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

Echtzeitverarbeitung

SS 2021
LV 4511 / LV 8481

Übungsblatt 1
Laborversuch
Abgabe: 2. Woche (26.04.2021)

Aufgabe 1.1. (Praktikumsumgebung):

Eine ausführliche Beschreibung der Praktikumsumgebung finden auf der Website der Lehrveranstaltung. Arbeiten Sie diese bitte zuerst durch. Folgende Hinweise seien zur Sicherheit hier nochmals wiederholt:

Wichtig: Schießen Sie niemals eine externe Spannungsversorgung und gleichzeitig ein USB-Netzteil an. Dies kann das EchtzeitHat und oder den Raspberry Pi beschädigen.

Achtung: Unter dem Poti ist eine 4-Pin Pfostenleiste, die die verschiedenen Spannungen des Boards herrausführen. Stellen Sie sicher, dass sie niemals die 12V und 5V verbinden! Wenn das passiert, verlässt der Magic-Smoke [3] das EchtzeitHat und ohne diesen können Sie die Aufgaben nicht mehr lösen!

Aufgabe 1.2. (Serielle Konsolenverbindung):

Schließen Sie hierzu ein Micro-USB-Kabel an die entsprechende Buchse des EchtzeitHats (nicht an die Buchse des Raspberry Pi selbst!). Verwenden Sie ein Terminalemulationsprogramm wie `minicom` oder `picocom` mit folgenden Einstellungen:

```
Speed (baud rate): 115200
Bits: 8
Parity: None
Stop Bits: 1
Flow Control: None
```

Alternativ können Sie (unter Linux und MacOS[1] auch `screen` verwenden:

```
screen <Pfad zum tty> 115200
```

[1] Unter MacOS verwenden Sie das cu-Device `dev/cu.usbserial-<*>`. Ggf. müssen Sie noch den FTDI-Treiber installieren.

Verbinden Sie sich mit dem Pi.

```
Username: pi
Passwort: raspberry
```

Probieren Sie ein paar Linux/Unix-Kommandos aus. Sie werden alle Programme auf dem Pi direkt entwickeln, übersetzen und ausführen. Das erspart es Ihnen, eine Cross-Entwicklungsumgebung aufsetzen zu müssen.

Aufgabe 1.3. (GPIOs über sysfs steuern):

Es ist möglich, die GPIOs über das Sys-Filesystem zu steuern, welches vom Kernel im Userspace angeboten wird. Die GPIOs müssen zuvor konfiguriert werden. Das soll zuerst getestet werden. Die Kommunikation erfolgt über die shell mit den Befehlen: `cat` und `echo`. Die GPIO-Struktur liegt unter `/sys/class/gpio`

Zuerst muss der GPIO aktiviert werden. Dafür wird die GPIO-ID in `/sys/class/gpio/export` geschrieben. Danach erzeugt der Kernel ein neues Verzeichnis für diesen GPIO mit weiteren Pseudo-Dateien für die weitere Kontrolle, wovon „active_low“, „direction“, „edge“ und „value“ primär relevant sind.

Aktivieren des GPIO:

```
echo <ID> > /sys/class/gpio/export
```

Setzen der Richtung des GPIO („in“ == Eingang, „out“ == Ausgang):

```
echo in >/sys/class/gpio/gpio<ID>/direction
echo out >/sys/class/gpio/gpio<ID>/direction
```

Setzen des Zustands des GPIO:

```
echo 1 >/sys/class/gpio/gpio<ID>/value
echo 0 >/sys/class/gpio/gpio<ID>/value
```

Pin-Logik des GPIO invertieren

```
echo 0 >/sys/class/gpio/gpio<ID>/active_low
echo 1 >/sys/class/gpio/gpio<ID>/active_low
```

Die GPIOs 20 und 21 sind auf dem EchtzeitHat mit einer LED nach außen geführt. Steuern Sie die GPIOs über die Kommandozeile.

Aufgabe 1.4. (GPIO Werte lesen über sysfs):

Die Taster sind über die GPIOs 27 und 22 angebunden. Aktivieren Sie diese als Eingang und lesen Sie den Wert aus.

Wiederholen Sie dies, wenn die Taster geschlossen sind.

Konfigurieren Sie den GPIO so, dass die Logik invertiert wird (also gegenteiligen Wert von zuvor bei gedrückt / nicht gedrückt).

Aufgabe 1.5. (Geschwindigkeit der GPIOs über sysfs):

Konfigurieren Sie den GPIO 21 als Ausgang und bauen Sie eine Bash-Schleife, die den Port so schnell wie möglich an- und ausschaltet.

Wir verwenden in diesen Aufgaben Digital-Oszilloskope, um Messungen durchzuführen. Wenn Sie nicht wissen, was ein Oszilloskop ist und wofür man es normalerweise verwendet, dann informieren Sie sich [7].

Für die Experimente daheim werden ADALM2000 von Analog ausgegeben. Die Dokumentation dieser Geräte finden Sie unter [8].

Eine gute Einführung in die Bedienung eines Digital-Oszilloskops finden sie unter [9].

Verbinden die den ersten Kanal des Oszilloskops mit GND und dem Port. Starten Sie ihr Bash-Script und wählen Sie die Autoset-Funktion des Oszilloskops. Schauen Sie sich das Signal an.

Drücken Sie Cursor und verwenden Sie die Drehregler für die Vertical Position, um die vertikalen Linien auf dem Display zu verschieben. Damit können Sie die Dauer zwischen den Linien messen.

- Wie hoch ist die Frequenz mit dem Bash-Script? Messen Sie sie.
- Was fällt Ihnen bei der Betrachtung des Signals auf? Woran könnte das liegen?

Aufgabe 1.6. (GPIO-Programmierung mit C):

Für die Verwendung der GPIOs unter C wird die WiringPi-Bibliothek [10] verwendet. Dieses stellt eine benutzterfreundliche API bereit, um die IOs eines Raspberry Pis zu verwenden. Die Bibliothek ist auf dem bereitgestellten Raspbian-Image bereits installiert.

Machen Sie sich mit der Dokumentation [11] vertraut. Entwickeln Sie ein C-Programm, das die LED von GPIO 21 in einer Schleife an- und ausschaltet und dazwischen eine Sekunde schläft.

Achten Sie darauf, dass WiringPi wieder eine andere Bezeichnung für die GPIOs verwendet. Wählen Sie die richtige ID aus der Tabelle aus.

Aufgabe 1.7. (GPIO-Eingabe mit C):

Entwickeln Sie ein Programm mit WiringPi, das in einer Schleife den Taster SW2 des EchtzeitHat ausliest und, wenn sich etwas ändert, diese Änderung auf der Konsole ausgibt.

Was können Sie beim Betätigen des Tasters als mögliches Problem feststellen? Bzw. welches Problem tritt generell (ggf. nicht hier) bei Tastern auf?

Wiederholen Sie das ganze mit dem einem GPIO (bspw. GPIO 20) und einem Jumper-Kabel. Tritt hier ein Problem auf? Welche Möglichkeiten gibt es das Problem programmatisch zu lösen?

Aufgabe 1.8. (Geschwindigkeit des C-Programms):

Entwickeln Sie wieder ein Programm, dass den GPIO 21 so schnell wie möglich toggelt. Messen Sie wieder die Frequenz mit dem Oszilloskop. Um wie viele Größenordnungen ist das C-Programm schneller?

Was fällt Ihnen noch auf, wenn Sie die Ausgabe auf dem Oszilloskop eine Weile betrachten? Spielen Sie ruhig ein wenig mit dem Single-Shot-Button und suchen Sie Auffälligkeiten. Was finden Sie und was könnte es bedeuten, bzw. wie dazu kommen?

Aufgabe 1.9. (Vorbereitung für die nächste Woche):

(a) Machen Sie sich mit dem Interrupt-System von WiringPi vertraut.

(b) Recherchieren Sie, was Pulsweitenmodulation (PWM) ist, wofür man sie benutzt und wie sie programmatisch umgesetzt werden kann.

A. (*):

Literatur

[1] `https://www.raspberrypi.org/`

[2] `https://github.com/kaibeckmann/EchtzeitHat`

[3] `https://en.wikipedia.org/wiki/Magic_smoke`

[4] `http://www.raspbian.org/`

[5] `https://wwwvs.cs.hs-rm.de/lehre/es20ss/material.html`

[6] `https://pinout.xyz`

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

[8] `https://wiki.analog.com/university/tools/adalm2000/users`

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

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

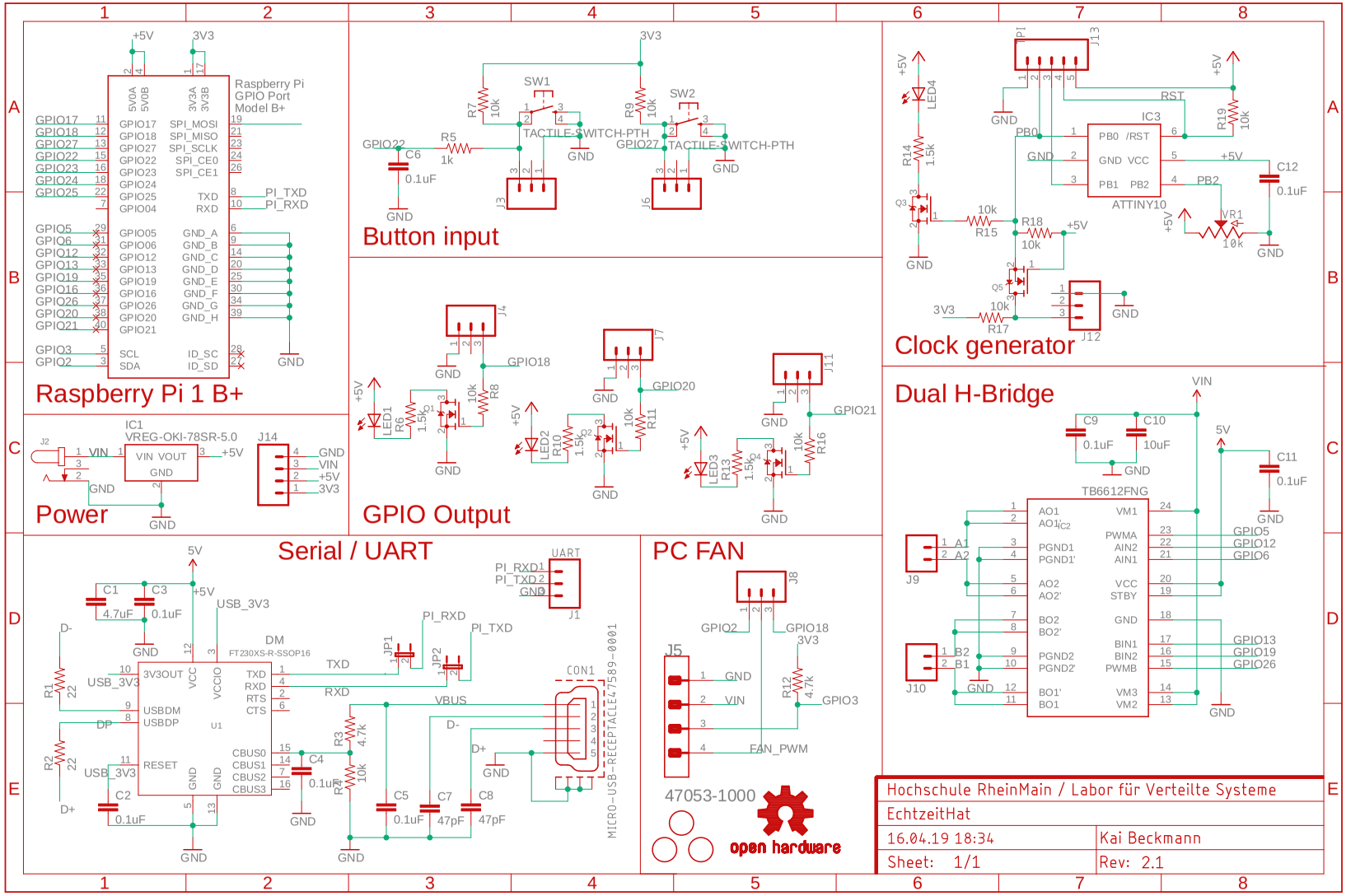
[11] `http://wiringpi.com/reference/`

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

C. (EchtzeitHat Schaltplan):



16.04.19 18:35 f=0.96 /users/k_beckma/ownCloud/EAGLE/projects/EchtzeitHat/EchtzeitHat.sch (Sheet: 1/1)

Abbildung 1: RPiRTHat Schaltplan