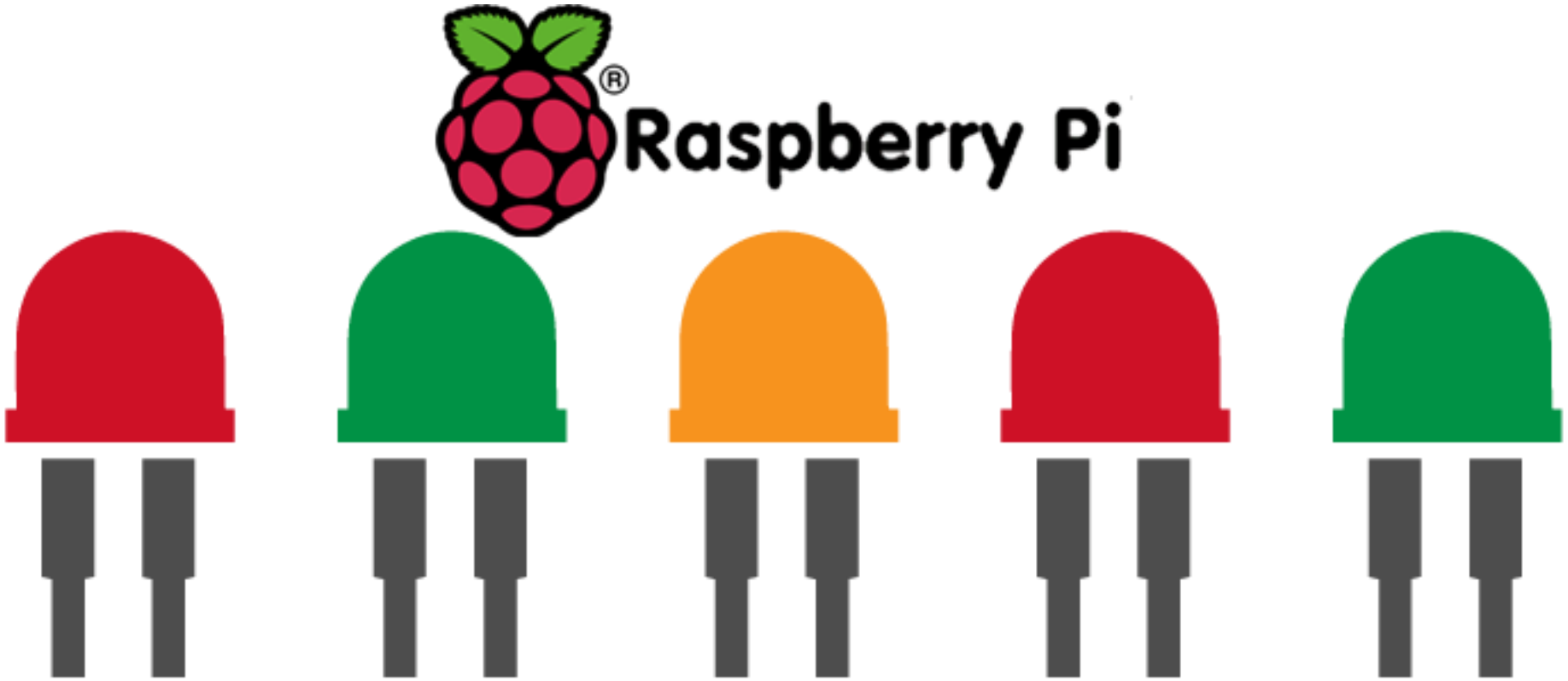
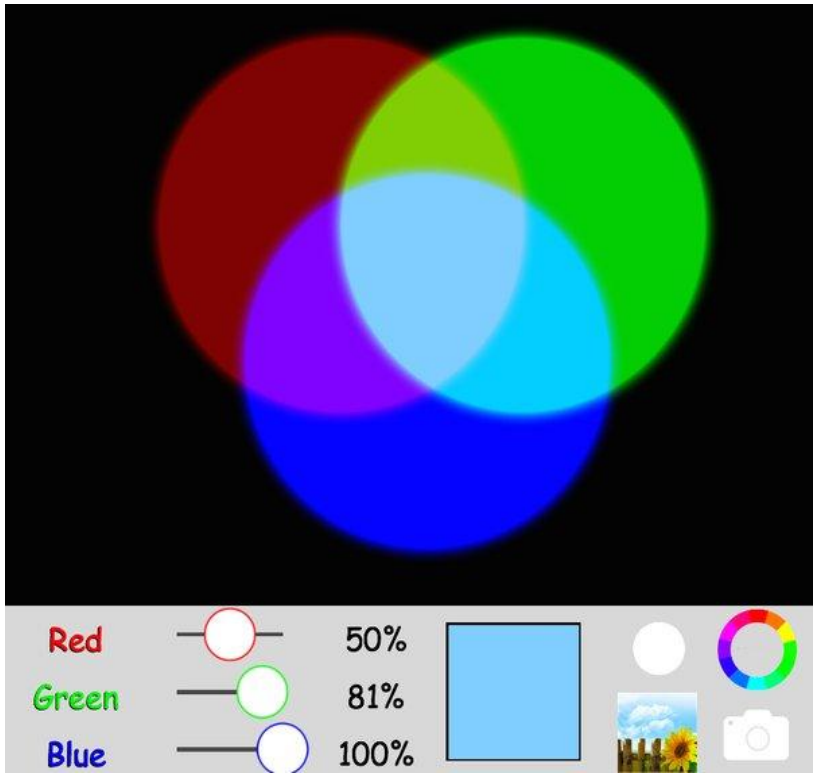
جامعة حلب
كلية الهندسة الإلكترونية والكهربائية

PROGRAMMING A RASPBERRY PI WITH PYTHON

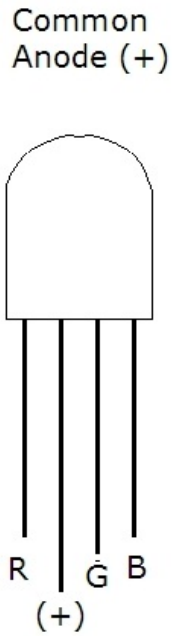


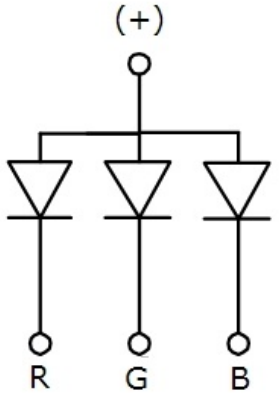
إعداد: م.علا جزماتي

تجربة قيادة الثنائي RGB LED باستخدام Raspberry pi



مبدأ العمل:





Red Green Blue	111	101	100	100	011	010	001	000
LED State	OFF	Blue	Green	Cyan	Red	Magenta	Yellow	White

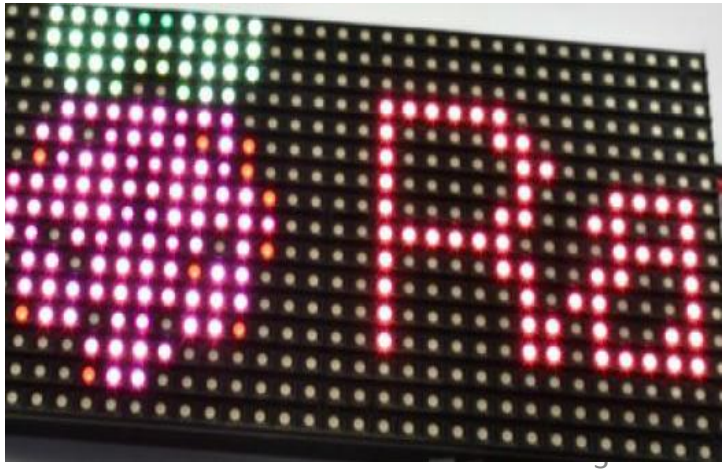
كما يمكن الحصول على تدرجات لونية أخرى حسب الجهود المطبقة على الأقطاب.

تجربة قيادة الثنائي RGB LED باستخدام Raspberry pi

أهمية التعرف على طريقة قيادة المتصلات الضوئية متعددة الألوان:

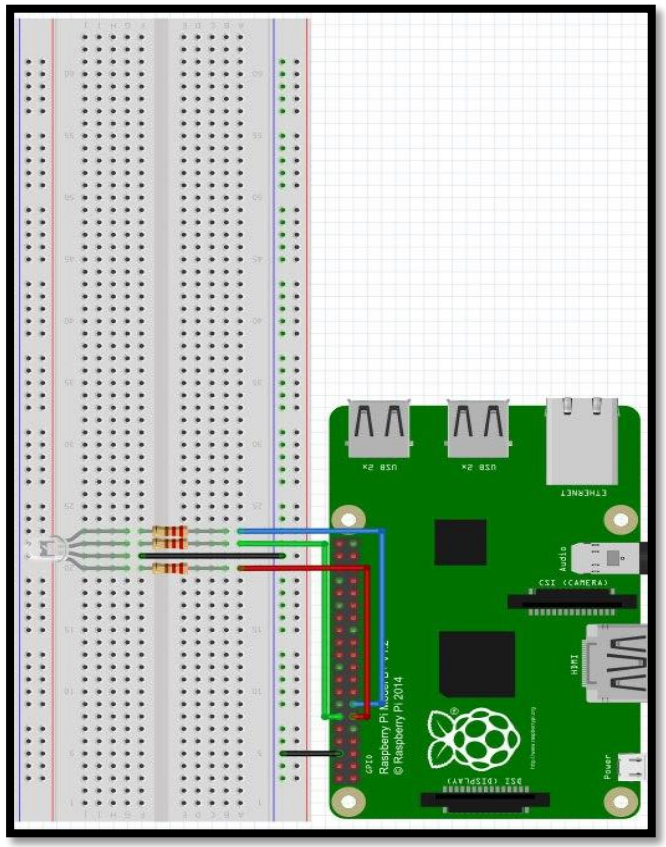
١- الحصول على أي تدرج لوني مطلوب من متصل وحيد.

٢- تصميم الجرائد الالكترونية الملونة.



تجربة قيادة الثنائي RGB LED باستخدام Raspberry pi

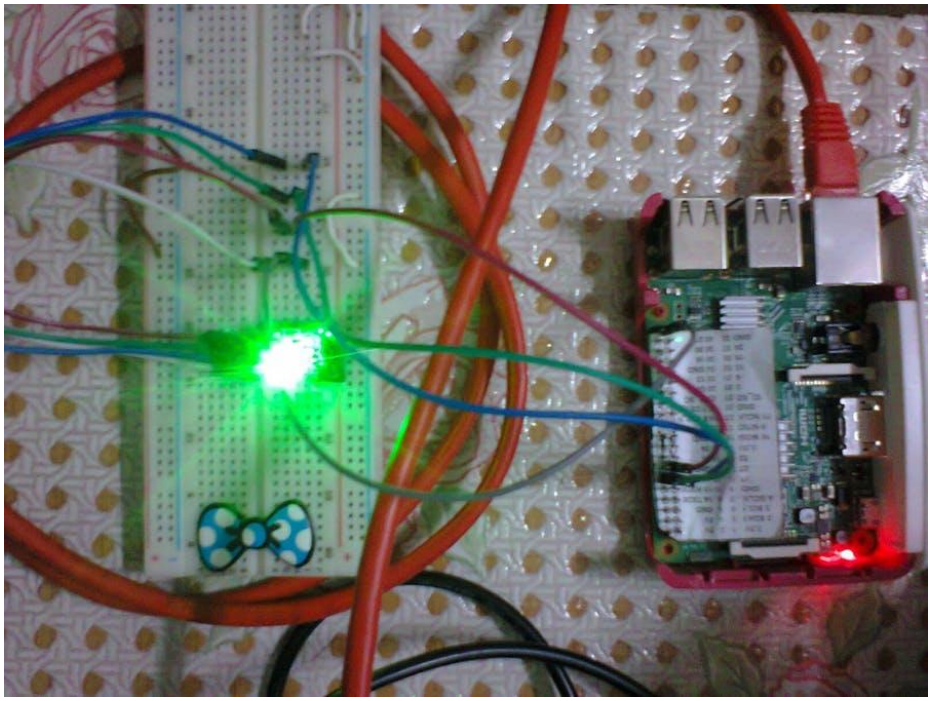
```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)

RED = 11
GREEN = 12
BLUE = 13

GPIO.setup(RED,GPIO.OUT)
GPIO.output(RED,0)
GPIO.setup(GREEN,GPIO.OUT)
GPIO.output(GREEN,0)
GPIO.setup(BLUE,GPIO.OUT)
GPIO.output(BLUE,0)

try:
   while (True):
      request = raw_input('RGB—>')
      if (len(request) == 3):
         GPIO.output(RED,int(request[0]))
         GPIO.output(GREEN,int(request[1]))
         GPIO.output(BLUE,int(request[2]))

except KeyboardInterrupt:
   GPIO.cleanup()
```

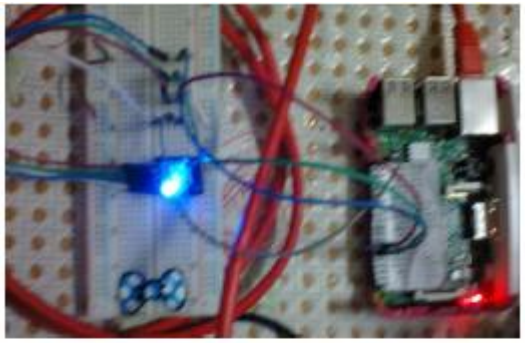


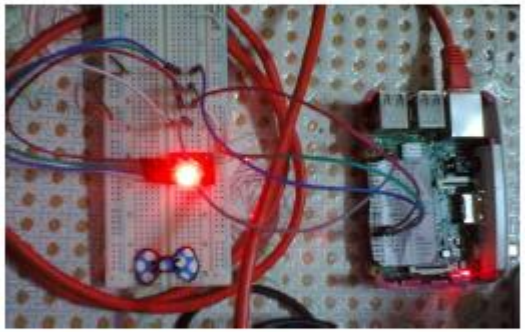
Note:
Python 3
Only input

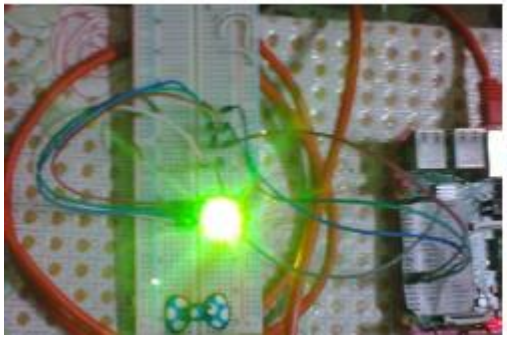
تجربة قيادة الثنائي RGB LED باستخدام Raspberry pi

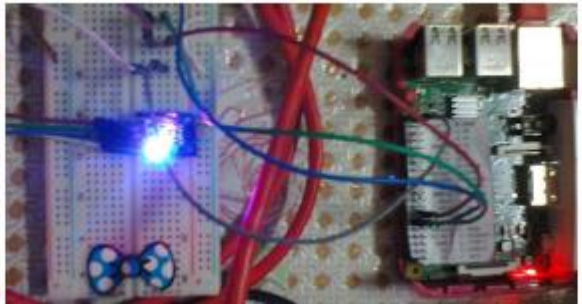
النتائج :

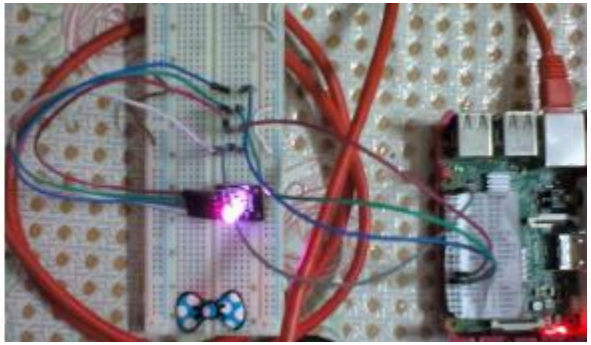












طريقة توليد نبضات PWM ضمن RPi.GPIO

To create a PWM instance:

p = GPIO.PWM(channel, frequency)

To start PWM:

p.start(dc) # where dc is the duty cycle (0.0 <= dc <= 100.0)

To change the frequency:

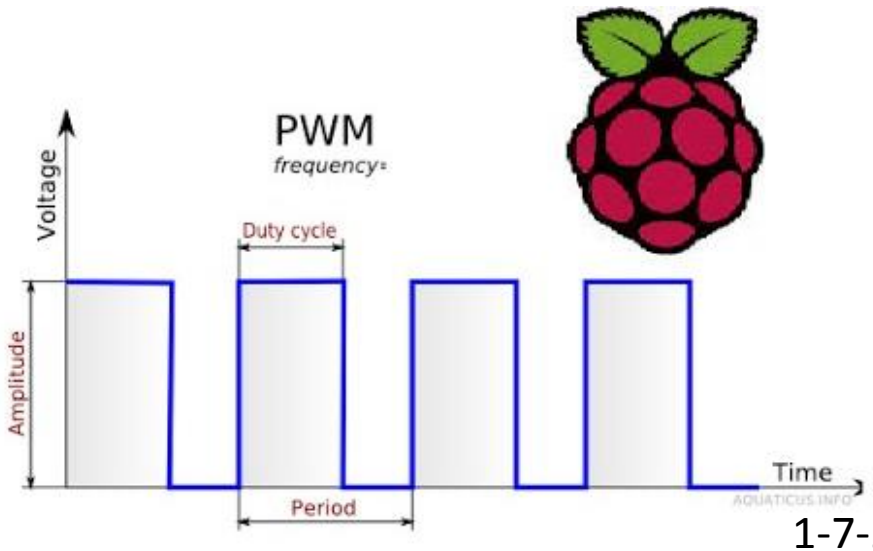
p.ChangeFrequency(freq) # where freq is the new frequency in Hz

To change the duty cycle:

p.ChangeDutyCycle(dc) # where 0.0 <= dc <= 100.0

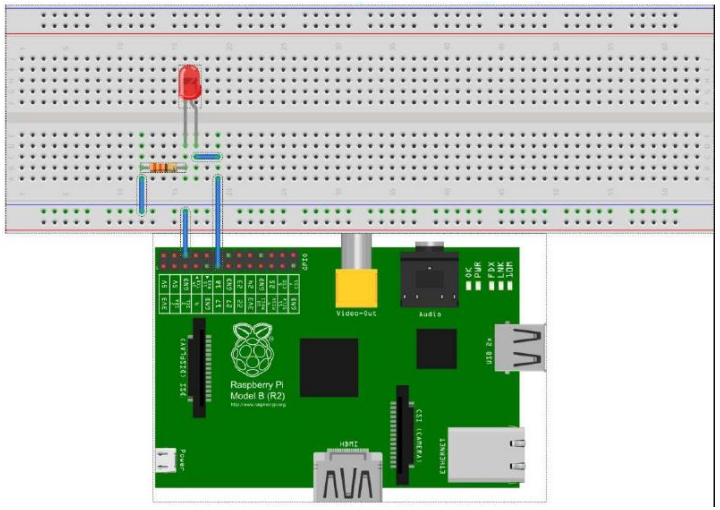
To stop PWM:

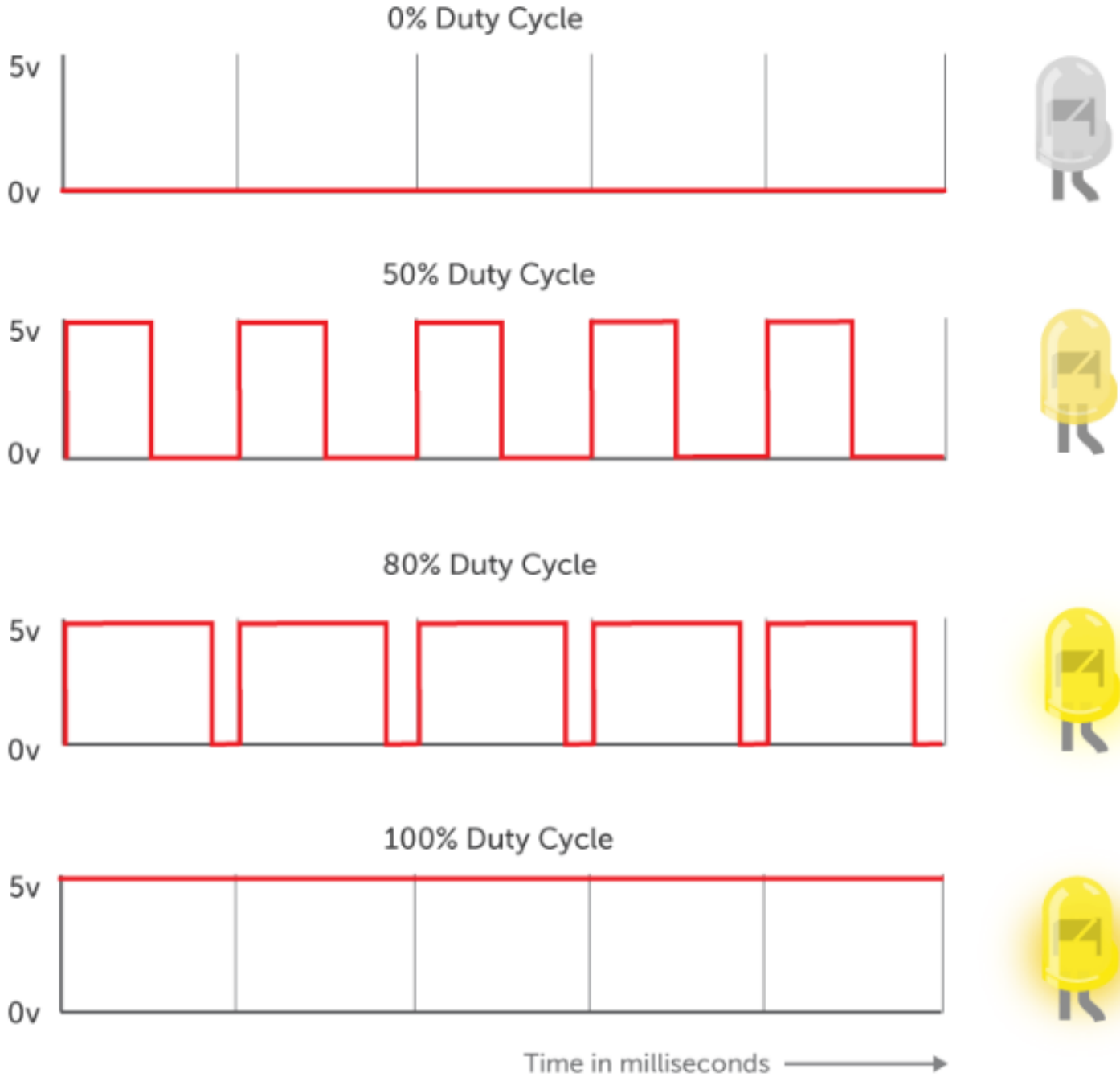
p.stop()



مثال ۱

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)

p = GPIO.PWM(11, 50)
p.start(5)

p.ChangeDutyCycle(10)
time.sleep(3)
p.ChangeDutyCycle(40)
time.sleep(3)
p.ChangeDutyCycle(80)
time.sleep(3)
p.ChangeDutyCycle(100)
time.sleep(3)

p.stop()
GPIO.cleanup()
```



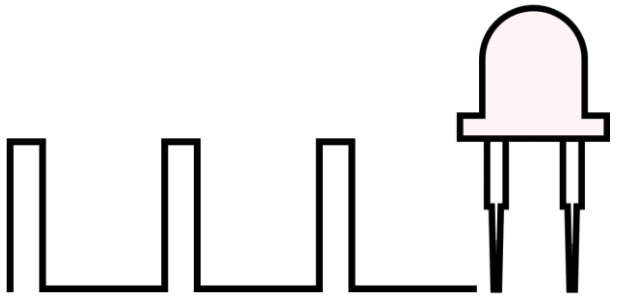

0% Duty Cycle
5v
0v
50% Duty Cycle
5v
0v
80% Duty Cycle
5v
0v
100% Duty Cycle
5v
0v
Time in milliseconds

Untitled

File Edit Format Run Options Windows Help

مثال ۲

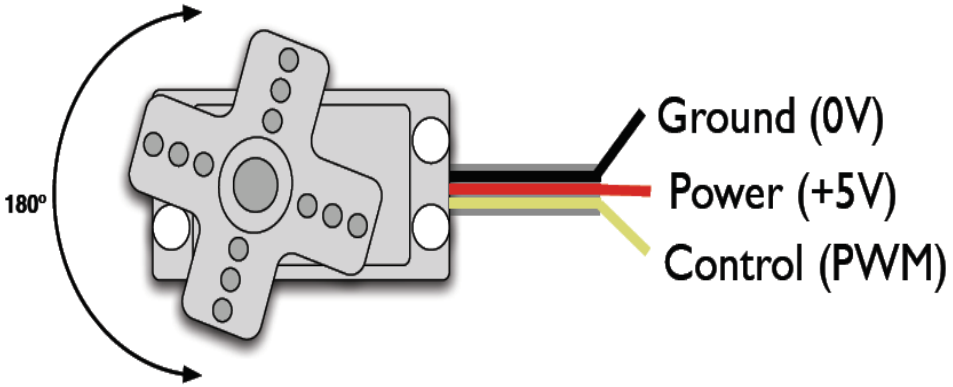
```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)

p = GPIO.PWM(12, 50)  # channel=12 frequency=50Hz
p.start(0)
try:
   while 1:
      for dc in range(0, 100, 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()
```



An example to brighten/dim an LED:

التحكم بمحرك servo

- محرك السيرفو يستخدم فى التحكم الموضعي أي أنك تستطيع أن تتحكم فى أي نقطة يجب أن يتوقف عندها المحرك وعدد درجات الدوران بسبب وجود علبة التروس بداخله التى تعطى دقة فى الحركة.
- حركة محرك السيرفو بطيئة جداً لذلك يتميز بالقدرة العالية على التحكم به وعزمه الكبير.
- يتحرك محرك السيرفو ١٨٠ درجة وتوجد أنواع منه تتحرك ٣٦٠ درجة .
- يتكون محرك السيرفو داخلياً من دارة تحكم "تكون في الغالب مايكروكنترولر "، و عندما نعطي المحرك نبضات Pulses بثابت زمني معين يدور المحرك للزاوية حسب هذا الثابت الزمني.



أنواع servo motor حسب نمط العمل:

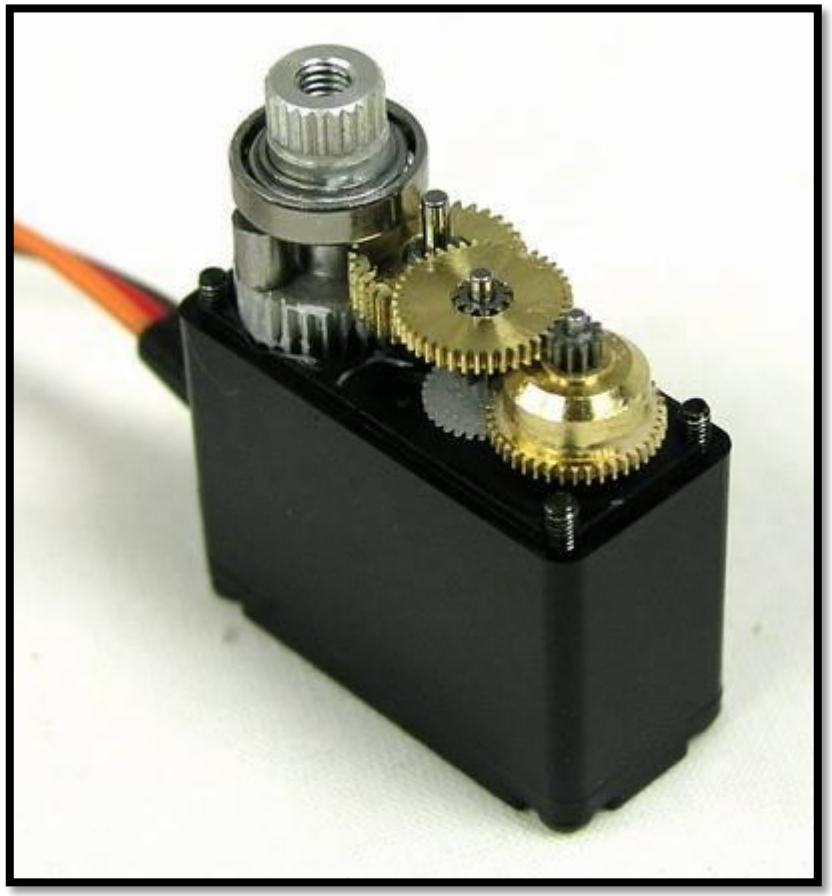
النوع القياسي :

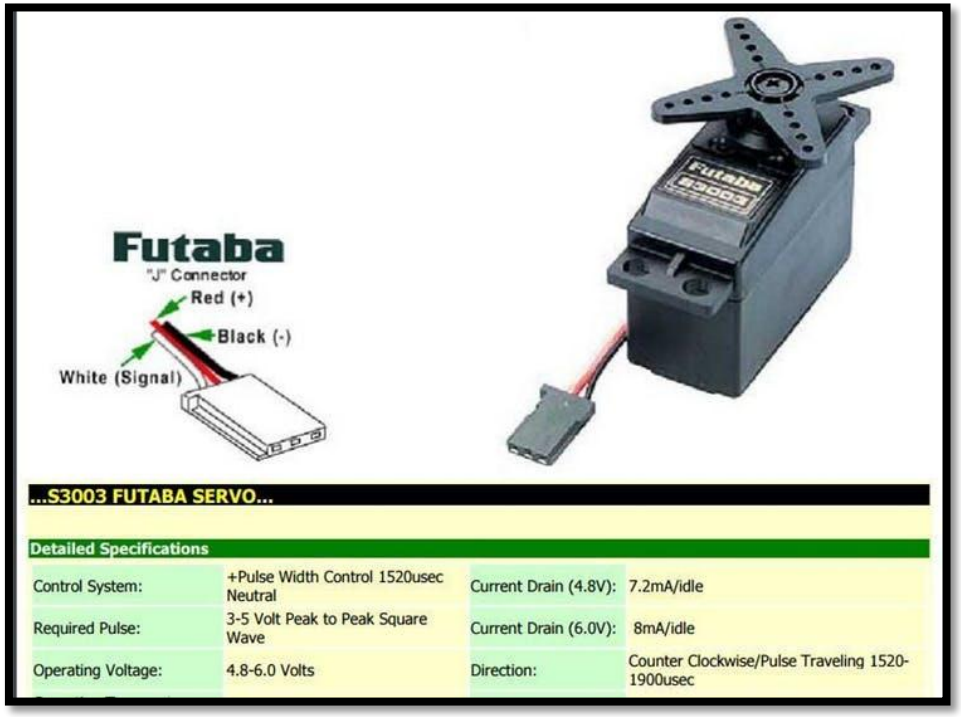
المحرك قادر على الدوران من 0 إلى 180 درجة في الاتجاهين مع وعكس عقارب الساعة.

معظم المحركات التي يطلق عليها اسم RC Servo كي تدور باتجاه معين نقدم لها مجموعة نبضات بتردد 50 HZ وبعرض نبضة 20 MS.

النوع المستمر:

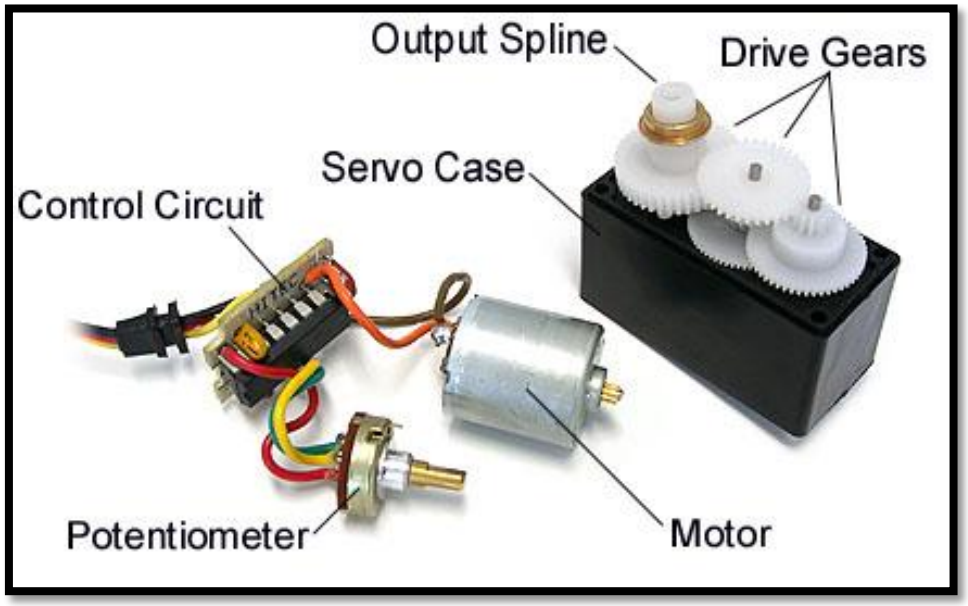
المحرك قادر على الدوران من 0 إلى 360 درجة في كلا الاتجاهين مع وعكس عقارب الساعة.

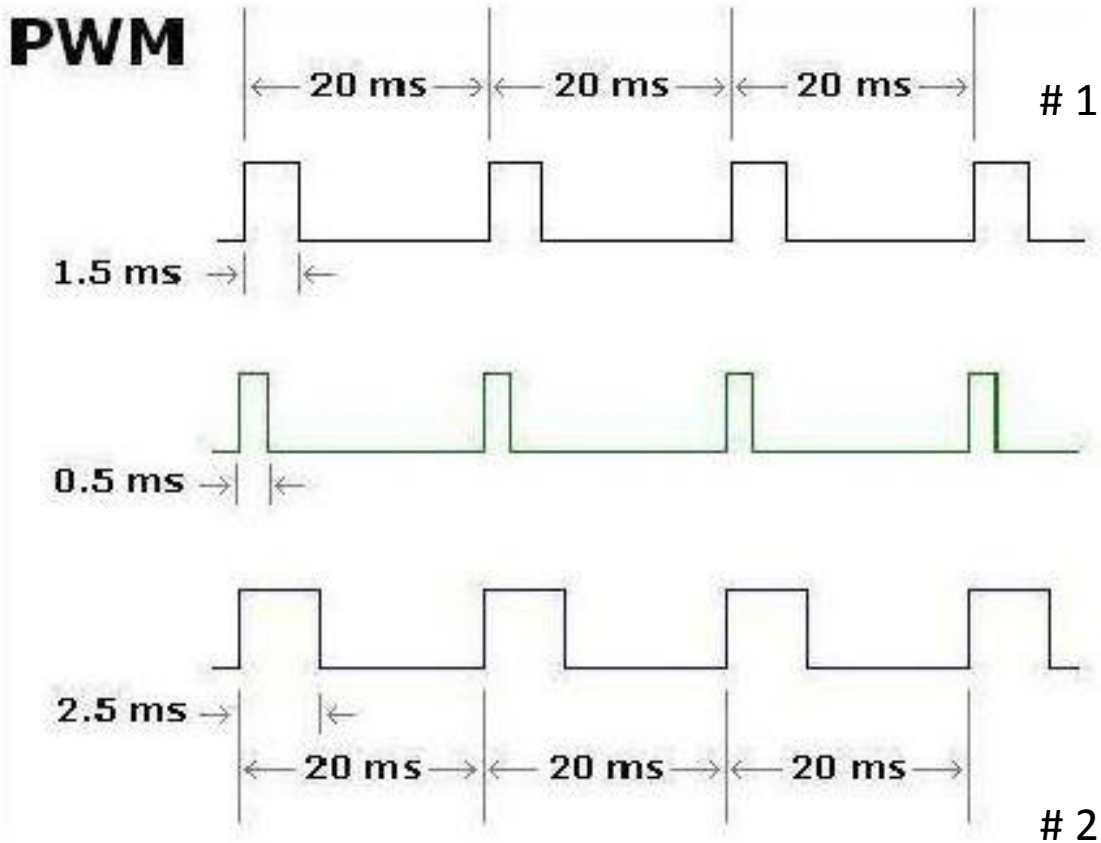




...S3003 FUTABA SERVO...

Detailed Specifications			
Control System:	+Pulse Width Control 1520usec Neutral	Current Drain (4.8V):	7.2mA/idle
Required Pulse:	3-5 Volt Peak to Peak Square Wave	Current Drain (6.0V):	8mA/idle
Operating Voltage:	4.8-6.0 Volts	Direction:	Counter Clockwise/Pulse Traveling 1520-1900usec





1.5ms pulse for 90 degree (MIDDLE)

0.5ms pulse for 0 degree (LEFT)

2.5ms pulse for 180 degree (RIGHT)

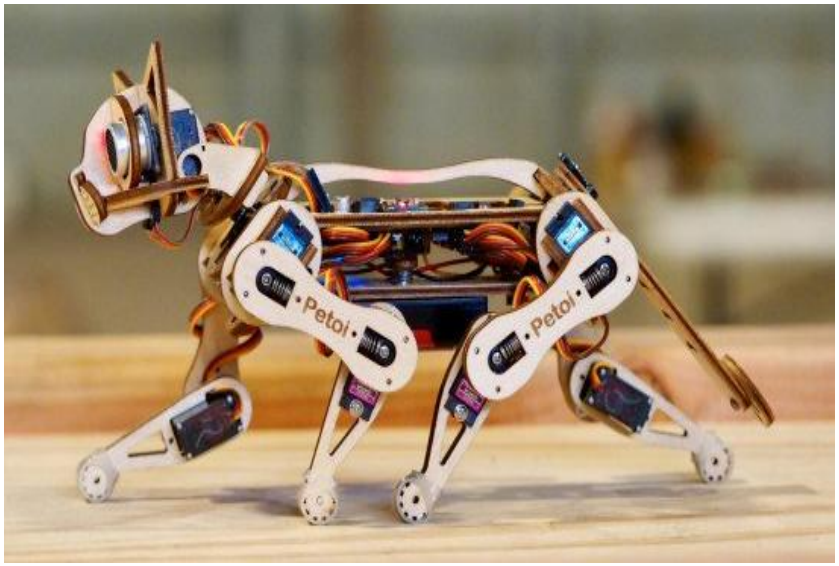
Dc % = (Ton / T)*100

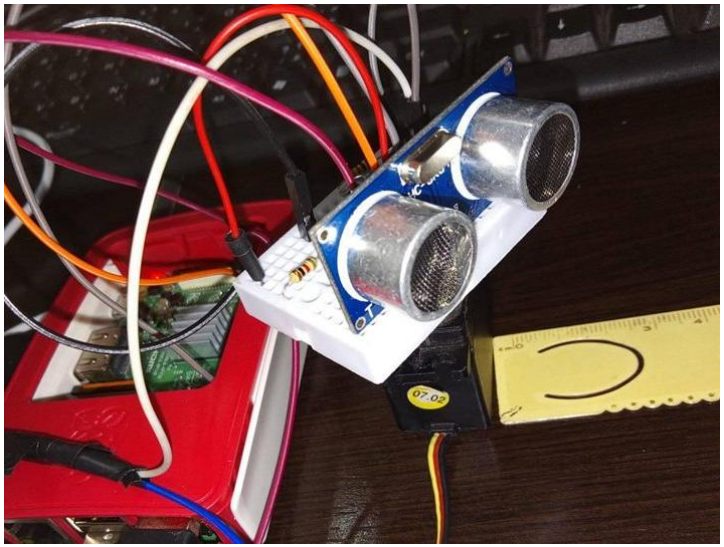
$$dc = \frac{0.5}{20} \times 100 = 2.5\%$$

$$dc = \frac{1.5}{20} \times 100 = 7.5\%$$

DutyCycle = PulseWidth*frequency=.001 *50 = .05 = 5%

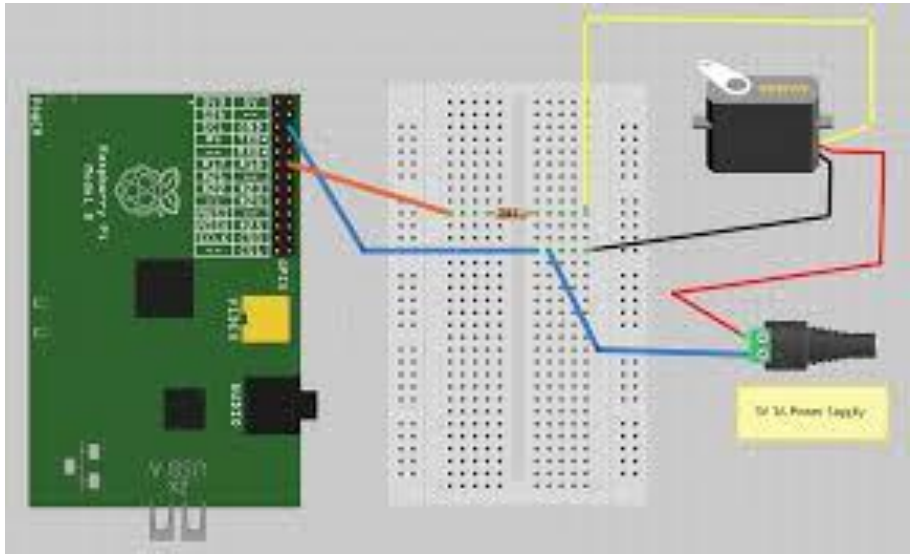
تطبيقات:





مثال ١ :

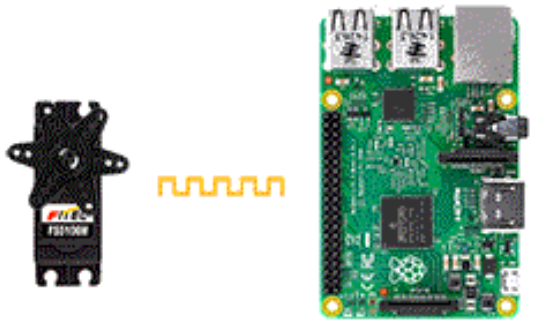
```
import RPi.GPIO as GPIO
import time


GPIO.setmode (GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)
p = GPIO.PWM(11,50)
p.start(7.5)
while True:
    p.ChangeDutyCycle(7.5)    # change duty cycle for getting the servo position to 90º
    time.sleep(1)             # sleep for 1 second
    p.ChangeDutyCycle(12.5)   # change duty cycle for getting the servo position to 180º
    time.sleep(1)             # sleep for 1 second
    p.ChangeDutyCycle(2.5)    # change duty cycle for getting the servo position to 0º
    time.sleep(1)             # sleep for 1 second
```

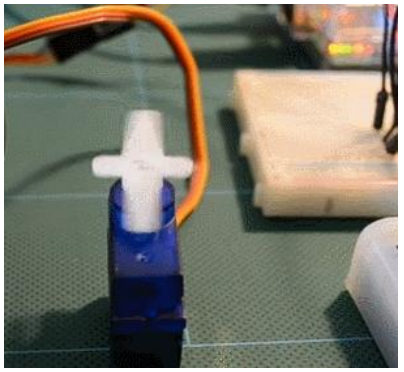


192.168.2.101 (raspberrypi) - VNC Viewer

File Edit Format Run Options Windows

```
import RPi.GPIO as io
import time
io.setmode(io.BOARD)
io.setup(12, io.OUT)
c= io.PWM(12,50)
c.start(7.5)
try:
    while (True):
        print("45")
        c.ChangeDutyCycle(5)
        time.sleep(4)
        print("90")
        c.ChangeDutyCycle(7.5)
        time.sleep(4)
        print("180")
        c.ChangeDutyCycle(12.5)
        time.sleep(5)
        print("0")
        c.ChangeDutyCycle(2.5)
except KeyboardInterrupt:
    pass
c.stop()
io.cleanup ()
```





Homework

قم بتعديل الكود السابق بحيث تظهر العبارة التالية:

“ where do you want the servo? 0 to 180”

وبناء عليها يتم تحديد زاوية دوران محرك السيرفو.

DC Motor Control with Raspberry Pi

Connecting pins

1KΩresistor (3)

Small DC Motor

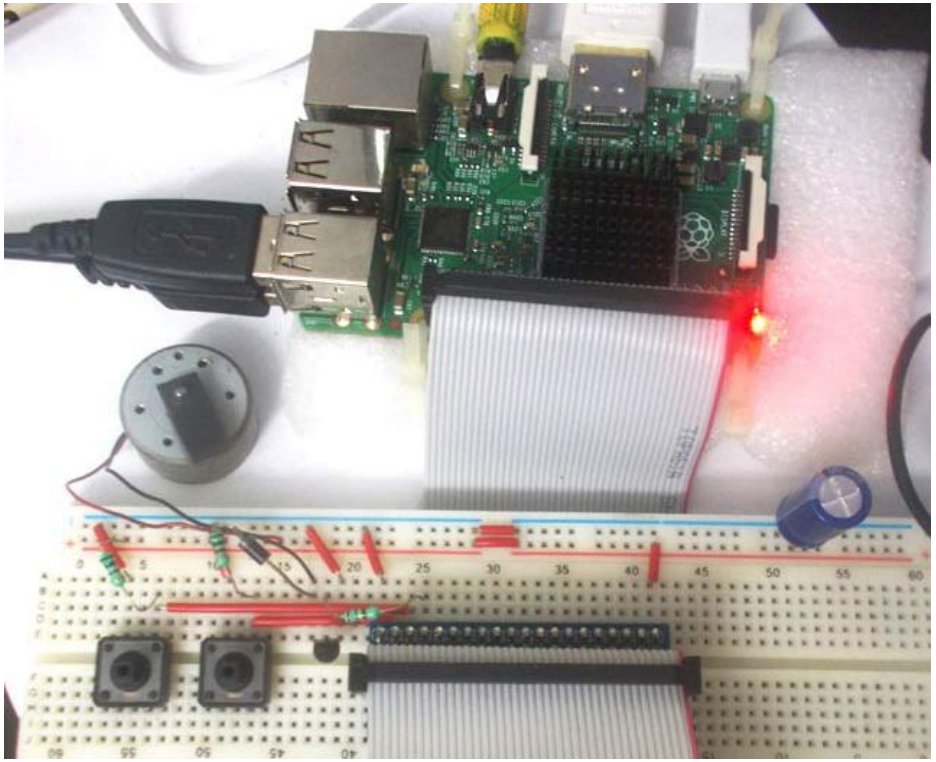
Buttons (2)

2N2222 Transistor

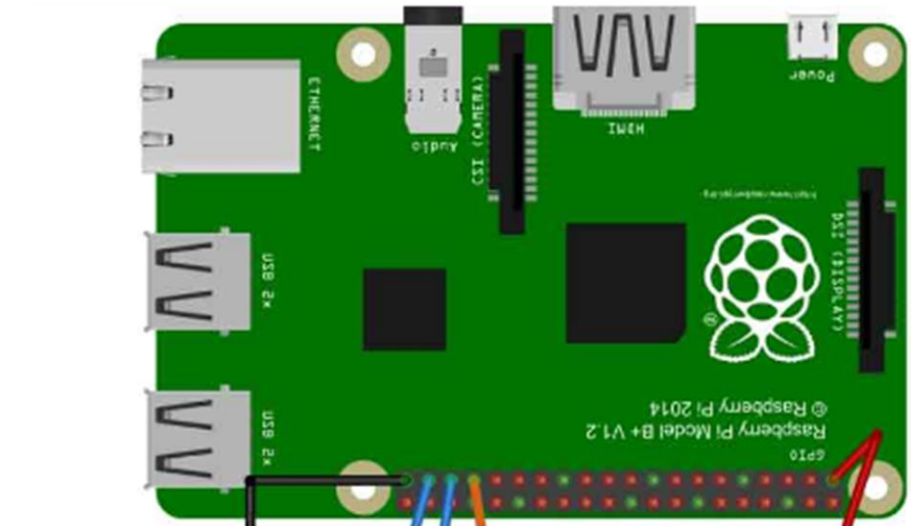
1N4007 Diode

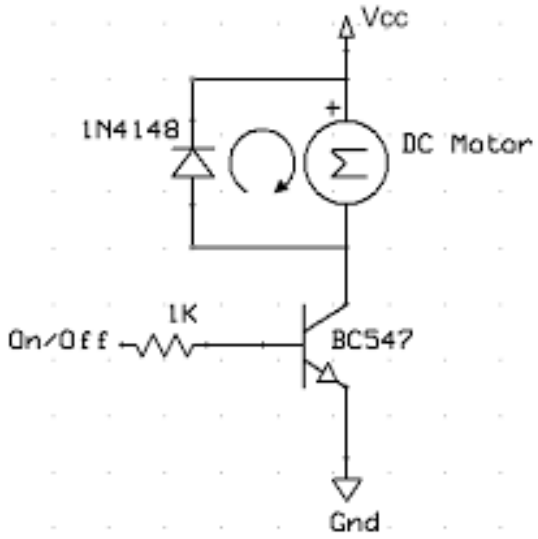
Capacitor- 1000uF

Bread Board

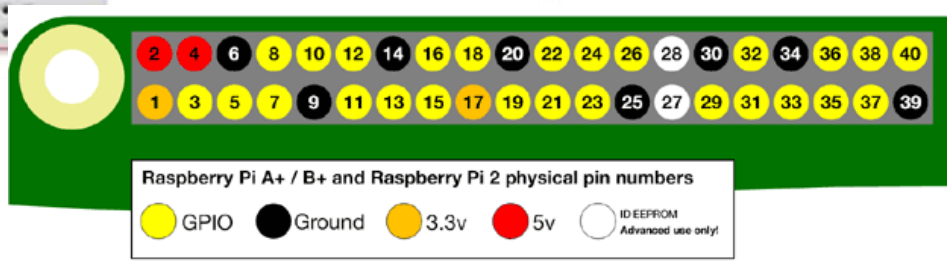


Variable Speed DC Motor





يفضل استخدام بطارية حتى وان كان تيار المحرك صغير



```
import RPi.GPIO as IO          # calling header file which helps us use GPIO's of PI
import time                    # calling time to provide delays in program
IO.setwarnings(False)          #do not show any warnings

x=0                            #integer for storing the duty cycle value
IO.setmode (IO.BOARD)
IO.setup(13,IO.OUT)            # initialize GPIO13 as an output.
IO.setup(19,IO.IN)             # initialize GPIO19 as an input.
IO.setup(26,IO.IN)             # initialize GPIO26 as an input.
p = IO.PWM(13,100)             #GPIO13 as PWM output, with 100Hz frequency
p.start(0)                     #generate PWM signal with 0% duty cycle
while 1:                       #execute loop forever
  p.ChangeDutyCycle(x)
    if(IO.input(26) == False):         #if button1 is pressed
       if(x<50):
         x=x+1                         #increment x by one if x<50
         time.sleep(0.2)               #sleep for 200ms
    if(IO.input(19) == False):         #if button2 is pressed
       if(x>0):
         x=x-1                         #decrement x by one if x>0
         time.sleep(0.2)               #sleep for 200ms
```

```
import RPi.GPIO as IO          # calling header file which helps us use GPIO's of PI
import time                    # calling time to provide delays in program
IO.setwarnings(False)          #do not show any warnings

x=0                            #integer for storing the duty cycle value
IO.setmode (IO.BOARD)
IO.setup(13,IO.OUT)            # initialize GPIO13 as an output.
IO.setup(19,IO.IN,pull_up_down=IO.PUD_UP)
IO.setup(26,IO.IN,pull_up_down=IO.PUD_UP)
p = IO.PWM(13,100)             #GPIO13 as PWM output, with 100Hz frequency
p.start(0)                     #generate PWM signal with 0% duty cycle
while 1:                       #execute loop forever
  p.ChangeDutyCycle(x)
    if(IO.input(26) == False):         #if button1 is pressed
       if(x<50):
          x=x+1                        #increment x by one if x<50
          time.sleep(0.2)              #sleep for 200ms
    if(IO.input(19) == False):         #if button2 is pressed
       if(x>0):
          x=x-1                        #decrement x by one if x>0
          time.sleep(0.2)              #sleep for 200ms
```

تعديل الكود باستخدام مقاومات الرفع برمجيا بدلا من المقاومات الخارجية المرسومة في مخطط الدارة

pp.py - /home/pi/pp.py (3.5.3)

File Edit Format Run Options Window Help

تعديل الكود باستخدام طريقة القفل البرمجي

```
import RPi.GPIO as io
import time
io.setmode(io.BOARD)
io.setwarnings(False)
io.setup(13,io.OUT)
io.setup(7,io.IN,pull_up_down=io.PUD_UP)
io.setup(11,io.IN,pull_up_down=io.PUD_UP)
x=0
v= 0

p=io.PWM(13,50)
p.start(0)
while True:
    p.ChangeDutyCycle(x)
    if((io.input(7)==False) & (v==0)):
        if(x<50):
            x=x+1
            print(x)
            v=1
     if((io.input(11)==False) & (v==0)):
        if(x>0):
            x=x-1
            print(x)
            v=1
    if((io.input(11)==True) & (io.input(7)==True) & (v==1)):
            v=0
io.cleanup()
```

اقتراح وتنفيذ الطلاب:

مازن ركبي

رياض طيفور

محمد العثمان

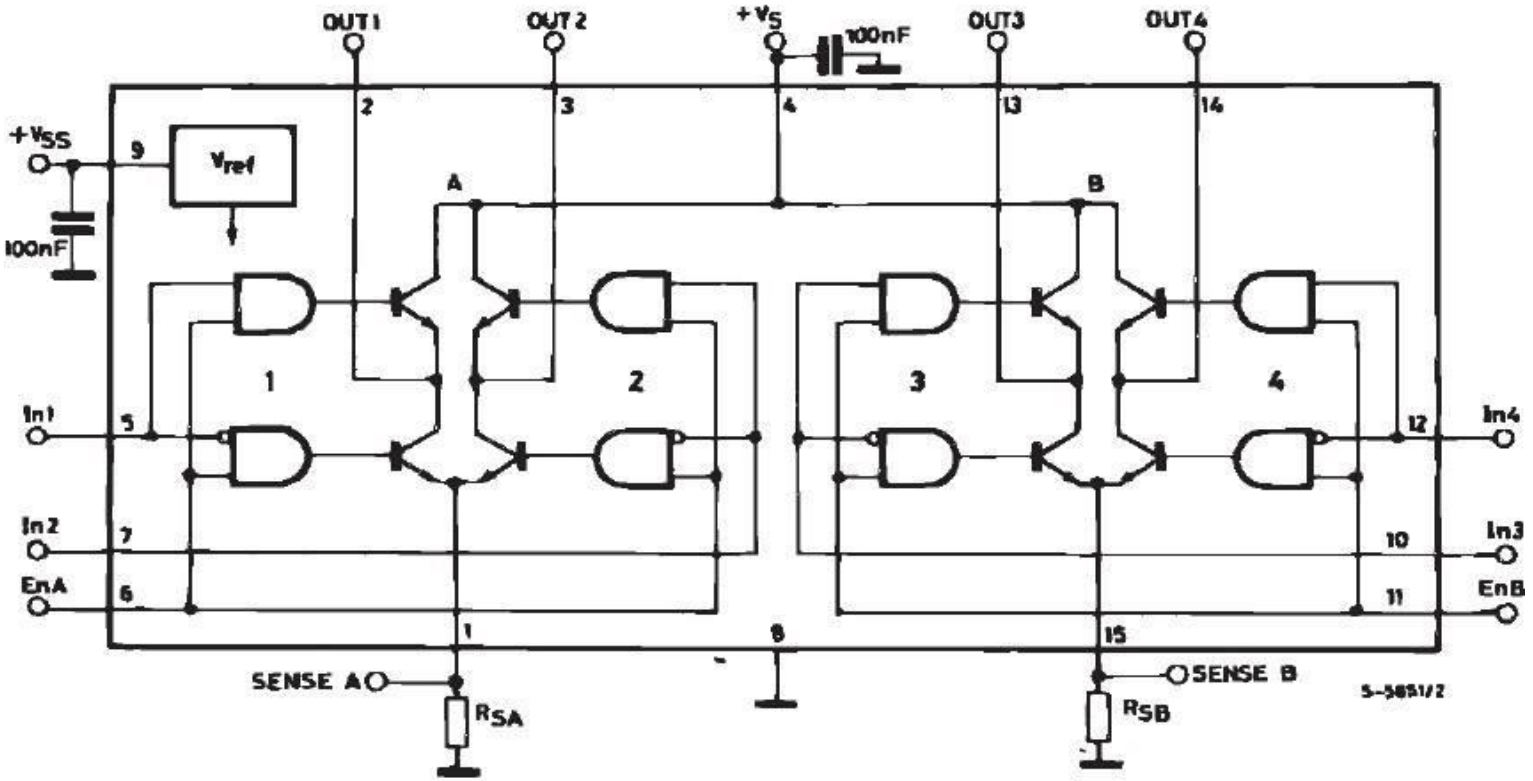
```
import time
import RPi.GPIO as GPIO
Forward=7
Backward=11

GPIO.setmode(GPIO.BOARD)
GPIO.setup(Forward, GPIO.OUT)
GPIO.setup(Backward, GPIO.OUT)
def forward(x):
    GPIO.output(Forward, GPIO.HIGH)
    print("Moving Forward")
    time.sleep(x)
    GPIO.output(Forward, GPIO.LOW)
def reverse(x):
    GPIO.output(Backward, GPIO.HIGH)
    print("Moving Backward")
    time.sleep(x)
    GPIO.output(Backward, GPIO.LOW)
while (1):
    forward(5)
    reverse(5)
GPIO.cleanup()
```

Homework2

ارسم الدارة للبرنامج التالي بعد أن قمنا بتعديل التصميم السابق باستخدام شريحة L298 لتغيير اتجاه دوران المحرك.

لمحة حول دارة القيادة الجسرية L298

البنية الداخلية:



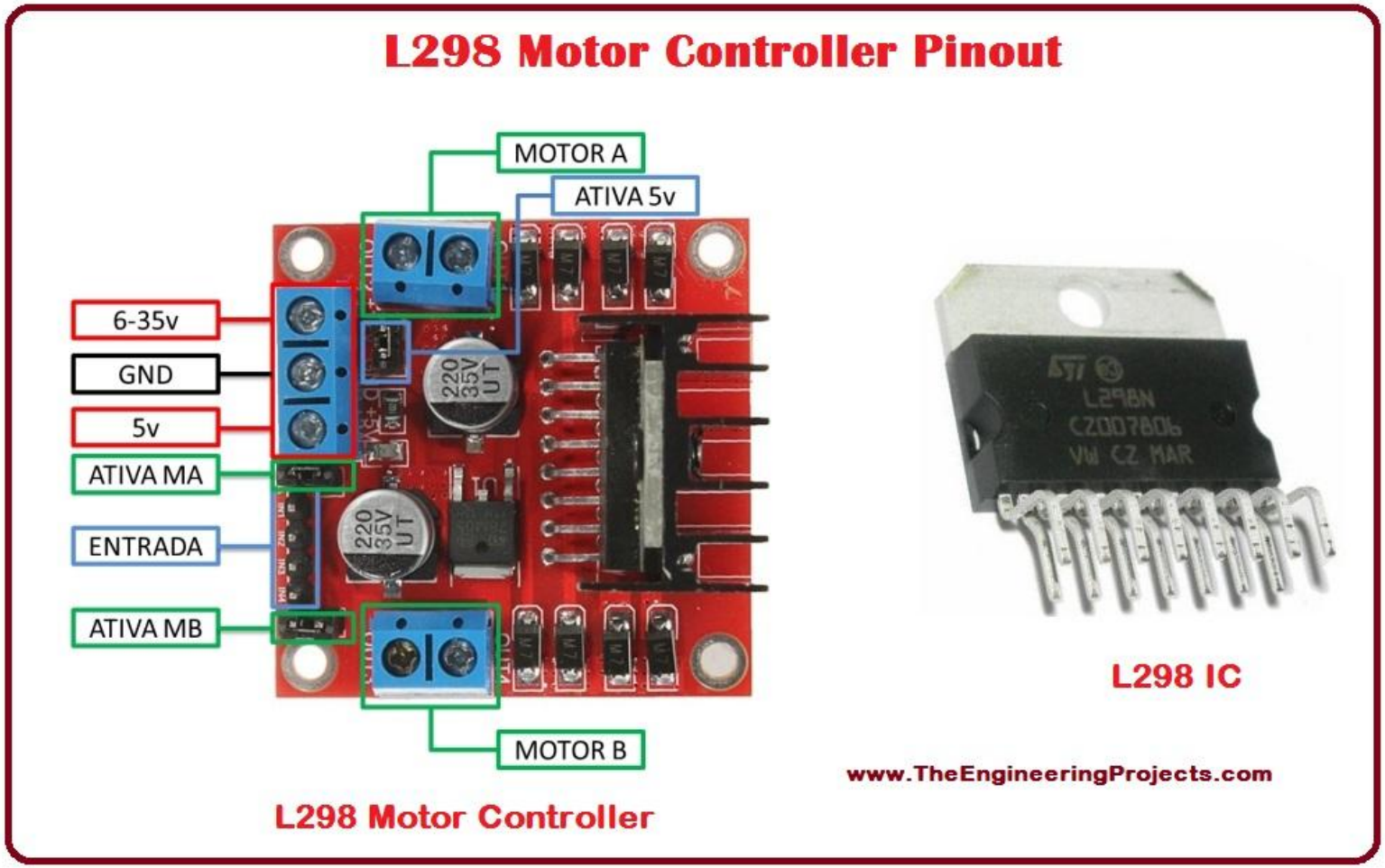
لمحة حول دارة القيادة الجسرية L298

القيم الحدية في النشرة الفنية:

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

لمحة حول دارة القيادة الجسرية L298

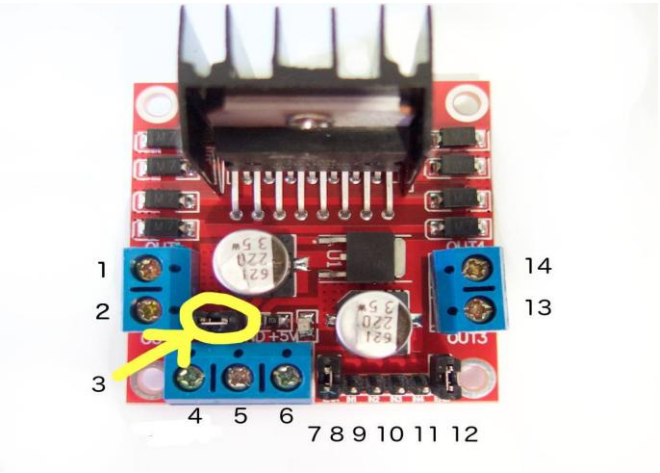


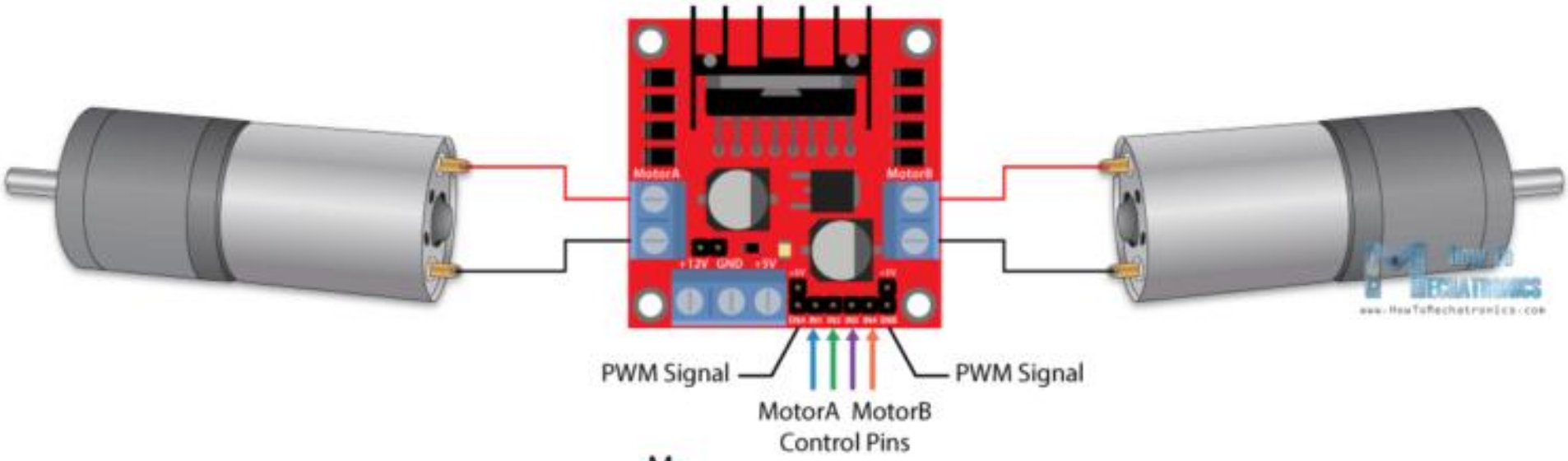
لمحة حول دارة القيادة الجسرية L298

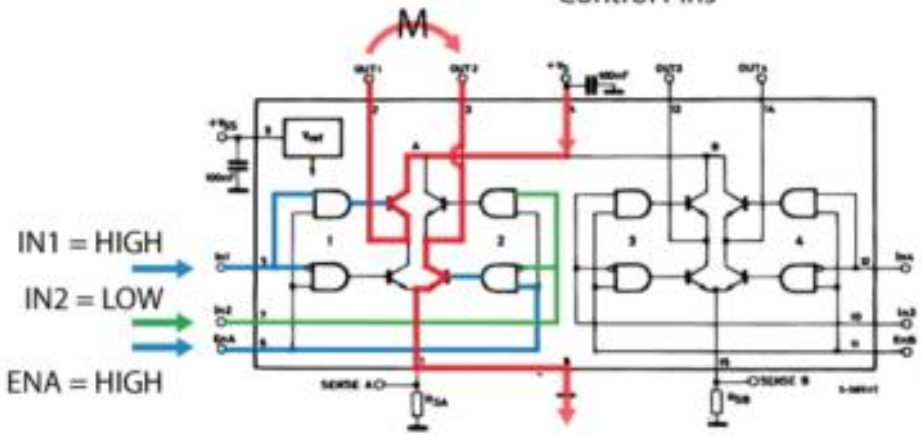
١. قطب التغذية الموجب للمحرك أو الملف الأول +A.
٢. قطب التغذية السالب للمحرك أو الملف الأول (في حالة المحرك الخطوي) A - .
٣. موصل لتفعيل منظم الجهد الداخلي يؤمن جهد بقيمة ٥ فولط للشريحة .
٤. قطب لتغذية المحركين من منبع جهد خارجي.

ملاحظة: مسموح لغاية ٤٦ فولط في حال استخدام الشريحة العادية مع تبريد مناسب، أما باستخدام الموديول الموضح في الصورة يمكن تغذيتها ضمن المجال(٦-٣٥) فولط، ملاحظة عند استخدام جهد اكبر من ١٢ يجب فصل الموصل بالقطب رقم ٣.

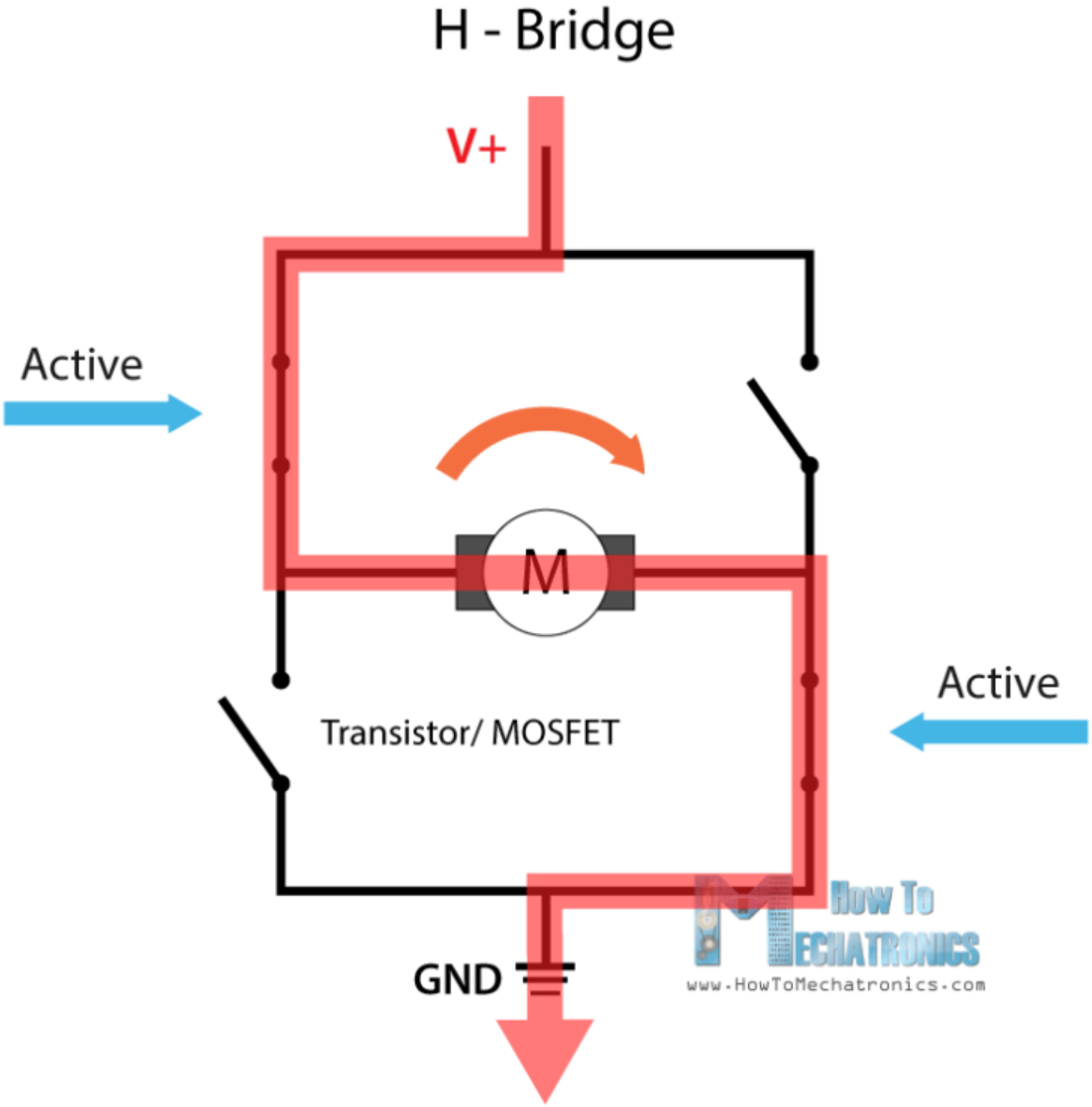
٥. قطب الأرضي.
٦. خرج منظم الجهد الداخلي ٥ فولط.
٧. موصل تغذية الملف الأول.
٨. مدخل نبضات الملف الأولin1 للمحرك الأول أو الملف الأول +A.
٩. مدخل نبضات الملف الثاني in2 للمحرك الأول أو الملف الأول A- .
١٠. مدخل نبضات الملف الأول in3للمحرك أو الملف الثاني+B .
١١. مدخل نبضات الملف الثاني in4للمحرك أو الملف الثاني-B.
١٢. موصل تغذية الملف الثاني.
١٣. قطب التغذية الموجب للمحرك أو الملف الثاني +B .
١٤. قطب التغذية السالب للمحرك أو الملف الثاني-B.





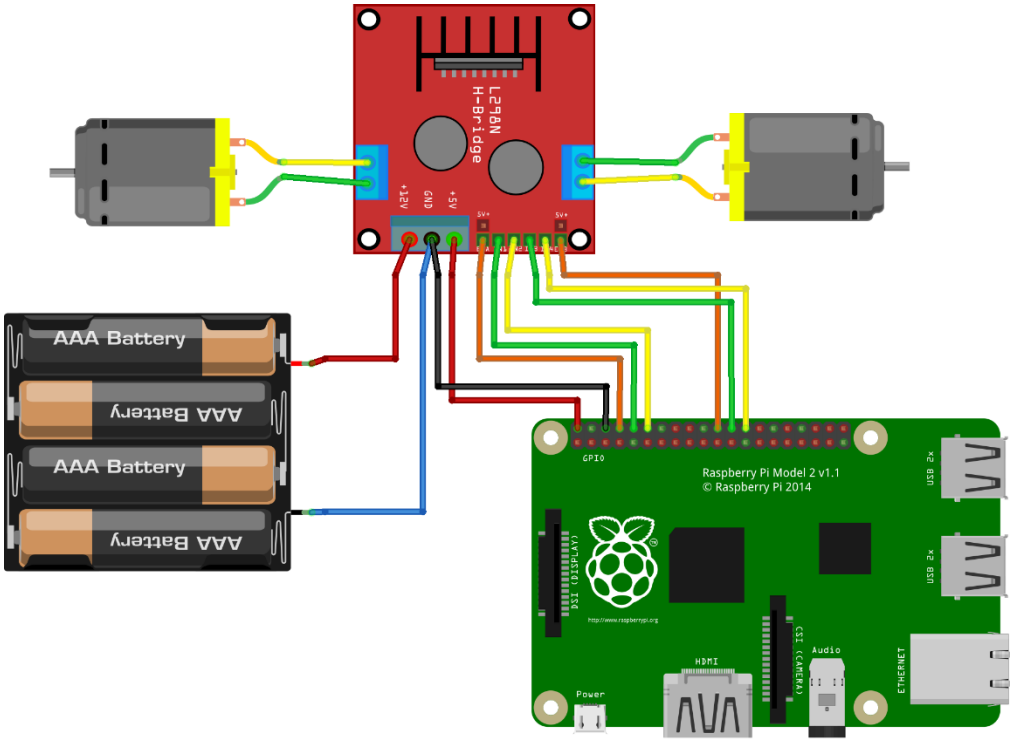


L298N Block Diagram

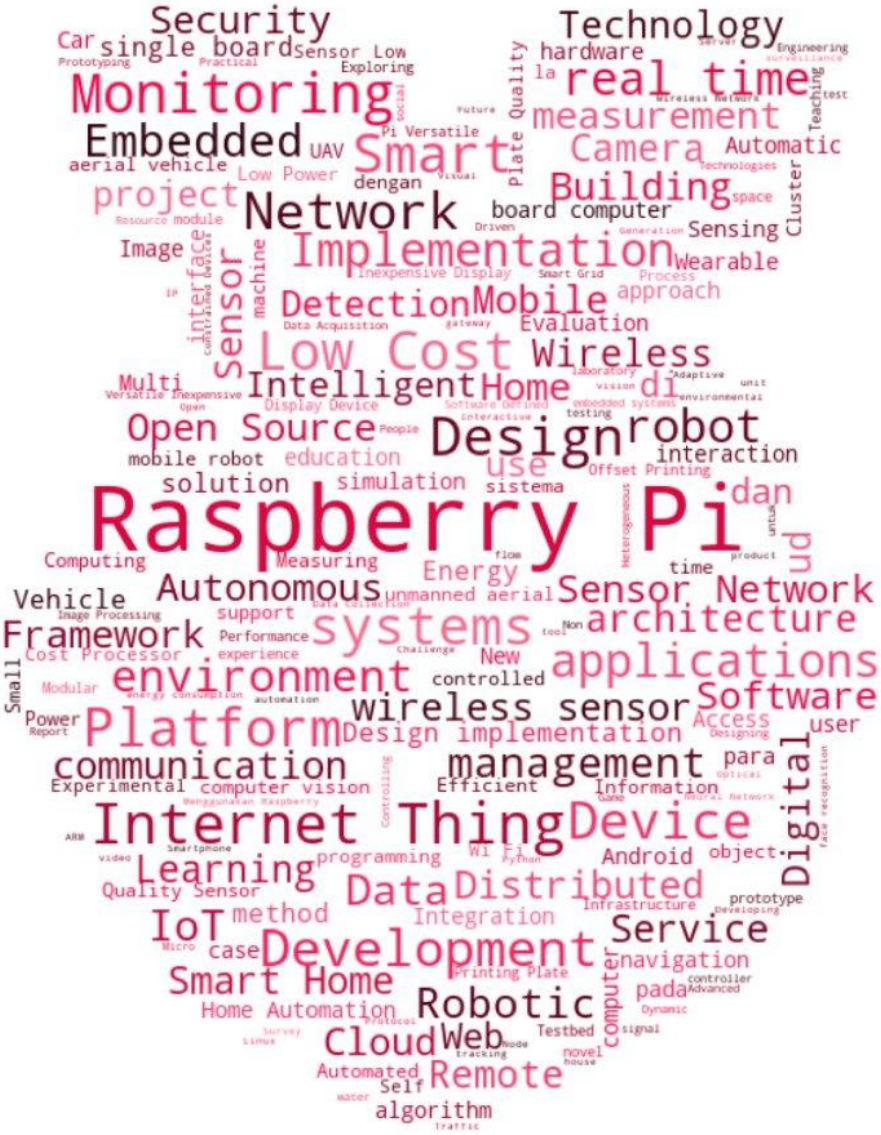

H - Bridge
V+
Active
M
Active
Transistor/ MOSFET
GND
How To
MECHATRONICS
www.HowToMechatronics.com

Homework3

اكتب برنامج للتحكم بمحركين تيار مستمر مثبتين على عربة روبوت باستخدام شريحة L298 لتحريك الروبوت إلى الأمام والخلف باستخدام إشارات PWM وذلك بعرض نبضة 50%.



للمساعدة في حل الوظائف يمكنكم مراجعة المخبر


Security
Technology
Car
single board
Sensor Low
Exploring
hardware
real time
Monitoring
measurement
Embedded
UAV
Smart
Camera
Automatic
aerial vehicle
Low Power
dengan
Building
project
Network
board computer
Sensing
Implementation
Wearable
Image
Sensor
Detection
Mobile
approach
Evaluation
Low Cost
Wireless
Multi
Intelligent
Home
Open Source
Design
robot
mobile robot
education
interaction
use
solution
simulation
sistema
dan
Raspberry Pi
Computing
Measuring
Energy
time
Autonomous
unmanned aerial
Sensor Network
Vehicle
support
systems
architecture
Framework
Performance
Cost Processor
experience
New
applications
environment
controlled
Software
wireless sensor
Power
Platform
Access
Design implementation
user
communication
management
para
Experimental
computer vision
Efficient
Information
Internet Thing
Device
Digital
Learning
programming
Android
object
Data
Distributed
Quality Sensor
prototype
IoT
method
Integration
Service
case
Development
navigation
Smart Home
Robotic
pada
Home Automation
Cloud
Web
Testbed
computer
Remote
novel
Automated
Self
algorithm

Thank You