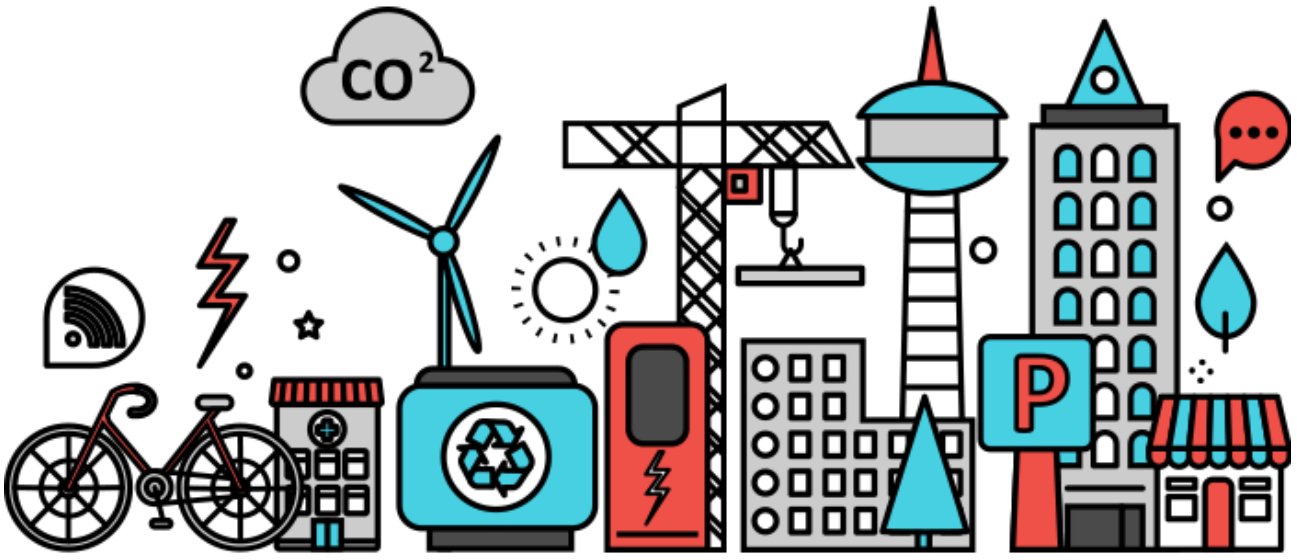
Rapport final

Projet capteur



Sommaire

Contexte et définition du projet

Dans tous les environnements, nous sommes immergés dans un monde de sons variés. La mesure de la qualité sonore devient cruciale, encore plus dans le monde moderne dans lequel on évolue, afin de concevoir des espaces qui évoquent la sérénité ou encore de qualifier et rendre compte des environnements excessivement bruyants.

Notre application finale consiste en un détecteur de niveau sonore. Ce niveau est directement visualisable sur la carte électronique ainsi que sur un afficheur distant. Cette conception vise à intégrer notre capteur dans le domaine de l'Internet des Objets (IoT). L'idée derrière est de pouvoir comparer le niveau sonore de l'endroit dans lequel le capteur est présent à des environnements dont on connaît déjà le niveau sonore moyen. Ce présent rapport a pour objectif de vous présenter le choix de capteur sonore non-Arduino effectué, la carte électronique conçue, l'étalonnage et la comparaison entre nos microphones, l'application développée ainsi que les problèmes que nous avons pu rencontrer.

Choix du capteur non-Arduino

Il existe plusieurs technologies de microphones sur le marché, en particulier les microphones MEMS (microsystèmes électromécaniques) et les microphones ECM (microphones à condensateur électret). Une première caractéristique des MEMS est leur taille de l'ordre du millimètre. Ils sont relativement difficiles à intégrer dans un prototype et encore plus à braser sur une carte. A contrario, les ECM sont bien plus gros et permettent ainsi un prototypage rapide. De plus, ils sont utilisés depuis des décennies, et l'on trouve dans la littérature et sur internet bon nombre de montages les utilisant. Enfin, les microphones MEMS sont plus récents dans leur technologie, et intègrent directement le conditionnement du microphone grâce à leur ASIC intégré. Concernant notre projet, nous trouvons cela dommage d'utiliser un capteur fournissant une sortie déjà conditionnée et numérique; cela enlève en effet une grosse partie de l'objectif initial du projet en lien avec le cours EE470. La figure 1 montre une comparaison entre deux capteurs Arduino avec leur chaîne de conditionnement. En haut un microphone MEMS (Adafruit I2S MEMS), et en bas un microphone ECM (Adafruit MAX4466).

Fig. 1 : Comparaison MEMS/ECM

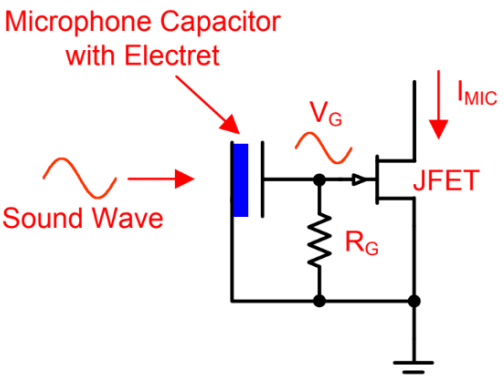


Fig. 2 : Fonctionnement d'un ECM

Nous avons donc opté pour les microphones ECM, nous permettant justement de réaliser la partie conditionnement du signal analogique, et permettant un prototypage plus aisé.

Un microphone électret est basé sur une capacité électrique de valeur variable. Quand une onde sonore arrive, la pression acoustique qu'elle véhicule fait bouger une des deux plaques du condensateur formé, et donc la distance entre ces plaques. Ainsi la capacité, et donc la tension aux bornes du microphone, varie en fonction de l'intensité acoustique reçue. La plupart des microphones électret intègrent en interne un transistor JFET. Ce dernier permet que le signal en tension produit par le son puisse moduler directement la tension de la grille, ce qui engendre une variation du courant circulant entre les deux broches (drain et source) du transistor. Ceci est illustré sur la figure 2.

Conditionnement du capteur et réalisation du PCB

Nous avons choisi d'intégrer notre microphone en s'inspirant d'un montage proposé par Adafruit, qui a fait ses preuves dans la communauté Arduino et du DIY. Il est basé sur un amplificateur opérationnel MAX4466, qui est optimisé pour être utilisé en tant que préamplificateur de microphone. La schématique du conditionnement du signal analogique fourni par le microphone est présentée figure 3.

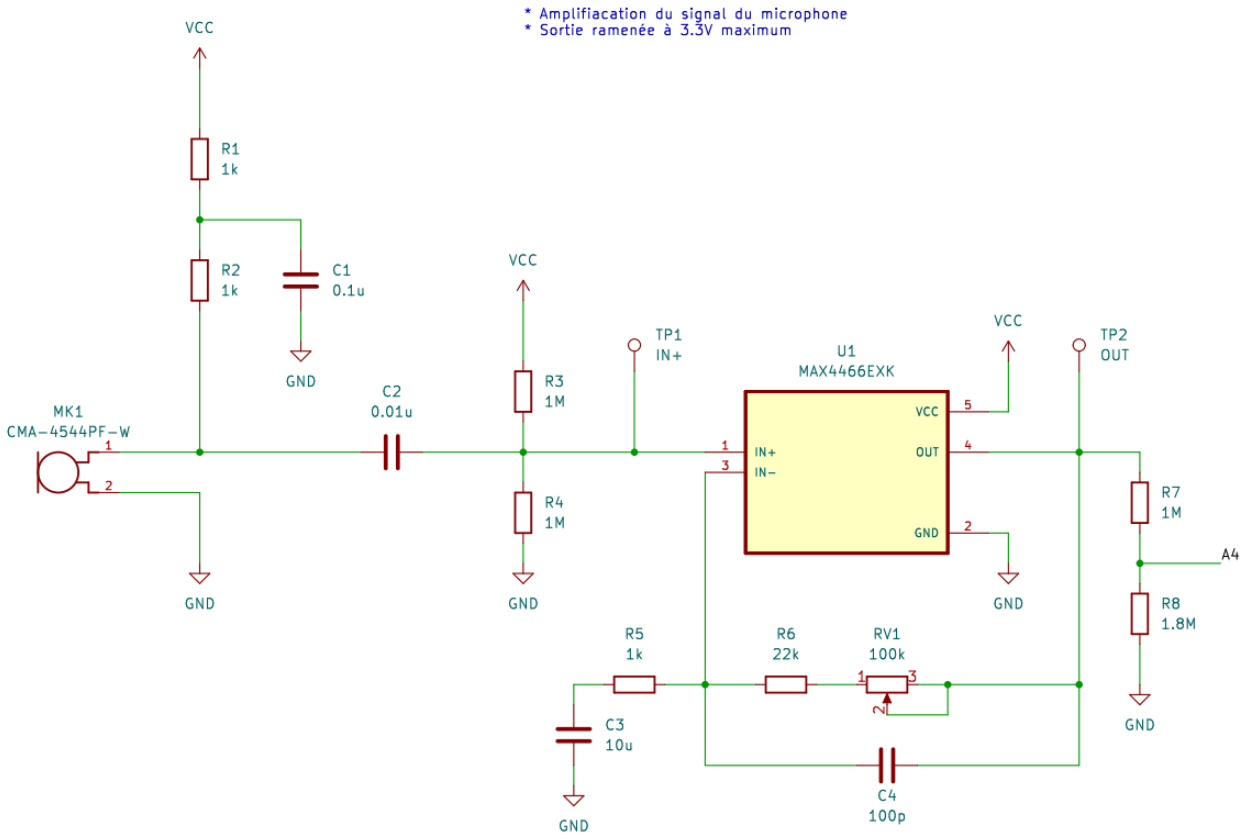


Fig. 3 : Schématique du conditionnement du signal analogique

Le microphone est alimenté par le VCC qui est découplé. Ensuite, le courant modulé par les ondes sonores passe à travers une capacité (C2) qui lui retire sa composante continue, pour ne garder que la modulation pure. L'AOP fonctionne en montage transimpédance (rétroaction RC). Cela permet de convertir la modulation de courant en une modulation en tension, le tout avec un gain ajustable. L'ESP32 fournissant une alimentation en 5 V, on vient alimenter notre AOP en asymétrique 0/5 V, avec un offset de 2.5 V (amené par R3R4) pour recentrer la plage de travail. Un dernier pont diviseur (R7R8) ramène la sortie à une valeur maximale de 3.3 V, tension maximale des entrées de l'ESP32.

Une fois la schématique dessinée, nous avons réalisé un prototype sur breadboard. Ce dernier ayant fonctionné, nous avons entrepris la réalisation d'un PCB. Le but était de réaliser un shield pour l'ESP32 (compatible avec Arduino). Nous avons rajouté au PCB des LEDs (vert, orange, rouge) pour servir d'afficheur du niveau sonore par seuils, ainsi qu'un connecteur pour le microphone déjà intégré (capteur Grove) en plus du notre (capteur "Custom"), afin de comparer les deux. Les deux microphones sont alors relativement proches l'un de l'autre, pour permettre un étalonnage plus juste. Le PCB a été réalisé avec KiCad, fabriqué par JLCPCB, et brasé à l'école.

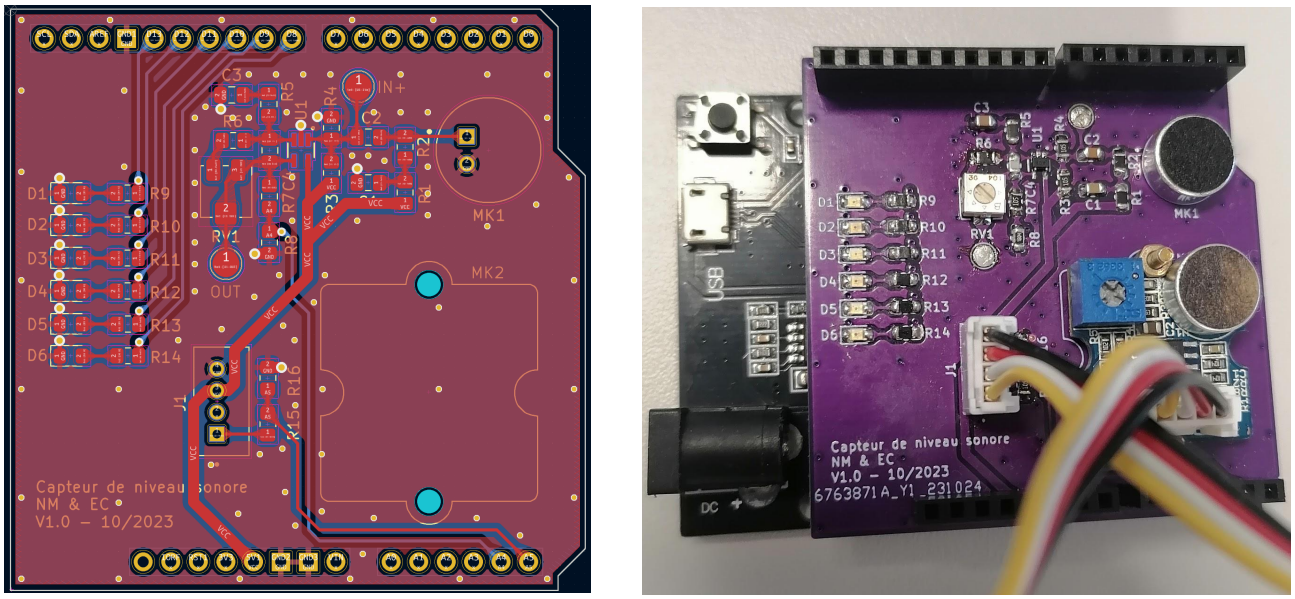


Fig. 4 : Réalisation du PCB avec KiCad (gauche), carte assemblée (droite)

Etalonnage et comparaison des deux capteurs

Il faut prendre garde aux unités ! Le son se mesure de façon logarithmique (dB), en utilisant comme référence le seuil d'audibilité de l'oreille humaine (20 µPa de pression acoustique). L'unité ainsi formée pour la mesure du son est le dB SPL (décibel Sound Presure Level). Vol [dB SPL] = 20 log(p [Pa] / 20µ [Pa]). En revanche l'oreille humaine est moins sensible aux basses fréquences qu'aux hautes fréquences (à volume d'émission identique, un son aigu sera perçu par l'oreille plus fort qu'un son grave). C'est pourquoi lorsque l'on s'intéresse à la mesure du bruit dans divers environnements on applique un filtre sur les dB SPL, pour atténuer les basses fréquences et amplifier légèrement les hautes fréquences. On obtient alors une nouvelle unité : les dB(A).

Commençons par décrire notre banc de test. Nous disposons d'un sonomètre issu du magasin de l'école, mesurant le volume en dB(A). Nous le considérons comme capteur de référence pour réaliser nos étalonnages. Nous utilisons une source sonore émettant un son constant : un PC avec une enceinte jouant une fréquence fixe de 500 Hz. A environ un mètre de l'enceinte, nous plaçons notre carte (sur laquelle sont situés les deux microphones côte-à-côte), à côté du sonomètre. Nous faisons varier le volume de l'enceinte en relevant à chaque fois le niveau sonore (en dB(A), mesuré par le sonomètre) et la tension en sortie de notre capteur. Nous obtenons alors une courbe avec en abscisses la tension relevée par notre capteur (après passage dans le convertisseur analogique/numérique) et en ordonnées le niveau sonore réel en dB(A). Nous linéarisons cette courbe pour obtenir une équation (donné par un tableur), nous permettant de convertir la tension lue par l'Arduino en un volume en dB(A), sous forme d'une fonction affine.

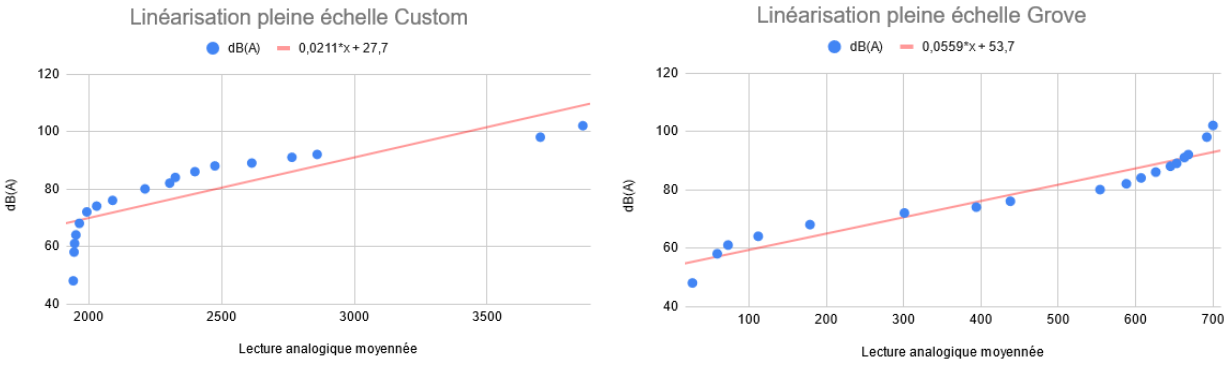


Fig. 5 : Courbes d'étalonnage des capteurs

On observe que les points du microphone Custom peuvent être scindés en deux parties linéaires (inférieur et supérieur à 70 dB(A)). Nous séparons alors les données en deux pour obtenir deux linéarisations (voir figure suivante). Pour le capteur Grove, les points sont tous globalement assez proches de la droite de linéarisation, nous avons choisi de garder l'équation obtenue.

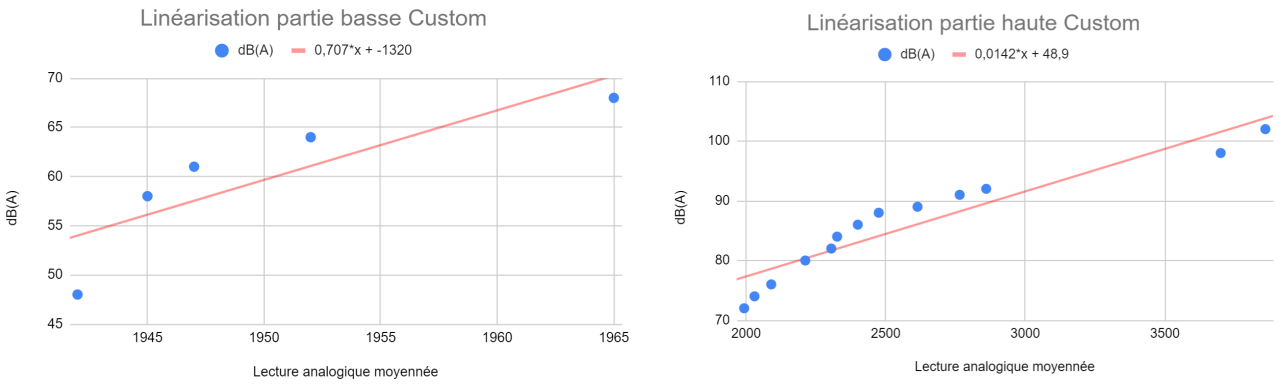


Fig. 6 : Double linéarisation pour le capteur Custom

Une fois nos équations saisies dans notre code, qui convertit alors la tension lue en une valeur en dB(A), nous pouvons tester et comparer nos deux microphones. Nous avons repris le même banc de test pour refaire les caractérisations.

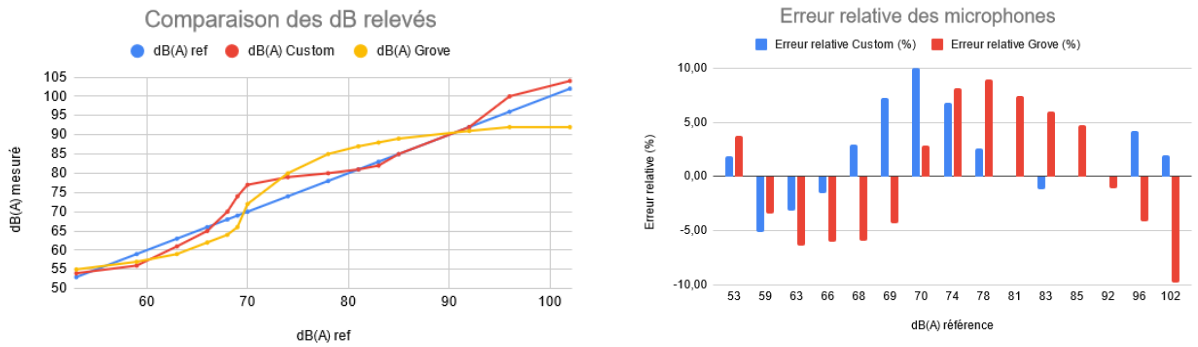


Fig. 7 : Test et comparaison des microphones

On remarque dans un premier temps que les deux courbes (Custome et Grove) sont relativement proches de la référence, les résultats obtenus ne sont donc pas aléatoires ou incohérents. Nous pouvons cependant constater que le microphone Custom suit plutôt bien la référence, avec une erreur relativement importante autour de 70 dB(A). Le microphone Grove quant à lui est globalement moins précis. Nous pouvons par ailleurs remarquer qu'il sature autour de 92 dB(A). Ces observations s'expliquent notamment par nos choix de finesse pour la linéarisation. Pour rappel, nous avons en effet choisi de calculer deux linéarisations pour le capteur Custom mais de n'en faire qu'une pour le capteur Grove. Nous constatons cela sur le tracé de l'erreur relative, le Custom présente un pic plus gros à 10% d'erreur, mais le Grove dépasse de plus de 5% sur une plage plus large.

Application développée

Pour atteindre l'objectif que nous nous sommes fixé lors de la définition du projet, nous avons découpé notre application en trois versions, allant de la plus simple à la plus complexe à mettre en œuvre.

Version 1 : Cette première itération, présentée lors de l'évaluation intermédiaire, repose sur la plateforme Arduino. Elle récupère le signal du capteur Grove, effectue la conversion en dB(A), puis affiche la valeur en dB(A) via la liaison série.

Version 2 : Cette deuxième version intègre à la fois le capteur Grove et un capteur non-Arduino. L'application est capable de calculer et d'afficher la valeur en dB(A) pour les capteurs Grove et non-Arduino. De plus, cette version offre la possibilité de visualiser le niveau sonore en temps réel à l'aide d'un vu-mètre composé de LEDs. Cela donne ainsi une information directe du niveau sonore sans avoir besoin d'utiliser un afficheur déporté.

Version 3 : La version avancée, répondant pleinement à notre cahier des charges, inscrit notre capteur dans le domaine de l'Internet des Objets (IoT) en incorporant la fonctionnalité WiFi à notre application. Un logiciel web est mis en place pour afficher la valeur du niveau sonore et déterminer le type d'environnement en fonction de ce niveau (par exemple : bureau calme, cantine, rue passante, etc).

Afin de concrétiser la mise en œuvre matérielle de la version 3, nous avons dû étendre les fonctionnalités de notre carte en y intégrant un module WiFi. Cependant, la recherche de modules facilement intégrables avec Arduino dans un délai raisonnable et à moindre coût s'est avérée être un défi. Après des investigations approfondies, il nous a paru plus judicieux de remplacer notre microcontrôleur Atmega328P (celui embarqué par l'Arduino) par un ESP32 d'Expressif, qui intègre un coprocesseur dédié à la gestion du WiFi. Afin de maintenir la compatibilité avec les shields Arduino, nous avons opté pour une carte de développement basée sur ESP32 (Wemos D1 R32), respectant le brochage des cartes Arduino Uno.

Il est à noter que l'ESP32 est un SoC 32 bits qui dépasse les performances des microcontrôleurs Atmega, avec une fréquence environ 10 fois supérieure et une capacité étendue de stockage, utile pour échantillonner notre signal plus longtemps. Son ADC 12 bits améliore la précision de la capture des signaux analogiques, renforçant ainsi la sensibilité de notre application de détection de niveau sonore.

Pour des raisons de simplicité et de démonstration, nous avons embarqué la version 2 et 3 dans un même programme, afin de pouvoir passer d'une version à l'autre par une commande envoyée depuis via la liaison série.

Après l'étalonnage des deux capteurs par rapport au sonomètre de référence, nous pouvons valider le fonctionnement de la version 2 de notre application avec l'affichage du niveau sonore en dB(A) par nos deux microphones.

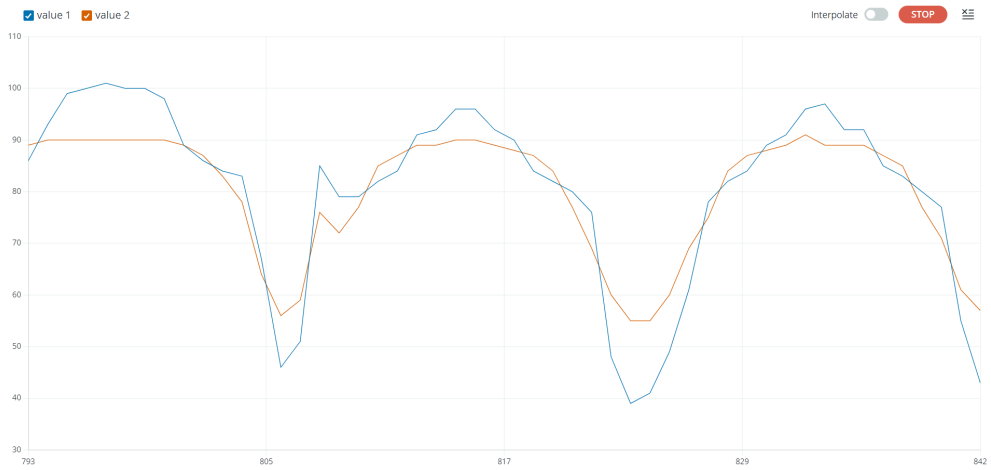


Fig. 8 : Niveaux sonores de nos capteurs

La courbe bleue est la valeur en dB(A) calculée avec notre capteur Custom et la rouge celle de notre capteur Grove. Nous pouvons voir que lors de silences (minimums des signaux), nous avons une forte différence car nous ne sommes pas dans la plage idéale de fonctionnement des capteurs. Cependant, pour des niveaux sonores plus conséquents (> 60 dB(A)), les mesures des capteurs présentent peu de différence entre elles, et permettent une analyse pertinente.

Pour la version 3, nous utilisons une base de donnée InfluxDB distante (base orientée séries temporelles hautes performances) afin de sauvegarder à la fois la valeur de l'ADC et la valeur en dB(A) calculée par nos deux microphones. Cette base est utilisée par Grafana, un logiciel permettant la visualisation de données, afin d'afficher les niveaux sonores en temps réel sous la forme de graphiques, et d'associer un environnement sonore avec le volume mesuré.

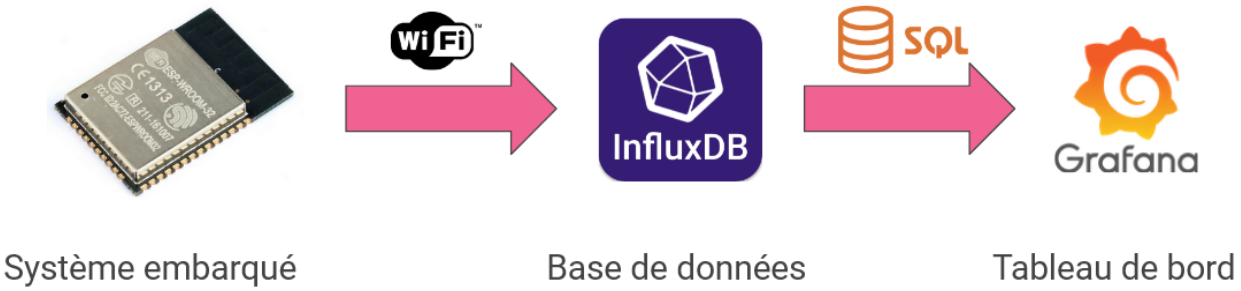


Fig. 9 : Diagramme de fonctionnement de la version 3

Fig. 10 : Visualisation à distance des niveaux et environnements sonores

Sur ce panneau d'affichage, nous avons choisi de représenter sur la première ligne la variation du niveau sonore en dB(A) de notre capteur Custom, ainsi que l'environnement "type" dans lequel il se trouve (par rapport au niveau sonore). La deuxième ligne est identique, si ce n'est qu'elle affiche les mesures du capteur Grove.

Cette version offre un bel exemple d'application IoT concrète, de la mesure au traitement de données à distance : du bas niveau (électronique...) à du haut niveau (web, base de données...). On pourrait imaginer que ce capteur soit placé dans une cantine scolaire, une boîte de nuit ou bien un environnement où il y a beaucoup de trafic routier, et enverrait à distance et de manière continue, une information sur l'environnement sonore. Ces données pourraient ensuite être traitées pour comprendre et alerter sur la pénibilité, voire dangerosité, de certains environnements.

Problèmes rencontrés et améliorations futures

Durant ce projet, nous avons pu rencontrer différentes difficultés, aussi bien sur la partie matérielle que logicielle. Premièrement, les LEDs utilisées pour le vu-mètre perturbaient la ligne analogique de notre microphone Custom lors de leur activation. Ceci était dû à la forte variation de courant lors du pilotage des LEDs, ce qui provoquait de la diaphonie malgré la mise en place de vias stitching pour réduire les effets des perturbations entre la partie numérique et analogique. Nous avons pu résoudre partiellement ce problème en augmentant la résistance en série des LEDs, réduisant la variation de courant lors de l'activation de ces dernières.

Un autre problème majeur que nous avons rencontré a été la perturbation de la ligne analogique de notre microphone Custom causée par l'activation du coprocesseur WiFi de l'ESP32. En effet, le rayonnement du SoC provoquait un couplage avec les lignes analogiques de notre shield placé juste au-dessus de l'ESP32. L'ADC n'arrivait

alors plus à lire une valeur cohérente du niveau sonore. Nous avons réglé ce problème en désactivant le coprocesseur WiFi avant chaque acquisition de données. Cependant, cela a pour effet de ralentir la fréquence maximale à laquelle notre programme envoie les mesures à la base de donnée distante, car il est nécessaire de réactiver le WiFi après chaque mesure. Toutefois, les données sont actuellement transmises toutes les 10 secondes environ, ce qui est suffisant pour notre application.

Nous pouvons enfin évoquer quelques perspectives d'améliorations sur ce projet. L'utilisation d'une carte de développement avec un shield compatible Arduino n'est pas idéal si l'on a besoin d'intégrer réellement ce capteur de manière autonome. Une perspective intéressante serait de créer une carte électronique autonome, qui intègre à la fois le capteur sonore, le vu-mètre, l'ESP32 et son interface de programmation USB. On pourrait également aller encore plus loin en intégrant des objectifs basse consommation avec une alimentation sur batterie, ou par récupération d'énergie afin d'intégrer nos capteur dans une solution autonome et basse consommation IoT.

Conclusion

Durant ce projet, nous avons pu appréhender les différentes étapes de la conception d'un système embarquant un capteur. Nous avons commencé par définir l'application finale, pour donner un réel sens au projet, avant de spécifier son cahier des charges. Nous nous sommes ensuite penchés sur le choix du capteur, ce qui nous a permis de faire une veille technologique sur le sujet. Une fois le capteur choisi, nous avons étudié la littérature sur le sujet, afin de recenser l'état de l'art et de ne pas repartir de zéro. Nous avons ainsi pu réaliser le conditionnement du signal analogique sortant de notre capteur. Par ailleurs, ce projet nous aura permis d'apprendre à réaliser une carte électronique, de la conception sur ordinateur jusqu'à la brasure des composants. Ensuite, nous avons pu nous intéresser à l'étalonnage d'un microphone, et à la linéarisation de sa réponse. Enfin, nous avons pu faire de la programmation sur microcontrôleur, avec envoi d'informations sans-fil (en WiFi) pour de l'affichage personnalisé accessible sur internet, afin d'inscrire ce projet dans ce qui constitue un des grands enjeux de demain : l'IoT.

Ce projet nous aura donc permis d'apprendre des choses et de les mettre en pratique sur un large secteur de compétences, de l'électronique analogique jusqu'à de l'IoT, en passant par la programmation embarquée sur microcontrôleur. Nous sommes très satisfaits de ce projet et du résultat obtenu, et nous avons également identifié certaines pistes d'améliorations possibles si nous souhaitons reprendre ce projet plus tard.