

CONCEPTION D'UN SYSTEME DE SURVEILLANCE ENVIRONNEMENTALE BASE SUR UN RASPBERRY PI

Smart package monitoring

Réaliser par : Saad Kouzmane
Encadré par : Mr Anass Mansouri
SOFT EMBARQUE



Tableau de matières

Introduction	2
Chapitre 1 contexte général du projet	3
Contexte.....	3
Cahier de charge	3
Methodologie.....	4
Chapitre 2 conception et modélisation du projet	5
Environnement technique du projet	5
Architecture générale du projet	5
Conception détaillée du projet	6
Chapitre 3 Réalisation et test du projet	9
Réalisation du projet.....	9
Test d'application.....	10
Conclusion	13

Introduction

Le projet que nous allons aborder concerne la surveillance environnementale pendant la livraison de colis et de marchandises. Le transport de ces produits peut être soumis à différentes conditions environnementales telles que la température, l'humidité et la pression, qui peuvent affecter leur qualité et leur état. Afin de garantir la sécurité et la qualité des produits transportés, il est essentiel de surveiller et de contrôler ces conditions environnementales pendant le transport.

La solution proposée consiste en un système de surveillance environnementale qui utilise des capteurs connectés à Internet pour collecter des données en temps réel sur les conditions environnementales pendant la livraison. Les capteurs sont installés dans les véhicules de transport et dans les conteneurs de stockage des produits. Les données collectées sont transmises sans fil à une plateforme de surveillance en ligne qui permet aux utilisateurs d'accéder à ces données en temps réel.

Les utilisateurs peuvent surveiller les données collectées et recevoir des alertes en cas de variation des conditions environnementales en dehors des limites de tolérance définies. Cela permet aux utilisateurs de prendre des mesures pour corriger les conditions environnementales et garantir la sécurité et la qualité des produits transportés.

En résumé, ce système de surveillance environnementale offre une solution efficace pour surveiller et contrôler les conditions environnementales pendant le transport de colis et de marchandises, ce qui peut contribuer à garantir la qualité et la sécurité des produits transportés.

I. chapitre I contexte général du projet

1. Contexte général :

Le projet client-serveur que nous présentons est un système de surveillance environnementale qui permet de collecter et de stocker des données environnementales telles que la température, l'humidité, la latitude, la longitude et le temps. Ces données peuvent être collectées dans des colis, les serres et les entrepôts pour surveiller les conditions environnementales et ajuster en conséquence les paramètres pour une meilleure production.

L'objectif principal de ce système est de fournir des données environnementales en temps réel aux utilisateurs, ce qui leur permet de surveiller l'état de l'environnement et d'ajuster les paramètres pour améliorer la qualité de la production. Le système est basé sur une architecture client-serveur, où les capteurs environnementaux agissent en tant que clients pour collecter les données et les transmettre à une unité centrale de contrôle (le serveur), qui stocke les données dans une base de données.

L'application du système de surveillance environnementale est très large et peut être utilisée dans différentes applications telles que les entrepôts, et les installations industrielles. Dans l'agriculture, par exemple, les données environnementales peuvent aider à ajuster les paramètres tels que la température et l'humidité pour les conditions les plus favorables.

En résumé, ce système de surveillance environnementale est une solution pratique et efficace pour collecter et stocker des données environnementales en temps réel. En fournissant des données précises sur les conditions environnementales, ce système peut aider les utilisateurs à améliorer la qualité de leur production et à prendre des décisions informées en matière de politique environnementale.

2. Cahier de charge

Le cahier des charges du projet proposé est de développer un système de surveillance pour les colis qui nécessitent une surveillance de la température et de l'humidité pendant le transport. Le but est de s'assurer que les produits transportés restent dans des conditions optimales, en particulier pour les produits alimentaires, pharmaceutiques ou autres produits sensibles.

Le système de surveillance sera basé sur un ensemble de capteurs environnementaux qui seront placés dans chaque colis. Ces capteurs enregistreront les données de température et d'humidité ainsi que les données de positionnement à intervalles réguliers et les transmettront sans fil à une unité de contrôle centrale.

L'unité de contrôle central (Raspberry pi 4) sera équipée des capteurs qui analysera les données collectées et les stockera dans une base de données. Les utilisateurs pourront accéder à ces données à tout moment et en tout lieu via une interface Web conviviale, qui affichera les données sous forme de graphiques et de tableaux.

Le système de surveillance devra répondre aux critères suivants :

- Être capable de surveiller la température et l'humidité à l'intérieur des colis en temps réel
- Enregistrer et stocker les données de positionnement ainsi que les données environnementales pour chaque colis

- Alerter les utilisateurs en cas de dépassement des limites de température et d'humidité prédéfinies
- Permettre aux utilisateurs d'accéder aux données collectées à tout moment et en tout lieu via une interface Web conviviale
- Fournir des analyses et des rapports détaillés sur les données collectées pour aider les utilisateurs à comprendre les tendances et à prendre des décisions éclairées

Le principal objectif du projet est de développer une solution de surveillance environnementale efficace et fiable pour les colis transportés, en particulier pour les produits alimentaires, pharmaceutiques ou autres produits sensibles. La solution devra répondre aux exigences du cahier des charges tout en étant facile à utiliser et accessible pour les utilisateurs.

3. Méthodologie

La méthodologie suivie pour la réalisation de ce projet de surveillance de colis sera basée sur le modèle en V [1], qui permet de décrire et d'organiser les différentes phases de développement du projet. Voici les différentes étapes de la méthodologie :

Étude de faisabilité :

- Analyse des besoins du client
- Analyse de l'état de l'art des technologies de surveillance de colis
- Évaluation de la faisabilité technique, économique et temporelle du projet
- Définition des contraintes techniques, environnementales et réglementaires

Conception :

- Spécification des fonctionnalités et des interfaces du système de surveillance
- Définition de l'architecture matérielle et logicielle du système
- Élaboration des plans de tests et de validation
- Choix des composants matériels et logiciels

Développement :

- Réalisation des cartes électroniques et de l'assemblage du système
- Programmation des logiciels embarqués et de l'interface utilisateur
- Tests unitaires et intégration du système

Validation :

- Validation fonctionnelle et performance du système
- Vérification de la conformité aux spécifications et aux normes applicables
- Validation de l'interface utilisateur et des fonctionnalités

Industrialisation :

- Industrialisation des composants matériels et logiciels
- Mise en production et gestion de la chaîne logistique
- Déploiement du système chez le client

Maintenance :

- Maintenance corrective et préventive du système

- Gestion des incidents et des évolutions du système.

En utilisant cette méthodologie, nous pourrions garantir la qualité et la fiabilité de l'application de surveillance de colis, ainsi que la satisfaction du client en termes de performance et de fonctionnalités.

II. Chapitre II Conception et Modélisation

1. Environnement technique du projet

Le système utilise Raspberry Pi 4 B pour le traitement des données en temps réel et le stockage des données collectées dans une base de données. Pour la communication sans fil entre les capteurs et l'unité de contrôle centrale, nous utilisons la technologie Wi-Fi.

Pour la collecte de données environnementales, nous utilisons la bibliothèque Python Adafruit [2] pour le capteur de température et d'humidité DHT11, qui fournit des données précises et fiables sur les conditions environnementales. Nous avons également développé un programme en langage C pour la communication avec le capteur GPS NEO 6M. Ce programme ouvre la communication série et traduit les phrases NMEA en coordonnées correspondantes.

Et on a également utiliser un driver C pour contrôler deux LED (vertes et rouges) qui indiquent si la température est conforme ou non aux limites prédéfinies. Si la température dépasse ces limites, la LED rouge s'allume. Cela permet à l'utilisateur de savoir immédiatement si la température est trop élevée ou trop basse, ce qui peut avoir des conséquences sur les colis transportés, cet ajout permet à l'utilisateur d'avoir une interface visuelle pour surveiller l'état de la température et agir rapidement en cas de défaillance du système.

Pour le contrôle de la fenêtre d'aération, nous avons créé un module de pilote de noyau Linux [3] qui permet d'ajuster l'angle de la fenêtre selon l'angle désiré par un signal PWM. Cela permet à l'utilisateur de régler la température à l'intérieur de l'endroit où le système est installé.

En résumé, l'environnement technique de notre projet utilise une combinaison de technologies et de langages tels que Python, C et Linux kernel driver pour permettre la collecte et le traitement de données environnementales en temps réel, ainsi que le contrôle de la température via la fenêtre d'aération.

2. Architecture générale du projet

L'architecture générale du projet comprend trois parties principales : la partie client, la partie serveur et la base de données.

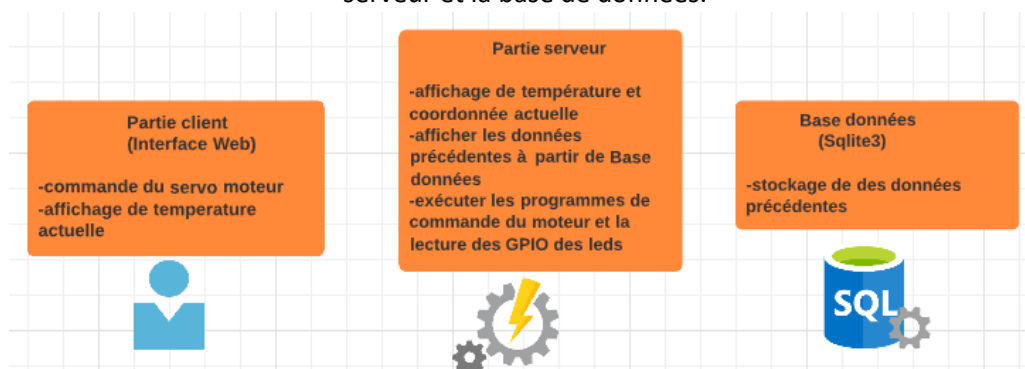


Figure1 : Vue général du projet

Partie client :

La partie client de l'application permet à l'utilisateur de contrôler l'angle de la fenêtre à partir d'un servo-moteur, et affiche également la température actuelle et les coordonnées GPS. Permettant à l'utilisateur de cliquer sur des boutons pour ajuster l'angle de la fenêtre en temps réel, et d'afficher les informations de température et de coordonnées GPS actuelles. Cette partie de l'application communique avec la partie serveur via des requêtes PHP.

Partie serveur :

La partie serveur de l'application est responsable de l'affichage de la température et des coordonnées GPS actuelles via un exécutable C et Python, et de la lecture de la broche GPIO de la LED, ainsi que de l'exécution du driver C du servo-moteur pour tourner l'angle de la fenêtre à la valeur souhaitée. Cette partie de l'application est développée en utilisant Python et C, avec une base de données SQLite3 pour stocker les données précédentes. Elle est hébergée par lighttpd. L'architecture est conçue de manière à être évolutive et flexible, permettant l'ajout de nouvelles fonctionnalités à l'avenir.

Partie base de données :

La partie base de données de l'application stocke les données précédentes de température et de coordonnées GPS dans une base de données SQLite3. Cela permet à l'application de récupérer facilement les données historiques et de les afficher à l'utilisateur. L'utilisation de SQLite3 offre également une grande fiabilité et une facilité d'utilisation pour l'application. Les données sont stockées dans une table unique avec une structure simple pour faciliter leur récupération et leur analyse. La partie serveur communique avec la base de données via des requêtes SQL pour stocker et récupérer les données nécessaires.

3. Conception détaillée du projet

Nous allons détailler la conception de notre application qui se divise en cinq parties distinctes:

La partie de stockage des coordonnées GPS :

Une composante essentielle de l'application, car elle permet de récupérer les coordonnées géographiques et de les stocker dans un fichier texte à des fins de traitement ultérieur. Dans cette partie, le code en C utilise le port série pour lire les données provenant du module GPS NMEA [4] connecté à la Raspberry Pi. Ensuite, il utilise une méthode de traitement de chaînes de caractères pour extraire les données de coordonnées géographiques pertinentes et les convertit en degrés décimaux. Enfin, il écrit ces données dans un fichier texte pour les stocker en permanence. Cela permettra de récupérer ces données et de les utiliser dans d'autres parties de l'application, telles que la partie d'affichage de la température ou la partie de stockage de données SQL.

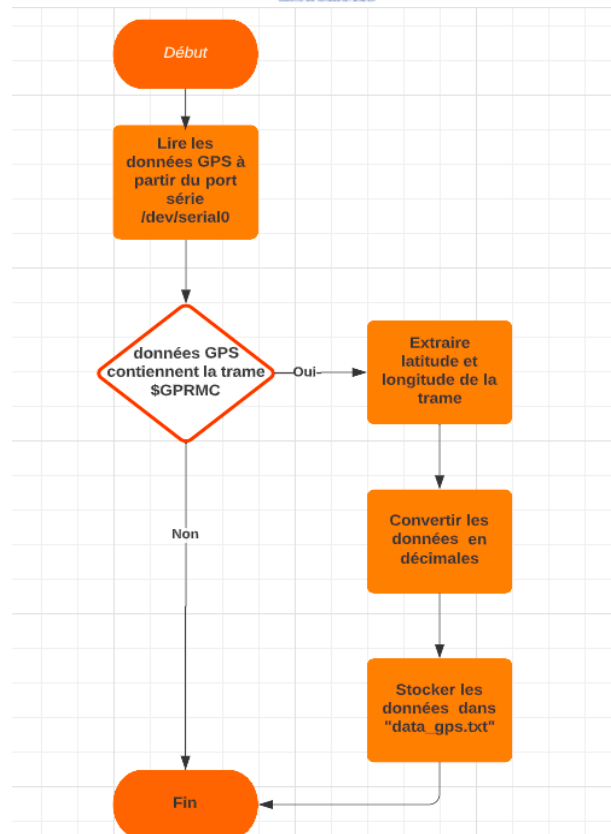


Figure2 : Organigramme du code pour la récupération des données GPS

La partie de stockage de température :

Cette partie du projet concerne la récupération des données de température et d'humidité à partir d'un capteur DHT11 connecté à la Raspberry Pi. Le capteur DHT11 est un capteur numérique capable de mesurer la température et l'humidité relative de l'air ambiant. et, nous les stockons dans un fichier texte nommé "data.txt". Les valeurs précédentes sont écrasées à chaque lecture.

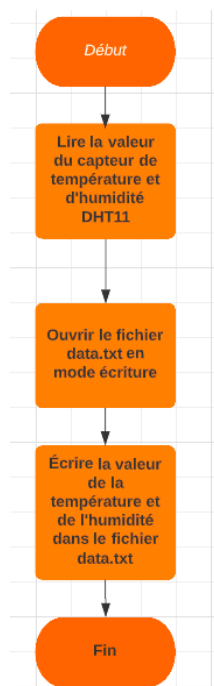


Figure2 : l'organigramme de lecture de la température et de l'humidité avec un capteur DHT11

La partie du contrôle d'un moteur de servo :

Nous utilisons un moteur de servo pour contrôler l'angle d'ouverture d'une vanne, en utilisant une application de device driver de servo réalisé et chargé dans le noyau, qui permet de fournir un signal de commande au moteur de servo. Dans le code, nous avons défini la plage d'angles possibles pour le moteur de servo, qui va de 0 à 90 degrés. Ensuite, nous récupérons un angle en tant qu'argument de ligne de commande et nous vérifions que cet angle se situe bien dans la plage autorisée, Si l'angle est valide, nous construisons une commande à exécuter en utilisant la fonction printf. Cette commande consiste à écrire l'angle dans le fichier driver "/dev/servo_motor" qui est utilisé pour contrôler le moteur de servo. Ensuite, nous exécutons cette commande en utilisant la fonction system.

La partie d'affichage d'état de température en LED :

Cette partie de code récupère les données de température et d'humidité à partir d'un fichier texte "data.txt" et allume ou éteint des LED en fonction de la température mesurée. la température est vérifiée pour déterminer si elle est supérieure à 30°C. Si c'est le cas, les LED sont allumées en utilisant la fonction "set_gpio_line" pour mettre la sortie à niveau logique haut (1). Dans le cas contraire, les LED sont éteintes en utilisant la même fonction pour mettre les sorties à niveau logique bas (0).

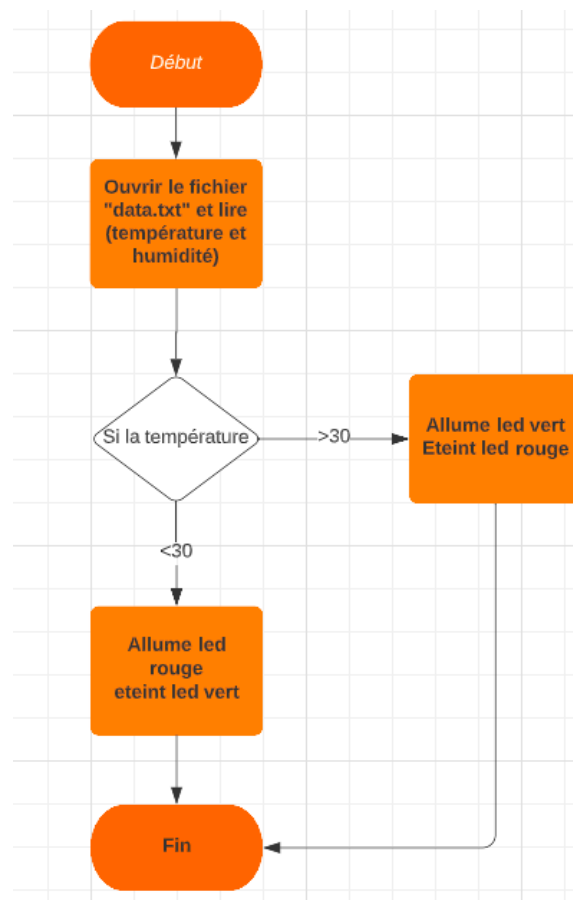


Figure4 : Organigramme du code de gestion de la LED en fonction de la température

La partie du stockage les données :

On stock les données température, d'humidité, de latitude, de longitude et de timestamp dans une base de données SQLite. Les données de température et d'humidité sont lues à partir du fichier "data.txt". Les données de latitude et de longitude sont lues à partir du fichier "data_gps.txt". Et sont ensuite insérées dans la table "sensor_data" en utilisant la méthode execute() de la connexion à la base de données. Les valeurs de température, d'humidité, de latitude, de longitude et de timestamp sont passées comme arguments à la méthode execute().

III. Un chapitre III Réalisation et test du projet

1. Réalisation du projet

Le projet est réalisé en utilisant plusieurs technologies et langages. Pour la partie capteur, nous avons utilisé le capteur DHT11 pour mesurer la température et l'humidité, ainsi que le module GPS pour obtenir les coordonnées géographiques. Ces données sont ensuite stockées dans un fichier .txt

Pour la partie actionneur, :

Nous avons utilisé le device driver de moteur servo de type (character device) permet de contrôler la position d'un moteur à travers un signal PWM. Il utilise un module du kernel Linux ([lien de mon driver sur GitHub](#)) pour gérer le PWM dans le kernel space et lit et écrit des données à travers un fichier de driver crée et peut être utilisé pour définir la position du moteur à travers une valeur numérique transmise par écriture au fichier du driver. La lecture du fichier retourne une chaîne de caractères avec des informations sur l'état du moteur

- On compile le driver avec le makefile
- On charge le driver dans le kernel : `sudo insmod \servo.ko`
- On donne permission à écrire dans le fichier driver : `sudo chmod 777 /dev/servo_motor`
- Et puis on utilise le code c qui permet d executer la commande :
`echo "angle_value" > /dev/servo_motor` pour faire tourner le moteur à l'angle désirer

Enfin, pour stocker et visualiser les données collectées, nous avons utilisé la base de données SQLite Les données stockées dans la base de données a partir d'un code python et peuvent être consultées via l'interface web que j'ai crée

L'ensemble du projet est interconnecté et fonctionne en continu pour surveiller les conditions de température et d'humidité par la page web.

La page web permet de contrôler un servo-moteur et afficher les lectures de température, d'humidité, de latitude et de longitude, ainsi que les données enregistrées dans une base de données.

- La première partie du code permet de contrôler le servo-moteur en envoyant une commande au serveur. Le code récupère l'angle souhaité depuis un formulaire de saisie et utilise la fonction `shell_exec()` pour exécuter l'application du driver `write_servo` qui définit l'angle souhaité pour le servo-moteur en utilisant le device driver. Le code affiche ensuite un message indiquant l'angle qui a été défini.
- La deuxième partie du code lit les données de température et d'humidité depuis un fichier `data.txt` et les affiche. Les données de latitude et de longitude sont lues depuis un fichier `data_gps.txt` et affichées. Les données de température sont également lues depuis le GPIO 22 et un message est affiché indiquant si la température est bonne ou mauvaise.

- La troisième partie du code interroge une base de données SQLite pour récupérer les données de température, d'humidité, de latitude et de longitude enregistrées dans la table data et les affiche dans un tableau. Le tableau est trié par ordre décroissant de timestamp.

Enfin, le code génère un lien vers Google Maps pour afficher la position GPS actuelle du paquet en utilisant les données de latitude et de longitude récupérées précédemment.

Test de l'application

Test de la lecture de la température :

- On s'assure que le capteur est correctement branché sur l'appareil (Pin 4) et que l'application est capable de détecter le capteur, et puis on exécute le programme "python3 dht11.py"

```
saad@pi:/www/c-bin $ python3 dht11.py
Temp=25.0*C Humidity=43.0%
```

Figure5 : exécution du code de température

- On lit le fichier data.txt:

```
saad@pi:/www/c-bin $ cat data.txt
25.0 43.0
```

Figure6 : commande de lecture de donne du fichier data.txt

Test de la lecture des données GPS :

- On lance l'application et on assure que l'appareil est connecté avec le capteur via serial 0.
- On place le capteur à l'extérieur ou dans un endroit où vous avez une vue dégagée du ciel jusque a la led début à clignoter signifie que le signal est bien reçu du satellite.

```
saad@pi:~ $ cat /dev/serial0
*****r0b0r0b0r0R00j
$GPGSV,3,1,09,01,37,275,17,02,22,062,44,10,31,044,45,16,25,170,*73
$GPGSV,3,2,09,21,58,307,22,22,14,120,39,23,00,037,,27,53,100,41*75
$GPGSV,3,3,09,32,20,095,50*4A
$GPGLL,3401.01602,N,00500.02056,W,150046.00,A,A*7A
```

Figure7 : Lecture des données NMEA du GPS depuis le port série /dev/serial

- On lance notre programme qui sert a prendre les donnes neesaire depuis la sentence NMEA Test de la fonction d'enregistrement de données :

```
saad@pi:/www/c-bin $ ./gps
Latitude: 34.017007, Longitude: -5.000242
saad@pi:/www/c-bin $ cat data_gps.txt
Latitude: 34.017007, Longitude: -5.000242
```

Figure8 : Exécution du code GPS et lecture de donne du fichier data_gps.txt

Test indicateur de la température :

- On connecte les broches GPIO à des LED ou d'autres périphériques pour indiquer la température.
- On exécute le programme sur la plateforme matérielle cible (par exemple, un Raspberry Pi). "/a.out" dans notre cas la température est de 26 et la led vert est allumée
- On modifie la valeur de la température dans le fichier "data.txt" avec une température plus grande et la led rouge s'allume (démonstration dans la vidéo).

Test insertion dans la base de données :

- Une fois que la base de données SQLite3 a été créée, nous pouvons tester notre code Python d'insertion de données dans la base.

```
saad@pi:/www/c-bin $ sudo sqlite3
```

Figure9 : Commande du lancement du sqlite3

```
saad@pi:~ $ sudo sqlite3
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open data_db.db
sqlite> SELECT * FROM data ORDER BY timestamp DESC
...> ;
27.0|38.0|34.017038|-5.000106|1683472581
```

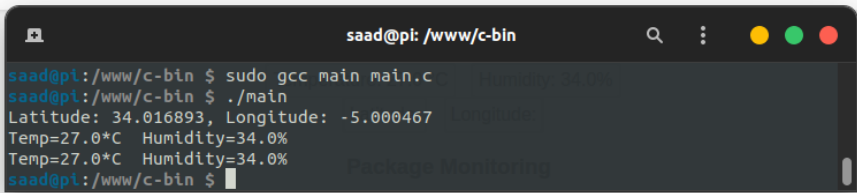
Figure10 : Affichage de la donnée de table SQL du projet

Test du code main:

Ce code en langage C utilise la fonction système pour exécuter plusieurs commandes. Il exécute les commandes suivantes :

- www/c-bin/gps" : cette commande exécute le script "gps" dans le répertoire "/www/c-bin/". On peut supposer que ce script permet de récupérer les données GPS.
- "python3 dht11.py" : cette commande exécute le script "dht11.py" en utilisant l'interpréteur Python 3. On peut supposer que ce script permet de récupérer les données de température.
- "sudo python3 sql.py" : cette commande exécute le script "sql.py" ce script permet d'insérer les données GPS et de température dans une base de données SQLite.
- "sudo /home/saad/a.out" : cette commande exécute un programme C qui allume la led correspondante Et voici le résultat de la fonction main :

Door angle (0-90):



Temperature	Humidity	Latitude	Longitude	Timestamp
27.0	34.0	34.016893	-5.000467	2023-05-07 16:42:44

Figure11 : Démonstration du stockage des données en temps réel dans l'interface web

Et puis on planifie la tache a s exécuter chaque minute avec crontab

```
# m h dom mon dow  command
* * * * * /www/c-bin/main
```

Figure12 : Planification de l'exécution du programme main à chaque minute

Test de la page web :

Voice La page web lit correctement la température et indique que la température est de 27 degrés. De plus, un lien vers la carte Google Maps affichant les coordonnées GPS actuelles est disponible.

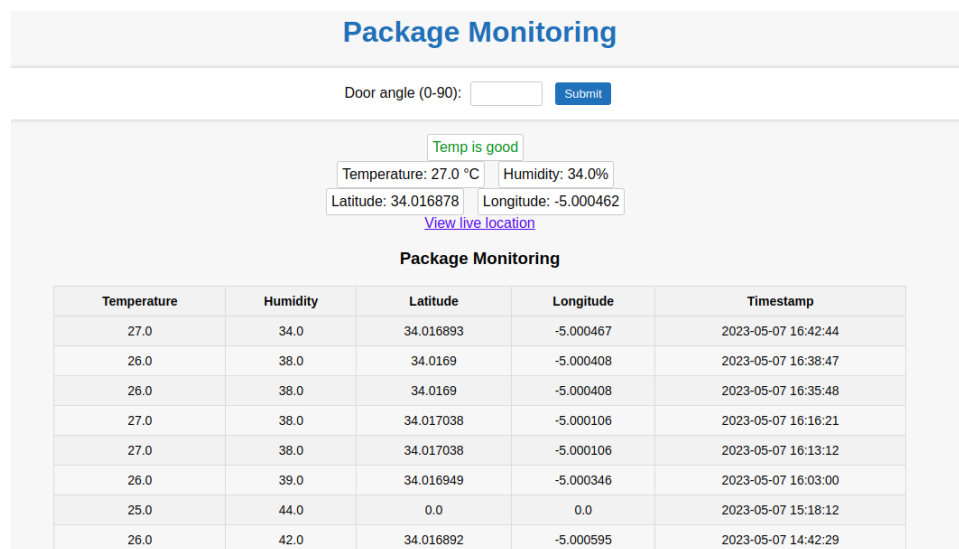


Figure13 : Interface web du projet

Et après qu'on clique sur View live location il nous dirige vers lien google map de coordonnée actuelle :

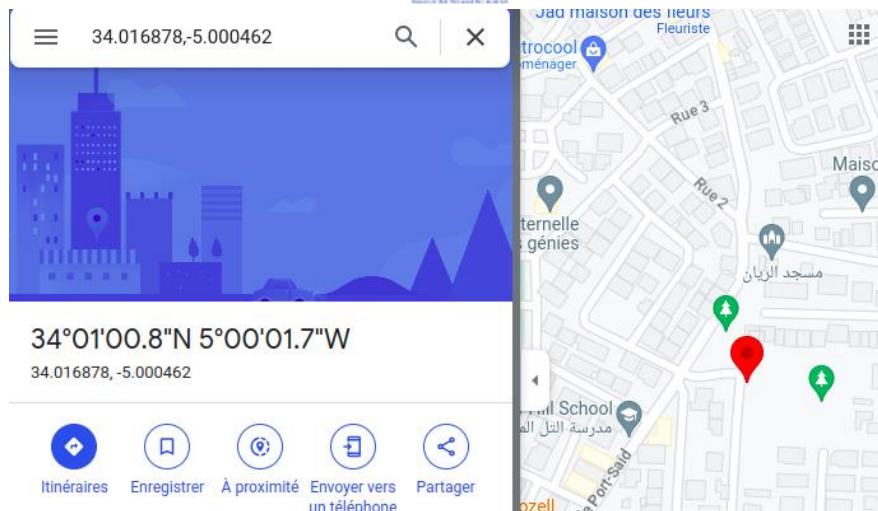


Figure14 : Visualisation de l'emplacement du colis à partir du l'interface web

Conclusion

En conclusion, ce projet a permis de concevoir et de développer une application qui permet de mesurer la température ambiante ainsi que de collecter les coordonnées GPS d'un appareil mobile.

Cette application peut être utilisée dans différents contextes, tels que la surveillance de l'environnement ou le suivi de la localisation d'un véhicule.

Nous avons utilisé un capteur de température DHT11 pour mesurer la température, et un module GPS pour collecter les données de localisation. Nous avons également créé une base de données SQLite pour stocker les données collectées, et nous avons conçu une interface utilisateur pour afficher les données sur une page web.

Nous avons testé chaque fonctionnalité de l'application, en vérifiant la précision des lectures de température et la récupération des données GPS, ainsi que la connexion à la base de données et l'affichage des données sur la page web.

Enfin, nous avons automatisé le processus en planifiant l'exécution du programme principal à chaque démarrage de l'appareil et en le faisant boucler. Cette automatisation permet une collecte continue de données sans avoir à exécuter manuellement l'application à chaque fois.

En somme, ce projet a permis de mettre en pratique plusieurs concepts de programmation embarquée tels que la lecture de capteurs et la conception des drivers personnalisés de différents niveaux kernel space (le cas de servo moteur) et user space le cas des autres drivers, et la manipulation de bases de données et la création d'interfaces utilisateur.

Reference :

- [1]"The V-Model: A Structured Process for Software Development and Testing", par Andreas Spillner et Tilo Linz
- [2]GitHub d'Adafruit : <https://github.com/adafruit>
- [3] Linux Device Drivers, Third Edition: <https://lwn.net/Kernel/LDD3/>
- [4]norme de communication de données pour les équipements de navigation maritime.
<https://www.nmea.org/>