64-211 Übung Eingebettete Systeme

Aufgabenblatt 2 Termine: 14.04. + 21.04. / 17.04. + 24.04.

Gruppe	
Name(n)	Matrikelnummer(n)

Bei der Bearbeitung des ersten Aufgabenblattes ist Ihnen sicherlich das **prellende** Verhalten (de.wikipedia.org/wiki/Prellen) der Taster aufgefallen. Da die Taster häufig zum Einsatz kommen werden, soll im ersten Teil des Aufgabenblattes eine funktionierende Lösung für das Problem entwickelt werden.

Das Entprellen eines mechanischen Schalters kann sowohl in Hardware als auch in Software realisiert werden. Ihre Aufgabe ist es eine Software-basierte Lösung zu entwickeln. Da ein konstanter Signalpegel die Grundlage für die Lösung darstellt, sollten Sie in Ihrer Lösung ein Verfahren entwickeln, das Ihnen die benötigte Information liefert. Eine sinnvolle Variante kann durch periodisches Abtasten des Eingangssignals realisiert werden. Hierbei sollen SIe einen Hardware-Timer verwenden, welcher, entsprechend parametrisiert, periodisch einen Interrupt auslöst. Die Interrupt-Behandlungsroutine soll verwendet werden um den Signalzustand zu bestimmen und Information an die Hauptschleife zu überreichen.

Aufgabe 2.1

Das Ziel dieser Aufgabe ist das Kennenlernen und Verwenden von Hardware-Timern. Betrachten Sie das Datenblatt des auf dem Arduino Due verwendeten Mikrocontrollers (`SAM3X8E`) unter: tams.informatik.uni-hamburg.de/lectures/2015ss/vorlesung/es/doc/sam3x8e.pdf. Schauen Sie sich insbesondere das Kapitel 37 (ab Seite 869) an.

Um (unter anderem) die Konfiguration und Verwendung der Hardware-Timer zu vereinfachen enthält das Arduino Framework eine von Atmel für die SAM-Reihe der Mikrocontroller zur Verfügung gestellte Funktionsbibliothek: **libsam**. Der Quellcode dieser Bibliothek kann unter github.com/arduino/Arduino/tree/master/hardware/arduino/sam/system/libsam eingesehen werden. Beachten Sie insbesondere die Dateien `tc.h`/`tc.c`, welche entsprechende Hilfsfunktionen für die Konfiguration und Verwendung der Timer enthalten.

Der Mikrocontroller des Arduino Due besitzt **3 frei nutzbare Timer-Blöcke** (`TC0`, `TC1` und `TC2`) mit jeweils **3 separaten Kanälen** (`0` bis `2`) und **32-Bit Zählern**. Somit stehen Ihnen - wie in der folgenden Tabelle dargestellt - 9 Timer zur Verfügung.

```
TC        Ch        NVIC "irq"        IRQ handler        PMC id
--------------------------------------------------------------------------
TC0       0         TC0_IRQn          TC0_Handler        ID_TC0
TC0       1         TC1_IRQn          TC1_Handler        ID_TC1
TC0       2         TC2_IRQn          TC2_Handler        ID_TC2
TC1       0         TC3_IRQn          TC3_Handler        ID_TC3
TC1       1         TC4_IRQn          TC4_Handler        ID_TC4
TC1       2         TC5_IRQn          TC5_Handler        ID_TC5
TC2       0         TC6_IRQn          TC6_Handler        ID_TC6
TC2       1         TC7_IRQn          TC7_Handler        ID_TC7
TC2       2         TC8_IRQn          TC8_Handler        ID_TC8
```

Verwenden Sie bitte in Ihrer `setup()`-Routine folgenden "Boilerplate"-Code für die Konfiguration des Hardware-Timers:

```
pmc_set_writeprotect(false);
pmc_enable_periph_clk( >> PMC id des Timers << );

>> Hier erfolgt die Konfiguration des Timers

NVIC_ClearPendingIRQ( >> NVIC irq des Timers << );
NVIC_EnableIRQ( >> NVIC irq des Timers << );

>> Hier wird der konfigurierte Timer gestartet
```

Hinweise zur Konfiguration:

- Konfigurieren Sie den Timer im `waveform mode` (auch wenn das erzeugte Rechtecksignal nicht verwendet werden wird).
- Der Timer soll den Zähler hochzählen und bei positiver Übereinstimmung mit dem Vergleichswert diesen automatisch zurücksetzen können (`automatic RC trigger`).
- Setzen Sie einen Vergleichswert um eine Event-Frequenz von 1kHz zu erhalten.
- Aktivieren Sie für den gewählten Timer das `RC-compare`-Interrupt, deaktivieren Sie die restlichen Interrupt-Typen.

Wichtig: Lesen Sie in der zugehörigen Interruptbehandlungsroutine das Status-Register des von Ihnen gewählten Timer-Kanals. Das Lesen des Registers löscht interne "Flags", wovon eines das Interrupt selbst signalisiert. Wird dieses Flag nicht gelöscht, werden nachfolgende Interrupts nicht ausgelöst.

Wählen Sie ausgehend von der Konfiguration des Timers einen sinnvollen Zeitraum für die Betrachtung des Signals. Bedenken Sie, dass es notwendig ist beide Zustände des Eingangssignals (Taster betätigt / Taster losgelassen) zu betrachten. Für die im Rahmen der Übung verwendeten Taster hat sich gezeigt, dass die Oszillationsdauer zwischen `10ms` und `30ms` liegt.

Verwenden Sie den Aufbau aus dem ersten Aufgabenblatt und erweitern Sie Ihr Programm aus Aufgabe 1.5(b) mit der in dieser Aufgabe erstellten Lösung.

Aufgabe 2.2

Bei dieser Aufgabe gilt es einen Gleichstrommotor nach einem bestimmten Schema anzusteuern. Für diesen Zweck wird ein integrierter Baustein verwendet, der die Funktionsweise einer **H-Brücke** bereitstellt. Informationen zum Funktionsprinzip einer H-Brücke finden Sie unter: en.wikipedia.org/wiki/H_bridge. Der für die Lösung der Aufgabenstellung zu verwendende Baustein **TB6612FNG** enthält zwei solcher Brücken.

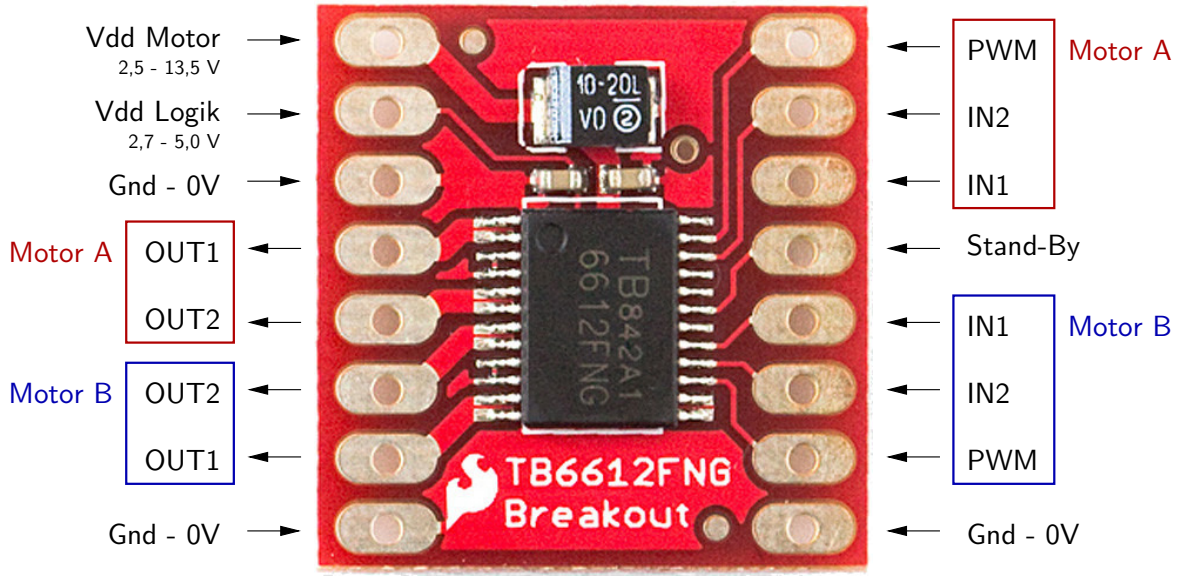

Abbildung 1: Anschlussbelegung des TB6612FNG Breakout-Boards.

Informieren Sie sich über die Anschluss-Belegung (Abb. 1) und betrachten Sie das entsprechende Datenblatt (tams.informatik.uni-hamburg.de/lectures/2015ss/vorlesung/es/doc/tb6612fng.pdf).

Für die Lösung der Aufgabe wird empfohlen den Versuchsaufbau gemäß Abbildung 2 zu verdrahten. Beachten Sie bei der Verdrahtung folgende Hinweise:

- Stellen Sie zunächst die Stromversorgung des des TB6612FNG Bausteins sicher. „Vdd Motor“ ist der Anschluss für die Versorgungsspannung des Motors (nutzen Sie hier bitte die vom Arduino Due Board bereitgestellten 5V). „Vdd Logik“ ist der Spannungsanschluss für die Logik (hier sind 3.3V ausreichend). Alle Gnd-Anschlüsse sind intern verbunden.
- Da der TB6612FNG Baustein zwei H-Brücken enthält, haben Sie die Wahl zwischen zwei Kanälen: Kanal A (rot gekennzeichnet) und Kanal B (blau markiert). Benutzen Sie alle fünf Anschlüsse eines der beiden Kanäle.
- Verbinden Sie den Kontrollanschluss „PWM“ mit einem PWM-fähigen Pin des Mikrocontrollers. Die Eingänge „IN1“ und „IN2“ ermöglichen die Wahl zwischen den im Datenblatt beschriebenen Steuerzuständen. Die Anbindung des Gleichstrommotors muss über die Anschlüsse „OUT1“ und „OUT2“ erfolgen.

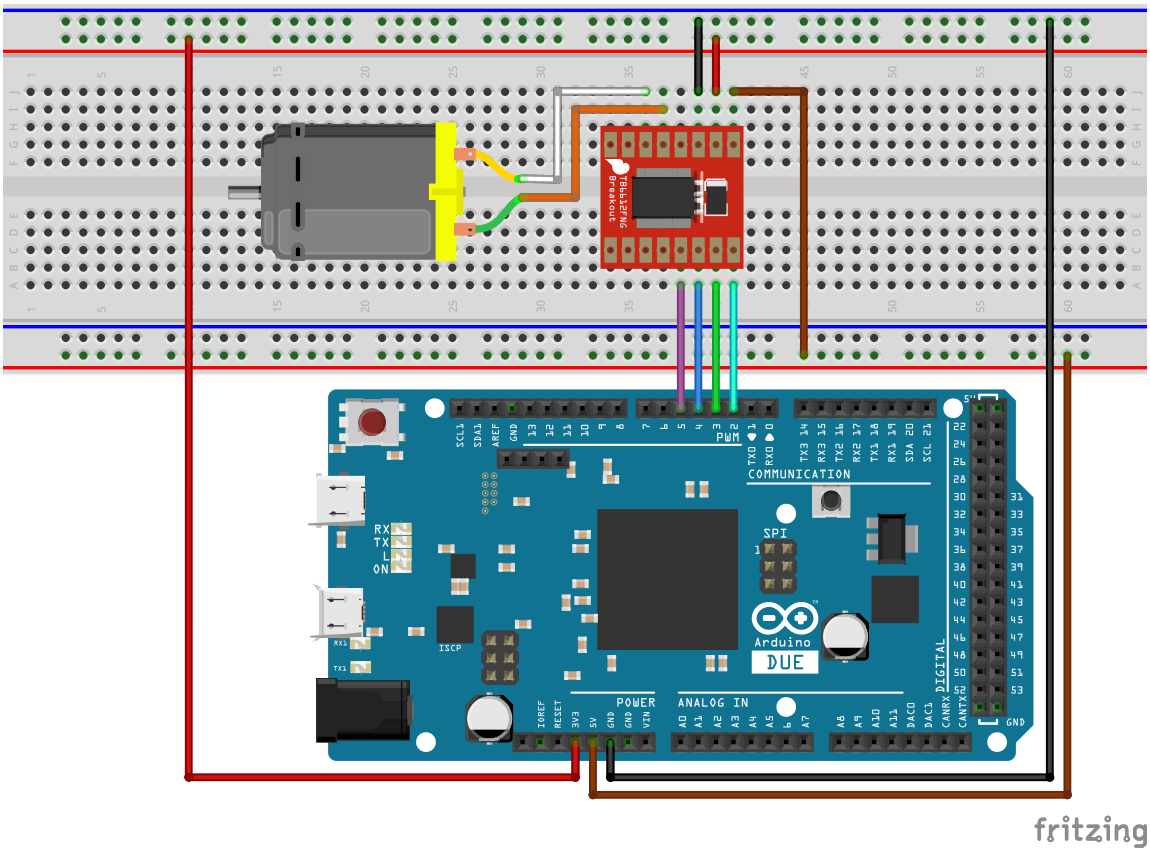

Abbildung 2: Vorschlag für die Verdrahtung des Versuchsaufbaus.

- Vergessen Sie nicht Stand-By mit einem definierten Pegel zu versehen:
  Stand-By = `HIGH` schaltet den Baustein ein
  = `LOW` schaltet diesen aus

Ihre Aufgabe ist es ein Programm für den Mikrocontroller zu erstellen, das den Gleichstrommotor nach folgendem Schema ansteuert:

1. Rotieren Sie den Motor im Uhrzeigersinn (CW). Fahren Sie dabei die Leistung innerhalb eines Zeitraums von 5 Sekunden gleichmäßig von 0% auf 100%.
2. Stoppen Sie den Motor indem Sie die Leistung schnell, jedoch gleichmäßig reduzieren.
3. Ändern Sie die Rotationsrichtung und rotieren Sie den Motor entgegen dem Uhrzeigersinn (CCW). Gehen Sie bei der Leistungsregelung analog zu 1. und 2. vor.

– Fahren Sie mit 1. fort.

Alle für die erfolgreiche Bearbeitung der Aufgabe notwendigen Funktionen sollten Ihnen bereits aus dem vorherigen Aufgabenblatt bekannt sein.

Aufgabe 2.3

Erweitern Sie die Lösung der vorherigen Aufgabe indem Sie in Ihren Versuchsaufbau zwei Taster einbinden und folgende Funktionsweise implementieren:

- Ein Taster schaltet (zyklisch) zwischen folgenden Zuständen des Systems um:
  1. Rotation im Uhrzeigersinn
  2. Rotation entgegen dem Uhrzeigersinn
  3. Gleichmäßiger Motorstopp.
- Der zweite Taster soll die Leistung wie folgt regeln:
  1. Steigerung der Leistung von 0% bis 100%
  2. Absenkung der Leistung von 100% bis 0%

Implementieren Sie die Lösung dieser Aufgabe indem Sie die in Aufgabe 2.1 entwickelte Lösung zum Entprellen mit einbeziehen.