Serial communication between Arduino and Octave

Many devices connect to the computer trough serial interface. Being able to easily establish connection trough serial port communication and then also analyse, process and visually represent received or send data, can therefore be useful in many cases. [GNU Octave](http://www.gnu.org/software/octave/) is an open software that recently, with release of instrument-control package, allows for all of that, the only downside being very limited documentation and lack of provided examples to help with debugging.

We present straightforward instructions on how to establish serial communication, manage connection settings, send and receive data. A sample code, illustrating described procedures is also enclosed. In particular, we established connection with Stellaris LaunchPad running [Energia](http://energia.nu/) code which is [Arduino](http://arduino.cc/) compatible.

First, an instrument-control package must be installed. It can be found on [Sourceforge] (http://octave.sourceforge.net/instrument-control/). Open the folder containing downloaded files in Octave and in prompt execute command:

pkg install instrument-control-0.2.1.tar.gz

If installed correctly, you will not receive any message, otherwise error message will appear. By default, package is not available from prompt and must be loaded each time with the command:

```pkg load instrument-control``

You can easily check if package is loaded correctly using the flowing:

if (exist("serial") == 3) disp("Serial: Supported") else disp("Serial: Unsupported") endif

Once the package is loaded, all functions for serial communication can be accessed. All of them, together with a brief description, can be found on the [website](http://octave.sourceforge.net/instrument-control/overview.html).

In order to establish the connection, the serial port object needs to be created first:

s = serial\_setup('COM5')

where ‘COM5’ is replaced with an appropriate interface path (which can be found in the Device Manager on Windows). Default connection settings can be changed. To view the initial state use

get(s)

which will return the current settings structure. Any feature can be changed with the set function

set (s, 'property',value)

Pay special attention to ’timeout’, as the default value -1, means blocking the call. More about other options can be found [online](http://octave.sourceforge.net/instrument-control/function/@octave_serial/set.html)

To send or receive data, functions ‘srl\_read’ and ‘srl\_write’ are used:

[data, count] = srl\_read(s,n)

reads n bytes from port and write them as uint8 array in the data variable. Count represents the number of bytes successfully read. Currently this is the only form in which Octave reads from and writes to the serial port, therefore uint8 array must be converted to desired form. ‘typecast’ [This](https://www.gnu.org/software/octave/doc/interpreter/Built_002din-Data-Types.html) function can be used. For example, reading a single float:

data = srl\_read(s,4); data\_float=typecast(uint8(data), 'single');

To just convert an uint8 array to a string use char(data). Similar, when writing back to the serial port, any structure needs to be converted to either string or uint8 array. For example, if we want to write variable data\_float back:

data = typecast( single(data\_float), 'uint8'); srl\_write(s, data);

or to write a string:

srl\_write(s, 'Hello world!')

To flush the pending input use:

srl\_flush(s)

Once you are done communicating with serial port, be careful to properly close it, as otherwise it cannot be used by any other device, or reopened in Octave:

fclose(s); clear s;

Note that communication may hang in certain cases, thus ‘pause’ is used between calls where this may happen.

You can find a sample code for Energia/Arduino device and Octave, containing main and three sub-functions,in our [GitHub repository](https://github.com/IRNAS/MathFunctions/tree/master/SerialCommunicationOctaveArduino). Function serial\_setup() establishes connection and settings, serial\_red1() reads array of different type of variables, then serial\_write1() writes back changed values. Note that this is an early release of an unoptimized code which at least on the micro-controller side can be significantly optimized.