Archivo Leeme – Taller 2 Robótica

Grupo 4

Bienvenido! El siguiente archivo te va permitir ejecutar los códigos asociados a este taller. En primer lugar se asumirá que para propósitos de estas instrucciones ya se tiene todas las dependencias de ROS y de Arduino instaladas y, por tanto, basta con compilar los archivos .INO y ejecutar los archivos .py para poder realizar lo demás. En general las dependencia usadas para el código de Python son:

```
import rospy
from gpiozero import LED
from std_msgs.msg import String, Int16
import pickle
import serial
import time
import numpy as np
from scipy import integrate
import pickle
import matplotlib.pyplot as plt
```

Mientras por su parte el código de Arduino utiliza las siguientes dependencias:

#include <LEANTEC_ControlMotor.h> //Incluimos la librería control de motores

#include <ros.h>

#include <std_msgs/XXX.h>

#include <string.h>

#include "TimerOne.h"

En donde XX.h corresponde a diferentes formatos de mensajes. La librería de std_msgs viene incluida en el módulo de ROS serial para Arduino junto con ros.h. No obstante, en caso de ser la primera vez que se descarga hay que tener presente que en lo archivos de la librería, en particular en msg.h hay que cambiar un par de líneas de código para que funcione:

```
Linea 40 -> #include <string.h> #Eliminar cstring
Linea 68 -> memcpy(&val, &f, sizeof(val)); #Eliminar std::
Linea 182 -> memcpy(f, &val, sizeof(val));
```

Después de realizado estas modificaciones y haber instalado todas las dependencias, se puede comenzar la ejecución. Para esto, se asumirá el nombre del paquete de ROS como robotica_pkg pero en principio cualquier nombre debería funcionar. Así, en la carpeta scripts del paquete que se tenga creado se debe hacer el push del repositorio de github donde se encuentran los códigos como:

git init

git clone https://github.com/lmgamba/Taller2Robotica_Grupo4

Una vez con esto, en una terminal diferente se puede iniciar el Roscore y la ejecución del taller. Para el primer punto se utilizaron los esquemas de comunicaciones ROS serial para comunicar la raspberry con el Arduino, por lo que en una terminal aparte se debe correr el script

rosrun rosserial_python python_node.py /dev/ttyACM*

En donde * puede ser un número entre 0 y 3. En general la definición del parámetro /dev/ttyACM* corresponde al puerto en el que está conectado el Arduino a la Raspberry, por lo que es necesario iterar entre los valores de /dev/ttyACM0, /dev/ttyACM1. /dev/ttyACM2 y /dev/ttyACM3 hasta encontrar el puerto que se está usando ya su asignación virtual, aún para el mismo puerto físico, tiene un número diferente cada vez que desconecta el Arduino de la Raspberry.

Punto 1:

Una vez se hayan importado los tópicos de ROS desde el Arduino con el rosserial_python y se esté ejecutando Roscore necesitará otras dos consolas. En ambas, en el workspace principal primeramente se debe realizar la actualización de los directorios del paquete, es decir, ejecutar:

catkin_make

source devel/setup.bash

Posteriormente se deben ejecutar los scripts del Publisher y el Subscriber del primer punto. Para el caso de la consola del Publisher debe correr

rosrun robótica_pkg pub_punto_1.py

Se le dará la bienvenida en consola y se le solicitarán dos datos asociado a la velocidad lineal y angular del robot. A la larga, estos datos serán esenciales para la estimación de la velocidad de las ruedas del robot y, con dicha información, calcular entre el rango de 0 a 255 el pulso PMW necesario para ejecutar dicho valor . Poteriormente, se le solicitará en interfaz que ingrese alguna orden. Hay 5 ordenes posibles:

w: Ir hacia adelnte

s: Ir hacia atrás

a: Ir hacia la izquierda

d: Ir hacia la derecha

p: Detenerse

exit: Finalizar ejecución del Publisher

Note que todas las ordenes deben ser ingresadas a consola en minúscula, de lo contrario se tomará como una orden “p” por defecto y el robot no se moverá. Antes de ingresar cualquier orden asegúerese de haber corrido en otra terminal el Suscriber, es decir, el comando:

rosrun robótica_pkg sus_punto_1.py

Se le dará la bienvenida nuevamente. Es fundamental que presione [ENTER] en dicha consola para inicializar el Susbcriber y con el todo el protocolo de comunicaciones para enviar la información y recibirla de Arduino. Una vez con esto, puede ejecutar todas las ordenes que requiera a voluntad hasta finalizar la ejecución del código.

ADVERTENCIA: Valores de velocidad muy bajos requieren de tensiones altas en el punte H para poder mover al robot. En caso de que la pila se esté descargando, es probable que estas bajas velocidades no se puedan ejecutar a cabalidad por limitaciones en el Hardware.

Punto 2

Para el segundo punto se va a mantener el mismo código de Publisher corriendo, es decir, se va a publicar sobre los mismos tópicos. Por generalidad, cierre las consolas del punto anterior y abra nuevamente otras dos, en la primera realice el mismo procedimiento para configurar el Publisher el punto 1, es decir, corra en el workspace

rosrun robotica_pkg pub_punto_1.py

Y finalmente ingrese los datos requeridos como la velocidad lineal y la velocidad angular a la que debe ir el Robot. Y a su vez, previo a comenzar a ingresar instrucciones, corra el nodo Subscriber. Para este caso, se debe correr el código

rosrun robótica_pkg punto_2.py

A diferencia del subscriber del punto anterior, este código es capaz de estimar la posición en X y en Y del robot tanto con la información obtenida de los Encoders como partiendo de los parámetros ingresados por el usuario inicialmente. Si se encuentra un GUI se graficarán estos datos. Sino, solo se mostrarán dos listas con la distancia en cm con respecto al eje global de referencia (punto desde el cual se inició el recorrido).

Punto 3

Nuevamente cierre las dos consolas usadas previamente. Para el almacenamiento de un recorrido en principio se podría utilizar cualquier subscriber definido previamente, pero por simplicidad se va a utilizar el del primer punto. Así, se abren dos consolas diferentes y en una de ellas se corre el comando:

rosrun robotica_pkg sus_punto_1.py

Y se presiona el [ENTER] para activar el suscriber. Por otro lado, en otra consola se correo el archivo:

rosrun robotica_pkg punto_3.py

En este archivo en principio se le pregunta al usuario si desea guardar el recorrido que efectúe el robot. En caso de recibir una respuesta afirmativa (y) entonces se le pide al usuario que ingrese el nombre del recorrido que pretenden realizar y así se ejecuta todo normalmente. En general se realizan dos tomas de datos, una asociada al tiempo y otra asociada a las ordenes en las que se ejecutaron. De este modo es posible repetir el recorrido en caso de que se necesite.

Punto 4

Finalmente, para el uso del servicio es esencial realizar cambios a los archivos del paquete de ROS, como el package.xml. Estos pasos no serán descritos en este Readme en la medida que se supone el lector ya realizó los mismos y solo se hará mención de aquellos necesarios para la ejecución del servicio desde el trabajo realizado por nuestro equipo de desarrollo. Así, se corre en primer lugar una consola con el código que contiene el servicio, es decir:

rosrun robotica_pkg punto_4_srv.py

Así, en otra consola ya se puede realizar el llamado del mismo. Con esto, se puede invocar en otra consola desde el workspace el servicio denominado recorrido_predefinido. Así, se puede invocar y correr usando en otra consola el comando

rosservice call /recorrido_predefinido #Nombre_del _recorrido #Velocidad_lineal #Velocidad_angular

Y así se puede volver a recorrer el mismo.