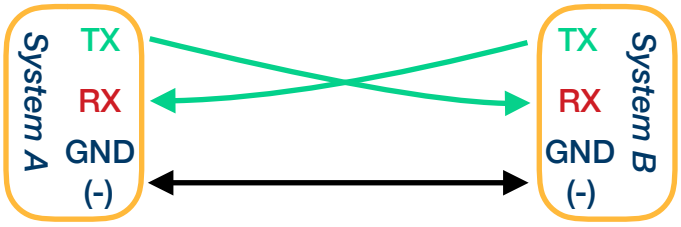
The Serial Monitor

As mentioned, many uses for microcontrollers have them embedded in a device, interacting with either humans or components though very specific inputs and outputs — but sometimes they need to communicate with other complex systems. The programming process itself is one example of such communication, and observing the logic and workings of your program, aka debugging, is another.

We have already used a serial link to program the microcontroller — the IDE did most of the work, we just had to specify the port. We also connected using USB, the Universal Serial Bus (which itself is just a fancy, high speed, multiplexed serial connection.) Various bits of software and a custom chip on the bottom of the Nano carrier board handle all of the USB stuff, and the AVR chip just sees a simple serial link.

What exactly is a serial link?

Serial communication is one of the oldest and most simplistic means for two digital computers to exchange data. In its most basic form a serial link has only three wires: a single DC ground wire, one for the system to transmit (**tx**) data on, and one for the system to receive (**rx**) data from. Given two systems A and B, system A's **tx** wire connects to system B's **rx** wire and visa-versa (see diagram). Pulses of voltage are sent down the wire to represent ones and zeros, with a positive voltage indicating a one.



Of course, it's not quite *that* simple. There must be some way to determine how often the pulsed can be sent — what if one computer was faster than the other? A specific rate must be agreed upon between the two systems and the pulses must come at that specified number per second, called the baud rate. Almost any rate can be used, though there are some standard ones, as long as both sides use the same one and both are fast enough to send and receive at that rate. Although there are other consideration and options to handle things like errors and disconnects, that's all there is to the basic serial link.

Of course, our IDE handles the baud and various other details of the serial link. All we have to do is set a few options. In the Tools menu of the Arduino IDE there is a 'Serial Monitor' option. Assuming your Nano is connected and the serial port specified correctly, selecting that item will open a new window. At the bottom right of the window there are a couple of pull-downs. We are primarily concerned with the far right one which determines the baud rate. The baud rate selector offers options from 300 through 250,000, although many are some slightly unusual numbers that are multiples of each other. We'll look later at why some of those numbers are what they are, but for now select 115200 for the baud, and make sure that the other pull down is set to 'No Line Ending'.

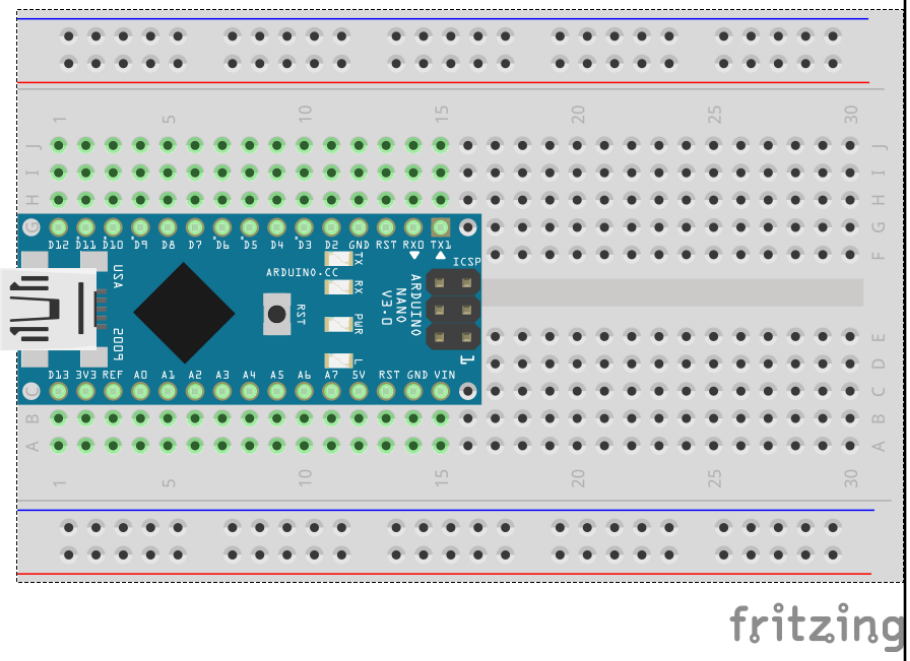
Note also the 'autoscroll' checkbox on the left hand side. When new data comes in over the serial link it is added to the bottom of the scrolling text area of the window — if you have scrolled the window back up to review some earlier items, it will jump to the bottom when new data arrives. If you need to review older data while new data is still coming in, you will probably want to uncheck that feature.

The Arduino IDE also uses the serial connection to send your software to the device, but it is somewhat intelligent about it. If you have the serial monitor open when you send new software to the device, it will stop the serial monitor, blank the window, then resume monitoring after the software is loaded and normal execution has resumed.

When you first open the serial monitor, it will send a reset to the device, causing it to begin again with the 'setup' as though it was just powered up or the reset button was pushed. This is because there are some optional extra control pins many serial ports have to help negotiate an initial baud rate (aka hardware flow control) and the serial monitor tries to use these. However on most prototyping boards such as our nano, one of these pins is connected directly to the reset line.

In this experiment, we are just loading a pre-built program and interacting with the serial monitor, but once it is working, try hitting reset, and starting and stopping the serial monitor, or even disconnecting the USB connection while it is running.

And, as time permits feel free to try to understand or even modify the program to do something slightly different. If you see how it works, you can even try altering the baud rate.

Components	Wiring Diagram
✓Computer ✓Arduino IDE	

Connection Instructions

Only the Nano in the breadboard are neeeded.

Sketch(es)	serialIO.ino

Analysis Questions

Did your program upload and run okay?
If not, can you see the error?
If so, how do you know?

Programming Tasks

- Open the serialIO.ino sketch and upload it to your Nano.
- Open the Serial Monitor found in the Tools menu, and select 115200 baud.
- Answer the questions that appear in the Serial Monitor by typing into the top line and clicking Send.