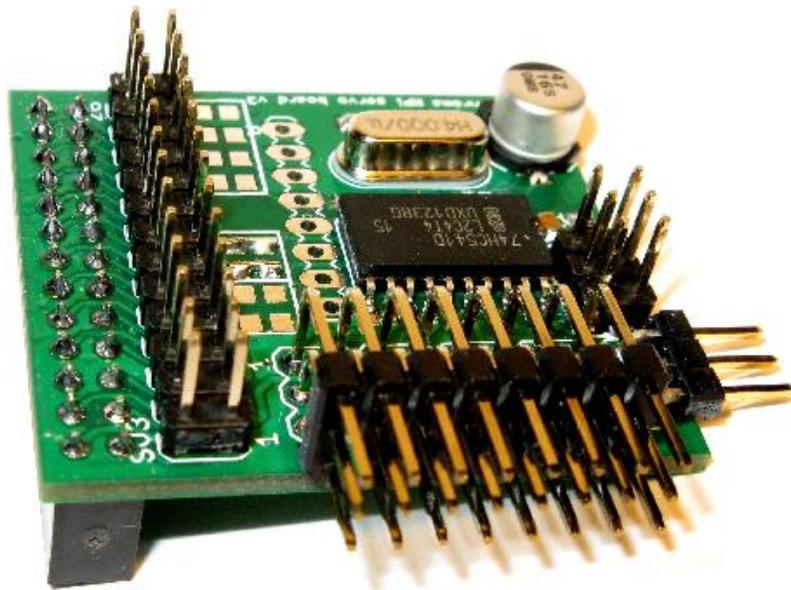


Chroma Servo Board v3
for
ODROID-C1
(and Raspberry Pi)

(Firmware 0.1 and 0.2)

2015-01-17

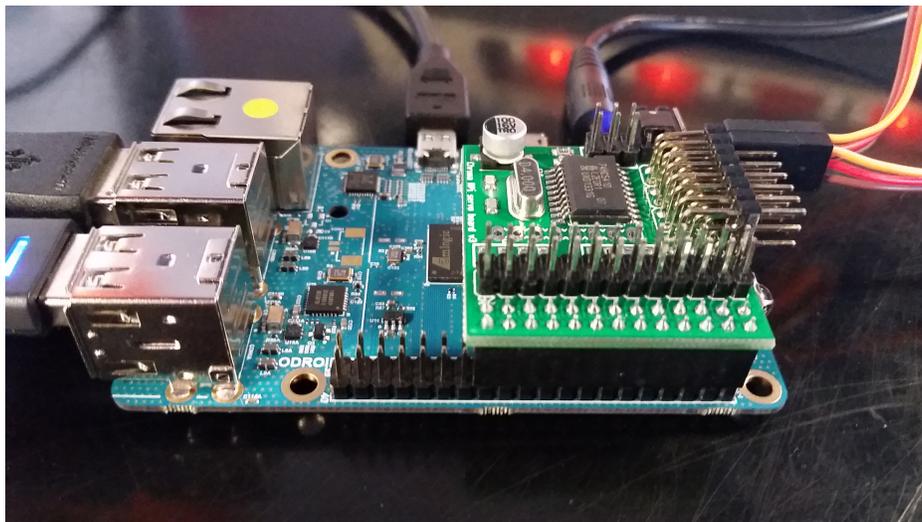


Setup for ODROID-C1

Section "Connecting Servos" apply to both the ODROID-C1 and the Raspberry Pi. "Power Options" below MIGHT also apply to both. **The variants where the ODROID-C1 is fed with power from the electronic speed controller or any other way where power is fed into the GPIO port has not been tested with the ODROID-C1.**

Connecting the servo board

Be very careful not to misalign the board. It should go all the way to the left.



Do not connect one servo board on top of another!

Raspberry Pi Setup

Before connecting the servo board

Before connecting the servo board to your Raspberry Pi you will need to disable the boot console output to the serial port and disable the default serial login console. If you don't do this before connecting the servo board, your Raspberry Pi might not boot properly.

Disable boot time output

Edit the file: `/boot/cmdline.txt` Remove the following text:

```
console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
```

You can use the command: `sudo nano /boot/cmdline.txt`

Disable serial console

Edit the file: `/etc/inittab` Remove the following line:

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

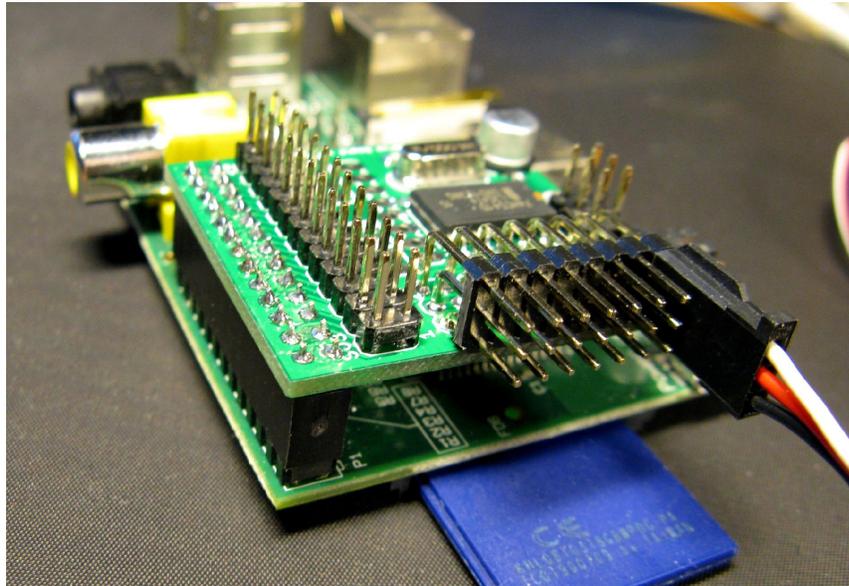
You can use the command: `sudo nano /etc/inittab`

Before connecting the servo board, shutdown RPi and remove power!

In all the examples in this guide, change all occurrences of “`ttyS2`” to “`ttyAMA0`” for the Raspberry Pi.

Connecting the servo board to Raspberry Pi

The servo board should be mounted above the Raspberry Pi, as seen on the picture below.

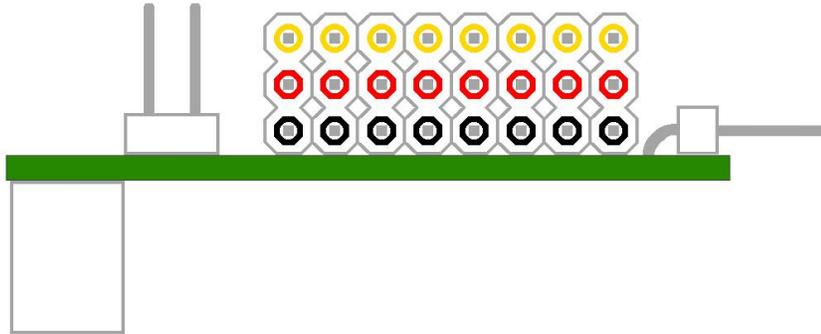


Make sure the the 2x13 female header on the servo board align in both directions with the 2x13 pin header on the Raspberry Pi.

Do not connect one servo board on top of another!

Connecting servos

When connecting servos make sure ground wire of the 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.



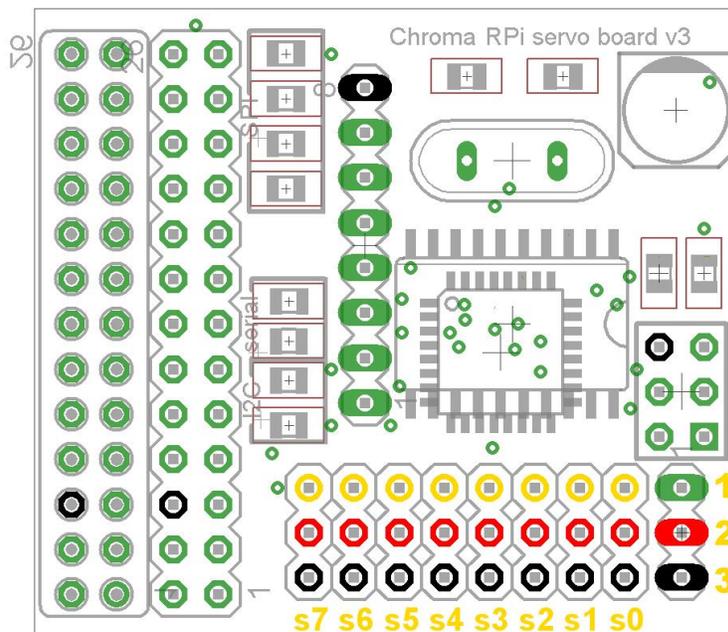
Power options

There are many ways to power the servo board.

WARNING for Raspberry Pi:

When the servo board is powered by the RPi and the RPi is powered by 5V from micro-USB, the poly fuse (On early 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. These issues do not seem to be a factor for the ODROID-C1.

Power connector/jumper



The green-red-black 3-pin connector/jumper to the right of the servo-connection block on the above illustration is:

- +5V on Raspberry Pi
- +5V on servos
- Ground

With no jumper installed servo +5V is disconnected from Raspberry Pi +5V and servos are powerless unless an ESC/BEC or battery is connected to one of the eight servo connections.

Examples:

Power: RPi and servo powered from micro-USB connector.

Jumper: Between 1 and 2.

Note: Since power from micro-USB connector goes via poly fuse (on early RPi's), voltage will drop even with modest load. RPi might reboot possibly trashing content of SD-card. Can work with one servo and no load.

Power: RPi powered via micro-USB, Servos powered via connected ESC/BEC.

Jumper: No jumper!

Note: RPi is driven from micro-USB, servos are driven from ESC's BEC.

Power: RPi and servos powered via connected ESC/BEC. (BEC must be 5V stable)

Jumper: Between 1 and 2.

Note: Do not connect micro-USB!

Power: RPi powered via micro-USB, servos from external source (battery) 5V.

Jumper: No jumper: Connect external 5V to pin 2 and external ground to pin 3.

Note:

Power: RPi and servos powered from external source (battery) 5V.

Jumper: Between 1 and 2.

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.

Power: RPi and servos powered from external source (battery) 5V, another way.

Jumper: Between 1 and 2.

Connect external 5V to pin 2 of 2x13 pin header, connect external ground to pin 6 of 2x13 pin header.

Note: All 8 servo connectors can be used.

Getting started

How to get started to actually control servos. See the protocol section to learn more about the different commands.

Using shell

Set serial port to 9600 bps:

```
stty -F /dev/ttyS2 9600 (/dev/ttyAMA0 on RPi)
```

Start servo test:

```
echo "st" > /dev/ttyS2
```

Stop servo test with any command or a wrong command:

```
echo "serr" > /dev/ttyS2
```

Using minicom

Install minicom (Internet connection required)

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

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

```
sudo minicom -s
```

Select "Serial port setup"

Press "a" and change Serial device to /dev/ttyS2

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 dfi" 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/ttyS2",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 versionN	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 indicate servo position 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 from -1000 to 1000 (-2500 to 1900 is allowed for going beyond defined range)

Second parameter indicates 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 different 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

500000 bps does not seem to work with Raspberry Pi uart.

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 "02" for version 0.2. 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. (by tristating buffer IC 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.

New commands for Firmware version 0.2

(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 3.3V power from Raspberry Pi. Pin 2..7 is controllable output signals, either 0V or 3.3V 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 = 3.3V.

Example:

so 1 0 0 0 0 1

Will set pin 2 and 7 of 8 hole connector to 3.3V, the rest to 0V.

so 1 0 1

Will turn pin 2 and 4 of 8 hole connector to 3.3V, 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 3.3V.

so 12 77 0 123 33 1

Will turn all but pin 4 of 8 hole connector to 3.3V.

Specification

Size

PCB Size: 39.6 mm x 35.7 mm

Total Size: 43.8 mm x 42.5 mm

Total height: 18.5 mm

Electrical

(No servos connected.)

3.3V average current consumption: 3 mA

5V current consumption: <1 mA

Servo rail maximum continuous current: 4A

Pinout

All pins in the 26 pin female header are passed through to the 26 pin male header. The servo board is only internally connected to the following pins within 26 pin header:

1. 3.3V
2. 5V
6. Ground
8. Servo board RX (This is C1/RPi TX pin)
10. Servo board TX (This is C1/RPi RX pin)

If the two unmounted 0805-resistors (0 ohm) in the I2C-square is mounted, the following pins will also be connected:

3. Raspberry Pi SDA0 (SDA1 on newer RPi's)
Connected to Pin 1 in 8-hole connector and to ATmega8 SDA line (pin 27 of MCU)
5. Raspberry Pi SCL0 (SCL1 on newer RPi's)
Connected to Pin 2 in 8-hole connector and to ATmega8 SCL line (pin 28 of MCU)

If the four unmounted 0805-resistors (0 ohm) in the SPI-square is mounted, the following pins will also be connected:

- 19. Raspberry Pi SPI0 MOSI
Connected to Pin 4 in 8-hole connector
and to ATmega8 MOSI pin (pin 15 of MCU)

- 21. Raspberry Pi SPI0 MISO
Connected to Pin 5 in 8-hole connector
and to ATmega8 MISO pin (pin 16 of MCU)

- 23. Raspberry Pi SPI0 SCLK
Connected to Pin 6 in 8-hole connector and to
ATmega8 SCK pin (pin 17 of MCU)

- 24. Raspberry Pi SPI0 CE
Connected to Pin 3 in 8-hole connector and to
ATmega8 SS pin (pin 14 of MCU)

Hints for hacking

There are no i2C pull -up resistors on the servo board. The Raspberry Pi already has 1.8 kohm pull-up resistors.

If running the board WITHOUT a C1/RPi and only having access to one voltage level, pin 1 (3.3V) and pin 2 (5V) of the 26 pin header (male or female) can be connected together and the board can then be fed with a single voltage between 3.3V and 5V.