64-211 Übung Eingebettete Systeme

Aufgabenblatt 1 Termine: 07.04./10.04.

Gruppe	
Name(n)	Matrikelnummer(n)

Im Rahmen der Übungen zur Vorlesung "Eingebettete Systeme" werden Sie Aufgaben mit dem **Arduino Due** Board lösen (arduino.cc/en/Main/ArduinoBoardDue). Im Gegensatz zur weit verbreiteten **8-bit AVR** Architektur, kommt beim Arduino Due Board ein **32-bit ARM** basierter Mikrocontroller zum Einsatz.

Folgendes bitten wir Sie während der gesamten Veranstaltung zu beachten: **Die maximale, an den I/O-Anschlüssen anliegende, Spannung darf unter keinen Umständen 3,3 V überschreiten**! Eine höhere Spannung führt in der Regel zur permanenten Zerstörung des Mikrocontrollers.

Machen Sie sich bitte zunächst mit der **Pinbelegung** des Arduino Due Boards vertraut, bevor Sie mit der Lösung der Aufgaben beginnen. Eine entsprechende Übersicht finden Sie unter: tams.informatik.uni-hamburg.de/lectures/2015ss/vorlesung/es/doc/due_pins.pdf.

Zur Programmierung wird die Arduino Entwicklungsumgebung (arduino.cc/en/Guide/Environment) verwendet, die auf den Rechnern des Arbeitsbereiches TAMS über ein Terminal mit dem Befehl "`$tamsSW/arduino-1.6.1/arduino`" gestartet wird.

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 folgenden 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

Ihre erste Aufgabe ist es ein Programm für den Mikrocontroller 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.

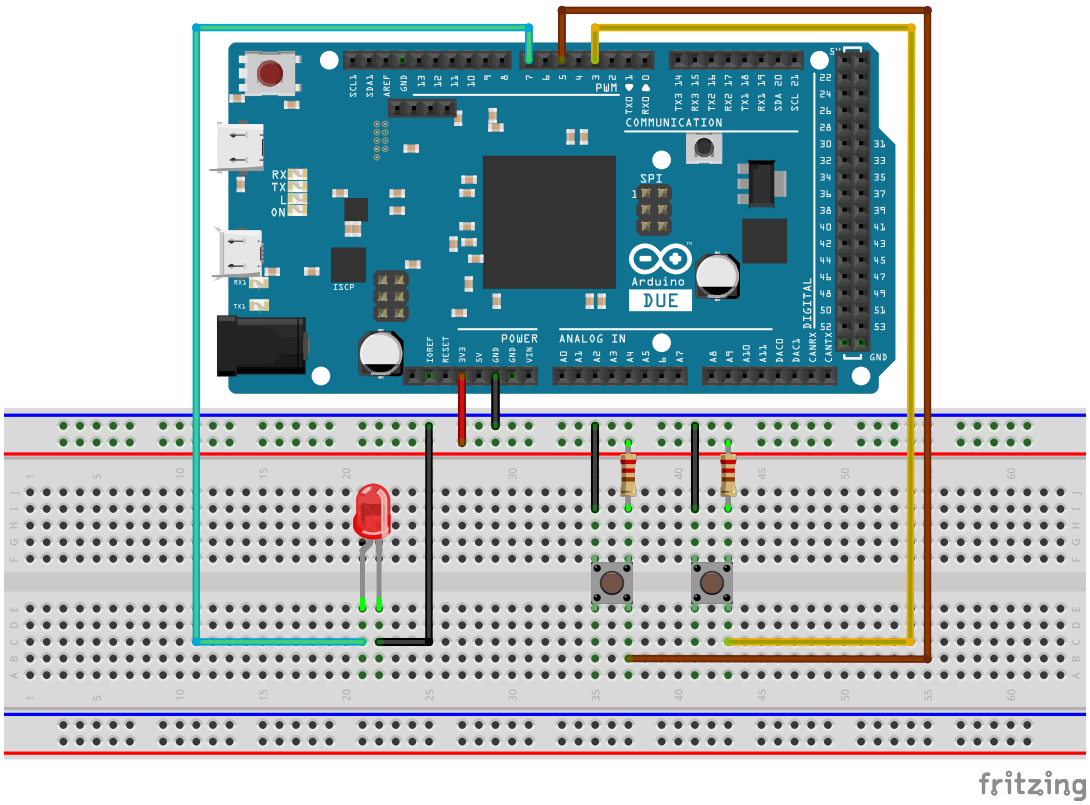

Abbildung 1: Vorschlag für die Verdrahtung des Versuchsaufbaus.

Die Programmstruktur entspricht folgendem Template:

```
void setup()
{
  // Anweisungen für die initiale Konfiguration des Mikrocontrollers
}

void loop()
{
  // Anweisungen, die der Mikrocontroller 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

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 ist es notwendig den Taster-Eingang zusätzlich mit einem **pull-up bzw. pull-down Widerstand** (rn-wissen.de/index.php/Pullup_Pulldown_Widerstand) auszustatten (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 Taster benötigt. Eine Erläuterung des dazu notwendigen Vorgehens finden Sie unter: arduino.cc/en/Tutorial/DigitalPins.

Aufgabe 1.3

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

Entwerfen Sie ein Programm, welches die Betätigung des Tasters als **Interrupt** innerhalb einer entsprechenden **Interruptroutine** behandelt. Lesen Sie dazu den Inhalt der Arduino Referenz unter arduino.cc/en/Reference/AttachInterrupt, um zu erfahren welche Funktionalität Arduino zur Interruptbehandlung anbietet. 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

Ziel dieser Aufgabe ist es die LED mit einem **pulsweitenmodulierten Signal** variabel zu regeln. Eine kurze Erläuterung des Begriffes "**Pulsweitenmodulation**" finden Sie unter arduino.cc/en/Tutorial/PWM.

Das pulsweitenmodulierte Signal wird mit Hilfe eines im Mikrocontroller integrierten 8-bit Timers erzeugt. Erstellen Sie ein Programm, das die Helligkeit der LED mit einer Sägezahnfunktion ansteuert - bei einem 8-bit Timer müssen Sie also für Ihre Kontrollvariable eine Rechnung Modulo 256 durchführen. Verwenden Sie für Ihr Programm folgende Arduino Funktion:

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

Hinweis: Vergessen Sie bei Ihrem Programm nicht die `delay` Anweisung in der Schleife. Verwenden Sie zusätzlich die erste serielle Schnittstelle des Mikrocontrollers, um die Ausgabe des gegewärtigen Wertes der Kontrollvariable zu implementieren. Wie im folgenden Template skizziert, werden für die serielle Kommunikation Methoden der `Serial` Klasse verwendet. Machen Sie sich mit dem angebotenen Funktionsumfang vertraut (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

Aufbauend auf den bisherigen Aufgaben, geht es bei dieser Aufgabe darum die Einstellung der LED-Helligkeit mit zwei Tastern zu ermöglichen. Dabei soll jeweils ein Taster für das inkrementieren bzw. dekrementieren der Kontrollvariablen zuständig sein. Enwerfen Sie zwei Versionen einen Programms, die jeweils folgendes Verhalten aufweisen:

(a) Abfragen (**Polling**) der Taster in der Hauptschleife und die damit einhergehende Manipulation der Kontrollvariablen.

(b) Behandlung der jeweiligen Taster in zwei entsprechenden Interruptroutinen, die den Wert der Kontrollvariablen beeinflussen.