ROBOTICS RUTESHEIM

Dokumentation

Dokumentation unserer Arbeit am Roboter für den World Robot Olympiad Future Engineers Wettbewerb - Saison 2024.

by Robotics Rutesheim.
Future Engineers Team: Max, Maxi, Darian
Robotik AG des Gymnasium Rutesheim.

Inhalt

1. Motorisierung

Beim Thema Motorisierung haben wir uns für zwei Motoren entschieden, und zwar für einen Servomotor und einen Antriebsmotor. Der Servomotor, welcher für die Lenkung der Vorderachse zuständig ist, wird von zwanzig bis achtzig Grad angesteuert.

Dabei bedeuten zwanzig Grad in der Software eine Lenkung nach Links, fünfzig Grad die Fahrt geradeaus und achtzig Grad eine Lenkung nach Rechts.

Der Antrieb unseres Roboters besteht aus vier wesentlichen Bestandteilen:

Die wichtigste Rolle hierbei spielt der Antriebsmotor selbst, welcher mit 12 V, die mit einem 9 V Block und zwei 1.5 V Batterien erzeugt werden, am Motorshield betrieben wird.

Der Motor liegt direkt an einem Zahnrad, welches daraufhin an einem weiteren, kleineren Zahnrad liegt. Dies haben wir so gestaltet, damit wir eine höhere Drehzahl auf unsere Räder übertragen können. Die Übertragung von Zahnrad auf Räder ist durch eine Hinterachse möglich. Jedoch haben wir uns gegen die Verwendung eines Differentials entschieden, da dies für unsere Kurvenfahrt mithilfe des Servos nicht notwendig war und somit ausschließlich Platz verbraucht hätte.

Wir haben uns für diese Art von Hinterachsenantrieb entschieden, da dies eine sehr einfache, aber doch sehr effiziente Art des Antriebs ist. Durch das direkte Zusammenspiel von Motor und Hinterachse bietet diese Art des Antriebs eine gute Traktion und des Weiteren funktioniert sie sehr gut mit der durch den Servomotor generierten Vorderachsenlenkung.

2. Hindernisse

Umfahrung der Hindernisse und Vermeidung von Kollisionen

Um alle Hindernisse korrekt zu umfahren und nicht mit Begrenzungen zu kollidieren, berechnen wir, im Eröffnungsrennen, am Anfang des Eröffnungsrennens und im Hindernisrennen am Anfang jeder Geraden eine Linie, auf welcher der Roboter die Geraden befahren soll. Diese Linie wird als Abstand zur äußeren Begrenzung angegeben. Deshalb sind die drei Ultraschallsensoren auch sehr wichtig, um den Roboter auf der korrekten Spur zu halten und ihn dort hin zu navigieren. Im Eröffnungsrennen geben wir diese Linie am Anfang vor. Im Hindernisrennen wird sie je nach der Farbe des Blocks auf der Geraden festgelegt, indem eine Kamera, die Farbe des Objekts feststellt und ein Algorithmus dann zwischen zwei verschiedenen schon vorprogrammierten Linien entscheidet (siehe Abbildung 1 im Anhang).

Die Linien sind so gesetzt, dass egal in welcher Hindernismarkierung das Hindernis steht, grüne Blöcke immer links und rote Hindernisse immer rechts umfahren werden und der Roboter weder mit Hindernissen noch mit Begrenzungen kollidiert.
Die Position dieser Linien ändert sich natürlich je nach Fahrtrichtung.

Spurhalte-Algorithmus – Dreiecks Algorithmus

Um auf diese, wie vorher beschreiben, bestimmten Linien zu fahren und den Kurs des Roboters zu korrigieren, nutzen wir einen Algorithmus, welcher mithilfe Trigonometrie den Winkel berechnet, mit welchem der Servo gegenlenken muss, um immer wieder auf unsere Linie zu gelangen. Dafür berechnet der Code, sobald der Roboter von der vorgesehenen Linie abweicht, ein rechtwinkliges Dreieck (siehe Abbildung 2 im Anhang). Die Grundseite wird aus der Differenz von der Linie und der tatsächlichen Distanz zur Begrenzung berechnet. Das ist möglich, da die Linie ebenfalls als Distanz zur Begrenzung angegeben wird. Als erster Schritt wird die Grundseite B des Dreiecks mit der gegebenen aktuellen und der gewünschten Distanz zur äußeren Begrenzung berechnet.

```
if (distanceout < line){ //distanceout ist die Distanze zur äußeren Begrenzung
      b = line - distance1;
      }
      else {
      b = distance1 - line;
      }
```

Als nächstes wird die Hypotenuse des Dreiecks mithilfe des Satz des Pythagoras berechnet. Dafür haben wir die Höhe a des Dreiecks frei, auch 20 cm gewählt. Das heißt, dass der Roboter in 20 cm wieder auf der Linie sein wird.

```
c2 = (b*b)+(a*a);//a = 20cm
```

```
      c = sqrt(c2);
```

Im nächsten Schritt nutzen wir den Arkustangens, um den Winkel β zu berechnen. Außerdem wird mithilfe der Winkelsumme α berechnen, da wir wissen, dass α + β = 90°. A wird dann von einem Float zu Integer konvertiert, da für servo.write(Winkel) nur Integer zulässig sind. Zuletzt wird Gamma mit der Winkelsumme 180° im Dreieck berechnet.

```
beta = atan(20/b) * 180 / PI;
alpha = 90 - beta;
alphaint = alpha;
float gamma = 180 - 90 - beta;
```

Mit den berechneten Winkeln α und γ kann der Servo nun im ersten Schritt nun zuerst einlenken und dann wieder auslenken, um auf die Linie zurückzukehren.

Während des gesamten Algorithmus wird mit einem Infrarotsensor überprüft, ob der Roboter die orangene oder blaue Linie vor den Enden der Gerade überfährt. Wenn das zutrifft, beenden wir den „Dreiecks-Algorithmus“ um eine Kurve zu lenken (siehe Code). Wir haben uns für diese Aufgabe für einen Infrarotsensor entschieden, da wir in ersten Versuchen mit einem nach vorne gerichteten Ultraschallsensor oft den Algorithmus fälschlicherweise beendeten, da der Ultraschallsensor auch unter den
70 cm Threshold fällt, wenn der Roboter während des korrigieren der Bahn kurz schräg fährt.

```
if (ir == LOW) {
  break;
}
```

Funktionen zur einfacheren Programmierung

Wir haben für alle wichtigen Abläufe des Roboters Funktionen programmiert, um so schnell und einfach im Algorithmus auf sie zugreifen zu können, Z.B. werden in der Funktion getData() die Ultraschall- und der Gyrosensor abgerufen. In der Funktion cam() wird die Kamera abgerufen. Die Trennung beim Aktualisieren von Kamera und Ultraschall- und Gyro-Sensoren ist bewusst gewählt, da die Daten der Kamera deutlich seltener gebraucht werden und sie immer wieder aufzurufen und nicht zu nutzen das Programm unnötig verlangsamen würde. Mit den Funktionen motorForward(int speed) (siehe folgender Code) oder motorStop() können wir den Motor steuern.

```
void motorForward(int speed){

 digitalWrite(motor1 , HIGH);
 digitalWrite(motor2 , LOW);
 analogWrite(motorspeed , speed);
}
```

Probleme bei der Programmierung

Eines der größten Probleme bei der Programmierung waren Ausfälle der Sensoren. Besonders die Ultraschallsensoren und der Gyro lieferten oftmals falsche oder ungenaue Werte und es hat uns einige Zeit gekostet, herauszufinden, wie wir die Werte richtig interpretieren. Beispielsweise geben die Sensoren 0 cm aus, wenn sie kein Objekt in den nächsten 90 cm erkennen oder es einen Fehler bei der Messung gibt. Deshalb haben wir in den Code auch Algorithmen integriert, die falsche Werte herausfiltern (siehe folgender Code) und 0 cm Entfernung als kein Hindernis gewertet.

```
//Falls der Abstand 0 ist liegt evtl. eine Fehlmessung vor. Dann wird nochmal
gemessen. Die Messung wird aber höchstens 3x wiederholt um in keine
Endlosschleife zu kommen.
do{
  distance1 = sonar1.ping_cm();
  distance2 = sonar2.ping_cm();
  distance3 = sonar3.ping_cm();
  zeahler++;}
  while(distance1 == 0 && zeahler < 3|| distance2 == 0 && zeahler < 3||
distance3 == 0 && zeahler < 3);
  zeahler = 0;
```

Lenkung

Um Kurven zu lenken, nutzen wir entweder die Winkel, welche wir aus unserem „Dreiecks-Algorithmus“ erhalten, oder die Winkel 20° und 80° um scharfe Kurven am Ende der Geraden zu fahren. Um zu überprüfen, wie lange der Roboter lenken muss nutzen wir den Winkel des Gyrosensors (Bsp. siehe folgender Code).

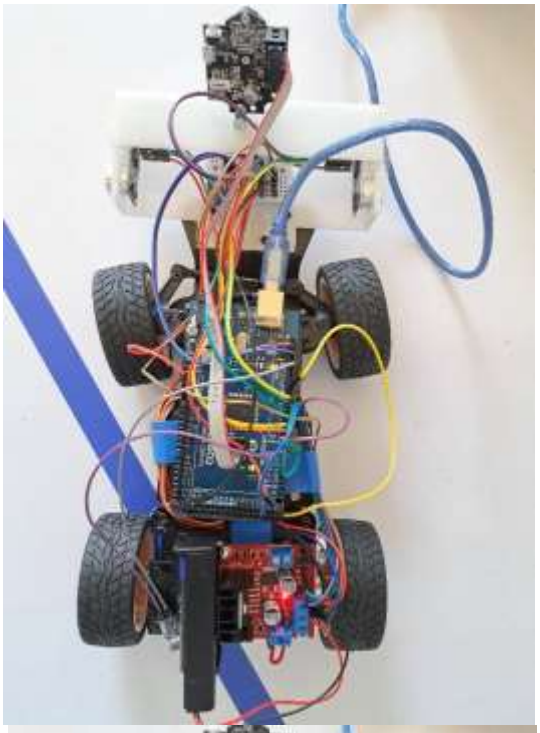
```
while (z < curve) {
        servo.write(20);
        getData();
    }
```

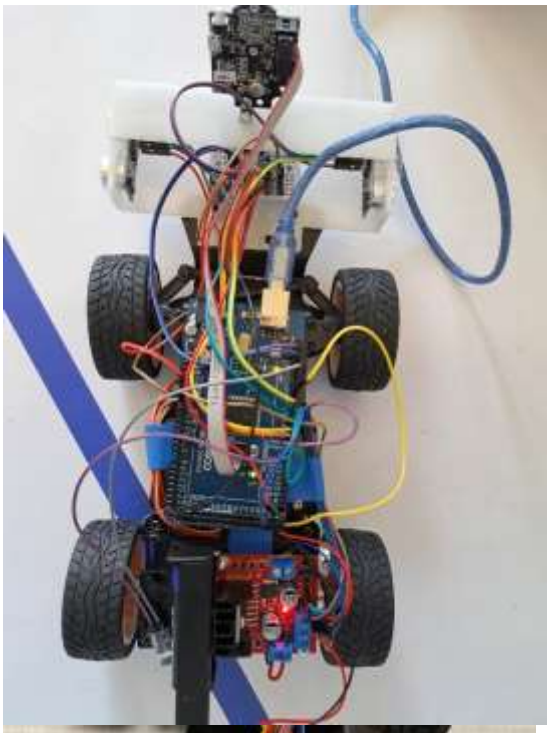
Bestimmung der Fahrtrichtung

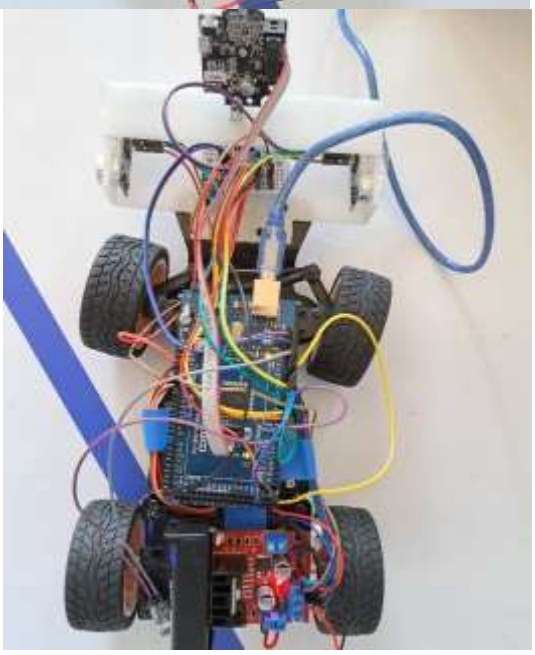
Um die Fahrtrichtung zu bestimmen, vergleichen wir am Ende der ersten Geraden den Abstand auf der rechten und linken Seite des Roboters. Daraus könne wird die Fahrtrichtung folgern: Wenn der Abstand auf der linken Seite größer als auf der rechten ist fahren wir im Uhrzeigersinn, sonst gegen den Uhrzeigersinn. Das Ergebnis aus dieser Abfrage wird in der booleschen Variabel „clockwise“ gespeichert (siehe folgender Code). Diese Information ist sehr wichtig, um die Hindernisse richtig zu umfahren und am Ende der Gerade in die richtige Richtung zu lenken.

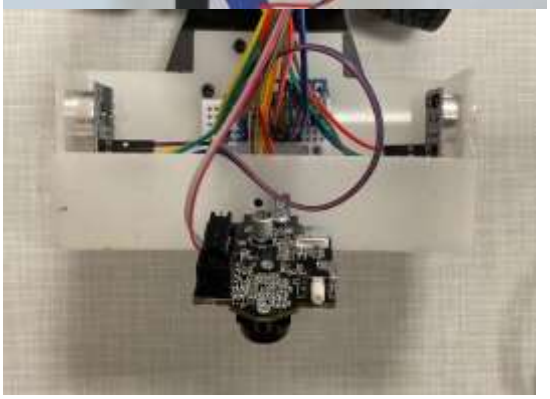
```
if (distance1 > distance3 && distance3 != 0 || distance1 == 0) {
  clockwise = false;}
  else {
  clockwise = true;}
```

3. Fotos







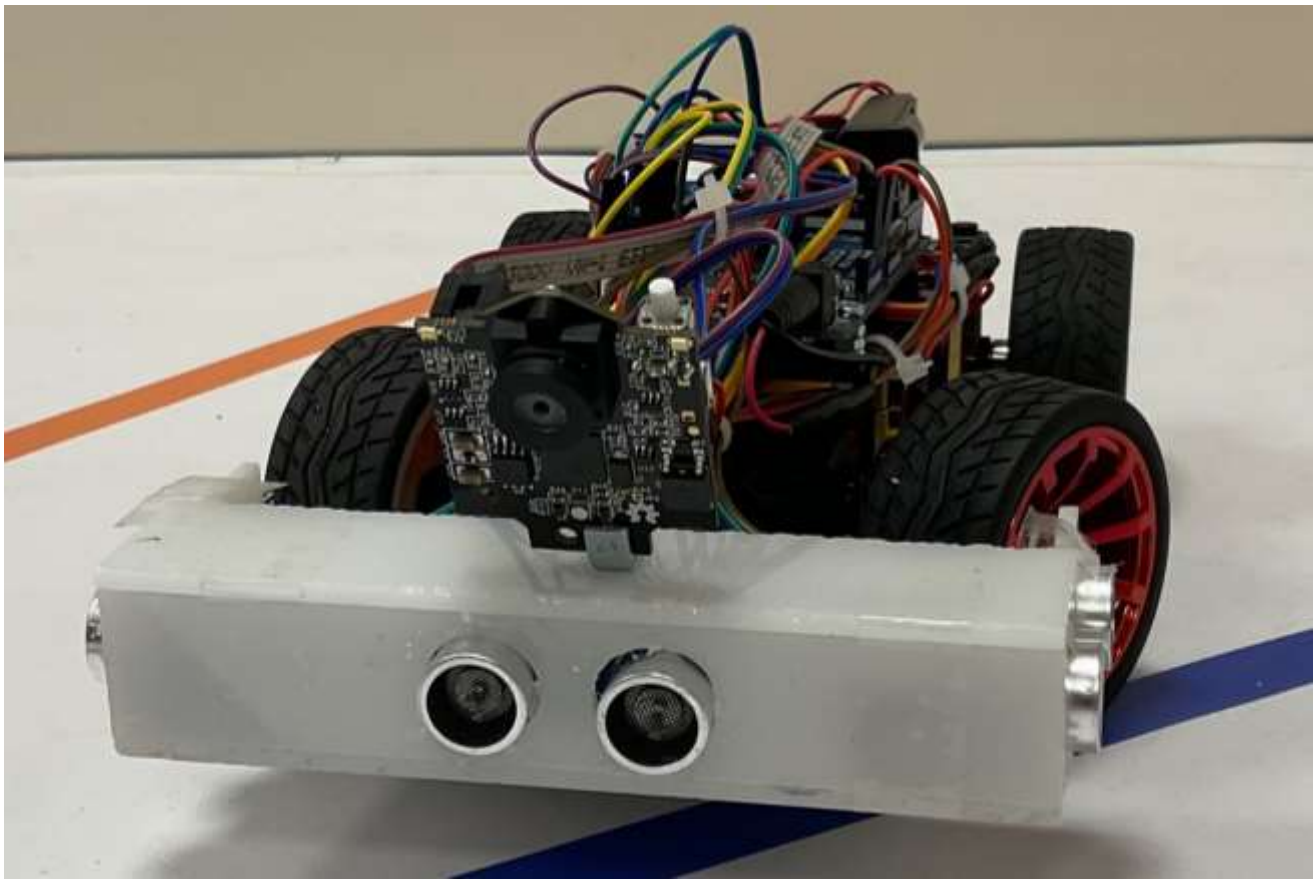


Oben links: Lenkung auf 20°

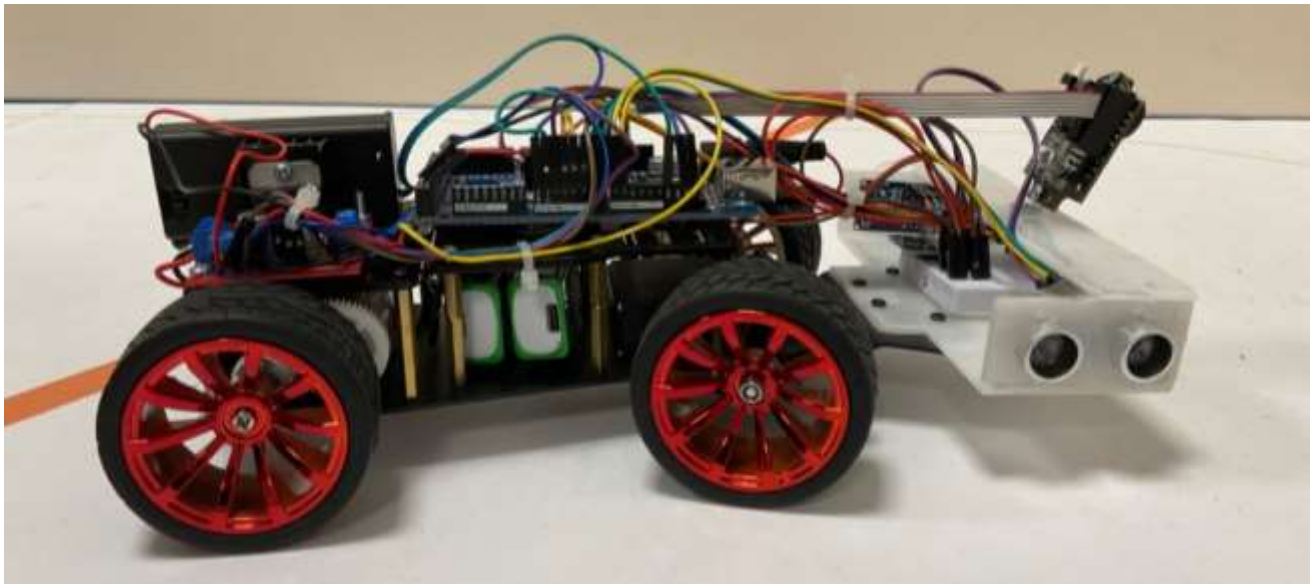
Oben rechts: Lenkung auf 50°

Unten links: Lenkung auf 80°

Unten links: Front Anbau



Roboter Vorderansicht



Roboter Seitenansicht

4. Energie/Sensoren

Bei der Sensorik haben wir uns entschieden so wenig Sensoren wie Möglich zu verwenden um möglichst energiesparend zu bleiben. Dadurch reicht uns abgesehen von der Stromversorgung des Arduinos eine weitere 9Volt-Blockbatterie sowie zwei 1.5 Volt-Batterien. Daher nutzen wir drei Sensoren-Gruppen: einen Gyro-Sensor, drei Ultraschallsensoren sowie eine Kamera.

Ultraschallsensoren

Ein Ultraschallsensor sendet Ultraschallwellen aus, welche an semtlichen Objekten reflektiert werden können und dann wieder vom Ultraschallsensor erkannt werden. Der Ultraschallsensor misst dann die Zeitspanne zwischen Aussenden der Wellen und wiedererkennen und berechnet damit den Abstand eines Objekts.

Unsere Ultraschallsensoren haben drei Aufgaben: Zum einem sollen sie dem Gyro-Sensor helfen sich mittig in der Bahn auszurichten, zum anderen sind sie für den Start des Algorithmus, welcher die Kurve fährt, verantwortlich. Des Weiteren sollen sie Kollisionen mit der Bande vermeiden. Dafür haben wir einen nach vorne, eine an der linken Seite sowie einen an der rechten Seite montiert, um einen Möglichst großen Winkel um unseren Roboter abzudecken.

Gyro-Sensor

Um den Roboter korrekt zu lenken nutzen wir einen Gyrosensor. Dieser ermöglicht und die Lenkung des Roboters präzise zu Steuern, indem wir durchgehend den Winkel, der an der Z-Achse anliegt messen und ihn mit dem gewünschten Wert vergleichen können. Wir haben uns dazu entschieden den Sensor vorne im Front Anbau zu platzieren, da sich das Breadboard dort einerseits gut zur Befestigung und zum Anschließen eignet und sich der Punkt auch relativ mittig auf der Achse des Roboters befindet.

Kamera

Besonders im Hindernisrennen liefert die Kamera uns wichtig Informationen. Für unseren Algorithmus ist es daher sehr wichtig, da die Kamera einen guten Überblick über die komplette Gerade hat. Deshalb haben wir sie direkt auf dem Front Anbau mittig platziert. So ist sie auch etwas erhöht, was uns einen noch besseren Überblick gibt. Als Kamera nutzen wir die Pixy Cam 2, da sie das Bild für uns verarbeitet. Dadurch sparen wir Rechenleistung auf dem Arduino und die Programmierung wird deutlich erleichtert.

Energie-Management

Als Stromversorgung für unseren Roboter haben wir uns für zwei getrennte Stromkreise entschieden, wobei wir mit einem den Arduino betreiben, mit dem anderen das Motor-Shield bzw. den Antriebs- und den Servomotor. Im Stromkreis des Motor-Shields haben

wir zudem noch zwei 1.5-Voltbatterien in Reihe geschalten, um auf die maximal möglichen 12 Volt des Motor-Shields zu kommen. Zudem haben wir in beiden Stromkreisen noch einen Kippschalter verbaut, mit dem wir den Roboter „scharf stellen“.

5. Engineering/Design

Da wir dieses Jahr das erste Mal dabei sind haben wir uns für das, von der WRO gesponserte, Funduino-Set entschieden, dieses allerdings an einigen Stellen modifiziert und angepasst.

Chassis

Beim Chassis verwenden wir das 4WD Chassis mit Servo-Lenkachse von Funduino. Mit den vielfach gebohrten Löchern konnten wir bereits vieler unserer Bauteile befestigen.

Anbauten

Um noch zusätzlichen Platz zu gewinnen, sowie eine bestmögliche Position für unsere Ultraschallsensoren zu gewähren, haben wir an der Front noch eine Erweiterung aus Plexiglas angebaut, da dieses leicht, stabil und gut zu verarbeiten ist. Diese haben wir dann an Stelle des Schaumstoffblocks befestigt und sowohl drei Ultraschallsensoren, den Gyrosensor sowie die Kamera befestigt (siehe Fotos).

Umbauten

Die übriggebliebenen Sechskant-Säulenschrauben haben wir dafür verwendet um die vordere Plastikplatte höher zu setzen. Dies hat für uns zwei Vorteile: zum einen kann der Arduino etwas über die Platte hinausragen, ohne an der hinteren Plastikplatte anzustoßen. Zum anderen haben wir so genügend Platz um unter dem Arduino noch zwei 9Volt-Blockbatterien zu platzieren (siehe Fotos). Des Weiteren haben wir eine Holzplatte, mit eingelassenen Schaltern, hinter dem Antriebsmotor angebaut. Über diese können wir getrennt Strom auf den Arduino und/oder auf das Motor-Shield geben.

Probleme

Vor allem bei der Platzierung des Arduinos haben wir lange nach einer Lösung gesucht um sowohl an alle Pins als auch an alle weiteren Anschlüsse zu gelangen. Zudem hatten wir das Problem, dass der Arduino seine Bohrlöcher teilweise so dicht an Lötstellen hat, dass wir nicht alle Löcher benutzen konnten. Letztendlich haben wir uns

dazu entschieden mit nur zwei, diagonal liegenden Schrauben, den Arduino zu befestigen. Dies funktioniert allerdings sehr gut weshalb wir gar nicht auf mehr Schrauben angewiesen sind. Ein weiteres Problem stellte die Befestigung unserer 9V-Blöcke dar, da wir diese fest am Roboter, gleichzeitig aber auch schnell austauschbar haben wollten. Nach einigen Versuchen mit Kabelbindern und Klebeband sind wir letztlich zur Lösung mit Klettaufklebern gekommen. Auch mit der Platzierung unseres Gyros haben wir lange experimentiert, da dieser möglichst mittig platziert werden muss. Final haben wir ihn nun, mittels eines kleinen Steckbretts, im vorderen Anbau platziert. Dies hat zudem den Vorteil, dass wir das Steckbrett ebenfalls zur Stromverteilung der Ultraschallsensoren nutzen können.

6. Anhang

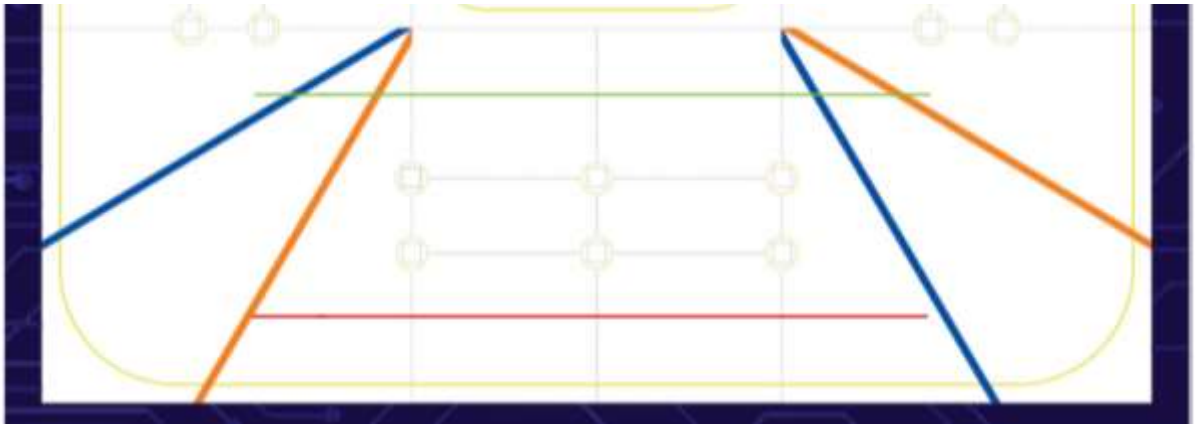


Abb.1

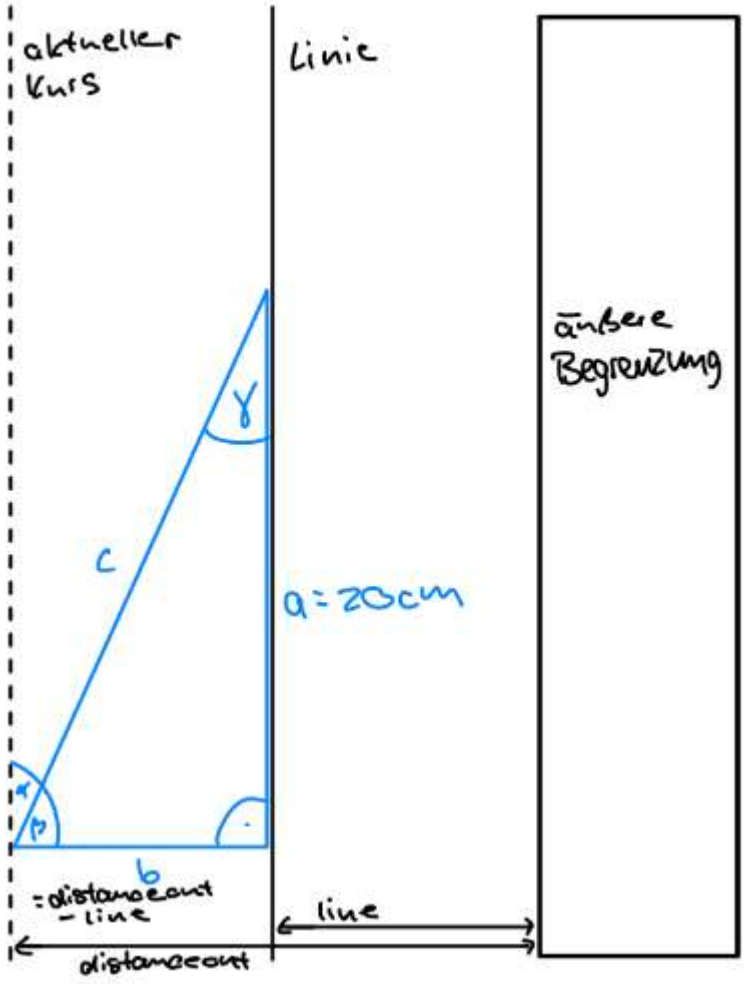


Abb.2