



PCB Prototyper

Zur Erlangung des Grades

Master of Advanced Studies

im Studiengang Mikroelektronik

der FHNW / FHO

vorgelegt von

Sebastian Stapfer

Eingereicht am:

28.02.2013

EIDESSTATTLICHE ERKLÄRUNG

Hiermit erkläre ich, Sebastian Stapfer geboren am 16.09.1980 in Kirchberg ZH, dass die vorgelegte Masterarbeit mit dem Titel PCB Prototyper durch mich selbstständig verfasst wurde. Ich habe keine anderen, als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit nicht bereits in gleicher oder ähnlicher Fassung zur Erlangung eines akademischen Grades eingereicht.

Datum: 28.02.2013 **Unterschrift** _____

ABSTRACT

PCB Prototyper

Von Sebastian Stapfer

In dieser Arbeit wird die Entwicklung einer autonomen Steuerung auf Basis der Arduino-Plattform für eine CNC Maschine dokumentiert. Die CNC Maschine wird verwendet, um Prototypen von Elektronikplatinen mechanisch zu fräsen. Die Fräsdaten können dabei mit Hilfe eines Webserver von irgendeinem Standort mit Zugang zum Netzwerk der FHNW auf die Steuerung geladen werden. Eingesetzt werden ein Arduino Mega2560, ein Ethernetshield und ein selber entwickeltes Schnittstellenshield zur Anbindung an eine ISEL ICP 4050 CNC Maschine. Es wird gezeigt, wie Schritt für Schritt das jetzt produktiv eingesetzte System evaluiert und entworfen wurde.

This paper shows the development of an autonomous controller for a CNC milling machine based on the Arduino platform. The CNC is used for milling prototype circuit boards mechanically. The milling data can thereby be uploaded to the controller from any location with access to the network of the UAS of Northwestern Switzerland using a web server. The system consist of an Arduino Mega2560 an attached Ethernet shield and a self-developed interface shield to connect to an ISEL ICP 4050 CNC machine. Readers will get step by step from evaluation to the now productively used system.

INHALTSVERZEICHNIS

ABSTRACT	V
ABBILDUNGSVERZEICHNIS	IX
TABELLENVERZEICHNIS	XI
DANKSAGUNG.....	XIII
1 EINLEITUNG.....	15
1.1 Motivation	15
1.2 Problemstellung.....	15
1.3 Zielsetzung.....	16
1.4 Struktur.....	17
2 HAUPTTEIL.....	18
2.1 Darstellung und Analyse	18
2.2 Zeitplanung.....	18
2.3 Lösungsschritte.....	20
2.3.1 Recherche und Auswahl eines CNC Fräsroboters	20
2.3.2 Auswahl eines geeigneten Mikrocontrollers.....	24
2.3.3 Entwicklung des autonomen Systems.....	30
2.3.4 Elektronikentwicklung der Schnittstelle zur Fräsmaschine	36
2.3.5 Programmierung der Fräsmaschinensteuerung.....	38
2.3.6 Automatisierte Messung der Werkzeuglänge.....	42
2.3.7 Implementierung einer Benutzerschnittstelle	43
2.4 Lösung.....	46
3 SCHLUSS.....	47
3.1 Zusammenfassung	47
3.2 Bewertung	47
3.3 Ausblick.....	48
4 ANHANG	49
4.1 Elektronik.....	49
4.1.1 Schrittmotorsteuerung.....	49
4.1.2 RS232 - Arduino Schnittstelle	54
4.2 Software	55
4.2.1 Bibliotheken	55
4.2.2 Zustandsdiagramme.....	59
4.2.3 G-Code Referenz	63
4.2.4 Sourcecode Hauptprogramm	65
4.2.4.1 CNCInterface.ino	65
4.2.4.2 CNCHandler.ino	74
4.2.4.3 Streaming.ino	76
4.2.4.4 UI.ino	81
4.2.5 Sourcecode CNCInterpreter Klasse	83
4.2.5.1 CNCInterpreter.h	83
4.2.5.2 CNCInterpreter.cpp	84
4.2.6 HTML Code des Webserver	89

4.2.6.1	Index.htm	89
4.2.6.2	Upload.htm.....	89
4.2.6.3	Done.htm.....	91
4.2.6.4	Logout.htm	92

ABBILDUNGSVERZEICHNIS

Abbildung 1: Portalmaschine	20
Abbildung 2: Knickarmroboter	21
Abbildung 3: Evaluierungsboard von Embest, EM- STM3210C.....	26
Abbildung 4: JTAG Programmieradapter J-Link EDU.....	27
Abbildung 5: CrossStudio for ARM v2.2, Entwicklungsumgebung für ARM Mikrocontroller	27
Abbildung 6: Arduino Mega2560 R3, 8 Bit Mikrocontroller.....	28
Abbildung 7: Visual Studio 2012 mit Arduino	29
Abbildung 8: Blockdiagramm Systemübersicht.....	30
Abbildung 9: Datenfluss beim Öffnen einer Webseite.....	31
Abbildung 10: Datenfluss beim Hochladen von Daten	31
Abbildung 11: Datenfluss bei der Dateiauswahl	31
Abbildung 12: Datenfluss beim Fräsen	32
Abbildung 13: Geplanter Ablauf aus Sicht des Bedieners	33
Abbildung 14: Übersicht Gesamtsystem mit Motorsteuerungen und Spannungsversorgung	34
Abbildung 15: Schrittmotor Leistungsendstufe	36
Abbildung 16: Schrittmotorsteuerung	37
Abbildung 17: Erweiterungsplatine für den Arduino MEGA2560	37
Abbildung 18: Verwendete Bibliotheken	38
Abbildung 19: Unterteilung in Hilfsdateien.....	38
Abbildung 20: Schematischer Ablauf des Hauptprogrammes	39
Abbildung 21: Bestimmung der Werkzeuglänge mit „virtuellem“ Schalter.....	42
Abbildung 22: Startseite des Webservers (index.htm)	43
Abbildung 23: Upload Formular zum Hochladen einer CNC Datei (upload.htm).....	44
Abbildung 24: Zeilenweise Darstellung der Übermittelten Daten	44
Abbildung 25: LC Display eines Nokia 5110	45
Abbildung 27: Komponentenübersicht des entwickelten Systems.....	46
Abbildung 26: Fotografie der CNC Schnittstelle.....	46
Abbildung 28: Schrittmotorcontroller	49
Abbildung 29: Ein- und Ausgabesignale	50
Abbildung 30: Endschalter	51
Abbildung 31: Treiberstufen	52
Abbildung 32: Speisung.....	53
Abbildung 33: CNC Erweiterungsplatine für Arduino.....	54
Abbildung 34: Verwendete Bibliotheken	55

Abbildung 35: Übersicht Zustandsdiagramm Fräsvorgang	59
Abbildung 36: Zustandsdiagramm Initialisierung.....	60
Abbildung 37: Zustandsdiagramm Fräsvorbereitung.....	61
Abbildung 38: Zustandsdiagramm Fräsen.....	62

TABELLENVERZEICHNIS

Tabelle 1: Detaillierte Auflistung der Ziele	17
Tabelle 2: Geplanter Projektablauf	19
Tabelle 3: Vor- und Nachteile Robotiksysteme	21
Tabelle 4: Übersicht Portalfräsen	23
Tabelle 5: Übersicht ausgewählter Mikrocontrollerarchitekturen	24
Tabelle 6: Auswahl von 32 Bit Mikrocontrollern	25
Tabelle 7: Befehlssyntax CNC Controller	40
Tabelle 8: Initialisierungsablauf CNC Controller	41
Tabelle 9: Übersicht Allgemeine G-Code Befehle	63
Tabelle 10: Beschreibung einiger ausgewählter G-Code Befehle	63

DANKSAGUNG

Ich möchte mich bei allen bedanken, welche mich mit Rat und Tat unterstützt haben.

*Besonderen Dank gelten meinem Betreuer Herrn Lukas Kurmann und meinem Arbeitgeber,
dem Studiengang Mechatronik Trinational, ohne dessen Finanzierung diese Arbeit gar nicht
möglich gewesen wäre.*

1 EINLEITUNG

1.1 MOTIVATION

Um die drei CAS¹ Kurse (Systeme, Mikroelektronik Analog, Mikroelektronik Digital) mit einem MAS² Abschluss abzuschliessen ist eine grössere eigenständige Arbeit erforderlich. Sie muss sich im Bereich Mikroelektronik befinden, wobei da mit Systemen, Analog- und Digitalelektronik immer noch ein sehr weites Feld abgesteckt ist. Die Arbeit ermöglicht das Einsetzen von hauptsächlich theoretischen Grundlagen an einem konkreten Produkt. Da beim jetzigen Arbeitgeber, dem Studiengang Mechatronik an der FHNW, aktuell eine Lösung zum Herstellen von (Elektronik-) Prototypen gesucht wird, bietet sich die MAS Arbeit mit einer Entwicklung in diesem Gebiet an.

1.2 PROBLEMSTELLUNG

Um für Studierendenprojekte Leiterplattenprototypen herzustellen, mussten diese bisher entweder extern gefertigt werden (z.B. PCB-Pool) oder selber in aufwändiger Handarbeit belichtet und mit relativ problematischen Chemikalien entwickelt und geätzt werden.

Die Problematik der extern gefertigten Prototypen ist, dass selbst bei Expresslieferung in der Regel 3 Tage gewartet werden muss, bis der erste Prototyp getestet werden kann. Dazu kommen der hohe Preis³ für Expresslieferung und die eigentlich nicht benötigte professionelle Qualität des Prototyps.

¹ Certificate of Advanced Studies (früher Nachdiplomkurs)

² Master of Advanced Studies (früher Nachdiplomstudium)

³ PCB-Pool: 100 x 160 mm, eine Lage ohne Druck, 3 Tage Lieferfrist ab 140€, www.pcb-pool.com

1 Einleitung

Bei eigener Prototypenfertigung (Ätzverfahren) kann der Vorgang zwar beschleunigt werden, es braucht aber immer noch eine zusätzliche Betreuung der Studierenden und eine hohe manuelle Bearbeitungszeit (Belichten, Entwickeln, Ätzen, Reinigen, Bohren). Ausserdem ist das Hantieren mit den Chemikalien nicht ganz unproblematisch, ganz abgesehen von deren Entsorgung. Aus diesen Gründen soll eine Maschine entwickelt werden, welche mit Fräsdaten aus der Layoutsoftware Target 3001⁴ mittels Isolationsfräsen und Bohren einen Leiterplattenprototypen herstellt. Die Maschine soll die Bearbeitungsdaten im CNC Format ohne einen externen Steuerrechner verarbeiten und nach manueller Bestätigung den Prototypen selbstständig fräsen und bohren. Ausserdem muss die Maschine den gesetzlichen Sicherheitsbestimmungen entsprechen.

1.3 ZIELSETZUNG

Die Maschine soll am Schluss der Arbeit zur Prototypenherstellung eingesetzt werden. Damit diese Aufgabe erfüllt werden kann, soll sich die Arbeit auf die Entwicklung einer Elektronikchnittstelle zu einem Fräsroboter beschränken. Dieser muss zu Beginn evaluiert und eingekauft werden. Anschliessend muss ein Schnittstellensystem entwickelt werden, das einen externen Steuerrechner ersetzt, den Fräsroboter steuert und Daten von einem externen Rechner entgegennehmen kann. Für das Frässystem muss am Schluss ein Benutzerhandbuch geschrieben werden. In folgender Tabelle sind Muss- und Sollziele detailliert aufgelistet:

⁴ Elektronik CAD Software der Firma Ing.-Büro FRIEDRICH, <http://www.ibfriedrich.com/> Abgerufen am 03.01.13

1 Einleitung

Tabelle 1: Detaillierte Auflistung der Ziele

Beschreibung des Ziels	MUSS oder SOLL
Mechanische Konstruktion auswählen	MUSS
Wahl eines geeigneten Mikrocontrollers	MUSS
Entwicklung einer autonomen ⁵ Prozesssteuerung	MUSS
Programmierung der XY und Z-Achsensteuerung	MUSS
Automatisierte Messung der Werkzeuglänge	MUSS
Implementierung einer intuitiven Benutzerschnittstelle	MUSS
Erstellen eines Benutzerhandbuches	MUSS
Automatischer Werkzeugwechsel	SOLL
Schmutz- (Span) Absaugung	SOLL
Webserver zur Fernsteuerung /-überwachung implementieren	SOLL

1.4 STRUKTUR

Der Hauptteil gliedert sich in eine Darstellung und Analyse, danach folgen die Lösungsschritte und zum Schluss die finale Lösung. Die Lösungsschritte orientieren sich dabei am Vorgehen während der Arbeit, wobei sich die Programmierung über mehrere Phasen erstreckt hat. Im Schlussteil wird die Arbeit noch einmal zusammengefasst, gewertet und ein Ausblick auf mögliche zukünftige Entwicklungen und Arbeiten gegeben.

⁵ Die Steuerung interpretiert die Fräsdaten ohne ständige Verbindung zu einem „Steuerrechner“

2 HAUPTTEIL

2.1 DARSTELLUNG UND ANALYSE

Beim mechanischen Herstellen von Platinen wird die Kupferschicht auf einem Trägermaterial nicht wie beim chemischen Verfahren weggeätzt, sondern mit einem Fräser abgetragen. Dabei ist das Verfahren besonders effektiv, wenn nicht eine ganze Kupferfläche, sondern nur die Leiterbahnumrisse (Auren) entfernt werden müssen. Als zusätzlicher Schritt bietet sich das Bohren der Löcher für Bauteile an, da dies nur einen Wechsel des Werkzeuges erfordert. Diese Art mechanischer Herstellung heisst im englischen “Printed circuit board milling”⁶ und wird schon in einigen Projekten⁷ zur Prototypenfertigung erfolgreich eingesetzt. Alle untersuchten Lösungen basieren auf einem PC, welcher den Fräsvorgang (mit einer speziellen Software) steuert.

2.2 ZEITPLANUNG

Das Projekt gliedert sich grob in vier Phasen:

1. Recherche und Vergleich von verschiedenen CNC Maschinen
2. Entwicklung einer Elektronikchnittstelle
3. Programmierung des eingebetteten Systems
4. Tests und verfahrenstechnische Verbesserungen

⁶ Ein Artikel zum PCB milling findet sich unter: http://en.wikipedia.org/wiki/Printed_circuit_board_milling
Abgerufen am 27.02.13

⁷ Zum Beispiel auf RepRap: http://reprap.org/wiki/PCB_Milling Abgerufen am 27.02.13

2 Hauptteil

Ziel des Projektes ist nicht „nur“ einen Prototypen herzustellen, sondern eine produktive und gut bedienbare Maschine, weshalb vor allem für Tests genügend Zeit eingerechnet werden muss.

Als Zeitaufwand für diese Arbeit sollten ungefähr 380 Stunden eingesetzt werden. Dies ergibt folgenden geplanten Ablauf:

Tabelle 2: Geplanter Projektablauf

Aufgabe Nr.	Aufgabenname	Beginn	Ende
1	Projektorganisation & -planung	13.08.2012 08:00	24.08.2012 16:00
2	Recherche und Evaluierung mechanische Konstruktion	15.08.2012 08:00	20.08.2012 16:00
3	Konzept zur Elektronik erstellen	21.08.2012 08:00	24.08.2012 16:00
4	Bestellung mechanische Konstruktion	29.08.2012 08:00	
5	Entwicklung und Test der Motorsteuerplatine	27.08.2012 08:00	31.08.2012 16:00
6	Entwicklung und Test der Hauptplatine	03.09.2012 08:00	14.09.2012 16:00
7	Elektronikentwicklung abgeschlossen	17.09.2012 08:00	
8	Programmierung der CNC-Steuerung	05.11.2012 08:00	28.12.2012 16:00
9	Programmierung abgeschlossen	31.12.2012 08:00	
10	Dokumentation, Benutzerhandbuch	02.01.2013 08:00	11.01.2013 16:00
11	Abschlusspräsentation	18.01.2013 08:00	

Tests und Verbesserungen an der Maschine haben länger gedauert als geplant, weshalb die Abgabe der Arbeit um einen Monat verschoben werden konnte.

2 Hauptteil

2.3 LÖSUNGSSCHRITTE

2.3.1 RECHERCHE UND AUSWAHL EINES CNC FRÄSROBOTERS

Es gibt verschiedene Konzepte von CNC Bearbeitungsstationen. Sehr häufig werden Portalfräsmaschinen eingesetzt, bei welchen sich der Bearbeitungskopf (Fräser oder Bohrer) senkrecht über der horizontalen Tischfläche befindet. Er kann sich dabei parallel zur Tischfläche in X- und Y-Richtung bewegen. Zusätzlich kann der Kopf in vertikaler Z-Achse bewegt werden. Vorteile sind die kompakte Konstruktion und die drei orthogonalen Achsen, welche konstruktionsbedingt einen dreidimensionalen Raum aufspannen. Die Zuordnung von X,Y und Z Koordinaten in Achsbewegungen ist dabei trivial. Ausserdem sind sie durch die einfache mechanische Konstruktion sehr preiswert. Oft werden die Portalfräsmaschinen als sogenannte 2.5 dimensionale Bearbeitungsmaschinen bezeichnet, da das Werkstück dabei immer von oben bearbeitet wird. Die Portalfräsmaschinen können deshalb mit einer vierten (Dreh-)Achse aufgerüstet werden, um jeden Punkt eines dreidimensionalen Körpers zu erreichen.

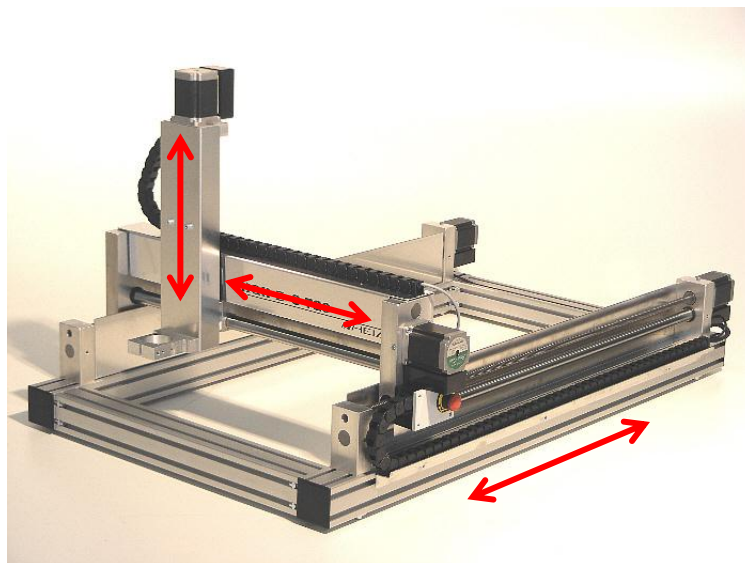


Abbildung 1: Portalmaschine

2 Hauptteil

Ein anderes Konzept sind sogenannte Knickarmroboter mit Fräskopf. Dabei bilden mindestens drei motorisierte Gelenke einen Arm. Die Bearbeitungsmaschine befindet sich am Ende des Armes und kann so innerhalb einer Halbkugel um das erste Gelenk positioniert werden. Die Bewegungskinematik ist dabei relativ komplex, es kann aber jeder Punkt im Bearbeitungsraum von verschiedenen Seiten angefahren werden. Die Kosten für einen Knickarmroboter sind allerdings relativ hoch.

Tabelle 3: Vor- und Nachteile Robotiksysteme

Merkmal	Portalfräsmaschine	Knickarmroboter
Komplexität der Kinematik	einfach	schwierig
3D Bearbeitung	2.5D	3D
Platzbedarf	kaum grösser als Werkstück	gross
Preis (sFr)	500 – 10'000	> 30'000



Abbildung 2: Knickarmroboter

2 Hauptteil


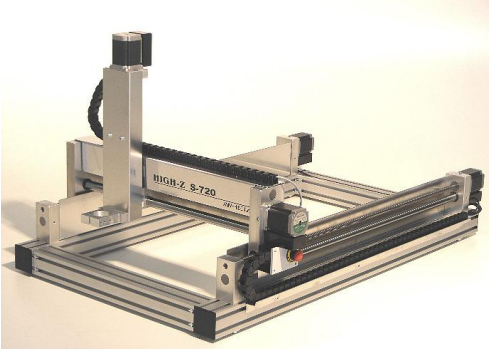

Aufgrund der vielen Vorteile einer Portalfräsmaschine wird der Ansatz mit einem Knickarmroboter nicht mehr weiterverfolgt. Bei der Auswahl einer Portalfräsmaschine gibt es je nach Hersteller grosse Unterschiede. Es gibt mechanisch einfache Systeme mit Aluminiumrahmen und Gewinde- oder Trapezstangen zu einem günstigen Preis (unter 1000 sFr.). Es ist allerdings fraglich, ob diese Maschinen die geforderten Genauigkeiten erfüllen können. Ausserdem muss für diese Maschinen ein Gehäuse angefertigt werden, um die Sicherheit zu gewährleisten. Es gibt auch teurere Systeme, welche deutlich robuster gebaut sind, Kugelumlaufspindeln haben und bereits in einem Gehäuse eingebaut sind. So eine Maschine ist die ICP 4050 von ISEL. Ein weiterer Vorteil ist, dass die Maschine bereits mit einer funktionsfähigen Schrittmotorsteuerung⁸ und einem Fräscontroller ausgestattet ist. Zudem sind für dieses System eine spezielle Gravierspindel mit hoher Umdrehungszahl⁹ und ein mechanischer Höhenausgleich verfügbar. Die Maschinensteuerung über RS232 mit einem PC und mitgelieferter Software kann eventuell zur Inbetriebnahme verwendet werden, soll aber anschliessend durch ein Mikrocontrollersystem ersetzt werden. Aufgrund der genannten Vorteile wird die ISEL ICP 4050 als Frässystem ausgewählt.

⁸ 3 Achsen werden angesteuert, es ist aber bereits ein vierter Controller für eine zusätzliche Achse vorhanden.

⁹ Maximale Umdrehungszahl der Gravierspindel sind 60'000 Umdrehungen pro Minute

2 Hauptteil

Tabelle 4: Übersicht Portalfräsen

Hersteller	Eigenschaften	
BZT (www.bzt-cnc.de) PF-750P	Grösse Arbeitsraum (X/Y/Z): 410 x 550 x 110 mm Achsenausführung: Kugelrollspindeln Steuerung: 3 Schrittmotoren ohne Endstufen und Controller	
CNC-Step (www.cnc-step.de) High-Z S-720	Grösse Arbeitsraum (X/Y/Z): 720 x 420 x 110 mm Achsenausführung: Trapezspindeln Steuerung: 4 Schrittmotoren ohne Endstufen und Controller	
ISEL (www.imes-core.de) ICP 4050	Grösse Arbeitsraum (X/Y/Z): 400 x 300 x 140 mm Achsenausführung: Kugelumlaufspindeln Steuerung: Step-Controller mit 4 Endstufen und Netzteil Schnittstelle: RS232	

2 Hauptteil

2.3.2 AUSWAHL EINES GEEIGNETEN MIKROCONTROLLERS

Als erstes muss die Architektur, also die Bitbreite des Mikrocontrollers festgelegt werden. Zur Auswahl stehen 8, 16 und 32bit. 8bit Architekturen sind immer noch weit verbreitet (Atmel, Pic usw.), sie sind billig und es gibt eine grosse Anzahl an Unterlagen und Beispielanwendungen. Ausserdem sind die Systeme relativ einfach aufgebaut. Nachteile sind die geringe Bandbreite und die Begrenzung in der Speicheradressierung. Modernere Architekturen sind 32bit breit, haben also eine deutlich grössere Bandbreite und können mehr Speicher verwalten. Nachteile sind dabei die oft hohe Komplexität und die geringe Anzahl an guten Unterlagen und Beispielanwendungen. Dies führt zu einer grossen Einstiegshürde, wenn die Wahl auf einen neuen 32bit Mikrocontroller fällt.

Tabelle 5: Übersicht ausgewählter Mikrocontrollerarchitekturen

Hersteller	8 Bit	16 Bit	32 Bit
ATMEL	AVR	-	AVR32 / ARM
Microchip	PIC10/12/16	PIC24	PIC32
STMicroelectronics	STM8	ST10	STM32
Cypress	PSoC 1/3	-	PSoC 5
Xilinx	PicoBlaze [*]	-	MicroBlaze [*]
Altera	-	-	NIOS II [*] / ARM [*]

^{*} Bei diesen Controllern handelt es sich um sogenannte SoftCores, welche auf einem FPGA realisiert werden

Zu Beginn der Arbeit wurde geprüft, ob sich das System mit einem 32 Bit Mikrocontroller realisieren lässt. Dazu wurden zuerst verschiedene 32 Bit Controller mit einander verglichen. Wichtige Kriterien waren der Preis, dass ein Ethernet MAC, ein USB Host und eine JTAG Schnittstelle integriert sind und dass die Programmierertools nicht mehr als 250 Fr. kosten.

2 Hauptteil

Tabelle 6: Auswahl von 32 Bit Mikrocontrollern

Hersteller / Modell	Preis / Stück	Schnittstellen	Programmiertools
ATMEL AT32UC3A1512	25 sFr.	Ethernet MAC, USB-Host, 7 16bit PWM, 69 GPIO	Atmel Studio 5/6 AVR JTAG ICE-MK3 ¹⁰
ATMEL ATSAM3X8E	Nicht verfügbar	Ethernet MAC, USB-Host, 9 32bit Timer, 8 16bit PWM, 103 IO	Atmel Studio 5/6 AVR JTAG ICE-MK3
ST STM32F107RCT6	9 sFr.	Ethernet MAC, USB-Host, 4 16bit Timer, 1 16bit PWM, 51 DI, 80 DO	Segger J-Link EDU ¹¹ Rowley CrossStudio ¹²
TI LM3S9D96	18 sFr.	Ethernet MAC, USB-Host, 8 16bit Timer, 8 16bit PWM, 65 GPIO	DK-LM3S9D96 ¹³

Diese Kriterien grenzten die Wahl auf einen ARM Cortex M3 von STMicroelectronics ein. Also wurden das EM-STM3210C Evaluierungsboard von Embest, ein J-Link Programmiergerät von SEGGER und die IDE CrossStudio for ARM von Rowley Associates Ltd ausgewählt.

¹⁰ AVR JTAG ICE MK3 kostet bei Reichelt.de 180 sFr, Abgerufen am 21.02.13

¹¹ Segger J-Link EDU für nicht kommerziellen Gebrauch 70 sFr.

¹² CrossStudio Rowley Associates Ltd für nicht kommerziellen Gebrauch 150 sFr.

¹³ DK-LM3S9D96 bei Digikey.com 415 sFr, Abgerufen am 21.02.13

2 Hauptteil

Das Evaluierungsboard EM-STM3210C besteht aus einem Mikrocontroller von ARM, dem STM32F107VC, Ethernet, USB und RS232 Schnittstellen, einem MicroSD Kartenadapter, einem LCD mit Touchscreen, LEDs und Knöpfen. Es wird mit dem J-Link von Segger über die JTAG-Schnittstelle programmiert und es können bis zu 6 Hardwarebreakpoints gesetzt werden. Die Einbindung in die Entwicklungsumgebung ist vorbildlich, der Code kann in Einzelschritten ausgeführt werden und der aktuelle Speicherverbrauch wird immer angezeigt. Leider sind selbst die Beispielprogramme so komplex, dass die Einarbeitung und Konfiguration der Entwicklungsumgebung sehr viel Zeit beansprucht.

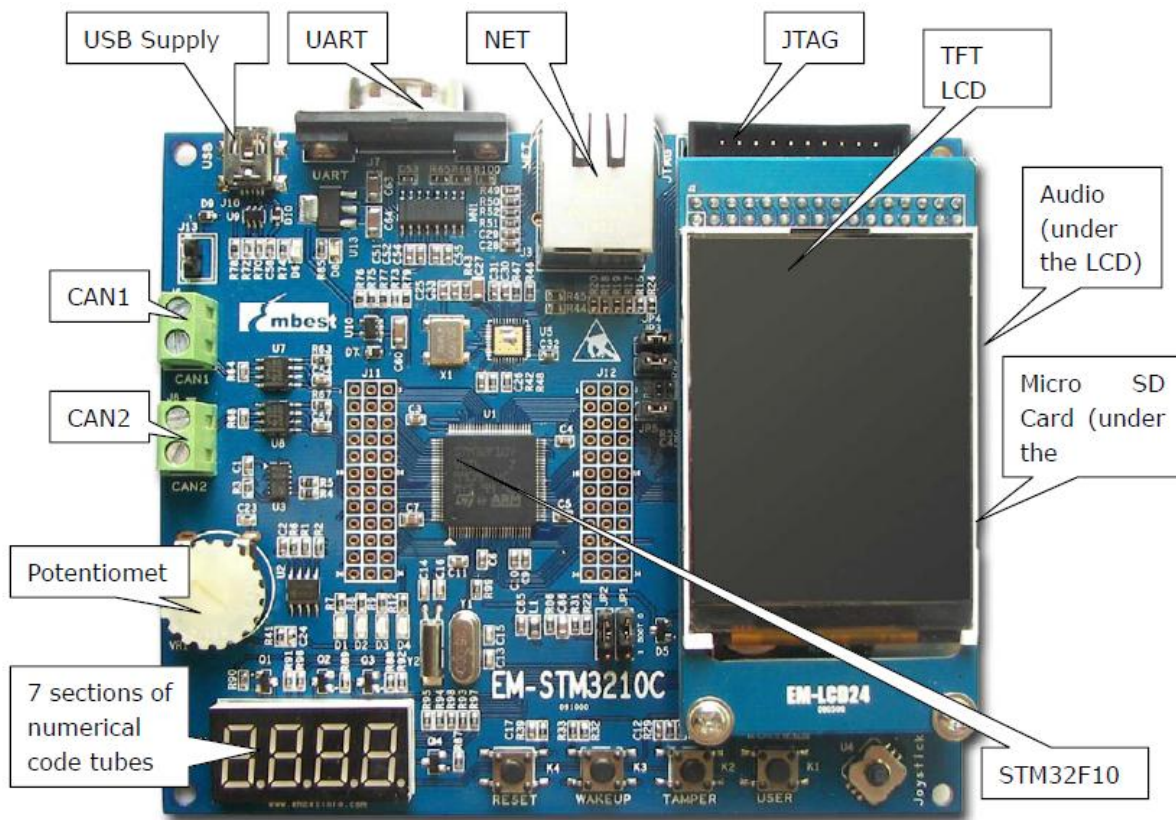


Abbildung 3: Evaluierungsboard von Embest, EM- STM3210C

Das JTAG Programmiergerät J-Link von Segger verbindet den Entwicklungsrechner mittels USB mit der JTAG Schnittstelle des Evaluierungsboards. Es erlaubt die Programmierung und das Debugging in Einzelschritten.

2 Hauptteil



Abbildung 4: JTAG Programmieradapter J-Link EDU

Die Entwicklungsumgebung muss zu Beginn auf den Mikrocontroller und die Bibliothekspfade eingestellt werden, bietet dann aber unter Anderem Funktionen wie Codevervollständigung, fortlaufender Test auf Syntaxfehler und eine Anzeige des Speicherverbrauchs. Sehr angenehm ist auch das Debugging, welches in Einzelschritte ausgeführt werden kann.

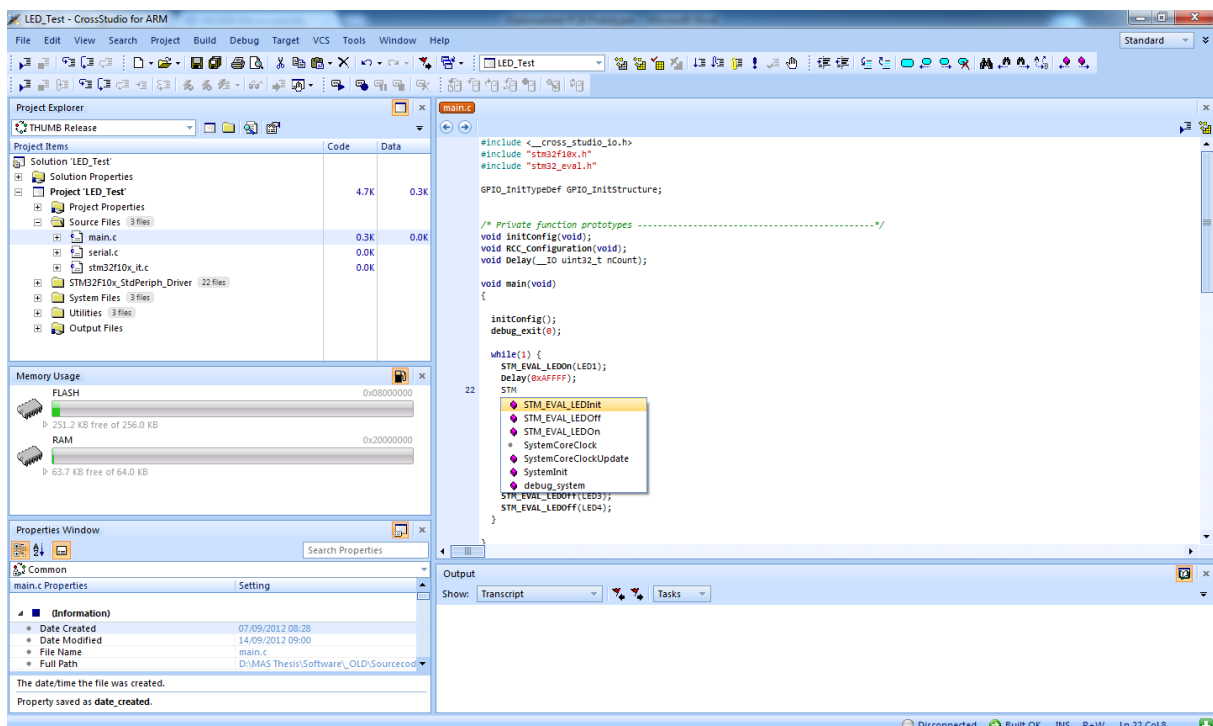


Abbildung 5: CrossStudio for ARM v2.2, Entwicklungsumgebung für ARM Mikrocontroller

2 Hauptteil

Aufgrund der hohen Komplexität und der verbrauchten Zeit für die Einarbeitung, wurde die Idee das System mit einem neuen 32 Bit Mikrocontroller zu realisieren verworfen. Stattdessen wurde auf den bekannten und einfach zugängigen 8 Bit Mikrocontroller ATmega2560 auf einem Arduinoboard¹⁴ (Arduino Mega2560) gewechselt. Ein grosser Vorteil dieser Plattform ist es, dass für die unterstützten Mikrocontroller Bootloader vorhanden sind, welche den Einstieg und Nutzung der vielen Bibliotheken sehr stark vereinfachen. Ausserdem ist es möglich, selber Bibliotheken in C/C++ zu schreiben und einzubinden. Zudem gibt es Erweiterungsmodule, so genannte Shields, welche zum Beispiel eine Ethernetschnittstelle zur Verfügung stellen. Die oftmals stark kritisierte IDE wurde durch Visual Studio 2012 mit dem Plugin Visual Micro ersetzt.

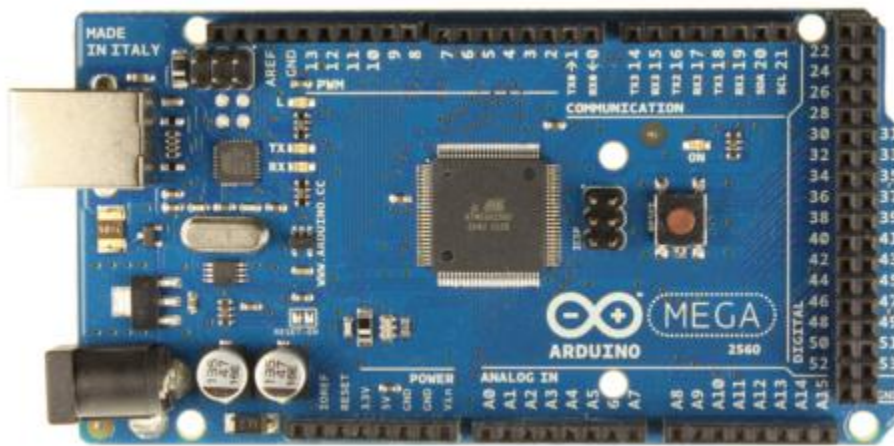


Abbildung 6: Arduino Mega2560 R3, 8 Bit Mikrocontroller

Die Programmierung und das Debugging erfolgen über USB und einen virtuellen COM Port. Dies schränkt vor allem die Debugmöglichkeit sehr stark ein. Der Code kann leider nicht in Einzelschritten ausgeführt werden und es existieren keine Breakpoints. Als Entwickler muss man sich damit begnügen, Variabelwerte und Nachrichten auf die Konsole zu schreiben und kleine Testprogramme zu schreiben um den Code zu prüfen.

¹⁴ Das Arduinoprojekt ist eine opensource Elektronik-Entwicklungsplattform www.arduino.cc Abgerufen am 21.02.13

2 Hauptteil

Der Einsatz von Visual Studio bringt trotzdem grosse Vorteile, wie in CrossStudio wird der Code ergänzt und es werden Syntaxfehler beim Tippen angezeigt. Ausserdem kann Quellcode sehr einfach verfolgt werden, ein Klick auf eine Methode und es kann zur Implementierung gesprungen werden.

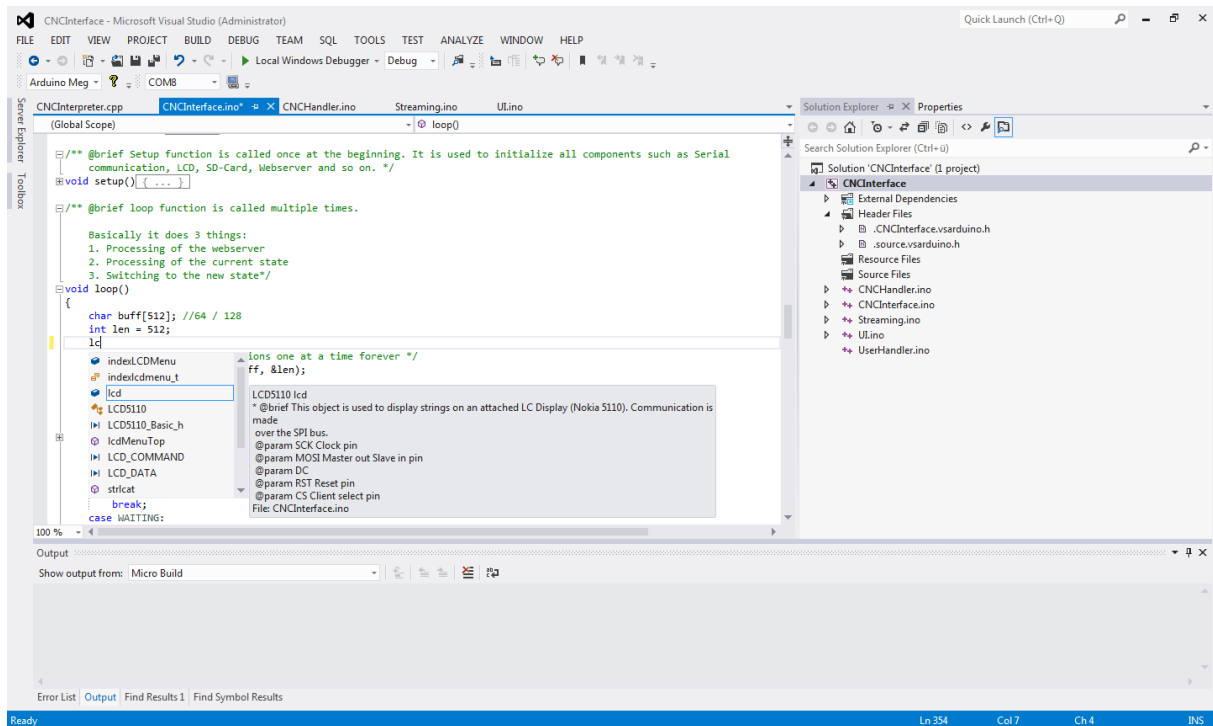


Abbildung 7: Visual Studio 2012 mit Arduino

Mittlerweile (Stand Januar 2013) ist auch ein ARM Cortex M3 Board verfügbar (Arduino Due), welches eventuell für Nachfolgeprojekte eingesetzt werden kann.

2 Hauptteil

2.3.3 ENTWICKLUNG DES AUTONOMEN SYSTEMS

Die CNC Fräsmaschine wird über einen RS232 Port von einem Rechner aus gesteuert. Auf diesem Rechner müssen Windows XP und ein Steuerprogramm RemoteWIN installiert sein. Dieser Rechner soll durch einen Mikrocontroller mit eigenem Steuerprogramm ersetzt werden. Dieser Mikrocontroller hat verschiedene Schnittstellen zu seiner Umwelt.

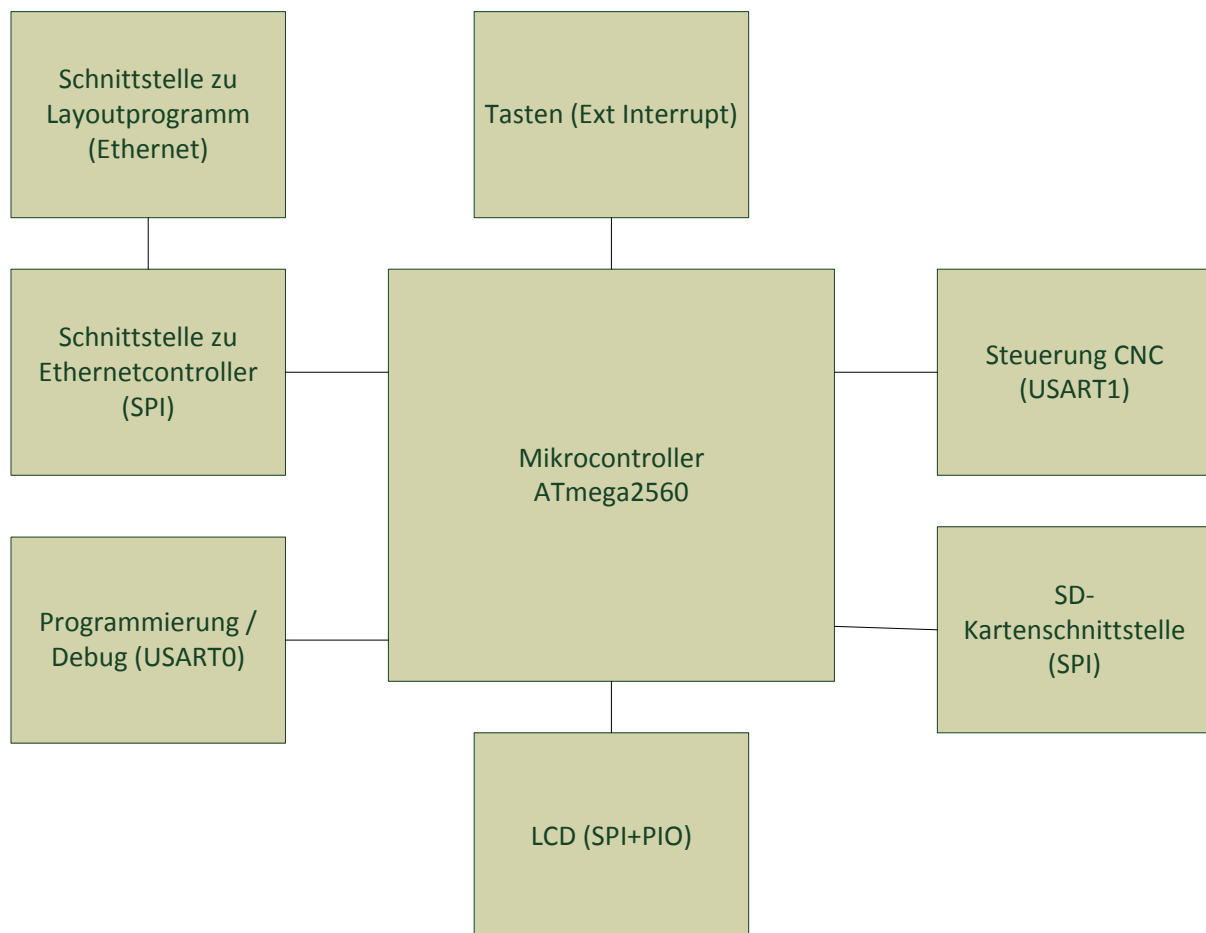


Abbildung 8: Blockdiagramm Systemübersicht

Tasten und LCD dienen dabei der direkten Steuerung des Systems vor Ort (Auswahl und Start eines Fräsprogrammes usw.). Mittels Ethernetcontroller stellt er ein Webinterface zur Verfügung. Damit kann der Benutzer über ein lokales Netzwerk Fräsdaten auf den Mikrocontroller hochladen. Zur Datenspeicherung (Webseiten, Einstellungen und Fräsdaten) wird dabei eine SD-Karte genutzt. Die Fräsmaschine wird über eine USART (RS232) Schnittstelle gesteuert. Die zweite USART-Schnittstelle wird als virtueller COM Port über einen USB 2.0-B-Stecker zur Verfügung gestellt und dient der Programmierung und dem Debugging.

2 Hauptteil

Der Datenaustausch läuft vom Browser des Benutzers zum Webserver, dieser lädt die entsprechende Webseite von der SD-Karte und sendet sie zum Browser des Benutzers zurück.

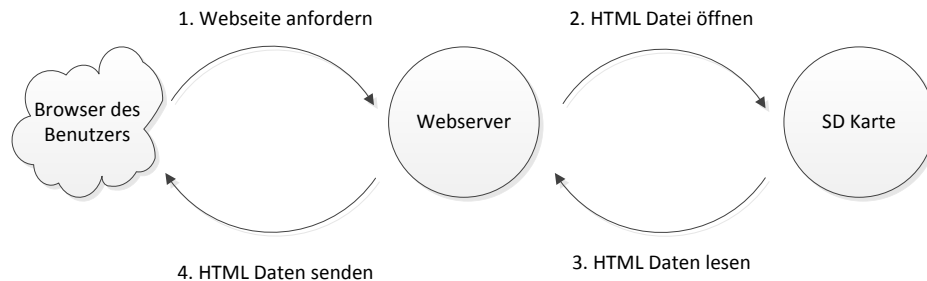


Abbildung 9: Datenfluss beim Öffnen einer Webseite

Lädt ein Benutzer CNC Daten hoch, leitet Sie der Webserver an die SD-Karte weiter und meldet dem Benutzer, wenn die Speicherung erfolgreich war.

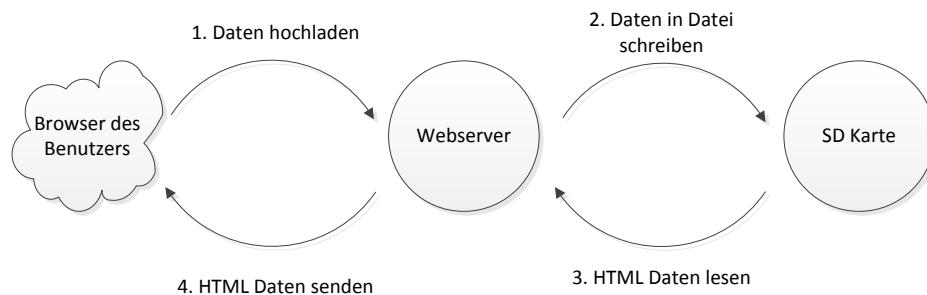


Abbildung 10: Datenfluss beim Hochladen von Daten

Wird das System lokal über die Knöpfe gesteuert, werden die Datendateien von der SD-Karte gelesen und auf dem LCD angezeigt.

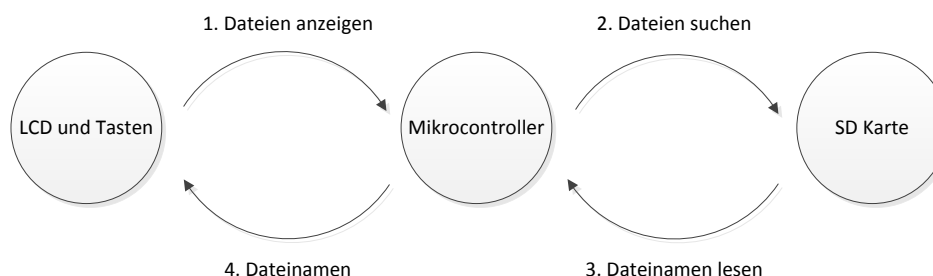


Abbildung 11: Datenfluss bei der Dateiauswahl

2 Hauptteil

Wählt der Benutzer eine Datei zum Fräsen aus, werden die Daten von der SD-Karte geladen, in Steuersignale übersetzt und an die serielle Schnittstelle (USART1) gesendet. Kommt ein Bestätigungszeichen von der Schnittstelle zurück, wird der nächste Befehl gesendet.

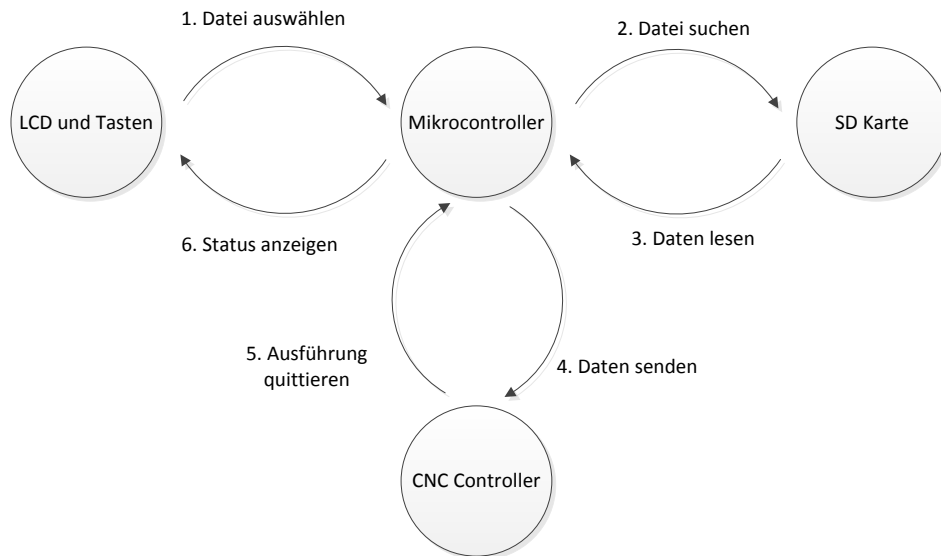


Abbildung 12: Datenfluss beim Fräsen

Als Anzeigeelement wird ein Nokia 5110 LCD mit Breakoutboard von Sparkfun¹⁵ eingesetzt. Es ist ein 45x45 mm grosses, monochromes Grafik-LCD mit zugehörigem CMOS Treiber. Neben der Spannungsversorgung von 5 V werden fünf Logiksignale für den SPI-Bus, für Daten oder Befehlsmodus und den Reset des Controllers benötigt: SCK, MOSI, CS, D/C und RST.

¹⁵ Sparkfun Electronics, Online retail store aus den USA. Webseite: www.sparkfun.com abgerufen am 26.02.13

2 Hauptteil

Der geplante Ablauf aus Sicht des Bedieners soll folgendermassen aussehen:

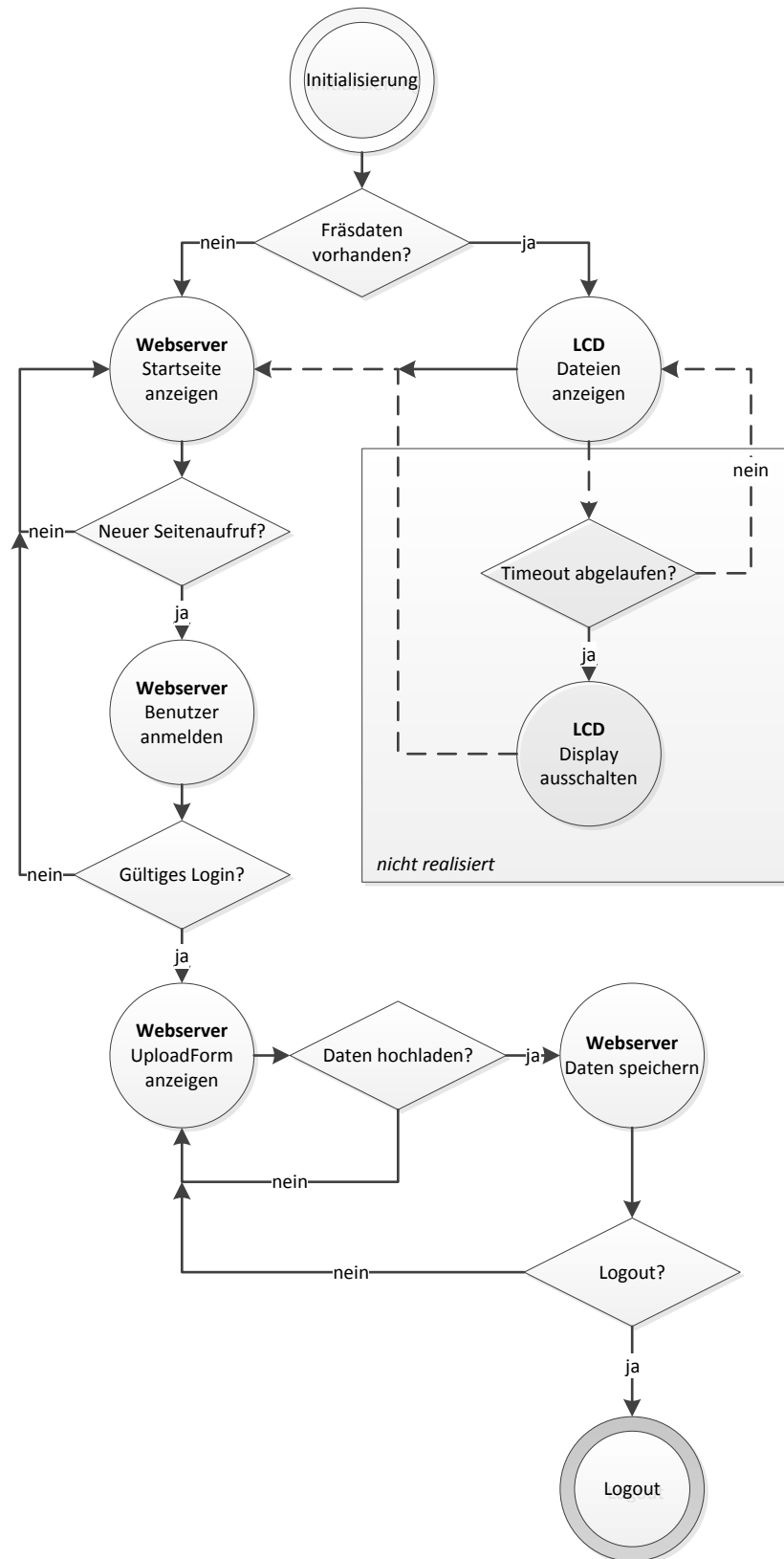


Abbildung 13: Geplanter Ablauf aus Sicht des Bedieners

2 Hauptteil

Das zu entwickelnde autonome System ist in Hardware eingebettet, zur Übersicht folgt ein Blockdiagramm mit dem Gesamtsystem:

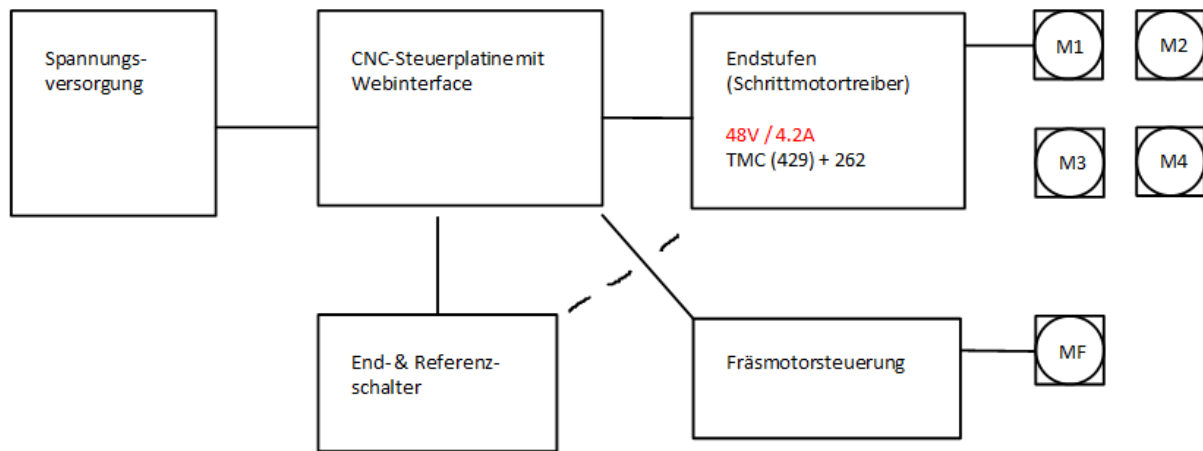


Abbildung 14: Übersicht Gesamtsystem mit Motorsteuerungen und Spannungsversorgung

Use Case 1: Einschalten der Maschine

Die CNC Maschine muss vor dem Fräsvorgang vom Benutzer eingeschaltet werden. Sie muss anschliessend eine Referenzfahrt durchführen. Sobald sie bereit ist, wird auf dem Anzeigeelement eine Meldung ausgegeben. Jetzt können über den Webserver Daten hochgeladen oder eine bereits hochgeladene Datei gefräst werden.

Use Case 2: Benutzen des Webserver

Der Webserver lässt gleichzeitig nur einen Benutzer zu. Der Benutzer muss sich mit einem Passwort einloggen. Die Datei, welche im Upload-Formular ausgewählt wurde, muss auf ihre Grösse überprüft werden. Ausserdem müssen die Daten während dem hochladen überprüft werden. Die gespeicherten Daten sollen dem Benutzer angezeigt werden. Der Datensatz wird auf der SD-Karte gespeichert.

2 Hauptteil

Use Case 3: Starten des Fräsvorganges

Bevor der Fräsvorgang starten kann, muss der Benutzer zur CNC Maschine gehen und eine neue Platine einlegen. Ausserdem muss er den eingespannten Fräser überprüfen und danach die gewünschte Datei mit der Benutzerschnittstelle an der CNC Maschine auswählen. Nachdem die Maschine den Werkstücknullpunkt eingestellt hat, muss der Benutzer den Fräsmotor starten und über die Benutzerschnittstelle quittieren.

Use Case 4: Bohren der Platinenlöcher

Sollen nach dem Fräsvorgang die Platinenlöcher für die Bauteile gebohrt werden, muss der Benutzer das Werkzeug manuell wechseln. Danach muss er die zuvor hochgeladene Bohrdatei auswählen. Nachdem die Maschine den Werkstücknullpunkt eingestellt hat, muss der Benutzer den Fräsmotor starten und über die Benutzerschnittstelle quittieren.

Use Case 5: Abschliessen der Platinenbearbeitung

Sind der Fräs- und Bohrvorgang abgeschlossen wird der Fräskopf von der Platinenoberfläche entfernt und die Platine in Richtung Tür bewegt. Der Benutzer muss die Platine vom Schmutz befreien und anschliessend mit einem Schleifpapier glätten.

2 Hauptteil

2.3.4 ELEKTRONIKENTWICKLUNG DER SCHNITTSTELLE ZUR FRÄSMASCHINE

Um die Schrittmotoren anzusteuern wurden Schrittmotorleistungsstufen und ein 3-Achs Schrittmotorcontroller entworfen. Aufgrund der knappen Zeit wurde das Layout allerdings nicht fertiggestellt, sondern die vorhandene Schrittmotorsteuerung der CNC Maschine übernommen und über RS232 ferngesteuert.

Die geplante Leistungsstufe für einen Schrittmotor besteht aus einem Schrittmotortreiber IC von Trinamic (TMC-262) kombiniert mit einer externen H-Brücke, um dem Motor bis zu 8 Ampère Strom zur Verfügung zu stellen.

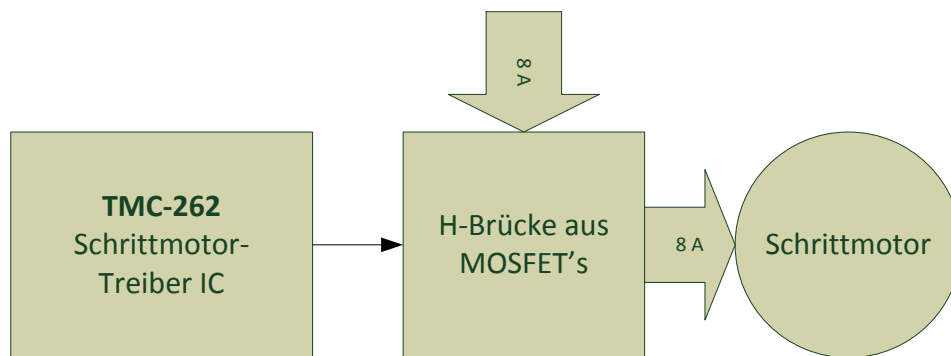


Abbildung 15: Schrittmotor Leistungsstufe

Je drei dieser Endstufen werden von einem Trinamic Motorcontroller (TMC 429) aus gesteuert. Zur Konfiguration können die Schrittmotortreiber über SPI angesprochen werden, im Betrieb werden nur noch Schritt- und Richtungssignale vom Motorcontroller zum Treiber benötigt. Der Motorcontroller wird mit SPI an den Mikrocontroller angeschlossen. Ausserdem werden die Testsignale der einzelnen IC's zum Mikrocontroller zurückgeführt und alle Signale mit Stiftleisten verbunden. So kann jedes Signal mit einem externen Messgerät überprüft werden. Die Schaltpläne befinden sich im Anhang.

2 Hauptteil

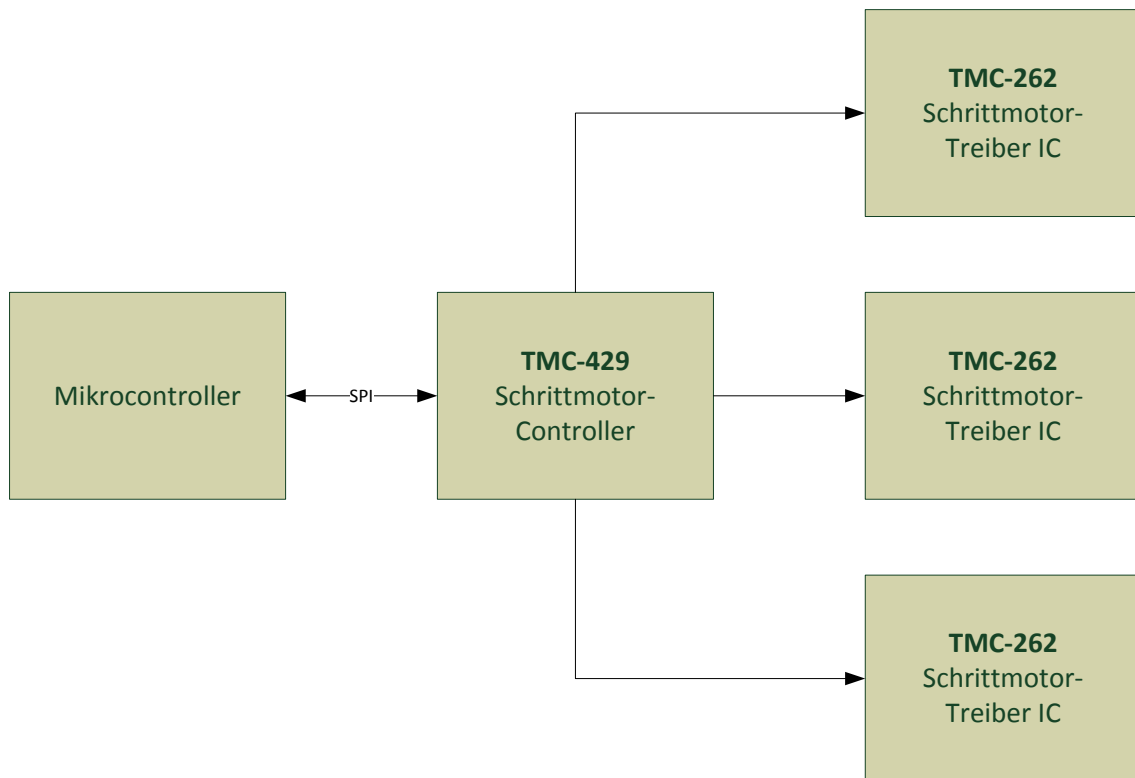


Abbildung 16: Schrittmotorsteuerung

Zum Verbinden von Mikrocontroller mit der Schrittmotorsteuerung wurde eine Erweiterungsplatine für den Arduino MEGA 2560 entwickelt. Auf dieser Platine befinden sich ein MAX232 Pegelwandler mit RS232 Buchse zur Anbindung an die CNC Maschine, Anschlüsse für Tasten und das LCD und ein Anschluss für ein analoges Eingangssignal. Die Erweiterungsplatine wird über Stiftheuten mit dem Ethernetshield und dem Arduinoboard verbunden.

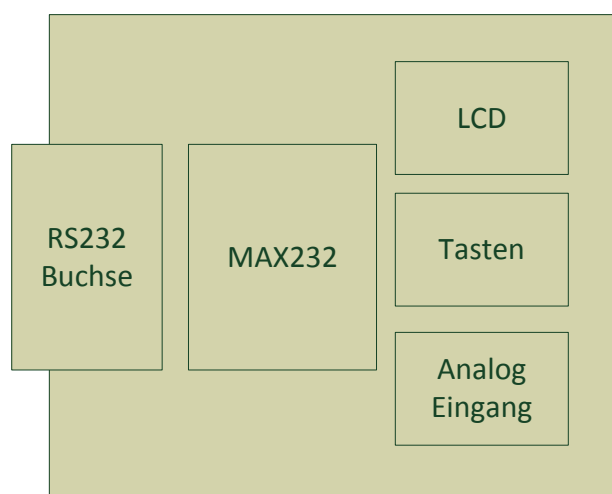


Abbildung 17: Erweiterungsplatine für den Arduino MEGA2560

2 Hauptteil

2.3.5 PROGRAMMIERUNG DER FRÄSMASCHINENSTEUERUNG

Das System der Fräsmaschinensteuerung verwendet neben den Arduino Standardbibliotheken drei zusätzliche Bibliotheken (WebDuino, LCD5110_Basic und CNCInterpreter).

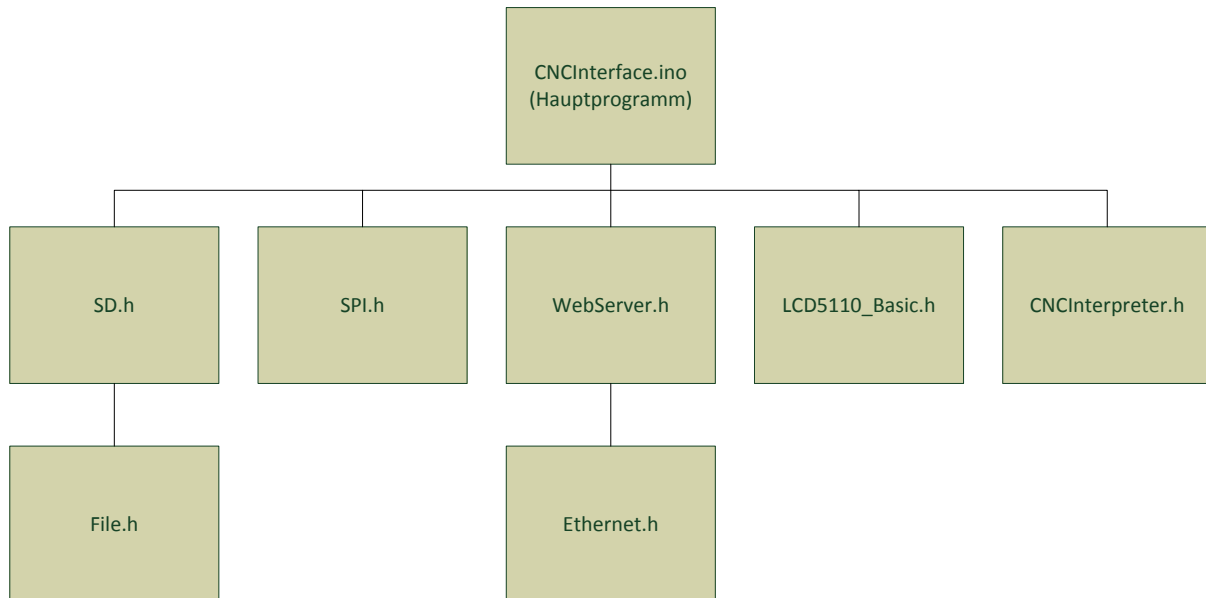


Abbildung 18: Verwendete Bibliotheken

Ausserdem sind einige Hilfsfunktionen aus dem Hauptprogramm in verschiedene Dateien ausgegliedert worden.

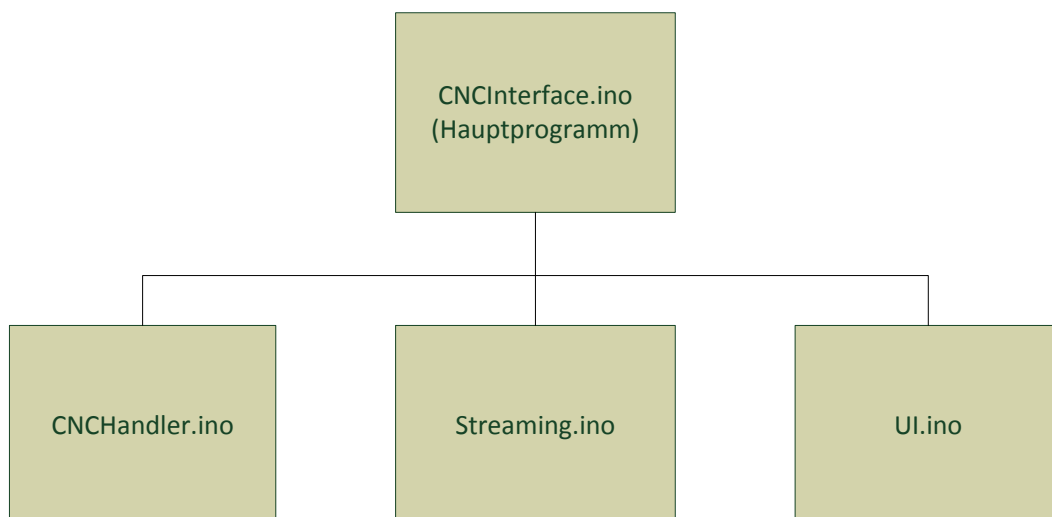


Abbildung 19: Unterteilung in Hilfsdateien

Die Datei `CNCHandler` beinhaltet alle Methoden zur Steuerung der Fräsmaschine, dazu gehört neben der Initialisierung auch der gesamte Fräsablauf.

2 Hauptteil

In der Datei Streaming sind alle Methoden zum Senden der Daten auf die SD-Karte und von der SD-Karte auf die RS232 Schnittstelle gruppiert.

UI bedeutet „user interface“, weshalb hier alle Methoden zur Darstellung und Steuerung der Benutzerschnittstelle vereint sind.

Das Hauptprogramm initialisiert die einzelnen Bibliotheken und geht dann in einen Zyklus aus Webserver aktualisieren, Eingänge verarbeiten und je nach Zustand Befehle an die Fräsmaschine senden über.

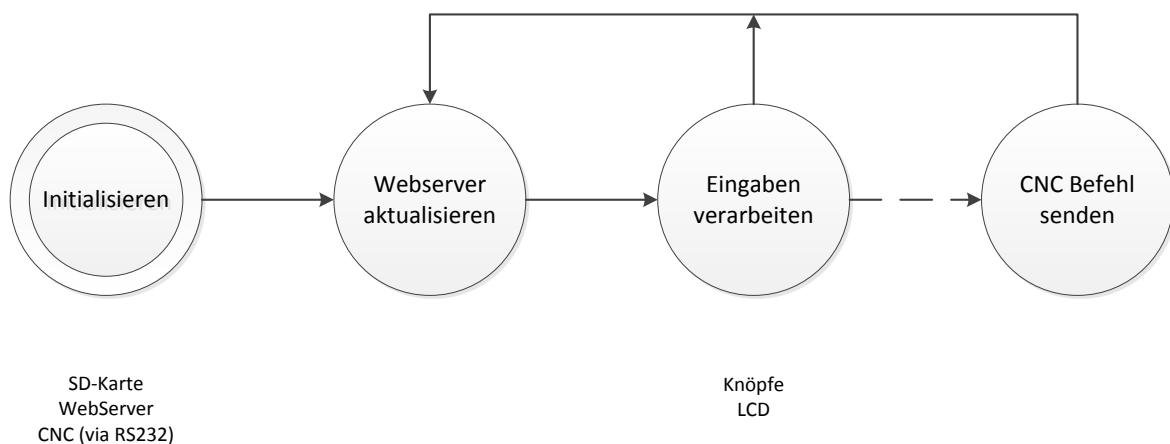


Abbildung 20: Schematischer Ablauf des Hauptprogrammes

Ausserdem sind die Funktionen, welche bei Aufruf der einzelnen Webseiten ablaufen, hier definiert.

Zur Kommunikation mit der SD-Karte über SPI wird die offizielle Arduino SD-Klasse verwendet. Sie erlaubt den Einsatz von FAT16 oder FAT32 Dateisystemen auf Standard oder SDHC Karten. Allerdings werden Dateinamen nur nach dem „8.3 DOS-Schema“ (8 Buchstaben Dateiname + 3 Buchstaben Dateityp) unterstützt. Das kleine Dateisystem ermöglicht die Verwaltung von Unterverzeichnissen, das Öffnen und Löschen von Dateien.

2 Hauptteil

Für die Funktionalitäten des Webserver wird die Webduino¹⁶ Bibliothek in der Version 1.7 verwendet. Sie erlaubt die Verarbeitung von URL Daten aus GET und POST Aufrufen. Neben den HTTP Methoden „GET“ und „POST“ werden auch „HEAD“, „PUT“, „PATCH“ und „DELETE“ unterstützt, in diesem Projekt jedoch nicht eingesetzt. Die HTTP Basic Authentifizierung wird verwendet, um den Webserver mit einem Passwort zu sichern.

Zur Programmierung des LCD's wird die Bibliothek LCD5110_Basic¹⁷ von Henning Karlsen in der Version 1.1 eingesetzt. Dabei wird ein Objekt z.B. „lcd“ mit den Anschlusspins als Parameter erzeugt. Anschliessend kann mit „lcd.print“ auf das LCD geschrieben werden. Da die verwendete Schrift „SmallFonts“ 8 Pixel hoch ist, können als Zeilen nur Vielfache von 8 verwendet werden (0, 8, 16, 24, 32, 40). Ausserdem werden noch Methoden zum Löschen des Bildschirms, zum Initialisieren des LCDs und zur Definition der Schrift bereitgestellt.

Zur Kommunikation mit der CNC Maschine über RS232 wird eine USART Schnittstelle (USART1) eingesetzt. Diese kann über das Serial1 Objekt von Arduino angesprochen werden.

Für die Fernsteuerung der Schrittmotorsteuerung der CNC Maschine über RS232 wird eine spezielle Befehlssyntax benötigt. Sie ist wie folgt aufgebaut:

Tabelle 7: Befehlssyntax CNC Controller

Datenzeichen	Gerätenummer	Befehl	Sendezeichen
@	0	7	<CR>

Das Datenzeichen und die Gerätenummer sind vor jedem Befehl notwendig, bleiben aber immer konstant. Der Befehl muss mit einem „carriage return“ Zeichen abgeschlossen werden.

¹⁶ Arduino Webserverbibliothek Quelle Github: <https://github.com/sirleech/Webduino>, Abgerufen am 26.02.13

¹⁷ Nokia 5110 LCD Bibliothek für Arduino: <http://www.henningkarlsen.com/electronics> Abgerufen am 26.02.13

2 Hauptteil

Zu Beginn der Fernsteuerung ist ein Initialisierungsablauf aus Definition der Achsenzahl und anschliessender Referenzfahrt nötig:

Tabelle 8: Initialisierungsablauf CNC Controller

Achsenanzahl auf 3 setzen (X/Y/Z)	@07
Referenzfahrt auf X=0, Y=0, Z=0	@0r7

Danach können dem Controller weitere Befehle zur Bewegung (Absolut oder Relativ), zum Setzen eines Werkstücknullpunktes usw. gesendet werden. Jeder Befehl wird vom CNC Controller mit einem Fehlercode quittiert. Dabei bedeutet das Quittierungszeichen „0“, dass kein Fehler aufgetreten ist.

Die weiteren Befehle zur ISEL Steuerung können im Programmierhandbuch (IMC4-Programmierung Deutsch) nachgelesen werden.

Die Übersetzung von CNC G-Code¹⁸ in die Befehlssyntax der CNC Maschine wird von einer selbst erstellten Klasse (CNCInterpreter) erledigt. Dabei wird die eingehende Codezeile geprüft, die Bewegungsdistanzen von Millimeter in Schritte umgerechnet und in der benötigten Befehlssyntax (siehe oben) zurückgegeben. Nach dem Erzeugen eines Objektes der Klasse CNCInterpreter z.B. „cnc“ kann die Methode „translate“ zur Übersetzung verwendet werden.

¹⁸ ISO 6983, Erklärung auf Wikipedia <http://en.wikipedia.org/wiki/G-code> , abgerufen am 26.02.13

2 Hauptteil

2.3.6 AUTOMATISIERTE MESSUNG DER WERKZEUGLÄNGE

Zur Messung der Werkzeuglänge wird nicht wie sonst üblich mit dem Werkzeug auf einen Drucktaster gefahren. Stattdessen wird die Leitfähigkeit der zu bearbeitenden Kupferplatine ausgenutzt um zwischen Platinenoberfläche und Fräswerkzeug einen elektrischen Kreis zu schliessen. Dazu wird der Fräskopf elektrisch auf einen analogen Eingang des Mikrocontrollers geführt. Dieser Eingang wird per Pullup auf 5V gezogen. Schliesst sich der Kreis aus Fräskopf und geerdeter Platinenoberfläche, sinkt die Spannung am Analogeingang gegen 0V.

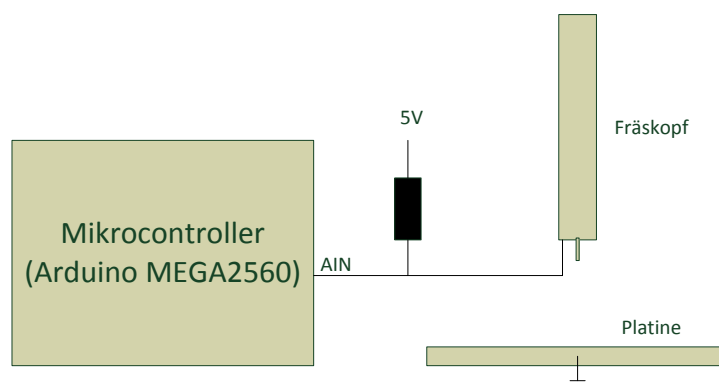


Abbildung 21: Bestimmung der Werkzeuglänge mit „virtuellem“ Schalter

Damit dieser Messvorgang nicht ewig lange dauert, fährt der Fräskopf im Eilgang auf eine Distanz von 1 cm über die Platinenoberfläche und nähert sich anschliessend in Schritten von 0.05 mm (entspricht 10 Einzelschritten) der Platine, bis der elektrische Kontakt entsteht.

2 Hauptteil

2.3.7 IMPLEMENTIERUNG EINER BENUTZERSCHNITTSTELLE

Das System hat zwei Benutzerschnittstellen. Zum einen den Webserver zum Hochladen von Bohr- und Fräsdaten, zum anderen das LCD mit den Tasten um die gewünschte Datei an die CNC Maschine zu senden und den Bearbeitungsvorgang zu starten.

Der Webserver stellt insgesamt vier HTML Seiten zum Hochladen von Daten über das Netzwerk zur Verfügung. Er besitzt eine statische IP Adresse und kann innerhalb des EDU Netzes angesprochen werden. Die IP-Adresse wird übrigens während der Referenzfahrt der Maschine auf dem LC Display angezeigt.

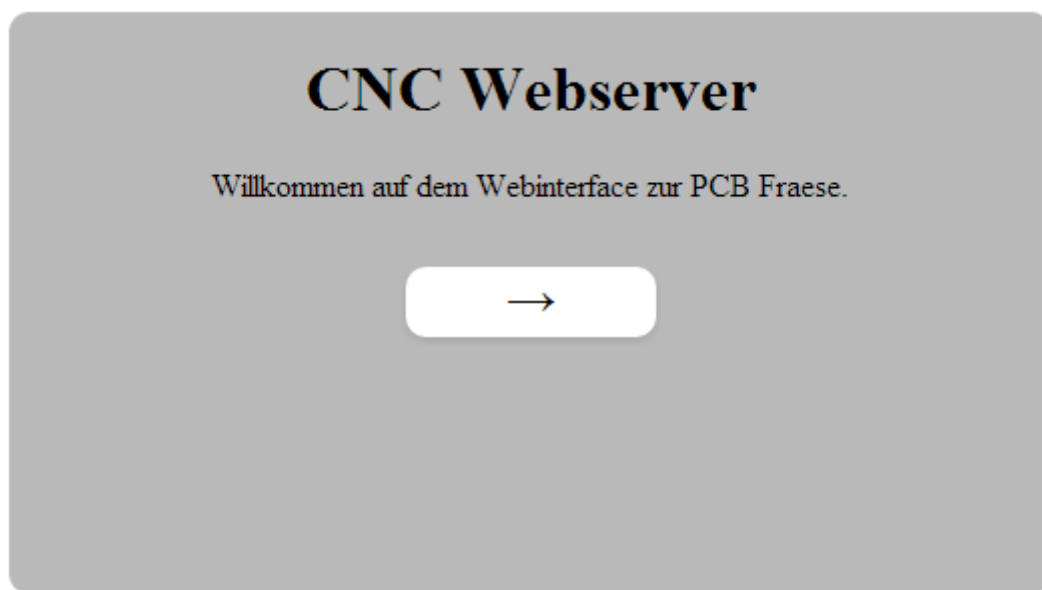
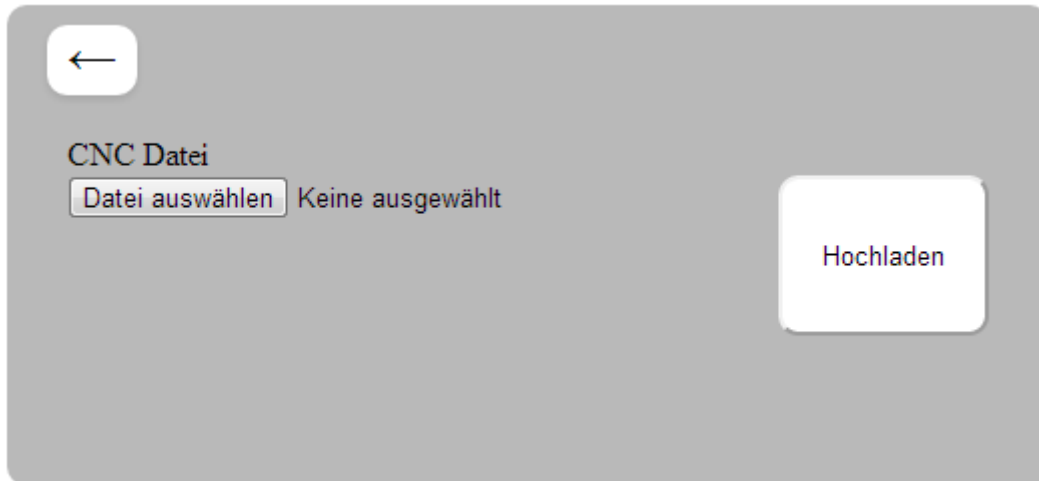


Abbildung 22: Startseite des Webservers (index.htm)

Der Webserver ist durch eine einfache HTTP-Basic Authentication vor unerwünschtem Zugriff geschützt. Ausserdem kann während einem Fräsvorgang keine neue Datei hochgeladen werden. Die HTML Dateien befinden sich auf der SD-Karte im Unterverzeichnis /html. Sie sind in HTML 5 und CSS 3 geschrieben, das Upload Formular benötigt zusätzlich Java Script.

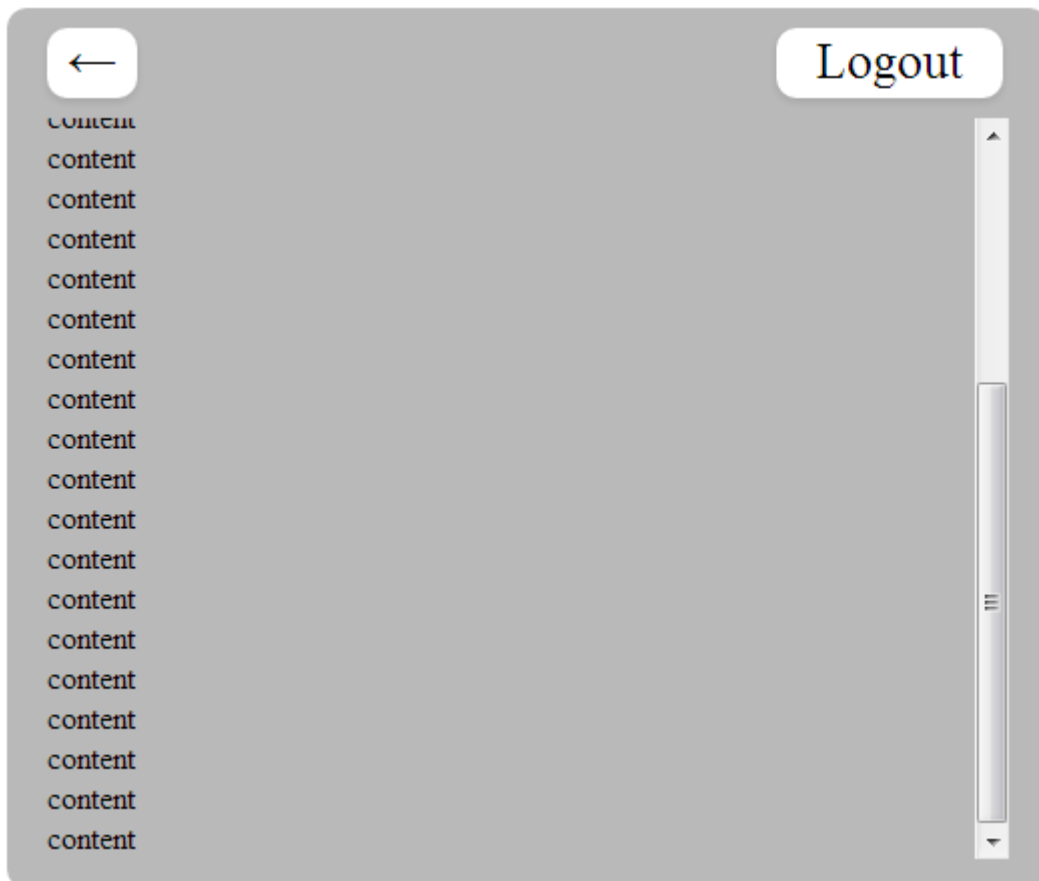
2 Hauptteil



A screenshot of a web form for uploading a CNC file. At the top left is a back arrow button. Below it, the text "CNC Datei" is followed by a file selection button labeled "Datei auswählen" and the text "Keine ausgewählt". On the right side of the form is a large button labeled "Hochladen".

Abbildung 23: Upload Formular zum Hochladen einer CNC Datei (upload.htm)

Die Datenübermittlung des Formulars ist über einen POST Aufruf realisiert.



A screenshot of a web page displaying data received from a POST request. At the top left is a back arrow button. At the top right is a button labeled "Logout". The main area of the page contains a list of 20 lines, each starting with the word "content". A vertical scrollbar is located on the right side of the list.

Abbildung 24: Zeilenweise Darstellung der Übermittelten Daten

2 Hauptteil

Die lokale Benutzerschnittstelle besteht aus einem LCD und zwei Bedientasten. Mit der Pfeiltaste kann durch einzelne Einträge auf dem Display gewechselt, mit der OK-Taste der aktive Eintrag ausgewählt werden. Die Menüstruktur ist dabei sehr einfach gehalten, im Hauptmenü wird zwischen CNC Daten und einem Setup Untermenü ausgewählt.

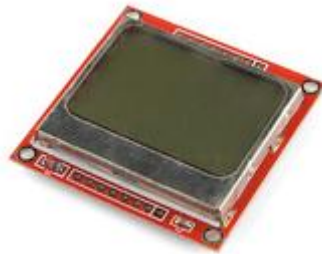
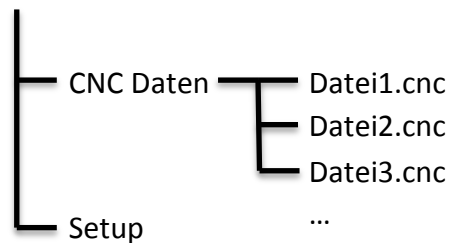


Abbildung 25: LC Display eines Nokia 5110

Im CNC Datenmenü werden alle auf der SD-Karte gespeicherten Dateien aufgelistet. Mit der Pfeiltaste kann anschliessend durch die Dateien durchgewechselt werden. Ein Druck auf die OK-Taste startet den Fräsvorgang für die aktuell angezeigte Datei. Die benötigten Tastenbestätigungen während dem Fräsvorgang werden auf dem Display angezeigt.



2 Hauptteil

2.4 LÖSUNG

Die fertig entwickelte Lösung besteht aus einem Arduino Mega2560 Rev 3, mit einem Ethernet Shield und dem selber gefertigten CNC Shield. Die lokale Benutzerschnittstelle besteht aus dem Nokia 5110 LCD und drei Tasten (OK, Pfeiltaste und Reset). Ein analog Signal führt zum Fräskopf für die Messung der Werkzeuglänge, die CNC Maschine wird über RS232 gesteuert. Sowohl die CNC Daten, als auch die HTML-Seiten werden auf der MicroSD-Karte abgespeichert. Das System ist per Ethernet ans Netzwerk der FHNW gehängt (EDU-Netz) und erlaubt somit den Zugriff von entfernten Orten im lokalen Netz oder von extern (via Anyconnect). Die CNC Fräse wurde nur minimal verändert, um die Messung der Werkzeuglänge zu ermöglichen.

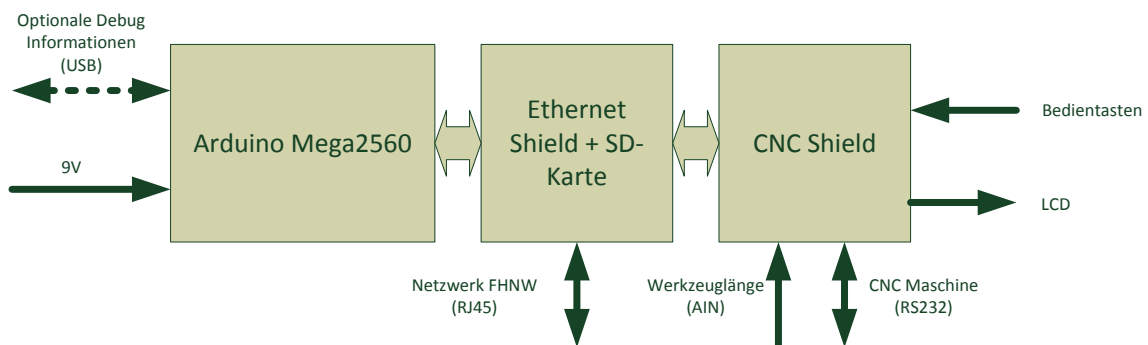


Abbildung 27: Komponentenübersicht des entwickelten Systems

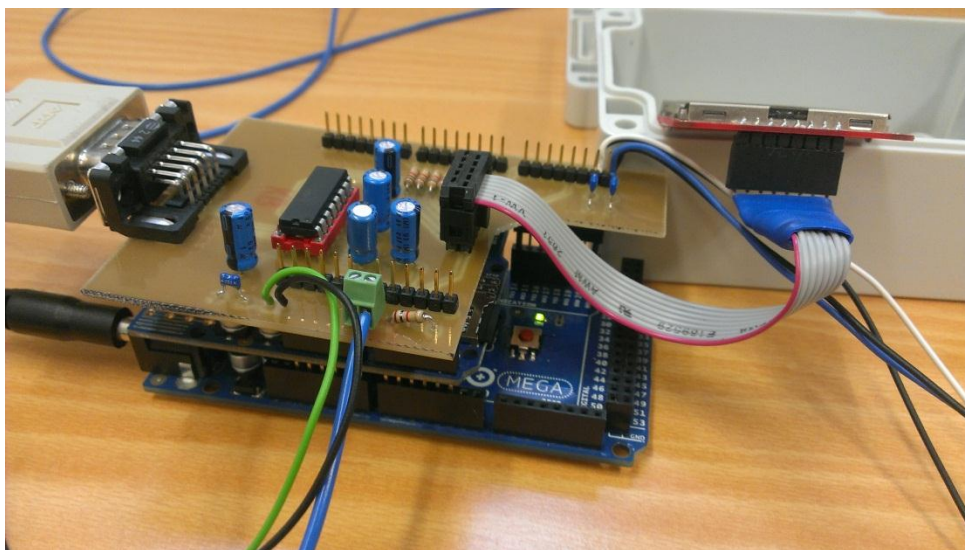


Abbildung 26: Fotografie der CNC Schnittstelle

3 SCHLUSS

3.1 ZUSAMMENFASSUNG

Es konnte gezeigt werden, dass sich auch mit einem vergleichsweise einfachen Mikrocontroller, ohne Betriebssystem und mit eingeschränkten Ressourcen, relativ komplexe und produktiv einsetzbare Geräte entwickeln lassen. Die vielen verfügbaren und einfach zu verwendenden Arduino Bibliotheken ermöglichen ein rasches Vorankommen des Projektes. Man muss sich nicht auf die Registerebene des Mikrocontrollers begeben um grundlegende Funktionen zu programmieren. Dennoch ermöglicht die Plattform das problemlose Einfügen von eigenen C oder C++ Bibliotheken, oder das Bitweise manipulieren von Registern. Durch die Steuerung der CNC Maschine über RS232 musste die CNC nur minimal abgeändert werden und könnte jederzeit auch wieder mit der Originalsoftware auf einem PC gesteuert werden.

3.2 BEWERTUNG

Die Programmierung von Arduinoboards unter Visual Studio bietet einige Annehmlichkeiten wie laufende Syntaxüberprüfung, Codevervollständigung und inkrementelle, also schnelle Builds. Dennoch ist das Debugging sehr mühsam: Der Code kann nicht Schritt für Schritt ausgeführt werden und es können keine Breakpoints gesetzt werden. Das Debugging beschränkt sich also auf die Ausgabe auf die Konsole und auf den Gebrauch von selbst erstellten Testprogrammen. Da es sich beim ATmega2560 um einen 8 Bit Mikrocontroller handelt, kann der Code auch nicht ohne weiteres auf dem PC (mit Breakpoints) getestet werden. Ausserdem ist der Abstraktionslevel dank Arduino so hoch, dass in einzelnen Fällen sehr lange in den unteren Treiberschichten gesucht werden muss, um eine Einstellung zu ändern und den Fehler somit zu lösen.

Schluss

3.3 AUSBLICK

Aktuell unterstützt der Webserver nur Datendateien mit einer maximalen Grösse von 48 kB. Sind die Dateien grösser, läuft der TCP Eingangspuffer des Arduino Mega2560 voll und der Webserver hängt sich auf. Möglicherweise kann auf der Treiberebene (Ethernet.h) der Puffer geändert werden. Mit der Verfügbarkeit des Arduino Due wäre auch ein Einsatz eines 32 Bit Cortex-M3 Prozessors mit einem Vielfachen an Leistung denkbar.

Die Platinenhalterung ist momentan durch das Festschrauben der Platine auf einem Holzbrettchen sehr provisorisch gelöst. Denkbar wäre eine Art Nut oder eine festschraubbare Leiste am Rand, um die Montage zu vereinfachen.

Die Schmutzabsaugung ist ebenfalls noch nicht realisiert. Benötigt wird auf jedenfall eine Düse, welche sehr nahe am Fräser platziert werden kann.

4 ANHANG

4.1 ELEKTRONIK

4.1.1 SCHRITTMOTORSTEUERUNG

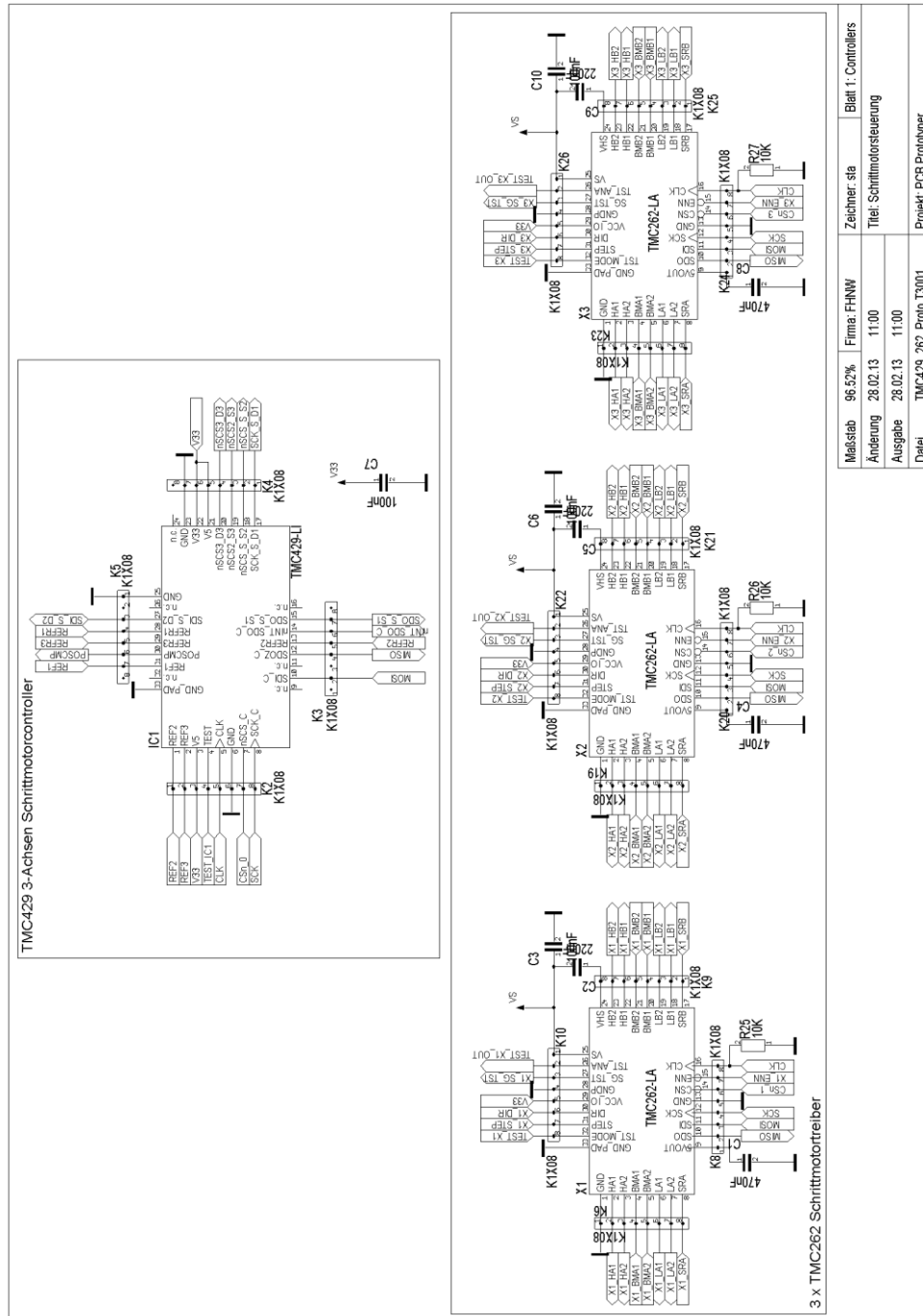


Abbildung 28: Schrittmotorcontroller

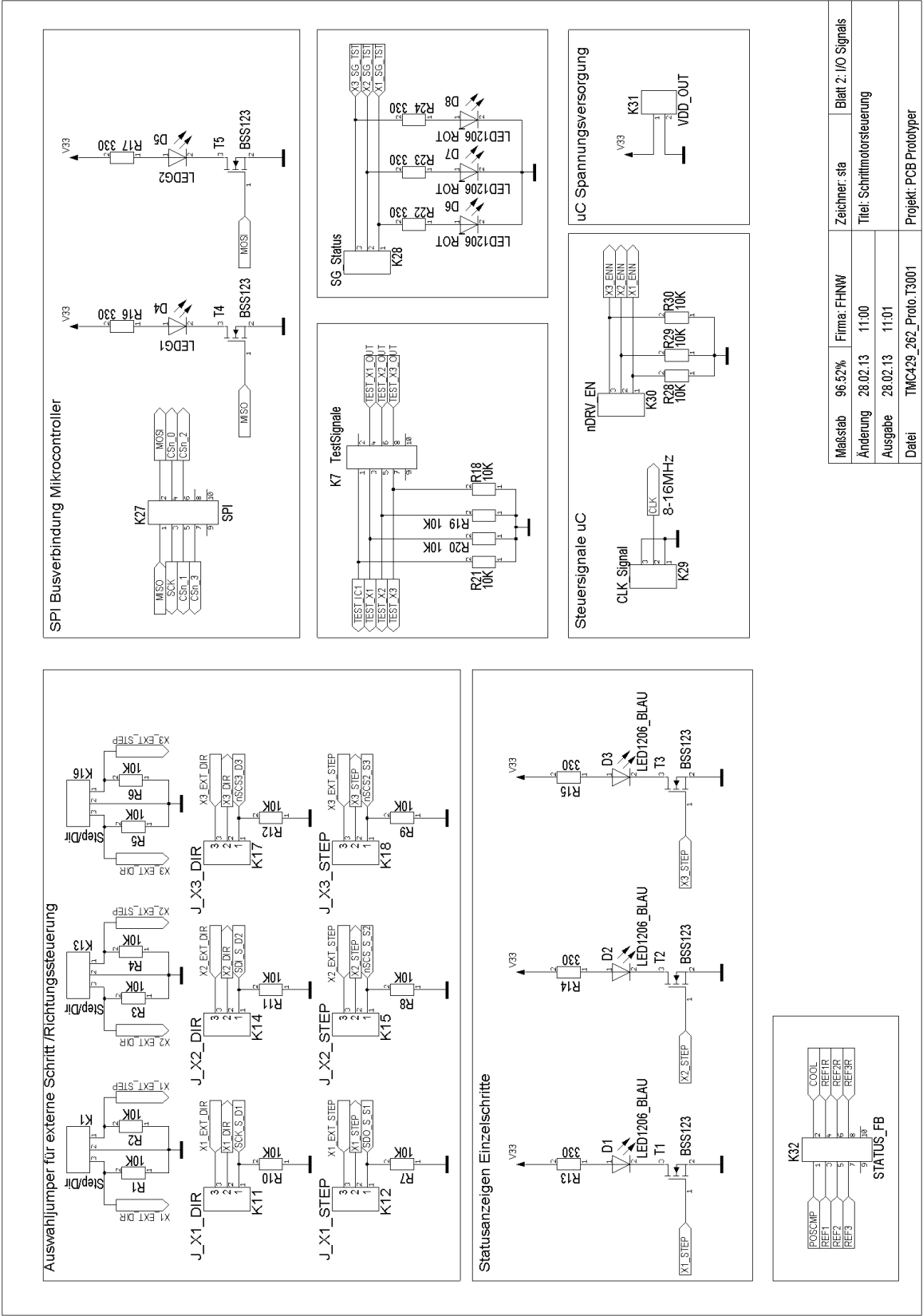
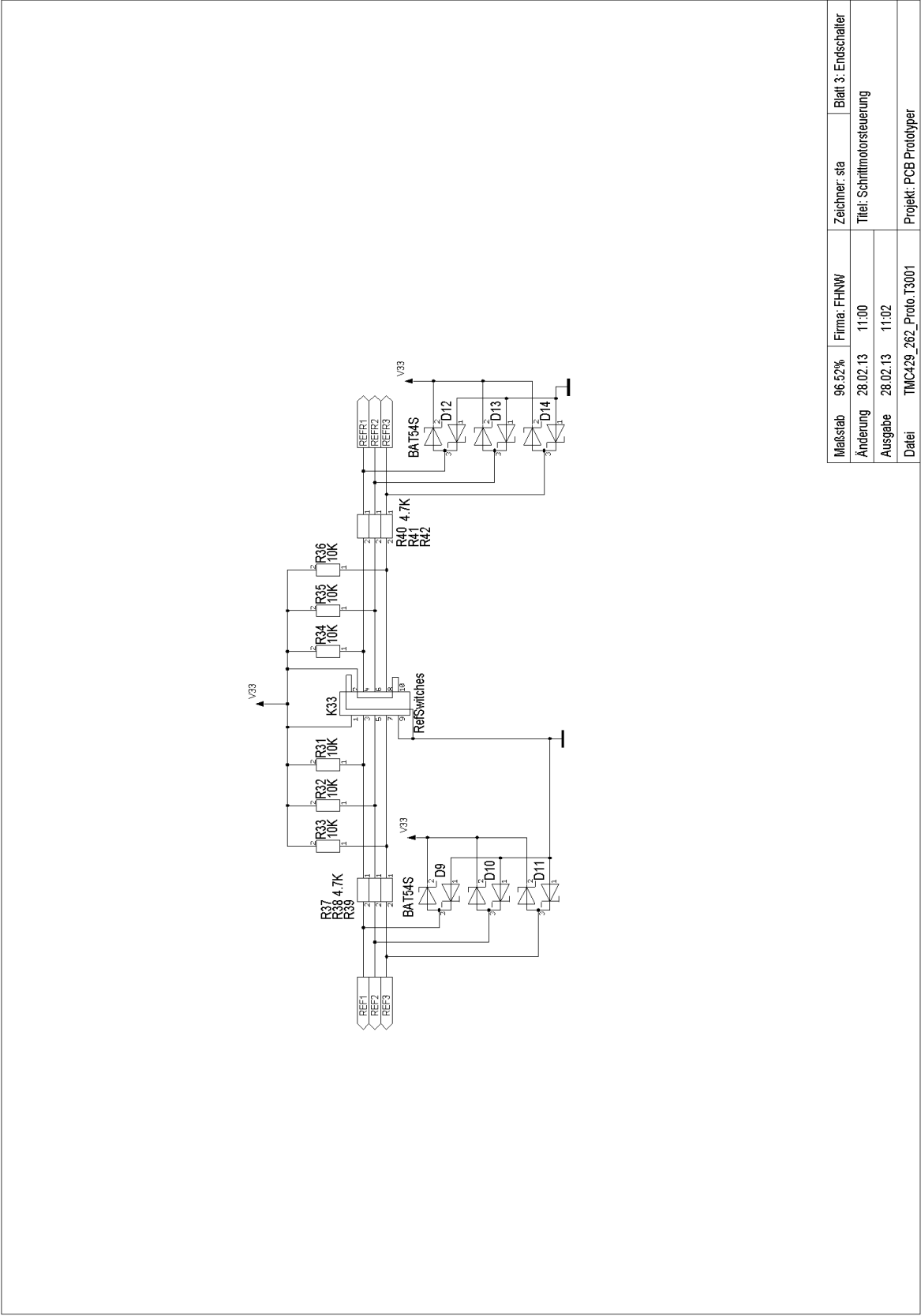


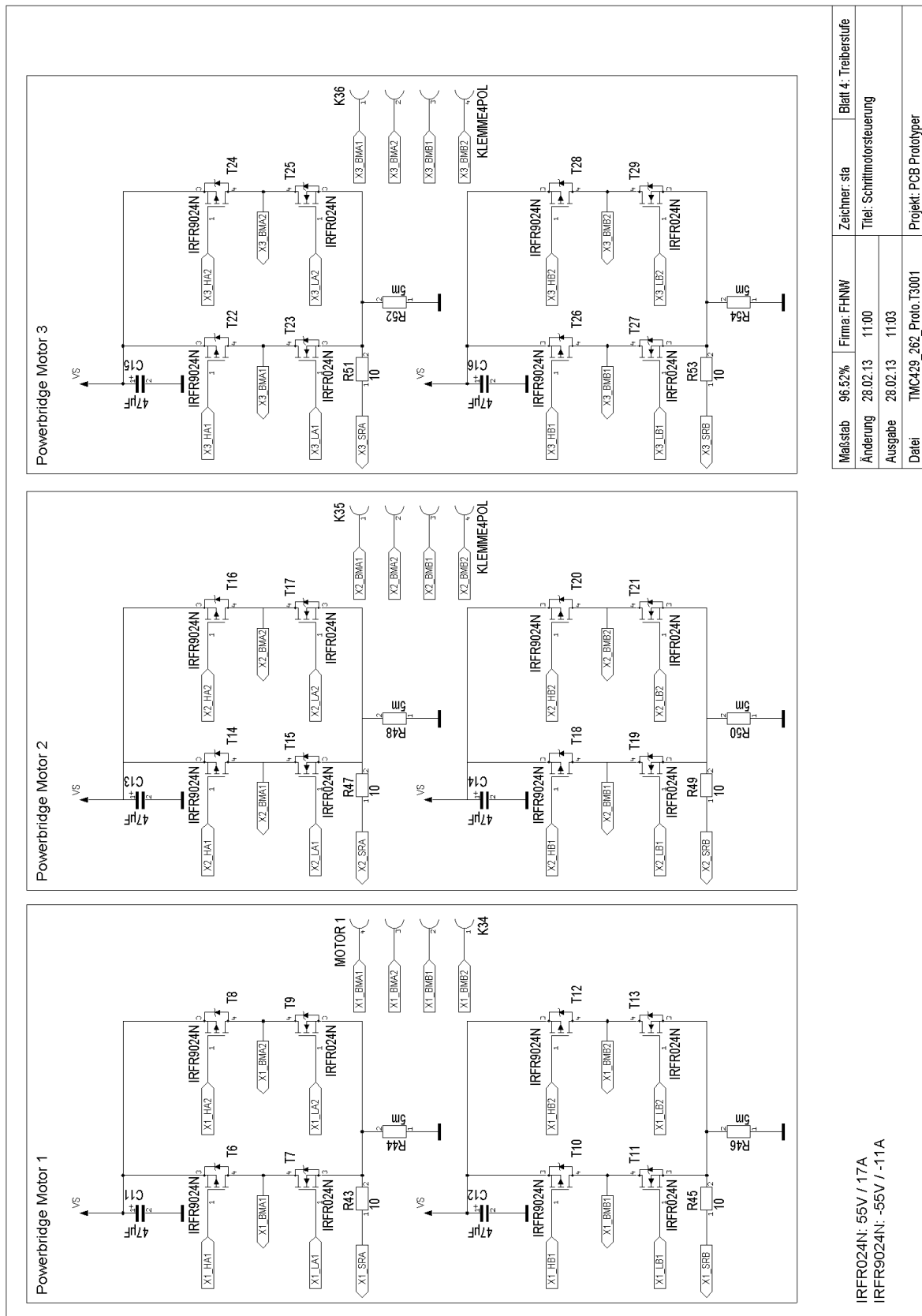
Abbildung 29: Ein- und Ausgabesignale

Maßstab	96.52%	Firma: FHNW	Zeichner: sia	Blatt 2: I/O Signals
Änderung	28.02.13	11:00	Titel: Schrittmotorsteuerung	
Ausgabe	28.02.13	11:01		
Datei	TMC429_262_Proto.T3001			
Projekt: PCB Prototypen				



Maßstab	96.52%	Firma: FHNW	Zeichner: sta	Blatt 3: Endschalter
Änderung	28.02.13	11:00	Titel: Schrittmotorsteuerung	
Ausgabe	28.02.13	11:02		
Datei	TMC429_262_Proto.T3001 Projekt: PCB Prototypen			

Abbildung 30: Endschalter



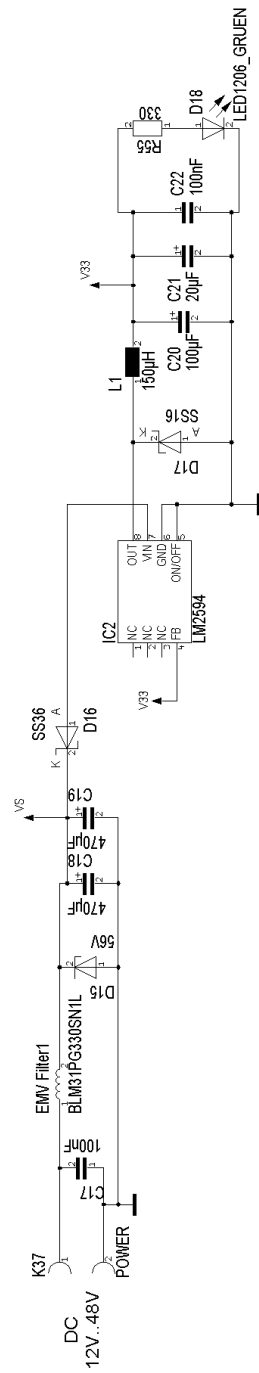


Abbildung 32: Speisung

Maßstab:	96.52%	Firma: FHNW	Zeichner: sta	Blatt 5: Power
Änderung	28.02.13	11:00	Titel: Schrittmotorsteuerung	
Ausgabe	28.02.13	11:04		
Datel	TMC429_362_Proto_T3001			Projekt: PCB Prototyp

4.1.2 RS232 - ARDUINO SCHNITTSTELLE

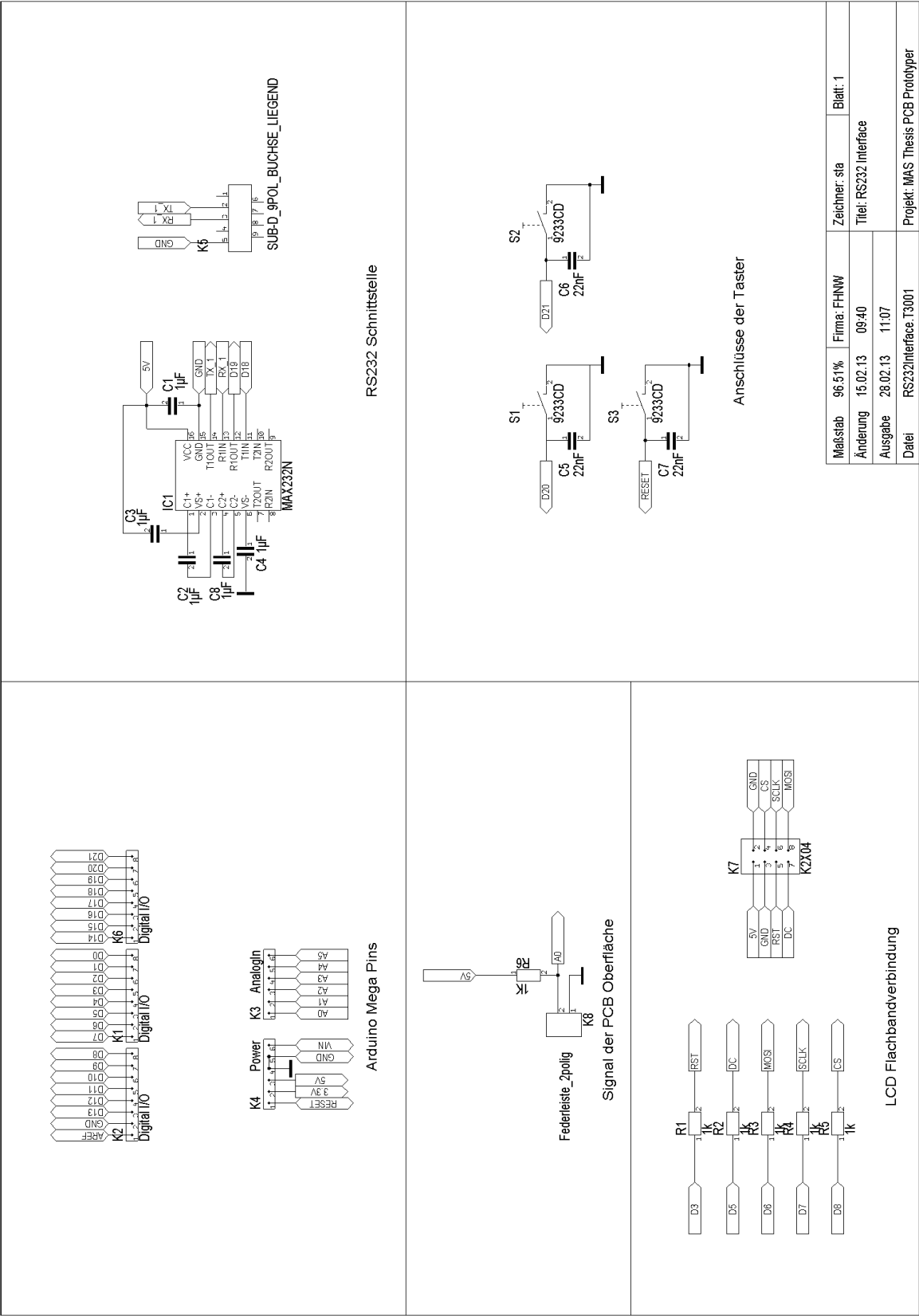


Abbildung 33: CNC Erweiterungsplatine für Arduino

4.2 SOFTWARE

4.2.1 BIBLIOTHEKEN

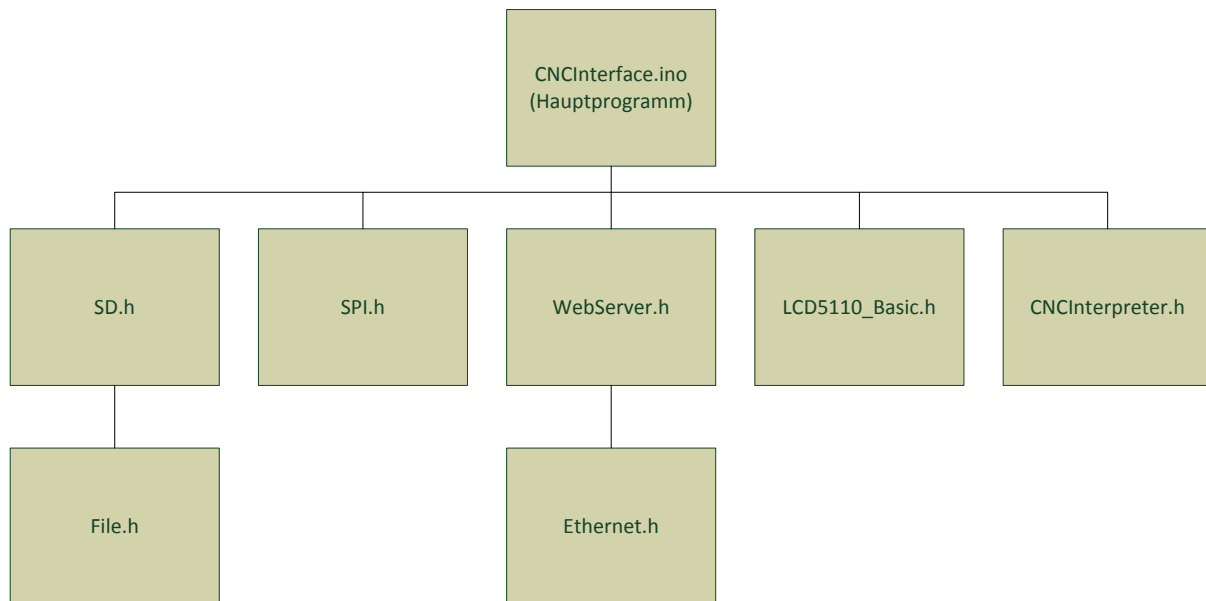


Abbildung 34: Verwendete Bibliotheken

SD.h
- card : Sd2Card - volume : SdVolume - root : SdFile - fileOpenMode : int
+ begin(uint8_t csPin) : boolean + exists(char *filepath) : boolean + mkdir(char *filepath) : boolean + remove(char *filepath) : boolean + rmdir(char *filepath) : boolean

File.h
- _name : char - _file : SdFile*
+ File(SdFile f, const char *name) + File() + write(uint8_t) : virtual size_t + write(const uint8_t *buf, size_t size) : virtual size_t + read() : virtual int + peek() : virtual int

Anhang

```
+ available() : virtual int
+ flush() : virtual void
+ read(void *buf, uint16_t nbyte) : int
+ seek(uint32_t pos) : boolean
+ position() : uint32_t
+ size() : uint32_t
+ close()
+ bool() : operator
+ name() : char *
+ isDirectory() : boolean
+ openNextFile(uint8_t mode) : File
+ rewindDirectory()
```

SPI.h

```
+ transfer(byte _data) : inline static byte
+ attachInterrupt() : inline static void
+ detachInterrupt() : inline static void
+ begin() : static void
+ end() : static void
+ setBitOrder(uint8_t) : static void
+ setDataMode(uint8_t) : static void
+ setClockDivider(uint8_t) : static void
```

WebServer.h

```
- m_server : EthernetServer
- m_client : EthernetClient
- m_urlPrefix : const char *
- m_pushback[32] : unsigned char
- m_pushbackDepth : char
- m_contentLength : int
- m_authCredentials[51] : char
- m_readingContent : bool
- m_failureCmd : Command *
- m_defaultCmd : Command *
- m_cmdCount : char
- m_urlPathCmd : UrlPathCommand *
- m_commands[8] : struct CommandMap {verb : const char *, cmd : Command *}
+ ConnectionType : enum

+ Command(WebServer &server, ConnectionType type, char *url_tail, bool tail_complete) : typedef
+ UrlPathCommand(WebServer &server, ConnectionType type, char **url_path, char *url_tail, bool
tail_complete) : typedef
+ WebServer(const char *urlPrefix, int port)
+ begin()
+ processConnection()
```


Anhang

```
+ processConnection(char *buff, int *bufflen)
+ setDefaultCommand(Command *cmd)
+ setFailureCommand(Command *cmd)
+ addCommand(const char *verb, Command *cmd)
+ setUrlPathCommand(UrlPathCommand *cmd)
+ printCRLF()
+ printP(const prog_uchar *str)
+ writeP(const prog_uchar *data, size_t length)
+ radioButton(const char *name, const char *val, const char *label, bool selected)
+ checkBox(const char *name, const char *val, const char *label, bool selected)
+ read() : int
+ push(int ch)
+ expect(const char *expectedStr) : bool
+ readInt(int &number) : bool
+ readHeader(char *value, int valueLen)
+ readPOSTparam(char *name, int nameLen, char *value, int valueLen) : bool
+ nextURLparam(char **tail, char *name, int nameLen, char *value, int valueLen) :
URLPARAM_RESULT
+ checkCredentials(const char authCredentials[45]) : bool
+ httpFail()
+ httpUnauthorized()
+ httpServerError()
+ httpNoContent()
+ httpSuccess(const char *contentType, const char *extraHeaders)
+ httpSeeOther(const char *otherURL)
+ available() : uint8_t
- reset()
- getRequest(WebServer::ConnectionType &type, char *request, int *length)
- dispatchCommand(ConnectionType requestType, char *verb, bool tail_complete) : bool
- processHeaders()
- outputCheckboxOrRadio(const char *element, const char *name, const char *val, const char *label,
bool selected)
- defaultFailCmd(WebServer &server, ConnectionType type, char *url_tail, bool tail_complete) :
static void
- noRobots(ConnectionType type)
- favicon(ConnectionType type)
```

Ethernet.h

```
- IPAddress _dnsServerAddress
- DhcpClass *_dhcp
+ _state[] : static uint8_t
+ _server_port[] : static uint16_t

+ begin(uint8_t *mac_address) : int
+ begin(uint8_t *mac_address, IPAddress local_ip)
+ begin(uint8_t *mac_address, IPAddress local_ip, IPAddress dns_server)
+ begin(uint8_t *mac_address, IPAddress local_ip, IPAddress dns_server, IPAddress gateway)
+ begin(uint8_t *mac_address, IPAddress local_ip, IPAddress dns_server, IPAddress gateway,
```

Anhang

```
IPAddress subnet)
+ maintain() : int
+ localIP() : IPAddress
+ subnetMask() : IPAddress
+ gatewayIP() : IPAddress
+ dnsServerIP() : IPAddress
```

LCD5110_Basic.h

```
- struct _current_font {font: uint8_t *, x_size: uint8_t, y_size: uint8_t, offset:
uint8_t, numchars: uint8_t, inverted: uint8_t}
# P_SCK, P_MOSI, P_DC, P_RST, P_CS: volatile uint8_t *
# B_SCK, B_MOSI, B_DC, B_RST, B_CS: volatile uint8_t
# cfont _current_font
```

```
# _LCD_Write(data: unsigned char, mode: unsigned char)
# _print_char(c: unsigned char, x: int, row: int)
+ LCD5110(SCK: int, MOSI: int, DC: int, RST: int, CS: int)
+ InitLCD()
+ clrScr()
+ clrRow(row: int, start_x: int, end_x: int)
+ invert(bool mode)
+ invertText(bool mode)
+ print(st: char *, x: int, y: int)
+ printNumI(num: long, x: int, y: int)
+ printNumF(num: double, dec: byte, x: int, y: int)
+ setFont(font: uint8_t *)
+ drawBitmap(x: int, y: int, bitmap: uint8_t *, sx: int, sy: int, flash: bool)
```

CNCInterpreter.h

```
- orderNr : int
- paramNr : int
- X : signed long
- Y : signed long
- Z : signed long
+ data_valid : bool
+ order : struct CNC_Order {index : int, orders[4] : char, param[4] :
signed long, translation[50] : char}
```

```
- analyzeString(const char *input_string)
- toSteps(char value)
+ CNCInterpreter()
+ translate(const char *input_string) : char *
```

4.2.2 ZUSTANDSDIAGRAMME

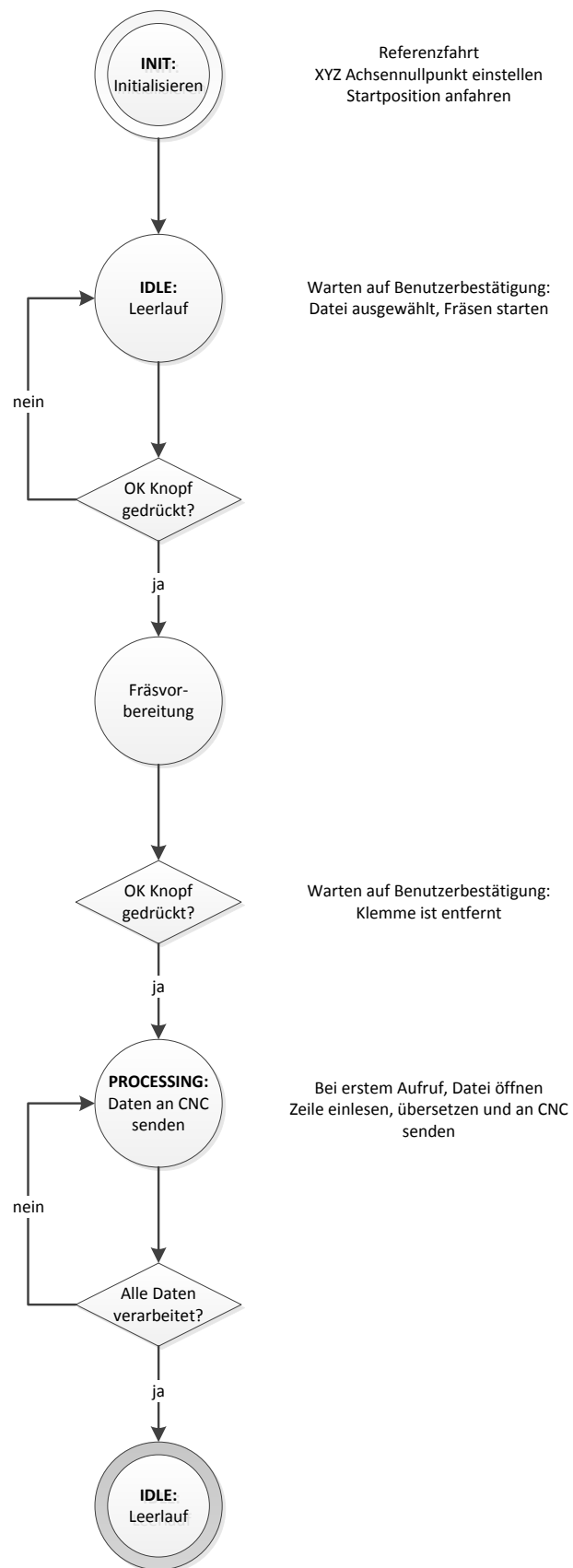


Abbildung 35: Übersicht Zustandsdiagramm Fräsvorgang

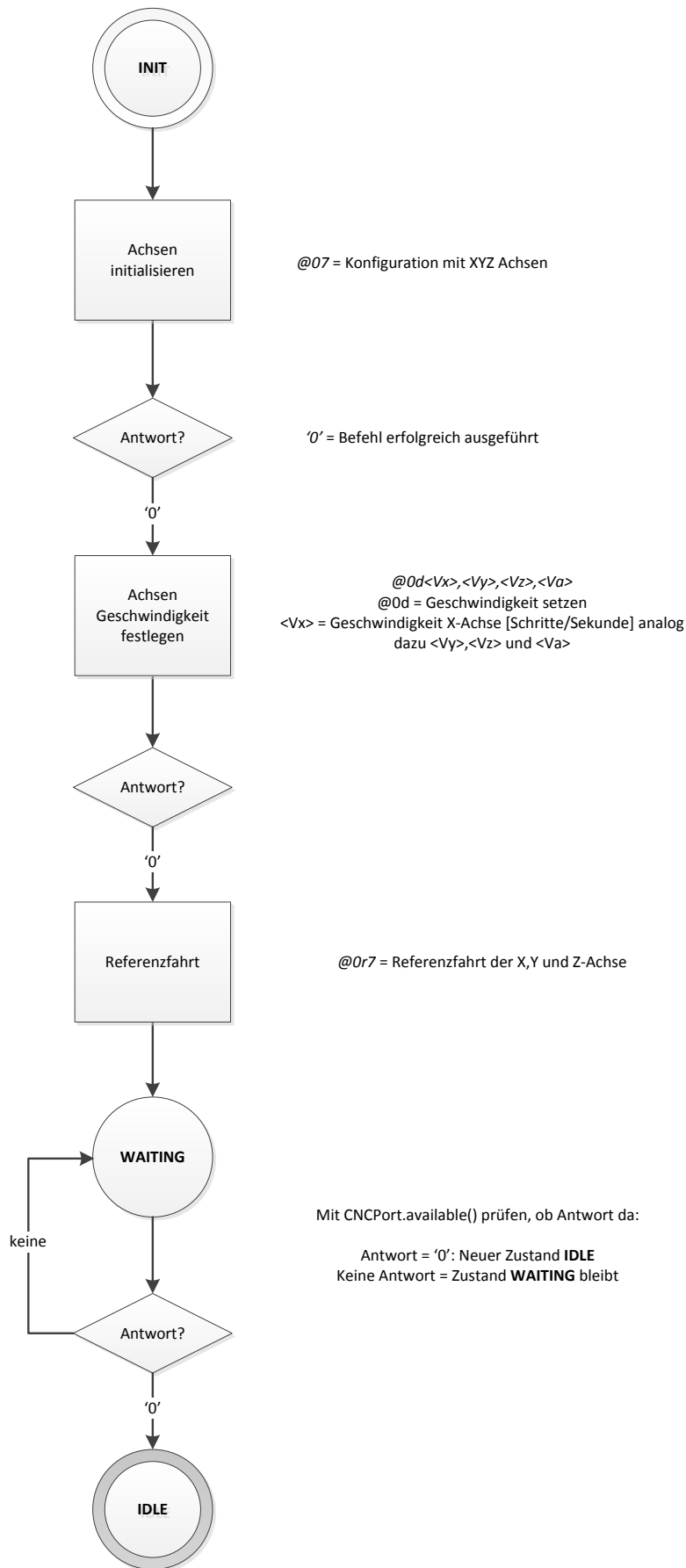


Abbildung 36: Zustandsdiagramm Initialisierung

Anhang

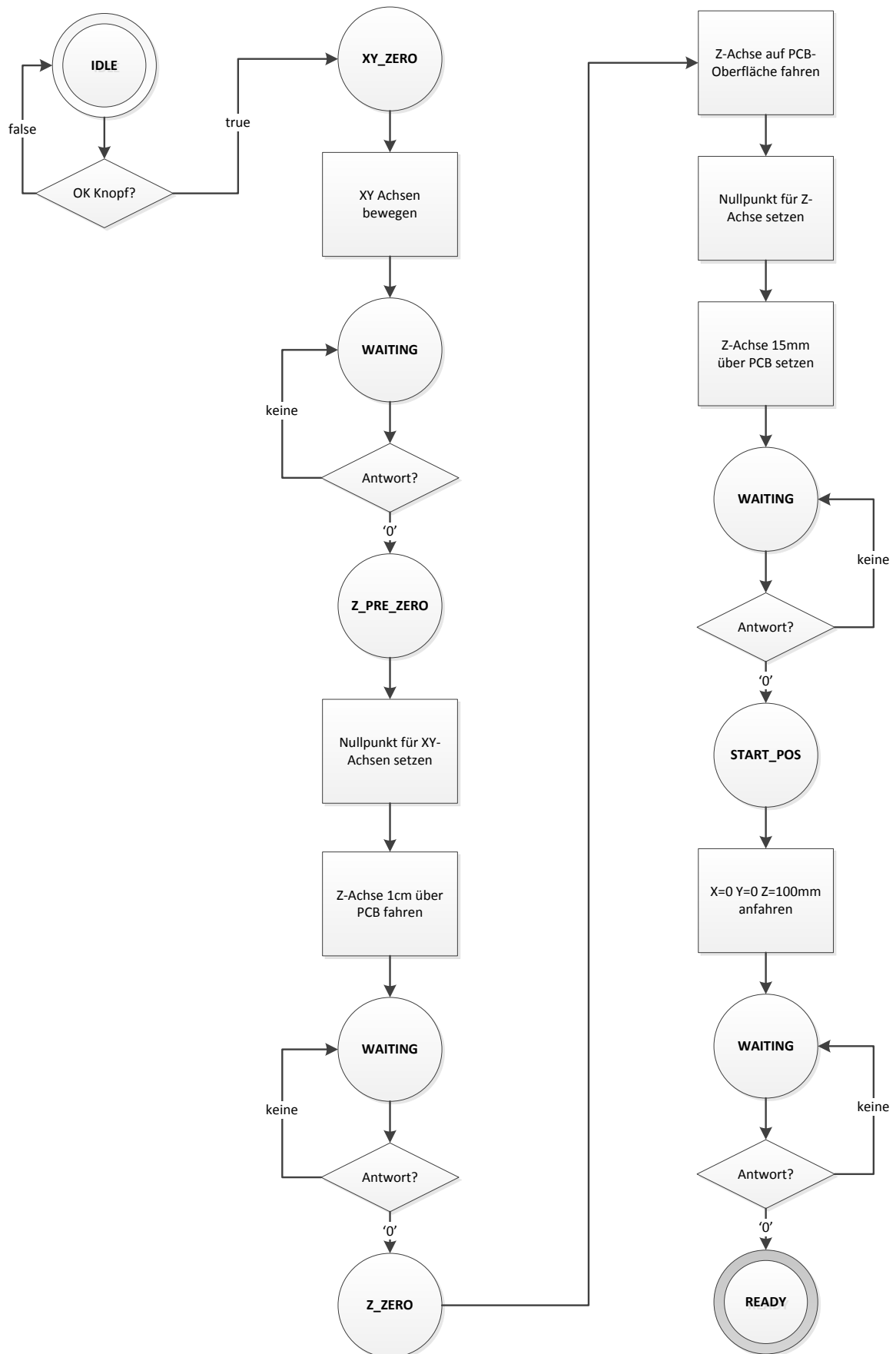


Abbildung 37: Zustandsdiagramm Fräsvorbereitung

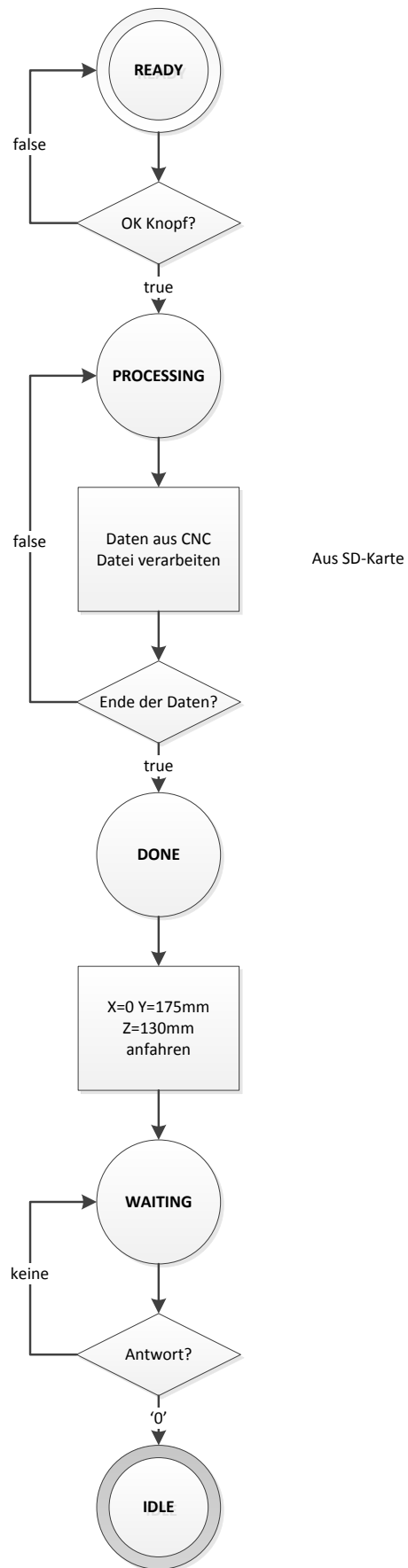


Abbildung 38: Zustandsdiagramm Fräsen

4.2.3 G-CODE REFERENZ

Tabelle 9: Übersicht Allgemeine G-Code Befehle

Zeichen	Beschreibung	Beispiel
N*	Satznummer beginnt mit 1	N10
G*	Vorbereitende Funktion (Interpolation, Ebene, Nullpunkte)	G17
X*	Neue X-Koordinate (Achse auf Ebene orthogonal zu Portal)	X-20
Y*	Neue Y-Koordinate (Portalachse)	Y45
Z*	Neue Z-Koordinate (Parallel zu Frässpindel)	Z2
A*, B*, C*	Drehachsen von X, Y und Z	A5
M*	Maschinenfunktion (z.B. Kühlmittel ein)	M0
T*	Werkzeugnummer	T1
F*	Vorschub in mm/min	F10

Beispiel für einen CNC-Satz:

N10 G90 X100 Y50 Z-0,2 M3

Bedeutung:

(N1) Satz Nr.1

(M3) Spindel im Uhrzeigersinn starten

(G90) die Koordinaten sind absolut

(X100) X-Koordinate von 100mm anfahren, (Y50) Y-Koordinate von 50mm anfahren, (Z-2) Z-Koordinate von -0,2mm (also 0.2mm ins Werkstück) anfahren

Tabelle 10: Beschreibung einiger ausgewählter G-Code Befehle

G-Befehle		M-Befehle	
G00	Eilgang (höchste Geschwindigkeit)	M00	Programm halt
G01	Vorschub geradlinig	M01	Programmhalt nach externer Schalterstellung
G02	Vorschub Kreisbogen UZS	M30	Programm Ende
G03	Vorschub Kreisbogen GUZS	M03	Spindel ein im UZS
G09	Vorschub anhalten vor neuer Bewegung	M04	Spindel ein im GUZS

Anhang

G17	X/Y-Ebene	M05	Spindel stopp
G20 / G21	Programmierung in inch / mm	M06	Werkzeugwechsel
G40	Werkzeug Bahnkorrektur aus	M07	Kühlung ein (Sprühnebel)
G41	Werkzeug Bahnkorrektur links von Kontur	M08	Kühlung / Schmierung ein (flüssig)
G71	Roughing (Z-axis emphasis) Vorschleifen	M09	Kühlung / Schmierung aus
G90	Absolutbemassung		
G91	Relativbemassung		
G94	Vorschub in mm/min		
G95	Vorschub in mm/Umdrehung		
F	Vorschubrate		

Anhang

4.2.4 SOURCECODE HAUPTPROGRAMM

4.2.4.1 CNCINTERFACE.INO

```
/**
  @file CNCInterface.ino

  @brief This is an ethernet interface (Webserver) to control an ISEL CNC
  mill.
  Users can upload their G-Code files and select which file to
  process to create PCB Prototypes.

  @author S. Stapfer

  @version 0.9
*/

#define WEBDUINO_AUTH_REALM "CNC Interface"
#define WEBDUINO_READ_TIMEOUT_IN_MS 10000
#define WEBDUINO_SERIAL_DEBUGGING 0
#define PREFIX ""
#define p_bufferLEN 80
#define PMEM(str) (strcpy_P(p_buffer, PSTR(str)), p_buffer) // used to save
RAM. Strings are stored in Flash instead.

#define LOCAL_DEBUG 0

/* ----- Debug redirection ----- */
#if LOCAL_DEBUG == 1
  #define CNCPort Serial
#else
  #define CNCPort Serial1
#endif
/* ----- */

#include <SD.h>
#include <SPI.h>
#include <Ethernet.h>
#include <WebServer.h>
#include <CNCInterpreter.h>
#include <LCD5110_Basic.h>

/* ----- Server config ----- */
byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0x9A, 0xEB};

#if LOCAL_DEBUG == 1
  byte ip[] = {192, 168, 0, 1};
#else
  byte ip[] = {10, 202, 2, 24}; // IP Labor
#endif

/** @brief This object is the webserver. Function calls can be made by
"server.function()"
  @param PREFIX
  @param Port Port of the webserver. Usually 80*/
WebServer server(PREFIX, 80);
/* ----- End of Server config ----- */

// global variables
```

Anhang

```
bool serverBusy = false;    // if CNC milling is in progress, file upload
                             is not permitted -> serverBusy
bool orderComplete = false; // used to determine if order is complete
volatile bool buttonOK=false;
volatile bool buttonDOWN=false;
int actualFile = 0;         // index number of actual file (from 0 to
                             numFiles)
int numFiles = 0;          // number of datafiles (G-Code) on SD-Card
char p_buffer[p_bufferLEN];
extern uint8_t SmallFont[]; // font definition for LCD

// Definition of the available states
enum state_t
{INIT, WAITING, IDLE, XY_ZERO, Z_PRE_ZERO, Z_ZERO, START_POS, READY, PROCESSING, DONE, CONFIG};
int CNCprevState;
int CNCcurrentState;
int CNCnextState;

/** @brief This object is used to translate G-Code to ISEL specific "@...."
syntax using the CNCInterpreter class*/
CNCInterpreter cnc;

/** @brief This object is used to display strings on an attached LC Display
(Nokia 5110). Communication is made
over the SPI bus.
@param SCK Clock pin
@param MOSI Master out Slave in pin
@param DC Data or Command mode
@param RST Reset pin
@param CS Client select pin*/
LCD5110 lcd(7,6,5,3,8);

/** @brief This object is the file object on the SD-Card.*/
File datafile;

/** @brief This string object is the list of files on the SD-Card.
@param Arraywidth defines the number of files displayed.*/
String fileList[10];

/** @brief This displayIndexHTML function is called if a user request the
/index.htm file. It reads the
content of a index.htm file on the SD-Card and send it to the client.
@param server Pointer to the server object
@param type Type of connection (INVALID, GET, HEAD, POST, PUT, DELETE
or PATCH)
@param url_tail Rest of the URL, that doesn't fit in the given pattern
@param tail_complete True, if the complete URL fit in url_tail buffer,
false if not */
void displayIndexHTML(WebServer &server, WebServer::ConnectionType type,
char *url_tail, bool tail_complete)
{
    server.httpSuccess(); /* send header 400 to client */
    if (type != WebServer::HEAD)
    {
        if (!SD.exists("html/index.htm")) {
            P(helloMsg) = "index.htm not found. Insert SD card.";
            server.printP(helloMsg);
        }
        else {
```

Anhang

```
        /* Load index.htm file from SD-Card and send it to the client. */
        streamFileToServer("html/index.htm");
    }
    Serial.println(PMEM("index...done.));
}
}

/** @brief This displayUploadForm function is called from the index.htm
site or if a user requests the
/upload.htm file. It generates an upload dialog for storing files on
the embedded SD-Card.
@param server Pointer to the server object
@param type Type of connection (INVALID, GET, HEAD, POST, PUT, DELETE
or PATCH)
@param url_tail Rest of the URL, that doesn't fit in the given pattern
@param tail_complete True, if the complete URL fit in url_tail buffer,
false if not */
void displayUploadForm(WebServer &server, WebServer::ConnectionType type,
char *url_tail, bool tail_complete)
{
    bool userValid = false;

    if(!serverBusy)
    {
        /* if the user has requested this page using the following credentials
        * username = user
        * password = user
        * display a page saying "Hello User"
        *
        * the credentials have to be concatenated with a colon like
        * username:password
        * and encoded using Base64 - this should be done outside of your Arduino
        * to be easy on your resources
        *
        * in other words: "dXNlcjplc2Vy" is the Base64 representation of
"user:user"
        *
        * if you need to change the username/password dynamically please search
        * the web for a Base64 library */

        if (server.checkCredentials("dXNlcjplc2Vy"))
        {
            serverBusy = true;
            server.httpSuccess();
            if (type != WebServer::HEAD)
            {
                if (!SD.exists("html/upload.htm")) {
                    #if WEBDUINO_SERIAL_DEBUGGING
                    Serial.println("File not found!");
                    #endif
                    P(helloMsg) = "upload.htm not found. Insert SD card.";
                    server.printP(helloMsg);
                }
                else {
                    streamFileToServer("html/upload.htm");
                }
            }
            Serial.println(PMEM("upload...done.));
        }
        /* if the user has requested this page using the following credentials
```

Anhang

```
* username = admin
* password = admin
* display a page saying "Hello Admin"
*
* in other words: "YWRtaW46YWRtaW4=" is the Base64 representation of
"admin:admin" */
else
{
    /* send a 401 error back causing the web browser to prompt the user for
credentials */
    server.httpUnauthorized();
}
}
else
{
    server.httpSuccess();
    if (!SD.exists("html/locked.htm")) {
        #if WEBDUINO_SERIAL_DEBUGGING
        Serial.println("File not found!");
        #endif
        P(helloMsg) = "locked.htm not found. Insert SD card.";
        server.printP(helloMsg);
    }
    else {
        streamFileToServer("html/locked.htm");
    }
}
}

/** @brief This processDataUpload function is called from the upload.htm
form. It starts the upload of data to
the SD-Card.
@param server Pointer to the server object
@param type Type of connection (INVALID, GET, HEAD, POST, PUT, DELETE
or PATCH)
@param url_tail Rest of the URL, that doesn't fit in the given pattern
@param tail_complete True, if the complete URL fit in url_tail buffer,
false if not */
void processDataUpload(WebServer &server, WebServer::ConnectionType type,
char *url_tail, bool tail_complete)
{
    URLPARAM_RESULT rc;

    /* if we're handling a GET or POST, we can output our data here.
For a HEAD request, we just stop after outputting headers. */
    if (type == WebServer::HEAD)
        return;

    switch (type)
    {
        case WebServer::GET:
            Serial.println(PMEM("GET Request"));
            server.httpSeeOther("index.htm");
            break;
        case WebServer::POST:
            /* this line sends the standard "we're all OK" headers back to the
browser */
            server.httpSuccess();

            Serial.println(PMEM("POST Request"));
```

Anhang

```
    if (!SD.exists("html/done.htm")) {
        #if WEBDUINO_SERIAL_DEBUGGING
        Serial.println("File not found!");
        #endif
        P(helloMsg) = "done.htm not found. Insert SD card.";
        server.printP(helloMsg);
    }
    else {
        streamFileToServer("html/done.htm");
    }
    delay(10);
    streamDataToFile();
    server.println(PMEM("</div></div></body></html>"));
    break;
default:
    server.httpSuccess();
    server.print("Unknown request\n");
    return;
}
Serial.println(PMEM("file parsing...done.));
serverBusy = false;
}

/** @brief This displayLogoutHTML function is called if the users clicks on
the logout button after an upload.
@param server Pointer to the server object
@param type Type of connection (INVALID, GET, HEAD, POST, PUT, DELETE
or PATCH)
@param url_tail Rest of the URL, that doesn't fit in the given pattern
@param tail_complete True, if the complete URL fit in url_tail buffer,
false if not */
void displayLogoutHTML(WebServer &server, WebServer::ConnectionType type,
char *url_tail, bool tail_complete)
{
    server.httpSuccess();
    if (!SD.exists("html/logout.htm")) {
        P(helloMsg) = "logout.htm not found. Insert SD card.";
        server.printP(helloMsg);
    }
    else {
        streamFileToServer("html/logout.htm");
    }
}

/** @brief This function is waiting for the acknowledge byte '0' from the
CNC machine */
void waitForCNC()
{
    while(CNCPort.read() != '0');
}

/** @brief Setup function is called once at the beginning. It is used to
initialize all components such as Serial
communication, LCD, SD-Card, Webserver and so on. */
void setup()
{
    Serial.begin(115200);    // initialize serial console

    char myIP[16] = ""; //maximale Größe Strings sind 15 Zeichen plus
Terminierung
```

Anhang

```
int len = 0;
len = sprintf(myIP, "%d.%d.%d.%d",ip[0],ip[1],ip[2],ip[3]);

lcd.InitLCD();
lcd.setFont(SmallFont);
lcd.clrScr();

// attach 2 interrupts for the buttons (OK and Down)
attachInterrupt(3,okPressed,FALLING);
attachInterrupt(2,downPressed,FALLING);

/* Begin SD Card initialization */
if (!SD.begin(4)) {
    Serial.println(PMEM("initialization failed!"));
    return;
}
lcd.print("SD Card found.",LEFT,0);
Serial.println(PMEM("SD Card found."));
/* End of SD Card initialization */

/* Begin Webserver initialization */
Ethernet.begin(mac,ip);
server.setDefaultCommand(&displayIndexHTML);
// register functions (see above) for specific pages i.e. "index.htm"
server.addCommand("index.htm", &displayIndexHTML);
server.addCommand("upload.htm",&displayUploadForm);
server.addCommand("done.htm",&processDataUpload);
server.addCommand("logout.htm",&displayLogoutHTML);
server.begin();
lcd.print("Serveraddress:",LEFT,8);
lcd.print(myIP,LEFT,16);
Serial.print(PMEM("Serveraddress: "));
Serial.println(myIP);
/* End of Webserver initialization */

/* Begin CNC (Serial) initialization */
#if LOCAL_DEBUG == 0
    CNCPort.begin(19200);
#endif
CNCcurrentState = INIT;
/* End CNC (Serial) initialization */

}

/** @brief loop function is called multiple times.

Basically it does 3 things:
1. Processing of the webserver
2. Processing of the current state
3. Switching to the new state*/
void loop()
{
    char buff[512]; //64 / 128
    int len = 512;

    /* process incoming connections one at a time forever */
    server.processConnection(buff, &len);

    /* main state machine */
    switch(CNCcurrentState)
    {
```

Anhang

```
/** States:
- Init: Configuration of CNC\n
- Waiting: Wait for CNC to complete the last order. Depending on
previous state there are different displays on
the LCD\n
- Idle: Processing of the user interface\n
- XY_Zero: Sets the X and Y origin 10 mm on the inner side of the
PCB\n
- Z_Pre_Zero: Approach the PCB as preparation for the next step\n
- Z_Zero: Move drill down on the surface of the PCB\n
- Startpos: Move drill to a start position 100mm above the PCB\n
- Ready: Wait for user to switch on the drilling motor\n
- Processing: Send the next G-Code order to the CNC machine\n
- Done: Move the milling tool away and display done to the user\n*/
case INIT:
    initCNC();
    CNCprevState = CNCcurrentState;    // save previous state
    CNCnextState = WAITING;
    break;
case WAITING:
    if(CNCPort.read()=='0')
    {
        switch(CNCprevState)
        {
            case INIT:
                Serial.println("Init done.");
                lcd.clrScr();
                CNCnextState = IDLE;
                break;
            case XY_ZERO:
                CNCnextState = Z_PRE_ZERO;
                break;
            case Z_PRE_ZERO:
                CNCnextState = Z_ZERO;
                break;
            case Z_ZERO:
                CNCnextState = START_POS;
                break;
            case START_POS:
                CNCnextState = READY;
                break;
            case DONE:
                Serial.println(PMEM("Bitte fertige Platine entfernen."));
                lcd.clrScr();
                lcd.print("Bitte fertige",CENTER,8);
                lcd.print("Platine",CENTER,16);
                lcd.print("entfernen",CENTER,24);
                while(!buttonOK);
                buttonOK=false;
                lcd.clrScr();
                CNCnextState = INIT;
                break;
            default:
                Serial.print("Unknown state: ");
                Serial.println(CNCprevState);
                CNCnextState = IDLE;
        }
    }
    break;
case IDLE:
    #if LOCAL_DEBUG == 1
```

Anhang

```
//      --- OK button simulation ---
char btnRead;
btnRead = CNCPort.read();
if(btnRead=='o')
{
    buttonOK=true;
}
if(btnRead=='l')
{
    buttonDOWN=true;
}
#endif

processUserInterface();
break;
case XY_ZERO:
    lcd.clrScr();
    lcd.print("In Bearbeitung",CENTER,16);
    serverBusy = true;
    setXYAxesZero();
    CNCprevState = CNCcurrentState;    // save previous state
    CNCnextState = WAITING;
    break;
case Z_PRE_ZERO:
    moveZAxis();
    CNCprevState = CNCcurrentState;    // save previous state
    CNCnextState = WAITING;
    break;
case Z_ZERO:
    setZAxisZero();
    CNCprevState = CNCcurrentState;    // save previous state
    CNCnextState = WAITING;
    break;
case START_POS:
    moveToStartPos();
    CNCprevState = CNCcurrentState;    // save previous state
    CNCnextState = WAITING;
    break;
case READY:
    Serial.println(PMEM("Bitte Fraesmotor starten.));
    lcd.clrScr();
    lcd.print("Bitte Fraes-",CENTER,8);
    lcd.print("motor starten",CENTER,16);
    lcd.print("OK = Weiter",CENTER,24);
    while(!buttonOK)
    {
#if LOCAL_DEBUG == 1
        // simulation of OK Button
        if(CNCPort.read()=='o')
        {
            buttonOK=true;
        }
#endif
    }

    detachInterrupt(3);    // Interrupt des Ok-Knopfes deaktivieren
    buttonOK=false;
    lcd.clrScr();
    sendCNC();
    CNCprevState = CNCcurrentState;    // save previous state
    break;
case PROCESSING:
```


Anhang

```
        lcd.clrScr();
        sendCNC();
        CNCprevState = CNCcurrentState;    // save previous state
        break;
    case DONE:
        moveToolAway();
        attachInterrupt(3,okPressed,FALLING); // Interrupt des Ok-Knopfes
wieder aktivieren
        lcd.clrScr();
        CNCprevState = CNCcurrentState;    // save previous state
        CNCnextState = WAITING;
        break;
    default:
        ;
}

CNCcurrentState = CNCnextState;    // set new state
}
```

Anhang

4.2.4.2 CNCHANDLER.INO

```
/** @file CNCHandler.ino
    @brief These functions are used in the milling process to move the CNC
    and send the new orders.*/

/** this function initializes the CNC machine by\n
    - defining the axes XYZ\n
    - the speed of the different axes\n
    - doing a reference movement*/
void initCNC()
{
    Serial.println(PMEM("--- init ---"));
    CNCPort.println("@07");// definition of axes to use: X,Y and Z
    waitForCNC();
    CNCPort.println("@0d2000,2000,2000,0");// config of axis speeds
    waitForCNC();
    CNCPort.println("@0R7");// reference movement
}

/** this function changes the X and Y origin of the coordinate system to
the corner in front on the left side*/
void setXYAxesZero()
{
    Serial.println(PMEM("--- XY Zero ---"));
    CNCPort.println("@0M4500,5000,-37600,5000,0,1000,0,1000");// <---
change X and Y here to set coordinate origin of PCB
}

/** this function moves the drilling head 10 mm above the PCB surface and
25mm in X and Y direction*/
void moveZAxis()
{
    CNCPort.println("@0n3");// set x- and y-Axis 0 on actual position
    waitForCNC();
    CNCPort.println("@0M-1000,5000,5000,5000,-26000,5000,0,1000");//
prepare for PCB touch
}

/** this function moves the drilling head down, until it touches the PCB.
The new position is saved as Z = 0.\n
After resetting the Z position the head is moved away from the PCB.*/
void setZAxisZero()
{
    int ZValue=0;

    Serial.println(PMEM("--- set Z zero ---"));
    Serial.println(analogRead(0));

    // Move down until tool touches PCB surface
    while((ZValue=analogRead(0))>100)
    {
        CNCPort.println("@0A0,1000,0,1000,-10,100,0,1000");// move z-Axis
10 steps down
        waitForCNC();
        Serial.print(PMEM("Z-Wert: "));
        Serial.println(ZValue);
    }
    delay(1000);
    CNCPort.println("@0n4");// set z-Axis 0 on actual position
    waitForCNC();
    Serial.println(PMEM("--- move Z 15mm up ---"));
```

Anhang

```
CNCPort.println("@0A0,5000,0,5000,3000,5000,0,1000");// move z-Axis
15mm above PCB
}

/** this function moves away the head at about 100mm above PCB*/
void moveToStartPos()
{
    Serial.println(PMEM("--- move to Startpos ---"));
    CNCPort.println("@0M0,5000,0,5000,20000,5000,0,1000");// move to X,Y=0
    Z 100mm above PCB
}

/** this function is called to send the next order to the CNC machine. If
there is no new line the order is\n
completed and the state changes to Done.*/
void sendCNC()
{
    String tmpFilename;

    tmpFilename = "data/";
    tmpFilename += fileList[actualFile];
    #if WEBDUINO_SERIAL_DEBUGGING > 0
        Serial.println(tmpFilename);
    #endif
    tmpFilename.toCharArray(p_buffer,p_bufferLEN);
    streamDataToCNC(p_buffer);
    if(orderComplete)
    {
        CNCnextState = DONE;
        Serial.println(PMEM("Order complete.));
        serverBusy = false;
    }
    else
    {
        CNCnextState = PROCESSING;
    }
}

/** this function moves the table in front to the user and the drilling
head up.*/
void moveToolAway()
{
    Serial.println(PMEM("--- move Tool away ---"));
    CNCPort.println("@0M0,5000,35000,5000,26000,5000,0,1000");// move to
X=0 Y=175mm Z=130mm above PCB
}
```

Anhang

4.2.4.3 STREAMING.INO

```
/** @file Streaming.ino
    @brief These functions are used to stream data from the SD card to the
    webserver and the CNC and from the webserver
    to the SD Card.*/

/** @brief this function streams the specified file from the SD card to the
    webserver. Used to display HTML pages
    saved on SD card.
    @param filename Name of the file to stream*/
void streamFileToServer(char *filename)
{
    char buf[128];
    int16_t c;
    int16_t index = 0;

    datafile = SD.open(filename, FILE_READ);
    delay(10);
    while((c=datafile.read()) > 0) {    // read file until EOF
        switch(c) {
            case 10:    // line feed
                buf[index] = '\0';
                server.println(buf);
                index = 0;
                break;
            default:
                buf[index++] = (char)c;
        }
        delay(1);
    }
    buf[index] = '\0';
    server.println(buf);
    datafile.close();
    delay(10);
    free(buf);
}

/** @brief this function streams data from a user selected file to the SD
    card using the upload dialog.*/
bool streamDataToFile()
{
    int ch=0, line=0;
    char databuffer[80];
    char *ptr_databuffer = databuffer;
    char filename[24] = "data/";
    char tmpFilename[24];
    char *ptr_filename = tmpFilename;

#ifdef WEBDUINO_SERIAL_DEBUGGING > 0
    Serial.println(PMEM("-----Header-----"));
#endif
    // remove header
    server.readHeader(databuffer,80);
    server.readHeader(databuffer,80);

    /* get Filename */
    ptr_databuffer = strstr(databuffer,"filename=");
    ptr_databuffer += 10;
    while(*ptr_databuffer != '.')
    {
        *ptr_filename++ = *ptr_databuffer++;
    }
}
```

Anhang

```
    }
    *ptr_filename = '\0';
    strcat(filename, tmpFilename);
    free(tmpFilename);
    strcat(filename, ".cnc");
#ifdef WEBDUINO_SERIAL_DEBUGGING > 0
    Serial.println(filename);
#endif
    ptr_databuffer = databuffer;          // reset index of ptr_databuffer to
databuffer[0]
    /* End of get Filename */

    server.readHeader(databuffer,80);

    for(int index=0;index<4;index++)      // remove CR's and LF at beginning
    {
        server.read();
    }
#ifdef WEBDUINO_SERIAL_DEBUGGING > 0
    Serial.println(PMEM("-----Params-----"));
#endif

    while((ch = server.read()) != -1)
    {
        while(ch!=10) {
            if( (ch>64 && ch<91) || (ch>96 && ch<123) || (ch>47 && ch<58)
|| (ch==45) || (ch==46) || (ch==32)) {
                *ptr_databuffer++ = (char)ch;
            }
            ch = server.read();
        }
        *ptr_databuffer = '\0';
        ptr_databuffer = databuffer;          // reset index of
ptr_databuffer to databuffer[0]
        if(databuffer[0]=='N') {
            line++;
#ifdef WEBDUINO_SERIAL_DEBUGGING > 0
            Serial.print("Line ");
            Serial.print(line);
            Serial.print(": ");
            Serial.println(ptr_databuffer);
#endif
            if(line==1) {
                if(SD.exists(filename))
                {
                    Serial.println(PMEM("WARNING: file already used!"));
                    Serial.println(PMEM("Remove existing file?"));
                    Serial.println(PMEM("Ok = Remove file, Down = Leave
file"));

                    lcd.clrScr();
                    lcd.print("Datei ersetzen?",CENTER,0);
                    lcd.print("Ok = Ja",CENTER,8);
                    while(!buttonOK)
                    {
#ifdef LOCAL_DEBUG == 1
                        //      --- OK button simulation ---
                        char btnRead;
                        btnRead = CNCPort.read();
                        if(btnRead=='o')
                        {
                            buttonOK=true;

```

Anhang

```
        }
        if(btnRead=='1')
        {
            buttonDOWN=true;
        }
#endif

        if(buttonDOWN)
        {
            buttonDOWN=false;
            Serial.println("File NOT removed.");
            server.println("Datei nicht ersetzt.<br />");
            return false;
        }
        buttonOK=false;
        SD.remove(filename);
        Serial.println("Saving new file...");
        lcd.print("Wird gespeichert...",CENTER,8);
        datafile = SD.open(filename, FILE_WRITE);
        delay(10);
    }
    else
    {
        datafile = SD.open(filename, FILE_WRITE);
        delay(10);
    }
    }
    server.println(ptr_databuffer);
    server.println("<br />");
    while(*ptr_databuffer!='\0') {
#endif WEBDUINO_SERIAL_DEBUGGING > 0
        Serial.print(*ptr_databuffer);
#endif

        datafile.write(*ptr_databuffer++);
    }
    datafile.write('\n');
    ptr_databuffer = databuffer;
}

}
if(line>0) {
#endif WEBDUINO_SERIAL_DEBUGGING > 0
    Serial.println(PMEM("File closed."));
#endif
    datafile.close();
    delay(10);
    Serial.println("written to file.");
    lcd.print("Fertig",CENTER,12);
}
return true;
}

/** @brief this function streams data on the SD card to the USART1 port
    @param filename Name of file to stream from SD card to the USART1
    port*/
void streamDataToCNC(char *filename)
{
    char buf[80];    // buffer of 80 bytes too big for Arduino Uno
    char lcdTmpBuffer[12];
    int16_t c;
    int16_t index = 0;
    long lineNumber = 0;
```

Anhang

```
char firstStreamChar;
char *translatedString = NULL;

if(CNCcurrentState == READY)
{
    datafile = SD.open(filename, FILE_READ);
    delay(10);
    Serial.println(PMEM("File open.));
    orderComplete = false;
    CNCnextState = PROCESSING;
}
else
{
    while(c=datafile.read() > 0)
    {
        buf[index++] = (char)(c+77);    // problem in uploaded file?
        while(((c=datafile.read())!=10) && (c!=13)) {    // read file
until LF or CRLF (Windows)
            buf[index++] = (char)c;
        }
        buf[index] = '\0';
        Serial.println(buf);
        Serial.println(PMEM("-----"));
        translatedString = cnc.translate(buf);
        if(translatedString[0]!='@')
        {
            Serial.println(translatedString);
            CNCPort.println(translatedString);
#if LOCAL_DEBUG == 1
#else
            lcd.print("Line Nr.:",0,CENTER);
            lcd.print(ltoa(lineNumber++,lcdTmpBuffer,10),8,CENTER);
            waitForCNC();
#endif
            CNCnextState = PROCESSING;
        }
        index = 0;
    }
    datafile.close();
    delay(10);
    orderComplete = true;
}
}

/** @brief this function saves the filenames in the data directory in the
fileList an prints the first 6 on the lcd screen.*/
int getFileList()
{
    int numFiles = 0;

    File dir = SD.open("data/");

    while(true) {
        File entry = dir.openNextFile();
        if (! entry) {
//            Serial.println("**nomorefiles**");
            entry.close();
            dir.close();
            return numFiles;
        }
        if (!entry.isDirectory()) {
```

Anhang

```
    if(numFiles < 10)
    {
        Serial.println(entry.name());
        lcd.print(entry.name(),LEFT,numFiles*8);
        fileList[numFiles] = entry.name();
        numFiles++;
    }
    else
    {
        Serial.println(PMEM("***zu viele Dateien**"));
        entry.close();
        dir.close();
        return 9;
    }
}
}
```


Anhang

4.2.4.4 UI.INO

```
/** @file UI.ino
    @brief These functions are used to build a user interface logic and to
    process user input.*/

int menulayer = 0;

enum indexlcdmenu_t {LISTFILES=1,SETUP};
int indexLCDMenu=1;

/** @brief this is the small interrupt routine for the ok button.*/
void okPressed()
{
    buttonOK=true;
}

/** @brief this is the small interrupt routine for the down button.*/
void downPressed()
{
    buttonDOWN=true;
}

/** @brief this function displays the user menu on top layer.
    @param index This is the current cursor position*/
void lcdMenuTop(byte index)
{
    switch (index)
    {
        {
            lcd.clrScr();
            case 1:
                lcd.print("=> CNC Daten",LEFT,0);
                lcd.print("    Setup",LEFT,8);
                break;
            case 2:
                lcd.print("    CNC Daten",LEFT,0);
                lcd.print("=> Setup",LEFT,8);
                break;
            default:
                lcd.print("    CNC Daten",LEFT,0);
                lcd.print("    Setup",LEFT,8);
        }
    }
}

/** @brief this function processes the user input logic*/
void processUserInterface()
{
    char *buf = p_buffer;

    if(menulayer==0)
    {
        lcdMenuTop(indexLCDMenu);

        if(buttonOK)
        {
            lcd.clrScr();
            switch(indexLCDMenu)
            {
                {
                    case LISTFILES:
                        Serial.println(PMEM("Available files:"));
                        numFiles = getFileList();
                        menulayer = 1;
                }
            }
        }
    }
}
```

```

        break;
    case SETUP:
        Serial.println(PMEM("Config mode."));
        lcd.print("Configuration:",LEFT,0);
        break;
    default:
        ;
    }
    buttonOK=false;
    delay(100);
}
if(buttonDOWN)
{
    if((indexLCDMenu<0)|| (indexLCDMenu>1))
    {
        indexLCDMenu=1;
    }
    else
    {
        indexLCDMenu++;
    }
    buttonDOWN=false;
    delay(500);
}
}
if(menulayer == 1)
{
    if(buttonOK)
    {
        Serial.print("> ");
        Serial.println(fileList[actualFile]);
        CNCnextState = XY_ZERO;
        buttonOK=false;
        menulayer=0;
        delay(100);
    }
    if(buttonDOWN)
    {
        if(actualFile>(numFiles-2))
        {
            actualFile = 0;
        }
        else
        {
            actualFile++;
        }
        Serial.println(fileList[actualFile]);
        lcd.clrScr();
        fileList[actualFile].toCharArray(buf,80);
        lcd.print(buf,LEFT,0);
        buttonDOWN=false;
        delay(1000);
    }
}
}

```

Anhang

4.2.5 SOURCECODE CNCINTERPRETER KLASSE

4.2.5.1 CNCINTERPRETER.H

```
#include <string.h>
#include <stdlib.h>
#include <stdint.h>

class CNCInterpreter {
private:
    int orderNr, paramNr;
    char stringValue[14];

public:
    signed long X, Y, Z;
    CNCInterpreter();
    int analyzeString(const char* input_string);
    char* translate(const char* input_string);
    char* longToString(char* buf, long val, unsigned char pad, char
padchar);
    long toSteps(char value);
    bool data_valid;

    struct CNC_Order{
        unsigned long lineNumber;
        char orders[4];
        signed int param[4];
        char translation[128];
    } order;

};
```

Anhang

4.2.5.2 CNCINTERPRETER.CPP

```
/**
 *
 */
#include "CNCInterpreter.h"
#include <ctype.h>

#define stepsPerRot 400      // Schritte pro Umdrehung 400
#define gearFact 2          // Getriebefaktor 2
#define spindlePitch 4      // Spindelsteigung 4

CNCInterpreter::CNCInterpreter()    // Konstruktor
{
    data_valid = true;
    paramNr = 0; orderNr = 0;
    X = 0; Y = 0; Z = 0;
}

long CNCInterpreter::toSteps(char value)
{
    long steps=0;
    int num=0;

    steps = ((long)value-48)*stepsPerRot*gearFact/spindlePitch;
    return steps;
}

int CNCInterpreter::analyzeString(const char* input_string)
{
    int i = 0, neg = 0;
    static int exponent = 0;
    static signed long tempNr = 0; // static to not loose value!
    char last_char = '?';
    char c = input_string[0];

    for(i;i<4;i++) {
        order.orders[i] = '\\0';
        order.param[i] = 0;
    }
    i=0; orderNr = 0; paramNr = 0;

    while((c!='\\0'))
    {
        // input character is a number
        if(isdigit(c))
        {
            if (orderNr < 2) { // linewidth or ordercode?
                if(isupper(last_char)) {
                    tempNr = ((int)c-48);
                }
                if(isdigit(last_char)) {
                    tempNr = tempNr*10;
                    tempNr += ((int)c-48);
                }
            }
            else { // parameter needs to be converted to steps
                switch (exponent) {
                    case 0:
                        tempNr *= 10;
                        tempNr += toSteps(c);
                }
            }
        }
    }
}
```

```

        break;
    case 1:
        tempNr += (toSteps(c)/10);
        exponent++;
        break;
    case 2:
        tempNr += (toSteps(c)/100);
        exponent++;
        break;
    case 3:
        tempNr += (toSteps(c)/1000);
        exponent=0;
        break;
    }
}
}
else if( c=='.' )
{
    // last char was a number
    if(isdigit(last_char)) {
        exponent++;
    }
}
else if( c=='-' )
{
    neg = 1;
}
// input character is (A-Z) or ( or %
else if (isupper(c) || c=='(' || c=='%')
{
    // last char was a number
    if( isdigit(last_char) ) {
        order.lineNumber = tempNr;
        tempNr = 0; exponent = 0;
    }
    if(c == 'G' || c == 'M' || c == '%' || c == 'X' || c == 'Y' ||
c == 'Z' || c == 'F') {
        order.orders[orderNr++] = c;
    }
    if(c == '(') {
        order.orders[orderNr++] = 'H';
        while(orderNr<4) {
            order.orders[orderNr++] = '?';
        }
        return i;
    }
    if(c == ' ') {
        paramNr++;
        exponent = 0;
    }
}
else if(c == ' ')
{
    // last char was a number
    if( isdigit(last_char) ) {
        if(neg==1) {
            tempNr *= -1;
            neg=0;
        }
        order.param[paramNr++] = tempNr;
        tempNr = 0; exponent = 0;
    }
}

```

Anhang

```
        }
    }
    else
    {
        order.orders[orderNr++] = '?';
    }
    last_char = c;
    c = input_string[++i];
}
// last char was a number
if ( isdigit(last_char) ) {
    if(neg==1) {
        tempNr *= -1;
        neg=0;
    }
    order.param[paramNr] = tempNr;
}
while(orderNr<4) {
    order.orders[orderNr++] = '?';
}

return i;
}

char* CNCInterpreter::translate(const char* input_string)
{
    char buf[12];
    int e = 1;
    strcpy(order.translation, "\0");
    if(data_valid) {
        if(input_string[0]!='N') {
            data_valid = false;
            return "First char wrong";
        }
        else {
            analyzeString(input_string);

            if(order.orders[0] == 'G') {

                /** ToDo: Insert translations here **/

                switch(order.param[0]) {
                case 0:
                    strcat(order.translation, "@0M");
                    if(order.orders[e] == 'X') {
                        X = order.param[e++];
                    }
                    strcat(order.translation, ltoa(X,buf,10));
                    strcat(order.translation, ",5000,");
                    if(order.orders[e] == 'Y') {
                        Y = order.param[e++];
                    }
                    strcat(order.translation, ltoa(Y,buf,10));
                    strcat(order.translation, ",5000,");
                    if(order.orders[e] == 'Z') {
                        Z = order.param[e++];
                    }
                    strcat(order.translation, ltoa(Z,buf,10));
                    strcat(order.translation, ",5000,0,5000");
                    break;
                case 1:
```

Anhang

```
        strcat(order.translation, "@0M");
        if(order.orders[e] == 'X') {
            X = order.param[e++];
        }
        strcat(order.translation, ltoa(X,buf,10));
        strcat(order.translation, ",1000,");
        if(order.orders[e] == 'Y') {
            Y = order.param[e++];
        }
        strcat(order.translation, ltoa(Y,buf,10));
        strcat(order.translation, ",1000,");
        if(order.orders[e] == 'Z') {
            Z = order.param[e++];
        }
        strcat(order.translation, ltoa(Z,buf,10));
        strcat(order.translation, ",1000,0,1000");
        break;
    case 21:
        strcat(order.translation, "0"); break; // do nothing
(Metrisches System vorgeben)
    case 71:
        strcat(order.translation, "0"); break; // do nothing
(roughing = Vorschleifen)
    case 90:
        strcat(order.translation, "0"); break; // do nothing
(Absolute Werte)
    case 91:
        strcat(order.translation, "0"); break; // do nothing
(Relative Werte)
    case 94:
        strcat(order.translation, "0"); break; // do nothing
(Vorschub mm/min)
    case 95:
        strcat(order.translation, "0"); break; // do nothing
(Vorschub mm/Umdrehung)
    default:
        strcat(order.translation, "0");
    }
}
else if(order.orders[0] == 'M') {
    switch(order.param[0]) {
    case 2:
        strcat(order.translation, "0"); break; // do nothing
(Ende of Programm)
    case 3:
        strcat(order.translation, "0"); break; // do nothing
(Spindel im UZS)
    case 5:
        strcat(order.translation, "0"); break; // do nothing
(Fraser stoppen)
    case 7:
        strcat(order.translation, "0"); break; // do nothing
(Khlmittel ein)
    case 8:
        strcat(order.translation, "0"); break; // do nothing
(Schmierung ein)
    case 9:
        strcat(order.translation, "0"); break; // do nothing
(Khlmittel aus)
    case 30:
```

Anhang

```
        strcat(order.translation, "0"); break; // do nothing
(End of programm)
        default:
            strcat(order.translation, "0");
        }
    }
    else if(order.orders[0] == 'F') {
        strcat(order.translation, "Vorschubrate: ");
        strcat(order.translation, ltoa(order.param[0],buf,10));
    }
    else {
        strcat(order.translation, "0"); // do nothing
    }
    return order.translation;
}
}
else { return "Data not valid"; }
}
```


4.2.6 HTML CODE DES WEBSERVERS

```
<head>
  <title>CNC Webserver</title>
</head>
<body><h1>CNC Webserver</h1>
<p>Willkommen auf dem Webinterface zur PCB Fraese.</p>
<a href="upload.htm">Print hochladen</a></body>
```

```
<html><head>
<title>CNC Uploadcenter</title>
<script type="text/javascript">
    function fileSelected() {
        var file = document.getElementById('fileToUpload').files[0];
        if (file) {
            var fileSize = 0;
            if (file.size > 1024 * 32) {
                fileSize = 'Datei zu gross, bitte auf SD-Karte kopieren';
                document.getElementById('submitButton').disabled = true;
                document.getElementById('submitButton').style.backgroundColor
= "#b9b9b9";
                document.getElementById('submitButton').style.cursor =
"default";
            }
            else {
                fileSize = (Math.round(file.size * 100 / 1024) /
100).toString() + 'KB';
                document.getElementById('submitButton').disabled = false;
                document.getElementById('submitButton').style.backgroundColor
= "white";
                document.getElementById('submitButton').style.cursor =
"pointer";
            }

            document.getElementById('fileName').innerHTML = 'Name: ' +
file.name;
            document.getElementById('fileSize').innerHTML = 'Size: ' +
fileSize;
        }
    }

    function uploadProgress(evt) {
        if (evt.lengthComputable) {
            var percentComplete = Math.round(evt.loaded * 100 / evt.total);
            document.getElementById('progressNumber').innerHTML =
percentComplete.toString() + '%';
        }
        else {
            document.getElementById('progressNumber').innerHTML = 'unable to
compute';
        }
    }

    function uploadComplete(evt) {
        /* This event is raised when the server send back a response */

```

Anhang

```
        alert(evt.target.responseText);
    }

    function uploadFailed(evt) {
        alert("There was an error attempting to upload the file.");
    }

    function uploadCanceled(evt) {
        alert("The upload has been canceled by the user or the browser
dropped the connection.");
    }
</script>
<style>
    * {
        margin: 0;
        padding: 0;
    }
    body {
        text-align: center;
    }
    #wrapper {
        width: 480px;
        height: 200px;
        margin: 40px auto;
        padding: 20px;
        text-align: left;
        border-radius: 10px;
        background-color: #b9b9b9;
    }
    .row {
        padding: 10px;
    }
    .key {
        color: black;
        display: inline-block;
        padding: 2px 10px 3px 10px;
        font-size: 25px;
        line-height: 30px;
        text-shadow: none;
        letter-spacing: 0;
        bottom: 10px;
        position: relative;
        -moz-border-radius: 10px;
        -khtml-border-radius: 10px;
        -webkit-border-radius: 10px;
        border-radius: 10px;
        background: white;
        box-shadow: rgba(0, 0, 0, 0.1) 0 2px 5px;
        -webkit-box-shadow: rgba(0, 0, 0, 0.1) 0 2px 5px;
        -moz-box-shadow: rgba(0, 0, 0, 0.1) 0 2px 5px;
        -o-box-shadow: rgba(0, 0, 0, 0.1) 0 2px 5px;
        cursor: pointer;
    }
    #submitButton {
        color: black;
        background: white;
        display: inline-block;
        padding: 30px 20px;
        border-radius: 10px;
        cursor: pointer;
        float: right;
    }
```

Anhang

```
    }
</style>
</head>
<body><div id="wrapper">
    <form id="form1" enctype="multipart/form-data" method="post"
action="done.htm">
        <a id="left-init-key" href="index.htm"><span
class="key">&larr;</span></a>
        <div class="row">
            <label for="fileToUpload">CNC Datei</label><br />
            <input id="fileToUpload" type="file" name="fileToUpload"
onchange="fileSelected();" />
            <button id="submitButton">Hochladen</button>
        </div>
        <div class="row">
            <div id="fileName"></div>
            <div id="fileSize"></div>
        </div>
        <div class="row">
            <div id="progressNumber"></div>
        </div>
    </form></div>
</body></html>
```

4.2.6.3 DONE.HTM

```
<html><head><title>CNC Uploadcenter</title>
    <script type="text/javascript">
        function scrollDown() {
            var objDiv = document.getElementById("content");
            objDiv.scrollTop = objDiv.scrollHeight;
        }
    </script>
<style>
    * {
        margin: 0;
        padding: 0;
    }
    body {
        text-align: center;
    }
    #wrapper {
        width: 480px;
        height: 400px;
        margin: 40px auto;
        padding: 20px;
        text-align: left;
        border-radius: 10px;
        background-color: #b9b9b9;
    }
    #content {
        height: 370px;
        overflow-y: scroll;
    }
    .key {
        color: black;
        display: inline-block;
        line-height: 30px;
        text-shadow: none;
```

Anhang

```
        letter-spacing: 0;
        position: relative;
        bottom: 10px;
        -moz-border-radius: 10px;
        -khtml-border-radius: 10px;
        -webkit-border-radius: 10px;
        border-radius: 10px;
        box-shadow: rgba(0, 0, 0, 0.1) 0 2px 5px;
        -webkit-box-shadow: rgba(0, 0, 0, 0.1) 0 2px 5px;
        -moz-box-shadow: rgba(0, 0, 0, 0.1) 0 2px 5px;
        -o-box-shadow: rgba(0, 0, 0, 0.1) 0 2px 5px;
        cursor: pointer;
        background: white;
        font-size: 25px;
    }
    #buttonheader {
        margin-top: 0px;
    }
    #backkey {
        text-align: left;
        padding: 2px 10px 3px 10px;
    }
    #logoutkey {
        text-align: right;
        padding: 2px 20px 3px 20px;
        margin-left: 315px;
    }
</style>
</head>
<body onload="scrollDown();"><div id="wrapper">
    <div id="buttonheader">
        <a href="upload.htm"><span id="backkey" class="key">&larr;</span></a>
        <a href="logout.htm"><span id="logoutkey"
class="key">Logout</span></a></div>
    <div id="content">
```

4.2.6.4 LOGOUT.HTM

```
<html><head><title>CNC Uploadcenter</title>
<style>
    * {
        margin: 0;
        padding: 0;
    }
    body {
        text-align: center;
    }
    #wrapper {
        width: 480px;
        height: 200px;
        margin: 40px auto;
        padding: 20px;
        border-radius: 10px;
        background-color: #b9b9b9;
    }
</style>
</head>
<body><div id="wrapper">
    <br /><h2>Logout</h2>
```

Anhang

```
<br /><p>Zum ausloggen bitte Browser schliessen</p>
</div></body></html>
```