FAKULTÄT ELEKTROTECHNIK UND WIRTSCHAFTSINGENIEURWESEN

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN LANDSHUT

Projektbericht im Modul Eingebettete autonome Systeme II zum Thema

Entwicklung einer Raspberry Pi basierten Handheld-Konsole mit eingebettetem GameBoy Cartridge Reader I

Autor:	Felix Weichselgartner
Betreuer:	Prof. Dr.-Ing. Mathias Rausch
Eingereicht:	28.12.2020
Aktualisierte Version:	24.06.2021

Hiermit erkläre ich, dass ich die Arbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Tiefenbach, 28.12.2020 Felix Weichselgartner

Acknowledgments

I want to thank the Open Source and GameBoy community,
who made this project possible.

Abstract

A popular project in the internet community is the „GameBoy Zero“. Here a Raspberry Pi Zero [1] (with buttons, battery, power electronics, display, etc.) is integrated in the shell of an original GameBoy. This result in a RPi based handheld game console, that can emulate games from various systems. More about the history of the project can be found here [2]. The new contribution to the project is the embedded cartridge reader, which makes it possible to interact with real GameBoy cartridges. Therefore, it is not neccessary to download ROMs from dubious websites and you are able to use your real GameBoy cartridges and emulate them, all in one device.

In this work, the design of a replacement GameBoy mainboard is presented. This pcb has a cartridge adapter on it, which is accessible by gpio port expanders, which can be controlled over the I²C-interface. Furthermore, the code for interfacing with a GameBoy cartridge will be developed.

Inhaltsverzeichnis

Acknowledgments **ii**

Abstract **iii**

1. Motivation **1**
1.1. Problemstellung 1
1.2. Zielsetzung 1

2. Grundlagen **2**
2.1. GameBoy 2
2.1.1. Funktionsweise 2
2.1.2. GameBoy Catridges 3
2.1.3. Cartridge Schnittstelle 8
2.2. GPIO Port Erweiterung 9

3. Implementierung **12**
3.1. Hardware 12
3.2. Software 14

4. Testen **19**

5. Ergebnisse **23**

Literaturverzeichnis **25**

Abbildungsverzeichnis **27**

Tabellenverzeichnis **28**

A. Anhang - Testprotokoll **30**

1. Motivation

1.1. Problemstellung

Ende 2018 wurde ich auf ein populäres Elektronikprojekt namens „GameBoy Zero“ [2] aufmerksam. Im Laufe der nächsten Monate, baute ich meine eigene Version [3], womit ich schlussendlich verschiedene Systeme auf einem Raspberry Pi Zero [1] emulieren konnte. Der Haken ist jedoch, sofern man nicht seine eigenen GameBoy-Spiele auslesen kann, ist man auf sogenannte ROMs aus dem Internet angewiesen. Diese beinhalten den ausgelesenen ROM-Speicher der GameBoy-Cartridges, womit das Spiel emuliert werden kann [4]. Dies führte dazu, dass ich Mitte 2019 einen Catridge-Reader [5] baute, welcher es mir ermöglichte, meine eigenen Catridges auszulesen. Hierbei kann aber nicht nur der ROM ausgelesen werden, sondern auch der RAM, welche die Speicherstände der Spiele enthält. Das Problem hierbei ist jedoch, dass dies zwei separate Geräte und nicht wie beim originalen GameBoy in einem Gerät vereint sind.

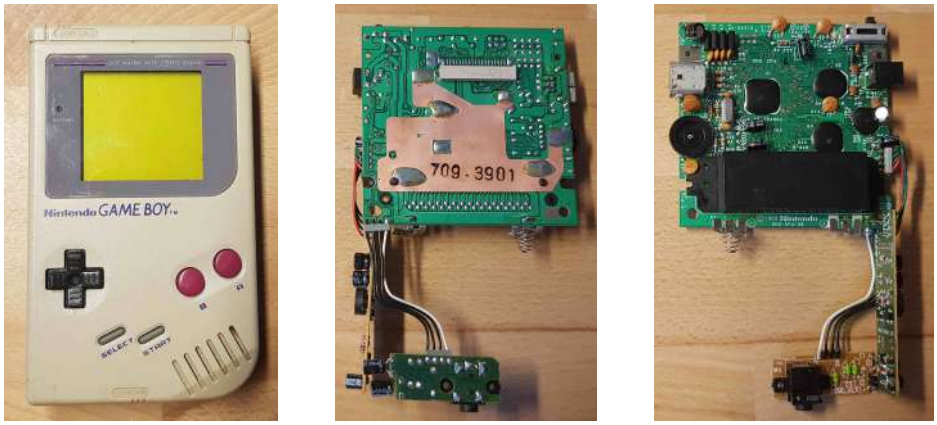
1.2. Zielsetzung

Der neue Beitrag zu dem „GameBoy Zero“-Projekt ist somit das Erstellen einer Platine, welche es erlaubt eine Cartridge über einen „GameBoy Zero“ auszulesen und die Entwicklung der benötigten Software. Da ein Raspberry Pi nicht genug Pins hat, um neben dem Cartridge-Adapter auch noch Inputs etc. zu erkennen, werden die GPIO-Pins mit einem MCP23017 Port Expander [6] erweitert. Damit ein sauberer und ordentlicher Einbau der Platine möglich ist, soll diese in den physischen Abmessungen dem Mainboard des originalen GameBoys entsprechen. Der Cartridge-Adapter hat somit dieselbe Position wie beim originalen GameBoy und kann über die Schnittstelle des MCP23017 angesprochen werden.

2. Grundlagen

2.1. GameBoy

Im Laufe der Jahre erschienen verschiedene Versionen des GameBoys. Der erste GameBoy hatte die Modellnummer DMG-01 und wird daher oft kurz als DMG bezeichnet. Andere Bezeichnungen sind „Originaler GameBoy“ oder auch „Klassischer GameBoy“. Die Bezeichnung DMG bezieht sich hierbei auf „Dot Matrix Game“, womit das Display des GameBoys gemeint ist. In Abbildung 2.1 (a) wird das Aussehen des DMG gezeigt. Beim Zerlegen des DMG erhält man zwei Hälften. Die Platine der untere Hälfte wird in Abbildung 2.1 (b) und (c) gezeigt. In diesem Dokument wird sich, sofern nicht anders spezifiziert, auf den DMG bezogen. In [7] wird der Hardware-Aufbau eines DMG genau analysiert.



(a) Frontansicht (b) Logikboard Oberseite (c) Logikboard Unterseite

Abbildung 2.1.: DMG Frontansicht und Mainboard

2.1.1. Funktionsweise

In dem folgenden Blockschaltbild (Abbildung 2.2) wird die grundsätzliche Funktionsweise eines DMG gezeigt.

https://gbdev.gg8.se/wiki/images/0/03/Gameboy1-cpuboard.gif

Abbildung 2.2.: Blockschaltbild des DMG [8]

Hier fällt auf, dass der GameBoy mit einer Adressbreite von 16 Bits (A0-A15) und einer Datenbreite von 8 Bit (D0-D7) arbeitet (sprich 1 Byte). Die Aufteilung des Adressbereichs in Read Only Memory (ROM), Random Access Memory (RAM) und internen Speicher wird in Tabelle 2.1 gezeigt. Das Verhalten der Cartridge auf Lese- und Schreibbefehle über den Memory Bank Controller (MBC) wird in Kapitel 2.1.2 erklärt.

Start	Ende	Funktion
0000	3FFF	16KB ROM Bank 00 von der Cartridge (i. d. R. fest)
4000	7FFF	16KB ROM Bank 01-N von der Cartridge, kann über MBC gewechselt werden (falls MBC vorhanden)
8000	9FFF	8KB Video RAM (VRAM), bei DMG nur Bank 0
A000	BFFF	8KB External RAM von der Cartridge, falls vorhanden
C000	CFFF	4KB Work RAM (WRAM) Bank
D000	DFFF	4KB Work RAM (WRAM) bei DMG feste Bank 1
E000	FDFF	Kopie von C000-DDFF (ECHO RAM), wird i. d. R. nicht benutzt
FE00	FE9F	Sprite attribute table (OAM)
FEA0	FEFF	Nicht benutzbar
FF00	FF7F	I/O Registers
FF80	FFFE	High RAM (HRAM)
FFFF	FFFF	Interrupts Enable Register (IE)

Tabelle 2.1.: Memory Map des DMG [8]

Die Bereiche 0000-7FFF für ROM und A000-BFFF für externen RAM zeigen hierbei auf externen Speicher auf der Catridge. Auf diese wird in Kapitel 2.1.2 genauer eingegangen.

2.1.2. GameBoy Catridges

Grundlegend ist festzuhalten, dass alle DMG-Cartridges mit einer Spannung von 5 V arbeiten. Des Weiteren kann sich je nach Cartridge auf dieser ein sog. MBC befinden. Dieser ermöglicht es, durch eine Schreiboperation in den reinen Lesebereich (dies wird in [9] ausführlicher erklärt), einen Speicherblock gegen einen anderen zu tauschen. Dadurch kann im sehr begrenzten Speicherbereich wesentlich mehr Speicher zur Verfügung zu gestellt werden. Natürlich kann immer nur auf eine Bank gleichzeitig zugegriffen werden.

Damit der GameBoy weiß, um was für eine Cartridge es sich handelt, gibt es den sog. Cartridge Header (siehe Tabelle 2.2). Dieser befindet sich am Anfang der ROM im Bereich 0100-014F, direkt hinter den Interruptvektoren. In diesem Bereich wird unter anderem der MBC (siehe Tabelle 2.3), die ROM-Größe (siehe Tabelle 2.4) und die RAM-Größe (siehe Tabelle 2.5) spezifiziert. Diese Werte sind essenziell zum Auslesen der Cartridge, da sie die Größe und das Verhalten der Cartridge kennzeichnen.

Start	Ende	Funktion
0100	0103	Einstiegspunkt i. d. R. mit "no operation"(NOP) und Sprung zum Programmstart
0104	0133	Nintendo Logo
0134	0143	Name des Spiels
0144	0145	Publisher des Spiels
0146	0146	Super GameBoy Flag
0147	0147	Cartridge Typ (spezifiziert welcher MBC genutzt wird)
0148	0148	ROM Größe (spezifiziert die Anzahl der vorhandenen ROM Banken)
0149	0149	Externe RAM Größe (spezifiziert die Anzahl der vorhandenen RAM Banken
014A	014A	Ländercode
014B	014B	I/O Registers
014C	014C	High RAM (HRAM)
014D	014D	Header Checksumme
014E	014F	Globale Checksumme

Tabelle 2.2.: Memory Map des Cartridge Headers [10]

Mit Tabelle 2.3 können die auf der Cartridge verbauten Bauteile erkannt werden. „MBCX“ gibt hier den verbauten MBC an, „RAM“ bedeutet, dass ein externer RAM auf der Cartridge verbaut wurde und „Batterie“ bedeutet, dass die Spannung am RAM durch eine Batterie gehalten wird. Folglich lohnt es sich, den RAM einer Cartridge auszulesen, wenn eine Batterie vorhanden ist, da hier den entsprechende Spielstand des Nutzers abgelegt wurde.

Wert	Bedeutung
00	kein MBC, ROM ist fest
01	MBC1
02	MBC1 + RAM
03	MBC1 + RAM + Batterie
05	MBC2
06	MBC2 + Batterie
08	ROM + RAM
09	ROM + RAM + Batterie
0B	MMM01
0C	MMM01 + RAM
0D	MMM01 + RAM + Batterie
0F	MBC3 + TIMER + Batterie
10	MBC3 + TIMER + RAM + Batterie
11	MBC3
12	MBC3 + RAM
13	MBC3 + RAM + Batterie
19	MBC5
1A	MBC5 + RAM
1B	MBC5 + RAM + Batterie
1C	MBC5 + RUMBLE
1D	MBC5 + RUMBLE + RAM
1E	MBC5 + RUMBLE + RAM + Batterie
20	MBC6
22	MBC7 + SENSOR + RUMBLE + RAM + Batterie
FC	POCKET CAMERA
FD	BANDAI TAMA5
FE	HuC3
FF	HuC1 + RAM + Batterie

Tabelle 2.3.: Cartridge Header: Cartridge Typ [10]

Die Tabelle 2.4 gibt die Anzahl der ROM-Banken und Tabelle 2.5 die Anzahl der RAM-Banken an. Diese Werten werden später auf der Softwareebene (Kapitel 3.2) verwendet, um in einer Schleife den Cartridge-Speicher auszulesen.

Wert	Bedeutung
00	nur die 2 festen Banken → 32 KB Speicher
01	4 Banken → 64 KB Speicher
02	8 Banken → 128 KB Speicher
03	16 Banken → 256 KB Speicher
04	32 Banken → 512 KB Speicher
05	64 Banken (bei MBC1 nur 63 genutzt) → 1 MB Speicher
06	128 Banken (bei MBC1 nur 125 genutzt) → 2 MB Speicher
07	256 Banken → 4 MB Speicher
08	512 Banken → 8 MB Speicher
52	72 Banken → 1.1 MB Speicher
53	80 Banken → 1.2 MB Speicher
54	96 Banken → 1.5 MB Speicher

Tabelle 2.4.: Cartridge Header: ROM Größe [10]

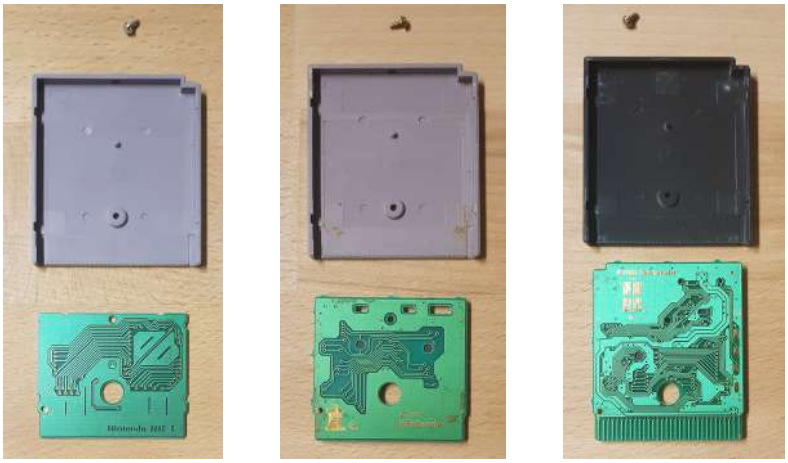
Wert	Bedeutung
00	Kein externer RAM vorhanden
01	1 Bank mit 2 KB (nutzt nicht den vollen Adressbereich aus)
02	1 Bank mit 8 KB
03	4 Banken → 32 KB Speicher
04	16 Banken → 128 KB Speicher
05	8 Banken → 64 KB Speicher

Tabelle 2.5.: Cartridge Header: RAM Größe [10]

Zum Testen wurden in diesem Projekt hauptsächlich 3 Cartridges (siehe Abbildung 2.3 und 2.4) mit verschiedenen MBC verwendet: Ohne MBC, MBC1 und MBC5.

(a) kein MBC (b) MBC1 (c) MBC5

Abbildung 2.3.: Frontansichten der verwendeten Cartridges
(a) Tetris, (b) Yoshi's Cookie, (c) Pokémon Trading Card Game (TCG)



(a) kein MBC (b) MBC1 (c) MBC5

Abbildung 2.4.: Rückansichten der verwendeten Cartridges
(a) Tetris, (b) Yoshi's Cookie, (c) Pokémon TCG

Eine genauere Analyse einer großen Menge an Cartridges und MBC kann hier [11] gefunden werden.

2.1.3. Cartridge Schnittstelle

In Tabelle 2.1 ist zu erkennen, dass sich der Programmspeicher auf der Cartridge (ROM) im Bereich 0000-7FFF befindet und der Cartridge RAM im Bereich A000-BFFF. Bei beiden können je nach MBC (wie in Kapitel 2.1.2 bereits erklärt) unterschiedlich viele Banken vorhanden sein.

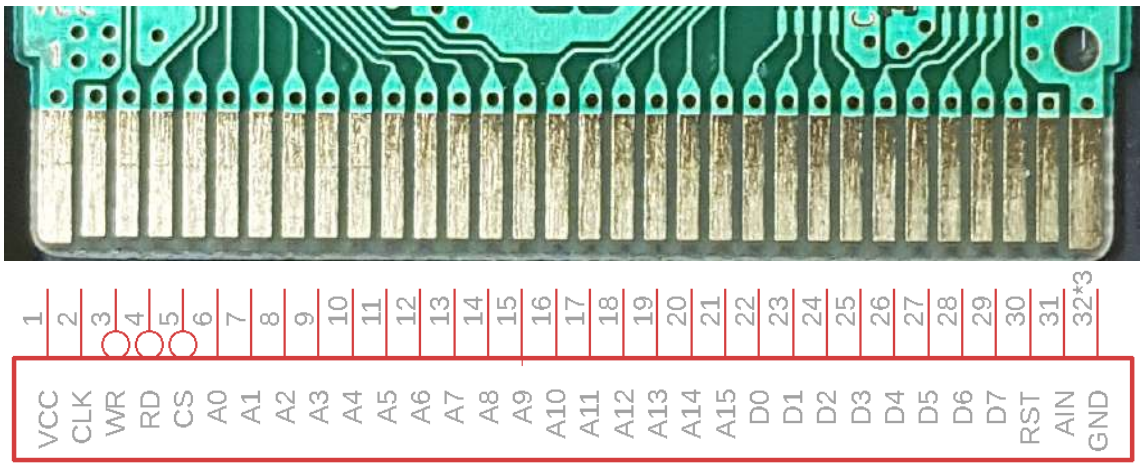


Abbildung 2.5.: Cartridge Pinout

Diese Cartridge wird nun in einen Cartridge-Adapter gesteckt, der die Platine und die Catridge elektrisch verbindet. Der in Abbildung 2.6 gezeigte Cartridge-Adapter ist eigentlich für den NintendoDS produziert und daher in SMD-Bauweise ausgeführt worden, besitzt aber dieselbe Funktion wie die THT-Variante des DMG.

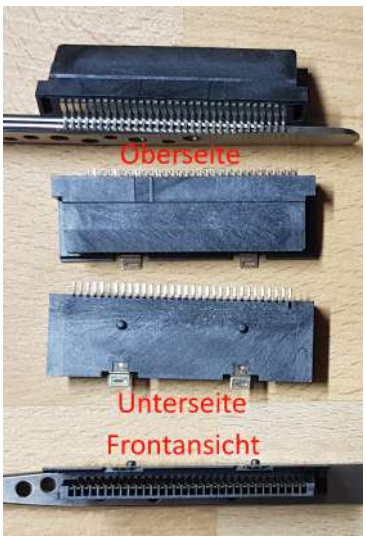


Abbildung 2.6.: Cartridge Adapter

2.2. GPIO Port Erweiterung

Wie in Abbildung 2.5 zu erkennen ist, werden zum Auslesen einer Cartridge 27 Pins (16 Adresspins + 8 Datenpins + 3 weitere Pins = 27 Pins) benötigt. Daher wird zur Erweiterung der GPIO-Pins ein GPIO-Port-Expander (MCP23017 [6]) verwendet. Die Pins des ICs können über das I²C-Protokoll beschrieben bzw. gelesen werden.

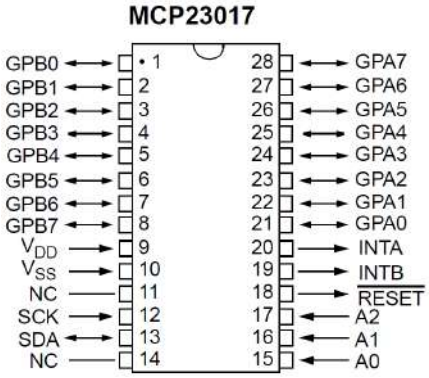


Abbildung 2.7.: Pinout des MCP23017 [6]

Der Pin VDD wird an 3.3 V verbunden und der VSS Pin an GND. SDA und SCL werden entsprechend an die I²C-Pins des Raspberry Pi angeschlossen. Wichtig ist, dass

für beide I²C-Leitungen jeweils ein Pull-Up-Widerstand zu VDD verbunden werden muss. Beim Raspberry Pi 3B+ ist bereits auf dem Board jeweils ein Pull-Up-Widerstand von 1.8 kΩ verbaut. Der Reset-Pin muss ebenfalls auf VDD verbunden werden, da der Pin low-activ ist. Bei A0, A1 und A2 kann die I²C-Adresse des IC ausgewählt werden. Hierbei steht GND für eine 0 und VDD für eine 1. Die entsprechende Adresse ergibt sich aus Formel 2.1:

$$a = 20_{16} + (A2 << 2) + (A1 << 1) + A0 \tag{2.1}$$

Die minimale Adresse ist somit 0x20 und die maximale 0x27. Entsprechend können maximal 7 MCP23017 in einem I²C-Adressbereich verwendet werden.

Die Funktionsweise des MCP23017 kann in Abbildung 2.8 entnommen werden. In den folgenden Abbildungen wird SDA immer gelb und SCL immer grün dargestellt.

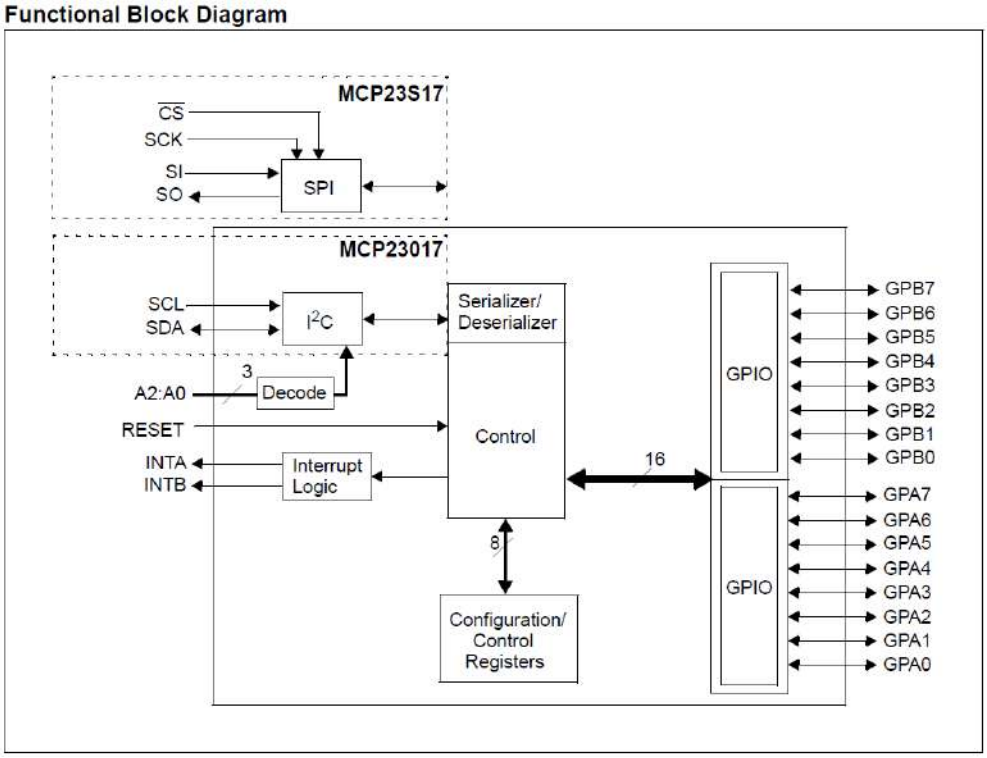


Abbildung 2.8.: Blockschaltbild des MCP23017 [6]

Die Funktion des IC wurde, mit Hilfe der in Abbildung 2.9 gezeigen Schaltung überprüft. Im gesamten Projekt wurde zum Zugreifen auf die ICs die Wiring Pi [12] Bibliothek verwendet.

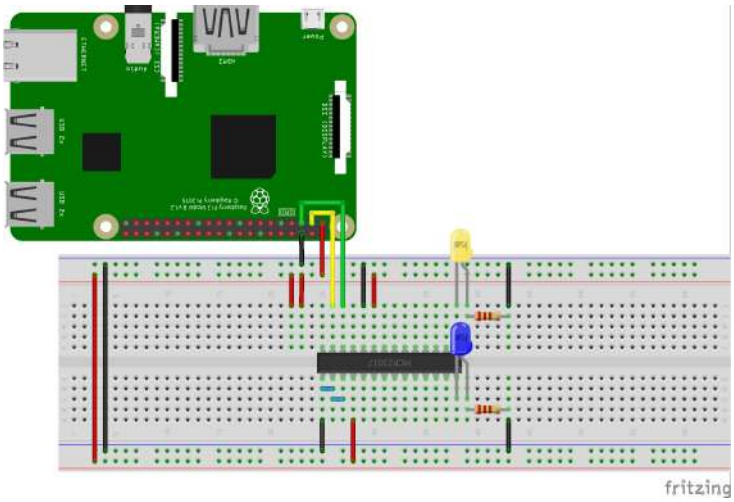


Abbildung 2.9.: Testaufbau des MCP23017 an einem Raspberry Pi 3B+

In dem bisher gezeigten Anwendungsfall des Port-Expanders wurde für VCC immer eine Spannung von 3.3 V verwendet, da der RPi an seinen GPIO-Pins mit einer Spannung von 3.3 V arbeitet. Wie bereits in Kapitel 2.1.2 erwähnt, verwenden die DMG-Cartridges eine Spannung von 5 V, weshalb auch der MCP23017 mit dieser Spannung betrieben werden muss. Damit weiterhin eine I²C-Kommunikation möglich ist, wird zwischen den SDA- und SCL-Leitungen ein Pegelwandler benötigt. In Abbildung 2.10 wird eine einfache bidirektionale Pegelwandlerschaltung gezeigt, die im Folgenden zum Umwandeln der Spannungspegel auf dem I²C-Bus verwendet wurde. Bei dieser Schaltung haben beide Seiten das gleiche GND-Potenzial. Des Weiteren wurden auf der 5 V Seite wieder Pull-Up-Widerstände für die SDA- und SCL-Leitung in Höhe von 1 kΩ eingefügt.

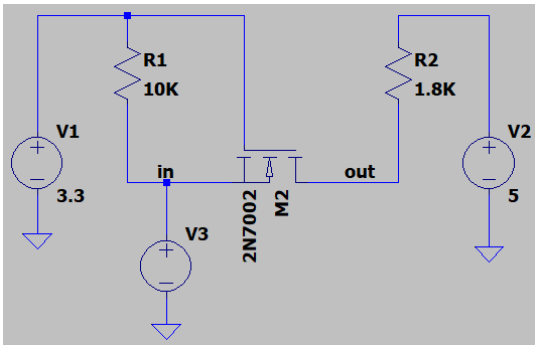


Abbildung 2.10.: Bidirektionale Pegelwandlerschaltung mit einem MOSFET

3. Implementierung

3.1. Hardware

In der in Abbildung 3.1 zu sehenden Schaltung, wurden im Wesentlichen die GPIO-Pins des MCP23017 über einen 470 Ω Widerstand an den Cartridge Adapter verbunden. Der Widerstand verhindert hierbei einen zu hohen Kurzschlussstrom, der die Catridge beschädigen könnte. Des Weiteren sind in dieser Schaltung und in dem Layout (siehe Abbildung 3.2) bereits einige Fehler ausgebessert, die in der PCB noch vorhanden sind (siehe Abbildung 3.3, behoben durch die Drähte).

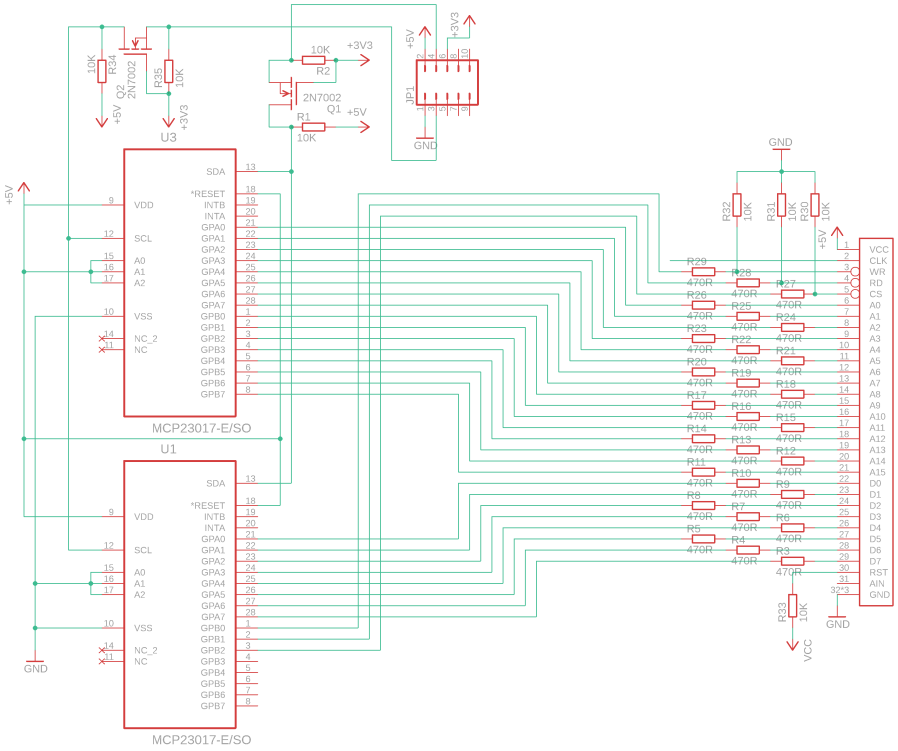


Abbildung 3.1.: Schaltplan des PCB

Da die Platine im folgenden Semester in einen GameBoy Zero eingebaut werden

sollte, war die Form der Platine entscheidend. Die physischen Abmessungen der Platine sind hier so gewählt, dass diese dieselbe Form wie das Mainboard eines GameBoys hat. Hierfür wurde eine Eagle-Vorlage verwendet, die das GameBoy Mainboard nachbildet [13]. Von diesem Layout wurden fast alle Bauteile gelöscht und nur die physischen Abmessungen, Bohrlöcher, Drehrad zur Lautstärkeregelung und der Schalter übernommen. Da diese Vorlage jedoch mit der „CC BY-NC-SA 2.5 CA“ [14] lizensiert wurde, welche eine kommerzielle Nutzung ausschließt, und die verbleibenden Bauteile bei der Weiterführung im nächsten Semester gegen verwendbare Bauteile ausgetauscht werden müssen, wird die Vorlage nicht mehr benötigt. Mit dem Übertragen der Schaltung in eine eigene Vorlage kann somit auch auf die Lizenz verzichtet werden. Das Layout der in Abbildung 3.1 gezeigten Schaltung, kann in Abbildung 3.2 gesehen werden.

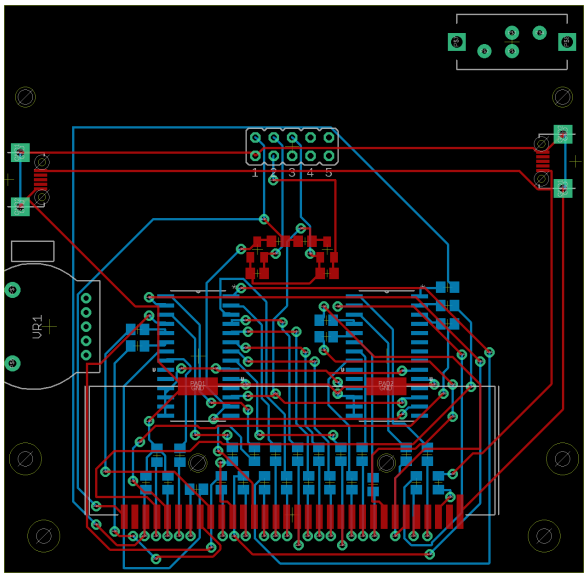
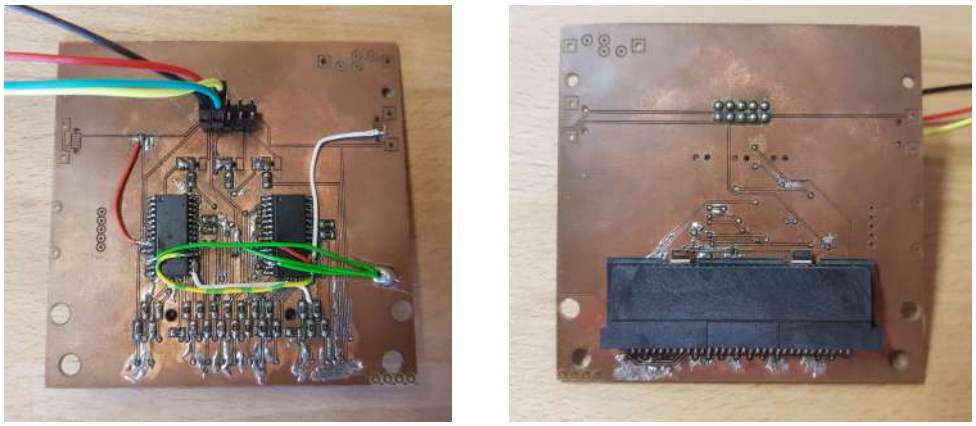


Abbildung 3.2.: Layout des PCB

Die Schaltung wurde mit Hilfe des „PCB-Milling“-Verfahren produziert. Ein großer Nachteil dieses Verfahrens ist, dass es sehr lange dauern würde, das ganze überflüssige Kupfer zu entfernen. Aus diesem Grund wird nur das Kupfer um die Leiterbahnen herum entfernt. Dementsprechend ist es schwieriger, solch eine Platine von Hand zu löten, da kein Lötstopplack vorhanden ist, der z. B. benachbarte Leiterbahnen vor einem Kurzschluss schützen würde. Des Weiteren wurden hier alle Vias von Hand gelötet. Das Löten der SMD-Bauteile wurde vor allem durch das Verwenden von zusätzlichem Flussmittel erleichtert.



(a) Oberseite (b) Unterseite

Abbildung 3.3.: Leiterplatte

Des Weiteren ist anzumerken, dass die Hardware so aufgebaut ist, dass neben DMG-Cartridges mit Softwaremodifikationen auch GameBoy Color Cartridges ausgelesen werden könnten.

3.2. Software

Die Software orientiert sich stark an diesem Arduino Shield [15] [5] [16] [17], welches ich Mitte bis Ende 2019 erstellte. Der wesentliche Unterschied ist, dass bei dem Arduino Shield Shift-Register bzw. die GPIO-Pins des Arduinos und hier die GPIO-Port-Expander verwendet wurden.

Abbildung 3.4 zeigt das Vorgehen zum Auslesen eines Bytes von der Cartridge. Nach dem Anlegen der Adresse an der Cartridge, werden der rd-Pin und der cs-Pin auf eine logische 0 geschaltet. Für eine Leseoperation auf den ROM-Bereich müsste der cs-Pin nicht berücksichtigt werden, da dieser nur beim RAM zum Einsatz kommt. Nach einer Verzögerung von 10 µs können die Datenleitungen ausgelesen werden und die Pins wieder auf eine logische 1 gesetzt werden.

```
Byte GPIO::getByte(Word address) {
    this->setAddress(address);
    this->rd_low();
    this->cs_low();
    usleep(10);

    Byte val = 0x00;
    for (int i = 0; i < 8; i++) {
        if (digitalRead(base_io_expander_data_misc + i)) {
            val |= (0x01 << i);
        }
    }

    this->rd_high();
    this->cs_high();
    usleep(10);

    return val;
}
```

Abbildung 3.4.: Lesen von Bytes von der Cartridge

Abbildung 3.5 zeigt das Vorgehen zum Schreiben eines Bytes in den RAM-Speicher (oder ROM-Bereich zum Wechseln der Banken) der Cartridge. Hierfür müssen zuerst alle Datenpins als Ausgang definiert werden. Nach dem Anlegen der Adresse und des Datenbytes an der Cartridge, wird der wr-Pin auf eine logische 0 geschaltet. Nach einer Verzögerung von 10 µs kann der wr-Pin wieder auf eine logische 1 gesetzt werden.

```
void GPIO::setByte(Word address, Byte data) {
    this->dataOutput();

    setAddress(address);
    for (int i = 0; i < 8; i++) {
        digitalWrite(base_io_expander_data_misc + i, data & (1 << i));
    }

    this->wr_low();
    usleep(10);
    this->wr_high();
    usleep(10);

    this->dataInput();
}
```

Abbildung 3.5.: Schreiben von Bytes zur Cartridge

Zum Auslesen des ROM-Speichers kann der in Abbildung 3.6 gezeigt Code verwendet werden. Vorab wird der, in Kapitel 2.1.2 erklärte, Catridge-Header ausgelesen. Die, mittels Tabelle 2.4 ermittelte, Anzahl der ROM-Banken wird in der Variable „romBanks“ gespeichert. Nun wird in einer Schleife die gewünschte Bank ausgewählt, ausgelesen und in ein Array geschrieben. Dieses Array kann später in eine Datei abgelegt werden. Hierfür hat sich die Dateiendung „.gb“ etabliert.

```
void Cartridge::DumpROM() {
    rom_array = new char[romBanks * 0x4000];
    Word address = 0;

    // Read number of banks and switch banks
    for (Word bank = 1; bank < this->romBanks; bank++) {
        if (this->cartridgeType >= 5) {
            // MBC2 and above
            // Set ROM bank
            this->gpio.setByte(0x2100, bank & 0xFF);
            this->gpio.setByte(0x3000,
              bank > 0x100 ? 0x01 : 0x00);
        } else {
            // MBC1
            // Set ROM Mode
            this->gpio.setByte(0x6000, 0);
            // Set bits 5 & 6 (01100000) of ROM bank
            this->gpio.setByte(0x4000, bank >> 5);
            // Set bits 0 & 4 (00011111) of ROM bank
            this->gpio.setByte(0x2000, bank & 0x1F);
        }
        if (bank > 1) { address = 0x4000; }

        for (; address <= 0x7FFF; address++) {
            rom_array[address + 0x4000 * (bank - 1)] =
              this->gpio.getByte(address);
        }
    }
}
```

Abbildung 3.6.: Auslesen von ROM

Zum Auslesen des RAM-Speichers kann der in Abbildung 3.7 gezeigte Code verwendet werden. Die mittels Tabelle 2.5 ermittelte Anzahl der RAM-Banken wird in der Variable „ramBanks“ gespeichert. Ebenfalls wird in einer Schleife die gewünschte Bank ausgewählt, ausgelesen und in ein Array geschrieben. Dieses Array kann später in eine Datei abgelegt werden. Hierfür hat sich die Dateiendung „.sav“ etabliert.

```
void Cartridge::DumpRAM() {
    this->gpio.getByte(0x0134);

    // Does cartridge have RAM
    if (this->ramEndAddress > 0) {
        ram_array = new char[this->ramBanks * 0x2000];

        if (cartridgeType <= 4) { // MBC1
            // Set RAM Mode
            this->gpio.setByte(0x6000, 1);
        }

        // Initialise MBC
        this->gpio.setByte(0x0000, 0x0A);

        // Switch RAM banks
        for (Byte bank = 0; bank < this->ramBanks; bank++) {
            this->gpio.setByte(0x4000, bank);

            // Read RAM
            for (unsigned int address = 0xA000;
                 address <= this->ramEndAddress;
                 address++) {

                ram_array[address - 0xA000 + bank * 0x2000] =
                  this->gpio.getByte(address);
            }
        }

        // Disable RAM
        this->gpio.setByte(0x0000, 0x00);
    }
}
```

Abbildung 3.7.: Auslesen von RAM

4. Testen

Zuerst, musste überprüft werden, ob die ICs über den I²C-Bus am RPi erkannt werden können. Mit dem Paket „i2c-tools“ [18] kann nach I²C-Geräten auf dem Bus gesucht werden. Entsprechend muss hier ein Gerät bei 0x20 und 0x27 gefunden werden. Um dies zu erreichen, wurden mit einem Oszilloskop die SDA- und SCL-Leitungen auf beiden Seiten der Pegelanpassung vermessen, entsprechende Fehler behoben und schlussendlich eine funktionierende Kommunikation verifiziert. Der Fokus bei dem Vorgehen war, ein gültiges SCL-Signal auf dem Bus zu sehen. Hierfür musste (wie in Abbildung 3.3 gesehen werden kann) die SCL-Leitung über Drähte neu angelötet werden. Hierbei war [19] eine große Hilfe beim Finden der I²C-Fehler. In Abbildung 4.1 kann der Testaufbau zur Verifikation der I2C-Kommunikation gesehen werden. Auf dem Oszilloskop sieht man die 5V Pegel des I²C-Bus nach dem Pegelwandler.

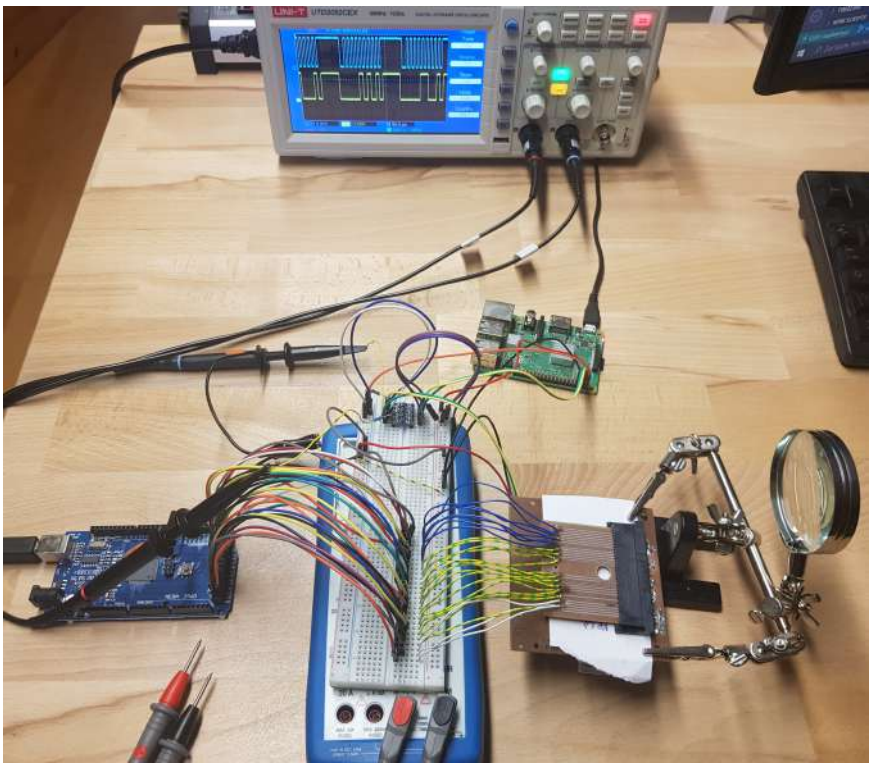
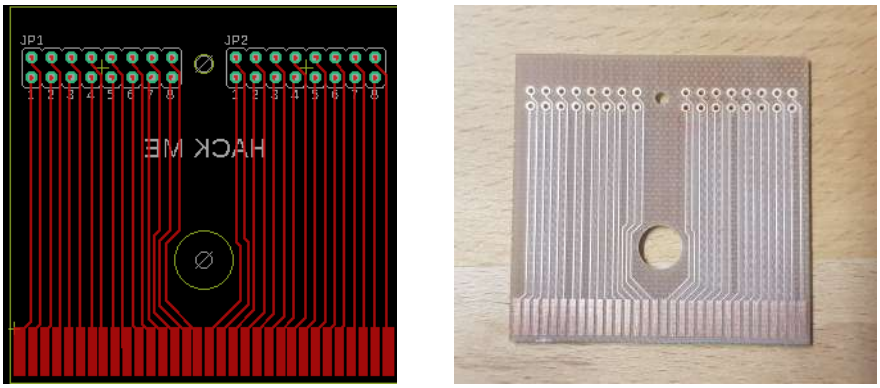


Abbildung 4.1.: Testbench zum Überprüfen der Funktionalität des I²C-Bus und der PCB

Die in Abbildung 4.2 gezeigte Cartridge basiert auf [20]. Hier wurden alle Komponenten entfernt und die Pads mit einem Header verbunden. Mit Hilfe dieser Catridge konnten die am Cartridge-Adapter anliegenden Signale sehr einfach getestet werden. Diese Platine wurde ebenfalls mit Hilfe des „PCB-Milling“-Verfahren realisiert, wobei alles überflüssige Kupfer entfernt wurde, da sonst die Cartride-Pins kurzgeschlossen worden wären. Der Aufbau wird in Abbildung 4.1 gezeigt.



(a) Layout (b) PCB

Abbildung 4.2.: Debug Cartridge

In dem in Abbildung 4.1 gezeigten Testaufbau, können die Lese- und Schreibeoperationen der verschiedenen Pins der GPIO-Expander überprüft werden.
Zur Überprüfung der Schreibbefehle läuft auf dem Arduino Mega Microcontroller ein Logic Analyzer. Dieser liest alle 10 ms den logischen Zustand aller Pins aus und schreibt die Werte im CSV-Format über die serielle Schnittstelle an den Rechner. Auf diesem Recher können im Anschluss die CSV-Werte mit einem Python-Skript visuell dargestellt werden. Der Logic Analyzer kann hier [21] gefunden werden. Auf dem DUT (in Abbildung 4.1) läuft ein Programm, das nacheinander, im Abstand von 20 ms jeden Pin einzeln auf eine logische 1 schaltet (wr, rd, cs werden auf 0 geschaltet, da diese low-activ sind). Um ein funktionierendes Programm zu gewährleisten, muss entsprechend das Ändern aller Pins im Plot erkennbar sein.
Zur Überprüfung der Lesebefehle läuft auf dem Arduino Mega Microcontroller ein Programm, welches auf die Datenpins alle möglichen Kombinationen von 0x00 bis 0xFF schreibt. Diese werden auf dem RPi ausgelesen und die Abfolge der geschriebenen Bytes validiert.
Durch das Testen mit diesen Programmen konnten verschiedene Fehler (z. B. eine schlechte Verbindung beim Löten) erkannt und entsprechend behoben werden.
Im Anhang in Kapitel A kann das Messprotokoll zu den oben genannten Tests gefunden werden. Unter „Input Tester - Raspberry Pi“ (ab Zeile 3) können die ausgelesenen Datenbytes vom Arduino gesehen werden. Hierbei tritt einmal ein Bitfehler auf, was aber auch am Testprogramm liegen kann, da die später ausgelesenen Cartridges einwandfrei funktionieren. Unter „Output Tester - Arduino Logic Analyzer“ (ab Zeile 120) kann die CSV-Datei des Logic Analyzers gesehen werden. Hier sieht man, dass jeder Pin entsprechend beschrieben werden kann.

Nachdem die oben erwähnten Tests bestanden wurden, konnte das Auslesen von GameBoy-Cartridges getestet werden. Der Aufbau wird in Abbildung 4.3 gezeigt. Hierfür wurden die in Kapitel 3.2 erklärten Funktionen verwendet.

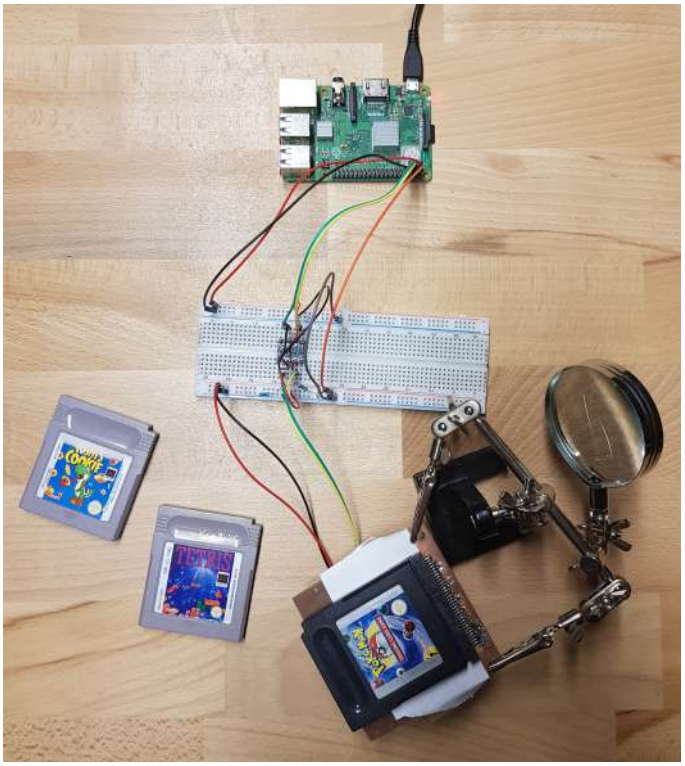
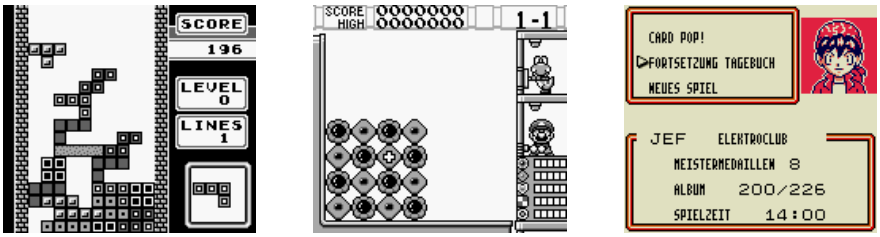


Abbildung 4.3.: Auslesen der Cartridges

5. Ergebnisse

Im Laufe des Projekts, konnte die am Anfang gestellte Zielsetzung erfüllt werden. Es wurde eine Platine erstellt, welche das Auslesen von GameBoy-Cartridges ermöglicht. Diese besitzt den Formfaktor eines GameBoy-Mainboards und kann somit in einem GameBoy-Zero eingebaut werden. Entsprechend befindet sich der Cartridge-Adapter an derselben Stelle wie beim originalen GameBoy. Um die GPIO-Pins des RPi zu erweitern, wurde ein Port-Expander, welcher über I²C mit dem RPi kommuniziert, verwendet.

In Abbildung 5.1 können Screenshots gesehen werden, die bei der Emulation der ausgelesenen Cartridges aufgenommen wurden. Es konnten alle in Abbildung 2.3 gezeigten Cartridges ausgelesen werden. Da diese die verschiedenen MBC kein MBC, MBC1 und MBC5 hatten, kann davon ausgegangen werden, dass auch das Auslesen anderer Cartridges mit denselben Controllern ebenfalls funktionieren würde.



(a) Tetris (b) Yoshi's Cookie (c) Pokémon TCG mit Speicherstand

Abbildung 5.1.: Emulation der ausgelesenen Cartridges

Des Weiteren gibt es einige Verbesserungsvorschläge. Mit der für die Kommunikation mit den MCP23017-ICs verwendete Bibliothek „WiringPi“ [12] kann jeder Pin nur über einen einzelnen Befehl beschrieben werden. Die ICs würden auch die Möglichkeit bieten, einen ganzen Port (8 Pins) jeweils mit einem Byte zu beschreiben. Dies wäre eine signifikante Zeitersparnis und würde das Auslesen deutlich beschleunigen. In der aktuellen Version wird für das Auslesen einer ROM-Bank eine Zeit von 4 min benötigt. Dies führt dazu, dass das Auslesen großer Cartridges mehrere Stunden dauern kann. Im Gegensatz dazu wird mit diesem Arduino Shield [15] nur eine Zeit von 7.7 s benötigt. Des Weiteren könnte der Prozess durch das Verwenden der SPI-Version des MCP23017

durch die wesentlichen höheren Transferraten noch weiter beschleunigt werden.

Darüber hinaus gibt es einige Verbesserungen, die in die in Abbildung 3.1 gezeigte Schaltung bereits eingezeichnet wurden. Beim Testaufbau wurde ein Pegelwandler (siehe Abbildung 4.1) mit Pull-Up-Widerständen auf einem Steckbrett verwendet. Diese wurden nachträglich mit in die Schaltung und das Layout (siehe Abbildung 3.2) eingezeichnet. Außerdem wurde bei der gefrästen Platine die Konfiguration der I²C-Adresse der ICs vergessen und die Unter- und Oberseite der Mainboard-Vorlage vertauscht. Dies wurde ebenfalls ausgebessert.

Literaturverzeichnis

[1] "Raspberry Pi Zero W," https://www.raspberrypi.org/products/raspberry-pi-zero-w/, Zugriff: 17.12.2020.

[2] "ZeroBoy: Geschichte," https://www.zeroboy.eu/geschichte/, Zugriff: 05.12.2020.

[3] F. Weichselgartner, "GameBoy Zero," https://felixweichselgartner.github.io/2019/01/29/GameBoy_Zero.html, Zugriff: 12.12.2020.

[4] ——, "GitHub: GameBoy-Classic-Emulator," https://github.com/FelixWeichselgartner/GameBoy-Classic-Emulator, Zugriff: 07.12.2020.

[5] ——, "GB Cartridge Reader Writer," https://felixweichselgartner.github.io/2019/09/01/GB_Cartridge_Reader_Writer.html, September 2019, Zugriff: 07.12.2020.

[6] *MCP23017/MCP23S17: 16-Bit I/O Expander with Serial Interface*, Microchip Technology Inc., 06 2016, https://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf.

[7] B. Byers, "Brendan's Website: A Look at the Gameboy DMG," https://b13rg.github.io/Gameboy_DMG/, Zugriff: 06.12.2020.

[8] "Gameboy Development Wiki," https://gbdev.gg8.se/wiki/articles/Main_Page, Zugriff: 06.12.2020.

[9] B. Byers, "Brendan's Website: Gameboy DMG ROM and RAM Bank Switching," https://b13rg.github.io/Gameboy-Bank-Switching/, Zugriff: 12.12.2020.

[10] "Gameboy Development Wiki: The Cartridge Header," https://gbdev.gg8.se/wiki/articles/The_Cartridge_Header, Zugriff: 07.12.2020.

[11] B. Byers, "Brendan's Website: Exploring the Gameboy Memory Bank Controller," https://b13rg.github.io/Gameboy-MBC-Analysis/, Zugriff: 07.12.2020.

[12] "Wiring Pi - GPIO Interface library for the Raspberry Pi: I2C: MCP23008 & MCP23017," http://wiringpi.com/extensions/i2c-mcp23008-mcp23017/, Zugriff: 05.12.2020.

[13] "DMG-Rolf GameBoy schematics," https://gbdev.gg8.se/files/schematics/DMG-Rolf/, Zugriff: 12.12.2020.

[14] "Attribution-NonCommercial-ShareAlike 2.5 Canada (CC BY-NC-SA 2.5 CA)," https://creativecommons.org/licenses/by-nc-sa/2.5/ca/, Zugriff: 12.12.2020.

[15] F. Weichselgartner, "GitHub: GB-Cartridge-Reader-Writer," https://github.com/FelixWeichselgartner/GB-Cartridge-Reader-Writer, Zugriff: 07.12.2020.

[16] ——, "GB Cartridge Reader Writer continued," https://felixweichselgartner.github.io/2020/11/28/GB_Cartridge_Reader_Writer_continued.html, November 2020, Zugriff: 07.12.2020.

[17] "GBCartRead – Gameboy Cart Reader," https://github.com/insidegadgets/GBCartRead, Zugriff: 25.12.2020.

[18] "Paket: i2c-tools (3.1.2-3) - Heterogener I2C-Werkzeugsatz für Linux," https://packages.debian.org/de/stretch/i2c-tools, Zugriff: 24.12.2020.

[19] "The Problems of I2C - common problems and errors with using I2C," https://www.microforum.cc/blogs/entry/42-the-problems-of-i2c-common-problems-and-errors-with-using-i2c, Zugriff: 25.12.2020.

[20] "Xyl2k - Homebrew Gameboy Color Cartridge," https://github.com/Xyl2k/Gameboy-Color-Cartridge, Zugriff: 19.12.2020.

[21] F. Weichselgartner, "GitHub: Arduino_Logic_Analyzer," https://github.com/FelixWeichselgartner/Arduino_Logic_Analyzer, Zugriff: 07.12.2020.

Abbildungsverzeichnis

2.1. DMG Frontansicht und Mainboard 2
2.2. Blockschaltbild des DMG 3
2.3. Frontansichten der verwendeten Cartridges 7
2.4. Rückansichten der verwendeten Cartridges 7
2.5. Cartridge Pinout 8
2.6. Cartridge Adapter 9
2.7. Pinout des MCP23017 9
2.8. Blockschaltbild des MCP23017 10
2.9. Testaufbau des MCP23017 an einem Raspberry Pi 3B+ 11
2.10. Bidirektionale Pegelwandlerschaltung mit einem MOSFET 11

3.1. Schaltplan des PCB 12
3.2. Layout des PCB 13
3.3. Leiterplatte 14
3.4. Lesen von Bytes von der Cartridge 15
3.5. Schreiben von Bytes zur Cartridge 16
3.6. Auslesen von ROM 17
3.7. Auslesen von RAM 18

4.1. Testbench zum Überprüfen der Funktionalität des I²C-Bus und der PCB 20
4.2. Debug Cartridge 21
4.3. Auslesen der Cartridges 22

5.1. Emulation der ausgelesenen Cartridge 23

Tabellenverzeichnis

2.1. Memory Map des DMG 3
2.2. Memory Map des Cartridge Headers 4
2.3. Cartridge Header: Cartridge Typ 5
2.4. Cartridge Header: ROM Größe 6
2.5. Cartridge Header: RAM Größe 6

Abkürzungsverzeichnis

DMG Originaler GameBoy, Klassischer GameBoy

MBC Memory Bank Controller

ROM Read Only Memory

RAM Random Access Memory

IC Integrated Circuit

I²C Inter-Integrated Circuit

SPI Serial Peripheral Interface

RPi Raspberry Pi

DUT Device Under Test

CSV Comma Separated Values

A. Anhang - Testprotokoll

```
# Input Tester

## Raspberry Pi
WiringPi Setup was successfull.
You started the input pin tester!
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
e0 e1 e2 e3 e4 e5 e6 e7 e8 could not find e9 -> continueing with f0
f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff

finished with errors
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff

finished without errors
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff

finished without errors
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
```

```
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff

finished without errors
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff

finished without errors


## Arduino
0 -> 0: 0   1: 0   2: 0   3: 0   4: 0   5: 0   6: 0   7: 0
1 -> 0: 1   1: 0   2: 0   3: 0   4: 0   5: 0   6: 0   7: 0
2 -> 0: 0   1: 2   2: 0   3: 0   4: 0   5: 0   6: 0   7: 0
3 -> 0: 1   1: 2   2: 0   3: 0   4: 0   5: 0   6: 0   7: 0
4 -> 0: 0   1: 0   2: 4   3: 0   4: 0   5: 0   6: 0   7: 0
5 -> 0: 1   1: 0   2: 4   3: 0   4: 0   5: 0   6: 0   7: 0
6 -> 0: 0   1: 2   2: 4   3: 0   4: 0   5: 0   6: 0   7: 0
7 -> 0: 1   1: 2   2: 4   3: 0   4: 0   5: 0   6: 0   7: 0
8 -> 0: 0   1: 0   2: 0   3: 8   4: 0   5: 0   6: 0   7: 0
9 -> 0: 1   1: 0   2: 0   3: 8   4: 0   5: 0   6: 0   7: 0
10 -> 0: 0   1: 2   2: 0   3: 8   4: 0   5: 0   6: 0   7: 0
...
254 -> 0: 0   1: 2   2: 4   3: 8   4: 16   5: 32   6: 64   7: 128
255 -> 0: 1   1: 2   2: 4   3: 8   4: 16   5: 32   6: 64   7: 128

# Output Tester
## Raspberry Pi
pi@raspberrypi:~/EmbeddedBoy/debug/cartridge_pin_tester/output_tester $
./pintester.out
WiringPi Setup was successfull.
You started the pin tester!

# Arduino Logic Analyzer
D7 D6 D5 D4 D3 D2 D1 D0 15 14 13 12 11 10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 CS RD WR
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 0
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 0
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 1; 1; 1
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 1; 1; 1
```

```
146  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 1; 1; 1
147  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 1; 1; 1
148  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
149  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
150  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
151  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
152  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
153  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
154  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
155  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
156  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
157  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
158  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
159  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
160  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
161  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
162  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
163  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
164  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
165  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
166  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
167  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
168  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
169  0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
170  0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
171  0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
172  0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
173  0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
174  0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
175  0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
176  0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
177  0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
178  0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
179  0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
180  0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
181  0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
182  0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
183  0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
184  0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
185  0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
186  0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
187  0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
188  1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
189  1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
190  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 0
191  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 0
192  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 1
193  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 1
194  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1
195  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1
196  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
197  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
198  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
199  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
200  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
201  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
202  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
203  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
204  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
205  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
206  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
207  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
208  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
209  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
210  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
211  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
212  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
213  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
214  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
215  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
216  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
217  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
218  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
```

```
219  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
220  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
221  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
222  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
223  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
224  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
225  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
226  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
227  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
228  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
229  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
230  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
231  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
232  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
233  0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
234  0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
235  0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
236  0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
237  0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
238  0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
239  0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
240  0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
241  0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
242  0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
243  0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
244  0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
245  0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
246  0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
247  0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
248  0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
249  0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
250  0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
251  0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
252  0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
253  1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
254  1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1
255
256
```