Arduino Servo Controller

Jan Krieger <j.krieger@dkfz.de> / <jan@jkrieger.de>[*]

June 5, 2012

Contents

[*]German Cancer Research Center (DKFZ), Department: Biophysics of Macromolecules (Prof. J. Langowski), Im Neuenheimer Feld 580, D-69120 Heidelberg, Germany, tel: +49-6221/42-3395

1 Introduction

This circuit controls up to four servos connected to the I/O ports of an Arduino Nano. A USB-to-serial converter allows a host PC to control the servos. The circuit is optimized to úse the servos as laser shutters, so each servo may only reside in one of two possible states (0 or 1).

2 USB Interface & Command reference

2.1 USB Interface

As mentioned before the control box contains a USB to serial converter. Thus the communication with a PC may be done by simply sending text commands over a terminal (e.g. `http://sites.google.com/site/braypp/terminal` for windows). Several commands are defined that allow to control the laser and the remote control box. Each command starts with a single character identifying it and the additional characters that give parameters. Commands with additional parameters require a closing line feed character (`0x0A`, or `\n` in C/C++). Any characters which are not recognized as commands will be ignored, so the interface usually does not stall. If a command features a return value two line feed characters (`0x0A`, or `\n` in C/C++) end the return value.

The interface has a configurable baud rate (fixed to 9600 baud), uses 8 data bits, no parity and one stop bit. The interface is implemented, using a serial to USB converter chip from FTDI. These chips register to the operating system as a virtual serial port, if an appropriate driver is installed. There are drivers for many Windows variants, Linus and MacOS 8,9 and X. Usually a Linux kernel will recognize the interface out of the box. The drivers may be downloaded and installed from `http://www.ftdichip.com/Drivers/VCP.htm`.

2.2 Command Reference

This section will give an overview over the available commands together with examples (identifiers in brackets `<...>` have to be replaced by the described data, a value in brackets like `<0xD4>` stands for the binary representation of the value, <LF> stands for a line feed character, <CR> for carriage return). Note that commands are generally case-sensitive!

command	*description*
Status Report & Configuration	

`?`	identifies the remote control box and sends back copyright and version information (multi-line output ended by two consecutive `<LF>` characters. Example output:
	`Servo Controller<LF>`
	`  (c) 07.2010 by Jan Krieger (DKFZ)<LF>`
	`  j.krieger@dkfz.de --  jan@jkrieger.de<LF><LF>`

`V`	return version information. Example output:
	`1.3.319.1735 (05.2010)<LF>`

Servo Control & Status	
`s<SERVO><STATE>`	set the given servo (1..4) to the given state (0,1)
`q<SERVO>`	returns the state (0,1) of the given servo (1..4)

3 Technical Realization

3.1 Hardware Overview (Electronics)

3.2 Implementation Details and Hints

- The FTDI USB to serial converter has areprogrammed device name

***

Still you can use the standard FTDI virtual COM port driver for this device:

`http://www.ftdichip.com/FTDrivers.htm`

- The FTDI USB to serial converter has to be programmed to be self-powered. Otherwise the serial connection does not work on Linux (on Windows system the problem does not seem to exist).
- You can bind a box to a spoecific serial port. On **Windows** this can be done in the driver settings. On **Linux** this may be achieved using the `udev` hardware enumeration system:

  1. create a udev rules file with a name like e.g.: `/etc/udev/rules.d/99-usbboxes.rules`
  2. add a line like this:

     ```
     #LASERBox
     KERNEL=="ttyUSB*", SUBSYSTEMS=="usb", ATTRS{product}=="B040SERVO", SYMLINK+="ttyUSB_B040SERVO"
     ```

     which will map the ttyUSB device with the product name B040SERVO to the device file `/dev/ttyUSB_B040SERVO` in addition to its default file `/dev/ttyUSB`*x*. Instead of the product name you can also use any other USB device property, as reported by

     ```
     lsusb -v -s <bus>:<devnum>
     ```

     Possible filters are e.g.:

     ```
      ATTRS{serial}=="A800d0B1",
      ATTRS{idVendor}=="1a72",
      ATTRS{idProduct}=="1007",
      ATTR{vendor}=="0x149a",
      ATTR{device}=="0x0005"
     ```

  3. After saving the rules file you will have to tell udev that there are new/changed rules by executing

     ```
     /etc/init.d/udev restart
     ```

     The rules will be executed when the new device is connected to the computer. So if it's already connected: pull and reconnect the plug.

4 Literature & Datasheets

- FT232RL: `http://www.ftdichip.com/Documents/DataSheets/DS_FT232R.pdf`