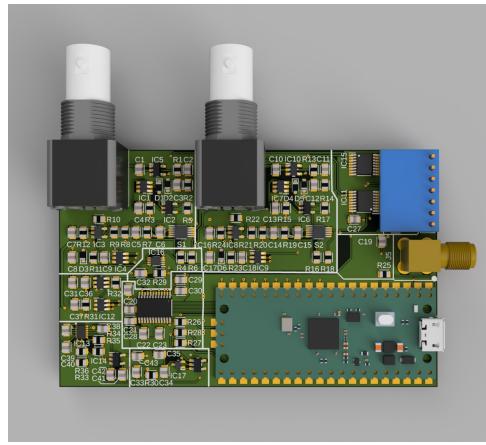


HTWK Leipzig
Elektrotechnik und Informationstechnik



Abschlussbeleg Projekt Oszilloskop
Embedded Systems 1

vorgelegt von:
Johannes Pautz, Max Sämann

Matrikelnummer:
77879, 77709

eingereicht:
Leipzig, 17.03.2024



Github-Repository

Inhaltsverzeichnis

Inhaltsverzeichnis	I
1 Aufgabe	1
1.1 Ursprüngliche Aufgabe	1
1.2 Aufgabenanalyse	1
1.2.1 Problemfall Kanalanzahl und Sample Rate	1
1.2.2 Zielsetzung	2
2 Orientierung	3
2.1 Umsetzung Analog-Digital Wandlung	3
2.1.1 interner ADC	3
2.1.2 externer ADC	3
2.2 Inspirationsfindung	4
2.3 Projekt „RPScope“	4
2.4 Was kann verwendet werden?	5
2.5 Notwendige Anpassungen	6
3 Umsetzung	8
3.1 neuer Schaltplan	8
3.1.1 Austausch von Komponenten	9
3.1.2 Zusätzliche Komponenten	12
3.1.3 Resultat	14
3.2 Neues Platinenlayout	16
3.3 Inbetriebnahme	17
3.3.1 Programmierung des Pi Pico mit Micropython	18
4 Probleme	19
4.1 Schwierigkeiten der Nutzung externer Bibliotheken mit Micropython	19
4.2 Abweichende Software	19
4.3 Fehler im neuen Platinenlayout	20
5 Mögliche Problemlösungen	22
5.1 Displayansteuerung	22
5.2 Softwareanpassung	24
5.3 Verbesserung des Platinenlayouts	24
6 Fazit	25
6.1 Stand zu Projektende	25
6.2 Offene Punkte	25
6.3 Eigene Betrachtung	25

1 Aufgabe

1.1 Ursprüngliche Aufgabe

Aus der ursprünglichen Aufgabenstellung ergeben sich folgende technische Anforderungen:

- Bis zu 3 Spannungssignale
- Einstellbare t/div und u/div
- RMS-Berechnung pro Kanal
- Start/Stopp
- Frequenzanzeige pro Kanal
- f_{max} 100 MHz
- Steuerung durch Raspberry Pi Pico oder STM32
- Einfache und stabile Installation

1.2 Aufgabenanalyse

Die Punkte der einstellbaren $\frac{t}{div}$, sowie $\frac{u}{div}$, RMS-Berechnung, Start/Stopp und Frequenzanzeige werden sich in der Software lösen lassen und sollten somit keine Probleme darstellen.

Die Wahl des Microcontrollers für die Steuerung ergibt sich durch die diesjährige Vorgabe, es wird der Raspberry Pi Pico mit RP2-B2 Prozessor gewählt. Dieser kann sowohl mit C als auch Micropython programmiert werden.

1.2.1 Problemfall Kanalanzahl und Sample Rate

Bei der Kombination aus 3 Kanälen und einer Maximalfrequenz von 100 MHz ergeben sich Probleme. Würde man versuchen diese Kombination auszureißen so würde man nach „Faustformel“ eine Sample Rate von etwa 1.5 Gsps benötigen, welche nicht mit dem Prozessor des Pi Pico zu verarbeiten ist.

Im Normalfall sollte man das Fünffache der gewünschten Maximalfrequenz als Sample Rate haben um jegliche Funktionen ausreichend darstellen zu können. Über drei Kanäle kommt man so auf die nötigen 1.5 Gsps. Der Prozessor des Pi Pico läuft jedoch nur mit 160 MHz Taktfrequenz auf 2 Kernen. Bei einem perfekt organisierten Programm wären so 320 Msps, bzw. 21,3 MHz Bandbreite pro Kanal möglich. Jedoch birgt das Programmieren für beide Kerne jedoch einige Tücken, was den Umfang des Projektes um einiges erhöhen würde. Zusätzlich dazu lässt sich der Pi Pico auf bis zu 400 MHz pro kern übertakten, dies birgt jedoch wieder Risiken wie Überhitzung und unvorhergesehenes Verhalten, was die Anforderung an die Zuverlässigkeit nicht erfüllen würde.

1 Aufgabe

So bleibt uns am Ende ein Prozessorkern mit 130 MHz, der wenn nötig recht sicher auch auf 200 MHz gebracht werden kann. Zusätzlich kann die Kanalanzahl auf 2 reduziert werden, um pro Kanal mehr Bandbreite bereit stellen zu können. So ergibt sich für ein 2 Kanal Oszilloskop eine realistische Bandbreite von 13 MHz pro Kanal.

1.2.2 Zielsetzung

Nach Rücksprache mit dem Projektbetreuer ergaben sich folgende Ziele, die umzusetzen sind:

- 2 Kanäle
- Möglichst hohe Sample Rate: gängige ADC-Werte sind 100 oder 150 Msps
- Möglichkeit für integrierten Frequenzgenerator
- Andere, nicht widersprüchliche Ziele aus Ursprungsaufgabe

2 Orientierung

2.1 Umsetzung Analog-Digital Wandlung

Den kritischen Punkt eines Digitaloszilloskops stellt immer die Wandlung des Analogen Eingangssignals in ein Digitales Signal dar. Hier entscheidet sich die Sample Rate, welche neben der Bandbreite der davor liegenden Analogschaltung festlegt wie hoch die Frequenz der zu messenden Signale maximal sein kann.

Uns bieten sich zwei Möglichkeiten wie folgend:

2.1.1 interner ADC

Die gängige Methode ein simples Oszilloskop mit einem Microcontroller zu realisieren ist den integrierten ADC des Controllers zu verwenden, im Fall des Pi Pico handelt es sich um einen 12 Bit - 4 Kanal ADC mit einer Sample Rate von 500ksps¹. So wären uns nach schneller Rechnung 4 Kanäle mit jeweils 100kHz Bandbreite möglich, was nicht annährend der Anforderung entspricht. Somit ist diese Variante hinfällig.

2.1.2 externer ADC

Anbindung per I2C

Ausschlaggebend, ob der ADC über I2C angebunden werden könnte, ist die maximale Geschwindigkeit des I2C Bus am Pi Pico. In diesem Fall sind im „fast mode plus“ maximal 1000kbps möglich². Somit ergibt sich auch eine maximale Sample Rate von 1Msps pro Kanal bei Nutzung von 2 ADCs, um die zwei I2C Controller des Pi Pico auszureizen, was für unsere Anforderungen nicht ausreichend ist.

Anbindung per parallel Datenbus

Im Falle des Datenbusses wird für jedes Bit des ADCs ein Pin des Pi Pico genutzt. Die Pins des Pi Pico können maximal mit der Prozessorfrequenz gelesen werden, so könnte man theoretisch ohne weitere Eingriffe eine Sample Rate von 130 Msps realisieren, was aufgeteilt auf 2 Kanäle eine Sample Rate von 65 Msps oder eine Bandbreite von 13 MHz pro Kanal ergibt.

Die zwei Nachteile, die diese Methode mit sich bringt, sind folgende: Einerseits werden so am Pi Pico auch pro Bit Auflösung des ADCs ein Pin am Pi Pico benötigt. Wenn man ein hoch genaues Gerät bauen möchte mit z.B. einem 12 Bit ADC werden so auch 12 Pins benötigt, möglicherweise sogar 24 bei Nutzung von 2 ADCs. Der Pi Pico selbst besitzt nur 28 integrierte GPIO-Pins, welche alle mit voller Geschwindigkeit genutzt werden können. Davon müssen einige frei gehalten werden für weitere Anwendungen, die eine hohe Geschwindigkeit erfordern, wie z.B. der Kanalwechsel per Multiplexing.

¹Nach RP2040 Datenblatt S.10

²Nach RP2040 Datenblatt S.442

2 Orientierung

Um die Pins des Pi Pico mit voller Prozessorgeschwindigkeit lesen zu können, muss in der Programmierung direkt auf den Speicher des Prozessors zugegriffen werden, was die Programmierung im Vergleich zur gängigen „pin read“ Methode durch zusätzliche Abstraktion erschwert.

Trotz der zusätzlichen Komplexität in Hard- und Software ist dies der einzige Weg ein Oszilloskop mit entsprechender Sample Rate zu bauen und somit die einzige Wahl für dieses Projekt.

2.2 Inspirationsfindung

In der Welt der Microcontroller lässt sich oft ein Projekt finden, dass in etwa dem entspricht, was man sucht und zudem auch noch öffentlich zur Verfügung steht. Dies röhrt aus dem allgemeinen „Open-Source Gedanken“ dieser Community welcher durch Firmen wie Arduino, Raspberry Pi und Adafruit stetig vorangetrieben wird. So lässt sich in der Regel viel Zeit sparen, indem man ein solches Projekt als Startpunkt nimmt und es auf die eigenen Bedürfnisse anpasst. Zusätzlich dazu lässt sich auch immer etwas Neues lernen, wenn man in die Projekte anderer eintaucht. Aus diesem Grund soll in diesem Projekt solch eine öffentliche Arbeit als Grundstein dienen.

2.3 Projekt „RPScope“

Nach einiger Recherche über mögliche Kandidaten fand sich das Projekt „RPScope³“ von „jpgeiro“ auf Hackaday.io. Hier handelt es sich um ein Oszilloskop auf Basis des Raspberry Pi Pico mit 2 Kanälen und einem externen Texas Instruments ADC08100 Analog-Digital Wandler, welcher mit einer Sample Rate von 100 Msps arbeitet. So ergibt sich nach unserer Rechnung eine Bandbreite von 10 MHz pro Kanal, was eine deutliche Verbesserung zur Verwendung des internen ADC darstellt und das Oszilloskop auch für Aufgaben wie z.B. die Logikanalyse von einem I2C Bus oder der Betrachtung von PWM-Signalen nützlich macht. Die Bedienung erfolgt über ein 3,5 Zoll großes Touch-Display.

Das Projekt wirkt als Grundlage vielversprechend und sollte vieles an Arbeit ersparen wenn es um die Einbindung einzelner Komponenten, die Benutzeroberfläche und das Design der Analogstrecke geht.



Abb. 2.1: Platinenvorlage

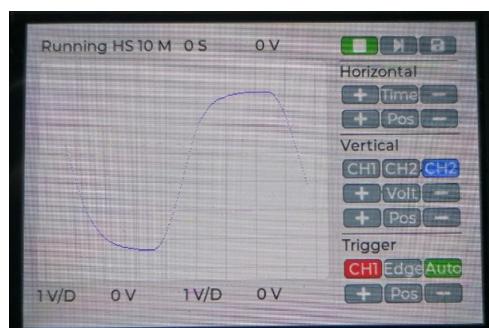


Abb. 2.2: Displayvorlage

³<https://hackaday.io/project/188051-rpscope>

2 Orientierung

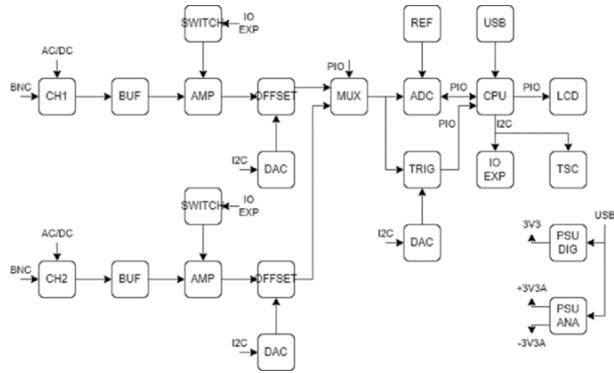


Abb. 2.3: Übersicht Elektronikaufbau

2.4 Was kann verwendet werden?

Das Projekt soll nur als Basis dienen und nicht nur nachgebaut werden. Als Grundstein liefert es uns jedoch einige kritische Bestandteile:

Schaltplan

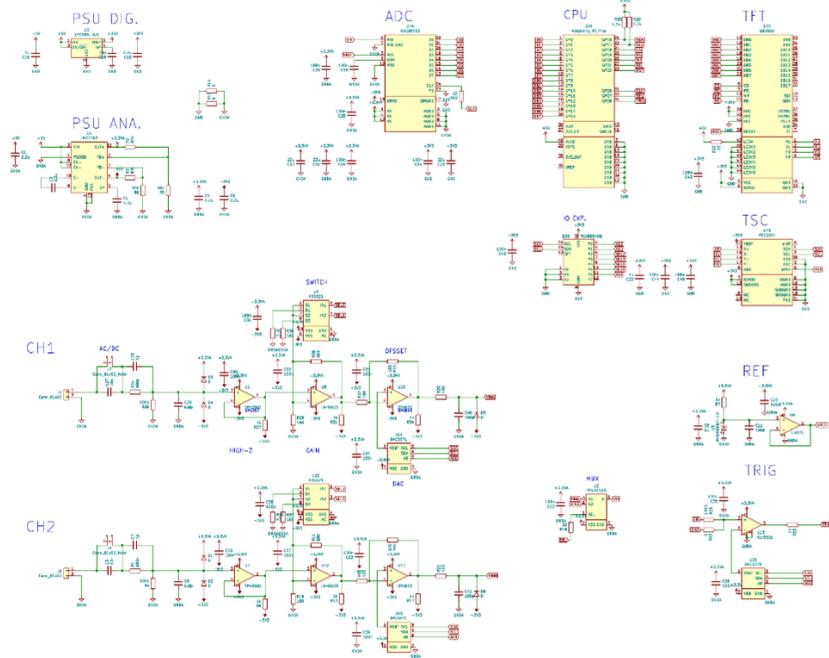


Abb. 2.4: Schaltplan des Ursprungsprojektes

2 Orientierung

Der Schaltplan bietet uns fertige Funktionsblöcke für die unterschiedlichen Bestandteile, die das Gerät benötigt. Besonders aufwendig ist hier die Analogschaltung für die beiden Eingänge, welche aufwändig zusammengestellt und simuliert werden müsste. In diesem Fall ist diese bereits auf eine Bandbreite von 100MHz ausgelegt.

Software

Eine sehr große Hilfe stellt die vorhandene Software dar. In ihr sind bereits komplexe Funktionen wie das Auslesen des ADCs über direkten Speicherzugriff, wie in 2.1.2 bereits erwähnt, implementiert, sodass die Daten auch ausreichend schnell eingelesen werden können. Zusätzlich existieren bereits Funktionen für die Messwerterfassung, den Edge-Trigger und Die Benutzeroberfläche inklusive Display Treiber für die verwendete Grafikbibliothek. Um all das zu realisieren wurden auch bereits Klassen für die jeweiligen ICs die auf dem Board zusätzlich verwendet werden deklariert.

Dokumentation

Die vorhandene Hard- und Software ist zusätzlich in englischer Sprache recht ausgiebig dokumentiert du in ihrer Funktion zumindest grundlegend erklärt was im besten Fall dem Verständnis des Vorhandenen helfen sollte.

2.5 Notwendige Anpassungen

Platine

Im Projekt existiert bereits ein Platinenlayout was durchaus verwendet werden könnte, jedoch wollen wir noch einige Funktionen hinzufügen, welches zusätzliche ICs erfordert. Ebenfalls werden im Original SMD Komponenten der Größe „0603“ verwendet, welche aufgrund ihrer sehr minimalen Größe schwierig von Hand zu Löten sind, hier möchten wir lieber auf die nächstgrößere „0805“ zurückgreifen. Zusätzlich besitzt die Originalplatine vier Lagen, was in diesem Fall zwar das Routing der Traces einfacher macht aber nicht wirklich nötig ist, die Platine soll auf ein zweilagiges Design mit nur Vor- und Rückseite reduziert werden. Dazu kommen noch kleinere Änderungen wie z.B. die Umstellung der SMA Buchsen für die Eingänge auf Oszi-übliche BNC Buchsen sowie das Hinzufügen einer FPC Buchse für das Display, anstatt dieses fest anzulöten. Trotz all dem soll die Platine ähnliche Abmaße behalten, damit sie in etwa dieselbe Größe wie das Display besitzt. So kann das Gerät klein gehalten werden, indem das Display direkt hinter der Platine sitzt. Es wird ein komplett neues Platinenlayout benötigen.

Ersetzen einzelner Komponenten

Einige Komponenten der ursprünglichen Auswahl entsprechen nicht ganz den benötigten Anforderungen, was auch teils in den Kommentaren des Projektes von anderen Nutzern angemerkt wird. Diese gilt es zu ersetzen und bei möglicherweise Abweichendem Pinout oder Ansteuerung die Anbindung auf der Platine oder im Code zu realisieren.

Integrierter Frequenzgenerator

Um unser Ziel zu erfüllen, muss ein Frequenzgenerator zur Platine hinzugefügt werden, der grundlegende Funktionen wie Sinus, Rechteck und Dreieck in unterschiedlichen Frequenzen und Amplituden ausgeben können sollte.

2 Orientierung

Umschaltung der AC/DC Kopplung

Auf dem vorhandenen Schaltplan erfolgt die Umschaltung der Kopplung über das Stecken, bzw. Abziehen von Jumpern, was nicht wirklich nutzerfreundlich ist und auch etwas improvisiert wirkt. Das gilt es zu ändern, damit dies auch über das Display getan werden kann.

Software

Die Software sollte größtenteils in Originalform verwendet werden können wie oben bereits erwähnt. Trotzdem müssen ein paar mathematische Funktionen zur Berechnung der RMS-Werte der Signale und der Frequenzanzeige hinzugefügt werden. Dazu kommt noch die Implementierung eines Frequenzgenerators sowie das Hinzufügen einer Möglichkeit die Umschaltung von AC/DC Kopplung über das Display, statt Jumpern vorzunehmen. All das muss nicht nur im Code angesteuert werden, sondern auch entsprechende Schaltflächen zur Benutzeroberfläche hinzugefügt werden.

3 Umsetzung

3.1 neuer Schaltplan

Der originale Schaltplan soll anhand folgender Punkte angepasst werden, um einige Schwachstellen der Schaltung zu entfernen und weitere Funktionalität hinzuzufügen. Ebenfalls soll der Elektronikentwurf in Autodesk Fusion 360 statt KiCAD wie das Original getätigt werden. Das soll die spätere Modellierung eines Gehäuses in Fusion 360 erleichtern, da so Änderungen an Gehäuse und Platine direkt übertragen werden können. Dazu erfordert es aber den gesamten Schaltplan in Fusion 360 nachzubauen und auch für alle Bauteile kompatible Bibliotheken herauszusuchen und zu importieren.

Aufheben der Trennung von GND und AGND

Im Original werden die GND-Potenziale der analogen und digitalen Schaltung auf der Platine voneinander getrennt. Aufgrund von möglichen Einkopplungen und den Vorteilen die eine große, einheitliche GND-Fläche mit sich bringt ist es vorteilhaft diese Trennung aufzulösen. Ebenfalls erleichtert es das Platinenlayout, wenn nicht zwei Arten von GND unterschieden werden müssen.



Abb. 3.1: GND Fläche

3.1.1 Austausch von Komponenten

Multiplexer

Bereits im originalen Projekt wird vom Ersteller bemängelt, dass das Oszilloskop aufgrund des zu langsamem Multiplexers nicht die maximale Sample Rate des ADC ausschöpfen kann. Der verwendete Texas Instruments TMUX1119 IC besitzt laut Datenblatt⁴ eine Umschaltzeit von 14ns. Für 100 Msps benötigen wir ein Multiplexer mit maximal $\frac{1}{100.000.000}$ s Umschaltzeit, also maximal 10ns.

DYNAMIC CHARACTERISTICS					
t_{TRAN}	Switching time between channels	$V_S = 2V$ $R_L = 200\Omega$, $C_L = 15\text{pF}$ Refer to Section 6.4	25°C	14	ns
			-40°C to +85°C	20	ns
			-40°C to +125°C	21	ns

Abb. 3.2: Charakteristik Multiplexer

Nach Filterung der verfügbaren ICs mit passender Betriebsspannung von 3-5VDC, gleicher Kanalanzahl und ähnlichem Footprint erscheint der Texas Instruments SN74LVC2G53DCUR geeignet. Er besitzt laut Datenblatt⁵ lediglich eine Umschaltzeit von max. 7,2ns.

$t_{\text{en}}^{(2)}$	A	COM or Y	$V_{CC} = 1.8 \text{ V} \pm 0.15 \text{ V}$	2.9	10.3	ns
			$V_{CC} = 2.5 \text{ V} \pm 0.2 \text{ V}$	2.1	7.2	
			$V_{CC} = 3.3 \text{ V} \pm 0.3 \text{ V}$	1.9	5.8	
			$V_{CC} = 5 \text{ V} \pm 0.5 \text{ V}$	1.3	5.4	
			$V_{CC} = 1.8 \text{ V} \pm 0.15 \text{ V}$	2.1	2.1	
$t_{\text{dis}}^{(3)}$	A	COM or Y	$V_{CC} = 2.5 \text{ V} \pm 0.2 \text{ V}$	1.4	7.9	ns
			$V_{CC} = 3.3 \text{ V} \pm 0.3 \text{ V}$	1.1	7.2	
			$V_{CC} = 5 \text{ V} \pm 0.5 \text{ V}$	1	5	

Abb. 3.3: Kennzahlen neuer Multiplexer

Gain-Verstärker der Analogstufe

Der verwendete Texas Instruments LMH6629 IC hat eine maximal zulässige Betriebsspannung von 6VDC, während er auf der Platine aber mit +3,3V und -3,3V, also 6,6V betrieben wird. Dieser kann problemlos mit einem Analog Devices LT1818 mit identischem SOT23-5 Footprint ersetzt werden. Der LT1818 kann mit bis zu 11V betrieben werden und stellt somit kein Problem mehr dar.

⁴<https://www.ti.com/lit/ds/symlink/tmux1119.pdf>

⁵<https://www.ti.com/lit/ds/symlink/sn74lvc2g53.pdf>

Offset Verstärker der Analogstufe

Der ohnehin als Alternative angegebene Texas Instruments OPA820 Operationsverstärker besitzt eine höhere Bandbreite als der originale OPA830 aus derselben Serie. Der OPA830 stellte vorher mit seiner 240MHz Bandbreite laut Datenblatt das „schwächste Glied“ der Eingangskette dar. Der OPA820 kostet dasselbe besitzt aber unter denselben Bedingungen eine 800MHz Bandbreite. Dieser besitzt aber jedoch einen Pin mehr, einen „Enable“ Pin, welcher zur Aktivierung des OPVs auf Betriebsspannung gebracht werden muss. Das sollte im Schaltplan und Platinenlayout beachtet werden.

IO-Expander

Der eingeplante Texas Instruments TCA9554 IO-Expander, welcher benötigt wird, da der Pi Pico selbst nicht mehr genügend Pins frei hat, um die Analogschaltung zu steuern, ist derzeit nicht lieferbar. Aus derselben Serie existiert aber der TCA9554A (selbe Nr. nur mit einem A dran), welcher derselbe IC ist aber mit einem anderen Adressbereich und dazu noch lieferbar. Die abweichende Adresse muss in der Software angepasst werden.

Table 2. Address Reference

INPUTS			I ² C BUS SLAVE ADDRESS
A2	A1	A0	
L	L	L	32 (decimal), 20 (hexadecimal)
L	L	H	33 (decimal), 21 (hexadecimal)
L	H	L	34 (decimal), 22 (hexadecimal)
L	H	H	35 (decimal), 23 (hexadecimal)
H	L	L	36 (decimal), 24 (hexadecimal)
H	L	H	37 (decimal), 25 (hexadecimal)
H	H	L	38 (decimal), 26 (hexadecimal)
H	H	H	39 (decimal), 27 (hexadecimal)

Abb. 3.4: TCA9554

Table 2. Address Reference

INPUTS			I ² C BUS SLAVE ADDRESS
A2	A1	A0	
L	L	L	56 (decimal), 38 (hexadecimal)
L	L	H	57 (decimal), 39 (hexadecimal)
L	H	L	58 (decimal), 3A (hexadecimal)
L	H	H	59 (decimal), 3B (hexadecimal)
H	L	L	60 (decimal), 3C (hexadecimal)
H	L	H	61 (decimal), 3D (hexadecimal)
H	H	L	62 (decimal), 3E (hexadecimal)
H	H	H	63 (decimal), 3F (hexadecimal)

Abb. 3.5: TCA9554A

Display

Das Display im Original ist undefiniert bis auf den ILI9488 Controller und der Auflösung. Anfänglich wirkte das Adafruit HX8357D Display⁶ vielversprechend da es eine identische Größe und identische Pins besitzt. Dieses stellte sich jedoch durch den hohen Preis von 40€ und seinen vielen THT-Pins als sehr unpraktisch dar und wurde somit für die zweite Platine verworfen.

Nach weiterer Suche ließ sich ein passendes Display auf Aliexpress⁷ ausfindig machen. Dieses besitzt jedoch eine völlig andere Pin-Anordnung als auf der Originalplatine. Dafür wurde ein eigenes Symbol samt Footprint anhand der Artikelbeschreibung erstellt, um die richtige Anbindung zu gewährleisten. Bei diesem Display handelt es sich außerdem nur um ein blankes Display mit FPC-Flex Verbindung, was einiges an Platz spart und auch den Preis mit 8€ sehr niedrig hält.

Zusätzlich soll das Display statt fest verlötet zu werden durch eine FPC-Buchse mit der Hauptplatine verbunden werden. Eine entsprechende Buchse wurde als Footprint zum Display hinzugefügt, um auf der Platine platziert zu werden.

⁶<https://www.adafruit.com/product/2050>

⁷<https://de.aliexpress.com/item/32829704426.html>

3 Umsetzung

Wichtig ist beim Display zusätzlich, dass je nachdem ob die Pins IM0, IM1, IM2 jeweils auf VCC-Potenzial (+3.3V) oder auf GND-Potenzial sind der Eingang des Displays gewählt werden kann. Die entsprechende Konfiguration ist dem ILI9488 Datenblatt⁸ zu entnehmen.

Table 1: MIPI-DBI Operating Mode

MIPI-DBI Type B				
IM2	IM1	IM0	Interface	Data Pin in Use
0	0	0	24-bit bus (DB_EN=1)	DB [23:0]
0	0	0	18-bit bus (DB_EN=0)	DB [17:0]
0	0	1	9-bit bus	DB [8:0]
0	1	0	16-bit bus	DB [15:0]
0	1	1	8-bit bus	DB [7:0]
MIPI-DBI Type C				
1	0	1	Option1 (3-line SPI)	SDA,SDO
1	1	1	Option3 (4-line SPI)	SDA,SDO

Abb. 3.6: Ansteuerung Display

⁸<https://www.hpinfotech.ro/ILI9488.pdf>

3 Umsetzung

In unserem Fall möchten wir den 8 Bit Parallel Port nutzen. Daraus ergibt sich folgende Belegung am Display:

Gerät	Pin	Funktion	Display Pin	Name	Bemerkung
Touch	16	X-	1	XL(X-)	
Touch	17	Y-	2	YU(Y-)	
Touch	13	X+	3	XR(X+)	
Touch	14	Y+	4	YD(Y+)	
PSU	-	GND	5	GND	
PSU	-	+3.3V	6	VCC	
PSU	-	+3.3V	7	VCC	
-	-	-	8	FMARK	
Pico	9	CS	9	CS	
Pico	10	SCL	10	RS/SCL	
Pico	8	AO/DC	11	WR/AO	
-	-	-	12	RD	
Pico	11	MOSI	13	SDA	
Pico	12	MISO	14	SDO	
Pico	22	RST	15	RESET	
PSU	-	GND	16	GND	
-	-	-	17-32	DB0-15	Parallel Ansteuerung
PSU	22R	Backlight	33	A	Anode LED Backlight
PSU	-	GND	34	K1	Kathode LED Backlight
PSU	-	GND	35	K2	Kathode LED Backlight
PSU	-	GND	36	K3	Kathode LED Backlight
PSU	-	GND	37	GND	
PSU	-	+3.3V	38	IM0	Input Mode: setzt genutzten Eingang fest, hier 4 Wire SPI
PSU	-	+3.3V	39	IM1	
PSU	-	+3.3V	40	IM2	

3.1.2 Zusätzliche Komponenten

Umschaltung der AC/DC Kopplung

Um die AC/DC Kopplung mit Hilfe des Displays umschalten zu können, muss statt dem vorhandenen Jumper ein durch die Software steuerbarer Schalter an derselben Stelle eingesetzt werden. Hierfür gibt es Analogschalter ICs, welcher eine A/B Strecke je nach Eingangssignal an C leiten oder nichtleitend schalten. Aufgrund der 3.3VDC Betriebsspannung und der niedrigen Schaltfrequenz reicht hier das Modell SN74LVC1G66DBVRE4 von Texas Instruments. Anstatt der Jumper wird jeweils einer dieser ICs an dieser Stelle eingesetzt. Dadurch benötigt es aber nun noch 2 IO-Pins mehr um diese anzusteuern, was einen weiteren TCA9554A IO-Expander erfordert. Bei diesem sollte beachtet werden, dass die Adress-Pins anders angeschlossen werden müssen als beim ersten.

3 Umsetzung

Funktionsgenerator

Als möglicher Kandidat hat sich der Analog Devices AD9833 IC herauskristallisiert. Dieser kann Sinus-, Dreieck- und Rechtecksingale von 0-12.5MHz ausgeben, was sehr gut den 10MHz Bandbreite der Spannungseingänge entspricht. Ebenfalls kann dieser mit 5VDC betrieben werden. Angesteuert wird der IC über I2C, was mit dem Pi Pico machbar sein sollte. Aufgrund des engen Platzangebotes auf der Platine und dem Fakt, dass eine fertige Platine mit diesem IC samt Peripherie günstiger ist als der Chip selbst, wird solch eine fertige Platine mittels Stiftleiste über der Hauptplatine verbaut. Auf der Hauptplatine wird eine SMA-Buchse für den Ausgang hinzugefügt.

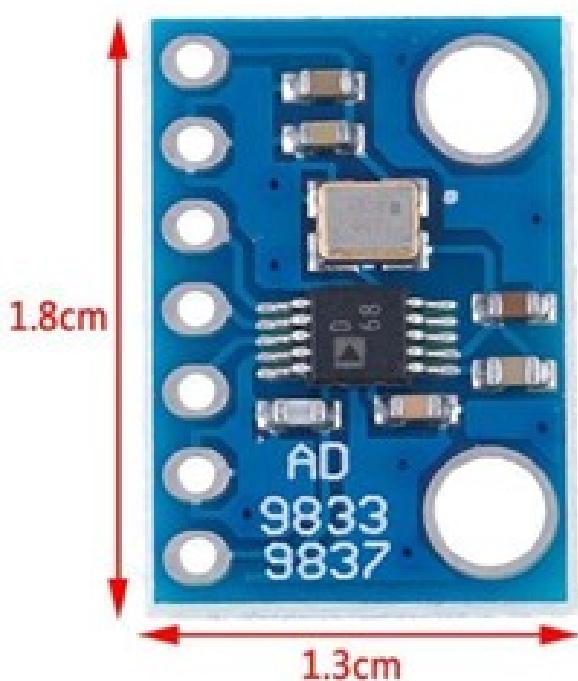


Abb. 3.7: AD9833 Funktionsgenerator



Abb. 3.8: AD9833 Funktionsgenerator

3 Umsetzung

3.1.3 Resultat

Nachdem alle Anpassungen beachtet wurden und der Schaltplan in Fusion 360 neu erstellt wurde ergibt sich folgendes Ergebnis:

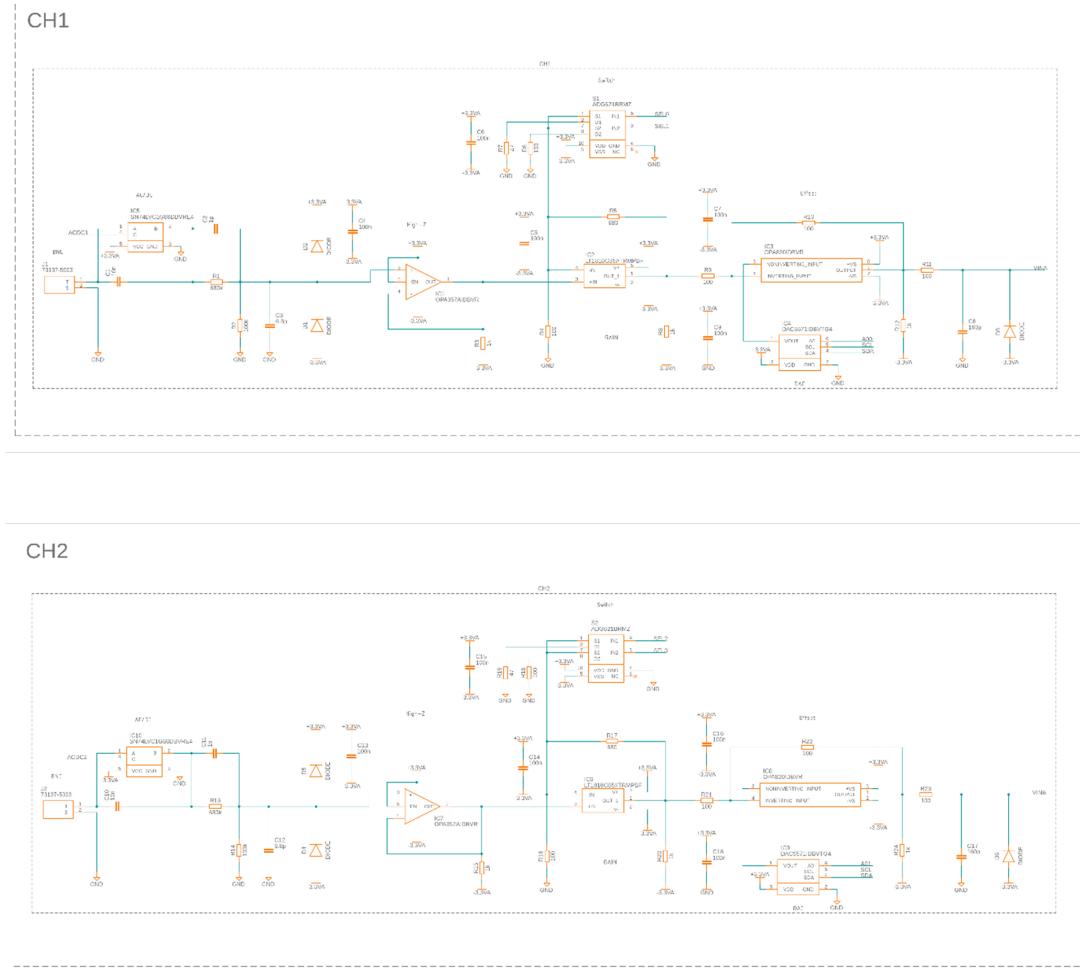


Abb. 3.9: Schaltplan

3 Umsetzung

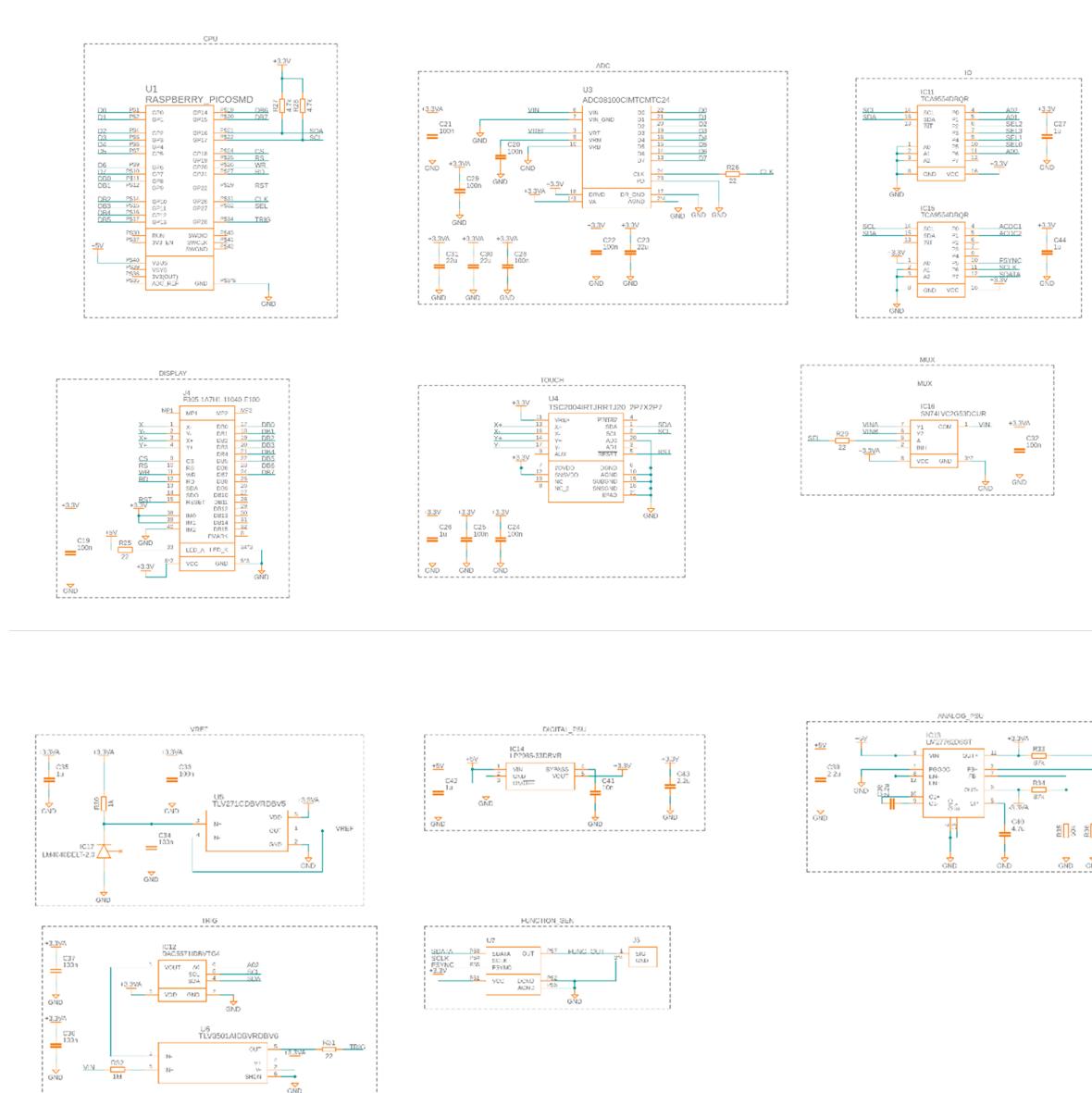


Abb. 3.10: Schaltplan

3.2 Neues Platinenlayout

Nachdem der Schaltplan feststand, konnte mit dem neuen Platinenlayout begonnen werden. Es sollten wie im Schaltplan bestimmte Funktionsgruppen in sinnvoller Anordnung zueinander platziert werden. So z.B. der Multiplexer und ADC möglichst nahe an den Analogstrecken für die Eingänge. Die Komponenten sollen sich bis auf die FPC-Buchse des Displays alle auf der Vorderseite befinden, so könnte die Platine z.B. auch ohne viel Aufwand samt Bestückung bestellt werden. Auf der Oberseite sollen Polygongüsse der jeweiligen Versorgungsspannung der Funktionsblöcke, also +3.3VA (Analog) und +3.3V (Digital), die Flächen ohne Traces ausfüllen, während die gesamte Rückseite an freien Stellen ein großer Polygonguss des GND-Potenzials sein soll.

Erster Versuch - Fehlschlag

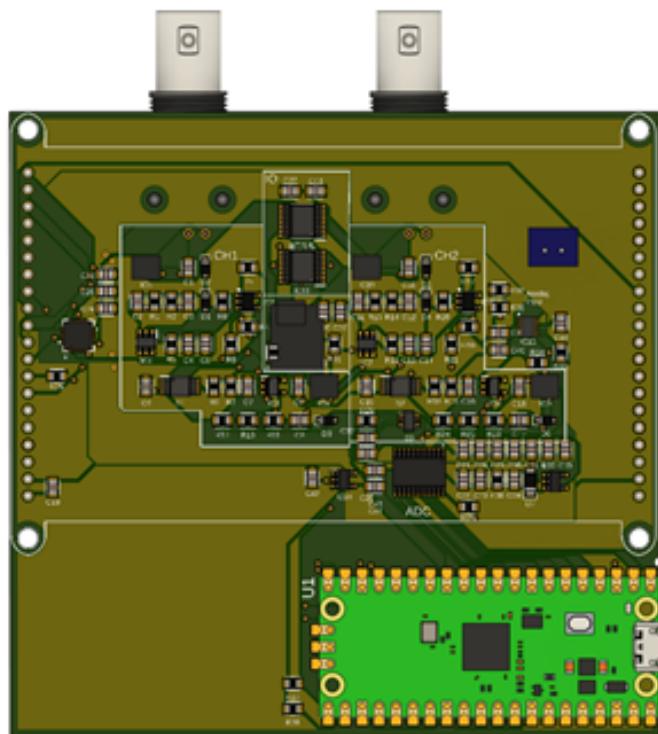


Abb. 3.11: 1. Platinendesign

Der erste Versuch endete leider in einer zu großen Platine mit teils sehr wirren Verbindungen und Anordnungen und das schon bevor alle Komponenten untergebracht waren. Ebenfalls ist noch das Adafruit Display eingeplant, welches durch seine 40 durchbohrenden Verbindungen das Layout zusätzlich erschwert. Drum wurde das erste Layout recht schnell verworfen um neu anzufangen.

3 Umsetzung

Zweiter Versuch - brauchbar

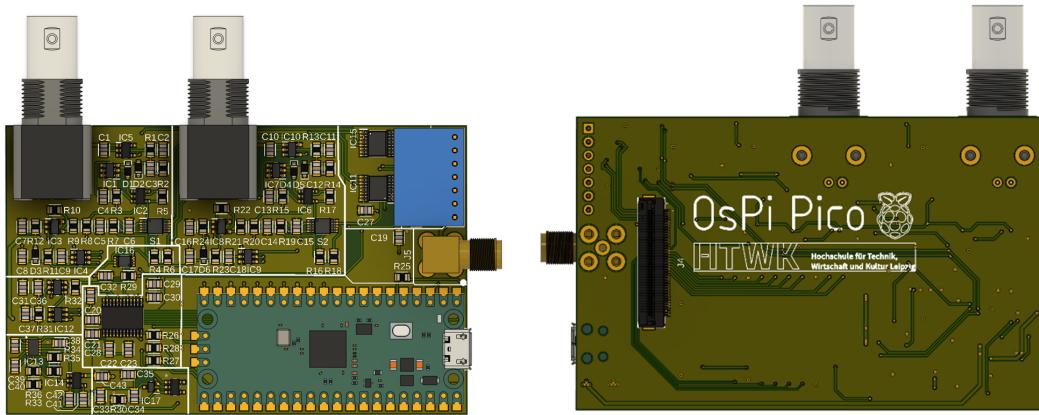


Abb. 3.12: 2.Platinendesign

Für den zweiten Versuch wurde versucht aus den Fehlern der ersten Platine zu lernen, merklich die Platine nun nur nach Bedarf zu vergrößern und alles möglichst eng aber immer noch von Hand lötbar zu platzieren. Ebenfalls half die Umstellung auf das blanke Display, was auf der Platine selbst nur den Platz für die FPC-Buche benötigt. Die weißen Linien im Bestückungsdruck trennen die einzelnen Funktionsblöcke, um die Fehlersuche zu vereinfachen. Beim blauen Rechteck handelt es sich um Das Funktionsgenerator-Board, welches über der Platine sitzt.

Alle Traces über die Analogsignale geleitet werden wurden in erhöhter 12mil Dicke ausgeführt um Leitungsverluste minimal zu halten.

3.3 Inbetriebnahme

Erster Versuch - Fehlschlag

Nachdem Platine und Komponenten angekommen waren wurden die Komponenten Stück für Stück auf die Platine aufgelötet und getestet. Meist in der Struktur eines ICs und der zugehörigen SMD-Komponenten. Leider kam es öfters zu Kurzschlüssen, was auch andere schon auf der Platine befindliche ICs beeinflusste und teilweise zerstörte. Diese wiederkehrenden Probleme ließen sich auf fehlerhafte SMD-Kondensatoren sowie Fehler beim Löten in form von überbrückten Pins zurückführen. Trotzdem musste die erste Platine verworfen werden, da durch die oben beschriebenen Fehlschläge der Pi Pico zerstört wurde und sich Komponenten fehlerhaft verhielten.

Zweiter Versuch - Funktionsfähig

Platine Nummer zwei verhielt sich deutlich vorhersehbarer und lieferte bei Überprüfung mit einem anderen Oszilloskop die erwarteten Ergebnisse, wenn auch mit einem hohen Grad an Rauschen. Trotzdem zeigte sich, dass sich die Analogstrecke im inkompletten Zustand völlig wahrlos verhält, was auch einmal zu Zerstörung des ICs für die analoge Spannungsversorgung führte. Dieser beschädigte glücklicherweise keine weiteren Komponenten und konnte ausgetauscht werden. Alle SMD-Kondensatoren wurden nun sicherheitshalber vor dem Löten auf richtigen Wert und korrekte Funktion überprüft. Ebenfalls wurde der Pi Pico nun erst auf die Platine gebracht als diese sich normal verhielt und nicht übermäßig viel Strom benötigte. Anfangs wurde erst nur ein Kanal aufgelötet, um vorerst die Funktion der Platine zusammen mit der Software verifizieren zu können. Es zeigte sich trotzdem, dass das Platinenlayout selbst nicht, wie einige Male vermutet, schuld an den anfänglichen Problemen ist. Man kann guten Gewissens alle Komponenten auf einmal auflöten und die Platine in Betrieb nehmen.

3.3.1 Programmierung des Pi Pico mit Micropython

Der nächste Schritt war nun die Software auf den Pi Pico zu bringen. Das wird im Falle von Micropython auf dem Pi Pico über die „Thonny“ IDE oder Visual Studio Code mit der „MicroPico“ Erweiterung gemacht. Bei C kann die Arduino IDE oder die PlatformIO Erweiterung in VS Code genutzt werden. Zuerst muss die Firmware von der Raspberry Pi Website heruntergeladen werden und dann auf den Pi gezogen werden. Dafür hält man den „BOOTSEL“ Knopf am Pico gedrückt, während man diesen per USB verbindet. Dieser taucht als externer Speicher auf dem PC auf, die Firmwaredatei muss nun nur auf diesen Speicher kopiert werden. Anschließend startet der Pico neu und kann nun in der Entwicklungsumgebung gefunden werden.

Einzelne .py Code-Dateien können direkt aus der IDE heraus auf dem Pico ausgeführt werden, wenn es sich aber wie in unserem Fall um mehrere Dateien handelt, müssen die Dateien mit Hilfe des in die IDE integrierten Dateibrowsers auf den Pico übertragen werden. Bei größeren Programmen hängt dieser sich gerne während des Kopierens auf, was einen Neustart und eine Neuinstallation der Firmware erforderten. Hier hat sich als wirksam herausgestellt, eine Hälfte zu übertragen, den Pico am USB-Port ab- und wieder anzustecken und dann die zweite Hälfte zu kopieren.

4 Probleme

4.1 Schwierigkeiten der Nutzung externer Bibliotheken mit Micropython

Für die Nutzeroberfläche des Oszilloskops wird die Grafikbibliothek lvgl⁹ benötigt, welche offiziell über lv_micropython¹⁰ auch die Verwendung unter Micropython unterstützt. In unserem Fall kann aber maximal eine 8.X Version von lvgl verwendet werden, da sich ab Version 9 wieder einige Codes geändert haben. Zusätzlich stellt sich die Installation mehr als kompliziert heraus. Die passende Firmware für den Pi muss vorerst selbst auf einem Linux Rechner oder auf Windows über WSL kompiliert werden. Dafür müssen folgende Schritte im Terminal durchlaufen werden:

```
1 1. git clone https://github.com/lvgl/lv_micropython.git
2 2. cd lv_micropython
3 3. git submodule update --init --recursive lib/lv_bindings
4 4. make -C ports/rp2 BOARD=PICO submodules
5 5. make -j -C mpy-cross
6 6. make -j -C ports/rp2 BOARD=PICO USER_C_MODULES=../../lib/lv_bindings/bindings.cmake
```

Die benötigte .uf2-Firmwaredatei, in der die Bibliothek integriert wurde, kann bei Erfolg anschließend im Verzeichnis lv_micropython/Ports/rp2/build-PICO/firmware.uf2 gefunden und auf den Pico kopiert werden. Dieser Prozess birgt jedoch einige Eigenheiten und neigt dazu, mit kryptischen Fehlermeldungen abzubrechen. In diesem Fall muss der Prozess komplett von neuem gestartet werden. Sollte eine Fehlermeldung fehlende Packages wie z.B. gcc-... bemängeln, so sollte vor dem Neustart das Package mit „sudo apt-get install [Packagename]“ installiert werden.

4.2 Abweichende Software

Nachdem lvgl in richtiger Version installiert werden konnte, mussten noch zwei Fehler im Display Treiber display_driver_utils.py beseitigt werden. Dort kann die Variable lv.color_t.SIZE nicht gefunden werden, da diese in einem lvgl Update entfernt wurde. Gesucht ist hier die Bittiefe der Farben für das Display, der erwartete Wert ist 16. Als die Software nun lauffähig war, stellte sich heraus, dass diese für eine Hardwarerevision 1 gedacht ist und auf diese angepasst ist. Unsere Hardware basiert auf der Revision 2 und ist somit nur begrenzt mit der Software kompatibel. Dadurch ergeben sich folgende Probleme:

Displayanbindung

Das Display ist softwareseitig über den SPI-Bus angebunden, welcher auf dieser Hardware gar nicht genutzt wird. Das Display ist stattdessen übern eine 8 Bit Parallel Anbindung nach 8080 Standard mit den Pins 8 bis 15 des Picos verbunden. Für diese Anbindung liegt kein passender Display Treiber vor.

⁹<https://github.com/lvgl/lvgl>

¹⁰https://github.com/lvgl/lv_micropython

Umsetzung des Triggers

Der Trigger ist softwareseitig mit PWM direkt über einen Pin des Picos umgesetzt. Auf dem Board ist jedoch schon eine ausgebautere Variante mit Komparator und einem eigenen DAC zur Einstellung der Spannungsschwelle verbaut. Für den DAC existiert eine Einbindung jedoch ist diese sehr simpel und enthält derzeit noch keine Möglichkeit zur Auswahl der einzelnen DACS. Von diesen befinden sich 3 identische auf dem Board (jeweils einer zur Einstellung des Offsets), was die Auswahlpins A00-A02 erfordert. Diese sind aber softwareseitig noch nicht umgesetzt.

Touch Controller

In der Software wird ein XPT2046 Touch Controller verwendet, auf dem Board befindet sich aber das Modell TSC2004 was noch stattdessen implementiert werden müsste. Möglicherweise ist auch der Code für den XPT2046 für den TSC2004 nutzbar.

4.3 Fehler im neuen Platinenlayout

Fälschliche Verbindung von RESET und RS

Beim Erstellen der neuen Platine sind ein paar Fehler nicht ausgeblieben. Fälschlicherweise wurden die beiden Steuersignale RESET und RS miteinander verbunden. Eigentlich sollte jedoch RESET einzeln zu Pin 15 am Display und Pin 5 am Touch Controller gehen. Jedoch wurden diese aufgrund von mangelndem Verständnis ebenfalls über Display Pin 10 „RS“ verbunden. Der Fehler kann durch gezielt Trennung der Leiterbahn zwischen Display und Touch Controller sowie zwischen den RESET und RS-Pin am Pico behoben werden, um zumindest die Display Funktionalität gewährleisten zu können. Jedoch geht so die RESET Funktionalität verloren, dafür müsste ein Draht gelegt werden.

Polygonguss für die Versorgungsspannung

Aufgrund eines bestimmten Bildes zu den Spannungsebenen auf der originalen Platine kam die Idee die Versorgungsspannungen +3.3VA und +3.3V auf der Oberseite der Platine als Polygonguss umzusetzen. Damit wird die gesamte markierte Kupferfläche als das jeweilige Potenzial umgesetzt. Der Gedanke dahinter war weniger Leiterbahnen setzen zu müssen, da die Pins für Versorgungsspannung der ICs so direkt verbunden wären. Das Problem was sich daraus ergibt ist, dass der Guss so mit dem auf der Unterseite gegenüber liegendem GND-Guss einen Plattenkondensator bildet, was das Verhalten der Spannungsversorgung negativ beeinflusst. Zusätzlich werden so die Potenziale +3.3VA und +3.3V über GND kapazitiv gekoppelt, was auch zu deutlich mehr Rauschen in den Spannungsebenen führt. Zusätzlich fängt der große Guss auch noch mehr Störungen ein als es Leiterbahnen tun würden. Verbindungen wurden so auch nicht eingespart, da es fast genauso viele GND-Pins an den Bauteilen gibt, welche stattdessen manuell verbunden werden mussten.

Für zukünftige Layouts sollten auf der Vorder- und Rückseite leere Flächen nur mit GND-Güssen gefüllt werden. Das verbessert die Stabilität des GND-Potenzials, spart auch so Leiterbahnen und vermeidet zusätzliche Einkopplungen. +3.3VA und +3.3V sollten genauso wie beispielsweise -3.3VA auf der Platine durch Leiterbahnen verteilt werden.

4 Probleme

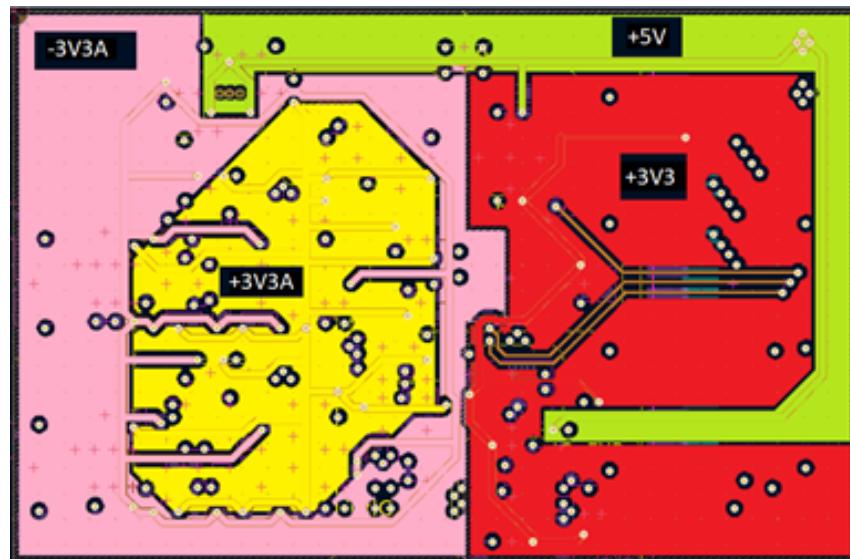


Abb. 4.1: Verbesserung GND-Problem

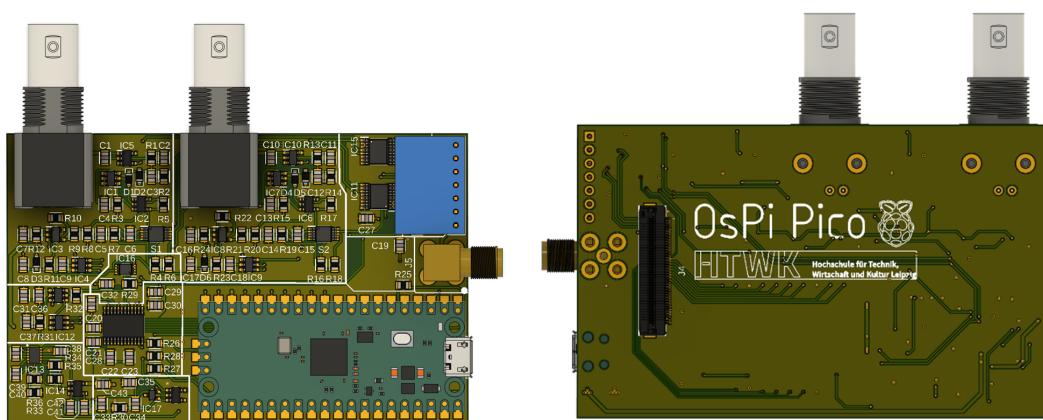


Abb. 4.2: 3.Platinendesign

5 Mögliche Problemlösungen

5.1 Displayansteuerung

Um das Display korrekt nutzen zu können, könnten zwei Herangehensweisen genutzt werden:

Neuer Displaytreiber

Damit das Display über den 8 Bit 8080 Parallel-Port genutzt werden kann, muss ein passender Display Treiber gefunden und angepasst / geschrieben werden. Vorteil davon wäre, dass das Display über diese Schnittstelle deutlich schneller angesteuert werden kann als über SPI, da auch die 8-fache Bandbreite zur Verfügung steht. So ließen sich in etwa 60 Bilder/Sekunde realisieren, das Display unterstützt maximal 70.

Anbindung auf SPI umbauen

Durch Drähte oder eine neue Platine könnte das Display wieder, wie in der Software spezifiziert per SPI angebunden werden. Hierfür würden die Pins am Pico frei werden, welche vorher für die parallele Übertragung genutzt wurden. Daraus würde sich folgende Belegung am Display ergeben:

5 Mögliche Problemlösungen

Gerät	Pin	Funktion	Display Pin	Name	Bemerkung
Touch	16	X-	1	XL(X-)	
Touch	17	Y-	2	YU(Y-)	
Touch	13	X+	3	XR(X+)	
Touch	14	Y+	4	YD(Y+)	
PSU	-	GND	5	GND	
PSU	-	+3.3V	6	VCC	
PSU	-	+3.3V	7	VCC	
-	-	-	8	FMARK	
Pico	9	CS	9	CS	
Pico	10	SCL	10	RS/SCL	
Pico	8	AO/DC	11	WR/AO	
-	-	-	12	RD	
Pico	11	MOSI	13	SDA	
Pico	12	MISO	14	SDO	
Pico	22	RST	15	RESET	
PSU	-	GND	16	GND	
-	-	-	17-32	DB0-15	Parallel Ansteuerung
PSU	22R	Backlight	33	A	Anode LED Backlight
PSU	-	GND	34	K1	Kathode LED Backlight
PSU	-	GND	35	K2	Kathode LED Backlight
PSU	-	GND	36	K3	Kathode LED Backlight
PSU	-	GND	37	GND	
PSU	-	+3.3V	38	IM0	Input Mode: setzt genutzten Eingang fest, hier 4 Wire SPI
PSU	-	+3.3V	39	IM1	
PSU	-	+3.3V	40	IM2	

Im Vergleich zur parallelen Anbindung kann das Display nicht so schnell aktualisiert werden, jedoch sollte ein Oszilloskop auch mit grade mal 10 Bildern/Sekunde¹¹ nutzbar sein. Nach eigenem Empfinden würde die Pin-Ersparnis und die Verfügbarkeit fertiger Bibliotheken hier zum Vorteil von SPI überwiegen.

¹¹https://www.reddit.com/r/esp32/comments/r5awqp/made_a_refresh_rate_test_comparison_between_the/

5.2 Softwareanpassung

Trigger

Damit der Trigger wie gewollt funktioniert, muss der PWM-Wert, der von der Scope Funktion ausgegeben wird in den passenden Wert für den DAC5571 IC in der Trigger Schaltung gewandelt werden. Schließlich muss dieser Wert noch auf den IC übertragen werden, indem der Adress-Pin A02 HIGH gesetzt werden und die passenden Daten über I2C ausgegeben werden. Währenddessen müssen die Pins A00 und A01 der identischen DACs für den Offset der Eingänge LOW gezogen werden.

Die genaue Adressierung kann auf Seite 15 des Datenblatts¹² nachgelesen werden.

5.3 Verbesserung des Platinenlayouts

Behebung RS/RST Verbindung

Um die Reset-Verbindung korrekt umzusetzen, wird diese vom RS-Potenzial getrennt und eine eigene Leiterbahn gelegt. Das wurde bereits in der aktuellen Version des Layouts korrigiert.

+3.3VA und +3.3V Güsse durch GND-Guss ersetzen

Für eine neue Revision der Platine sollten die Polygongüsse für +3.3VA und +3.3V auf der Oberseite entfernt und durch GND-Güsse ersetzt werden. Dadurch können einige Vias und GND-Leiterbahnen entfernt werden. Dafür müssen nun für die Spannungsversorgung neue Leiterbahnen verlegt werden.

Montagepunkte

Ein kleines, aber nicht unwichtiges Manko der aktuellen Revision sind fehlende Montagepunkte, wie z.B. Löcher für Schrauben/Abstandshalter. Solche Merkmale sollten hinzugefügt werden.

¹²<https://www.ti.com/lit/ds/symlink/dac5571.pdf>

6 Fazit

6.1 Stand zu Projektende

Am Ende des Projekts ist das Gerät leider nicht funktionsfähig. Dies ist auf eine Kombination von Hardwareproblemen zurückzuführen, sowie darauf, dass die aktuelle Software in ihrer gegenwärtigen Form nicht mit der Platine kompatibel ist. Bis zur Präsentation wird jedoch versucht, zumindest das Display mithilfe eines neuen C-Codes zum Anzeigen zu bringen. Es existiert eine Platine, auf der alles bis auf den zweiten Eingangskanal aufgebaut und funktionsfähig ist.

6.2 Offene Punkte

- Displayansteuerung durch passenden Displaytreiber oder Hardwareänderung auf SPI realisieren
- Frequenzgenerator per SPI anbinden, bzw. wenn das Display parallel laufen soll Alternative finden oder komplett weglassen
- Berechnung von RMS und Frequenzwerten in der Benutzeroberfläche
- Firmware verstehen und anpassen oder bestenfalls in C neu schreiben
- Montagepunkte zur Platine hinzufügen
- Gehäuse für das Gerät

6.3 Eigene Betrachtung

Am Ende des Projekts wird deutlich, dass es leider als Misserfolg betrachtet werden muss. Das Gerät ist nicht funktionsfähig, und die Software lässt sich kurz gesagt überhaupt nicht nutzen. Obwohl das Projekt zeitlich knapp gestartet wurde und der Jahreswechsel eine Rolle spielte, liegt das Scheitern auch teilweise daran, dass wir uns zu sehr auf ein vorhandenes Projekt als Vorbild verlassen haben.

Ursprünglich war die Idee, eine eigene Platine zu entwickeln, diese schnell in Betrieb zu nehmen und dann die Software an unsere Bedürfnisse anzupassen. Jedoch verlief alles anders als geplant. Wir hatten von Anfang an Schwierigkeiten mit der Inbetriebnahme der Platine, was die anschließenden Softwareprobleme noch verschlimmerte. Dies führte zu einem Rückgang der Motivation und des Teamgeistes. Was waren die Hauptgründe dafür?

6 Fazit

Das Vorbild-Projekt hat uns fälschlicherweise die Sicherheit vermittelt, dass wir alles so übernehmen könnten, wie es ist, und es einfach funktionieren würde. Darüber hinaus haben wir uns in ein recht komplexes Geflecht aus Hard- und Software vertieft, das wir zunächst vollständig hätten verstehen müssen - was in unserem Fall bisher nicht eingetreten ist.

Es wäre besser gewesen, ein einfacheres Design zu entwickeln, das zwar weniger leistungsstark gewesen wäre, aber einfacher zu verstehen gewesen wäre. Anschließend hätten wir die Firmware von Grund auf selbst schreiben können. Trotzdem sollte es mit der vorhandenen Arbeit möglich sein, ein durchaus brauchbares Oszilloskop zu realisieren.

Trotz des Scheiterns hat das Projekt wertvolles Know-how vermittelt. Zum Beispiel war es das erste Mal, dass wir ein eigenes, richtiges Platinenlayout erstellt haben, das wir dann auch selbst herstellen und bestücken konnten. Dies erforderte eine Einarbeitung in den Elektronik-Arbeitsbereich von Fusion 360 und das Erlernen seiner Eigenheiten.

Zusätzlich war es das erste Mal, dass wir mit einem Raspberry Pi Pico und Micropython gearbeitet haben, was eine interessante Alternative zum typischen „C auf Arduino“ Ökosystem darstellt. Allerdings müssen wir zugeben, dass Micropython in Bezug auf Benutzerfreundlichkeit und Bibliotheken noch deutlich ausgebaut werden muss. Das Einbinden der LVGL-Bibliothek war im Vergleich zu Arduino IDE ein Albtraum - dort ist sie nur durch drei Knopfdrücke erreichbar. Einige könnten argumentieren, dass dies zur Erfahrung in der Embedded-Entwicklung gehört. Dennoch hätte sich dies nicht als Unterpunkt während der bereits stressigen Inbetriebnahme-Phase manifestieren sollen.

Insgesamt ist der Umfang des Projekts etwas außer Kontrolle geraten, was dazu führte, dass die Zeit immer knapper wurde. Trotzdem gab es definitiv einen großen Lerneffekt. Es ist bedauerlich, dass am Ende nicht einmal ein halb funktionales Gerät dabei herausgekommen ist.