GENTIL Solène
BEN HASSEN Yasmine
VIALY Carla
BELGUERRAS Orane
Groupe 4

DOSSIER TECHNIQUE :

SCOY : la voiture autonome

1. **Comprendre le projet**

   A) Introduction
   B) Découpage fonctionnel

2. **Réalisation du prototype**

   A) Schémas électriques
   B) Algorithmes

3. **Résultats finaux**

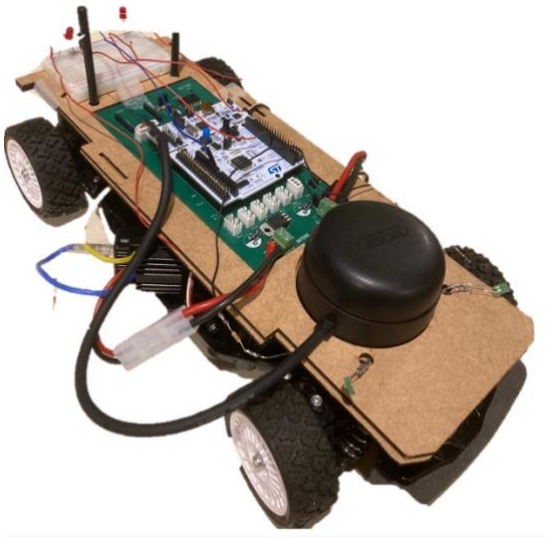
4. **Comprendre les étapes de réalisation/choix**

   A) Planning
   B) Difficultés rencontrées, analyse du travail d'équipe

ANNEXE

1. Comprendre le projet

A) Introduction



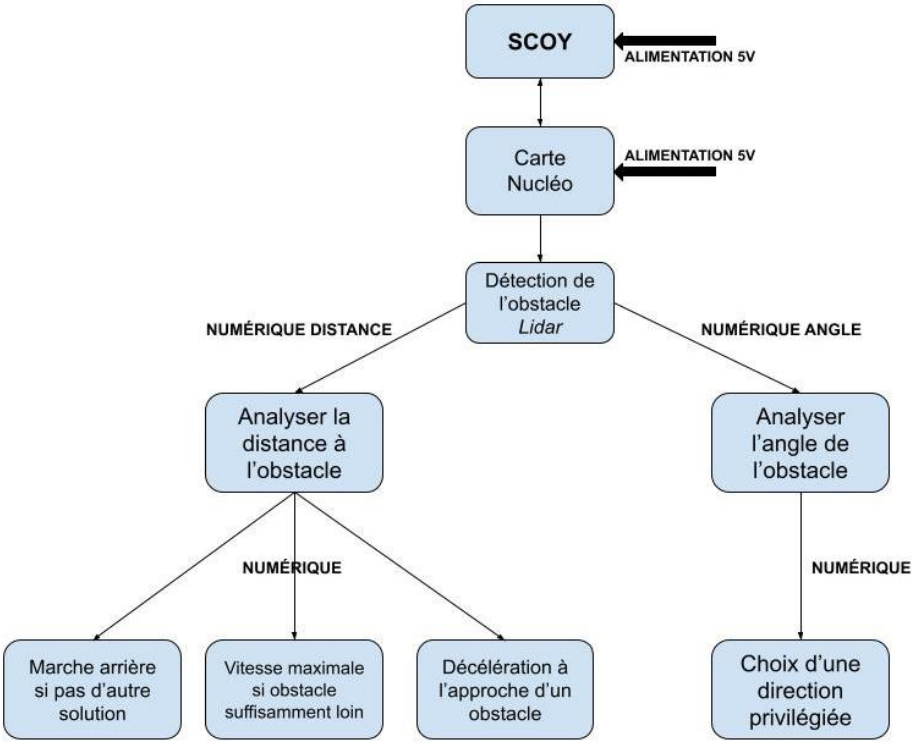
Pour ce projet de Iéti, nous avons choisi de réaliser une voiture autonome, SCOY, dont le but est d'être capable de se diriger sans l'intervention d'un conducteur. Pour se faire, nous avons choisi d'utiliser un Lidar RPLidar A2M8, qui envoie l'information à un microcontrôleur qui définira, après avoir traité l'information, la vitesse, la marche avant ou marche arrière et l'angle des roues.

Le cadre de notre projet était de faire en sorte que SCOY puisse se diriger sur une surface plane sans heurter d'obstacles, quitte à faire marche arrière si l'obstacle n'est pas évitable par un virage.

Les tâches que nous nous sommes réparties pour mener à bien notre projet sont les suivantes :

- Choisir entre les capteurs et le Lidar en fonction de la praticité et de nos objectifs (voiture rapide mais peu mobile avec les capteurs, ou mobile mais moins rapide avec le Lidar)
- Déterminer la relation entre l'angle envoyé aux roues et l'angle envoyé par le Lidar
- Ecrire un code d'initialisation de la voiture pour vérifier que toutes les vitesses et que tous les angles sont accessibles
- Réaliser un code efficace pour envoyer l'information de l'angle et de la vitesse à la voiture en fonction de l'environnement extérieur
- Déclencher une marche arrière si l'obstacle est trop proche
- Faire en sorte que SCOY ne prenne pas des virages de manière excessive, juste quand c'est utile

B) <u>Découpage fonctionnel</u>

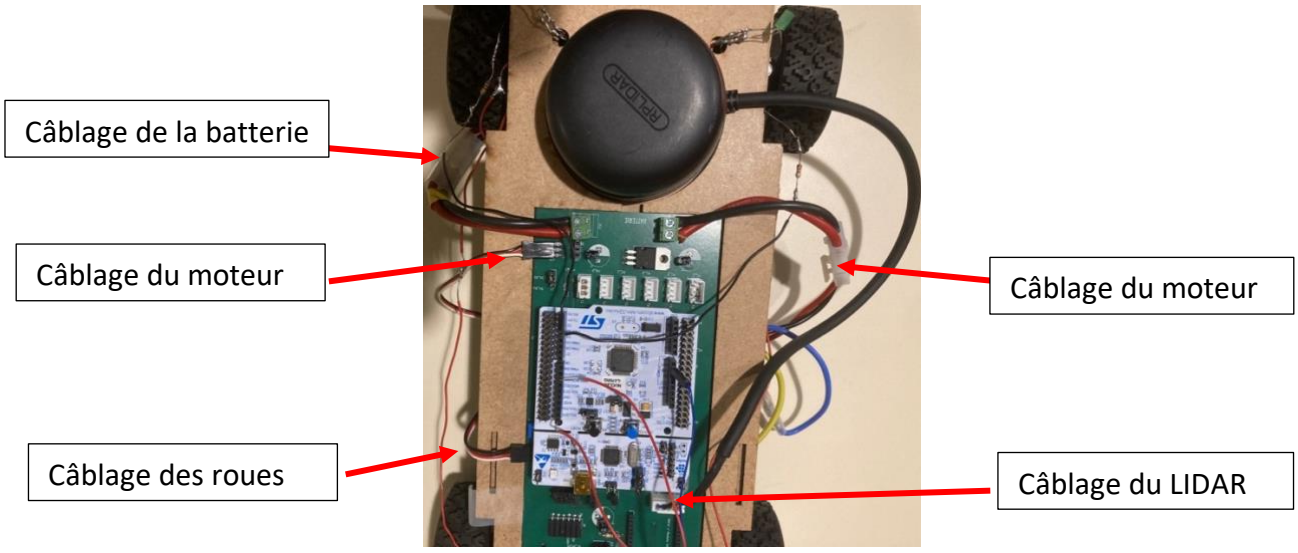


2. Réalisation du protoype

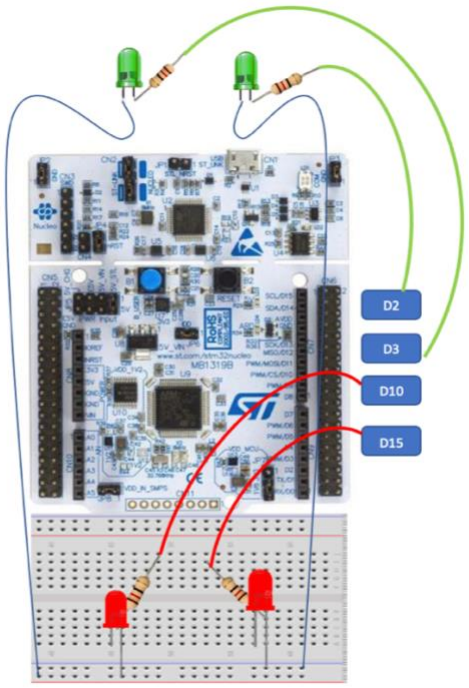
I) <u>Schémas électriques</u>

Afin de réaliser le prototype, nous avons réalisé 2 câblages principaux : le câblage de la voiture (moteur, roues) et celui des accessoires (LIDAR, LEDs avants et arrières).

1. <u>Câblage de la voiture et du LIDAR</u>:



2. Câblage des LEDs :



Calcul de la résistance de protection :

$$\boldsymbol{R = \frac{U_{max-V_f}}{I_f}}$$

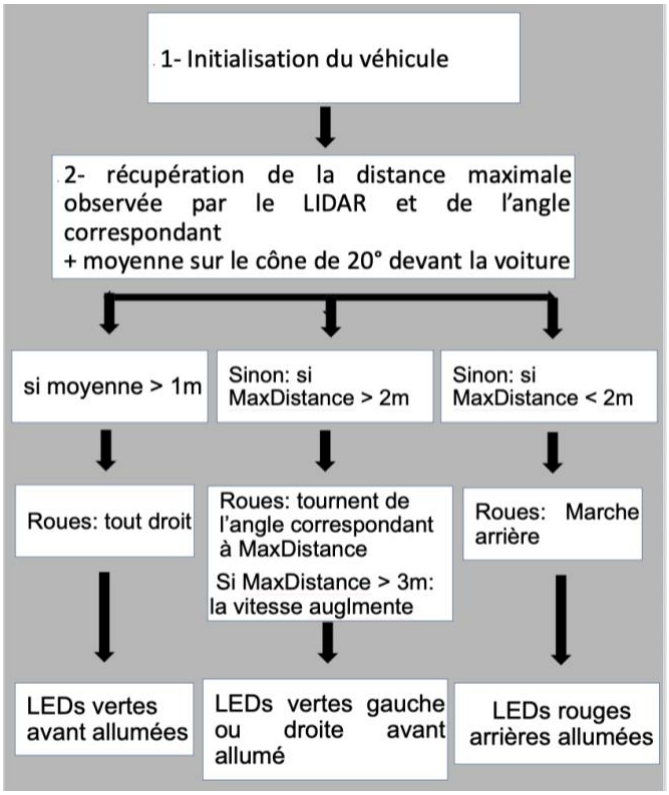
Avec U_{max} la tension maximale convertit par la carte Nucléo soit 3.3V, V_f le « forward Voltage » et I_f la « forward Intensity » donnés dans la fiche technique de la LED rouge (≈verte).

$$AN: R = \frac{3{,}3 - 2{,}2}{30 \times 10^{-3}} = \boldsymbol{37\ \Omega}$$

Nous avons choisi une résistance **R=100 Ω** pour les 4 LEDs.

II) Algorithmes

Le code que nous avons mis en place au cours de ce projet est détaillée dans le schéma suivant :



Voici les codes finaux :

- Initialisation moteur et roues

```
int nb_tour = 0;
// Initialisation moteur et direction
direction.period_ms(20);        //période de réception du servomoteur
direction.pulsewidth_us(1300);  //tout droit
moteur.period_ms(20);           //période de réception du moteur
moteur.pulsewidth_us(1500);     // 1500 = point mort

rotation.period(1/25000.0);
rotation.write(0.4);

wait_s(3.0);
pc.printf("\r\nLIDAR Testing\r\n");
lidar.attach(&IT_lidar);
wait_s(1.0);
pc.printf("\r\nLIDAR OK\r\n");

 // TRANSMISSION PC et BT
pc.printf("Programme Test \r\nVoiture Autonome \r\n6 capteurs\r\n");

for(angle_roues = 1300; angle_roues < 1500; angle_roues+=10){
    direction.pulsewidth_us(angle_roues);
    pc.printf("Ang=%d  \r\n", angle_roues);
    wait_us(100000);
}
for(angle_roues = 1500; angle_roues > 1100; angle_roues-=10){
    direction.pulsewidth_us(angle_roues);
    pc.printf("Ang=%d  \r\n", angle_roues);
    wait_us(100000);}

angle_roues = 1300;
```

- Récupération des données du LIDAR + calcul de la moyenne sur la zone de 20° devant la voiture. (En notant que la direction tout droit représente un angle de 90° du LIADR)

```
getHealthLidar();
getInfoLidar();
getSampleRate();
// Start a new scan
startScan();

// Infinite Loop
while (true) {

    if(trame_ok){
        debug_tour = !debug_tour;
    }

    if(tour_ok == 6){
        int maxDistance, maxAngle;
        int i;
        double m=0;
        tour_ok = 0;
        // moyenne distance
        int C=1;
        int S=0;
        for (i=0;i<=20;i++){
            if (distance_scan_old[80+i]!=0){
                C+=1;
                S+=distance_scan_old[80+i];
            }
        }
        m=1.0*S/C;  //on fait une moyenne entre 80° et 100° des distances max d'un obstacle
        findMax(distance_scan_old, 60, 120, &maxDistance, &maxAngle);
        pc.printf("m=%lf  %d\r\n", m, C);
        pc.printf("A=%d %d\r\n", maxAngle, maxDistance);
```

- Ordres donnés au moteur, aux roues et aux LEDs en fonction de la distance séparant la voiture de l'obstacle le plus proche et en prenant en compte le calcul de la moyenne sur le cône de 20° devant la voiture :

```
if (m>1000){   // Si cette moyenne est supérieur à 1000mm : la voiture va tout droit

    led1_avant=1;   //on éteint les leds avant et on allume les leds arrière
    led2_avant=1;
    led1_arriere=0;
    led2_arriere=0;

    angle_roues = 1300;
    vitesse = 1620;
    direction.pulsewidth_us(angle_roues);
    moteur.pulsewidth_us(vitesse);
}
else{

    if (maxDistance>2000){      //la voiture va tourner mais continue d'avancer
        led1_arriere=0;
        led2_arriere=0;
        led1_avant=0;
        led2_avant=0;
        AR = 1300 + 13.3*(maxAngle-90);
        if (AR<1100) {
            angle_roues=1100;
            direction.pulsewidth_us(angle_roues);}
        else {
            if (AR>1500){
                angle_roues=1500;
                direction.pulsewidth_us(angle_roues);}
            else {
                angle_roues=AR;
                direction.pulsewidth_us(angle_roues);}
                  }

       pc.printf("\tAvant\r\n");
       if (angle_roues<1300){
            led1_avant=0;  //la voiture va à gauche : on allume led avant gauche
            led2_avant=1;
       }
       else{
            led1_avant=1;  //la voiture va à droite : on allume led avant droite
            led2_avant=0;
       }
       if (maxDistance<3000){
            vitesse = 1615;
            moteur.pulsewidth_us(vitesse);}
       else{
            vitesse = 1620;
       //wait_s(0.2);
            moteur.pulsewidth_us(vitesse);
       }
   }
    else {
        led1_avant=0;     //on éteint les leds avant et on allume les leds arrière
        led2_avant=0;
        led1_arriere=1;
        led2_arriere=1;

        angle_roues = 1300;
        direction.pulsewidth_us(angle_roues);
        pc.printf("\tArriere\r\n");
        vitesse = 1380;
        moteur.pulsewidth_us(vitesse);
    }
}
```

3. Résultats finaux

Pour quantifier la performance de notre voiture, nous avons mis en place **des parcours d'obstacles** (cartons…) **dans le couloir** et observions la réponde de la voiture aux différentes contraintes de son environnement (murs, poteaux, obstacles ajoutés…). Au cours de ces nombreux tests (chaque séance), nous avons optimisé notre algorithme et ainsi améliorer les performances de SCOY. Voici une liste des principaux problèmes rencontrés lors de ces essais :

1. Problèmes rencontrés et réglés

Problème	**La voiture tournait à droite quand elle devait aller tout droit**	**Pas le temps d'éviter l'obstacle : vitesse trop importante à l'approche de l'obstacle**	**Beaucoup de virages inutiles**
Solution	Le châssis n'était plus utilisable (roues désaxées) : échange de châssis → pause pb pour l'optimisation	Réduction de la vitesse à l'approche de l'obstacle, pour prendre le virage	Mise en place d'un calcul de moyenne sur les distances → la voiture ne tourne pas pour rien

Des problèmes restent en suspens mais nous avons réfléchi à des idées pour les résoudre :

2. Points d'amélioration à considérer

Problème	**Arrêt de la voiture quand elle échoue à éviter l'obstacle (pas de marche arrière)**	**Optimisation de l'angle des roues : changement de châssis à chaque séance**	**LIDAR qui tombe facilement**	**SCOY ne prend pas en compte son gabarit**
Solution à considérer	Lancer l'initialisation automatiquement quand la voiture est au point mort (vitesse = 1500)	Avoir un châssis par groupe ☺	Le fixer sur le châssis	Faire des moyennes sur des cônes d'au moins 10°

Nous avons finalement observé des résultats concluants. Vous trouverez joint au mail des vidéos de SCOY en action.

4. Comprendre les étapes de réalisation/choix

A) Planning

Séance 1 :
Découverte du sujet
Planification des objectifs
Répartition des tâches
Séance 2 :
Séparation en deux groupes de deux
Un binôme travaille à faire démarrer la voiture, réussite du programme d'initialisation de la voiture
Un autre étudie les capteurs et le Lidar et les compare pour faire un choix --> choix du Lidar
Séance 3 :
Première version du code : marche avant jusqu'à détection d'obstacle, virage si obstacle détecté en dessous d'une certaine distance
Détermination relation entre angle envoyé par le Lidar et angle des roues à adopter
Premiers essais au sol
Séance 4 :
Optimisation du code
Ajout du calcul de moyenne sur les distances captées par le Lidar
Ajout de la marche arrière si obstacle trop proche
Essais très concluants au sol
Séance 5 :
Reprise du code car problèmes de comportement inexpliqués (code identique mais la voiture ne réagissait plus aussi bien)
Séance 6 :
Toujours problèmes de comportement inexpliqués
Revissage de la roue droite car SCOY partait à droite pendant la commande "tout droit"
Changement de châssis
Ajout des LEDs avants et arrières
Séance 7 :
Dernières vérifications et optimisations de la voiture
Problèmes apparus en séance 5 partiellement mais pas totalement réglés
Séance 8 :
Présentation du projet

B) Difficultés rencontrées, analyse du travail d'équipe

Travail d'équipe :
Le travail s'est naturellement divisé en deux au commencement du projet, lorsqu'il s'agissait de rassembler et d'intégrer beaucoup d'informations (fonctionnement du moteur, choix

entre capteurs et Lidar). Après ces études préliminaires, les deux duos se sont naturellement regroupés en un groupe pour travailler sur le code. Nous alternions entre optimisation du code/brainstorming à 4 et essais au sol pendant la quasi-totalité des séances.
La seule difficulté concernant le travail d'équipe que nous pourrions mentionner est une certaine désorientation et peine à commencer concrètement le projet --> quelques lenteurs et tâtonnements au démarrage.

Difficultés rencontrées :
Si le travail en équipe s'est déroulé de manière fluide, nous avons en revanche dû faire face à quelques difficultés mécaniques, algorithmiques et électroniques :

- Changement de châssis au milieu du projet pour régler une réaction étrange au code qui marchait bien jusqu'à présent
- Changement de batterie car la voiture s'arrêtait parfois sans redémarrer et roulait trop doucement
- Trop de virages inutiles
- Situations difficiles à gérer : parfois SCOY faisait des allers-retours marche avant/marche arrière sans jamais "changer de stratégie", et restait donc bloquée au même endroit
- Pas de prise en compte de son gabarit --> veut passer dans des endroits parfois trop exigus

ANNEXE :

Définitions des variables utilisées dans le code

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

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

//IMPORTANT : le 180° est au niveau du fil et le fil est sur le côté droit
//(perpendiculairement à la direction de la voiture) tel que l'angle "tout droit" est 90°
// distance donnée par le LIDAR en mm

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;
double AR;


DigitalOut my_led(LED1);
PwmOut direction(PB_13);    // Servomoteur de direction
PwmOut moteur(PC_9);        // Motorisation / ESC

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);
int angle_roues, vitesse;

Serial              lidar(A0, A1, 115200);
PwmOut              rotation(PB_9);     // D14

//LEDs rouges arrières
DigitalOut          led1_arriere(D15);  //led arriere droite
DigitalOut          led2_arriere(D10);
DigitalOut          led1_avant(D3);  //led avant droite
DigitalOut          led2_avant(D2);

struct lidar_data   ld_current;
```