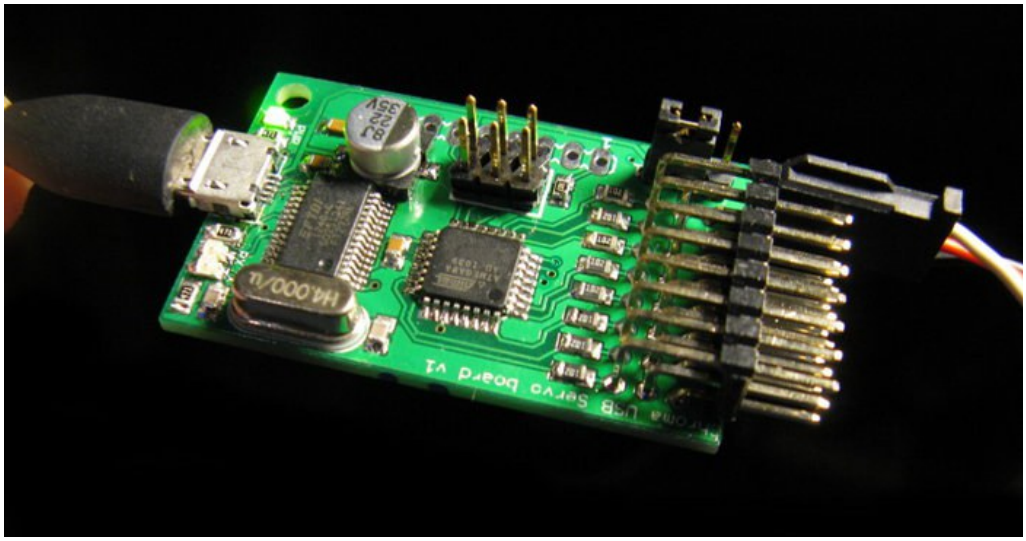
Chroma USB Servo Board v1
for
Raspberry Pi / PC / MAC

(Firmware 0.1)

2014-11-16



Content

Setup

Connecting the servo board

Connect the servo board to the computer/Raspberry Pi using a micro USB cable.

Make sure the cable is not a power only cable. Also some cell phone cables seems not to work. Be gentle and try flipping the micro USB connector if it will not fit.

How to resolve possible Windows driver problems

If you connect the board to a Windows computer and you do not get a USB serial port as you would expect, check the Device manager. If you have a device named ”FT232R USB UART” with a little yellow ”!” on it, do the following:

1. Download:
http://www.ftdichip.com/Drivers/CDM/CDM%20v2.10.00%20WHQL%20Certified.zip
and extract it somewhere.

2. Right click the ”FT232R USB UART” and select properties.
3. On the Driver tab: Select ”Update driver”.
4. Click ”Select Driver already on your computer” (Choose and install driver manually)
5. Click ”Let me choose from a list of drivers already on this computer”
6. Select ”Show all devices” (option at top of list) and ”Next”.
7. Click ”Have disk” and click ”Browse”.
8. Select the file ”ftdibus.inf” from the top level folder you created in step 1. Click ”Ok”.
9. Select ”USB Serial Converter” click ”next”.
10. The driver is not certified but it's from FTDI. Click ”Yes”. And ”Close”.
11. Close until you are back at the Device manager.
12. You should now have ”USB Serial Port” with a yellow ”!” instead.

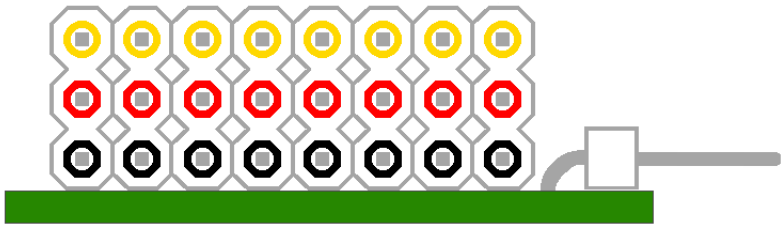
The below is very similar to the step 2-12 above! Diffrences marked in **bold!**

13. Right click the **”USB Serial Port”** and select properties.
14. On the Driver tab: Select ”Update driver”.
15. Click ”Select Driver already on your computer” (Choose and install driver manually)
16. Click ”Let me choose from a list of drivers already on this computer”
17. Select ”Show all devices” (option at top of list) and ”Next”.
18. Click ”Have disk” and click ”Browse”.
19. Select the file **”ftdiport.inf”** from the top level folder you created in step 1. Click ”Ok”.
20. Select **”USB Serial Port”** (at bottom) click ”next”.
21. The driver is not certified but it's from FTDI. Click ”Yes”. And ”Close”.
22. Close until you are back at the Device manager.

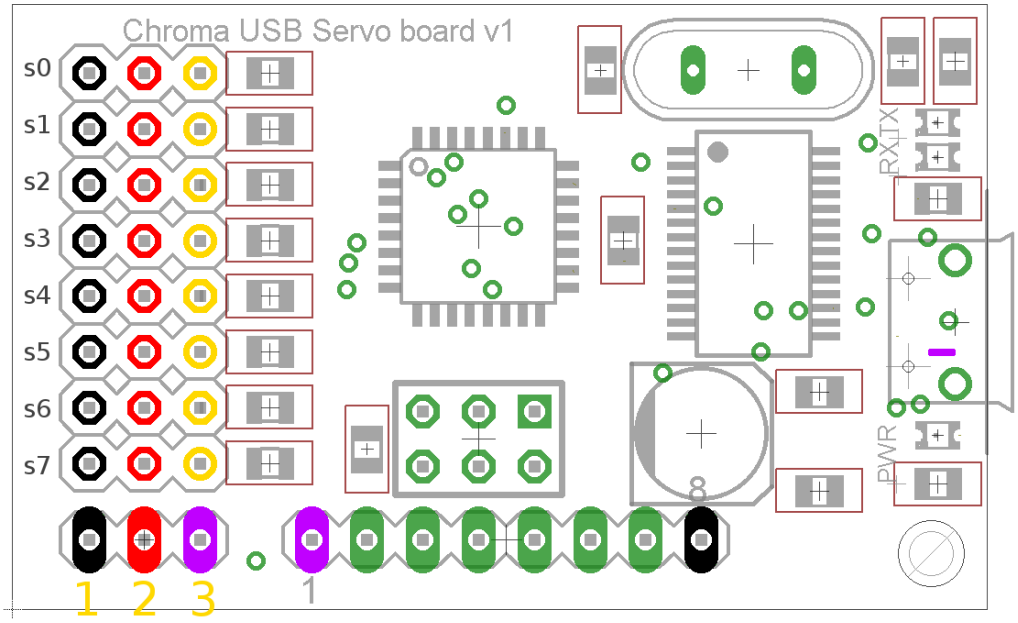
23. You should now have a working COM-port: ”USB Serial Port”.

Connecting servos

When connecting servos make sure ground wire of servo is closest to the edge of the servo board. See picture below (black – ground, red – 5V, orange – signal). Also make sure the servo connector is aligned.



Power options



If used with a Raspberry Pi there are a few ways to power the RPi and the servo board.

WARNING:

When the servo rail is powered via USB from RPi and the RPi is powered by 5V from micro-USB, the poly fuse on earlier RPi's will make the voltage drop if much current is drawn. Commanding one or more servos to move at full speed at the same time might reboot your RPi and destroy contents of SD-card. The same goes for attaching a servo in run time.

Please read and understand the chapter below especially if you plan on connecting an ESC or BEC!

1. **Power connector/jumper**

   The black-red-purple 3-pin connector/jumper below the servo-connection block on the above illustration is:

1 – black	Ground
2 – red	+5V on servos
3 – purple	+5V from USB

   With no jumper installed servo rail +5V is disconnected from USB power +5V and servos are powerless unless an ESC/BEC or battery is connected to one of the eight servo connections. C**onnecting a jumper between 1 and 2 is usually a bad idea.**

   Examples:

Power:	Servos powered from USB
Jumper:	Between 2 and 3.
Note:	This configuration on older Raspberry Pi's will make voltage drop even with small loads. RPi might reboot possibly trashing content of SD-card. Can work with one servo and no servo load. If connected to PC/MAC USB current is limited, but works well with light loads.
Power:	Servos powered via connected ESC/BEC.
Jumper:	No jumper!
Note:	Make sure jumper is removed!
Power:	Servos powered from external source (battery) 5V.
Jumper:	No jumper: Connect external 5V to pin 2 and external ground to pin 1.
Note:	Make sure jumper is removed!
Power:	RPi and servos powered via connected ESC/BEC. (BEC must be 5V stable)
Jumper:	Between 2 and 3.
Note:	Do not connect micro-USB on RPi! **For RPi only**.
Power:	RPi and servos powered from external source (battery) 5V.
Jumper:	Between 2 and 3. Connect external 5V to servo 5V, connect external ground to servo ground.
Note:	Since power is supplied via a servo-connector, a servo can not be connected there. Max 7 servos. **For RPi only.**
Power:	RPi and servos powered from external source (battery) 5V, another way.
Jumper:	Between 2 and 3. Solder external 5V to hole 1 of 8 hole connector, Solder external ground to hole 8 of 8 hole connector.
Note:	All 8 servo connectors can be used. **For RPi only.**

Getting started

How to get started to actually control servos. See the protocol section to learn more about the diffrent commands. For Raspberry Pi if you have no other FTDI USB serial ports connected the USB servo board should end up as /dev/ttyUSB0. If you connect one more board it should get /dev/ttyUSB1

To check: just connect the servo board and run dmesg in Linux.

Using shell

Set serial port to 9600 bps:

```
stty -F /dev/ttyUSB0 9600
```

Start servo test:

```
echo "st" > /dev/ttyUSB0
```

Stop servo test with any command or a wrong command:

```
echo "serr" > /dev/ttyUSB0
```

Using minicom

Install minicom (Internet connection requiered)

```
sudo apt-get install minicom
```

Start minicom in setup mode as root to configure it, only needed once:

```
sudo minicom -s
```

Select ”Serial port setup”
Press ”a” and change Serial device to /dev/ttyUSB0
Press enter
Press ”e” and choose 9600 bps by pressing ”c”.
Press enter
Press enter to exit ”serial port setup”

Select ”Screen and keyboard”
Press ”q” to enable local echo.
Press enter to exit ”Screen and keyboard”

Select ”Save setup as dfl” to save this as your default setting.

Select ”Exit from Minicom”

From now on you can start minicom and issue commands to the servo board when you like:

```
minicom
```

Just try issuing the servo test command:

```
st
```

Using Python

To use the serial port from Python, you will need the serial module:

```
sudo apt-get install python-serial
```

Sample Python script:

```
import time
import serial

s = serial.Serial("/dev/ttyUSB0",9600)

s.open()

s.write("st\n")     # Servo test command
time.sleep(5)       # Wait for 5 seconds
s.write("serr\n")   # Wrong command to stop servo test

s.close()
```

The protocol

It's a small and simple ASCII protocol. Start of command is ”s” and end of command is enter (line feed or return). Default serial port setting of servo board is 9600 8N1.

Every command will either be answered with an ”ACK” or a ”NACK” if not understood. The only exception is the firmware version command ”sn” that will answer with the string ”01” for firmware version 0.1.

The only command that makes a persistent change (remembered after power loss) is the Set Initial position All command: ”sia”.

Overview

Command	Name	Description
`st`	Servo Test	All servos move slowly between -100% and 100%
`sa`	Servo All	Change position of all 8 servos with one command
`sav`	Servo All Velocity	Change speed for all 8 servos with one command

`S0`
...
`s7`
	Servo N	Set position (and speed) for servo N.

`sia`	Servo Initial position All	Set default position for all 8 servos. Saved and used at power up.
`sbr`	Servo Bit Rate	Change serial communication speed.
`sn`	Servo versioN	Returns firmware version ”NN”. Example ”02” for version 0.2.
`se`	Servo Enable	Enable servo output (default)
`sd`	Servo Disable	Disables servo output (tristates output buffer IC)
`so`	(Servo) Output	Experimental: Controls 8 hole connectors center 6 pins.

Servo Test

Has no parameters and will move all servos in sync between -100% and 100% at about 0.2% per millisecond. Any command or a wrong command will abort.

Servo All Velocity

Set speed at which servos should move when using ”sa”-command.

Takes 0 to 8 parameters. Any parameter omitted is interpreted as 0.
Parameters is servo speed in steps of 10% per second from 1 to 255 or 0 for fastest possible speed. First parameter is speed for first servo, second parameter is speed for second servo and so on.

Examples:

```
sav 1 10 20 20
```

Sets first servo to move at slowest possible speed: 10% per second, second servo at 100% per second, third and fourth at 200% per second. Servo five to eight will move as fast as possible.

```
sav
```

Sets all servos to move as fast as possible

sav 100

First servo is set to move at 1000% per second. Remaining seven is set to move as fast as possible.

Servo All

Set position for servos

Takes 0 to 8 parameters. Any parameter omitted is interpreted as 0.
Parameters is servo position in promille from -1000 to 1000 (-2500 to 1900 is allowed for going beyond defined range). First parameter is position for first servo, second parameter is position for second servo and so on.

Examples:

```
sa 0 1000 500 -1000
```

Moves first servo to 0%, second to 100%, third to 50% and fourth to -100% the remaining 4 will be moved to 0%.

```
sa
```

All eight servos are moved to 0%

sa -1500

First servo is moved to -150% all other servos are moved to 0%.

Servo N

Move one servo to a given position at a given speed.

Takes 0 to 2 parameters. Any parameter omitted is interpreted as 0.

First parameter is servo position in promille from -1000 to 1000 (-2500 to 1900 is allowed for going beyond defined range)
Second parameters is servo speed in steps of 10% per second from 1 to 255. 0 for fastest possible.

Examples:

```
s0 1000 0
```

Will move the first servo to 100% Only physical speed of servo limits speed.

```
s0 1000
```

Will move the first servo to 100% Only physical speed of servo limits speed. The same as above

```
s1 500
```

Will move the second servo to 50% Only physical speed of servo limits speed.

```
s6 -1000 5
```

Will move the seventh servo to -100% at roughly 50% / second.

```
s2 1300 1
```

Will move the third servo to 130% at roughly 10% / second.

If the second parameter is given and is diffrent from 0, the given speed will be used in subsequent ”sa”-commands overriding any previous ”sav” commands.

Servo Initial position All

Sets default position for servos, used at powerup.

Takes 0 to 8 parameters. Any parameter omitted is interpreted as 0.
Parameters is servo position in promille from -1000 to 1000 (-2500 to 1900 is allowed for going beyond defined range). First parameter is position for first servo, second parameter is position for second servo and so on. This command will not move any servo, just save the values to be used on power up.

Examples:

```
sia 0 1000 500 -1000 -780 -50 -5 -2000
```

At every power up first servo is set to 0%, second to 100%, third to 50%, fourth to -100%, fifth to -78%, sixth to -5%, seventh to 0.5% and eight to -200%.

Servo Bit Rate

Set bit rate to be used in serial communication.

Takes 0 to 1 parameter. Any parameter omitted is interpreted as 0.
Takes a number 0..6 for bit rate according to below table:

Parameter	**Bit rate**
0	9600 (default)
1	19200
2	38400
3	N/A
4	N/A
5	N/A
6	500000

Examples:

```
sbr 2
```

Serial port speed is changed to 38400 bps. ACK is sent before changing speed.

Servo versioN

Returns firmware version.

Takes no parameters.

Example:

```
sn
```

Returns firmware version ”NN”. Example ”01” for version 0.1. Only command that will not return ACK/NACK. (” and ” not included in return string)

Servo Enable

Enables servo pulses output (default)

Takes no parameters

Example:

```
se
```

Servo Disable

Disables servo pulses output.

Takes no parameters

Example:

```
sd
```

No output pulses will be sent to servos, commands will be accepted as usual, and any changes to servo positions will be seen upon giving the ”se”-command.

This makes it possible to save power by ”disabling” servos.

(Servo) Output

Experimental – might be removed/replaced in future firmware.

Controls state of the center 6 pins of the 8 hole connector in the middle of the board. Pin 1 of the connector is 5V power from USB. Pin 2..7 is controlable output signals, either 0V or 5V max current draw is a few mA. Pin 8 is GND.

Pin 1 is the one closest to servo connectors.

Takes 0 to 6 parameters. Any parameter omitted is interpreted as 0. First parameter is state of pin 2 of 8 hole connector., second is pin 3 and so on. Parameters interpreted as such:
0 = 0V, anything else = 5V.

Examle:

```
so 1 0 0 0 0 1
```

Will set pin 2 and 7 of 8 hole connector to 5V, the rest to 0V.

```
so 1 0 1
```

Will turn pin 2 and 4 of 8 hole connector to 5V, the rest to 0V.

```
so
```

Will turn all 6 signal pins of 8 hole connector to 0V.

```
so 1 1 1 1 1 1
```

Will turn all 6 signal pins of 8 hole connector to 5V.

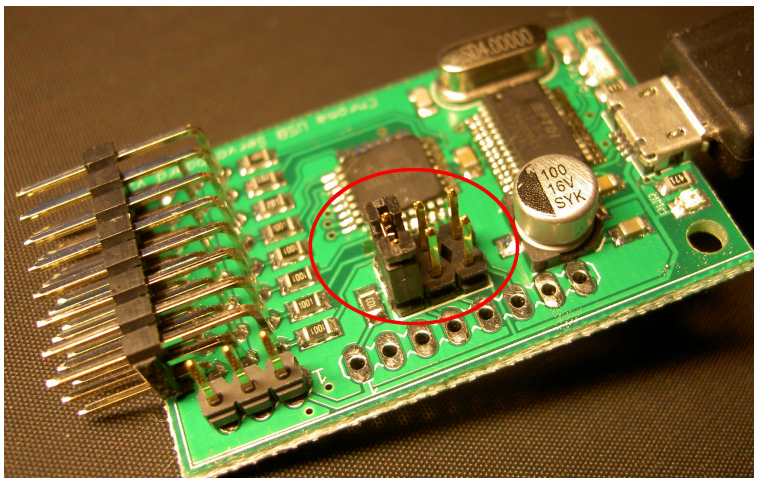
```
so 12 77 0 123 33 1
```

Will turn all but pin 4 of 8 hole connector to 5V.

Installing new firmware

Boards shipped after 2014-11-16 has optiboot installed which allows for easy installation of new firmware without tools.

1. Disconnect all servos/ESC's
2. Download the new firmware to the Pi/PC/Mac (or compile your own). Let's assume it's named: main.hex
3. Install Avrdude on your Pi/PC/Mac (In Windows it's part of WinAVR)



4. Put a jumper on the servo boards ISP-connector shorting pin 5 and 6. (You can borrow the power jumper that came with the board). See picture.

5. Now type the following as one line on your Pi/PC/Mac:
   (Replace COMn with the COM-port your USB servo board has. For RPi it's typically: `/dev/ttyUSB0`)

   ```
   avrdude -c arduino -p atmega8 -P COMn -b 19200
   -U flash:w:main.hex
   ```

6. Hit enter and quickly remove the jumper (in that order).

If it fails, repeat steps 4 to 6.

Changes

Version 0.1

- First version.

Specification

Size: 44.5 mm x 27.5 mm

Total height:10 mm