17 janvier 2020

BORIBOUN Steven

ProxyDej

3 Rue carre de vigne, 44690, Monnières

## Table des matières

[Table des matières 1](#_Toc31097636)

[1 - Spécifications Techniques 2](#_Toc31097637)

[1.1 – Contexte 2](#_Toc31097638)

[1.2 – Micro-ordinateurs 2](#_Toc31097639)

[1.3 - Sonde de Température 3](#_Toc31097640)

[1.5 Afficheur 5](#_Toc31097641)

[1.5 Langage De Programmation 6](#_Toc31097642)

[1.6 Alimentation de la Raspberry de livraison. 7](#_Toc31097643)

[2 - L’analyse 8](#_Toc31097644)

[2.1 Le diagramme de cas d’utilisation 8](#_Toc31097645)

[2.2 Les diagrammes de séquences 8](#_Toc31097646)

# 1 - Spécifications Techniques

## 1.1 – Contexte

Pour notre projet, nous avons besoin de justifier la traçabilité de la chaîne du froid de nos produits. Ces produits sont transportés dans différents véhicules, pour l’instant il n’y a aucun moyen de justifier que les aliments ne dépassent pas la chaîne du froid. Pour ceci nous allons installer deux systèmes. L’un afin de se connecter sur le point d’accès Wi-Fi, puis récupérer les informations de deux capteurs (position géographique et de température), pour ensuite les envoyer ces informations sur un serveur. Si la connexion entre le serveur et le système est interrompu, le système doit enregistrer ces informations sur son espace mémoire. Puis quand elle est revenue, envoyer les données non transmises. L’autre afin de récupérer les informations d’un capteur de température et d’afficher celle-ci sur un écran LCD. Le serveur indiquera si la température est trop haute ou non. Si celle-ci dépasse la température maximum, elle enverra des alertes sur différents support de communication afin d’alerter les employés.

## 1.2 – Micro-ordinateurs

Dans le contexte du projet, il nous faut deux **micro-ordinateurs**, pour la gestion des deux capteurs de température et du capteur de géolocalisation. Ces micro-ordinateurs permettront de gérer le flux d’information reçu et l’enverra sur le serveur de l’entreprise.

Les Micro-ordinateurs devront respecter ces conditions :

* Doit posséder un **espace de stockage**.
* Doit pouvoir **se connecter en Wi-Fi**.
* Doit pouvoir **résister un minimum le froid**.
* Doit avoir des **pins des entrées et des sorties**.

Les deux propositions de micro-ordinateurs sont :

* **Raspberry PI3B+**
* **Arduino UNO Rev3**

Ces deux **micro-ordinateurs** sont connus dans leur milieu. Néanmoins, ils ne sont **pas faits pour les mêmes utilisations**, nous listerons leurs caractéristiques afin de choisir le plus **adapté**.

Vous trouverez ci-dessous le tableau comparatif entre le Raspberry PI3B+ et l’Arduino UNO Rev3 :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Raspberry PI3B+** | **Arduino UNO Rev3** | **Meilleur choix** |
| Consommation électrique | 5V | 7-12V | Raspberry |
| Température de fonctionnement | 0-50°C | Non communiqué | Raspberry |
| Ports entrées/sorties | 40 pins | 12 pins | Raspberry |
| Langage de programmation | Python | C | Raspberry |
| Mémoire vive | 1GB | 2 kB | Raspberry |
| Emplacement carte SD | Oui | Non | Raspberry |
| Connexion Wi-Fi | Oui | Non | Raspberry |
| Prix | ≃ 40€ | ≃ 20€ | Arduino |

Notre choix final sera donc, le **Raspberry PI3B+**. Il nous faudra quatre Raspberry, un pour chaque véhicule de livraison, plus undans la chambre froide.

Pour le fonctionnement d’une carte Raspberry, il faut :

* Un **Système d’alimentation 5V** environ
* Une **carte micro SD** avec un **système d’exploitation flashé**

**Pour la simplification du projet, nous allons installer et programmer une seule carte (celles qui seront dans les véhicules). Ensuite, nous allons la cloner pour les deux autres véhicules.**

## 1.3 - Sonde de Température

Nous avons besoin d’utiliser des **Sondes de température** afin de surveiller la Température d’une chambre froide ainsi qu'à l’intérieur des véhicules de livraison dans le but d’effectuer une traçabilité du froid des véhicules et de la chambre.

Les Sondes Que nous aurons à choisir devront respecter ces conditions :

* Doit pouvoir être **connectée à un Raspberry** directement ou **via GPIO.**
* Doit être la **moins encombrante** **possible** (plus petite, moins de câble).
* Doit être **peu onéreuse**.

Les deux Sonde à nos dispositions sont les suivantes :

* **DS18B20**
* **DB130-10**

La **DS18B20** est une sonde assez connue dans le milieu des sondes de températures et est produite par l’entreprise **Maxim integrated**. En opposition peu d’informations sont à disposition sur la **DB130-10**, nous utiliserons les caractéristiques de celles produites par l’entreprise **DFROBOT** pour nos comparatifs.

Veuillez trouver ci-dessous un comparatif des caractéristiques de chacune des sondes ainsi que celle que nous avons choisi pour ce projet :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **DS18B20** | **DB130-10** | **Meilleur Choix** |
| **Consommation électrique** | 3V à 5,5V | 5V | **DS18B20** |
| **Type d’interface** | Câble unique | Interface 2 câbles | **DS18B20** |
| **Plage de température** | -55°C à +125°C | -10°C à 80°C | **DS18B20** |
| **Taille** | 30mm x 6mm | 49mm x 14mm | **DS18B20** |
| **Longueur du Cable** | 91cm | 90 cm | **DB130-10** |
| **Prix** | 10€ | 27€ (29.50$) | **DS18B20** |

Notre Choix de Sonde final sera donc la **DS18B20**, il nous en faudra 4, une pour la chambre froide et une pour chaque Véhicule de Livraison.

Nous avons trouvé plus intéressant de **conserver la même sonde pour chaque cas d’utilisation.**

1.4 - GPS

Nous serons amenées à récupérer des **informations GPS** pour reproduire le parcours d’un véhicule de livraison et récupérer toutes les informations souhaitées en fonction d’une position a un temps donné. Par conséquent des r**écepteurs GPS** sera nécessaire.

Les Récepteurs GPS Que nous aurons à choisir devront respecter ces conditions :

* Doit avoir une **faible consommation électrique.**
* Doit être la **moins encombrante possible** (plus petite).
* Doit être **peu onéreuse**.

Les deux Récepteurs GPS à nos dispositions sont les suivants :

* **BU-353**
* **Here 2**

Le récepteur GPS BU-353 est produit par **Globalsat**, une entreprise spécialisée dans les **GPS Tracker, récepteurs GPS...**

**Pixhawk 2** est un **projet** datant de **2008** qui vise à fournir du **matériel de contrôle de pilotage** (autopilote) **haut de gamme**, à **coûts bas ou raisonnables** à des communautés d'utilisateurs universitaires, de loisirs et industrielles.

Veuillez trouver ci-dessous un comparatif des caractéristiques de chacun des Récepteurs GPS ainsi que celui que nous avons choisi pour ce projet :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **BU-353** | **Here 2** | **Meilleur Choix** |
| **Chipset GPS** | L1 SiRF Star III | QZSS | **BU-353** |
| **Interface** | USB | JST-GH | **BU-353** |
| **Fréquence trames NMEA** | 1 Hz | Max 10 Hz | **Here 2** |
| **Récepteur 20 canaux** | L1, C/A code | L1 C/A, GLONASS L10F | **Here 2** |
| **Dimensions** | 53mm\*19mm | 76mm x 16,6 mm | **BU-353** |
| **Consommation** | 80 mA | 2.5A | **BU-353** |
| **Précision** | 10m (2D RMS) | 2,5 m | **Here 2** |
| **Prix** | 39€ | 115€ | **BU-353** |

On choisira par conséquent Le **Globalsat BU-353** car il est plus **économique** et **moins** **consommateur** en énergie.

## 1.5 Afficheur

Pour pouvoir afficher les données de la sonde chargée d’observer la température, nous aurons besoin d’un **afficheur** à l’entrée de la chambre froide nous donnant toutes les informations que l’on souhaite

Les différents **afficheurs** que nous aurons à choisir devront respecter ces conditions :

* Doit avoir une **faible consommation électrique.**
* Doit être la **moins encombrante possible** (plus petite).
* Doit être **peu onéreuse**.
* Doit être **tactile.**

Les deux afficheurs à nos dispositions sont les suivants :

* **Raspberry Pi 7’’ Touchscreen Display**
* **Raspad**

Le récepteur GPS BU-353 est produit par **Globalsat**, une entreprise spécialisée dans les **GPS Tracker, récepteurs Gps...**

**Pixhawk 2** est un **projet** datant de **2008** qui vise à fournir du **matériel de contrôle de pilotage** (autopilote) **haut de gamme**, à **coûts bas ou raisonnables** à des communautés d'utilisateurs universitaires, de loisirs et industrielles.

Veuillez trouver ci-dessous un comparatif des caractéristiques de chacun des afficheurs ainsi que celui que nous avons choisi pour ce projet :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Touchscreen Display** | **Raspad** | **Meilleur Choix** |
| **Consommation** | Avec la Raspberry (3 à 5,5V) | 6000mAh | **Touchscreen Display** |
| **Dimension** | 194mm x 110mm x 20mm | 25,4 cm de diagonale | **Touchscreen Display** |
| **Tactile** | Oui | Oui | **Equivalent** |
| **Connectique** | DSI port | USB | **Touchscreen Display** |
| **Résolution** | 800\*480 pixels | 1280\*800 pixels | **Raspad** |
| **Fréquence** | 60Hz | 60Hz | **Equivalent** |
| **Prix** | 60€ | 200€ | **Touchscreen Display** |

On choisira par conséquent Le **Raspberry Pi 7’’ Touchscreen Display** car il est plus économique et **moins** **consommateur** en énergie. De plus sa **connectique** est plus adaptée dans notre contexte.

## 1.5 Langage De Programmation

Enfin la partie la plus importante a nos yeux , nous devons choisir un langage de programmation dans lequel

Les différents **afficheurs** que nous aurons à choisir devront respecter ces conditions :

* Doit être **facile** **d’utilisation**
* Doit être **dynamique (**pour le **temps réel)**

Les deux langages à notre disposition sont :

* **Python**
* **C++**

D’autres langages peuvent être utilisés mais ces deux-là sont les deux **meilleures options** sur un système comme une **Raspberry**

Veuillez trouver ci-dessous un comparatif des caractéristiques de chacun des afficheurs ainsi que celui que nous avons choisi pour ce projet :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Python** | **C++** | **Meilleur Choix** |
| **Compilation** | Besoin d’un interpréteur | Pré-compilé avant exécution | **C++** |
| **Efficience** | Language Simplifié | Language traditionnel | **Python** |
| **Difficulté** | Facile | Difficile | **Python** |
| **Collection des déchets** | Oui | Non | **Python** |
| **Cross-platform** | Oui | Non | **Python** |
| **Langage Dynamique** | Oui | Non | **Python** |

On choisira par conséquent Le **Python**, car parmi les deux il est le seul langage dynamique, de plus il est beaucoup plus **simple** d’utilisation que le **C++.**

## 1.6 Alimentation de la Raspberry de livraison.

La Raspberry a besoin d’alimentation de 5V 2.5A en micro USB pour fonctionner.

Il faut que l’alimentation soit :

* Peu consommatrice en énergie (5V 2.5A).
* Que l’alimentation soit éteinte quand la voiture est à l’arrêt

Nous avons deux solutions qui sont à notre disposition :

* Un adaptateur allume-cigare à micro USB de 5V
* Régulateur 5V/12V

Veuillez trouver ci-dessous un comparatif des caractéristiques de chacun des afficheurs ainsi que celui que nous avons choisi pour ce projet :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Adaptateur allume-cigare 5V** | **Régulateur 5V/12V** | **Meilleur Choix** |
| **Consommation** | Sortie : 5V/1A | Sortie : 5V/2.4A | **Régulateur 5V/12V** |
| **Longueur des câbles** | 1.20 m | 5 cm | **Régulateur 5V/12V** |
| **Difficulté de mise en place** | Plug-and-Play | Besoin d’une installation | **Adaptateur allume-cigare 5V** |
| **Courant qui passe plus quand la voiture et à l’arrêt** | Oui | Oui | **Equivalent** |
| **Protection de surintensité** | Non | Oui | **Régulateur 5V/12V** |
| **Prix** | 10.90€ | 13.90€ | **Adaptateur allume-cigare 5V** |

On choisira par conséquent **Régulateur 5V/12V**, car parmi les deux, c’est celui qui gardera un courant stable peu importe l’intensité de la voiture.

## 1.7 Symfony

# Symfony est un framework français open source pour PHP. Les frameworks, en général, permettent un gain de productivité mais pas seulement. Ils permettent d'avoir une bonne organisation du code et on sait qu'un code bien organisé est plus maintenable et évolutif. Symfony est open source. On n'a donc pas de contraintes imposées et on peut développer des solutions propriétaires. Il est considéré comme un des framework PHP les plus puissants et les plus flexibles. Un des principal inconvénients d'un framework est le temps d'apprentissage qui est bien plus élevé que pour un langage en lui-même. Symfony, plus particulièrement nécessite d'apprendre plusieurs autres technologies qui lui sont propres comme YAML ou Doctrine.

Une Doctrine ORM implémente 2 patterns objets pour mapper un objet PHP à des éléments d'un système de persistance :

* Le pattern "Data Mapper" ;
* Le pattern "Unit of Work".

Le Data Mapper est une couche qui synchronise la donnée stockée en base avec les objets PHP. En d'autres termes :

* Il peut insérer, mettre à jour des entrées en base de données à partir de données contenues dans les propriétés d'un objet ;
* Il peut supprimer des entrées en base de données si les "entités" liées sont identifiées pour être supprimées ;
* Il "hydrate" des objets en mémoire à partir d'informations contenues en base.

L'implémentation dans le projet Doctrine de ce Data Mapper s'appelle l'Entity Manager, les entités ne sont que de simples objets PHP mappés.

# 2 - L’analyse

## 2.1 Le diagramme de cas d’utilisation

Le diagramme de cas d’utilisation permet de montrer les différents types de scenarios que les acteurs peuvent rencontrer lors de l’utilisation d’un projet. Ces scenarios permettront de créer des diagrammes de séquences. Voici ci-dessous le diagramme de cas d’utilisations pour notre projet :



Figure 1 Diagramme de cas d'utilisation du projet

## 2.2 Les diagrammes de séquences

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique. Voici le premier diagramme permettant