Praktikum 5 – Zustandsautomat

Dauer: 2 Lektionen

In diesem Praktikum werden Sie das Zusatzboard „Traffic lights“ verwenden. Es soll mithilfe eines Zustandsautomaten eine Strassenkreuzung geregelt werden.

Falls Sie für das RaspberryPi eine eigene SD-Karte verwenden, folgen Sie den Anweisungen aus PROGC_C-P5_Voraussetzungen.pdf, bevor Sie mit dem Praktikum beginnen können.

Zusatzboard

Auf dem Zusatzboard befindet sich eine T-Kreuzung mit drei Fussgängerstreifen. Um den Verkehrsfluss zu steuern, ist ein Lichtsignalsystem vorhanden. Dieses besteht aus drei Ampeln für den Strassenverkehr (Signale: rot, orange, grün, orange) und drei Ampeln für die Fussgängerstreifen (Signale: rot, orange, grün).

Es wurden insgesamt sechs Buttons angebracht. Drei davon sind für Fussgänger, die über den Fussgängerstreifen möchten (Button_W1, Button_W2, Button_W3). Die anderen drei signalisieren ein heranfahrendes Fahrzeug (Button_C1, Button_C2, Button_C3).

Die LEDs werden über Ports gesteuert. Es werden Ihnen Funktionen zur Verfügung gestellt, die Sie für die Kommunikation verwenden können.

unsigned char **progc_init**()

> progc_init initialisiert das Zusatzboard. Über eine I2C Schnittstelle wird ein Chip von NXP angesprochen, über dessen Ports die LEDs und Buttons verbunden sind.

unsigned char **progc_read**(unsigned char port_nr)

> Mit progc_read können Sie die Register der Ports auslesen. Wollen Sie zum Beispiel wissen, wie das Lichtsignal P3 zurzeit gestellt ist, müssen Sie Port 3 auslesen.

unsigned char **progc_write**(unsigned char port_nr, unsigned char value)

> Mit progc_write können Sie ein Port-Register beschreiben. Auf diese Weise können Sie die Lichtsignale an den Ports 0 – 2 steuern.

unsigned char **progc_set_7Seg_Display**(unsigned char value)

> Über progc_set_7Seg_Display können Sie auf der Siebensegmentanzeige einen hexadezimalen Wert zwischen 0 und F anzeigen lassen.

	Port 0	**Port 1**	**Port 2**	**Port 3**	**Port 4**
Bit 0	S1 red	S3 red	P2 red	Button P1	7seg: dp
Bit 1	S1 yellow	S3 yellow	P2 yellow	Button P2	7seg: g
Bit 2	S1 green	S3 green	P2 green	Button P3	7seg: f
Bit 3	S1 yellow	S3 yellow	P3 red	Button C1	7seg: e
Bit 4	S2 red	P1 red	P3 yellow	Button C2	7seg: d
Bit 5	S2 yellow	P1 yellow	P3 green	Button C3	7seg: c
Bit 6	S2 green	P1 green		Switch_4	7seg: b
Bit 7	S2 yellow			Switch_5	7seg: a

Aufgabe 1: LEDs ansteuern und Buttons verwenden

Es soll ein C-Programm erstellt werden, welches bei Klick auf Button_P2 das Lichtsignal P2 von rot über gelb auf grün und wieder zurückschaltet. Das Lichtsignal P3 soll dabei immer auf rot bleiben. Die restlichen Buttons und Ampeln können Sie bei dieser Teilaufgabe noch ignorieren.

Verwenden Sie die Funktion progc_write, um den entsprechenden Port 2 zu beschreiben.

Bit = 1 → LED leuchtet nicht.
Bit = 0 → LED leuchtet.

Wird ein Button gedrückt, so wird dies automatisch in die Variable ***interrupt_states*** geschrieben. Sie müssen somit nicht auf die Ports zugreifen. Überprüfen Sie über diese Variable, ob der Button_P2 gedrückt wurde. Falls ja (Bit_1 = 1), führen Sie das Umschalten der LEDs durch (rot-gelb-grün-gelb-rot). Danach muss das entsprechende Bit in der Variable interrupt_states manuell wieder auf 0 zurückgesetzt werden!

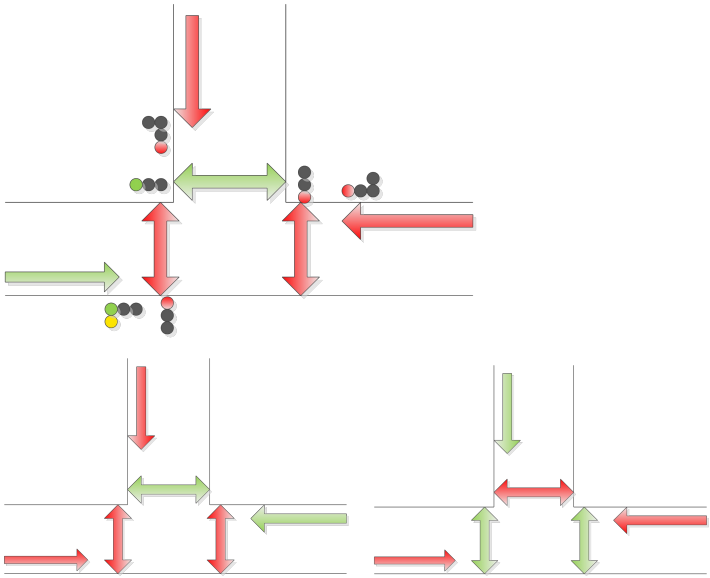
Verwenden Sie die Funktion ***void delay(unsigned int milliseconds)***, um für einen bestimmten Zeitraum zu warten. Dadurch können Sie zum Beispiel das Grün-Signal für fünf Sekunden leuchten lassen.

Verwenden Sie für diese Aufgabe das vorbereitete Grundgerüst **progc_*traffic_lights_aufgabe_1.c*.** Sie können auch auf Definitionen aus progc_crossroad.h zurückgreifen.

Aufgabe 2: Zustandsmaschine skizzieren

Im Digitaltechnik-Teil haben Sie den Moore-Automaten behandelt. Bevor Sie mit dem Programmieren beginnen, sollen Sie auf einem Blatt Papier einen solchen Automaten für die Lichtsignalsteuerung entwerfen.

In den Grafiken sehen Sie, welche Lichtsignale zusammen auf grün geschaltet werden sollen.



Hinweise:

- Als Inputs dienen die sechs Buttons (Cx und Wx).

Aufgabe 3: Zustandsmaschine entwickeln

Schreiben Sie ein C-Programm, welches mit Hilfe der in Aufgabe 2 entworfenen State Machine den Verkehrsfluss regelt. Verwenden Sie hierfür das Grundgerüst, welches Sie auf OLAT herunterladen können (**progc_*traffic_lights_aufgabe_1.c***).

Verwenden Sie die bereits bekannten Funktionen aus progc_crossroad.c sowie die folgende:

unsigned char **progc_set_traffic_light_state**(unsigned char light_identifier,unsigned char new_light_position)

> Mittels progc_set_traffic_light_state können Sie die Ampeln steuern, in dem Sie die gewünschte Ampel sowie die Farbe wählen. In Aufgabe 3 können Sie entweder diese zur Verfügung gestellte Funktion verwenden oder selbst eine entwickeln.

Hinweise:

- Zeigen Sie auf der Siebensegmentanzeige an, in welchem State Sie sich befinden.

Aufgabe 4: Zustandsmaschine erweitern

Erweitern Sie das System aus Aufgabe 3.

- Über den Switch können Sie zwischen Tag- und Nachtmodus wechseln. Implementieren Sie diesen zusätzlichen Nachtmodus, in welchem alle Verkehrsampeln gelb blinken sollen.
    - Benutzen Sie **progc_read**, um das Input Register an Port 3 auszulesen. Anhand des Zustands von Bit 6 können Sie bestimmen, ob es Tag oder Nacht ist.
- Die Länge, wie lange eine Ampel auf grün sein soll, soll nun variabel sein. Sie soll von der Anzahl Autos, die an der Kreuzung warten, abhängig sein.
- Denken Sie sich zusätzliche Erweiterungen aus und implementieren Sie diese.