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

Echtzeitverarbeitung

SS 2021
LV 4511 / LV 8481

Übungsblatt 3
Laborversuch
Abgabe: 6. Woche (24.05.2021)

Aufgabe 3.1. (Ereignisgetriebene Aktionen – Kernel-Interrupts):

Im Home-Verzeichnis Ihres Pis finden Sie zwei Kernelmodule, jeweils eine Version für den Kernel mit bzw. ohne RT-Erweiterung. Alternativ sind diese auch noch auf der Lehrveranstaltungswebseite in einer Archivdatei bereitgestellt.

Die Kernelmodule stellen die analoge Funktionalität wie ihr C-Programm aus Aufgabe 2.5 bereit: Taster SW2 (GPIO 27) löst den Interrupt aus und wird GPIO 18 an und wieder ausschalten (Achtung: der GPIO 18 Puls ist nur 7μs lang).

Wiederholen Sie Ihre Messungen. Was sind die Unterschiede?

Aufgabe 3.2. (Zeitgetriebene Aktionen – Abtastrate):

Neben der Frage wie „schnell“ Ereignisse erkannt werden und wie lange es dauert, darauf reagieren zu können, ist es auch wichtig zu wissen, wo grundsätzlich die Grenzen bei der Erkennung von Ereignissen oder Signalen liegen. Oder anders gefragt: wie „schnell“ muss das System sein, damit relevante Ereignisse zuverlässig erkannt werden können.

Ziel dieser Übung ist es, die Frequenz des Signals am OSC-Pin zu messen. Verbinden Sie dazu den Ausgang des Ihnen bereits bekannten Taktgenerators (Pin „OSC“) mithilfe eines Jumper-Kabels mit dem GPIO 20, den Sie als Eingang konfigurieren. Ihr Programm sollte diesen Eingang wiederholt einlesen. Immer, wenn dabei einen „Takt“ (d.h. eine steigende oder fallende Signalflanke) erkannt wird, nehmen Sie einen Zeitstempel. Verwenden Sie dazu die Funktion `unsigned int micros (void)`, die Zeitmarken mit Mikrosekunden-Auflösung liefert. Sobald Sie den nächsten Takt erkennen, bilden Sie die Differenz (das „Delta“) zur zuvor gespeicherten Zeitmarke (Achtung: um den nächsten Takt zu erkennen, müssen Sie das „Tal“ dazwischen erkennen).

Kopieren Sie Ihre Programme aus den Aufgaben 2.4 und 2.5 und passen Sie sie an diese Aufgabe an (d.h. verwenden Sie anstelle von Taster SW2 den GPIO 20 als Eingang). Ertellen Sie sowohl eine Version für aktives Polling, als auch eine Interrupt-gesteuerte Version unter Verwendung der „Software-Interrupt“ Funktionen der WiringPi-Bibliothek. Sorgen Sie dafür, dass bei der Polling-Version die „Schlafzeit“ zwischen den aufeinander folgenden Lesevorgängen konfigurierbar ist. Sehen Sie auch die Möglichkeit vor, gar nicht zu schlafen, sondern das Signal mit der maximal möglichen Rate abzutasten.

Es macht nur für langsame Takte Sinn, die gemessenen Zeitdifferenzen direkt auszugeben. Summieren Sie stattdessen die Deltas auf und führen sie einen Zähler für die Anzahl der erkannten Takte. Damit lässt sich der Durchschnitt berechnen. Erfassen Sie auch das jeweilige Maximum und Minimum. Rechnen Sie die Deltas in die entsprechenden Frequenzen um[1] und geben Sie diese Werte alle 5 Sekunden aus.

Schließen Sie ggf. auch das Oszilloskop zusätzlich am OSC-Pin an, um die Frequenzen zu bestimmen. In diesem Fall können Sie auch bei jedem Polling des Eingangs z.B. auf GPIO 21 einen kurzen Impuls ausgeben und dieses Signal auf dem zweiten Kanal sehen. Auf diese Weise können Sie gut erkennen, an welchen Stellen Ihr Programm das Signal abtastet.

Erhöhen sie die Frequenz mit dem Potentiometer. Stimmen die von Ihrer Software gemessenen Frequenzen mit denen aus Aufg. 2.3 überein? Wie sieht der Unterschied aus, zwischen Polling und der Interrupt-Version.

Bis zu welcher Frequenz wird der richtige Wert erkannt? Wenn Sie dies mit den Ergebnissen von den Aufgaben 2.4 und 2.5 vergleichen, lässt sich eine grobe Abschätzung ermitteln, wie erkennbare Frequenz und mögliche Abtastrate zusammen hängen könnten?

Recherchieren Sie nach der dahinter stehenden Theorie. Wer hat Sie entdeckt? Warum sollte jeder Informatiker darüber grundsätzlich Bescheid wisse?

Wie groß ist die Abweichung zwischen dem theoretisch möglichen Werten und Ihren gemessenen?

Aufgabe 3.3. (Vorbereitung nächste Woche):

(a) Nächste Woche beschäftigen wir uns mit Real-Time und dem Linux-Kernel. Machen Sie sich mit der FAQ des verwendeten Patch-Sets [5] vertraut.

(b) Als API werden wir POSIX Real Time verwenden. Frischen Sie Ihre Betriebssystemkentnisse bezüglich POSIX wieder auf. Finden Sie die API-Aufrufe die notwendig sind, um das Scheduling von Threads beeinflussen zu können.

A. (*):

Literatur

[1] `http://de.wikipedia.org/wiki/Oszilloskop`

[2] `http://www3.physik.uni-stuttgart.de/studium/praktika/ap/pdf_dateien/Allgemeines/OsziAnleitung.pdf`

[3] `http://wiringpi.com/`

[4] `http://wiringpi.com/reference/`

[5] `https://rt.wiki.kernel.org/index.php/Frequently_Asked_Questions`

[1] $Frequenz = \frac{1}{Periodendauer}$, achten Sie aber darauf, dass zu einer vollen Periode immer **zwei** Flanken gehören, eine steigende und eine fallende.

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