Aufgabenblatt 01 Termine: KW 15

Gruppe	
Name(n)	Matrikelnummer(n)

1 Ein-/Ausgabe digitaler Signale

Diese Aufgabe befasst sich mit elementaren elektrotechnischen Grundlagen, die beim Einlesen bzw. bei der Ausgabe digitaler Signale berücksichtigt werden müssen. Der Schwerpunkt liegt allerdings auf Strategien der Beobachtung von Eingangssignalen.

1.1 Etwas Elektrotechnik

Für die Lösung der folgenden Aufgaben könnte ein Versuchsaufbau, wie in Abbildung 1 exemplarisch mit einem Arduino-MEGA 2560 Board dargestellt, aussehen. Mit dem Aufbau lassen sich alle folgenden Teilaufgaben ohne Änderungen der Verdrahtung umsetzen. Um beispielsweise die angeschlossene Leuchtdiode blinken zu lassen, können Sie den Test-Sketch aus der einführenden Ausgabe 0 (siehe dort Listing 1) verwenden. Die beiden folgenden Aufgaben befassen sich mit der Dimensionierung der in dem Aufbau verwendeten Widerstände.

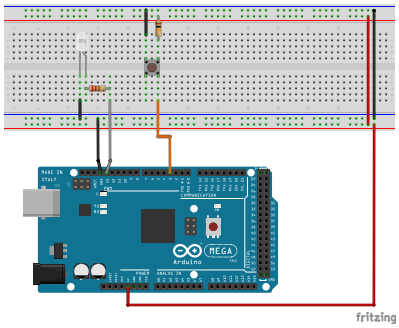

Abbildung 1: Vorschlag für die Verdrahtung des Versuchsaufbaus.

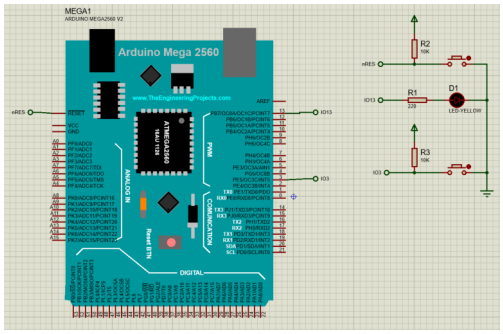

Abbildung 2: Äquivalenter Aufbau: Proteus VSM Scematic

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.

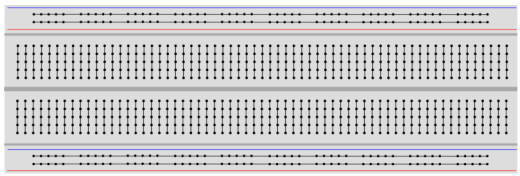

Abbildung 3: Verschaltung des Steckbrettes

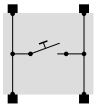

Abbildung 4: Verschaltung des Tasters (gleiche Orientierung wie im Verdrahtungsvorschlag)

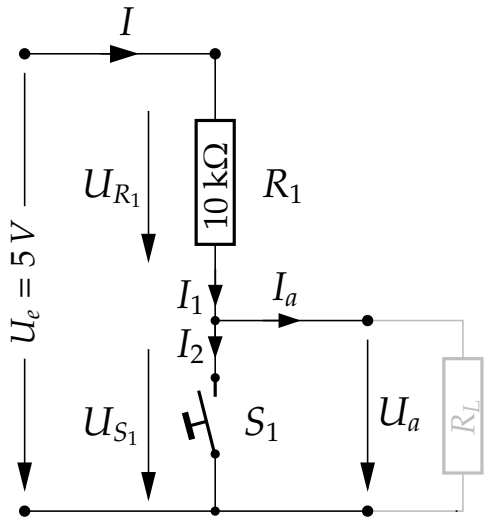

Abbildung 5: Taster mit Pullup-Widerstand

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 $I_a \to 0$)
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 ($I_a \to 0$). 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 über der LED hat also einen exponentiellen Anstieg des Stromes durch die Diode zu Folge. Die Ausgänge z. B. des Arduino MEGA 2560 liefern Spannungswerte bis zu 5V, was also einen Stromfluss durch die Diode (siehe Abb. 6) von mehreren Ampere zur Folge hätte; dabei möchten wir den Strom zum Schutz der Ausgangstreiber des Arduino und zum Schutz der Diode selbst auf 10 mA begrenzen. Eine Lösung wäre auch hier die Verwendung eines Vorwiderstandes, wie in Abb. 7 gezeigt.

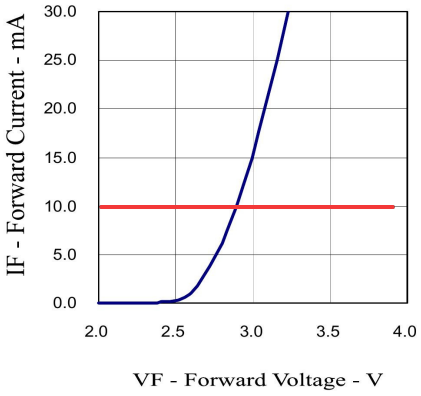

Abbildung 6: Kennlinie der LED (Datenblatt: L200-SIW-20D_LL)

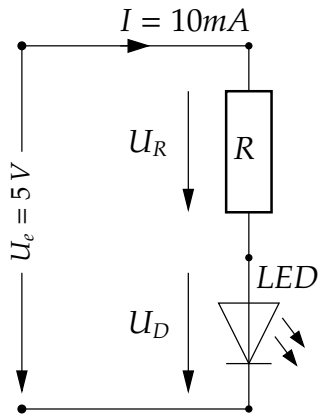

Abbildung 7: LED mit Vorwiderstand

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 220 Ω ist nicht die Umsetzung der exakten Berechnung, sondern lediglich der Tatsache geschuldet, dass dieser gerade in ausreichender Zahl vorhanden war. Weshalb ist dieses laxe Vorgehen im Falle des Widerstandes möglich?

1.2 Beobachtung von Eingangssignalen

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

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 eines der angeschlossenen Taster. 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 MEGA 2560 z.B. werden Eingangsspannungen $\leq 0.3V_{CC}$ sicher **LOW** und Eingangsspannungen $\geq 0.6V_{CC}$ sicher **HIGH** zugeordnet (vgl. ATmega2560_Datasheet, p.355). Diese Grenzwerte variieren allerdings über die Prozessorfamilien.
Die im Verdrahtungsvorschlag abgebildeten Taster schließen bei Betätigung gegenüber dem Masse-Potenzial (`GND`, 0 V) kurz; es wird also sicher ein **LOW** detektiert werden. Allerdings muss sichergestellt werden, dass im nicht betätigten Zustand des Tasters am Anschlusspin eine Spannung $\geq 0.6V_{CC}$ anliegt, damit sicher **HIGH** detektiert wird. Dieses kann mit einem **Pull-up Widerstand** umgesetzt werden. Angeschlossen an die Versorgungsspannung (`VCC` = 5V) und unter der Annahme, dass der Ausgangsstrom $\rightarrow 0$ geht, dürfte im nicht betätigten Zustand des Tasters eine Ausgangsspannung von $\rightarrow$`VCC` 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` 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`.

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 bestimmte Inputpins, die in der Lage sind, auf einen Wechsel bzw. eines bestimmten Zustandes des anliegenden Signals (LOW, CHANGE, RISING, FALLING, HIGH) 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 den Inhalt der 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 Funktonen 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?