A4 NE

Embedded System

PoC - Smokie

Mary Dal Zuffo
Sébastien Dintrich
Quentin Haimez
Léa Gobert

05/01/2020

ESILV LÉONARD DE VINCI | ÉCOLE D'INGÉNIEURS PARIS-LA DÉFENSE

Table of content

1 INTRODUCTION

In order to improve our skills in the field of embedded systems, especially in Architecture and Programming, we have to design a Proof of Concept. The aim of the project is to use some components to assemble an embedded system that is going to answer to a daily issue.

A proof of concept is a demonstration which aim to verify that certain concepts or theories have the potential for real world application. Our prototype is designed to determine feasibility but does not represent deliverables.

First of all, we have chosen to answer to a cigarette smoke related issue in closed areas: **how to automatize aeration in a room depending on the ambient smoke quantity, keeping in mind a comfortable temperature?**

Our solution is to create a system to manage the quantity of smoke in the air. We are going to use two sensors, the first one will be able to measure the room temperature and the second one the ambient smoke quantity. We also will need a servomotor. The idea is to open a window depending on the ratio of smoke on the air. The temperature sensor will allow us to close or block the window if the inside temperature gets below a given threshold. In order to do that we will need an Arduino, and a Raspberry Yùn.

To realise our prototype we are going to use a wi-fi connexion between the two boards.

2 DESIGN

To design this system, we will need four different features, but we had also to work on the link between those different elements.

We need a **temperature sensor and a smoke sensor**, connected to an Arduino and fixed in the room, to measure the ambient temperature and the ambient smoke quantity in the room.

We need a part of **decision**, in that part, we decide if the window has to be opened or closed regarding as the values collected by the two sensors.

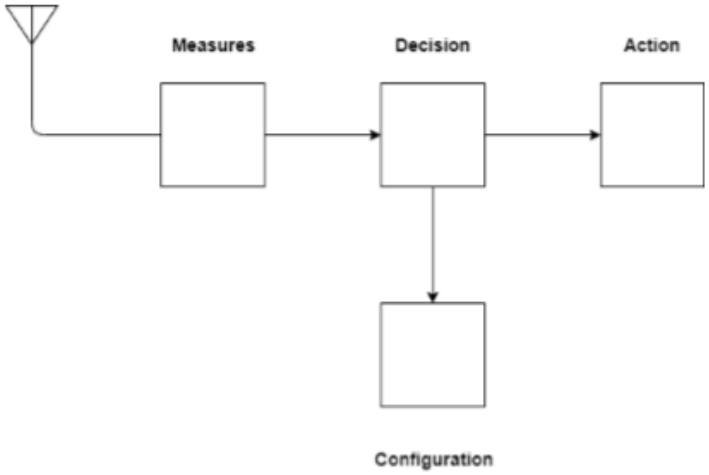
We need a **motor**, connected to a raspberry and fixed on the window, used to manually open/close the window.

We need a **secured Webpage** that displays the data measured and register them on a csv file.

We need a **secured communication** between our Arduino and raspberry and to do that, we used a WIFI-connection and the Arduino send information via an URL which open a webpage on the raspberry.

SCHEMA

Here is below the schema for our system design:



3 TUTORIALS

3.1 SENSORS

This is a beginning of the program in Arduino. We define the pin in Output or Input mode. But to operate, each sensor has his own code.

```
#include <Bridge.h>
#include <HttpClient.h>
const int sensorPin=A2;
double sensorValue=0;
const int AOUTpin=3;//the AOUT pin of the CO sensor goes into analog pin A3 of the arduino
const int DOUTpin=8;//the DOUT pin of the CO sensor goes into digital pin D8 of the arduino
const int ledPin=11;//the anode of the LED connects to digital pin D13 of the arduino

int limit;
int value;


void setup() {

  Serial.begin(9600);
  pinMode(DOUTpin, INPUT);//sets the pin as an input to the arduino
  pinMode(ledPin, OUTPUT);//sets the pin as an output of the arduino

  pinMode(sensorPin,INPUT);
  // Bridge takes about two seconds to start up
  // it can be helpful to use the on-board LED
  // as an indicator for when it has initialized
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  Bridge.begin();
  digitalWrite(13, HIGH);
  SerialUSB.begin(9600);
  while (!SerialUSB); // wait for a serial connection
}
```

1. The first sensor is the temperature one: **TMP36GZ.**

We realise the following assembly.

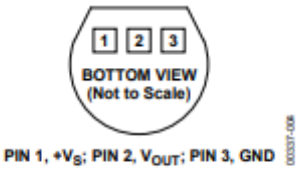
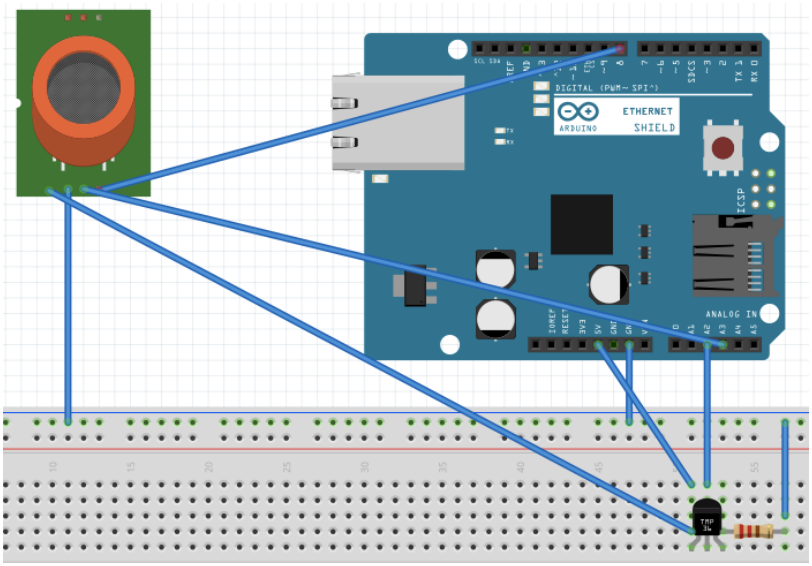


Figure 4. T-3 (TO-92)



This sensor is connected to the Arduino. So we code in C the program.

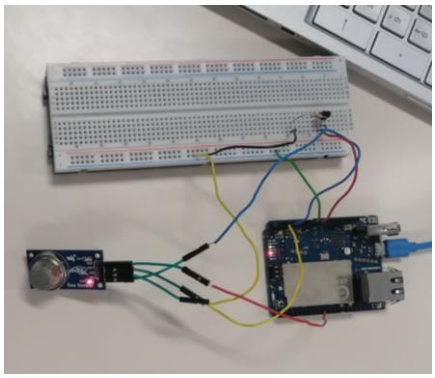
```
void loop() {
  sensorValue = analogRead (sensorPin);
  sensorValue = sensorValue;
  sensorValue=((sensorValue*10)-500)/45;
  Serial.println(sensorValue);
  delay(1000);
```

We put a delay of 1000 milliseconds between each measure. To have a value in Celsius degree, we use the conversion formula given in the datasheet :

((sensor value *10)-500)/45

2. The other sensor is a Smoke sensor. It is a gas sensor : **MQ-7**.

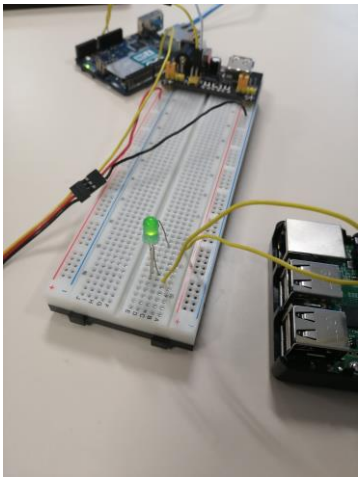
This sensor has a really high sensitivity to carbon monoxide. When the sensor is measuring, it has an adaptation period while it considers the humidity and temperature influence, before being stabilized. Its detecting range is between 20ppm and 2000ppm. So when we do the assembly we have to wait few seconds before consider the value given as reliable.



```
value= analogRead(AOUTpin);//reads the analaog value from the CO sensor's AOUT pin
limit= digitalRead(DOUTpin);//reads the digital value from the CO sensor's DOUT pin
Serial.print("CO value: ");
Serial.println(value);//prints the CO value
Serial.print("Limit: ");
Serial.print(limit);//prints the limit reached as either LOW or HIGH (above or underneath)
delay(100);
digitalWrite(ledPin, LOW);
if (value >= 200){
digitalWrite(ledPin, HIGH);//if limit has been reached, LED turns on as status indicator
}
```

To see if our program is working, we try to turn on a led when the value returned by the smoke sensor is bellow a defined threshold.

In fact, we check the operation of each sensor separately, to detect a problem more quickly. That is why we don't try turn the motor directly but used a led.



3.2 DECISION

Because we have to take in account two sensors, so two information, we have to think about the priority of one on the other, keeping in mind the comfort of the user.

For our PoC, we decided to set a minimal temperature we don't want to reach to have a tempered room. This minimal data is set in the code by us, but we can imagine letting the user choosing himself the threshold.

In our demonstration we set the temperature at 22 degree, as a *tempmin*. We also set a *fummin* rate, in order to open the window if we are above this value.

To represent the opened window, we turn on the led and vice versa. In our code we set the priority to the temperature.

The decision is made on the Arduino.

```
//prise de decision
int LED; // va renvoyer la decision prise
double tempmin= 22;
double fummin = 185;
//les valeurs sont recuperees par les capteurs precedemment
if(sensorValue <= tempmin){
  LED = 0;
}
else{
  if(value <= fummin){
    LED = 0;
  }
  else{
    LED = 1;
  }
}
```

Limit thresholds

Closed window

Opened window

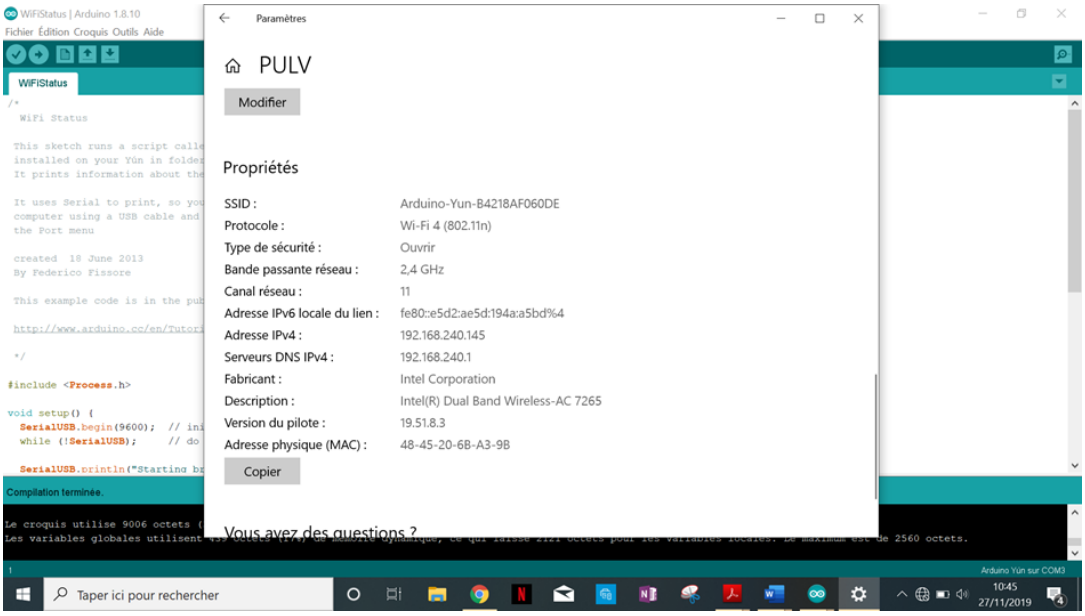
On the future, we want to have different option for the user, depending on its preferences.

In the case we install our system in a non-smoker's flat, we can say the smoke in the air is more disturbing than the temperature of the room so for him the smoke quantity take advantage on the decision making. Whereas, for a smoker's flat, we suppose that the smoke quantity is less disturbing so the temperature of the room is the most important indicator to decide if the window should be opened or closed.
We might let choose the user if he prefers to leave the advantage on the smoke or on the temperature. So, the program could be adaptable on the user preferences, thanks to the web interface.
To do this part we have to do a simple algorithm based on "if" test to know witch action we must do and when we have to start the motor.

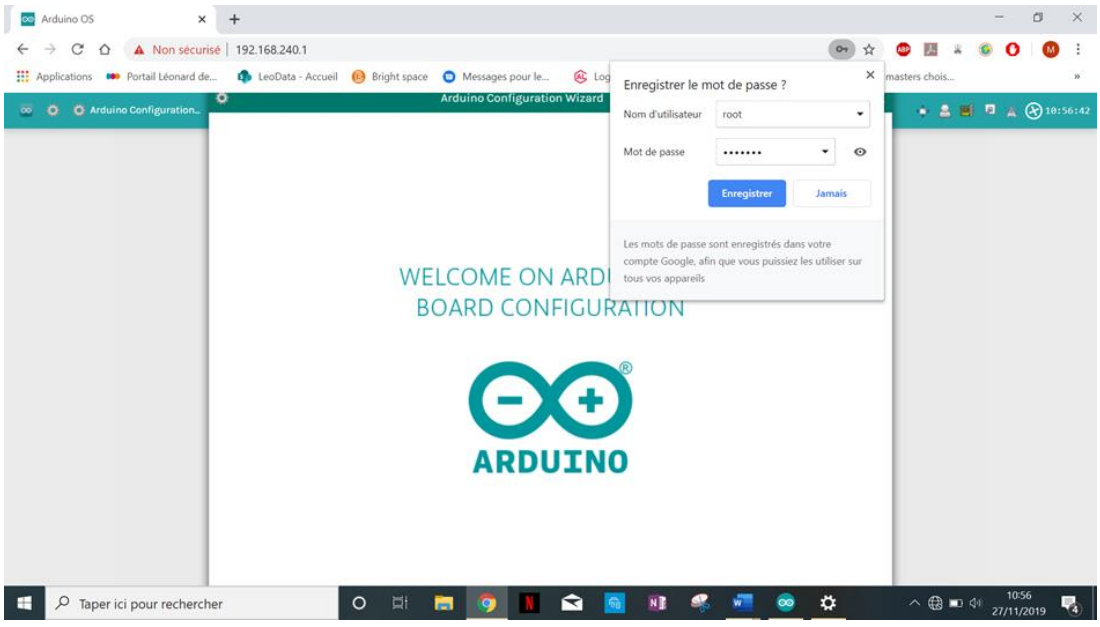
3.3 WIFI CONNEXION

In order to free us form the wire, we decide to put everything in a wi fi connexion. We use a smartphone as a modem, by sharing its internet connexion.

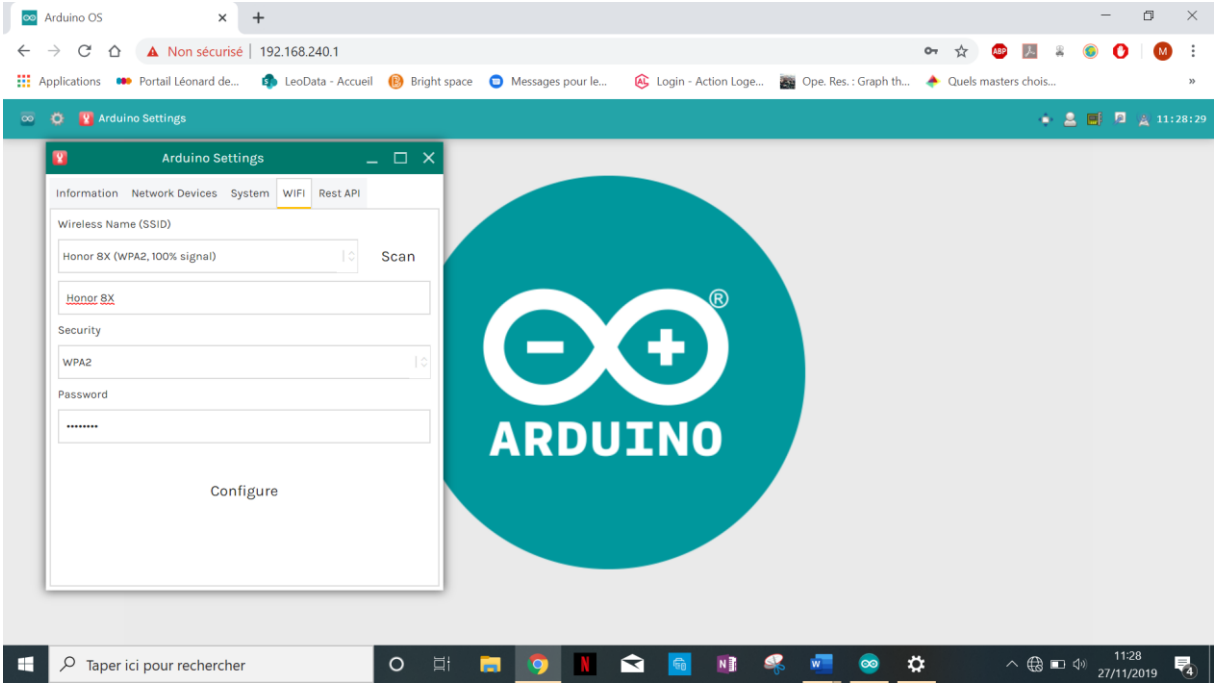
First of all we went to the Arduino webpage by browsing the IP address of the board. To get the address we can look into the parameters of our computer, after connecting the board with an ethernet wire.



After that, we browse the IP address of the Arduino. On the webpage, the username is root and the password Arduino.

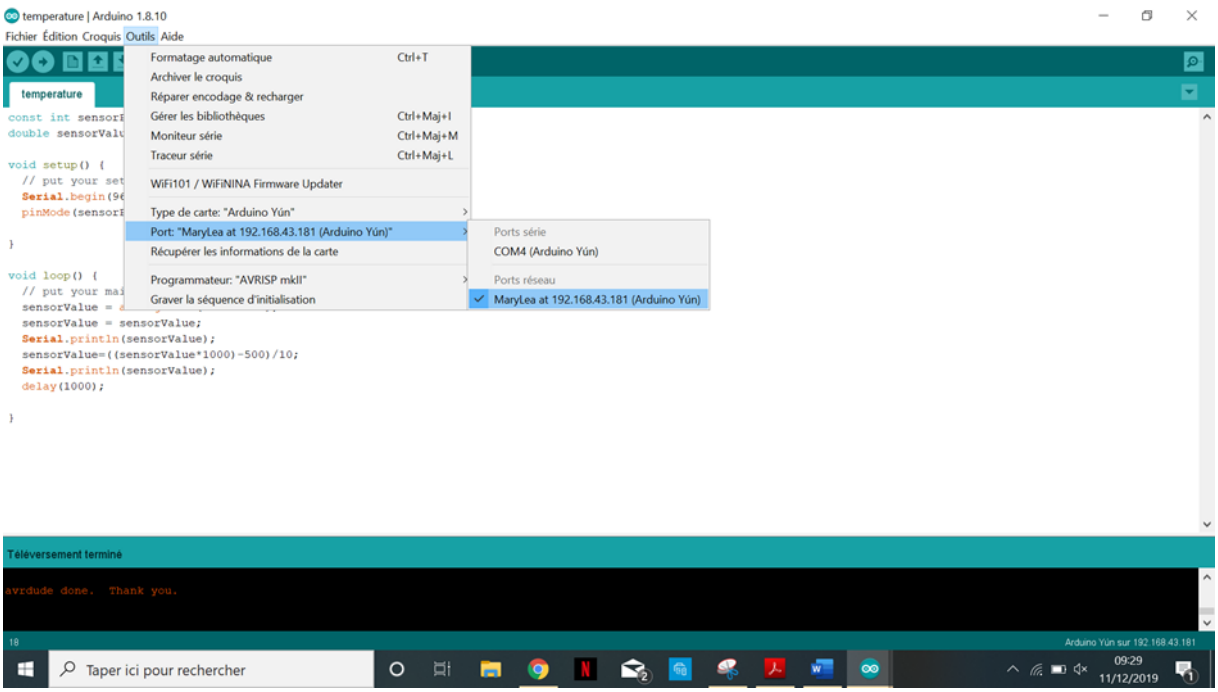


We create the password as maryLea1234.



Into the setting, we choose the smartphone's wifi, after activated the sharing connexion on the phone.

We configured the board, now even if the it is not connected to the computer, but only on an alimentation wire, if the computer is on the share point of my phone, the board will connect itself at this share point.



We choose the port : MaryLéa and the password MaryLea1234

Now, we have the Arduino connected to the wifi, we have to connect the raspberry to the same wifi, to do this part, our professor give us an important help. To do that we used a graphic user interface, so we use VNC. We have found the tutorial on the TD2 of our embedded system lesson.

We follow the following link to install all the package needed and understand how all of this works: https://www.raspberrypi.org/documentation/remote-access/vnc/

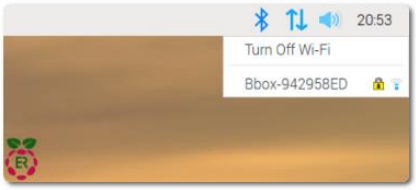
We first install the package "tightvncserver", then launch the command tightvncserver and configure a password to protect the display. (We use the same password than the board: "raspberry"). We launch the VNC server thanks to the command "vncserver :0 -geometry 1920x1080 -depth 24"

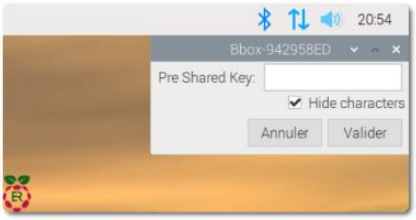
Finally, we install a VNC client on our computer with the following link:
https://www.realvnc.com/en/connect/download/viewer/windows/
And launch a connection to the VNC server of your Raspberry board.

"Server: IP address of your board followed by :1

Password: the one chosen previously, for us it's raspberry"

Once we have done that, we can have a GUI of our raspberry and we can connect it to the WIFI. Here we have the different steps of the connection: we have to select our WIFI reseau and to enter the good password.





3.4 DATA TRANSFER

To create a communication between the Arduino and the raspberry, we decided to use a webpage, specially the URL to transmit all the data.

In the Raspberry we create a webpage in order to display on a graph the information.

In the Arduino, we have three information to transmits at the raspberry:

- The temperature, in degree Celsius.
- The Smoke rate,
- An integer, 0 or 1 depending on if the window is closed or opened.

On the code, we made a loop, that browse the URL with the information refreshed.

We use the library HttpClient to do that.

```
// Initialize the client library
HttpClient client;
String raspberry = String("192.168.43.59");
String page = String("/Button.php");
String chemin = String("http://");
double temperature = sensorValue;
double fumee = value;
chemin = chemin+raspberry;
chemin = chemin+page;
chemin = chemin+String("?temperature=");
chemin = chemin+temperature;
chemin = chemin+String("&fumee=");
chemin = chemin + fumee;
chemin = chemin +String("&decision=");
chemin = chemin + LED;
Serial.println(chemin);
// Make a HTTP request:
client.get(chemin);


// if there are incoming bytes available
// from the server, read them and print them:
while (client.available()) {
  char c = client.read();
  SerialUSB.print(c);
}
SerialUSB.flush();
```

Creation of the URL

Oppening of the URL

On Raspberry, we store the information received on a .CVS file. In fact, at the first try we don't storage the information on a file, and at each sending from the Arduino we lost he previous data.

pi@192.168.43.59:22 - Bitvise xterm - pi@raspberrypi: /var/www/html

GNU nano 3.2 Button.php

```
<htlm>
 <head>
  <title>Controle</title>
 </head>
 <body>
  <?php
/*      $fp = fopen ("Fichier.txt", "r");
        $contenu_du_fichier = fgets($fp,255);
        fclose($fd);
*/      $fp = fopen ("Fichier.csv", "a+");
        $today = date("YmdHi");
        $contenu_du_fichier = $contenu_du_fichier.$today.",".htmlspecialchars($_GET["temperature"]).",".htmlspecialchars($_GET["fumee"])."\n";
        fwrite($fp,$contenu_du_fichier);
        fclose($fp);
/*      $fp = fopen ("Fichier.txt", "r");
        $contenu_du_fichier = fgets($fp,255);
        fclose($fd);
*/      echo 'Notre fichier contient : '.$contenu_du_fichier;
  ?>
 </body>
</html>
```

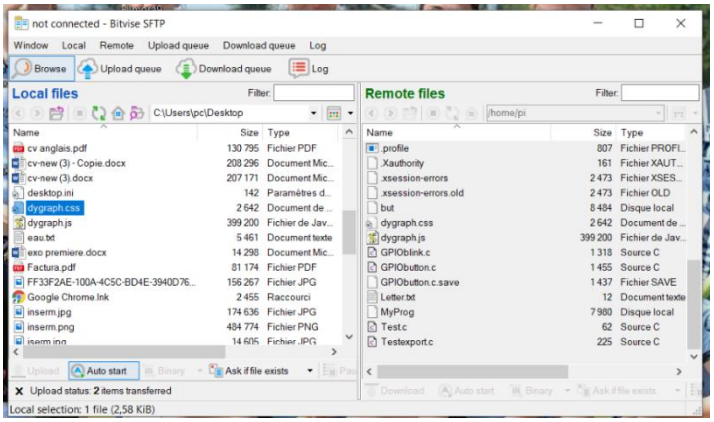
All the information is timestamped:

Here, the first value of a line is each time the date corresponding to the following value. The year, the month, the day, the hour and the minutes are all following each other without blank.

```
Date,High,Low
201912171947,15550.00,167.00
201912171948,15750.00,177.00
201912171949,15150.00,167.00
201912171950,15150.00,175.00
201912171951,14650.00,176.00
201912171952,14650.00,175.00
```

3.5 GRAPH

We decided to have access to every data measured by the two sensors when we want to, so we created a web page with the same basis than previously. To read the



data correctly we decided to trace a graph. For that, we have to install two different files on the raspberry: digraph.css and digraph.js (you can find it on the following website: *"dygraph.com/tutorial.html"*). First you have to install those on your computer and then to move them on the raspberry.

Then we write the following code on the raspberry files named "Graph" and which is on html because it's a webpage.

GNU nano 3.2 Graph.php

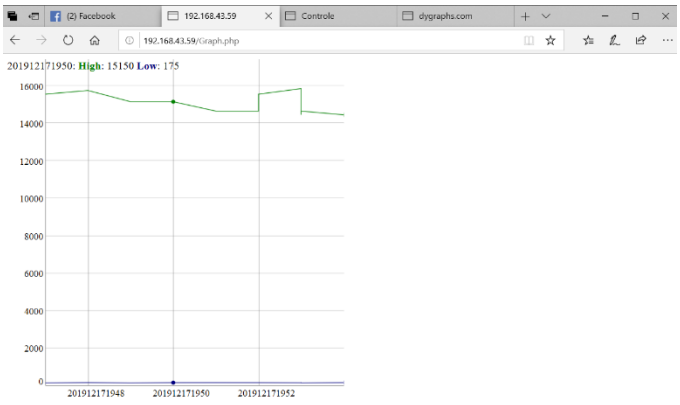
```
<html>
<head>
<script type="text/javascript"
  src="dygraph.js"></script>
<link rel="stylesheet" src="dygraph.css" />
</head>
<body>
<div id="graphdiv2"
  style="width:500px; height:500px;"></div>
<script type="text/javascript">
  g2 = new Dygraph(
    document.getElementById("graphdiv2"),
    "valeur.csv", // path to CSV file
    {}          // options
  );
</script>
</body>
</html>
```

Here, we use both files downloaded before, the width and the height define the tall of the graph and the values come from the file "valeur.csv" which is the file modified each time by the values from the sensors.

So here an example of graph obtained by the file corresponding. For the date, it's not clear to read because we must compact the date with the hour of the day.

The series high and low represented respectively the temperature and the CO2 ratio (the values of temperature have been adjusted after).

```
Date,High,Low
201912171947,15550.00,167.00
201912171948,15750.00,177.00
201912171949,15150.00,167.00
201912171950,15150.00,175.00
201912171951,14650.00,176.00
201912171952,14650.00,175.00
```



3.6 SERVO

For now, to indicate the decision and the state of the window, we used a LED. To activate or not the LED we used the page button.php where we had a line that used another code from our first Tutorial.

```
exec("./Led " .$_GET["gpio"] ." " .$_GET["value"]);
```

The gpio was the pin for the LED and value was the decision sent by the Arduino that is 1 or 0.

Led is the c program "GPIOblink.c" which has been compiled.

```
 GNU nano 3.2                          GPIOblink.c

#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

void export(int gpio)
{
        int fd;
        char buf[40];
        fd= open("/sys/class/gpio/export", O_WRONLY);
        sprintf(buf,"%d", gpio);
        write(fd, buf, strlen(buf));
        close(fd);
}
```

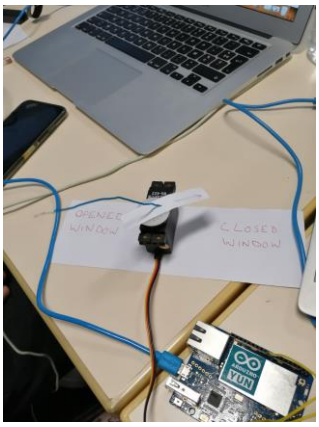
```
void direction(int gpio)
{
        int fd;
        char buf[40];
        sprintf(buf, "/sys/class/gpio/gpio%d/direction", gpio);
        fd=open(buf,O_WRONLY);
        write(fd,"out",3); //write(fd, "in",2);
        close(fd);
}
void value(int gpio,int choice)
{
        int fd;
        char buf[40];
        sprintf(buf, "/sys/class/gpio/gpio%d/value", gpio);
        fd=open(buf, O_WRONLY);
        //set gpio high status if choice=1, low status if choice=0
        if(choice==0){write(fd,"0",1);}
        if(choice==1){write(fd,"1",1);}
        // low status : "0",1 and high status : "1",1
        close(fd);
}
```

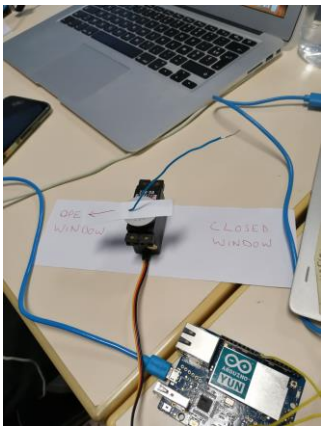
```
void unexport(int gpio)
{
        int fd;
        char buf[40];
        fd= open("/sys/class/gpio/unexport", O_WRONLY);
        sprintf(buf,"%d", gpio);
        write(fd, buf, strlen(buf));
        close(fd);
}

int main(int argc, char **argv)
{
        int gpio;
        int choice;
        if(argc>2)
        {
                gpio=atoi(argv[1]);
                choice=atoi(argv[2]);
                printf("Gpio:%d\n",gpio);
                printf("Choice:%d\n",choice);
        }
        if (choice==3)
        {
                unexport(gpio);
        }
        else
        {
                export(gpio);
                sleep(2);
                direction(gpio);
                value(gpio,choice);
        }
}
```

We used a servo to modelized the ability to open or close the window if the decision asks to do this. So, we connected the servo to an Arduino card, with the following code, that was functional.

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
                    // opening the window (as if the command 1 is received from the sensors)
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(10);
  }
  delay(5000);
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);                 //closing the window (as if the command 0 is received)
    delay(10);
  }
  delay(5000);
}
```

To connected it to our program and system, we must use a python program, to right the thing that when the decision's led was shining, the window should be opened. And to send it from the raspberry receiving the decision from the first Arduino, to the Arduino governing the servo. We used and USB B connection between the raspberry and the second Arduino.





4 FAQ

Here we list the main questions and issues we had.

First, we have a problem with the user rights, especially when the Arduino open the URL and when with the URL we want to write on the csv file and register the information. So, we have to authorize all users to write on the csv file and to open the URL of the webpage coded on the raspberry.

With the command "chmod og+w" we had to all the other users the right to write on the file.

```
pi@raspberrypi:/var/www/html $ sudo chmod og+w Fichier.txt
pi@raspberrypi:/var/www/html $ ls -Al
total 68
-rw-r--r-- 1 root root   444 Nov 25 20:09 ajax.php
-rw-r--r-- 1 root root   590 Dec 11 09:56 Button.php
-rw-rw-rw- 1 root root     6 Dec 11 09:59 Fichier.txt
-rw-r--r-- 1 root root 10725 Nov 20 10:02 index.html
-rwxr-xr-x 1 pi   pi    8444 Nov 21 13:40 Led
-rw-r--r-- 1 root root   498 Nov 25 19:55 LedOnOff.php
-rwxr-xr-x 1 pi   pi    8324 Nov 20 08:54 test
-rwxr-xr-x 1 pi   pi    8324 Nov 20 10:23 Test
-rw-r--r-- 1 root root   260 Nov 25 20:35 test.php
```

Then we have a problem with the WIFI connexion on the raspberry because our first raspberries were not equipped with a WIFI module. So, we have to change the raspberry 3 for a 4.

Finally, because of a missing communication in the group, the servo was code to function with a python code and all the other parts were made with a C code. We can adapt our different codes to fusion all the parts, but we don't have the time.

5 CONCLUSION

To conclude, our Proof of Concept is functioning: we can, thanks to the data given by the two sensors, monitor the window. In our case, the servo which monitors the window is just representing by a LED, that is lighting if the window is currently opened. We managed to get the engine, but we don't connect it with the other parts of the system because we were too short in time. We see that it could be relatively easy to connect it, and we could even improve our creation by connecting it directly to the air conditioning system of a room or building.

Another axis of improvement would be to allow the user by accessing the web page, to change the temperature limits etc...

In an economic way, our poc can be usefull in every building to manage the quality of the air in a room. Nowadays, buildings are more and more connected so we think it can be interesting to improve our work to adapt it for office building or school for example. It can be a cheap installation because it is only few boards, no construction work because it is a wireless system. We have make some research, if one raspberry can make the decision for different Arduino, and how many.

To sell our work, we can do a simulation in 3D of the system, and how it can be incorporate into a room in a discreet way. We also can improve the esthetic of our webpage and make it more user friendly.

Finally, a question we can ask and has to be deepen is the security of our system. In fact, the data send with the URL are not from a known source, so everybody can modify that without any difference that if it was the Arduino sending data.