

Travail de fin d'études

Conception d'une solution intégrée pour l'acquisition, le stockage et la visualisation de données

Présenté par

Aoufia Samir

En vue de l'obtention du
grade de

**Bachelier en informatique
et systèmes, orientation
informatique industrielle**

Année Académique 2023–2024

Je tiens à exprimer ma gratitude à toutes les personnes qui ont contribué à la réalisation de travail de fin d'étude ainsi que pendant mon stage.

Mes remerciements vont tout d'abord à mes professeurs, particulièrement M. Lisson et M. Michaux pour leur conseil et leurs encadrements tout le long de mon stage.

Je tiens aussi à remercier toute l'équipe du Smart Gastronomy Lab pour leur accueil. Un remerciement particulier au Pr. Haubruge et au Dr. Goffin pour m'avoir accueilli au sein de leur entreprise. Je souhaite également exprimer ma reconnaissance à M. Codutti et Mme. Bradfer pour leur collaboration.

Enfin, je souhaite remercier mes proches pour leur soutien, leur patience et leurs encouragements.

1. Introduction	- 1 -
2. Cadre de travail	- 2 -
2.1. Présentation de l'entreprise	- 2 -
2.2. Cahier des charges	- 3 -
3. Système embarqué	- 4 -
3.1. Esp-01S	- 7 -
3.2. Arduino Méga 2560	- 8 -
3.3. Esp32	- 9 -
3.4. Système de cuisson	- 10 -
3.5. Plateau	- 17 -
3.6. Bluetooth	- 18 -
4. Serveur	- 21 -
4.1. Raspberry PI 3 (serveur)	- 21 -
4.2. Docker	- 22 -
4.3. Portainer Community Edition	- 23 -
4.4. Mosquitto	- 23 -
4.5. InfluxDB	- 24 -
4.6. PostgreSQL	- 26 -
5. Application web	- 27 -
5.1. Nodejs	- 27 -
5.2. Nextjs	- 28 -
5.3. React	- 29 -
5.4. TypeScript	- 29 -
5.5. Prisma	- 30 -
5.6. Shadcn/ui	- 31 -
5.7. Recharts	- 32 -
5.8. Site web	- 32 -
6. Conclusion	- 43 -

7. Lexique	- 44 -
8. Médiagraphie	- 45 -
9. Annexes	- 47 -
9.1. Schéma Prisma complet	- 47 -
9.2. Schéma système de cuisson	- 47 -
9.3. Cahier des charges	- 48 -
9.4. Code esp-01S	- 50 -
9.5. Code Bluetooth	- 53 -
9.6. Code système de cuisson	- 56 -
9.7. Code site web	- 72 -

1. Introduction

Dans le cadre de l'obtention de mon bachelier en informatique industrielle, la réalisation d'un stage de 15 semaines est obligatoire. À la suite du stage un projet m'a été confié au sein du Smart Gastronomy Lab. Ce projet consistait à apporter sur des systèmes embarqués déjà existant afin de permettre une connexion sans fil ainsi que l'affichage et le traitement des données sur un site internet.

Dans ce travail de fin d'étude, je vais expliquer les différents systèmes embarqués sur lesquels j'ai travaillé, décrire le développement du site internet, et de présenter les différents outils qui m'ont permis de mener à bien ce projet.

Je vais commencer par présenter l'entreprise dans laquelle j'ai réalisé ce projet et le cahier des charges qui m'a été demandé. Ensuite, je vais parler des divers dispositifs utilisés qui m'ont permis de faire communiquer les différents systèmes embarqués sans fil. Ainsi que le serveur qui m'a permis d'héberger mes différents outils, afin de permettre la communication entre les systèmes embarqués et le site internet. Pour finir, je vais expliquer les outils utilisés pour le développement du site internet.

Je vous souhaite une bonne lecture.

2. Cadre de travail

2.1. Présentation de l'entreprise

Le Smart Gastronomy Lab est un living lab wallons qui fait partie de l'université de Liège. Le Smart Gastronomy Lab se consacre à l'alimentation au sens large et à 2 activités principales : le service aux entreprises et la recherche & développement.

Le service aux entreprises est un coaching dans le domaine de l'alimentation et l'aide au développement de recettes ou de produits accompagnés de phase de test. Le Smart Gastronomy Lab fait également de la location de cuisine pour professionnels ou des événements sur-mesure.

La recherche & développement dans l'agro-alimentaire est très vaste, le but de l'équipe est de trouver la manière dont les technologies peuvent participer au "mieux manger".

Le Smart Gastronomy Lab possède un laboratoire culinaire de prototype de produit, un laboratoire de recherche ainsi qu'un laboratoire de fabrication et d'électronique. Ils réalisent également des ateliers sur la sensibilisation aux tendances alimentaires.



Figure 1 : Logo du Smart Gastronomy Lab

2.2. Cahier des charges

Le but de mon travail a été de mettre en place un site web qui permet de visualiser les données de plusieurs systèmes embarqués et de traiter les données collectées. Les données peuvent être traitées en temps réel ou de manière différée, en fonction des heures et dates introduites par un utilisateur, et sont téléchargeables en format csv (Comma-Separated Values).

Pour ce faire, il a fallu façonner différentes composantes :

- Connectivité des appareils,
- Gestion du réseau,
- Base de données,
- Création d'API (Application Programming Interface),
- Application web.

Voici un schéma qui explique, le fonctionnement global du projet.¹

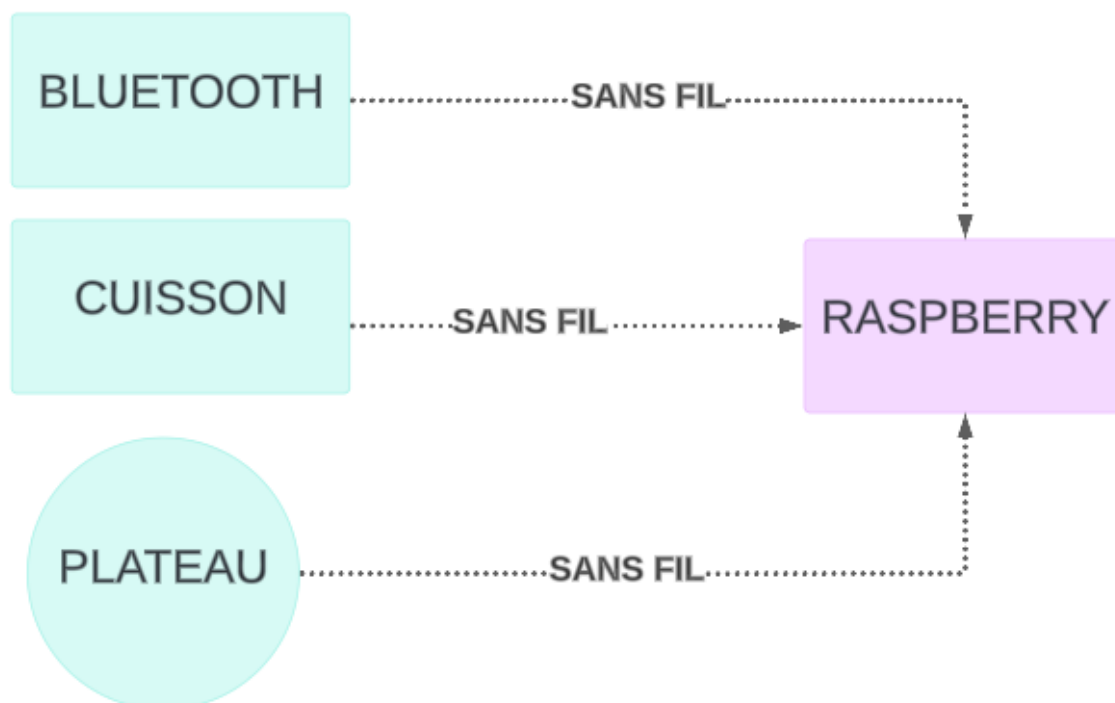


Figure 2 : Schéma bloc du fonctionnement global du projet

¹ Cahier des charges complet, voir annexe.

3. Système embarqué

Dans cette partie, les divers composants que j'ai utilisés pour créer des objets connectés, ainsi que les différents projets sur lesquels j'ai travaillé durant mon stage vont être expliqués. De plus, je vais éclaircir certains points pour permettre une meilleure compréhension des composants et des protocoles de communication.

Un CPU (Central Processing Unit) est le processeur d'un système embarqué, il est responsable de l'exécution des différentes instructions et permet le traitement des données.

Un microcontrôleur est un circuit intégré qui rassemble un CPU, de la mémoire et des interfaces d'entrées / sorties qui permettent le contrôle de divers capteur et actionneur.

Des entrées ou sorties numériques sont des signaux qui peuvent varier entre 2 états haut (1) ou bas (0).

Des entrées ou sorties analogiques sont des signaux continus qui peuvent varier d'une valeur à une autre. Pour les sorties PWM est utilisé.

Le PWM (Pulse Width Modulation) est utilisé pour simuler une sortie analogique à partir d'un signal numérique en modulant la largeur des impulsions avec une fréquence et une période définie.

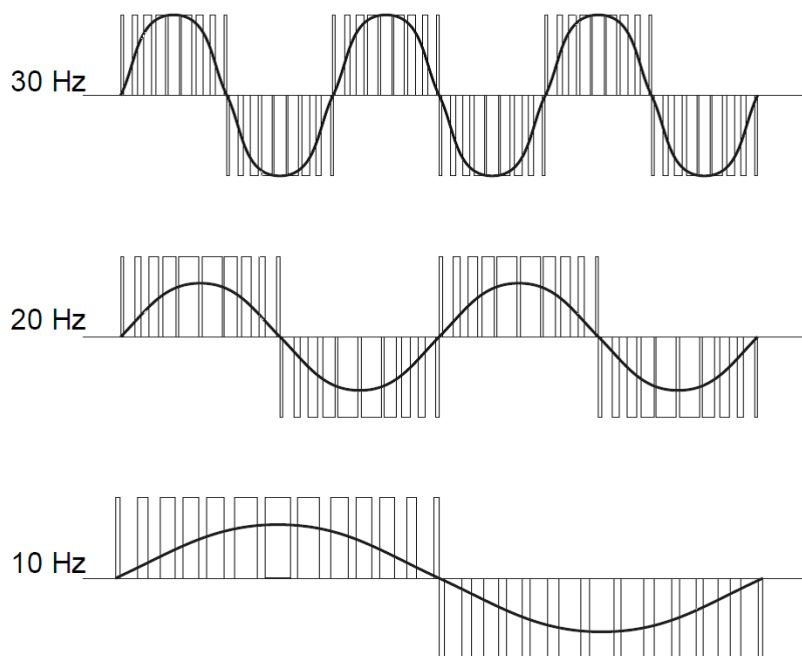


Figure 3 : Illustration de PWM

L'UART (Universal Asynchronous Receiver Transmitter) est un protocole de communication de série asynchrone qui permet de transmettre des données entre un microcontrôleur et un dispositif. La liaison est faite avec 2 fils RX & TX que l'on croise de l'Arduino au composant ainsi que la masse.

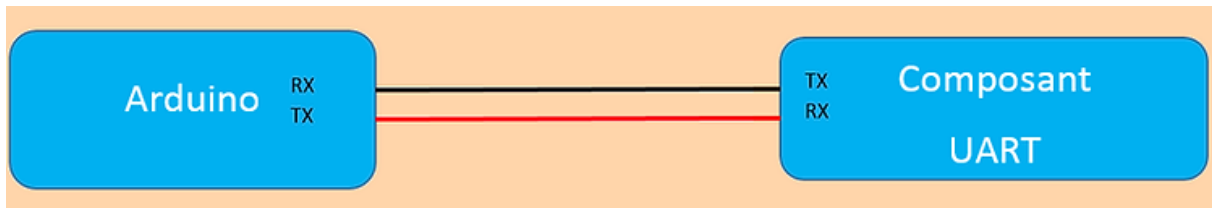


Figure 4 : Illustration du câblage UART

Le SPI (Serial Peripheral Interface) est un protocole de communication de série synchrone qui fonctionne en duplex, permettant la communication dite full duplex (les deux circuits peuvent communiquer en même temps). La liaison est faite selon un schéma maître-esclave avec 4 fils :

- SCLK (Serial Clock),
- MOSI (Master Output, Slave Input),
- MISO (Master Input, Slave Output),
- SS (Slave Select).

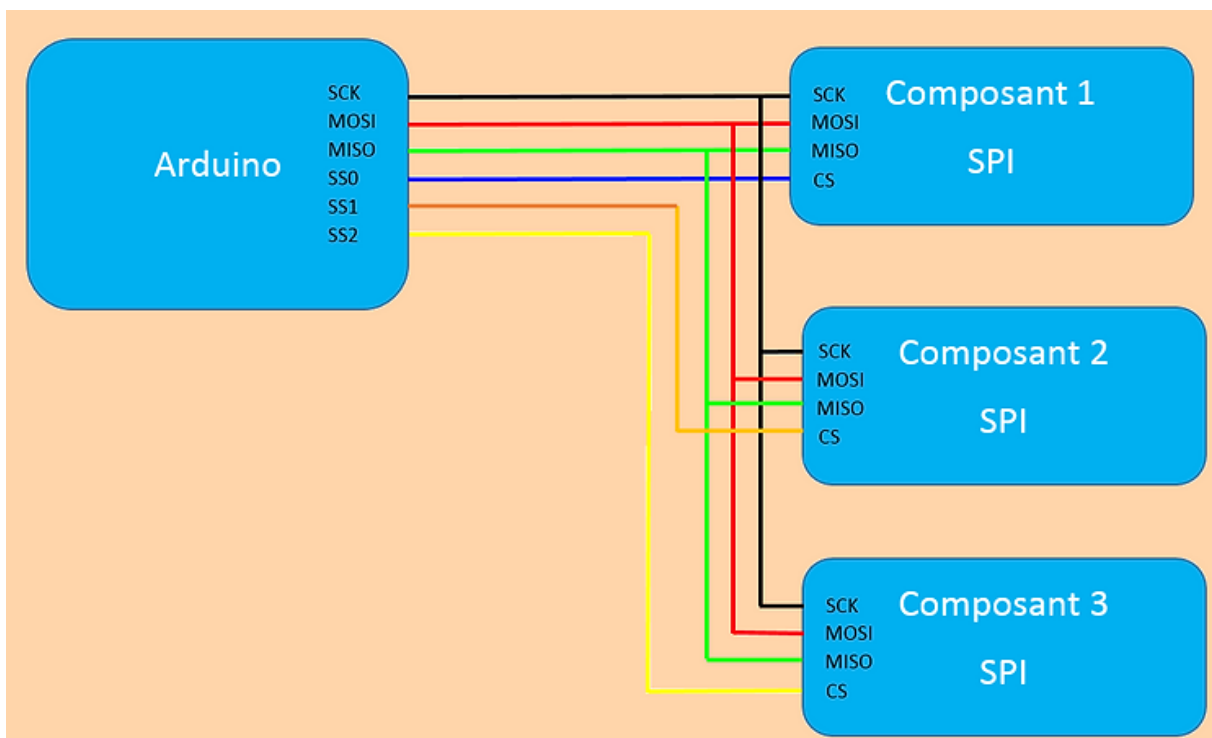


Figure 5 : Illustration du câblage SPI

L'I2C (inter-Integrated Circuit) est un protocole de communication de série synchrone qui permet de transmettre des données entre différents composants électroniques. La liaison est faite selon un schéma maître-esclave avec 3 fils :

- SDA (Signal des données),
- SCL (Signal d'horloge),
- Masse (0 Volt).

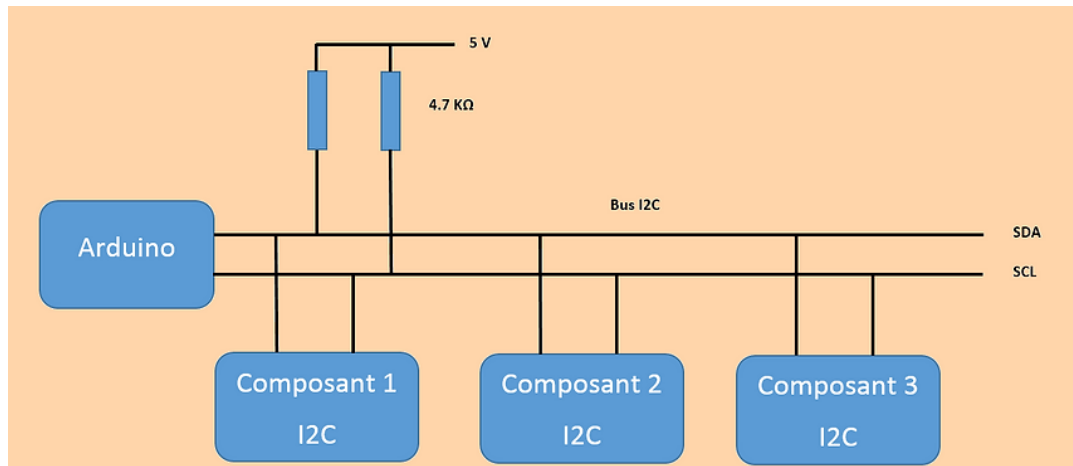


Figure 6 : Illustration du câblage I2C

3.1. Esp-01S

L'esp-01S est un microcontrôleur qui utilise l'esp8266 comme CPU, avec une mémoire flash de 1 MB. Il peut fonctionner seul grâce à ses broches GPIO (General Purpose Input Output), ou être intégré en série avec le protocole de communication UART a un autre microcontrôleur qui ne possède pas de module wifi intégré.

Ce module est doté de 2 modes : écriture et lecture, pour basculer entre ces modes il y a une manipulation à faire avec les GPIO, cette manipulation consiste à brancher et/ou débrancher certains GPIO au moment du téléversement du code dans l'esp-01S, le mode écriture permet de téléverser le code dans l'esp-01S, tandis que le mode lecture sert à exécuter le code en boucle. Il dispose aussi de plusieurs modes de fonctionnement wifi :

- Station : Se connecte à un réseau existant
- Point d'accès : crée son propre réseau
- Station + point d'accès : fonctionne en simultané

L'esp-01S est utilisé pour des systèmes embarqués nécessitant une connexion réseau, tel que des projets IoT (Internet of Things) ou domotique.

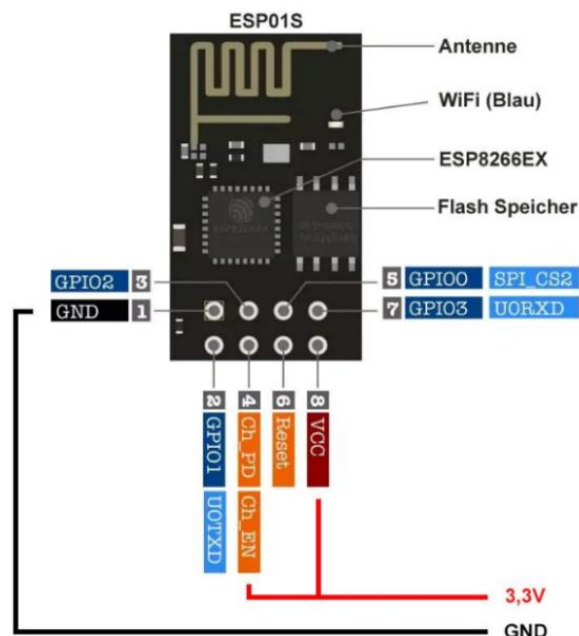


Figure 7 : ESP-01S

3.2. Arduino Méga 2560

L'Arduino méga 2560 est un microcontrôleur qui possède un ATmega2560 comme CPU avec une mémoire flash de 256 kB et 8kB de SRAM, il possède 54 entrées/sorties numériques, dont 14 sorties digitales (PWM) et 16 entrées analogiques. L'Arduino méga possède une alimentation de 5V et de 3.3V et 1 bouton reset.

Il peut prendre en charge plusieurs protocoles de communication tels que l'UART, I2C et le SPI, ce qui fait de lui un microcontrôleur très polyvalent permettant de communiquer avec divers module et capteur.

De plus Arduino méga est une marque énormément utiliser dans le monde du système embarqué, il y a donc beaucoup de bibliothèque, documentation et tutoriel, ce qui facilité sont intégration dans des systèmes complexes.

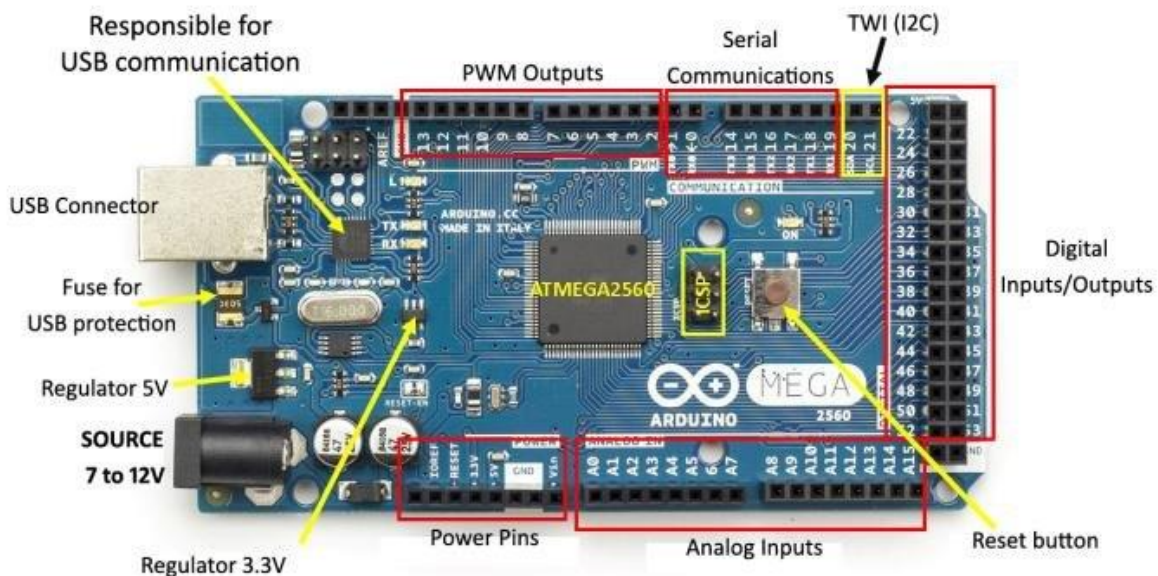


Figure 8 : Arduino Mega 2560

3.3. Esp32

L'esp32 est un microcontrôleur qui possède comme CPU un Xtensa avec une mémoire flash de 4MB, ainsi que 15 entrées/sorties numériques dont 10 qui peuvent utiliser le PWM, 2 sorties analogiques et 15 entrées analogiques. Il possède un module wifi 802.11 2GHz et un module bluetooth qui permet de faire du BLE (Bluetooth Low Energy) ou du bluetooth classique.

/	BLE	Bluetooth classique
Consommation	Très faible	Elevée
Débit de données	Max 1 Mbps	Max 2 à 3 Mbps
Connexion	Connexions sporadiques et rapides	Connexion continue
Utilisation	IoT, capteurs	Audio, multimédia, transferts de fichiers

L'esp32 peut prendre en charge plusieurs protocoles de communication tel que l'UART, I2C et le SPI, ce qui offre une grande flexibilité pour communiquer avec divers composants, capteur et module.

De plus l'esp32 est fortement utilisé pour des systèmes embarqués sans fil (IoT – Internet of Things) du a sa faible consommation et de ses multiples options de connexion. Il peut être programmé en MicroPython, Arduino et C++. Grâce à une grande communauté, il possède une bonne documentation ce qui facilite l'utilisation.

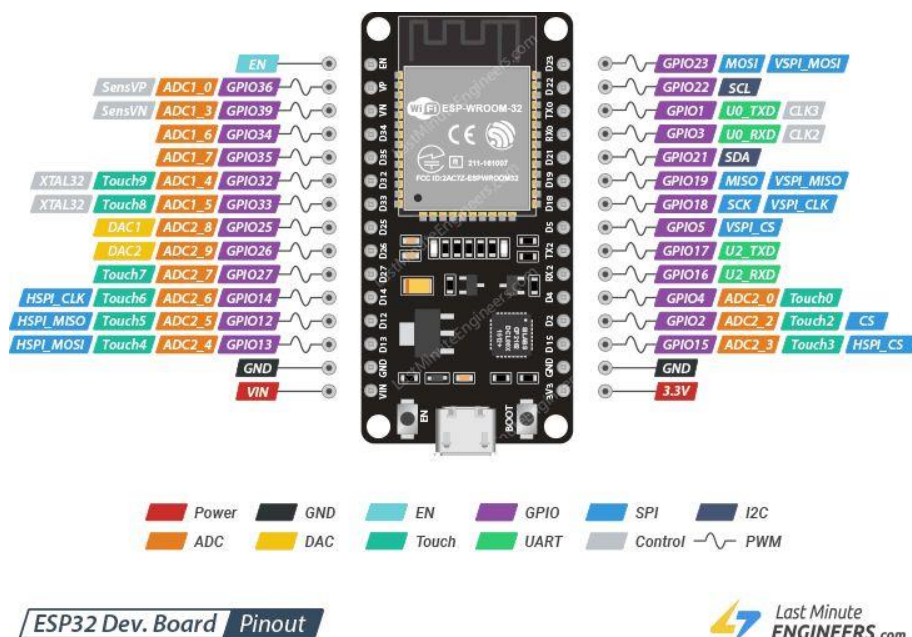


Figure 9 : ESP32

3.4. Système de cuisson

Le système de cuisson a été conçu par Nathan Codutti, pour récupérer diverses données durant la cuisson.

Le système comprend plusieurs capteurs, permettant une surveillance en temps réel des conditions de cuisson. Les capteurs sont :

- DS18B20 est un capteur de température numérique,
- MXL90614 est un capteur infrarouge sans contact pour mesurer la température des surfaces,
- 2 MAX31856 est un convertisseur thermocouples, capable de lire des températures élevées avec une grande précision,
- DHT22 est un capteur de température et d'humidité.

Ces différents capteurs permettent de collecter des données précises sur les conditions de cuisson.

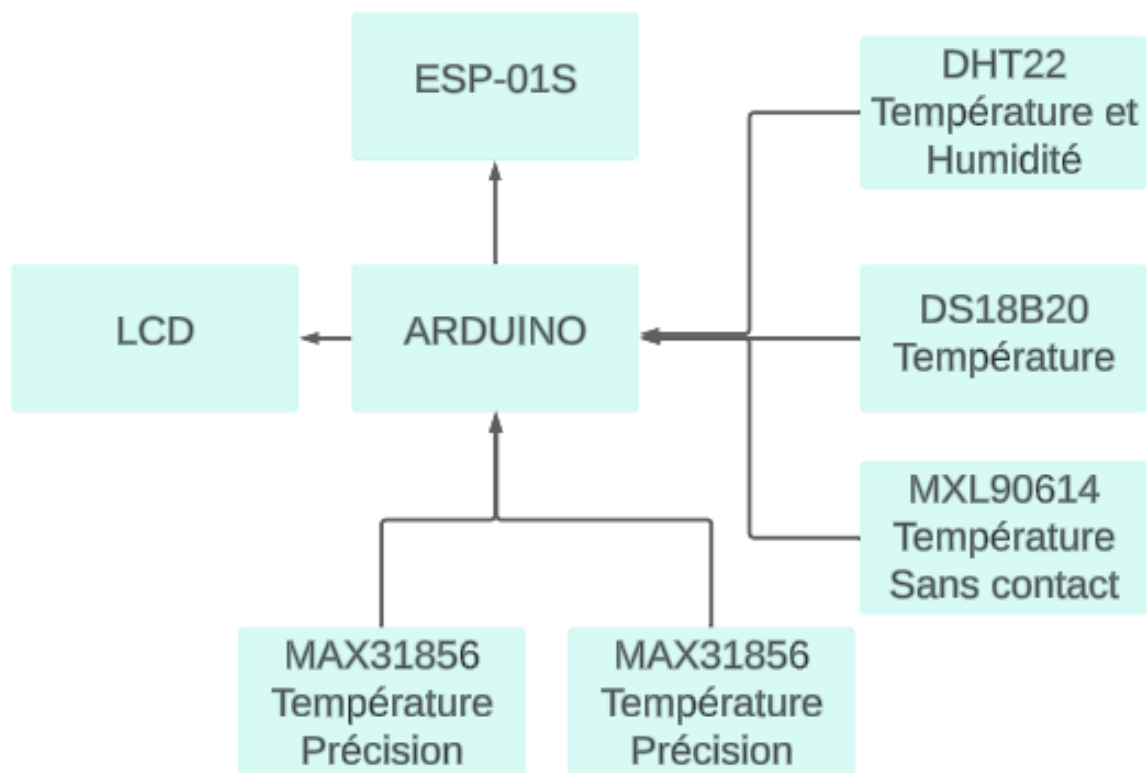


Figure 10 : Schéma bloc du système de cuisson

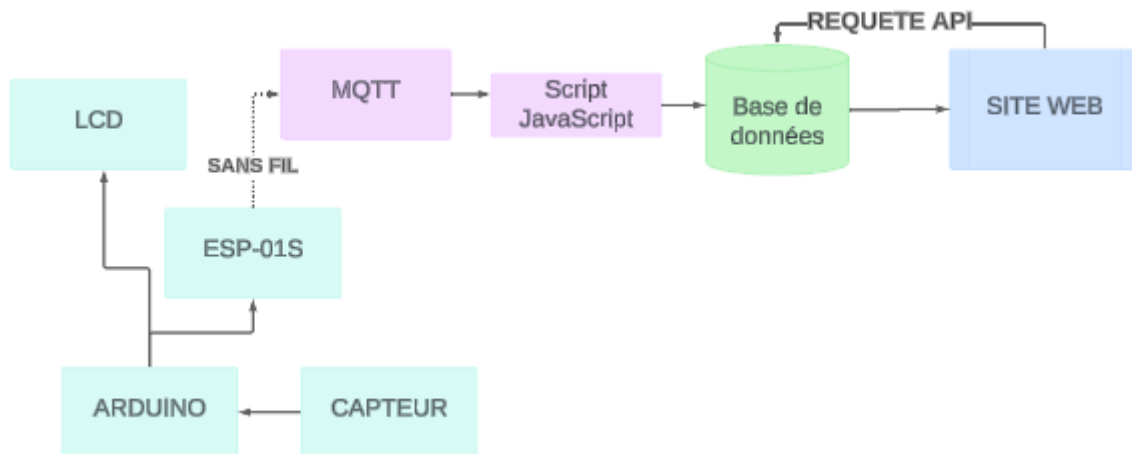


Figure 11 : Schéma bloc complet du système de cuisson

L'Arduino méga est utilisé pour récupérer les données des différents capteurs pour les afficher sur un écran LCD. Ma partie a été d'ajouter un esp-01S pour permettre une connexion sans fil et ainsi envoyer les données de l'Arduino sur un serveur MQTT. Pour faire cela, l'esp-01S a été connecté à l'Arduino en série via les pins RX et TX qui utilise le protocole de communication UART. Cela permet de recevoir les données des capteurs sur l'esp-01S et de l'envoyer sur le serveur.

Envoi du code de l'Arduino vers l'esp-01S

Le code dans l'Arduino, envoie les données sous forme de chaînes de caractères (strings), séparés par des virgules, permettant le tri.

Le « `String(variable).c_str()` » permet de convertir les données en caractère grâce au string puis de convertir le caractère en chaînes de caractères grâce au « `.c_str` ». La conversion est obligatoire pour envoyer les données en série. Le « `Serial.write` » permet d'envoyer la chaîne de caractères sur le port serie.

```

Serial.write(String(humidity).c_str());
Serial.write(",");
Serial.write(String(temperature).c_str());
Serial.write(",");
Serial.write(String(power).c_str());
Serial.write(",");
Serial.write(String(ds18b20Temp).c_str());
Serial.write(",");
Serial.write(String(mlxTemp).c_str());
Serial.write(",");
Serial.write(String(thermocoupleTemp).c_str());
Serial.write(",");
Serial.write(String(thermocoupleTestoTemp).c_str());

```

Figure 12 : Code pour l'envoi des données de l'Arduino vers l'esp-01S

Traitement et envoi du code de l'esp-01S vers le serveur MQTT

Le code dans l'esp-01S, lis les données reçues en série par l'Arduino, les extrait et les publie dans le bon topic. Le code est séparé en plusieurs parties : la lecture des données, l'extraction et la publication.

On commence par la lecture des données, il y a une boucle « while » qui vérifie s'il y a des données sur le port série via « mySerial ». S'il y a des données, elles seront lues jusqu'à ce qu'un caractère de nouvelle ligne (« \n ») apparaisse et sont stockées dans une variable « IncomingString ». La variable « StringReady » passe alors à vrai pour indiquer que les données on finit d'être reçues.

```
while (mySerial.available()) {
    IncomingString = mySerial.readStringUntil('\n');
    StringReady = true;
}
```

Figure 13 : Lecture des données reçues par l'Arduino

Ensuite il y a l'extraction des données, il y a une condition : si la variable « StringReady » est égale à vrai on procède à l'extraction des données. On déclare un tableau « Variable [7] », pour stocker les valeurs, ensuite, il y a une boucle « FOR » qui parcourt les 6 premières valeurs en trouvant les indices représentés par les caractères de la virgule dans « IncomingString » et extrait les sous-chaînes entre ces indices pour les stocker dans le tableau « Variable [7] ».

```
if (StringReady) {
    String variables[7];
    int startIndex = 0;

    for (int i = 0; i < 6; i++) {
        int commaIndex = IncomingString.indexOf(',', startIndex);
        if (commaIndex != -1) {
            variables[i] = IncomingString.substring(startIndex, commaIndex);
            startIndex = commaIndex + 1;
        }
    }

    variables[6] = IncomingString.substring(startIndex);
}
```

Figure 14 : Extraction et traitement des données reçues par l'Arduino

Pour finir, on déclare une constante avec les noms de tous les topics du serveur MQTT, une boucle « FOR » convertit les chaînes de caractères du tableau « Variable [7] » en entier grâce à «.toInt()» et publie les valeurs sur le bon topic du serveur MQTT. Ensuite, il y a 1 seconde de délai. La fonction « client.loop » permet de maintenir une connexion MQTT et de traiter les données entrantes.



```
const char* topics[] = {"Humidity", "Temperature", "Power", "Ds18b20Temp",  
"MlxTemp", "ThermocoupleTemp", "ThermocoupleTestoTemp"};  
  
for (int i = 0; i < 7; i++) {  
    int value = variables[i].toInt();  
  
    // Publier chaque variable dans un topic MQTT spécifique  
    client.publish(topics[i], String(value).c_str());  
}  
  
delay(1000);  
client.loop();  
}  
}
```

Figure 15 : Envoi des données dans sur le serveur MQTT

Envoi du code du serveur MQTT vers la base de données

Une fois les données envoyer sur le serveur MQTT, il y a un script javascript permettant de récupérer les données et de les envoyés directement dans une base de données, un exemple du script a été fourni par la base de données influxDB et a été modifier, afin de correspondre aux paramètres du projet.

```
import mqtt from 'mqtt';
import {InfluxDB, Point} from '@influxdata/influxdb-client'

const token = 'K7MVFGETSqtNpc4vWArvmRw'
const url = 'http://192.168.0.153:8086'

const client1 = new InfluxDB({url, token})

let org = `Stage`
let bucket = `PLAQUE_DE_CUISSON`

let writeClient = client1.getWriteApi(org, bucket, 'ns')

const client = mqtt.connect('mqtt://192.168.0.153:1883');

client.on('connect', function () {
  //console.log('Connected to MQTT broker');
  client.subscribe('Humidity');
  client.subscribe('Temperature');
  client.subscribe('Power');
  client.subscribe('Ds18b20Temp');
  client.subscribe('MlxTemp');
  client.subscribe('ThermocoupleTemp');
  client.subscribe('ThermocoupleTestoTemp');
});

client.on('message', function (topicMqtt, message) {
  let point = new Point(topicMqtt)
    .tag('tagname1', 'tagvalue1')
    .intField('field1', message.toString())

  void setTimeout(() => {
    writeClient.writePoint(point)
  },)
});
```

Figure 16 : Script Javascript pour l'envoi des données dans InfluxDB

Il existe plusieurs solutions pour envoyer les données sans fil, comme l'Arduino GIGA R1 wifi ou bien l'ATWINC1510.

Comparaison entre l'Arduino méga + l'esp-01S et l'Arduino Giga R1

L'Arduino Giga R1 wifi contient déjà un module wifi et peut donc faire du sans-fil sans l'aide d'un module externe.

Avec le tableau ci-dessous, on comprend que l'Arduino méga 2560 avec l'esp-01S est une solution simple d'intégration et peu coûteuse, mais est limitée en termes de performance et a besoin de module externe pour certaines fonctionnalités. Tandis que l'Arduino Giga R1 wifi est plus coûteux, mais offre une intégration plus transparente car il intègre le wifi et le bluetooth sans module externe et offre une meilleure performance.

/	Arduino Mega + ESP-01S	Arduino giga r1 wifi
Microcontrôleur intégré	ATmega2560 + ESP8266EX	STM32H747XI
Mémoire flash	256 KB	2MB
Entrée / sortie numérique	54 (15 PWM)	76 (12 PWM)
Entrée analogique	16	12 (ADC), 2 (DAC)
Wifi	ESP-01S (802.11 b/g/n)	802.11 b/g/n
Bluetooth	Non	Bluetooth 5.2
Programmation	UART	USB ou UART
Communication	UART, SPI et I2C	UART, SPI, I2C et Can
Prix	43€	70€
Avantage	Moins chère	Wifi et bluetooth intégré, plus de pin, plus de protocole de communication
Inconvénient	Configuration wifi et bluetooth plus complexe, moins de pin, moins de protocole de communication	Plus chère, plus haute consommation d'énergie

Comparaison entre l'esp-01S et l'ATWINC1510

Comme autre possibilité, il y avait aussi le module wifi ATWINC1510. Il n'a pas de microcontrôleur et doit être contrôlé via le protocole de communication SPI par un microcontrôleur hôte.

L'esp-01S a été un meilleur choix pour sa simplicité d'intégration, de programmation, et pour son fonctionnement autonome, car il offre une solution directe pour l'ajout d'une connexion sans fil. Il possède également une documentation complète et détaillée, car il utilise le même processeur que l'esp8266 qui a une grande communauté active. De plus, l'Arduino étant déjà presque entièrement utilisé, l'UART a permis de réduire le nombre de fils nécessaire pour une connexion sans fil, évitant ainsi la surcharge de fils.

/	ESP-01S	ATWINC1510
Microcontrôleur intégré	Esp8266EX	Non
Programmabilité	Arduino ide	Non, contrôlé via SPI
Autonome	Oui	Non
Interfaces	UART	SPI
Wi-fi	802.11 b/g/n	802.11 b/g/n
Bluetooth	Non	Oui, 5.0
Avantages	Programmable, autonome, simple d'utilisation	Faible consommation, Bluetooth intégré
Inconvénients	Consommation plus élevée, Bluetooth non intégré	Complexe à mettre en place, Doit être contrôlé

3.5. Plateau

Le plateau a été conçu pour mesurer la force des personnes quand ils mangent. Il fonctionne avec un esp32, car il prend peu de place et dispose de suffisamment de broches pour connecter tous les composants nécessaires au bon fonctionnement du plateau. De plus, il permet une connexion wifi et bluetooth. Le plateau utilise quatre pesons qui mesurent la pression exercée. Ensuite, les mesures sont utilisées pour calculer le poids total, ce qui permet d'avoir des données précises et fiables sur le poids exercé en temps réel. Ma partie a été de rajouter le code permettant la connexion sans fil et l'envoi des données.

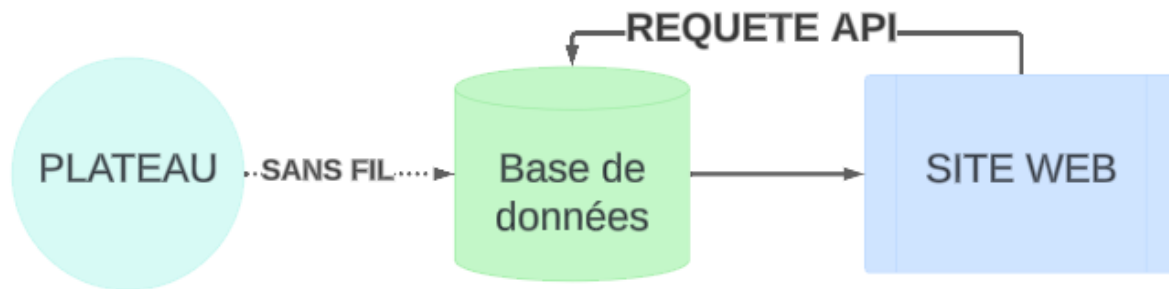


Figure 17 : Schéma bloc du plateau

Code d'exemple d'influxDB

Pour cela, le code d'exemple fourni par influxDB a été utilisé, que j'ai modifié pour le bon fonctionnement du plateau et l'envoi direct des données dans la base de données. Voici l'exemple fourni par influxDB.

```

// Code example provided by InfluxDB

#include <ESP32>
#include <WiFiMulti.h>
WiFiMulti wifiMulti;
#define DEVICE "ESP32"
#elif defined(ESP8266)
#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti wifiMulti;
#define DEVICE "ESP8266"
#endif

#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>

// WiFi AP SSID
#define WIFI_SSID "YOUR_WIFI_SSID"
// WiFi password
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"

#define INFLUXDB_URL "http://192.168.3.4:8086"
#define INFLUXDB_TOKEN "LM3WcQYhkTy-yYEB500XMHMe"
#define INFLUXDB_ORG "d049158eccc7bcd"
#define INFLUXDB_BUCKET "YOUR_BUCKET"

// Time zone info
#define TZ_INFO "UTC2"

// Declare InfluxDB client instance with preconfigured InfluxCloud certificate
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET,
INFLUXDB_TOKEN, InfluxDbCloud2CACert);

// Declare Data point
Point sensor("wifi_status");
  
```

Figure 18 : Code exemple fourni par InfluxDB

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

  // Setup wifi
  WiFi.mode(WIFI_STA);
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);

  Serial.print("Connecting to wifi");
  while (wifiMulti.run() != WL_CONNECTED) {
    Serial.print(".");
    delay(100);
  }
  Serial.println();
  |
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");

  // Check server connection
  if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
  } else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
  }
  // Add tags to the data point
  sensor.addTag("device", DEVICE);
  sensor.addTag("SSID", WiFi.SSID());
}
```

Figure 20 : Code exemple fourni par InfluxDB

```
void loop() {
  // Clear fields for reusing the point. T
  sensor.clearFields();

  // Store measured value into point
  // Report RSSI of currently connected network
  sensor.addField("rssi", WiFi.RSSI());

  // Print what are we exactly writing
  Serial.print("Writing: ");
  Serial.println(sensor.toLineProtocol());

  // Check WiFi connection and reconnect if needed
  if (wifiMulti.run() != WL_CONNECTED) {
    Serial.println("Wifi connection lost");
  }

  // Write point
  if (!client.writePoint(sensor)) {
    Serial.print("InfluxDB write failed: ");
    Serial.println(client.getLastErrorMessage());
  }

  Serial.println("Waiting 1 second");
  delay(1000);
}
```

Figure 19 : Code exemple fourni par InfluxDB

3.6. Bluetooth

Une connexion bluetooth a été mise en place pour récupérer des valeurs. Pour se faire un esp32 et un Raspberry Pi ont été utilisés. Cela fonctionne grâce à un script python et un script javascript.

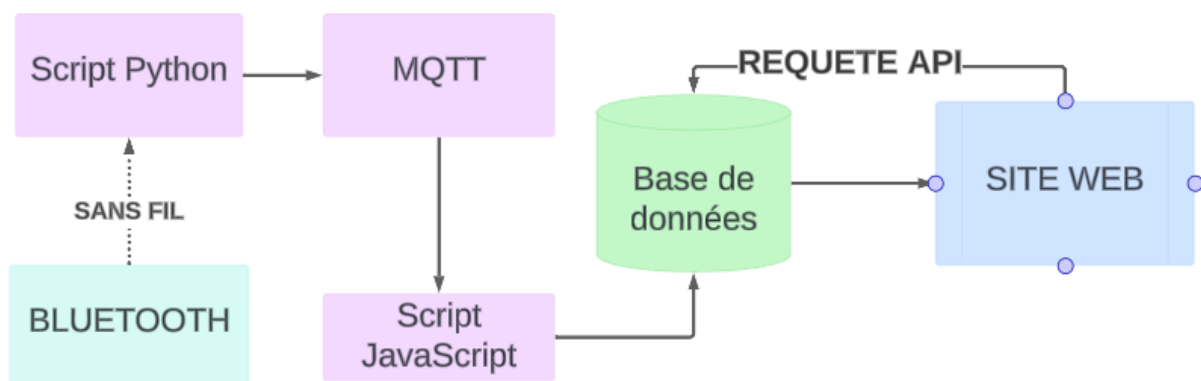


Figure 21 : Schéma bloc de bluetooth

Script python pour connecter l'esp32 au Raspberry Pi

Le script python permet de connecter l'esp32 au Raspberry Pi. Une fois connecté, l'esp32 envoie les données par bluetooth au Raspberry Pi, qui les récupère et les envoie au serveur MQTT.

```

from bluepy import btle
import paho.mqtt.client as mqtt
import time

SERVEUR = '192.168.0.153'

client1 = mqtt.Client()
client1.connect(SERVEUR, 1883, 60)

class MyDelegate(btle.DefaultDelegate):
    def __init__(self):
        btle.DefaultDelegate.__init__(self)

    def handleNotification(self, cHandle, data):
        print(data[0])
        client1.publish('bluetooth',data[0])
        time.sleep(1)

# Initialisation
address = "34:86:5D:FC:49:DE"
service_uuid = "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
char_uuid = "beb5483e-36e1-4688-b7f5-ea07361b26a8"

p = btle.Peripheral("34:86:5d:fc:49:de")
p.setDelegate(MyDelegate())

# Setup to turn notifications on, e.g.
svc = p.getServiceByUUID("4fafc201-1fb5-459e-8fcc-c5c9c331914b")
ch = svc.getCharacteristics("beb5483e-36e1-4688-b7f5-ea07361b26a8")[0]

setup_data = b"\x01\x00"
p.writeCharacteristic(ch.valHandle + 1, setup_data)
ch_data = p.readCharacteristic(ch.valHandle + 1)

while True:
    if p.waitForNotifications(1.0):
        # handleNotification() was called
        continue
    # Perhaps do something else here

```

Figure 22 : Script Python qui permet la connexion entre l'esp32 et le Raspberry

Envoi du code du serveur MQTT vers la base de données

Pour finir, un script javascript similaire à celui du système de cuisson a été utilisé pour récupérer les données sur le serveur MQTT et les envoyer dans une base de données, un exemple du script a été fourni par la base de données influxDB et a été modifié et a été également modifié afin de correspondre aux paramètres du projet.

```
import mqtt from 'mqtt';
import {InfluxDB, Point} from '@influxdata/influxdb-client'

const token = 'K7MVFGETSqtNpc4vWArvmRw'
const url = 'http://192.168.0.153:8086'

const client1 = new InfluxDB({url, token})

let org = `Stage`
let bucket = `Bluetooth`

let writeClient = client1.getWriteApi(org, bucket, 'ns')

const client = mqtt.connect('mqtt://192.168.0.153:1883');

client.on('connect', function () {
  //console.log('Connected to MQTT broker');
  client.subscribe('bluetooth');
});

client.on('message', function (topicMqtt, message) {
  let point = new Point(topicMqtt)
    .tag('tagname1', 'tagvalue1')
    .intField('field1', message.toString())

  void setTimeout(() => {
    writeClient.writePoint(point)
  },)
});
```

Figure 23 : Script Javascript pour l'envoi des données dans InfluxDB

4. Serveur

Dans cette partie, les matériels et outils utilisés pour héberger les bases de données et exécuter le script python, vont être détaillés. Le serveur est composé d'un Raspberry Pi qui fait tourner le script python, le site web et docker. Docker héberge plusieurs services :

- Portainer,
- Mosquitto,
- InfluxDB,
- PostgreSQL.

4.1. Raspberry Pi 3 (serveur)

Le Raspberry pi 3 est un nano-ordinateur avec un processeur ARM (Advanced RISC Machine), il possède un module wifi 802.11 2,4 GHz et 5GHz ainsi qu'un module bluetooth 4.0 (BLE), 4 ports USB, un port lan, un port HDMI, un emplacement pour une carte micro-SD et 40 GPIO permettant une connectivité avec divers périphériques. Le Raspberry est utilisé comme un serveur central, toutes les données passent par lui avant d'être affichées sur le site web.

Avant de commencer à utiliser un Raspberry Pi, il faut choisir et mettre en place un système d'exploitation idéal, il existe plusieurs versions différentes, pour ma part la version 12 Bookworm a été utilisée.

Le Raspberry Pi 3 a été choisi pour sa combinaison de performances et de polyvalence, il a tous les atouts pour servir de serveur local.

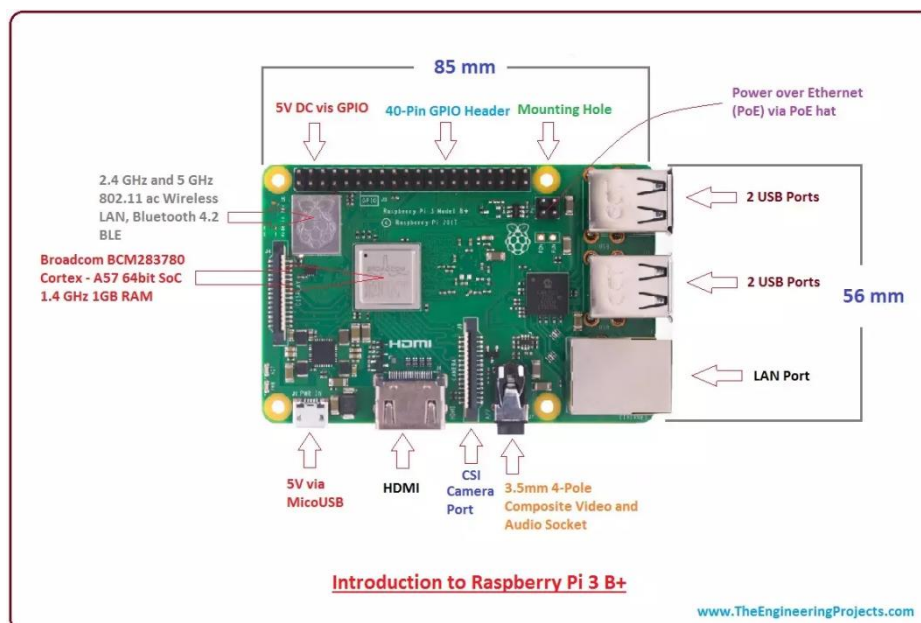


Figure 24 : Raspberry

4.2. Docker

Docker est un logiciel de conteneurisation, qui permet de déployer des applications et leurs dépendances dans des conteneurs qui sont isolés.

Grâce à Docker Compose, il est possible de créer, gérer et déplacer plusieurs applications en utilisant simplement un fichier YAML (Yet Another Markup Language) pour faire la configuration de différentes applications. YAML est un langage de sérialisation de données, utilisé pour écrire des fichiers de configuration dans le cadre d'un déploiement



Figure 25 : logo Docker

```
version: '3.8'

services:
  portainer:
    ports:
      - '9443:9443'
      - '9000:9000'
    container_name: portainer
    image: portainer/portainer-ce
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - data:/data

    #restart: always

volumes:
  data:
```

Figure 26 : Exemple docker compose

Pour lancer un fichier YAML avec docker compose il faut utiliser la commande :

DOCKER COMPOSE UP -D

De plus, Docker est flexible, et offre la possibilité de déployer des applications rapidement dans différent environnement et propose une large gamme d'application facile à mettre en place grâce à sa documentation très bien fournie.

Docker a été utilisé pour créer une stack Portainer qui tourne sur le Raspberry Pi. Le reste des applications ont été créées et déployées directement depuis Portainer, ce qui facilite la gestion des différentes applications.

4.3. Portainer Community Edition

Portainer CE est une image docker avec une interface graphique conviviale et intuitive qui permet de créer, gérer et supprimer des applications docker.



Figure 27 : Logo Portainer

Portainer a permis d'avoir une interface graphique pour mettre en place les autres services nécessaires pour le bon fonctionnement du projet.

Dans Portainer, il y a 2 stacks local, il y a Portainer lui-même et les autres applications regroupées dans un Docker Compose.

Name	State	Quick Actions	Stack	Image	Created	IP Address	Published Ports	Ownership
influxdb	running	[Icons]	mosquitto_nodered	influxdb:latest	2024-03-06 16:15:09	172.20.0.3	8088:8088	administrators
mosquitto	running	[Icons]	mosquitto_nodered	eclipse-mosquitto	2024-03-06 16:15:09	172.20.0.2	1883:1883	administrators
mosquitto_nodered-admin-1	exited	[Icons]	mosquitto_nodered	adminer	2024-03-06 16:15:10	-	-	administrators
mosquitto_nodered-db-1	running	[Icons]	mosquitto_nodered	postgres	2024-03-06 16:15:09	172.20.0.4	-	administrators
portainer	running	[Icons]	docker	portainer/portainer-ce	2024-01-30 14:47:13	172.18.0.2	9000:9000 9443:9443	administrators

Figure 28 : Interface de Portainer

De plus, Portainer a des fonctionnalités de monitoring avancées, permettant de suivre en temps réel l'état des différents stacks. Grâce à cela, il est possible de vérifier simplement si les stacks sont en lancée ou s'il a des problèmes.

4.4. Mosquitto

Mosquitto est une image d'un serveur MQTT (Message Queuing Telemetry Transport) open source qui permet la communication entre divers dispositifs, tel que des objets connecter et des serveurs. Il intègre le chiffrement TLS (Transport Layer Security) ainsi qu'une authentification utilisateur, assurant ainsi la sécurisation des données échangées.

Mosquitto a été utilisé pour la communication entre les systèmes embarqués et la base de données.



Figure 29 : Logo Mosquitto

MQTTx a été utilisé car c'est un gestionnaire de MQTT qui permet de s'abonner à différents topics, envoyer et visualiser des données. Il peut être utilisé en CLI (Command ligne interface) ou bien en GUI (Graphical user interface).

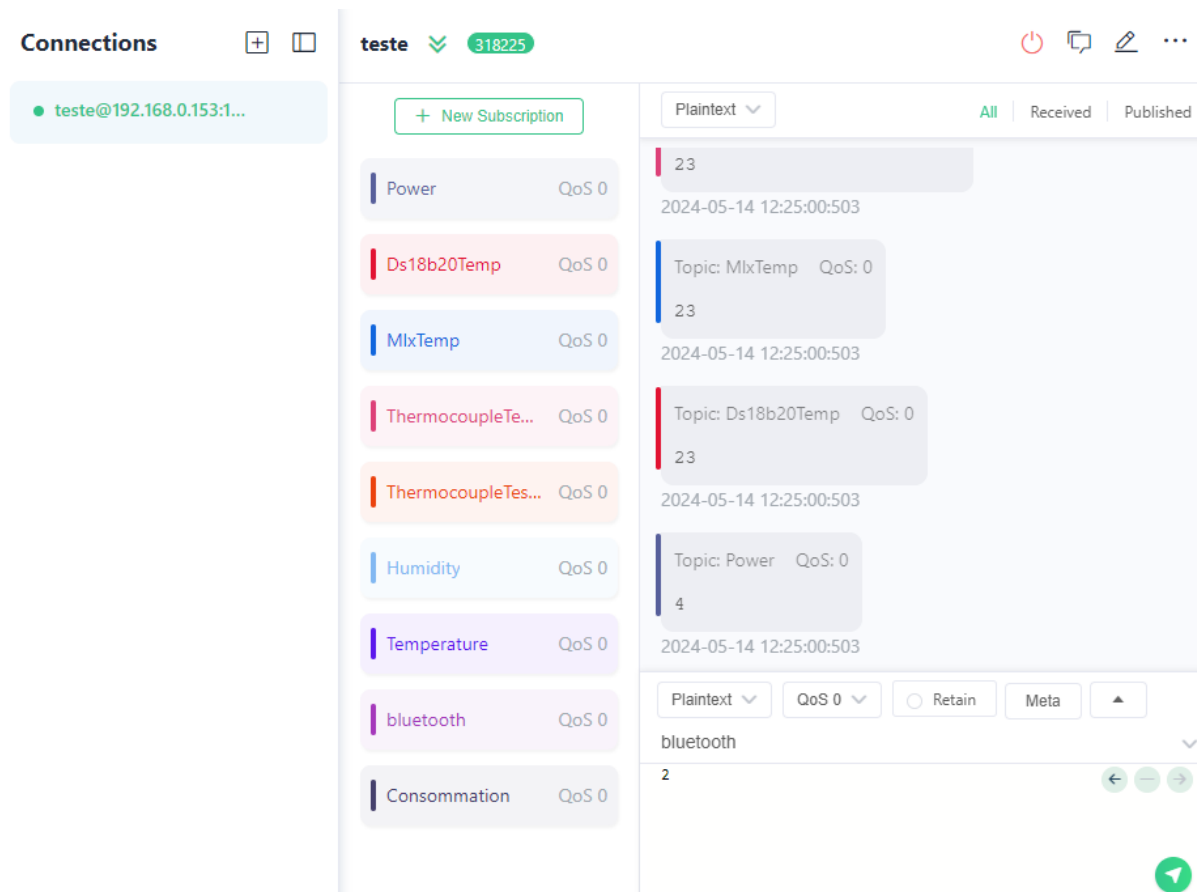


Figure 30 : Interface de MQTTx

4.5. InfluxDB

InfluxDB est une base de données temporelles open source, qui permet de gérer les données en temps réel et a des fonctionnalités avancées pour l'analyse et la visualisation avec un affichage graphique intégré.



Figure 31 : Logo InfluxDB

Influxdb a été choisi et utilisé pour stocker les données reçues des différents systèmes embarqués, permettant d'avoir les données en temps réel. Ce qui permet d'avoir les données reliées à un temps précis.

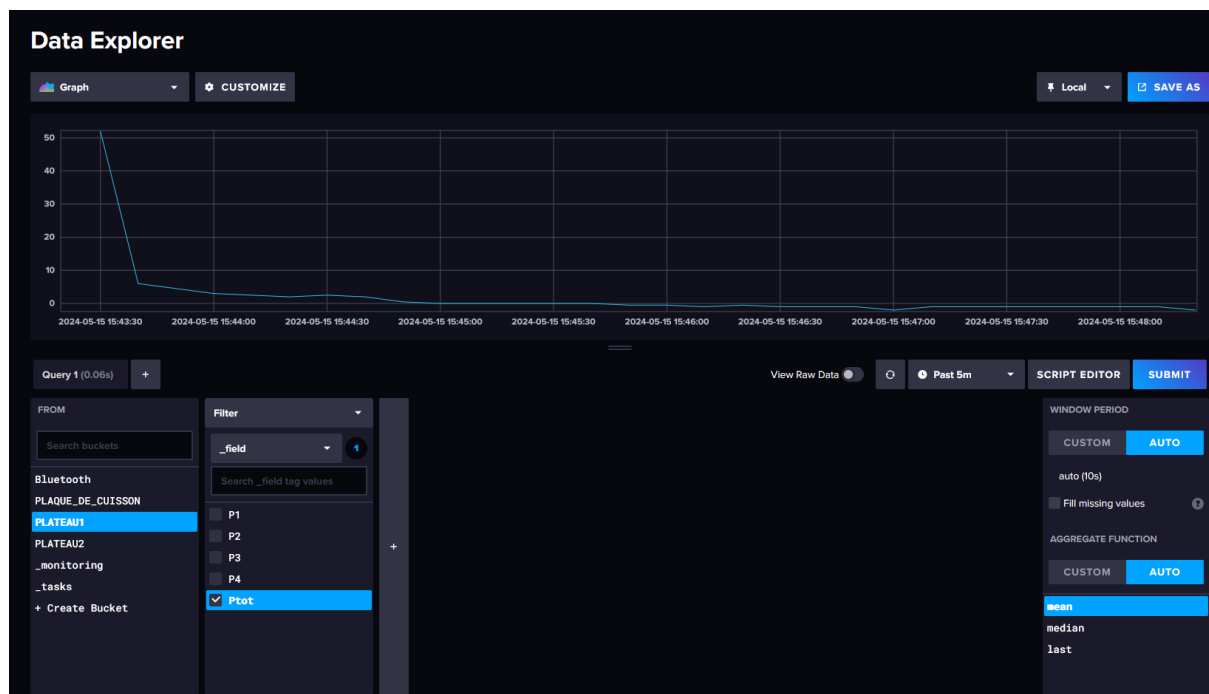


Figure 32 : Interface de InfluxDB

InfluxDB possède une large gamme de librairies client complète, permettant de rajouter des objets connectés de manière rapide et avec plusieurs langages de programmation.

Le code d'exemple fourni par les librairies a été utilisé dans chaque système embarqué ce qui a facilité leur intégration.

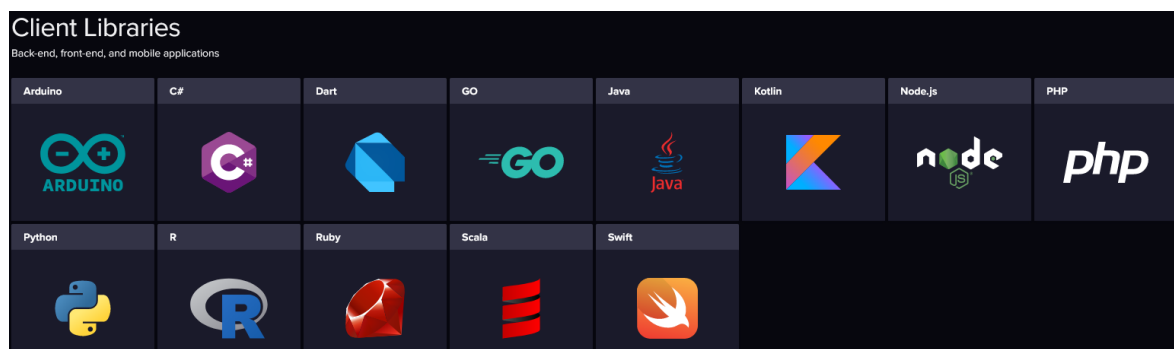


Figure 33 : Interface des librairies de InfluxDB

4.6. PostgreSQL

PostgreSQL est une base de données relationnelle orientée objet open-source. Elle sert à stocker les utilisateurs ainsi que les dates et les heures saisies par les utilisateurs, afin d'afficher les données sous forme de graphique.



Figure 34 : Logo PostgreSQL

Une deuxième base de données a été mise en place, car influxDB n'est pas adaptée pour gérer ce type de données relationnelles.

Prisma a été utilisée pour envoyer et visualiser les données enregistrées dans ma base de données, car il était déjà intégré dans mon projet web. Cependant, il existe d'autres gestionnaires de bases de données comme Adminer ou bien phpMyAdmin qui pourraient également être utilisés pour afficher les données de ma base de données.

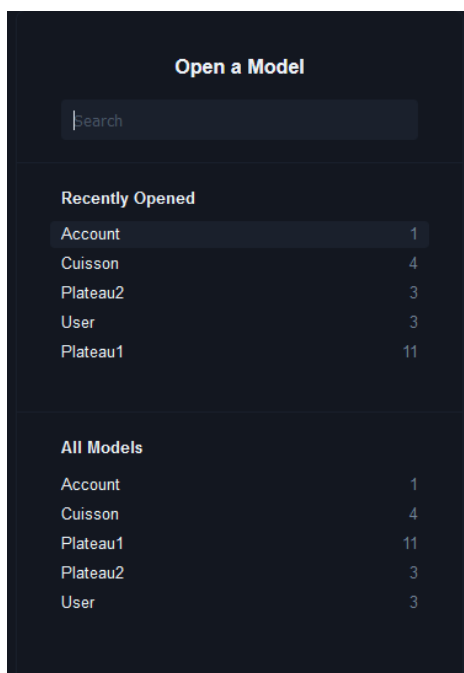


Figure 35 : Interface de Prisma Studio

id A	name A?	email A?	emailVerified B?	username A?	password A?	image A?	role	accounts []
cltn0d44k0000rsebw3y...	user	null	null	user	\$2a\$10\$H7NrVgwCifDVdS...	null	User	Account
clul794dt0000134myhy5...	Admin	null	null	Admin	\$2a\$10\$155moQHLbTmvf9...	null	Admin	Account
clw66btsp0000w5xu6puh...	Samir Aoufia	sa.aoufia@gmail.com	null	null	null	null	User	Account

Figure 36 : Interface de Prisma Studio

5. Application web

Dans cette partie, vont être expliqués les outils et technologies qui ont été utilisés pour création du site web. Le site est séparé en plusieurs parties : le backend et le frontend.

Le backend est la partie invisible, il est le serveur du site web, ce qui permet de gérer la logique du site web les interactions entre les différentes pages, les API internet et externe. Pour le backend ces outils ont été utilisés :

- Nodejs,
- Nextjs,
- Typescript,
- Prisma.

Le frontend est la partie visible, c'est l'interface graphique du site web, qui permet à l'utilisateur d'interagir avec le site et d'afficher les données. Pour le frontend ces outils ont été utilisés :

- Typescript,
- React,
- Shadcn/ui,
- Recharts.

5.1. Nodejs

Nodejs est un environnement polyvalent open-source permettant d'exécuter du javascript ou du typescript en dehors du navigateur web. Il est accompagné d'un grand écosystème de module grâce à son gestionnaire intégré NPM, mais peut aussi en utiliser d'autre tel que YARN, PNPM ou bien BUN. Les modules permettent de rajouter des fonctionnalités.

Nodejs peut être utilisé dans diverses applications, telles que des serveurs web, des applications IOT et des API. Tout cela en utilisant un seul langage de programmation

Nodejs a été utilisé en tant qu'exécuteur de javascript / typescript ce qui veut dire que c'est Nodejs qui va interpréter le code. Pour ajouter les modules dont avait besoin le site web, le gestionnaire de package NPM a été utilisé.



Figure 37 : Logo NodeJS

5.2. Nextjs

Nextjs est un Framework basé sur React qui a été conçu pour construire des stacks d'applications web. Il fusionne les avantages de React et rajoute ces propres fonctionnalités telles que le rendu coté serveur, le routage dynamique et les composants dynamiques. Cette fusion offre une expérience de développement optimal.



Figure 38 : Logo NextJS

Pour créer un projet nextJS on utilise la commande :

NPX CREATE-NEXT-APP@LATEST

Nextjs a été choisi car c'est un framework dit full-stack, a la création d'un nouveau projet, Nextjs mets à disposition : Nodejs, React, typescript et tailwind CSS, sans avoir besoin de les rajouter manuellement avec un ***NPM install***.

Nextjs a également mis en place :

- Server-side rendering (SSR)
 - Le contenu de la page est généré du coté serveur avant de l'envoyer à l'utilisateur, ce qui permet d'améliorer les performances et l'expérience.
- Static site generation (SSG)
 - Des pages entières sont pré-rendues du coté serveur pour être envoyées au client, ce qui permet de charger plus rapidement les pages pour les utilisateurs.
- Single-page application (SPA)
 - La page internet ne charge qu'un seul document en HTML (HyperText Markup Language), puis est mise à jour via des API. Nextjs peut aussi créer des SPAs avec React.

5.3. React

React est une bibliothèque javascript open-source développée par Facebook, il est souvent utilisé pour créer l'interface utilisateur interactive et dynamique.

React utilise des composants permettant de découper l'interface en plusieurs parties réutilisables, ce qui facilite la création, la mise à jour automatique de l'interface et la maintenance du code. Il a été utilisé pour rendre l'application web fluide et interactive.



Figure 39 : Logo React

5.4. TypeScript

TypeScript est un langage de programmation open-source développé par Microsoft. C'est un javascript mais typé, ce qui permet aux développeurs de spécifier des types pour les variables, les paramètres de fonctions et les retours de fonctions. Donc, il permet de voir les erreurs plus facilement avant même l'exécution ce qui rend typescript plus fiable et sécurisé.

Typescript est compilé en simple javascript permettant d'utiliser n'importe quelle librairie javascript en typescript et d'exécuter du typescript sur tous les moteurs javascript.

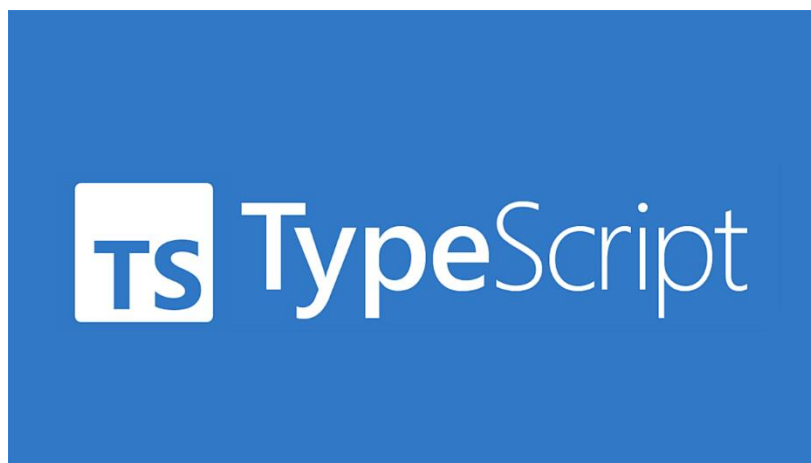


Figure 40 : Logo Typescript

5.5. Prisma

Prisma est un ORM (Object-Relational mapping) moderne et open-source. Il a été conçu pour simplifier l'interaction avec une base de données, il permet de définir des modèles pour créer des tables dans une base de données et de faire des requêtes sans avoir à passer par le langage de la base de données, les requêtes sont écrites en javascript ou typescript, ensuite Prisma s'occupe de faire la traduction.



Figure 41 : Logo Prisma

Des variables d'environnement sont utilisées pour stocker des informations privées, telles que les informations de la base de données, les clés API et d'autres paramètres de configuration. De plus les variables d'environnement permettent de publier le code en toute sécurité sur internet comme GitHub par exemple car les variables sont stockées dans le fichier « .env » que l'on ne publie pas.

Voici un exemple de schéma Prisma :

Le « generator client » est la partie qui dit à Prisma de générer un client, le client est une API javascript qui permet d'interagir avec la base de données.

```
generator client {
  provider = "prisma-client-js"
}
```

Figure 42 : Schéma Prisma

Dans « datasource db », la base de données est déclarée. Le provider spécifie quelle base de données est utilisée, là c'est une postgresql, « URL » est une variable d'environnement utilisée pour obtenir la connexion avec la base de données. Le « directURL » est une autre variable d'environnement qui permet d'avoir une url (Uniform Resource Locator) pour des opérations spécifiques.

```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
  directUrl = env("DATABASE_URL_UNPOOLED")
}
```

Figure 43 : Schéma Prisma

Dans « Model Plateau1 », un modèle est déclaré avec diverses données. Ce modèle permet de créer une table dans PostgreSQL.

```
model Plateau1{
  id Int @id @default(autoincrement())
  name String
  description String
  date String
  debutheure String
  debutminute String
  finheure String
  finminute String
}
```

Figure 44 : Schéma Prisma

Pour envoyer les schémas Prisma que l'on a créés dans notre base de données, il faut utiliser la commande :

NPX PRISMA DB PUSH

Pour avoir un affichage graphique permettant de visualiser les données dans notre base de données, il faut utiliser la commande :

NPX PRISMA STUDIO

Prisma a été utilisé pour simplifier les requêtes et la création de schéma dans la base de données PostgreSQL mais aussi pour visualiser les données contenues dans celle-ci.

5.6. Shadcn/ui



Figure 45 : Logo
Shadcn/UI

Shadcn n'est pas une simple bibliothèque de composants, mais propose directement des composants prêts à l'emploi mis à disposition des développeurs. Il a été utilisé, car il répondait à tous les besoins du site web, sa personnalisation et pour sa simplicité d'utilisation et de modification. De plus, shadcn garantit des performances élevées, assurant une expérience fluide et réactive pour les utilisateurs.

5.7. Recharts

Recharts est une bibliothèque open-source de graphique et de diagrammes créés pour React. Il a été utilisé pour sa simplicité, sa performance et sa compatibilité avec React.

Recharts

Figure 46 : Logo Recharts

5.8. Site web

Le site web se compose de plusieurs parties :

- Administration : gestion utilisateur,
- Consultation des données : Systèmes embarqués (Cuisson, Plateau, Bluetooth),
- Informations : Documentations et données de l'utilisateur,
- API.

Voici un mind map pour visualiser l'arborescence du site web.

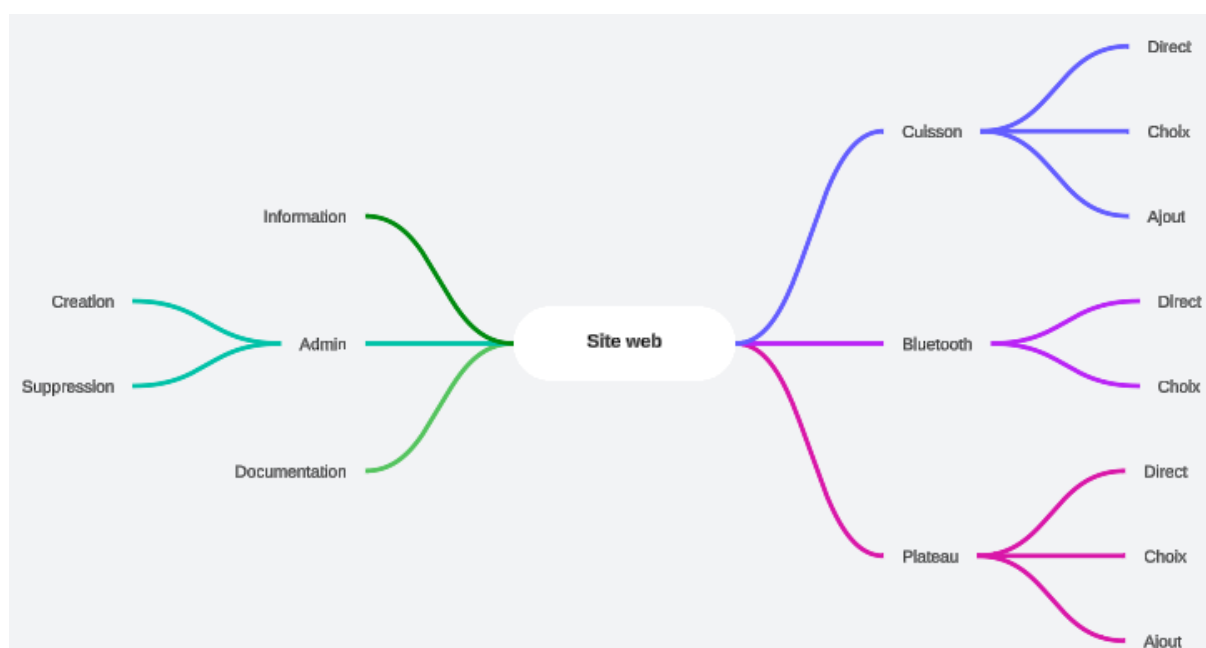


Figure 47 : Mind map du site web

Pour avoir accès au site, il faut être obligatoirement se connecter, il y a 2 façons de se connecter les credenciales (nom d'utilisateur/mot de passe) ou bien par un compte Microsoft.

Pour pouvoir se connecter, il faut demander l'admin de nous crée un compte ou de nous ajouter à l'annuaire de Microsoft.

Pour ajouter une personne via l'annuaire, il faut se rendre dans la partie Microsoft Entra id et ajouter l'email de la personne et lui mettre le rôle de l'invité. L'admin peut facilement supprimer des utilisateurs mais ne peut pas réinitialiser le mot de passe.

Nom d'affichage ↑	Nom d'utilisateur princ...	Type d'utilisateur	Synchronisatio...	Identités	Nom de l'entreprise	Type de création
SA Samir Aoufia	sa.aoufia_gmail.com#EXT...	Membre	Non	MicrosoftAccount		
S Samir2	sa.aoufia_outlook.com#E...	Invité	Non	MicrosoftAccount		Invitation

Figure 48 : Interface de Microsoft Entra ID

Pour ajouter une personne via les credentials, il faut se rendre dans la partie admin ou l'on trouve une carte à remplir avec les informations de la personne. Ici l'administrateur peut supprimer les utilisateurs, mais aussi réinitialiser leur mot de passe.

Welcome

Nom
Nathan

Username
Samir

Mot de passe

Role
Select a role

Créer un compte

[Retour à la liste des utilisateurs](#)

Figure 49 : Formulaire d'ajout d'utilisateur

ID	Username	Nom	Role	Reset MDP	Supprimer Utilisateur
cltn0d44k00009rsebw3yzsfc	user	user	User	Reset	Supprimer
clul794dt0000134myhy5ie8c	Admin	Admin	Admin	Reset	Supprimer

Figure 50 : Tableau d'utilisateur

Quand on arrive sur le site web, il y a 2 boutons : connexion et information.

Le bouton connexion redirige vers la page de connexion permettant la connexion et le bouton information vers une page ou il y a les coordonnées de l'administrateur du site web.

Figure 51 : Formulaire de connexion

Figure 52 : Page d'information

Une fois la connexion faite, il y a les paramètres de l'utilisateur avec le nom, le rôle et le nom de l'utilisateur. Il y a aussi en dessous un lien qui permet d'aller sur la page de réinitialisations de mot de passe. Pour changer de mot de passe on a juste besoin de taper le nouveau, le site se charge de récupérer l'id de l'utilisateur pour faire le changement de mot de passe.

Figure 53 : Paramètres de l'utilisateur

Figure 54 : Changement de mot de passe

On retrouve une barre de navigation, qui permet de naviguer entre toutes les pages.

Information Documentation Cuisson ▾ Plateau ▾ Bluetooth ▾

Figure 55 : Barre de navigation

Pour commencer il y a « information » qui renvoi sur la page précédemment vue. Ensuite il y a cuisson qui permet de rediriger sur 3 autres pages « direct », « ajout » et « choix ».

Direct

Graphique de la cuisson en direct.

Choix

Choix de la date et du repas.

Ajout

Ajouter le repas avec la date.

Figure 56 : Page pour la partie système de cuisson

La page « direct » de cuisson, renvoi sur une page avec un graphique ou il y a 3 boutons pour choisir la durée à afficher en direct : 5 dernières minutes, 15 dernières minutes ou 1 heure, et également 7 boutons pour choisir quelles valeurs à afficher dans le graphique.

GRAPHIQUE EN DIRECT DE LA CUISSON

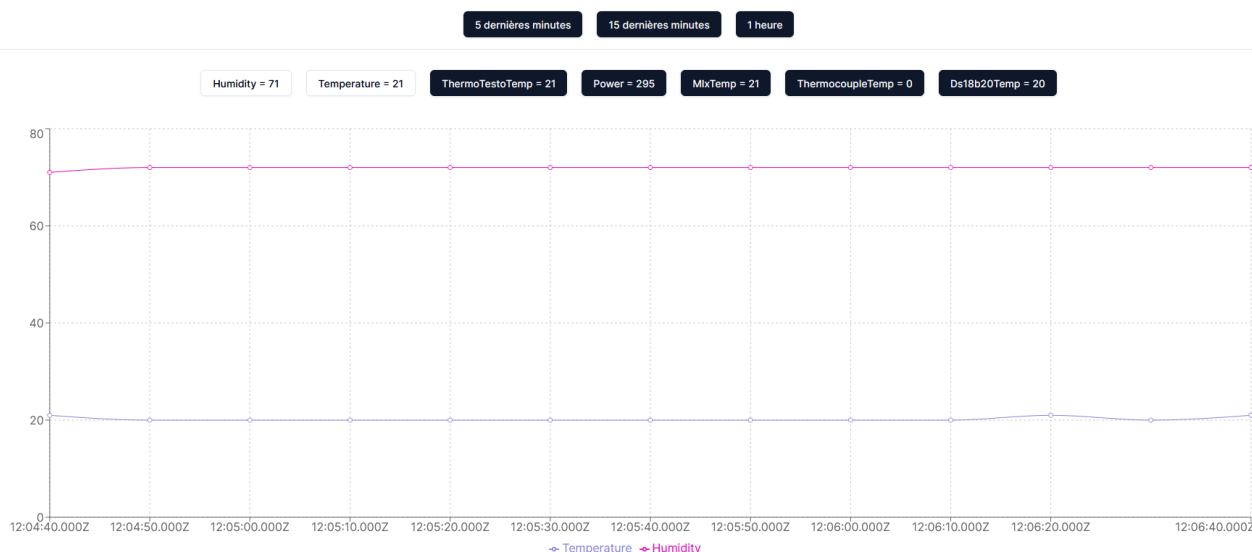



Figure 57 : Graphique en direct de la cuisson

La page « ajout » de cuisson, renvoi sur une page avec un formulaire à remplir avec le nom du plat, une description, une date ainsi que les heures et minute de début et de fin.

Ajout Cuisson

Plat

Description

Date
 

Debut

Heure

Minute

Fin

Heure

Minute

Enregistrer

Figure 58 : Formulaire d'ajout de cuisson

La page « choix » de cuisson, renvoi sur une page avec 2 sélections à faire : une pour choisir le plat et une pour choisir la date. Une fois les sélections faites, une description apparait ainsi qu'un graphique et qu'une datatable pour visualiser les données mais également les différents boutons pour choisir les données.

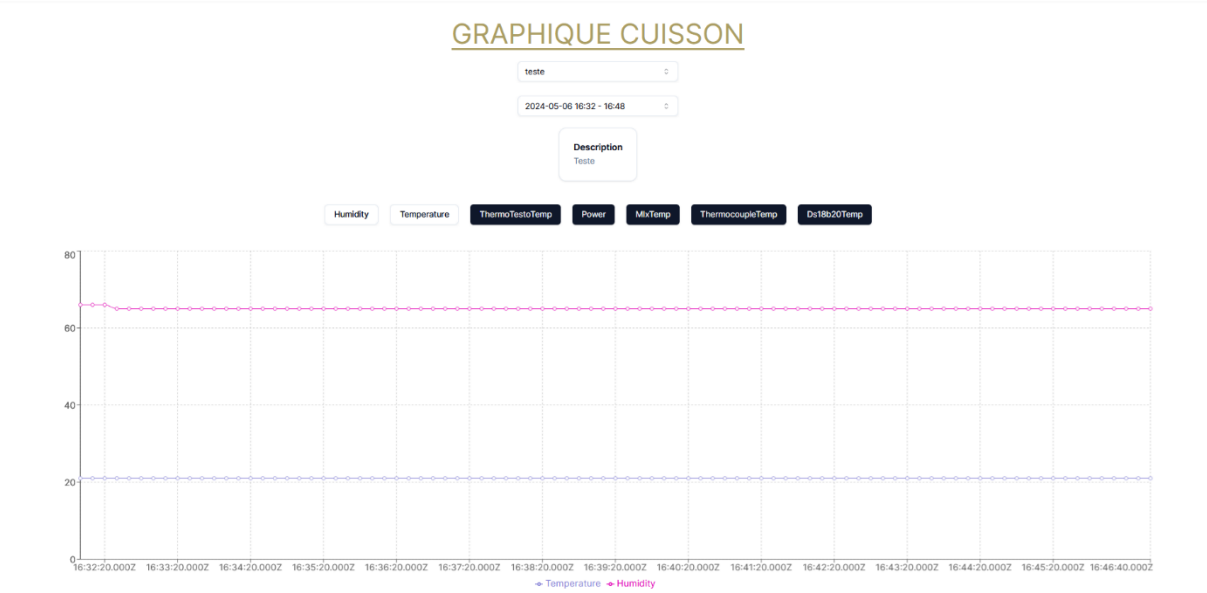


Figure 59 : Graphique de cuisson par rapport au nom et date choisi

Temps	Humidity	Ds18b20Temp	MixTemp	Power	Temperature	ThermocoupleTemp	ThermocoupleTestoTemp
06/05/2024 16:32:00	66	20	22	3	21	21	21
06/05/2024 16:32:10	66	20	21	3	21	21	21
06/05/2024 16:32:20	66	20	21	4	21	21	21
06/05/2024 16:32:30	65	20	21	4	21	21	22
06/05/2024 16:32:40	65	20	21	4	21	21	22
06/05/2024 16:32:50	65	20	21	4	21	21	22
06/05/2024 16:33:00	65	20	21	4	21	21	22
06/05/2024 16:33:10	65	20	21	4	21	21	22
06/05/2024 16:33:20	65	20	21	4	21	21	22
06/05/2024 16:33:30	65	20	21	3	21	21	21

Figure 60 : Datatable de choix de cuisson

Ensuite il y a plateau qui permet de rediriger sur 3 autres pages « direct », « choix » et « ajout ».



Figure 61 : Page pour la partie plateau

La page « direct » de plateau, renvoi sur une page avec une sélection pour savoir quel plateau afficher, après avoir choisi le plateau, il y a 3 boutons pour choisir la durée à afficher en direct : 5 dernières minutes, 15 dernières minutes ou la dernière heure.

GRAPHIQUE EN DIRECT DES PLATEAUX



Figure 62 : Graphique en direct du plateau

La page « ajout » de plateau, renvoi sur une page avec une sélection pour choisir le plateau. Après le choix fait un formulaire apparait, il faut le remplir : avec le nom de la personne, une description, une date ainsi que les heures et minute de début et de fin.

Ajout Personne

plateau1

Nom de la personne

Description

date

Debut

Heure
Minute

Fin

Heure
Minute

enregistrer

Figure 63 : Formulaire d'ajout de personne

La page « choix » du plateau, renvoi sur une page avec 3 sélections à faire pour choisir le plateau, la personne et la date. Après avoir fait ces choix la description apparait ainsi qu'un bouton pour télécharger les données en format csv, un graphique et une datatable pour afficher les données.

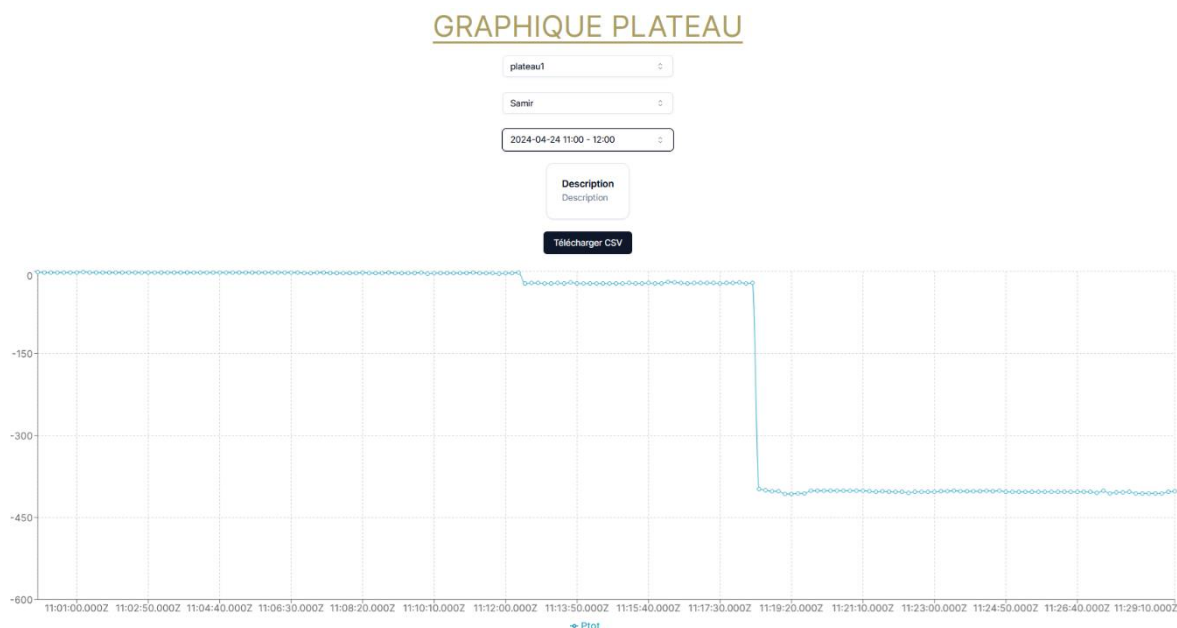


Figure 64 : Graphique du plateau par rapport au nom et date choisi

Colonne ▾

Temps	Poids total
24/04/2024 11:00:00	-1
24/04/2024 11:00:10	-2
24/04/2024 11:00:20	-2
24/04/2024 11:00:30	-2
24/04/2024 11:00:40	-2
24/04/2024 11:00:50	-2
24/04/2024 11:01:00	-2
24/04/2024 11:01:10	-1
24/04/2024 11:01:20	-2
24/04/2024 11:01:30	-2

Previous

Next

Figure 65 : Datatable de choix de plateau

Après il y a bluetooth qui redirige sur 2 pages : « direct » et « choix ».

Direct

Graphique en direct de la connexion bluetooth.

Choix

Choix de la date.

Figure 66 : Page pour la partie bluetooth

La page « direct » de bluetooth, renvoi sur une page avec un graphique ou il y a 3 boutons pour choisir la durée à afficher en direct : 5 dernières minutes, 15 dernières minutes ou la dernière heure.

GRAPHIQUE EN DIRECT BLUETOOTH

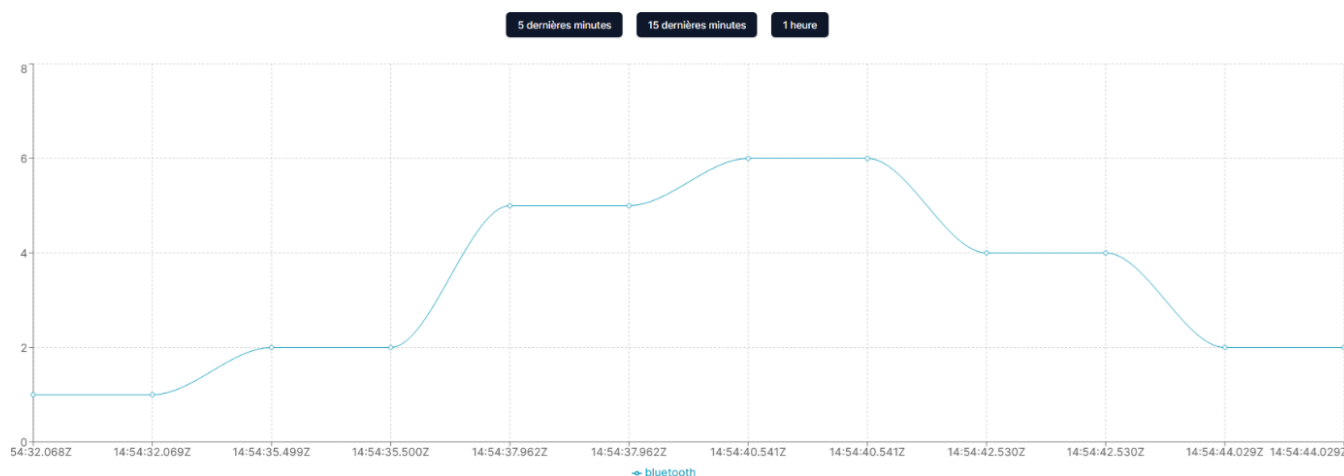


Figure 67 : Graphique en direct de bluetooth

La page choix de bluetooth, renvoi sur une page avec un datepicker et 2 boutons : un pour télécharger les données en format csv et un pour afficher une datatable.

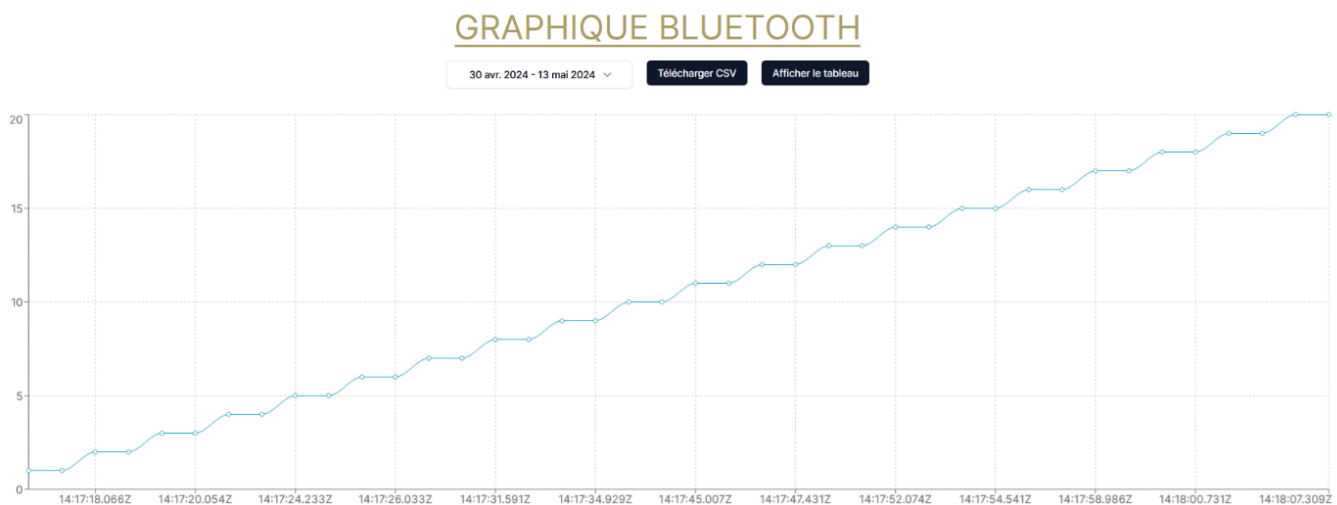


Figure 68 : : Graphique de bluetooth par rapport à la date choisi

Pour finir il y a la page « documentation », sur cette page il y a les instructions pour bien utiliser le site web.

Instructions Utilisateur

Cuisson en Direct
Pour avoir la cuisson en direct, il faut aller sur cette [page](#) et choisir le temps à afficher puis quelle données.

Ajouter une Cuisson
Pour ajouter une cuisson, il faut aller sur cette [page](#) et remplir le formulaire. Il faut bien remplir les heures et les minutes puis enregistrer.

Choix de la cuisson
Pour choisir la date et l'heure, il faut aller sur cette [page](#). Il y aura une sélection à faire contenant tous les noms des plats enregistrés. Si il n'y a rien d'affiché, cela signifie qu'il n'y a pas de plats enregistrés et qu'il faut en enregistrer. Après avoir choisi le plat, une autre sélection apparaîtra avec les dates et heures. Après avoir fait votre sélection, une description, des boutons, un graphique et une datatable apparaîtront. Choisissez quelle valeur afficher dans le graphique en appuyant sur les boutons. La datatable affichera toutes les valeurs sans avoir besoin d'appuyer sur un bouton.

Plateau en Direct
Pour avoir le plateau en direct, il faut aller sur cette [page](#) et sélectionner quel plateau afficher. Des boutons permettront de choisir la durée à afficher.

Ajouter une Personne
Pour ajouter une personne, il faut aller sur cette [page](#) et remplir le formulaire. Il faut d'abord choisir quel plateau, puis bien remplir les heures et les minutes puis enregistrer

Choix de la personne
Pour voir les valeurs des heures entrées pour une personne, il faut aller sur cette [page](#). Il y aura une sélection pour le plateau, puis une sélection avec les noms des personnes enregistrées. Si rien n'est affiché, cela signifie qu'il n'y a pas de personnes enregistrées. Ensuite, une autre sélection apparaîtra avec les dates et heures. Choisissez celle que vous voulez afficher. Après avoir fait votre sélection, une description, des boutons, un graphique et une datatable apparaîtront.

Bluetooth en Direct
Pour avoir les données de Bluetooth en direct, il faut aller sur cette [page](#). Des boutons permettront de choisir la durée à afficher et un graphique sera affiché.

Choix bluetooth
Pour avoir les valeurs de Bluetooth avec une date, il faut aller sur cette [page](#). Il y aura un calendrier et deux boutons. Il faut choisir la date précise et appuyer sur "update". Après cela, le graphique s'affichera. Une fois la date choisie, vous pouvez appuyer sur le bouton "afficher le tableau" pour afficher une datatable.

Figure 69 : Page de documentation

Exemple d'API

Les API sont une partie importante du site web car c'est à grâce au différent API et appelle d'API que le site web fonctionne.

Voici un exemple d'API utiliser pour permettent l'affichage des données de cuisson, quand l'API est appelée, elle fait une requête vers InfluxDB et récupère les 30 derniers jours. Ensuite, il y a un tri des données reçu pour envoyer au site web seulement le nom, le temps et la donnée. Après avoir fait le tri, l'API renvoie les données en format JSON.

```
import { InfluxDB } from '@influxdata/influxdb-client';
import { NextResponse } from 'next/server';

const influxdb = new InfluxDB({
  url: 'http://192.168.0.153:8086',
  token: 'Grm2Vpl_k9QB7kaoIpCMwib4KICZ0nHuVyP88RgmNU-5zYVNuCwChNH5GbToM7yHFGzyH6MwngiQI093GmijxQ==',
});

const queryApi = influxdb.getQueryApi('17980c3d8cf14b98');

export async function GET() {

  const fluxQuery = 'from(bucket: "PLAQUE_DE_CUISSON") |> range(start: -30d) |> drop(columns: ["_start", "_stop", "_field", "table", "tagname1", "result", "table"]) |> truncateTimeColumn(unit: 1s) |> pivot(rowKey:["_time"], columnKey: ["_measurement"], valueColumn: "_value")';

  const data = await queryInfluxDB(fluxQuery);

  return NextResponse.json(data);
}

async function queryInfluxDB(fluxQuery: string) {
  const result = await queryApi.collectRows(fluxQuery);
  return result;
}
```

Figure 70 : Exemple d'API

6. Conclusion

Ce stage m'a permis de développer plusieurs types de compétence sociale et informatique.

Pour la partie informatique, j'ai surtout approfondi mes compétences dans l'IoT, en travaillant sur différents systèmes embarqués déjà existant avec plusieurs modes de communication tels que le wifi et le bluetooth. J'ai pu approfondir mes compétences en développement web, ce qui m'a permis d'avoir une meilleure compréhension globale du fonctionnement d'un site web. De plus, ce projet m'a permis l'utilisation de nouveaux outils comme influxDB et PostgreSQL.

Pour ce qui est des compétences sociales, cela m'a permis de comprendre le monde professionnel et son fonctionnement. De plus pour trouver ce stage j'ai dû faire de la recherche de stage et passer des entretiens.

7. Lexique

IoT : Internet of Things

MQTT : Message Queuing Telemetry Transport

I2C : Inter Integrated Circuit

SPI : Serial Peripheral Interface

PWM : Pulse Width Modulation

GPIO : General Purpose Input/Output

BLE : Bluetooth Low Energy

API : Application Programming Interface

Framework : cadre de travail.

Datapicker : un composant permettant de sélectionner une date sur un calendrier.

8. Médiagraphie

InfluxDB OSS v2 Documentation. (s. d.). InfluxData Inc.

<https://docs.influxdata.com/influxdb/v2/>

Docs. (s. d.). Next.js. <https://nextjs.org/docs>

Recharts. (s. d.). <https://recharts.org/>

Shadcn. (s. d.). *shadcn/ui*. Shadcn/Ui. <https://ui.shadcn.com/>

ESP-IDF Programming Guide - ESP32 - — ESP-IDF Programming Guide v5.2.2

documentation. (s. d.). <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html>

ATWINC1510. (s. d.). <https://www.microchip.com/en-us/product/atwinc1510#>

Arduino GIGA R1 WiFi. (s. d.). Arduino Official Store. <https://store.arduino.cc/products/giga-r1-wifi>

Arduino Mega 2560 rev3. (s. d.). Arduino Official Store.

<https://store.arduino.cc/products/arduino-mega-2560-rev3>

ESP8266WiFi library — ESP8266 Arduino Core 3.1.2-21-ga348833 documentation. (s. d.).

<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>

PostgreSQL : documentation. (s. d.). The PostgreSQL Global Development Group.

<https://www.postgresql.org/docs/>

Prisma Documentation. (s. d.). <https://www.prisma.io/docs>

Index | Node.js v22.2.0 Documentation. (s. d.). <https://nodejs.org/docs/latest/api/>

Docker : Accelerated Container Application Development. (2024, 20 mai). Docker.

<https://www.docker.com/>

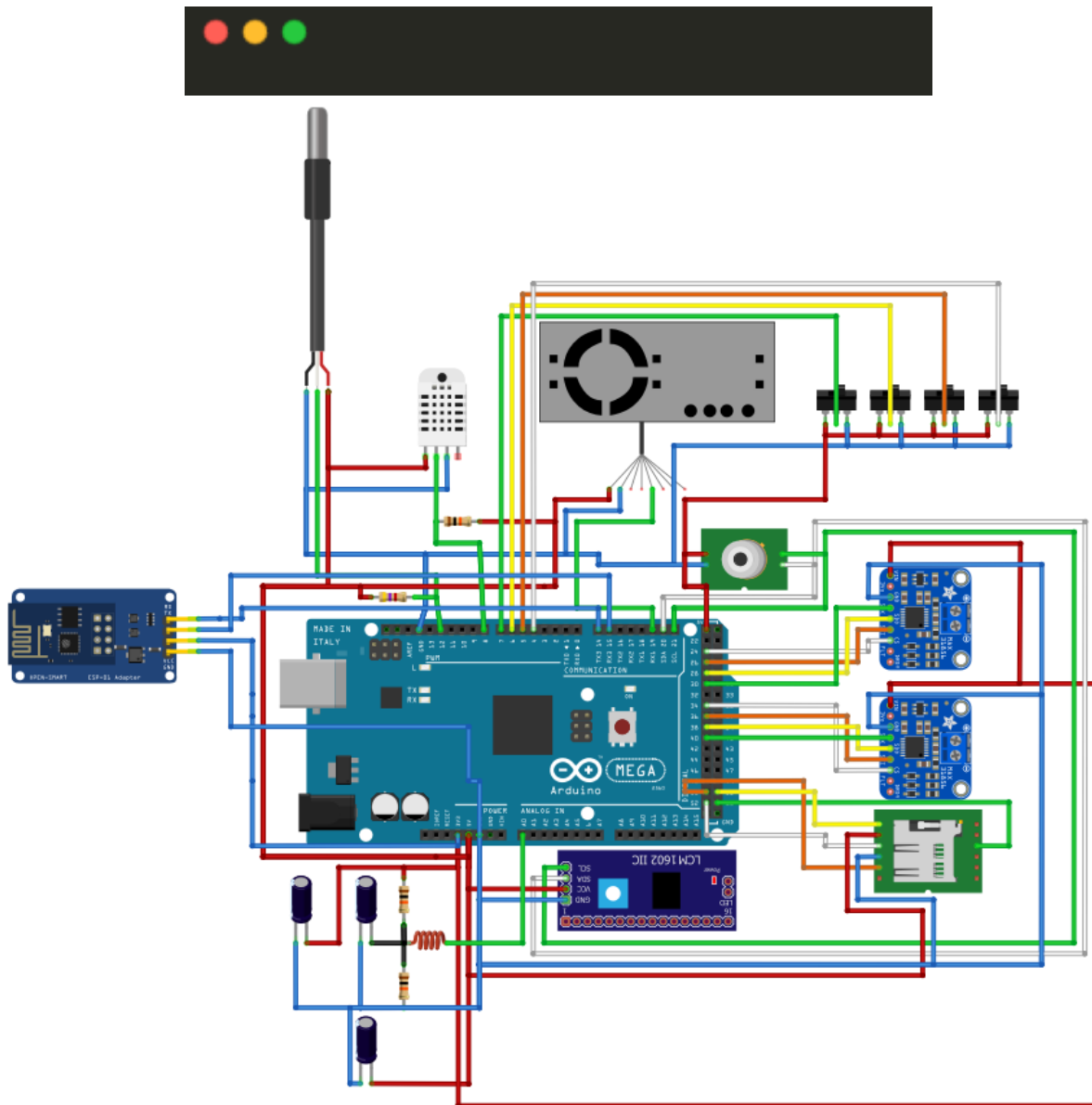
Welcome | 2.19 | Portainer Documentation. (s. d.). <https://docs.portainer.io/>

React. (s. d.). <https://fr.react.dev/>

Documentation. (2020, 6 juillet). Eclipse Mosquitto. <https://mosquitto.org/documentation/>

9. Annexes

9.1. Schéma Prisma complet



9.2. Schéma système de cuisson

9.3. Cahier des charges

Cahier des charges transmis le 30-10-23 par Madame/Monsieur Codutti Nathan de la société Smart Gastronomy Lab – Université de Liège située à Gembloux à l'étudiant de la HELHa Aoufia Samir dans le cadre de son stage de fin d'études.

Titre du projet : Conception d'une solution intégrée pour l'acquisition, le stockage et la visualisation de données.

Présentation du cadre : Le Smart Gastronomy Lab est un centre d'innovation, de prototypage et de recherche alimentaires et culinaires. Il travaille sur le développement de l'alimentation du futur en développant des procédés et produits plus sains, durables et personnalisés via l'intégration des nouvelles technologies. Il développe 3 types d'activités :

1) L'accompagnement à l'innovation : Le Smart Gastronomy Lab accompagne les porteurs de projet, start-up et entreprises dans le prototypage, le développement et les phases de test de leurs produits alimentaires. Depuis sa création en 2015, the Smart Gastronomy Lab a accompagné plus de 200 porteurs de projet et entreprises.

2)Le développement de nouvelles technologies : Grâce à sa vision transversale de l'alimentation et sa méthodologie centrée sur l'utilisateur, le Smart Gastronomy Lab se positionne à la source de l'innovation de rupture en cohérence avec les nouveaux besoins du consommateur

3)De la recherche universitaire : le Smart Gastronomy Lab développe de nombreux projets de recherche alliant sciences culinaires et nouvelles technologies. L'objectif est de proposer des solutions pour un système alimentaire durable, sain et personnalisé.

En outre le SGL possède un laboratoire culinaire de prototypage de produits, un laboratoire de recherche, un laboratoire de fabrication et d'électronique.

Résumé du travail à effectuer : Le stagiaire sera chargé de mener à bien un projet visant à améliorer la connectivité et la gestion des appareils présents au sein du laboratoire. Ce projet comprend plusieurs composantes essentielles :

1) Connectivité des Appareils :

- Utilisation d'un module approprié pour établir une connexion réseau entre les appareils du laboratoire.
- Modification du code des appareils existants, le cas échéant, pour les rendre compatibles avec le réseau.

2) Gestion du Réseau :

- Mise en place d'une infrastructure réseau permettant la connectivité et la déconnexion des appareils en toute simplicité.
- Assurer une gestion efficace de la configuration réseau pour garantir la stabilité et la sécurité.

3) Base de Données :

- Mise en œuvre d'une ou de plusieurs bases de données pour stocker les données générées par les appareils.
- Assurer la sécurisation et la sauvegarde régulière des données.

4) Création d'une API :

- Développement d'une interface de programmation (API) pour permettre la récupération des données stockées.
- Assurer que l'API soit documentée et facile à utiliser pour les applications tierces.

5) Application Web :

- Conception et développement d'une application web conviviale pour la visualisation des données collectées.
- Intégration d'outils de visualisation de données pour une expérience utilisateur optimale.
- Optimisation de l'interface ainsi que des données qui y seront affichées pour que celui-ci soit utilisable par tous.

Le stagiaire devra collaborer étroitement avec l'équipe et suivre les meilleures pratiques en matière de développement, de sécurité et de gestion de données. Tout au long du projet, il sera encouragé à résoudre des problèmes de manière autonome, à rechercher des solutions innovantes et à documenter ses travaux.

Détail des éléments techniques/technologiques à maîtriser :

Développement/maitrise des systèmes embarqués (c++). Maitrise de Linux et du réseau. Création/gestion de base de données. Création/compréhension d'API.

Création d'application WEB. Maitriser la lecture/création de documentation technique.

9.4. Code esp-01S

```
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <PubSubClient.h>

SoftwareSerial mySerial(0, 2); // RX, TX

//wifi
const char* ssid = "TP-Link_3406";
const char* password = "43241668";

//mqtt
const char* mqtt_server = "192.168.0.153"; //ip du pc portable
const int mqttPort = 1883;
ESP8266WiFiMulti WiFiMulti;
WiFiClient espClient;
PubSubClient client(espClient);

void setup()
{
    Serial.begin(115200);
    //Serial.println("Hello");
    mySerial.begin(115200);
    setup_wifi();
    setup_mqtt();
    delay(5000);
}

void loop() {
    String IncomingString = "";
```

```
boolean StringReady = false;
while (mySerial.available()) {
    IncomingString = mySerial.readStringUntil('\n');
    StringReady = true;
}
if (StringReady) {
    String variables[7];
    int startIndex = 0;
    for (int i = 0; i < 6; i++) {
        int commaIndex = IncomingString.indexOf(',', startIndex);
        if (commaIndex != -1) {
            variables[i] = IncomingString.substring(startIndex, commaIndex);
            startIndex = commaIndex + 1;
        }
    }
    variables[6] = IncomingString.substring(startIndex);
    // Nommer vos topics MQTT spécifiques
    const char* topics[] = {"Humidity", "Temperature", "Power", "Ds18b20Temp",
    "MlxTemp", "ThermocoupleTemp", "ThermocoupleTestoTemp"};
    for (int i = 0; i < 7; i++) {
        int value = variables[i].toInt();
        // Publier chaque variable dans un topic MQTT spécifique
        client.publish(topics[i], String(value).c_str());
    }
    delay(1000);
    client.loop();
}

void setup_mqtt(){
    client.setServer(mqtt_server, mqttPort);
    client.setCallback(callback); //Déclaration de la fonction de souscription
```

```
    reconnect();
}

void reconnect(){
    while (!client.connected()) {
        Serial.println("Connection au serveur MQTT ...");
        if (client.connect("ESP32Client")) {
            Serial.println("MQTT connecté");
            client.subscribe("value");
        }
        else {
            Serial.print("echec, code erreur= ");
            Serial.println(client.state());
            Serial.println("nouvel essai dans 2s");
            delay(2000);}}}

void setup_wifi(){
    Serial.begin(115200);
    Serial.println();
    WiFi.begin(ssid, password);
    Serial.print("Connecting");
    while (WiFi.status() != WL_CONNECTED)
    { delay(500);
      Serial.print(".");}
    Serial.println();
    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());}

void callback(char* topic, byte *payload, unsigned int length) {
    Serial.println("-----Nouveau message du broker mqtt-----");
    Serial.print("Canal:");
    Serial.println(topic);
    Serial.print("donnee:");
    Serial.write(payload, length);
```



```
Serial.println();  
}
```

9.5. Code Bluetooth

```
#include <BLEDevice.h>  
#include <BLEServer.h>  
#include <BLEUtils.h>  
#include <BLE2902.h>  
  
BLEServer* pServer = NULL;  
BLECharacteristic* pCharacteristic = NULL;  
bool deviceConnected = false;  
bool oldDeviceConnected = false;  
uint32_t value = 0;  
  
#define SERVICE_UUID      "4fafc201-1fb5-459e-8fcc-c5c9c331914b"  
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"  
  
class MyServerCallbacks: public BLEServerCallbacks {  
    void onConnect(BLEServer* pServer) {  
        deviceConnected = true;  
    };  
    void onDisconnect(BLEServer* pServer) {  
        deviceConnected = false;  
    }  
};  
  
void setup() {  
    Serial.begin(115200);  
    // Create the BLE Device  
    BLEDevice::init("ESP32");  
  
    // Create the BLE Server  
    pServer = BLEDevice::createServer();  
    pServer->setCallbacks(new MyServerCallbacks());
```

```
// Create the BLE Service
BLEService *pService = pServer->createService(SERVICE_UUID);

// Create a BLE Characteristic
pCharacteristic = pService->createCharacteristic(
    CHARACTERISTIC_UUID,
    BLECharacteristic::PROPERTY_READ |
    BLECharacteristic::PROPERTY_WRITE |
    BLECharacteristic::PROPERTY_NOTIFY |
    BLECharacteristic::PROPERTY_INDICATE
);

// Create a BLE Descriptor
pCharacteristic->addDescriptor(new BLE2902());

// Start the service
pService->start();

// Start advertising
BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
pAdvertising->addServiceUUID(SERVICE_UUID);
pAdvertising->setScanResponse(false);
pAdvertising->setMinPreferred(0x0); // set value to 0x00 to not advertise this
parameter
BLEDevice::startAdvertising();
Serial.println("Waiting a client connection to notify...");
}

void loop() {
    // notify changed value
    if (deviceConnected) {
        pCharacteristic->setValue((uint8_t*)&value, 4);
        pCharacteristic->notify();
        value++;

        delay(100); // bluetooth stack will go into congestion, if too many packets are
        sent, in 6 hours test i was able to go as low as 3ms
    }
}
```

```
// disconnecting
if (!deviceConnected) {
    delay(5000); // give the bluetooth stack the chance to get things ready
    pServer->startAdvertising(); // restart advertising
    Serial.println("start advertising");
    oldDeviceConnected = deviceConnected;
}
// connecting
if (deviceConnected && !oldDeviceConnected) {
    // do stuff here on connecting
    Serial.println("Connector");
    oldDeviceConnected = deviceConnected;
}
}
```

9.6. Code système de cuisson

```
// Auteur: Codutti Nathan + rajoute d'un module wifi par Samir Aoufia  
// Entreprise: Smart Gastronomy Lab / Laboratoire des sciences gastronomiques  
(ULG: Université de Liège)  
// Date: 09-2023  
// Version: 1.0  
// Description: Code pour le boîtier de monitoring de cuisson  
// Microcontrôleur: Arduino Mega 2560
```

```
// Librairies  
#include <SPI.h>  
#include <SD.h>  
#include <EmonLib.h>  
#include <DHT.h>  
#include <LiquidCrystal_I2C.h>  
#include "Adafruit_PM25AQI.h"  
#include <Adafruit_MLX90614.h>  
#include <Adafruit_MAX31856.h>  
#include <DallasTemperature.h>  
#include <OneWire.h>  
// #include <avr/dtostrf.h>  
// Définitions  
#define DHT_PIN 8  
#define DHT_TYPE DHT22  
#define AC_VOLTAGE 230  
#define ONE_WIRE_BUS 12  
#define SD_CHIP_SELECT 53  
// instances  
DHT dht(DHT_PIN, DHT_TYPE);  
LiquidCrystal_I2C lcd(0x27, 20, 4);  
EnergyMonitor emon1;
```

```
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
Adafruit_MAX31856 maxthermo = Adafruit_MAX31856(34, 36, 38, 40);
Adafruit_MAX31856 maxthermoTesto = Adafruit_MAX31856(24, 26, 28, 30);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
char dataString[200];
char humidityStr[10];
char temperatureStr[10];
char ds18b20TempStr[10];
char mlxTempStr[10];
char thermocoupleTempStr[10];
char thermocoupleTestoTempStr[10];
const byte DS_SWITCH = 7;
const byte MLX_SWITCH = 5;
const byte THERMO_SWITCH = 4;
const byte THERMOTESTO_SWITCH = 6;
float ds18b20Temp = 0.0;
float mlxTemp = 0.0;
float thermocoupleTemp = 0.0;
float thermocoupleTestoTemp = 0.0;
byte DS_SWITCH_ETAT;
byte MLX_SWITCH_ETAT;
byte THERMO_SWITCH_ETAT;
byte THERMOTESTO_SWITCH_ETAT;
void initThermo(){
    maxthermo.begin();
    maxthermo.setThermocoupleType(MAX31856_TCTYPE_K);
    maxthermo.setConversionMode(MAX31856_CONTINUOUS);
    lcd.setCursor(0, 2);
    lcd.print("Thermo init ");
}
```

```
void initTesto(){
    maxthermoTesto.begin();
    maxthermoTesto.setThermocoupleType(MAX31856_TCTYPE_T);
    maxthermoTesto.setConversionMode(MAX31856_CONTINUOUS);
    //maxthermo.setConversionMode(MAX31856_ONESHOT_NOWAIT);
    lcd.setCursor(0, 3);
    lcd.print("ThermoTesto init ");
}

void initDS(){
    DS18B20.begin();
    lcd.setCursor(0, 0);
    lcd.print("DS init");
}

void initMLX(){
    lcd.setCursor(0, 1);
    if (!mlx.begin()) {
        lcd.print("Error connecting to MLX sensor. Check wiring.");
    }
    else {
        lcd.print("MLX init");
    }
}

void setup() {
    Serial.begin(115200); //////////////////////////////////////
    pinMode(DS_SWITCH, INPUT);
    pinMode(MLX_SWITCH, INPUT);
    pinMode(THERMO_SWITCH, INPUT);
    pinMode(THERMOTESTO_SWITCH, INPUT);
    delay(250);
    DS_SWITCH_ETAT = digitalRead(DS_SWITCH);
    MLX_SWITCH_ETAT = digitalRead(MLX_SWITCH);
}
```

```
THERMO_SWITCH_ETAT = digitalRead(THERMO_SWITCH);
THERMOTESTO_SWITCH_ETAT = digitalRead(THERMOTESTO_SWITCH);
lcd.init();
lcd.backlight();
dht.begin();
emon1.current(A0, 28);
if (!SD.begin(SD_CHIP_SELECT)) {
    lcd.print("Card not found");
}
else {
    lcd.print("Card init");
    lcd.setCursor(0, 1);
}
Serial1.begin(9600); //TX pin19
delay(1000);
lcd.clear();
delay(100);
if(DS_SWITCH_ETAT)
    initDS();
if(MLX_SWITCH_ETAT)
    initMLX();
if(THERMO_SWITCH_ETAT)
    initThermo();
if(THERMOTESTO_SWITCH_ETAT)
    initTesto();
File dataFile = SD.open("data.csv", FILE_WRITE);
if (dataFile) {
    dataFile.println("Time; Humidity; Temperature; Power; DSTemp; IRTemp;
ThermocoupleTemp; ThermocoupleTestoTemp");
}
dataFile.close();
```

```
    delay(2000);
    lcd.clear();
}

void loop() {
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();
    double currentRMS = emon1.calcIrms(11500);
    int power = AC_VOLTAGE * (currentRMS / 4);

    if (DS_SWITCH_ETAT)
    {
        DS18B20.requestTemperatures();
        ds18b20Temp = DS18B20.getTempCByIndex(0);
    }
    if (MLX_SWITCH_ETAT)
    {
        mlxTemp = mlx.readObjectTempC();
    }
    if (THERMO_SWITCH_ETAT)
    {
        thermocoupleTemp = maxthermo.readThermocoupleTemperature();
    }
    if (THERMOTESTO_SWITCH_ETAT)
    {
        thermocoupleTestoTemp = maxthermoTesto.readThermocoupleTemperature();
    }

    displayData(humidity, temperature, power, ds18b20Temp, mlxTemp,
thermocoupleTemp, thermocoupleTestoTemp);

    writeDataToSD(humidity, temperature, power, ds18b20Temp, mlxTemp,
thermocoupleTemp, thermocoupleTestoTemp);

    Serial.write(String(humidity).c_str());////////////////////
    Serial.write(",");
}
```



```

Serial.write(String(temperature).c_str());//////////
Serial.write(",");
Serial.write(String(power).c_str());//////////
Serial.write(",");
Serial.write(String(ds18b20Temp).c_str());//////////
Serial.write(",");
Serial.write(String(mlxTemp).c_str());//////////
Serial.write(",");
Serial.write(String(thermocoupleTemp).c_str());//////////
Serial.write(",");
Serial.write(String(thermocoupleTestoTemp).c_str());//////////
delay(1000);
}

void writeToSD(float humidity, float temperature, int power, float ds18b20Temp,
float mlxTemp, float thermocoupleTemp, float thermocoupleTestoTemp) {
    unsigned long time = millis() / 1000;
    dtostrf(humidity, 6, 2, humidityStr);
    dtostrf(temperature, 6, 2, temperatureStr);
    dtostrf(ds18b20Temp, 6, 2, ds18b20TempStr);
    dtostrf(mlxTemp, 6, 2, mlxTempStr);
    dtostrf(thermocoupleTemp, 6, 2, thermocoupleTempStr);
    dtostrf(thermocoupleTestoTemp, 6, 2, thermocoupleTestoTempStr);
    snprintf(dataString, sizeof(dataString), "%lu;%s;%s;%d;%s;%s;%s;%s", time,
humidityStr, temperatureStr, power, ds18b20TempStr, mlxTempStr,
thermocoupleTempStr, thermocoupleTestoTempStr);
    File dataFile = SD.open("data.csv", FILE_WRITE);
    if (dataFile) {
        dataFile.println(dataString);
        dataFile.close();
    }
}

```

```
void displayData(float humidity, float temperature, int power, float ds18b20Temp,
float mlxTemp, float thermocoupleTemp, float thermocoupleTestoTemp) {

    lcd.clear();

    if (DS_SWITCH_ETAT && MLX_SWITCH_ETAT && THERMO_SWITCH_ETAT &&
THERMOTESTO_SWITCH_ETAT) // 1*4
    {
        lcd.setCursor(0, 0);
        lcd.print("Liquid Temp: ");
        lcd.print(ds18b20Temp);
        lcd.print("*C");
        lcd.setCursor(0, 1);
        lcd.print("Object Temp: ");
        lcd.print(mlxTemp);
        lcd.print("*C");
        lcd.setCursor(0, 2);
        lcd.print("Thermocouple: ");
        lcd.print(thermocoupleTemp);
        lcd.print("*C");
        lcd.setCursor(0, 3);
        lcd.print("Testo: ");
        lcd.print(thermocoupleTestoTemp);
        lcd.print("*C");
    }
    else if (DS_SWITCH_ETAT && MLX_SWITCH_ETAT && THERMO_SWITCH_ETAT)
    {
        lcd.setCursor(0, 0);
        lcd.print("Humidity: ");
        lcd.print(humidity);
        lcd.print("%");
        lcd.setCursor(0, 1);
        lcd.print("Liquid Temp: ");
        lcd.print(ds18b20Temp);
```

```
    lcd.print("*C");
    lcd.setCursor(0, 2);
    lcd.print("Object Temp: ");
    lcd.print(mlxTemp);
    lcd.print("*C");
    lcd.setCursor(0, 3);
    lcd.print("Thermocouple: ");
    lcd.print(thermocoupleTemp);
    lcd.print("*C");
}

else if (DS_SWITCH_ETAT && MLX_SWITCH_ETAT &&
THERMOTESTO_SWITCH_ETAT) //2*3
{
    lcd.setCursor(0, 0);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");
    lcd.setCursor(0, 1);
    lcd.print("Liquid Temp: ");
    lcd.print(ds18b20Temp);
    lcd.print("*C");
    lcd.setCursor(0, 2);
    lcd.print("Object Temp: ");
    lcd.print(mlxTemp);
    lcd.print("*C");
    lcd.setCursor(0, 3);
    lcd.print("Testo: ");
    lcd.print(thermocoupleTestoTemp);
    lcd.print("*C");
}

else if (DS_SWITCH_ETAT && THERMO_SWITCH_ETAT &&
THERMOTESTO_SWITCH_ETAT) // 3*3
```

```
{  
    lcd.setCursor(0, 0);  
    lcd.print("Humidity: ");  
    lcd.print(humidity);  
    lcd.print("%");  
    lcd.setCursor(0, 1);  
    lcd.print("Liquid Temp: ");  
    lcd.print(ds18b20Temp);  
    lcd.print("*C");  
    lcd.setCursor(0, 2);  
    lcd.print("Thermocouple: ");  
    lcd.print(thermocoupleTemp);  
    lcd.print("*C");  
    lcd.setCursor(0, 3);  
    lcd.print("Testo: ");  
    lcd.print(thermocoupleTestoTemp);  
    lcd.print("*C");  
}  
  
else if (MLX_SWITCH_ETAT && THERMO_SWITCH_ETAT &&  
THERMOTESTO_SWITCH_ETAT) // 4*3  
{  
    lcd.setCursor(0, 0);  
    lcd.print("Humidity: ");  
    lcd.print(humidity);  
    lcd.print("%");  
    lcd.setCursor(0, 1);  
    lcd.print("Object Temp: ");  
    lcd.print(mlxTemp);  
    lcd.print("*C");  
    lcd.setCursor(0, 2);  
    lcd.print("Thermocouple: ");
```

```
lcd.print(thermocoupleTemp);  
lcd.print("*C");  
lcd.setCursor(0, 3);  
lcd.print("Testo: ");  
lcd.print(thermocoupleTestoTemp);  
lcd.print("*C");  
}  
else if (DS_SWITCH_ETAT && MLX_SWITCH_ETAT) // 1*2  
{  
    lcd.setCursor(0, 0);  
    lcd.print("Humidity: ");  
    lcd.print(humidity);  
    lcd.print("%");  
    lcd.setCursor(0, 1);  
    lcd.print("Temperature: ");  
    lcd.print(temperature);  
    lcd.print("*C");  
    lcd.setCursor(0, 2);  
    lcd.print("Liquid Temp: ");  
    lcd.print(ds18b20Temp);  
    lcd.print("*C");  
    lcd.setCursor(0, 3);  
    lcd.print("Object Temp: ");  
    lcd.print(mlxTemp);  
    lcd.print("*C");  
}  
else if (DS_SWITCH_ETAT && THERMO_SWITCH_ETAT) // 2*2  
{  
    lcd.setCursor(0, 0);  
    lcd.print("Humidity: ");  
    lcd.print(humidity);
```

```
lcd.print("%");
lcd.setCursor(0, 1);
lcd.print("Temperature: ");
lcd.print(temperature);
lcd.print("*C");
lcd.setCursor(0, 2);
lcd.print("Liquid Temp: ");
lcd.print(ds18b20Temp);
lcd.print("*C");
lcd.setCursor(0, 3);
lcd.print("Thermocouple: ");
lcd.print(thermocoupleTemp);
lcd.print("*C");
}
else if (DS_SWITCH_ETAT && THERMOTESTO_SWITCH_ETAT) // 3*2
{
    lcd.setCursor(0, 0);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");
    lcd.setCursor(0, 1);
    lcd.print("Temperature: ");
    lcd.print(temperature);
    lcd.print("*C");
    lcd.setCursor(0, 2);
    lcd.print("Liquid Temp: ");
    lcd.print(ds18b20Temp);
    lcd.print("*C");
    lcd.setCursor(0, 3);
    lcd.print("Testo: ");
    lcd.print(thermocoupleTestoTemp);
```

```
    lcd.print("*C");
}
else if (MLX_SWITCH_ETAT && THERMO_SWITCH_ETAT) // 4*2
{
    lcd.setCursor(0, 0);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");
    lcd.setCursor(0, 1);
    lcd.print("Temperature: ");
    lcd.print(temperature);
    lcd.print("*C");
    lcd.setCursor(0, 2);
    lcd.print("Object Temp: ");
    lcd.print(mlxTemp);
    lcd.print("*C");
    lcd.setCursor(0, 3);
    lcd.print("Thermocouple: ");
    lcd.print(thermocoupleTemp);
    lcd.print("*C");
}
else if (MLX_SWITCH_ETAT && THERMOTESTO_SWITCH_ETAT) // 5*2
{
    lcd.setCursor(0, 0);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");
    lcd.setCursor(0, 1);
    lcd.print("Temperature: ");
    lcd.print(temperature);
    lcd.print("*C");
```

```
    lcd.setCursor(0, 2);
    lcd.print("Object Temp: ");
    lcd.print(mlxTemp);
    lcd.print("*C");
    lcd.setCursor(0, 3);
    lcd.print("Testo: ");
    lcd.print(thermocoupleTestoTemp);
    lcd.print("*C");
}
else if (THERMO_SWITCH_ETAT && THERMOTESTO_SWITCH_ETAT) // 6*2
{
    lcd.setCursor(0, 0);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");
    lcd.setCursor(0, 1);
    lcd.print("Temperature: ");
    lcd.print(temperature);
    lcd.print("*C");
    lcd.setCursor(0, 2);
    lcd.print("Thermocouple: ");
    lcd.print(thermocoupleTemp);
    lcd.print("*C");
    lcd.setCursor(0, 3);
    lcd.print("Testo: ");
    lcd.print(thermocoupleTestoTemp);
    lcd.print("*C");
}
else if (DS_SWITCH_ETAT) // 1*1
{
    lcd.setCursor(0, 0);
```



```
lcd.print("Humidity: ");
lcd.print(humidity);
lcd.print("%");
lcd.setCursor(0, 1);
lcd.print("Temperature: ");
lcd.print(temperature);
lcd.print("*C");
lcd.setCursor(0, 2);
lcd.print("Power: ");
lcd.print(power);
lcd.print("W");
lcd.setCursor(0, 3);
lcd.print("Liquid Temp: ");
lcd.print(ds18b20Temp);
lcd.print("*C");
}
else if (MLX_SWITCH_ETAT) // 2*1
{
    lcd.setCursor(0, 0);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");
    lcd.setCursor(0, 1);
    lcd.print("Temperature: ");
    lcd.print(temperature);
    lcd.print("*C");
    lcd.setCursor(0, 2);
    lcd.print("Power: ");
    lcd.print(power);
    lcd.print("W");
    lcd.setCursor(0, 3);
```

```
    lcd.print("Object Temp: ");
    lcd.print(mlxTemp);
    lcd.print("*C");
}
else if (THERMO_SWITCH_ETAT) // 3*1
{
    lcd.setCursor(0, 0);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");
    lcd.setCursor(0, 1);
    lcd.print("Temperature: ");
    lcd.print(temperature);
    lcd.print("*C");
    lcd.setCursor(0, 2);
    lcd.print("Power: ");
    lcd.print(power);
    lcd.print("W");
    lcd.setCursor(0, 3);
    lcd.print("Thermocouple: ");
    lcd.print(thermocoupleTemp);
    lcd.print("*C");
}
else if (THERMOTESTO_SWITCH_ETAT) // 4*1
{
    lcd.setCursor(0, 0);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");
    lcd.setCursor(0, 1);
    lcd.print("Temperature: ");
```

```
lcd.print(temperature);  
lcd.print("*C");  
lcd.setCursor(0, 2);  
lcd.print("Power: ");  
lcd.print(power);  
lcd.print("W");  
lcd.setCursor(0, 3);  
lcd.print("Testo: ");  
lcd.print(thermocoupleTestoTemp);  
lcd.print("*C");  
}  
else // 1*0  
{  
  lcd.setCursor(0, 0);  
  lcd.print("Humidity: ");  
  lcd.print(humidity);  
  lcd.print("%");  
  lcd.setCursor(0, 1);  
  lcd.print("Temperature: ");  
  lcd.print(temperature);  
  lcd.print("*C");  
  lcd.setCursor(0, 2);  
  lcd.print("Power: ");  
  lcd.print(power);  
  lcd.print("W");  
}  
}
```

9.7. Code site web

Voici le liens Github pour avoir accès au code du site web :

<https://github.com/SamirAoufia/Stage>