Prof. Dr.-Ing. Ingo Elsen,
B.Sc. Hanna Babilon, M.Sc. Tim Höner, M.Eng. Marcel Stüttgen

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

ARBKVS, Modulnr.: 53107

Praktikum

Liebe Studierende,

dieses Dokument enthält alle Praktikumsaufgaben. Es gibt während des Semesters Meilensteine, zu denen bestimmte Aufgaben erledigt sein müssen. Ist die Aufgabe zu dem Meilensteintermin nicht fertig, kann ich leider kein Gesamttestat erteilen! Ihre schriftliche Ausarbeitung zu den Versuchen protokollieren Sie bitte auf den beigefügten Bögen am Ende dieses Dokumentes.

Viel Erfolg und freundliche Grüße
Ingo Elsen

Teil 1
ARBKVS Testate

Name:	Vorname:	Matr.-Nr.:

Tabelle 1: Meilensteine im Praktikum

Aufgabe	Meilenstein[1]	Datum	Teiltestat
Aufgabe 1	A, C, E, G: 2019.11.08 B, D, F, H: 2019.10.31		
Aufgabe 2	A, C, E, G: 2019.11.29 B, D, F, H: 2019.11.22		
Aufgabe 3	A, C, E, G: 2019.12.20 B, D, F, H: 2019.12.13		
Aufgabe 4	A, C, E, G: 2020.01.17 B, D, F, H: 2020.01.10		
Aufgabe 5	A, C, E, G: 2020.01.17 B, D, F, H: 2020.01.10		
Gesamttestat			

Hinweise zu den Testaten:

1. Der Testatbogen ist der einzige Nachweis Ihrer Teilnahme am Praktikum und ist sorgfältig aufzubewahren. Es ist empfehlenswert diesen Testatbogen auch nach erfolgreichem Abschluss des Praktikums aufzubewahren. Er kann, z.B. bei Verlust des Praktikumspasses, zur Neuausstellung als Nachweis herangezogen werden.

2. Der Versuch ist von jedem bzw. jeder Gruppe unter Beteiligung aller Praktikumsteilnehmer aufzubauen.

3. Zur erfolgreichen Teilnahme an einem Praktikumsversuch gehören die Vorbereitung zum Versuch, praktische Durchführung während des Praktikumstermins, so-

[1] je nach Gruppe. Meilenstein bedeutet, dass zu diesem Datum der Versuch testiert sein muss.

wie das nachzuweisende Verständnis für die durchgeführten Versuche. Anderenfalls kann eine aktive Mitarbeit bei der Versuchsdurchführung nicht bescheinigt und damit kein Testat erteilt werden. Die erfolgreiche Teilnahme wird am Ende des Versuchstermins durch den Betreuer testiert.

4. Zur Anerkennung des Praktikums und zur Erteilung des Gesamttestates im Praktikumspass (bitte zum letzten Termin mitbringen) müssen alle Versuche testiert sein.

5. Das Praktikum muss in dem laufenden Semester abgeschlossen werden. Eine Verteilung einzelner Versuche über mehrere Semester ist nicht möglich.

6. Zur sinnvollen Durchführung des Praktikums ist eine Vorbereitung mit Hilfe der Versuchsanleitung und eventuell weiterer Literatur erforderlich. In ILIAS findet sich bereits Material zum Praktikum.

Teil 2
Sicherheitshinweise

1. In den Praktikumsräumen herrschen Laborbedingungen. Es wird dazu besonders vor Berührung spannungsführender Teile gewarnt.

2. Im Gefahrenfall sofort den NOT-AUS-Schalter betätigen und den Betreuer umgehend verständigen. Erst nach Freigabe durch den Betreuer darf die Spannung wieder eingeschaltet werden.

3. Die Geräte und Versuchsanordnungen bitte sorgfältig behandeln.

4. Beschädigte Geräte bitte dem Betreuer melden, damit eine umgehende Reparatur veranlasst werden kann.

Teil 3
Aufgaben

Aufgabe 1 Lauflicht in Assembler

Steuern Sie die drei LEDs auf dem Microcontrollerboard so an, dass sich ein Lauflicht ergibt. Implementieren Sie das Lauflicht so, dass es am Ende seine Bewegungsrichtung ändert. Als Takt soll zunächst ca. 1/5 Sekunde Leuchtdauer für jede LED genommen werden.

a) Ermitteln Sie über den Schaltplan des Boards, an welchen Anschlüssen Sie die LEDs anschließen können! Zeigen Sie den Betreuern die Anschlüsse!

b) Verkabeln Sie die LEDs über Steckbrücken, so genannte Jumper-Wires!

c) Legen Sie im Atmel-Studio ein entsprechendes Assembler-Projekt an!

d) Programmieren Sie die Lichtsteuerung und legen die Verzögerungsschleife als Unterprogramm (Aufruf mit RCALL) an. Kommentieren Sie Ihren Assembler-Code so, dass ein Fremder ihn verstehen kann!

e) Starten Sie das Programm im Simulator, setzen Sie einen Breakpoint (Haltepunkt für der Debugger) und zeigen Sie den Betreuern wie das Lauflicht funktioniert!

f) Bringen Sie das Programm auf dem Controller zum Laufen. Übertragen Sie dazu das Programm auf den Controller. Welche Funktion hat der Reset-Pin?

g) Protokollieren Sie Ihre Entwicklungszeiten für jeden Schritt!

h) OPTIONAL: Integrieren Sie auch die blaue LED ("LED-low active")! Hierbei müssen Sie bedenken, dass die LED leuchtet, wenn das entsprechende Portbit = 0 gesetzt ist. Wie müssen Sie in diesem Fall die anderen LEDs setzen?

Ziele:

- Die Entwicklungsumgebung bedienen können.
- Ein Assemblerprogramm schreiben und debuggen können.
- Den Simulator bedienen können.
- Die ATMega Microcontroller-Architektur kennen lernen.
- Ein Programm auf einen Microcontroller übertragen können.

Aufgabe 2 Lauflicht in C/C++

Steuern Sie die drei LEDs auf dem Microcontrollerboard sollen so an, dass sich ein Lauflicht ergibt, das am Ende seine Bewegungsrichtung ändert. Als Takt soll zunächst ca. 1/5 Sekunde Leuchtdauer für jede LED genommen werden.

a) Legen Sie im Atmel-Studio ein entsprechendes C/C++-Projekt an!

b) Programmieren Sie die Lichtsteuerung! Strukturieren Sie Ihr Programm in mehrere Funktionen! Kommentieren Sie Ihren C-Code so, dass ein Fremder ihn verstehen kann! Sie können für die Zeitverzögerung die Funktion _delay_ms() nutzen:

```
1  #include <avr/io.h>
2  #define F_CPU 1000000UL // hier muss die passende Taktfrequenz rein!
3  #include <util/delay.h>
4
5  int main( void )
6  {
7      DDRB |= (1 << PB0);
8
9      while(1) {
10         PORTB ^= (1 << PB0);
11         _delay_ms(1000);
12     }
13     return 0;
14 }
```

Was passiert, wenn Sie eine falsche Taktfrequenz eintragen (`#define FCPU ...`)?

c) Starten Sie das Programm im Simulator, setzen Sie einen Breakpoint (Haltepunkt für der Debugger) und zeigen Sie den Betreuern wie das Lauflicht funktioniert!

d) Bringen Sie das Programm auf dem Controller zum Laufen! Übertragen Sie dazu das Programm auf den Controller!

e) Protokollieren Sie Ihre Entwicklungszeiten für jeden Schritt!

f) Vergleichen Sie die Programmgrößen aus Aufgabe 1 mit der Programmgröße aus diesem Versuch!

g) Wie verändern sich die Programmgrößen des C-Programmes, wenn Sie verschiedene Optimierungsstufen auswählen? Protokollieren Sie Ihre Ergebnisse in einer Tabelle!

Ziele:

- Die Entwicklungsumgebung bedienen können.
- Ein C/C++-Programm schreiben und debuggen können.
- Den Simulator bedienen können.
- Die ATMega Microcontroller-Architektur kennen lernen.
- Ein Programm auf einen Microcontroller übertragen können.

- Unterschiede zwischen den Entwicklungsansätzen erkennen.
- Einschätzen der zeitlichen Effizienz.
- Einschätzung der Resourcennutzung.

Aufgabe 3 Interrupts — Blinker mit Steuerung über Tasten

In diesem Versuch steht es Ihnen frei, ob Sie mit C oder Assembler programmieren. Hinweis: Sollten Sie in C programmieren müssen sie spezielle ISR (Interrupt Service Routinen Schreiben). Wie das geht finden Sie in der Dokumentation zum AVR GCC.
Bisher haben Sie alle Versuche ohne Nutzung von Interrupts realisieren können. Entwickeln Sie ein Programm, das einen Blinker realisiert. Hierzu sollen die rote und die gelbe LED auf dem Board genutzt werden. Über die beiden Taster können die LEDs gesteuert werden. Taster 1 schaltet die gelbe LED zwischen aus und blinken um, Taster 2 die rote. Dabei soll es nicht möglich sein, dass beide LEDs gleichzeitig blinken. Blinkt also beispielsweise die rote LED und Taster 1 (gelb) wird gedrückt, wird die rote LED ausgeschaltet und die gelbe fängt an zu blinken.

a) Zeichnen Sie einen UML Zustandsautomaten der Funktion und zeigen Sie diesen den Betreuern vor!

b) Ermitteln Sie, an welche Anschlüsse Sie die Taster anschließen können, so dass Sie diese über Interrupts abfragen können!

c) Schließen Sie zwei LEDs und zwei Taster an!

d) Legen Sie ein Projekt entsprechend der von Ihnen gewählten Programmiersprache an und realisieren Sie die Aufgabe über Polling, also die dauernde Abfrage der Taster!

e) Legen Sie ein Projekt entsprechend der von Ihnen gewählten Programmiersprache an und realisieren Sie die Aufgabe unter Nutzung von Interrupts für die Eingabe!

f) (Falls Sie in C programmieren): Wie bildet der Compiler die ISR auf die Assembler Interrupt Vektoren ab?

g) Welche Vor- und Nachteile hat die Implementierung über Interrupts?

Ziele:

- Funktionsweise Interrupts verstehen.
- Interrupts als Mittel für nebenläufige Programmierung nutzen können.
- Verstehen, wie ein Betriebssystem mit Eigenheiten der Hardware umgehen muss.
- Verstehen, wie ein BS Hardware-Funktionalitäten abbildet.

Aufgabe 4 Multitasking — Zeitbasis

Implementierungssprache für diese Aufgabe ist C.

Schreiben Sie ein Programm, dass eine Zeitmessung auf dem AVR realsiert. Gehen Sie dazu folgendermaßen vor:

a) Nutzen Sie einen Timer des AVR um eine Taktrate von einer Millisekunde (oder einer guten Annäherung) zur erzeugen!

b) Mit diesem Takt soll ein monotoner Zähler realisiert werden, der bei Programmstart losläuft und die Millisekunden seit Programmstart zählt.

c) Erstellen Sie zwei Funktionen

- `waitFor(int32_t ms)` Wartet für `ms` millisekunden.
- `waitUntil(int32_t ms)` Wartet bis der Zähler den Wert `ms` erreicht hat.

d) Schreiben Sie ein Testprogramm, in dem Sie eine oder mehrere LEDs unter Nutzung dieser Funktionen blinken lassen!

Beantworten Sie folgende Fragen:

a) Wie lange dauert es, bis der Zähler überläuft?

b) Was könnte man tun, um größere Zeiten zu verarbeiten?

Ziele:

- Einfache Scheduling Algorithmen verstehen und anwenden können.
- Eine komplexe Lösung Schritt für Schritt entwickeln
- Komplexen Code testen können.

Aufgabe 5 Multitasking — Nutzung von Pre-Emptive-Multitasking und geteilten Resourcen

Implementierungssprache für diese Aufgabe ist C++. Implementierungsumgebung ist der PC, Entwicklungsumgebung ist VisualStudio C++!

Schreiben Sie ein Programm das drei Threads hat. Sie können hierfür die Funktionalitäten aus C++11 nutzen, d.h. insbesondere die Containerklassen und Funktionen der Standardbibliothek:

```
1 #include <iostream>
2 #include <thread>
3 #include <mutex>
4
5 // ...
```

T1 Thread T1 schreibt alle Kleinbuchstaben des Alphabets auf `cout`!

T2 Thread T2 schreibt alle natürlichen Zahlen von $0 \ldots 32$ auf `cout`!

T3 Thread T3 schreibt alle Großbuchstaben des Alphabets auf `cout`!

a) Entwickeln Sie ein Programm, dass die obige Ausgabe so realisiert, dass alle drei Threads unsynchronisert laufen! Wie sieht die Ausgabe aus?

b) Synchronisieren Sie die Ausgabe mittels Mutex!

c) Schreiben Sie eine Klasse `Semaphore` die ein Semaphor realisiert! Wie müssen Sie das Semaphor initialisieren, damit sie es statt des Mutexes nutzen können? Demonstrieren Sie die Lösung im Programm!

Ziele:

- Einfache Scheduling Algorithmen verstehen und anwenden können.
- Die Notwendigkeit von Thread-Synchronisierung bei Nutzung geteilter Betriebsmittel verstehen!
- Synchronisationsprimitive verstehen und anwenden können.
- Eine komplexe Lösung Schritt für Schritt entwickeln.
- Komplexen Code testen können.

Name:	Vorname:	Matr.-Nr.:

Name:	Vorname:	Matr.-Nr.:

Name:	Vorname:	Matr.-Nr.:

Name:	Vorname:	Matr.-Nr.:

Name:	Vorname:	Matr.-Nr.:

Name:	Vorname:	Matr.-Nr.: