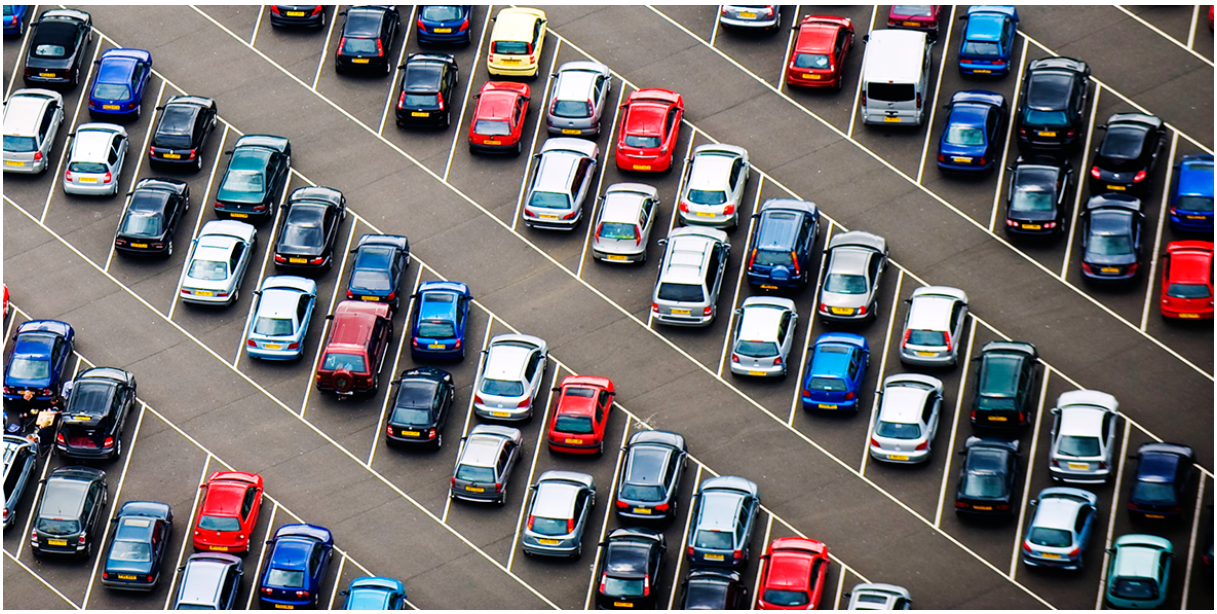
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Mejjad Mohamed Amine
Moumen Abderazzak
Quiya Zhang

4AE-SE

RAPPORT DE PROJET DE PROGRAMMATION ORIENTÉ OBJET

SMART PARKING



2021-2022

SOMMAIRE

Introduction.....3
Présentation du projet.....3
Diagramme de classe et code.....4
Conclusion.....5

INTRODUCTION :

Dans le cadre de l'UF Langage C++, il nous a été demandé de réaliser un bureau d'études dont l'objectif est de réaliser un système innovant. Nous avons choisi, après de longues réflexions, de réaliser un parking intelligent.
Le projet en question a été codé en langage orienté objet C++ en utilisant des capteurs et des actionneurs. Ce BE a donc pour but à nous familiariser avec la notion d'orienté objet et surtout à affiner nos connaissances acquises en cours et TD comme les notions d'héritage, encapsulation, polymorphisme...
Le support principal est une carte Arduino ESP8266 fournie par Expressif Systems et qui comporte la fonctionnalité WIFI. Nous avons utilisé l'IDE d'Arduino pour pouvoir implémenter notre code sur la carte et nous lui avons ajouté nos différents capteurs et actionneurs.

Entrées	**Sorties**
2x Capteurs ultrasons	2x Servomoteurs
	1x Ecran OLED

PRESENTATION DU PROJET :

Notre idée du projet est de simuler la gestion d'un parking, nous avons donc mis en place deux barrières (commandées par deux Servo Moteurs), deux capteurs de présence (Capteur à Ultrason) et un afficheur OLED.

Le tableau ci-dessous résume tous les capteurs et les actionneurs que nous avons utilisés ainsi que les différents Pin où ils sont connectés :

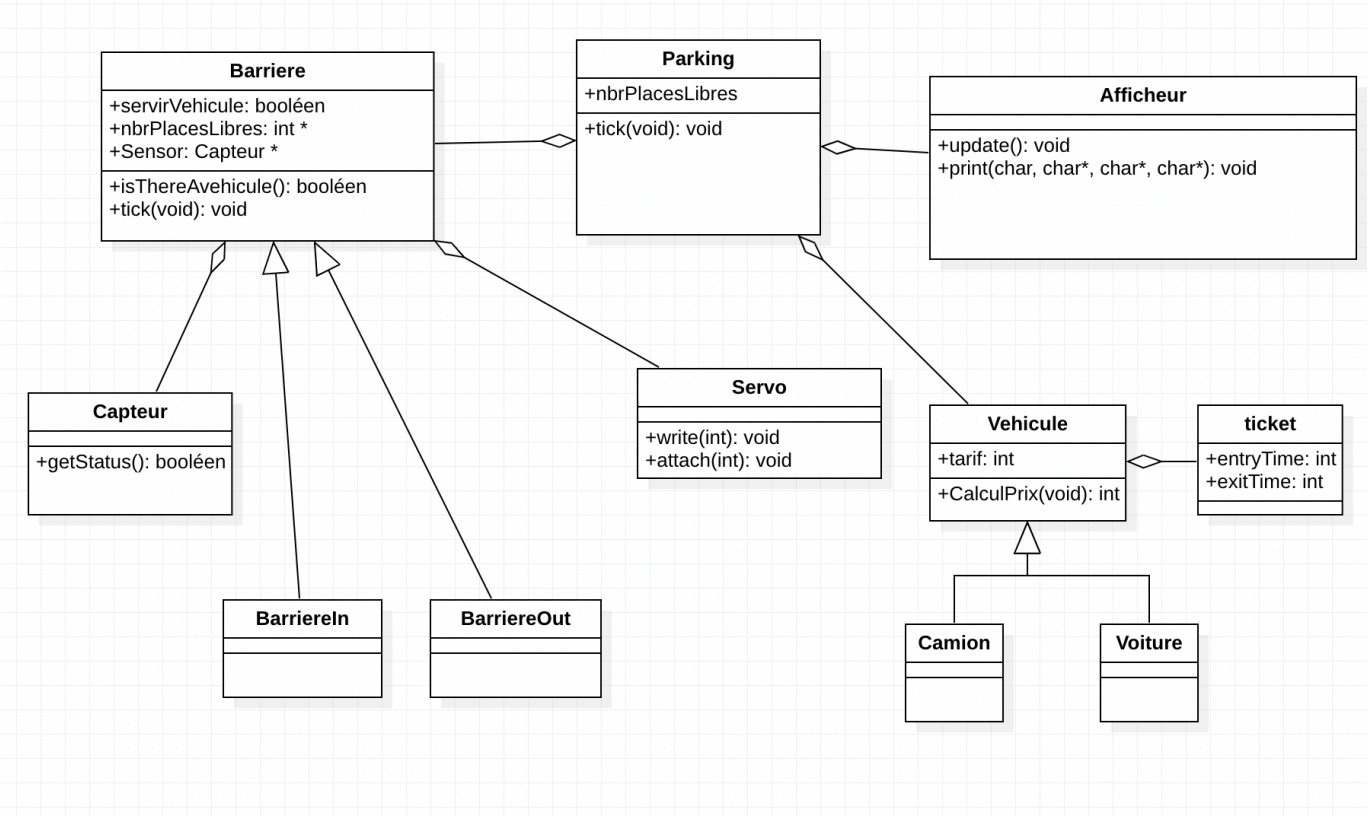
Capteurs/Actionneurs	Pins
1er capteur d'accès : UltraSonic Sensor HC-SR04	Trig-> GPIO12 (pin D6) Echo-> GPIO14 (pin D5)
2ème capteur de sortie : UltraSonic Sensor HC-SR04	Trig-> GPIO2 (pin D4) Echo -> GPIO13 (pin D7)
1er servo moteur actionnant la Barriere d'entrée	Connecté à GPIO0 (pin D3)
2ème servo moteur actionnant la Barriere de sortie	Connecté à GPIO15 (pin D8)
Afficheur OLED	GPIO5 & GPIO4 (pin D1 & D2)

Notre projet est un prototype d'un parking intelligent. Nous l'avons réalisé sur une breadboard. Sur celle-ci nous avons mis la carte ESP8266 au centre. D'un côté (côté 1) de la breadboard, nous avons mis un écran OLED ainsi qu'un capteur ultrasons et un servomoteur. De l'autre côté (côté 2), nous avons mis un capteur ultrasons et un servomoteur.

Le côté 1 est l'entrée de notre **parking**. Ainsi, un utilisateur peut voir sur l'afficheur OLED le nombre de places disponibles. S'il y a des **places libres**, l'écran « Soyez les bienvenues ». S'il n'y a pas de places libres, le message « Revenez plus tard » est ainsi affiché.
Dans le cas où il y a des places disponibles : en se rapprochant de la **barrière d'entrée**, un **capteur** ultrasons relève la présence d'un **véhicule** permettant ainsi à la barrière de commander le **servomoteur** afin de permettre l'accès et laisser la voiture entrer.
Dans le cas où il n'y a pas de places, même s'il y a une voiture en attente pour entrer dans le parking, l'accès lui sera refusé. Il faudra donc attendre qu'une place se libère.

Le côté 2 est la sortie de notre parking. Elle fonctionne comme l'entrée. Si un véhicule veut sortir du parking, le capteur ultrasons relève sa présence et il envoie l'information à la **barrière de sortie**. Cette dernière commande le servomoteur pour laisser passer le véhicule.
Chaque fois qu'un véhicule entre dans le parking, le nombre de places disponibles se décrémente d'une place. Si une voiture sort alors le nombre des places libres est incrémenté automatiquement d'une place mais quand le nombre de places disponibles atteint le nombre maximum des places du parking, la barrière de sortie ne s'ouvre plus quand le capteur de sortie capte une présence.

DIAGRAMME DE CLASSE :



Notre diagramme de classe se compose de plusieurs classes : Parking, Barrière, BarriereIn, BarriereOut, Afficheur, Capteur, Servo, Véhicule, Camion, Voiture et ticket.
Les classes Camion et Voiture sont des véhicules alors héritent de la classe Véhicule et de même les classes BarriereIn et BarriereOut héritent de la classe Barrière.

Ces classes comportent des attributs et des méthodes qui **ont été implémenté dans le code** pour pouvoir réaliser notre projet comme la méthode print () de la classe Afficheur qui nous sert à afficher un message à l'écran ou bien la méthode getStatus () de la classe Capteur qui sert à calculer la distance de l'Ultrason et de déterminer s'il y a présence ou non d'un véhicule en retournant soit true soit false.

Nous pouvons aussi améliorer notre programme en rajoutant des capteurs Ultrasons en dessus de nos capteurs existants pour pouvoir faire la distinction entre un camion et une voiture (Voiture va être captée que par le capteur du bas alors que le Camion va être capté par les deux capteurs) et rendre aussi le parking payant en rajoutant une class ticket qui va comptabiliser le temps de présences des véhicules dans le parking pour calculer le prix d'utilisation.

Dans notre code, on a pu utiliser des exceptions, la notion de redéfinition d'opérateurs, polymorphisme (au niveau de la classe Barriere), forward déclaration (en utilisant class Parking ; avant la classe Afficheur qui devait utiliser un objet de la classe Parking), héritage...

Ensuite, pour ce qui concerne les difficultés rencontrées lors du projet, c'est surtout le nombre de Pin de la carte qui était limité et donc nous n'avons pas pu rajouter les deux capteurs à Ultrasons pour pouvoir faire la distinction entre les types de véhicules (voitures, camions...). Aussi, on a souffert d'un grand manque de temps surtout qu'on voulait changer de carte pour pouvoir avoir plus de Pin et pourvoir améliorer notre programme en utilisant la STL.

CONCLUSION :

Ce bureau d'étude a été d'un grand aide pour nous. Il nous a permis de mieux assimiler les notions vues en cours et de pouvoir les appliquer. Il nous a permis aussi de combler certaines lacunes du codage ainsi que d'apprendre à utiliser l'IDE d'Arduino.

C'était aussi l'occasion pour faire preuve d'imagination et innovation. Comme on l'a cité avant, plusieurs fonctionnalités auront pu être ajouter à notre projet si seulement on n'avait pas ces contraintes technologiques et logistiques.

Travailler en autonomie et en total liberté nous a permis de faire nous même un cahier de charge, de créer un planning de travail et aussi de bien maitriser les phases de conduite d'un projet.