64-211 Übung Eingebettete Systeme

Aufgabenblatt 1 Termine: 10.04. / 11.04.

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 1.1 (Punkte 15)

Machen Sie sich mit der Belegung der Anschlüsse des Teensy 2.0 Boards vertraut (http://www.pjrc.com/teensy/pinout.html). Ihre Aufgabe ist es ein Programm für den AVR-Microcontroller unter Verwendung der Arduino IDE zu erstellen, das die auf dem Board integrierte 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.

Die Programmstruktur entspricht folgendem Template:

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

void loop()
{
  // Anweisungen, die der AVR-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 auf dem Board integrierte LED ein- und auszuschalten.

Wählen Sie einen passenden Anschluss auf dem Teensy 2.0 Board aus. Wichtig ist dabei, dass der Anschluss 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 ausstatten.

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 Schaltung mit einer Interruptsteuerung. Informieren Sie sich dazu im Internet, wie die Arduino-Interruptbehandlung funktioniert. Benutzen Sie:

* `attachInterrupt(⟨intr⟩, ⟨function⟩, ⟨mode⟩)` arduino.cc/en/Reference/attachInterrupt
* `detachInterrupt(⟨intr⟩)` 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 nicht die `delay` Anweisung in der Schleife. In einer Konsole (serieller Anschluss, der über USB emuliert wird) soll der Wert der „Helligkeitsvariablen“ ausgegeben werden. Wie in dem Template skizziert, werden dazu die `Serial.`-Funktionen benutzt, siehe arduino.cc/en/Reference/Serial.

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

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

Aufgabe 1.5 (Punkte 15+15)

Wie in den Aufgabenteilen 2 und 3 soll die Einstellung der LED-Helligkeit mit zwei Tastern geregelt werden, die den PWM-Wert inkrementieren, bzw. dekrementieren.

(a) Fragen Sie die Taster direkt in der Schleife ab.

(b) Implementieren Sie zwei Interruptroutinen, die die Helligkeit regeln.