Kapitel 1

Schaltungsentwurf/ Design des Systems

Noch in der Vorbereitungsphase dieser Arbeit wird die Schaltung aus Experiment 5 auf dem ASLK-PRO-Board aufgebaut. Dabei fällt auf, dass am SF-Pin des Multiplizierers statt der im Datenblatt angegebenen 10 V nur etwa 8,78 V anliegen. Der Grund dafür ist die höhe der Versorgungsspannung, die auf dem ASLK-PRO-Board nur bei ±10 V, anstatt der im Datenblatt vorgeschlagenden ±15 V liegt. Dieser Unterschied sollte die Funktion des Multiplizierers zwar nicht beeinträchtigen, für eine bessere Verständlichkeit der Schaltung wären die ±15 V aber hilfreich. Der verwendete opv TL082B ist laut Datenblatt bis zu ±20 V verwendbar, sodass das erste pcb mit ±15 V Versorgungsspannung geplant wird.

1.1 Design des Schaltplans

In folgenden wird die Entwicklung des Schaltplans genauer beschrieben. Dabei kann der Schaltplan aufgrund der Größe und damit zusammenhängender leserlichkeit leider nicht in einer Abbildung dargestellt werden. Stattdessen werden die einzelnen Module und Teilschaltungen in separaten Abbildungen gezeigt und erläutert. Für die Gesamtübersicht wird auf den im **Anhang/Repo/USB-Stick** hinterlegten Schaltplan verwiesen.

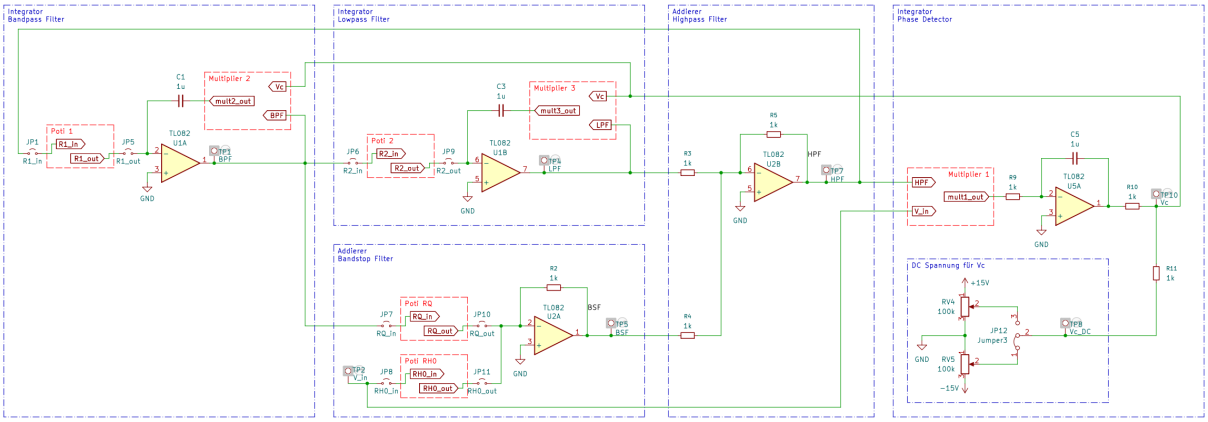

Abbildung 1.1: Darstellung des VCF ohne Perepherie

Die anderen im vcf verwendeten Bauteile werden größtenteils vom Aufbau des letzten Semesters bzw. dem ALSK-PRO Manual [1] übernommen. Bei der Wahl des opv soll in

Version 1 des pcb der gleiche OpAmp verwendet werden wie schon zuvor. Dabei wird darauf geachtet, den für Filteranwendungen etwas besseren TL082B zu verwenden. Dieser baut auf der gleichen Architektur auf, bietet aber leicht verbesserte Werte im Bereich der Input Offset Voltage und Input Offset Drift. Dadurch werden besonders im Bereich der Integration Fehler minimiert, da diese Version etwas präziser arbeitet. Das gbw und die Slew-Rate werden durch die Wahl der Version nicht beeinflusst.

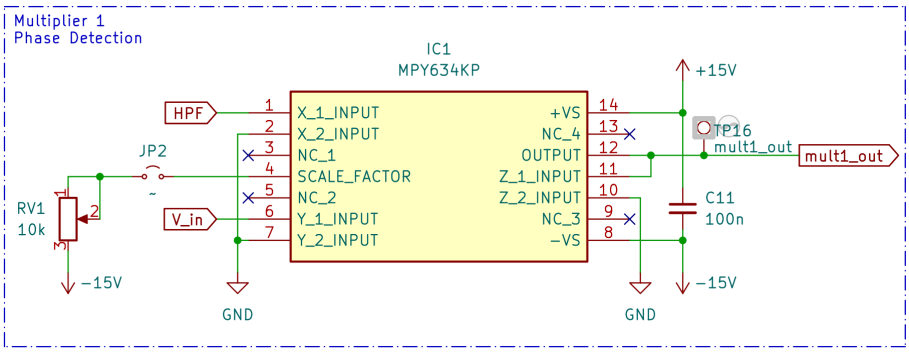

Abbildung 1.2: Verschaltung des Analog-Multiplizierers MPY634

neue Abbildung mit funktionierendem Poti

In Abbildung 1.2 ist die Verschaltung des ersten von drei analogen Multiplizierern zu sehen. Die weiteren beiden Bausteine werden abseits der Ein- und Ausgänge identisch verschaltet, sodass die resultierende Übertragungsfunktion der Multiplizierer der Gleichung **??** aus Kapitel **??** entspricht. Dabei kann das Potentiometer am rechten Rand der Abbildung zur Einstellung der Referenzspannung V_r innerhalb des Multiplizierers verwendet werden. Durch Entfernen des Jumpers kann diese Funktion aber auch deaktiviert werden, wodurch die $-10\,\mathrm{V}$ anliegen sollten. Der $100\,\mathrm{nF}$-Kondensator zwischen den Versorgungsspannungen dient der Glättung von Störspannungen.

Für die Digitalpotentiometer fällt die Wahl auf das von Herrn Ziemann vorgeschlagene MCP4261. Bei der weiteren Auswahl wird Wert auf eine möglichst große Schrittzahl und einen angemessenen Maximalwert gelegt. Da zur Einstellung des Güte- und Verstärkungsfaktors sowie der Mittenfrequenz des Filters insgesamt vier Potentiometer gebraucht werden, wird ein Modul gewählt, bei dem sich zwei Potis in einem Gehäuse befinden. Der Baustein ist zudem als Potentiometer und Rheostat erhältlich, wobei der Rheostat die gewollten strombegrenzenden Eigenschaften besitzt, **während das Potentiometer eine Spannung herausgibt.** Trotzdem wird in diesem Fall die Potentiometerversion verwendet, da diese leichter in PDIP-Gehäusen erhältlich ist und sich durch das Offenlassen eines Pins (nicht der Abgriff) als verstellbarer Widerstand einsetzen lässt.

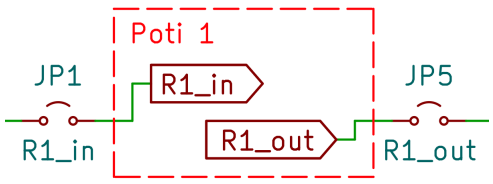

Abbildung 1.3: Darstellung der Ein- und Ausgänge der Digitalpotentiometer

Die Ein- und Ausgänge der Potentiometer werden, wie in Abbildung 1.3 zu sehen, jeweils mit Jumpern versehen, damit der eingestellte Widerstandswert schnell und ohne Beeinflussung durch die restliche Schaltung gemessen werden kann. Die Multiplizierer werden an den vorgesehenen Stellen in den Biquad eingesetzt, wobei der Scale-Faktor-Pin so verschaltet wird, dass dessen Potential über einen Spindeltrimmer einstellbar ist und sich die Proportionalitätskonstante des Multiplizierers anpassen lässt.

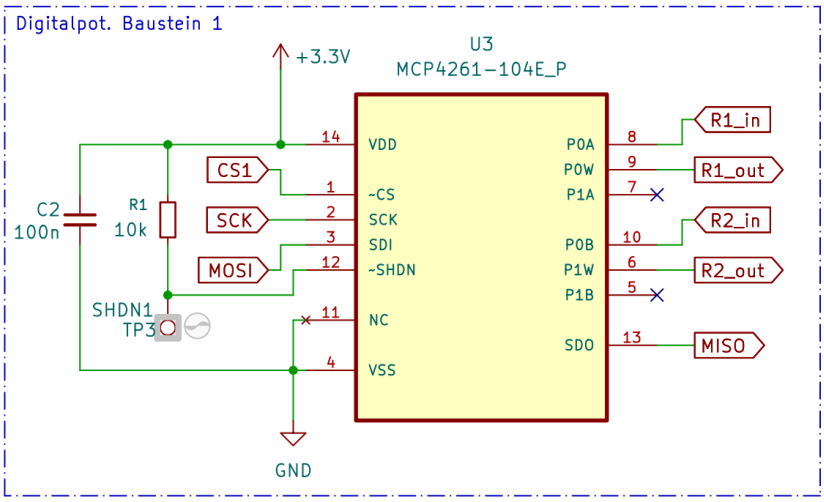

Abbildung 1.4: Verschaltung der Digitalpotentiometer MCP4261

Abbildung 1.4 zeigt die Verschaltung des ersten Digitalpotentiometers. Das zweite Modul ist dazu identisch und die Verschaltung erfolgt wie im Datenblatt angegeben. Dazu gehören die Anschlüsse zum spi-Bus sowie des Glättungskondensator zwischen V_{SS} und V_{DD} **und dem 10 kΩ Widerstand um den SHDN-Pin auf ein definiertes High zu ziehen**. Das Schaltsymbol für den MCP4261 wird dabei nicht selbst erstellt, sondern von Mouser heruntergeladen und in eine eigene Bibliothek eingefügt. Problematisch ist dabei, dass die Zuteilung der Pins für die internen Potentiometer nicht ideal im Symbol wiedergegeben wird. Pin PA1 und Pin PB0 sind im Schaltsymbol vertauscht, was leider erst nach der Bestellung des ersten pcb, jedoch vor der Testung der Schaltung, bemerkt wurde. Durch die Setzung einer Drahtbrücke konnte dieser Fehler jedoch schnell beseitigt werden. In der 2. Version der Platine wird dies überarbeitet.

Zum Zeitpunkt der Schaltplanerstellung ist die genaue Funktion der Hilfsspannung V_H noch nicht genau bekannt. Damit später trotzdem der optimale Wert einstellbar ist, soll die in Abbildung **??** zusehende Schaltung einen Spannungswert zwischen ±15 V ausgeben können.

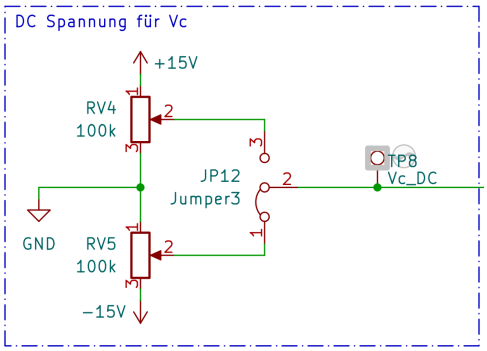

Abbildung 1.5: Implementierung der einstellbaren Hilfsspannung V_H

Um die Signale der verschiedenen Filtertypen besser mit geeeigneten Messinstrumenten wie Oszilloskop oder Spektrumanalysator aufnehmen zu können, werden BNC-Anschlüsse auf der Platiene angebracht, da diese Messgeräte meist BNC-Eingänge besitzen.

Auf dem pcb übernimmt die mcu die Ansteuerung der Digitalpltentiometer. Später soll diese zudem ein ui zur Bedienung der Filterparameter bereitstellen. Hierfür wird ein Raspberry Pico 2 W verwendet, dessen Mikrokontroller RP2350 in Europa entworfen wurde. Er ist eine Weiterentwicklung des RP2040 **und sollte für diese Aufgabe stark genug sein.** Zusätzlich befindet sich auf dem Pico 2 Modul der WLAN Baustein CYW43439 von Infinion, der die drahtlose Kommunikation (WLAN und Bluetooth) zwischen Eingabegerät und Filter ermöglicht.

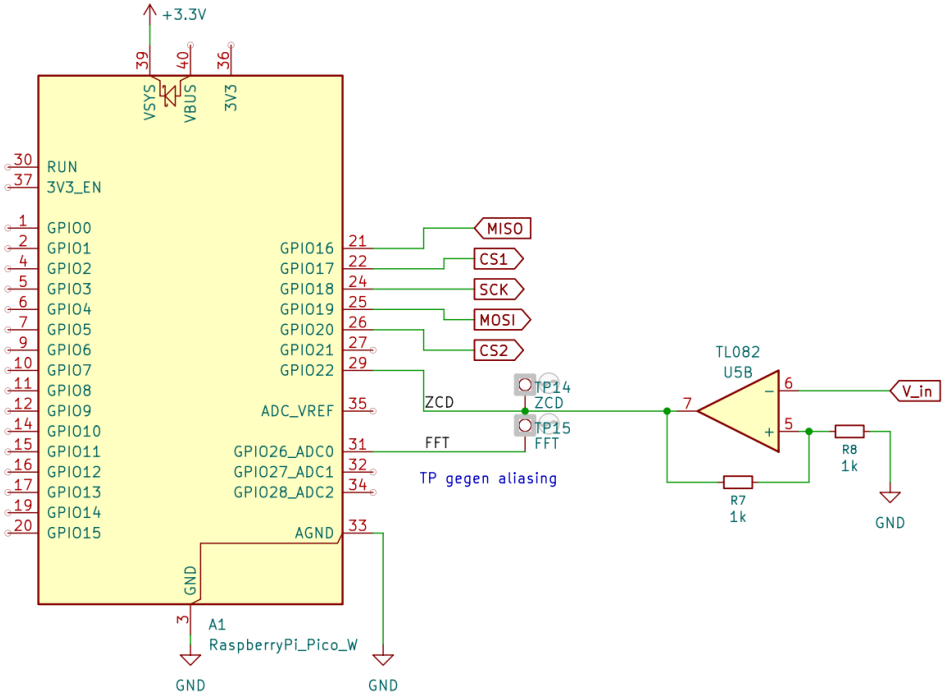

Abbildung 1.6: Verschaltung des Raspberry Pico 2 W

Ebenfalls in Abbildung 1.6 zu sehen ist ein weiterer opv, der zur Frequenzbestimmung des Eingangssignals genutzt werden soll. Der opv wird als invertierender Schmitt-Trigger beschaltet, um das eingehende Sinussignal in ein Rechtecksignal zu überführen. Dieses

Rechtecksignal soll dann über die Nulldurchgangsmethode in der mcu die Eingangsfrequenz bestimmen. Ebenfalls wird der Zugriff für einen weiteren gpio-Pin über den Testpunkt ermöglicht. Falls noch eine fortschrittlichere Frequenzbestimmung eingebaut werden soll, kann dieser Testpunkt einfach erreicht werden.

Sowohl die Digitalpotentiometer als auch die mcu benötigen Versorgungsspannungen im Bereich von etwa 1,8 V bis 5,5 V. Da auch Busse und allgemein Pins der mcu mit 3,3 V betrieben werden, wird eine Versorgungsspannung 3,3 V integriert. Um dieses Potential zu erreichen ohne sehr hohe Verluste zu generieren, soll ein Buck-Converter die 15 V auf ein Niveau von 3,3 V absenken. Die zugehörigen externen Bauteile des Buck-Converters werden wie im Datenblatt angegeben anhand des Maximalstroms, der maximalen Eingangsspannung und der Ausgangsspannung dimensioniert. Zusehen ist diese Schaltung in Abbildung 1.7.

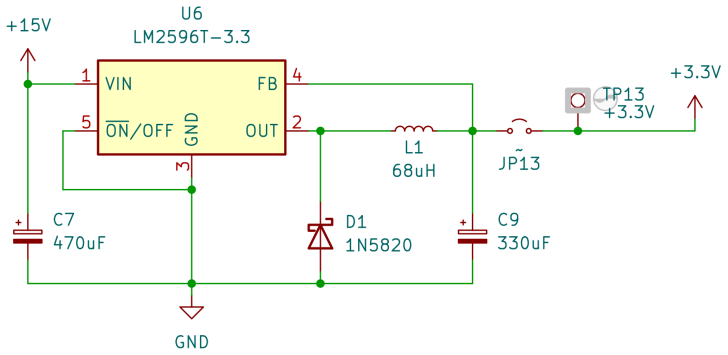

Abbildung 1.7: Aufbau des Buck-Converters

Zuletzt wird überprüft, ob alle relevanten Signalpfade mit Testpunkten versehen sind, um während der Messungen möglichst viele interne Signale des Self-Tuned Filters aufnehmen zu können. Das soll später dabei Helfen potentielle Fehler schneller zu identifizieren. Zusätzlich werden Mounting-Holes gesetzt, damit das pcb sicher fixiert werden kann und der Zugriff auf die Edge-Mount-BNC-Steckverbinder problemlos funktioniert. Insgesammt wird der Schaltplan so gestaltet, dass er möglichst übersichtlich und gut nachvollziehbar bleibt.

1.2 Design der Leiterplatte (PCB)

Beim Platinendesign muss zuerst entchieden werden, wie viele Lagen das pcb haben soll. An sich kann jeder beliebige Schaltplan auf einem zweilagigen pcb umgesetzt werden. Dies geht allerdings auf Kosten der Platinengröße und kann Störeinflusse begünstigen. Um beide Aspekte zu minimieren, wird ein vierlagiges Layout gewählt, da dies das Routing erheblich einfacher macht und die Mehrkosten überschaubar sind.

Ein wesendlicher Vorteil von mehrlagigen pcb besteht darin, dass Bauteile deutlich dichter an einander plaziert werden können, ohne dass sich dazwischenliegende Leiterbahnen gegenseitig behindern. Dieser Vorteil wirkt sich noch stärker auf pcb mit SMD-Komponenten aus, da der Footprint nur auf der ersten Kupferlage zu erkennen ist und die anderen Lagen nicht aktiv beeinflusst. Trotzdem wurde in der ersten Iteration des pcb absichtlich

auf SMD-Komponenten verzichtet, da THT-Elemente im Umgang einfacher sind und bei Anpassungen leichter auszutauschen sind.

Platzbedingte Vorteile des vielagigen pcb bestehen vor allen Dingen darin, dass die verwendeten Bauteile näher an einander plaziert werden können, ohne dass sich die dazwischenligenden Leiterbahnen behindern. Dieser Vorteil wirkt sich noch stärker auf pcb mit SMD-Komponenten aus, jedoch wurde auf diese in der ersten Iteration absichtlich verzichtet, da der Umgang mit THT-Bauteilen einfacher ist und diese bei eventullen anpassungen leichter auszutauschen sind.

Die allgemeinen Aufgaben der vier Lagen werden wie folgt zugeordnet:

1. Layer 1: Signalrouting

   Die oberste Lage des pcb dient hauptsächlich dem Signalrouting. Hierbei sollen so weit es geht alle an der Filterung beteiligten Signalpfade über diese Ebene geleitet werden. Auch wenn der in dieser Abeit betrachtete Frequenzbereich noch recht unanfällig für Störungen ist, wird darauf geachtet, diese Störeinflüsse so gering wie möglich zu halten.

2. Layer 2: Groundlayer

   Direkt darunter befindet sich eine durchgehende Massefläche. Diese Schicht bleibt ununterbrochen, sodass das Ground-Potential über die THT-Pins beziehungsweise Vias an jeder Stelle des pcb einfach erreichbar ist. Zudem kann diese Fläche potentiell störenden Bussignale abschirmen.

3. Layer 3 Powerlayer

   Diese Schicht dient ausschließlich der Spannungsversorgung der Bauteile auf dem pcb. Die $-15\,\mathrm{V}$ sowie die $3{,}3\,\mathrm{V}$ werden dabei ganz normal geroutet, die $15\,\mathrm{V}$ verlaufen hingegen großflächig über die Ebene, da dieses Potential sowohl die opv betreibt, als auch die $3{,}3\,\mathrm{V}$ von diesem Signal aus abgehen. Dies ist die einzige Ebene in der die Zone nicht auf Ground gelegt wird.

4. Layer 4 Signallayer

   Die unterste Schicht dient als Ausweichmöglichkeit für sich kreuzende Signale. An sonsten soll sie hauptsächlich für das Routing von Bussignalen verwendet werden. In Version 1 wurde darauf noch nicht so genau geachtet, in der (**Endversion!!**) hingegen schon.

Bei der Plazierung der Komponenten wird darauf geachtet, das Komunikationsmodul der mcu ganz an den Rand des pcb zusetzen, um die Abschirmung der Antenne durch die Kupferflächen der Platine zu vermeiden. Bei der Platizerung des Buck-Converter und dazgehörigen weiteren Komponenten wird darauf geachtet, dass diese wie im Datenblatt beschrieben auf dem pcb plaziert werden. Aufgrund von Platzmangel hat sich das Layout dennoch etwas verändert.

Da in der linken oberen Ecke des pcb noch etwas Platz ist, wird für einen einfachen Zugang auf die Dokumentation des Projekts ein QR-Code zum Git-Reposatorium der Arbeit hinzugefügt.

1.3 Design des Skripts für die Steuerung

Bei der Entwicklung der Steuersoftware für die mcu muss die Vorgehensweise aufgrund geringer Vorerfahrung mit dem RP2350 und der Programmierung in MicroPython gut geplant werden.

Nach der Installation der aktuellen Micropython-Firmware auf der verwendeten mcu soll zuerst das spi-Interface initialisiert werden, um die Digitalpotentiometer ansteuern zu können. Dafür wird im Code der spi-Bus so konfiguriert, dass Miso, Mosi und SCK den geplanten gpio-Pins zugeteilt werden. Anschließend werden die CS-Pins als normale gpio-Pins definiert. So können nun Funktionen geschrieben werden, welche die MCP4261-Befehlswörter senden und die Wiperpositionen der Potentiometer setzen.

Das Digitalpotentiometer verwendet bei der Komunikation über SPI standardmäßig 16-Bit-Datenwörter. Dabei bilden die ersten 8 Bits das "Command Byte", das die Registeradresse und den Command (den jeweiligen Wiper und seine Funktion) beinhaltet. Das darauffolgende "Data Byte"liefert den neuen 8-Bit-Wiperwert.

Neben der Programmierung der Widerstandswerte für das ram-Register des Chips werden zudem nichtflüchtige Start- und Resetwerte im eeprom des Digitalpotentiometer hinterlegt. Diese dienen dazu, dass die Potentiometer nach einem Power-Up oder Reset des Systems auf einen definierten Anfangswert gesetzt werden, anstatt sich auf die Mittelstellung einzustellen. Bei eventuellen späteren Komplikationen bei der Messung des Systems mit dem mcu auf der Platine können so die gewollten Werte sichergestellt werden. Da die Anzahl der Schreibzyklen des eeprom begrenzt ist (ca. 1 Mio.), wird dieser nicht für das laufende Einstellen der Werte, sondern nur für die initiale Konfiguration verwendet.

Zum Debugging dient zudem eine Funktion, welche die aktuellen Registerwerte der Portentiometer ausliest, um die Kommunikation auf fehlerfreiheit zu prüfen.

Anschließend wird der Fokus auf die Umwandlung von Filterparameter in den beschreibenen Bitwert gelegt, der den Widerstandswert des Potentiometer repräsentiert. Dafür müssen die bekannten Formeln aus [1] in Funktionen eingebaut werden. Diese nehmen die gewünschten Filterparameter als Eingabe entgegen und geben den entsprechenden Wiperwert für das jeweilige Poti zurück. Dabei wird sichergestellt, dass die Werte immer im gültigen Bereich von 0 bis 255 liegen. Zudem wird der ausgegebene Bitwert gerundet, damit das Poti mit den Ergebnissen arbeiten kann. Zur besseren Kontrolle gibt zuletzt eine weitere Funktion die aktuell eingestellten Widerstandswerte ausgibt.

Darauf folgend wird die wlan-Schnittstelle der mcu eingerichtet. Der Pico soll hierbei automatisch eine Verbindung mit dem angegebenen wlan aufbauen, um die Filterparameter plattformunabhängig über einen http-Server im Browser eines PCs (im gleichen Netzwerk) anpassen zu können. **Da für die html-Programmierung keine Vorkenntnisse vorhanden sind, wird dieser Teil mit hilfe eines Large Language Models (LLM) umgesetzt.** Der Server generiert hierbei eine html-Seite, die neben den Eingabefeldern für Frequenz, Güte und Verstärkung auch eine tabellarische Übersicht der aktuellen Zustände enthält. Diese Tabelle wird im laufe der Arbeit noch erweitert, um eine bessere Übersicht über die Stellung der Wiper und derern Genauigkeit zu erhalten.

Damit das System nach Anlegen der Versorgungsspannung selbstständig startet, ohne den

Code in Thonny ausführen zu müssen, wird das Skript als main.py im Flash-Speicher der mcu hinterlegt. Für die eigenständige Nutzung des Systems ist es zudem wichtig, dass der Server unter einer IP-Adresse erreichbar ist, die sich bei Neustart nicht ändert. Leider sind Protokolle wie mDNS nicht auf der mcu unter MicroPython verfügbar, weswegen der Pico im Access-Point-Modus betriben wird. Hierbei erzeugt der Pico ein eigenes lokales Netzwerk mit der SSID „Pico-Filter“. Durch die Konfiguration einer statischen IP-Adresse (192.168.4.1) ist gewährleistet, dass das Steuergerät den HTTP-Server nach der Verbindung mit dem Netzwerk jederzeit unter derselben Adresse erreicht.

1.3.1 Frequenzdetektion über Nulldurchgangszähler