„Lagemaro“

(Laser-gesteuerter Malroboter)

Gruppenmitglieder: Henrika Runde, Philipp Kückes, Julian Stöhr, Benjamin Schnack

Der Lagemaro ist ein Roboter, der die kreativen Gedanken eines Menschen verwirklicht. Eine Kamera und eine fahrbare Mal-Mechanik sind seine Hilfsmittel. Frei nach dem Motto: „*Malen nach Strahlen*“!

Gliederung

1.Einleitung

Das Projekt „Lagemaro“ besitzt zwei Hauptbestandteile. Einerseits ist das die Kamera und andererseits die fahrbare Stiftvorrichtung, also der Roboter selbst.
Die Kamera fungiert als „Auge“ und soll den Roboter, Laser und die einzelnen Abstände im Malbereich erkennen. Über Processing werden die übermittelten Informationen in mathematisch sinnvolle Werte umgerechnet. Des Weiteren kann das Programm die gezeichnete Linie des Laserpunktes visualisieren und ebenfalls als Daten speichern. Anschließend sendet der PC die jeweiligen Daten und Werte per W-LAN an den Roboter.
Nun muss der Roboter die empfangenen Daten mithilfe von mathematischen Formeln in Bewegungsanweisungen umrechnen. Diese Anweisungen führt der Roboter aus und malt zusätzlich die von dem Menschen gezeichnete Linie nach. Das geschieht mit einer vertikal beweglichen Stiftvorrichtung, die durch einen Servomotor angetrieben wird.

2. Bauteil

2.1 Kamera

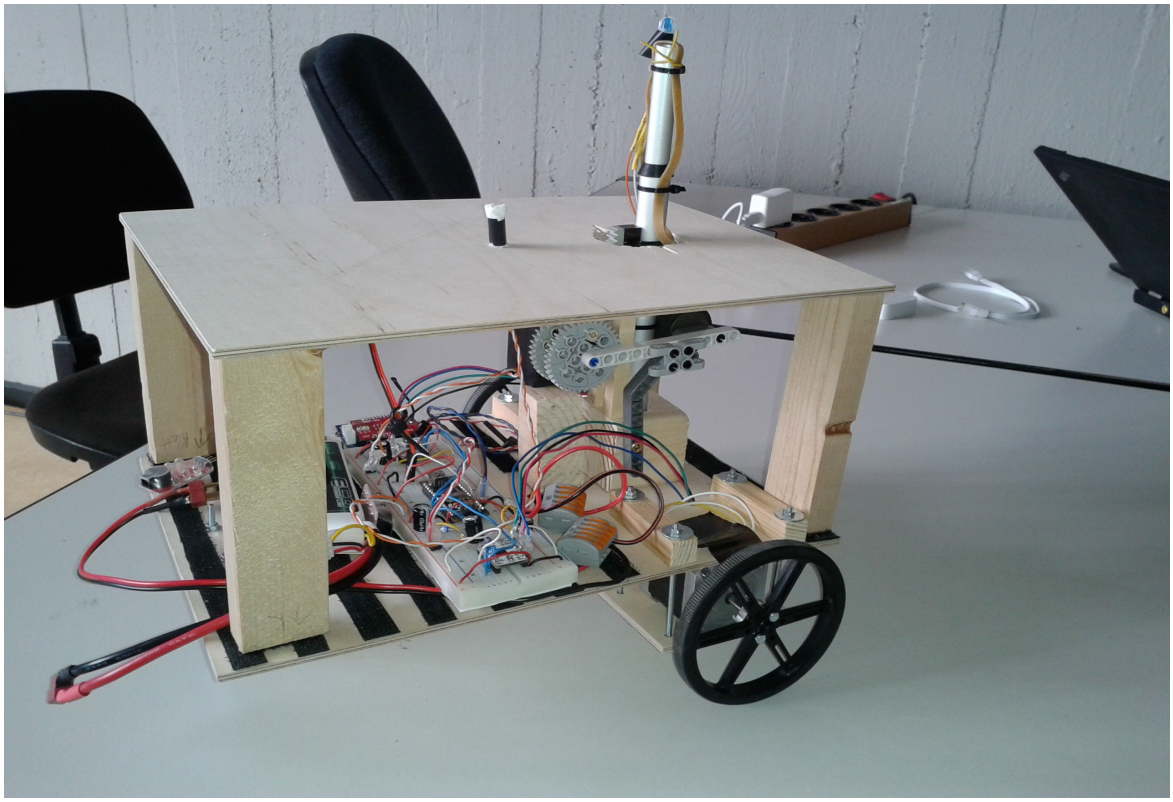
Zur Verwendung kam eine PS3-Kamera. Diese wurde an einer Holzlatte befestigt. Mithilfe einer gekürzten Gewindestange wird die Holzlatte auf einem Kartenständer eingehängt. Außerdem wird das kürzere Ende der Holzlatte mit einer Schnur an dem Fuß des Kartenständers fixiert. Am längeren Ende der Holzlatte wurde die Kamera befestigt, ohne ihre Dreheigenschaften zu beeinträchtigen. Die Kamera ist durch ein ausreichend langes USB-Kabel mit einem Laptop verbunden. Das gesamte Gerüst ist höhenverstellbar.

2.2 Roboter

2.2.1 Karosserie

Eine Holzplatte in der Größe DINA 4 dient als Grundfläche des Roboters.
Auf der Unterseite dieser Platte bildet eine 6.5 cm hohe Wagenrolle, das Vorderrad. Ideen wie ein Kugelrad oder ein abgeschliffener Holzklotz wurden wegen zu hoher Reibung verworfen.
Hinzu kommt ein Dachgerüst, welches ebenfalls eine DINA 4 Holzplatte ist und mit etwa 10 cm hohen Stelzen auf der Grundplatte steht. Dieses Dachgerüst soll störende Lichteinflüsse vom z.B. Arduino abdecken. Alle Komponenten die keinen großen mechanischen Belastungen ausgesetzt sind, wurden mit Klett auf der Grundplatte befestigt (Akku, Dach, Breadboard, W-Lan Modul).

Bild: Vollständiger Roboter mit Dach



2.2.2 Räder

Hier kamen zwei Steppermotoren zum Einsatz, da ein präzises Anfahren der Punkte erforderlich ist. Die Steppermotoren wurden zwischen der Grundplatte und jeweils einer kleineren Hilfsplatte (ebenfalls aus Holz) durch Gewindestangen eingeklemmt. Industriell vorgefertigte Profile waren zu teuer.

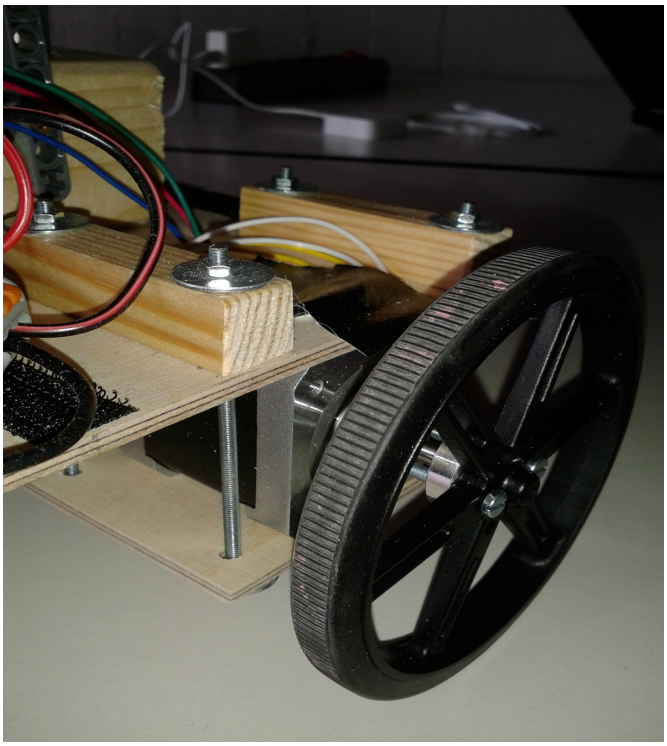


Bild: Antrieb (Steppermotor)

Die beiden Steppermotoren wurden jeweils an einen Pololu-Treiberchip angeschlossen. Der Pololu regelt die Spannung und die Stromzufuhr für den Steppermotorbetrieb. Die benötigte Spannung beträgt 8 - 35 V, die ein Akkumulator liefert. Eine genaue Betriebsanleitung findet man unter http://www.pololu.com/product/1182

2.2.3 Stiftvorrichtung

Als Orientierung diente die Konstruktion einer Gruppe aus dem Vorjahr. Übernommen wurde hierbei der Antrieb (Servomotor) und die Verbindung zwischen Stift und Motor (Lego). Ein Nachteil dieser Idee war jedoch die fehlende Federung. Die Lösung dieses Problems waren zwei ineinander gesteckte Aluminiumrohre (15mm beträgt der Durchmesser des größeren Rohres). Ein Haushaltsgummi dient bei dieser Bauweise als Federung.

Außerdem kam ein Holzblock mit einem 16mm-Loch und darauf zwei schmale Holzstege, als Halteführung zum Einsatz. Ein weiterer Faktor der zur Stabilität beiträgt, ist ein Stützrad, das auf der gegenüberliegenden Seite des Servozahnrades als Konterstück angebracht wurde. Der Durchmesser der Aluminiumrohre wurde so gewählt, dass man unterschiedliche Kreide- und Stiftgrößen einsetzen kann. Zum Schluss wurde eine Aussparung in das Dach gesägt.

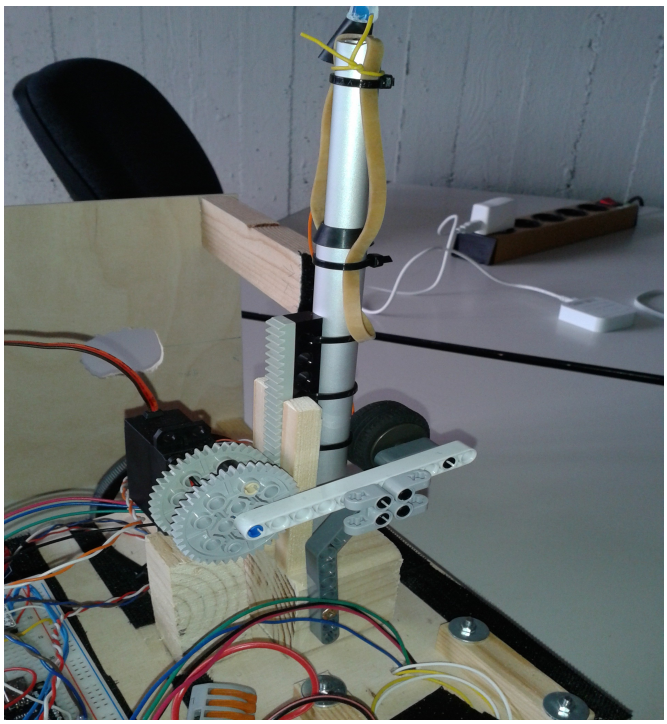


Bild: Stiftkonstruktion

2.2.4 Akku, W-Lan, LED's

Als Stromversorgung für den Roboter kommt ein 7,5 V leistender Lithium-Ionen-Akku zum Einsatz. Dieser ist mit Klett an der Grundplatte befestigt und lässt sich so leicht abnehmen.

Um eine Verbindung zwischen Roboter und PC herzustellen, verwenden wir einen W-Lan-Chip (RN-XV).

Um die Ausrichtung des Roboters für die Kamera erkennbar zu machen, war es sinnvoll zwei LED's mit unterschiedlichen Farben zu benutzen (gelb und blau). Die gelbe LED steht für die Front des Roboters und ist in der Mitte des Daches platziert. Die blaue LED gibt die Stelle des Stiftes an und wurde deshalb auf der Spitze der Stiftkonstruktion angebracht.

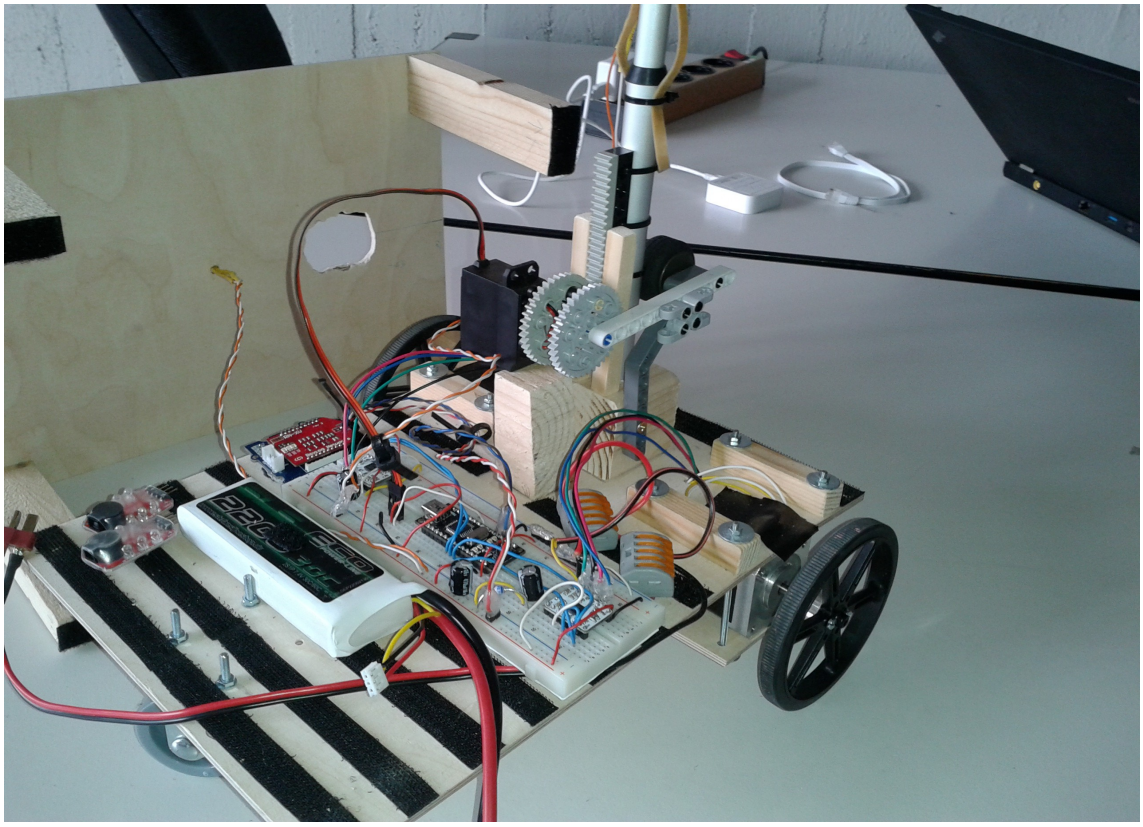


Bild: Roboter ohne Dach (darauf Breadboard, Akku, WLan)

3. Programmierteil

Kamera

Bildanalyse

Jeder Pixel besteht aus einer 4 Byte großen binären Zahl, wobei das zweite Byte den Rotanteil, das dritte den Grünanteil und das vierte den Blauanteil der Pixelfarbe festlegt. Das erste Byte legt die Transparenz fest, die aber sowohl konstant, als auch irrelevant für die Farbe des Punktes ist.

Um die einzelnen Farben zu isolieren, kann man zuerst eine bitweise Verschiebung durchführen, so dass das Byte mit den Informationen für die jeweilige Farbe an letzter Stelle steht. Um alles was an anderen Stellen in dem Pixel steht auf null zu setzen, kann man eine und-Verknüpfung mit der binären Zahl 0000 0000 0000 0000 0000 0000 1111 1111 , was das gleiche ist, wie die Zahl FF im Hexadezimalsystem (0xFF).

Im Code sieht das dann zum Beispiel für rot folgendermaßen aus:

```
rot= pixel>>16&0xFF;
```

(im folgenden wird die Erklärung sich auf die Farbe rot beschränken. Die vorgehensweise für die anderen Farben ist jedoch die gleiche)

Im ersten Ansatz wird der „roteste“ Punkt dadurch definiert, dass die Differenz zwischen dem Rotanteil und dem Grün- und Blauanteil in diesem Punkt am größten ist.

Um diesen Punkt zu finden wird jeder Pixel der Reihe nach mit dem höchsten bisherigen Wert verglichen. Wenn der Pixel einen höheren Rotwert hat, wird dieser Höchstwert durch den aktuellen ersetzt und die Koordinaten werden abgespeichert. Wenn der Höchstwert des ganzen Bildes unterhalb eines festgelegten Wertes liegt wird keine rote LED/ kein Laser erkannt. Ansonsten wird der Punkt in dem dieser Höchstwert erstmals erreicht wurde als rote LED /

Laser erkannt. Bevor ein neues Bild durchgegangen wird, wird der Höchstwert wieder auf 0 gesetzt, da sonst der alte Höchstwert auch der neue ist, falls es im neuen Bild keinen höheren Wert gibt.

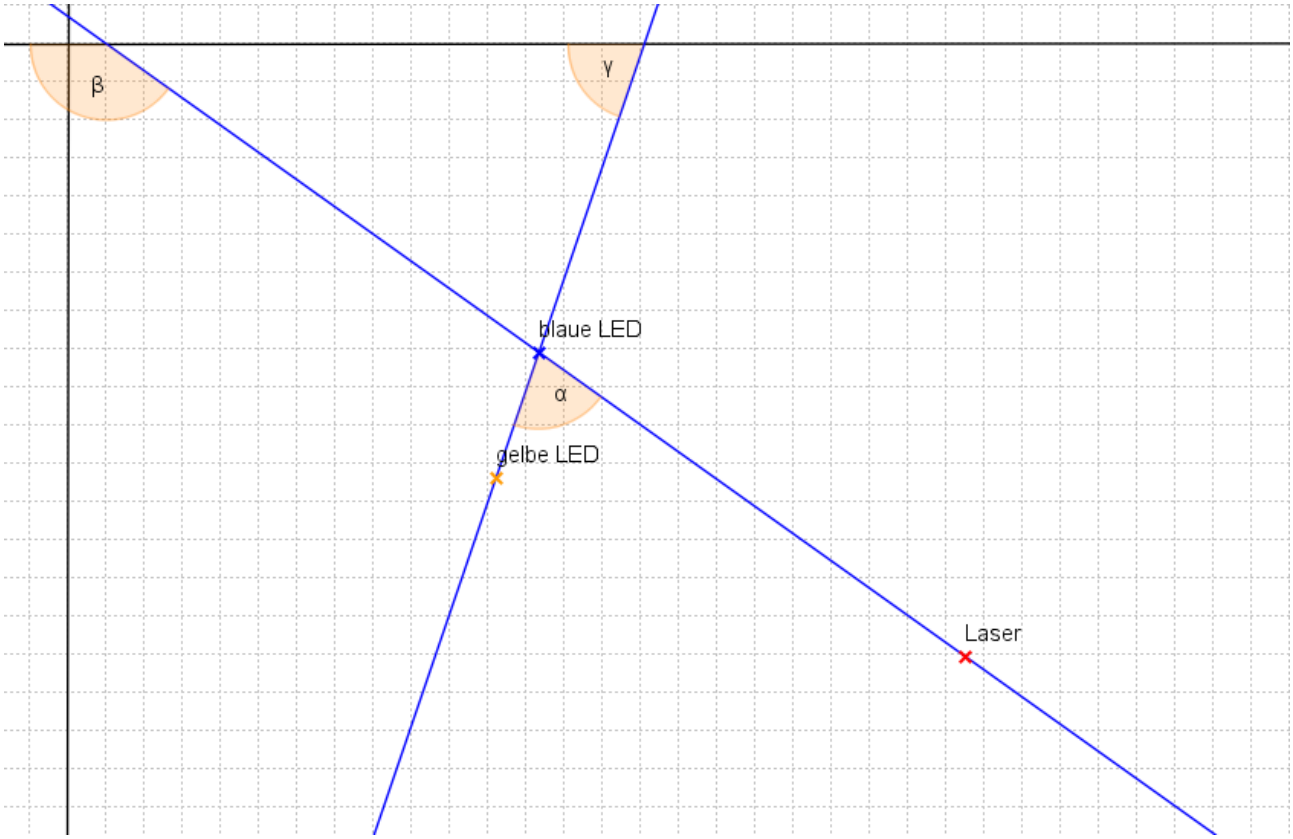
Die Verwendung des HSV-Farbsytems war ein Versuch, die Erkennung der gelben LED zu verbessern, weil gelb eine Mischung aus rot und grün ist und daher nicht so einfach mit dem Algorithmus erkannt werden kann, wie die reinen Farben. Das hat aber auch nicht viel besser funktioniert, als das vorherige System, weil auch hierbei kein Vergleich mit der tatsächlichen LED sondern mit einer Art Idealwert der LED stattfindet.

Das beste System um alle LEDs möglichst gut zu erkennen, ist ein Vergleich mit dem Wert der auf der LED gemessen wurde. Dabei ist es eigentlich egal welches Farbsystem man verwendet. Hier wird das RGB-Farbsystem verwendet. Um die LEDs auch bei unterschiedlichen Lichtverhältnissen erkannt werden können, wird der Durchschnitt über dem Gesamten Bild von den Werten an den einzelnen Pixeln abgezogen. Allerdings funktioniert das nicht gut genug, damit man mit einem einmal nachgemessenen Wert bei allen Lichtverhältnissen die LEDs erkennen kann. Daher wird vor der Suche eine Kalibrierung der Werte durchgeführt, bei der man die LEDs einmal per Hand anklickt. Das hat auch den weiteren Vorteil, dass man die LEDs beliebig austauschen kann, ohne etwas am Programm ändern zu müssen.

Winkel und Abstand

Distanz: Die Distanz kann ganz einfach mittels des Satzes von Pythagoras berechnet werden.

Winkel:



Die Winkel beta und gamma können mit Hilfe des Arcus Tangens berechnet werden:

$$\tan(beta) = \frac{Gegenkathete}{Ankathete}$$

$$beta = \arctan\left(\frac{Gegenkathete}{Ankathete}\right)$$

$$beta = \arctan\left(\frac{(X_{Laser} - X_{blaue\ LED})}{(Y_{Laser} - Y_{blaue\ LED})}\right)$$

$$gamma = \arctan\left(\frac{(X_{gelbe\ LED} - X_{blaue\ LED})}{(Y_{gelbe\ LED} - Y_{blaue\ LED})}\right)$$

$$alpha = beta - gamma$$

Das Problem, welches bei dieser Rechnung auftritt, ist, dass die Richtung in der der Roboter steht, in die Winkelberechnung nicht betrachtet wird. Es gibt bei Proccesing die Funktion tan2, die genau diese Richtung mit berücksichtigt und einen Winkel zwischen -π und π ausgibt. Damit immer der kleinere Winkel gefahren wird, wird das Ergebnis von π abgezogen (oder falls es negativ ist von -π), falls der Betrag des Winkels größer als π ist.

Kommunikation mit dem Arduino

Die Daten über Winkel, Distanz und Stift werden über W-LAN an den Roboter gesendet. Dabei werden die Zahlen in int- oder float-Variablen gespeichert, die dann, mit Hilfe der importierten oscp5 Bibliothek, an eine festzulegende IP-Adresse in dem W-LAN-Netzwerk gesendet wird. Die IP-Adresse, die der Arduino hat, kann man sich über den serial monitor anzeigen lassen, wenn dieser sich mit dem W-LAN verbindet. Das W-LAN-Modul empfängt die gesendeten Variablen die ganze Zeit über und übermittelt diese auf Anfrage an den Arduino.

Route

Die Route wird in einer Reihe von Punkten in einer Arraylist gespeichert. Der Vorteil dabei ist, dass im Gegensatz zu einem normalen Array die Anzahl der Elemente nicht vorher festgelegt werden muss. Wenn also ein Laserpunkt erkannt wird, wird der Abstand zum vorherigen Punkt berechnet. Nur wenn dieser größer ist als ein vorher festgelegter Wert (oder falls es noch keinen vorherigen Punkt gibt), wird der Punkt an die Arraylist angehängt, ansonsten passiert nichts, damit die Punkte nicht näher aneinander liegen, als es bei der

Genauigkeit der Messungen sinnvoll ist.
Um die Route auf dem Bildschirm darzustellen werden immer zwei aufeinander folgende Punkte durch eine Linie verbunden.
Die Route wird Punkt für Punkt abgefahren. Sobald der erste Punkt existiert, werden Winkel und Distanz berechnet, die der Roboter Fahren muss um zu diesem zu gelangen. Wenn der Abstand zu diesem Punkt gering genug ist wird überprüft, ob es einen nächsten Punkt in der Arraylist gibt und falls es einen gibt, wird dieser angefahren.
Die Genauigkeit mit der der Roboter die Route abfährt hängt vor Allem davon ab, wie nah die aufeinanderfolgenden Punkte beieinander liegen.

Roboter

Teilprobleme des Programms

Problem der Kommunikation

Ein großes Problem bei der Kommunikation war es die einzelnen Programmbausteine zu verstehen um auftretende Fehler im Programm oder der Library zu finden und zu beheben. Oft sind Fehler nur kurzzeitig aufgetreten, was die Fehlersuche stark erschwerte.
Ein weiteres Problem, welches die Library mit sich brachte, war der im Vergleich zum Speicherplatz des verwendeten Arduinomodells enorme Speicherplatzverbrauch.

Winkel und Strecken abfahren

Um den Roboter möglichst gut fahren zu lassen, musste das Programm immer wieder verbessert oder komplett erneuert werden, da oft bestehende Teile zwar funktionierten, jedoch keine Verbesserungen ohne radikale Änderungen mehr zuließen.

Beschreibung der einzelnen Systembestandteile

Es werden Daten über W-Lan empfangen und ausgewertet oder wenn keine neuen Daten empfangen, aber eine alte Route noch nicht komplett abgefahren wurde, wird diese weiter verfolgt. Sind also neu Daten eingetroffen wird ein neuer Weg eingeschlagen und wenn keine neuen Daten existieren wird lediglich der alter Kurs weiter verfolgt.

Kommunikation

Für die Kommunikation über W-Lan mussten sowohl die bereits existierenden Programmbauteile angepasst und implementiert, als auch die entsprechende Library.

Bei der W-LAN-Kommunikation wird zuerst eine Verbindung mit dem jeweiligen W-LAN-Netzwerk aufgebaut und auf den Empfang der Werte mit der im Programm festgelegten Variablennamen gewartet. Die zu empfangenen Werte sind die Distanz, der Winkel des Roboters in Abhängigkeit zum anzufahrenden Punkt und der Position des Stiftes.

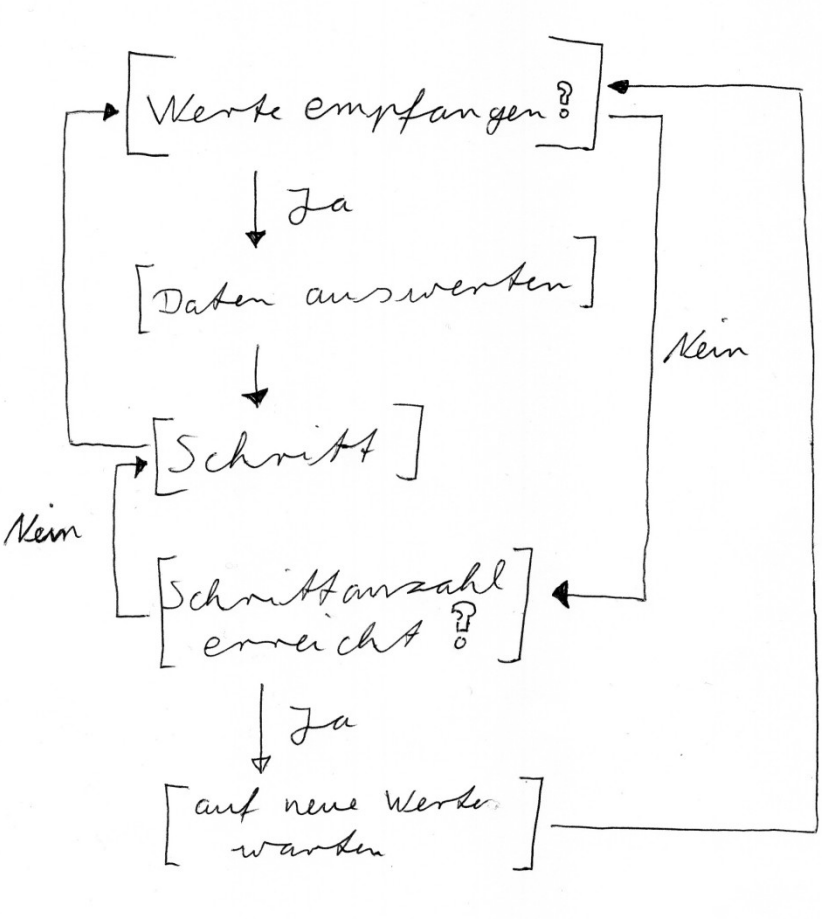
Sobald neue Werte empfangen wurden, werden die Variablen, welche für die Kursberechnung benötigt werden neu beschrieben.

Fahren

Um den Roboter fahren zu lassen, müssen zuerst verschiedene Werte aus den empfangenen Daten berechnen Werden.

Anhand der Distanz in Zentimetern wird zuerst die Distanz in Anzahl der Einzelschritte errechnet und anschließend um ein Maß entsprechend der Größe des empfangenen Winkels verringert. Eine winkelabhängige Distanz verbessert die Fahrt in so fern, dass bei einem besonders großen Winkel die Distanz sehr klein wird, was den Drehradius deutlich verbessert und den Roboter genauer fahren lässt. Eine weitere Variable ist die Anzahl der Steps, welche für die Drehung zum Zielpunkt über den Winkel ermittelt und zur Bestimmung der Einzelstepzahlen der beiden Steppermotoren verwendet werden. Ist ein neues Wertepaar empfangen oder eine alte Strecke noch nicht komplett abgefahren worden, kann ein neuer Schritt in Richtung des Ziels gemacht werden. Hierzu werden in Abhängigkeit von Winkel und Distanz die Abstände zwischen den einzelnen Schritten in Mikrosekunden berechnet, um die Strecke in Form einer Kurve abfahren zu können. Über die interne Uhr des Arduinos können anschließend die Links- und Rechtsschritte getimt werden. Dieser Aufbau hat den Vorteil, dass nach jedem Schritt über neue Daten der Kurs geändert werden kann und auch bei nicht empfangenen Daten der Kurs entsprechend der letzten Daten weiter angefahren wird.

Funktionsweise des Fahrprogramms



Werden neue Werte empfangen können die Daten ausgewertet und somit eine Route berechnet werden. Anhand der Daten kann nun ein Schritt in die entsprechende Richtung gemacht werden.
Anschließend wird wieder auf neue Werte überprüft. Sind keine neuen Werte gekommen und die Schrittanzahl der Route noch nicht erreicht wird wieder ein Schritt getätigt.
Nur wenn das Ziel anhand der Schrittanzahl erreicht ist aber keine neuen Daten empfangen werden, muss gewartet werden bis diese eintreffen.

Stiftsteuerung

Wird ein Befehl empfangen, welcher den Stift in eine neue Position bringen würde und nicht der momentanen Position des Stiftes entspricht, wird der Stift in die neue Position durch Ansteuerung des Servomotors gebracht.

4. Ergebnis und Diskussion

Aufgrund der Einklemmung der Steppermotoren verbiegt sich die Grundplatte ein wenig. Außerdem sind die Stützen des Daches leicht schief geschnitten worden. Diese Mängel stören den Betrieb des Roboters aber nicht. Das Gerüst hält die mechanischen Belastungen aus.
Die Anbringung der Steppermotoren hält ausreichend gut. Ein kleines Problem liegt darin, dass sich die Gewindestangen, Muttern aber auch die Räder bei zu hohen Geschwindigkeiten leicht lösen. Eine Verbesserungsmöglichkeit hierzu wäre ein spezieller „Schraubenkleber“ den man an die einzelnen Problemstellen auftragen könnte.
Die Idee der Stiftkonstruktion ist auch aufgegangen. Eine leichte Instabilität beeinträchtigt die Funktion nicht.
Insgesamt sind wir mit der Konstruktion des Roboters zufrieden.

Das Programm ermöglicht es dem Roboter ein Bild mit schönen Kurven nachzufahren und sich dabei immer auf die neu eintreffenden Daten einzustellen. Es erlaubt dem Roboter jedoch nicht Hindernissen auszuweichen oder bei Verlassen des Sichtfeldes der Kamera zurückzukehren.
Über ein extra Signal, welches dem Programm bescheid gibt, wenn sich der Roboter aus dem Bild bewegt, könnte eine Rückwärtsfahrt in Gang gebracht werden bis neu Information über die genaue Position des Roboters erkannt werden. Das Problem des Hindernisausweichens kann entweder mittels Entfernungsmessers oder Berührungssensoren und einer entsprechenden Ausweichprozedur.