

## Aufgabenblatt 01 Termine: KW 15

Gruppe	
Name(n)	Matrikelnummer(n)

### 1 Ein-/Ausgabe digitaler Signale

Bereits auf Aufgabenblatt 0 wurde für die Ausgabe von Signalen bei dem ersten, beispielhaften Aufbau zur Demonstration eines Arduino-Sketches (siehe Blatt 0, Abb. 2 und 1) unkommentiert eine Leuchtdiode mit Vorwiderstand verwendet. Dieses Aufgabenblatt befasst sich nun mit elementaren elektrotechnischen Grundlagen, beispielsweise der Berechnung des o.g. Vorwiderstandes oder der Grundlagen, die beim Einlesen bzw. bei der Ausgabe digitaler Signale berücksichtigt werden müssen. Der Schwerpunkt liegt allerdings auf Strategien der Beobachtung von Eingangssignalen.

#### Aufgabe 1.0 Test des Blink-Sketches aus Blatt 0

Erstellen Sie in Anlehnung an Abb. 1 (wobei der Taster mit Pullup-Widerstand hier noch nicht benötigt wird), den Aufbau zum Test des Blink-Sketches aus Aufgabenblatt 0, Aufgabe 0.2. Nutzen Sie zum Upload ihres Programmes auf den DUE den Programming-Port des Boards.

#### 1.1 Etwas Elektrotechnik

Für die Lösung der folgenden Aufgaben könnte ein Versuchsaufbau, wie in Abbildung 1 exemplarisch mit einem Arduino-DUE Board dargestellt, aussehen. Mit diesem Aufbau lassen sich die folgenden Teilaufgaben dieses Blattes ohne Änderungen der Verdrahtung umsetzen. Um beispielsweise die angeschlossene Leuchtdiode blinken zu lassen, können Sie den Test-Sketch aus der Aufgabe 0 (siehe dort Listing 1) verwenden. Die Aufgaben 1.1 und 1.2 behandeln die Dimensionierung der in dem Aufbau verwendeten Widerstände.

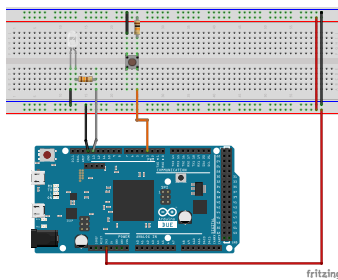


Abbildung 1: Vorschlag für die Verdrahtung des Versuchsaufbaus.

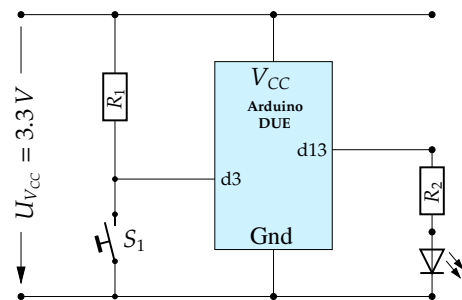


Abbildung 2: Dazugehöriger Schaltplan

Hinweis zur Verschaltung von Steckbrett (Abb. 3) und Taster (Abb. 4):

Die Steckkontakte zwischen der horizontalen roten und blauen Linie am unteren bzw. am oberen Rand des Steckbrettes sind jeweils horizontal durchkontaktiert und die sich daraus ergebenden beiden Zeilen pro Bereich stellen quasi „Stromschienen“ dar. Die übrigen Kontakte des Verdrahtungsfeldes sind spaltenweise jeweils in 5er-Gruppen verbunden.

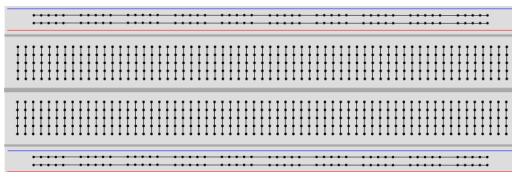


Abbildung 3: Verschaltung des Steckbrettes

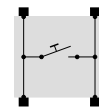


Abbildung 4: Verschaltung des Tasters (gleiche Orientierung wie im Verdrahtungsvorschlag)

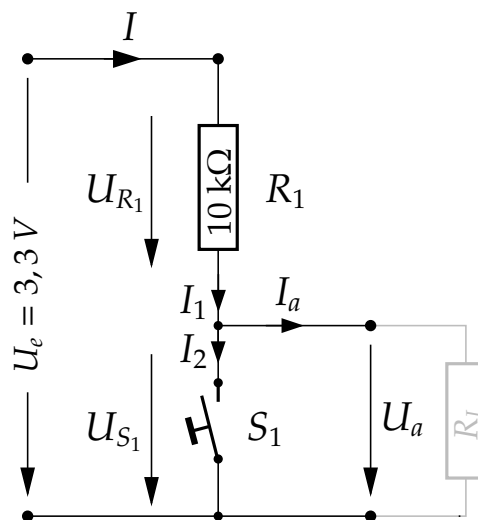


Abbildung 5: Taster mit Pullup-Widerstand

**Aufgabe 1.1** Schaltung mit Pullup-Widerstand

Vergegenwärtigen Sie sich die Schaltung nach Abb. 5 und stellen Sie den Bezug zu Abb. 1 her.

- (a) Welche Aufgabe hat der Widerstand  $R_1$ ?
- (b) Welche Spannungswerte erwarten Sie am Ausgang der Schaltung ( $U_a$ ):  
 Taster  $S_1$  gedrückt:  $U_a =$   
 Taster  $S_1$  Ruhestellung:  $U_a =$
- (c) Wie hoch ist der Querstrom  $I$  ( $I = I_1 = I_2$ ; mit  $I_a \rightarrow 0$ )  
 Taster  $S_1$  gedrückt:  $I =$   
 Taster  $S_1$  Ruhestellung:  $I =$

Die Werte für die Eingangsspannung  $U_e$  und für den Widerstand  $R_1$  entnehmen sie bitte der Skizze in Abb. 5. Weiterhin soll angenommen werden, dass der Ausgangsstrom  $I_a$  vernachlässigbar klein ist ( $I_a \rightarrow 0$ ). Weitere **Beschreibungen** finden Sie auch bei Aufgabe 1.3.

**Aufgabe 1.2** Schaltung mit Vorwiderstand

In Ihrer nächsten Aufgabe werden Sie mit dem Mikrocontroller die auf dem Steckbrett platzierte LED (Light-Emitting Diode) ansteuern (siehe Abb. 1). Hierbei handelt es sich um ein Bauelement, das im Gegensatz zu einer Glühbirne keine annähernd lineare I-U-Kennlinie (graphische Darstellung der Strom-Spannungs-Paare) aufweist. Im Falle der Diode lässt sich der Zusammenhang von Strom und Spannung durch eine exponentielle Funktion approximieren.

Abb. 6 zeigt die Kennlinie der im Versuch verwendeten LED. Ein linearer Anstieg der Spannung

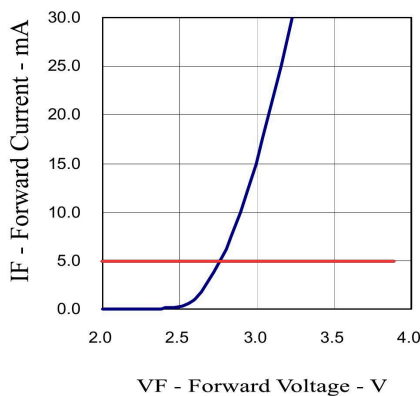


Abbildung 6: Kennlinie der LED (Datenblatt: **L200-SIW-20D\_LL**)

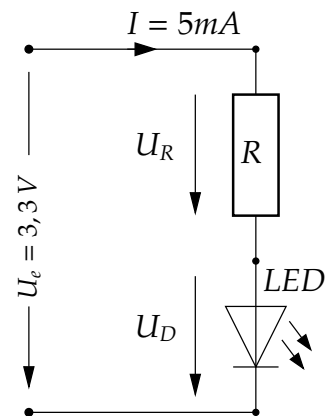


Abbildung 7: LED mit Vorwiderstand

über der LED hat also einen exponentiellen Anstieg des Stromes durch die Diode zu Folge. Die Ausgänge z. B. des Arduino DUE liefern Spannungswerte bis zu 3.3V, was also einen Stromfluss durch die Diode (siehe Abb. 6) von  $> 30\text{ mA}$  zur Folge hätte; dabei möchten wir den Strom zum

Schutz der Ausgangstreiber des Arduino und zum Schutz der Diode selbst auf 5 mA begrenzen. Eine Lösung wäre auch hier die Verwendung eines Vorwiderstandes, wie in Abb. 7 gezeigt. Berechnen Sie den erforderlichen Lastwiderstand  $R$ . Die Durchlassspannung  $U_D$  der Diode im geforderten Arbeitspunkt entnehmen sie bitte der Abb. 6

Hinweis: Der in Abbildung 1 gezeigte Vorwiderstand von  $130\ \Omega$  ist nicht die Umsetzung der exakten Berechnung, sondern lediglich der Tatsache geschuldet, dass dieser gerade in ausreichender Zahl vorhanden war. Weshalb ist dieses laxer Vorgehen im Falle des Widerstandes möglich?

## 1.2 Beobachtung von Eingangssignalen

Die folgenden Aufgaben behandeln verschiedene Strategien zur Beobachtung/Überwachung von Eingangssignalen.

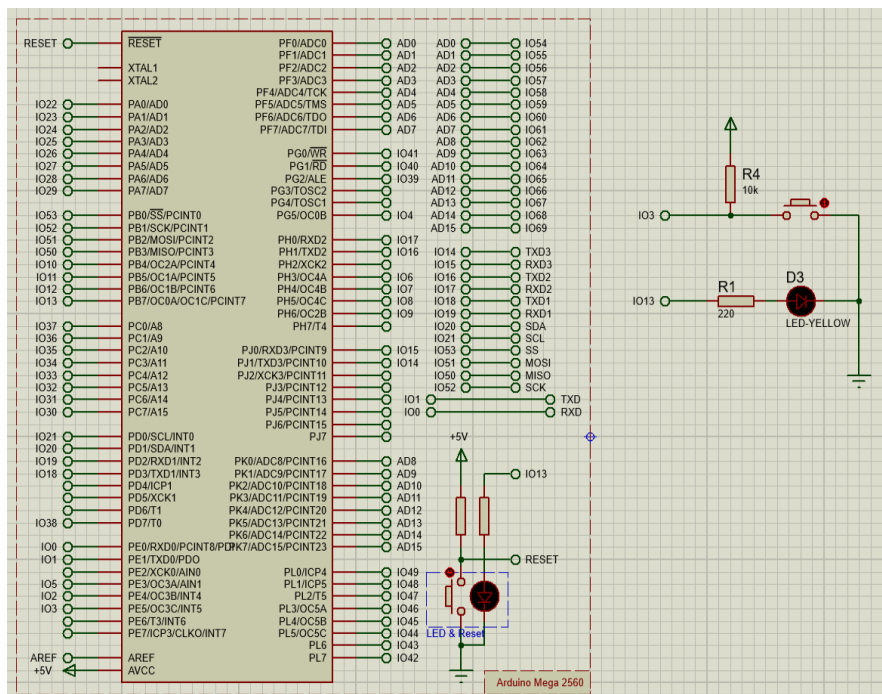


Abbildung 8: Proteus Firmware-Projekt: Aufbau zum Ein- und Ausschalten einer LED mittels Taster

### Aufgabe 1.3 Verarbeitung von Eingaben mittels Polling

Erweitern Sie das Beispielprogramm aus Blatt 0 (siehe Listing 1) um eine periodische Abfrage des Logikpegels am Anschlusspin des angeschlossenen Tasters (siehe Abb. 1 bzw. 8). Wird der Taster betätigt, so soll die externe LED ein- bzw. ausgeschaltet werden (*state toggle*). Beachten Sie: **Eine** Betätigung des Tasters soll auch **nur einen** Zustandswechsel hervorrufen - die Dauer der Betätigung soll keinen Einfluss haben.

Bedenken Sie bitte auch, dass an den Inputpins zum Auslesen der Taster jederzeit ein Spannungspegel anliegen muss, der eindeutig einem der Logikpegel **LOW** oder **HIGH** zugeordnet

werden kann. Beim Arduino DUE z.B. werden Eingangsspannungen  $\leq 0.3V_{CC}$  sicher **LOW** und Eingangsspannungen  $\geq 0.7V_{CC}$  sicher **HIGH** zugeordnet (vgl. [SAM3X\\_Datasheet, p.1379](#)). Diese Grenzwerte variieren allerdings über die Prozessorfamilien.

Die im Verdrahtungsvorschlag abgebildeten Taster schließen bei Betätigung gegenüber dem Masse-Potenzial (GND, 0V) kurz; es wird also sicher ein **LOW** detektiert werden. Allerdings muss sichergestellt werden, dass im nicht betätigten Zustand des Tasters am Anschlusspin eine Spannung  $\geq 0.7V_{CC}$  anliegt, damit sicher **HIGH** detektiert wird. Dieses kann mit einem **Pull-up Widerstand** umgesetzt werden. Angeschlossen an die Versorgungsspannung ( $V_{CC} = 3,3V$ ) und unter der Annahme, dass der Ausgangsstrom  $\rightarrow 0$  geht, dürfte im nicht betätigten Zustand des Tasters eine Ausgangsspannung von  $\rightarrow V_{CC}$  anliegen und somit sicher **HIGH** detektiert werden. Wird der Taster betätigt, so erfolgt eine Verbindung gegenüber dem Masse-Potenzial und der durch den **Pull-up Widerstand** begrenzte Strom fließt gegen Masse ab. Am Anschlusspin des Tasters liegt in diesem Fall eine Spannung  $\rightarrow 0$  an und es wird sicher ein **LOW** detektiert (siehe auch Aufgabe 1.1). Weitere Information zu dem Thema können Sie auch unter [rn-wissen.de/index.php/Pullup\\_Pulldown\\_Widerstand](http://rn-wissen.de/index.php/Pullup_Pulldown_Widerstand) erhalten.

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

```
* digitalRead(<pin>) → digitalRead
```

Alternativ 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: [www.arduino.cc/en/Tutorial/DigitalPins](http://www.arduino.cc/en/Tutorial/DigitalPins).

#### Aufgabe 1.4 Hardware Interrupt auslösende Digitalpins

Die für die Aufgabe 1.3 entwickelte Lösung, bei der der Taster ständig, wiederkehrend mit `digitalRead` abgefragt wird (Polling), hat einen grundlegenden Nachteil. Erläutern Sie diesen?

Hardware Interrupts sind Interrupts, die als Reaktion auf ein externes Ereignis ausgelöst werden. So besitzen die Prozessoren der Arduino-Boards ausgewählte Inputpins, die in der Lage sind, auf einen Wechsel bzw. auf einen bestimmten Zustand des anliegenden Signals (LOW, CHANGE, RISING, FALLING, HIGH) zu reagieren und einen Interrupt auszulösen.

Ein Hardware-Interrupt ist ein (in der Regel) asynchrones Ereignis, das die Ausführung des Programmcodes unterbricht und zu einer dem Interrupt zugewiesenen Interruptroutine springt. Nach Ausführung der Interruptroutine erfolgt ein Rücksprung an die Stelle im Programmcode, an der die Unterbrechung ausgelöst wurde.

Lesen Sie die Arduino Referenz unter <https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt>, um zu erfahren, welche Inputpins Ihres Arduino-Boards Interruptfähig sind und welche Funktionalität das Arduino Framework zur Interruptbehandlung anbietet.

Entwerfen Sie eine abgewandelte Variante des bisherigen Programms, in der Sie die Betätigung des Tasters als **Hardware-Interrupt** innerhalb einer entsprechenden **Interruptroutine** behandeln.

Für Ihr Programm werden Sie folgende Funktionen benötigen:

```
* attachInterrupt(digitalPinToInterrupt(<pin>), <function>, <mode>) → attachInterrupt
* detachInterrupt(<pin>) → detachInterrupt
```

Welche Aufgabe hat die Funktion *digitalPinToInterrupt(pin)*, als erster Parameter von *attachInterrupt()*? Bei welchen Prozessoren kann die Funktion entfallen?

**Aufgabe 1.5** Praxistest der Lösungen aus Aufgabe 1.3 und 1.4

Testen Sie die Lösungen aus Aufgabe 1.3 und 1.4 am realen Aufbau (siehe Abb. 1). Entdecken Sie Unterschiede im Verhalten? Welcher Lösung würden Sie den Vorzug geben?