

## Informatik Projekt

# Fernsteuerung eines “Korsel”-Roboters mit Hilfe einer Software für die Android-Plattform

vorgelegt von:

Steffen Pfiffner

am 09.06.2011

---

## **Einleitung**

In dieser Projektarbeit wurde eine Software für das Betriebssystem “Android” entwickelt, die es ermöglicht, einen “Korsel”-Roboter fernzusteuern. Der Accelerometer des Android-Gerätes wird verwendet, um dessen Lage im Raum zu bestimmen. Durch Betätigung einer Taste kann die aktuelle Lage des Geräts als Ausgangslage definiert werden. Von dieser Lage aus wird das “Korsel” durch Vor- und Zurückkippen und Rechts- und Linksdrehen des Gerätes gesteuert. Die ermittelten Accelerometer-Informationen werden in Steuerkommandos für das “Korsel” umgesetzt und per Bluetooth an dieses übertragen.

# **Inhaltsverzeichnis**

<b>1 Die “Korsel” Roboter-Plattform</b>	<b>4</b>
<b>2 Der Accelerometer</b>	<b>5</b>
<b>3 Korsel BT</b>	<b>8</b>
<b>4 Kommunikations Protokoll</b>	<b>9</b>
<b>5 “Korsel embedded” Software</b>	<b>10</b>
<b>6 Korsel Control Software</b>	<b>13</b>
<b>7 Ergebnis</b>	<b>21</b>
<b>8 Erweiterungsmöglichkeiten</b>	<b>21</b>
<b>9 Anhang</b>	<b>23</b>
<b>Literatur</b>	<b>25</b>

## 1 Die “Korsel” Roboter-Plattform

Hier soll eine Übersicht über die für dieses Projekt wichtigen Eigenschaften des Korsels gegeben werden. Auf der Korsel-Website [Fes10] wird der Roboter genauer beschrieben. Außerdem sind alle Bauanleitungen, Platinenlayouts und Programme erhältlich, die zum Nachbau erforderlich sind.

Es existieren momentan mehrere Korsel-Varianten, die sich im Aufbau grundsätzlich folgendermaßen unterscheiden:

- Beim Standard-Korsel befindet sich die Schaltung auf einer Leiterplatte, die an beliebigen Chassis angebracht werden kann. Es gibt z.B. ein Gabel-Korsel, welches eine Gabel als Chassis verwendet.
- Das HD-Korsel besteht aus einer Leiterplatte, die gleichzeitig auch als Chassis funktioniert. Die Platine ist mit SMD Bauteilen bestückt.

Da der in diesem Projekt verwendete Korsel auf dem HD-Korsel basiert, wird nur er hier näher beschrieben. Ein HD-Korsel besteht, wie auf Abbildung 1 zu sehen, aus einer Leiterplatte als Chassis. Zwei Kunststoff-Kugellager, die an einer auf der Platine angebrachten Achse befestigt sind, dienen als Räder. Die Kraftübertragung von den Motoren zu den Kugellagern geschieht mit Haushaltsgummis. Die Motorachsen sind für mehr Traktion mit einer doppelten Schicht Schrumpfschlauch überzogen. Ein 9 Volt Block liefert den Strom. An der Spitze befindet sich auf der Unterseite eine Reflektionslichtschranke, die als Liniensensor dient. Ein Atmel ATtiny 2313 Mikrocontroller übernimmt die Steuerung. Er wird über eine ISP<sup>1</sup> Schnittstelle programmiert.

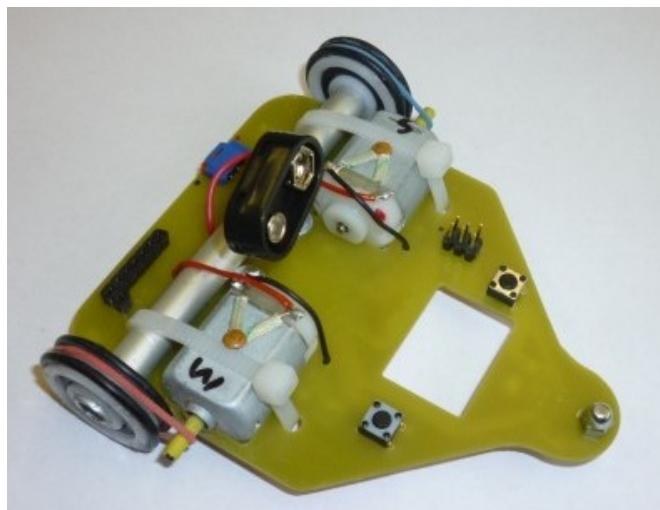


Abbildung 1: Das HD Korsel [Fes10]

---

<sup>1</sup>In-System-Programmierung

Das Korsel kann, entsprechend programmiert, einer schwarzen Linie auf weißem Untergrund folgen. „In der einfachsten Form sind die Korsels Kantenfolger. Das heißt, der Sensor erkennt entweder die schwarze Linie oder den hellen Untergrund. Das Korsel fährt dann entlang dem Übergang von Linie zum Untergrund. Jedoch sind den „Ausbaumöglichkeiten“, sei es an Software oder Hardware, keine Grenzen gesetzt.“ [Fes10]

Die ausgeführte Software ist sehr einfach gehalten:

- Sensor „sieht“ die schwarze Linie: Linker Motor an und rechter Motor aus.
- Sensor „sieht“ hellen Untergrund: Rechter Motor an und linker Motor aus.

Mögliche Motor Zustände sind: Volle Leistung vorwärts und ausgeschaltet. Dies resultiert in einer sehr ruckartigen Fahrweise des Roboters, da er nicht geradeaus fahren kann.

## 2 Der Accelerometer

Dieses Kapitel gibt eine Einführung in die Verwendung des in Android Geräten eingebauten Accelerometers. Der Accelerometer kann bei allen Geräten auf die selbe Weise abgefragt werden. Alle Informationen stammen aus der Android Referenz. ([Goo11]) Abbildung 2 zeigt wie das Koordinatensystem des Accelerometers definiert ist. Es bleibt immer gleich, auch wenn sich die Bildschirmorientierung ändert.



Abbildung 2: **Das Accelerometer Koordinatensystem** Das Massezentrum des Geräts bildet den Ursprung des Koordinatensystems.

In der Luftfahrt werden die Dreh- und Kippbewegungen eines Flugzeuges mit den Begriffe Pitch, Roll und Yaw, wie in Abbildung 3 dargestellt, bezeichnet. In dieser Dokumentation werden die Begriffe "Pitch" und "Roll" folgendermaßen definiert:

- **Pitch** Das Gerät wird um die Y-Achse gedreht, also nach vorn oder hinten gekippt. Dies entspricht später dem Vorwärts- bzw. Rückwärtsfahren.
- **Roll** Das Gerät wird um die Z-Achse gedreht, also nach rechts oder links gedreht. Dies entspricht später dem Rechts- bzw. Linkslenken.



Abbildung 3: Pitch, Roll und Yaw

Der Zugriff auf den Accelerometer geschieht wie folgt:

Die Klasse `SensorManager` erlaubt den Zugriff auf die Sensoren eines Gerätes.  
`mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);`

Der Sensor vom Typ `ACCELEROMETER` wird abgefragt.  
`mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);`

Eine Instanz der Klasse `AccelerometerListener` wird angelegt.  
`mAccListener = new AccelerometerListener(this);`

Der AccelerometerListener wird mit Hilfe des Sensor Managers bei dem Accelerometer registriert. Das dritte Argument gibt an, mit welcher Rate der Accelerometer Events erzeugen soll. In diesem Beispiel wird die niedrigste Rate `SENSOR_DELAY_UI` verwendet, die zur Ermittlung der Bildschirmausrichtung gedacht ist.

```
mSensorManager.registerListener(mAccListener, mAccelerometer, SensorManager.SENSOR_DELAY_UI);
```

Die oben verwendete `AccelerometerListener`-Klasse implementiert das Interface `SensorEventListener`.

In der Funktion `onSensorChanged(SensorEvent event)` werden die Accelerometer-Events verarbeitet. Das Array `event.values` enthält den Beschleunigungsvektor. Dabei gilt:

```
X = event.values[0];
Y = event.values[1];
Z = event.values[2];
```

Diese Werte können nun beliebig weiterverarbeitet werden.

In Abbildung 4 sind für verschiedene Lagen des Gerätes die Werte des Accelerometer-Vektors aufgelistet.

Die Werte werden nach dieser Vorschrift berechnet:

Wert = Beschleunigung in Achsenrichtung ( $0 \frac{m}{s^2}$ ) - Erdbeschleunigung in Achsenrichtung ( $-9,81 \frac{m}{s^2}$ )

Die Beschleunigung in Achsenrichtung ist in diesem Beispiel immer 0, da das Gerät nicht bewegt wird.

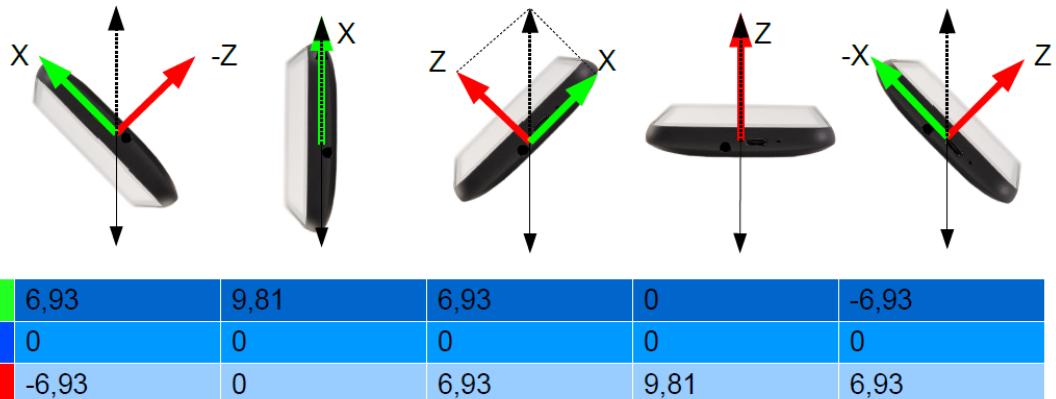


Abbildung 4: Ansicht von unten. Beispiele für Accelerometer-Werte bei einer Drehung um die Y-Achse

## 3 Korsel BT

In meiner Bachelorarbeit habe ich auf Basis des HD-Korsels das Korsel-BT (BT für Bluetooth) entwickelt (siehe Abb. 5). Diese Hardware wird in diesem Projekt ebenfalls verwendet.

In Zusammenarbeit mit Joachim Feßler und Johannes Kessler wurde das Design des HD-Korsels folgendermaßen verändert:

- Der Motortreiber Baustein L293DD wurde hinzugefügt, um einen Rückwärtsbetrieb der Motoren zu ermöglichen und deren Geschwindigkeit per PWM<sup>2</sup> zu regeln.
- Die Pins PD0 und PD1 (USART RX UND TX) des Mikrocontrollers, eine Spannung von 3.3V und Masse sind an einer Pfostenleiste abgreifbar. Dies ermöglicht den Anschluss eines “Parani ESD 1000” Bluetooth Moduls.

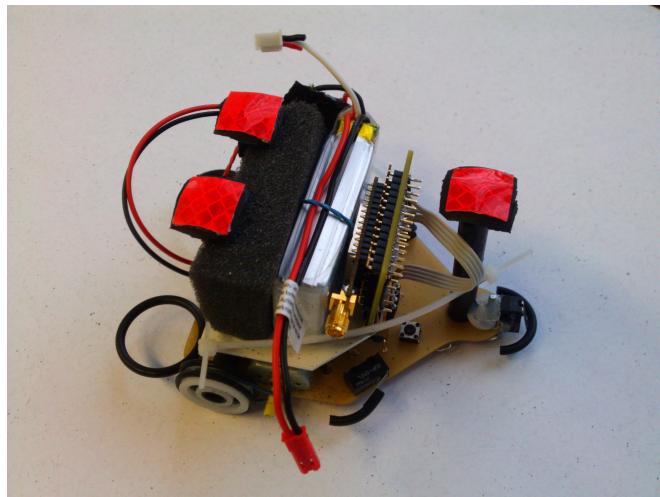


Abbildung 5: Das Korsel BT

### 3.1 Parani ESD 1000 Bluetooth Modul

Das Parani-ESD 1000 Bluetooth Modul ermöglicht eine kabellose serielle Kommunikation über Bluetooth. Nach einer einmaligen Konfiguration kann zu dem Modul eine Bluetooth Verbindung hergestellt werden. Über diese Verbindung werden dann wie über eine serielle Kabelverbindung Daten ausgetauscht.

Das Benutzerhandbuch ist unter [Sen10] erhältlich.

Für dieses Projekt wird das Modul folgendermaßen konfiguriert: (Dieser Vorgang muss nur einmalig mit einem neuen Modul durchgeführt werden.)

---

<sup>2</sup>Pulse Width Modulation

Das Modul wird, wie im Handbuch beschrieben, in das “RS232 interface Development Board” gesteckt und seriell mit einem Computer und mit der Stromversorgung verbunden. Mit dem “Reset” Knopf wird der Auslieferungszustand hergestellt. Am Computer wird mit einem RS232 Terminal Programm, unter Linux z.B. cutecom, ein Verbindung zum Modul aufgebaut. Dazu werden folgende Einstellungen verwendet:

- Baudrate: 9600
- Data bits: 8
- Stop bits: 1
- Parity: No
- Hardware Flow Control: Use

Ist eine Verbindung hergestellt, werden folgende Befehle (“AT-Commands”) an das Modul gesendet:

- **AT+BTMODE,3** bewirkt, dass das Modul auf alle eingehenden Bluetooth Verbindungen hört.
- **AT+UARTCONFIG,9600,N,1,0** belässt alle Einstellungen auf Standard, deaktiviert jedoch Hardware Flow Control. Dieses wird nicht benötigt, da das Modul mit dem Korsel später nur über die RX und TX Leitungen verbunden ist.
- **ATZ** startet das Modul neu.

Die Konfiguration ist somit abgeschlossen und kann mit den Befehlen **AT+BTINFO** überprüft werden.

Das Bluetooth Modul kann ab jetzt von jedem Bluetooth Host gefunden und eine Verbindung dazu aufgebaut werden. Der standardmäßig eingestellte Bluetooth Pin lautet “1234”.

## 4 Kommunikations Protokoll

Zur Kommunikation zwischen Roboter und Android Software werden Datenpakete verwendet, die aus zwei mal 8 Bit bestehen (Abb. 6). Das erste Byte gibt den Typ des Datenpakets an und das zweite ist der Inhalt. In der Tabelle sind alle verwendeten Datenpakete aufgelistet.



Abbildung 6: Datenpaket

Typ (HEX-Wert)	Wert	Bedeutung
LEFT MOTOR SPEED FORWARD (0x01)	[0..255]	Linkes Rad mit <Wert> vorwärts
LEFT MOTOR SPEED BACKWARD (0x11)	[0..255]	Linkes Rad mit <Wert> rückwärts
RIGHT MOTOR SPEED FORWARD (0x02)	[0..255]	Rechtes Rad mit <Wert> vorwärts
RIGHT MOTOR SPEED BACKWARD (0x12)	[0..255]	Rechtes Rad mit <Wert> rückwärts
BUTTON PRESSED (0x22)	[1]	Kontakt Sensor wurde ausgelöst
PHOTO SENSOR (0x33)	[0,1]	Liniensensor Zustand ist <Wert> [hell, dunkel]

## 5 “Korsel embedded” Software

Die in C programmierte “Korsel embedded”-Software wird auf dem ATtiny 2313 Mikrocontroller des Korsel ausgeführt und kommuniziert mit der auf dem Android-Gerät ausgeführten “Korsel Control” Software. Der Sourcecode ist unter <http://code.google.com/p/korsel-control/> erhältlich. (Im Ordner KorselEmbedded-Software)

Alle den ATtiny 2313 betreffenden Informationen stammen aus dessen Datenblatt [ATM10].

Das Programm besteht aus einer Initialisierungsphase, der Hauptschleife und zwei Interrupt Service Routinen (ISR) und befindet sich in einer Datei mit dem Namen `Korsel_BT.c`. Mit `#include` ist die Datei `v24_commands.h` eingebunden. Dort werden die in Kapitel 4 “Protokoll” aufgeführten Befehle definiert. Die Headerdateien `avr/io.h`, `avr/interrupt.h` und `util/dely.h` werden eingebunden. Die CPU Taktrate wird als 4 MHz definiert.

### 5.1 Initialisierungsphase

In der Initialisierungsphase werden folgende Funktionen in der dargestellten Reihenfolge aufgerufen:

- `init_io()` initialisiert die verwendeten Pins als Eingänge bzw. Ausgänge im DDR<sup>3</sup> Register und legt den Standard Pegel der Pins fest.
- `pwm()` initialisiert das PWM<sup>4</sup> Signal für die Motoren auf 8 Bit. Dies ermöglicht die Ansteuerung der Motoren mit Werten zwischen 0 und 254. 0 bedeutet Stillstand, 254 volle Geschwindigkeit.
- `USART_Init()` initialisiert den USART<sup>5</sup> des Controllers auf eine Baudrate von 9600, No Parity, 1 Stopbit und eine Paketgröße von 8 Bit. Außerdem wird das Auslösen von Interrupts durch den Empfang von Daten aktiviert.
- `setup_interrupt()` aktiviert Interrupts für den Liniensensor. Jede logische Änderung des Liniensensors erzeugt einen Interrupt.
- `sei()` aktiviert Interrupts global.

## 5.2 Hauptschleife

Die Hauptschleife fragt die Zustände der drei am Korsel angebrachten Berührungstaster ab. Wenn ein Taster betätigt wird, wird die Funktion `Stop_Motor()` aufgerufen. Der oben am Korsel angebrachte Taster führt eine Selbstdiagnose durch. Dabei drehen sich beide Motoren 500ms vorwärts, dann 500ms rückwärts. So wird sichergestellt, dass die Motoren korrekt funktionierten und angesteuert werden.

## 5.3 Liniensensor ISR

Diese ISR wird aufgerufen, wenn sich der Pegel des Liniensensors ändert. Das heißt, wenn die Farbe des erkannten Untergrundes von Weiß auf Schwarz beziehungsweise von Schwarz auf Weiß wechselt. Ein neuer V24Command Vektor wird mit dem Typ `PHOTO_SENSOR` und dem Inhalt 0 bzw. 1, entsprechend dem Pegel des Liniensensors, erstellt. Ist dieser Pegel 1 ist der Untergrund schwarz, d.h. der Sensor befindet sich auf der Linie. 0 bedeutet, der Sensor befindet sich nicht auf der Linie. Der Vektor wird mit dem Aufruf von `USART_Transmit_Command(Vektor)` über den USART an das Bluetooth Modul und von dort an das Smartphone gesendet. Dort könnte diese Information, wie in Kap. 8 beschrieben, verwendet werden.

## 5.4 USART ISR

Diese ISR wird aufgerufen, wenn am USART ein Byte ankommt. Entspricht dieses Byte einem der in Kapitel 4 “Protokoll” definierten Typen, wird als nächstes Byte

---

<sup>3</sup>Data Direction Register

<sup>4</sup>Pulse Width Modulation

<sup>5</sup>Universal Synchronous and Asynchronous serial Receiver and Transmitter

der dazugehörige Inhalt erwartet. Das bedeutet, dass beim nächsten Aufruf dieser USART ISR das Byte als Inhalt des zuvor erkannten Typs interpretiert wird. Wird dem Korsel z.B. folgendes Paket geschickt Typ: LEFT\_MOTOR\_SPEED\_FORWARD

Inhalt: 254, bekommt die USART ISR zunächst das erste Byte “Typ” mit dem Wert LEFT\_MOTOR\_SPEED\_FORWARD (0x01). Die USART ISR wird dann für das zweite Byte erneut aufgerufen. Dieses Byte mit dem Wert 254 wird als die Geschwindigkeit für den Linken Motor vorwärts interpretiert. Entsprechend wird die Motorrichtung gesetzt und der Wert 254 als PWM Wert für die Geschwindigkeitssteuerung des Motors verwendet.

## 5.5 Funktion “Stop Motor”

Die Funktion `Stop_Motor()` wird in der Hauptschleife aufgerufen, wenn ein Beührungssteller betätigt wird. Sie wird zum Stoppen des Korsels und zum Senden des `BUTTON_PRESSED` Befehls verwendet.

Die PWM Werte für den linken und rechten Motor werden auf 0 gesetzt. Ein V24Command Vektor mit dem Inhalt `BUTTON_PRESSED` als Typ und 1 als Wert wird erstellt und mit Hilfe von `USART_Transmit_Command(Vektor)` versendet. Dieses Paket wird von der Android Software noch nicht ausgewertet, könnte jedoch dazu verwendet werden, bei einer Kollision eine Warnung auf dem Display des Smartphones auszugeben.

## 5.6 Funktion “USART Transmit Command”

`USART_Transmit_Command(Vektor)` verwendet die Funktion `USART_Transmit(Byte)`, um die beiden im Vektor enthaltenen Bytes über den USART zu versenden. Die Funktion `USART_Transmit(Byte)` stammt aus dem ATtiny 2313 Datenblatt. Sie wartet bis der Übertragungs-Buffer leer ist, kopiert das übergebene Byte in den Sende-Buffer und wartet bis es erfolgreich gesendet worden ist. Das an den USART angeschlossene Bluetoothmodul leitet die Daten zur verbundenen Bluetooth Gegenstelle, dem Android Smartphone, auf dem die “Korsel Control”-Software ausgeführt wird, weiter.

## 5.7 V24Command Vektor

Der V24Command Vektor ist ein C-struct, bestehend aus zwei Bytes. Das erste Byte “command” enthält, wie in Kapitel 4 definiert, den Typ des Paketes. Das zweite Byte “data” enthält den Inhalt.

## 6 Korsel Control Software

Der Sourcecode des Projekts und eine für Android 2.2 kompilierte Version sind unter <http://code.google.com/p/korsel-control/> erhältlich.

### 6.1 Verwendete Bibliotheken

Folgende Java Bibliotheken werden von der Software verwendet:

- **Java 3d VECMATH** bietet Klassen zur 3D Vektorberechnung. [Ora10]

### 6.2 Kernkomponenten

Die Software besteht aus drei Kernkomponenten:

- Der **KorselControl** Klasse welche die GUI initialisiert und die Komponenten steuert.
- Der **AccelerometerListener** Klasse, welche die vom Accelerometer gelieferten Daten verarbeitet und in Steuerbefehle für den Roboter umsetzt.
- Der **KorselBluetoothSerial** Klasse, welche mit der auf dem Korsel ausgeführten “Korsel embedded”-Software kommuniziert und dessen Steuerung ermöglicht.

Eine Übersicht gibt Abbildung 7.

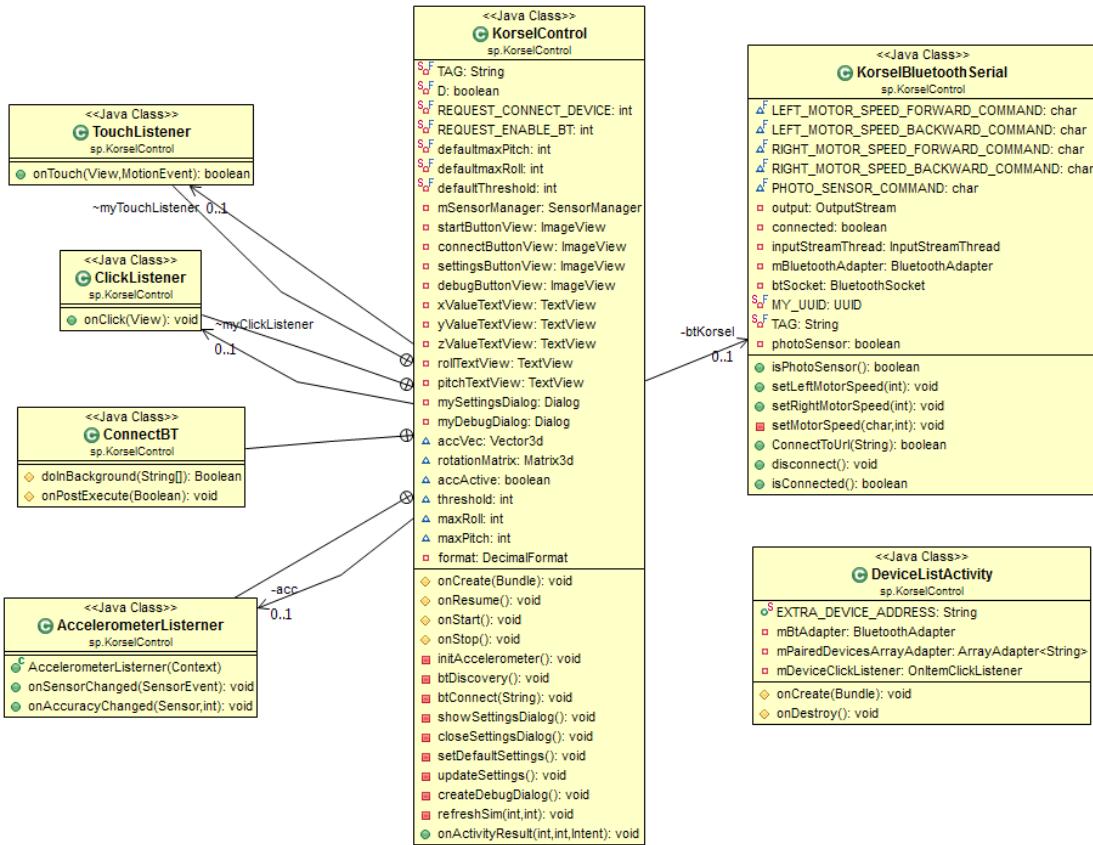


Abbildung 7: Klassendiagramm

### 6.2.1 GUI

Die Benutzeroberfläche ist bewusst einfach gehalten und orientiert sich an einem Armaturenbrett eines Autos. Folgende Aktionen sind in der Startansicht (Abb. 8) möglich:

- **START STOP ENGINE** Solange dieser Button betätigt wird ist die Fernbedienung aktiv. D.h. es werden Steuerbefehle an das Korsel gesendet. Wird er losgelassen, hält das Korsel sofort an. Im Moment des Drückens wird die aktuelle Position des Smartphones als Ausgangspunkt für Steuerbefehle definiert. (siehe Kap. 6.2.2)
- **Connect** Öffnet eine Auswahl von gepairten Bluetoothgeräte zu denen eine Verbindung hergestellt werden kann. (Abb. 9)
- **Settings** Öffnet eine Einstellungs-Ansicht. (Abb. 10)
- **Debug** Aktiviert den Debug Modus. (Abb. 11)

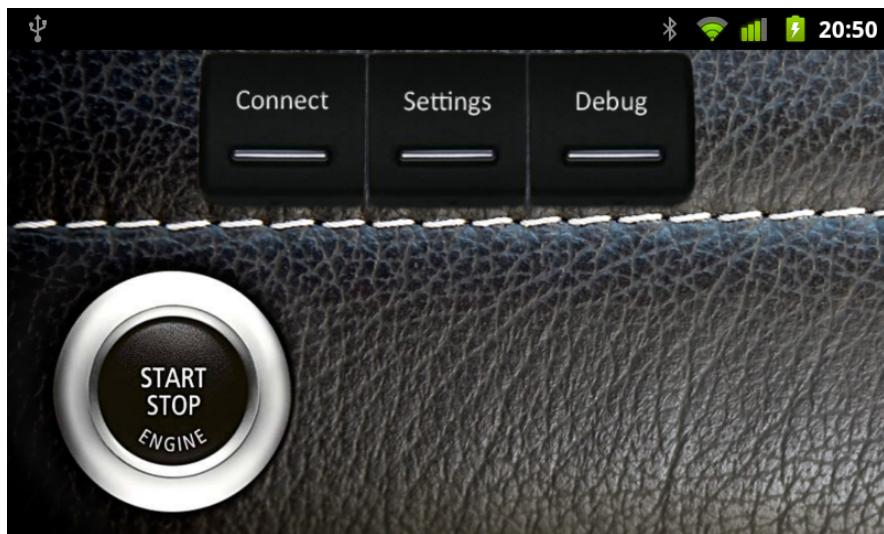


Abbildung 8: Startansicht

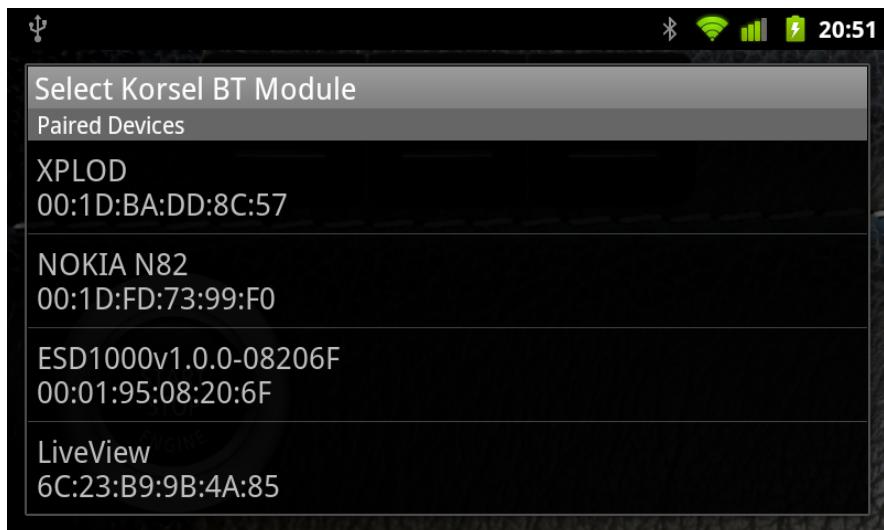


Abbildung 9: Verbindung aufbauen

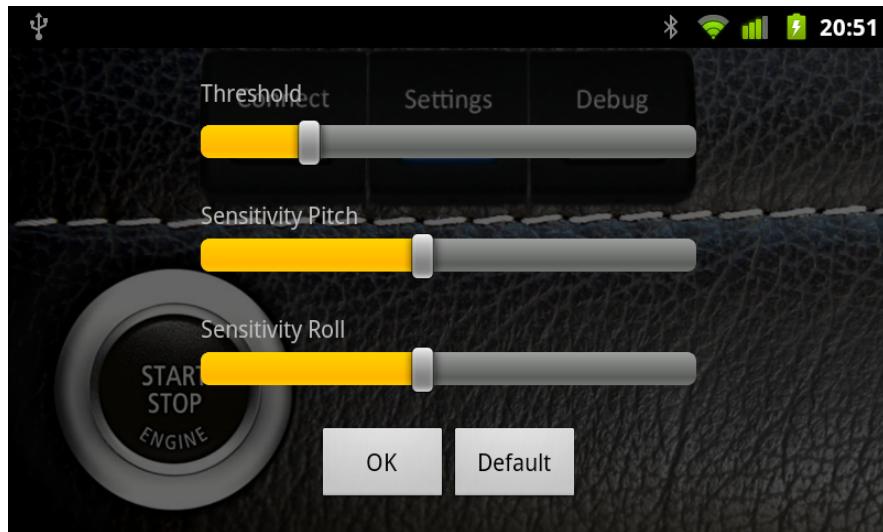


Abbildung 10: **Einstellungen** Hier kann die Empfindlichkeit der Steuerung eingestellt und ein Schwellenwert (“Threshold”) definiert werden. Dabei kann die Empfindlichkeit für “Pitch” und “Roll” getrennt bestimmt werden. (Schieber weiter nach rechts bedeutet höhere Empfindlichkeit.) Der “Threshold” Schieber dient zur Eliminierung des toten Punktes der Motoren. (Schieber weiter nach rechts bedeutet Motoren sprechen schneller an.)

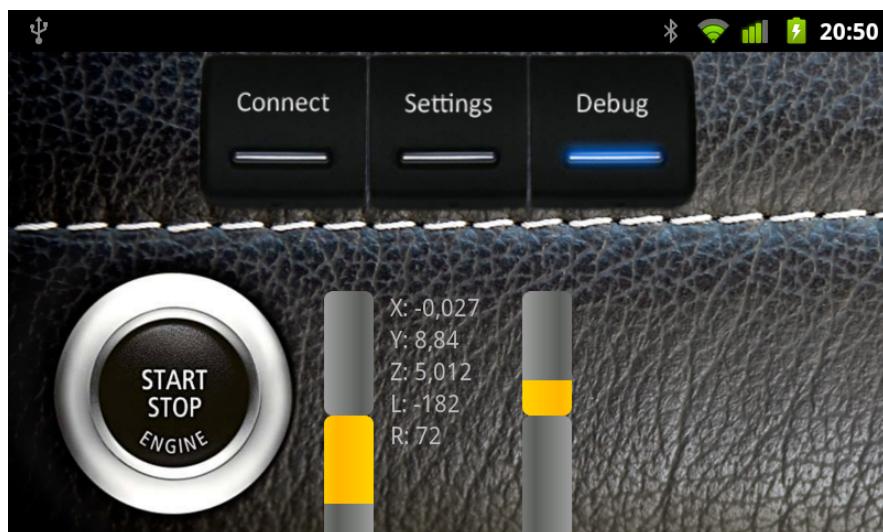


Abbildung 11: **Debug Ansicht** Die Debug Ansicht liefert die aktuellen Accelerometer Werte (X, Y, Z), die ermittelte Geschwindigkeiten der Motoren (L,R) und eine grafische Repräsentation der Motorgeschwindigkeiten.

### 6.2.2 Kalibrierung des Accelerometers

Im Moment der Betätigung des “START-STOP-ENGINE”-Buttons wird die aktuelle Lage des Geräts als Ausgangslage definiert. Von dieser Lage aus kann es nach vorn oder hinten gekippt, oder nach links oder rechts gedreht werden um den Roboter zu steuern. Abbildung 12 zeigt wie die zur Kalibrierung benötigte Rotationsmatrix berechnet wird.

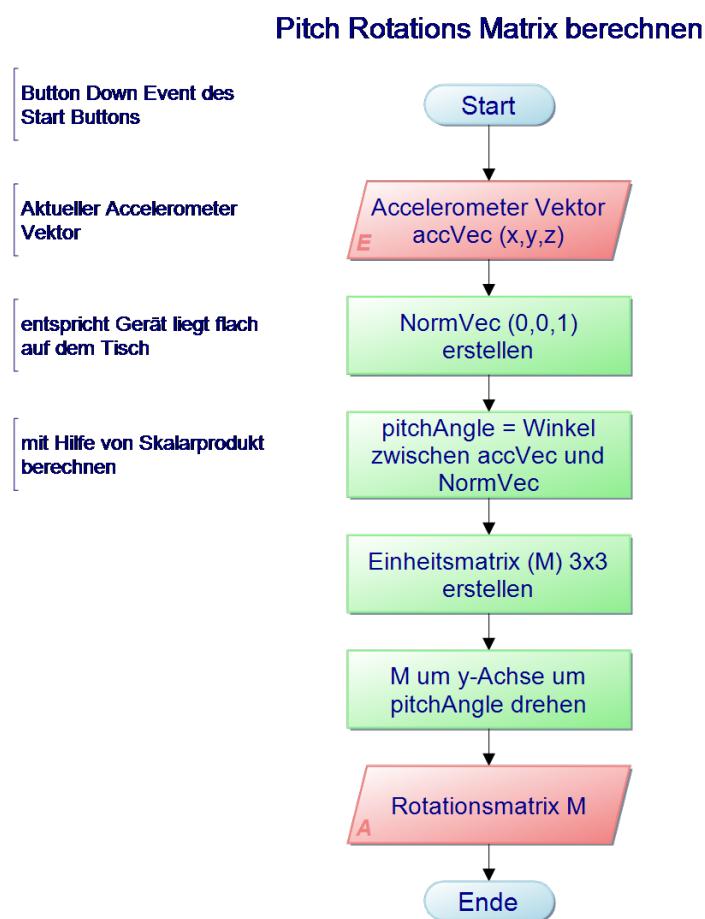


Abbildung 12: Ermittlung der Rotationsmatrix aus einem Accelerometer Vektor.

Abbildung 13 zeigt wie sich das Koordinatensystem durch die Rotationsmatrix verändert. Die berechnete Rotationsmatrix wird, wie im nächsten Kapitel beschrieben, verwendet.

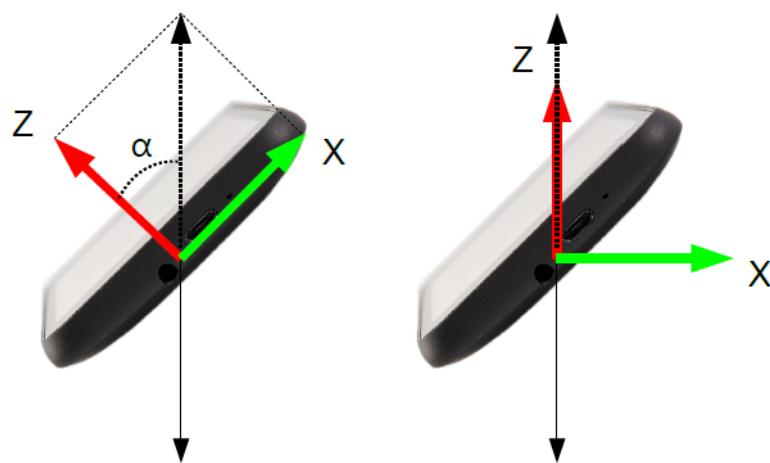


Abbildung 13: Wird ein Vektor aus dem Koordinatensystem (links) mit der ermittelten Rotationsmatrix multipliziert, wird er in ein neues Koordinatensystem (rechts) transformiert.

### 6.2.3 AccelerometerListener-Klasse: Berechnung der Motorgeschwindigkeiten aus den Accelerometer-Daten

Die AccelerometerListener-Klasse verarbeitet `onSensorChanged`-Events die vom Accelerometer ausgelöst werden und berechnet daraus Motorgeschwindigkeiten, welche dann mit Hilfe der **KorselBluetoothSerial**-Klasse an das Korsel übermittelt werden. (siehe Abb. 14)

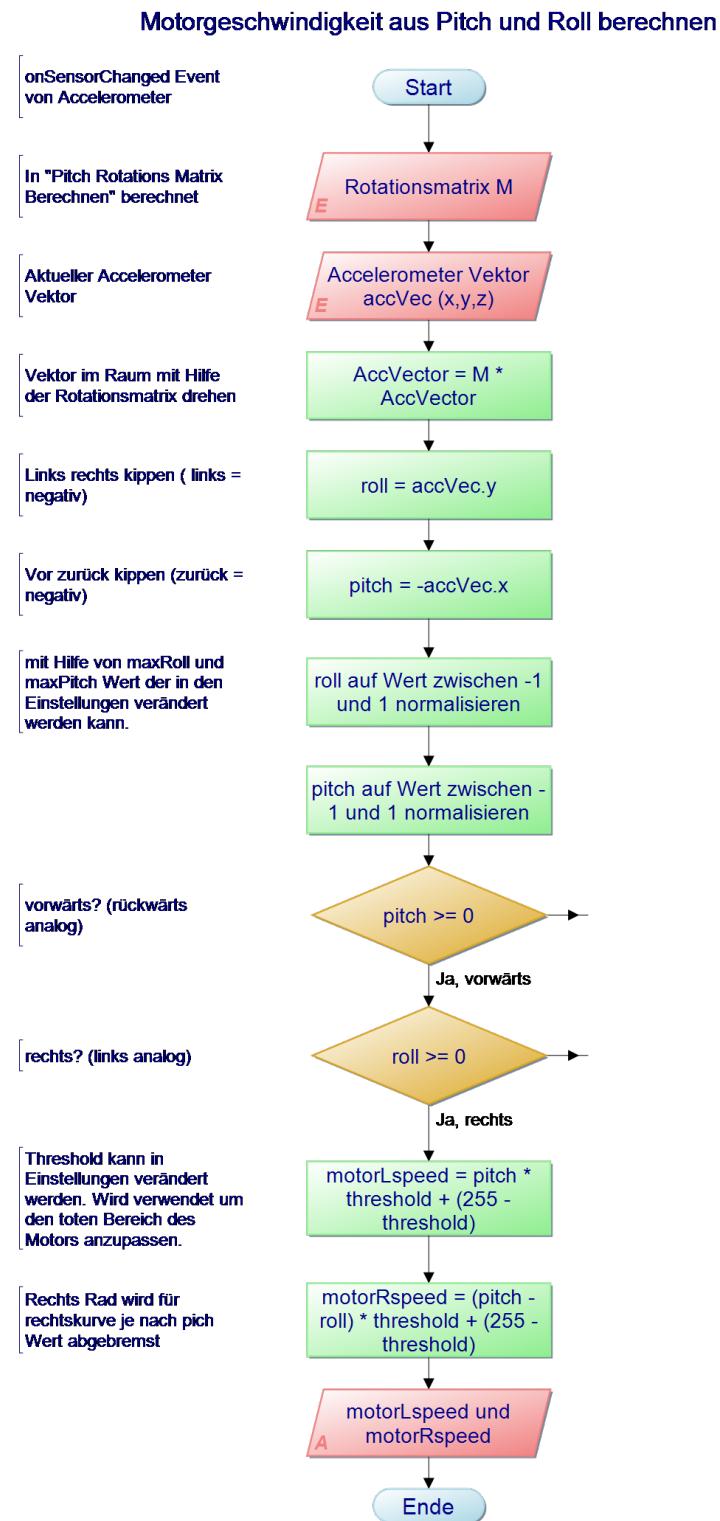


Abbildung 14: Ermittlung der Motorgeschwindigkeiten aus den Accelerometer-Daten.

### 6.2.4 KorselBluetoothSerial-Klasse

Die **KorselBluetoothSerial**-Klasse wird zum Verbindungsauflaufbau und Senden von Befehlen an das Korsel verwendet. `ConnectToUrl(address)` baut eine Verbindung zu dem Gerät mit der übergebenen Adresse auf. Die Funktionen `setLeftMotorSpeed(MotorSpeed)` und `setRightMotorSpeed(MotorSpeed)` senden die übergebenen Werte an das Korsel. Mögliche Werte für "MotorSpeed" sind 0 bis 255. Positive Werte bedeuten vorwärts-, negative Werte rückwärtsfahren. Der übermittelte Wert wird von der "Korsel Embedded"-Software als 8-bit PWM Signal an den Motor gesendet. Wird beispielsweise `setLeftMotorSpeed(-200)` aufgerufen, wird dem Korsel ein Paket mit dem Typ "LEFT MOTOR SPEED BACKWARD" und dem Wert "200" gesendet. `setLeftMotorSpeed(0)` sendet ein Paket vom Typ "LEFT MOTOR SPEED FORWARD" und dem Wert "0".

## **7 Ergebnis**

Die entwickelte Software ermöglicht eine intuitive und genaue Steuerung des Korsel-Roboters. Steuerbefehle werden zuverlässig und ohne spürbare Latenz übermittelt.

Das Projekt kann Informatik- und Elektrotechnikstudenten dazu inspirieren sich interdisziplinär zu beschäftigen.

## **8 Erweiterungsmöglichkeiten**

- Die entwickelte Software kann, mit kleineren Anpassungen, auch für andere Roboterplattformen verwendet werden.
- Die Korsel-Plattform kann hinsichtlich Kosteneffizienz (z.B. günstigeres Bluetooth Modul) und Geschwindigkeit (bessere Kraftübertragung) weiter optimiert werden.
- Die Abfrage des Liniensor-Zustandes (schwarzer Untergrund oder weißer Untergrund) ermöglicht das Einlesen von barcode-ähnlichen Sequenzen auf dem Untergrund. Rundkurs-Rennen mit automatischem Rundenzähler und Strafzeiten beim Verlassen der Strecke werden so möglich.
- Steuerbefehle könnten aufgezeichnet und wiedergegeben werden. Markos könnten zur Steuerung verwendet werden. (z.B. Fahre im Kreis!)



9 Anhang

## 9.1 Korsel Schaltplan

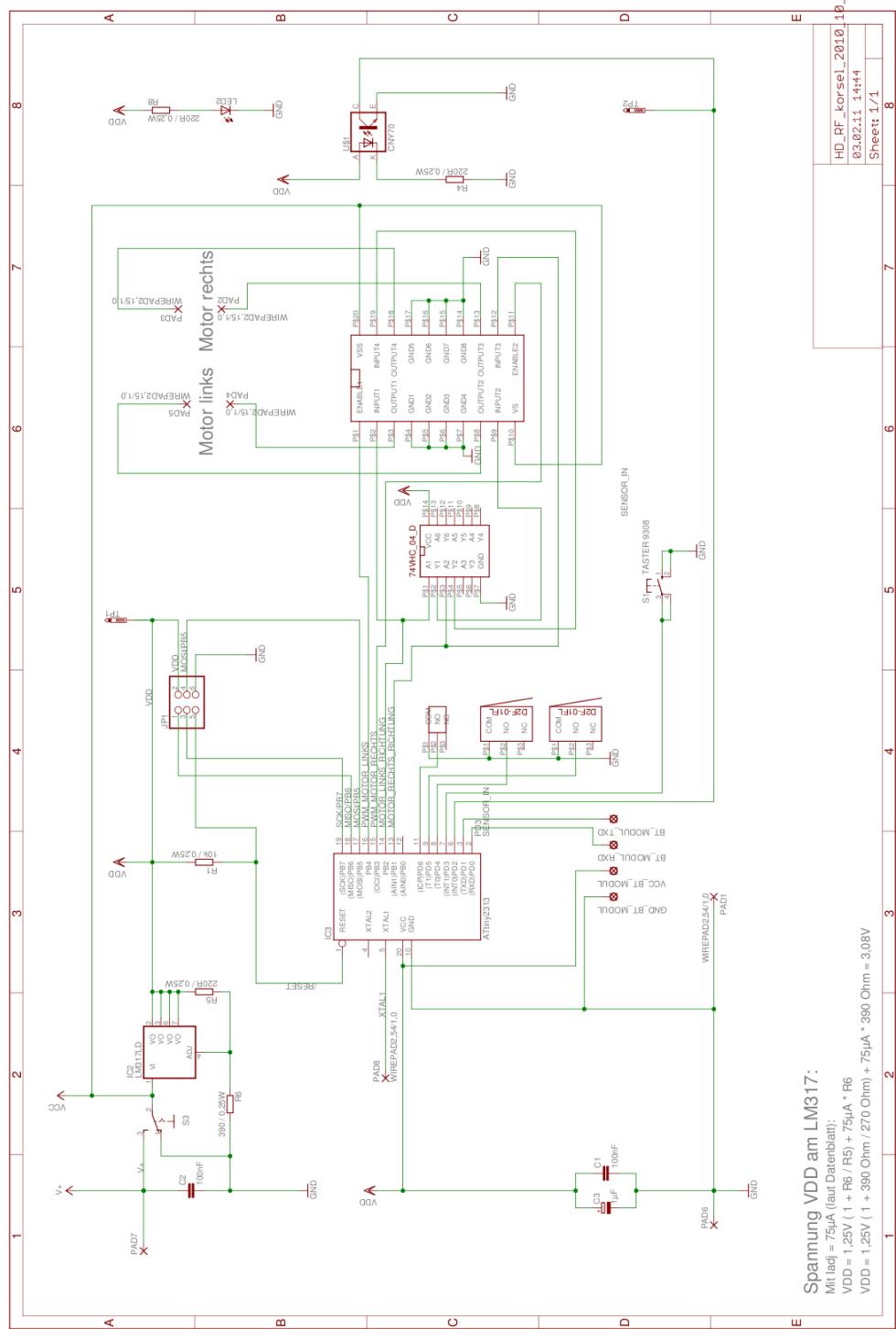


Abbildung 15: Schaltplan des Korsels

## 9.2 Korsel Platinenlayout

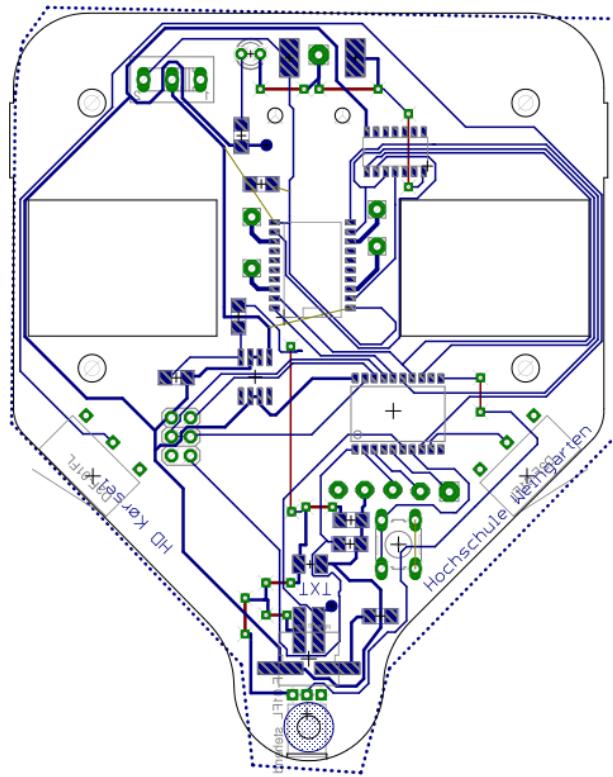


Abbildung 16: Platinenlayout des Korsels

## Literatur

- [ATM10] ATMEL: *8 bit AVR Microcontroller ATtiny2313 Datasheet*. Website, 2010. [http://www.atmel.com/dyn/resources/prod\\_documents/doc2543.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc2543.pdf); Stand: 05.06.2011.
- [Fes10] FESSLER, JOACHIM: *Das Korsel*. Website, 2010. <http://korsel.hs-weingarten.de/>; Stand: 05.06.2011.
- [Goo11] GOOGLE: *SensorEvent*. Website, 2011. <http://developer.android.com/reference/android/hardware/SensorEvent.html>; Stand: 05.06.2011.
- [Ora10] ORACLE CORPORATION: *VECMATH*. Website, 2010. <http://java.net/projects/vecmath/>; Stand: 05.06.2011.
- [Sen10] SENA TECHNOLOGIES: *User Guide for the Parani-ESD1000 v2.0.2*. Website, 2010. [http://www.sena.com/download/manual/manual\\_parani\\_esd1000-v1.0.1.pdf](http://www.sena.com/download/manual/manual_parani_esd1000-v1.0.1.pdf); Stand: 05.06.2011.