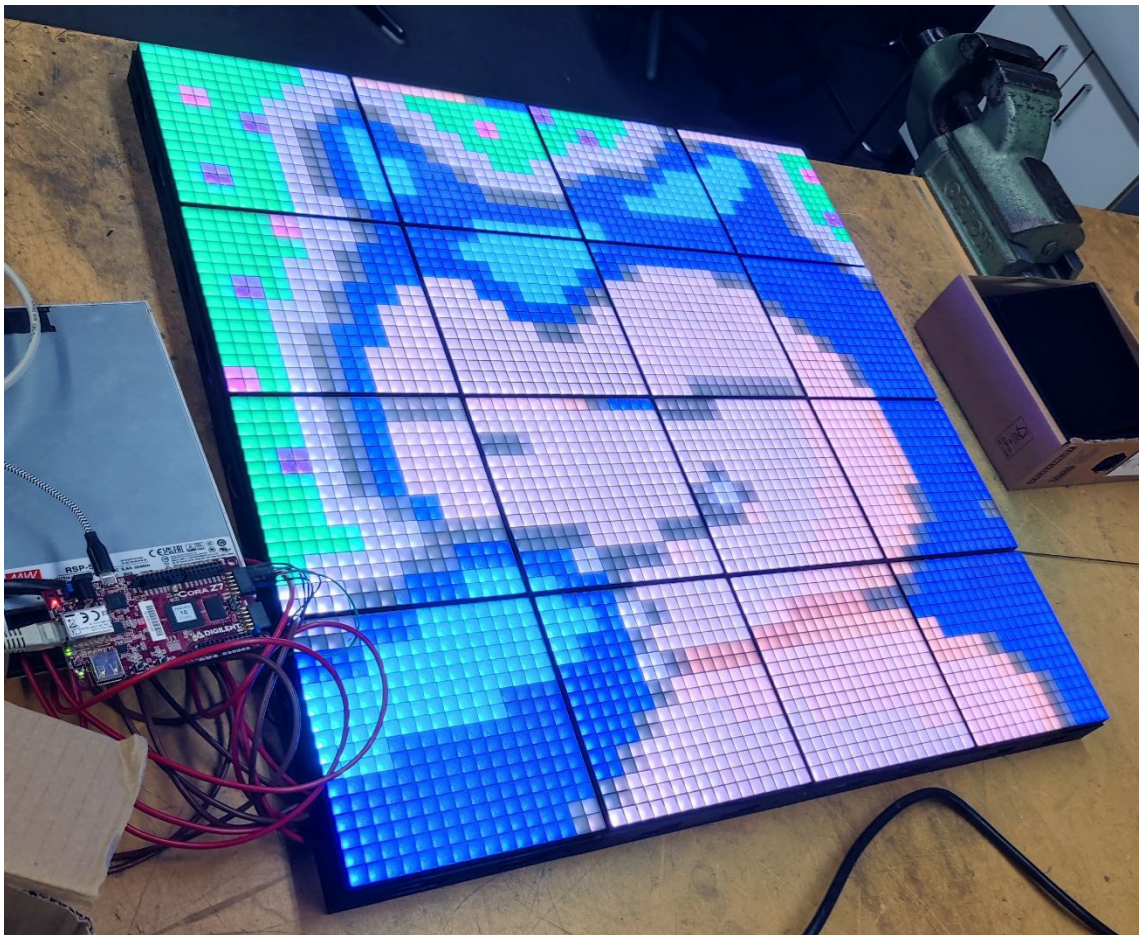


## Bau einer LED-Matrix mit bis zu 64x64 Leds

Nicole Meyer, Markus Schmidhuber, Paul Schlößer



Datum: 09.02.2025

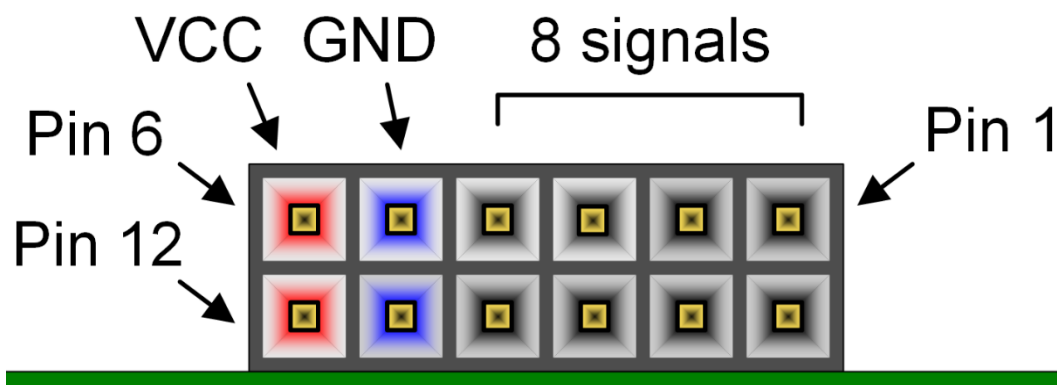
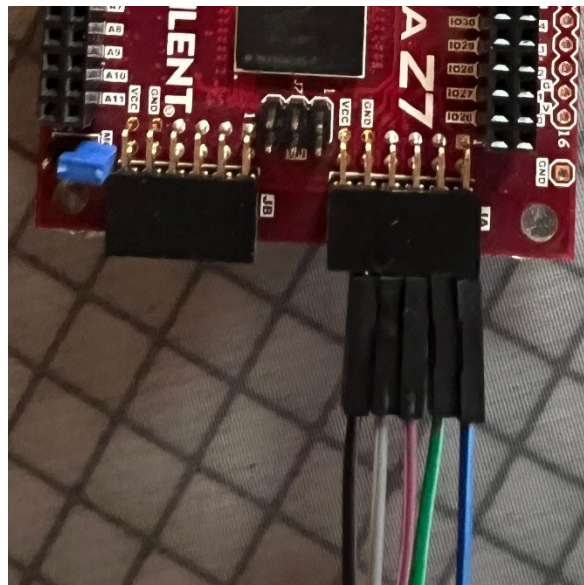
# Inhaltsverzeichnis

Inhaltsverzeichnis .....	2
1 Quickstart-Guide.....	3
2 Einleitung .....	4
3 Informationsbeschaffung.....	5
3.1 Vorangegangenen Projekt.....	5
3.2 TCP als Übertragungsmedium .....	6
4 Entwurf.....	8
4.1 Hierarchie.....	8
4.2 Python.....	8
4.3 C-Programm .....	11
4.4 3D-Druck.....	13
4.5 Fehler und Probleme .....	14
5 Fazit.....	16
6 Abbildungsverzeichnis .....	17

# 1 Quickstart-Guide

Benötigte Komponenten:

- LED-Matrix mit Kabel
- Spannungsquelle
- Computer mit Vitis und Python vorinstalliert
- Ethernet Kabel



Verdrahtet wird das Cora Z7 Board mit der LED-Matrix wie folgt:

- Blau auf Pin1
- Grün auf Pin2
- Lila auf Pin3
- Weiß auf Pin4
- Schwarz auf GND


In Vitis wird das Projekt „LED\_Matix\_variable“ im Ordner „Vitis\_Software“ geöffnet und auf dem Cora Z7 Board ausgeführt. Dazu muss Board mit dem Computer per Ethernet Kabel verbunden werden und der entsprechende Port am Computer bekannt sein.

Das Python Programm „Python\_client\_application“ muss ausgeführt werden, um die entsprechende Auswahl Maske zu erhalten.

## 2 Einleitung

In diesem Projekt wird eine Ansteuerung für eine LED-Matrix mit bis zu 4x4 Panels realisiert. Die Matrix besteht aus 16 einzelnen 16x16 LED-Panels, die zu einem großen 64x64 Raster verbunden werden können. Insgesamt können somit 4096 WS2812B LEDs individuell angesteuert werden. Die Ansteuerung erfolgt über bis zu 4 Datenkanäle, wodurch ein ausreichender Datendurchsatz gewährleistet wird.

Die Darstellung komplexer Muster und Bilder wird durch ein Python-Programm ermöglicht, welches die Verarbeitung und Steuerung der Bilddaten übernimmt. Auf dem Cora-Board läuft FreeRTOS und steuert die LEDs über ein in C implementiertes Programm an. Dieses Programm empfängt die über TCP übertragenen Daten und verarbeitet sie, um sie für die Ausgabe an die LEDs vorzubereiten.

Aufgrund  der großen Anzahl an LEDs musste auch eine stabile und leistungsstarke Stromversorgung sichergestellt werden, um den hohen Energiebedarf der Matrix zu decken. Eine LED ist mit 0,3 Watt angegeben. Das ergibt bei einem Panel eine maximale Leistung von ca. 76 Watt. Bei 4x4 Paneelen steigt der maximal mögliche Bedarf somit auf 1.230 Watt. Diese hohen Werte werden jedoch nur bei Darstellung von weißen Pixeln auf maximaler Helligkeit erreicht. In der Praxis wurde bei der Darstellung eines Bildes jedoch maximal 400 Watt verbraucht. Deshalb ist die Stromversorgung mit 450 Watt ausreichend dimensioniert.

## 3 Informationsbeschaffung

### 3.1 Vorangegangenen Projekt

Dieses Projekt basiert auf dem bereits existierenden Projekt von Benedikt Gareis und baut darauf auf. Es wird der derselbe Hardwareaufbau genutzt, der im Folgenden kurz erläutert wird:

Das Cora Z7-10 Development Board basiert auf dem Zynq-7000 APSoC von Xilinx, der einen leistungsstarken Cortex-A9 Dual Core Prozessor und ein Artix-7 FPGA kombiniert. Es unterstützt die Datenübertragung zwischen Mikrocontroller und FPGA über DMA und nutzt hierfür den DDR3L-Speicher sowie bis zu acht DMA-Kanäle. Die WS2812B LEDs, die in diesem Projekt verwendet werden, sind adressierbare RGB-LEDs mit integriertem Mikrocontroller, die einfach über ein serielles Protokoll gesteuert werden können. Jede LED verarbeitet 24-Bit-Daten in der Reihenfolge Grün, Rot, Blau (GRB) und leitet verbleibende Daten an die nächste LED weiter.

## 3.2 TCP als Übertragungsmedium

Das Transmission Control Protocol (TCP) ist ein verbindungsorientiertes Protokoll, das für die zuverlässige Übertragung von Daten zwischen zwei Geräten im Netzwerk sorgt. Es teilt Daten in kleinere Pakete auf, sendet sie und setzt sie am Zielort wieder zusammen. TCP stellt sicher, dass die Pakete in der richtigen Reihenfolge und ohne Fehler ankommen. Im folgendem werden die wichtigsten Funktionen von TCP aufgeführt:

- **Segmentierung:** Zerlegung der Anwendungsdaten in Pakete zur Übertragung und anschließende Wiederherstellung zu einer Datei oder einem Datenstrom.
- **Verbindungsmanagement:** Aufbau und Abbau der Verbindung zwischen Client und Server.
- **Flusssteuerung:** Dynamische Anpassung der Übertragungsgeschwindigkeit an die Kapazität des Empfängers und des Netzwerks.
- **Überlastungskontrolle:** Vermeidung von Überlastungen im Netzwerk durch Anpassung der Datenübertragungsrate.

Zur Implementierung der Netzwerkkommunikation wurde zunächst der Ethernet-Port am FPGA über Vivado aktiviert (siehe Abbildung 2.1). Dies ermöglicht die grundlegende Verbindung zur Datenübertragung über den auf

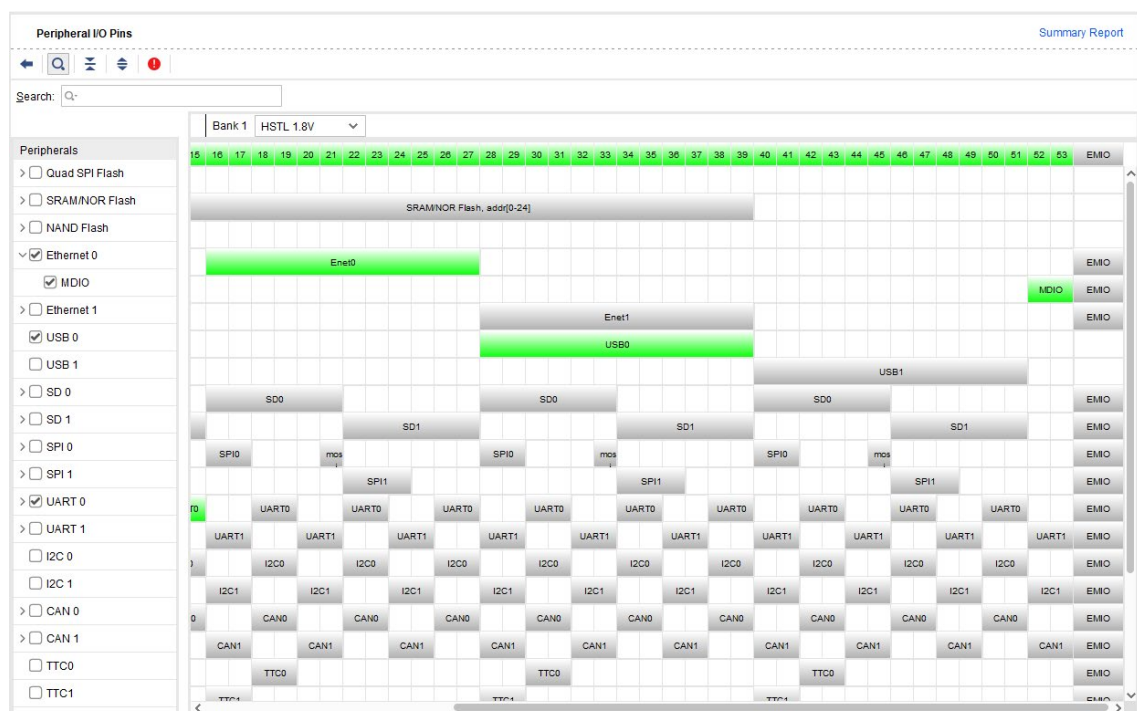


Abbildung 2.1: Konfiguration Ethernet im ZYNQ



dem Cora Board befindlichen Ethernet-Port. Dies kann über MDIO oder EMIO erfolgen:

MDIO (Management Data Input/Output) ist ein serielles Protokoll, das zur Konfiguration und Überwachung von Ethernet-Physikschichten (PHYs) verwendet wird. Es ermöglicht die Kommunikation zwischen dem MAC (Media Access Controller) und dem PHY. MDIO wird verwendet, um Register im PHY zu lesen und zu schreiben, die Geschwindigkeit und den Duplex-Modus zu konfigurieren und den Link-Status zu überwachen.

EMIO (Extended Multiplexed I/O) erweitert die I/O-Fähigkeiten, indem es zusätzliche I/O-Pins bereitstellt, die nicht direkt auf dem FPGA verfügbar sind. EMIO kann verwendet werden, um zusätzliche Peripheriegeräte anzuschließen oder spezielle Funktionen zu implementieren.

Auch die korrekte Konfiguration der Teilnehmer war wichtig. Der Aufbau einer Ethernet-Verbindung wurde in Vitis realisiert. Dabei fungiert der FPGA als Server, an den die Bilddateien geschickt werden.

Die IP-Konfiguration des Servers ist im Code festgelegt worden. Die IP-Adresse vom Client kann im Python Programm frei eingestellt werden.

Das Projekt besteht aus einem Backend und einem Frontend, die zusammenarbeiten, um Bilder zu verarbeiten und an einen Server zu senden.

Das Backend enthält mehrere Funktionen zur Bildverarbeitung und Kommunikation mit einem Server. Es konvertiert Bilder in ein 2D-Array von Hex-Farbwerten im Format 0xGGRRBB, passt die Helligkeit der Bilder an und sendet diese Daten in Blöcken an einen Server. Eine spezielle Stopp-Funktion sendet ein vordefiniertes Signal, um den Server zu stoppen. Außerdem gibt es eine Funktion, die das 2D-Array wieder in ein Bild umwandelt und speichert.

Das Frontend ist dabei eine grafische Benutzeroberfläche, die mit Tkinter erstellt wurde. Es ermöglicht Benutzern, einen Ordner auszuwählen und die darin enthaltenen Bilder anzuzeigen. Benutzer können die Helligkeit der Bilder anpassen, die Bildgröße festlegen und die Server-IP sowie den Port eingeben. Es gibt Optionen, um Bilder einmalig oder zyklisch an den Server zu senden. Die Benutzeroberfläche zeigt die Bilder in einem Raster an und ermöglicht das Umschalten der Bildauswahl mit visueller Rückmeldung. Zyklisches Senden wird in einem separaten Thread verwaltet, um die Benutzeroberfläche reaktionsfähig zu halten.

Zusammen bieten das Backend und das Frontend eine vollständige Lösung zur Bildverarbeitung und -übertragung, die benutzerfreundlich und effizient ist.

Dieser Prozess stellt also sicher, dass alle beteiligten Komponenten ordnungsgemäß eingerichtet und aufeinander abgestimmt werden.

Die AXI DMA (Direct Memory Access) Kanäle im Projekt sind dafür da, Daten effizient zwischen dem Speicher und den Peripheriegeräten zu übertragen, ohne die CPU zu belasten. Hier wird die Unterstützung für vier Kanäle im BSB (Block Support Package) durch die Verwendung von mehreren AXI DMA (Direct Memory Access) Kanälen realisiert.

## 4 Entwurf

### 4.1 Hierarchie

Um Bilder von einem Computer an die LED-Matrix schicken zu können, wird eine Reihe an Programmabläufen benötigt. Zunächst werden die Bilder in Python eingelesen und auf die erforderliche Größe skaliert, um sie für die Übertragung zu optimieren. Diese vorbereiteten Bilddaten werden dann mittels TCP-Protokoll an das Cora Board gesendet. Auf dem Board läuft FreeRTOS als Betriebssystem, welches ein C-Programm ausführt. Dieses Programm ist für den Empfang der TCP-Datenpakete zuständig und verarbeitet die eingehenden Bildinformationen. Nach dem Empfang werden die Daten über den Direct Memory Access (DMA) weitergeleitet, wobei bis zu vier Kanäle genutzt werden können.

### 4.2 Python

Die Hauptfunktionen des Python Programms umfassen folgende Funktionen:

#### **Bildverarbeitung**

Vor dem Versenden werden die Bilder verschiedenen Verarbeitungsschritten unterzogen. Die Größe der Bilder kann auf 32x32, 48x48 oder 64x64 Pixel angepasst werden, abhängig von der zu betreibenden Größe der LED-Matrix. Die Helligkeit der Bilder lässt sich mittels eines Schiebereglers von 1% bis 100% einstellen. Anschließend werden die Bilder in ein zweidimensionales Array von Pixelwerten übergeben.

#### **Bildübertragung**

Die verarbeiteten Bilder können an das Cora Board das als Server fungiert



gesendet werden. Der Benutzer kann die IP-Adresse und den Port des Zielserverns in der Benutzeroberfläche angeben. Es gibt zwei Übertragungsmodi:

1. Einzelbildübertragung: Sendet das erste Bild in der Liste, wenn keine Auswahl getroffen wurde.
2. Zyklische Übertragung: Sendet alle Bilder in der Liste nacheinander in einem benutzerdefinierten Zeitintervall

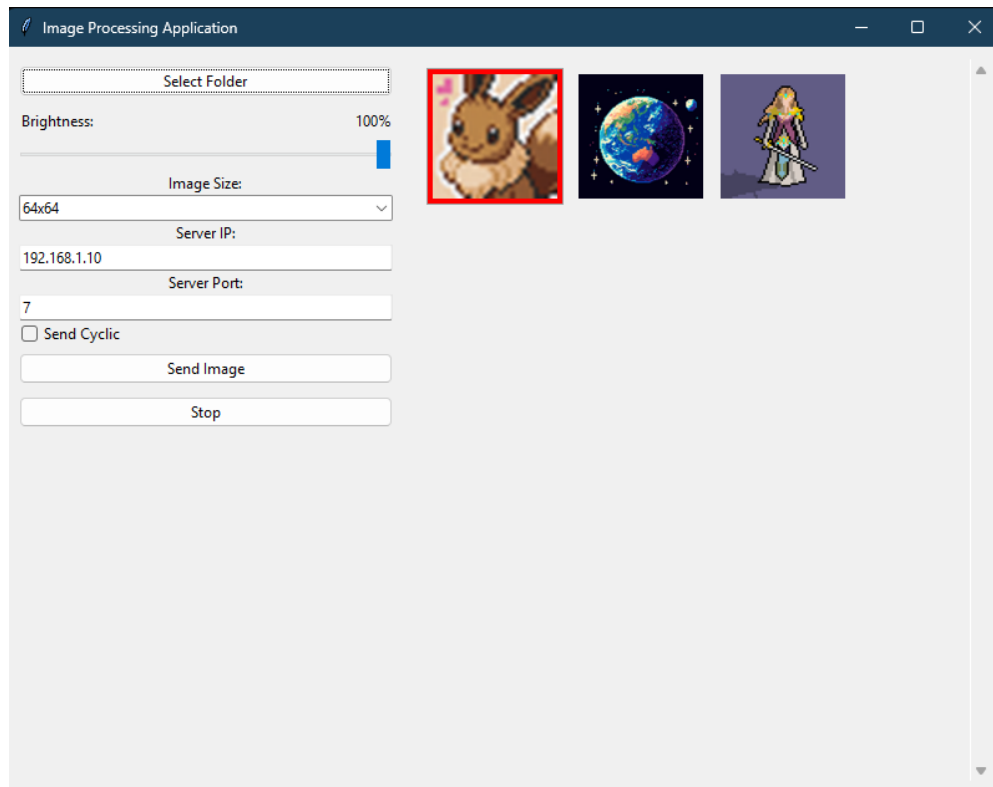
### **Bildauswahl und -anzeige**

Die Anwendung ermöglicht es dem Benutzer, einen Ordner mit Bildern auszuwählen. Die enthaltenen Bilder werden dann in einer scrollbaren Ansicht im rechten Bereich der Benutzeroberfläche dargestellt. Der rote Rahmen um das erste Bild in Abbildung 3.1 zeigt die Auswahl eines einzogen Bildes, das Übertragen werden soll.

### **Technische Implementierung**

#### **Benutzeroberfläche**

Die Anwendung verwendet Tkinter für die grafische Benutzeroberfläche. Sie besteht aus einem linken Bereich für Steuerelemente und einem rechten Bereich zur Bildanzeige. Die Oberfläche ist responsiv und passt sich dynamisch an Benutzerinteraktionen an. So wird ein zusätzliches Bedienfeld eingeblendet, wenn man „Send Cyclic“ ausgewählt hat, um die Intervallzeit einstellen zu können.



**Abbildung 3.1: Python Programm zur Bilderauswahl**

## **Bildverarbeitungsfunktionen**

Die Hauptfunktionen zur Bildverarbeitung umfassen:

- `image_to_pixel_array`: Liest das jeweilige Bild ein und konvertiert dieses in ein zweidimensionales Pixel-Array.
- `stop_function`. Wird aufgerufen um die Übertragung zu beenden.
- `adjust_brightness`: Ändert die Helligkeit des Bildes basierend auf dem Benutzereingabewert.

Für eine bessere Farbwiedergabe ist die minimale Helligkeit auf 20% geschränkt.

## **Netzwerkcommunication**

Die Funktion „`send_large_array_to_server`“ ist für die Übertragung der Bilddaten an den Server verantwortlich. An die angegebene IP-Adresse werden die Daten in Chunks gesendet.

## **Multithreading**

Für die zyklische Bildübertragung wird ein separater Thread verwendet, um die Benutzeroberfläche während des Sendevorgangs reaktionsfähig zu halten. Dies ermöglicht eine ununterbrochene Interaktion mit der Anwendung, während Bilder im Hintergrund gesendet werden

## **4.3 C-Programm**

Die Anwendung ist ein komplexes System zur Steuerung und Darstellung von LED-Matrizen, das sowohl DMA (Direct Memory Access) als auch Netzwerkcommunication nutzt. Mit Hilfe verschiedener Hardwarekomponenten und Softwarebibliotheken wird eine effiziente und flexible Lösung umgesetzt.

Für die Steuerung kommen vier DMA-Kanäle zum Einsatz, die über die Strukturen „`XAxiDma`“ abgebildet sind. Jeder dieser Kanäle ist mit einem eigenen LED-Puffer verbunden, der 1024 32-Bit-Werte speichern kann. Die Initialisierung der DMA-Kanäle und deren Vorbereitung für den Datentransfer erfolgt über die Funktion „`DMA_INIT`“.

Die Verarbeitung der Daten für die LED erfolgt in mehreren Schritten. Zuerst werden insgesamt vier Buffer erstellt welche später die Hex-Werte beinhalten die über den DMA an die jeweiligen LEDs gesendet werden. Mit der Funktion

„Clear\_Buffer“ wird sichergestellt das die Buffer initial korrekt mit 0 initialisiert sind.

In der Main loop werden dann zyklisch die einzelnen Buffer befüllt und an die jeweiligen DMA Instanzen übertragen. Die Buffer werden über die Funktion „display\_matrix\_dma“ befüllt. Zusätzlich wird während der Befüllung der Buffer noch eine Gamma Korrektur vorgenommen. Dazu wird eine Gamma-Korrektur-Tabelle verwendet, welche die Hexwerte der einzelnen LEDs in Rot, Grün und Blau aufteilt und aus der Tabelle die entsprechenden korregierten Werte zieht.

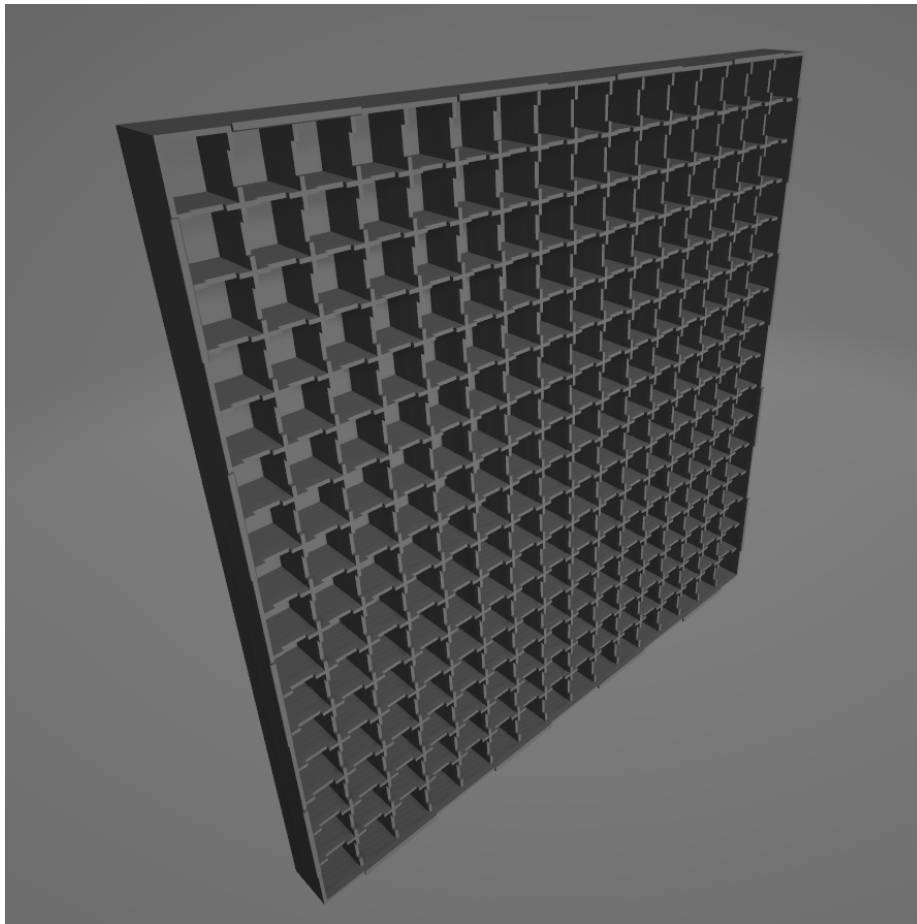
Für die Netzwirkommunikation greift die Anwendung auf das lwIP-Framework zurück. Dabei kann entweder eine statische IP-Adresse genutzt oder eine dynamische Adresse über DHCP bezogen werden. Die Funktion „network\_thread“ ist für die Initialisierung der Netzwerkschnittstelle sowie den Betrieb eines TCP-Servers zuständig.

Das System verwendet FreeRTOS für Multithreading und das parallele Ausführen mehrerer Aufgaben. So übernimmt „network\_thread“ die Netzwirkommunikation, „led\_thread“ kümmert sich um die LED-Steuerung, und „main\_thread“ bildet die Hauptlogik der Anwendung ab.

Schließlich bietet die Anwendung Mechanismen zur Fehlerbehandlung und zum Debugging. Über die serielle Schnittstelle werden Statusinformationen wie IP-Einstellungen und Header der Anwendung ausgegeben, um eine einfache Überwachung und Analyse zu ermöglichen.

## 4.4 3D-Druck

Es wurden insgesamt drei Teile zum 3D-Drucken entworfen, die zusammen eine LED-Matrix aufnehmen. Gedruckt wurden alle Teile mit einer 0,4er Düse. Der Diffusor wird mit zwei Farben gedruckt. Dabei ist die Oberfläche in 3 Schichten in einem diffus-durchsichtigen Material gedruckt und der Rest einfarbig. Die LEDs sind innerhalb des Diffusors voneinander getrennt, um ein Crossbleed zu minimieren.



**Abbildung 3.2: Diffusor-Rückseite**

Der Rahmen mit Abschlussplatte, in dem das LED-Panel sitzt, wurde so entworfen, dass die Verkabelung über rechteckige Löcher ermöglicht wird. Auch eine Belüftung durch die enorme Abwärme soll dadurch gewährleistet werden. Die Rahmen können über die dafür vorgesehenen Löcher miteinander verschraubt werden, um so eine ausreichende Stabilität zu erzeugen. So sollten auch größere Anordnungen wie 4x4 Matrizen oder noch größer ermöglicht werden ohne externe Halter zu benötigen.

Eine Nut im Rahmen sorgt für die Verbindung der zwei Bauteile untereinander. Die Abschlussplatte kann hinten bei Bedarf mit Schrauben am Rahmen selbst befestigt werden.

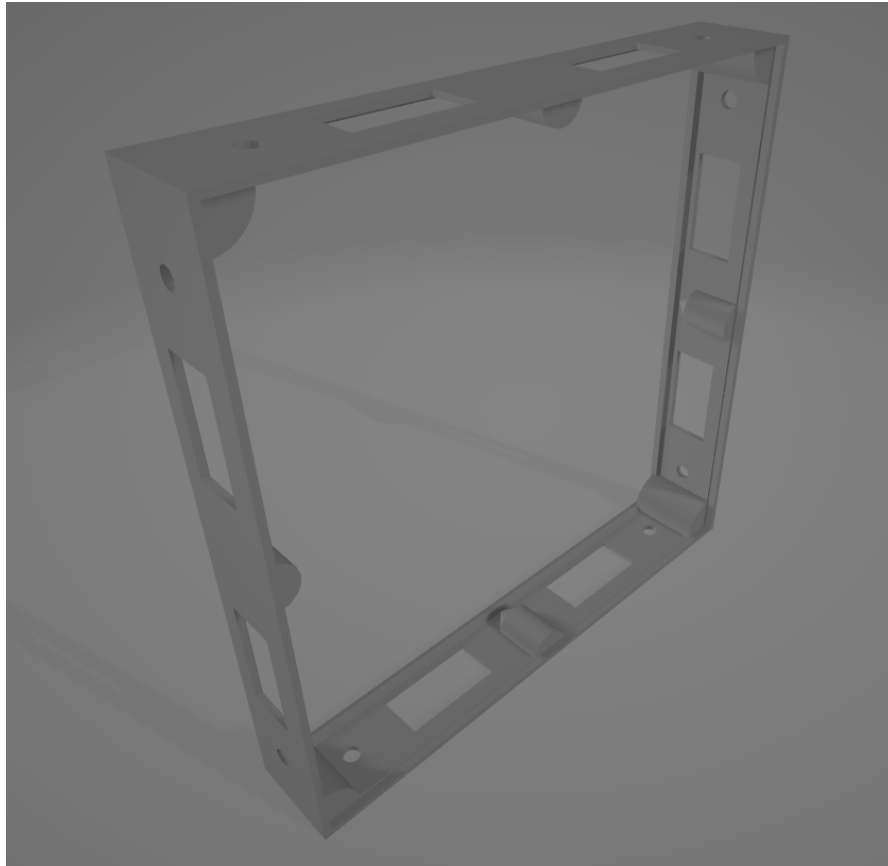


Abbildung 3.3:Rahmen

## 4.5 Fehler und Probleme

### Probleme im Hardwaredesign

Da wir wie bereits in der Einleitung erwähnt auf einem vorrausgegangenen Projekt des letzten Semesters aufgebaut haben, mussten wir erstmal dieses zum laufen bringen. Nach langwieriger Fehlersuche und Rücksprache mit dem Vorentwickler hat sich dann herausgestellt das der letzte Stand den wir Anfangs bekommen hatten nicht lauffähig ist. Durch Zugriff auf das Git Repository konnten die fehlerhaften Änderungen jedoch zurückgesetzt und das Programm in einen lauffähigen Zustand versetzt werden.

### Probleme im C-Code



Da die Netzwerk Komponente unabhängig von der LED Komponente entwickelt wurde gab es beim Verheiraten der beiden Teile Probleme mit der Interrupt initialisierung. Dieses Problem wurde gelöst indem von einer Bare Metall Anwendung auf eine FreeRTOS Anwendung gewechselt wurde. Dadurch wurden für die Netzwerkkomponente keine Interrupts mehr benötigt und es kam zu keinen Problemen durch überschriebene Interrupts mehr.

Ursprünglich war zudem geplant die LED-Ansteuerung und den TCP-Server in separaten Threads laufen zu lassen. Dies wurde auch entsprechend mit Semaphoren realisiert, um den Zugriff auf die Daten zu synchronisieren. Leider kam es dabei jedoch zu dem Problem das sich der Thread des TCP-Servers der die eingehenden Daten verarbeitet nichtmehr terminieren ließ und somit im Speicher festhing. Dadurch kam es nach ca. 50 übertragenen Bildern zu einem `heap_allocation_failed` Error. Dieser wurde gelöst indem nach jedem empfangenem TCP-Paket jetzt genau ein DMA-Transfer durchgeführt wird. Dadurch ist das Programm zwar nicht so elegant wie vorher, es funktioniert jedoch und kann für Diashows etc. Verwendet werden.

Die Vitis IDE bietet leider verglichen mit den üblichen IDE's wenig Quality of life funktionalitäten und ist einfach unhandlich. Deshalb haben wir unser C-Programm zu 95% mit Visual Studio Code / Windsurf geschrieben und Vitis nur zum compilieren und um das Programm aufs Cora Board zu ziehen benutzt.

### **Probleme im Python Code**

Bei der Entwicklung des Python Programms musste die `Send_Array_to_Server` Funktion so angepasst werden das das Chunking der TCP-Pakete manuell erfolgt. Über TCP ist es nämlich leider nicht möglich größere TCP-Pakete auf einmal zu senden ( $4096 \times 32 \text{ Bit} \sim 16 \text{ kbyte}$ ). Da nun jedoch insgesamt 17 TCP-Pakete gesendet wurden (16x256 Daten Pakete + 1 End of Transmission Paket) gab es Probleme mit dem standartmäßig aktivierten Nagle-Algorithmus. Dieser verhindert das mehrere kleine Pakete gesendet werden, um so die Auslastung des TCP-Netzwerkes gering zu halten. Dadurch kam es jedoch zu Problemen mit dem manuellen Empfangsalgorithmus auf der C-Seite weshalb dieser vor dem senden in Python mit `"sock.setsockopt(socket.IPPROTO_TCP, socket.TCP_NODELAY, 1)"` deaktiviert werden musste.

Das von uns verwendete GUI-Toolkit Tkinter war zudem in der Visual Studio IDE nicht sehr intuitiv zu bedienen, hier wäre eine andere IDE mit besserer Kompabilität besser gewesen um die anordnung der Elemente einfacher zu gestalten.

## 5 Fazit

In diesem Projekt wurden zahlreiche neue Systeme und Abläufe entdeckt, insbesondere im Zusammenhang mit der LED-Matrix-Steuerung und der Netzwirkommunikation. Die Kombination aus dem FPGA-basierten Cora Board und der komplexen LED-Ansteuerung bot einige Herausforderungen, die durch intensive Recherche und den Austausch mit Kommilitonen bewältigt wurden. Das Einarbeiten in dieses neue, umfangreiche Konzept, das von der Python-basierten Bildverarbeitung über TCP-Kommunikation bis hin zur DMA-gesteuerten LED-Ansteuerung reicht, bot mehr Herausforderungen als vorangegangene Projekte.

Die Entwicklung des Gehäuses für die LED-Matrix mittels 3D-Druck stellte eine zusätzliche Herausforderung dar, die kreative Lösungen erforderte. Besonders die Aspekte der Verkabelung, Belüftung und Stabilität mussten sorgfältig bedacht werden.

AMD Support Forum bot eine wertvolle Wissensbasis auf die oft zurückgegriffen wurde. Es lieferte zahlreiche Einblicke in verschiedene Anwendungsmöglichkeiten von FPGAs, wie auch in deren spezifische Herausforderungen bei der Systementwicklung. Diese Ressource war besonders hilfreich bei der Implementierung der Ethernet-Kommunikation und der effizienten Datenverarbeitung auf dem Cora Board mit FreeRTOS. Leider erschwerten tote Web-Links durch die Übernahme von AMD die Recherche oder gar die Lösung einiger Probleme.

## 6 Abbildungsverzeichnis

Abbildung 2.1: Konfiguration Ethernet im ZYNQ .....	6
Abbildung 3.1: Python Programm zur Bilderauswahl.....	10
Abbildung 3.2: Diffusor-Rückseite .....	<b>Fehler! Textmarke nicht definiert.</b>
Abbildung 3.3:Rahmen .....	14