Brandenburgische Technische Universität Cottbus - Senftenberg

BRANDENBURGISCHE TECHNISCHE UNIVERSITÄT COTTBUS - SENFTENBERG

Fakultät 3 für Maschinenbau, Elektrotechnik und Wirtschaftsingenieurwesen
Lehrstuhl Kommunikationstechnik
Prof. Dr.-Ing. habil. Matthias Wolff
Bachelorarbeit

Implementierung eines LTI Systems in ein „Raspberry Pi“ Einplatinen-Computer

Vorgelegt von

Student	Georg Klann
Studiengang	Informations- und Medientechnik
Matrikelnummer	3014780
Geburtsdatum	29.07.1990
Adresse	Petersilienstraße 1, 03046 Cottbus
E-Mail	klanngeo@b-tu.de
Betreuer	Dipl. Ing. Christian Richter

Cottbus, den 01.09.2018

Aufgabenstellung

In der Vorlesung „Grundzüge der Kommunikationstechnik“ wird ein Demonstrator für lineare zeitinvariante Systeme (LTI) in Form eines Hardware-Moduls verwendet. Der Demonstrator soll anliegende Signale in Echtzeit verarbeiten und am Audioausgang ausgeben können.

Gegenstand der Arbeit ist die Nachbildung von akustischen Umgebungen im Zeit- bzw. Frequenzbereich auf Grundlage vorliegender Matlab-Programme sowie die Umsetzung auf eine Hardware-Zielplattform

Für die praktische Umsetzung der vorliegenden Algorithmen ist das vorhandene Raspi 3 Kit sowie Matlab/Simulink zu verwenden. Dabei ist eine Experimentierumgebung zu entwickeln und zu testen, die für die weitere Nutzung als dauerhaften Entwicklungsplatz für die Umsetzung von Matlab-Code auf die Hardware Zielplattform geeignet ist.

Kurzfassung

Diese Arbeit befasst sich mit der Möglichkeit den Raspberry Pi Ein-Platinen-Computer in Verbunding mit dem Programm MATLAB Simulink für Hörsaalexperimente des Lehrstuhls Kommunikationstechnik zu verwenden. Dabei wird ein bereits vorhandenes Experiment zur Verdeutlichung eines LTI-Systems sowie ein neues Hörsaalexperiment zur Erzeugung eines Hallklangs versuchsweise in Simulink implementiert und auf einen Raspberry Pi gespielt.

Diese Arbeit umfasst die theoretischen Grundlagen der Experimente, die Herleitung der Arbeitsweise und Durchführung der Experimente.

Der Verfasser/Die Verfasserin erklärt, dass er/sie die vorliegende Arbeit selbständig,
ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt
oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht.
Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

_____	_____
Ort, Datum	Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ADC	Analog to Digital Converter (Analog zu Digital Wandler)
DAC	Digital to Analog Converter (Digital zu Analog Wandler)
DF	Direktform
DFT	Diskrete Fourier-Transformation
DSP	Discrete Signal Processing (Diskrete Signalverarbeitung)
dt.	deutsch
FFT	Fast Fourier Transformation (Schnelle Fourier-Transformation)
FIR	Finite Impulse Response (Endliche Impulsantwort)
GB	Gigabyte
GHz	Gigahertz
GPIO	general purpose input/output (Allzweck Eingang/Ausgang)
I2C	Inter-Integrated Circuit
IIR	Infinite Impulse Response (Unendliche Impulsantwort)
IP	Internet Protocol
LAN	Local Area Network
LTI	linear,time-invariant (linear, zeitinvariant)
MIPS	Million Instructions Per Second (Millionen Instruktionen pro Sekunde)
n.d.	nicht definiert
q.e.d	quod erat demonstrandum (was zu beweisen war)
RasPi	Raspberry Pi
SD	Secure Digital
u.a.	unter anderem
UDP	User Datagram Protocol
USB	Universal Serial Bus
z.B.	zum Beispiel

1. Einleitung

Der Lehrstuhl Kommunikationstechnik führt im Rahmen der Veranstaltung „Einführung in die Kommunikationstechnik“ ein Hörsaalexperiment durch um den Studierenden die Grundlagen der Systemanalyse nahezulegen.
In diesem Experiment wird ein bekanntes Signal am Eingang einer Black Box, die ein LTI-System darstellt, angelegt und die daraus resultierende Systemantwort gemessen. Die ein- und ausgehenden Signale werden auf einem Oszillographen dargestellt und die Ergebnisse mit den Studierenden diskutiert. Mit Kippschaltern des technischen System können verschiedene Veränderungen des Eingangssignals erzielt werden.
Das Ziel dieser Arbeit ist es das Hörsaalexperiment mithilfe von Raspberry Pis und MATLAB Simulink zu realisieren. Zusätzlich zum bestehenden Hörsaalexperiment soll ein Modus zur Erzeugung eines sogenannten Faltungshalls bzw. Halleffektes ergänzt werden. Dabei soll eine Tonaufnahme in Echtzeit um den Hall bzw. das Echo ergänzt werden was für einen bestimmten Ort typisch wäre. Dafür werden zunächst die Hintergründe der verwendeten Technik erläutert und anschließend die technischen und theoretischen Grundlagen der jeweiligen Experimente erklärt. Für das LTI-Experiment wird außerdem der Aufbau und die Durchführung des Experiments erläutert. Für das Hallfaltungsexperiment werden mögliche Algorithmen, deren Realisierung in Simulink und deren Verwendbarkeit im Kontext des Experiments betrachtet.

2. Verwendete Technik

2.1 Raspberry Pi

Der Raspberry Pi oder RasPi (siehe Abbildung 1) ist ein von der britischen Raspberry Pi-Stiftung entwickelter Einplatinencomputer welcher erstmals im Februar 2012 auf den Markt kam.

Die verwendete Version des RasPi ist die im Februar 2016 veröffentliche Version 3b. Diese Version verfügt über:

- vier USB 2.0 Buchsen,
- eine 4-Pol 3,5mm Audiobuchse,
- einen HDMI-Anschluss,
- einen Ethernet-Anschluss,
- eine 40-Pin Stiftleiste (von denen 26 als GPIO verwendbar sind),
- sowie einen Micro-SD Steckplatz.

Die Stromversorgung findet über eine Micro-USB Buchse mit maximal 4W (5V/800mA) statt. Die Rechenleistung wird über einen 64 Bit 4-Kern ARM Cortex-A53 mit jeweils 1,2GHz pro Kern als Prozessor und 1 GB RAM erzeugt [RASP].

Abbildung 1: Raspberry Pi 3b
Bildquelle: [RPTV]

2.2 MathWorks MATLAB

MATLAB ist eine von MathWorks entwickelte Entwickleroberfläche für Ingenieure mit dem verschiedene numerische Simulationen durchgeführt und bewertet werden können (siehe Abbildung 2). Durch die Verwendung so genannter „Toolboxen“ ist MATLAB je nach Anwendungsgebiet frei konfigurier- und erweiterbar. Diese Toolboxen sind Vergleichbar mit Bibliotheken in anderen Programmiersprachen. Diese ermöglichen es auf eine größere Menge von Funktionen und Funktionalitäten zuzugreifen (wie z.B. Funktionen für die Erfassung und Auswertung von Statistiken, Strömungsmodelle oder Operationen für komplexe Matrizenberechnungen).

Für diese Arbeit ist die Toolbox Simulink von besonderer Bedeutung.

Die in dieser Arbeit verwendeten Versionen von MATLAB sind 2016b, 2017b sowie 2018a (Version 9.4.0.813654 vom 23. Februar 2018).

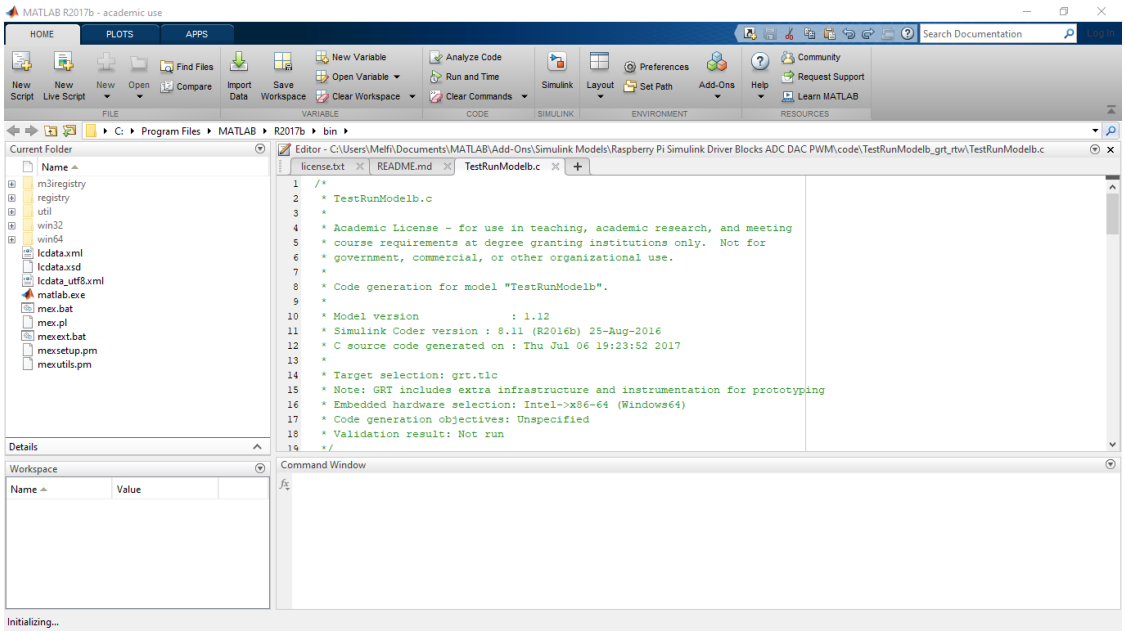


Abbildung 2: Benutzeroberfläche von MathWorks MATLAB
Bildquelle: eigene Darstellung

2.3 MATLAB Simulink

Bei Simulink handelt es sich um eine Entwicklungsumgebung in der über Funktionsblöcke ein Programm erstellt und simuliert werden kann (siehe Abbildung 3). Programme lassen sich auf die Hardware (wie z.B. den Raspberry Pi) übertragen. Simulink wurde aufgrund der Effizienz des erzeugten Quelltextes gewählt. Vor allem die Zeit zur Ausführung der Quelltexte auf der gewählten Hardware stand im Vordergrund.

Das hat jedoch zur Folge, dass die Menge der zur Verfügung stehenden Funktionen auf die Vorgegebenen beschränkt sind. Durch eine Vielzahl von Zusatzpaketen, die von MathWorks oder anderen Nutzern erstellt worden sind, kann die Anzahl der verwendbaren Blöcke für bestimmte Anwendungsfälle erweitert werden (wie z.B. durch die Einbettung der Ein- und Ausgangsblöcke des Raspberry Pi).

Ein besonderer Block in Simulink ist der MATLAB-Funktionsblock. Dieser ermöglicht es MATLAB-Quelltext direkt zu verwenden. Die Effizienz dieser selbst geschriebenen Blöcke ist jedoch nicht unbedingt vergleichbar mit den nativen Simulink-Blöcken.

Die Verwendete Simulink-Version ist 9.1(R2018a) vom 06.02.2018.

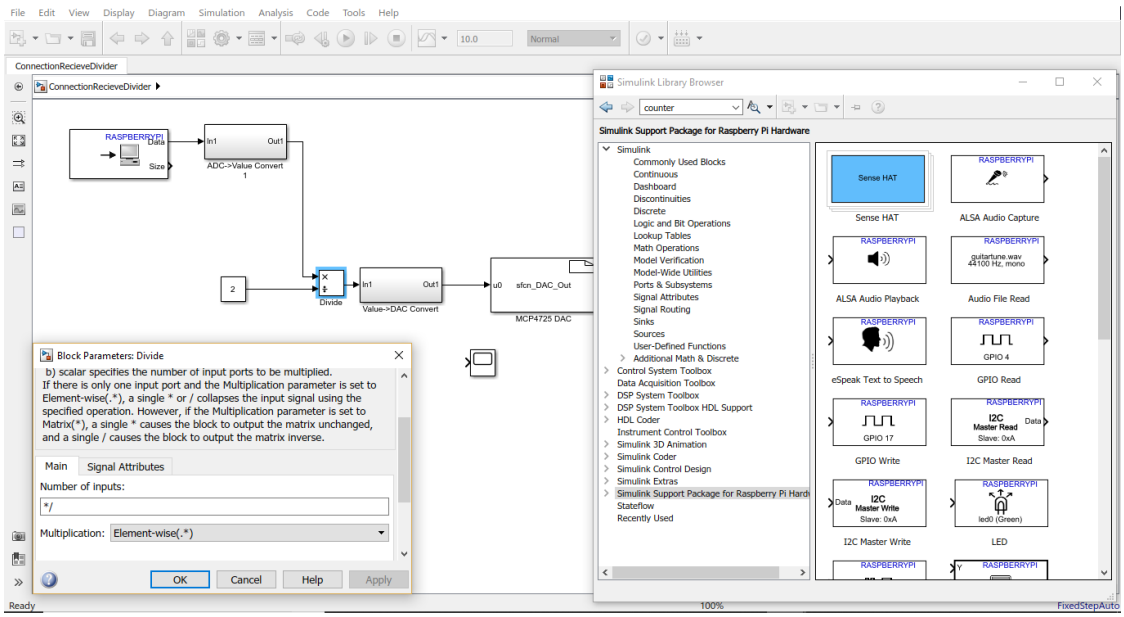


Abbildung 3: Benutzeroberfläche Simulink inkl. Bibliothek- und Parameterfenster (2018a) Bildquelle: eigene Darstellung

2.4 MCP4725 DAC

Der MCP4725 (siehe Abbildung 4) ist ein Einzelkanal DAC (Digital to Analog Converter) mit einem gepufferten 12-Bit Spannungsausgang und einem nichtflüchtigen Speicher, welcher den letzten Spannungsausgangscode auch nach Verlust der Arbeitsspannung beibehalten kann.
Der MCP4725 wird über das I2C Interface des Raspberry Pi angesteuert und bezieht von diesem eine Arbeitsspannung V_{DD} von 3,3V. Diese dient gleichzeitig als Vergleichsspannung wodurch sich Ausgangsspannungen zwischen 0 und 3,3V ergeben [MC47].

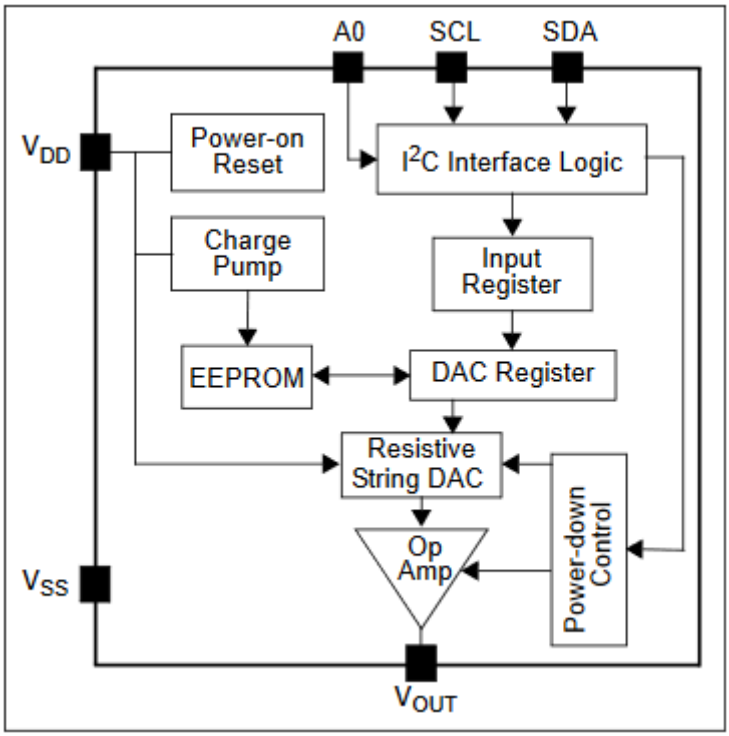


Abbildung 4: Blockdiagramm des MCP4725
Bildquelle: [MC47]

2.5 MCP3008

Der MCP3008 (siehe Abbildung 5) ist ein 10-Bit Digital zu Analog Wandler (DAC) mit On-Board Sample and Hold Schaltkreisen. Der MCP3008 hat 8 Eingangskanäle an denen unterschiedliche Eingangssignale anliegen und einzeln ausgelesen werden können. Der Chip bezieht seine Arbeitsspannung V_{DD} sowie seine Vergleichsspannung V_{REF} aus den 3,3V-Pins der Stiftleiste des Raspberry Pi. Die 10-Bit Quantisierung ermöglicht es dabei die eingehenden

Signale mit einer Auflösung von $\frac{V_{REF}}{2^{10}-1}=\frac{3{,}3\,V}{1023}=3{,}22\,mV$ zu quantisieren [MC38].

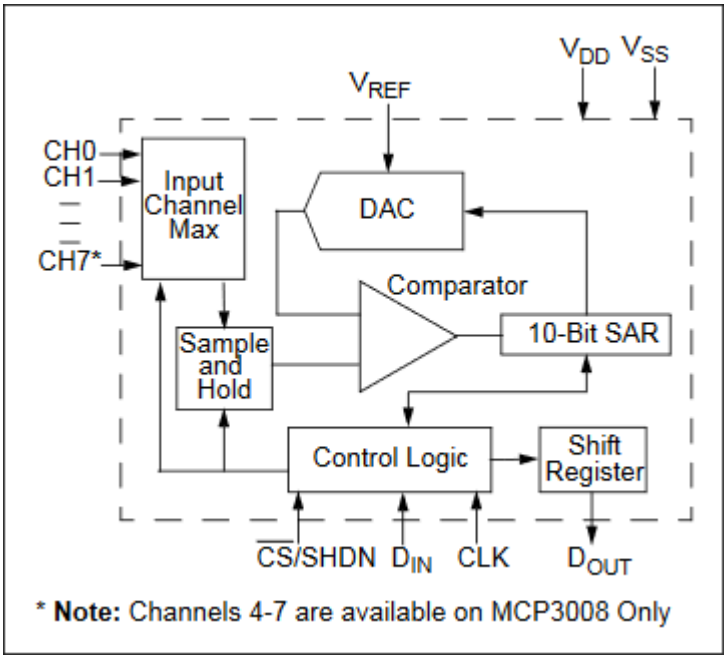


Abbildung 5: Blockdiagramm des MCP3008
Bildquelle: [MC38]

3. Modernisierung des Hörsaalexperiments

3.1 Theoretische Grundlagen

3.1.1 Black Box

Bei dem Laborversuch handelt es sich um einen Black Box-Versuch der ein LTI-System darstellen soll. Bei einem Black Box-Versuch wird ein bekanntes Eingangssignal in ein unbekanntes System gegeben. Anschließend wird das Ausgangssignal gemessen und mit dem Eingangssignal verglichen. Dieser Vorgang wird wiederholt bis eine Hypothese über das Verhalten der Black Box ermittelt werden kann. Diese Hypothese wird über eine Reihe von Testeingängen geprüft, welche diese entweder bestätigen oder widerlegen.

3.1.2 LTI-System

Bei einem LTI-System (dt. lineares zeitinvariantes System) handelt es sich um ein System für das die Eigenschaften der Linearität und der Zeitinvarianz gelten.

Ein System ist genau dann linear wenn das Superpositionsprinzip oder Überlagerungsprinzip in diesem gilt. Führt das System eine Transformation durch die sich als Funktion $g(t)=T\{f(t)\}$ darstellen lässt, mit $f(t)$ als beliebigem Eingangssignal und $g(t)$ als Systemantwort so gilt, dass die Summe aller einzelnen Systemantworten $g_i(t)$ gleich der Summe einer beliebigen Anzahl Eingangssignale $f_i(t)$ ist. Also gilt, dass

$\sum_i a_i g(t)=\sum_i a_i T\{f_i(t)\}=T\left\{\sum_i a_i f_i(t)\right\}$ sein muss [KRKA].

Die Zeitinvarianz eines Systems ist genau dann gegeben, wenn eine Zeitverschiebung um ein konstantes t_0 keinen Einfluss auf das System hat und die resultierende Systemantwort damit ebenfalls um t_0 verschoben ist. Es gilt hier also $T(f(t-t_0))=g(t-t_0)$. Damit ist die Systemantwort unabhängig vom Zeitpunkt der Anregung [KRKA].

3.1.3 Diskrete Signale

Basierend auf ihrer Definition im Zeit- oder Wertebereich können Signale als kontinuierlich oder diskret beschrieben werden.Ein Signal wird als zeitabhängige Funktion S(t) angenommen. Wenn dieses Signal zwischen seinem Anfangszeitpunkt T_0 und seinem Endzeitpunkt T_E durchgehend definiert ist gilt ,dass dieses Signal zeitkontinuierlich ist. Es existiert also im Intervall $t \in [T_{0,} T_E]$ kein Punkt t_u für den gilt $S(t_u)$ = n.d. .

Zeitdiskrete Signale sind nicht über alle Zeitpunkte im Intervall T_0 bis T_E definiert, sondern nur an bestimmten Abtastpunkten innerhalb dieses Intervalls. In der Regel haben diese Abtastpunkte den selben Abstand zueinander (Äquidistanz). Dies muss aber nicht immer der Fall sein.

Ähnliches gilt für die Funktionswerte von Signalen. Bei wertekontinuierlichen Signalen kann das Ergebnis von S(t) eine unendliche Menge von Werten annehmen. Bei wertediskreten Signalen sind alle Ergebnisse von S(t) in einer endlichen, vordefinierten Menge von Ergebnissen. Wenn ein Messwert von S(t) nicht in dieser Menge liegen sollte, wird der Messwert basierend auf dem Aufnahmesystem angepasst, so dass der Messwert innerhalb dieser Menge von vordefinierten Ergebnissen liegt (z.B. durch Rundung des Ergebnisses) [MAPR].

3.2 Bisheriger Versuchsaufbau

Der Versuchsaufbau im Allgemeinen findet als ein Black Box-Versuch statt. Die Black Box beinhaltet in diesem Fall einen Arduino Ein-Platinen Computer, dessen Modus über vier Kippschalter gesteuert wird (siehe Abbildung 6). Die dabei vorhandenen Modi sind „Dämpfung eines Signals“, „Phasenverschiebung eines Signals“, „Bandpassfilter“, „Integration eines Signals“ sowie „Ableitung eines Signals“. Das hauptsächliche Problem, welches mit dem Arduino besteht, ist, dass das Simulationsprogramm im Arbeitsspeicher abgelegt wird. Sobald der Arduino von seiner Spannungsquelle getrennt wird geht auch das aufgespielte Programm verloren und der Chip muss neu bespielt werden. Ein weiteres Problem besteht darin, dass die Technik die verwendet wird um den Arduino zu bespielen bereits sehr veraltet ist. Das Programm das dazu

verwendet wird ist auf Rechnern mit einem Betriebssystem nach Windows XP nicht verwendbar und die Kompatibilität des Quelltextes mit neueren Versionen des Programms wurde nicht geprüft.

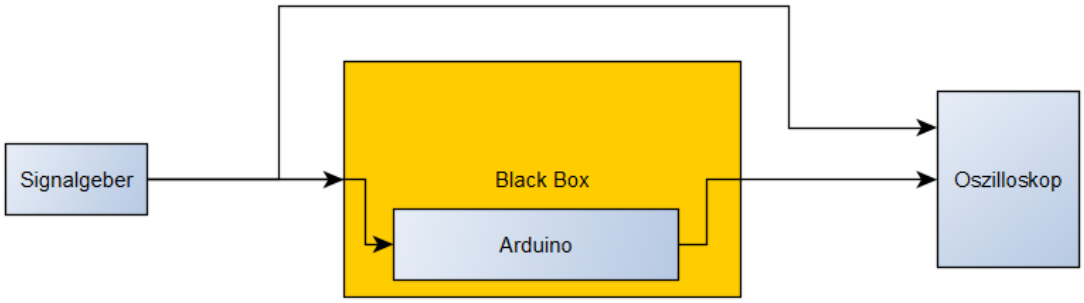


Abbildung 6: Schematischer Aufbau des alten Hörsaalexperiments
Bildquelle: eigene Darstellung

3.3 Erarbeiteter Versuchsaufbau

Der neue Aufbau sollte nach Aufgabenstellung mit RasPis als simulierende Computer erfolgen. Die Wahl des zu verwendenden Dateneingangs lag zwischen der GPIO-Schnittstelle und der 3,5mm Klinken-Buchse. Für beide Eingänge stellte die verwendete Version von Simulink (2016b) Funktionsblöcke zur Verfügung um Signale aus diesen Schnittstellen auszulesen. Jedoch existierten in MATLAB 2016b keine Blöcke die es ermöglichen einen Audioeingang des Mikrofons zu simulieren. Das würde Testvorgänge innerhalb der Simulation erschweren wenn die Klinken-Buchse als Schnittstelle gewählt würde. Bei der GPIO-Schnittstelle können bis zu 26 verschiedene Signale gelesen werden. Allerdings stellte sich bei frühen Versuchen heraus, dass die Pins des RasPi nur digitale Signale einlesen können. Die eingehenden und auszulesenden Signale liegen aber in analoger Form vor. Dadurch muss vor das System ein ADC und hinter das System ein DAC geschaltet werden (siehe Abbildung 7). Da zusätzliche verwendete Teile mit Simulink kompatibel sein müssen wurde die Menge der möglichen Komponenten auf den MCP3008 und ADS1115 als ADC sowie den MCP4725 und SoftPWC als DAC gesetzt. Für diese Teile steht eine funktionierende Bibliothek an Treiberblöcken zur Verfügung. Vom Lehrstuhl wurden Exemplare des MCP3008 und MCP4725 bereitgestellt, die im finalen Aufbau verwendet werden.

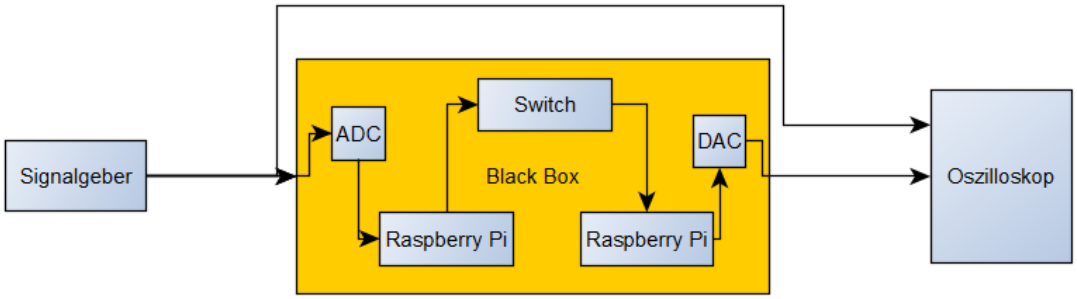


Abbildung 7: Schematischer Aufbau des neuen Hörsaalexperiments
Bildquelle: eigene Darstellung

3.4 Durchführung

Zunächst müssen die verwendeten Wandler an den RasPi angeschlossen werden. Dieser Anschluss führte zu Problemen. Der ADC und der DAC verwenden nach Verkabelung entsprechend des Handbuchs die selben Versorgungs-Pins des RasPi. Eine parallele Versorgung beider Wandler durch einen RasPi hat zu unkontrolliertem Verhalten der Wandler geführt. Aus diesem Grund wurde die Arbeit auf zwei RasPis verteilt.
Die RasPis verwenden statt einer internen Festplatte den Speicher der eingelegten SD-Karte. Um den RasPi mit Simulink nutzen zu können muss die SD-Karte beider RasPis mit einer MATLAB-Firmware vorbereitet werden. Die Firmware ist vergleichbar mit einem Betriebssystem. Zusätzlich ist zur Verwendung der genannten AD- bzw. DA-Wandler das Treiberpaket WiringPi notwendig, welches manuell aus dem Internet auf die RasPis heruntergeladen und installiert werden muss. Anschließend können die RasPis direkt über Ethernet konfiguriert und bespielt werden (siehe Anlage 1).
Die zuerst verwendete Version, MATLAB 2016b, hat bei automatischer Vergabe der Netzwerkoptionen beiden verwendeten RasPis die selbe IP-Adresse zugewiesen. Aufgrund dessen war eine Kommunikation zwischen den RasPis nicht möglich. Durch eine manuelle Vergabe der IP-Adresse des zweiten RasPi verlief das Versenden von Daten über die UDP Package Send/Recieve Blöcke im lokalen Netzwerk ohne Probleme. Dadurch war jedoch der Zugriff auf das Internet nicht mehr möglich. Bei einem erneuten Bespielen der SD-Karten mit einer neue MATLAB-Firmware wird die SD-Karte formatiert. Aus diesem Grund

muss für das erneute Herunterladen und Installieren des Treiberpakets WiringPi eine Verbindung zum Internet hergestellt werden. Mit dem Wechsel auf die MATLAB-Version 2017b wurde das Problem der identischen Vergabe der IP-Adressen behoben. Die MATLAB-Version 2018a stellte die Funktionalität nur automatischen Ausführung von Programmen bei Systemstart zur Verfügung. Bis dahin musste das Programm manuell dem Systemstart hinzugefügt werden.
Da über die Versionen hinweg keine der verwendeten Blöcke als veraltet entfernt wurden und die in MATLAB 2016b erstellten Programme ohne Probleme von MATLAB 2018a erkannt und kompiliert wurden zogen die Versionswechsel keine weiteren Nachteile mit sich.
Das neue System besteht damit aus zwei RasPis, welche über einen Switch zu einem lokalen Netzwerk verbunden sind. Eine direkte Kommunikation der RasPis über Patch- oder Crossover-Kabel konnte nicht hergestellt werden.
Der erste Pi des Systems wurde damit zum „Sender“ dessen einzige Aufgabe darin besteht die Werte des MCP3008 einzulesen und mithilfe des UDP Send Blocks an den Empfänger-Pi zu senden (siehe Abbildung 8).

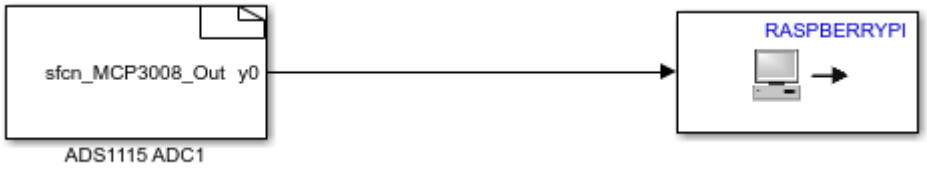


Abbildung 8: Programm des Sender-Raspberry Pi
Bildquelle: eigene Darstellung

Der Empfänger-Pi nimmt das eingehende Signal aus dem Netzwerk mithilfe des UDP Recieve Blocks auf. Dieses Signal wird im ADC→Value Converter-Block auf seinen ursprünglichen Spannungswert umgewandelt (siehe Abbildung 9).
Der MCP3008 arbeitet mit einer 10-Bit Quantisierung, welches das Eingangssignal E_1 als Ganzzahl zwischen 0 und 1023 übergibt. Um die ursprüngliche Spannung herauszufinden wird zunächst der Anteil an der Eingangsspannung über die Formel $p_u = \frac{E_1}{1023}$ ermittelt. Um daraus die ursprüngliche Spannung zu erhalten wird p_u mit der Vergleichsspannung U_{REF} =3,3V multipliziert.

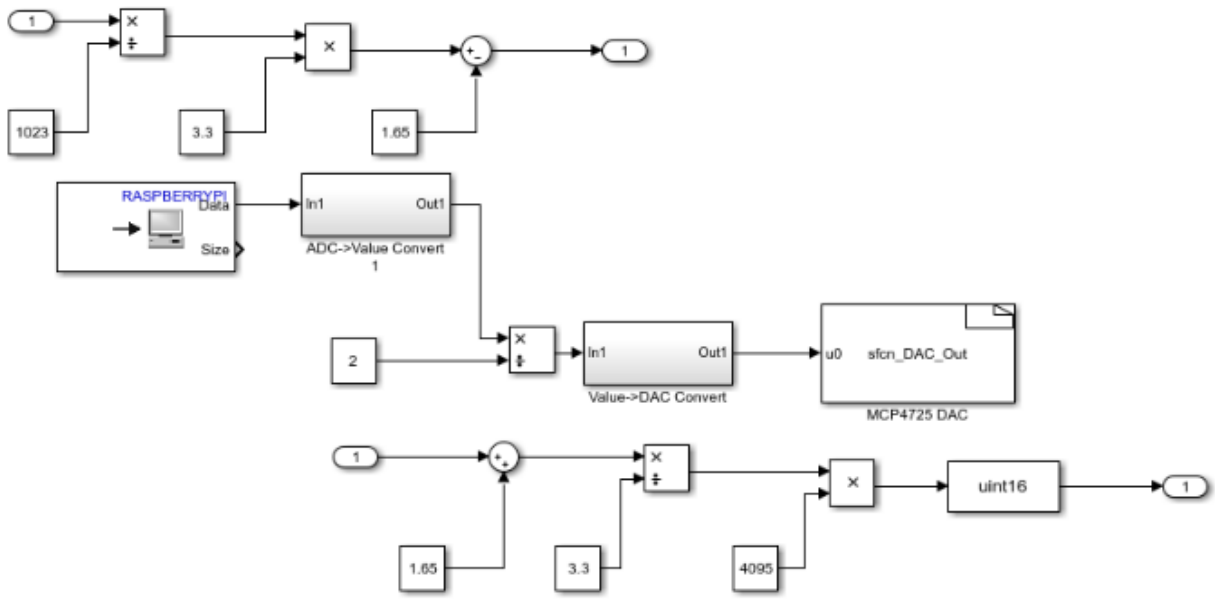


Abbildung 9: Beispielhaftes Programm des Empfängers
Bildquelle: eigene Darstellung

Da der MCP3008 nur Spannungen zwischen 0 und U_{REF} aufnehmen kann, muss die resultierende Eingangsspannung um $\frac{U_{REF}}{2}$ reduziert werden um eine harmonische Schwingung um die 0V Achse zu erhalten. Die resultierende Spannung basierend auf dem eingehenden Signal lässt sich damit mit der Formel $U=(\frac{E_1 \cdot 3{,}3V}{1023})-\frac{3{,}3V}{2}$ ermitteln.

Diese Spannung kann, je nach Anwendungsfall, bearbeitet werden. In Abbildung 9 wird das Signal um einen Faktor 2 gedämpft und danach über den Value → DAC Block in einen für den MCP4725 lesbaren Bitwert umgewandelt. Dabei werden die Rechenschritte aus dem DAC→Value Block in umgekehrter Reihenfolge ausgeführt. Da es sich beim MCP4725 um einen 12-Bit Wandler handelt wird das Ausgehende p_u mit 4095 (2^{12}-1) multipliziert, woraus sich die Formel $A_1=\frac{U-1{,}65V}{3{,}3}\cdot 4095$ ergibt. Der Modus des Systems kann durch das Austauschen der SD-Karte des Empfängers erreicht werden. Dazu muss das System neu gestartet werden. Auf diese Weise kann das System einfach

erweitert werden da für neue Anwendungsfälle nur die Erstellung einer neuen SD-Karte mit einer Variante des grundlegenden Empfängerprogramms benötigt wird. Eine Anleitung zum Erstellen solcher SD-Karten findet sich in Anhang 1.

3.5 Ergebnisse

Die Verwendung eines RasPi zur Durchführung des Hörsaalexperimentes bringt Vor- und Nachteile mit sich. Einer der wohl größten und vorwiegendsten Vorteile ist dabei, dass die Programme nicht mehr auf flüchtigen Speichern abgelegt werden und die Black Box damit nicht vor jedem Einsatz neu bespielt werden muss. Ebenfalls von Vorteil ist, dass das Bespielen der SD-Karten über MATLAB Simulink erfolgt für das die BTU Cottbus-Senftenberg eine campusweite Lizenz für Studierende und Angestellte besitzt. Weiterhin betreibt MathWorks das Programm MATLAB als kommerzielles Produkt mit weitreichenden Anwendungsspektren, was die Wahrscheinlichkeit, dass die Software, die zur Erstellung oder Wartung der genutzten SD-Karten benötigt wird vom Markt verschwindet zumindest in der absehbaren Zukunft eher gering ist.

Nachteile zeigen sich in den verwendeten ADC und DAC da diese eine Eingabe- bzw. Ausgabespanne von 0V bis 3.3V haben. Diese Bauteile sind damit nicht in der Lage negative Spannungen zu erzeugen oder zu lesen, welche in einer regulären Wechselspannung vorkommen. Damit muss der im Versuchsaufbau vorkommende Signalgeber einen Offset und maximale Amplitude von $\frac{U_{max}}{2}$ mit U_{max} = 3,3V erzeugen, damit an keinem Punkt des erzeugten Signals eine Amplitude unter 0V oder über 3,3V erreicht wird. Damit dieser Ausgleich den Studierenden nicht auffällt, muss dabei gleichzeitig der Oszillograph auf einen negativen Offset von $\frac{U_{max}}{2}$ eingestellt werden. Der Aufbau mit zwei RasPis die durch einen Switch miteinander verbunden sind ist ebenfalls ein großer Nachteil, da sich die Bestandteile des Systems entweder alle in der selben Black Box befinden müssen und die damit verwendete Anzahl von Kabeln eine mögliche Fehlerquelle sein kann oder das sich der Switch außerhalb der Black Box befindet, was die Illusion des geschlossenen Systems zerstört.

Zu beachten bei der Durchführung des Experimentes ist, dass die Frequenz des eingehenden Signals nach dem Nyquist-Shannon-Abtasttheorem nicht höher sein darf als die in Simulink angegebene Abtastfrequenz. Aufgrund der Verwendung einer Netzwerkverbindung sind die Signale die im Empfänger-RasPi verwertet werden diskret. Das kann bei einer zu hohen Frequenz des Eingangssignals dazu führen, dass einige der Signalwerte nicht aufgenommen werden können oder das in der erzeugten Systemantwort deutliche Stufen zu erkennen sind.
Eine andere Möglichkeit, der zunächst nicht weiter nachgegangen wurde, ist die Signale über die Audio-Klinken-Buchse einzulesen oder auszugeben. In MATLAB 2017b wurde ein Funktionsblock eingeführt um ein Signal, wie es vom Mikrofoneingang erzeugt werden würde, zu simulieren. Erste Versuche mit den Funktionen Integration und Ableitung über native Simulink-Blöcke lieferten jedoch keine zufriedenstellenden Ergebnisse.

4. Implementierung einer Hallfaltung

4.1 Versuchsbeschreibung

Den Studierenden soll in einem Hörsaalexperiment die Funktionsweise einer Hallfaltung erörtert werden und dabei die Operationen der Faltung und der Fouriertransformation und deren Bedeutung in der Nachrichtentechnik gezeigt werden. Dafür wird über ein Mikrofon eine Tonaufnahme in ein Black Box-System eingespielt. Diese wird darin verarbeitet und mit einem nachträglich hinzugefügten Halleffekt wieder ausgegeben.
Im Folgenden werden bestimmte Begriffe der Nachrichtentechnik im Kontext der Hallfaltung verwendet. Diese Begriffe definieren sich für dieses Kapitel wie folgt:

- Impulsantwort: Das Echo, das ein Raum (System) auf einen auditiven Impuls zurückgibt.
- Nutzsignal: Das Signal auf welches die Hallfaltung angewendet wird.
- Systemantwort: Das Ergebnis der Faltung von Nutzsignal und Impulsantwort. Damit ist die Systemantwort das Echo auf ein beliebiges auditives Signal durch den Raum.

4.2 Theoretische Grundlagen

4.2.1 Schall

Als Schall werden Luftdruckschwankungen bezeichnet, die durch mechanische Schwingungen von Festkörpern (wie z.B. Saiten eines Musikinstruments oder Schwingung einer Lautsprechermembran) entstehen und von Organismen oder technischen Geräten auditiv aufgenommen werden können [LFUB].
Ein einzelner Schall kann dabei aus einer Kombination von Schwingungen verschiedener Frequenz, Phase und Amplitude bestehen. Die reinste Form des Schalles ist eine harmonische Sinusschwingung. Ein Beispiel für die Überlagerung von Schällen ist in Telefonen mit Tonwahl vorhanden (Mehrfrequenzwahlverfahren). Jeder Spalte und Zeile des Nummernfeldes ist ein Ton mit bestimmter Frequenz zugeordnet. Wird eine Taste gedrückt wird der Zeilenton und der Spaltenton der jeweiligen Taste zusammen abgespielt und

damit überlagert. Das daraus resultierende Geräusch kann von einer Telefonanlage als Ziffer bzw. gedrückte Taste interpretiert und weiter verarbeitet werden.

4.2.2 Fourier-Transformation

Wie bereits im vorherigen Abschnitt erläutert setzt sich ein Schall aus einer endlichen Anzahl harmonischer Sinusschwingungen mit unterschiedlichen Frequenzen zusammen. Diese bilden das Spektrum des Schalls. Mithilfe der Fouriertransformation ist es möglich einem Schall (oder einem Signal im Allgemeinen) , der sich als Funktion $f(t)$ beschreiben lässt seine Fourier-transformierte $F(\omega)$ in Form des Fourier-Integrals

$F(\omega) := \frac{1}{\sqrt{2\pi}} \int f(t) e^{-i\omega t} dt$ zuzuordnen. Eine solche Transformation ist jedoch nur dann möglich, wenn das Signal über den gesamten Zeitraum auch integrierbar ist. In der Signalanalyse wird jedoch eine Sonderform der Transformation, die diskrete Fourier-Transformation angewendet.

4.2.3 Faltung

Unter einer Faltung oder Konvolution versteht man eine Operation, die für zwei gegebene Funktionen $f(t)$ und $g(t)$ eine dritte Funktion $f * g$ erzeugt. Diese Funktion wird als $(f * g)(t) := \int f(\tau) g(t-\tau) d\tau$ definiert. Dies ist jedoch nur dann möglich wenn die Funktionen f und g in ihrem Definitionsbereich integrierbar sind. In der Nachrichtentechnik wird aufgrund des Auftretens diskreter Signale stattdessen die diskrete Faltung verwendet. Diese verwendet anstatt eines Integrals eine Summe. Die Funktion für die diskrete Faltung lautet

$$(f * g)(n) := \sum_k f(k) g(n-k)$$

und umschreibt damit nicht mehr den vollständigen Signalverlauf, sondern nur den Signalverlauf an diskreten Messpunkten.

4.2.4 Faltungshall

Ein Faltungshall ist ein Effekt bei dem das Echo eines Raumes auf ein endliches und bekanntes Signal simuliert wird. Anders als bei synthetischen Hallklängen wird für einen Faltungshall die Impulsantwort eines Raumes benötigt um den Effekt zu erzeugen. Es ist möglich diesen Faltungshall im

Zeitbereich oder Frequenzbereich der Töne zu generieren. Aufgrund des Faltungstheorems können Berechnungen die im Zeitbereich durchgeführt werden auch im Frequenzbereich durchgeführt werden. Im allgemeinen gilt hierbei das Faltungstheorem $F(f*g)=F(f)\cdot F(g)$ sowie die durch die Spiegelsymmetrie der Fouriertransformation ergebene Umkehr $F(f\cdot g)=F(f)*F(g)$. Die Symmetrie der Fourier-Transformation ist u.a. in [HAYM] beschrieben.

Der Beweis für dieses Theorem basiert auf den Definitionen für die Faltung, welche als

$$(f*g)(x):=\int f(\tau)g(x-\tau)d\tau \quad (1)$$

gegeben ist sowie die Definition der Fouriertransformation, welche durch die Formel

$$F[f(t)](p)=\frac{1}{\sqrt{2\pi}}\int f(t)e^{-ipt}dt \quad (2)$$

dargestellt wird. Wird nun (1) als $f(t)$ in (2) eingesetzt erhält man die Gleichung:

$$F[(f*g)(x)](p)=\frac{1}{\sqrt{2\pi}}\int\int f(\tau)g(x-\tau)e^{-ipx}dx\,d\tau \quad (3)$$

Wenn in (3) eine Substitution mit $u=(x-\tau)$ durchgeführt wird, wird aus der rechten Seite der Gleichung:

$$\frac{1}{\sqrt{2\pi}}\int\int f(\tau)g(u)e^{-ip(u+\tau)}du\,d\tau$$

Das Doppelintegral lässt sich dabei in

$$\frac{1}{\sqrt{2\pi}}\int f(\tau)e^{-ip(\tau)}d\tau\cdot\frac{1}{\sqrt{2\pi}}\int g(u)e^{-ip(u+\tau)}du \quad (4)$$

aufspalten. Da nach [MAPR] die Definition der Fouriertransformierten

$$F[f(t)](x)=\frac{1}{\sqrt{2\pi}}\int f(t)e^{-ixt}dt$$

ist, ist also die Fouriertransformierte von $f(\tau)$ gleich

$$F[f(\tau)](p)=\frac{1}{\sqrt{2\pi}}\int f(\tau)e^{-ip\tau}d\tau \quad (5)$$

und die Fouriertransformierte von $g(u)$ gleich

$$F[g(u)](p)=\frac{1}{\sqrt{2\pi}}\int g(u)e^{-ipu}du \quad (6)$$

Werden (5) und (6) in (4) eingesetzt ergibt sich damit, dass

$F[(f*g)(x)](p)=F[f(\tau)](p)\cdot F[g(u)](p)$ ist. q.e.d.

Im Verlauf des Kapitels wird auf verschiedene Methoden der Berechnung des Hallklanges im Zeit- sowie im Frequenzbereich eingegangen. Die möglichen Implementierungen in Simulink werden behandelt und gegenübergestellt. Anschließend wird die Verwendbarkeit der verwendeten Algorithmen für das geplante Hörsalexperiment bewertet.

4.3 Evaluation der Methoden

4.3.1 Faltung im Zeitbereich: FIR-Filter

Solange die Impulsantwort ein endliches Signal ist kann ein FIR-Filter – Kurz für Finite Impulse Response, also Endliche Impulsantwort – verwendet werden. Ein solcher Filter wird in Abbildung 10 dargestellt. Die Teilsignale b_0 bis b_{N_b} stellen dabei die Filterkoeffizienten und gleichzeitig die einzelnen Teilwerte der Impulsantwort dar. Bei jedem eingehenden Takt wird das Signal um T verzögert bevor die einzelnen Werte mit den jeweiligen Koeffizienten der Impulsantwort multipliziert werden. Die daraus resultierenden Produkte werden miteinander addiert und bilden damit das gefaltete Ergebnis für Takt n.

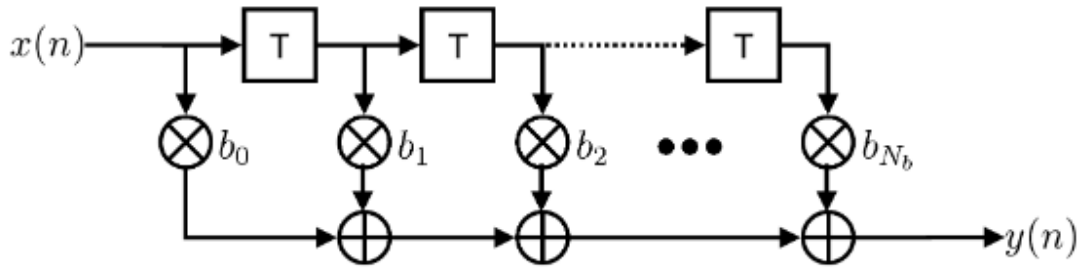


Abbildung 10: Schematische Darstellung eines FIR-Filters *Bildquelle: [JUWE]*

Beispielhaft wäre in diesem Fall $y(1)=x(1)\cdot b_0$, $y(2)=x(2)\cdot b_0+x(1)\cdot b_1$,..., $y(b_{n_b})=x(b_{n_b})\cdot b_0+x(b_{n_b}-1)\cdot b_1+...+x(2)\cdot b_{n_b-1}+x(1)\cdot b_{n_b}$.

Diese Berechnungsfunktion entspricht der Summe der diskreten Faltung

$y(k)=\sum_{n=0}^{b_{N_b}} x(n)\,g(n-k)$. Der verwendete RasPi nimmt Tonaufnahmen in 2-Kanal 44,1 kHz auf. Das entspricht 88 200 Abtastungen pro Sekunde. Bei einer beispielhaften Impulsantwort von 3 Sekunden ergeben sich damit 264 600

Multiplikationen und Additionen für einen Abtastwert des Nutzsignals. Zur Vereinfachung wird das Stereosignal in ein Monosignal umgewandelt, wodurch die zu verrichtende Rechenarbeit halbiert wird. Diese 132 300 Berechnungen müssen mit jedem Abtastwert des Nutzsignals, ebenfalls 44 100 pro Sekunde, verrechnet werden. Das führt zu jeweils über 5 Milliarden Multiplikationen und Additionen pro Sekunde. Mit den etwa 2 800 MIPS [MEDB] die der Raspberry Pi 3b erzielt liegt die Zahl der nötigen Berechnungen viermal so hoch wie die mögliche Anzahl von Rechenoperationen die der RasPi in einer Sekunde durchführen kann. Wenn jedoch die Länge der Impulsantwort und gegebenenfalls die Abtastrate der Impulsantwort angepasst wird kann eine Faltung mit dieser Methode möglich sein.

4.3.2 Faltung im Zeitbereich: IIR-Filter

Eine weitere Form der Filterung kann über einen IIR-Filter vorgenommen werden. Der allgemeine Aufbau des IIR-Filters entspricht dem des FIR-Filters. Der Unterschied besteht in darin, dass Ausgangswerte über einen Rückkopplungszweig in folgende Ausgangswerte eingerechnet werden (siehe Abbildung 11).

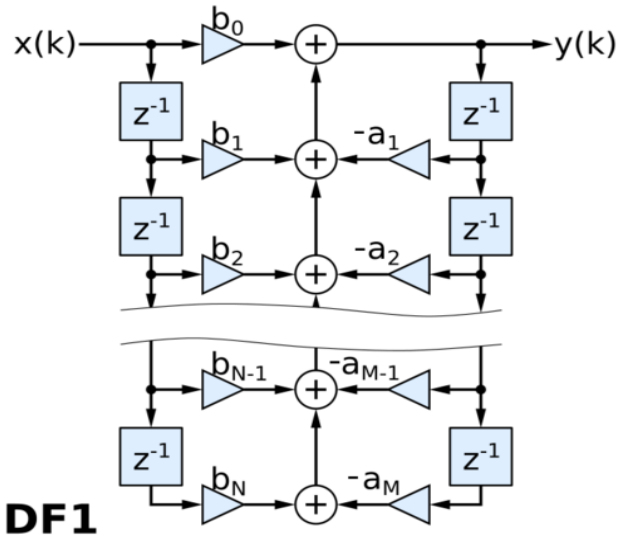


Abbildung 11: Schematische Darstellung eines DF1-IIR Filters *Bildquelle: [WPDF]*

Die Übertragungsfunktion eines DF1-IIR-Filter ergibt sich damit als

$$H(z)=\frac{\sum_{i=0}^{N} b_i \cdot z^{-i}}{\sum_{j=0}^{M} a_j \cdot z^{-j}} \; mit\, a_0=1$$ [MAPR].

Da der Aufbau eines IIR-Filters allerdings extrem sensibel gegenüber Parameterquantisierungen ist wird stark davon abgeraten diese praktisch zu verwenden [MAPR].

4.3.4 Berechnung im Frequenzbereich: Fourier-Transformation

Aufgrund des in 2.1.3 hergeleiteten und bewiesenen Faltungstheorems ist es möglich anstatt einer Faltung der Signale im Zeitbereich den Faltungshall über eine Multiplikation der Signale im Frequenzbereich zu erzeugen. Mithilfe der Diskreten Fouriertransformation ist es möglich ein Signal in sein Frequenzspektrum zu zerlegen. Die DFT (diskrete Fourier-Transformation) für ein Datenwort der Länge N ist dabei durch die Formel

$$X(k)=\sum_{n=0}^{N-1} x(n)\cdot e^{\frac{-2j\pi nk}{N}}, k\in(0,1,2,\ldots,N-1)$$ [BLCH]

definiert. Damit benötigt die Berechnung der DFT für ein Datenwort der Länge N genau N komplexe Additionen von N komplexen Multiplikationen, also N^2 Rechenschritte.

Eine 1965 veröffentliche Variante der DFT, die Schnelle Fouriertransformation (oder auch FFT, kurz für Fast Fourier Transformation), erlaubt es diese Umwandlung um einen Faktor von $\frac{N}{ld(N)}$ effizienter durchzuführen. Das bedeutet, dass bei einer Signallänge von 4096 der FFT-Algoritmus etwa 350-mal schneller ist als die reguläre DFT [BLCH]. Basierend auf dem verwendeten Algorithmus muss die Länge des zu transformierenden Signals eine bestimmte Länge haben. So müssen Signale die mit einem Radix 2-Algorithmus transformiert werden eine Länge von 2^n haben.

Da es sich bei der Berechnung der DFT um eine Summe von Produkten der Form $x(n)\cdot K_N^{nk}$ mit $K_N = e^{\frac{-2j\pi}{N}}$ handelt, kann die Summe als

$$x(0)+x(1)\cdot K_N^{k}+x(2)\cdot K_N^{2k}+...+x(N-1)\cdot K_N^{(N-1)k}$$

geschrieben werden. Aufgrund der Produktgesetze kann jedes Element mit einem $n=2p+1$ mit $p\in\mathbb{Z}$ auch als $x(n)\cdot K_N^{k}\cdot K_N^{k(n-1)}$ geschrieben werden, wobei $n-1=2p$ ist.

Damit ergeben sich zwei Mengen: die Menge der Summanden bei denen n eine gerade Zahl ist und die Menge der Summanden bei denen n eine ungerade Zahl ist. Beide Mengen enthalten die selben Faktoren der Form K_N^{nk} .

$$X(k)=x(1)\cdot K_N^{k}+x(2)\cdot K_N^{2k}+...+x(N-2)\cdot K_N^{(N-2)k}+x(N-1)\cdot K_N^{(N-1)k}$$

$$X(k)=K_N^{k}(x(1)+x(3)\cdot K_N^{2k}+x(5)\cdot K_N^{4k}...)+(x(0)+x(2)\cdot K_N^{2k}+x(4)\cdot K_N^{4k}...)$$

Die beiden Mengen enthalten dabei N/2 Elemente wobei $K_N^{2k}=K_{(N/2)}^{k}$ ist.

Die damit entstandenen Teilsummen ergeben dabei jeweils eine Summe, die der DFT der Länge N/2 entspricht. Die für diese Operationen nötigen Rechenschritte sind, Länge des Datenwortes zum Quadrat Rechenschritte. Bei einer Länge von N/2 benötigt eine Teilsumme $N^2/4$ komplexe Berechnungen, und beide Teilsummen zusammen $2N^2/4 = N^2/2$. Damit konnte durch eine Teilung der Berechnungen die Menge der Rechenschritte halbiert werden. Das ist allerdings nur dann Möglich wenn N ohne Rest durch den Teilfaktor teilbar ist. Der in der Herleitung gezeigte Algorithmus ist der Radix 2 Algorithmus da die Länge des Datenwortes in jedem Schritt um den Faktor 2 geteilt wird. Analog lässt sich dieser Algorithmus für jede andere Zahl (z.B. Radix 3, Radix 4, Radix 5 …) durchführen.

Die so umgewandelten Datenströme des Nutzsignals und der Impulsantwort werden anschließend miteinander multipliziert und über die Inverse FFT in den Zeitbereich zurücktransformiert. Da das Nutzsignal nicht endlich ist muss dieses vorher segmentiert und das Ergebnis im Anschluss mithilfe des Overlap-Add-Verfahrens welches schematisch in Abbildung 12 dargestellt wird wieder zusammengesetzt werden.

Die DFT, und in Erweiterung dessen die FFT, funktionieren nur bei periodischen Signalen. Wird das zu transformierenden Teilstücke als eine Periode eines periodischen Signals betrachtet ist die Transformation im Allgemeinen möglich.

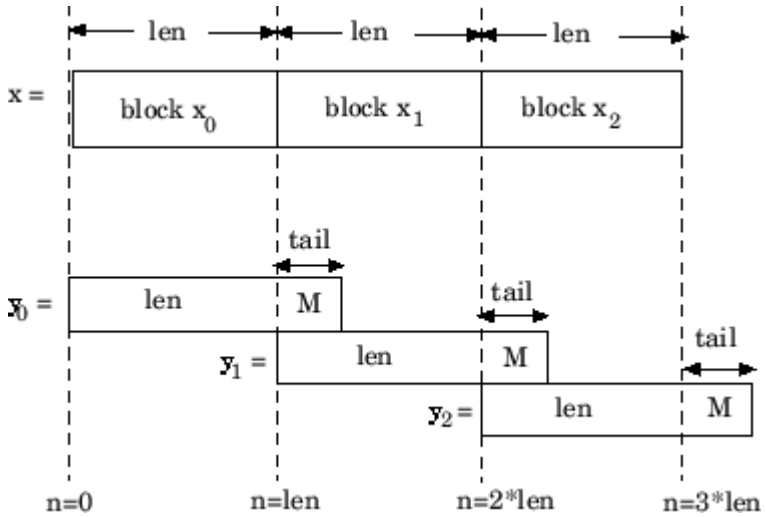


Abbildung 12: Schematische Darstellung des Overlap-Add-Verfahrens Bildquelle: [MATF]

In einer zuvor erbrachten Abschlussarbeit wurde der Versuch mit dieser Methode bei einer geringeren Datenrate (16 kHz statt 44 kHz) durchgeführt. Der Verfasser kam zu dem Schluss, dass die verwendete Hardware nicht die erforderliche Leistung erbringen kann um die Hallfaltung mit dieser Methode durchzuführen. Aus diesem Grund wurde dieses Verfahren in dieser Arbeit nicht weiter verfolgt.

4.4 Durchführung des Versuches

Die hauptsächlichen Begrenzungen unter denen die Simulation durchgeführt werden kann sind der Arbeitsspeicher des RasPi, welcher auf 1GB begrenzt ist, sowie die Anzahl der Instruktionen pro Sekunde die der RasPi durchführen kann. Double-Werte sind mit einer Größe von 64 Bit, was 8 Byte entspricht, der größte verwendbare Datentyp. Unter der Annahme das als Toleranz die Hälfte des Arbeitsspeichers für interne Prozesse genutzt wird verbleiben 500 MB oder umgerechnet 500 Millionen Byte. In diesem Speicher lassen sich 62,5 Millionen Double-Werte speichern, was etwa 1 400 eine-Sekunde-Paketen mit jeweils 44 100 Werten entspricht. Unter der Annahme, dass die Linux-Distribution, die MATLAB auf dem RasPi verwendet, den Speicher effizient verwaltet und nicht mehr verwendete Daten überschreibt, stellt die Größe des Arbeitsspeichers für die Simulation kein Problem dar. Damit müssen das Nutzsignal und die verwendete Impulsantwort an die Prozessorleistung und dessen MIPS gekoppelt werden.

Die optimale Anzahl von Werten, die ein Rahmen der Impulsantwort für eine FIR-Faltung haben sollte, darf also bei einer Berechnung nicht über die 2 800 MIPS, oder umgerechnet 2,8 Milliarden Instruktionen pro Sekunde steigen.
In diesen 2,8 Milliarden Instruktionen sollte eine ausreichend große Toleranz eingerechnet werden. Wird diese Toleranz mit 10% angenommen bleiben noch 2,52 Milliarden Instruktionen für die Berechnung übrig. Bei einer Datenrate von 44 100 Abtastwerten pro Sekunde, die das Nutzsignal besitzt, darf die

Impulsantwort maximal $\frac{2.520.000.000}{44.100} = 57.142,85714$ Abtastwerte lang sein,

was bei einer Datenrate der Impulsantwort von ebenfalls 44 100 Abtastwerten

pro Sekunde, also $\frac{57.142,85714}{44.100} = 1,29$ Sekunden entspricht.

Simulink besitzt einen Block der diese Funktionalität zur Faltung unterstützt. Dieser führt die Faltung allerdings nicht als strikten FIR-Filter aus sondern als numerische Berechnung der diskreten Faltung [MATC]. Es existiert zwar ein FIR-Filter-Block, jedoch hat dieser nur einen Eingang und kann in diesem Fall nicht zum Zweck einer Faltung verwendet werden.
Die in Simulink gegebenen Blöcke in der DSP System Toolbox ermöglichen es sich einen solchen Filter selber zusammenzusetzen. Dieser müsse jedoch für jede Länge der Impulsantwort neu aufgebaut werden. Bei 44 100 Filterkoeffizienten pro Sekunde stellt das bei längeren Impulsantworten einen zu hohen Aufwand dar.

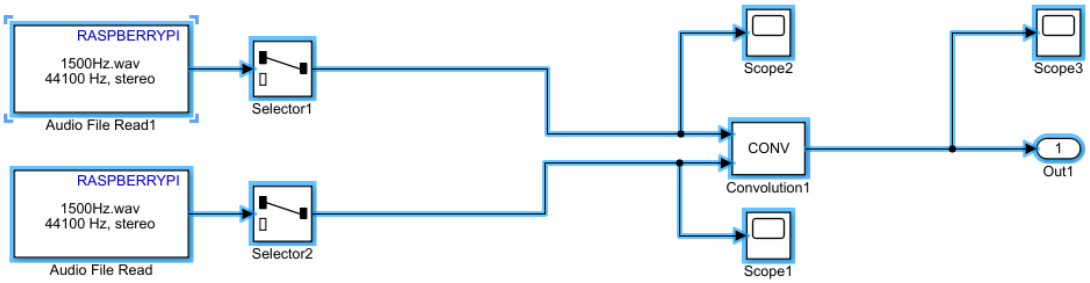


Abbildung 13: Versuchsaufbau Test der Faltung *Bildquelle: eigene Darstellung*

Der Versuchsaufbau zum Test der Faltung in Simulink baut sich wie in Abbildung 13 gezeigt auf. In den Audio File Read-Blöcken können beliebige Sounddateien hinterlegt werden. Zu Testzwecken wurde zunächst ein sinusförmiges Signal mit einer Frequenz von 1,5 kHz mit einem Stille-Signal (Nullpegel über die gesamte Zeit) gefaltet. Das Endergebnis sollte dabei ein

weiteres Nullsignal sein, da eine der Funktionen $g(t)=0$ ist wodurch die Produkte, welche innerhalb der Faltung aufsummiert werden jeweils 0 ergeben. Dieses Ergebnis ist wie in Abbildung 14 zu sehen auch eingetreten.

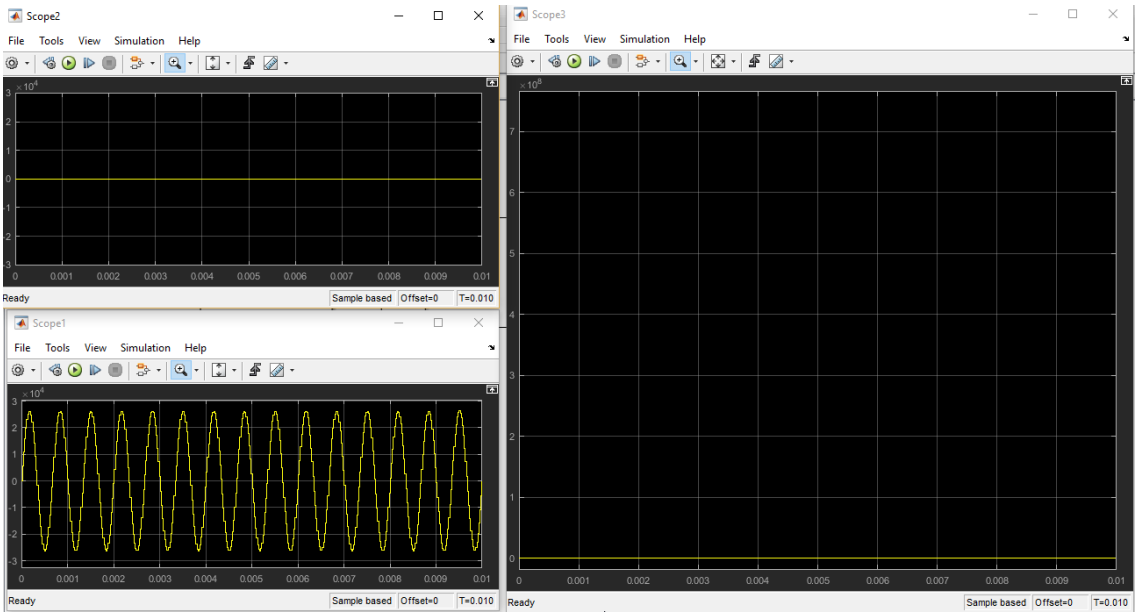


Abbildung 14: Resultat der Faltung mit g(t) = 0 und f(t) als 1,5kHz Sinuswelle
Bildquelle: eigene Darstellung

Bei einem zweiten Versuch bei dem $g(t)=2$ gesetzt wurde fiel jedoch auf, dass sich der Ergebniswert nicht wie erwartet über die ersten 44 100 Werte auf ein Maximum steigert und von dort an konstant bleibt, sondern eine konstante Verstärkung über das gesamte Signal um den Faktor 2 erzielt wurde.
Das führte zu dem Schluss, dass der Faltungsblock die Eingangssignale als vollständige Datenfenster ansieht und diese miteinander verrechnet. Die Impulsantwort muss demzufolge als vollständiges Signal über seine gesamte Länge eingelesen werden.
Ein Test mit einem Signal mit der Länge von einer Sekunde (44 100 Einzelwerte), dass einen einzelnen Dirac-Impuls beinhaltete, führte bei Simulationsdauern von über 0,6 Sekunden dazu, dass das Rechner, welcher das System simulierte, vollständig einfror. Bei kürzeren Simulationsdauern fiel auf, dass die Ergebnisse der Faltung mit einzelnen Werten des Nutzsignals nicht aufaddiert werden. Im Idealfall müssen am Faltungsblock das vollständigen Nutzsignal und die vollständigen Impulsantwort als Eingangssignale anliegen. Da es sich aber bei dem Nutzsignal um einen konstanten Datenstrom handelt kann dieser nicht in seiner Gesamtheit als ein

Vektor als Eingangssignal anliegen. Es wäre möglich das Nutzsignal zu Puffern und den gepufferten Vektor als Eingangssignal zu verwenden. Da der Faltungsblock während seiner Berechnung keine weiteren Eingangswerte zulässt müssten die Faltungsergebnisse auf eine Verschiebung der vorangegangenen Systemantworten addiert werden.
Alternativ bietet sich die Möglichkeit an das Verhalten eines FIR-Filters als MATLAB-Quelltext zu verfassen und als MATLAB-Funktionsblock einzubinden (siehe Abbildung 15).

```
function [iValue,aSysAnswerOut]= fcn(iSignalValue,aImpAnswer, aSysAnswerIn)

    %Berechne das Ergebnis der Faltung aus Engangsskalar und Impulsantwort
    y = aImpAnswer .* iSignalValue;
    %Overlap and Add der alten Systemantwort
    y = y + aSysAnswerIn;
    %Auslesen des Ergebniswertes für diesen Schritt.
    %Dieser Wert wird im Overlap and Add Verfahren nicht mehr verändert
    iValue = y(1);
    %Erzeuge den Overlap. Shifte den Ergebnisvektor um einen Wert nach
    %links
    y(1) = 0;
    y = circshift(y,-1);
    %Ausgabe der Overlap-Modifizierten Systemantwort. Wird als aSysAnswerIn
    %im nächsten Durchlauf rückgekoppelt.
    aSysAnswerOut = y;
```

Abbildung 15: MATLAB-Quelltext für FIR-Filter Simulation
Bildquelle: eigene Darstellung

Das Ziel des Blockes ist es die Systemantwort für einen Wert des Nutzsignals durch Faltung mit der Impulsantwort zu ermitteln und diese mithilfe einer Addition auf die verschobene vorherige Systemantwort zu verknüpfen.
Der MATLAB-Funktionsblock erhält drei Eingangssignale: iSignalValue, ein Skalarwert der den jeweils nächsten Wert im Datenstrom darstellt, aImpAnswer, die Impulsantwort mit der das Nutzsignal gefaltet werden soll sowie aSysAnswerIn, die Systemantwort aus dem vorherigen Takt, auf die das Ergebnis des aktuellen Taktes aufaddiert wird. Damit der Block kompilieren und arbeiten kann muss aSysAnswerIn zunächst als ein leerer Vektor initialisiert werden. Dies geschieht über den Block, der silenceMono in den Switch einspielt (siehe Abbildung 16). Der Signalgeber Signal2 schaltet den Switch nach dem ersten Takt um, so dass das Signal aSysAnswerOut in den Eingang aSysAnswerIn rückgekoppelt werden kann. Da der MATLAB Funktionsblock

keine Werte nach dem Beenden seiner Operation speichert, muss die im vorherigen Takt erzeugte Systemantwort in einem Memory-Block zwischengespeichert werden.

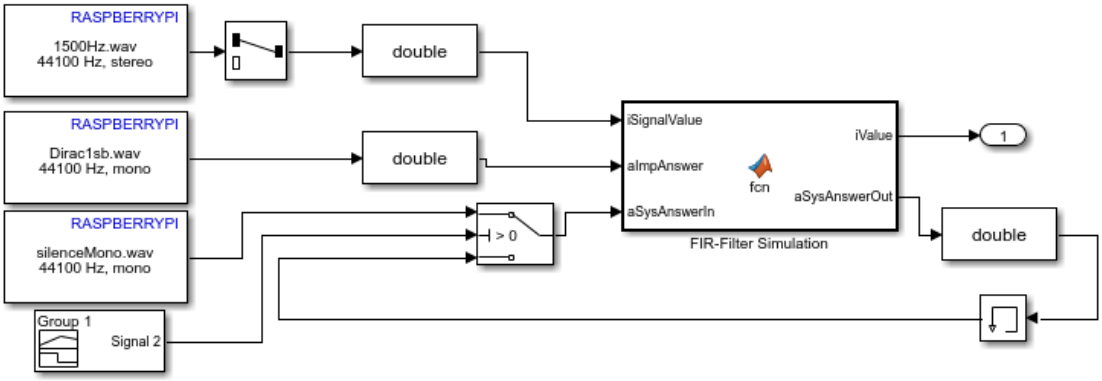


Abbildung 16: Der MATLAB-Funktionsblock im Simulink-Kontext. Testaufbau mit einem Diracstoß als Impulsantwort und einem 1.5kHz Sinuston als Nutzsignal
Bildquelle: eigene Darstellung

Der so erzeugte MATLAB-Block hat damit die selbe Funktionsweise des Faltungsblockes, welcher numerisch einen FIR-Filteraufbau simuliert. Die Anzahl der Rechenschritte ist bei beiden Implementierungen gleich (N Multiplikationen sowie N-1 Additionen mit N = Anzahl der Werte in einem Rahmen der Impulsantwort). Der MATLAB-Block hat jedoch den Vorteil einen Strom von einzelnen Datenwerten falten zu können ohne das zusätzliche Puffer oder Funktionalitäten, die einzelne Faltungsergebnisse verschieben und aufaddieren, implementiert werden müssen. Der Nachteil des Blockes ist, dass dieser eine Faltung für den speziellen Fall durchführt, dass das Nutzsignal in einzelnen Abtastwerten eingelesen wird.
Bei einem Testlauf mit diesem Block hat sich jedoch erneut der simulierende Rechner vollständig aufgehangen. Eine Überprüfung zur Laufzeit ergab, dass MATLAB während der Ausführung der Simulation den Arbeitsspeicher des simulierenden Rechners von 4 GB sehr schnell füllt. Das führt dazu, dass das Betriebssystem nicht in der Lage ist, Speicher zu prüfen und möglicherweise nicht mehr verwendete Speicherblöcke freizugeben und das System „verhungert“.
Es scheint allerdings, dass es sich dabei weniger um ein Problem von MATLAB, sondern vom Windows-Betriebssystemen und dessen Routinen zur Speicherverwaltung handelt.

„It has more to do with Windows and how its memory-management routines work (or not) regarding what memory that is marked as unused by the application is actually physically released and when.“ (Es hat mehr mit Windows zu tun und wie dessen Speicherverwaltungsroutinen im Bezug auf vom Programm als unbenutzt markierte Speicherblöcke und deren physische Freigabe funktionieren (oder auch nicht)) [MATA].

4.5 Ergebnisse

Bei Simulationen mit einer kürzeren Impulsantwort von 4 860 Samples (etwa 0,11 s) ist das Ergebnis des MATLAB-Funktionsblock das selbe wie das des MATLAB-eigenen Faltungsblockes mit angefügtem Block zum Zusammenfügen der Systemantworten. Die Zusammenführung geschieht basierend auf dem Prinzip des selbst geschriebenen MATLAB-Blockes. Allerdings ist das Verhalten des MATLAB-eigenen Faltungsblockes mit einer Ausführungsrate (Simulationszeit zu Laufzeit) von 38,70 bei der genannten Impulsantwort deutlich besser als der selbst geschriebene MATLAB-Funktionsblock, der eine Ausführungsrate von 51,11 besitzt.
Da die Simulation bereits auf einem leistungsstärkeren Rechner nicht in Echtzeit durchgeführt werden kann ist eine Umsetzung in Echtzeit mithilfe der RasPis nicht möglich. Weiterhin behindert die strikte Durchsetzung der Trennung von diskreten und kontinuierlichen Signalen und das Fehlen von Blöcken, mit denen kontinuierliche Signale in diskrete Signale umgewandelt werden können, die Umsetzung auf der Hardware. Auch ist die in der erfolgreichen Simulation verwendete Impulsantwort mit 0,11 s nicht geeignet um einen Halleffekt zu erzeugen, da ein Echo von maximal 0,11 Sekunden kaum wahrnehmbar ist.

5. Literaturverzeichnis

Bücher und Skripte

[MAPR] Manolakis, Dimitris G./Proakis, John G. (1996): Digital Signal Processing. Principles, Algorithms and Applications. 3. Auflage. New Jersey: Prentice Hall International, Inc.

[HAYM] Hayes, Monson H.(1999): Schaum‘s Outlines. Digital Signal Processing. McGraw-Hill Companies

[KRKA] Kroschel,Kristian/Kammeyer Karl Dirk (2002): Digitale Signalverarbeitung. Filterung und Spektralanalyse mit MATLAB-Übungen. 5. Auflage. B.G. Teubner Verlag

[BLCH] Blanchet, Gérard/Charbit, Maurice (2006): Digital Signal and Image Processing using MATLAB. London: ISTE Ltd.

[JUWE] Jung/ Weiß/ Franzen (2015): Praktikum für Nachrichtentechnik Versuch 7: Digitale Filter. TU Braunschweig
https://www.ifn.ing.tu-bs.de/fileadmin/praktikum/skripte/digitalefilter.pdf

Webseiten

[RPTV] Raspberry Pi 3 model B launches today – 64-bit quad A53 1.2 GHz BCM2837 – RasPi.TV. Online verfügbar unter https://raspi.tv/2016/raspberry-pi-3-model-b-launches-today-64-bit-quad-a53-1-2-ghz-bcm2837 [Stand 30.07.2018]

[RASP] Raspberry Pi Foundation: Raspberry Pi 3 Model b. Online verfügbar unter https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ [Stand 30.07.2018]

[LFUB] Bayrisches Landesamt für Umwelt: Etwas Physik vom Schall. Online verfügbar unter https://www.lfu.bayern.de/laerm/etwas_physik_vom_schall/index.htm [Stand 17.08.2018]

[MEDB] Halfacree,Gareth: Benchmarking the Raspberry Pi 3+. Online verfügbar unter https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-3-b-plus-44122cf3d806 [Stand 02.08.2018]

[MATA] Tyler, Bonnie et al.: Matlab doesn't release memory when variables are cleared – MATLAB Answers – MATLAB Central. Online verfügbar unter https://de.mathworks.com/matlabcentral/answers/316681-matlab-doesn-t-release-memory-when-variables-are-cleared [Stand 13.08.2018]

[WPDF] Wikimedia Commons: DF1 - Filter mit unendlicher Impulsantwort – Wikipedia. Online verfügbar unter https://de.wikipedia.org/wiki/Filter_mit_unendlicher_Impulsantwort#/media/File:DF1.png [Stand 21.08.2018]

[RASP] Wikimedia Commons: Raspberry Pi – Wikipedia. Online verfügbar unter https://de.wikipedia.org/wiki/Raspberry_Pi [Stand 22.08.2018]

Technische Dokumente und Dokumentationen

[MC38] Microchip Technology Inc.: MCP3004/3008 - 2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial Interface. 2008

[MC47] Microchip Technology Inc.: MCP4725 - 12-Bit Digital-to-Analog Converter with EEPROM Memory in SOT-23-6. 2009

[MATC] MathWorks: Convolution of two inputs – Simulink – MathWorks Deutschland. Online verfügbar unter https://www.mathworks.com/help/releases/R2018a/dsp/ref/convolution.html [Stand 03.08.2018]

[MATF] MathWorks: Discrete-time, overlap-add, FIR filter - MATLAB dfilt.fftfir - MathWorks Deutschland. Online verfügbar unter https://de.mathworks.com/help/dsp/ref/dfilt.ffftfir.html [Stand 20.08.2018]

Anhang 1:Erstellung einer neuen Programm-SD-Karte

Zur Erstellung einer neuen Programm-SD-Karte werden benötigt:

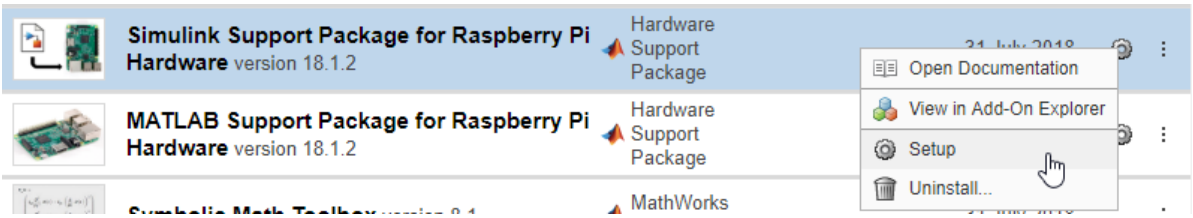
- Windows-PC mit SD-Kartenleser sowie MATLAB 2017b und MATLAB 2018a

  Zusätzlich müssen die folgenden Zusatzpakete installiert sein:

  - Simulink Coder und MATLAB Coder
  - Simulink Support Package for Raspberry Pi Hardware
  - Rpi Driver Blocks
- Micro-SD-Karte
- Router mit Internetzugang
- mindestens 2 Netzwerkkabel
- ein Raspberry Pi Ein-Platinen-System

1 in MATLAB 2017b: über den Menüpunkt Add Ons → Manage Add Ons

1.1 im Paket „Simulink Support Package for Raspberry Pi Hardware“ Rechtsklick auf das Zahnrad → „Setup“



1.2 Folgen Sie den Anweisungen der Installationsanleitung

2 Schließen sie eine Tastatur und einen Monitor an den Raspberry Pi an und verbinden sie den Raspberry Pi mit dem Internet indem sie diesen über ein Netzwerkkabel an einen mit dem Internet verbundenen Router anschließen.

3 Loggen sie sich mit dem während der Formatierung der SD-Karte festgelegten Nutzernamen und Passwort an (Standard ist Nutzername: „pi“, Passwort: „Raspberry“). Beachten Sie dabei, dass der Raspberry Pi das US-Tastaturlayout benutzt, in dem die Tastenpositionen für Z und Y vertauscht sind.

4 Laden Sie über den Befehl „git clone https://github.com/tuna-f1sh/wiringPi-mcp4725“ die für das Programm benötigten wiringPi-

Klassen auf den Raspberry Pi. Im US-Tastaturlayout befindet sich „\“ auf der Taste für „_“, „-“ auf der Taste für „ß“ und „:“ auf der Taste für „ö“.

5 Wechseln sie über „cd wiringPi-mcp4725“ in den Ordner mit den wiringPi-Klassen und installieren sie das Paket mit dem Befehl „.\build“

6 Wenn das aufzuspielende Programm ein Empfänger-Programm ist, aktivieren Sie die I2C-Schnittstelle

   6.1 Geben Sie den Befehle „sudo apt-get install python-smbus“ und „sudo apt-get install i2c-tools“ ein

   6.2 Aktivieren Sie das I2C Interface in der Raspi Config. Geben Sie dafür den Befehl „sudo raspi-config“ ein.

   6.3 Beantworten Sie die Fragen die Ihnen unter dem Menüpunkt „Interfacing Options“ → I2C gestellt werden mit Yes. Schließen sie das Menü und starten sie den Raspberry Pi über den Befehl „sudo reboot“ neu

   6.4 Prüfen Sie nach dem Neustart über den Befehl „sudo i2cdetect -y 0“ die Funktionalität des I2C Moduls. Wenn eine Tabelle angezeigt wird ist die Schnittstelle eingerichtet.

7 Wenn das Aufzuspielende Programm das Sender-Programm ist, Aktivieren Sie die SPI-Schnittstelle

   7.1 Gehen Sie über den Befehl „sudo raspi-config“ in die Konfiguration des Raspberry Pi

   7.2 Wählen Sie die SPI-Schnittstelle über die Menüpunkte „Interfacing Options“→SPI aus und beantworten Sie alle Fragen mit Yes

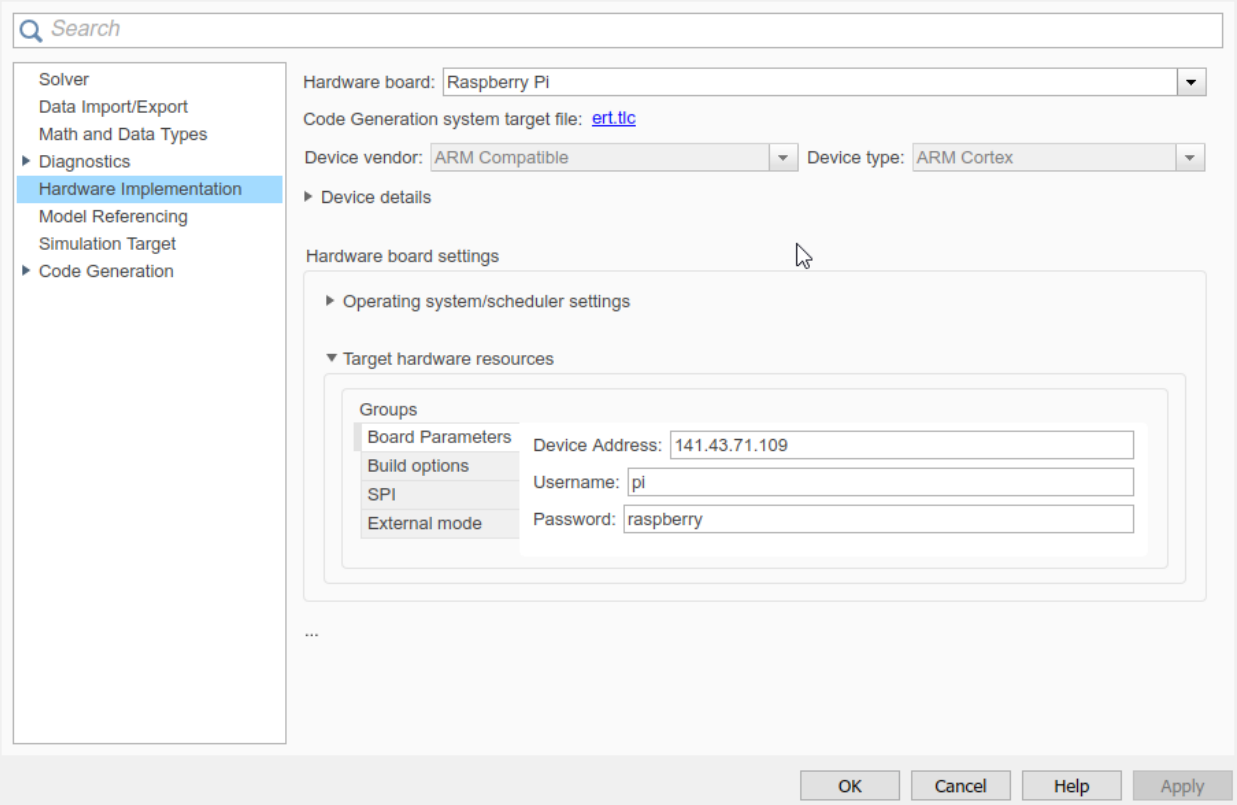
   7.3 der Raspberry Pi sollte sich neu starten. Wenn das nicht passiert, schließen Sie das Menü und benutzen den Befehl „sudo reboot“

   7.4 Prüfen Sie mit dem Befehl „ls -l /dev/spi*“ ob die Schnittstelle richtig konfiguriert wurde. Werden nach Eingabe des Kommandos /dev/spidev0.0 und/oder /dev/spidev0.1 ausgegeben ist das System richtig konfiguriert

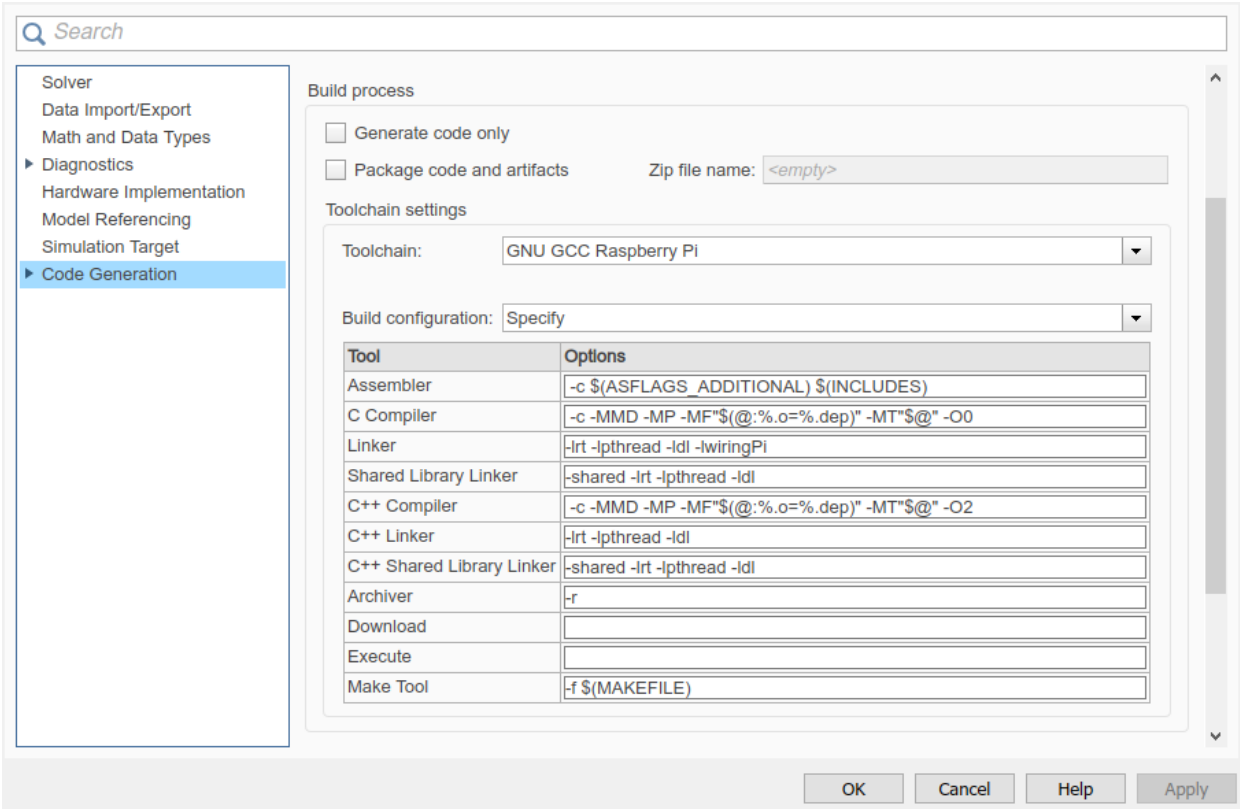
8 in MATLAB 2018a: Erstellen Sie das gewünschte Modell in Simulink

9 Öffnen sie die Konfiguration für Hardware-Deployment über „Tools“ → „Run on Target Hardware“ → „Prepare to Run“

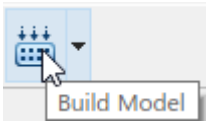
10 Stellen Sie in den Optionen das „Hardware board“ auf Raspberry Pi, achten Sie darauf, dass unter „Hardware board settings“ → „Target hardware resources“ → „Board Parameters“ → „Device Address“ die IP des Raspberry Pis eingestellt ist. Sollten entweder Kopfhörer oder Boxen an den Audioausgang des Raspberry Pi angeschlossen sein spricht dieser seine IP bei Abschluss des Bootprozesses. Setzen Sie außerdem unter „Build options“ ein Häkchen bei der Option „Run on boot“.



11 Stellen Sie im Menü-Unterpunkt „Code Generation“ unter „Build Process“ → „Toolchain Settings“ die Option „Build configuration“ auf „Specify“ und fügen sie der Zeile „Linker“ die Option „-lwiringPi“ hinzu.



12 Achten Sie darauf das sich der Raspberry Pi und der PC auf dem Sie das Simulink-Programm schreiben im selben Netzwerk befinden. Klicken Sie den „Build Model“-Knopf um das Programm auf den Raspberry Pi zu überspielen



Anhang 2: Aufbau der Blackbox

Zum Aufbau des Black Box-Experiments wird benötigt

- zwei Raspberry Pi Ein-Platinen-Systeme
- zwei Netzwerkkabel
- ein Netzwerk-Switch
- ein MCP3008 DA-Wandler
- ein MCP4725 AD-Wandler (inkl. 6-Pin Stiftleiste)
- 9 Jumperkabel Female/Male (14 bei irreversiblen Verbindungen)
- 5 Jumperkabel Female/Female (für reversible Verbindungen)
- Lötkolben und Lötzinn
- Steckplatine (für reversible Verbindungen) oder Lochrasterplatine (für irreversible Verbindungen)
- HDMI-fähiger Monitor mit HDMI Kabel
- USB-Tastatur
- Programm Micro-SD-Karten – für die Erstellung einer solchen SD-Karte folgen Sie bitte den Anweisungen des Handbuchs „Erstellung einer neuen Programm-SD-Karte“

Teilschritt 1: Vorbereitung des MCP4725

Der MCP 4725 wird mit einer 6-Pin Stiftleiste geliefert. Wird eine reversible Verbindung gewollt, muss diese an den MCP4725 angelötet werden. Wird eine irreversible Verbindung gewollt werden die Jumperkabel mit dem Male-Ende an die einzelnen Kontakte des Chips gelötet. Lassen sie in diesem Fall den Kontakt, der mit VOUT gekennzeichnet ist frei.

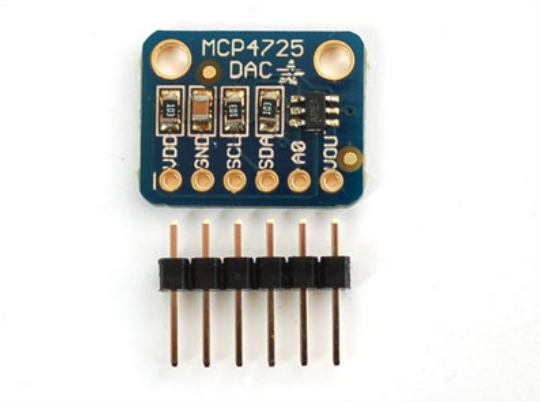


Abbildung 17: MCP 4725 inkl. 6-Pin Stiftleiste

Teilschritt 2: Verbinden des MCP4725 mit dem Raspberry Pi

Wenn eine reversible Verbindung aufgebaut werden soll, stecken Sie die 5 Female/Female Jumperkabel auf die mit VDD, GND, SCL, SDA und A0 gekennzeichneten Pins. Die Kabel werden anschließend wie folgt mit dem Raspberry Pi verbunden (Sie finden in Abbildung 19 eine Steckerkarte des Raspberry 3b):

- VDD mit +3,3 V
- GND mit GND (graue Stecker mit Erdungssymbol auf der Steckerkarte)
- SCL mit I2C1_SCL (GPIO 3)
- SDA mit I2C1_SDA (GPIO 2)

Das mit A0 gekennzeichnete Kabel führt später das Ausgangssignal der Black Box. Der mit VOUT gekennzeichnete Kontakt kann verwendet werden um zu steuern aus welchem Interface der MCP4725 die Daten vom Raspberry Pi bezieht.

Teilschritt 3: Vorbereiten des MCP 3008

Wenn eine irreversible Verbindung gewünscht ist, Löten Sie den MCP3008 auf eine Lochrasterplatine und verbinden Sie Male/Female Jumperkabel mit allen Beinen auf der rechten Seite des Chips über eine Lötverbindung (Achten Sie dabei auf die Position der Mulde auf dem Chip. Diese sollte sich oben befinden). Verbinden Sie ein Weiteres Jumperkabel mit dem „Bein“ welches in Abbildung 18 mit CH0 gekennzeichnet ist . Bei einer reversiblen Verbindung pressen Sie den MCP vorsichtig in die Löcher, so dass sich die „Beine“ auf beiden Seiten des Chips auf verschiedenen Seiten der Mulde in der Mitte der Steckplatine befinden. Achten Sie dabei ebenfalls auf die Mulde auf dem Chip.

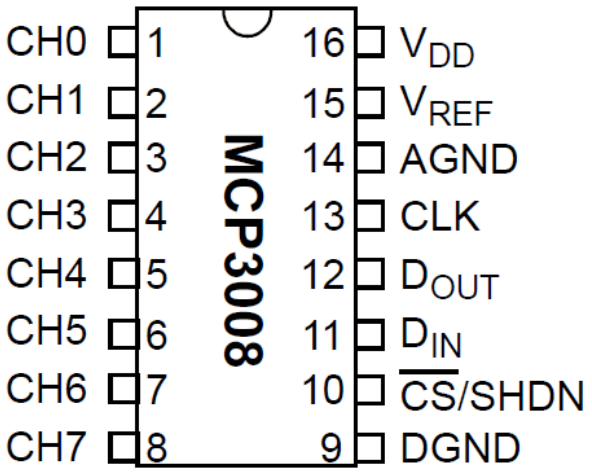


Abbildung 18: MCP3008

Verbinden Sie nun die Kabel wie folgt mit dem Raspberry Pi:

- VDD mit +3,3 V
- VREF mit +3,3V
- AGND mit GND (graue Stecker mit Erdungssymbol auf der Steckerkarte)
- CLK mit GPIO 11 (SPIO_SCLK)
- DOUT mit GPIO 9 (SPIO_MISO)
- DIN mit GPIO 10 (SPIO_MOSI)
- CS/SHDN mit GPIO 8 (CE0)
- DGND mit GND
- CH0 mit dem Eingangssignal

Teilschritt 4: Verbinden der Raspberry Pis in einem Netzwerk

Schließen Sie an beiden Raspberry Pi-Systemen jeweils ein Netzwerkkabel an. Verbinden Sie dieses Kabel mit dem Netzwerk-Switch und verbinden Sie alle drei Geräte mit dem Stromnetz. Das System ist damit einsatzbereit.

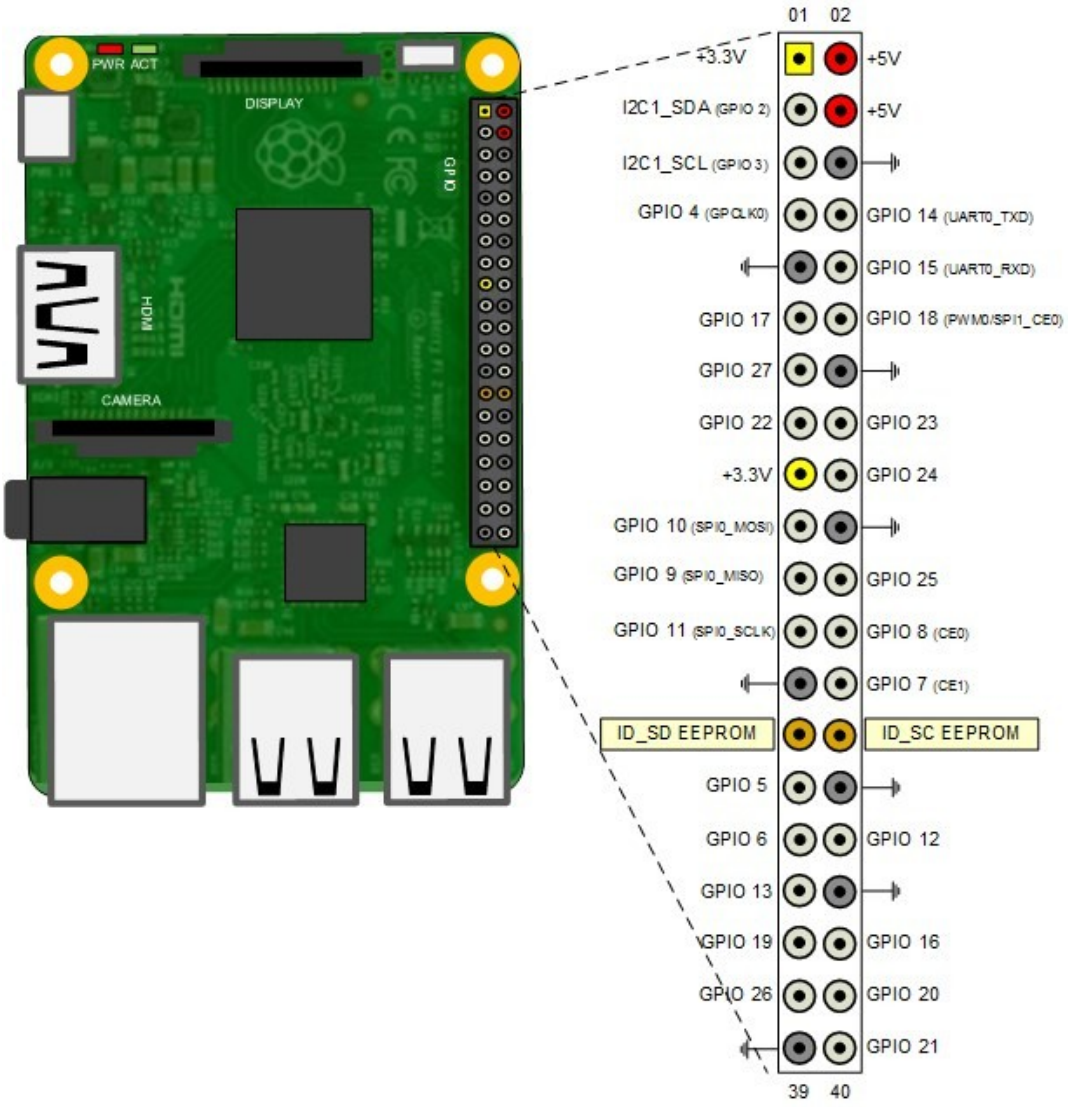


Abbildung 19: Pin-Karte des Raspberry Pi 3b