64-211 Übung Eingebettete Systeme

Aufgabenblatt 1 Termine: 01.04. / 04.04.

Gruppe	
Name(n)	Matrikelnummer(n)

Im Rahmen der Übung werden Sie Aufgaben mit dem **Arduino Due** Microcontroller-Board lösen (arduino.cc/en/Main/ArduinoBoardDue). Abgesehen von einer vollkommen anderen Microcontroller-Architektur (8-bit AVR vs. 32-bit ARM) ist folgendes Merkmal besonders wichtig: **die maximale, an den I/O-Eingängen anliegende, Spannung darf unter keinen Umständen 3,3 V überschreiten**! Eine höhere Spannung führt in der Regel zur Beschädigung des Microcontrollers.

- Machen Sie sich bitte mit der Pinbelegung des **Arduino Due** vertraut, bevor Sie mit der Lösung der Aufgaben beginnen: tams.informatik.uni-hamburg.de/lectures/2014ss/vorlesung/es/doc/duePinout.pdf.
- Zur Programmierung muss eine Version der Arduino IDE (Programmierumgebung) benutzt werden, die mit „`$tamsSW/arDue/arduino`“ gestartet wird, siehe auch tams.informatik.uni-hamburg.de/lectures/2014ss/vorlesung/es/uebung.

Genereller Hinweis: Es wird empfohlen jedes als Lösung erstellte Programm separat zu speichern. Dieses erleichtert zum einen die Kontrolle der Lösung, zum anderen hilft es Ihnen bei der Bearbeitung von Aufgaben die teilweise Aufgabenblatt-übergreifend aufeinander aufbauen.

Für die Lösung der vorliegenden Aufgaben wird nahegelegt den Versuchsaufbau gemäß Abbildung 1 zu verdrahten. Mit dem dargestellten Aufbau lassen sich alle Teilaufgaben ohne Änderungen an der Verdrahtung umsetzen.

Aufgabe 1.1 (Punkte 15)

Ihre erste Aufgabe ist es ein Programm für den ARM-Microcontroller unter Verwendung der Arduino IDE zu erstellen, das die auf dem Steckbrett platzierte LED nach folgendem Schema ansteuert:

1. Schalten Sie die LED für 2 Sekunden ein.
2. Schalten Sie die LED für 500 Millisekunden aus.

– Fahren Sie mit 1. fort.

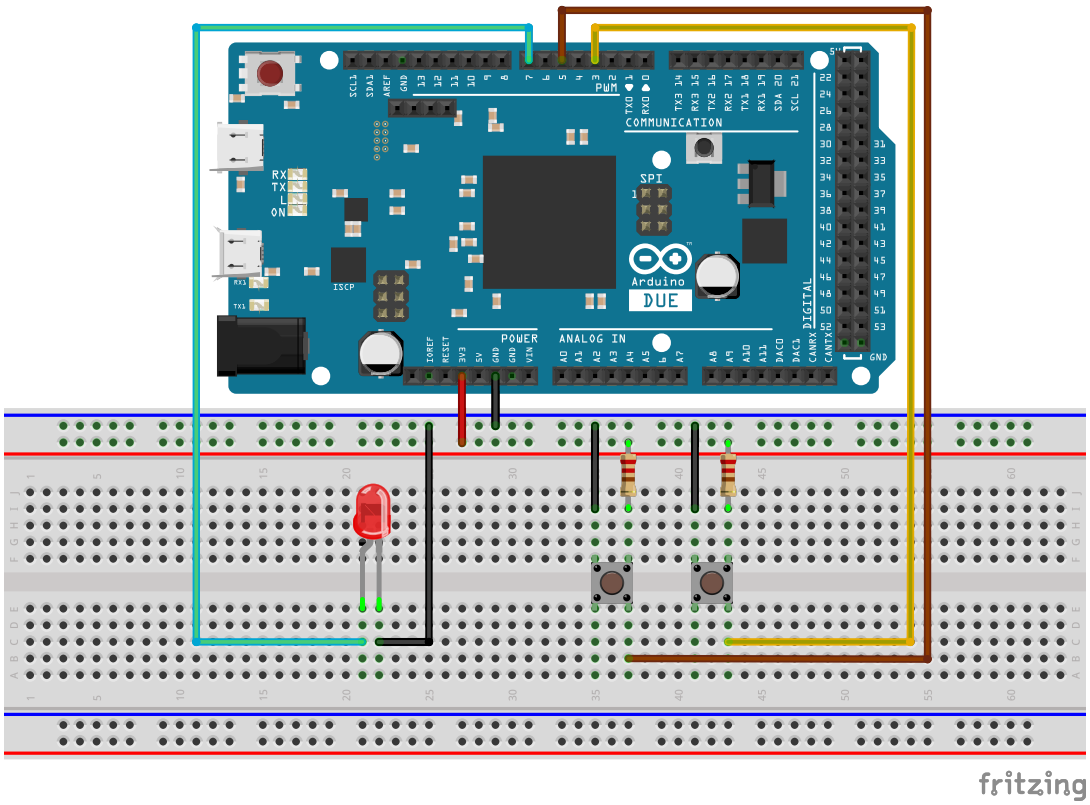

Abbildung 1: Vorschlag für die Verdrahtung des Versuchsaufbaus

Die Programmstruktur entspricht folgendem Template:

```
void setup()
{
  // Anweisungen für die initiale Konfiguration des ARM-Controllers
}

void loop()
{
  // Anweisungen, die der ARM-Controller in der Schleife ausführen soll
}
```

Folgende Funktionen sind bei der Bearbeitung der Aufgabe hilfreich:

* pinMode(⟨*pin*⟩, ⟨*mode*⟩)	arduino.cc/en/Reference/pinMode
* digitalWrite(⟨*pin*⟩, ⟨*value*⟩)	arduino.cc/en/Reference/digitalWrite
* delay(⟨*ms*⟩)	arduino.cc/en/Reference/delay

Aufgabe 1.2 (Punkte 15)

Aufbauend auf der ersten Aufgabe, soll ein externer Taster angebunden werden, dessen Aufgabe es ist die externe LED ein- und auszuschalten.

Beachten Sie bitte, dass der von Ihnen gewählte Anschluss-Pin für den Taster zu jeder Zeit einen definierten Zustand aufweisen muss, d.h. entweder LOW oder HIGH. Deshalb müssen Sie den Taster entsprechend mit einem pull-up bzw. pull-down Widerstand (rn-wissen.de/index.php/Pullup_Pulldown_Widerstand) ausstatten (siehe Abbildung 1).

Zusätzlich zu den bereits bekannten Funktionen, ist folgende Funktion bei der Bearbeitung dieser Aufgabe hilfreich:

* `digitalRead(⟨pin⟩)` arduino.cc/en/Reference/digitalRead

Alternativ oder ergänzend zu dem oben skizzierten Vorgehen lässt sich eine Lösung entwickeln, die einen internen pull-up Widerstand nutzt und somit keinen externen Widerstand am Schalter benötigt. Konsultieren Sie hierzu arduino.cc/en/Tutorial/DigitalPins.

Aufgabe 1.3 (Punkte 20)

Die in der vorigen Aufgabe entstandene Lösung, bei der der Taster mit `digitalRead` abgefragt wird, hat einen prinzipiellen Nachteil. Welcher ist das?

Entwerfen Sie die Behandlung des Tasters mit einer Interruptsteuerung. Informieren Sie sich dazu in der Arduino Dokumentation (arduino.cc/en/Reference/AttachInterrupt), wie die Arduino-Interruptbehandlung funktioniert. Benutzen Sie in Ihrem Programm folgende Funktionen:

* `attachInterrupt(⟨pin⟩, ⟨function⟩, ⟨mode⟩)` arduino.cc/en/Reference/attachInterrupt
* `detachInterrupt(⟨pin⟩)` arduino.cc/en/Reference/detachInterrupt

Aufgabe 1.4 (Punkte 20)

Steuern Sie die LED analog, also als Pulsweitenmodulation, an. Mit einer Integer-Variablen soll der Wert in der Schleife ständig inkrementiert und Modulo 256 an die analoge Ausgangsfunktion übergeben werden:

* `analogWrite(⟨pin⟩, ⟨value⟩)` arduino.cc/en/Reference/analogWrite

Vergessen Sie bei Ihrem Programm nicht die `delay` Anweisung in der Schleife. Geben Sie in einer Konsole (serieller Anschluss, der über USB emuliert wird) den Wert der „Helligkeitsvariablen“ aus um das Verhalten des Programms besser überprüfen zu können. Wie in dem Template skizziert, werden für die serielle Kommunikation die `Serial.`-Funktionen benutzt, siehe arduino.cc/en/Reference/Serial.

```
void setup()
{
  ...
  Serial.begin(9600);
}
```

```
void loop()
{
  ...
  Serial.print(⟨val⟩, ⟨format⟩);     // bzw. Serial.print(⟨val⟩)
//...    und/oder
  Serial.println(⟨val⟩, ⟨format⟩);   // bzw. Serial.println(⟨val⟩)
  ...
}
```

Aufgabe 1.5 (Punkte 15+15)

Aufbauend auf den Aufgabenteilen 2 und 3 soll die Einstellung der LED-Helligkeit in dieser Aufgabe mit zwei Tastern geregelt werden, die den PWM-Wert inkrementieren, bzw. dekrementieren. Enwerfen Sie zwei Versionen einen Programms, die jeweils folgendes Verhalten aufweisen:

(a) Direktes Abfragen der Taster in der Hauptschleife und die damit einhergehende Änderung des PWM-Wertes.

(b) Behandlung der jeweiligen Taster in zwei Interruptroutinen, die den PWM-Wert entsprechend beeinflussen.