

Universität
Rostock



Traditio et Innovatio

Erfahrungsbericht
zu dem Fachpraktikum bei
DIEHL Aerospace GmbH

Betreuer:

Prof. Dr. Ralf Salomon

vorgelegt dem

**Institut für Angewandte Mikroelektronik und
Datentechnik**

der

Universität Rostock

Maik Gotzmann

20.05.2011

Inhaltsverzeichnis

1	Einleitung	1
1.1	Allgemeines	1
1.2	Firmenporträt	1
2	Beschreibung der Tätigkeiten	3
2.1	Hardware- und Software-Integrationstest für Steuereinheiten	3
2.1.1	Konzept	3
2.1.2	Vorarbeit	5
2.1.3	Erstellung von Testscripten	6
2.2	Inbetriebnahme von Prototypen	6
2.2.1	Grundlagen und Testaufbau	6
2.2.2	Programmierung des DSP	7
2.3	Sonstige Tätigkeiten	9
3	Fazit	10

Abkürzungsverzeichnis

APU	:	Auxiliary Power Unit
CS	:	Chip Select
DSP	:	Digitaler Signalprozessor
ECB	:	Electronic Control Box
PCB	:	Printed Circuit Board
PWM	:	Pulsweitenmodulation
SIMO	:	Slave In Master Out
SOMI	:	Slave Out Master In

1 Einleitung

1.1 Allgemeines

Als Student des Studienganges Informationstechnik der Universität Rostock absolvierte ich mein Fachpraktikum vom 01.11.2010 bis 30.04.2011. Dabei entschied ich mich für das Unternehmen DIEHL Aerospace und wurde am Standort Rostock in der Abteilung Hardwareentwicklung eingesetzt.

Einer der Hauptaufgaben während meiner dortigen Tätigkeit war die Erstellung, Integration und Verifikation von Testskripten zur Durchführung von automatisierten Hard- und Softwaretests von elektronischen Steuereinheiten. Ein weiteres Aufgabenfeld bot sich durch die Inbetriebnahme zweier analoger Ausgangsstufen zur Spannungsversorgung. Die Regelung durch ein PWM (Pulsweitenmoduliertes)-Signal wurde dabei mit Hilfe eines Digitalen Signalprozessors (DSP) realisiert, den es zu programmieren galt. Darüber hinaus galt es für Laboranwendungen diverse kleinere Aufgaben zu erledigen.

Eine detaillierte Beschreibung zu den einzelnen Tätigkeitsfeldern erfolgt im zweiten Kapitel meines Erfahrungsberichtes. Im dritten Kapitel erfolgt eine abschließende Gesamtbewertung.

1.2 Firmenporträt

Das Unternehmen DIEHL Aerospace ist, neben DIEHL Aircabin und DASELL Cabin Interiors, ein Teil des Teilkonzerns DIEHL Aerosystems innerhalb der DIEHL Gruppe. Die 1250 bei DIEHL Aerospace beschäftigten Arbeitnehmer konzentrieren sich dabei hauptsächlich auf die deutschen Standorte Überlingen (HQ), Frankfurt/M., Nürnberg und Rostock. Entwicklung und Produktion erfolgen ausschließlich an diesen Orten.

DIEHL Aerospace ist dabei Systemlieferant für die zivile als auch für die militärische Luftfahrtindustrie. Die Kernkompetenzen des Unternehmens liegen im Bereich der

Avionik und Kabine mit folgenden Systemen:

- Kabinenbeleuchtung
- Kabinensicherheit
- Kabinen-Management-System
- Kabinen- und Versorgungssysteme
- Integrierte Modulare Avionik
- Cockpit- und Display-Systeme
- Primäre und Sekundäre Flugsteuerung
- Triebwerksregelung
- Energieumwandlung und -verteilung

Am Standort Rostock betreibt DIEHL Aerospace ein Entwicklungszentrum mit derzeit 26 Entwicklungsingenieuren/-innen, die sich in den Abteilungen Software- und Hardwareentwicklung aufteilen.

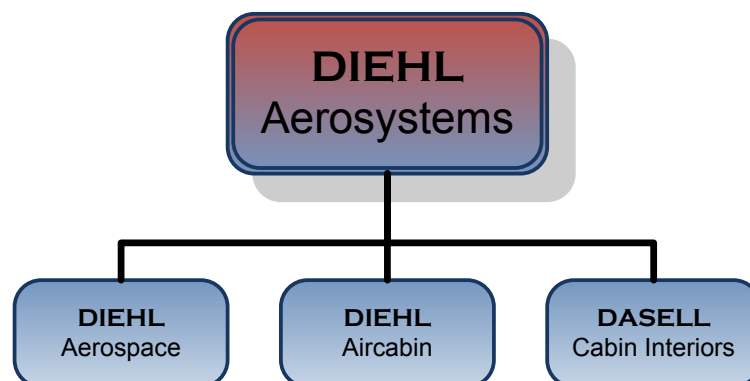


Abbildung 1.1: Struktur des Teilkonzerns DIEHL Aerosystems

2 Beschreibung der Tätigkeiten

2.1 Hardware- und Software-Integrationstest für Steuereinheiten

2.1.1 Konzept

Die hier zu testenden elektronischen Steuereinheiten (ECB, *engl. electronic control box*) dienen der Ansteuerung und Kontrolle einer Auxiliary Power Unit (APU). Diese Hilfstriebwerke sind Bestandteil in nahezu jedem modernen Flugzeug und dienen der Stromversorgung und dem Herstellen des Hydraulikdruckes für den autarken Betrieb der Flugzeugausrüstung am Boden, ohne dass dabei die Haupttriebwerke laufen. Die Stromerzeugung geschieht dabei durch Einsatz einer Gasturbine mit angeflanschem Generator. Für den Betrieb ist die APU mit einer Vielzahl an Sensoren und Aktoren ausgestattet, die über digitale und analoge Signale von der ECB verarbeitet werden.

Die zu testende ECB wird über ihr Interface zur Übertragung der digitalen und analogen In-/Outputs an einem Testrechnersystem, wie in Abbildung 2.1 dargestellt, angeschlossen. Grundsätzlich besteht der Testrechner aus einer Zentraleinheit mit aufgesetztem Echtzeitbetriebssystem und der Peripherie in Form diverser Interface Karten. Durch sie werden digitale und analoge Signale generiert, die dann als Stimuli an die ECB weitergeleitet werden. In entgegengesetzter Richtung werden Steuersignale von der ECB an den Testrechner über weitere spezielle Interface-Karten weitergeleitet, deren Auswertung dann in der Zentraleinheit erfolgt. Da in der Luftfahrtindustrie sehr hohe Anforderungen an die Ausfallsicherheit gestellt werden, kann durch die Interface Karten auch der Ausfall von Sensoren/Aktoren durch Stimuli eines Kurzschlusses oder offenen Leitung simuliert werden.

Durch einen handelsüblichen PC, hier als Programmer/Terminal bezeichnet, können dann die einzelnen Zustände der Interface Karten abgefragt und gesetzt werden. Damit erfolgt der elektronische Test der ECB. Der logische Test der Steuerungssoftware der ECB erfolgt mit einem speziellen Kommunikationsinterface zwischen ECB und Test-

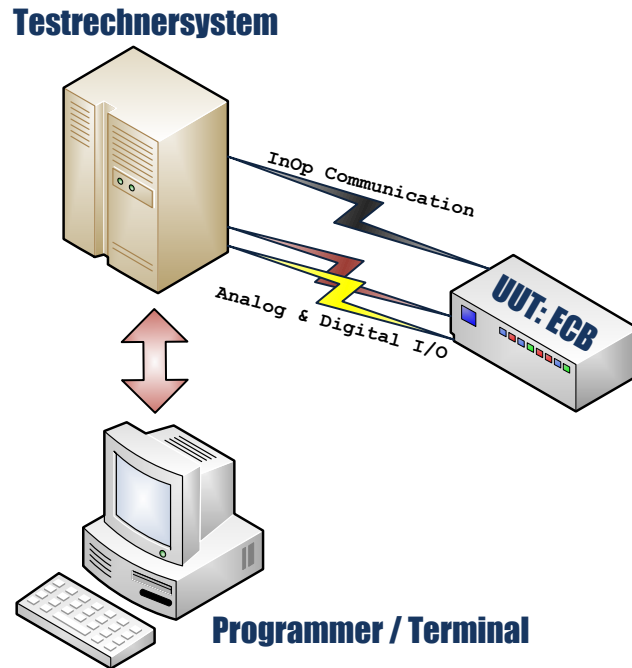


Abbildung 2.1: Schematische Darstellung der Testkonfiguration

rechner. Ähnlich einem Debugger können damit zur Laufzeit der Steuerungssoftware der Zustand beliebiger globaler Variablen überwacht und gesetzt werden.

Der Test einer ECB erfolgt dabei nach detaillierten Testvorschriften. Für jeden der vielen einzelnen Tests existiert ein genauer Prozedurablauf. Er enthält die zu setzenden Werte für die Stimuli/Variablen und einen Erwartungswert. Stimmt der vom Testrechner gelieferte Wert und der Sollwert überein, so gilt die einzelne Teststufe als bestanden.

Bisher wurde, quasi per Hand, jede einzelne Teststufe eines Einzeltest durch Setzen der entsprechenden Stimuli Werte über eine grafische Benutzeroberfläche am Terminal vorgenommen und die Reaktion mit dem Sollwert manuell verglichen und handschriftlich protokolliert. Der vollständige Test einer einzelnen ECB benötigt auf diese Weise mehrere Wochen.

Um die notwendige Testzeit zu verkürzen und den Output an verifizierten ECBs zu erhöhen, besteht die Bestrebung, die vorgegebenen Testprozeduren als Script in der Interpretersprache Python zu realisieren. Die einzelnen Tests können dann automatisch mittels Stapelverarbeitung ohne Zutun einer Person durchlaufen werden.

2.1.2 Vorarbeit

Das Setzen der einzelnen Stimulus über die Interface Karten und deren Zustandsabfrage erfolgt dabei über Testpunkte, die als konstante Objekte als Library in Python definiert werden. Die zur Deklaration notwendigen Daten können den Konfigurationsdaten des Testrechners entnommen werden. Da jedoch mehr als 100 Hardware Testpunkte existieren, ist eine automatische Generierung unumgänglich. Dies geschieht mit Hilfe eines Scriptes in der Interpretersprache Perl, da sie bereits mächtige Werkzeuge zur Verarbeitung regulärer Ausdrücke zum Filtern von Strings enthält.

Im Zuge meines Praktikums entstand unter Verwendung von Active Perl V.5.10 eine Konsolenanwendung zur automatischen Generierung der Python Library mit den Hardware Testpunkten.

Zum Beobachten/Setzen von globalen Variablen der Steuerungssoftware im laufenden Betrieb wurde ebenfalls solch ein Tool benötigt. Für das Lesen/Schreiben einer solchen Variable über das Kommunikationsinterface werden hier Daten wie Name, Speicheradresse im RAM, Speichergröße und evtl. Full Scale Wert benötigt. Name und Adresse konnten dem MAP File entnommen werden, welches nach dem Compilieren eines Softwareprojektes die Speicheraufteilung der Zielplattform angibt. Alle weiteren Informationen wurden durch Suchen im Quellcode realisiert. Ausgegeben wird dann eine valide Python Library mit über 2000 Software Testpunkten.

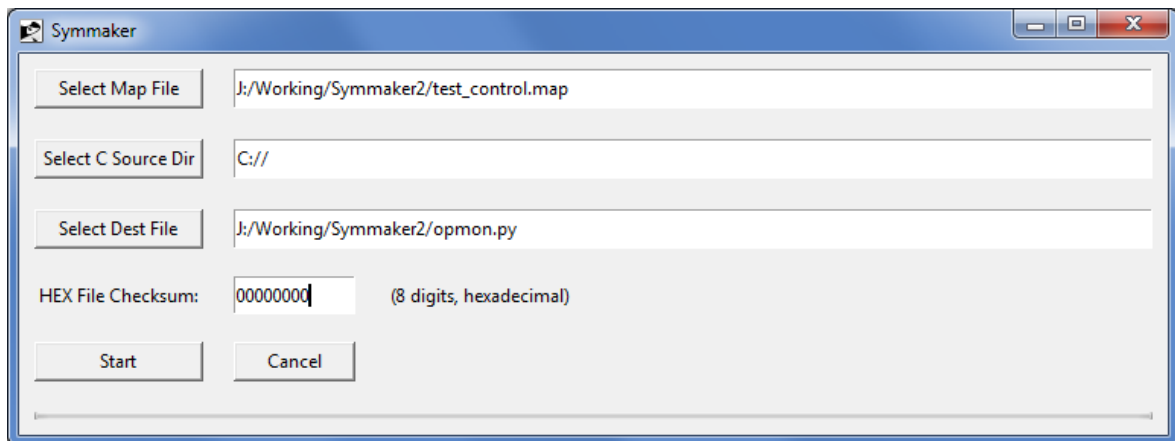


Abbildung 2.2: PERL Applikation zur Generierung der Software Testpunkte

Da letztgenannter Vorgang nach jeder Änderung des Softwarestandes durchgeführt werden muss, wurde die Bedienung dieser Applikation mit Hilfe einer grafischen Benutzeroberfläche realisiert. Hierzu wurde das standardmäßig in Perl enthaltene Modul **Tkx** verwendet. Die so programmierte grafische Benutzeroberfläche ist damit systemunabhängig, so wie die gesamte Applikation selbst auch.

2.1.3 Erstellung von Testscripten

Die Programmierung der Testscripte erfolgte in Python unter Benutzung von Eclipse als IDE. Im einfachsten Fall stellt das Testscript eine direkte Umsetzung der Vorschrift für einen Einzeltest dar, der aus 10 bis 100 Teststufen besteht.

Jedes Testscript beginnt dabei mit der Deklaration der verwendeten Testpunkte aus den zuvor generierten Libraries. Anschließend folgt der Anweisungsblock, der sequenziell abgearbeitet wird. In ihm werden einzelne Testpunkte gesetzt und die Reaktion abgefragt und mit dem Erwartungswert verglichen. Das Resultat wird in einem Prüfprotokoll gespeichert und kann nach Durchlauf des gesamten Tests als Logdatei eingesehen werden.

Die einzelnen Teststufen der Prüfvorschrift wurden dabei parallel mit den Forderungen des Auftragsgeber verglichen, um weitere Fehler auszuschließen.

2.2 Inbetriebnahme von Prototypen

2.2.1 Grundlagen und Testaufbau

Der zweite Aufgabenkomplex bestand darin, zwei analoge Ausgangsstufen für die Spannungsversorgung aufzubauen, in Betrieb zu nehmen und zu testen. Das erste der beiden Boards liefert dabei eine geregelte Ausgangsspannung von 0-10 V DC. Das zweite ist für die Erzeugung einer Ausgangsspannung von 0-28 V DC vorgesehen.

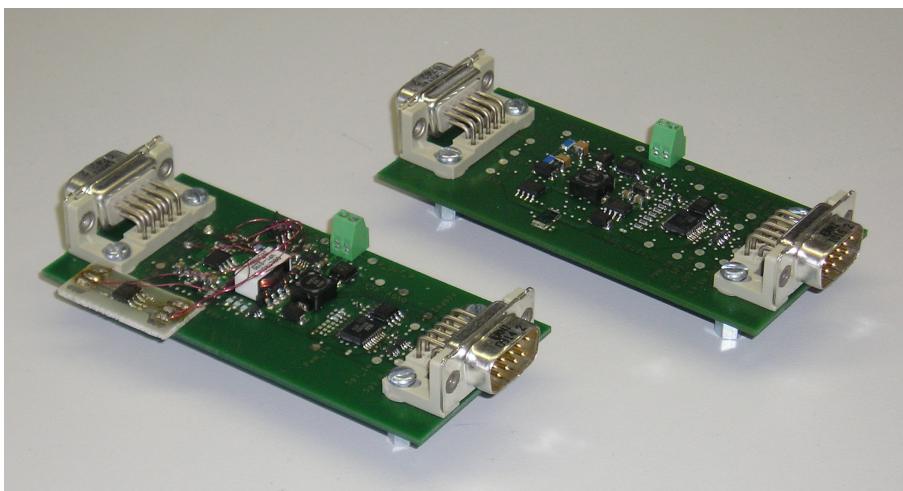


Abbildung 2.3: Analoge Ausgangsstufen, links: 10 V DC, rechts: 28 V DC

Beide elektronischen Leiterplatten (PCB, engl. Printed Circuit Board) wurden durch

die Industrie gefertigt und lagen bereits vor. Das Bestücken dieser PCBs mit SMD Bausteinen erfolgte dann manuell durch meine Person.

Zur Erstinbetriebnahme wurde zur Regelung der Ausgangsspannung ein Funktionsgenerator zur Erzeugung des notwendigen PWM Signals herangezogen. Auftretende Fehler, die bei manueller Bestückung nicht auszuschließen sind, galt es bei der Fehlersuche/-beseitigung zu beheben. Zu diesem Zeitpunkt erfolgte ebenso die Einarbeitung erster Schaltungsmodifikationen.

Die tatsächliche Regelung/Steuerung sollte dabei ein DSP übernehmen. Dazu wurde das Evaluation Board eZdsp der Firma Spectrum Digital verwendet. Als DSP kommt dort der TMS320F2008 zur Anwendung. Da sich das einzelne Abgreifen der notwendigen digitalen In- und Outputs über die 40polige Steckerleiste des Evaluation Boards als unpraktisch erwies, wurde eine entsprechende Adapterplatine kreiert. Diese wird auf einfache Weise auf das Evaluation Board gesteckt und leitet unter anderem die Signale der PWM und die der SPI Kommunikation an eine 9polige SUB-D Buchse weiter. Über sie erfolgt dann die Kommunikation zwischen DSP und des Prototypen. Über die vier aufgesetzten 2 mm Laborbuchsen können zusätzlich Signale abgegriffen oder eingespeist werden.

2.2.2 Programmierung des DSP

Die Programmierung des DSPs erfolgte mit der IDE Code Composer von Texas Instruments in der Sprache C.

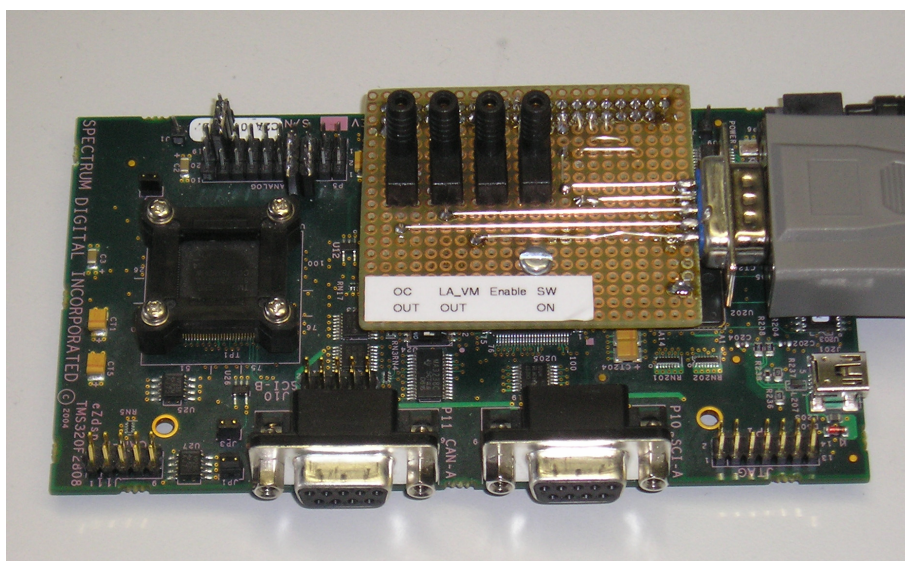


Abbildung 2.4: DSP Evaluation Board eZdsp mit Adapterplatine

Ein Schwerpunkt dabei bildete die Realisierung der SPI Kommunikation zwischen dem DSP (Master) und dem auf den Ausgangsstufen verbauten A/D-Wandler (Slave). Dieser besitzt insgesamt 8 analoge Eingangskanäle und liefert 16-Bit Samples bei einer maximalen Abtastfrequenz von 100 kHz. Einer dieser Kanäle dient der Wandlung des analogen Spannungspegels, welcher die aktuell anliegende Ausgangsspannung repräsentiert. Dieser wird später für die Regelung benötigt. Der DSP erzeugt somit die drei Signale Slave-In-Master-Out (SIMO), Chip Select (CS) und SPI Clock. Ein viertes Signal, Slave-Out-Master-In (SOMI) wird durch den DSP verarbeitet und erhält den Sample als Rückgabewert der Anfrage, die über einen timergesteuerten Interrupt erfolgt.

Zur Realisierung der CAN Bus Kommunikation lag bereits eine Implementierung vor, die ohne aufwendige Modifikation in das Softwareprojekt integriert werden konnte. Über ihr können globale Variablen zur Laufzeit der DSP Software beobachtet und manipuliert werden. Durch geeignete PC-basierende Softwaretools konnten so Analysen zu einzelnen Signalen durchgeführt werden.

Durch einen in der Schleife ablaufenden P-Regler wird das PWM Signal zur Steuerung der Ausgangsspannung erzeugt. Die Regelung hält die Ausgangsspannung bei Lastwechsel konstant und verringert das Überschwingen der steigenden/fallenden Flanken bei Erhöhung/Verringerung der Ausgangsspannung.

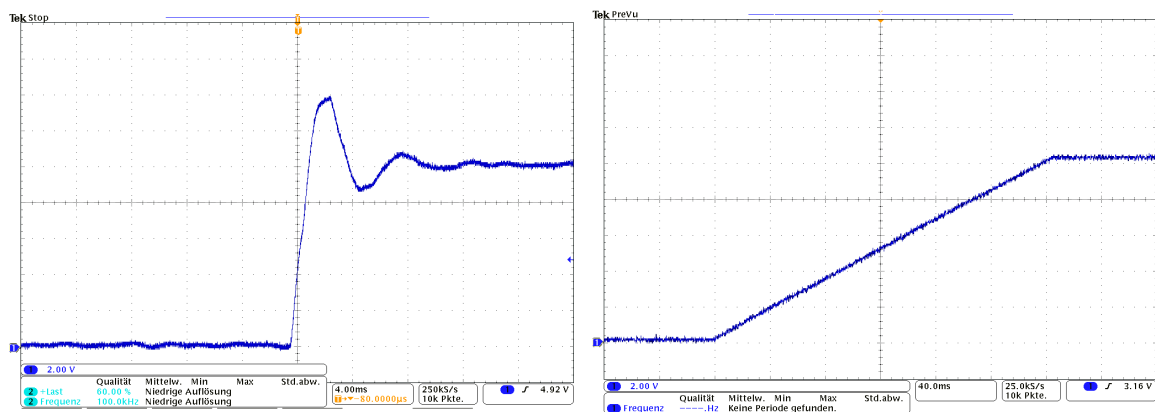


Abbildung 2.5: Verringerung des Überschwingens durch Regleranpassung

Da für beide Ausgangsstufen der selbe DSP mit dem einen Softwareprojekt genutzt werden sollte, wurde durch Setzen einer Variablen die Möglichkeit realisiert, im laufenden Betrieb die Ausgangsstufe zu tauschen.

2.3 Sonstige Tätigkeiten

Für die Inbetriebnahme der Prototypen und die Durchführung erster Tests an ihnen, wurden durch mich im Laufe des Praktikums diverse Kabel konfektioniert. Eine weitere Aufgabe war die Erweiterung und Modifizierung von speziellen Testapparaturen.

Eine langfristige angelegte Nebentätigkeit war es, ein Programmiergerät für FPGAs nach Schaltungsvorgabe neu zu entwickeln. Bei der Konzeptionierung standen Robustheit und Kompaktheit im Vordergrund, um eine hohe Zuverlässigkeit bei täglichem Gebrauch im Labor zu gewährleisten. Die gesamte Elektronik sollte dazu innerhalb eines Steckergehäuses platz finden. Dieser verbindet die parallele Schnittstelle des PC's mit dem JTAG Programmierinterface des FPGA's. Alle Bauteile mussten dazu in passender Baugröße von entsprechenden Anbietern ausgewählt und beschafft werden. Bei dem Design des PCBs zur Aufnahme der elektronischen Schaltung bestand die Herausforderung darin, alle Komponenten auf der Platinenfläche von lediglich 35 mm x 24 mm unterzubringen. Für das PCB Layout wurde die Software EAGLE benutzt und das Routing aufgrund der kompakten Abmessungen vollständig manuell durchgeführt. Da mein Praktikum noch vor der Fertigstellung der PCBs durch einen externen Auftragnehmer endete, verbleibt dieses Programmiergerät in der Konzeptphase.

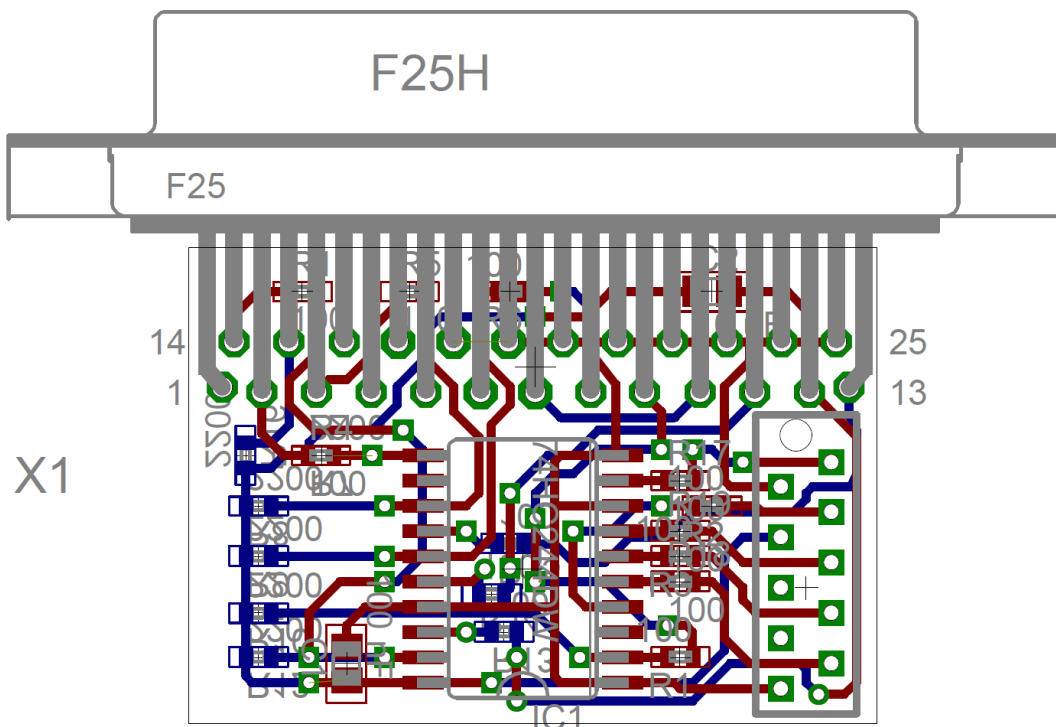


Abbildung 2.6: Layout der Programmierschaltung

3 Fazit

Während meiner 6-monatigen Tätigkeit als Praktikant im Unternehmen DIEHL Aerospace wurde ich mit einem breiten und anspruchsvollen Aufgabenspektrum konfrontiert.

Zur Bewältigung dieser konnte ich die in der Universität Rostock erworbenen theoretischen Kenntnisse zum Teil nutzen. Neue Kenntnisse konnten durch das Erlernen der Programmiersprachen PERL und PYTHON hinzugewonnen werden.

Durch die Vielzahl an anfallenden praktischen Arbeiten konnte so auch die Brücke zwischen der Theorie und Praxis geschlagen werden.