

*Montage und Programmierung
eines Roboters für
ROBOCUP JUNIOR RESCUE
mit Elegoo Car Kit
Teil 2.10: Drive Analogue*

Von Charlotte und Andreas

Line

Maze

Ab jetzt gibt es von mir nur noch Hinweise! Ab jetzt fängt eure Arbeit erst richtig an!
Jetzt müsst **ihr** programmieren und experimentieren.

Kommentiert den unteren Teil „Maze“ aus oder löscht ihn aus dem Programm.

Legt in der Funktion `Data_Visualizer` die Werte und deren Anzeigenreihenfolge fest. Ihr braucht z.B. `Distance_1` und `-_2` nicht.

Überlegt euch, welche Werte ihr zusätzlich angezeigt bekommen wollt.

Was bewirkt das Programm?

Wie könnte die Gleichung für den „`Correction_Value`“ noch aussehen?

Änderungen programmieren, debuggen, hochladen, testen, Änderungen programmieren, debuggen...

Kommentiert den oberen Teil „Line“ aus oder löscht ihn aus dem Programm.

Legt in der Funktion `Data_Visualizer` die Werte und deren Anzeigenreihenfolge fest. Ihr braucht z.B. alle „`Line_...`“-Werte nicht.

Überlegt euch, welche Werte ihr zusätzlich angezeigt bekommen wollt.

Was bewirkt das Programm?

Wie nah wollt ihr an der Wand entlang fahren?

Ist 80 ein guter Wert? Sind die Parameter der Funktion „`Forward`“ gut gewählt?

Wäre es nicht sinnvoller dynamische Werte zu benutzen?

Änderungen programmieren, debuggen, hochladen, testen, Änderungen programmieren...

Line

Wenn ihr es geschafft habt, mit dem Roboter eine mäßig kurvige Strecke zu fahren, kommt der nächste Schritt:

Für diese Kacheln müsst ihr auch die digitalen Sensorwerte einsetzen:



Überlegt euch welche Werte ihr von den Sensoren bekommt, wenn ihr an eine Ecke kommt.

Wie reagiert ihr darauf? Fahrt ihr z.B. einfach ein Stück nach rechts oder links?

Oder gibt es noch andere Möglichkeiten?

Wie bewegt sich der Roboter, wenn ihr „Turn_right“ benutzt?

(Solange der rechte oder linke Digitalwert NICHT auf Null springt, könnt ihr „analog“ fahren.)

Maze

Die Grundfunktion „An der Wand lang“ habt ihr bewältigt.

Baut euch mit Büchern, Kisten oder sonstigem Material ein kleines Labyrinth auf.

Was soll der Roboter machen, wenn vor ihm eine Wand auftaucht?

Wie nah an die Wand soll er fahren?

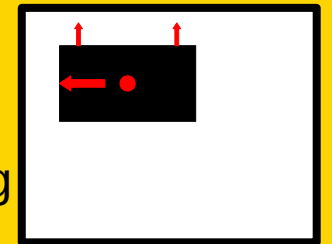
Wie lange, in Millisekunden, muss der Roboter „Turn_left“ ausführen, um eine 90° Drehung zu vollführen? (Ja, mit `_delay_ms()`)

Wie steht der Roboter nach der Drehung? Haben wir die Möglichkeit zu korrigieren? Wie?

Hat der Roboter bei der Drehung eine Wand berührt?

Wenn ja, welche Werte müssen anders gewählt werden, damit das nicht mehr passiert?

Um welchen Punkt dreht sich der Roboter.



Line

Nächster Schritt, Kreuzung:

Die grünen Punkte zu benutzen, um eine Kreuzung zu erkennen, ist nahezu unmöglich!

Dazu bräuchten wir andere Sensoren!

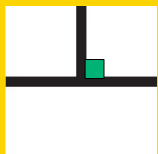
Aber, wodurch unterscheiden sich diese beiden Kacheln, wenn der Roboter von rechts kommt?



Genau! Erstmal garnicht!

D.h., wir müssen erstmal weiter fahren, um entscheiden zu können, ob das eine Kreuzung oder eine Ecke ist.

Na und wenn dann noch ein grüner Punkt hinzu kommt, wie erkennen wir den dann?



Da müsst ihr euch was schlaues ausdenken! Das ist nicht einfach! Aber, das kriegt ihr schon hin!!

Maze

Nächster Schritt, Wand zu Ende:

(Das Rastermaß des Labyrinths ist 30cmx30cm, gestrichelte Linien.)

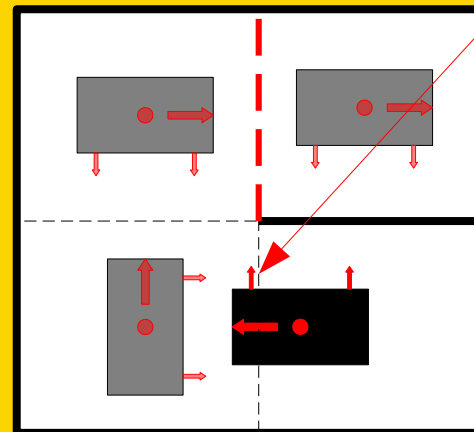
Welchen Wert nimmt der vordere, rechte Sensor an, wenn plötzlich keine Wand mehr da ist?

Wie weit muss der Roboter dann fahren, bevor er nach rechts drehen soll? (Lest dazu **jetzt** die Seite 6 der „Line-Spalte“, v.a. den grünen Teil)

Wie weit darf er maximal fahren? (Ultraschall)

Was kommt **zwangsläufig** nach der Rechtsdrehung?

Wie geht es dann weiter, wenn eine Wand (rot) folgt oder wenn keine folgt?

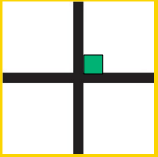


Nehmt euren Roboter in die Hand und „fährt“ damit durch das Labyrinth und überlegt welche Bewegungen der Roboter jeweils ausführen soll.

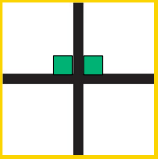
Line



Dieses Problem habt ihr, für den Fall gelöst, dass das Fahrzeug von rechts kommt! Was ist, wenn es von links kommt? Wie muss es sich dann verhalten?



Wodurch unterscheidet sich diese Situation, von der vorherigen? Wir erkennen früher, dass da eine Kreuzung ist. Wodurch?



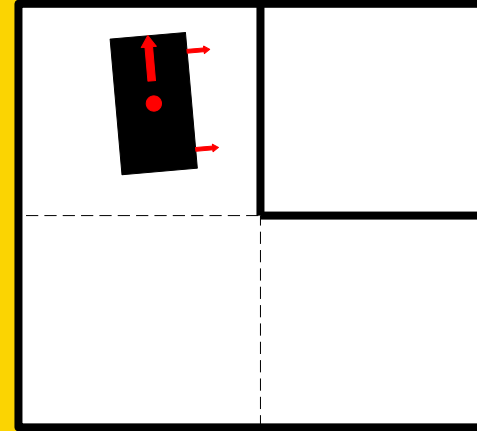
Na, das ist ja jetzt ein Klacks! Aber, wenn ihr von oben kommt und daher wenden müsst, wie stellt ihr sicher, dass ihr eine komplette 180°-Drehung ausgeführt habt? Sicher **NICHT**, in dem ihr eine bestimmte Zeit in eine Richtung dreht!!!



Wenn plötzlich der Strich zu Ende ist, das lässt sich ja ganz einfach fest stellen! Aber, wann sollte er wieder sichtbar sein? Wie stellen wir sicher, dass wir die Linie nicht verloren haben?

Maze

Der Roboter hat die Ecke bewältigt, aber er musste sich dabei, ohne weitere Orientierungsmöglichkeit, zweimal um 90° drehen. Er steht z.B. jetzt so:



Wie sieht das Programm aus, um den Roboter aus zu richten? Dabei müsst ihr bedenken, dass der IR-Abstandssensor sehr träge ist. Um einen Wert zu generieren, benötigt er 20ms!!! Wenn ihr also volle Pulle nach rechts dreht, steht ihr womöglich schon zur anderen Seite verdreht, bevor der Sensor einen neuen Wert ausgespuckt hat. Und dann korrigiert ihr wieder in die andere Richtung. Und so weiter. Der Roboter schüttelt sich.

Line

Noch ein paar Tipps für die Gap-Kachel:



Da wir an unseren Motoren keine sog. Encoder haben, müssen wir Distanzen über eine Zeitmessung „abschätzen“.

Dafür haben wir ja eine Uhr mit einer

Messgenauigkeit von 128 Mikrosekunden!

Ihr müsst also herausfinden, wie lange der Roboter für die 20cm Strecke ohne Linie braucht.

Hierbei hilft euch die `_delay_ms()` Funktion.

Lasst euren Roboter genau eine Sekunde lang mit der üblichen Geschwindigkeit fahren, misst die Strecke, die er dabei gefahren ist und rechnet aus, wie lange er dann für 20 cm braucht.

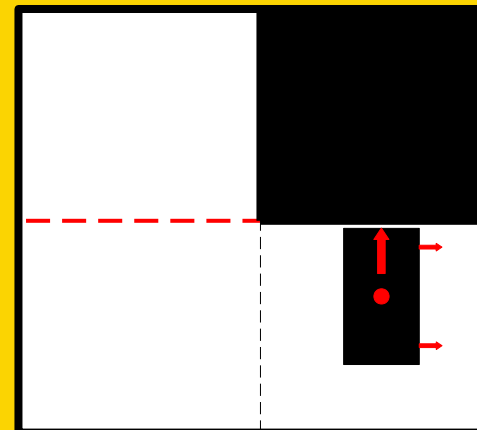
Da die Motoren unterschiedlich schnell drehen, je nach Ladezustand der Akkus, solltet ihr eine gewisse Toleranz einbauen.

Also: Linie verloren, Uhr auf Null stellen, sollte nach Ablauf der „20-cm-Zeit“ der Strich nicht wieder aufgetaucht sein, Linie suchen.

Maze

Jetzt fehlt gar nicht mehr so viel:

Ihr dürft auf die schwarze Kachel fahren, aber ihr dürft euch nicht auf ihr drehen. Wenn ihr also mit Hilfe der Werte des Farbsensors festgestellt habt, dass ihr auf eine schwarze Kachel gefahren seid, dann fahrt ruhig voll drauf. Denn, womöglich gibt es eine Wand, an der ihr euch ausrichten könnt. Und dann? Ja dann müsst ihr euch überlegen, was der Roboter machen muss, und zwar völlig fest gelegt, bis zu einem ganz bestimmten Punkt.



Wer es besonders gut machen will, merkt sich, dass er gerade auf einer schwarzen Kachel war. Ist nämlich als nächstes keine Wand (rot) da, dann kann der Roboter eine bestimmte Fahrt vermeiden. Geht aber auch ohne!

Line

Rescue Zone:

Den Silberstreifen am Eingang zur Rescue Zone eindeutig zu erkennen, ist relativ schwer.

Welchen Sensor könnt ihr noch benutzen, um zu entscheiden, dass ihr euch dort befindet?

Wenn ihr Bälle mit Hilfe von Greifarmen o.ä. einsammeln wollt, könnt ihr zusätzliche Servos an folgende Pins anschließen:

Die gelben Drähte an A2 und A3, wenn ihr keine Farbsensoren benutzt auch 7 und 11.

Dann müsst ihr zusätzlich die DDR-Register verändern: Auf „Out“ bzw. auf „1“!!! (Init.h)

Der braune oder schwarze Draht auf einen schwarzen Pin auf dem Board, den roten auf einen roten(+5V, **NICHT** 3,3V!!!).

Zur Verwendung von Servos gibt es ein Beispielprogramm: „102_Servo“.

Maze

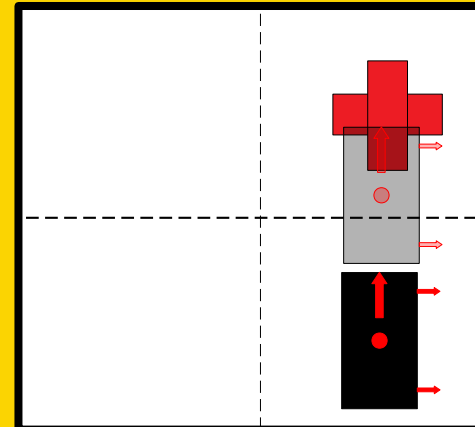
(Vor-)letztes Problem, rotes Kreuz:

Dass der Roboter auf der Kachel mit dem roten Kreuz angekommen ist, bemerkt er ja womöglich schon recht früh. Was ist dann zu tun?

Da es durchaus Situationen gibt, wo der Farbsensor meldet, dass etwas rotes unter ihm ist, solltet ihr euch NICHT mit einer Messung begnügen.

Letztes Problem: Wer ein Rescue-Kit abwerfen will: Das 3-D-Gedruckte Chassis lässt sich im Prinzip

wie Sperrholz bearbeiten. Aber, für z.B. 3mm Spax, solltet ihr 2,5mm große Löcher vorbohren. Ihr könntet z.B. den US-Sensor fest installieren und für die Abwurfseinheit den Servo benutzen. (Siehe auch „Line“-Spalte. Alle „Line_“-Pins!)



Montage und Programmierung
eines Roboters für
ROBOCUP JUNIOR RESCUE
mit Elegoo Car Kit
Teil 2.11: Remote Control

Von Charlotte und Andreas