

ELEKTRONIKER FÜR GERÄTE UND SYSTEME
BETRIEBLICHER AUFTRAG 2015

DDS-Signalgenerator



HENDRIK LÜTH
PRÜFLINGSNUMMER: 20050

SCHLESWIGER STR. 21
24392 SÜDERBRARUP

Ausbildungsbetrieb:

Ausbildungswerkstatt der Marine
Mürwiker Str. 203
24944 Flensburg
Tel.Nr: 046131355080

Projektbetreuer:

Bastian Kaul
Tel.Nr: 046131355081

Inhaltsverzeichnis

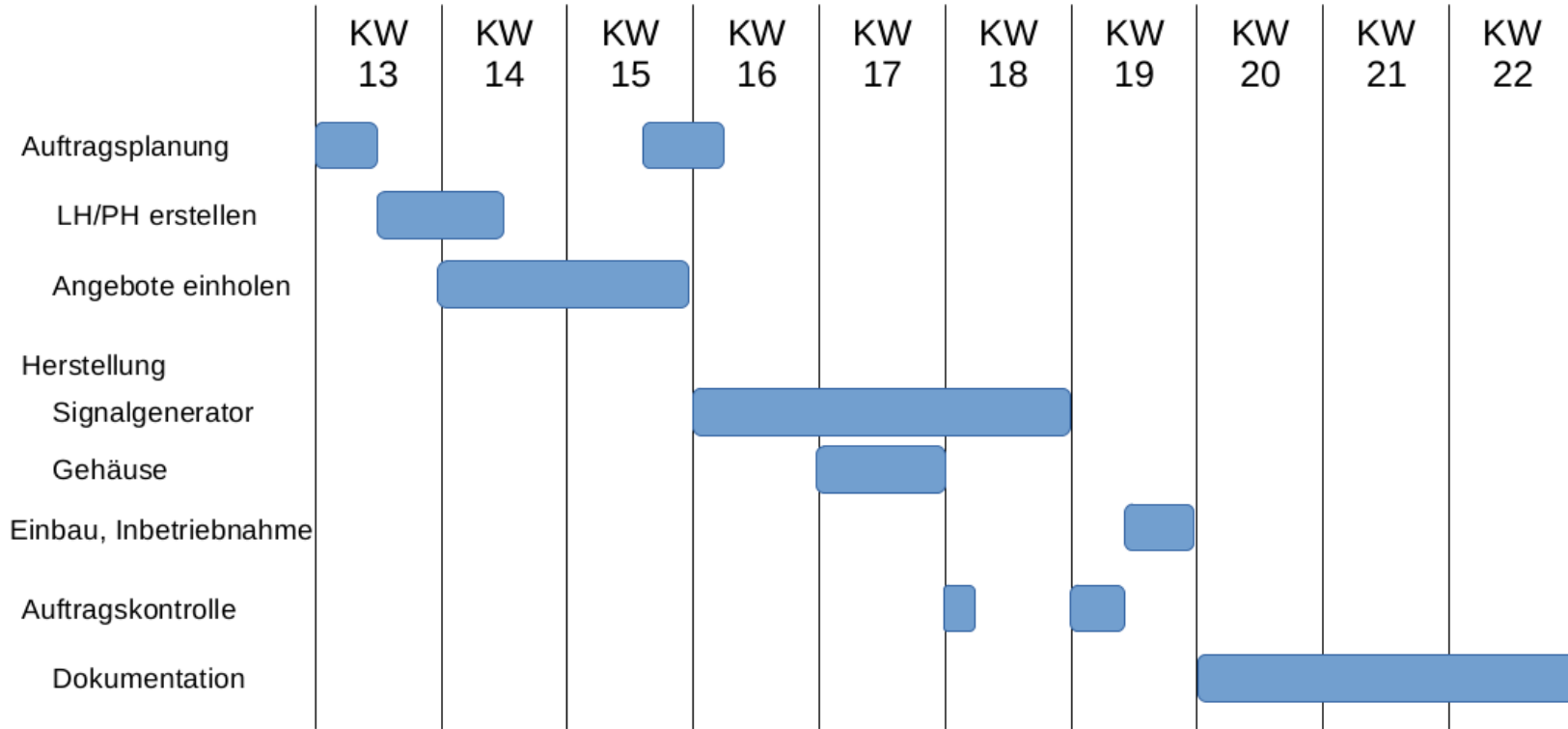
1	Arbeitsablaufplan	5
1.1	Zeitlicher Verlauf der Arbeiten	6
2	Lastenheft betriebl. Auftrag	7
2.1	Zielbestimmungen	7
2.2	Funktionale Anforderungen	7
2.2.1	Gehäuse	7
2.2.2	Platine & Layout	7
2.2.3	Firmware	7
3	Pflichtenheft betriebl. Auftrag	8
4	Datenblatt Signalgenerator	10
4.1	Features	10
4.2	Eigenschaften	10
4.3	Absolute Maximum Ratings	10
5	Lastenheft zur Herstellung des Signalgenerators	11
5.1	Zielbestimmung	11
5.2	Funktionale Anforderungen	11
5.2.1	Platine & Layout	11
5.2.2	Beschaffen der Bauteile	11
5.2.3	Bestücken der Platine	11
5.2.4	Firmware des Mikrocontrollers	11
5.3	Anhang	12
5.3.1	Prüfungsvorgabe für die Platinen	12
5.3.2	Gehäuseplan	13
6	Übergabeprotokoll, Teilauftrag Signalgenerator	14
7	Lastenheft zur Herstellung eines Gehäuses	15
7.1	Auftrag	15
7.2	Gehäuseplan	15
7.3	Änderungen am Gehäuse	16
8	Übergabeprotokoll, Teilauftrag Gehäuse	17
9	Kostenübersicht	18
10	Inbetriebnahme-Protokoll für den Signalgenerator	21
10.1	Funktionskontrolle	21
11	Übergabeprotokoll	23
12	Anhang	24
12.1	Kommunikationsprotokoll	25
12.1.1	Allgemeines	25
12.1.2	Aufbau einer Kommunikationseinheit	25

12.1.3	Aufbau einzelner Befehle	25
12.1.4	Berechnung der Registerwerte	28
12.1.5	Fehlercodes	30
12.2	Programmablaufplan der Firmware	31
12.3	Firmware des Mikrocontrollers	33
12.4	Schaltplan	40
12.5	Layout	41
12.6	Bestückungsplan	42
12.7	Bauteilliste	43

1 Arbeitsablaufplan

Pos.	Arbeitsschritt	Soll	Ist
1	Informationsphase	2,0h	2,0h
2	Planungsphase	1,0h	1,0h
3	Anfertigung eines Arbeitsablaufplans	1,0h	1,0h
4	Schreiben eines Lastenheftes für die Herstellung des Signalgenerators	2,5h	2,5h
5	Anfertigung eines Inbetriebnahmeprotokolls für den Signalgenerator	2,5h	3,0h
6	Schreiben eines Lastenheftes für die Herstellung des Gehäuses	2,0h	1,0h
7	Anfertigung eines Übergabeprotokolls für das Gehäuse und Signalgenerator	1,0h	1,0h
8	Ausfüllen des Inbetriebnahmeprotokolls für den Signalgenerator	1,0h	1,0h
9	Ausfüllen des Übergabeprotokolls des Gehäuses	0,5h	0,5h
10	Montage des Signalgenerators in das Gehäuse	0,5h	1,0h
11	Aufstellen einer Kostenübersicht des Projektes	2,0h	2,5h
12	Kontrollphase	4,0h	4,0h
		20,0h	20,5h

1.1 Zeitlicher Verlauf der Arbeiten



2 Lastenheft des betrieblichen Auftrages

2.1 Zielbestimmungen

Der Auszubildende soll für den Ausbildungsbetrieb einen DDS-basierten Signalgenerator herstellen, welcher über eine PC-Software unter Linux und Windows gesteuert werden kann. Die Software für Linux-Systeme ist in C++ geschrieben, die Software für Windows-Systeme in C#. Hierzu soll der Azubi anhand des vorgegebenen Schaltplanes den Signalgenerator herstellen und in ein Gehäuse montieren. Des Weiteren soll der Azubi die Firmware für den verbauten Mikrocontroller schreiben.

Elektronische Eigenschaften des Signalgenerators:

- Signalform: Sinus, Dreieck, Rechteck
- Frequenzbereich: 1Hz bis 12MHz
- maximale Ausgangsspannung: $12V_{ss}$
- Offset-Spannung: $\pm 6V$
- USB-Anschluss: Mikro-USB
- Signalausgang: SMA-Reverse

2.2 Funktionale Anforderungen

2.2.1 Gehäuse

Für den Signalgenerator soll ein möglichst kleines Gehäuse gewählt werden, nach Möglichkeit im „Hosentaschen-Format“. Das Material des Gehäuses soll schwarzer Kunststoff sein.

2.2.2 Platine & Layout

Die Platine ist den Innenmaßen des Gehäuses anzupassen. Die USB-Buchse und die SMA-Reverse-Buchse sollen in der Mitte der kürzeren Seite der Platine positioniert werden. Die Platine ist zu bestücken, die Funktion und Einhaltung der elektronischen Eigenschaften und Anforderungen ist zu überprüfen und zu dokumentieren.

2.2.3 Firmware

Für den Mikrocontroller ist eine Firmware zu schreiben, welche es dem Signalgenerator ermöglicht über USB mit dem Computer zu kommunizieren. Hierzu soll das LUFA-Framework¹ verwendet werden. Der Quellcode hierzu ist auf GitHub² einsehbar und kann auch von dort heruntergeladen werden. Die Kommunikation des Signalgenerators mit dem Computer ist im Kommunikationsprotokoll dokumentiert, welches im Anhang zu finden ist.

¹<http://www.fourwalledcubicle.com/files/LUFA/Doc/120730/html/index.html>

²<https://github.com/abcminiuser/lufa>

3 Pflichtenheft des betrieblichen Auftrages

Es soll ein Signalgenerator nach dem vorgegebenen Schaltplan hergestellt werden. Die PC-Software wird vom Auftraggeber bereitgestellt. Die Aufträge für die Herstellung des Gehäuses und des Signalgenerators werden an externe Auftragsnehmer vergeben. Als Gehäuse soll das Gehäuse „GEH KS 21“ von Reichelt Elektronik genutzt werden. Die Kommunikation zwischen Computer und Signalgenerator ist im Kommunikationsprotokoll im Anhang spezifiziert. Dieses wird an den Auftragsnehmer zur Herstellung des Signalgenerators zum programmieren einer Firmware weitergegeben. Der Auftragsnehmer wird folgende Tätigkeiten ausführen:

- Auswahl der Auftragsnehmer und schreiben der Lastenhefte der Teilaufträge
- Anfertigung eines Datenblattes
- Erstellen einer Kostenübersicht
- Erstellen eines Inbetriebnahme-Protokolls
- Erstellen eines Arbeitsablaufplanes
- Funktionsprüfung des Signalgenerators
- Erstellung einer Zeichnung zur Darstellung der Gehäuseänderungen
- Erstellen eines Übergabeprotokolls
- Einbau des Signalgenerators in das Gehäuse

Mit der Herstellung des Signalgenerators wird Werkstatt 2 beauftragt. Folgende Anforderungen werden gestellt werden:

- Zeichnen eines Layoutes auf Grundlage des Schaltplanes
- Herstellen einer Platine, welche in das oben genannte Gehäuse passt
- Beschaffen der verwendeten Bauteile
- Bestücken der Platine
- Testen der Platine anhand der mit gereichten Messvorgaben
- Programmieren der Firmware des Mikrocontrollers

Mit der Herstellung des Gehäuses wird Werkstatt 1 beauftragt. Folgende Anforderungen werden gestellt werden:

- Beschaffen des Gehäuses
- Herstellung des Gehäuses nach den vorgegebenen Zeichnungen.

Für den Auftrag wird ein Stundenkontingent von 20 Stunden veranschlagt. Sollte der Schaltplan fehlerhaft sein, so sind diese Fehler, nach Möglichkeit, zu korrigieren. Sollten Fehler in der mitgelieferten Software sein, so sind diese im Übergabeprotokoll zu vermerken.

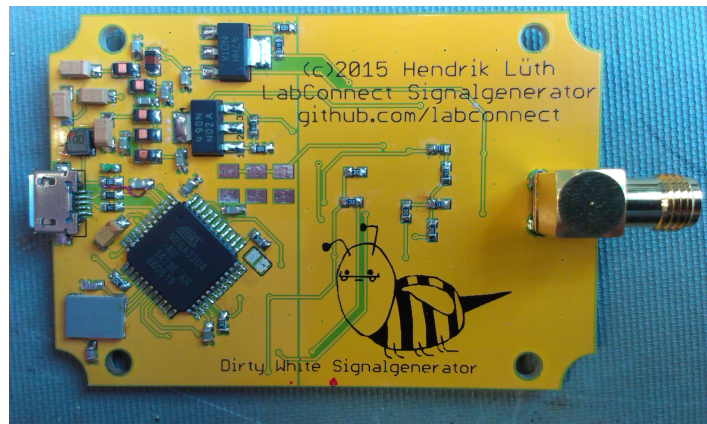
Datum/Unterschrift Hendrik Lüth

Datum/Unterschrift Auftraggeber

4 Datenblatt des DDS-Signalgenerators

4.1 Features

Der Signalgenerator basiert auf dem Prinzip der Direkten Digitalen Synthese (DDS) und ist in der Lage, Ausgangs ein breites Spektrum an Frequenzen, Signalformen und Ausgangsspannungen zu erzeugen. Die Steuerung des Gerätes erfolgt ausschließlich über den Computer, ein autarker Betrieb ist möglich. Im Gerät können hardware-spezifische Kalibrierungswerte gespeichert werden.



4.2 Eigenschaften

Parameter	Min	Typ	Max	Einheit	Testbedingungen
Betriebsspannung U_B	4,7	5	5,5	Volt	$U_B=5V$
Stromaufnahme $I_{ges.}$	30	50	100	mA	
Leistungsaufnahme P	0,14	0,25	0,55	W	

4.3 Absolute Maximum Ratings

Parameter	Max	Einheit
Betriebsspannung U_B	6	Volt
Spannung an D+ U_{D+}	3,6	Volt
Spannung an D- U_{D-}	3,6	Volt
Ausgangsstrom I_a	95	mA

5 Lastenheft zur Herstellung des Signalgenerators

5.1 Zielbestimmung

Es soll ein Signalgenerator angefertigt werden. Hierbei ist sich an den Schaltplan und die dort verzeichneten Werte zu halten. Die verwendeten Bauteile sollen, soweit möglich, SMD-Bauteile sein.

Der Umfang des Auftrages umfasst:

- das Entwerfen und Herstellen der Platine
- die Beschaffung der Bauteile
- das Bestücken der Platine
- das Programmieren einer Software für den Mikrocontroller des Signalgenerators
- das Erstellen einer Stückliste
- die mechanische und optische Prüfung der Platine

5.2 Funktionale Anforderungen

5.2.1 Platine & Layout

Da ein bestimmtes Gehäuse verwendet werden soll, darf die Platine des Signalgenerators nicht größer als die Innenbemaßung des Gehäuses sein. Eine technische Zeichnung des Gehäuses ist dem Lastenheft beigelegt.

Des Weiteren sollte das Layout so angefertigt werden, dass die Mikro-USB Buchse und die SMA-R Buchse gegenüber an den kurzen Seiten der Platine platziert werden. Die Platine soll Doublelayer sein und eine Dicke von 1,6mm haben.

5.2.2 Beschaffen der Bauteile

Die Bauteile sollen so günstig wie möglich beschafft werden.

5.2.3 Bestücken der Platine

Die Platine ist vollständig zu bestücken und optisch auf Kurzschlüsse zu prüfen.

5.2.4 Firmware des Mikrocontrollers

Die Firmware für den Mikrocontroller soll in C geschrieben werden.

Zur Ansteuerung des USB-Controllers soll das LUFA Framework verwendet werden.

Der Mikrocontroller soll in der Lage sein Daten vom Computer über USB zu empfangen und diese an die entsprechende Peripherie weiterzugeben. Für die Peripherie sollen ebenfalls die in LUFA enthaltenen Bibliotheken benutzt werden.

Für die USB-Kommunikation soll der Mikrocontroller die VID 0x1209 und die PID 0x2222 benutzen und sich als Vendorspezifisches HID Gerät anmelden. Für die Kommunikation sollen nur HID-Reports verwendet werden. Der Mikrocontroller soll in der Lage sein bestimmte Daten zurück an den PC zu geben, um Fehleranalyse und Fehlererkennung auf der PC-Seite durchführen zu können. Auch hierzu ist der Aufbau der zu übertragenden Daten in der Angelegten Protokollspezifikation zu finden.

5.3 Anhang

5.3.1 Prüfungsvorgabe für die Platinen

Optische Kontrolle

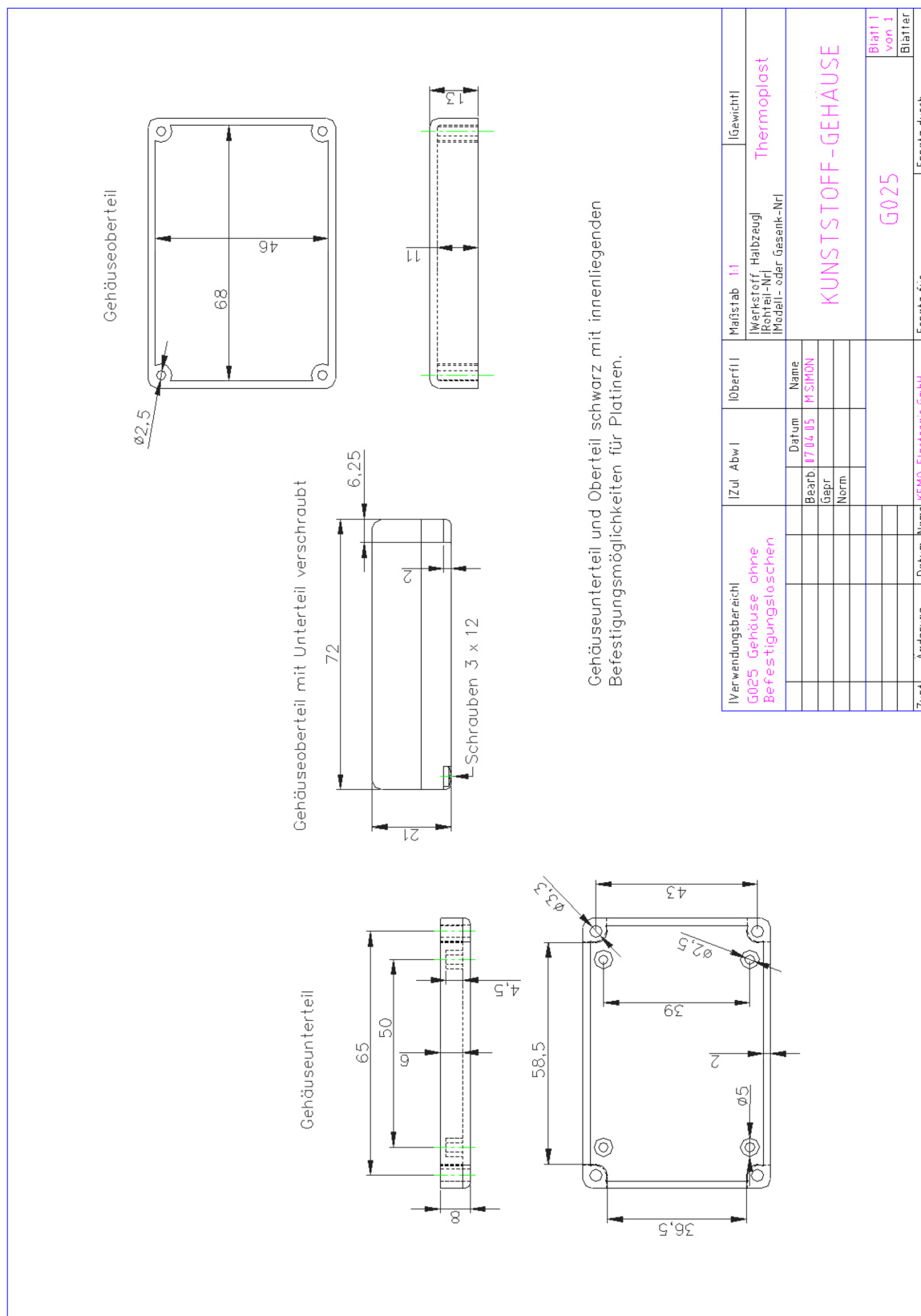
Nr.	Prüfauftrag	Ja	Nein
1	Sind alle Bauteile bestückt?		
2	Sind alle Bauteile fachgerecht gelötet?		
3	Sind IC-Beine miteinander verbunden, die nicht miteinander verbunden sein dürfen?		
4	der Fädeldraht zum aktivieren der USB-Schnittstelle des Mikrocontrollers ist eingelötet		
5	Der Lötjumper zum aktivieren des LT1615 ist gesetzt		

Elektrische Kontrolle

Die folgenden Messungen sind mit einem Digitalmultimeter durchzuführen. Sollte einer der Widerstände nicht den Anforderungen entsprechen, so darf die Platine unter keinen Umständen einer Funktionskontrolle unterzogen werden.

Nr.	Prüfauftrag	Ja	Nein	Wert
1	Messen des Widerstandes zwischen V_{cc} und GND. Ist der Wert größer als 900Ω ?			
2	Messen des Widerstandes zwischen USB_{D+} und GND. Ist der Wert größer als $500k\Omega$?			
3	Messen des Widerstandes zwischen USB_{D-} und GND. Ist der Wert größer als $500k\Omega$?			
4	Messen des Widerstandes zwischen $+12V$ und GND. Ist der Wert größer als $50k\Omega$?			
5	Messen des Widerstandes zwischen $-12V$ und GND. Ist der Wert größer als $10k\Omega$?			
6	Messen des Widerstandes zwischen $+3.3V$ und GND. Ist der Wert größer als $2k\Omega$?			
7	Messen des Widerstandes zwischen $-3.3V$ und GND. Ist der Wert größer als $2k\Omega$?			
8	Messen Sie die Stromaufnahme des Signalgenerators. Ist der Strom kleiner gleich $100mA$?			

5.3.2 Gehäuseplan



6 Übergabeprotokoll des Signalgenerators

Der Auftraggeber und der Auftragnehmer bestätigen hiermit, dass der Auftrag abgeschlossen wurde. Folgende Dinge werden mit dem Abschluss dieses Projektes übergeben:

- Der Signalgenerator
- Das Prüf- und Funktionsprotokoll des Signalgenerators
- Der Quellcode der Mikrocontroller-Firmware

Bemerkungen zu dem Projekt

- Es befindet sich ein Fehler in der PC-Software. Die eingestellten Werte werden vom Signalgenerator nicht übernommen und nicht gespeichert. Ein Fehler in der Mikrocontroller-Firmware wird ausgeschlossen.
- Im Schaltplan fehlt die Verbindung zwischen Pin 2 und Pin 7 des Mikrocontrollers. Ohne diese Verbindung hat die USB-Schnittstelle keine Funktion. Der Fehler wurde mit einem Stück Fädeldraht behoben.

Ich/Wir bestätige/n den Empfang und die Vollständigkeit und Richtigkeit aller oben aufgelisteter Dinge.

Datum/Unterschrift

Ich/Wir bestätige/n die Übergabe und die Vollständigkeit und Richtigkeit aller oben aufgelisteter Dinge.

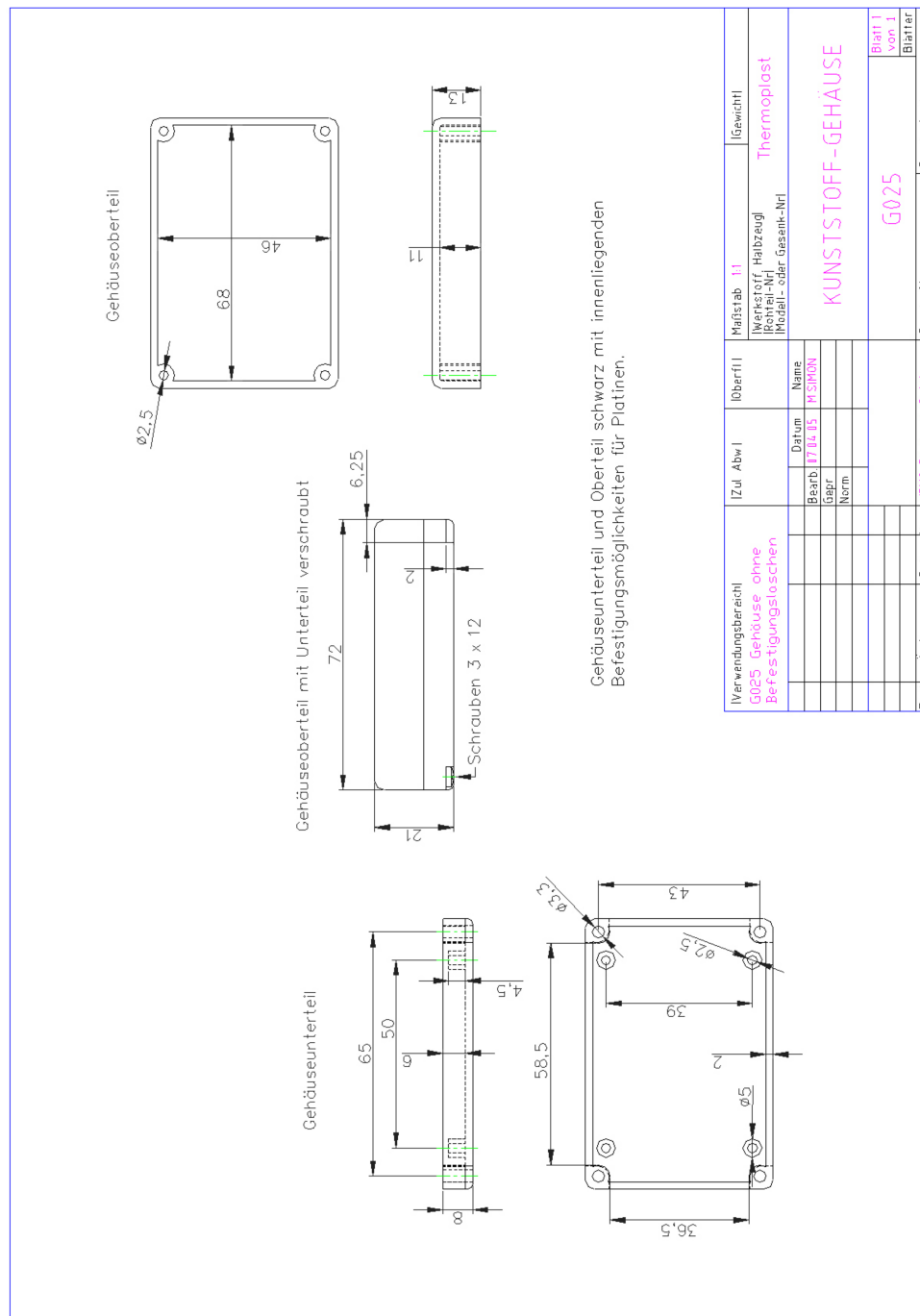
Datum/Unterschrift

7 Lastenheft zur Herstellung eines Gehäuses

7.1 Auftrag

Es ist ein Gehäuse für den Signalgenerator herzustellen. Als Gehäuse wird ein Plastikgehäuse benutzt werden, welches bei dem Lieferanten Reichelt Elektronik³ unter der Bestellnummer „GEH KS 21“ zu bestellen ist. Der Gehäuseplan liegt diesem Lastenheft bei. Ebenfalls liegt eine Zeichnung mit den nötigen Modifikationen bei.

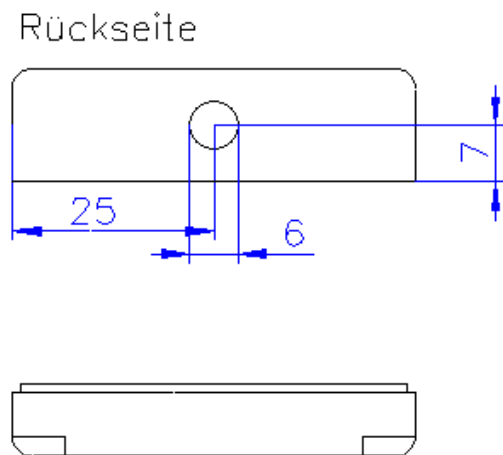
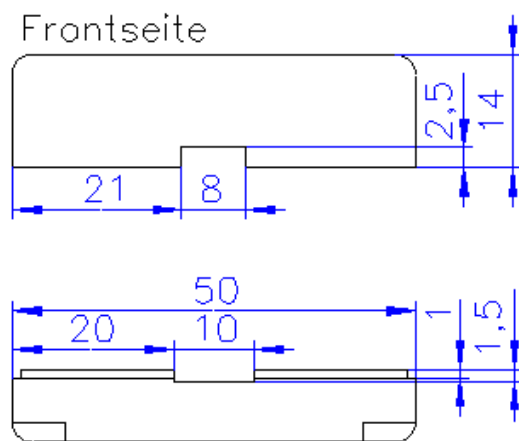
7.2 Gehäuseplan



³<https://www.reichelt.de/>

7.3 Änderungen am Gehäuse

Die Zeichnung bezieht sich auf die schmale Seite des Gehäusedeckels. Bei dem Gehäuse handelt es sich um das Gehäuse mit der Bestellnummer "GEH KS 21" von Reichelt.



8 Übergabeprotokoll des Gehäuses

Der Auftraggeber und der Auftragnehmer bestätigen hiermit, dass der Auftrag abgeschlossen wurde. Folgende Dinge werden mit dem Abschluss dieses Projektes übergeben:

- Das Gehäuse

Bemerkungen zu dem Projekt

Die Maße wurden eingehalten.

Ich/Wir bestätige/n den Empfang und die Vollständigkeit und Richtigkeit aller oben aufgelisteter Dinge.

Datum/Unterschrift

Ich/Wir bestätige/n die Übergabe und die Vollständigkeit und Richtigkeit aller oben aufgelisteter Dinge.

Datum/Unterschrift

9 Kostenübersicht

Projekt: Signalgenerator

In der folgenden Tabelle sind alle Kosten für die Herstellung des Signalgenerators zusammengestellt. In den Kostenpunkten für die Herstellung des Signalgenerators und für den Bau des Gehäuses sind neben den Materialkosten auch die Lohnkosten enthalten. Eine genaue Aufschlüsselung hierzu ist in den Angeboten auf den folgenden Seiten zu finden.

Pos.	Posten	Menge	Einzelpreis	Gesamtpreis
1	Herstellung des Signalgenerators	1	528,79€	528,79€
2	Herstellung des Gehäuses des Signalgenerators	1	15,00€	30,00€
3	Montagematerial für Montage des Signalgenerators	1	10,00€	10,00€
4	Arbeitszeit	20	15,00€	300,00€
Alle Preise inkl. 19% Gesetzl.MwSt.			MwSt.:	138,71€
			Gesamt:	868,79€

Flensburg, 04.05.2015

Werkstatt 2
Ausbildungswerkstatt Flensburg
Mürwiker Str. 203
24944 Flensburg

Hendrik Lüth
Ausbildungswerkstatt Flensburg
Mürwiker Str. 203
24944 Flensburg

Angebot für die Herstellung des Signalgenerators

Sehr geehrter Herr Lüth,

ich übersende ihnen das Angebot zur Herstellung des Signalgenerators nach Ihrem Schaltplan und Wünschen.

Pos.	Posten	Menge/Zeit	Einzelpreis	Gesamtpreis
1	Anfertigung des Platinenlayouts	10 Std.	30,00€	300,00€
2	Herstellung der Platine	1	14,84€	14,84€
3	Bauteilkosten	1	53,95€	53,95€
4	Beschaffung der Bauteile	2 Std.	20,00€	40,00€
5	Bestücken der Platine	3 Std.	35,00€	105,00€
6	Prüfen des Signalgenerators	0,5 Std.	30,00€	15,00€
			MwSt.:	100,47€
Alle Preise inkl. 19% Gesetzl.MwSt.			Gesamt:	528,79€

Bitte melden Sie sich bei uns, wenn ihnen das Angebot zusagt, wir werden dann mit der Herstellung beginnen. Das Angebot besitzt eine Gültigkeit von 2 Wochen.

mit freundlichen Grüßen,

Max Mustermann

Flensburg, 04.05.2015

Werkstatt 1
Ausbildungswerkstatt Flensburg
Mürwiker Str. 203
24944 Flensburg

Hendrik Lüth
Ausbildungswerkstatt Flensburg
Mürwiker Str. 203
24944 Flensburg

Angebot für die Herstellung eines Gehäuses

Sehr geehrter Herr Lüth,

ich übersende Ihnen das Angebot zur Herstellung zur Herstellung eines Gehäuses.
Aufgrund des niedrigen Umfangs des Projektes ist es uns möglich Ihnen eine
Pauschalpreis von 30,00€ anbieten zu können. Dieser beinhaltet Beschaffung und
Anfertigung des Gehäuses.

Pos.	Posten	Menge/Zeit	Einzelpreis	Gesamtpreis
1	Materialkosten Gehäuse	1	10,00€	10,00€
2	Herstellung des Gehäuses	0.25	20,00€	5,00€
			MwSt.:	2,34€
Alle Preise inkl. 19% Gesetzl.MwSt.			Gesamt:	15,00€

Bitte melden Sie sich bei uns, wenn Ihnen das Angebot zusagt, wir werden dann mit der
Herstellung beginnen. Das Angebot besitzt eine Gültigkeit von 2 Wochen.

mit freundlichen Grüßen,

Max Mustermann

10 Inbetriebnahme-Protokoll für den Signalgenerator

Name des Prüfers:	
Prüfdatum:	
Seriennummer:	

10.1 Funktionskontrolle

Sollte einer der Widerstände in der elektrischen Kontrolle nicht den Anforderungen entsprechen, so darf die Platine unter keinen Umständen einer Funktionskontrolle unterzogen werden.

Starten Sie das Testprogramm am Computer und stecken Sie den Signalgenerator an.

Nr.	Prüfauftrag	Ja	Nein
1	Wird der Signalgenerator vom Computer erkannt?		
2	Starten Sie den Software-Test. Ist der Test erfolgreich verlaufen?		
3	Messen Sie den Master-Clock (MCLK) an Pin 5 des AD9833. Ist der Wert im 5% Rahmen von 25MHz?		

Tragen Sie die Frequenz des Master-Clock in das entsprechende Feld für Kalibrierungswerte ein. Starten Sie die Messung Nr.1 bis Nr.4 und folgen Sie den Anweisungen in der Software. Diese enthalten auch die Einstellungen für das Oszilloskop. Messen und vergleichen Sie das Ausgangssignal mit den Vorgaben. Liegt es im Toleranzbereich? Toleranzen im Bereich von $\pm 2.5\%$ sind akzeptabel.

Nr.	Prüfauftrag	Ja	Nein
4	Messung Nr. 1: Ausgangsfrequenz: 1kHz, Ausgangsspannung: $5V_{ss}$, kein Offset, Rechteck		
5	Messung Nr. 2: Ausgangsfrequenz: 10kHz, Ausgangsspannung: $12V_{ss}$, Offset: $4V_{ss}$, Dreieck		
6	Messung Nr. 3: Ausgangsfrequenz: 1MHz, Ausgangsspannung: $8V_{ss}$, Offset: $-6V_{ss}$, Sinusform		
7	Messung Nr. 4: Ausgangsfrequenz: 8MHz, Ausgangsspannung: $2V_{ss}$, kein Offset, Sinusform		
8	Messen Sie die Ausgangsspannung an Pin 10 des AD9833. Liegt ihr Wert bei $700mV \pm 100mV$?		

Tragen Sie die Ausgangsspannung in das entsprechende Feld für Kalibrierungswerte ein.

Nr.	Prüfauftrag	Ja	Nein
9	Führen Sie den Speichertest durch. War der Test erfolgreich?		
10	Führen Sie den Lesetest durch. War der Test erfolgreich?		

Wenn alle Tests erfolgreich waren tragen Sie die Seriennummer des Gerätes in das entsprechende Feld ein und schreiben Sie die Kalibrierungswerte auf den Signalgenerator. Trennen Sie den Signalgenerator ordnungsgemäß von der Software und vom Computer.

11 Übergabeprotokoll des Signalgenerators

Der Auftraggeber und der Auftragnehmer bestätigen hiermit, dass der Auftrag abgeschlossen wurde. Folgende Dinge werden mit dem Abschluss dieses Projektes übergeben:

- Der Signalgenerator, eingebaut in ein Gehäuse
- Die Dokumentation des Signalgenerators
- Das Prüf- und Funktionsprotokoll des Signalgenerators
- Der Quellcode der Mikrocontroller-Firmware

Bemerkungen zu dem Projekt

- Es befindet sich ein Fehler in der PC-Software. Die eingestellten Werte werden vom Signalgenerator nicht übernommen und nicht gespeichert. Ein Fehler in der Mikrocontroller-Firmware wird ausgeschlossen.
- Im Schaltplan fehlt die Verbindung zwischen Pin 2 und Pin 7 des Mikrocontrollers. Ohne diese Verbindung hat die USB-Schnittstelle keine Funktion. Der Fehler wurde mit einem Stück Fädeldraht behoben.

Ich/Wir bestätige/n den Empfang und die Vollständigkeit und Richtigkeit aller oben aufgelisteter Dinge.

Datum/Unterschrift

Ich/Wir bestätige/n die Übergabe und die Vollständigkeit und Richtigkeit aller oben aufgelisteter Dinge.

Datum/Unterschrift

12 Anhang

12.1 Kommunikationsprotokoll

12.1.1 Allgemeines

In diesem Dokument wird die Datenübertragung zwischen dem Mikrocontroller des Signalgenerators und eines Computers definiert. Die Daten werden über den USB übertragen. Die USB-Spezifikationen⁴ enthalten alle nötigen Informationen, welche für Kommunikationen über den Bus nötig sind.

Der Signalgenerator wird als HID (Human Interface Device) am Computer angemeldet, wodurch keine Installation von zusätzlichen Treibern nötig ist. Die Übertragung der Daten erfolgt über HID-Reports. Zum aktuellen Zeitpunkt benutzt LabConnect für den Signalgenerator die VID 0x1209 und die PID 0x2222, welche unter Linux als GenericHID-Gerät von InterBiometrics zu finden ist. Da es sich bei der VID um eine VID handelt, welche für Open-Source-Projekte gedacht ist, ist es fraglich ob der Signalgenerator je richtig angezeigt wird. Von dem Kauf einer eigenen VID für LabConnect wird derzeit abgesehen.

12.1.2 Aufbau einer Kommunikationseinheit

Eine Kommunikationseinheit, im Folgenden als „Paket“ bezeichnet, besteht aus 13 Byte. Jedes Paket hat einen 1 Byte großen Header an seinem Anfang und einen 1 Byte großen Tail an seinem Ende. Der Header enthält die Paket-ID, an welcher sich Flussrichtung der Daten und Art der Daten erkennen lassen. Ist das fünfte Bit des Headers gleich 0, so ist die Flussrichtung der Daten vom Computer zum Mikrocontroller, ist es gleich Eins, vom Mikrocontroller zum Computer. An den unteren vier Bit lässt sich der Typ des Paketes erkennen.

In der folgenden Tabelle sind alle Befehle nach Paket-ID sortiert aufgelistet:

Paket-ID	Flussrichtung	Bezeichnung	Größe der Daten
0x00	PC → μ C	Config-Request	1 Byte
0x01	PC → μ C	Set-Command	12 Byte
0x02	PC → μ C	Data-Request	0 Byte
0x03	PC → μ C	Error/Status-Request	0 Byte
0x10	μ C → PC	Config-Response	10 Byte
0x12	μ C → PC	Data-Response	12 Byte
0x13	μ C → PC	Error/Status-Response	5 Byte

12.1.3 Aufbau einzelner Befehle

In diesem Abschnitt wird der Aufbau einzelner Befehle erläutert. Ob ein Befehl vom Computer zum Signalgenerator geht ist an der Paket-ID zu erkennen. Dies ist im Abschnitt „Aufbau einer Kommunikationseinheit“ beschrieben.

⁴http://www.usb.org/developers/docs/usb20_docs/usb_20_031815.zip

Datenübertragung vom Computer zum Signalgenerator

Config-Request

Der Config-Request steht am Anfang jeglicher Kommunikation zwischen Signalgenerator und Computer nach dem Anstecken des Signalgenerators. Der Config-Request fragt beim Signalgenerator diverse Kalibrierungsdaten wie die Frequenz des Referenztaktes oder die Boot-Daten an.

Byte	Wert	Beschreibung
0	0x00	Paket-ID
1	0x55	Prüfdaten, damit der Inhalt des Paketes nicht null ist. Der Wert ist auf 0x55 festgesetzt.

Set-Command

Mit dem Set-Command werden alle nötigen Informationen wie Frequenz, Registerwerte für die digitalen Potentiometer und Boot-Daten übergeben. Die folgende Tabelle zeigt den Aufbau:

Byte	Wert	Beschreibung
0	0x01	Paket-ID
1	*	Diese beiden Bytes enthalten die Daten für das Kontrollregister des AD9833.
2	*	
3	*	Diese vier Bytes enthalten die Daten für das Frequenzregister des AD9833. Die Berechnung dieser Werte ist im Verlauf dieses Dokumentes erklärt.
4	*	
5	*	
6	*	
7	*	In diesen beiden Bytes sind die Registerwerte des Digi-Poti für die Ausgangsspannung enthalten.
8	*	
9	*	In diesen beiden Bytes sind die Registerwerte des Digi-Poti für die Offset-Spg. enthalten.
10	*	
11	*	Multiplexer
12	*	Boot-Daten

Data-Request

Nach einem Config-Request werden die Daten ausgewertet. Sollten die Boot-Daten anzeigen, dass bereits beim Einschalten des Signalgenerators die gespeicherte Konfiguration geladen wurde, so wird ein Data-Request gesendet, um herauszufinden wie die Konfiguration ist um sie später in der graphischen Oberfläche anzuzeigen. Dieses Paket hat keine Nutzdaten.

Byte	Wert	Beschreibung
0	0x02	Paket-ID

Error/Status-Request

Ein Error/Status-Request kann zu jedem Zeitpunkt, z.B. nach einer Datenübertragung gestellt werden um den Status des Systems zu prüfen. Dieses Paket enthält keine Nutzdaten.

Byte	Wert	Beschreibung
0	0x03	Paket-ID

Datenübertragung vom Signalgenerator zum Computer

Config-Response

Byte	Wert	Beschreibung
0	0x10	Paket-ID
1	*	Seriennummer
2	*	Boot-Daten
3	*	Kalibrierungsdaten des DDS-IC Frequenz des MCLK in Hz
4	*	
5	*	
6	*	
7	*	Kalibrierungsdaten für das Digi-Poti Multiplikatoren für Berechnung
8	*	
9	*	Wert der Ausgangsspannung in mV_{ss}
10	*	

Data-Response

Dieses Paket ist die Antwort auf einen Data-Request. Es enthält die selben Daten wie ein Set-Command. Die Daten müssen vom Host dann in Frequenzen und Spannungen umgerechnet werden.

Byte	Wert	Beschreibung
0	0x01	Paket-ID
1	*	Diese beiden Bytes enthalten die Daten für das Kontrollregister des AD9833.
2	*	
3	*	Diese vier Bytes enthalten die Daten für das Frequenzregister des AD9833. Die Berechnung dieser Werte ist im Verlauf dieses Dokumentes erklärt.
4	*	
5	*	
6	*	
7	*	In diesen beiden Bytes sind die Registerwerte des Digi-Poti für die Ausgangsspannung enthalten.
8	*	
9	*	In diesen beiden Bytes sind die Registerwerte des Digi-Poti für die Offset-Spg. enthalten.
10	*	
11	*	Multiplexer
12	*	Boot-Daten

Error/Status-Response

Der Error/Status-Response enthält alle Error/Status-Meldungen die angefallen sind.

Byte	Wert	Beschreibung
0	0x13	Paket-ID
1	*	Error-Codes, bis zu fünf Stück.
2	*	
3	*	
4	*	
5	*	

12.1.4 Berechnung der Registerwerte

In dieser Sektion ist aufgelistet, wie die Registerwerte für den Signalgenerator berechnet werden. Es ist sich zwingend an die Formeln zu halten, da der Signalgenerator sonst nicht die gewünschten Ausgangssignale liefert.

Frequenz

Die Frequenzregister sind die Register, welche die Frequenz des Ausgangssignals kontrollieren. Mit einer Formel muss in Abhängigkeit vom Referenztakt und der gewünschten Frequenz des Ausgangssignals der Wert für dieses Register errechnet werden. Hier die allgemeine Formel:

$$\text{Registerwert} = F_{out} \div \frac{F_{MCL}}{2^{28}}$$

Und hier ein Beispiel für die Werte $F_{MCLK} = 25MHz$ und $F_{out} = 7,325MHz$:

$$\begin{aligned} \text{Registerwert} &= F_{out} \div \frac{F_{MCL}}{2^{28}} = 7,325MHz \div \frac{25MHz}{2^{28}} \\ \text{Registerwert} &= 78651588,61 \approx 78651589 \end{aligned}$$

In diesem Fall ist es möglich zu runden, da ein Bit nur c.a. 0,093Hz entsprechen. Nun muss der Wert noch in Binär umgerechnet werden:

$$\text{Dec} "78651589" = \text{Bin} "100\ 1011\ 0000\ 0010\ 0000\ 1100\ 0101"$$

Um den Wert in das Frequenzregister zu schreiben wird der binäre Wert in LSBs und MSBs aufgeteilt und es wird die Adressierung des Registers "01 und die fehlenden Nullen, um auf 28 Bit zu kommen, davor angehängt:

MSBs: 0101 0010 1100 0000
LSBs: 0110 0000 1100 0101

Dies ist ein Beispielcode in C++, in dem die entsprechenden Register berechnet werden:

```

1 float mclk = 25000000, register_size = 268435456;
2 float teiler = mclk / register_size;
3 int f_regwert = frequenz / teiler;
4 //block1=lsb Block4=msb
5 unsigned char block1, block2, block3, block4;
6
7 block1 = f_regwert;
8 f_regwert = f_regwert >> 8;
9 block2 = f_regwert;
10 block2 = (block2 | 0x40) & (~0x80);
11 f_regwert = f_regwert >> 6;
12 block3 = f_regwert;
13 f_regwert = f_regwert >> 8;
14 block4 = f_regwert;
15 block4 = (block4 | 0x40) & (~0x80);

```

Signalform

Für das Register der Signalform gibt es nicht viel zu berechnen, da es nur drei Signalformen gibt. Die zwei Byte, mit denen die Signalform gesteuert wird können folgende Werte annehmen:

Wert	Signalform
0x2000	Sinus
0x2002	Dreieck
0x0000	Rechteck

Es ist hierbei darauf zu achten, dass auch der Zustand des Multiplexers angepasst wird, da es ansonsten zu unerwünschten Ausgangsspannungen kommen kann.

Spitzenspannung

Die Amplitude des Ausgangssignals lässt sich über den Multiplexer und das Digitalpotentiometer einstellen. Hierzu wird die Ausgangsspannung des DDS-IC als Berechnungsgrundlage hinzugezogen. Mit dem Multiplexer kann man auswählen, ob das Signal direkt auf den Verstärker gegeben wird oder ob eine Teilung von 5:1 bzw. eine Verstärkung von ungefähr 3 stattfinden soll, bevor das Signal auf den Verstärker gegeben wird. Welchen Wert das entsprechende Byte annehmen muss ist unter SSonstige Registerim Unterabschnitt Multiplexernachzulesen. Die Registerwerte des Digitalpotentiometers werden wie folgt berechnet:

$$Registerwert_{gesamt} = \left(\frac{100k\Omega}{\frac{U_{Ausgang}}{U_{Eingang}} - 1} - 2,2k\Omega \right) \div \frac{200k\Omega}{512}$$

Hier ein Beispiel für die Werte $U_{Ausgang} = 7,5V_{ss}$ und $U_{Eingang} = 1V_{ss}$:

$$Registerwert_{gesamt} = \left(\frac{100k\Omega}{\frac{7,5V_{ss}}{1V_{ss}} - 1} - 2,2k\Omega \right) \div \frac{200k\Omega}{512}$$

$$Registerwert_{gesamt} \approx 13184,61 \div 390,625$$

$$Registerwert_{gesamt} \approx 34$$

Da der Registerwert für zwei in Reihe geschaltete Widerstände gilt, muss dieser Wert noch auf beide Register aufgeteilt werden. Sollte das Ergebnis eine ungerade Anzahl Bits annehmen so erhält eines der Register einfach ein Bit mehr. Daraus ergibt sich, dass beide Register den dezimalen Wert "17" haben bzw 0x11 in Hexadezimal.

Der folgende Beispielcode ist in C# geschrieben und berechnet den Gesamtwert beider Register.

```

1 int umax = 12000, bits = 512, register1, register2;
2 int ergebnis = u_amplitude_mv / (umax / bits);
3
4 if (510 < ergebnis)
5 {
6     ergebnis = 510;
7 }
8
9 if (ergebnis%2==0)
10 {
11     register1 = 255 - ergebnis / 2;
12     register2 = 255 - ergebnis / 2;
13 }
14 else
15 {
16     ergebnis = ergebnis - 1;
17     register1 = 255 - ((ergebnis / 2) + 1);
18     register2 = 255 - ergebnis / 2;
19 }

```

Offsetspannung

Sonstige Register

Boot-Daten

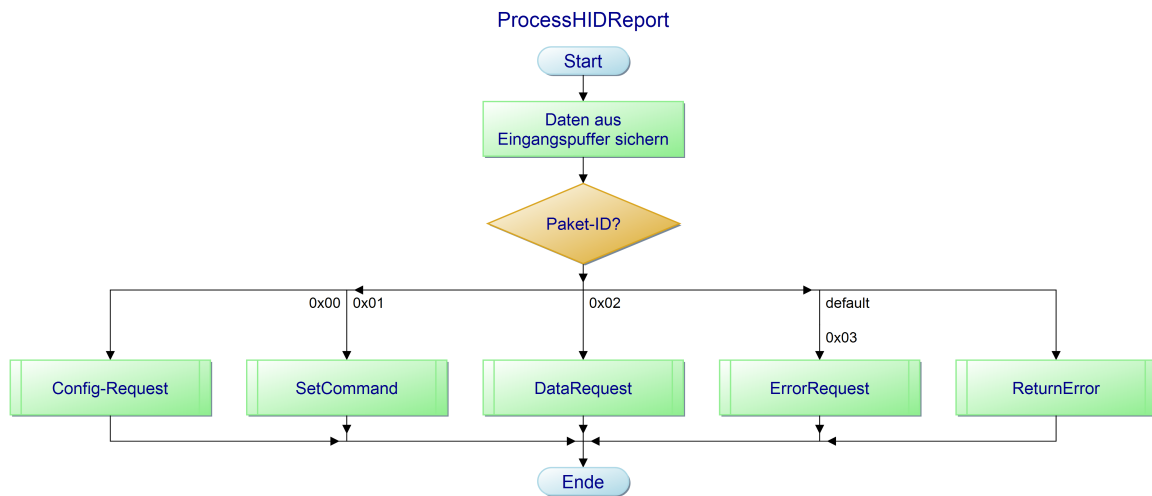
Boot-Daten	Beim Boot laden	Werte speichern
0x00	Nein	Nein
0x01	Nein	Ja
0x10	Ja	Nein
0x11	Ja	Ja

Multiplexer

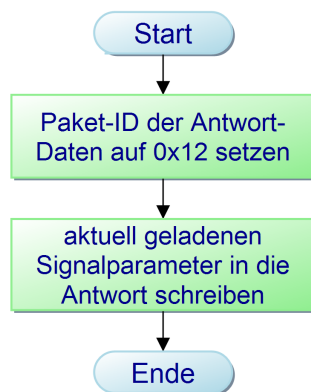
12.1.5 Fehlercodes

Fehlercode	Beschreibung des Fehlers
0x00	Kein Fehler
0x01	Keine gültige Package-ID
0x02	Transportdaten des Config-Requests sind falsch
0x03	Digitalpotentiometer ist nicht erreichbar

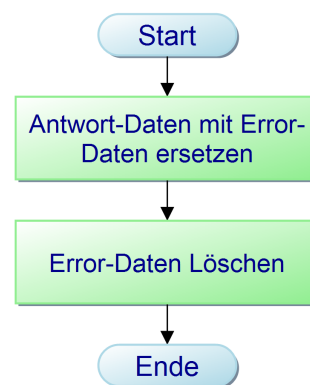
12.2 Programmablaufplan der Firmware



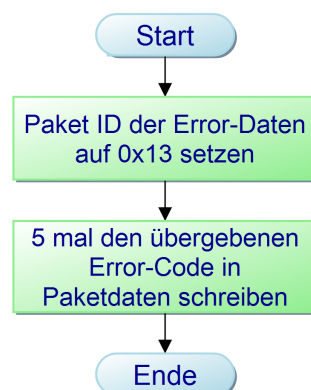
DataRequest

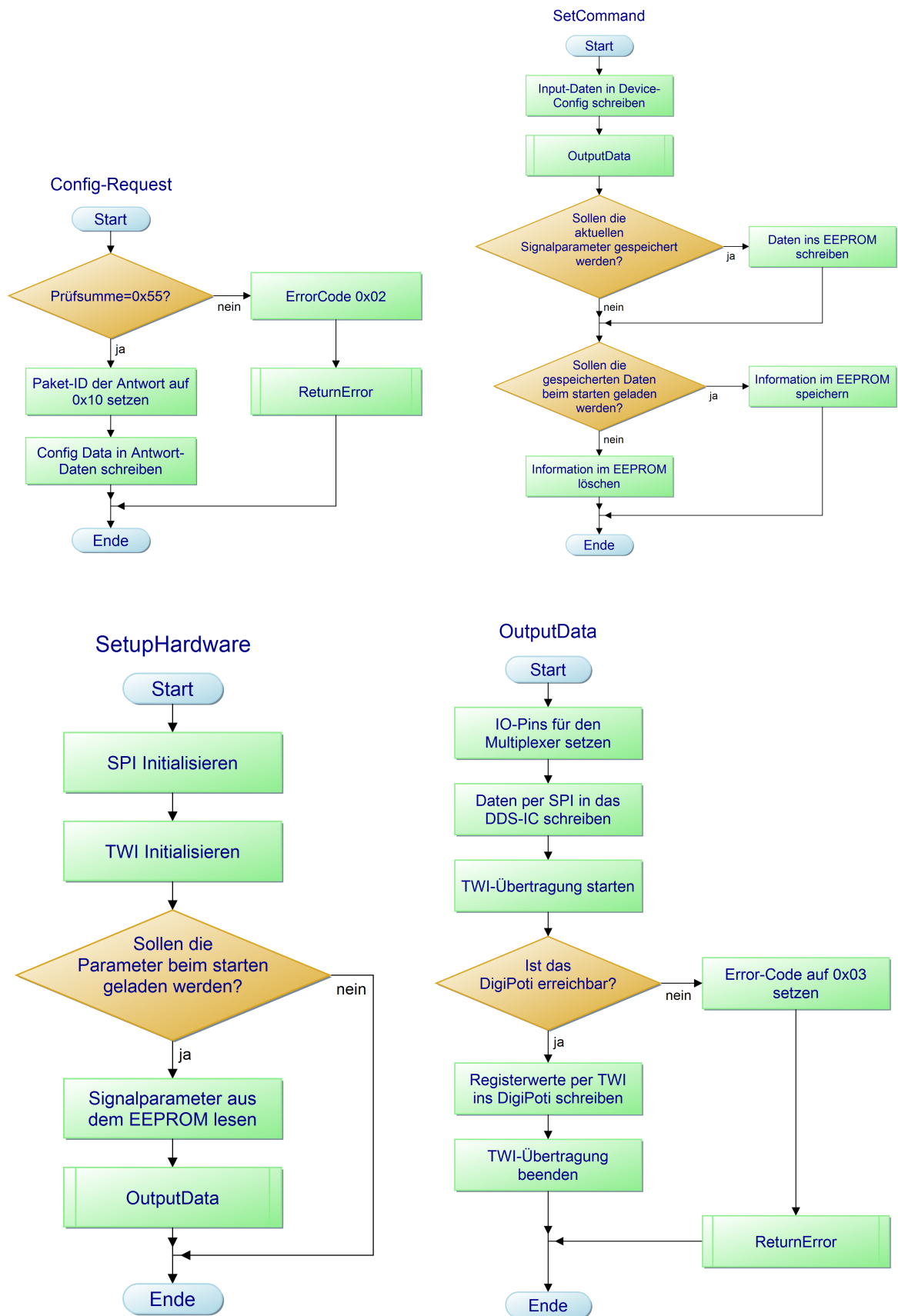


ErrorRequest



ReturnError





12.3 Firmware des Mikrocontrollers

```

1  /*
2      LUFA Library
3      Copyright (C) Dean Camera, 2014.
4
5      dean [at] fourwalledcubicle [dot] com
6      www.lufa-lib.org
7  */
8
9  /*
10     Copyright 2014 Dean Camera (dean [at] fourwalledcubicle [dot] com)
11
12     Permission to use, copy, modify, distribute, and sell this
13     software and its documentation for any purpose is hereby granted
14     without fee, provided that the above copyright notice appear in
15     all copies and that both that the copyright notice and this
16     permission notice and warranty disclaimer appear in supporting
17     documentation, and that the name of the author not be used in
18     advertising or publicity pertaining to distribution of the
19     software without specific, written prior permission.
20
21     The author disclaims all warranties with regard to this
22     software, including all implied warranties of merchantability
23     and fitness. In no event shall the author be liable for any
24     special, indirect or consequential damages or any damages
25     whatsoever resulting from loss of use, data or profits, whether
26     in an action of contract, negligence or other tortious action,
27     arising out of or in connection with the use or performance of
28     this software.
29  */
30
31 #include "Sgen-Firmware.h"
32
33 #define DigiPoti 0x12
34 #define ConfigSize 12
35 #define EEPROM_CONFIG_OFFSET 20
36 #define EEPROM_CAL_OFFSET 10
37 #define EEPROM_BOOT_VALUES 5
38
39 /** Buffer to hold the previously generated HID report, for comparison
40     purposes inside the HID class driver. */
41 static uint8_t PrevHIDReportBuffer[GENERIC_REPORT_SIZE];
42
43 //contains the Deviceconfiguration Data
44 uint8_t DeviceConfig[ConfigSize] = {0x20, 0x00, 0x40, 0x00, 0x69, 0xf1, 0
45     x00, 0x00, 0x00, 0x00, 0x18, 0x11};
46
47 uint8_t Response_Data[ConfigSize] = {};
48 uint8_t Error_Data[6] = {};
49
50 uint8_t input_data[14] = {};
51 uint8_t Config_Data[10] = {0x01, 0x01, 0x01, 0x7d, 0x78, 0x40, 0x00, 0x00,
52     0x02, 0xee};
53
54 //Indicates weather the digipot is respondig for error feedback to the
55     computer
56 bool StatusDigiPoti = false;

```

```

53
54 /** LUFA HID Class driver interface configuration and state information.
    This structure is
55  * passed to all HID Class driver functions, so that multiple instances of
    the same class
56  * within a device can be differentiated from one another.
57  */
58 USB_ClassInfo_HID_Device_t Generic_HID_Interface =
59 {
60     .Config =
61     {
62         .InterfaceNumber           = INTERFACE_ID_GenericHID,
63         .ReportINEndpoint         =
64         {
65             .Address               = GENERIC_IN_EPADDR,
66             .Size                   = GENERIC_EPSIZE,
67             .Banks                  = 1,
68         },
69         .PrevReportINBuffer        = PrevHIDReportBuffer,
70         .PrevReportINBufferSize    = sizeof(PrevHIDReportBuffer)
71     },
72 };
73
74 int main(void)
75 {
76     SetupHardware();
77
78     GlobalInterruptEnable();
79
80     for (;;)
81     {
82         HID_Device_USBTask(&Generic_HID_Interface);
83         USB_USBTask();
84     }
85 }
86
87
88 void SetupHardware(void)
89 {
90
91     /* Disable watchdog if enabled by bootloader/fuses */
92     MCUSR &= ~(1 << WDRF);
93     wdt_disable();
94
95     /* Disable clock division */
96     clock_prescale_set(clock_div_1);
97
98     /* Hardware Initialization */
99     USB_Init();
100
101     // http://avrbeginners.net/architecture/spi/spi.html
102     SPI_Init(SPI_SPEED_FCPU_DIV_32 | SPI_SCK_LEAD_FALLING |
103             SPI_SAMPLE_LEADING | SPI_ORDER_MSB_FIRST | SPI_MODE_MASTER);
104
105     // initialize TWI-Bus
106     TWI_Init(TWI_BIT_PRESCALE_1, TWI_BITLENGTH_FROM_FREQ(1, 200000));

```

```
107
108 //check wether to load data at boot and do it
109 if (eeprom_read_byte((uint8_t*)EEPROM_BOOT_VALUES) == 0x10 )
110 {
111     for (int i = 0; i < ConfigSize; i++)
112     {
113         uint8_t eeprom_addr = EEPROM_CONFIG_OFFSET + i;
114         DeviceConfig[i] = eeprom_read_byte((uint8_t*)eeprom_addr);
115     }
116
117     Output_data();
118 }
119
120 }
121
122 /** Event handler for the library USB Connection event. */
123 void EVENT_USB_Device_Connect(void)
124 {
125
126 }
127
128 /** Event handler for the library USB Disconnection event. */
129 void EVENT_USB_Device_Disconnect(void)
130 {
131
132 }
133
134 /** Event handler for the library USB Configuration Changed event. */
135 void EVENT_USB_Device_ConfigurationChanged(void)
136 {
137     bool ConfigSuccess = true;
138
139     ConfigSuccess &= HID_Device_ConfigureEndpoints(&Generic_HID_Interface);
140
141     USB_Device_EnableSOFEvents();
142
143 }
144
145 /** Event handler for the library USB Control Request reception event. */
146 void EVENT_USB_Device_ControlRequest(void)
147 {
148     HID_Device_ProcessControlRequest(&Generic_HID_Interface);
149 }
150
151 /** Event handler for the USB device Start Of Frame event. */
152 void EVENT_USB_Device_StartOfFrame(void)
153 {
154     HID_Device_MillisecondElapsed(&Generic_HID_Interface);
155 }
156
157 /** HID class driver callback function for the creation of HID reports to
158     the host.
159     * \param[in] HIDInterfaceInfo Pointer to the HID class interface
160     configuration structure being referenced
161     * \param[in,out] ReportID Report ID requested by the host if non-zero,
162     otherwise callback should set to the generated report ID
```

```

161 * \param[in]      ReportType  Type of the report to create , either
    HID_REPORT_ITEM_In or HID_REPORT_ITEM_Feature
162 * \param[out]     ReportData  Pointer to a buffer where the created report
    should be stored
163 * \param[out]     ReportSize  Number of bytes written in the report (or
    zero if no report is to be sent)
164 *
165 * \return Boolean \c true to force the sending of the report , \c false to
    let the library determine if it needs to be sent
166 */
167 bool CALLBACK_HID_Device_CreateHIDReport(USB_ClassInfo_HID_Device_t* const
    HIDInterfaceInfo ,
168                                         uint8_t* const ReportID ,
169                                         const uint8_t ReportType ,
170                                         void* ReportData ,
171                                         uint16_t* const ReportSize)
172 {
173     uint8_t* Data          = (uint8_t*)ReportData;
174
175
176     *ReportSize = GENERIC_REPORT_SIZE;
177
178     return false;
179 }
180
181 /** HID class driver callback function for the processing of HID reports
    from the host .
182 *
183 * \param[in] HIDInterfaceInfo  Pointer to the HID class interface
    configuration structure being referenced
184 * \param[in] ReportID         Report ID of the received report from the host
185 * \param[in] ReportType       The type of report that the host has sent ,
    either HID_REPORT_ITEM_Out or HID_REPORT_ITEM_Feature
186 * \param[in] ReportData      Pointer to a buffer where the received report
    has been stored
187 * \param[in] ReportSize      Size in bytes of the received HID report
188 */
189 void CALLBACK_HID_Device_ProcessHIDReport(USB_ClassInfo_HID_Device_t* const
    HIDInterfaceInfo ,
190                                         const uint8_t ReportID ,
191                                         const uint8_t ReportType ,
192                                         const void* ReportData ,
193                                         const uint16_t ReportSize)
194 {
195     uint8_t* Data          = (uint8_t*)ReportData;
196
197     for (int i=0; i<ConfigSize; i++)
198     {
199         input_data[i] = Data[i];
200     }
201
202     switch(input_data[0])
203     {
204         case 0x00: ConfigRequest(); break;
205         case 0x01: SetCommand(); break;
206         case 0x02: DataRequest(); break;
207         case 0x03: ErrorRequest(); break;
208         default: ReturnError(0x01); break;

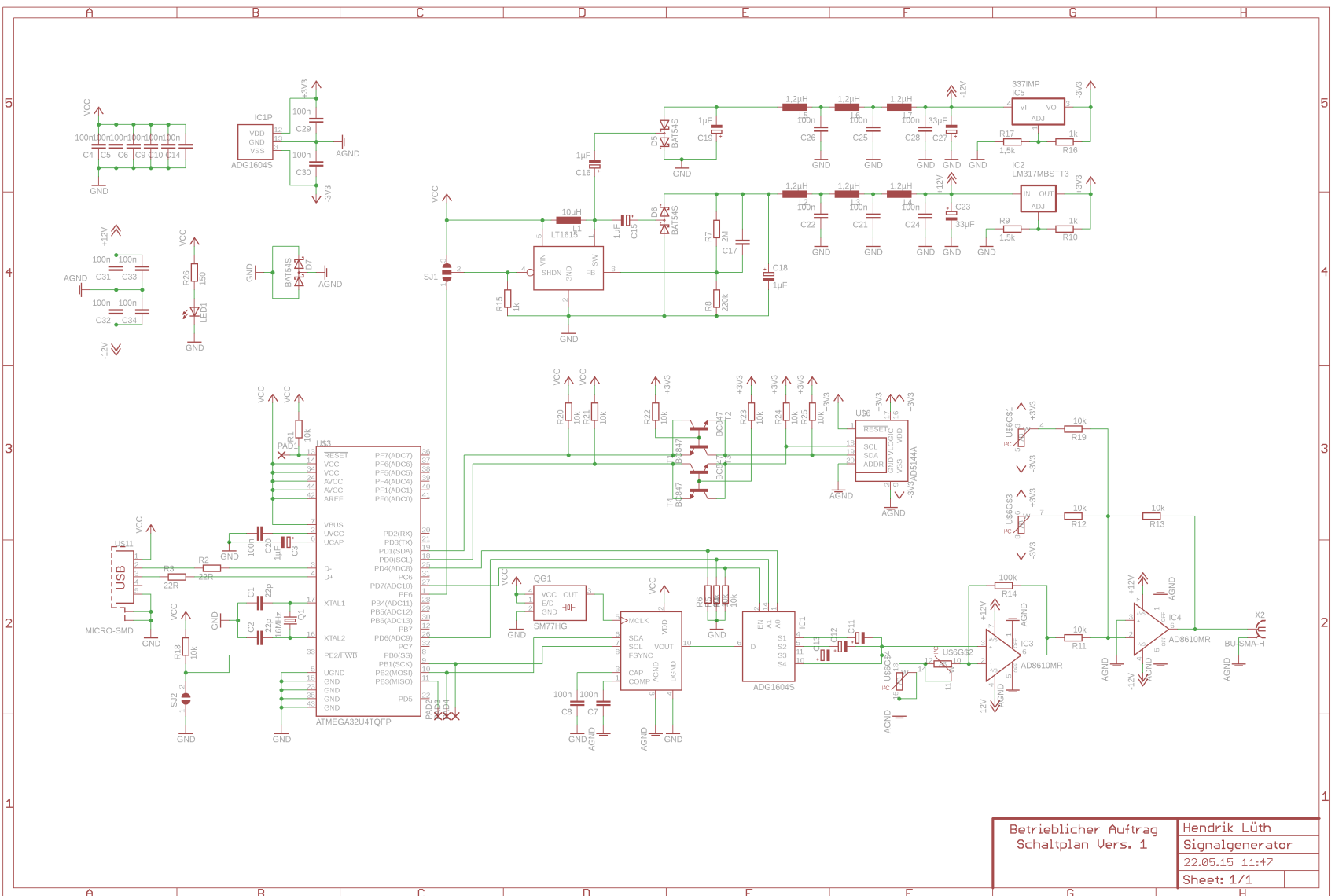
```

```
209     }
210
211
212     return;
213 }
214
215 /*****
216 here starts the sections for the functions called by the usb code
217 *****/
218
219 void ConfigRequest()
220 {
221     if (input_data[1] != 0x55)
222     {
223         ReturnError(0x02);
224         return;
225     }
226
227     Response_Data[0] = 0x10;
228
229     for (int i = 0; i < 10; i++)
230     {
231         Response_Data[i+1] = Config_Data[i];
232     }
233
234     return;
235 }
236
237 void SetCommand()
238 {
239
240     for (int i = 0; i < 13; i++)
241     {
242         DeviceConfig[i] = input_data[i+1];
243     }
244
245     Output_data();
246
247     if ((DeviceConfig[12] & 0x01) == 0x01)
248     {
249         for (int i = 0; i < 12; i++)
250         {
251             uint8_t eeprom_addr = EEPROM_CONFIG_OFFSET + i;
252             eeprom_update_byte((uint8_t*)eeprom_addr, DeviceConfig[i]);
253         }
254     }
255
256     if ((DeviceConfig[12] & 0x10) == 0x10)
257     {
258         eeprom_update_byte((uint8_t*)EEPROM_BOOT_VALUES, 0x10);
259     }
260     else
261     {
262         eeprom_update_byte((uint8_t*)EEPROM_BOOT_VALUES, 0x00);
263     }
264
265     return;
266 }
```

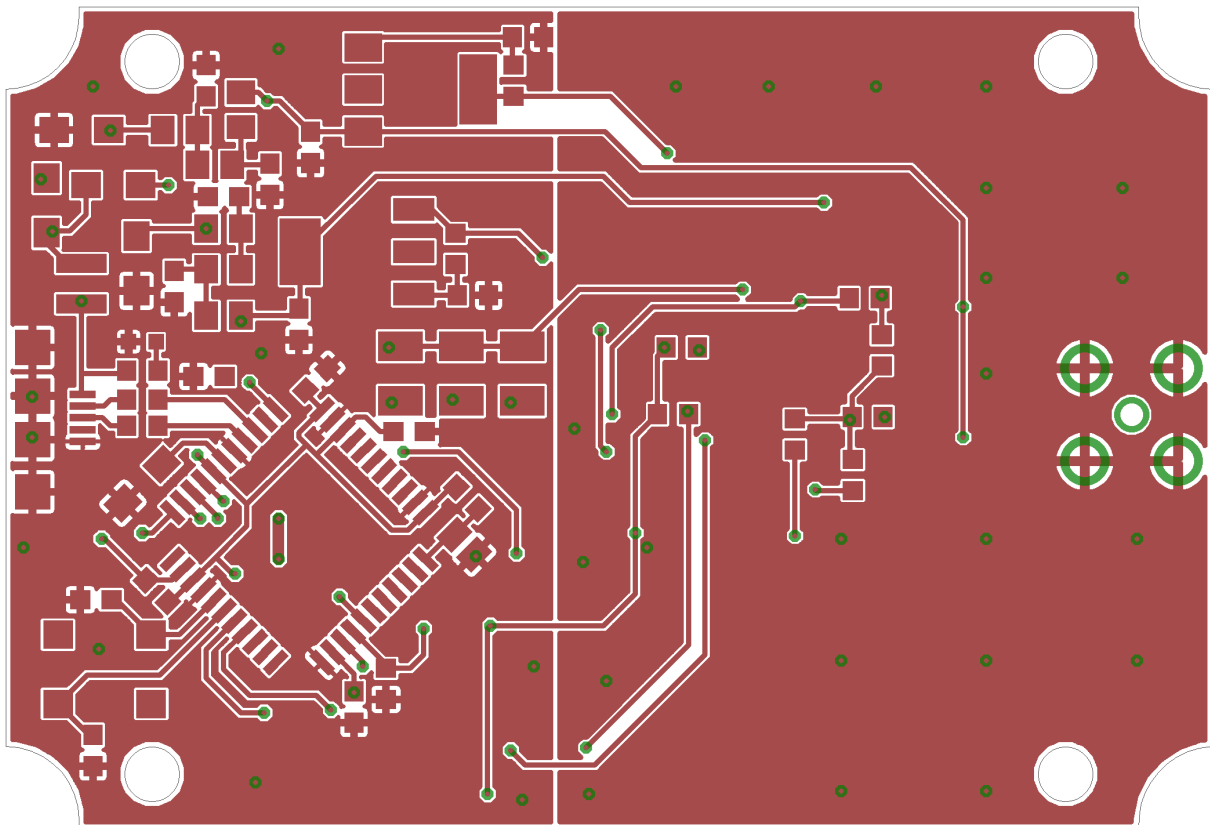
```
267
268 void DataRequest()
269 {
270     Response_Data[0] = 0x12; //send dataresponse
271
272     for (int i = 0; i < 10; i++)
273     {
274         Response_Data[i+1] = DeviceConfig[i];
275     }
276
277     return;
278 }
279
280 void ErrorRequest()
281 {
282     for (int i = 0; i < 6; i++)
283     {
284         Response_Data[i] = Error_Data[i];
285         Error_Data[i] = 0x00;
286     }
287
288     return;
289 }
290
291
292 /*****
293 Here starts the code called by the requests or commands
294 *****/
295
296 void Output_data()
297 {
298     //Set the IO-Pins for the analog Multiplexer
299     PORTD = DeviceConfig[10];
300
301     //Send the frequency and formdata to the AD9833
302     SPI_Send2Byte(DeviceConfig[0], DeviceConfig[1]);
303     SPI_Send2Byte(DeviceConfig[2], DeviceConfig[3]);
304     SPI_Send2Byte(DeviceConfig[4], DeviceConfig[5]);
305
306     //Send the TWI-Data, but only if device is responding.
307     if (TWI_StartTransmission(DigiPoti, 1) == 0)
308     {
309         for (int i = 0; i < 4; i++)
310         {
311             int PotiWert = i + 6;
312
313             TWI_SendByte(0x10 | i);
314             TWI_SendByte(DeviceConfig[PotiWert]);
315         }
316
317         TWI_SendByte(0x78);
318         TWI_SendByte(0x01);
319         TWI_StopTransmission();
320     }
321     else
322     {
323         ReturnError(0x03);
324     }
}
```

```
325     return;
326 }
327
328 void ReturnError(uint8_t ErrorType)
329 {
330     Error_Data[0] = 0x13;
331     for (int i = 1; i < 6; i++)
332     {
333         Error_Data[i] = ErrorType;
334     }
335     return;
336 }
```

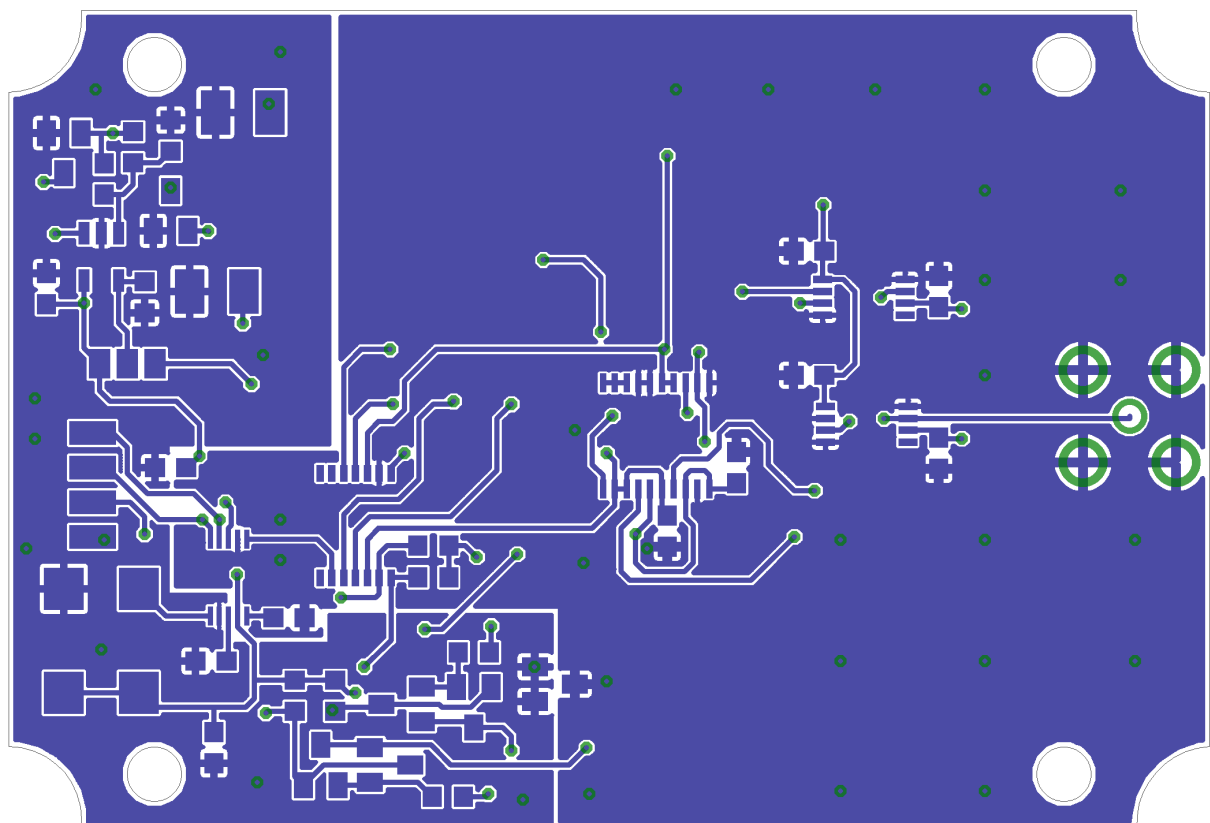
12.4 Schaltplan



12.5 Layout

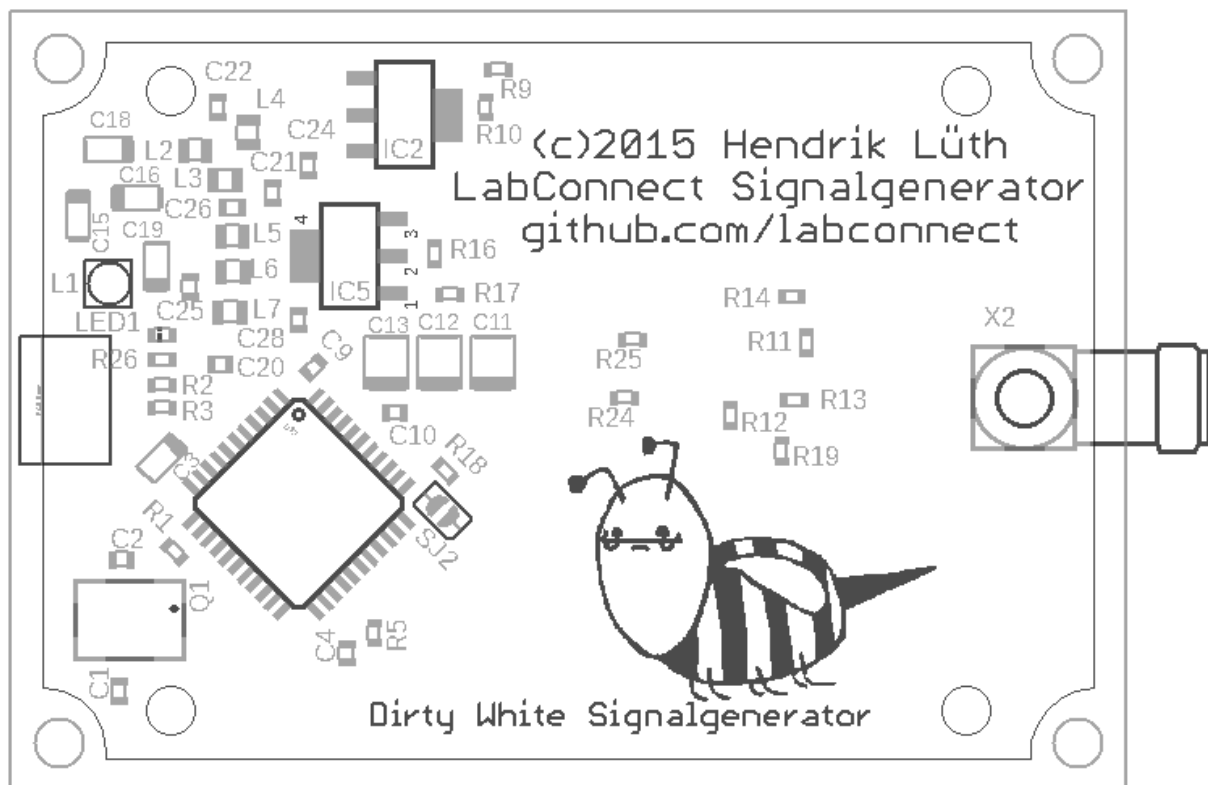


Layout, Top-Seite

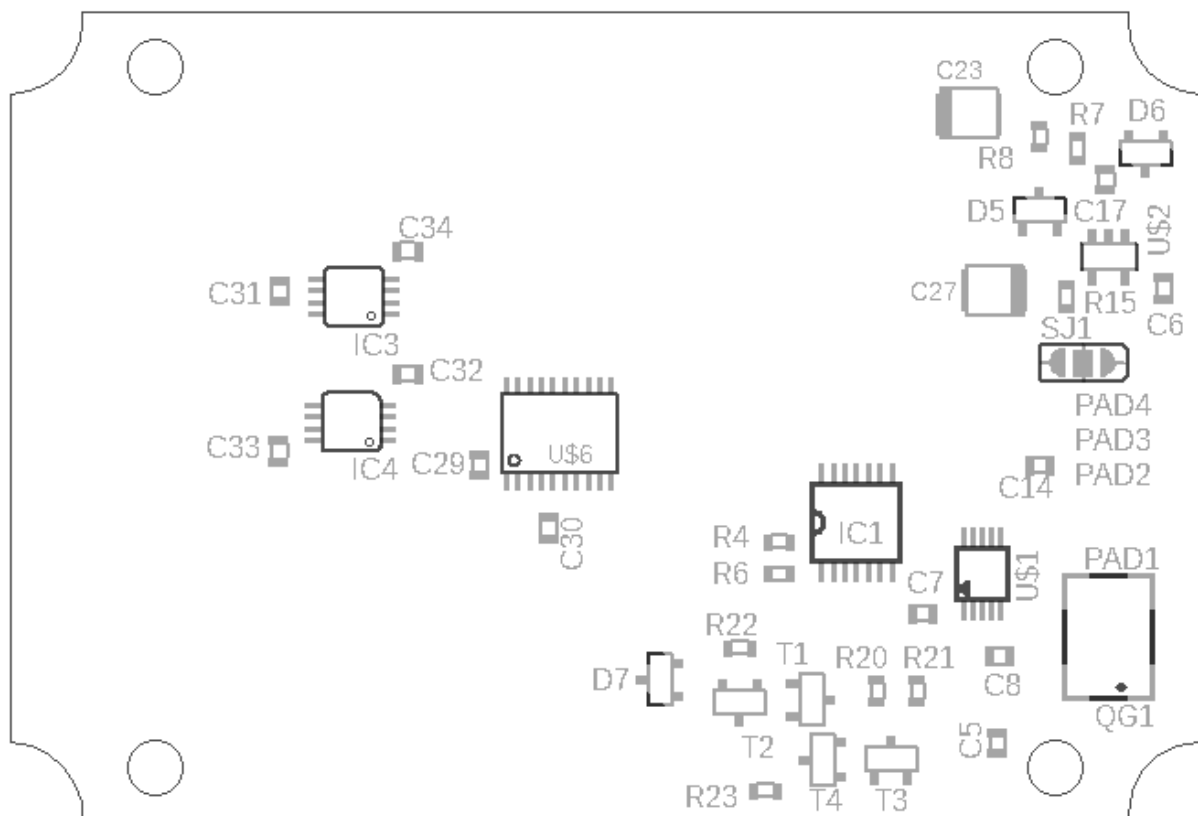


Layout, Bottom-Seite

12.6 Bestückungsplan



Bestückungsplan, Top-Seite



Bestückungsplan, Bottom-Seite

12.7 Bauteilliste

Pos.	Bauteil	Anzahl	Wert	Bauform
1	C1, C2	2	22pF	0603
2	C4, C5, C6, C7, C8, C9, C10, C14, C17, C20, C21, C22, C24, C25, C26, C28, C29, C30, C31, C32, C33, C34	22	100nF	0603
3	C3, C15, C16, C18, C19	5	1 μ F	SMC_A
4	C23, C27	2	33 μ F	SMC_B
5	C11	1	33nF	SMC_B
6	C12	1	470nF	SMC_B
7	C13	1	1 μ F	SMC_B
8	D5, D6, D7		BAT54S	SOT23
9	IC1	1	ADG1604S	TSSOP14
10	IC2	1	LM317	SOT223
11	IC3, IC4	2	AD8610MR	MSOP08
12	IC5	1	LM337	SOT223
13	L1	1	10 μ H	L3015
14	L2, L3, L4, L5, L6, L7	6	1,2 μ H	0805
15	LED1	1	grün	0603
16	Q1	1	16MHz	SM77H
17	QG1	1	25MHz	SM77H
18	R2, R3	1	22R	0603
19	R26	1	150	0603
20	R10, R15, R16	3	1k	0603
21	R9, R17	2	1,5k	0603
22	R1, R4, R5, R6, R11, R12, R13, R18, R19, R20, R21, R22, R23, R24, R25		10k	0603
23	R14	1	100k	0603
24	R8	1	220k	0603
25	R7	1	2M	0603
26	T1, T2, T3, T4	4	BC847	SOT23
27	U\$1	1	AD9833-SMD	MSOP10
28	U\$2	1	LT1615	SOT23-5
29	U\$3	1	ATMEGA32U4TQFP	TQFP44
30	U\$6	1	AD5144A	TSSOP20
31	U\$11	1	MICRO-SMD	MIKRO
32	X2	1	BU-SMA-R	SMA-R