Entwicklung eines Mikrocontroller-Systems zur sicheren Abschaltung eines Roboterantriebs

Bachelorarbeit
von
Markus Nuber

Hochschule:	Ravensburg-Weingarten 
Studiengang:	Elektro- und Informationstechnik 
Studienrichtung:	Automatisierungstechnik
Name:	Markus Nuber
Matr.Nr.	17487
Adresse:	Am Dambach 1 88099 Neukirch
E-Mail:	m_nuber@web.de
Abgabetermin:	26. Februar 2010
Betreuer:	Dipl.Wirtsch.-Ing(Fh) Philipp Ertle, M.Sc.
Erstkorrektor:	Prof. Dr.-Ing. Holger Voos
Zweitkorrektor:	Prof. Dr.-Ing. Walter Ludescher

Erklärung

Die vorliegende Bachelorarbeit wurde von mir selbst verfasst und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt.

Neukirch, im Februar 2010

Markus Nuber

Danksagung

Diese Bachelorarbeit wurde im Zeitraum vom 1.Dezember 2009 bis 28.Februar 2010 im Mobile Robotik Labor der Hochschule Ravensburg-Weingarten angefertigt. Ich möchte mich für die Aufgabenstellung und Bereitstellung des Arbeitsplatzes bedanken.

Ein besonderer Dank gilt Philipp Ertle, der mich während dieser Zeit unterstützt hat. Ein weiterer Dank gilt allen Mitarbeitern des Mobile Robotik Labors, die mir helfend zur Seite standen, wenn ich Fragen oder Probleme hatte.

Meinen Betreuern Prof. Dr.-Ing. Holger Voos und Prof. Dr.- Ing. Walter Ludescher gilt ebenfalls mein Dank.

Inhaltsverzeichnis

1 Kurzfassung

Die vorliegende Arbeit liefert das Ergebnis zur Entwicklung eines „Safety-Boards“. Dieses wird in den mobilen Roboter „Pioneer 3“ eingebaut. Mit diesem Board soll der sichere Antrieb des Roboters gewährleistet werden. Über das Board findet eine Abfrage statt, ob das Hauptprogramm unter definierten Bedingungen funktioniert. Des Weiteren kann eine maximale Geschwindigkeit vorgegeben werden.

Zunächst wird in der Board Version 1.0 nur eine Heartbeat-Abfrage gemacht. Dies geschieht einmal über einen Mikrocontroller und parallel dazu über einen Tiefpass-Filter. Erst wenn vom Mikrocontroller und vom Tiefpass-Filter ein korrektes Heartbeat-Signal erkannt wird, schaltet ein Relais. Dieses ist zwischen Steuerplatine und Motor geschaltet.

Solange das Relais nicht aktiviert ist, kann kein Motorstrom fließen. Das heißt der Stromkreis ist unterbrochen und der Roboter bleibt stehen. Sobald das Hauptprogramm aktiviert ist und kein Fehler auftritt, muss dieses ein Heartbeat-Signal senden. Wird dieses vom „Safety-Board“ erkannt, gibt es die Motorsteuerung frei.

In der erweiterten Version 1.1 wird die Version 1.0 um eine Geschwindigkeitsbeschränkung und ein Notausschalter erweitert.

Mit dem Notausschalter kann der Roboter manuell zum Stillstand gebracht werden. Das Motor PWM Signal wird vor dem „Motor-Power Distribution Board“ abgegriffen. Mit einem vom Mikrocontroller vergebenen PWM-Signal wird dieses durch eine logische UND-Funktion verbunden. Das nun resultierende Signal wird an das „Motor-Power Distribution Board“ weitergegeben. Sobald nun entweder das Originalsignal oder das PWM-Signal vom µC auf Null geht, folgt die Motor-PWM diesem Signal und wird ebenfalls „Null“. So kann eine maximale Geschwindigkeit vom „Safety-Board“ vorgegeben werden.

Zum Testen der Grundfunktionen des Controllers und des Boards wird ein Testprogramm in „C“ geschrieben.

Zum Abschluss der Arbeit werden noch einige Testfälle betrachtet und die Funktion der einzelnen Komponenten des „Safety-Board“ überprüft.

2 Einleitung

2.1 Motivation

Mobile moderne Serviceroboter sind prinzipiell lernfähig. Diese Lernfähigkeit ermöglicht es ihnen, einfache Aufgaben erledigen zu können. Das heißt, die Steuerung des Roboters ändert sich im Laufe der Zeit. Je nach Arbeitseinsatz und Umgebung des Roboters wird das Steuerprogramm komplexer. Der Entwickler kann nicht mehr vorhersehen, wohin sich das System entwickelt.

Damit ein sicherer Betrieb gewährleistet werden kann, muss sichergestellt sein, dass bestimmte Zustände nicht eintreten können. Dies kann entweder durch eine Spezifikation der Software oder durch eine externe Schaltung erreicht werden. Diese Schaltungen geben eine Art Rahmen vor (z.B. max. Geschwindigkeit, Abstand zum nächsten Hindernis,…), in welchem sich der Roboter dann bewegen darf. Um den sichereren Betrieb zu gewährleisten, müssen die zusätzlichen Schaltungen zuverlässig funktionieren.

Im Bereich mobiler Roboter sind bis zum jetzigen Zeitpunkt, sehr wenig Sicherheitsspezifikationen vorhanden, weil diese erst in den letzten Jahren immer mehr an Bedeutung gewinnen. Mit dieser Arbeit soll eine, vom Steuerprogramm externe, Schaltung entstehen, die in den Roboter eingebaut werden kann. Durch Einbau dieser Schaltung in den mobilen Roboter, soll eine Grundsicherheit gewährleistet werden können.

2.2 Aufgabenstellung

Es soll eine Hardwareschaltung zum sicheren Abschalten eines Roboterantriebs entwickelt werden. Diese soll in den mobilen Roboter „Pioneer3“ eingebaut werden.

Das „Safety-Board“ soll folgende Aufgaben erfüllen:

- Das korrekte Vorhandensein eines Heartbeat-Signals soll überwacht werden
- Vorgabe einer maximalen Geschwindigkeit, die der Roboter nicht überschreiten darf
- Vorgabe eines maximalen Neigungswinkel (optional)

Der Motorantrieb darf erst freigegeben werden, wenn ein korrektes Hearbeat-Signal anliegt. Die Schaltung überprüft des Weiteren, ob eine maximale Geschwindigkeit nicht überschritten wird.

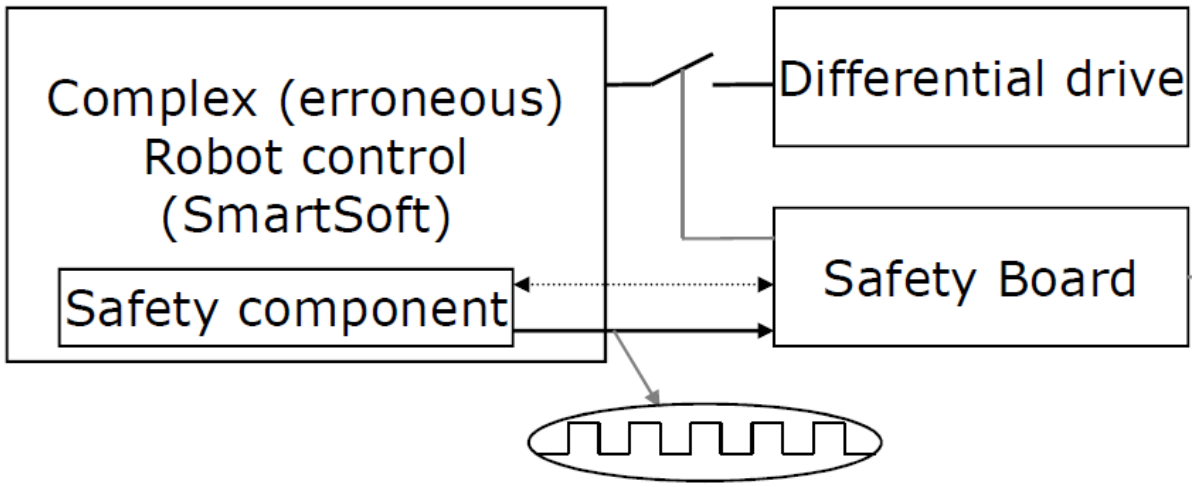


Abbildung 1: Blockschaltbild Aufgabenstellung [1]

2.3 Grundlagen und Begriffe

2.3.1 Mobiler Roboter: Pioneer 3

Der Pioneer ist ein mobiler Roboter der Firma „ActivMedia Robotics“. Die Pioneer Familie umfasst die mobile Roboter Pioneer 1,Pioneer AT, Pioneer 2-DX, -DXf, -CE, -AT,Pioneer 2-Dx8/Dx8 Plus und –At8/At8 Plus, so wie die neueste Generation den Pioneer 3-Dx und –At. Manche sind mit Zweirad- andere mit Vierradantrieb ausgestattet.

Der Pioneer 3 besitzt vier luftbereifte Räder. Seine Höchstgeschwindigkeit beträgt 0,7 m/s. Der Antrieb ist differential, d.h. die linken und rechten Räder werden unabhängig voneinander angesteuert. Dadurch ist es möglich den Roboter auf der Stelle drehen zu lassen. Um die nähere Umgebung abzumessen, besitzt der Roboter Ultraschall-Abstandssensoren. Zur Basisausstattung gehören ein festes Aluminiumgehäuse, DC-Motoren, Motoransteuerung und Antriebselektronik, sowie eine Batterie. Je nach Anwenderwunsch kann er mit einer Kamera oder einem Laserscanner erweitert werden. Um den Roboter testen zu können, wird das Demo-Programm „Aria“ mitgeliefert. Mit diesem kann vom PC aus über die Pfeiltasten die Fahrtrichtung vorgegeben werden.

Das Ziel des ZAFH[1] ist, dem Roboter kleine Serviceaufgaben beizubringen. Wie zum Beispiel: Kaffee bringen, Tisch decken, Gegenstände bringen uvm. [2]

[1] ZAFH=Zentrum für angewandte Forschung an Fachhochschulen

Abbildung 2: Pioneer3

2.3.2 Sicherheit

Es gibt zwei Varianten der Sicherheit. Diese beiden Gebiete lassen sich am besten mit den englischen Begriffen „security“ („Betriebsschutz“) und „safety“ („Betriebssicherheit“) beschreiben. Im Fall „security“ wird das Objekt (in unserm Fall der Roboter) vor der Umgebung geschützt. Im zweiten Fall ist genau das Gegenteil der Fall. Die Umgebung wird vor dem Roboter geschützt.

In der vorliegenden Arbeit wird das Thema „safety“ behandelt. Hierbei stellt sich jedoch die Frage, wann ein System sicher ist. Sicherheit ist ein relativer Zustand. Von Sicherheit ist die Rede, wenn unvertretbare Risiken ausgeschlossen sind. Es muss festgelegt werden, bis zu welchem Zustand man ein System als sicher betrachtet, das heißt ab welchem Zustand ein tolerierbares Risiko erreicht ist.

Eine hundertprozentige Sicherheit wird es nie geben, weil immer nicht vorhersehbare oder eingeplante Ereignisse auftreten können. [3]

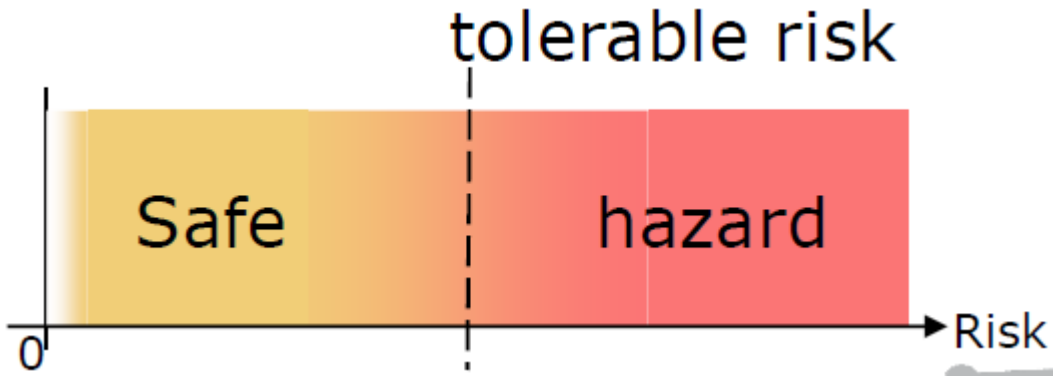


Abbildung 3: Sicherheit

2.3.3 Mikrocontoller

Mikrocontroller (µC) sind programmierbare integrierte Schaltungen. Ein großer Vorteil von µC ist, dass je nach Controller Speicher, Digital- und Analog Ein- und Ausgänge, Zähler, u.v.m. meist auf einem Chip integriert sind. Des Weiteren weisen sie meist unkomplizierte Architekturmerkmale auf. Außerdem werden sie in großer Stückzahl produziert und sind daher günstig erhältlich.

Mikrocontroller besitzen kein eigenes Betriebssystem, haben nur einen begrenzten Funktionsumfang und verfügen über keinen umfangreichen Massenspeicher (Harddisc). Auf die Verwendung von grafischen Benutzeroberflächen wird häufig verzichtet.

Für viele kleinere Anwendungen sind diese Komponenten nicht notwendig, daher erfreut der µC sich eines weiten Einsatzbereichs. Dieser reicht von der Anwendung im Fahrzeugbereich z.B. ABS-System über Steuerung von Haushaltshaltsgeräten (z.B. Spülmaschine) bis hin zur Steuerung kleiner Systeme. In der Robotertechnik findet der Mikrocontroller eine große Beliebtheit.

Die klassische Programmiersprache ist „Assembler“. Diese Sprache rückt aber immer mehr in den Hintergrund. Neuere Controller haben für Hochsprachen optimierte Befehlssätze. Daher kann ein µC problemlos in „C“ programmiert werden.

Es wird unterschieden zwischen 4bit, 8bit, 16bit und 32bit Mikrocontrollern. Dies ist die Breite des internen Datenbusses.

Einige beliebte µC-Familien sind: 8051,AVR,PIC,68HC,ARM [4]

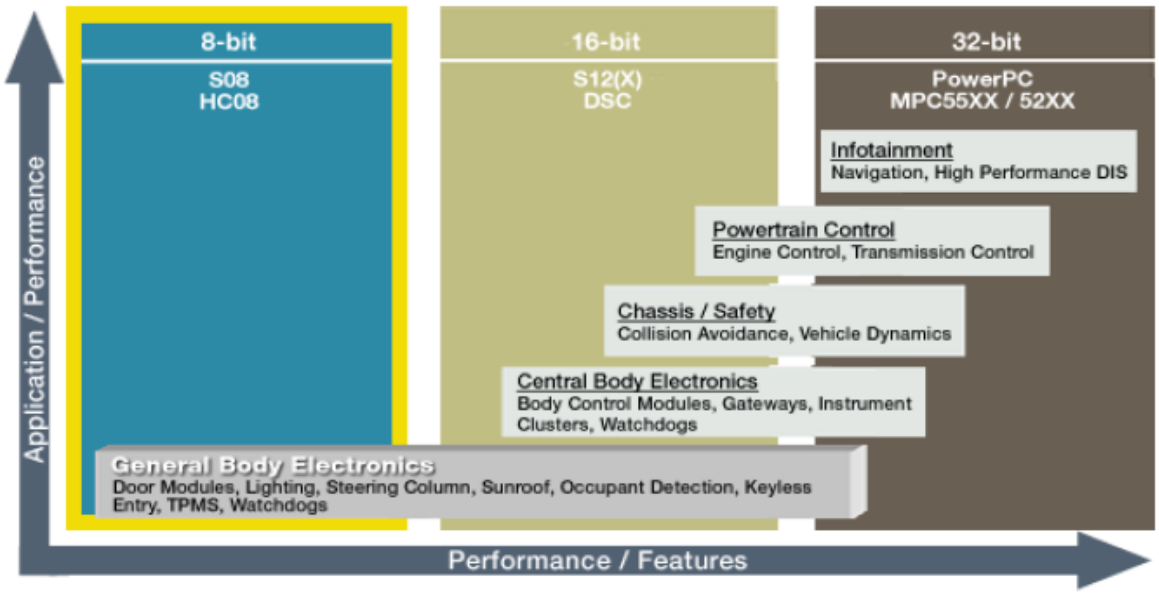


Abbildung 4: Typische Anwendung für Mikrocontroller [5]

2.3.4 Heartbeat Signal

Heartbeat heißt ins Deutsche übersetzt „Herzschlag“. Bei einer Netzwerkverbindung zwischen mindestens zwei Systemen kann mit einem sogenannten „Heartbeat“ kommuniziert werden. Die Systeme verwenden das Heartbeat-Signal zur gegenseitigen Information. Meist ob das Betriebssystem noch arbeitet. Daher der Begriff „Heartbeat“.

Es besteht die Möglichkeit, dass sich nur ein System beim anderen melden muss oder aber eine gegenseitige Abfrage herrscht. Sobald das Signal ausbleibt, geht das andere System davon aus, dass sein Partner nicht mehr zur Verfügung steht. Dieses kann die für diesen Fall vorgesehenen Maßnahmen vornehmen.

Ein Heartbeat ist meist so definiert, dass in einem gewissen Abstand ein Impuls gesendet wird. Es kann z.B. ein optimales Rechtecksignal mit einer Frequenz von 50 Hz und einer Pulsbreite von 50% sein. Ebenso ist denkbar, dass nur jede Sekunde ein Impuls von 5ms kommen muss. Dieses muss dann vom jeweiligen Programmierer definiert werden.

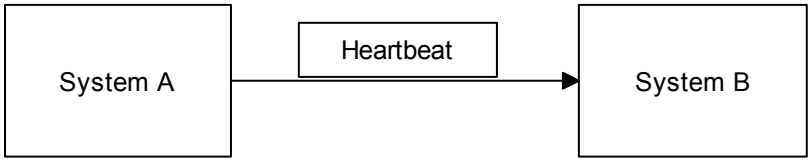


Abbildung 5: Heartbeat

In Abbildung 5 muss sich das System A bei System B mit einem Heartbeat melden.

2.3.5 Redundanz

Von Redundanz spricht man, wenn mehrere Systeme für die gleiche Aufgabe vorhanden sind. Man unterscheidet zwischen Hardware-, Software, Zeit- und Informationsredundanz.

In der Sicherheitstechnik wird Redundanz oft eingesetzt, um ein System sicherer zu machen. Dabei arbeiten mehrere Schaltkreise (bzw. Softwaretask) das gleiche Ereignis ab. Falls ein Kreis ausfällt kann somit gewährleistet werden, dass das System dennoch arbeitet. Dies kann mit einem Oder-Gatter verglichen werden. Eine andere Möglichkeit besteht darin, dass beide Kreise ein „true“ liefern müssen, damit das System läuft. Dieses kann mit einem Und-Gatter verglichen werden und wird häufig bei fehlerkritischen Ereignissen verwendet.

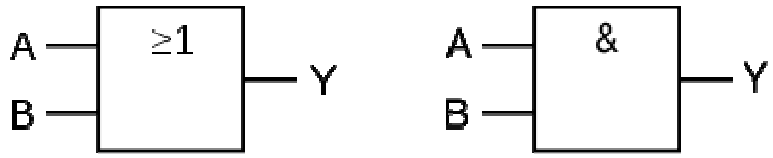


Abbildung 6: Oder bzw. Und Gatter

3 Entwicklung des Safety-Boards

3.1 Einführung

Um den Roboter sicherer zu machen, werden Rahmenbedingungen auferlegt. Diese werden über das Safety-Board abgefragt. Erst wenn das Board die Motorkomponente frei gibt, darf sich der Roboter in Bewegung setzen. Die Rahmenbedingungen sind eine vorgegebene Maximalgeschwindigkeit und das korrekte Vorhandensein des Heartbeat-Signals vom Hauptprogramm.

Damit die Sicherheit gewährleistet wird, muss das Safety-Board selbst zuverlässig funktionieren. Es nützt nichts, wenn Rahmenbedingung gestellt werden, aber mit dem Safety-Board eine größere Fehlerwahrscheinlichkeit auftritt, als ohne.

3.2 Funktion

Das Safety-Board erwartet vom Hauptprogramm ein Heartbeat-Signal. Die Abfrage des Signals erfolgt einmal über eine analoge Schaltung und parallel dazu über den Mikrocontroller „Atmega32“. Die analoge Schaltung ist über einen Tiefpassfilter realisiert.

Der Stromkreis zwischen Motor-Ansteuerung und Motor kann unterbrochen werden. Erkennt das Board nun einen unzulässigen Zustand (z.B. eine fehlerhafte Meldung vom Hauptprogramm), wird der Stromkreis unterbrochen und der Roboter kommt zum Stehen. Ist der Stromkreis unterbrochen, hat das Hauptprogramm (Robotersteuerung) keinen Einfluss mehr auf den Roboterantrieb. Es muss zuerst der Fehler behoben werden, damit sich der Roboter wieder in Bewegung setzt.

Die Robotersteuerung gibt an die Motoransteuerung ein Motor PWM[2] Steuersignal weiter. (Abbildung 7 grau gezeichnet). Dieses PWM-Signal gibt die Geschwindigkeit des Motors vor. Um die Geschwindigkeit kontrollieren zu können, wird dieses Signal vom Safety-Board abgegriffen. Dort wird es dann verarbeitet und an die Motoransteuerung weitergeleitet (Abbildung 7 schwarz gezeichnet). Das Signal kann nun ohne Veränderung weitergegeben werden oder es kann die Pulsbreite über das Safety-Board verkürzt werden.

In der Version V1.0 erfolgt nur eine Heartbeat-Abfrage. Nach dem Version V1.0 funktionierte, erweiterte ich das Board zusätzlich mit einer Geschwindigkeitsbegrenzung zur Board Version V1.1.

[2] PWM=Pulsweitenmodulation

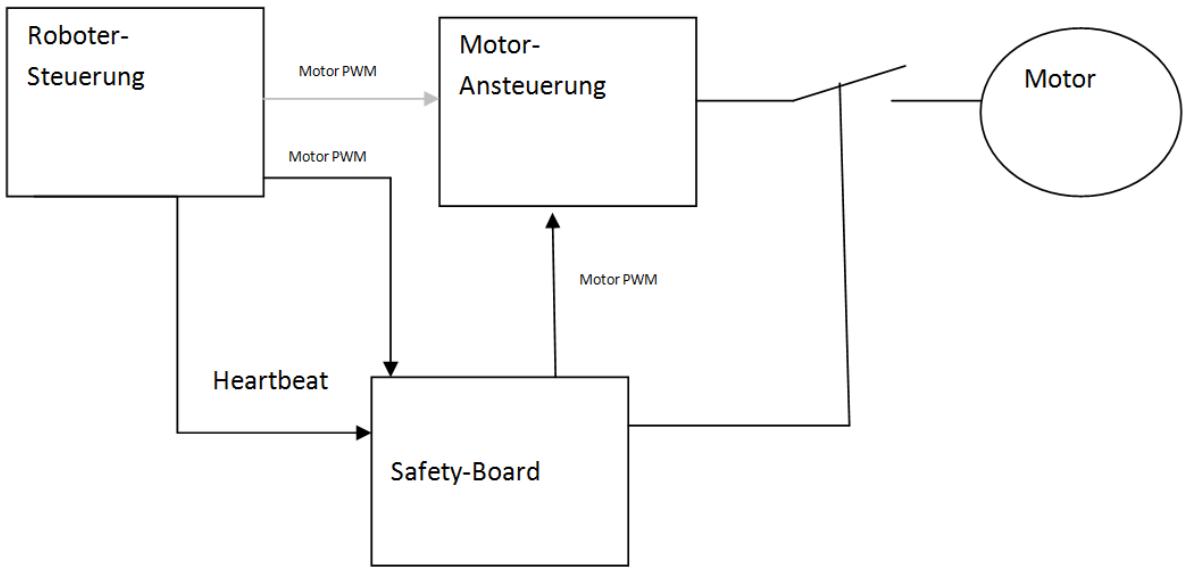


Abbildung 7: Blockschaltbild Einbau des Safety-Boards

3.3 Definieren des Heartbeat

Eine Rechteckspannung mit einer Frequenz von f_H=100 Hz mit einer Pulsbreite von 50% wird als Heartbeat erwartet. Mit einem Spitzenwert U_{pp}=5V.

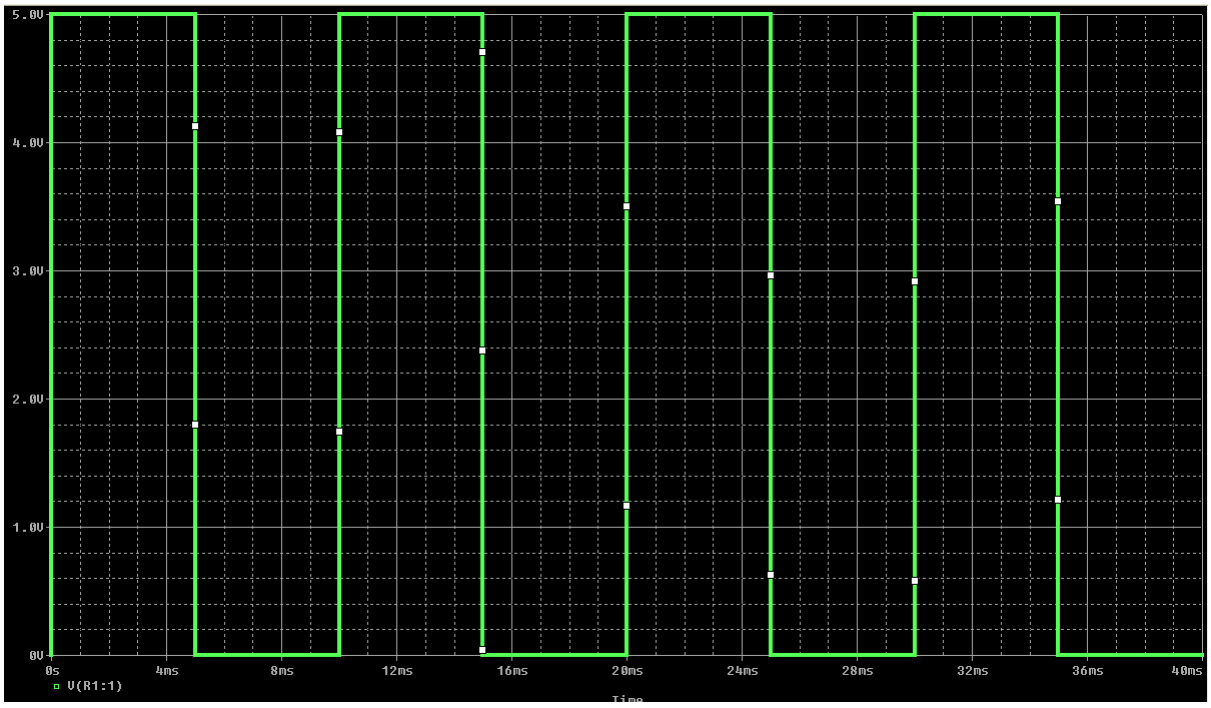


Abbildung 8: erwartetes Heartbeat-Signal

3.4 Tiefpass-Filter

Um die Heartbeat-Abfrage analog zu gestalten wird ein Tiefpassfilter zweiter Ordnung eingesetzt. Ein „Sallen-Key-Filter“ (Abbildung 9) findet Verwendung.

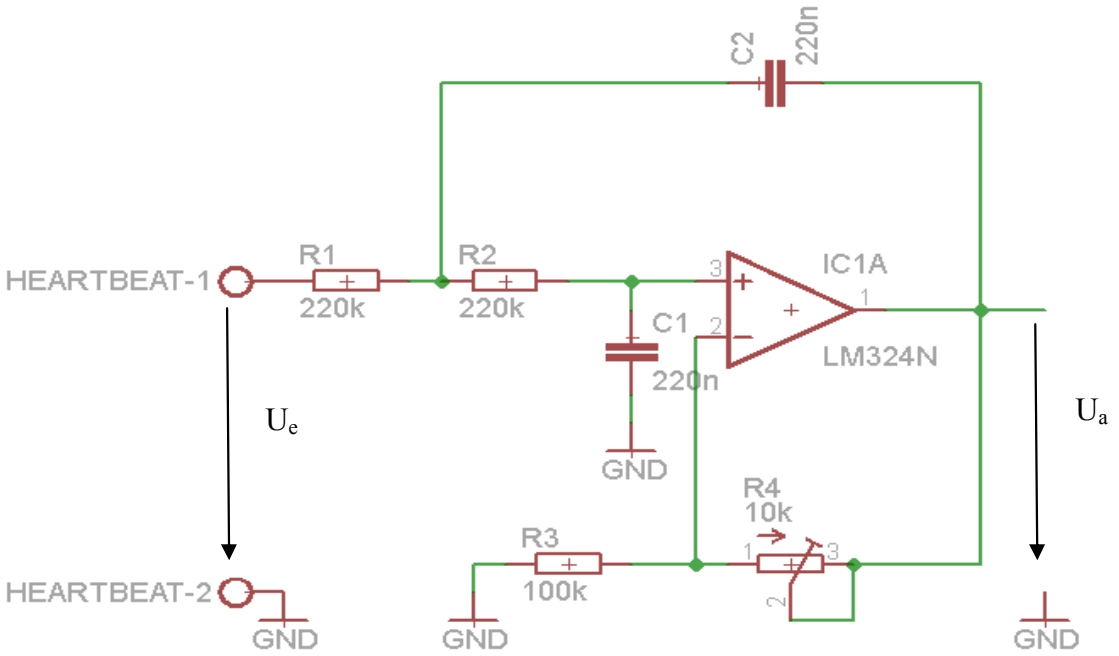


Abbildung 9: Tiefpassfilter des Safety Boards

Ist R_1=R_2 und C_1=C_2 wird durch den Filter der Mittelwert der Spannung ausgegeben. Um zu prüfen, ob ein Heartbeat anliegt, wird genau dieser Effekt ausgenützt. Liegt an U_e ein erwartungsgemäßes Heartbeat von einer Pulsbreite von 50% und U_{pp}=5V an, so ist der Ausgang U_a=2,5V.

Mit Hilfe von „OrCad[3]“ wurde das Filter so dimensioniert, das ein optimaler Mittelwertbilder entsteht. Nach mehreren Simulationen wurden die Werte R_1=R_2=220 kΩ und C_1=C_2=2220nF ermittelt. Über das Verhältnis R_3 zu R_4 kann eine Feinabstimmung des Endwertes von U_a erfolgen. Je größer R_4 gewählt wird, desto größer wird U_a. Bei einem Verhältnis von R_4 zu R_1 von 5 zu100 wird annähernd ein Endwert von U_a=2,5V erreicht. Wird R_4 zu groß gewählt, beginnt das System über zu schwingen.

Die Abfrage mit Hilfe des Tiefpasses hat den Vorteil, dass nur auf die Höhe des Spannungswerts abgefragt wird. Das heißt, es kann ebenso einfach auf ein Heartbeat mit U_{pp}=1V oder

[3] OrCad ist ein Programm zur Entwicklung elektrischer Schaltungen von der Firma „Cadence Design System“. Es besteht aus einem Tool zur Schaltplaneingabe und ein Tool zur Simulation der Schaltung

einer Pulsbreite von 10% abgefragt werden. Es wird dann als Mittelwert nicht mehr U_a= 2,5V sondern U_a=0,5V erreicht.

Der Mittelwert wird nach 5-mal der Zeitkonstante τ =R*C erreicht. Beim Einschalten des Systems muss gewartet werden bis der Spannungswert U_a eine vorgegebene minimale Spannung U_{min}=2,25V überschreitet.

Berechnung nach welcher Zeit der Endwert U_a=2.5 V erreicht wird:

$$5*\tau=5*(R*C)=5*(220k\Omega*220nF)=5*48,4\ ms=242\ ms$$

Es wird nach 242ms U_a = 2,5 V erreicht.

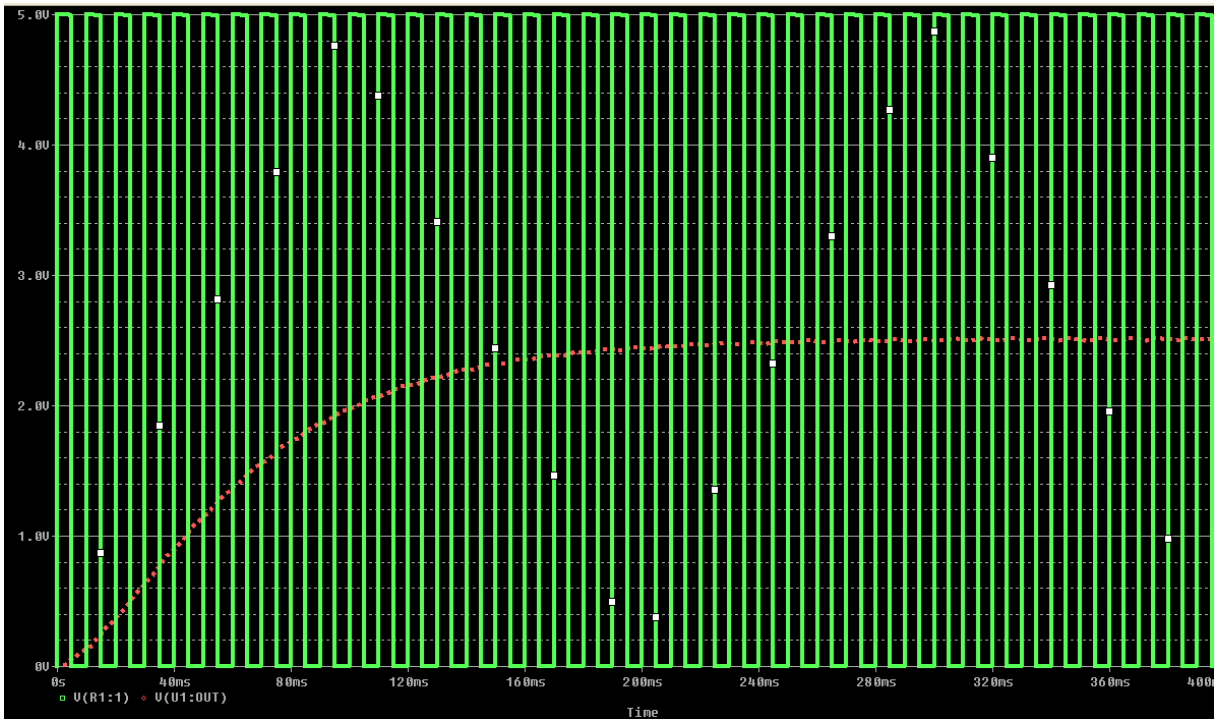


Abbildung 10: Simulation Tiefpass

Abbildung 10 stellt das Simulationsergebnis des Filters dar. V(R1:1) ist der Eingang U_e und soll das Heartbeat simulieren. V(V1:OUT) ist die Ausgangsspannung U_a. Es ist zu sehen, dass nach ca. 240ms ein Mittelwert von U_a=2,5 V erreicht wird.

Als Operationsverstärker wird ein LM324N verwendet.

Abbildung 11 zeigt die Spannungswerte des aufgebauten Filters. Mit einem Frequenzgenerator wird das Heartbeat simuliert. Die Signale werden auf dem Oszilloskop dargestellt. CH1 ist des Eingangssignal und CH2 das Ausgangssignal. Wie in der Simulation ist der Ausgangswert des Tiefpasses der Mittelwert der Eingangsspannung.

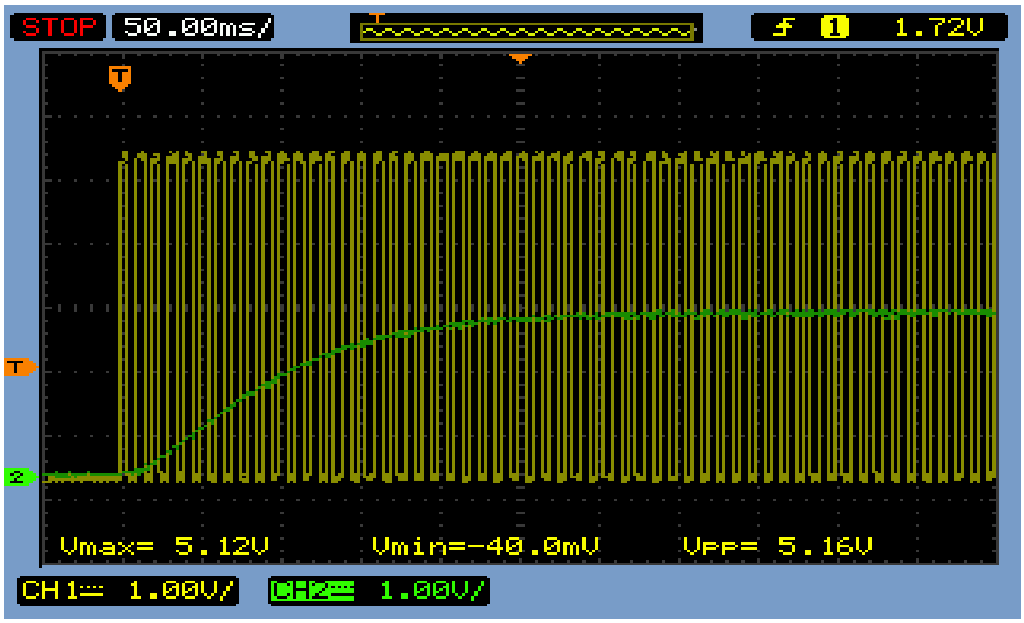


Abbildung 11: Tiefpass

3.5 Erkennen ob ein Heartbeat vorhanden ist

Wie im vorigen Abschnitt erklärt, ist der Ausgang des Tiefpasses U_a=2.5V, wenn ein Heartbeatsignal anliegt. Um dies zu überprüfen werden dem Tiefpass zwei Komparatoren nachgeschaltet. Diese zwei Komparatoren geben ein Fenster vor, in dem das Heartbeatsignal als korrekt anerkannt wird. Komparator 2 (Komp2) prüft, ob ein minimaler Spannungswert überschritten ist. Ob ein maximaler Spannungswert nicht überschritten wird, kann mit Komparator 1(Komp1) geprüft werden.

Befindet sich U_a im Bereich dieses Fensters, gilt der Heartbeat als korrekt. Liefert Komp2 eine Logische „1“ und Komp1 eine Logische „0“ ist der Heartbeat in Ordnung. Liegt kein Heartbeat an, so gibt der Komp2 eine „Null“ aus. Falls eine konstante 1 als Heartbeat anliegt, liefert sowohl Komp1 als auch Komp2 eine „1“. In diesem Fall ist das Heartbeat Signal falsch. Es wird davon ausgegangen, dass das Hauptprogramm nicht mehr richtig arbeitet. Ein Zustand, der nicht auftreten darf ist, dass Komp1 eine „Eins“ und Komp2 eine „Null“ liefert. Tritt dieser Fall auf, muss ein Fehler in der Komparatorenschaltung vorliegen. Es ist nicht möglich, dass der maximale Wert überschritten wird und der minimale Wert nicht.

Komp1	Komp2	Heartbeat Ok
0	0	0
0	1	1
1	0	ungültig
1	1	0

Tabelle 1: Wahrheitstabelle Heartbeat

Der minimale Spannungswert soll über das Potentiometer R_{14} zwischen U_{min}= 2,0V und 3,0 V einstellbar sein. Der maximale Spannungswert soll zwischen U_{max} =2,5V und 4,2 V einstellbar sein. Dies ist über das Potentiometer R_9 möglich.

Es wird $R_{12}=R_7=10k\Omega$ gewählt.

Berechnung von R_{13} und R_{14}:

Um R_{13} zu bestimmen wird $R_{14}=0$ und U_{min} =2,0V gesetzt.

$$\frac{U_{min}}{U_{ges}} = \frac{R_{13}}{R_{12}+R_{13}} \Rightarrow \frac{2V}{5V} = \frac{R_{13}}{10k+R_{13}} \Rightarrow R_{13}=6{,}66k\Omega$$

Gewählt: $R_{13}=6{,}8k\Omega$

Um R_{14} zu bestimmen wird $U_{min}=3{,}0V$ gesetzt und $X=R_{13}+R_{14}$

$$\frac{U_{min}}{U_{ges}} = \frac{X}{R_{12}+X} \Rightarrow \frac{3V}{5V} = \frac{X}{10k+X} \Rightarrow X=15k\Omega$$

$R_{14}=X-R_{13}=15k\Omega-6{,}8k\Omega=8{,}2k\Omega$

Berechnen von R_8 und R_9:

Um R_{13} zu bestimmen wird $R_9=0$ und U_{max} =2,5V gesetzt.

$$\frac{U_{max}}{U_{ges}} = \frac{R_8}{R_7+R_8} \Rightarrow \frac{2{,}5V}{5V} = \frac{R_8}{10k+R_8} \Rightarrow R_8=10\ k\Omega$$

Um R_{14} zu bestimmen wird $U_{max}=4{,}2V$ gesetzt und $X=R_8+R_9$

$$\frac{U_{max}}{U_{ges}} = \frac{X}{R_7+X} \Rightarrow \frac{4{,}2V}{5V} = \frac{X}{10k+X} \Rightarrow X=52{,}5k\Omega$$

$R_9=X-R_8=52{,}5k\Omega-10k\Omega=42{,}5k\Omega$

Gewählt: $R_9=47k\Omega$

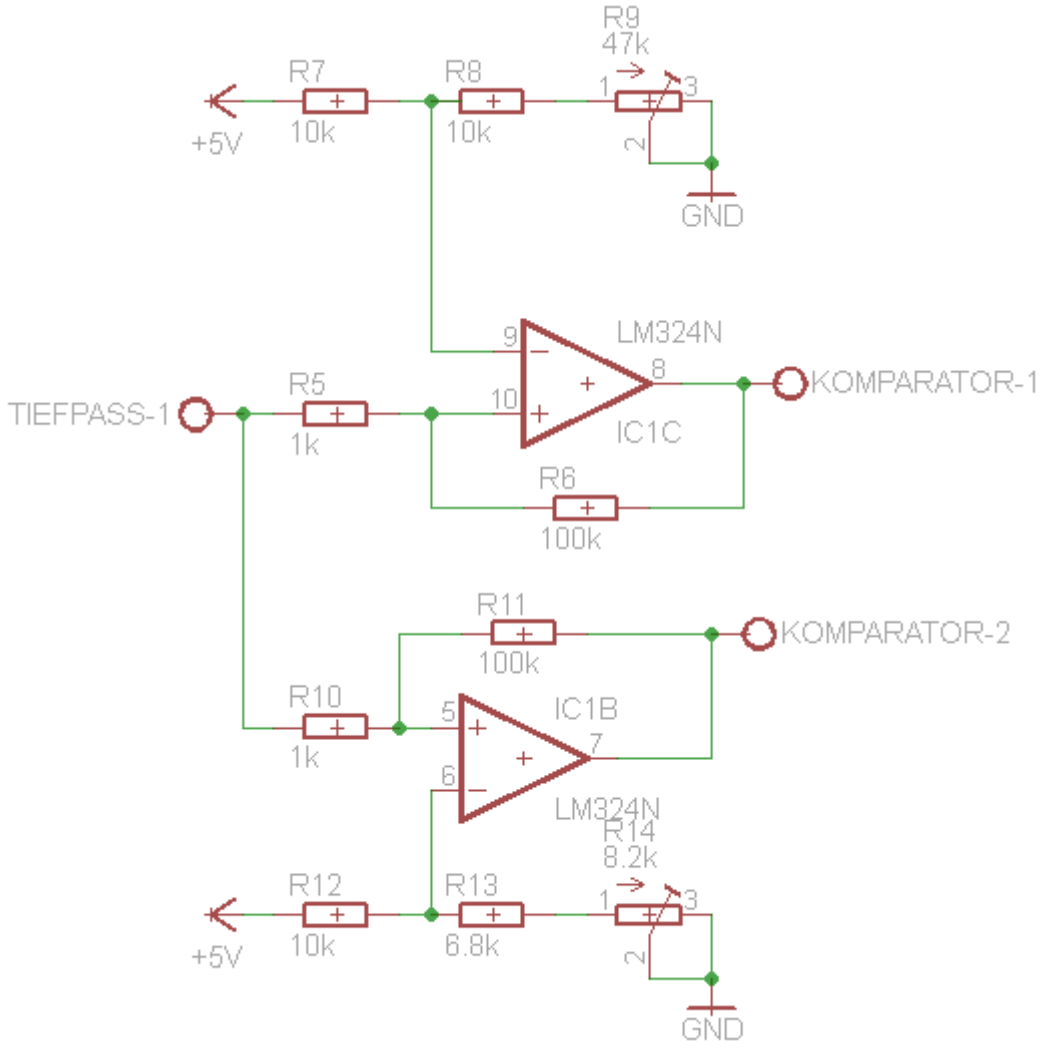


Abbildung 12: Komparatorenschaltung

3.6 Freigabe des Antriebes

Der Antrieb wird freigegeben, wenn die Schaltung einen korrekten Heartbeat erkennt. Er muss von der analogen Schaltung und vom Mikrocontroller erkannt werden. Danach schalten die Relais, welche zwischen Motoransteuerung und Motor eingebaut sind (siehe Abbildung 7). Die analoge Schaltung besteht aus dem Tiefpass mit nachgeschalteten Komparatoren.

µC	analog	Relais
0	0	0
0	1	0
1	0	0
1	1	1

Tabelle 2: Wahrheitstabelle Relais

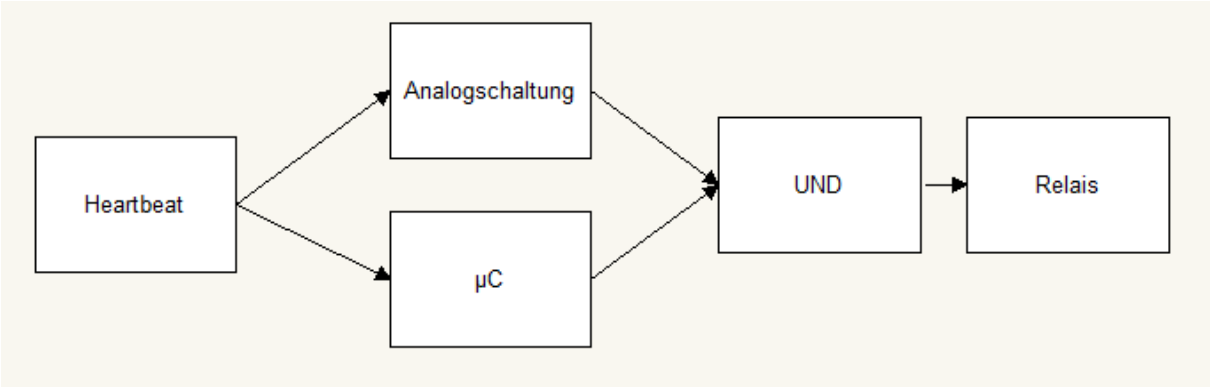


Abbildung 13: Blockschaltbild: Freigabe des Antriebs

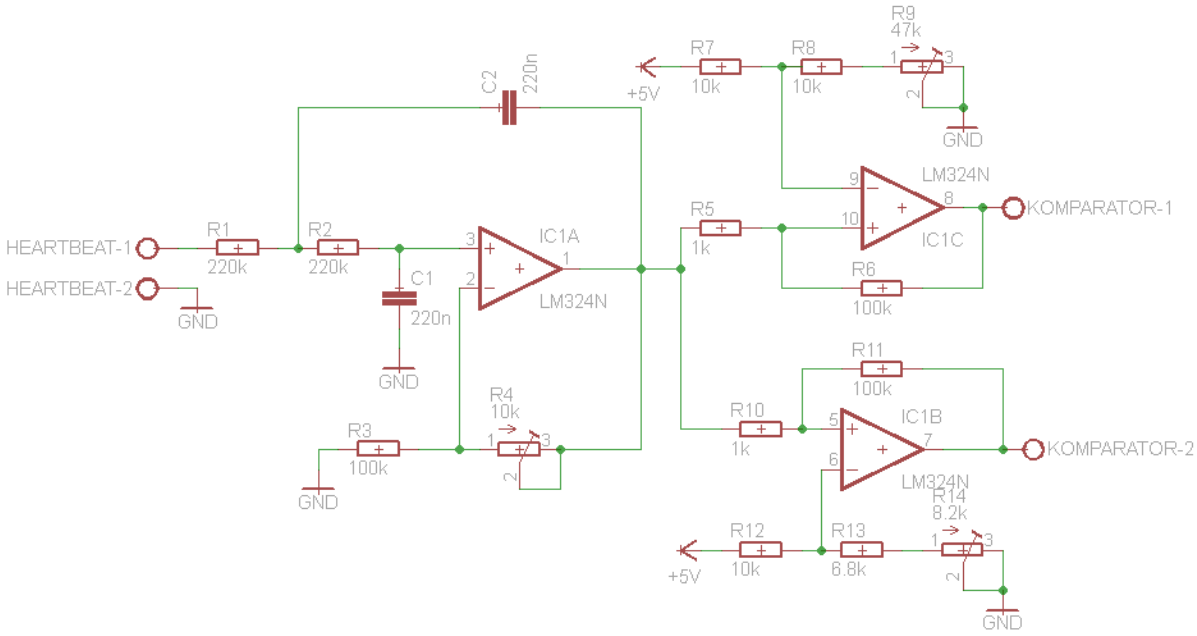


Abbildung 14: Analogschaltung

Die beiden Wahrheitstabellen „Heartbeat“ und „Relais“ werden zu einer zusammengefügt um Bauteile zu sparen. Als Verknüpfungsglied wird der 3-fach Input NAND Baustein „7410“ verwendet. Ein NAND-Gatter ist von Vorteil, weil dadurch die Negierung von Komp1 einfach implementiert werden kann.

µC	Komp1	Komp2	Relais
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Tabelle 3: Wahrheitstabelle Schalten

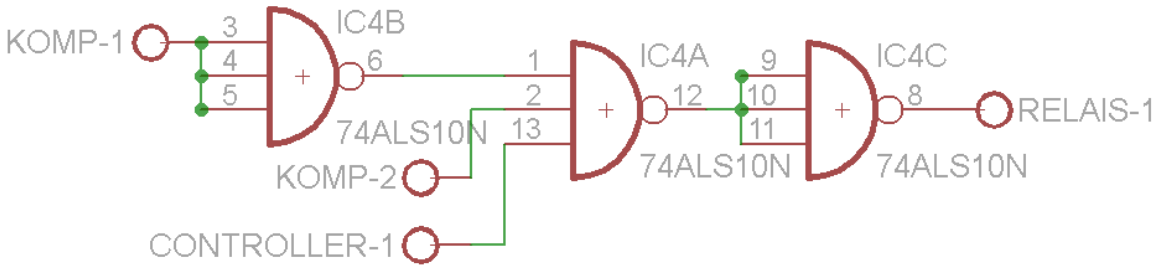


Abbildung 15: Logik-Schaltung

µC	Komp1	Komp2	IC4B out	IC4A out	Relais
0	0	0	1	1	0
0	0	1	1	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	1	0	1	0

Tabelle 4: 7410 Baustein

Die beiden Wahrheitstabellen „ Schalten“ und „7410 Baustein“ liefern das gleiche Ergebnis bei „Relais“. Dadurch wurde gezeigt, dass der Aufbau der Logik-Schaltung (Abbildung 18) stimmt.

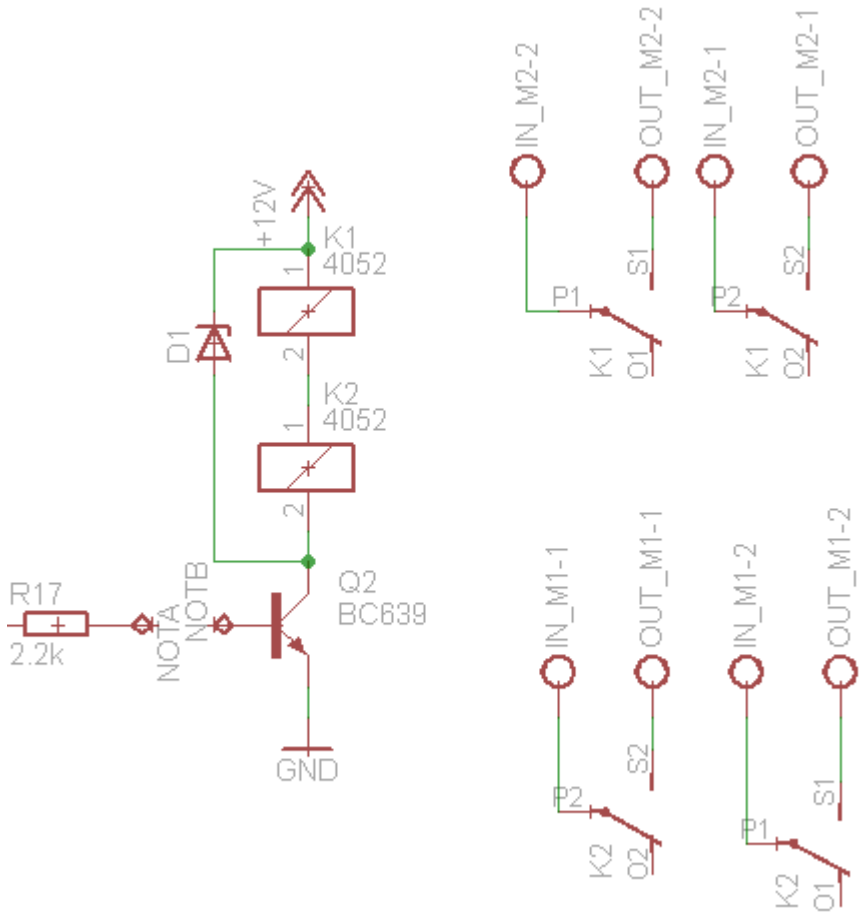


Abbildung 16: Relaisschaltung

Um die Relais schalten zu können, wird zwischen dem Relais und der Logikschaltung ein Transistor (Q_2) eingebaut, welcher als Schalter funktioniert. Kommt von der Logikschaltung ein High-Pegel (5V) so schaltet der Transistor durch. Der Widerstand R_{17} hat die Funktion, dass die Logikschaltung einen eindeutigen High-Pegel liefern kann. Lässt man diesen Widerstand weg, ist der High-Pegel 0,7V durch die Basis-Emitter-Spannung. Die Relais sind zwischen der Spannungsversorgung 12V und dem Kollektor des Transistors eingebaut. Schaltet der Transistor durch, kann ein Strom fließen und die Relais schalten. Im nichtgeschalteten Zustand des Relais, ist der Motorstromkreis unterbrochen. Die Diode D_1 funktioniert als Schutzdiode, beim Abschalten des Relais. Zwischen Logikschaltung und Transistor kann ein Notaus-Schalter (Öffner) eingebaut werden. Wird dieser nicht verwendet, muss eine Brücke eingebaut werden.

Abbildung 17: Notaus

3.7 Spannungsstabilisierung

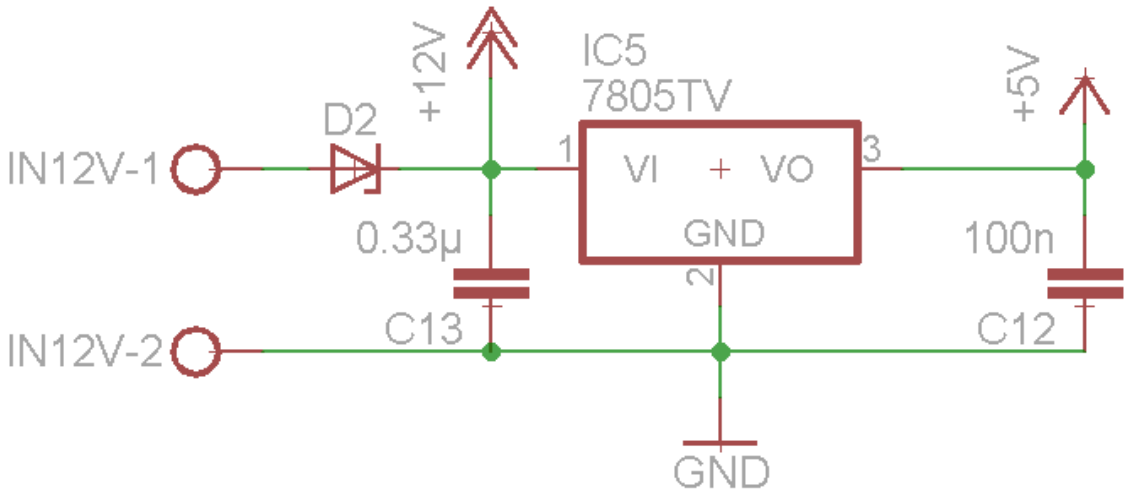


Abbildung 18: Spannungsversorgung

Die Eingangsspannung des Boards beträgt 12V. Um sie auf 5V zu reduzieren, wird der integrierte Festspannungsregler 7805 eingesetzt. Als Kondensatoren werden die im Datenblatt vorgeschlagenen Werte verwendet. Am Eingang (Pin1) des Reglers wird ein 0,33 µF Kondensator gegen Masse geschaltet, am Ausgang (Pin3) ein 100nF Kondensator. [6] Als Schutz vor Verpolung der Eingangsspannung dient die Diode D_2.

3.8 Schnittstellen

Es werden zwei Schnittstellen auf dem Board implementiert. Zum Programmieren des Mikrocontrollers wird eine ISP (In-System-Programmierung) verwendet. Eine weitere Schnittstelle, die auf dem Board zu finden ist, ist die RS232-Schnittstelle. Diese kann zur Kommunikation zwischen Controller und Steuerprogramm verwendet werden. Beide Schnittstellen entsprechen der RN-Definition[4].

3.8.1 ISP

Über die ISP-Schnittstelle kann der µC von einem externen System programmiert werden. Mit einem Standard ISP-Kabel kann der Controller direkt an den Parallelport des PCs angeschlossen werden. Als ISP-Schnittstelle wird eine 10-pinige Wannenbuchse verwendet. Die Pinbelegung der Schnittstelle wurde vom RN-Controll-Board übernommen. [7]

[4] Die RN-Definitionen sind Steckerbelegungen für den Bereich Robotik-und Mikrocontroller. Diese wurden im Internetportal „RoboterNetz.de“ erarbeitet und empfohlen um Schaltungen kompatibler zueinander zu gestalten. [8]

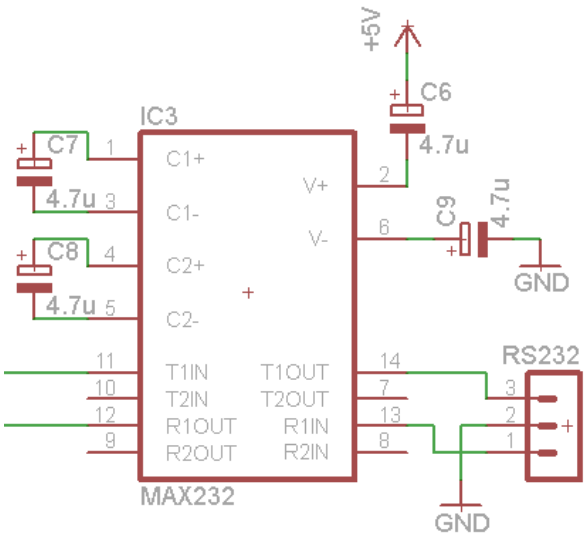
Pin	Belegung	Pin	Belegung
1	MOSI	6	GND
2	+5V	7	SCK
3	frei	8	GND
4	GND	9	MISO
5	RESET	10	GND

Tabelle 5: Pinbelegung ISP

3.8.2 RS232

Die serielle Schnittstelle RS232 ist dafür gedacht, dass eine Kommunikation zwischen dem Safety-Board und dem Hauptprogramm stattfinden kann. Des Weiteren kann durch einfache PRINT Anweisungen eine Ausgabe über ein Terminalprogramm angezeigt werden.

Zur Pegelwandlung wird ein MAX232 verwendet. Der Pin11 des MAX232 ist mit dem TxD des µC verbunden. Der RxD des Controllers geht auf Pin12 vom MAX232. Die externe Beschaltung des MAX232 wird mit 4 Elko's realisiert (Abbildung 19). Jeder hat einen Wert von 4,7µF.



Pin	Belegung
1	RX
2	GND
3	TX

Tabelle 6: Pinbelegung RS232

Abbildung 19: RS2322-Schnittstelle

3.8.3 Wannenbuchsen für die Motor-PWM

Um die Motor-PWM abzugreifen werden zwei Wannenbuchsen mit je 26 Pins verwendet. Pinbelegung ist dieselbe wie bei der Motor-Power-Schnittstelle des Pioneer 3. Siehe dazu Handbuch „Pioneer3 Operations Manual"[2]. Die Pins der beiden Wannenbuchsen werden direkt miteinander verbunden, weil auf dem Safety-Board nur die Motor-PWM benötigt wird. Zum Beispiel: Pin2 mit Pin2, Pin4 mit Pin4, usw.

Ausnahme bilden Pin1 und Pin 3. Diese beiden Pins liefern die Motor-PWM. Mit Stecker 1 wird diese abgegriffen. Nach Verarbeitung der Motor-PWM vom Safety-Board wird über Stecker 2 dies an die Motoransteuerung übergeben.

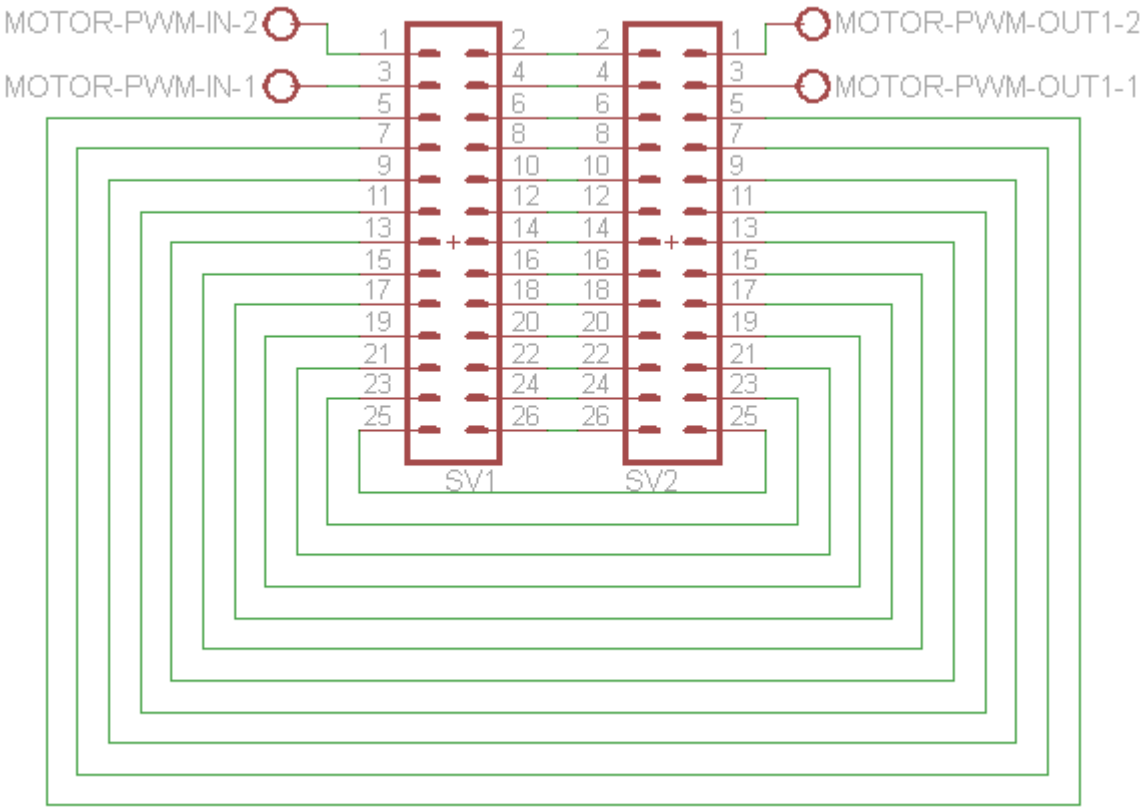


Abbildung 20: Wannenstecker

3.9 Geschwindigkeitsbegrenzung

Das Steuerprogramm gibt die Geschwindigkeit des Roboters durch ein PWM-Signal vor. Dieses wird auf das Safety-Board umgeleitet, damit die Geschwindigkeit begrenzt werden kann. Der Controller des Safety-Boards gibt eine maximale Motor-PWM vor, welche mit der Motor-PWM des Steuerprogramms durch ein UND-Gatter verknüpft wird. Ist die High-Phase des Safety-Boards länger als die des originalen PWM-Signals, wird diese unverändert weitergeben. Durch die UND-Logik liefert immer das „kürzere“ Signal das Ende der High-Phase. Ist nun die maximale PWM kürzer, wird durch diese die PWM begrenzt. So kann eine Geschwindigkeitsbegrenzung auf z.B. maximal 50% vorgenommen werden. Liegt eine dauerhafte 1 (5V) als maximale PWM, so findet keine Geschwindigkeitsbegrenzung vom Safety-Board statt. Das größte Problem stellt die Synchronisation der beiden PWMs dar.

Das Motor-PWM-Signal wird zusätzlich vom Mikrocontroller abgegriffen. Dieses Signal liegt am Int 1 (rechte Motor-PWM) bzw. Int0 (linke Motor-PWM) Eingang des Controllers. Über diesen Eingang kann ein externer Interrupt bei steigender oder fallender Flanke ausgelöst werden.

Durch das Auslösen des Interrupts beginnt der µC mit dem Senden der maximalen PWM. Auf diese Art und Weise wird das Synchronisieren der PWM-Signale realisiert.

Die Geschwindigkeit des linken und rechten Motors muss begrenzt werden. Beide Motoren werden vom Steuerprogramm separat voneinander angesteuert. Somit sind zwei UND-Gatter nötig, für jeden Motor eines.

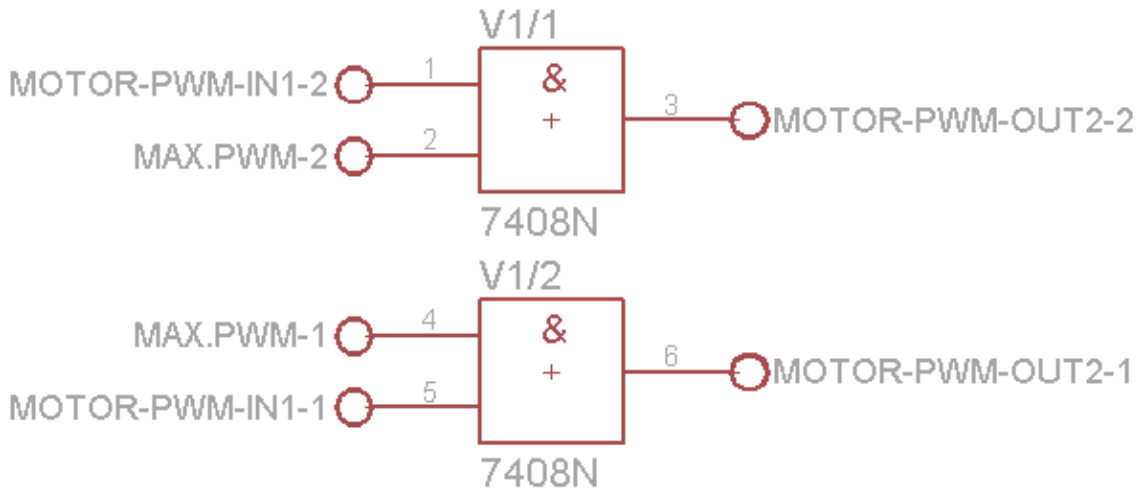


Abbildung 21: Geschwindigkeitsbegrenzung

3.10 Mikrocontroller: Atmega32

Der auf dem Board eingesetzte µC ist ein Atmega32 der Firma „Atmel“. Er gehört zur 8-bit Mikrocontrollerfamilie AVR. Die Hauptaufgabe des Controllers ist es, zu überwachen ob ein Heartbeat-Signal korrekt anliegt. Eine weitere wichtige Aufgabe ist die Vorgabe der maximalen Motor-PWM. Falls kein Atmega32 vorhanden ist, kann auch der Pin kompatible Atmega16 eingesetzt werden. Dieser verfügt lediglich über weniger Speicher.

Einige wichtige Eigenschaften des Atmega32 [9]:

- 32 programmierbare I/O Ports
- vier PWM Kanäle
- zwei 8-bit Timer/Counter
- ein 16-bit Timer/Counter
- 3 externe Interrupt Eingänge
- externer Quarz max.16 MHz
- 32 K Bytes Flash Speicher
- 1 K Bytes EEProm
- 2 K Bytes interner SRAM Speicher
- Master/Slave SPI Serial Interface

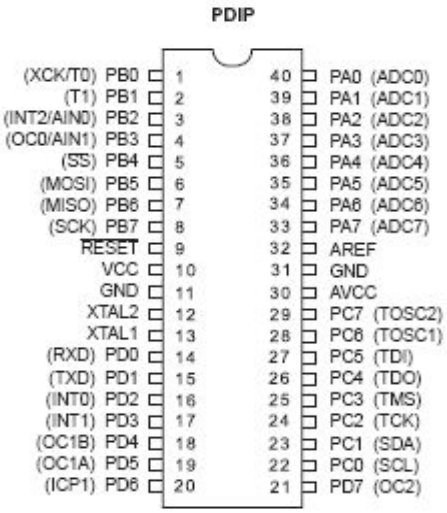


Abbildung 22: Pinbelegung Atmega32

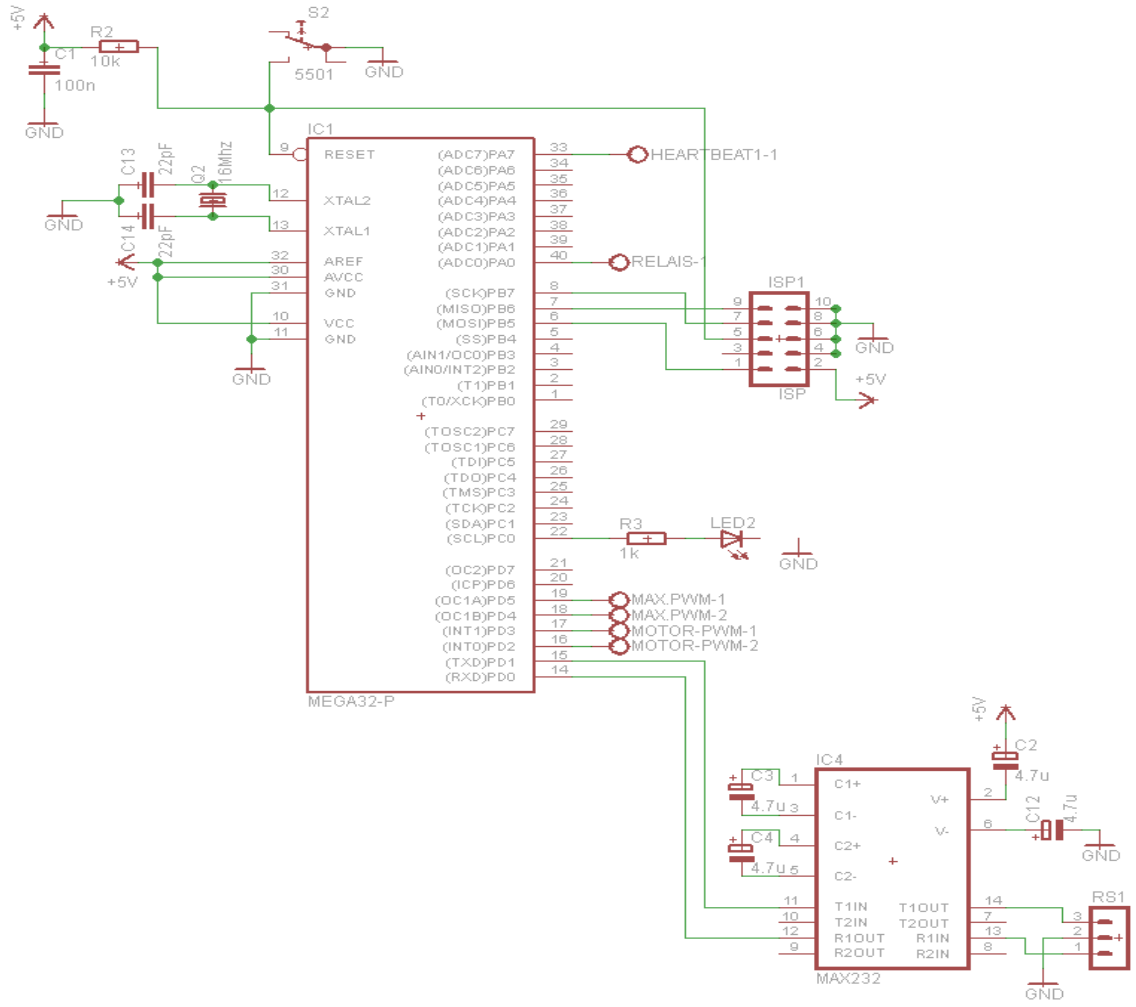


Abbildung 23: µC-Schaltung

Der Mikrocontroller ist mit einem externen Quarz Q_2 ausgestattet. Q_2 hat einen Wert von 16 MHz. Dieser wird für die Übertragung mit der RS232-Schnittstelle benötigt. Die Versorgungsspannung des Controllers beträgt 5V. Über den Taster S_2 kann der Controller resetet werden. An dem Port PC0 ist eine LED angeschlossen, welche für Testzwecke verwendet werden kann. Beispielweise für die Signalisierung des Betriebszustandes des Boards. Die LED ist so verschaltet, dass sie bei einem High-Pegel leuchtet. Der Widerstand R_3 dient als Vorwiderstand der LED. Das Heartbeat-Signal liegt an Pin33 (PA7). Mit dem Ausgang PA0 wird das Relais geschaltet. Die beiden Motor-PWMs liegen am INT0 bzw. INT1-Eingang. Über den PWM-Ausgang OC1A bzw. OC1B kann die max. PWM vorgegeben werden. Die Pins SCK, MISO und MOSI sind mit der ISP-Schnittstelle verbunden. Die Pins TxD und RxD sind über „MAX232“ mit der RS232-Schnittstelle verbunden.

Pin	Bezeichnung	Funktion
9	RESET	Reseten
12,13	XTAL	Ext. Quarz 16 MHz
32,30,10	AREF,AVCC,VCC	+5V
31,10	GND	GND
14	RxD	RS232-Schnittstelle
15	TxD	RS232-Schnittstelle
8	SCK	ISP-Schnittstelle
7	MISO	ISP-Schnittstelle
6	MOSI	ISP-Schnittstelle
33	PA7	Heartbeat In
40	PA0	Relais
22	PC0	LED
16	INT0	Motor-PWM2
17	INT1	Motor-PWM1
18	OC1B	max. PWM2
19	OC1A	max.PWM1
restliche		keine Belegung

Tabelle 7: Pinbelegung Mikrocontroller

3.11 Board Version V1.0

Auf dem Board V1.0 findet nur eine Heartbeat-Abfrage über die Analogschaltung und den µC statt. Der Analogschaltung ist die Logikschaltung nachgeschaltet, welche mit der Relaisschaltung verbunden ist. Des Weitereren, sind auf dem Board die Spannungsstabilisierung (ohne Schutzdiode D_2) und die Mikrocontroller-Schaltung untergebracht. Die µC-Schaltung ist hier ohne die PWM-Regulierung ausgeführt. Diese kommt erst im Board V1.1 hinzu. Die beiden Schnittstellen ISP und RS232 befinden sich ebenfalls auf dem Board. Es befinden sich zwei Relais auf dem Board, wobei Relais1 zur Unterbrechung des linken und Relais2 zur Unterbrechung des rechten Motors dient.

Die Platine wurde in mit dem Programm „EAGLE“ gelayoutet. Die Platinengröße beträgt halbes Euroformat (100x80 mm). (Platinenlayout: siehe Anhang B)

Die Spannungsversorgung der Platine beträgt + 12V. Dies ist die vom Roboter bereitgestellte Batteriespannung.

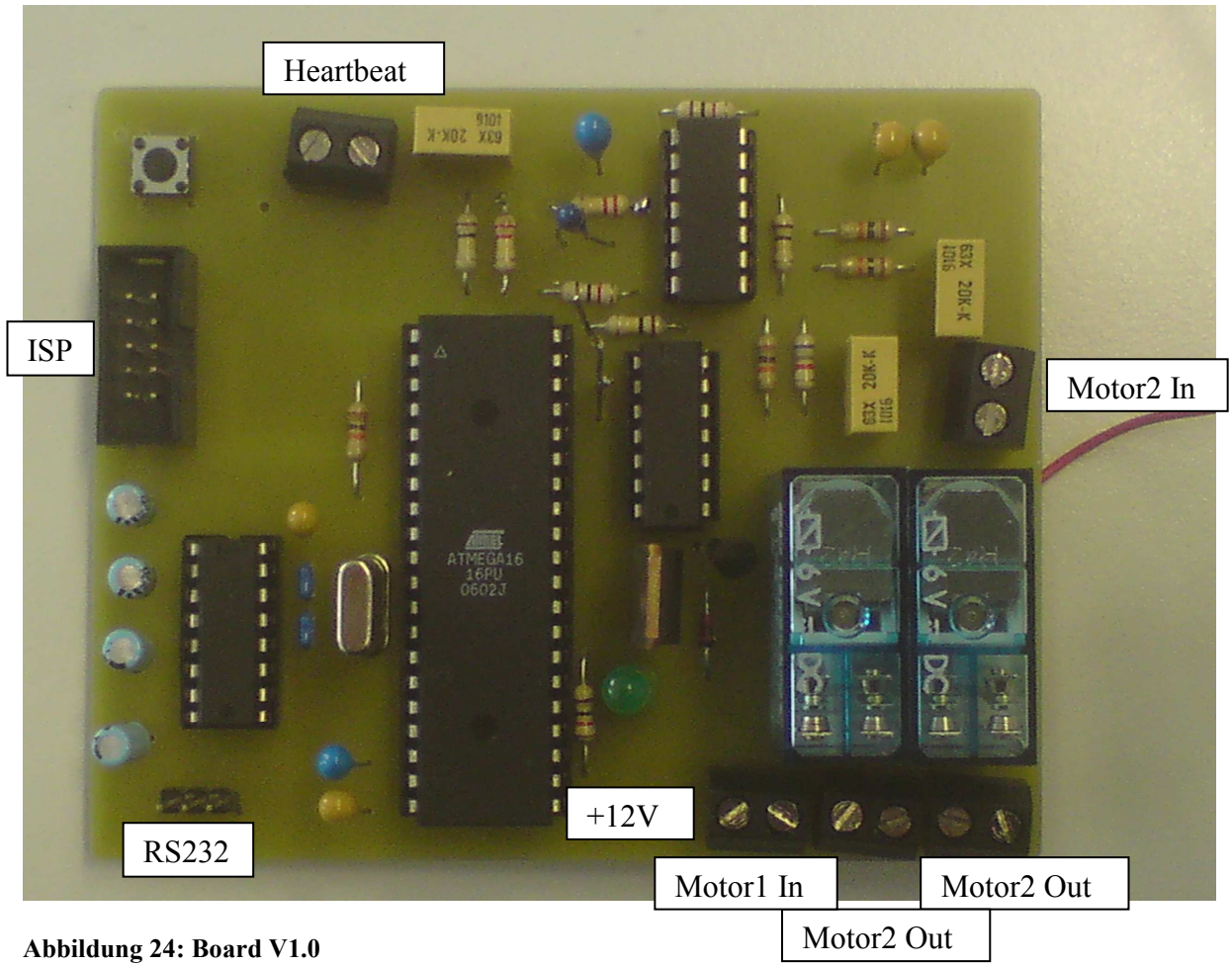


Abbildung 24: Board V1.0

3.12 Board Version V1.1

Die Version 1.0 wurde um die Geschwindigkeitsbegrenzung erweitert. Dafür sind als Schnittstelle zwei Wannenbuchsen angebracht, über welche die Motor-Power abgegriffen werden kann. Vom Mikrocontroller werden die PWM Ausgänge zur Vorgabe der maximalen PWM verwendet. Die Diode D_2 dient als Schutzdiode vor Verpolung der Eingangsspannung. Es besteht die Möglichkeit einen zusätzlichen Notaus-Schalter anzubringen. Wird dieser nicht verwendet, muss anstelle des Notaus-Schalters eine Brücke eingebaut werden. Als Versorgungsspannung benötigt das Board +12V. Schaltplan und Platinenlayout sind im Anhang C zu finden.

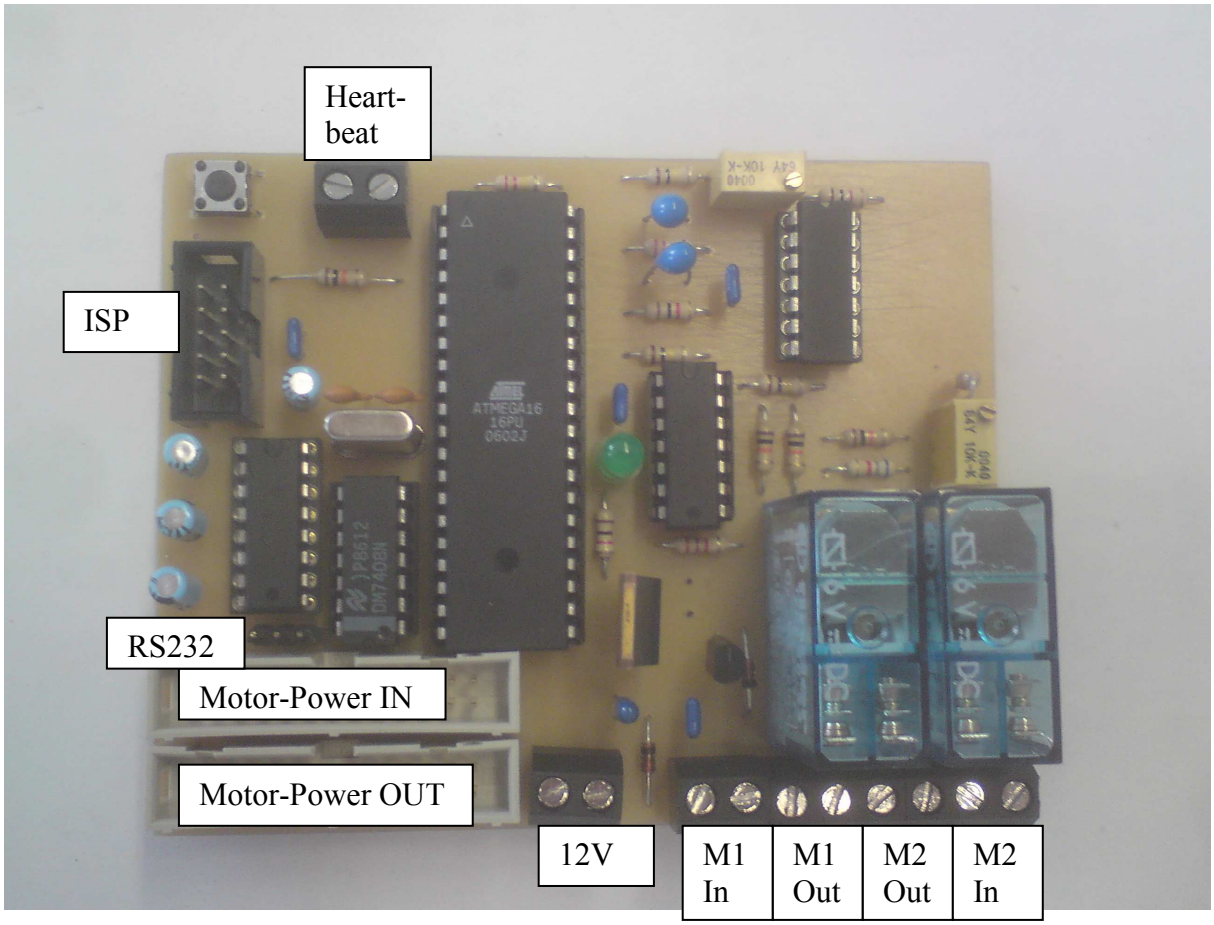


Abbildung 25: Board V1.1

4 Mikrocontroller-Testprogramm

Für den Mikrocontroller wurde ein kleines Testprogramm entwickelt, welches zum Testen der Grundfunktionen des µC bzw. des Boards dient. Die nicht für den jeweiligen Test benötigten Programmteile sollten vor dem Programmieren des µC auskommentiert werden. Dies ist notwendig, da manche Bauteile (z.B. LED) für mehrere Tests (Heartbeat und Mikrocontroller) verwendet werden. Es ist zu beachten, welcher Test durchgeführt wird.

Die Anforderungen an sicherheitsrelevante Software sind sehr hoch und würden den Rahmen dieser Arbeit sprengen. Daher wird das Mikrocontrollerprogramm in einem weiteren Projekt realisiert.

4.1 Heartbeattest

Das Heartbeatsignal liegt an PA7 an. Liegt ein Heartbeatsignal an, soll die LED mit der Frequenz des Signals blinken. Liegt eine Logische „1“ an, leuchtet die LED, bei einer „0“ ist sie aus.

Programmcode:

```
DDRA = 0x00; // Port A als Eingang festlegen
DDRC = 0xff; //Port C als Ausgang festlegen
if(PINA & (1<<PINA7))
          {
               PORTC=0x01;
          }else
          {
                PORTC=0x00;
          }
```

Über das Datenrichtungsregister DDRx wird die Funktion des Ports definiert. Ist das Bit im Register gesetzt (1), ist der jeweilige Pin als Ausgang geschaltet. Für die Beschaltung als Eingang wird das Bit gelöscht (0). Im Testprogramm ist Port A als Eingang definiert und Port C als Ausgang. Über die Eingangsadresse PINx kann der Zustand des Pins abgefragt werden. Zum Ansteuern der Ausgänge wird das Datenregister PORTx verwendet. Wobei x jeweils durch den verwendeten Port A, B, C oder D ersetzt wird.

Es findet eine if-Abfrage statt. Ist der PortA7 gesetzt (Heartbeat=1), wird der Ausgang PortC ebenfalls gesetzt (LED leuchtet). Im anderen Fall, wird der Ausgang zurückgesetzt.

4.2 Relais Ausgang

Um den Relais-Ausgang zu testen, wird er jede Sekunde umgeschaltet. Um die Analogschaltung zu testen, wird dem Relais-Ausgang eine dauerhafte „1“ zugewiesen.

Programmcode:

```
DDRA = (1<<PA0); // PortA0 als Ausgang fest-
legen
// Ausgang PA0 (Relais-Ausgang) jede Sekunde
Toggeln
while(1){
     PORTA &= ~(1<<PA0); //Pin löschen (0)
     _delay_ms(1000);
     PORTA |= (1<<PA0);  //Pin setzen (1)
     _delay_ms(1000);
     }
```

Das Programm läuft in einer Endlosschleife. Der Pin wird gelöscht und danach eine Sekunde abgewartet. Anschließend wird der Pin gesetzt und wieder eine Sekunde gewartet. Nun beginnt die Schleife wieder von neuem.

4.3 Mikrocontroller-Test

Die LED wird jede Sekunde getoggelt, sobald der Mikrocontroller aktiviert ist. Das Funktionsprinzip ist selbiges, wie in Kapitel 4.2 bei Relais Ausgang beschrieben. Es wird nur PORTA durch PORTC ersetzt. Eine weitere Änderung in der Programmierung ist der Programmierstil. Anstelle des Schiebens eines einzelnen Bits (PORTA &= ~(1<<PA0)), wird das Bit durch Zuweisung eines Hexwertes programmiert (z.B. PORTC=0x01).

Programmcode:

```
DDRC = 0xff; //Port C als Ausgang fest-
legen
while(1){
     PORTC=0x01;         //LED anschal-
ten
     _delay_ms(1000);
     PORTC=0x00;         //LED aus-
schalten
     _delay_ms(1000);
     }
```

4.4 Ausgabe über RS232-Schnittstelle

Über die RS232-Schnittstelle soll auf dem Terminal „Hallo ☺“ ausgegeben werden. Dazu wird das UART (Universal Asynchronous Receiver und Transmitter) verwendet. Um eine ganze Zeichenkette zu senden, muss diese zerlegt werden. Es ist lediglich möglich, einzelne Zeichen zu übertragen.

Programmcode:

```
//Funktion zum Senden einzeler Char-Zeichen über RS232
void sendChar(unsigned char c){
     while(!(UCSRA & (1<<UDRE))) {} //Warten, bis Senden
möglich ist
     UDR=c; // schreiben von c
     return;
}

// Funktion: String als einzel Zeichen an sendChar zu über-
geben
void sendUSART(char *s){
     while(*s){
          sendChar(*s);
          s++;
     }
}
int main(void){

     // Ausgabe über RS232-Schnittstelle
     UCSRB |= (1<<TXEN); //UART TX (Senden) einschalten
     UCSRC |= (1<<URSEL)|(3<<UCSZ0);   //Modus Asynchron
                                       8N1 (8 Datenbits, No Par-
                                       ity, 1 Stopbit)
     UBRRH = 0;     //Highbyte ist 0
     UBRRL = 103;   //Lowbyte ist 103 Baudrate=9600
     sendUSART("Hallo :-) \r\n"); //Senden von "Hallo :-)"
     Return 0;
}
```

Zur Datenübertragung wird der Modus „Asynchron 8N1“ verwendet. Das heißt 8 Datenbits, kein Paritybit und ein Stoppbit. Die Baudrate soll 9600 betragen, diese wird über das UBRR eingestellt. Berechnung des UBRR mit f_{osc}=16MHz und Baud=9600:

$$UBRR = \frac{fosc}{16BAUD} - 1 = \frac{16MHz}{16 * 9600} - 1 = 103$$

Im Hauptprogramm (main) werden der Modus und die Baudrate festgelegt. Des Weiteren werden die zu sendenden Zeichen an die Funktion „sendUSART“ übergeben. Diese Funktion übergibt die Zeichenkette, einzeln an die Funktion „sendCHAR“. Diese wartet ab, bis der Sendepuffer frei ist. Danach wird das Zeichen in den Puffer gelegt und gesendet.

4.5 maximale PWM

Über den PWM-Ausgang (PD4 und PD5) wird die maximale Motor-PWM ausgegeben. Wird dieser Ausgang auf „Eins“ gesetzt, findet keine Geschwindigkeitsbegrenzung statt. Das größte Problem stellt die Synchronisation der Signale dar. Das Original-Signal löst bei steigender Flanke einen Interrupt aus. In diesem Interrupt werden die beiden Ausgänge PD4 und PD5 für 25µs gesetzt und anschließend wieder gelöscht. Dadurch hat die maximale PWM die gleiche Frequenz wie die originale Motor-PWM. Die Pulsbreite der maximalen PWM beträgt 25µs. Durch Ändern des Wertes in der Warteschleife (_delay_us(25)) kann eine andere Pulsbreite eingestellt werden. Dieser Wert darf nicht größer als 49 sein, weil die Moter-PWM eine Periodendauer von T=50µs hat.

Programmcode:

```
// Interrupt-Service-Routine die bei einer steigenden Flanke
an Pin INT0 aufgerufen wird
ISR(INT0_vect){

// PWM Eingänge setzen und dann 25µs warten, anschließen wi-
der zurücksetzen
     PORTD |= (1<<PD4);
     PORTD |= (1<<PD5);
     _delay_us(25);
     PORTD &= ~(1<<PD4);
     PORTD &= ~(1<<PD5);

}
int main (void){
     //Als Ausgänge definieren um die max.PWM vorgeben
     DDRD |= (1<<PD4) | (1<< PD5);
     // Beide Ausgänge auf Null setzen.
     PORTD &= ~(1<<PD4);
     PORTD &= ~(1<<PD5);
}
```

5 Board testen

Das Board wird in den Pioneer3 eingebaut. Zur Simulation des Heartbeat-Signals wird das RN-Controll-Board verwendet. Anstelle des Notaus-Schalters wird eine Brücke eingebaut.

5.1 Analogschaltung

Zum Testen der Analogschaltung wird der Relais-Ausgang des Mikrocontrollers auf „1“ gesetzt. Das heißt der Controller gibt den Antrieb ständig frei. Nur wenn jetzt die Analogschaltung ebenfalls das „OK“ gibt, wird der Antrieb freigegeben. Das RN-Controll-Board liefert je nach Testfall die unterschiedlichen Heartbeats.

Das Oszilloskop wird an das Heartbeatsignal (CH1) angeschlossen. Mit dem zweiten Kanal (CH2) wird der Ausgang der Logikschaltung dargestellt. Zur Erinnerung, liefert dieser eine Logische „1“ schaltet der Transistor. Hat der Transistor geschaltet, wird der Antrieb durch die Relais freigegeben.

Die Schaltschwelle des Komparators1 beträgt U_{min} = 2,25V. Die maximale Spannung beträgt U_{max} =3,3V vorgegeben durch Komparator 2.

5.1.1 Heartbeat in Ordnung

Beschreibung:
Als erstes liefert das RN-Controll-Board dauerhaft ein korrektes Heatbeatsignal. Das Signal hat eine Frequenz von f_H=100 Hz und eine Pulsbreite von 50%.

Ergebnis:
Das Safetyboard gibt den Antrieb frei.

Messergebnisse:

Funktion	U/[V]	Logisch
Tiefpass out	2,48	
Komparator1 out	0,63	0
Komparator2 out	3,68	1
Logikschaltung out	3,52	1

Tabelle 8: Messwerte Heartbeat korrekt

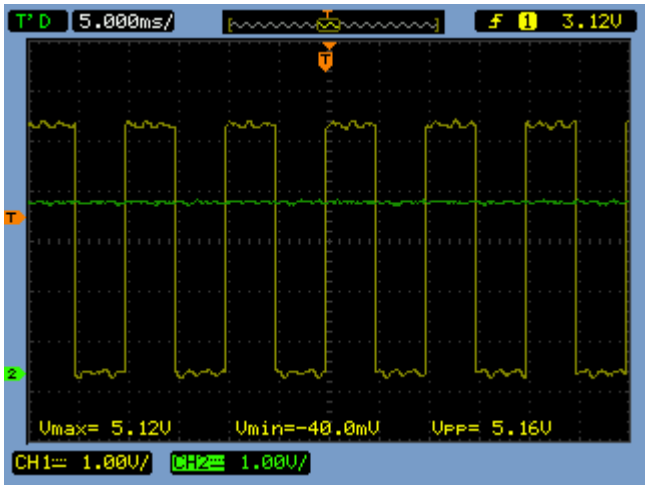


Abbildung 26: Testfall korrektes Heartbeat

5.1.2 Heartbeat dauerhaft auf 5V

Beschreibung:
Das Heartbeatsignal ist fehlerhaft. Es bleibt konstant auf 5V.

Ergebnis:
Der Antrieb ist nicht freigegeben. Die Relais schalten nicht. Sowohl Komparator1, als auch Komparator2 liefern ein Logisch „1“. Der Operationsverstärker geht in die Spannungssättigung.

Funktion	U/[V]	Logisch
Tiefpass out	3,63	
Komparator1 out	3,65	1
Komparator2 out	3,65	1
Logikschaltung out	0,18	0

Tabelle 9: Heartbeat dauerhaft auf 5V

5.1.3 Es liegt kein Heartbeat an

Beschreibung:
Das Heartbeatsignal beträgt 0V.

Ergebnis:
Sobald kein Heartbeatsignal anliegt, wird der Antrieb abgeschaltet. Das Steuerprogramm hat keinen Zugriff mehr auf den Antrieb.

Funktion	U/[V]	Logisch
Tiefpass out	0,01	
Komparator1 out	0,62	0
Komparator2 out	0,62	0
Logikschaltung out	0,18	0

Tabelle 10: kein Heartbeat Signal

5.1.4 Heartbeat fehlerhaft

Beschreibung:
Das Heartbeatsignal hat eine Frequenz f_H=100Hz. Die Pulsbreite beträgt a) 20% und b) 70%.

Ergebnis:
In beiden Fällen wird, der Antrieb nicht freigegebenen. Der Ausgangswert des Tiefpassfilters liegt nicht im vorgegebenen Fenster (U_{min}=2,25V und U_{max}=3,3V).

Fall A:

Funktion	U/[V]	Logisch
Tiefpass out	1,03	
Komparator1 out	0,62	0
Komparator2 out	0,62	0
Logikschaltung out	0,18	0

Tabelle 11: Heartbeat 20%

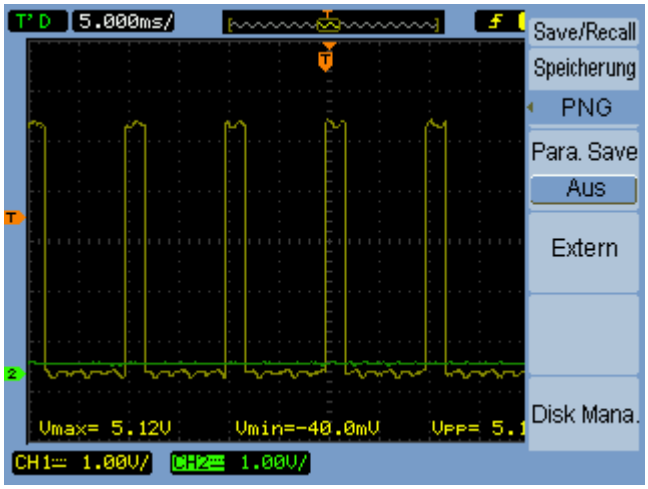


Abbildung 27: Heartbeat 20%

Fall B:

Funktion	U/[V]	Logisch
Tiefpass out	3,56	
Komparator1 out	3,64	1
Komparator2 out	3,62	1
Logikschaltung out	0,18	0

Tabelle 12: Heartbeat 70%

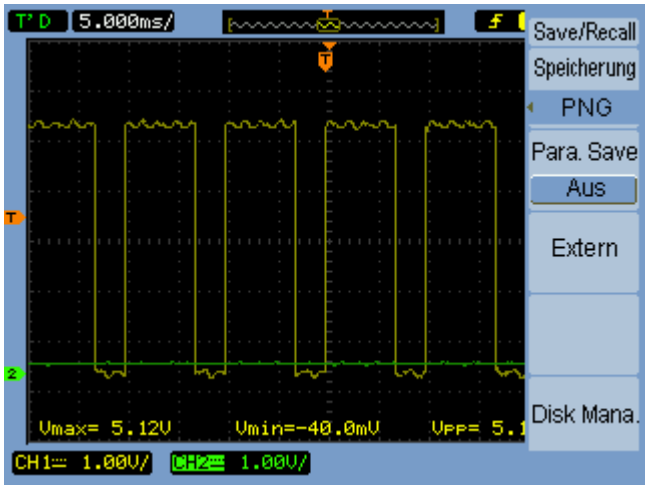


Abbildung 28: Heartbeat 70%

5.1.5 Ausfall des Heartbeat

Beschreibung:
Das Heartbeatsignal fällt nach 200 Impulsen für eine Sekunde aus. Es wird die Aus- und Einschaltzeit nach Erkennen des Heartbeats bzw. Ausfall des Signals gemessen.

Ergebnis:
32,8 ms nach Ausfall des Heartbeat-Signals schaltet das Relais den Antrieb ab.

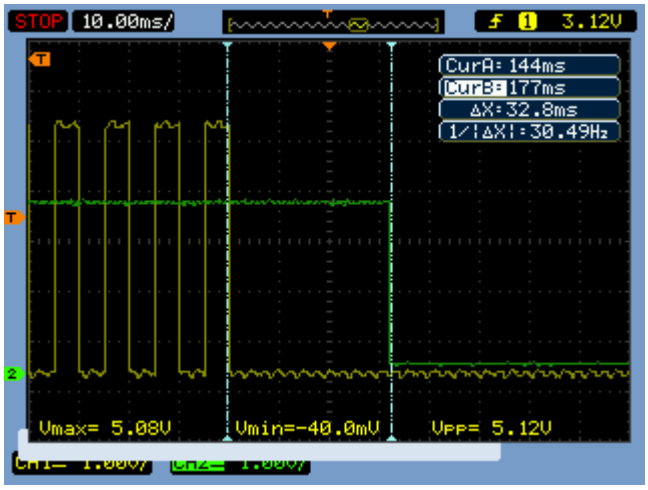


Abbildung 29: Ausschaltzeit

Kommt wieder ein Heartbeat-Signal an, dauert es t_{an}=186ms bis das Relais schaltet.

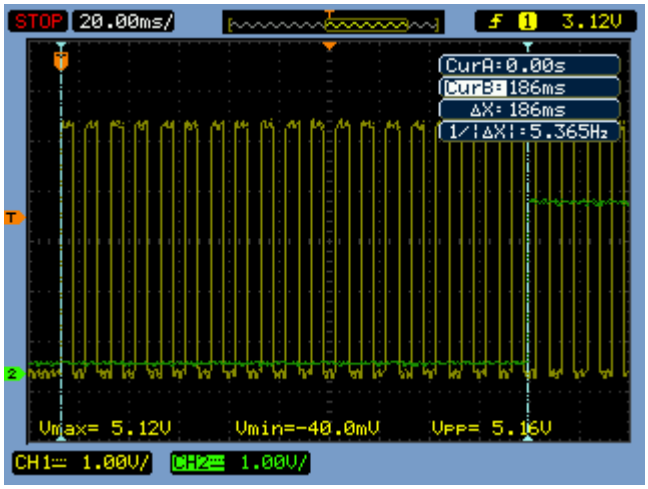


Abbildung 30: Einschaltzeit

Bemerkung:
Diese Zeiten sind von der Schaltschwelle U_{min} und der Zeitkonstante τ des Tiefpasses abhängig. Je näher die Schaltschwelle U_{min} an dem theoretischen Mittelwert U_{mit}=2,5V liegt, desto schneller schaltet das Relais bei einem Ausfall ab. Je kleiner die Abschaltzeit, desto größer die Einschaltzeit. Durch das Erhöhen der Schaltschwelle, braucht der Tiefpass länger um diesen Wert zu erreichen.

Beschreibung:
Vom Heartbeat-Signal fallen nach 200 Impulsen 3 Stück aus.

Ergebnis:
Des Relais bleibt dauerhaft geschaltet, auch bei Ausfall der drei Impulse.

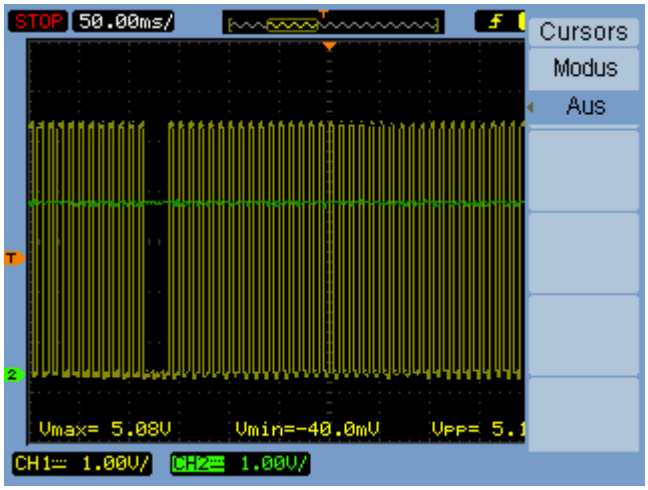


Abbildung 31: Ausfall von 3 Impulsen

Bemerkung:
Die Analogschaltung soll erst reagieren, wenn mehr als 3 Impulse ausfallen. So bleibt dem Mikrocontroller Zeit einzugreifen, ohne dass der Antrieb gleich stehen bleibt. Möchte man dem Controller länger Zeit geben als die 3 Impulse, so muss die Zeitkonstante τ vergrößert werden. Die kann durch Erhöhen des Widerstands R_1 geschehen. Dabei ist zu beachten, dass dadurch die Aus- bzw. Einschaltzeit größer wird. Bei einem Widerstandswert von R_1=1 MΩ anstatt R_1=220kΩ ändert sich die Abschaltzeit von t_{ab}=32,8ms auf t_{ab}=98,4ms. (siehe Abbildung 32).

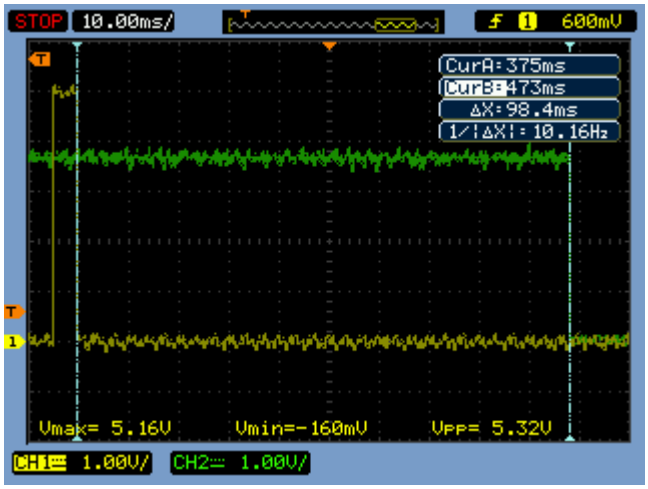


Abbildung 32: Ausfallzeit bei R_1=1 MΩ

5.2 Mikrocontrollertest

Zum Testen der Grundfunktionen des Controllers, wird das Testprogramm auf den Controller geladen.

5.2.1 Einlesen des Heartbeat

Beschreibung:
Die LED soll mit der Frequenz des Heartbeats blinken. Dazu wird vom Testprogramm der „Heartbeattest“ verwendet. Der Heartbeat Eingang des µC liegt an CH1 des Oszilloskop. Mit CH2 wird die Spannung der LED gemessen. (siehe Abbildung 33)

Ergebnis:
Sobald ein Heartbeat-Signal anliegt, beginnt die LED zu blinken. Sie blinkt mit einer Frequenz f_{LED}=100Hz.

Wird die Frequenz des Heartbeat-Signals verkleinert, blinkt die LED langsamer.

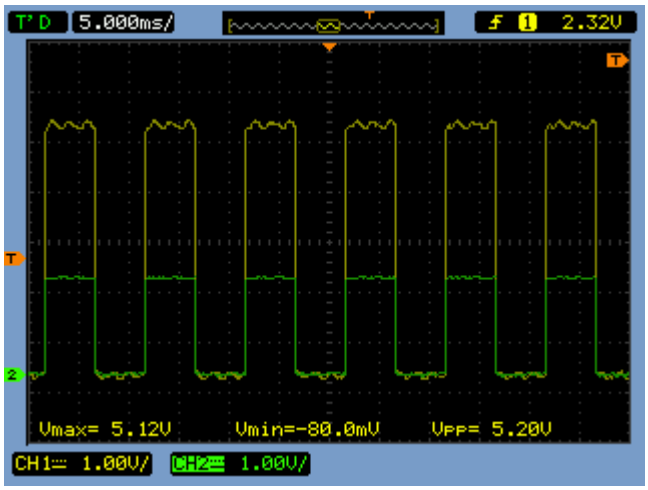


Abbildung 33: µC-Test Heartbeat einlesen

5.2.2 Relais Ausgang

Beschreibung:
Im Testprogramm wird der „Relais Ausgang Test“ verwendet. Der Ausgang wird jede Sekunde getoggelt. Zum Testen liegt ein korrektes Heartbeatsignal an. Dadurch kommt von der Analogschaltung eine „1“. Das Relais wird über den Mikrocontroller geschaltet.

Ergebnis:
Das Relais schaltet jede Sekunde. Schaltet der Relais-Ausgang des Controllers (Abbildung 34 CH1), schaltet das Relais ebenfalls (Abbildung 34 CH2).

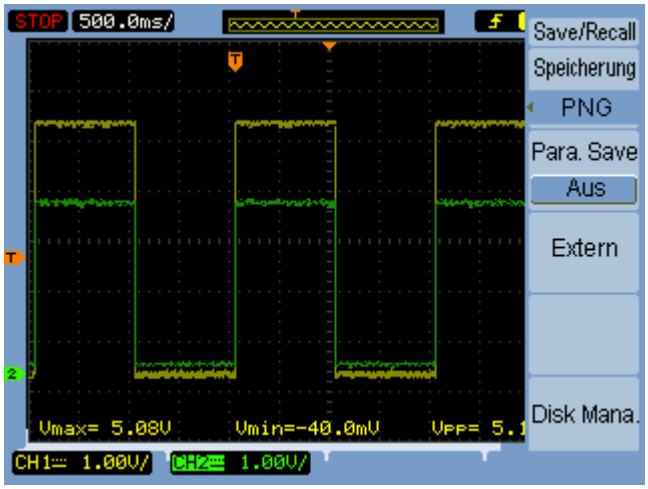


Abbildung 34: µC-Test Relais-Ausgang

5.3 Schnittstellen

5.3.1 ISP-Schnittstelle

Beschreibung:
Zum Testen der ISP-Schnittstelle wird vom PC das Testprogramm auf den Controller übertragen.

Ergebnis:
Nach dem Übertragen des Testprogramms, blinkt die LED. Damit wurde die Funktionsfähigkeit der ISP-Schnittstelle nachgewiesen. Es ist möglich den µC über diese Schnittstelle zu programmieren.

5.3.2 RS232-Schnittstelle

Beschreibung:
Das Safety-Board wird über die RS232-Schnittstelle an den COM1-Port des PC angeschlossen. Auf dem µC befindet sich das Testprogramm „Ausgabe über RS232-Schnittstelle“. Am PC wird über das HyperTerminal eine Verbindung zum Board hergestellt. Die einzustellenden Parameter sind:

- Connect using: COM1
- Bits per second (Baud): 9600
- Data bits: 8
- Parity: None
- Stop bits: 1
- Flow control: Hardware

Ergebnis:
Auf dem HyperTerminal wird „Hallo ☺“ ausgegeben. (Abbildung 35).

Damit ist nachgewiesen, dass eine Kommunikation über die RS232-Schnittstelle statt finden kann.

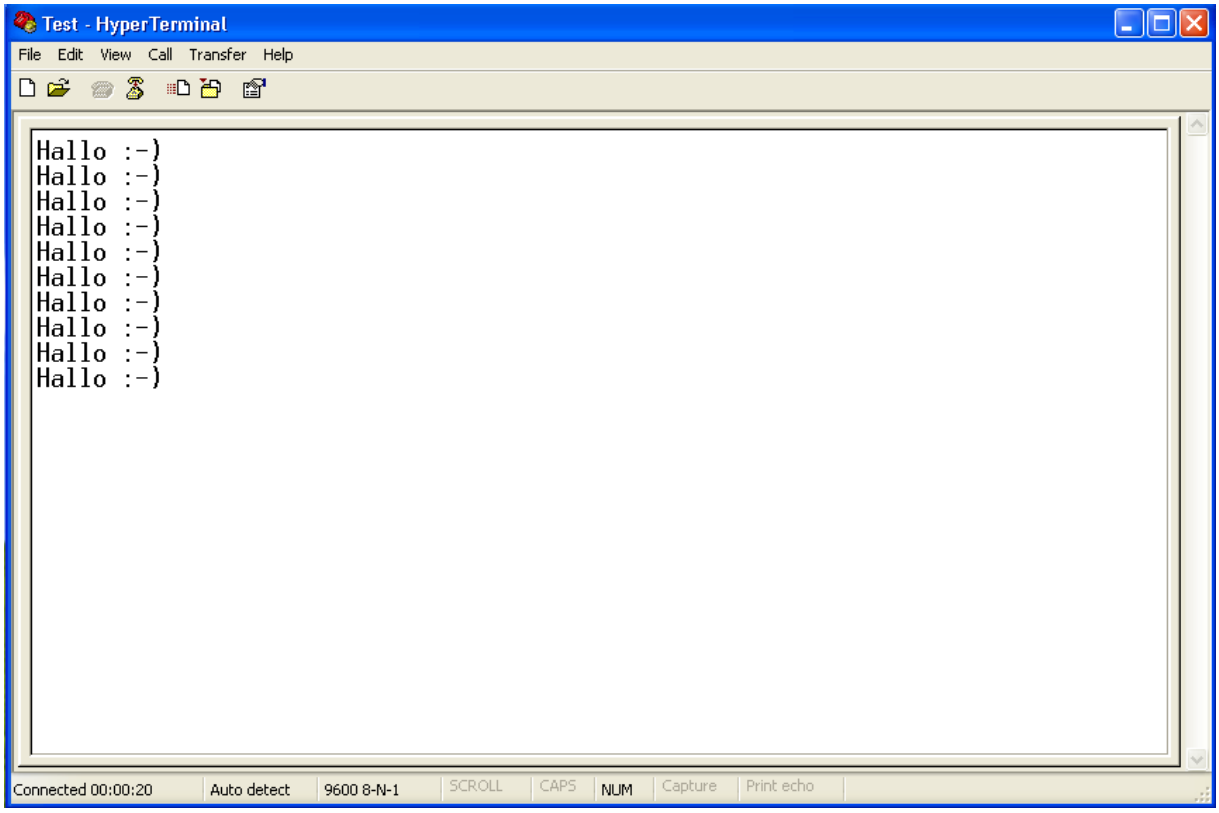


Abbildung 35: Ausgabe mit dem HyperTerminal

5.4 Geschwindigkeitsregulierung

5.4.1 maximale Motor-PWM ohne Geschwindigkeitsregulierung

Die Geschwindigkeit wird vom Safety-Board nicht begrenzt. Die beiden Ausgänge PD4 und PD5 sind auf „1“ geschaltet. Das heißt das PWM-Signal wird ohne Veränderung weitergegeben. Wird bei einem UND-Gatter einen Eingang auf „1“ gesetzt, so ist der Ausgang gleich dem zweiten Eingang. Der Roboter wird vom PC aus mit dem Demoprogramm „Aria“ gesteuert.

Ergebnis:
Die maximale Motor-PWM hat eine Periodendauer von T=50µs. Daraus ergibt sich eine Motor-PWM Frequenz von

$$f_{PWM} = \frac{1}{T} = \frac{1}{50\mu s} = 20kHz$$

Die maximale Pulsweite der Motor-PWM beträgt T_{Ein}=30µs. Als maximale Geschwindigkeit wird im Demoprogramm der Wert 749 angezeigt. Die Motorspannung beträgt U_M=7,35V, welche mit dem Multimeter gemessen wird.

Funktion	Abkürzung	Wert
Periodendauer	T	50µs
maximale Einzeit Motor-PWM	T_{Ein}	30µs
Spannung Motor	U_M	7,35V

Tabelle 13: Messwerte maximale Motor-PWM ohne Geschwindigkeitsregulierung

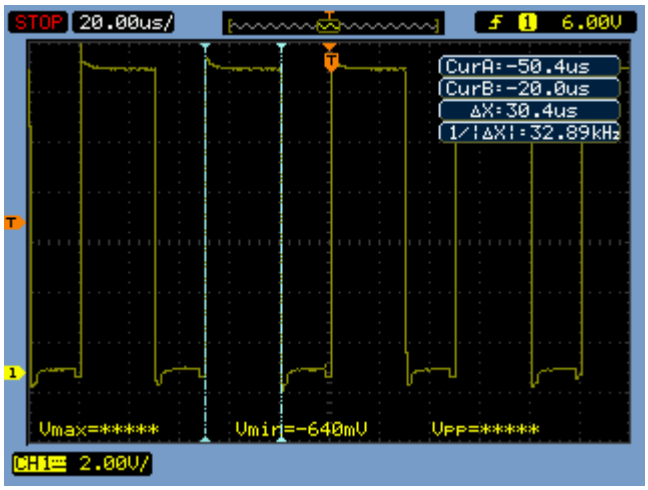


Abbildung 36: maximale Motor-PWM ohne Geschwindigkeitsregulierung

Die Motor-PWM in Abbildung 36 ist direkt am Motor gemessen.

5.4.2 maximale Motor-PWM mit Geschwindigkeitsregulierung

Zur Geschwindigkeitsregulierung wird eine maximale PWM vom Safety-Board vorgegeben. Mit dem Testprogramm „maximale PWM“ wird eine maximale Pulsbreite von 25µs vorgegeben. Die Frequenz ist die gleiche wie die der originalen Motor-PWM.

Ergebnis:
Die maximale Geschwindigkeit ist reduziert. Die Motor-PWM hat nur eine Pulsbreite T_{Ein}=22,4µs. Die Periodendauer der Frequenz ist f=50Hz. Die Motorspannung beträgt U_M=5,52V. Im Demoprogramm wird bei maximaler Geschwindigkeit der Wert 569 angezeigt.

Funktion	Abkürzung	Wert
Periodendauer	T	50µs
maximale Pulsweite Motor-PWM	T_{Ein}	22,4µs
Spannung Motor	U_M	5,52V

Tabelle 14: Messwerte maximale Motor PWM mit Geschwindigkeitsregulierung

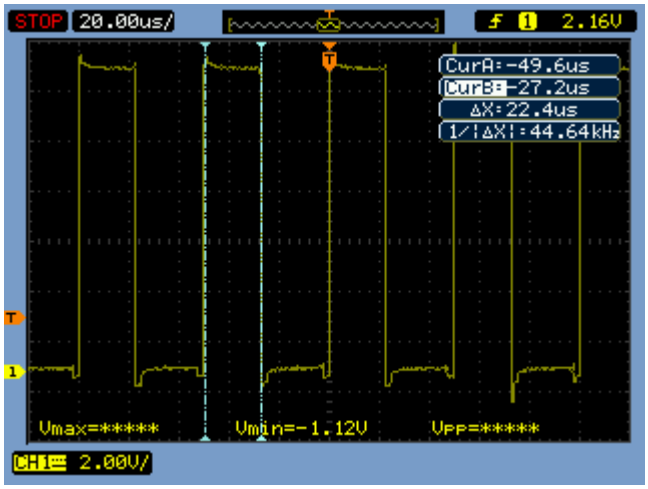


Abbildung 37: maximale Motor-PWM mit Geschwindigkeitsregulierung

Ist die Motor-PWM, die vom Steuerprogramm gesendet wird, größer als die maximale PWM vom Safety-Board, versucht die Steuerplatine die Geschwindigkeit nach zu regeln. Die Steuerplatine liefert eine dauerhafte „1“, dadurch ist die Ausgangs-PWM des Safety-Board die maximale PWM.

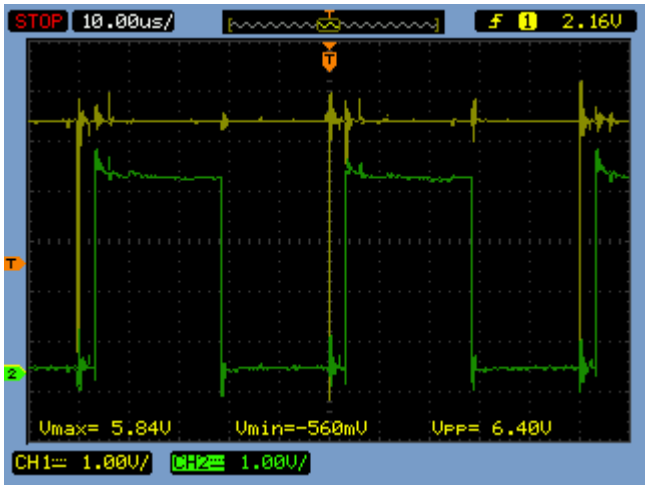


Abbildung 38: PWM Safety-Board

In Abbildung 38 ist an Kanal1 die Eingangs-PWM und mit Kanal2 die Ausgangs-PWM des Safety-Board dargestellt. Die Motor-PWM ist größer, als die maximale PWM vom Safety-Board. Es ist zu sehen, dass die Steuerplatine durch Senden einer „1“ versucht, die Geschwindigkeit zu vergrößern (CH1). Diese bleibt aber durch die maximale PWM begrenzt. Das Safety-Board sendet als Motor-PWM die maximale PWM weiter (CH2).

5.4.3 Motor-PWM kleiner maximale PWM

Die maximale PWM des Safety-Board hat eine Pulsbreite von T_{Ein}=24µs. Das Steuerprogramm sendet eine Motor-PWM mit einer Pulsbreite von T_{Ein}=19µs.

Ergebnis:
Das Safety-Board sendet das PWM-Signal weiter. Die Pulsbreite beträgt aber nicht mehr T_{Ein}=19µs sonderen ist ein wenig kleiner, nämlich T_{Ein}=17,8µs. Dies kommt daher, dass die Synchronisation der beiden PWM-Signale auf dem Board nicht hundertprozentig exakt ist. Es kann die Vorgabe der Geschwindigkeit weiterhin vom Steuerprogramm vorgenommen werden. Dies aber nur so lange, bis die maximale Geschwindigkeit vom Safety-Board nicht überschritten wird.

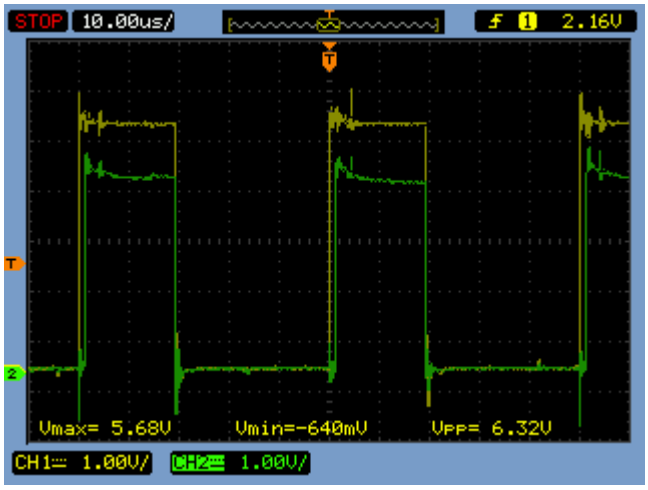


Abbildung 39: Motor-PWM

In Abbildung 39 ist Kanal 1 (CH1) die Eingangs-PWM und Kanal 2 (CH2) die Ausgangs-PWM des Safety-Board.

5.4.4 Verzögerungszeit

Es wird die Zeit ermittelt, die die Ausgangs-PWM, dem Eingangssignal des Safety-Boards nacheilt.

Ergebnis:
Es tritt eine kleine Zeitverzögerung zwischen Eingangssignal (CH1) und Ausgangssignal (CH2) auf. Diese Verzögerungszeit beträgt t=1,28µs und tritt beim Übergang von „0“ nach „1“ auf. (Abbildung 42).

Geht das Eingangssignal auf „Null“ folgt das Ausgangssignal sofort.

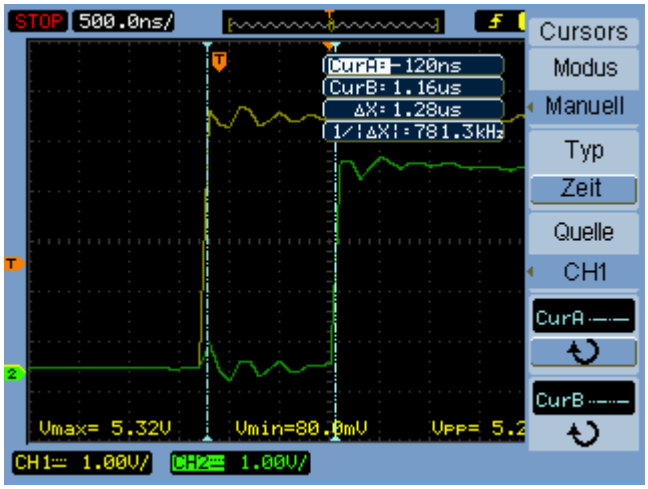


Abbildung 40: Verzögerungszeit

Diese Zeit kommt durch den Mikrocontroller und das UND-Gatter zustande. Der Mikrocontroller braucht eine gewisse Zeit bis er den Interrupt aufruft.

6 Zusammenfassung/Ausblick

Mit dem „Safety-Board“ kann ein Rahmen vorgegeben werden, in dem sich der Roboter bewegen kann. Hiermit wird überprüft, ob ein korrektes Heartbeat-Signal anliegt. Das Board gibt den Antrieb des mobilen Roboters nur frei, wenn das korrekte Signal erkannt wird. Des Weiteren gibt das Board eine maximale Geschwindigkeit vor, die nicht überschritten werden kann. Der Antrieb des Roboters kann durch ein Relais, das zwischen Motor und Motoransteuerung angebracht ist, geschaltet werden. Ist das Relais nicht geschaltet, ist der Antriebskreis unterbrochen und der Roboter bleibt stehen.

Die Heartbeat-Abfrage erfolgt über den Mikrocontroller Atmega32 und parallel dazu über eine Analogschaltung. Die Analogschaltung besteht aus einem Tiefpassfilter mit nachgeschalteten Komparatoren. Mit dem Tiefpassfilter wird der Mittelwert der Eingangsspannung ermittelt. Ist das Heartbeat-Signal korrekt, muss ein Mittelwert von U_a=2,5V am Ausgang des Tiefpasses ankommen. Mit den Komparatoren wird geprüft, ob die Ausgangsspannung des Tiefpasses in einem vorgegebenen Fenster liegt. Ist das Heartbeat-Signal korrekt, liegt U_a in diesem Fenster. Mit Komparator2 wird die minimale Spannung vorgegeben und mit Komparator1 die maximale Spannung. (Schaltung: siehe Kapitel 3.6 Abbildung 14)

Um den Antrieb frei zu geben, wird zwischen Analogschaltung und Relais eine Logikschaltung geschaltet. Diese Logikschaltung besteht aus drei NAND-Gattern (Kapitel 3.6 Abbildung 15). Der Ausgang der Logikschaltung ist mit der Basis eines Transistors, der als Schalter dient, verbunden. Mit diesem Transistor können die Relais geschaltet werden. Durch die Logikschaltung wird überprüft ob Komparator2 und der Mikrocontroller eine „1“ und Komparator1 eine „0“ liefert. Tritt dieser Fall ein, ist das Heartbeat-Signal korrekt und die Relais geben den Antrieb frei.

Um die Geschwindigkeit zu begrenzen, gibt das „Safety-Board“ eine maximal zulässige PWM vor. Diese PWM wird mit dem Mikrocontroller generiert. Die Motor-PWM wird zwischen Robotersteuerung und Motoransteuerung abgegriffen. Auf dem „Safety-Board“ werden die beiden PWM-Signale mit einer logischen UND-Funktion verknüpft. Ein Problem das dabei Auftritt ist die Synchronisation der beiden Signale. Diese wird mit dem Controller über einen Interrupt realisiert.

Zum Testen der Grundfunktionen des Boards und des µC ist ein Testprogramm entwickelt worden. Dies ist in der Programmiersprache „C“ geschrieben.

Über eine ISP-Schnittstelle kann der Controller programmiert werden. Zur Kommunikation mit dem Hauptprogramm ist auf dem Board eine RS232-Schnittstelle implementiert.

Als Ergebnis entstand ein Board, mit dem eine Grundsicherheit des Antriebs garantiert werden kann. Durch die Tests in Kapitel 5 wurde bewiesen, dass die Platine die gewünschten Anforderungen erfüllt. Diese Tests haben ergeben, dass das Heartbeat mit der Analogschaltung abgefragt werden kann und der Antrieb des Roboters nur freigegeben wird, wenn dieses auch korrekt ist. Sobald ein fehlerhaftes Heartbeat-Signal anliegt, schaltet das Relais den Antrieb ab. Ebenfalls zeigen die Tests, dass die Grundfunktionen des µC funktionieren. Der letzte Test zeigt, dass die Geschwindigkeit des Roboters über das Safety-Board begrenzt werden kann.

Das „Safety-Board“ kann in den „Pioneer3“ eingebaut werden. Dazu muss noch eine mechanische Halterung im Roboter angebracht werden, auf der das Board befestigt wird.

Das Mikrocontrollerprogramm wird in einem weiteren Projekt realisiert, weil die Anforderungen an sicherheitsrelevante Software sehr hoch sind.

Mit dem Board wird eine Grundsicherheit gewährleistet. Um eine größere Sicherheit garantieren zu können, sollte das Board mit zusätzlichen Sensoren ausgestattet werden. Dies könnten zum Beispiel ein Neigungssensor oder ein Abstandssensor sein. Im derzeitigen Entwicklungszustand wird nur eine maximale Geschwindigkeit vorgegeben. Es ist auch denkbar, dass ein Geschwindigkeitssensor angebracht wird, um die aktuelle Geschwindigkeit zu messen.

7 Quellenverzeichnis

[1] Blockschaltbild Aufgabenstellung
- Präsentation „SafetyBoard-Einführung.pdf“ von Philipp Ertle, HS Ravensburg-Weibngarten

[2] Pioneer3
- Handbuch „Pioneer3 Operations Manual“,Version5, Juli2007 von Mobilerobots

[3] Sicherheit
- Präsentation „SafetyBoard-Einführung.pdf“ von Philipp Ertle, HS Ravensburg-Weingarten

[4] Mikrocontroller
- Microcontroller www.rn-wissen.de/index.php/Microcontroller, Stand: 09. Februar 2010

[5] Abbildung 4
- „Typische Anwendungen für Mikrocontroller“ Skript von Prof.Dr.-Ing. Franz Brümmer, HS Ravensburg-Weingarten

[6] Dimensionierung der Kondensatoren für die Spannungsstabilisierung
- Datenblatt „7805“ Seite 21 Figure 16 „Fixed output regulator“ von der Firma „ST Microelectronics“

[7] ISP- In System Programming
- Handbuch „RN-Control Version 1.4“ von Roboternetz.de Stand: 18.7.2007

[8] RN-Definitionen
- http://www.rn-wissen.de/index.php/RN-Definitionen, Stand: 24. Februar 2010

[9] Atmega 32
- Datenblatt „Atmega32“ von der Firma Atmel, Kapitel „Features“

8 Formelzeichen

Formelzeichen	Einheit	Beschreibung
U_e	V	Eingangsspannung Tiefpass (Heartbeat)
U_a	V	Ausgangsspannung Tiefpass
U_{pp}	V	Spitze-Spitze Spannung Heartbeat
U_{min}	V	minimale Spannung
U_{max}	V	maximale Spannung
U_M	V	Motorspannung
τ	s	Zeitkonstante Tiefpass
T	s	Periodendauer Motor-PWM
T_{Ein}	s	Einzeit PWM
t_{an}	s	Anschaltzeit Relais nach Erkennen des Heartbeat
t_{ab}	s	Abschaltzeit Relais nach Ausfall des Heartbeat
f_H	Hz	Frequenz Heartbeat
f_{OSC}	Hz	Frequenz Quarz
f_{LED}	Hz	Blinkfrequenz LED
f_{PWM}	Hz	Frequenz der PWM

Tabelle 15: Formelzeichen

9 Anhang

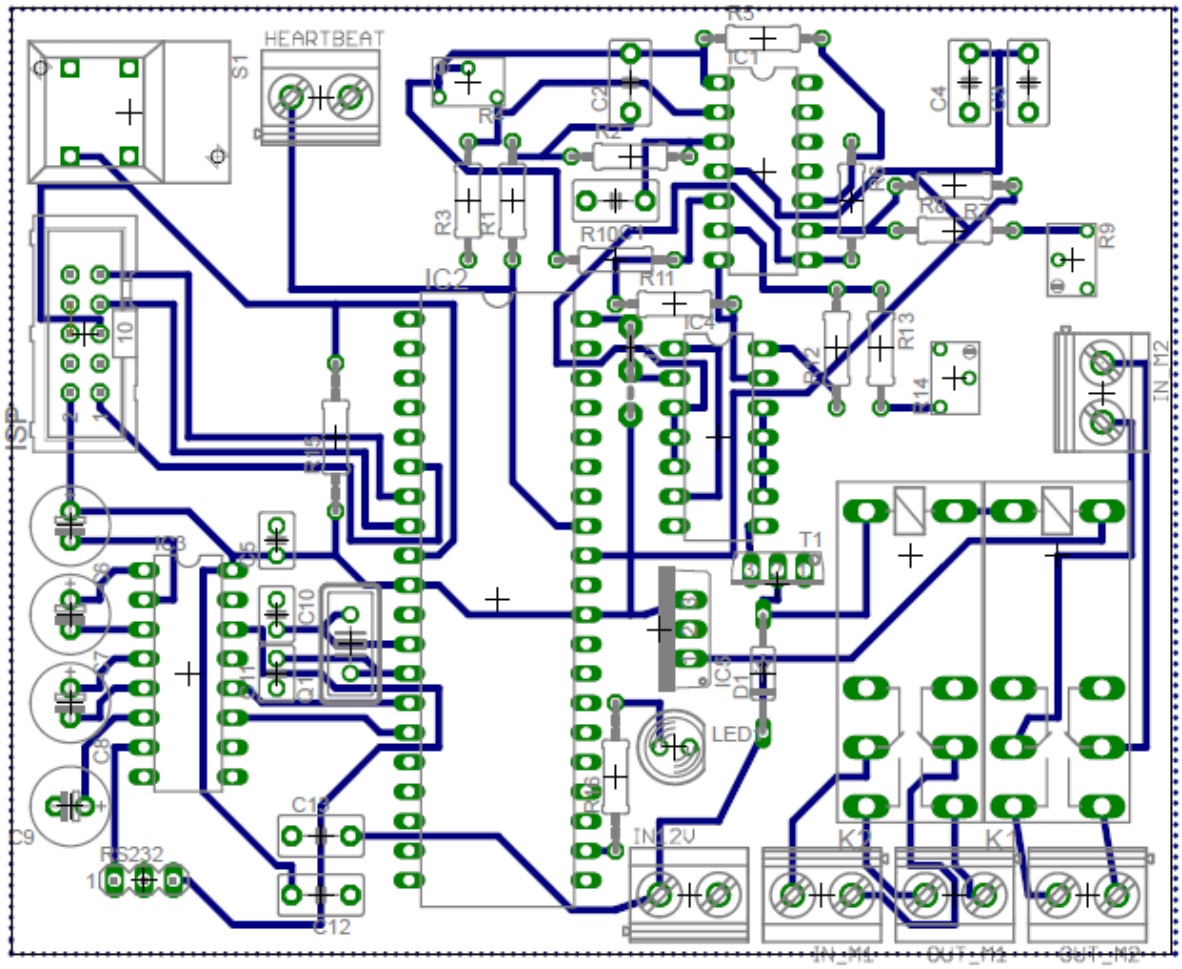
Anhang A: Verzeichnisse

Abbildungsverzeichnis

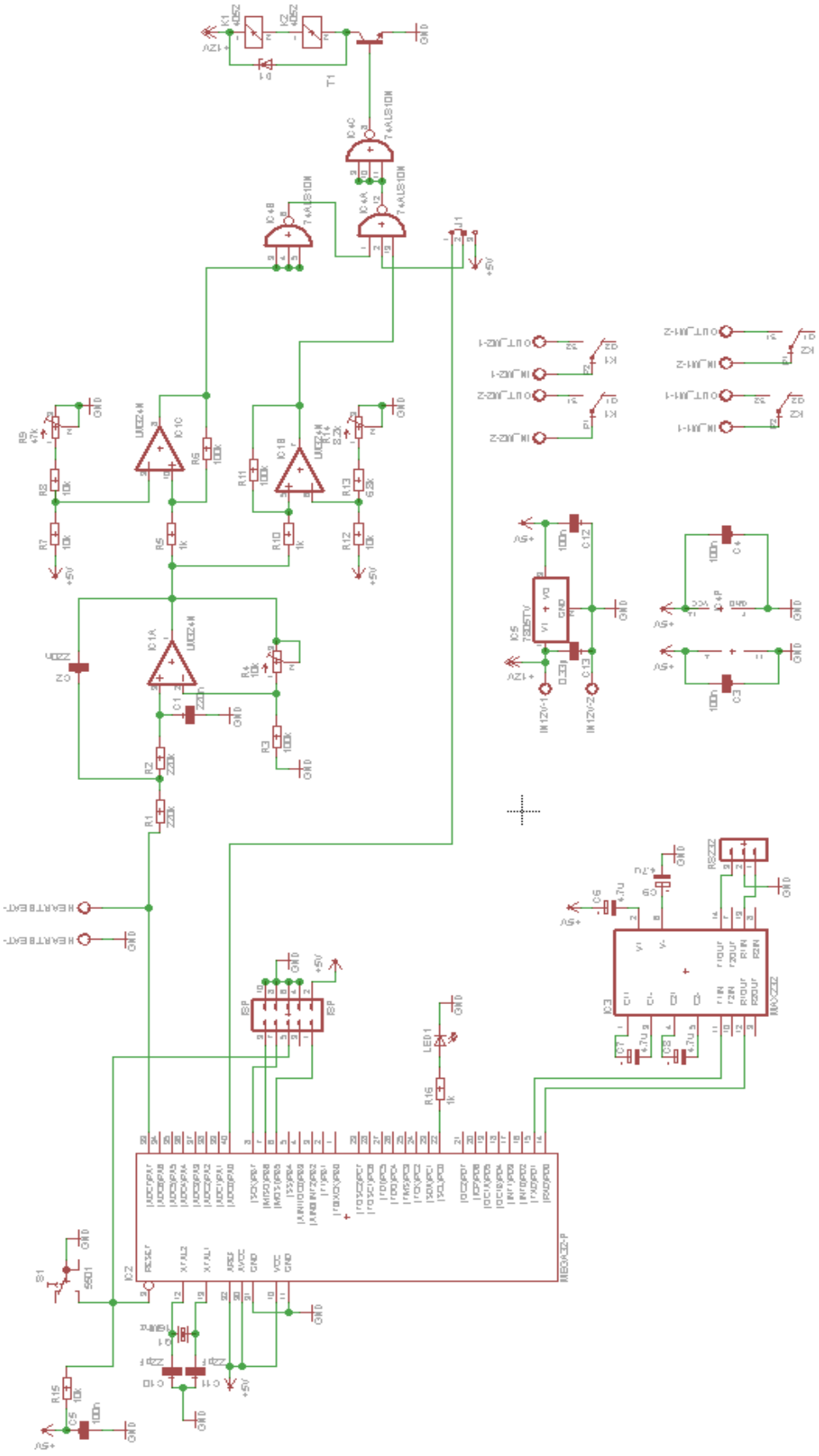
Tabellenverzeichnis

Anhang B: Board V1.0

Platinenlayout:

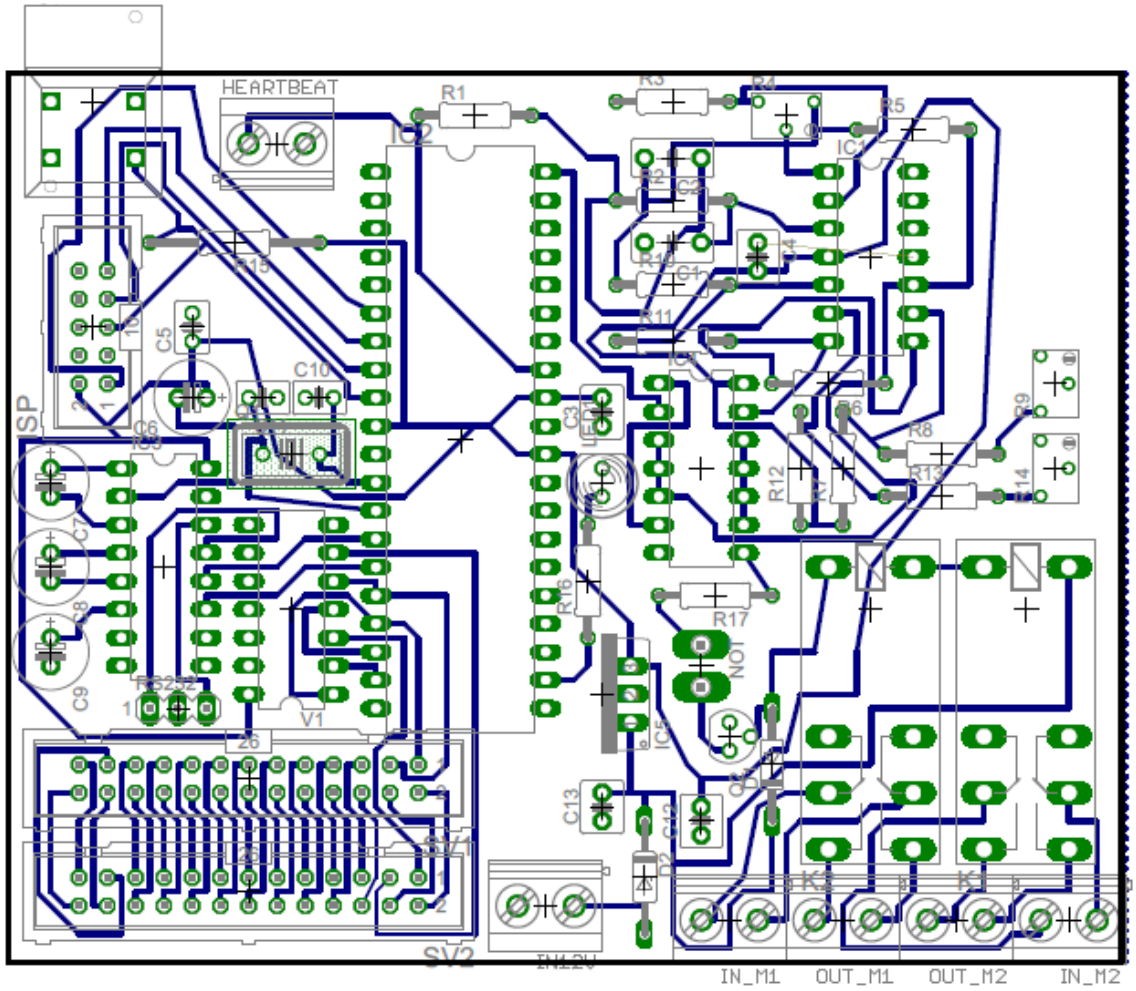


Schaltplan:

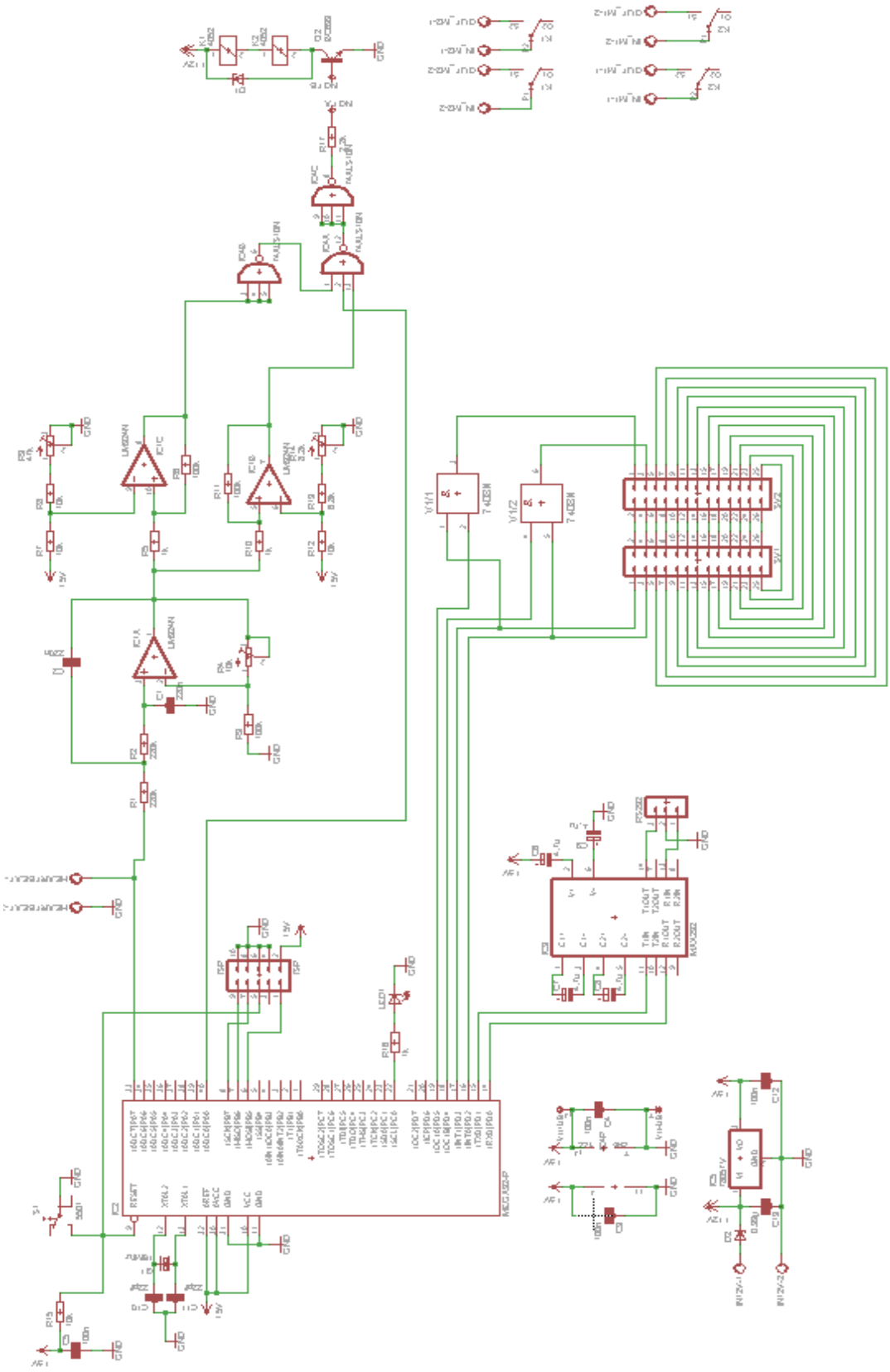


Anhang C: Board V1.1

Platinenlayout:



Schaltplan:



Anhang D: Quellcode Testprogramm

```
#ifndef MCU
#define MCU atmega32
#endif
#ifndef F_CPU
#define F_CPU 16000000UL //16MHz
#endif
#include <util/delay.h>
#include<stdlib.h>
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>


ISR(INT0_vect){
// PWM Eingänge setzen und dann 25µs warten, anschließen wieder
zurücksetzen
          PORTD |= (1<<PD4);
          PORTD |= (1<<PD5);
          _delay_us(25);
          PORTD &= ~(1<<PD4);
           PORTD &= ~(1<<PD5);

}
//Funktion zum Senden einzeler Char-Zeichen über RS232
void sendChar(unsigned char c){
     while(!(UCSRA & (1<<UDRE))) {} //Warten, bis Senden mög-
lich ist
     UDR=c; // schreiben von c
     return;
}

// Funktion: String als einzel Zeichen an sendChar zu übergeben

void sendUSART(char *s){
     while(*s){
          sendChar(*s);
          s++;
     }
}


int main(void)
{

    DDRA = 0x00; // Port A als Eingang festlegen
     DDRA = (1<<PA0); // PortA0 als Ausgang festlegen
     DDRC = 0xff; //Port C als Ausgang festlegen

     //externen Interrupt IN einschalten

     MCUCR |= (1<<ISC01);
     MCUCR |= (1<<ISC00);
     GICR |= (1<<INT0);
```

```
     sei();

     //Als Ausgänge definieren umd die max.PWM vorgeben
     DDRD |= (1<<PD4) | (1<< PD5);
     // Beide Ausgänge auf Null setzen.
     PORTD &= ~(1<<PD4);
    PORTD &= ~(1<<PD5);

    while(1)
     {
          //Ausgänge setzten, damit keine Geschwindigkeitsbe-
grenzung statt findet
          /*
          PORTD |= (1<<PD4);
          PORTD |= (1<<PD5);
          */

          /*
          Heartbeattest
          Die LED an PC0 soll Blinken wenn ein Heartbeatsignal
eingelesen wird.
          Sie leuchtet, wenn das Heartbeat-Signal 1 ist
          */

          /*
          if(PINA & (1<<PINA7))
          {

               PORTC=0x01;


          }else
          {
                PORTC=0x00;
          }
          */

          /*
          // Ausgang PA0 (Relais-Ausgang) jede Sekunde Toggeln

          PORTA &= ~(1<<PA0);
          _delay_ms(1000);
          PORTA |= (1<<PA0);
          _delay_ms(1000);
          */

          //PA0 dauerhaft setzen, zum Testen der Analogschal-
tung
          PORTA |= (1<<PA0);

          // LED jede Sekunde Toggeln
          PORTC=0x01;         //LED anschalten
          _delay_ms(1000);
          PORTC=0x00;         //LED ausschalten
          _delay_ms(1000);
```

```
          /*
          // Ausgabe über RS232-Schnittstelle
          UCSRB |= (1<<TXEN); //UART TX (Senden) einschalten
          UCSRC |= (1<<URSEL)|(3<<UCSZ0);    //Modus Asynchron
8N1 (8 Datenbits, No Parity, 1 Stopbit)
          UBRRH = 0;     //Highbyte ist 0
          UBRRL = 103;   //Lowbyte ist 103 Baudrate=9600
          sendUSART("Hallo :-) \r\n"); //Senden von "Hallo :-)"
          */
      }

}
```

Anhang E: Stückliste

Bauteil	Wert	Beschreibung
R_1,R_2	220kΩ	Widerstand 220kΩ
R_3,R_6,R_{11}	100 kΩ	Widerstand 100kΩ
R_7,R_8,R_{12},R_{15}	10kΩ	Widerstand 10kΩ
R_5,R_{10},R_{16}	1kΩ	Widerstand 1kΩ
R_{13}	6,8 kΩ	Widerstand 6,8kΩ
R_{17}	2,2kΩ	Widerstand 2,2kΩ
R_4	10kΩ	Potentiometer 10kΩ
R_9	47kΩ	Potentiometer 47kΩ
R_{14}	8,2kΩ	Potentiometer 8,2kΩ
C_1,C_2	220nF	Kondensator 220nF
C_5,C_3,C_4,C_{12}	100nF	Kondensator 100nF
C_{10},C_{11}	22pF	Kondensator 22pF
C_{13}	0,33µF	Kondensator 0,33µF
IC_1	LM324	Komparator LM324
IC_2	Atmega32	Mikrocontroller Atmega32
IC_3	MAX232	RS232 Treiber MAX232
IC_4	7410	3-fach NAND-Gatter 7410
IC_5	7805	Festspannungsregler 7805
V_1	7408	UND-Gatter 7408
D_1		Schutzdiode für Relais
D_2		Schutzdiode Spannungsversorgung
LED_1		Leuchtdiode
Q_1	16MHz	Quarz 16 MHz
Q_2	BC639	npn-Transistor BC639
K_1,K_2		Relais Type 40.52
S_1		Miniaturtaster
RS232		Stiftleiste 3 polig
ISP		Wannenbuchse 10 polig
SV_1,SV_2		Wannenbuchse 26 polig
Heartbeat,+12V		Schraubklemme 2 polig
IN_M1,IN_M2		Schraubklemme 2 polig
OUT_M1,OUT_M2		Schraubklemme 2 polig