64-211 Übung Eingebettete Systeme

Aufgabenblatt 1 Termine: 03.04./06.04.

Gruppe	
Name(n)	Matrikelnummer(n)

Grundlagen

Im Rahmen der Übungen zur Vorlesung „Eingebettete Systeme“ werden Sie Aufgaben mit dem **Arduino Due** Board lösen. Im Gegensatz zur weit verbreiteten **8-bit AVR** Architektur, kommt beim Arduino Due Board ein **32-bit ARM** basierter Mikrocontroller (Atmel SAM3X8E ARM Cortex-M3) zum Einsatz.

Arduino Due Entwicklungsboard `store.arduino.cc/arduino-due`

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 Online unter: Arduino Due Pinbelegung.

ACHTUNG: die maximale, an den I/O-Pins anliegende, Spannung darf unter keinen Umständen 3,3 V überschreiten! Eine höhere Spannung kann zur permanenten Zerstörung des Mikrocontrollers führen. Sollte ein eindeutiger Hinweis bezüglich der Spannung für eine der zu verwendenden Komponenten in der Aufgabenstellung fehlen (nobody is perfect), schliessen Sie diese bitte an 3,3 V an!

Arduino Entwicklungsumgebung `www.arduino.cc/en/Main/Software`

Zur Programmierung wird die Arduino IDE verwendet, die auf den Rechnern des Arbeitsbereiches TAMS über ein Terminal mit dem Befehl `$tamsSW/arduino-1.8.2/arduino` gestartet werden kann.

Online Hilfe `www.arduino.cc/reference/en`

Die Online-Referenz des Arduino Frameworks ist nicht immer perfekt, stellt jedoch eine wertvolle Quelle für grundlegende Information zur Syntax und Semantik enthaltener Funktionen dar. Vereinzelt sind Code-Beispiele enthalten, welche die Verwendung der jeweiligen Funktion darstellen. **Empfehlung:** Sollte eine der Funktionen des Arduino Frameworks unverständlich erscheinen, schlagen Sie diese zuerst in der Online-Referenz nach!

Genereller Hinweis: Speichern Sie die Lösung zu jeder Aufgabenstellung gesondert ab. Dieses erleichtert einerseits die Kontrolle Ihrer Lösung, andererseits hilft es Ihnen bei der Bearbeitung von Aufgaben die teilweise Aufgabenblatt-übergreifend aufeinander aufbauen. **Kommentare im Code sind ausdrücklich erwünscht!**

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.

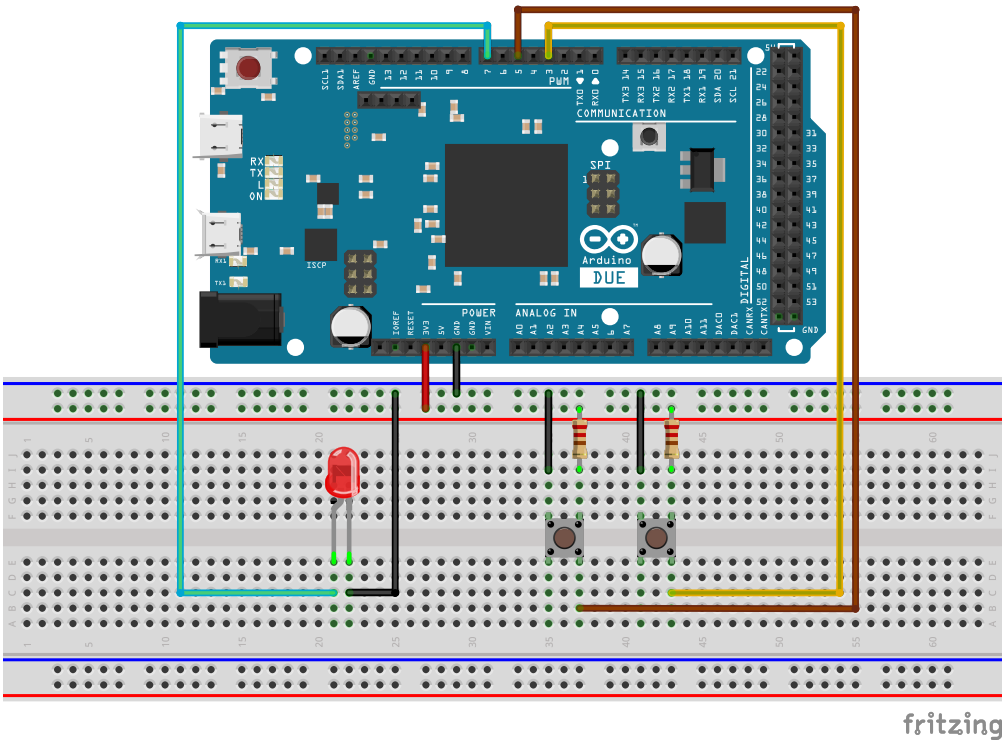

Abbildung 1: Vorschlag für die Verdrahtung des Versuchsaufbaus.

Aufgabe 1.1

Vergegenwärtigen Sie sich die Schaltung in Abb. 2 und stellen Sie den Querbezug zu Abb. 1 her.

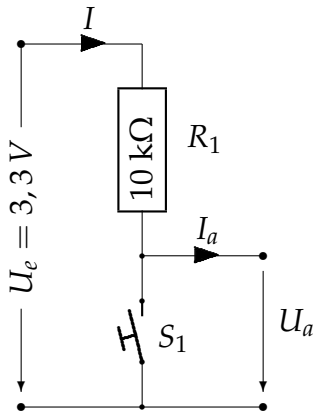

Abbildung 2: Taster mit Pullup-Widerstand

(a) Welche Aufgabe hat der Widerstand R_1?

(b) Welche Spannungswerte erwarten Sie am Ausgang der Schaltung bei
Taster S_1 gedrückt: $U_a =$
Taster S_1 Ruhestellung: $U_a =$

(c) Wie hoch ist der Querstrom I bei
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. 2. Weiterhin soll angemommen werden, dass der Ausgangsstrom I_a vernachlässigbar klein ist.

Aufgabe 1.2

Ihre nächste Aufgabe ist es, ein Programm für den Mikrocontroller zu erstellen, das die auf dem Steckbrett platzierte LED nach folgendem Schema ansteuert:

1. LED an, Dauer: 500 Millisekunden.
2. LED aus, Dauer: 2 Sekunden.

– Fahren Sie mit 1. fort.

Die grundlegende Programmstruktur baut auf folgendem Template auf:

```
// Deklaration/Definition von Variablen mit globaler Sichtbarkeit

// Beispiel: Variable, die den Logikpegel eines digitalen I/O
//           Anschlusspins speichert
int pin_state = LOW;

void setup()
{
  // Anweisungen für die initiale Konfiguration des Mikrocontrollers
  // Diese Funktion wird während des Startvorgangs einmalig aufgerufen

  // Beispiel: Konfiguration der Richtung des digitalen I/O Anschlusspins
  // mit der Nummer 10 (hier: als Ausgangspin)
  pinMode(10, OUTPUT);
}

void loop()
{
  // Anweisungen, die der Mikrocontroller innerhalb einer Iteration der
  // Hauptschleife ausführen soll

  // Beispiel: Wechsel des Logikpegels am Ausgangspin 10 im Sekunden-Takt
  digitalWrite(10, pin_state);
  pin_state = !pin_state;
  delay(1000);
}
```

Folgende Funktionen sind bei der Bearbeitung der Aufgabe hilfreich:

* `pinMode(`⟨*pin*⟩`,` ⟨*mode*⟩`)` → `pinMode`
* `digitalWrite(`⟨*pin*⟩`,` ⟨*mode*⟩`)` → `digitalWrite`
* `delay(`⟨*ms*⟩`)` → `delay`

Aufgabe 1.3

Erweitern Sie die Lösung der ersten Aufgabe 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, dass am von Ihnen für den Taster gewählten Anschlusspin zu jeder Zeit ein definierter Logikpegel vorhanden sein muss, d.h. entweder **LOW** oder **HIGH**. Die im Verdratungsvorschlag abgebildeten Taster schliessen bei Betätigung gegenüber dem Masse-Potenzial (`GND`, 0 V) kurz. Es muss also sichergestellt werden, dass im nicht betätigten Zustand des Tasters am Anschlusspin der Logikpegel **HIGH** anliegt. Dieses kann mit einem **Pull-up Widerstand** umgesetzt werden. Angeschlossen an die Versorgungsspannung (`VCC`, 3,3 V) fließt im nicht betätigten Zustand des Tasters ein dem Widerstandswert ensprechender Strom und am gewählten Anschlusspin liegt der benötigte Logikpegel **HIGH** an. Wird der Taster betätigt so erfolgt eine Verbindung gegenüber dem Masse-Potenzial und der durch den **Pull-up Widerstand** produzierte Strom fließt dorthin ab. Am Anschlusspin des Tasters liegt in diesem Fall der Logikpegel **LOW** an. Weitere Information zu dem Thema können Sie 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`. Gestalten Sie das Programm entsprechend um.

Aufgabe 1.4

Die in der Aufgabe 1.2 entstandene Lösung, bei der der Taster mit `digitalRead` abgefragt wird, hat einen grundlegenden Nachteil. Erläutern Sie diesen?

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

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 Funktionalität das Arduino Framework zur Interruptbehandlung anbietet. Benutzen Sie in Ihrem Programm folgende Funktionen:

* `attachInterrupt(`⟨*pin*⟩`,` ⟨*function*⟩`,` ⟨*mode*⟩`)` → `attachInterrupt`
* `detachInterrupt(`⟨*pin*⟩`)` → `detachInterrupt`