Regelung eines Hebelarms

Aufgabenstellung

Freiwilliges praktisches Projekt
Systemtheorie II
Zeitraum 1

WS 24/25

Kontaktadresse:

acs-teaching-sys2@eonerc.rwth-aachen.de

Inhaltsverzeichnis

2 Einleitung

Vorsicht: Es besteht Quetschgefahr. Stellen Sie jederzeit sicher, dass sich der an dem Motor befestigte Hebelarm frei bewegen kann.

Verwenden Sie die freie Fläche hinter dem Arduino, um den Aufbau mit einem Gewicht (z.B. einem Buch) zu stabilisieren.

Ein seperates Dokument mit Hilfestellungen zur Konfiguratrion der Hard- und Software finden Sie im Moodle Lernraum (Setup-Hilfe).

Diese Aufgabe ist zum Verständnis und zur Vertiefung der in der Vorlesung und Übung Systemtheorie II gelehrten Inhalte gedacht. Bei Bearbeiten der Aufgabe erhalten die Studierenden maximal 5 Bonuspunkte für die Klausur Systemtheorie II (Wintersemester 24/25 und Sommersemester 2025). Die Bonuspunkte werden den in der Klausur erzielten Punkte aufaddiert, auch wenn die Klausur ohne Anrechnung der Bonuspunkte nicht bestanden wurde. Teilweise gehen die Inhalte der Aufgaben über diejenigen der Vorlesung hinaus. Trifft das zu, werden entsprechende Hinweise bzw. Lösungshilfen gegeben.

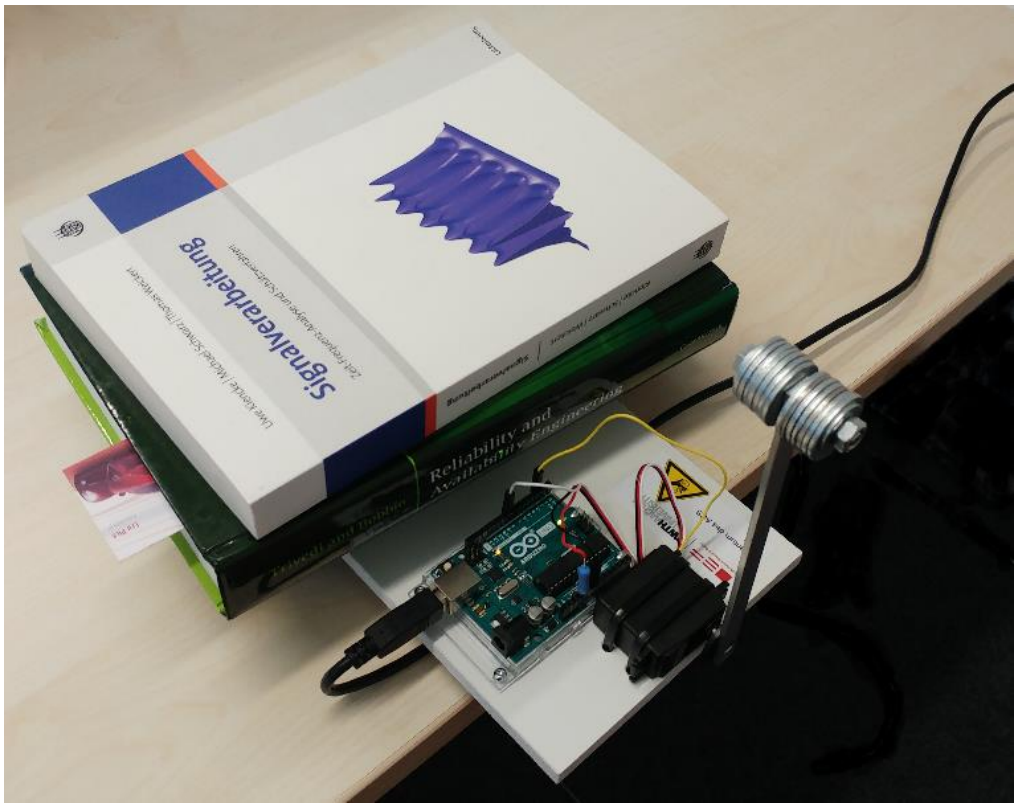
Die gesamte Hardware ist Eigentum des ACS und sie muss unbeschädigt und unverändert am Lehrstuhl wieder abgegeben werden. Beschädigte Teile müssen auf eigene Kosten ersetzt werden.

Die Lösung ist als schriftliche Ausarbeitung bis spätestens zum **16. Dezember 2024** für den ersten Zeitraum und bis zum **15 Januar 2025** für den zweiten Zeitraum über eine entsprechende Upload-Funktion auf RWTH Moodle einzureichen. Einreichungen nach dieser Frist werden nicht akzeptiert! **Die Lösung muss je nach Aufgabenteil eine nachvollziehbare Lösung (mit Lösungsweg), Screenshots von Plots (mit erkenntlicher Legende und Beschriftung) und ihre student.ino enthalten.** Ein Template im Word-Format zur Anfertigung der schriftlichen Ausarbeitung wird auf RWTH Moodle bereitgestellt. **Die Einreichung (d.h. das Hochladen) der Lösung sollte dabei im PDF-Format mit einer separaten *ino-Datei* erfolgen.**

2.1 Versuchsbeschreibung

In dieser Aufgabe soll mit der Arduino IDE, sowie einem Python-Skript eine Positionsregelung für einen Hebelarm ausgelegt, implementiert und dann in einem „Hardware-in-the-Loop“ (HiL) Aufbau getestet werden. Der erste Aufgabenteil dient als Einführung in die Hardware. Im zweiten Aufgabenteil werden verschiedene Störungen auf dem Hebelarm identifiziert. Im dritten Aufgabenteil befassen Sie sich mit der Entwicklung eines Reglers zur Positionsregelung des Hebelarms und im letzten Aufgabenteil werden verschiedene Störungssignale mit der Steuergröße überlagert, welche dann kompensiert werden müssen. Es wird empfohlen, die Aufgaben in dieser Reihenfolge zu bearbeiten.

Es wird ein Arduino und ein Servomotor[1], montiert auf einer Kunststoffplatte (siehe Bild unten), bereitgestellt. Der Motor ist elektrisch in seinem maximalen Drehmoment begrenzt, um eine Zerstörung bei Überlastung zu vermeiden. Überschreiten der Drehmomentgrenze hat einen sogenannten Brownout der Motorsteuerung zu Folge. Bei einem Brownout schaltet die Motorsteuerung ab und nach wenigen Sekunden wieder ein. Der Brownout ist zu vermeiden, da sich der Motor trotz Schutzbeschaltung stark erhitzen kann.



Exemplarisches Hardware-Setup aus dem WS19/20: Hebelarm als „Inverses Pendel“ mit Gewichten

[1] https://www.mouser.de/datasheet/2/321/900-00360-Feedback-360-HS-Servo-v1.2-1147206.pdf

2.2 Bereitgestellte Software

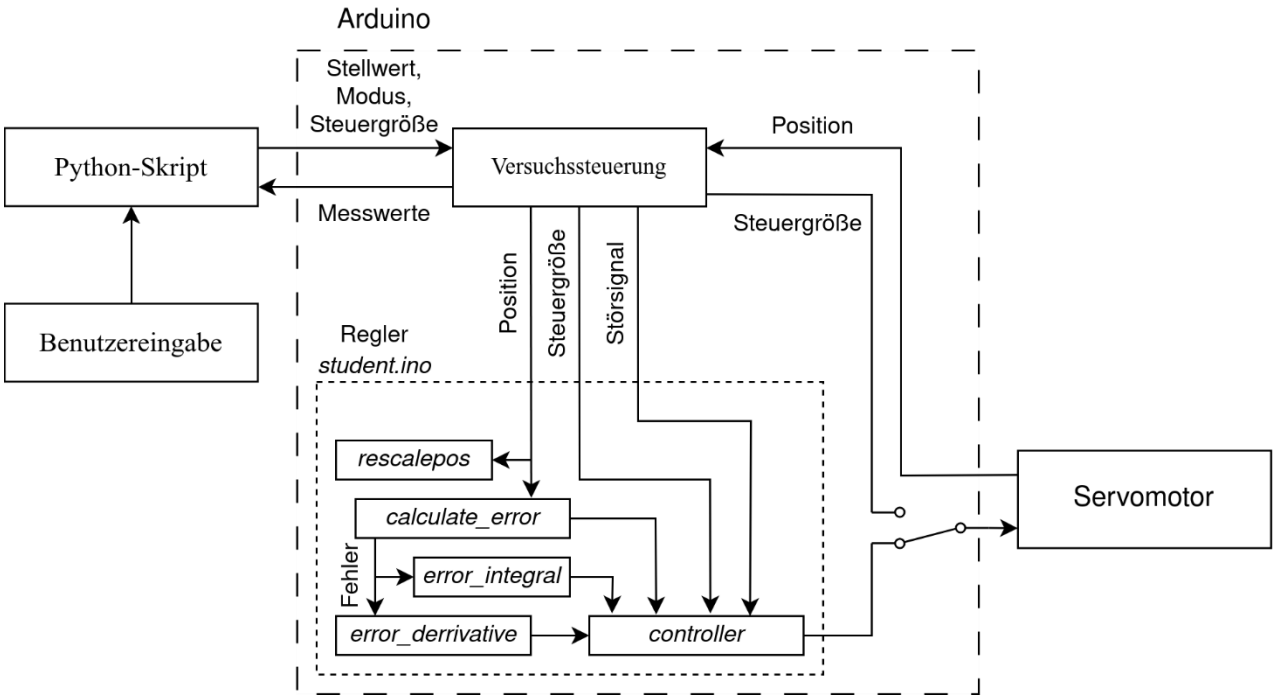
Ein seperates Dokument mit einer Einführung zur Arduino Software, sowie dem Python Skript finden Sie im Moodle Lernraum (Software-Einführung).

2.2.1 Aufbau

Der Arduino Code stellt die Verbindung zu unserem Microcontroller und dem darauf liegenden Servo-Motor her. An den Microcontroller werden mit dem Python-Skript das Störsignal, die Führungsgröße für die Position und optional der *Torque* übergeben. Der Servo-Motor gibt ein rohes Positionssignal zurück. Der Eingang *Torque* ist ein Parameter der proportional zu dem am Motor eingestellten Drehmoment ist.

Als Teil des Modus, können verschiedene *Störsignal ausgewählt werden, die den Servomotor beeinflussen.* Für den Modus *PYTON_TORQUE*, ist der Schalter vor dem Servomotor umgeschaltet und der Wert aus der Benutzereingabe wird direkt umgesetzt.

Der Zusammenhang zwischen den Parametern und Funktionen ist im folgenden Bild veranschaulicht.



3 Aufgabenstellungen

Ein seperates Dokument mit Hilfestellungen zur Konfiguratrion der Hard- und Software finden Sie im Moodle Lernraum (Setup-Hilfe).

3.1 Aufgabe 1: Aufbau und grundlegende Motorsteuerung

Verbinden Sie den Servomotor entsprechend der folgenden Tabelle mit dem Arduino.

Servomotor	Arduino
Rot mit Widerstand	+5V
Schwarz	GND
Weis	Pin 6
Gelb	Pin 3

Für alle Aufgabenteile wird das Python-Skript verwendet. Bevor Sie mit den Aufgaben beginnen, vergewissern Sie sich, dass Sie alles, wie im Setup-Hilfe Dokument beschreiben, aufgebaut und installiert haben.

Hinweis:

Machen Sie sich mit Hilfe der beigelegten Dokumente mit dem Python-Skript und seinen Funktionen vertraut, sowie mit der Arduino-Umgebung. Achten Sie vor allem auf den richtigen Anschluss des Arduinos, sowie die COM-Auswahl.

Aufgabe 1.1

Führen Sie das Python-Skript aus und testen Sie die Verbindung mit dem Arduino (Achten Sie darauf, das der richtige Port ausgewählt ist). Überprüfen Sie, ob sie einen *Torque* von 5 einstellen können. Der Befehl hierfür sollte etwa so aussehen: *python3 plotData.py --duration 10 --torque 5 PYTHON_TORQUE*

Machen Sie sich mit dem Positionssignal vertraut (Form, Frequenz …). Stellen Sie nun eine Umlaufzeit von 5 Sekunden ein (im Uhrzeigersinn).

- Speichern Sie das Positionssignal mit einer Umlaufzeit von 5s als PNG. Sie können dafür alle anderen Signale in Plotly ausblenden. Fügen Sie das PNG ihrem Bericht hinzu und beschreiben Sie den Verlauf sowie markante Punkte des Signals. Gehen Sie so für alle weiteren Aufgaben vor, in denen die Dokumentation von Signalen gefordert wird.

Aufgabe 1.2

- Bestimmen Sie mit welcher maximalen Frequenz das System geregelt werden darf. *Hinweis: Betrachten Sie das Positionssignal. Sie können in plotly das Signal vergrößern.*
- Was ist demnach die kleinste mögliche Regelzeitkonstante dieses Systems?
- Beschreiben Sie, welche Vorteile eine Regelung auf dem Arduino gegenüber einer Regelung auf einem PC in diesem Aufbau hat in Hinsicht auf die Regelgüte.

Aufgabe 1.3

- Vervollständigen Sie die Funktion *rescalepos* in der *student.ino* Datei, sodass das Positionssignal mit Hilfe eines Umrechnungsfaktors und einer Nullpunktverschiebung auf 360° abgebildet wird. Die untere Gleichgewichtslage soll dabei 180° entsprechen. Ihre *rescalepos* Funktion muss dabei zwingend eine Nullpunktverschiebung enthalten. Dokumentieren Sie ihr Ergebnis. Dokumentieren Sie wie in Aufgabe 1.1 beschreiben das „Rescaled Position“ Signal aus *plotly*.

3.2 Aufgabe 2: Störsignale des Hebelarms

Bestimmen Sie zunächst den Übertragungsfaktor zwischen dem *Torque* Eingang und dem *Pos* Ausgang des Servo-Motors. Untersuchen Sie folgende Werte für den Torque:

τ_{in} : -20, -10, -8, -5, 5 ,8, 10, 20

Aufgabe 2.1

- Stellen Sie *Torque* auf die verschiedenen Werte ein und untersuchen sowie dokumentieren Sie *Pos* sowie die Ableitung von *Pos*.
- Erklären Sie, wie möglicherweise auftretende Sprünge zustande kommen.
- Erklären Sie, wieso die Ableitung nicht immer konstant ist, sondern z.B. zwischen Null und einem Wert x springt.

Aufgabe 2.2

- Bestimmen Sie für die verschiedenen τ_{in} die Amplitude der Ableitung.
- Bestimmen Sie entsprechend einen Proportionalitätsfaktor zwischen *Torque* und der Ableitung von *Pos* im Bereich. Gehen Sie auf mögliche Nichtlinearitäten ein.
- Bestimmen Sie einen Offset für τ_{offset}, welcher Ungleichheiten in der Drehrichtung ausgleicht, basierend auf den vorherigen Ergebnissen.

Aufgabe 2.3

Bestimmen Sie nun die verschiedenen Fehlersignale. **Führen Sie diese Aufgabe ohne eine Implementierung eines Reglers in ihrer *student.ino* durch. Diese folgt in Aufgabe 3.** Sie können das Python-Skript wie folgt ausführen:

python3 plotData.py --duration 10 ERROR1

- Bestimmen und dokumentieren Sie für jedes Störsignal den Signaltyp, Amplitude und Frequenz gemäß der Spalte „Signalbeschreibung“ in der Tabelle. *Plotly* Abbildungen sind hier nicht nötig.
- *Hinweis: Betrachten Sie mehrere Perioden des Störsignals und bestimmen Sie ggf. Mittelwerte, um Ungenauigkeiten des realen Systems zu verringern.*

Störsignaltyp / Ausgewählter Modus	**Signalbeschreibung**
NO_ERROR	Kein Störsignal vorhanden.
ERROR1	
ERROR2	
ERROR3	
ERROR4	
ERROR5	
ERROR6	
ERROR7	
ERROR8	
ERROR9	

Aufgabe 2.4

- Welche maximale Störsignalfrequenz kann bei den verwendeten Simulationseinstellungen gemessen werden? Begründen Sie Ihre Aussage mathematisch.

3.3 Aufgabe 3: Positionsregelung

In dieser Aufgabe wird ein Regler zur Positionsregelung des Hebelarms entworfen. In dieser und der folgenden Aufgabe wird das unskalierte Positionssignal betrachtet.

Aufgabe 3.1

- Implementieren Sie die Funktion *calculate_error* in der *students.ino.* Die Funktion soll immer den kleinsten Winkel zu der Regelgröße als Grundlage für den Fehler nutzen.
- Beachten Sie bei der Implementierung mögliche Sprünge im Positionssignal.
- Dokumentieren Sie das *Error* Signal mit dem *Position* Signal bei einer Regelgröße von 250 und einem Torque von 7. Sie können das Skript dafür wie folgt aufrufen:
  python3 plotData.py --duration 10 --torque 7 --position 250 PYTHON_TORQUE
- Implementieren Sie zusätzlich die Funktionen *error_derivative* und *error_integral* für eine mögliche Nutzung als Teil eines PID-Reglers.

Aufgabe 3.2

- Überführen Sie zunächst die zeitkontinuierliche PID-Funktion in Ihre zeitdiskrete Version und machen sich mit dieser vertraut. Beschreiben Sie ihr vorgehen und geben Sie die Lösung im Blockschaltbild an.
- Implementieren Sie einen PID-Regler in der Funktion *controller* in der *students.ino*, der die Position des Hebelarms auf eine beliebige Position einstellen kann.
- Nutzen Sie hierbei τ_{offset} , den in Aufgabe 2 ermittelten Wert.

Aufgabe 3.3

Stellen Sie zuerst nur einen Verstärkungsfaktor von 0,1 ein. Sie nutzen hier dadurch einen P-Regler.

- Dokumentieren und erklären Sie den stationären Regelfehler. Kann der stationäre Fehler auf diese Weise eliminiert werden? Wie ändert sich das Systemverhalten in Bezug auf den stationären Fehler sowie den Einschwingvorgang?
- Dokumentieren Sie das Systemverhalten für die verschiedenen Verstärkungsfaktoren.

Aufgabe 3.4

- Entwerfen Sie einen Regler mit Hilfe der zeitdiskreten Einstellregeln. Wählen Sie den aus Ihrer Sicht besten Reglertyp (P, PI, PID) aus und begründen Sie ihre Wahl.
  Hinweis: Es sind keine mathematischen Herleitungen erforderlich und Sie können von der untenstehenden Tabelle Gebrauch machen. Beachten Sie allerdings, dass die Tabelle für kontinuierliche Systeme gilt. Sie können die Frequenz, mit der der Regler ausgeführt wird, an dem k-Signal in plotly ableiten, welches bei jeder Ausführung hochzählt.

Regler	K_R	T_I	T_D
P	$0{,}5 \cdot K_{R,krit}$	–	–
PI	$0{,}45 \cdot K_{R,krit}$	$0{,}83 \cdot T_{krit}$	–
PID	$0{,}6 \cdot K_{R,krit}$	$0{,}5 \cdot T_{krit}$	$0{,}125 \cdot T_{krit}$

3.4 Aufgabe 4: Kompensation einer Störung

In diesem Aufgabenteil wird untersucht, wie eine Störung am Eingang der Regelstrecke ausgeregelt werden kann. Verwenden Sie Ihre Reglerparameter aus Aufgabe 3.4.

Aufgabe 4.1

- Kompensieren die verschiedenen Störungen (TYPE 1 – TYPE9), indem Sie Ihre *controller*-Implementierung modifizieren und den *measured_disturbance* Parameter nutzen.
- Untersuchen, vergleichen und dokumentieren Sie den Einfluss der verschiedenen Störungen (TYPE1 – TYPE9) auf die Regelung (d.h. jeweils mit und ohne Kompensation).
- Erläutern Sie, welche Rolle eine Totzeit der Kommunikation zwischen dem Arduino und einem Messgerät für ein Störsignal spielen könnte.