Hochschule RheinMain
Fachbereich Design Informatik Medien
Prof. Dr. Robert Kaiser

Echtzeitverarbeitung

SS 2022
LV 4511 / LV 8481

Übungsblatt 5
Laborversuch
Abgabe: 11. Woche (22.06.2022)

Aufgabe 5.1. (PID-Regler-Anwendung – Überblick):

Ziel der folgenden Übungen ist es eine PID-Regler-Anwendung für einen PC-Lüfter zu entwickeln, also eine Lüftersteuerung!

Diese Lüftersteuerung soll aus zwei möglichen Frontends und einem Regelungsmodul bestehen. Das erste Frontend soll über die Programmparameter die Regelungsparameter übergeben bekommen und mit diesen die Regelung durchführen, bis es beendet wird. Das zweite Frontend soll mit den Programmparametern initialisiert und danach zur Laufzeit über Benutzereingaben gesteuert werden.

Die Zuordnung der GPIOs und die Einstellparameter für das Hardware-PWM soll zur Compile-Zeit konfigurierbar sein; eine Konfiguration über eine Config-Datei zum Startpunkt des Programms ist optional.

Für die Umsetzung soll primär die Funktionalität der wiringPi-Bibliothek verwendet werden.

Aufgabe 5.2. (Info: PC-Lüfter ansteuern):

PC-Lüfter besitzen meist einen bürstenlosen Gleichspannungsmotor mit entsprechenden Motor-Controller, sie können daher i.d.R. nur in eine Richtung drehen und die Polarität der Spannungsversorgung kann nicht vertauscht werden. Zur Überwachung verfügen PC-Lüfter häufig über ein Tacho-Signal, mit dem die Umdrehungszahl ermittelt werden kann.

Die Geschwindigkeit des Lüfters kann über PWM gesteuert werden. Im einfachsten Fall in dem in die Spannungsversorgung eingegriffen wird. Dies führt dazu, dass das Tacho-Signal nicht mehr so einfach auszuwerten ist (Kein Strom, kein Signal). Neuere Lüfter (die mit 4-Pins) haben eine eigene Leitung für ein PWM-Signal. Die Umsetzung erfolgt dabei im Motor-Controller des Lüfters. Die Spezifikation ist hier [1] zu finden.

Das EchtzeitHat hat einen eigenen PC-Lüfter-Anschluss. Folgende Informationen sind dazu relevant:

- Das Tacho-Signal des Lüfters erzeugt zwei Pulse pro Umdrehung.
- Das Tacho-Signal liegt an GPIO 3 an.
- Das PWM-Signal für einen Lüfter muss eine Frequenz von 25kHz haben (Range 21-28kHz).

- Das PWM-Signal kann entweder mit GPIO 18 oder GPIO 2 erzeugt werden. Zum Umschalten gibt es eine entsprechende Jumper-Brücke über dem Lüfter-Stecker.
- Der Lüfter benötigt eine Spannungsversorgung mit 12V. D.h. sie müssen das EchtzeitHat mit dem 12V-Steckernetzteil betreiben.

Ein PWM-Signal von 25kHz können wir sowohl in Software wie auch Hardware erzeugen.

Für den Fall, dass das Hardware-PWM mit der wiringPi-Bibliothek verwendet wird, muss der PWM konfiguriert werden!

- `pwmSetMode()` muss auf `PWM_MODE_MS` gesetzt werden.
- `pwmSetClock()` muss auf "8" gesetzt werden (Die Clock des PWM hat die Frequenz 19.2MHz, mit der Funktion wird ein Teiler gesetzt. 19.2MHz/8 ist nahe an 25kHz)
- Mit `pwmSetRange()` kann der Zyklus unterteilt werden (bspw. in 100 Teile für 0 bis 100 %).

*Achtung: der Lüfter hat recht scharfe Ecken! Greifen Sie **nicht** in den laufenden Lüfter (Sonst passiert Ihren Fingern das selbe wie meinen und Sie verschmutzen Ihre Umgebung. Die Lüftungsgitter sind da nicht zum Spaß …).*

Aufgabe 5.3. (PC-Lüfter ansteuern):

Schließen Sie den ausgegebenen PC-Lüfter an das EchtzeitHat an. *Achtung: der Lüfter läuft direkt los!.*

Schreiben Sie ein Mini-Programm, das mit PWM die Geschwindigkeit des Lüfters steuert. Sehen Sie die Möglichkeit vor, zwischen Software-PWM (auf GPIO 2) und Hardware-PWM (auf GPIO 18) umzuschalten. Schaffen Sie es mit dem Software-PWM die Drehzahl zu steuern?

Erweitern Sie das Programm mit der Funktionalität über (Software-) Interupts die Pulse des Tachos zu erfassen und sekündlich auszugeben. Rechnen Sie diese auch in Umdrehungen pro Minute um.

Lassen Sie den Lüfter entsprechend dem Standard ([1]) mit maximaler und minimaler Geschwindigkeit laufen. Vergleichen Sie dies mit den Angaben auf der Packung. Was fällt auf?

Aufgabe 5.4. (PID-Regler-Modul):

Entwickeln Sie ein PID-Regler-Modul, das in einem Regelkreis eingesetzt werden kann. Das Modul soll mit den Konstanten der drei Übertragungsfunktionen initialisiert werden. Die gesetzten Konstanten sollen später über sechs Funktionen/Methoden auslesbar und veränderbar sein. Es soll zwei Funktionen/Methoden geben, die die Regeldifferenz und die Führungsgröße übergeben bekommen und die Stellgröße berechnen. Eine dieser Funktionen soll von einem konstanten Δt ausgehen, die andere zusätzlich t_n und t_{n-1} übergeben bekommen. Denken Sie daran, dass das Modul ein „Gedächtnis" für vorherige Werte benötigt. Sehen Sie auch eine Funktion/Methode vor, dieses Gedächtnis zurückzusetzen.

Sehen Sie vor, dass die Führungsgröße die Einheit Umdrehungen pro Minute und die Stellgröße den Wertebereich 0 bis 100 (also Prozent) hat.

Aufgabe 5.5. (Erster schneller Regler):

Bauen Sie ein erstes kleines Programm, dass feste Regler-Parameter hat und in einem Regelkreis den PC-Lüfter steuert. Verwenden Sie die SoftPWM-Funktionalität der wiringPi-Bibliothek oder das Hardware-PWM des Raspberry Pis. Finden Sie durch ausprobieren Konstanten, die zu einer brauchbaren Regelleistung führen.

Aufgabe 5.6. (Konfiguration – Programmparameter):

Erweitern Sie Ihr Programm so, dass über die Kommandozeile beim Starten des Programms Parameter übergeben werden können.

Diese Parameter sollen mindestens umfassen:

- `-c` Zykluszeit (in ms) des Regelkreises
- `-p` P-Konstante (als Gleitpunktzahl)
- `-i` I-Konstante (als Gleitpunktzahl)
- `-d` D-Konstante (als Gleitpunktzahl)
- `-s` Sollwert in Umdrehungen pro Minute (rpm)

Wenn eine Konstante 0 ist bzw. nicht angegeben wird, dann soll diese Übertragungsfunktion deaktiviert werden.

Nach dem Start soll der Regler initialisiert werden und der Regelkreis gestartet.

Geben Sie formatiert maximal 1x pro Sekunde die wichtigsten Werte auf der Standardausgabe aus.

Aufgabe 5.7. (Regeleinstellungen):

Probieren Sie die Methode von Ziegler und Nichols aus, um die richtigen Parameter für den Regler zu ermitteln.

Nehmen Sie eine nicht zu kleine Zykluszeit, damit es auch zu Schwingungen kommt, da der Motor recht träge ist. Verwenden Sie als Soll-Wert 50% der maximalen Geschwindigkeit, die der Motor erreichen kann. Eine Schwingung wäre, wenn der Motor periodisch schnell und wieder langsam wird, aber nie den Soll-Wert erreicht (ähnlich einer Sinus-Welle).

Funktioniert die Methode? Wann treten für Probleme auf (wenn überhaupt)?

*Achtung es kann gut sein, dass dies **nicht** funktioniert. Probieren Sie es aus, dokumentieren Sie Ihr Vorgehen und die Ergebnisse, aber investieren Sie nicht zu viel Zeit, wenn es nicht geht!*

Aufgabe 5.8. (Konfiguration – Laufzeit):

Erweitern Sie Ihr Programm so, dass zur Laufzeit die Standardeingabe eingelesen wird. Am besten Sie verschieben hierzu den Regler, oder die Eingabe in einen eigenen Thread.

Die aktuelle Konfiguration soll auf der Standard-Ausgabe ausgegeben werden (am besten mit Überschreiben, so dass nicht immer eine neue Zeile begonnen wird).

Die Zykluszeit soll weiterhin ausschließlich per Programmparameter eingestellt werden.

Mit einem Tastendruck, soll der Wert inkrementiert, bzw. dekrementiert werden. Verwenden Sie dazu Inkremente, die sinnvoll sind.

Verwenden Sie

- s um den Soll-Wert zu dekrementieren
- S um den Soll-Wert zu inkrementieren
- p um die P-Konstante zu dekrementieren
- P um die P-Konstante zu inkrementieren
- i um die I-Konstante zu dekrementieren
- I um die I-Konstante zu inkrementieren
- d um die D-Konstante zu dekrementieren
- D um die D-Konstante zu inkrementieren

Aufgabe 5.9. (Regeleinstellungen – Experimentell):

Beginnen Sie ggf. mit den Regeleinstellungen aus der vorherigen Aufgabe.

Probieren Sie verschiedene Regler-Konfigurationen aus (P, I, PI, PD, PID, ID). Können Sie deren Eigenschaften experimentell ermitteln, bzw. die in der Vorlesung genannten bestätigen? (also die Annäherung des Ist-Werts an den Soll-Wert über die Zeit)

A. (*):

Literatur

[1] `https://usermanual.wiki/Collections/Hardware-Manuals/pc_hardware/4_Wire_PWM_Spec.pdf.html`

B. (Raspberry Pi GPIO Pins):

Raspberry Pi – GPIO-connector

HAT	WiringPi	GPIO	Name	Header		Name	GPIO	WiringPi	HAT
			3.3V	1	2	5V			
FanSoftPWM	8	2	SDA	3	4	5V			
Fan Tacho	9	3	SCL	5	6	GND			
	7	4		7	8	TxD	14	15	
			GND	9	10	RxD	15	16	
	0	17		11	12		18	1	PWM/GPIO18
SW2	2	27		13	14	GND			
SW1	3	22		15	16		23	4	
			3.3V	17	18		24	5	
	12	10	MOSI	19	20	GND			
	13	9	MISO	21	22		25	6	
	14	11	SCLK	23	24	CE0	8	10	
			GND	25	26	CE1	7	11	
				27	28				
DRV_A_en	21	5		29	30				
DRV_A_in1		6		31	32		12	26	DRV_A_in2
DRV_B_in1	23	13		33	34				
DRV_B_in2	24	19	MISO	35	36				
DRV_B_en	25	26		37	38	MOSI	20	28	GPIO20
			GND	39	40	SCL	21	29	GPIO21