Ein Bild, das Text, draußen, Schild, Teller enthält.

Automatisch generierte Beschreibung

DIPLOMARBEIT

SumoBots

Schuljahr 2023/24

Ausgeführt im Schuljahr 2023/24 von:

EISMANN Bastian | 5AHEL

LUCUT Ioan Lukas | 5AHEL

ZICKLER Yannick | 5AHEL

Betreuer/Betreuerin:

DI (FH) Michael Zatl

**Erklärung über die eigenständige Verfassung**

**der Diplomarbeit**

Wir, die Herren Bastian EISMANN, Ioan LUCUT und Yannick ZICKLER, Schüler der Klasse 5AHEL der Höheren Technischen Bundeslehranstalt Wien 10, erklären hiermit an Eides statt, dass wir die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst haben, einschließlich auch andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die benutzten Quellen, wörtlich und inhaltlich entnommenen Stellen, als solche erkenntlich in der Diplomarbeit gekennzeichnet haben.

Wien am 31.05 2024

Verfasser:

Bastian EISMANN | 5AHEL U: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ioan Lukas LUCUT | 5AHEL U: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Yannick ZICKLER | 5AHEL U: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Danksagung

xxxxxxxx

Inhalt

[1 Danksagung 3](#_Toc153040148)

[3 Zusammenfassung (Deutsch) 6](#_Toc153040149)

[3.1 Abstract (English) 6](#_Toc153040150)

[4 Einleitung – Projektbeschreibung 8](#_Toc153040151)

[4.1 Vorgeschichte – Ausgangslage 9](#_Toc153040152)

[4.1.1 Projektwoche 2023 9](#_Toc153040153)

[4.1.2 CLIL 2023 9](#_Toc153040154)

[5 Struktur des Projekts – Arbeitspakete 10](#_Toc153040155)

[5.1 Objektstrukturplan (OSP) / Arbeitspakete 10](#_Toc153040156)

[5.2 Projektstrukturplan (PSP) 11](#_Toc153040157)

[5.3 Arbeitspaket 1 3D-Design (EIS) 11](#_Toc153040158)

[5.4 Arbeitspaket 2 Hardware-Entwicklung (LUC) 11](#_Toc153040159)

[5.5 Arbeitspaket 3 Software-Entwicklung (ZIC) 11](#_Toc153040160)

[5.6 Arbeitspaket 4 Projektmanagement 12](#_Toc153040161)

[5.7 Arbeitspaket 5 Bedienungsanleitung 12](#_Toc153040162)

[6 3D-Design und Konstruktion 12](#_Toc153040163)

[6.1 Materialeigenschaften und Konstruktionsvorgänge 12](#_Toc153040164)

[6.2 3D-Modellierung des Roboters 12](#_Toc153040165)

[6.3 3D-Modellierung des Controllers 12](#_Toc153040166)

[7 Hardwareentwicklung 12](#_Toc153040167)

[7.1 Konzept 12](#_Toc153040168)

[7.2 Entwicklungsablauf 13](#_Toc153040169)

[7.3 Hardware-Design Roboter 14](#_Toc153040170)

[7.3.1 Roboter Revision A 14](#_Toc153040171)

[7.3.2 Motorshield 16](#_Toc153040172)

[7.3.3 H-Brücke 17](#_Toc153040173)

[7.3.4 NRF24L01 Funkmodul 17](#_Toc153040174)

[7.3.5 Serial Peripheral Interface Bus 18](#_Toc153040175)

[7.3.6 Spannungsversorgung des Funkmoduls 19](#_Toc153040176)

[7.4 Roboter Revision B 21](#_Toc153040177)

[7.4.1 Gyroskop MPU6050 21](#_Toc153040178)

[7.5 Hardware-Design Controller 22](#_Toc153040179)

[7.5.1 xxxTriggerxxx 23](#_Toc153040180)

[7.5.2 Joystick 23](#_Toc153040181)

[8 Softwareentwicklung 23](#_Toc153040182)

[8.1 Konzept, Programmierung und Debugging 24](#_Toc153040183)

[8.1.1 Konzept 24](#_Toc153040184)

[8.1.2 Programmierung und Debugging 24](#_Toc153040185)

[8.1.3 In-System-Programming über SPI-Bus 24](#_Toc153040186)

[8.1.4 Versionsverwaltung und Style – Guide 25](#_Toc153040187)

[8.1.5 Variablen- und Funktionsnamen 26](#_Toc153040188)

[8.2 Software-Design Controller 26](#_Toc153040189)

[8.2.1 Ansteuerung der Status-LED 26](#_Toc153040190)

[8.2.2 Auslesen der Elektronik 27](#_Toc153040191)

[8.3 Software-Design Roboter 27](#_Toc153040192)

[8.3.1 Ansteuerung der Status-LED 27](#_Toc153040193)

[8.3.2 Berechnung der Motorsteuerung 27](#_Toc153040194)

[8.3.3 Ansteuerung des Motorshields 29](#_Toc153040195)

[8.3.4 Gyroskop – Code 30](#_Toc153040196)

[8.3.5 Einstellen des Watchdog-Timers 31](#_Toc153040197)

[8.3.6 Debug - Mode 32](#_Toc153040198)

[8.4 Networking 32](#_Toc153040199)

[8.4.1 Initialisierung des Moduls 33](#_Toc153040200)

[8.4.2 Standard – Datenpakete 33](#_Toc153040201)

[9 Projektmanagement 34](#_Toc153040202)

[10 Bedienungsanleitung 34](#_Toc153040203)

[11 Finale Ergebnisse 34](#_Toc153040204)

[12 Tabellen und Abbildungsverzeichnis 34](#_Toc153040205)

[13 Literaturverzeichnis und Quellenangaben 34](#_Toc153040206)

[14 Anhänge 35](#_Toc153040207)

[15 3D-Modelle und Konstruktionszeichnungen 35](#_Toc153040208)

[16 Schaltpläne und PCB Designs 35](#_Toc153040209)

[17 Programm-Codes 35](#_Toc153040210)

[18 Mess & Labor-Protokolle 35](#_Toc153040211)

# Zusammenfassung (Deutsch)

Das Ziel dieser Diplomarbeit war es, einen Fahrzeugroboter, welcher durch eine Funkverbindung mit einem selbst entwickeltem Controller, gesteuert werden kann, zu entwickeln und konstruieren. Joystick und Schultertasten des Controllers geben Geschwindigkeit und Richtung des Roboters vor.

In diesem Projekt werden zwei separate Printed Circuit Boards (PCBs) entwickelt, die mithilfe von KiCAD entworfen werden. In jeden der beiden Platinen ist ein AVR-Mikrocontroller integriert, der für die Steuerung der jeweiligen Schaltung verantwortlich ist. Die Kommunikation zwischen den beiden erfolgt über das NRF24L01+ Funkmodul.

In der Diplomarbeit spielen die Auswahl des richtigen Materials und die Kalibrierung der 3D-Drucker eine zentrale Rolle. Die Gewährleistung des Druckererfolgs erforderte wesentliche Verbesserungen am Drucker. Ein weiterer entscheidender Schwerpunkt liegt auf der Modellierung von Gehäusen, Rädern, Antriebssystemen und dem Controller. Hierbei müssen zahlreiche Variablen berücksichtigt werden, um den Anforderungen verschiedener Abteilungen gerecht zu werden.

Die Mikrocontroller werden mit der Arduino – Umgebung und SPI – Schnittstelle programmiert. Der Code des Controllers muss seine Eingabeelemente einlesen und an den Roboter weiterschicken. Dieser muss die Werte einlesen, berechnen wie stark er den linken und rechten Motor ansteuern muss, und diesen Wert mit PWM – Signal an das Motorshield weiterleiten.

## Abstract (English)

The goal of the diploma thesis is to develop and construct a vehicle robot that can be controlled through a wireless connection with a self-designed controller. The joystick and shoulder buttons of the controller determine the speed and direction of the robot.

In this project, two separate Printed Circuit Boards (PCBs) will be developed using KiCAD. Each of these boards will integrate an AVR microcontroller responsible for controlling its respective circuit. Communication between the two will be facilitated through the NRF24L01+ wireless module.

The thesis focuses on the selection of appropriate materials and the calibration of 3D printers, which are crucial for ensuring successful printing. Significant improvements to the printer were necessary to guarantee printing success. Another key emphasis is placed on modeling the robot's housing, wheels, propulsion systems, and the controller. This involves considering numerous variables to meet the requirements of different departments.

The microcontrollers will be programmed using the Arduino environment and SPI interface. The controller's code must read its input elements and transmit them to the robot. The robot, in turn, will need to read these values, calculate how much to control the left and right motors, and send this value to the motor shield using a PWM signal.

# Einleitung – Projektbeschreibung

xxxx

Abbildung : Systemarchitektur des Projekts

## Vorgeschichte – Ausgangslage

### Projektwoche 2023

Im Zuge der vergangenen Projektwoche wurde bereits ein Roboter und eine Fernsteuerung entwickelt und in Betrieb genommen. Beide Prototypen wurden von dem   
AVR-Mikrocontroller ATMEGA 328p angesteuert. Die Kommunikation zwischen den beiden, wurde durch das Funkmodul NRF24L01+ ermöglicht, welches die Daten vom Controller an den Roboter sendet. Mit diesen Daten steuert das implementierte Motorshield Cytron MDD3A die Motoren. Als Spannungsquelle wurde ein 12V Bleiakku genutzt.

### CLIL 2023

Während des CLIL-Unterrichts 2023, wurde als Ziel die Weiterentwicklung der entwickelten SumoBots, sowie die Verbesserung von Konstruktion und Hardwarefehlern gesetzt, um die Milestones der Diplomarbeit genau definieren zu können. Bei der Konstruktion wurde auf das Arbeiten mit Fusion 360 gesetzt, um verbesserte und professionellere 3D-Modelle konstruieren zu können. In der Hardwareentwicklung wurde ein Fehler, welcher im Zusammenhang mit den unterschiedlichen Logikpegeln des Mikrocontrollers und Funkmoduls steht, durch einen Logic-Level-Converter verbessert. Des Weiteren wurden einige Recherchen, sowie eine Testschaltung entwickelt. In der Software wurde der Fokus auf   
In-System-Programming gelegt, in welchen einige Testcodes geschrieben wurden.

# Struktur des Projekts – Arbeitspakete

## Objektstrukturplan (OSP) / Arbeitspakete

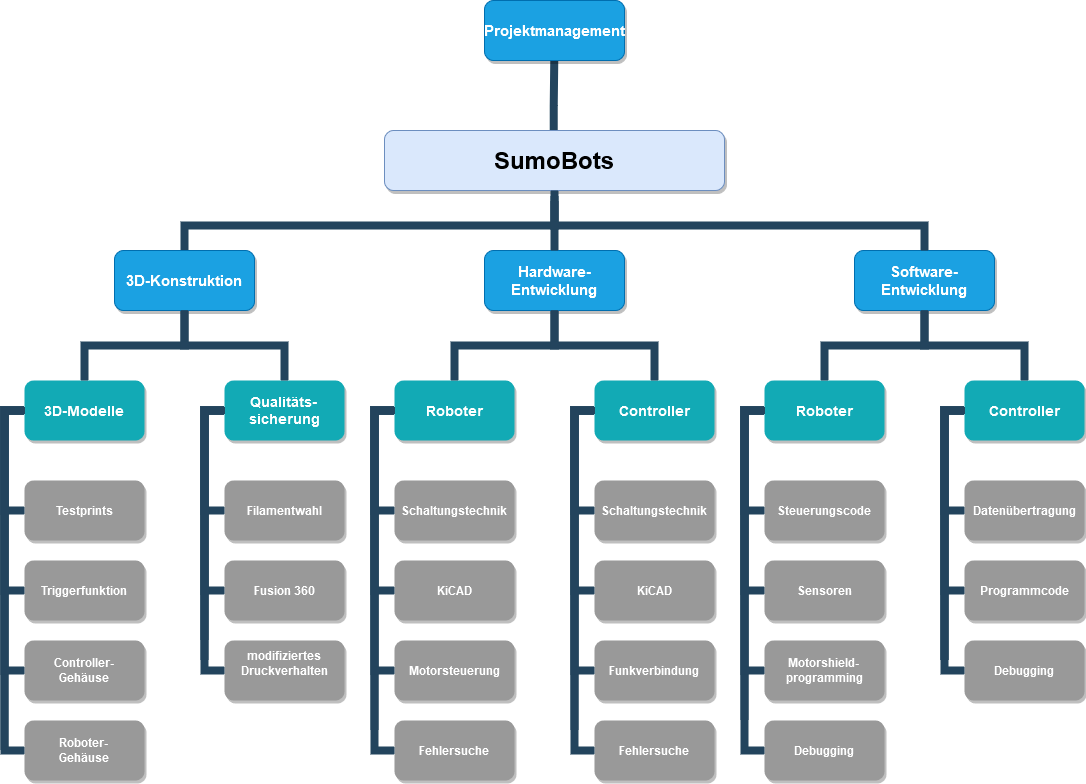


Abbildung : Objektstrukturplan (OSP) / Arbeitspakete

Um einen besseren Einblick der Tätigkeiten, sowie deren Aufteilung in den Arbeitspaketen zu gewähren, wird die Diplomarbeit in einem Objektstrukturplan dargestellt. Hier wurden die dazu gehörigen Hilfsmittel, Aufgaben, sowie die benötigten Fähigkeiten für das jeweilige Arbeitspaket genau beschrieben.

## Projektstrukturplan (PSP)

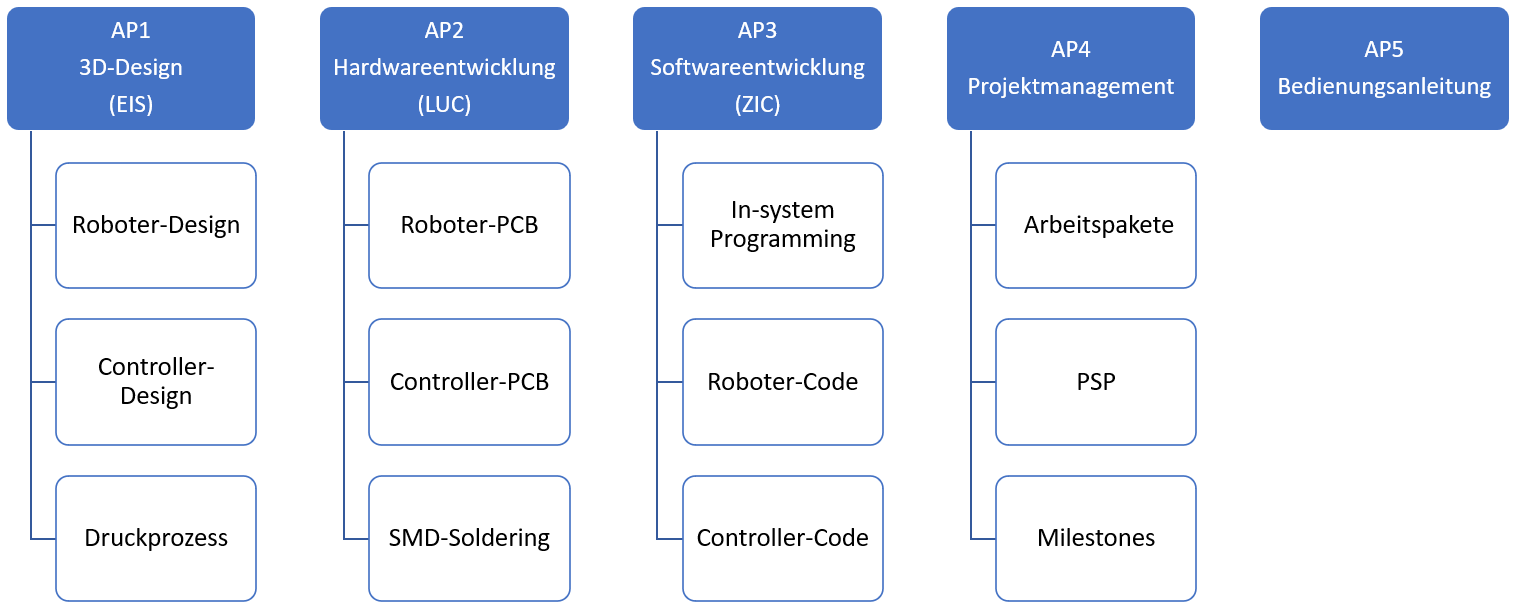


Abbildung :Projektstrukturplan (PSP) AP1-AP6

Die im oberen Diagramm beschriebenen Tätigkeiten wurden in Arbeitspakete und daraufhin auf die Diplomanden aufgeteilt.

## Arbeitspaket 1 3D-Design (EIS)

Im Arbeitspaket „3D-Konstruktion“ wird neben den 3D-Modellen, ebenfalls Qualitätssicherung benötigt, da dieses Aufgabengebiet die Stabilität und Qualität der 3D-Prints sicherstellen.

## Arbeitspaket 2 Hardware-Entwicklung (LUC)

Die Schaltungsentwicklung wird im Arbeitspaket „Hardware-Entwicklung“ bearbeitet, in welchen der Entwicklungsprozess des Roboters und des Controllers Schrittweise abgebildet ist. Der Hauptfokus liegt hierbei in der Entwicklung des Schaltplans und des darauffolgenden Platinenlayouts, welches in KiCAD designet wird. Es werden hierbei doppelseitige Platinen erstellt, um die in der Projektwoche eingebauten Drahtbrücken zu vermeiden. Diese Platinen werden zum Großteil mit SMD-Bauteilen bestückt, daher ist das SMD-Löten ein wichtiger Bestandteil und stellt damit auch ein Risikofaktor des Arbeitspakets da.

## Arbeitspaket 3 Software-Entwicklung (ZIC)

Das Arbeitspaket „Software-Entwicklung“ beinhaltet Softwaredesign, Debugging, Ansteuerung der SPI und I2C Buse zur Programmierung von µC und einlesen verschiedener Sensormodule und die Kommunikation mithilfe von Funkmodulen zwischen Controller und Roboter sowie Roboter und Server.

## Arbeitspaket 4 Projektmanagement

Die Struktur, Koordination und Organisation der Diplomarbeit wird im Arbeitspaket „Projektmanagement“ bearbeitet und beinhaltet sämtliche Pläne, Zielsetzung, sowie der dazugehörigen erstellten Milestones-Pläne und Arbeitszeiterfassungen. Um die Arbeitszeiten, samt den Tätigkeiten der Diplomanden im Blick zu behalten, werden diese erfasst und in eine Arbeitszeiterfassung AZE dokumentiert. Des Weiteren werden die Arbeitspakete in einem Projektstrukturplan auf die Diplomanden aufgeteilt, die die Tätigkeiten der Diplomanden auf gewählte Bereiche einschränken. Dazu wird der Fortschritt des jeweiligen Arbeitspakets in einem Granttchart in Projektlibre als Graph dargestellt.

## Arbeitspaket 5 Bedienungsanleitung

# 3D-Design und Konstruktion

## Materialeigenschaften und Konstruktionsvorgänge

## 3D-Modellierung des Roboters

## 3D-Modellierung des Controllers

# Hardwareentwicklung

Im folgenden Kapitel werden die Funktionen, sowie die jeweiligen Entwicklungsschritte der Hardware des Roboters und Controllers erklärt. In Zuge dessen wird das Konzept der Steuerung, sowie der Kommunikation mit der dazugehörigen elektronischen Schaltung aufgegriffen.

## Konzept

xxx

## Entwicklungsablauf

Da in diesen Teil des Arbeitspakets mehrere Fehler auftreten können, wurde ein Flussdiagramm erstellt, der die Abläufe der Entwicklung, sowie weitere optionale Schritte darstellt. Das Vorplanen der Tätigkeiten ist hier besonders wichtig, aufgrund der aufkommenden Kosten der durch die PCB-Bestellungen von externen Firmen entsteht.

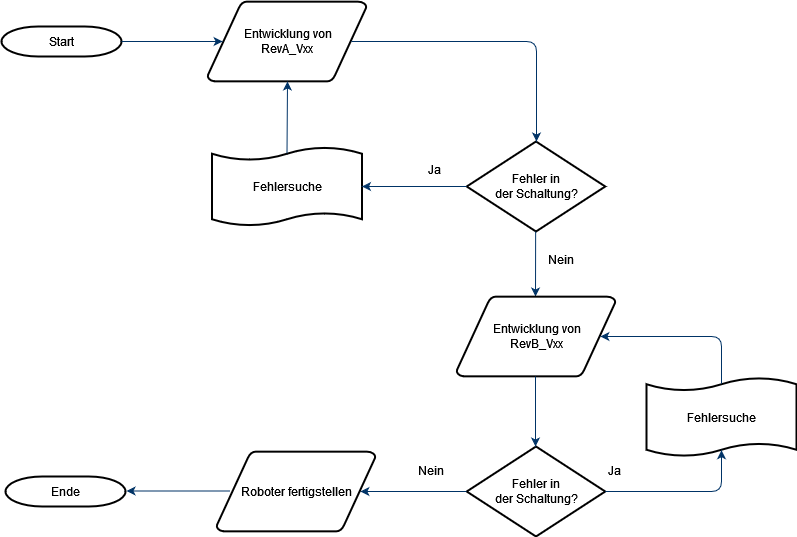


Abbildung Entwicklungsablauf der Roboter Hardware

Dabei wurde die Hardware des Roboters in Revision A (RevA) und Revision (RevB) geteilt. In   
Revision A wird eine Platine mit den elektronischen Grundkomponenten entworfen, die benötigt werden den Roboter zum Fahren zu bringen. Bei Revision A wird der Fokus auf das Ermitteln und Korrigieren von vermeintlich auftretenden Fehlern gelegt. Falls in der ersten Version des PCBs gravierende Fehler auftreten, die das in Betrieb nehmen der Schaltung nicht ermöglichen, wird eine weitere Version von Revision A entwickelt. Wenn dieser optionale Fall nicht eintritt, wird die Revision B Schaltung entwickelt, bei welcher die funktionsfähige Schaltung von Revision A mit Sensoren und weiteren Bauelementen aufgewertet wird.

## Hardware-Design Roboter

In diesem Unterkapitel wird der Aufbau der jeweiligen Roboterschaltung erklärt, sowie die Funktion deren eingebauten Komponenten.

### Roboter Revision A

Wie bereits beschrieben, beschränken sich die Hauptaufgaben dieses Schaltungsentwurfs auf die Kommunikation mit einem Controller, die Steuerung der Motoren mittels dieser, sowie der In-System-Programmierung.

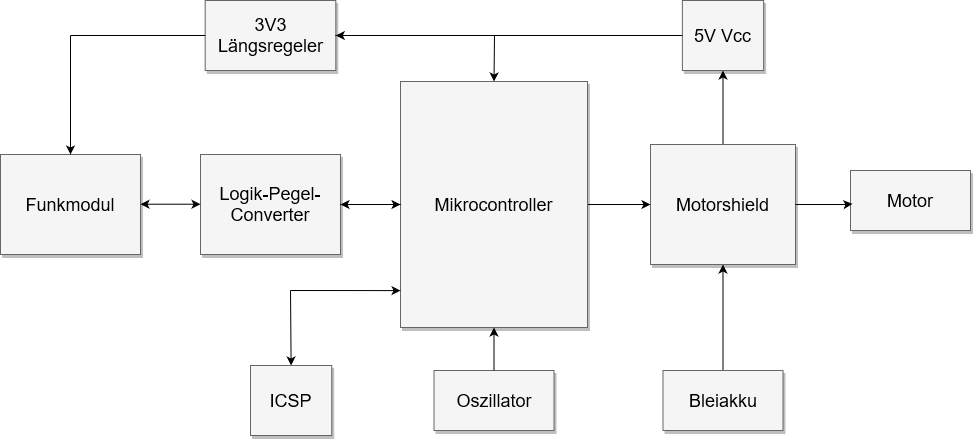


Abbildung Blockschaltbild von Roboter Rev A

Als Mikrocontroller wird der ATMEGA2560 eingebaut, der als essentieller Teil der Schaltung gilt. Damit dieser funktioniert, muss eine 5V Gleichspannungsquelle und ein digitales Taktsignal vorhanden sein. Diese Versorgungsspannung wird durch das Motorshield ermöglicht, da auf diesen ein Spannungsregler verbaut ist, der die 12V vom Bleiakku in 5V umwandelt. Die 12V vom Bleiakku werden benötigt, um die Motoren zu versorgen. Das Taktsignal wird durch einen externen Quarzoszillator mit 16MHz generiert, der die Geschwindigkeit der Programmabläufe beeinflusst. Des Weiteren wird eine 3.3V Spannungsquelle benötigt, um das Funkmodul NRF24L01+ in Betrieb zu nehmen. Das Funkmodul arbeitet hierbei mit unterschiedlichen Logikpegeln als der Mikrocontroller. Um Fehler bei der Übertragung von Daten zu vermeiden, wird ein Logik-Pegel-Wandler (Logic Converter) dazwischen geschaltet, der die Pegeln von 3.3V auf 5V umwandelt. Da es sich hierbei um ein selbstentwickeltes PCB handelt, ist keine serielle Schnittstelle vorhanden. Um den Mikrocontroller programmieren zu können, wird über den SPI-Bus programmiert. Die   
SPI-Pins (MOSI, MISO, SCK) sind auf den ICSP-Header vorhanden.

#### Roboter Revision A V1.00 Printed Circuit Board

Das im Kapitel 5.3.1 illustrierte Blockschaltbild wurde in KiCAD als Schaltplan, sowie als Platinenlayout realisiert. Aufgrund der dünnen Bahnen von 0,25mm können beim Ätzen Fehler auftreten. Daher wird für die Roboterplatinen auf den schulexternen Hersteller Aisler zurückgegriffen.

Ein Bild, das Schaltung, Text, Screenshot, Elektronik enthält.

Automatisch generierte Beschreibung

Abbildung 6 Roboter Rev A V1.00 PCB

Da hier das Löten mit SMD-Komponenten im Vordergrund steht, wurde der Platz zwischen den Bauteilen vergrößert, um leichter Fehler, die durch das Löten mit Heißluft zustande kamen, auszubessern. Im Fokus liegt hierbei der Mikrocontroller, der durch die Anzahl von 100 Pins, welche jeweils 0,25mm breit sind, fehleranfällig auf Kurzschlüsse ist. Des Weiteren wurden die Reihenfolge Pins des ICSP-Header vom Arduino UNO übernommen, um Fehler zu vermeiden. Nach dem Erhalt der Platinen machte sich ein Fehler beim Logik-Pegel-Wandler bemerkbar. Dieser ist weder mit der oberen (Front Mask) noch mit der unteren (Back Mask) Seite verbunden. Der Grund liegt hierbei an der Bibliothek des Konverters, der die Pins keinem Layer zugewiesen hat. Daher wurden Leitung vom ICSP-Header zum Logik-Pegel-Wandler, welcher daraufhin mit dem Funkmodul NRF24L01+ verlötet, um die Inbetriebnahme des PCBs zu ermöglichen.

Bild vom fertig gelötetem PCB

Mit der anschließenden bestückten Platine konnte die Kommunikation, samt der Fernsteuerung des Motorshields getestet werden und resultierte eine erfolgreiche Schaltung. Das Debugging der Software wird im Kapitel 8 Softwareentwicklung genauer beschrieben.

### Motorshield

Da die Information, die der Mikrocontroller ausgibt, lediglich 0 oder 1 ist (0V / 5V) kann ein einzelner DC-Motor diese nicht verarbeiten. Bei einer direkten Verbindung zwischen dem Motor und des Mikrocontroller würde sich dieser nur mit einer konstanten Geschwindigkeit in einer Richtung oder gar nicht bewegen. Deswegen ist es möglich mit Hilfe eines dazwischen geschalteten Motorshields einen Motor mit Software in verschiedenen Modi anzusteuern.

Ein Bild, das Text, Screenshot, Schrift, Rechteck enthält.

Automatisch generierte Beschreibung

Abbildung Motorshield-Blockschaltbild

Das Motorshield steuert mittels einer „H-Brücke“ den Motor mit PWM-Signalen an. Je nach Motorgröße ist eine weitere Spannungsquelle notwendig, da nicht jeder mit 5V in Betrieb genommen werden kann. Da in der Diplomarbeit ein 12V Motor benutzt wurde, wird im Blockdiagramm die dazu gehörige Spannungsquelle eingebaut.

### H-Brücke

Eine H-Brücke besteht aus vier Transistoren, welche in einer „H“-Form geschaltet sind, um den Stromfluss in zwei Richtungen durch den Motor steuern zu können. In diesem Motorshield besteht die H-Brücke aus N- und P-Channel MOSFETs.

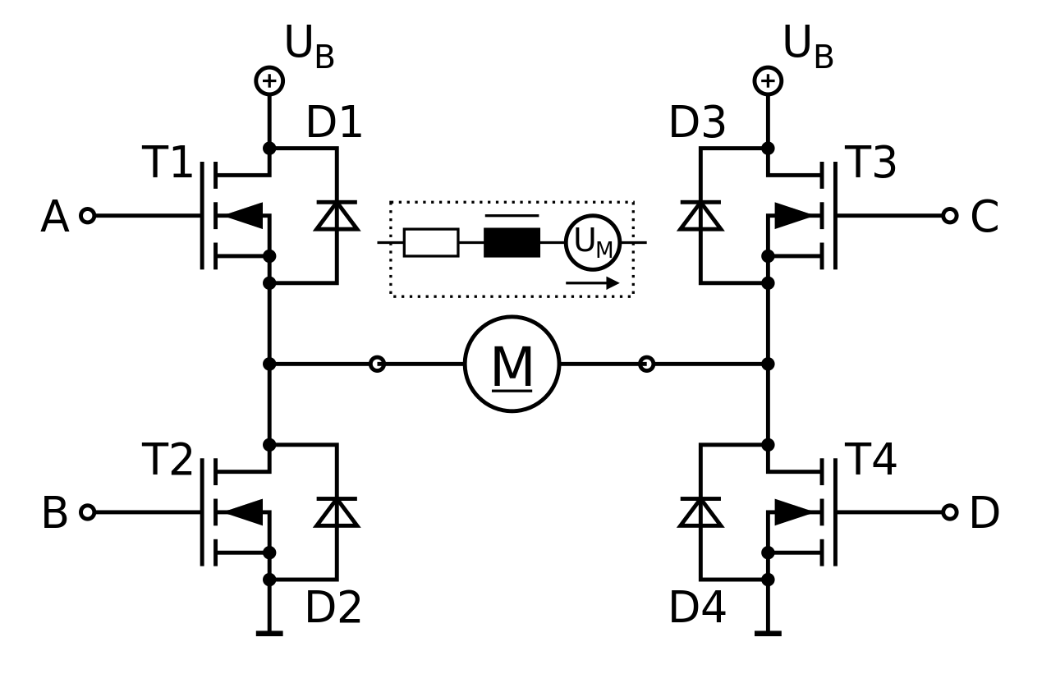


Abbildung H-Brücke Schaltplan ( <https://de.wikipedia.org/wiki/Vierquadrantensteller>)

Um eine Steuerung eines Motors zu ermöglichen, werden die Gate-Eingänge an den PWM-Pins des Mikrocontroller angeschlossen. Durch das Duty-Cycle-Verhältnis ist es nun möglich den Motor in verschiedenen Drehrichtungen mit einer dazugehörigen Geschwindigkeit rotieren zu lassen.

### NRF24L01+ Funkmodul

Die Kommunikation zwischen dem Controller und dem Roboter wird durch das Funkmodul NRF24L01+ ermöglicht. Das Funkmodul arbeitet auf einer 2.4GHz ISM-Frequenzband, mit welchen ein drahtloser Austausch ermöglicht wird. Die Informationen werden mittels der GFSK-Modulation (Gaussian frequency-shift keying) übertragen. Bei dieser handelt es sich um eine Art der Frequenzmodulation

#### Spannungsschwankungen des NRF24L01

Wenn man das Funkmodul als Receiver betreibt, können Spannungsschwankungen beim Empfangen von Daten auftreten, die den Datenaustausch hintern können.

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

Abbildung NRF24L01 Messung der Spannungsversorgung

In der oberen Messung wurde die Versorgungsspannung des Funkmoduls mit einem Oszilloskop gemessen. Hierbei kann man die Schwankungen, welche von 600mV – 800mV variieren erkennen, die während der Kommunikation zwischen zwei Modulen auftreten. Daher wurde ein C=10µF Elko zwischen den der Spannungsversorgung eingebaut. Um die resultierende Messung besser interpretieren zu können, wurde ein GPIO-Pin des Arduinos auf „1“ geschaltet.

Ein Bild, das Text, Screenshot, Diagramm, Zahl enthält.

Automatisch generierte Beschreibung

Abbildung NRF24L01 Messung der Spannungsversorgung mit C = 10µF Elko und Logiksignal

Die in der oberen Messung erkennbaren Störung tritt nur einmal zu Beginn des Datenaustausch auf. Mit einem größeren Kondensator ist eine Verbindung, samt einen korrekten Datenaustausch möglich.

### Serial Peripheral Interface Bus

Die Serial Peripheral Interface (SPI) ist ein synchroner serieller Bus ermöglicht eine effiziente bidirektionale Datenübertragung zwischen einem Master-Gerät und mehreren Slave-Geräten. Der Bus besteht typischerweise aus vier Hauptleitungen, wobei jede Leitung eine spezifische Funktion erfüllt.

Ein Bild, das Text, Diagramm, Reihe, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung SPI-Bus (<https://www.rc-modellbau-portal.de/index.php?threads/spi-schnittstelle-beim-arduino-co.6994/>)

Die MOSI-Leitung (Master Out Slave In) dient dazu, dass der Master Daten an die Slaves übertragen kann. Umgekehrt ermöglicht die MISO-Leitung (Master In Slave Out) den Slaves, Daten an den Master zu senden. Die SCLK-Leitung (Serial Clock) überträgt den Takt vom Master an die Slaves, wodurch eine synchronisierte Datenübertragung möglich wird. Die SS/CS-Leitung (Slave Select/Chip Select) wird für jedes Slave-Gerät separat verwendet, um es auszuwählen und die Kommunikation mit dem Master zu initiieren.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Abbildung SPI-Bus CLK-Konfiguration (<https://edistechlab.com/wp-content/uploads/2020/11/Screenshot-2020-11-22-at-09.37.57.png>)

Des Weiteren muss eine Synchronisation des Protokolls geeinigt werden, welche mit der CLK-Leitung gewählt wird. Dadurch ist es möglich den Zeitpunkt der Datenübertragung, welche durch fallende oder steigender Flanke, sowie des aktiven Zustandes (LOW/HIGH) zu konfigurieren.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Abbildung SPI-Bus Kommunikation (https://edistechlab.com/wie-funktioniert-spi/?v=fa868488740a)

Die Kommunikation erfolgt durch das gleichzeitige Senden und Empfangen von Daten, in einem vorher festgelegten Taktzyklus. Zuerst wird mit der CS-Leitung bestimmt, ob der Master oder der Slave senden darf. Danach kommt es auf die Konfiguration des Clocksignal an, in welcher bestimmt wird, wann die Daten übertragen werden. Als letztes werden die Daten über der MOSI- oder MISO-Leitung Bit für Bit in Serie versendet.

### Spannungsversorgung des Funkmoduls

Für den Betrieb des Funkmoduls ist eine Spannungsversorgung von 3,3V erforderlich. Daher wird eine Schaltung benötigt, die die 5V Betriebsspannung des Mikrocontrollers in 3,3V umwandelt. Um dies zu ermöglich wird der MCP1703A als Längsregler eingebaut, da dieser einen simplen Schaltungsaufbau zur Stabilisierung der Spannung bietet. Aufgrund der Fehler, die durch Spannungsabweichungen beim Übertragen, sowie Senden der Daten auftreten können, können Fehler vorgebeugt werden. Ein Längsregler ist ein Spannungsregler, der überschüssige Energie in Form von Wärme abführt, um die Ausgangsspannung zu stabilisieren. Im Gegensatz zu einem Querregler kann die Ausgangsspannung mittels Dimensionierung von Widerständen nicht geändert werden, da diese vom Hersteller in der Entwicklung festgelegt wurden.

Ein Bild, das Text, Diagramm, Reihe, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 3,3V-Längsreglerschaltung

Um Spannungsschwankungen und Rauschauftreten zu verhindert, wird im Datenblatt empfohlen zwei gleichgroße Kondensatoren am Eingang und Ausgang parallel zu schalten. Durch den Kondensator C1 wird sichergestellt, dass der Reger in einem stabilen Betriebszustand bleibt. Der Ausgangskondensator C2 dient dazu, die Ausgangsspannung zu stabilisieren und Spannungsspitzen zu minimieren, damit die Ausgangsspannung auch bei sich ändernden Lastbedingungen konstant bleibt.

### Pierce-Oszillator

Da für die Inbetriebnahme des Mikrocontrollers ein Taktsignal benötigt wird, ist eine Oszillatorschaltung notwendig. Mit den Informationen aus dem Datenblatt wird ein Quarzoszillator dazu gebaut, der den ATMEGA2560 mit einem Rechtecksignal mit 16MHz versorgt.

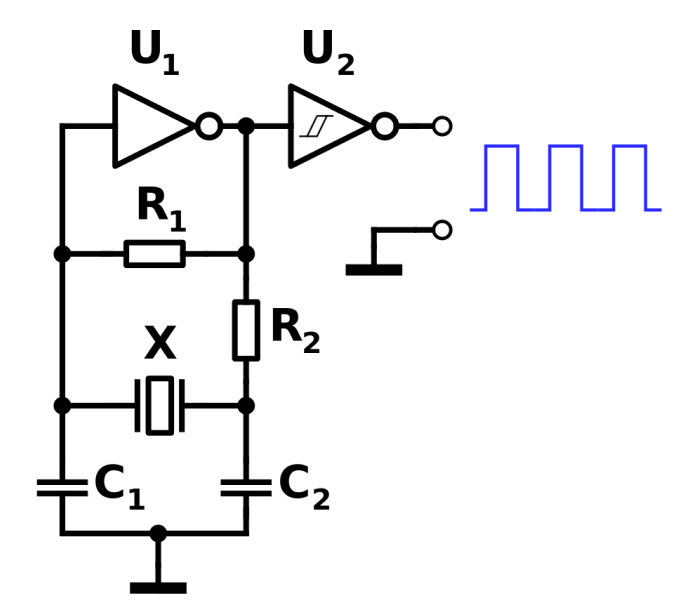


Abbildung Schaltplan Pierce-Oszillator (<https://de.wikipedia.org/wiki/Quarzoszillator>)

Die Bauteile sind essentiell, um die Erzeugung und Stabilisierung der Taktfrequenz zu sichern. Der Quarz fungiert als frequenzbestimmendes Element, indem er aufgrund seiner piezoelektrischen Eigenschaften Schwingungen erzeugt, die als präzise Taktfrequenz dienen. Die Kondensatoren sind in der Regel parallel zum Quarz geschaltet. Sie dienen dazu, die nötige Kapazität für die Schwingungsbildung des Quarzes bereitzustellen. Die Kondensatorwerte von 22pF wurden gemäß den Angaben im Datenblatt ausgewählt, das die Gleichheit der Kapazitäten für beide vorschrieb. In einigen Schaltungen werden Widerstände parallel zu den Kondensatoren geschaltet, um die Impedanz des Oszillators zu beeinflussen. Die Widerstände können dazu beitragen, die Schwingungsfrequenz zu stabilisieren und unerwünschte Schwingungsmoden zu unterdrücken. Der Inverter U1 erzeugt eine um 180° phasenverschobene Ausgangsspannung, die für die positive Rückkopplung und Stabilisierung der Schwingungen im Oszillatorkreis, insbesondere bei Verwendung eines Quarzresonators, entscheidend ist. Der Schmitt-Trigger dient dazu, das Ausgangsignal des Oszillators zu stabilisieren und sicherzustellen, dass das Signal sich im Pegelbereich des Mikrocontroller befindet. Beim ATMEGA 2560 ist der Großteil der Oszillatorschaltung im IC implementiert, daher sieht die Beschaltung wie folgt aus:

Ein Bild, das Text, Schrift, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung

Abbildung Oszillatorschaltung beim ATMEGA2560

Da der Mikrocontroller mit einer Taktfrequenz von 16MHz arbeitet, wird ein Quarz mit diesem Wert eingebaut. Die Pins XTAL1 und XTAL2 stellen hierbei die Anschlüsse des im IC eingebauten Oszillator da. Die Frequenz kann somit mit dem Wert des Quarzes gewählt werden.

## Roboter Revision B

### Gyroskop MPU6050

Der MPU6050 – Sensor verfügt über ein 3- Achsen Accelerometer, ein 3- Achsen Gyroskop und einen Temperatursensor. Der MPU6050 besitzt intern einen eigenen DMP (Digital Motion Processor). Der DMP wurde ausgelegt für Sensorfusion und Bewegungserkennung. Der Prozessor kombiniert die Daten des Accelerometer und des Gyroskop, um Fehler innerhalb der einzelnen Sensoren zu minimieren. Er kann die Daten außerdem verrechnen und das Ergebnis in Euler-Winkel umrechnen. Der Algorithmus, auf welche Weise die Daten kombiniert werden, wird vom Hersteller Invensense nicht offen gelegt. Die Daten werden über den I2C – Bus eingelesen. Im Sensor befinden sich MEMS (= „Micro-Electrical-Mechanical Systems“). Das sind winzige elektro-mechanische Systeme die billig und in Masse produziert werden können.

Ein Bild, das Elektronik, Schaltung, Elektronisches Bauteil, passives Bauelement enthält.

Automatisch generierte Beschreibung

Abbildung Gyroskop MPU6050

Im rechten Bild von Abb. 9 drückt die Beschleunigung nach links die mittlere Kondensatorplatte näher zur rechten. Das verändert die Kapazität zwischen den Platten. Misst man die Kapazitäten kann man so auf die Beschleunigung rückschließen.

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

Abbildung Funktionsweise des Accelerometers (<https://www.elektronik-kompendium.de/sites/bau/1503041.htm>)

Der Accelerometer misst Beschleunigungen entlang der x, y, z-Achsen, während das Gyroskop Geschwindigkeiten um die x,y,z – Achsen misst.

## Hardware-Design Controller

Die Steuerung des Roboters erfolgt über einen eigens dazu entwickelten Controller. Der Controller verfügt über zwei Joysticks, mit dem der Roboter hinsichtlich der Bewegung gesteuert werden kann. Zusätzlich……(welche weiteren Funktionen ?)

Die Daten, die von den Bedienelementen des Controllers durch den uC erfasst werden, werden mittels des Funkmoduls NRF24L01+ zum Receiver, welcher hierbei der Roboter ist, übertragen.

xxxxxBlockschaltbildxxxxxx

Als Mikrocontroller wird der ATMEGA328p genutzt, welcher für die Nutzung des Controllers ausreichend ist. Das Taktsignal wird durch einen externen Quarzoszillator mit 16MHz generiert, der die Geschwindigkeit der Programmabläufe beeinflusst.

Die Platine wird mit 4,5V betrieben, die durch drei 1,5V AAA-Batterien zustande kommen.

Die Spannungsversorgung wird mit einem Schalter getrennt, der ein Ausschalten der kompletten Hardware ermöglicht. Des Weiteren wird eine 3.3V Spannungsquelle benötigt, um das Funkmodul NRF24L01+ in Betrieb zu nehmen. Da der Controller als Transmitter fungiert ist kein Logik-Pegel-Konverter notwendig und kann das Funkmodul direkt mit den Pins des SPI-Bus verbunden werden. Der ATMEGA328p wird mit einem Sockel in die Schaltung integriert und wird für die Programmierung in einem Arduino UNO Evaluierungsboard eingesteckt. Daher wird die Schaltung nicht über den SPI-Bus programmiert und kein ICSP-Header wird benötigt.

Um die Fahrrichtung des Roboters ändern zu können, wird ein Joystick eingebaut, welcher einen Widerstandswert besitzt und von den Analog-Digital-Converter des Mikrocontrollers zu einem Wert bis 1024 konvertiert, da dieser eine Auflösung von 10 Bit besitzt. Des Weiteren besitzt der Joystick einen eingebauten Pushbutton, mit welchen XXXnoch fehlende FunktionXXX.

xxxxxTriggerfunktionxxxx

### xxxTriggerxxx

### Joystick

Die Fahrtrichtung des Roboters wird mittels eines 2-Achsen-Joysticks bestimmt, der sich in allen Richtungen bewegen lässt. Dieser besitzt jeweils einen Potentiometer für die X- und Y-Achse, welche als veränderbare Spannungsteiler fungieren.

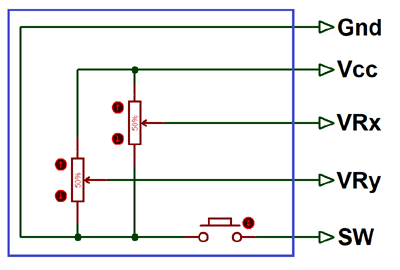


Abbildung Joystick Schaltplan

Um die Daten des Spannungsteilers auslesen zu können, müssen die Pins VRx und VRy an den ADC-Pins des Mikrocontrollers verbunden sein. Dafür muss der Joystick zunächst mit 5V versorgt werden, da ansonsten unerwünschte Fehler beim Konvertieren auftreten könnten. Des Weiteren bietet der Joystick einen eingebauten Taster an, der eine logische „0“ oder „1“ ausgibt, daher wird der Pin SW an einem digitalen Pin angeschlossen.

< BILD : Verbindung Arduino ⬄ Joystick >

# Softwareentwicklung

In diesem Kapitel wird die Softwareentwicklung, für dieses Projekt, erklärt. Das reicht von: „Wie laden wir den Code hoch?“, „Wie dokumentieren wir Änderungen an den Codes?“,   
„Wie benennen wir Variablen und Funktionen“ bis hin zu „Wie funktioniert der Code, was soll er tun?“

Alle Codes die in diesem Kapitel verwendet werden, sind an C angelehnte „Pseudocodes“. Diese sind nicht in einer Programmiersprache verfasst und sollen nur die Funktionsweise des tatsächlichen Codes erklären. Dabei können gewisse Funktionsaufrufe, Variablendeklarationen, Typecasting und andere Kleinigkeiten, die der Erklärung der eigentlichen Funktionsweise des Codes im Weg stehen könnten, ausgelassen werden. Teilweise wird auf diese Kleinigkeiten in Fußnoten eingegangen. Im Anhang ist allerdings eine gut kommentierte Version der tatsächlich verwendeten Codes zu finden.

## Konzept, Programmierung und Debugging

### Konzept

Grundsätzlich beinhaltet das Projekt zwei separate Codes die alle mit RF24 – Funkmodulen miteinander kommunizieren. Der Controller, welcher elektrische Komponente mithilfe von ADCs einliest und diese Daten an den Roboter weiterleitet. Dieser verwendet die Daten und entscheidet anhand dieser, wie die Motoren angesteuert werden sollen.

### Programmierung und Debugging

In diesem Kapitel wird der Style Guide definiert und die Weise, wie wir den Code hochladen und Textnachrichten vom µC bekommen erklärt. Der Roboter und Controller verwenden jeweils den atMega2560 und atMega328p µC. Das sind beides µC die für verschiedene Arduino Boards verwendet werden, weshalb für die Softwareentwicklung die gewöhnliche Arduino Umgebung verwendet werden kann. Ein gewöhnliches Arduino-Board wird mit einem USB – Kabel mit dem Computer verbunden. Ein zweiter µC ermöglicht „Serial over USB“ also eine (virtuelle) serielle Verbindung über die USB – Schnittstelle. Auf dem Arduino läuft ein Bootloader, ein kleines Programm welches beim Hochfahren des Arduinos startet, und das Programmieren über die Serielle Verbindung ermöglicht. Über diese können auch Debug – Nachrichten an den Computer übertragen werden. Wir haben allerdings weder den zweiten µC noch den Bootloader, da wir die µC direkt vom Händler kaufen. Deshalb mussten für Programmierung und Debugging andere Methoden verwendet werden, die folgend näher beschrieben werden.

### In-System-Programming über SPI-Bus

Um Funktionalitäten von Arduino-Boards verwenden zu können schalten wir für Programmierung und Debugging einen zweiten Arduino dazwischen, wie folgt:

A picture containing text, screenshot, font, software

Description automatically generated

Abbildung : Blockschaltbild In-System-Programming

Programmiert kann der µC über den SPI – Bus werden. Für diesen Zweck haben Arduinos einen veränderten SPI - Bus am ICSP[[1]](#footnote-2) – Header ausgeführt:

Ein Bild, das Text, Schrift, Design enthält.

Automatisch generierte Beschreibung

Abbildung : ICSP – Header der Arduinos

Verändert, weil statt dem üblichen „Slave Select“, der „Reset“ Pin ausgeführt ist. Auf unserem PCB wird der gleiche Header implementiert, für Testzwecke Programmieren wir derweilen den Arduino Mega2560, auf dem sich der atmega2560 befindet.

Das Programmieren erfolgt wie folgt:

1. Zunächst muss das Programmierer-Board selbst den Code „Programmer.ino“ hochgeladen bekommen.
2. Danach kann man die von uns erstellten ICSP – Stecker miteinander verbinden. Der Stecker mit zusätzlichem Jumper-Draht wird auf die ICSP - Schnittstelle des Programmierer-Boards angesteckt, das zusätzliche Kabel auf Pin 10. Dieser ist mit Pin 5: „reset“ verbunden. Die andere Seite wird auf das ICSP – Interface des zu programmierenden Boards angesteckt. Die schwarz angemalte Seite zeigt nach oben, ist also näher an Pin 1 der Schnittstelle.
3. Nun kann das Programm hochgeladen werden. Als Board wird „Arduino Mega2560“ ausgewählt, als Programmer „ArduinoISP“ und „Upload with Programmer“ betätigt.
   * 1. Debugging over Serial

Für das Debugging müssen einfach nur die seriellen Schnittstellen von µC und Arduino verbunden werden. Der Computer bekommt dann die Nachrichten des µC, als würden sie vom Arduino kommen.

### Versionsverwaltung und Style – Guide

Jedes Programm bekommt einen Header, in folgender Form:

/\*\*

\* Version: Projektversionsnummer.Programmversionsnummer

\* Author: Author

\*

\* Created: Datum der Erstellung

\* Last modified: Datum der letzten Änderung

\*

\* Desc: Kurzbeschreibung des Programms

\*/

Die Projekt- und Programmversionsnummern sind laufende, bei 0 beginnende Kennzahlen. Ändert man ein Programm, wird die Programmversionsnummer um eins erhöht. Verlangt eine Änderung auch Anpassung der anderen Programme, wird stadtessen die Programmversionsnummer rückgesetzt und die Projektversionsnummer um eins erhöht.

Verschiedene Codes sind also nur dann kompatibel, wenn die Projektversionsnummer übereinstimmt. Zusätzlich wird ein Dokument geführt, welches alle Veränderung einer neuen Version beschreibt.

### Variablen- und Funktionsnamen

Alle Variablen und Funktionen sind Englisch[[2]](#footnote-3) und im „Camel case“ benannt. Statt einem Abstand wird also das nächste Wort großgeschrieben (z.B. doDebug).

Alle Pinnummern beginnen mit p\_. Einige Wörter werden so oft verwendet, dass sie mit einzelnen Buchstaben abgekürzt werden dürfen:

l … left, r … right

b … backwards, f … forwards

Die Reihenfolge ist Left / Right dann Backwards / Forwards. Der Pin, mit dem der Rechte Motor nach vorne angesteuert werden kann heißt somit:

p\_rf

In den Pseudo-Codes der Diplomarbeit sind diese Wörter der Klarheit halber oft ausgeschrieben.

## Software-Design Controller

In diesem Kapital wird der Code des Controllers erklärt. Dieser soll die elektronische Komponente auslesen, die Daten kurz verarbeiten und anpassen, um diese dan an den Roboter weiterzusenden. Um den Status zu erkennen, soll eine Status-LED angesteuert werden.

### Ansteuerung der Status-LED

Die ersten Zeilen im Code des µC schalten die LED ein. Das zeigt 1) das der richtige Code hochgeladen ist und 2) das der µC mit Strom versorgt ist. Danach wird das Funkmodul gestartet und es wird versucht, ein Paket an den Roboter zu senden. Wird diese angenommen, blinkt die LED kurz auf und leuchtet dann wieder konstant. Somit kann auf einem Blick erkannt werden, ob eine Verbindung hergestellt wurde.

### Auslesen der Elektronik

Der Controller muss die verschiedenen elektrischen Komponenten, also den Joystick und die Schultertasten auslesen und an den Roboter weiterschicken. Unabhängig von der Hardware wurden die Wertebereiche in Kapitel 8.4.2 definiert. Die Schultertasten entsprechen diesem Standard bereits, der Wertebereich des Joysticks muss jedoch darauf angepasst werden. (XXX richtiger code in lukas laptop)

array[0] = analogRead(p\_joyX)

array[1] = analogRead(p\_joyY)

array[2] = analogRead(p\_lSchoulder)

array[3] = analogRead(p\_rShoulder)

radio.write(array)

“radio.write()” verwendet das Funkmodul, um diese Daten an den Roboter zu senden. Die Ansteuerung des Funkmoduls wird im Kapitel 8.4.1 genauer beschrieben.

## Software-Design Roboter

In diesem Kapitel wird der Code des Roboters erklärt. Dieser soll Daten des Controllers empfangen, und mit diesen die Motoren ansteuern. Um den Status zu erkenn soll eine RGB-LED angesteuert werden.

### Ansteuerung der Status-LED

Die ersten Zeilen im Code des µC schalten die LED auf Rot. Das zeigt, dass 1) der richtige Code hochgeladen ist und 2) dass der Controller mit Strom versorgt ist. Danach wird das Funkmodul gestartet und es wird versucht, ein Paket vom Controller einzulesen. Ist das geschehen, blinkt die LED kurz auf und leuchtet dann konstant grün. Somit kann auf einem Blick erkannt werden, ob eine Verbindung hergestellt wurde.

### Berechnung der Motorsteuerung

Als Input – Größen bekommt der Roboter die Position des Joysticks und der zwei Schultertasten. Die Ansteuerung der Motoren wird anhand dieser berechnet. Die Schultertaste bestimmt die Geschwindigkeit der Motoren, das Verhältnis zwischen den Motoren bleibt aber unverändert. Je nachdem ob die rechte oder linke Schultertaste gedrückt wird, fährt der Roboter nach vorne oder hinten. Die Schultertaste, die stärker gedrückt wird, „gewinnt“. Durch die Wrapper – Funktion, auf die später noch eingegangen wird, laufen Motoren durch negative Werte rückwärts. Die linke Schultertaste gibt also dieselbe Magnitude, aber ein negatives Vorzeichen.

Damit kommen wir auf folgenden Code:

if (rSchulter > lSchulter)

speed = rSchulter

else

speed = -lSchulter

lMotor = lMotorTurn \* speed

rMotor = rMotorTurn \* speed

lMotorTurn und rMotorTurn geben das Verhältnis zwischen linkem und rechtem Motor an. Sie sind also Variablen, die von -1 bis 1 gehen und sich aus der x-Achse des Joycons berechnen lassen.

Nimmt man an, das „joyX“ eine floating-point-Zahl ist, die von -1 (ganz links) bis +1 (ganz rechts) geht[[3]](#footnote-4), kommen wir auf folgendes Diagramm:

Ein Bild, das Kreis, Licht enthält.

Automatisch generierte Beschreibung

Abbildung : Diagramm für Veranschaulichung der Berechnung von motorTurn Werten

Ist der Stick auf ganz links positioniert, soll der Roboter sich im Stand nach links drehen. Deshalb soll sich die linke Kette Rückwärts und die rechte vorwärts bewegen. Die Anderen Positionen können mit einer ähnlichen Argumentation verstanden werden. Zwischen diesen Extrempositionen sollen die Variablen kontinuierlich und linear alle Zwischenwerte einnehmen. Mittels map – Befehl[[4]](#footnote-5) werden diese Wertebereiche aneinander angepasst.

if (joyX < 0) {

l\_motor\_turn = map(joyX, -1, 0, -1, 1)

r\_motor\_turn = 1

}

else if (joyX >= 0) {

l\_motor\_turn = 1

r\_motor\_turn = map(joyX, 0, 1, 1, -1)

}

lMotor = lMotorTurn \* speed

rMotor = rMotorTurn \* speed

Steuert dieser Code die Motoren direkt an und falls der Fahrer nicht vorsichtig ist und schnelle Richtungsänderungen vornimmt, kann das die Motorbrücke überlasten (XXX Erklärung im Hardware teil?). Deshalb sollen diese Änderungen nur langsam übernommen werden. Der Controller ändert nur die „\_soll Werte“, tatsächlich an das Motorshield weitergegeben werden die „\_ist Werte“. Sind ein Soll-Wert und Ist-Wert nicht gleich, wird der Ist-Wert nur einen Schritt Richtung Soll-Wert verändert.

lMotor\_soll = lMotorTurn \* speed

if (lMotor\_ist > lMotor\_soll)

lMotor\_ist -= 1

else if (lMotor\_ist < lMotor\_soll)

lMotor\_ist += 1

Dieser Code gilt genauso für den rechten Motor.

### Ansteuerung des Motorshields

Wie die Motorbrücke funktioniert wurde im Kapitel 7.3.2 bereits genauer beschrieben, wird hier aber noch einmal zusammengefasst. Er hat für beide Motoren zwei Pins die ein PWM – Signal akzeptieren; eins, falls sich der Motor nach vorne drehen soll, eins falls sich der Motor rückwärts drehen soll. Diese Ansteuerung wurde, wie oben bereits erwähnt, durch eine Wrapper – Funktion „versteckt“ werden. Diese könnte auf unterschiedliche Weise designed werden, es wurde aber beschlossen, dass sie für rechten und linken Motor Werte von -255 bis 255 akzeptieren soll. Negative Werte stehen hier für einen sich rückwärts-drehenden Motor. Wie man im vorherigen Kapitel erkennen konnte, macht das die Mathematik ein wenig bequemer.

In der Funktion muss dann nur abgeprüft werden, ob der Wert negativ oder positiv ist, und je nachdem auf den Vorwärts- oder Rückwärtspin der absolute Wert[[5]](#footnote-6) des Arguments als PWM – Signal angelegt werden.

Wie im Kapitel XXX zu sehen ist der linke Motor des Roboters verkehrt herum eingebaut. Das kompensieren wir Softwaretechnisch, indem wir den linken Wert negieren und somit seine Fahrtrichtung umkehren.

drive(left, right) {

left = -left

if (left > 0) {

analogWrite(left\_forward, |left|)

analogWrite(left\_backward, 0)

}

else {

analogWrite(left\_forward, 0)

analogWrite(left\_backward, |left|)

}

}

Auch dieser Code gilt, abseits der Negation des Wertes, genauso für den rechten Motor.

### Gyroskop – Code

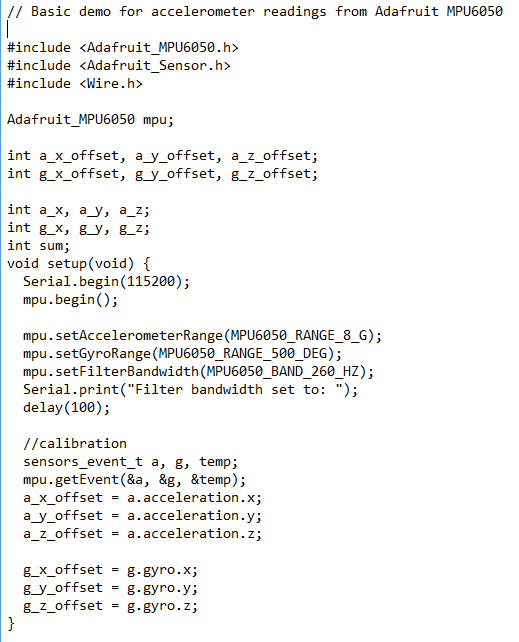


Abbildung Setup- Code für das Gyroskop

Am Anfang werden Wertebereiche des MPUs definiert. Der Accelerometer misst jetzt bis zu ± 8G also bis zu 78.4 m/s². Das Gyro ±500 °/s. setFilterBandwidth() setzt die Grenzfrequenz eines digitalen Filters, der Teil des Moduls ist. Ist die Grenzfrequenz niedriger wird über mehr Messdaten integriert. Das macht den Sensor langsam aber weniger „shaky“. Des Weiteren wurde das Modul zunächst kalibriert. Es wurde eine Messung im Ruhezustand getätigt und diese wurden von folgenden Messungen abgezogen. Das lässt Gravitation und Potentielle Sensor-Drifts wegfallen. Dann wurden die absoluten Werte aller Sensoren addiert und ausgegeben.

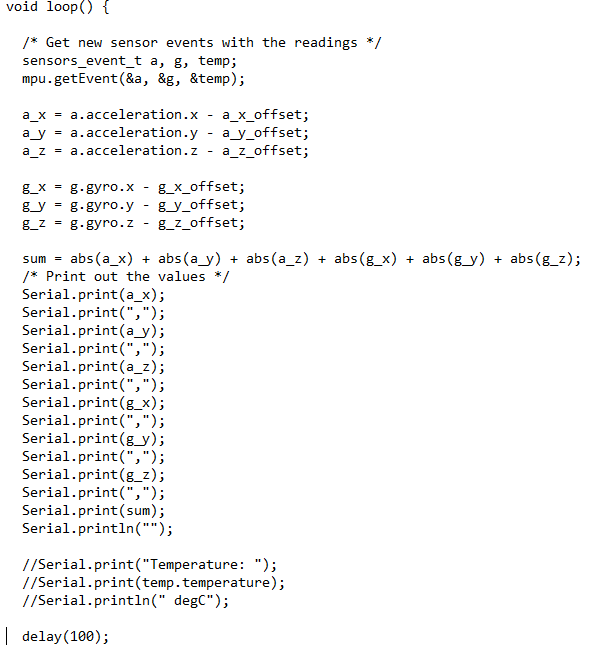


Abbildung Main-Loop für Gyroskop

Im Zuge der Recherche fiel jedoch auf, dass hier lineare Beschleunigungen einfach mit Winkelgeschwindigkeiten addiert werden, was nur bedingt Sinn macht. Für einen sehr groben Richtwert genügt es jedoch.

### Einstellen des Watchdog-Timers

Der Watchdogtimer soll eingerichtet werden, um einen periodischen Interrupt alle 150ms zu erzeugen. In der dazugehoerigen Interrupt Service Routine werden Sensoren eingelesen und die eingelesenen Daten an den Server weitergeleitet.

### Debug - Mode

Um das Projekt debuggen zu können wurde ein Debug – Mode implementiert. Der Roboter bekommt die Daten des Controllers, steuert aber nicht die Motoren an, sondern gibt die vom Controller bekommenen Daten und die sich ausgerechneten Daten an den Serial Monitor aus. Dieser besteht also nur aus einer if – Anweisung und einer Reihe an „Serial.print“ befehlen vor dem Wert wird der Name des Wertes und sein Wertebereich angegeben.

if (doDebug) {

Serial.print("X: ")

Serial.println(inputs[0])

Serial.print("Y: ")

Serial.println(inputs[1])

…

}

## Networking

Es gibt drei verschiedene Komponente, die über das im Kapitel XXX beschriebene Funkmodel miteinander kommunizieren. Das daraus resultierende Netzwerk kann durch folgendes Diagramm veranschaulicht werden:

Ein Bild, das Diagramm, Screenshot, Reihe, Text enthält.

Automatisch generierte Beschreibung

Abbildung : Blockschaltbild der Netzwerkarchitektur

Der Controller schickt dem Roboter die (verarbeiteten) Zustände des Joysticks und der Schultertasten. Der Roboter schickt dem Server im 250ms Takt Daten seiner verschiedenen Sensoren.

### 

### Initialisierung des Moduls

Software-seitig verwenden wir für die Ansteuerung des Funkmoduls die dazugehörige Library “nRF24L01.h”. Einige der anfänglichen Befehle zur Initialisierung des Moduls bleiben bei allen Komponenten gleich.

RF24 radio(p\_CE, p\_CSN);

radio.begin();

Der Sender, startet eine „Pipe“ also eine Verbindung mit einem anderen Modul mit “radio.openWritingPipe(address)”, der Empfänger mit “radio.openReadingPipe(n, address)“. Jedes Modul kann eine Writing-pipe und 6 Reading-pipes offen haben. Es kann also einem Modul schreiben, aber Daten von 6 anderen Modulen empfangen. Jede Pipe muss einer Unterschiedlichen Adresse zugeteilt werden. Um zwischen Schreib-Modus und Lese-Modus zu wechseln gibt es die Befehle “radio.stopListening()” und “radio.startListening()”[[6]](#footnote-7).

radio.openReadingPipe(controllerAddress)

radio.startListening()

radio.read(inputData)

if ( 250ms passed )[[7]](#footnote-8) {

radio.stopListening()

radio.openWritingPipe(serverAddress)

radio.write(sensorData)

radio.startListening()

}

Für die Verbindung zwischen Funkmodul und Roboter verwenden wir das nRF24L01+ Modul. Die elektrischen Charakteristiken sind im Kapitel XXX beschrieben, hier wird auf die Software-seitige Ansteuerung eingegangen. Das Modul wird angesteuert über die nRF24L01.h library.

(XXX Quelle: <https://nrf24.github.io/RF24/classRF24.html#a025fcbad6f062d18252485c1d6ba574f>)

### Standard – Datenpakete

Zwischen Controller und Roboter werden Packete in folgender Form gesendet:

[lSchulter, rSchulter, joyX, joyY]

lSchulter 0 (losgelassen) … 255 (vollstaendig gedrueckt)

rSchulter 0 (losgelassen) … 255 (vollstaendig gedrueckt)

joyX -255 (ganz links) … 255 (ganz rechts)

joyY -255 (ganz unten) … 255 (ganz oben)

Zwischen Roboter und Server werden Pakete in folgender Form gesendet:

[sensor0, sensor1 , …]

(XXX wie viele und was für Sensoren wir haben noch nicht klar)

# Projektmanagement

xxx

# Bedienungsanleitung

# Finale Ergebnisse

# Tabellen und Abbildungsverzeichnis

[Abbildung 1: Systemarchitektur des Projekts 8](#_Toc153120591)

[Abbildung 2: Objektstrukturplan (OSP) / Arbeitspakete 10](#_Toc153120592)

[Abbildung 3:Projektstrukturplan (PSP) AP1-AP6 11](#_Toc153120593)

[Abbildung 4 Entwicklungsablauf der Roboter Hardware 13](#_Toc153120594)

[Abbildung 5 Blockschaltbild von Roboter Rev A 14](#_Toc153120595)

[Abbildung 6 Roboter Rev A V1.00 PCB 15](#_Toc153120596)

[Abbildung 7 Motorshield-Blockschaltbild 16](#_Toc153120597)

[Abbildung 8 H-Brücke Schaltplan ( https://de.wikipedia.org/wiki/Vierquadrantensteller) 17](#_Toc153120598)

[Abbildung 9 NRF24L01 Messung der Spannungsversorgung 18](#_Toc153120599)

[Abbildung 10 NRF24L01 Messung der Spannungsversorgung mit C = 10µF Elko und Logiksignal 18](#_Toc153120600)

[Abbildung 11 SPI-Bus (https://www.rc-modellbau-portal.de/index.php?threads/spi-schnittstelle-beim-arduino-co.6994/) 20](#_Toc153120601)

[Abbildung 12 SPI-Bus CLK-Konfiguration (https://edistechlab.com/wp-content/uploads/2020/11/Screenshot-2020-11-22-at-09.37.57.png) 20](#_Toc153120602)

[Abbildung 13 SPI-Bus Kommunikation (https://edistechlab.com/wie-funktioniert-spi/?v=fa868488740a) 21](#_Toc153120603)

[Abbildung 14 3,3V-Längsreglerschaltung 22](#_Toc153120604)

[Abbildung 15 Gyroskop MPU6050 23](#_Toc153120605)

[Abbildung 16 Funktionsweise des Accelerometers (https://www.elektronik-kompendium.de/sites/bau/1503041.htm) 23](#_Toc153120606)

[Abbildung 17 Joystick Schaltplan 25](#_Toc153120607)

[Abbildung 18: Blockschaltbild In-System-Programming 26](#_Toc153120608)

[Abbildung 19: ICSP – Header der Arduinos 27](#_Toc153120609)

[Abbildung 20: Diagramm für Veranschaulichung der Berechnung von motorTurn Werten 30](#_Toc153120610)

[Abbildung 21 Setup- Code für das Gyroskop 32](#_Toc153120611)

[Abbildung 22 Main-Loop für Gyroskop 33](#_Toc153120612)

[Abbildung 23: Blockschaltbild der Netzwerkarchitektur 34](#_Toc153120613)

# Literaturverzeichnis und Quellenangaben

# Anhänge

# 3D-Modelle und Konstruktionszeichnungen

# Schaltpläne und PCB Designs

# Programm-Codes

# Mess & Labor-Protokolle

1. ICSP (= „In Circuit Serial Programming“), ist eine andere Abkürzung für ISP (= „In System Programming“) und meint das Programmieren eines µC, während dieser im System eingebaut ist. [↑](#footnote-ref-2)
2. Die einzigen Ausnahmen sind die \_soll und \_ist Suffixe. [↑](#footnote-ref-3)
3. Tatsächlich ist diese nach Kapitel 8.4.2 eine Integer-Zahl zwischen -255 und 255. Diese wird zuerst auf einen Float gecasted und dann durch 256 dividiert, um auf den Wertebereich [-1; 1] zu kommen. [↑](#footnote-ref-4)
4. Der Map – Befehl von Arduino verwendet nur Integer Mathematik. Deshalb ist im Code ein eigener floating-point-map Befehl implementiert. lMotor und rMotor werden dann wieder zurück auf einen Integer – Wert gecasted, um als PWM – Signal an das Motorshield weitergeleiten werden zu können. [↑](#footnote-ref-5)
5. Im Code wurde der Einfachheit halber, der positive Wert gelassen und der Negative negiert. [↑](#footnote-ref-6)
6. startListening() sollte nach openReadingPipe() ausgeführt werden, stopListening() jedoch vor openWritingPipe() [↑](#footnote-ref-7)
7. Tatsächlich wird das periodische Senden der Daten mit Interrupts des Watchdogs-Timers implementiert und nicht z.B. einer Zeitmessung mit der millis() Funktion, wie dieser Code suggeriert. Auf die Implementierung des Watchdogtimers wird in Kapitel XXX genauer eingegangen. [↑](#footnote-ref-8)