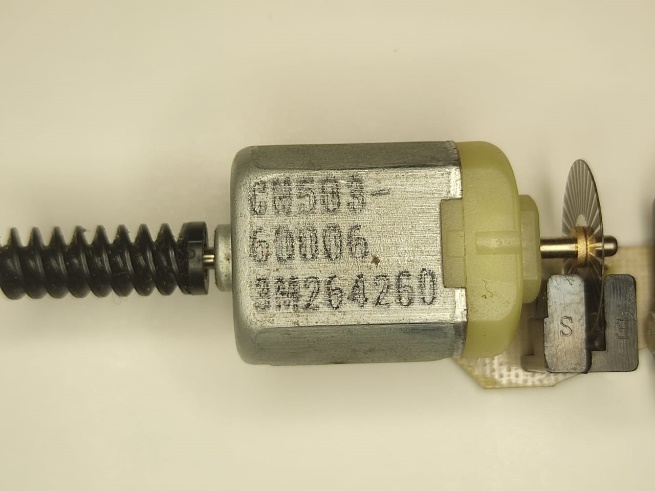
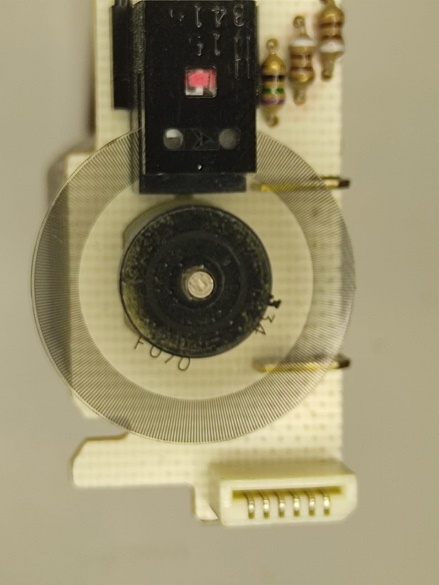
Medir Velocidad del Encoder del Motor

En esta sección tenemos como objetivo hacer uso del encoder que trae integrado el motor CN503. Existen básicamente dos tipos de encoder según su diseño básico y funcionalidad: encoder Incremental y encoder Absoluto. Adicionalmente existen otros tipos de encoders como por ejemplo el encoder óptico, lineal y el encoder de cuadratura. El encoder óptico es el tipo de encoder más comúnmente usado y consta básicamente de tres partes: una fuente emisora de luz, un disco giratorio y un detector de luz conocido como “foto detector”.

El disco esta montado sobre un eje giratorio y cuenta con secciones opacas y transparentes sobre la cara del disco. La luz que emite la fuente es recibida por el foto-detector o interrumpida por el patrón de secciones opacas produciendo como resultado señales de pulso.

El Código que se produce con dichas señales de pulso es entonces leída por un dispositivo controlador el cual incluyen un micro-procesador para determinar el ángulo exacto del eje.

**Materiales necesarios.**

1. Raspberry Pi 4

2. Conector tipo T

3. Flex para conector tipo T

4. Protoboard.

5. Modulo puente H, L298N

6. Motor de corriente directa de 12 v con encoder,

7. Adaptador fuente de 12 v.

8. Fuente de voltaje para Raspberry pi

Diagrama del Circuito Encoder-Raspberry

Circuito propuesto para la conexión del encoder del motor con el adaptador tipo T de la raspberry

Diagrama

Descripción generada automáticamente

Obviamente se necesita que el motor también se conecte al mismo tiempo, por lo que el diagrama total sería:

Diagrama

Descripción generada automáticamente

Código en Python

Antes de poner el código es necesario hacer unas aclaraciones,

1.- Se necesita el archivo read\_RPM.py (<https://github.com/AlexAlaffita/Simulacion-y-control-de-un-motor-cc/Python>) en el mismo directorio que en donde esté el presente script

2.- En la raspberry se tiene que activar el Daemon pigpiod, en una terminal de las raspberry ponemos: >>sudo pigpiod

import RPi.GPIO as GPIO

from time import sleep

import pigpio

from read\_RPM import reader

# Pines de conexión con el shield L298N

in1 = 24

in2 = 23

en = 25

GPIO.setmode(GPIO.BCM) # Para fijar el modo para los pines

GPIO.setup(in1,GPIO.OUT) # Definimos a in1 como de salida en raspberry

GPIO.setup(in2,GPIO.OUT) # Definimos a in2 como de salida en raspberry

GPIO.setup(en,GPIO.OUT) # Definimos a en como de salida en raspberry

GPIO.output(in1,GPIO.LOW)# Para estar seguros que el motor se detiene antes de comenzar

GPIO.output(in2,GPIO.LOW)# Para estar seguros que el motor se detiene antes de comenzar

p=GPIO.PWM(en,1000) # La frecuencia para el pulso PWM en el pin en, de la variable p

p.start(25) # Para comenzar con el 25% con el movimiento del motor

print("\n")

print("Recuerda que debes de suministrar un porcentaje del voltaje (entre 0 y 100)")

print("Esperando...")

print("\n")

#Iniciamos el motor de dc con el 0% del voltaje total (12 v)

GPIO.output(in1,GPIO.HIGH)

GPIO.output(in2,GPIO.LOW)

p.ChangeDutyCycle(0)

# Set up RPM reader

RPM\_GPIO = 6 # Pin del que vamos a leer

SAMPLE\_TIME = 0.01 # Tiempo de muestreo

pi = pigpio.pi() # Creación de nuestra variable para las RPM

pi.set\_servo\_pulsewidth(6, 2000) # Maximum throttle.

sleep(2)

pi.set\_servo\_pulsewidth(6, 1000) # Minimum throttle.

sleep(2)

tach = reader(pi, RPM\_GPIO,334) # Creación de la clase reader, el 334 son los pulsos por ciclo

while True:

x=input("% del voltaje\n") # Pedimos un porcentaje o v para la velocidad o -1 salir

# z=int(x) # Convertimos el valor de str de x a int (entero)

print(x)

if x == '-1': # Para salirnos del ciclo

p.ChangeDutyCycle(0)

break

elif x == 'v': # Medir una vez la velocidad

rpm = tach.RPM() # Aqui es donde obtenemos la interrupcion para medir la velocidad

print(rpm)

sleep(SAMPLE\_TIME)

break

p.ChangeDutyCycle(int(x)) # Cambiamos el porcentaje de voltaje (0-100)

print("Fin del programa, ¡¡¡Saludos!!!")

#Para asegurarnos de apagar el motor

GPIO.output(in1,GPIO.LOW)

GPIO.output(in2,GPIO.LOW)

GPIO.cleanup() # Limpiamos todos los pines

pi.stop() # Disconnect pigpio.