Dossier Technique
Voiture Autonome
2022

Lisa FISSON
Ewen YOU
Anthony DONABEDIAN
David HAYOT

Introduction

Nous avons choisi ce sujet car nous nous intéressons à l'électronique embarquée. De fait, concevoir un véhicule autonome qui puisse se diriger sans intervention humaine nous semblait être le plus ambitieux dans ce domaine. Nous avons alors imaginé lui donner la consigne d'aller vers un point défini de manière autonome mais le matériel, le temps et la complexité du projet ont dû nous faire changer nos objectifs.

Cahier des charges

Pour notre voiture autonome nous souhaitions concevoir un prototype qui se déplace librement sans se prendre d'obstacles.

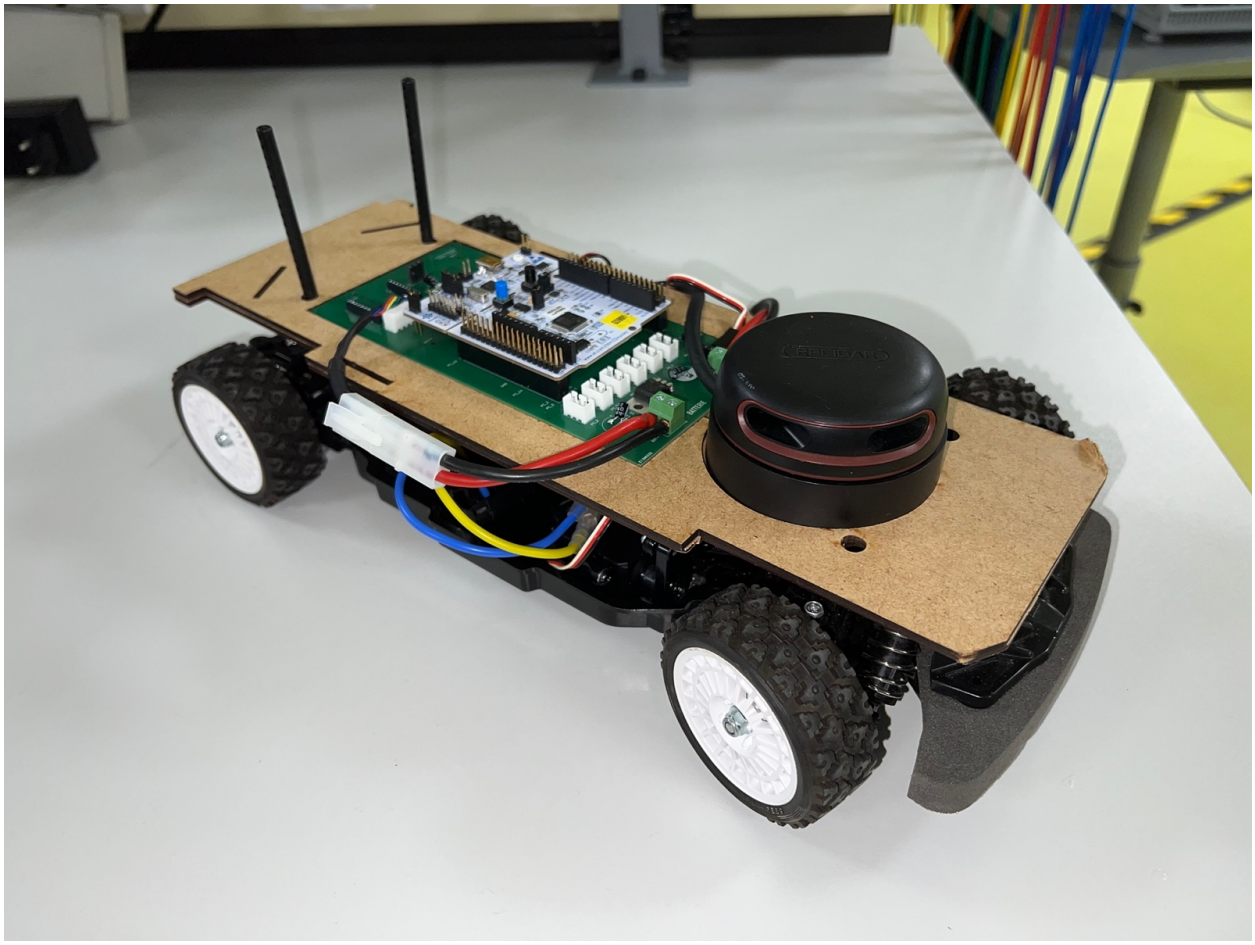
Ainsi nous avons choisi de la faire se diriger vers les espaces les plus dégagés accessibles et d'adapter sa vitesse en fonction de l'encombrement de l'environnement. Nous avons également défini qu'il était nécessaire de prévoir un arrêt d'urgence pour ne pas dégrader le matériel.

Nous avons donc défini le cahier des charges de notre voiture autonome comme tel :

1. Déplacement
    - 1.1 Avancer
        - 1.1.1 Vitesse maximale de 5 km/h
        - 1.1.2 Vitesse de croisière de 4 km/h
        - 1.1.3 Vitesse variable
    - 1.2 Tourner
        - 1.2.1 Vitesse de rotation des roues : 200° /s
        - 1.2.2 Tourner minimum à 90°
        - 1.2.3 Angle limite sur le servomoteur : +-(10°-15°)
2. Détection
    - 2.1. Récupérer une carte 2D des environs avec un lidar
        - 2.1.1 Acquisition en continu des distances en fonction de l'angle
        - 2.1.2 Fréquence d'acquisition de 10 ms maximum
        - 2.1.3 Récupération des données par une carte Nucléo
    - 2.2 S'arrêter lorsque la distance avec obstacles est inférieure à 20 cm
3. Commande
    - 3.1 Être commandée à distance
        - 3.1.1 Temps de réponse après la commande de 1.5s maximum
        - 3.1.2 Se déplacer d'un point A à un point B en évitant les obstacles
        - 3.1.3 Ajuster la trajectoire avec une fréquence de 1s
4. Énergie
    - 4.1 Batterie rechargeable de 7.2 V
        - 4.1.2 Temps de recharge totale maximum de 10h
        - 4.1.3 Autonomie de 1h minimum
5. Structure
    - 5.1 Supporter un masse d'environ 3 kg sans casser

Montage

Dans ce projet nous disposions d'une plaquette sur laquelle connecter la carte Nucléo aux autres composants. Nous avons donc pu connecter facilement la batterie, le régulateur du moteur à courant continu, le servomoteur de direction ainsi que le Lidar à la carte de contrôle.



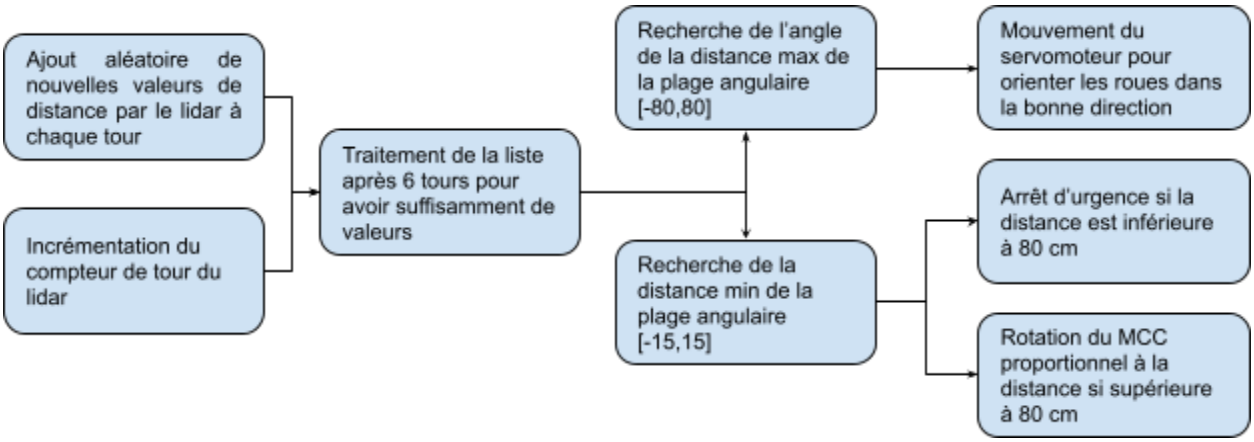
Nous pouvons voir sur la photo ci dessus l'intégration du lidar (cylindre noir). sur une plaque de bois solidaire du châssis. Cette plaque supporte également la plaquette électronique connectant la carte Nucléo (en blanc) avec les autres composants électroniques (en vert). Le moteur à courant continu et son régulateur, le servomoteur ainsi que la batterie étant déjà présents sur le modèle réduit qui nous sert de base, ils sont intégrés au châssis (en noir) et non visibles sur cette photo.

Algorithme

Pour que la voiture se dirige vers l'espace le plus dégagé, nous étudions la plage angulaire [-80,80] degrés avec 0 degré correspondant à l'avant du robot. Dans la liste des distance fournie par le lidar on recherche l'angle de la distance maximale et on oriente les roues dans cette direction.

Pour la vitesse, on étudie à présent la plage angulaire [-15,15] degrés. Si la plus petite distance de cette liste est inférieure à 800 mm la voiture se stop, c'est l'arrêt d'urgence. Sinon la vitesse du moteur suit une rampe de vitesse jusqu'à la distance maximale détectable par le lidar : 15000 mm.

Vue d'ensemble schématique du programme embarqué :



La détection par lidar

Le lidar est un télémètre laser rotatif qui permet de faire une imagerie en deux dimensions des alentours du robot. Sa plage de détection est comprise entre 100 et 15 000 mm. Le lidar que nous utilisons nous permet de former une liste de 360 (de 0 à 359 en C) valeurs correspondant à la distance pour un angle en degré équivalent à l'indice dans la liste. Cette liste est cependant complétée dans un ogre assez aléatoire par le lidar. En effet l'ajout des valeurs ne se fait pas pour tous les angles en un tour. On attend donc que le lidar ait fait 6 tours pour traiter les données dans la suite du code pour avoir suffisamment de valeurs dans la liste. Cette valeur a été choisie par empirisme pour faire un compromis entre le temps d'acquisition et un nombre de distances suffisant dans les plages étudiées dans la suite.

```
// Boucle infinie
while(true)
{
   if(trame_ok){
      debug_tour = !debug_tour;
   }
   //print_int("tour", tour_ok);
   if(tour_ok == 6){       //Réalise 6 tours
      pc.putc('a');
      tour_ok = 0;         //Remise du compte des tours à 0
```

L'asservissement de la direction

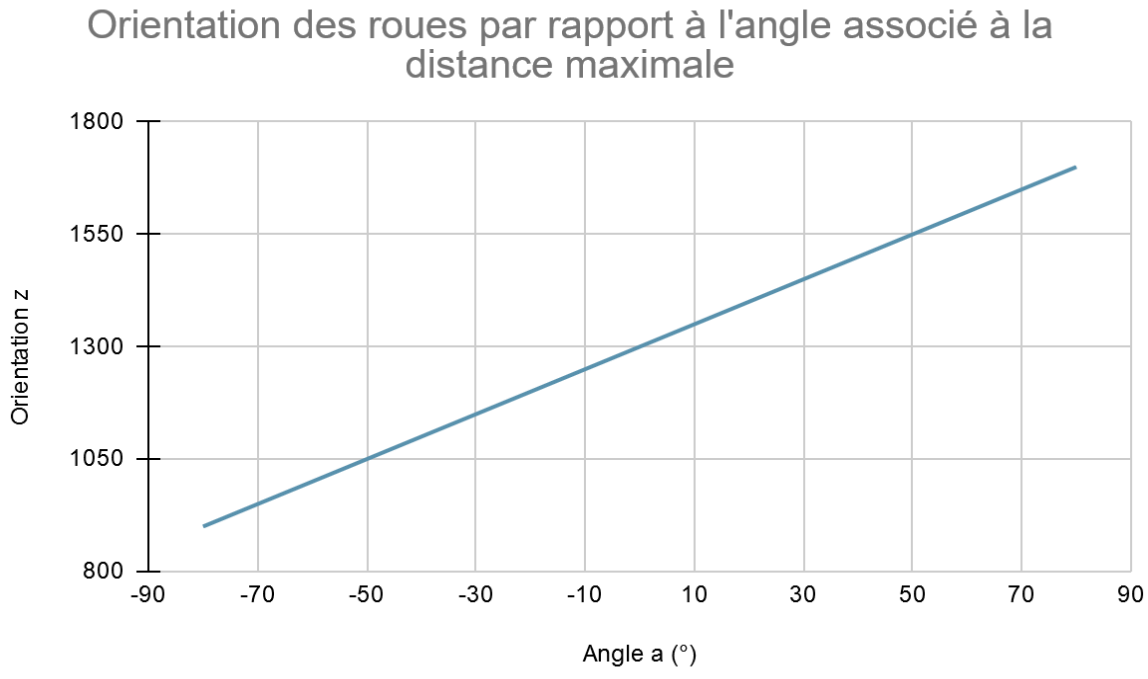
Pour asservir la direction de notre voiture, on récupère la liste des distances formée par 6 tours de lidar. On calcule l'angle correspondant à la distance la plus grande dans la plage angulaire [-80,80] degrés. Pour cela on compare les mesures de part et d'autre du 0 pour trouver le maximum. L'angle de cette distance maximale est stocké dans la variable a.

```
int a=0;
  // angle de la distance max entre [-80,80]°

for(i=0;i<80;i++)
{
  if(distance_scan_old[i] > distance_scan_old[a])
  {
     a = i;
  }
 if(distance_scan_old[359-i] >  distance_scan_old[a])
 {
    a = 359-i;
 }
}
```

```
//Servomoteur
 if (a>90)
 {
   a=a-360; // Ramène a dans la plage [-80;80]°
 }

//Orientation des roues en fonction de la direction de a
z=5*a+centre;
print_int("z", z);
servo_mot.pulsewidth_us(z);
```

Cette variable est ensuite transposée de la plage [280,360] U [0,80] à la plage [-80,80] pour être exploitée. On définit enfin une variable z qui correspond au temps de pulsation en microseconde qui sera transmis au servomoteur.

On remarque que nous l'avons codé par rapport à une variable centre car en fonction du montage physique du servomoteur sur la voiture, la pulsation correspondant aux roues droites peut varier. La multiplication par 5 permet quant à elle à augmenter l'influence de l'angle sur la direction des roues. Ainsi notre voiture tournera fortement à droite si a est proche de 80 degrés par exemple. On obtient ainsi la courbe ci contre.



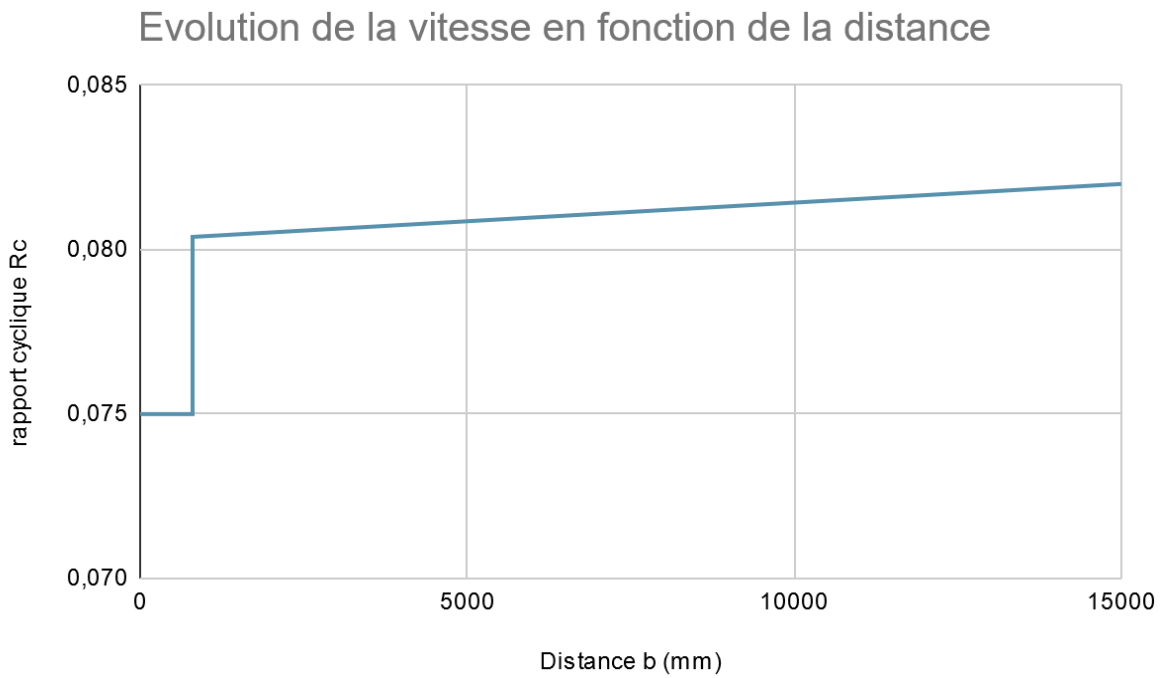
L'asservissement de la vitesse

Pour asservir la vitesse de notre voiture, on étudie maintenant la plage angulaire [-15,15] degrés comme indiqué précédemment. On récupère cependant cette fois la distance et non plus l'angle correspondant à une distance. La structure du code est très similaire à la partie précédente.

La variable b correspond donc après cette boucle à la plus petite distance différente de 0 dans la fenêtre d'angle étudiée.

Notre but étant d'obtenir une courbe du rapport cyclique du moteur en fonction de la distance b tel que ci dessous. On observe un palier correspondant à l'arrêt total du moteur quand b < 800 et puis une courbe affine croissante du rapport cyclique pour 800 < b < 15 000. La valeur maximale de cette pente a été définie par empirisme pour que la voiture n'ait pas trop d'inertie et puisse s'arrêter rapidement si besoin. La partie gauche de la pente a été fixée pour de telle sorte que la voiture puisse avancer en sécurité si des objets ne sont pas très éloignés (situation du passage d'une porte par exemple).

```
// Distance max entre [-15,15]°
b=15000;
for(i=0;i<15;i++)
{
   if((distance_scan_old[i] < b) && (distance_scan_old[i] > 0))
   {
      ind = i;
      b = distance_scan_old[i];
   }
   if((distance_scan_old[359-i] < b) && (distance_scan_old[359-i] > 0))
   {
      ind = 359-i;
      b = distance_scan_old[359-i];
   }
}

//Modulation de la vitesse
if (b<800)

{
    moteur_cc.write(0.075);      //Arrêt d'urgence
}
else
{
   rc = 0.0803 + (0.0017/14200)*b;  //Modulation de la vitesse en f° de la distance max
   moteur_cc.write(rc);
}
}
```



Conclusion

Notre projet est une réussite car nous avons rempli les principales exigences du cahier des charges. En effet, notre voiture autonome est capable de se déplacer dans les couloirs de l'école sans se prendre de murs en allant vers les espaces les plus dégagés et en passant les portes lorsque cela est nécessaire ! Cependant nous n'avons pas rempli toutes les exigences chiffrées du cahier des charges. En effet la vitesse ainsi que les angles remplis dans le cahier des charges ne reposant sur aucune mesure ils se sont avérés être des critères peu pertinents lors de la mise en pratique.

Pour la gestion globale du projet et la répartition des tâches nous avons utilisé Notion. Cela nous a également servi pour les échanges de fichiers et la préparation des livrables.

Annexe

Code en C embarqué final chargé dans la carte Nucléo du prototype :

```
//Fichiers entête
#include "mbed.h"
#include "platform/mbed_thread.h"
#include "rplidar.h"

// Déclaration des constantes
#define BLINKING_RATE_MS 500
#define NB_DATA_MAX 20
#define AFF_DATA 0

//Variables Lidar
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;
int maxDistance, maxAngle;
int a,b;
int i,j;
int z;
int centre=1300;

//variable moteur
double rc;

//Entrées et sorties lidar
Serial pc(USBTX, USBRX, 115200);
DigitalIn bp(USER_BUTTON);
DigitalOut debug_data(D10);
DigitalOut debug_tour(D9);
DigitalOut debug_out(D7);
DigitalOut data_ok(D5);
DigitalOut data_ok_q(D4);
Serial lidar(A0, A1, 115200);
PwmOut rotation(PB_9);

//Sortie servo
PwmOut servo_mot(PB_13);

//Sortie moteur
```

```
PwmOut moteur_cc(PC_9);

//Lidar
struct lidar_data ld_current;

int main()
{
int nb_tour = 0;
//Initialisations servo
// int time = 4000;
servo_mot.period_ms(20); // Initialisation période
servo_mot.pulsewidth_us(centre); // Initialisation en position 0

//Initialisations moteur
moteur_cc.period_ms(20); // Initialisation période
moteur_cc.write(0.075); // Initialisation en position 0

// Vérifications lidar
wait_s(3.0);
sendResetReq();
wait_s(1.0);
rotation.period(1/25000.0);
rotation.write(0.4);
wait_s(2.0);
pc.printf("\r\nLIDAR Testing\r\n");
lidar.attach(&IT_lidar);
wait_s(1.0);
pc.printf("\r\nLIDAR OK\r\n");
getHealthLidar(); //Retourne good si le lidar en bon focntionnement, bad sinon
getInfoLidar();
getSampleRate(); //Infos sur la vitesse d'acquisition

// Start a new scan
startScan();

// Boucle infinie
while(true)
        {
        if(trame_ok)
                {
                debug_tour = !debug_tour;
                }
        //print_int("tour", tour_ok);
        if(tour_ok == 6) //Réalise 6 tours
                {
                pc.putc('a');
                tour_ok = 0; //Remise du compte des tours à 0
                //findMax(distance_scan_old, 0, 360, &maxDistance, &maxAngle); //Détermine la distance max et
                angle associé
                int a=0;
```

```
// angle de la distance max entre [-80,80]°
for(i=0;i<80;i++)
        {
        if(distance_scan_old[i] > distance_scan_old[a])
                {
                a = i;
                }
        if(distance_scan_old[359-i] > distance_scan_old[a])
                {
                a = 359-i;
                }
        }
int ind = 60;

// Distance max entre [-15,15]°
b=15000;
for(i=0;i<15;i++)
        {
        if((distance_scan_old[i] < b) && (distance_scan_old[i] > 0))
                {
                ind = i;
                b = distance_scan_old[i];
                }
        if((distance_scan_old[359-i] < b) && (distance_scan_old[359-i] > 0))
                {
                ind = 359-i;
                b = distance_scan_old[359-i];
                }
        }
print_int("D1", b); //Retourne la distance la distance max entre [-15,15]°
print_int("A1", ind); //Retourne l'angle de la distance max entre [-15,15]°
print_int("a", a); //Retourne l'angle de la distance max entre [-80,80]°

//Servomoteur
if (a>90)
        {
        a=a-360; // Ramène a dans la plage [-80;80]°
        }

//Orientation des roues en fonction de la direction de a
z=5*a+centre;
print_int("z", z);
servo_mot.pulsewidth_us(z);

//Modulation de la vitesse
if (b<800)
        {
        moteur_cc.write(0.075); //Arrêt d'urgence
        }
Else
        {
```

```
                    rc = 0.0803 + (0.0017/14200)*b; //Modulation de la vitesse en f° de la distance max
                    moteur_cc.write(rc);
                    }
                }

            }
        }
```