COMMON PITFALLS

That's about all there is to serial communication. I'd like to leave you with a few common mistakes that are easy for an engineer of any experience level to make:

RX-to-TX, TX-to-RX

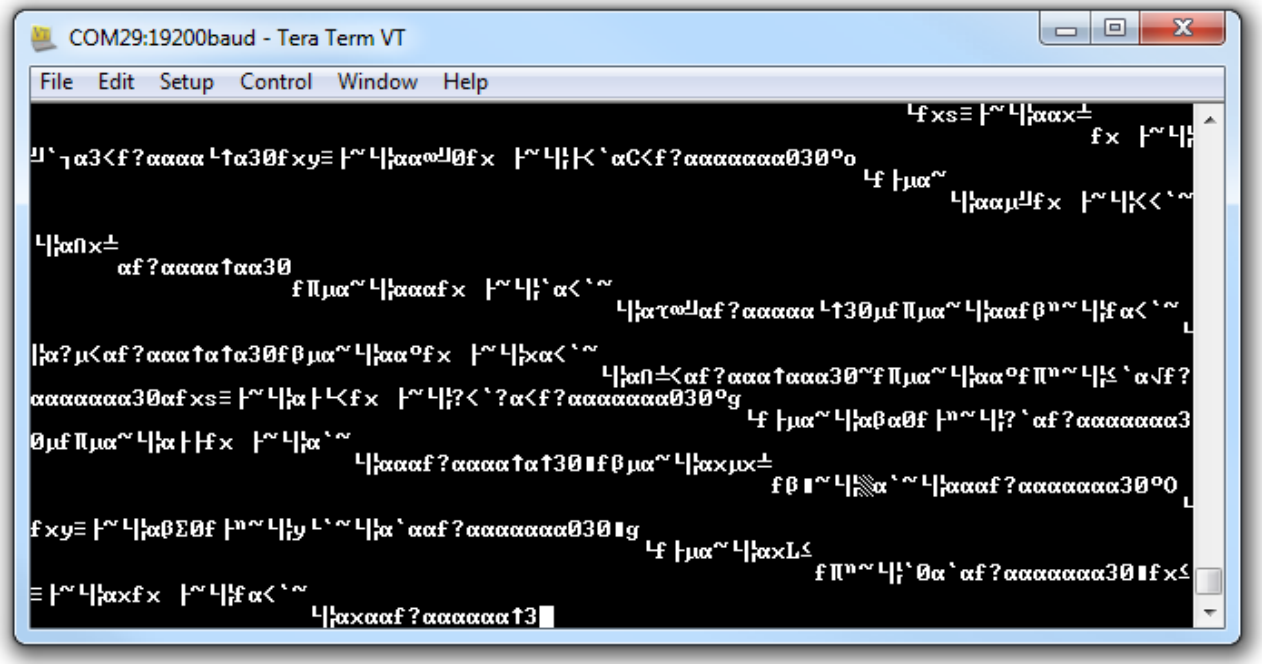
Seems simple enough, but it's a mistake I know I've made more than a few times. As much as you want their labels to match up, always make sure to cross the RX and TX lines between serial devices.

FTDI Basic programming a Pro Mini. Note RX and TX's crossed!

Contrary to what the esteemed Dr. Egon Spengler would warn, **cross the streams**.

Baud Rate Mismatch

Baud rates are like the languages of serial communication. If two devices aren't speaking at the same speed, data can be either misinterpreted, or completely missed. If all the receiving device sees on its receive line is garbage, check to make sure the baud rates match up.

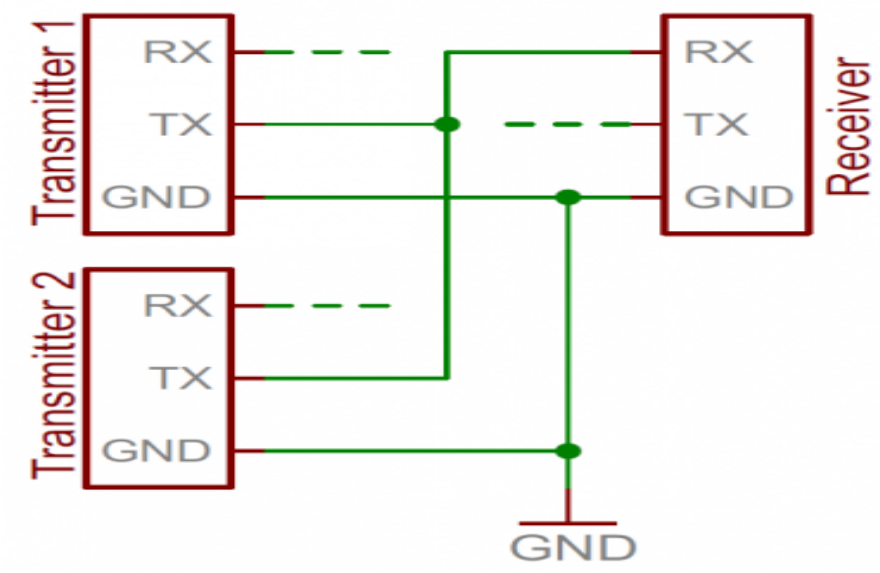


Data transmitted at 9600 bps, but received at 19200 bps. Baud mismatch = garbage.

Bus Contention

Serial communication is designed to allow just two devices to communicate across one serial bus. If more than one device is trying to transmit on the same serial line you could run into bus-contention. Dun dun dun….

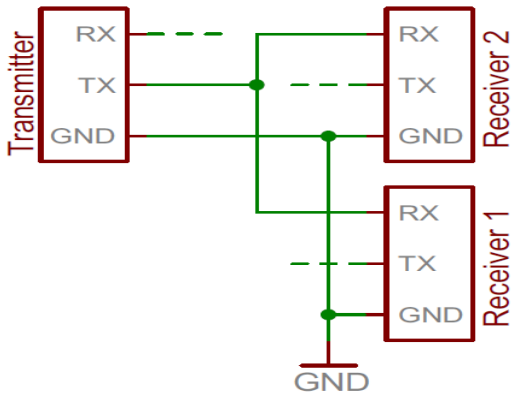
For example, if you're connecting a GPS module up to your Arduino, you may just wire that module's TX line up the Arduino's RX line. But that Arduino RX pin is already wired up to the TX pin of the USB-to-serial converter, which is used whenever you program the Arduino or use the *Serial Monitor*. This sets up the potential situation where both the GPS module and FTDI chip are trying to transmit on the same line at the same time.



Two transmitters sending to a single receiver sets up the possibility for bus contention.

Two devices trying to transmit data at the same time, on the same line, is bad! At "best" neither of the devices will get to send their data. At worst, both device's transmit lines go poof (though that's rare, and usually protected against).

It can be safe to connect multiple receiving devices to a single transmitting device. Not really up to spec and probably frowned upon by a hardened engineer, but it'll work. For example, if you're connecting a serial LCD up to an Arduino, the easiest approach may be to connect the LCD module's RX line to the Arduino's TX line. The Arduino's TX is already connected to the USB programmer's RX line, but that still leaves just one device in control of the transmission line.



Distributing a TX line like this can still be dangerous from a firmware perspective, because you can't pick and choose which device hears what transmission. The LCD will end up receiving data not meant for it, which could command it to go into an unknown state.

Source: https://learn.sparkfun.com/tutorials/serial-communication