arduino-gameboy-printer-emulator

Code to emulate a gameboy printer via the gameboy link cable

View on GitHub

Arduino Gameboy Printer Emulator (V3)

Main project website located at https://mofosyne.github.io/arduino-gameboy-printer-emulator/

Telegram Gameboy Camera Chatroom

Got telegram instant messaging and have some questions or need any advice, or just want to share? Invite link below:

https://t.me/gameboycamera

Media Coverage And Other Projects Spinoff

There is more examples located in our **showcase page** page, but below is a few actively used cases:

Featured On Hack A Day Article

WestM's Arduino Gameboy Printer Emulator Tutorial

CristoferCruz : gbpxl (multi-tone) : Fork of gbpxl with multitone support (Here until merged into mainline)

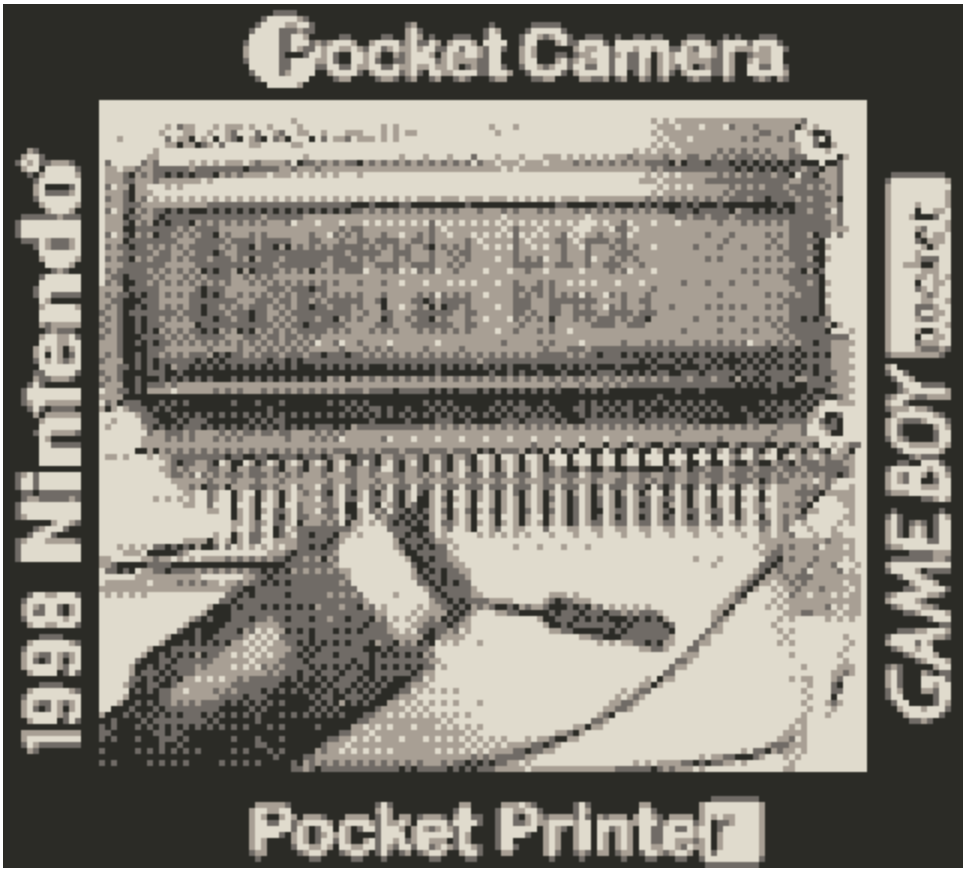
Raphaël BOICHOT : Game boy printer emulator with e-paper feature (CrapPrinter) for Matlab and Octave

HerrZatacke : A set of node js functions to decode the raw packet stream from a gameboy to gameboy printer

Click For More Examples In Our Showcase Page

About this project

Code to emulate a gameboy printer via the gameboy link cable and an arduino module



- Blog Post

Goal is to provide an easy way for people to quickly setup and download the images from their gameboy to their computer before the battery of these gameboy cameras dies of old age.

I hope there will be a project to collate these gameboy images somewhere.

Official Releases

Downloads: Version Release Downloads at GitHub

Release Notes Located Here

Quick Start

Construct the Arduino Gameboy Printer Emulator

Use an arduino nano and wire the gameboy link cable as shown below. If you can fit the gameboy camera to gameboy advance etc... you may need a differen pinout reference. But the wiring should be similar.

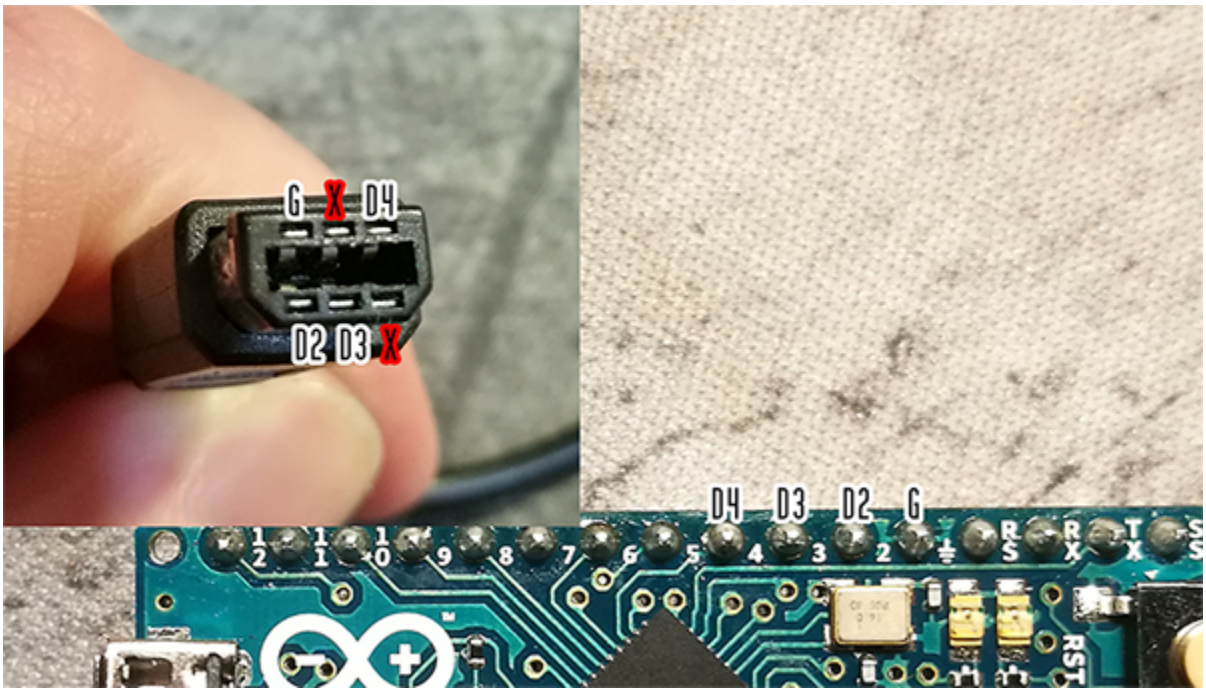
- Pinout Reference

You should avoid cutting old genuine gameboy link cables, there is plenty new cables you can purchase online. Do note that you cannot trust the color code of these cables, you must always check the wire against the plug pins. Especially considering the RX/TX pair of the pins may be flipped.

Else if you have a 3D printer, you can use (Game Boy DMG-01 Link Port plug for dupont jumper wire by Marko Štamcar from Slovenian Computer Museum, created as part of a retro tech exhibition)[https://www.thingiverse.com/thing:4685189]

Pinout Diagram

Thanks to West McGowan (twitter: @imwestm) who was able to replicate this project on his Arduino Nano plus Gameboy Color and helpfully submitted a handy picture of how to wire this project up. You can find his tutorial in here



General Pinout

```
Gameboy Original/Color Link Cable Pinout
```

```
 ___________
|  6  4  2  |
 \_5__3__1_/   (at cable)
```

Arduino Pin	Gameboy Link Pin
unused	Pin 1 : 5.0V
D4	Pin 2 : Serial OUTPUT
D3	Pin 3 : Serial INPUT
unused	Pin 4 : Serial Data
D2	Pin 5 : Serial Clock (Interrupt)
GND	Pin 6 : GND (Attach to GND Pin)

Programming the emulator

- Arduino Project File: `./GameBoyPrinterEmulator/gpb_emulator.ino`
- Baud 115200 baud

Next download `./GameBoyPrinterEmulator/gpb_emulator.ino` to your arduino nano. After that, open the serial console and set the baud rate to 115200 baud.

Download the image

Press the download button in your gameboy. The emulator will automatically start to download and dump the data as a string of hex in the console display.

After the download has complete. Copy the content of the console to the raw packet javascript decoder in `./GameBoyPrinterDecoderJS/gameboy_printer_js_raw_decoder.html` . Press click to render button.

One you done that, your image will show up below. You can then right click on the image to save it to your computer. Or you can click upload to imgur to upload it to the web in public, so you can share it. (Feel free to share with me at mofosyne@gmail.com).

A copy of the raw decoder is accessible here as well:

- V3 Raw JS Decoder: Click Here To Open Javascript Gameboy Printer Emulator Web Decoder

Need example raw packet captures to test out the raw js decoder without the gameboy printer emulator hardware? You can check a few out in the Real Packet Capture Example folder of the gameboy printer sniffer project.

You are all done!

Note: V3 now uses raw packet decoder, rather than the original tile decoder. This allows us to better support gameboy printers enabled games using compression.

Project Makeup

- Arduino sketch emulating a gameboy printer to a computer via a serial link.
  - `./GameBoyPrinterEmulator/gpb_emulator.ino` : Main source file
  - `./GameBoyPrinterEmulator/gameboy_printer_protocol.h` : Reusable header containing information about the gameboy protocol
  - The serial output is outputting a gameboy tile per line filled with hex. (Based on http://www.huderlem.com/demos/gameboy2bpp.html) Only if in tile output mode.
  - If set to tile mode, then a tile in the serial output is 16 hex char per line: e.g. `55 00 FB 00 5D 00 FF 00 55 00 FF 00 55 00 FF 00`
  - If set to raw mode, it will output the raw packet in hex, where last two bytes of each packet is the printer's response: e.g. `88 33 01 00 00 00 01 00 81 00`
- Javascript gameboy printer hex encoded packets stream rendering to image in browser.
  - js decoder page
  - `./GameBoyPrinterDecoderJS/gameboy_printer_js_raw_decoder.html` :
  - This is a more complicated but standard compliant way to convert the packet hex payload into canvas image that can be downloaded.
- Javascript gameboy printer hex tiles stream rendering to image in browser.
  - js decoder page
  - `./GameBoyPrinterDecoderJS/gameboy_printer_js_decoder.html` :
  - This is a convenient way to convert the hex payload into canvas image that can be downloaded.
- Research folder.
  - Contains some files that I found online to research this project
  - Also a sniffer to listen to the communication between a real gameboy and a real printer that was now moved to https://github.com/mofosyne/GameboyPrinterSniffer instead
- Sample Images
  - Some sample images along with the sample logs. There has been some changes to the serial output, but the hex format remains the same.
- showcase

- Example of what other people has used this project for.

Technical Information

Sniffer / Real Gameboy Printer Captures

Refer to https://github.com/mofosyne/GameboyPrinterSniffer for more information.

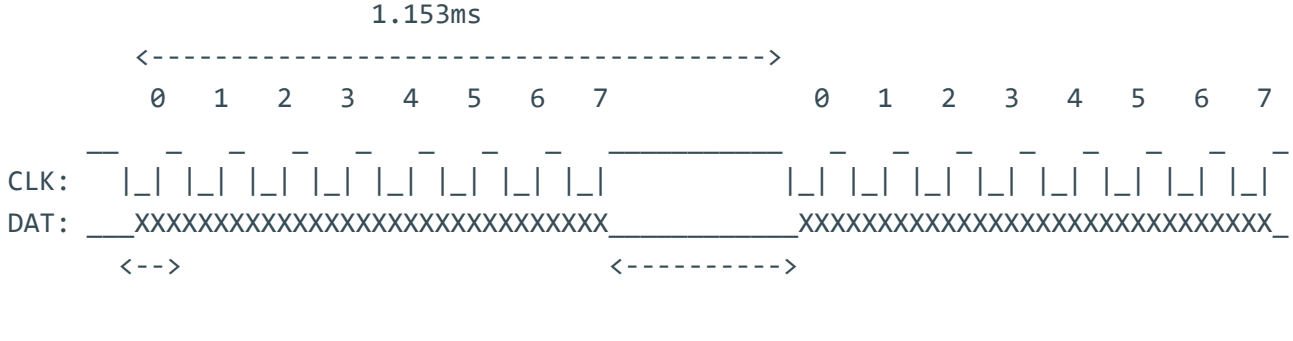
Protocol

BYTE POS :	0	1	2	3	4	5
SIZE	2 Bytes		1 Byte	1 Byte	1 Bytes	1 Byte
DESCRIPTION	SYNC_WORD		COMMAND	COMPRESSION	DATA_LENGTH(X)	
GB TO PRINTER	0x88	0x33	See Below	See Below	Low Byte	High By
TO PRINTER	0x00	0x00	0x00	0x00	0x00	0x00

- Header is the Command, Compression and Data Length
- Command field may be either Initialize (0x01), Data (0x04), Print (0x02), or Inquiry (0x0F).
- Compression field is a compression indicator. No compression (0x00), Yes Compression (0x01)
- Payload byte count size depends on the value of the `DATA_LENGTH` field.
- Checksum is 2 bytes of data representing the sum of the header + all data in the data portion of the packet
- Status byte is a bitfield byte indicating various status of the printer itself. (e.g. If it is still printing)

Gameboy Printer Timing

Below measurements was obtained via the ANALOG DISCOVERY via digilent



```
127.63 us                       229.26 us
```

- Clock Frequency: 8kHz (127.63 us)
- Transmission Speed: 867 baud (1.153ms per 8bit symbol)
- Between Symbol Period: 229.26 us

Research/Dev log

2021-01-26

- V3 enable raw only mode to take advantage of better decompression on PC side
- Updated and added a showcase folder

2020-08-30

- V2 rewrite completed for more game support
- Sniffer created https://github.com/mofosyne/GameboyPrinterSniffer

2017-11-30

- Time to wrap this up. I have pushed the arduino to it's maxium ability. The show stopper to futher dev is the puny ram.
- Still cannot get CRC working. But don't really care anymore. Since its taking too much time to debug crc. Works good enough.
- There is not enough time/ram to actually do much processing of the image in the arduino. Instead the image data has to be transferred raw to over serial at 115200baud via hex, and processed futher in the computer.

2017-4-12

- Checksum works for init, inqiry, but not data, and possibly inqury. Possibly messed up the summation somehow. But I found that this may not matter, as the gameboy camera doesn't seem to check for it.
- While checksum worked for short messages, the checksum system didn't work for the normal data bytes. But... it seems that the gameboy printer doesn't actually pay attention

to the checksum bit.
- Investigation with http://www.huderlem.com/demos/gameboy2bpp.html shows that this is actually encoded as “16 byte standard gameboy tile format”.
- According to http://furrtek.free.fr/?a=gbprinter&i=2 the `The GameBoy Camera buffers tile data by blocs of 2 20 tiles wide lines.` .

My Face In BW

Credits / Other Resources

Resources Referenced

- GameBoy PROGRAMMING MANUAL Version 1.0 DMG-06-4216-001-A Released 11/09/1999
  - Is the original programming manual from nintendo. Has section on gameboy printer. Copy included in research folder.
- http://www.huderlem.com/demos/gameboy2bpp.html Part of the js decoder code is based

on the gameboy tile decoder tutorial
- https://github.com/gism/GBcamera-ImageSaver - Eventally found that someone else has already tackled the same project.
    - However was not able to run this sketch, and his python did not run. So there may have been some code rot.
    - Nevertheless I was able to get some ideas on using an ISR to capture the bits fast enough.
    - I liked how he outputs in .bmp format
- http://gbdev.gg8.se/wiki/articles/Gameboy_Printer - Main gb documentation on gb protocol, great for the inital investigation.
- http://furrtek.free.fr/?a=gbprinter&i=2 - Previous guy who was able to print to a gameboy printer
- http://playground.arduino.cc/Main/Printf - printf in arduino
- https://www.mikrocontroller.net/attachment/34801/gb-printer.txt
    - Backup if above link is dead here
    - Most detailed writeup on the protocol I found online.
- http://gbdev.gg8.se/wiki/articles/Serial_Data_Transfer_(Link_Cable)
- https://github.com/avivace/awesome-gbdev Collections of gameboy development resources
- https://shonumi.github.io/articles/art2.html An in-depth technical document about the printer hardware, the communication protocol and the usual routine that games used for implementing the print feature.

Contributors / Thanks

- @BjornB2 : For adding improvements in downloading images for jsdecoder folder https://github.com/mofosyne/arduino-gameboy-printer-emulator/pull/15 For adding color palette dropdown https://github.com/mofosyne/arduino-gameboy-printer-emulator/pull/18
- @virtuaCode : For helping to fix rendering issues with the jsdecoder https://github.com/mofosyne/arduino-gameboy-printer-emulator/pull/9
- @HerrZatacke : For adding the feature to render a separate image for each received image in jsdecoder https://github.com/mofosyne/arduino-gameboy-printer-emulator/pull/19
- @imwestm : West McGowan for submitting a handy picture of how to wire this project up as well as feedback to improve the instructions in this readme.
- Raphaël BOICHOT : For assistance with capturing gameboy communications and timing for

support with gameboy printer fast mode and compression. Assisting in the support of more games. Also contributed a matlab/octave decoder implementation.

- @crizzlycruz (@23kpixels) : For adding support to js decoder for zero margin multi prints
- @markostamcar : For contributing a 3D printed plug so users can construct a cable without sacrifing a gameboy link cable.

---

arduino-gameboy-printer-emulator is maintained by mofosyne.
This page was generated by GitHub Pages.