Projet de TP d'électronique
Réalisation d'une station météo

Gabin Blanchet, Quentin Davin, Thomas Giroud, Firmin Launay

ESIREM — Deuxième année de prépa intégrée, semestre 4

Juin 2023

1 Introduction

Ce projet vise à créer une station météo. Les mesures sont effectuées à l'aide d'un Arduino connecté à des capteurs, communiquant avec un serveur central, hébergé sur un Raspberry Pi par exemple.

Afin de réaliser ce projet nous avons utilisé le matériel suivant :

— Raspberry Pi 4
— Arduino Uno Wifi Rev2
— Arduino Sensor Kit
— Grove - Air quality sensor v1.3

Seuls les capteurs de température et de pression du kit sont nécessaires pour effectuer la suite du projet.

Remarque Tous les codes mentionnés dans ce rapport sont disponible sur notre dépôt Git : `https://github.com/Firmin-ESIREM/station-meteo`.

2 Arduino

Cette partie est dédié à l'implémentation de l'Arduino Uno Wifi Rev2. Cette dernière est chargée de récupérer les différentes données issues des capteurs. Une fois l'ensemble de données regroupé, l'Arduino réalise une requête web sur le serveur.

Dans un premier temps, nous verrons l'aspect réseaux, notamment la connexion Wi-Fi ainsi que les requêtes web. Puis, nous verrons comment collecter les données des capteurs.

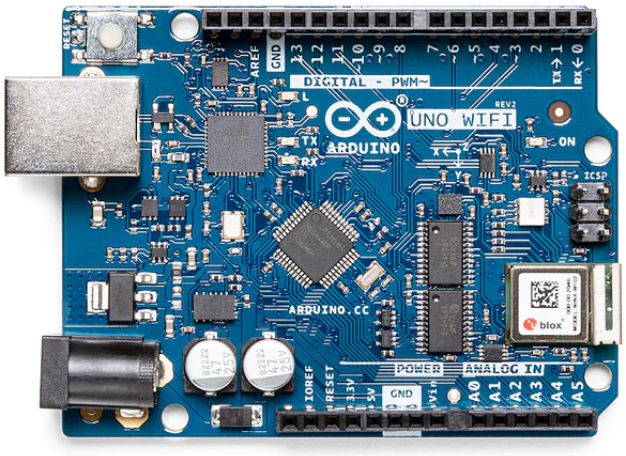


FIGURE 1 – Arduino Uno Wifi Rev2

2.1 Réseau

Notre projet est adapté pour le microcontrôleur *Arduino uno Wifi rev 2* équipé d'un module Wi-Fi u-blox NINA-W102. Ce module est compatible avec la librairie WiFiNINA. Il est important de noter que la mise à jour du firmware est impératif.

Les codes pour la mise à jour du firmware sont CheckFirmWareVersion.ino et FirmWareUpdater.ino.

2.1.1 Connexion au serveur

Pour établir une connexion avec le serveur, il faut dans un premier temps se connecter dans le même réseau Wi-Fi. Il suffit de modifier le mot de passe et le SSID du Wi-Fi dans le fichier arduino_secrets.h. On procède de la même manière pour l'adresse IP du serveur ainsi que le port associé. Ainsi, avec la librairie ArduinoHttpClient, l'Arduino pourra se connecter en tant que client et le Raspberry Pi fera office de serveur : `HttpClient client = HttpClient(wifi, serverAddress, port);`

2.1.2 Requête Web

Pour les requêtes web, nous avons utilisé les librairies ArduinoHttpClient et ArduinoJson. Nous avons utilisé une requête GET, qui permet de récupérer à l'URL /get_refresh_rate/ l'intervalle temporel de transmission des données fixé côté serveur. De plus, nous envoyons par une requête POST nos données sous format JSON à /add_data/.

```
//Get data
client.get("/get_refresh_rate/");
String newtimer = client.responseBody();
int val_timer = newtimer.toInt();
if(val_timer > 0) timer = val_timer;
//
```

(a) Requête get

```
// Create Json
StaticJsonDocument<200> doc;
doc["temperature"]=humidity_tempsensor.get_temp_value();
doc["air_quality_value"]=airsensor.getqualityair_value();
doc["air_quality"]=airsensor.getqualityair();
doc["pressure"]=pressionsensor.getpression_value();
doc["humidity"]=humidity_tempsensor.get_humidity_value();
//
```

(b) Création d'un json

```
//Post data
String contentType = "application/json";
String postData;
serializeJson(doc, postData);
client.post("/add_data/", contentType, postData);
//
```

(c) Requête post

FIGURE 2 – Code pour différentes type de requête

2.2 Capteurs

Nous avons utilisé trois capteurs pour mesurer des grandeurs physiques distinctes. Ces derniers sont connectés au shield par des câbles de type Grove. Pour initialiser les capteurs, nous avons créé une fonction `init()`, puis la fonction `debug()` qui permet d'afficher sur le port 9600 la valeur retournée par le capteur.

2.2.1 Qualité de l'air

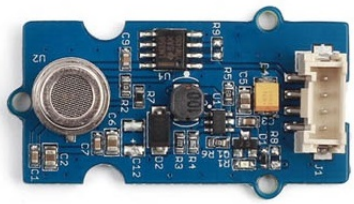


FIGURE 3 – Capteur de qualité de l'air – modèle *MP503*

La figure 3 représente le capteur dont nous disposions. Il s'agit d'un capteur de qualité de l'air (MP503) qui évalue la concentration en gaz dans l'air environnant. Il détecte en particulier la fumée ou l'alcool. La librairie utilisé est `Air_Quality_Sensor`. Nous utilisons par défaut la sortie A_0. La valeur retournée est un indice :

- 0 correspond à forte pollution ;
- 1 correspond à pollution moyenne ;
- 2 correspond à faible pollution ;
- 3 correspond à air frais.

2.2.2 Température et humidité

FIGURE 4 – Capteur de température et d'humidité *DHT10*

Dans la figure 4, on peut voir le capteur DHT10 du fabricant ASAIR®. Ce dernier peut capter des températures allant de $-40\,°C$ à $80\,°C$ avec une erreur de précision de $\pm 0.5\,°C$. L'humidité est un pourcentage entre 0% à 100%, avec une précision de 3 points. La librairie DHT est nécessaire à l'utilisation de ce capteur.

2.2.3 Pression

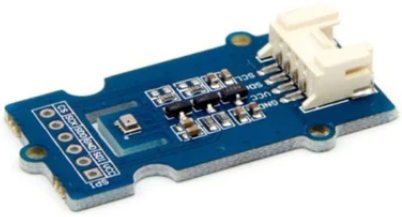


FIGURE 5 – Capteur de pression *BMP280*

Nous utilisons le capteur BMP280 de Bosch Sensortec, représenté en . À l'aide de ce capteur, nous pouvons enregistrer des valeurs minimales de 300 hPa et des valeurs maximales de 1100 hPa. L'incertitude absolue de mesure est de ± 0.12 hPa. La librairie Seeed_BMP280 est nécessaire pour piloter le capteur.

3 Raspberry Pi

Pour collecter, compiler et traiter les données recueillies par l'Arduino, nous utilisons un Raspberry Pi. On peut d'ailleurs utiliser tout autre ordinateur : il s'agit simplement d'un « serveur ».

3.1 Base de données

Nous avons tout d'abord mis en place une base de données dans laquelle stocker les valeurs obtenues depuis l'Arduino. Pour ce faire, à chaque fois que l'Arduino envoie ses mesures, un nouvel enregistrement est créé dans la base de données. Celui-ci comporte la date et l'heure de l'enregistrement et les données numériques obtenues. Nous avons ensuite implémenté des fonctions permettant de récupérer des éléments de la base, soit par type, soit uniquement le dernier enregistrement, soit l'intégralité de la base.

Nous avons choisi d'utiliser une base SQLite. En effet, ce type de base est assez aisé à manipuler en Python : on exécute simplement des requêtes SQL avec la fonction `cursor.execute()`. De plus, cela ne recquiert pas d'installation préalable sur la machine (pas de dæmon hors Python), et reste bien plus fiable qu'un simple fichier JSON ou CSV.

3.2 Interface Web

Nous avons aussi créé une interface Web, fonctionnant à l'aide d'un serveur Flask. Celle-ci permet d'afficher les données météorologiques collectées.

Cette interface est essentiellement codée en HTML, CSS et JS, en s'aidant du moteur de rendu Jinja, intégré à Flask.

Elle comporte deux principales parties :

— l'écran d'« accueil », qui affiche les dernières données à jour ;

— les « archives », qui, pour chaque donnée, affichent un graphique de l'évolution de cette donnée au cours du temps, réalisé à l'aide de la bibliothèque JavaScript JSCharting.

FIGURE 6 – Écran d'accueil de la station météo

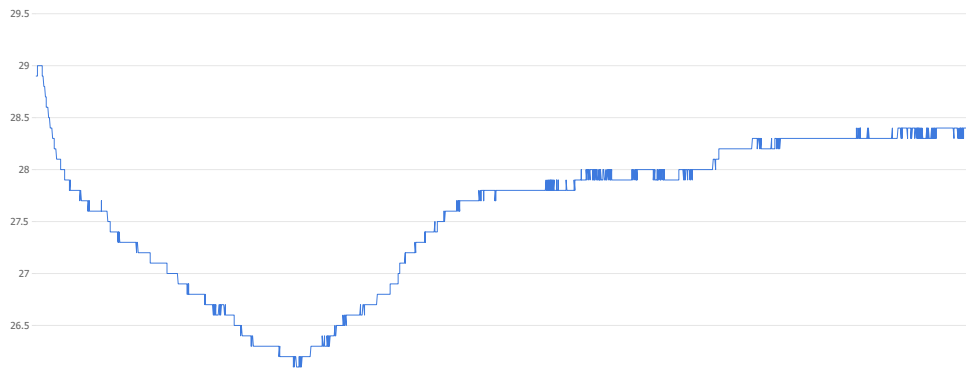


FIGURE 7 – Page archives de la station météo

Cette interface web est également modulée en fonction des paramètres définis par l'utilisateur. En effet, celle-ci dispose d'un mode sombre et est traduite en plusieurs langues (français, anglais, espagnol, italien) pour s'adapter automatiquement au plus d'utilisateurs possibles.

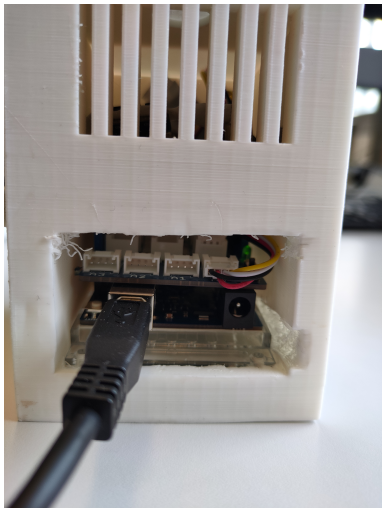
Une fois lancé sur le Raspberry Pi, le programme écoute au port 7657. Pour un usage en production, il est recommandé de remplacer le serveur de développement Flask que nous utilisons ici par un serveur de production, comme Waitress par exemple. Si on utilise le port 80, et que l'on redirige ce port (dans le cas où le serveur est protégé par un NAT), on peut accéder à l'interface depuis le monde entier. Par ailleurs, à l'aide d'un proxy inverse comme Nginx ou d'un serveur adapté comme Gunicorn, on peut sécuriser cette connexion par HTTPS et écouter le port 443, en fournissant un certificat et une clef privée.

4 Modélisation 3D

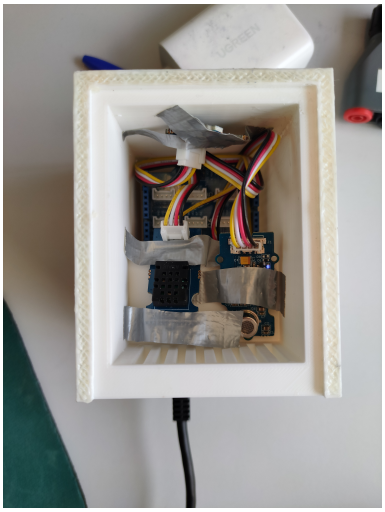
Pour apporter une finition au projet, nous avons modélisé une petite boîte accueillant l'Arduino et ses capteurs. Pour cela, nous avons utilisé le site Onshape. Des images de la première version de la boîte sont en figure 8.

(a) Boîte vue de devant

(b) Boîte vue de derrière



(c) Vue de l'Arduino dans la boîte



(d) Boîte vue de dessus

FIGURE 8 – Boîte imprimée en 3D logeant l'Arduino et ses capteurs