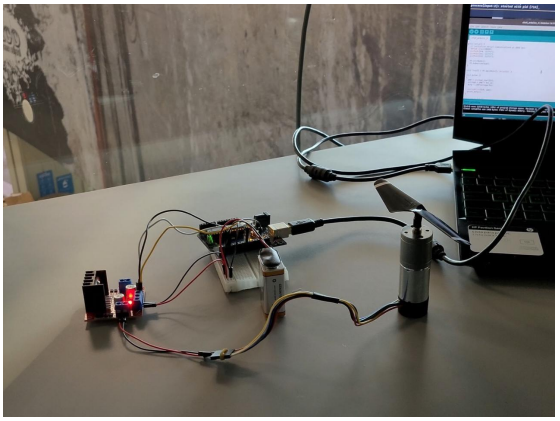
Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

Fundamentación de Robótica - TE3001B GPO (101)

Challenge 03



Jonathan Josafat Vázquez Suárez	**A01734225**
Jhonatan Yael Martinez Vargas	**A01734193**
Diego García Rueda	**A01735217**

Profesor

Rigoberto Cerino Jiménez

Fecha de Entrega: 28 de Febrero del 2023

Objetivos

Con este challenge se espera que nosotros podamos familiarizarnos con el concepto y uso de las señales PWM para controlar el giro y velocidad de un motor con encoder de cuadratura, al poder integrar todo esto en un mismo sistema interconectado por ROS y arduino.

Solución

En este mini reto se espera poder realizar la interconexión de un dispositivo arduino con ROS y poder controlar la velocidad y sentido de giro de un motor , usando una señal PWM. Por lo que lo primero que hay que tomar en cuenta es que dentro del código de arduino es necesario incluir las librerias ***ros.h*** *y* ***std_msgs,*** esto con la finalidad de indicarle a arduino que se va a trabajar con ROS, posteriormente hay que seguir la lógica de ***publishers*** y ***subscribers***, tanto para enviar datos como para recibir, y de esta manera queda configurada la parte del arduino.

Posteriormente dentro de nuestro script de python realmente no tenemos que aplicar cambio alguno, en cuestión de terminologías y aplicaciones de un nodo, en donde si tenemos que tener en cuenta algunos ajustes es dentro de archivo ***.launch*** pues es aquí donde tenemos que especificar que la información va a provenir de un puerto externo

```
<node name="motor" pkg="rosserial_python" type="serial_node.py">
    <param name="port"    type="string"       value="/dev/ttyACM0"/>
</node>
```

Código Arduino

```
#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 = 9; // 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 direction_fcn(const std_msgs::Float32& msg)
{
 sgn_ros = msg.data;
 Serial.println("DR");
 digitalWrite(In1, sgn_ros>0.01);
 digitalWrite(In2, sgn_ros<0.01);
 pulse();
}

ros::Subscriber<std_msgs::Float32> sub("cmd_pwm", direction_fcn);

void setup() {
 // initialize serial communications at 9600 bps:
 Serial.begin(9600);
 pinMode(EnA, OUTPUT);
 pinMode(In1, OUTPUT);
 pinMode(In2, OUTPUT);

 nh.initNode();
 nh.subscribe(sub);
}

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" );
}
```

Código Python

```
#!/usr/bin/env python
import rospy
import numpy as np
from std_msgs.msg import Float32

def triangle():

   global actual_value, direction, step, max_value, min_value

   actual_value += (direction*(-step)) or (step)
   if (actual_value > max_value):
       direction = True
   elif (actual_value < min_value):
       direction = False
   return actual_value

def cuadrada():

   global actual_value, direction, step, lon_1, lon_2

   if(actual_value <= 0):
```

```
        actual_value = (direction*(lon_1)) or (lon_2)
        direction =  not direction
        return 0.0

    actual_value -= 1

    return (direction*(1)) or (-1)


if __name__ == "__main__":

    global actual_value, direction, step, max_value, min_value

    actual_value = rospy.get_param("initial_pos", 0)
    direction = rospy.get_param("boolean", True)
    step = rospy.get_param("step", 0.01)
    lon_1 = rospy.get_param("lon_1", 5)
    lon_2 = rospy.get_param("lon_2", 15)
    max_value = rospy.get_param("max_value", 1)
    min_value = rospy.get_param("min_value", -1)
    f_s = rospy.get_param("function", "crd")

    pub_1 = rospy.Publisher("cmd_pwm", Float32, queue_size=10)
    rospy.init_node("Input")
    rate = rospy.Rate(10)

    scale_value = max_value - min_value

    fcn = {"trg": triangle, "crd": cuadrada}

    while not rospy.is_shutdown():
        #value = (((triangle()-min_value)/scale_value)*2)-1
        pub_1.publish((fcn[f_s])())
        rate.sleep()
```

Resultado

Accediendo al siguiente enlace se podrá observar un pequeño video explicativo de la solución obtenida.

https://drive.google.com/file/d/14MYm7d6iKe6ZErhfzjO7jjBtD8hM3MFl/view?usp=share_link

Conclusión

Además de aprender acerca de las señales PWM para control de motores, aprendimos que ROS es una herramienta bastante completa pues nos permite interconectar diferentes dispositivos para que interactúen juntos dentro del mismo sistema de ficheros que maneja ROS, lo cual no dio a entender que se puede aplicar a múltiples dispositivos a la vez, por lo que el potencial que tiene es muy elevado y por ese motivo nos motiva a seguir aprendiendo y practicando.