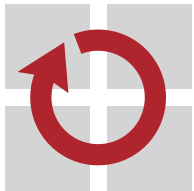
Übungen zu Systemnahe Programmierung in C

Abschnitt WS.4: Aufgabe (button)

23.11.2020

Tim Rheinfels
Benedict Herzog
Bernhard Heinloth

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

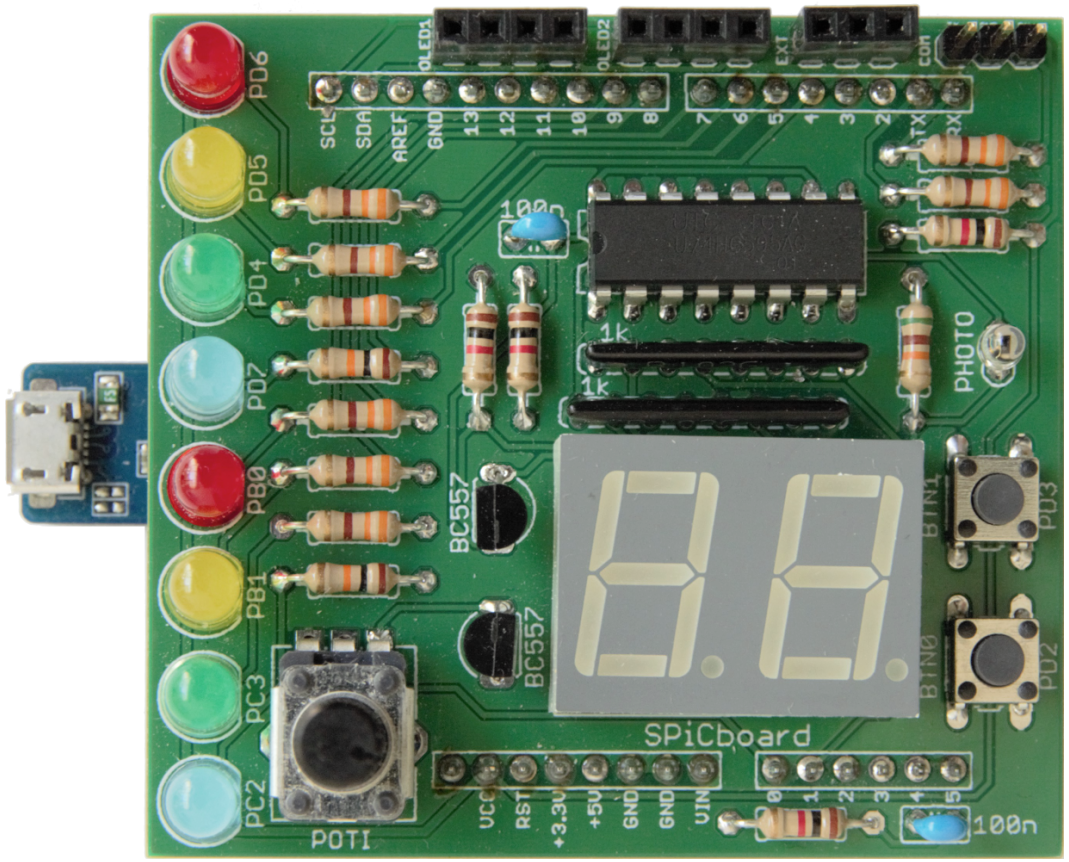


Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

BUTTON-Modul – Übersicht

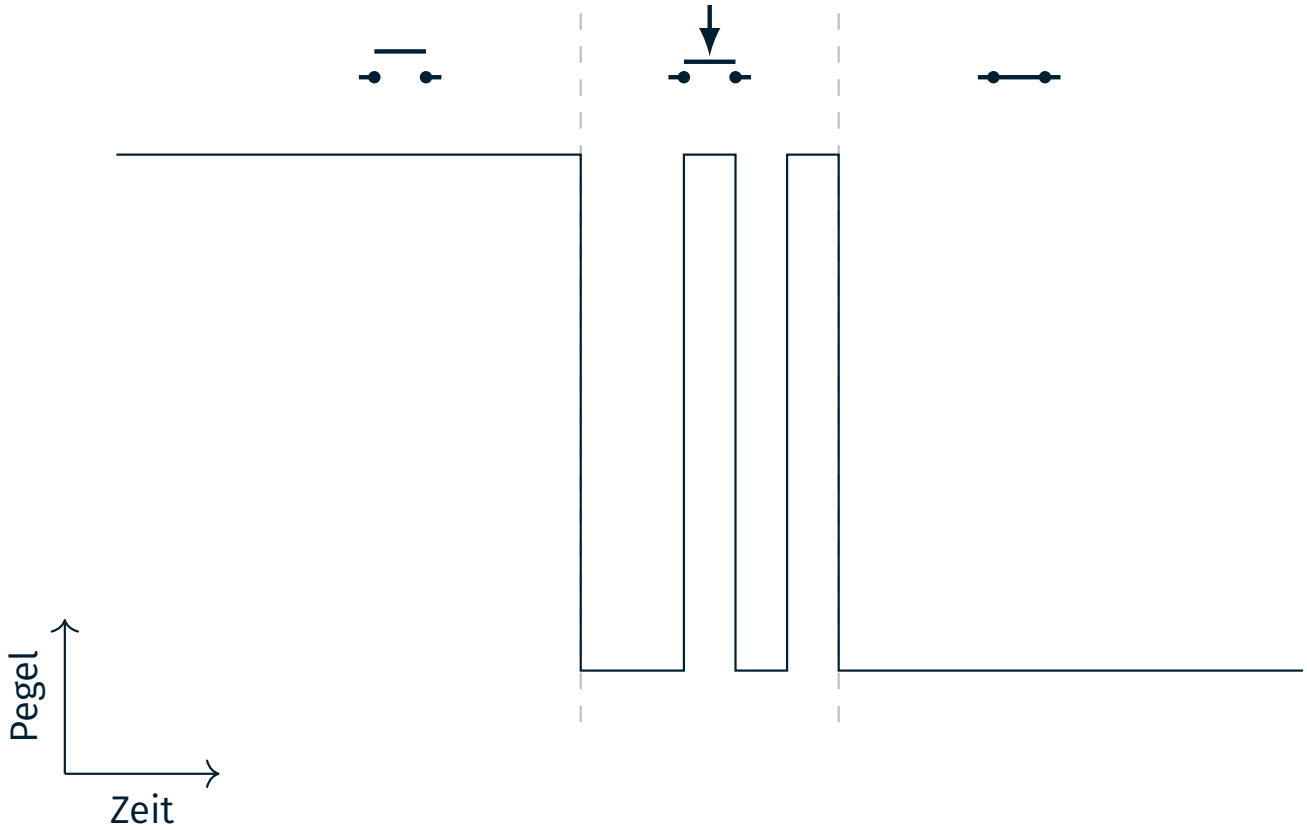


- Die `libspicboard` besteht aus verschiedenen Modulen

BUTTON-Modul – Aufgabe

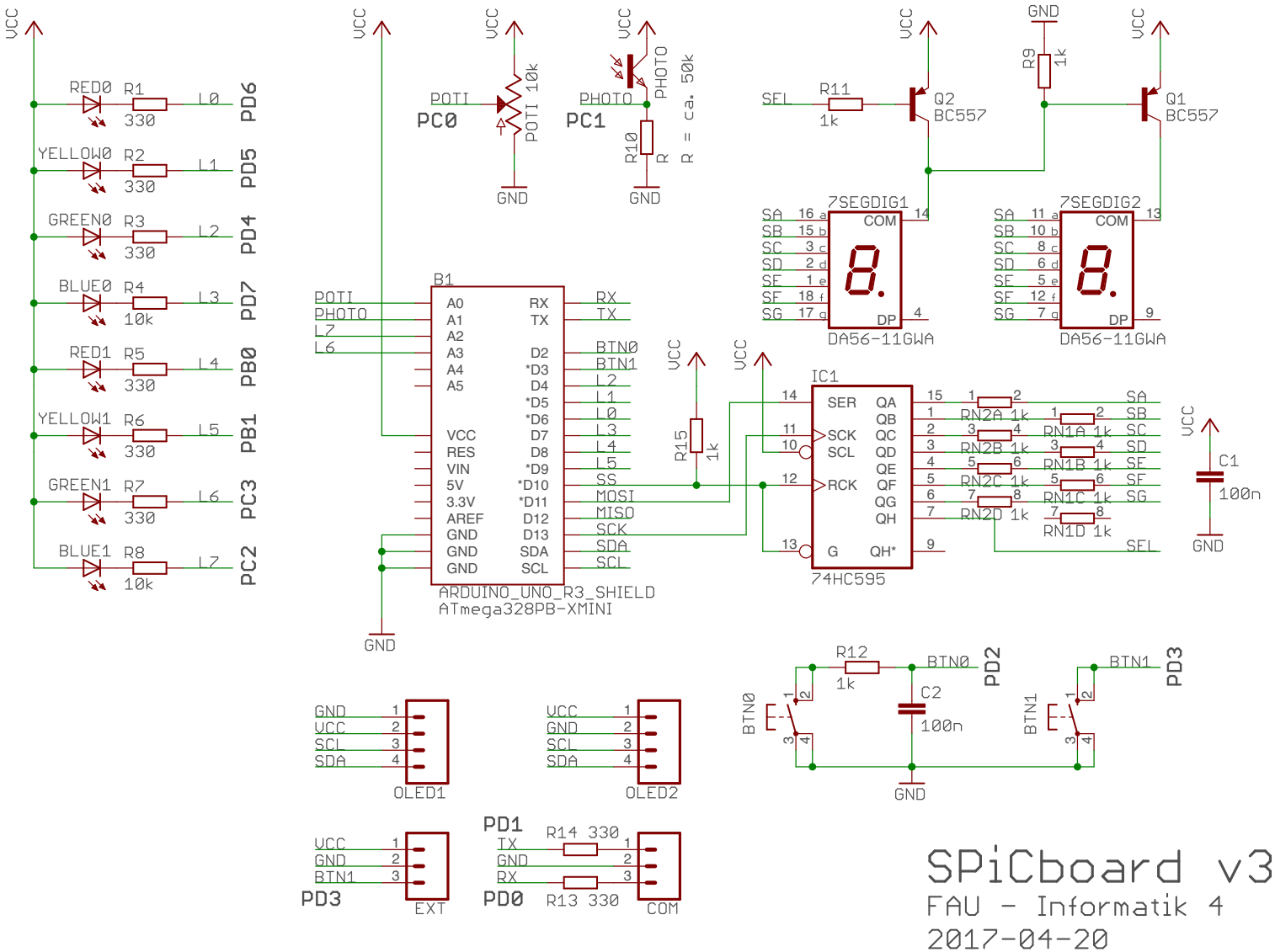
- BUTTON-Modul der `libspicboard` selbst implementieren
  - Zustandsabfrage: Gleiches Verhalten wie das Original
  - Callbacks: Platzhalter
  - Beschreibung: `http://www4.cs.fau.de/Lehre/WS20/V_SPIC/SPiCboard/group__Button.shtml`
- Testen des Moduls
  - Eigenes Modul gegen ein Testprogramm (`test-button.c`) linken
  - Andere Teile der Bibliothek können für den Test benutzt werden
- Taster des SPiCboards
  - Beide Taster sind **active-low** beschalten
  - Anschluss: `BUTTON0` an `PD2`, `BUTTON1` an `PD3`
  - Nomenklatur: `PD2` = Port D, Pin 2

Kontaktprellen

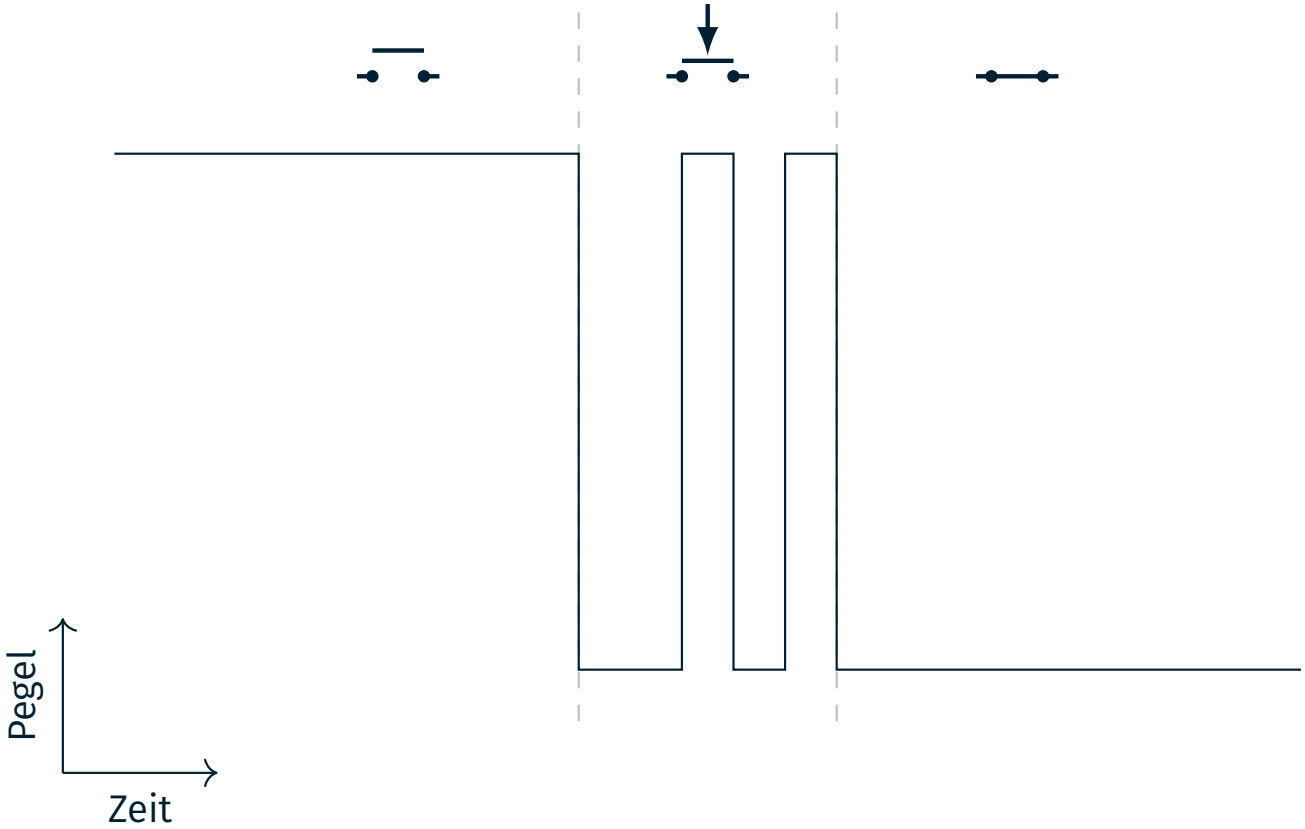


- Mechanischer Effekt beim Umschalten
- Erzeugt überzählige Flanken
- Entprellen in Hardware/Software möglich

SPiCboard Schaltplan

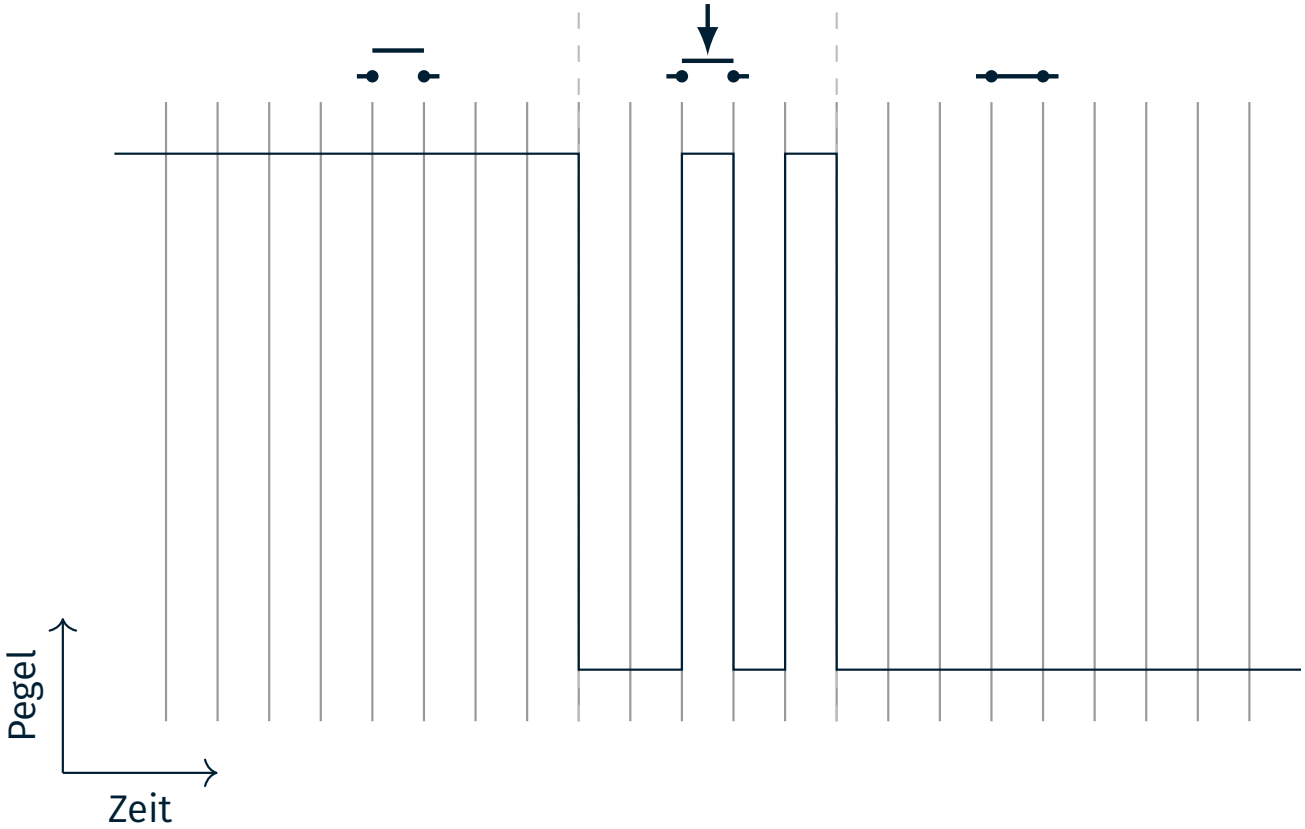


Mittelwertfilter



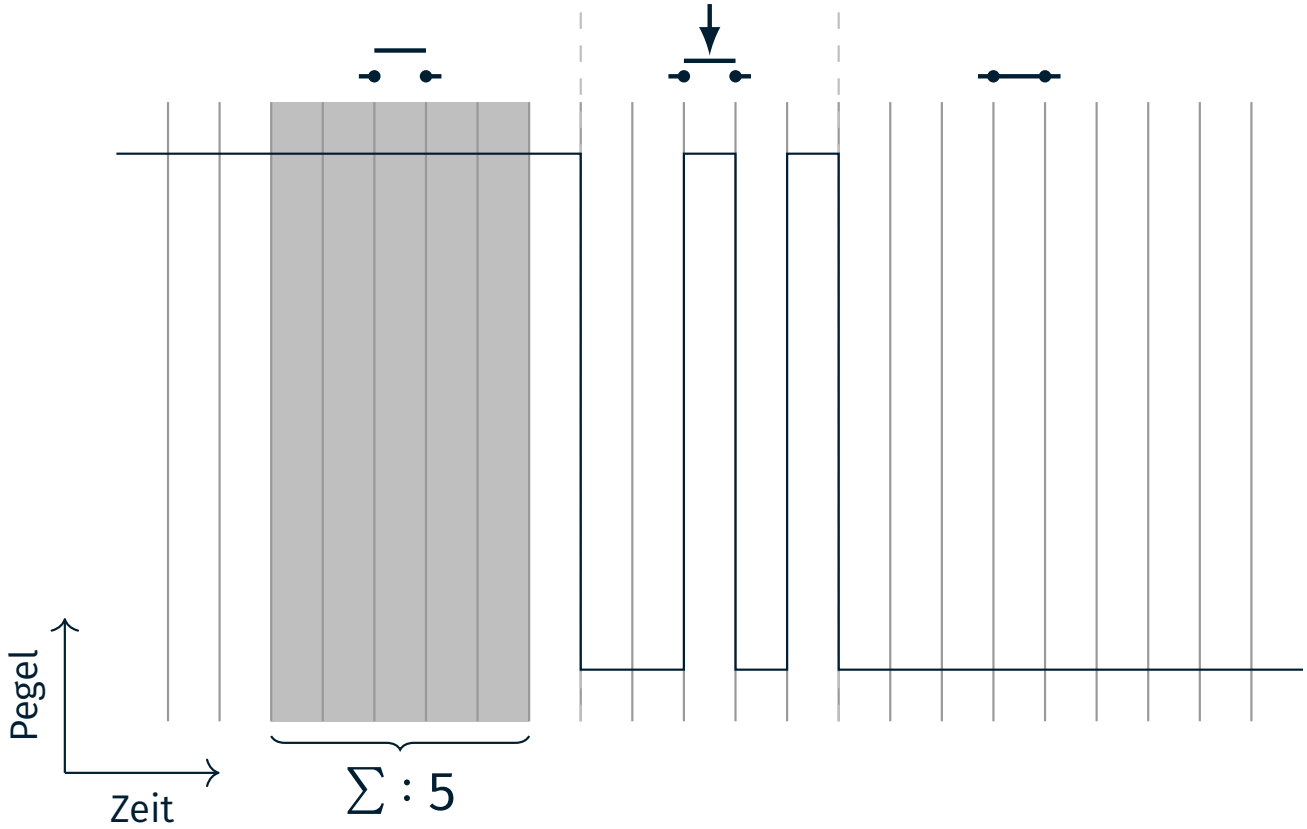
- Summe des Signalwertes über Zeitfenster
- Entscheidung für häufiger auftretenden Pegel
  - ⇒ Glättet die Kontaktprellen
- Aktives Warten mittels `_delay_us()` aus `<util/delay.h>`

Mittelwertfilter



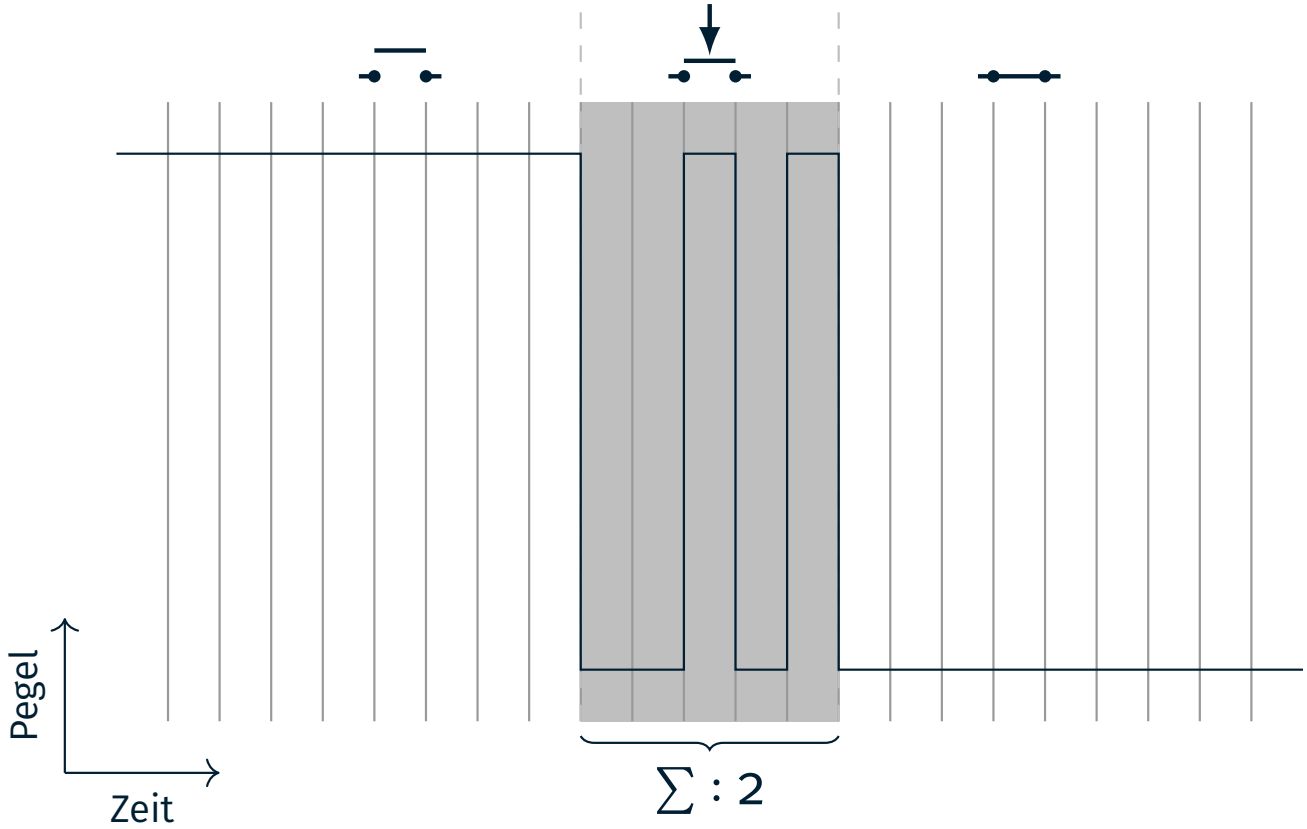
- Summe des Signalwertes über Zeitfenster
- Entscheidung für häufiger auftretenden Pegel
  - ⇒ Glättet die Kontaktprellen
- Aktives Warten mittels `_delay_us()` aus `<util/delay.h>`

Mittelwertfilter



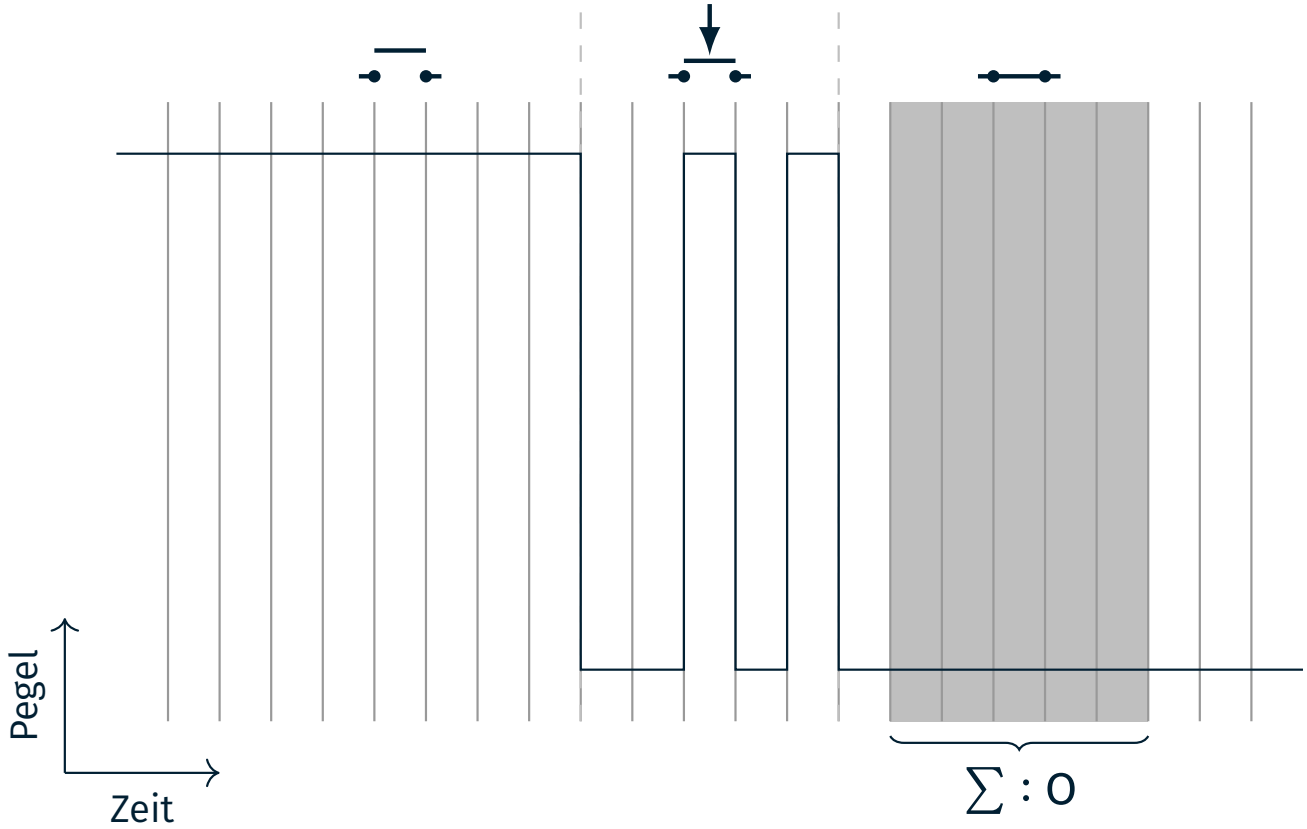
- Summe des Signalwertes über Zeitfenster
- Entscheidung für häufiger auftretenden Pegel
  - ⇒ Glättet die Kontaktprellen
- Aktives Warten mittels `_delay_us()` aus `<util/delay.h>`

Mittelwertfilter



- Summe des Signalwertes über Zeitfenster
- Entscheidung für häufiger auftretenden Pegel
  - ⇒ Glättet die Kontaktprellen
- Aktives Warten mittels `_delay_us()` aus `<util/delay.h>`

Mittelwertfilter



- Summe des Signalwertes über Zeitfenster
- Entscheidung für häufiger auftretenden Pegel
  - ⇒ Glättet die Kontaktprellen
- Aktives Warten mittels `_delay_us()` aus `<util/delay.h>`

Übersetzereinstellungen

- Projekt wie gehabt anlegen
  - Initiale Quelldatei: `test-button.c`
  - Dann weitere Quelldatei `button.c` hinzufügen
- Wenn nun übersetzt wird, werden die Funktionen aus dem eigenen BUTTON-Modul verwendet
- Andere Teile der Bibliothek werden nach Bedarf hinzugebunden
- Temporäres Deaktivieren zum Test der Originalfunktionen:

```
01 #if 0
02   ....
03 #endif
```

  ⇒ Sieht der Compiler diese “Kommentare”?

  ⇒ Wie kann der Code wieder einkommentiert werden?

Testen des Moduls

```
01 void main(void){
02   ...
03   // 1.) Callback Funktionen
04   int8_t result = sb_button_registerCallback(BUTTON0, ONPRESS,
       ↪ NULL);
05   if(result != -1) { // Nicht implementiert -> -1
06     // Test fehlgeschlagen
07     // Ausgabe z.B. auf 7-Segment-Anzeige
08   }
09
10   // 2.) Testen bei ungueltiger Button ID
11   ...
12 }
```

- Schnittstellenbeschreibung genau beachten
- Testen **aller möglichen Rückgabewerte**
- Fehler wenn Rückgabewert nicht der Spezifikation entspricht
- Für `sb_button_getState()`: Endlosschleife mit LEDs