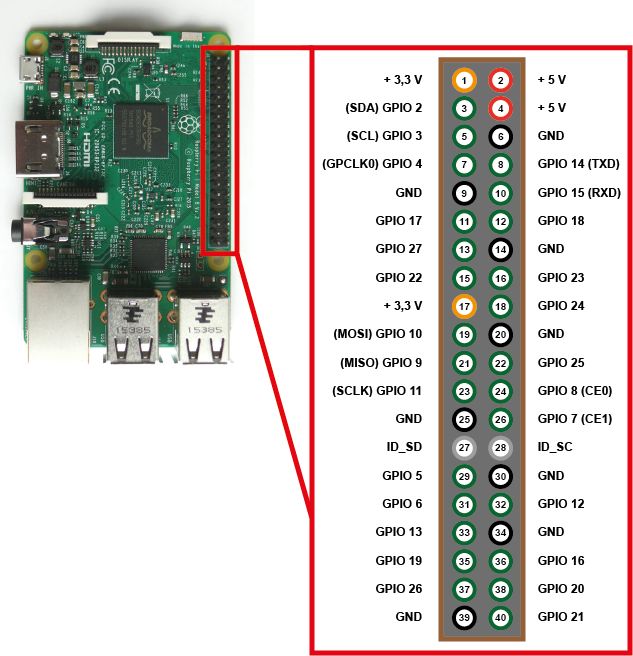
**Laboratorio 5 – PWM, Motores e Instalación de QtCreator**

Prerequisitos de Instalación

* Instalar librería wiringpi y serialControl
  + sudo pip install wiringpi2
* Si gusta puede desarrollar en el raspberry pi, sin embargo es mejor instalar qtcreator en la PC de <http://www.qt.io>
* Instalación de pyqt y qtcreator en raspberry pi
  + sudo apt-get install python3-pyqt5
  + (Opcional) sudo apt-get install qt5-default qtcreator pyqt5-dev pyqt5-dev-tools



**Modulación de ancho de pulso**

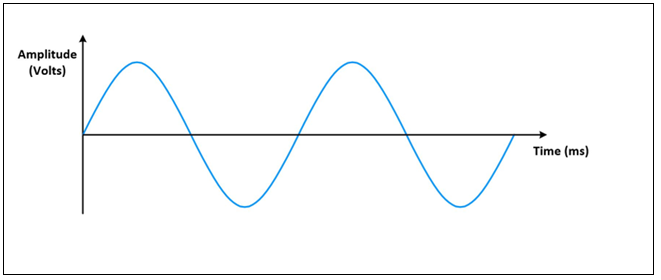
Antes de seguir adelante para conectar nuestro laboratorio y realizar la programación, entenderemos un concepto importante de PWM. En nuestro proyecto, PWM se utiliza para controlar la velocidad del motor de CC.

Entonces, ¿qué es PWM? PWM es una técnica para generar una señal de salida desde una fuente digital que se comporta como una señal analógica. Esto significa que podemos controlar circuitos analógicos o dispositivos analógicos usando una fuente de entrada digital. Esta técnica se utiliza en una amplia variedad de aplicaciones como mediciones, comunicación y control de potencia.

Primero entendamos la diferencia entre señales analógicas y digitales.

*Señal analoga*

Una señal analógica tiene un valor continuo y variable en cualquier intervalo de tiempo, con resolución infinita en tiempo y magnitud. Una señal analógica simple es una onda sinusoidal que se describe usando la amplitud y el tiempo como se muestra en la Figura inferior:



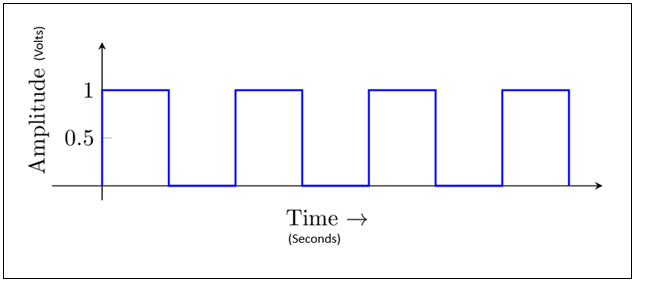
Las señales analógicas se utilizan para controlar las operaciones de los dispositivos analógicos, como el control de la velocidad de un motor de CC (un motor de CC es un dispositivo analógico). Para cambiar la velocidad del motor, simplemente modificamos la resistencia en su trayectoria actual debido a la corriente disponible en el motor y, por lo tanto, a la velocidad.

Consideremos que usamos una batería como fuente de señal analógica para controlar la velocidad del motor. Durante la operación, el voltaje de salida y la corriente de la batería caen durante un período de tiempo, debido a que el cambio en la resistencia no cambiará la velocidad del motor en la misma proporción en que lo estaba haciendo anteriormente. Debido a la naturaleza variable de una señal analógica, nunca podremos lograr la precisión y la consistencia en el control de velocidad. Una desventaja más del uso de señalización analógica en nuestro caso es que es propenso al ruido, lo que distorsionará la señal y reducirá la calidad de la transmisión.

Habiendo discutido los problemas con la señalización analógica, aún es posible controlar la operación, pero requiere circuitos analógicos de precisión que sean voluminosos, costosos y consuman mucha energía, lo cual no es ideal para muchos casos de uso.

Señal digital

Una señal digital tiene un valor no continuo y discreto, con resolución finita en tiempo y magnitud. La señal digital siempre tiene dos estados: completamente activado (alto) o completamente apagado (bajo) y la magnitud de la amplitud siempre se mantendrá estable. Una simple señal digital se representa como una onda cuadrada como se muestra en el siguiente diagrama:

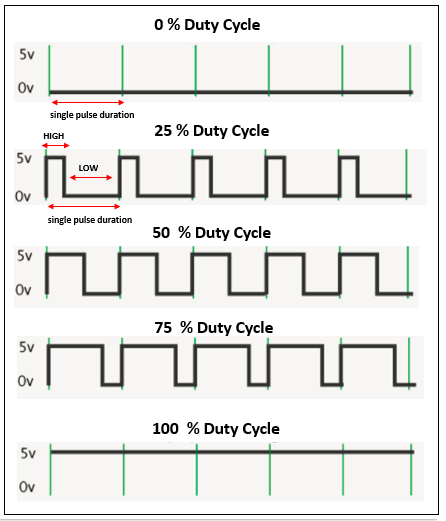


Debido a las deficiencias de la señal analógica en la aplicación de control, usamos la señal digital para controlar los mismos circuitos analógicos debido a la naturaleza estable y discreta de la señal digital. Y aquí PWM entra en escena.

PWM es una técnica de generación de señales analógicas utilizando una fuente digital. Las señales PWM consisten principalmente en dos componentes que definen su comportamiento, uno es el ciclo de trabajo (DC) y otro es la frecuencia.

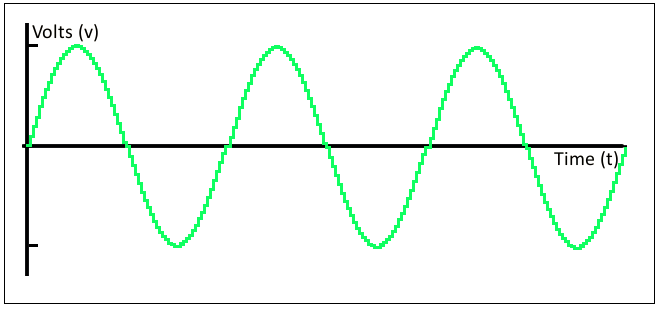
El ciclo de trabajo representa la cantidad de tiempo que la señal permanece en estado alto (encendido) como un porcentaje del tiempo total que toma completar un ciclo. La frecuencia define la rapidez con la que una señal cambia entre el estado alto (encendido) y bajo (apagado). Por ejemplo, si la frecuencia de una señal digital es de 100 Hz, la señal cambiará su estado entre alta y baja 100 veces.

El siguiente diagrama muestra cómo se comporta la señal digital cuando se varía el ciclo de trabajo:



Por lo tanto, al aplicar el ciclo de trabajo y la frecuencia apropiados, podemos crear una señal digital que se comportará como una señal analógica de voltaje constante y se puede usar para operar nuestro motor de CC.

La principal diferencia entre una señal analógica pura y una señal analógica creada digitalmente es que la curva de la señal analógica pura es suave (como la figura de señal analógica), mientras que la curva de una señal analógica creada digitalmente es escalonada, cuadrada y de naturaleza discreta (consulte la figura inferior):



**PROGRAMA DE PWM DE LED**

import time

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)

GPIO.setup(26, GPIO.OUT)

p = GPIO.PWM(26, 50) # channel=26 frequency=50Hz

p.start(0)

try:

while 1:

for dc in range(0, 101, 5):

p.ChangeDutyCycle(dc)

time.sleep(0.1)

for dc in range(100, -1, -5):

p.ChangeDutyCycle(dc)

time.sleep(0.1)

except KeyboardInterrupt:

pass

p.stop()

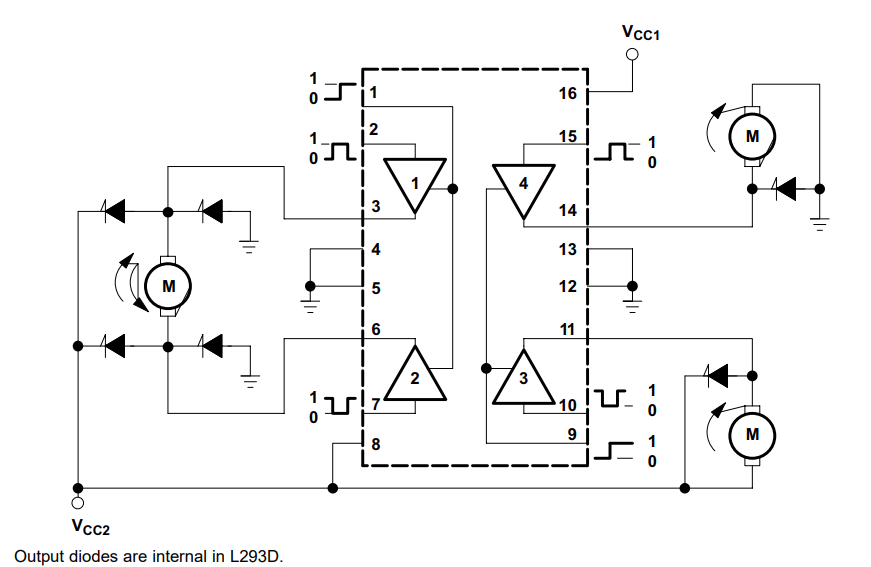
GPIO.cleanup()

**L293D**

El L293D es un chip de circuito integrado que se usa más comúnmente para impulsar el motor de CC / motor paso a paso / motor de engranajes. Este IC consta de dos puentes en H que pueden impulsar dos motores de CC simultáneamente y en sentido horario y antihorario. Actúa como una interfaz entre la Raspberry Pi (o cualquier otro microprocesador) y el motor.

El motor de CC necesita una cantidad significativa de corriente para arrancar, que la Raspberry Pi no puede proporcionar como salida a través de GPIO. Por ejemplo, un motor de 5V DC necesita alrededor de 300 a 400 mA de corriente, que es mucho más allá de lo que está disponible en el GPIO de la Raspberry Pi, y si conectamos directamente el motor al GPIO, puede dañar el Pi. Por lo tanto, se recomienda evitar la conexión directa del motor a Pi.

Veamos el diagrama de pines de IC L293D para comprender su funcionamiento:



El L293D tiene una configuración de 16 pines. Hay dos patillas de activación en el chip, la patilla 1 y la patilla 9; este pasador de habilitación se utiliza para accionar el motor. Si la activación es alta, entonces el motor funcionará y si está bajo, el motor se detendrá. Los pines 4, 5, 12 y 13 están molidos (GND).

Hay cuatro pines como entrada, que deciden la dirección de rotación del motor; los pines 2, 7, 10 y 14 están marcados como entrada. Estas cuatro clavijas de entrada se utilizan para impulsar dos motores en una dirección particular: dos clavijas para cada motor.

Si la entrada 1 (pin 2) es alta y la entrada 2 (pin 7) es baja, el motor girará en sentido horario; Si la entrada 1 (pin 2) está baja y la entrada 2 (pin 7) está alta, entonces el motor girará en sentido antihorario. De manera similar, para el otro conjunto de dos pines, si la entrada 3 (pin 10) es alta y la entrada 4 (pin 15) está baja, el motor girará en el sentido de las agujas del reloj; Si la entrada 3 (pin 10) está baja y la entrada 4 (pin 15) está alta, el motor girará en sentido contrario a las agujas del reloj. La dirección de rotación del motor también depende de la conexión entre el terminal del motor y los pines de salida de L293D.

Cuatro pines de salida están ahí para proporcionar salida a dos motores. Los pines 3 y 6 están conectados a los dos terminales de un motor, y los pines 11 y 14 están conectados a los terminales de otro motor. Tenga en cuenta que la dirección de rotación del motor también depende de la conexión entre las clavijas de salida y los terminales del motor.

Hay dos pines Vcc: uno es el pin 8 (Vcc2), que está conectado a un 9V / 5V (el voltaje de la batería depende de la tensión nominal del motor utilizado) terminal positivo de la batería de CC. Este Vcc2 se utiliza para encender el motor. El otro es el pin 16 (Vcc1), que se utiliza para encender el IC L293D.

ACLARACIONES:

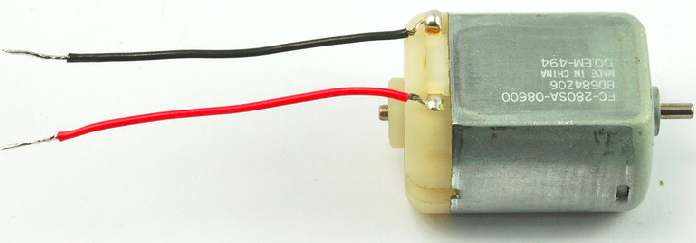
* *No energizar el integrado con el voltaje de la fuente de Raspberry Pi*
  + *Usar una fuente externa (bateria)*
  + *Conectar GND de Raspberry a GND (Vss) del integrado*
* *Energizar EN1 para activar los buffers del L293D 1 y 2*

**Motor DC**

Un motor de corriente continua es un motor eléctrico que convierte la energía eléctrica en energía mecánica. Un motor de CC funciona con una corriente continua y se basa en el principio de que, cuando un conductor portador de corriente se coloca en un campo magnético, experimenta un par y tiene la tendencia a girar. La dirección de la corriente que fluye en el conductor decide la dirección de rotación del conductor.

No vamos a entrar en mucho detalle sobre el funcionamiento interno de un motor de CC, ya que esto está fuera del alcance de este capítulo; Le sugiero que lea más sobre esto más tarde.

Para este laboratorio, usaremos un motor de 9V o simular de CC, como se muestra en el siguiente diagrama:



**CÓDIGO DE PRUEBA DE MOTOR CONTROL RASPBERRY PI**

#!/usr/bin/env python3

#rover\_drivefwd.py

#HARDWARE SETUP

# GPIO

# 2[==X====LR====]26[=======]40

# 1[=============]25[=======]39

import time

import wiringpi

ON=1;OFF=0

IN=0;OUT=1

STEP=0.5

PINS=[16,18] # PINS=[L-motor,R-motor]

FWD=[ON,ON]

RIGHT=[ON,OFF]

LEFT=[OFF,ON]

DEBUG=True

class motor:

# Constructor

def \_\_init\_\_(self,pins=PINS,steptime=STEP):

self.pins = pins

self.steptime=steptime

self.GPIOsetup()

def GPIOsetup(self):

wiringpi.wiringPiSetupPhys()

for gpio in self.pins:

wiringpi.pinMode(gpio,OUT)

def off(self):

for gpio in self.pins:

wiringpi.digitalWrite(gpio,OFF)

def drive(self,drive,step=STEP):

for idx,gpio in enumerate(self.pins):

wiringpi.digitalWrite(gpio,drive[idx])

if(DEBUG):print("%s:%s"%(gpio,drive[idx]))

time.sleep(step)

self.off()

def cmd(self,char,step=STEP):

if char == 'f':

self.drive(FWD,step)

elif char == 'r':

self.drive(RIGHT,step)

elif char == 'l':

self.drive(LEFT,step)

elif char == '#':

time.sleep(step)

def main():

import os

if "CMD" in os.environ:

CMD=os.environ["CMD"]

INPUT=False

print("CMD="+CMD)

else:

INPUT=True

roverPi=motor()

if INPUT:

print("Enter CMDs [f,r,l,#]:")

CMD=input()

for idx,char in enumerate(CMD.lower()):

if(DEBUG):print("Step %s of %s: %s"%(idx+1,len(CMD),char))

roverPi.cmd(char)

if \_\_name\_\_=='\_\_main\_\_':

try:

main()

finally:

print ("Finish")

#End

Ejecute el programa

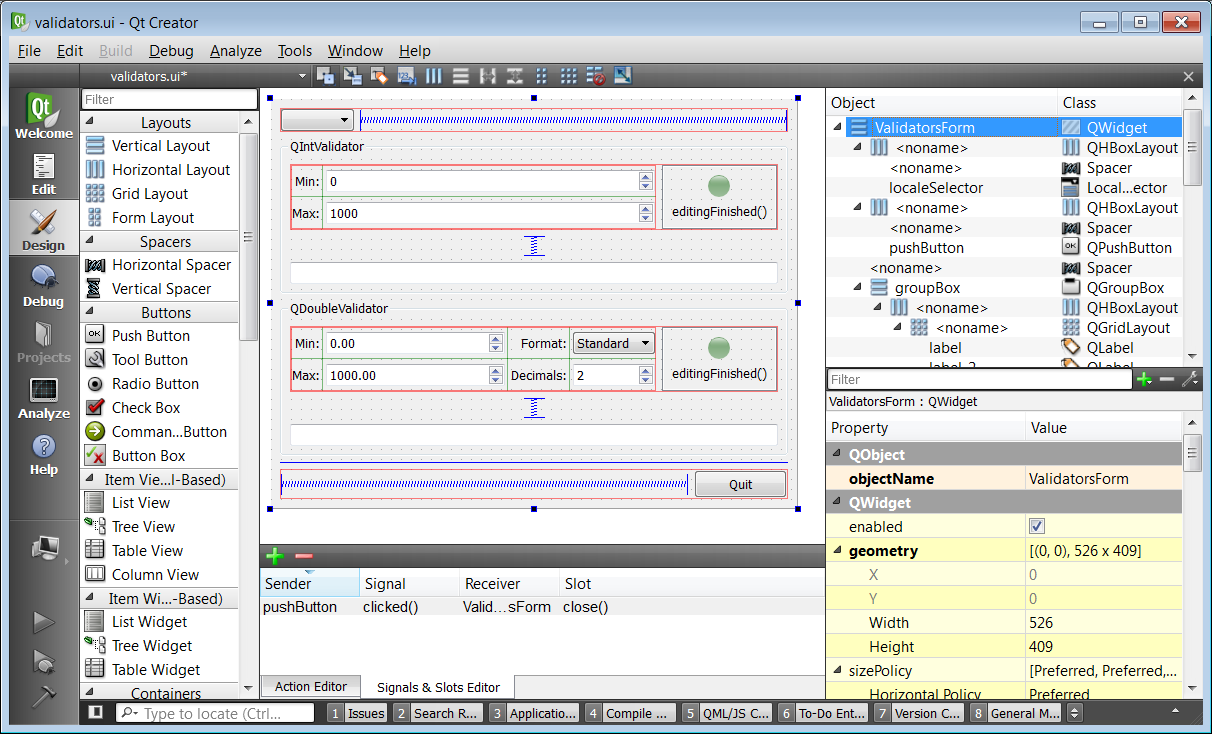
sudo python3 motor.py

E ingrese los commandos de la siguiente manera, por ejemplo: “ffrr#llfrll”

f simboliza adelante, r reversa, l izquierda, r derecha y # un espacio de espera de tiempo.

**Qt Creator, ¿Qué es?**

Qt Creator es un entorno de desarrollo integrado multiplataforma C ++, JavaScript y QML que forma parte del SDK para el marco de desarrollo de aplicaciones Qt GUI. [3] Incluye un depurador visual y un diseño de GUI integrado y diseñador de formularios. Las características del editor incluyen resaltado de sintaxis y autocompletado. Qt Creator usa el compilador C ++ de la colección de compiladores GNU en Linux y FreeBSD. En Windows, puede usar MinGW o MSVC con la instalación predeterminada y también puede usar el Microsoft Console Debugger cuando se compila desde el código fuente. Clang también es compatible.



Crear una interfaz sencilla para encender un LED

**CODIGO DE PRUEBA DE PYQT EN RASPBERRY PI**

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtWidgets import QMessageBox

def btn\_clicked():

print ("Button Pressed")

QMessageBox.information(MainWindow, 'Welcome',

'PyQt5 Running on Raspberry Pi')

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(284, 123)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.btn = QtWidgets.QPushButton(self.centralwidget)

self.btn.setGeometry(QtCore.QRect(60, 30, 151, 51))

self.btn.setObjectName("btn")

MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate(

"MainWindow", "Message Box Example"))

self.btn.setText(\_translate("MainWindow", "Click Me"))

'''User Code'''

self.btn.clicked.connect(btn\_clicked)

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

MainWindow = QtWidgets.QMainWindow()

ui = Ui\_MainWindow()

ui.setupUi(MainWindow)

MainWindow.show()

sys.exit(app.exec\_())

Ejecute la aplicación en el Raspberry Pi (Consejo, usar IDLE para ejecutar).

Asignación

* 40% Hace lo anterior
* 10% Instalar QtCreator en La PC
* 50% Realizar un programa en el Raspberry Pi que realice
  + Un PWM que ejecute tres ciclos de trabajo diferentes en tres segundos.
  + Cambiar la velocidad del motor de plena potencia 0, 25, 50, 75, 100 por entrada de usuario