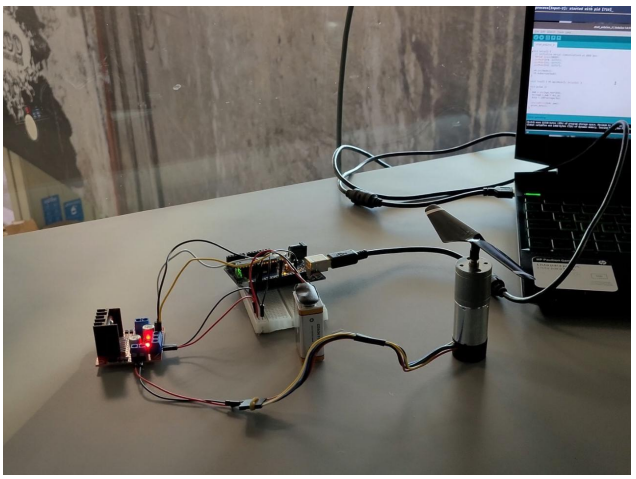
Instituto Tecnológico y de Estudios Superiores de Monterrey

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 ***sentido de giro y velocidad*** de un motor con ***encoder*** de cuadratura, al poder integrar todo esto en un mismo sistema interconectado por ***ROS y arduino***.

Introducción

Para poder controlar el ***sentido de giro y la velocidad*** del motor es necesario conocer que su funcionamiento se basa en dos conceptos muy utilizados cuando se habla de electromagnetismo ***(la ley de Lorenz y el efecto de atracción y repulsión entre los polos de un imán)***, con estos principios se establece que el movimiento de un material conductor que se encuentra envuelto en un campo magnético puede generar una corriente eléctrica.,

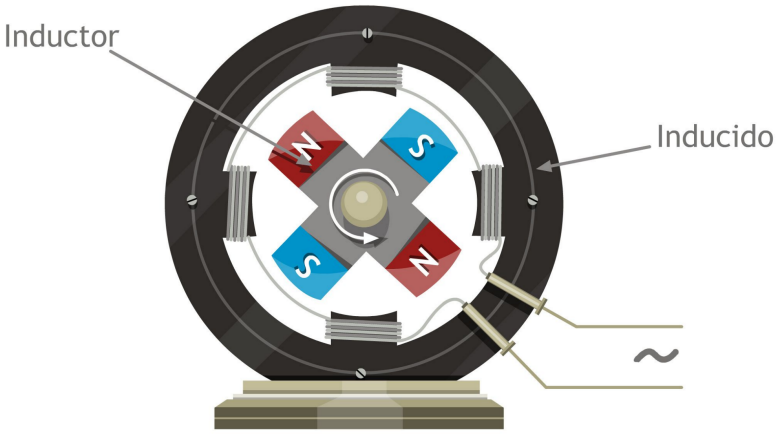


Figura 01. Motor eléctrico.

Air-Farez. (2022, septiembre). Solicitud de instalación de energía [Entrada de blog]. Recuperado de https://air-farez.blogspot.com/2022/09/solicitud-de-instalacion-de-energia.html

Con ayuda de la *figura 01*, se puede entender que los imanes del centro serán los encargados de dictaminar la dirección de giro del motor, mientras que la velocidad va a depender de la intensidad del campo magnético que a su vez depende del voltaje suministrado.
De esta manera si lo que se busca es configurar el motor en un solo estado, lo que se hace es fijar los valores de voltaje y sentido de la corriente para que este se mueva de manera constante, sin embargo muchas veces los problemas de hoy en día requieren de sistemas dinámicos por lo que para poder ir variando las configuraciones en tiempo real es necesario hacer uso de dos dispositivos ***(un puente H, y un encoder)***, además de una señal ***PWM***.

En este caso el ***puente H***, se puede definir como un circuito integrado que sirve para para variar la intensidad de la corriente y el sentido de la misma, mientras que el ***encoder*** es el circuito encargado de transformar esa corriente en una señal digital, para que así la computadora sea capaz de conocer el sentido de giro, la velocidad, cantidad de vueltas, etc, y utilizar esta información para poder reajustar el comportamiento del motor.

Respecto a la velocidad de motor, como ya se mencionó antes, esta depende del voltaje suministrado, por lo que si se quiere la velocidad sea variable el voltaje también tiene que serlo y una manera de lograrlo es con las señales ***PWM (Pulse Modular Width)*** pues esta al ser una señal digital que únicamente tiene dos estados High y Low que se envían periódicamente a cierta frecuencia, el promedio de voltaje generado al final de un ciclo va a ir variando dependiendo de la cantidad y duración de los pulsos High y los Low. (ver *figura 02)*, de esta manera se puede ir variando la frecuencia de los pulsos para obtener el voltaje que se desee para cada configuración.

Por lo que en esta ocasión para nosotros poder lograr diseñar este sistema primero utilizamos una laptop la cual es la encargada de dictaminar la velocidad y dirección final del motor, posteriormente esta información es recibida por el arduino a través del nodo ***cmd_pwm*** en forma de un mensaje customizado para ser procesado aquí mismo, para posteriormente mandar a través de los puertos designados para pwm *(2, 3)* la señal que va a recibir el puente h y el encoder *(ver figura 03)*.

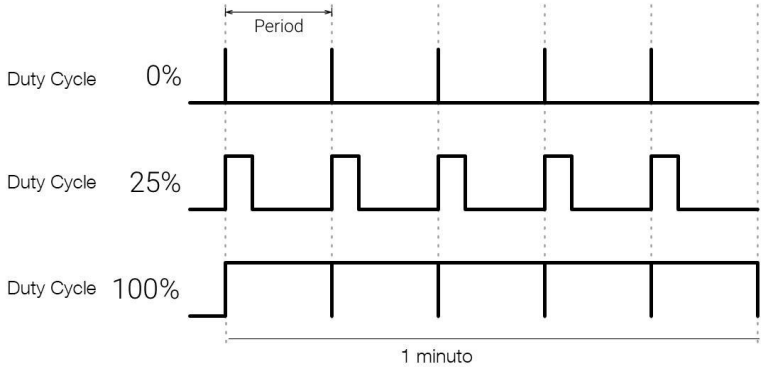


Figura 02. Visualización de señales

PWM Adafruit. (s.f.). Fading a RGB LED on BeagleBone Black using PWM [Tutorial]. Recuperado de https://learn.adafruit.com/fading-a-rgb-led-on-beaglebone-black/pwm

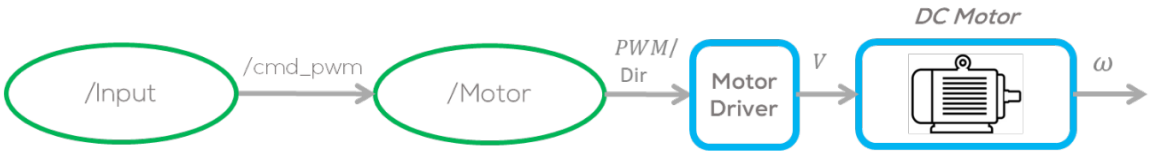


Figura 03. Sistema final usando ROS

Solución

Antes de explicar la parte del código, es importante armar la parte física del sistema (ver figura 04).

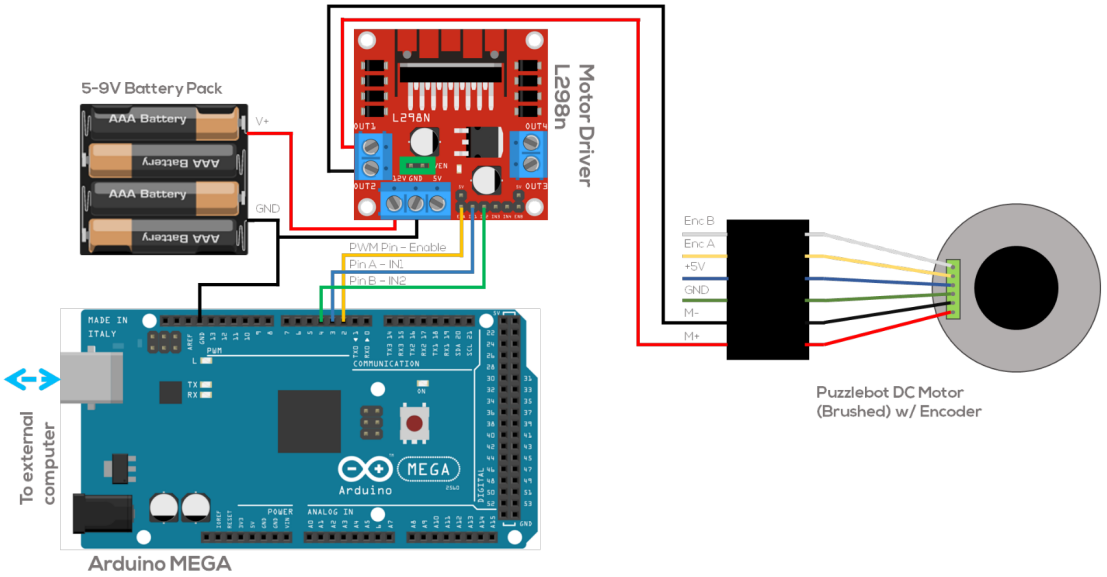


Figura 04. Diagrama de conexiones físicas

Una vez armado el sistema se pasa al código de arduino, por lo que es necesario incluir las librerías ***ros.h*** *y* ***std_msgs,*** esto con la finalidad de indicar 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.

Código Arduino (Transforma el mensaje customizado en una señal PWM)

```
#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" );
}
```

Posteriormente dentro de nuestro script de python realmente no tenemos que aplicar cambio alguno, en cuestión de terminologías y aplicaciones de un nodo, por lo que el código queda de la siguiente manera:

Código Python (Establece la dirección y el tipo de señal)

```
#!/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()
```

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>
```

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.