



11_Pacchetto-RF24-Raspberry_Relazione

Citterio Giorgio e Colombo Umberto


Lo scopo di quest'attività è convertire i programmi python da seriale a programmi per interfacciarsi con il modulo radio RF24, per poi caricarli su raspberry.

Attività preliminare

Per prima cosa abbiamo studiato la piedinatura del GPIO del nostro raspberry e ci abbiamo collegato il modulo RF24.

Raspberry Pi: il connettore GPIO

Le diverse versioni del connettore GPIO

 <https://www.vincenzov.net/tutorial/RaspberryPi/connettore-GPIO.htm>

Raspberry Pi: LED e interruttori con bash

Come accendere un LED usando la GPIO. Come usare un interruttore

 <https://www.vincenzov.net/tutorial/RaspberryPi/helloREALworld-sh.htm>

Ricezione dati sensore (parte 2)

In questa prima parte abbiamo fatto il programma per ricevere ed elaborare i dati del sensore inviati da arduino tramite modulo RF24 a raspberry.

programma python che riceve i dati dal sensore e crea il file JSON:

```
import time
import sys
import struct
import json
import pigpio
from nrf24 import *
import os

PIGPIONAME='localhost'
PIGPIOPORT=8888
READINGPIPE='00001'
IDCORRETTO='BE'
DESTINATARIOCORRETTO='P001'
```

```

MIO_TIPO='S1'
lista = []
pathJ = os.getcwd()+'/datiSensore.json'

# connessione a pigpiod
pi = pigpio.pi(PIGPIONAME, PIGPIOPORT)
if not pi.connected:
    print("Pigpiod non connesso. Lanciare: SUDO PIGPIOD")
    sys.exit()

# Crea l'oggetto NRF24
nrf = NRF24(pi, ce=17, payload_size=32, channel=76,data_rate=RF24_DATA_RATE.RATE_2MBPS, pa_level=RF24_PA.LOW)

# apre la pipe
nrf.set_address_bytes(5)
nrf.open_reading_pipe(RF24_RX_ADDR.P1, READINGPIPE)

#lettura pacchetto 32 byte
while True:
    if nrf.data_ready():
        pack=(struct.unpack("2s 4s 4s 2s 4s 16s",nrf.get_payload()))
        id=pack[0].decode()
        mittente=pack[1].decode()
        destinatario=pack[2].decode()
        tipo=pack[3].decode()
        valoreSensore=pack[4].decode()
        vuoto=pack[5].decode()
        if ((id==IDCORRETTO)and(destinatario==DESTINATARIOCORRETTO)):
            print(pack)
            print("id e destinatario corretti e valoreSensore = " + valoreSensore)
            s=int(valoreSensore)
            dataOra = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
            dizionario = {'DataOra': dataOra, 'Valore' : s}
            lista.append(dizionario)
            data = json.dumps(lista[-10:])
            with open(pathJ, 'w') as fp:
                fp.write(data)
            with open(pathJ, 'r') as fp:
                lista2 = json.load(fp)
            print(lista2)

```

programma python che crea il server Flask con i dati del sensore:

```

from flask import render_template
from flask import Flask
import os
import json

pathJ = os.getcwd()+'/datiSensore.json'
app = Flask(__name__)
@app.route('/')

def returnHtml():
    with open(pathJ, 'r') as fp:
        lista = json.load(fp)
    return render_template('index.html', dizValori=lista)
if __name__=="__main__":
    app.run(debug=True)

```

template HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv=refresh content=2>
  <title>Dati sensore</title>
  <style>table, th, td{border: 1px solid black;}</style>
</head>
<body>
  <table>
    <tr>
      <th>Data e ora</th>
      <th>Valore</th>
      {% for i in dizValori %}
        <tr>
          <td>{{i["DataOra"]}}</td>
          <td>{{i["Valore"]}}</td>
        </tr>
      {% endfor %}
    </tr>
  </table>
</body>
</html>
```

Invio comandi motore (parte 3)

In questa parte dell'attività abbiamo creato il programma python che prende i dati dal server Flask su raspberry e li invia al motore tramite RF24, il tutto collegato ad arduino.

codice python elaborazione dati e invio pacchetto:

```
import time
import sys
import pigpio
from nrf24 import *
import struct
from flask import Flask, request, render_template, redirect

PIGPIONAME='localhost'
PIGPIOPORT=8888
WRITINGPIPE='00001'
ID=b"BE"
MITTENTE=b"P001"
DESTINATARIO=b"A001"
TIPO=b"A1"
VUOTO=(-"*16).encode()
app = Flask(__name__)

# connessione a pigpiod
pi = pigpio.pi(PIGPIONAME, PIGPIOPORT)
if not pi.connected:
    print("Pigpiod non connesso. Lanciare: SUDO PIGPIOD")
    sys.exit()
```

```

# Crea l'oggetto NRF24
nrf = NRF24(pi, ce=17, payload_size=32, channel=76,data_rate=RF24_DATA_RATE.RATE_2MBPS, pa_level=RF24_PA.LOW)

# apre le pipe
nrf.set_address_bytes(5)
nrf.open_writing_pipe(WRITINGPIPE)

#scrittura pacchetto 32 byte
@app.route("/")
def inviaFormVuoto():
    return(render_template("index.html"))

@app.route("/ricevi")
def riceviForm():
    val = request.args["velocita"]
    if(request.args["btn"] == "avanti"):
        direzione = b"A"
    else:
        direzione = b"I"
    v = str(val).zfill(3).encode()
    msg=struct.pack("2s 4s 4s 2s 1s 3s 16s", ID,MITTENTE,DESTINATARIO,TIPO,direzione,v,VUOTO)
    nrf.send(msg)
    print(msg)
    return redirect('/')

```

template HTML:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Controllo motore</title>
</head>
<body>
  <h2>Controllo motore</h2>
  <form action="/ricevi">
    <label for="velocita">Velocità:</label>
    <input type="number" id="velocita" name="velocita" required min="0" max="255" placeholder="0-255">
    <br>
    <input type="radio" name="btn" checked="checked" value="avanti">
    <label> Avanti </label>
    <br>
    <input type="radio" name="btn" value="indietro">
    <label> Indietro </label>
    <br>
    <input type="submit" value="Invia">
  </form>
</body>
</html>

```

Test d'insieme (parte 4)

In quest'ultima parte dell'attività abbiamo creato il programma python che crea un server Flask che gira su raspberry dal quale si può sia inviare i comandi al motore sia visualizzare i dati del sensore.

programma python che legge i dati del sensore e crea il file JSON:

```

import time
import sys
import struct
import json
import pigpio
from nrf24 import *
import os

PIGPIONAME='localhost'
PIGPIOPORT=8888
READINGPIPE='00001'
IDCORRETTO='BE'
DESTINARIOCORRETTO='P001'
MIO_TIPO='S1'
lista = []
pathJ = os.getcwd()+'/datiSensore.json'

# connessione a pigpiod
pi = pigpio.pi(PIGPIONAME, PIGPIOPORT)
if not pi.connected:
    print("Pigpiod non connesso. Lanciare: SUDO PIGPIOD")
    sys.exit()

# Crea l'oggetto NRF24
nrf = NRF24(pi, ce=17, payload_size=32, channel=76,data_rate=RF24_DATA_RATE.RATE_2MBPS, pa_level=RF24_PA.LOW)

# apre la pipe
nrf.set_address_bytes(5)
nrf.open_reading_pipe(RF24_RX_ADDR.P1, READINGPIPE)

#lettura pacchetto 32 byte
while True:
    if nrf.data_ready():
        pack=(struct.unpack("2s 4s 4s 2s 4s 16s",nrf.get_payload()))
        id=pack[0].decode()
        mittente=pack[1].decode()
        destinatario=pack[2].decode()
        tipo=pack[3].decode()
        valoreSensore=pack[4].decode()
        vuoto=pack[5].decode()
        if ((id==IDCORRETTO)and(destinatario==DESTINARIOCORRETTO)):
            print(pack)
            print("id e destinatario corretti e valoreSensore = " + valoreSensore)
            s=int(valoreSensore)
            dataOra = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
            dizionario = {'DataOra': dataOra, 'Valore' : s}
            lista.append(dizionario)
            data = json.dumps(lista[-10:])
            with open(pathJ, 'w') as fp:
                fp.write(data)
            with open(pathJ, 'r') as fp:
                lista2 = json.load(fp)
            print(lista2)

```

programma python che crea il server Flask con controllo motore e visualizzazione dati sensore:

```

from flask import Flask, request, render_template, redirect
import struct
import os
import json

```

```

import time
import sys
import pigpio
from nrf24 import *

#costanti rf24
PIGPIONAME='localhost'
PIGPIOPORT=8888
WRITINGPIPE='00001'

#costanti motore
ID=b"BE"
MITTENTE=b"P001"
DESTINATARIO=b"A001"
TIPO=b"A1"
VUOTO=(-"*16).encode()

lista = []
pathJ = os.getcwd()+'/datiSensore.json'
direzione = b"A"
app = Flask(__name__)

# connessione a pigpiod
pi = pigpio.pi(PIGPIONAME, PIGPIOPORT)
if not pi.connected:
    print("Pigpiod non connesso. Lanciare: SUDO PIGPIOD")
    sys.exit()

# Crea l'oggetto NRF24
nrf = NRF24(pi, ce=17, payload_size=32, channel=76, data_rate=RF24_DATA_RATE.RATE_2MBPS, pa_level=RF24_PA.LOW)

# apre le pipe
nrf.set_address_bytes(5)
nrf.open_writing_pipe(WRITINGPIPE)

#programma sensore
@app.route('/')
def returnHtml():
    with open(pathJ, 'r') as fp:
        lista = json.load(fp)
        return render_template('index.html', dizValori=lista)
if __name__=="__main__":
    app.run(debug=True)

#programma attuatore
@app.route("/ricevi")
def riceviForm():
    val = request.args["velocita"]
    if(request.args["btn"] == "avanti"):
        direzione = b"A"
    else:
        direzione = b"I"
    v = str(val).zfill(3).encode()
    msg=struct.pack("2s 4s 4s 2s 1s 3s 16s",ID,MITTENTE,DESTINATARIO,TIPO,direzione,v,VUOTO)
    nrf.send(msg)
    print(msg)
    nrf.wait_until_sent() #funzione per la sincronizzazione del modulo rf24 dopo l'invio
    nrf.power_up_rx() #funzione che rimette il modulo in ascolto
    return redirect("/")

```

template HTML:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv=refresh content=5>
  <title>Controllo motore e sensore</title>
  <style>
    table, th, td{border: 1px solid black;}
  </style>
</head>
<body>
  <h2>Controllo motore</h2>
  <form action="/ricevi">
    <label for="velocita">Velocità:</label>
    <input type="number" id="velocita" name="velocita" required min="0" max="255">
    <input type="submit" value="Invia">
    <br>
    <input type="radio" name="btn" checked="checked" value="avanti">
    <label> Avanti </label>
    <br>
    <input type="radio" name="btn" value="indietro">
    <label> Indietro </label>
  </form>
  <h2>Tabella valori del sensore</h2>
  <table>
    <tr>
      <th>Data e ora</th>
      <th>Valore</th>
      {% for i in dizValori %}
        <tr>
          <td>{{i["DataOra"]}}</td>
          <td>{{i["Valore"]}}</td>
        </tr>
      {% endfor %}
    </tr>
  </table>
</body>
</html>

```