Exercises in System Level Programming (SLP) – Summer Term 2025

Exercise 1

Maxim Ritter von Onciul
Eva Dengler

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Faculty of Engineering

Organizational Matters

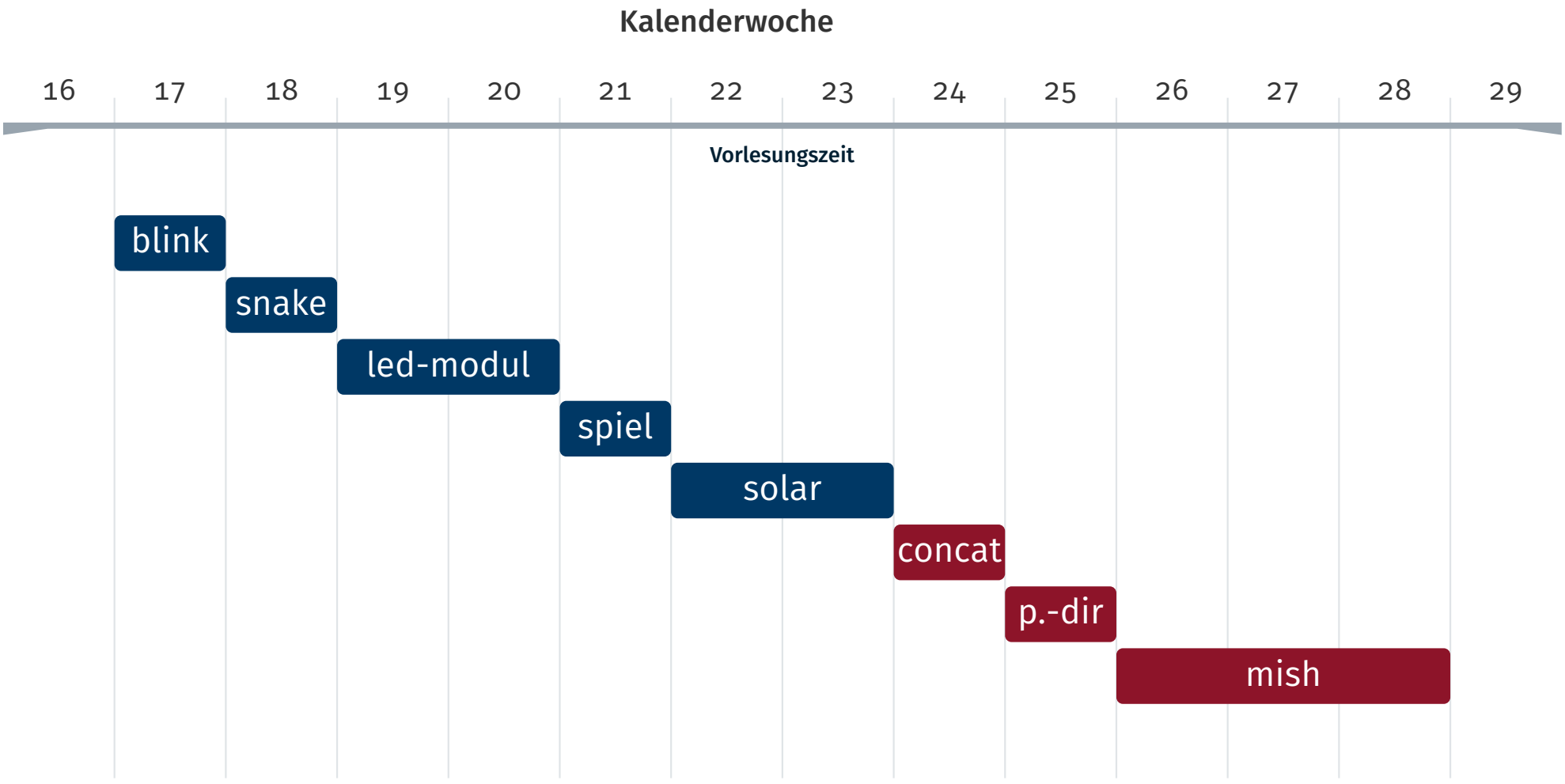
Tafelübungen

- Ablauf der Tafelübungen:
  1. Besprechung der alten Aufgabe
  2. Praxisnahe Vertiefung des Vorlesungsstoffes
  3. Vorstellung der neuen Aufgabe
  4. Ggf. Entwicklung einer Lösungsskizze der neuen Aufgabe
  5. Hands-on: gemeinsames Programmieren
- Folien nicht unbedingt zum Selbststudium geeignet
  → Anwesenheit, Mitschrift
- Semesterplan und Übersicht aller SPiC-Termine:
  `https://sys.cs.fau.de/lehre/ss25/spic/`

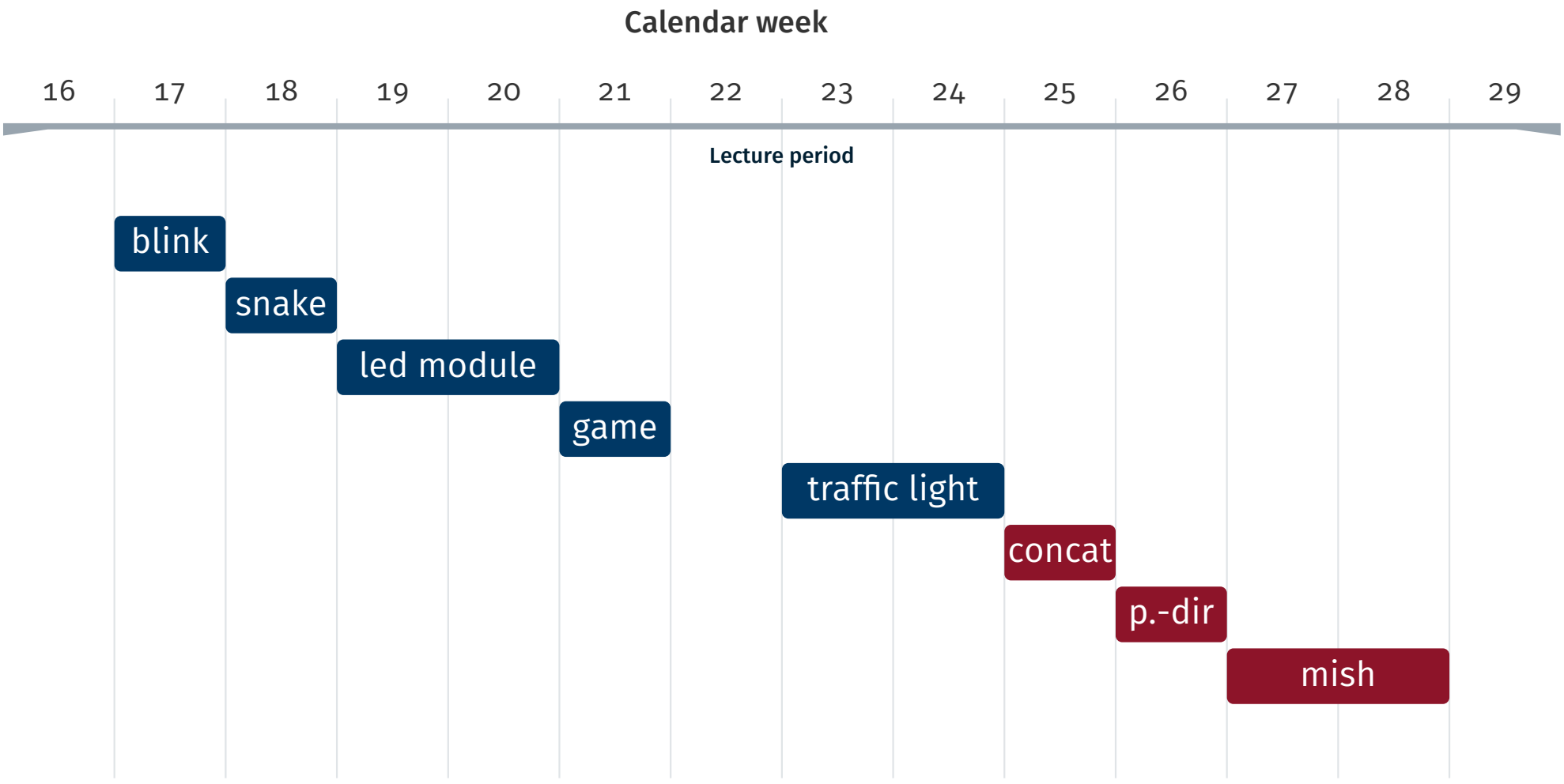
Tutorial

- Concept of Tutorial:
  1. Correct the last programming assignment
  2. Deepen lecture contents
  3. Introduction to the new programming assignments
  4. Possibly development of a solution sketch
  5. Hands-on: joined programming
- Slides are not necessarily made to be studied on their own
  → attendance required, write along
- Overview for the term and SLP appointments:
  `https://sys.cs.fau.de/lehre/ss25/slp/`

Aufgaben


Kalenderwoche
16
17
18
19
20
21
22
23
24
25
26
27
28
29
Vorlesungszeit
blink
snake
led-modul
spiel
solar
concat
p.-dir
mish

Assignments



GSPiC

- Studierende, die GSPiC belegen, müssen nur die Mikrocontroller-Aufgaben abgeben

  $\longrightarrow$ blink, snake, led-modul, spiel, solar
- Freiwillige Teilnahme an den Linux-Aufgaben ist selbstverständlich möglich
- Empfehlung: Letzte bzw. letzten Übungen zur Klausurvorbereitung

Lösungen

- Abgabe unter Linux
- Automatische Plagiatsprüfung
  - Vergleich mit allen anderen (auch älteren) Lösungen
  - abgeschriebene Lösungen bekommen 0 Punkte

  $\Rightarrow$ Im Zweifelsfall beim Übungsleiter melden
- Punktabzug
  - -1 Punkt je Compilerwarnung
  - -50% der möglichen Punkte falls nicht übersetzbar
- (Hilfreiche) Kommentare im Code helfen euch und dem Korrektor

Solutions

- Assignments are submitted via Linux
- Automatic check for plagiarism
  - Comparison to all other solutions (including old ones)
  - Plagiarism yields 0 points

  $\Rightarrow$ If in doubt talk to your tutor
- Deduction of points
  - -1 point for each compiler warning
  - -50% of possible points if the code does not compile
- (Helpful) comments in the code can help you and your tutor

Bonuspunkte

- Abgegebene Aufgaben werden mit Übungspunkten bewertet
- Ab 20% der erreichbaren Übungspunkte gibt es Bonuspunkte für die Klausur
- Ab 80% der erreichbaren Übungspunkte gibt es die vollen Bonuspunkte
- Umrechnung der Übungspunkte in Bonuspunkte für die Klausur (bis zu 10% der Punkte)

  ⟶ Beispiel: 80% der Übungspunkte führen bei 90 möglichen Klausurpunkten zu 9 Bonuspunkten
- Bestehen der Klausur durch Bonuspunkte *nicht möglich*
- Bonuspunkte nicht in nächste Semester übertragbar

Bonus Points

- Submitted assignments get graded with bonus points
- If you reach 20% or more of all bonus points, there is a bonus for the exam
- For 80% or more you get rewarded with full bonus points for the exam
- Conversion of points from the assignments into bonus points for the exam (up to 10% of points)

  $\longrightarrow$ Example: 80% of points from the assignments yield 9 bonus points if the exam has 90 points total
- However, you *cannot pass* the exam by the help of bonus points
- Bonus points cannot be transferred to the next semester

Rechnerübungen

- Raum der Rechnerübungen: 01.153-113 (WinCIP)
- Unterstützung durch Übungsleiter bei der Aufgabenbearbeitung

  Freie Plätze nach dem „First come, first served“-Prinzip
- Falls 30 Minuten nach Beginn der Rechnerübung niemand anwesend ist, kann der Übungsleiter gehen
- Termine auf der Webseite:

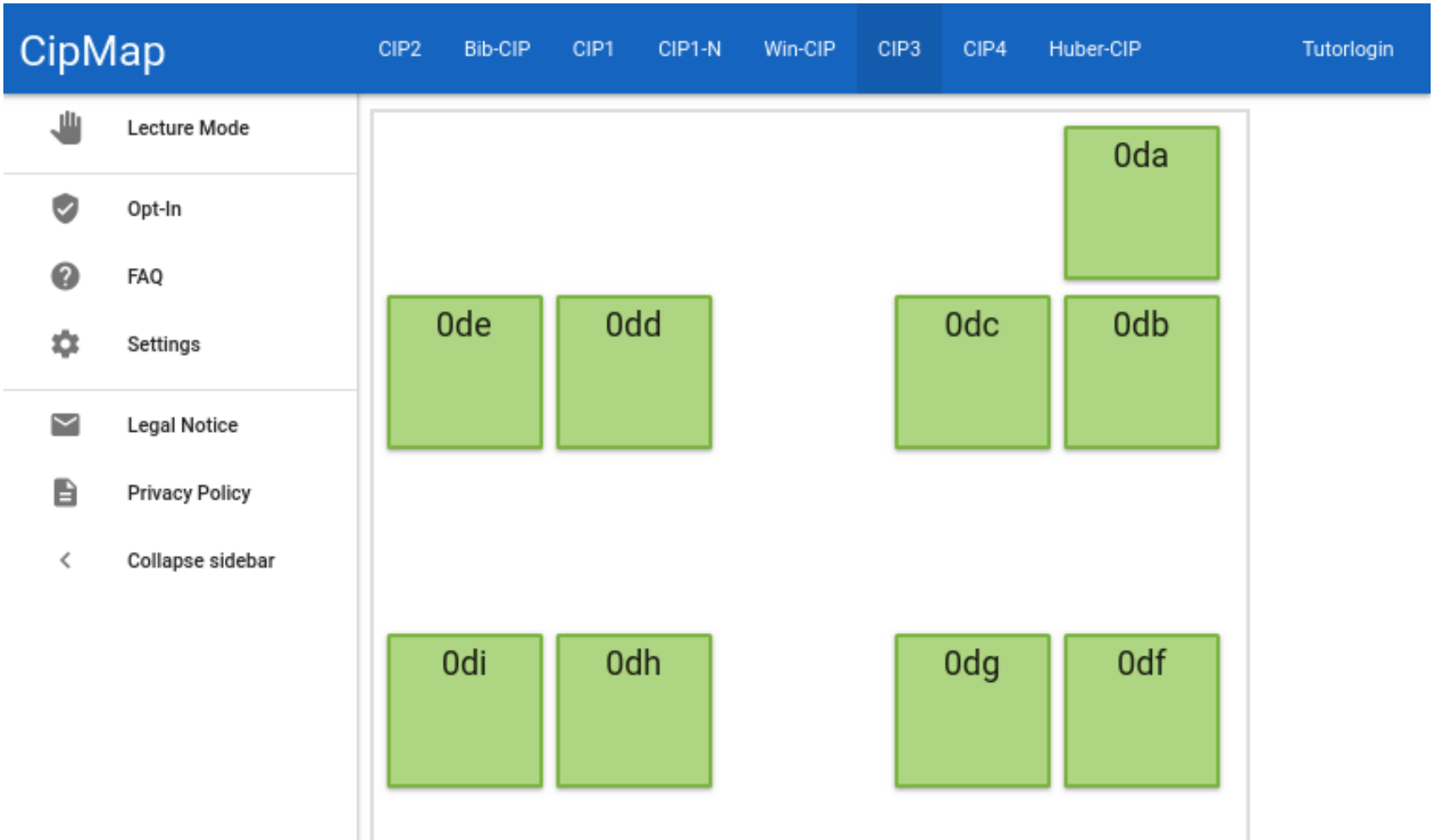
  `https://sys.cs.fau.de/lehre/ss25/spic/`

Computer Exercise

- Room for the Computer exercise: 01.153-113 (WinCIP)
- Help from the tutor during your work with the assignment

  „First come, first served“-principle
- If after 30 minutes after the beginning of the Computer exercise no student is present, the exercise is cancelled

CipMap


CipMap
CIP2
Bib-CIP
CIP1
CIP1-N
Win-CIP
CIP3
CIP4
Huber-CIP
Tutorlogin
Lecture Mode
Opt-In
FAQ
Settings
Legal Notice
Privacy Policy
Collapse sidebar
0da
0de
0dd
0dc
0db
0di
0dh
0dg
0df

Anfragen via CipMap stellen

1. Besuche die Seite cipmap.cs.fau.de
2. Wähle den Raum der Rechnerübung aus (z.B. 01.153-113)
3. Klicke auf *Lecture Mode*.
   - **farbiger Rechner:** Hat einen Request gestellt
   - **grauer Rechner:** Kein Request gestellt
4. Durch einen Klick auf *Request Tutor* wird deine Anfrage in die Warteschlange eingereiht
5. Nachdem deine Frage beantwortet wurde: Schaltfläche erneut klicken, um die Anfrage zurückzuziehen

Bitte beachte:

- Anfragen können nur während einer Übung gestellt werden
- Loggst du dich aus, werden deine Requests gelöscht

Make Requests via CipMap

1. Visit the site cipmap.cs.fau.de
2. Choose the room where the Computer exercise takes place (e. g. 01.153-113)
3. Click on *Lecture Mode*.
   - **colored PC:** request sent
   - **grey PC:** no request
4. By clicking *Request Tutor*, a request will be queued
5. After your question is answered: click on the button again to mark the request as finished

Please note:

- You can only make requests during the time of Computer exercises
- When logging off, all open requests get deleted

Bei Problemen

- Folien konsultieren
- Häufig gestellte Fragen (FAQ) und Antworten:

  `https://sys.cs.fau.de/lehre/ss25/spic/uebung/spicboard/faq`
- Fragen zum Stoff gerne im StudOn Forum:

  `https://www.studon.fau.de/studon/go/frm/6351556`
- Darüber hinaus gehende Fragen:

  Inhaltliche Fragen (Tutoren):
  i4spic@lists.cs.fau.de

  Organisatorische Fragen (Mitarbeiter):
  i4spic-orga@lists.cs.fau.de

If a Problem occurs

- Consult the slides
- Ask in the StudOn Forum:

  `https://www.studon.fau.de/studon/go/frm/6371345`
- Write an e-mail

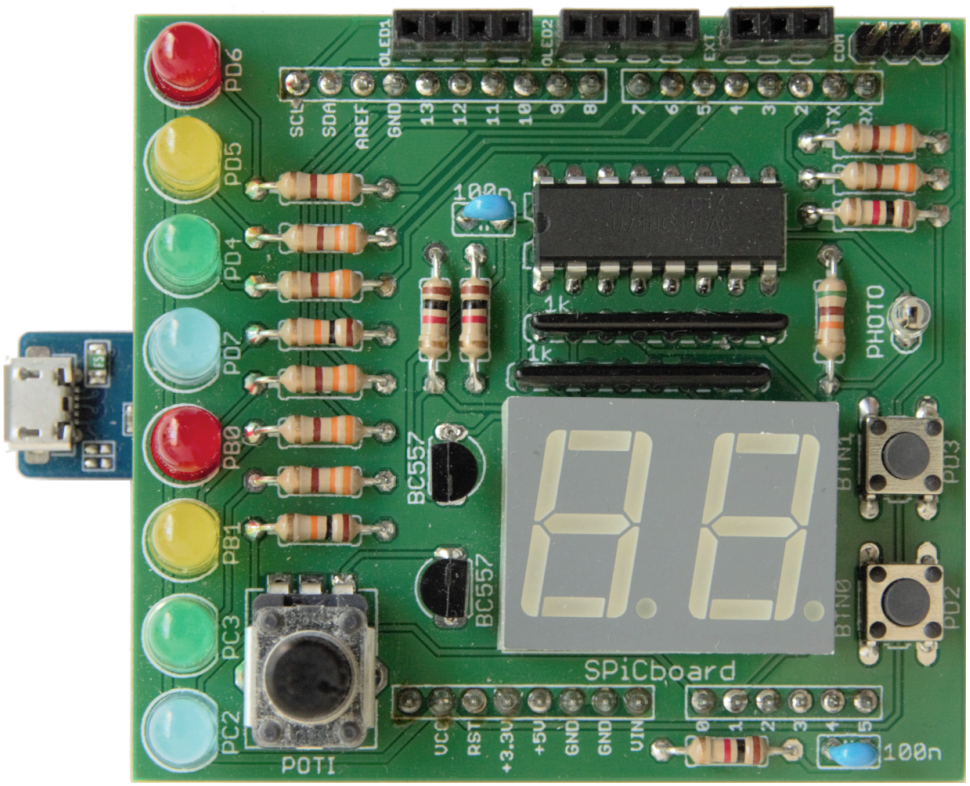
  Questions on lecture contents (tutors):
  i4slp@i4.cs.fau.de

  Organizational questions (all staff):
  i4slp-orga@i4.cs.fau.de

Development Environment

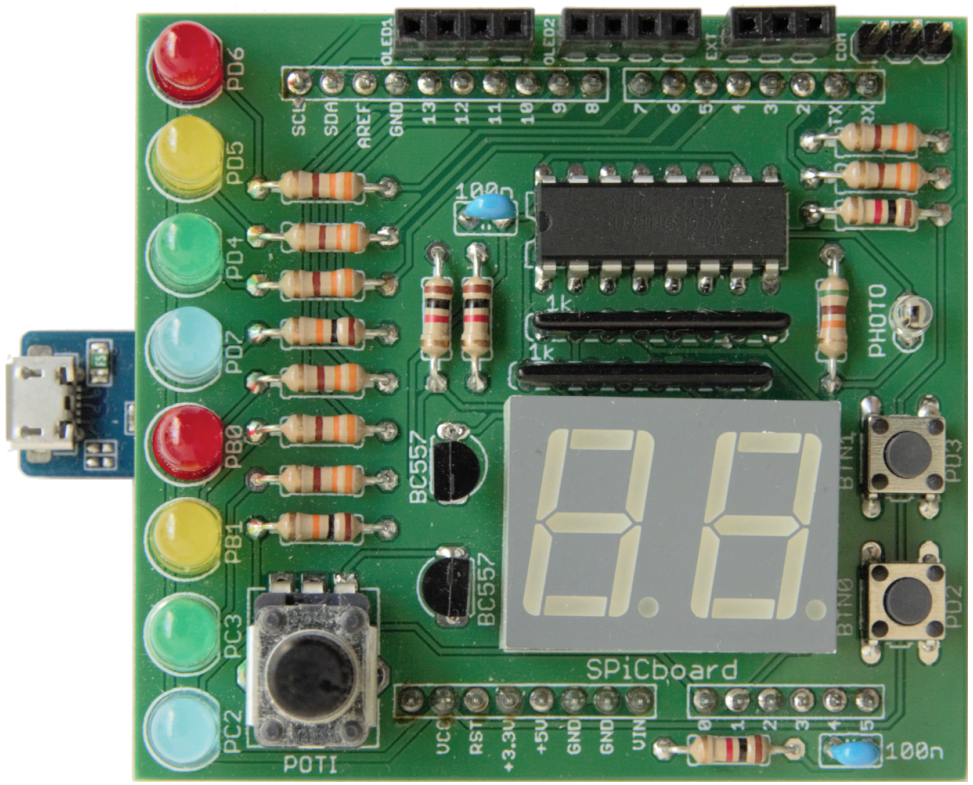
Hardware: SPiCboard

- **ATmega328PB Xplained Mini:**
  Mikrocontroller-Board mit integriertem Programmer/Debugger
- Speziell für SPiC angefertigte **SPiCboard**s als Erweiterungsplatine



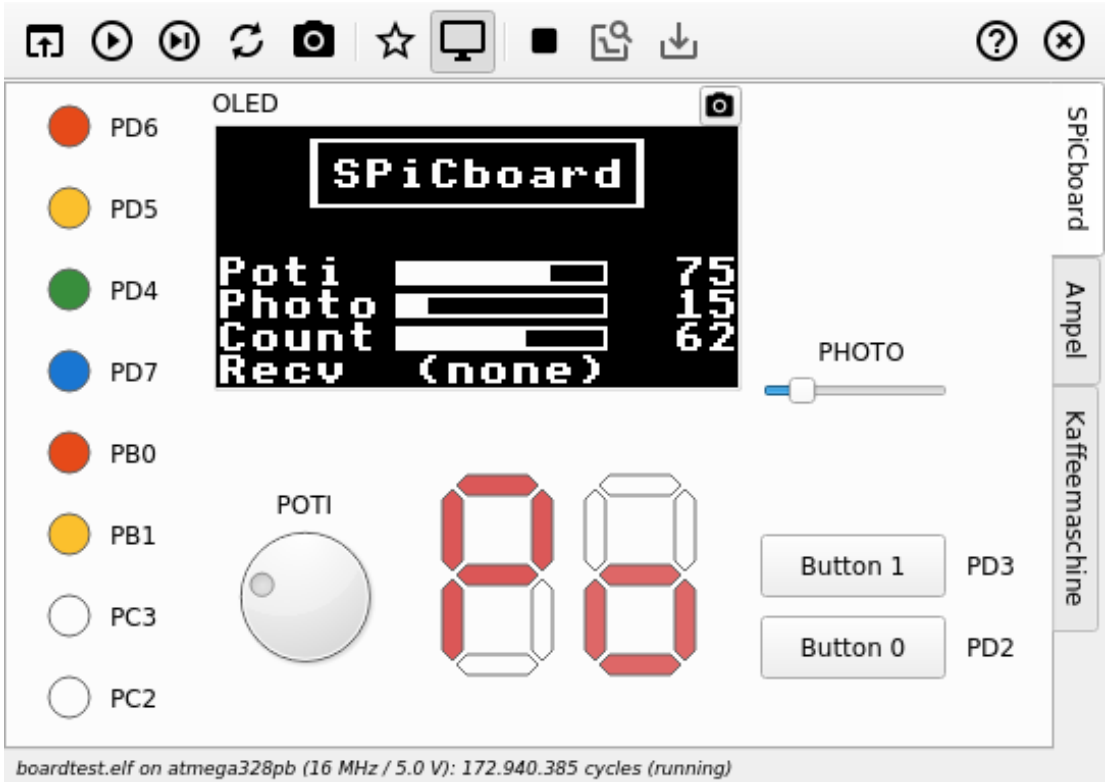
Hardware: SPiCboard

- **ATmega328PB Xplained Mini:**
  Micro-controller board with integrated programmer/debugger
- Custom-made extension PCB for SPiC/SLP



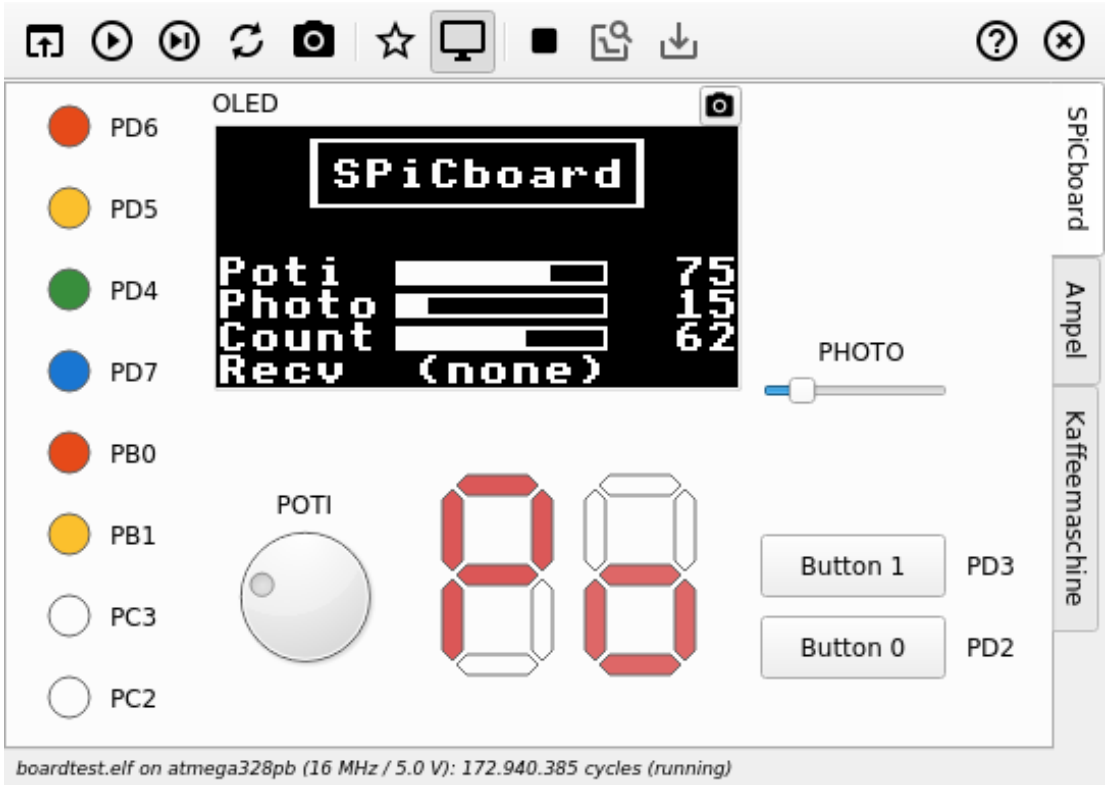
Simulator: SPiCsim

- **SPiCsim:**
  Simuliert ATmega328PB und SPiCBoard
- Erlaubt Aufzeichnung und Darstellung der Signale



Simulator: SPiCsim

- **SPiCsim:**
  Simulates ATmega328PB and SPiCBoard
- Makes recording and visualizing of signals possible



Aufgabenbearbeitung

- Betreute Bearbeitung der Aufgaben während der Rechnerübungen
  - ⇒ Hardware wird während der Übung zur Verfügung gestellt
- Selbständige Bearbeitung teilweise nötig
  - eigenes SPiCboard: Anfertigung am Lötabend (nur im Sommersemester)
  - SPiCboard Simulator: `SPiCsim`

Handling the Assignments

- Supervised programming for the assignments during Computer exercises
  - ⇒ Hardware is made available during the exercises
- Independent working style (partially) required
  - Using own SPiCboard: can be soldered at the soldering night
  - SPiCboard Simulator: `SPiCsim`

Funktionsbibliothek

- `libspicboard`: Funktionsbibliothek zur Ansteuerung der Hardware

  Beispiel: `sb_led_on(GREEN0);` schaltet 1. grüne LED an
- Direkte Konfiguration der Hardware durch Anwendungsprogrammierer nicht nötig
- Verwendung vor allem bei den ersten Aufgaben, später muss `libspicboard` teils selbst implementiert werden
- Dokumentation online:

  `https://sys.cs.fau.de/lehre/ss25/spic/uebung/spicboard/libapi`

Function Libraries

- `libspicboard`: function library for addressing the hardware

  Example: `sb_led_on(GREEN0);` switches on the first green LED
- Direct configuration of the hardware by the application developer is not needed
- Usage mainly for the first assignments, later the functions of the `libspicboard` have to be implemented by yourself
- Documentation online:

  `https://sys.cs.fau.de/lehre/ss25/spic/uebung/spicboard/libapi`

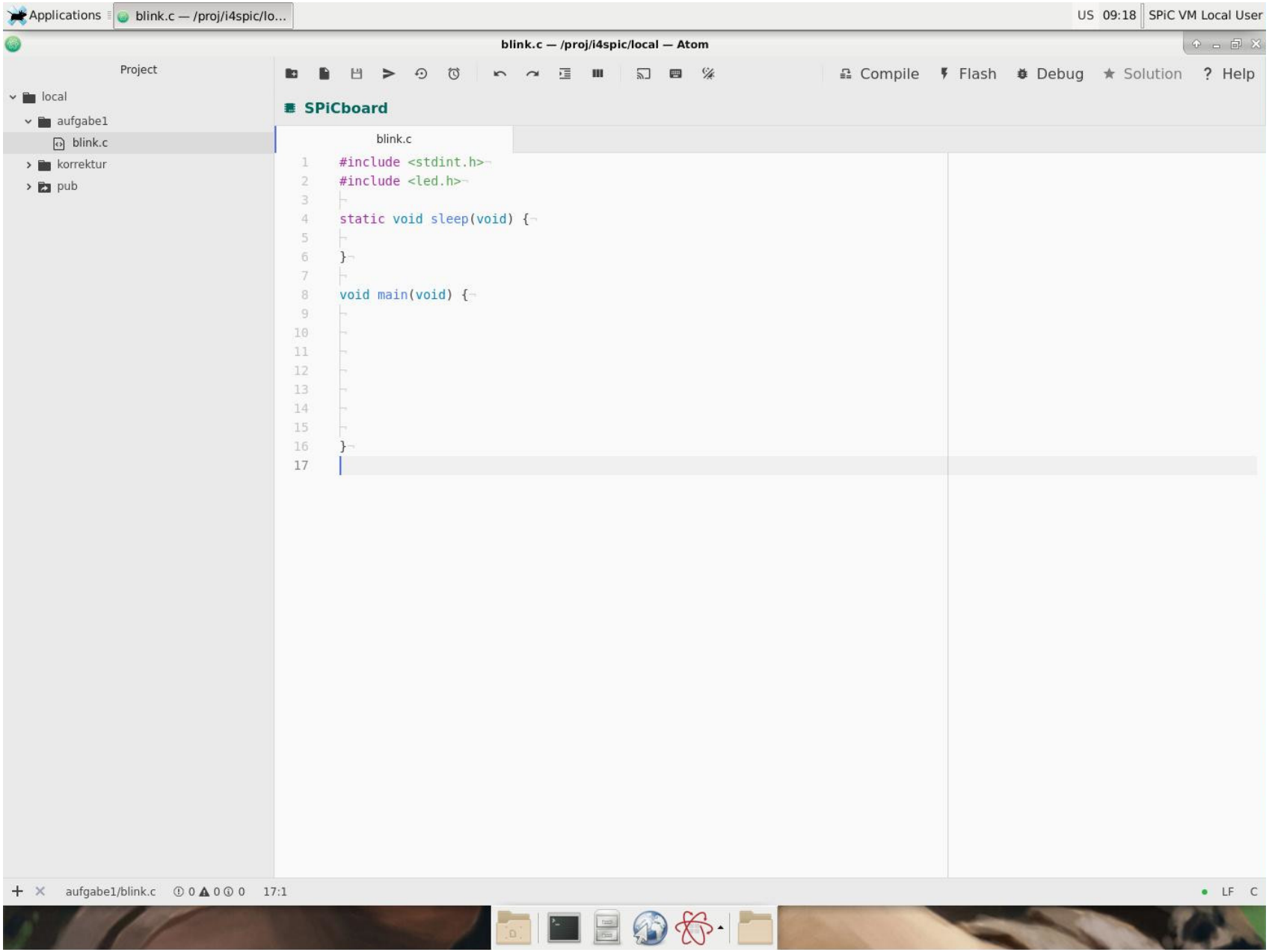
Wichtige Verzeichnisse

- Vorgabeverzeichnis `/proj/i4spic/<idm-login>/pub/`
  - Hilfsmaterial zu jeder Übungsaufgabe unter `aufgabeX/`
  - `libspicboard` mit Dokumentation sowie minimalem Beispiel
  - Die Vorlesungsfolien in `vorlesung/` (VM: Nur in der Remote-IDE)
  - Die Übungsfolien in `uebung/` (VM: Nur in der Remote-IDE)
  - Hilfestellung zur Programmiersprache *C* (VM: Nur in der Remote-IDE)
- Projektverzeichnis
  - `/proj/i4spic/<idm-login>/`
  - Lösungen hier in Unterordnern `aufgabeX` speichern
    - ⇒ Das Abgabeprogramm sucht (nur) dort
  - Für andere nicht lesbar
  - Wird automatisch erstellt
  - Enthält symbolische Verknüpfung zum Vorgabeverzeichnis

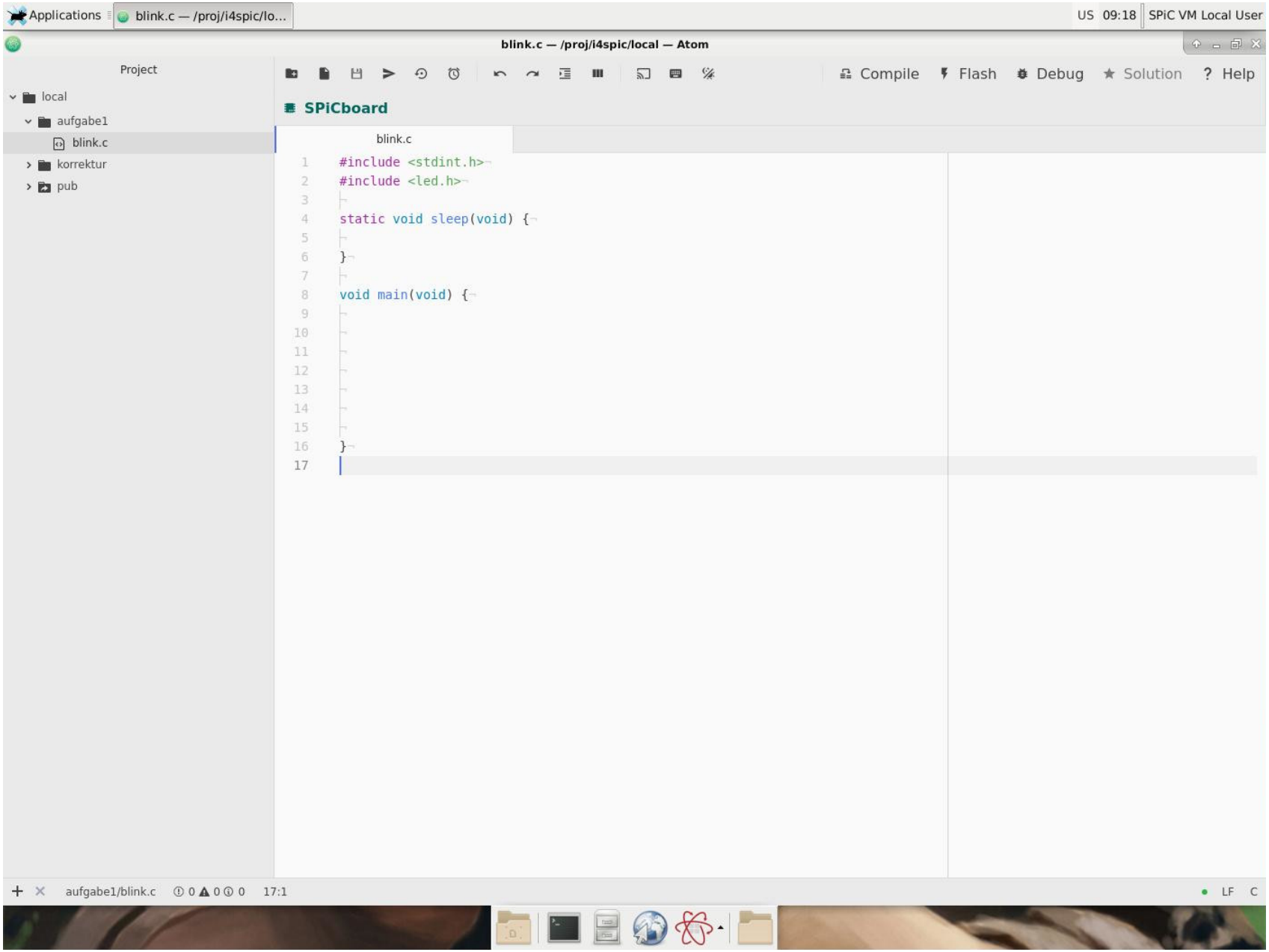
Important Directories

- Public directory `/proj/i4spic/<idm-login>/pub/`
  - Auxiliary material for each assignment can be found in `aufgabeX/`
  - `libspicboard` with documentation and minimal working examples
  - All lecture slides in `lecture/`
  - All exercise slides in `exercise/`
  - Assistance for dealing with the language *C*
- Project directory
  - `/proj/i4spic/<idm-login>/`
  - Solutions have to be saved in subdirectories `aufgabeX`
    - ⇒ The program for submitting searches only there
  - Others cannot read this directory
  - Directory is created automatically
  - Contains symbolic links to the public directory

Der Editor


Applications
blink.c — /proj/i4spic/lo...
US 09:18
SPiC VM Local User
blink.c — /proj/i4spic/local — Atom
Project
local
aufgabe1
blink.c
korrektur
pub
Compile
Flash
Debug
Solution
Help
SPiCboard
blink.c
#include <stdint.h>
#include <led.h>
static void sleep(void) {
}
void main(void) {
}
aufgabe1/blink.c
17:1
LF
C

The Editor


Applications
blink.c — /proj/i4spic/lo...
US 09:18
SPiC VM Local User
blink.c — /proj/i4spic/local — Atom
Project
local
aufgabe1
blink.c
korrektur
pub
Compile
Flash
Debug
Solution
Help
SPiCboard
blink.c
#include <stdint.h>
#include <led.h>
static void sleep(void) {
}
void main(void) {
}
aufgabe1/blink.c
17:1
LF
C

Manuals

CIP-Login

- Für die Benutzung der CIP Infrastruktur (und damit des Abgabesystems) ist ein CIP Login nötig

  Bei Problemen bitte an die CIP Admins wenden

- Kriterien für sicheres Passwort:
  - Mindestens 8 Zeichen, besser 10
  - Mindestens 3 Zeichensorten, besser 4 (Groß-, Kleinbuchstaben, Zahlen, Sonderzeichen)
  - Keine Wörterbuchwörter, Namen, Login, etc.

CIP-Login

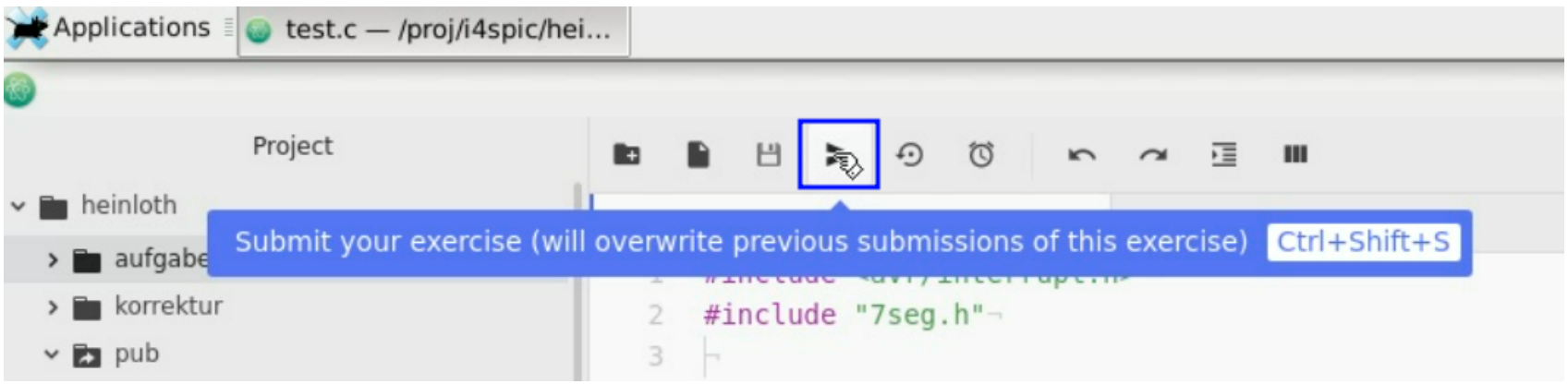
- To use the CIP infrastructure (and therefore the tools for assignment submission) a login for the CIP is required

  When running into problems, please contact the CIP Admins

- Criteria for a secure password
  - At least 8 characters, 10 is better
  - At least 3 different types of characters, 4 are better (capitalized letters, small letters, digits, special characters)
  - **Do not** use any dictionary words, names, login, etc.

Abgabe (1)

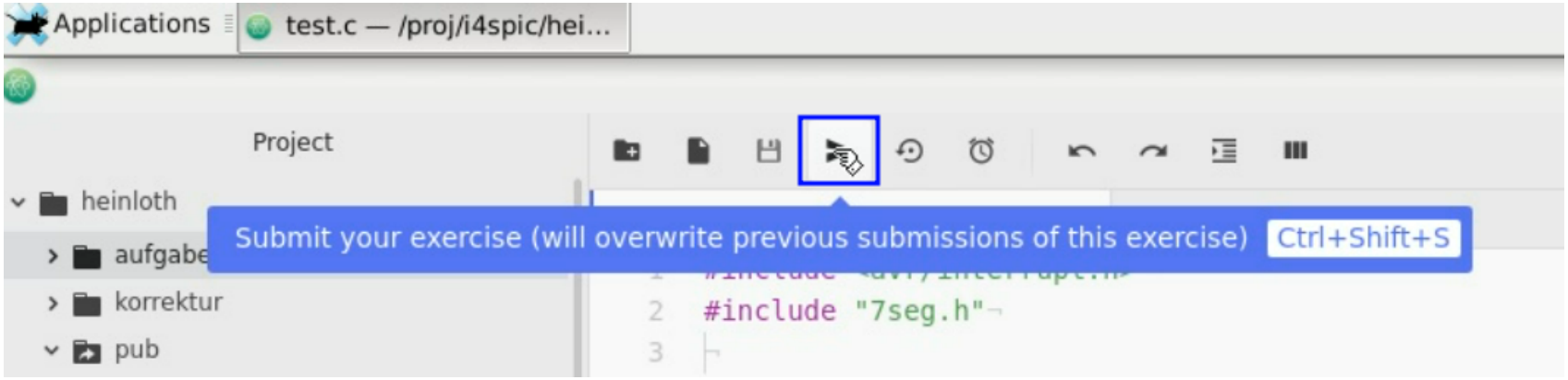
- Spätestens nach erfolgreichem Testen des Programms müssen Übungslösungen zur Bewertung abgegeben werden
- Bei Zweiergruppen darf nur ein Partner abgeben!
  - Der Partner muss aus der selben Gruppe sein
  - Bei der Abgabe wird der Partner-Login hinterlegt
- Abgabe entweder per SPiC IDE Button



- Oder Terminal-Fenster öffnen und folgendes Kommando ausführen (`aufgabeX` entsprechend ersetzen):
  `/proj/i4spic/bin/submit aufgabeX`
  - Wichtig: **Grüner Text** signalisiert erfolgreiche Abgabe, **roter Text** einen Fehler!

Submitting Assignments (1)

- At the latest after testing the program, you should submit your solution for grading
- When working with a partner, only ONE of you is allowed to submit the assignment!
  - Your partner has to take part in the same Tutorial
  - When submitting, you can specify your partner
- Submission in the SPiC IDE with the click of a button



- Or open a terminal window and execute the following command (`aufgabeX` has to be replaced):
  `/proj/i4spic/bin/submit aufgabeX`
  - Important: **green text** indicates that the submission was

Abgabe (2)

- Fehlerursachen
  - Notwendige Dateien liegen nicht im richtigen Ordner
  - `aufgabeX` muss klein geschrieben sein
  - `.c`-Datei falsch benannt
  - Abgabetermin verpasst
- Nützliche Tools
  - Quelltext der abgegebenen Aufgabe anzeigen:
    `/proj/i4spic/bin/show-submission aufgabeX`
  - Unterschiede zwischen abgegebener Version und Version im Projektverzeichnis `/proj/i4spic/<login>` anzeigen:
    `/proj/i4spic/bin/show-submission aufgabeX -d`
  - Eigenen Abgabetermin anzeigen:
    `/proj/i4spic/bin/get-deadline aufgabeX`

Submitting Assignments (2)

- Causes for an error
  - Necessary files are not present in the right directory
  - `aufgabeX` has to be written without capitalization
  - `.c`-file has been wrongly named
  - Deadline was missed
- Useful tools
  - Show the source code of the submitted assignment:
    `/proj/i4spic/bin/show-submission aufgabeX`
  - Differences between submitted version and current version in the project directory `/proj/i4spic/<login>`:
    `/proj/i4spic/bin/show-submission aufgabeX -d`
  - Show deadline:
    `/proj/i4spic/bin/get-deadline aufgabeX`

Eure Todos

1. Anmeldung in StudOn:

   `https://www.studon.fau.de/studon/go/crs/6151950`

   - Forum zum Fragen stellen
2. Anmeldung zu den Übungen über Waffel: `https://waffel.cs.fau.de`

   - Für Abgabe und Korrektur der Aufgaben

   ⇒ ab **Freitag, 25.04.2025, 18:00 Uhr**
3. Anmeldung im Informatik CIP: `https://account.cip.cs.fau.de`
   - Für Bearbeiten, Abgabe und Korrektur der Aufgaben

Da es bis zu 24h dauern kann, bis nach der Anmeldung die erforderlichen Änderungen aktiv sind, solltet ihr euch **umgehend darum kümmern**. Vorher ist eine Bearbeitung und Abgabe der Übungsaufgaben nicht möglich!

Your Todos

1. Registration in StudOn:

   `https://www.studon.fau.de/studon/go/crs/6151967`

   - Forum for Questions
2. Registration for the exercises via Waffel: `https://waffel.cs.fau.de`
   - For submission and correction of assignments

   ⇒ from **Friday, 25.04.2025, 6:00 PM**
3. Registration for the CIP: `https://account.cip.cs.fau.de`
   - For working on the assignments, submitting them and receiving feedback

Since the registration for the CIP can take up to 24 hours until you can log in with your new account, please make sure to **register asap**. Without an account you cannot take part in working on the assignments!

Compiler Optimizations

Compileroptimierung: Hintergrund

- AVR-Mikrocontroller, sowie die allermeisten CPUs, können ihre Rechenoperationen nicht direkt auf Variablen ausführen, die im Speicher liegen
- Ablauf von Operationen:
  1. **Laden** der Operanden aus dem Speicher in Prozessorregister
  2. **Ausführen** der Operationen in den Registern
  3. **Zurückschreiben** des Ergebnisses in den Speicher

  $\Rightarrow$ Detaillierte Behandlung in der Vorlesung
- Der Compiler darf den Code nach Belieben ändern, solange der “globale” Zustand beim Verlassen der Funktion gleich bleibt
- Optimierungen können zu drastisch schnellerem Code führen

Compiler Optimizations: Background

- AVR micro-controller, as well as nearly all CPUs cannot execute operations directly on memory
- Procedure of operations:
  1. **Load** the operands from the memory into processor registers
  2. **Execute** the operations using the registers
  3. **Store** the result into memory

  $\Rightarrow$ More detailed description in the lecture
- The compiler is allowed to arbitrarily change the code as long as the “global” state after exiting a function stays the same
- Optimizations can lead to drastically faster code

Compileroptimierung: Beispiele

- Typische Optimierungen:
  - Beim Betreten der Funktion wird die Variable in ein Register geladen und beim Verlassen in den Speicher zurückgeschrieben
  - Redundanter und “toter” Code wird weggelassen
  - Die Reihenfolge des Codes wird umgestellt
  - Für automatic Variablen wird kein Speicher reserviert; es werden stattdessen Prozessorregister verwendet
  - Wenn möglich, übernimmt der Compiler die Berechnung (Konstantenfaltung):
    `a = 3 + 5;` wird zu `a = 8;`
  - Der Wertebereich von automatic Variablen wird geändert:
    Statt von `0` bis `10` wird von `246` bis `256` (`= 0` für `uint8_t`) gezählt und dann geprüft, ob ein Überlauf stattgefunden hat

Compiler Optimizations: Examples

- Typical optimizations:
  - When entering a function the variable is loaded into a register and only written back to memory when leaving the function
  - Redundant and “dead” code is removed
  - Some instructions get reordered
  - For automatic variables no memory is reserved; they are placed in processor registers instead
  - If possible, the compiler does some calculations (constant folding):
    `a = 3 + 5;` is replaced `a = 8;`
  - The range of values of automatic variables gets adapted: Instead of `0` to `10`, one can count from `246` to `256` (`= 0` for `uint8_t`) and then check if an overflow occurred

Compileroptimierung: Beispiel (1)

```
01 void wait(void) {
02   uint8_t u8 = 0;
03   while(u8 < 16) {
04     u8++;
05   }
06 }
```

- Inkrementieren der Variable u8 bis 16
- Verwendung z.B. für aktive Warteschleifen

Compiler Optimizations: Example (1)

```
01 void wait(void) {
02   uint8_t u8 = 0;
03   while(u8 < 16) {
04     u8++;
05   }
06 }
```

- Incrementing the variable u8 up to a value of 16
- Used for e. g. active waiting

Compileroptimierung: Beispiel (2)

- Assembler ohne Optimierung

```
01 ; void wait(void){
02 ; uint8_t u8;
03 ; [Prolog (Register sichern, Y initialisieren, etc.)]
04 rjmp while      ; Springe zu while
05 ; u8++;
06 addone:
07 ldd r24, Y+1    ; Lade Daten aus Y+1 in Register 24
08 subi r24, 0xFF  ; Ziehe 255 ab (addiere 1)
09 std Y+1, r24    ; Schreibe Daten aus Register 24 in Y+1
10 ; while(u8 < 16)
11 while:
12 ldd r24, Y+1    ; Lade Daten aus Y+1 in Register 24
13 cpi r24, 0x10   ; Vergleiche Register 24 mit 16
14 brcs addone     ; Wenn kleiner, dann springe zu addone
15 ;[Epilog (Register wiederherstellen)]
16 ret             ; Kehre aus der Funktion zurück
17 ;}
```

Compiler Optimizations: Example (2)

- Assembler without optimizations

```
01 ; void wait(void){
02 ; uint8_t u8;
03 ; [Prologue (store registers, initialize Y, etc.)]
04 rjmp while       ; jump to while
05 ; u8++;
06 addone:
07 ldd r24, Y+1     ; load data from Y+1 into register 24
08 subi r24, 0xFF   ; subtract 255 (add 1)
09 std Y+1, r24     ; write data from register 24 into Y+1
10 ; while(u8 < 16)
11 while:
12 ldd r24, Y+1     ; load data from Y+1 into register 24
13 cpi r24, 0x10    ; compare register 24 with 16
14 brcs addone      ; if smaller, jump to addone
15 ;[Epilogue (restore registers)]
16 ret              ; return from the function
17 ;}
```

Compileroptimierung: Beispiel (3)

- Assembler mit Optimierung

```
01 ; void wait(void){
02 ret              ; Kehre aus der Funktion zurück
03 ; }
```

- C kennt die Wartesemantik der Schleife nicht
- Die Schleife hat keine Auswirkung auf den (globalen) Zustand

⇝ Der Compiler optimiert sie komplett weg

Compiler Optimizations: Example (3)

- Assembler with optimizations

```
01 ; void wait(void){
02 ret              ; Return from the function
03 ; }
```

- C does not know the semantics of a waiting loop
- The loop does not have any effect on the (global) state

⇝ The compiler optimises the loop by removing it

Schlüsselwort volatile

- Variable können als `volatile` (engl. unbeständig, flüchtig) deklariert werden

⇝ Der Compiler darf die Variable nicht optimieren:

  - Für die Variable muss **Speicher reserviert** werden
  - Die **Lebensdauer** darf nicht verkürzt werden
  - Die Variable muss vor jeder Operation aus dem **Speicher geladen** und danach ggf. wieder in diesen zurückgeschrieben werden
  - Der **Wertebereich** der Variable darf nicht geändert werden

- Einsatzmöglichkeiten von `volatile`:
  - Warteschleifen: Verhinderung der Optimierung der Schleife
  - Nebenläufigen Ausführungen (später in der Vorlesung)
    - Variable wird im Interrupthandler und der Hauptschleife verwendet
    - Änderungen an der Variable müssen “bekannt gegeben werden”
  - Zugriff auf Hardware (z.B. Pins) ⇝ wichtig für das LED Modul
  - (Debuggen: der Wert wird nicht wegoptimiert)

Keyword volatile

- Variables can be declared as `volatile`

⇝ The compiler is not allowed to optimise the variable:

  - **Memory has to be reserved** for the variable
  - The **life span** cannot be shortened
  - Prior to each operation, the variable has to be **loaded from memory** and afterwards it has to be written back to memory
  - The **range of value** of the variable cannot be adapted

- Possible uses of `volatile`:
  - Active waiting loops: prevents optimization of the loop
  - Concurrent execution (later in the lecture)
    - Variable is used in the interrupt handler and in the main loop
    - Changes of the variable have to be “made observable”
  - Access to hardware (e. g. pins) ⇝ important for the LED module
  - (Debugging: the value cannot be removed due to optimizations)

Task: blink

Aufgabenbeschreibung: blink

- Lernziel:
  - Umgang mit Programmierwerkzeugen und dem Abgabesystem
  - Aktives Warten
- Blinkende LEDs `YELLOW0` und `YELLOW1`
  - Abwechselnd an- bzw. ausschalten (Warnlicht)
  - Frequenz ca. 1 mal pro halbe Sekunde
  - Nutzung der Bibliotheksfunktionen für LEDs
  - Implementierung durch aktives Warten (Schleife mit Zähler)
- Dokumentation der Bibliothek:

  `https://sys.cs.fau.de/lehre/ss25/spic/uebung/spicboard/libapi`
- Abzugebende Datei: `blink.c`

Task Description: blink

- Learning objective:
  - Make first experiences with the programming environment and the submission system
  - Active waiting
- Flashing LEDs `YELLOW0` and `YELLOW1`
  - Switching on and off alternately (warning light)
  - Frequency of approx. 2 times per second
  - Use of the library functions for addressing the LEDs
  - Implementation by active waiting (loop with counter)
- Documentation of the library:

  `https://sys.cs.fau.de/lehre/ss25/spic/uebung/spicboard/libapi`
- File to be submitted: `blink.c`

Hands-on: Licht

Screencast: `https://www.video.uni-erlangen.de/clip/id/13444`

Hands-on: Light

Screencast: `https://www.video.uni-erlangen.de/clip/id/13444`

Hands-on: Licht

- In der SPiC-IDE:
  - Neuen Ordner erstellen (z.B. hands-on/licht)
  - Neue Quellcodedatei erstellen (z.B. `licht.c`)
- Programm erstellen:
  - Schalte eine LED ein (z.B. `GREEN0`)
  - Warte in einer Endlosschleife
- In der SPiC-IDE:
  - Programm übersetzen
  - Programm im Simulator oder auf dem SPiCboard testen.

Hands-on: Light

- Inside the SPiC-IDE:
  - Create new folder (e. g. hands-on/licht)
  - Create new source file (e. g. `licht.c`)
- Create the program:
  - Switch on one LED (e. g. `GREEN0`)
  - Wait inside an endless loop
- Inside the SPiC-IDE:
  - Compile the program
  - Test and execute the program in the simulator or on an actual SPiCboard