrique Condes>	
fde803d	- Double sampling rate in high concentrations of co2 (3 hours ago) <Enrique Condes>	
803e00f	- Set up node responsiveness (1 hour ago) <Enrique Condes>	
14fe0d9	- CO2 sensor wrapper (22 hours ago) <Enrique Condes>	
adaf10c	- PM2 sensor wrapper (22 hours ago) <Enrique Condes>	
001e04d	- Neomatix wrapper (22 hours ago) <Enrique Condes>	
a51b7a5	- Move scheduled tasks to their own file (22 hours ago) <Enrique Condes>	
b348351	- (cppfiles) Display maximum temp/humidity on Neomatrix (4 days ago) <Enrique Condes>	
70300ed	- Clean up remote device alias (4 days ago) <Enrique Condes>	
d5bb5e1	- Display icon with max temperature (4 days ago) <Enrique Condes>	
11ae301	- Send remote device alias with data (10 days ago) <Enrique Condes>	
d5fb0f8	- Don't overwrite nodes file on OTA (10 days ago) <Enrique Condes>	
ff3e7db	- Move nodes functions to their own module (10 days ago) <Enrique Condes>	
ff16043	- Dynamically generates nodes data containers (10 days ago) <Enrique Condes>	
70300ed	- Register nodes via the main device dashboard (10 days ago) <Enrique Condes>	
001e04d	- Register nodes via the main device dashboard (10 days ago) <Enrique Condes>	
0773c13	- Minor fixes (10 days ago) <Enrique Condes>	
2747a20	- Optimize remote node reading (10 days ago) <Enrique Condes>	
0476050	- Send network data to remote sensors (10 days ago) <Enrique Condes>	
1915529	- Change main device alias (10 days ago) <Enrique Condes>	
70300ed	- Add remote device alias file (10 days ago) <Enrique Condes>	
5fa1f51	- Set value for remote sensor (11 days ago) <Enrique Condes>	
00f0848	- Blink red led if CO2 levels is too high (12 days ago) <Enrique Condes>	
70ef100	- Define leds (12 days ago) <Enrique Condes>	
39e8b0d	- Display colors according to values (12 days ago) <Enrique Condes>	
47407eb	- Add API for sensors (12 days ago) <Enrique Condes>	
451000d	- Display icons size (12 days ago) <Enrique Condes>	
374706e	- Link task name to device (12 days ago) <Enrique Condes>	
c279170	- Compensate environment (17 days ago) <Enrique Condes>	
078344d	- Rush first CO2 readings until sensor stabilizes (12 days ago) <Enrique Condes>	
50191e6	- Add DHT sensor (12 days ago) <Enrique Condes>	
41a4c0b	- Add co2 sensor (12 days ago) <Enrique Condes>	
00e6971	- Use neomatix for displaying (13 days ago) <Enrique Condes>	
5131154	- Fix display for Neomatix (13 days ago) <Enrique Condes>	
70112f4	- Disable server and tasks when updating via OTA (2 weeks ago) <Enrique Condes>	
92f0c6f	- Reduce style footprint (2 weeks ago) <Enrique Condes>	
eb89ed4	- Display AIQ colors according to values (2 weeks ago) <Enrique Condes>	
41620ab	- Responsive dashboard (2 weeks ago) <Enrique Condes>	
20d600b	- Show internal AQI value (2 weeks ago) <Enrique Condes>	
34929c2	- Call API from pm25 value (2 weeks ago) <Enrique Condes>	
413d10b	- Add dust sensor (2 weeks ago) <Enrique Condes>	
963d3ab	- Network upload FM and FS (2 weeks ago) <Enrique Condes>	
00d0080	- Add temperature and humidity readings (2 weeks ago) <Enrique Condes>	
754cabb	- Add dashboard icons (2 weeks ago) <Enrique Condes>	
9774ac0	- Read display WAOI values (2 weeks ago) <Enrique Condes>	
50191e6	- Use WPA network to transmit the new sensor (4 weeks ago) <Enrique Condes>	
3008039	- Turn off AP after pairing new sensor (4 weeks ago) <Enrique Condes>	
d54494d	- Rapid blink led on sense when on pair mode (4 weeks ago) <Enrique Condes>	
dd90e0f	- Sense fetches network config from main device (4 weeks ago) <Enrique Condes>	
4415722	- Pass number of attempts to network connection func (4 weeks ago) <Enrique Condes>	
b0b1019	- Basic sense structure (4 weeks ago) <Enrique Condes>	
958b7cc	- Implement API (2 weeks ago) <Enrique Condes>	
78112f4	- Disable server and tasks when updating via OTA (2 weeks ago) <Enrique Condes>	
92fc0fd	- Reduce style footprint (2 weeks ago) <Enrique Condes>	
4161a0b	- Responsive dashboard according to values (2 weeks ago) <Enrique Condes>	
24dd0b0	- Show internal AQI value (2 weeks ago) <Enrique Condes>	
a3e92c7	- Calculate AQI from pm25 value (2 weeks ago) <Enrique Condes>	
4f24940	- Add dust sensor (2 weeks ago) <Enrique Condes>	
431d0b2	- Duplicate icons size (2 weeks ago) <Enrique Condes>	
993a50b	- Network upload from FS (2 weeks ago) <Enrique Condes>	
754cabb	- Add temperature and humidity readings (2 weeks ago) <Enrique Condes>	
9f74ac6	- Add dashboard icons (2 weeks ago) <Enrique Condes>	
bea6622	- Use a WPA network to transmit the network data (3 weeks ago) <Enrique Condes>	
3008039	- Turn off AP after pairing new sensor (4 weeks ago) <Enrique Condes>	
d54494d	- Rapid blink led on sense when on pair mode (4 weeks ago) <Enrique Condes>	
dd90e0f	- Sense fetches network config from main device (4 weeks ago) <Enrique Condes>	
4415722	- Pass number of attempts to network connection func (4 weeks ago) <Enrique Condes>	
bbb1d15	- Basic sense structure (4 weeks ago) <Enrique Condes>	
2160a18	- Main device code for sharing network credentials (4 weeks ago) <Enrique Condes>	
2c78009	- Change hostname and redirect (4 weeks ago) <Enrique Condes>	
03d500d	- Retrieve hostname from FS (4 weeks ago) <Enrique Condes>	
6014000	- Network upload from FS (4 weeks ago) <Enrique Condes>	
6049f43	- Avoid repeated connection attempt (4 weeks ago) <Enrique Condes>	
9487909	- Enable network detection by name (5 weeks ago) <Enrique Condes>	
e477f06	- Enable OTA updates (5 weeks ago) <Enrique Condes>	
4520b54	- Set device name according to chip id (5 weeks ago) <Enrique Condes>	
de0a959	- Network status is hidden at boot (5 weeks ago) <Enrique Condes>	
5910e49	- Avoid network connection when power button is pressed (5 weeks ago) <Enrique Condes>	
3748940	- Check connection status via REST api (5 weeks ago) <Enrique Condes>	
07e0303	- Refresh networks on backend via API (5 weeks ago) <Enrique Condes>	
9750e04	- Set available networks as a JSON object (5 weeks ago) <Enrique Condes>	
81e146c	- Proper way to store wifi password (5 weeks ago) <Enrique Condes>	
c2aefc0	- Try to connect with received credentials (5 weeks ago) <Enrique Condes>	
5313163	- Fix connection stored in memory (5 weeks ago) <Enrique Condes>	
040d466	- Factorize wifi connection code (5 weeks ago) <Enrique Condes>	
c28fa9c	- Enforce coherence between FS size and address (5 weeks ago) <Enrique Condes>	
2353c3c	- Refactor module code to clean main file (5 weeks ago) <Enrique Condes>	
50891b1	- Reboot device after receiving wifi credentials (5 weeks ago) <Enrique Condes>	
cd1fb45	- Allow toggling password visibility (5 weeks ago) <Enrique Condes>	
401348c	- Add image cropping and generating script (5 weeks ago) <Enrique Condes>	
204964c	- Add image cropping and generating script (5 weeks ago) <Enrique Condes>	
06b1363	- Fix bug in build script (5 weeks ago) <Enrique Condes>	
dd5d50f	- Enable support for css and js on AP (5 weeks ago) <Enrique Condes>	
f5dc39d	- Condition AP init to unsuccessful STA connection (5 weeks ago) <Enrique Condes>	
ebe7681	- Receive and store WiFi settings (5 weeks ago) <Enrique Condes>	
73fbaf0	- Use template to show available networks on AP (5 weeks ago) <Enrique Condes>	
f5bf7fd	- Configure device as AP (5 weeks ago) <Enrique Condes>	
255eaef	- Add auto minification to build script (5 weeks ago) <Enrique Condes>	
1933841	- FS minifying script (5 weeks ago) <Enrique Condes>	
05f5393	- Comprehensive build script (5 weeks ago) <Enrique Condes>	
5395c6f	- Load Wifi config from external fs (5 weeks ago) <Enrique Condes>	

Figure 1: Development version control system log

6.2 Code compilation

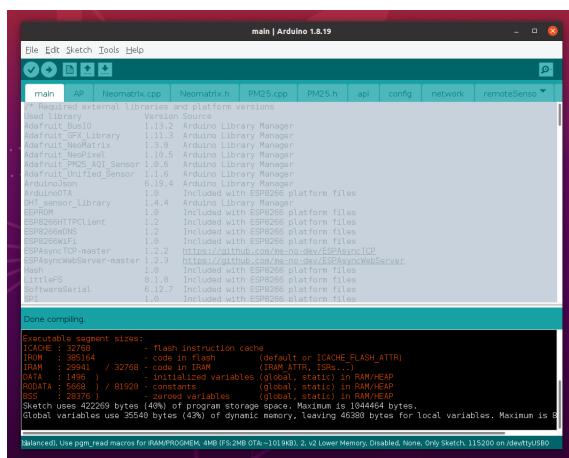


Figure 2: Main node code compilation

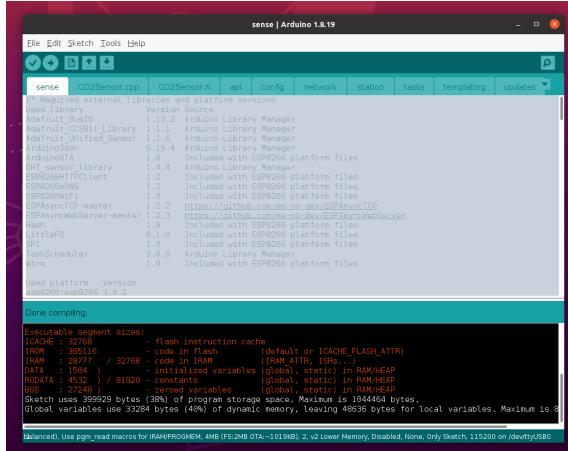


Figure 3: Sensor node code compilation

6.3 Deployment

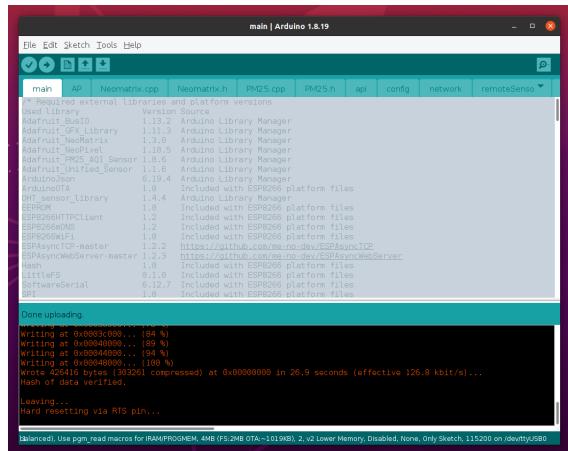


Figure 4: Main node code deployment

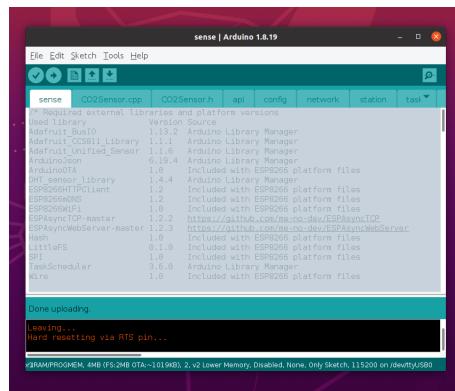


Figure 5: Sensor node code deployment

6.4 Bill of materials

6.4.1 Main node

Component	Amount
NodeMCUv1.0(ESP-12E module)	1
Plantronics PMS5003 air quality sensor	1
NeoMatrix (WS2812 led 8x8 board)	1
330Ω resistor	1
DHT11 sensor	1
10kΩ resistor	1*

*This is not always required as some breakout boards already include it

Table 1: Main node bill of materials

6.4.2 Sensor node

Component	Amount
NodeMCUv1.0(ESP-12E module)	1
CCS811 CO2 sensor	1
RGB LED	1*
220Ω resistor	3*
DHT11 sensor	1
10kΩ resistor	1 ⁺

*This is not always required as some development boards already include it

⁺This is not always required as some breakout boards already include it

Table 2: Sensor node bill of materials

6.5 Circuitry

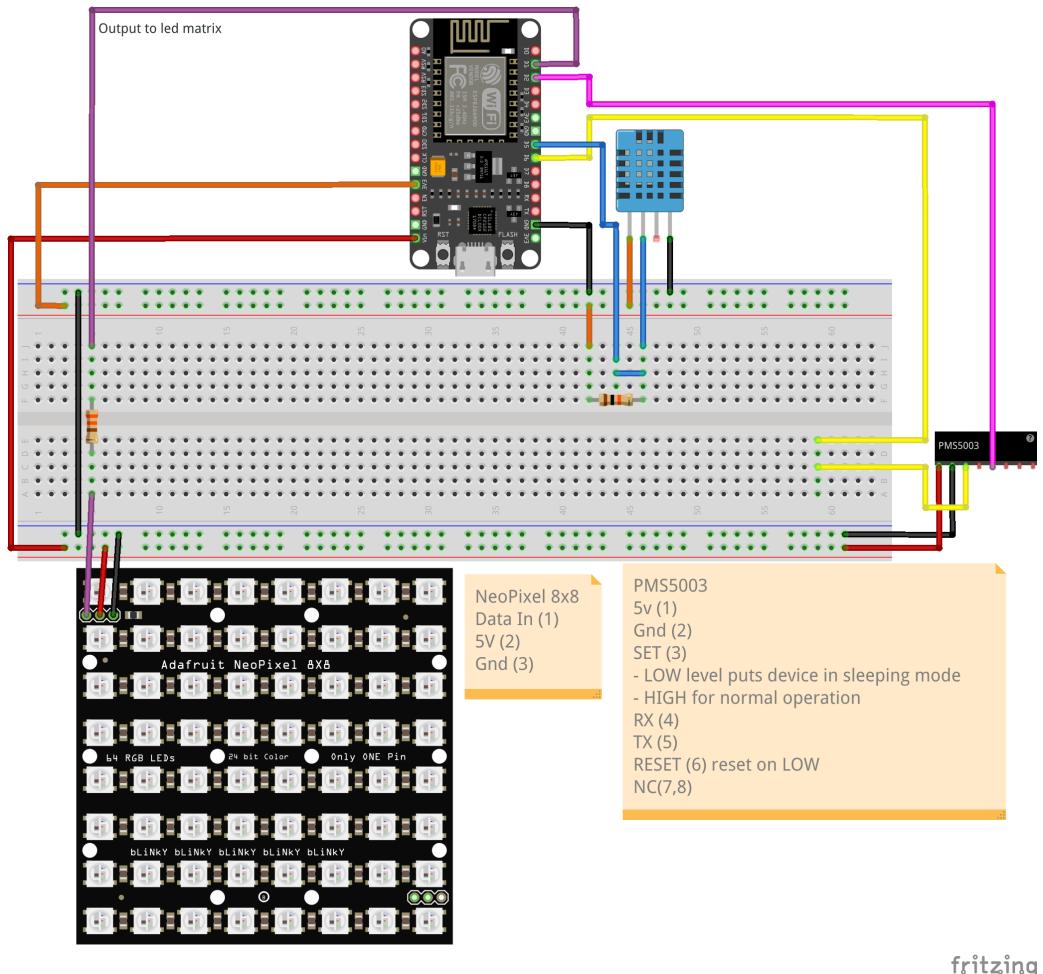


Figure 6: Main node connection diagram

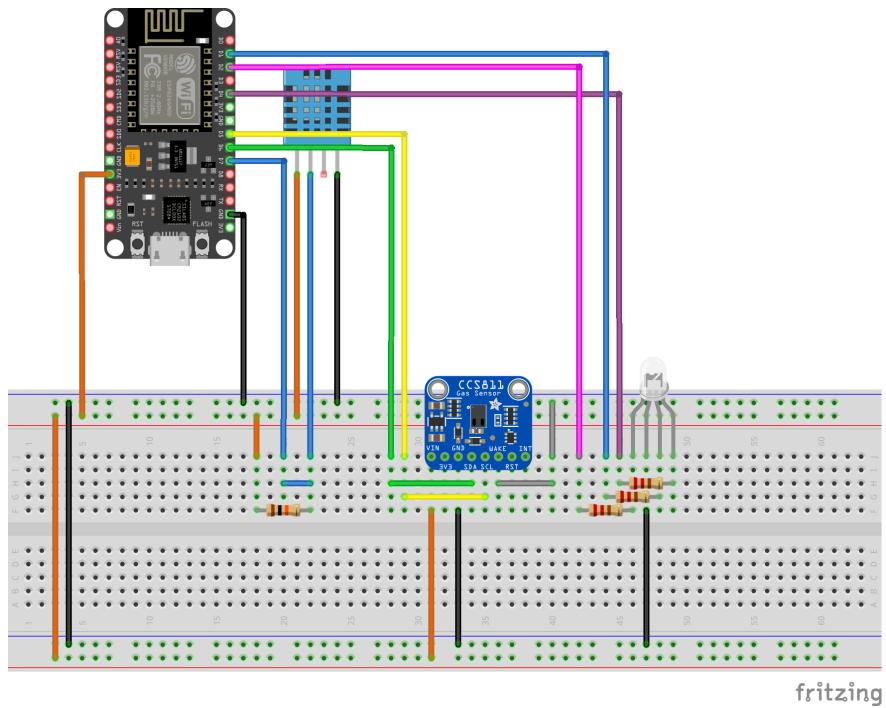


Figure 7: Sense node connection diagram

6.6 Images

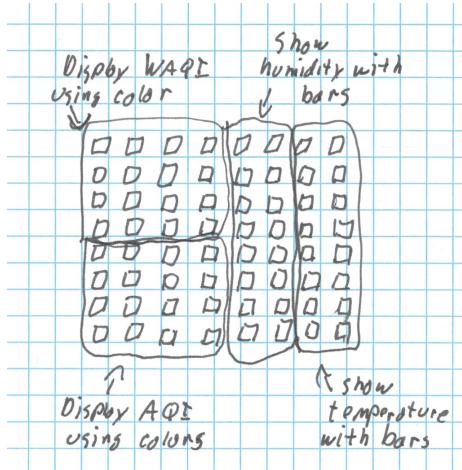


Figure 8: Led matrix information diagram

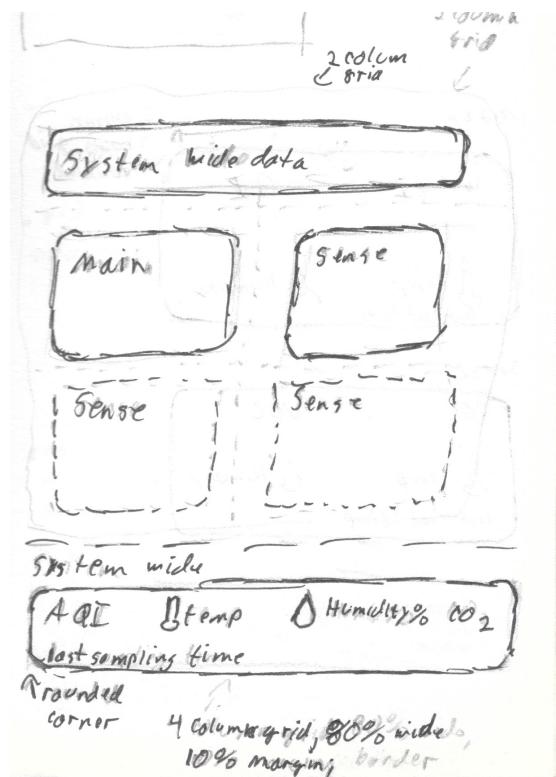


Figure 9: Dashboard sketch

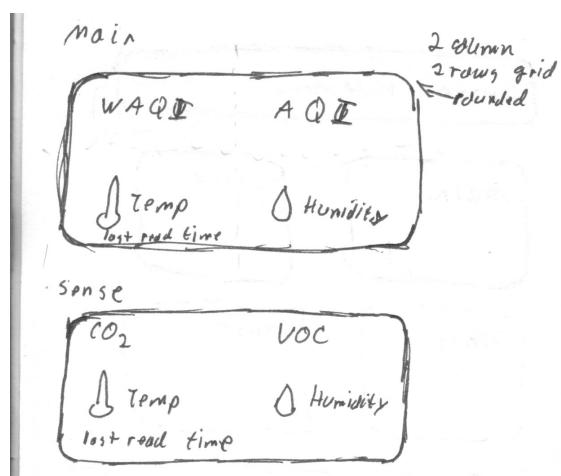


Figure 10: Dashboard sketch

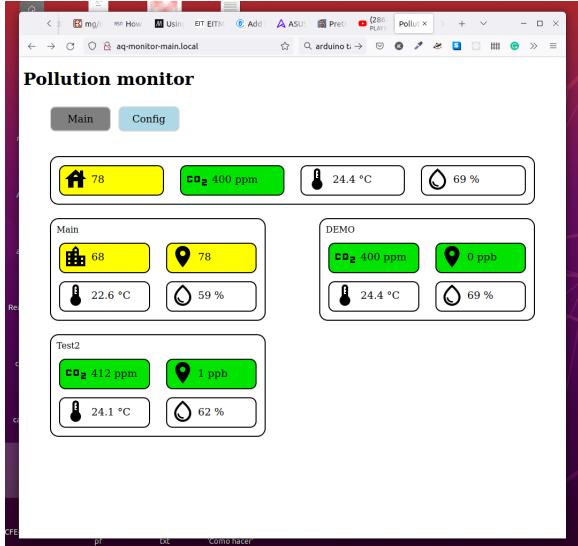


Figure 11: Main node dashboard

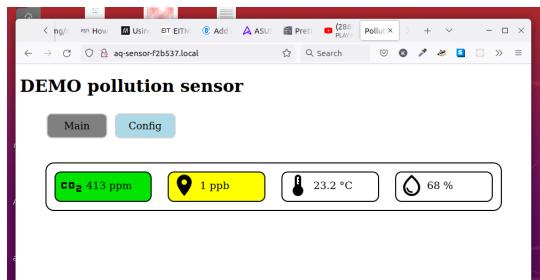


Figure 12: Sensor node dashboard