MACÉ Quentin

NOUGUIER Thibaut

RAMEL Régis

Projet RICM4 – Tachymètre

Développement d'une application Cohorte de mesure de trafic routier

RICM4 – Multimédia 11/01/2016 – 06/04/2016

Table des matières

Introduction

Ce projet a été réalisé dans le cadre de notre 2eme année d'étude à Polytech' Grenoble en Réseaux informatiques et communication multimédia. Le travail a été effectué entre le 11 janvier et le 06 avril 2016. Les liens vers la fiche de suivi, le SRS et le dépôt github du projet sont disponibles en annexes.

Contexte du projet

Sujet du projet

Le but du projet est d'utiliser un détecteur de proximité pour développer une application web qui affiche l'historique de tous les passages devant le capteur avec date, heure et vitesse.

Le capteur est connecté à un Arduino qui récupère les mesures de vitesse et de distance. L'Arduino communique ensuite ces valeurs au Raspberry Pi qui sert au traitement des données et à l'envoi des données au serveur. Le serveur gère le stockage de l'historique des valeurs et l'envoi des données lors des demandes par le client web.

Le client est une interface web qui affiche l'historique des passages de manière lisible et interactive (plage de dates, graphiques).

Cohorte

Ce projet doit être développé en Cohorte, une plateforme de développement crée par l'entreprise Isandlatech qui permet de séparer l'application en différents composants. Ces composants sont indépendants. Ils peuvent être codés dans des langages de programmation différents, c'est Cohorte qui s'occupe de la communication entre les différents composants. L'application est une interaction entre les composants qui fournissent et consomment des services.

Acquisition des données

Capteur

Le capteur utilisé dans ce projet est un HB100. Il s'agit d'un émetteur récepteur qui mesure la différence de fréquence entre l'émission et la réception pour calculer la vitesse d'un mobile (voir effet Doppler).

Ce capteur renvoie deux types de donnée :

- Un signal dont la tension correspond à la distance avec le mobile en mouvement. Cette mesure va servir à déterminer le sens de circulation du mobile. En effet si la tension est décroissante, le véhicule s'éloigne, et s'il la tension est croissante, le véhicule se rapproche.

- Un signal carré dont la fréquence correspond à une vitesse. La vitesse du mobile est déterminée grâce à un calcul basé sur l'effet Doppler (voir « Effet Doppler).

Les mesures du capteur sont obtenues en branchant le HB100 à un Arduino, un microcontrôleur programmable qui récupère les données du capteur.

Effet Doppler

Il s'agit d'un phénomène physique qui consiste en un décalage de fréquence des ondes entre l'émetteur et le récepteur lorsque la distance entre ces deux points varie dans le temps. Ce phénomène permet de calculer la vitesse de déplacement d'un mobile grâce à une relation mathématique entre la fréquence mesurée, l'angle entre l'axe de mesure du capteur et le déplacement du mobile.

Pour obtenir une vitesse V en m/s avec une fréquence f en Hz :

$$V = \left| \frac{f}{2 * f_c * \cos(\alpha / c)} \right|$$

Avec f_c la fréquence d'émission du capteur (notre capteur : 10,525 GHz)

α l'angle en radians entre l'axe du capteur et l'axe de la route

c la vitesse de la lumière (300 000 000 m/s)

Traitement des données

Arduino

L'Arduino est une carte possédant un microcontrôleur programmable qui a la capacité de générer et d'analyser des signaux électriques. Il possède des entrées numériques qui vont nous permettre d'acquérir les mesures du capteur HB100.

Traitement :

A chaque mesure, le programme vérifie que la fréquence obtenue est supérieure à 60Hz, ce qui permet de filtrer le bruit fréquentiel. En effet, une fréquence de 60Hz correspond à une vitesse d'environ 3 km/h, ce qui permet de ne garder que les mouvements qui nous intéressent.

Si la mesure est supérieure à 60Hz, la fréquence est transformée en vitesse. Au bout de 20 mesures consécutives de fréquence au-dessus de 60Hz, une moyenne est faite sur les 20 échantillons, de façon à ne pas transmettre des valeurs erronées dues à un bruit important. Puis l'Arduino envoie au Raspberry un doublon (tension, vitesse) contenant les moyennes de tension et de vitesse via une liaison série.

Si la mesure est inférieure à 60Hz, le calcul de la vitesse est ignoré. Au bout de 20 mesures consécutives de fréquence en dessous de 60Hz, l'Arduino envoie au Raspberry un doublon (tension, vitesse) contenant des valeurs nulles.

Malheureusement, le signal envoyé par le capteur n'est pas suffisamment fort pour être traitée par l'Arduino au-dessus d'une certaine fréquence. C'est pour cela que nous ne pouvons pas obtenir des valeurs de vitesses supérieures à 25 km/h sans l'utilisation d'un amplificateur de fréquence dont le schéma de câblage électrique est fourni en annexe.

Raspberry Pi

Le Raspberry Pi est un ordinateur à taille réduite compatible avec de nombreux périphériques (claviers, souris, …) qui peut héberger un programme. Pour notre projet, le Raspberry héberge un isolat Cohorte contenant plusieurs modules en python qui vont servir à la génération des données élaborées à partir des données brutes du de tension et de vitesse.

Le traitement consiste à attendre une valeur non nulle. Dès qu'une valeur supérieure à 0 arrive au Raspberry, c'est qu'un passage est détecté par le capteur. Le Raspberry stocke alors toutes les valeurs non nulles jusqu'au doublon (0.0, 0.0) suivant. Tous les échantillons ainsi stockés forment ainsi l'historique d'un passage. La vitesse maximum du passage est tout simplement la valeur de vitesse la plus élevée, la vitesse moyenne est calculée avec tous les échantillons, et la comparaison entre la tension initiale et la tension finale du passage permet de déterminer les sens de circulation. En effet, si la tension au début du passage est plus élevée qu'à la fin, c'est que le mobile s'éloigne, et inversement. De plus, la date et l'heure du passage sont définies avec le premier timestamp du passage.

Un des problèmes de ce traitement est que si deux voitures se suivent d'un peu trop près, il est possible que le Raspberry n'obtienne aucune valeur nulle. Ce problème peut être résolu en étudiant également l'évolution de la tension. En effet, si la tension est globalement décroissante, et que soudainement, il y a une forte augmentation, c'est qu'un nouvel objet passe devant le capteur. De la même façon, si la tension est croissante, s'il y a une forte diminution, c'est que le premier objet est sorti du champ et que la détection se fait maintenant sur le deuxième objet.

Stockage et affichage des données

Serveur SQLite

Une fois qu'une donnée est élaborée et renvoyée par le Raspberry, il faut pouvoir la stocker pour une interrogation par le client web. Pour cela, on crée une table relationnelle SQL qui contient les dates, les vitesse moyennes et maximum, ainsi que les sens de circulation de tous passages. Les données sont donc accessibles grâce à des requêtes SQL, ce qui permet de les afficher ensuite dans une interface web. De plus paramétrer les requêtes permet à l'utilisateur de filtrer les données selon les valeurs (voir « Fonctionnalités du client web »).

Client web

Au niveau du client, l'accès à l'administration des capteurs et à l'affichage des données se fait par un client http sur navigateur. Les fonctions sont l'affichage des passages, dans un simple historique ou sous la forme de graphiques synthétiques, ainsi que l'administration des capteurs, avec un système de logs.

Fonctionnalités du client web

L'interface utilisateur de l'application comporte une liste des capteurs avec leur état courant (Connecté/Déconnecté). Les données relatives au capteur sélectionné sont réparties dans 3 onglets :

L'ongle « données élaborées » permet de voir tous les passages dans une liste paginée. L'utilisateur peut filtrer les données en sélectionnant une plage de dates et/ou en définissant un seuil de vitesse minimum ou maximum.

L'onglet « données synthétisées » permet de faire des analyses paramétrées des données élaborées sous forme d'historique ou de distribution. La sélection des valeurs peut se faire sur le type de données (historique ou distribution du nombre de passages, des vitesses moyennes ou des vitesses maximum, sens de circulation) et/ou sur les timestamps (plage de dates et/ou plage horaire).

L'onglet « administrateur » contient la liste des évènements relatifs au capteur sélectionné (connexions et déconnexions). De la même façon que pour l'affichage des données, ces évènements peuvent être triées selon une plage de dates.

Conclusion

Dans l'état actuel de notre application, nous arrivons à obtenir les mesures du HB100 et à les transformer en vitesse sur l'Arduino. Malheureusement, comme nous l'expliquions ci-dessus, l'absence d'un circuit d'amplification des hautes fréquences nous empêche de mesurer des vitesses supérieures à 25km/h.

L'envoie sur le Raspberry Pi est correctement implémenté, et le traitement des échantillons permet bien d'isoler les passages et de calculer les vitesses maximum et minimum, ainsi que le sens de circulation. Les valeurs sont ensuite envoyées au serveur, excepté si la connexion est impossible, auquel cas les données sont stockées sur le Raspberry jusqu'au rétablissement de la connexion.

Nous n'avons pas implémenté autant de fonction que prévu au niveau de l'interface utilisateur, mais un affichage rudimentaire des données permet de constater que l'interrogation du serveur SQL se fait correctement. Le reste du travail à faire sur le site consiste à enrichir les fonctionnalités via des interrogations SQL paramétrées, ainsi que du code HTML et CSS pour enrichir le fonctionnement du client web, ce qui ne constitue pas le cœur du projet. C'est la raison pour laquelle nous avons mis la priorité.

Remerciements

Nous tenons à remercier l'entreprise Isandlatech, et tout particulièrement M. Olivier GATTAZ et M. Bassem DEBBABI, pour nous avoir fourni notre sujet, ainsi que le matériel nécessaire, et pour nous avoir accueilli dans leurs locaux pour répondre à toutes nos questions.

Annexes

Diagramme de composant :

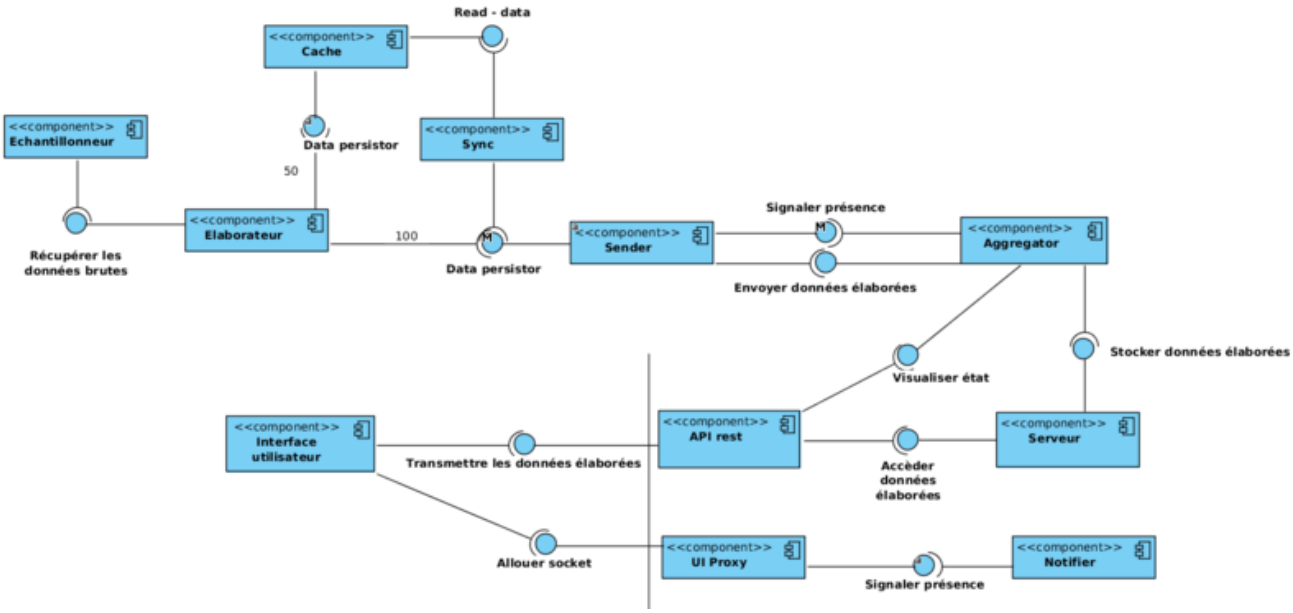
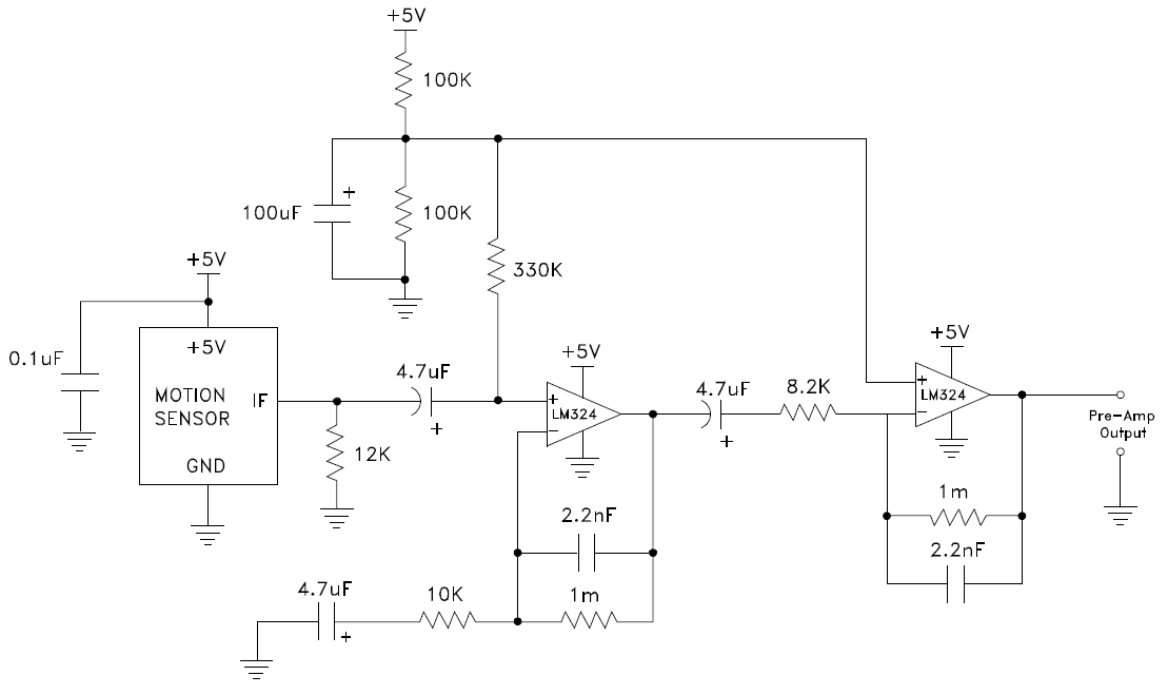


Schéma de câblage de l'amplificateur de fréquence :



Liens :

SRS du projet Tachymètre

Fiche de suivi du projet Tachymètre

Dépôt github du projet Tachymètre