3DS Streaming Console with External Control Interface

Technical overview

1. Introduction

The 3DS Streaming Console with External Control Interface, or 3xtDS in short, is a modified Nintendo 3DS console equipped with an capture card and an USB interface that allows the control of buttons, circle pad, and touch screen from an external source, as well as streaming high quality video of the content of both screen and sound back to the host to be recorded, or broadcasted. This allows some very interesting applications such as interactive streaming on Twitch or other websites, gameplay automation, remote play over internet, using alternative controllers, and much more.

The construction of this project is relatively straight forward. A 3DS with capture card is taken apart, various contacts to the buttons, circle pad, and touchscreen is then brought out to be controlled by a microcontroller, who executes commands from USB serial communication.

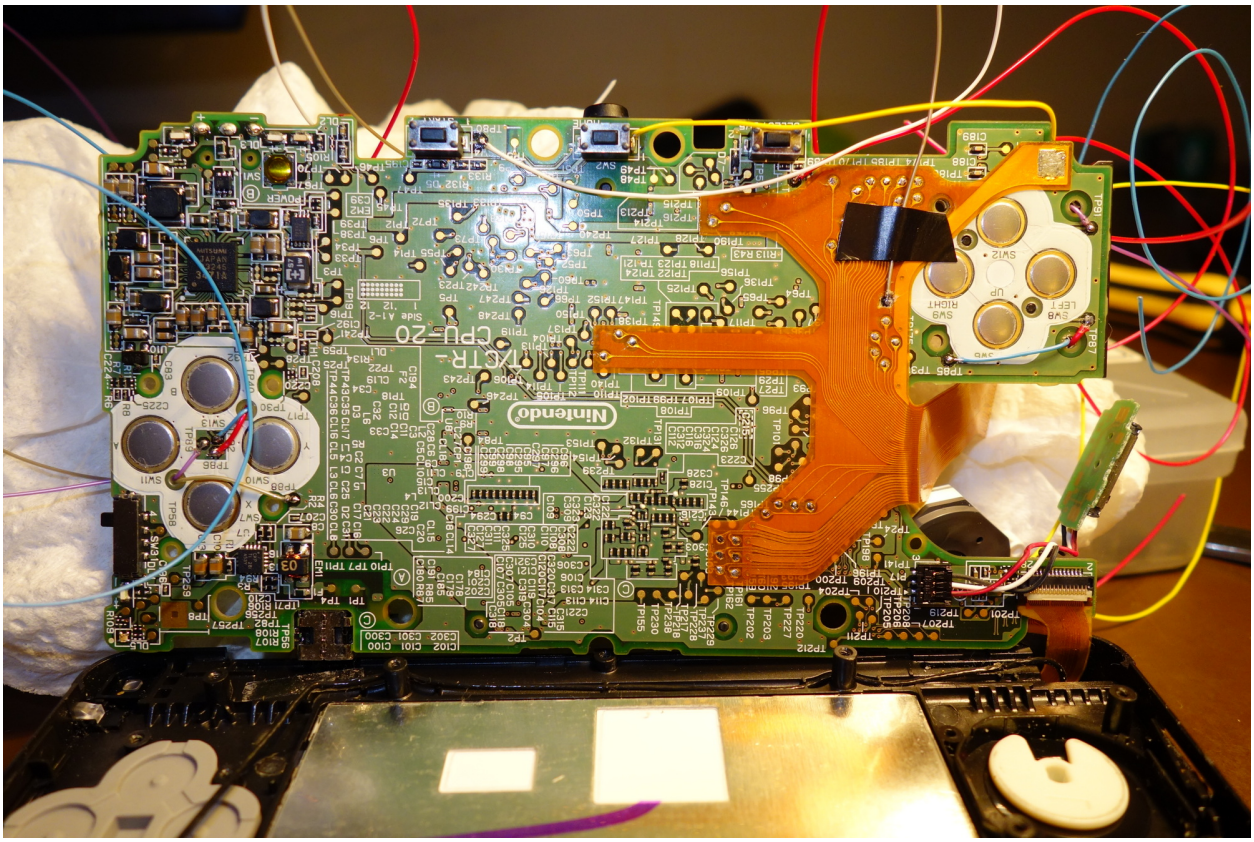
Please note that this document is just a brief overview of several aspects of this project, not a step-by-step guide of making one from scratch. Unless you have some decent experience with analog/digital circuits, microcontroller programming, taking stuff apart and being able to put it back together again, I would not encourage you trying creating one yourself, since you'll most likely break a expensive piece of gaming console.

2. Hardware Overview

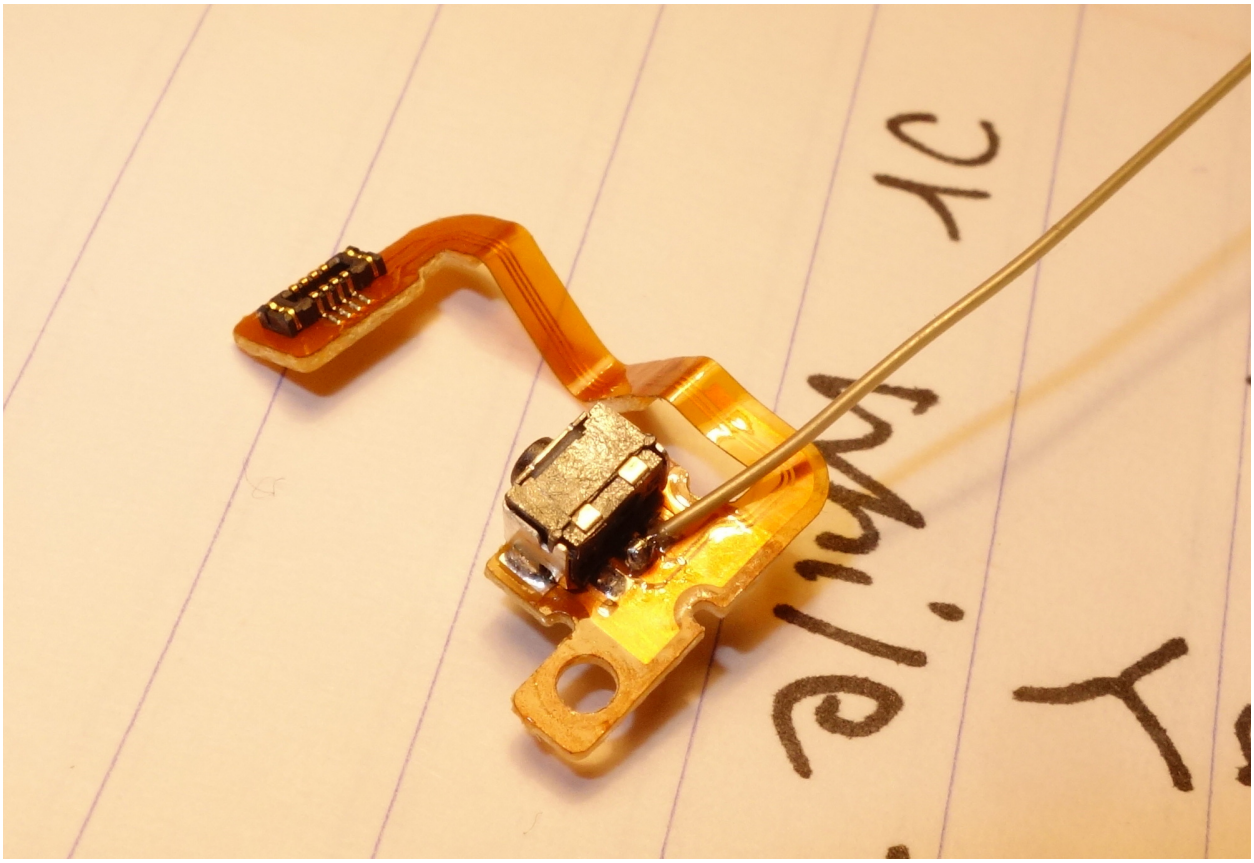
The hardware aspect of this project is relatively simple: just take a 3DS apart and solder some wires on some test points. But first of all you have to obtain a 3DS with a capture card. One was kindly lent to be by Reddit user velocikoopa, and it was him that made the entire project possible. You can buy a new one, but it's rather expensive. velocikoopa paid almost $600 for his 3DS with capture card. A much cheaper capture card-enhanced 2DS is coming out at $380. Although I haven't seen it in person yet.

Before we start, something to keep in mind: Don't forget to bring out the ground of the 3DS and connect it to your own circuit, otherwise 3DS will not respond to the control signals. And also 3DS uses 1.8V logic, so whenever I mention VCC it's 1.8V.

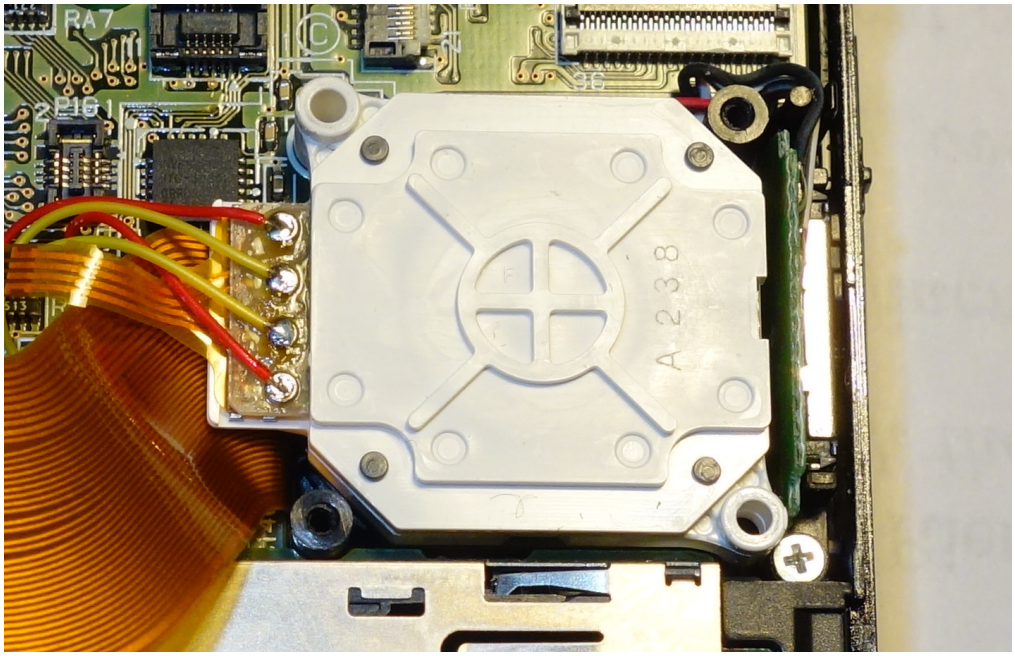
Follow iFixit's teardown until you have the mother board out, be careful as there are a lot of small components and fragile ZIF flat cables. Look for buttons' test points on the front side of the motherboard, namely TP85 for up, TP87 for left, TP91 for down, TP90 for right (this might be under the flat cable for the capture card, cut a little opening to access this test point), TP81 for select and TP80 for start. Three of the test points for A, B, X, Y are right in the middle of those four buttons, the other one is right above Y button. There is no test point for home button, but you can just solder a wire on the left pin of the button. Here is a picture of all the wires soldered in place:



There are no test points for L and R shoulder buttons either. I soldered wire directly to each button itself, on the middle pin:



Now onto the circle pad. It is basically an potentiometer with two outputs, one for each axis. When looking at this picture:



The pinout from top to bottom is: GND, x potential, VCC, y potential. To control the circle pad with microcontroller, simply desolder the potential output wires and feed them the DAC output from microcontroller instead. Range is from 0V to VCC.

Everything so far has been pretty trivial, buttons and circle pad. But now we face the biggest challenge: tricking 3DS into thinking someone is using the touch screen. 3DS uses a standard 4-wire interface, to get a general idea of how resistive touch screens work, I suggest taking a look at this document:

http://www.ti.com/lit/an/slaa384a/slaa384a.pdf

As with the method described in Section 1.3 the document:

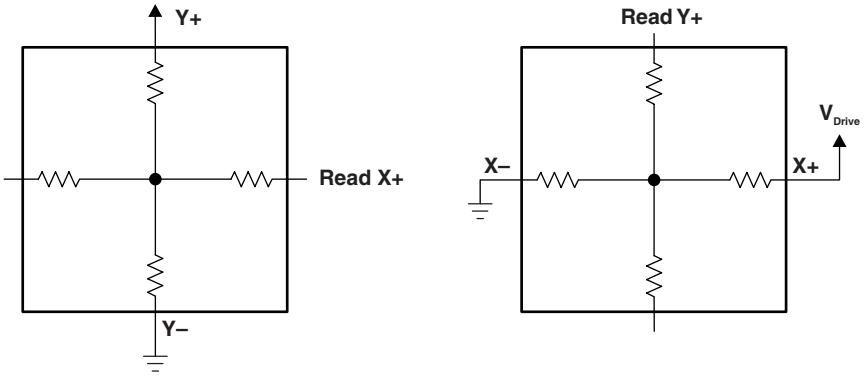
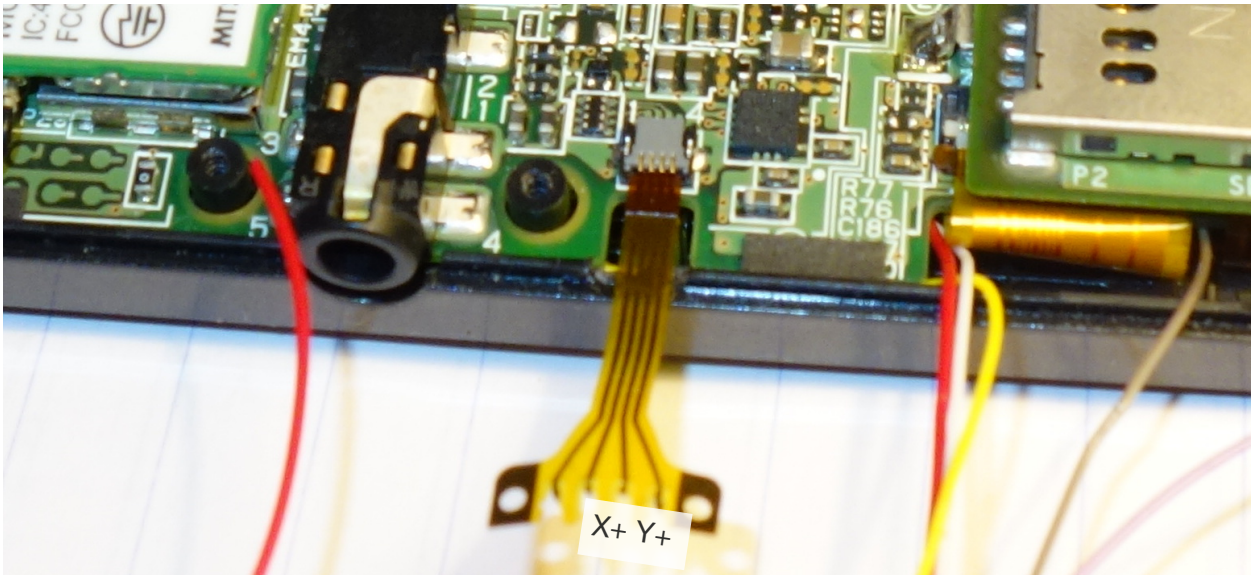


Figure 3. 4-Wire Touch Coordinate Reading

3DS drives Y+ to VCC, Y- to ground, and measure the output at X+ for X coordinate. Similarly, driving X+ to VCC and X- to ground yield Y coordinate at Y+. To get a complete reading of X

and Y coordinate, X+, X-, Y+, Y- are rapidly switched between input and output, at around 200Hz.

I suggest getting a extra 4-pin flat cable so that you can interface with the touch screen input on the motherboard without destroying the socket:



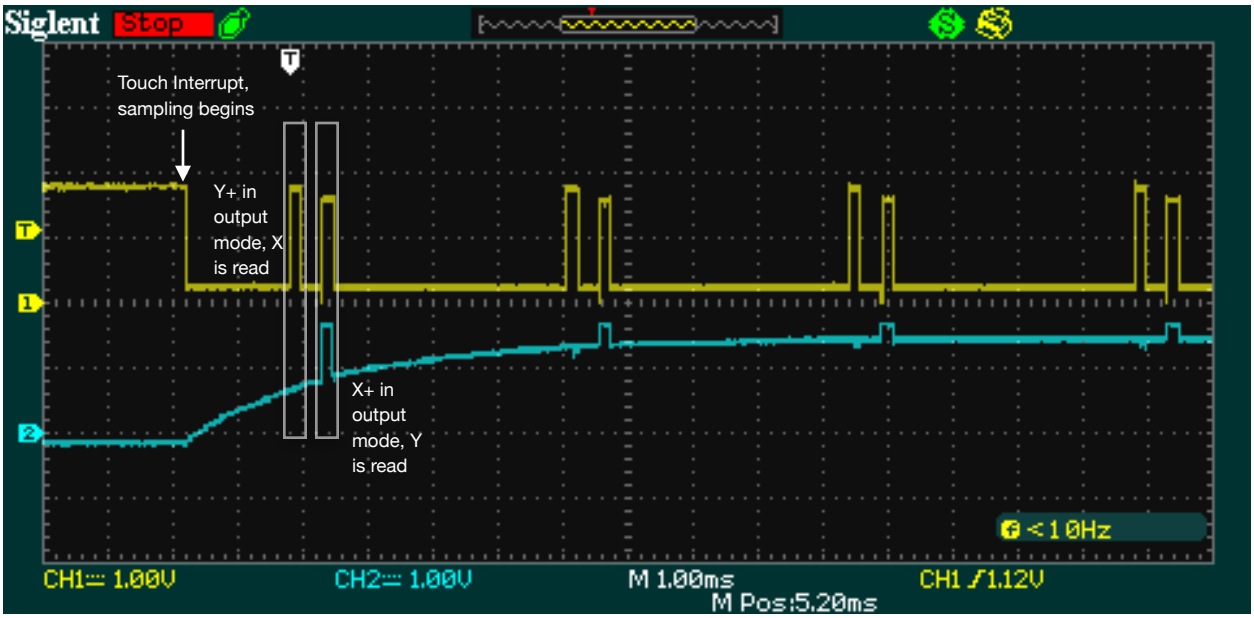
We only need the last two pins on the right in order to simulate a touch screen for 3DS, they are X+ and Y+ respectively.

The reading of the touch screen is interrupt based, as described in Section 1.2 of the document. Y+ pin is normally at VCC, which is pulled to ground when a touch occurs, generating a touch interrupt, and starting the sampling process.

To write X coordinates, you can tie the DAC output of microcontroller directly to X+ pin in series with a 10K resistor, so when X+ is in input mode, the DAC voltage is read, and when X+ is in output mode, the resistor prevents the current flow. Write 0V to 1.8V for coordinate 0 to 320.

For Y coordinates, the Y+ pin is tricker, since it also generates an interrupt, so some timing is required. After connecting it to DAC in series of a 1K resistor, pull it to ground to initiate touch interrupt, then monitor this pin using an on-board ADC. Once the Y+ pin has been switched to output (reading X coordinate), wait 500 microseconds, so now Y+ is in input mode, then write the Y coordinate to Y+ using the DAC, hold for 200 microseconds, then pull it down to ground again. Write 0V to 1.8V for coordinate 0 to 240.

The timing should look like this:



Yellow is Y+, blue is X+

Now you have all the control wires of 3DS, all that's left to do is hook them up to an microcontroller and have it control it for you! Please take a look at the eagle file for the control board.

3. The Control Board

All the wires that came out of 3DS end up in the control board, let at the mercy of the microcontroller, in this case a Teensy 3.1. You can use plain old Arduino as well, but make sure it has enough pins. Also notes that USB serial on Arduino is not as fast as 12Mbit/s full speed USB serial on Teensy 3.1.

The size of the board is quite large for what it is, I did this because of the limitation of the milling machine I'm using, so do feel free to shrink it down.

The circuit for the control board is pretty straightforward as well. All the buttons are active low, so pull them to ground to press them. One problem is the logic levels. 3DS uses 1.8V, while teensy uses 3.3V. I used a 74540 buffer as a level shifter. Power the chip with 1.8V, send 3.3V signal in, 1.8V signal comes out of the other end, and to 3DS.

Teensy 3.1 has a real DAC on A14, as well as a number of PWM pins. I used the real DAC to control the Y+ of touch screen, since it requires very fast response. PWM output can be filtered to create a DAC as well, albeit the tradeoff between rise time and high frequency component left in the output. I used 4 PWM DACs for circle pad X, circle pad Y, touch screen X(hence the

slow rise time in blue channel in the above picture), as well as powering the logic switching chip.

I have also added a header for SPI and hardware serial pins, should you need to use those communication protocols.

3. Commands

Command	Example	Notes
cn x y	cn 127 200	Circle pad nudge. Pushes circle pad to x, y, hold it there for c_pad_nudge_delay_ms, then release. x, y should be less than 255 and greater than 0. 127 is neutral position, greater than 127 for left/up, smaller than 127 for right/down.
ch x y	ch 20 127	Circle pad hold. Hold circle pad at x, y. x, y range is same as circle pad nudge.
cr	cr	Circle pad release. Return circle pad to default location.
cd c_pad_nudge_delay_ms	cd 200	Circle pad set nudge delay. Set how long the circle pad is held before being released when circle pad nudge is called.
tc x y	tc 133 70	Touch screen click. Press down on touch screen at x, y, hold it there for touch_screen_click_delay_ms, then release. x should be within 320 y should be within 240

td touch_screen_click_delay_ms	td 200	Touch screen set click delay. Set how long the touch screen is held before releasing when touch screen click is called.
bc [button]	bc a	Button click. Press down a button, hold it for button_click_delay_ms, then release. See next table for argument of buttons.
bh [button]	bh a	Button hold. Press down a button
br [button]	br a	Button release. Release a button
br	br	Button release all. Release all buttons. Note that this command has no arguments.
bd button_click_delay_ms	bd 100	Button set click delay. Set how long a button is held down before being released when button click is called.

3DS Button	Argument for bc, bh and br:
A	a
B	b
X	x
Y	y
D-Pad Up	u
D-Pad Down	d
D-Pad Left	l
D-Pad Right	r
Left Shoulder	ls

Right Shoulder	rs
Start	st
Select	sl
Home	h