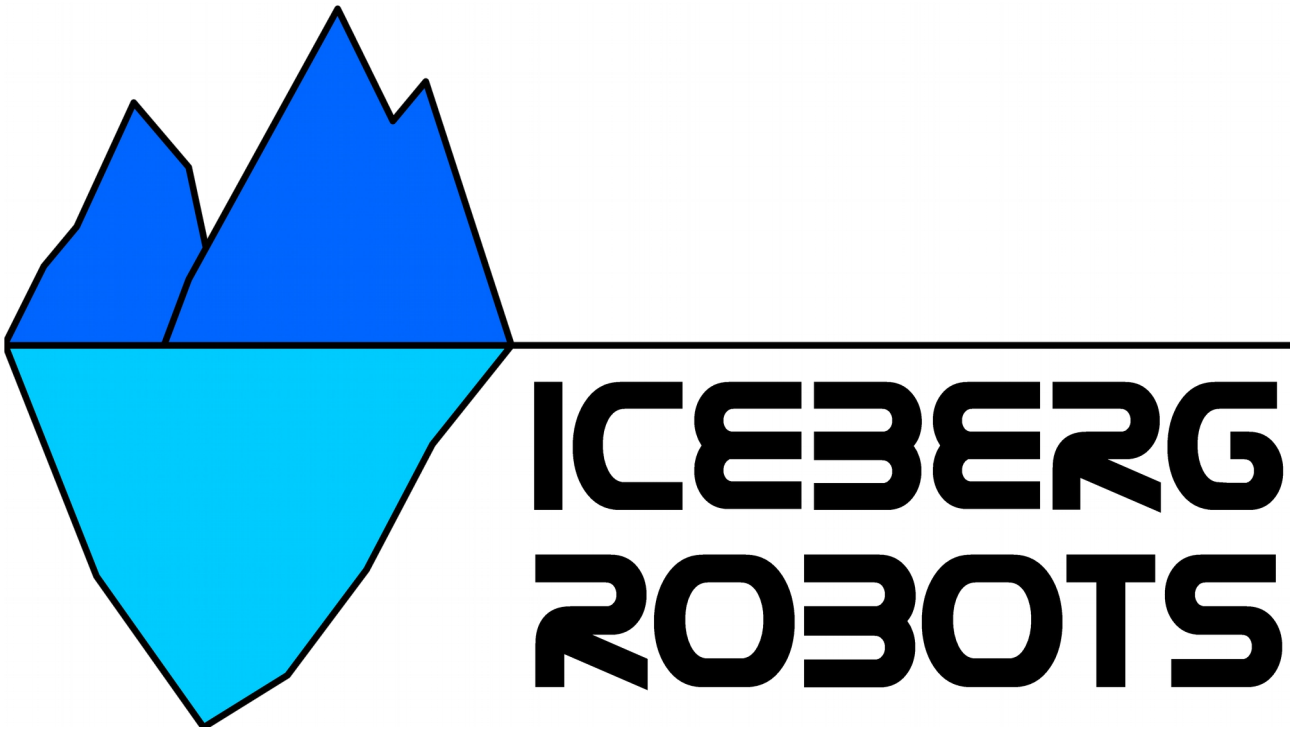
Dokumentation Iceberg Robots

Lehrer: Herr Abend



Seminarkurs Robotik

2.Semester Schuljahr 2016/2017

Inhaltsverzeichnis

1. Wettbewerb

Das Team Iceberg Robots nimmt in der Kategorie Soccer Open 2v2 am RoboCup Junior teil. In dieser Kategorie müssen zwei Roboter konstruiert und programmiert werden, die beim Wettkampf gegen die zwei Roboter des anderen Teams Fußball spielen.

1.1 Regeln

Im folgendem sind die wichtigsten Punkte aus den Regeln von 2016 zusammengefasst. Das komplette Regelwerk befindet sich im Anhang dieser Dokumentation.

1.1.1 Feld

Abbildung 1: Aufbau und Maße des Spielfeld

Das Feld hat eine Größe von 182cm x 243cm und ist umrandet von 14cm hohen schwarzen Banden. Die Spielfläche besteht aus grünem Filz. Auf der Spielfläche sind weiße Linien, die ein Rechteck von 122cm x 183cm markiert, in dem sich die Roboter bewegen dürfen. Außerhalb davon befindet sich das „Aus“. Die Tore sind 60cm breit und in den Farben Blau und Gelb gefärbt (Abb. 1). Die Roboter dürfen während des Spiels nicht in das „Aus“ fahren.

1.1.2 Ball

Der Ball ist Orange gefärbt und hat einen Durchmesser von 74mm +/- 10mm. Dies ist eine Neuerung zu letztem Jahr, wo der Ball Infrarotsignale ausgesendet hat. (Abb. 2)

Abbildung 2: passiver Ball

1.1.3 Roboter

Die Roboter dürfen einen maximalen Durchmesser von 22cm haben. Die Höhe ist ebenfalls

auf 22cm begrenzt. Das maximale Gewicht beträgt in unser Kategorie 2400g. Die maximale Betriebsspannung liegt bei 15V. Zudem darf der Ball nicht tiefer als 3cm in den Roboter hineinragen.

2. Aufbau unseres Roboters

Im folgendem gehe ich auf den Aufbau unseres Roboters ein. Es ist zu beachten, dass es sich um zwei baugleiche Roboter handelt.

2.1 Sensoren

2.1.1 Ultraschallsensoren

In unserem Roboter sind 4 Ultraschallsensoren (Abb. 3 A; Abb. 4) des Typs HC-SR04 verbaut, mit denen wir die Entfernung zu den Banden bestimmen, um uns mittig auf dem Feld zu positionieren.

Die Sensoren sind nach vorne, hinten, links und rechts ausgerichtet, sodass wir zu jeder der Banden den Abstand bestimmen können, und uns so auf dem Feld orten können. Falls ein Roboter zwischen uns und der Bande steht, können wir das bemerken, indem wir die Summe der gemessenen Abstände von zwei gegenüberliegenden Sensoren bilden und dies anhand der Spielfeldbreite auf Korrektheit überprüfen.

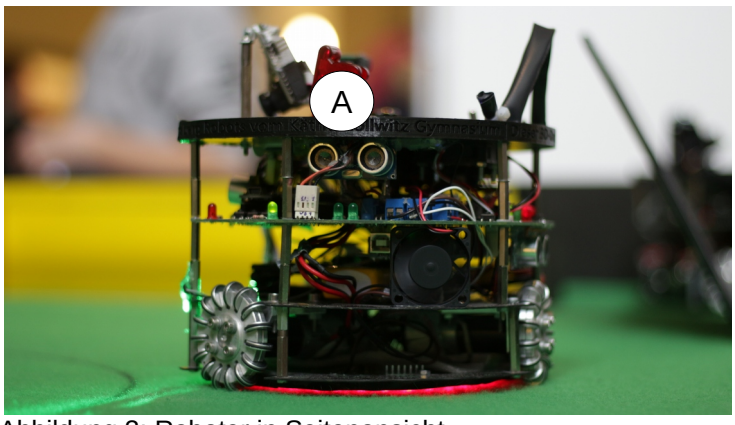


Abbildung 3: Roboter in Seitenansicht

Abbildung 4: HC-SR04 Ultraschallsensor

2.1.2 Kamera - Pixy CMUcam5

Da dieses Jahr der passive Ball eingeführt wurde standen wir erneut vor dem Problem der Ballerkennung. Da der Arduino nicht viel Rechenleistung besitzt war es unmöglich eine Kamera über den Arduino auszuwerten. Im Internet fanden wir eine Lösung für dieses Problem: die Pixy CMUcam5 (Abb 5). Diese Kamera übernimmt selbst die Bildauswertung und gibt die Ergebnisse über ISP direkt an den Arduino weiter, der mit den Ergebnissen dann

weiterverwertet. Die Pixy ist bei der Bilderkennung auf Farben limitiert, was für unsere Zwecke jedoch ausreicht, da der Ball und die Tore gefärbt sind. Die Kamera kann bis zu 50 mal in der Sekunde neue Ergebnisse liefern, was für eine schnelle Reaktion auf den Ball sehr hilfreich ist. Angebracht ist die Kamera oben auf dem Roboter (Abb 6 A), damit sie das Feld überblicken kann. Das Objektiv der Kamera haben wir durch ein 170° Weitwinkelobjektiv ersetzt, damit wir einen größeren Blickwinkel haben und größere Bereiche des Spielfeldes überblicken können.

Abbildung 5: Pixy CMUcam5

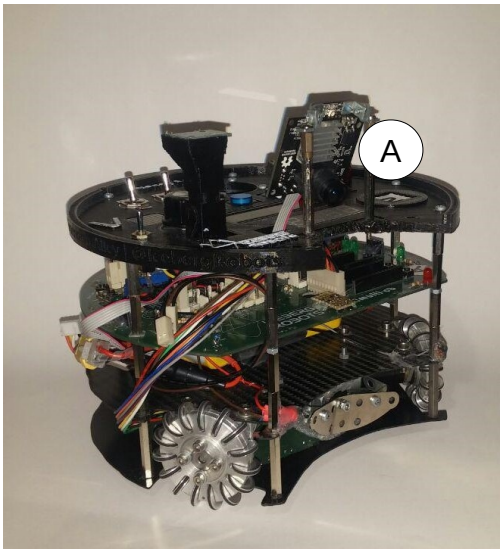


Abbildung 6: Position der Kamera

2.1.3 Bodensensor

Zur Erkennung der Linien am Spielfeldrand haben wir einen Bodensensor entwickelt, bestehend aus 32 Helligkeitssensoren (Fotowiderstände) und 32 Leuchtdioden, der von uns auf der Unterseite des Roboters angebracht wurde (Abb 7 A; Abb 8 A). Er wird über ein Kabel mit der Hauptplatine verbunden. Die Platine (Abb 7 A; Abb 8 A) für den Bodensensor haben wir vollständig selbst entworfen. Anschließend wurde sie uns von der Firma multi-cb im Rahmen eines Sponsorings drei mal angefertigt.

Um alle 32 Sensoren auswerten zu können benutzen wir 2 analoge Multiplexer der Firma Sparkfun, die jeweils einen von 16 Sensoren mit dem Arduino Nano auf der Hauptplatine verbinden.

Platinenentwurf und Schaltplan des Bodensensors: siehe Anhang.

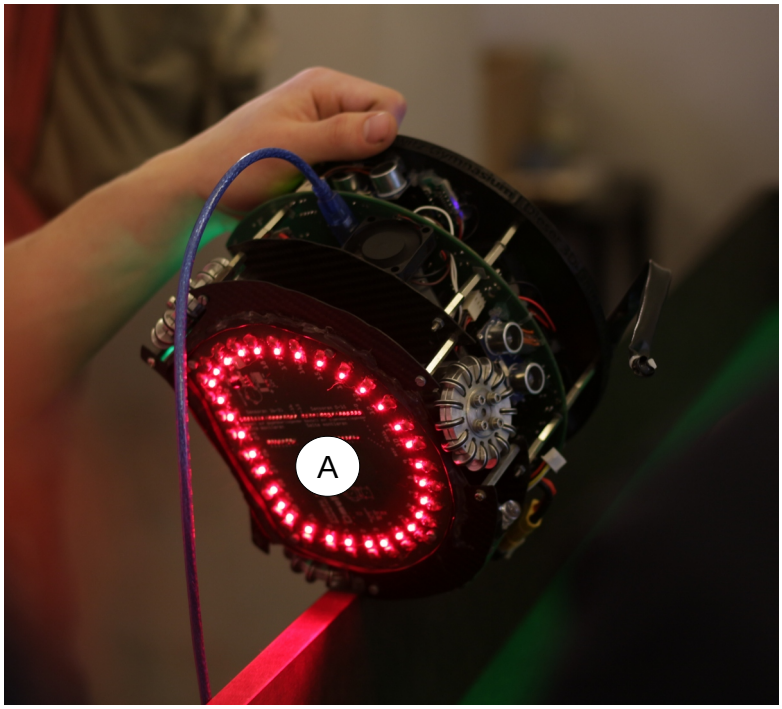


Abbildung 7: Bodensensor mit leuchtenden LEDs

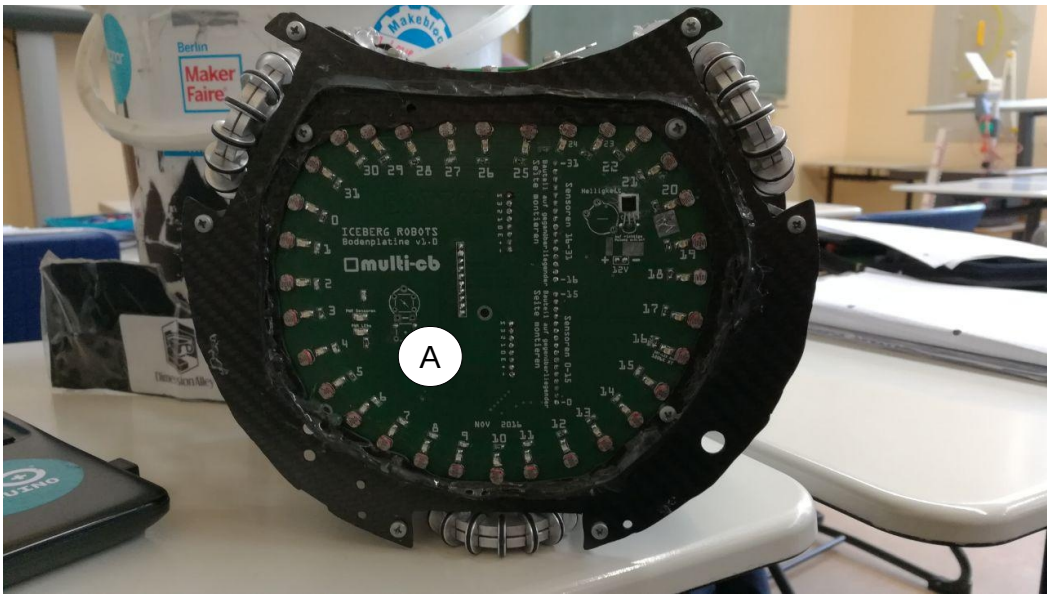


Abbildung 8: Bodensensor ausgeschaltet

2.1.4 Lichtschranke

Um zu erkennen, ob wir im Ballbesitz sind, haben wir eine Lichtschranke an unserem Roboter angebracht, bestehend aus einer grünen LED (Abb 9 A) und einem Fotowiderstand (Abb 9 B). Wenn der Ball in den Auffangbereich gelangt wird das Licht unterbrochen, und wir erkennen den Ballbesitz.

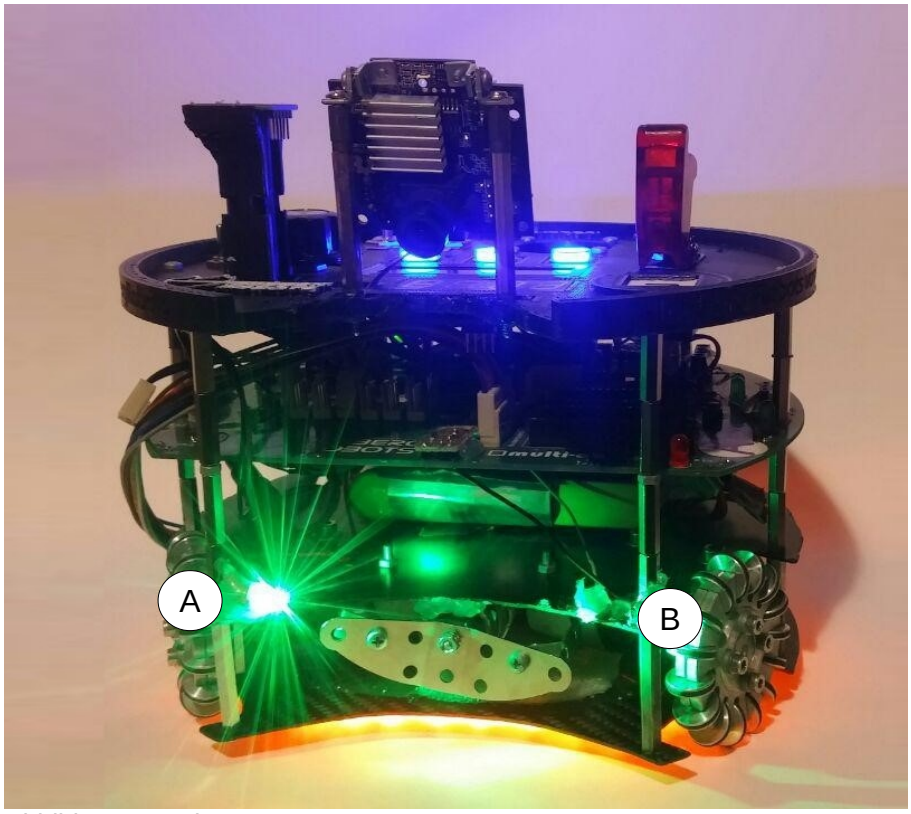


Abbildung 9: Roboter von vorne

2.1.5 Kompass

Um uns zum Tor auszurichten benutzen wir den Kompass CMPS11 (Abb10). Dieser hat zusätzlich auch noch einen Gyrosensor sowie einen Beschleunigungssensor eingebaut. Dies bietet einige zusätzliche Möglichkeiten, die momentan allerdings noch nicht zum Einsatz kommen. Der Kompass ist ganz oben auf dem Roboter angebracht (Abb 11 A), um den Abstand zu den Motoren zu maximieren, da diese den Kompass beeinflussen würden.

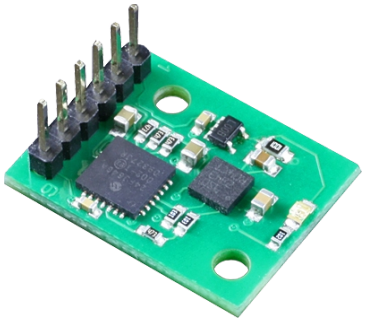


Abbildung 10: CMPS11

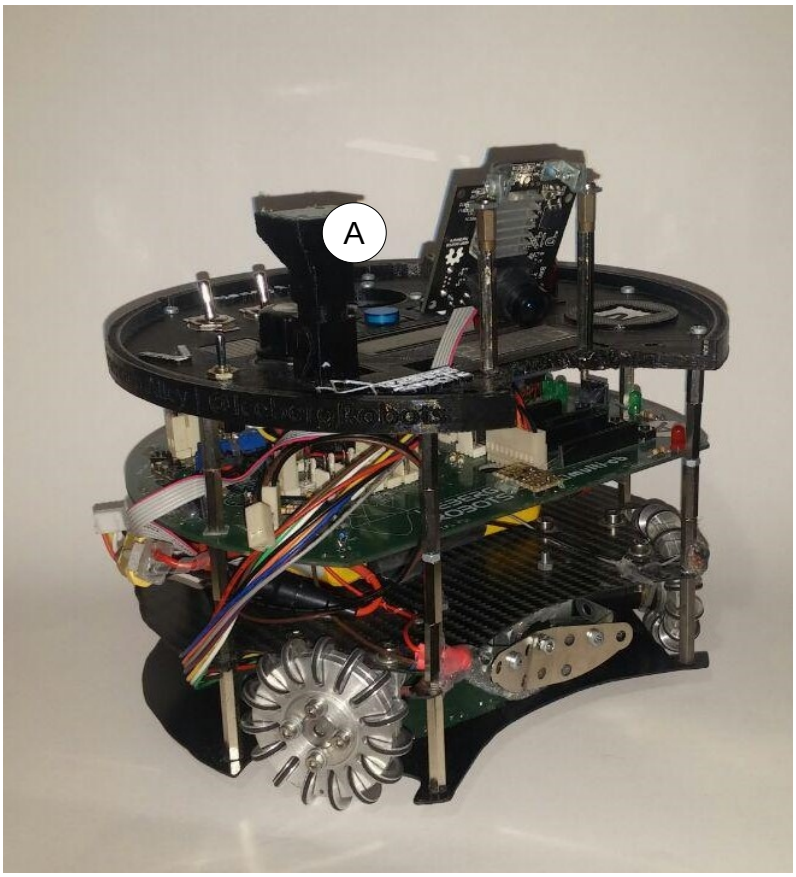


Abbildung 11: Roboter von der Seite

2.2 Aktoren

2.2.1 Motoren

Zur Fortbewegung benutzen wir 3 kernlose Motoren der Firma Maxon (Maxon 222037). Dazu haben wir das Planetengetriebe GP 22 C. Angesteuert werden die Motoren über 2 dual-H-Brücken des Typs L6205N.

2.2.2 OmniWheels

Als Räder kommen OmniWheels zum Einsatz, die in einem 120°-Winkel zueinander angeordnet sind. Das ermöglicht es uns, aus dem Stand in Jede Richtung zu fahren und uns beim fahren zu drehen. Unsere OmniWheels wurden von uns selbst entworfen und anschließend im 3D-Drucker gefertigt (Abb 12 B). Sie haben 8 Walzen, die sogenannten „Subwheels“ die auf kleinen Achsen im Rad gelagert sind, sodass diese sich gegen die Drehrichtung des Rades drehen können. Mit diesen OmniWheels haben wir die alte Version der metallenen OmniWheels (Abb 12 A) ersetzt.

(3D-Modell und technische Zeichnung im Anhang)

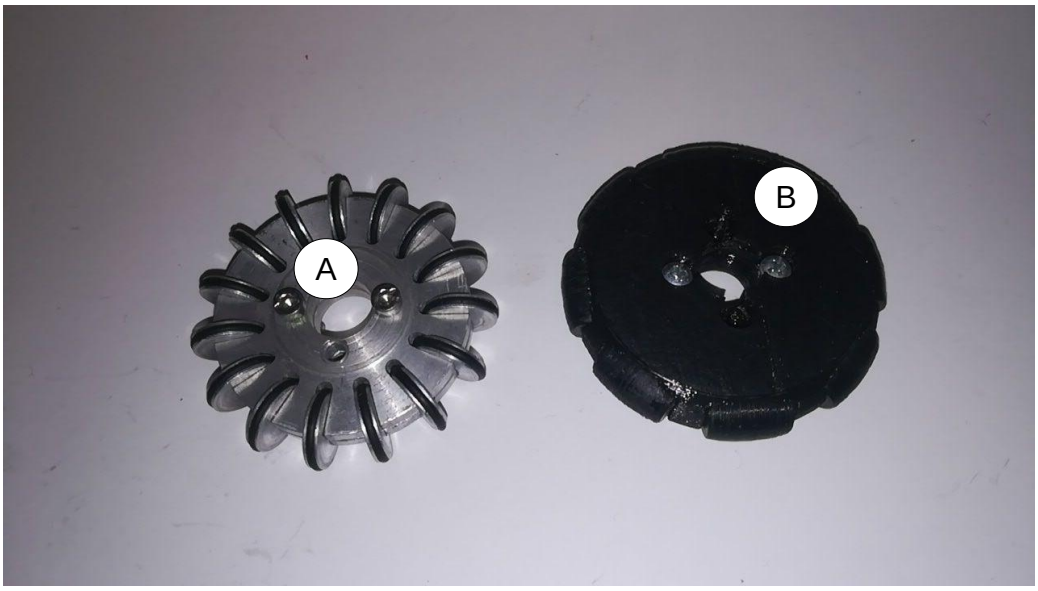


Abbildung 12: altes (A) und neues (B) OmniWheel

2.2.3 Kicker

Als Kicker verwenden wir einen 3V-Solenoid, den wir auf 12V über ein Relais ansteuern. Dieser Bewegt dann eine Platte (Abb 13 A), die den Ball nach vorne drückt. Die hohe Spannung ist kein Problem für den Solenoid, da wir den Schuss nicht länger als 50ms einschalten. Diese Zeit reicht aus, um den Ball zu schießen.

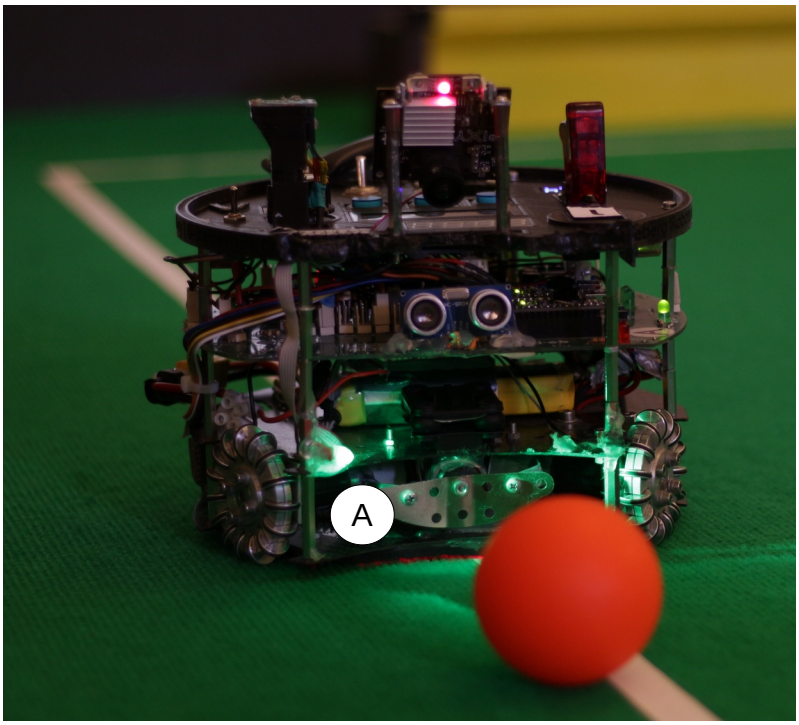


Abbildung 13: Roboter von Vorne

2.3 Controller

Um den Hauptcontroller zu entlasten haben wir die Aufgaben in unserem Roboter auf drei Arduinos aufgeteilt.

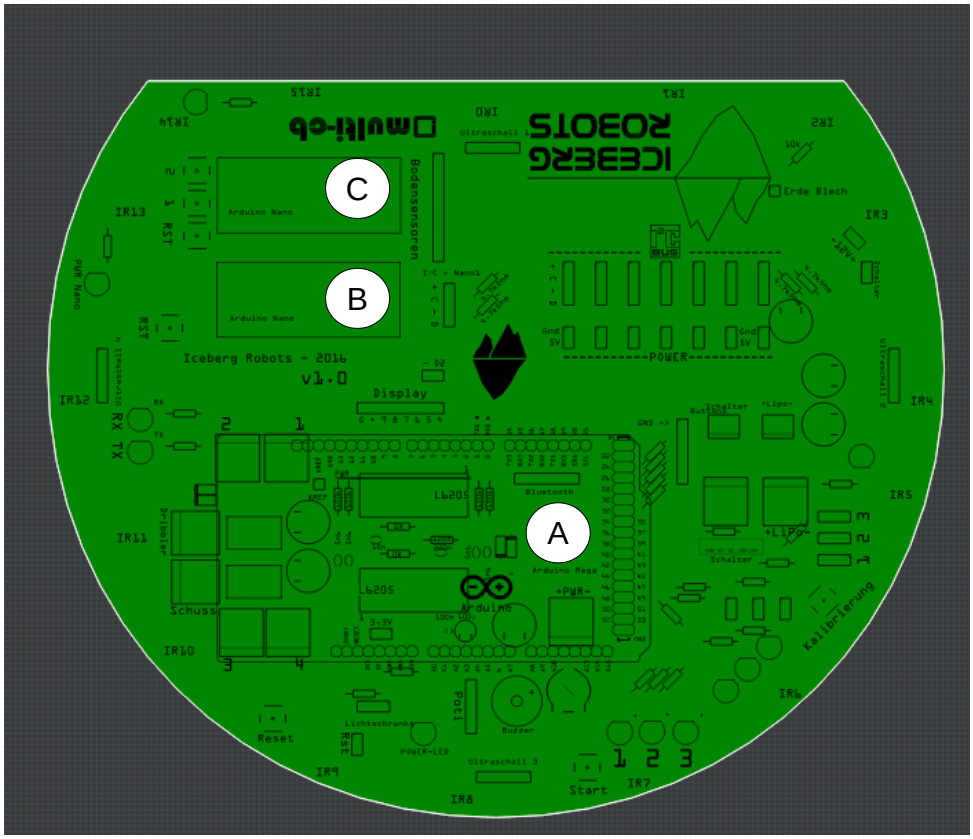


Abbildung 14: Entwurf der Platine

2.3.1 Hauptcontroller

Den Hauptcontroller (Abb 14 A) stellt ein Arduino Mega 2560 dar. Dieser erledigt die meisten Aufgaben in unserem Roboter, darunter die Berechnung der Motorsteuerwerte, das Ansteuern der Motoren und das Auslesen der Kamera. Zudem ist er für die Strategie verantwortlich.

2.3.2 Bodensensor-Controller

Da unser Hauptcontroller in der Hauptschleife für einen Durchlauf durchschnittlich etwa 100ms braucht, wäre es schwer die Linie mit dem Hauptcontroller zu erkennen, da die Linienerkenung öfter als zehn mal in der Sekunde geschehen sollte. Deswegen verwenden wir einen Arduino Nano allein dazu, den Bodensensor auszulesen (Abb 14 C) und zu überprüfen, ob wir uns auf der Linie befinden. Sollte der Arduino eine Linie erkennen, so sendet er einen Interrupt an den Arduino Mega, der dann entsprechend reagieren kann und von der Linie wegfahren kann.

2.3.3 Ultraschall-Controller

Da wir die Durchlaufszeit des Hauptcontrollers so gering wie möglich halten wollten, haben wir uns dazu entschlossen das Auslesen der Ultraschallsensoren auf einen weiteren Arduino Nano (Abb 14 B) auszulagern, da dies für alle vier Ultraschallsensoren relativ lange dauert. Dieser Arduino Nano nimmt demnach durchgehend Messwerte auf und stellt auf Anfrage des Hauptcontrollers über UART die zuletzt gemessenen Messwerte zur Verfügung. Urspünglich war geplant, diesem Arduino auch die Aufgabe zu geben, den Ring aus Infrarotsensoren auszuwerten. Dies wurde durch die Einführung des passiven Balls aber unnötig.

2.4 Platine

Die Platine bildet das Herzstück unseres Roboters. Auf ihr befinden sich die drei Controller (Abb 15 C), die Motortreiber (Abb 15 D), die Relais für Schuss und einen Dribbler (Abb 15 E), der ergänzt werden kann, ein Bluetoothmodul (HC-05) (Abb 15 B), einige Signalleds, einen Buzzer (Abb 15 F), einige Buttons und Anschlüsse für alle Sensoren, die wir verwenden. Die Hauptplatine enthält auch Plätze für 16 Infrarotsensoren (TSOP31240) (Abb 15 A), da wir bei der Entwicklung der Platine nicht davon ausgegangen sind, dass dieses Jahr der passive Ball eingeführt wird.

Durch die Platine sind die Schaltkreise des Roboters deutlich übersichtlicher und platzsparender.

(Platinenentwurf und Schaltplan: siehe Anhang)

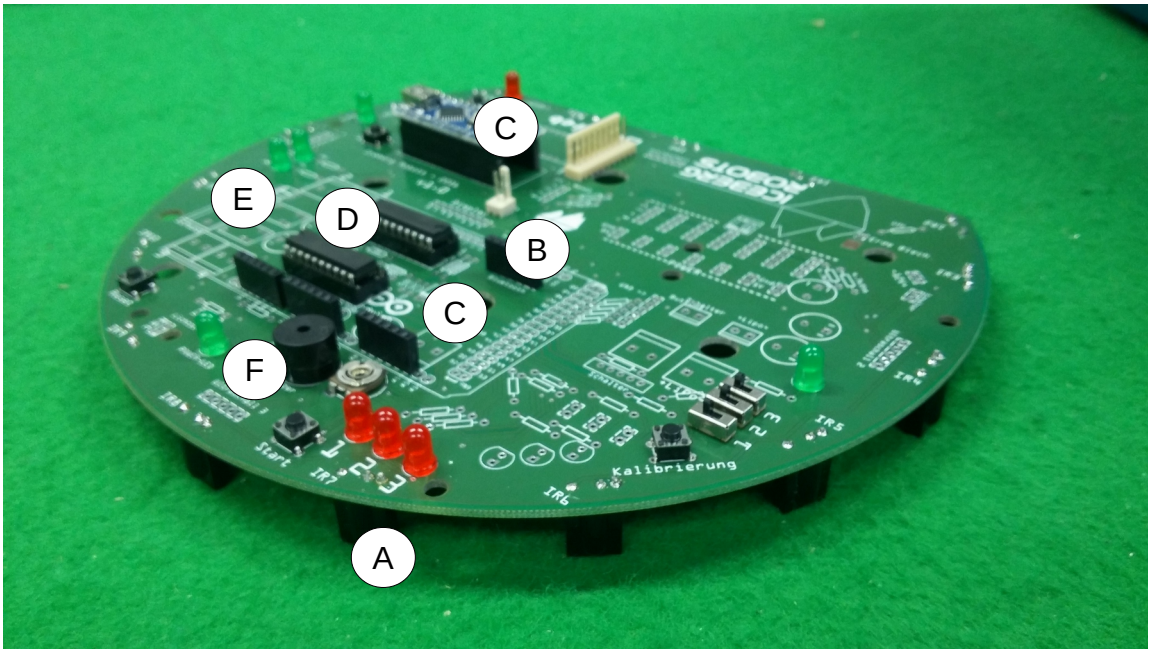


Abbildung 15: unvollständig Bestückte Platine

2.5 Top-Platte

Oben auf unserem Roboter befindet sich die Top-Platte, auf der sich Buttons und Schalter befinden, mit denen wir unseren Roboter starten und Einstellungen im Spiel vornehmen.

Die Platte wurde von uns entworfen und dann von DimensionAlley im Rahmen eines Sponsoring in einem 3D-Drucker gefertigt.

Auf der Platte befinden sich drei Druckschlater (Abb 16 A) mit LEDs, zwei Kippschalter (Abb 16 B), ein Hauptschalter mit LED (Abb 16 C), ein Potentiometer (Abb 16 D), eine 4x7Segment-Anzeige (Abb 16 E), eine 8x8 LED-Matrix (Abb 16 F), ein Voltmeter (Abb 16 G), mit dem wir die Spannung unseres Akkus überprüfen, ein LCD-Display (Abb 16 H), ein Lüfter (Abb 16 I), der die Motortreiber kühlt, die Halterung für unsere Kamera (Abb 16 J) und der Turm auf dem der Kompass (Abb 16 K) sitzt.

(3D-Modell: siehe Anhang)

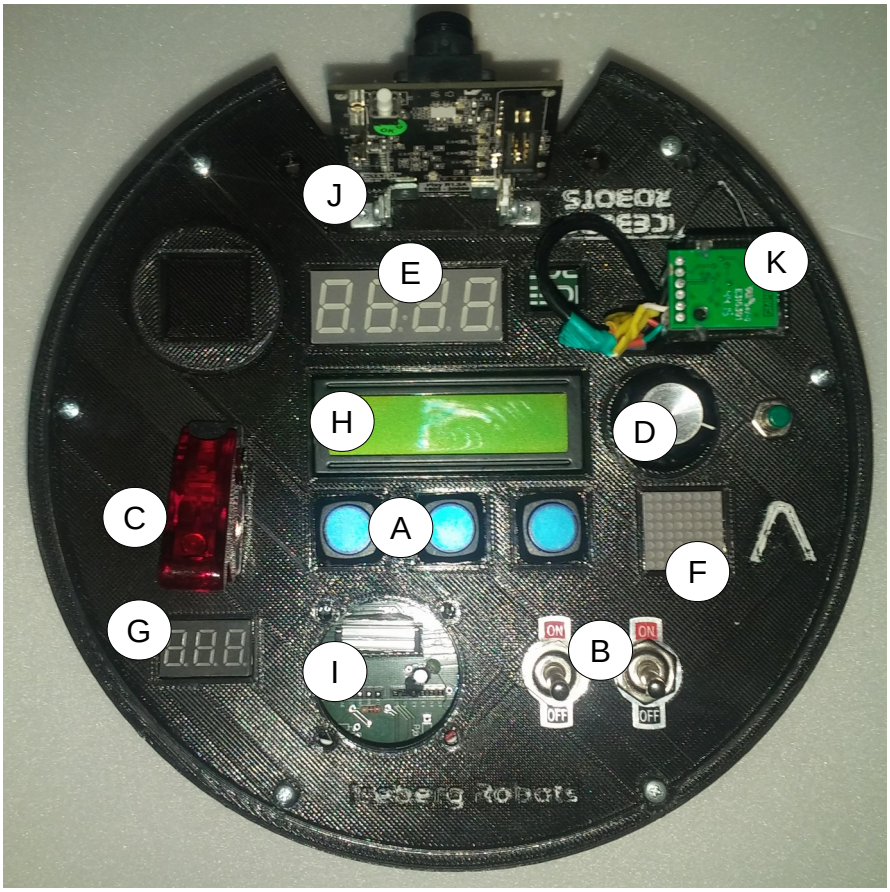


Abbildung 16: Topplatte

2.6 Stromversorgung

Als Stromversorgung verwenden wir einen dreizelligen LiPo (Litium Polymer Akkumulator) mit einer Nennspannung von 11,1V und einer Kapazität von 2,2 Ah (Abb 17). Geladen hat dieser eine Spannung von 12,6V. Dies passt gut zu unseren 12V-Motoren, und auch die

Arduinos vertragen diese Spannung (zulässig sind 9V bis 20V).

Abbildung 17: LiPo

3. Software

Da unser Roboter Arduinos als Controller verwenden, sind wir auf Arduino als Programmiersprache angewiesen. Diese Sprache beruht auf C++ und dient dazu, die Mikrocontroller zu programmieren.

Im Anhang befindet sich das Programm, welches wir auf der MakerFaire 2017 verwendet haben. Dieses ist komplett Wettkampfsfähig und funktioniert sehr gut. Autor dieses Programmes ist zum größten Teil Finn Harms.

3.1 Strategie

Unser Roboter hat die Strategie sich während des Spiels nicht zu drehen und durchgehend in Richtung des Gegnerischen Tors zu stehen. Wenn sich der Ball nicht in unserem Blickfeld befindet, fahren wir Rückwärts und positionieren uns vor unserem eigenen Tor. So decken wir unser eigenes Tor und sind gleichzeitig in einer guten Position um einen neuen Angriff zu starten, wenn wir den Ball wieder sehen.

3.2 Kameraauswertung

Da die Kamera die Kameraauswertung selbst übernimmt muss diese nur über die Software PixyMon (Abb 18) am PC auf die entsprechenden Farben eingestellt werden, und kann dann über den Arduino ausgelesen werden.

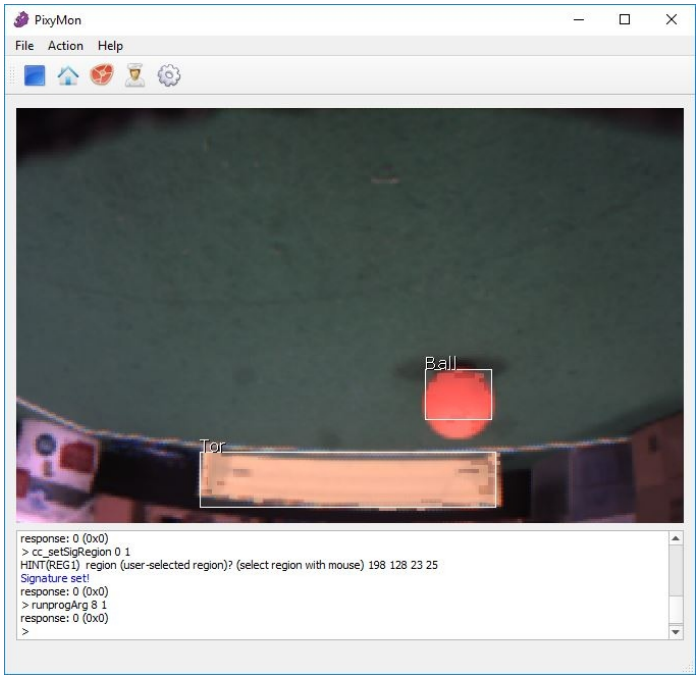


Abbildung 18: PixyMon

3.3 Fahrlogik

Da wir zum fahren OmniWheels (siehe 2.2.2) verwenden bedarf es zum fahren in eine bestimmte Richtung an speziellen Berechnungen. Hierzu verwenden wir die Chassisbibliothek von Herrn Kreutel (siehe Anhang).

3.4 Linienerkennung

Zur Erkennung der Linien verwenden wir den von uns erstellten Bodensensor (2.1.3). Auf diesem befindet sich ein analoger Multtplexer, an dem die Fotowiderstände angeschlossen sind. Angesteuert wir der Bodensensor über einen eigenen Arduino Nano, der sich nur um den Bodensensor kümmert. Das zugrundeliegende Problem war, dass der Arduino Mega lange Schleifendurchläufe hatte, und dadurch nicht durchgehend den Bodensensor auslesen konnte. Somit ist es vorgekommen, dass wir die Linie überfahren haben.

Jetzt ließt der Arduino Nano durchgehend den Liniensensor aus und sendet einen Interrupt an den Arduino Mega, sobald eine Linie befahren wird. Anschließend wird die Position der Linie ermittelt und dem Arduino Mega mitgeteilt, worauf dieser reagieren kann. Zur Ermittlung der Position der Linie werden die 32 einzelnen Sensoren in vier Sektoren unterteilt. Jetzt wird gezählt, in welchem der Sektoren die meisten Sensoren ausschlagen. So kann bestimmt werden, in welchem Sektor sich die Linie befindet (siehe Bodensensor_v7.ino Zeile 132-199 und 282-294).

Der Arduino Nano hat einen Codeabschnitt der sich um das kalibrieren des Sensors

kümmert. Dazu wird für jeden Sensor ein Linienwert und ein Feldwert gemessen und daraus ein Schwellwert gebildet. Dieser Schwellwert wird dann im EEPROM des Arduinos gespeichert, damit dieser nicht bei jedem Neustart neu kalibriert werden muss. Bei der Kalibrierung wird automatisch geprüft ob einer der 32 Sensoren defekt ist, sodass dieser aus der Messung ausgeschlossen werden kann (siehe Bodensensor_v7.ino Zeile 77-95 und 202-269)

Das senden des Interrups passiert indem ein Pin am Arduino Nano auf HIGH gesetzt wird. Dieser Pin ist zu einem Interruptfähigen Pin am Arduino Mega verbunden. Zudem wird die interne LED (Pin 13) als Statusmeldung eingeschaltet (siehe Bodensensor_v7.ino Zeile 271-279 und Abb 19).

```
271 //-- Diese Methode sendet einen Interrupt an den Arduino Mega
272 void interrupt(){
273   digitalWrite(13,HIGH);
274   digitalWrite(3,HIGH);
275   delay(5);
276   digitalWrite(3,LOW);
277   digitalWrite(13,LOW);
278   delay(5);
279 }
```

Abbildung 19: Die Interruptmethode

3.5 Auslesen der Ultraschallsensoren

Die Ultraschallsensoren werden über die NewPing Bibliothek ausgelesen. Diese ermöglicht es uns ein Objekt für jeden Ultraschallsensor zu erstellen (siehe UsVers_0.2.ino Zeile 17-20) das dann über eine Methode verfügt, um den Ultraschallsensor auszulesen.

In der Hauptloop werden alle vier Ultraschallsensoren ausgelesen und die Ergebnisse werden in einem Array gespeichert, aus welchem die Werte später ausgelesen werden können (Abb 20).

```
36 void loop(){
37   wert[0] = sonarR.ping_cm();
38   wert[1] = sonarF.ping_cm();
39   wert[2] = sonarL.ping_cm();
40   wert[3] = sonarB.ping_cm();
41   delay(30);
42 }
```

Abbildung 20: Auslesen der Ultraschallsensoren

Wenn der Arduino Mega die Ultraschallwerte braucht setzt dieser den Interrupfähigen Pin 3 am Arduino Nano auf High, worauf dieser die Methode usAusgeben() (Abb 21) aufruft, in der die die Werte an den Arduino gesendet werden (Kommunikation siehe 3.6).

```
46 void usAusgeben(){
47   for(int i=0;i<4;i++){
48     Serial.write(wert[i]);
49   }
50   delay(5);
51 }
```

Abbildung 21: Senden der Werte an den Arduino Mega

Das komplette kommentierte Programm befindet sich im Anhang.

3.6 Kommunikation der einzelnen Controller

Die verschiedenen Arduinos kommunizieren untereinander über UART (kurz für Universal Asynchronous Receiver Transmitter). UART ist eine Schnittstelle zur Datenübertrageng wobei daten bei einer vorher festgelegten Bitdauer übertragen werden.

Am Arduino Nano gibt es den RX0 und den TX0 pin, der eine Pin ist für den Datenausgang, der andere für den Dateneingang. Der Arduino Mega hat 4 UART-Schnittstellen. Für uns ist das optimal. Zwei der Schnittstellen werden verwendet um mit den Arduino Nanos zu kommunizieren. Eine weitere um mit dem Blueoothmodul zu kommunizieren. Die Schnittstelle RX0, TX0 ist reserviert für die Kommunikation mit dem Computer für Debuggingzwecke.

In Arduino wird die UART-Kommunikation über die Serial-Klasse gelöst. Mit der Methode Serial.print() können Strings in einzelnen Bytes übertragen werden. Mit Serial.write() können einzelne Bytes gesendet werden. Mit Serial.read() können selbige empfangen werden.

```
byte output = 42;
byte input;

//senden:
Serial.write(output);         //senden eines Bytes

//empfangen:
if(Serial.available() > 0){   //prüfen, ob ein Byte gesendet wurde
  input = Serial.read()       //empfangen eines Bytes
}
```

Abbildung 22: Aufbau einer seriellen Kommunikation

3.7 Kommunikation der beiden Roboter

Unsere Platine bietet Platz für ein HC-05 Bluetoothmodul (Abb. 23), mit welchem die beiden Roboter untereinander kommunizieren können. Wir haben es geschafft die beiden

Roboter kommunizieren zu lassen, im Wettbewerb jedoch haben wir dieses Feature nicht verwendet. Die Kommunikation funktioniert ähnlich wie in 3.6 beschrieben, mit dem Unterschied, dass die der die RX und TX Leitungen das Bluetoothmodul und den Arduino verbinden.

Abbildung 23: HC05-Modul

4. Sponsoren

Für unsere Arbeit haben wir auch nach Sponsoren gesucht, die uns finanziell und materiell unterstützen. Zu unseren wichtigsten Sponsoren gehören DimensionAlley, die uns unsere 3D-Drucke gesponsert haben und MultiCB, die uns unsere Leiterplatten hergestellt haben.

5. Ausblick

Im nächsten Jahr werden wir auf vier Motoren umsteigen, die zudem eine deutlich höhere Drehzahl haben werden. Des Weiteren werden wir eine neue Platine fertigen lassen und ein neues Chassis entwerfen.

3D-Modell, technische Zeichnung, Schaltplan Hauptplatine siehe Anhang

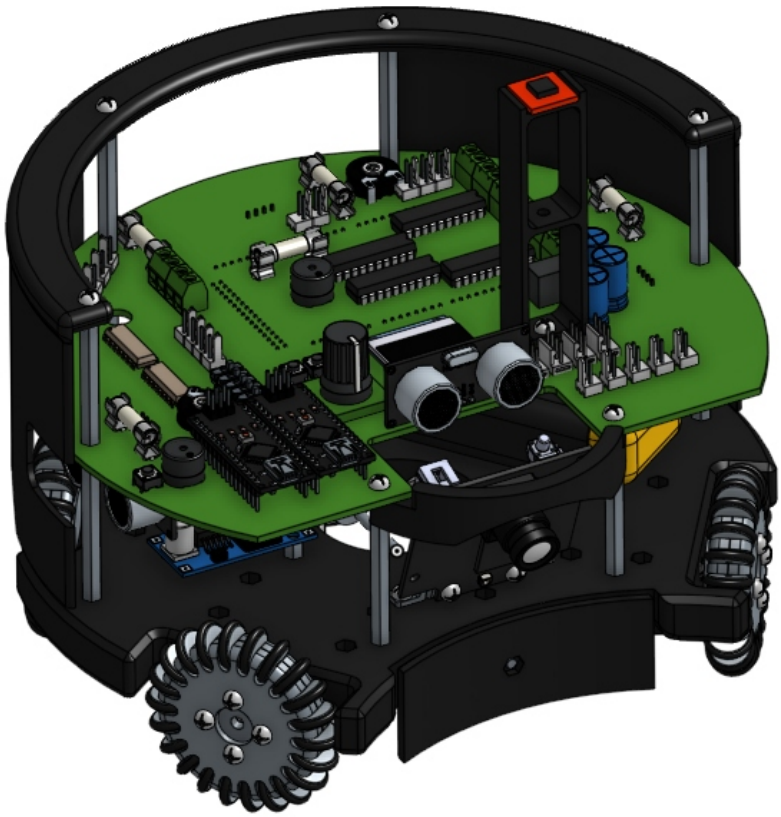


Abbildung 24: Entwurf des Roboters für die nächste Saison

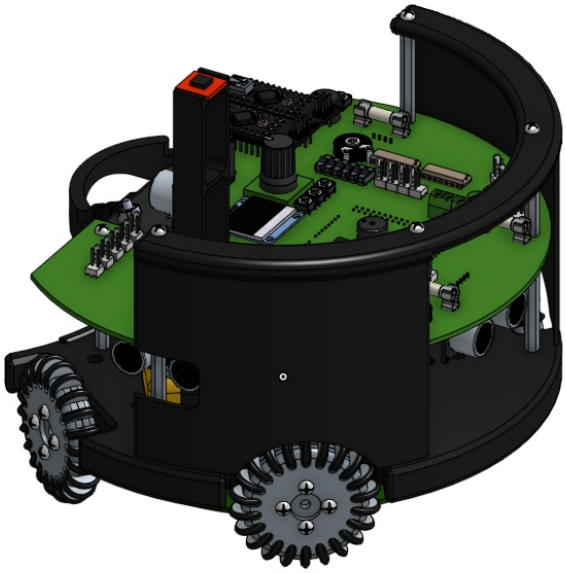


Abbildung 25: Entwurf des Roboters für die nächste Saison (hinten)

6. Glossar

EEPROM

Ein eingebaute Speicher über den jeder Arduino verfügt. In ihm können Bytes gespeichert werden (Werte von 0 bis 255).

Nähere Informationen hier: https://www.arduino.cc/en/Reference/EEPROM

Interrupt

Ein Arduino verfügt über interruptfähige Pins. Man kann den Arduino so programmieren, dass dieser seinen Momentanen Prozess sofort unterbricht, und eine bestimmte Methode ausführt, sobald sich das an diesem Pin angelegte Signal auf eine festgelegte Weise ändert.

In dieser Interruptmethode dürfen keine delays stehen, da diese in Interrupts nicht funktionieren.

Nähere Informationen hier: https://www.arduino.cc/en/Reference/attachInterrupt

OmniWheel

OmniWheels (kurz für Omni-Directional-Wheels) sind Räder die senkrecht zu ihrer Drehrichtung rollen können. Dafür haben sie viele kleine Subwheels, welche senkrecht zur Drehachse des Rades angeordnet sind.

7. Bildquellen

Pixy (3.6.17):
http://www.robot-electronics.co.uk/products/sensors/cameras/cmucam5-pixy.html

Ball (3.6.17):
http://schweikert-shop.he-hosting.de/index.php?cat=2259&product=93011

Feld (3.6.17):
https://www.robocupgermanopen.de/sites/default/files/soccer_2016.pdf

HC-SR04 (3.6.17):
http://www.ezsbc.com/index.php/products/hc-sr04-ultrasonic-distance-sensor.html#.WTMuTMakLcs

CMPS11 (3.6.17):
https://www.robot-electronics.co.uk/htm/cmps11doc.htm

LiPo (4.6.17):
https://hobbyking.com/media/catalog/product/cache/3/image/9df78eab33525d08d6e5fb8d27136e95/legacy/catalog/21346.jpg

HC05 (5.6.17):
http://www.ebay.com/itm/MODULO-BLUETOOTH-HC-05-MAESTRO-ESCLAVO-6-PIN-JY-MCU-SERIAL-TTL-UART-HC05-ARDUINO-/152088485996