universität freiburg

Hardware-Praktikum 2025

O. Amft und L. Häusler, Intelligent Embedded Systems Lab,
Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau

Übungsblatt 4 (26 Punkte)

Ziel:

Der Roboter fährt die Form eines Dreiecks mit Drehungen von möglichst exakt 120 Grad. Dazu wird ein Drehratensensor durch den Arduino ausgelesen, daraus die aktuelle Fahrtrichtung berechnet und die Moteren dementsprechend angesteuert.

Vorbereitung:

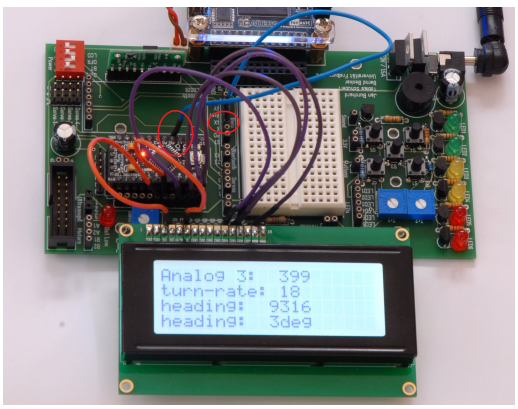
Auf dem Experimentierboard befindet sich ein analoger Drehratensensor. Dieser misst die Drehrate um zwei Achsen und gibt sie als Spannung aus. Ohne Rotation liegt diese Spannung bei ca. 1.23 V. Öffnen Sie zunächst die Datei "TurnRateSensor.ino". Verbinden Sie den Drehratensensor (Buchsenleiste "Gyro") wie folgt mit dem Microcontroller:

Gyro	Microcontroller
4X	A3

Der $4X$ Ausgang ist 4 mal empfindlicher als der X Ausgang des Drehratensensors. Er eignet sich daher besonders für langsamere Drehungen, die mit dem X Ausgang nicht so gut detektierbar sind.

Verbinden Sie außerdem das LCD wie im Quelltext beschrieben. Schalten Sie den Drehratensensor und das LCD an (DIP-Schalter).

Beachten Sie, dass das Auslesen des Drehratensensors so oft wie möglich erfolgen sollte. Vermeiden Sie daher Anweisungen wie "delay()" und "lcd.clear()" in Ihrem Arduino Code.



Aufgabe 1 (2 Punkte):

Geben Sie den aktuellen ADC Wert am Analog-Pin A3 des Microcontrollers in der ersten Zeile des LCD aus. Wenn das Board nicht bewegt wird, sollte ein konstanter Wert gelesen werden (dieser entspricht ca. 1.23 V, die der Drehratensensor ohne Rotation ausgibt).

Aufgabe 2 (2 Punkte):

Der Drehratensensor gibt die aktuelle Drehrate als Differenz zu dem in Aufgabe 1 beobachteten Wert aus. Speichern Sie die Ausgabe ohne Rotation in einer Variablen und geben Sie in der zweiten Zeile auf dem LCD die Drehrate aus (diese berechnet sich durch “aktueller Wert - Wert ohne Rotation”).

Hinweis: Warten Sie nach dem Einschalten des Boards einen Moment und speichern Sie dann den aktuellen Wert an A3 als Ausgabe des Drehratensensors ohne Rotation ab. Sie können auch zusätzlich noch über mehrere Messungen mitteln. Damit ermitteln Sie den Ruhewert für den Sensor des verwendeten Roboters.

Hinweis: Beachten Sie, dass die Drehrate sowohl positiv als auch negativ sein kann (verwenden Sie zum Speichern also keine *unsigned* Variablen).

Aufgabe 3 (5 Punkte):

Berechnen Sie nun die Richtung des Boards. Dazu müssen Sie über die Drehrate integrieren. Führen Sie hierzu eine neue Variable “heading_int” ein. Messen Sie die aktuelle Drehrate und multiplizieren Sie diese mit der Zeit, die seit der letzten Messung vergangen ist. Addieren Sie das Ergebnis zur Variable “heading_int”. Geben Sie den Wert von “heading_int” in der dritten Zeile des LCD aus.

Hinweis: Verwenden Sie Variablen mit einer hohen Anzahl an Bits um einen Overflow zu vermeiden.

Hinweis: Da der in Aufgabe 2 bestimmte Mittelwert leider nie genau in der Mitte liegt und der Sensor rauscht können Sie ein Deadband um die in Aufgabe 2 ermittelte Drehrate legen: Ist die absolute Drehrate kleiner als eine Schranke wird “heading_int” nicht verändert.

Hinweis: Wenn sich das Board nicht bewegt, sollte der Wert von “heading_int” quasi konstant bleiben, ansonsten je nach Drehrichtung entweder fallen oder steigen.

Aufgabe 4 (3 Punkt)

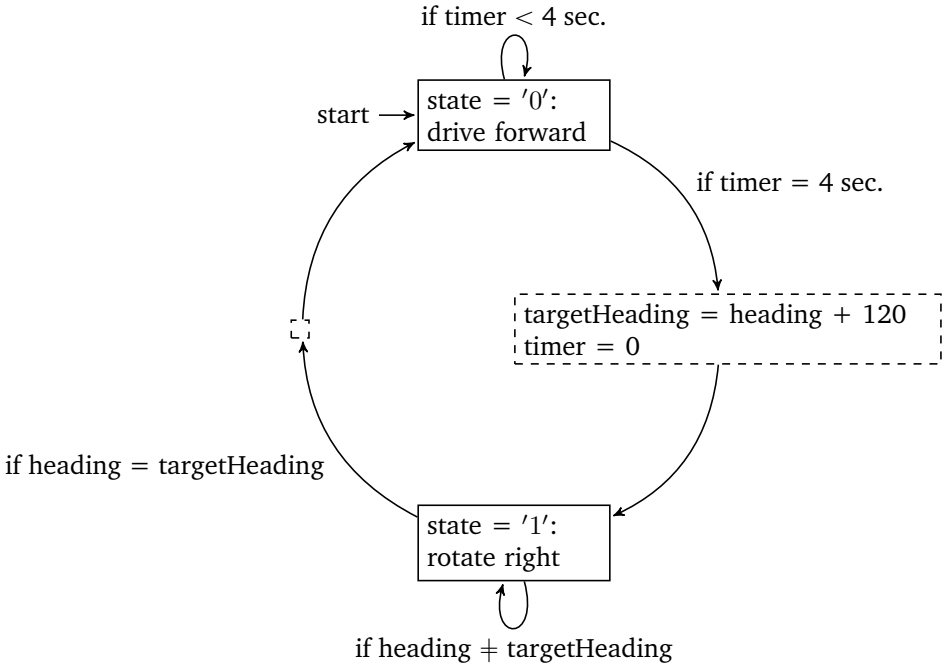
Geben Sie in der vierten Zeile des LCD die Richtung des Boards “heading” in Grad (eine ganze Umdrehung entsprechen 360 Grad) aus. Finden Sie dazu einen passenden Skalierungsfaktor für “heading_int”. Beachten Sie außerdem, dass die Ausgabe zwischen 0 und 359 Grad liegen soll (eine Drehung um 360 Grad soll also als 0 Grad ausgegeben werden). Im Folgenden wird mit diesem skalierten Wert gearbeitet.

Aufgabe 5 (2 Punkt)

In den folgenden Aufgaben muss nun eine Steuerlogik auf dem Arduino implementiert werden, die den Roboter im Dreieck fahren lässt. Implementieren sie hierfür die Funktion “setMotor” mit drei Parametern um die Richtung, den Motor sowie die Geschwindigkeit auszuwählen. Die Regulierung der Geschwindigkeit über PWM sollte mit “analogWrite” erfolgen, verwenden Sie zum erzeugen des LOW Pegels des anderen Steuersignales weiterhin “digitalWrite”.
Sie können hierfür ihre Implementierung aus dem zweiten Übungsblatt ggfs. anpassen und wiederverwenden.
Verbinden Sie die Motoren zudem so wie in dem Kommentar der gestellten .ino Datei. Achten Sie auch hier darauf, ihre Implementierungen zuerst ohne Bodenkontakt der Räder zu testen um Beschädigungen zu vermeiden.

Aufgabe 6 (8 Punkte):

Für die Dreiecksfigur soll der Roboter zuerst geradeaus fahren und dann alle 4 Sekunden eine Drehung um 120 Grad vollziehen. Fügen sie ihrem Programm für die Implementierung die Variablen “state” , “timer” und “targetHeading” hinzu. Die Steuerung dieser Variablen und dadurch später der Motoren wird über die Implementierung des folgenden einfachen Zustands-Automaten vorgenommen:



Wenn der Zustand ’0’ ist, fährt der Roboter vorwärts. Der “timer” zählt dabei die Zeit. Nachdem der Roboter 4 Sekunden geradeaus gefahren ist, wird die Ziel-Richtung (targetHeading) berechnet und gespeichert. Gleichzeitig wird auch der Timer wieder zurückgesetzt. Nun ist der Zustand ’1’ erreicht und es dreht sich der Roboter so lange, bis die aktuelle Richtung der Ziel-Richtung entspricht. Ist die gewünschte Richtung erreicht, wechselt der Zustand wieder zurück zu ’0’ und der Roboter fährt geradeaus usw.

Implementieren sie den beschriebenen Zustandsautomaten vorest ohne die Ansteuerung der Motoren und geben Sie die Variabel “targetHeading” auf der dritten Zeile des LCD aus. (Kommentieren sie hierfür die Ausgabe von “heading_int” aus der dritten Aufgabe aus). Überprüfen Sie, ob “targetHeading” richtig ermittelt wird, indem sie den Roboter in die entsprechende Richtung drehen, bevor Sie mit der nächsten Aufgabe fortfahren.

Hinweise:

- “heading” wird nur für die Drehung, nicht aber für das eigentliche Fahren genutzt. Dadurch kann der Einfluss des Drifts[1] des Drehraten-Sensors minimiert werden.
- “heading” ist ein Wert zwischen 0 und 359 Grad. Die “targetHeading” Werten liegen also auch in diesem Bereich. Prüfen sie auf einen potentiellen Überlauf des Werts von “targetHeading” und ziehen Sie dann 360 Grad ab (es gilt z.B.: “350+120Grad $\equiv$ 350+120-360 = 110 Grad”).
- Wenn Sie “heading” mit “targetHeading” vergleichen, sollten Sie einen kleinen ϵ-Bereich von z.B. ± 2 um “targetHeading” legen, damit der Roboter die Drehung auch stoppt, wenn der genaue target heading Wert übersprungen wird. Mit einem ϵ-Bereich von ± 2 müssen Sie die Grenzfällen ($\geqslant 358$ und $\leqslant 1$ Grad) separat betrachten. Dies können Sie z.B. durch vier separate *if* Abfragen realisieren.

Aufgabe 6 (4 Punkte):

Implementieren sie zu guter Letzt die Ansteuerung der Motoren. Im Zustand ’0’ des implementierten Automats soll der Roboter geradeaus fahren. In Zustand ’1’ dreht sich der Roboter in die entsprechende Richtung.
Hinweise:

- Sie können nach Bedarf zusätzliche Variablen einführen, um z.B. die Richtungen der Moten abhängig vom Zustand zu speichern.
- Wählen Sie eine angemessene Geschwindigkeit. Zu schnelle oder zu langsame Drehungen können je nach Untergrund für eine nicht korrekten Erkennung der Drehungen führen.

Abgabe:

Archivieren Sie das Programm in einer ZIP-Datei und laden Sie diese im Übungsportal hoch. Das Programm sollte die Lösung für alle Aufgabenteile beinhalten. Aufgabenteile und/oder Zwischenschritte die durch weitere Aufgaben verändert oder herausgenommen wurden, sollten auskommentiert und nicht gelöscht werden. Überprüfen Sie die Archiv-Datei auf Vollständigkeit. Überprüfen Sie insbesondere auch die im Übungsportal hochgeladene Datei.

[1] Unter Drift versteht man den Fehler, der durch das Integrieren der Drehrate entsteht: Leichte Schwankungen im Drehraten-Signal addieren sich bei kontinuierlicher Integration auf und verfälschen damit die Richtungsbestimmung. Bei dem verwendeten Gyro liegt der Drift meistens bei unter 45 Grad pro Minute. Falls Sie einen hohen Drift beobachten (der Roboter steht still, aber "heading"verändert sich), starten Sie den Microcontroller mittels resetButton neu. Mehr zum Thema: https://towardsdatascience.com/dead-reckoning-is-still-alive-8d8264f7bdee