PYTHON

POR. Christian Quispe Canchari

PYTHON I

CARACTERISTICAS

Tipado dinámico
Multiplataforma
Interpretado
Orientado a objetos

Salidas y Entradas

Print : Permite vizualisar texto o valores de variables
Input : Permite ingresar un dato al programa

Ejemplo:

```
valor = input ("ingrese un numero")
print (valor)
```

VARIABLES

Nombre_de_variable = valor_asignado

Ejemplo

```
dato1 = 15
dato2 = 5.32
valor_prueba = 4
a,b,c = "Hola",2,0.2
```

Tipos de datos

Int : una variable de tipo integer o entero solo puede guardar números enteros

Float: Solo puede almacenar números decimales, tomar en cuenta que en Python los números decimales se escriben con punto en lugar de coma.

Bool: una variable booleana solo puede guardar uno de los siguientes valores:

True o False (Verdadero o False)

```python
#Entero
valor = 5
#octal
valor = 048
#hexadecimal
valor = 0xFF
#real
valor = 2.5
#String
valor = "Hola Mundo"
#booleano ( V o F)
valor = True #Respetando la mayuscula
valor = False
```

SENTENCIA IF

```python
nota = int ( "ingrese un numero")

if nota >= 10 :
    print("Aprobado")
else:
    print("Desaprobado")
```

```python
nota = int (input( "ingrese un numero"))

if nota >= 20 :
    print("Fuera de Rango")
elif nota >= 10:
    print("Aprobado")
else:
    print("Desaprobado")
```

while

```
i = 0
while i <= 5:
    print i
    i+=1
```

Funciones

```
def mensaje():
    print("Hola mundo")


mensaje()
```

```
def suma(a,b):
    print(a+b)

val  = sumar(2,3)

print("La suma es" + val)
```

MÓDULOS

main.py

```
from FOO import *

sumar(1,2)
```

FOO.py

```
def suma(a,b):
    print a+b
```

Choque de nombres

```
from FOO import *
from BAZ import *

suma(1,2)
baz()
```

BAZ.py

```
def baz():
    print("Funcion Baz")
def sumar():
    print("Funcion suma")
```

Propuesta 1

```
from FOO import suma
from BAZ import restar

suma(1,2)
restar(2,0)
```

```
def suma(a,b):
    print a+b
```

Propuesta 2

```
import FOO
FOO.suma(2,1)
```

```
def restar(m,n):
    print (m-n)
def suma(x,y):
    print(2+x+y)
```

Renombrando módulos y componentes con “as”

También es posible darle otro nombre a un módulo o a un componente de un módulo para evitar colisiones de nombres utilizando la palabra reservada “a”

Funcion_sumar.py

```
def sumar(x,y):
    print(3+x+y)
```

```
import Funcion_sumar as su

su.sumar(2,1)
```

LISTAS Y TUPLAS

Son los que en otros lenguajes llaman vectores o arrays. Sin embargo presentan varias diferencias.

Listas: Dinamicas
A=[1,”Hola mundo”,False]

Tuplas : estáticas
B=(1,”Hola mundo”,False)

Modulos RPI.GPIO

Para importar el módulo RPi.GPIO:

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

Numeración de pines
Hay dos formas de numerar los pines IO en una Raspberry Pi dentro de RPi.GPIO. El primero es usar el sistema de numeración BOARD. Esto se refiere a los números de pin en el encabezado P1 de la placa Raspberry Pi. La ventaja de usar este sistema de numeración es que su hardware siempre funcionará, independientemente de la revisión de la placa del RPi. No necesitará volver a cablear su conector o cambiar su código.

El segundo sistema de numeración son los números BCM. Esta es una forma de trabajo de nivel inferior: se refiere a los números de canal en el Broadcom SOC. Siempre debe trabajar con un diagrama de qué número de canal va a qué pin en la placa RPi. Su script podría romper entre las revisiones de los tableros Raspberry Pi.

Para especificar cuál está utilizando usando (obligatorio):

```
GPIO.setmode(GPIO.BOARD)
# or
GPIO.setmode(GPIO.BCM)
```

Advertencias

Es posible que tenga más de un script / circuito en el GPIO de su Raspberry Pi. Como resultado de esto, si RPi.GPIO detecta que un pin se ha configurado con un valor diferente al predeterminado (entrada), aparece una advertencia cuando intenta configurar un script. Para deshabilitar estas advertencias:

```
GPIO.setwarnings(False)
```

Configurar un canal

Debe configurar cada canal que esté utilizando como entrada o salida. Para configurar un canal como entrada:

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

(donde canal es el número de canal según el sistema de numeración que ha especificado (BOARD o BCM)).

Para configurar un canal como salida:

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

(donde canal es el número de canal según el sistema de numeración que ha especificado (BOARD o BCM)).

También puede especificar un valor inicial para su canal de salida:

```
GPIO.setup(channel, GPIO.OUT, initial=GPIO.HIGH)
```

Configurar más de un canal
Puede configurar más de un canal por llamada (versión 0.5.8 en adelante). Por ejemplo:

```
chan_list = [11,12]
GPIO.setup(chan_list, GPIO.OUT)
```

Entrada
Para leer el valor de un pin GPIO:

```
GPIO.input(channel)
```

(donde canal es el número de canal según el sistema de numeración que ha especificado (BOARD o BCM)). Esto devolverá 0 / GPIO.LOW / False o 1 / GPIO.HIGH / True.

Salida
Para establecer el estado de salida de un pin GPIO:

```
GPIO.output(channel, state)
```

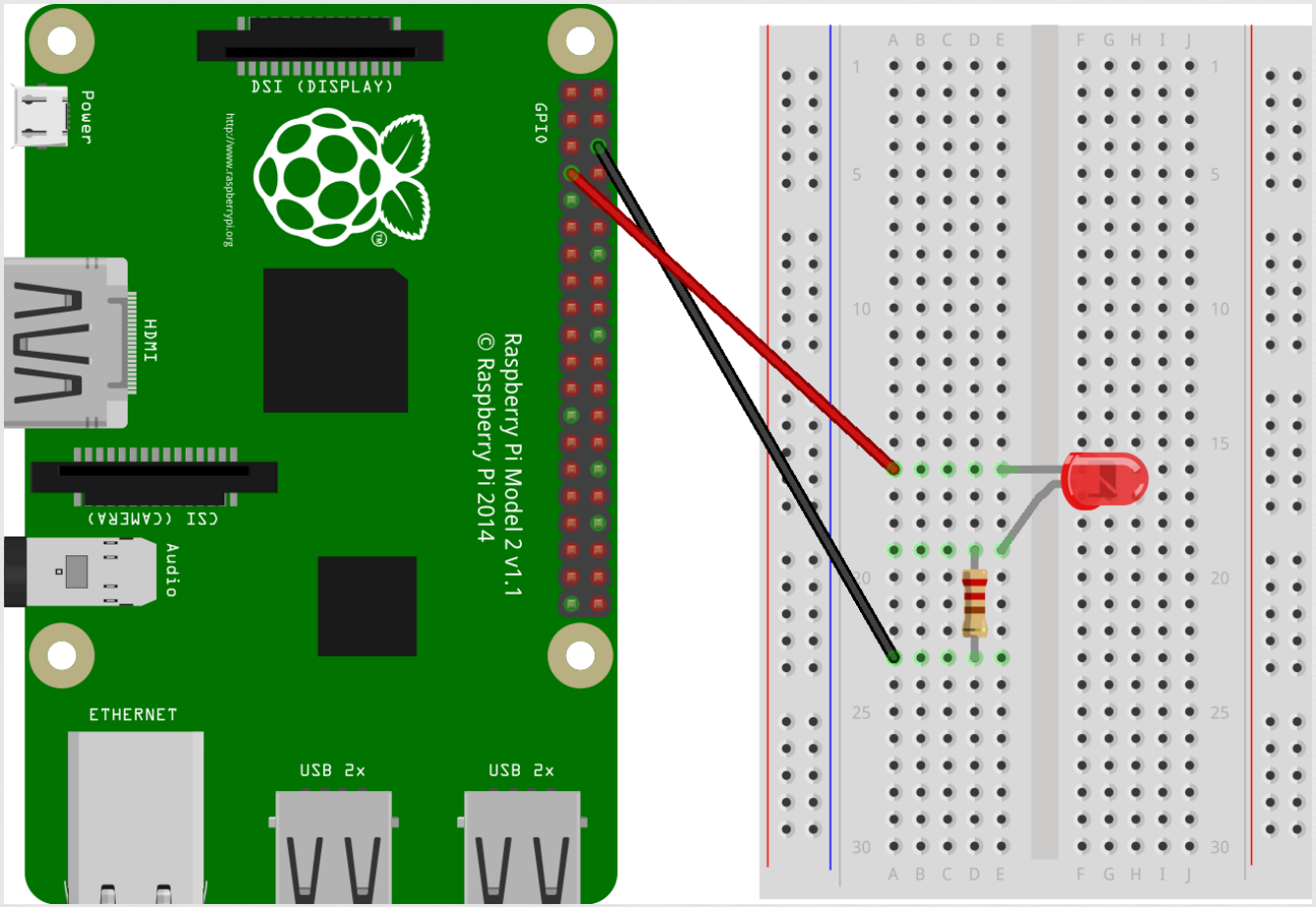
(donde canal es el número de canal según el sistema de numeración que ha especificado (BOARD o BCM)).

El estado puede ser 0 / GPIO.LOW / False o 1 / GPIO.HIGH / True.

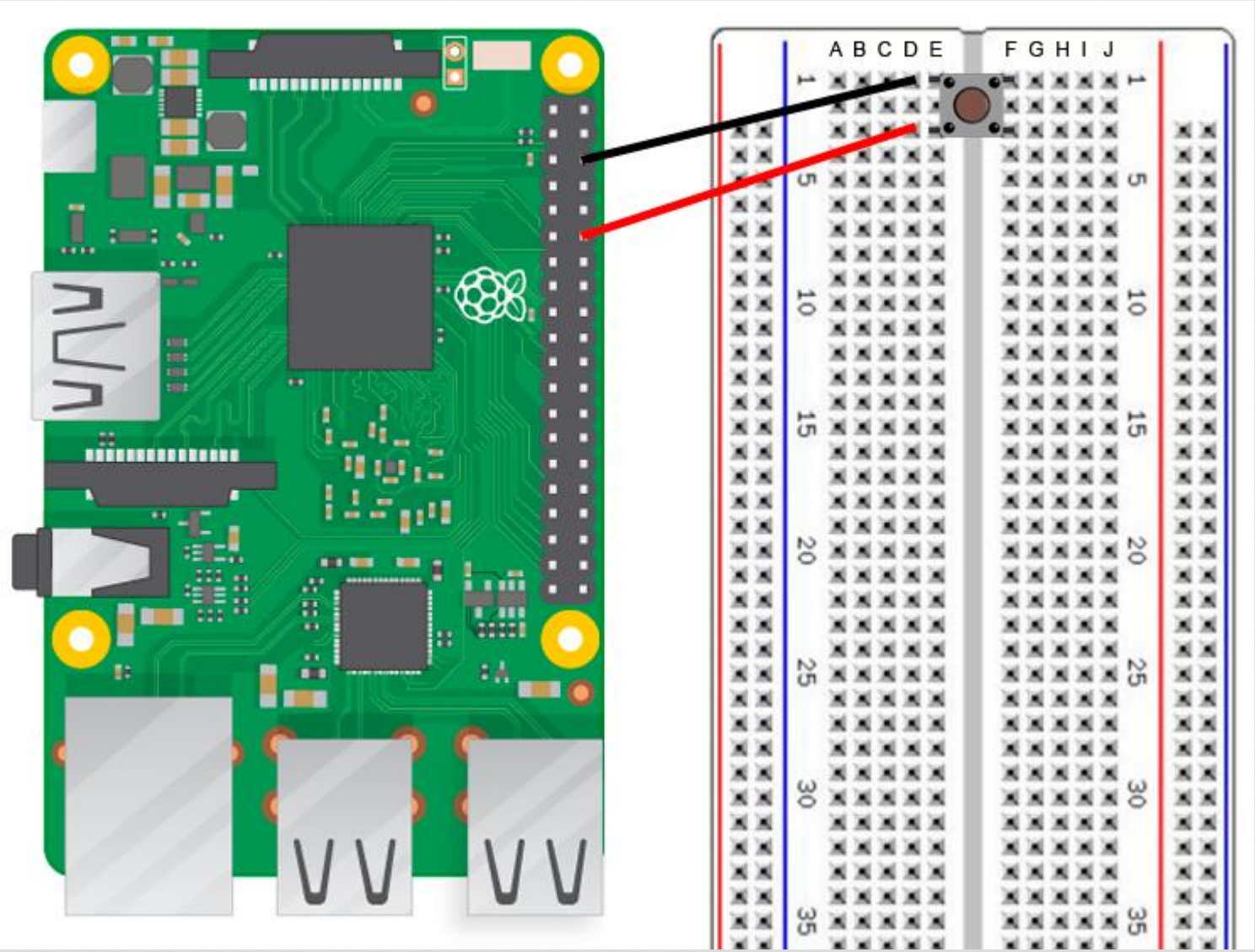
Salida a varios canales.
Puede enviar a muchos canales en la misma llamada (versión 0.5.8 en adelante).
Por ejemplo:

```
chan_list = [11,12]
GPIO.output(chan_list, GPIO.LOW)
GPIO.output(chan_list, (GPIO.HIGH, GPIO.LOW))
```

EJEMPLO1 : Parpadeo de un Led



EJEMPLO2 : Pulsador



EJEMPLO3 : Pulsador y Led