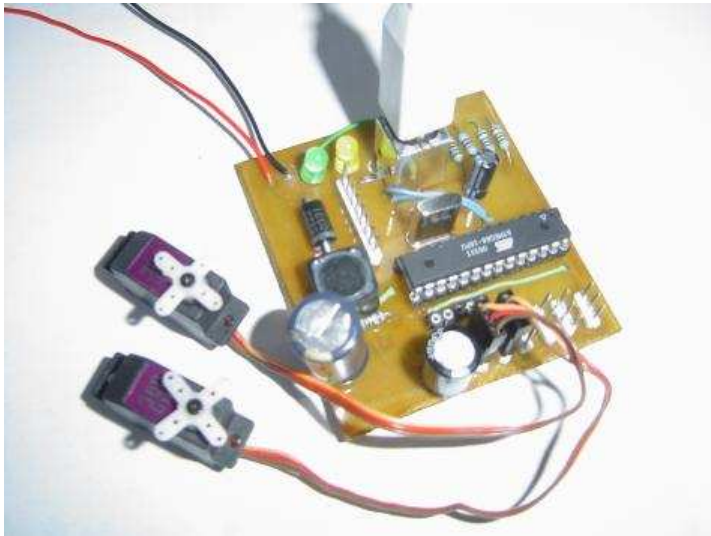


# USB servo controller

USB-servo-2. (aka RobotController)

(a continuation of [USB-Servo \(aka Mousepointerpointer\)](#))



## General.

This device was designed to control standard hobby radio control servos via a PC's USB port.

Standard RC servos need a power supply of between 4.8 and 6 volts. They also have a Pulse Width Modulation (PWM) signal input which controls the angle of the servo. This device supplies up to 6 such servos with a 5V supply and the PWM signal to control the servo. The combined load of your servos should not exceed 3 amps. Check your servo's documentation for pin out and max current draw.

It takes an external power supply of anywhere between 7 and 30 Volts to power the servos. The on-board AVR microcontroller is powered directly from the USB port.

I built it to control servos (and PWM motor controllers) on my latest robot. It should work with any RC servo or other device designed to work with radio control receiver's outputs.

The firmware for the AVR microcontroller is written for the AVR-GCC compiler and as such should compile under either Windows or Linux (but is untested under windows so far).

The device currently has a Linux application program to control the device. There is no reason that this device should not work under windows other than a controlling application or driver has not been written yet.

If anyone has experience programming AVR's in a windows environment or knows how to write a windows or Mac driver for this USB controller circuit please contact me so we can document this for others.

Note that this device has not been tested on a wide range of servos or computers. No warranty or support is offered in any way. It is up to the user to research their choice of hardware and decide whether they are safe. In short, if this destroys your computer and/or servo, don't blame me. On the other hand, mine works fine thank you.

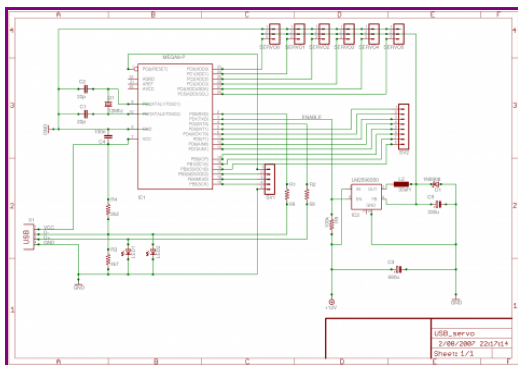
---

## Building and Installing.

### 1. Circuit.

The circuit diagram was created using the [Freeware](#) version of CadSoft's [EAGLE Layout Editor](#).

The schematic can be found in the subdirectory "circuit".



[\[click for fullsize image\]](#)

There is an image in the "circuit" directory for making your own PCB but there is no reason why it couldn't be created on stripboard as well.

Part	Value	Device	Notes	Digi-Key PN	Farnell PN
C1	22pF	Capacitor	Ceramic capacitor		
C2	22pF	Capacitor	Ceramic capacitor		
C3	680uF	Capacitor	Aluminium Electrolytic		9692304
C4	100nF	Capacitor			
C5	330uF	Capacitor	Aluminium Electrolytic		9451307
D1	SR503-T or SB530/54	Shotkey diode	Rated at 20V or greater, 5A or greater	SR503DICT-ND or SB530/54GICT-ND	
IC1	MEGA8-P	AVR ATMEGA8-P			9171380
IC2	LM2596S50	Switching regulator IC			952760 (surface mount version)
LED1		LED			
LED2		LED			
Q1	12Mhz	Cristal			9712950
R1	68k?	Resistor			
R2	68k?	Resistor			
R3	4k7?	Resistor			
R4	2k2?	Resistor			
R5	100k?	Resistor			
X1	USB connector				
L2	33uH	Inductor	Rated at 33uH, 3A or greater		7430396 (surface mount version)

## 2. Firmware.

The firmware for this project requires `avr-gcc` and `avr-libc` (a C-library for the AVR controller). Please read the instructions at [http://www.nongnu.org/avr-libc/user-manual/install\\_tools.html](http://www.nongnu.org/avr-libc/user-manual/install_tools.html) for how to install the GNU toolchain (`avr-gcc`, assembler, linker etc.) and `avr-libc`.

Once you have the GNU toolchain for AVR microcontrollers installed, you can run "make" in the subdirectory "firmware". You may have to edit the Makefile to use your preferred downloader with "make program".

If working with a brand-new controller, you may have to set the fuse-bits to use the external crystal:

```
make fuse
```

Afterwards, you can compile and flash to the device:

```
make flash
```

If you are new to AVR programming, there is a good starting point here:

<http://www.instructables.com/id/E5H5UDWB5UEUKIKV8V?ALLSTEPS>

### 3. Commandline client.

The command line tool requires libusb. Please take the packages from your system's distribution or download libusb from <http://libusb.sourceforge.net/> and install it before you compile. Change to directory "commandline", check the Makefile and edit the settings if required and type

```
make
```

This will build the Unix executable "usb-servo" which can be used to control the device.

---

#### Usage:

Connect the device to the USB-port. If it isn't already, the servos will all move to the 0-position.

Then use the commandline-client as follows:

```
# usb-servo status <servo>
```

```
# usb-servo set <servo> <angle>
```

```
# usb-servo test
```

Parameters:

- servo: The servo to control. 0 is the first servo, 1 is the 2nd, etc. (6 to 13 turn the header pins SV1 on or off.)
- angle: The angle you want to set the servo to. 0 is full left, 255 is full right. (For channels 6 to 13, '0' turns pin off. Any other value turns pin on.)

---

#### Examples.

##### Set a new angle on a servo:

```
# usb-servo set 4 23
```

This sets the servo number 0 to the angle 23. 4 is full left, 255 is full right, so with 23 the servo will be almost on the left side.

##### Set a pin on header SV1.

```
# usb-servo set 9 0
```

This sets the 4th pin to "off".

```
# usb-servo set 9 1
```

This sets the 4th pin to "on". (Any value greater than 0 sets pin to on.)

#### **Query the last servo set:**

```
# usb-servo status
```

This will tell you the angle the servo is currently put to.

```
Current servo angle: 42
```

---

#### **Thanks for the Code Guys!**

This module is based on the following 3 projects.

I'd like to take this opportunity to thank all the contributors to these projects.

I had the easy job here of copying and pasting their code. Thanks guys! Rocking!

[AVR USB](#) A Firmware-Only USB Driver for Atmel AVR Microcontrollers.

[USB-Servo \(aka Mousepointerpointer\)](#) The USB-Servo is a device to control a servo via USB by Ronald Schaten.

[Procyon AVRlib](#) C-Language Function Library for Atmel AVR Processors.

Ronald Schaten's USB-Servo project is based on the AVR USB project.

I basically took Ronald's project and in the AVR firmware replaced his single servo timing routine with the more robust AVRlib servo control routines allowing for control of more than one servo and better PWM timing.

In the PC based control program i then added options for more than one servo and a few checks to make sure the USB-Servo code has not timed out and repeat the command if necessary

The only real change i made to the circuit was to add a power source independent from the USB port to prevent too much current draw from the USB port.

---

#### **ToDo.**

The header pins SV1 are output only. I intend to make them configurable so you can read inputs on them as well.

There is still no way to control the "enable" line from the AVR to the voltage regulator. This will eventually allow all the servos to be powered down when not in use for power saving.

---

#### **About the Licence.**

My work, as i have already covered in the previous section, is based on the work of others. Any changes i have made are licensed under the GNU General Public License (GPL).

The others work, except for the USB driver, are licensed under the GNU General Public License (GPL). A copy of the GPL is included in License.txt. The USB driver itself is licensed under a special license by Objective Development. See firmware/usbdrv/License.txt for further info.

---

#### Contact info.

I'm not particularly interested in providing support for this project but you can always try: [mrdunk\(at\)gmail.com](mailto:mrdunk(at)gmail.com)

Or alternatively try on <http://www.societyofrobots.com/robotforum/index.php>