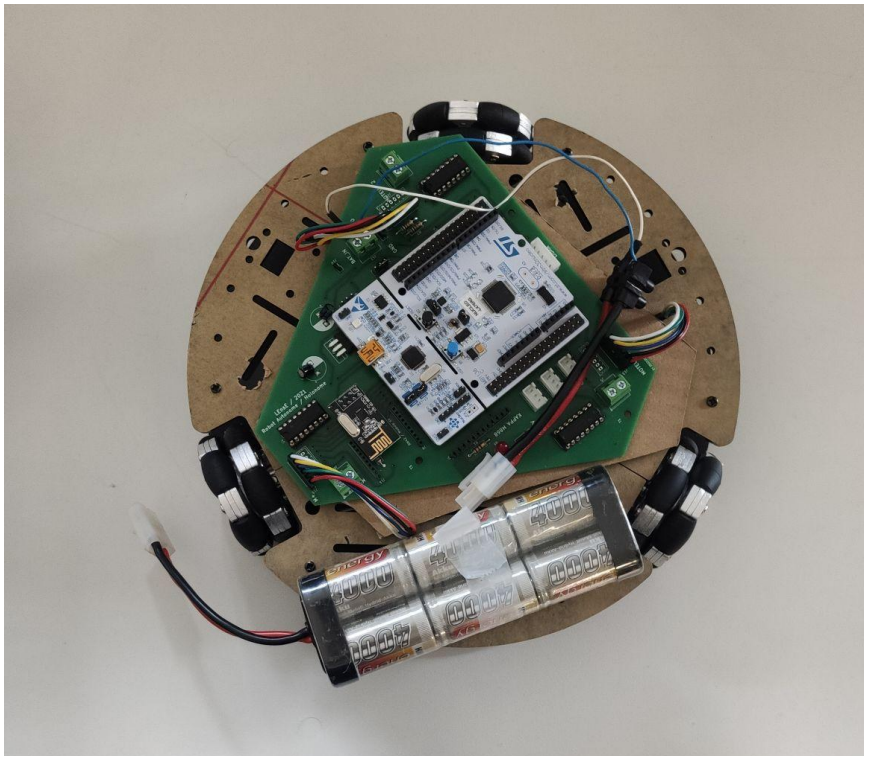
Projet Robot holonome piloté par joystick:

Aymane Hamim, Mathéo Kina, Marwane Youssoufi, Matthieu Prouteau



Tables des matières

Objectifs du projet:

Le Robot holonome est un robot à trois degrés de liberté (2 translations et une rotation), capable de se translater dans toutes les directions sans aucune rotation du châssis. Ce robot est de plus en plus présent dans l'industrie et le quotidien. Notre but sera de piloter un robot holonome à trois roues à l'aide d'une télécommande(Joystick), et donc coder les commandes du pilotage d'un côté et assurer la communication avec la télécommande d'un autre côté, afin de réaliser cela nous disposons de :

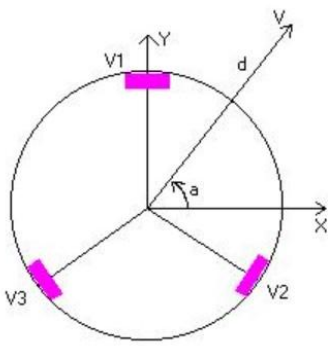
- 2 Cartes de communication nRF24L01
- 3 Moteurs à courant continu
- 2 Cartes Nucléo
- 2 Joystick (une pour la rotation et l'autre pour les translations)

Cahier des charges :

A l'aide de ces objectifs nous avons fixé le cahier de charges suivant :

-Déplacement du robot dans toutes les directions du sol
-Rotation dans les deux sens à 360°
-Contrôle à distance via une télécommande:
Joystick
Distance émetteur récepteur 10 m
Temps de réponse<1s
-Alimentation par batterie

Principe de fonctionnement de la partie robot:



1)Translation

Pour que le centre du robot avance à une vitesse V et un angle a, la vitesse des moteurs est donnée par les relations suivante:
V1 = V * sin(a - 90)
V2 = V * sin(a + 30)
V3 = V * sin(a + 150)
Nos premiers expérience ont montré que pour un angle 'a' le robot avance à un angle$\sim a - 30$
il fallait alors décaler l'angle par trente afin d'obtenir des réponses satisfaisantes.

figure 1 :Schéma des vitesses:http://sitecv.free.fr/holonome/

Les moteurs continues sont alimenté par des hacheurs ou bien dans notre cas par des sorties PWM, (modulation de largeur d'impulsion) on contrôle alors la vitesse des moteurs par les rapports cyclique, et donc :$V \equiv r_c$

```
a=a+30;
if (sin( (a - 90) *toRad )>0) {
    rc11 = vitesse*sin( (a - 90)*toRad );
    rc12 = 0;
}
else {
    rc11 = 0;
    rc12 = - vitesse*sin( (a - 90)*toRad );
}
```

Le programme de pilotage d'un moteur est alors le suivant (cas moteur 1), chaque moteur est alimenté par deux sorties PWM chacune correspond à un sens de rotation (figure 3), et alors toutes les vitesses sont à signe positives car pour fonctionner dans le sens inverse il suffit d'alimenter l'entre sortie.

figure 2: Code pilotage moteur 1

```
PwmOut MOT2_D1(PB_14);  /* Moteur 2 dans le sens directe */
PwmOut MOT2_D2(PB_13); /* Moteur 2 dans le sens indirecte */
PwmOut  MOT3_D1(PB_8); /* Moteur 3 dans le sens directe */
PwmOut  MOT3_D2(PC_9); /* Moteur 3 dans le sens indirecte */
```

figure 3 : Définition de sortie pour l'alimentation des moteur du robot (montre le principe de fonctionnement du robot et du code)

La définition de la fonction de direction nécessite alors 2 arguments: la vitesse qui est un réel compris entre 0 et 1 ou 1 est la vitesse maximale possible, et a un angle compris entre 0 et 360.

2) Rotation:

Le pilotage en rotation est plus facile comme il suffit de faire tourner les trois roues dans un même sens et en même vitesse, la fonction rotation prend alors deux argument, une vitesse de rotation et un $epsilon = 0\ ou\ 1$, qui donne le sens de rotation du robot 1 étant le sens trigonométrique.

3) Remarques:

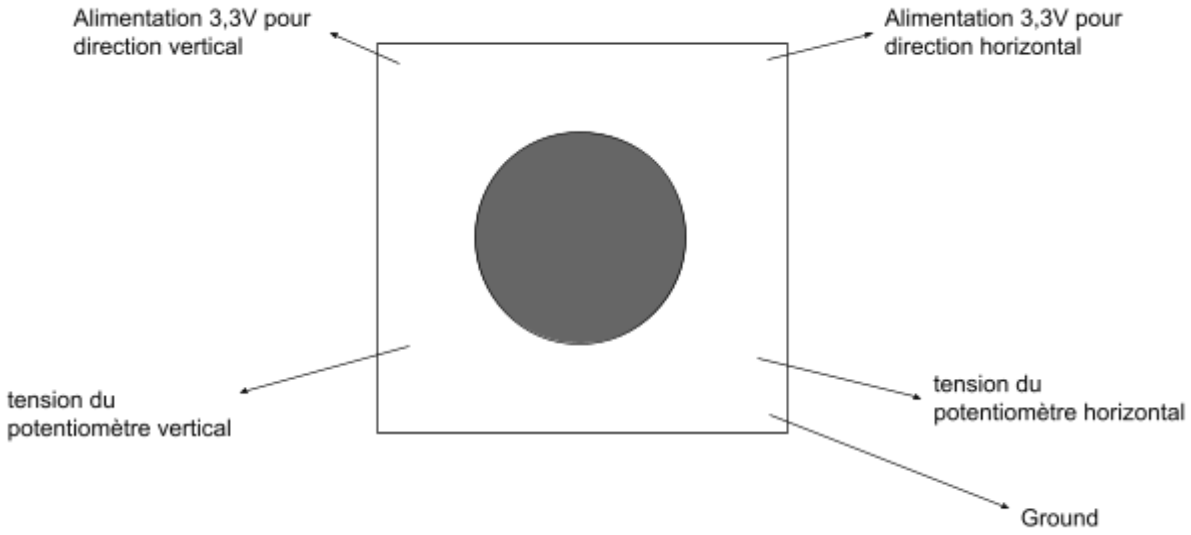
Ce qui est le plus compliqué dans cette partie est la définition des sorties et la compréhension des branchages des moteurs. Le principe du code en soi est simple, mais il est nécessaire de le modifier empiriquement, limiter la vitesse à une valeur empirique, décaler l'angle…etc
Le choix des noms des variables et les commentaires sont très importants puisqu'on travaille avec un nombre très grand de variables.

Parties télécommande

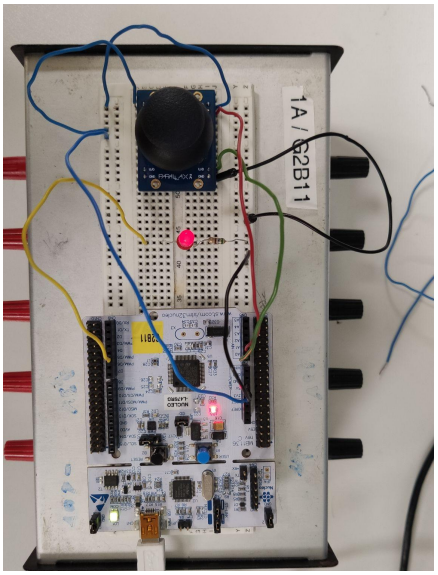
1) Fonctionnement d'un joystick

Un joystick permet de faire varier la tension selon deux axes distincts x et y. En récupérant la tension sur les deux axes on peut ainsi connaître la direction où le joystick a été incliné.
Pour notre robot, il est nécessaire aussi d'avoir l'inclinaison du joystick qui correspondra à la vitesse de notre robot.

Pour tester le fonctionnement de notre joystick de manière plus simple, on commence par le réglage de l'intensité d'une led.
On récupère la tension normée entre 0 et 1 sur une des coordonnées et on le fait correspondre au rapport cyclique de notre led en fonctionnement PWM.



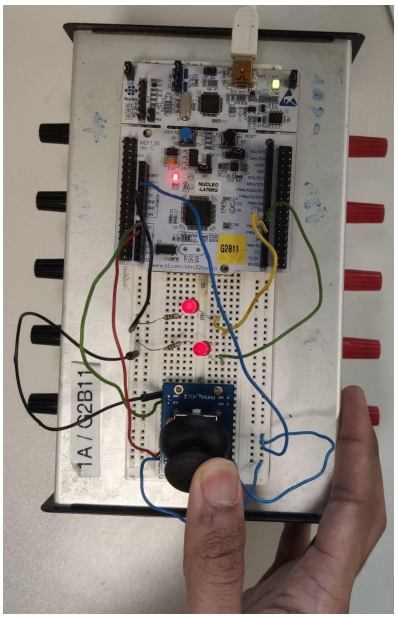
Câblage d'un joystick

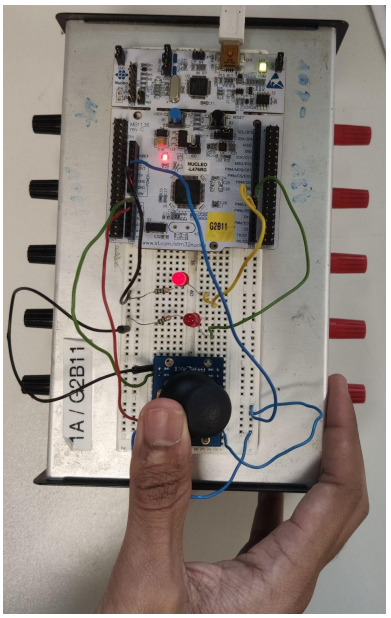


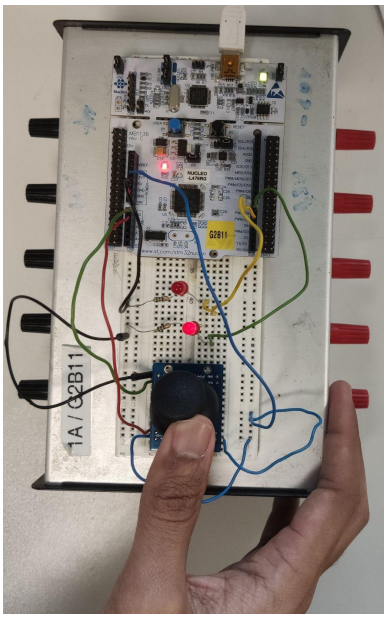
On a donc l'intensité de la led qui varie en fonction de l'inclinaison du joystick:

- intensité max en position haute
- intensité moyenne en position neutre
- intensité nulle en position basse

Ensuite on teste le même fonctionnement en ajoutant une led sur l'autre axe du joystick.







On a donc un joystick fonctionnel où l'on est capable de récupérer la direction et l'inclinaison du joystick.

2) Carte réseau nRF24

Pour utiliser la carte on utilise la carte pcb du robot connecté avec la carte nucleo. Grâce à cela on utilise la bibliothèque de fonction disponible sur le site du lense:
https://github.com/IOGS-LEnsE-embedded/MBED6_SupOpLibraries/tree/main/nRF24

En particulier, on utilisera les fonctions suivantes :

initNRF24() : qui permet d'initialiser la carte réseau et de vérifier qu'il n'y est aucun dysfonctionnement de la carte.

nRF24_mod.write(NRF24L01P_PIPE_P0, tab, TRANSFER_SIZE) : qui permet d'envoyer les chaînes de caractères contenu dans le tableau tab (de type char).

nRF24_mod.read(NRF24L01P_PIPE_P0, dataReceived, TRANSFER_SIZE) : qui permet de stocker dans le tableau dataReceived les chaînes de caractères reçues.

3) Conversion des données des joysticks

La carte réseau ne peut envoyer que des variables de type char. Les tensions lues sur les joysticks sont des float, il faut donc les convertir.

La première option envisagée était de convertir la tension lue et de l'envoyer directement à la carte réseau. Cependant les valeurs reçues avait de forte variation après le processus et on en perdait toutes l'information.

On opte donc finalement pour une quantification des plages de tensions avant l'envoi par la carte réseau.

Pour les commandes de directions

Un joystick donne 2 valeur de tension :

- joyTX entre 0 et 1 pour la direction horizontal
- joyTY entre 0 et 1 pour la direction vertical

Cependant on souhaite avoir le robot immobile quand le joystick est en position neutre et une vitesse max égale à 1.
On normalise alors les variables

```
x=-2*(joy_TX-0.5);     // entre -1 et 1
y=2*(joy_TY-0.5);      // entre -1 et 1
```

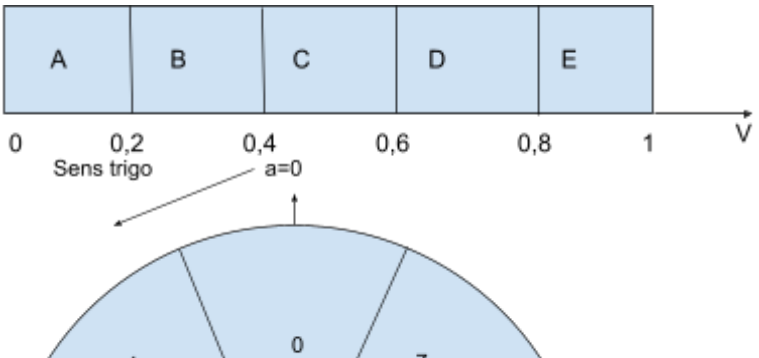
Nos fonctions directions prennent en entrées les coordonnées (a,v) où a est l'angle de la direction souhaité et v la vitesse normalisé entre 0 et 1
On effectue donc le changement de variable (x,y)→(a,v)

```
v=sqrt(pow(x,2)+pow(y,2));
```

```
if (x>0)
        {
            a=atan(y/x);

        }
        else if (x==0)
        {
            if(y>0)
            {
                a=pi/2;
            }
        }
        else{
            a=atan(y/x)+pi;
        }
        theta=a-pi/2;
```

On quantifie ensuite la vitesse et l'angle. On associe un caractère à chaque plage.



On quantifie le cercle trigonométrique en arc de cercle d'angle pi/4.

On envoie les caractères correspondant aux zones. La partie robot associe à chaque caractère la bonne valeur :

- pour la vitesse on associe la borne minimum de l'interval
- pour l'angle on associe l'angle médian de l'interval

<u>Pour les commandes de rotations</u>

Un joystick donne 1 valeur de tension :

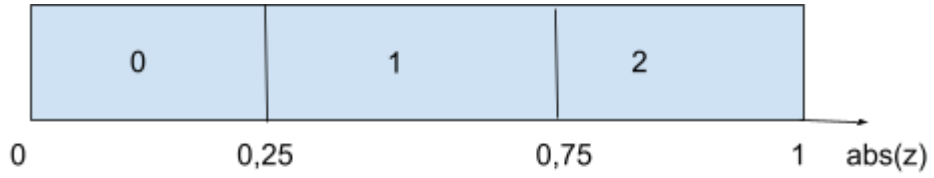
- rotation entre 0 et 1 pour la direction horizontal

On normalise rotation pour avoir une nouvelle variable z entre -1 et 1.

On associe le signe de z a un caractère :

- si z négatif : 0
- si z positif : 1

On quantifie la valeur absolue de z :



On envoie les 2 caractères (le signe et la valeur absolue) avec la carte réseau.
Le robot interprète les caractères comme la valeur minimale de l'intervalle correspondant.