Robot Operating System - ROS

Leslie P. Brenes, Diego Villafuerte

Resumen—En esta práctica se aprendieron los comandos básicos en ROS, un meta-sistema operativo para robots. [1] También se conectó ROS con un arduino para aprender el funcionamiento de tópicos y suscripciones.

1. INTRODUCCIÓN

ROS es un sistema meta-operativo de código abierto para controlar robots, y es un requisito generar el proyecto final utilizando esta herramienta. Por ello, es importante aprender el funcionamiento del código así como los conceptos técnicos básicos necesarios.

Con ello en mente, primero se descargó un repositorio proporcionado por el profesor para aprender la inicialización, compilación y probar la funcionalidad básica del sistema de tópicos y suscripciones. Posteriormente, se creó un programa básico llamado Turtlesim cuyo objetivo fue mandar instrucciones de movimiento. Finalmente, se abordó el control de un programa a través de Arduino (para enviar y recibir mensajes) y mediante una segunda computadora.

2. MARCO TEÓRICO

ROS proporciona la funcionalidad del paso de mensajes entre procesos y la administración de paquetes, así como las herramientas y bibliotecas para obtener, crear, escribir y ejecutar código en varios equipos.

Implementa varios estilos diferentes de comunicación, incluida la comunicación a través de servicios sincrónica de RPC (Remote Procedure Call, donde utiliza una computadora para ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas), la transmisión asíncrona de datos sobre temas y el almacenamiento de datos en un servidor de parámetros. [1]

Las áreas que incluye ROS son:

- Un nodo principal de coordinación.
- Publicación o subscripción de flujos de datos: imágenes, estéreo, láser, control, actuador, contacto, etc.
- Multiplexación de la información.
- Creación y destrucción de nodos.
- Los nodos están perfectamente distribuidos, permitiendo procesamiento distribuido en múltiples núcleos, multiprocesamiento, GPUs y clústeres.
- Login.
- Parámetros de servidor.
- Testeo de sistemas.

3. DESARROLLO

Dentro del Repositorio podemos encontrar los archivos de ejemplo para la implementación de tópicos y suscripciones.

El primer paso fue editar los archivos para poder generar la comunicación bilateral y que el tipo de respuesta aceptada fuera de tipo String.

Posteriormente, pasamos a la parte de Turtlesim. En la que primero tuvimos que familiarizarnos con el uso de mensajes de tipo `geometry_msgs::Twist`. Con ello, pudimos hacer manipulaciones en la Tortuga para que se moviera de manera específica con ciertas teclas.

Finalmente vino la parte de Arduino, bajamos la libreria necesaria donde usando la librería rosserial se generó la comunicación ROS - Arduino. El objetivo era hacer distintas manipulaciones para prender un LED.

En la úlimta sección, se buscó la comunicación entre dos computadoras para que una manipulara a la otra mediante Turtlesim.

4. RESULTADOS

Primero modificacmos el código del ejemplo.

```
#include <ros/ros.h>
#include <iostream>
#include <std_msgs/Int32.h>
#include <std_msgs/String.h>

#define RATE_HZ 2

using namespace std;
int num = -1;

void get_msg(const std_msgs::Int32& msg) {
        num = msg.data;
        ROS_INFO_STREAM("Valor recibido: "
        << num);
}

int main(int argc, char **argv)
{
        ros::init(argc,argv,
        "ejemplo_sub_node");
        ros::NodeHandle nh;
        ROS_INFO_STREAM("ejemplo_sub_node
         initialized");
        ROS_INFO_STREAM(
```

```
        ros::this_node::getName());

        ros::Subscriber sub_vel =
        nh.subscribe(
        "/msg_ejemplo", 1000, &get_msg);
        ros::Rate rate(RATE_HZ);

        ros::Publisher pub = nh.advertise
        <std_msgs::String>
        ("/msg_respuesta", 1);
        std_msgs::String msg;
        while (ros::ok())
        {

                ROS_INFO_STREAM(".");

                ros::spinOnce();

                msg.data = "Recib
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣tu␣mensaje";
                if(num!=-1){
                        pub.publish(msg);
                        num = -1;
                }


                // ros::spin();
                rate.sleep();
        }

    return 0;
}
```

Para la sección de Turtlesim, generamos el siguiente código:

```
ros::Publisher pub =
nh.advertise<geometry_msgs::Twist>
("turtle1/cmd_vel", 10);

//ros::Subscriber sub =
nh.Subscribe("turtle1/pose",10,
poseCallback);

geometry_msgs::Twist msg;

msg.angular.x=2;
msg.angular.y=4;
msg.angular.z=2;
int ms;
//ros::RATE rate(10);
ROS_INFO("COMINO");
while (ros::ok())
{
cout << " Movimiento ?" << endl;
cin >> ms;
switch(ms){
        case 2:
            msg.linear.x = FORWARD_SPEED_MPS;
                msg.angular.z = GIRO;
        break;
        case 4:
                msg.angular.z = 0;
                msg.linear.x = -FORWARD_SPEED_MPS;
        break;
        case 6:
                msg.angular.z = 0;
                msg.linear.x = FORWARD_SPEED_MPS;
        break;
        case 8:
                msg.angular.z = -GIRO;
                msg.linear.x = FORWARD_SPEED_MPS;
        break;
        }
pub.publish(msg);
ros::spinOnce();
//rate.sleep();
}

return 1;
}
```

Para la sección de ROS-arduino no pudimos obtenerla debido a problemas con las librerías y la instalación de Arduino en las computadoras del laboratorio. Adicionalmente, tuvimos una limitante de tiempo que no nos permitió intentarlo en otra computadora. Este problema no se presentó en otros equipos del laboratorio. Para futuras ocasiones sería importante revisar las configuraciones de ROS y Arduino en la computadora antes de comenzar a programar.

5. CONCLUSIONES

- Leslie: Lo más importante de esta práctica fue aprender a usar ROS y entender como funcionan los tópicos y suscripciones, así como los comandos básicos para ejecutar el código. Creo que ROS requiere una configuración complicada, pero la práctica nos sirvió para experimentar, y aclarar dudas previas al desarrollo del proyecto final.
- Diego: En esta práctica pudimos experimentar con ROS y aprender a configurarlo, pudimos experimentar con las librerías de ROS y Arduino pero desafortunadamente no pudimos ver la interacción por errores de instalación.

6. ROL O PAPEL

- Leslie: Investigación, reporte y corrección de código.
- Diego: Código y configuración.

7. FUENTES CONSULTADAS

REFERENCIAS

[1] "Ros." [Online]. Available: http://wiki.ros.org/ROS/Introduction