Intermediate Report: Tutored Project Third Semester

CO2 concentration notifier

TP01, TI:

BIZEAU Max
CLICHEROUX Shayne
VEYSSIERE Axel
ZERBIB Thomas

Table des matières

The goal of this intermediate project is to build a CO2 concentration notifier which will be placed in classrooms. This device is designed to notify the people in the classrooms if the air is healthy or not so they know when to open windows to ventilate the classroom to stay in a healthy environment. This is essential especially nowadays during the COVID19 pandemic as it can help prevent from transmitting the virus.

In order to achieve such a project, we had to buy some products, that is to say a CO2 sensor to collect data about the air of the room, an Arduino UNO to connect to the sensor which can also process the information and LEDs to notify to the user that it is time to open the windows.

1. The sensor

We used the intelligent gas module MH-Z16 to measure the quantity of Carbon Dioxide in the ambient air of a classroom.

This sensor allows us to measure CO2 with a fairly high resolution and with high sensitivity. Although the sensor is not the best in the market (the most popular CO2 sensor in the market seems to be the SCD30 which the other group worked with), it is pretty low-cost.

Thanks to research we made, we decided that we wanted the notifier to tell if the air was healthy if the CO2 concentration was below 1000PPM. Although, beyond this value, the device will notify the room that the air quality is degrading and, above 1 500PPM, we consider the air unhealthy and dangerous for health especially in these COVID times where air quality is paramount.

2. Designing the case

A case design was mandatory as soon as we decided its purpose was to be placed in the classrooms. Would it be to protect it from the dust or from any student trying to temper with the Arduino card. We needed a slim design with easy access to the card itself and an easy way for users to gather the information the card could give.

2.1. Using resources from the Coh@bit FabLab

We worked in collaboration with the Coh@bit FabLab Technoshop to cut the pieces in a wood medium after concertation with the team.

We had two options to create our case, using a 3D printer that would cost time and optimized design to avoid wasting plastic or cut it in wood and assemble the pieces after.

We chose to cut our case through wood as the medium is easier to manipulate and requires less optimization of the case design which meant a greater liberty of design and placement of the components. Moreover, cutting wood only takes up to two minutes whereas 3D printing takes more than a few hours so we could rectify problems in almost no time.

2.2. Leaving space for the LEDs

The eight LEDs had to be visible and as all the LEDs are aligned on the same piece of wire we had to think of where the LED strip could be visible at any angle.

As the LEDs are small squares we wanted to try and insert each led in a 5x5mm square (same size as the LEDs), however, there are small resistors as thick as the LEDs which prevent from fully inserting the LEDs in their squares. Moreover, as we didn't have much time to try and adjust the case with multiple prototypes and encountered issues with the space between each square, we ended up cutting a single rectangle in the wood to place the LEDs.

As the LEDs have to be easily visible, we decided to place them on the top panel of the box which is the most visible when place on a table for example, moreover, if the case is ever hanged on a wall, the top panel would end up being parallel to the wall and be also the most visible side of the case.

2.3. Where to put the long sensor 'tube'?

Choosing where to place the CO2 sensor was a tricky question and it generated great debate in our group. Should we put it in the case or outside the case? If we put it outside on which side should we put it?

As the datasheet mentioned that the sensor had to be well ventilated in small space, we decided that it would be better to leave it outside the case. The question was now choosing the side of the box to put it. We quickly came to the conclusion that placing it on the top panel would mean having the most stable box possible, thus we decided to place it on the top panel just as the LEDs.

2.4. Creating a place for the Arduino UNO and shield

Unlike the placement of the sensor, deciding where to put the Arduino card was very easy: inside the box, sitting on the bottom plate. The case would be a perfect protection for the shield, the Arduino board and for the cables. One thing was mandatory, as we didn't have enough time to make a prototype for a case that could have a simple opening system, we decided to entirely seal the case when it would be finished, this is why we needed to make holes for the Arduino input connections so that we could modify the program if needed and also calibrate regularly the sensor.

2.5. The cutting process

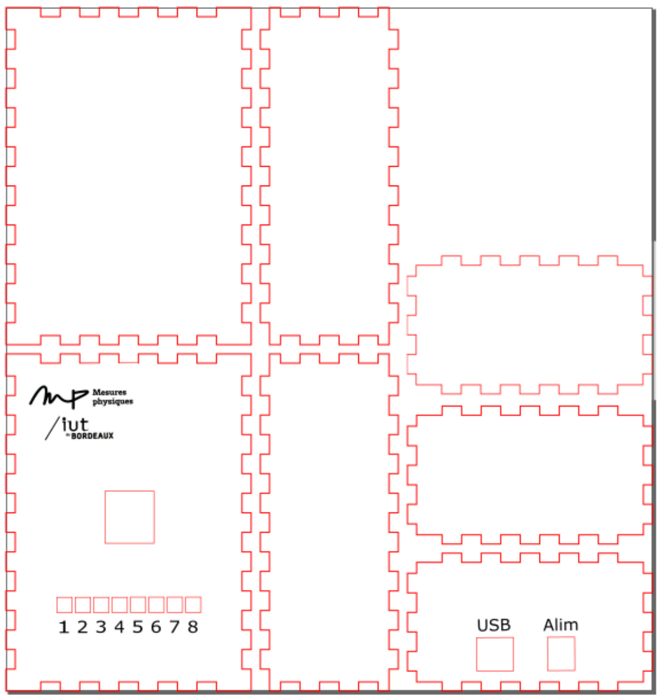

Figure 1 The first case design

Once we had an idea of how the case should be, we designed it on Inkscape, an open-source vectorial drawing software. To help us create the case we also used an online website which generated all the panels of the case.

Once the design was complete, we met at the FabLab to cut our pieces. The process was fairly easy as our vectorial drawing in an *.svg* format could be imported in the laser wood-cutting device software.

Once the wood was cut, we tested the design with the different components and some adjustments were needed: the alimentation and USB hole were misplaced, the sensor hole wasn't very optimal and the LEDs holes were misplaced too.

3. Developing the Arduino card program

One of the most crucial parts of making our device was to develop the program that the Arduino UNO card would run. In order to work, the program has to focus on three major steps: getting the values from the CO2 sensor, processing these values and then alert the user according to these said values with the help of eight LEDs.

3.1. Trying the CO2 sensor

We started focusing on retrieving the values from the sensor and processing them, in order to achieve this goal, we needed to study the datasheet of the sensor to understand how the is sensor working. Moreover, we also took a look at wiki.seeedstudio.com as they had sample code to calibrate and also read the CO2 concentration values.

The sensor works on 9 bytes long commands, the ninth bit being a checksum byte so that we can verify is the values sent by the sensor are not wrong. So, when we want the sensor to give us the values it retrieved, we have to send these 9 bytes: 0xff, 0x01, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79. 0xff corresponds to the starting byte, 0x01 is a reserved, 0x86 is the command that asks the sensor to give us its values, 0x00 are empty bytes and 0x79 is the checksum to make sure that the command is valid.

To send this command to the device and then check that the values returned were right, we used the sample code provided by Wikiseeed, however, their code was in fact far from great as most of the time the values were considered incorrect by the algorithm.

```
if((i != 9) || (1 + (0xFF ^ (byte)(data[1] + data[2] + data[3] + data[4] + data[5] + data[6] + data[7]))) != data[8])
  {
     return false;
  }
```

That means that this condition of the algorithm was almost always true so the values were in theory wrong. To check if the values were really wrong, we got rid of this condition and it turned out that the values were really odd so the problem was elsewhere. And so, we searched the Internet hopping to find a solution and we actually came upon a website (Grove – Carbon Dioxide Sensor (MH-Z16) trial | devMobile's blog) in which the person encountered the same problem as ours. Turns out the issue was the *Delay(10)* on line 59 which was supposed to work as a "flush", so we replaced that line by *Serial.flush()* and then everything worked just fine.

Another small issue we encountered was with these two lines:

```
sensor.begin(9600);
Serial.begin(115200);
```

Indeed, having two different Baud Rates can often be confusing and as only the sensor needed a specific rate of 9600, we could simply setup the rate of the Arduino board to 9600 also. This prevented encountering issues when using the Arduino IDE's Serial Monitor.

Now that our CO2 sensor seemed to work, we then calibrated it by placing it outside and letting it measure the CO2 concentration values for about an hour. After one hour we sent to the sensor the 9 bytes long command to calibrate it (for that we simply used the sample code provided by Wikiseeed as it was very simple and straightforward).

3.2. Trying the LEDs

After the calibration was done, we put away the sensor and tried to use the LEDs.

Unfortunately, once we connected the LEDs and sent to the Arduino UNO a sample code provided by Adafruit for the NeoPixel LEDs that we had, nothing happened. After a minute short investigation, we realized that the LEDs stick was soldered on the wrong side: the cables were soldered on the output pins instead of the input pins so the LEDs couldn't properly work as they had power but no instructions.

After soldering the LEDs on the right side of the stick at the IMS Lab were Mr. Zimmer works, everything then worked as intended. These LEDs were really simple to use and fun to work with but no more tests were required.

3.3. Carving out algorithm

Instead of coding right away we thought it would be much easier to imaging first the algorithm and how the LEDs would behave.

First, we decided that it could be great if the loop restarted every second so that from one loop to the start of another, one second would have passed.

There is 8 LEDs, so we had to think of 8 things the LEDs could inform us of:

1. The first LED could inform if the Arduino card was correctly set up and powered so this LED could be set up in the *setup()* function
2. The second LED could testify that the reading loop of the algorithm was working properly by blinking every second. If the blinking stops, this means that the loop is stuck
3. The sensor needs to heat for 3 minutes before giving good values so the third LED could show if the sensor is hot or not by switching from a color to another after 180s
4. The fourth LED could be used to tell if the last value received from the sensor was verified by the checksum or not. It can be very useful if there is a connection issue between the board and the sensor as the checksum won't be verified anymore.
5. The fifth LED could show if the last value of CO2 concentration is high or not thanks to a gradient of color
6. The sixth would have the same use as the fifth except it would be based on the average of the 10 last values
7. Same as before but over the 60 last values (1min)
8. Same as before but overt the last 120 values (2min). This LED showing a high value of concentration would mean that it is more than time to open the windows

As you can see, we decided to use a gradient of color for the LEDs showing the CO2 values as we thought it would be convenient as well as pleasing for the eyes. Unfortunately, as of the way our eyes see the colors, doing a gradient of color based on the CO2 values was in fact very difficult (mixing a half of red and a half of green doesn't really make orange but a slightly reddish green color) so we decided to simply use a three colors grade.

Knowing the function of each of the LEDs we then worked all together, each creating a piece of the algorithm that we then merged together into the final algorithm.

3.4. Choosing the LEDs colors

To display information to the users, we decided to use the least colors possible so that the notifier colors would be easy to understand and to remember.

For the first LED we chose white because it is a neutral color.

For the second LED we chose blue because it is a reassuring one, meaning that everything is alright, the LED will simply blink every second from turned off to blue, then from blue to turned off.

For the third and fourth diode we chose pink and blue, the pink because it is a bright color which is close to the red to be flashy and show that the values given by the sensor are not correct (because the sensor is not hot or because of a connection issue) and blue again to show that everything is alright and the captor is working as intended.

The fifth, sixth, seventh and eighth LED follow a traffic light system so they can be red, orange or green which are colors which are easy to understand even if we don't know how the sensor works. As mentioned earlier, we chose this system over a gradient of color as it would have been overly complicated.

3.5. Coding the algorithm on Arduino IDE

Coding the algorithm on Arduino IDE was surprisingly very easy and we encountered only one problem: doing an average of more than a hundred values, values sometimes above 1 000, didn't allow the use of integer variable as on UNO the integer are coded on 2 bytes, so integer were too small to hold such huge numbers. To counter that issue we used unsigned integer but it was still not enough so we used long variables that store information on 4 bytes.

We could have foreseen this issue if we had calculated the order of magnitude of each variable that we used.

4. Organization as a group

In order to work efficiently within a group, we need to be organized. Thus, we started by creating a Google Drive folder where we shared any file we used or created. Moreover, we created a mind map to help organize ourselves, however, we quickly stopped updating it as we thought it was not really useful and helpful.

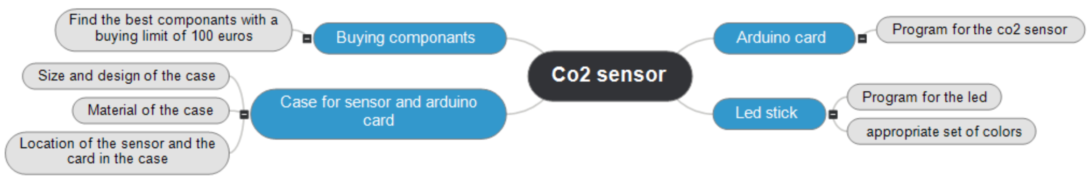
Co2 sensor
Buying componants
Find the best componants with a buying limit of 100 euros
Case for sensor and arduino card
Size and design of the case
Material of the case
Location of the sensor and the card in the case
Arduino card
Program for the co2 sensor
Led stick
Program for the led
appropriate set of colors

Figure 2 The mind map we quickly stopped using

The two main means of communication we used within our group was text messages (with a Messenger group) and voice chat (with Discord). These means of communication, especially voice chat, were very useful to work efficiently from home.

5. Final result

Unfortunately, we did not have the time to finish the case, that is to say we didn't glue the case and placed the components inside the case yet. But here are the results of our CO2 notifier project:

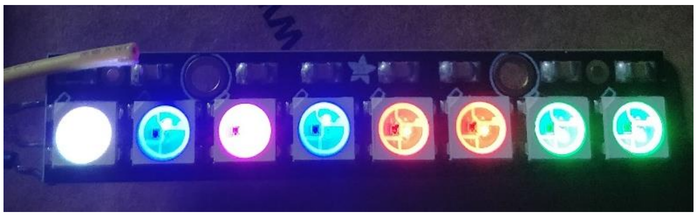
Figure 4 The LEDs stick with their respective colors

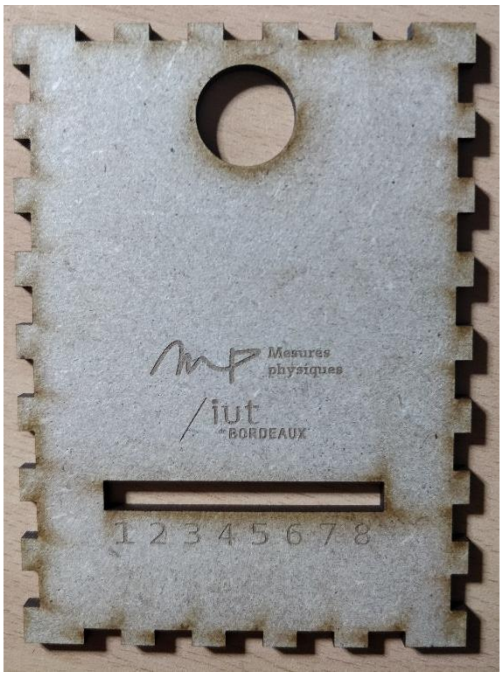

Figure 6 The top plate with the LEDs hole and sensor hole

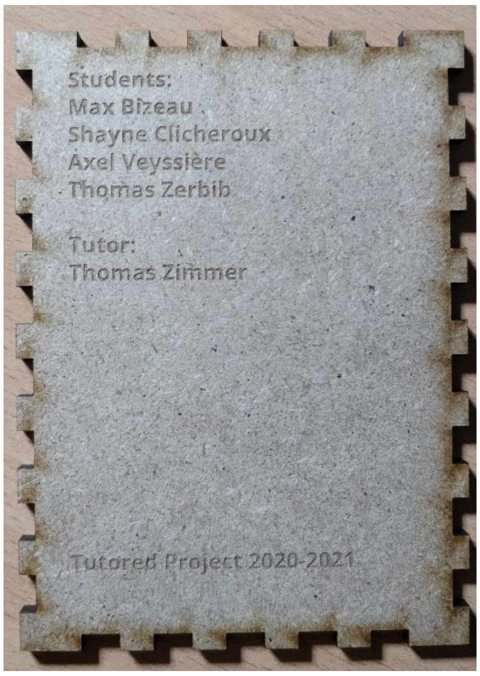

Figure 5 The bottom plate with our names

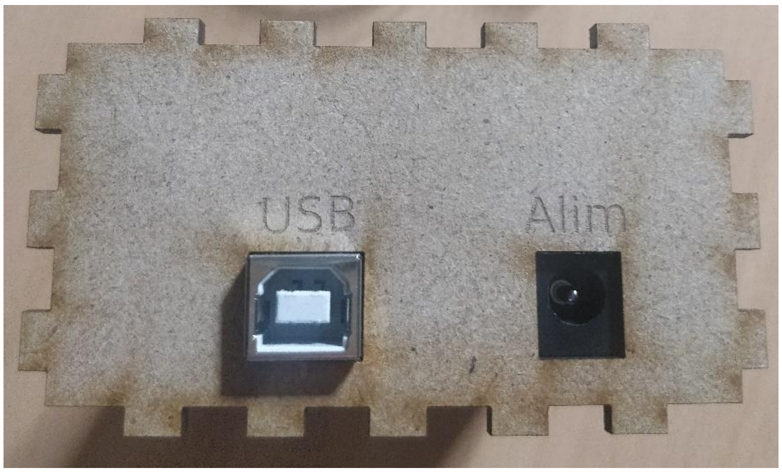

Figure 3 The I/O plate

6. Annex: code for the Arduino UNO

```
#include <SoftwareSerial.h>
#include <Adafruit_NeoPixel.h>

SoftwareSerial s_serial(2, 3);     // TX, RX
#define sensor s_serial

#define PIN          4
#define NUMPIXELS    8

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

//Construction of the instruction for the sensor
const unsigned char cmd_get_sensor[] =
{
   0xff, 0x01, 0x86, 0x00, 0x00,
   0x00, 0x00, 0x00, 0x79
};

unsigned char dataRevice[9];
int temperature=0;
int CO2PPM=0;
int CO2PPMArray[120];
unsigned int avg10=0;
long avg60=0; //unsigned int is just big enough for an average of 60 values, we use a long variable to be safe
long avg120=0; //unsigned int is too small for an average of 120 values
int time=0;
int switchingLight=0;
int test;

void setup()
{
   sensor.begin(9600);
   Serial.begin(9600);
   Serial.println("Beginning to read from sensor!");
   Serial.println("**********************************************************");
   Serial.println();

   pixels.begin(); //Initializes the NeoPixel library
   pixels.setPixelColor(0, pixels.Color(10,10,10));
   pixels.show();

   for(int i=0; i<100; i++)
   {
     CO2PPMArray[i] = 0;
   }
}

void loop()
{
   if(dataRecieve())
   {
      Serial.print("Temperature: ");
      Serial.print(temperature);
      Serial.print("  CO2: ");
      Serial.print(CO2PPM);
      Serial.println("");
```

```
  pixels.setPixelColor(3, pixels.Color(0,0,10));
}
else
{
  pixels.setPixelColor(3, pixels.Color(16,0,8));
}


CO2PPMArray[0]=CO2PPM;
for (int i=119; i>0; i--)
{
  CO2PPMArray[i] = CO2PPMArray[i-1];
}

average(CO2PPMArray, &avg10, &avg60, &avg120);

Serial.println(avg10);
Serial.println(avg60);
Serial.println(avg120);

if (switchingLight==0)
{
  pixels.setPixelColor(1, pixels.Color(0,0,10));
}
else
{
  pixels.setPixelColor(1, pixels.Color(0,0,0));
}

if(time<180)
{
  time++;
  pixels.setPixelColor(2, pixels.Color(16,0,8));
}
else
{
  pixels.setPixelColor(2, pixels.Color(0,0,10));
}
Serial.println(time);

if (CO2PPM<1000)
{
  pixels.setPixelColor(4, pixels.Color(0,10,0));
}
else if (CO2PPM>1500)
{
  pixels.setPixelColor(4, pixels.Color(10,0,0));
}

else
{
  pixels.setPixelColor(4, pixels.Color(13,5,0));
}

if (avg10<1000)
{
  pixels.setPixelColor(5, pixels.Color(0,10,0));
```

```
    }
    else if (avg10>1500)
    {
       pixels.setPixelColor(5, pixels.Color(10,0,0));
    }

    else
    {
       pixels.setPixelColor(5, pixels.Color(13,5,0));
    }

    if (avg60<1000)
    {
       pixels.setPixelColor(6, pixels.Color(0,10,0));
    }
    else if (avg60>1500)
    {
       pixels.setPixelColor(6, pixels.Color(10,0,0));
    }

    else
    {
       pixels.setPixelColor(6, pixels.Color(13,5,0));
    }

    if (avg120<1000)
    {
       pixels.setPixelColor(7, pixels.Color(0,10,0));
    }
    else if (avg120>1500)
    {
       pixels.setPixelColor(7, pixels.Color(175,0,0));
    }

    else
    {
       pixels.setPixelColor(7, pixels.Color(65,25,0));
    }

    pixels.show();
    delay(1000);
}

bool dataRecieve(void)
{
    byte data[9];
    int i=0;
    //transmit command data
    for(i=0; i<sizeof(cmd_get_sensor); i++)
    {
       sensor.write(cmd_get_sensor[i]);
    }
    Serial.flush();
    //begin reveiceing data
    if(sensor.available())
    {
       while(sensor.available())
       {
            for(int i=0;i<9; i++)
            {
                data[i] = sensor.read();
            }
        }
    }

    for(int j=0; j<9; j++)
    {
        Serial.print(data[j]);
        Serial.print(" ");
    }
    Serial.println("");

    if((i != 9) || (1 + (0xFF ^ (byte)(data[1] + data[2] + data[3] + data[4] + data[5] + data[6] + data[7]))) != data[8])
    {
        return false;
    }

    if (switchingLight==0)
    {
        switchingLight++;
    }
    else
    {
        switchingLight--;
    }

    CO2PPM = (int)data[2] * 256 + (int)data[3];
    temperature = (int)data[4] - 40;

    return true;
}

void average(int val[100], unsigned int *average10, long *average60, long *average120)
{
    for (int i = 0; i < 10; i++)
    {
        *average10+=val[i];
    }
    *average10=*average10/10;

    for (int i = 0; i < 60; i++)
    {
        *average60+=val[i];
    }
    *average60=*average60/60;

    for (int i = 0; i < 120; i++)
    {
        *average120+=val[i];
    }
    *average120=*average120/120;
}
```