Rollladensteuerung

Projektdokumentation

Für die Prüfung zum

Bachelor of Science

Des Studiengangs **Informatik**
Studienrichtung **Angewandte Informatik**

An der

Dualen Hochschule Baden-Württemberg Karlsruhe

Von

Leon Sauer (1543553)
Christopher Klammt (7849985)
Florian Jochem (3568778)

Abgabedatum	29. Juni 2018
Kurs	TINF16B2
Vorlesung	Systemnahe Programmierung
Dozent	Prof. Dr. Ralph Lausen

Erklärung

(gemäß §5(3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 29.09.2015)

Wir versichern hiermit, dass unsere Projektdokumentation mit dem Thema „Rollladensteuerung“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden.
Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

_____

Ort, Datum Unterschrift

Sperrvermerk

Der Inhalt dieser Dokumentation darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung der Ausbildungsstätte vorliegt.

Vorwort

Diese Projektdokumentation ist im Rahmen der Vorlesung **Systemnahe Programmierung** erarbeitet worden.

Die in der Dokumentation behandelten Themen, Konzepte sowie Implementierungen basieren auf den in der Vorlesung behandelten Inhalten.

Unter dem Kapitel „Implementierung“ werden die wesentlichen Problempunkte während der Implementierungsphase herausgegriffen und, soweit möglich, diskutiert und gelöst. Die komplette Lösung der Aufgabenstellung ist dem GitHub Repository zu entnehmen (Link s. Anhang).

Soweit möglich werden Bilder und Skizzen direkt in den Text eingebunden, um dem Leser so ein besseres Verständnis zu ermöglichen. Weitere Quelltextauszüge, Bilder und Skizzen finden sich im Anhang.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

n/A

Einleitung

Motivation

Ziel des Projekts ist es eine Rollladensteuerung zu entwickeln. Hierbei soll es möglich sein, zunächst mit Hilfe von 2 „Buttons“ die Rolllade hoch- oder runterfahren lassen zu können. Wünschenswert ist hier ein Pausieren zwischen Hoch- und Herunterfahren.
Erweitert werden soll die Anwendung um eine Automatik, sodass die Rolllade zu einer festgesetzten Uhrzeit automatisch hoch- und zu einer anderen Uhrzeit runterfährt.
Falls genug Zeit verbleibt soll zudem die Möglichkeit implementiert werden, die Uhrzeit, zu der die Rolllade sich automatisch bewegt, selber setzen zu können.

Aufgabenstellung

Aus der vorangegangenen Darstellung der Motivation ergeben sich drei große Aufgabenstellungen, welche wie folgt umrissen werden können.

1. Rollladensteuerung mit zwei Schaltern
   - 2 Schalter darstellen
   - Display für Anzeige, ob Rolllade offen oder geschlossen ist sowie Uhrzeit
   - Inputhandling
2. Erweiterung bestehender Anwendung um Automatik
   - Automatik durch Timer oder Zeitauslesung
3. Manipulation Automatik durch Endanwender
   - Werte für Automatik-Timer zugänglich machen und dynamisch gestalten

Diese Aufgaben sind entsprechend der Nummerierung anzugehen, um so eine möglichst angenehme Umsetzung des Projekts gewährleisten zu können.

Zur Umsetzung wird die Entwicklungsumgebung **MCU-8051 IDE** (s.S.8) verwendet.

Grundlagen

Assembler

Unter einem Assembler versteht man im Allgemeinen ein Computerprogramm, das Assemblersprache in Maschinensprache umwandelt.

Grundsätzliche Aufgaben eines Compilers belaufen sich also u.a. auf die Verwaltung von Adressen für Befehle oder Daten, die Verwaltung von Konstanten, die Umsetzung von Befehlsmnemonics sowie Datenmnemonics in deren binäre Repräsentation, das Ignorieren von Kommentaren, das Inkludieren weiterer Dateien, die Definition und Anwendung von Makros als auch das Übersetzen von Maschinencode, Objektdateien bzw. grundlegenden Übersetzungs-Listen.

Der 8051 Mikrocomputer

Der 8051 Mikrocontroller ist der Produktfamilie *MCS-51* von Intel zuzuordnen und ist trotz seiner Ersterscheinung in den 19080er Jahren bis heute einer der beliebtesten Mikrocontroller. Hierbei handelt es sich um einen 8-bit Controller in drei unterschiedlichen Größen: **Short**, **Standard**, **Extended**.

Während *Short* und *Standard* häufig als DIP zur Verfügung stehen und problemlos „Drop-In Kompatibilität“ unterstützen, basieren die *Extended*-Modelle auf einem anderen Formfaktor und liefern keine Unterstützung für „Drop-In“. Obwohl alle drei Modelle individuelle, spezielle Features liefern, teilen sie viele Eigenschaften und können zudem in 8051 Assembly „beschrieben“ werden.

Wesentliche Merkmale, die dem 8051 zu Popularität verholfen haben, sind:

- 128 nutzerdefinierte „Software Flags“
- 8-bit Datenbus, 16-bit Adressbus
- 16-bit Timer
- 3 interne sowie 2 externe Interrupts
- 4 8-bit Ports
- 128 bytes RAM

Mikrocontroller der Familie MCS-51 nehmen aktuellen Schätzungen zu Folge 50% des Einsatzes von Mikrocontrollern in Embedded-Systems ein und sind in einer breiten Palette von Anwendungsgebieten vorzufinden, so u.a. in der Kontroll-, Automobil- und Roboterindustrie.

Entwicklungsumgebung MCU-8051 IDE

Wie bereits in dem Kapitel *Einleitung* angemerkt, wird zur Umsetzung des Projekts die MCU-8051 IDE verwendet.

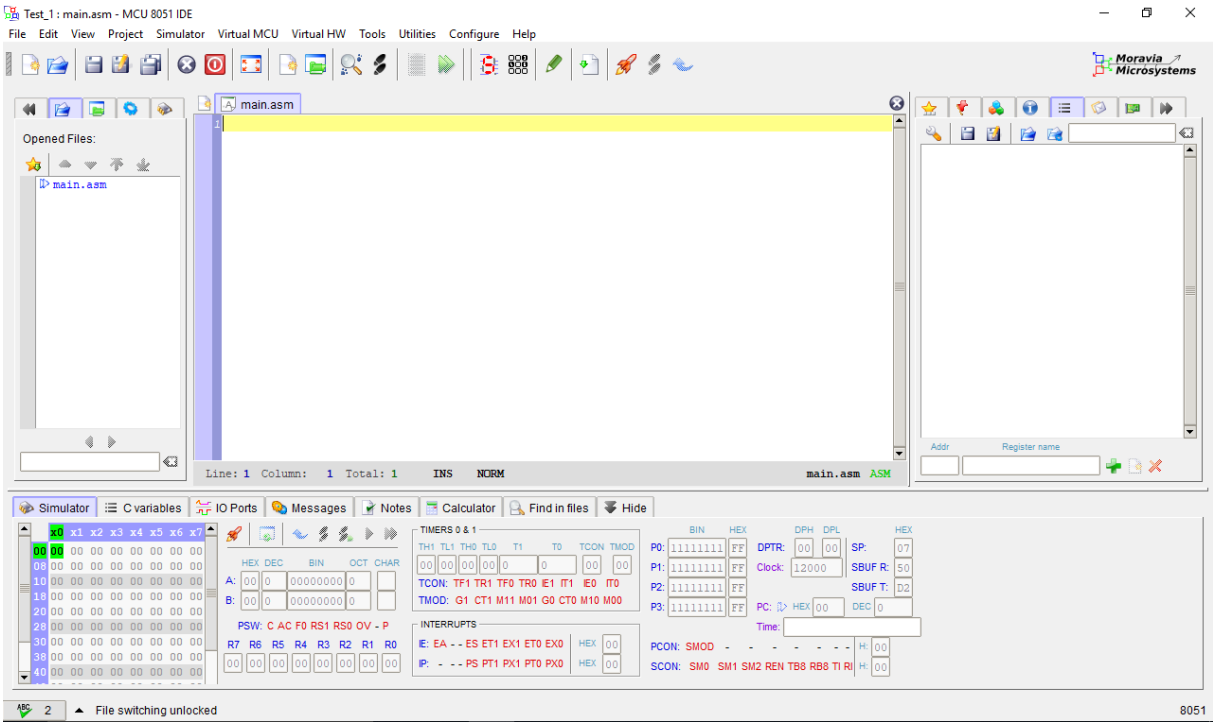

Abb.1: MCU-8051 IDE — Übersicht

Neben der Möglichkeit speziell für den 8051 Mikrocontroller Programme zu schreiben, liefert die Entwicklungsumgebung zusätzlich die Möglichkeit Hardware zu simulieren und stellt dem Entwickler so u.a. verschiedene Displays zur Verfügung.

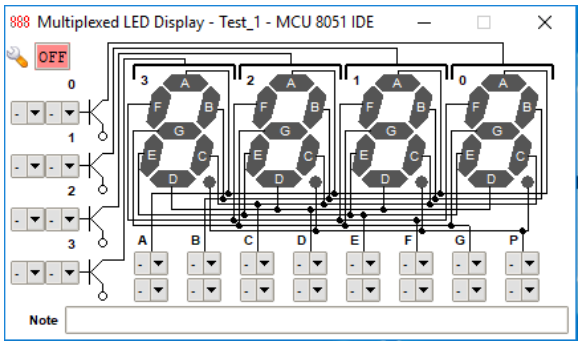

Abb.2: MCU-8051 IDE — Multiplex LED

Konzept

Programmentwurf

- Pinbelegung

Abb.3: Programmentwurf — Pinbelegung

Zusätzlich zu der obigen Pinbelegung kommt noch die Pinbelegung für das Darstellen der Uhr (P1 & P2) sowie die Nutzung von 3 Pins auf P3 zur Darstellung des aktuellen Status.

- Programmablauf

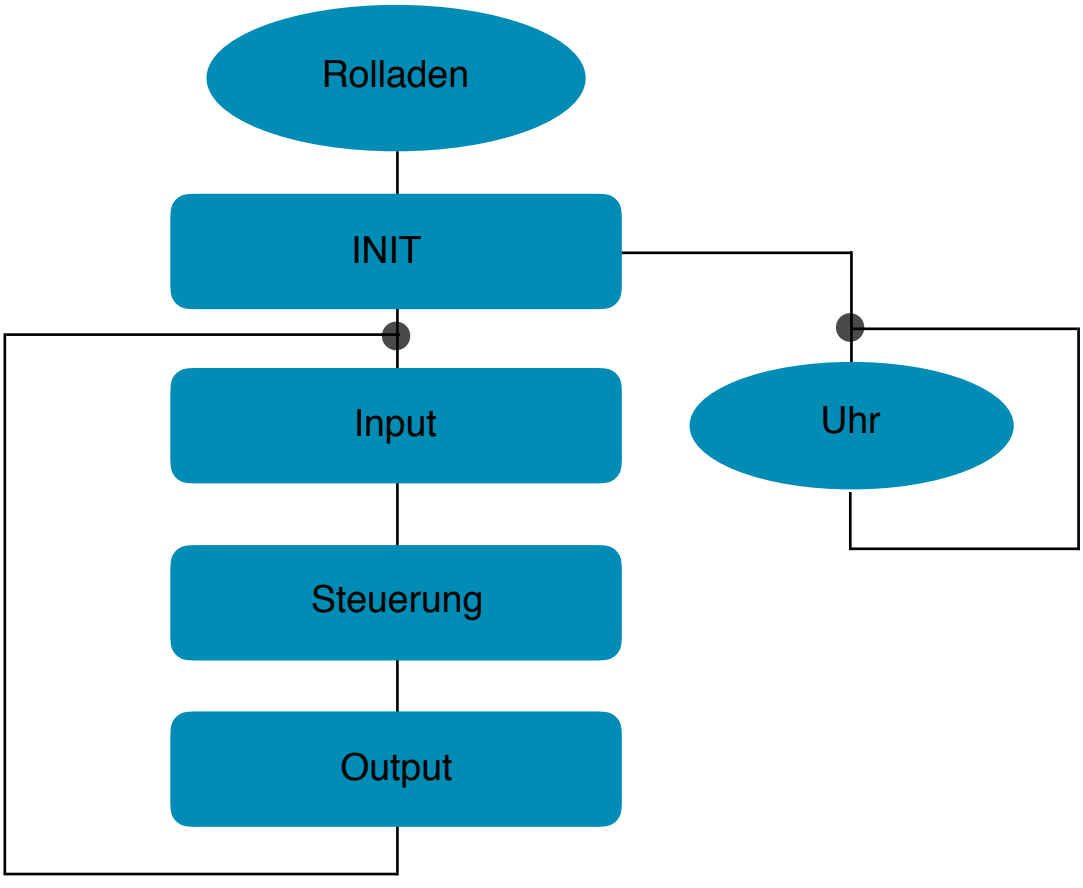

Abb.4: Programmentwurf — Programmablauf

- Button (1): *Fahre hoch*

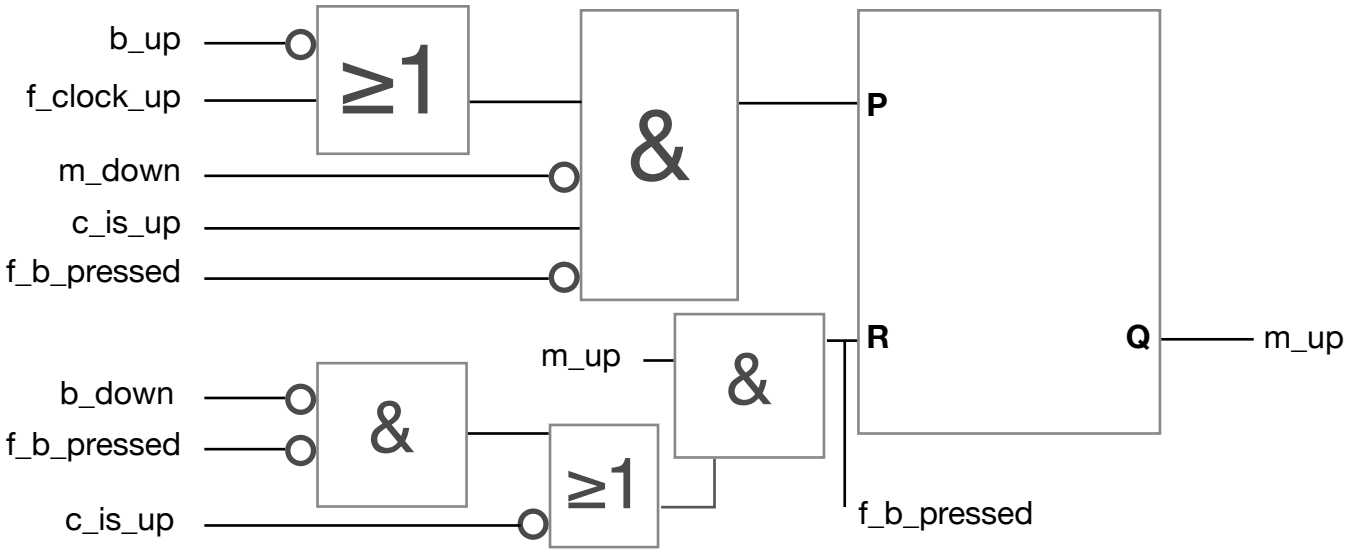

Abb.5: Programmentwurf — Button(1)

- Button (2): *Fahre herunter*

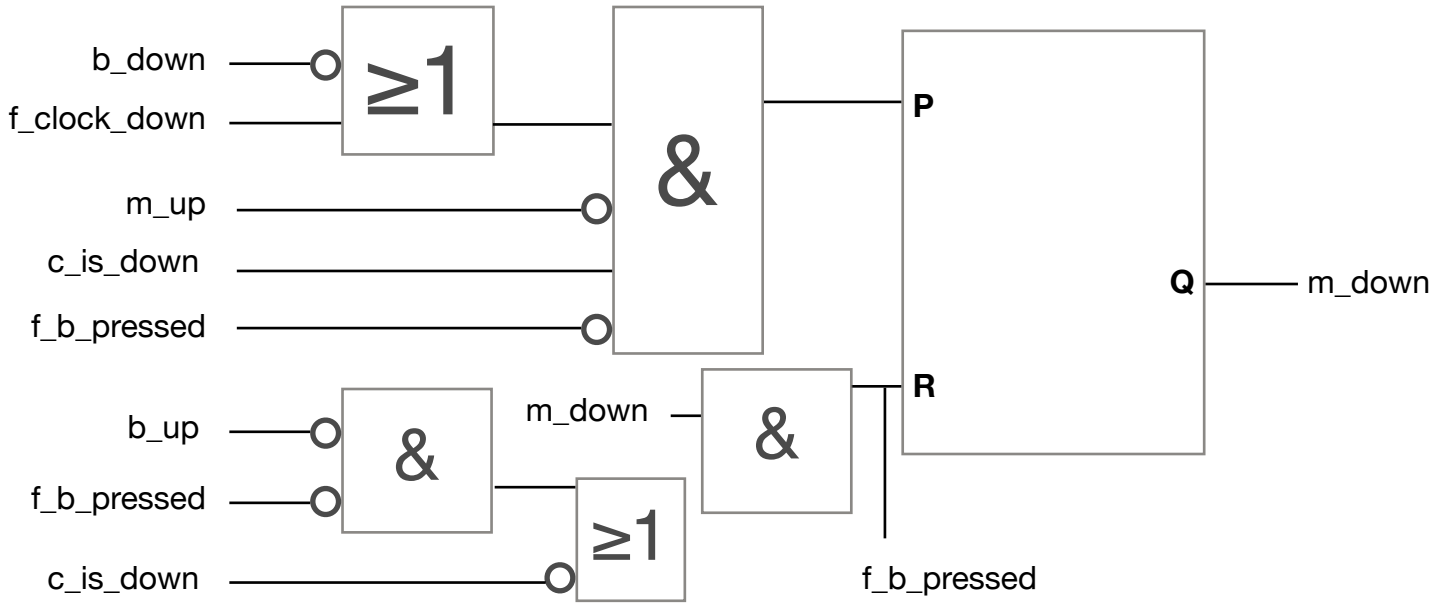

Abb.6: Programmentwurf — Button(2)

Implementierung

Im Rahmen der Implementierung wurde zur finalen Abgabe der Stand der zweiten Aufgabenstellung erreicht - die Umsetzung einer Uhr, mit der Möglichkeit unter Verwendung zweier Buttons die Rolllade hoch- bzw. herunterfahren zu lassen und die bei einer festgelegten Zeit zusätzlich automatisch hoch- / herunterfährt.

Uhr

In diesem Sinne bestand das erste Ziel darin, eine **Uhr-Logik** zu implementieren. Hierbei wurde sich auch ein 24 Stunden Muster geeinigt. Daraus schlussfolgernd galt es einige Kriterien bei der Implementierung zu berücksichtigen. Die Uhr als solche soll einen *Minuten-Wert* von 0 bis 59 anzeigen können. Bei dem Erreichen von 60 Minuten, muss die Minutenanzeige auf „00“ überschlagen und zugleich die Stundenanzahl um den Wert 1 erhöhen. Die *Stundenanzeige* hingegen soll einen Wertebereich von 0 bis 23 abdecken. Erreicht die Uhr den Zustand „23:59“ und zählt um einen Minuten-Wert weiter, so ist die Anzeige auf „00:00“ umzustellen.
Die eben geschilderten Anforderungen wurden unter Verwendung eines Timers erreicht, welcher in definierten zeitlichen Abständen den Minutenwert erhöht und anschließend das Multiplex-Display aktualisiert (s. Abb. 3, 4).

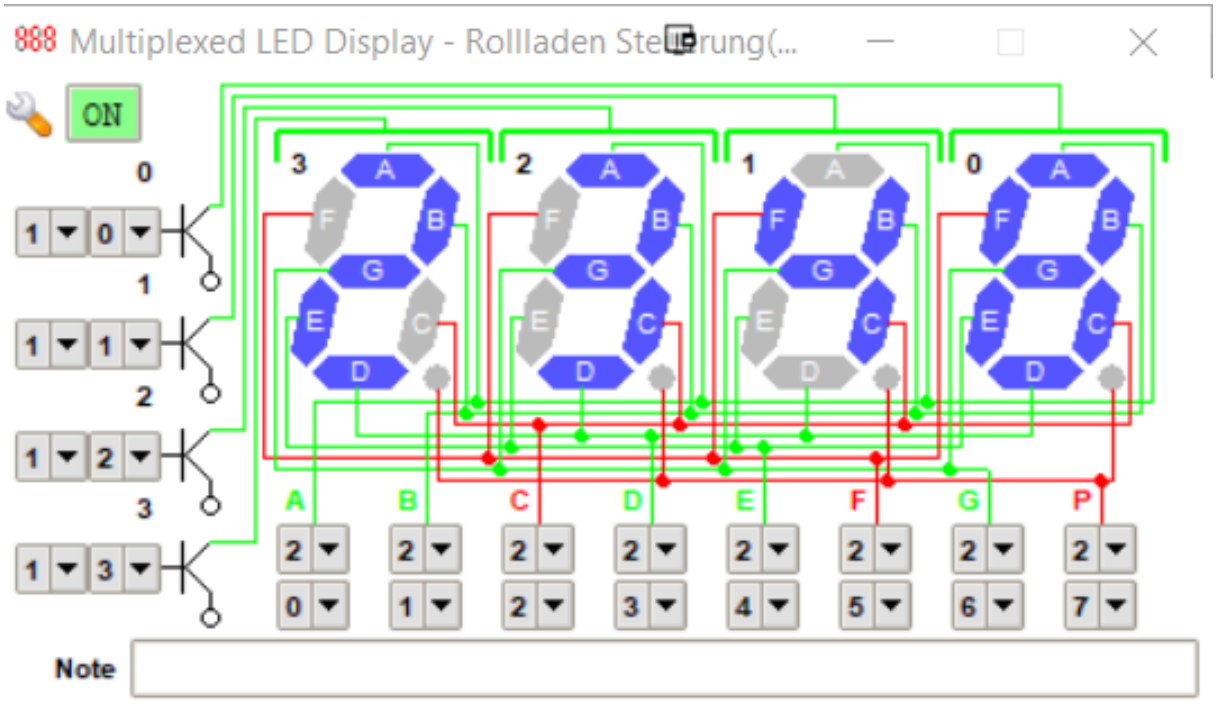

Abb.7: Multiplex-Display — Uhrzeit 1, vor 00.00

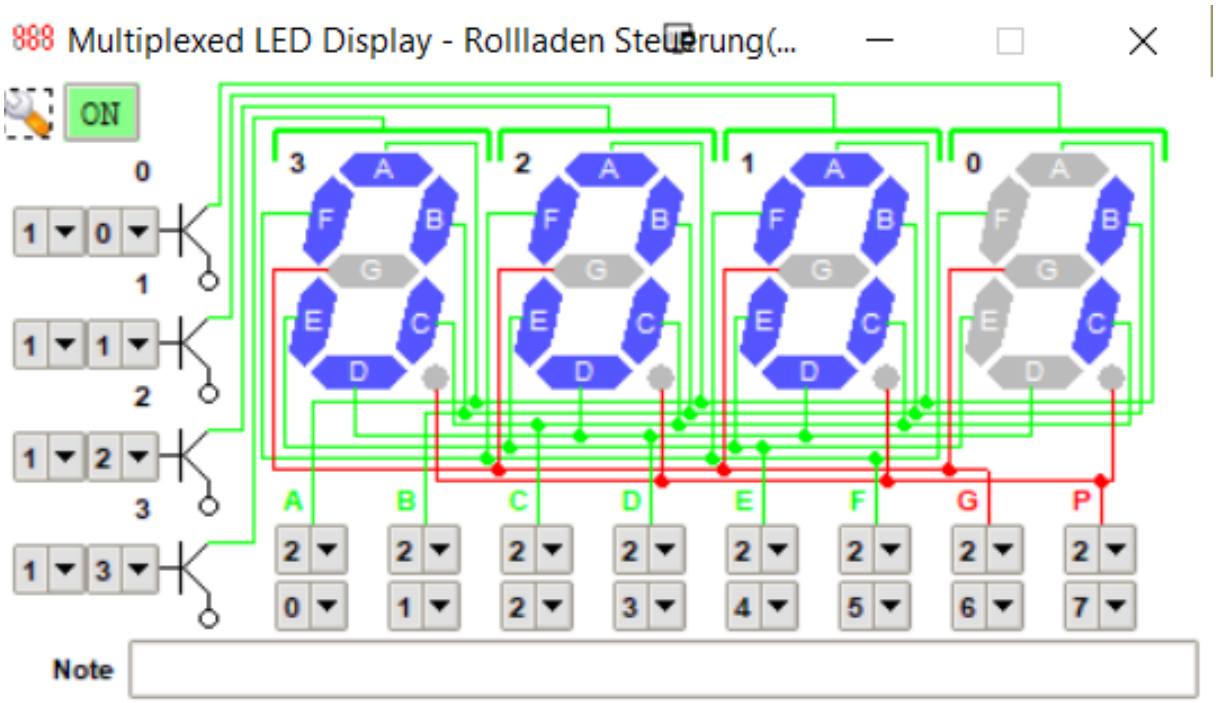

Abb.8: Multiplex-Display — Uhrzeit 2, nach 00.00

Genauer gesagt „zählt“ der Timer eine Minute und fügt daraufhin das Aktualisieren des Displays durch. Diese Logik ist im folgenden Code-Ausschnitt ansatzweise dargestellt.

```
 ; timer
timer:
inc r1
cjne r1, #09h, nuranzeige
mov r1, #00h
call countdown
ret

nuranzeige:
call zeigen
ret

countdown:
cjne r6, #3bh, minuten
cjne r7, #17h, stunden
mov r6, #00h
mov r7, #00h
jmp zeigen

stunden:
mov r6, #00h
inc r7
call zeigen
ret
minuten:
inc r6
call zeigen
 ret
```

Wie dem Code-Ausschnitt zu entnehmen ist, kommt es zum Aufruf der Funktion „zeigen“. Diese aktualisiert das Programm bzgl. der Minuten und Stunden (s.U.).

```
zeigen:
mov DPTR, #table
mov a, R6
mov b, #0ah
div ab
mov R0, a
movc a, @a+dptr
mov r3, a
mov a, r0
xch a, b
movc a, @a+dptr
mov r2, a

;————————————————-

mov a , R7
mov b, #0ah
div ab
mov R0, a
movc a, @a+dptr
mov r5, a
mov a, r0
xch a, b
movc a, @a+dptr
mov r4, a
call display
ret
```

Dem obigen Abschnitt zu entnehmen, wird ebenso nach erfolgten Änderungen auf. das Display verwiesen, welches nun noch neu geladen werden muss (s.U.).

```
display:
mov P2, R2
clr P1.0
setb P1.0

mov P2, R3
clr P1.1
setb P1.1

mov P2, R4
clr P1.2
setb P1.2

mov P2, R5
clr P1.3
setb P1.3

ret

org 300h
table:
db 11000000b
db 11111001b, 10100100b, 10110000b
db 10011001b, 10010010b, 10000010b
db 11111000b, 10000000b, 10010000b

end
```

Aufgrund der hinzukommenden Programmfunktionalitäten (Button-Input) ist anzumerken, dass dieser Timer als *Unterprogramm* gehalten wird.

Logik

```
 anfang:

mov EIN, P0

CLR C
ORL C, /B_UP
ORL C, F_CLOCK_UP

ANL C, /F_B_PRESSED
ANL C, /M_DOWN
ANL C, C_IS_UP

JNC S1
SETB M_UP
CLR F_CLOCK_UP

S1:
SETB C
ANL C, /B_DOWN
ANL C, /F_B_PRESSED

ORL C, /C_IS_UP

ANL C, M_UP

JNC S2
CLR M_UP
SETB F_B_PRESSED

S2:
CLR C
ORL C, /B_DOWN
ORL C, F_CLOCK_DOWN

ANL C, /F_B_PRESSED
ANL C, /M_UP
ANL C, C_IS_DOWN

JNC S3
SETB M_DOWN
CLR F_CLOCK_DOWN

S3:
SETB C
ANL C, /B_UP
ANL C, /F_B_PRESSED

ORL C, /C_IS_DOWN

ANL C, M_DOWN

JNC S4
CLR M_DOWN
SETB F_B_PRESSED

S4:
CLR C
ORL C, /B_UP
ORL C, /B_DOWN
MOV F_B_PRESSED, C

jnb tr0, da
ajmp anfang
da:
call display
ajmp anfang
```

Obenstehend ist das Hauptprogramm zu finden. Dieses überprüft anhand dem zuvor aufgezeigtem Ablaufplan die Eingänge (Buttons und Kontakte) und steuert dann den Rollladen-Motor (Hoch oder Runter).
Weiterhin wird zu einer bestimmten Uhrzeit (fest im Programmcode hinterlegt) der Rollladen hoch- bzw. Runtergefahren. Dazu werden Flags gesetzt (s. Anhang).

Manipulation Automation

Im Rahmen der Aufgabenstellung 3 wäre die Bearbeitung des Nutzers für das Setzen der automatischen Fahrzeit der Rolllade gesetzt. Aufgrund des begrenzten Zeitraums, in dem produktiv an dem Projekt gearbeitet werden konnte, war es uns leider nicht möglich, dies umzusetzen.

Zusammenfassung und Probleme

Vor Benutzung des f_b_pressed flags lief das Programm nicht wie gewollt; bei Gedrückthalten eines Buttons zum Anhalten des Motors wurde die Richtung des Motors nämlich umgekehrt, anstatt einfach auf einen zweiten Knopfdruck zu warten. Durch Setzen des eben genannten Flags, konnte dies jedoch behoben werden.
Weitere nennenswerte Probleme traten nicht auf.

Zusammenfassend lässt sich sagen, dass die Bearbeitung des Projekts ideal die Inhalte der Vorlesung praktisch behandelt und somit einen guten Einblick in die Arbeit mit Mikrocontrollern gegeben hat. Die praktische Bearbeitung eines eigenen Projekts mit den vorgestellten Methodiken hat sowohl das Interesse als auch die Mitarbeit sehr motiviert.

Anhang

Sämtliche Code-Dateien sowie dieses Dokument finden sich auf folgender GitHub-Seite: GitHub Repository