Yassine Bakkouch, Nassreddine Amiche, Christophe Ye, Tony Ren, Antoine Bourhis

La Voiture Autonome

L'Homo Erectus connu des moments forts lors de son évolution : la découverte du feu (et de la bière), ses premiers pas sur la lune (et sur les plages de Copacabana), sans oublier la fameuse voiture Lexan Lancia Delta Integrale autonome. En effet, nul ne peut admettre de ne jamais avoir entendu cette célèbre phrase énoncée par ses créateurs géniaux, à jamais gravée dans la mémoire de tous : c'est une petite voiture pour l'homme mais elle fooooonce. Afin de se démarquer de ses concurrents, tels que Tesla, BMW, Porsche ou encore la Fiat multiplat, elle dût faire preuve de créativité et d'ingéniosité en mélangeant la fameuse carte Nucléo, qu'il n'est plus nécessaire de présenter, ainsi que de capteurs hautes performances : les capteurs de distance Sharp que l'on peut retrouver à 80cents sur Alibaba mais aussi le fameux capteur Lidar qui coûte le prix d'une année de repas au Crous. Une fois les composants trouvés, il ne restait à l'équipe plus qu'à étudier le Moteur type 540, variateur électronique TEU-105BK du véhicule. Enfin ces programmeurs de grands talents n'avaient plus que l'honneur de raccorder les différentes pièces du puzzle afin de... libérer la bête.

1 Comprendre le projet

1.1 Problématique et cahier des charges du projet

L'objectif du projet est de faire fonctionner une voiture miniaturisée, roulant de manière autonome à partir du matériel à disposition en salle de TP.

FIGURE 1 – Voiture utilisée

1.2 Découpage fonctionnel

Afin de caractériser les valeurs de cycle de fonctionnement du moteur directionnel et du moteur vitesse, nous avons opté pour l'utilisation d'un GBF et d'une tension d'alimentation. En faisant varier le cycle des impulsions, on contrôle les deux moteurs. Dès lors, on conservait la valeur de fonctionnement des moteurs pour ensuite les implémenter dans le compiler Mbed qui permet de transférer ensuite le programme vers la carte Nucléo. Finalement il suffit de faire fonctionner la carte de façon autonome en l'alimentant avec la batterie de la voiture pour arriver à nos fins. Le cahier des charges que l'on s'est fixé est le suivant :

1. Voiture qui roule de façon autonome.
2. Détection d'obstacles.
3. Esquiver de manière autonome les objets sur sa route.
4. Arrêt de la voiture lorsque l'obstacle est inévitable.

FIGURE 2 – Roule petite voiture !

2 Réaliser le prototype

2.1 Réalisations du programme

2.1.1 Voiture à 3 capteurs

L'idée de base du code est très simple, la voiture se dirige vers la partie « libre » :

1. D'abord les capteurs, détectent les objets aux alentours et renvoient l'information à la carte.
2. La carte traite l'information, la fonction « ChercheSolution » dirige la voiture vers la zone où la distance est maximale. Si celle ci dépasse un seuil sinon la voiture s'arrête et on appelle ça situation "infranchissable".

Pour éviter le dérapage, on fait tourner la voiture en continue grâce aux fonctions « tournerdroit » et « tournergauche » et on initialise des compteurs au début qui permettent d'éviter d'appliquer ces fonctions plus d'une fois car celle-ci mettent la voiture en position droite et la tourne en continue. BONUS : On a écrit 2 fonctions qui permettent de contourner la situation infranchissable elle a été plus dur à coder que les autres fonctions (Voir explication sur le schéma explicatif de l'algorithme).

FIGURE 3 – Roule petite voiture !

2.1.2 Voiture à capteur LIDAR

Pareil que pour les 3 capteurs l'idée reste la même, on veut diriger la voiture vers la zone la plus sûre, mais comme le lidar a la possibilité de scanner à 360 degrés, on le limite à seulement $-20°$ à $+20°$, qui sont aussi les angles maximum dont peut tourner les roues de la voiture. L'algorithme reste le même on demande à la voiture de se diriger vers la zone sans risque seulement si la distance maximale détectée dépasse un seuil sinon elle s'arrête. Aussi pour éviter les zigzags, on initialise un compteur c qui prendra la valeur de l'angle dans la distance est maximale et si la prochaine valeur reste aux alentours de c ce qui veut dire dans l'intervalle [c-1,c+1], la voiture ne change pas de direction.

2.2 Algorithme

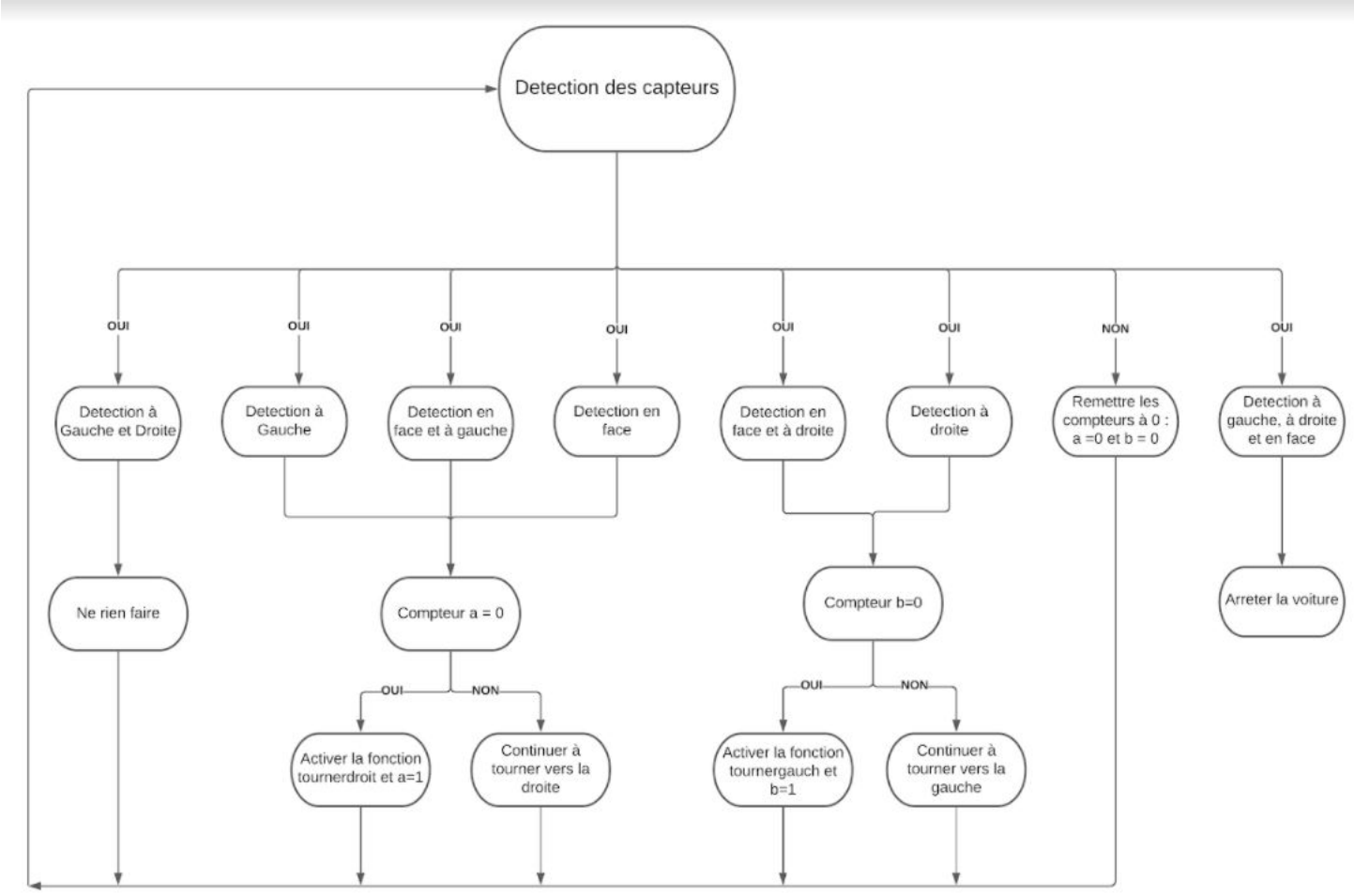

FIGURE 4 – Algorithme à 3 capteurs

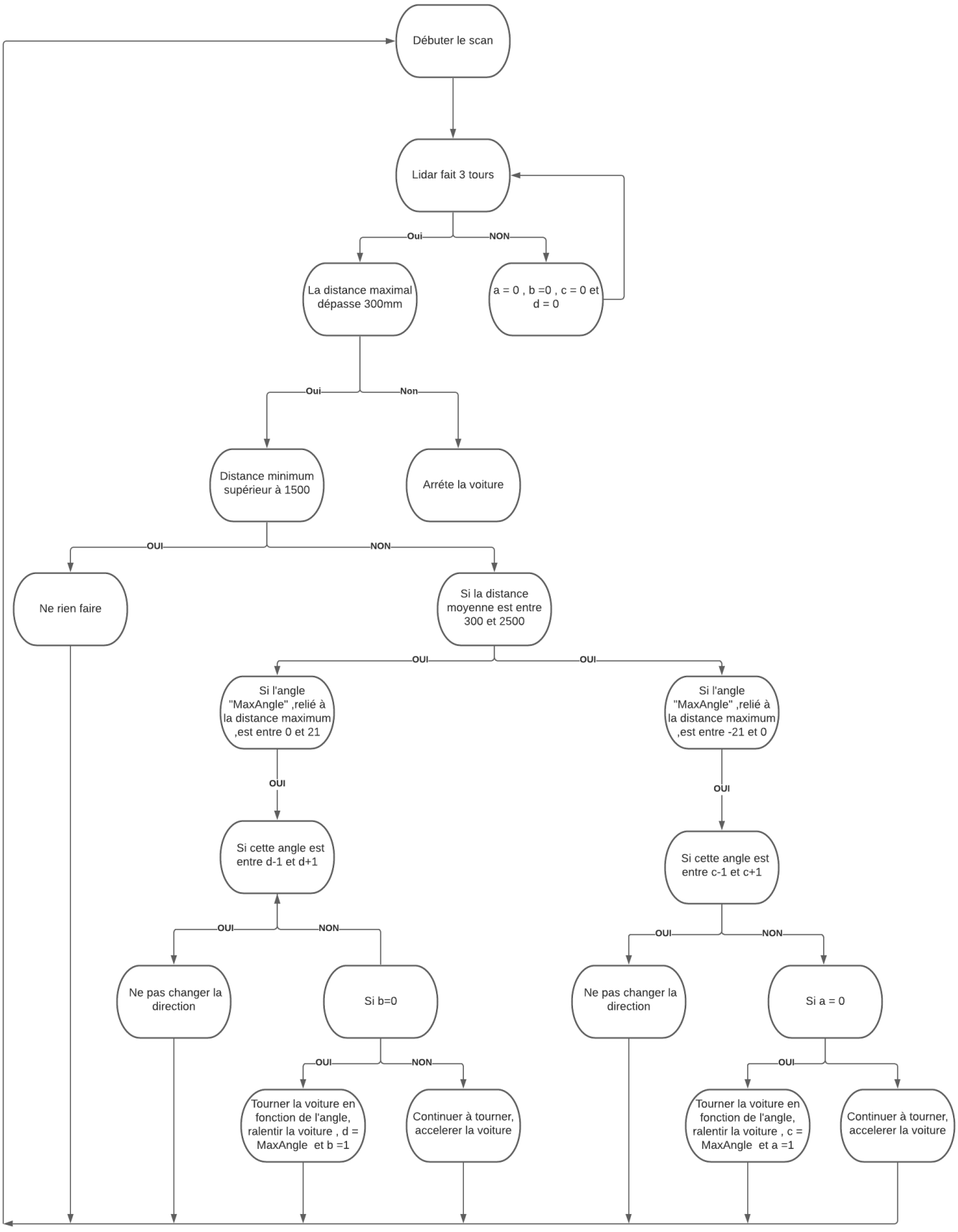

FIGURE 5 – Algorithme avec le capteur LIDAR

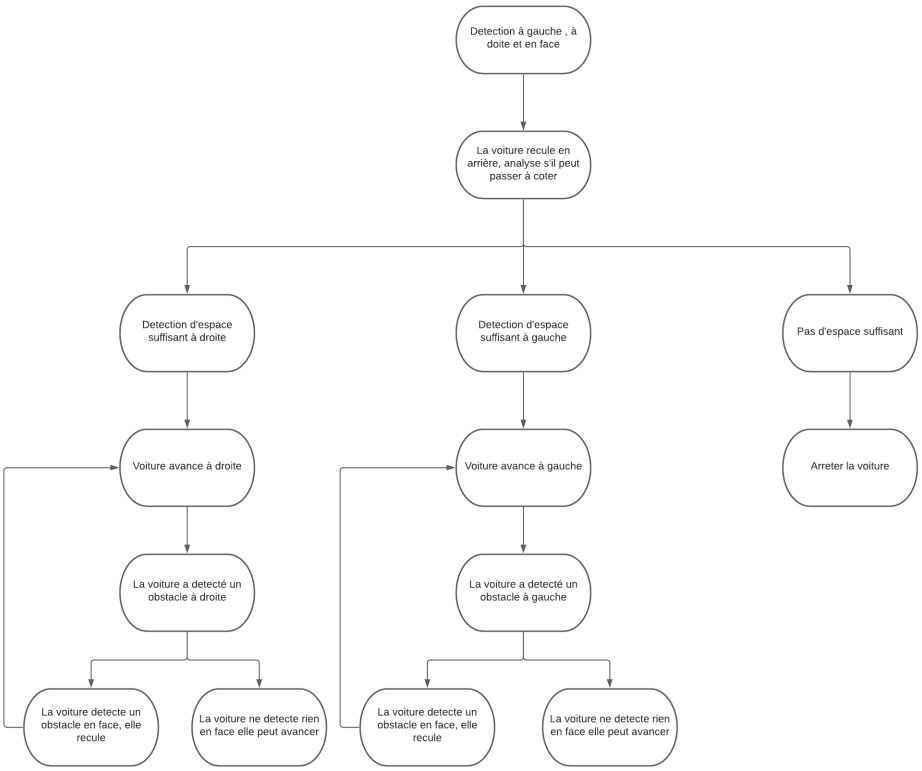

FIGURE 6 – Algorithme Bonus

Suite à une réflexion sur l'incapacité de la voiture à faire face à un obstacle dit "inévitable", le dernier algorithme est proposé afin de permettre à la voiture de trouver une solution. On imagine que la voiture fait marche arrière et choisit une direction pour tourner. Ainsi par tâtonnements, celle-ci pourrait parvenir à sortir d'une situation où elle serait bloquée.

3 Valider et caractériser le système final

3.1 Tests et validation

3.1.1 Voiture équipé d'un capteur

Cette première version de la voiture avec un capteur va nous servir à programmer l'arrêt de la voiture. C'est le premier test à réaliser pour éviter les collisions directes à pleine vitesse. Ce qu'on cherche à réaliser : la voiture avance en ligne droite et dès qu'elle risque de heurter un obstacle, elle doit s'arrêter. On commence par des simples tests sur la paillasse en utilisant notre main pour vérifier si la voiture s'arrête bien. Une fois le test sur le bureau validé, il faut vérifier si elle est bien capable de s'arrêter avant la collision. On a donc réalisé des tests au sol pour choisir quand la voiture doit commencer à ralentir. Ce premier test validé, on peut passer à la programmation sur l'évitement d'obstacle.

3.1.2 Voiture équipé de 3 capteurs

On souhaite un pilotage automatique de la voiture donc en plus de pouvoir s'arrêter en cas de collision, elle doit pouvoir l'éviter quand elle peut éviter l'arrêt. On choisit la procédure la plus simple, à savoir : la voiture doit tourner à droite (resp. à gauche) lorsque que l'obstacle se trouve à gauche (resp. à droite) Comme dans la première partie, on réalise des tests simples sur le bureau en mettant la main devant la voiture pour vérifier qu'elle tourne bien du côté souhaité. Une fois le test sur le bureau validé, il faut vérifier si elle est bien capable d'éviter les obstacles en situation réelle sur le terrain. Au sol, il arrivait parfois que la voiture n'ait pas le temps de tourner à temps, donc il a donc fallu adapter le programme de façon à ce que la voiture ralentit à l'avance lorsqu'elle décide de contourner un obstacle. Les questions auxquelles on a dû s'interroger,

de combien doit-elle ralentir, à quelle distance de l'obstacle doit-elle commencer à ralentir, quand doit-elle commencer à tourner pour éviter l'impact ? Il aura fallu de nombreux tests avant de trouver des paramètres optimaux. Il arrive toutefois qu'on se retrouve face à des obstacles dits "infranchissables" typiquement lorsque l'obstacle est une impasse. Dans ce cas là, la voiture n'a d'autres choix que de s'arrêter. Globalement, on a réussi à obtenir une voiture à peu près autonome avec ce système, elle évite bien les obstacles, elle s'arrête bien lorsque la collision est inévitable. Cependant, la précision des capteurs utilisés ne renvoyait pas assez d'informations à la voiture donc il arrivait que la voiture ne détecte pas certains obstacles.

3.1.3 Voiture équipé du LIDAR

FIGURE 7 – Capteur LIDAR

Le problème majeur avec le système des 3 capteurs est dû à un manque d'information reçu par la voiture de la part des capteurs. Avec le LIDAR, on n'aura pas à se soucier de ce problème, la voiture est donc censée mieux éviter les obstacles. On réalise des tests au sol qui sont assez concluants ; la voiture évite bien tous les obstacles comme souhaité. Le LIDAR permet en plus d'éviter de se retrouver dans la situation de l'obstacle infranchissable car la voiture reçoit assez d'informations pour se diriger toujours vers la direction de la zone où il y a le plus d'espace. Ainsi, elle ne s'approche de aucun danger.

Finalement, avec ce dernier système on obtient une voiture qui répond bien au cahier des charges. La voiture est capable d'assumer une conduite en autonomie, c'est-à-dire qu'elle peut rouler continuellement en évitant tous les obstacles sur son chemin. Son seul petit défaut est qu'elle tangue un peu causé par le fait qu'elle cherche constamment à se diriger vers la zone de non danger.

4 Comprendre les étapes de réalisation/choix

4.1 Difficultés rencontrées / Analyse du travail d'équipe

Difficultés rencontrées :

1. Sabotage : Montages défaits, voitures différentes, composants disparus. Soudure, nouvelle caractérisation de la voiture etc.. (Séances 3 et 4)
2. Limite des capteurs : pas assez de retour d'informations sur ce qui se passe autour de la voiture qui induit une conduite autonome imparfaite
3. Les valeurs des caractéristiques de la voiture changeaient tout le temps. Le fonctionnement du moteur vitesse et directionnel était inconstant.

Nous avons tous été ravis de travailler en équipe sur un projet aussi long qui était une première pour chacun des membres du groupe. On a donc dû développer nos compétences relationnelles afin de s'organiser et répartir les tâche. La gestion du temps (sur une aussi longue période) à du être maitrisée. Enfin la communication et l'écoute était une partie essentielle pour la bonne réalisation du projet.

4.2 Planning

	GROUPE 1	GROUPE 2
Séance N°1	Caractérisation des moteurs Détermination de la vitesse optimale pour éviter les accidents du temps et de l'angle nécessaires pour faire tourner la voiture.	Caractérisation des capteurs Détermination des angles optimaux de détection des objets.
Séance N°2	Idem que séance 1	Idem que séance 1
Séance N°3	Idem que séance 1 + 1ier essai	Idem que séance 1 + 1ier essai
Séance 4	Programmation de l'arrêt de la voiture.	Récupération de l'ancien code et adaptation du programme suite aux essais de la séance n°3.
Séance 5	Compréhension du capteur Lidar et des valeurs caractéristiques d'angles et des distances	Adaptation du code pour le capteur Lidar
Séance n°6	FILM + optimisation de la voiture	FILM + Poster
Séance N°7	FILM + réglage des problèmes rencontrés à la séance 6	FILM + réglage des problèmes rencontrés à la séance 6
Séance n°8	Présentation	Présentation

FIGURE 8 – Planning des séances

5 Conclusion

On a tous retenu que le projet était très intéressant et le travail en équipe était bénéfique pour à la fois apprendre à travailler en équipe mais aussi apprendre à se connaître au vu du contexte actuel. La soudure, la vitesse de la voiture et les premiers échecs était plutôt drôle à observer dans un premier temps. Cependant, les échecs répétés, le sabotage, les problèmes moteur ont été plusieurs coups dur portés à notre motivation.

Néanmoins le projet est arrivé à son bon terme. Le cahier des charges a été remplis et la voiture fonctionnait correctement.La vidéo est explicite : Cliquez ici pour voir la vidéo

6 annexe

6.1 LIDAR

Lidar qui signifie « Light Detection and Ranging » fait partie de la famille des capteurs. Celui-ci permet de déterminer la distance qui le sépare d'un objet en émettant un faisceau laser dans sa direction.

Les caractéristiques principales du lidar sont :

- La longueur d'onde : utilisation des longueurs d'ondes suivantes,

(a) Infrarouge (1500 -2000 nm)

(b) Dans le proche infrarouge (850 -940 nm)

(c) Bleu-rouge (500 -750 nm)

(d) Ultraviolet (250 nm)

- La distance :

  La portée du Lidar peut varier entre 0.01m jusqu'à 200m. Cela dépend de l'endroit ou on en fait l'usage si on se trouve à l'intérieur d'un bâtiment celui-ci n'aura aucun problème à viser loin mais si on se trouve à l'extérieur, la présence de lumière du soleil lui pose problème pour analyser le monde qui l'entoure.

- Le balayage :

  Le lidar effectue un balayage de 360° , à une vitesse de rotation variant entre 1 HZ et 100 Hz, et cela afin de récolter toute les informations nécessaires et les stocker dans un tableau de 360 éléments. La quantité d'information, ce qui veut dire le nombre de points non nul dans le tableau dépend du nombre de tours qu'on lui demande d'effectuer avant de récolter l'information, plus le nombre de tours est élevé, plus la quantité d'information le sera et plus le temps d'assimilation le sera aussi.

- Les erreurs :

  Des erreurs de mesure peuvent se glisser dans les données, il y a les erreurs systématiques qui sont inévitables mais qui peuvent être minimisées et en raison de paramètres environnementaux et physiques (réfraction, diffraction, etc.), des erreurs supplémentaires peuvent également se produire. Lorsque exactement la même mesure effectuée par Lidar montre des valeurs différentes, des erreurs aléatoires se produisent

6.2 La voiture

La voiture se décompose en deux parties majeures :

- Le Moteur directionnel

  Le fonctionnement du moteur direction se base sur la longueur des pulsations qu'il reçoit. Par exemple si celui-ci reçoit des pulsations de durée 1.5ms, la voiture se mettra automatiquement dans la direction face à elle, ce qui veut dire avec un angle de 0°. Et si cette durée était de 1.0ms elle tournera vers la gauche tandis que pour 2.0ms elle tournera vers la droite.

  En première séance, on a dû délimiter les angles de rotation des roues, puisque la voiture ne nous permet pas de tourner de $-90°$ à 90°. On a alors découvert que les angles maximums étaient plutôt $-20°$ à 20°.

- Le Moteur vitesse

  Son fonctionnement est similaire à celui du moteur direction, sauf qu'ici la durée de 1.5 ms des impulsion permet d'arrêter la voiture. Si celle-ci est de 2.0 ms la voiture est accélérée et si la durée est de 1.0 ms elle recule en arrièr. On a dû aussi déterminer les valeurs adéquate au bon fonctionnement de la voiture. Expérimentalement, on a trouvé que 1.68 ms correspond à une vitesse maximum et 1.6 ms pour une vitesse minimum. Certains moteurs vitesse en TP ont aussi une autre caractéristique. La première valeur de durée d'impulsion qu'ils reçoivent dévient automatiquement celle qui arrête la voiture, il faut alors préalablement envoyer un signal d'impulsion de durée 1.5 ms afin de bloquer celle-ci comme valeur d'arrêt.

6.3 Code 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
#define VIT_MIN                 1600
#define VIT_MAX                 1620

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;

Serial              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 vitess(PC_9);
PwmOut  ma_sortie_pwm(PB_13);

Serial              lidar(A0, A1, 115200);
PwmOut              rotation(D14);
double Distance ;
struct lidar_data   ld_current;


/** MAIN FUNCTION
 */
int main()
{
    vitess.period_ms(20);
    vitess.pulsewidth_us(1500);
    ma_sortie_pwm.period_ms(20);
    ma_sortie_pwm.pulsewidth_us(1.27*1000);
    wait(3.0);
vitess.pulsewidth_us( VIT_MAX);
ma_sortie_pwm.pulsewidth_us(1.27*1000);
```

```
52 ma_sortie_pwm.pulsewidth_us(1.27*1000);
53     int nb_tour = 0;
54     wait_s(3.0);
55     rotation.period(1/25000.0);
56     rotation.write(0.6);
57     wait_s(2.0);
58     pc.printf("\r\nLIDAR Testing\r\n");
59     lidar.attach(&IT_lidar);
60     wait_s(1.0);
61     pc.printf("\r\nLIDAR OK\r\n");
62 
63     getHealthLidar();
64     getInfoLidar();
65     getSampleRate();
66     // Start a new scan
67     startScan();
68     // Infinite Loop
69     int list[42];
70     int i;
71     int a=0;
72    int b=0;
73    int c=0;
74    int d=0;
75 
76   /*
77     while (true) {
79         if(tour_ok == 4){
80             int maxDistance, maxAngle;
81             int minDistance, minAngle;
82                         findMax(list, 42, &maxDistance, &maxAngle);
83             findMin(list, 42, &minDistance, &minAngle);
84             print_int("A", maxAngle);
85             print_int("D", maxDistance);
86             print_int("Am", minAngle);
87             print_int("Dm", minDistance);
88         }
89 
90     }
91     */
92     while(1)
93     {
94             for (i=0;i<42;i++)
95      {
96          if (i<21)
97          {
98             list[i]=distance_scan_old[i];
99          }
100         else
101         {
102            list[i]=distance_scan_old[i+319];
103         }
104     }
```

```
105
106
107         if(tour_ok == 3){
108             int maxDistance, maxAngle;
109             int minDistance, minAngle;
110             int moyenne = 0;
111             int nb = 0;
112             for (i=0;i<42;i++)
113             {
114                 if(list[i] != 0){
115                     moyenne = moyenne + list[i];
116                     nb++;
117                 }
118             }
119             moyenne = moyenne/(double)nb;
120             tour_ok = 0;
121
122
123             findMax(list, 42, &maxDistance, &maxAngle);
124             findMin(list, 42, &minDistance, &minAngle);
125             print_int("A", maxAngle);
126             print_int("D", maxDistance);
127             print_int("Am", minAngle);
128             print_int("Dm", minDistance);
```

```
131     if (300 > maxDistance)
132     {
133     vitess.pulsewidth_us(1500);
134
135             pc.printf("STOP\r\n");
136     }
137     else{
138         pc.printf("M=%d\r\n", moyenne);
139         if (((300 < moyenne) && (moyenne < 2500)) || (300 > maxDistance))
140         {
141             pc.printf("D1\r\n");
142             if ((300 < moyenne) && (moyenne < 2500))
143             {
144                 pc.printf("D2\r\n");
145                 if ((0 < maxAngle) && (maxAngle < 21))
146                 {
147                     pc.printf("D3\r\n");
148                     if ((d-1 <= maxAngle) && (maxAngle <= d+1))
149                     {
150
151                     pc.printf("D4\r\n");
152                         if (a=0)
153                         {
154                         d = maxAngle;
155                         ma_sortie_pwm.pulsewidth_us(1320+d*20);
156                         vitess.pulsewidth_us(VIT_MIN);
```

```
                            a=a+1;
                            }

                            else {
                               ma_sortie_pwm.pulsewidth_us(1320+d*20);
                            vitess.pulsewidth_us(VIT_MAX);
                           }
                        }
                        else
                        {

                        pc.printf("D5\r\n");
                        if (a=0)
                            {
                            d = maxAngle;
                            ma_sortie_pwm.pulsewidth_us(1320+maxAngle*20);
                            vitess.pulsewidth_us(VIT_MIN);
                            a=a+1;
                            }

                        else {
                           ma_sortie_pwm.pulsewidth_us(1320+maxAngle*20);
                        vitess.pulsewidth_us(VIT_MAX);
                       }
                    }
                }
                    if ((21 < maxAngle) && (maxAngle < 41))
                   {   pc.printf("D6\r\n");
                      if ((c-1 <=maxAngle) && (maxAngle <= c+1))
                      {
                           if (b=0)
                       {
                       c = maxAngle;
                       ma_sortie_pwm.pulsewidth_us(1320-c*20);
                       vitess.pulsewidth_us(VIT_MIN);
                       b=b+1;
                       }

                       else {
                          ma_sortie_pwm.pulsewidth_us(1320-c*20);
                       vitess.pulsewidth_us(VIT_MAX);
                          }
                          }
                       else
                       {
                       if (b=0)
                       {
                       c = maxAngle ;
                       ma_sortie_pwm.pulsewidth_us(1320-maxAngle*20);
                       vitess.pulsewidth_us(VIT_MIN);
                       b=b+1;
                       }
```

```
                    }

                    else {
                       ma_sortie_pwm.pulsewidth_us(1320-maxAngle*20);
                    vitess.pulsewidth_us(VIT_MAX);
                       }
                }
                }
                }
                else
                {
                     a=0;
                     b=0;
                     c=0;
                     d=0;
                }
            }
        }
        /*
        stopScan();
        tour_ok = 0;
        print_int("NB ", data_scan_nb);
        // affichage données
        if(AFF_DATA){
            for(int k = 0; k < 360; k++){
                if(distance_scan_old[k] != 0){
                    sprintf(pc_debug_data, "\t%d = %d", k, distance_scan_old[k]);
                    pc.write(pc_debug_data, strlen(pc_debug_data));
                }
            }
        }
        getHealthLidar();
        wait_s(0.001);
        startScan();
        */
    }
}
}
```

FIGURE 9 – code Mbed