



ÉCOLE
D'INGÉNIEURS
PARIS-LA DÉFENSE

EMBEDDED SYSTEMS

Proof of Concept: *Smart Room* *Design Report*

Corentin HOCHMAN

Victor MIR EMARATI

Maureen PIERRE

Martin PINTIAU

ESILV 4A NE2

Mr COURBIN et Mr FAUBERTEAU

March 2019

Contents

1	Team presentation	2
1.1	Corentin Hochman	2
1.2	Victor Mir Emarati	2
1.3	Maureen Pierre	2
1.4	Martin Pintiau	2
2	Problem description	2
3	Modus operandi	2
4	Materials	3
5	Electronic scheme	3
6	Features detailing	4
6.1	Arduino	4
6.1.1	Sensing humidity and pressure	4
6.1.2	Data transmission	4
6.1.3	LCD display to the user	4
6.2	Raspberry	5
6.2.1	Data saving, python code	5
6.2.2	Data display	6
7	Problems encountered	7

1 Team presentation

1.1 Corentin Hochman

I am happy to work on a topic useful to the planet. This project will be complementary to my Pi2. Indeed, with Maureen our goal is to model in 3D the L building of the pole leonard de vinci. In addition to this modeling we must add Lora sensors to find ways to reduce its ecological and economic impact. These two projects will allow us to put into practice all that we learn during the theoretical courses.

1.2 Victor Mir Emarati

I'm proud to be part of the smart room project. In fact this project is born thanks to different observations we had done in the school. That makes this project more concrete. Act on the different poles of energy consumption to record data and try to improve the system gives interest to the project.

1.3 Maureen Pierre

This project is in line with the PI2 project I'm working on with Corentin. On this project in SmartCities, the goal is to make the working environment more enjoyable thanks to the technologies available. By promoting this process with Lora sensors, we reduce consumption and therefore the environmental impact. The projects are mainly linked, which makes it possible to put theoretical knowledge into practice.

1.4 Martin Pintiau

Such as Victor I'm not in the PI² project about the smart building modeling. This doesn't impact my motivation on this project, we often talk about the Smart buildings in class but this is the first time we truly design and implement a solution for this problem of controlling and monitoring a room in a building but automatically and the smart way, to save power.

2 Problem description

The origin of our problem comes from the sustainable problem. Today humanity has to respond to the global warming problem as the limited resources. One of the sector where there is a lot of power economy to do is the buildings sector. Humans doesn't have their senses sharpened enough to optimize the energy consumption in buildings.

Basically, the problem is to monitor every energy consumption pole in the room and regularly check if they are useful, if not, try avoid useless energy consumption.

For our case, we focused on the data collection and transmission but not on the active system. In other words, we provided information to detect and solves the problem but our system doesn't intervene on it. Also, we focused on the sensing of humidity and temperature to have an example for indoor climate control. Of course, there are other power consumption pole in a building that would need others sensors.

3 Modus operandi

Our system is divided in 4 parts :

- First the sensing part which job is to collect data.
- Then, we also have a display function for the user in the place
- The transmission of the data to a server part that here is played by a Raspberry
- And finally the organization of data to make them humanly readable.

4 Materials

To do so, we needed :

- An arduino Yün (even if an arduino Uno would have been enough)
- A Raspberry PI
- A DHT22 sensor for humidity and temperature.
- A 2 line LCD display screen.
- A potentiometer
- A computer for initialization and HTML display in the browser
- Wiring

5 Electronic scheme

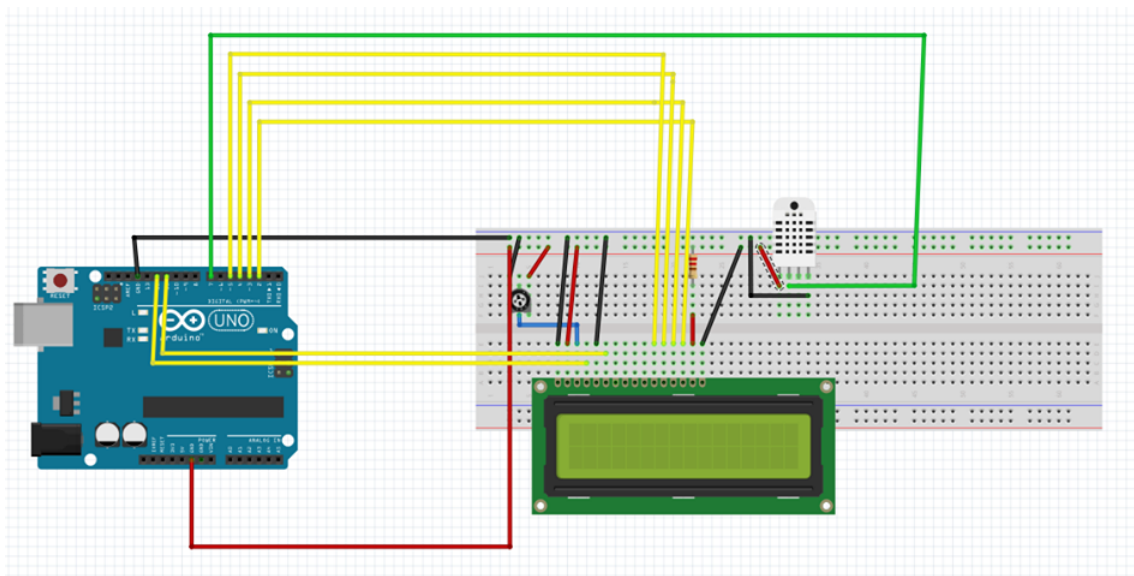


Figure 1: Electronic scheme

6 Features detailing

6.1 Arduino

Here is our Arduino program.

```

1  #include <LiquidCrystal.h>
2  #include <DHT.h>
3  #define DHTPIN 8
4  const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
5  LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
6
7  void setup() {
8      Serial.begin(9600);
9      dht.begin();
10     lcd.begin(16, 2);
11 }
12
13 void loop() {
14     // Wait a few seconds between measurements.
15     delay(2000);
16     // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
17     float h = dht.readHumidity();
18     // Read temperature as Celsius (the default)
19     float t = dht.readTemperature();
20     // Read temperature as Fahrenheit (isFahrenheit = true)
21     float f = dht.readTemperature(true);
22
23     // Check if any reads failed and exit early (to try again).
24     if (isnan(h) || isnan(t) || isnan(f)) {
25         Serial.println(F("Failed to read from DHT sensor!"));
26         return;
27     }
28
29     Serial.print(t);
30     Serial.print(",");
31     Serial.println(h);
32
33     lcd.setCursor(0, 0);
34     lcd.print("Hum: " + h);
35     lcd.setCursor(0, 1);
36     lcd.print("Temp: " + t);
37 }

```

Figure 2: Arduino code

6.1.1 Sensing humidity and pressure

To detect humidity and pressure values, we use the Sensor DHT22. This sensor send us two float representing humidity and temperature, we called the "h" and "t" in the following arduino code. We use the DHT.H library to use the DHT function. Our code is built on the demonstration code given with the library, it's enough to get our two values once every two seconds (delay line 22 in the code). This code sample also provide an acquisition test (if line 33 in the code) to be able to return something in case of failure of the data collection. In our case it return a string which isn't compatible with our graph. That way the program continues even if data aren't readable and it doesn't save random or null data.

6.1.2 Data transmission

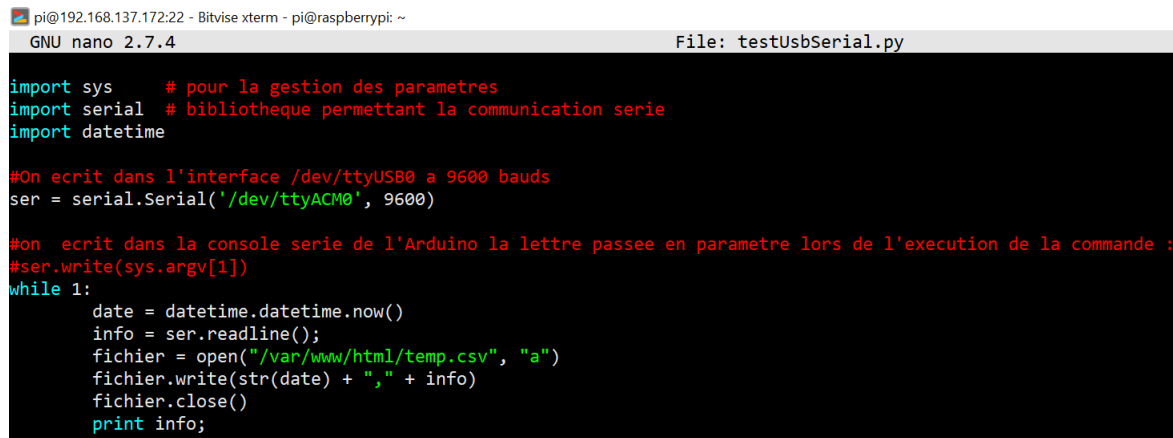
Our two values are sent to the raspberry trough the serial connection (USB Cable). In terms of code, this is done by the 3 Serial.print line 38 in the code. The two values are transmitted as a string with a coma between the two values because after, it will be saved in .csv that split values with comas. Also, last print is a println to signify to the raspberry that a full package of data has been fully transmitted. As long as the arduino is running, data will be sent trough the serial connection every 2 seconds.

6.1.3 LCD display to the user

We also implemented a display of out two values to the user on an LCD screen. The LCD screen has 2 lines so we decided to use one for each value. In the code, first we use the corresponding library and set the several pin. Then we basically set the cursor on the first line first char, print the humidity level, set the cursor on the second line first char and print the temperature.

6.2 Raspberry

6.2.1 Data saving, python code



```

pi@192.168.137.172:22 - Bitvise xterm - pi@raspberrypi: ~
GNU nano 2.7.4 File: testUsbSerial.py

import sys      # pour la gestion des parametres
import serial   # bibliotheque permettant la communication serie
import datetime

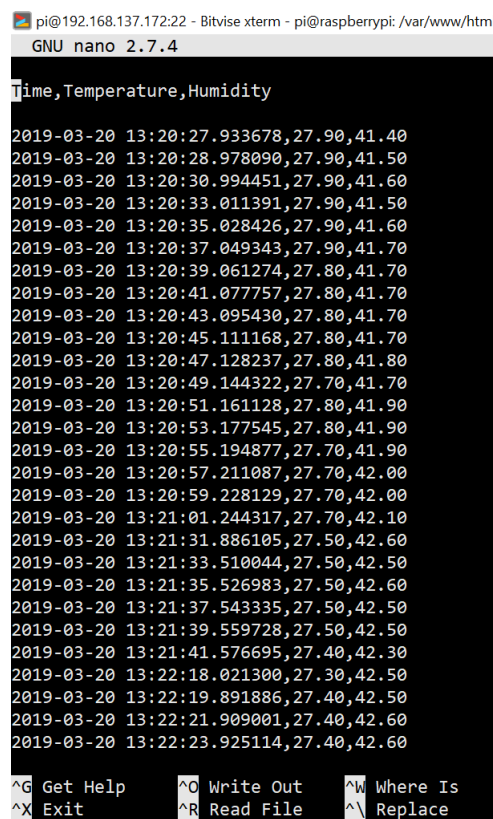
#On ecrit dans l'interface /dev/ttyUSB0 a 9600 bauds
ser = serial.Serial('/dev/ttyACM0', 9600)

#on ecrit dans la console serie de l'Arduino la lettre passee en parametre lors de l'execution de la commande :
#ser.write(sys.argv[1])
while 1:
    date = datetime.datetime.now()
    info = ser.readline();
    fichier = open("/var/www/html/temp.csv", "a")
    fichier.write(str(date) + "," + info)
    fichier.close()
    print info;

```

Figure 3: Python code for the raspberry

This code use the library serial to use the serial connection and the library datetime to recover the direct time. First we set our serial frequency to 9600 like the arduino to make the connection synchronous. Then in an infinite loop (program shut down only when we do it manually), every times we receive a data, we write it with the time in a CSV file the following way : every line correspond to a set of data (one every 2 sec) and each of those data set are composed of the time(AAAA-MM-DD HH:MM:SS:XXXXXX), the temperature (XX.XX) and finally humidity (XX.XX), values are separated with a coma and the first line is here to name each columns.



```

pi@192.168.137.172:22 - Bitvise xterm - pi@raspberrypi: /var/www/html
GNU nano 2.7.4

Time,Temperature,Humidity

2019-03-20 13:20:27.933678,27.90,41.40
2019-03-20 13:20:28.978090,27.90,41.50
2019-03-20 13:20:30.994451,27.90,41.60
2019-03-20 13:20:33.011391,27.90,41.50
2019-03-20 13:20:35.028426,27.90,41.60
2019-03-20 13:20:37.049343,27.90,41.70
2019-03-20 13:20:39.061274,27.80,41.70
2019-03-20 13:20:41.077757,27.80,41.70
2019-03-20 13:20:43.095430,27.80,41.70
2019-03-20 13:20:45.111168,27.80,41.70
2019-03-20 13:20:47.128237,27.80,41.80
2019-03-20 13:20:49.144322,27.70,41.70
2019-03-20 13:20:51.161128,27.80,41.90
2019-03-20 13:20:53.177545,27.80,41.90
2019-03-20 13:20:55.194877,27.70,41.90
2019-03-20 13:20:57.211087,27.70,42.00
2019-03-20 13:20:59.228129,27.70,42.00
2019-03-20 13:21:01.244317,27.70,42.10
2019-03-20 13:21:31.886105,27.50,42.60
2019-03-20 13:21:33.510044,27.50,42.50
2019-03-20 13:21:35.526983,27.50,42.60
2019-03-20 13:21:37.543335,27.50,42.50
2019-03-20 13:21:39.559728,27.50,42.50
2019-03-20 13:21:41.576695,27.40,42.30
2019-03-20 13:22:18.021300,27.30,42.50
2019-03-20 13:22:19.891886,27.40,42.50
2019-03-20 13:22:21.909001,27.40,42.60
2019-03-20 13:22:23.925114,27.40,42.60

```

Figure 4: CSV file architecture

6.2.2 Data display

To make the data humanly readable, we decided to group them in a graph printed on a HTML page. To do so, we used the following code :

```

pi@192.168.137.172:22 - Bitwise xterm - pi@raspberrypi: /var/www/html
GNU nano 2.7.4

<html>
<head>
<script type="text/javascript"
  src="dygraph.js"></script>
<link rel="stylesheet" src="dygraph.css" />
<meta http-equiv="refresh" content="1">
</head>
<body>
<div id="graphdiv2"
  style="width:500px; height:300px;"></div>
<script type="text/javascript">
  g2 = new Dygraph(
    document.getElementById("graphdiv2"),
    "temp.csv", // path to CSV file
    {}         // options
  );
</script>
</body>
</html>

```

Figure 5: HTML code

This code has the path of our CSV file and print the corresponding graph using the first column as abscissa values, so here it will print temperature and humidity in function of time. The sixth line starting with "<meta" make this page refresh every seconds so that the graph is actualized with new data incoming every 2 seconds. Our graph is built with the help of dygraphs which is known for being a fast, flexible open source JavaScript charting library. The result is really functional:

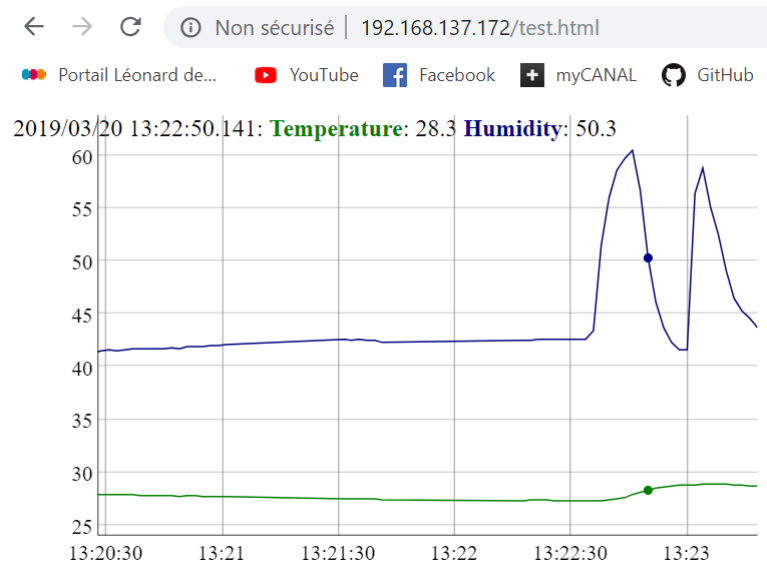


Figure 6: Graph on the HTML page

7 Problems encountered

At the beginning we wanted to build an active system that could intervene on the climate regulation of the room but we couldn't do anything better than simulate it because we didn't had access to climate system that could interact with our Arduino. So we decided to focus more on the data compilation and transmission.

In this project, none of us had already worked with python, so there was a lot of miss understandings at the beginning but achieved to learn the basic quickly and to get the code done.

First of all, we also wanted to send our data report by mail but we started to code this additional feature the last night and we didn't manage to make it work.

This is for the major issues we had on this project, of course there are others less important but we won't see them here because that's the point of this kind of project, try, fail, and try again.

List of Figures

1	Electronic scheme	3
2	Arduino code	4
3	Python code for the raspberry	5
4	CSV file architecture	5
5	HTML code	6
6	Graph on the HTML page	6