# Practical 4: Data Communication

## Arduino to PC

The Arduino environment has a feature to receive serial characters via the USB connection. This is a very convenient option, especially to see if you are working and communication to debug. The terminal in the Arduino IDE can be opened with the icon in the upper right corner of the editor.

* Open the example "Communications / ASCIITable" and upload it.
* Try to understand what is going to do this sketch.
* Open the terminal and verify the operation of the sketch.
* The sketch uses both Serial.write (aByte) and Serial.print (aByte). Look in the output and read the comments. What is the difference?

## PC to Arduino

With the serial port the Arduino can not only transmit but also receive data. Similarly, you can also send data to the Arduino via serial monitor.

Make a sketch that reads the serial port and put a character on the LED display. You can do this by using the previously supplied use bitmap sketch or create a whole new sketch.

Then take the following steps into your Arduino IDE to receive data from your PC:

* Use Serial.begin (9600)[[1]](#footnote-0) in setup to open the port.
* Use Serial.available () and Serial.read () in a loop.
* Open the terminal in the Arduino IDE
* type a character in the edit box at the top and press enter or the "send" button.
* If the character appears on your LED-display you've done well!
* What happens when you type a word?
* What happens when you paste a whole piece of text in the box and send?

|  |  |
| --- | --- |
| **Question:** What is the speed of the data connection? How many bytes can I send per second at the set speed?   |  | | --- | |  | |

You can write custom applications that can communicate with your Arduino. Many programming languages and programming environments provide opportunities to deal with serial communication. In this exercise we use the Processing environment to send data from the PC to the Arduino.

## Processing

Processing is a Java-based programming environment that provides the same simple interface as what you're used to with the Arduino. The two have also much in common because the Arduino IDE is derived from the processing environment.

* Install processing ([www.processing.org](http://www.processing.org) )
* Download the sketch Mouse Sender from VLO.
* If you run the sketch he gives a list of available ports and their number. This line of code "System.out.println (Serial.list())" is in your setup. You can find the output of this println at the bottom of the processing environment. Look what port belongs to the Arduino and fill it in with portNr. Refer to the explanation of the serial interface of the Processing documentation. <https://www.processing.org/reference/libraries/serial/index.html>

If you have selected the correct serial port you can send data from the Processing sketch to the Arduino.

* Make sure Arduino's serial terminal is closed. A serial port can only be used by one program at the time!
* Run the sketch and move the mouse. If the mouse moves over the blue area the Arduino must display the same character as the Processing sketch.
* Take the Counter Arduino sketch from last time.
* Customize it to read the serial port and make sure it shows the same number as the Processing sketch.
* Now change in processing size (255, 255) in size**(500,**255).
* Retest the sketch. The display can show numbers up to 899, and the processing window is 500 pixels wide. What goes wrong?

|  |  |
| --- | --- |
| **Question:** How can you send numbers greater than 255?   |  | | --- | |  | |

It's no coincidence that you can display numbers up to 255 with ease but not higher. Recall the lesson from two weeks ago about data types and their limits. 255 Is the largest number that fits in one byte. Both Processing and Arduino can only communicate with one byte at a time. So what to do with larger numbers? You will have to split them up! For example, you can send 1024 as 10 and then 24, if you know the "formula".

A int consists of two bytes. The order in which you can send each byte is your own business, as long as they are received in the same order on the Arduino side.

* Divide the number by 256 to get the "high" byte
* Use the high byte and your original int to get the "low" byte
* order to reconstruct multiply you and count .
* If all goes well, you can now numbers more than 255 successful transmission.

Time left:

Try to achieve the same result, use bitwise manipulation.

* Use the operators >> (shift-right) and & (bitwise-and) to split the number into two bytes.
* Use the operators << (shift-left) and | (bitwise-or) to combine the two bytes to form one number.

|  |  |
| --- | --- |
| **Question:** We want to send both the x-coordinate of the mouse as the y-coordinate to the PC. Please describe your "communications protocol", ie which bytes are you going to send and in what order?   |  | | --- | |  | |

Now you can create a mouse pointer on your LED matrix.

* Use processing to send two numbers, an x and the y-coordinate.
* Start each (x,y) pair with a special (made-up) character so the two numbers are always identifiable in your stream.
* ensure that this number is up to 8 (using width, height and function map ()).
* Highlight one pixel on your LED matrix.

1. All serial functions are again searching for the Arduino page. Serial is not a function but an object has its own page <http://arduino.cc/en/Reference/Serial> [↑](#footnote-ref-0)