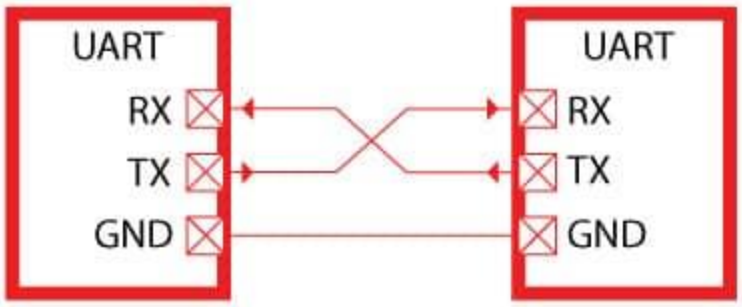
Debugging ESP8266 code

Remember, ESP8266 is a microcontroller. Meaning it operate it own. The program we downloaded to the chip will continue to run even after we unplug it from the computer and power up the unit with standalone power source.

Sometime, however it will be useful to add some breakpoints and debug the code while it is running on the unit. One simple way to do this is to add print statements and printout the status. However, there are very limited peripherals on the device (ESP8266). One good way to communicate with the device is through the serial port.

Setting up serial port on the device.

A UART (Universal Asynchronous Receiver/Transmitter) one of the oldest protocols that can use to communicate with two devices. It is simple to configure and provide moderate communication speeds.



As shown in the figure above, it uses only two pins to communicate, and it is very efficient. ESP8266 has two UARTs.

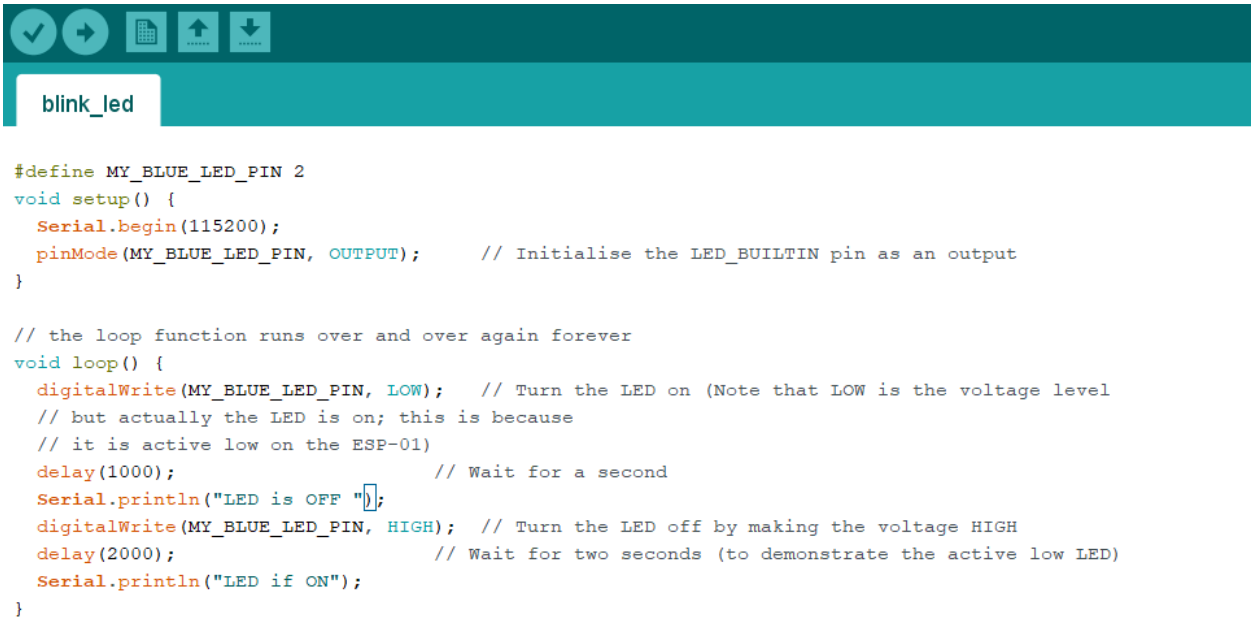
TX: This line is used to send data to the device at the other end.

RX: This line is used to receive data from the device at the other end.

However, in order to meaningfully communicate between two devices, each UART should operate at same rate. In general, UART bus can operate at different speeds (baud rates), you needs to specify what speed to receive/send data at using the baud_rate option. The most common baud rates are 9600 and 115200

ESP Arduino library provides Serial implementation, where you can print data to the serial port as human-readable ASCII text.

Let’s modify the blink LED code to print status of the LED to the serial port



Please node this code has 3 new lines

Serial.begin(115200);

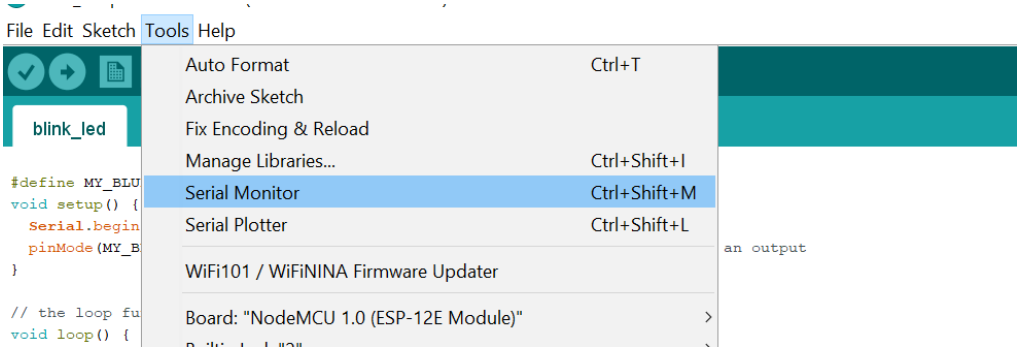
This line initiate serial port with baud rate 115200 bits/s

Serial.println("LED is OFF "); and Serial.println("LED is ON"); lines print the state of the LED to the serial console and move the curser to the next line

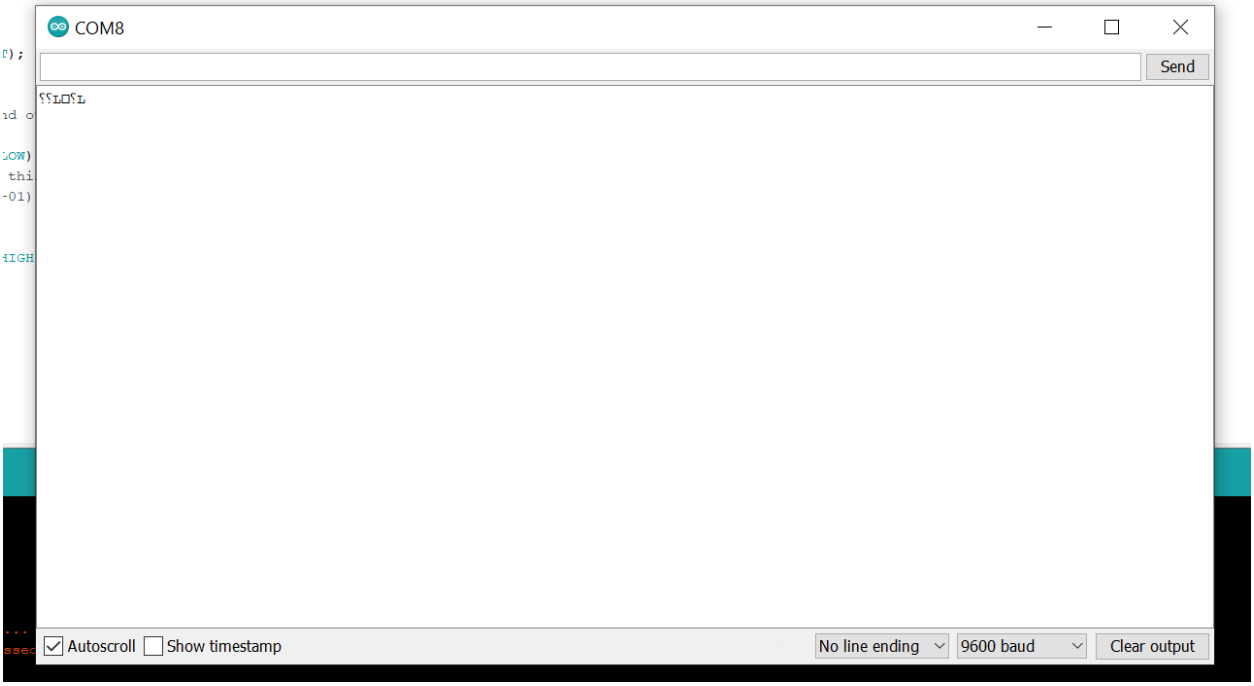
Compile the above code and update it to the device.

Now we need to open Serial Client and check the output.

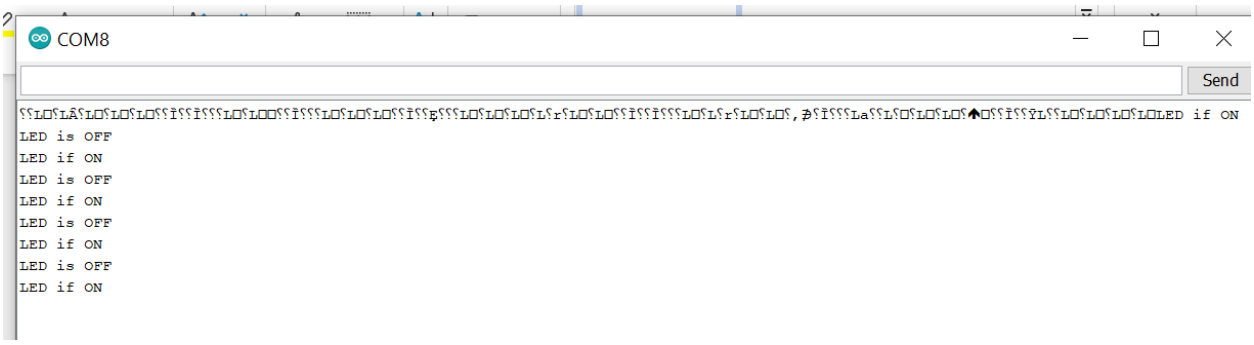
Open the Serial monitor



This will bring the serial monitor up



Sometimes you will have to select correct COM port. If you get an error, please check the available ports. As you can see in the above figure, we are not getting correct output from the device. Rason for this is the baud rate client communicate is different from the baud rate device communicate. When we initiate the device, we have set the device baud rate to 115200. However, client is communicating at 9600 baud rates. So, it does not understand the message it is receiving. Let's correct this and see we get the output we are expecting.



It did the job. Congratulations, you now know how to communicate with ESP8266.