ALLEGRE Flore
CAILLY Adrien

PoC – Proof Of Concept
Engineering Design Process Application

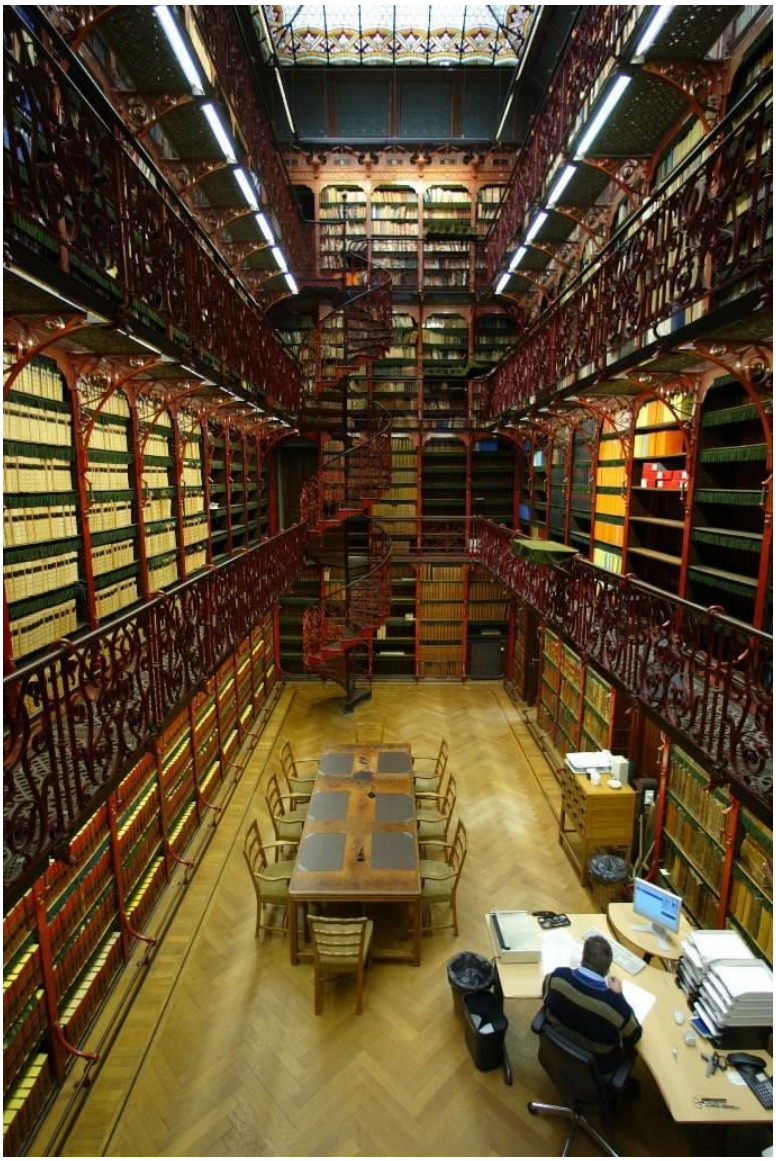


Table des matières

I) Definition of the problem and theoretical solutions

1) Definition of the problem

Our solution is designed for the archives needing a reliable system allowing them to preserve sensitive documents. This system meets the keeping constraints imposed by such documents. Furthermore, it facilitates the storing and the borrowing of those documents, while ensuring their safety and these of the people using it.

2) Specify requirements

Design requirement:

- Conserve the books:

  -Control the humidity (sensor + ventilation + humidifier)

  -Control the air quality (sensor + ventilation)

  -Control the luminosity (sensor + adaptable light)

  -Control the temperature (sensor + climatization + heater)

  -Adapt the position of the books (different size of shelfs)

- Manage the books (WebApp):

  -Easy to use Web Page (few steps)

  -Show available books (complete list with indicators)

  -Indicate the characteristics of the books (entries for: binding, conservation state, …)

  -Classify clearly the books (multiple filters)

- Protect the book from fire:

  -Control the entry of dioxygen in the room (create a vacuum inside the room)

  -Control the heat and carbon dioxide inside the room (sensor)

- Protect the access to the room:

  -Detect motion in the room (motion sensor + alarm)

- Protect the users inside the room:

-Disable the robotic arm when someone enters the room (presence sensor + sleep mode for the arm)

- Retrieve/store the books:

  -Robotic arm

  -Books localisation systems (LED + plan)

3) Brainstorm solutions

- Conservation of the books:

  - For the luminosity, we suppose that the lighting is uniform in all the room. For that, there a no windows and the lights are symmetrically distributed on the ceiling.

  - For the temperature, it must be kept around 18°C, one sensor should be enough, but it is important that the insulation of the room is well made in order not to create differences of temperature inside. Any type of heating can be used if the source is not to close to the books, but we would prefer it to be from the floor or the ceiling. The cooling can be managed with some air conditioners.

  - For the air quality, we just need to make sure it is kept good. We can control its level with one sensor and ventilate the room when needed with an AHU.

  - Some books need to be lengthened for a better conservation. Provide a special zone for these books.

  - We will separate the room in two parts and install in each one sensor, one ventilation and a humidifier to make sure the humidity stays as close as possible to 50%.

- Manage the books:

  - Simple app with two functionalities: A list of the books with the information about whether they are disponible or not (a list of filters allows the user to sort the books); an interface to return a book with the name of the author, the title, the year of the book and the category (stores the book if it already has an emplacement or allows it one if it has not).

  - When you select a book, a description indicates the characteristics of the book: type of binding, state of conservation, …

- Protect the books form fire:

  -We adapt the number of fire detectors to the size and the shape of the room.

  -In case a fire start is detected, a vacuum pump vacuums all the air outside of the room to suffocate the fire.

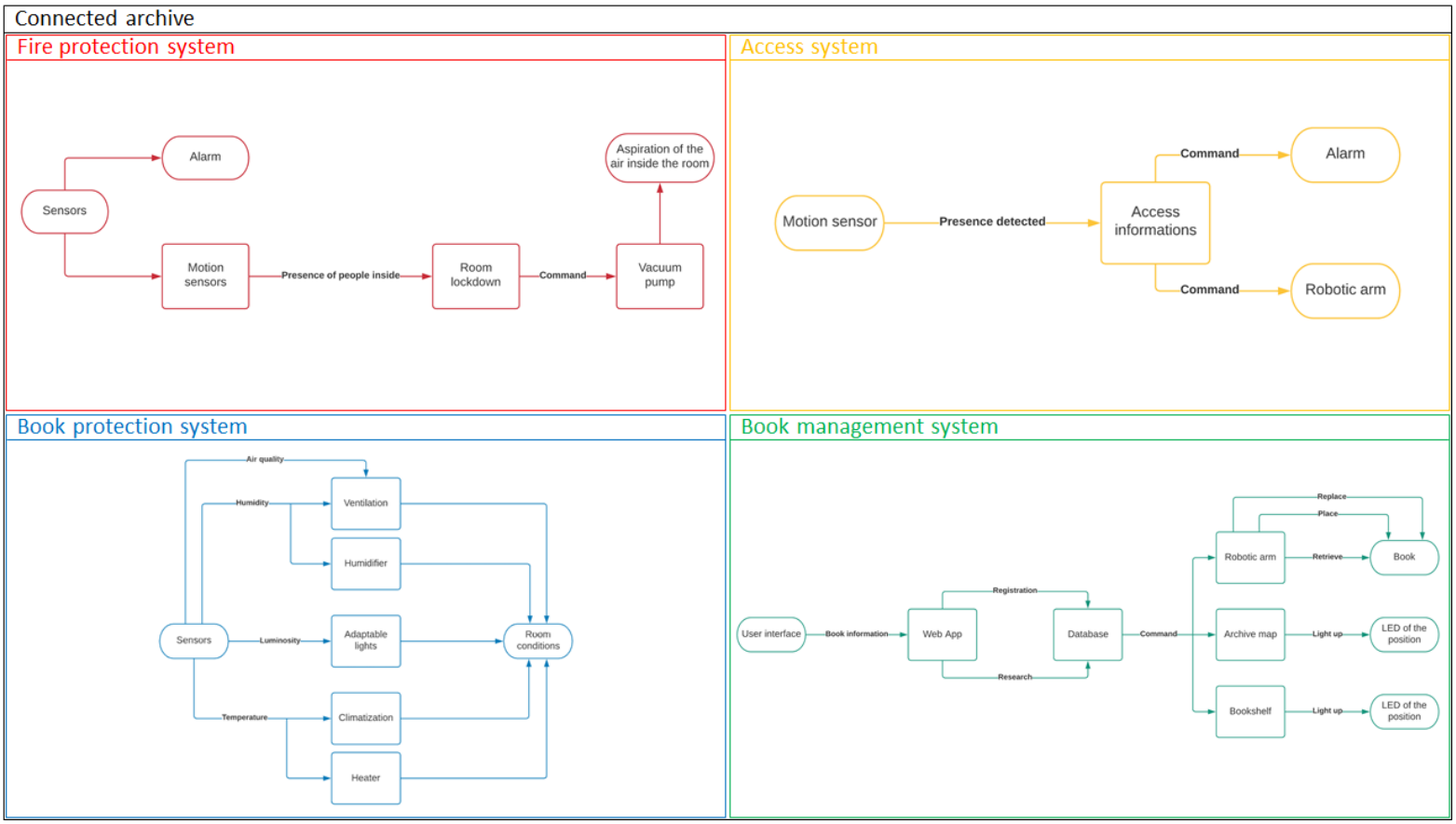
- To protect the access to the room we put motion sensors in many places (depending the form and the room size). If a motion is detected whereas no one should be there, an alarm sounds in the building.

- To protect the users inside the room: if the robotic arm is not installed or not working, there is no other problem else than putting to sleep the vacuum until the door is closed and the people gone. If the robotic arm is installed, it must be put to sleep and store somewhere in the room before the people get inside.

- To retrieve and store the books, we're going to use a robotic arm that will navigate inside the room and grab the book we want to retrieve and bring it to the user or store the one we want to return and bring it back to its place.

II) Block Diagram and equipements

1) Block Diagram



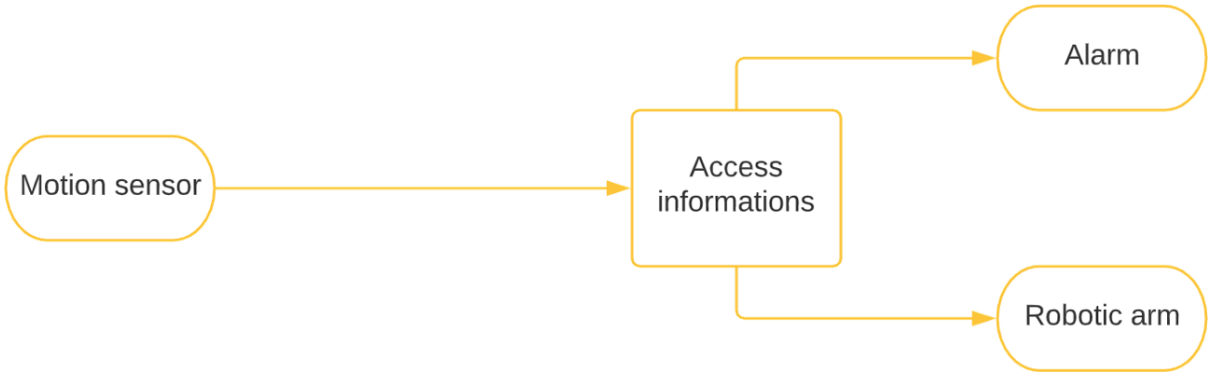
2) List of necessary equipments

- HC-SR04 (ultrasounds)
- BM 680 (Temperature, humidity, gas)
- 6 LEDS

III) The solution

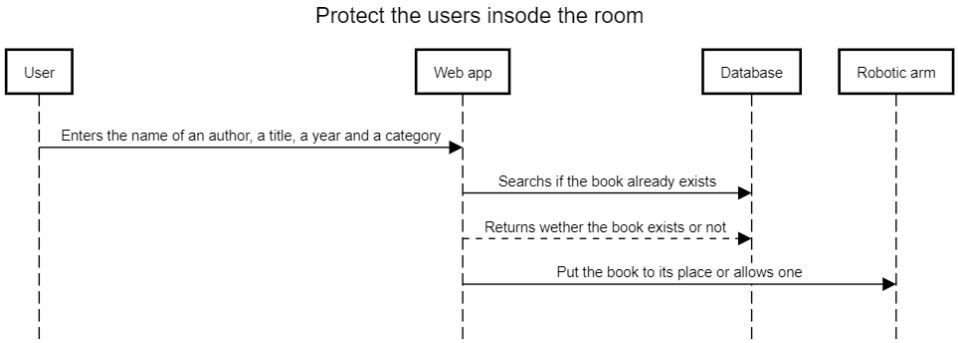
1) Access system

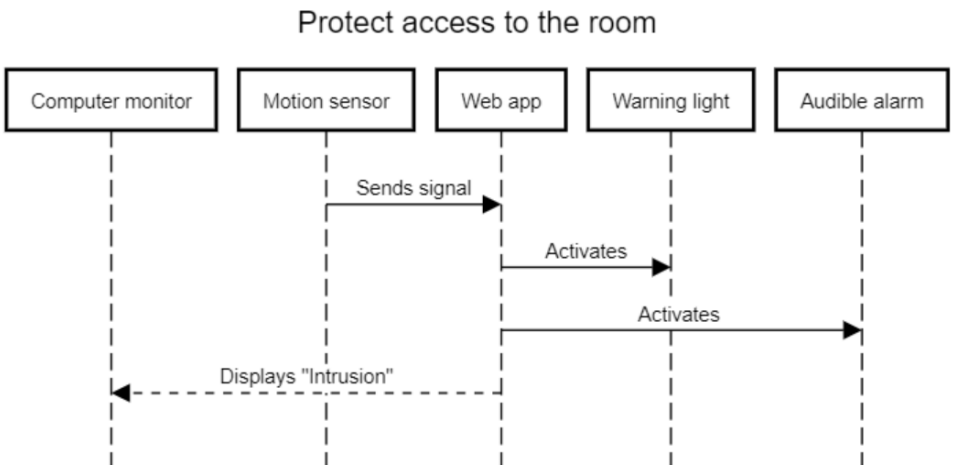
a) Block Diagram



Access system: If a motion is detected when the access to the room is authorized, then the robotic arm is disabled by an electric signal from the Web app; If a motion is detected when the access to the room is denied, then a signal from the web app activate the alarm.

b) Sequence Diagrams

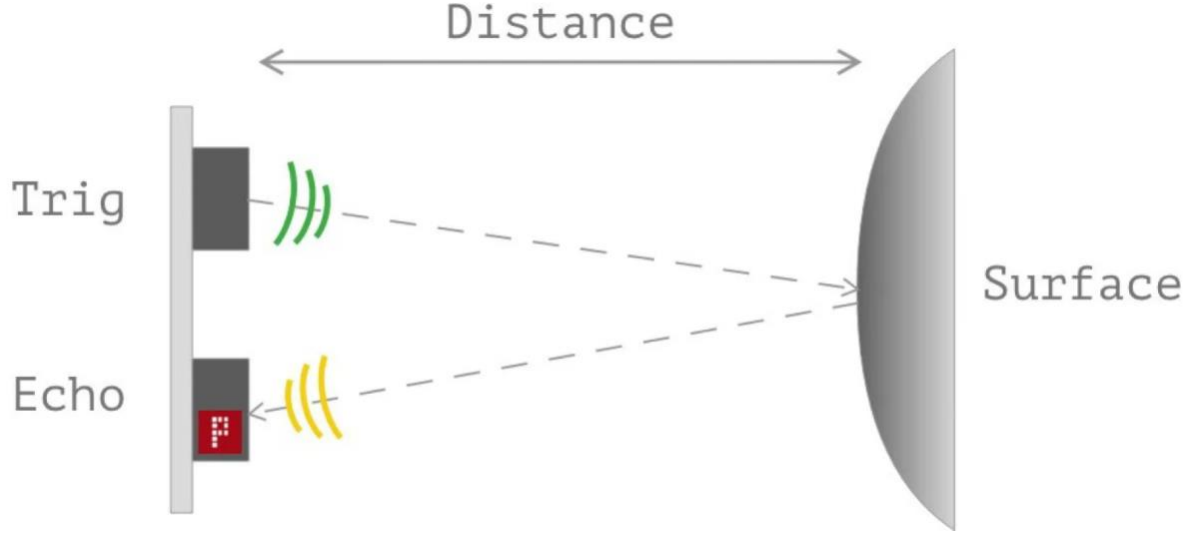




c) Explanation

As we couldn't have a motion sensor, we used an ultrasonic sensor (HC-SR04 sensor) to know if there is someone in the room. The idea is to put an ultrasonic sensor on each row, then with this sensor it is possible to calculate the distance between the sensor and the first item encountered.

Indeed, the ultrasonic sensor is made of two parts: one transmitter and one receiver. Or we know that the distance is given by: d = (C x T)/2 where c is the celerity of the sound in the air, T the time taken by the wave to go from the transmitter to the wall and the wall to the receiver. Then we divide by 2 to have just one time the distance between the wall and the sensor. [1]



As nobody should enter in the room, the distance should be always the same (distance sensor – wall). In our example, the distance between the wall and the sensor is 20m. We have decided to display the graphic of the distance in real time on the dashboard so that the user should always see a straight line on the graph. If it is not the case, it signifies that there is a person is inside the room.

d) Code [7]

Many steps are required to get the distance thanks to the ultrasonic sensor and to display it on a web server:

1) Install the filesystem uploader plugin to upload the HTML file to the ESP32 flash memory [2]
2) Installing libraries. Some libraries are not available to install through the Arduino Library Manager, so we have to copy the library files to the Arduino Installation folder.
3) Organize the files. To build the web server we need two different files. The Arduino sketch and the HTML file which has to be inside a folder called data inside the Arduino sketch folder
4) Create the HTML file. The file created is named index.html. You can find it in the Annex.
5) Program Sketch

First, we call the libraries which are going to be needed in the program.

- **Ultrasonic** calculates automatically the distance
- **WIFI** permit to connect the ESP32 in WIFI
- **AsyncTCP** and **ESPAsyncWebServer** permit to connect to the Web Server
- Thanks to **SPIFFS**, we can write the HTML and CSS in separated files and save them on the ESP32 filesystem. [3]

```
#include <Ultrasonic.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>
```

Then, we instantiate trig and echo pins for ultrasonic sensors. We also prepare the future connexion to the WIFI with entering our WIFI's ID and password and create an AsyncWebServer object on port 80.[4]

```
Ultrasonic ultrasonic (4, 5);
const char* ssid = "AndroidAPb78d";
const char* password = "********";
AsyncWebServer server(80);
```

We create a function to get the distance thanks to the ultrasonic library.

```
String getDistance() {
  // Read Distance
  float d = ultrasonic.read();
  if (isnan(d)) {
    Serial.println("Failed to read from HC-SR04
sensor!");
    return "";
  }
  else {
    Serial.println(d);
    return String(d);
  }
}
```

Initialize the filesystem

```
void setup () {
  // Serial port for debugging purposes
  Serial.begin (115200);


  if (! SPIFFS.begin ()) {
    Serial.println ("An Error has occurred while mounting
SPIFFS");
    return;
  }
```

Connexion to Wi-fi thanks to ssid and password which have been entered previously

```
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
```

This line is useful to print ESP32 Local IP Address that we are going to type in the search bar :

```
  Serial.println(WiFi.localIP());
```

The following part of the code is the route for the web page. When a request is received on the URL, we send the HTML text that is saved in SPIFFS under the index.html.

```
  server.on ("/", HTTP_GET, [] (AsyncWebServerRequest *
request) {
    request-> send (SPIFFS, "/index.html");
  });

  server.on ("/distance", HTTP_GET, []
(AsyncWebServerRequest * request) {
    request-> send_P (200, "text / plain", getDistance().
c_str ());
  });
```

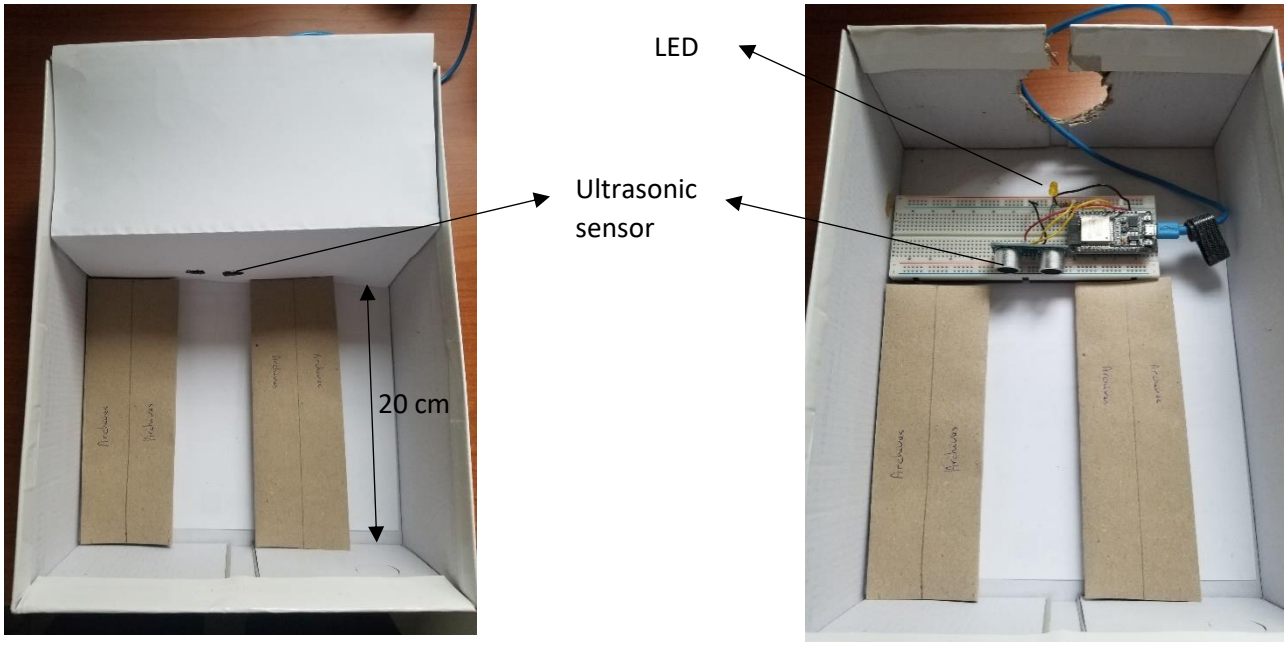
Start the server:

```
server.begin ();
```

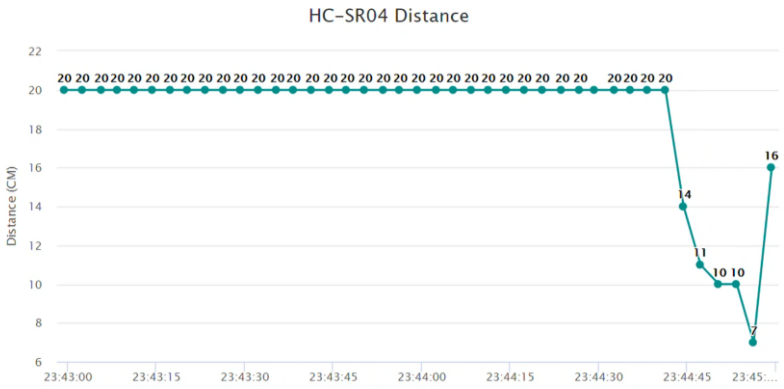
We leave the Arduino loop function empty because the server handles the requests asynchronously:

```
void loop()
{
}
```

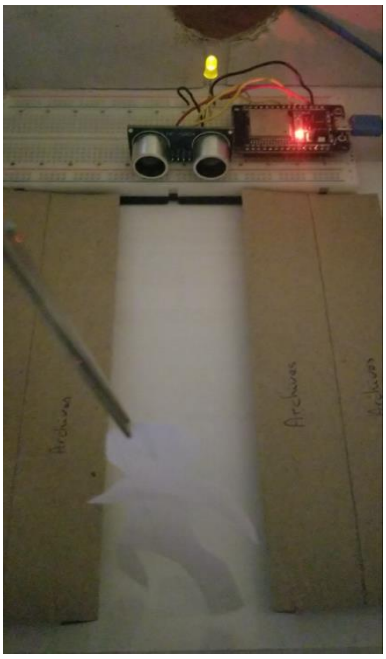
6) Results



Model of a row with motion detection



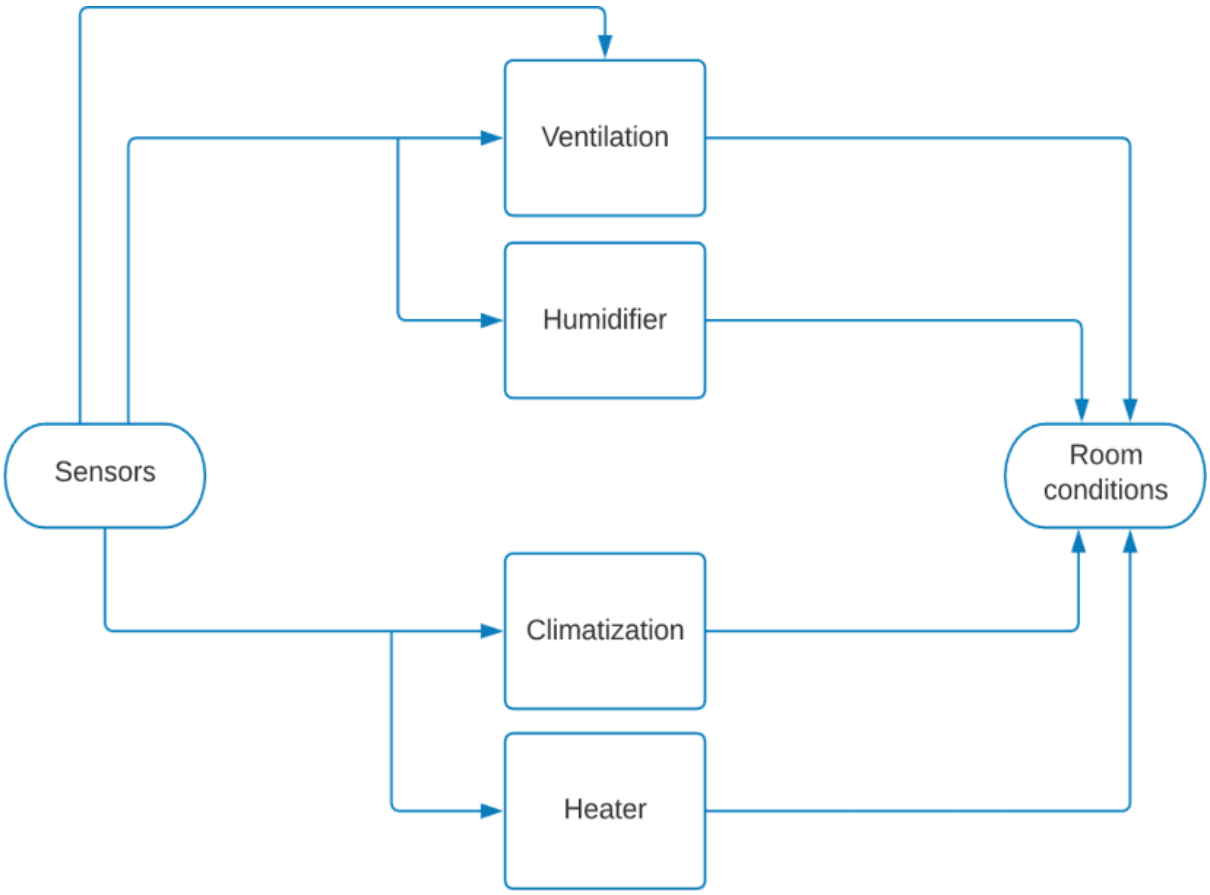
Display of the graph of distances on the Web server



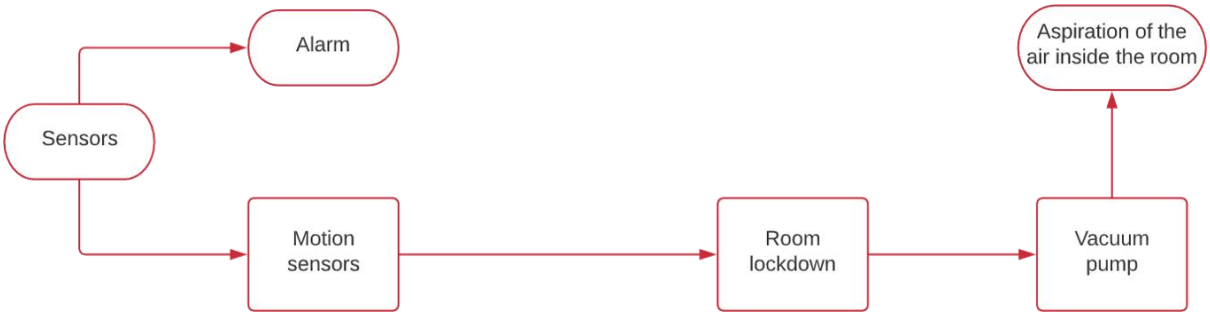
The LED before and after the presence of someone on the row

2) Books conservation and fire protection systems

a) Block Diagrams

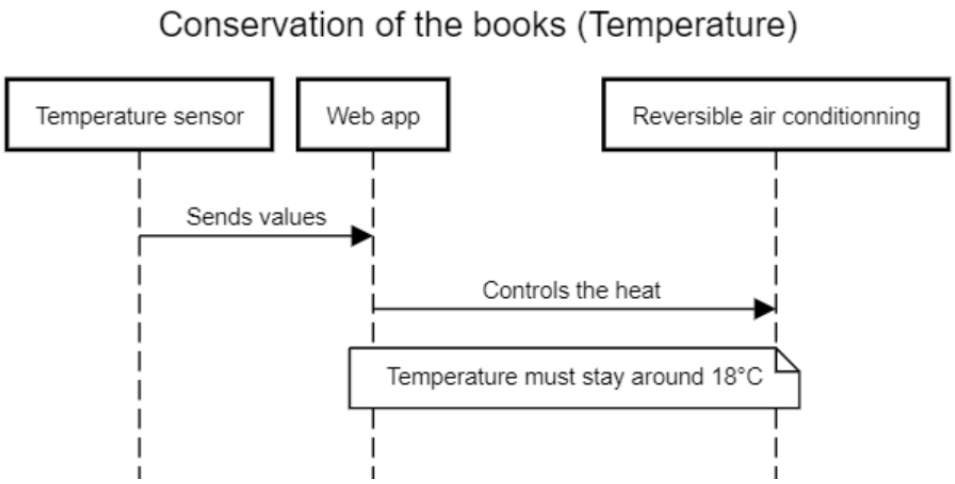


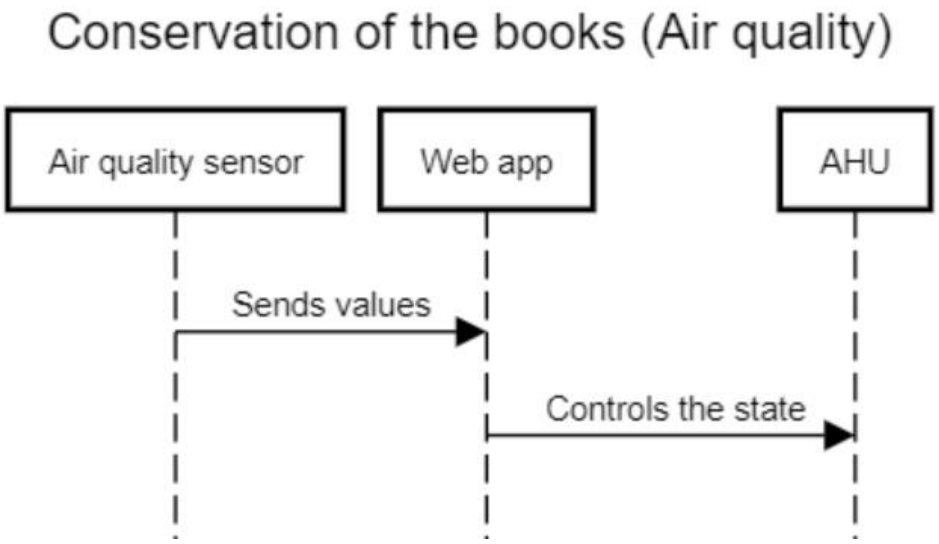
Book protection system: Different sensors (air quality, humidity, and temperature) are connected to the mechanisms (ventilation, humidifier, climatization, and heater) inside the room via the ESP32, which will automatically start and stop them once the values send by the sensors are back in the desired range. The ESP32 sends binary signals to the equipment.

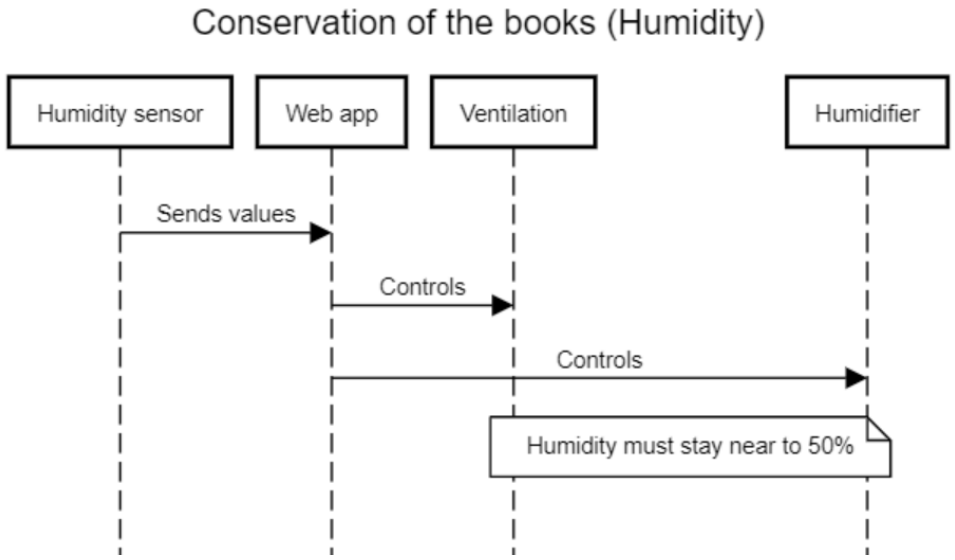


Fire protection system: The fire sensors are directly connected to the alarm. If they detect a start of fire, they will send an electric signal, which will make the alarm ring. They are also connected to the motion sensors, which will authorise the initiation of the room lockdown if they do not detect any presence inside the room (they send a binary message: 0 for nobody inside, 1 for people inside). Once the lockdown is initiated, it starts the vacuum pump and the process of aspiring the air outside the room.

b) Sequence Diagrams







c) Codes

Here is the code we used to create our WebApp with the information about the conditions inside the room and the simulation of the conditioning system, which must keep them steady. This code is an adaption of the one available on Random Nerd Tutorials[5].

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#include <WiFi.h>
#include "ESPAsyncWebServer.h"

const char* ssid = "Livebox-Adrien Cailly";
const char* password = "57806400EF283953E82E118669";

Adafruit_BME680 bme;

float temperature;
float humidity;
float pressure;
float gasResistance;

AsyncWebServer server(80);
AsyncEventSource events("/events");

unsigned long lastTime = 0;
unsigned long timerDelay = 3000;

void getBME680Readings(){

  unsigned long endTime = bme.beginReading();

  if (endTime == 0) {
    Serial.println(F("Failed to begin reading :("));
    return;
  }
  if (!bme.endReading()) {
    Serial.println(F("Failed to complete reading :("));
    return;
  }

  temperature = bme.temperature;
  pressure = bme.pressure / 100.0;
  humidity = bme.humidity;
  gasResistance = bme.gas_resistance / 1000.0;
}

String processor(const String& var){

  getBME680Readings();

  if(var == "TEMPERATURE"){
    return String(temperature);
  }
  else if(var == "HUMIDITY"){
    return String(humidity);
  }
  else if(var == "PRESSURE"){
    return String(pressure);
  }
 else if(var == "GAS"){
    return String(gasResistance);
```

```
  }
}

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <title>BME680 Web Server</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
  <link rel="icon" href="data:,">
  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    p {  font-size: 1.2rem;}
    body {  margin: 0;}
    .topnav { overflow: hidden; background-color: #4B1D3F; color: white;
font-size: 1.7rem; }
    .content { padding: 20px; }
    .card { background-color: white; box-shadow: 2px 2px 12px 1px
rgba(140,140,140,.5); }
    .cards { max-width: 700px; margin: 0 auto; display: grid; grid-gap:
2rem; grid-template-columns: repeat(auto-fit, minmax(300px, 1fr)); }
    .reading { font-size: 2.8rem; }
    .card.temperature { color: #0e7c7b; }
    .card.humidity { color: #17bebb; }
    .card.pressure { color: #3fca6b; }
    .card.gas { color: #d62246; }
    .card.fire { color: #fe1b00; }
  </style>
</head>
<body>
  <div class="topnav">
    <h3>Tableau de bord</h3>
  </div>
  <div class="content">
    <div class="cards">

      <div class="card temperature" id="chaud" style="display:none;">
        <h4><i class="fas fa-temperature-high"></i> TOO HOT</h4><p><span
class="reading"><span id="temp1">%TEMPERATURE%</span> &deg;C</span></p>
      </div>

      <div class="card temperature" id="temp_ok" style="display:none;">
        <h4><i class="fas fa-thermometer-half"></i> CORRECT</h4><p><span
class="reading"><span id="temp2">%TEMPERATURE%</span> &deg;C</span></p>
      </div>

      <div class="card temperature" id="froid" style="display:none;">
        <h4><i class="fas fa-temperature-low"></i> TOO COLD</h4><p><span
class="reading"><span id="temp3">%TEMPERATURE%</span> &deg;C</span></p>
      </div>

      <div class="card humidity" id="dry" style="display:none;">
```

```
        <h4><i class="fas fa-tint"></i> TOO DRY</h4><p><span
class="reading"><span id="hum1">%HUMIDITY%</span> &percnt;</span></p>
      </div>

      <div class="card humidity" id="humid_ok" style="display:none;">
        <h4><i class="fas fa-tint"></i> CORRECT</h4><p><span
class="reading"><span id="hum2">%HUMIDITY%</span> &percnt;</span></p>
      </div>

      <div class="card humidity" id="humid" style="display:none;">
        <h4><i class="fas fa-tint"></i> TOO HUMID</h4><p><span
class="reading"><span id="hum3">%HUMIDITY%</span> &percnt;</span></p>
      </div>

      <div class="card pressure">
        <h4><i class="fas fa-angle-double-down"></i> PRESSURE</h4><p><span
class="reading"><span id="pres">%PRESSURE%</span> hPa</span></p>
      </div>

      <div class="card gas" id="gas_ok" style="display:none;">
        <h4><i class="fas fa-wind"></i> GAS</h4><p><span
class="reading"><span id="gas1">%GAS%</span> K&ohm;</span></p>
      </div>

      <div class="card gas" id="vent" style="display:none;">
        <h4><i class="fas fa-exclamation-triangle"></i></i>
VENTILATE</h4><p><span class="reading"><span id="gas2">%GAS%</span>
K&ohm;</span></p>
      </div>

      <div class="card fire" id="onFire" style="display:none;">
        <h4><i class="fas fa-fire"></i> FIRE </h4><p><span
class="reading"><span id="fire">%FIRE%</span></span></p>
      </div>

    </div>
  </div>
<script>
if (!!window.EventSource) {
 var source = new EventSource('/events');

 source.addEventListener('open', function(e) {
  console.log("Events Connected");
 }, false);
 source.addEventListener('error', function(e) {
  if (e.target.readyState != EventSource.OPEN) {
    console.log("Events Disconnected");
  }
 }, false);

 source.addEventListener('message', function(e) {
  console.log("message", e.data);
 }, false);

 source.addEventListener('temperature', function(e) {
  console.log("temperature", e.data);
  var temp = parseFloat(e.data);
```

```
  if(temp>19){
  document.getElementById("temp1").innerHTML = e.data;
document.getElementById("chaud").style.display = 'block';
document.getElementById("temp_ok").style.display = 'none';
document.getElementById("froid").style.display = 'none'; }
  else if(temp<17){
  document.getElementById("temp3").innerHTML = e.data;
document.getElementById("chaud").style.display = 'none';
document.getElementById("temp_ok").style.display = 'none';
document.getElementById("froid").style.display = 'block';}
  else{
  document.getElementById("temp2").innerHTML = e.data;
document.getElementById("chaud").style.display = 'none';
document.getElementById("temp_ok").style.display = 'block';
document.getElementById("froid").style.display = 'none';}
 }, false);

 source.addEventListener('humidity', function(e) {
  console.log("humidity", e.data);
  var hum = parseFloat(e.data);
  if(hum>55){ document.getElementById("hum3").innerHTML = e.data;
document.getElementById("humid").style.display = 'block';
document.getElementById("humid_ok").style.display = 'none';
document.getElementById("dry").style.display = 'none'; }
  else if(hum<45){ document.getElementById("hum1").innerHTML = e.data;
document.getElementById("humid").style.display = 'none';
document.getElementById("humid_ok").style.display = 'none';
document.getElementById("dry").style.display = 'block';}
  else{ document.getElementById("hum2").innerHTML = e.data;
document.getElementById("humid").style.display = 'none';
document.getElementById("humid_ok").style.display = 'block';
document.getElementById("dry").style.display = 'none';}
 }, false);

 source.addEventListener('pressure', function(e) {
  console.log("pressure", e.data);
  document.getElementById("pres").innerHTML = e.data;
 }, false);

 source.addEventListener('gas', function(e) {
  console.log("gas", e.data);
  var gaz = parseFloat(e.data);
  if(50<gaz){
  document.getElementById("gas1").innerHTML = e.data;
document.getElementById("gas_ok").style.display = 'block';
document.getElementById("vent").style.display = 'none';}
  else{
  document.getElementById("gas2").innerHTML = e.data;
document.getElementById("gas_ok").style.display = 'none';
document.getElementById("vent").style.display = 'block';}
 }, false);

 source.addEventListener('fire', function(e) {
  console.log("fire", e.data);
  var fi = parseFloat(e.data);
  if(fi==1){ document.getElementById("onFire").style.display = 'block';}
  else{ document.getElementById("gas_ok").style.display = 'none';}
```

```
 }, false);
}
</script>
</body>
</html>)rawliteral";

const int LedChaud = 15;
const int LedFroid=4;
const int LedSec = 13;
const int LedHumide = 2;
const int LedPollue = 18;
const int LedFeu = 19;


void setup() {
  Serial.begin(115200);

  WiFi.mode(WIFI_AP_STA);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Setting as a Wi-Fi Station..");
  }
  Serial.print("Station IP Address: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  if (!bme.begin()) {
    Serial.println(F("Could not find a valid BME680 sensor, check
wiring!"));
    while (1);
  }
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150);

  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });

  events.onConnect([](AsyncEventSourceClient *client){
    if(client->lastId()){
      Serial.printf("Client reconnected! Last message ID that it got is:
%u\n", client->lastId());
    }
    client->send("hello!", NULL, millis(), 10000);
  });

  server.addHandler(&events);
  server.begin();

  pinMode(LedChaud, OUTPUT);
  pinMode(LedFroid, OUTPUT);
  pinMode(LedHumide, OUTPUT);
```

```
  pinMode(LedSec, OUTPUT);
  pinMode(LedPollue, OUTPUT);
  pinMode(LedFeu, OUTPUT);
}

void loop() {
  if ((millis() - lastTime) > timerDelay) {

    getBME680Readings();

    Serial.printf("Temperature = %.2f °C \n", temperature);
    Serial.printf("Humidity = %.2f % \n", humidity);
    Serial.printf("Pressure = %.2f hPa \n", pressure);
    Serial.printf("Gas Resistance = %.2f KOhm \n", gasResistance);
    Serial.println();

    events.send("ping",NULL,millis());
    events.send(String(temperature).c_str(),"temperature",millis());
    events.send(String(humidity).c_str(),"humidity",millis());
    events.send(String(pressure).c_str(),"pressure",millis());
    events.send(String(gasResistance).c_str(),"gas",millis());

    if(50>gasResistance&&temperature>60){
      events.send(String(1).c_str(),"fire",millis());
    }
    lastTime = millis();
  }

  if(temperature>19){
    digitalWrite(LedChaud, HIGH);
  }
  if(temperature<17){
    digitalWrite(LedFroid, HIGH);
  }
  if(humidity>55){
    digitalWrite(LedHumide, HIGH);
  }
  if(humidity<45){
    digitalWrite(LedSec, HIGH);
  }
  if(gasResistance<50){
    digitalWrite(LedPollue, HIGH);
  }
  if(gasResistance<50&&temperature>50){
    digitalWrite(LedFeu, HIGH);
  }
}
```

We are now going to explain each part of the code :

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#include <WiFi.h>
#include "ESPAsyncWebServer.h"
```

This first part is the one where we include all the libraries we need. “Wire.h”, “SPI.h”, “Adafruit_sensor.h”, “Adafruit_BME680.h” are the libraries we need to collect the data. The first two are for the communication protocols (Wire.h for I2C and SPI.h for SPI) and the two others are needed to interface with the sensor.

“WiFi.h” and “ESPAsyncWebServer.h” are the one we use to create and connect to the WebServer on which we send our data and create our dashboard.

```
const char* ssid = "Livebox-Adrien Cailly";
const char* password = "57806400EF283953E82E118669";
```

These two lines are the network credentials we use to connect the ESP32 to the WiFi.

```
Adafruit_BME680 bme;
```

Here we create an Adafruit_BME680 object called bme on the default ESP32 I2C pins.

```
float temperature;
float humidity;
float pressure;
float gasResistance;
```

These are the variables we will use to hold the BME680 readings.

```
AsyncWebServer server(80);
AsyncEventSource events("/events");
```

With these two lines, we create an asynchronous web server on port 80 and create a new event source on “/events”.

```
unsigned long lastTime = 0;
unsigned long timerDelay = 3000;
```

These variables will be used to update the sensor readings periodically. Here we chose to refresh the values every 3 seconds.

```
void getBME680Readings(){

  unsigned long endTime = bme.beginReading();

  if (endTime == 0) {
    Serial.println(F("Failed to begin reading :("));
    return;
  }
  if (!bme.endReading()) {
    Serial.println(F("Failed to complete reading :("));
    return;
  }

  temperature = bme.temperature;
  pressure = bme.pressure / 100.0;
  humidity = bme.humidity;
  gasResistance = bme.gas_resistance / 1000.0;
}
```

This function gets the BME680 readings and saves them on the relevant variables.

```
String processor(const String& var){

  getBME680Readings();

  if(var == "TEMPERATURE"){
    return String(temperature);
  }
  else if(var == "HUMIDITY"){
    return String(humidity);
  }
  else if(var == "PRESSURE"){
    return String(pressure);
  }
 else if(var == "GAS"){
    return String(gasResistance);
  }
}
```

This function replaces any placeholders on the HTML text used to build the web page with the current sensor readings.

```
<script>
if (!!window.EventSource) {
 var source = new EventSource('/events');

 source.addEventListener('open', function(e) {
  console.log("Events Connected");
 }, false);
 source.addEventListener('error', function(e) {
  if (e.target.readyState != EventSource.OPEN) {
    console.log("Events Disconnected");
  }
 }, false);

 source.addEventListener('message', function(e) {
  console.log("message", e.data);
 }, false);

 source.addEventListener('temperature', function(e) {
  console.log("temperature", e.data);
  var temp = parseFloat(e.data);
  if(temp>19){
  document.getElementById("temp1").innerHTML = e.data;
document.getElementById("chaud").style.display = 'block';
document.getElementById("temp_ok").style.display = 'none';
document.getElementById("froid").style.display = 'none'; }
  else if(temp<17){
  document.getElementById("temp3").innerHTML = e.data;
document.getElementById("chaud").style.display = 'none';
document.getElementById("temp_ok").style.display = 'none';
document.getElementById("froid").style.display = 'block';}
  else{
  document.getElementById("temp2").innerHTML = e.data;
document.getElementById("chaud").style.display = 'none';
```

```
document.getElementById("temp_ok").style.display = 'block';
document.getElementById("froid").style.display = 'none';}
 }, false);

 source.addEventListener('humidity', function(e) {
  console.log("humidity", e.data);
  var hum = parseFloat(e.data);
  if(hum>55){ document.getElementById("hum3").innerHTML = e.data;
document.getElementById("humid").style.display = 'block';
document.getElementById("humid_ok").style.display = 'none';
document.getElementById("dry").style.display = 'none'; }
  else if(hum<45){ document.getElementById("hum1").innerHTML = e.data;
document.getElementById("humid").style.display = 'none';
document.getElementById("humid_ok").style.display = 'none';
document.getElementById("dry").style.display = 'block';}
  else{ document.getElementById("hum2").innerHTML = e.data;
document.getElementById("humid").style.display = 'none';
document.getElementById("humid_ok").style.display = 'block';
document.getElementById("dry").style.display = 'none';}
 }, false);

 source.addEventListener('pressure', function(e) {
  console.log("pressure", e.data);
  document.getElementById("pres").innerHTML = e.data;
 }, false);

 source.addEventListener('gas', function(e) {
  console.log("gas", e.data);
  var gaz = parseFloat(e.data);
  if(50<gaz){
  document.getElementById("gas1").innerHTML = e.data;
document.getElementById("gas_ok").style.display = 'block';
document.getElementById("vent").style.display = 'none';}
  else{
  document.getElementById("gas2").innerHTML = e.data;
document.getElementById("gas_ok").style.display = 'none';
document.getElementById("vent").style.display = 'block';}
 }, false);

 source.addEventListener('fire', function(e) {
  console.log("fire", e.data);
  var fi = parseFloat(e.data);
  if(fi==1){ document.getElementById("onFire").style.display = 'block';}
  else{ document.getElementById("gas_ok").style.display = 'none';}
 }, false);
}
</script>
```

This is the HTML script we use to create the web page. It is the logic part of the page. This is the part that decides which data received goes in which placeholder depending on different conditions.

```
const int LedChaud = 15;
const int LedFroid=4;
const int LedSec = 13;
const int LedHumide = 2;
```

```
const int LedPollue = 18;
const int LedFeu = 19;
```

Here we assign a pin to each LED. Eventually, these pins are meant to be connected to the different systems inside the room (heater, climatization, ventilation, humidifier, and the vacuum pump).

```
void setup() {
  Serial.begin(115200);

  WiFi.mode(WIFI_AP_STA);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Setting as a Wi-Fi Station..");
  }
  Serial.print("Station IP Address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
```

In the setup, we begin by initialising the serial monitor. Then, we set the device as a station and soft access point simultaneously and connect it to the local network and print the ESP32 IP address.

```
  if (!bme.begin()) {
    Serial.println(F("Could not find a valid BME680 sensor, check
wiring!"));
    while (1);
  }

  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150); // 320*C for 150 ms
```

We then initialise the BME680 sensor.

```
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });
```

When you access the ESP32 IP address on the root "/" URL, this function sends the text that is stored on the *index_html* variable to build the web page and passes the processor as an argument, so that all placeholders are replaced with the latest sensor readings.

```
  // Handle Web Server Events
  events.onConnect([](AsyncEventSourceClient *client){
    if(client->lastId()){
      Serial.printf("Client reconnected! Last message ID that it got is:
%u\n", client->lastId());
    }
    // send event with message "hello!", id current millis
    // and set reconnect delay to 1 second
    client->send("hello!", NULL, millis(), 10000);
```

```
  });
  server.addHandler(&events);

  server.begin();
```

We set up the event source on the server and end up by starting the server.

```
  pinMode(LedChaud, OUTPUT);
  pinMode(LedFroid, OUTPUT);
  pinMode(LedHumide, OUTPUT);
  pinMode(LedSec, OUTPUT);
  pinMode(LedPollue, OUTPUT);
  pinMode(LedFeu, OUTPUT);
```

We set the different pins on the *OUTPUT* mode.

```
void loop() {
  if ((millis() - lastTime) > timerDelay) {

    getBME680Readings();
```

In the loop, we start by adding the time condition and getting the new sensor readings.

```
    Serial.printf("Temperature = %.2f °C \n", temperature);
    Serial.printf("Humidity = %.2f % \n", humidity);
    Serial.printf("Pressure = %.2f hPa \n", pressure);
    Serial.printf("Gas Resistance = %.2f KOhm \n", gasResistance);
    Serial.println();
```

We print the new values in the serial monitor.

```
    events.send("ping",NULL,millis());
    events.send(String(temperature).c_str(),"temperature",millis());
    events.send(String(humidity).c_str(),"humidity",millis());
    events.send(String(pressure).c_str(),"pressure",millis());
    events.send(String(gasResistance).c_str(),"gas",millis());

    if(50>gasResistance&&temperature>60){
      events.send(String(1).c_str(),"fire",millis());
    }
    lastTime = millis();
  }
```

Then, we send the events to the browser with the newest sensor readings to update the web page.

```
  if(temperature>19){
    digitalWrite(LedChaud, HIGH);
  }
  if(temperature<17){
    digitalWrite(LedFroid, HIGH);
  }
  if(humidity>55){
    digitalWrite(LedHumide, HIGH);
  }
  if(humidity<45){
```

```
    digitalWrite(LedSec, HIGH);
  }
  if(gasResistance<50){
    digitalWrite(LedPollue, HIGH);
  }
  if(gasResistance<50&&temperature>50){
    digitalWrite(LedFeu, HIGH);
  }
```

Finally, we activate the necessary LED considering the readings. This is the part simulating the systems inside the room.

d) Results

Here is what the WebApp looks like :

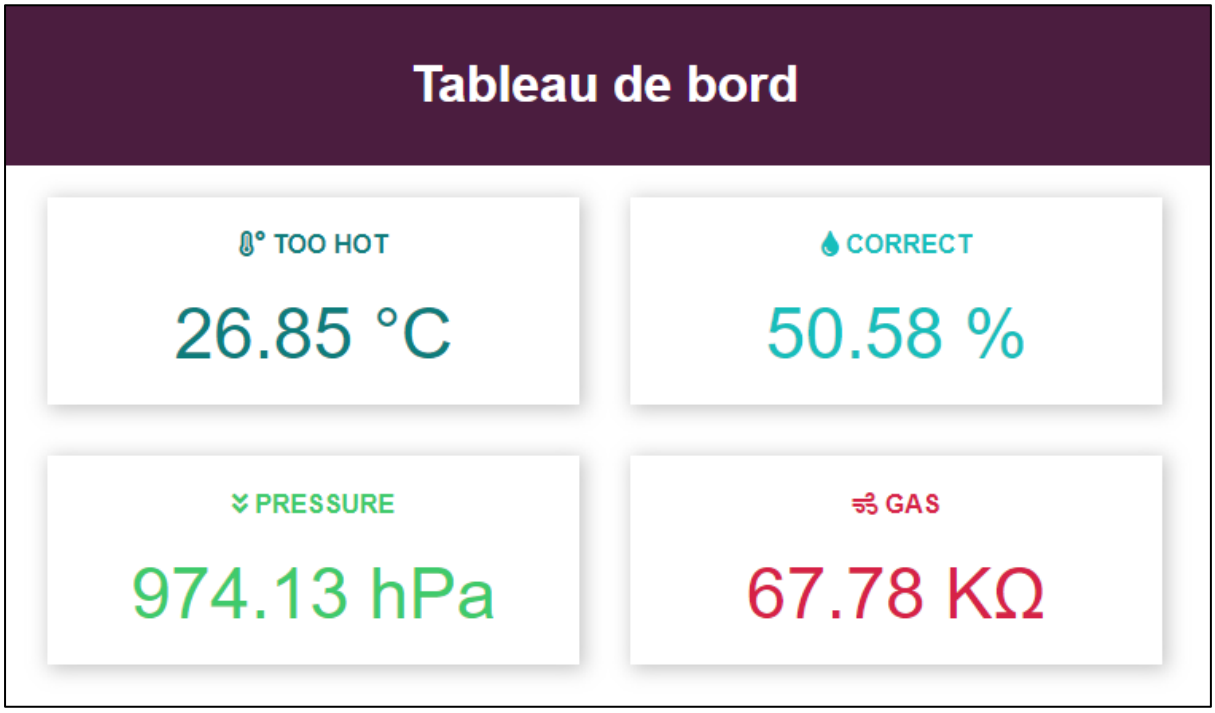


Figure 1 : Dashboard's display

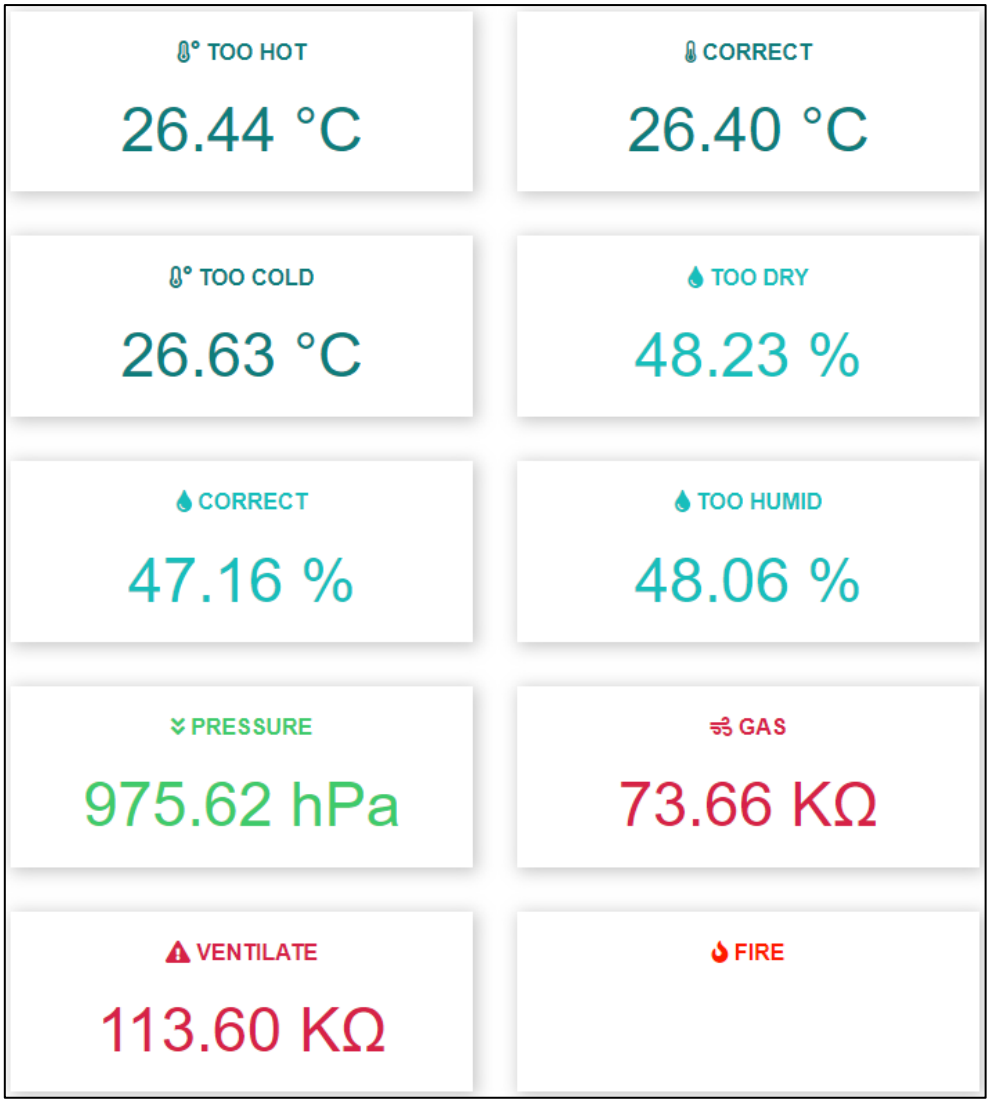


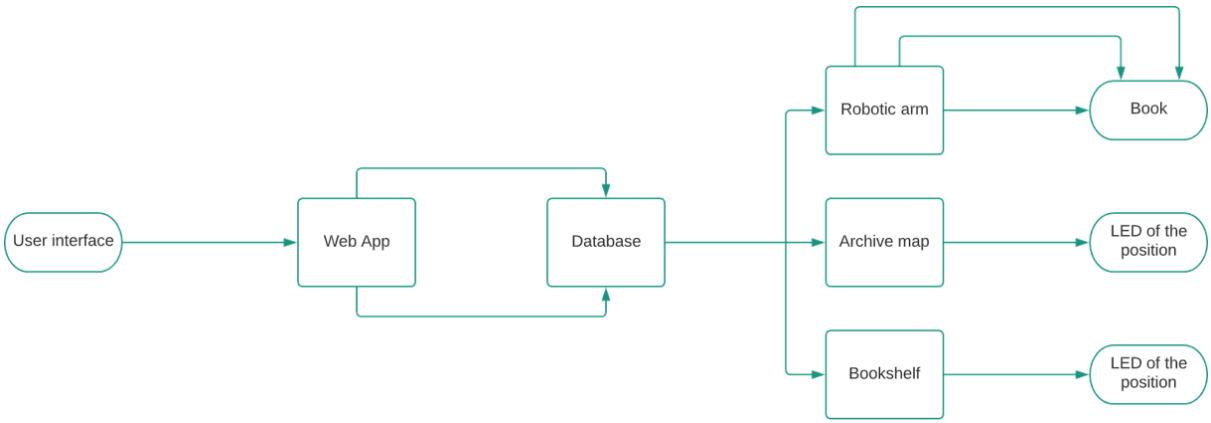
Figure 2 : The different displays of the dashboard

We can see that the dashboard has different displays, which alternates to indicate the actual values of conditions inside the room and inform the user whether they are correct or not.

The dashboard contains the values for the humidity the temperature and the air quality (gas), which are the most important ones to control the conditions inside the room. The value of the pressure is also displayed as it can be interesting to have it and because it can be used later to make the system more accurate. Finally, we have the card for the fire, which only appears when the conditions inside the room meet the one of a fire starts (high temperature and high proportion of gas in the air).

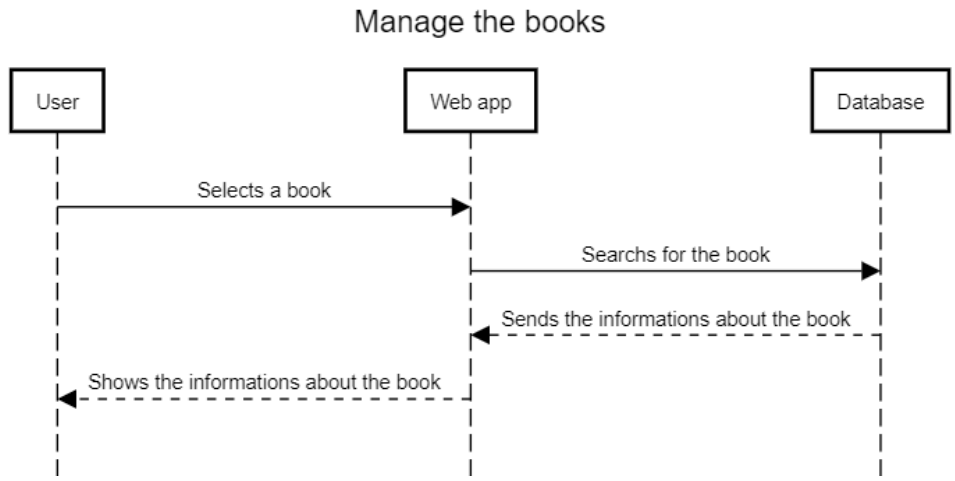
3) Books management system

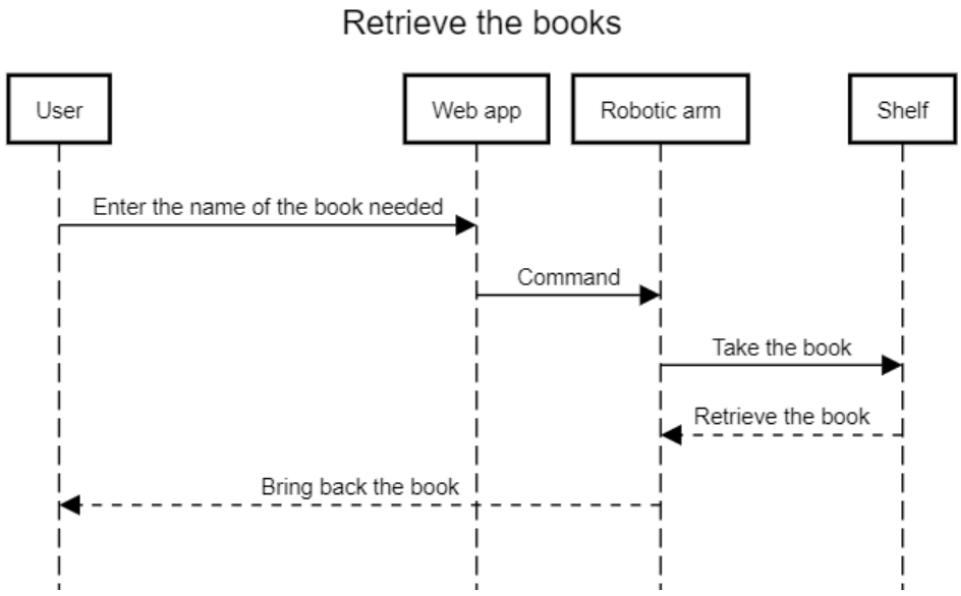
a) Block Diagram



Book management system: The user can either register or research a book in the Database via the web app. Then it's going to command many actions: light up a LED where there is the book chosen and command the robotic arm to retrieve, place or replace a book.

b) Sequence Diagrams





c) Code

Here is the code we used to create our Web Library with the information about the different books and the possibility to select one of them. This code is an adaption of the one available on Tommy Desrochers[6].

```
#include <WiFi.h>
#include <WebServer.h>

const char *ssid = "Livebox-Adrien Cailly";
const char *password = "57806400EF283953E82E118669";
WebServer server(80);

const int led1 = 19;
const int led2 = 18;
const int led3 = 2;
const int led4 = 4;

void handleRoot()
{
    String page = "<!DOCTYPE html>";

    page += "<html lang='fr'>";

    page += "<head>";
    page += "    <title>Serveur ESP32</title>";
    page += "    <meta http-equiv='refresh' content='60' name='viewport'
content='width=device-width, initial-scale=1' charset='UTF-8' />";
    page += "    <link rel='stylesheet'
href='https://www.w3schools.com/w3css/4/w3.css'>";
    page += "</head>";

    page += "<body>";
    page += "    <div class='w3-card w3-blue w3-padding-small w3-jumbo w3-
center'>";
    page += "        <p>Liste des archives présentes</p>";
```

```
    page += "    </div>";

    page += "    <div class='w3-bar'>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Titre</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponibilitée</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Sélection</a>";
    page += "    </div>";

    page += "    <div class='w3-bar'>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre1</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur1</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponible</a>";
    page += "        <a href='/1' class='w3-bar-item w3-button w3-border
w3-jumbo' style='width:25%;'>Sélectionner</a>";
    page += "    </div>";

    page += "    <div class='w3-bar'>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre2</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur2</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponible</a>";
    page += "        <a href='/2' class='w3-bar-item w3-button w3-border
w3-jumbo' style='width:25%;'>Sélectionner</a>";
    page += "    </div>";

    page += "    <div class='w3-bar'>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre3</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur2</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Indisponible</a>";
    page += "        <a href='/0' class='w3-bar-item w3-button w3-border
w3-jumbo' style='width:25%;'>Sélectionner</a>";
    page += "    </div>";

    page += "    <div class='w3-bar'>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre4</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur3</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponible</a>";
    page += "        <a href='/3' class='w3-bar-item w3-button w3-border
w3-jumbo' style='width:25%;'>Sélectionner</a>";
    page += "    </div>";
```

```
    page += "    <div class='w3-bar'>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre5</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur4</a>";
    page += "        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponible</a>";
    page += "        <a href='/4' class='w3-bar-item w3-button w3-border
w3-jumbo' style='width:25%;'>Sélectionner</a>";
    page += "    </div>";

    page += "</body>";

    page += "</html>";

    server.setContentLength(page.length());
    server.send(200, "text/html", page);
}

void handle1()
{
    digitalWrite(led1, HIGH);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    server.sendHeader("Location","/");
    server.send(303);
}
void handle2()
{
    digitalWrite(led1, LOW);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    server.sendHeader("Location","/");
    server.send(303);
}
void handle3()
{
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, LOW);
    server.sendHeader("Location","/");
    server.send(303);
}
void handle4()
{
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, HIGH);
    server.sendHeader("Location","/");
    server.send(303);
}
void handle0(){
    digitalWrite(led1, LOW);
```

```
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    server.sendHeader("Location","/");
    server.send(303);
}

void handleNotFound()
{
    server.send(404, "text/plain", "404: Not found");
}

void setup()
{
    Serial.begin(115200);
    delay(1000);
    Serial.println("\n");

    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    pinMode(led3, OUTPUT);
    pinMode(led4, OUTPUT);
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);

    WiFi.persistent(false);
    WiFi.begin(ssid, password);
    Serial.print("Tentative de connexion...");

    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(100);
    }

    Serial.println("\n");
    Serial.println("Connexion etablie!");
    Serial.print("Adresse IP: ");
    Serial.println(WiFi.localIP());

    server.on("/", handleRoot);
    server.on("/0", handle0);
    server.on("/1", handle1);
    server.on("/2", handle2);
    server.on("/3", handle3);
    server.on("/4", handle4);
    server.onNotFound(handleNotFound);
    server.begin();

    Serial.println("Serveur web actif!");
}

void loop()
{
    server.handleClient();
```

```
}
```

We are now going to explain each part of the code :

```
#include <WiFi.h>
#include <WebServer.h>
```

These are the two libraries we need to connect to the WiFi and create and use the web server.

```
const char *ssid = "Livebox-Adrien Cailly";
const char *password = "57806400EF283953E82E118669";

WebServer server(80);
```

These lines are the network credentials we use to connect the ESP32 to the WiFi and create a server on port 80.

```
const int led1 = 19;
const int led2 = 18;
const int led3 = 2;
const int led4 = 4;
```

We create constants for the different LED supposed to be on the map. We could have had another for the one on the bookshelf, but since the manipulation of LED is simple and we did not have any shelf, we choose not to add it.

```
void handleRoot()
{
    String page = "<!DOCTYPE html>";
        …
    page += "</html>";

    server.setContentLength(page.length());
    server.send(200, "text/html", page);
}
```

The function *handleRoot* contains the html of the page we create and sends it to the server to build it.

```
void handle1()
{
    digitalWrite(led1, HIGH);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    server.sendHeader("Location","/");
    server.send(303);
}
void handle2()
{
    digitalWrite(led1, LOW);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    server.sendHeader("Location","/");
```

```
    server.send(303);
}
void handle3()
{
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, LOW);
    server.sendHeader("Location","/");
    server.send(303);
}
void handle4()
{
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, HIGH);
    server.sendHeader("Location","/");
    server.send(303);
}
void handle0(){
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    server.sendHeader("Location","/");
    server.send(303);
}
```

The different *handle* functions define the action needed when the user presses one of the different buttons. The vision was that a first would look in the database for the region where the book is stored and then call the appropriate function. Here, we just have one function per book, each of them supposed to be in one of the different regions. The function *handle0* is called when the book is not available and returns nothing.

```
void handleNotFound()
{
    server.send(404, "text/plain", "404: Not found");
}
```

The function *handleNotFound* is called when a "/…" link is not found by the server.

```
void setup()
{
    Serial.begin(115200);
    delay(1000);
    Serial.println("\n");

    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    pinMode(led3, OUTPUT);
    pinMode(led4, OUTPUT);
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
```

We begin the setup by initialising the connection with the serial monitor, defining the modes of the different pins, and setting the LED to *LOW*.

```
    WiFi.persistent(false);
    WiFi.begin(ssid, password);
    Serial.print("Tentative de connexion...");

    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(100);
    }

    Serial.println("\n");
    Serial.println("Connexion etablie!");
    Serial.print("Adresse IP: ");
    Serial.println(WiFi.localIP());
```

We initialise the connection to the local network print the IP address in the serial monitor.

```
    server.on("/", handleRoot);
    server.on("/0", handle0);
    server.on("/1", handle1);
    server.on("/2", handle2);
    server.on("/3", handle3);
    server.on("/4", handle4);
    server.onNotFound(handleNotFound);
    server.begin();

    Serial.println("Serveur web actif!");
}
```

We define the functions to call depending on the button pressed and the link it was associated with. We finish by starting the server and printing it in the monitor.

```
void loop()
{
    server.handleClient();
}
```

In the *loop*, we just call the *handleClient* function, which will handle every action on the server by itself, allowing us not to worry about it.

d) Results

Here is what the Web library looks like :

Liste des archives présentes			
Titre	Auteur	Disponibilitée	Sélection
Livre1	Auteur1	Disponible	Sélectionner
Livre2	Auteur2	Disponible	Sélectionner
Livre3	Auteur2	Indisponible	Sélectionner
Livre4	Auteur3	Disponible	Sélectionner
Livre5	Auteur4	Disponible	Sélectionner

Figure 3 : Display of the Web library

As you can see, there is a list of different books, listed by title, author and availability. The column on the right contains buttons allowing you to select which book you want to retrieve from the archive. In the end, the library would use a real database and the aim would be to allow the user to add filters for theme, literary genre, or any other genre the user might need to use to easily find the books he needs.

When you select a book, the robotic arm will go grab it and bring it back to you. On addition, If the arm is dysfunctional or if you need to go in the room for any reason, a map will appear with an indicator where the book is supposed to be. Once where the book is supposed to be, a LED on its exact position on the shelf will indicate help you find it.

IV) Bibliography

[1] https://projetsdiy.fr/mesure-distance-ultrason-hc-sr04-arduino/

[2] https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/

[3] https://randomnerdtutorials.com/

[4] https://theiotprojects.com/esp8266-plot-sensor-readings-to-webserver-in-real-time-chart/

[5] Random Nerd Tutorials, *ESP32 Web Server with BME680 – Weather Station (Arduino IDE).* URL: https://randomnerdtutorials.com/esp32-bme680-web-server-arduino/

[6] Tommy Desrochers, *CONTRÔLEZ VOTRE ESP32 À PARTIR D'UNE PAGE WEB! (VERSION FACILE) [ESP32 ÉP#3].* URL : https://tommydesrochers.com/controlez-votre-esp32-a-partir-dune-page-web-version-facile-esp32-ep3/

[7] https://randomnerdtutorials.com/esp32-esp8266-plot-chart-web-server/

V) Annex

1) Access system HTML

```
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script src="https://code.highcharts.com/highcharts.js"></script>
  <style>
    body {
      min-width: 310px;
        max-width: 800px;
        height: 400px;
      margin: 0 auto;
    }
    h2 {
      font-family: Arial;
      font-size: 2.5rem;
      text-align: center;
    }
  </style>
</head>
<body>
  <h2>ESP8266 Distance Plot</h2>
  <div id="chart-distance" class="container"></div>
</body>
<script>
var chartT = new Highcharts.Chart({
  chart:{ renderTo : 'chart-distance' },
  title: { text: 'HC-SR04 Distance' },
  series: [{
    showInLegend: false,
    data: []
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#059e8a' }
  },
  xAxis: { type: 'datetime',
    dateTimeLabelFormats: { second: '%H:%M:%S' }
  },
  yAxis: {
    title: { text: 'Distance (CM)' }
  },
  credits: { enabled: false }
});
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var x = (new Date()).getTime(),
          y = parseFloat(this.responseText);
      //console.log(this.responseText);
      if(chartT.series[0].data.length > 40) {
        chartT.series[0].addPoint([x, y], true, true, true);
      } else {
        chartT.series[0].addPoint([x, y], true, false, true);
      }
    }
  };
  xhttp.open("GET", "/distance", true);
  xhttp.send();
}, 3000 ) ;
```

```
      }
    }
  };
  xhttp.open("GET", "/distance", true);
  xhttp.send();
}, 3000 ) ;
</script>
</html>
```

2) Books conservation and fire protection HTML

```
<!DOCTYPE HTML><html>
<head>
  <title>BME680 Web Server</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
  <link rel="icon" href="data:,">
  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    p {  font-size: 1.2rem;}
    body {  margin: 0;}
    .topnav { overflow: hidden; background-color: #4B1D3F; color: white;
font-size: 1.7rem; }
    .content { padding: 20px; }
    .card { background-color: white; box-shadow: 2px 2px 12px 1px
rgba(140,140,140,.5); }
    .cards { max-width: 700px; margin: 0 auto; display: grid; grid-gap:
2rem; grid-template-columns: repeat(auto-fit, minmax(300px, 1fr)); }
    .reading { font-size: 2.8rem; }
    .card.temperature { color: #0e7c7b; }
    .card.humidity { color: #17bebb; }
    .card.pressure { color: #3fca6b; }
    .card.gas { color: #d62246; }
    .card.fire { color: #fe1b00; }
  </style>
</head>
<body>
  <div class="topnav">
    <h3>Tableau de bord</h3>
  </div>
  <div class="content">
    <div class="cards">

      <div class="card temperature" id="chaud" style="display:none;">
        <h4><i class="fas fa-temperature-high"></i> TOO HOT</h4><p><span
class="reading"><span id="temp1">%TEMPERATURE%</span> &deg;C</span></p>
      </div>

      <div class="card temperature" id="temp_ok" style="display:none;">
        <h4><i class="fas fa-thermometer-half"></i> CORRECT</h4><p><span
class="reading"><span id="temp2">%TEMPERATURE%</span> &deg;C</span></p>
      </div>
```

```
      <div class="card temperature" id="froid" style="display:none;">
        <h4><i class="fas fa-temperature-low"></i> TOO COLD</h4><p><span
class="reading"><span id="temp3">%TEMPERATURE%</span> &deg;C</span></p>
      </div>

      <div class="card humidity" id="dry" style="display:none;">
        <h4><i class="fas fa-tint"></i> TOO DRY</h4><p><span
class="reading"><span id="hum1">%HUMIDITY%</span> &percnt;</span></p>
      </div>

      <div class="card humidity" id="humid_ok" style="display:none;">
        <h4><i class="fas fa-tint"></i> CORRECT</h4><p><span
class="reading"><span id="hum2">%HUMIDITY%</span> &percnt;</span></p>
      </div>

      <div class="card humidity" id="humid" style="display:none;">
        <h4><i class="fas fa-tint"></i> TOO HUMID</h4><p><span
class="reading"><span id="hum3">%HUMIDITY%</span> &percnt;</span></p>
      </div>

      <div class="card pressure">
        <h4><i class="fas fa-angle-double-down"></i> PRESSURE</h4><p><span
class="reading"><span id="pres">%PRESSURE%</span> hPa</span></p>
      </div>

      <div class="card gas" id="gas_ok" style="display:none;">
        <h4><i class="fas fa-wind"></i> GAS</h4><p><span
class="reading"><span id="gas1">%GAS%</span> K&ohm;</span></p>
      </div>

      <div class="card gas" id="vent" style="display:none;">
        <h4><i class="fas fa-exclamation-triangle"></i></i>
VENTILATE</h4><p><span class="reading"><span id="gas2">%GAS%</span>
K&ohm;</span></p>
      </div>

      <div class="card fire" id="onFire" style="display:none;">
        <h4><i class="fas fa-fire"></i> FIRE </h4><p><span
class="reading"><span id="fire">%FIRE%</span></span></p>
      </div>

    </div>
  </div>
<script>
if (!!window.EventSource) {
 var source = new EventSource('/events');

 source.addEventListener('open', function(e) {
  console.log("Events Connected");
 }, false);
 source.addEventListener('error', function(e) {
  if (e.target.readyState != EventSource.OPEN) {
    console.log("Events Disconnected");
  }
 }, false);
```

```
 source.addEventListener('message', function(e) {
  console.log("message", e.data);
 }, false);

 source.addEventListener('temperature', function(e) {
  console.log("temperature", e.data);
  var temp = parseFloat(e.data);
  if(temp>19){
  document.getElementById("temp1").innerHTML = e.data;
document.getElementById("chaud").style.display = 'block';
document.getElementById("temp_ok").style.display = 'none';
document.getElementById("froid").style.display = 'none'; }
  else if(temp<17){
  document.getElementById("temp3").innerHTML = e.data;
document.getElementById("chaud").style.display = 'none';
document.getElementById("temp_ok").style.display = 'none';
document.getElementById("froid").style.display = 'block';}
  else{
  document.getElementById("temp2").innerHTML = e.data;
document.getElementById("chaud").style.display = 'none';
document.getElementById("temp_ok").style.display = 'block';
document.getElementById("froid").style.display = 'none';}
 }, false);

 source.addEventListener('humidity', function(e) {
  console.log("humidity", e.data);
  var hum = parseFloat(e.data);
  if(hum>55){ document.getElementById("hum3").innerHTML = e.data;
document.getElementById("humid").style.display = 'block';
document.getElementById("humid_ok").style.display = 'none';
document.getElementById("dry").style.display = 'none'; }
  else if(hum<45){ document.getElementById("hum1").innerHTML = e.data;
document.getElementById("humid").style.display = 'none';
document.getElementById("humid_ok").style.display = 'none';
document.getElementById("dry").style.display = 'block';}
  else{ document.getElementById("hum2").innerHTML = e.data;
document.getElementById("humid").style.display = 'none';
document.getElementById("humid_ok").style.display = 'block';
document.getElementById("dry").style.display = 'none';}
 }, false);

 source.addEventListener('pressure', function(e) {
  console.log("pressure", e.data);
  document.getElementById("pres").innerHTML = e.data;
 }, false);

 source.addEventListener('gas', function(e) {
  console.log("gas", e.data);
  var gaz = parseFloat(e.data);
  if(50<gaz){
  document.getElementById("gas1").innerHTML = e.data;
document.getElementById("gas_ok").style.display = 'block';
document.getElementById("vent").style.display = 'none';}
  else{
  document.getElementById("gas2").innerHTML = e.data;
document.getElementById("gas_ok").style.display = 'none';
document.getElementById("vent").style.display = 'block';}
```

```
 }, false);

 source.addEventListener('fire', function(e) {
  console.log("fire", e.data);
  var fi = parseFloat(e.data);
  if(fi==1){ document.getElementById("onFire").style.display = 'block';}
  else{ document.getElementById("gas_ok").style.display = 'none';}
 }, false);
}
</script>
</body>
</html>
```

3) Books management system HTML

```
<!DOCTYPE html>

<html lang='fr'>

<head>
    <title>Serveur ESP32</title>
    <meta http-equiv='refresh' content='60' name='viewport'
content='width=device-width, initial-scale=1' charset='UTF-8' />
    <link rel='stylesheet' href='https://www.w3schools.com/w3css/4/w3.css'>
</head>

<body>
    <div class='w3-card w3-blue w3-padding-small w3-jumbo w3-center'>
        <p>Liste des archives présentes</p>
    </div>

    <div class='w3-bar'>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Titre</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponibilitée</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Sélection</a>
    </div>

    <div class='w3-bar'>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre1</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur1</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponible</a>
        <a href='/1' class='w3-bar-item w3-button w3-border w3-jumbo'
style='width:25%;'>Sélectionner</a>
    </div>

    <div class='w3-bar'>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre2</a>
```

```
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur2</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponible</a>
        <a href='/2' class='w3-bar-item w3-button w3-border w3-jumbo'
style='width:25%;'>Sélectionner</a>
    </div>

    <div class='w3-bar'>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre3</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur2</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Indisponible</a>
        <a href='/0' class='w3-bar-item w3-button w3-border w3-jumbo'
style='width:25%;'>Sélectionner</a>
    </div>

    <div class='w3-bar'>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre4</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur3</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponible</a>
        <a href='/3' class='w3-bar-item w3-button w3-border w3-jumbo'
style='width:25%;'>Sélectionner</a>
    </div>

    <div class='w3-bar'>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Livre5</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Auteur4</a>
        <a class='w3-bar-item w3-border w3-jumbo'
style='width:25%;'>Disponible</a>
        <a href='/4' class='w3-bar-item w3-button w3-border w3-jumbo'
style='width:25%;'>Sélectionner</a>
    </div>

</body>

</html>
```

Github Link : https://github.com/Kvothearlen/ArchivCO