Universal Wireless Retro Controller v1.2

I. Introduction

Have you ever wished you had a wireless controller for your retro console? I don't mean those crappy IR controllers back then which you had to point at the console and looked like shit anyways.

This guide will show you how to build sleek, fully functional wireless controllers and receivers for your retro consoles.

But be warned, it's not a simple task. It involves flashing microcontrollers, dealing with lithium-ion batteries, soldering wires, cutting holes, and getting along with the limited space inside a controller.

But it's also very rewarding. When you're playing some SNES games with your finished controller and you're realizing that there's no cable between you and the console you'll start smiling, surely ;)

II. **Compatibility**

Currently, following consoles are supported:

•SNES

•NES

•Sega Saturn

•Nintendo 64 (no rumble & memory pak support, though)

As you might know these controllers and receivers are cross-console compatible. This means you can play SNES with a Saturn controller and vice versa, for example.

But this compatibility has its limits, of course. If you want to play Super Metroid with the Saturn controller you'll run into problems soon because you can't switch weapons without the select button.

Also, fancy things like remapping the buttons or turbo fire aren't supported. But I made some other tweaks I considered important:

•Z on N64 controller & L+R on Saturn controller act as select button on NES.

•Z+C on Saturn controller are L+R on SNES.

•When playing N64 with a SNES controller, press select and SNES-D-pad will be N64-analog stick, SNES-L will become N64-Z.

You should always keep in my mind that this mod was developed to make the controllers fit my needs. Being able to use the Saturn 6-button controller with fighting games on SNES and N64 was very important to me, supporting memory- and rumble pak on the N64 wasn't...

I tested the controllers on my consoles and every game I own works.

Console	Games
PAL NES	50x PAL , 2x US
US SNES Mini	60x US, 5x PAL, 1x JAP
US N64	22x US
JAP Saturn	About 10x JAP with 50&60Hz

I can't guarantee that the controllers will work on different console versions I haven't tested, although they should. So if you run into problems with a specific game or console version then just let me know. Maybe it can be fixed. =)

III. Main Parts

I will comment on the main parts that are needed for this mod.

Microcontrollers:

We need two different microcontrollers for the receiver (RX) and the transmitting controller (TX). The microcontrollers are always the same for every console. ATMega8 for the receiver and ATTiny2313 for the transmitter, both prduced by Atmel.

Battery:

I always use a small rechargeable lithium-ion polymer battery for the 1st generation Ipod Shuffle. It measures 36 mm x 19 mm, making it easy to fit inside the controller. Its capacity is 250 mAh. That's not very much but when you consider that the transmitting circuit draws just about 2 mA you will see that this battery will last quite a while. Also, this battery features built-in protection against under-voltage and short circuit. Nevertheless, please be very careful when dealing with this type of batteries. When attaching the wires to the battery don't apply more heat to the battery than necessary.

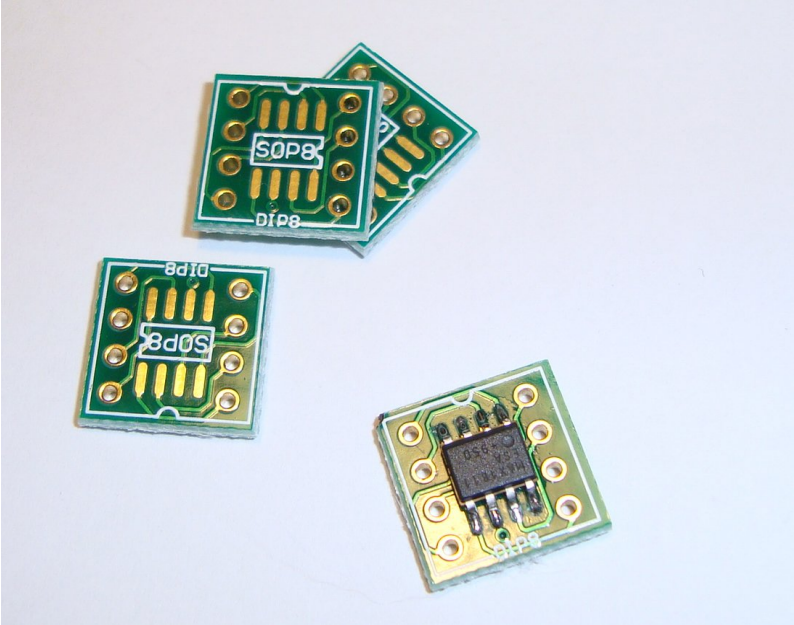
Note that for the N64 version we don't need that Li-ion battery. Instead we'll use the case of a rumble pak. That way 2 normal AAA batteries can be used. The reason for that is the high power consumption of the N64 controller (about 14 mA). The tiny Ipod battery would be drained quickly.

MAX1811:

The MAX1811 will do the difficult job of charging the battery. It just needs 2 capacitors. We'll provide power through a USB cable. This means you can charge the controller with every PC, Xbox 360, USB wall adapter, etc.

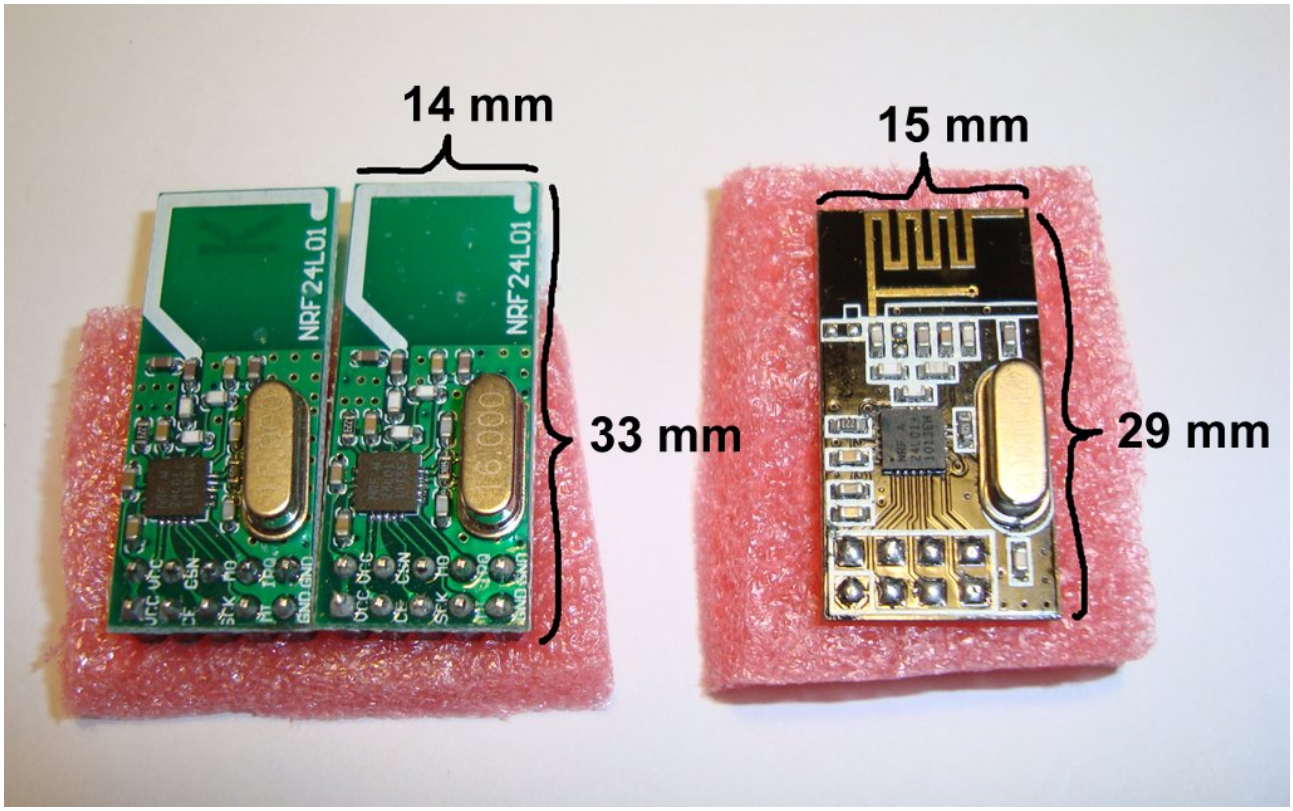
Unfortunately the MAX1811 is only available in a SO8 SMD package. This makes soldering quite difficult. Therefore I recommend using a SO>DIP adapter. These adapters can be found on eBay

easily.



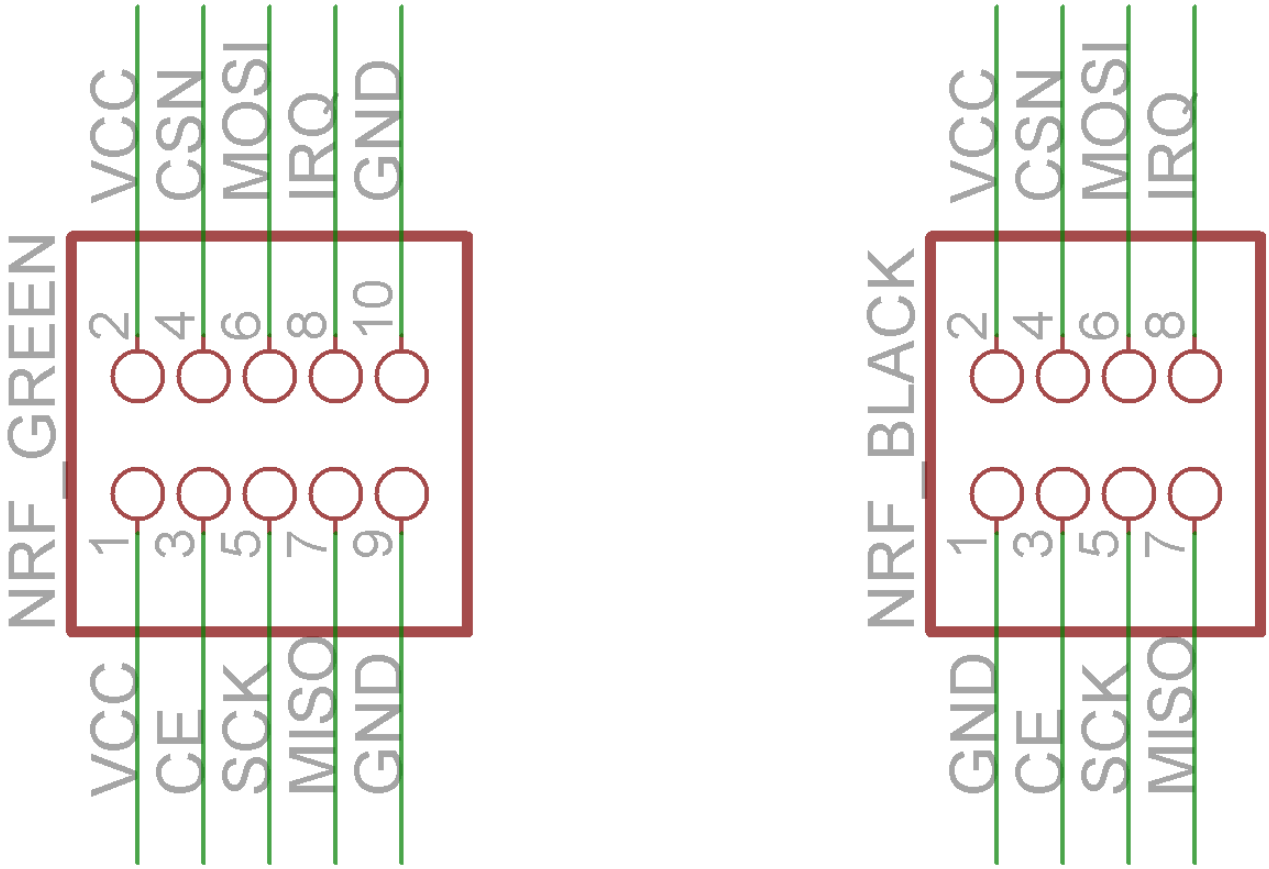
NRF24L01+:

Probably the most important part is the NRF24L01+ 2.4GHz transceiver by Nordic Semiconductor. There are many different ready-to-use modules with built-in antennas, all of them featuring the same NRF24L01+ chip. Again, they can be found on eBay easily.



By now I've had 2 different versions, green and black ones. The green module is a little bit taller. The pinout is printed on the green one but not on the black one. The pinout is almost the same, only the pins for VCC & GND are different.

In my schematics I've used the pinout of the black module. So if you got the green module you have to use a different pinout, of course:

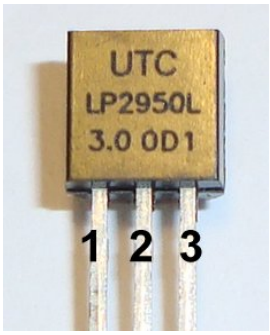


(pin no.1 is in the lower-left corner)

LP2950 Voltage Regulator

Unfortunately the transceiver module runs on 1.9 to 3.6 V only. So the voltage of a fully charged Li-ion battery is too high (4.1 to 4.2 V). That's the reason we need a 3.0 V voltage regulator.

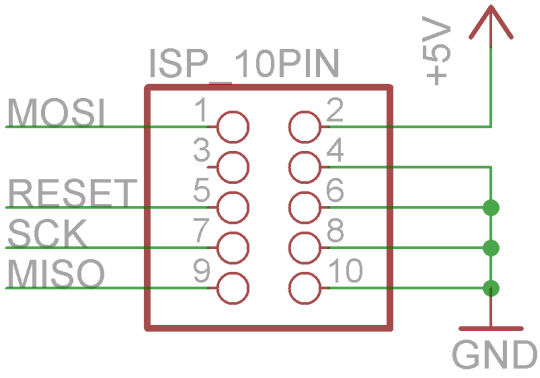
The LP2950 is available in many packages but I prefer the TO-92 package. Mind the pinout:

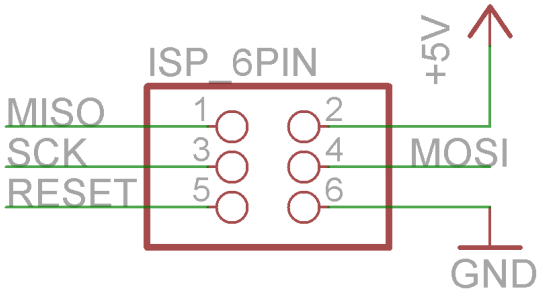


IV. Programming the Microcontroller

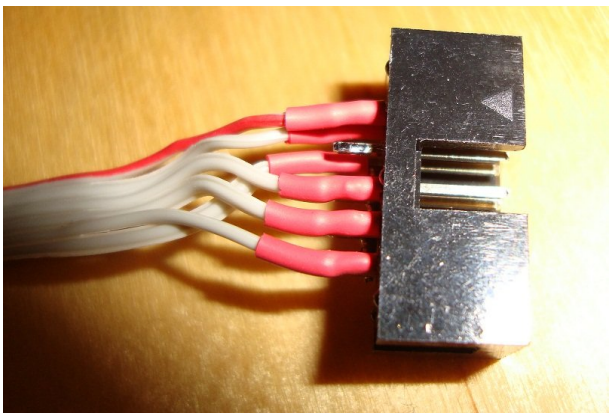
The microcontroller (MCU) must be flashed with the appropriate program (.hex file) to make it work. This requires a programmer for AVR MCUs. There are tons of different programmers on eBay: Cheap ones for the parallel port, more expensive ones for the USB port.

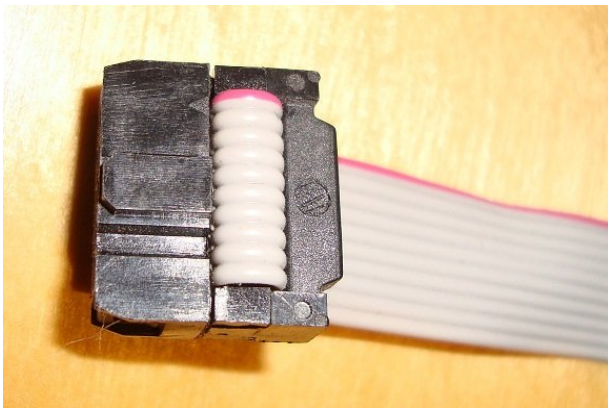
The programmer will have either a 10 pin or 6 pin cable, again the pinout differs:





You should mind that the red wire of the programming cable is pin 1 always. Usually there's also a little arrow on the plug and on the socket designating pin 1.





There are many programs capable of flashing MCUs. Maybe your programmer came with an included software, then you can use it. Or you can use AVR Studio, ISP Programmer (recommended), myAVR Progtool (also recommended), or AVR Dude. I will not explain how to use all those different programs, so you better read the documentation. Besides, there are already many guides and tutorials out there explaining how to program AVR MCUs.

Newer programmers can provide power to the MCU. Programmers for the parallel port don't! In that case you need to provide power externally by yourself.

You need to flash the MCU with the appropriate .hex file AND you also need to set the so-called fuses. These fuses determine how fast the MCU runs, which clock source to use, etc. You must program the high fuse and the low fuse in order to make the MCU work.

Both the .hex file and the needed settings for the fuse bytes can be found in the according folder (N64_RX for example). Setting the wrong fuses can brick your MCU!

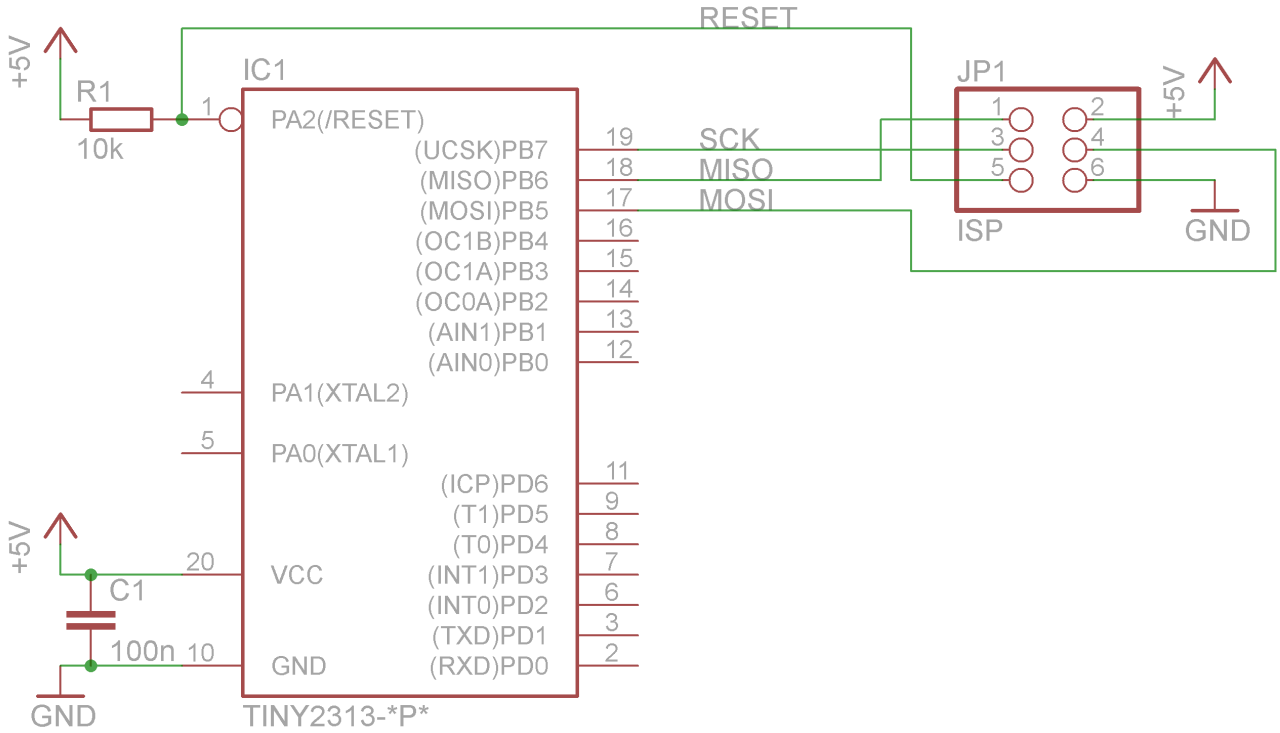
Some of the MCUs need an external crystal in the finished circuit. Once you set the fuses for these MCUs to use the external clock source you will not be able to program the MCU without that crystal again.

So you better flash the MCU first and after that you should set the fuses. Though, the best way is to include the ISP programming socket or pin header inside the finished controller / receiver. That way you can always re-flash your MCU in case something went wrong or when a new version of the

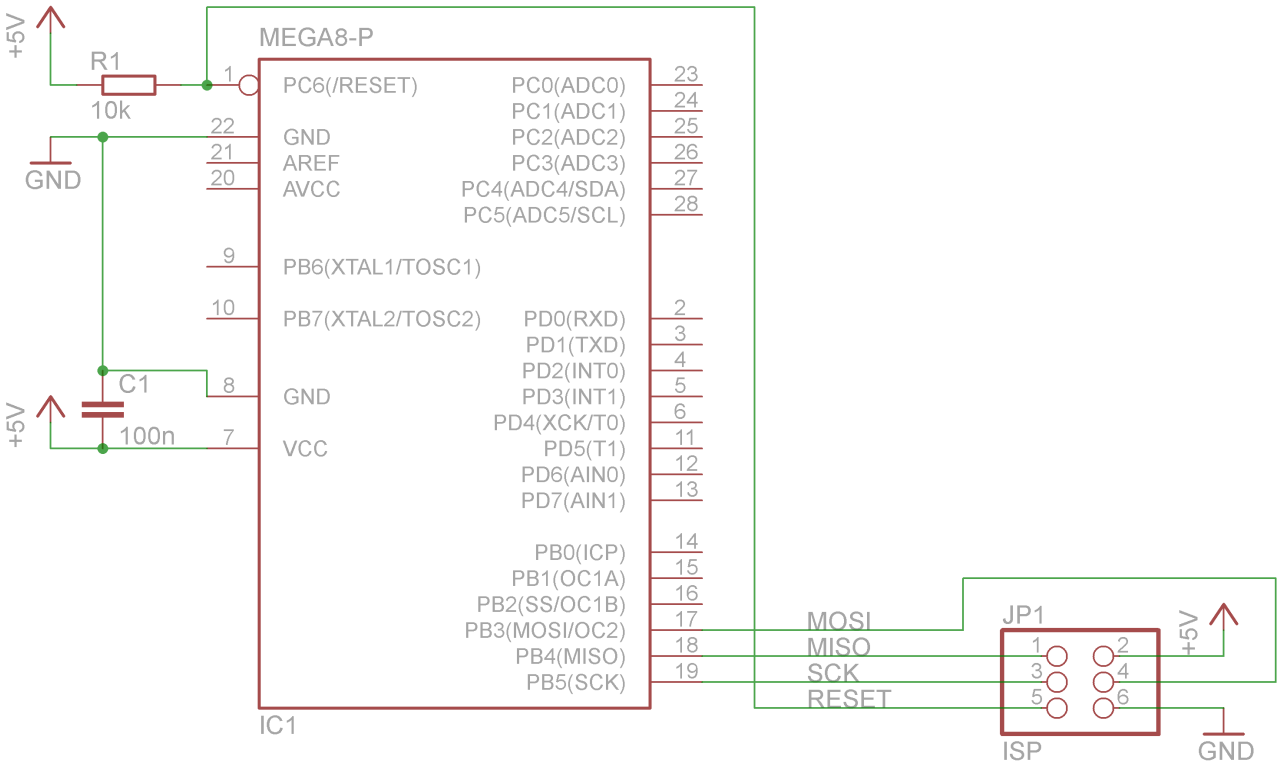
program is released. Another advantage of this method is that the MCU is powered by the console (inside the receiver) or by the Li-ion battery inside the controller. That's also the actual idea behind ISP (In-System Programming). =)

Nevertheless, these are the basic schematics for flashing the ATTiny2313 and ATMega8 MCU:

ATTiny2313:



ATMega8:



V. Building the Wireless Controller

In the folders are schematics and part lists for each receiver and controller. Additionally I will give you some tips and pics for each system.

Nintendo 64

Pin no.	Function	Wire color (usually)
1	VCC	**red**
2	DATA	**white**
3	GND	**black**

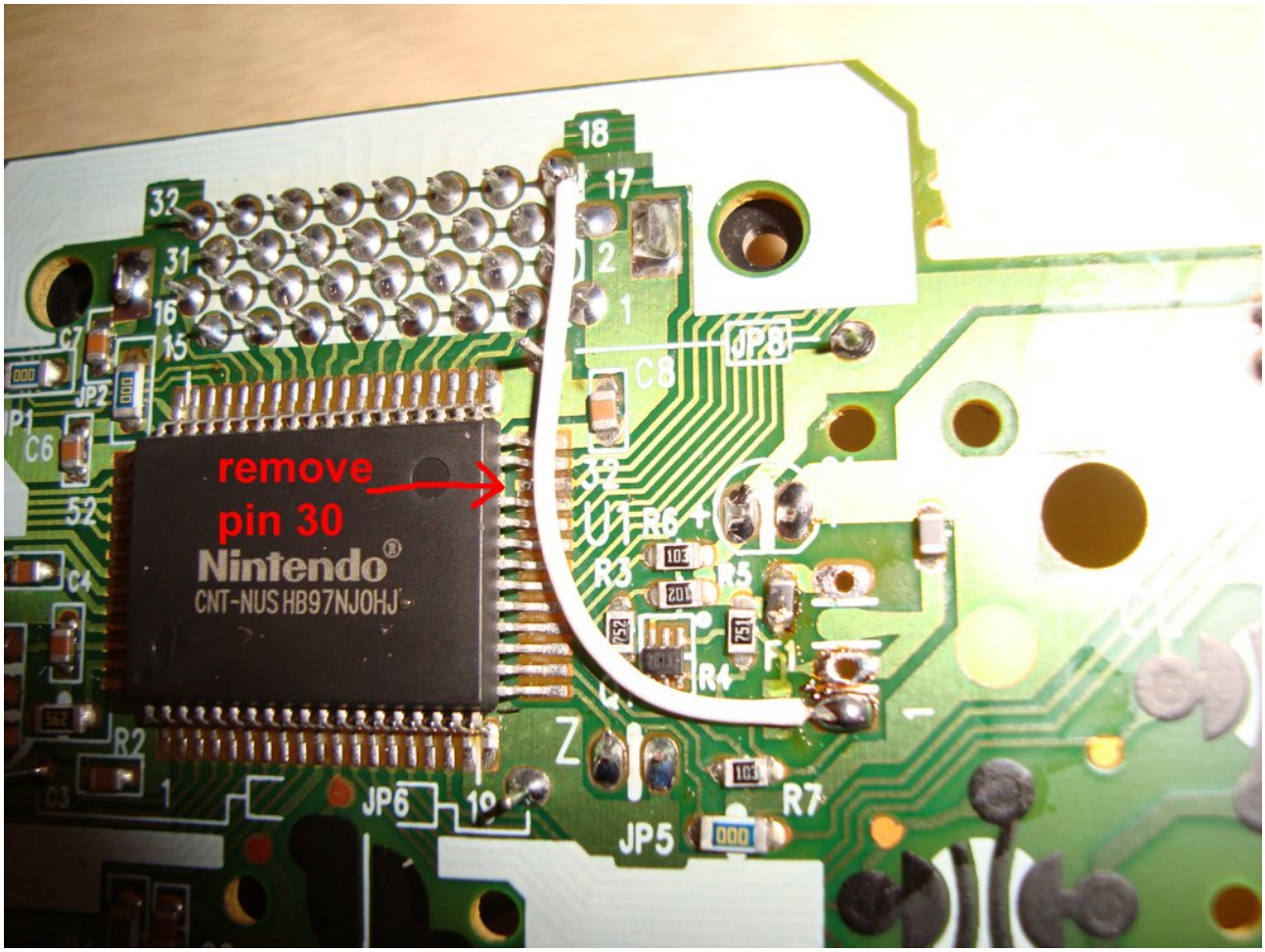
Making a wireless controller for the N64 is quite easy compared to the other consoles for 3 reasons:

1.)No need for voltage regulators because the console provides 3.3 V. Since we mod the case of a Rumble Pak we can use two regular AAA batteries. That means no voltage regulators, no Li-ion battery, no battery charger.

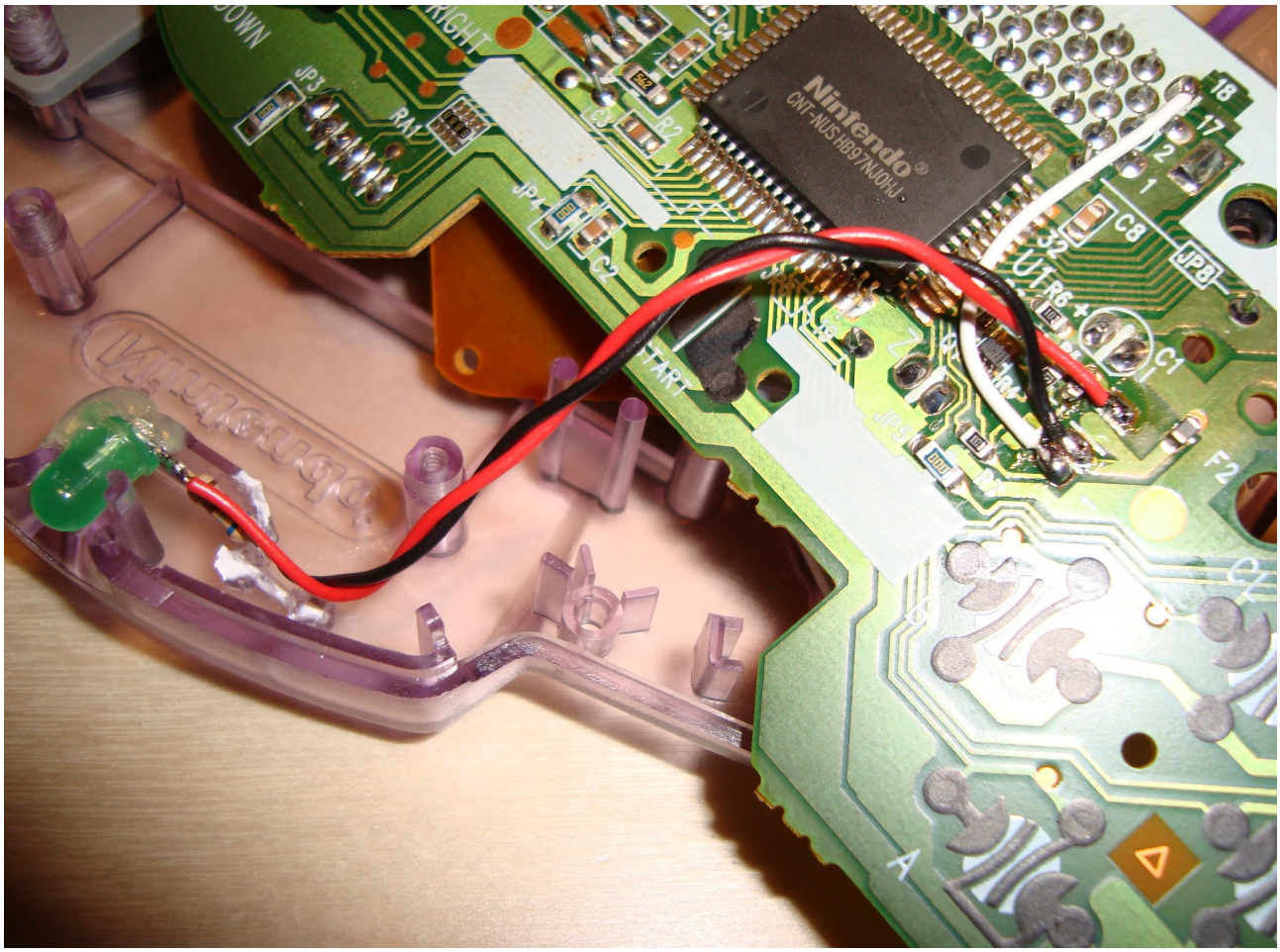
2.)The N64 uses only 1 data wire, other consoles 6 or more.

3.)Inside the Rumble Pak is a lot of space for the MCU, transceiver, and even an ISP socket.

Modding the controller is done quickly: Just remove pin no. 30 of the main IC and solder a wire between pin no. 18 of the expansion slot and the data pin.

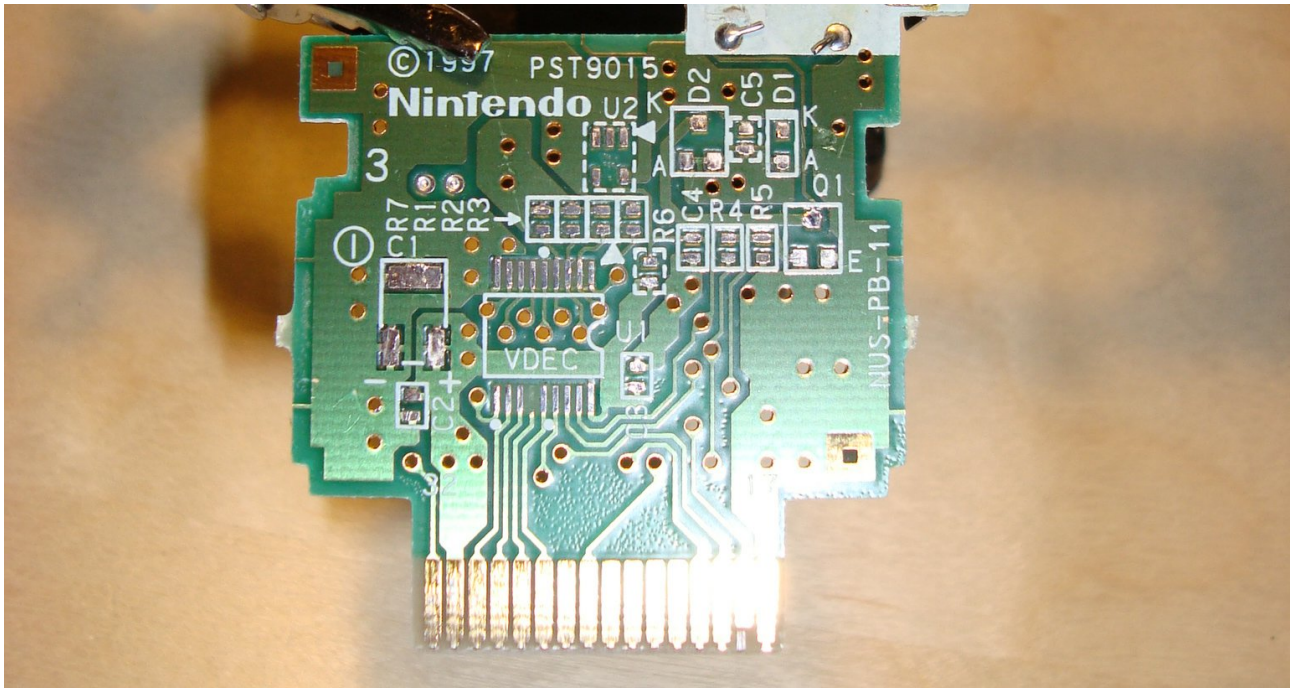


Now install the power LED where once the controller cable's outlet has been:



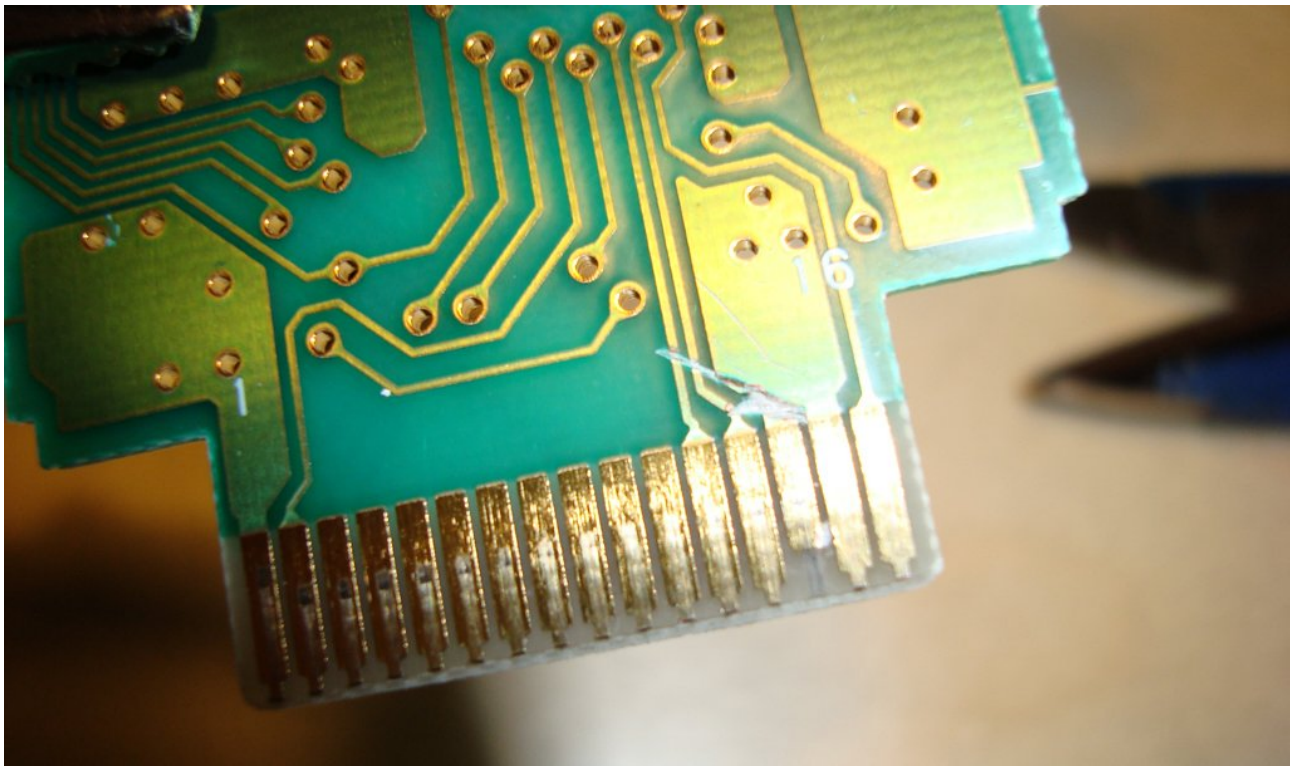
The controller itself is finished.

Now it's time to open the Rumble Pak:
Discard the large vibrating motor and desolder all the components on the PCB.

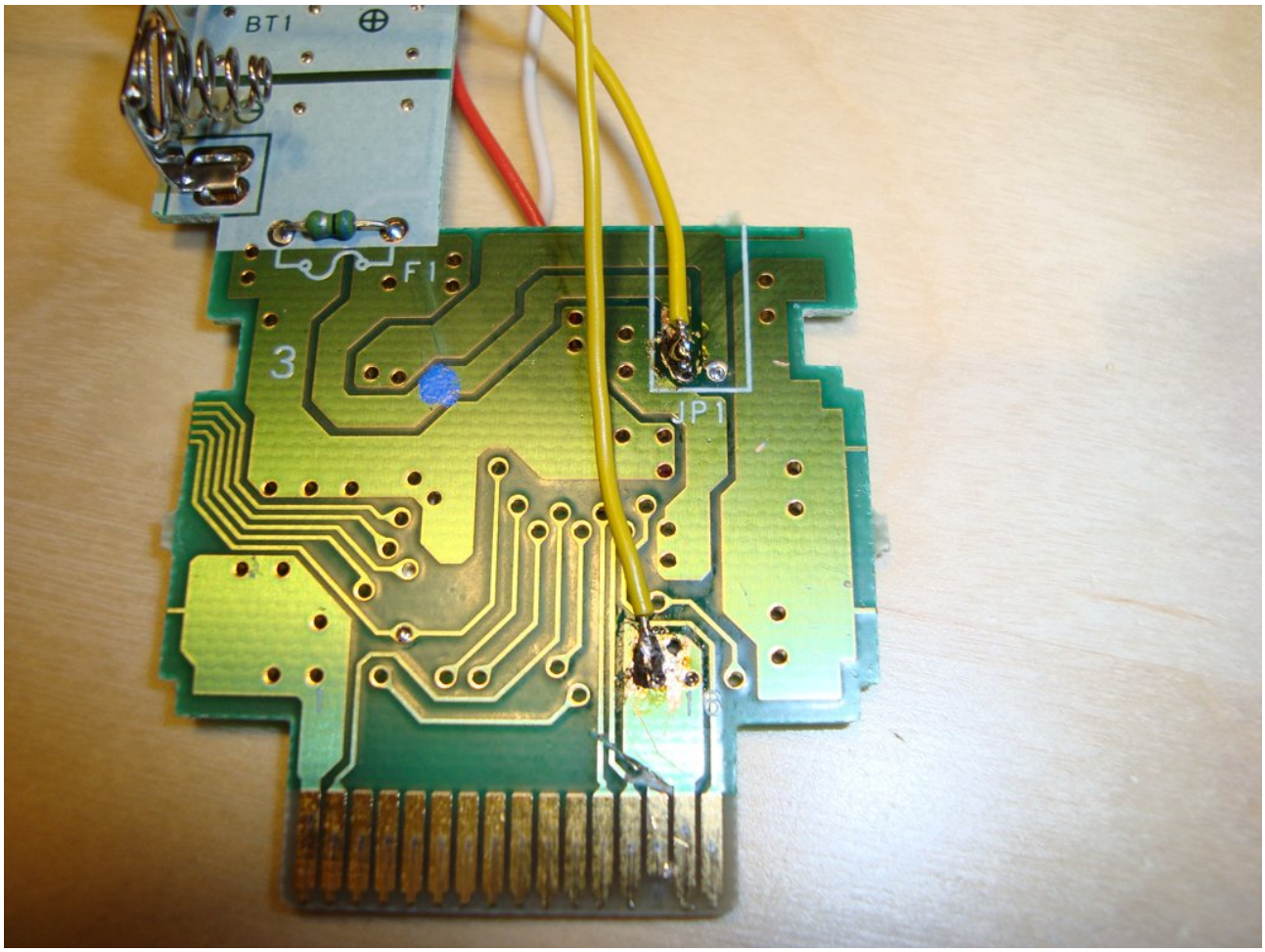


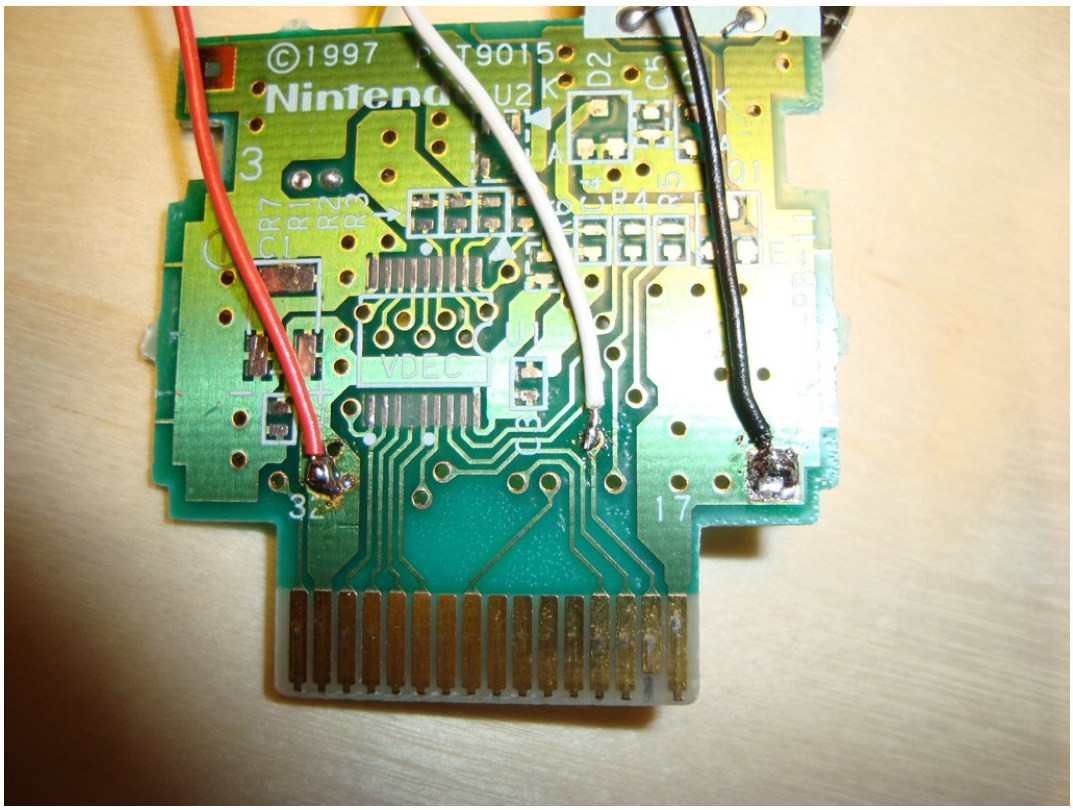
Now you should cut the connection between pin no. 14 and 15. Otherwise the N64 controller would

assume that there's something plugged into its expansion slot.



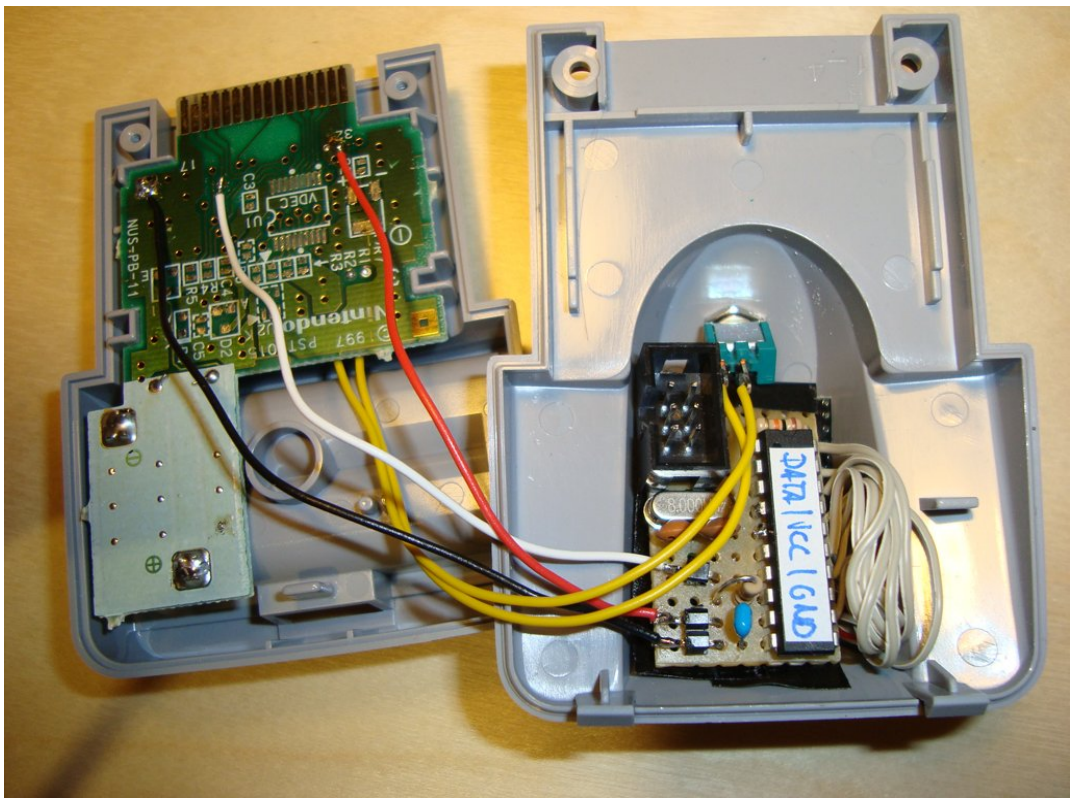
Now attach 5 wires to the PCB:





The 2 yellow cables will go to your power switch. If you turn on the switch you will power the controller as well as the MCU and the transceiver. The black wire is GND (pin 17), the red one is VCC (pin 31) and the white one is DATA (pin 18).

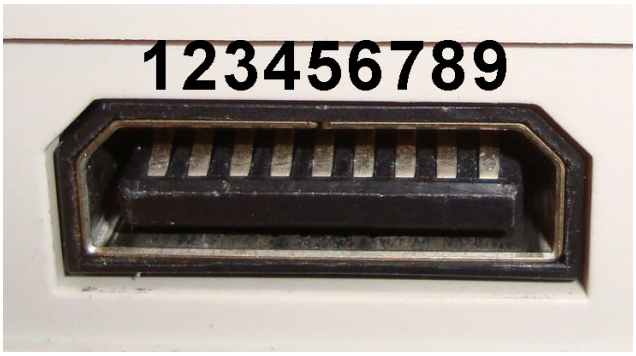
Build the rest of the circuit according to the schematic. Here's a pic of how I ended up:



There was even enough space for an ISP socket. The transceiver module is under the MCU.

Building the receiver is pretty straight forward if you realize the schematic faithfully =)

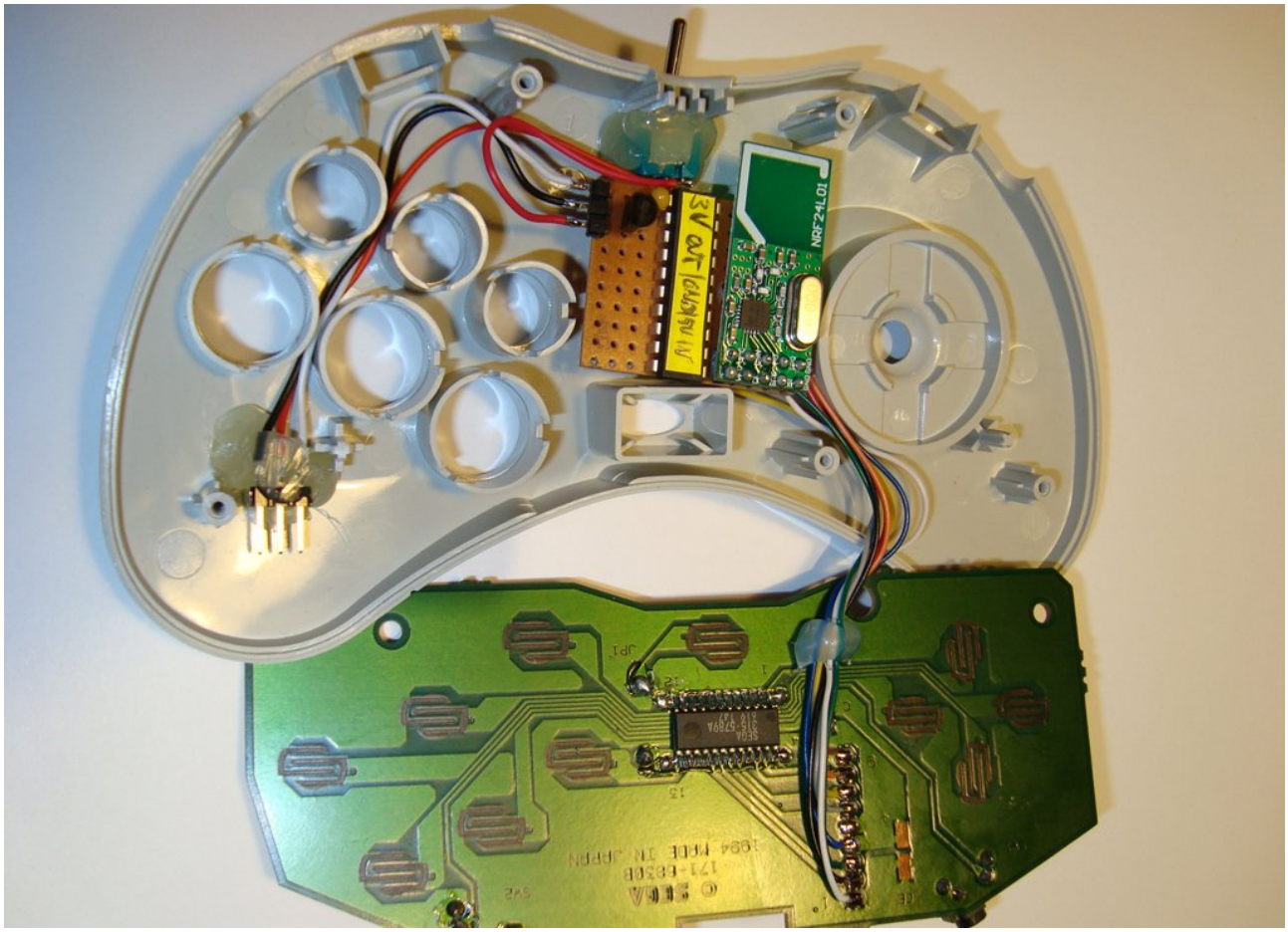
Sega Saturn



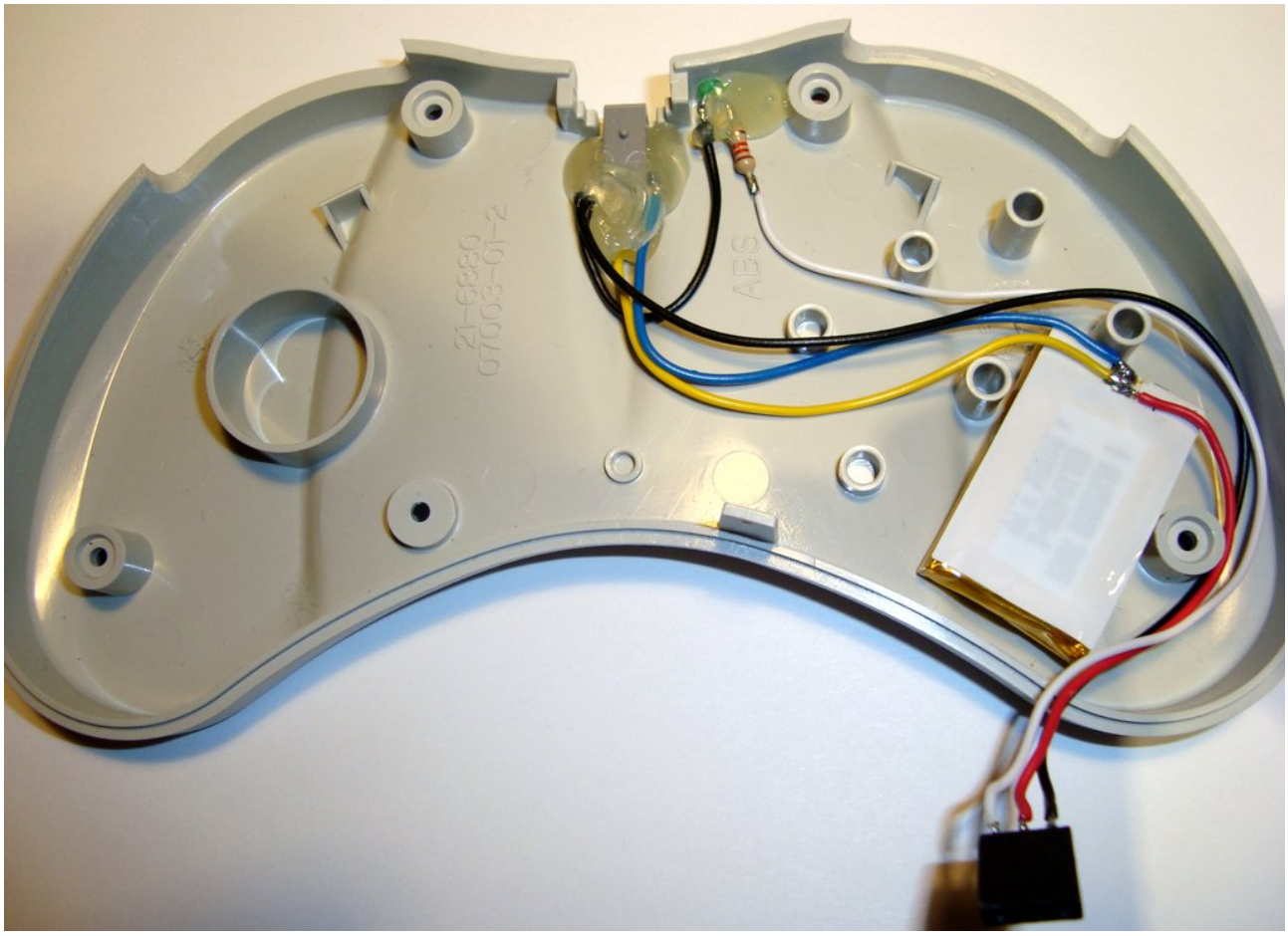
Pin	Name	a.k.a.	Wire color (usually)
1	VCC	VCC	**blue**
2	D	D1	**green**
3	U	D0	**black**
4	TH	S0	**orange**
5	TR	S1	**red**
6	TL	+5V IN	**brown**
7	R	D3	**yellow**
8	L	D2	**grey**
9	GND	GND	**white**

(**Watch out!** The pinout on the PCB inside the controller is different, check the wire color!)

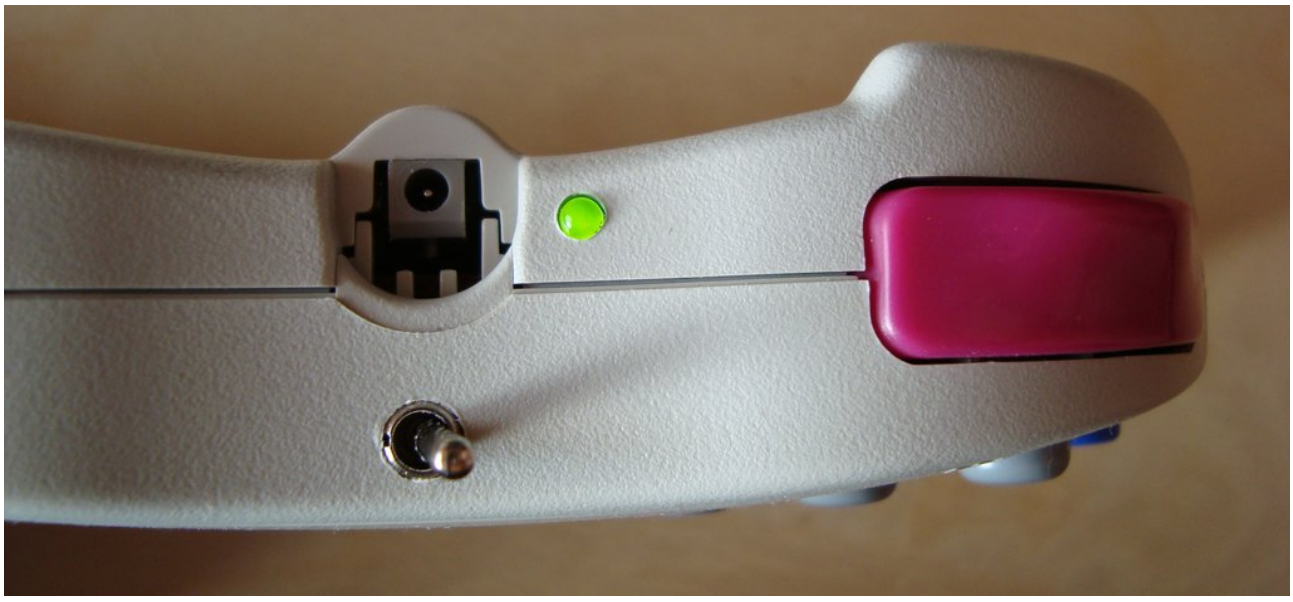
I'd install the power switch and the main circuit into the upper half. There should even be enough space to add a ISP socket too (not pictured).



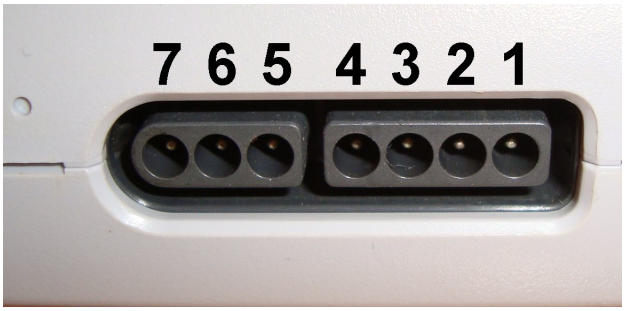
In the lower half you'll find enough space for the Li-ion battery, power LED, and the charge jack. I've used a very small jack, but you can choose any jack that fits inside the controller. (Maybe a 2.5 mm mono jack?)



Finished controller:



SNES

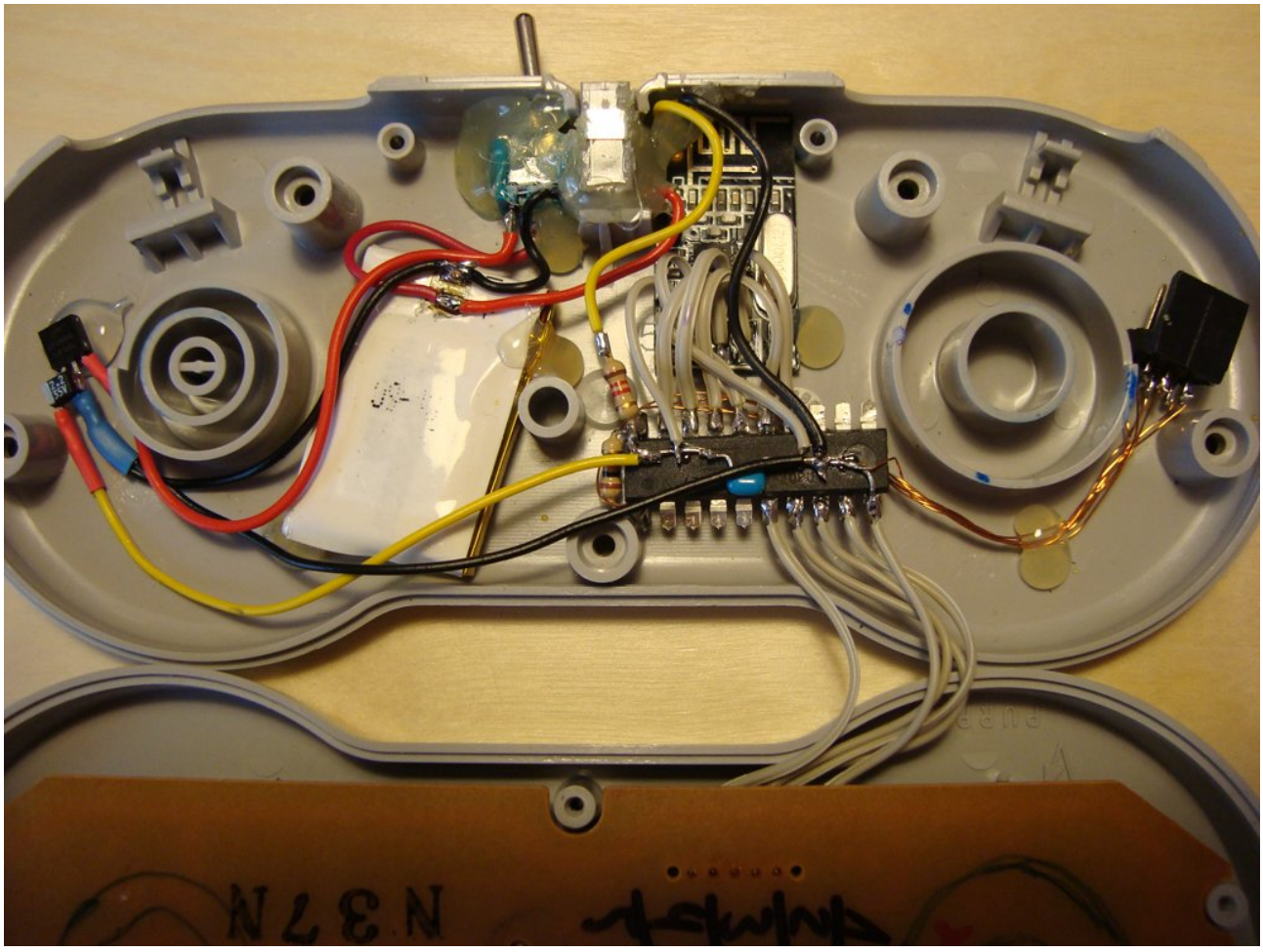


Pin	Name	Wire color (usually)
1	VCC	white
2	CLOCK	yellow
3	P/S	orange
4	DATA	red
7	GND	brown

You can install the whole circuit including battery and ISP socket into the lower half of the SNES controller.
But unlike the Saturn controller you have to keep everything as flat as possible. This means you can't solder the MCU and NRF24L01+ module on perfboard this time. You even have to unsolder the transceiver module's pin header, otherwise it's too tall.

Other than that it's not too difficult. There's space for all the components, just keep below all those round standoffs and you'll be fine.

This is what my SNES controller looks like:



Building the SNES receiver is fairly easy if you stick to the schematic.

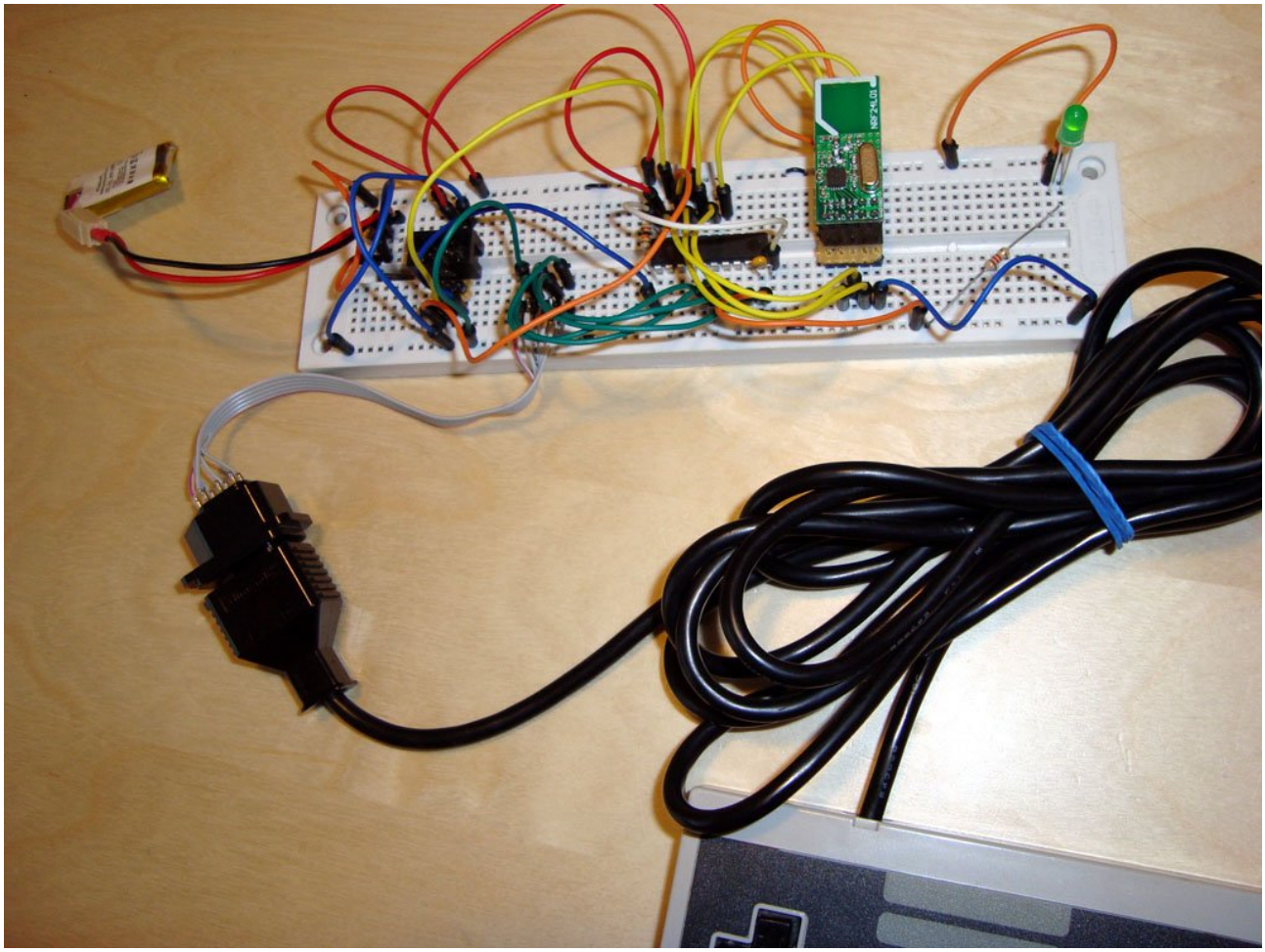
NES:

Pin no.	Name
1	GND
2	CLOCK
3	P/S
4	DATA
7	VCC

The reason why I haven't added the wire color to the table is that the colors for CLOCK and DATA seems to vary. So better take a multimeter and measure which wire connects to which pin. Don't rely on the wire color alone!

The schematics for the NES are very similar to the SNES version. That's no surprise as the serial communication protocols of the NES and SNES are identical.

I haven't integrated the transmitting circuit into my NES controller yet, but on the pic below you can see the whole transmitting circuit on a breadboard: (using a NES controller socket salvaged from a “NES Four Score” 4-player adapter)



Li-ion Battery Charger:

Actually the charger only consists of the MAX1811, 2 capacitors, and a LED indicating beginning and end of the charging process.
On the one end there's a USB cable providing power to the MAX1811. On the other end you must attach a cable with appropriate charge plug that connects directly to the battery inside your wireless controller. (Be careful not to mess up polarity!)

Here's my charger:

It's also possible to put the MAX1811 inside the wireless controller. Then you can install a Mini-USB socket into your controller. With the charger integrated into the controller you can charge it almost everywhere. Nowadays everyone owns a Mini-USB cable.

The downside to this is that the MAX1811 is rather expensive. If you want to make 3 or 4 wireless controllers you'll feel the difference for sure.

VI. General Tips / Usage

•Don't put the transceiver module where it's covered by your hands/fingers all the time! The antenna should be at the top of the controller.

•You can select the channel on the receiver any time. Channel switch open > channel 1; switch closed > channel 2.
To select the channel on the wireless controller hold either A or B while turning on the controller (A: channel 1; B: channel 2). The selected channel is saved so when you turn on the controller without holding any buttons, it will use the channel you selected before.
That means if you turn on the controller for the very first time you have to hold either A or B.

•Turn off the controller after play! The controller doesn't have an auto-turn-off function. After some time it won't transmit any longer and thus reducing power consumption. But the LED will still be on.
It's kinda comparable to the Gamecube Wavebird controller. You have to learn to turn off the controller after use, otherwise the battery will be empty next time ;)

VII. Version History

v1.0 initial version released on October 4, 2011.

v1.1 NES transmitter support added on April 10, 2012.

v1.2 There were reports that the SNES receiver didn't work properly on some PAL SNES systems. That issue has been fixed (hopefully) on January 1, 2014.

Additionally the source files have been added. I know, it's a mess!