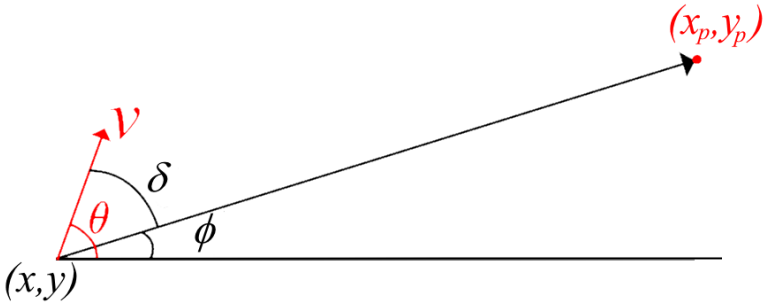
# Navigation autonome

## 1 Turtlesim

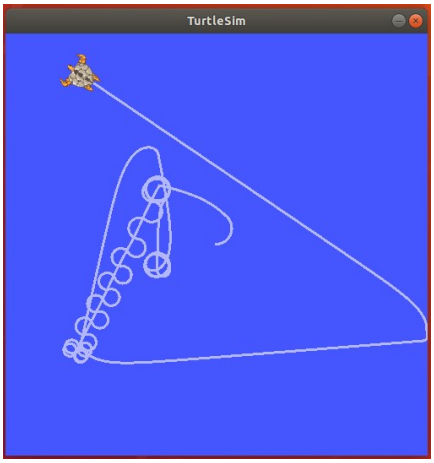
1. Dans le nœud **talker** programmé dans le TP d'introduction à ROS, ajouter un subscriber à la pose de la tortue (/**turtle1**/**pose**), celle-ci est équivalente à une odométrie donnant la position et l'orientation du robot (message **turtlesim**/**Pose**). Pour cela, s'inspirer du code suivant (extraire uniquement les lignes indispensables, ne pas créer un nouveau .py dans le dossier script ou .cpp dans le dossier src)
   Python : `https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/listener.py`
   C++ : `https://raw.githubusercontent.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/listener/listener.cpp`
2. Ajouter dans le programme un code de régulation vers une référence spatiale $(x_p, y_p)$.
   On note $(x, y)$ la position en coordonnées cartésiennes et $\theta$ l'orientation par rapport à l'horizontale. Le modèle dynamique à temps discret du robot s'écrit comme :

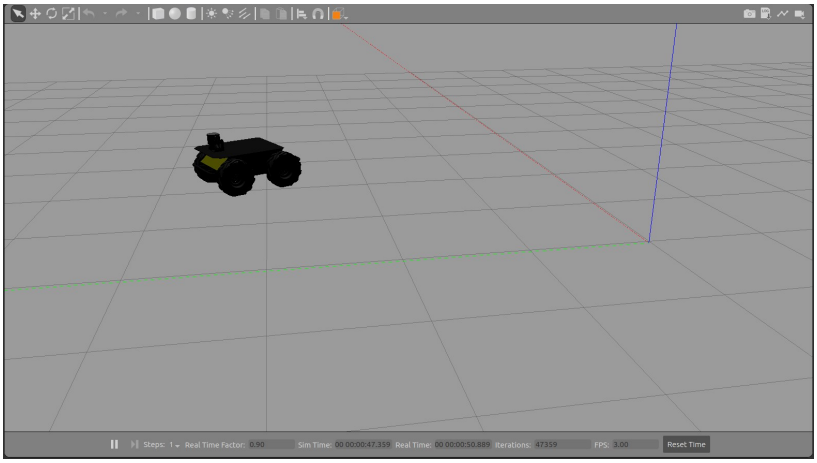
$$\begin{cases} x_{k+1} = x_k + \Delta t \cdot v_k \cdot \cos(\theta_k) \\ y_{k+1} = y_k + \Delta t \cdot v_k \cdot \sin(\theta_k) \\ \theta_{k+1} = \theta_k + \Delta t \cdot \omega_k \end{cases} \tag{1}$$

   Choisir une vitesse $v_k = v_{\text{ref}}$ à une valeur désirée, l'annuler lorsque la distance au point devient inférieure à un seuil donné. La vitesse angulaire $\omega_k$ sera calculée afin d'aligner le vecteur vitesse du robot (son axe principal) sur la ligne de vue, c'est-à-dire le vecteur entre le point courant $(x, y)$ et le point à atteindre $(x_p, y_p)$.




   Le modèle dynamique est donné par $\theta_{k+1} = \theta_k + \Delta t \cdot \omega_k$ et l'on souhaite asservir $\theta$ sur la référence $\phi$. La commande par retour d'état est donc $\omega_k = -k_p \cdot \delta_k$, avec $k_p$ un gain à régler et $\delta_k = \theta_k - \phi_k$ (pour calculer $\phi_k$, utiliser les coordonnées cartésiennes $x, y, x_p, y_p$). Attention à la gestion du modulo : l'écart angulaire $\delta$ doit être maintenu entre $-\pi$ et $\pi$ pour assurer la régulation à la valeur 0 le long de la ligne de vue. Ecrire ces deux valeurs dans le message à publier sur **/turtle1/cmd_vel.**
3. Ajouter un autre subscriber pour pouvoir indiquer une nouvelle référence $(x_p, y_p)$ via la publication sur un topic. Utiliser pour cela un message de type **geometry_msgs**/**Point**.
4. Faire varier le couple de paramètres {gain $k_p$, timer du noeud} afin de reproduire les différents comportements possibles de la boucle fermée en temps discret.



# 2 Robot mobile Husky

## 2.1 Navigation autonome

Le nœud **talker** développé pour déplacer Turtlesim peut être réutilisé (en faire une copie) pour commander le robot mobile Husky présenté dans le TP d'introduction à ROS.
Cela nécessite les modifications suivantes :

- le topic de localisation est maintenant /**odometry**/**filtered** (de type **nav_msgs**/**Odometry**). Pour transformer les composantes d'un quaternion $q[x, y, z, w]$ en angle de lacet $\psi$ (yaw), utiliser la formule suivante : $\psi = \text{atan2}\left(2 \cdot q.w \cdot q.z, 1 - 2 \cdot q.z \cdot q.z\right)$.
- la commande publiée est maintenant sur le topic /**husky_velocity_controller**/**cmd_vel** (même type que précédemment).

Ces informations peuvent être obtenues via la commande **rostopic list**, ou via l'interface graphique **rqt**.

## 2.2 Suivi de chemin

Ajouter la possibilité de souscrire à un message **nav_msgs**/**Path** composé de points de passage à valider successivement par le correcteur.
Le visualiser également dans **rviz**. Il faut pour cela avoir renseigné un nom de repère ('world' ou 'map' par exemple) dans le paramètre **frame_id** du champ **header** du message Path publié.