Robotica avanzata: Raspberry PI e Python

Lezione 4: programmare un robot Alphabot

La classe AlphaBot.py

La classe AlphaBot implementa alcuni metodi comodi per gestire le uscite digitali GPIO che comandano i motori.

La classe implementa i metodi:

- forward()
- stop()
- backward()
- left()
- right()
- setPWMA(value)
- setPWMB(value)
- setMotor(valueL, valueR)

La classe AlphaBot.py

La classe funge da wrapper sui metodi della libreria RPi.GPIO!

In questo modo risulta molto più semplice e intuitivo azionare i motori, visto che per ogni motore occorrono 3 uscite:

- IN1
- IN2
- PWM

La classe AlphaBot.py

La classe, specifica per il robot AlphaBot, è in realtà del tutto generale e permette di pilotare qualunque driver di motori collegato al GPIO.

Inoltre anche in soluzioni più complesse in cui ad esempio Rasperry PI utilizzi Arduino per pilotare i motori tramite una interfaccia seriale, risulta fortemente consigliato sviluppare un wrapper in tutto e per tutto simile alla classe AlphaBot.py in modo da creare un software che sia il più possibile modulare.

I sensori di ostacoli a infrarossi

Questi corrispondono a input digitali e vengono utilizzati nel solito modo tramite il modulo RPI.GPIO.

DL e DR sono i nomi degli input da usarsi nel programma, mentre GPIO.PUD_UP serve per attivare la resistenza di pull up utile per gli input


```
GPIO.setup(DR,GPIO.IN,GPIO.PUD_UP)
GPIO.setup(DL,GPIO.IN,GPIO.PUD_UP)
```

I sensori di ostacoli a infrarossi

I sensori ad infrarossi vengono letti semplicemente tramite le chiamate sottostanti. La lettura di un 1 corrisponde a sensore che NON rileva ostacoli!

```
DR_status = GPIO.input(DR)
DL_status = GPIO.input(DL)
```

Gli encoder ottici

Gli encoder ottici sono montati sugli assi dei motori e permettono di rilevare la rotazione delle ruote tramite sensori ottici IR che generano un impulso tachimetrico. Ogni giro di ruota corrisponde a 20 impulsi quindi a 40 transizioni alto-basso/basso-alto.

Come strutturare il main loop di un robot

```
import RPi.GPIO as GPIO
import time
from AlphaBot import AlphaBot
Ab = AlphaBot()
DR = 16
DL = 19
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(DR,GPIO.IN,GPIO.PUD_UP)
GPIO.setup(DL,GPIO.IN,GPIO.PUD_UP)
try:
    while True:
        """
            QUESTO E' IL LOOP PRINCIPALE DEL ROBOT
            Ad ogni ciclo si leggono i sensori e si controllano i motori.
            In questo loop si implementano gli algoritmi che implementano
            la logica del robot.
        """
except KeyboardInterrupt:
    GPIO.cleanup() #pulitura GPIO
```

Un primo robot!

Ora tocca a voi!

Utilizzando la classe AlphaBot.py e le indicazioni fornite sinora progettate un semplice robot in grado di muoversi nell’ambiente rilevando gli ostacoli!

Gli encoder ottici

```
import RPi.GPIO as GPIO
import time
from AlphaBot import AlphaBot
Ab = AlphaBot()
Ab.stop()
ENCR = 8
ENCL = 7
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(ENCL,GPIO.IN,GPIO.PUD_UP)
GPIO.setup(ENCR,GPIO.IN,GPIO.PUD_UP)
ENCR_status = 0
ENCL_status = 0
ENCR_counter = 0
ENCL_counter = 0
try:
    while True:
        ENCR_status_new = GPIO.input(ENCR)
        ENCL_status_new = GPIO.input(ENCL)
        if ENCR_status_new != ENCR_status:
            ENCR_counter = ENCR_counter + 1
        ENCR_status = ENCR_status_new
        if ENCL_status_new != ENCL_status:
            ENCL_counter = ENCL_counter + 1
        ENCL_status = ENCL_status_new

        print (ENCR_counter,ENCL_counter)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Un database su un robot

In varie applicazioni pratiche un robot può necessitare di una persistenza dati. I tradizionali DB server sarebbero però pesanti e chiaramente sovra-dimensionati per una applicazione robotica in cui l'obiettivo principale sono le performance!

Esiste un DB relazionale molto utilizzato nelle applicazioni di tipo mobile che risulta privo di server (serverless) e che risulta implementato nella libreria nativa di Python:

SQLITE3

Un database su un robot

Utilizziamo SQLITE3:

- Non occorre installare nulla, tutto è implementato dentro la libreria sqlite3 di Python e risulta pre-installata.

- Il db è un file binario creato da sqlite3 nel momento in cui si realizza una “connessione”.

- Si tratta di un db relazionale dotato di linguaggio SQL standard, per cui non occorre imparare davvero nulla di nuovo per iniziare a utilizzarlo.

- Nel caso: https://docs.python.org/3/library/sqlite3.html

Un database su un robot

Creiamo un db con una tabella:

```
import sqlite3
conn = sqlite3.connect('robot.db')
c = conn.cursor()
# Create table
c.execute('''CREATE TABLE position
             (date text, rwheelspace real, lwheelspace real)''')
# Save (commit) the changes
conn.commit()
# We can also close the connection if we are done with it.
# Just be sure any changes have been committed or they will be
lost.
conn.close()
```

Un database su un robot

Inseriamo un record ed eseguiamo una query:

```
import sqlite3
from time import strftime
conn = sqlite3.connect('robot.db')
c = conn.cursor()
# Insert a row of data
c.execute("INSERT INTO position VALUES ('"+strftime("%a, %d %b %Y
%H:%M:%S +0000")+"',10.3,20.4)")

# Save (commit) the changes
conn.commit()
for row in c.execute('SELECT * FROM position ORDER BY date asc'):
        print(row)

# We can also close the connection if we are done with it.
# Just be sure any changes have been committed or they will be lost.
conn.close()
```

Riconoscere oggetti colorati con OpenCV

Nonostante le apparenze risulta piuttosto semplice utilizzare OpenCV per il riconoscimento di oggetti nelle immagini.

Il punto di partenza è il semplice programma visto nella lezione 3 che permette di mostrare a schermo le immagini della Pi Camera in streaming…

Infatti il riconoscimento degli oggetti va eseguito su tutti i frame, uno dopo l'altro. L'operazione è piuttosto onerosa, ma fattibile anche dal Raspberry PI!

Riconoscere oggetti colorati con OpenCV

Occorre implementare una funzione che utilizzando la libreria OpenCV svolga il compito desiderato:

```
def findObject(frame):
    """
    La funzione legge in input un frame letto tramite VideoCapture
    e individua gli oggetti rossi presenti nell'immagine.
    La funzione ritorna una tupla contenente:
    - objPresent: bool che indica se almeno un oggetto e presente
    - xcenter: coordinata x del centro dell'oggetto
    - ycenter: coordinata y del centro dell'oggetto
    - frame: il frame contenente un rettangolo che evidenzia l'oggetto rilevato
    """
    font=cv2.FONT_HERSHEY_SIMPLEX
    kernelOpen=np.ones((5,5))
    kernelClose=np.ones((20,20))

    lower_red_0 = np.array([0, 100, 100])
    upper_red_0 = np.array([10, 255, 255])
    lower_red_1 = np.array([180 - 10, 100, 100])
    upper_red_1 = np.array([180, 255, 255])
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    mask_0 = cv2.inRange(hsv, lower_red_0 , upper_red_0)
    mask_1 = cv2.inRange(hsv, lower_red_1 , upper_red_1 )

    mask = cv2.bitwise_or(mask_0, mask_1)

    maskOpen=cv2.morphologyEx(mask,cv2.MORPH_OPEN,kernelOpen)
    maskClose=cv2.morphologyEx(maskOpen,cv2.MORPH_CLOSE,kernelClose)

    maskFinal=maskClose

    im,conts,h=cv2.findContours(maskFinal.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
    cv2.drawContours(frame,conts,-1,(255,0,0),3)
    objPresent = False
    xcenter = -1
    ycenter = -1

    if len(conts) > 0:
        x,y,w,h=cv2.boundingRect(conts[0])
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255), 2)
        xcenter=x+w/2
        ycenter=y+h/2
        cv2.putText(frame, str(int(xcenter))+','+str(int(ycenter)),(x,y+h),font,.5,(0,255,255))
        objPresent = True
    return objPresent, xcenter, ycenter, frame
```

Un secondo robot!

Ora tocca a di nuovo voi!

Utilizzando la classe AlphaBot.py e a vostra scelta:

- la funzionalità di OpenCV
- la possibilità di avere una persistenza dati con Sqlite3
- gli encoder ottici

create un robot che svolga un compito complesso!