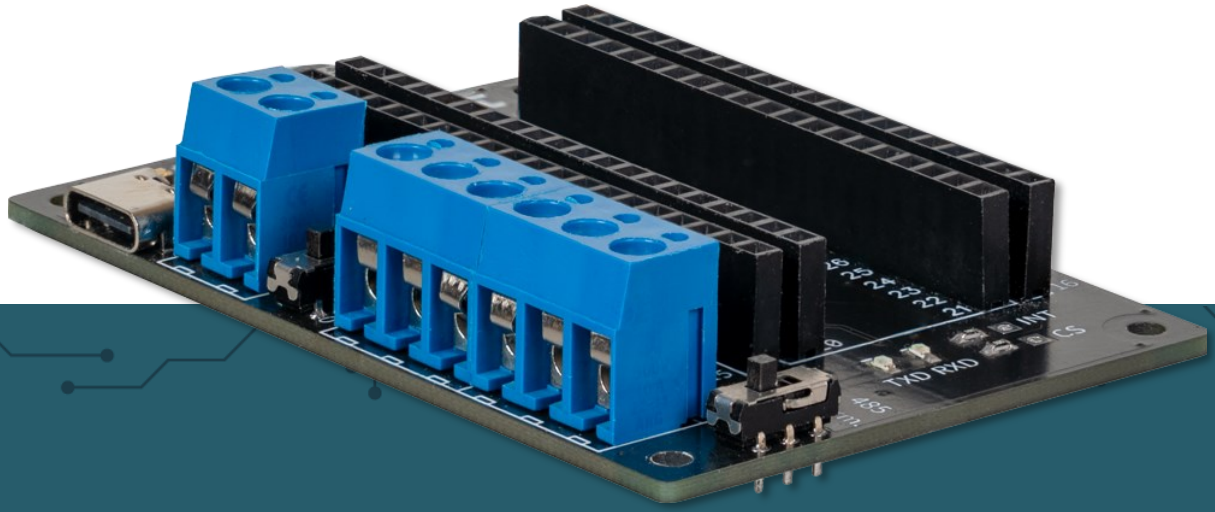


RB-P-CAN-485

Placa de expansão para Raspberry Pi Pico

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

Atenção! Este manual foi traduzido automaticamente, em caso de dúvida, consulte o manual em inglês ou contacte o nosso serviço de apoio ao cliente.
Todos os pedidos de apoio devem ser efectuados em alemão ou inglês.

1. INFORMAÇÕES GERAIS

Caro cliente,
Muito obrigado por ter escolhido o nosso produto.
De seguida, apresentamos-lhe o que deve observar durante a colocação em funcionamento e a utilização deste produto.

Se tiver algum problema inesperado durante a utilização, não hesite em contactar-nos.

2. EXPLICAÇÃO DO MÓDULO

Nesta secção, explicaremos brevemente as funções individuais da placa de expansão à qual pode ligar o seu Pico.

LED de alimentação: Indica quando a tensão de 5V está presente, quer a partir da porta USB, quer a partir do conversor de tensão interno da placa

USB (ligação USB-C):
5V Fonte de alimentação.

Entrada: alimentação variável 6 - 12V alimentação.

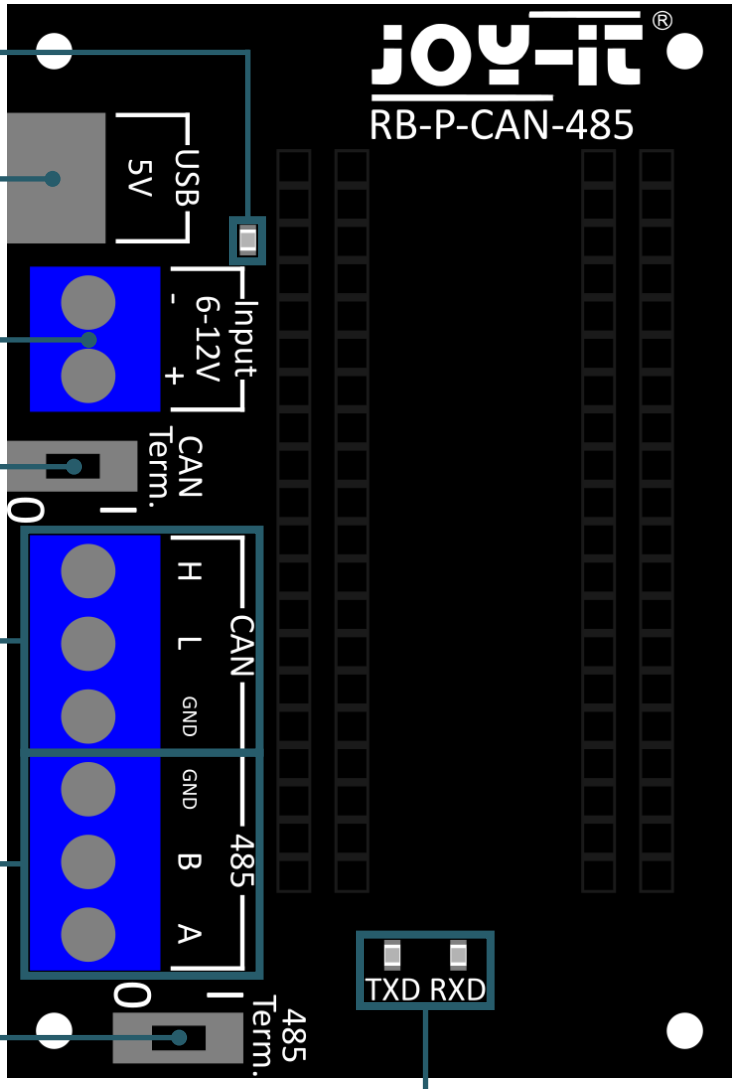
CAN Term. (Terminação CAN) : Comuta uma resistência de 120Ω para a posição "I", que é utilizada para a terminação especializada do bus.

CAN :
H : Ligação à linha CAN-High.
L : Ligação à linha CAN-Low.
GND : Ligação à terra para a equalização do potencial entre os nós.

485 :
A : Ligação à linha de dados negativos.
B : Ligação à linha de dados positivos.
GND : Ligação à terra para equalização do potencial.

485 Term. (Terminação 485) :
Comuta uma resistência de 120Ω para a posição "I", que é utilizada para a terminação especializada do bus.

TXD | RXD (LEDs de estado 485):
Apresenta o estado atual na linha de transmissão e receção.
Um sinal alto faz com que os LEDs se acendam.



3. RASPBERRY PI PICO

Agora, ligue um cabo micro USB ao seu computador e ao Raspberry Pi Pico para programação.

ATENÇÃO: A ligação USB-C na placa é utilizada apenas para a alimentação eléctrica. Nenhum dado é transferido para o Raspberry Pi Pico.

Pode utilizar um programa de desenvolvimento adequado à sua escolha para transferir os programas de amostra. Recomendamos o **Thonny IDE**.

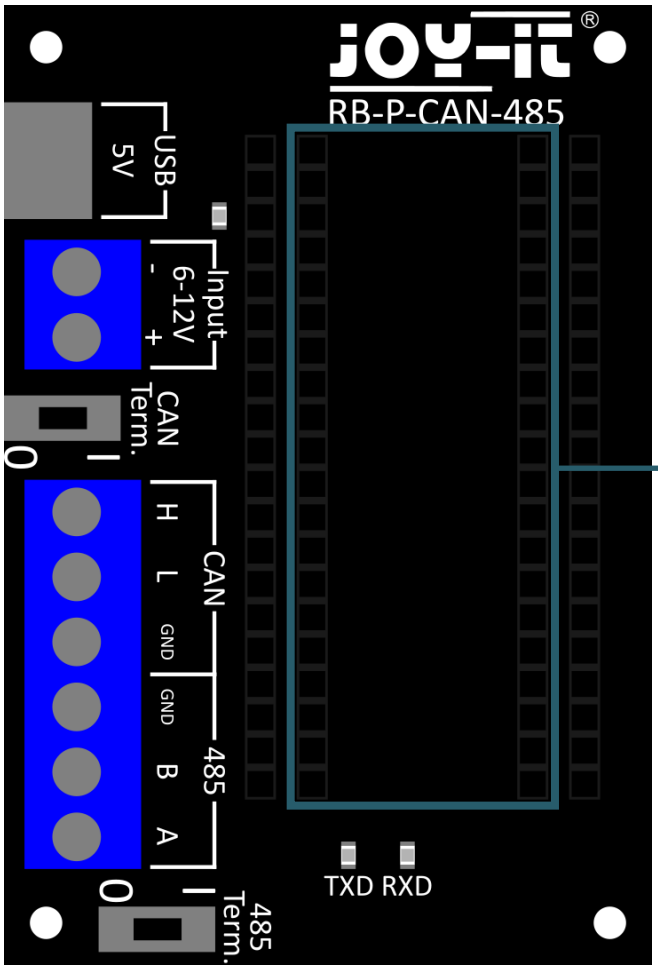
ATENÇÃO: Se é novo no mundo dos microcontroladores e da eletrónica, não se preocupe! Preparámos para si um guia especial para principiantes. Este guia foi especialmente concebido para as necessidades dos principiantes e explica como utilizar o Raspberry Pi Pico passo a passo.

Desde a configuração básica até à execução de projectos - neste guia acompanhamo-lo ao longo de todo o processo. O nosso guia inclui explicações fáceis de compreender e dicas úteis para o ajudar a desenvolver as suas competências à escala com o Raspberry Pi Pico de forma rápida e eficaz. Pode descarregar o nosso guia **aqui**.

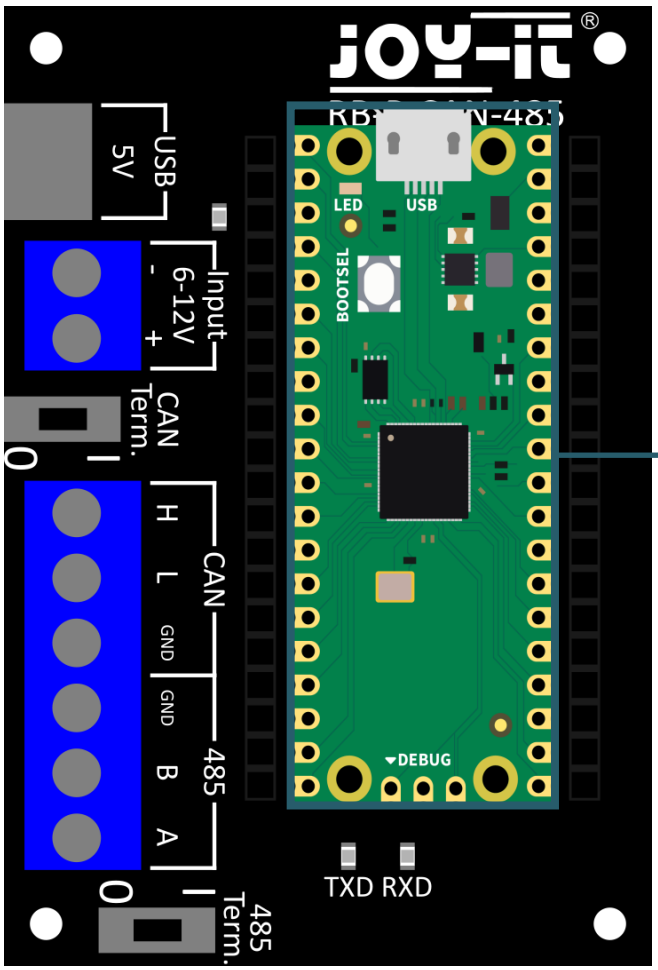
4. LIGAÇÃO DO RASPBERRY PI PICO

Na secção seguinte, veremos como ligar corretamente o Raspberry Pi Pico à placa de expansão.

Abaixo pode ver uma ilustração de como o Raspberry Pi Pico deve ser ligado à placa de expansão.



Ranhura para o Raspberry Pi Pico:
O Raspberry Pi Pico é ligado aqui.



Se o Raspberry Pi Pico tiver sido ligado, a ranhura deve ter o seguinte aspeto.

5. EXEMPLO DE CÓDIGO RS485

Agora que já configurou o seu Raspberry Pi Pico, pode carregar os dois exemplos de código seguintes para comunicação através da interface RS485 para o seu Raspberry Pi Pico. Em alternativa, pode também descarregar os exemplos de código **aqui**.

Exemplo de código do transmissor:

```
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))
```

Exemplo de código do recetor:

```
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. EXEMPLO DE CÓDIGO PODE

Agora que já configurou o seu Raspberry Pi Pico, pode carregar os dois exemplos de código seguintes para comunicação através da interface CAN para o seu Raspberry Pi Pico.

Para utilizar estes exemplos, é necessária a biblioteca correspondente. Utilizamos a biblioteca **MicroPython_CAN_BUS_MCP2515** da **Longan-Labs**, que é publicada sob a **licença MIT**.

Exemplo de código do transmissor da biblioteca:

```
'''
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
```

Exemplo de código do recetor da biblioteca:

```
'''
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. INFORMAÇÕES ADICIONAIS

As nossas obrigações de informação e de retoma de acordo com a Lei dos Equipamentos Eléctricos e Electrónicos (ElektroG)

Símbolo nos equipamentos eléctricos e electrónicos:

Este caixote do lixo barrado com uma cruz significa que os aparelhos eléctricos e electrónicos não devem ser colocados no lixo doméstico. Deve entregar os aparelhos velhos num ponto de recolha.

Antes de entregar os resíduos, as pilhas e os acumuladores que não estejam incluídos nos resíduos de equipamentos devem ser separados dos mesmos.

Opções de devolução:

Como utilizador final, pode devolver gratuitamente o seu aparelho antigo (que desempenha essencialmente a mesma função que o novo aparelho que nos foi comprado) para ser eliminado quando comprar um novo aparelho.

Os pequenos aparelhos sem dimensões exteriores superiores a 25 cm podem ser eliminados em quantidades domésticas normais, independentemente da compra de um novo aparelho.

Possibilidade de devolução nas instalações da nossa empresa durante o horário de funcionamento:

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

Possibilidade de devolução na sua área:

Enviar-lhe-emos um selo de encomenda com o qual nos poderá devolver o aparelho gratuitamente. Por favor, contacte-nos por e-mail para Service@joy-it.net ou por telefone.

Informações sobre a embalagem:

Se não dispuser de material de embalagem adequado ou não pretender utilizar o seu próprio material, contacte-nos e enviar-lhe-emos uma embalagem adequada.

8. APOIO

Se ainda houver questões pendentes ou problemas que surjam após a sua compra, apoiá-lo-emos por correio eletrónico, telefone e através do nosso sistema de apoio através de bilhetes.

Email: **service@joy-it.net**
Sistema de bilhetes: **http://support.joy-it.net**
Telefone: +49 (0)2845 9360-50 (Seg - Qui: 09:00 - 17:00 horas,
Sexta-feira: 09:00 - 14:30 horas)

Para mais informações, visite o nosso sítio Web:

www.joy-it.net

Publicado: 27.03.2024