Bericht zum autonomen Autoprojekt Lego NXT für den Uni Freiburg System Design Projekt Kurs, Wintersemester 21/22

Kyle Stevens

28 Januar 2022

1 Einführung

Unter Verwendung des Lego NXT-Designkits und der Programmiersprache NXC führte ich die Aufgabe aus, ein autonomes Auto zu bauen und zu programmieren, das einen durch eine schwarze Linie dargestellten Pfad bewältigen konnte, der gekrümmten Linie folgte, an einem Tor anhielt und den Pfad bewältigte Aufteilen und am Ende aufhören. Neben verschiedenen Legosteinen und Rädern habe ich den NXT-Stein, zwei der drei mitgelieferten Motoren, zwei der drei mitgelieferten Lichtsensoren und den Ultraschallsensor verwendet. Die gesamte Bau- und Programmierzeit betrug ungefähr 25 Stunden.

2 Konstruktion

Ich beschloss, mich vor der Programmierung auf die Konstruktion des Fahrzeugs zu konzentrieren. Ich beschloss, das Chassis und die Lenkung des Fahrzeugs zu bauen, bevor ich die NXT-Basis und die Sensoren anbrachte. Hier stellte sich das erste Problem: Wie man die Motoren so einstellt, dass sie nicht nur die Räder vorwärts und rückwärts antreiben, sondern auch zum Lenken.

2.1 Räder, Motoren und Lenkung

Zuerst überlegte ich, ein Differentialgetriebe zu bauen, um die Lenkung zu erleichtern. Mein NXT-Kit wurde mit einem Differentialgehäuse geliefert (Teil: 4525184). Ich hatte jedoch nicht die Ausrüstung, um es zu füllen, also habe ich diese Idee missachtet. Die Lenkräder müssten auf getrennten Achsen sitzen. Ich habe drei Designs für die Lenkung ausprobiert:

2.1.1 Ein Heckmotor und ein Frontmotor, beide vertikal

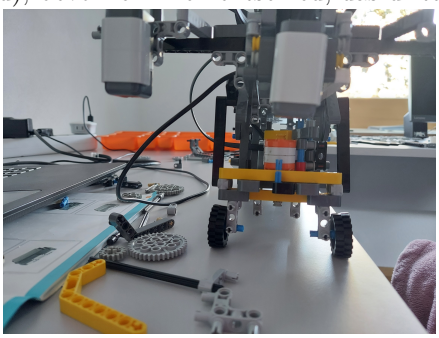
Diese Anordnung wäre symmetrisch gewesen, aber als ich versuchte, sie zu implementieren, war sie problematisch. Insbesondere der vordere Motor hätte zum

Lenken seine Kraft über mehrere Gänge übertragen müssen. Siehe das Bild für ein sehr grundlegendes Konzept.

Aufgrund des Platzes und Spiels zwischen den Zahnrädern war es unmöglich, die Bewegung genau zu steuern, also wurde darauf verzichtet.

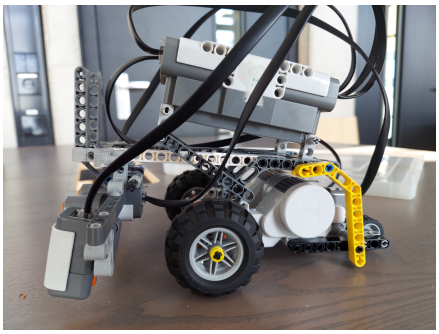
2.1.2 Ein hinterer Motor und ein vorderer Motor, wobei der vordere Motor horizontal ist

Dieses zweite Design hatte eine präzisere Kontrolle über die Räder, obwohl es asymmetrisch war. Es gab jedoch ein Problem bei der Einstellung der Lenkung, in der Lage zu sein, nach einer scharfen Rechts- oder Linkskurve wieder in eine Geradeausfahrt zurückzukehren. Ich baute das Auto mit diesem Design (siehe Bild), bevor ich mich entschied, das dritte Design auszuprobieren.



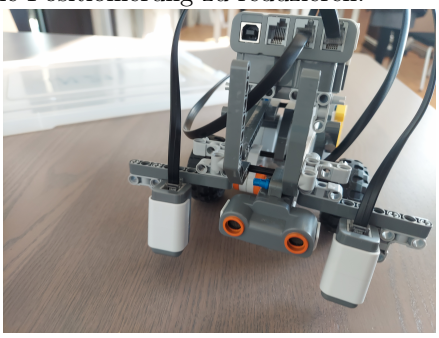
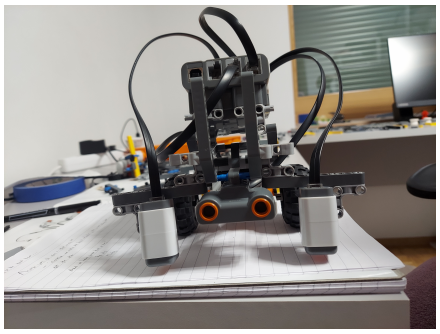
2.1.3 Zwei Motoren auf derselben geometrischen Achse

Dies war das erfolgreichste Design. Die beiden Motoren konnten zusammenarbeiten, um sich vorwärts und rückwärts zu bewegen, und wenn das Auto wenden musste, konnte sich ein Motor vorwärts bewegen, während sich der andere leicht rückwärts bewegte, um es auszugleichen. Mit mehr Zeit hätte man das besser ausgleichen können, aber es hat trotzdem gut funktioniert. Siehe die Bilder.



2.2 Sensorplatzierung

Zuerst habe ich alle drei Lichtsensoren hintereinander verwendet, bevor mir klar wurde, dass der mittlere überflüssig war, sodass nur zwei zusammen mit dem Ultraschall verwendet wurden. Ich habe versucht, die Sensoren so nah wie möglich am Boden zu positionieren, um mögliche Fehler durch Licht oder zu hohe Positionierung zu reduzieren.

3 Programmierung

3.1 Vorwärtsbewegung

Dies wurde leicht erreicht, indem die Motoren so programmiert wurden, dass sie gleichzeitig vorwärts ausgaben.

3.2 Lichtkalibrierung

Die Lichtkalibrierung wurde erreicht, indem Eingaben von beiden Lichtsensoren genommen wurden, ihre Mittelwerte genommen wurden und alle Werte über einem Mittelwert gesetzt wurden, um als hell gelesen zu werden, und alle Werte darunter, um als dunkel gelesen zu werden. Dies war meiner Meinung nach einfacher als die vorgeschlagene Kalibrierungsmethode im Powerpoint-Kurs. Für eine zusätzliche Sicherheitsebene habe ich auch die Werte der beiden Sensoren

und den Durchschnitt auf dem Display ausgegeben. Das würde mir später ersparen, da ich versehentlich zuerst die Lichter mit beiden Sensoren auf schwarze Balken kalibriert habe.

3.3 Drehen

Das Wenden wurde erreicht, indem ein Motor/Rad vorwärts bewegt wurde, während sich der andere Motor/Rad mit verringerter Geschwindigkeit rückwärts bewegte.

3.4 Anhalten für das Tor

Das war ganz einfach. Wenn der Ultraschall irgendetwas innerhalb von 25 cm entdeckte, unterbrach er sofort die Schleifen-Hauptfunktion.

3.5 Behandlung der Pfadaufteilung

Ich habe mich nicht besonders bemüht, zu programmieren, wie das Auto mit der Pfadaufteilung umgeht. Ich hoffte einfach, dass das Auto weiter einen Weg hinunterfahren würde. Meine größte Sorge war, dass sich das Auto komplett dreht oder vom Kurs abkommt. Dies geschah jedoch nicht.

3.6 Bremsen am Ende

Ich habe das Auto so programmiert, dass es stoppt, nachdem ich eine bestimmte Anzahl von Fällen registriert habe, in denen beide Sensoren gleichzeitig Schwarz erkannt haben. Das war riskant, weil ich nicht vorhersagen konnte, ob es zu viele solcher Fälle vorzeitig erkennen würde, zum Beispiel zwei schwarze Balken erkennen, wenn sich die Pfade teilen.

4 Wettbewerbahn

Zuerst gab es, wie bereits erwähnt, einen Fehler meinerseits, indem ich die Sensoren mit beiden auf schwarze Balken kalibrierte. Nachdem dies korrigiert wurde, fuhr das Auto fehlerfrei, wenn auch langsam weiter. Es hielt am Tor an, navigierte durch die Wegteilung und hielt am Ende des Weges an. Mit mehr Zeit hätte es ein effizienteres Design in Bezug auf die Wendegeschwindigkeit und ein sichereres Design in Bezug auf die risikoärmere Handhabung der Wegteilung und des Endstopps geben können.