
GSPiC-Aufgabe #5: Ampel

(15 Punkte, Abgabe bis Freitag, 22.07.2011, um 12:00, keine Gruppen)

Entwerfen Sie eine Steuerung für die Ampelanlage an einem Fußgängerüberweg in einer Datei `ampel.c`. Hierbei wird die den Autos zugewandte Ampel durch die LEDs `REDO`, `YELLOW0` und `GREEN0` dargestellt, die Fußgängerampel durch die LEDs `RED1` und `GREEN1` (kein gelbes Licht). Durch das Drücken von `BUTTON0` können Fußgänger eine Umschaltung anfordern. Die LED `BLUE1` signalisiert den Fußgängern, dass eine Umschaltanforderung entgegengenommen wurde.

Die Steuerung soll im Detail wie folgt arbeiten:

- Im Ausgangszustand zeigt die Autoampel grün, die Fußgängerampel rot.
- Eine Umschaltanforderung wird durch Druck auf `BUTTON0` ausgelöst. Der Druck der Taste wird durch Aktivierung der LED `BLUE1` (= Signal kommt) bestätigt. Diese LED wird beim Start der Umschaltung wieder deaktiviert. Weitere Tastendrucke werden bis zum Ende der Fußgänger-Grünphase ignoriert (nicht aktiv).
- Nach erfolgter Umschaltanforderung zählt die Ampel über die Siebensegmentanzeige 8 Sekunden herunter, welche die Fußgänger noch warten müssen, bis ihre Ampel grün wird; in den übrigen Phasen bleibt die Siebensegmentanzeige aus. Von den 8 Sekunden bleibt die Autoampel noch 5 Sekunden grün, dann bleibt sie 1 Sekunde im Zustand gelb, bevor sie rot wird. Erst nach weiteren 2 Sekunden, während welcher beide Ampeln rot sind, schaltet die Fußgängerampel auf grün.
- Die Grünphase der Fußgängerampel soll exakt fünf Sekunden andauern, bevor sie wieder auf rot wechselt. Anschließend wechselt die Autofahrerampel für 1 Sekunde auf gelb-rot und wieder auf grün in den Ausgangszustand.

Hinweise:

- Verwenden Sie die Module `LED` und `7SEG` der `libspicboard` für die Ausgabe, sowie das Modul `Timer` für die zeitkritischen Aktionen.
- Verwenden Sie *nicht* das Modul `Button` der `libspicboard`! Konfigurieren Sie stattdessen direkt die Interruptbehandlung und den -handler für `BUTTON0`; dieser ist angeschlossen am Pin `PD2` und an der externen Interruptquelle `INT0` des `ATmega32`.
- Achten Sie darauf, dass der Mikrocontroller in Ruhephasen, während welcher keine Berechnungen durchgeführt werden, in den Sleep-Modus wechselt. Dies geschieht entweder implizit, z. B. in `sb_timer_delay()` (siehe Online-Dokumentation zu dieser Funktion), oder explizit durch die entsprechenden Funktionen in `avr/sleep.h`.
- Falls Sie die Alarm-Callback-Funktionen des `Timer`-Moduls verwenden, beachten Sie, dass diese im Kontext einer Unterbrechungsbehandlungsfunktion (ISR) ausgeführt werden! Achten Sie deswegen darauf, diese Funktionen, genau wie die `Button`-ISR, sehr kurz zu halten. (In keinem Fall soll die komplette Ampelschaltung in diesen ISRs realisiert werden!)
- Achten Sie auf die korrekte Verwendung des `volatile`-Schlüsselworts. Beschreiben Sie in einem Kommentar zu jeder verwendeten `volatile`-Variable, weshalb Sie dieses Schlüsselwort dort benötigen. Im Übrigen soll Ihr Programm mit der Optimierungsstufe `-Os` kompiliert korrekt funktionieren (und damit mit der Build-Konfiguration des Projekt-Templates mit Optimierungen).
- Im Verzeichnis `/proj/i4gspic/pub/aufgabe5/` befindet sich die Datei `ampel.hex`, welche eine Beispielimplementierung enthält.