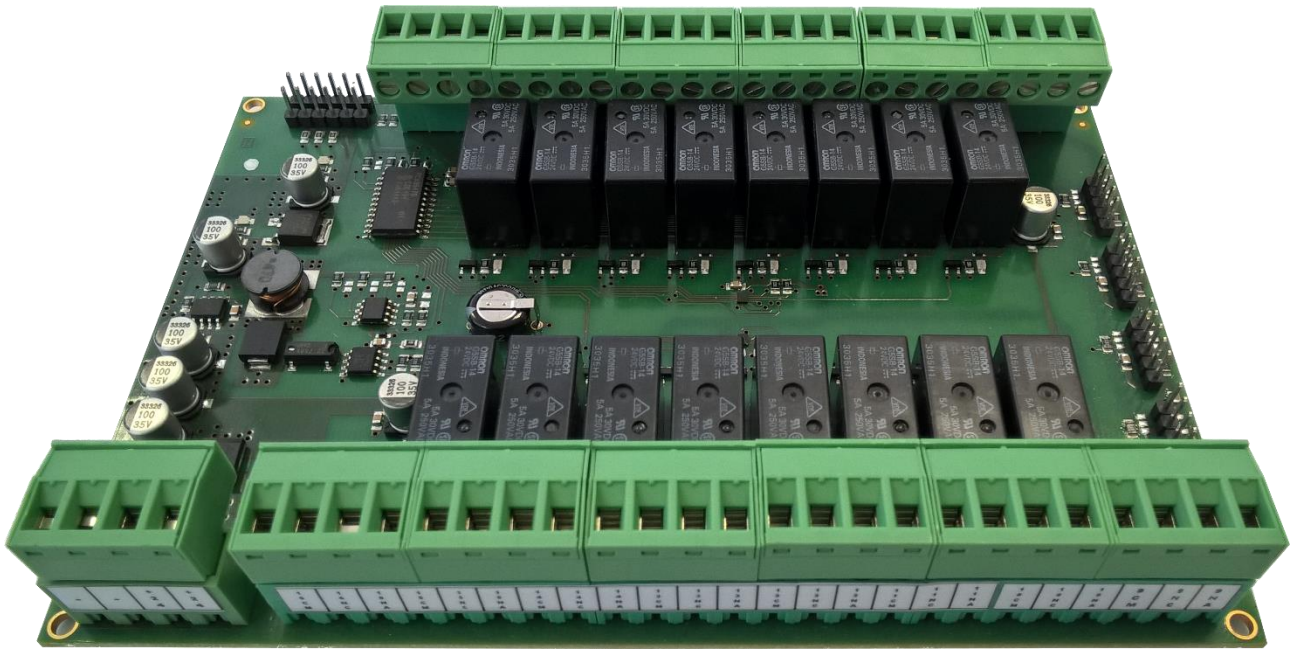
Modulo I2C relè 16 canali DC 24V con RTC e EEPROM



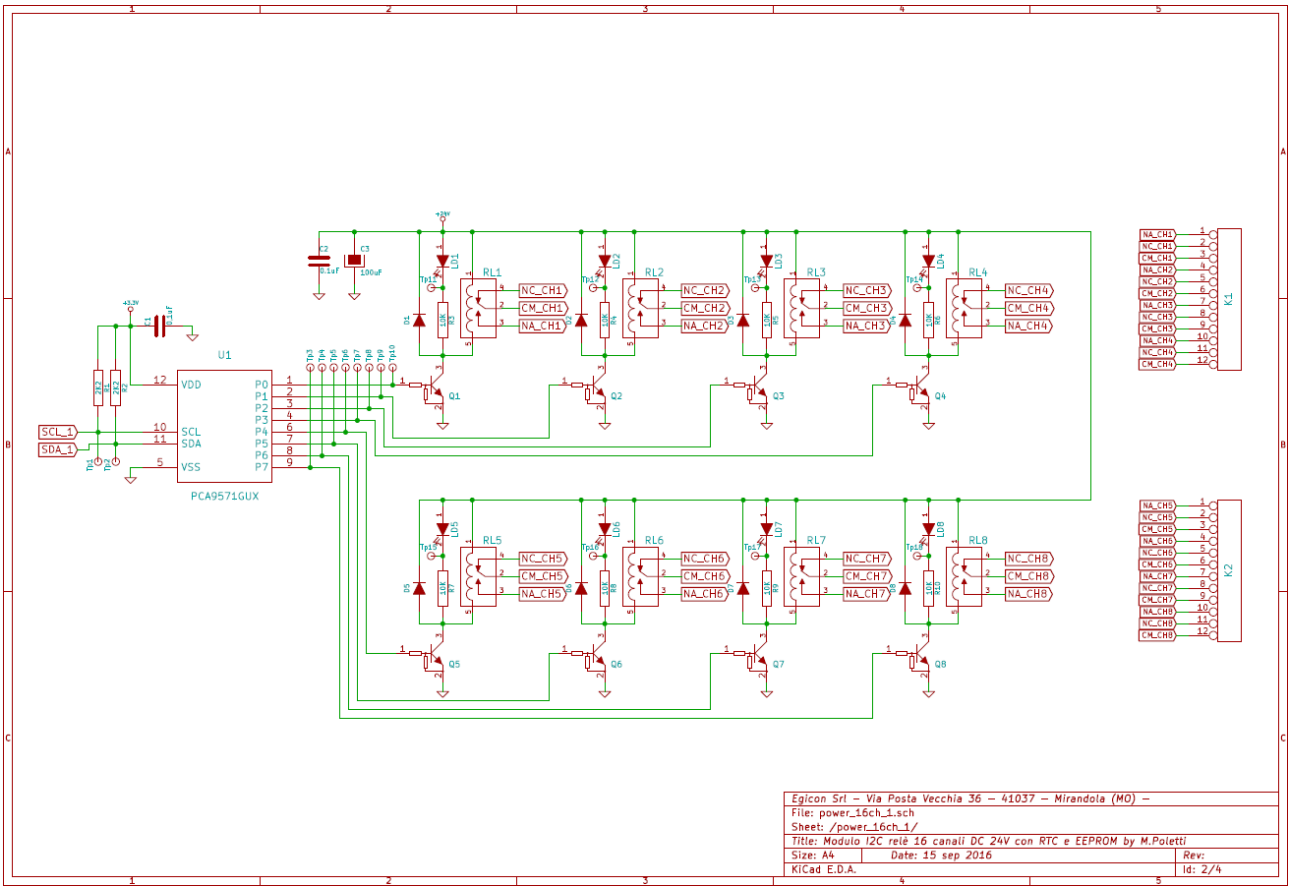
SPECIFICHE

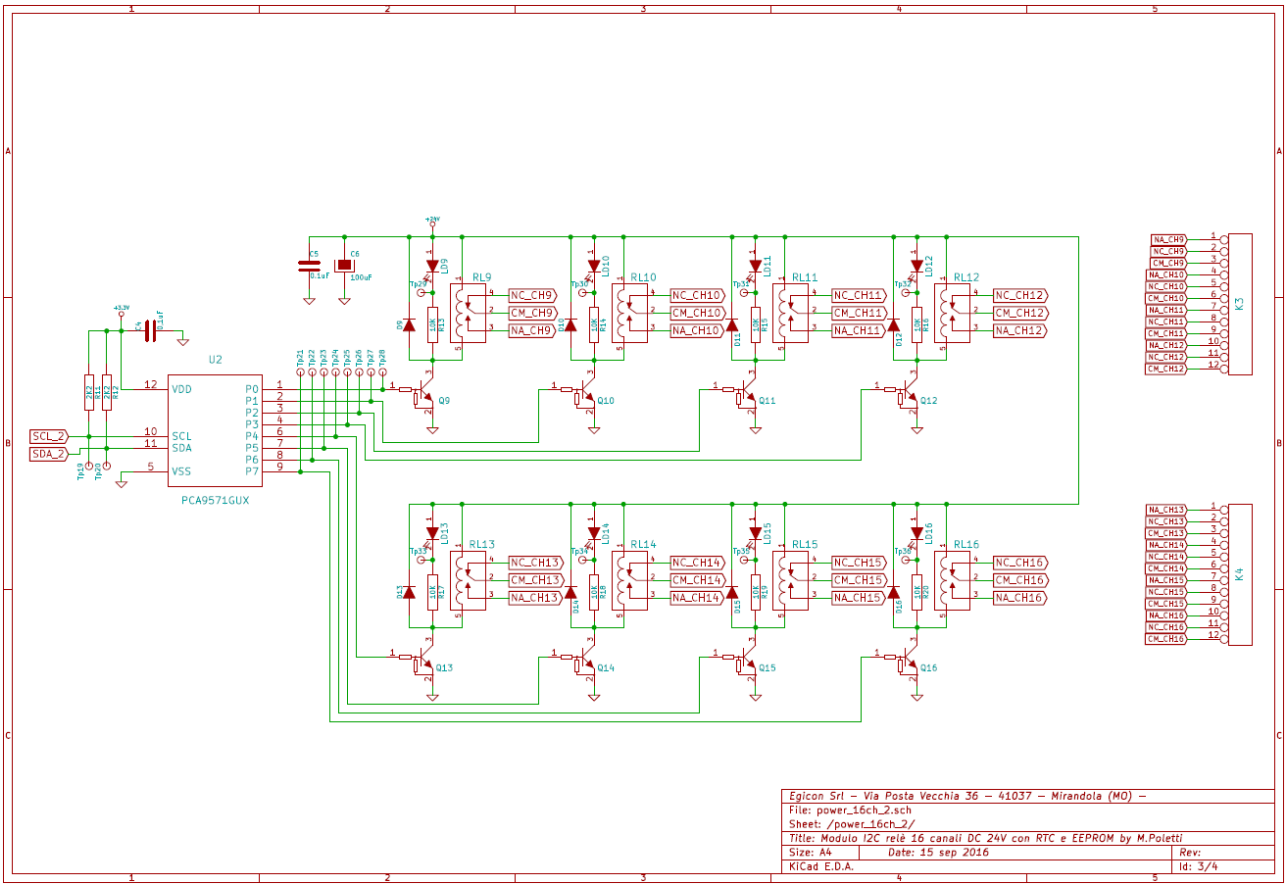
- Tensione di alimentazione 24VDC 0.5A
- Protezione contro inversione polarità di alimentazione
- Indicatori led stato alimentazioni di servizio 5VDC e 3.3VDC
- 16 uscite a relè con contatti (CM, NA, NC) su connettori Phoenix
- Potenza contatto relè 250VAC 3A , 30VDC 3A
- Indicatori led per lo stato di uscita dei relè
- 1 RTC (real time clock) PCF8563 con batteria tampone (battery supercap)
- 1 EEPROM 16K ST24C16 I2C
- 4 canali di espansione bus I2C
- Alimentazioni di servizio disponibili a connettore per poter alimentare eventuali periferiche
- Pilotaggio della scheda mediante protocollo I2C
- Possibilità di cambiare l'indirizzo I2C della scheda mediante jumper a saldare
- Compatibile con Arduino e Raspberry Pi
- Dimensioni: 16 x 10,8 cm
- Peso: 280g
- Possibilità di montaggio su barra omega
- ROHS compliant

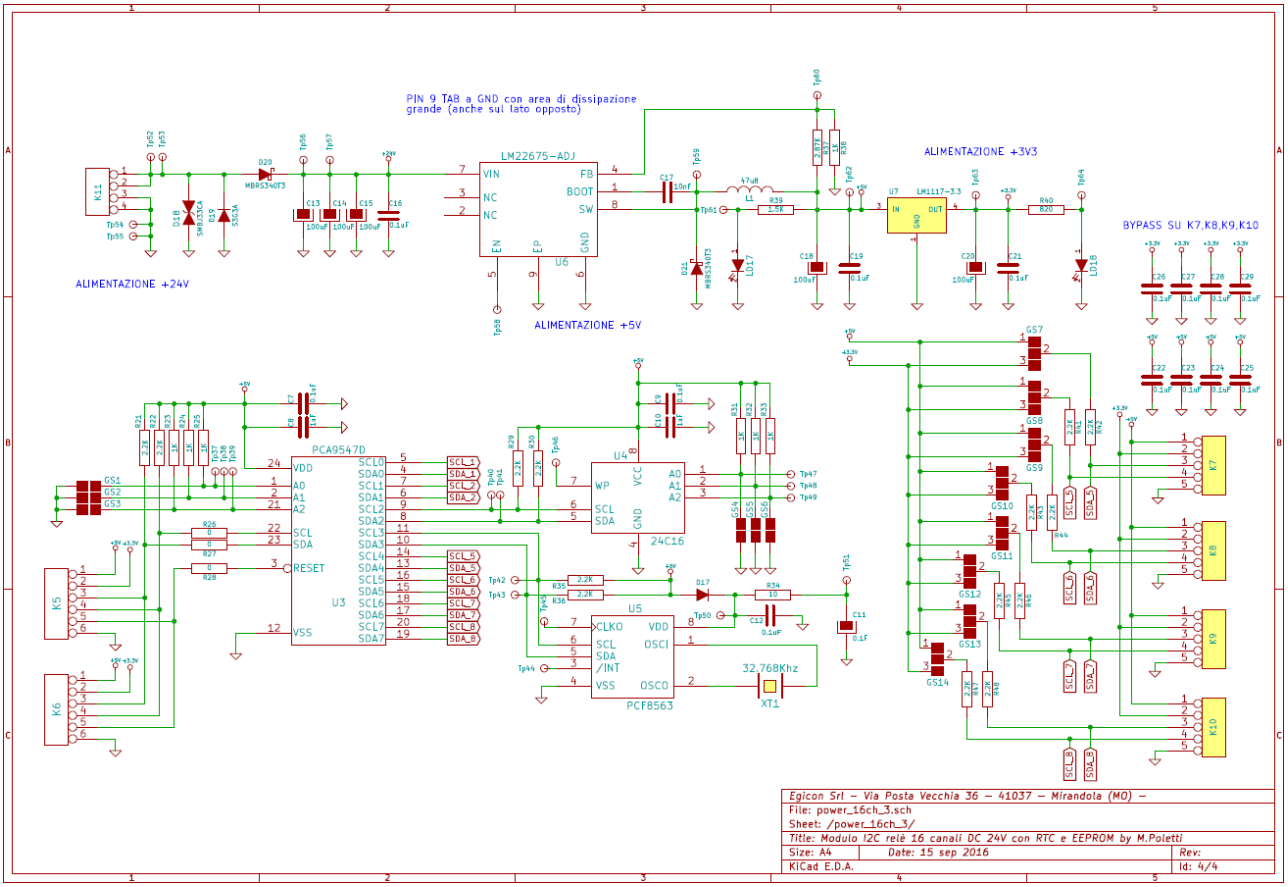
CAMPI DI APPLICAZIONE

- Domotica e illuminazione
- Impianti Irrigazione
- Industriale PLC
- Smart Home

SCHEMA ELETTRICO



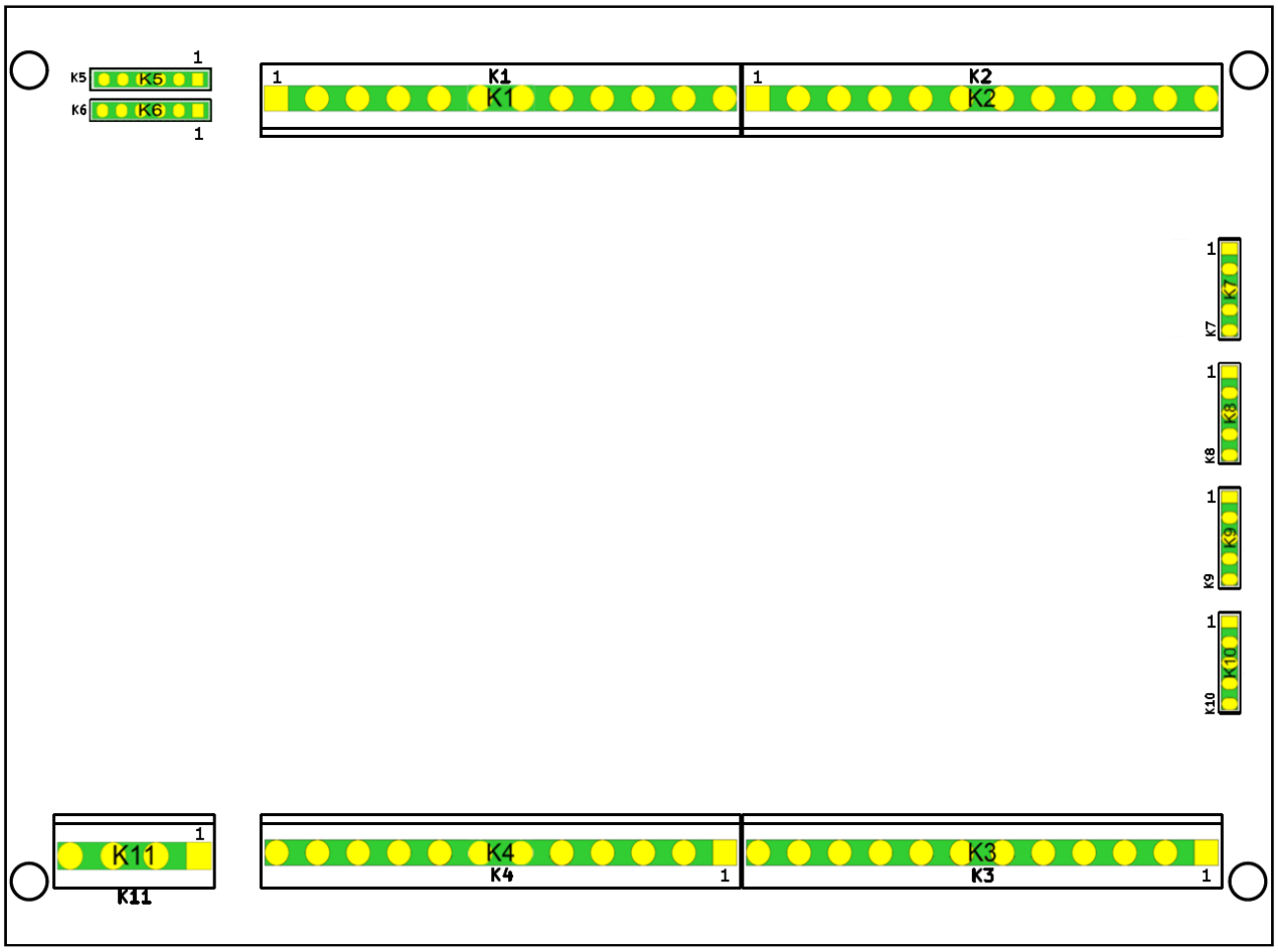




DISTINTA TECNICA

QUANTITY	DESCRIPTION	SMT-THT	REFERENCE
1	Cap.cer. 0805 1uF50V X7R -40+85	SMT	C8
18	Cap.cer. 0805 100nF50V X7R	SMT	C1,C2,C4,C5,C7,C9,C12,C16,C19,C21,C22,C23,C24,C25,C26,C27,C28,C29
7	Cap.El. 100uF 35V -40+85 6.3x7.7	SMT	C3,C6,C13,C14,C15,C18,C20
1	Cap.cer. 0805 10nF50V X7R -40+85	SMT	C17
1	SuperCAP 0.1F 5.5V	THT	C11
17	DIODO 0.2W 0.3A SOD-523F 1N4148WT	SMT	D1,D2,D3,D4,D5,D6,D7,D8,D9,D10,D11,D12,D13,D14,D15,D16,D17
18	LED ROSSO SMD 0805	SMT	LD1,LD2,LD3,LD4,LD5,LD6,LD7,LD8,LD9,LD10,LD11,LD12,LD13,LD14,LD15,LD16,LD17,LD18
16	BCR512 NPN + pol SOT23 50V 0,5A	SMT	Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,Q10,Q11,Q12,Q13,Q14,Q15,Q16
7	RES.SMT 1K 1% 0805 125mW	SMT	R23,R24,R25,R31,R32,R33,R38
18	RES.SMT 2.2K 1% 0805	SMT	R1,R2,R11,R12,R21,R22,R29,R30,R35,R36,R41,R42,R43,R44,R45,R46,R47,R48
3	RES.SMT 0R 1% 0805	SMT	R26,R27,R28
16	RES.SMT 10K 1% 0805	SMT	R3,R4,R5,R6,R7,R8,R9,R10,R13,R14,R15,R16,R17,R18,R19,R20
1	RES.SMT 1.5K 1% 0805	SMT	R39
1	RES.SMT 10R 5% 0805 0.125W	SMT	R34
1	RES.SMT 2.87K 1% 0805	SMT	R37
1	RES.SMT 820R 1% 0805	SMT	R40
1	Indutt.SDR1006-470KL 47UH, 2.3A -40+125	SMT	L1
1	POWER RECT. S3G 3A 400V SMC	SMT	D19
1	BITRANSIENT SUPP. SMDJ33CA 33V	SMT	D18
2	SCHOTTKY POWER RECT MBRS340 3A 40V SMC	SMT	D20,D21
2	PCA9571GUX 8CH I2C MUX XQFN12 NXP	SMT	U1,U2
16	Relè THT OMRON G5SB-14 24 VDC	SMT	RL1,RL2,RL3,RL4,RL5,RL6,RL7,RL8,RL9,RL10,RL11,RL12,RL13,RL14,RL15,RL16
1	LM1117IDTX-3.3 0.8A LowDr. -40+125 TO252	SMT	U7
1	UGPCF85634084X0	SMT	U5
1	M24C16WMN6P SO8 16Kb 2.5V-5.5V -40+85 ST	SMT	U4
1	PCA9547D 8CH I2C MUX SO24 NXP	SMT	U3
1	LM22675MR-ADJ 1A 42V DcDc StepDown	SMT	U6
1	Quarzo MC-306 32.768kHz 20ppm 12.5pF	SMT	XT1
2	CONN.CS.STRIP THT GENERICA 1x6 2.54 vert	THT	K5,K6
4	CONN.CS.STRIP THT GENERICA 1x6 2.54 vert	THT	K7,K8,K9,K10
1	HEADER VERTICALE 5.08MM 4P PHO. 1755752	THT	K11
4	MORSETTIERA MODULARE P.5 VERTICALE 8P 1758076	THT	K1,K2,K3,K4

MAPPA DELLE CONNESSIONI



PIN	REFERENCE	NOME	NOTE	MODULO	PIN	REFERENCE	NOME	NOTE
1	**K1**	**NA_CH1**	RELE' POT 1	1	1	**K11**	**+24V**	IN ALIMENTAZIONE +24V
2	**K1**	**NC_CH1**	RELE' POT 1	1	2	**K11**	**+24V**	IN ALIMENTAZIONE +24V
3	**K1**	**CM_CH1**	RELE' POT 1	1	3	**K11**	**GND**	IN GND
4	**K1**	**NA_CH2**	RELE' POT 2	1	4	**K11**	**GND**	IN GND
5	**K1**	**NC_CH2**	RELE' POT 2	1	1	**K5**	**+5V**	+5V
6	**K1**	**CM_CH2**	RELE' POT 2	1	2	**K5**	**+3.3V**	+3.3V
7	**K1**	**NA_CH3**	RELE' POT 3	1	3	**K5**	**SDA_EXP**	I2C BUS EXTERNAL
8	**K1**	**NC_CH3**	RELE' POT 3	1	4	**K5**	**SCL_EXP**	I2C BUS EXTERNAL
9	**K1**	**CM_CH3**	RELE' POT 3	1	5	**K5**	**RESET_EXP**	I2C BUS EXTERNAL
10	**K1**	**NA_CH4**	RELE' POT 4	1	6	**K5**	**GND**	GND
11	**K1**	**NC_CH4**	RELE' POT 4	1	1	**K6**	**+5V**	+5V
12	**K1**	**CM_CH4**	RELE' POT 4	1	2	**K6**	**+3.3V**	+3.3V
1	**K2**	**NA_CH5**	RELE' POT 5	2	3	**K6**	**SDA_EXP**	I2C BUS EXTERNAL
2	**K2**	**NC_CH5**	RELE' POT 5	2	4	**K6**	**SCL_EXP**	I2C BUS EXTERNAL
3	**K2**	**CM_CH5**	RELE' POT 5	2	5	**K6**	**RESET_EXP**	I2C BUS EXTERNAL
4	**K2**	**NA_CH6**	RELE' POT 6	2	6	**K6**	**GND**	GND
5	**K2**	**NC_CH6**	RELE' POT 6	2	1	**K7**	**+5V**	+5V
6	**K2**	**CM_CH6**	RELE' POT 6	2	2	**K7**	**+3.3V**	+3.3V
7	**K2**	**NA_CH7**	RELE' POT 7	2	3	**K7**	**SCL_5**	I2C BUS EXTERNAL 5
8	**K2**	**NC_CH7**	RELE' POT 7	2	4	**K7**	**SDA_5**	I2C BUS EXTERNAL 5
9	**K2**	**CM_CH7**	RELE' POT 7	2	5	**K7**	**GND**	GND
10	**K2**	**NA_CH8**	RELE' POT 8	2	1	**K8**	**+5V**	+5V
11	**K2**	**NC_CH8**	RELE' POT 8	2	2	**K8**	**+3.3V**	+3.3V
12	**K2**	**CM_CH8**	RELE' POT 8	2	3	**K8**	**SCL_5**	I2C BUS EXTERNAL 6
1	**K3**	**NA_CH9**	RELE' POT 9	3	4	**K8**	**SDA_5**	I2C BUS EXTERNAL 6
2	**K3**	**NC_CH9**	RELE' POT 9	3	5	**K8**	**GND**	GND
3	**K3**	**CM_CH9**	RELE' POT 9	3	1	**K9**	**+5V**	+5V
4	**K3**	**NA_CH10**	RELE' POT 10	3	2	**K9**	**+3.3V**	+3.3V
5	**K3**	**NC_CH10**	RELE' POT 10	3	3	**K9**	**SCL_5**	I2C BUS EXTERNAL 7
6	**K3**	**CM_CH10**	RELE' POT 10	3	4	**K9**	**SDA_5**	I2C BUS EXTERNAL 7
7	**K3**	**NA_CH11**	RELE' POT 11	3	5	**K9**	**GND**	GND
8	**K3**	**NC_CH11**	RELE' POT 11	3	1	**K10**	**+5V**	+5V
9	**K3**	**CM_CH11**	RELE' POT 11	3	2	**K10**	**+3.3V**	+3.3V
10	**K3**	**NA_CH12**	RELE' POT 12	3	3	**K10**	**SCL_5**	I2C BUS EXTERNAL 8
11	**K3**	**NC_CH12**	RELE' POT 12	3	4	**K10**	**SDA_5**	I2C BUS EXTERNAL 8
12	**K3**	**CM_CH12**	RELE' POT 12	3	5	**K10**	**GND**	GND
1	**K4**	**NA_CH13**	RELE' POT 13	4				
2	**K4**	**NC_CH13**	RELE' POT 13	4				
3	**K4**	**CM_CH13**	RELE' POT 13	4				
4	**K4**	**NA_CH14**	RELE' POT 14	4				
5	**K4**	**NC_CH14**	RELE' POT 14	4				
6	**K4**	**CM_CH14**	RELE' POT 14	4				
7	**K4**	**NA_CH15**	RELE' POT 15	4				
8	**K4**	**NC_CH15**	RELE' POT 15	4				
9	**K4**	**CM_CH15**	RELE' POT 15	4				
10	**K4**	**NA_CH16**	RELE' POT 16	4				
11	**K4**	**NC_CH16**	RELE' POT 16	4				
12	**K4**	**CM_CH16**	RELE' POT 16	4				

SOFTWARE DI TEST

Sono stati creati 2 script in python per l'attivazione dei relè della scheda che è stata collegata ad un raspberry pi configurato per utilizzare il bus seriale I2C.

==========

gpo_active.py

==========

```
#!/usr/bin/python
# coding=utf-8

# Select address and channel of PCA9571 I2C general purpose outputs
# I2C Address: 0x25 Fixed
# Multiplexer Channel: 1 - 8
# sudo ./gpo_active.py CHANNEL
# Example: sudo ./gpo_active.py 25 1

import time
import argparse

import RPi.GPIO as GPIO
```

```
import smbus

def I2C_setup(multiplexer_i2c_address, i2c_channel_setup, state):
   I2C_address = 0x25
   if GPIO.RPI_REVISION in [2, 3]:
      I2C_bus_number = 1
   else:
      I2C_bus_number = 0

   bus = smbus.SMBus(I2C_bus_number)
   status_outputs=bus.read_byte(I2C_address)
    #i2c_channel_setup=(i2c_channel_setup-1)**2
   if state == 1:
    i2c_channel_setup=status_outputs|i2c_channel_setup
   elif state == 0:
    i2c_channel_setup=(-i2c_channel_setup-1)&status_outputs
   elif state == -1:
    i2c_channel_setup=0
   print bin(status_outputs)
   bus.write_byte(I2C_address, i2c_channel_setup)
   time.sleep(0)
   print("PCA9571 I2C channel status_outputs:{}".format(bin(bus.read_byte(I2C_address))))

def menu():
   parser = argparse.ArgumentParser(description='Select channel outputs of PCA9571')
   parser.add_argument('address', type=int)
   parser.add_argument('channel_outputs', type=int)
   parser.add_argument('state', type=int)

   args = parser.parse_args()

   I2C_setup(args.address, args.channel_outputs, args.state)

if __name__ == "__main__":
   menu()
```

==========

mux_channel.py

==========

```
#!/usr/bin/python
# coding=utf-8

# Select address and channel of PCA9547 I2C multiplexer
# I2C Address: 0xYY, where YY can be 70 through 77
# Multiplexer Channel: 1 - 8
# sudo ./multiplexer_channel.py ADDRESS CHANNEL
# Example: sudo ./multiplexer_channel.py 70 1

import time
import argparse

import RPi.GPIO as GPIO
import smbus

def I2C_setup(multiplexer_i2c_address, i2c_channel_setup):
   I2C_address = 0x70 + multiplexer_i2c_address % 10
   if GPIO.RPI_REVISION in [2, 3]:
      I2C_bus_number = 1
   else:
```

```
        I2C_bus_number = 0

    bus = smbus.SMBus(I2C_bus_number)
    i2c_channel_setup=i2c_channel_setup + 0x08
    bus.write_byte(I2C_address, i2c_channel_setup)
    time.sleep(0.1)
    print("PCA9547 I2C channel status:{}".format(bin(bus.read_byte(I2C_address))))

def menu():
    parser = argparse.ArgumentParser(description='Select channel of PCA9547 I2C multiplexer')
    parser.add_argument('address', type=int)
    parser.add_argument('channel', type=int)

    args = parser.parse_args()

    I2C_setup(args.address, args.channel)

if __name__ == "__main__":
    menu()
```

Progetto interamente realizzato con software open-source KiCad EDA http://kicad-pcb.org/