Air quality monitoring with an IoT device in hospital environments.

Author: Pol Méndez Heredero.
Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain.[*]

Advisor: Dr. Manel Puig i Vidal

Abstract: In this project an Arduino IoT based board connected to some sensors, is used to control the temperature, the humidity and the concentration of CO_2 and combustible gases. The device will be installed at the top of an autonomous moving robot that will be in clean hospital rooms. The user will be able to consult the data via smartphone or web.

I. INTRODUCTION

The air quality of a hospital has to be under control, moreover if we appoint clean rooms, like labs or surgeries. The main point of this project is to set up an IoT device carried by an autonomous robot that monitors air parameters like temperature, humidity, smoke, CO_2, gas and less important gases, and being able to check the data through an app or a web. Here, we will only talk about the device, the final result and the difficulties that we faced, without considering the robot. IoT stands for Internet of things, so, an IoT device is able to connect to a WiFi network to receive and send data remotely. The boards chosen for this project are the MKR1010 and Nano 33 IoT, from *Arduino*, and the ESP32 from *Espressif*. The software needed is the web from Arduino Cloud, where you can code and set up the network, the Arduino Create Agent and a few drivers. In the following sections we will go through how this project started, a bit of explanation about the sensors, the final result and possible improvements.

II. CONCEPTUAL ENGINEERING

It is necessary to study the different options for every magnitude that will be measured and which type of board is better to use. We will divide the sensors in three categories, CO_2, temperature with humidity and other gases. We will only consider those ones that are compatible with Arduino.

A. CO_2

The top 3 "low-cost" CO_2 sensors are the CCS811, the MG811 and the MQ135. The price of CCS811 and the MQ135 is between 2 and 5 €, both are way cheaper than the MG811, with a cost of 20 €. We discard the MG811, because we want an accurate device but at a reasonable price. Then, the MQ135 is a good option but it detects lot of different gases, not only CO_2, so this may cause troubles when it comes the time to stabilize the levels. To sum up, we choose the CCS811 because is cheap and it gives two different values, one for CO_2 and one for TVOC, we will see later what it means.

B. Temperature and Humidity

Here we have another time the CCS811, but we eliminate it quickly because it only reads temperature and it doesn't work very well. So we have two more options, the DHT22 and the BME280. The BME280 is great, because it also detects the pressure and it's a bit more accurate than DHT22 and has better resolution but the pins needs to be welded and we had difficulties to get good electrical contacts between the pins. Therefore, we stayed with DHT22 although being a worse choice.

C. Other gases

In order to prepare a better device, we implemented an extra sensor, that detects less important gases or less frequent gases in a hospital. This category has a wide range of options, but the "MQ" family sensors fit quite good. In particular, the MQ-2, for being able to detect smoke and LPG that might be important too.

D. Boards

A mandatory feature of a board for this project is the IoT technology. As we said before, we use the MKR1010 [1] and the Nano33 IoT [2] because they are easy to use and the code is pretty basic. The ESP32 [3], in contrast, is less accessible in coding terms, but it is stronger for external utilities and has a security key to modify the code. These three boards are widely used in all kind of tasks.

III. DETAIL ENGINEERING

In this section we will go a bit deeper in the chosen sensors and look at their specs, in the References section,

[*]Electronic address: `pmendehe7@alumnes.ub.edu`

you will find attachments in case that someone wants to know more about any one.

A. CCS811

The CCS811 [4] is an ultra-low power digital gas sensor solution which integrates a metal oxide (MOX) gas sensor to detect a wide range of Volatile Organic Compounds (VOCs) using an I2C communication protocol, that simplifies the hardware and software integration. It is based on *ams* unique micro-hotplate technology which enables a highly reliable solution for gas sensors, fast cycle times and a significant reduction in average power consumption. CCS811 supports intelligent algorithms to process raw sensor measurements to output a TVOC value or equivalent CO_2 (eCO_2) levels, where the main cause of VOCs is from humans. VOC are responsible for the odor of scents and perfumes as well as pollutants, and plays an important role for animals and plants communication. Not all VOCs are dangerous, but it is better to control them, especially the ones that come from paints, industrial glues and cleaning products which contain risky substances. The equivalent CO_2 output range is from 400 to 8192 parts per million (ppm), with a resolution of 1 ppm. The Total Volatile Organic Compound (TVOC) output range for CCS811 is from 0 to 1187 parts per billion (ppb), with a resolution of 1 ppb. Values outside this range are clipped in both magnitudes.

As you can see in figure 1, CCS811 has 7 pins of which 5 are used in this project, the power pins, 3.3V and GND, the data pins, SDA and SCL, for the values measured and the clock respectively, and the WAK, WakeUp pin, that starts the sensor when it is connected to ground. Once it is started, it should run for 20 minutes before accurate readings are generated.

If we look now at the software requirements, we will need the "SparkFunCCS811" library.

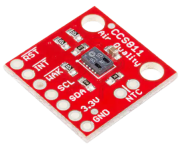


FIG. 1: CCS811

B. DHT22

The DHT22 [5] also named as AM2303, integrates a capacitive humidity sensor and a thermistor to measure the surrounding air, and sends the data via a digital signal. It is widely used for its toughness and simplicity. It only has three pins, from left to right, VCC, data and GND, as you can see in figure 2.

On the one hand, the operating range for the temperature is from -40 to 80 Celsius degrees. The accuracy is about half a degree, with a resolution of 0.1 degrees.

On the other hand, the relative humidity has a range from 0 to 100 % RH. With an accuracy of 2 %RH and a sensitivity of 0.1%RH. The DHT22 can only read different values every 2 seconds from the last measure.

In the software demands, we have to install two libraries, "DHT.h" and "DHT_U.h" [6].

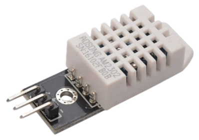


FIG. 2: DHT22

C. MQ-2

The last sensor that we deal with is the MQ-2, displayed in figure 3. MQ-2 detects combustible gases as LPG, i-butane, propane, methane, alcohol, Hydrogen and it also is sensible to smoke [7]. It is composed of an AI_2O_3 ceramic micro-tube with a sensitive layer of SnO_2, the measurement electrode and the heater are fixed in a shell made of plastic and stainless steel mesh. The heater provides the necessary working conditions for the work of sensitive components.

The MQ-2 doesn't give separated data for every gas that is sensible to. It only gives a concentration based on the voltage transmitted by the electrode. The range varies for every gas as shown in the Table I.

TABLE I: Range in parts per million for every gas detected by MQ-2.

Gas	Range (ppm)
LPG & propane	200 - 5000
Butane	300 - 5000
Methane	5000 - 20000
H_2	300 - 5000
Alcohol	100 - 2000

In the specs of MQ-2 there is no mention for the resolution nor the sensitivity. It has four pins, from left to right, the two power pins, VCC and GND, and two outputs, the digital and the analog one. Finally, the MQ-2 doesn't demand any library in specific.

D. Arduino Cloud

The Arduino Cloud interface is quite clean and sleek, we will comment the distinct functionalities that the web

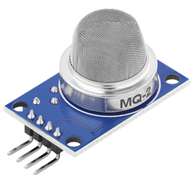
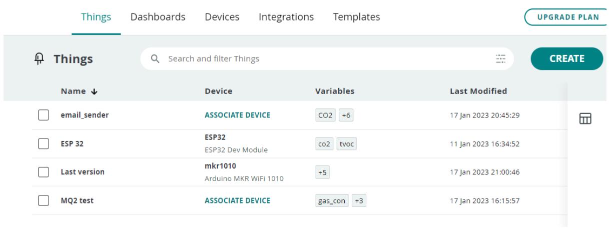


FIG. 3: MQ-2

offers, you can see how it looks in figure 4. Once we are in Arduino cloud, we have two options, coding in IoT Cloud or code in the Web Editor. The web editor is fully equipped with a library manager, allows you to add more files into a project or select advanced options of USB connections for the boards. The IoT cloud is faster and smoother. Arduino Cloud has a "Things" tab , where you declare all the variables for your project, set the network and type code.



	Name	Device	Variables	Last Modified
☐	email_sender	ASSOCIATE DEVICE	CO2 +6	17 Jan 2023 20:45:29
☐	ESP 32	ESP32 ESP32 Dev Module	co2 tvoc	11 Jan 2023 16:34:52
☐	Last version	mkr1010 Arduino MKR WiFi 1010	+5	17 Jan 2023 21:00:46
☐	MQ2 test	ASSOCIATE DEVICE	gas_con +3	17 Jan 2023 16:15:57

FIG. 4: Main view of Arduino Cloud. At the top, we appreciate the different tabs as Things or Dashboards.

Then, there is a useful tab called Dashboard. Dashboard provides you the tools to control in real time the measured values. You can set to see only the last value, to have a historic, with different time views as live, 1 week or 1 month. It is possible to set a message mini app, to communicate with the script or vice versa. The biggest advantage of Arduino Cloud and dashboards is that you can consult it via an app in your smartphone. When it is configured, it has two viewing modes, monitor dimensions, or a mobile screen size. In the next section, figure 7 shows the device's Dashboard. Another advantage from Arduino Cloud is the synchronisation of variables from different things, hence, from different boards as we will see later.

IV. FINAL RESULT

After all the details in the previous sections, it is time to see how the device works, what it offers to the user, and how it looks all assembled. I would like to attach a visual circuit diagram but any site has more than two components that are used. In figures 8 and 9 the entire device is shown from different viewpoints.

The device is equipped with an MKR WIFI 1010 board, figure 5, that controls the DHT22 through a digital pin and the MQ-2 by an analog one. In the other side of the breadboard we find an ESP32 board, figure 6, connected to the CCS811.

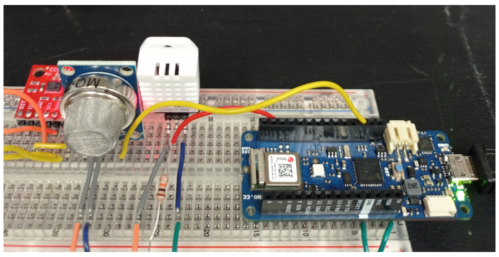


FIG. 5: Detail of MKR1010 with DHT22 and MQ-2

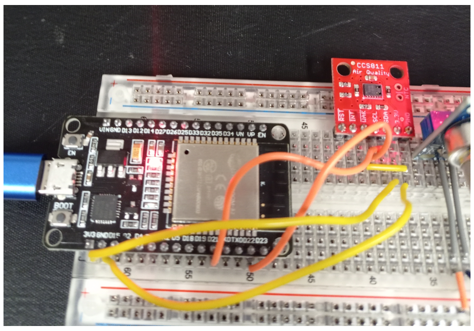


FIG. 6: Detail of the ESP32 and CCS811.

The device is set to take measures every 5 seconds in test mode, it is intended to increase this interval to 1 or 2 minutes, because the magnitudes quoted don't change fast, even so the interval could be increased a bit more. Taking advantage from the sync variables mode that offers Arduino Cloud, we send to the MKR1010 the CO_2 and VOCs values and then display them with the temperature, humidity and other gases in a unique Dashboard. It is also possible to set a Dashboard with variables from different boards without syncing them. Figure 7 shows how it looks in the monitor view mode.

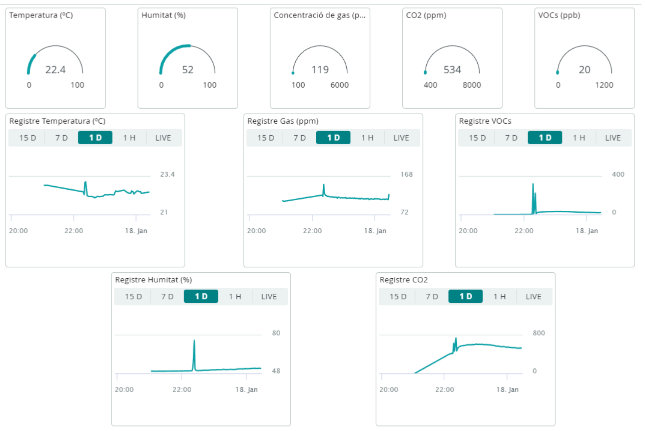


FIG. 7: Dashboard of our device. Real-time values are shown in gauges with a historic of the last 24 hours.

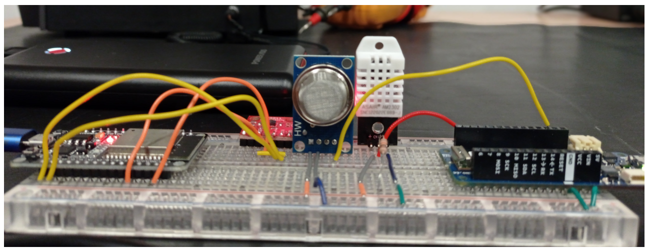


FIG. 8: Frontal view of the device

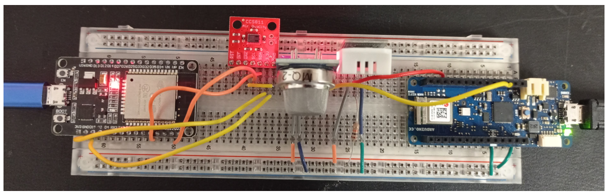


FIG. 9: Device seen from above

V. CONCLUSIONS

When the project started, the point was to prepare a device that measures the quality of the air, ideally with an only board, equipped with an automatic ozone blower to adjust the CO_2 levels, and all this integrated in a Printed Circuit Board, PCB, in order to add it on the autonomous robot. We found several issues that delayed us in preparing a better device, as boards' communication protocols, the stability of Arduino Cloud to code properly or obsolete libraries that did not work for the MKR1010 or the Nano33 IoT. There was a point when the CCS811 was dismissed because it did not work with any board whatever the code or the library. Finally, we tried everything until we added the ESP32, which gets along with CCS811[8].

However, I am glad where we got and what we achieved. Nowadays there is a working device with two boards communicating via WiFi.

VI. VIABLE UPGRADES

This device could be improved in several ways. First, it would be interesting and more efficient, having only one board, and it must be the ESP32 if it is important to have the CO_2 under control. In case of both boards stay, the MKR1010 can be replaced by the Nano 33 IoT, reducing the total weight. Second, as we said before, the device could be assembled in a PCB making it stronger and decrease the possibilities of bad electrical contacts. Adding an external holed case would help to protect the device too. Finally, it would be a huge improvement setting an automatic email sender function that whenever a value exceeds a threshold notifies the user to check the values in the Dashboard.

VII. APPENDIX

In Spain, the *Reglamento de instalaciones termicas, RITE* [9], fixes a few parameters of the air and divides the air quality in 4 categories called, *IDAs*. Hospitals are in IDA1, so whenever we set a threshold to trigger some action as sending an email, we will have to look at the maximum values accepted. There is also the *ISO 14644* regulation that determines air quality parameters for clean rooms [10].

Acknowledgments

I want to express my gratitude with my advisor, Dr. Manel Puig, who has guided and helped me through this project and supplied me all the equipment needed. Also, thanks to Dr. Manuel Carmona for getting in touch with Dr. Manel Puig and conceding me this opportunity that I learned many things.

[1] Arduino, *Mkr wifi 1010 datasheet*, URL `https://docs.arduino.cc/hardware/mkr-wifi-1010`.
[2] Arduino, *Nano 33 iot datasheet*, URL `https://docs.arduino.cc/hardware/nano-33-iot`.
[3] E. Systems, *Esp32 datasheet*, URL `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`.
[4] Sparkfun, *Ccs811 datasheet*, URL `https://drive.google.com/file/d/0BzaKjvCRihgbNUx4UmEycDFYWTg/view?resourcekey=0-3o_1F2g6-CqGiDu7Vkwukw`.
[5] A. E. Co, *Dht22 datasheet*, URL `https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf`.
[6] Adafruit, *Adafruit/dht-sensor-library: Arduino library for dht22*, URL `https://github.com/adafruit/DHT-sensor-library`.
[7] H. E. Co, *Mq-2 datasheet*, URL `https://www.mouser.com/datasheet/2/321/605-00008-MQ-2-Datasheet-370464.pdf`.
[8] esp32learning, *Esp32 and ccs811*, URL `http://www.esp32learning.com/code/esp32-and-ccs811-gas-sensor-example.php`.
[9] M. de la Presidencia, *Real decreto 1027/2007* (2007), URL `boe.es/buscar/doc.php?id=BOE-A-2007-15820`.
[10] A. UNE, *Iso 14644* (2016), URL `https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0057435`.