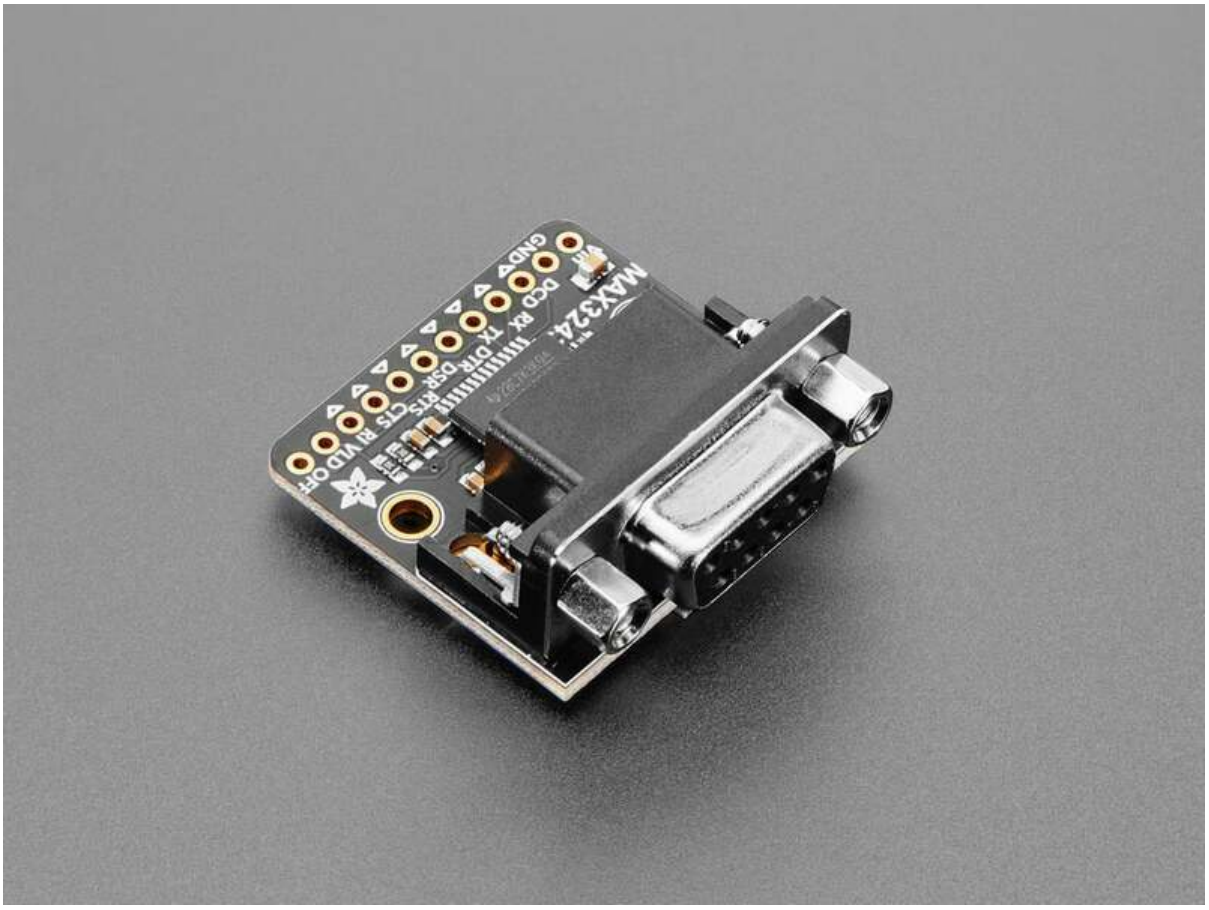
Adafruit RS-232 Full Pinout Level-Shifter Breakout

Created by Liz Clark

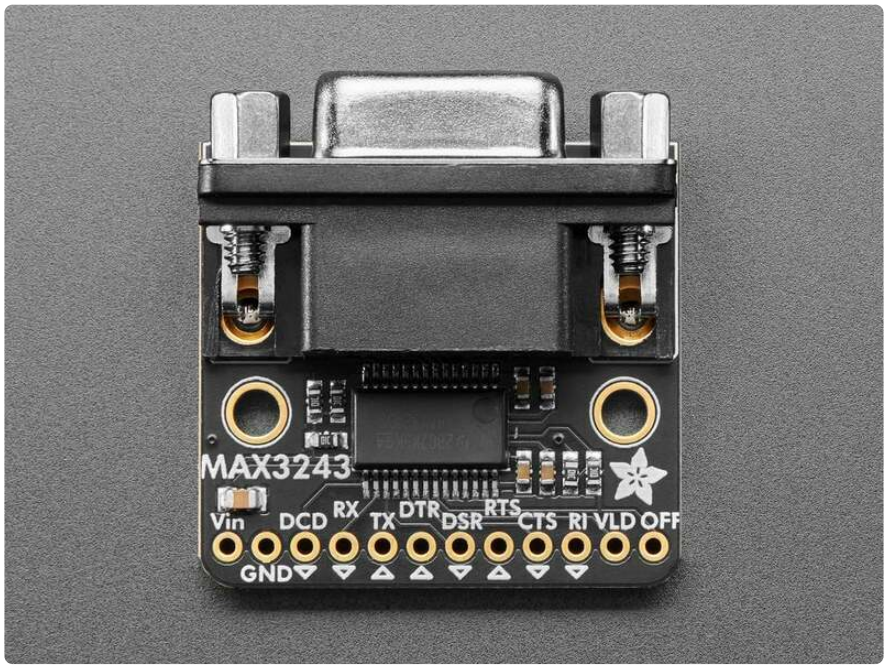


https://learn.adafruit.com/adafruit-rs-232-full-pinout-level-shifter-breakout

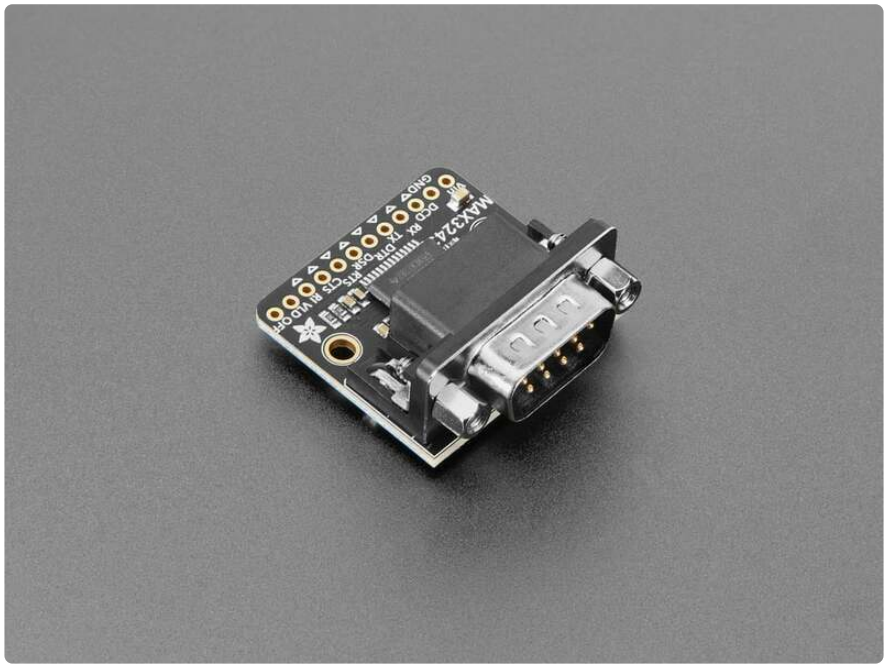
Last updated on 2025-03-06 11:11:14 AM EST

Table of Contents

Overview

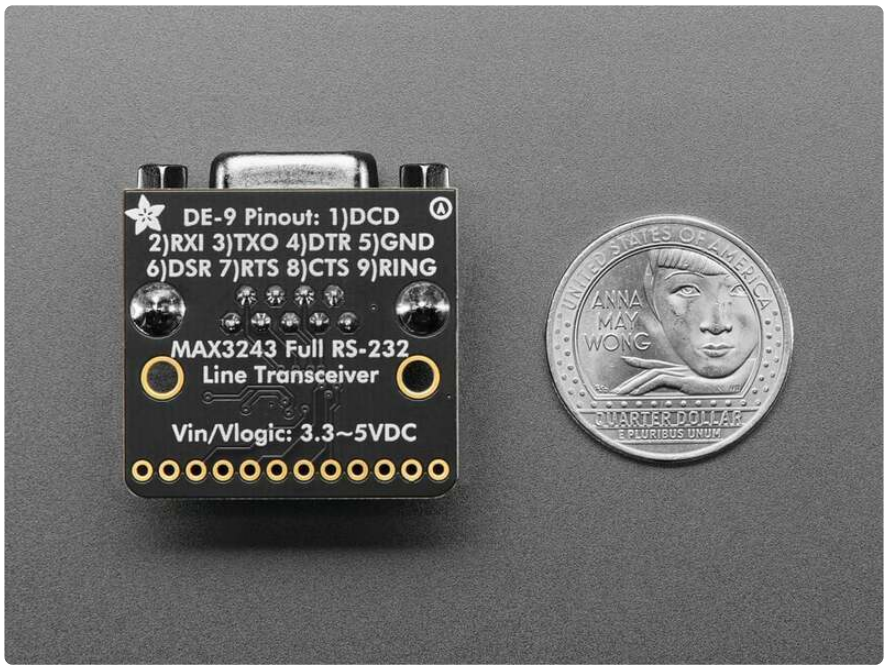


This breakout is available in two variants: one with a DE-9 Male connector and one with a DE-9 Female connector.

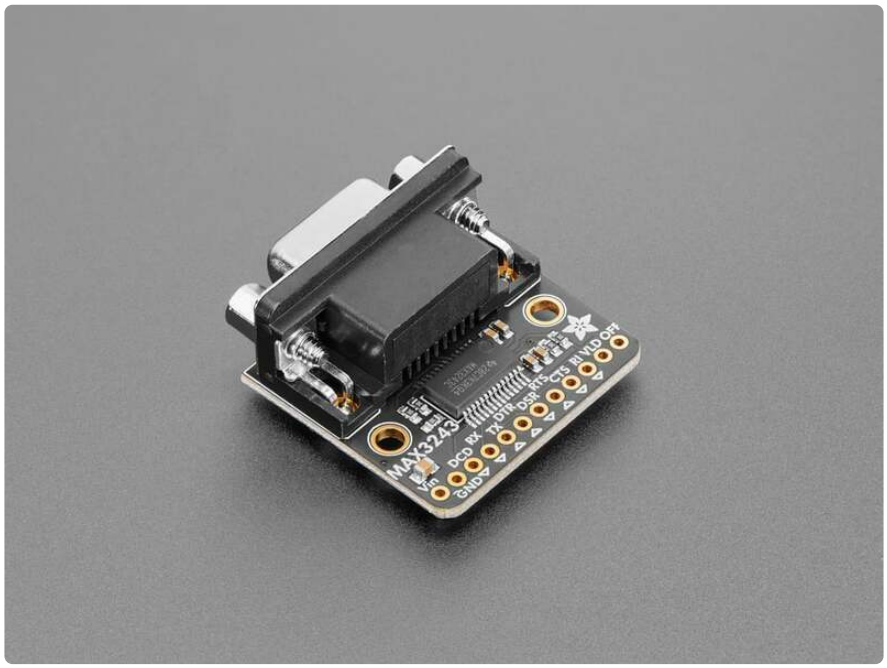


If you want to interface with telco, retro, or industrial equipment, you'll probably run into RS-232 interfaces. **The Adafruit RS232 Full-Pinout Breakout with 8 Channels of UART to RS-232 Level Shifters** is your friend in such cases. It gives you 5 input and 3

output channels of level shifting and takes care of the high/negative voltage generation all in a low-cost breakout board. We use the trusty MAX3243 (https://adafru.it/1a61) from TI, a classic chip part of the MAX232 lineage, so you know it will work great for all your RS-232 needs, up to 250Kbps.



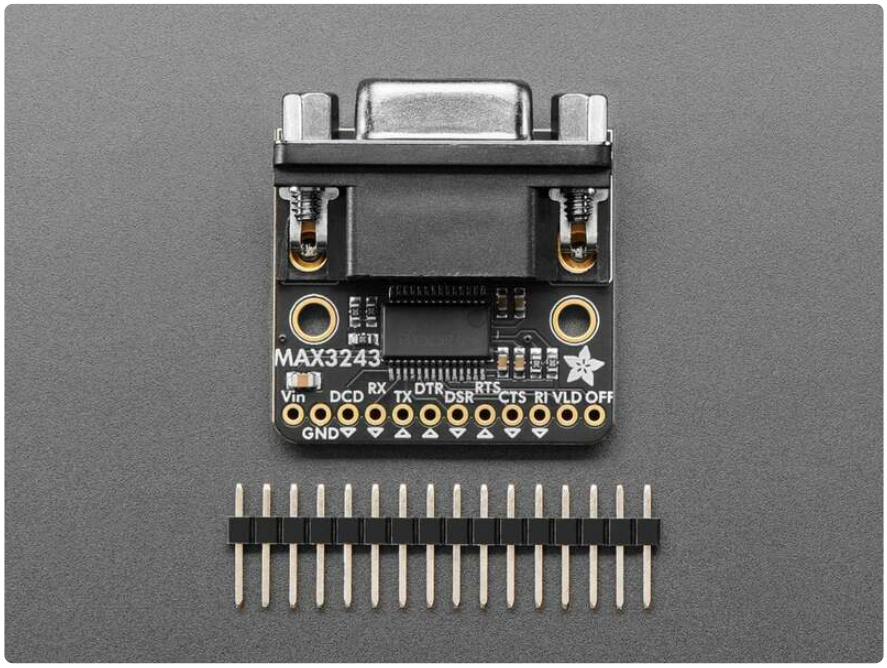
RS-232 is what we had before USB: a 9- or 25-pin D-Sub connector that allowed data plus flow control lines. Many folks may remember these interfaces were used for mice, modems, barcode scanners, teletypes, and more. We still find devices sold with RS-232 ports, although many folks use a USB to RS-232 adapter (https://adafru.it/1a45) these days.



If you want to use a microcontroller or microcomputer to chat with an RS-232, then thankfully, all you need is a serial port / UART (something just about any microcontroller has) and a level shifter. The level shifter is required because while most UARTs are 0-3.3V or 0-5V logic level, RS-232 requires +-6 to +-10V, yep the signal voltage goes negative! That means a specialized shifter is required to generate extra high and low voltages and safely convert the logic levels.

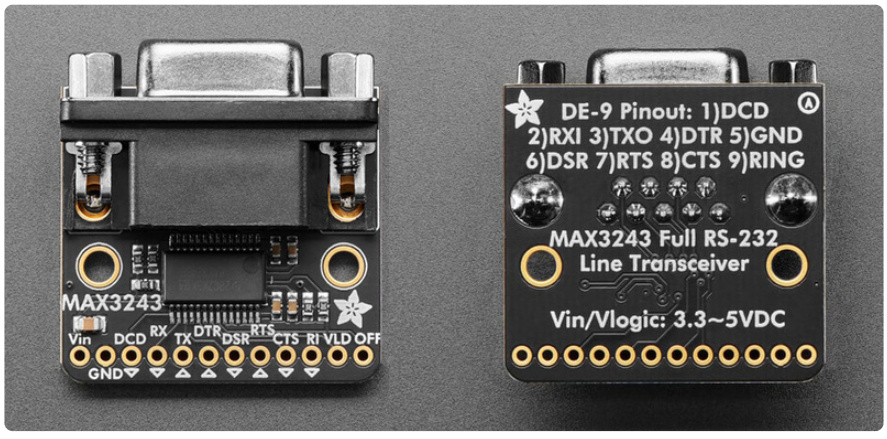
Sure you could buy a raw MAX232 chip (https://adafru.it/1a46) and wire up the necessary capacitors, but this board does it all for you and even comes with a DE-9 connector for plugging in directly into your 'client' device. It can run on 3.3V power and logic, which many older chips can't do. It also can do all 8 data pins, so you can use all of the flow control signals like RTS, CTS, DTR, DSR, DCD and RI.

We also include two separate lines from the MAX3243: 'Valid' and 'Off'. The **Valid** line output will have logic level high when the chip detects signal voltages from the device its plugged into. So you can use it as a 'connection made' signal. There's also the **Off** pin, which when set to logic high by the microcontroller will tri-state all the pins for power reduction.



This breakout comes fully assembled with a UART side for low-voltage power/logic level, and a DE-9F RS-232 side for high-voltage signals. We also include some header so you can solder to a breadboard in a few minutes.

Pinouts



Power Pins

- **Vin** - this is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino,

use 5V. The MAX3243 can generate the extra high and low voltages and also safely convert the logic levels needed for RS-232.

- **GND** - common ground for power and logic.

DE-9 Connector

At the edge of the board is the DE-9 connector (https://adafru.it/1a62). It is a 9-pin connector for interfacing directly with RS-232 equipment. Its pinout is as follows:

- Pin 1: **DCD**
- Pin 2: **RXI**
- Pin 3: **TXO**
- Pin 4: **DTR**
- Pin 5: **GND**
- Pin 6: **DSR**
- Pin 7: **RTS**
- Pin 8: **CTS**
- Pin 9: **RING**

UART Pins

- **RX** - The data input for receiving serial data from the RS-232 line **(pin 2 on the DE-9 connector)**
- **TX** - The data output for sending serial data to the RS-232 line **(pin 3 on the DE-9 connector)**

Flow Control Pins

- **DCD** - The data carrier detect pin. This output from the **DE-9 connector (pin 1)** is used to indicate whether a device is connected.
- **DTR** - The data terminal ready pin. This output from the **DE-9 connector (pin 4)** indicates that the device has data ready to transmit.
- **DSR** - The data set ready pin. This input to the **DE-9 connector (pin 6)** indicates that the connected microcontroller is ready to connect.
- **RTS** - The request to send pin. This output from the **DE-9 connector (pin 7)** indicates that the RS-232 device is ready to send data.
- **CTS** - The clear to send pin. This input to the **DE-9 connector (pin 8)** indicates that the RS-232 device can send data.
- **RI** - The ring indicator pin. This input to the **DE-9 connector (pin 9)** indicates that an incoming call signal has been received; usually with RS-232 modems.

Extra MAX3243 Pins

- **VLD** - The **Valid** pin is a line output that will have logic level high when the chip detects signal voltages from the RS-232 device its plugged into. You can use it as a 'connection made' signal.
- **OFF** - This pin can be set to logic high by the microcontroller to tri-state all the pins for power reduction.

RS-232 Wiring

Making a connection between the adapter and an RS-232 connection on a cable or other piece of equipment may involve more than a compatible cable. The RS-232 standard and various implementations can be tricky, as people found "back in the day".

Below is a mini-FAQ on wiring used for RS-232. Refer to Wikipedia (https://adafru.it/1a69) or other sources for more detailed information.

The characters on the line are garbled

There is likely a mismatch in baud rate or character sizes. First be sure you are transmitting or receiving at the same baud rate on both sides. Then confirm if the interface number of data bits and stop bits is correct.

While more modern equipment will take 8 bit data, one stop bit, older equipment may specify 7 bit data and 1 or 2 stop bits. Also older equipment could specify a parity bit but usually in an 8 data bit, 1 stop bit configuration there is no parity.

In MS-DOS, for example, using RS-232 port COM1 at 19200 baud would be set up at the DOS command line `MODE COM1:19200,N,8,1`. Other operating systems may need a similar setup.

I know the connection is wired correctly and trying to communicate but there is no activity

From Wikipedia: RS-232 devices may be classified as Data Terminal Equipment (DTE) or Data Circuit-terminating Equipment (DCE); this defines at each device which wires will be sending and receiving each

signal. According to the standard, male connectors have DTE pin functions, and female connectors have DCE pin functions. Other devices may have any combination of connector gender and pin definitions. Many terminals were manufactured with female connectors but were sold with a cable with male connectors at each end; the terminal with its cable satisfied the recommendations in the standard.

Thus it could be the wild west out there. Just crossing TX and RX like on a microcontroller might not be enough as the RTS and CTS signals likely need swapping too. The RS-232 solution is either a cable or adapter called a crossover or null modem adapter.

The connector on the equipment doesn't mate with the Adafruit board connector

First double check the port you want to connect to is truly RS-232 (some folks use the same connectors for Centronics parallel or other things).

You may see male or female 25-pin DE-25M or 9-pin DE-9M connectors on RS-232 connections. This includes 25 pin on one end, 9 pin on the other. It was common to use adapters from one to another to make the connection. Some cords like for computer to computer communication ("Laplink cables") had both connectors at each end of the cable for maximum compatibility.

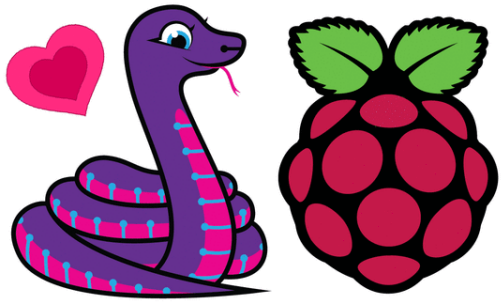
If you have a mismatch, there are adapters to go from 9 pin to 25 pin and the other way. Also see above about the possibility of a null modem adapter also in the mix to get the signals to the right pins on each side. It can be a mess. Many folks made their own cables to get from once piece of gear to another.

CircuitPython and Python

It's easy to use the **RS-232 Full Pinout Breakout** with Python or CircuitPython, and the built-in UART module. This module allows you to easily write Python code to send and receive UART messages.

You can use this driver with any CircuitPython microcontroller board or with a computer that has GPIO, UART and Python thanks to Adafruit_Blinka, our

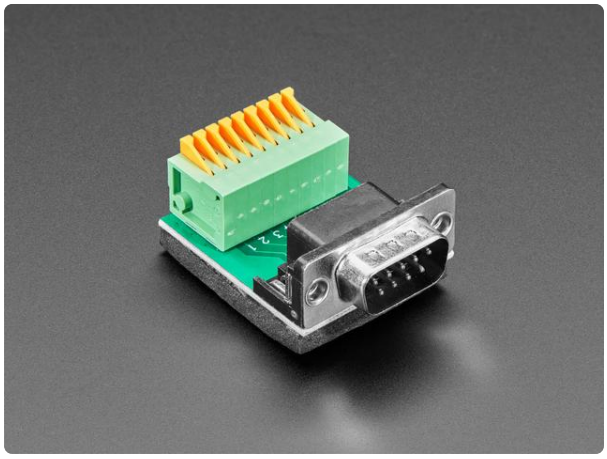
CircuitPython-for-Python compatibility library (https://adafru.it/BSN). For Blinka, this example was tested with a Raspberry Pi. To use the example as-is with built-in UART, you'll need to follow these configuration steps (https://adafru.it/CEk) outlined in the Blinka Learn Guide.



CircuitPython Libraries on Linux and Raspberry Pi
By M. LeBlanc-Williams
UART / Serial

https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/uart-serial

You'll need an external RS-232 device to use this example with the breakout:

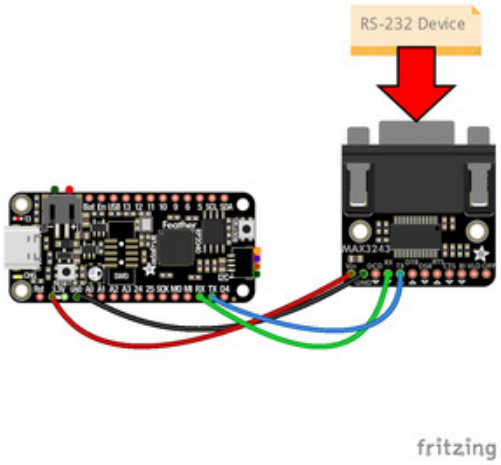


DE-9 (DB-9) Male Plug to Terminal Spring Block Adapter
Make your own custom DE-9 cable or DE-9 interface with ease, using this DE-9 Plug to Terminal Spring Block Adapter. You get a nice solid PCB with a DE-9 plug and...
https://www.adafruit.com/product/4512

USB/Serial Converter
This USB cable adds a DB9 serial port to your computer or laptop. Works with MacOS X, Linux and Windows. Note that this provides (approximately) +-6V serial (RS232) not 5V serial...
https://www.adafruit.com/product/18

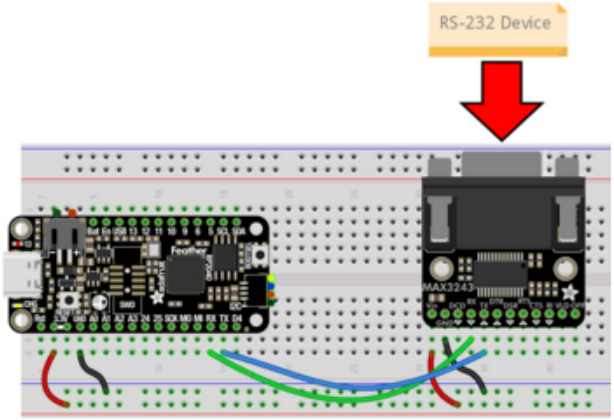
CircuitPython Microcontroller Wiring

First, plug in an RS-232 device to the DE-9 connector on the RS-232 breakout. Then, wire up the RS-232 breakout to your board. The following is the breakout wired to a Feather RP2040:



Feather 3.3V to **breakout Vin (red wire)**
Feather GND to **breakout GND (black wire)**
Feather RX to **breakout RX (green wire)**
Feather TX to **breakout TX (blue wire)**

The following is the breakout wired to a Feather RP2040 using a solderless breadboard:

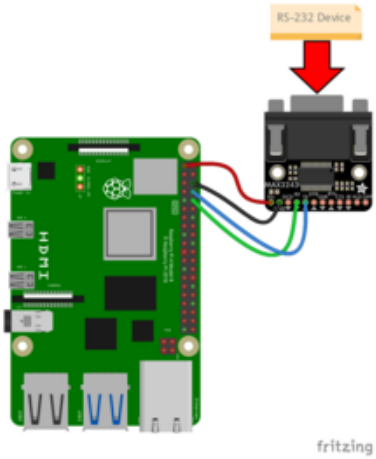


Feather 3.3V to **breakout Vin (red wire)**
Feather GND to **breakout GND (black wire)**
Feather RX to **breakout RX (green wire)**
Feather TX to **breakout TX (blue wire)**

Python Computer Wiring

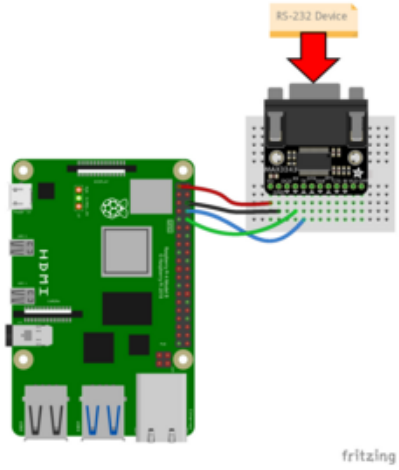
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN)](https://adafru.it/BSN).

Here's the Raspberry Pi wired to the RS-232 breakout. You'll plug in an RS-232 device to the DE-9 port on the breakout:



Pi 3.3V to **breakout Vin (red wire)**
Pi GND to **breakout GND (black wire)**
Pi RX to **breakout RX (green wire)**
Pi TX to **breakout TX (blue wire)**

Here's the Raspberry Pi wired up using a solderless breadboard:



Pi 3.3V to **breakout Vin (red wire)**
Pi GND to **breakout GND (black wire)**
Pi RX to **breakout RX (green wire)**
Pi TX to **breakout TX (blue wire)**

CircuitPython Usage

To use with CircuitPython, you'll 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 **code.py** file to your **CIRCUITPY** drive.

There are no additional libraries needed for this example. All of the modules used are built-in.

No additional libraries need to be added to the /lib folder on your CIRCUITPY drive.

Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

```
python3 code.py
```

Make sure to follow the built-in UART setup steps if you are using the UART on the Raspberry Pi.

Example Code

If running CircuitPython: Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!

If running Python: The console output will appear wherever you are running Python.

```
# SPDX-FileCopyrightText: 2024 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board

# baud rate for your device
baud = 38400
# Initialize UART for the CH9328
# check for Raspberry Pi
# pylint: disable=simplifiable-condition
if "CE0" and "CE1" in dir(board):
    import serial

    uart = serial.Serial("/dev/ttyS0", baudrate=baud, timeout=3000)
# otherwise use busio
else:
    import busio

    uart = busio.UART(board.TX, board.RX, baudrate=baud)

print("Enter commands to send to the RS-232 device. Press Ctrl+C to exit.")
while True:
    user_input = input("Please enter your command: ").strip()
    if user_input:
        # send the command with a telnet newline (CR + LF)
        uart.write((user_input + "\r\n").encode('ascii'))
```

```
    # empty buffer to collect the incoming data
    response_buffer = bytearray()

    # check for data
    time.sleep(1)
    while uart.in_waiting:
        data = uart.read(uart.in_waiting)
        if data:
            response_buffer.extend(data)

    # decode and print
    if response_buffer:
        print(response_buffer.decode('ascii'), end='')
        print()
```

The code begins by initializing the UART port. There is a check for a Raspberry Pi single board computer. If a Raspberry Pi is detected, then `serial` is used instead of `busio` for UART over GPIO. The baud rate for your RS-232 device is defined with `baud`. Edit this value to match your RS-232 device.

```
# baud rate for your device
baud = 38400
```

In the loop, the code waits for user input in the serial console. If any incoming data is received from the RS-232 device, it is printed to the serial console.

The commands that you send will vary by the RS-232 device that you connect to the RS-232 breakout. The output below was sent to a StarTech HDMI switcher (https://adafru.it/1a47). The `AVI=n` commands switch the selected HDMI port and the `VS` command gives a current status of the device.

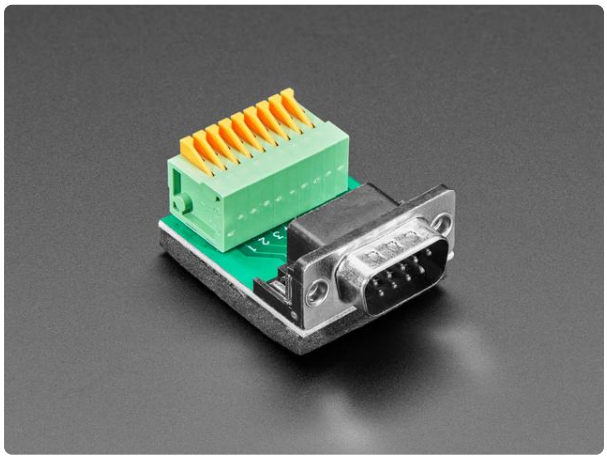
```
CircuitPython REPL
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Enter commands to send to the RS-232 device. Press Ctrl+C to exit.
Please enter your command: AVI=2
AVI=2
Done

>

Please enter your command: VS
VS

EGO Switch - Auto-Sensing
RC ID - none
================================================
Please enter your command: AVI=7
AVI=7
Error: Invalid parameter

AVI=n   - Select input port n as the source of all output port
Please enter your command: AVI=1
AVI=1
Done

>

Please enter your command:
```

Python Docs

Python Docs (https://adafru.it/1a42)

Arduino

Using the RS-232 Full Pinout Breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller with an external RS-232 device and running the provided example code.



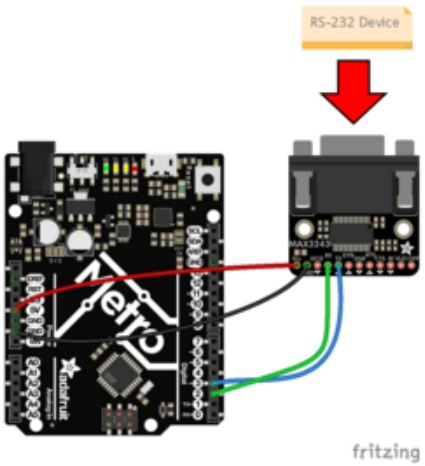
DE-9 (DB-9) Male Plug to Terminal Spring Block Adapter
Make your own custom DE-9 cable or DE-9 interface with ease, using this DE-9 Plug to Terminal Spring Block Adapter. You get a nice solid PCB with a DE-9 plug and...
https://www.adafruit.com/product/4512

USB/Serial Converter
This USB cable adds a DB9 serial port to your computer or laptop. Works with MacOS X, Linux and Windows. Note that this provides (approximately) +-6V serial (RS232) not 5V serial...
https://www.adafruit.com/product/18

Wiring

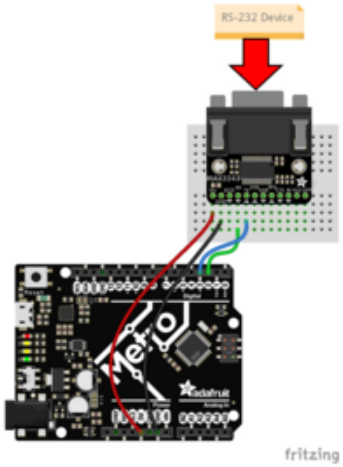
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the breakout VIN.

First, plug in an RS-232 device to the DE-9 connector on the RS-232 breakout. Then, wire up the Adafruit Metro to the breakout:



Metro 5V to **breakout Vin (red wire)**
Metro GND to **breakout GND (black wire)**
Metro pin 2 to **breakout RX (green wire)**
Metro pin 3 to **breakout TX (blue wire)**

Here is an Adafruit Metro wired up using a solderless breadboard:



Metro 5V to **breakout Vin (red wire)**
Metro GND to **breakout GND (black wire)**
Metro pin 2 to **breakout RX (green wire)**
Metro pin 3 to **breakout TX (blue wire)**

There are no additional libraries needed for this example.

Example Code

You can change the baud rate for your RS-232 device at the top of the code by editing the `baud` define:

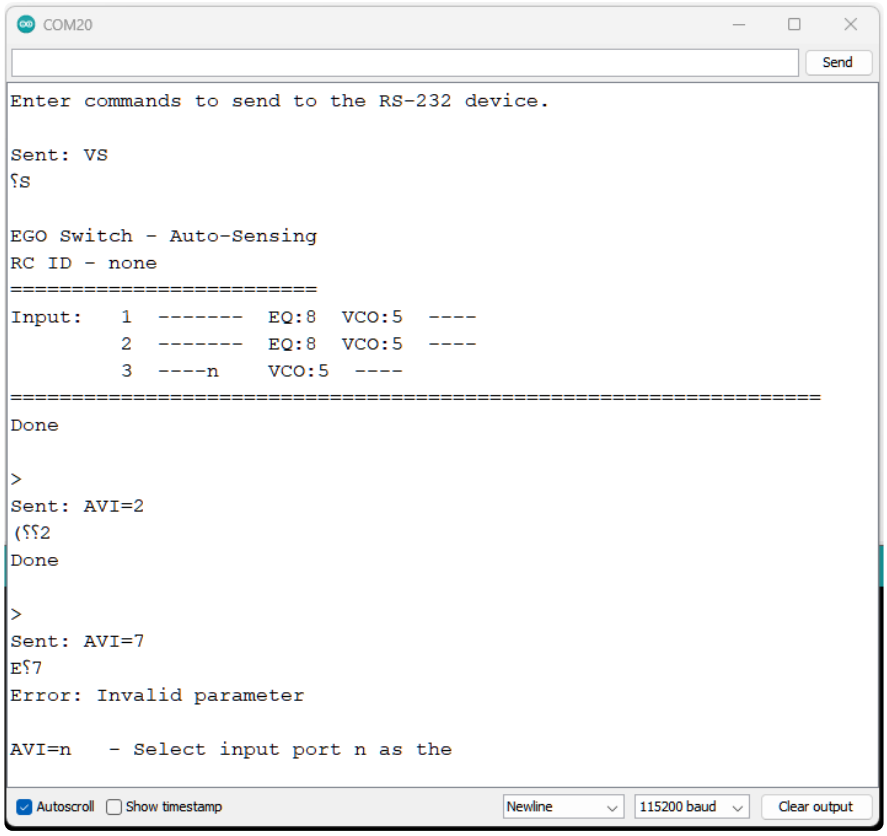
```
#define baud 38400
```

```
// SPDX-FileCopyrightText: 2024 Liz Clark for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <SoftwareSerial.h>

// update this for your RS-232 device baud rate
#define baud 38400
// define RX and TX pins for the software serial port
```

```
#define RS232_RX_PIN 2
#define RS232_TX_PIN 3

SoftwareSerial rs232Serial(RS232_RX_PIN, RS232_TX_PIN);

void setup() {
  Serial.begin(115200);
  while ( !Serial ) delay(10);

  rs232Serial.begin(baud);

  Serial.println("Enter commands to send to the RS-232 device.");
  Serial.println();
}

void loop() {

  if (Serial.available() > 0) {
    String userInput = Serial.readStringUntil('\n');
    userInput.trim();  // remove any trailing newlines or spaces
    if (userInput.length() > 0) {
      // send the command with a telnet newline (CR + LF)
      rs232Serial.print(userInput + "\r\n");
      Serial.print("Sent: ");
      Serial.println(userInput);
    }
  }

  // check for incoming data from RS-232 device
  while (rs232Serial.available() > 0) {
    char response = rs232Serial.read();
    // print the incoming data
    Serial.print(response);
  }

  delay(50);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You can send commands to your RS-232 device via the Serial Monitor. If any messages are received from the RS-232 device, they will print to the Serial Monitor.

The baud rate in the bottom right of the serial window is not the same as the baud rate of the RS-232 connection. It is the data rate to the serial monitor which is independent.



The commands that you send will vary by the RS-232 device that you connect to the RS-232 breakout. The output above was sent to a StarTech HDMI switcher (https://adafru.it/1a47). The `AVI=n` commands switch the selected HDMI port and the `VS` command gives a current status of the device.

Arduino Docs

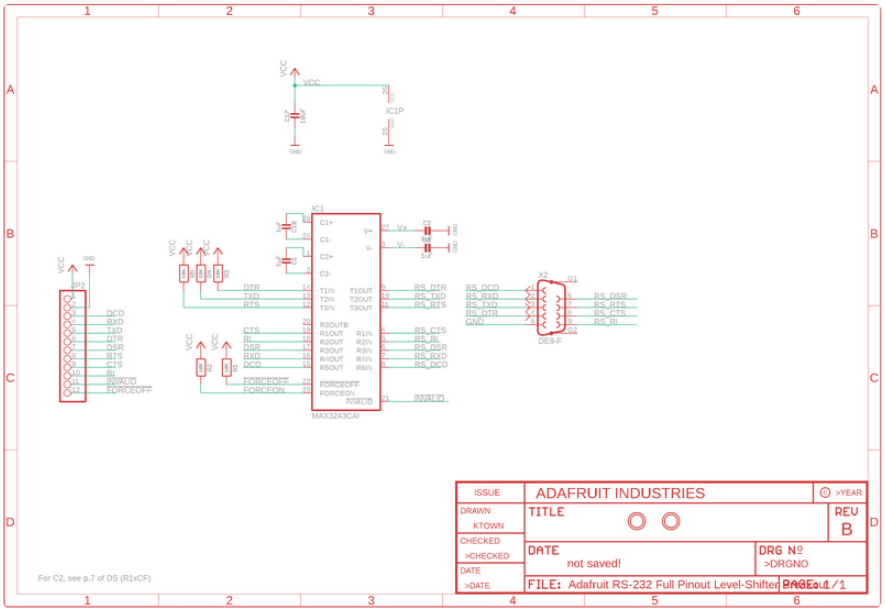
Arduino Docs (https://adafru.it/1a43)

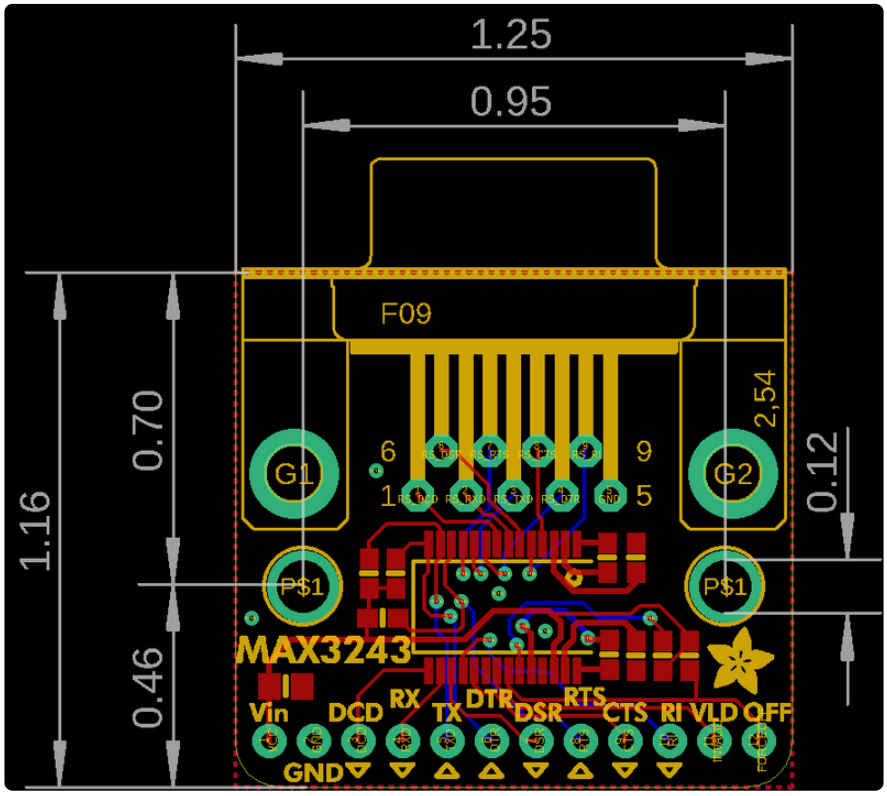
Downloads

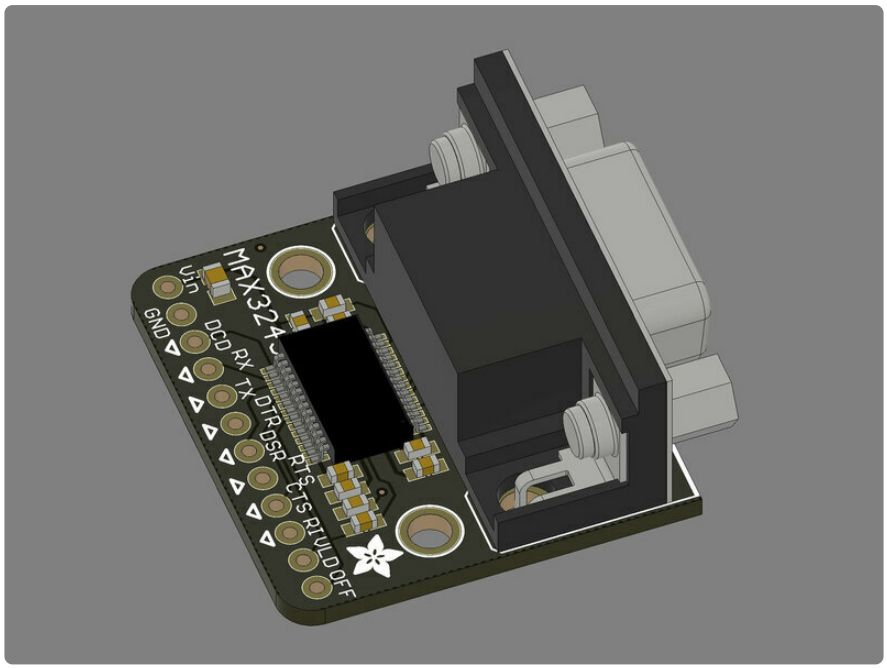
Files

- MAX3243 Datasheet (https://adafru.it/1a63)
- EagleCAD PCB Files on GitHub (https://adafru.it/1a64)
- 3D models on GitHub (https://adafru.it/1a6L)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a65)

Schematic and Fab Print






MAX324
VIN
GND
DCD RX TX DTR DSR RTS CTS RI VL DOFF