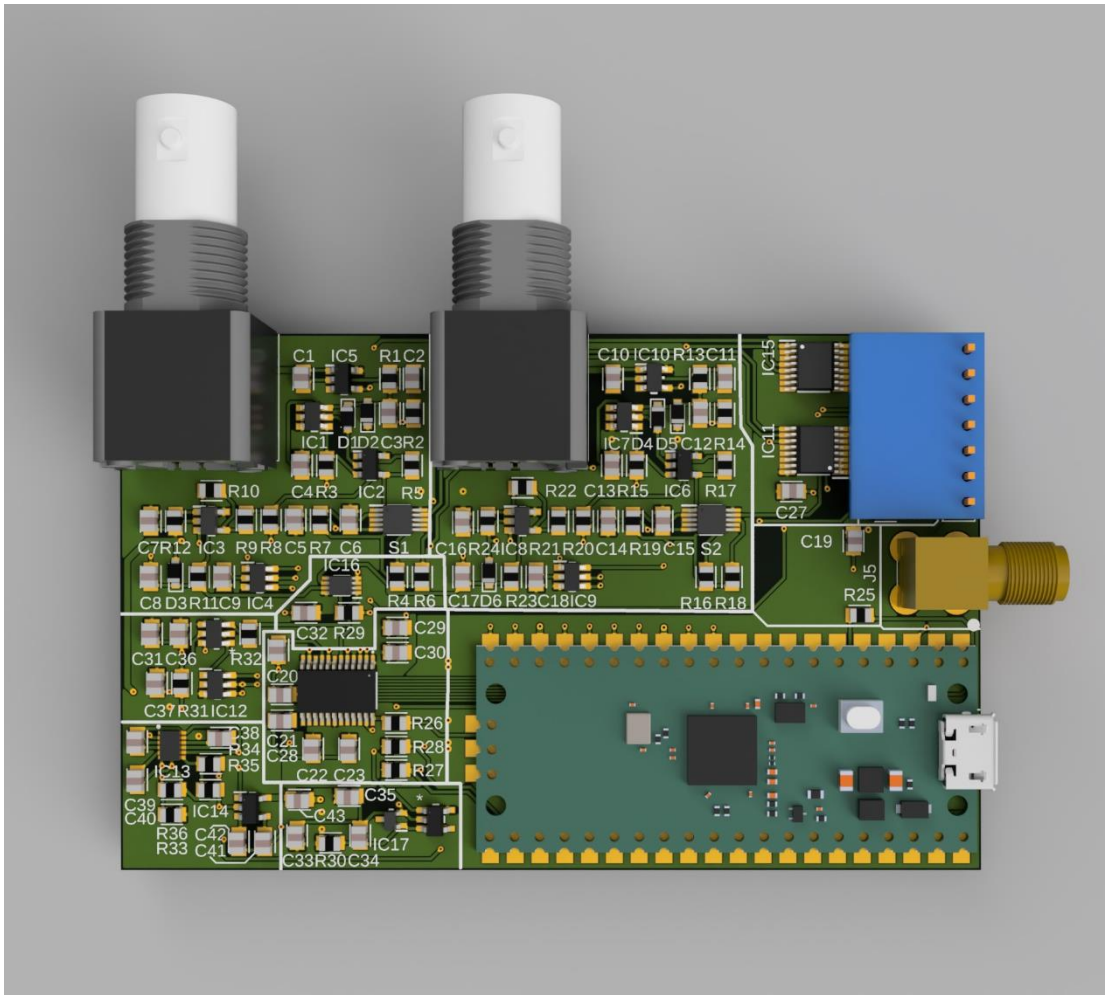
PROJEKT OSZI

„OsPi Pico“ Version 1

Projektdokumentation



Embedded Systems I Projekt 2024

Johannes Pautz, Max Sämann

20EIK-IAS

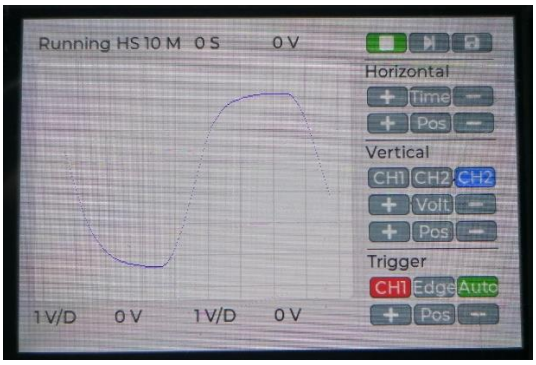
GitHub - Repository

Inhaltsverzeichnis

1. Kenndaten

1.1 Grundlage: Projekt „RPScope“

Als Grundstein für das Projekt diente das Projekt „RPScope“ von „jpgeiro“, welches auf Hackaday.io veröffentlicht wurde. Von hier wurde die Software, sowie der Schaltplan übernommen und überarbeitet. Viele unbekannte Punkte sowie nähere Erklärungen zu Software und Hardware Aufbau sind dort zu finden.



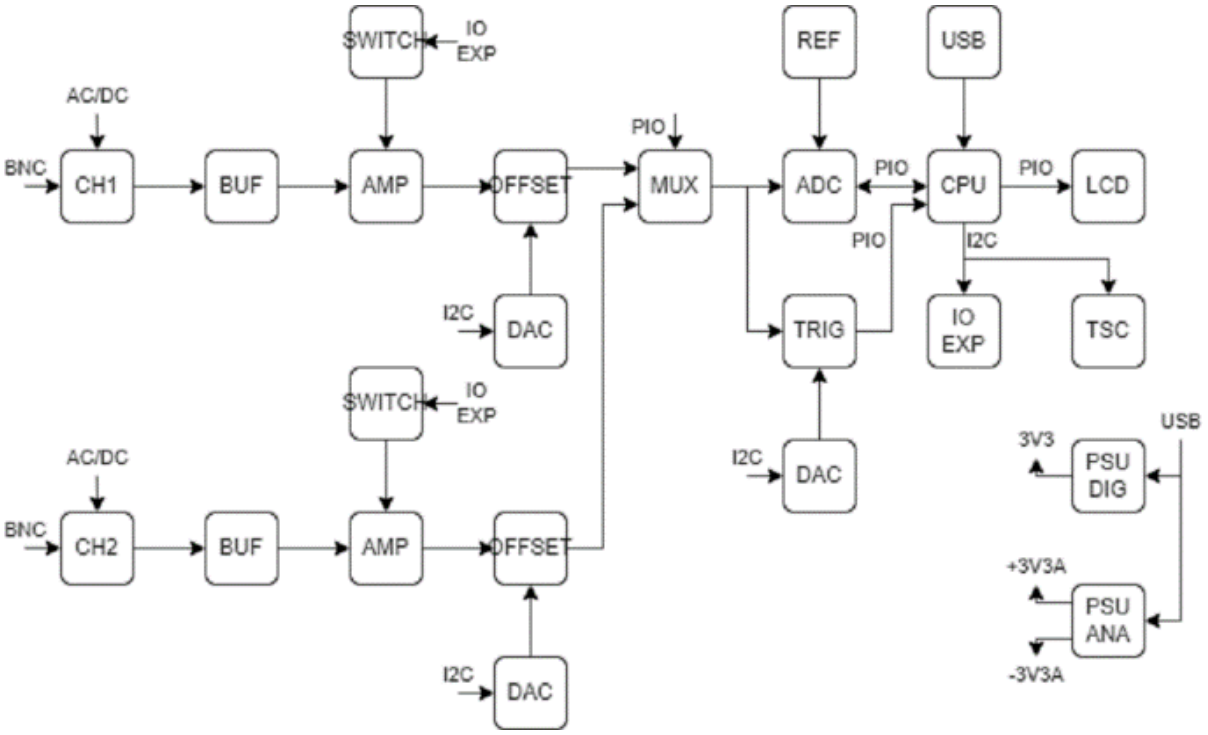
1.2 Eigenschaften der „OsPi Pico“ Platine

- 2 Eingangskanäle mit 100MHz Analog-Bandbreite
- 100Msps ADC mit 8 Bit Auflösung
- 10MHz zur Messung nutzbare Bandbreite pro Kanal
- ±27V Eingangsbereich
- Bedienung über 3,5 Zoll Touch Display
- Stromversorgung über Micro USB (USB-C mit anderem Pi Pico möglich)

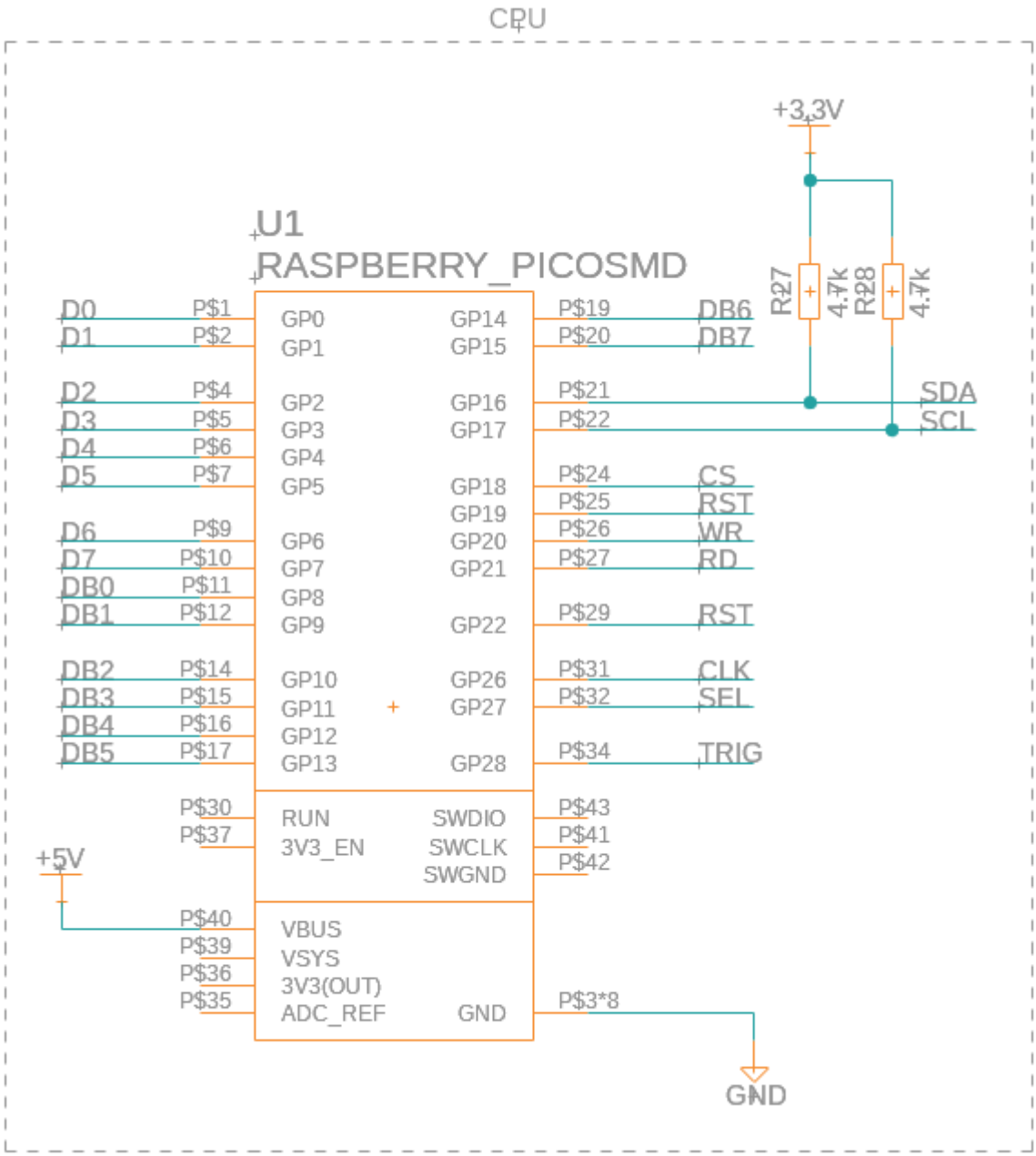
2. Aufbau

2.1 Schaltplan

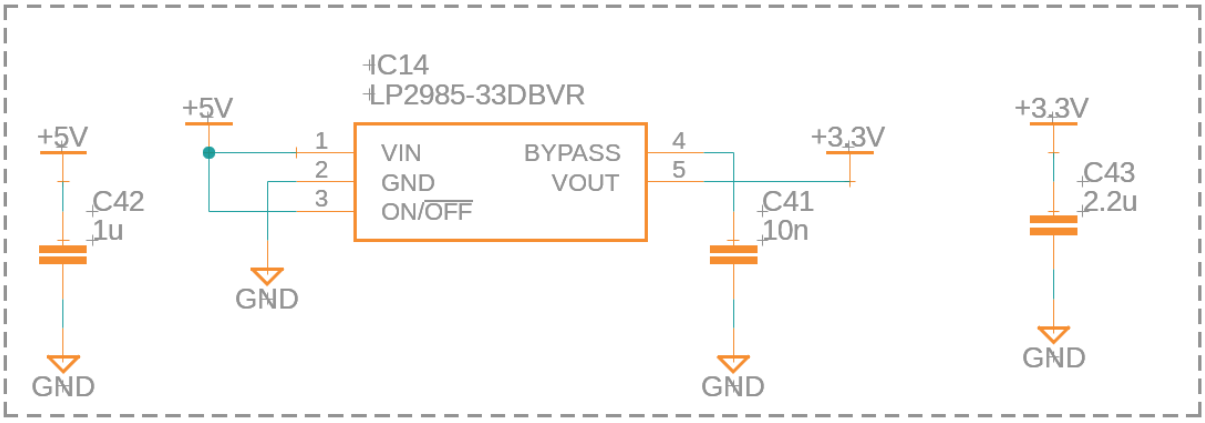
Die Struktur des Gerätes ist wie folgend aufgebaut:



2.1.1 Belegung Pi Pico



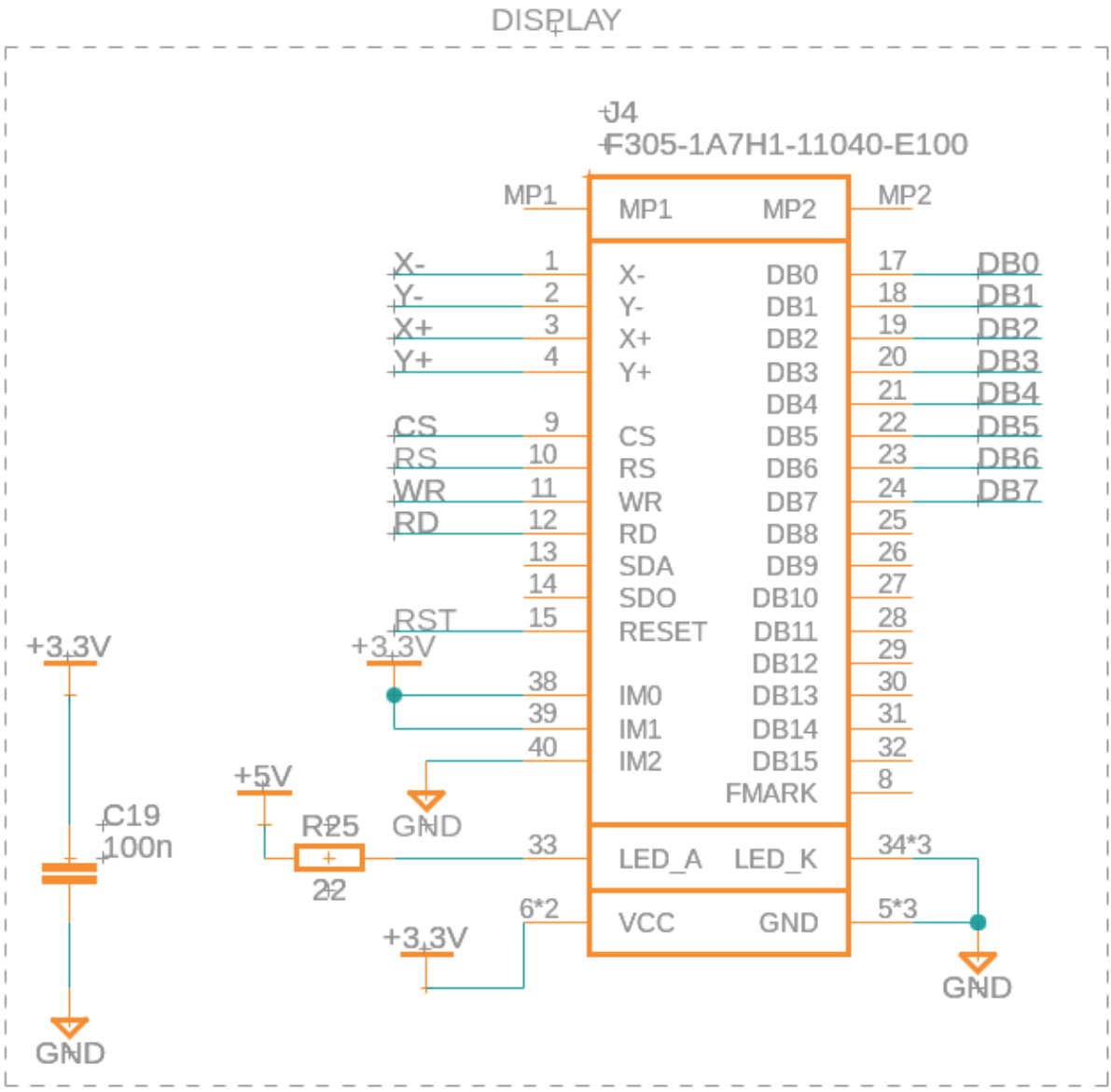
2.1.2 Stromversorgung Digital



Die Stromversorgung für den digitalen Teil wird durch einen einfachen Spannungsregler umgesetzt. +5V wird durch den USB-Port am Pi Pico gespeist. Die Kondensatoren C42 und C43 dienen zu Stabilisierung.

C43 kann bei Problemen auch entfernt werden. In unseren Versuchen stellte sich dieser Kondensator nach einiger Zeit als kaputt heraus, was teils bis zum Kurzschluss von +3.3V und somit auch zur Zerstörung des Spannungsreglers folgen kann. Eine Ursache konnte bisher noch nicht ermittelt werden.

2.1.3 Display



Das Display ist ein 3,5 Zoll großes resistives Touchdisplay mit einer Auflösung von 480x320 Pixeln. Integriert ist der ILI9488 Displaytreiber. Zu finden ist das Modell auf Aliexpress.

Das Display ist per 8 Bit parallel Bus nach 8080 Standard an den Pi Pico angebunden. Die Pins 9 bis 12 dienen zur Steuerung des Datenverkehrs.

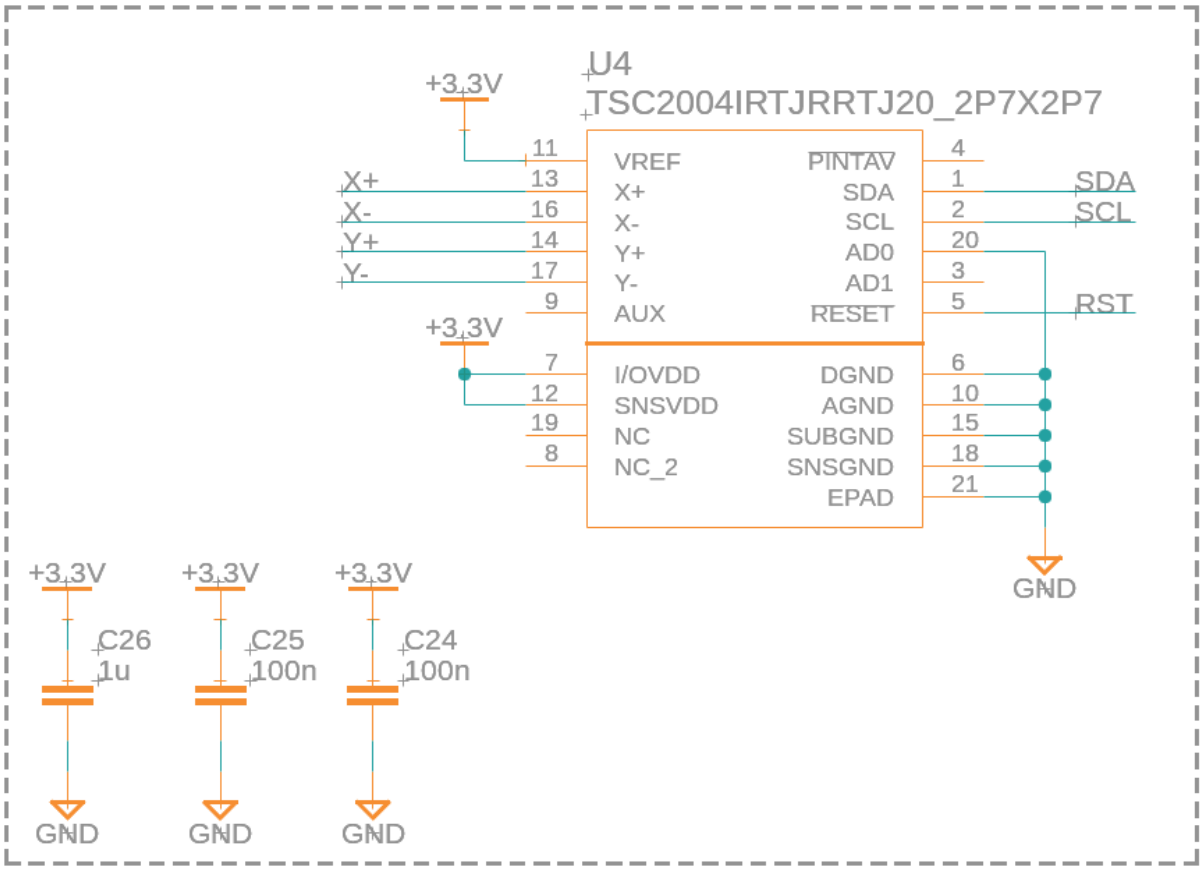
Leider existiert bisher keine passende Anbindung in der Software für das 8 Bit Interface. Ebenfalls ist auf Charge 1 der Platinen eine Fehlerhafte Verbindung zwischen RST und RS vorhanden.

Näheres zu den Problemen und möglichen Lösungen ist in Kapitel 4 nachzulesen.

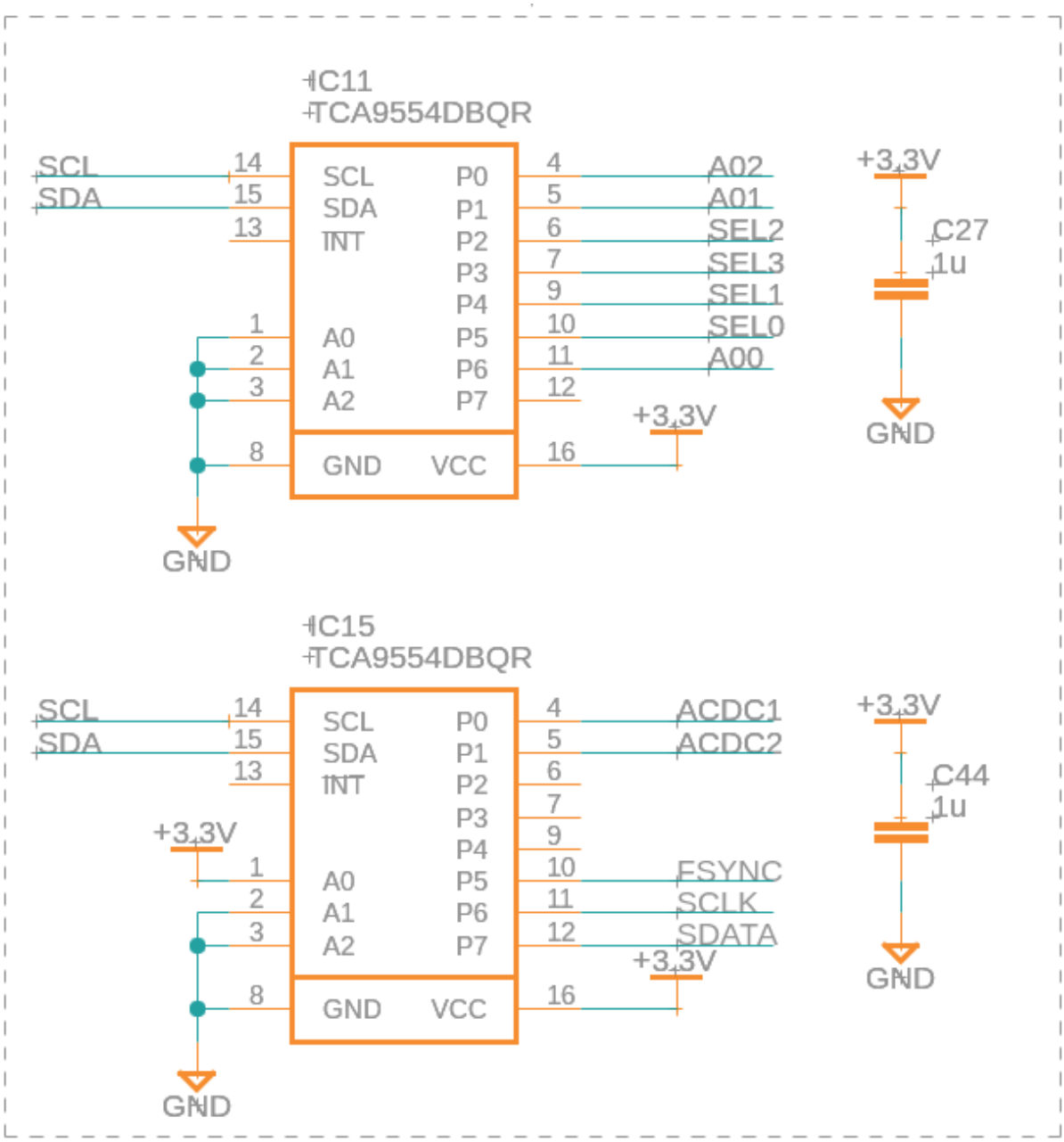
Gerät	Pin	Funktion	Display Pin	Name	Bemerkung
Touch	16	X-	1	XL(X-)	
Touch	17	Y-	2	YU(Y-)	
Touch	13	X+	3	XR(X+)	
Touch	14	Y+	4	YD(Y+)	
PSU[1]	-	GND	5	GND	
PSU	-	+3.3V	6	VCC	
PSU	-	+3.3V	7	VCC	
-	-	-	8	FMARK	
Pico	18	CS	9	CS	
Pico	19	RS	10	RS/SCL	
Pico	20	WR	11	WR/AO	
Pico	21	RD	12	RD	
-	-	-	13	SDA	
-	-	-	14	SDO	
Pico	22	RST	15	RESET	
PSU	-	GND	16	GND	
Pico	8-15	DB0-7	17-32	DB0-15	Parallel Ansteuerung
PSU	+5V, 22R	Backlight	33	A	Anode LED Backlight, +5V durch 22R Wiederstand
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 8 Bit Parallel
PSU	-	+3.3V	39	IM1	
PSU	-	GND	40	IM2	

[1] Spannungsversorgung

2.1.4 Touch-Controller

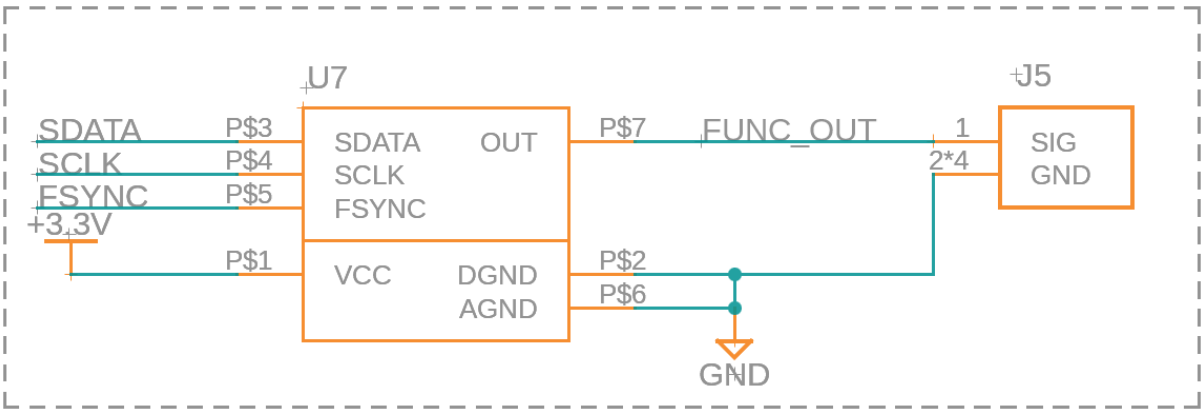


2.1.5 IO-Expander



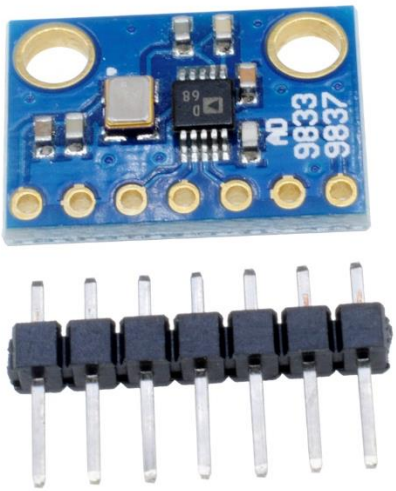
Der zweite IO-Expander wird im Grunde nur für die AC/DC Umschaltung benötigt. Die Anbindungen FSYNC, SCLK und SDATA vom Funktionsgenerator funktionieren so nicht un müssten umgelegt werden, wie in Kapitel 4 beschrieben.

2.1.6 Funktionsgenerator

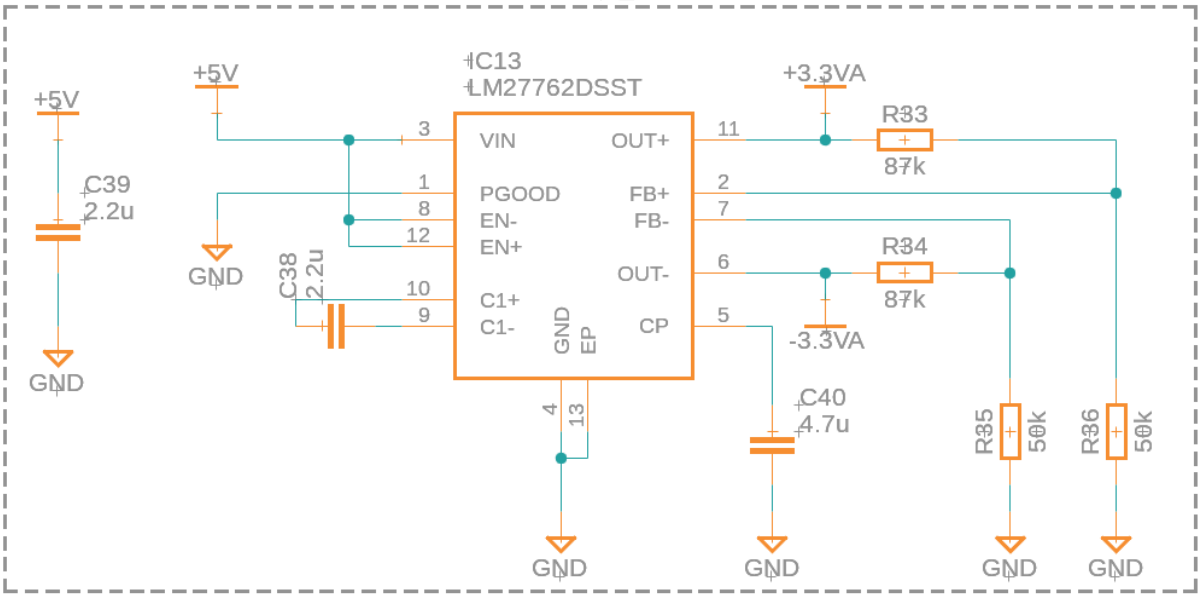


Der Funktionsgenerator basiert auf einer fertigen Platine mit dem Analog Devices AD9833 IC. Er kann über 3-wire SPI angesteuert werden.

Die Platine selbst wird mit Hilfe einer Steckleiste über der Hauptplatine angebracht. Ein passendes Board kann auf eBay erworben werden. Da es sich um kein Modell eines wirklich namhaften Herstellers handelt kann folgendes Bild zusammen mit der Modellnummer des ICs als Referenz verwendet werden.

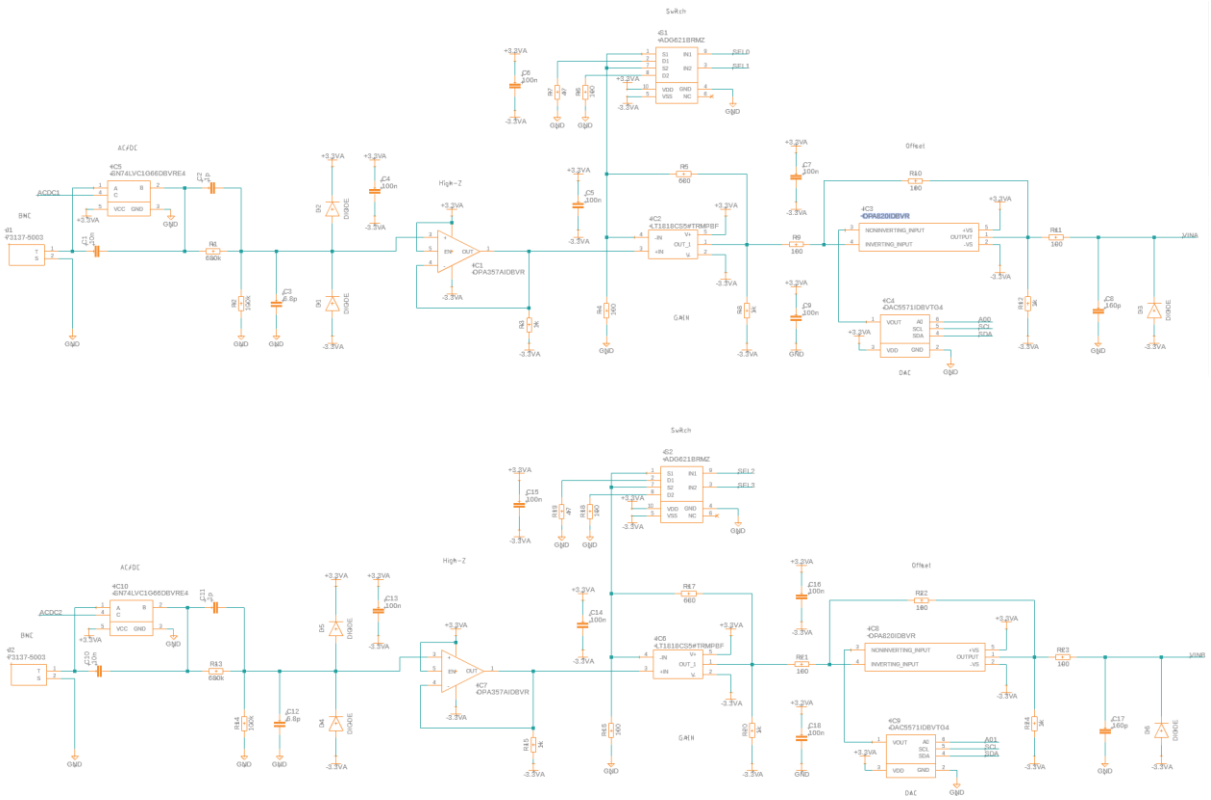


2.1.7 Stromversorgung analog



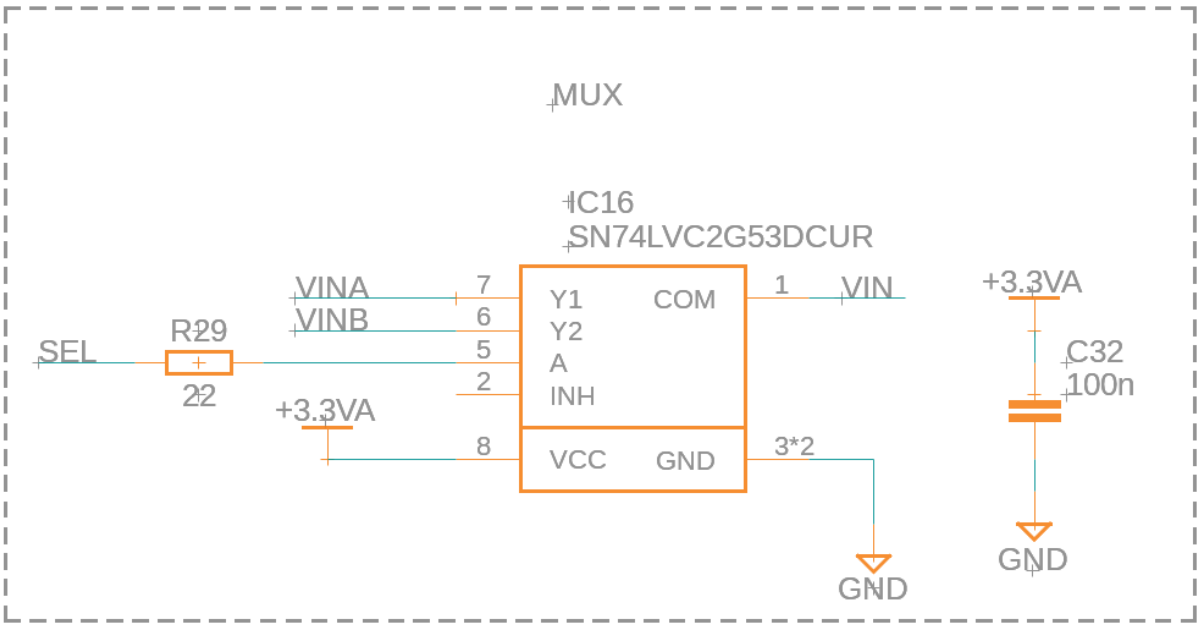
Die Analoge Stromversorgung wird durch den LM27762 IC ermöglicht. Dieser kann aus der +5VDC Versorgungsspannung sowohl +3.3V als aus -3.3V generieren und ermöglicht so die einfache Nutzung traditioneller OPVs.

2.1.8 Eingangskanäle



Die Eingangskanäle bestehen aus 4 Abschnitten. Zuerst kommt die Einkopplung, hier kann auch durch die Analogschalter zwischen AC und DC-Kopplung gewechselt werden. Anschließend kommt die „High-Z“ Stufe, welche dazu dient, die Eingangsimpedanz heraufzusetzen. So werden die Messstrecken nicht durch zusätzliche Stromflüsse belastet, was die Messergebnisse verfälschen würde. Darauf folgt die Gain-Stufe welche je nach gewünschtem Eingangsbereich das Signal „verkleinert“. Die Verstärkung wird durch den Analogschalter oben eingestellt, welcher verschiedene Spannungsteiler darstellt. Am Ende sitzt die Offset-Stufe, hier wird dem Signal eine DC Spannung hinzu gefügt, damit das letztendliche Signal dem 0-VREF Eingangsbereich des ADC entspricht.

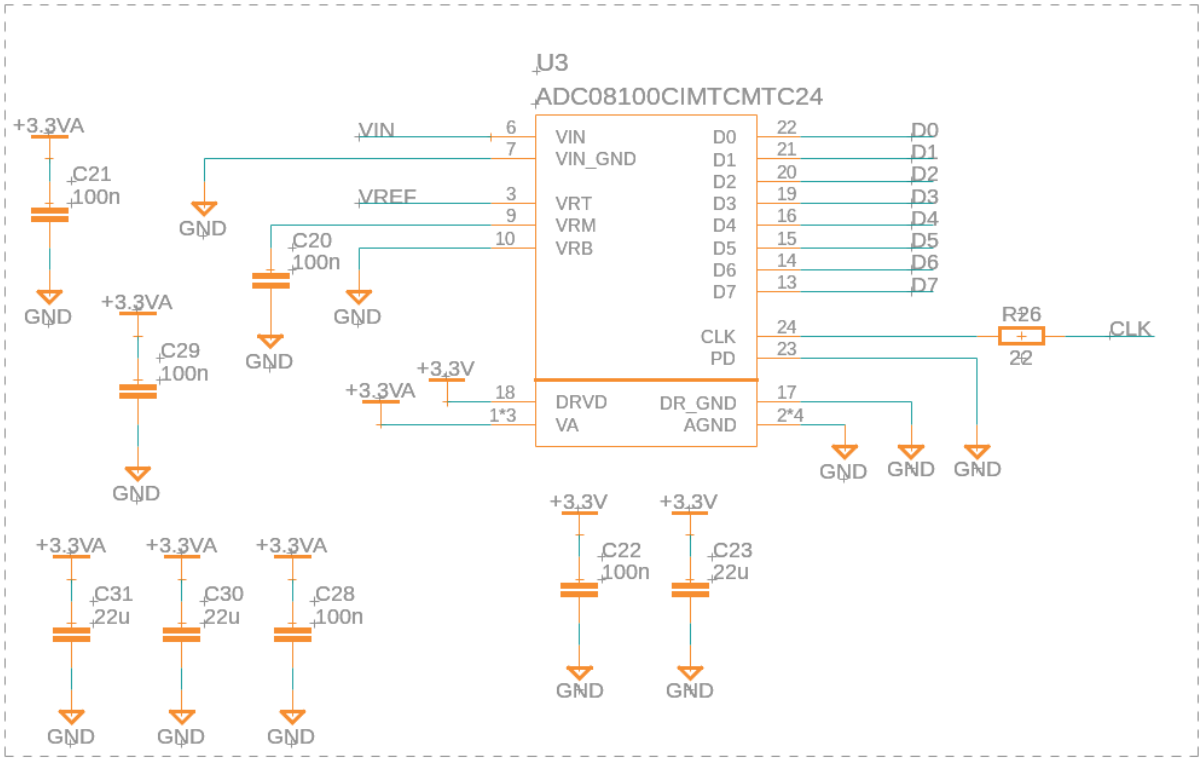
2.1.9 Multiplexer



Normal leitet der Multiplexer Kanal Y1 / VINA auf den Ausgang COM weiter. Liegt SEL am Steuereingang A an, so wird Y1 von COM getrennt und stattdessen Kanal Y2 / VINB auf COM gelegt.

In Betrieb passiert dies blitzschnell um jeweils abwechselnd die Kanäle mit dem ADC samplen zu können.

2.1.10 ADC

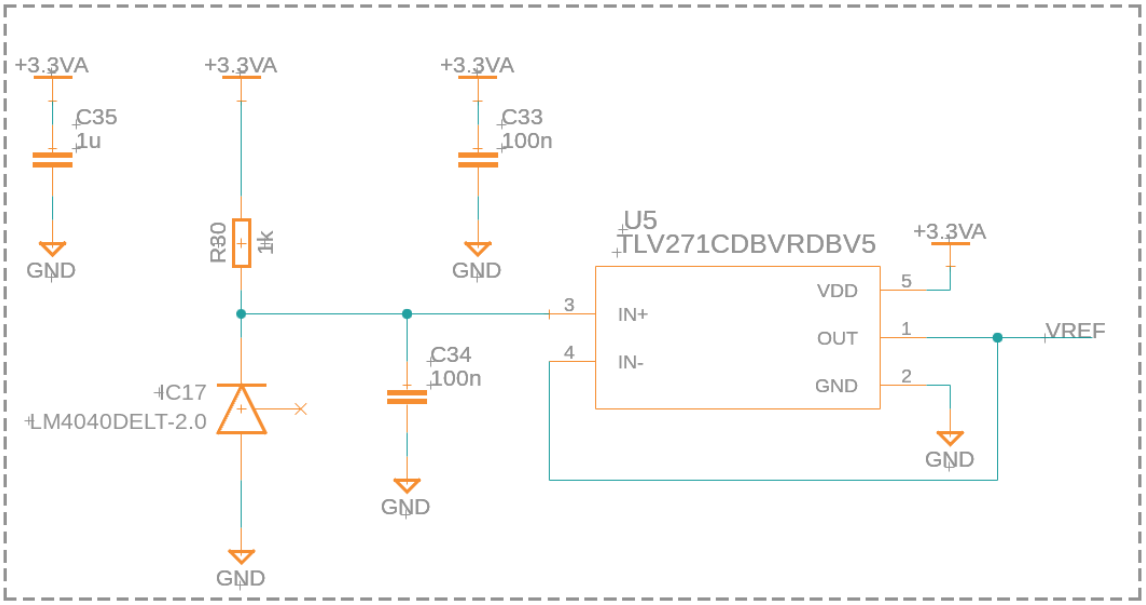


Der ADC wertet jeweils seinen VIN-Eingang aus, nachdem CLK (vom Pi kommend) von HIGH auf LOW fällt. Bei voller Sample Rate liegt der passende Digitalwert zum VIN Eingang nach 3,5 Samples Verzögerung an den Ausgangspins an.

Der Eingangsbereich beträgt 0V-VREF, in unserem Fall also 0-2V. Das bedeutet, dass das Eingangssignal vorher durch die Offset-Stufe im Analogeingang verschoben werden muss. 0V im Eingangssignal muss zu 1V am ADC Eingang werden.

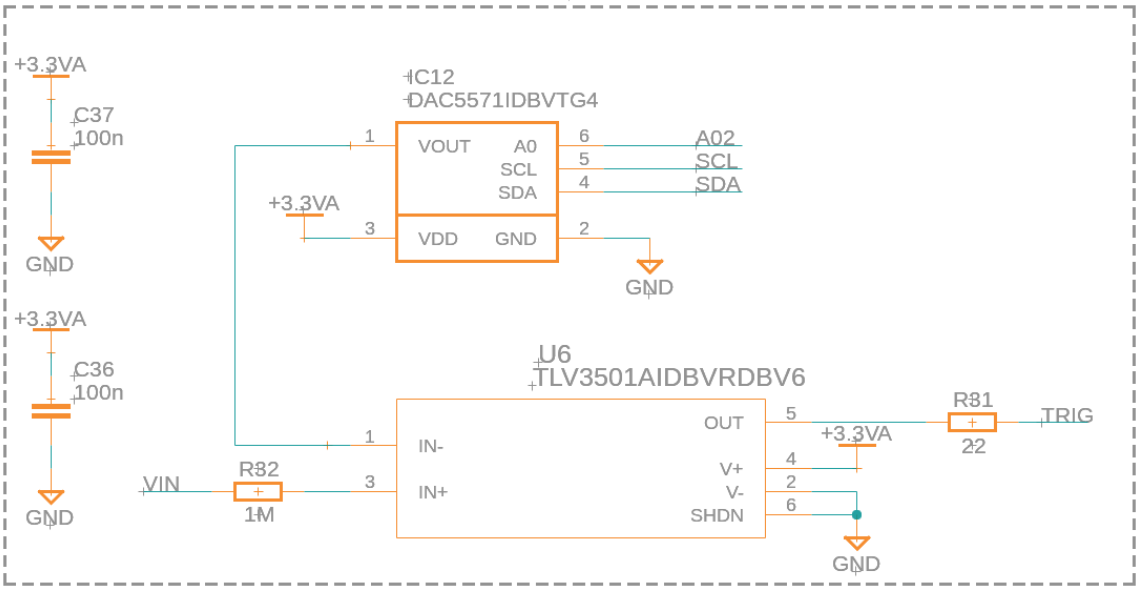
Die Daten des ADCs werden über 8 Bit parallel Bus (1 Pin für jedes Bit an Auflösung) an den Pi Pico übertragen. Das ermöglicht uns deutlich höheren Durchsatz als zum Beispiel bei I2C oder SPI.

2.1.11 Referenzspannungsquelle



Die Referenzspannung wird mit Hilfe einer hochgenauen Z-Diode erzeugt. Da diese Referenzspannung aber bei Belastung einbrechen würde, wird ein OPV als Impedanzwandler genutzt, um den Strom über die Z-Diode möglichst nah an 0 zu halten.

2.1.12 Trigger



Das Herzstück des Triggers ist der TLV3501 Highspeed-Komparator. Er gibt über das TRIG-Signal ein HIGH Pegel an den Pi Pico, wenn die Eingestellte Spannung im Eingangssignal erreicht wird. Der Spannungspegel wird durch einen per I2C an den Pico angebundenen DAC eingestellt.

Für VIN sollte VINA verwendet werden, das ist ein Fehler, der beim Layout unterlaufen ist, mehr dazu in Kapitel 4.

2.2 Platine

2.2.1 Materialliste

Für das Gerät ergibt sich abgesehen von der Platine selbst und dem Display folgende Materialliste:

Menge	Wert	Bauteil
10	100	R_CHIP-0805(2012-METRIC)
2	100k	R_CHIP-0805(2012-METRIC)
23	100n	C_CHIP-0805(2012-METRIC)
3	10n	C_CHIP-0805(2012-METRIC)
2	160p	C_CHIP-0805(2012-METRIC)
1	1M	R_CHIP-0805(2012-METRIC)
7	1k	R_CHIP-0805(2012-METRIC)
2	1p	C_CHIP-0805(2012-METRIC)
5	1u	C_CHIP-0805(2012-METRIC)
3	2.2u	C_CHIP-0805(2012-METRIC)
4	22	R_CHIP-0805(2012-METRIC)
3	22u	C_CHIP-0805(2012-METRIC)
2	4.7k	R_CHIP-0805(2012-METRIC)
1	4.7u	C_CHIP-0805(2012-METRIC)
2	47	R_CHIP-0805(2012-METRIC)
2	50k	R_CHIP-0805(2012-METRIC)
2	6.8p	C_CHIP-0805(2012-METRIC)
2	680	R_CHIP-0805(2012-METRIC)
2	680k	R_CHIP-0805(2012-METRIC)
2	73137-5003	73137-5003
2	87k	R_CHIP-0805(2012-METRIC)
1	AD9833FUNCGEN	AD9833FUNCGEN
1	ADC08100CIMTCMTC24	ADC08100CIMTCMTC24
2	ADG621BRMZ	ADG621BRMZ
3	DAC5571IDBVTG4	DAC5571IDBVTG4
6	DIODE	DIODE_DO-219-AC(SOD323F)
1	F305-1A7H1-11040-E100	F305-1A7H1-11040-E100
1	LM27762DSST	LM27762DSST
1	LM4040DELT-2.0	LM4040DELT-2.0
1	LP2985-33DBVR	LP2985-33DBVR
2	LT1818CS5#TRMPBF	LT1818CS5#TRMPBF
2	OPA357AIDBVR	OPA357AIDBVR
2	OPA820IDBVR	OPA820IDBVR
1	PCB.SMAFRA.HT	PCB.SMAFRA.HT
1	RASPBERRY_PICOSMD	RASPBERRY_PICOSMD
2	SN74LVC1G66DBVRE4	SN74LVC1G66DBVRE4
1	SN74LVC2G53DCUR	SN74LVC2G53DCUR
2	TCA9554DBQR	TCA9554DBQR
1	TLV271CDBVRDBV5	TLV271CDBVRDBV5
1	TLV3501AIDBVRDBV6	TLV3501AIDBVRDBV6
1	TSC2004IRTJRRTJ20_2P7X2P7	TSC2004IRTJRRTJ20_2P7X2P7

Alle nicht-Standard SMD-Komponenten sind zur einfacheren Bestellung als Projektliste auf Mouser verfügbar:
https://www.mouser.com/ProjectManager/ProjectDetail.aspx?AccessID=aab402505e

2.2.2 Bestückung

Die Bestückung der einzelnen Komponenten ist wie folgt:

Part	Value	Device
C1	10n	C_CHIP-0805(2012-METRIC)
C2	1p	C_CHIP-0805(2012-METRIC)
C3	6.8p	C_CHIP-0805(2012-METRIC)
C4	100n	C_CHIP-0805(2012-METRIC)
C5	100n	C_CHIP-0805(2012-METRIC)
C6	100n	C_CHIP-0805(2012-METRIC)
C7	100n	C_CHIP-0805(2012-METRIC)
C8	160p	C_CHIP-0805(2012-METRIC)
C9	100n	C_CHIP-0805(2012-METRIC)
C10	10n	C_CHIP-0805(2012-METRIC)
C11	1p	C_CHIP-0805(2012-METRIC)
C12	6.8p	C_CHIP-0805(2012-METRIC)
C13	100n	C_CHIP-0805(2012-METRIC)
C14	100n	C_CHIP-0805(2012-METRIC)
C15	100n	C_CHIP-0805(2012-METRIC)
C16	100n	C_CHIP-0805(2012-METRIC)
C17	160p	C_CHIP-0805(2012-METRIC)
C18	100n	C_CHIP-0805(2012-METRIC)
C19	100n	C_CHIP-0805(2012-METRIC)
C20	100n	C_CHIP-0805(2012-METRIC)
C21	100n	C_CHIP-0805(2012-METRIC)
C22	100n	C_CHIP-0805(2012-METRIC)
C23	22u	C_CHIP-0805(2012-METRIC)
C24	100n	C_CHIP-0805(2012-METRIC)
C25	100n	C_CHIP-0805(2012-METRIC)
C26	1u	C_CHIP-0805(2012-METRIC)
C27	1u	C_CHIP-0805(2012-METRIC)
C28	100n	C_CHIP-0805(2012-METRIC)
C29	100n	C_CHIP-0805(2012-METRIC)
C30	22u	C_CHIP-0805(2012-METRIC)
C31	22u	C_CHIP-0805(2012-METRIC)
C32	100n	C_CHIP-0805(2012-METRIC)
C33	100n	C_CHIP-0805(2012-METRIC)
C34	100n	C_CHIP-0805(2012-METRIC)
C35	1u	C_CHIP-0805(2012-METRIC)
C36	100n	C_CHIP-0805(2012-METRIC)
C37	100n	C_CHIP-0805(2012-METRIC)
C38	2.2u	C_CHIP-0805(2012-METRIC)
C39	2.2u	C_CHIP-0805(2012-METRIC)
C40	4.7u	C_CHIP-0805(2012-METRIC)
C41	10n	C_CHIP-0805(2012-METRIC)
C42	1u	C_CHIP-0805(2012-METRIC)
C43	2.2u	C_CHIP-0805(2012-METRIC)
C44	1u	C_CHIP-0805(2012-METRIC)
D1	DIODE	DIODE_DO-219-AC(SOD323F)
D2	DIODE	DIODE_DO-219-AC(SOD323F)
D3	DIODE	DIODE_DO-219-AC(SOD323F)
D4	DIODE	DIODE_DO-219-AC(SOD323F)
D5	DIODE	DIODE_DO-219-AC(SOD323F)
D6	DIODE	DIODE_DO-219-AC(SOD323F)
IC1	OPA357AIDBVR	OPA357AIDBVR
IC2	LT1818CS5#TRMPBF	LT1818CS5#TRMPBF
IC3	OPA820IDBVR	OPA820IDBVR
IC4	DAC5571IDBVTG4	DAC5571IDBVTG4
IC5	SN74LVC1G66DBVRE4	SN74LVC1G66DBVRE4
IC6	LT1818CS5#TRMPBF	LT1818CS5#TRMPBF
IC7	OPA357AIDBVR	OPA357AIDBVR
IC8	OPA820IDBVR	OPA820IDBVR
IC9	DAC5571IDBVTG4	DAC5571IDBVTG4
IC10	SN74LVC1G66DBVRE4	SN74LVC1G66DBVRE4
IC11	TCA9554DBQR	TCA9554DBQR
IC12	DAC5571IDBVTG4	DAC5571IDBVTG4
IC13	LM27762DSST	LM27762DSST
IC14	LP2985-33DBVR	LP2985-33DBVR
IC15	TCA9554DBQR	TCA9554DBQR
IC16	SN74LVC2G53DCUR	SN74LVC2G53DCUR
IC17	LM4040DELT-2.0	LM4040DELT-2.0
J1	73137-5003	73137-5003
J2	73137-5003	73137-5003
J4	F305-1A7H1-11040-E100	F305-1A7H1-11040-E100
J5	PCB.SMAFRA.HT	PCB.SMAFRA.HT
R1	680k	R_CHIP-0805(2012-METRIC)
R2	100k	R_CHIP-0805(2012-METRIC)
R3	1k	R_CHIP-0805(2012-METRIC)
R4	100	R_CHIP-0805(2012-METRIC)
R5	680	R_CHIP-0805(2012-METRIC)
R6	100	R_CHIP-0805(2012-METRIC)
R7	47	R_CHIP-0805(2012-METRIC)
R8	1k	R_CHIP-0805(2012-METRIC)
R9	100	R_CHIP-0805(2012-METRIC)
R10	100	R_CHIP-0805(2012-METRIC)
R11	100	R_CHIP-0805(2012-METRIC)
R12	1k	R_CHIP-0805(2012-METRIC)
R13	680k	R_CHIP-0805(2012-METRIC)
R14	100k	R_CHIP-0805(2012-METRIC)
R15	1k	R_CHIP-0805(2012-METRIC)
R16	100	R_CHIP-0805(2012-METRIC)
R17	680	R_CHIP-0805(2012-METRIC)
R18	100	R_CHIP-0805(2012-METRIC)
R19	47	R_CHIP-0805(2012-METRIC)
R20	1k	R_CHIP-0805(2012-METRIC)
R21	100	R_CHIP-0805(2012-METRIC)
R22	100	R_CHIP-0805(2012-METRIC)
R23	100	R_CHIP-0805(2012-METRIC)
R24	1k	R_CHIP-0805(2012-METRIC)
R25	22	R_CHIP-0805(2012-METRIC)
R26	22	R_CHIP-0805(2012-METRIC)
R27	4.7k	R_CHIP-0805(2012-METRIC)
R28	4.7k	R_CHIP-0805(2012-METRIC)
R29	22	R_CHIP-0805(2012-METRIC)
R30	1k	R_CHIP-0805(2012-METRIC)
R31	22	R_CHIP-0805(2012-METRIC)
R32	1M	R_CHIP-0805(2012-METRIC)
R33	87k	R_CHIP-0805(2012-METRIC)
R34	87k	R_CHIP-0805(2012-METRIC)
R35	50k	R_CHIP-0805(2012-METRIC)
R36	50k	R_CHIP-0805(2012-METRIC)
S1	ADG621BRMZ	ADG621BRMZ
S2	ADG621BRMZ	ADG621BRMZ
U1	RASPBERRY_PICOSMD	RASPBERRY_PICOSMD
U3	ADC08100CIMTCMTC24	ADC08100CIMTCMTC24
U4	TSC2004IRTJRRTJ20_2P7X2P7	TSC2004IRTJRRTJ20_2P7X2P7
U5	TLV271CDBVRDBV5	TLV271CDBVRDBV5
U6	TLV3501AIDBVRDBV6	TLV3501AIDBVRDBV6
U7	AD9833FUNCGEN	AD9833FUNCGEN

3. Software

3.1.1 Programmierung des Pi Pico mit Micropython

Zuerst muss die Firmware von der Raspberry Pi Website heruntergeladen werden und dann auf dem 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. Als Entwicklungsumgebung kann entweder die Thonny DIE oder Visual Studio Code mit der Micropico Erweiterung dienen.

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.

3.1.2 Schwierigkeiten der Nutzung externer Bibliotheken mit Micropython

Für die Nutzeroberfläche des Oszilloskops wird die Grafikbibliothek lvgl[2] benötigt, welche offiziell über lv_micropython[3] auch die Verwendung unter Micropython unterstützt. In unserem Fall kann aber maximal einer 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  git clone https://github.com/lvgl/lv_micropython.git
2  cd lv_micropython
3  git submodule update --init --recursive lib/lv_bindings
4  make -C ports/rp2 BOARD=PICO submodules
5  make -j -C mpy-cross
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_micropyton/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

[2] https://github.com/lvgl/lvgl

[3] https://github.com/lvgl/lv_micropython

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. Bestehende Probleme

4.1.1 Problem Abweichende Software

Leider befindet sich die Software nicht auf dem richtigen Hardwarestand, was dazu führt dass die Platine derzeit nicht nutzbar ist. Einige Komponenten weichen dadurch ab oder können nicht korrekt angesteuert werden. 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.

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.

4.1.2 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.

SPI-Anbindung des Funktionsgenerators

Der Funktionsgenerator wird über den SPI-Bus des Pi Pico angebunden, jedoch wurde durch Versäumnis diese Verbindung auf den IO-Expander gelegt. Der IO-Expander ist nicht fähig mit seinen Ein-/Ausgängen einen weiteren SPI-Bus zu erzeugen. Die SPI-Pins des Funktionsgenerators müssten and eine SPI-Schnittstelle des Pi Picos angebunden werden.

Leider ist hier kein geeigneter Pin mehr verfügbar. Würde eine Umstellung des Displays auf SPI erfolgen, könnte der Funktionsgenerator wohlmöglich mit an denselben Bus angebunden werden. So muss dieser Punkt in Schaltplan sowie dem Platinenlayout überarbeitet werden.

4.2 Mögliche Problemlösungen

4.2.1 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:

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/Sekunde[4] nutzbar sein. Nach eigenem Empfinden würde die Pin-Ersparnis und die Verfügbarkeit fertiger Bibliotheken hier zum Vorteil von SPI überwiegen.

4.2.2 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. Das Trigger Signal selbst erfolgt dann über das TRIG-Signal, welches als Eingang im Pi Pico umgesetzt werden müsste und dort an die Nutzeroberfläche übermittelt werden muss.

Die genaue Adressierung kann auf Seite 15 des Datenblatts[5] nachgelesen werden.

4.2.3 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.

[4] https://www.reddit.com/r/esp32/comments/r5awqp/made_a_refresh_rate_test_comparison_between_the/

[5] https://www.ti.com/lit/ds/symlink/dac5571.pdf

5. Fazit

5.1 Stand zu Projektende

Zu Projektende ist das Gerät nicht funktionsfähig. Das rührt aus einer Kombination von Hardwareproblemen und dem, dass die Software in ihrer jetzigen Form nicht mit der Platine funktioniert. Bis zur Präsentation wird probiert zumindest das Display mit neuem C-Code etwas anzeigen zu lassen. Es existiert eine Platine auf der alles bis auf den zweiten Eingangskanal aufgebaut und funktionsfähig ist.

5.2 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

Leider endet das Projekt nicht mit einem vorzeigbaren funktionierenden Gerät.