TAS Console Replay Device

Exploring the accuracy of retro console emulation

Cody Morrison

University of Calgary, Mathematics & Statistics

cody.morrison1@ucalgary.ca

Abstract

Driven by a desire to improve the accuracy of emulation of retro consoles, specifically the Nintendo Entertainment System (NES) and Super NES (SNES), the project built a pair of controllers (NES and SNES respectively) using an arduino to play back controller inputs that had been produced using an emulator for the console. The goal is to verify all of the current tool assisted gameplay currently available on actual console hardware, and if it is unverifiable determine whether it is an issue with emulation or something else and improve the accuracy of console emulation. It was found that for a small subset of software available it is not difficult to replay the inputs, however there is a large issue with the quality of clock signal generation on the SNES and various hardware quirks for both the NES and SNES that introduced additional complexity.

Main Objectives

1. Understand NES and SNES controller signal encodings via observation with oscilloscope and digital logic analyser
2. Program an Arduino to emulate the console controller.
3. Convert TAS controller encoding into compatible encoding to transit via UART
4. Test input on console.
5. Sit back and watch the games beat themselves with machine like accuracy.
6. At the inevitable desync add/remove frames as necessary then goto 4 .
7. If game is unable to be beaten on console document and determine whether it's related to one of the hardware issues or if it's an emulation issue. If emulation issue then report to emulator author else investigate hardware further

Hardware Issues

NES

The NES makes use of DMC (delta modulation channel) to play back audio samples from memory, however due to a hardware issue in the 2A03 when using Direct Memory Access corruption of controller data can occur. From a game perspective several solutions arose such as Super Mario Bros. 3 (SMB3) reading the controller twice, if the input matches then it passes control back to the main game cycle, else it reads another pair of inputs. The solution for this project was to implement a window mode for controller latches, such that the arduino would wait until there had been at least X milliseconds between latches to occur. SMB3 required an approximately 9ms window, whilst Rescue Rangers required a 4ms window. Brute force search is unfortunately still the most effective way to determine windowing size.

SNES

Due to the use of a ceramic oscillator for the APU, a quartz oscillator for the CPU and the fact that some SNES games wait for certain audio signals to finish before the CPU advances frames have resulted in difficulty getting most SNES games to sync beyond a few seconds. Several options have been attempted to increase the chances of syncing, including replacing the ceramic oscillator with a high quality quartz oscillator, however none of have been successful as of yet.

SNES Successses

A limited number of games including but not limited to

- Super Mario World
- Mega Man X
- Super Metroid

Are known to not be afflicted, or have only a limited number of triggers affected by the above issue, allowing for short runs to be completed on hardware.

Hardware

Both replay devices are based on an Arduino uno, making use of 74h595c SIPO and 4021 PISO shift registers. The 595 was selected due to being a well documented and commonly used shift register, and natively supported chaining which allowed for easy extension for multiple controllers on the NES as well as to the SNES replay device. Where as the 4021 was selected because it is the same shift register used in original NES and SNES controllers which allowed for easier integration.

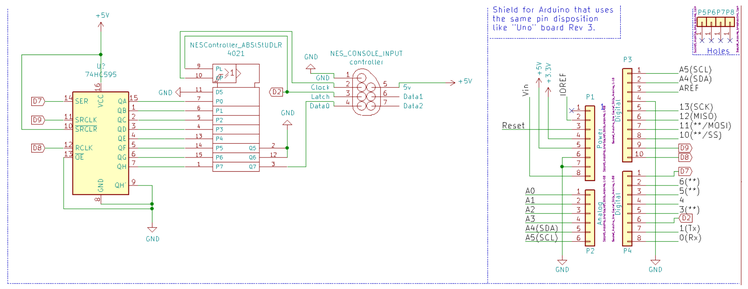


Figure 1: The NES replay device was built using an Arduino uno, 1 74h595c SIPO shift register, and 1 4021 PISO shift register. The 4021 was connected directly to the NES via 5 wires, $D0$. V_{cc}, GND, $LATCH$, $CLOCK$ in the same manner as a normal controller, however as detailed later a 7 wire setup including D1 and D2 would be much more functional. The 595 was connected so that the Most Significant Byte (MSB) was connected to the MSB of the 4021.

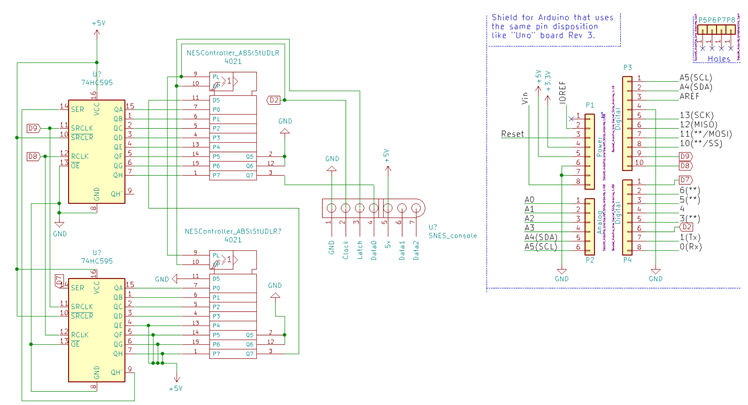
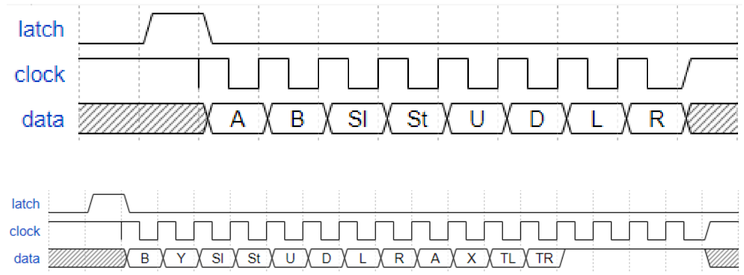


Figure 2: The SNES replay device was built in a similar manner, using 2 74h595c registers, and 2 4021 registers. They were daisy chained together, and the SNES has the same pins on the controller adaptor as the NES.

Controller Protocols

The arduino was programmed to accept information over a serial connection, and the controller latch wire was connected to an interupt to update the controller state. A high level overview of the standard NES controller protocol and timing: Ideally once per frame the NES sends a 12μs long pulse on the latch line, then pulses the clock line 8 times and reads HIGH or LOW on the data line, and as shown in the lower picture the standard SNES controller is similar, with 16 clock pulses of which the last 4 are always high which are identification bits and always high for a standard controller.



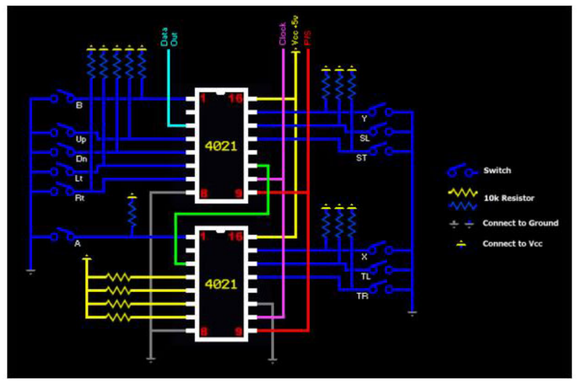


Figure 3: The SNES controller was built in a similar manner to the NES controller using 2 4021 PISO shift registers, while the NES used 1. The SNES has the same pins on the controller as the NES, however in a different configuration. Use of an adaptor will allow a SNES controller to be used on a NES.

Acknowledgements

The TASBot discord channel #tasbot-dev was exceptionally helpful in researching and prototyping of the arduino solution. The NESDev wiki, http://wiki.nesdev.com/w/index.php/Nesdev_Wiki/. The SFC Dev Wiki, https://wiki.superfamicom.org/. The Higan/BSNES source code for investigating desync issues on SNES, available at https://byuu.org/. dwangoAC's github repository for TASBOT-Projects, https://github.com/dwangoac/TASBot-Projects/.