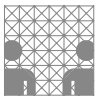
64-211 Übung Eingebettete Systeme

Aufgabenblatt 04 Termine: KW 20, KW 22



Gruppe	
Name(n)	Matrikelnummer(n)

4 Steuerung externer Komponenten

Die Steuerung von Aktoren ist ein Standard-Einsatzfall für eingebettete Systeme. Dabei werden verschiedene Typen von Aktoren zur Erledigung unterschiedlicher Aufgaben eingesetzt. Die Beispiele reichen von sehr kleinen Anwendungen (Steuerung des Antriebs eines RC-Fahrzeugs, Steuerung der Ruder eines Flugzeugs etc.) bis hin zu hochkomplexen verteilten Systemen (Prozesskontrolle in industriellen Fertigungsstraßen, Regelung des Kühlkreislaufs eines Atomkraftwerks etc.).

Gegenstand dieser Aufgabe ist die Entwicklung der Steuerung eines Gleichstrommotors. In Aufgabe 4.1 werden Sie ein über lediglich zwei Taster zu bedienendes Benutzerinterface und den für die Ablaufsteuerung erforderlichen Automaten entwickeln, während Sie in Aufgabe **??** die Ausgaben an den Benutzer implementieren und in Aufgabe 4.3 schließlich die Ansteuerung des Treiber-ICs des Motors realisieren werden.

Aufgabe 4.1 Steuerung eines Gleichstrommotores

Entwickeln Sie ein Programm, das die Ansteuerung eines Gleichstrommotors ermöglicht. Implementieren Sie dafür eine Benutzerschnittstelle, die es erlaubt, mit nur zwei Tastern sowohl die Leistung des Motors zu regeln, als auch zwischen den Rotationsrichtungen umzuschalten. Ihren Code aus Aufgabe 3.3 bzw. 3.4 zum Entprellen der Taster und zur Erkennung einer Doppel- bzw. Parallelbetätigung der Taster können Sie unmittelbar wiederverwenden.

Erstellen Sie ein Programm für den Mikrocontroller, das eine Steuerung des Gleichstrommotors nach folgendem Schema ermöglicht:

- Doppelbetätigung der Taster
  Wenn beide Taster gleichzeitig betätigt werden, dann soll zyklisch zwischen den beiden Zuständen (Bedien-Modi) umgeschaltet werden:
  1. *Einstellung der Rotationsrichtung*
  2. *Einstellung der Leistung*

- Einzelbetätigung der Taster
  1. Im Zustand *Einstellung der Rotationsrichtung* haben die Taster folgende Funktion:[1]
     - Taster 1 verändert die Rotationsrichtung wie folgt: $CW \rightarrow STOP \rightarrow CCW$
     - Taster 2 verändert die Rotationsrichtung wie folgt: $CW \leftarrow STOP \leftarrow CCW$
  2. Im Zustand *Einstellung der Leistung* haben die Taster die Funktion:
     - Taster 1 regelt die Leistung um eine von Ihnen definierte Schrittweite hoch, und nach Erreichen des Maximalwertes wird dort verblieben.
     - Taster 2 verhält sich entgegengesetzt zur Funktion des Tasters 1.

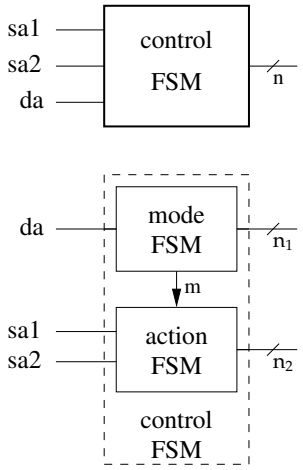


Abbildung 1: Zerlegung eines Automaten
(sa1 – single button1
sa2 – single button2
da – double actuation
$n = n_1 + n_2$)

Die **Initialisierung des Systems** soll im Modus *Einstellung der Rotationsrichtung* mit der Rotationsrichtung *STOP* erfolgen.

Für die Umsetzung des obigen Verhaltens ist es wieder naheliegend, einen endlichen Automaten, wie bereits in Aufgabe 3.3 geschehen, zu implementieren. Entwerfen Sie hierfür wieder zu Beginn das Zustandsdiagramm des Au-

[1] *CW* clock-wise : im Uhrzeigersinn
CCW counter clock-wise : gegen den Uhrzeigersinn

tomaten. Bedenken Sie hierbei, dass es nicht zwingend notwendig ist, einen „großen“Automaten zu entwerfen, sondern diesen, wie es die Aufgabenstellung bereits suggeriert, in zwei gekoppelte Automaten nach Abbildung 1 zu zerlegen.

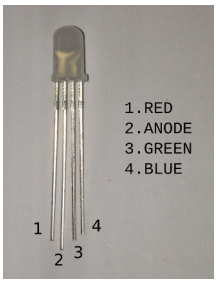
Aufgabe 4.2 Ausgaben des Systems

Ausgaben des Systems sollen über eine Dreifarben-LED und über die serielle Schnittstelle getätigt werden.

Die RGB-LED soll zur Anzeige des aktuellen Bedien-Modus des Systems eingesetzt werden:

1. *Einstellung der Rotationsrichtung* = „Grün“
2. *Einstellung der Leistung* = „Rot“
3. Ist in der Rotationsrichtung *STOP* gewählt, soll zusätzlich noch „Blau“ aktiviert werden.

Beachten Sie die Anordnung der Anschlüsse der **RGB-LED** (siehe Abbildung 2). Die Farbkanäle (Red, Green, Blue) der LED besitzen eine **gemeinsame Anode** (der „Pluspol“). Schließen Sie diese an `VCC` an. Schließen Sie jeden einzelnen Kanal über den zugehörigen Vorwiderstand an einen Ausgangspin des Arduinos an. Wenn Sie beliebige Farben mischen möchten, können Sie auch **PWM-fähige** Anschlusspins des Arduinos verwenden und die Intensität der drei Farbkanäle über PWM-Signale steuern. Das ist für diese Aufgabe aber nicht erforderlich.
Da bei der Simulation mit Proteus die PWM-Signale nicht in Realzeit generiert werden können, verwenden Sie bitte bei den Vorwiderständen und den LEDs das digitale Simulationsmodell (Edit Properties/Model Type - DIGITAL). Dann wird die langsam voranschreitende Simulationszeit berücksichtigt und die Helligkeit der LED angepasst. Sonst würde die LED entsprechend der voranschreitenden Simulationszeit im Takt des PWM-Signals blinken. Darüberhinaus weist das Simulationsmodell der RGB-Led in Proteus (RGBLED-CA) insbesondere beim Mischen von Farben Unsauberkeiten auf, so dass unplausible Farben entstehen. Deshalb sollten Sie für die Simulation auf die Proteus RGB-Led verzichten und anstelle dessen – wie in Abb. 7 gezeigt – drei einzelne LEDs verwenden.
Welche Auswirkung hat die gemeinsame Anode der RGB-LED auf die Ansteuerung der drei Farbkanäle mit der `analogWrite()` bzw. `digitalWrite()` Funktion?



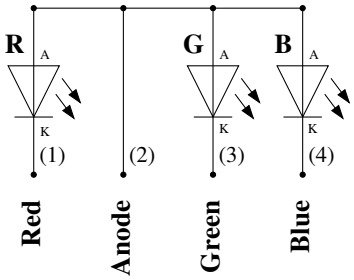


Abbildung 2: Anordnung der Anschlüsse der RGB-LED.

Benutzen Sie für den Anschluss der RGB-LED in Abhängigkeit der Versorgungsspannung V_{CC} Ihres Aufbaues folgende Vorwiderstände:[2]

		V_{CC}	
Farbe	U_D	3.3 V	5.0 V
Red	1,7 V	270 Ω	560 Ω
Green	3,1 V	27 Ω	330 Ω
Blue	3,1 V	27 Ω	330 Ω
		$I_D \approx 6mA$	

Tabelle 1: Die Vorwiderstände der jeweiligen Farben werden zum Schutz der Ausgangstreiber des Arduino für einen Durchlassstrom (I_D) von $\approx 6mA$ (vgl. Datenblatt RGB-Led) bemessen. Bei der Simulation mit Proteus achten Sie bitte darauf, dass Durchlassspannung (U_D) und Durchlassstrom (I_D) entsprechen der obigen Tabelle gewählt wurden. Im Proteus Template-Projekt ist dieses bereits geschehen.

Für die **Ausgaben des Systems** über den seriellen Port gilt:

- Informieren Sie den Benutzer immer über den Zustand des Systems, indem Sie Veränderungen der Kontrollwerte über den seriellen Monitor der Arduino IDE (sinnvoll präsentiert) ausgeben. Sollte eine Aktion des Benutzers mit keiner Veränderung der Kontrollwerte in Verbindung stehen, z.B. Veränderung der Leistung während Rotationsrichtung *STOP* eingestellt ist, erzeugen Sie eine entsprechende Ausgabe auf der seriellen Konsole.

  In Abb. 3 finden Sie ein Beispiel dafür, wie diese Information präsentiert werden könnte.

[2] Vgl. Sie hierzu die Aufgabe 1.2 des Aufgabenblattes 1 und berücksichtigen die unterschiedlichen Durchlassspannungen U_D der RGB-Dioden (vgl. Datenblatt).

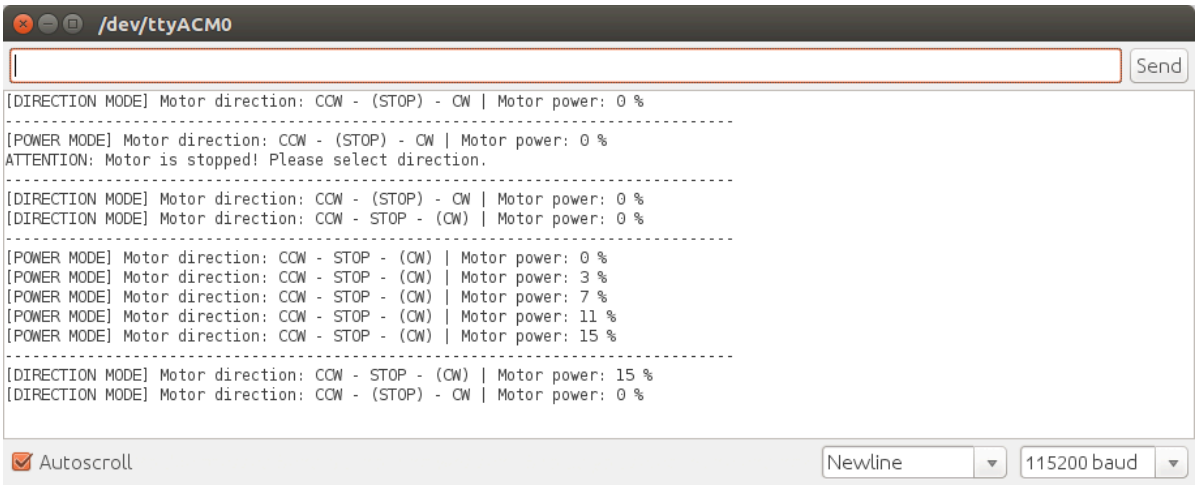


Abbildung 3: Kommunikation des Systemzustandes über den seriellen Monitor.

Aufgabe 4.3 Ansteuerung des Gleichstrommotors

Für die unmittelbare Ansteuerung des Gleichstrommotors verwenden wir ein Arduino Erweiterungsboard, das auf dem integrierten Schaltkreis **TB6612FNG** basiert. Dieses IC enthält die Steuerlogik und die Leistungsendstufen zur Ansteuerung der Motoren. Die Leistungsendstufen sind in Form einer **H-Brücke** verschaltet. Nähere Informationen zum Funktionsprinzip einer H-Brücke finden Sie unter: `en.wikipedia.org/wiki/H_bridge`.

Informieren Sie sich über die Anschlussbelegung (siehe Abbildung 5) und lesen Sie das zugehörige Datenblatt. Für die Lösung der Aufgabe wird nahegelegt, den Versuchsaufbau gemäß Abbildung 4 zu verdrahten. Für die Simulation mit Proteus verwenden Sie bitte anstelle des Breakout-Boards mit dem Motortreiber TB6612FNG direkt das Device `TB6612FNG` (siehe auch: Proteus Template-Projekt).

Beachten Sie bei der Verdrahtung des Breakout-Boards bzw. des TB6612FNG Bausteins folgende Hinweise:

- Stellen Sie zunächst die Stromversorgung des TB6612FNG Bausteins sicher. Der Anschluss `VCC Motor` des Breakout-Boards bzw. die Pins 24, 14 und 13, also `VM1`, `VM2` und `VM3` des TB6612FNG-IC dienen der Versorgung der Motortreiber (**2.5 – 13.5 V**), während `VCC Logik` bzw. Pin 20 (`Vcc`) die Spannungsversorgung der Logik (**2.7 V – 5.5 V**) darstellt. Auf dem Breakout-Board sind alle `GND`-Anschlüsse boardseitig verbunden, während seitens des TB6612FNG-IC Motortreiber-GND (Pin 3, 4 - `PGND1` und Pin 9, 10 - `PGND2`) und Logik-GND (Pin 18 - `GND`) getrennt sind.
- Da der TB6612FNG Baustein zwei H-Brücken enthält, haben Sie die Wahl

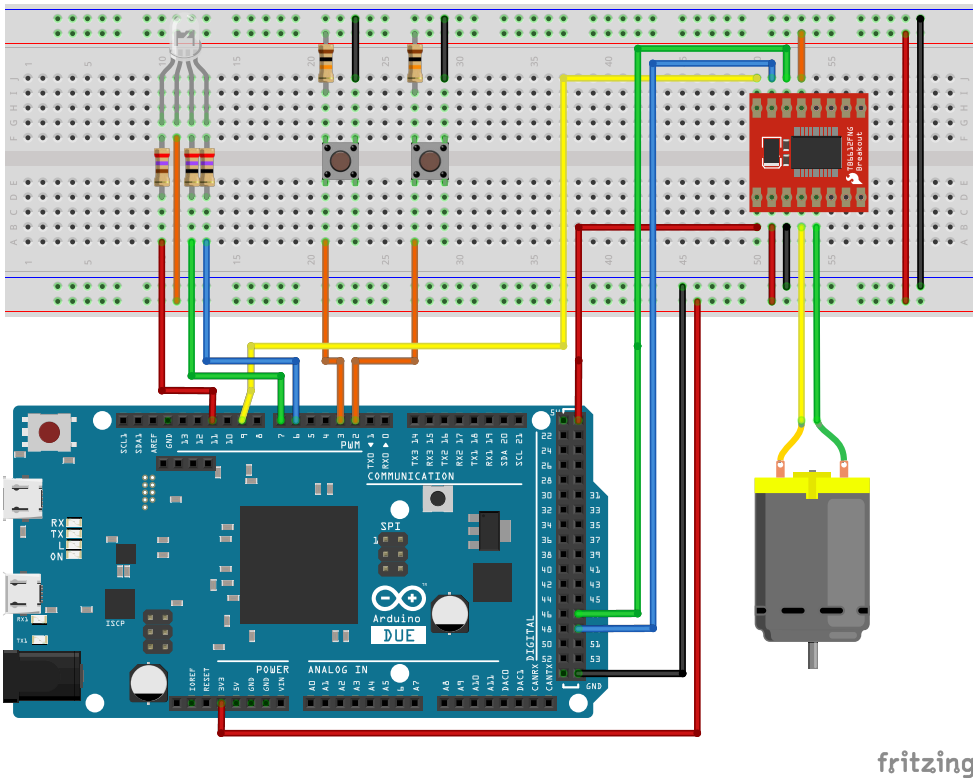


Abbildung 4: Vorschlag für die Verdrahtung des Versuchsaufbaus.

zwischen zwei Kanälen: Kanal A (rot markierte Anschlüsse des Breakout-Boards, bzw. die Anschlüsse `AO1`, `AO2`, `PWMA`, `AIN1`, `AIN2`) und Kanal B (blau markierte Anschlüsse des Breakout-Boards, bzw. die Anschlüsse `BO1`, `BO2`, `PWMB`, `BIN1`, `BIN2`). Benutzen Sie einen der beiden Kanäle.

- Verbinden Sie den Kontrollanschluss PWM mit einem PWM-fähigen Anschlusspin des Mikrocontrollers. Stellen Sie sicher, dass der vom Microcontroller für die Generierung des PWM-Signals des von Ihnen gewählten Anschlusspins vorgesehenen Timers nicht bereits durch Ihr Programm verwendet wird (siehe hierzu Aufgabenblatt 3, Abschnitt: 3.1). Die Eingänge IN1 und IN2 ermöglichen die Einstellung der im Datenblatt beschriebenen Steuerzustände. Die Anbindung des Gleichstrommotors muss über die Anschlüsse A OUT1,2 oder B OUT1,2 bzw. `AO1,2` oder `BO1,2` erfolgen.
- Vergessen Sie nicht, Stand-By bzw. `STBY` mit einem definierten Pegel zu versehen (entweder festverdrahtet, oder über digitalen Ausgabepin):
  Stand-By = `HIGH` schaltet den Baustein ein
  = `LOW` schaltet diesen aus

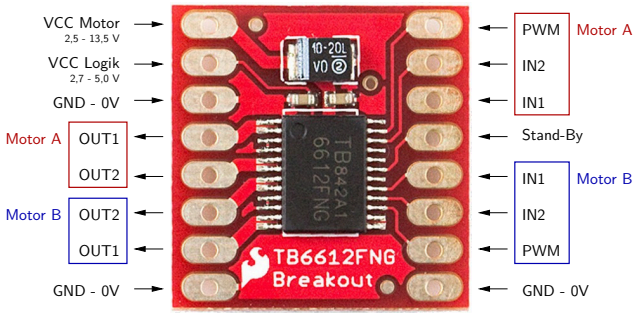


Abbildung 5: Anschlussbelegung des TB6612FNG Breakout-Boards.

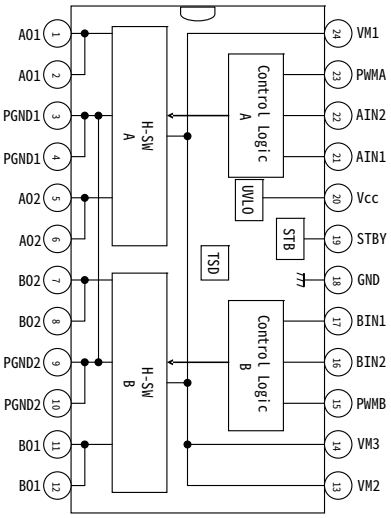


Abbildung 6: Blockdiagramm des TB6612FNG-Motortreibers

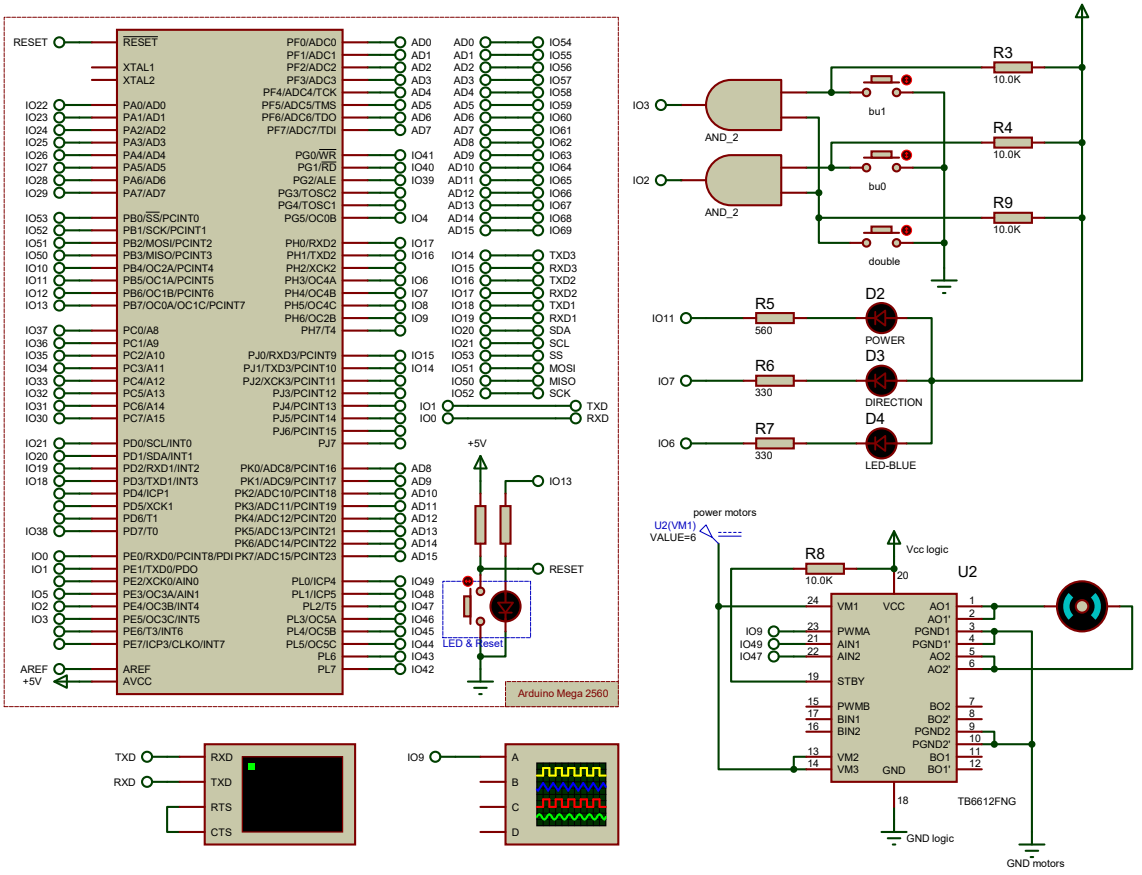


Abbildung 7: Schematic des Proteus Template-Projektes