RO52 - TP Robot Suiveur

> Ce rapport est écrit par **RASSIÉ Nathan** et **DERAISIN Nicolas**.
>
> Il a été réalisé en **Markdown**.
>
> Le programme présenté ici a été écrit en **Python 3**.

Préparation

Pour préparer ce TP, nous avons récupéré notre code du TP1 contenant les **fonctions de déplacement de base du robot** en suivi de ligne, avec ses différentes corrections pour asservir sa vitesse angulaire, et l'avons réorganisé pour le rendre plus modulaire.

En paralèlle, nous avons implémenté les différentes fonctions demandées (celle pour le Leader et celles pour le Follower).

Enfin, nous avons intégré à notre classe Robot les deux modes du Robot.

Robot Leader

Le robot leader est le plus simple à mettre en place, il se contente d'avancer pendant un temps T puis de s'arrêter pendant ce même temps. Nous avons donc réalisé cela, en y intégrant les éléments d'envoi de sa vitesse par wifi pour le 4eme mode du robot follower.

```
# Duration of run / wait (in ms)
T = 3000

def lead_driving(robot: DriveBase, speed: float, turn_rate: float, send_speed: bool):

    if send_speed:
        # Instanciate this robot as a Server
        server = BluetoothMailboxServer()
        mbox = TextMailbox("SpaceTab", server)

        # The server must be started before the client!
        print("Waiting for connection...")
        server.wait_for_connection()
        print("Connected !")

    timer = StopWatch()
    timer.reset()

    while(True):
        # Drive 10sec then wait 10sec
        robot.drive(speed, turn_rate)

        if timer.time() > T:
            if send_speed: mbox.send(str(speed))
            wait(T)
            timer.reset()
```

Robot Follower

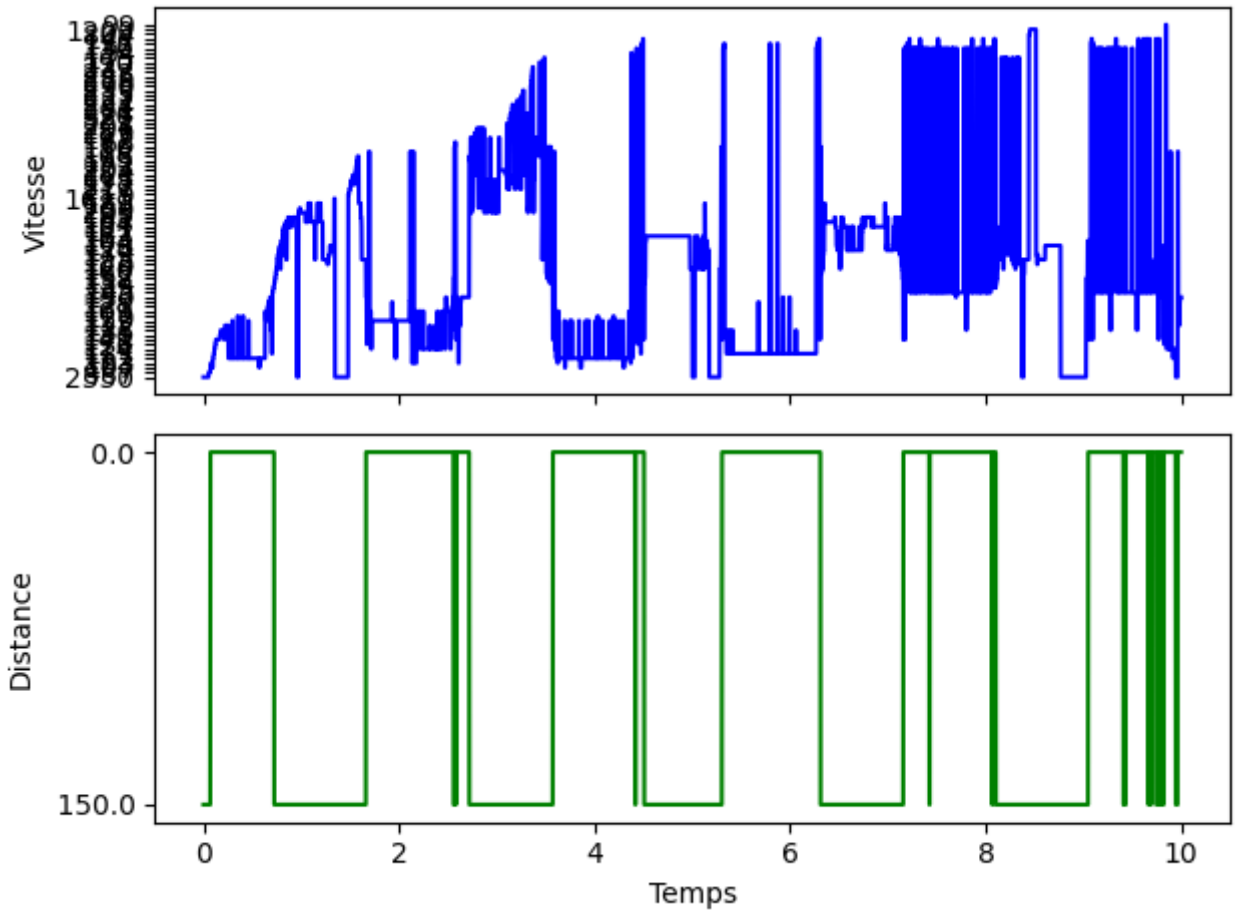
Toutes les fonctions du robot follower se basent sur la distance mesurée devant lui à l'aide d'un capteur à ultrasons.

All or None

Le mode tout ou rien consiste à immédiatement arrêter le robot lorsqu'il se trouve à 15cm ou moins du leader.

```
# Robot stop driving if it is closer than 15cm of the lead robot
def all_or_none(distance: int):
    if distance < 150: speed = 0.0
    else: speed = 0.5 * MAX_SPEED
    return speed
```

Ceci nous donne la courbe vitesse / distance suivante :



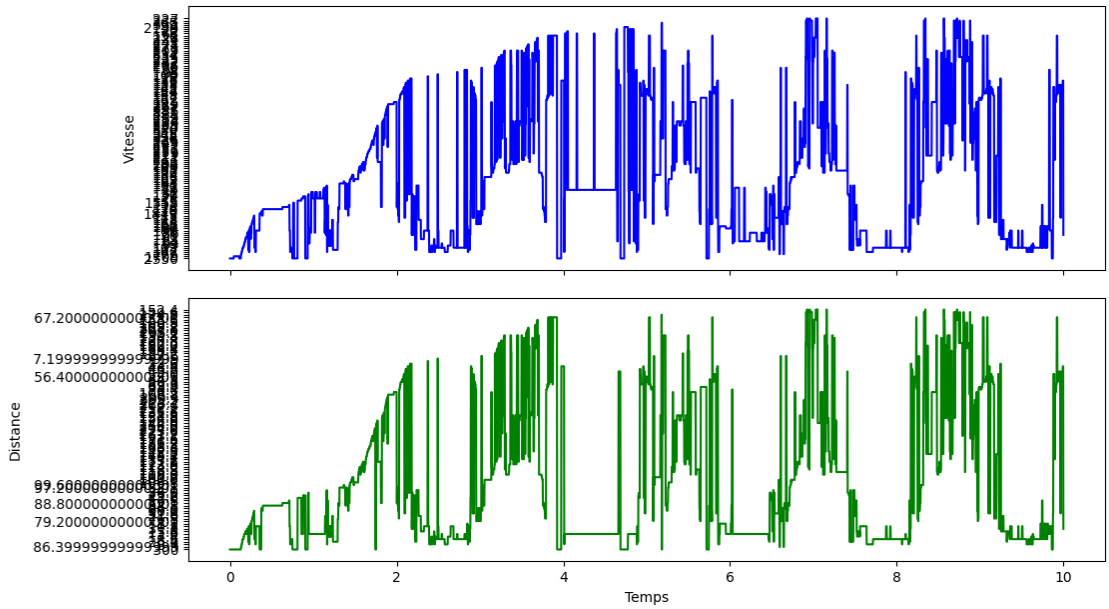
One point

L'asservissement de vitesse à un point consiste à utiliser un facteur de proportionnalité sur la distance, à partir d'une certaine distance donnée. Ces deux paramètres sont utilisé à la fois pour la vitesse d'accélération que pour la vitesse de décélération.

```
BREAK_DISTANCE = 100 # Distance threshold to start breaking
BREAK_FACTOR = 100 # Break factor


# Robot's speed is proportional to a defined factor and break distance, unless it
overcomes MAX_SPEED
def one_point(distance: int):
    speed = max(min(MAX_SPEED, BREAK_FACTOR * (distance - BREAK_DISTANCE)), 0)
    return speed
```

Ceci nous donne la courbe vitesse / distance suivante :



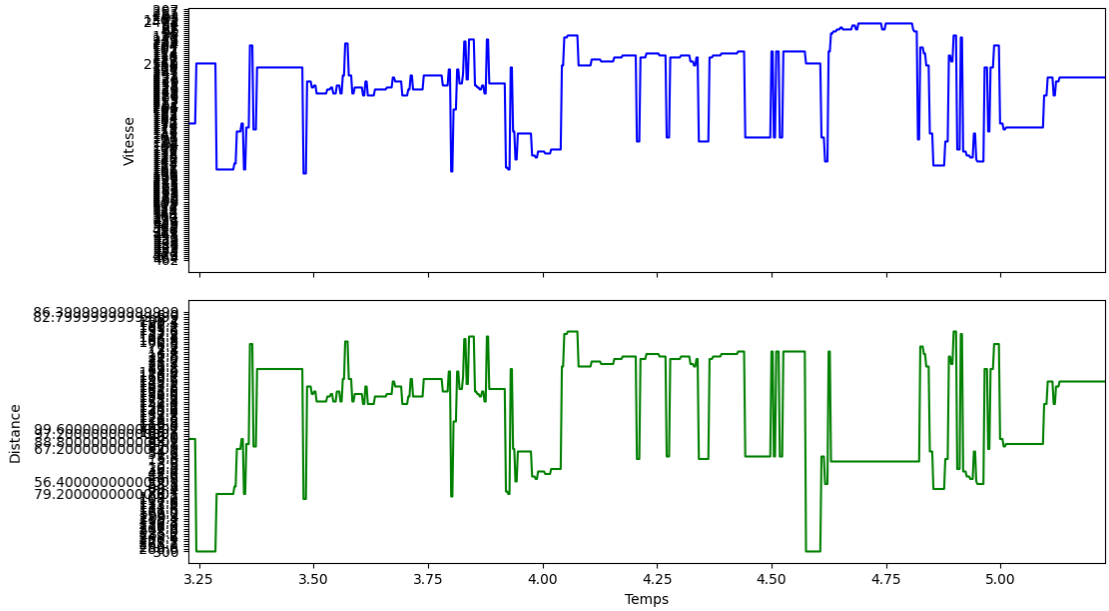
Two Points

L'asservissement de vitesse à deux points garde le même principe que le précédent, mais différencie les deux paramètres pour la vitesse d'accélération et celle de décélération.

```
BREAK_DISTANCE = 100 # Distance threshold to start breaking
ACCELERATION_DISTANCE = 100 # Distance threshold to start accelerating
BREAK_FACTOR = 1.2 # Break factor
ACCELERATION_FACTOR = 1.2 # Acceleration factor# Robot's acceleration and breaking are
proportional to defined factors and distances, unless they overcomes MAX_SPEED

def two_point(distance: int, previous_speed: int):
 break_speed = min(max(BREAK_FACTOR * (distance - BREAK_DISTANCE), 0), MAX_SPEED)
 acceleration_speed = min(max(ACCELERATION_FACTOR * (distance -
ACCELERATION_DISTANCE), 0, previous_speed), MAX_SPEED)
 speed = min(break_speed, acceleration_speed)
 return speed
Ceci nous donne la courbe vitesse / distance suivante :
```

Ceci nous donne la courbe vitesse / distance suivante :



Wifi

Le mode wifi permet de connecter le robot leader avec le suiveur. De ce fait, le leader transmet sa vitesse actuelle au suiveur qui applique la même.

(Le wifi fonctionne, veillez à bien apparayer les robots ensemble)

```
def wifi(distance):
    # This is the name of the remote EV3 or PC we are connecting to.
    SERVER = "ev3dev"

    # Instanciate this robot as a Client
    client = BluetoothMailboxClient()
    mbox = TextMailbox("SpaceTab", client)

    print("Establishing connection...")
    client.connect(SERVER)
    print("Connected !")

    # Speed equals to LEAD robot's speed
    # If there is a communication message, it will use all_or_none policy
    if mbox.read() == None:
        speed = all_or_none(distance)
    else: speed = int(mbox.read())

    return speed
```

Ceci nous donne la courbe vitesse / distance suivante :

Conclusion