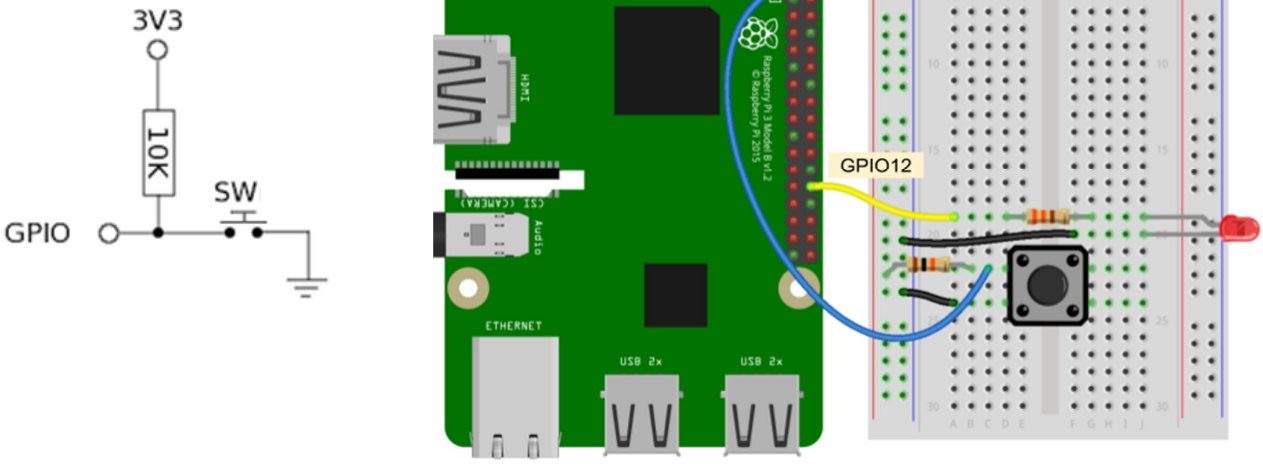
GPIO Pin

Alternate Function						Alternate Function
	3.3V PWR	1	2	5V PWR		
I2C1 SDA	GPIO 2	3	4	5V PWR		
I2C1 SCL	GPIO 3	5	6	GND		
	GPIO 4	7	8	UART0 TX		
	GND	9	10	UART0 RX		
	GPIO 17	11	12	GPIO 18		PWM0
	GPIO 27	13	14	GND		
	GPIO 22	15	16	GPIO 23		
	3.3V PWR	17	18	GPIO 24		
SPI0 MOSI	GPIO 10	19	20	GND		
SPI0 MISO	GPIO 9	21	22	GPIO 25		
SPI0 SCLK	GPIO 11	23	24	GPIO 8		SPI0 CS0
	GND	25	26	GPIO 7		SPI0 CS1
	Reserved	27	28	Reserved		
	GPIO 5	29	30	GND		
	GPIO 6	31	32	GPIO 12		PWM0
PWM1	GPIO 13	33	34	GND		
SPI1 MISO	GPIO 19	35	36	GPIO 16		SPI1 CS0
	GPIO 26	37	38	GPIO 20		SPI1 MOSI
	GND	39	40	GPIO 21		SPI1 SCLK

การทดลองที่ 1 GPIO Input

- ต่อขา GPIO4 เข้ากับ Switch และ ตัวต้านทาน 10 กิโลโอห์ม ดังรูป



การทดลองที่ 1. GPIO Input

การกำหนดค่าให้ GPIO4 เป็น Input ใช้คำสั่ง

```
GPIO.setup(4,GPIO.IN)        # กำหนด GPIO17 เป็นขา Input
```

และการอ่านค่าจาก GPIO4 ใช้คำสั่ง

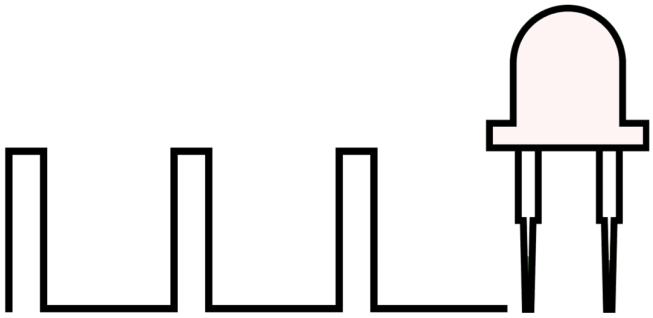
```
val = GPIO.input(4)          # ค่าที่อ่านได้ มาเก็บในตัวแปร val โดย
                               จะได้ค่าเป็น 0 หรือ 1
```

แบบฝึกหัด

เมื่อกดปุ่ม ให้ LED ติด และเมื่อปล่อยให้ LED ดับ

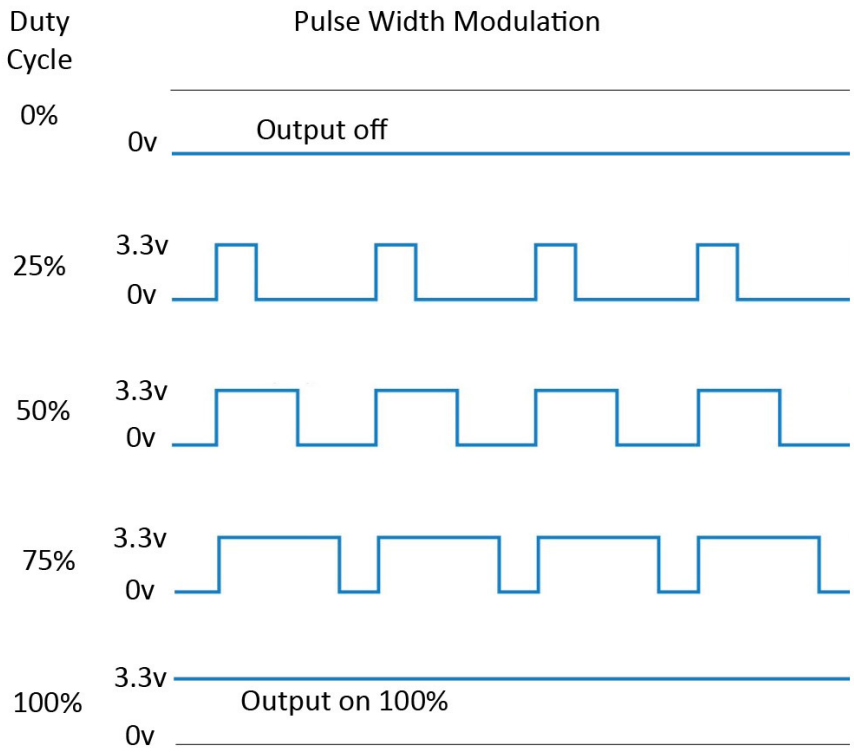
2. PWM : Pulse Width Modulation

PWM คือ ค่า Duty Cycle มีค่าแปรผันตามแรงดัน input ที่ป้อนเข้ามา
โดยที่ค่า Duty cycle คือ ค่าความกว้าง Pulse ต่อ ค่าคาบเวลาของสัญญาณ

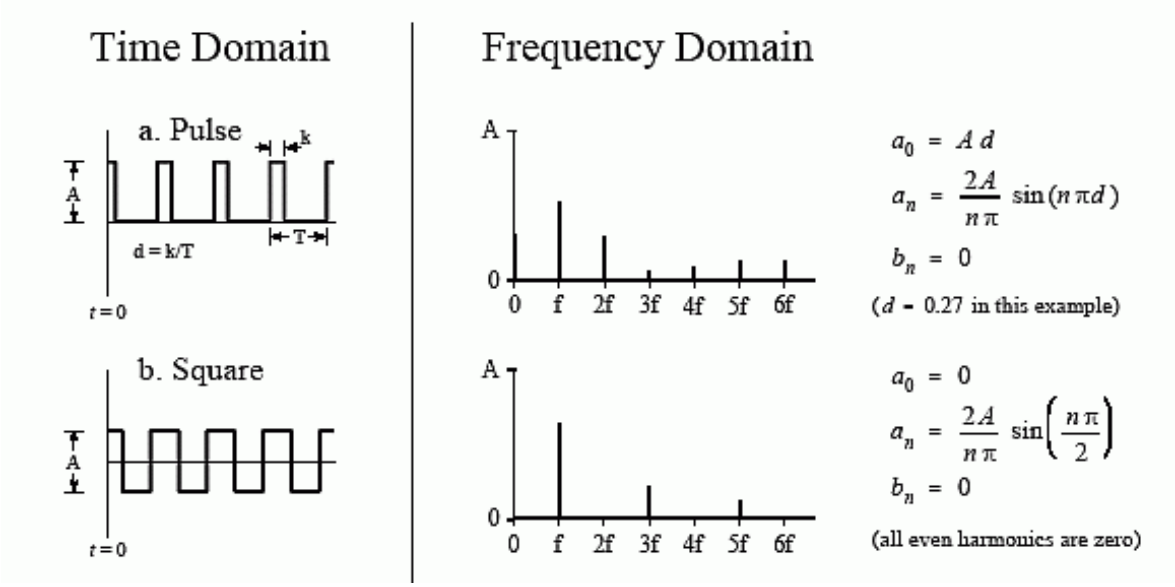


ถ้าค่า Duty cycle มีค่าน้อย ก็จะให้แรงดัน DC ออกมาน้อย
ถ้าค่า Duty cycle มีค่ามาก ก็จะให้แรงดัน DC ออกมามีค่าเยอะ
นิยม นำสัญญาณ PWM ไปใช้ควบคุม ความเร็ว motor เป็นต้น

Pulse Width Modulation

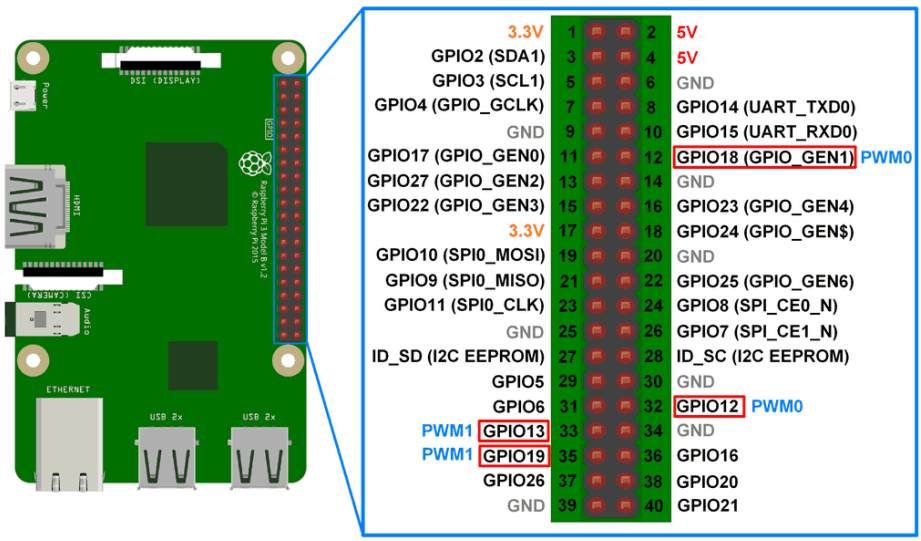


Pulse Width Modulation



Pulse Width Modulation

ใน Raspberry Pi สามารถสร้างสัญญาณ PWM ได้ที่ขา
GPIO12 GPIO18 GPIO13 และ GPIO19



Using PWM in 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
```

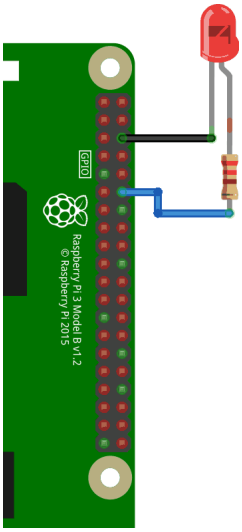
To change the duty cycle:

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

To stop PWM:

```
p.stop()
```

การทดลองที่ 2. Pulse Width Modulation



สร้างไฟล์โดยใช้คำสั่ง **$nano pwm.py**

เขียนโปรแกรม

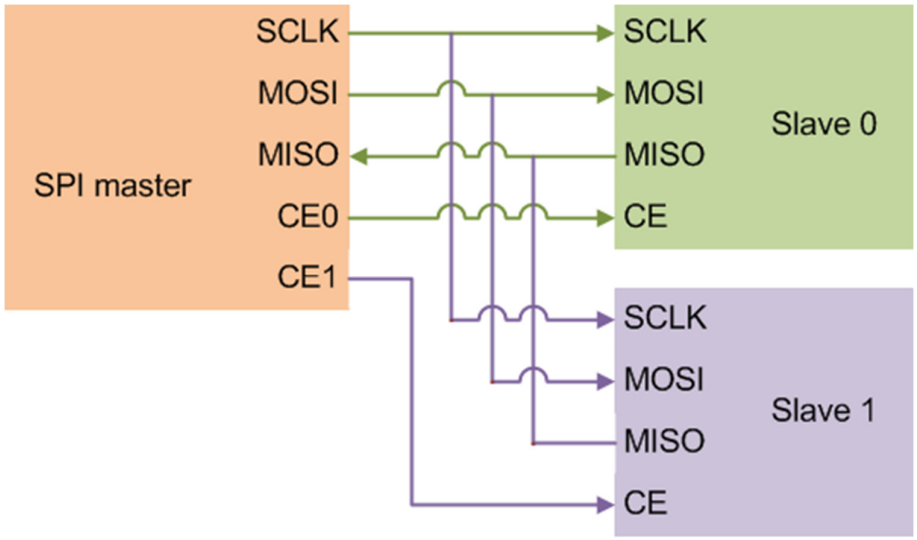
```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setwarnings(False)          #disable warnings
GPIO.setmode(GPIO.BOARD)         #set pin numbering
ledpin = 12                      #PWM pin connected to LED

system GPIO.setup(ledpin,GPIO.OUT)
pi_pwm = GPIO.PWM(ledpin,1000)   #create PWM instance with frequency 1000
pi_pwm.start(0)                  #start PWM of 0% Duty Cycle

while True:
        for duty in range(0,101,1):
        pi_pwm.ChangeDutyCycle(duty)  #provide duty cycle in the range 0-100
        sleep(0.05)
```

Run โปรแกรมโดยใช้คำสั่ง

$python pwm.py

3. SPI : Serial Peripheral Interface

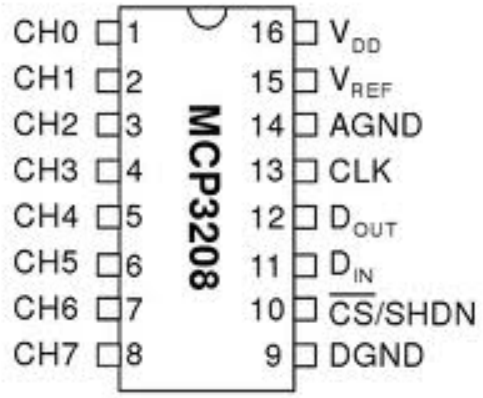
เป็นวิธีการสื่อสารอนุกรมแบบ Synchronous อีกรูปแบบหนึ่ง ซึ่งทำงานในรูปแบบที่ให้อุปกรณ์ตัวหนึ่งทำหน้าที่เป็น Master ในขณะที่อีกตัวหนึ่งทำหน้าที่เป็น Slave และสามารถส่งข้อมูลในโหมด Full-duplex นั่นหมายความว่า สัญญาณสามารถส่งหากันได้ระหว่าง Master และ Slave ได้อย่างต่อเนื่อง



3. SPI : Serial Peripheral Interface

- การสื่อสารระหว่าง Master กับ Slave จะใช้ Port ของอุปกรณ์ประกอบด้วย
  - **MISO** (Master In Slave Out) คือ ออกจาก Slave มาเข้า Master
  - **MOSI** (Master Out Slave In) คือ ออกจาก Master มาเข้า Slave
  - **SCK** (Serial Clock) ขานี้เป็นขาที่ Master ใช้กำหนดจังหวะในการส่งข้อมูล
  - **Chip Select** ขานี้ไว้ใช้เลือกอุปกรณ์ที่เป็น Slave ซึ่งอาจจะมีได้มากกว่า 1 ตัว
  - 3.3 VDC
  - GND

Analog to Digital : MCP3208



- 12-bit resolution
- เป็น Analog Input ทั้ง 8 Channel สามารถ Sampling ได้ประมาณ 100 kSamp/s

(ค่ารวมทั้ง 8 ช่อง ถ้าใช้มากกว่า 1 ช่อง ความเร็วก็ต้องหารด้วยจำนวนช่องที่ใช้งานด้วย)

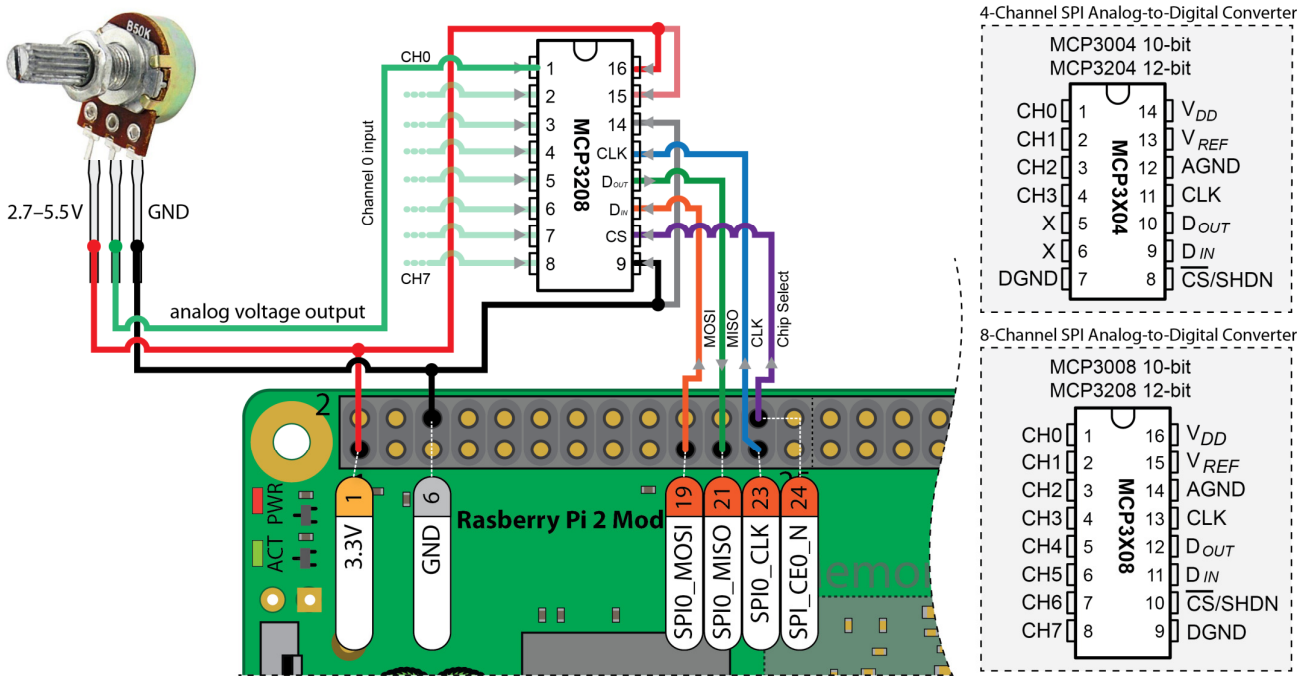
- ใช้การติดต่อแบบ Serial Peripheral Interface (SPI) ซึ่งประกอบด้วย CLK, DOUT, DIN, CS, DGND

Analog to Digital : MCP3208

- เราจะใช้ Raspberry Pi เป็นตัวสั่งให้ MCP3208 ทำงาน โดยการเลือก Channel ของสัญญาณที่ต้องการ

- ดังนั้น Raspberry PI ก็เป็น Master และ MCP3208 ก็จะเป็น Slave

Analog to Digital : MCP3208



การทดลองที่ 3. Analog to Digital : MCP3208

พิมพ์คำสั่ง **raspi-config -> Interface option** to enable SPI

ติดตั้ง

```
$sudo apt-get install python3-pip
$pip3 install mcp3208
```

สร้างไฟล์โดยใช้คำสั่ง **$nano test.py**

เขียนโปรแกรม

```
from mcp3208 import MCP3208
import time
adc = MCP3208()
while True:
        for i in range(8):
                print('ADC[{}]: {:.2f}'.format(i, adc.read(i)))

        time.sleep(0.5)
```

Run โปรแกรมโดยใช้คำสั่ง

```
$python3 test.py
```

แบบฝึกหัด

- อ่านค่าจากตัวต้านทานปรับค่าได้ แล้วไปปรับความสว่างของหลอด LED
- อ่านค่าจาก Sensor อุณหภูมิ แล้วแสดงผลค่าอุณหภูมิ