Projekt: MEXLE 2020 Handoszilloskop

Student Group

First Name	Surname	Matrikel Nr.

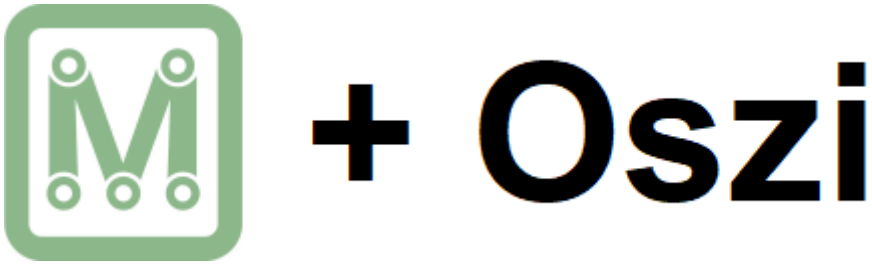
Table of Contents

Projekt: MEXLE 2020 Handoszilloskop 4
1. Einführung 4
Anforderungen 4
Vorhandene Arbeiten 4
Gründe für die Neuauslegung 4
2. Konzept 4
3. Funktionen des Handoszilloskop 5
4. Hardware 6
4.1 Auswahl der Hardwarekomponenten 6
4.1.1 Mikrocontroller 6
Vergleich potenzieller Mikrocontroller 6
Auswahl 6
Eigenschaften der PIC32MK-Familie 7
Weitere Hardware 7
4.1.2 Drahtlose Konnektivität 7
Standards 7
Auswahl 8
ESP32 8
Vergleich mit dem Vorgänger 8233 9
Weitere Alternative 9
4.1.3 Analoge Eingänge 10
Gabotronics 10
Beschreibung 10
Schaltung 11
Differentielle Messung und Single Ended Messung 11
Simulation 12
Verbesserungspotential 17
4.1.4 Display 17
Spezifikation 17

Anschlüsse ... 18
Spannungsversorgung ... 19
4.2 Integration in Vorarbeiten ... 19
4.3 Development Board ... 20
Anschlüsse und Komponenten ... 20
PIC32MK1024GPE100 ... 22
4.4 MikroBUS Prototypen ... 22
4.4.1 MikroBUS-Standard ... 22
Platinengrößen ... 23
Konfigurierbarkeit ... 23
4.4.2 Oszilloskop-Modul mit Display ... 23
Varianten ... 25
4.4.3 ESP32-Modul ... 25
Layoutvarianten ... 27
4.5 Akku-Schaltung und Spannungsversorgung ... 27
4.6 PIC32 Peripherieschaltung und Spannungsversorgung ... 28
4.7 Preisvergleich Neuauslegung und Vorarbeiten ... 28
4.8 Weitere Entwicklung der Hardware ... 29
5. Software ... 30
5.1 PIC32 ... 30
5.1.1 MPLAB X ... 30
5.1.1.1 Benutzeroberfläche ... 30
5.1.1.2 Projekt anlegen ... 32
5.1.1.3 PIC32 config settings ... 32
5.1.1.3 MPLAB Harmony ... 32
5.1.1.4 Harmony ADCHS ... 33
5.1.1.5 Harmony Pinzuweisung ... 34
5.1.1.6 Bare Metal Pinzuweisung ... 35
5.1.1.7 Interrupts ... 36
5.1.1.8 Unterschied Harmony und MPLAB Code Configurator ... 36
5.1.1.9 Data Visualizer ... 36
5.1.1.10 MPLAB und Harmony Eigenheiten ... 37
5.1.2 MPLAB Beispielcode ... 38
5.1.3 Handoszi Programmierung ... 38
OLED ... 38
Ausblick Displayprogrammierung ... 39
ADC ... 40
GPIO ... 40
5.1.4 Testen des Handoszi ... 41
Terminal Programme ... 41
Vergleich ADC-Performance ... 44
Auswertung ADC ... 45
5.2 Kommunikation zwischen ESP und PIC32 ... 46
Fazit: ... 48
5.3 ESP32 ... 48
5.3.1 Programmierumgebungen ... 48
Erforderliche Software ... 48
Entwicklungsumgebungen ... 48
Einrichten eines Projektes in Eclipse ... 48
5.3.2 Nutzung eines Betriebssystems ... 52
5.3.3 Testkit mit ESP32 ... 52

5.3.4 implementierte Funktionen 52
I²C-Testprogramm 53
UART-Testprogramm 53
Webserver mit Demo-Webseite 53
5.3.5 Geplante Funktionen 54
5.4 Anbindung an Oszilloskopsoftware für PCs 54
5.4.1 Sigrok PulseView 54
5.4.2 Saleae Logic 55
6 Ausblick 55
6.1 Hardware 55
6.2 PIC32 Programmierung 55
6.3 ESP32 Programmierung 56
Quellenverzeichnis 56

Projekt: MEXLE 2020 Handoszilloskop



1. Einführung

Als Teil des MEXLE Systems soll ein Oszilloskopstift entwickelt werden. Mit diesem soll eine kostengünstige Möglichkeit für die Studierenden geschaffen werden im Labor und bei Projekten zu Hause die Funktionen eines Oszilloskops zu nutzen.

Anforderungen

Die Anforderungen an die Funktionen des Oszilloskopstiftes ergeben sich wie folgt:

- Messung einer analogen Spannung im Bereich von -10V bis +10V
- Die Abtastung soll Signale (Rechtecksignale und sinusförmige Signale) mit mindestens 50 kHz, im Optimalfall bis 1 MHz erfassen können
- Ausgabe der Messdaten über ein Display
- Anbindungsmöglichkeiten an einen PC
    - über USB-Kabel
    - über WLAN
- Möglichkeit die gemessenen Spannungen mit einer Oszilloskopoberfläche darzustellen
- Der Zielpreis eines Exemplars sollte unter 60€ liegen

Vorhandene Arbeiten

Aus einer vorherigen Arbeit war bereits ein Entwurf eines Oszilloskopstiftes vorhanden. Dieser umfasste einen Mikrocontroller auf ARM Basis sowie eine Schaltung zur Analog-Digital-Wandlung von gemessenen Spannungen.

Gründe für die Neuauslegung

Die Analog-Digital-Wandlung des im Entwurf vorhandenen ARM-Mikrocontroller erfüllte nicht die Anforderungen an eine ausreichend schnelle Messwerterfassung und sollte daher durch einen Mikrocontroller mit besserer ADC-Performance ausgetauscht werden. Die Steuerung der Analog-Schaltung war durch die hohe Komplexität auf eine große Anzahl an Ein- und Ausgängen des Mikrocontroller angewiesen. Um die Kosten durch den Einsatz eines kleineren Mikrocontroller-Modells zu senken, war eine Vereinfachung des Analog-Teils erforderlich.

2. Konzept

Das Handoszilloskop an sich dient nur zur Aufnahme und Verarbeitung der Spannungssignale. Die Darstellung der Messung erfolgt nicht auf dem Handoszilloskop selbst. Stattdessen wird ein Endgerät, zum Beispiel ein PC oder ein Smartphone, mittels WLAN mit dem Accesspoint den der ESP32 erzeugt

verbunden. Mit einem Webbrowser kann die Webanwendung auf dem ESP32 genutzt werden, welche die Oszilloskopoberfläche darstellt. Die komplette Systemübersicht des Handoszilloskops ist in figure 1 zu sehen.

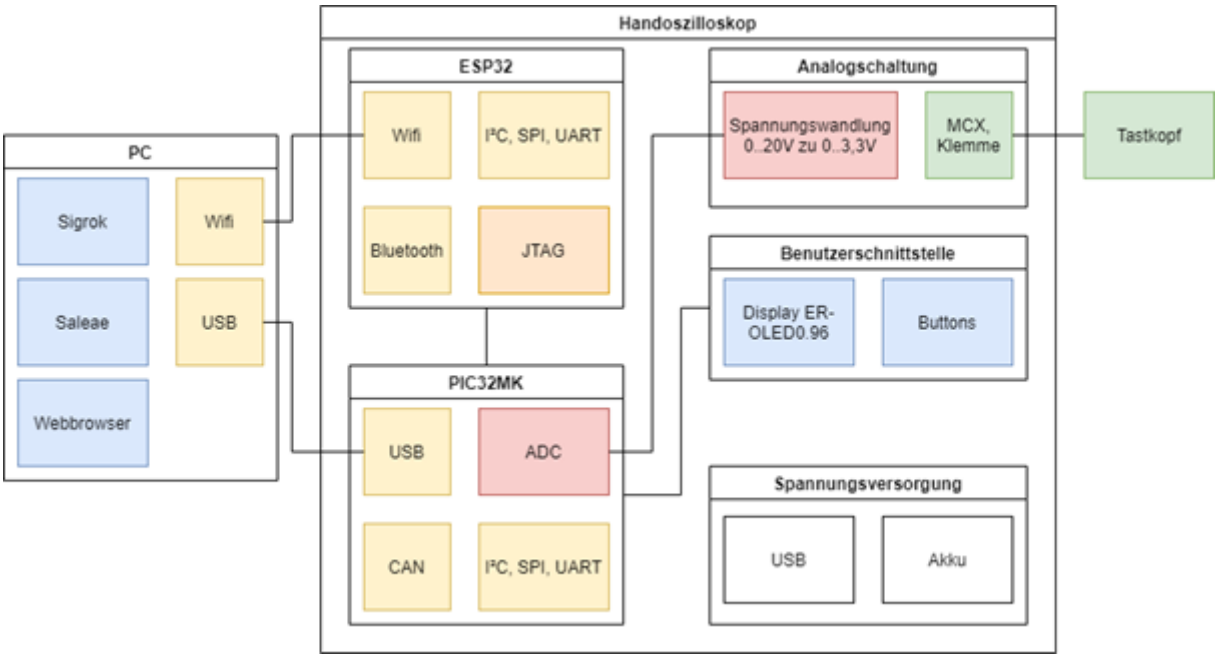


Fig. 1: Handoszilloskop Systemübersicht

Das Gerät soll möglichst kompakt und portabel sein, damit Studenten und Hochschulmitarbeiter flexibel in der Handhabung sein können und das Gerät nicht an die Räume der Hochschule gebunden ist.

3. Funktionen des Handoszilloskop

Die Erfassung der Spannungswerte geschieht über vier miteinander verschachtelte ADC-Wandler, womit eine Sampling-Rate von 12 MSps erreicht wird. Laut dem Nyquist-Abtasttheorem können somit Frequenzen bis 6 MHz gemessen werden. [1)]
Bedient wird das Handoszilloskop mithilfe von drei auf dem Gerät verbauten Tastern. Auf einem OLED-Display werden wichtige Parameter zur Messung angezeigt. Eine mögliche Form der Darstellung ist in figure 2 abgebildet.

Fig. 2: GUI OLED-Display

Die daten die auf dem Display angezeigt werden können, umfassen beispielsweise die Frequenz des gemessenen Signals, den Mittel-, bzw. Effektivwert der gemessenen Spannung oder die verwendete Sampling-Rate des Handoszilloskops. Die Datenübertragung an einen Computer ist über USB oder WLAN möglich. Eingestellt wird diese mithilfe der vorhandenen Taster. Über die USB-Schnittstelle erfolgt ebenfalls die Anbindung an Software wie sigrok PulseView oder Saleae Logic.

4. Hardware

Die überarbeitete Version der Hardware umfasst im Wesentlichen die 4 Funktionsbereiche: Mikrocontroller, Drahtlose Konnektivität, Display und analoger Schaltungsteil.

4.1 Auswahl der Hardwarekomponenten

Zunächst war es erforderlich die neuen Komponenten auszuwählen.

4.1.1 Mikrocontroller

Als Ersatz für den ARM-basierten Mikrocontroller, wurden Mikrocontroller der PIC32-Serie von Microchip betrachtet. Diese sind auf industrielle Anwendungen ausgelegt. Sie zeichnen sich besonders durch leistungsstarke Analog-Digital-Komponenten aus und sind daher sehr gut für die Analog-Digital-Wandlung und den Einsatz in einem Oszilloskop geeignet.

Vergleich potenzieller Mikrocontroller

Es erfolgte ein Vergleich mehrerer Modelle der PIC32-Serie. In die engere Auswahl wurden die in nachfolgender Tabelle aufgelisteten zwei Modelle aufgenommen.

Eigenschaften	**PIC32MK GP/MC**	**PIC32MZ DA**
ADC Speed (Msps)	25	18
DAC (channels/bits)	3/12	-
USB (FS/HS)	2F + P	1H + P
LCD Controller (Controllerless, Integrated)	E/L	I
Mouser Preis	ca. 4 -5 €	ca. 12 €
Mouser Link (alle Varianten der Serie)	Mouser Suche: "PIC32MK GP MC"	Mouser Suche: "PIC32MZ DA"
Link	PIC32MK Produkt Familie	PIC32MZ DA Familie
Development Board	PIC32MK GP Development Kit	PIC32MZ Embedded Graphics with external DRAM (DA) Starter Kit

Auswahl

Da der Schwerpunkt des Oszilloskops auf der Messung von Spannungen liegt ist hier besonders der PIC32MK geeignet. Dieser ermöglicht höhere ADC Geschwindigkeiten und verfügt zudem über Digital-Analog-Wandler, welche die Möglichkeit bieten, das Oszilloskop um einen Funktionsgenerator zu erweitern. Da auch der Preis einen entscheidenden Anteil an der Wahl des Mikrocontrollers hatte war hier ebenfalls der PIC32MK zu bevorzugen.

Eigenschaften der PIC32MK-Familie

Je nach Ausstattung und Leistungsfähigkeit bietet ein Mikrocontroller der PIC32MK Familie folgende Merkmale [2)]:

- Architektur: MIPS32 microAptiv™
- CPU mit 120MHz Takt
- Flash Speicher bis zu 1 MB
- SRAM mit bis zu 256 KB
- Anzahl an Pins 64 oder 100
- Package als QFP oder QFN
- integrierte USB Schnittstelle
- Kommunikationsschnittstellen: CAN FD, I²C, UART und SPI
- 12 Bit Analog-Digital-Wandler mit bis zu 25,45 Msps Leistung kombiniert
- 12 Bit Digital-Analog-Wandlung

Weitere Hardware

Neben dem Mikrocontroller als Hauptkomponenten des Oszilloskops wurden weitere Hardwarekomponenten für die Nutzerinteraktion, Messung von Spannungen und Ausgabe von Daten benötigt, diese sind im Folgenden näher beschrieben.

4.1.2 Drahtlose Konnektivität

Zusätzlich zu einer kabelgebundenen Spannungsversorgung und Datenkommunikation soll das Oszilloskop die Möglichkeit bieten, Daten, welche bei der Messung erfasst werden, aber auch Einstellungen drahtlos an einen PC übertragen und von diesem empfangen zu können. Hierfür bieten sich die Kommunikationsstandards Bluetooth und WLAN an, da die erforderlichen Schnittstellen in der Regel in allen modernen PCs vorhanden sind und als einfache Controllerbausteine in Mikrocontrollerschaltungen integriert werden können.

Standards

Bluetooth:

- Bluetooth wurde als Industriestandard von der Bluetooth Special Interest Group entwickelt und dient der Datenübertragung per Funk zwischen zwei Geräten. Der Fokus liegt dabei auf einer vergleichsweise kurzen Distanz zwischen den Geräten. Die Frequenz der Funkübertragung liegt bei 2,4 GHz, daher sind Störungen durch andere Geräte wie Telefone, Mikrowellen oder WLAN Netze möglich. [3)]
- Aktuelle Geräte verwenden den Standard in der Version V4.x oder V5.x. Zudem beherrschen die meisten Geräte den Standard Bluetooth Low Energy , dieser ist besonders auf geringen Stromverbrauch ausgelegt. Auch die Datenrate hat sich mit den Versionen gesteigert und kann mit Version 5 bis zu 50 MBit/s betragen. [4)], [5)]

WLAN:

- Englisch für Wireless Local Area Network, wird umgangssprachlich für Funknetze nach dem Standard IEEE 802.11 verwendet. Es wird hauptsächlich der 2,4 GHz und 5 GHz Bereich des Frequenzspektrums, je nach Version des Standards verwendet. Aktuelle Geräte beherrschen in der Regel mindestens den Standard 802.11n im 2,4 GHz Funkbereich. [6)]
- Wi-Fi ist eine Zertifizierung nach den IEEE 802.11 Standards und wird von der Wi-Fi Alliance durchgeführt. [7)]

Auswahl

WLAN- und Bluetooth-Controller gibt es in der einfachsten Form als reine ICs die es erfordern die umgebende Peripherie, Spannungsversorgung und Antenne mit einem eigenen Design zu entwickeln. Zusätzlich gibt es die meisten Controller auch als Komplettmodule, diese integrieren die zuvor genannten peripheren Komponenten in einem bereits fertigen Layout mit Platine. Die Antenne kann dabei auf der Platine vorhanden sein oder extern durch einen entsprechenden Antennenstecker angeschlossen werden. Diese Komplettmodule sind bereits zum kleinen Preis zu haben und erfordern einen geringeren Aufwand für die Integration in ein eigenes Layout. Daher wurden für das Oszilloskop solche Module betrachtet. Der Funktionsumfang der Module ist von Hersteller zu Hersteller unterschiedlich. Einige Module integrieren Bluetooth und Wifi wie die Modelle des ESP32, andere bieten nur Wifi- oder Bluetooth-Funktionalität an. Nachfolgende Tabelle listet die Module auf, die für das Handoszilloskop zur Auswahl standen.

Eigenschaften	**ESP32-SOLO Serie**	**ESP32-WROOM Serie**	**ESP32-WROVER Serie**	**PIC32MZ-W1 Wi-Fi®**	**ATWINC1500**
Beschreibung	Single Core MCU mit WLAN und Bluetooth	Dual Core mit WLAN und Bluetooth	Dual Core mit WLAN und Bluetooth, integriert neben Flash auch Pseudostatisches RAM	Mikrocontrollerkern mit integriertem WLAN	SPI to Wi-Fi module
Interfaces	GPIOs, SPI, I2S, I2C, PWM, RMT, ADC, DAC and UART			ADC, CAN, Ethernet, USB und weitere	SPI zur Verbindung mit einem Host Mikrocontroller wie der PIC32
Datenblatt	ESP32SOLO1 Datenblatt	ESP32WROOM32E & ESP32WROOM-32UE Datenblatt	ESP32WROVERE & ESP32WROVER-IE	PIC32MZ-W1 Wi-Fi® SoC and Module Family	ATWINC1500
Preis	2,88 €	ca. 2-3 €	ca. 2-3 €	ca. 12 €	ca. 6 €
Mouser Link	Mouser: ESP32-SOLO-1(M113SH3200PH3Q0)	Mouser Suche: "wroom32"	Mouser Suche: "wrover"	nicht bei Mouser erhältlich	Mouser Suche: "ATWINC1500"

Stand: 09.11.2020

Weitere Übersichtsseiten für WLAN fähige Mikrocontroller oder WLAN Module.

- ESP32 Wi-Fi & Bluetooth Module
- Microchip Embedded Wi-Fi®

Das ATWINC1500 bietet für den doppelten Preis wie ein ESP32 Modul lediglich eine WLAN Funktionalität und wurde daher nicht weiter betrachtet. Interessante Alternative zu einem zusätzlichen WLAN Modul war der PIC32MZ-W1 Wi-Fi®. Dieser besteht aus einem PIC Mikrocontrollerkern und erweitert diesem mit WLAN. Ein Preis von ca. 12 € ist im Vergleich teurer als ein eigenständiger PIC32MK und ein ESP32 zusammen und damit ein entscheidender Nachteil für ein kostengünstiges Oszilloskop. Da der PIC32MZ-W1 nicht lieferbar war, fiel dieser ebenfalls aus der Auswahl für die drahtlose Erweiterung des Oszilloskops.

Nach Betrachtung der einzelnen Module, fiel die Wahl auf ein ESP32-WROOM Modul.

ESP32

Hauptkriterium für die Auswahl eines Funkmoduls war der Preis. Da die ESP32 einen deutlich größeren Funktionsumfang bei gleichzeitig kleinerem Preis liefern als die Konkurrenzprodukte von Microchip

wurde der ESP32 zur Erweiterung des Oszilloskops mit Drahtloskonnektivität ausgewählt. Da im Labor Entwicklungsplatinen mit ESP32-WROOM-Modulen zu Verfügung standen, wurde dieses Version gewählt. Einen großen Unterschied im Preis oder Funktionsumfang gab es zwischen den zu Auswahl stehenden ESP32-Modulen nicht.

Das ausgewählte ESP32 Modell beherrscht WLAN und Bluetooth. WLAN ist im Standard 802.11 b/g/n mit 2.4 GHz. Bluetooth in Version v4.2 BR/EDR und mit BLE (Bluetooth Low Energy) vorhanden. Die erforderliche Antenne war bereits auf der Platine des Moduls vorhanden und erforderte daher keine gesonderte Auslegung.[8)]

Daten des ESP32-WROOM-32E [9)]

- Xtensa® Dual-core 32-Bit LX6 Mikroprozessor mit einer Taktrate bis zu 240MHz
- 520 KB SRAM
- Wi-Fi im Standard 802.11b/g/n, mit Bitraten bis zu 150 Mbps bei 802.11n
- Bluetooth V4.2 BR/EDR und Bluetooth LE
- Version 32E mit PCB Antenne
- Schnittstellen: UART, SPI, I2C, PWM, I2S, GPIO, ADC, DAC und weitere
- 40 MHz Oszillator
- 4 MB Flash Speicher

Das ESP32-Modul soll auch als Benutzerschnittstelle eingesetzt werden können. Der Vorteil der ESP32 gegenüber einfacheren Wifi- und Bluetooth Controllern ist der integrierte 2-Kern Prozessor [10)]. Durch die ausreichend vorhandene Leistung ist es möglich diesen als Server zu verwenden und so ein Webinterface für ein Oszilloskop zu bieten. Mit diesem sollen sich die Funktionen des Oszilloskops steuern lassen und die Messdaten angezeigt werden können.

Vergleich mit dem Vorgänger 8233

Am Markt sind neben dem ESP32 auch die Vorgängermodule ESP8266 erhältlich. Diese bieten einen ähnlichen Funktionsumfang wie ein ESP32 Modul.

Um das Oszilloskop möglichst lange mit Komponenten versorgen zu können, ist eine ausreichend lange Lieferbarkeit der Bauteile erforderlich. Da der ESP32 die Modelle des ESP8266 am Markt ersetzt und damit noch länger unterstützt wird, ist dies ein entscheidender Vorteil der neuen ESP32-Module. Einen preislichen Vorteil durch die Wahl des ESP8266 gibt es nicht, da auch die neuen Module des ESP32 in derselben Preisspanne erhältlich sind.

Weitere Alternative

Bei anderen Projekten zum MEXLE System wird der TTGO Micro32 von Lilygo [11)] eingesetzt. Daher ist auch ein Vergleich zwischen dem verwendeten ESP32 und dem TTGO Micro sinnvoll.

Preislich ist der TTGO Micro32 mit 5,26 € [12)] etwa doppelt so teuer wie der ESP32 WROOM mit 2,31 € [13)]. Für das Oszilloskop kommt dieser daher nur in Frage, wenn sich ein wesentlicher Vorteil durch die Funktionen des TTGO Micro32 bietet.

Der Größenvergleich, zu sehen in nachfolgender Abbildung, zeigt den Vorteil des TTGO Micro32 gegenüber dem ESP32 WROOM Modell.

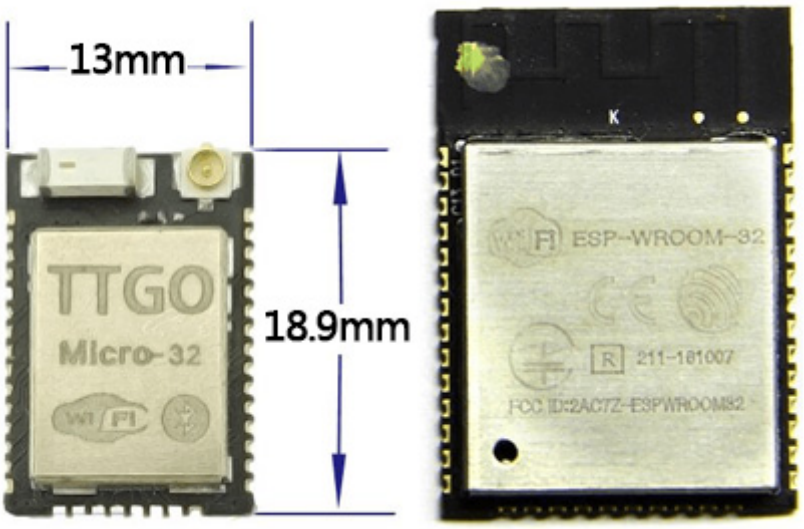


Fig. 4: Vergleich TTGO Micro32 und ESP32 WROOM Quelle figure 4: [14)]

Ziel des TTGO ist laut Hersteller den ESP32 in kleinerem Formfaktor anzubieten. Dies stellt den größten Unterschied zwischen den Modulen dar.

Der Kern der beiden Module ist jeweils ein ESP32, was in den sehr ähnlichen Eigenschaften resultiert. Beide Module besitzen 38 Pins, verfügen über die üblichen Schnittstellen wie I²C, UART und SPI und funken mit WLAN im Standard 802.11 b/g/n sowie Bluetooth 4.2 und BLE. Einziger Unterschied ist bei der verwendeten Antennentechnik. Der TTGO Micro32 verfügt sowohl über eine Keramik Antenne als auch über einen IPEX Stecker zum Anschluss einer externen Antenne. Das ESP32 WROOM Modul wie in der Abbildung dargestellt verfügt lediglich über eine PCB Antenne. Das WROOM Modul ist einer weiteren Konfiguration erhältlich, diese besitzt ebenfalls einen IPEX Stecker, verzichtet dabei aber auf die PCB Antenne. [15)], [16)]

Da es neben dem Größenunterschied keine wesentlichen Vorteile des TTGO gegenüber dem ESP32 WROOM gibt, der TTGO jedoch doppelt so teuer ist, ist der ESP32 WROOM besser für ein kostengünstiges Oszilloskop geeignet.

4.1.3 Analoge Eingänge

Für die Neuauslegung des Analog-Eingang-Teils wurde auf die Schaltpläne des Gabotronics Xscope zurückgegriffen.

Gabotronics

Die Firma Gabotronics stellt kleine, Mikrocontroller-basierte Oszilloksope her. Dieses basieren wie z.B. das XMEGA Xprotolab auf einem XMEGA Prozessor mit daran angeschlossener Analogschaltung zur Wandlung der Messsignale auf die Logikspannung des Mikrocontroller.[17)]

Beschreibung

Für das Oszilloskop wurde die Schaltung des Gabotronics Xprotolab an den PIC32 angepasst und auf einen Kanal verkleinert. Die Anpassung der Bauteilwerte an die gewünschte Messspannung erfolgte durch die Simulation der Schaltung mit TINA.

Gabotronics verwendet für das Xprotolab eine Spannung von 2 V bei der Analog-Digital-Wandlung. Für das Handoszilloskop war diese auf den vollen Bereich der möglichen Spannungen von 0 V bis 3,3 V für

die Analog-Digital-Wandlung zu erweitern. Die restliche Schaltung konnte größtenteils übernommen werden. [18)]

Da das Gabotronics Oszilloskop mehrere Kanäle verwendet, waren auch die Hardwarekomponenten dementsprechend groß dimensioniert [19)]. Bei der Begrenzung der Schaltung auf einen Kanal war der verwendete Operationsverstärker überdimensioniert und wurde ersetzt.

Anstelle des TL064 Operationsverstärker des Gabotronics, kommt bei der Schaltung des Handoszilloskops ein TLV9101 [20)] zu Einsatz. Dieser kann ebenfalls mit +5 V und -5 V Versorgt werden, ist aber auf einen anstelle von 4 Kanälen begrenzt. Ein kleineres Gehäuse sowie eine höhere Slew - Rate bieten einen weiteren Vorteil für das Handoszilloskop.[21)]

Schaltung

Als Anschlussmöglichkeit zum Messen von Spannungen sind ein MCX Anschluss sowie eine Klemme vorhanden. Über den MCX Anschluss können Oszilloskoptastköpfe verwendet werden. Steht dieser nicht zur Verfügung ist auch der Anschluss mittels Kabel über die Klemme möglich.

Die Schaltung wurde auf eine Messspannung im Bereich von -10 V bis +10 V ausgelegt. Über einen Spannungsteiler wird die gemessene Spannung zunächst auf eine Amplitude im Bereich von +3,3 V bis -3,3 V begrenzt. Durch eine angeschlossene Operationsverstärkerschaltung erfolgt eine Wandlung auf den gewünschten Spannungsbereich von 0 V bis 3,3 V für die Analog-Digital-Wandlung.

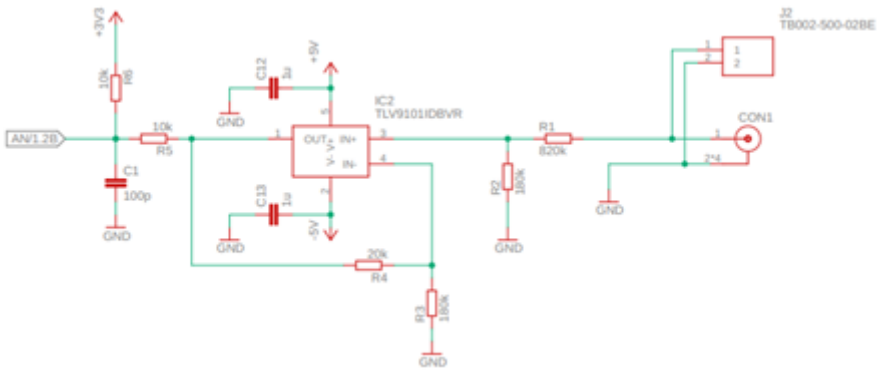


Fig. 5: Schaltplan der Messschaltung

Differentielle Messung und Single Ended Messung

Die zuvor dargestellte Schaltung ist auf eine Single Ended Messung ausgelegt. Hierdurch ist es erforderlich, dass die zu messende Spannung immer in Bezug auf die Masse der Oszilloskopschaltung gemessen wird.[22)]

Erfolgt eine Single Ended Messung bei einem Potentialunterschied der Massen die größer ist als die maximale Messamplitude, kann es zu Schäden am Messgerät führen. Dies tritt meist auf, wenn das Messgerät und das Messobjekt über unterschiedliche Netzteile mit Spannung versorgt werden und keine gemeinsame Erdung nutzen. Über eine differentielle Messschaltung kann eine Messung unabhängig vom Bezugspotential des Oszilloskops erfolgen. Dies ist der große Vorteil der differentiellen Messung. Potentialunterschiede der Massen können so ausgeglichen und Schäden am Messgerät vermieden werden.[23)]

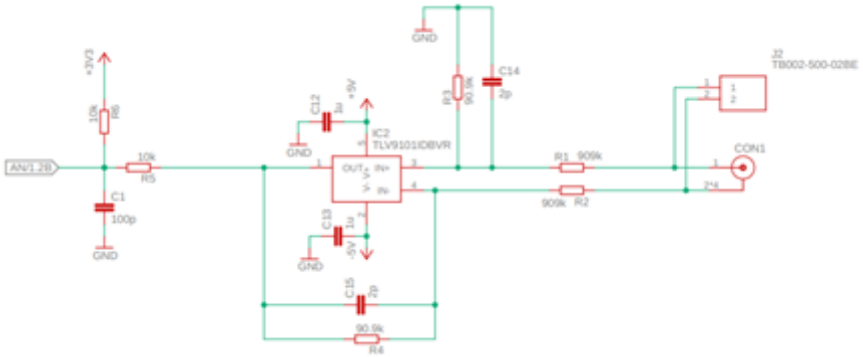


Fig. 6: Schaltplan der differentiellen Messschaltung

Um die Vorteile der differentiellen Messung auch mit dem Handoszilloskop zu nutzen, wurde eine zweite Schaltung entwickelt. Diese basiert ebenfalls auf der Schaltung des Gabotronics Xprotolab, führt jedoch eine differentielle Messanordnung ein.

Simulation

Um die erforderlichen Bauteilwerte der Analogschaltung nach der Einführung des TLV9101 zu ermitteln, wurde eine Simulation der Standardschaltung sowie der Schaltung mit differentieller Messung durchgeführt. Die Simulation erfolgte mit dem Modell des TLV9102 [24]. Dieses unterscheidet sich vom TLV9101 nur durch einen zusätzlichen Kanal und ein anderes Gehäuse. Als Simulationssoftware wurde TINA von Texas Instruments genutzt. Die Ergebnisse der Simulation sind im folgenden Abschnitt dargestellt.

Standardschaltung:

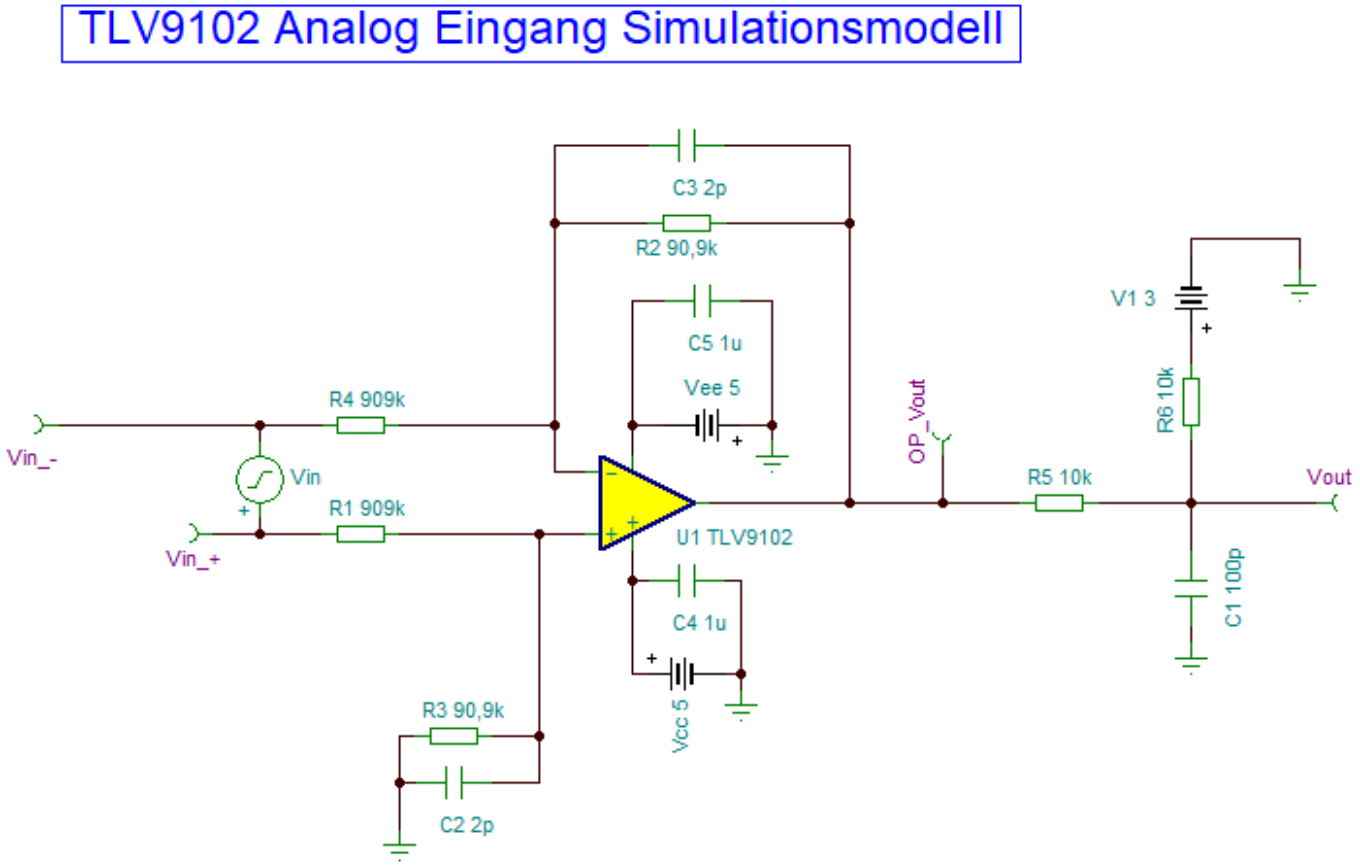


Fig. 7: Simulation der Messschaltung

Aus den Anforderungen an das Oszilloskop geht hervor das eine Spannung mit einer Amplitude von ±10 V gemessen werden können soll. Für die Simulation wurde daher zunächst ein Testsignal mit einer Amplitude von 10 V und einer Frequenz von 50 kHz verwendet, um die Umwandlung der Spannung zu analysieren. Das Ergebnis ist in figure 8 dargestellt. Bei einer Frequenz von 50 kHz erfolgt die Wandlung des Signals wie gewünscht unter Einhaltung des Spannungsbereichs von 0 V bis 3,3 V am Ausgang und kann somit in Verbindung mit dem Spannungslevel von 3,3 V der Logik des PIC32 verwendet werden.

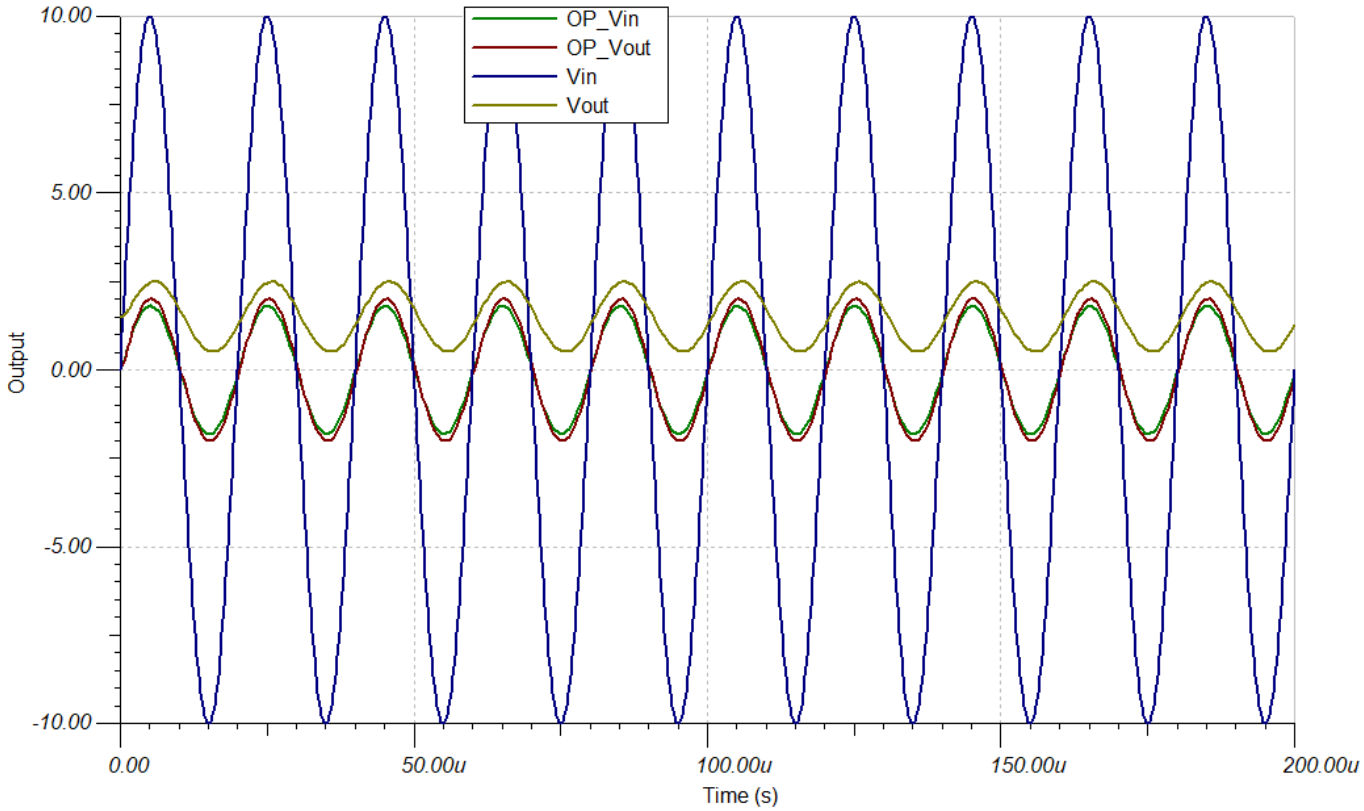


Fig. 8: Simulationsergebnis Messschaltung

Im zweiten Schritt war es erforderlich herauszufinden, welchen Frequenzbereich die Schaltung abdecken kann. Hierzu wurden der Frequenzgang und Phasengang der Schaltung simuliert. Dargestellt sind diese in nachfolgender Abbildung

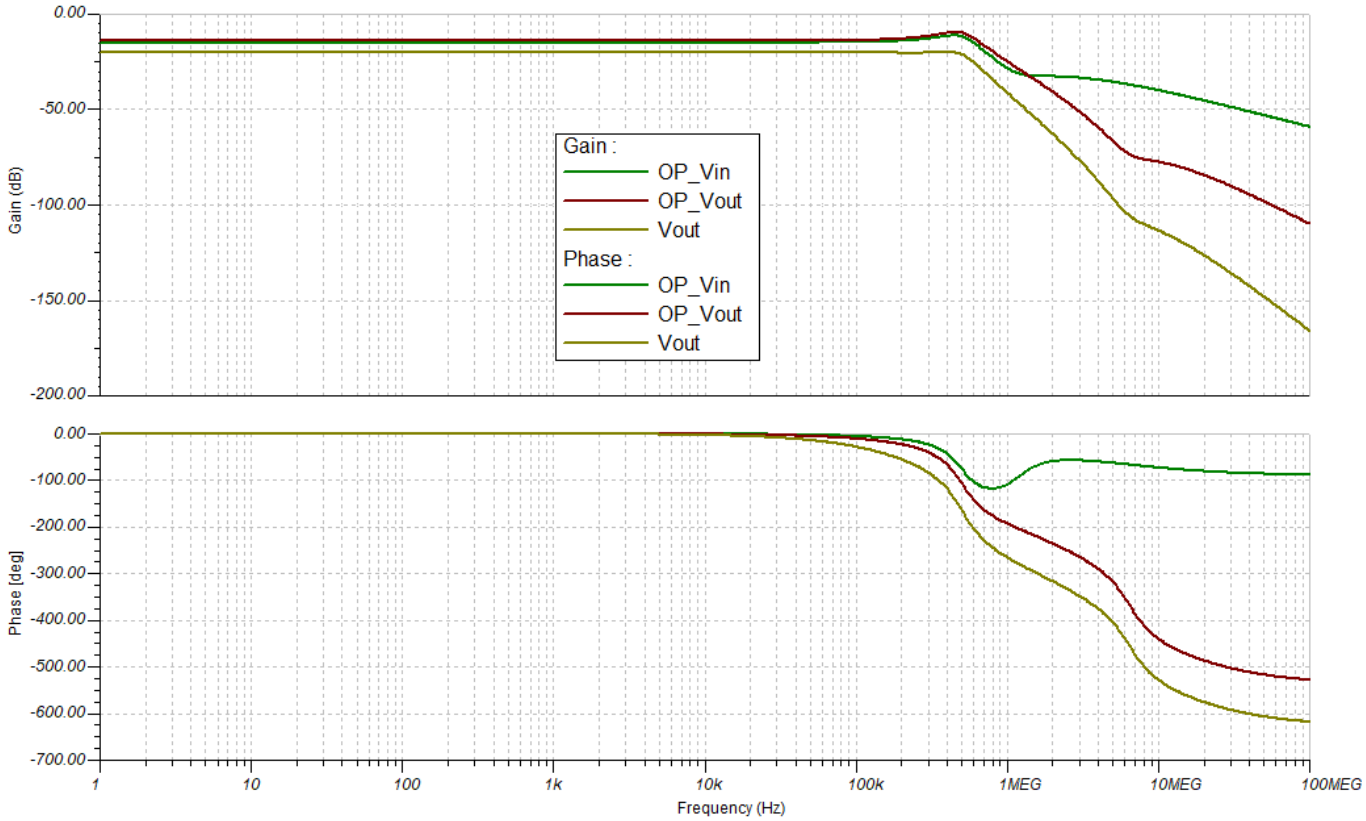


Fig. 9: Frequenz- und Phasengang Messschaltung

Ab einer Frequenz von 500 kHz fällt die Verstärkung der Ausgangsspannung ab. Der Frequenzgang zeigt damit, dass die Schaltung bis zu einer Frequenz von ca. 500 kHz sinnvoll nutzbar ist. Die Verstärkung der Schaltung liegt immer unter 0 dB, dies ist dadurch zu erklären, dass neben der Operationsverstärkerschaltung, ein vorgelagerter Spannungsteiler in der Schaltung vorhanden ist. Zudem wird der Ausgang des Operationsverstärker nicht direkt an den PIC32 angeschlossen. Als Zwischenglied ist ein Tiefpass integriert, welcher ebenfalls zu einer Dämpfung beiträgt.

Die Simulationsdateien sind zu finden unter:
Redmine->Handoszilloskop->Schaltplan->mikroBUS_Oscilloscope->Simulation

Differentielle Messschaltung:

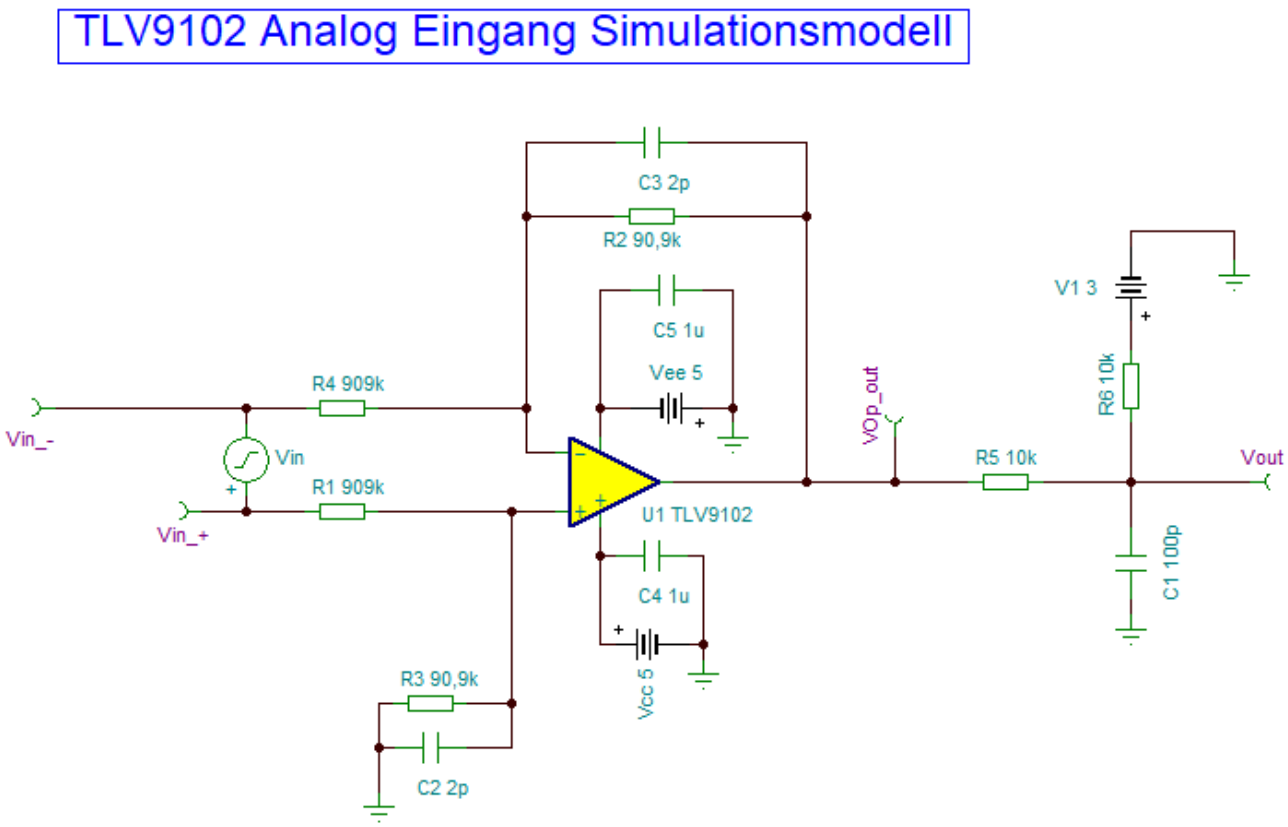


Fig. 10: Simulation differentielle Messschaltung

Wie auch bei der Messung mit der Standardschaltung wurde eine Simulation der Spannungswandlung mit einem Messsignal mit einer Amplitude von ±10 V und einer Frequenz von 50 kHz durchgeführt. Das Ergebnis der Simulation ist in figure 11 dargestellt. Auch hier wird der Bereich der Ausgangsspannung von 0 V bis 3,3 V eingehalten und kann somit an den PIC32 angeschlossen.

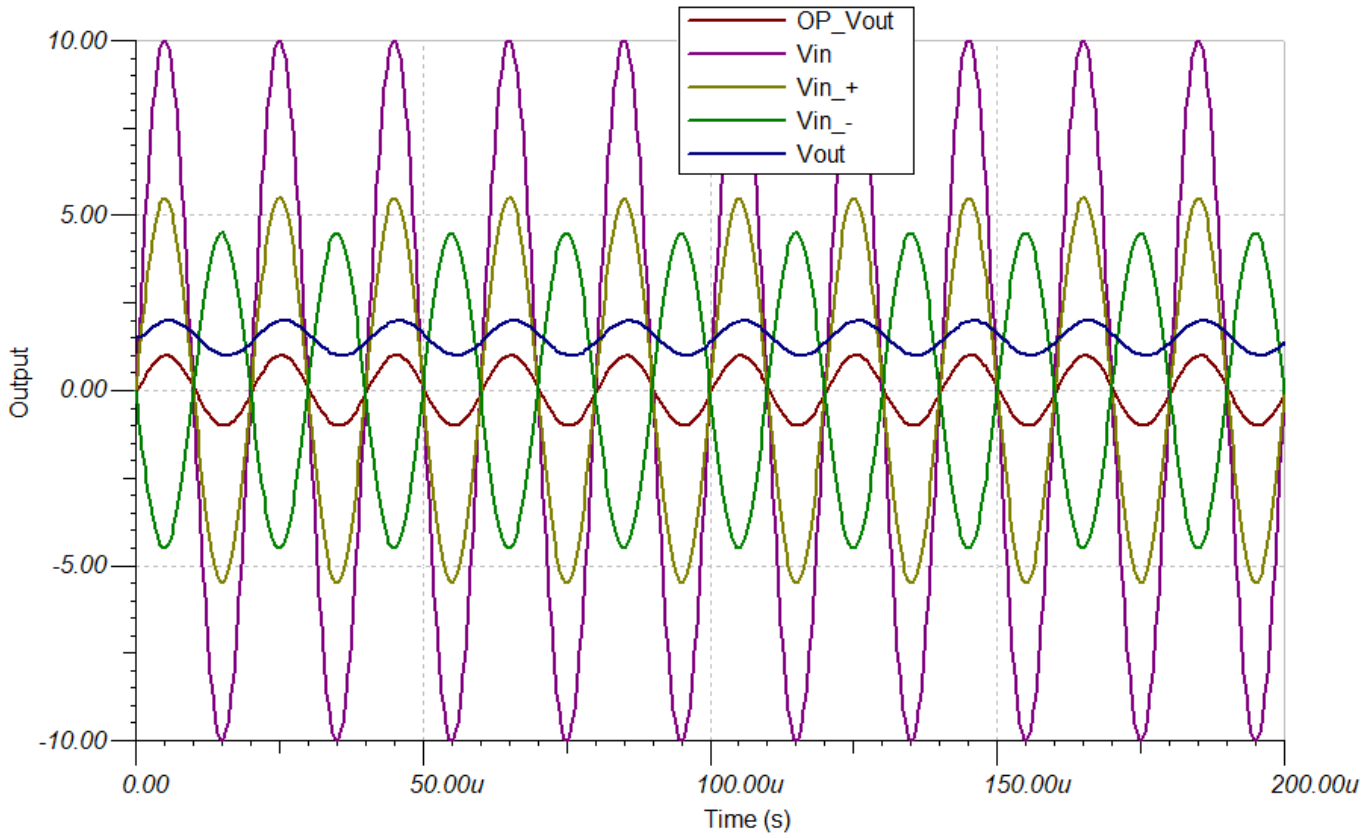


Fig. 11: Simulationsergebnis differentielle Messschaltung

Auch für die differentielle Messung war es erforderlich den Frequenz- und Phasengang zu betrachten. Das Ergebnis der Simulation ist in nachfolgender Abbildung zu sehen.

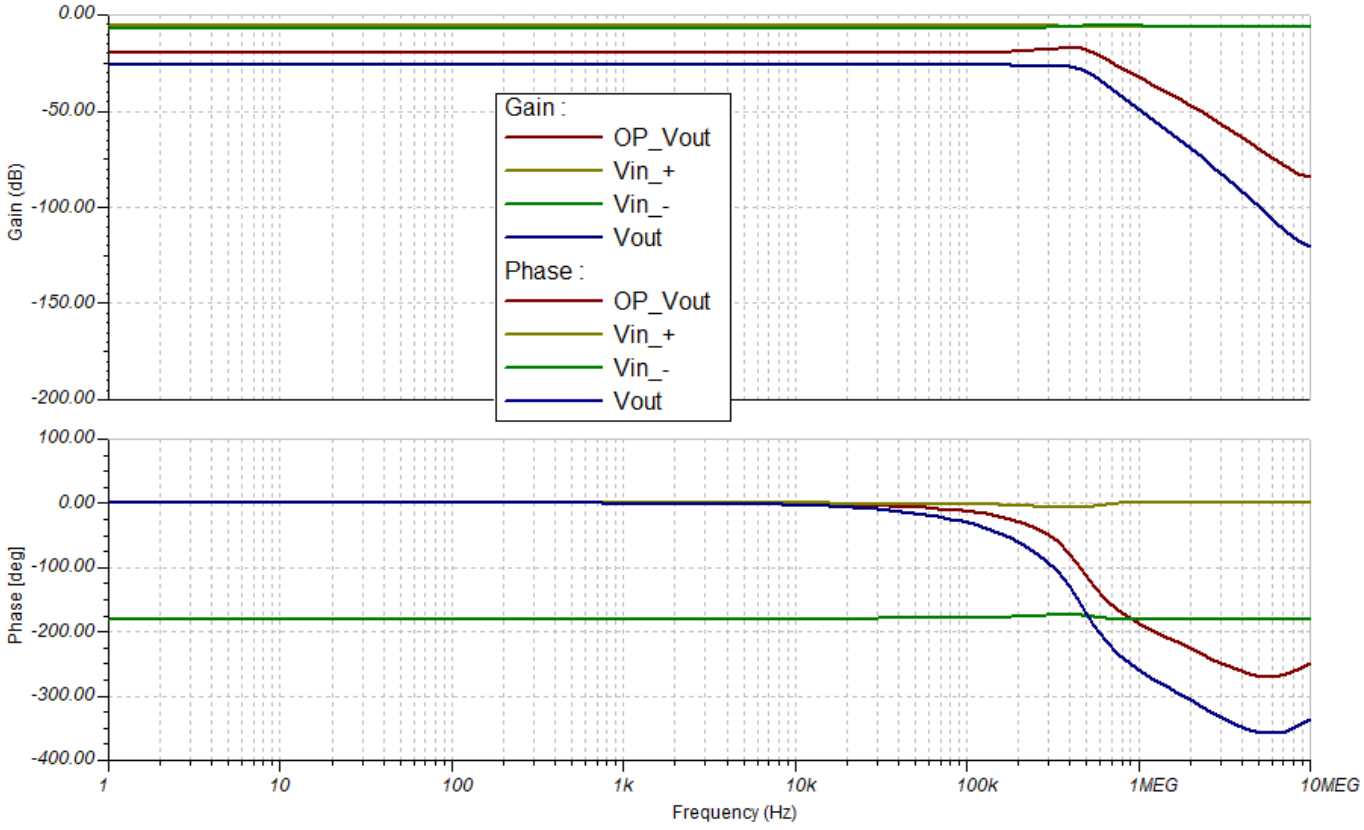


Fig. 12: Frequenz- und Phasengang differentielle Messschaltung

Ab einer Frequenz von 400 kHz fällt die Verstärkung der Ausgangsspannung ab. Der Frequenzgang zeigt damit, dass die Schaltung bis zu einer Frequenz von ca. 400 kHz sinnvoll nutzbar ist. Im Vergleich zu obiger Standardschaltung ist die Bandbreite der differentiellen Messschaltung um 100 kHz verringert.

Die Simulationsdateien sind zu finden unter:
Redmine->Handoszilloskop->Schaltplan->mikroBUS_Oscilloscope_differential->Simulation

Verbesserungspotential

Die Bandbreite des verwendeten Operationsverstärker beträgt 1,1 MHz. Hierdurch ist auch die Bandbreite, die für das Oszilloskop genutzt werden kann begrenzt. Da die Analog-Digital-Wandler des PIC32 eine Samplerate von bis zu 25,45 Msps bei einer Auflösung von 12 Bit ermöglichen, kann das Messsignal ausreichend gut erfasst werden. [25)], [26)]

Die Verstärkung der Schaltung zu den analogen Eingängen fällt bereits bei 400-500 kHz ab. Daher sollte durch eine Anpassung der Bauteilwerte eine Verbesserung und Annäherung der Grenzfrequenz an ca. 1 MHz erreicht werden können, um die Bandbreite der verwendeten Komponenten optimal nutzen zu können.

4.1.4 Display

Als Display wird ein ER-OLED0.96-1.3W OLED-Display verwendet, dieses kann über einen MOLEX 30 Pin FPC-Connector einfach auf das Board aufgesteckt werden. Ein Auflöten des Displays ist nicht erforderlich. Das Display kann hierdurch ohne größeren Aufwand durch ein Display mit anderer Größe oder Seitenverhältnis getauscht werden. Dieselbe Pinbelegung des neuen und alten Displays vorausgesetzt.[27)]

Spezifikation

Laut Datenblatt besitzt das ausgewählte Display eine Auflösung von 128×64 Pixeln bei einer Bildschirmdiagonale von 0.96 Zoll.[28)]

Das Display ist in verschiedenen Anzeigeversionen mit Ein- oder Zweifarbdarstellung erhältlich. Die einfachen Versionen des Displays können Bilder in Schwarz und Weiß (ER-OLED0.96-1W) oder Schwarz und Blau (ER-OLED0.96-1B) darstellen. Daneben gibt es eine Multicolor Version, die farbige Pixel in Blau und Gelb sowie einen schwarzen Hintergrund darstellen kann (ER-OLED0.96-1YB). Da es sich bei dem Display um ein OLED handelt ist keine zusätzliche Hintergrundbeleuchtung erforderlich. Das ursprüngliche Design des Oszilloskops hatte zur Anzeige LC-Displays geplant. Der Wegfall der Hintergrundbeleuchtung spart somit Platz auf der Platine ein.[29)]

Das Display hat einen integrierten Controller SSD1306 mit SRAM Buffer zur Ansteuerung des OLED Panel und erleichtert dadurch die Integration in die Schaltung.[30)]

Zur Anbindung des Displays an den PIC32 stehen mehrere Schnittstellen zur Verfügung. Die gewünschte Schnittstelle muss auf Hardwareebene durch die Konfiguration von 3 Pins (auf Masse legen oder mit Vcc verbinden) festgelegt werden. Die verfügbaren Schnittstellen sowie die jeweilige Pinkonfiguration sind in nachfolgender Tabelle aufgelistet.

	BS0	**BS1**	**BS2**
I2C	0	1	0
3-wire SPI	1	0	0

	BS0	BS1	BS2
4-wire SPI	0	0	0
8-bit 68XX Parallel	0	0	0
8-bit 80XX Parallel	0	1	1

Quelle Tabelle: [31)]

Die Auswahl der geeignetsten Schnittstelle folgte einem Vergleich der vorhanden Varianten.

Die Anbindung des Displays an den PIC32 sollte mit möglichst geringem Hardwareaufwand erfolgen, dementsprechend war eine Ansteuerung mit einer so geringen Anzahl an Leitungen wie möglich zu finden. Die Anbindung über die beiden 8 Bit Parallelbusse entfiel, da diese neben den 8 Datenleitungen weitere zusätzliche Steuerleitungen erfordern.

Alternativ kann das Display über I²C oder SPI angesteuert werden. Da das Display auch die Möglichkeit bieten soll, die Messdaten anzuzeigen, ist auch die Geschwindigkeit der Ansteuerung ein wichtiger Faktor der Betrachtung. Im Vergleich mit I²C ermöglicht SPI höhere Übertragungsgeschwindigkeiten. Daher war SPI gegenüber I²C zu bevorzugen.

Das Display kann mit SPI in zwei unterschiedlichen Anbindungskonfigurationen verwendet werden. Die Anbindung kann über 3 oder 4 Leitungen erfolgen. Da das Display nur Daten über die Schnittstelle empfangen kann, jedoch keine Daten zurücksendet, kann auf die vierte Leitung verzichtet werden. Das Display wurde daher über 3 wire SPI angebunden.

Anschlüsse

Neben den unterschiedlichen Farbdarstellungen gibt es das Display auch mit unterschiedlichen Anschlüssen. Zum einen kann das Display direkt mit dem angefügten Flachkabel auf die Platine aufgelötet werden. Weitere Varianten des Displays ermöglichen auch die Nutzung eines Klemmverbinders zur Montage auf der Platine. Dies hat den Vorteil, dass das Display ausgetauscht werden kann. Der Verbinder für die Verwendung mit einer Klemmbuchse besitzt schmalere Kontakte als die Lötversion. Dadurch sind diese nicht miteinander kompatibel. Zudem gibt es die Klemmversion in zwei Ausführungen. Die Kontakte sind entweder auf der Unterseite oder Oberseite des Kabels aufgebracht. Dies ist bei der Wahl und Lage der entsprechenden Buchse auf der Platine zu berücksichtigen.[32)]

Das Display des Handoszilloskops besitzt den Anschluss zum Klemmen mit den Kontakten auf der Oberseite. Beim Einbau des Display wird das Kabel unter das Display gebogen. Dadurch ist der Platzbedarf für das Display auf der Platine geringer.

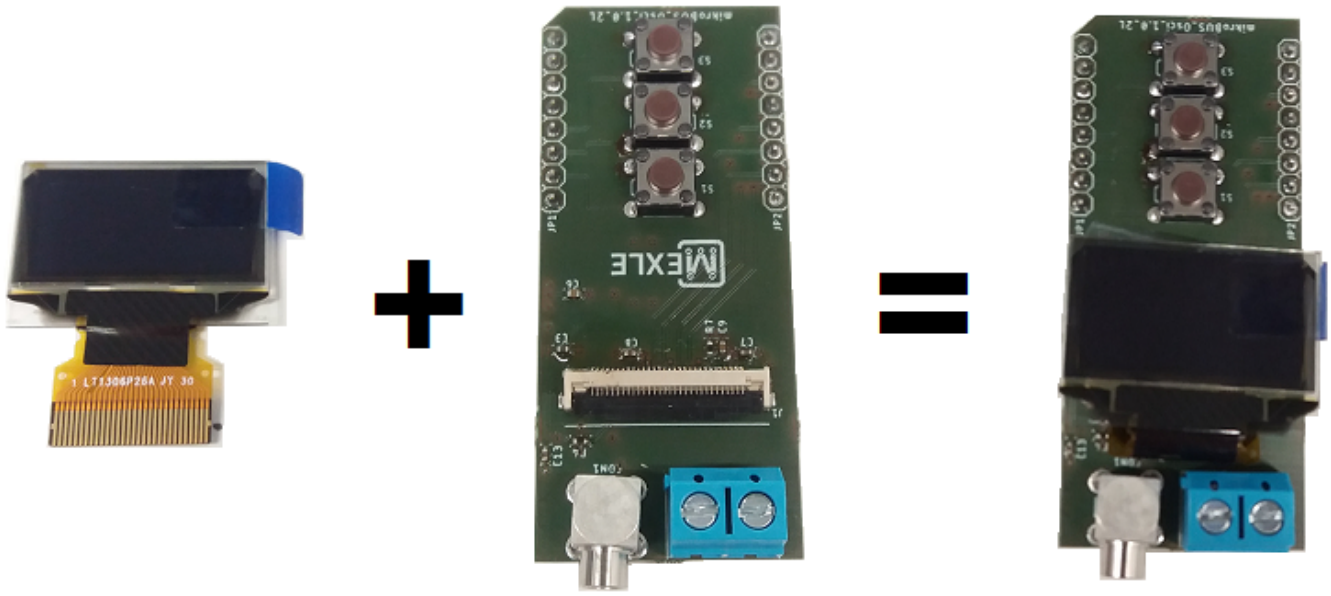


Fig. 13: Einbau des ER-OLED0.96-1.3W mit Klemmverbinder

Spannungsversorgung

Das Display unterstützt zwei Arten der Spannungsversorgung. Bei der ersten erfolgt die Versorgung mit den erforderlichen Spannungen vollständig durch die externe Schaltung. Dies erfordert, dass alle erforderlichen Spannungen extern zur Verfügung stehen. Hierzu ist eine Versorgungsspannung von 1,65 V bis 3,3 V für die Logik des Displays und 7 V bis 15 V zur Ansteuerung des Display Panel zu erzeugen.[33)]

Die zweite Art ist die Nutzung des integrierten DC/DC Wandler. Dieser erzeugt die für die Ansteuerung des Display Panel erforderliche Spannung intern selbst. Vorteil hierbei ist, dass, bei einer Versorgungsspannung von 3,3 V auf der Gesamtplatine des Oszilloskops, welche bei dieser Art der Spannungsversorgung sowohl für die Logik des Displays als auch den DC/DC Wandler verwendet werden kann und somit weitere Hardware zur Spannungswandlung eingespart werden kann.[34)]

4.2 Integration in Vorarbeiten

Durch die Vorarbeiten bei der Entwicklung des Handoszilloskops war bereits ein Gehäuse vorgegeben. Die Abmessung der Platine der vorhandenen Schaltung war auf die Größe dieses Gehäuse ausgelegt worden. Sie verfügte über einen Atmel® SMART SAM L21 Mikrocontroller[35)], zwei LC Displays sowie einen microSD Kartenschacht und eine Batterie.

Nach der Auswahl des PIC32 war zunächst angedacht, diesen in das bestehende Design zu integrieren. Hierbei zeigten sich Probleme durch die starke Komprimierung der Leiterbahnen im Design. Zudem sind die Anschlüsse und Schnittstellen bei dem PIC32 anders verteilt, wodurch die Integration mehr Platz für die Leiterbahnen benötigt als für den SAM L21 vorgesehen war.

Da als Batterie ein 9 V Block zum Einsatz kam, war bereits ein großer Teil des Gehäuses durch diese belegt, der nicht für die Platine verwendet werden konnte. Auch eine Reduktion von zwei auf ein Display sorgte nicht für ausreichenden Platzgewinn, um den ESP32 im Gehäuse unterbringen zu können, ohne dass ein von Grund auf neues Layout für die gesamte Schaltung erforderlich wäre.

Da der Wunsch bestand, die sehr komplexe und hohe Anzahl an Pins erfordernde analoge Schaltung zu vereinfachen wurde die Strategie der Entwicklung geändert.

Die im Projekt zu erstellende Hardware sollte nicht mehr als Integration der neuen Komponenten und dem Austausch der alten Komponenten in die bestehende Schaltung durchgeführt werden. Stattdessen sollten kleine Module mit den Hauptfunktionalitäten des Oszilloskops entwickelt werden. Die Anbindung an den PIC sollte nicht mehr auf der gleichen Platine erfolgen. Stattdessen sollte mittels Developmentkit und geeigneter Aufsteckschnittstelle eine Funktionsdemonstration ermöglicht werden.

Der Schaltplan der durchgeführten Teilintegration ist unter folgendem Link zu finden: Redmine->Handoszilloskop->Schaltplan->alte_ Schaltungen->Oszilloskopstift_Lite

4.3 Development Board

Nach der Anpassung des Projektziels wurde der Fokus des Projektes stärker auf die Oszilloskopfunktionalitäten als auf die Integration in das bestehende Layout gelegt. Darum wurde ein Developmentboard mit dem ausgewählten PIC32 beschafft, das als Basis für die zu entwickelnden Aufsteckplatinen dienen sollte. Als Developmentboard kommt das PIC32MK GP Development Kit (DM320106)[36] zum Einsatz. dieses bietet folgende Anschlüsse und Funktionalitäten.

Anschlüsse und Komponenten

Auf der Oberseite sind folgende Komponenten vorhanden:[37]

1. Mikrocontroller: PIC32MK1024GPE100
2. Grüne Power Indicator LED
3. Shunt Diode der Spannungsversorgung
4. Anschluss für die Spannungsversorgung
5. In-Circuit Serial Programming™ (ICSP™) Verbindung
6. USB Type-C Buchse
7. 120 Ohm Abschlusswiderstände für CAN Bus
8. USB Type-A Verbindung für PIC32 Host-basierte Anwendungen
9. X32 Stiftleisten
10. MikroBUS Schnittstelle
11. 3 konfigurierbare Schalter
12. 3 konfigurierbare LEDs
13. DB-9F CAN Anschlüsse
14. CAN 3 & 4 Header Verbindung

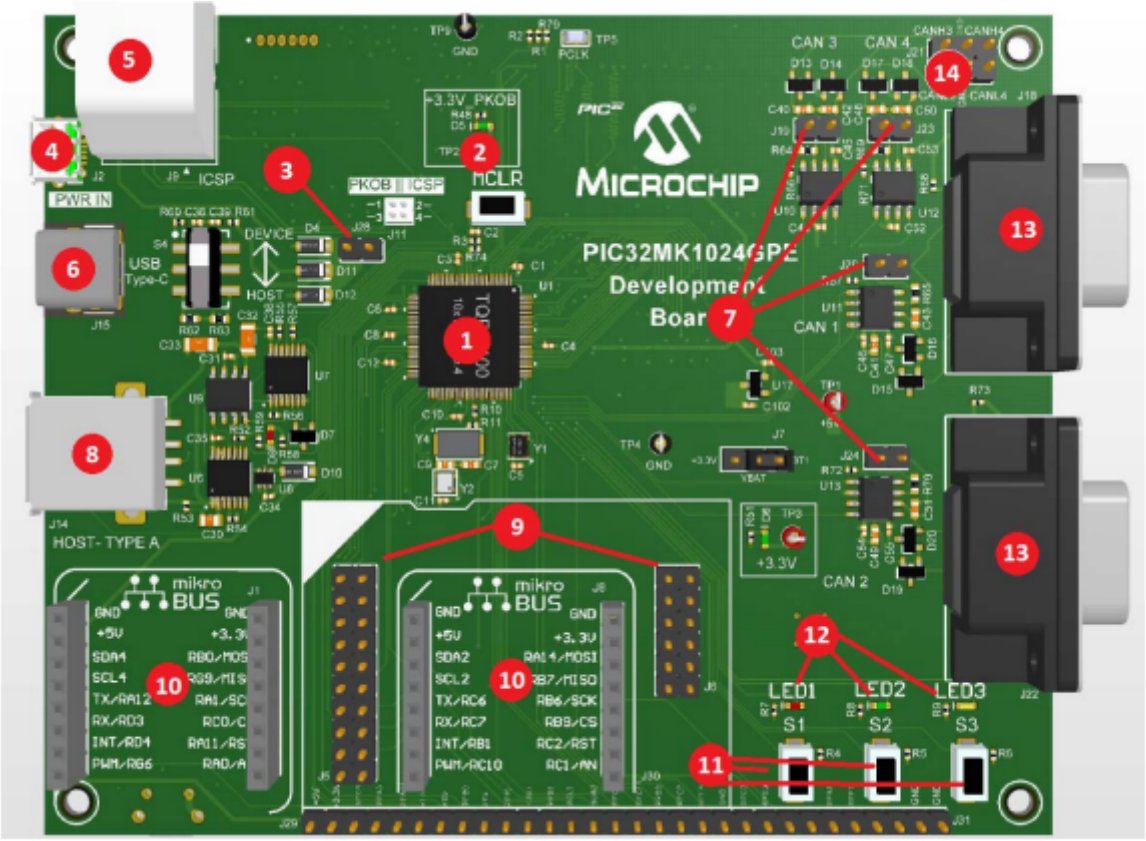


Fig. 14: Oberseite des Developmentboards DM320106

Quelle figure 14:[38)]

Auf der Unterseite sind folgende Komponenten vorhanden:[39)]

1. 50-Pin LCD Buchse
2. SSD1963QL9 Display Controller
3. PIC24FJ256GB106 Debug IC
4. Mini-USB 2.0 Anschluss für Debug Funktionen
5. USB Host and OTG Spannungsversorgung für PIC32 USB Anwendungen
6. Mini-USB 2.0 Anschluss (OTG)

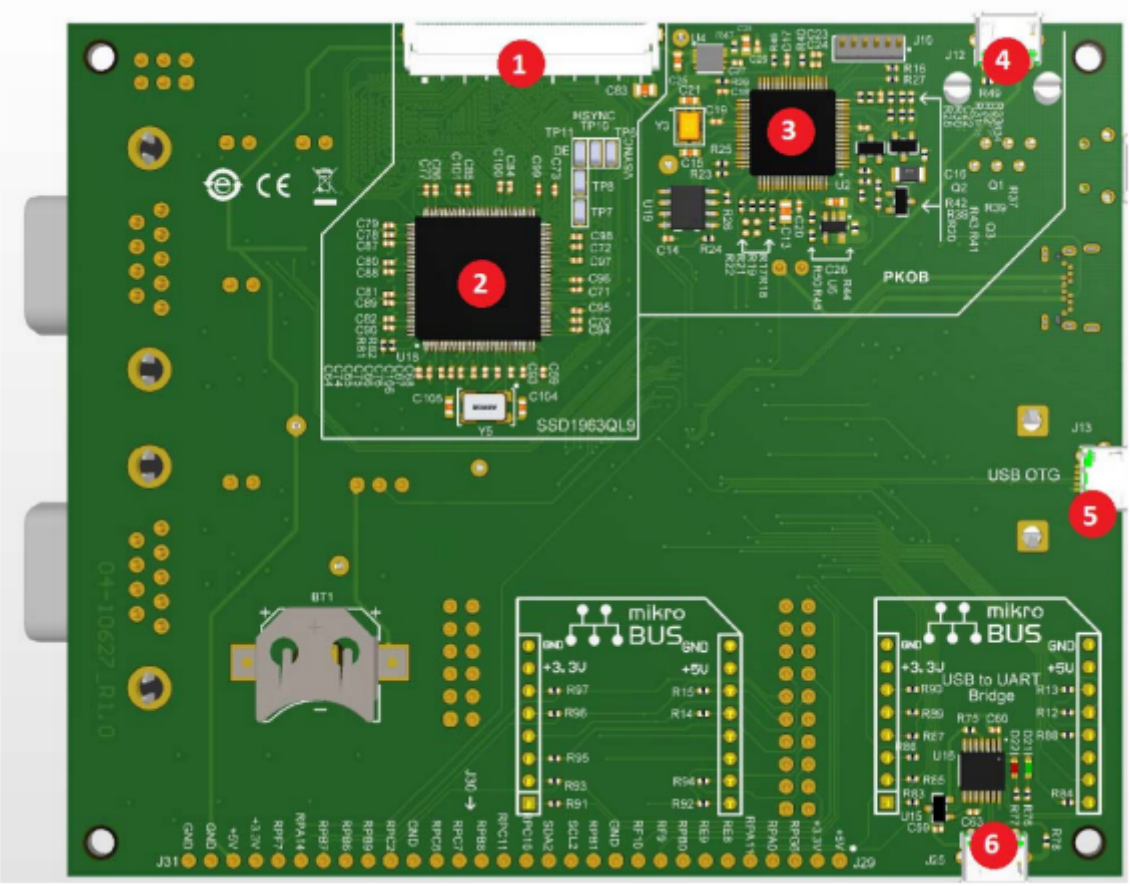


Fig. 15: Unterseite des Developmentboards DM320106

Quelle figure 15: [40)]

PIC32MK1024GPE100

Auf dem Board ist ein PIC32MK1024GPE100 bestückt, dieser ist das leistungsfähigste Modell der PIC32MK Serie [41)]. Hierdurch kann bei der Entwicklung der Software auf die volle Leistungsfähigkeit zurückgegriffen werden. Wird für die finale Version des Handoszilloskops weniger Leistung oder eine geringere Anzahl an Pins benötigt kann innerhalb der PIC32MK Familie auf eine weniger leistungsfähige oder kleine Version skaliert werden. Dies reduziert auch die Kosten eines Exemplars des Handoszilloskops, ohne dass Leistungseinbußen entstehen.

4.4 MikroBUS Prototypen

Wie zuvor beschrieben, wurde für die Tests der grundlegenden Funktionen zunächst auf die Entwicklung eines Gesamtboards verzichtet. Die Funktionalitäten um den PIC32 wurden auf zwei Prototypen Platinen aufgeteilt, um die Funktionen des Oszilloskops einzeln zu testen. Als Basis zum Anschluss der Platinen auf dem Developmentboard wurde der MikroBUS-Standard ausgewählt.

4.4.1 MikroBUS-Standard

MikroBUS ist ein offener Standard entwickelt vom Unternehmen MikroElektronika als eine Erweiterungsschnittstelle für Aufsteckboards. Ziel des Standards ist eine größtmögliche Erweiterbarkeit mit zugleich kleiner Anzahl an Pins zu bieten. Der Standard wird von vielen Herstellern unterstützt und ist auf vielen derer Entwicklungsboards zu finden. Mit sogenannten Click Boards listet MikroElektronika auf seiner Webseite eine große Auswahl an Aufsteckboards dieses Standards auf [42)] [43)].

Platinengrößen

Der Standard definiert für die Aufsteckboards 3 mögliche Platinengrößen. Diese sind S, M und L. Die Breite eines Boards entspricht dabei immer 25,4 mm, die Länge variiert dabei von 28,6 mm bei Platinen der Größe S bis 57,15 mm bei Platinen der Größe L. Zudem definiert der Standard die Lage der Pins sowie die Beschriftung. [44)]

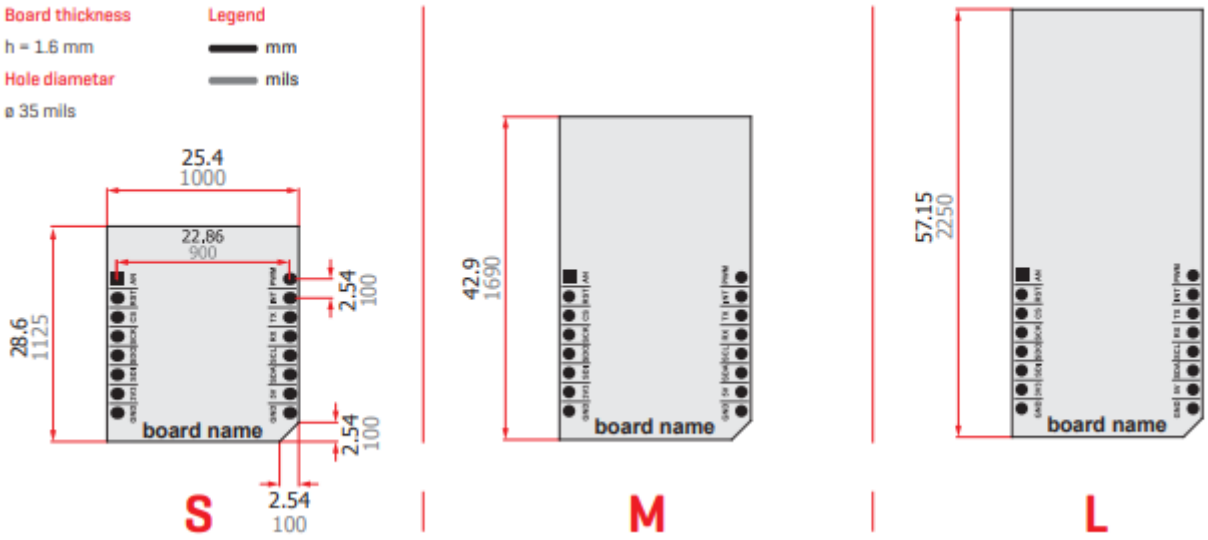


Fig. 16: Platinengrößen nach mikroBUS™ Standard

Quelle figure 16: [45)]

Konfigurierbarkeit

Die MikroBUS Schnittstelle definiert für die Versorgung eine Spannung von 3,3 V und 5 V. Zwei Pins sind für die Masseleitung reserviert. Für alle weiteren Pins gibt es eine Standardbelegung. Diese können bei Bedarf und wenn die Konfiguration des Mikrocontroller es erlaubt auch mit anderen Funktionen (GPIO) belegt werden.[46)]

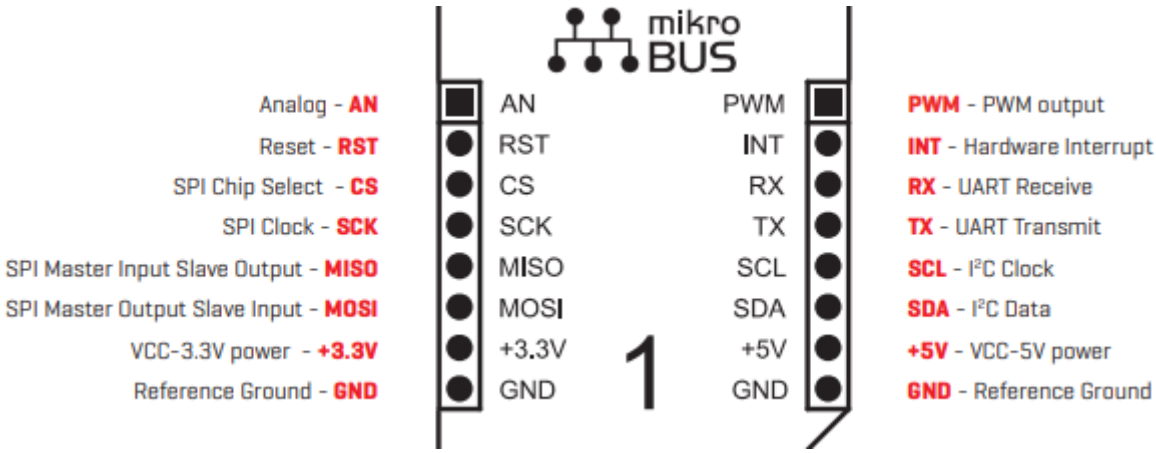


Fig. 17: Pinbelegung eines Boards nach mikroBUS™ Standard

Quelle figure 17:[47)]

4.4.2 Oszilloskop-Modul mit Display

Die erste Platine umfasst den Analog-Eingang-Teil der Oszilloskopfunktion, welcher die

Vorverarbeitung angelegter Spannungen übernimmt und an den Spannungspegel des PIC32 anpasst. Zur Verwendung von Tastköpfen ist eine MCX-Buchse vorhanden. Alternativ können Messleitungen auch an einer zweipoligen Klemme angeschlossen werden. Zulässige Messspannungen sind -10 V bis +10 V. Der angepasste Spannungsegel beträgt 0 V bis 3,3 V.

Zusätzlich zur Analogschaltung umfasst die Platine auch ein OLED-Display welches über einen FPC - Steckverbinder mit 30 Kontakten angeschlossen werden kann. Zur Ansteuerung wird 3 Wire SPI verwendet. Nutzereingaben können über 3 vorhandene Taster erfolgen.

Nachfolgend ist das Pinout der Platine und der bestückte Prototyp abgebildet.

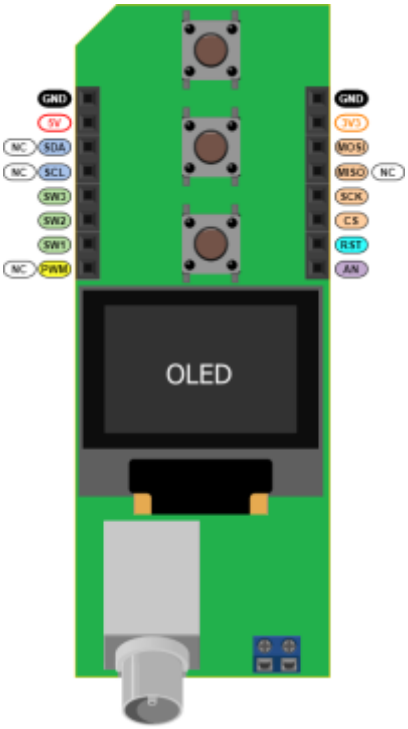


Fig. 18: MikroBUS Oszilloskopplatine

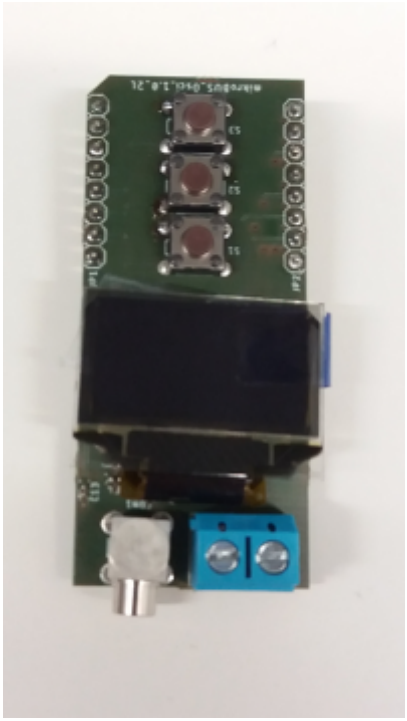


Fig. 19: Bestückte mikroBUS Oszilloskop Platine

Die Schaltpläne können über den Redmine Server unter den folgenden Links heruntergeladen werden (Die Ordner enthalten auch die jeweilige Stückliste inklusive der Preise der Komponenten):

Redmine->Handoszilloskop->Schaltplan->mikroBUS_Oscilloscope_differential

Redmine->Handoszilloskop->Schaltplan->mikroBUS_Oscilloscope

Varianten

Es wurden zwei Varianten der Schaltung erstellt, um unterschiedliche Arten der Messung zu testen. Die erste Schaltung kann für eine Single Ended Messung genutzt werden. Die zweite Schaltung ermöglicht eine differentielle Messung. Gefertigt wurde bisher nur die nicht differentielle Messschaltung.

4.4.3 ESP32-Modul

Die zweite erstellte Prototypenplatine umfasst den ESP32, dieser ist mit SPI, I²C und UART über die MikroBUS Schnittstelle mit dem PIC32 verbunden.

Der Schaltplan basiert auf dem Referenzdesign welches Espressif mit dem Datenblatt des ESP32 liefert und wurde an den MikroBUS Formfaktor angepasst.

Für die Programmierung des ESP32 ist eine JTAG Schnittstelle vorgesehen. Ein Programmer auf der Platine ist nicht vorhanden. Stattdessen muss auf einen externen JTAG Programmer zurückgegriffen werden.

Hierzu kann zum Beispiel der ESP Prog von Espressif Systems verwendet werden. Mit diesem können sowohl ESP32 Modelle als auch die Vorgängerversion ESP8266 programmiert werden. Die Umsetzung erfolgt über eine FTDI USB Bridge und kann über Stecker unterschiedlicher Abmessung durchgeführt werden. Neben der Programmierfunktion ist auch das Debugging über den ESP Prog möglich. [48)], [49)]

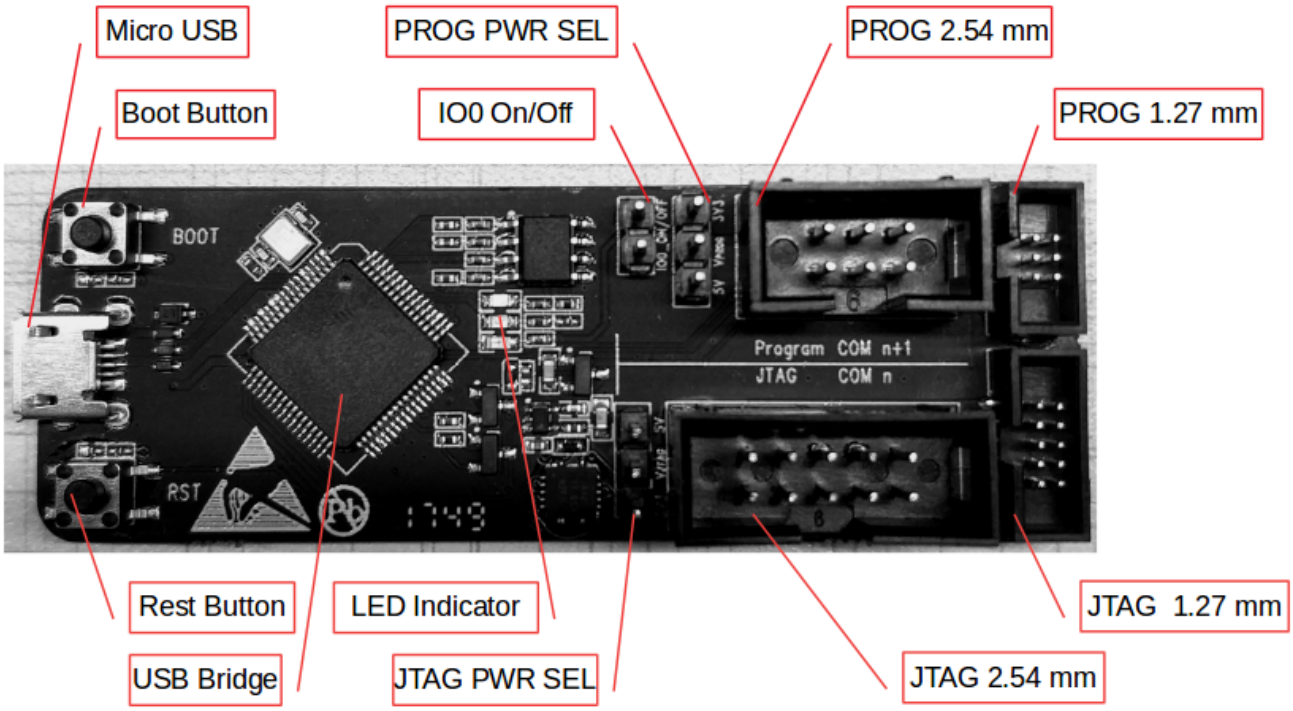


Fig. 20: ESP-Prog Programmer Platine für den ESP32 und ESP8266

Quelle figure 20: [50)]

Zusätzlich zu der JTAG Schnittstelle besitzt die Platine einen Jumper für die Bootoption des ESP32 sowie einen Reset Taster.

Die Platine basiert auf dem MikroBUS Format der Größe M, wurde jedoch etwas in der Länge erweitert, damit zwischen den Bauteilen ein ausreichend großer Abstand für das Auflöten besteht. Die Antenne des ESP32 steht über den Rand der Platine über, damit die Funkverbindung nicht gestört wird. Die nachfolgenden Abbildungen zeigen das Pinout der Platine sowie den bestückten Prototypen.

Fig. 21: Pinout der mikroBUS-ESP32 Platine

Fig. 22: Bestückte mikroBUS-ESP32 Platine

Die Schaltpläne können über den Redmine Server unter den folgenden Links heruntergeladen werden:

Redmine->Handoszilloskop->Schaltplan->mikroBUS_ESP32_4_Layer

Redmine->Handoszilloskop->Schaltplan->mikroBUS_ESP32_2_Layer

Layoutvarianten

Es wurden zwei Layouts der Schaltung erstellt. Der Schaltplan ist in beiden Fällen identisch. Ein Unterschied gibt es bei der Anzahl der verwendeten Schichten bei der Fertigung der Platine. Das erste Layout ist auf ein zweilagiges Design begrenzt und nutzt nur die Ober- und Unterseite der Platine für die Leiterbahnen. Das zweite Design sieht vier Lagen vor. Die inneren Lagen der Platine dienen der Spannungsversorgung mit von 3,3 V sowie als Masselage.

4.5 Akku-Schaltung und Spannungsversorgung

Für spätere Ausbaustufen des Oszilloskops ist der autarke Betrieb ohne Anschluss an einen PC über USB vorgesehen. Hierfür wird eine Batterie oder ein Akku benötigt.

Als Vorbereitung auf den autarken Betrieb wurde eine Batterieschaltung zur Nutzung eines LiPo Akkus entwickelt. Als Basis diente das Design eines Adafruit Arduino Boards [51)]. Die Schaltung ermöglicht die folgenden Betriebsarten:

- Betrieb über USB (hierbei wird das Board mit Spannung versorgt und gleichzeitig der Akku aufgeladen)
- Akkubetrieb (Das Board wird durch den Akku mit Spannung versorgt)

Wird im Betrieb über USB das USB Kabel abgezogen schaltet die Schaltung automatisch auf den Akkubetrieb um. Wird das Kabel wieder eingesteckt, erfolgt die Versorgung wieder über die USB Schnittstelle.

Zum Laden des Akkus kommt ein Lademanagement Controller der Firma Microchip zum Einsatz. Der MCP73831/2 Lademanagement-Controller kann für Schaltungen zum Laden von Lithium Ionen und Lithium Polymer Akkus genutzt werden. Der Ladestrom für den jeweiligen Akku kann über die Auslegung des Ladewiderstands fest eingestellt werden. Bei der Auswahl des Akkus ist dadurch

darauf zu achten, dass dieser auch mit dem festgelegten Strom geladen werden kann, ohne beschädigt zu werden.[52)]

Der Schaltplan wurde erstellt. Ein Layout wurde noch nicht angelegt, da dieses auf die Gesamthardware abgestimmt werden muss und somit erst bei der Erstellung eines Gesamtlayouts mit allen Funktionen des Oszilloskops ein optimales Layout erreicht werden kann.

Der Schaltplan kann über folgenden Link heruntergeladen werden:

Redmine->Handoszilloskop->Schaltplan->Handoszilloskop_Akkuschaltung

4.6 PIC32 Peripherieschaltung und Spannungsversorgung

Im Rahmen der zunächst durchgeführten Integration des PIC32 in die Schaltung aus den Vorarbeiten, wurde ein Schaltplan erstellt, der den PIC32 mit allen notwendigen Versorgungsspannungen, Programmierschnittstellen und Peripherie anbindet.

Dieser kann unter folgendem Link heruntergeladen werden:

Redmine->Handoszilloskop->Schaltplan->PIC32_Peripherie

4.7 Preisvergleich Neuauslegung und Vorarbeiten

Für die in den Vorarbeiten entwickelte Schaltung wurde eine Stückliste erzeugt. Mithilfe dieser wurde eine Preisliste der erforderlichen Komponenten angelegt, um zu prüfen bei welchen Komponenten Einsparungspotential besteht. Nach der Neuausrichtung des Projektziels wurden die meisten Komponenten neu ausgewählt. Daher war ein Preisvergleich zwischen der neuen Version und der Vorarbeit sinnvoll, um die Veränderung des Stückpreises untersuchen zu können.

Die Preise, die im Folgenden genannt werden, beziehen sich nur auf die Komponenten, die auf der Platine bestückt werden. Der Preis für die Fertigung und Bestückung der Platine selbst sind darin nicht enthalten. Die Preise wurden auf Basis der Fertigung von 10 Oszilloskopen ermittelt. Die unten aufgelisteten Preisangaben beziehen sich jeweils auf ein Stück.

Die Version der Vorarbeit ergab für den Preis der Komponenten für die Herstellung einen Betrag von ca. 62 €. Dieser setzt sich zusammen aus:

- Preis Mainboard : 49,47 €
- Preis LCD und SD-Karte: 12,04 €
- resultierender Gesamtpreis: 61,51 € pro Stück bei Fertigung von 10 Oszilloskopen

Für die Neuauslegung wurde ein Preis von ca. 35 € pro Stück ermittelt. Dieser setzt sich aus den folgenden Teilen zusammen:

- ESP32 MikroBus: 5,58 € bzw. bei reiner Nutzung der USB Schnittstelle 0€
- Oszilloskop MikroBUS: 9,39 €, bei optionaler MCX Buchse reduzierbar auf 6,40 €
- PIC32 + Peripherie: 10,37 €, Einsparung durch kleineres PIC32 Modell möglich
- Spannungsversorgung Akku: 9,30 €
- resultierender Gesamtpreis: 34,64 € pro Stück bei Fertigung von 10 Oszilloskopen

Die Preislisten sind im jeweiligen Ordner der Schaltpläne als Teil der Stückliste enthalten.

Der Preis hat sich durch die Neuauslegung um ≈43 % von 62 € auf ca. 35 € pro Stück verringert und

erfüllt damit deutlich besser das gesetzte Preisziel von maximal 60 € für ein Exemplar.

Durch die Fertigung einer größeren Stückzahl, von z.B. 100, können weitere Einsparpotentiale bei der Beschaffung der Komponenten ausgeschöpft werden.

Zusätzlich besteht Einsparpotential durch Modellvarianten. Zum einen kann durch die Wahl eines kleineren PIC32 Modells mit geringerer Pin Anzahl der Anteil des PIC32 an den Beschaffungskosten reduziert werden. Wird nur die Klemme zum Anschluss der Messobjekte verwendet und ist die MCX Buchse optional kann der Preis des Oszilloskopmoduls auf 6,40 € gesenkt werden. Ein Verzicht auf Drahtlose Konnektivität und dadurch ausschließliche Nutzung der USB Schnittstelle zur Kommunikation kann weitere 5,58 € einsparen. Ein Handoszilloskop mit minimaler Ausstattung wäre bereits ab ca. 26 € möglich.

4.8 Weitere Entwicklung der Hardware

Die erforderlichen Hardwareschaltungen für den Test einer Oszilloskopfunktion wurden in einem Schaltplan umgesetzt. Realisiert wurde diese Funktion mittels zwei Prototypenplatinen, die zusammen mit einem Developmentboard eingesetzt werden können. Es wurde ein Schaltplan für die Spannungsversorgung über USB und mit Akkubetrieb erstellt. Für die Verwendung mit dem Developmentkit ist diese nicht erforderlich und wurde daher nicht als Layout umgesetzt. Die nachfolgende Abbildung zeigt die auf das Developmentboard aufgesteckten mikroBUS-Prototypen die im Rahmen des Projektes hergestellt wurden.

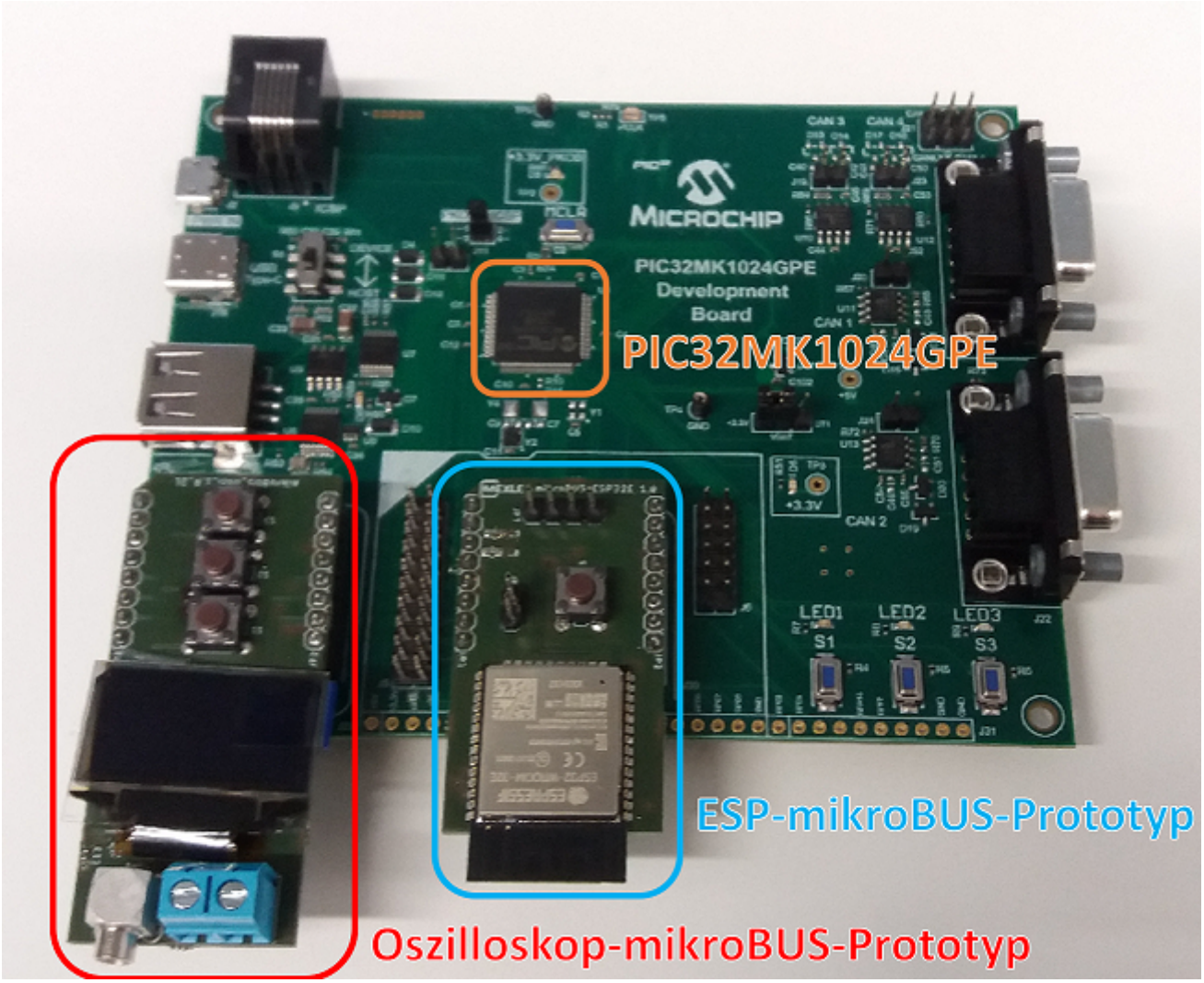


Fig. 23: Developmentboard mit aufgesteckten mikroBUS-Prototypen

Der nächste Schritt der Hardwareentwicklung besteht darin, die Schaltpläne der Prototypen zusammen mit der Schaltung zur Spannungsversorgung und Akkubetrieb in ein gemeinsames Layout zu überführen.

5. Software

Die Software ist in verschiedene Bereiche gegliedert. Die Software für den PIC32, die Programmierung des ESP32-Moduls und die Software-Schnittstelle für die Anbindung an einen PC.

5.1 PIC32

Die Software für den PIC32 umfasst die Funktionen zur Erfassung der Messwerte, Umsetzung der Nutzereingaben die über die Taster erfolgen und die Kommunikation mit dem Display sowie dem PC (über USB) und dem ESP32. Programmiert wird der PIC32 über die von Microchip veröffentlichte Entwicklungsumgebung MPLAB X IDE.

5.1.1 MPLAB X

5.1.1.1 Benutzeroberfläche

Im folgenden Abschnitt werden die wichtigsten Funktionen von MPLAB und Harmony beschrieben. figure 24 zeigt die Oberfläche von MPLAB, die nummerierten Elemente des Bildes werden genauer beschrieben.[53)]

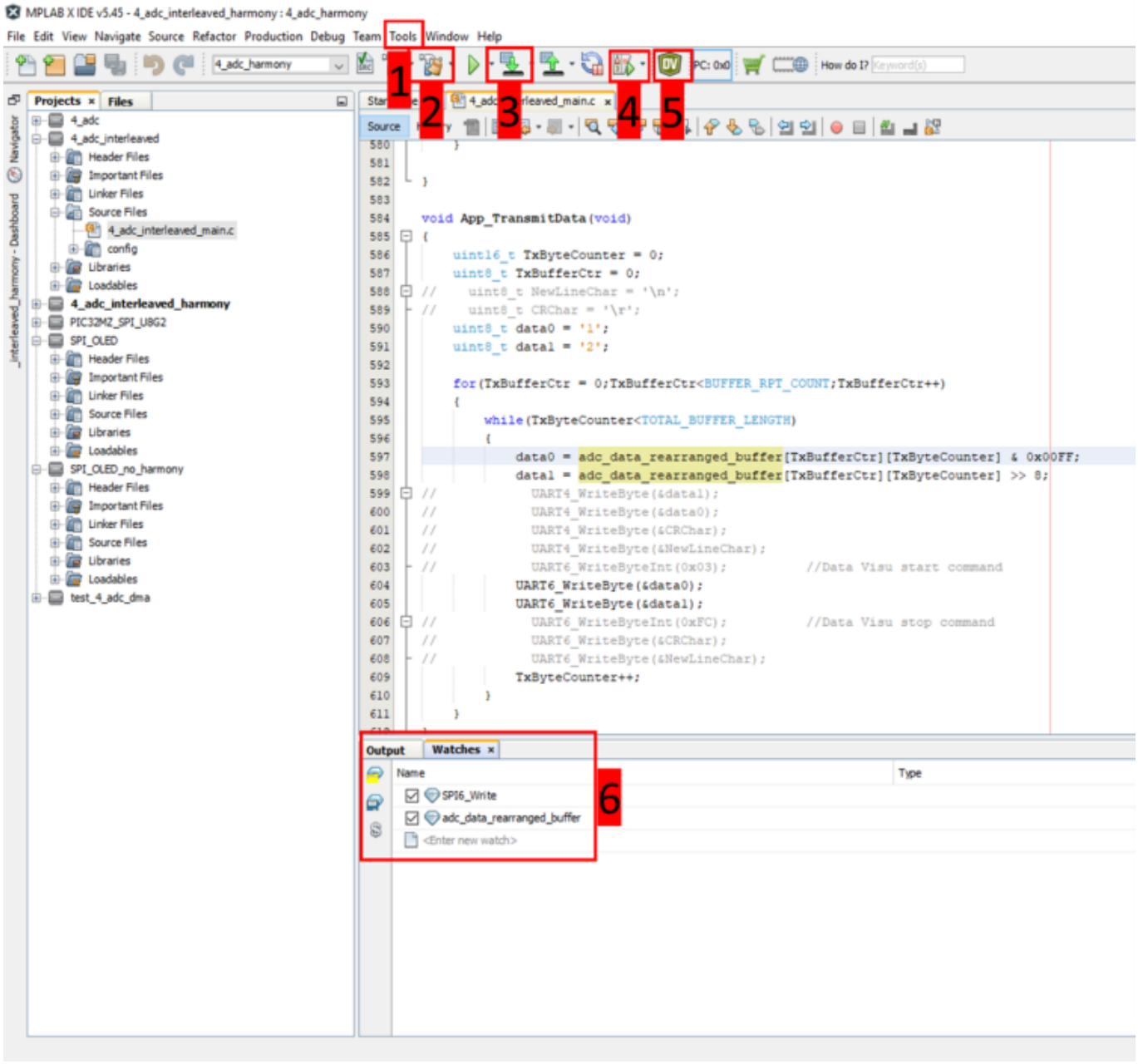


Fig. 24: MPLAB Benutzeroberfläche

1. Tools: Hier kann auf verschiedene Unterprogramme zugegriffen werden, sowie deren Installation verwaltet werden. MPLAB Harmony und der Data Visualizer müssen vor der Verwendung installiert werden.
2. Clean & Build: Über diese Schaltfläche wird der Programmcode erstellt und auf Fehler geprüft. Erstellt wird das momentan ausgewählte Hauptprojekt, nicht die momentan geöffnete main-Datei. (Das Hauptprojekt ist in der linken Spalte des Bildes Fett hervorgehoben)
3. Make and Program Device: Der Code wird erstellt und auf Fehler geprüft, anschließend wird der Code auf den µC geladen. Dafür muss das Development-Kit mit dem PC verbunden sein, sowie in der Konfiguration als Hardware-Tool ausgewählt sein.
4. Debugger: Damit kann das Programm debuggt werden.
5. Data Visualizer: Darstellung der Datenströme des µC. (Muss erst heruntergeladen und intalliert werden)
6. Output & Variablen Watchlist: Output zeigt den Status des Programms an, z.B. Fehler beim Kompilieren, Verbindung zum Board, etc. mithilfe der Variablen Watchlist kann während des Debuggens der momentane Wert der Variablen ausgegeben werden. (Rechtsklick auf neue Variable -> New Watch)
7. Sonstiges:

1. Über den Tab Production können der verwendete Prozessor und Compiler ausgewählt werden.

5.1.1.2 Projekt anlegen

Für die Installation wichtiger Komponenten und das Anlegen von neuen Projekten wurde von MicroChip ein Tutorial für das Erstellen eines Beispielprojektes veröffentlicht. Dieses Tutorial ist unter folgendem Link zu finden: PIC32MK Beispielprojekt

Die Schritte zum Erstellen eines Projekts für den bis jetzt verwendeten PIC32MK1024GPE100 werden schrittweise erläutert:

1. Unter File -> New Project... auswählen
2. Das 32-bit MPLAB Harmony 3 Project auswählen
3. Den Harmony Framework Pfad angeben. (Harmony muss vorher installiert werden)
4. Dem neuen Projekt einen Ordner- und Projektnamen geben.
5. Der Konfigurationsdatei einen Namen geben und den gewollten Chip auswählen.
    1. Es wird empfohlen, nicht den Namen „default“ für die Configuration zu verwenden, da Konfigurationen mit demselben Namen für mehrere Projekte gelten können.
6. Es wird ein leeres Projekt erstellt. Soll Harmony für die Programmierung verwendet werden, muss es als nächster Schritt gestartet werden. Die weiteren Schritte werden in den folgenden Kapiteln erläutert. Ansonsten müssen die „.c-“ und „.h-“ Dateien manuell hinzugefügt werden.[54)]

5.1.1.3 PIC32 config settings

Die Konfiguration des PIC32 kann manuell oder über Harmony vorgenommen werden. Wird Harmony nicht verwendet, können die Settings über Window-> Target Memory Views->Configuration Bits geändert werden. So muss nicht jede Option im Datenblatt nachgelesen werden. Sind die gewünschten Änderungen vorgenommen, kann über die Schaltfläche “Generate Source Code to Output” der Code erzeugt werden. Um den Code dem Projekt hinzuzufügen muss er mit Copy-Paste aus dem Outputfenster in ein Projektfile eingefügt werden. Alternativ kann mit Rechtsklick->Save as... ein separates File erzeugt werden, welches in das Projekt eingebunden werden kann. [55)]

5.1.1.3 MPLAB Harmony

Im folgenden Abschnitt wird auf die Details der Harmony Benutzeroberfläche eingegangen. figure 25 zeigt die Standardansicht mit einigen bereits eingefügten Modulen des PIC32MK.

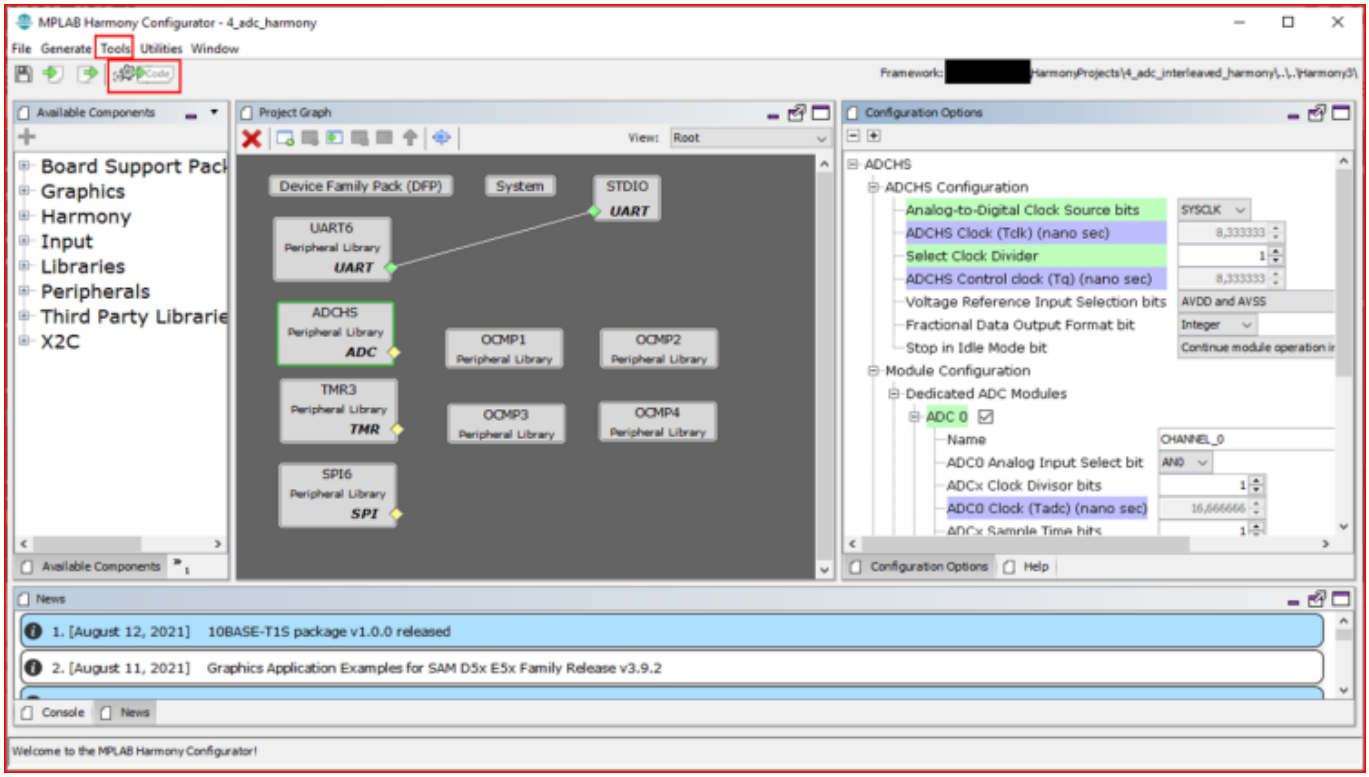


Fig. 25: Harmony Benutzeroberfläche

Um neue Module der Programmierung hinzuzufügen, genügt es, sie aus der Liste am linken Bildrand in das graue Feld zu ziehen. Wählt man einen Block mit Linksklick aus, erscheint am rechten Bildrand die Einstellungsoptionen für diese Komponente. Ein einfaches Beispiel von MICROCHIP ist unter folgendem Link zu finden: PIC32MK Beispielprojekt

Manche Komponenten, wie die ADCHS, Clock, Interrupts und Pinzuweisung haben gesonderte Oberflächen zur Konfiguration, welche unter Tools zu finden sind. ADCHS und Pinzuweisung werden nachfolgend erklärt, der Rest erfordert im Grunde keine weiteren Erklärungen. Um den in Harmony konfigurierten Code zu erstellen, muss auf das „Generate Code“ Symbol geklickt werden. Dann erstellt MPLAB eine main.c-Datei und die Bibliotheken der ausgewählten Module.[56)]

5.1.1.4 Harmony ADCHS

figure 26 stellt die Oberfläche zur Konfiguration der ADC des PIC32MK dar.

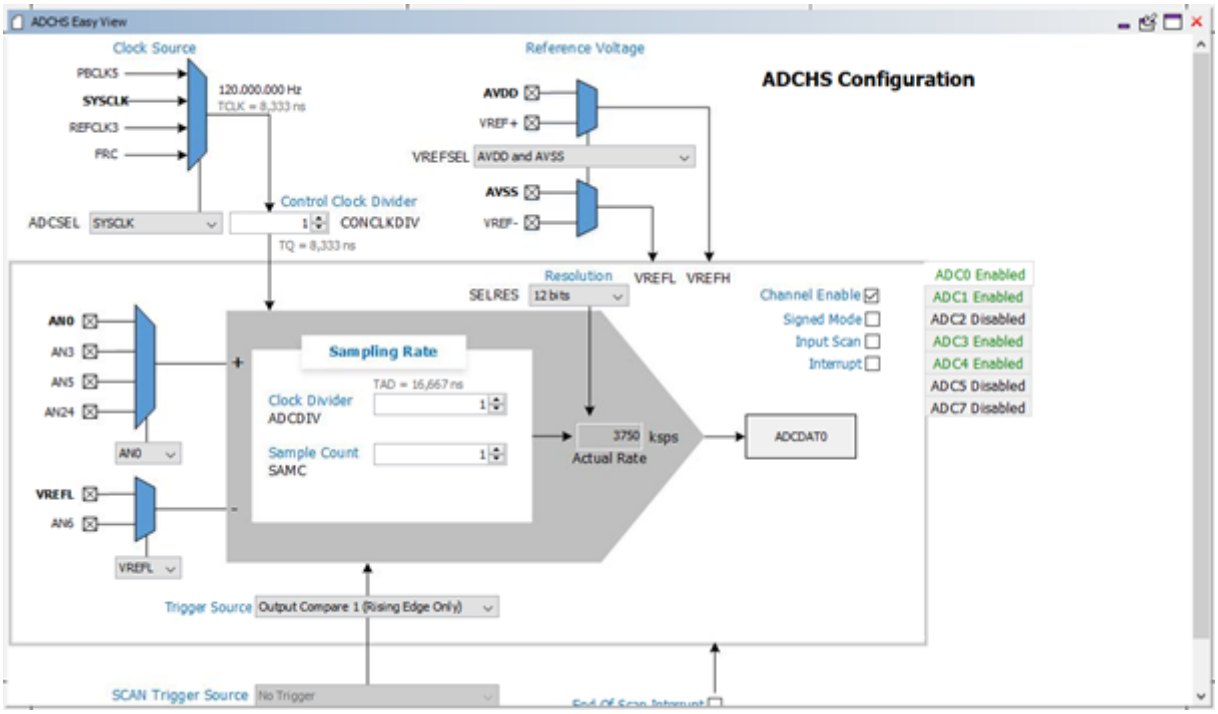


Fig. 26: Harmony ADCHS

Über diese Oberfläche lässt sich leicht die Auflösung, Sampling-Rate, Triggerquelle und analoger Eingang eines ADC auswählen. Bei Verwendung mehrerer ADC ist darauf zu achten, dass ein Haken bei Channel Enable gesetzt ist. Für die Verschachtelung mehrerer ADC-Kanäle muss derselbe analoge Eingang ausgewählt werden.

5.1.1.5 Harmony Pinzuweisung

Für die Pinzuweisung in Harmony sind mehrere Oberflächen vorhanden, mit denen sich die Konfiguration visualisieren lässt. Diese sind in figure 27 bis figure 29 dargestellt.

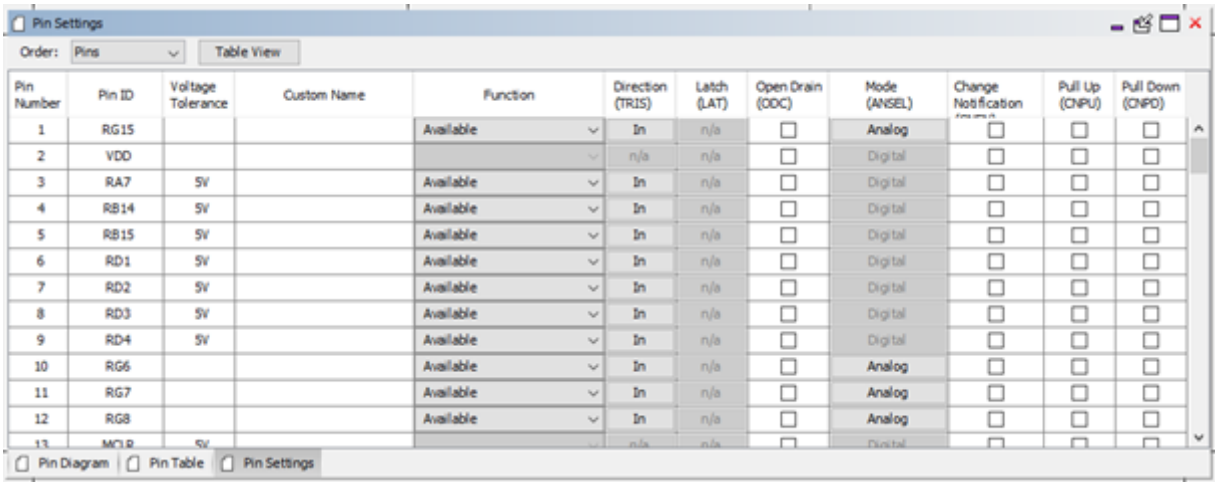


Fig. 27: Harmony Pin Settings

Im Pin Settings Menü können den einzelnen Pins Funktionen (z.B. UART oder SPI) und selbst erstellte Namen zugewiesen werden. Außerdem kann definiert werden, ob es sich um einen Input oder Output handelt und ob ein Pull Up oder Pull Down Wiederstand verwendet werden soll.

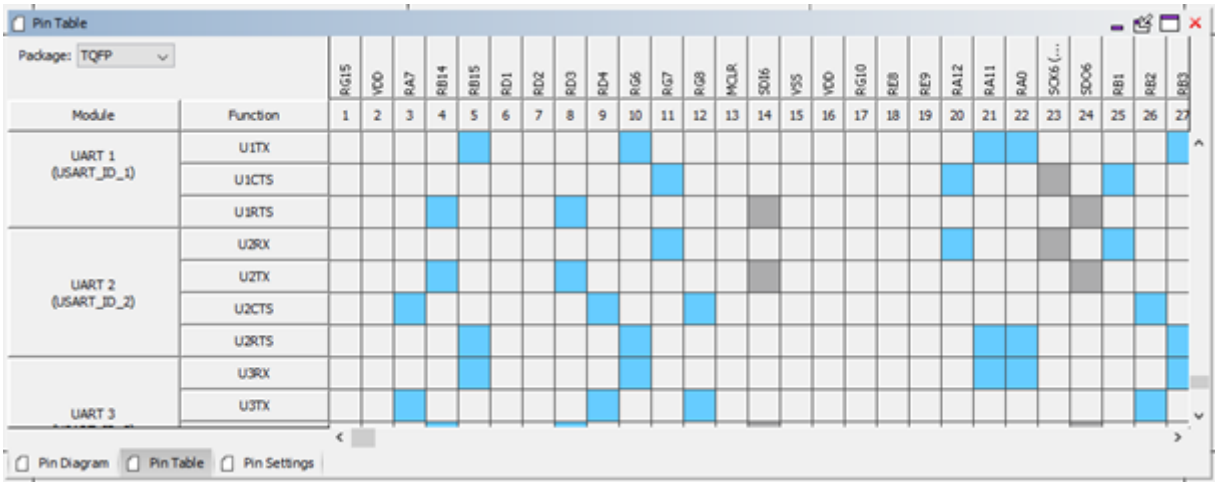


Fig. 28: Harmony Pin Table

Die Pin Table stellt dar, welchem Modul welche Pins zur Verfügung stehen. Pins, welche bereits von anderen Modulen verwendet werden, werden ausgegraut. Als Beispiel kann UART2 Pin RB14 oder RD3 zum Senden (U2TX) verwenden.

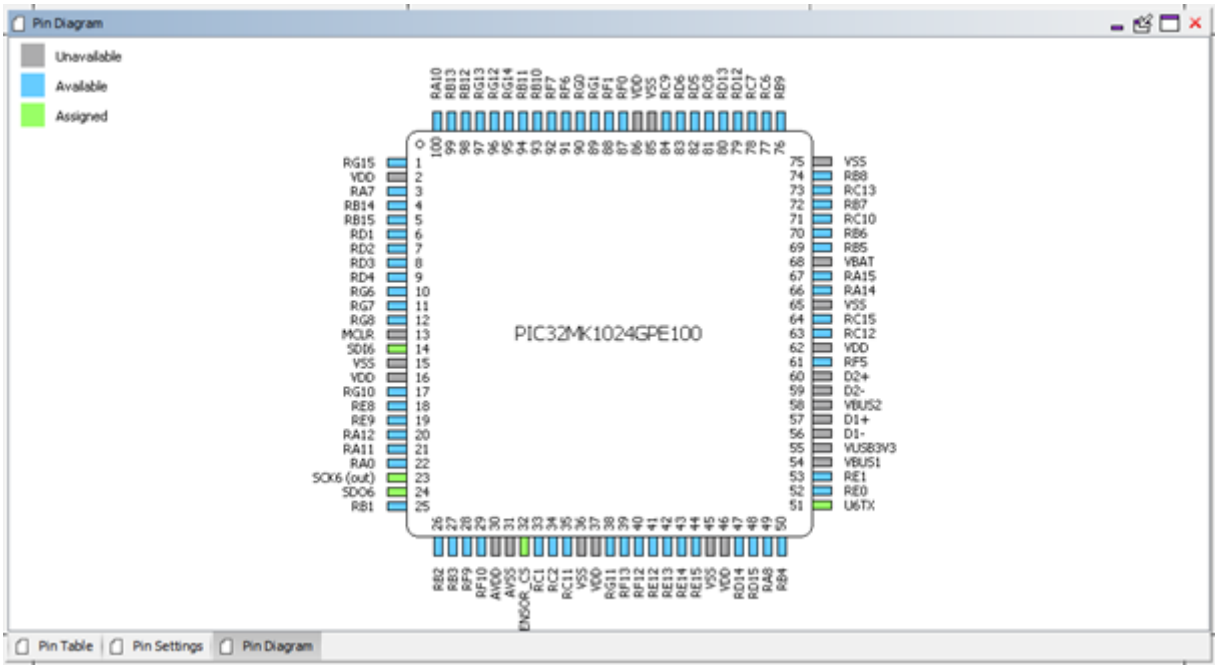


Fig. 29: Harmony Pin Diagram

Die Position und die Funktion der einzelnen Pins werden im Pin Diagram dargestellt. Damit lässt sich nachvollziehen, welche Verbindungen noch zur Verfügung stehen.

5.1.1.6 Bare Metal Pinzuweisung

Die Definition von Input oder Output wird über die TRISx-Register definiert, eine 1 seht für Input, eine 0 für Output. Das x gibt die einzelnen Port Module an und kann beim PIC32MK die Werte A bis D annehmen. Im Code sieht das folgendermaßen aus:

```
TRISAbits.TRISA15 = 1;
```

Für die Zuordnung von Funktionen zu den Pins müssen den Pins bzw. Registern bestimmte Zahlenwerte zugewiesen werden. Inputs und Outputs unterscheiden sich in der Art der Zuweisung. Die Bedeutung der Zahlenwerte ist im Datenblatt zu finden.[57)]

- Input:
  - Den Modulregistern wird ein Pin zugewiesen
  - Beispiel: UART 6 Empfangen an Pin A15: U6RXR = 13
- Output:
  - Dem Pin-Register wird ein Peripheriemodul zugewiesen
  - Beispiel: UART 6 Senden an Pin A4: RPA4R = 11

Weitere Infos sind unter folgendem Link zu finden: IO-Ports

5.1.1.7 Interrupts

figure 30 zeigt die Harmony Oberfläche zur Konfiguration der Interrupts. Es kann eingestellt werden, welche Interrupts verwendet werden sollen und welche Priorität sie haben.

EVIC Settings

Vector Number	Vector	Use	Priority (1 = Lowest)	SubPriority	Handler Name
0	CORE_TIMER (Core Timer Interrupt)	☐	1	0	CORE_TIMER_Handler
1	CORE_SOFTWARE_0 (Core Software Interrupt 0)	☐	1	0	CORE_SOFTWARE_0_Handler
2	CORE_SOFTWARE_1 (Core Software Interrupt 1)	☐	1	0	CORE_SOFTWARE_1_Handler
3	EXTERNAL_0 (External Interrupt 0)	☐	1	0	EXTERNAL_0_Handler
4	TIMER_1 (Timer1)	☐	1	0	TIMER_1_Handler
5	INPUT_CAPTURE_1_ERROR (Input Capture 1 Error)	☐	1	0	INPUT_CAPTURE_1_ERROR_Handler
6	INPUT_CAPTURE_1 (Input Capture 1)	☐	1	0	INPUT_CAPTURE_1_Handler
7	OUTPUT_COMPARE_1 (Output Compare 1)	☐	1	0	OUTPUT_COMPARE_1_Handler
8	EXTERNAL_1 (External Interrupt 1)	☐	1	0	EXTERNAL_1_Handler
9	TIMER_2 (Timer2)	☐	1	0	TIMER_2_Handler
10	INPUT_CAPTURE_2_ERROR (Input Capture 2 Error)	☐	1	0	INPUT_CAPTURE_2_ERROR_Handler
11	INPUT_CAPTURE_2 (Input Capture 2)	☐	1	0	INPUT_CAPTURE_2_Handler
12	OUTPUT_COMPARE_2 (Output Compare 2)	☐	1	0	OUTPUT_COMPARE_2_Handler
13	EXTERNAL_2 (External Interrupt 2)	☐	1	0	EXTERNAL_2_Handler
14	TIMER_3 (Timer3)	☐	1	0	TIMER_3_Handler

Fig. 30: Harmony Interrupts

Nicht alle vom PIC32MK unterstützten Interrupts sind in Harmony implementiert, zum Beispiel fehlt der ADC DMA Datentransfer, welcher aber nach Initialisierung des Harmony-Codes manuell hinzugefügt werden kann. Dieser Interrupt triggert, sobald die Zwischenspeicher der ADC-Module voll sind und wird für die im Projekt notwendige hohe Sample-Rate benötigt.[58)]

5.1.1.8 Unterschied Harmony und MPLAB Code Configurator

Die MPLAB X IDE unterstützt zwei visuelle Code Konfiguratoren, MPLAB Harmony und MPLAB Code Configurator (MCC). Die beiden Konfiguratoren dienen grundsätzlich demselben Zweck, unterscheiden sich aber in den unterstützten Prozessoren und den dadurch unterschiedlichen Konfigurationsmöglichkeiten. Auch die visuelle Darstellung ist eine andere. Der PIC32MK wird nur von MPLAB Harmony unterstützt, MCC ist eher für Baureihen bis PIC18 verwendbar.[59)], [60)] Wird zur Programmierung nicht Harmony oder MCC verwendet, können die Benutzeroberflächen bei der Programmierung helfen, indem man zum Beispiel die verfügbaren Pins und deren Namen nachschauen kann, ohne die Datenblätter durchsuchen zu müssen.

5.1.1.9 Data Visualizer

Der Data Visualizer figure 31 kann die vom PIC32MK Development Kit übertragenen Daten in Echtzeit empfangen und darstellen. Es besteht die Möglichkeit die Daten in Form eines Graphen oder als

Rohdaten anzeigen zu lassen. Die Darstellung der Rohdaten kann als Integer, ASCII oder Hex-Wert ausgegeben werden.[61)]

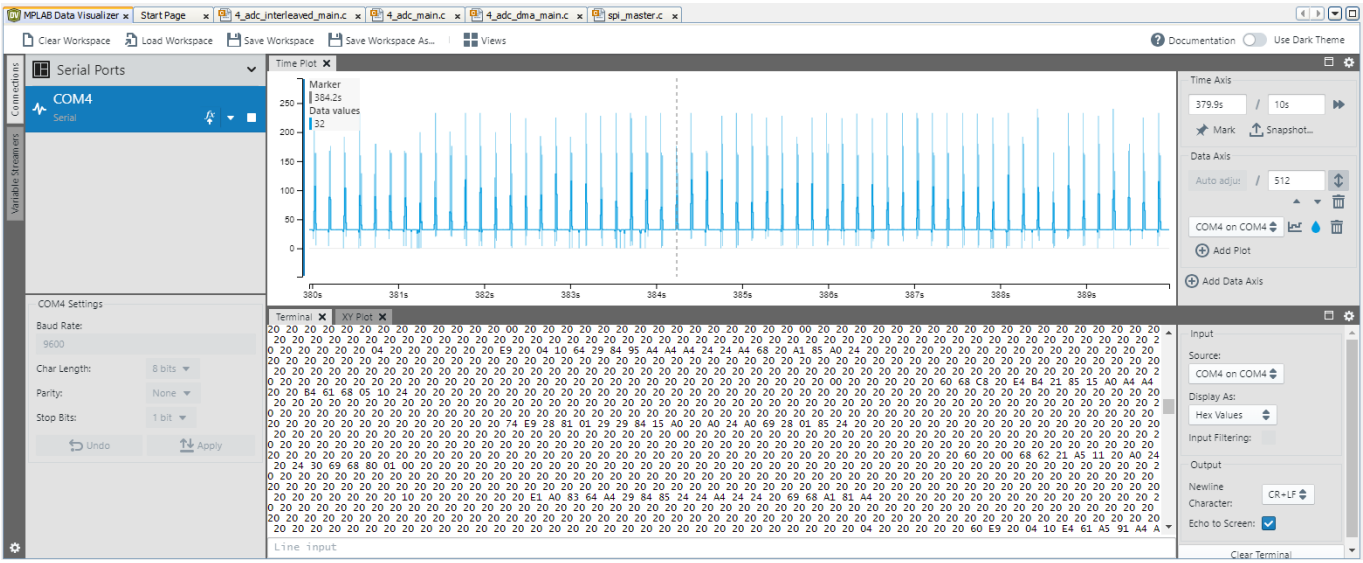


Fig. 31: Data Visualizer

Je nach gesendeten Daten muss vor und nach der Übertragung eine Bitfolge gesendet werden, damit der Data Visualizer die Daten korrekt anzeigen kann. Der Ablauf der Übertragung sieht dann beispielweiße folgendermaßen aus:

```
UART1_Write(0x03); // Data Visualizer Start command
UART1_Write(ADC_Value);
UART1_Write(0xFC); // Stop command
```

Als Alternative zum Data Visualizer können auch externe Programme wie TerraTerm, RealTerm oder PuTTY verwendet werden, welche sich in ihren Funktionsumfängen und Darstellungsmöglichkeiten unterscheiden. Details dazu sind in Kapitel 5.1.4 zu finden.

5.1.1.10 MPLAB und Harmony Eigenheiten

Nachfolgend werden einige Besonderheiten bei der Bedienung von MPLAB aufgelistet, welche während der Bearbeitung des Projekts festgestellt worden sind:

- Kombination von Bibliotheken von Harmony und extern nur schlecht möglich, der von Harmony generierte Code blockiert alle Module des Prozessors, die nicht hinzugefügt wurden.
- Move und Copy von Projekten nicht innerhalb des Programms ausführen, sonst unerwünschte Nebenwirkungen. (Copy führt zu Verschachtelung des Projektordners, kopiertes Projekt erzeugt Unterordner, keinen neuen Projektordner)
- Harmony Einstellungen werden über Project-Properties und Configuration gespeichert. Inkonsistentes Verhalten bei Move und Copy, Configuration wird für kopiertes Projekt übernommen, bei Veränderungen der Configuration werden beide Projekte geändert. Besser: neues Projekt und Configuration anlegen.
- Wenn ein Laptop mit zweitem Bildschirm verwendet wurde, kann es passieren, dass falls der zweite Bildschirm nicht mehr angeschlossen ist und MPLAB mit Harmony und dessen Oberflächen neugestartet wird die Position der Fenster gespeichert wurde und auf dem einzelnen Bildschirm nicht alle geöffneten Fenster angezeigt werden.
- Bei Verwendung eines Laptops oder kleinen Bildschirms können manche Menüoptionen

abgeschnitten sein. (Beispiel: Rechtsklick-Menü der Projektliste.)
- Bei Anschluss des Development-Kit über USB-Hub mit zusätzlichem USB-Kabel für das Senden von Daten an den PC wird die Verbindung nicht erkannt.

5.1.2 MPLAB Beispielcode

Es können mehrere Code-Beispiele über den Harmony 3 Content Manager heruntergeladen werden. Diese sind Teil der verschiedenen Packages, welche für den Betrieb von Harmony mit spezifischen Chips heruntergeladen werden können. Die Beispiele sind im Benutzer Ordner des Computers zu finden, für gewöhnlich im Ordner “C:\Users\Name\Harmony3”. Es können mehrere Packages heruntergeladen werden. Dazu gehören core, csp und gfx Beispiele. Die Beispielprojekte sind in den Unterordnern mit Namen „x_apps_pic32mk“ gespeichert und können von dort importiert werden. Beispiele für das im Projekt verwendete Development Kit haben den Projektnamen „pic32mk_gp_db.X“. Andere Beispiele können auch geladen werden, können sich aber in den verwendeten Modulen unterscheiden, der PIC32MK1024GPE100 besitzt z.B. kein separates Modul für PWM-Steuerung. Beschreibungen der einzelnen Projekte sind in den Unterordnern unter „docs“ zu finden. Alternativ ist die Beschreibung der Projekte unter folgendem Link zu finden: MPLAB Harmony Application Examples|MPLAB Harmony Application Examples[62)]

5.1.3 Handoszi Programmierung

Für die Programmierung des Handoszilloskop existieren mehrere Teilprogramme, dies beinhaltet unterschiedliche Versionen der AD-Wandlung, sowie Kommunikation mit dem OLED-Display über verschiedene Bibliotheken. Das nachfolgende Kapitel beschreibt die unterschiedlichen Programmierungen und deren Vor- beziehungsweise Nachteile gegeneinander.

OLED

Test-Display

Um die Software für die Displays vor der Erstellung der mikrobus-Platinen testen zu können, wurden zwei OLED-Displays beschafft, welche ebenfalls den SSD1306 Displaytreiber verwenden, allerdings schon mit Pins für die Montage auf Lochrasterplatinen vorkonfektioniert sind. Diese Displays besitzen sieben Pins für den Anschluss von Spannungsversorgung und SPI-Leitungen Der Anschluss ist wie folgt:

- GND: Masse
- VCC: Spannungsversorgung von 3,3 V bis 5 V
- D0: Taktungsleitung (SPI: SCK, I2C: SCL)
- D1: Datenleitung (SPI: MOSI, I2C: SDA)
- RES: OLED Reset, Display einmal resetten nach Aktivierung
- DC: SPI: MISO, Adressauswahl I2C
- CS: SPI Chip Select, falls nicht verwendet: Verbindung mit GND

Die OLED-Displays können wie in der Spezifikation beschrieben durch Umlöten von Widerständen auf der Rückseite mit I2c, 3 wire SPI oder 4 wire SPI betrieben werden. Standardkonfiguration ist 4 wire SPI.[63)]

Adafruit Library

Redmine->Handoszilloskop->Code->PIC32MK_GP->SPI_OLED:
Verwendet wird die SSD1306 Display Treiber Software von Adafruit[64)] mit den von Harmony erzeugten Funktionen für SPI und Timer.[65)] Das Display zeigt keine Reaktion auf Signale. Da das Display keine

Reaktion zeigt wurde ein weiteres Programm erstellt, welches für die Kommunikation selbst geschriebene Funktionen verwendet. Allerdings brachte auch diese Version nicht den gewünschten Erfolg. Diese Version des Programms ist unter folgendem Link zu finden: Redmine->Handoszilloskop->Code->PIC32MK_GP->SPI_OLED_no_harmony

U8G2Lib

Redmine->Handoszilloskop->Code->PIC32MK_GP->PIC32MZ_SPI_U8G2:
Im nächsten Schritt wurde die Implementierung der U8G2 Library[66] für PIC32 getestet. Diese beinhaltet Software für verschiedene Display Treiber. Das Beispiel [67] wurde ursprünglich für den PIC32MZ programmiert, kann mit minimalen Änderungen der Konfigurationsbits auch mit dem PIC32MK verwendet werden. Ergänzt wurde die Pinbelegung, so dass das OLED-Display an mikroBus Steckplatz J1 angeschlossen werden kann. Das ist in figure 32 dargestellt.

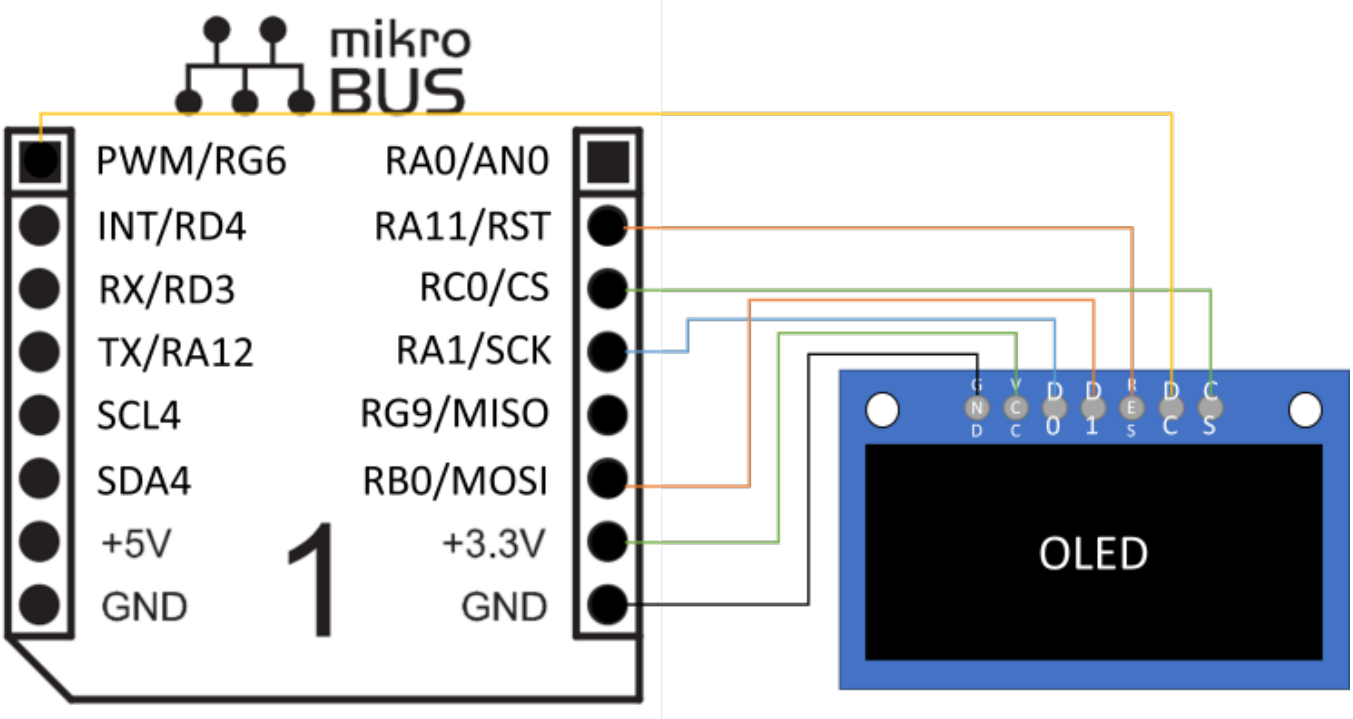


Fig. 32: Anschluss OLED

Mit dieser Umsetzung reagiert das Display auf die SPI-Signale, allerdings ist die Darstellung fehlerhaft und zeigt meistens nur Rauschen an.

Ausblick Displayprogrammierung

Mithilfe eines Arduino wurde die Funktionsfähigkeit des OLED-Displays getestet (figure 2). Das Display hat mit der Adafruit SSD1306 Library und dem darin enthaltenen Beispielcode[68] einwandfrei funktioniert. Damit konnte ein defektes Display als Fehlerquelle ausgeschlossen werden. Die Programme wurde mit dem in MPLAB implementierten Debugger auf Fehler untersucht und getestet, ob der Code die erforderlichen Stellen zur Displaysteuerung erreicht. Das hat in allen Fällen funktioniert. Anschließend wurde mit einem Logic Analyzer (figure 33, baugleich dem Saleae Logic Analyzer[69] getestet, ob über die SPI-Schnittstelle Daten gesendet werden, was zutreffend war.

Fig. 33: Logic Analyzer

Für die weitere Analyse sollte mit einem Oszilloskop nachgemessen werden, ob das Display die richtigen Befehle durch den PIC32 erhält. Dies konnte aufgrund der Covid19-Situation noch nicht an der Hochschule durchgeführt werden.

ADC

Harmony

Redmine->Handoszilloskop->Code->PIC32MK_GP->test_4_adc_dma:
Das Programm sampelt kontinuierlich Messwerte an AN0 mit 4 ADCs. Sobald Messwerte vorliegen, werden sie mittels DMA von den jeweiligen ADC-Buffern an den UART6Tx-Buffer übertragen. Darüber werden die Daten per USB als Rohdaten an einen Computer übertragen. [70)]

Bare Metal

Redmine->Handoszilloskop->Code->PIC32MK_GP->4_adc_interleaved:
Nach Vorbild des MICROCHIP AN2785 Application Note[71)] erstellt, ergänzt um die Datenübertragung per UART an PC und mikroBus Steckplatz J1 und die dafür notwendige Unterbrechung des Samplings. Die Unterbrechung ist notwendig, da die AD-Wandler ihren AD-Wandler Speicher neu füllen würden, bevor die vorherigen Daten per UART übertragen sind. Außerdem wurde der ADC Trigger auf Output Compare 1-4 umgestellt, da der PIC32M…GPE keine separate PWM Unterstützung hat. Die Rohdaten der ADC werden mit UART6 über das an J25 angeschlossene USB Kabel an den PC gesendet.[72)]

GPIO

Redmine->Handoszilloskop->Code->PIC32MK_GP->4_adc:

Separates Triggern von 4 ADCs und Datenübertragung nach Samplen der einzelnen Kanäle als Rohdaten über USB und mikrobus J8 für die Weiterverarbeitung mit Computer, beziehungsweise ESP32. Das Toggeln der Datenübertragung kann mithilfe von Buttons an mikroBus J1 geregelt werden.

5.1.4 Testen des Handoszi

Um die Performance der verschiedenen ADC-Varianten miteinander vergleichen zu können wurden zwei unterschiedliche Tests mit einem Signalgenerator durchgeführt. Verwendet wurde ein PWM & Pulse Generator vom Typ ZK-PP1K (figure 34), welcher Frequenzen bis 150 kHz erzeugen kann.

Fig. 34: PWM Generator

Er bietet die Möglichkeit, ein PWM-Signal mit variablem Duty-Cycle, oder Pulse einzeln oder mehrere mit Zeitverzögerung abzugeben[73]. Die aufgenommenen Messungen wurden mittels eines Terminal-Programms eingelesen und mit MATLAB wurden aus den Rohdaten die Spannungswerte errechnet.

Terminal Programme

Die folgende Auflistung an Programmen kann verwendet werden, um die korrekte Funktionsweise des PIC32MK zu prüfen und die von den ADC gemessenen Werten zur späteren Auswertung zu speichern. Die Programme sind alle kostenlos und frei im Internet herunterladbar.

Tera Term
Tera Term (figure 35) ist ein einfaches Terminalprogramm zur Ein- und Ausgabe, sowie zur Aufzeichnung von USB und Drahtloskommunikation.

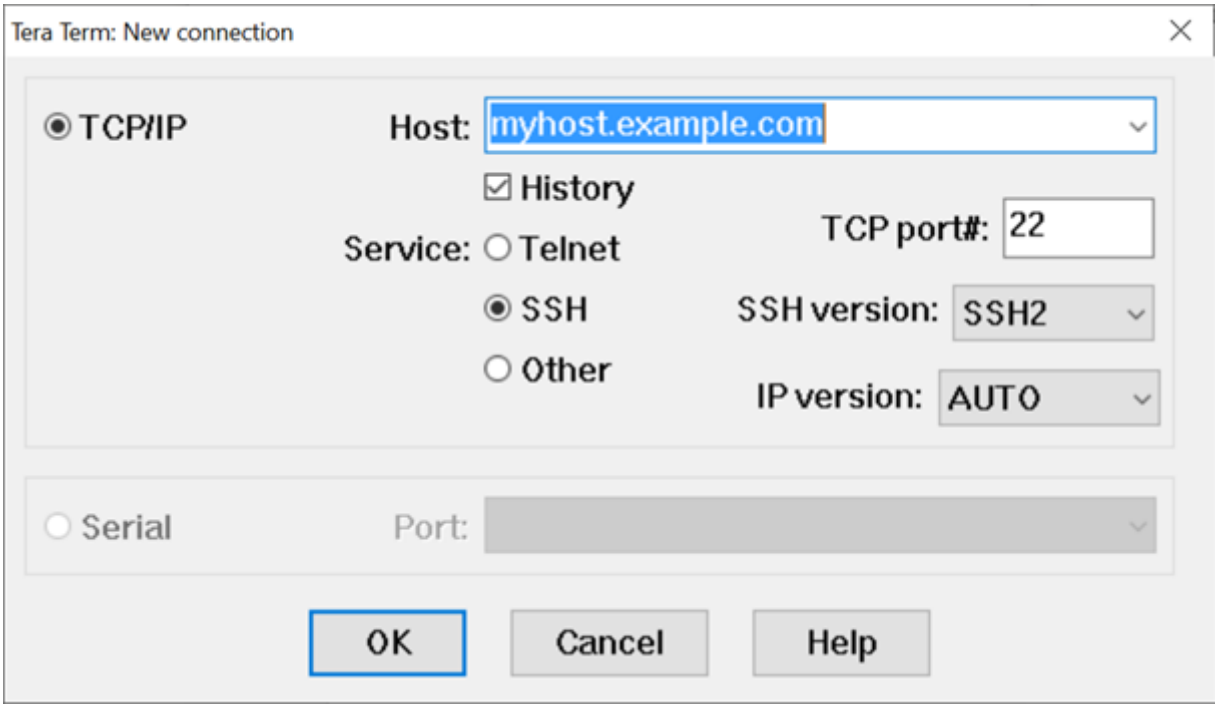


Fig. 35: Tera Term

Die USB-Kommunikation kann verwendet werden, um die ADC-Werte des PIC32 an einem externen Computer darstellen zu können und die Messwerte z.B. als .csv-Datei zu speichern. Die Darstellung in Tera Term ist beschränkt auf ASCII Zeichen, kann in den Log-Dateien aber in Binärform gespeichert werden.[74)]

RealTerm

RealTerm (figure 36) ist im Vergleich zu Tera Term weitaus umfangreicher. Es bietet die Möglichkeit, die eintreffende Kommunikation in unterschiedlichen Datenformaten anzuzeigen.

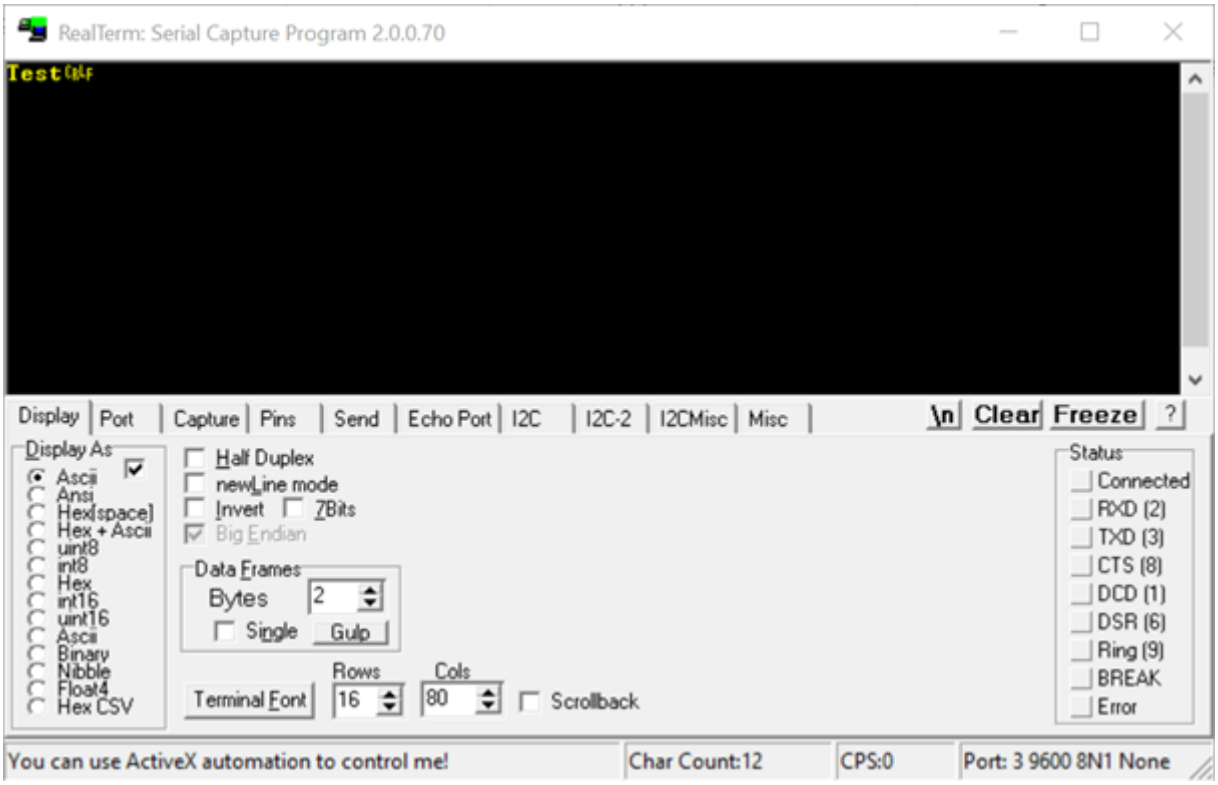


Fig. 36: RealTerm

Ein Unterschied zu Tera Term ist, dass RealTerm die Auswertung von I2C und SPI-Kommunikation unterstützt.[75)]

PuTTY
Als letztes Terminalprogramm wird PuTTY (figure 37) vorgestellt. Wie Tera Term kann es für USB- und Drahtloskommunikation verwendet werden. Eine Besonderheit ist, dass die Konfiguration des Programms für spätere Verwendung gespeichert und neu geladen werden kann. So muss zum Beispiel nicht immer die korrekte Baudrate neu eingestellt werden.

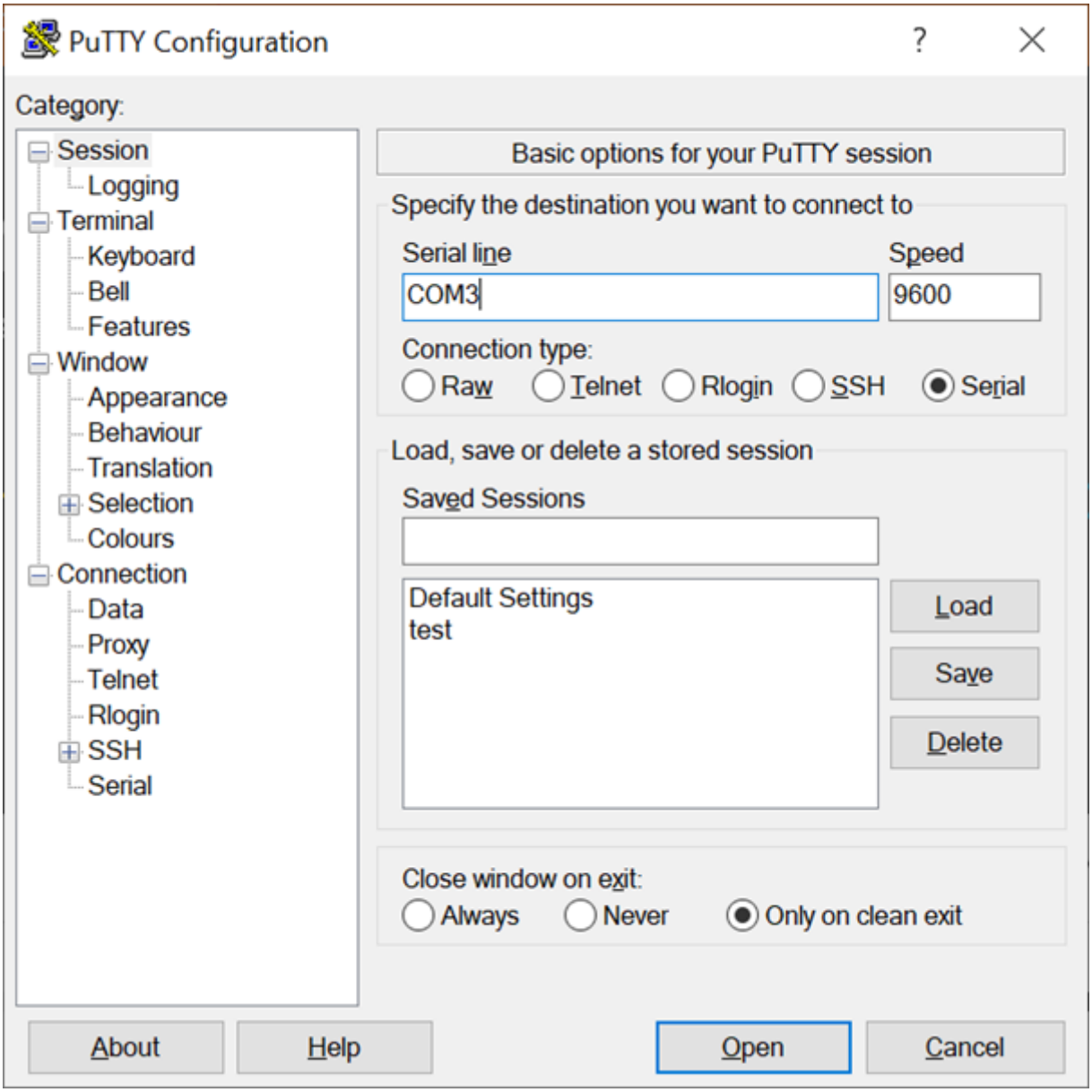


Fig. 37: PuTTY

Ein weiterer Unterschied zu den beiden anderen Programmen ist, dass die Einstellungen der Übertragung vor dem Öffnen der Verbindung stimmen müssen, da sich das Setup-Fenster nach dem Öffnen der Verbindung automatisch schließt. Dasselbe gilt für das Loggen der Daten.[76)]
Keines der Programme bietet die Möglichkeit, die empfangenen Daten in Form eines Graphen darzustellen. Die einfachste Möglichkeit für die visuelle Darstellung ist, die Log-Datei im .csv-Format zu speichern und anschließend mit einem MATLAB Skript zu plotten. Für die Aufzeichnung der durchgeführten Versuche wurde TeraTerm aufgrund der einfachen Bedienbarkeit verwendet.

Vergleich ADC-Performance

PWM Signal

Der erste Test war, ein PWM Signal mit bekannter Frequenz und bekanntem Duty-Cycle zu messen. Die Frequenz wurde so gewählt, dass bei einem erwarteten ADC-Sampling von 12 MSps und 4 ADCs mit jeweils Speicher für 256 Samples ein Zyklus gespeichert wird[77)78)]. Daraus ergibt sich eine Frequenz von 12 kHz. Verwendet wurde ein 50% Duty-Cycle. Die Messung ist in figure 38 verzeichnet.

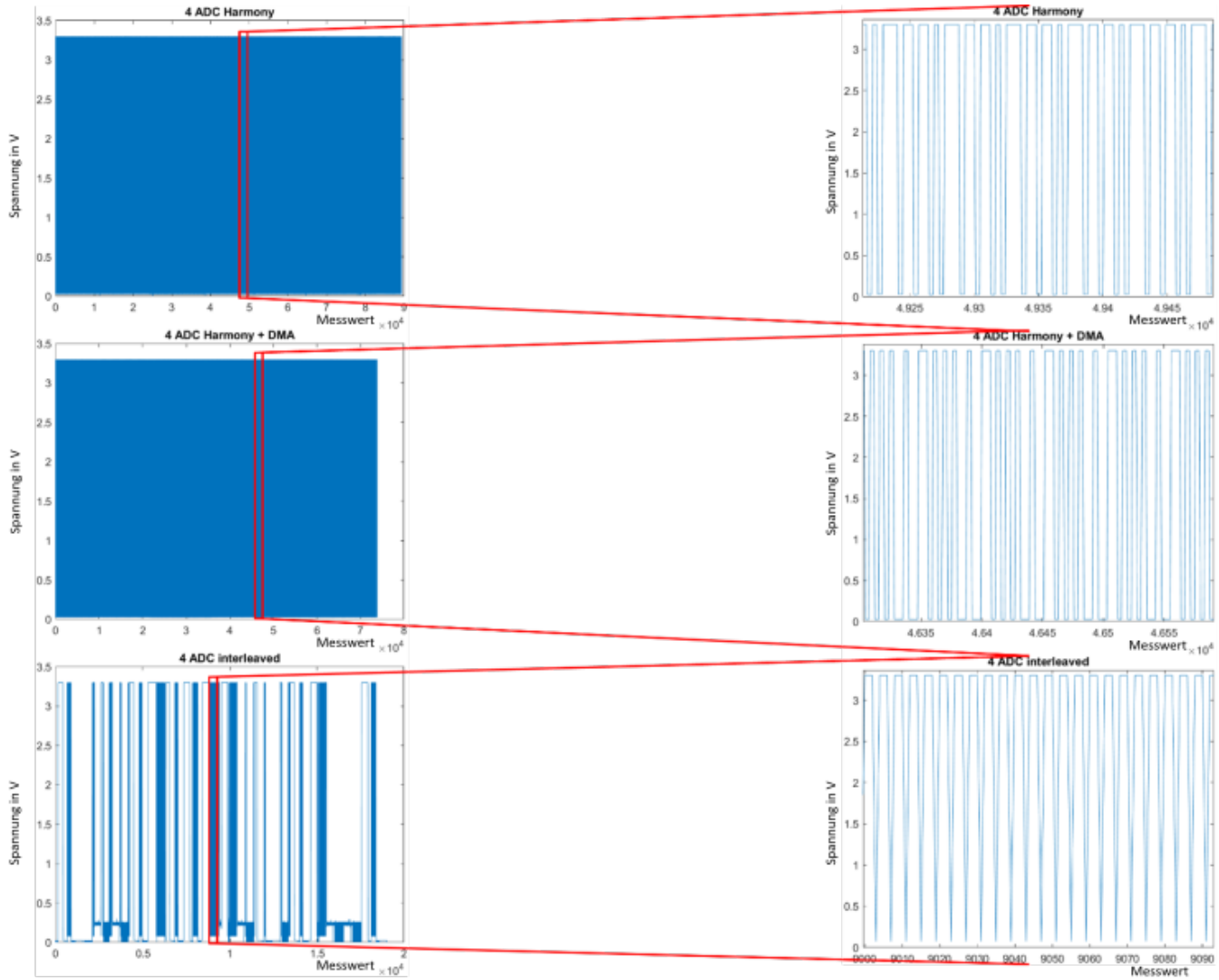


Fig. 38: ADC_Vergleich_PWM

Vergleicht man die einzelnen Messungen fällt auf, dass bei der Bare Metal Programmierung der verschachtelten ADCs das Signal nur teilweise aufgezeichnet wird. Der Grund dafür ist die notwendige Unterbrechung des Samplings für die Datenübertragung. Zoomt man in die aufgenommenen Daten zeigt sich, dass die Pulse gleichmäßig aufgezeichnet werden, bei den Versionen mit Harmony wird das Signal nicht gleichmäßig abgetastet, die Abtastfrequenz ist zu gering.

Puls

Im zweiten Versuch wurden jeweils fünf Pulse über den ZK-PP1K Funktionsgenerator erzeugt. Die Pulse waren jeweils 50 ms lang und wurden mit 50 ms Abstand nacheinander gesendet. Die Aufzeichnung ist in figure 39 dargestellt.

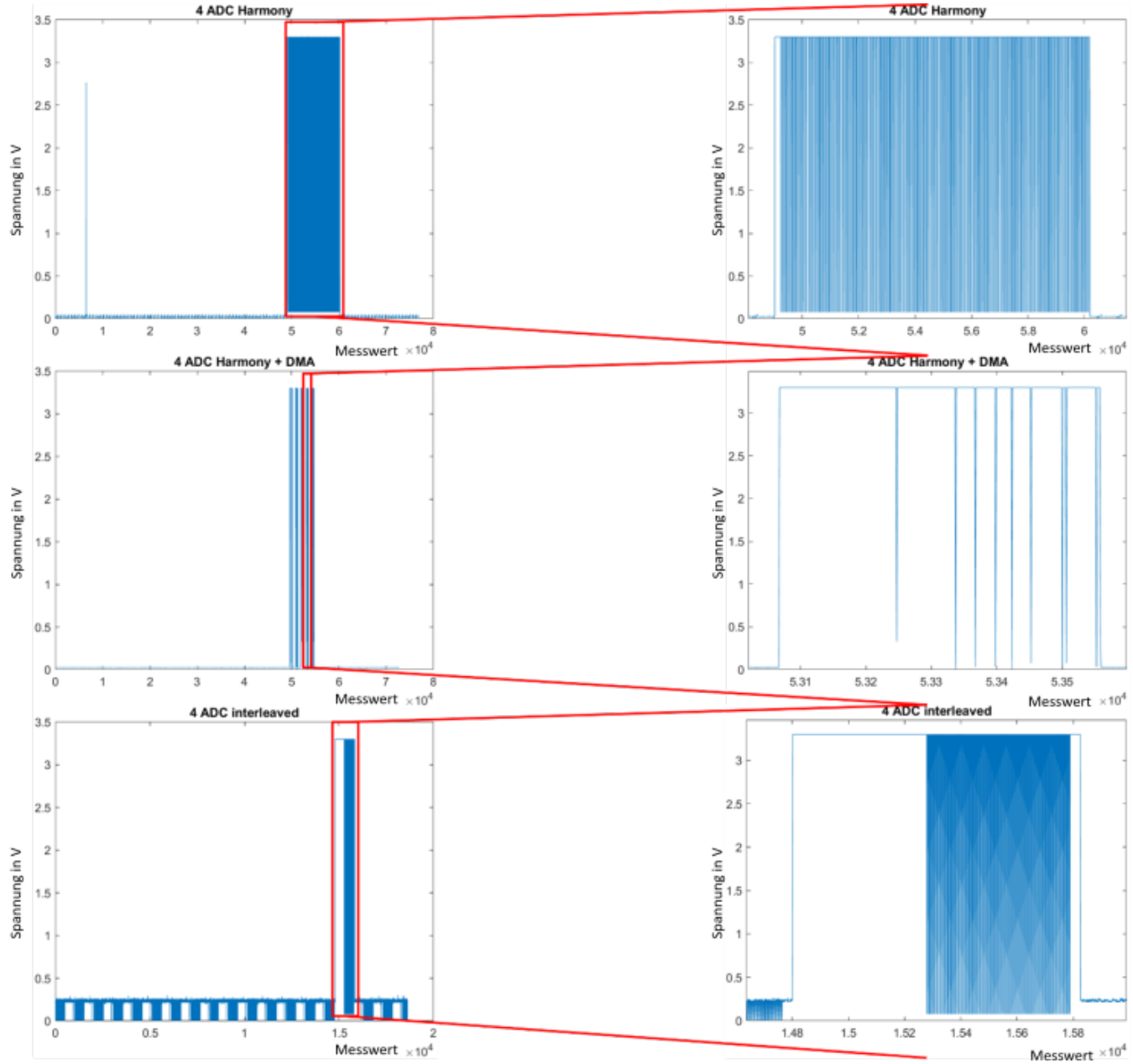


Fig. 39: ADC_Vergleich_Puls

Im Vergleich der drei Aufzeichnungen kann man sehen, das nur die Version mit Harmony und DMA die fünf Pulse richtig anzeigt. Die Bare Metal Programmierung der vier verschachtelten ADC sendet die Daten mit einer BAUD Rate von 115200 Baud/s, die beiden anderen mit 962100 Baud/s. Die Reduzierung war notwendig, da es sonst zu Übertragungsfehlern kommt. Durch die langsamere Übertragung ist es möglich, dass die einzelnen Pulse stärker geblockt werden. Die Programmierung ohne DMA zeigt ebenfalls keine einzelnen Pulse, hier ist es möglich, dass die Sample Rate zu niedrig ist und somit Teile des Signals verloren gehen.

Auswertung ADC

Die Messungen werden so dargestellt, dass auf die X-Achse der ADC-Wert in Volt, auf der Y-Achse die dazugehörigen Sample-Werte angegeben wird. Bevorzugt wäre anstatt des Sample-Werts der Zeitpunkt des Samplings angegeben worden, allerdings war das mit den vorhandenen Mitteln nicht sinnvoll. Bei der Umsetung mit vier verschachtelten ADC werden die Sample-Werte in einem Buffer zwischengespeichert und erst bei vollem Speicher ausgelesen. Dadurch lässt sich kein genauer Zeitpunkt zu den Werten zuweisen. Die beiden Umsetzungen mit Harmony könnten den genauen Zeitpunkt des Samplings mittels Timer per USB an den PC übertragen, allerdings würde durch die

momentane Umsetzung des Codes dadurch die Sample-Rate niedriger werden. Die Terminalprogramme haben ebenfalls die Möglichkeit den Zeitpunkt der eingehenden Übertragung aufzuzeichnen, allerdings ist die zeitliche Auflösung zu niedrig, bei einer Sampling-Rate im MSps-Bereich ist eine Auflösung in Mikrosekunden notwendig, die Programme können nur bis Millisekunden aufzeichnen. Die Auswertung der Versuche zeigt vor allem bei der Puls-Messung sonderbare Ergebnisse, daher sollten weitere Versuchsreihen durchgeführt werden, sowie der Programmcode angepasst werden. Die Verschachtelung der ADC zeigt die höchste Auflösung des Signals, schneidet aber im kontinuierlichen Betrieb aufgrund der notwendigen Unterbrechungen große Teile des Signals ab. Ein Einsatz im kontinuierlichen Betrieb ist daher nicht sinnvoll. Es sollte möglich sein, ein hochfrequentes Signal komplett aufzuzeichnen, so lange die Signaldauer nicht länger als 82 ms ist. (Berechnet aus: 4 ADC mit je 2*128 Sample Speicher -> Gesamt 1024 Samples / 12,5 MSps)[79)]. Im kontinuierlichen Betrieb ergibt eine Umsetzung ohne den „ADC DMA Buffer Full“-Interrupt mehr Sinn, hier können die Daten übertragen werden, sobald Daten im ADC DMA Buffer liegen. Damit reduziert sich allerdings die Sample Rate. Eine weitere Überlegung kann sein, ob für das Handoszilloskop die restlichen ADC verwendet werden, um die Sample Rate weiter zu erhöhen, oder ob sie als weiterer separater Oszilloskop-Kanal verwendet werden sollen. ADC 2, 5 und 6 können nicht so konfiguriert werden, dass sie ebenfalls die Spannung an AN0 einlesen. Dadurch wäre es bei einer gewünschten Erhöhung der Sample Rate notwendig, mehrere analoge Eingänge miteinander zu verbinden.

5.2 Kommunikation zwischen ESP und PIC32

I2C:
I2C steht für Inter-Integrated-Circuit und wurde 1982 von Philips Semiconductors (heute NXP Semiconductors) für den Datenaustausch zwischen Mikrocontrollern und IC’s entwickelt. Bei I2C handelt es sich um einen Master-Slave-Bus, dass heißt ein Datentransfer wird nur durch den Master initiiert. Der Aufbau des I2C-Datenbus ist in figure 40[80)] dargestellt.

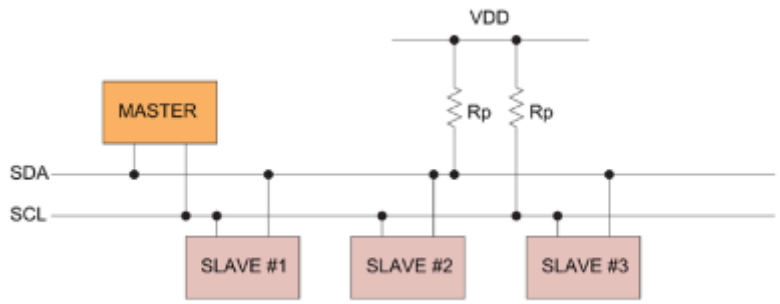


Fig. 40: Busaufbau I2C

Der I2C-Bus besteht aus vier Leitungen, davon zwei Datenleitungen (SDA und SCL). Die beiden anderen Leitungen sind für die Spannungsversorgung (Vdd und Gnd). Der Master kann die einzelnen Slaves über eine vom Hersteller vergebene Hardware-Adresse erreichen. Dabei stellen die ersten 7 übertragenen Bit die Adresse des Slave dar, das achte Bit bestimmt über die Richtung der Kommunikation. 0 steht für Schreiben, 1 für Lesen. Die Daten werden über die SDA-Leitung gesendet, die SCL-Leitung gibt den Takt vor. I2C arbeitet mit unterschiedlichen Datenraten Modi, von Standard Mode (0,1 Mbit/s) bis High Speed Mode (3,4 Mbit/s). Es ist auch möglich, dass nur unidirektionale Kommunikation verwendet wird, dann erreicht I2C im Ultra Fast-mode 5,0 Mbit/s. [81)]

SPI:
SPI (Serial Peripheral Interface) wurde 1987 bei Motorola entwickelt und ist ein synchroner serieller Datenbus. Wie I2C basiert er auf dem Master Slave Prinzip. Neben der Spannungsversorgung verwendet SPI drei Datenleitungen. Der SPI-Busaufbau zwischen Master und einem Slave ist in

figure 41[82] zu sehen.

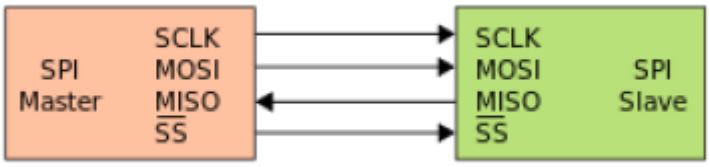


Fig. 41: Busaufbau SPI

Für die Datenleitungen existieren mehrere Namenvarianten:

- SCLK (auch SCK): wird vom Master zur Taktung verwendet
- MOSI (auch SIMO): steht für Master Out Slave In
- Es existiert auch die Bezeichnungen SDO und SDI für die Datenleitungen. Sie steht für Serial Data Out bzw. In. Im Gegensatz zu MOSI und MISO ist die Benennung oft aus Sicht der an den Bus angeschlossenen Komponenten. Die Leitungen müssen dann überkreuzt werden.
- SS: Steht für Slave Select, damit kann der Master steuern, mit welchem Slave kommuniziert werden soll. Je nach Anwendung auch Bezeichnung als CS (Chip Select), STE (Slave Transmit Enable) oder CE (Chip Enable).

Aufgrund der zwei Datenleitungen ist SPI vollduplexfähig, das heißt es kann gleichzeitig in beide Richtungen kommuniziert werden. Im Vergleich ist I2C nur im Halbduplex Betrieb verwendbar, Daten können nicht gleichzeitig in beide Richtungen übertragen werden. Mit SPI können Datenraten bis zu 20 Mbit/s erreicht werden. [83]

UART:
UART steht für Universal Asynchronous Receiver Transmitter. Es benötigt für die Kommunikation keine separate Bustaktung, die Synchronisierung zwischen Sender und Empfänger geschieht über vom Transmitter gesendete Start- und Stop-bits. So benötigt UART für die bidirektionele Kommunikation nur zwei Datenleitungen, wird nur in eine Richtung kommuniziert kann auch nur eine Leitung verwendet werden. figure 42[84] zeigt eine typische UART-Kommunikation.

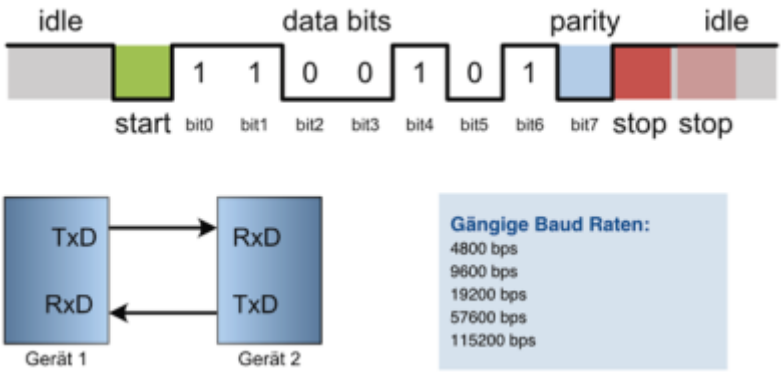


Fig. 42: Busaufbau UART

Eine UART-Übertragung besteht aus einer Nachricht mit festem Rahmen, zuerst wird ein Start-bit gesendet, anschließend fünf bis neun Daten-bits, einem Parity-bit zur Fehlererkennung und einem Stop-bit. Die Baud Rate entspricht bei UART der Bit Rate und kann Geschwindigkeiten von 50 bit/s bis 3 Mbit/s erreichen.[85]

Fazit:

Für die Kommunikation zwischen PIC32 und ESP32 ist UART die vielversprechendste Option, da wenige Datenleitungen benötigt werden und es nicht auf dem Master-Slave-Prinzip basiert. Dadurch können die beiden Bauteile unabhängig voneinander Daten und Befehle senden, ohne dass ein Bus-Master die Kommunikation initiieren muss.

5.3 ESP32

Die finale Schnittstelle zum Datenaustausch zwischen dem PIC32 und dem ESP32 wurde noch nicht festgelegt. Daher wurden hardwareseitig die Schnittstellen für UART, SPI sowie I²C vorbereitet. Eine Evaluation durch Software soll das geeignetste Protokoll ermitteln. Dieses soll dann als einzige in die nächsten Iterationen der Hardware übernommen werden.

5.3.1 Programmierumgebungen

Die Programmierung des ESP32 kann über die Eclipse IDE und das für Eclipse verfügbare IDF Plugin oder die Arduino IDE mit ESP32 Integration erfolgen.

Erforderliche Software

Für die vollständige Nutzung der Entwicklungsumgebungen sind folgende Softwarekomponenten auf dem Entwicklungsrechner erforderlich:

- Java Laufzeitumgebung: Java SE Downloads
- Python Interpreter (mindestens Version 3.5): Python Downloads
- Git (wird für die Versionsverwaltung der Projekte verwendet die mithilfe der ESP IDF erstellt werden): Git Downloads

Entwicklungsumgebungen

Für die Verwendung mit Eclipse:

- Die Entwicklungsumgebung Eclipse IDE for C/Cpp Developers: Eclipse IDE for C/Cpp Developers
- Eclipse IDF Plugin, kann über den integrierten Plugin-Manager von Eclipse heruntergeladen werden: IDF Eclipse Plugin
- Eine ausführliche Anleitung zur Installation der Entwicklungsumgebung sowie der erforderlichen Laufzeitumgebungen und Plugins ist in folgendem Artikel beschrieben: GitHub idf-eclipse-plugin

Für die Verwendung mit der Arduino IDE:

- Arduino IDE Arduino Software
- ESP Integration Installing Arduino-ESP32 support

Zur Programmierung des MikroBUS-ESP32-Prototypen kann die integrierte JTAG Schnittstelle genutzt werden. Dies erfordert einen externen Programmer. Die ESP Developmentkits besitzen einen integrierten Programmer und erfordern daher keine externe Hardware für diese Aufgabe.

Einrichten eines Projektes in Eclipse

Start Von Eclipse:

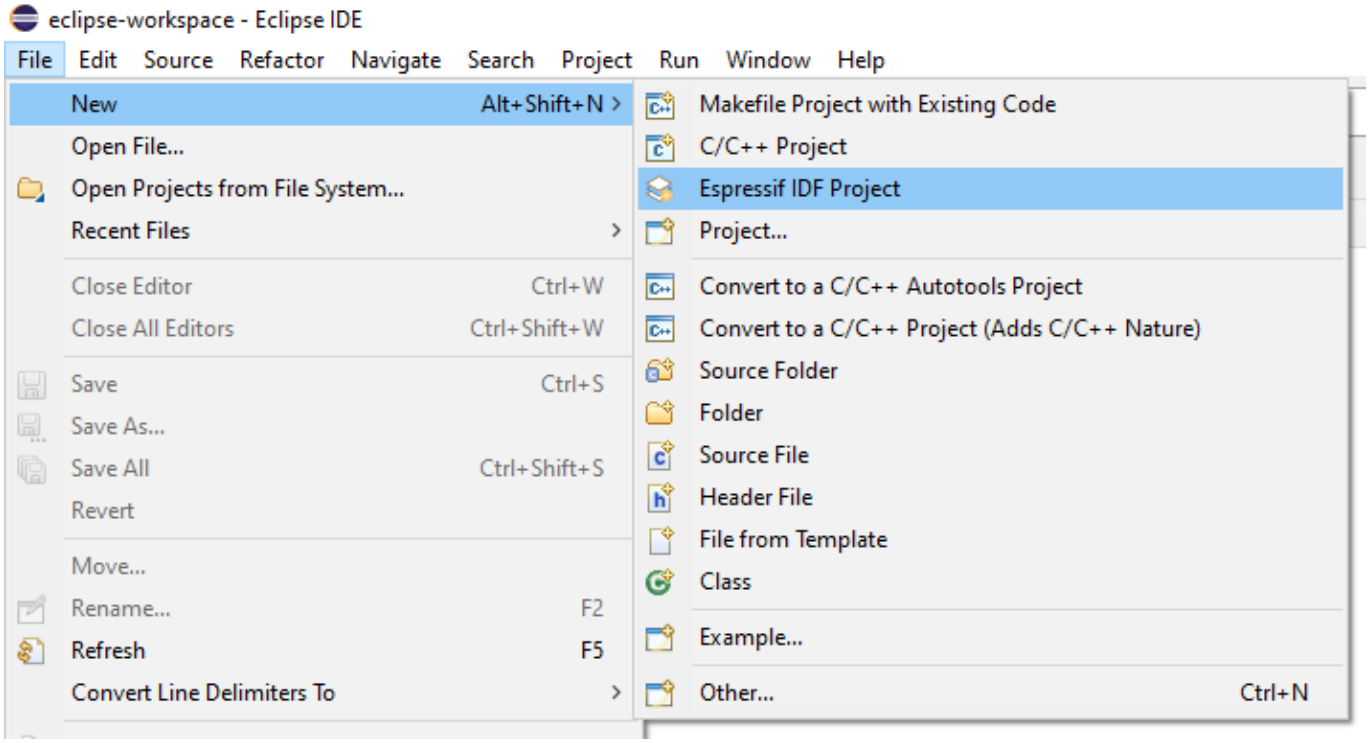


Fig. 43: Neues Projekt in Eclipse anlegen

Über die Schaltfläche „File->New->Espressif IDF Project“ kann ein neues Projekt angelegt werden. Über das Menü können vorhandene Beispielprojekte geöffnet und bearbeitet werden. Alternativ kann auch ein leeres Projekt erzeugt werden. Zunächst muss der Projektname und das Verzeichnis angelegt werden.

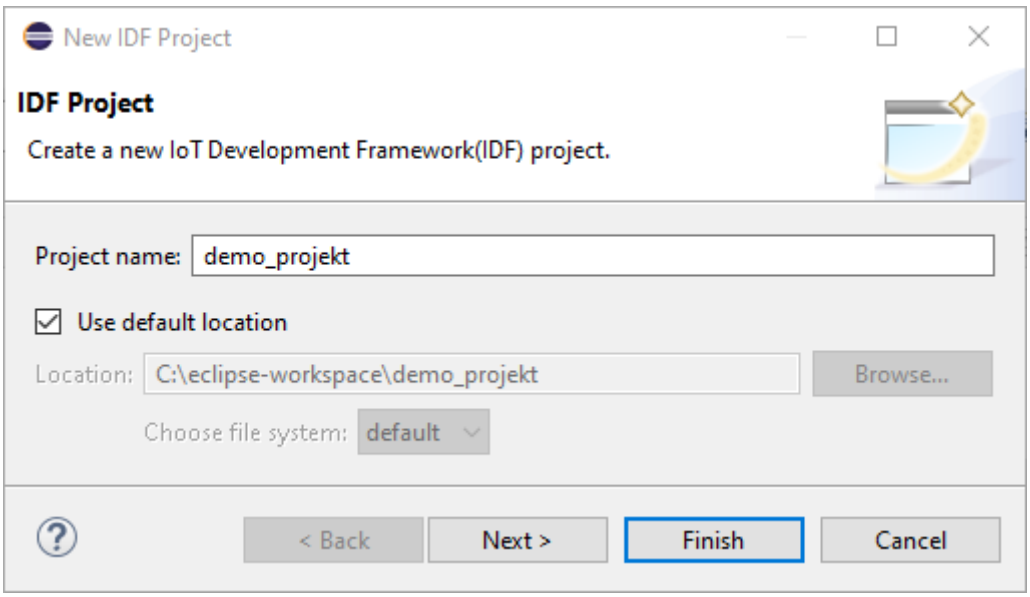


Fig. 44: Projektname festlegen

Im nächsten Schritt folgt die Auswahl des Projektes. Wird an dieser Stelle nichts ausgewählt legt Eclipse ein leeres FreeRTOS basiertes Projekt an. Alternativ kann eines der aufgelisteten Beispielprojekte ausgewählt werden, welches dann als Basis für das eigene Projekt dient.

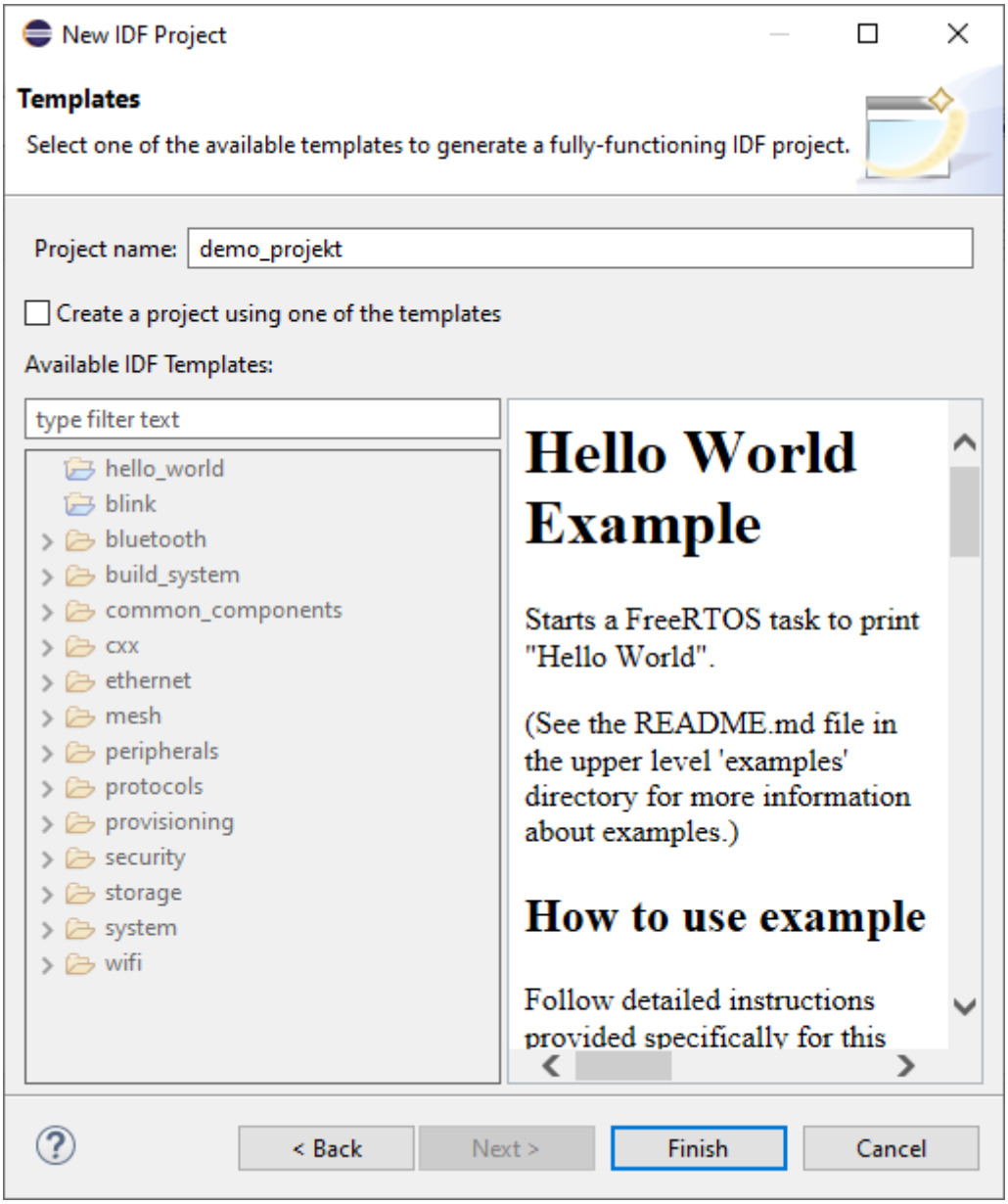


Fig. 45: Auswahlmöglichkeit leeres Projekt oder Beispielprojekt

Nach Drücken auf „Finish“ wird das Projekt erstellt. Die Ordnerstruktur besteht aus dem Build Ordner welcher die Ergebnisse des Buildprozess enthalten wird. Im Ordner main, befinden sich die Quelltextdateien und Konfigurationsdateien. Im Hauptverzeichnis befinden sich weitere Dateien wie die readme Datei die eine Beschreibung des Beispiels enthält, eine Lizenzdatei und weitere.

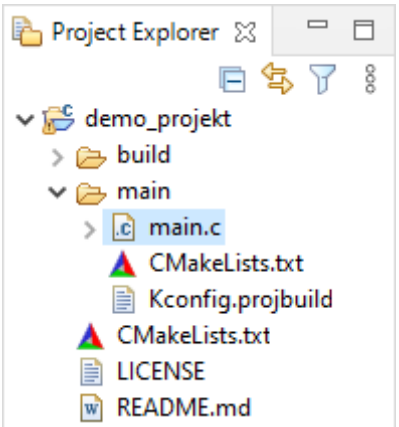


Fig. 46: Ordnerstruktur des IDF-Projekt

Wurde das Projekt erfolgreich angelegt kann mit dem Programmieren begonnen werden. Über das Auswahlmenü, dargestellt in figure 47, kann zwischen den Buildoptionen gewählt werden. Dabei kann die finale Version kompiliert und ausgeführt werden oder das Programm wird im Debugmodus gestartet.

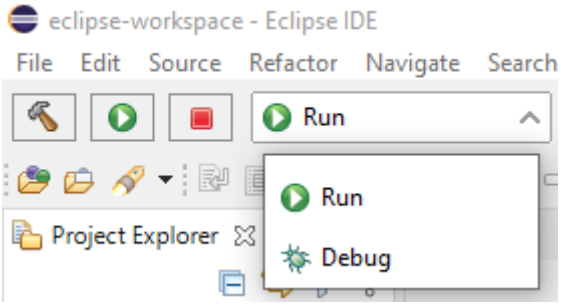


Fig. 47: Auswahl Ausführungsart

Sind mehrere Projekte im Workspace von Eclipse angelegt kann im zweiten Dropdown-Menü, zu sehen in figure 48, ausgewählt werden, welches der Projekte aktuell verwendet und ausgeführt werden soll.

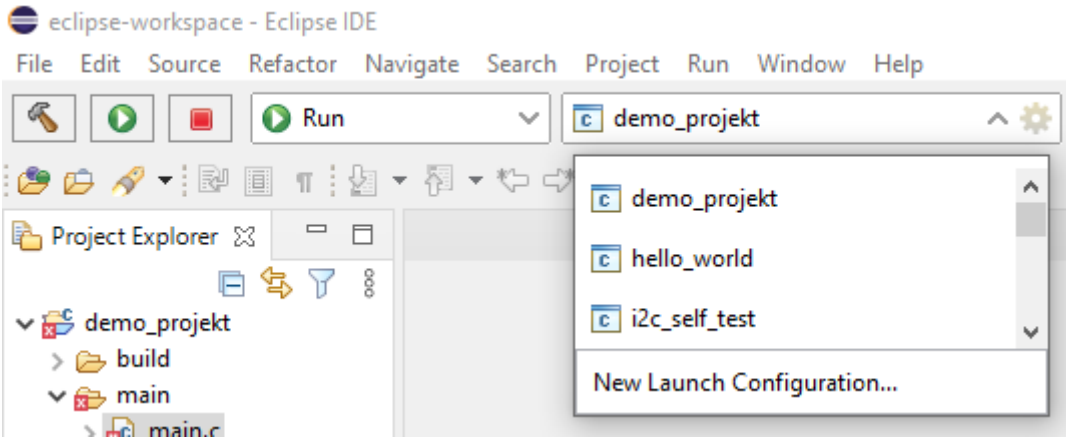


Fig. 48: Auswahl aktives Projekt

Über das dritte Dropdown-Menü, dargestellt in figure 49, kann die Zielhardware bzw. der angeschlossene Programmer eingestellt werden.

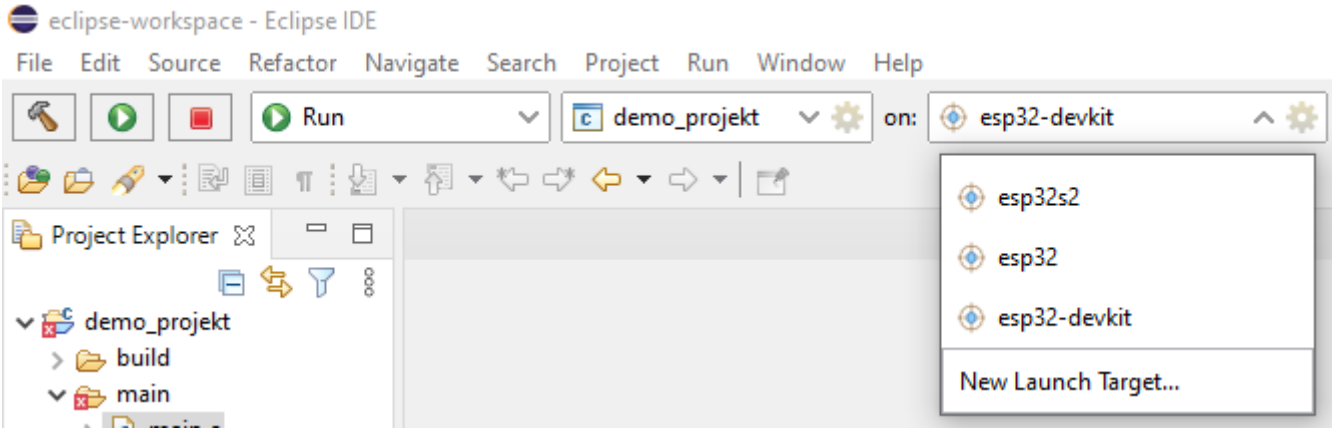


Fig. 49: Auswahl Zielhardware

Wird der Programmer mittels USB an den PC angeschlossen erscheint dieser in der Regel automatisch in der Liste. Ist dies nicht der Fall muss dieser von Hand ausgewählt werden. Hierzu muss das Menü

„New Launch Target“ geöffnet und der Programmer auswählt werden.

5.3.2 Nutzung eines Betriebssystems

Bei der Nutzung der ESP-IDF und Eclipse wird der ESP32 nicht wie der PIC32 direkt auf Hardwareebene programmiert. Die Programme werden als Tasks implementiert und laufen auf einer unterliegenden Betriebssystemschicht. Als Betriebssystem kommt hierbei FreeRTOS zum Einsatz. Die verwendeten Programmiersprachen sind C und C++. [86)]

Vorteil dieses Ansatzes ist das Vorhandensein von Treibern. Über dieses lassen sich die Hardwareschnittstellen für I²C, UART, SPI und weitere ansteuern. Die jeweilige Funktion wird mittels Softwaretreiber im jeweiligen Task geladen und kann anschließend über Funktionsaufrufe genutzt werden. Hierdurch entfällt der zusätzliche Aufwand durch die Verwendung von Registern und sorgt gleichzeitig für eine bessere Lesbarkeit des Quelltextes.

5.3.3 Testkit mit ESP32

Für die Nutzung der Kommunikationsschnittstellen und des Webservers wurden bereits erste Testprogramme für den ESP32 implementiert. Als Basis wurden die verfügbaren Beispielprogramme verwendet und auf die wesentlichen Kernfunktionen gekürzt. Die Implementierung erfolgte für die im Labor zur Verfügung stehenden ESP32 Devkits. Diese verwenden dasselbe Modell des ESP32 wie der MikroBUS ESP32-Prototyp und können daher auch für diesen verwendet werden. Gegebenenfalls ist eine Anpassung der Pinnummern zu Nutzung der Kommunikationsschnittstellen erforderlich.

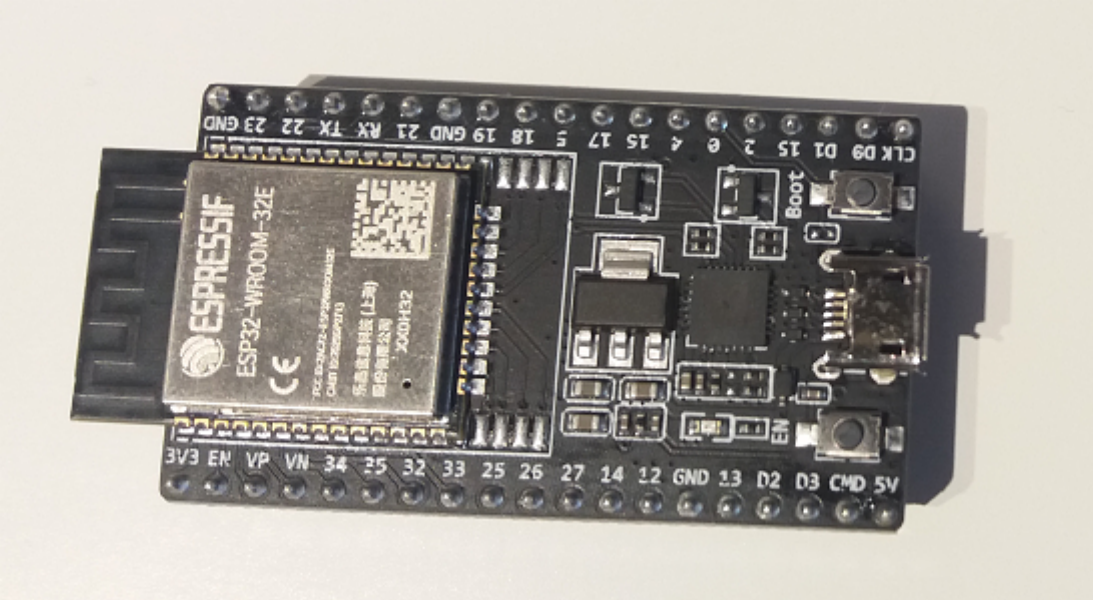


Fig. 50: Zur Programmierung der Demoprogramme verwendetes ESP32-DevKitC V4 Testkit

Die Eigenschaften des Devkit sind unter folgendem Link zu finden:

ESP32-DevKitC V4 Getting Started Guide

5.3.4 implementierte Funktionen

Für das ESP32 Devkit wurden die I²C und UART Testprogramme implementiert und getestet. Eine Beschreibung der einzelnen Beispiele sind im Folgenden aufgelistet:

I²C-Testprogramm

Das Programm dient als Demonstration der I²C Funktion des ESP32 mittels FreeRTOS Tasks.

Implementiert sind hierbei Master und Slave gleichzeitig in jeweils einem eigenständigen Task. In einer Schleife werden Daten vom Master an den Slave und umgekehrt gesendet. Da die Implementierung beider Funktionen auf dem selben ESP läuft, können anschließend die empfangenen Daten mit den Gesendeten verglichen werden um mögliche Fehler in der Übertragung festzustellen.

Das Programm ist zu finden unter:

Redmine->Handoszilloskop->Code->ESP32->i2c_self_test

UART-Testprogramm

Für die UART Funktion wurde ein einfaches Beispiel erstellt. Mit einer einfachen Echo Funktion können über UART Daten empfangen werden. Anschließend werden diese wieder über UART an den ursprünglichen Sender zurückgesendet.

Das Programm ist zu finden unter: Redmine->Handoszilloskop->Code->ESP32->uart_echo

Webserver mit Demo-Webseite

Zur Demonstration der WLAN-Fähigkeit wurde eine Webserver Demo mit Accesspoint Funktionalität erstellt. Für die Entwicklung wurden die Arduino IDE sowie vorhandene Arduino Bibliotheken genutzt. Basis für den Webserver war ein Quelltext zur Erstellung eines Accesspoints für den ESP32 [87)]. Dieser wurde angepasst und mit UART Kommunikation ergänzt.

Der Accesspoint wird über die Arduino Wifi.h Bibliothek und deren SoftAP Funktion erzeugt. Passwort und Netzwerkname (ssid) des zu hostenden Netzwerk können beliebig festgelegt werden. Über einen PC kann man sich mit dem WLAN des ESP32 verbinden und über dessen IP-Adresse auf die Webseite zugreifen. [88)]

Die Webseite ist einfach gestaltet, um mögliche Funktionen zu demonstrieren. Ein Button wird genutzt, um das Senden von Befehlen an den Webserver zu simulieren. Es können zwei Befehle gesendet werden. Ein Textfeld gibt einen Beispielmesswert an, ist das Board mittels UART mit einem anderen verbunden, können Werte an den ESP gesendet werden. Nach einer Aktualisierung der Webseite wird der empfangene Wert als Beispielmesswert angezeigt.

Das Beispiel benötigt für eine sinnvolle Nutzung als Oszilloskop jedoch Anpassungen. Der angezeigte Wert wird nur aktualisiert, wenn die Seite neu geladen wird. Für die kontinuierliche Anzeige eines Messwertes ist dies ungeeignet. Eine mögliche Lösung wäre die Integration eines JavaScript Codes der die entsprechenden Teile der Webseite, die den Messwert anzeigen automatisch aktualisiert, ohne auf eine Nutzereingabe zu warten.

Nachfolgende Abbildung zeigt die Weboberfläche, die mit dem ESP32 erzeugt wird.

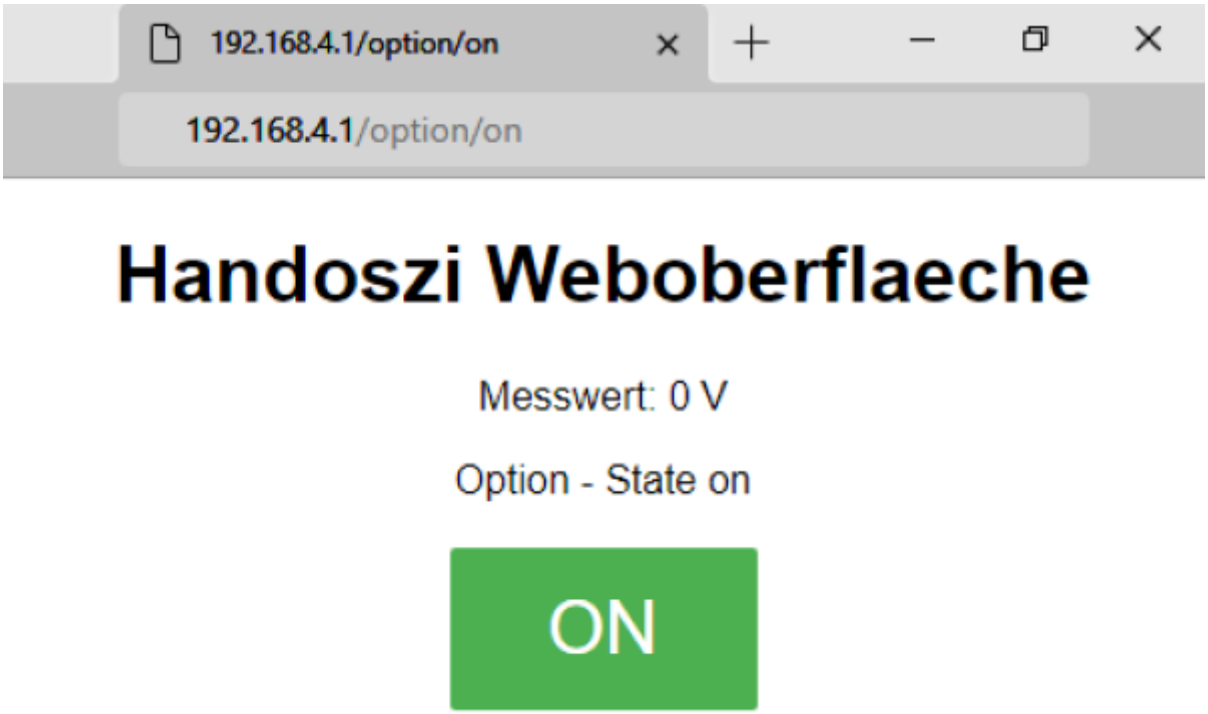


Fig. 51: Demooberfläche zur Demonstration eines Accesspoint mit dem ESP32

Das Programm ist zu finden unter:

Redmine->Handoszilloskop->Code->ESP32->esp32_access_point

5.3.5 Geplante Funktionen

Bisher nicht implementiert wurden die Kommunikation über SPI und eine vollständige Weboberfläche zur Darstellung des Oszilloskops. Weitere Implementierungen finden sich unter:

Redmine->Handoszilloskop->Code->ESP32

5.4 Anbindung an Oszilloskopsoftware für PCs

Für die Darstellung eines Oszilloskops auf einem PC stehen bereits verschiedene Programme zur Verfügung. Um Kosten zu reduzieren, soll hierbei auf Open Source Software und kostenlose Software gesetzt werden. Mögliche Programme für diesen Verwendungszweck sind Sigrok PulseView und Saleae Logic.

5.4.1 Sigrok PulseView

Sigrok PulseView ist ein Open Source Logic Analyzer und Oszilloskop Software, welche als flexible, cross-plattform und hardware unabhängige Software konzipiert wurde. PulseView unterstützt Stand 24.08.2021 241 Geräte von verschiedenen Herstellern, zum Beispiel:

- Logic Analyzer (Microchip PICkit, Saleae Logic)
- Oszilloskope (Agilent, Hameg)
- Multimeter
- Geräuschpegelmesser
- Thermometer

Eine vollständige Liste der unterstützten Hardware ist unter folgendem Link zu finden: PulseView unterstützte Hardware
Die Umsetzung geschieht mit der libsigrok API. Entwickler können für Geräte die Software für PulseView selbst einbinden. Generell sollten Geräte, welche auf dem Cypress FX2 µC basieren, oder das SCPI Protokoll verwenden, kompatibel sein. Mehr Informationen dazu im PulseView User Manual. PulseView bietet ebenfalls Unterstützung für den Obenbench Logic Sniffer, welcher einen PIC18 für die USB Kommunikation verwendet.
Vor der Umsetzung im Handoszilloskop sollte die Möglichkeit der Implementierung mit den sigrok PulseView Entwicklern abgesprochen werden. Diese sind unter folgenden Links zu errreichen:

- https://lists.sourceforge.net/lists/listinfo/sigrok-devel
- https://web.libera.chat/#sigrok

5.4.2 Saleae Logic

Das von Saleae entwickelte Programm ist nur für die von der Firma hergestellten Logic-Analyzer gedacht und ist somit nicht Open-Source wie sigrok PulseView. Eine Implementierung des Hand-Oszilloskops wird somit schwierig. Allerdings gibt es Third-Party-Hersteller, welche das Programm verwenden können. Dabei handelt es sich meist aber um Klone der Saleae-Produkte.
Weitere Informationen unter:

- https://www.saleae.com/de/downloads/
- https://support.saleae.com/logic-software
- https://support.saleae.com/

6 Ausblick

6.1 Hardware

Nach Änderung des Projektziels wurden Funktionsmodule für die einzelnen Funktionen des Oszilloskops entwickelt. Der nächste Schritt besteht darin, die einzelnen Funktionsmodule in ein gemeinsames Layout zu überführen.

Ein detaillierter Ausblick der Hardwareentwicklung findet sich in Kapitel 4.8 Weitere Entwicklung der Hardware

6.2 PIC32 Programmierung

Die Basisfunktionen zur Spannungsmessung sind umgesetzt. Dazu gehört das Ausmessen und Verarbeiten von Eingangssignalen. Zudem ist die Übertragung der Spannungswerte an Computer und ESP32 implementiert. Die Messwerte können per SPI an ein OLED-Display gesendet werden, allerdings funktioniert die Ansteuerung des Displays noch nicht fehlerfrei und die Ursache der Anzeigefehler konnte noch nicht ermittelt werden.

Im nächsten Schritt sollte mithilfe eines Oszilloskops die Datenübertragung zwischen PIC32 und OLED-Display geprüft werden und, sobald die Fehler behoben sind, eine Benutzeroberfläche zur Einstellung von Messparametern und der Darstellung momentaner Messwerte implementiert werden. Das OLED-Display im Handoszilloskop soll über 3 wire SPI gesteuert werden, das heißt, dass der Code von 4 wire SPI zu 3 wire SPI angepasst werden muss.

Die Sample-Rate der AD-Wandler sollte ebenfalls, zum Beispiel mit einem Oszilloskop, überprüft werden. Anschließend fehlt für die Oszilloskopfunktion eine Möglichkeit, die Spannungsmessung bei

Erfassung eines Eingangssignals zu starten.

Für die Kommunikation soll im fertigen Produkt nur ein Protokoll verwendet werden. Momentan ist nur die Kommunikation via UART programmiert, um die beste Schnittstelle zwischen den beiden Komponenten zu finden, muss getestet werden welche Art am schnellsten bzw. besten funktioniert.

Geplant ist, dass das Handoszilloskop auch zur Protokollanalyse (zum Beispiel I2C und SPI) verwendet werden kann, das kann mithilfe von externer Software wie sigrok Pulseview umgesetzt werden, es wäre aber auch möglich diese Funktion direkt im Oszilloskopstift zu programmieren.

Zur Anbindung an PulseView kann die Kommunikation über USB mittels SCPI-Protokoll implementiert werden, für die genaue Umsetzung sollte das Vorgehen zunächst mit den Entwicklern der Software besprochen werden.

Weitere Details zu den PIC32-Arbeitsschritten sind zu finden unter:

- Auswertung ADC
- Ausblick Displayprogrammierung
- Kommunikation mit ESP32
- Externe Oszilloskop-Software

6.3 ESP32 Programmierung

Zur Demonstration der Funktionen des ESP32 wurden verschiedene Demoprogramme entwickelt. Diese verwenden die Funktion um den I²C-Bus, die UART-Schnittstelle und die Verwendung des ESP32 als Webserver.

Verbesserungspotentiale zeigen sich vor allem beim Funktionsumfang des Webserver. Zudem ist noch kein Demoprogramm zur Nutzung der SPI-Schnittstelle vollständig erstellt.

Da die mikroBUS-Platine des ESP32 keinen integrierten Programmer enthält, ist die Anschaffung eines ESP-Prog wie in Kapitel 4.4.3 beschrieben sinnvoll um dieses anstelle der Testplatine aus Kapitel 5.3.3 zu nutzen.

Ein detaillierter Überblick findet sich in Kapitel 5.3 ESP32.

Quellenverzeichnis

[1)]
Nyquist-Shannon-Abtasttheorem
[2) 26)]
PIC32MK Product Family
[3) 5)]
Bluetooth - Wikipedia, 18.08.2021
[4)]
Bluetooth Low Energy - Wikipedia, 18.08.2021
[6)]
WLAN - Wikipedia, 18.08.2021
[7)]
Wi-Fi - Wikipedia, 18.08.2021
[8) 9) 10) 16)]
Datenblatt ESP32 WROOM 32E & ESP32 WROOM 32UE, 18.08.2021
[11) 15)]

TTGO-Micro32
12)

TTGO-Micro32 Aliexpress
13)

Mouser - ESP32-WROOM-32E(M113EH3200PS3Q0), 2021
14)

Größenvergleich TTGO-Micro32 und ESP32-WROOM
17) , 18) , 19) , 21)

Gabotronics XMEGA Xprotolab
20) , 25)

TLV9101 Datenblatt
22) , 23)

Beckhoff Information System - Typisierung SingleEnded / Differentiell, 18.08.2021
24)

TLV9102 Datenblatt
27) , 28) , 29) , 30) , 31) , 32) , 33) , 34)

Buydisplay ER-OLED0.96-1.3W
35)

SAM L21 Datenblatt
36) , 37) , 38) , 39) , 40) , 41)

PIC32MK General Purpose (GP) Development Board User’s Guide
42)

Click boards™ - MikroElektronika
43)

mikroBUS™ - MikroElektronika
44) , 45) , 46) , 47)

mikroBUS™ standard specifications
48)

ESP-Prog - PlatformIO
49)

ESP-PROG - Digi-Key
50)

Introduction to the ESP-Prog Board, 10.09.2021
51)

Adafruit STM32F405 Feather Express
52)

Datenblatt MCP73831/2 Lademanagement-Controller
53)

MPLAB X User Guide
54)

MPLAB Creating a Harmony Project
55)

MPLAB view and set configuration bits
56)

MPLAB Harmony Tutorial
57)

PICE32 Family Datasheet
58)

PIC32 Interrupts
59)

MPLAB X MCC
60)

MCC Release Notes
61)

MPLAB Data Visualizer

62)

MPLAB Harmony MHC

63)

OLED-Display

64)

Adafruit SSD1306 Library

65)

PIC MCU with SSD1306 SPI OLED

66)

Github u8g2 Library

67)

Github PIC32MZ examples

68)

Arduino Adafruit example

69)

Saleae Logic Analyzer Amazon

70)

PIC32 DMA Controller

71)

MicroChip AN2786

72)

PIC32 Output Compare

73)

ZK-PP1K Manual

74)

TeraTerm Manual

75)

RealTerm Manual

76)

PuTTY Documentaion

77)

PIC32 Worlds fastest 12 bit ADC

78) , 79)

PIC32 12 bit SAR ADC

80)

I2C_Busaufbau.png

81)

Wikipedia I2C

82)

SPI_Master-Slave.png

83)

Wikipedia SPI

84)

UART-Bus.png

85)

Wikipedia UART

86)

FreeRTOS - ESP32 - ESP-IDF Programming Guide

87) , 88)

RandomNerdTutorials.com - How to Set an ESP32 Access Point (AP) for Web Server

From:
https://wiki.mexle.org/ - **MEXLE Wiki**

Permanent link:
https://wiki.mexle.org/projekt_mexle_handoszi/start?rev=1631871119

Last update: **2021/09/17 11:31**