BASIC TERMINOLOGY

Here are some terms you should be familiar with when working within a serial terminal window. Many of these terms are covered in a lot more detail in our Serial Communication tutorial. It highly recommended that you read that page as well to get the full picture.

ASCII - Short for the American Standard Code for Information Interchange's character encoding scheme, ASCII encodes special characters from our keyboards and converts them to 7-bit binary integers that can be recognized by a number of programs and devices. ASCII charts are very helpful when working with serial terminals.

Baud Rate - In short, baud rate is how fast your data is being transmitted and received. 9600 is the standard rate, but other speeds are typical amongst certain devices. Just remember that all the links in your chain of communication have to be "speaking" at the same speed, otherwise data will be misinterpreted on one end or the other.

Transmit (TX) - Also known as Data Out or TXO. The TX line on any device is there to transmit data. This should be hooked up to the RX line of the device with which you would like to communicate.

Receive (RX) - Also known as Data In or RXI. The RX line on any device is there to receive data. This should be hooked up to the TX line of the device with which you would like to communicate.

COM Port - Each device you connect to your computer will be assigned a specific port number. This helps to identify each device connected. Once a device has a port assigned to it, that port will be used every time that device is plugged into the computer. Note that Mac and Linux COM ports have a different naming convention.

TTY - TTY stands for teletypewriter or teletype. Much like terminal is synonymous with the terminals of old, so too is teletype. These were the electromechanical typewriters used to enter information to the terminal and, thus, to the mainframe. When working with terminals on Mac and Linux, you will often see TTY used to represent a communication port rather than 'COM port'.

Data, Stop, and Parity Bits - Each packet of data sent to and from the terminal has a specific format. These formats can vary, and the settings of your terminal can be adjusted accordingly to work with different packet configurations. One of the most common configurations you'll see is 8-N-1, which translates to 8 data bits, no parity bit, and one stop bit.

Flow Control - Flow control is controlling the rate at which data is sent between devices to ensure that the sender is not sending data faster than the receiver can receive the data. In most applications used throughout these tutorials, you will not need to use flow control. The flow control may also be present in the shorthand notation: 8-N-1-None, which stands for no flow control.

Carriage Return & Line Feed - Carriage return and line feed are the ASCII characters sent when you press the enter key on your keyboard. These terms have roots from the days of typewriters. Carriage return meant the carriage holding the paper would return to the starting point of that particular line. Line feed (aka new line) meant the carriage should move to the next line to prevent typing over the previous line.

When typing on a modern keyboard, these terms still apply. Every time you press enter (or return) you are telling your cursor to move down to the next line and move to the beginning of that new line.

Consulting our handy dandy ASCII table, we can see that the character for line feed is 10 (0x0A in hex) and carriage return is 13 (0x0D in hex). The importance of these two characters cannot be stressed enough. When working in a terminal window you'll often need to be aware of which of these two characters, if not both, are being used to emulate the enter key. Some devices only need one character or

the other to know that a command has been sent. More importantly, when working with microcontrollers, be aware of how you are sending data. If a string of 5 characters needs to be sent to the micro, you may need a string that can actually hold 7 characters on account of the 10 and 13 sent after every command.

Local Echo - Local echo is a setting that can be changed in either the serial terminal or the device to which you are talking, and sometimes both. This setting simply tells the terminal to print everything you type. The benefit from this is being able to see if you are in fact typing the correct commands should you encounter errors. Be aware, though, that sometimes local echo can come back to bite you. Some devices will interpret local echo as double type. For example, if you type `hello` with local echo on, the receiving device might see `hheellllloo`, which is likely not the correct command. Most devices can handle commands with or without local echo. Just be aware that this can be an issue.

Serial Port Profile (SPP) - The Serial Port Profile is a Bluetooth profile that allows for serial communication between a Bluetooth device and a host/slave device. With this profile enabled, you can connect to a Bluetooth module through a serial terminal. This can be used for configuration purposes or for communication purposes. While not exactly pertinent to this tutorial, it's still good to know about this profile if you want to use Bluetooth in a project.

Source: https://learn.sparkfun.com/tutorials/terminal-basics/basic-terminology-