

Tarea 1: Configuración de parámetros de cinemática inversa y controlador

Materiales

Cada pareja de alumnos tendrá asignado un *duckiebot* y un equipo con el *software* necesario para interactuar con él.



Caja duckiebot

Tarea 1.1 - Cinemática

La idea es en primer lugar ajustar los parámetros referentes a la cinemática del robot, es decir, a cómo se mueve. En concreto son: ##### Desvío (*trim*) En teoría si los dos motores son iguales y reciben la misma entrada (tensión), deberían moverse ambos a la misma velocidad y por tanto el robot circular en línea recta. En la práctica esto no es así y hay que aplicar un desfase entre la entrada que se le da a cada motor. ##### Ganancia (*gain*) La señal de control generada para controlar el robot es una señal entre 0 y 100%, que se traduce en una tensión aplicada a los dos motores (p.ej.: $G=1$, 100% \rightarrow 5V | , 100% $-(G=0.5)\rightarrow$ 2.5V). El escalado entre la señal de control y la tensión se especifica con el parámetro *gain* que por defecto es 1 y se puede hacer mayor o menor según sea necesario.

De manera premeditada se han desconfigurado los parámetros de **desvío** y **ganancia**, y por lo tanto el robot no se maneja bien. El objetivo de esta tarea es conseguir un *duckiebot* que circule en línea recta cuando así se le pida y que manualmente sea sencillo de controlar por la ciudad sin derrapar.

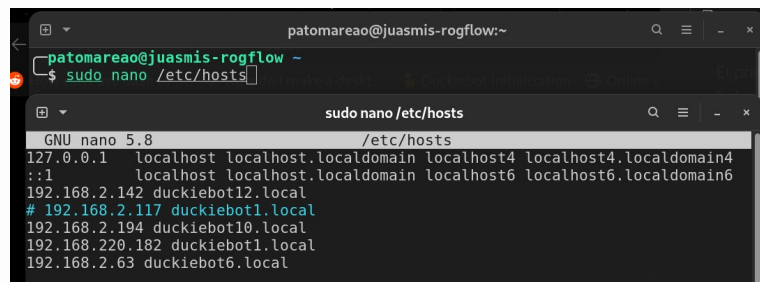
Para ello se ejecutará el programa que permite el control manual del vehículo, además de un programa que genera un mando para controlarlo.

Pasos a seguir

El primer paso es encender el robot conectando los dos cables a la batería. Automáticamente se encenderán las luces, si no es así pulsar el botón en el lateral de la batería. Tras un tiempo o cuando se apaguen las luces del robot probar si hay comunicación con la *Raspberry* haciendo: `ping duckiebotX.local`. Si no funciona introducir directamente la dirección IP en lugar del nombre de *host*.

NOTA: Sustituir en todos los comandos donde se indica `duckiebotX`, la X por el número del *duckiebot* usado.


NOTA: Si no funciona así, hay que hacerlo introduciendo la dirección IP que se mostrará en clase, y además añadir una nueva entrada en el archivo `/etc/hosts` como se muestra:



```
patomareao@juasmis-rogflow:~  
$ sudo nano /etc/hosts  
GNU nano 5.8 /etc/hosts  
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
192.168.2.142 duckiebot12.local  
# 192.168.2.117 duckiebot1.local  
192.168.2.194 duckiebot10.local  
192.168.220.182 duckiebot1.local  
192.168.2.63 duckiebot6.local
```

Edición de `/etc/hosts`

Pulsar `CTRL+X` para salir y confirmar para guardar.

A continuación, a través de la interfaz gráfica de *Docker* (*Portainer*), se pueden acceder a los distintos contenedores previamente creados (en un navegador en la barra de búsqueda introducir `duckiebotX.local:9000`):  Debe haber un contenedor llamado `practica_laboratorio`, se inicia y accede a él pulsando sobre el icono: `>_`.

Si no se encuentra disponible, primero descargar la última versión de la imagen:

```
docker -H duckiebotX.local pull  
patomareao/duckietown_ual:practicaLab_seed5
```

Posteriormente iniciarlo con:

```
docker -H duckiebot1.local run -it --net host --memory="800m" --  
memory-swap="2.8g" --privileged --name clase_practica_1 -t -i  
patomareao/duckietown_ual:practicaLab_seed5
```

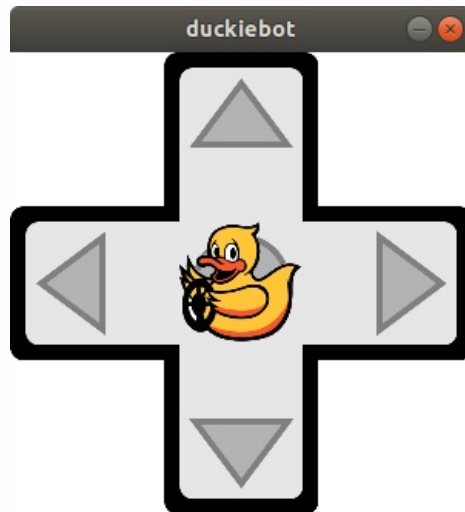
Este contenedor se ha creado a partir de una imagen que contiene todos los programas disponibles para un *duckiebot* (control manual, seguidor de líneas, navegación indefinida, etc). Estos son *scripts* programados en *Python* que se ejecutan mediante el Sistema Operativo para Robots (ROS, *Robot Operating System*). El programa de interés para esta tarea es el del control manual, para ejecutarlo se introduce:

```
roslaunch duckietown joystick.launch veh:=duckiebotX
```

Después en una terminal en el ordenador de sobremesa se abre el mando, para ello se usa la siguiente instrucción:

```
dtc duckiebot keyboard_control duckiebotX
```

Resultado esperado:



Resultado esperado

*Si esto no funciona consultar final del manual para forma alternativa de generar mando.

Para configurar los parámetros se abre en una nueva ventana del navegador el contenedor practica_laboratorio y se ejecutan las siguientes instrucciones:

```
rosservice call /duckiebotX/inverse_kinematics_node/set_gain --  
VALOR
```

```
rosservice call /duckiebotX/inverse_kinematics_node/set_trim --  
VALOR
```

Donde con `set_gain` se especifica la ganancia y con `set_trim` el desvío. Ir sustituyendo VALOR por los distintos valores de prueba comprendidos entre 0 y 10 comprobando el desempeño moviendo el *duckiebot* por la ciudad.

Una vez encontrados valores que generan un desempeño satisfactorio guardar con:

```
rosservice call /duckiebotX/inverse_kinematics_node/save_calibration
```

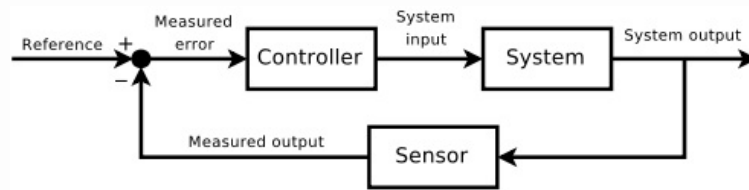
Resultado esperado: un duckiebot que sigue una trayectoria recta cuando sólo se pulsa el botón de avanzar en el mando, y que manualmente se pueden seguir las curvas sin derrapes o pérdidas de control:



Resultado esperado

Tarea 2.2 - Controlador

El otro grupo de parámetros son los referentes al controlador implementado para mantener el robot dentro del carril. Hay implementados dos controladores que forman de manera conjunta, uno que controla la posición dentro del carril y otro que controla el ángulo del *duckiebot*. En este caso se intentará ajustar el control desplazamiento dentro del carril. Se trata de un controlador PI (**P**roporcional - **I**ntegral).



Closed loop diagram

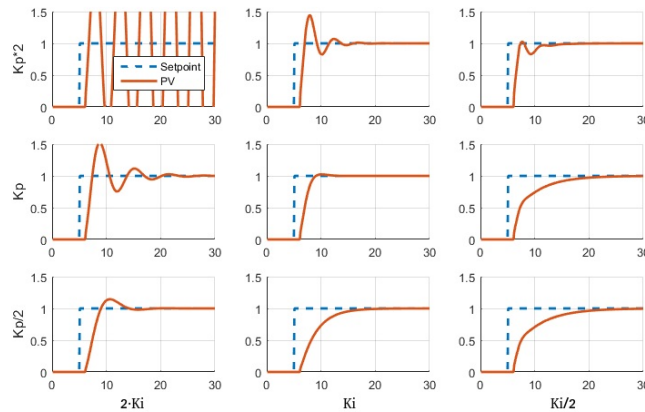
Por lo tanto los parámetros son la ganancia proporcional e integral: - Ganancia proporcional (K_p). La parte proporcional de la acción de control se calcula como el producto entre el error (diferencia entre salida actual y salida deseada) y la ganancia proporcional:

$$P = K_p \cdot e(t)$$

- Ganancia integral (K_i). La parte integral de la acción de control se calcula como la integral del error en el tiempo. El resultado de esto es multiplicado por la ganancia integral.

$$I = K_i \int_0^t e(t) dt$$

La señal de control es la suma de ambas $U = P + I$



Salida en función de parámetros

Pasos a seguir

Detener el programa de control manual (ejecutado desde *Portainer*) pulsando CTRL + C, y ejecutar el programa de seguimiento de líneas:

```
roslaunch duckietown_demos lane_following.launch veh:=duckiebotX
```

Una vez se vea que se está ejecutando (cuando se muestran mensajes repetitivos en la consola), desde el mando pulsar la letra **A** del teclado para activar la circulación en modo autónomo y **S** para volver al modo manual. En función del comportamiento modificar los parámetros del contenedor de manera similar a la anterior, es decir, en una terminal distinta ejecutar:

```
rosparam set /duckiebotX/lane_controller_node/k_d -- VALOR
```

Para ajustar la ganancia proporcional, y:

```
rosparam set /duckiebotX/lane_controller_node/k_Id -- VALOR
```

Para la ganancia integral. Una forma de determinar si la ganancia integral es lo suficientemente grande, es, si cuando el *duckiebot* se encuentra pisando alguna de las líneas es muy pasivo (tarda mucho en corregir la trayectoria) en volver al centro de carril, entonces hay que aumentar la ganancia. La ganancia proporcional se ve por los cambios inmediatos que hace el *duckiebot* frente a errores, se ve por la intensidad instantánea que hace el vehículo para volver al centro del carril.

Resultado esperado: un *duckiebot* que es capaz de navegar satisfactoriamente por la ciudad siguiendo las marcas de la carretera:



Información adicional

- Documentación del proyecto: [link](#)
- Repositorio en GitHub de práctica: [link](#)

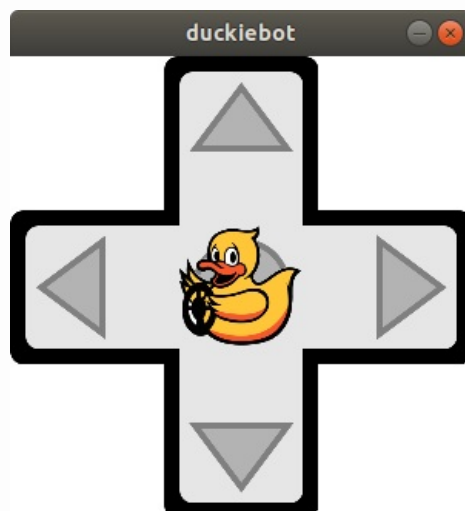
Forma alternativa de generar mando

Desde una terminal cambiar al directorio raíz donde están todos los programas del *duckiebot* clonados en el ordenador de sobremesa y ejecutar la instrucción `make`:

```
cd Software
```

```
make virjoy-duckiebotX
```

Resultado esperado:



Mando

Comando para meterse en un contenedor ya en funcionamiento y ver mensajes

```
docker -H duckiebotX.local attach clase_practica2
```

Iniciar nueva terminal en contenedor en ejecución

```
docker -H duckiebotX.local exec -it clase_practica2 bash
```

Ver valor actual de parámetros

```
rosparam get /direccion/al/nodo
```