Raphaël LIVET Alexandre MERSCH Lucile Tilloy Yassir Amazouz

IETI

—

Rapport technique

Nous attestons que ce travail est original, que nous citons en référence toutes les sources utilisées et qu'il ne comporte pas de plagiat.

Table des matières

INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental

Un robot omnidirectionnel est un appareil capable de se déplacer vers un point donné, en prenant le chemin le plus court, tout en évitant les obstacles. Il est utilisé dans l'industrie pour déplacer des charges mais aussi chez des particuliers comme aspirateur ou tondeuse.

Pour ce projet, l'objectif était de pouvoir donner la position d'arrivée du robot à distance grâce à une interface Bluetooth et que le robot s'y rende en autonomie tout en évitant les obstacles sur son chemin à l'aide d'un capteur LIDAR. Le but était aussi d'optimiser la distance à parcourir.

Nous avons réussi à coupler les fonctions d'interfaçage Bluetooth et d'optimisation du trajet et d'un autre côté les fonctions d'évitement d'obstacle et d'optimisation du trajet mais nous n'avons pas eu le temps de coupler l'interface Bluetooth et l'évitement des obstacles. Les principes des deux programmes seront donc présentés dans ce rapport technique.

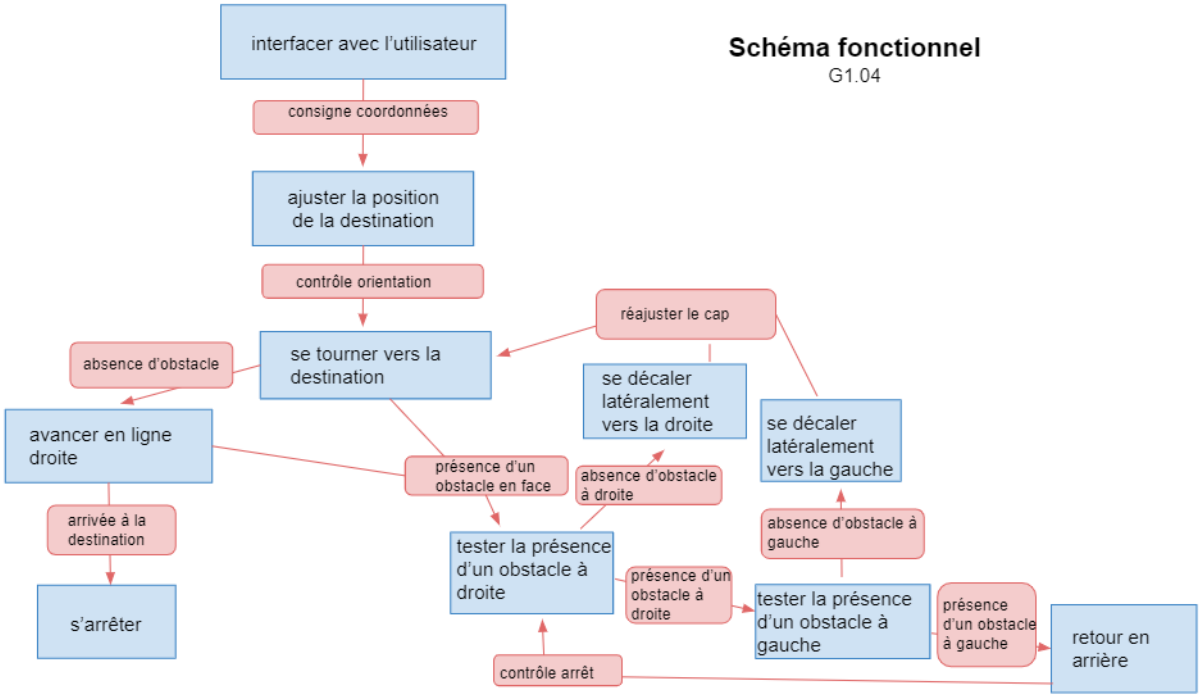


FIGURE 1 – Schéma fonctionnel complet du projet

1 Principe des deux programmes étudiés

1.1 Interfacage Bluetooth

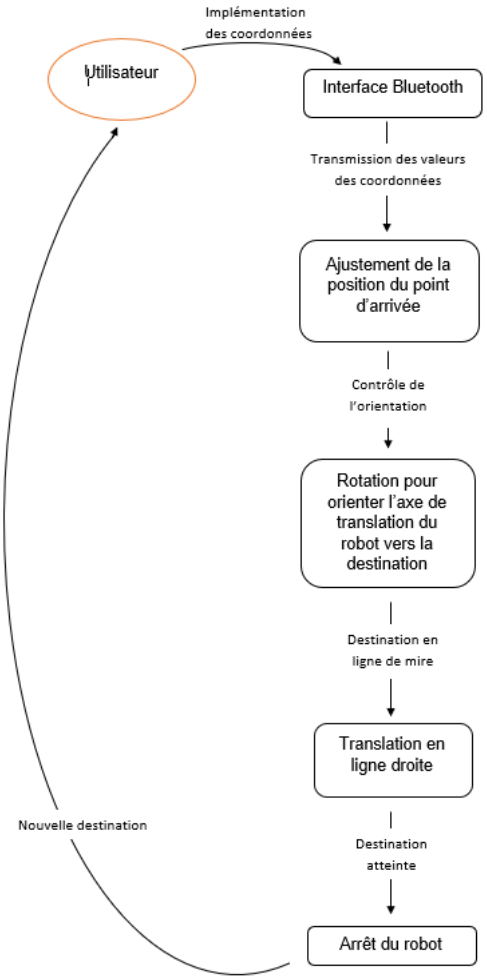


FIGURE 2 – Principe de l'interfacage bluetooth

1.2 Algorithme d'évitement d'obstacles

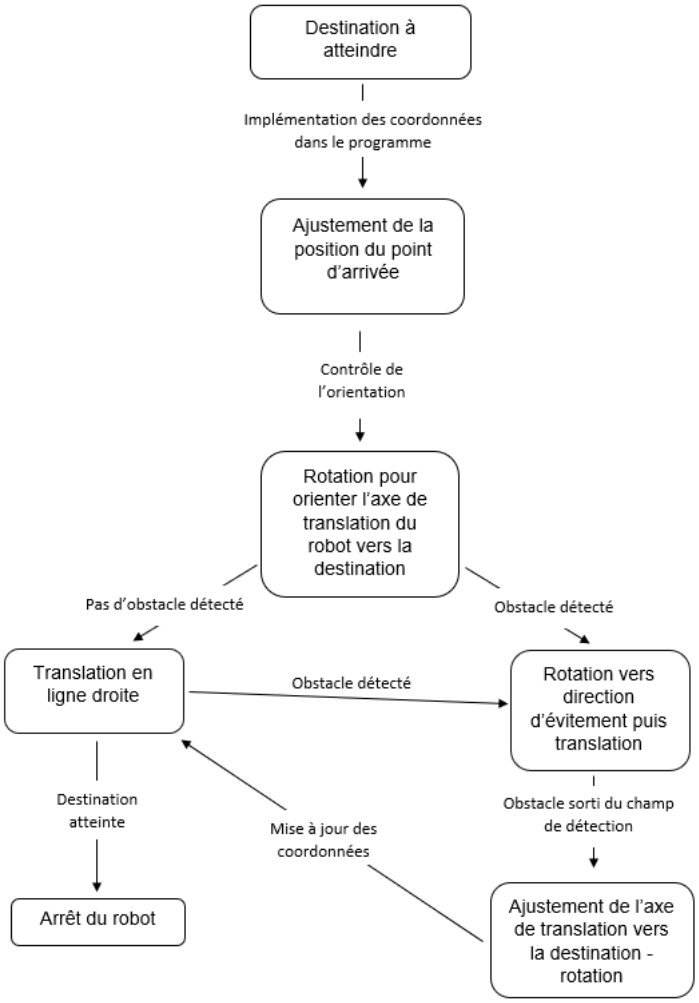


FIGURE 3 – Principe de l'évitement d'obstacle

2 Réalisation pratique des principes

2.1 Fonction d'optimisation de la distance à parcourir

Cette fonction prend en entrée les coordonnées x et y du point d'arrivée. On suppose que le robot est initialement au point de coordonnées (0 ;0) et qu'il est orienté selon un axe y défini de façon conventionnelle. La fonction calcule ensuite la distance minimale à la destination et l'angle dont doit tourner le robot pour aligner son axe de translation avec la direction de la destination. Le robot avance ensuite en ligne droite pendant que l'on décrémente la distance à parcourir jusqu'à ce qu'elle soit nulle : le robot est arrivé à destination.

Le code est en ANNEXE 1.

2.2 Asservissement de la direction

Nos premiers tests nous ont fait réaliser que les moteurs n'effectuent pas leur rotation à la même vitesse malgré des instructions similaires. Cette différence engendre une déviation de la voiture. Afin de pallier ce problème, nous avons utilisé les capteurs situés au niveau de chaque roue, afin de comptabiliser la rotation exacte de chaque roue. Le principe de la fonction est d'accélérer la roue la plus lente et ralentir la roue la plus rapide, dès que l'écart de rotation dépasse un certain seuil.

2.3 Interfacage Bluetooth : fonction et application

L'enjeu de cette partie est de simplifier l'utilisation du système. Jusque-là, notre système recevait les coordonnées de sa destination en même temps que le code lui permettant de fonctionner. Il était nécessaire de séparer l'envoie des coordonnées du reste du code. De plus, l'envoi des coordonnées se faisait à travers le câble USB relié à l'ordinateur. Ce qui n'était pas compatible avec notre volonté de faire une voiture autonome avec un système embarqué.

Ainsi nous avons créé un interface sur Smartphone par l'intermédiaire d'une application créée avec MITinventor2. Nous avons désigné l'application pour permettre la connexion entre le smartphone et le système, ainsi que l'entrée des coordonnées par l'utilisateur et leur envoie au système. La connexion s'est faite par Bluetooth avec un module RN42 connecté à notre système.

Après la mise en place du système d'envoie des coordonnées. Il était nécessaire de s'intéresser à la réception et la compréhension par notre système de ces données. Nous avons donc ajouté à notre code des instructions pour initialiser le module Bluetooth et permettre à notre système de déterminer que sont les coordonnées dans le signal qu'il reçoit.

Pour cela nous avons notamment modifié le signal envoyé. Pour imposer l'envoie de caractères particulier pour repérer début et fin des 2 coordonnées. Ainsi les instructions envoyées sont de la forme : a(coordonnées x)b(coordonnées y)z

Ce signal est enregistré sous forme de liste dans la mémoire du système. Nous avons ensuite pris les valeurs présentes entre les caractères comme valeurs de x et y.

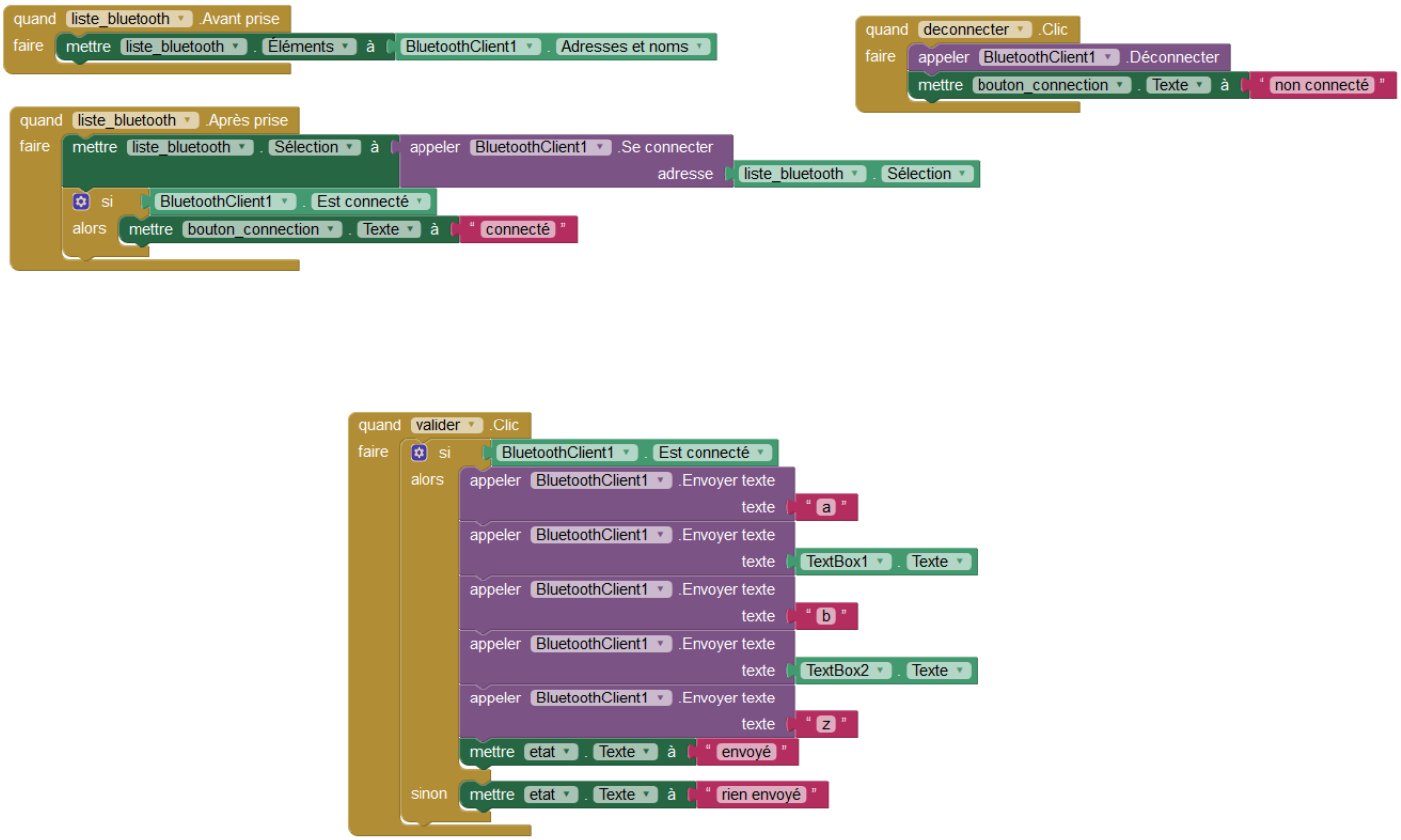


FIGURE 4 – Programme de l'application cellulaire Bluetooth

2.4 Fonction d'évitement des obstacles

Le bout de programme permettant d'aller jusqu'à une destination précédemment définie peut être séparer en deux parties, la première est une rotation et la seconde une translation jusqu'à la destination. On commence par supposer que notre robot ne rencontre pas d'obstacle pendant sa puisqu'il tourne sur lui même. Ainsi, on définit une condition de détection d'obstacle grâce au LIDAR qui est représenté par une zone circulaire de 20 centimètres de rayon autour du robot. Si, pendant la phase de translation, un obstacle est détecté dans cette zone alors le robot fait d'abord une manoeuvre d'évitement puis recalcule son itinéraire en fonction du trajet deja parcouru et de sa destination.

La manoeuvre d'évitement est telle que le robot tourne sur lui même d'un angle de $\frac{\pi}{3}$ dans le sens opposé à la direction de l'obstacle puis le robot avance jusqu'à que l'obstacle ne puisse plus être détecter dans la zone de détection définie autour de celui-ci. Une fois l'obstacle évité, la direction du robot est corrigé pour se diriger à nouveau vers sa destination. Malheureusement, par manque de temps, la correction de la trajectoire qui suit la manoeuvre d'évitement n'est pas opérationnelle. De plus, on suppose que le robot ne rencontre pas obstacles pendant la manoeuvre d'évitement d'un premier obstacle.

3 Tests de validation effectués

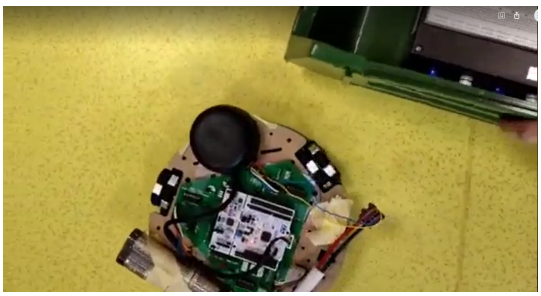
3.1 Avec l'interface Bluetooth

Pour tester ce programme, nous avons marqué au sol avec des morceaux de scotch différents points d'arrivée pour le robot dont on connaît les coordonnées. On rentre successivement dans l'interface créée les différentes positions des marquages, sachant que l'application ne garde en mémoire qu'un seul point : pendant que le robot va vers un point, on peut déjà rentrer les coordonnées du prochain point d'arrivé mais pas celles d'un troisième point.

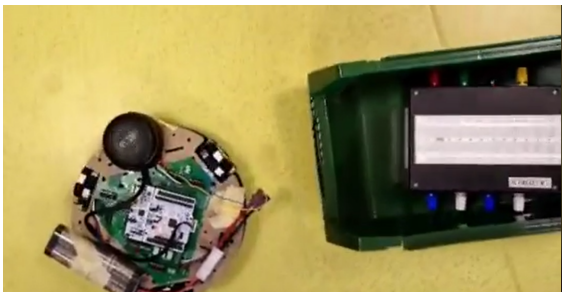
Une amélioration possible serait de pallier à l'écartement de la trajectoire : le robot n'avance pas parfaitement en ligne droite ce qui engendre des écarts avec la position d'arrivée exacte. Si on cumule les destinations successives sans correction, le robot finit par faire des écarts de plus en plus grands avec les positions d'arrivée souhaitées.

3.2 Avec l'évitement des obstacles

Pour tester ce programme, nous avons repéré au sol à l'aide d'un morceau de scotch la position d'arrivée voulue du robot et avons placé un obstacle assez gros sur le parcours. Nous avons ensuite placé le robot au point de départ que nous avons aussi repéré au préalable. Le robot repère alors l'obstacle et l'évite de la façon souhaitée. Il attend ensuite un court instant afin de pouvoir recalculer sa nouvelle trajectoire. Puis il s'oriente et poursuit jusqu'à atteindre la position finale.



(a) Pendant la manoeuvre d'évitement



(b) Après la redirection vers le point d'arrivé

FIGURE 5 – Images aériennes du robot DORA

4 Étapes suivies au cours du projet et difficultés

Etape 1 :

- Fonctionnement basique, translation selon x et y avec rotation sur lui même.
- Définition du système de coordonnées, notion de distance
- Trajet d'un point A à un point B en suivant les axes x et y

Etape 2 :

- Déterminer le chemin le plus court vers le point B
- Orientation vers la direction déterminer, notion d'angle

Etape 3.1 :

- Définition du champ de détection : distance minimale à un obstacle et angle du champ de détection
- Évitement de l'obstacle
- Recalcule de la direction la plus courte vers le point B

Etape 3.2 :

- Création d'une interface avec le système de coordonnées
- Transmettre les coordonnées en fonctionnement

Difficultés rencontrées :

- Déterminer la batterie adaptée au robot
- Réunir deux programmes qui fonctionnent séparemment en un seul
- Comprendre certaines des fonctions associées au LIDAR
- Coupler les fonctions du LIDAR et de notre robot

5 ANNEXES

5.1 Annexe 1

```
#include "mbed.h"
Serial pc(USBTX, USBRX); // configuration pour TeraTerm

InterruptIn encoche_moteur2(PB_2);
InterruptIn encoche_moteur3(PB_4);
DigitalOut led(LED2);

PwmOut moteur1_rot_gauche(PA_10);
PwmOut moteur1_rot_droite(PA_15);
PwmOut moteur2_rot_gauche(PB_14);
PwmOut moteur2_rot_droite(PB_13);
PwmOut moteur3_rot_gauche(PB_8);
PwmOut moteur3_rot_droite(PC_9);

DigitalOut  enablePower(PB_3);
int dist2;
int dist3;

int x = 20;
int y = 13;
int teta = atan(1.0*y/x)*(180/3.14);
int nbr_points=11765; //pour 10 cm
int nbr_points_rot360=99000;

int xy = sqrt(x*x+y*y);


void fonction_interruption_moteur2(void);
void fonction_interruption_moteur3(void);


int main() {
    enablePower = 0;
    pc.baud(115200); // les 115200 bauds initialisés dans TeraTerm
    encoche_moteur2.rise(&fonction_interruption_moteur2);
    encoche_moteur3.rise(&fonction_interruption_moteur3);

    double rc2=0.5;
    double rc3=0.5;
    char c;
    led=0;

    wait(1);

    moteur1_rot_gauche.period_ms(10);
    moteur1_rot_droite.period_ms(10);
    moteur2_rot_gauche.period_ms(10);
    moteur2_rot_droite.period_ms(10);
    moteur3_rot_gauche.period_ms(10);
    moteur3_rot_droite.period_ms(10);
    enablePower = 1;

    //while(1){

    dist2=dist3=floor(1.0*nbr_points_rot360*(teta/360.0)); // pour une rotation de teta
    while (dist3>0) {
        moteur1_rot_gauche.write(0);
        moteur1_rot_droite.write(rc2);
        moteur2_rot_gauche.write(0);
        moteur2_rot_droite.write(rc2);
        moteur3_rot_gauche.write(0);
        moteur3_rot_droite.write(rc2);
        }
```

```
64      dist2=dist3=nbr_points*xy;
65      while (dist3>0) {
66          if (abs(dist2-dist3)>10) {
67              led = 1;
68              if ((dist2-dist3)>10){
69                  rc2=0.6;
70                  rc3=0.5;
71                  }
72              else {
73                  rc3=0.6;
74                  rc2=0.5;
75                  }
76              }
77          else{
78              rc2=rc3=0.5;
79              led=0;
80              }
81          moteur1_rot_gauche.write(0);
82          moteur1_rot_droite.write(0);
83          moteur2_rot_gauche.write(rc2);
84          moteur2_rot_droite.write(0);
85          moteur3_rot_gauche.write(0);
86          moteur3_rot_droite.write(rc2);
87          }
88      moteur1_rot_gauche.write(0);
89      moteur1_rot_droite.write(0);
90      moteur2_rot_gauche.write(0);
91      moteur2_rot_droite.write(0);
92      moteur3_rot_gauche.write(0);
93      moteur3_rot_droite.write(0);
94
95      enablePower = 0;
96      while(1);
97
98      }
99
100   //  }
```

```
101
102
103 /* Routine d'interruption */
104 void fonction_interruption_moteur2()
105 {
106   // ACTIONS A REALISER LORS DE L'ACTIVATION DE L'INTERRUPTION 1
107 dist2=dist2-1;
108 }
109 void fonction_interruption_moteur3()
110 {
111   // ACTIONS A REALISER LORS DE L'ACTIVATION DE L'INTERRUPTION 1
112 dist3=dist3-1;
113 }
```

5.2 Annexe 2

```
/* 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

//Initialisation 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;
char            etat;

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);

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

struct lidar_data   ld_current;


//Initialisation moteurs
InterruptIn encoche_moteur1(PB_0);
InterruptIn encoche_moteur2(PB_2);
InterruptIn encoche_moteur3(PB_4);



PwmOut moteur1_rot_gauche(PA_10);
PwmOut moteur1_rot_droite(PA_15);
PwmOut moteur2_rot_gauche(PB_14);
PwmOut moteur2_rot_droite(PB_13);
PwmOut moteur3_rot_gauche(PB_8);
PwmOut moteur3_rot_droite(PC_9);

DigitalOut  enablePower(PB_3);
int dist2;
int dist3;
int dist;
int distEvit;

int x = 5;
int y = 10;
int teta = floor(atan(1.0*y/x)*(180/3.14));
int nbr_points=11765; //pour 10 cm
int nbr_points_rot360=99000;

int xy = sqrt(x*x+y*y);

void fonction_interruption_Evit(void);
void fonction_interruption_moteur2(void);
void fonction_interruption_moteur3(void);
void fonction_Lidar(void);

int minDistance;
```

```
74 int minAngle;
75
76 int rangeDetection=150;
77 int angleDetection=floor(180/3);
78 //int rangeEvitemment=2*rangeDetection; //)*sqrt(3)/2.0; //R*sin(pi/3)
79
80 char mod_rotation=true;
81 char mod_translation=true;
82
83
84 /** MAIN FUNCTION
85  */
86 int main()
87 {
88     moteur1_rot_gauche.period_ms(10);
89     moteur1_rot_droite.period_ms(10);
90     moteur2_rot_gauche.period_ms(10);
91     moteur2_rot_droite.period_ms(10);
92     moteur3_rot_gauche.period_ms(10);
93     moteur3_rot_droite.period_ms(10);
94
95     double rc2=0.5;
96     double rc3=0.5;
97     char c;
98     enablePower = 0;
99     pc.baud(115200); // les 115200 bauds initialisés dans TeraTerm
100    encoche_moteur2.rise(&fonction_interruption_moteur2);
101    encoche_moteur3.rise(&fonction_interruption_moteur3);
102    encoche_moteur1.rise(&fonction_interruption_Evit);
103
104    int nb_tour = 0;
105    wait_s(3.0);
106    rotation.period(1/25000.0);
107    rotation.write(0.4);
108    wait_s(2.0);
109    pc.printf("\r\nLIDAR Testing\r\n");
```

```
110    lidar.attach(&IT_lidar);
111    wait_s(1.0);
112    pc.printf("\r\nLIDAR OK\r\n");
113
114
115
116    enablePower = 1;
117
118
119    etat=getHealthLidar();
120    pc.printf("%c",etat);
121    if (etat=='G'){
122        getInfoLidar();
123        getSampleRate();
124        // Start a new scan
125        startScan();
126        // Infinite Loop
127        tour_ok=0;
128        while(true) {
129            pc.printf("%d\n",tour_ok);
130            //wait(0.1);
131            if (tour_ok >=6) {
132                if(mod_rotation==true){
133                    tour_ok=0;
134                    stopScan();
135                    mod_rotation=false;
136                    //Module de rotation
137                    dist2=dist3=floor(1.0*nbr_points_rot360*(teta/360.0)); // pour une rotation de teta
138
139
140                    moteur1_rot_gauche.write(0);
141                    moteur1_rot_droite.write(rc2);
142                    moteur2_rot_gauche.write(0);
143                    moteur2_rot_droite.write(rc2);
144                    moteur3_rot_gauche.write(0);
145                    moteur3_rot_droite.write(rc2);
```

```
                    while (dist3>0) {
                        pc.printf("\r\nd3 = %d\r\n", dist3);
                        }
                    startScan();
                    }

                if (mod_translation==true){
                    tour_ok=0;
                    mod_translation=false;
                    //Module de translation
                    findMin(distance_scan_old, 0, 360, &minDistance, &minAngle);
                    dist=dist2=dist3=nbr_points*xy;
                    while (dist3>0 && (minDistance>rangeDetection || (minAngle>angleDetection && minAngle<360-angleDetection))) {
                        findMin(distance_scan_old, 0, 360, &minDistance, &minAngle);
                        pc.printf("\r\nd3 = %d\r\n", dist3);
                        if (abs(dist2-dist3)>10) {
                            led = 1;
                            if ((dist2-dist3)>10){
                                rc2=0.6;
                                rc3=0.5;
                                }
                            else {
                                rc3=0.6;
                                rc2=0.5;
                                }
                            }
                        else{
                            rc2=rc3=0.5;
                            led=0;
                            }
                        moteur1_rot_gauche.write(0);
                        moteur1_rot_droite.write(0);
                        moteur2_rot_gauche.write(rc2);
                        moteur2_rot_droite.write(0);
                        moteur3_rot_gauche.write(0);
                        moteur3_rot_droite.write(rc2);
```

```
                        }

                    moteur1_rot_gauche.write(0);
                    moteur1_rot_droite.write(0);
                    moteur2_rot_gauche.write(0);
                    moteur2_rot_droite.write(0);
                    moteur3_rot_gauche.write(0);
                    moteur3_rot_droite.write(0);

                    findMin(distance_scan_old, 0, 360, &minDistance, &minAngle);
                    if ((minDistance<rangeDetection) && ((minAngle<=angleDetection) || (minAngle>360-angleDetection))) {
                        fonction_Lidar();
                        }

                    }
                if (dist3==0) {
                    enablePower = 0;
                }
            }
        }
    }
}


/* Routine d'interruption */
void fonction_interruption_moteur2() {
  // ACTIONS A REALISER LORS DE L'ACTIVATION DE L'INTERRUPTION 1
dist2=dist2-1;
}
void fonction_interruption_moteur3() {
  // ACTIONS A REALISER LORS DE L'ACTIVATION DE L'INTERRUPTION 1
dist3=dist3-1;
}
```

```
216
217 void fonction_interruption_Evit() {
218   // ACTIONS A REALISER LORS DE L'ACTIVATION DE L'INTERRUPTION 1
219 distEvit=distEvit+1;
220 }
221
222
223 void fonction_Lidar() {
224     double rc2=0.5;
225     int minDistance, minAngle;
226         findMin(distance_scan_old, 0, 360, &minDistance, &minAngle);
227         stopScan();
228         //pc.printf("MA=%d",minAngle);
229         //pc.printf("M=%d",minDistance);
230         if (minAngle<=angleDetection) {
231                 moteur1_rot_gauche.write(0);
232                 moteur1_rot_droite.write(rc2);
233                 moteur2_rot_gauche.write(rc2);
234                 moteur2_rot_droite.write(0);
235                 moteur3_rot_gauche.write(0);
236                 moteur3_rot_droite.write(0);
237                 startScan();
238                 distEvit=0;
239             while ((minDistance<rangeDetection) && ((minAngle<=90) || (minAngle>270))) {
240
241                 pc.printf("A");
242                 if (tour_ok >= 6) {
243                     tour_ok = 0;
244                 findMin(distance_scan_old, 0, 360, &minDistance, &minAngle);
245                 pc.printf("M=%d",minDistance);
246                 }
247             }
248             pc.printf("MA=%d",minAngle);
249
250             moteur1_rot_gauche.write(0);
251             moteur1_rot_droite.write(0);
252             moteur2_rot_gauche.write(0);
253             moteur2_rot_droite.write(0);
254             moteur3_rot_gauche.write(0);
255             moteur3_rot_droite.write(0);
256         }
257         else{
258             if (minAngle>360-angleDetection) {
259                     moteur1_rot_gauche.write(rc2);
260                     moteur1_rot_droite.write(0);
261                     moteur2_rot_gauche.write(0);
262                     moteur2_rot_droite.write(0);
263                     moteur3_rot_gauche.write(0);
264                     moteur3_rot_droite.write(rc2);
265                     startScan();
266                     distEvit=0;
267                 while ((minDistance<rangeDetection) && ((minAngle<=90) || (minAngle>270))) {
268                     if (tour_ok >= 6) {
269                     tour_ok = 0;
270                     findMin(distance_scan_old, 0, 360, &minDistance, &minAngle);
271                     }
272                 }
273                 stopScan();
274
275                 moteur1_rot_gauche.write(0);
276                 moteur1_rot_droite.write(0);
277                 moteur2_rot_gauche.write(0);
278                 moteur2_rot_droite.write(0);
279                 moteur3_rot_gauche.write(0);
280                 moteur3_rot_droite.write(0);
281             }
282         }
283         xy = 10; // sqrt(((dist-dist3-distEvit/2)*(dist-dist3-distEvit/2)+(distEvit*sqrt(3)/2)*(distEvit*sqrt(3)/2))/nbr_points);
284         teta = 10; // acos(1.0*(dist-dist3-distEvit/2)/(xy*nbr_points))*(180/3.14);
285         pc.printf("teta=%d",teta);
286         mod_rotation=true;
287         mod_translation=true;
288         startScan();
289         wait(1);  //Pour que la manoeuvre d'evitement fonctionne
290 }
```