Julien CAGNIART – Chayma ABBAS

Rapport projet Internet des Objets

Rapport de projet de l'UE Internet des Objets du M1 Informatique de l'Université de Pau et des Pays de l'Adour.

Table des matières

Introduction

Pau et son agglomération sont desservies par le service de bus Idelis. L'ensemble des positions des bus est collecté, et est accessible via diverses APIs.

Cela permet notamment d'afficher aux arrêts de bus les plus fréquentés le délai estimé avant le passage du prochain bus.

Image 1 - Affichage fixe des prochains passages de bus

L'application mobile Idelis offre aussi une fonction similaire pour connaître les prochains passages aux arrêts de bus les plus proches de l'utilisateur. Malheureusement, cette fonctionnalité est cachée derrière de nombreux menus, et pose un problème lorsqu'on est déjà en retard.

En binôme, nous avons donc voulu reproduire cet affichage pour son logement à l'aide d'un ESP32.

De plus, pour promouvoir l'utilisation des transports en commun, nous avons en parallèle calculé le temps de trajet en voiture afin d'offrir une comparaison.

Présentation du projet

Image 2 - Rendu final de l'application

ESP32

Nous possédons une carte de développement ESP32 connectée à un écran TFT de 2.8 pouces. Nous l'avons donc réutilisé pour produire ce projet.

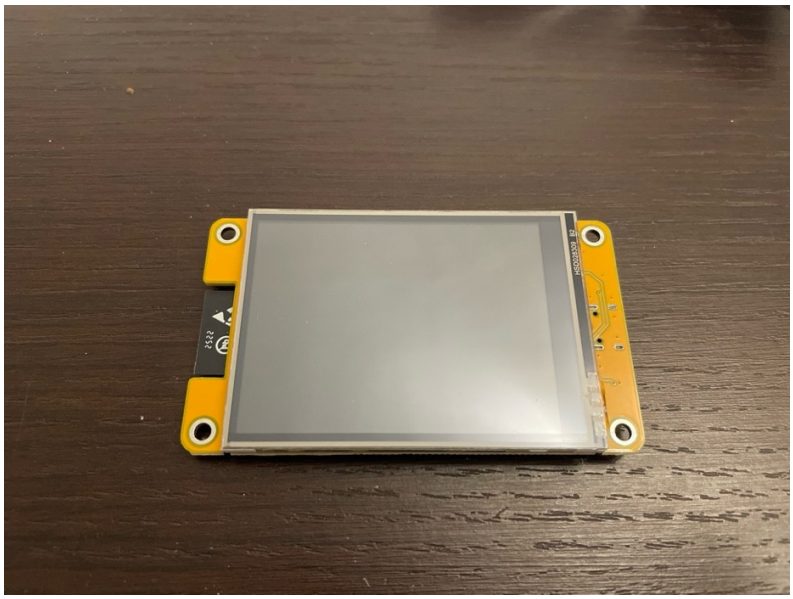

Image 3 - L'ESP32 éteint

La partie embarquée du projet a été entièrement développé avec Arduino IDE. Dans un premier temps, la carte se connecte au Wi-Fi en utilisant la bibliothèque Wi-Fi d'Arduino. Ensuite, elle récupère par MQTT (architecture détaillée plus bas) les prochains horaires en utilisant la bibliothèque PubSubClient par Nick O'Leary. Enfin, elle affiche à l'écran ces horaires en utilisant LVGL, une bibliothèque C pour construire des interfaces sur des supports variés.

Architecture du projet

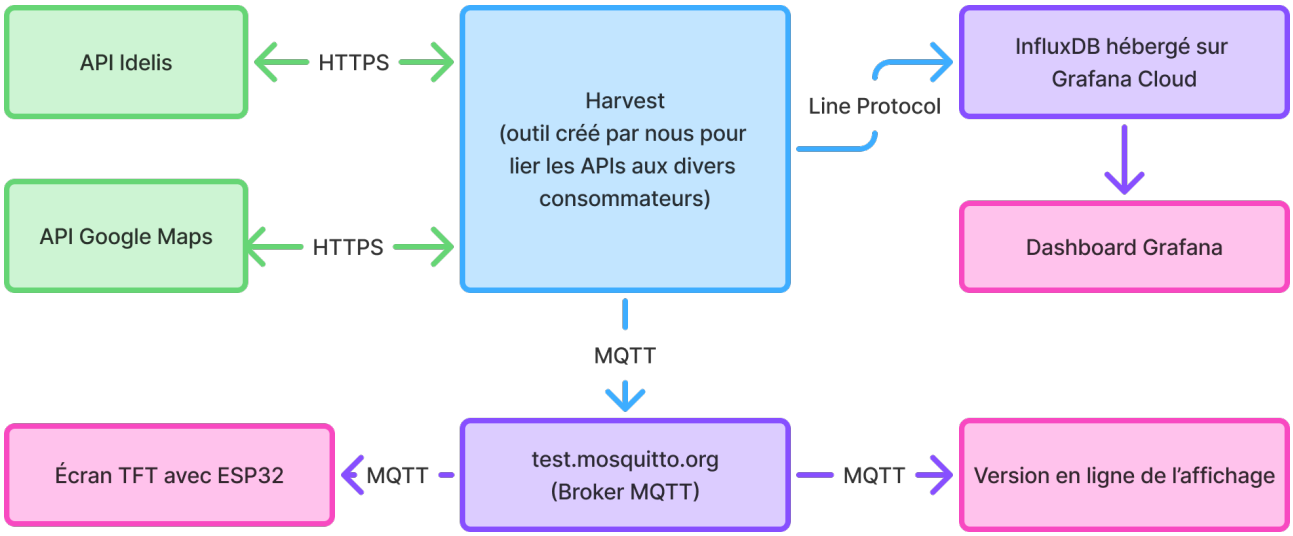

Image 4 - Architecture du projet

L'application lie plusieurs éléments pour fonctionner :

- Harvest est l'outil central qui se charge d'orchestrer l'ensemble du flux d'informations. Il se compose d'un script Javascript exécuté sur un VPS par NodeJS. Son travail consiste à :
  - Récolter les horaires des bus depuis l'API Idelis
  - Récolter le temps de trajet suivant le trafic en utilisant l'API Routes de Google Maps Platform[1]

[1] Le temps de trajet est récolté à une fréquence plus faible qu'Idelis (20 minutes) dû à des coûts exorbitants

    - Convertir les données des différentes APIs en un format commun (par exemple en seconde)
    - Transmettre ces données à Influx DB en utilisant le Line Protocol
    - Publier des messages par MQTT sur les topics « UPPA_M1_IOT/harvest/travelTime/car », « UPPA_M1_IOT/harvest/travelTime/bus » et « UPPA_M1_IOT/harvest/nextBusDepartures » afin que les différents consommateurs puissent les obtenir
- Influx DB collecte et stocke les différentes métriques avec leur horodatage.
- L'écran TFT avec ESP32 se charge de récupérer par MQTT les informations et les afficher comme expliqué plus haut
- « test.mosquitto.org » est un broker MQTT gratuit qui permet de diriger les informations vers les bons consommateurs
- Un dashboard Grafana permet d'afficher des graphiques des métriques collectées par Influx DB
- Enfin, une version en ligne de l'affichage de l'ESP32 existe.

APIs

Idelis

Idelis offre une API pour permettre d'obtenir les prochains bus pour un arrêt de bus donnée. Le service Harvest se charge de faire des requêtes à cette API toutes les 30s, et de le publier par MQTT et à Influx DB.

Google Maps Routes API

Google Maps offre une API de calcul de temps trajet qui prend en compte le trafic routier. Cela permet de comparer le temps de trajet en bus calculé par le service Harvest au temps de trajet en voiture.

Dashboard Grafana

Influx DB

Grafana offre une version hébergée d'Influx DB pour stocker des métriques. Nous l'utilisons pour sauvegarder les temps de trajet au fil du temps.

Dashboard

Nous utilisons ensuite Grafana pour afficher ces données.

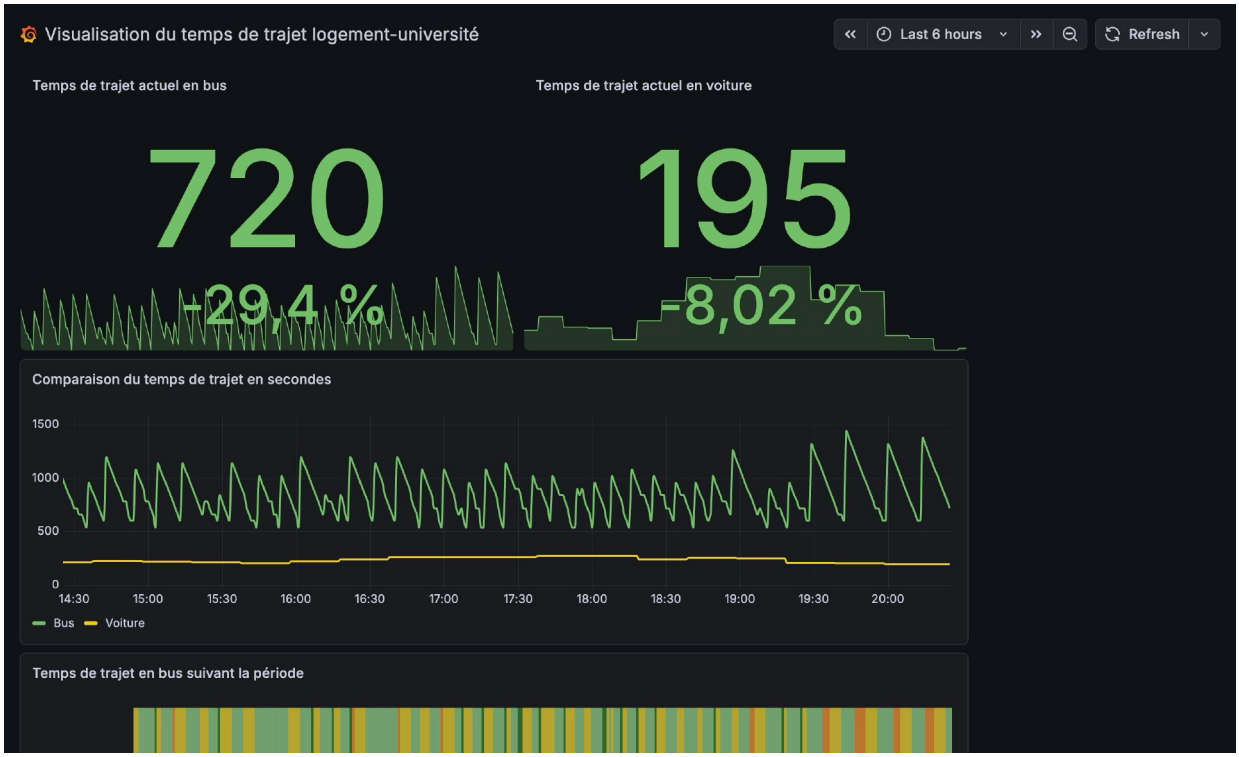

Image 5 - Dashboard Grafana des métriques de temps de trajet

Il est possible d'accéder au tableau de bord en suivant ce lien :
https://m1uppa.grafana.net/public-dashboards/603909983a9749928f01e88cf376a2e1

Version en ligne

Enfin, pour ceux qui ne peuvent avoir accès à l'écran, nous avons construit une version en ligne de l'affichage[2]. Elle est identique dans son fonctionnement. Elle se connecte elle aussi par MQTT à « test.mosquitto.org » et affiche les mêmes informations que l'écran en même temps.

Image 6 - Version en ligne de l'affichage

Il est possible d'y accéder en suivant ce lien :
https://m1-iot.julienc.me/

Code source

L'ensemble du code source est accessible sur GitHub : https://github.com/julien040/M1_IoT_projet. Il est séparé en trois dossiers :

- « ESP32/main » contient le code Arduino (C++) pour se connecter au Wi-Fi, pour se connecter au broker MQTT et pour construire l'affichage de l'écran.
- « harvest » contient le service Harvest expliqué plus haut
- « website » contient la version en ligne de l'affichage des horaires.

[2] Avec le framework Astro en Javascript

Table des illustrations