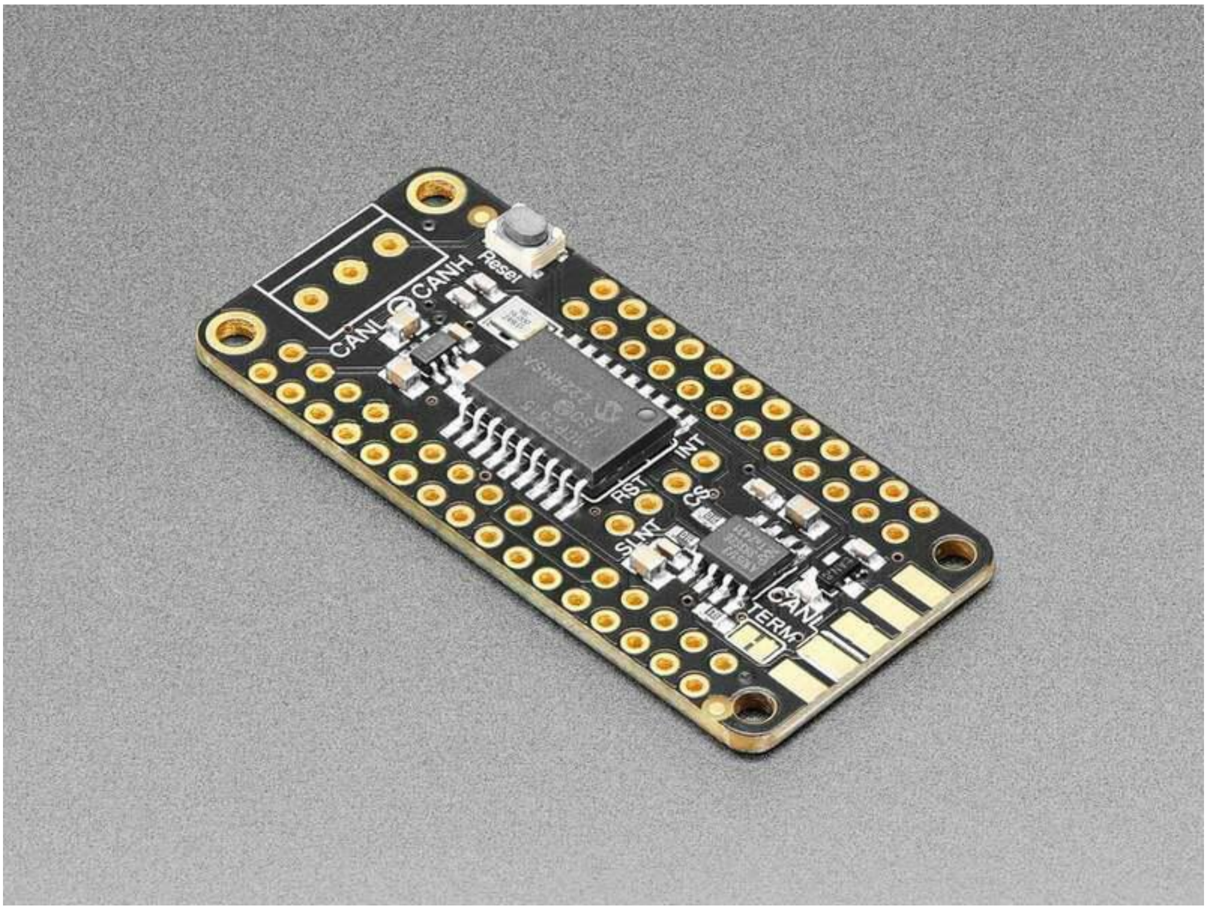
Adafruit CAN Bus FeatherWing

Created by Liz Clark

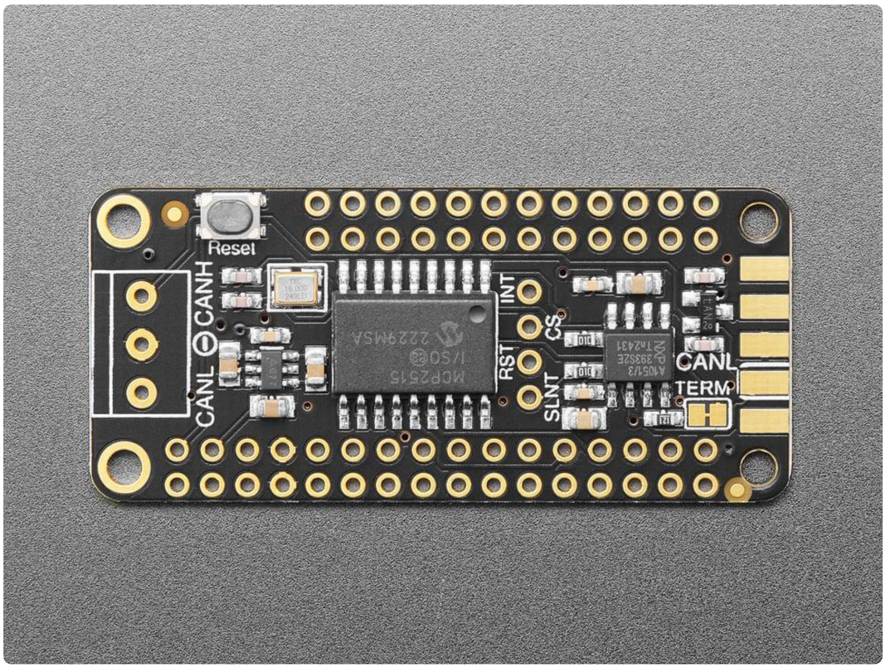


https://learn.adafruit.com/adafruit-can-bus-featherwing

Last updated on 2023-03-29 09:28:47 AM EDT

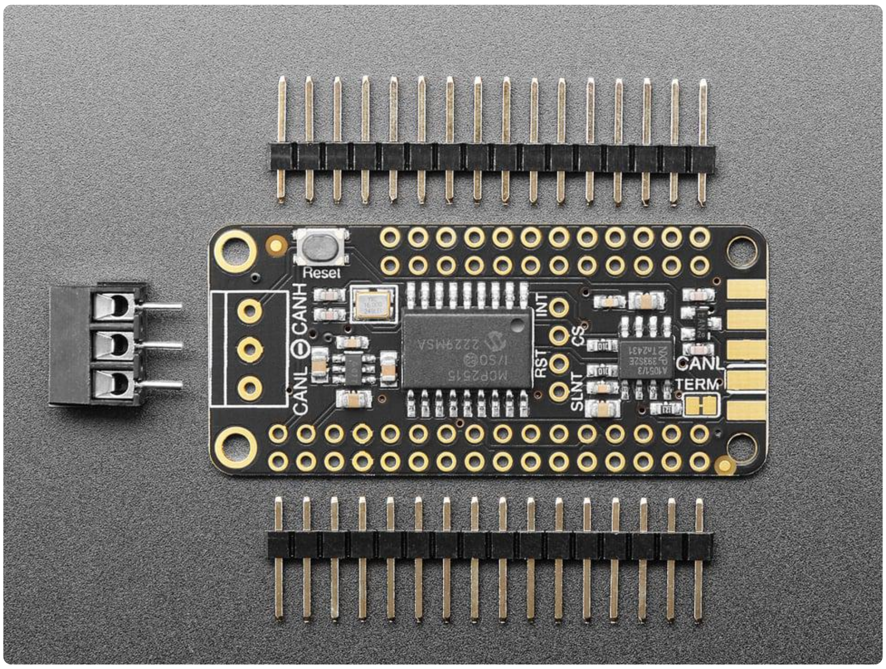
Table of Contents

Overview

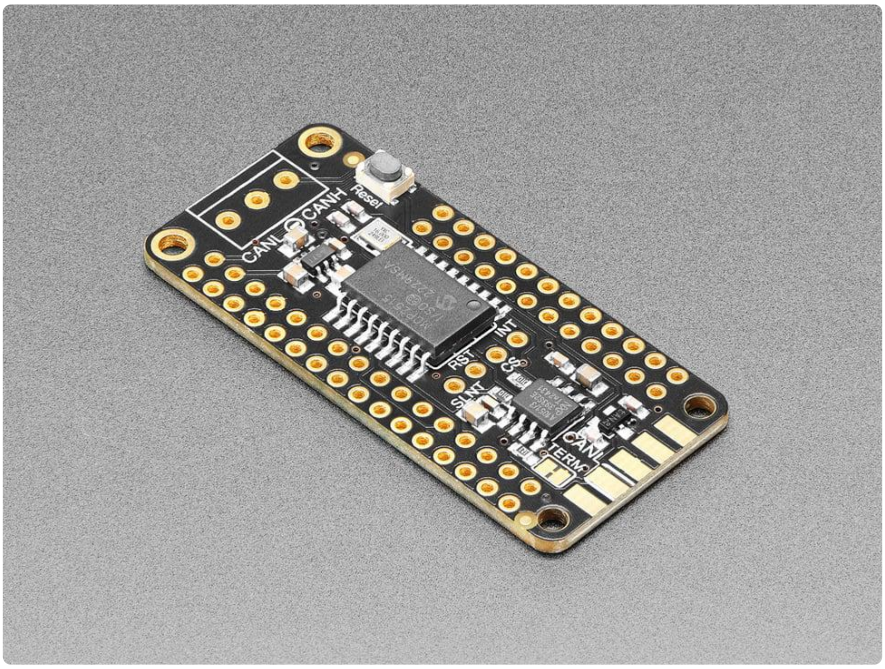


CAN Bus is a small-scale networking standard (), originally designed for cars and, yes, busses, but is now used for many robotics or sensor networks that need better range and addressing than I2C, and don't have the pins or computational ability to talk on Ethernet. CAN is 2 wire differential, which means it's good for long distances and noisy environments.

Messages are sent at about 1Mbps rate - you set the frequency for the bus and then all 'joiners' must match it, and have an address before the packet so that each node can listen in to messages just for it. New nodes can be attached easily because they just need to connect to the two data lines anywhere in the shared net. Each CAN devices sends messages whenever it wants, and thanks to some clever data encoding, can detect if there's a message collision and retransmit later.



If you'd like to connect your Feather to a CAN Bus, the Adafruit CAN Bus FeatherWing with MCP2515 controller and TJA1051/3 transceiver will work with any and all Feathers! The controller used is the MCP2515, an extremely popular and well-supported chipset () that has drivers in Arduino and CircuitPython () and only requires an SPI port and two pins for chip-select and IRQ. Use it to send and receive messages in either standard or extended format at up to 1 Mbps.

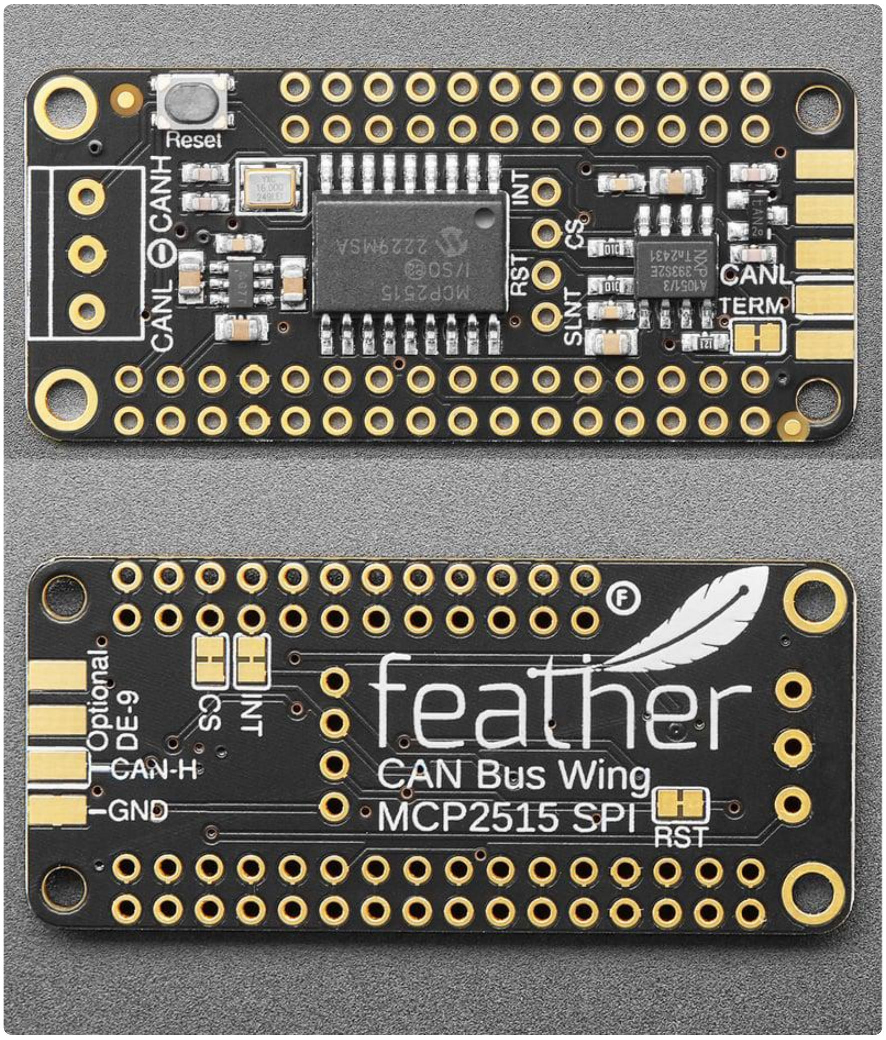


We've added a few nice extras to this Feather to make it useful in many common CAN scenarios:

- 5V charge-pump voltage generator (), so even though you are running 3.3V on a Feather board, it will generate a nice clean 5V as required by the transceiver.
- 3.5mm terminal block () that can be soldered in to get quick access to the High and Low data lines as well as a ground pin.
- 120 ohm termination resistor on board, you can remove the termination easily by cutting the jumper marked TERM on the top of the board.
- A spot that one can solder an optional (not included!) edge-launch DE-9 connector () that is commonly used for connecting to CAN devices. L is connected to pin 2 and H is connected to pin 7. You could then plug this into an ODB-II cable to connect to a car's CAN network () - you'll still need software to decode the messages though!
- Pre-connected CS and INT pins on the two pins to the left of the I2C Feather port - usually these are pins #5 and #6 but some feathers may have different numbering! You can cut the bottom solder jumpers and use the breakout pads to connect to any two IO pins you like.

Each order comes with an assembled 'Wing, terminal block and header. You will need to solder in the header yourself but it's a quick task.

Pinouts



Terminal Block Pins

On the front left of the board, are the three pins for the included terminal block. It is outlined in white on the silk.

- CANL - the CAN low signal for the CAN Bus standard.
- - - common ground shared between the two CAN connections
- CANH - the CAN high signal for the CAN Bus standard.

Both CANL and CANH are connected to a 5V charge-pump voltage generator (). Even if you are using 3.3V logic on a Feather board, it will generate a nice clean 5V as required by the CAN Bus transceiver.

Edge Launch DE-9 Connector Solder Pads

On the front-right end of the board are the solder pads for attaching an optional edge launch DE-9 connector (). This is commonly used for connecting to CAN devices. CAN L is connected to DE-9 pin 2 and CANH is connected to DE-9 pin 7. You could then plug this into an ODB-II cable to connect to a car's CAN network (). Then, you can decode messages with software.

Terminator Jumper

The terminator jumper is located on the front of the board, to the left of the solder pads for the edge launch DE-9 connector. It is labeled TERM and is outlined in white on the board silk.

- TERM - The board has a 120 ohm terminator connected between CANH and CANL. You can disable (remove) the terminator by cutting the TERM jumper, if your bus is already terminated.

Reset Pin and Jumper

- Reset Jumper - located on the back of the board, to the right of the board name description, is the reset jumper. It is labeled RST and is outlined in white on the board silk. You can cut this jumper to disconnect the MCP2515 reset pin from the Feather reset pin.
- Reset Pin - located on the front of the board in the center. It is labeled RST on the silk. If you cut the reset jumper, you can use this pin to connect the MCP2515 reset pin to a different IO pin.

Chip Select Pin and Jumper

- CS Jumper - located on the back of the board, below Feather pin 5, is the chip select jumper. It is labeled CS and is outlined in white on the board silk. By default, the MCP2515 chip select pin is connected to pin 5 on the Feather. You can cut this jumper to disconnect the MCP2515 CS pin from Feather pin 5.
- CS pin - located on the front of the board in the center. It is labeled CS on the silk. If you cut the CS jumper, you can use this pin to connect the MCP2515 CS pin to a different IO pin.

Interrupt Pin and Jumper

- INT Jumper - located on the back of the board, below Feather pin 6, is the interrupt jumper. It is labeled INT and is outlined in white on the board silk. By default, the MCP2515 interrupt pin is connected to pin 6 on the Feather. You can cut this jumper to disconnect the MCP2515 INT pin from Feather pin 6.
- INT pin - located on the front of the board in the center. It is labeled INT on the silk. If you cut the INT jumper, you can use this pin to connect the MCP2515 INT pin to a different IO pin.

Silent Mode Pin

- SLNT pin - located on the front of the board in the center. It is labeled SLNT on the silk. This pin can be pulled high to put the TJA1051/3 transceiver into silent mode. From the datasheet:

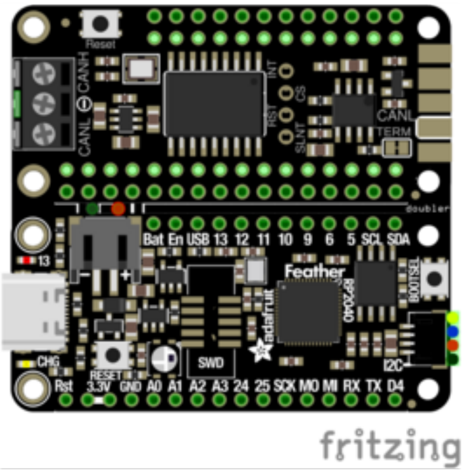
> In Silent mode, the transmitter is disabled, releasing the bus pins to recessive state. All other IC functions, including the receiver, continue to operate as in Normal mode. Silent mode can be used to prevent a faulty CAN controller from disrupting all network communications.

CircuitPython

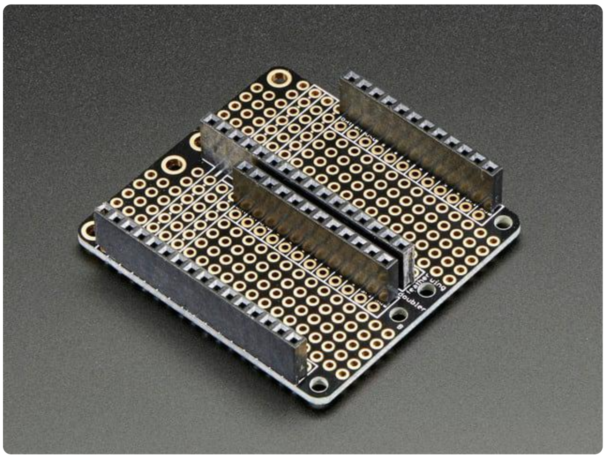
It's easy to use the CAN Bus FeatherWing with CircuitPython and the Adafruit_CircuitPython_MCP2515 () module. This module allows you to easily write Python code that lets you utilize the MCP2515 CAN bus controller.

CircuitPython Microcontroller Wiring

A Feather RP2040 can connect to the CAN Bus FeatherWings by plugging them both into a FeatherWing Doubler ():



Plug the Feather RP2040 into one slot on the doubler and then plug the CAN Bus FeatherWing into the remaining slot. This will connect all of the FeatherWing pins to the Feather RP2040.



FeatherWing Doubler - Prototyping Add-on For All Feather Boards

This is the FeatherWing Doubler - a prototyping add-on and more for all Feather boards. This is similar to our https://www.adafruit.com/product/2890

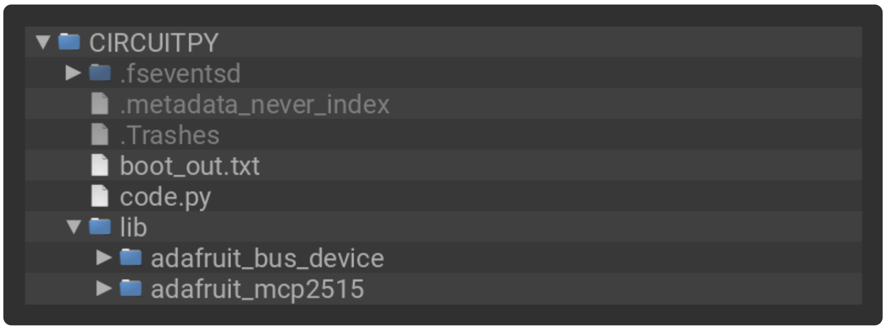
CircuitPython Usage

To use with CircuitPython, you need to first install the MCP2515 library, and its dependencies, into the lib folder onto your CIRCUITPY drive. Then you need to update code.py with the example script.

Thankfully, we can do this in one go. In the example below, click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the c ode.py file to your CIRCUITPY drive.

Your CIRCUITPY/lib folder should contain the following folders:

- adafruit_bus_device/
- adafruit_mcp2515/



Simple Test Example

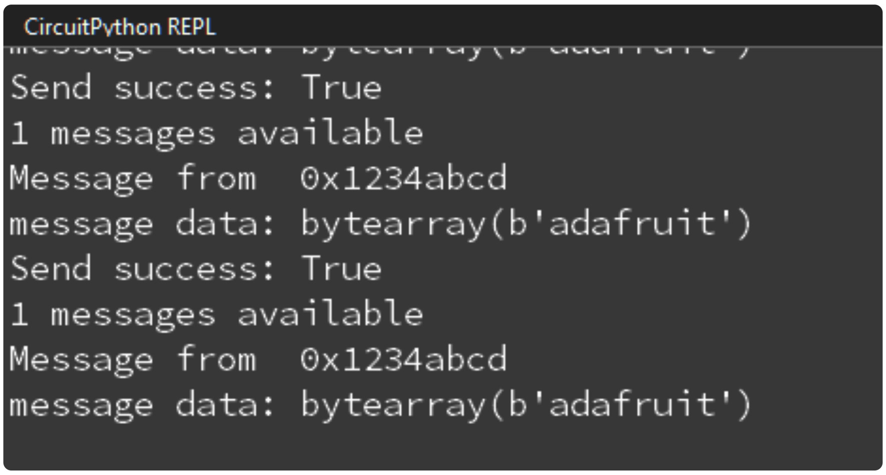
Once everything is saved to the CIRCUITPY drive, connect to the serial console () to see the data printed out!

```
# SPDX-FileCopyrightText: Copyright (c) 2020 Bryan Siepert for Adafruit Industries
#
# SPDX-License-Identifier: MIT
from time import sleep
import board
import busio
from digitalio import DigitalInOut
from adafruit_mcp2515.canio import Message, RemoteTransmissionRequest
from adafruit_mcp2515 import MCP2515 as CAN


cs = DigitalInOut(board.D5)
cs.switch_to_output()
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)

can_bus = CAN(
    spi, cs, loopback=True, silent=True
)  # use loopback to test without another device
while True:
    with can_bus.listen(timeout=1.0) as listener:

        message = Message(id=0x1234ABCD, data=b"adafruit", extended=True)
        send_success = can_bus.send(message)
        print("Send success:", send_success)
        message_count = listener.in_waiting()
        print(message_count, "messages available")
        for _i in range(message_count):
            msg = listener.receive()
            print("Message from ", hex(msg.id))
            if isinstance(msg, Message):
                print("message data:", msg.data)
            if isinstance(msg, RemoteTransmissionRequest):
                print("RTR length:", msg.length)
    sleep(1)
```

In the code, the instantiation of `can_bus` sets `loopback` and `silent` to `True`. This allows you to test the CAN Bus messaging without another CAN device.

```
can_bus = CAN(
    spi, cs, loopback=True, silent=True
)  # use loopback to test without another device
```

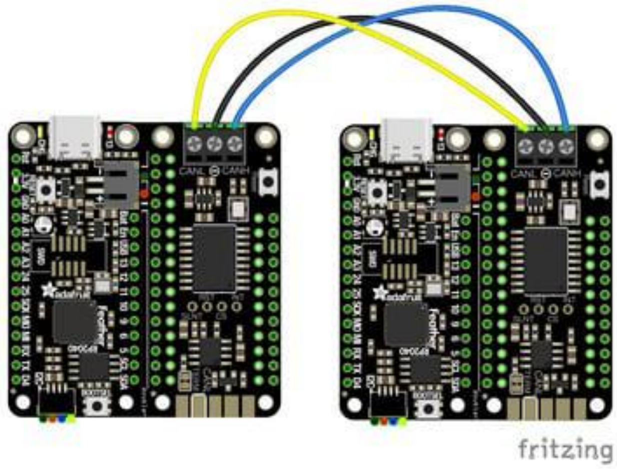
In the REPL, you'll see the byte array message be sent successfully every second.



```
message data: bytearray(b'adafruit')
Send success: True
1 messages available
Message from  0x1234abcd
message data: bytearray(b'adafruit')
Send success: True
1 messages available
Message from  0x1234abcd
message data: bytearray(b'adafruit')
```

Testing with Two CAN Bus FeatherWings

You can test with two CAN Bus FeatherWings by preparing two FeatherWing Doublers with Feather RP2040s and CAN Bus FeatherWings.



Once you have two Feather RP2040's connected to CAN Bus FeatherWings, you can connect the CAN signals:

FeatherWing A CANH terminal block to FeatherWing B CANH terminal block (blue wire)
FeatherWing A - (ground) terminal block to FeatherWing B - (ground) terminal block (black wire)
FeatherWing A CANL terminal block to FeatherWing B CANL terminal block (yellow wire)

Upload the Project Bundle to both CIRCUITPY drives. In the code, change the `can_bus` instantiation on both Feather RP2040's to have `loopback` and `silent` be `False`.

```
can_bus = CAN(
    spi, cs, loopback=False, silent=False
)  # use loopback to test without another device
```

When you run the code on both Feather boards, you can check each REPL window and see messages being exchanged between the two boards.

Python Docs

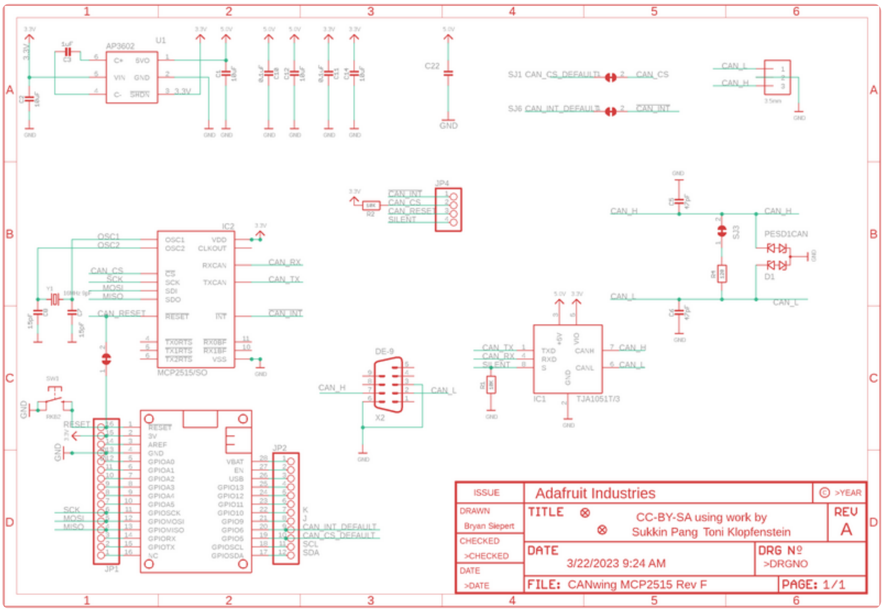
Python Docs ()

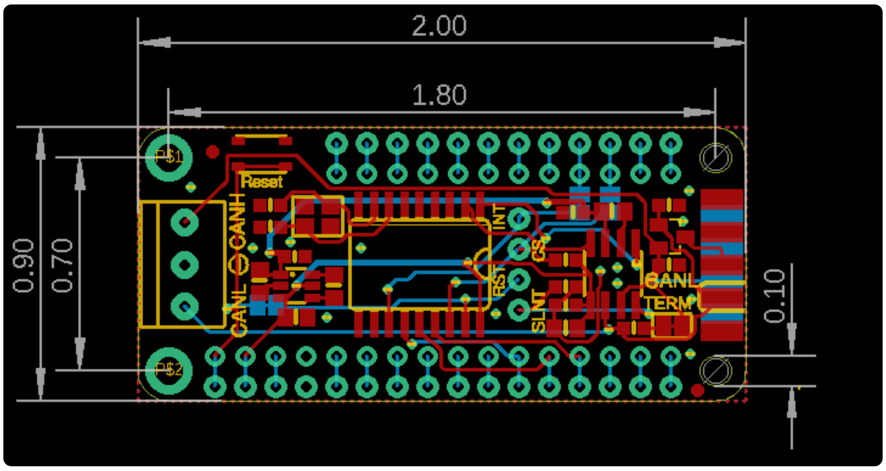
Downloads

Files

- MCP2515 Datasheet ()
- TJA1051/3 Datasheet ()
- EagleCAD PCB files on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()

Schematic and Fab Print





Dimensions are in inches.