ติดตั้ง geanny เพิ่ม

ติดตั้ง Module i2C **sudo apt-get install -y python-smbus i2c-tools**

ติดตั้ง WiringPi

ติดตั้ง WiringPi2

git clone https://github.com/Gadgetoid/WiringPi2-Python.git

cd WiringPi2-Python

sudo python3 setup.py

คำสั่งที่ใช้งานบ่อย

apt-get update อัพเดตเวอร์ชั่น Raspbian

apt-get upgrade อัพเกรดซอฟต์แวร์ทั้งหมดที่ติดตั้ง

clear เคลียร์หน้าต่าง Teminal

date แสดง วันเวลา ปัจจุบัน

```
pi@raspberrypi ~ $ date
Mon Apr  6 16:44:32 ICT 2015
pi@raspberrypi ~ $
```

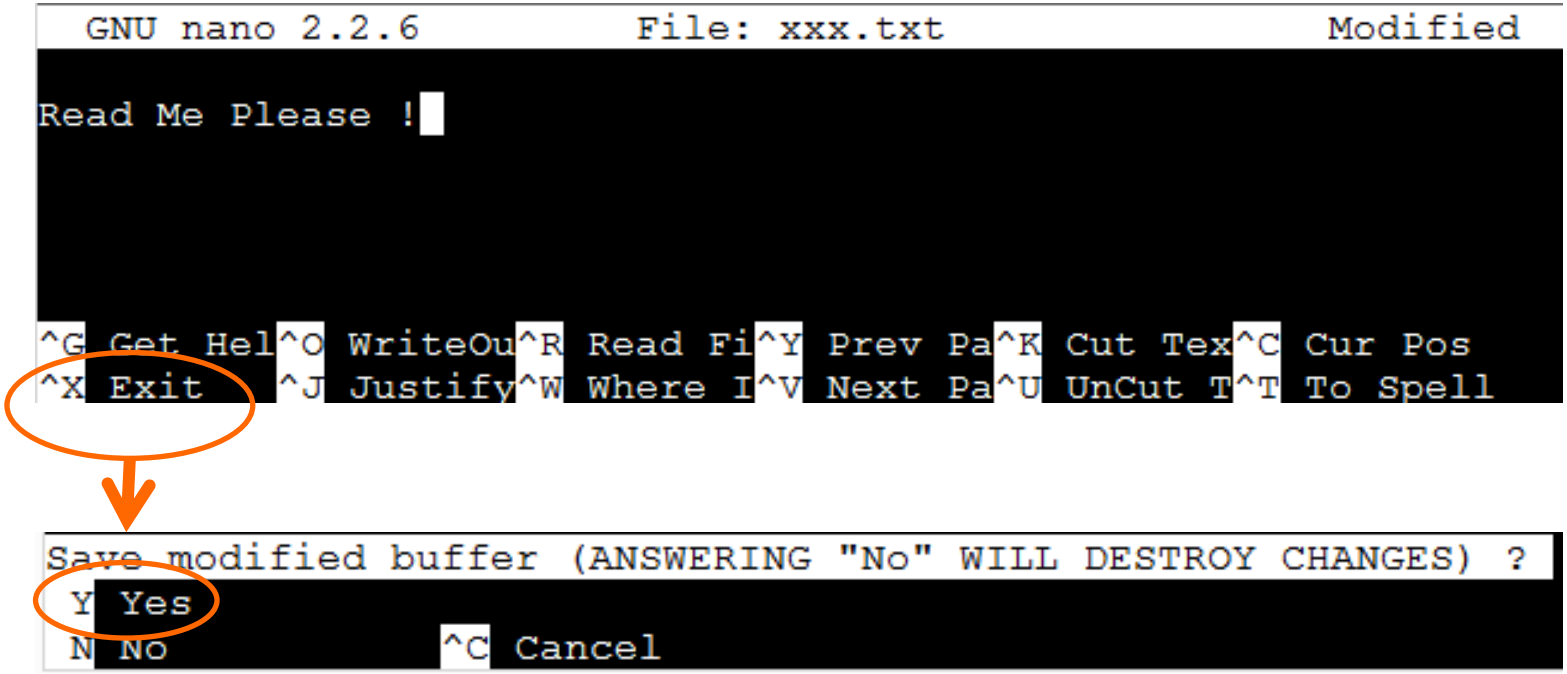
find -name *.* ค้นหาชื่อไฟล์ที่ต้องการในทุกโฟลเดอร์

```
pi@raspberrypi ~ $ find -name *.txt
./python_games/starPusherLevels.txt
./.arduino/preferences.txt
pi@raspberrypi ~ $
```

คำสั่งที่ใช้งานบ่อย

touch xxx.txt สร้างไฟล์ใหม่

nano xxx.txt เปิดไฟล์ด้วย Text Editor

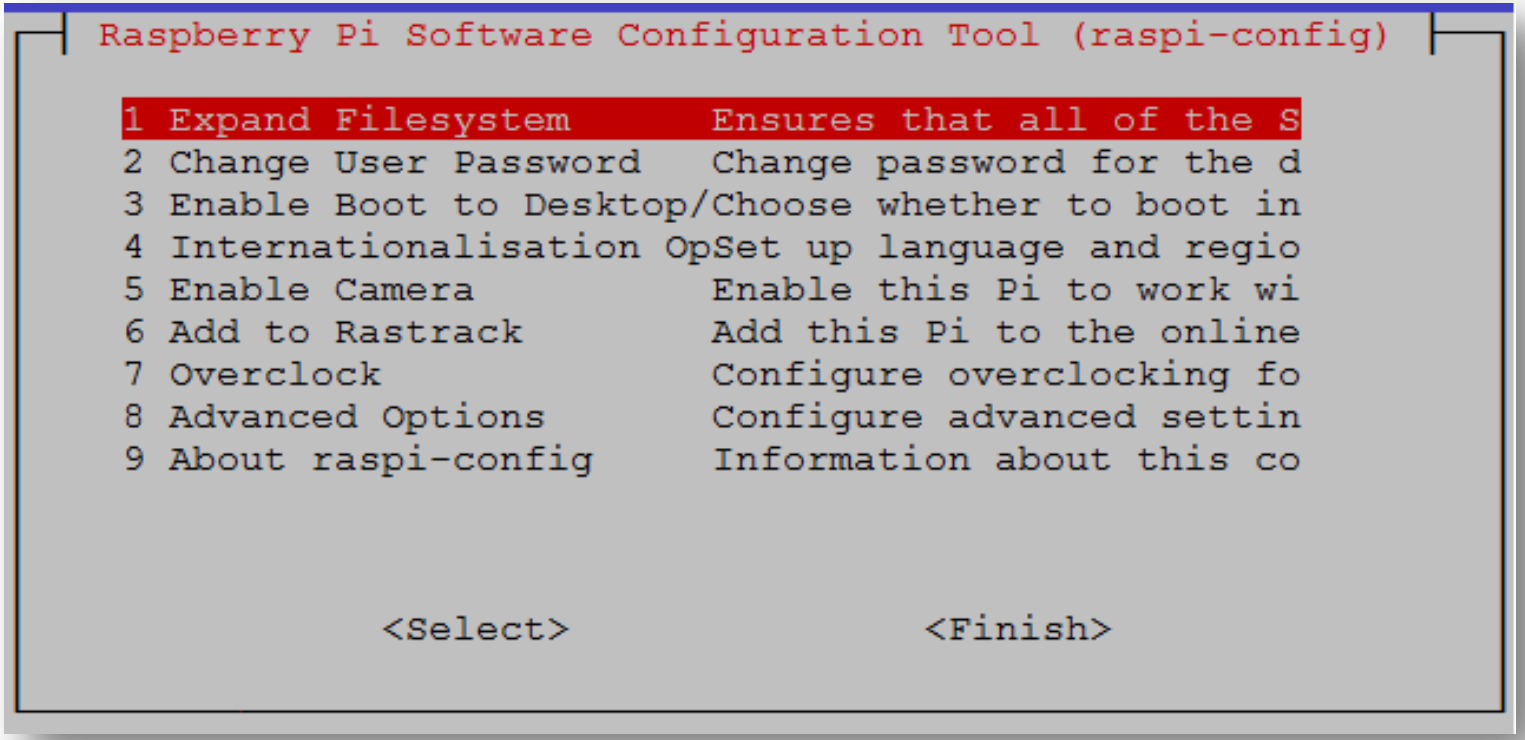


คำสั่งที่ใช้งานบ่อย

`sudo poweroff`	ปิดเครื่องแบบทันที ทันใด
`sudo reboot`	รีสตาร์ตเครื่องใหม่
`startx`	เปิดหน้าต่าง GUI ของ R-Pi
`startlxde`	เปิดหน้าต่าง GUI ของ R-Pi (Remote)

คำสั่งที่ใช้งานบ่อย

sudo raspi-config เปิดหน้าต่างตั้งค่าของ R-Pi



คำสั่งที่ใช้งานบ่อย

File/Directory Basics

ls	List files	แสดงรายชื่อไฟล์และไดเร็คทอรี่
cp	Copy files	สำเนาไฟล์
mv	Rename files	เปลี่ยนชื่อไฟล์
rm	Delete files	ลบไฟล์
cd	Change directory	ย้ายไปยังไดเร็คทอรี่ที่ต้องการ
pwd	Print directory name	แสดงชื่อไดเร็คทอรี่ปัจจุบัน
mkdir	Create directory	สร้างไดเร็คทอรี่ใหม่
rmdir	Delete directory	ลบไดเร็คทอรี่ (ที่ว่างเปล่าเท่านั้น)

คำสั่งที่ใช้งานบ่อย

File Viewer

cat	View files	**ดูเนื้อหาของ text file**
less	Page trough files	**เลื่อนดูเนื้อหาของไฟล์** **ออกก่อน Crl+Z**
head	View file beginning	**แสดงส่วนต้นของไฟล์**
tail	View files ending	**แสดงส่วนท้ายของไฟล์**
nl	Number lines	**แสดงหมายเลขบรรทัด**
od	View binary files	**แสดงเนื้อหาในไฟล์ไบนารี่**

คำสั่งที่ใช้งานบ่อย

tee

Copy stdin to file and to stdout simultaneously

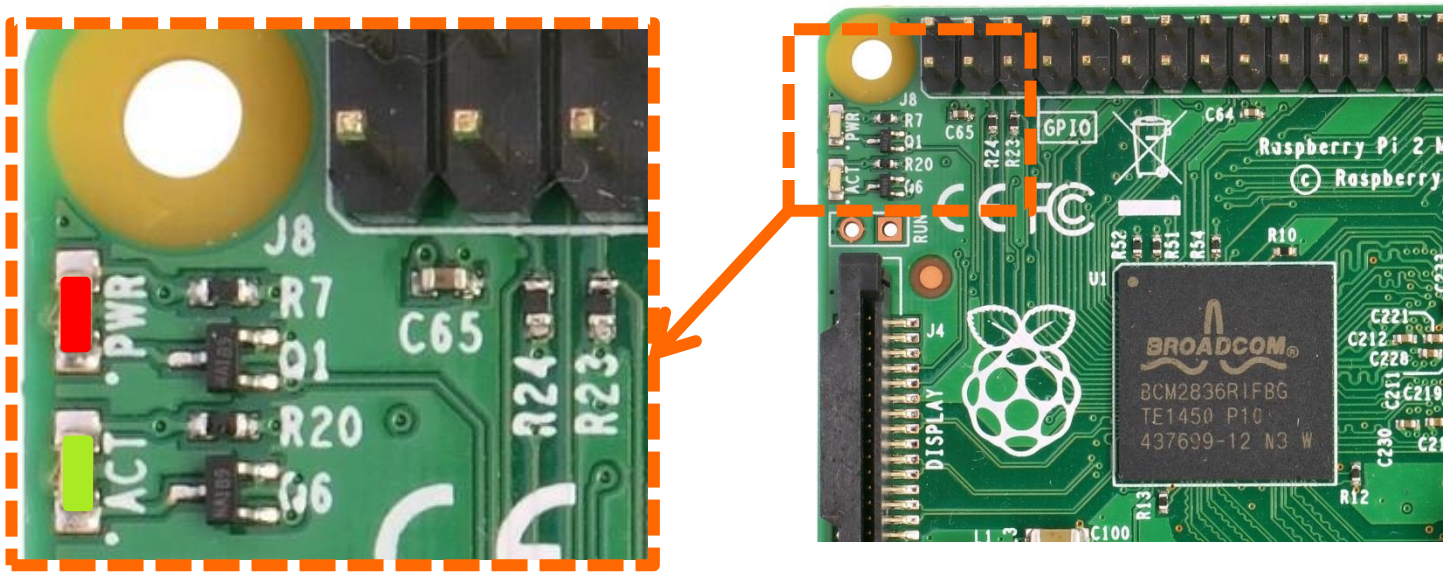
สำเนาข้อความออกทางไฟล์และ stdout พร้อม ๆ กัน

```
echo "Hello, world" | tee hello.txt
Hello, world
```

รูปแบบการติดต่อ LED ACT

Linux ใช้การตั้งค่า Config ต่าง ๆ ผ่าน Text ไฟล์ใน /etc/

การเปลี่ยนสถานะของอุปกรณ์ระบบ ทำกับไฟล์เสมือน จะถูกเก็บใน /sys/ และ /proc/



LED ACT บน R-Pi

```
cat /sys/class/leds/led0/trigger
```

```
pi@raspberrypi ~/test $ cat /sys/class/leds/led0/trigger
none [mmc0] timer oneshot heartbeat backlight gpio cpu0 cpu1 cpu2 cpu3 default-on input
```

ปกติโชว์ค่าการอ่านเขียน SD-CARD

```
echo "timer"|sudo tee /sys/class/leds/led0/trigger
```

```
pi@raspberrypi ~/test $ echo "timer" | sudo tee /sys/class/leds/led0/trigger
timer
pi@raspberrypi ~/test $ cat /sys/class/leds/led0/trigger
none mmc0 [timer] oneshot heartbeat backlight gpio cpu0 cpu1 cpu2 cpu3 default-on input
```

ย้ายมาโชว์ค่า Timer แทน

LED ACT กะพริบ

LED ACT บน R-Pi

```
echo "none"|sudo tee /sys/class/leds/led0/trigger
```

สั่ง LED ไม่ให้ถูกควบคุมจากใคร

```
echo "1" |sudo tee /sys/class/leds/led0/brightness
```

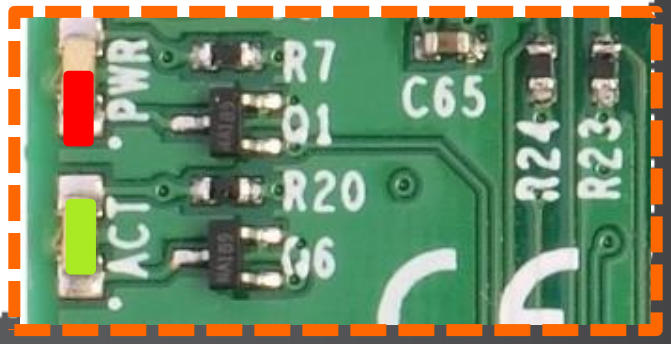
สั่ง LED ติด

```
echo "0" |sudo tee /sys/class/leds/led0/brightness
```

สั่ง LED ดับ

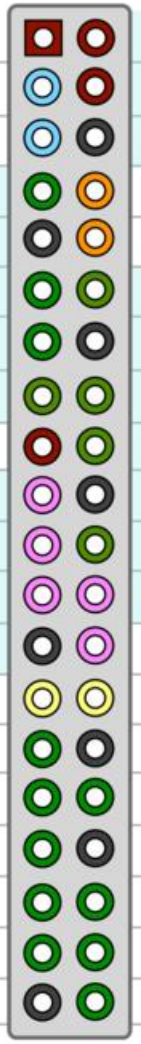
```
echo "mmc0"|sudo tee /sys/class/leds/led0/trigger
```

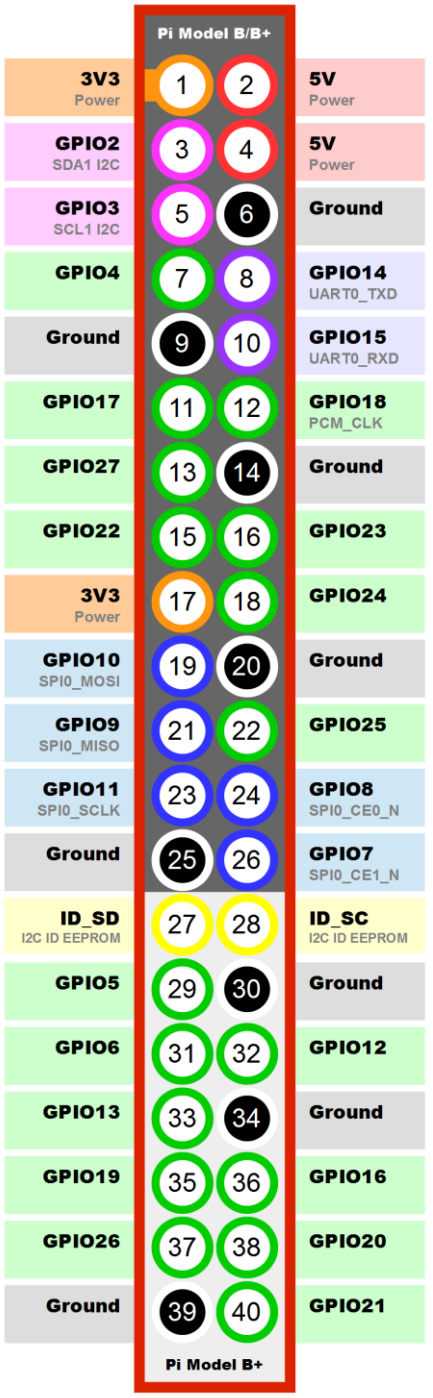
คืนค่าเดิม (แสดงสถานะ อ่านเขียน SDCARD



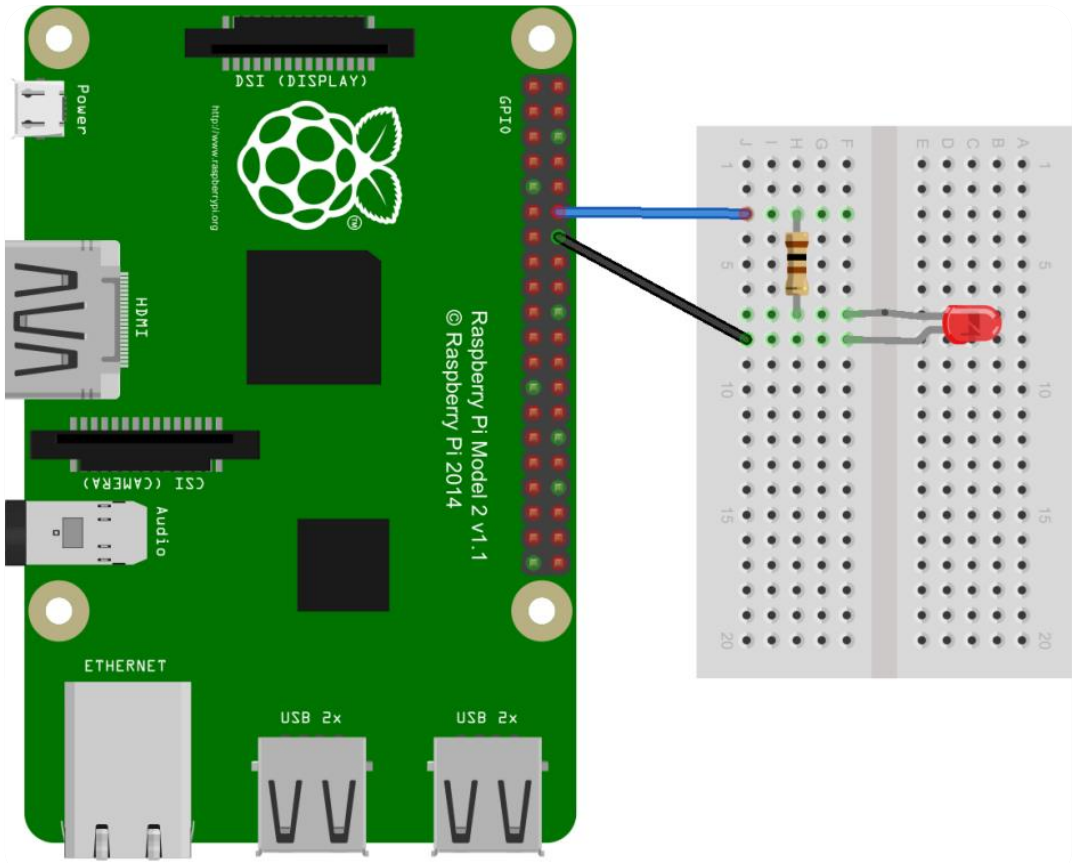
GPIO Pinout

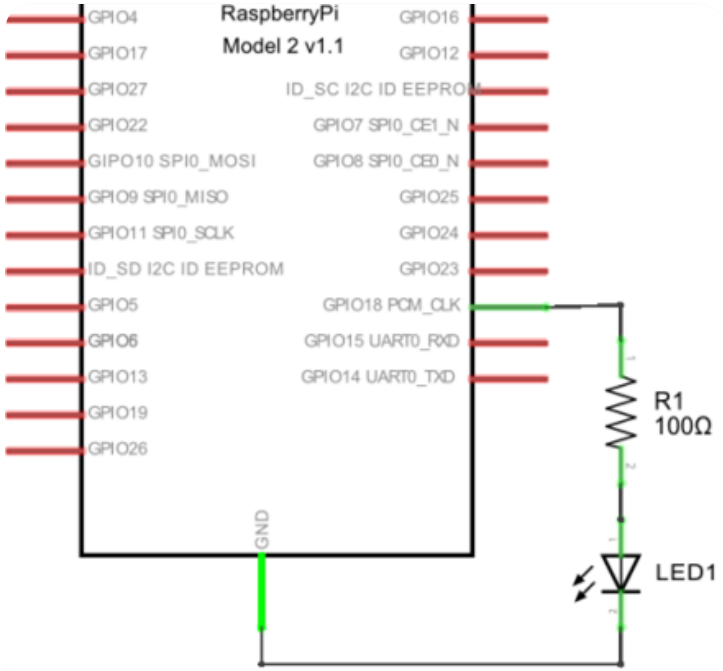
Pin#	NAME	Connection	Connection	NAME	Pin#
01	3.3V		5V (Cupcade)	5V	02
03	GPIO 2		5V (Powerboost)	5V	04
05	GPIO 3		GND (Powerboost)	Ground	06
07	GPIO 4	START		GPIO 14	08
09	Ground	GND (Cupcade)		GPIO 16	10
11	GPIO 17	UP	SELECT	GPIO 18	12
13	GPIO 27	DOWN	GND (Select/Start)	Ground	14
15	GPIO 22	LEFT	RIGHT	GPIO 23	16
17	3.3V		A	GPIO 24	18
19	GPIO 10	B	GND (ABXYR)	Ground	20
21	GPIO 09	X	Y	GPIO 25	22
23	GPIO 11	L Shoulder	R Shoulder	GPIO 08	24
25	Ground	GND (L)		GPIO 07	26
27	ID_SD			ID_SC	28
29	GPIO 05			Ground	30
31	GPIO 06			GPIO 12	32
33	GPIO 13			Ground	34
35	GPIO 19			GPIO 16	36
37	GPIO 26			GPIO 20	38
39	Ground			GPIO 21	40





เปิด/ปิด LED GPIO18





ติดต่อ GPIO โดยตรง

เพื่อขอสิทธิ์เข้าใช้งานเป็น Super User จะต้องพิมพ์

```
sudo su
```

```
pi@raspberrypi ~ $ sudo su
root@raspberrypi:/home/pi#
```

```
echo  18 |sudo tee /sys/class/gpio/export
```

สร้าง ส่วน Export การติดต่อกับ GPIO18 โฟลเดอร์ GPIO18 จะถูกสร้าง

```
cd /sys/class/gpio/gpio18
```

เข้าไปโฟลเดอร์ GPIO18

```
root@raspberrypi:/home/pi# cd /sys/class/gpio/gpio18
root@raspberrypi:/sys/class/gpio/gpio18# ls
active_low  device  direction  edge  power  subsystem  uevent  value
```

ภายในโฟลเดอร์จะมีไฟล์ Text สำหรับ Config ค่าต่าง ๆ

```
echo out > direction
```

กำหนดให้ GPIO18 เป็นเอาต์พุต

```
echo 1 > value
```

กำหนดให้ GPIO18 เป็น "1" LED ติด

```
echo 0 > value
```

กำหนดให้ GPIO18 เป็น "0" LED ดับ

แก้ไขโปรแกรม PYTHON

เปิด editor

```
nano prog1.py
```

เขียนโค้ด

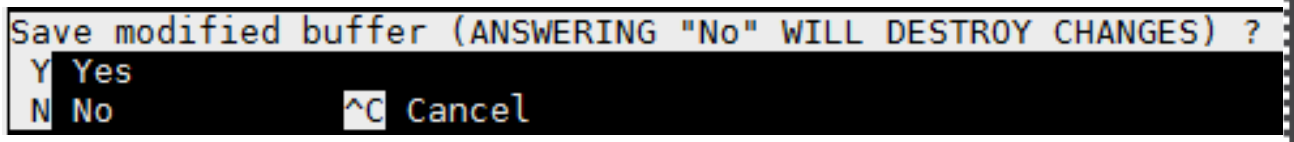
```
print ("Innovative")
print ("Experiment")
```

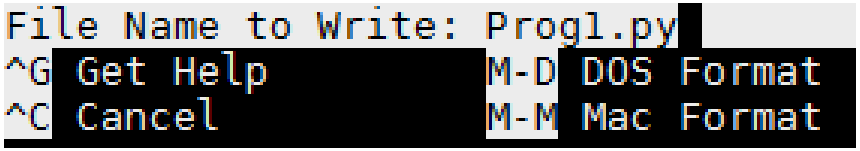
บันทึก แล้วออก

CTRL + X

Y

ENTER





รันโปรแกรมด้วย python3

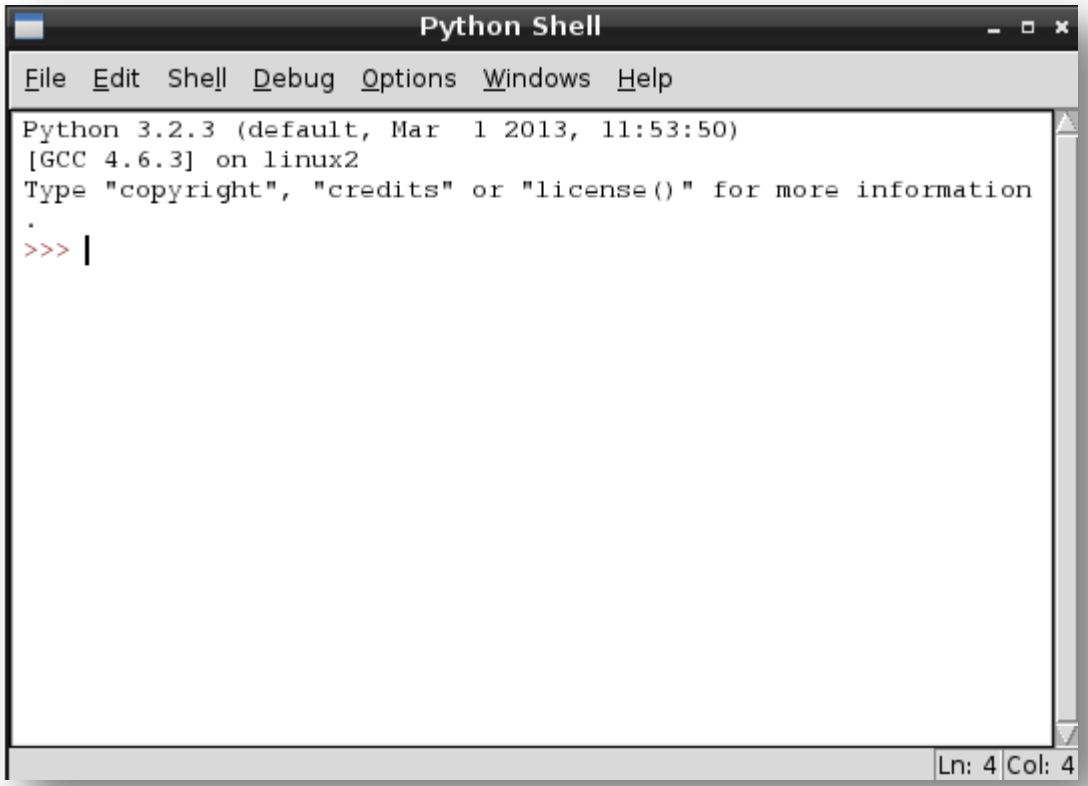
รันโปรแกรม

```
sudo python3 Prog1.py
```

```
pi@raspberrypi ~ $ sudo python3 Prog1.py
Innovative
Experiment
```

ผลลัพธ์

ฝึก Python ด้วย Python Shell



บวก ลบ คูณ หาร

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5
1.6
>>> 17 / 3
5.666666666666667
>>> 5 ** 2
25
```

ผลลัพธ์จากการบวกเป็น int

ผลลัพธ์จากการหารเป็น Float

ยกกำลัง

เครื่องหมาย = ใช้กำหนดตัวแปร

```
>>> x = 100
>>> y = 20
>>> x+y
120
```

```
>>> x = 3.0
>>> y = 5
>>> x*y
15.0
```

ฟังก์ชั่นคณิตศาสตร์ภายใน ที่สำคัญ

```
range(start, end [,step])
```
กำหนดค่าช่วงของตัวเลข

```
 range (1,6) # จะประกอบด้วย 1,2,3,4,5
```

```
sum(n1,n2 [,nn])
```
หาผลรวมทั้งหมดในชุด

```
 sum(1,2,3,4,5) # ผลลัพธ์คือ 15
```

```
round(var, digits)
```
กำหนดจำนวนทศนิยม

```
 round(1234.56789, 2) # ผลลัพธ์คือ 1234.57
```

```
min(var), max(var)
```
หาค่าต่ำสุด สูงสุด ของเลขในชุด

```
 min(range(5,10)) # ผลลัพธ์คือ 5
 max(range(5,10)) # ผลลัพธ์คือ 9
```

```
abs(var)
```
แสดงค่าสัมบูรณ์ (เปลี่ยนเป็นค่าบวก)

```
 abs(-50) # ผลลัพธ์คือ 50
```


```
abs(var)
```
แสดงค่าสัมบูรณ์ (เปลี่ยนเป็นค่าบวก)

```
 abs(-50) # ผลลัพธ์คือ 50
```

ตัวแปรภายใน Python

ใช้คำสั่ง `type (var)` ตรวจสอบชนิดตัวแปร

```
>>> x = 400
```
x เป็น Integer

```
>>> y = 4.5
```
y เป็น Float

```
>>> Z = True
```
Z เป็น Boolean

```
>>> S = 'Rasp'
```
S เป็น String

Python ไม่ใช้ Array ใช้ List แทน หลายชนิดอยู่ที่เดียวกันได้

```
>>> i = [1, 2, 3, 4, 'Rasp', 5, 'Pi']
```
i เป็น List

```
>>> print (i[4])
Rasp
>>> print (i[2:6])
[3,4,'Rasp',5]
```

การเปรียบเทียบ

`>`	มากกว่า
`>=`	มากกว่าหรือเท่ากับ
`<=`	น้อยกว่าหรือเท่ากับ
`<`	น้อยกว่า
`==`	เท่ากับ
`!=`	ไม่เท่ากับ
`in`	ตรงกับค่าใน List หรือเปล่า

AND	มากกว่า
OR	มากกว่าหรือเท่ากับ
NOT	น้อยกว่าหรือเท่ากับ

ตัวอย่าง

```
>>> i = [1, 2, 3, 4, 'Rasp', 5, 'Pi']
>>> 3 in i
>>> True
```

การควบคุมทิศทางของโปรแกรม

1. การตัดสินใจ (if)
2. การวนทำซ้ำ (loop)
3. การจัดการความผิดปกติของโปรแกรม (error)

การตัดสินใจ : IF

```
if Condition:
   Statements
   .....
   .....
else:
   Statements
   .....
   .....
```

ตัวอย่าง

```
>>> x = 1
>>> if x is 1:
>>>     print 'Yes'
>>> else:
>>>     print 'No'
```

การตัดสินใจ : if elif else

```
if Condition:
   Statements
   .....
   .....
elif Condition:
   Statements
   .....
   .....
else:
   Statements
   .....
   .....
```

ตัวอย่าง

```
>>> x = 3
>>> if x < 1:
>>>    print 'x < 1'
>>> elif x > 1:
>>>    print 'x > 1'
>>> else:
>>>    print 'x = 1'
```

การวนทำซ้ำ loop : while

```
while Condition:
   Statements
   .....
   .....
```

ตัวอย่าง

```
>>> x = 1
>>> while x < 5:
>>>   print 'Yes\n'
>>>   x+=1
```

```
s='0'
while s!='x':
   s = input("input")
   print (s)
```

การวนทำซ้ำ loop : for

```
for var in range(m, n [, step = 1]):
   Statements
   .....
   .....
```

ตัวอย่าง

```
>>> for i in range(0, 10):
>>>     if i % 2 == 0:
>>>         print i, 'is even '
>>>     else:
>>>         print i, 'is odd '
```

การดักจับ error

```
try:
   Standard operation
except:
   Error operation
finally:
   End operation
```

```
>>> try:
       x = y
    except:
       print 'y not defined'

y not defined
```

```
try:
   main()
finally:
   GPIO.cleanup()
   print("END")
```

การสร้างฟังก์ชั่น

```
def function_name( [Argument] ):
     Statement
     ……
     [return]
```

def	บอกว่าเป็นฟังก์ชั่น
function_name	ชื่อฟังก์ชั่น
Argument	ค่าที่ส่งเข้าไปในฟังก์ชั่น
Statement	ชุดคำสั่ง
return	การคืนค่าของฟังก์ชั่น

```
def add(x,y):
     return x + y
```

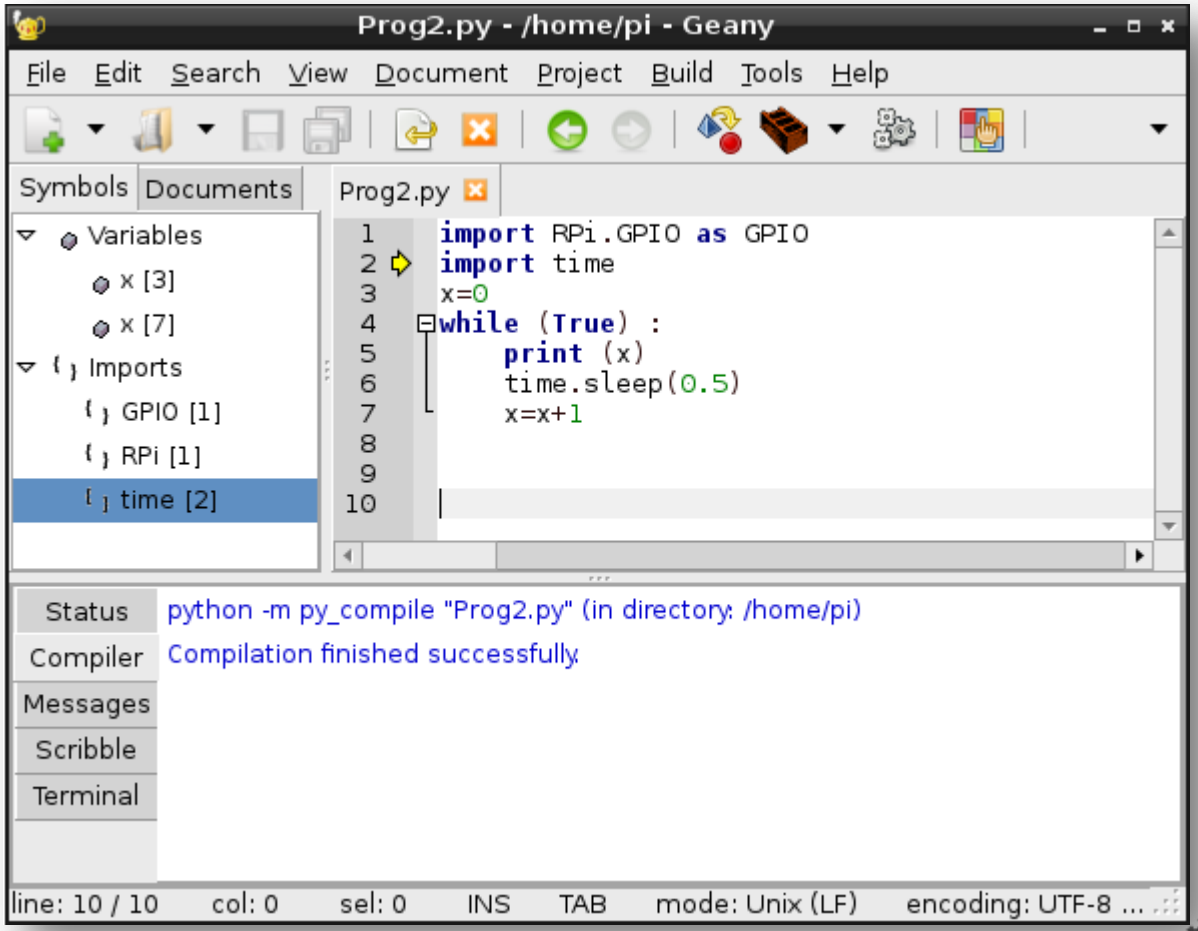
```
add(3,4)
7
```

Graphic Mode : Geany

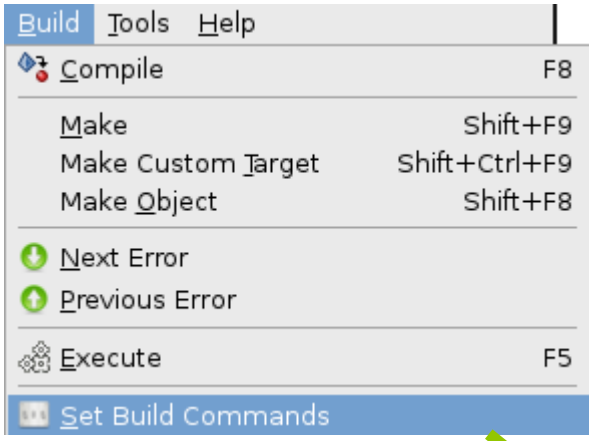
โปรแกรมช่วยให้เขียน **Python** ให้ง่ายขึ้น

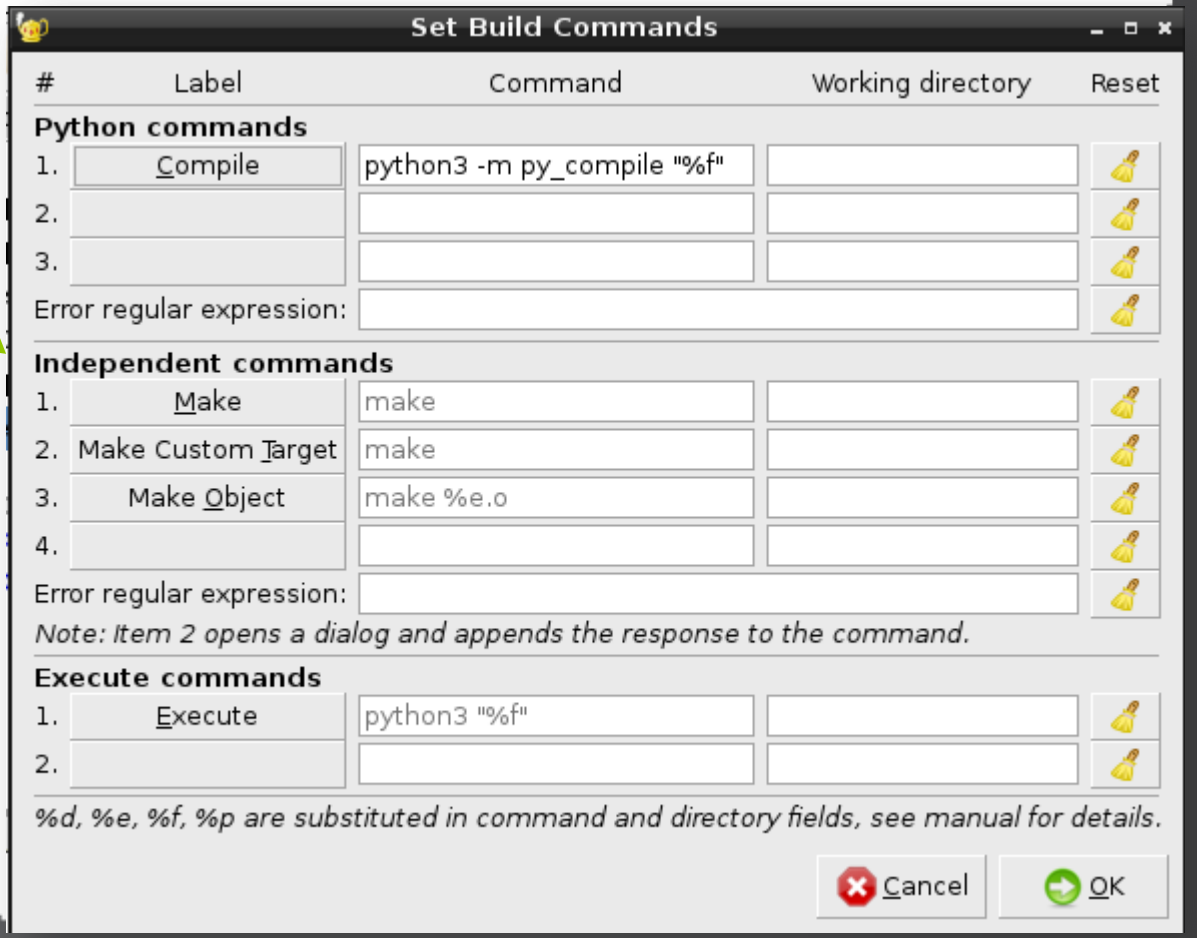
```
sudo apt-get install geany
```

ติดตั้งโปรแกรมผ่าน **Internet**



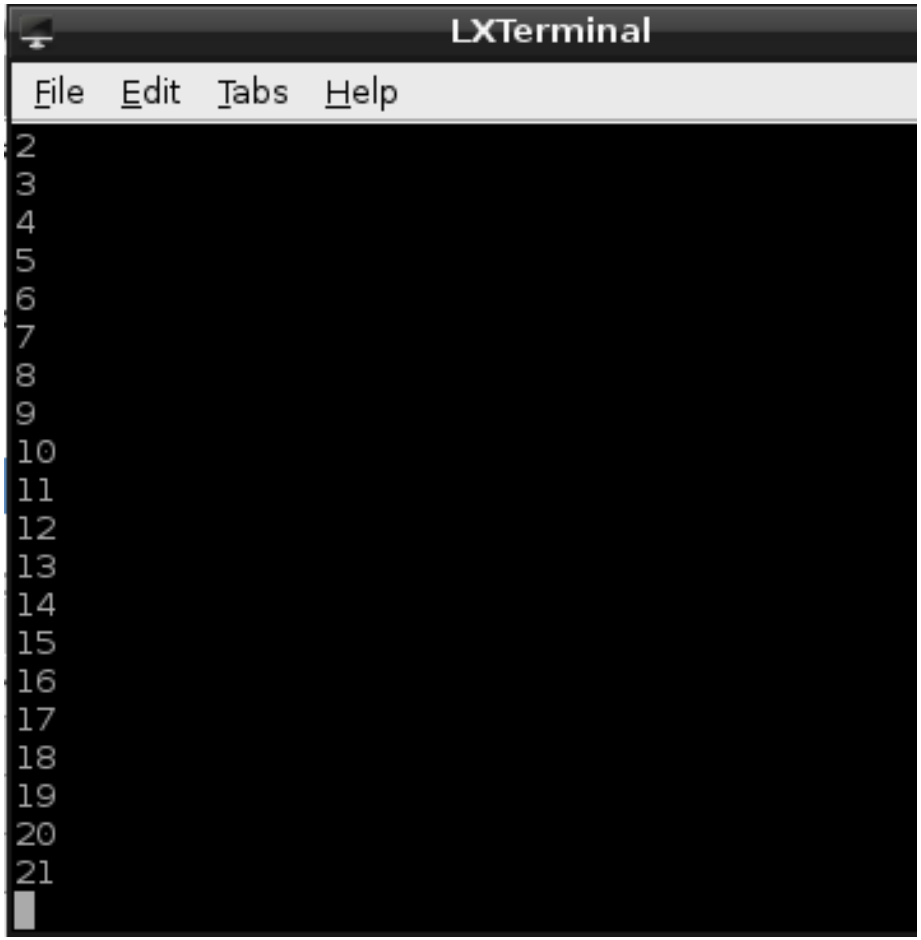
ตั้งให้คอมไพล์ด้วย python3





ทดสอบโปรแกรม python ผ่าน Geany

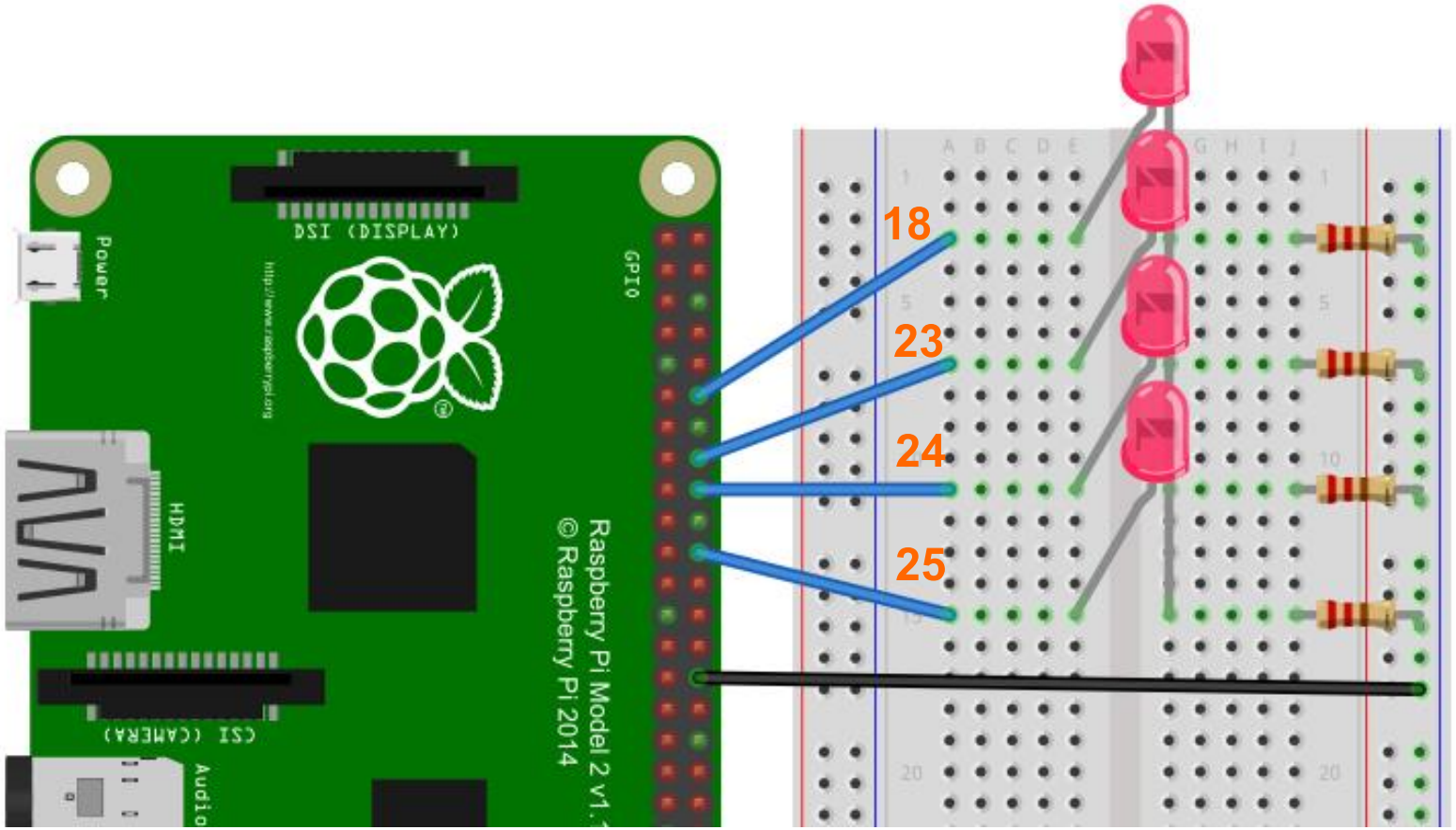
```
import RPi.GPIO as GPIO
import time
x=0
while (True) :
        print (x)
        time.sleep(0.5)
        x=x+1
```



กด F5 RUN

กด CTRL+C ออก

ต่อวงจร LED 4 ดวง

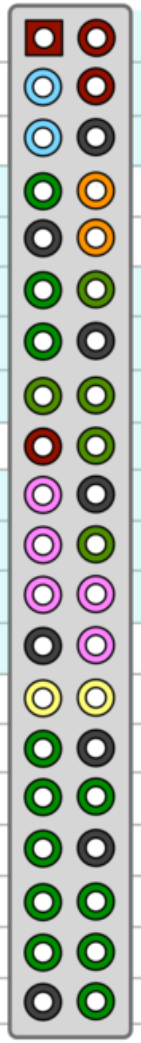


ไลบรารี่ RPi.GPIO

BCM

BOARD

Pin#	NAME	Connection		Connection	NAME	Pin#
01	3.3V			**5V** (Cupcade)	5V	02
03	GPIO 2			**5V** (Powerboost)	5V	04
05	GPIO 3			**GND** (Powerboost)	Ground	06
07	GPIO 4	**START**			GPIO 14	08
09	Ground	**GND** (Cupcade)			GPIO 16	10
11	GPIO 17	**UP**		**SELECT**	GPIO 18	12
13	GPIO 27	**DOWN**		**GND** (Select/Start)	Ground	14
15	GPIO 22	**LEFT**		**RIGHT**	GPIO 23	16
17	3.3V			**A**	GPIO 24	18
19	GPIO 10	**B**		**GND** (ABXYR)	Ground	20
21	GPIO 09	**X**		**Y**	GPIO 25	22
23	GPIO 11	**L Shoulder**		**R Shoulder**	GPIO 08	24
25	Ground	**GND** (L)			GPIO 07	26
27	ID_SD				ID_SC	28
29	GPIO 05				Ground	30
31	GPIO 06				GPIO 12	32
33	GPIO 13				Ground	34
35	GPIO 19				GPIO 16	36
37	GPIO 26				GPIO 20	38
39	Ground				GPIO 21	40



16mA MAX

ไลบรารี่ RPi.GPIO

เพิ่มไลบรารี่ในชื่อ GPIO

```
import RPi.GPIO as GPIO
```

setmode กำหนดรูปแบบขา

```
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setmode(GPIO.BOARD)
```

setup กำหนด input/output

```
GPIO.setup(pin,GPIO.OUT)
```

```
GPIO.setup(pin,GPIO.IN)
```

output ส่งค่าออกไปที่ขา

```
GPIO.output(pin,True)
```

```
GPIO.output(pin,False)
```

input รับค่าลอจิกจากขา

```
x = GPIO.input(pin)
```

cleanup เคลียร์ค่าพอร์ตกลับ

```
GPIO.cleanup()
```

PWM เปิดการทำงาน PWM

```
GPIO.PWM(pin,freq)
```

ไฟกะพริบ 4 ดวง

```
import RPi.GPIO as GPIO
import time
x=0
GPIO.setmode(GPIO.BCM)
GPIO.setup(18,GPIO.OUT)
GPIO.setup(23,GPIO.OUT)
GPIO.setup(24,GPIO.OUT)
GPIO.setup(25,GPIO.OUT)
try:
        while (True) :
                GPIO.output(18,1)
                GPIO.output(23,1)
                GPIO.output(24,1)
                GPIO.output(25,1)
                time.sleep(0.5)
                GPIO.output(18,0)
                GPIO.output(23,0)
                GPIO.output(24,0)
                GPIO.output(25,0)
                time.sleep(0.5)
finally:
        GPIO.cleanup()
```

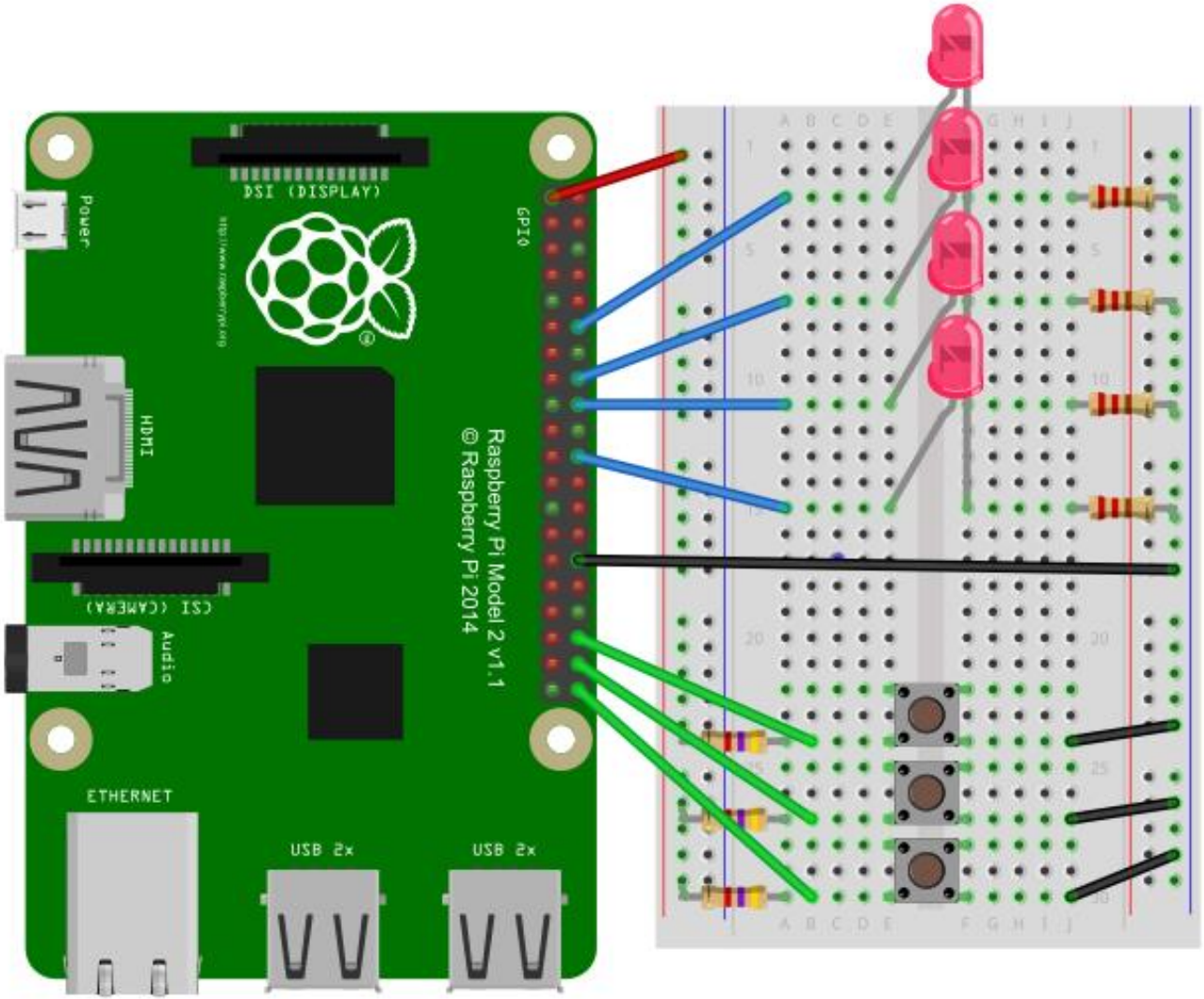
ไฟกะพริบ 4 ดวง

```
import RPi.GPIO as GPIO
import time
x=0
pins = [18,23,24,25]
GPIO.setmode(GPIO.BCM)
for gpios in pins:
   GPIO.setup(gpios,GPIO.OUT)
try:
   while (True) :
      for gpios in pins:
         GPIO.output(gpios,1)
         time.sleep(0.5)
         GPIO.output(gpios,0)
         time.sleep(0.5)
finally:
   GPIO.cleanup()
```

ไฟวิ่ง 4 ดวง

```
import RPi.GPIO as GPIO
import time
x=0
pins = [18,23,24,25]
GPIO.setmode(GPIO.BCM)
for gpios in pins:
   GPIO.setup(gpios,GPIO.OUT)
try:
   while (True) :
      for gpios in pins:
         GPIO.output(gpios,1)
         time.sleep(0.2)
      for gpios in pins:
         GPIO.output(gpios,0)
         time.sleep(0.2)
finally:
   GPIO.cleanup()
```

สวิตช์ควบคุม เปิด/ปิดไฟ



สั่ง Pull up Pull Down ด้วยซอฟต์แวร์

กำหนดให้ขา 20 เป็นขาอินพุต และมีการพูลอัพ

```
GPIO.setup(20,GPIO.IN,pull_up_down=GPIO.PUD_UP)
```

กำหนดให้ขา 21 เป็นขาอินพุต และมีการพูลดาวน์

```
GPIO.setup(21,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
```

50 กิโลโอห์ม พูลอัพ

อ่านค่าสวิตช์ขอบขาขึ้น ขอบขาลง

```
GPIO.wait_for_edge(23, GPIO.FALLING)
```

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN, pull_up_down =
GPIO.PUD_DOWN)
GPIO.setup(24, GPIO.IN, pull_up_down =
GPIO.PUD_UP)
while True:
   GPIO.wait_for_edge(23, GPIO.RISING)
   print("Button 1 Pressed")
   GPIO.wait_for_edge(23, GPIO.FALLING)
   print("Button 1 Released")
   GPIO.wait_for_edge(24, GPIO.FALLING)
   print("Button 2 Pressed")
   GPIO.wait_for_edge(24, GPIO.RISING)
   print("Button 2 Released")
   GPIO.cleanup()
```

EVENTS AND CALLBACK FUNCTIONS

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)
GPIO.setup(24, GPIO.IN, pull_up_down = GPIO.PUD_UP)
def PntFnc(channel):
  print("Button 1 pressed!")
  print("Note how the bouncetime affects the button press")
GPIO.add_event_detect(23, GPIO.RISING, callback=PntFnc,
bouncetime=300)
while True:
  GPIO.wait_for_edge(24, GPIO.FALLING)
  print("Button 2 Pressed")
  GPIO.wait_for_edge(24, GPIO.RISING)
  print("Button 2 Released")
GPIO.cleanup()
```

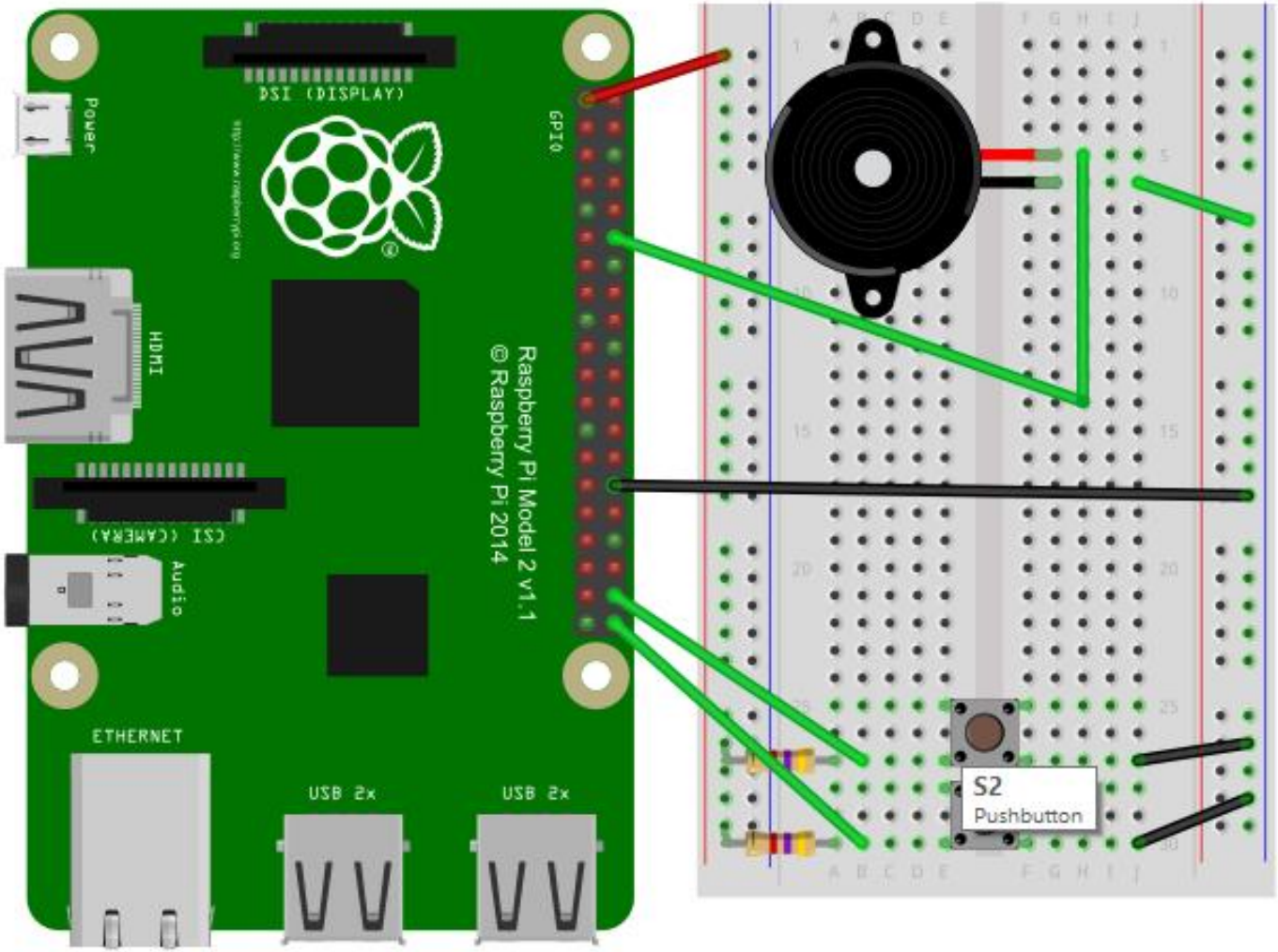
จบงานด้วย

```
GPIO.remove_event_detect(23)
```

สวิตช์ควบคุม เปิด/ปิดไฟ

```
import time
import RPi.GPIO as GPIO
led= [18,23,24,25]
GPIO.setmode(GPIO.BCM)
for i in led:
    GPIO.setup(i,GPIO.OUT)
GPIO.setup(8,GPIO.IN)
GPIO.setup(20,GPIO.IN)
GPIO.setup(21,GPIO.IN)
GPIO.output(18,1)
try:
    while True:
        if not (GPIO.input(8)):
            GPIO.output(18,1)
        else:
            GPIO.output(18,0)
        if not (GPIO.input(20)):
            GPIO.output(23,1)
        else:
            GPIO.output(23,0)
finally:
    GPIO.cleanup()
```

สร้างเสียงออกลำโพง



สร้างเสียงออกลำโพง

```
def sound(pins,freq,times):
   for i in range(0,times):
      GPIO.output(pins,1)
      time.sleep(0.5/freq)
      GPIO.output(pins,0)
      time.sleep(0.5/freq)
```

PWM : พัลส์วิธมอดูเลเตอร์

เปิดการใช้ PWM

```
p=GPIO.PWM(pin,freq)
```

เริ่มต้น PWM

```
p.start(%dutycycle)
```

เปลี่ยนความถี่ PWM

```
p.ChangeFrequency(freq)
```

เปลี่ยนค่าความกว้างพัลส์

```
p.ChangeDutyCycle(%duty)
```

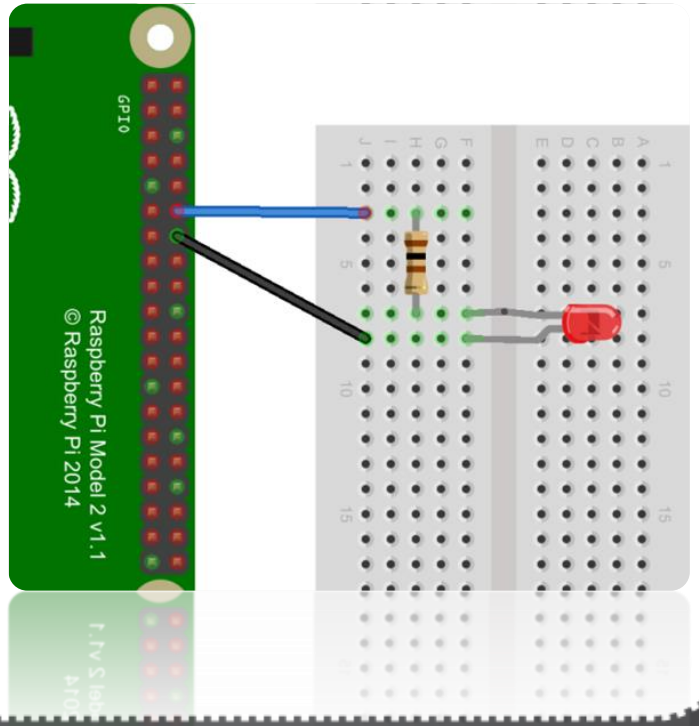
หยุด PWM

```
p.stop()
```

ไฟกะพริบด้วย PWM

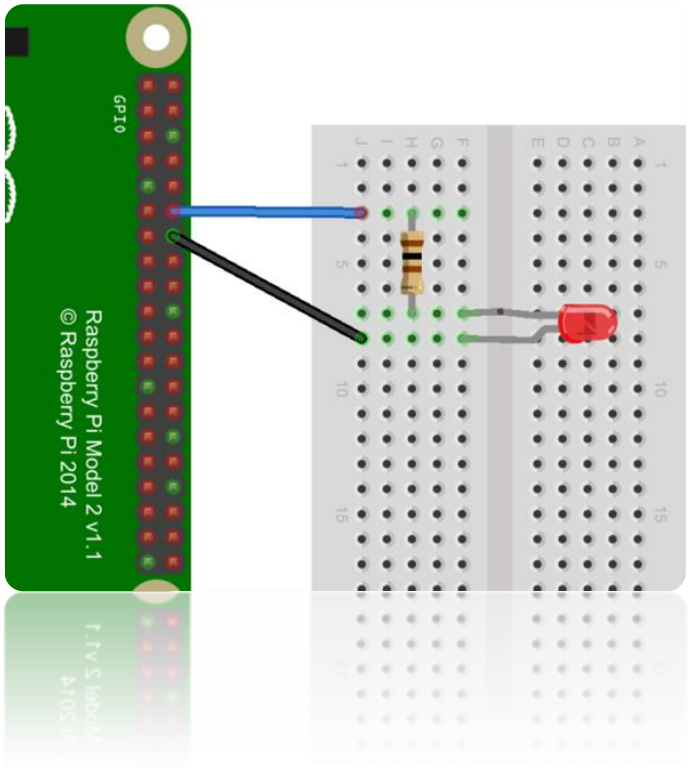
```
# Test PWM
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18,GPIO.OUT)
GPIO.setup(20,GPIO.IN)
p=GPIO.PWM(18,1)
p.start(20)
input('Press return to stop:')
p.stop()
GPIO.cleanup()
```



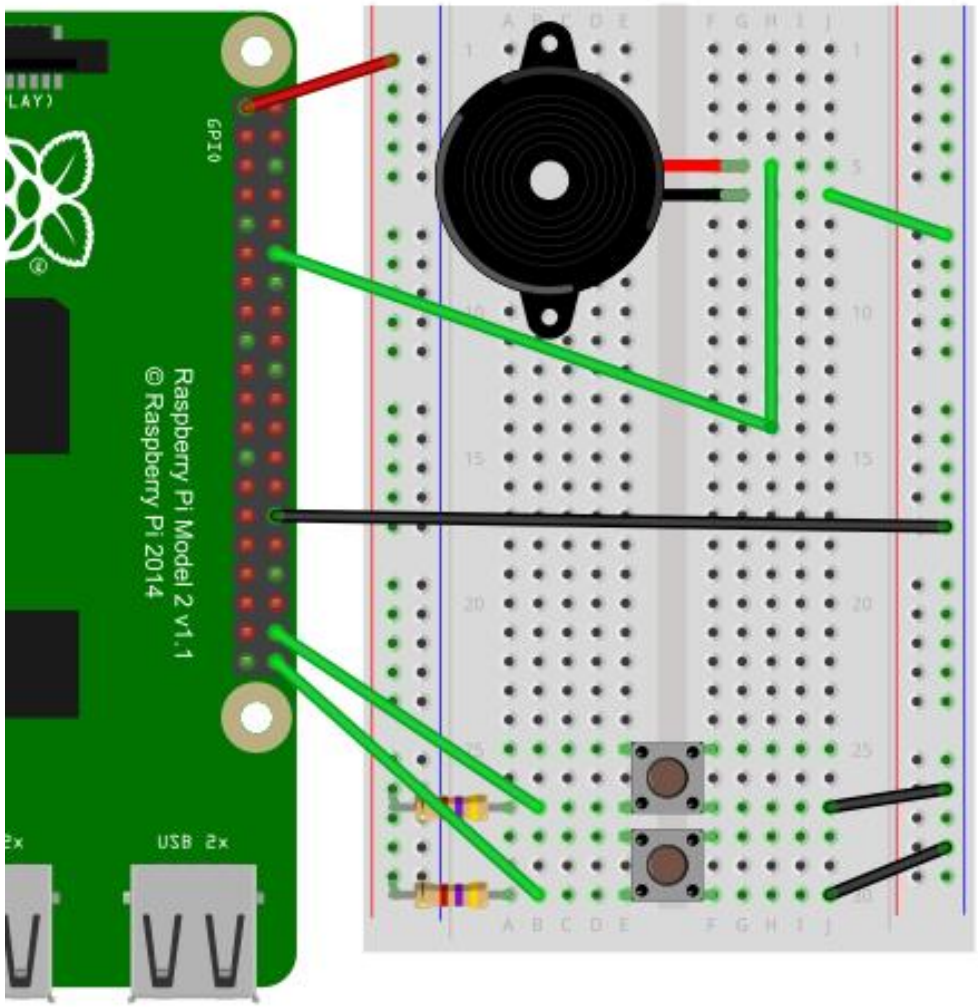
ไฟกะพริบด้วย PWM

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
p = GPIO.PWM(18, 50)
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()
```



ปรับความถี่เสียง ด้วย ChangeFrequency

```
# Test PWM
import RPi.GPIO as GPIO
import time
x=2000
GPIO.setmode(GPIO.BCM)
GPIO.setup(18,GPIO.OUT)
GPIO.setup(20,GPIO.IN)
GPIO.setup(21,GPIO.IN)
p=GPIO.PWM(18,2000)
p.start(20)
try:
    while True:
        if not GPIO.input(20):
            x=x+100
            print (x)
            p.ChangeFrequency(x)
            time.sleep(0.2)
        if not GPIO.input(21):
            if x>100:
                x=x-100
            print (x)
            p.ChangeFrequency(x)
            time.sleep(0.2)
finally:
    p.stop()
    GPIO.cleanup()
```



WiringPi เขียนภาษา C บน RPi

ติดตั้ง wiringPi

```
git clone git://git.drogon.net/wiringPi
cd wiringPi
git pull origin
./build
```

ทดสอบผ่าน Terminal (สั่ง LED ติดและดับที่ ขา 18)

```
gpio -g mode 18 output
gpio -g write 18 1
gpio -g write 18 0
```

ทดสอบผ่าน Terminal (ให้ขา 22 เป็นอินพุตพูลอัพ และอ่านค่า)

```
gpio -g mode 22 up
gpio -g read 22
```

WiringPi เขียนภาษา C บน RPi

กำหนดค่าเริ่มต้น

```
wiringPiSetupGpio();  // กำหนดค่าเริ่มต้นให้ RPi แบบบขา BCM
```

กำหนดค่าเริ่มต้น

```
pinMode(22,INPUT)
pinMode(23,OUTPUT)
pinMode(18,PWM_OUTPUT)
```

สั่งงานเอาต์พุต

```
digitalWrite(23, HIGH);
pwmWrite(18, 723);
```

อ่านค่าอินพุต

```
if (digitalRead(22))
   printf("Pin 22 is HIGH\n");
else
   printf("Pin 22 is LOW\n");
```

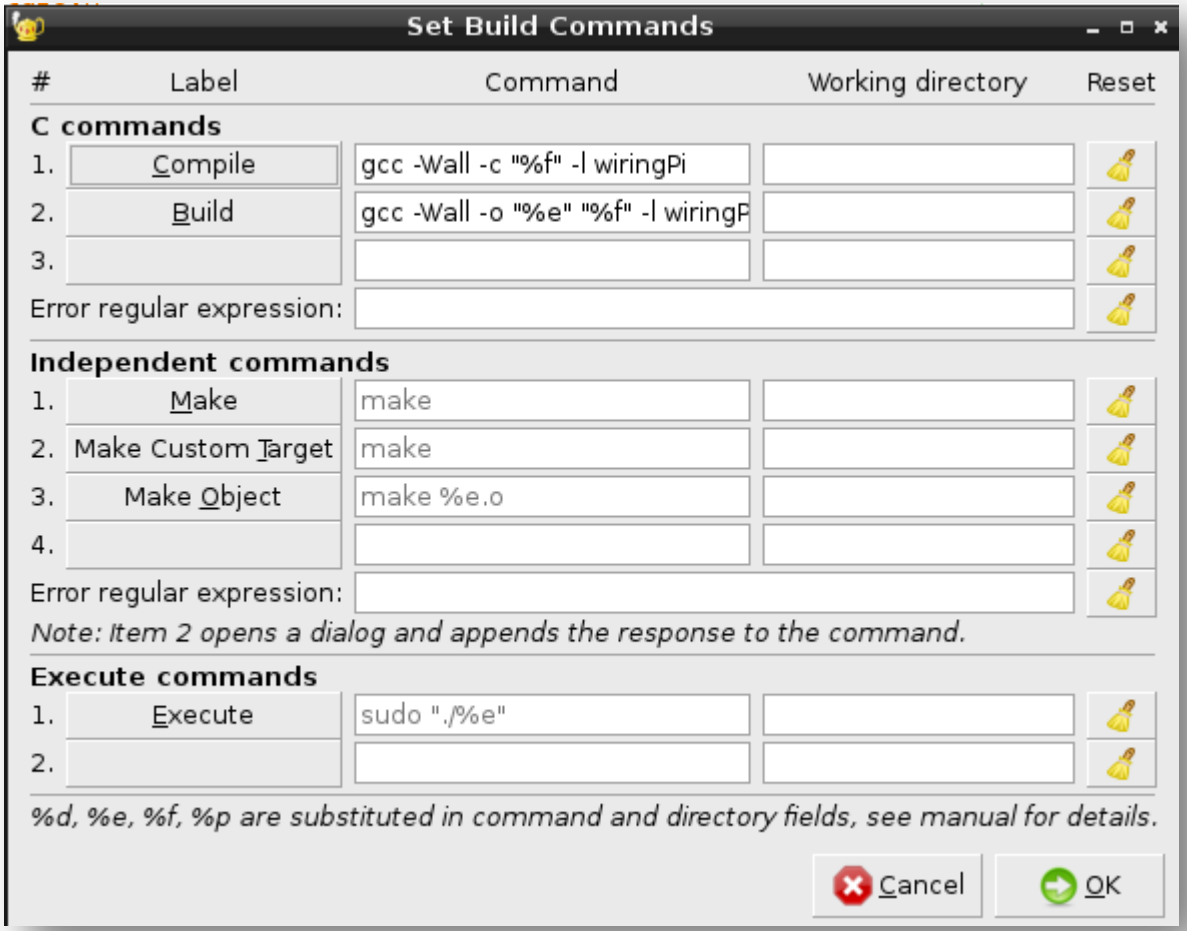
กำหนดขาพูลอัพ

```
pullUpDnControl(22, PUD_UP);
```

หน่วงเวลา

```
delay(2000);
```

ปรับค่าการคอมไพล์สำหรับ WiringPi



การใช้ wiringpi2 กับ Python และ Rpi

by Gordon ‘Drogon’ Henderson

wiringPi พัฒนา Rpi ให้ง่ายเหมือนเขียน Arduino

ติดตั้งผ่าน github

```
git clone https://github.com/Gadgetoid/WiringPi2-Python.git
cd WiringPi2-Python
sudo python3 setup.py
```

ไลบรารี่ wiringPi2

เพิ่มไลบรารี่

```
import wiringpi2 as gpio
```

setmode กำหนดรูปแบบขา

```
gpio.wiringPiSetupGpio()
```

```
gpio.wiringPiSetup()
```

กำหนดขาเป็น output

```
gpio.pinMode(pin,1)
```

กำหนดขาเป็น input

```
gpio.pinMode(pin,0)
```

output ส่งค่าออกไปที่ขา

```
gpio.digitalWrite(pin,1)
```

```
gpio.digitalWrite(pin,0)
```

input รับค่าลอจิกจากขา

```
gpio.digitalRead(pin)
```

pull up

```
gpio.pullUpDnControl(pin,2)
```

pull down

```
gpio.pullUpDnControl(pin,1)
```

ไม่ pull up/down

```
gpio.pullUpDnControl(pin,0)
```

ใช้คำสั่ง pwm ด้วย wiringPi2

กำหนดขา 18 เป็น PWM

```
wiringpi.pinMode(18,2)
```

กำหนดค่า dutycycle

```
wiringpi.pwmWrite(18,duty)
```

```
import wiringpi2 as pi
import time
leds = [18,23,24,25]
sw = [16,20,21]
pi.wiringPiSetupGpio()
pi.pinMode(16,0)
pi.pinMode(18,2)
while True:
        for i in range(1025):
                pi.pwmWrite(18,i)
                time.sleep(0.001)
        for i in range(1025,-1,-1):
                pi.pwmWrite(18,i)
                time.sleep(0.001)
```