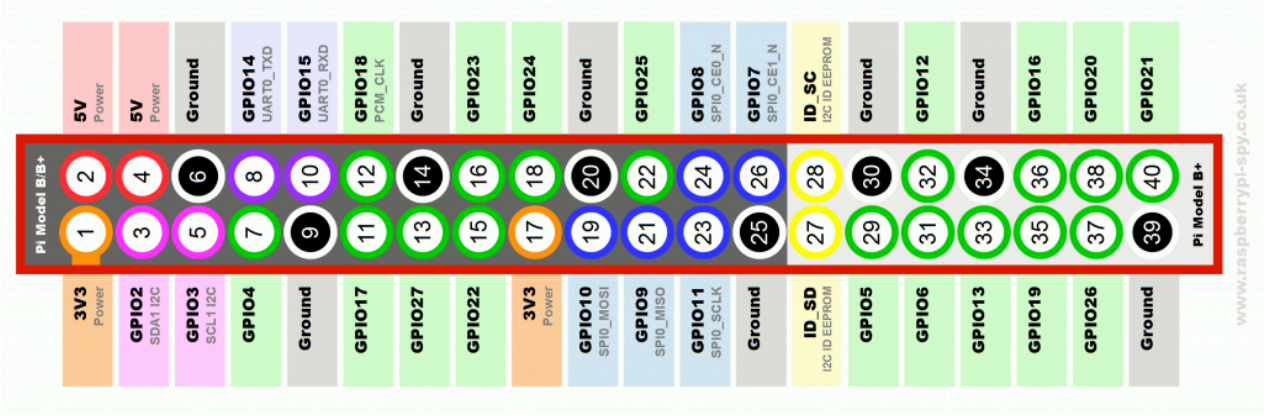
RaspBasicCode



- Alimentazione richiesta: 5V e 2.5Amps.
- Installazione libreria di accesso GPIO:

  sudo apt-get install wiringpi
- Installazione package RPi.GPIO per il controllo dei GPIO:

  pip install RPi.GPIO

wiringpi

wiringPi include un'utilità da riga di comando **gpio** che può essere utilizzata per programmare e configurare i pin GPIO.

```
sudo apt-get install wiringpi

gpio -v
     Raspberry Pi Details:
     Type: Model B+, Revision: 02, Memory: 512MB, Maker: Sony
     * Device tree is enabled.
     * This Raspberry Pi supports user-level GPIO access.
```

Per usare wiringpi su Bullseye si veda *wiringpi on Bullseye*.

```
Esempi di uso:
     gpio unexportall
     gpio export 25 out
     gpio export 1 out
     gpio write 25 0
     gpio write 1 0

gpio readall
   +-----+-----+---------+------+---+---Pi B+--+---+------+---------+-----+-----+
   | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
   +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
   |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
   |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5v      |     |     |
   |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
   |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
   |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
   |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
   |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
   |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
   |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
   |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
   |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | OUT  | GPIO. 6 | 6   | 25  |
   |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
   |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
```

```
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31 | 1  |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |    |    |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26 | 12 |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |    |    |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27 | 16 |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28 | 20 |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29 | 21 |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi B+---+---+------+---------+-----+-----+
```

BCM sta per: *Broadcom SOC channel*.

Comandi utili

Richiamiamo alcuni comandi di uso frequente

Azione	Comando
Elenco dei package installati	`apt list --installed`
Aggiorna l'elenco dei package disponibili e le loro versioni	`sudo apt-get update -y`
Installa le versioni più recenti dei packages	`sudo apt-get upgrade -y`
Cerca reti wireless	`sudo iwlist wlan0`
Visualizza processi	`ps -fA` o `ps -fA | grep <name>`
Termina un processo	`sudo kill -s KILL <process number>`
Visualizza la configurazione delle interfacce di rete	`cat /etc/network/interfaces`

Primi programmi

Usiamo un LED

Colleghiamo l'anodo (gambo lungo, +) di un LED al GPIO pin BCM25 (fisico 22) e il catodo (gambo corto, –) a GND (fisico 20).

Accendiamo il led usando shell script

In un file `led25OnOff.sh` scriviamo:

```
echo Unexporting.
echo 25 > /sys/class/gpio/unexport #
echo 25 > /sys/class/gpio/export #
cd /sys/class/gpio/gpio25 #

echo Setting direction to out.
echo out > direction #
echo Setting pin high.
echo 1 > value #
sleep 1 #
echo Setting pin low
echo 0 > value #
sleep 1 #
echo Setting pin high.
echo 1 > value #
sleep 1 #
echo Setting pin low
echo 0 > value #
```

Uso: `sudo bash led25OnOff.sh`

Accendiamo il led usando gpio

In un file **led25Gpio.sh** scriviamo:

```
gpio readall #
echo Setting direction to out
gpio mode 6 out #
echo Write 1
gpio write 6 1 #
sleep 1 #
echo Write 0
gpio write 6 0 #
```

Uso: **bash led25Gpio.sh**

Accendiamo il led usando Python

In un file **ledPython25.py** scriviamo:

```
import RPi.GPIO as GPIO
import time
'''
---------------------------------
CONFIGURATION
---------------------------------
'''
GPIO.setmode(GPIO.BCM)
GPIO.setup(25,GPIO.OUT)
'''
---------------------------------
main activity
---------------------------------
'''
while True:
   GPIO.output(25,GPIO.HIGH)
   time.sleep(1)
   GPIO.output(25,GPIO.LOW)
   time.sleep(1)
```

Uso: **python ledPython25.py**

Usiamo un SONAR HC-SR04

Il sensore è composto da un trasmettitore ad ultrasuoni e un ricevitore, ha una portata da **2 a 400 cm** e una precisione della misurazione di **+/- 0,5 cm**. La velocità del suono è di **343,3 m/s** o **34330 cm/s**.

Colleghiamo il nostro *UltraSonic Distance Measure Module Range Sensor* come segue:

```
- VCC : pin fisico 4 (+5v)
- GND : pin fisico 6 (GND)
- TRIG: pin fisico 11 (WPI 0, BCM 17)
- ECHO: pin fisico 13 (WPI 2, BCM 27)
```

Attiviamo il sonar usando Python

```
# File: sonar.py
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
TRIG = 17
ECHO = 27

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)

GPIO.output(TRIG, False)    #TRIG parte LOW
print ('Waiting a few seconds for the sensor to settle')
```

```
time.sleep(2)

while True:
   GPIO.output(TRIG, True)     #invia impulsoTRIG
   time.sleep(0.00001)
   GPIO.output(TRIG, False)

   #attendi che ECHO parta e memorizza tempo
   while GPIO.input(ECHO)==0:
      pulse_start = time.time()

   # register the last timestamp at which the receiver detects the signal.
   while GPIO.input(ECHO)==1:
      pulse_end = time.time()

   pulse_duration = pulse_end - pulse_start
   distance = pulse_duration * 17165    #distance = vt/2
   distance = round(distance, 1)
   #print ('Distance:',distance,'cm')
   print ( distance )
   sys.stdout.flush()   #Importante!
   time.sleep(0.25)
```

Questo codice visualizza sul dispositivo standard di output l'informazione sulla distanza rilevata.

Attiviamo il sonar usando C

Definiamo un programma analogo a precedente, ma scritto in linguaggio C++, che invia sul dispositivo standard di output il valore della distanza rilevata dal Sonar.

Si veda: *SonarAlone.c*

```
#Compilazione
   g++  SonarAlone.c -l wiringPi -o  SonarAlone
#Esecuzione
   ./SonarAlone
```

Accendiamo il LED se qualcosa si avvicina

Consideriamo il seguente requisito:

(requisito *LedSonar*) : Accendere il Led se il Sonar rileva una distanza inferiore a un limite prefissato.

Soluzione in C

WORKTODO: LedSonar

- Scrivere un programma **LedSonar.c** che risolve il problema estendendo il comportamento del programma *SonarAlone.c*.

Soluzione in Python

Si tratta di realizzare una prima semplice catena *input-elaborazione-output*; i dati emessi dal sonar sullo dispositivo standard di uscita possono essere acquisiti da un altro programma attraverso il meccanismo delle *pipe* di Linux/Unix.

Definiamo quindi un semplice programma Python che legge da standard input e scrivere quanto letto su standard output.

```
#File: ReadInput.py
import sys
for line in sys.stdin:
   print(line)
```

Testiamo il programma facendo visualizzare il programma stesso:

```
cat ReadInput.py | python ReadInput.py
```

A questo punto ridirigiamo i dati generati dal sonar al nostro programma:

```
   python sonar.py | python ReadInput.py
oppure:
   ./SonarAlone | python ReadInput.py
```

Ora modifichiamo il programma che riceve i dati di ingresso in modo da attivare/disattivare il LED:

```
#File: LedControl.py
import sys
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(25,GPIO.OUT)

for line in sys.stdin:
   print(line)
   v = float(line)
   if v <= 10 :
      GPIO.output(25,GPIO.HIGH)
   else:
      GPIO.output(25,GPIO.LOW)
```

Concateniamo i programmi:

```
python sonar.py | python LedControl.py
```

WORKTODO: blinking the Led

- Scrivere un programma **LedBlinkSonar** che attiva il *blinking* del Led quando il Sonar rileva una distanza inferiore a un limite prefissato.
