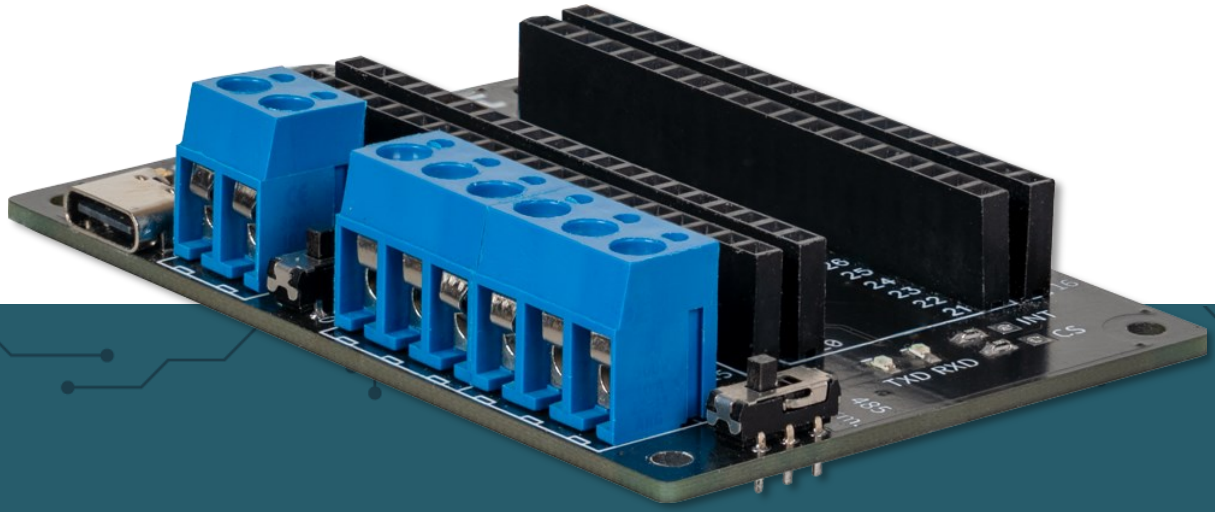


RB-P-CAN-485

Placa de expansión para Raspberry Pi Pico

Atención! Este manual se ha traducido automáticamente. En caso de duda, consulte el manual en inglés o póngase en contacto con nuestro servicio de atención al cliente.
Todas las solicitudes de asistencia deben realizarse en alemán o inglés.

1. INFORMACIÓN GENERAL

Estimado cliente,
muchas gracias por elegir nuestro producto.
A continuación, le indicaremos qué debe tener en cuenta durante la puesta en marcha y el uso de este producto.

En caso de que surja algún problema inesperado durante el uso, no dude en ponerse en contacto con nosotros.

2. EXPLICACIÓN DEL MÓDULO

En esta sección, explicaremos brevemente las funciones individuales de la tarjeta de expansión a la que puede conectar su Pico.

LED de encendido: Indica cuando hay tensión de 5V, ya sea desde el puerto USB o desde el convertidor de tensión interno de la placa.

USB (conexión USB-C):
5V Fuente de alimentación.

Entrada : 6 - 12V alimentación variable alimentación.

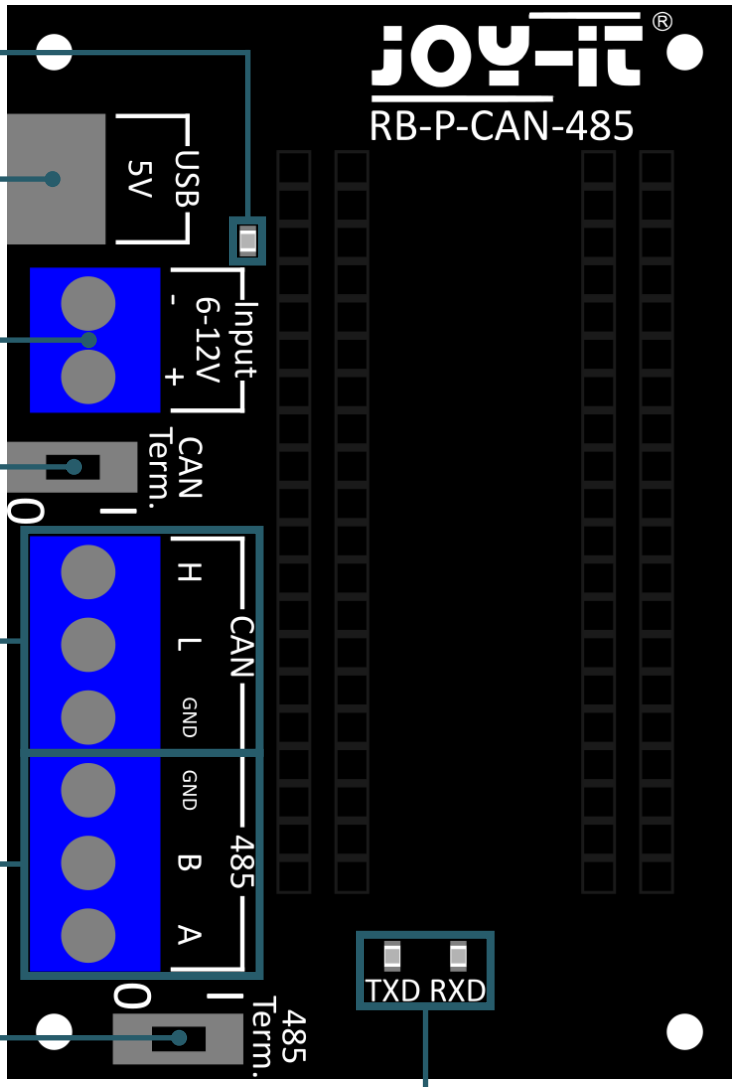
CAN Term. (Terminación CAN) :
Conmuta una resistencia de 120Ω a la posición "I", que se utiliza para la terminación especializada del bus.

CAN :
H : Conexión a la línea CAN-High.
L : Conexión a la línea CAN-Low.
GND : Conexión a tierra para igualación de potencial entre nodos.

485 :
A : Conexión a la línea de datos negat va.
B : Conexión a la línea de datos positiva.
GND : Conexión a tierra para ecualización de potencial.

485 Term. (Terminación 485) :
Conmuta una resistencia de 120Ω a la posición "I", que se utiliza para la terminación especializada del bus.

TXD | RXD (LEDs de estado 485):
Muestra el estado actual en la línea de Transmisión y Recepción.
Una señal alta hace que se enciendan los LEDs.



3. RASPBERRY PI PICO

Ahora conecta un cable micro USB a tu ordenador y a la Raspberry Pi Pico para programarla.

ATENCIÓN: La conexión USB-C de la placa solo se utiliza para la alimentación. No se transfieren datos a la Raspberry Pi Pico.

Puede utilizar un programa de desarrollo adecuado de su elección para transferir los programas de ejemplo. Recomendamos el **Thonny IDE**.

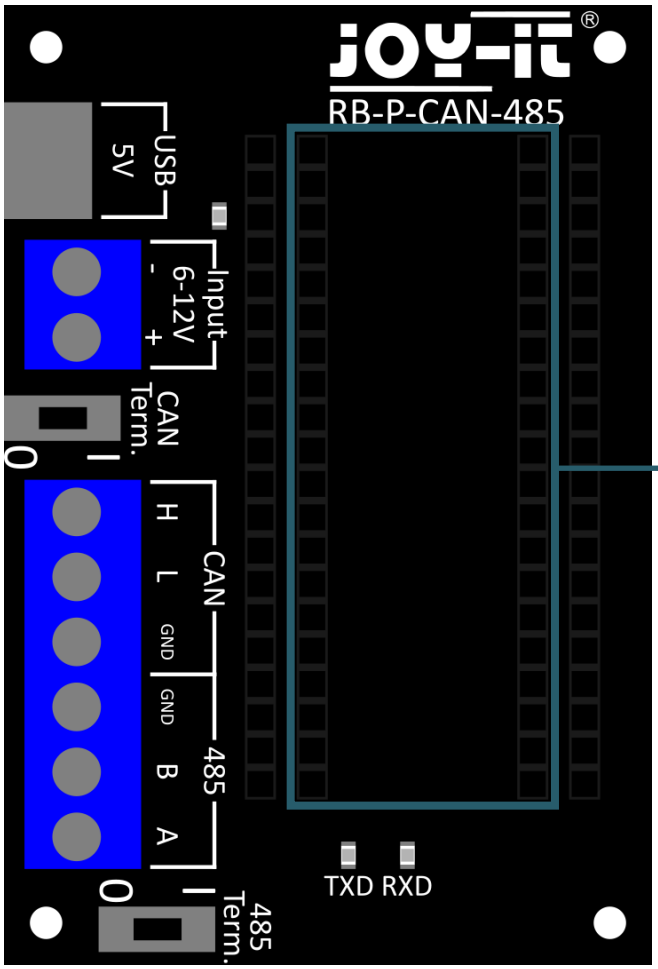
ATENCIÓN: Si eres nuevo en el mundo de los microcontroladores y la electrónica, ¡no te preocupes! Hemos preparado una guía especial para principiantes. Esta guía está especialmente adaptada a las necesidades de los principiantes y explica cómo utilizar la Raspberry Pi Pico paso a paso.

Desde la configuración básica hasta la ejecución de proyectos - en esta guía te guiamos a través de todo el proceso. Nuestra guía incluye explicaciones fáciles de entender y consejos útiles para ayudarle a desarrollar sus habilidades a escala con la Raspberry Pi Pico de forma rápida y eficaz. Puede descargar nuestra guía **aquí**.

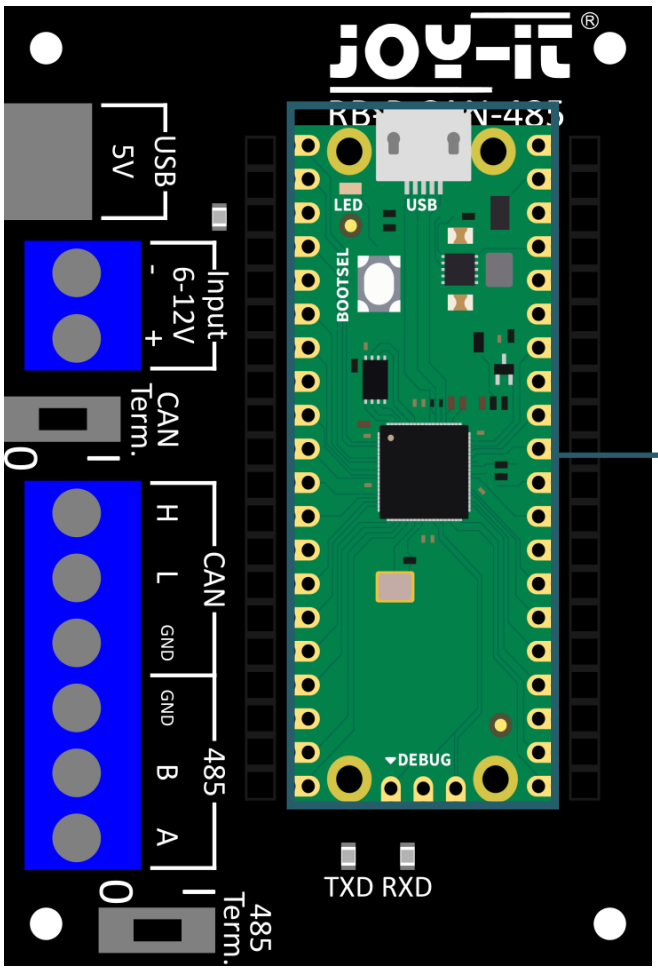
4. CONEXIÓN DE LA RASPBERRY PI PICO

En la siguiente sección, veremos cómo conectar correctamente la Raspberry Pi Pico a la placa de expansión.

A continuación puedes ver una ilustración de cómo debe conectarse la Raspberry Pi Pico a la placa de expansión.



Ranura para la Raspberry Pi Pico:
La Raspberry Pi Pico se conecta aquí.



Si la Raspberry Pi Pico ha sido conectada, tu ranura debería tener ahora este aspecto.

5. EJEMPLO DE CÓDIGO RS485

Ahora que ha configurado su Raspberry Pi Pico, puede cargar los siguientes dos ejemplos de código para la comunicación a través de la interfaz RS485 a su Raspberry Pi Pico. Alternativamente, también puede descargar los ejemplos de código **aquí**.

Ejemplo de código del transmisor:

```
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))
```

Ejemplo de código del receptor:

```
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. EJEMPLO DE CÓDIGO PUEDE

Ahora que ha configurado su Raspberry Pi Pico, puede cargar los siguientes dos ejemplos de código para la comunicación a través de la interfaz CAN a su Raspberry Pi Pico.

La biblioteca correspondiente es necesaria para utilizar estos ejemplos. Utilizamos la biblioteca **MicroPython_CAN_BUS_MCP2515** de **Longan-Labs**, que se publica bajo la **licencia MIT**.

Ejemplo de código del transmisor de la 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
```

Ejemplo de código del receptor de la 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. INFORMACIÓN ADICIONAL

Nuestras obligaciones de información y recuperación según la Ley de aparatos eléctricos y electrónicos (ElektroG)

Símbolo en aparatos eléctricos y electrónicos:

Este cubo de basura tachado significa que los aparatos eléctricos y electrónicos no deben tirarse a la basura doméstica. Debe devolver los aparatos viejos a un punto de recogida.

Antes de entregar los residuos hay que separar las pilas y acumuladores que no estén incluidos en los residuos de aparatos.

Opciones de devolución:

Como usuario final, puede devolver gratuitamente su aparato antiguo (que cumple esencialmente la misma función que el aparato nuevo que nos ha comprado) para que lo eliminemos al comprar un aparato nuevo. Los aparatos pequeños sin dimensiones externas superiores a 25 cm pueden desecharse en cantidades domésticas normales independientemente de la compra de un aparato nuevo.

Posibilidad de devolución en la sede de nuestra empresa en horario de apertura:

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

Posibilidad de devolución en su zona:

Le enviaremos un sello de paquetería con el que podrá devolvernos el aparato de forma gratuita. Póngase en contacto con nosotros por correo electrónico en Service@joy-it.net o por teléfono.

Información sobre el embalaje:

Si no dispone de material de embalaje adecuado o no desea utilizar el suyo propio, póngase en contacto con nosotros y le enviaremos un embalaje adecuado.

8. APOYO

Si después de la compra queda algún asunto pendiente o surge algún problema, le ayudaremos por correo electrónico, por teléfono y con nuestro sistema de tickets de asistencia.

Email: **service@joy-it.net**
Sistema de tickets: **http://support.joy-it.net**
Teléfono: +49 (0)2845 9360-50 (Lun - Jue: 09:00 - 17:00 horas,
viernes: 09:00 - 14:30 horas)

Para más información, visite nuestro sitio web:
www.joy-it.net

Publicado en: 27.03.2024