Arduino mit Ferbedienung Steuern & Hinderniserkennung

# Einleitung

Während ein wenig über einem Semester in Automatisation und Robotik habe ich einen Arduino mittels Fernsteuerung steuern können. Zudem habe ich eine Hinderniserkennung mit einer Basic Umfahr Strategie umsetzen können.

# Ziel meines Projektes

Ich hatte das Ziel vor Augen, einen Roboter zu bauen und programmieren, welcher mittels einer Fernbedienung und Hinderniserkennung gesteuert werden kann. Anfangs hatte ich ein Hindernissensor nur vorne. Dies habe ich erweitert für alle vier Seiten. Da ich nun auf allen Seiten ein Hindernissensor angebracht und programmiert habe, kann ich eine einfache Umfahrfunktion einbauen (Softwaretechnisch).

# Vorgehen

## Motoren

Im ersten Schritt habe ich das Arbeitsblatt «Motortreiber» durchgelesen und mich mit den Grafiken vertraut gemacht. Danach habe ich angefangen, auf dem Arduino die Kabel und den Käfer zu stecken. Dann habe ich Herrn Escher gebeten, das Steckbrett auf Fehler zu prüfen. Obwohl Herr Escher keinen Fehler entdeckt hatte, fuhr der Arduino nicht. Das Problem war die Stromzufuhr. Der Arduino hat vom MacBook den Strom bezogen. Die Stromleistung war einfach zu schwach. Mit einer 5-Volt Batterie sah die Sache anders aus. Der Arduino ist gefahren. Softwaretechnisch habe ich fixe Abläufe dem Arduino programmiert.

## Fernbedienung

Mittels der Anleitung von funduino konnte ich den Empfänger auf dem Arduino anbringen, und die übertragenen Zahlen mit Abläufen bestücke. Konkret heisst das, wenn ich auf der Fernbedienung auf die Nummer 1 drücke, fährt der Arduino beispielsweise nach vorne. So konnte ich die Funktionalität der Fernbedienung ins Skript implementieren.

## Hindernissensoren

Mittels der Anleitung von funduino habe ich den Sensor auf das Steckbrett des Arduinos gesteckt. Mit den richtigen Kabelverknüpfungen konnte ich recht schnell die Entfernung zum nächsten Gegenstand auslesen. Die Daten wurden auf dem sogenannten «Seriellen Monitor» angezeigt. Damit der Entfernungsmesser zu einer Hinderniserkennung umgerüstet werden kann, habe ich eine Bedingung im Skript erstellt. Diese lautete, wenn die Distanz zum Gegenstand kleiner als x cm ist, dann stoppe die Motoren. Somit hatte ich mit einfachen Mitteln eine Hinderniserkennung für vorne. Damit die Hinderniserkennung akkurat und präziser funktioniert, werde ich mehrere Hinderniserkennungssensoren einbauen. Am besten auf allen Seiten einen Entfernungmesssensor. Das Skript für die Hindernissensoren auf allen Seiten konnte ich von zu Hause ausschreiben. Die Hardware musste ich in der Schule stecken, da ich den Arduino nicht mit nach Hause genommen habe. Als ich das Skript zu Hause angepasst hatte, habe ich einen Pin doppelt verwendet. Daher funktionierte das Skript nach dem Übertragen auf den Arduino nicht. Zum Glück ist mir das recht schnell aufgefallen.

## Umfahrungsfunktion

# Probleme

Wie schon in den vorherigen Texten beschrieben, hatte ich ein paar Probleme, wie beispielsweise mit den Pins. Einmal hatte ich auch das Problem das ich die Pins für das Vorwärts- und Rückwärtsfahren gleichzeitig angeschalten habe. Somit konnte der Arduino nicht fahren. Zuerst dachte ich ganz lange, das dies ein Hardwarefehler von mir ist und kein Softwarefehler. Danach habe ich das Skript gründlich überprüft, dabei ist mir dies aufgefallen. Die Hardware hat mir persönlich mehr Kopfschmerzen bereitet als das Skript schreiben.

# Funktionsweise der Software (Arduino Skript)

Ganz am Anfang des Skriptes habe ich die Pins und Variablen definiert. Zudem starte ich auch den «seriellen Monitor». Danach kommen die Manöver, die der Arduino unterstützt also:

* Vorwärtsfahren
* Rückwärtsfahren
* Alle Pins ausscahlten
* Nach links drehen
* Nach rechts drehen

Nun habe ich in der Funktion loop() definiert, welcher Tastendruck auf der Fernbedienung, welche Funktion aufrufen soll. Danach kommen die Konfigurationen für die Hindernissensoren vorne, hinten und auf den Seiten. Damit das Skript gewisse Abläufe nicht mehrmals drin haben muss und somit das Skript kleiner wird, habe ich für jedes Manöver eine eigene Funktion erstellt.

# Funktionsweise der Hardware