

**Laborprotokoll**

**IndInf02: Ampelsteuerung**

**Systemtechnik Labor**

**5BHIT 2015/16, Gruppe z**

**Michael Weinberger**

|  |  |
| --- | --- |
|  | **Version 1.0** |
| **Note:** | **Begonnen am 13. November 2015** |
| **Betreuer: Weiser** | **Beendet am 13. November 2015** |

Inhaltsverzeichnis

[1Einführung 3](#__RefHeading___Toc105_1963609346)

[1.1Ziele 3](#__RefHeading___Toc107_1963609346)

[1.2Voraussetzungen 3](#__RefHeading___Toc109_1963609346)

[1.3Aufgabenstellung 3](#__RefHeading___Toc111_1963609346)

[2Ergebnisse 4](#__RefHeading___Toc113_1963609346)

# Einführung

## Ziele

Mithilfe der vorgestellten State Machines soll eine Ampelsteuerung implementiert werden.

## Voraussetzungen

* STM32F3 Discovery-Mikrocontroller
* USB-Kabel
* OpenSTM32-Workbench
* Einarbeitung in Doxygen
* Grundwissen in C

## Aufgabenstellung

Installiere die auf Eclipse-CDT basierende Workbench von STMicroelectronics für die Familie von STM32-Boards. Implementiere auf unserem Testboard eine Ampel. Verwende hierbei eine beliebige State Machine.

# Ergebnisse

In der heruntergeladenen Workbench wird ein neues C-Projekt angelegt, mit folgenden Schritten:

* Projektnamen vergeben
* Toolchain ‘Ac6 STM32 MCU GCC’ wählen
* 2x Next
* Unter Series ‘STM32F3’ & Board ‘STM32F3DISCOVERY’ wählen + Next
* Bei Firmware wird der Hardware Abstraction Layer (Cube HAL) ausgewählt
* Dessen Firmware wird anschließend per Knopfdruck heruntergeladen  
  (im Schulnetz zur Laborzeit hat dies viel Zeit in Anspruch genommen)
* Des Weiteren empfiehlt es sich die Option ‚As sources in the application project‘ auszuwählen, da es in meiner Umgebung mit anderen Einstellungen zu Fehlern gekommen ist.

## Code-Struktur bzw. Idee dahinter

Zu allererst habe ich Methoden zum Togglen der drei von uns benötigten LEDs erstellt und im Header-File definiert, und diese per Delay beliebig lang leuchten lassen bzw. wieder abgeschaltet. **– control.c & control.h**

Im File **ampel.h** habe ich nach einer Empfehlung über das konkrete Vorgehen seitens des Professors alle States & Events festgelegt. In der Datei **ampelsteuerung.c** ist der Kern dieses Programms untergebracht: Die Implementierung der State Machine. Ich verstehe die State Machine als Quasi-Design Pattern, die Sinn macht und sehr einfach zu implementieren ist.

Ein kurzer Einblick in den Code:

|  |
| --- |
| **case** FAHREN**:**  **if** **(**repr**->**zustand **==** GRUEN**)** **{**  led\_gruen**();**  repr**->**zustand **=** GRUEN\_BLINKEN**;**  repr**->**event **=** VORBEREITUNG\_HALT**;**  **}** **else** **{**  led\_off**();**  repr**->**event **=** FALSE**;**  **}**  **break;** |

In Worten: Wenn die Ampel grün leuchtet, wird der Übergang zu Gelb, also *grün blinken* ausgelöst, ansonsten tritt ein Fehler auf.

In der Hauptklasse **main.c** wird lediglich die Funktion mit den Ampelparametern versorgt aufgerufen und in einer Endlosschleife ausgeführt.