



[Welcome \(/index_en.php\)](#)
 [Software projects \(/programmation/programmation_en.php\)](#)
[Electronic projects \(/electronique/electronique_en.php\)](#)
[Design \(/design/index_en.php\)](#)
[Miscellaneous \(/divers/divers_en.php\)](#)
[Contact \(/contact/contact_en.php\)](#)
[Links \(/liens/liens_en.php\)](#)
[Shop \(https://www.raphnet-tech.com\)](#)

 (/electr
/snes_
/index.
 (/electr
/snes_
/index.

SNES/NES gamepad (and mouse) to USB adapter

Contents

 [@raphnetlabs \(https://twitter.com/raphnetlabs/\)](https://twitter.com/raphnetlabs/)

- [The idea \(#0\)](#)
- [News \(#1\)](#)
- [Solution overview \(#2\)](#)
- [Schematic \(#3\)](#)
- [Famicom \(#4\)](#)
- [Programming the microcontroller \(#5\)](#)
- [Source code \(#6\)](#)
- [Printed Circuit Board for surface-mount version \(#7\)](#)
- [Assembly of the surface-mount version \(#8\)](#)
- [Pictures and screenshots \(#9\)](#)
- [Pictures from users \(#10\)](#)
- [Disclaimer \(#11\)](#)

The idea




When playing NES and SNES games with an emulator, nothing is better than using the original gamepads (and mouse).

Many people (including me) have been happy doing this with homebuilt **parallel port adapters** (http://www.raphnet.net/electronique/snes_adaptor/snes_adaptor_en.php) for years but unfortunately, parallel ports are becoming less common these days.

USB is now the way to go for such devices so I decided to build my own NES/SNES gamepad to USB adapter. However, due to the way USB works, the adapter is a little more complicated.



Note: There is now a new version which supports up to **4 NES and/or SNES controllers** ([../4snes4snes/index_en.php](http://www.raphnet.net/electronique/snes4snes/index_en.php)) at the same time.



Ready-to-use adapters available!

(http://www.raphnet-tech.com/products/snes_to_usb_cable/index.php)

[↑ \(#home\)](#)

News

- **17 Novembre 2011:**
New firmware release (version 1.9) supporting the Famicom controller with built-in microphone.
- **Mar 7, 2007:** New schematic rev.D available. Fixes a mistake in revision C (It's PD1 and PD0 that must be connected together, not PD1 and PD2!)
- **Dec 3, 2006:**
New firmware v1.4:
 - Fixed a problem in NES mode which prevented Windows from detecting the board. That's what happens when you test only in Linux. Lesson learned...
 - Added support for Atari style joysticks. More details on [a separate page \(../atari_usb/index_en.php\)](#).
- **Oct 29, 2006:**
 - New firmware v1.3: Fix a problem that occurred when no controller was connected to the circuit, better NES controller support and PCB rev.C support.
 - New PCB Rev C: Diodes now have a place to be soldered and the board is a little smaller.
- **Oct 2, 2006:** New firmware v1.2 which adds Snes mouse support. In snes mode, the mouse is auto-detected. The 3 mouse sensitivity levels are selectable by holding either the left, right or both buttons when connecting the usb cable.
- **Jul 12, 2006:** I'd like to see how others build this project. Send me your pictures and I will put them in the [Pictures from users \(#user\)](#) section. No pictures yet...
- **Jul 11, 2006:** Added screenshots and released firmware version 1.1, which behaves better in Windows and MacOS X.

[↑ \(#home\)](#)

Solution overview

I built my adapter using an ATmega8 microcontroller from Atmel. This microcontroller does not support USB in hardware so I used the software-only usb driver from [Objective Development \(http://www.obdev.at/products/avrusb/index.html\)](http://www.obdev.at/products/avrusb/index.html). This driver allows an AVR microcontroller such as the ATmega8 to talk USB with minimal external components. As a result, the interface can be built cheaply and easily, thanks to the low component count.

[\(1snesusb_component1.jpg\)](#) Depending on your skills, you may build the interface using breadboard and thru-hole components or the surface mount version, using the PCB artwork I provide. I also sell pre-assembled PCBs, and pre-programmed Atmega8 (dip package only).



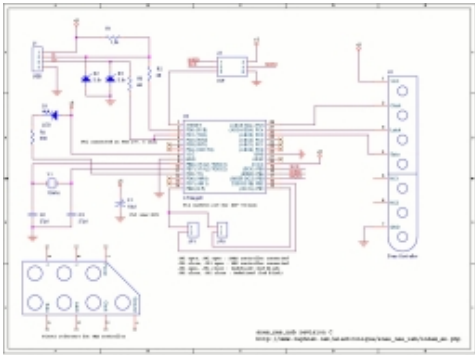
No drivers required!

That's right, since the USB standard defines device classes. I'm using the human input device (HID) which allows me to tell to the computer that the connected USB device is a joystick and has 2 axis and 4 or 8 buttons. Another nice thing about this is that the adapter should work with all operating systems supporting HID devices. (I tested and it works at least on Win98, Win2K, WinXP and Linux)

[↑ \(#home\)](#)

Schematic

Here is the schematic:



(sch-revD.png)

A pdf version is also available (easier to look at and prints nicely):

[sch-revD.pdf](#) (sch-revD.pdf)

Component list:

- **U2:** Atmega8 microcontroller. ATMEGA8-16PC, ATMEGA8-16PI, ATMEGA8-16PJ or ATMEGA8-16PU. Dont use an ATMEGA8L-*, the 12Mhz clock would be too high.
- **R1:** 1.5k resistor. Ordinary carbon film 1/4 watt resistors will do.
- **R2, R3:** 68 ohm resistors. Ordinary carbon film 1/4 watt resistors will do.
- **D2, D3:** 3.6 volts zener diodes.
- **R4:** Do not install, not used anymore.
- **D1:** Do not install, not used anymore.
- **Y1:** 12 Mhz crystal.
- **C2, C3:** 27 pf capacitors. If the crystal datasheet recommends another value, use it instead.
- **C1:** 10uf capacitor. Install it near the ATmega8.
- **JP1, JP2:** Jumpers. You can also use dip switches, ordinary switchs or solder bridges.
- **J2:** 6 pin header, 2.54mm spacing. Needed for programming the ATmega8.

For the USB connection, just strip the USB cable and solder the wires directly to the board. USB uses standard wire colors:

Color	Description
Red	+5 volts
Black	Ground
Green	D+
White	D-

Previous schematic revisions:
[schematic-revA.png](#) (schematic-revA.png)
[schematic-revA.pdf](#) (schematic-revA.pdf)
[sch-revB.png](#) (sch-revB.png)
[sch-revB.pdf](#) (sch-revB.pdf)

[↑ \(#home\)](#)

Famicom



(famicom_a_usb21.jpeg)

Famicom II to USB

Since release 1.9, the famicom controllers are supported. Well in fact, no changes were necessary for controller 1 since it is functionally identical to the NES version. Controller 2 however, is a different story. While it does not have start and select buttons, it adds a microphone! As requested by a customer, I implemented a new firmware for using this microphone as a butotn.

The lack of start and select buttons is not a problem (excluding that those buttons can be useful!), but the microphone

required an additional input because it uses a separate wire.

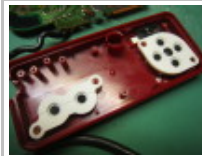
First the clock, data and latch signals are wired to the circuit as you would a standard NES controller. Next the microphone output must be wired to the MCU PC2 pin. Then, to enable the special NES-Famicom microphone mode, PC1 and PC0 must be tied together.

Here is an example of a Famicom controller 2 converted to USB. Not much space is available in the enclosure (less than in a NES controller)... Some plastic had to be cut away for the PCB to fit inside.



(famicon_a_usb03.jpeg)

Inside



(famicon_a_usb06.jpeg)

Inside



(famicon_a_usb07.jpeg)

Inside



(famicon_a_usb08.jpeg)

Development



(famicon_a_usb13.jpeg)

Does not fit



(famicon_a_usb14.jpeg)

Fits



(famicon_a_usb18.jpeg)

Fully wired.

I found a full Famicom schematic on this japanese website. The controllers are on page 1, lower left.

<http://green.ap.teacup.com/junker/116.html> (<http://green.ap.teacup.com/junker/116.html>)

Here is the color code for controller 2, from the schematic and confirmed in real life:

Pin	Color	Function
1	Brown	Microphone output
2	Red	Gnd
3	Orange	Data
4	Yellow	Latch
5	White	Clock
6	Blue	5 volt supply

[↑ \(#home\)](#)

Programming the microcontroller

A microcontroller is a component which must be programmed in order to do something useful. So here is the hexfile which must be uploaded to the microcontroller:

File	Comments
nes_snes_db9_usb-1.9.hex (releases/nes_snes_db9_usb-1.9.hex)	<ul style="list-style-type: none"> • Special features added to support the Famicom controller with microphone. Wire the 'jumpers' in NES mode and short PC0 and PC1 together to use this new mode. Wire the extra input (PC2) for the microphone digital signal from the controller. • 3 button mappings implemented for the abovementioned Famicom mode. • Fixed the device name. (It would be corrupted or would not appear since the last few releases)
nes_snes_db9_usb-1.8.hex (releases/nes_snes_db9_usb-1.8.hex)	<ul style="list-style-type: none"> • TGFX has its own device name and VID/PID. • Added Sega Paddle support (HPD-200). Auto-detected in DB9 mode. If the circuit is installed inside the controller, the two buttons can be wired independently. • Controller polling synchronized with USB polls. Helps maintain an accurate timing when polling the controllers.
nes_snes_db9_usb-1.7.hex (releases/nes_snes_db9_usb-1.7.hex)	<ul style="list-style-type: none"> • Improved Genesis 6 button auto-detection • Mode button now usable on Genesis 6 button controllers • Sega multi-tap support (tested with MK-1654 only!) (technical info) (.../divers/documentation/genesis_multitap.txt) • Added TGFX controller support
nes_snes_db9_usb-1.6.hex (releases/nes_snes_db9_usb-1.6.hex)	<ul style="list-style-type: none"> • Changed the button IDs. Atari and SMS controllers were using button 2 and 3. Some emulators require button 1 so.... • Added a way to force 6 Button mode for genesis controllers. Just in case your 6 button controller is not detected. Hold 'start' when connecting the USB cable to use this feature.
nes_snes_db9_usb-1.5.hex (releases/nes_snes_db9_usb-1.5.hex)	<ul style="list-style-type: none"> • Added support for Genesis 3 and 6 button controllers. • All DB9 controllers (Genesis 3/6, SMS, Atari style 1 or 2 button and compatible) are auto-detected. • The wiring is now different for DB9 controllers. Easier to wire to the multiuse PCB. • NES controllers are now detected in SNES mode. This works great with original Nintendo controllers. If you have trouble with clones, use NES-only mode. • The project is now released under the GPL v2 license
snes_nes_atari_usb-1.4.hex (releases/snes_nes_atari_usb-1.4.hex)	<ul style="list-style-type: none"> • Added support for Atari style joysticks (Atari, Commodore, etc). Two button variations (eg: Sega Master System) are also supported. • Added padding to the nes report descriptor. This makes it work correctly on Windows. Linux sure is

	tolerant...
snes_nes_usb-1.3.hex (releases/snes_nes_usb-1.3.hex)	<ul style="list-style-type: none"> • Fixed a bug which resulted in random button toggling when no controller was connected to the circuit. • NES mode now has it's own USB product ID. • PD1 is left in input (and no pull-up) since PD1 and PD0 must be shorted together in PCB rev.C (cleaner workaround than throwing away a batch of 50 PCBs and easier than cutting a track and soldering a wire...)
snes_nes_usb-1.2.hex (releases/snes_nes_usb-1.2.hex)	Added Snes mouse support.
snes_nes_usb-1.1.hex (releases/snes_nes_usb-1.1.hex)	Reworked the report descriptor for better behaviour under Windows and MacOS X
snes_nes_usb-1.0.hex (releases/snes_nes_usb-1.0.hex)	Initial release.

Many microcontrollers have what is called 'Fuse bytes'. In the case of the ATmega8, there are two bytes: The high byte, and the low byte. Those bytes are used to configure some aspects of the microcontroller. What type of clock to use? Crystal? Resonator? Internal RC clock? Allow programming via ISP? It's very important to set the fuses to the right values. Using the wrong values can render your MCU unusable.

For this project, here are the appropriate fuse values:
high byte = **0xc9**, low byte = **0x9f**

For details about how to program an AVR, visit my [AVR programming](#) ([../divers/avrprogindex_en.php](#)) page.



[↑ \(#home\)](#)

Source code

For those who wish to modify the device behaviour or add support for new type of gamepads, here is the source code. Starting with version 1.5, it is released under the GPL v2 license. Previous version were released under the Objective Development license, which is basically GPL + extensions to cover hardware. See License.txt for more information.

File	Comments
nes_snes_db9_usb-1.9.tar.gz (releases/nes_snes_db9_usb-1.9.tar.gz)	<ul style="list-style-type: none"> • Special features added to support the Famicom controller with microphone. Wire the 'jumpers' in NES mode and short PC0 and PC1 together to use this new mode. Wire the extra input (PC2) for the microphone digital signal from the controller. • 3 button mappings implemented for the abovementionned Famicom mode. • Fixed the device name. (It would be corrupted or would not appear since the last few releases)

nes_snes_db9_usb-1.8.tar.gz (releases/nes_snes_db9_usb-1.8.tar.gz)	<ul style="list-style-type: none"> • TGFX has its own device name and VID/PID. • Added Sega Paddle support (HPD-200). Auto-detected in DB9 mode. If the circuit is installed inside the controller, the two buttons can be wired independently. • Controller polling synchronized with USB polls. Helps maintain an accurate timing when polling the controllers.
nes_snes_db9_usb-1.7.tar.gz (releases/nes_snes_db9_usb-1.7.tar.gz)	<ul style="list-style-type: none"> • Improved Genesis 6 button auto-detection • Mode button now usable on Genesis 6 button controllers • Sega multi-tap support (tested with MK-1654 only!) (technical info) (../divers/documentation/genesis_multitap.txt) • Added TGFX controller support
nes_snes_db9_usb-1.6.tar.gz (releases/nes_snes_db9_usb-1.6.tar.gz)	<ul style="list-style-type: none"> • Changed the button IDs. Atari and SMS controllers were using button 2 and 3. Some emulators require button 1 so.... • Added a way to force 6 Button mode for genesis controllers. Just in case your 6 button controller is not detected. Hold 'start' when connecting the USB cable to use this feature.
nes_snes_db9_usb-1.5.tar.gz (releases/nes_snes_db9_usb-1.5.tar.gz)	<ul style="list-style-type: none"> • Added support for Genesis 3 and 6 button controllers. • All DB9 controllers (Genesis 3/6, SMS, Atari style 1 or 2 button and compatible) are auto-detected. • The wiring is now different for DB9 controllers. Easier to wire to the multiuse PCB. • NES controllers are now detected in SNES mode. This works great with original Nintendo controllers. If you have trouble with clones, use NES-only mode. • The project is now released under the GPL v2 license
nes_snes_usb-1.4.tar.gz (releases/nes_snes_usb-1.4.tar.gz)	<ul style="list-style-type: none"> • Added support for Atari style joysticks (Atari, Commodore, etc). Two button variations (eg: Sega master system) are also supported. • Added padding to the nes report descriptor. This makes it work correctly on Windows. Linux sure is tolerant...
nes_snes_usb-1.3.tar.gz (releases/nes_snes_usb-1.3.tar.gz)	<ul style="list-style-type: none"> • Fixed a bug which resulted in random button toggling when no controller was connected to the circuit. • NES mode now has it's own USB product ID. • PD1 is left in input (and no pull-up) since PD1 and PD0 must be shorted together in PCB rev.C (cleaner workaround than throwing away a batch of 50 PCBs and easier than cutting a track and soldering a wire...)
nes_snes_usb-1.2.tar.gz (releases/nes_snes_usb-1.2.tar.gz)	Added SNES mouse support.

nes_snes_usb-1.1.tar.gz (releases/nes_snes_usb-1.1.tar.gz)	Reworked the report descriptor for better behaviour under Windows and MacOS X
nes_snes_usb-1.0.tar.gz (releases/nes_snes_usb-1.0.tar.gz)	Initial release. Snes and Nes support.

Please contact me at raph@raphnet.net (<mailto:raph@raphnet.net>) if you do useful changes.

Objective Development driver modifications:

The NES/SNES mode selection jumper changes the HID report descriptor. I had to modify Objective Development's usb driver to support this. Here is a diff against the usb driver from HIDKeys.2006-03-14:

[usbdrv-diff \(usbdrv-diff\)](#)

USB Vendor ID/Product ID pair:

Please do not re-use my VID/PID pair for derived or other projects. Instead, get your own. I bought my VID/PID pairs from [mecanique](http://www.mecanique.co.uk/products/usb/pid.html) (<http://www.mecanique.co.uk/products/usb/pid.html>), and they are much less expensive than the 2000\$ US it cost to get a vendor ID from the usb implementers forum.

[↑ \(#home\)](#)

Printed Circuit Board for surface-mount version

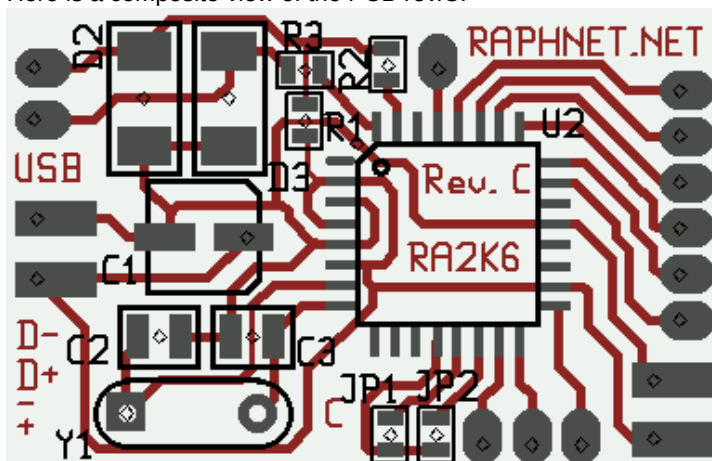
([snesusb_revC_pcb.jpg](#)) The surface mount version has many advantages:

- Very small (approximately 1"1/4 x 3/4" or 32mm x 20mm).
- Fits inside 3/4 pvc pipes. Ideal for installing inline with USB connector.
- Can be installed directly inside an Snes or Nes gamepad.
- Less wires to solder than on a breadboard. In some ways, it's easier to build.
- Looks more professional.
- You can use the board for other purposes. (After all, you have a reprogrammable MCU with USB and a few IO pins)



On the other hand, soldering surface mount components is harder if you dont have appropriate equipment or if you are a beginner.

Here is a composite view of the PCB rev.C:



For reference, here is the composite view of rev.B: [snes_nes_usb_pcb_revB.png](#) ([snes_nes_usb_pcb_revB.png](#))

Here are the gerber files you can use to build your PCB(s):

Revision C: [snesusb_revC.zip](#) ([releases/snesusb_revC.zip](#))

Revision B: [snes_nes_usb_PCB-revB.zip](#) ([releases/snes_nes_usb_PCB-revB.zip](#))



↑ (#home)

Assembly of the surface-mount version

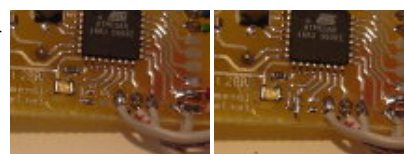
Step 1: Use the **composite view** ([pcb_revC_composite.png](#)), the **schematic and component list** ([#schematic](#)) to find out where each component goes. Solder everything in place and inspect the board carefully for solder bridges.

IMPORTANT: Do a solder bridge between PD1 and PD0 (see schematic).

([snesusb_revC_pcb_wiring.jpg](#)) **Step 2:** Use the appropriate diagram on the left to solder the USB, ISP and controller/joystick wires at the right places. The ISP cable can be built with a 2x3 header and a piece of flat cable. Pin numbers on diagram matches the standard atmel 6 pin AVR ISP.



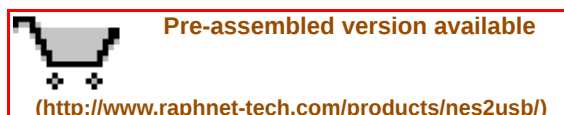
([solder_mode_select0.jpg](#)) ([solder_mode_select1.jpg](#)) **Step 3:** If you will be using a NES gamepad, close JP1 with a solder bridge or a small piece of wire. For DB9 gamepads, close JP1 *and* JP2. Refer to the pictures on the right for examples.



Step 4: Connect the USB and ISP cables. Use your programmer to program the hexfile into the ATmega8. Next, set the fuses bytes (High byte=0xc9, low byte=0x9f).

Step 5: Test the adapter with a game. If all works well, you can remove the ISP cable if you don't plan to update the firmware. Now find a way to protect the PCB. You can fit it inside a little box, a piece of PVC pipe or a heat-shrink. Also, adding hot glue near the wires you soldered on the PCB will prevent the wires from breaking near the solder points.

Step 6: Have fun!



↑ (#home)

Pictures and screenshots

My first prototype, built on breadboard with thru-hole components:



([snes_nes_usb_proto0.jpg](#))



([snes_nes_usb_proto1.jpg](#))



([snes_nes_usb_proto2.jpg](#))

Pictures of the surface mount version:



([1snesusb_component1.jpg](#))

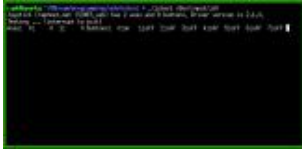


([snesusb_isp1.jpg](#))



(all_wires_soldered.jpg)

Screenshot of the adapter working under Linux (jstest):



(linux_jstest.png)

Screenshot of the adapter working under Windows XP (Control panel):



(snes_nes_usb_XP.png)

Screenshot of the adapter working under MacOS X (USB Prober):



(snes_nes_usb_MacOS_X.png)

USB SNES mouse:



(snesmouse_usb.jpg)

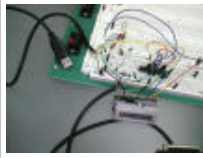
[↑ \(#home\)](#)

Pictures from users

I like to see how others build this project. Please send me your pictures and I will put some of them here.

29 Janvier 2014

Josafá Siqueira from Brazil sent me the following pictures of an adapter he built:



(user/IMG_20140116_105642.jpg)

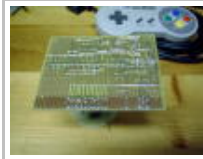


(user/IMG_20140122_102234.jpg)

Tjerk Kuperus build this project and my my **arcade style controller for SNES** (../arcade_control/arcade_control_en.php) on a single board:



(user_images/Tjerk/M5110380.JPG)



(user_images/Tjerk/M5110382.JPG)



(user_images/Tjerk/M5110384.JPG)



(user_images/Tjerk/M5110386.JPG)

Andreas Kronawetter, from Austria, built the circuit inside a NES controller:



(user_images/Andreas/nas.jpg)



(user_images/Andreas/nas1.jpg)

JP from the Netherlands built an USB SNES controller:



(user_images/Nin_01.jpg)



(user_images/Nin_02.jpg)

Michael Herzog converted a NES controller::



(user_images/controller.jpg)

Philip Buchegger built this nice computer in a NES case and used this circuit to convert the two controller ports to USB:



(user/15349_1282184491480_1136124949_872150_2054877_n.jpg)



(user/15349_1282184651484_1136124949_872154_3746423_n.jpg)



(user/15349_1282184531481_1136124949_872151_2545020_n.jpg)



(user/15349_1282184731486_1136124949_872155_2536236_n.jpg)



(user/15349_1282184611483_1136124949_872153_6904817_n.jpg)

Tobias Schulte, from Germany, sent me those pictures:



(user_images/DSC00156.JPG)



(user_images/DSC00157.JPG)



(user_images/DSC00160.JPG)

[↑ \(#home\)](#)

Disclaimer

I cannot be held responsible for any damages that could occur to you or your equipment while following the procedures present on this page. Also, I GIVE ABSOLUTELY NO WARRANTY on the correctness and usability of the informations on this page. Please note, however, that the procedures above have worked in my case without any damages or problems.

Now you cannot say that I did not warn you :)

[↑ \(#home\)](#)

Any trademarks used on this site are the property of their respective owners.
Copyright © 2002-2020, Raphaël Assénat

Website coded with  [vim \(http://www.vim.org\)](http://www.vim.org)

Last update: January 7, 2020 (Tuesday)