VOITURE

AUTONOME

Sommaire

Annexe

Comprendre le projet

Introduction

Avec l'essor des voitures autonomes, et l'incroyable valorisation de Tesla®, nous avons décidé de nous laisser nous aussi dans la conception de ce type de véhicule. Pour cela, nous allons utiliser deux types de capteur : des capteurs de distance SHARP, et un Lidar RPLidar A2M8.

Objectifs du projet :

- **Phase capteurs de distance SHARP :**
    - La voiture devra être capable d'avancer, reculer et tourner.
    - Les capteurs devront être capables de repérer un obstacle et de renvoyer l'information à l'utilisateur (ou à la machine).
    - Associer voiture et capteur pour permettre à la voiture d'éviter un obstacle latéral, et s'arrêter en cas d'obstacle frontal.
    - La voiture devra être capable de faire des manœuvres en cas d'obstacle frontal (marche arrière + braquer) et d'obstacle latéral.
- **Phase Lidar :**
    - La voiture devra être capable d'avancer, reculer et tourner
    - Les capteurs devront être capables de repérer un obstacle et de renvoyer l'information à l'utilisateur (ou à la machine)
    - Asservir en vitesse la voiture (ralentir progressivement à l'approche d'un obstacle frontal)
    - Etre capable de suivre une direction privilégiée (i.e. suivre la direction qui lui permet d'aller le plus loin/avec le moins d'obstacle).

Découpage fonctionnel

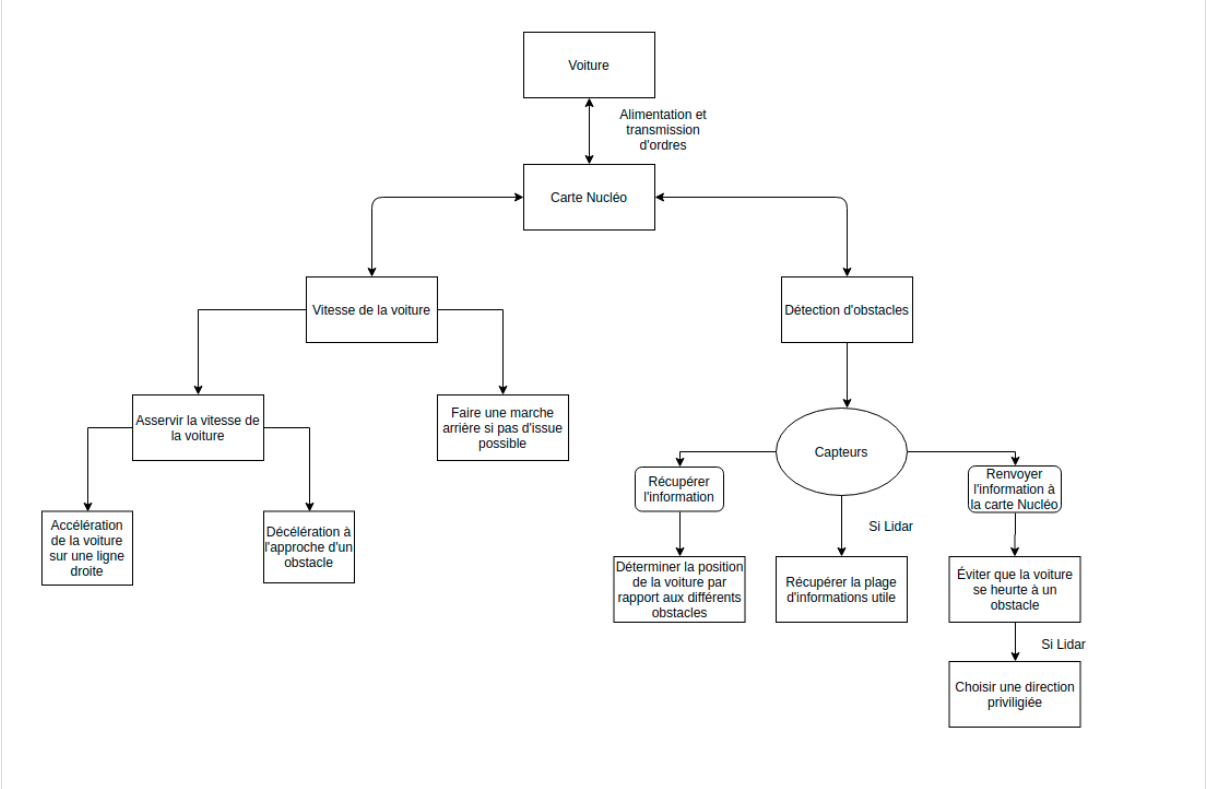


Figure 1 - Schéma fonctionnel de la voiture autonome

Schéma électrique

Pour l'électronique nous avons utilisé au cours de ce projet une carte sur laquelle étaient reliés tous les composants cela a permis de gagner beaucoup de temps concernant le câblage.

Liste des composants utilisés :

→ Carte Nucléo

→ servomoteur standard

→ contrôleur de vitesse standard

→ 3 capteurs de vitesse (type SHARP)

→ Lidar (RPLidar A2M8)

→ batterie NIMH de 7,2 V

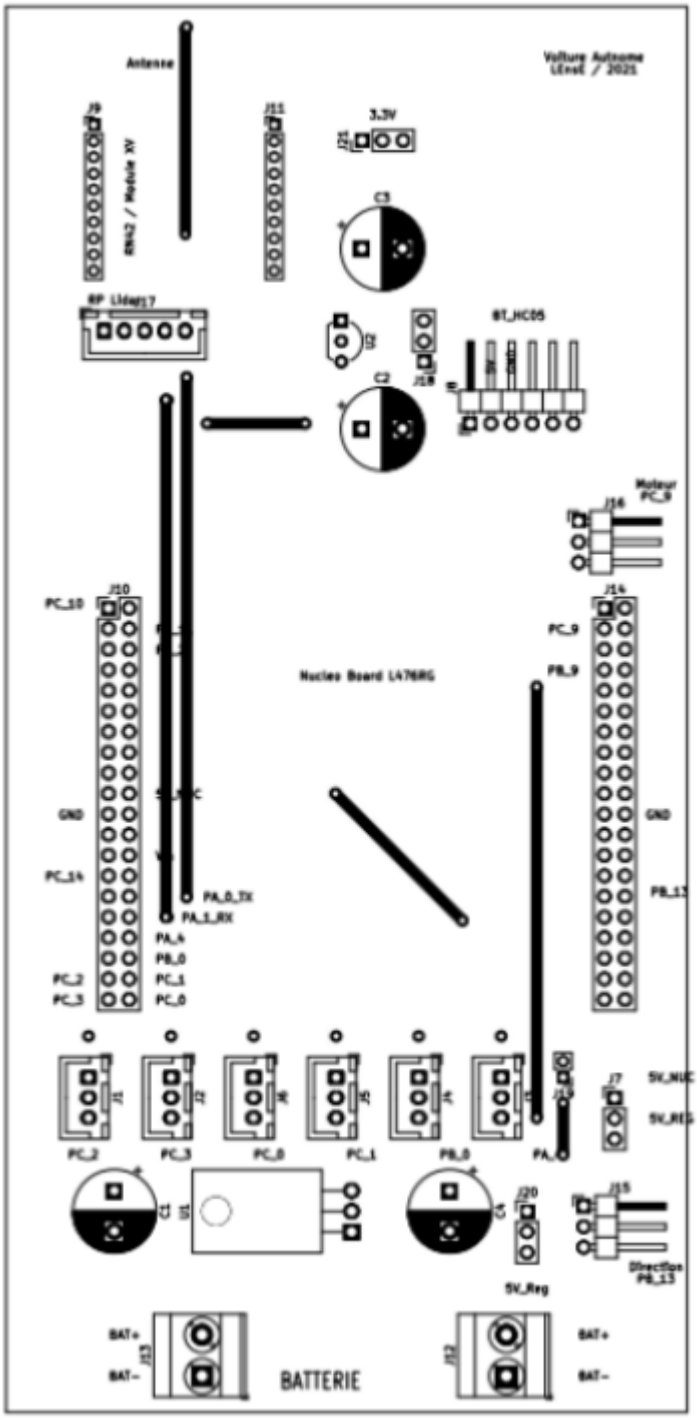


Figure 2 - Schéma électrique de la carte nucléo

Réaliser le prototype

Algorithme (code direct en annexe)

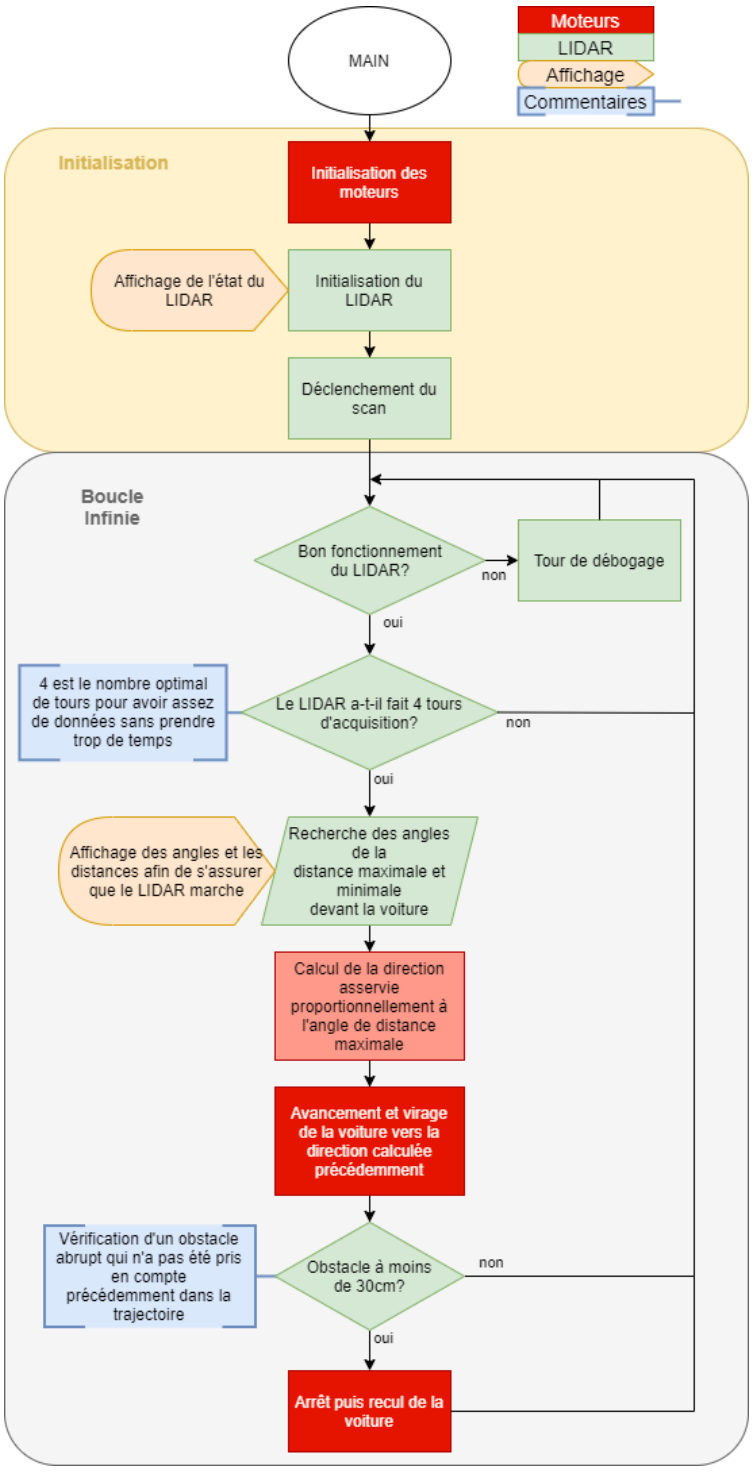


Figure 3 - Schéma de l'algorithme de la voiture autonome

Valider et caractériser le système

Caractérisation des capteurs.

Avant d'implémenter le LIDAR sur notre voiture autonome, nous avons cherché à la faire fonctionner simplement avec 3 capteurs (un frontal et deux latéraux). Nous avons donc dû caractériser les capteurs, c'est-à-dire quelles étaient leur plage de fonctionnement (tension de sortie en fonction de la distance au capteur).

On a ainsi constaté que ces capteurs étaient assez limités, ne détectant des obstacles qu'à une distance inférieure à 1m50 environ. Par ailleurs, les capteurs ne détectaient pas des obstacles situés à une distance plus petite que 10 cm approximativement, ce qui nous empêchait d'éviter un obstacle qui pourrait apparaître « soudainement ».

Caractérisation du LIDAR

Nous avons constaté que le LIDAR avait quant à lui une plage de détection beaucoup plus importante, pouvant « apercevoir » des murs ou du mobilier à plus d'une 15^{aine} de mètres. Le LIDAR a cependant comme les capteurs du début une « limite basse » l'empêchant de voir des obstacles à moins d'une dizaine de cm.

Test du parcours de la voiture autonome :

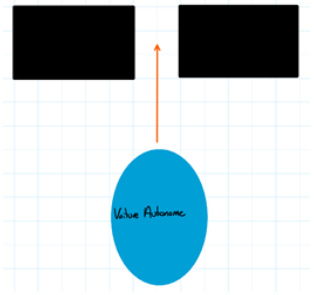
Afin de valider le cahier des charges et de constater le bon fonctionnement de la voiture, nous avons entrepris d'effectuer des tests dans les couloirs de l'IOGS et du LEnsE. En effet, cet environnement dispose de suffisamment de place pour permettre à la voiture d'effectuer différentes manœuvres de façon assez aisée, tout en proposant divers obstacles (colonnes, escaliers, portes, mobilier de grande taille). L'avantage est qu'il n'y a pas de « petit obstacle » que les capteurs ne pourraient pas détecter en raison de leur trop petite taille.

Nous avons ainsi mis en évidence plusieurs points :

- La voiture est capable de se déplacer en toute autonomie (avancer, tourner, reculer) dans son environnement à une vitesse d'environ 8 km/h, en évitant les divers obstacles sur son chemin. La voiture est capable de trouver assez rapidement (toutes les 3/5 secondes) une nouvelle distance maximale afin d'adapter sa trajectoire en temps réel.

- La voiture n'est pas capable de s'arrêter devant l'obstacle « jambe d'étudiant ».

- La voiture n'est pas capable de modifier sa trajectoire si elle est dans la situation suivante :



Comprendre les étapes de réalisation

Planning

Séance 1	26 janvier	• Prise en main de la voiture • Prise en main des capteurs (tracé de la caractéristique) • Planification des objectifs
Séance 2	2 février	• Réalisation d'un plan des différentes missions du LEnsE utiles à notre projet • Lecture et analyse de ces missions • Première version du code d'évitement des obstacles avec les capteurs distances • La voiture roule pour la première fois
Séance 3	9 février	• Finition du cahier des charges • Réalisation du schéma bloc de la voiture • Première version du code de marche arrière en cas d'obstacle frontal • La voiture contourne son premier obstacle frontal
Séance 4	23 février	• Apparition de la mascotte du BDS en salle de TP électronique • Prise en main du Lidar RPLidar A2M8 (analyse des données renvoyées, visualisation des points de la salle pour mieux comprendre)
Séance 5	31 mars	• Accueil d'un nouveau membre : Jesus • Premier code pour faire avancer la voiture avec le Lidar
Séance 6	7 avril	• La voiture effectue son premier trajet avec le RPLidar A2M8
Séance 7	12 mai	• Tentative de marche arrière avortée par un problème sur le RPLidar A2M8 que nous n'avons pas réussi à résoudre
Séance 8	25 mai	• Oral

Difficultés rencontrées

Les difficultés furent nombreuses lors de ce projet. Mais grâce à un travail d'équipe magistral et une cohésion à toute épreuve, nous avons réussi à (presque) toutes les surmonter.

- Le câblage de la voiture. Nous n'avons pu obtenir la carte électronique qu'à partir de la fin du mois de mars.. Nous avons donc du câbler nous même les capteurs de distance SHARP, ce qui nous prenait au moins 20-30 minutes au début de chaque séance.
- Effectuer la marche arrière. Nous ne réussissions pas à faire reculer la voiture. Nous devions en fait repasser deux fois par le "point mort" (mouvement nul du contrôleur de vitesse) avant de pouvoir faire la marche arrière.
- Acquisition des données du RPLidar A2M8. Le Lidar transmet trop de données pour la carte Nucléo. Il fallait donc déterminer comment choisir ces données, et comment les exploiter. Ce ne fut pas chose aisée, et finalement, Monsieur Villemejane nous a apporté la solution.
- Gestion de l'état de santé du Lidar. Malheureusement, lors de notre dernière séance, nous n'avons pas pu tester nos codes à cause d'un dysfonctionnement du Lidar.

Annexe

Code sous Mbed

```
/* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */

#include "mbed.h"
#include "rplidar.h"

#define BLINKING_RATE_MS        500
#define NB_DATA_MAX             20
#define AFF_DATA                0

char            pc_debug_data[128];
char            received_data[64];
int             data_nb = 0;
int             data_scan_nb = 0;
char            mode = LIDAR_MODE_STOP;
char            scan_ok = 0;
int             distance_scan[360] = {0};
int             distance_scan_old[360] = {0};
char            tour_ok = 0;
char            trame_ok = 0;

UnbufferedSerial    pc(USBTX, USBRX, 115200);
DigitalOut          led(LED1);
DigitalOut          debug_data(D10);
DigitalOut          debug_tour(D9);
DigitalOut          debug_out(D7);
DigitalOut          data_ok(D5);
DigitalOut          data_ok_q(D4);

PwmOut servo_motvitesse(PC_9);
PwmOut servo_motdirection(PB_13);

UnbufferedSerial    lidar(A0, A1, 115200);
PwmOut              rotation(PB_9);

struct lidar_data   ld_current;


/** MAIN FUNCTION
 */
int main()
{
    /** Initialisation moteurs*/
    servo_motvitesse.period_ms(20);
    servo_motvitesse.pulsewidth_us(1500);
    servo_motdirection.period_ms(20);
    servo_motdirection.pulsewidth_us(1300);

    /**Initialisation LIDAR*/
    int nb_tour = 0;
    wait_s(3.0);
    rotation.period(1/25000.0);
    rotation.write(0.4);
    wait_s(2.0);
    pc.write("\r\nLIDAR Testing\r\n", sizeof("\r\nLIDAR Testing\r\n")+1);
    lidar.attach(&IT_lidar);
    wait_s(1.0);
    pc.write("\r\nLIDAR OK\r\n", sizeof("\r\nLIDAR OK\r\n")+1);

    getHealthLidar();
    getInfoLidar();
    getSampleRate();

    // Start a new scan
    startScan();

    // Infinite Loop
    while (true) {
        if(trame_ok){                   //vérifie que le LIDAR est en bon fonctionnement
            debug_tour = !debug_tour;   //sinon il déclenche un tour de debogage
        }

        if(tour_ok == 4){               //le LIDAR a besoin de 4 tours d'acquisition pour avoir des données exploitables
            tour_ok = 0;
```

```
77
78              int maxDistance, maxAngle, minAngle, minDistance;
79              //recherche de la distance maximale (sans obstacles) devant la voiture (entre -60° et 60°)
80              findMax(distance_scan_old, -60, 60, &maxDistance, &maxAngle);
81              //recherche de l'obstacle le plus proche juste devant la voiture
82              findMin(distance_scan_old, -17, 17, &minDistance, &minAngle);
83
84              /*Affichage des valeurs pour tester le code*/
85              print_int("Amax", maxAngle);
86              print_int("Dmax", maxDistance);
87              print_int("Amin", minAngle);
88              print_int("Dmin", minDistance);
89
90              //la direction est asservie proportionnellement a l'angle de distance maximale
91              int dir = 1300 + 6*maxAngle;
92              servo_motvitesse.pulsewidth_us(1570);           //marche avant
93              servo_motdirection.pulsewidth_us(dir);          //virage vers la distance maximale  .
94
95              /*Cas d'obstacle abrupt*/
96              if (1<minDistance and minDistance<300){         //si l'obstacle le plus proche est à moins de 30cm
97                  servo_motvitesse.pulsewidth_us(1500);       //la voiture s'arrête
98                  wait_s(1.0);
99                  servo_motvitesse.pulsewidth_us(1430);       //pour faire marche arrière on a besoin
100                 wait_s(1.0);                                //de passer par le point mort 2 fois
101                 servo_motvitesse.pulsewidth_us(1500);
102                 wait_s(1.0);
103                 servo_motvitesse.pulsewidth_us(1430);       //fait marche arrière
104                 wait_s(2.0);
105                 servo_motvitesse.pulsewidth_us(1500);       //met le point mort et s'apprête a reprendre
106                 }                                           //en prenant compte du nouvel obstacle
107         }
108     }
109 }
```