Pokémon Forever: A Game Boy™ Save State Reader/Writer

Adam Lux[1]

[1]Department of Physics, Boston University, Boston, MA 02215, USA

April 29th, 2021

Abstract

The Nintendo Game Boy™ was a popular gaming system in the 1990's. Even today many people play Game Boy™ games, either on emulators or original hardware. There were — and still are — a number of technical limitations of the game system, and its associated games, one being the use of volatile SRAM (static random-access memory) to save game progress in many of the console's game cartridges. This SRAM requires a battery to keep it powered while the system is turned off, and these batteries often run out after a number of years. They can be replaced but all save data is destroyed when the battery can no longer power the SRAM. We intend to develop a solution to this shortcoming by using an Arduino to save data from a Game Boy™ cartridge to external non-volatile memory, then re-write it back onto the cartridge after a battery change.

1 Introduction

The Nintendo Game Boy™ was invented by Gunpei Yokoi and released in April 1989. It is an 8-bit portable gaming system that supported a multitude of games by way of interchangeable cartridges (carts). Along with its successor the Game Boy Color™, the Game Boy™ is currently the 3rd best selling game console of all time. Many fans still enjoy playing Game Boy™ games to this day, and some still prefer original hardware over digital emulators. However, using older hardware comes with some disadvantages. One such disadvantage is the use of volatile memory in many of the game carts. This volatile memory requires power to keep data stored. Game Boy™ carts often have batteries inside them to power this memory, but those batteries eventually run out. For some games, like Pokémon Gold, Silver, and Crystal versions, the batteries only last a few years due to an internal always-on clock feature which uses more power than other carts while the system is powered off. When one of these batteries dies, all of a player's progress is lost, and no more progress can be saved until the battery is replaced. If a player replaces one of these batteries before it dies, they would still lose all of their progress. We are proposing a solution to this technical shortcoming by using an Arduino system to save the data off of a Game Boy™ cart to non-volatile memory before a battery change and then to write the data back onto the cart after a new battery is installed.

2 Technical Concepts

2.1 Types of Memory

There are two general types of memory on any Game Boy™ cart: read-only memory (ROM) and random-access memory (RAM). ROM, as its name suggests, is intended not to be written to, but only read by the system. ROM is non-volatile and was used to store the game's logic and any static data like dialog lines and sprites. RAM, in particular SRAM, was volatile memory used to store dynamic data like how many items the character has, where they are located on the map, which levels have been beaten, etc. SRAM could be read from and written to by the system. Generally speaking, any data which changes as a player makes

progress in the game is stored in RAM, while the data which needs to be the same is stored in ROM. The SRAM is powered by a battery, which can be seen in the top right of Fig 2, when the system is not powered on. When this battery runs out the SRAM can no longer store data with the system off. Depending on what data is stored in SRAM, this could render the game completely unplayable, or simply make it so one can no longer save any progress.

2.2 Bank Switching

As one can see from Table 2.2, Game Boy™ carts have 16 address pins. If each address pin combination corresponded to one (1) byte (read from the eight (8) data pins), that would allow for a total of $2^{16} = 65,536$ bytes of memory that can be accessed. Even for 1989 this was not enough for many games. To overcome this obstacle, the carts use multiple "banks" of memory and chips called memory bank controllers (MBC). These chips allow the Game Boy™ system to switch between different banks of memory by using the chip select pin and writing a particular byte to the read pins. The MBC will then switch the address pins to point to the memory bank associated with whatever byte is written to the data pins and data can be read out normally. Not every cart has the same number and types of memory banks, and therefore also not the same MBC. The information for what memory chips and MBC are on a particular cart is stored in the games first ROM bank which is what the address pins point to by default. Creating an algorithm to figure out what memory chips and MBCs are on different carts, and then correctly accessing the necessary banks will be a challenge this project must overcome. Alex [2011]

3 Extra Necessary Components

Beyond the components provided in the Arduino Uno kit we will require the following to successfully develop this tool:

- 1 x Arduino Mega Buy Link
- 1 x Console which accepts and plays Game Boy™ and Game Boy Color™ games
- 2 x Sacrificial Game Boy™ games for testing
- 1 x Arduino SD card adapter board (or any other non-volatile memory compatible with the Arduino) Buy Link
- 1 x SD card (1 GB will be ample)
- 1 x Game Boy™ cartridge slot adapter Buy Link
- 1 x Game Boy™ game which needs it's data saved
- 1 x CR2025 battery
- 1 x soldering iron

4 Project Stages

1. Buy all necessary items
2. Construct necessary connection adapter between game cart and Arduino
3. Construct physical logic to talk to cart
4. Write code to access and copy SRAM

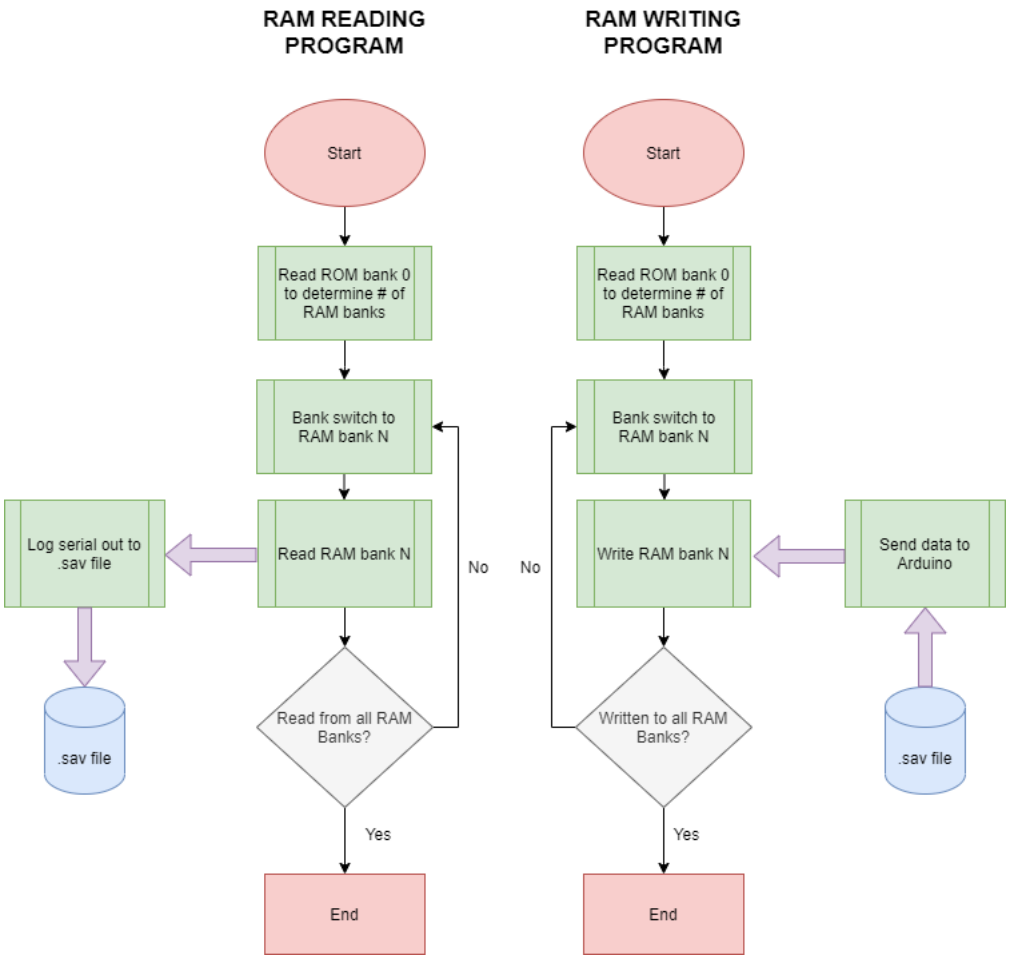

Figure 1: Proposed Software Flowchart

Pin #	Name	Description
1	V_{CC}	+5 V
2	?	?
3	/WR	Write
4	/RD	Read
5	CS	Chip Select
6	A0	Address 0
7	A1	Address 1
8	A2	Address 2
9	A3	Address 3
10	A4	Address 4
11	A5	Address 5
12	A6	Address 6
13	A7	Address 7
14	A8	Address 8
15	A9	Address 9
16	A10	Address 10
17	A11	Address 11
18	A12	Address 12
19	A13	Address 13
20	A14	Address 14
21	A15	Address 15
22	D0	Data 0
23	D1	Data 1
24	D2	Data 2
25	D3	Data 3
26	D4	Data 4
27	D5	Data 5
28	D6	Data 6
29	D7	Data 7
30	/RST	Reset
31	Audio	??
32	GND	Ground

Table 1: Game Boy™ Cart Pinout. Mollers [2019]

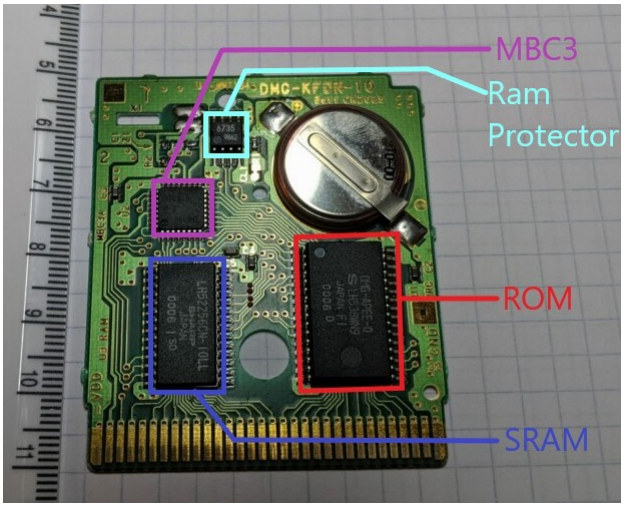

Figure 2: Logic board of Pokemon: Blue with labeled chips. Brendan [2019]

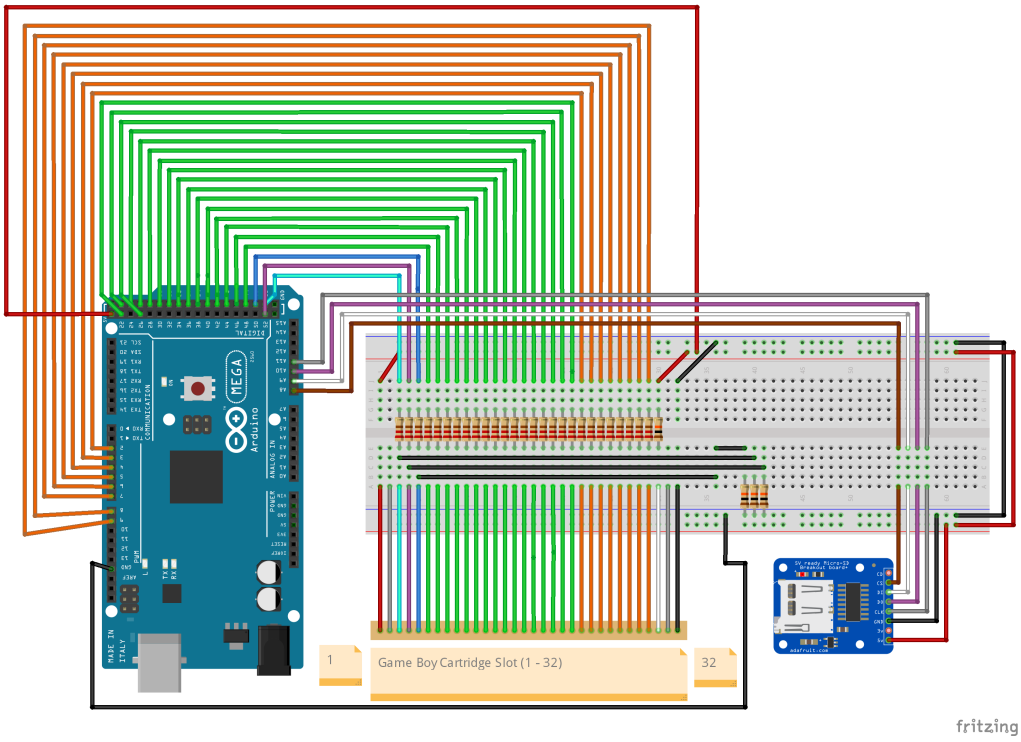

Figure 3: Proposed Schematic

5. Test on sacrificial carts
6. Write code to access and write data to SRAM
7. Test on sacrificial carts
8. Combine reading and writing codes and test on sacrificial carts
9. Add quality of life features (busy LED etc)
10. Final test on sacrificial carts
11. Use on non-sacrificial carts for a battery change

5 Conclusion

We are working to build the components, both hardware and software, necessary to read and write save files from a Game Boy™ cartridge using an Arduino. Once completed these tools can be used and improved by other parties such as enthusiasts, digital archivists, game developers, and artists to produce further works with the Game Boy™.

References

Alex. Arduino based gameboy cart reader, 2011. URL https://www.insidegadgets.com/2011/03/19/gbcartread-arduino-based-gameboy-cart-reader-%E2%80%93-part-1-read-the-rom/.

Brendan. Gameboy dmg rom and ram bank switching, 2019. URL https://b13rg.github.io/Gameboy-Bank-Switching/.

P. K. J. Mollers. Game boy cartridge pinout, 2019. URL https://old.pinouts.ru/Game/CartridgeGameBoy_pinout.shtml.