Livrable K : Manuel d'utilisateur

Rapport présenté à

Monsieur Emmanuel Bouendeu

dans le cadre du cours GNG 1503 C-01:

Génie de la conception

Par

Joanie Duguay

Chloé Monin

Lu-Yanne Trottier

Jaâfar Ziha

Université d'Ottawa

Le 5 avril 2020

SOMMAIRE

Le 20 janvier 2020, un mandat a été confié à un groupe d'étudiants de l'Université d'Ottawa, coordonné par Justine Boudreau, concernant un problème dans le STEM. Il est primordial pour l'équipe de concevoir un système automatisé pour réaliser ce mandat. Le groupe a trouvé une lacune dans le local Brunsfield, situé au deuxième étage du bâtiment. Suite à la consultation de plusieurs documents et d'utilisateurs, les étudiants ont identifié l'origine du conflit. Certaines machines ont une intensité sonore qui va au-delà des normes prescrites par CCHST. Rien n'avertit les utilisateurs des dommages que peut causer un tel niveau de bruit sans protection.

Dans le même ordre d'idées, l'équipe a exploré les solutions possibles pour résoudre ce problème afin de concevoir le dispositif automatisé demandé. En effet, parmi toutes les solutions trouvées, les étudiants ont retenu un appareil de mesure du niveau sonore. Ce rapport projettera le processus et les résultats que le groupe a obtenu tout au long de la session.

TABLE DES MATIÈRES

LISTE DE FIGURES

LISTE DE TABLES

LISTE D'ACRONYMES

Acronyme	Définition
STEM	Science, Technology, Engineering, and Mathematics
DEL/LED	Diode Électro Luminescente
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
CCHST	Centre canadien d'hygiène et de sécurité au travail
CGEC	Centre en génie entrepreneurial de la conception
OTA	Over The Air
MS Project	Microsoft Project
LCD	Liquid Crystal Display
3D	Trois dimensions ou tridimensionnel
IDE	Integrated Development Environment

1 INTRODUCTION

1.1 SOUS-TITRE

Ce sous-titre sert de séparer la section en plusieurs catégories, pour mieux diviser le travail.

1.1.1 Sous-sous-titre

Ce sous-sous-titre sera utilisé dans le document pour délimiter une section qui comprend plusieurs thèmes ou parties. Il n'y aura pas de sous-sous-sous-titre utilisés dans ce document, donc celui-ci est le dernier que nous allons utiliser.

1.2 QUOI

Dans le cadre de notre projet, le problème adressé par notre produit est celui du niveau sonore trop élevé de certains espaces du STEM, comme le centre Brunsfield, ce qui endommage l'ouïe des clients et des utilisateurs. En effet, les espaces ne sont pas pourvus d'un affichage indiquant aux personnes présentes l'état sonore du milieu et aucun dispositif existe, afin d'avertir les utilisateurs du danger d'exposition à certains niveaux de bruits, ce qui pourrait causer des problèmes auditifs chez certains utilisateurs, à long terme.

Nous considérons que ce problème est important, puisque dans ces espaces, il peut être très dangereux d'y rester trop longtemps, sans la protection nécessaire. Nous croyons que les locaux devraient être équipé de moyens efficaces pour afficher le niveau sonore d'une pièce, puis indiquer le temps d'exposition maximale à celui-ci. C'est donc pour ces raisons que nous jugeons que le problème est important et que nous devons y remédier

1.3 QUI

Les utilisateurs ont exprimé plusieurs besoins fondamentaux que nous avons pris en compte, lors de l'élaboration de notre produit. Le besoin le plus important était la sécurité des utilisateurs de l'espace Les utilisateurs avaient aussi besoin d'un système qui améliore la gestion des espaces, afin d'augmenter la qualité de travail dans ceux-ci. Le système devait également être visible pour que tous les utilisateurs soient sensibilisés à la question de l'exposition au bruit.

1.4 POURQUOI NOUS

Notre équipe se démarque des autres, puisque nous nous sommes concentrés sur un problème qui touchait la santé et la sécurité des utilisateurs et des clients. Parmi tous les besoins des utilisateurs, nous avons ciblé les plus importants et avons travaillé fort pour développer une solution répondant à ceux-ci, mais aussi à d'autres besoins qui étaient secondaires. Cela démontre que notre produit va au-delà des attentes et règle plusieurs failles du STEM, en augmentant l'automatisation de ce bâtiment et en facilitant le travail des employés et gérants. Ce produit constitue une solution complète, qui répond à de multiples besoins. C'est de cette façon que nous voulons nous distinguer des autres équipes et que nous croyons que notre produit devrait être adopté dans les espaces du STEM.

1.5 NOTRE PRODUIT

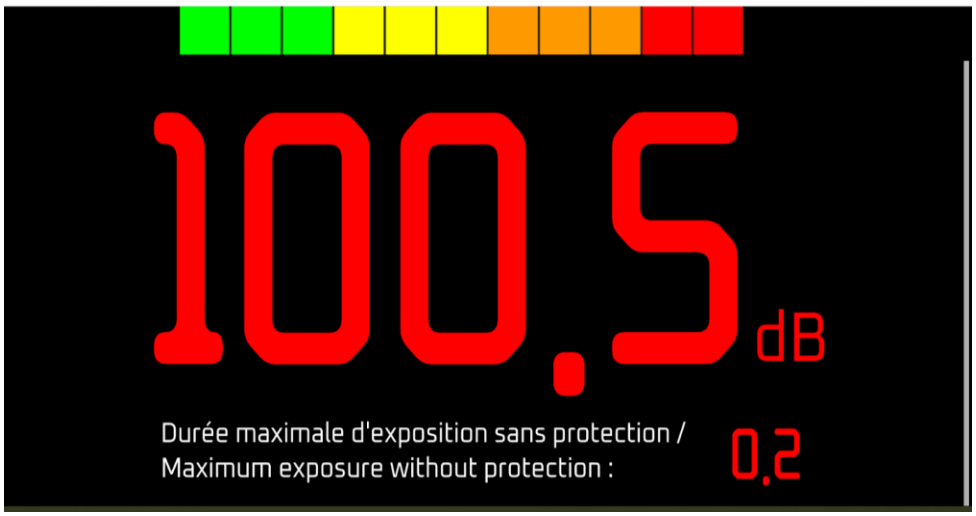

Figure 1: Photo de la page HTML, correspondant à notre produit final

Notre produit consiste en un affichage numérique rectangulaire, qui serait mis en place dans l'espace Brunsfield avec le principal but d'afficher le niveau de décibels ainsi que la durée maximale d'exposition au bruit sans protection. L'interface affiche le niveau sonore en décibel au centre, le temps d'exposition en bas, ainsi qu'une échelle de couleur qui correspond au niveau sonore de la pièce et qui permet à l'utilisateur de mieux prendre conscience du niveau de risque. L'affichage est connecté à un circuit qui contient un capteur, afin d'analyser le niveau sonore de la pièce, puis envoie les données au NodeMCU, qui lui les partage à la page HTML. Ce circuit est entreposé dans

un boîtier en acrylique, découpé au laser. Le produit serait relié à une alarme visuelle avertissant les utilisateurs que la durée maximale d'exposition est dépassée. Cela leur indique alors qu'ils doivent arrêter temporairement leurs activités ou qu'ils doivent se protéger à l'aide de casques ou de bouchons. Ce système est par conséquent préventif et vise à assurer le respect des normes du CCHST dans les espaces du STEM.

2 COMMENT LE PROTOTYPE EST CONSTRUIT

2.1 ASPECT PHYSIQUE

2.1.1 LDM (Liste des Matériaux)

Table1: Liste des matériaux des composantes physiques

Numéro	Matériel	Quantité
9	Acrylique	8000cm3
10	Boulons	24
11	Colle chaude	N/A
12	Écrous	24

Acrylique: le verre acrylique est un bon choix pour notre prototype puisque c'est un matériau qui est durable, facile à découper au laser et peu coûteux en comparaison aux autre alternatives transparentes.

Boulon: les boulons sont un bon choix puisque ce sont des éléments de fixation très durables à un prix minime.

Colle chaude: pour sceller le tout, rien de mieux que la colle chaude, que l'on retrouve très facilement dans tout atelier respectable, elle aide à garder toute particule indésirable hors du boîtier.

Écrous: les écrous ont été choisis pour les mêmes raisons que les boulons.

2.1.2 Liste d'équipements

1. Machine de découpe laser
2. Fraiseuse
3. Fusil à colle chaude

2.1.3 Instructions

1. Dessiner le boîtier de dimension 20cm x 15cm x 5cm ci-dessous dans le logiciel de dessin vectoriel Inskape

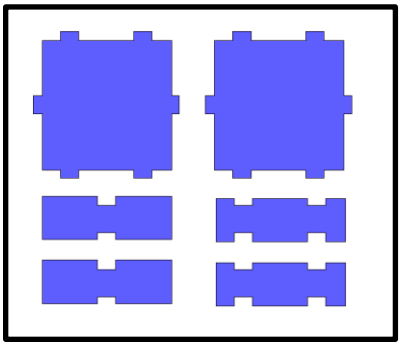

Figure 2: Photo du boitier en PDF

2. Découper à la machine de découpe laser les 6 faces du boîtier dans l'acrylique
3. Percer à la fraiseuse aux 4 extrémités des 6 faces des trous de diamètre 3/8 pouce

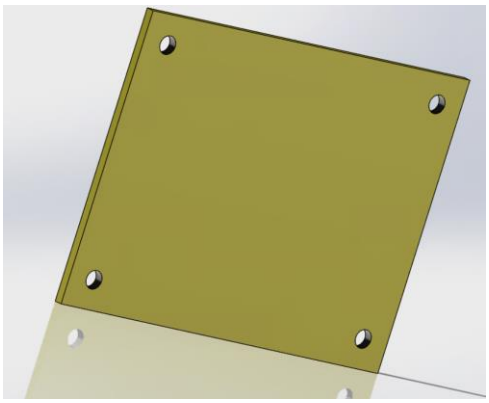

Figure 3: Perçage en SolidWorks

4. Percer à la fraiseuse sur l'une des faces latérales un trou de diamètre 1/16 pouce

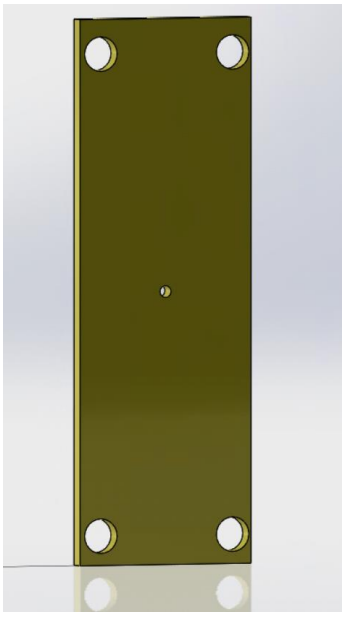

Figure 4: Perçage en SolidWorks d'un côté latérale

5. Fixer sur la face arrière le « soldering board »

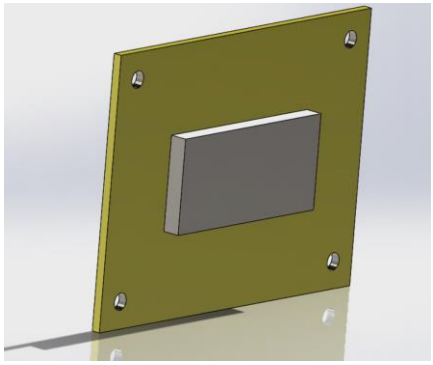

Figure 5: Fixage de la tablette en SolidWorks

6. Faire passer les fils d'alimentation dans le trou de la face latérale
7. Assembler le boîtier en installant les boulons

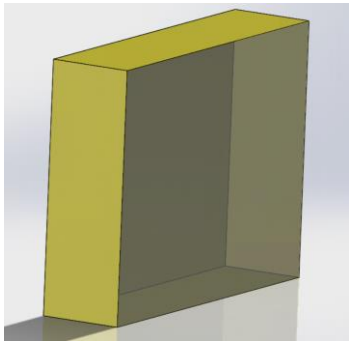

Figure 6: Assemblage du boitier en SolidWorks

2.2 ASPECT ÉLECTRIQUE

2.2.1 LDM (Liste des Matériaux)

Table 2: Liste des matériaux des composantes électriques

Numéro	Matériel	Quantité
2	Wires	10
4	Soldering board	1
6	Capteur sonore	1
8	Diodes LED	18

2.2.2 Liste d'équipements

1. Fer à souder

2. Station de soudage
3. Fil de zinc
4. Éponge humide

2.2.3 Instructions

1. Souder le Node MCU à la plaque de soudage
2. Souder des fils électriques aux pins digitales D0, D1, D2, D3 et D4
3. Souder l'extrémité de ces fils de manière à ce que 3 diodes luminescentes soient liées en série à chaque pin
4. Souder le capteur sonore à la plaque de soudage
5. Souder un fil électrique afin de lier la pin G du capteur et à la pin GND du Node
6. Souder un fil électrique afin de lier la pin A0 du capteur à celle du Node
7. Souder un fil électrique afin de lier la pin + du capteur à la pin 3V3 du Node
8. Brancher le fil d'alimentation micro USB au Node MCU et connecter l'extrémité à un chargeur mural

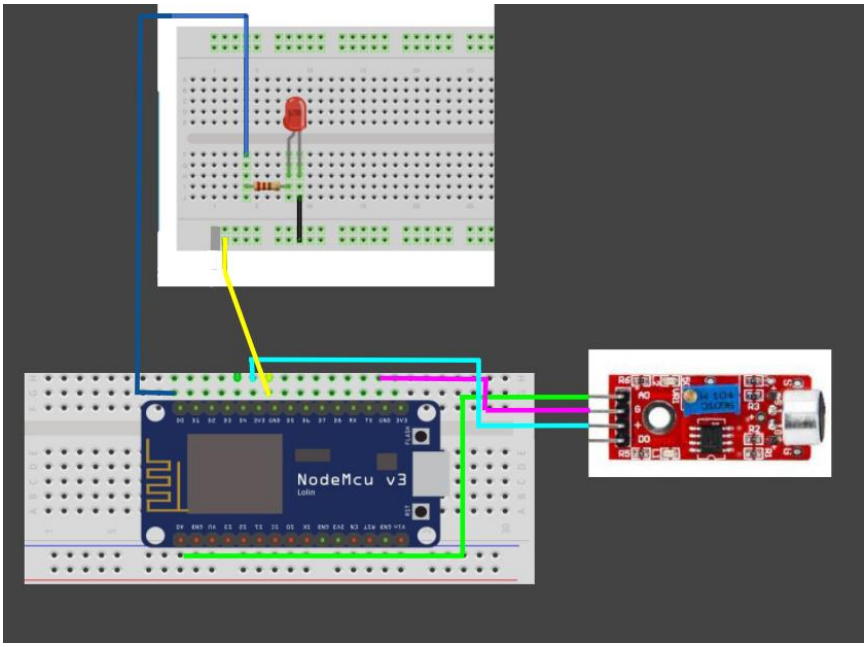

Figure 7: Circuit électronique avec exemple d'une diode connectée

2.3 ASPECT ÉLECTRIQUE

2.3.1 LDM (Liste des Matériaux)

1. Avoir les applications nécessaire sur l'ordinateur (Arduino, JavaScript, CSS et HTML)

2.3.2 Liste d'équipements

Voir les annexes pour les codes du produit

2.3.3 Instructions

2. Dans la barre de recherche Google, taper "www.esp8266.local"
3. En cas de problème informatique, modifier le code du
    a. Fichier CSS s'il est lié à l'apparence du produit
    b. Fichier JavaScript s'il est lié à l'apparition des bandes de couleur, l'affichage des données du capteurs ou du temps d'exposition
    c. Fichier main contrôlant le NodeMCU s'il est lié au lancement de l'interface ou si les changements précédents échouent
4. Dans l'IDE Arduino, sélectionner Tools->Port, puis "esp8266-xxxx"
5. Télécharger le nouveau sketch

3 COMMENT UTILISER LE PROTOTYPE

Le principal objectif du dispositif est la mesure du niveau sonore qui se fait par le capteur sonore recueillant les données qui seront par la suite traitées par le microcontrôleur NodeMCU. Le microcontrôleur est ensuite responsable de générer l'affichage, qui montre le nombre de décibels capté par le senseur sonore ainsi que le temps restant d'exposition sécuritaire. Le NodeMCU contrôle également une échelle de couleur composée de DELs qui montre si le niveau sonore est faible ou élevé si certains n'ont pas la notion de décibels. Lorsque le niveau sonore dépasse les limites et devient trop élevé, un interrupteur s'active et laisse passer le courant qui va alimenter une alarme visuelle (comme un gyrophare) qui va avertir d'arrêter la source de bruit. Le microcontrôleur est également connecté via wifi à un site internet qui affiche les mêmes données que l'affichage, soit le niveau de décibels, en temps réel, une échelle de couleur ainsi que le temps d'exposition restant.

Figure 8: NodeMCU

Figure 9: DELs

L'utilisation du prototype se fait de manière sécuritaire s'il est bien installé puisqu'il ne faut que regarder le boîtier de loin pour voir les données. Il n'y a donc aucun danger. Même pour le site web, le tout se fait automatiquement il suffit d'interpréter l'information.

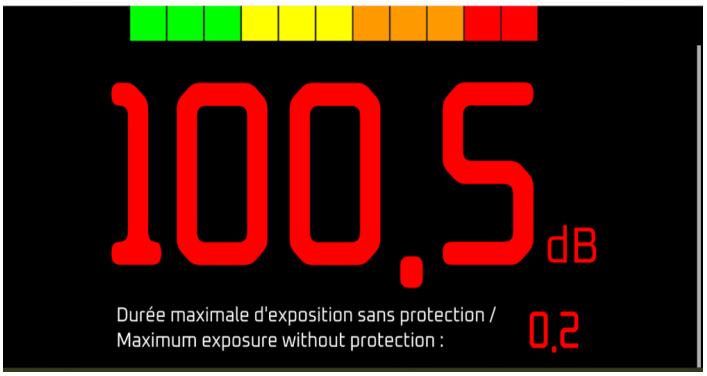

Figure 1: Photo de la page HTML, correspondant à notre produit final

Pour installer le prototype en toute sécurité, il suffit de placer le boîtier, ainsi que l'alarme à proximité d'une source de bruit, à proximité d'une prise murale et en hauteur sur un mur. Il ne faut que brancher le tout pour qu'il commence à fonctionner de façon appropriée. Le prototype ne comporte pas de danger à l'utilisateur si l'installation est faite tel que prévu.

4 COMMENT MAINTENIR LE PROTOTYPE

Il faut s'assurer, dès le début, que les éléments du circuit soient calibrés pour qu'il n'y ait pas de problèmes rencontrés pendant les premières utilisations. Il est évident, selon les données récoltées, que la calibration initiale des éléments, comme le capteur sonore, va leur assurer durabilité et fiabilité à long terme.

Malgré la fiabilité accrue du prototype, il serait important de régulièrement tester, avec son téléphone intelligent, si les décibels affichés par le prototype sont bien ceux de l'environnement. Les nouveaux téléphones intelligents ont des micros de qualité que l'on peut utiliser pour s'assurer que le capteur sonore du prototype n'est pas défectueux. Il faudrait aussi l'éteindre et le rallumer de temps en temps afin qu'il se reconnecte avec le site internet puisqu'à long terme, il y aura un délai qui fera en sorte que les données ne seront plus mises à jour en temps réel.

Si le prototype est souvent déplacé, il faudrait vérifier si les soudures du circuit sont toujours en bon état. Les pièces les plus susceptibles de se briser sont le capteur sonore et l'interrupteur, il faudrait donc en avoir de rechange au cas où il y aurait un bris.

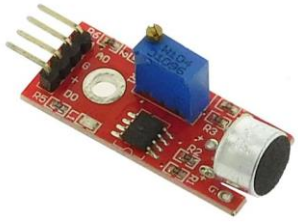

Figure 10: Capteur sonore

Figure 11: Interrupteur / Relais

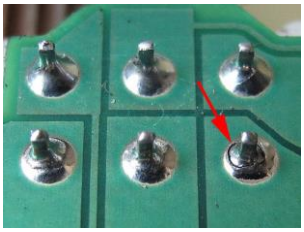

Figure 12: Soudure brisé

5 CONCLUSIONS ET RECOMMANDATIONS POUR LES TRAVAUX FUTURS

En conclusion, l'équipe a beaucoup appris tout au long de la session. Il y a eu plusieurs épreuves qui ont permis au groupe de développer de nouvelles compétences et de parfaire ses connaissances. Nous avons appris à communiquer de façon neutre, clair et concise afin de pouvoir faire des rétroactions autant positives que négatives sans brusquer personne. Lors du prototype 1, l'équipe a appris qu'il fallait faire dans la simplicité pour finir la réalisation sans embûche. Par ailleurs, pour cette étape de la conception, il aurait fallu déjà avoir fait la programmation pour le circuit et avoir commencé le codage pour le site internet. Pour ce qui est de la programmation, il aurait été préférable de la commencer dès le début puisque l'équipe a eu de la difficulté à accomplir cette étape. Il fallait apprendre de nouveaux langages informatiques tels que Java Script, CSS et HTML. De plus, il aurait fallu plus d'une personne pour faire cette tâche, de même qu'il aurait été mieux d'avoir déjà commandé les composantes pour le deuxième prototype. En concevant le prototype 2, le groupe a appris que la communication entre chacun était primordiale, ainsi que de donner des mises à jour de ce que chacun fait. Il est important de savoir que lors de cette étape le circuit et les essais devaient être réalisés. Donc, par le fait même, le codage devait être sans bogues. Pour le prototype 3, il aurait fallu être arrivé à l'étape d'améliorer le prototype esthétiquement. Nous avons réalisé que pour construire le boitier, il était plus simple de le fabriquer par découpe laser des composantes que par impression 3D. En effet, il y a beaucoup plus d'avantages de le modeler de cette façon, tels que le temps et la composition de matériel.

En résumé, les leçons que l'équipe a apprises sont d'adopter une communication neutre, claire et concise, de réaliser une conception simple et facile afin d'être en mesure de le terminer dans les délais prescrits et d'assurer une division équitable des tâches entre tous les membres. Nos recommandations pour les travaux futurs seraient de mieux s'organiser lors de chacune des étapes, de faire des recherches plus approfondies sur le sujet et de réaliser un dispositif plus simple.

6 RÉFÉRENCES

MÉDIAGRAPHIE

CCHST(2020), Réponse SST, Agent physique, “Limite d’exposition au bruit au Canada”, Récupéré sur le site du Gouvernement du Canada: https://www.cchst.ca/oshanswers/phys_agents/exposure_can.html

Pieter P.(2017), “A Beginner's Guide to the ESP8266”, Récupéré sur le site d’Arduino: https://tttapa.github.io/ESP8266/Chap01%20-%20ESP8266.html

Quinn S.(2017), Arduino, “Using ESP8266 Over the air programming in Arduino IDE”, Récupéré sur le site de instructable circuit : https://www.instructables.com/id/Using-ESP8266-SPIFFS/

A.(2020), Arduino, “Using ESP8266 SPIFFS”, Récupéré sur le site de Last minute engineer circuit : https://lastminuteengineers.com/esp8266-ota-updates-arduino-ide/

FIGURES

Figure 8 | NodeMCU ,
https://imgaz.staticbg.com/thumb/large/oaupload/ser1/banggood/images/C2/29/95ffaac5-a811-4764-b724-631504e0d257.jpg

Figure 9 | DELS
https://www.abavala.com/wp-content/uploads/LED-vert-jaune-rouge.png

Figure 10 | Capteur Sonore
https://www.gotronic.fr/ori-capteur-sonore-gt1146-26144.jpg

Figure 11 | Interrupteur / Relais
https://www.robotshop.com/media/catalog/product/cache/image/400x400/9df78eab33525d08d6e5fb8d27136e95/a/r/arduino-compatible-5v-relay-module-vel.jpg

Figure 12 | Soudure Brisée
https://lh3.googleusercontent.com/proxy/Gdf9tsnTZ4B3QMP_YmeHzP4LNjqgX6kV27bWHBPsCGeDnQHWV2LOw_f7pbv0nLczMLEPkoAdecoVcFgB198WZBunrwJ9TbeDJUoZoKbK5iHxIq1Fq9iKn0xXEj4W0Q

ANNEXE I: CODE HTML

```
<!DOCTYPE html>
<html>
  <!--entête de la page-->

  <head>
    <meta charset="utf-8" />
    <link href="https://fonts.googleapis.com/css?family=Oxanium|ZCOOL+QingKe+HuangYou&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="stylesheet.css"/>
    <title>Affichage numérique</title>
  </head>
<!--corps de la page-->

    <body>

      <img id="afficheur" class = "center" src = "images/afficheur.png" alt = "représentation visuelle de l'intensité">

      <p class="decibelAll"><span id = "decibel">888.8</span>dB<br /></h1>
      <div class="center" id = "exposure">
        <p  id = "textExposition">
          Durée maximale d'exposition sans protection / </br>
          Maximum exposure without protection :
        </p>
        <p id = "expositionMax">00</p>
      </div>
      <script src="javascript.js"></script>
    </body>

</html>
```

ANNEXE II: CODE CSS

```
body
{
  background-color:black;
}

#afficheur
{
  margin-top: 2em;
}

.decibelAll
{
  margin-top: 0;
  margin-bottom: 0;
  font-size: 5em;
  color:red;
  font-family: 'Oxanium', cursive;
  text-align:center;
}

#decibel{
  font-size: 5em;
  font-family: 'ZCOOL QingKe HuangYou', cursive;
}

.center
{
  display: block;
```

```
  margin-right: auto;
  margin-left: auto;
}

p {
  color:white;
  font-size: 2em;
  text-align:left;
}

.niveaux_sonores
{
  text-align:center;
}

#exposure{
  clear: both;
  display: table;
  padding: 1em 0;
}

#textExposition
{
  font-family: 'Oxanium', cursive;
  float: left;
  margin: 0 0.5em;
}

#expositionMax
```

```
{
  width: 1.1em;
  color:red;
  font-size: 5em;
  font-family:'ZCOOL QingKe HuangYou', cursive;
  text-align:center;
  float: right;
  margin: 0 0.5em;
}
```

ANNEXE III: CODE JAVASCRIPT

```
var timer = setInterval(updateSite, 1000);

function getValue(){ // doit être modifié pour lire le contenu du fichier CSV du node
    var max = 103;
    var min = 84;
    return (Math.random() * (max - min + 1) + min).toFixed(1);
}

function updateSite(){
var newValue = getValue();

    if (newValue < 100){
        valueToPrint = "0" + newValue;
    } else {
        valueToPrint = newValue;
    }

    document.getElementById('decibel').innerHTML = valueToPrint;
    document.getElementById('expositionMax').innerHTML = expositionMax(newValue);
    changeAfficheur(newValue);
}

function expositionMax(value){
    var exposition = 2.71 * (10**9) * (Math.E**(-0.231 * value));
    return exposition.toFixed(1);
}

function changeAfficheur(value){
```

```
var source;

  if (value <= 82) {              //vert
     source = "images/afficheur1.png";
  } else if (value <= 84){
     source = "images/afficheur2.png";
  } else if (value <= 86){
     source = "images/afficheur3.png";
  } else if (value <= 88){        //jaune
     source = "images/afficheur4.png";
  } else if (value <= 90){
     source = "images/afficheur5.png";
  } else if (value <= 92){
     source = "images/afficheur6.png";
  } else if (value <= 94){        //orange
     source = "images/afficheur7.png";
  } else if (value <= 96){
     source = "images/afficheur8.png";
  } else if (value <= 98){
     source = "images/afficheur9.png";
  } else if (value <= 100){       //rouge
     source = "images/afficheur10.png";
  } else if (value <= 102){
     source = "images/afficheur11.png";
  } else {
     source = "images/afficheur.png";
  }
  document.getElementById("afficheur").src = source;
  }
```

ANNEXE III: CODE NODEMCU

```
#include <OneWire.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <FS.h>
#include <WebSocketsServer.h>

#define ONE_MINUTE 60000UL
#define ONE_HOUR 3600000UL

#define SOUND_SENSOR_PIN A0

#define HALF_MINUTE 30000UL

ESP8266WebServer server(80);

File fsUploadFile;


const char *OTAName = "ESP8266";
const char *OTAPassword = "esp8266";

const char* mdnsName = "nodemcu";

WiFiUDP UDP;
```

```
IPAddress timeServerIP;

const char* ntpServerName = "time.nist.gov";

const int NTP_PACKET_SIZE = 48;

byte packetBuffer[ NTP_PACKET_SIZE];
void setup() {
  Serial.begin(115200);
  delay(10);
  Serial.println("\r\n");

  int SensorData = analogRead(soundSensor);

  if (SensorData == 0) {
    Serial.printf("No sound sensor found on pin %d. Rebooting.\r\n", SOUND_SENSOR_PIN);
    Serial.flush();
    ESP.reset();
  }

  startWiFi();

const unsigned long intervalNTP = HALF_MINUTE;
unsigned long prevNTP = 0;
unsigned long lastNTPResponse = millis();

const unsigned long intervalSound = 5000;
unsigned long prevTemp = 0;
bool sndRequested = false;
```

```
const unsigned long delay = 750;

uint32_t timeUNIX = 0;

void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - prevNTP > intervalNTP) {
    prevNTP = currentMillis;
    sendNTPpacket(timeServerIP);

  readSensor();
  }

  void readSensor()
  {
    float SensorData = analogRead(SOUND_SENSOR_PIN);

    float dbValue= 20*log10(SensorData) + 2;

    if(dBValue > 80){

        LEDStatus=true;
        digitalWrite(D0,HIGH);
      }
    if (dBValue > 82{
        LEDStatus=true;
        digitalWrite(D1,HIGH);
      }
```

```
  if(dBValue > 84){

      LEDStatus=true;
      digitalWrite(D2,HIGH);
  }
if (dBValue > 86{
      LEDStatus=true;
      digitalWrite(D3,HIGH);
  }
if(dBValue > 88){

      LEDStatus=true;
      digitalWrite(D4,HIGH);
  }
if (dBValue > 90{
      LEDStatus=true;
      digitalWrite(D5,HIGH);
  }
  if(dBValue > 92){

      LEDStatus=true;
      digitalWrite(D6,HIGH);
  }
if (dBValue > 94{
      LEDStatus=true;
      digitalWrite(D7,HIGH);
  }
}
```

```
uint32_t time = getTime();
if (time) {
  timeUNIX = time;
  Serial.print("NTP response:\t");
  Serial.println(timeUNIX);
  lastNTPResponse = millis();
} else if ((millis() - lastNTPResponse) >  ONE_HOUR) {
  Serial.println("More than 1 hour since last NTP response. Rebooting.");
  Serial.flush();
  ESP.reset();
}

if (timeUNIX != 0) {
  if (currentMillis - prevTemp > intervalTemp) {
    tempSensors.requestTemperatures();
    tmpRequested = true;
    prevTemp = currentMillis;
    Serial.println("Sound level requested");
  }
  if (currentMillis - prevTemp > delay && sndRequested) {
    uint32_t actualTime = timeUNIX + (currentMillis - lastNTPResponse) / 1000;
    // The actual time is the last NTP time plus the time that has elapsed since the last NTP response
    sndRequested = false;
    float snd = tempSensors.getTempCByIndex(0);
    temp = round(temp * 100.0) / 100.0; // round temperature to 2 digits

    Serial.printf("Appending sound level to file: %lu,", actualTime);
    Serial.println(temp);
    File tempLog = SPIFFS.open("/sound.csv", "a");
```

```
      tempLog.print(actualTime);
      tempLog.print(',');
      tempLog.println(snd);
      tempLog.close();
    }
  } else {
    sendNTPpacket(timeServerIP);
    delay(500);
  }

  server.handleClient();
  ArduinoOTA.handle();
}
void startWiFi() {

 WiFi.begin(ssid, password);
  Serial.print("Connecting to ");
  Serial.print(ssid); Serial.println(" ...");

  int i = 0;
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(++i); Serial.print(' ');
  }

  Serial.println('\n');
  Serial.println("Connection established!");
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());
```

```
}


void startOTA() {
  ArduinoOTA.setHostname(OTAName);
  ArduinoOTA.setPassword(OTAPassword);

  ArduinoOTA.onStart([]() {
    Serial.println("Start");
  });
  ArduinoOTA.onEnd([]() {
    Serial.println("\r\nEnd");
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  });
  ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
    else if (error == OTA_END_ERROR) Serial.println("End Failed");
  });
  ArduinoOTA.begin();
  Serial.println("OTA ready\r\n");
}

void startSPIFFS() {
```

```
  SPIFFS.begin();
  Serial.println("SPIFFS started. Contents:");
  {
    Dir dir = SPIFFS.openDir("/");
    while (dir.next()) {
      String fileName = dir.fileName();
      size_t fileSize = dir.fileSize();
      Serial.printf("\tFS File: %s, size: %s\r\n", fileName.c_str(), formatBytes(fileSize).c_str());
    }
    Serial.printf("\n");
  }
}

void startWebSocket() {
  webSocket.begin();
  webSocket.onEvent(webSocketEvent);
  Serial.println("WebSocket server started.");
}

void startMDNS() { // Start the mDNS responder
  MDNS.begin(mdnsName);
  Serial.print("mDNS responder started: http://");
  Serial.print(mdnsName);
  Serial.println(".local");
}

void startServer() {
  server.on("/edit.html",  HTTP_POST, []() {
    server.send(200, "text/plain", "");
```

```
  }

  server.onNotFound(handleNotFound);

  server.begin();
  Serial.println("HTTP server started.");
}
void handleNotFound(){
  if(!handleFileRead(server.uri())){
    server.send(404, "text/plain", "404: File Not Found");
  }
}
bool handleFileRead(String path) {
  Serial.println("handleFileRead: " + path);
  if (path.endsWith("/")) path += "index.html";
  String contentType = getContentType(path);
  String pathWithGz = path + ".gz";
  if (SPIFFS.exists(pathWithGz) || SPIFFS.exists(path)) {
    if (SPIFFS.exists(pathWithGz))
      path += ".gz";
    File file = SPIFFS.open(path, "r");
    size_t sent = server.streamFile(file, contentType);
    file.close();
    Serial.println(String("\tSent file: ") + path);
    return true;
  }
  Serial.println(String("\tFile Not Found: ") + path);
  return false;
}
```

```
void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t lenght) {
  switch (type) {
    case WStype_DISCONNECTED:
      Serial.printf("[%u] Disconnected!\n", num);
      break;
    case WStype_CONNECTED: {
        IPAddress ip = webSocket.remoteIP(num);
       Serial.printf("[%u] Connected from %d.%d.%d.%d url: %s\n", num, ip[0], ip[1], ip[2], ip[3], payload)
      }
      break;
  }
String formatBytes(size_t bytes) {
  if (bytes < 1024) {
    return String(bytes) + "B";
  } else if (bytes < (1024 * 1024)) {
    return String(bytes / 1024.0) + "KB";
  } else if (bytes < (1024 * 1024 * 1024)) {
    return String(bytes / 1024.0 / 1024.0) + "MB";
  }
}
String getContentType(String filename) {
  if (filename.endsWith(".html")) return "text/html";
  else if (filename.endsWith(".css")) return "text/css";
  else if (filename.endsWith(".js")) return "application/javascript";
  else if (filename.endsWith(".ico")) return "image/x-icon";
  else if (filename.endsWith(".gz")) return "application/x-gzip";
  return "text/plain";
}
```