Mathilde Ibled

Raphaël Rio

Mercredi 19 Mai 2021

Rapport du Bureau d'étude de C++

2020 – 2021

Le Robot "suiveur"

Professeur : David Gauchard

4AE-SE Groupe n°4

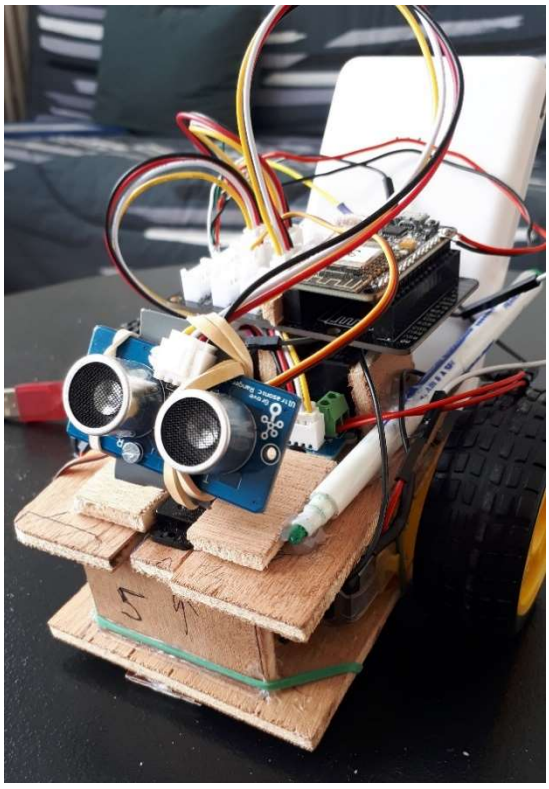
Introduction :

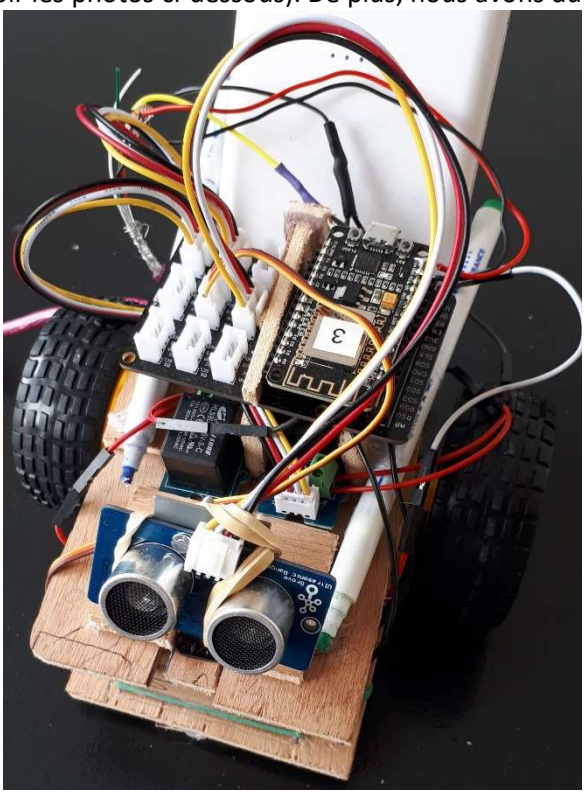
Lors du BE de C++, nous avons choisi de réaliser un « robot suiveur ». Ce robot est capable de détecter la présence d'une personne en mouvement dans une pièce, de se diriger vers elle et de la suivre. Pour réaliser ce projet, nous avons eu besoin de plusieurs capteurs : 3 capteurs de mouvement (PIR), 1 servo moteur, 1 capteur à ultrason, 2 roues, 2 moteurs, 2 relais, 1 batterie portable et 1 microcontrôleur ESP8266. Dans ce projet, nous devions mettre en place des relations d'héritage, des exceptions, la redéfinition d'opérateur et utiliser la librairie STL. Dans la suite de ce rapport, nous vous expliquons comment nous avons procédé pour ce projet. Vous trouverez les fichiers de notre projet sur un dépôt git via ce lien : https://github.com/RIO-Raphael/BE_Cpp

Un README est également présent afin d'expliquer la mise en œuvre du projet et comment le faire fonctionner.

Conception :

Tout d'abord, nous avons créé le support de notre robot pour intégrer les différents capteurs et actionneurs nécessaires à son fonctionnement (voir les photos ci-dessous). De plus, nous avons dû





Structure de notre robot fini

lester le châssis par la suite avec les moyens du bord car l'ajout de la batterie à l'arrière l'a déstabilisé. Les roues n'adhéraient plus très bien car le centre de gravité était trop reculé. Le robot avait donc une fâcheuse tendance à faire la toupie.

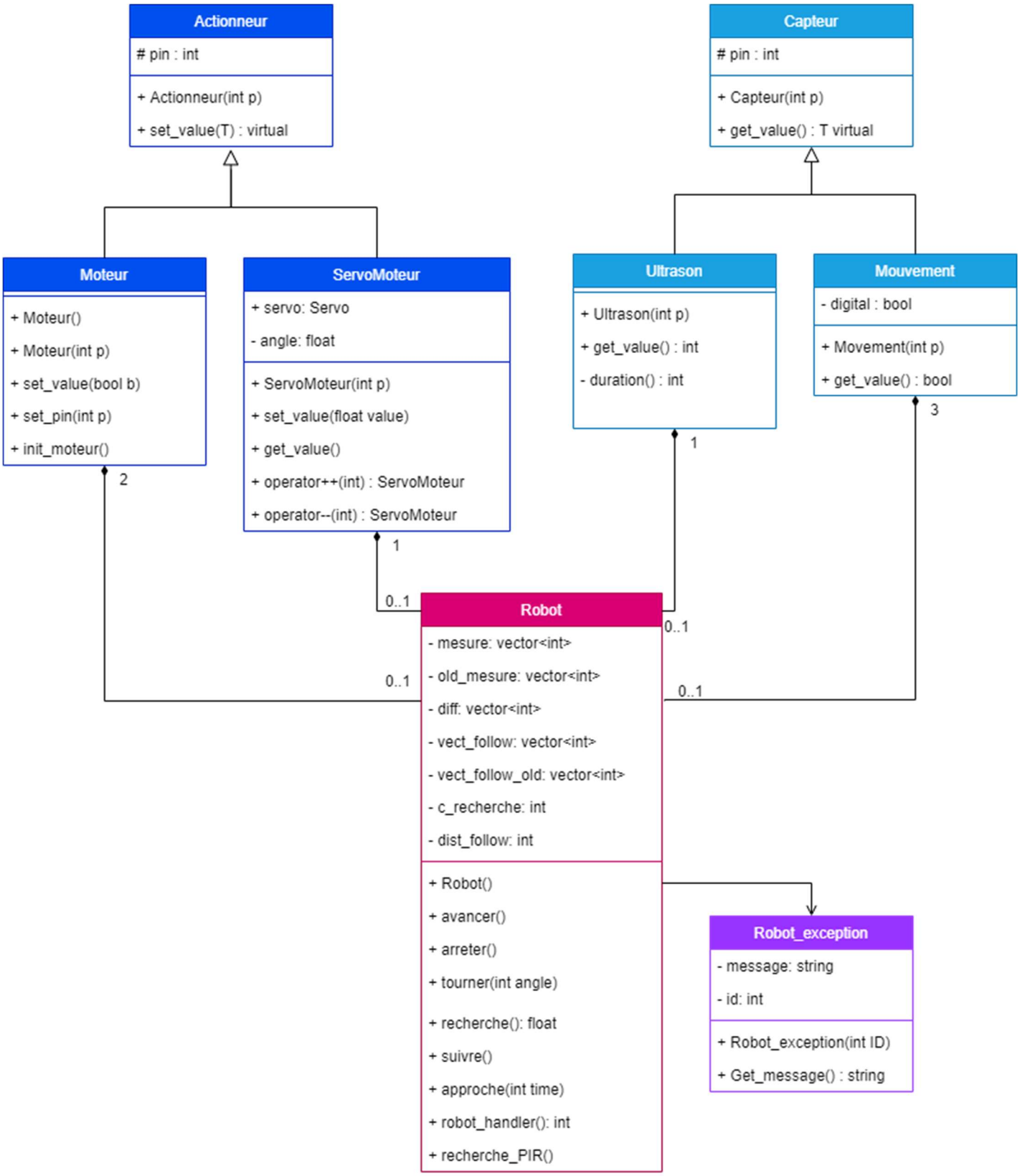
De plus, nous avons dû adapter la tension d'alimentation pour pouvoir utiliser les moteurs. En effet, l'alimentation en 3.3 V des ports n'était pas suffisante. Nous avons donc utilisé une alimentation externe que nous avons relié à l'entrée 5V de la carte et aux moteurs grâce à l'utilisation de relais. La batterie fournie par notre enseignant n'amener pas assez de courant à nos moteurs donc nous avons

dû également utiliser une autre batterie. L'utilisation de relais ne nous permettait pas de régler la vitesse des moteurs grâce à un PWM. Nous pouvons soit avancer, soit tourner en arrêtant un des deux moteurs et en actionnant l'autre. Cela convenait amplement pour notre application.

Les classes :

Puis, nous avons mis en place les différentes classes de notre projet. En premier lieu, nous avons mis en place une classe Capteur et une classe Actionneur. Les prochaines classes héritent de l'une ou de l'autre classe selon sa nature. Ainsi, il y a la classe Ultrason et Mouvement héritant de la classe Capteur. Les classes ServoMoteur et Moteur héritent de la classe actionneur. Ainsi, ces nouvelles classes héritent de la méthode get_value() et de l'entier pin qui correspond aux ports de branchement du capteur en question. La définition de ces classes se trouve dans le fichier « header.h » et leurs constructeurs et méthodes sont implémentés dans le fichier « source.cpp ».

Ensuite, nous avons mis en place une classe robot qui regroupe les différentes méthodes liées au fonctionnement de notre robot. Elle est définie dans le fichier « control.h ». Le constructeur de cette classe instancie les différents objets du robot explicités précédemment. Lui et les méthodes de la classe robot se retrouve dans le fichier « control.cpp ». Pour finir, le fichier « constantes.h » regroupe les différentes constantes de notre projet. Etrangement, les constantes définies dans header.h n'était pas visible dans le reste du code. Ainsi, il nous suffit de changer les paramètres seulement dans ce fichier au lieu de le changer dans tout le projet. Cela apporte une certaine portabilité à notre projet, notamment vis-à-vis des ports utilisés pour brancher les capteurs et les actionneurs. Ces relations entre classes sont représentées dans le diagramme de classe ci-dessous.


Actionneur
pin : int
+ Actionneur(int p)
+ set_value(T) : virtual
Capteur
pin : int
+ Capteur(int p)
+ get_value() : T virtual
Moteur
+ Moteur()
+ Moteur(int p)
+ set_value(bool b)
+ set_pin(int p)
+ init_moteur()
ServoMoteur
+ servo: Servo
- angle: float
+ ServoMoteur(int p)
+ set_value(float value)
+ get_value()
+ operator++(int) : ServoMoteur
+ operator--(int) : ServoMoteur
Ultrason
+ Ultrason(int p)
+ get_value() : int
- duration() : int
Mouvement
- digital : bool
+ Movement(int p)
+ get_value() : bool
Robot
- mesure: vector<int>
- old_mesure: vector<int>
- diff: vector<int>
- vect_follow: vector<int>
- vect_follow_old: vector<int>
- c_recherche: int
- dist_follow: int
+ Robot()
+ avancer()
+ arreter()
+ tourner(int angle)
+ recherche(): float
+ suivre()
+ approche(int time)
+ robot_handler(): int
+ recherche_PIR()
Robot_exception
- message: string
- id: int
+ Robot_exception(int ID)
+ Get_message() : string
2
1
0..1
1
0..1
3
0..1
0..1

L’organisation du code :

Le fichier « main.ino » permet de faire fonctionner le robot. Dans ce fichier on trouve une fonction de setup permettant d’initialiser le robot. Ensuite, les fonctions à faire à chaque itération sont regroupées dans la fonction loop du fichier arduino. Dans cette boucle, nous lançons la méthode robot_handler() qui regroupe les méthodes faisant fonctionner le robot. On y retrouve la méthode recherche() qui effectue la recherche de l’ultrason et la méthode recherche_PIR() qui permet la recherche des capteurs PIR. Nous avons mis en place une exception permettant d’arrêter le robot lorsqu’il détecte une distance avec le capteur ultrason inférieur à 5 cm, car cela impliquerait qu’il soit très proche d’un mur ou d’un obstacle.

Les difficultés :

Lors de ce projet, nous avons eu certaines difficultés. Tout d’abord, ce fut compliqué de bien commencer le projet car nous n’avions pas tous les composants nécessaires. Ainsi, nous avons codé nos classes et nos méthodes en les testant avec des LEDs pour les moteurs. Nous avons également dû réfléchir à la manière d’utiliser nos trois capteurs de mouvements. En effet, le nombre de port digital étant limité nous avons dû adapter notre code pour pouvoir utiliser le port analogique.

Conclusion :

Pour conclure, lors du déroulement de ce projet nous avons atteint notre objectif qui était de créer un robot suiveur. Malgré quelques difficultés, nous avons su les résoudre. Notre robot suit les mouvements d’une personne qui se déplace lentement près de lui car le temps d’exécution de nos commandes est assez lent particulièrement pour la détection grâce aux capteurs à ultrason. De plus les PIR ne sont pas très réactifs et ont parfois du mal à détecter les mouvements. Nous avons quand même pu structurer et coder notre projet de manière orientée objet.