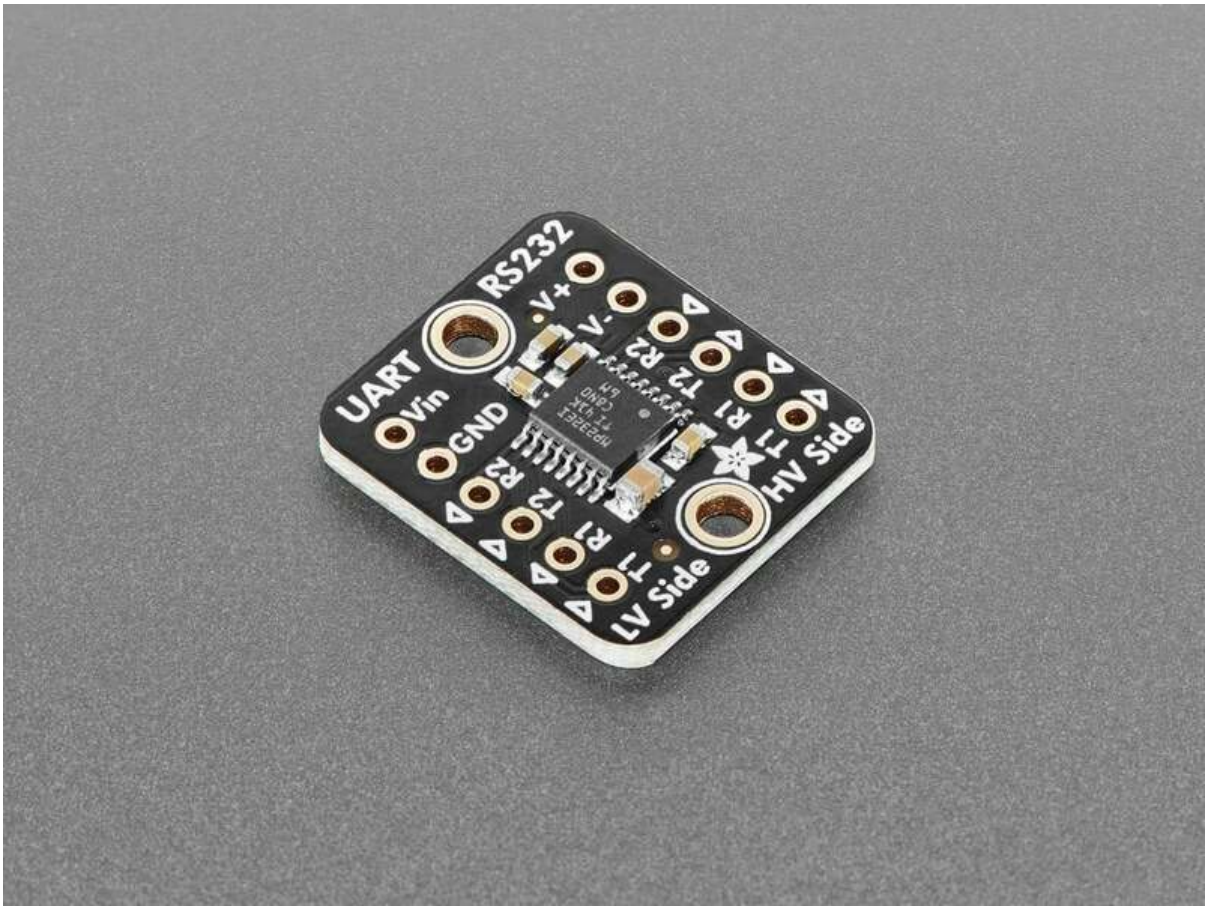
Adafruit RS232 Pal

Created by Liz Clark



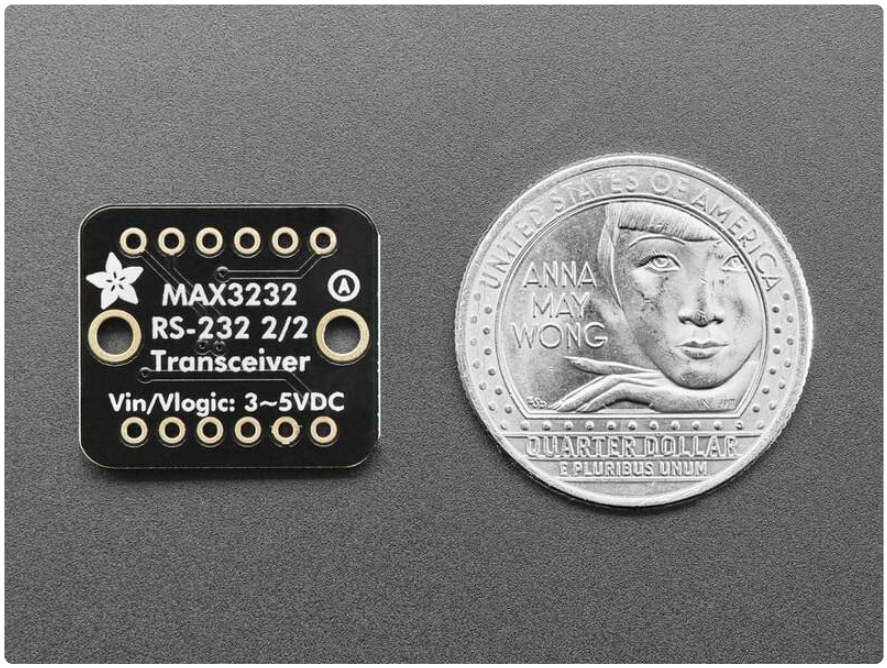
https://learn.adafruit.com/adafruit-rs232-pal

Last updated on 2024-07-12 04:47:01 PM EDT

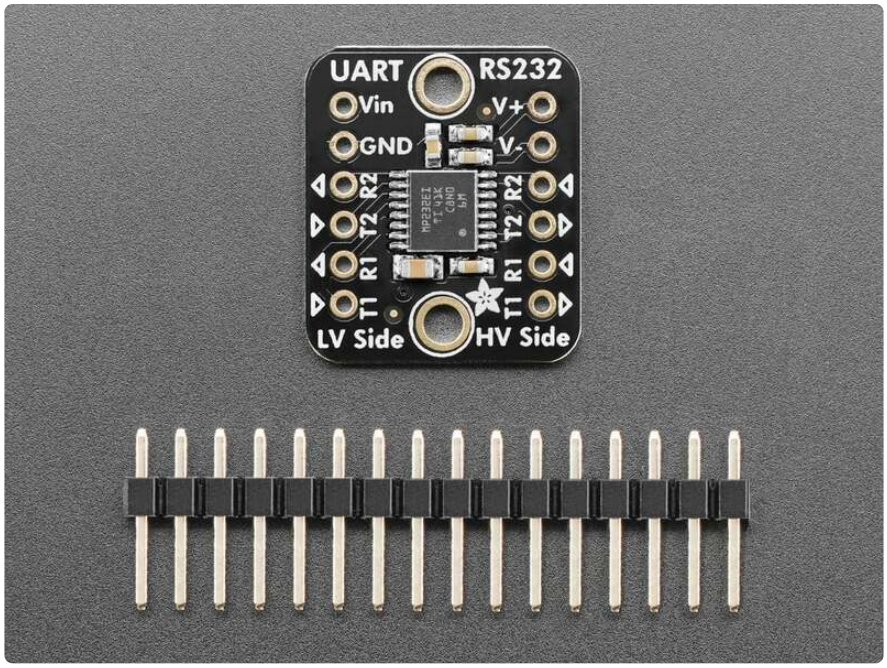
Table of Contents

Overview

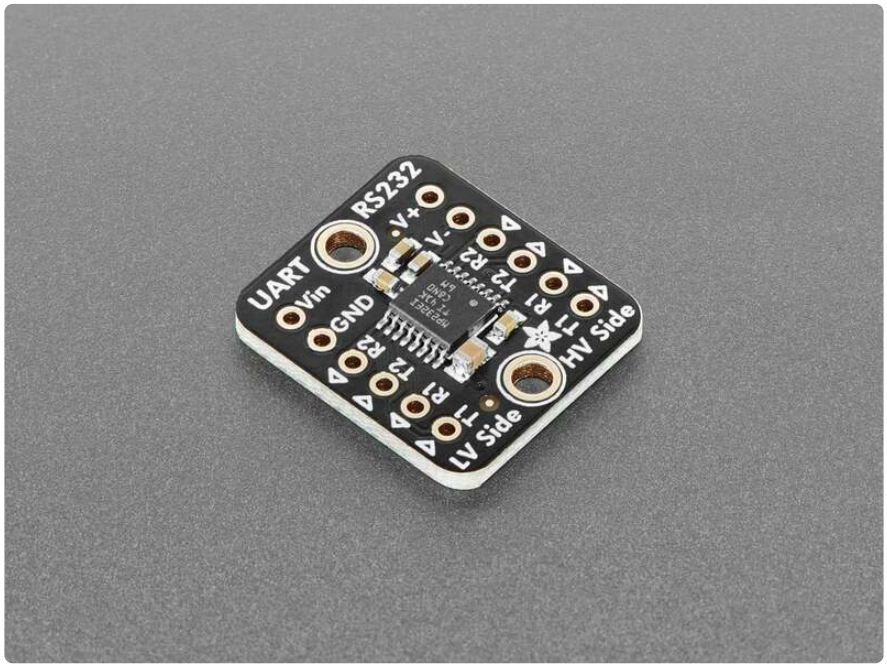
If you're looking to interface with telco, retro or industrial equipment you'll probably run into RS-232 interfaces. **The Adafruit RS232 Pal - Two Channel UART to RS-232 Level Shifters** is your friend in such cases, giving you two duplex channels of level shifting and taking care of the high/negative voltage generation all in a low cost breakout board. We use the trusty MAX3232E (https://adafru.it/1a44), a classic chip that is 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 would allow data plus flow control lines. Many folks may remember these interfaces were used for mice, modems, barcode scanners, tele-types and more. We still find devices sold that have RS-232 ports, although many folks use a USB to RS-232 adapter (https://adafru.it/1a45) these days.



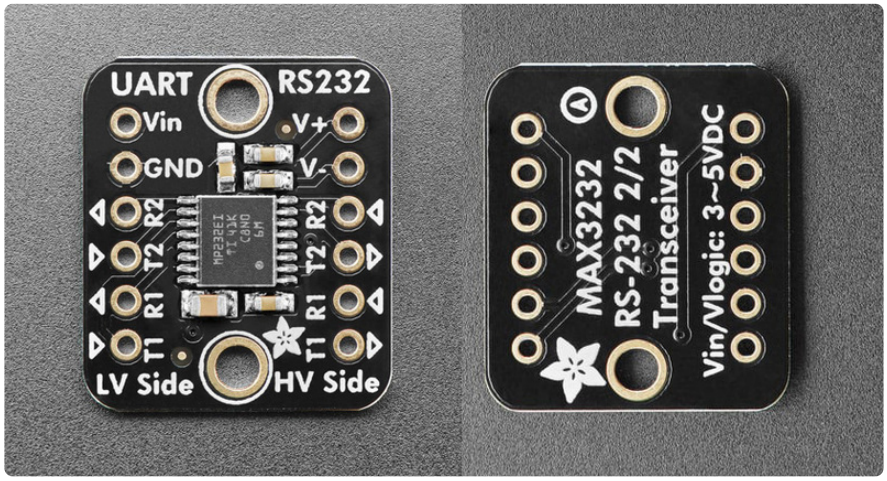
If you want to use a microcontroller or microcomputer to chat with a RS-232 then thankfully all you need is a serial port / UART (something just about any microcontroller has) and a level shifter. The level shifter is required because while most UARTs are 0-3.3V or 0-5V logic level, RS-232 requires +-6 to +-10V, yep the signal voltage goes negative! That means a specialized shifter is required, one that can generate the extra high and low voltages and also 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 plus it can run on 3.3V power and logic, which many older chips can't do. It also can do 2 duplex channels, so you can have both RX and TX plus two flow control pins like RTS and CTS.

This breakout comes fully assembled with a UART side for low voltage power/logic level and an RS-232 side for high voltage signals plus the doubled and negative voltage rails in case you need to reference them. We also include a bit of header so you can solder it 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 MAX3232E 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.
- **V+** - positive voltage rail output from the RS-232 connection. This pin can be used as a reference voltage.
- **V-** - negative voltage rail output from the RS-232 connection. This pin can be used as a reference voltage.

Low Voltage Side Pins

The low voltage side of the board, labeled **LV Side** on the board silk, are located on the left side of the board. These pins can interface with your microcontroller UART

pins with 3.3V or 5V logic. There are two duplex channels, so you can have both RX and TX plus two flow control pins like RTS and CTS.

- **T1** - Logic data input (from UART)
- **R1** - Logic data output (to UART)
- **T2** - Logic data input (from UART)
- **R2** - Logic data output (to UART)

High Voltage Side Pins

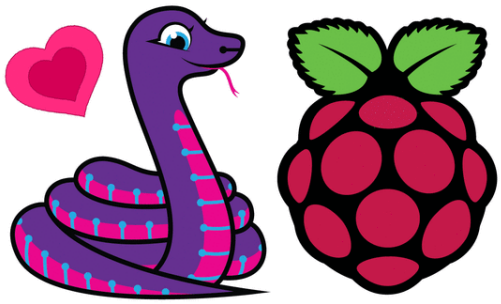
The high voltage side of the board, labeled **HV Side** on the board silk, are located on the right side of the board. These pins can interface with a RS-232 connection, which can require +-6 to +-10V. There are two duplex channels, so you can have both RX and TX plus two flow control pins like RTS and CTS.

- **T1** - RS-232 line data output (to remote RS-232 system)
- **R1** - RS-232 line data input (from remote RS-232 system)
- **T2** - RS-232 line data output (to remote RS-232 system)
- **R2** - RS-232 line data input (from remote RS-232 system)

CircuitPython and Python

It's easy to use the **RS232 PaI** 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.

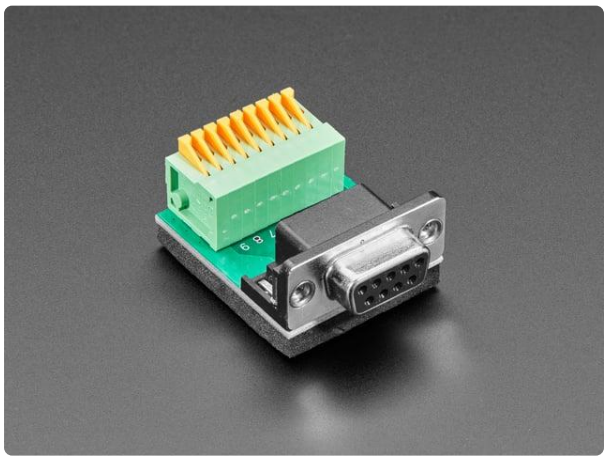
You can use this driver with any CircuitPython microcontroller board or with a computer that has GPIO, UART and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN). For Blinka, this example was tested with a Raspberry Pi. To use the example as-is 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:

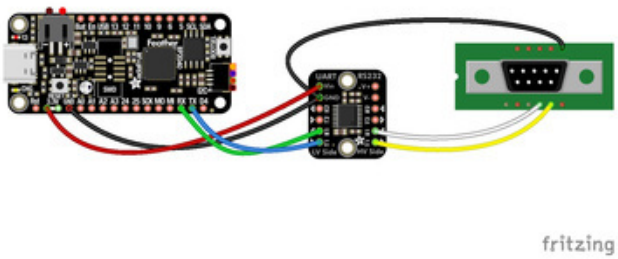


DE-9 (DB-9) Female Socket to Terminal Spring Block Adapter
Make your own custom DE-9 cable or DE-9 interface with ease, using this DE-9 Socket to Terminal Spring Block Adapter. You get a nice solid PCB with a DE-9 socket...
https://www.adafruit.com/product/4510

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

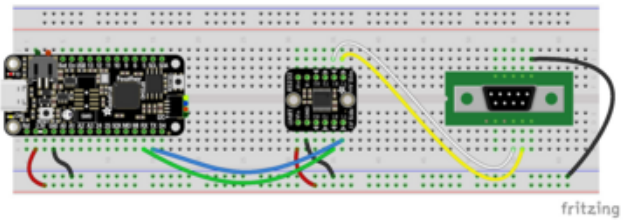
CircuitPython Microcontroller Wiring

First, wire up the RS-232 connection to the RS232 Pal exactly as follows. Then, wire up the RS232 Pal to your board. The following is the breakout wired to a Feather RP2040 and RS-232 connection:



Feather 3.3V to **breakout Vin (red wire)**
Feather GND to **breakout GND (black wire)**
Feather RX to **breakout low voltage R1 (green wire)**
Feather TX to **breakout low voltage T1 (blue wire)**
Breakout high voltage R1 to **RS-232 RXD [pin 2 on DE-9] (white wire)**
Breakout high voltage T1 to **RS-232 TXD [pin 3 on DE-9] (yellow wire)**
Breakout GND to **RS-232 GND [pin 5 on DE-9] (black wire)**

The following is the breakout wired to a Feather RP2040 and RS-232 connection using a solderless breadboard:

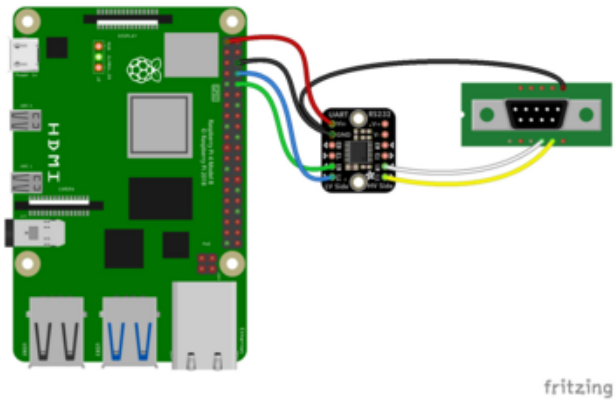


Feather 3.3V to **breakout Vin (red wire)**
Feather GND to **breakout GND (black wire)**
Feather RX to **breakout low voltage R1 (green wire)**
Feather TX to **breakout low voltage T1 (blue wire)**
Breakout high voltage R1 to **RS-232 RXD [pin 2 on DE-9] (white wire)**
Breakout high voltage T1 to **RS-232 TXD [pin 3 on DE-9] (yellow wire)**
Breakout GND to **RS-232 GND [pin 5 on DE-9] (black wire)**

Python Computer Wiring

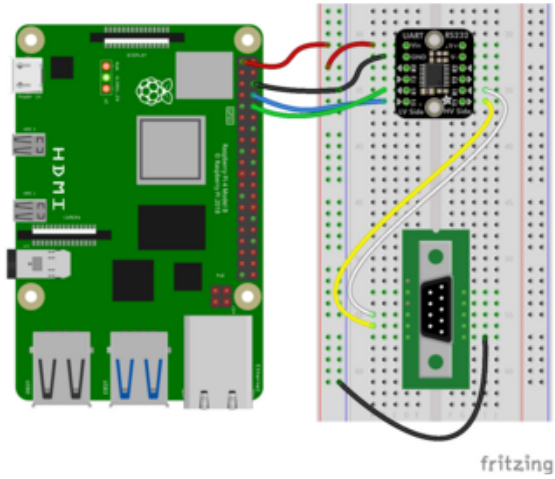
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN)](https://adafru.it/BSN).

Here's the Raspberry Pi wired to the RS232 Pal and RS-232 connection:



Pi 3.3V to **breakout Vin (red wire)**
Pi GND to **breakout GND (black wire)**
Pi RX to **breakout low voltage R1 (green wire)**
Pi TX to **breakout low voltage T1 (blue wire)**
Breakout high voltage R1 to **RS-232 RXD [pin 2 on DE-9] (white wire)**
Breakout high voltage T1 to **RS-232 TXD [pin 3 on DE-9] (yellow wire)**
Breakout GND to **RS-232 GND [pin 5 on DE-9] (black 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 low voltage R1 (green wire)**
Pi TX to **breakout low voltage T1 (blue wire)**
Breakout high voltage R1 to **RS-232 RXD [pin 2 on DE-9] (white wire)**
Breakout high voltage T1 to **RS-232 TXD [pin 3 on DE-9] (yellow wire)**
Breakout GND to **RS-232 GND [pin 5 on DE-9] (black 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 RS232 Pal. 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.

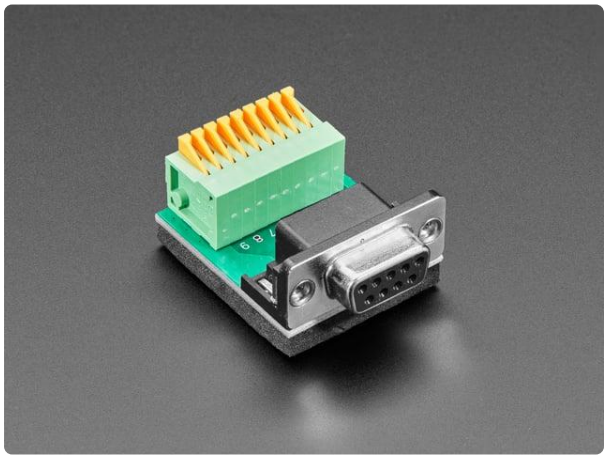
```
CircuitPython REPL
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Enter commands to send to the RS-232 device. Press Ctrl+C to exit.
Please enter your command: AVI=2
AVI=2
Done

>

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 RS232 PAL with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller with an external RS-232 device and running the provided example code.



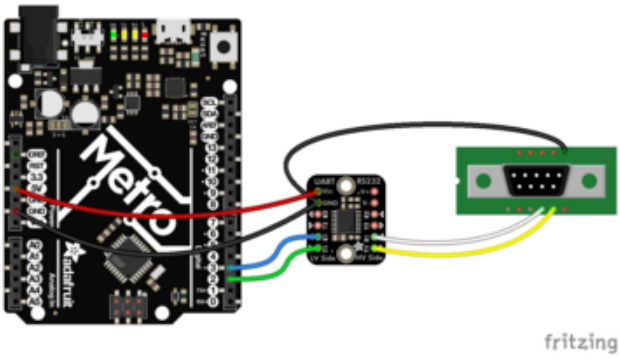
DE-9 (DB-9) Female Socket to Terminal Spring Block Adapter
Make your own custom DE-9 cable or DE-9 interface with ease, using this DE-9 Socket to Terminal Spring Block Adapter. You get a nice solid PCB with a DE-9 socket...
https://www.adafruit.com/product/4510

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

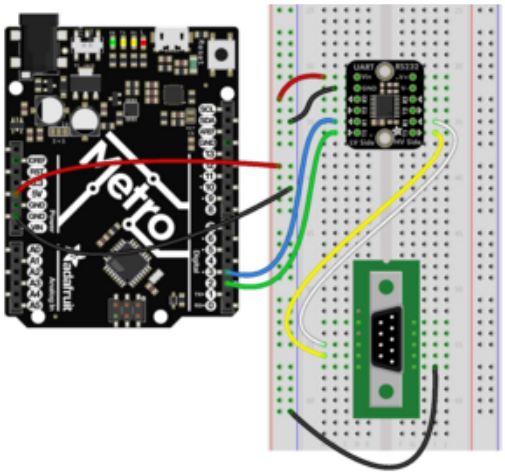
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the breakout VIN.

Here is an Adafruit Metro wired up to the breakout with an external RS-232 connection:



Metro 5V to **breakout Vin (red wire)**
Metro GND to **breakout GND (black wire)**
Metro pin 2 to **breakout low voltage R1 (green wire)**
Metro pin 3 to **breakout low voltage T1 (blue wire)**
Breakout high voltage R1 to **RS-232 RXD [pin 2 on DE-9] (white wire)**
Breakout high voltage T1 to **RS-232 TXD [pin 3 on DE-9] (yellow wire)**
Breakout GND to **RS-232 GND [pin 5 on DE-9] (black 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 low voltage R1 (green wire)**
Metro pin 3 to **breakout low voltage T1 (blue wire)**
Breakout high voltage R1 to **RS-232 RXD [pin 2 on DE-9] (white wire)**
Breakout high voltage T1 to **RS-232 TXD [pin 3 on DE-9] (yellow wire)**
Breakout GND to **RS-232 GND [pin 5 on DE-9] (black wire)**

There are no additional libraries needed for this example.

Example Code

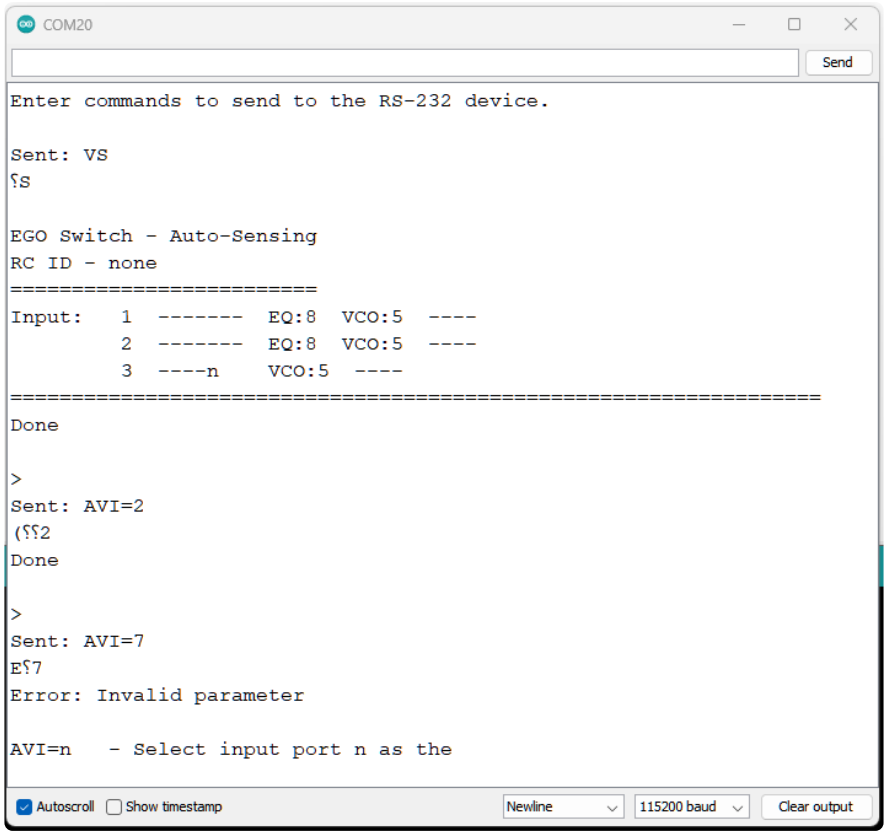
You can change the baud rate for your RS-232 device at the top of the code by editing the `baud` define:

```
#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 commands that you send will vary by the RS-232 device that you connect to the RS232 Pal. 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

- MAX3232E Datasheet (https://adafru.it/1a48)
- EagleCAD PCB Files on GitHub (https://adafru.it/1a49)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a4a)

Schematic and Fab Print