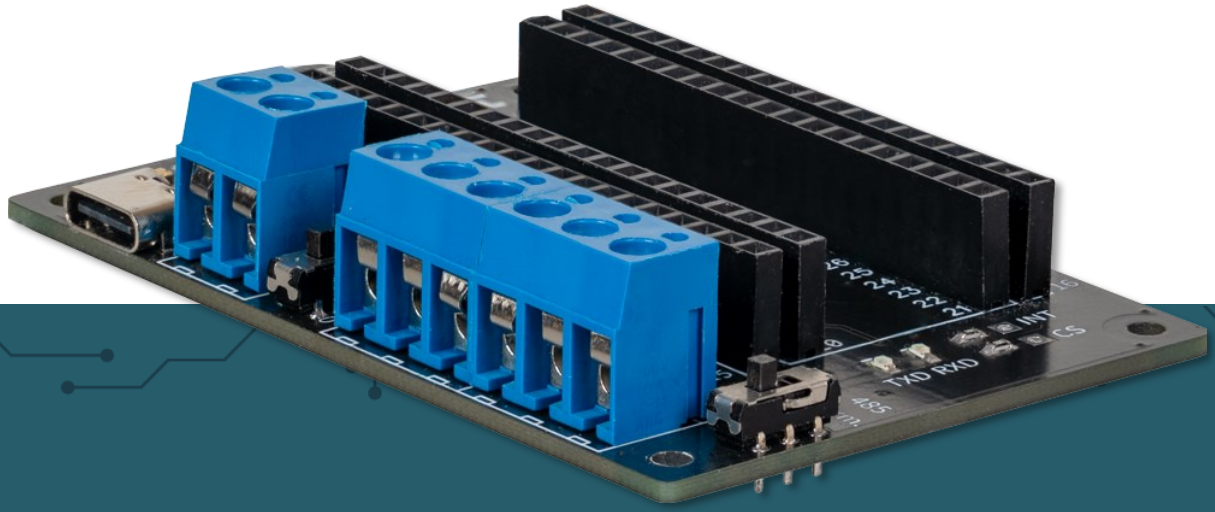


RB-P-CAN-485

Scheda di espansione per Raspberry Pi Pico

Joy-IT powered by SIMAC Electronics GmbH - Pascalstr. 8 - 47506 Neukirchen-Vluyn - www.joy-it.net

Attenzione! Questo manuale è stato tradotto automaticamente; in caso di dubbi, consultare il manuale in inglese o contattare il nostro servizio clienti.
Tutte le richieste di assistenza devono essere in tedesco o in inglese.

1. INFORMAZIONI GENERALI

Gentile cliente,
La ringraziamo per aver scelto il nostro prodotto.
Di seguito vi illustreremo cosa osservare durante la messa in funzione e l'utilizzo di questo prodotto.

In caso di problemi imprevisti durante l'uso, non esiti a contattarci.

2. SPIEGAZIONE DEL MODULO

In questa sezione vengono illustrate brevemente le singole funzioni delle schede di espansione a cui è possibile collegare il Pico.

LED di alimentazione: Indica la presenza di una tensione di 5 V dalla porta USB o dal convertitore di tensione interno alla scheda.

USB (connessione USB-C):
5V Alimentazione.

Ingresso: alimentazione variabile da 6 a 12 V.

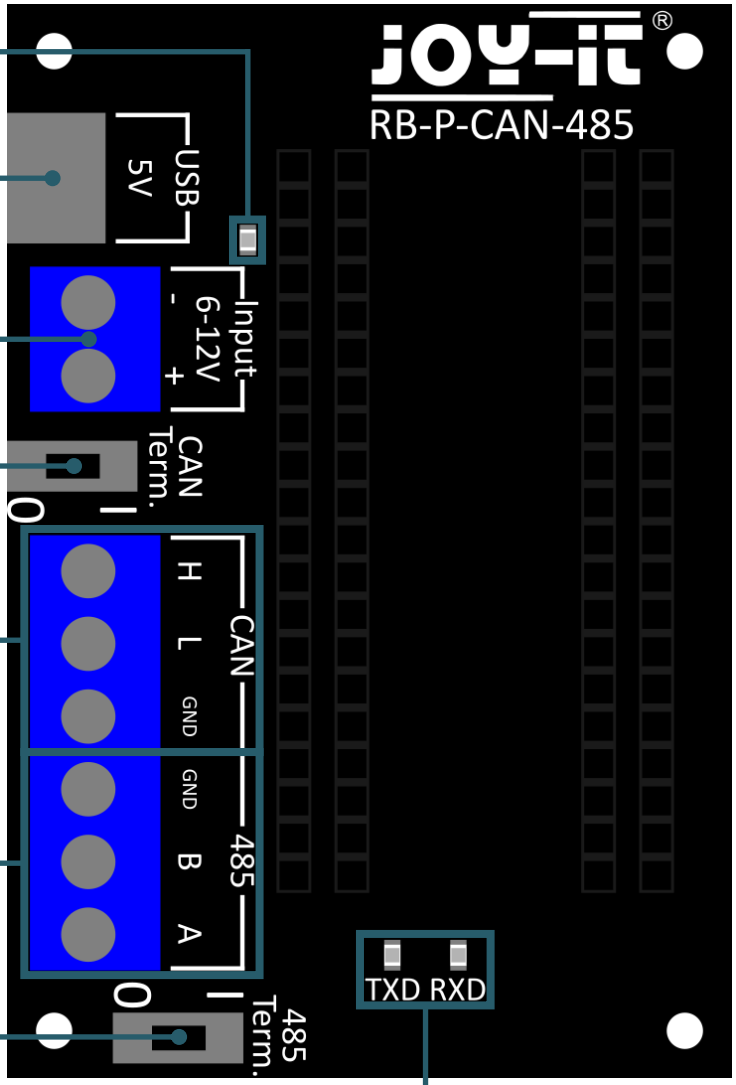
Terminazione CAN. (Terminazione CAN) :
Commuta un resistore da 120Ω sulla posizione "I", utilizzato per la terminazione specializzata del bus.

CAN :
H : Collegamento alla linea CAN-High.
L : Collegamento alla linea CAN-Low.
GND : Collegamento a terra per l'equalizzazione del potenziale tra i nodi.

485 :
A : Collegamento alla linea dati negativa.
B : Collegamento alla linea dati positiva.
GND : Collegamento a terra per equalizzazione del potenziale.

485 Term. (Terminazione 485) :
Commuta un resistore da 120Ω sulla posizione "I", utilizzato per la terminazione specializzata del bus.

TXD | RXD (LED di stato 485):
Visualizza lo stato attuale della linea di trasmissione e ricezione.
Un segnale alto provoca l'accensione dei LED.



3. RASPBERRY PI PICO

Collegare ora un cavo micro USB al computer e al Raspberry Pi Pico per la programmazione.

ATTENZIONE: la connessione USB-C sulla scheda viene utilizzata solo per l'alimentazione. Non vengono trasferiti dati al Raspberry Pi Pico.

Per trasferire i programmi di esempio è possibile utilizzare un programma di sviluppo adeguato di propria scelta. Si consiglia **l'Thonny IDE**.

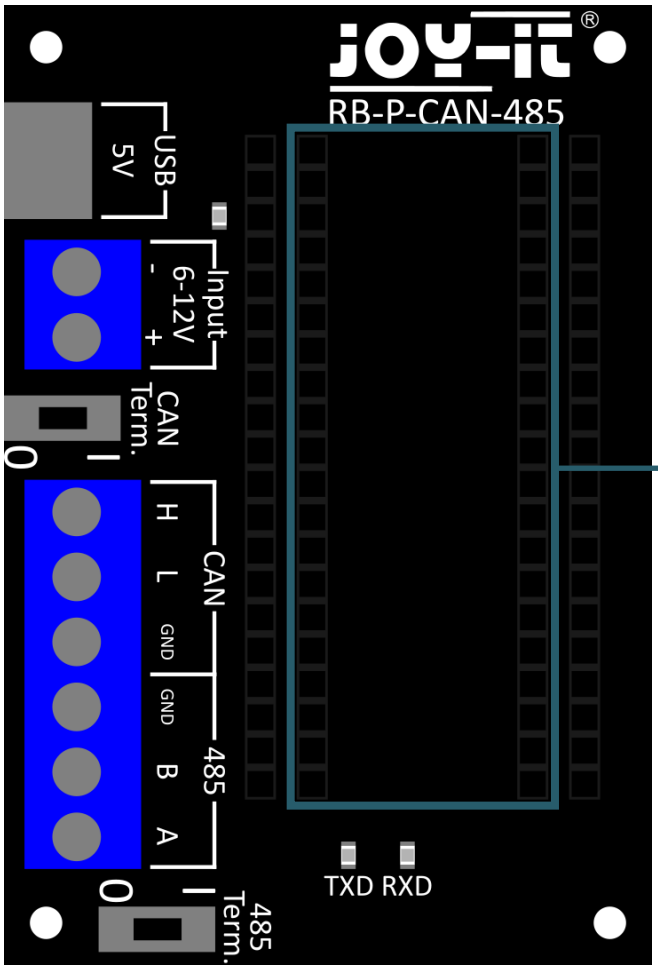
ATTENZIONE: Se siete nuovi al mondo dei microcontrollori e dell'elettronica, non preoccupatevi! Abbiamo preparato per voi una guida speciale per principianti. Questa guida è stata pensata appositamente per le esigenze dei principianti e spiega come utilizzare il Raspberry Pi Pico passo dopo passo.

Dalla configurazione di base all'esecuzione di progetti: in questa guida vi guidiamo attraverso l'intero processo. La nostra guida include spiegazioni di facile comprensione e consigli utili per aiutarvi a sviluppare le vostre capacità in scala con Raspberry Pi Pico in modo rapido ed efficace. Potete scaricare la nostra guida **qui**.

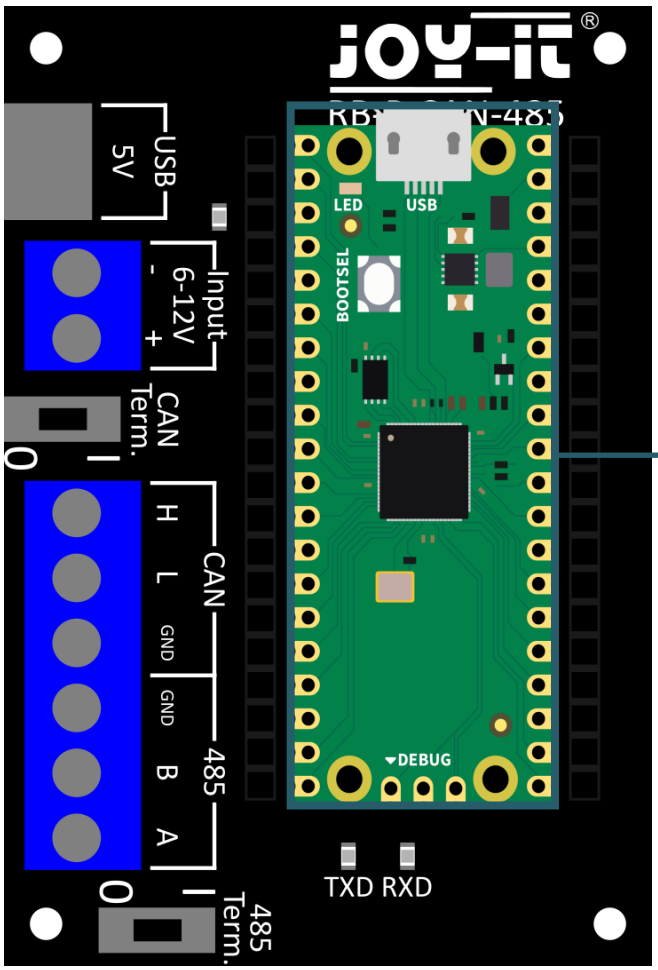
4. COLLEGAMENTO DEL RASPBERRY PI PICO

Nella sezione seguente vedremo come collegare correttamente il Raspberry Pi Pico alla scheda di espansione.

Di seguito è riportata un'illustrazione di come il Raspberry Pi Pico deve essere collegato alla scheda di espansione.



Slot per il Raspberry Pi Pico:
Qui viene inserito il Raspberry Pi Pico.



Se il Raspberry Pi Pico è stato collegato, lo slot dovrebbe avere questo aspetto.

5. ESEMPIO DI CODICE RS485

Ora che avete configurato il vostro Raspberry Pi Pico, potete caricare i due esempi di codice seguenti per la comunicazione tramite l'interfaccia RS485 sul vostro Raspberry Pi Pico. In alternativa, potete anche scaricare gli esempi di codice **qui**.

Esempio di codice del trasmettitore:

```
from machine import UART, Pin
import time

uart0 = UART(0, baudrate=115200, tx=Pin(12), rx=Pin(13))
c=0

txData = b'RS485 send test...\r\n'
uart0.write(txData)
print('RS485 send test...')
time.sleep(0.1)

while True:
    c=c+1
    time.sleep(0.5)
    print (c)#shell output
    uart0.write("{}\r\n".format(c))
```

Esempio di codice del ricevitore:

```
from machine import UART, Pin
import time

uart0 = UART(0, baudrate=115200, tx=Pin(12), rx=Pin(13))

flag = 1
txData = 'RS485 receive test...\r\n'
uart0.write(txData)
print('RS485 receive test...')
time.sleep(0.1)

while True:
    rxData = bytes()
    while uart0.any() > 0:
        rxData = uart0.read()
        print(rxData)
```

6. ESEMPIO DI CODICE PUÒ

Ora che avete configurato il vostro Raspberry Pi Pico, potete caricare i due esempi di codice seguenti per la comunicazione tramite l'interfaccia CAN sul vostro Raspberry Pi Pico.

Per utilizzare questi esempi è necessaria la libreria corrispondente. Noi utilizziamo la libreria **MicroPython_CAN_BUS_MCP2515** di **Longan-Labs**, pubblicata sotto **licenza MIT**.

Esempio di codice del trasmettitore dalla libreria:

```
'''
Comments reworked by Joy-IT, 25.03.2024
----------------------------------------------
A simple example to send data to a CAN bus.
'''

# Import necessary libraries
import sys  # System-specific parameters and functions
import time  # Time access and conversions

# Import Can, CanError, CanMsg, and CanMsgFlag classes from the custom
library 'canbus'
from canbus import Can, CanError, CanMsg, CanMsgFlag

# Create an instance of the Can class to interface with the CAN bus
can = Can()

# Initialize the CAN interface
ret = can.begin()  # Begin method initializes the CAN interface and returns
a status code
if ret != CanError.ERROR_OK:  # Check if the initialization was successful
    print("Error to initialize CAN!")  # Print error message if initializa-
tion failed
```

```
    sys.exit(1)  # Exit the script with an error code if CAN interface
couldn't be initialized
print("Initialized successfully!")  # Print success message if initializati-
on was successful

# Main loop to send data to the CAN bus
while True:
    data = b"\x12\x34\x56\x78\x9A\xBC\xDE\xF0"  # Data to be sent over the
CAN bus in bytes format

    # Create a standard format frame CAN message
    msg = CanMsg(can_id=0x123, data=data)  # can_id is the identifier for
the CAN message, data is the bytes to send
    error = can.send(msg)  # Send the CAN message and store the error status
    if error == CanError.ERROR_OK:  # Check if the message was sent success-
fully
        print('1------------------------------')  # Print a delimiter if the
message was sent successfully

    # Create an extended format frame CAN message
    msg = CanMsg(can_id=0x12345678, data=data, flags=CanMsgFlag.EFF)  # EFF
flag indicates an extended frame format
    error = can.send(msg)  # Send the CAN message and store the error status
    if error == CanError.ERROR_OK:  # Check if the message was sent success-
fully
        print('2------------------------------')  # Print a delimiter if the
message was sent successfully

    time.sleep(1)  # Wait for 1 second before sending the next set of messa-
ges
```

Esempio di codice del ricevitore dalla libreria:

```
'''
Comments reworked by Joy-IT, 25.03.2024
------------------------------------------
A simple example to receive data from a CAN bus.
'''

# Import necessary libraries
import sys  # System-specific parameters and functions
import time  # Time access and conversions

# Import the Can and CanError classes from the custom library/module
'canbus'
from canbus import Can, CanError

# Create a Can object instance for interfacing with the CAN bus
can = Can()

# Initialize the CAN interface
ret = can.begin()  # Begin method initializes the CAN interface and returns
a status code
if ret != CanError.ERROR_OK:  # Check if the initialization was successful
    print("Error to initialize CAN!")  # Print error message if initializa-
tion failed
```

```
    sys.exit(1)  # Exit the script with an error code if CAN interface
couldn't be initialized
print("Initialized successfully!")  # Print success message if initializati-
on was successful

# Main loop to receive data from the CAN bus
while True:
    if can.checkReceive():  # Check if there is data to receive on the CAN
bus
        error, msg = can.recv()  # Receive data from the CAN bus; returns an
error code and the message object
        if error == CanError.ERROR_OK:  # Check if data was received without
errors
            # Print received message details
            print('------------------------------')
            print("can id: %#x" % msg.can_id)  # Print the CAN ID in hexade-
cimal format
            print("is rtr frame:", msg.is_remote_frame)  # Print whether the
message is a Remote Transmission Request (RTR) frame
            print("is eff frame:", msg.is_extended_id)  # Print whether the
message uses an Extended Frame Format (EFF)
            print("can data hex:", msg.data.hex())  # Print the message data
in hexadecimal format
            print("can data dlc:", msg.dlc)  # Print the Data Length Code
(DLC), indicating the number of data bytes in the message
    else:
        time.sleep(0.1)  # If no data to receive, wait for 0.1 seconds befo-
re checking again
```

7. INFORMAZIONI AGGIUNTIVE

I nostri obblighi di informazione e ritiro ai sensi della legge sulle apparecchiature elettriche ed elettroniche (ElektroG)

Simbolo sulle apparecchiature elettriche ed elettroniche:

Questa pattumiera barrata significa che gli apparecchi elettrici ed elettronici non vanno gettati nei rifiuti domestici. È necessario restituire i vecchi apparecchi a un punto di raccolta.

Prima di consegnare i rifiuti, le batterie e gli accumulatori che non sono racchiusi nelle apparecchiature devono essere separati da queste ultime.

Opzioni di restituzione:

L'utente finale può restituire gratuitamente il vecchio apparecchio (che svolge essenzialmente la stessa funzione del nuovo apparecchio acquistato da noi) per smaltirlo al momento dell'acquisto di un nuovo apparecchio.

I piccoli apparecchi con dimensioni esterne non superiori a 25 cm possono essere smaltiti nelle normali quantità domestiche indipendentemente dall'acquisto di un nuovo apparecchio.

Possibilità di restituzione presso la nostra sede negli orari di apertura:

SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn, Germania

Possibilità di restituzione nella vostra zona:

Vi invieremo un francobollo con il quale potrete restituirci gratuitamente il dispositivo. Contattateci via e-mail all'indirizzo Service@joy-it.net o per telefono.

Informazioni sull'imballaggio:

Se non disponete di materiale d'imballaggio adatto o non desiderate utilizzare il vostro, contattateci e vi invieremo l'imballaggio adatto.

8. SOSTEGNO

Se dopo l'acquisto ci sono ancora questioni in sospeso o problemi, vi supporteremo via e-mail, telefono e con il nostro sistema di assistenza tramite ticket.

Email: **service@joy-it.net**
Sistema di ticket: **http://support.joy-it.net**
Telefono: +49 (0)2845 9360-50 (lun-gio: 09:00 - 17:00,
venerdì: dalle 09:00 alle 14:30)

Per ulteriori informazioni, visitate il nostro sito web:
www.joy-it.net

Pubblicato: 27.03.2024