



Tecnológico de Monterrey

**Instituto Tecnológico y de Estudios
Superiores de Monterrey**

TE3001B.101

Fundamentación de robótica (Gpo 101)

Semestre: Febrero - Junio 2022

Reto semanal 3 Manchester Robotics

Alumnos:

José Jezarel Sánchez Mijares

A01735226

Antonio Silva Martínez

A01173663

Frida Lizett Zavala Pérez

A01275226

Fecha de entrega: 10 de Marzo del 2023

Reto semanal 3. Manchester Robotics

Resumen

Este mini reto está destinado a que el alumno repase los conceptos introducidos en las sesiones anteriores. La actividad consiste en crear varios nodos ROS para regular la velocidad de un Motor DC. El motor se controla mediante una computadora externa, un microcontrolador y un controlador de motor.

Introducción

La comunicación entre ROS (Robot Operating System) y Arduino es fundamental para el desarrollo de sistemas robóticos complejos. ROS es una plataforma de software de código abierto que proporciona una serie de herramientas y bibliotecas para la programación de robots, mientras que Arduino es una plataforma de hardware de bajo costo y fácil acceso que se utiliza comúnmente para el prototipado rápido de sistemas electrónicos.

La combinación de ROS y Arduino permite a los desarrolladores de robots aprovechar las ventajas de ambas plataformas para construir sistemas robóticos más avanzados y complejos. La comunicación entre estas dos plataformas permite a los desarrolladores enviar y recibir datos entre el software y el hardware, lo que resulta en un control más preciso y eficiente de los robots. Además, la comunicación también permite la integración de sensores y actuadores en el sistema, lo que permite una mayor percepción y capacidad de acción por parte del robot.

Objetivos

Mantener una comunicación serial entre ros y arduino para conectarlo con el microcontrolador que a su vez nos permitirá mover el motor, para este ejercicio no se aplicará control, sin embargo desde la terminal se debe poder cambiar los parámetros para alterar la dirección del motor de 1 a -1 esto también nos indicará la velocidad con la que el motor se moverá

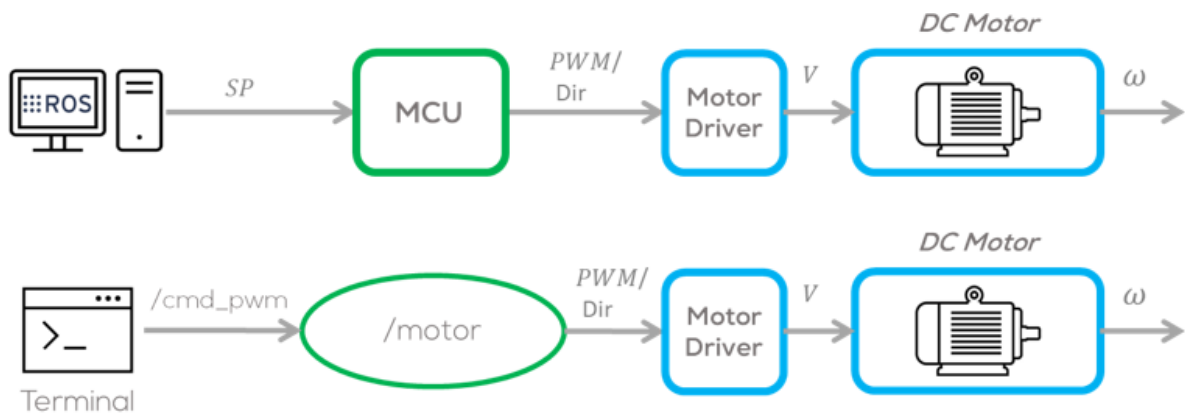
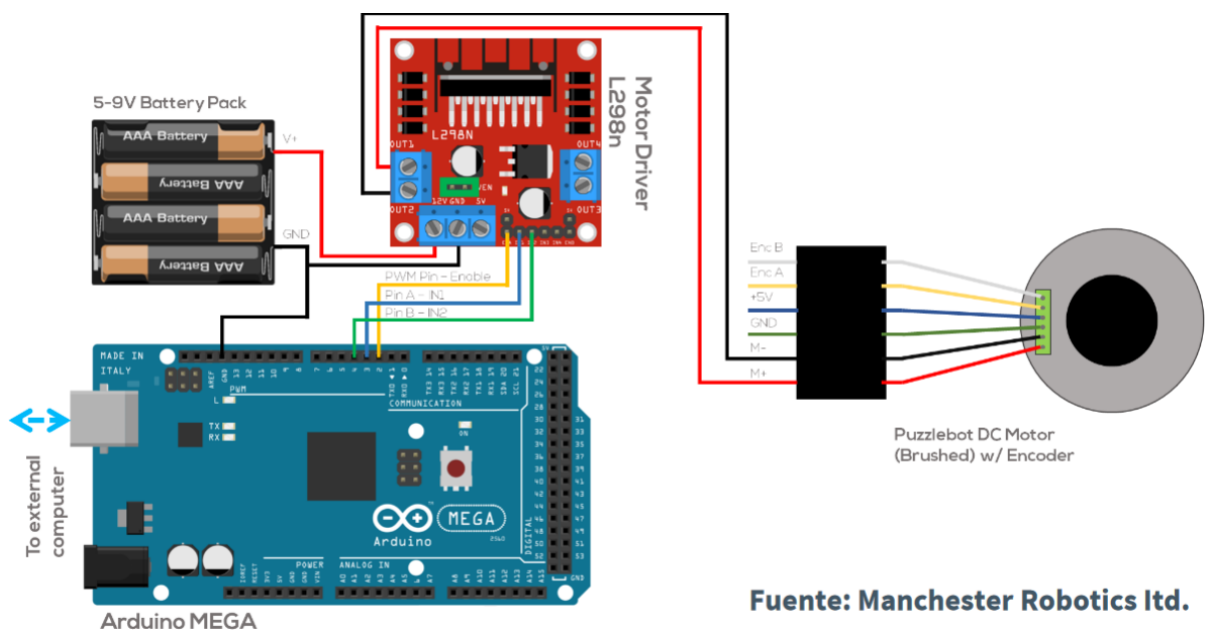


Diagrama de conexiones



Solución del problema

Creamos el nodo Input donde colocamos dos señales tanto la cuadrada como la señal senoidal, estas señales serán controladas desde la terminal modificando los parámetros, como parámetros colocamos la fase, el tipo de señal, la amplitud, offset, y tope y fondo para la cuadrática

```

#!/usr/bin/env python

import rospy

import numpy as np

from std_msgs.msg import Float32
  
```

```

tipo =3

def stop(self):

    pub.publish(0)

    rate.sleep()

    print("stopping")

if __name__=='__main__':

    rospy.init_node("Input")

    rate= rospy.Rate(30)

    rospy.on_shutdown(stop)

    pub = rospy.Publisher("cmd_pwm", Float32,queue_size=10)

    print("The set point generator is running")

    t0= rospy.Time.now().to_sec()

    #Run Node

    while not rospy.is_shutdown():

        tipo=rospy.get_param("tipo",2)

        if tipo== 2:

            P=rospy.get_param("P",21)

            phase= rospy.get_param("phase",0)

            amplitud= rospy.get_param("amplitud",1)

            offset= rospy.get_param("offset",0)

            w=2*np.pi/P

            t= rospy.Time.now().to_sec()-t0

```

```

        timeset=t

        setout=(np.sin(w*t+phase)*amplitud)+offset

elif tipo==3:

    P=rospy.get_param("P",21)

    phase= rospy.get_param("phase",0)

    amplitud= rospy.get_param("amplitud",1)

    offset= rospy.get_param("offset",0)

    w=2*np.pi/P

    t= rospy.Time.now().to_sec()-t0

    if(np.sin(w*t+phase)*amplitud)>=0:

        setout=rospy.get_param("tope",1)

        timeset=t

    else:

        setout=rospy.get_param("fondo",-1)

        timeset=t

#publicar

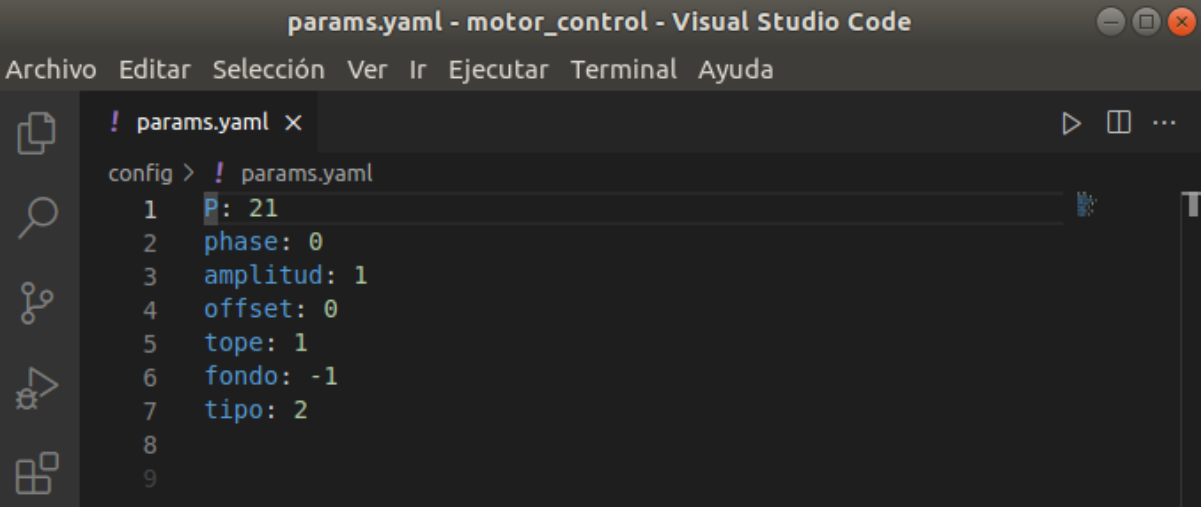
rospy.loginfo(setout)

pub.publish(setout)

```

nuestro topic donde el arduino estará suscrito será cmd_pwm.

Para completar nuestro paquete agregamos un folder de config donde colocamos los parámetros que toma nuestro nodo *Input*



```
params.yaml - motor_control - Visual Studio Code
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

! params.yaml x
config > ! params.yaml
1 P: 21
2 phase: 0
3 amplitud: 1
4 offset: 0
5 tope: 1
6 fondo: -1
7 tipo: 2
8
9
```

Para finalizar este proceso de comunicación entre ROS y arduino creamos el código .ino que nos permita usar el motor con encoder

```

#include <ros.h>
#include <std_msgs/Float32.h>

ros::NodeHandle nh;

const int In1 = 2; // Analog output pin
const int In2 = 3; // Analog output pin
const int EnA = 11; // Activar o desactivar Puente H

const int Vcc = 5;
const int Vcc_sc = Vcc/255;

float voltage,duty;
int pwm = 0;

float sgn_ros;

void loop() { nh.spinOnce(); delay(1); }

void pulse ()
{
    pwm = abs(sgn_ros*255);
    voltage = pwm * Vcc_sc;
    duty = 100*voltage/Vcc;

    analogWrite(EnA, pwm);
    print_data();
}

void print_data()
{
    Serial.print("EL ciclo de trabajo del pwm es: ");
    Serial.print(duty);
    Serial.print(" %");
    Serial.print("    EL cual corresponde a:  ");
    Serial.print(voltage);
    Serial.println(" Volts" );
}

```

finalmente se corre este código en arduino y se sube a la terminal, después en la terminal encendemos el roscore conectamos el arduino con la terminal para que empiece a recibir los datos después en la terminal imprimimos el topic al que está suscrito el arduino, en otra terminal corremos nuestro nodo y finalmente en otro terminal mediante el cambio de parámetros podemos cambiar el comportamiento del sistema

Resultados

Video demostrativo de la comunicación: <https://youtu.be/qG9UvpMGoHQ>

Conclusiones

La comunicación entre ROS y microcontroladores es importante porque permite la integración de robots físicos con sistemas de control de alto nivel. ROS es un marco de trabajo de código abierto para robótica que proporciona una plataforma para el desarrollo de aplicaciones robóticas complejas. Mientras tanto, los microcontroladores son dispositivos electrónicos programables que se utilizan para controlar el comportamiento de los componentes físicos de un robot, como motores, sensores y actuadores.

Al conectar ROS con un microcontrolador, se puede utilizar la potencia de procesamiento de ROS para realizar tareas de control de alto nivel, mientras que el microcontrolador se encarga de las tareas de bajo nivel necesarias para controlar los componentes físicos del robot. Esto permite que los robots sean más inteligentes y capaces de realizar tareas más complejas.

Además, la comunicación entre ROS y microcontroladores permite la recopilación y el análisis de datos de sensores en tiempo real, lo que permite tomar decisiones informadas sobre el comportamiento del robot. También permite el control remoto del robot, lo que es esencial en aplicaciones como la exploración espacial y la inspección de instalaciones peligrosas.

Referencias

1. ROS Wiki. (2023). recuperado el 27 de febrero del 2023, desde <http://wiki.ros.org/es>
2. ROS: Home. (2023). recuperado el 27 de febrero del 2023, desde <https://www.ros.org/>
3. Quigley, M., Gerkey, B., & Smart, W. D. (2015). Programming robots with ROS: A practical introduction to the Robot Operating System. O'Reilly Media, Inc.