Technische Hochschule Rosenheim

Fakultät für Informatik

Bachelorarbeit

Bachelorprüfung WS 2020/2021

Eric Gattinger, geb. 01.07.98

Matrikelnr. 866950

Entwicklung eines Embedded Displays mit einer high-density LED-Matrix

Abgabetermin: 22.03.2021

Erstprüfer: Prof. Florian Künzner

Zweitprüfer: Prof. Dr. Wolfgang Mühlbauer

Erklärung

Ich versichere, dass ich diese Arbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Frasdorf, 21.03.2021 E. Gaffinger

Ort, Datum Unterschrift

Abstract

Die Bachelorarbeit handelt von der Ansteuerung einer high-density LED-Matrix per Mikrocontroller. Diese fand bei der HEMI GmbH in Raubling statt.

Die Firma hat sich mehrere LED-Matrizen (je 20 × 15 cm, 160 × 120 = 19200 RGB-LEDs) zugelegt. Da es kein gutes Datenblatt zur LED-Matrix gibt, war die Aufgabe, die LED-Matrix mit Reverse Engineering zu erforschen und einen Treiber dafür zu entwickeln. Damit hat die Firma bereits vor der Bachelorarbeit begonnen. In dieser Bachelorarbeit wurde die Entwicklung fortgesetzt.

Um eine ansehnliche Konstruktion zu bilden, sollen vier LED-Matrizen zusammen ein größeres Display (40 × 30 cm, 320 × 240 RGB-LEDs) bilden und auf einer eigens entwickelten Leiterplatte befestigt werden.

Ferner war die Vision, Datenpakete von Drittsystemen über eine serielle Schnittstelle oder Ethernet zu empfangen. Diese beinhalten beispielsweise aktuelle Statusinformationen der Maschinen, die die Firma besitzt. Auf der LED-Matrix sollen diese Informationen dargestellt werden. Auch sollen bis zu zwei SNES-Spielecontroller angeschlossen werden können, um beispielsweise ein Menü, das auf dem Display angezeigt wird, zu bedienen.

Im Rahmen der Bachelorarbeit wurde das Ziel erreicht, ein statisches Bild über die serielle Schnittstelle zu übertragen und am Display anzuzeigen. Die Übertragung mittels USB oder Ethernet wäre ebenso möglich, wenn die entsprechenden Schnittstellen dazu programmiert werden.

Eine Demonstration des Ergebnisses dieser Arbeit gibt es Online als Video unter folgendem Link: youtu.be/PVObh-cLf8k

Schlagworte: Embedded, Display, LED-Matrix, Mikrocontroller

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

Analog-Digital-Wandler, auch „ADC“ („analog-to-digital converter“) genannt, ist ein Modul, das die Höhe der Eingangsspannung in einen digitalen Wert umwandelt [1 S. 900]

Arbeitsspeicher, auch „RAM“ („random access memory“) genannt, ist der Ort, an dem die Variablen eines Programms hinterlegt sind und kann während der Programmausführung ständig überschrieben werden [1 S. 991]

Array: Eine Menge von Variablen

Binärsystem: Darstellung von Zahlen zur Basis 2 [1 S. 705]

C: Eine Programmiersprache

Clock: Ein taktgebendes digitales Signal, indem es periodisch invertiert (HIGH auf LOW oder LOW auf HIGH) wird, meist, aber nicht unbedingt, in einem gleichbleibenden Takt

Crimpen: Eine Technik, mit der das Ende eines Kabels mit einem Anschluss versehen werden kann [18]

Debugging: Überwachung und Manipulation des Programms während des Programmablaufs

Digital-Analog-Wandler, auch „DAC“ („digital-to-analog converter“) genannt, ist ein Modul, das einen digitalen Wert in eine analoge Ausgangsspannung umwandelt [1 S. 881]

Flanke: Wenn ein digitales Signal von LOW auf HIGH beziehungsweise von HIGH auf LOW übergeht, nennt man den Übergang eine steigende beziehungsweise fallende Flanke [13]

Flash-Speicher, oftmals fälschlicherweise „ROM“ („read-only memory“) bezeichnet, da auf EEPROM-Technologie basierend, ist der Ort, an dem das Programm hinterlegt ist und kann während der Programmausführung nicht so einfach überschrieben werden [1 S. 991]

Galvanische Trennung: Bezeichnung für die Trennung zweier Stromkreise durch das Fehlen einer elektrisch leitfähigen Verbindung [23]

Hexadezimalsystem: Darstellung von Zahlen zur Basis 16 [1 S. 705]

HIGH: Ein Logikpegel, der digital einer binären 1 entspricht [1 S. 17]

Interferenz: Überlagerung von Signalen [8]

Interrupt: Eine durch bestimmte Ereignisse ausgelöste Unterbrechungsaufforderung, um ein Unterprogramm, die sogenannte „Interrupt Service Routine“ („ISR“), aufzurufen [27]

LOW: Ein Logikpegel, der digital einer binären 0 und bei digitalen Schaltungen idealerweise einer Spannung von 0 V entspricht [1 S. 17]

Matrix: Anordnung in Form einer Tabelle

Modul: Elektronisches Gerät, Bauteil oder Teil eines Bauteils mit einer bestimmten Funktion

Multitasking: Mehrere Aufgaben im selben Zeitraum zu erledigen [40]

Oszilloskop: Ein Gerät, das dazu verwendet wird, Spannungen im zeitlichen Verlauf zu messen

Pin: Nach außen offenstehende Signalleitung

Pixel: Punkt auf einem Display mit einer Lichtquelle

Pixel-Art: Wörtlich übersetzt „Pixel Kunst“, ist ein Zeichenstil, bei dem in einem entsprechenden Computerprogramm ein Pixel nach dem anderen gemalt wird [31]

Protokoll: Standards und Normen für Datenübertragungen zwischen verschiedenen Geräten [21]

(zentrale) Prozessoreinheit, auch bloß „Prozessor“ oder „CPU“ („central processing unit“) genannt, ist der Hauptbestandteil von Rechnern und in der Lage komplexe Berechnungen durchzuführen

(Schwing)Quarz: Ein Bauteil das durch die eigene Schwingung einen Takt vorgibt

Register: Bezeichnung für bestimmte Speicherstellen in Mikrocontroller oder ICs, welche bestimmte Funktionen auslösen oder konfigurieren

Reverse Engineering: Ist übersetzt das Zerlegen oder Nachbauen eines Wettbewerbsprodukts [41]

Schieberegister: Speicher für mehrstellige binäre Werte, welcher taktgesteuert einen neuen Binärwert aufnimmt und die bereits vorhandenen Werte um eine Binärstelle verschiebt [9]

Timer: Ein Modul im Mikrocontroller, das nach einer definierten Zeit einen bestimmten Vorgang auslöst, indem es einen Zähler anhand einer Clock hochzählt

Transceiver: Ein Modul das empfangen, als auch versenden kann

Variable: Ein jederzeit veränderbarer Wert, der im Arbeitsspeicher hinterlegt wird

Abkürzungen

ASCII: „American Standard Code for Information Interchange“ ist eine binäre Zeichenkodierung für Ziffern, Buchstaben und sonstige Zeichen [32]

BMP: „bitmap image“ ist ein verlustfreies und geräteunabhängiges Format für digitale Bilddateien [36]

DCLK: „Data Clock“ ist ein für das Verschieben der Daten im Schieberegister zuständiges Clocksignal

GCLK: „Gray Scale Clock“ ist ein für das Beleuchten der LEDs zuständiges Clocksignal

GIF: „grafical interchange format“ ist ein verlustbehaftetes und komprimierendes Format für digitale Bilderdateien von der Firma Unisys [37]

GPIO: „general purpose input/output“ ist ein digitaler Signalein- und/oder -ausgang

Hz: „Hertz“ ist eine internationale Einheit für die Frequenz [22]

IC: „integrated circuit“, deutsch „Integrierter Schaltkreis“, ist ein allgemeiner Begriff für Schaltungsbauteile [11]

IEEE: „institute of electronics engineers“, meist „i triple e“ ausgesprochen, ist ein weltweiter Verband von Ingenieuren, bekannt für ihre Publikationen, Konferenzen und Technologiestandards [14]

ISR: „Interrupt Service Routine“ ist ein durch ein Interrupt ausgelöstes Unterprogramm

JPG/JPEG: Ein verlustbehaftetes und komprimierendes Format für Bilddateien, standardisiert durch „Joint Photographic Experts Group“, woher auch die Abkürzung stammt [38]

JTAG: „joint test action group“, auch bekannt als „boundary scan“ oder „IEEE Standard 1149.1“, ist eine Schnittstelle mit der unter anderem Programme auf dem Mikrocontroller hochgeladen und debuggt werden [1 S. 1036]

g: „Gramm“ ist eine SI-Basiseinheit für Masse [3]

K: „Kelvin“ ist eine SI-Basiseinheit für thermodynamische Temperatur [3]

LE: „Latch Enable“, wird für die Übertragung der Daten vom Schieberegister in das Ausgaberegister, als auch für die Konfiguration der LED-Matrix benötigt

LED: „light-emitting diode“, deutsch „Leuchtdiode“

m: „Meter“ ist die SI-Basiseinheit für Länge [3]

M3: Bezeichnung für den metrischen Gewindedurchmesser von 3,0 mm [24]

PHY: „physical layer“ der Ethernet-Schnittstelle [1 S. 1032]

PNG: „portable network graphic“ ist ein verlustfreies Format für Bilddateien ohne Urheberrechtsbeschränkungen [39]

PWM: „Pulsweitenmodulation“/„pulse width modulation“ ist eine Technik, um mit einer digitalen Ausgabe einen scheinbar analogen Wert auszugeben [10]

RGB: Rot, grün und blau

RJ45: Speziell für die Datenübertragung in Netzwerken entwickelter Steckverbinder [16]

RS485: Eine Art der seriellen Datenübertragung mittels differenzieller Leitungen [17]

SNES: „Super Nintendo Entertainment System“ ist eine Spielekonsole [19]

UART: „universal asynchronous receiver transmitter“ ist eine serielle Kommunikationsschnittstelle [20]

USB-2.0-Typ-B: „Universal Serial Bus“ Version 2.0 des Peripherie-Typs (Gegenstück zum Typ-A, dem Host-Typ) ist ein Standard zur Verbindung eines Computers mit externen Geräten [15]

V: Volt ist die abgeleitete SI-Einheit für elektrische Spannung [3]

1 Einleitung

1.1 Einführung Mikrocontroller

Da in dieser Bachelorarbeit ein Mikrocontroller verwendet wurde, wird nun zuallererst erklärt, was ein Mikrocontroller ist und wozu dieser benötigt wird.

Wo man auch hinsieht, sei es das Auto, die digitale Armbanduhr oder allein die elektrische Zahnbürste, findet man heutzutage Mikrocontroller in allerlei Gerätschaften eingebettet. Es handelt sich hierbei um Kleinstcomputer, welche meistens dazu verwendet werden, analoge und digitale Signale zu verarbeiten, sowie wiederzugeben. Damit das möglich ist, bedarf es nicht nur einer zentralen Prozessoreinheit, also einem Rechner, sondern auch vieler verschiedener Peripheriemodule, welche in einem Mikrocontroller eingebaut werden. Diese wären oftmals ein Analog-Digital- sowie Digital-Analog-Wandler, verschiedene Kommunikationsschnittstellen, mehrere Timer und vieles mehr. Im sogenannten Flashspeicher sind die Programmdaten fest abgespeichert. Das Programm wird gestartet, sobald der Mikrocontroller mit genügend Strom versorgt wird. Im Arbeitsspeicher werden die Variablen, also Daten, welche das Programm ständig verändern kann, hinterlegt. Nach einer Stromunterbrechung sind die Daten im Arbeitsspeicher zwar verloren, aber sobald die Stromversorgung wiederhergestellt wird, wird das Programm problemlos neugestartet. [1 S. 1053]

Wie groß die Speicher und welche Module sich in einem Mikrocontroller befinden, hängt vom Modell des Mikrocontrollers ab.

1.2 Aufgabenstellung und Zielsetzung

Die Firma HEMI GmbH, in welcher die Bachelorarbeit abgehalten wurde, hat sich aus eigenem Interesse mehrere high-density LED-Matrizen zugelegt. Der Grund dafür war, sich mit dieser Art von Display vertraut zu machen und Erfahrungen zu sammeln. Durch die starke Helligkeit dieser Geräte, entstand die Idee, die LED-Matrix als Blickfänger an Messeständen der Firma einzusetzen. Dabei soll das Display auch etwas sinnvolles anzeigen, wie beispielsweise Statusinformationen von Drittsystemen, die über mehrere ausgewählte Schnittstellen empfangen werden sollen.

Speziell für diese Anwendung soll eine Leiterplatte hergestellt werden, auf der sich der ausgewählte Mikrocontroller, die Buchsen für die Kommunikationsschnittstellen

und der Rest der Schaltung befinden sollen. Ebenfalls sollen vier LED-Matrizen auf der Leiterplatte befestigt werden, um flächenmäßig ein vierfach so großes Display zu erschaffen. Der fertige Zustand soll eine stabile und ansehnliche Konstruktion ergeben.

2 Funktionsweise der LED-Matrix

Eine LED-Matrix ist eine Art von Display bei der jedes Pixel aus jeweils einer LED beziehungsweise, bei einer RGB-LED-Matrix, aus jeweils einer roten, grünen und blauen LED besteht. Da jedes Pixel eine eigene Lichtquelle besitzt, weisen LED-Matrizen eine starke Helligkeit auf, weswegen Sie auch in Sportstadien oder als Leuchtreklamen für einen weit entfernten Zuschauer ein gut sichtbares Bild liefern.

2.1 Eckdaten der LED-Matrix

In Tabelle 1 werden die technischen Daten der verwendeten LED-Matrix aufgelistet und in Abbildung 1 eigene Abmessungen der LED-Matrix gezeigt.

Ursprungsort	China
Hersteller	YCX (Yao Caixing)
Pixelmaße	□ 1,25 mm
Auflösung	160 × 120 Pixel
Reihen	4
Zeilen pro Reihe	30
Farbe	RGB
Farbtiefe	16 Bit
Farbtemperatur	5000-9300 K einstellbar
Displaymaße	200 × 150 mm
Gewicht	600 g
Zertifizierungen	CE, RoHS, FCC
Kommunikationsschnittstelle	26 Pins (2 × 13), □ 0,65 mm, 2,54 mm Raster
LED-Chip	SMD1010

Tabelle 1: Teil der technischen Daten der verwendeten LED-Matrix [2]

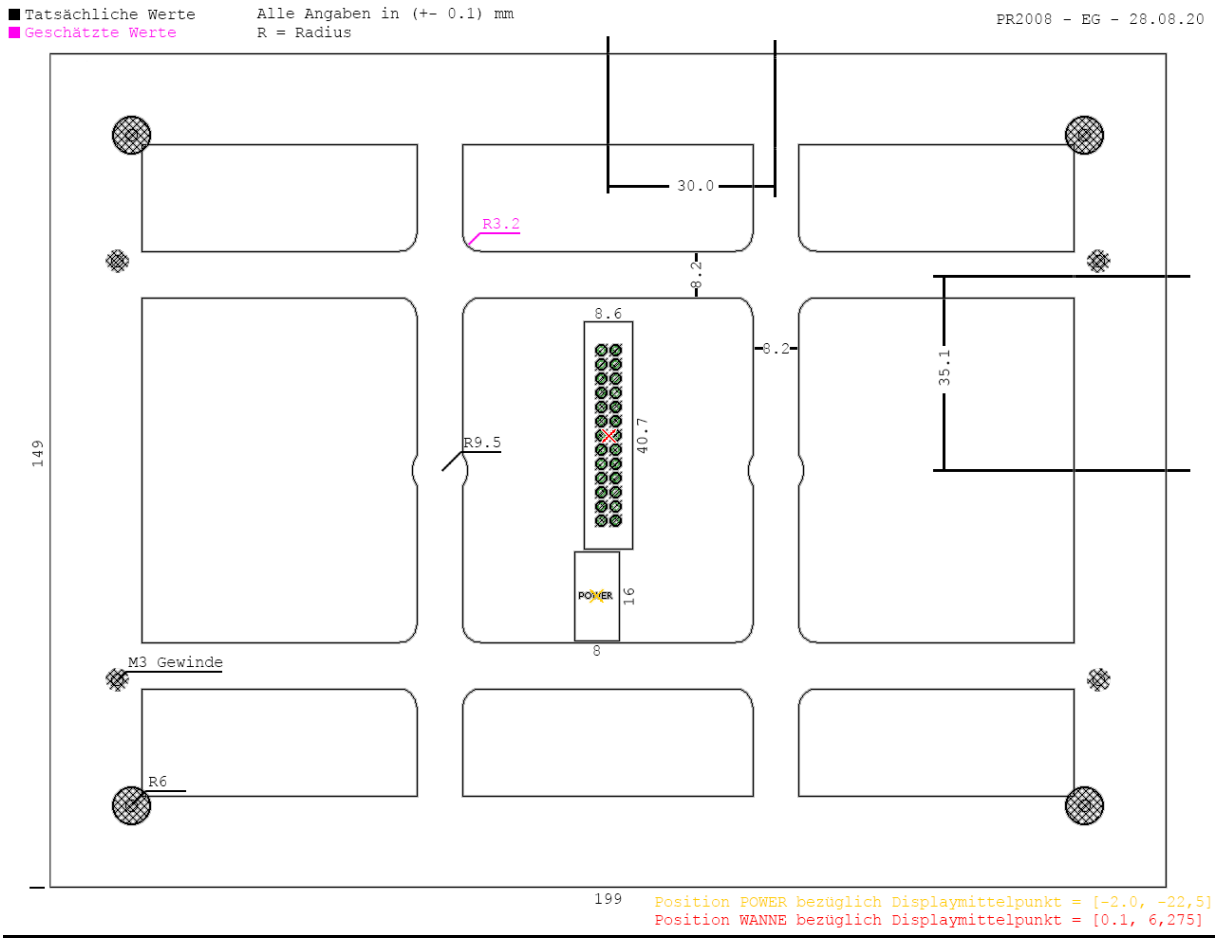

Abbildung 1: Eigene Abmessungen der verwendeten LED-Matrix.

Die offiziellen Datenblätter zur verwendeten LED-Matrix und dessen Bauteile bieten keine brauchbaren Informationen zur eigenen Ansteuerung. Durch Internetrecherche konnte trotzdem ein generelles Verständnis zur allgemeinen Funktionsweise von LED-Matrizen erlangt werden. Damit sei jedoch hingewiesen, dass in diesem Kapitel zum Teil eigene Annahmen getroffen wurden.

2.2 Kommunikationsschnittstelle

Viele verschiedene Varianten von LED-Matrizen verwenden in etwa denselben Aufbau eines mehrpoligen Steckers für die Signalübertragung, wie den in Abbildung 2 Dargestellten.

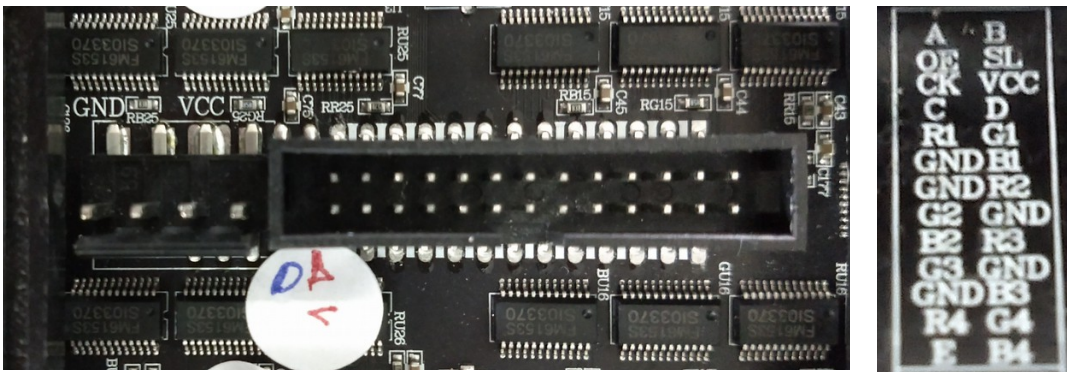

Abbildung 2: Links: 24-poliger Stecker für die Signalübertragung. Links daneben ein zusätzlicher Stecker für die Stromversorgung. Rechts: Pinbelegung. Relativ zum Stecker um 90° gegen den Uhrzeigersinn gedreht.

Kleinere beziehungsweise größere LED-Matrizen können weniger beziehungsweise mehr Pins besitzen. Das Prinzip bleibt jedoch das Gleiche. In Tabelle 2 werden die einzelnen Bedeutungen der Pins beschrieben.

Pin	Beschreibung
OE (vermutlich für „Output Enable“, auch **GCLK** für „Gray Scale Clock“ genannt)	Zuständig für das Beleuchten der LEDs.
CK (vermutlich für „Clock“, auch bekannt als **DCLK** für „Data Clock“)	Zuständig für das Verschieben der Daten im Schieberegister.
SL (vermutlich „Select Latch“, oft auch **LE** für „Latch Enable“ oder **LAT** genannt)	Wird für die Übertragung der Daten vom Schieberegister in das Ausgaberegister, als auch für die Konfiguration der LED-Matrix benötigt.
A, B, C, D und **E**	Bestimmt die Scan Line, beziehungsweise welche Zeile leuchten soll.
R#, G# und **B#**	Farbdaten für Rot, Grün und Blau. Das # steht für die Reihe, für welche die Daten bestimmt sind.

Tabelle 2: Bedeutungen der Signale der Pins am Stecker der LED-Matrix [4, 5 S. 1, 6 S. 3]

2.3 Funktionsweise der Scan Line

Der Unterschied zu anderen Bildschirmen, wie beispielsweise den LCD-Displays handelsüblicher PC-Monitoren, ist nicht nur der, dass die einzelnen Pixel einer LED-Matrix ihre eigene Lichtquelle besitzen und daher eine stärkere Helligkeit erzeugen können, sondern auch der, dass zu jedem Zeitpunkt immer nur eine Zeile in

jeder Reihe leuchtet [7]. Diese wird gebräuchlich als „Scan Line“ bezeichnet. Der Grund, dass nicht alle LEDs auf einmal leuchten können, also nicht parallel angesteuert werden können, ist der, dass dafür schlichtweg zu viele Pins benötigt werden würden. In diesem Fall wären das 160 Pins für die horizontale und 120 für die vertikale Ansteuerung pro Farbe. Bei drei Farben (rot, grün und blau) ergäbe das schließlich 840 Pins zum Ansteuern an der LED-Matrix. Stattdessen wird die Scan Line mithilfe der Pins **A, B, C, D** und **E** ausgewählt. Zusammen bilden diese Pins eine Zahl zwischen 0 und 31, welche die Auswahl der Scan Line bestimmt. Dabei ist **A** das niederwertige Bit mit einem Wert von 2^0 und **E** das höchstwertige Bit mit einem Wert von 2^4. Wird **A**, **C**, und **E** HIGH und **B** und **D** LOW gesetzt, ergibt sich ein Wert von

$$1(2^0)+0(2^1)+1(2^2)+0(2^3)+1(2^4) = 2^0+2^2+2^4 = 1+4+16 = 21 ,$$

also ist, da der Wert mit 0 beginnt, momentan Zeile 22 zum Leuchten ausgewählt.

Um ein flüssiges Bild auf dem Display zu bekommen, sollten alle Zeilen jeweils für dieselbe kurze Zeitspanne leuchten. Am einfachsten wird der Wert, der aus **A, B, C, D** und **E** entsteht, periodisch um 1 erhöht, um die Scan Line Zeile für Zeile umzuschalten, bis die letzte Zeile erreicht wurde. Danach wird der Wert auf 0 zurückgesetzt, um die erste Zeile auszuwählen und den Vorgang zu wiederholen. In Abbildung 3 wird bildlich mit einer kleineren Matrix als Beispiel erläutert, wie das Umschalten der Scan Line schrittweise aussieht.

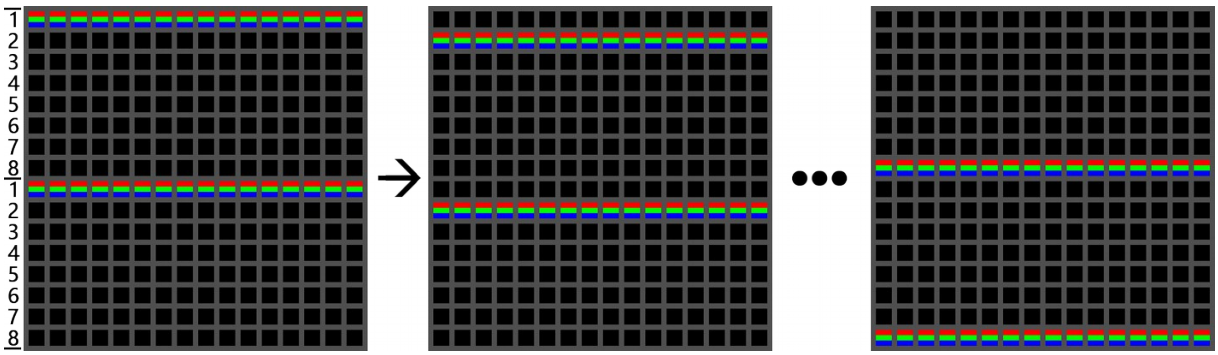

Abbildung 3: Skizze einer 16 × 16 RGB-Matrix mit zwei Reihen. Im ersten Durchgang (links) wurde Zeile 1 (Wert = 0) ausgewählt. **A, B** und **C** sind also LOW. Im nächsten Durchgang (mittig) ist die nächste Zeile ausgewählt, also Zeile 2 (Wert = 1). **A** muss also HIGH, **B** und **C** müssen LOW sein. Im letzten Durchgang (rechts) ist die letzte Zeile, also Zeile 8 (Wert = 7) ausgewählt. **A, B** und **C** sind also HIGH.

Mit dem Begriff „Reihe“ ist jeweils ein Abschnitt auf dem Display gemeint, den eine Scan Line durchläuft. In der verwendeten LED-Matrix gibt es vier Reihen mit jeweils 30 Zeilen. Von daher werden hier 5 Bits benötigt, mit denen sogar bis zu 32 Zeilen

einzeln eingeschaltet werden können. Bei kleineren LED-Matrizen, kann die Anzahl der Zeilen pro Reihe niedriger sein, weswegen auch weniger Pins zur Auswahl der Scan Line vorhanden wären. Auch die Anzahl der Reihen kann niedriger sein, wodurch weniger Pins für die Farbdaten vorhanden wären. Bei größeren Matrizen kann bei Beidem das Gegenteil gelten.

2.4 Anzeige durch die GCLK

Mit der Annahme, dass im Anzeigepuffer der LED-Matrix bereits Daten zum Anzeigen vorhanden sind, wird nun das **GCLK**-Signal benötigt, damit die Farben korrekt angezeigt werden. Am **GCLK**-Pin wird der LED-Matrix ein Takt vorgegeben, sodass die PWM-Treiber der LED-Matrix die Farbwerte korrekt widerspiegeln. Nach einer bestimmten Anzahl von **GCLK**-Flanken soll zur nächsten Scan Line gewechselt werden und der **GCLK**-Takt für eine bestimmte Zeitspanne ausgesetzt werden. Damit die LED-Matrix ein möglichst stabiles Bild liefert, soll die Scan Line immer so schnell wie möglich umgeschaltet werden, sodass der Zuschauer den Anschein bekommt, als würden alle LEDs gleichzeitig leuchten. Auch wenn der Zuschauer in Bewegung ist, sollte dieser Anschein so gut wie möglich bewahrt werden. Je schneller die Scan Line umgeschaltet wird, desto besser wird dieser Schein gewahrt.

Bis jetzt hat man sich aber noch nicht um die Anzeigedaten gekümmert, also sollte im Moment auch noch nichts angezeigt werden. Dies ist allerdings meistens gar nicht der Fall, da in den Anzeigepuffer ungewollt irgendwelche Daten gelangen können. Der Grund ist zum Beispiel der, dass die LED-Matrix sehr empfindlich gegenüber Interferenzen ist, die beispielsweise vom eigenen Körper ausgelöst werden, indem die Pins während des Programmablaufs berührt werden. Der angezeigte „Datenmüll" sollte aber ein statisches Bild ergeben, vorausgesetzt, das Timing der **GCLK,** sowie das Umschalten der Scan Line wurde korrekt umgesetzt.

2.5 Schieberegister und PWM-Treiber

Bei jedem Zyklus der **DCLK** wird ein weiteres Datenbit an den Farbpins ausgelesen und in sein entsprechendes Schieberegister geladen [9]: Dabei verschieben sich die Daten im Schieberegister um ein Bit weiter.

Es gibt jeweils für jede Farbe in jeder Reihe ein Schieberegister. Das heißt, dass bei dieser LED-Matrix die Farbdaten zum selben Zeitpunkt auf vier Reihen übertragen werden. Unterschieden wird also zwischen den Farbpins **R#, G#** und **B#** der Reihen **1** bis **4.**

In Abbildung 4, ist die Rückseite der LED-Matrix mit ihren PWM-LED-Treiber-Bauteilen (vereinfacht nur noch „PWM-ICs“ bezeichnet) zu sehen. Das Schieberegister jedes PWM-IC auf der LED-Matrix ist 16 Bit lang. Die Daten, die über die 16 Bit hinausgehen, werden ins Schieberegister des nächsten PWM-IC geschoben. Das letzte Datenbit im Schieberegister des letzten PWM-IC geht in der nächsten Verschiebung verloren. Da es sich bei der verwendeten LED-Matrix um jeweils 10 in Reihe geschaltete PWM-ICs handelt, hat das Schieberegister mit allen PWM-ICs insgesamt eine Gesamtlänge von 160 Bit. Dies entspricht der Anzahl der LEDs auf der Matrix in der Breite. Zu beachten ist, dass die einzelnen Datenbits des Schieberegisters nicht einer einzigen LED bestimmt sind, sondern dass der Wert im Schieberegister eines einzelnen PWM-IC für eines seiner 16 LEDs bestimmt ist. Auf die genauere Funktionsweise, wie die Farbdaten in die LED-Matrix geladen werden, wird in Kapitel 2.7 eingegangen.

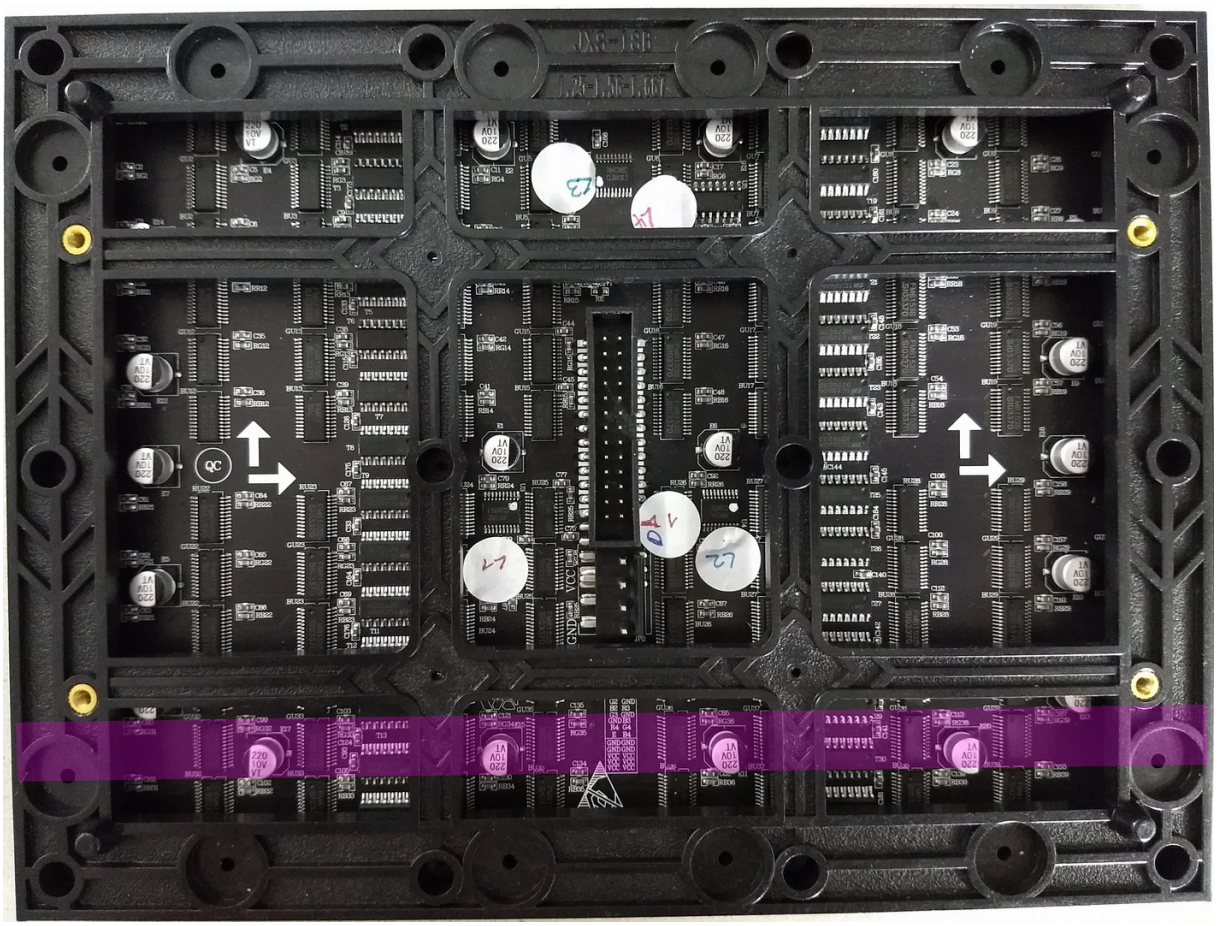

Abbildung 4: Rückseite der LED-Matrix. Die Vorletzte der zwölf Schieberegisterreihen ist violett markiert und gehört zum Pin **G4.** In jeder Schieberegisterreihe befinden sich jeweils zehn PWM-ICs. Manche PWM-ICs werden vom Gehäuse verdeckt.

2.6 Konfiguration

Bevor nun Farbdaten in die LED-Matrix geschoben werden, sollten zuallererst die PWM-ICs auf der LED-Matrix, die für das Auslesen und Abspeichern der Farbdaten zuständig sind, passend zur Anwendung konfiguriert werden. Diese PWM-ICs können universell in beliebig großen LED-Matrizen verwendet werden, auch wenn sich die Anzahl der LEDs unterscheidet. Von daher müssen die Konfigurationsregister der PWM-ICs so angepasst werden, damit die PWM-ICs vor allem über die Anzahl der Zeilen pro Reihe Bescheid wissen. Die anderen Konfigurationsmöglichkeiten beeinflussen wiederum den Stromverbrauch, die Bildqualität und die Interpretation der **GCLK**.

Alle PWM-ICs sind mit dem **LE**-Signal verbunden [6 S. 3]. Je nach dem, wie lange das **LE**-Signal gesetzt ist, wird dies von den PWM-ICs als ein bestimmter Befehl interpretiert. In Tabelle 3 ist ein Teil der möglichen Befehle aufgelistet und beschrieben.

Um beispielsweise das Konfigurationsregister 1 der PWM-ICs zu beschreiben, müssen die PWM-ICs in den Pre-Active Zustand versetzt werden, indem **LE** zu exakt 14 steigenden **DCLK-**Flanken HIGH gesetzt wird. Danach sollen die Konfigurationsdaten ins Schieberegister geladen und beim Schieben der letzten 4 Datenbits **LE** erneut HIGH gesetzt werden. Um das Konfigurationsregister 2 zu beschreiben, wird dasselbe Vorgehen verwendet, mit dem einzigen Unterschied, **LE** während den letzten 8 Datenbits HIGH zu setzen.

Die restlichen, in Tabelle 3 aufgelisteten Befehle benötigen keinen Pre-Active Befehl.

Um sich genauer vorstellen zu können, was die zwei jeweils 16 Bit langen Konfigurationsregister bezwecken, sind in Tabelle 4 alle Konfigurationsmöglichkeiten aufgelistet.

Befehl	Anzahl der steigenden Flanken der DCLK während LE gesetzt ist	Beschreibung
Data Latch	1	Daten im Schieberegister werden ins Ausgaberegister geladen.
VSYNC	2 oder 3	„Vertical Synchronous Signal“. Anzeigepuffer wird umgeschaltet.
Konfigurationsregister 1 beschreiben	4	Daten im Schieberegister werden ins Konfigurationsregister 1 geschrieben.
Konfigurationsregister 2 beschreiben	8	Daten im Schieberegister werden ins Konfigurationsregister 2 geschrieben.
Software Reset	10	PWM-IC zurücksetzen. Konfigurationsdaten bleiben bestehen.
Pre-Active	14	Wird benötigt, bevor bestimmte Befehle, wie zum Beispiel das Beschreiben der Konfigurationsregister ausgeführt werden.

Tabelle 3: Auflistung und Beschreibung der für dieses Projekt wichtigsten Befehle an die PWM-ICs der verwendeten LED-Matrix. Daten aus einem Datenblatt eines ähnlichen PWM-IC übernommen. Vollständige Auflistung und Beschreibung der Zustände sind im referenzierten Datenblatt zu finden. [6 S. 13-18]

Konfigurationsregister 1

Bit	Definition	Funktion
F	Eliminierung des Lower Ghost Problems	0: Duplikate von helleren Pixel können in der darauf-folgenden Scan Line leicht sichtbar sein [12 S. 12-13] 1: Duplikate hellerer Pixel werden verhindert
E-D	Reserviert	Unbekannt
C-8	Anzahl der Scan Lines	00000: 1 Zeilen ... 11111: 32 Zeilen
7	Farbtiefe	0: 16 Bit PWM 1: 14 Bit PWM (ignoriert die 2 niederwertigen Bits, beziehungsweise setzt sie auf 0) [12 S. 3 & 6]
6	GCLK Multiplizierer	0: Erhöht GCLK bei jeder steigenden Flanke 1: Erhöht GCLK bei jeder Flanke [12 S. 7]
5-0	Stromverstärkungs-faktor	000000: 12.5 % ... 111111: 200 % [12 S. 9-10]

Konfigurationsregister 2

Bit	Definition	Funktion
F-B	Reserviert	Unbekannt
A	Doppelte Aktualisierungsrate	0: Ausgeschaltet 1: Eingeschaltet
9-4	Reserviert	Unbekannt
3-1	Kompensation der Dimmung	000: 0 ns, 010: 10 ns, 100: 20 ns, 110: 30 ns 001: 5 ns, 011: 15 ns, 101: 25 ns, 111: 35 ns Dient zur Stabilisierung des Bildes, falls bestimmte Scan Lines unerwartet dunkler erscheinen [12 S. 14]
0	Reserviert	Unbekannt

Tabelle 4: Auflistung der Konfigurationsmöglichkeiten der PWM-ICs der verwendeten LED-Matrix. Daten aus einem Datenblatt eines ähnlichen PWM-IC übernommen. [6 S. 13-18]

2.7 Funktionsweise des Anzeigepuffers

Nachdem die LED-Matrix konfiguriert wurde, kann der Anzeigepuffer mit Daten befüllt werden. Die PWM-ICs der LED-Matrix lesen bei jeder steigenden **DCLK**-Flanke [12] die **R#**-, **G#**- und **B#**-Pins aus und schieben dieses Bit ins entsprechende Schieberegister. Wie schon erwähnt, haben die einzelnen Schieberegister für die verwendete LED-Matrix eine Gesamtlänge von 160 Bits. In der Breite hat die LED-Matrix zwar auch 160 LEDs, aber es sei zu beachten, dass die einzelnen Datenbits des Schieberegisters nicht einer einzigen LED bestimmt sind. Stattdessen wird das komplette 16 Bit große Schieberegister eines PWM-IC zu einem Zeitpunkt für eine einzige LED verwendet, indem der Wert im Schieberegister nach einem Data-Latch-Befehl in einem der 16 Ausgaberegister vom PWM-IC gespeichert wird. Der Wert im Ausgaberegister wird mittels PWM auf der LED ausgegeben. Im nächsten Durchgang wird der Wert für die nächste LED ins nächste Ausgaberegister gespeichert. Das wird so lange wiederholt, bis das letzte Ausgaberegister befüllt wurde. Nach diesem wird durch den Data-Latch-Befehl intern im Anzeigepuffer (nicht auf der Anzeige selbst) auf die nächste Zeile umgeschaltet. Der bisherige Vorgang wird von daher so lange wiederholt, bis die letzte Zeile erreicht wird und das Bild für das Display somit vollständig ist. Zum besseren Verständnis, dient das Beispiel in Abbildung 5 und dessen Beschreibung. Mit einem VSYNC-Befehl wird der gerade vorbereitete Anzeigepuffer angezeigt. Nach einem VSYNC-Befehl sollen für die Zeit, die es normalerweise braucht, um zwischen zwei Scan Lines umzuschalten, die **GCLK**, **DCLK** und die Datenleitungen **R#, G#** und **B#** nicht verändert werden. Damit der Wechsel problemlos verlaufen kann, muss die **GCLK** bereits seit Anfang des VSYNC-Befehls ausgeschaltet sein. [5 S. 11]

Nach dem VSYNC-Befehl kann jederzeit damit angefangen werden, auf dieselbe Weise das nächste Bild vorzubereiten. Bis zum nächsten VSYNC-Befehl bleibt das aktuell angezeigte Bild statisch.

Es muss darauf geachtet werden, dass zuerst die Daten für den letzten IC ins Schieberegister geschoben werden, dann die Daten vom Vorletzten und zuletzt die Daten vom Ersten, da die ersten abgeschickten Daten, zum Schluss am Ende des Schieberegisters liegen. Bei der verwendeten LED-Matrix wird das niederwertige Bit für die Farbdaten, welches die Helligkeit der LED am wenigsten beeinflusst, zuerst übertragen.

Bis hierhin sollten die Informationen genügen, damit nun ein grundlegendes Verständnis zur allgemeinen Ansteuerung einer LED-Matrix besteht. Um sicher zu gehen, ob die LED-Matrix so funktioniert, wie erwartet, nachdem die erwähnten

Schritte aus diesem Kapitel beachtet wurden, können dieselben Tests wie die in Kapitel 5 angewendet werden.

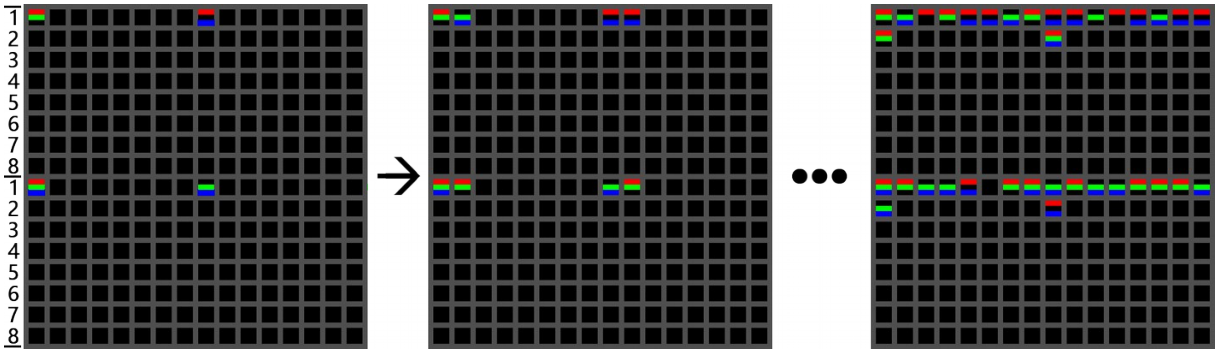

Abbildung 5: Skizze eines 16 × 16 Anzeigepuffers (nicht die tatsächliche Anzeige) mit zwei Reihen. Angenommen, die PWM-ICs haben acht Ausgaberegister und ein 16-Bit-Schieberegister. Der Anzeigepuffer ist vor dem ersten Durchgang idealerweise völlig schwarz. Nach dem ersten Durchgang (links) wurden 32 Bits (= Gesamtlänge des Schieberegisters) in die einzelnen Schieberegister geschoben und per Data-Latch-Befehl in den Anzeigepuffer geladen. Dasselbe wurde im nächsten Durchgang (mittig) gemacht. Die Darstellung rechts zeigt, wie der Anzeigepuffer nach dem neunten Durchgang aussehen könnte.

3 Anforderungen, Konzeption und Entwurf der Hardware

In diesem Kapitel wird die Entwicklung der Hardware für dieses Projekt beschrieben. Darunter fällt hauptsächlich die Entwicklung und Bestückung der Leiterplatte. Obwohl dieses Thema eher zum Fachgebiet der Elektrotechnik statt der Informatik gehört, war der Aufwand so signifikant für dieses Projekt, sodass einige Einzelheiten darüber erklärt werden.

3.1 Anforderungen und Peripherie der Leiterplatte

Die Bachelorarbeit soll als Ergebnis eine ansehnliche Konstruktion darbieten. Es sollen vier LED-Matrizen auf einer Leiterplatte montiert werden, sodass ein lückenloses Display mit einer Gesamtanzahl von 320 × 240 RGB-LEDs und einer Gesamtgröße von 40 × 30 cm entsteht.

Wie in Abbildung 4 zu sehen ist, haben die LED-Matrizen auf der Rückseite ein Gehäuse mit vier Gewinde (M3) und vier Zapfen. Um das auszunutzen, sollen die LED-Matrizen mithilfe von Schrauben (M3) fest auf eine Leiterplatte montiert werden.

Für die Schrauben und Zapfen sollen entsprechende Bohrungen an der Leiterplatte durchgeführt werden.

Die Seite der Leiterplatte, auf der die LED-Matrizen aufliegen, wird als Vorderseite gesehen. Auf der Vorderseite der Leiterplatte sollen sich lediglich vier Buchsenleisten befinden, die für die Kontaktierung der Kommunikationsschnittstellen der LED-Matrizen benötigt werden. Die Buchsenleisten sollen so gewählt und exakt positioniert werden, sodass die LED-Matrizen ohne zusätzliche Kabel direkt auf diese gesteckt werden können.

Die Rückseite der Leiterplatte wird mit dem für die Ansteuerung der LED-Matrizen benötigten Mikrocontroller, einem 14-poligen Steckverbinder für eine JTAG-Schnittstelle, einer USB-2.0-Typ-B-, einer RJ45-, einer RS485- und zwei 7-poligen-Crimp-Buchsen, sowie dem Rest der Schaltung bestückt. Ebenfalls benötigt wird ein Anschluss für die Stromversorgung der gesamten Leiterplatte, sowie vier zusätzliche Stromversorgungsbuchsen für die jeweiligen Matrizen. Diese vier Stromversorgungsbuchsen sind nötig, da die zusätzlichen Stromversorgungspins auf den Matrizen sehr ungenau platziert wurden und somit ungleichmäßig positioniert sind. Stattdessen soll sich in der Leiterplatte eine Aussparung über den Stromversorgungspins der LED-Matrizen befinden. Durch diese Aussparung kann mithilfe von entsprechend gecrimpten Kabel die Stromversorgung zwischen der Leiterplatte und den LED-Matrizen hergestellt werden.

JTAG: Der 14-polige Steckverbinder soll für das Hochladen und Debuggen des Programms auf dem Mikrocontroller mittels JTAG verwendet werden.

USB: Mit der USB-2.0-Typ-B-Buchse kann der Mikrocontroller als USB-Peripherie-Gerät verwendet werden. Es stehen viele verschiedene USB-Klassen zur Verfügung. Die sogenannte „CDC“ („communications device class“), welche einer seriellen Kommunikation ähnelt, würde sich für die Anwendung eignen, ebenso wie die Klassen „Still Imaging“ oder „Video“. [25]

RJ45: Der RJ45-Anschluss soll, wie üblich, als Ethernet-Anschluss verwendet werden. Die RJ45-Buchse soll sogenannte Magnetics beinhalten, womit der RJ45-Stecker und die Schaltung der Leiterplatte galvanisch getrennt werden. Das Signal geht dann an den PHY-Chip (LAN8710A von Microchip), welcher die ankommenden Ethernet-Daten für den Mikrocontroller dekodiert, sowie die ausgehenden Daten kodiert.

RS485: Der RS485-Anschluss soll als serielle Kommunikationsschnittstelle dienen. Die RS485-Schnittstelle hat zwei differenzielle Datenleitungen und eine Leitung für die

Erdung. Im Vergleich zu Schnittstellen mit einer einzelnen Datenleitung hat RS485 den Vorteil, dass sie weniger empfindlich gegenüber Störungen sind und somit längere Leitungen und höhere Übertragungsgeschwindigkeiten verwendet werden können. Das ankommende Signal muss mit dem Transceiver-Bauteil (SN65176BD von Texas Instruments) ins für den Mikrocontroller verständliche UART-Protokoll umgewandelt werden. Ebenso werden mit diesem Bauteil ausgehende UART-Daten für die RS485-Schnittstelle umgewandelt.

Crimp-Anschlüsse: An den zwei 7-poligen-Crimp-Anschlüssen soll jeweils ein Kabel angesteckt werden, das am anderen Ende mit einer SNES-Buchse verlötet ist. Zwischen dem Display und dem Boden soll sich ein kleines Gehäuse befinden, aus dem die zwei SNES-Buchsen austreten sollen. An diesen Buchsen sollen SNES-Controller für die Bedienung des Displays verbunden werden können.

Taster, Schalter und LEDs zum Debuggen: Außerdem soll auf der Leiterplatte ein Taster zum Ein- und Ausschalten, sowie eine Reihe von 5 Schaltern, deren Funktionen in Tabelle 5 aufgelistet sind, bestückt werden. Zudem sollen noch vier Debug-LEDs, wie sie gerne genannt werden, angebracht werden, um beispielsweise während der Software-Entwicklung effizientere Tests durchführen zu können.

Schalter	Funktion wenn geschlossen	Funktion wenn offen
1	JTAG: CPU Reset	JTAG: Run CPU
2	JTAG: User Boot HIGH	JTAG: User Boot LOW
3	JTAG: User Boot verbunden	JTAG: User Boot getrennt
4	Ethernet: COL verbunden	Ethernet: COL getrennt
5	Reserviert	Reserviert

Tabelle 5: Funktion der Schalter auf der Leiterplatte. Schalter 3 und 4 dürfen nicht zur selben Zeit geschlossen sein, ansonsten sind JTAG: User Boot und Ethernet: COL miteinander verbunden.

Kupferschichten: Die Leiterplatte soll aus zwei Kupferschichten bestehen. Im Gegensatz zu Leiterplatten mit vier oder mehr Kupferschichten wird dadurch zwar das Verlegen der Leitungen während der Entwurfsphase erschwert, aber die Komplexität und die Anschaffungskosten verringert. Jede freie Fläche soll als gemeinsame Massefläche genutzt werden. Das hat mehrere Vorteile. Einer wäre, dass sich der Strom selbst über die gesamte Massefläche den kürzesten Weg suchen kann. Abschnürungen der Massefläche sowie „Kupferinseln“ sollten deswegen so gut wie möglich vermieden werden, auch wenn das bei geringerer Anzahl an Kupferschichten schwieriger ist. Ein weiterer Vorteil ist, dass die Massefläche einen schirmenden

Effekt [26] haben kann. Andere elektrische Geräte werden also vor der elektromagnetischen Strahlung von Leiterbahnen hochfrequenter Signale, wie es bei Quarzen zum Beispiel der Fall ist, geschützt. Wenn die Leiterplatte nun mehr als zwei Kupferschichten hätte, könnten die Leiterbahnen dieser Quarze von beiden Seiten mit der Massefläche umschlossen werden, um den Effekt auch beidseitig zu nutzen. Auch umgekehrt, hat die Massefläche einen schirmenden Effekt vor parasitären Signalen anderer Elektrogeräte. Auch die bessere Wärmeverteilung und Kühlung ist eines der vielen weiteren Vorteile einer gesamten Massefläche.

Beine: Die Konstruktion soll mithilfe von zwei Kunststoffbeinen, die auf der Leiterplatte an entsprechend gebohrte Löcher angeschraubt werden, mit einem Winkel von 70° zum Boden stehen. Deswegen muss beachtet werden, dass die Anschlüsse hoch genug platziert sind, um zu verhindern, dass die Bauteile, sowie die eingesteckten Stecker auf den Boden anstoßen und dadurch die gesamte Konstruktion eventuell (um)kippen.

Passive Bauteile: Auf die vielen Widerstände, Kondensatoren, Dioden und Ferritperlen, welche unter anderem zur Entstörung von Signalen oder zur elektrischen Stabilisation notwendig sind, wird in dieser Arbeit nicht näher eingegangen.

3.2 Anforderungen an den Mikrocontroller

Als Mikrocontroller wurde das Renesas RX65N der 176-Pin-Variante gewählt. Grund dafür sind die ausreichend hohe Taktgeschwindigkeit von 120 MHz [28] und die benötigte Anzahl an GPIO-Pins, Timermodulen und -ausgängen, sowie die seriellen Kommunikationsmodule, das USB-Modul und Ethernetmodul.

Genutzt werden insgesamt 144 Pins:

- 66 Pins für alle vier Matrizen, von denen
  - jeweils 14 dedizierte Signale pro LED-Matrix sind,
  - 8 gemeinsame Signale sind,
  - 1 Pin die GCLK mitzählt und
  - 1 Pin die GCLK mittels &-Gatter de-/aktiviert,
- 19 Pins für die Ethernet-Schnittstelle,
- 3 Pins für die USB-Schnittstelle,
- 3 Pins für die serielle Kommunikationsschnittstelle RS485,
- 6 Pins für beide SNES-Controller zur Bedienung,
- 8 Pins für die JTAG-Schnittstelle,
- 1 Pin für einen Taster zum Ein- und Ausschalten,
- 4 Pins für vier Debug-LEDs,
- 2 Pins an einem Uhrenquarz (32768 Hz),
- 2 Pins an einem Quarz (12 MHz), das den Takt für den Mikrocontroller vorgibt,
- 14 Pins an die Stromversorgung angeschlossen,
- 16 geerdete Pins.

32 Pins sind ungenutzt. In Abbildung 6 wird die Pinbelegung dargestellt. Tabelle 6 listet auf, mit welchen Leitungen die Pins verbunden sind.

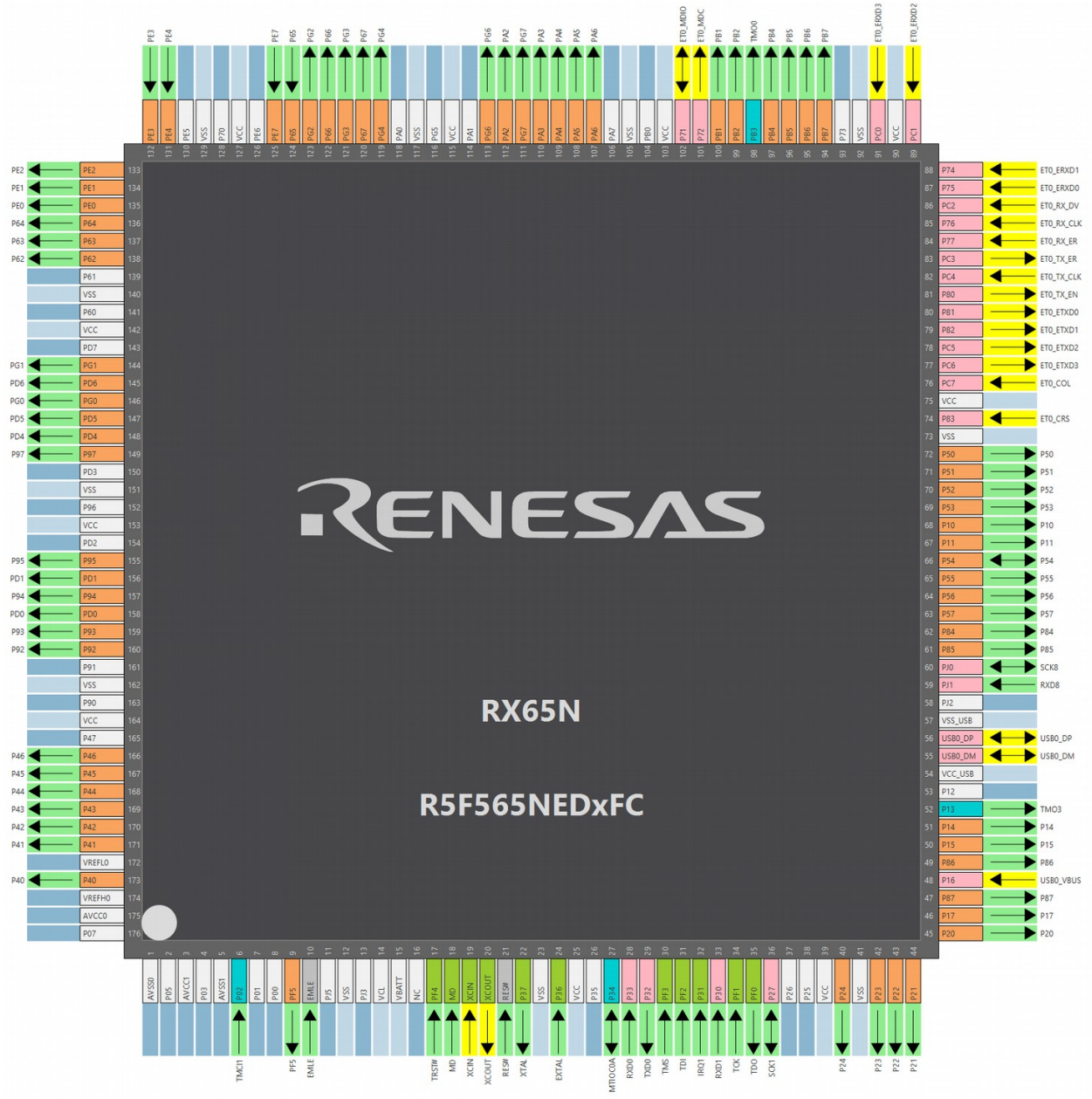

Abbildung 6: Darstellung der Pinbelegung des verwendeten Mikrocontrollers in der Entwicklungsumgebung e² studio.

Pin	Ausgewähltes Modul	Verbindung
1	AVSS0	Erde
2	-	-
3	AVCC1	Stromversorgung
4	-	-
5	AVSS1	Erde
6	TMCI1	LED-Matrix: GCLK (Zähler)
7	-	-
8	-	-
9	-	-
10	EMLE	JTAG: EMLE-Pin
11	-	-
12	VSS	Stromversorgung
13	-	-
14	VCL	Erde
15	VBATT	-
16	-	-
17	TRST#	JTAG: TRST-Pin
18	MD	JTAG: MD-Pin
19	XCIN	Uhrenquarz (32768 Hz)
20	XCOUT	Uhrenquarz (32768 Hz)
21	RES#	JTAG: RES-Pin
22	XTAL	Quarz (Mikrocontroller Taktgeber, 12 MHz)
23	VSS	Erde
24	EXTAL	Quarz (Mikrocontroller Taktgeber, 12 MHz)
25	VCC	Stromversorgung
26	-	-
27	MTIOC0A	Bedienung: SNES-Controller Latch Enable
28	RXD0	Serielle Komm. RS485: Empfangsleitung (aus Sicht des Mikrocontrollers)
29	TXD0	Serielle Komm. RS485: Sendeleitung (aus Sicht des Mikrocontrollers)
30	TMS	JTAG: TMS-Pin
31	TDI	JTAG: TDI-Pin
32	IRQ1	Taster: Externer Interrupt
33	RXD1	Bedienung: SNES-Controller 2 Datenleitung
34	TCK	JTAG: TCK-Pin

Pin	Ausgewähltes Modul	Verbindung
35	TDO	JTAG: TDO-Pin
36	SCK1	Bedienung: SNES-Controller Takt (Ausgabe)
37	-	-
38	-	-
39	VCC	Stromversorgung
40	P24	LED-Matrix: E
41	VSS	Erde
42	P23	LED-Matrix: D
43	P22	LED-Matrix: C
44	P21	LED-Matrix: B
45	P20	LED-Matrix: A
46	P17	LED-Matrix 4 Stromversorgung De-/Aktivierung
47	P87	LED-Matrix 3 Stromversorgung De-/Aktivierung
48	USB0_VBUS	USB: VBUS
49	P86	Bedienung: SNES-Controller Latch Enable
50	P15	LED-Matrix 2 Stromversorgung De-/Aktivierung
51	P14	LED-Matrix 1 Stromversorgung De-/Aktivierung
52	TMO3	LED-Matrix: GCLK (Ausgabe)
53	-	-
54	VCC_USB	Stromversorgung
55	USB0_DP	USB: Positive Datenleitung
56	USB0_DM	USB: Negative Datenleitung
57	VSS_USB	Erde
58	-	-
59	RXD8	Bedienung: SNES-Controller 1 Datenleitung
60	SCK8	Bedienung: SNES-Controller Takt (Empfängt Takt von Pin 36: SCK1)
61	P85	LED-Matrix 3: B3
62	P84	LED-Matrix 3: G3
63	P57	LED-Matrix 3: R4
64	P56	LED-Matrix 3: B3
65	P55	LED-Matrix 3: G3
66	P54	LED-Matrix 3: R3
67	P11	LED-Matrix 3: B2
68	P10	LED-Matrix 3: G2

Pin	Ausgewähltes Modul	Verbindung
69	P53	LED-Matrix 3: R2
70	P52	LED-Matrix 3: B1
71	P51	LED-Matrix 3: G1
72	P50	LED-Matrix 3: R1
73	VSS	Erde
74	ET0_CRS	Ethernet: CRS
75	VCC	Stromversorgung
76	ET0_COL	Ethernet: COL/CRS_DV/MODE2 oder JTAG: UB (siehe Tabelle 5)
77	ET0_ETXD3	Ethernet: TXD3
78	ET0_ETXD2	Ethernet: TXD2
79	ET0_ETXD1	Ethernet: TXD1
80	ET0_ETXD0	Ethernet: TXD0
81	ET0_TX_EN	Ethernet: TXEN
82	ET0_TX_CLK	Ethernet: TXCLK
83	ET0_TX_ER	Ethernet: NINT/TXER/TXD4
84	ET0_RX_ER	Ethernet: RXER/RXD3/PHYAD0
85	ET0_RX_CLK	Ethernet: RXCLK/PHYAD1
86	ET0_RX_DV	Ethernet: RXDV
87	ET0_ERXD0	Ethernet: RXD0/MODE0
88	ET0_ERXD1	Ethernet: RXD1/MODE1
89	ET0_ERXD2	Ethernet: RXD2/RMIISEL
90	VCC	Stromversorgung
91	ET0_ERXD3	Ethernet: RXD3/PHYAD2
92	VSS	Erde
93	-	-
94	PB7	Debug-LED 4
95	PB6	Debug-LED 3
96	PB5	Debug-LED 2
97	PB4	Debug-LED 1
98	TMO0	LED-Matrix: GCLK De-/Aktivierung
99	PB2	LED-Matrix: DCLK
100	PB1	LED-Matrix: Latch Enable
101	ET0_MDC	Ethernet: MDC
102	ET0_MDIO	Ethernet: MDIO

Pin	Ausgewähltes Modul	Verbindung
103	VCC	Stromversorgung
104	PB0	Ethernet: Reset
105	VSS	Erde
106	-	-
107	PA6	LED-Matrix 1: B4
108	PA5	LED-Matrix 1: G4
109	PA4	LED-Matrix 1: R4
110	PA3	LED-Matrix 1: B3
111	PG7	LED-Matrix 1: G3
112	PA2	LED-Matrix 1: R3
113	PG6	LED-Matrix 1: B2
114	-	-
115	VCC	Stromversorgung
116	-	-
117	VSS	Erde
118	-	-
119	PG4	LED-Matrix 1: G2
120	P67	LED-Matrix 1: R2
121	PG3	LED-Matrix 1: B1
122	P66	LED-Matrix 1: G1
123	PG2	LED-Matrix 1: R1
124	P65	LED-Matrix 3 Stromversorgung: Fault Signal
125	PE7	LED-Matrix 4 Stromversorgung: Fault Signal
126	-	-
127	VCC	Stromversorgung
128	-	-
129	VSS	Erde
130	ETH_REF50CK0	- (Verbindung mit Ethernet XTAL1/CLKIN nachträglich möglich)
131	PE4	LED-Matrix 1 Stromversorgung: Fault Signal
132	PE3	LED-Matrix 2 Stromversorgung: Fault Signal
133	PE2	LED-Matrix 2: R1
134	PE1	LED-Matrix 2: G1
135	PE0	LED-Matrix 2: B1
136	P64	LED-Matrix 2: R2

Pin	Ausgewähltes Modul	Verbindung
137	P63	LED-Matrix 2: G2
138	P62	LED-Matrix 2: B2
139	-	-
140	VSS	Erde
141	-	-
142	VCC	Stromversorgung
143	-	-
144	PG1	LED-Matrix 2: R3
145	PD6	LED-Matrix 2: G3
146	PG0	LED-Matrix 2: B3
147	PD5	LED-Matrix 2: R4
148	PD4	LED-Matrix 2: G4
149	P97	LED-Matrix 2: B4
150	-	-
151	VSS	Erde
152	-	-
153	VCC	Stromversorgung
154	-	-
155	P95	LED-Matrix 4: R1
156	PD1	LED-Matrix 4: G1
157	P94	LED-Matrix 4: B1
158	PD0	LED-Matrix 4: R2
159	P93	LED-Matrix 4: G2
160	P92	LED-Matrix 4: B2
161	-	-
162	VSS	Erde
163	-	-
164	VCC	Stromversorgung
165	-	-
166	P46	LED-Matrix 4: R3
167	P45	LED-Matrix 4: G3
168	P44	LED-Matrix 4: B3
169	P43	LED-Matrix 4: R4
170	P42	LED-Matrix 4: G4

Pin	Ausgewähltes Modul	Verbindung
171	P41	LED-Matrix 4: B4
172	VERFL0	Erde
173	P40	Serielle Komm. RS485: Senden De-/Aktivierung
174	VREFH0	Erde
175	AVCC0	Stromversorgung
176	-	-

Tabelle 6: Liste der Pins des verwendeten RX65N-Mikrocontrollers, deren ausgewähltes Modul und die Leitungen, mit denen die Pins verbunden werden sollen. (# kennzeichnet Active-LOW Signale)

Der Arbeitsspeicher soll groß genug sein, um darin vor allem ein 320 × 240 × 3 (240 LEDs in der Höhe, 320 LEDs in der Breite mit jeweils den Rot-, Grün- und Blauwerten) großes Anzeigepuffer-Array für das anzuzeigende Bild, sowie einige weniger speicherlastige Variablen hinterlegen zu können. Um Farbdaten mit einer Farbtiefe von 8 Bit im Anzeigepuffer-Array hinterlegen zu können, werden 225 KiB (= Anzahl der LEDs × Anzahl benötigter Bits pro Farbwert ÷ 8) benötigt. Der 250 KiB [28] große Arbeitsspeicher des RX65N ist somit ausreichend groß, um auch ein wenig Speicherplatz für die sonstigen Variablen übrig zu haben. Es gäbe noch einen zweiten getrennten Arbeitsspeicher mit einer Größe von 375 KiB, in welchem ein zweites Array für eine Doppelpufferung gespeichert werden könnte.

3.3 Entwurf der Leiterplatte

Der Entwurf der Schaltung und der Leiterplatte wurde mit EAGLE (Einfach Anzuwendender Grafischer Layout Editor) Version 7.0.0, einer Entwurfssoftware für elektronische Systeme, gestaltet. Abbildung 7 zeigt den zur Produktion freigegebenen Leiterplattenentwurf und Abbildung 8 die produzierte und bestückte Leiterplatte.

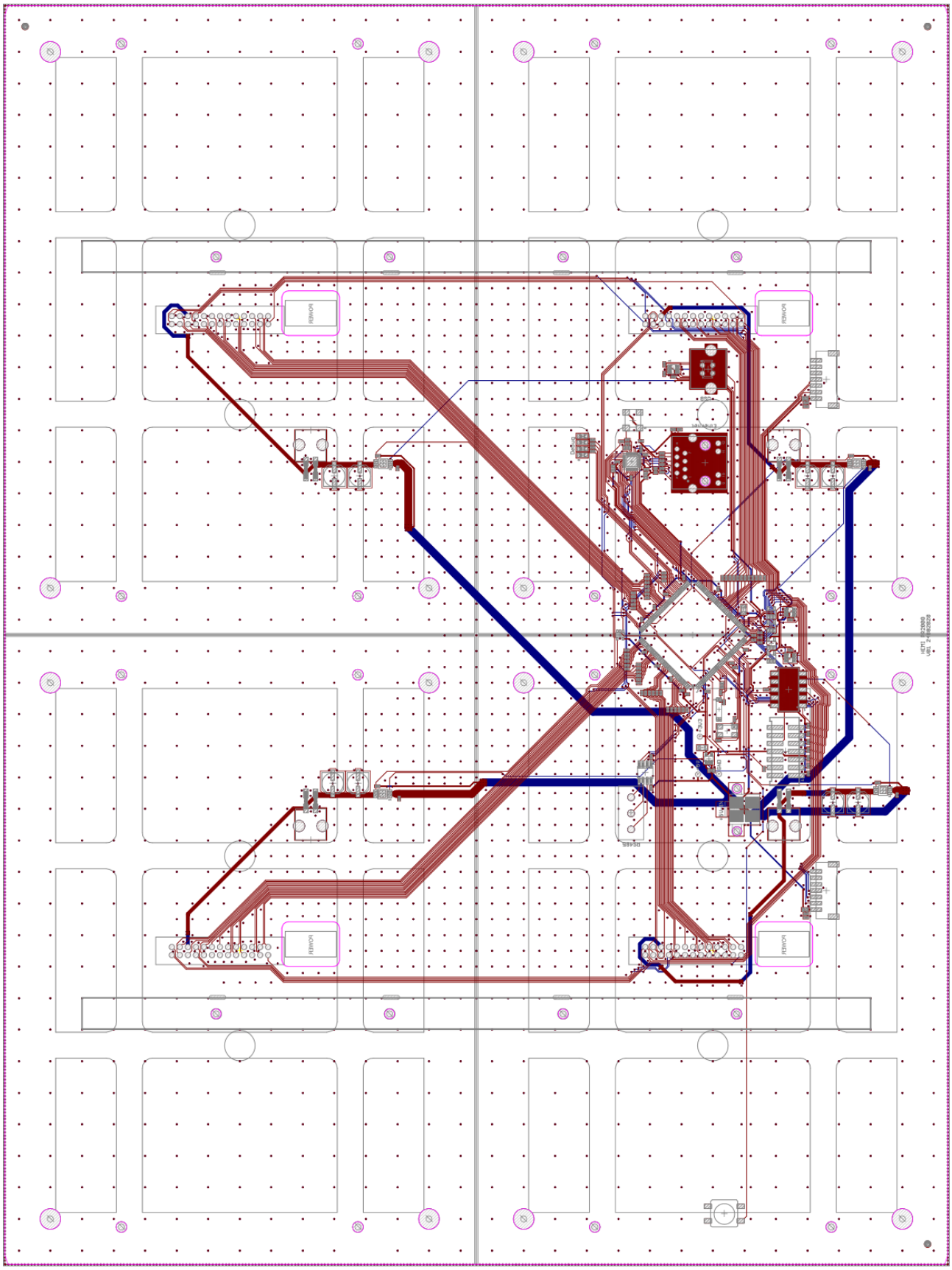

Abbildung 7: Leiterplattenentwurf im Maßstab 1:1,765.

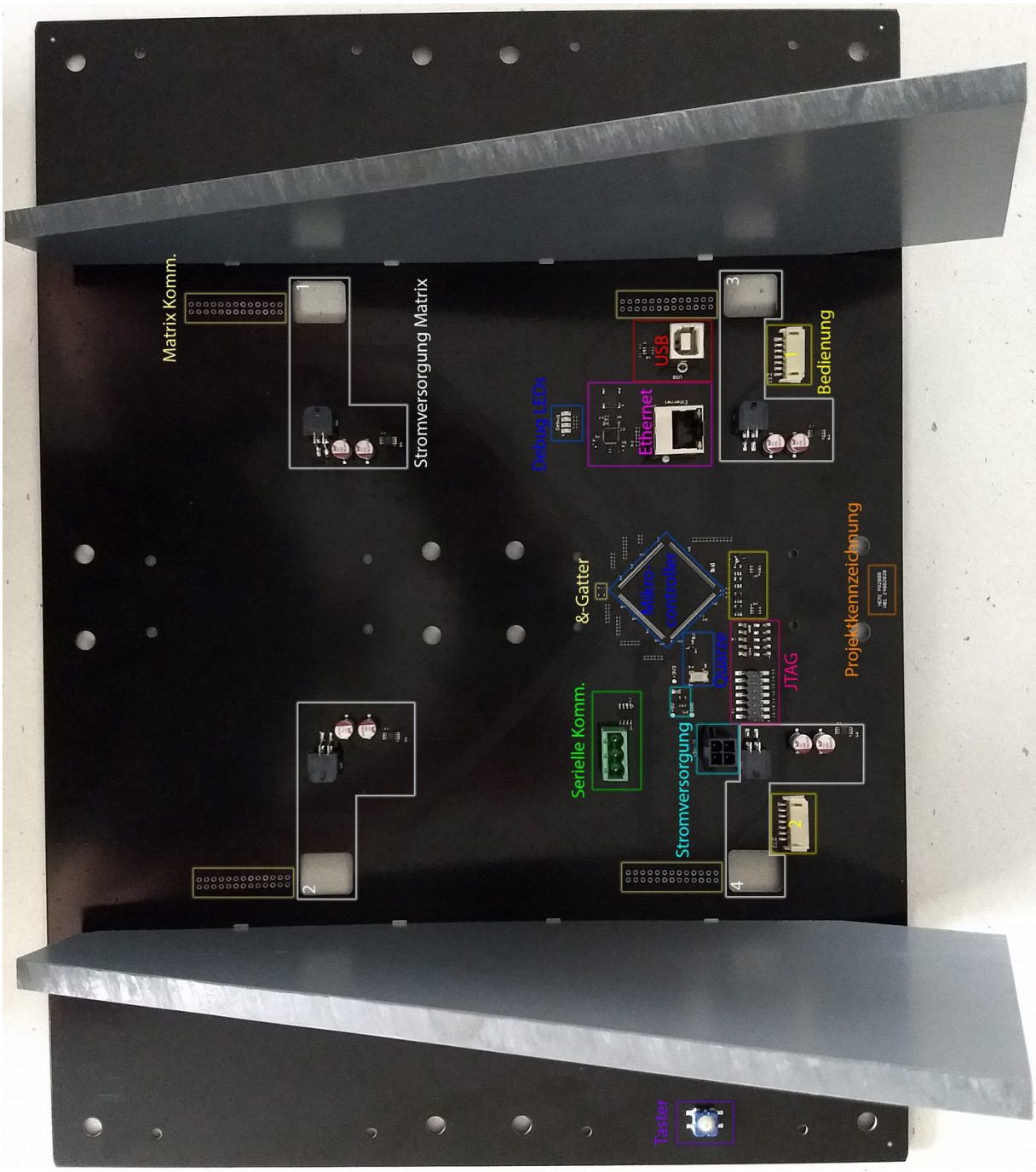

Abbildung 8: Bestückte Leiterplatte ohne LED-Matrizen.

3.4 Probleme und Fehler am Leiterplattenentwurf

Der Mensch ist dafür bekannt, Fehler zu machen und daraus zu lernen. Der erste Leiterplattenentwurf ist selten einwandfrei. Im Rahmen der Bachelorarbeit sind zwei Fehler am Leiterplattenentwurf bekannt geworden.

3.4.1 Falsch verbundene Bauteile

Ein bekannter Fehler in diesem Projekt ist, dass zwei von den vier Pegelwandlern, welche die Spannung von 5 V der SNES-Controller auf die für den Mikrocontroller verträglichen 3,3 V wandelt, verdreht in die Schaltungsskizze gezeichnet wurden. Ohne darauf zu achten, wurde dadurch die Stromversorgung und Erdung verpolt. Als versucht wurde, die Leiterplatte zum ersten Mal zu betreiben, verhinderte der daraus resultierende Kurzschluss die Weiterarbeit. Um das Problem zu beheben, mussten die zwei Pegelwandler entfernt werden. Dadurch wurde der Kontakt mit der Datenleitung beider SNES-Controller gekappt. Die Bedienung mittels SNES-Controller ist somit nicht möglich.

Damit die Pegelwandler nicht verloren gehen, wurde jeweils ein Pin dieser Bauteile an einer offen liegenden Kupferschicht verlötet, an welcher sie die Schaltung nicht beeinflussen.

3.4.2 Fiducials außerhalb des Sichtbereichs der Bestückungsmaschine

Der zweite Fehler hängt mit der Bestückungsmaschine zusammen, die in der Firma verwendet wird, um die Bauteile auf die Leiterplatte zu platzieren. Die Leiterplatte für dieses Projekt ist 399 mm breit und 299 mm hoch. Die vorgeschriebenen Maximalmaße für die in der Firma vorhandene Bestückungsmaschine FOX2 von Essemtec betragen 406 × 305 mm [29]. Der Sichtbereich der Kamera der Bestückungsmaschine ist allerdings kleiner.

Auf der Leiterplatte befinden sich runde offene Kupferschichten als Orientierungspunkte, sogenannte Fiducials, an denen sich die Bestückungsmaschine für den Bestückungsprozess ausrichten kann. Leider befinden sich die Fiducials an den Ecken der Leiterplatte mit 1 cm Abstand zum Leiterplattenrand, sodass die Kamera nicht mehr als ein Fiducial gleichzeitig erfassen konnte.

Stattdessen musste eine andere Kupferfläche als Orientierungspunkt ausgewählt werden, welche die Kamera nicht mit einer Anderen verwechseln kann, um die Bestückung zu beginnen. Zwei Orientierungspunkte sind ausreichend, da die Bestückung auf einer zweidimensionalen Fläche durchgeführt wird. Trotzdem ist es besser grundsätzlich drei Fiducials auf einer Leiterplatte zu haben. Wenn die Bestückungsmaschine eines der Fiducials mit einem Via, einer kleinen Bohrung welche zwei oder mehr Kupferschichten miteinander kontaktiert, verwechselt und die Abweichung die Toleranzwerte nicht überschreitet, kann sich die Bestückungsmaschine mit den beiden anderen Fiducials trotzdem noch korrekt ausrichten.

4 Implementierung eines high-density LED-Matrix Treibers und dessen Bedienung

Da die LED-Matrix kein Datenblatt besitzt, war die allererste Aufgabe, die LED-Matrix mithilfe eines RX65N Mikrocontrollers auf einem Evaluation Board (bei Renesas als „Target Board“ bezeichnet) mit einem „fliegenden“ Aufbau anzusteuern, welcher in Abbildung 9 zu sehen ist.

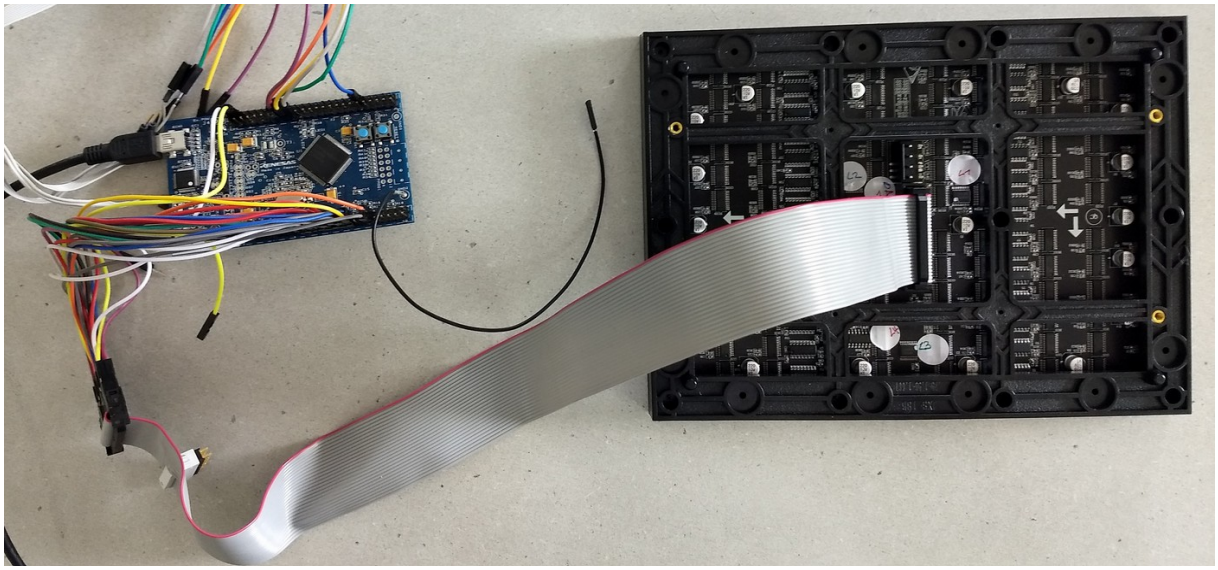

Abbildung 9: RX65N Target Board verbunden mit einer LED-Matrix.

Stattdessen waren für Ansteuerung der LED-Matrix Datenblätter [5, 11, 12] eines ähnlichen Produkts einer anderen Firma zum Teil hilfreich. Die zur allgemeinen Ansteuerung einer LED-Matrix wichtigsten Informationen aus diesen Datenblättern wurden in Kapitel 2 zusammengefasst und in der Implementierung angewendet.

Zum Programmieren wurde die Programmiersprache C und die Entwicklungsumgebung e² studio verwendet.

4.1 Timings

Die Clocks der Matrizen, sowie die der SNES-Controller, die mit Timerausgängen des Mikrocontrollers verbunden sind, geben jeweils einen unterschiedlichen Takt wieder.

4.1.1 Clocks der LED-Matrix

Für die LED-Matrix hat es sich bewährt, der **GCLK** bei eingeschaltetem GCLK Multiplizierer, welcher in Tabelle 4 beschrieben ist, eine Periodenlänge von 166 ns (≈ 6 MHz) zu geben. Siehe Kapitel 2.4 und Tabelle 2 für die Bedeutung der **GCLK**. Die Perioden-

länge der **DCLK** beträgt während der Übertragung der Farbdaten höchstens 16 µs (62,5 kHz). Siehe Kapitel 2.5 und Tabelle 2 für die Bedeutung der **DCLK**. Die Haltezeit der **GCLK** nachdem ein Scan-Line-Wechsel beträgt 25 **GCLK**-Flanken (= 2,075 µs). Wegen mangelnden Informationen aus dem offiziellen Datenblatt ist zu bezweifeln, dass die gewählten Timings den optimalen Timings für die verwendete LED-Matrix entspricht.

Nach jeder fallenden **DCLK**-Flanke werden die Farbdatenbits der aktuellen Pixel an den **R#**, **G#** und **B#**-Pins gesetzt. Die PWM-ICs der LED-Matrix lesen bei jeder steigenden **DCLK**-Flanke [12] die **R#**, **G#** und **B#**-Pins aus und schieben dieses Bit in das Schieberegister. Die Periodenlänge der **DCLK** ist unterschiedlich, da das Signal so lange HIGH bleibt, bis die ISR vollständig ausgeführt wird, während die LOW-Zeit so kurz wie möglich ist. Die HIGH-Zeit beträgt höchstens 15,4 µs und die LOW-Zeit 0,6 µs. Die HIGH-Zeit, als auch die Periodenlänge der **DCLK** ist kürzer, wenn keine Farbdaten übertragen werden, da das Auslesen und Setzen der Farbdatenbits die längste Rechenzeit während der ISR in Anspruch nimmt.

4.1.2 Clock und Datenübertragung von SNES-Controllern

Für die SNES-Controller soll eine Clockperiode 12 µs (83,33 kHz) lang sein. Ein SNES-Controller beinhaltet ein 16 Bit Ausgaberegister. Genau so wie die Daten vom Mikrocontroller ins Schieberegister einer LED-Matrix geschoben werden, sollen auch hier nacheinander die einzelnen Bits des Ausgaberegisters des SNES-Controllers in den Mikrocontroller gespeichert werden. Die einzelnen Bits im Ausgaberegister spiegeln einzelne Knöpfe wieder. Falls ein Knopf gedrückt wird, ist das dazugehörige Bit HIGH. Auch wie bei der LED-Matrix, wird hier ein Data-Latch-Befehl zum Übertragen der Daten verwendet. Dieser soll exakt eine steigende Clock-Flanke HIGH gesetzt werden, zudem jedoch konsistent alle 16,67 ms beziehungsweise mit einer Frequenz von rund 60 Hz ausgelöst werden. Nach dem Data-Latch-Befehl sind die Bits im Ausgangsregister wieder LOW gesetzt. [30]

4.2 Taster zum Ein- und Ausschalten

Das Drücken des Tasters löst einen Interrupt aus.

Im eingeschalteten Zustand wird in der ISR ein Timer gestartet. Der Timer löst nach 500 ms einen weiteren Interrupt aus. Falls der Taster zu diesem Zeitpunkt weiterhin beziehungsweise wieder gedrückt bleibt, wird die Stromversorgung der LED-Matrizen,

die mit GPIO-Pins gesteuert wird, unterbrochen und die Timermodule für die **GCLK** und **DCLK** gestoppt.

Im ausgeschalteten Zustand wird in der ISR der Mikrocontroller in einen Zustand versetzt, der die Timermodule für die **GCLK** und **DCLK** startet, Variablen und GPIO-Pins auf ihre Startwerte zurücksetzt, die LED-Matrizen konfiguriert und das im Flashspeicher abgelegte Bild anzeigen lässt.

4.3 Serielle Kommunikationsschnittstelle zur Remote-Bedienung

Über den RS485-Anschluss können zwischen dem UART-Modul des Mikrocontrollers und einer seriellen Kommunikationsschnittstelle eines PCs Daten in ASCII gesendet und empfangen werden. Hierzu werden am PC Anwendungen, wie beispielsweise PuTTY oder HTerm, verwendet und die Übertragungsgeschwindigkeit sowie das Format konfiguriert. Der Mikrocontroller ist so konfiguriert, sodass eine Übertragungsgeschwindigkeit von 25000 Baud und das Format 8N1 (acht Datenbits, ohne Paritätsbit, ein Stoppbit) verwendet werden soll. Als terminierendes Zeichen, das den Empfangspuffer des Mikrocontrollers zurücksetzt und gegebenenfalls den aktuellen Befehl abbricht, sollen die in Tabelle 7 aufgeführten Steuerungszeichen verwendet werden.

Name	Tastenkombination	String	Zeichen	Hex.	Dez.
Null	Strg+@	„\0“	**NUL**	0x00	00
End Of Text	Strg+C		**EOT**	0x03	03
End Of Transmission	Strg+D		**ETX**	0x04	04
Line Feed	Strg+J	„\n“	**LF**	0x0A	10
Carriage Return	Strg+M	„\r“	**CR**	0x0D	13
Cancel	Strg+X		**CAN**	0x18	24
Escape	Strg+[		**ESC**	0x1B	27
Space (Leerzeichen)		„ “	**SP**	0x20	32

Tabelle 7: Terminierende Zeichen der seriellen Kommunikation [1 S. 1040]

Alle Befehle, bis auf `help`, bestehen aus exakt zwei Zeichen, um die benötigte Zeit der ISR, in der die Zeichen ausgelesen werden, kurz zu halten. Auch wird aus Performancegründen ein `switch-case`-statement, statt beispielsweise einem `strcmp` verwendet, was allerdings oftmals abzuraten ist, da die Wartung des Programmcodes durch die schlechtere Lesbarkeit mühsam werden könnte.

Da die ISR trotzdem gelegentlich die ISR für die **DCLK** blockieren kann, wäre es vermutlich besser, den sogenannten „DMA"-Controller („direct memory access") vom RX65N-Mikrocontroller einzusetzen. Dieser würde bewirken, dass die ankommenden Zeichen ohne Interrupt oder Verwendung des Prozessors in den Speicher abgelegt werden und, weil auch nicht mehr auf den Abschluss einer ISR gewartet werden müsste, dadurch auch die Baudrate erhöht werden könnte.

In Tabelle 8 werden alle möglichen Befehle aufgelistet.

Befehl	**Parameter / Beispiel**	**Kurzbeschreibung**
`d#`	`d1/d2/.../d9`	Zeigt eines der Demobilder an
`ds`	keine	Startet/Stoppt die Diashow, in welcher die Demo- oder Testbilder periodisch in Endlosschleife umgeschaltet werden
`help`	keine	Listet alle möglichen Befehle auf
`im`	`x#x#x#...x#` `{0x12, 0x34, 0x56,..., 0xef}`	Mit der Übergabe eines C-Arrays, welches RGB Farbwerte, angefangen mit dem ersten Pixel, beinhaltet, wird temporär ein neues Bild ins Anzeigepuffer-Array eingespielt
`m1`	keine	Schaltet die LED-Matrix links-oben ein/aus
`m2`	keine	Schaltet die LED-Matrix rechts-oben ein/aus
`m3`	keine	Schaltet die LED-Matrix links-unten ein/aus
`m4`	keine	Schaltet die LED-Matrix rechts-unten ein/aus
`mx`	keine	Schaltet alle LED-Matrizen ein/aus
`px`	`x#y#x#x#x#` `x123 y9 {0x12, 0x34, 0x56}`	Einem Pixel an den eingegebenen x- und y-Koordinaten wird der eingegebene rote, grüne und blaue Farbwert übergeben
`t1`	keine	Testbild, welches das Display vollflächig einfärbt
`t2`	keine	Testbild, welches einzelne horizontale Linien erzeugt
`t3`	keine	Testbild, welches nur die Pixel am Rand des Displays einfärbt
`t4`	keine	Testbild, welches einzelne diagonale Linien erzeugt
`t5`	keine	Testbild, welches einen vollflächigen Farbverlauf auf dem Display erzeugt

Tabelle 8: Liste und Kurzbeschreibung der Befehle über die serielle Kommunikationsschnittstelle

4.3.1 LED-Matrix ein-/ausschalten (m1, m2, m3, m4 und mx)

Die Befehle `m#` und `mx` beeinflussen lediglich den Zustand der GPIO-Pins, welche eingeschaltet die Stromversorgung der LED-Matrizen erlauben.

Leider führt das Kappen der Stromversorgung einer beliebigen LED-Matrix dazu, dass auch alle anderen Matrizen nicht mehr leuchten. Der Grund für dieses Problem wurde im Rahmen der Bachelorarbeit nicht herausgefunden. Mit einem Oszilloskop wurde nachgewiesen, dass die restlichen Matrizen weiterhin mit Strom versorgt werden und die Clocksignale in Ordnung sind.

4.3.2 Übertragung eines neuen Bildes (im)

Nach Erhalt des Befehls `im` mit einem Leerzeichen dahinter wird als Parameter ein eindimensionales Array mit Farbdaten erwartet. Das Array soll mit dem Pixel in der linken-oberen Ecke beginnen, welches den Index [0][0] im Anzeigepuffer-Array hat. Von dort aus geht es zuerst von links nach rechts und danach von oben nach unten weiter. Da dieses Array wohl kaum mit der Hand geschrieben wird, wurde darauf geachtet, dass die Schreibweise eines C-Arrays mit hexadezimalen Farbwerten angenommen wird. Ein Beispiel, wie dieser Befehl korrekt verwendet werden soll, um den Rot-, Grün- und Blauwert des ausschließlich ersten Pixels (links-oben) zu bestimmen, wäre `im {0x12, 0x34, 0x56}`. Tatsächlich wird aber nur auf das Zeichen `x`, welches den Anfang einer hexadezimalen Zahl markiert, gehorcht, wodurch die Eingabe `im x12x34x56` dasselbe erzielen würde. Da bei fehlenden Eingaben von Nullwerten ausgegangen wird, kann mit der Eingabe `im xxx` ein ganz dunkles erstes Pixel gesetzt werden. Jedes nicht-hexadezimale Zeichen außer `x`, hinter dem Farbwert wird ignoriert. Somit werden die in einem C-Array üblicherweise vorhandenen Kommas, Leerzeichen, Klammern und der `0` vor dem `x` nicht berücksichtigt.

Das Ende des Befehls wird mit jedem beliebigen nicht-hexadezimalen Zeichen nach dem blauen Farbwert des letzten Pixels gesetzt. Darauffolgende Zeichen werden bis zu einem terminierenden Zeichen ignoriert. Falls frühzeitig ein terminierendes Zeichen übertragen wird, wird der Befehl abgebrochen. Der Mikrocontroller berichtet über die serielle Kommunikationsschnittstelle entsprechend über den Erfolg oder Abbruch der Befehlsausführung. Bei einer unvollständigen Übertragung wird nur ein Teil des übertragenen Bildes, zusammen mit dem Rest des alten Bildes, angezeigt, da die einzelnen Zeichen sofort als Farbwerte in das Anzeigepuffer-Array gespeichert werden, damit kein zusätzlicher Arbeitsspeicherplatz benötigt wird.

Die Ausführung dieses Befehls, egal ob erfolgreich oder abgebrochen, stoppt die Diashow, welche mit dem Befehl `ds` wieder gestartet werden kann.

Die Abarbeitungszeit dieses Befehls variiert je nach Anzahl der Zeichen im Parameter. In einem Array, das mit dot2pic, einer Webapplikation zur Umwandlung einer Bilddatei in ein C-Array, welches in Kapitel 6.1 noch vorgestellt wird, generiert wurde, befinden sich für das 320 × 240 Pixel große Display exakt 1152001 Zeichen. Die Formel zur Berechnung der Abarbeitungszeit ist

$$\text{Zeichen} \times (\text{Startbit} + \text{Datenbits} + \text{Paritätsbits} + \text{Stoppbits}) \div \text{Baudrate} .$$

Mit einer Baudrate von 25000 und dem Format 8N1 beträgt die Abarbeitungszeit

$$1152001 \text{ Zeichen} \times (1 + 8 + 0 + 1) \text{ Bit} \div 25000 \text{ Baud}$$
$$= 460{,}8 \text{ Sekunden} = 7 \text{ Minuten und } 40{,}8 \text{ Sekunden} .$$

4.3.3 Setzen eines einzelnen Pixels (px)

Nach Erhalt des Befehls `px` mit einem Leerzeichen dahinter, werden der Reihe nach Parameter für eine x- und y-Koordinate, sowie dem hexadezimalen roten, grünen und blauen Farbwert übergeben. Die Koordinatenangaben bestimmen das Pixel, welches verändert werden soll. Das Pixel in der linken-oberen Ecke hat die Koordinaten (0,0). Das Pixel in der rechten-unteren Ecke hat die maximal möglichen Koordinaten. Bei einer Displayauflösung von 320 × 240 sind die maximal möglichen Koordinaten (319,239). Der Befehl wird nicht ausgeführt, wenn die x- oder y-Koordinate außerhalb des Maximums liegt. Die Koordinatenangaben entsprechen dem zweidimensionalen Index vom Anzeigepuffer-Array.

Die Farbwerte können als C-Array übergeben werden. Es herrschen dieselben Regeln wie beim `im`-Befehl, sodass das `x` den Anfang des hexadezimalen Farbwertes markiert. Die Eingabe `px x123 y9 {0x12, 0x34, 0x56}` bewirkt somit dasselbe, wie `px x123y9x12x34x56`. Bei allen Parametern wird bei fehlenden Eingaben von Nullwerten ausgegangen, wodurch die Eingabe `px xyxxx` das Pixel mit den Koordinaten (0,0) ganz dunkel machen würde.

Das Ende des Befehls wird mit jedem beliebigen nicht-hexadezimalen Zeichen nach dem blauen Farbwert gesetzt. Darauffolgende Zeichen werden bis zu einem terminierenden Zeichen ignoriert. Falls frühzeitig ein terminierendes Zeichen übertragen wird, wird der Befehl abgebrochen und das Pixel bleibt unverändert. Der Mikrocontroller berichtet über die serielle Kommunikationsschnittstelle entsprechend über den Erfolg oder Misserfolg der Befehlsausführung.

Die Ausführung dieses Befehls stoppt die Diashow, welche mit dem Befehl `ds` wieder gestartet werden kann.

4.3.4 Anzeige von Testbildern (t1, t2, t3, t4, t5)

Die Befehle `t#` geben verschiedene Testbilder am Display wieder. Diese entsprechen den Tests in Kapitel 5.1-5.5. Die mehrfache Ausführung desselben Befehls schaltet zwischen 8 verschiedenen Farbmodi um. Diese wären der Reihe nach weiß/RGB, gelb/RG, pink/RB, rot/R, türkis/GB, grün/G, blau/B und schwarz/nichts.

Die Ausführung dieser Befehle stoppt die Diashow. Mit dem Befehl `ds` kann eine Diashow aller Testbilder im aktuellen Farbmodus gestartet werden.

Der Befehl `t1` lässt alle Pixel mit voller Leuchtkraft leuchten und erzeugt somit ein vollflächig einfarbig beleuchtetes Display. Siehe Kapitel 5.1 für den Nutzen dieses Tests.

Der Befehl `t2` lässt alle Pixel in jeder neunten Zeile leuchten. Siehe Kapitel 5.2 für den Nutzen dieses Tests.

Der Befehl `t3` lässt nur die ersten und letzten Zeilen und Spalten der einzelnen LED-Matrizen leuchten, sodass nur die Pixel am Rand der LED-Matrizen leuchten. Siehe Kapitel 5.3 für den Nutzen dieses Tests.

Der Befehl `t4` erzeugt diagonale Linien auf dem Display, welche jeweils 18 Pixel voneinander entfernt sind. Siehe Kapitel 5.4 für den Nutzen dieses Tests.

Der Befehl `t5` erzeugt einen vollflächigen Farbverlauf von der linken bis zur rechten Seite des Displays. Dabei ist die Farbe auf der einen Seite das Negative von der gegenüberliegenden Seite, was dem höchsten Kontrast entspricht. Siehe Kapitel 5.5 für den Nutzen dieses Tests.

4.3.5 Anzeige eines bestimmten Demobildes (d1, d2, ..., d9)

Mit dem Befehl `d#` (für `#` kann eine beliebige Zahl zwischen 1 und 9 gewählt werden) kann ein bestimmtes Demobild, welches zusammen mit dem Programm im Flashspeicher des Mikrocontrollers gespeichert ist, angezeigt werden.

Die Ausführung dieses Befehls stoppt die Diashow, welche mit dem Befehl `ds` wieder gestartet werden kann.

4.3.6 Starten/Stoppen der Diashow (ds)

Mit dem Befehl `ds` kann die Diashow angehalten oder weitergeführt werden. Diese läuft in Endlosschleife, sodass nach dem letzten Bild wieder mit dem ersten Bild fortgefahren wird.

4.3.7 Ungültige Benutzereingaben

Eine ungültige Benutzereingabe wird vom Mikrocontroller mit einer Nachricht beantwortet, die von dem ungültigen oder abgebrochenen Befehl berichtet. Erst nach Eingang eines terminierenden Zeichens wird der Empfangspuffers auf Befehle überprüft. Der Empfangspuffer beinhaltet bis zu fünf Zeichen, sodass der längste Befehl `help` und dahinter ein terminierendes Zeichen gespeichert werden können. Befindet sich kein terminierendes Zeichen hinter dem Befehl, ist der Befehl ungültig. Wenn sich unter den ersten fünf Zeichen sowieso kein terminierendes Zeichen befindet, ist der Befehl ungültig und der Empfangspuffer wird gar nicht erst auf seine Gültigkeit geprüft.

Nach dem Erhalt eines Leerzeichens nach der Eingabe von `px` oder `im`, geht der Mikrocontroller in einen Zustand über, in dem er Parameter erwartet. Solange sich der Mikrocontroller in diesem Zustand befindet, wird stattdessen nach jedem einzelnen empfangenen Zeichen der Empfangspuffer überprüft. Dieser Zustand wird mit einem terminierenden Zeichen beendet, beziehungsweise abgebrochen.

Im Rahmen der Bachelorarbeit wurde kein unerwartetes Verhalten bei Eingang verschiedener ungültiger Benutzereingaben festgestellt. Der Mikrocontroller sendet für jede ungültige als auch gültige Benutzereingabe über die serielle Kommunikationsschnittstelle eine entsprechende Antwort.

5 Test des high-density LED-Matrix Treibers

In diesem Kapitel werden einige Tests erklärt, um sicher zu gehen, dass die Ansteuerung der LED-Matrix erwartungsgemäß funktioniert.

Bestenfalls wird ein zweidimensionales Anzeigepuffer-Array angelegt, welches Farbdaten für jeweils rot, grün und blau eines jeden Pixels der LED-Matrix beinhaltet. Da die interne PWM-Ausgabe der LED-Matrix eine Anzahl von 2^{16} Abstufungen anbietet, werden die einzelnen Farbwerte im Anzeigepuffer-Array auch als vorzeichenlose 16

Bit Ganzzahlen abgespeichert. Natürlich kann aus Speicher- oder Performancegründen auch eine geringere Anzahl von Bits verwendet werden.

Die Unterkapitel dieses Kapitels fangen mit dem einfachsten Test, die LED-Matrix anzusteuern, an und arbeiten sich in hierarchischen Schritten bis zum Aufleuchten einzelner LEDs, sowie der Anzeige eines dynamischen Bildes hin.

5.1 Ganzflächiges ausfüllen des Displays mit einer Farbe

Der einfachste Test ist, alle LEDs der LED-Matrix leuchten zu lassen. Dazu reicht es die Pins **R#**, **G#** und **B#** aller Reihen für die gesamte Zeit HIGH zu setzen. Siehe Kapitel 2.2 Kommunikationsschnittstelle und Tabelle 2 für die Bedeutung dieser Pins.

Wenn alle Pixel weiß sind und dieselbe Helligkeit haben, ist der Test erfolgreich. Das bedeutet, dass das Umschalten der Scan Line korrekt funktioniert.

Mit einem Strommessgerät kann nun der maximal benötigte Strom gemessen werden, die die LED-Matrix braucht, wenn die gesamte Zeit über die maximale Anzahl an LEDs zur selben Zeit am hellsten leuchten. Das ist wichtig für die Bestimmung der Bauteile auf der Leiterplatte, an der die LED-Matrix angeschlossen wird. Der Stromverstärkungsfaktor, siehe Tabelle 4, kann gegebenenfalls auch erhöht werden.

Wenn die Pins **R#**, **G#** oder **B#** pro Reihe einzeln HIGH gesetzt werden, während die anderen Pins LOW gesetzt bleiben, um nur alle rote, grüne oder blaue LEDs in einer Reihe leuchten zu lassen, kann geprüft werden, ob alle Pins der Kommunikationsschnittstelle richtig kontaktiert wurden.

Dieser Test kann über die serielle Kommunikationsschnittstelle mit dem Befehl `t1` angezeigt werden.

5.2 Horizontale Linien über das komplette Display leuchten lassen

Im zweiten Test, wird Gebrauch von dem zweidimensionalen Anzeigepuffer-Array gemacht, welches das Bild für das Display beinhaltet. Auf dem Display soll beispielsweise jede vierte Zeile leuchten.

Wenn sich genau das widerspiegelt, was im Anzeigepuffer-Array hinterlegt wurde, ist der Test erfolgreich.

Falls sich stattdessen die Zeilen nach oben oder unten bewegen, sind die Timings der **GCLK** oder **DCLK** falsch; möglicherweise sogar beides. Das kann viele verschiedene

Gründe haben, weswegen nochmal die Datenblätter für die LED-Matrix gründlich durchgegangen werden sollten.

Dieser Test kann über die serielle Kommunikationsschnittstelle mit dem Befehl `t2` angezeigt werden.

5.3 Nur die äußersten Pixel vom Display leuchten lassen

Nun werden die Farbdaten im Anzeigepuffer-Array so hinterlegt, sodass die erste und letzte Zeile und Spalte, also nur die LEDs am Rand der LED-Matrix, leuchten.

Wenn dies der Fall ist, ist der Test erfolgreich. Die Dimensionierung des Anzeigepuffer-Arrays stimmt mit den Dimensionen des Displays überein und die Farbdaten werden korrekt in das Schieberegister geschoben.

Dieser Test kann über die serielle Kommunikationsschnittstelle mit dem Befehl `t3` angezeigt werden.

5.4 Diagonale Linien auf dem Display leuchten lassen

Mit einem Test, bei dem die Farbdaten im Anzeigepuffer-Array so hinterlegt werden, dass beispielsweise eine diagonale Linie auf dem Display erscheint, kann unter anderem betrachtet werden, ob die Farbdaten möglicherweise verkehrt herum in die Schieberegister der LED-Matrix abgelegt werden.

Falls das Bild zum erwarteten Ergebnis tatsächlich spiegelverkehrt auftritt, muss etwas an der Abarbeitung des Anzeigepuffer-Arrays mit den abgespeichert Farbdaten umgekehrt werden.

Dieser Test kann über die serielle Kommunikationsschnittstelle mit dem Befehl `t4` angezeigt werden.

5.5 Unterschiedliche Helligkeit der LEDs

Angenommen, es wurden bisher nur Farbwerte in das Anzeigepuffer-Array abgespeichert, welche die LED entweder immer maximal hell oder überhaupt nicht leuchten lassen. Es wäre nun an der Zeit die PWM-Treiber der LED-Matrix zu nutzen. Dazu können die Farbdaten im Array so angelegt werden, sodass beispielsweise von einer Seite des Displays bis zur Anderen ein Farbverlauf entsteht.

Falls sich das Ergebnis vom Erwarteten unterscheidet, werden im Programmcode womöglich die einzelnen Bits der Farbdaten falsch gezählt.

Dieser Test kann über die serielle Kommunikationsschnittstelle mit dem Befehl `t5` angezeigt werden.

5.6 Änderungen am Bild während des Programmablaufs

Falls alle vorangegangenen Tests erfolgreich durchgeführt wurden, sollte es kein Problem sein, während des Programmablaufs die Farbdaten im Anzeigepuffer-Array, beispielsweise per Hand während des Debuggings oder mit implementierten Befehlen über eine Kommunikationsschnittstelle, abzuändern, um somit das angezeigte Bild auf der LED-Matrix zu verändern. Dies stellt die Möglichkeit dar, ein interaktives Menü, ein Bewegtbild oder, wie in Kapitel 6.2 näher beschrieben, eine Diashow auf der LED-Matrix wiederzugeben.

5.7 Fazit der Tests

Nach dem Erfolg aller vorangegangen Tests, sollte es nun möglich sein, einzelne Pixel mit der richtigen Farbe und Helligkeit auf der LED-Matrix anzeigen zu lassen. Obwohl die Ansteuerung der LED-Matrix an diesem Punkt grundsätzlich und möglichst effizient funktionieren dürfte, können trotzdem noch viele weitere Testfälle überlegt werden, an denen im Rahmen dieses Projekts nicht gedacht wurde.

6 Beispielanwendung zur Benutzung der high-density LED-Matrix

In diesem Kapitel werden zwei Anwendungen vorgestellt, die zur Demonstration der LED-Matrix durchgeführt wurden. In gewisser Hinsicht waren diese Beispielanwendungen komplexere Tests an die LED-Matrix, als die, die in Kapitel 5 erwähnt wurden.

6.1 Anzeige eines statischen Bildes

Die erste Beispielanwendung war, ein beliebiges statisches Bild auf der LED-Matrix anzuzeigen. Hierfür wurde noch eine einzelne LED-Matrix mit dem „fliegenden“ Aufbau, wie dem in Abbildung 9, benutzt. Abbildung 10 zeigt die erste erfolgreiche Anzeige eines statischen Bildes.

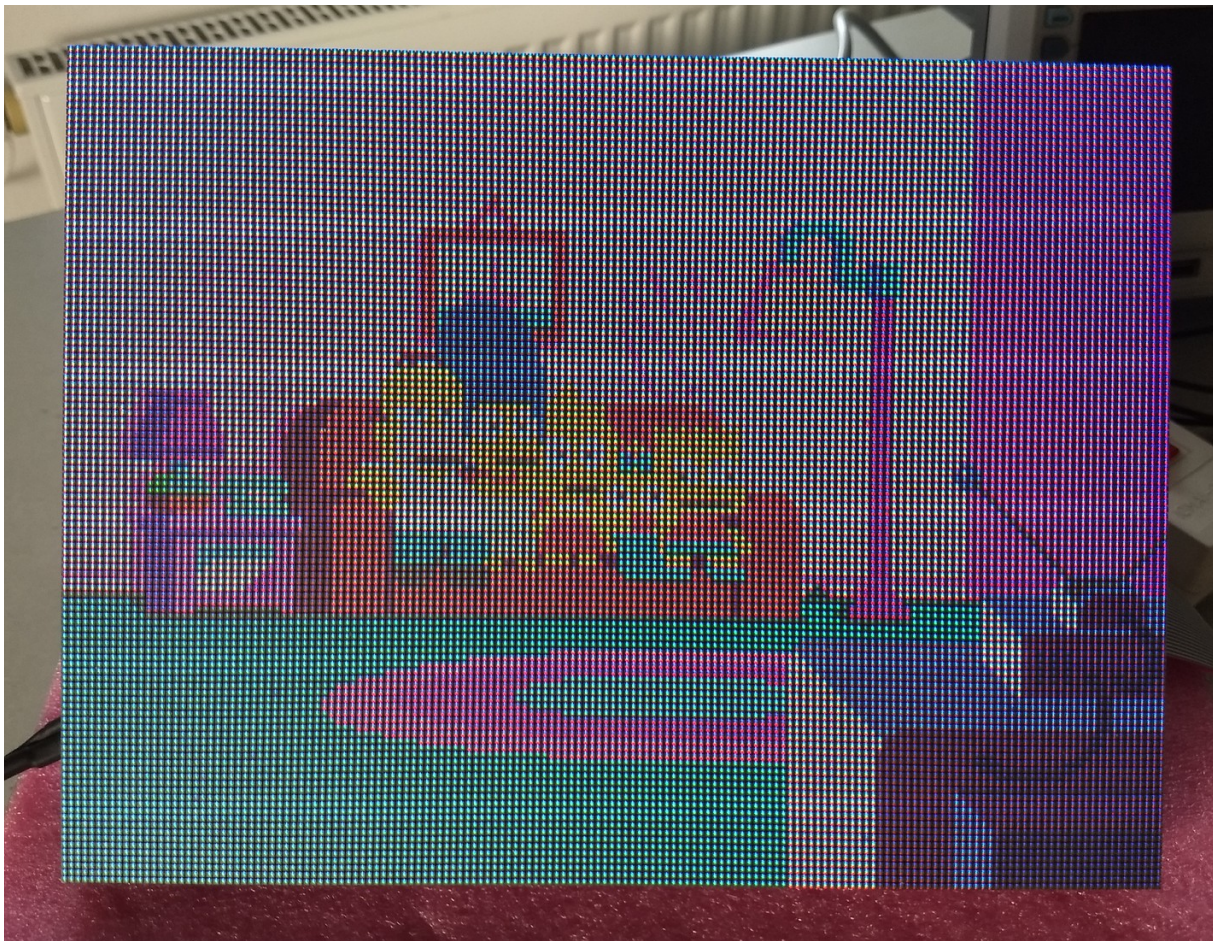

Abbildung 10: 160 × 120 Pixel großes statisches Bild eines selbstgemachten Pixel-Arts der beliebten TV-Serie „The Simpsons“ (Vorlage dazu war [33]) auf der LED-Matrix.

Das aktuelle Bild für das finale Display wird in einem 320 × 240 × 3 (320 LEDs in der Breite und 240 LEDs in der Höhe mit jeweils den Rot-, Grün- und Blauwerten) großen Anzeigepuffer-Array im Arbeitsspeicher des Mikrocontrollers hinterlegt.

Statt sich die Mühe zu machen, das Programm mit einer Funktion auszustatten, welches eine Bilddatei in ein C-Array umwandelt, wurden stattdessen zwei Webapplikationen evaluiert, die genau das machen.

Dot2pic: Eine dieser Webapplikationen nennt sich dot2pic [34]. Dot2pic erwartet als Eingang eine BMP-Datei, was ein geläufiges Format für Bilddateien ist. Dieses wandelt dot2pic in ein C-Array mit bis zu 3 Byte pro Pixel und 8 Bit pro Farbe um.

Wird die dot2pic-Webseite aufgerufen (Online verfügbar über http://dot2pic.com/) soll als erstes über Schieberegler, die Auflösung des Bildes bestimmt werden. Eigentlich wären die Schieberegler mit einem Maximum von 256 Pixel auf jeder Achse zu klein für das 320 Pixel breite Display. Allerdings können über die Browser-Entwicklungstools, die beispielsweise der beliebte Browser „Google Chrome“ anbietet, das

Maximum dieses Schiebereglers auf einen beliebigen Wert verändert werden. Einen Schritt weiter kann dann eine BMP-Datei ausgewählt werden. Zusätzlich bietet dot2pic dem Benutzer die Möglichkeit, das Bild im Pixel-Art-Stil zu bemalen. Wenn alles bereit ist, kann das Bild in ein C-Array umgewandelt werden. Das generierte C-Array soll als eindimensionales, konstantes Array mit konstanten Werten (`const int const arrayname[]`) in einer C-Header-Datei gespeichert werden, damit das Programm das Anzeigepuffer-Array im Arbeitsspeicher des Mikrocontrollers mit diesen Daten befüllen kann.

PicturetoC_Hex_converter.php: Die andere Webapplikation nennt sich PicturetoC_Hex_converter.php von Digole Digital Solutions [35]. Diese erwartet als Eingang eine Bilddatei im GIF-, JPG-/JPEG- oder PNG-Format in einer Auflösung von maximal 320 Pixeln in jeder Achse. Anders wie bei dot2pic, wird hier serverseitig geprüft, ob die Maximalauflösung eingehalten wird. Ausgegeben wird ein C-Array, mit bis zu 3 Byte pro Pixel und 6 Bit pro Farbe im Format 00XXXXXX, 0XXXXXX0 oder XXXXXX00. Im Vergleich zu dot2pic ist die Farbtiefe geringer und vom Verarbeitungsaufwand umständlicher. Eine erwähnenswerte Eigenschaft dieser Webapplikation ist allerdings die Funktion, das Bild in eine andere Auflösung auszugeben. Wird beispielsweise eine doppelt so hohe Ausgabeauflösung, als eingangs mitgegeben, angegeben, bleiben die Kanten scharf, beziehungsweise die Farben werden nicht gemischt.

Da das Anzeigepuffer-Array im Arbeitsspeicher des Mikrocontrollers liegt, kann dieses mühelos durch das Programm verändert werden, wodurch beispielsweise auch ein Bewegtbild möglich wäre. Deswegen wurde die Möglichkeit implementiert, über die serielle Kommunikationsschnittstelle ein neues Bild, das am Display angezeigt werden soll, zu senden.

6.2 Anzeige von mehreren statischen Bildern als Diashow

Die zweite Beispielanwendung erweitert die erste Beispielanwendung zu einer Diashow. Es werden mehrere Demonstrationsbilder als Teil des Programms im Flashspeicher des Mikrocontrollers hinterlegt. Wie bei einer Diashow werden die angezeigten Bilder auf dem Display periodisch nach einer definierten Zeitspanne gewechselt. Die Diashow läuft in einer Endlosschleife, sodass nach dem letzten Bild wieder das Erste angezeigt wird.

Über die serielle Kommunikationsschnittstelle ermöglichen bestimmte Befehle, dass die Diashow angehalten oder weitergeführt werden kann (siehe Kapitel 4.3.6) oder auf ein bestimmtes Bild aus der Diashow gewechselt wird (siehe Kapitel 4.3.5).

7 Evaluierung der high-density LED-Matrix Ansteuerung

Ohne Vorwissen ist die Ansteuerung einer LED-Matrix eine sehr große Herausforderung. Damit die LED-Matrix erst einmal etwas anzeigt, muss von Grund auf das Timing aller Clocks und die Konfiguration der LED-Matrix korrekt angepasst sein. Das macht das Debuggen sehr mühselig bis hin zu unnütz, solange bereits zum Programmstart die falschen Timing- und Konfigurationswerte benutzt werden. Oft hat es sich bewährt, Variablen zu verwenden, um während des Programmablaufs über den Debug-Tools die Timings oder sonstige Werte zu verändern.

Trotzdem ist die Technik hinter der LED-Matrix sehr interessant und die Funktionsweise praktisch umgesetzt.

8 Fazit und Zusammenfassung

8.1 Stand zum Ende des Projekts

Das Ziel, die LED-Matrix anzusteuern, wurde erreicht. Es kann zum Programmstart ein vorgefertigtes Bild angezeigt werden, als auch während des Programmablaufs über die serielle Kommunikationsschnittstelle RS485 unter anderem ein neues Bild in Form eines C-Arrays übermittelt, sowie einzelne Pixel verändert werden.

Auch hat das Display eine ansehnliche Konstruktion. Vier LED-Matrizen wurden so befestigt, dass diese zusammen ein 40 × 30 cm großes Display mit einer Auflösung von 320 × 240 RGB-LEDs ergeben. Die Rückseite der LED-Matrizen werden mit einer entsprechend großen, speziell für dieses Projekt entwickelten Leiterplatte bedeckt. Mit zwei mittels Schrauben an der Leiterplatte befestigten Kunststoffbeinen kann das Display auf eine ebene Unterlage, mit einer festen Neigung von 70° zur Unterlage, stehen.

Eine Demonstration des Ergebnisses dieser Bachelorarbeit gibt es Online als Video unter folgendem Link: youtu.be/PVObh-cLf8k

8.2 Ausblick

Die Vision, dass Statusinformationen von Drittsystemen auf dem Display dargestellt werden, ist mit den vorhandenen USB- und Ethernet-Anschlüssen möglich. Im Rahmen der Bachelorarbeit wurden diese Schnittstellen nicht in das Programm auf dem Mikrocontroller implementiert.

Für diese Vision wäre es außerdem nützlich, eine Schrift und einen weiteren Befehl für die serielle Kommunikationsschnittstelle zu implementieren, das als Parameter einen beliebig langen Text annimmt, um diesen daraufhin am Display anzuzeigen.

Die in Kapitel 3.4 erwähnten bekannten Fehler am Leiterplattenentwurf wurden korrigiert, um bei Gelegenheit einen weiteren Prototyp der Leiterplatte produzieren zu lassen.

Wegen dem im Kapitel 3.4.1 erwähnten Fehler war es nicht möglich, eine funktionierende Bedienung zu haben. Deswegen wurde die Schnittstelle für die beiden SNES-Controller ebenfalls nicht in das Mikrocontroller-Programm implementiert.

Es wäre eventuell sinnvoll, ein Echtzeitbetriebssystem wie beispielsweise das von Renesas empfohlene FreeRTOS einzusetzen und die Befehle in Tasks statt in mehreren ISR auszuführen. Dadurch könnten sich Timingprobleme lösen, indem Tasks mit derselben Priorität mit einem Multitasking-Verfahren abgearbeitet werden, statt immer komplett ausgeführt zu werden.

Da die ISR zum Senden und Empfangen über die serielle Kommunikationsschnittstelle gelegentlich die ISR für die **DCLK** blockieren kann, wäre es vermutlich besser, den sogenannten „DMA"-Controller („direct memory access") vom RX65N-Mikrocontroller einzusetzen. Dieser würde bewirken, dass die ankommenden Zeichen ohne Interrupt oder Verwendung des Prozessors in den Speicher abgelegt werden und, weil auch nicht mehr auf den Abschluss einer ISR gewartet werden müsste, dadurch auch die Baudrate erhöht werden könnte.

Literaturverzeichnis

[1] Paul Horowitz & Winfield Hill: *The Art of Electronics - Third Edition*. Cambridge University Press, New York, 16. Auflage, 2020.

[2] Alibaba.com: *Yao Caixing P1.25 Rgb indoor smd1010 matrix 160x120 pixel led-panel hohe reslutions display modul* [Online]. Verfügbar: https://german.alibaba.com/product-detail/yao-caixing-p1-25-rgb-indoor-smd1010-matrix-160x120-pixels-led-panel-high-reslutions-display-module-60831436294.html [Zugriff am 11.02.2021]

[3] Wikipedia: *Internationales Einheitensystem. SI-Einheiten*. [Online]. Verfügbar: https://de.wikipedia.org/wiki/Internationales_Einheitensystem#SI-Einheiten [Zugriff am 17.02.2021]

[4] Big Mess o' Wires: *64 x 32 LED Matrix Programming*. [Online]. Verfügbar: https://www.bigmessowires.com/2018/05/24/64-x-32-led-matrix-programming/ [Zugriff am 04.10.2020]

[5] Macroblock, Inc.: *MBI5050 Application Note*. Verfügbar: Anhang [A] oder auf Anfrage bei Macroblock [Stand: 2010, V1.00]

[6] Macroblock, Inc.: *Preliminary Datasheet MBI5153*. Verfügbar: Anhang [B] oder auf Anfrage bei Macroblock [Stand: Mai 2014, V1.00]

[7] Nick Poole: Ev*erything You Didn't Want to Know About RGB Matrix Panels*. [Online]. Verfügbar: https://www.sparkfun.com/news/2650 [Zugriff am 04.10.2020]

[8] Wikipedia: *Interferenz (Physik)*. [Online]. Verfügbar: https://de.wikipedia.org/wiki/Interferenz_(Physik) [Zugriff am 24.12.2020]

[9] Elektronik-Kompendium.de: *Schieberegister*. [Online]. Verfügbar: http://www.elektronik-kompendium.de/sites/dig/0210211.htm [Zugriff am 24.12.2020]

[10] Elektronik-Kompendium.de: *PWM - Pulsweitenmodulation*. [Online]. Verfügbar: https://www.elektronik-kompendium.de/sites/kom/0401111.htm [Zugriff am 24.12.2020]

[11] Elektronik-Kompendium.de: *Integrierte Schaltungen (IC)*. [Online]. Verfügbar: http://www.elektronik-kompendium.de/sites/bau/0206091.htm [Zugriff am 24.12.2020]

[12] Macroblock, Inc.: *MBI5152 Application Note*. Verfügbar: Anhang [C] oder auf Anfrage bei Macroblock [Stand: 2013, V1.00]

[13] Wikipedia: *Signalflanke*. [Online]. Verfügbar: https://de.wikipedia.org/wiki/Signalflanke [Zugriff am 24.12.2020]

[14] IEEE: *About IEEE*. [Online]. Verfügbar: https://www.ieee.org/about/index.html?utm_source=dhtml_footer&utm_medium=hp&utm_campaign=learn-more [Zugriff am 26.12.2020]

[15] Wikipedia: *USB*. [Online]. Verfügbar: https://en.wikipedia.org/wiki/USB [Zugriff am 26.12.2020]

[16] DATACOM Buchverlag GmbH: *RJ45-Stecker*. [Online]. Verfügbar: https://www.itwissen.info/RJ45-Stecker-RJ45-male-connector-RJ45.html [Zugriff am 26.12.2020]

[17] Olga Weis: *Was ist RS485? RS485 Kommunikationsanleitung*. [Online]. Verfügbar: https://www.eltima.com/de/article/rs485-communication-guide/ [Zugriff am 26.12.2020]

[18] reicheltpedia: *Crimp Anleitung*. [Online]. Verfügbar: https://www.reichelt.de/reicheltpedia/index.php5/Crimp_Anleitungen [Zugriff am 26.12.2020]

[19] Nintendo: *Super Nintendo*. [Online]. Verfügbar: https://www.nintendo.de/Unternehmen/Unternehmensgeschichte/Super-Nintendo/Super-Nintendo-627040.html [Zugriff am 25.12.2020]

[20] mikrocontroller: *UART*. [Online]. Verfügbar: https://www.mikrocontroller.net/articles/UART [Zugriff am 26.12.2020]

[21] Roland Mechling: *GK Informatik. 5.1 Beispiele für Kommunikation im Alltag*. [Online]. Verfügbar: http://www.gk-informatik.de/netze/protocol.html [Zugriff am 26.12.2020]

[22] Ragnar Tessloff GmbH & Co. KG: *Was sagt die Hertz-Zahl aus?* [Online]. Verfügbar: https://www.wasistwas.de/archiv-technik-details/was-sagt-die-hertz-zahl-aus.html [Zugriff am 26.12.2020]

[23] KSB SE & Co. KGaA: *Galvanische Trennung*. [Online]. Verfügbar: https://www.ksb.com/kreiselpumpenlexikon/galvanische-trennung/187162 [Zugriff am 29.12.2020]

[24] Schraube & Mutter: *Schrauben M3*. [Online]. Verfügbar: https://schraube-mutter.de/schrauben/m3/ [Zugriff am 1.2.2021]

[25] USB Implementers Forum, Inc.: *Defined Class Codes*. [Online]. Verfügbar: https://www.usb.org/defined-class-codes [Zugriff am 29.12.2020]

[26] Elektronik Praxis: *PCB-Design-Regeln: Sieben Sünden beim Leiterplatten-Design*. [Online]. Verfügbar: https://www.elektronikpraxis.vogel.de/pcb-design-regeln-sieben-suenden-beim-leiterplatten-design-a-356703/ [Zugriff am 02.02.2020]

[27] mikrocontroller: *Interrupt*. [Online]. Verfügbar: https://www.mikrocontroller.net/articles/Interrupt [Zugriff am 10.3.2021]

[28] Renesas: *RX65N Group, RX651 Group Datasheet*. [Online]. Verfügbar: https://www.renesas.com/us/en/document/dst/rx65n-group-rx651-group-datasheet [Stand: 20.6.2019, R01DS0276EJ0230 Rev.2.30]

[29] Essemtec AG Schweiz: *FOX SMD Pick & Place System*. [Online]. Verfügbar: https://essemtec.com/fileadmin/user_upload/produkt/pdf/DE/FOX_Spezifikation_DE_V3.1.pdf [Zugriff am 17.02.2021]

[30] Jim Christy: *Super Nintendo – Pinouts & Protocol FAQ*. [Online]. Verfügbar: https://gamefaqs.gamespot.com/snes/916396-super-nintendo/faqs/5395 [Zugriff am 29.12.2020]

[31] Technopedia Inc.: *Pixel Art*. [Online]. Verfügbar: https://www.techopedia.com/definition/8884/pixel-art [Zugriff am 31.12.2020]

[32] G. Brechmann, W. Dzieia, E. Hörnemann, H. Hübscher, D. Jagla, H. Petersen: *Elektrotechnik Tabellen Kommunikationselektronik*. Seite 276 & 277. Westermann Schulbuchverlag GmbH, Braunschweig, 3. Aufl., 2002.

[33] pixel: *SIMPSONS PIXELS*. [Online]. Verfügbar: https://youtu.be/FIZ_gDOrzGk [Zugriff am 31.12.2020]

[34] psymaster@o2.pl: *dot2pic*. [Online]. Verfügbar: http://dot2pic.com/ [Zugriff am 31.12.2020]

[35] Digole Digital Solutions: *PicturetoC_Hex_converter.php*. [Online]. Verfügbar: http://digole.com/tools/PicturetoC_Hex_converter.php [Zugriff am 31.12.2020]

[36] Aspose Pty Ltd: *What is a BMP file?*. [Online]. Verfügbar: https://docs.fileformat.com/image/bmp/ [Zugriff am 1.1.2021]

[37] Aspose Pty Ltd: *What is a GIF file?*. [Online]. Verfügbar: https://docs.fileformat.com/image/gif/ [Zugriff am 1.1.2021]

[38] Aspose Pty Ltd: *What is a JPEG file?*. [Online]. Verfügbar: https://docs.fileformat.com/image/jpeg/ [Zugriff am 1.1.2021]

[39] Aspose Pty Ltd: *What is a PNG file?*. [Online]. Verfügbar: https://docs.fileformat.com/image/png/ [Zugriff am 1.1.2021]

[40] Amazon Web Services: *FreeRTOS Multitasking*. [Online]. Verfügbar: https://www.freertos.org/implementation/a00004.html [Zugriff am 16.02.2021]

[41] LEO Dictionary Team: *Reverse Engineering*. [Online]. Verfügbar: https://dict.leo.org/englisch-deutsch/reverse%20engineering [Zugriff am 17.02.2021]

Anhänge

[A] Macroblock, Inc.: *MBI5050 Application Note*. Stand: 2010, V1.00 [5]

[B] Macroblock, Inc.: *Preliminary Datasheet MBI5153*. Stand: Mai 2014, V1.00 [6]

[C] Macroblock, Inc.: *MBI5152 Application Note*. Stand: 2013, V1.00 [12]