Embedded Systems

Prof. Dr.-Ing. Michael Uelschen
Prof. Dr.-Ing. Jürgen Wübbelmann
Hochschule Osnabrück
Laborbereich Technische Informatik
m.uelschen@hs-osnabrueck.de
j.wuebbelmann@hs-osnabrueck.de

Versuch 5: Regelung eines Motors

Einleitung

Ziel des Praktikums ist es, die Drehzahl eines Motors zu regeln. Dazu wird ein handelsüblicher CPU Prozessorlüfter (4-Pin Anschluss) mittels Pulsweitenmodulation (PWM) vom STM32-Board angesteuert und die Drehzahl des Lüfters über die Capture-Compare-Einheit gemessen. Über einen digitalen Regler wird die Drehzahl auf einen Sollwert geregelt.

Die folgenden Lernziele stehen bei dieser Aufgabe im Vordergrund:

- Anwenden der Peripherieeinheiten PWM und Capture/Compare des STM32L476
- Aufbau einer Applikation aus mehreren Tasks als Synthese der in den vorherigen Versuchen gelernten Prinzipien:
  - Regler Task
  - Task zur Erfassung von Ist- und Sollwerten.
  - Task zur Kommunikation mit dem Benutzer (Visualisierung und Parametrierung der Regelung)

Aufgabenstellung

Erstellen Sie die folgenden Tasks (mit `osThreadCreate` (CMSIS) oder der OOP-Kapselung):

- **ReglerTask**: Diese Task ruft periodisch alle 10 Millisekunden einen Regler auf, der die neue Stellgröße berechnet. Die Task setzt die neue Stellgröße.
  Zur Überprüfung der Laufzeit mittels Oszilloskop (PicoScope) der Task soll die Task vor der Berechnung der Stellgröße den Punkt der Siebensegmentanzeige (siehe Versuch 2) setzen, nach setzen der Stellgröße wird die Siebensegmentanzeige wieder ausgeschaltet.
- **MeasurementTask**: Diese Task bestimmt den Soll- und Istwert der Motordrehzahl. Der Sollwert wird zyklisch durch Abfragen eines Potentiometers bestimmt. Der Istwert der Drehzahl wird aus dem Tachogebersignal des Motors abgeleitet.
- **MenuTask:** Diese Task schreibt Ist- und Sollwert der Drehzahl auf das OLED. Über den Joystick soll eine Menüsteuerung die Konfiguration der Regelparameter ermöglichen.

Zur Drehzahlmessung soll der Capture und der Compare Interrupt genutzt werden. Aus den Interrupt Service Routinen sollen die Tasks signalisiert werden, ausschließlich in den Tasks erfolgen die Berechnungen.

Die Regelung und die Drehzahlmessung sollen ausschließlich in Integer-Arithmetik erfolgen. Fließkommaberechnungen dürfen nur in der Initialisierungsphase des Systems eingesetzt werden.

Hinweise

Hardware

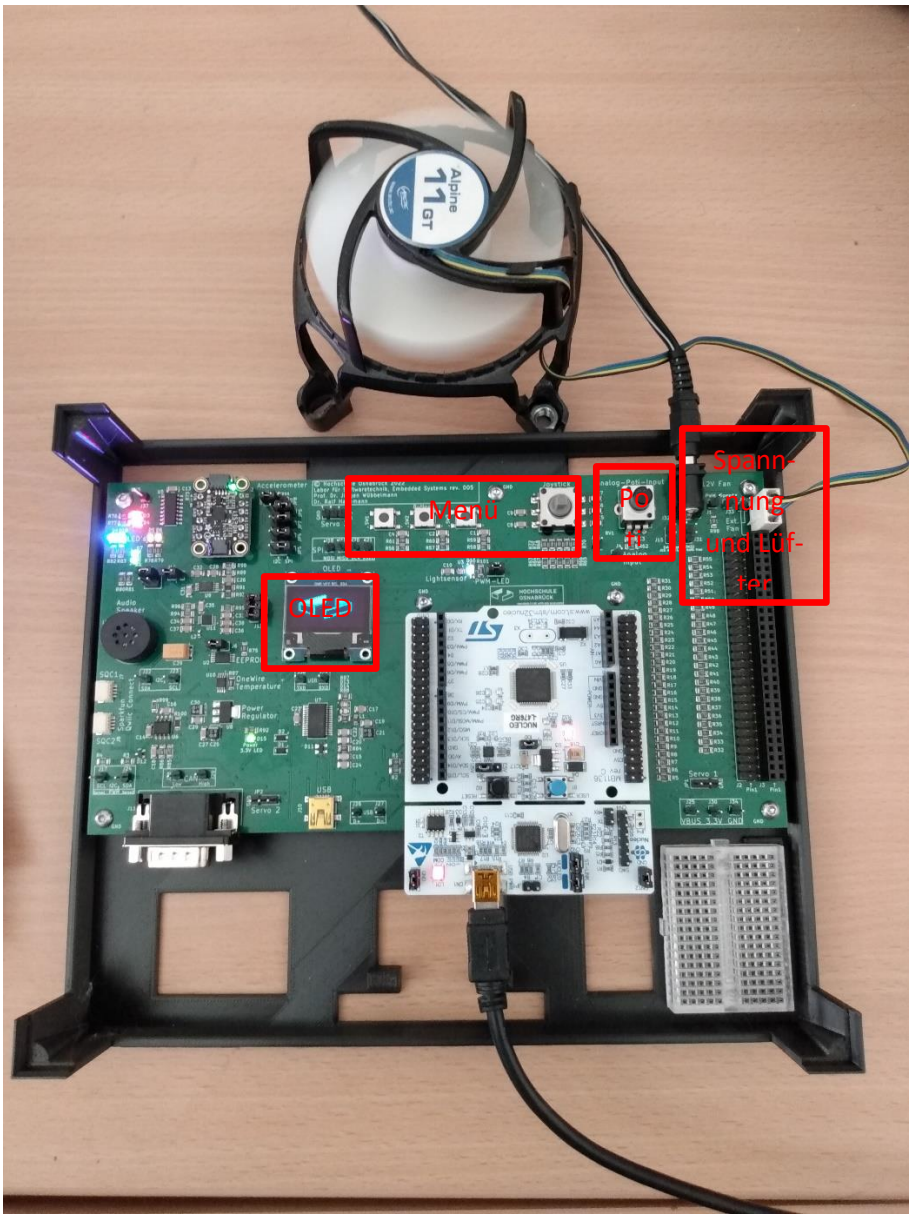

Abbildung 1: Physikalischer Aufbau

Abbildung 1 zeigt den physikalischen Versuchsaufbau. In die Basisplatine wird ein ein Motor und eine externe 12V Spannungsversorgung eingesteckt. Der Standard-CPU-Lüfter besitzt einen 4 poligen Stecker, an dem neben 12 V Versorgungsspannung und Masse auch das PWM Signal eingespeist werden kann und das Tachosignal des Motors abgegriffen wird.

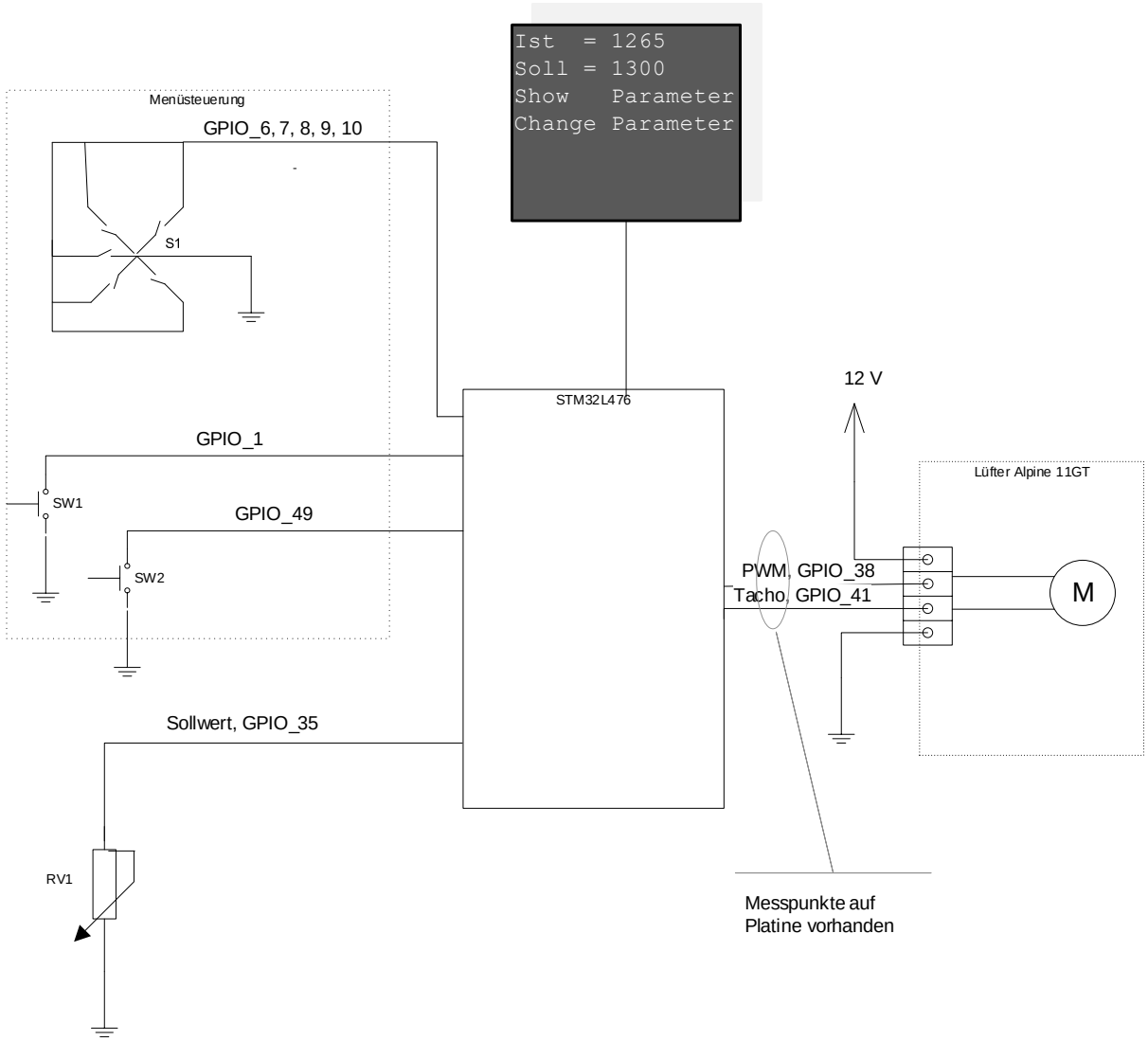

Abbildung 2: Blockschaltbild des Aufbaus

Dem Blockschaltbild Abbildung 2 ist die Verwendung der Prozessorpins zu entnehmen. Das Einlesen des Sollwertes erfolgt über den internen Analog Digital Umsetzer. Der Joystick S1 dient zur Menüsteuerung. Die Pins sollen jeweils einen Interrupt auslösen, in den zugehörigen ISR wird das entsprechende Ereignis der `MenuTask` signalisiert. Die Taster SW1 und SW2 können genutzt werden, um die Eingabewerte in den Einer- Zehner- Hunderter oder Tausenderstelle zu inkrementieren bzw. dekrementieren (bei Bedarf können die Multiplikationsfaktoren angepasst werden).

Das PWM Signal wird über GPIO_38 ausgegeben. Wählen Sie die PWM Grundfrequenz zu 10 kHz. Das Tastverhältnis darf zwischen 20-100 % liegen.
Der Lüfter liefert ein Tachosignal. Dieses ist eine Rechteckspannung mit 2 Impulsen pro Motorumdrehung. Dieses wird von GPIO_41 mit der Capture Funktion ausgewertet. Die Drehzahl 0 muss innerhalb einer Sekunde erkannt werden. Ein Capture Event soll einen Interrupt auslösen, der dann der `MeasurementTask` ein Signal sendet. Die Drehzahlberechnung erfolgt innerhalb der Task.

Tabelle 1: Nutzung Prozessorpins

Funktionsgruppe	GPIO	Pin	Verwendung	Pin Funktion
Regelung Eingabe	GPIO_35	PC3	Sollwert, ADC Eingang	ADC1_IN4
	GPIO_41	PC9	Istwert, Capture Eingang	TIM8_CH4
Regelung Ausgabe	GPIO_38	PC6	Stellwert, PWM Ausgang	TIM3_CH1
Menüsteuerung	GPIO_7	PA7	Joystick, Press	GPIO_EXTI7
	GPIO_10	PA10	Joystick, Down	GPIO_EXTI10
	GPIO_9	PA9	Joystick, Right	GPIO_EXTI9
	GPIO_8	PA8	Joystick, Up	GPIO_EXTI8
	GPIO_6	PA6	Joystick, Left	GPIO_EXT6
	GPIO_1	PA1	MULT10	GPIO_Input
	GPIO_49	PH0	MULT100	GPIO_Input

Grundlagen der Pulsweitenmodulation (PWM)

Die Ansteuerung des Motors erfolgt über eine Pulsweitenmodulation. Dazu besitzt der STM32L476 mehrere PWM Modulatoren in seiner Hardware (siehe[3], Kapitel 31.3.9). Ein Zähler (hier: Timer 3, TIM3) zählt dazu bis zu dem Wert eines Auto Reload Registers (hier:TIM3_ ARR) hoch und wird bei Erreichen dieses Wertes wieder auf 0 gesetzt. Dadurch wird die Grundfrequenz der PWM festgelegt. Ein Capture/Compare Register (hier: TIM3_CCR1) ist mit einem Prozessorpin (hier: PC6) gekoppelt: Der Pin wird bei Nulldurchgang des Timers auf high Pegel gesetzt, bei Erreichen des Capture/Compare Registers wieder auf low Pegel. Über den Wert des Capture/Compare Registers kann also die Pulsweite (Tastverhältnis, Duty Cycle) und damit der Mittelwert der Motorspannung und die Drehzahl eingestellt werden. Voraussetzung ist, dass Timer 3, Channel 1 als PWM Channel in der CUbeMX konfiguriert ist und PC6 diesem Channel zugewiesen wird.

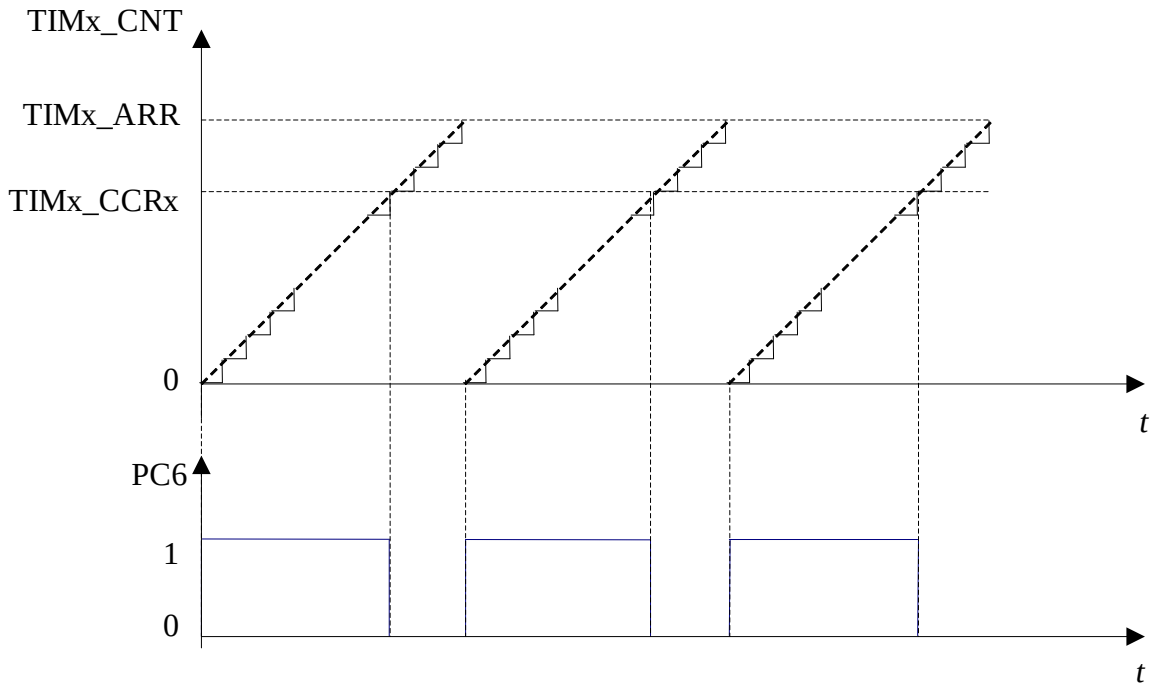

Abbildung 3: Prinzip der PWM

Motorkenndaten

Im Folgenden sind einige Kenndaten des Antriebes gegeben:

Um auch bei niedrigen Tastverhältnissen des PWM Signals ein sicheres Anlaufen zu gewährleisten, regelt die interne Lüfterelektronik die Drehzahl bei zu kleinen Tastverhältnissen auf den Maximalwert. Die Drehzahlregelung erlaubt somit keine niedrigen Drehzahlen. Im Versuch soll die Drehzahl zwischen 750… 2000 min^{-1} geregelt werden, das Tastverhältnis soll 20 % nicht unterschreiten.

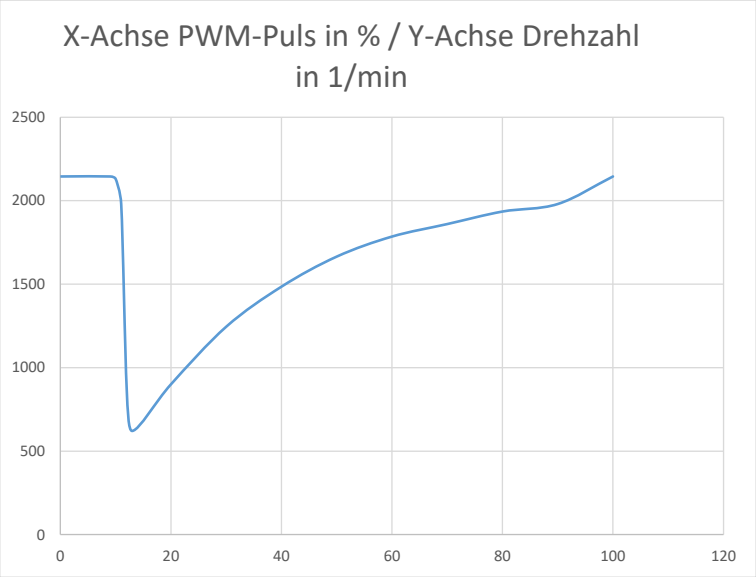

Abbildung 4: Tastverhältnis Drehmoment Kennlinie

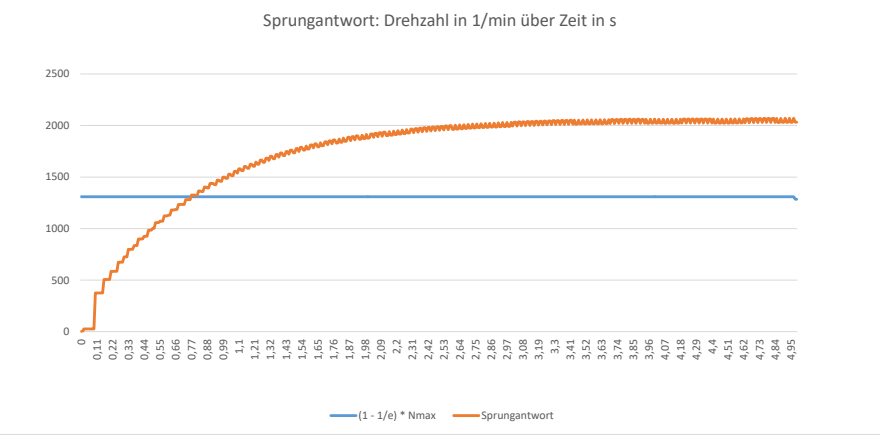

Abbildung 5: Sprungantwort des Antriebs

Für die Regelparameter kann die Zeitkonstante des Antriebs (näherungsweise als PT1-Glied aufzufassen) aus der Sprungantwort bestimmt werden. Gemäß Abbildung 5 beträgt diese ca. 0,77 s.

Software

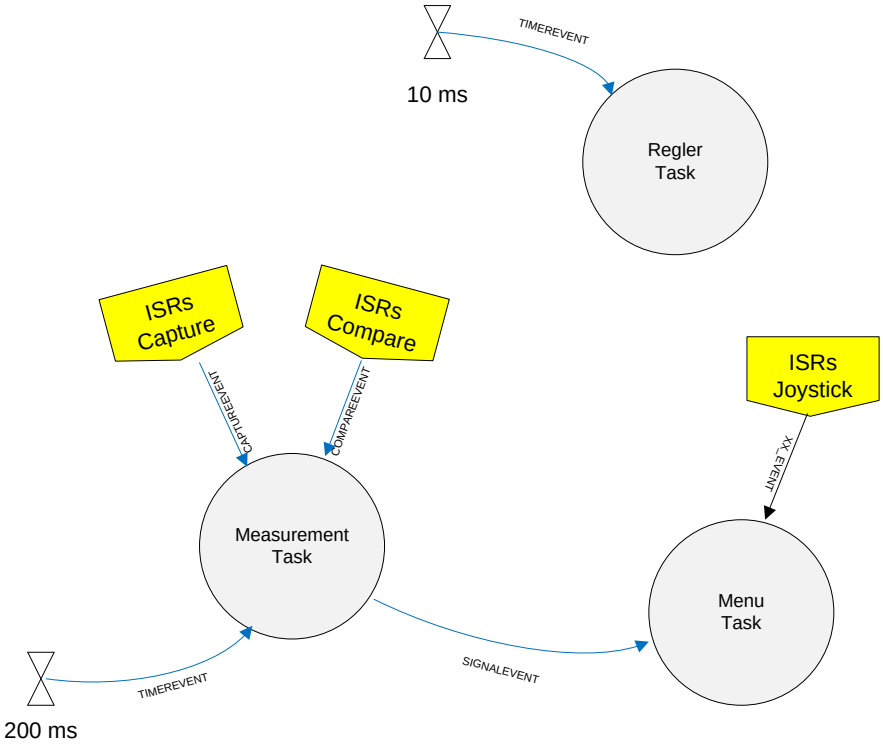

Abbildung 6: Zusammenspiel der Tasks

Die Software soll aus drei Tasks bestehen:

- ReglerTask: Die ReglerTask berechnet alle 10 ms den neuen Stellwert. Dazu benötigt sie die Ist- und Sollwerte von der MeasurementTask. Der neue Stellwert wird mittels PWM ausgegeben.
  Zur Laufzeitverifikation steuert die Task den Punkt der Siebensegmentanzeige an.
- MeasurementTask: Die Task wird durch einen Timer Interrupt in den Zustand bereit versetzt. Je nach Ereignis (Capture oder Compare) sendet die ISR ein unterschiedliches Event und die Task berechnet die neue Drehzahl. Achtung: keinerlei Programmlogik in ISR!
  Alle 200 ms bestimmt die Task den neuen Sollwert und sendet anschließend ein Event an die MenuTask.
- MenuTask: Die MenuTask wird durch das von der MeasurementTask gesendete Event in den Zustand bereit versetzt und gibt die aktuellen Soll- und Istwerte über das OLED aus.
  Ein Interrupt vom Joystick (5 Taster, also 5 mögliche Events) dient zur Steuerung des Menüs und ermöglicht, die Regelparameter zu verändern.

Die MeasurementTask stellt den anderen Tasks Soll- und Istwert zur Verfügung. Stellen Sie sicher, dass es zu keinen Race Conditions kommen kann.

Softwarevorgaben

Folgende Klassen sind für diesen Versuch vorgegeben und brauchen nicht angepasst werden:

- `Platform::BSP::PWM`:
  Die Klasse PWM kapselt die Pulsweitenmodulation. Im Konstruktor ist der gewünschte Timer, der PWM Kanal sowie die Minimal und Maximalaussteuerung in Prozent anzugeben. Über setter Methoden können PWM Frequenz (hier 10 kHz) und Dutycycle in Prozent gesetzt werden.
  Um die PWM zu konfigurieren, muss a) die gewünschte Frequenz eingestellt werden (Methode `setFrequency()`), die für alle PWM Kanäle des benutzten Timers gilt und b) der Output für den aktuellen Kanal muss aktiviert werden (Methode `outputEnable()`).
- `Platform::BSP::Adc`:
  Die Klasse Adc kapselt den Analog-Digital-Umsetzer. Im Konstruktor ist der gewünschte ADC sowie der gewünschte Kanal anzugeben, die Methode `getValue()` liest den ADC Wert für den gewählten Kanal.
- `Menu`:
  Stellt eine Basisklasse für eigene Menüs bereit. In der Datei UserMenus.h ist beispielhaft eine Anwendung mit einer Menüstruktur implementiert, die an die gegebene Aufgabe angepasst werden kann.
- `MenuTask`:
  Eine Menütask. Das eigene Startmenü (von `Menu` abgeleitet) muss der Task übergeben werden.
- `template<typename T> PidRegler`: Implementiert einen PID Regler. Die Regelparameter werden im Konstruktor übergeben. Da es sich um eine Template Klasse handelt, kann der Regler mit double oder int32_t als Datentypen für die Rechnung betrieben werden (integer oder Fließkommarechnung). Bei der Übergabe der Regelparameter ist zu beachten, dass zur Anpassung des Zahlenbereiches die berechnete Stellgröße im Regler durch den Faktor 2^{10} dividiert (Attribut `m_divider`) wird. Die Regelparameter, die dem Regler im Konstruktor übergeben werden, sind entsprechend anzupassen, also um den Faktor 2^{10} größer zu wählen. Der Regler ist nach dem Stellungsalgorithmus implementiert. Siehe dazu z. B. [4], Folie 51 ff. Der Stellwert wird der PWM übergeben, der Maximalwert der Stellgröße darf den Wert des ARR Registers nicht überschreiten. Eine Vollaussteuerung erfolgt bei einer Stellgröße von 8000 (Attribut `m_limit`). Dieses sollte bei der Reglerauslegung beachtet werden.

In der Datei measurement.cpp stehen Funktionen zur Initialisierung des Timer8 (Drehzahlerfassung) zur Verfügung.

Literatur / Dokumentation

[1] R. Hagemann; Embedded Systems Board Schaltplan
../platform/bsp/doc/board/base/PCB_Embedded_Systems_Praktikum.pdf

[2] R. Hagemann, J. Wübbelmann; Embedded Systems Board Benutzerhinweise
../platform/bsp/doc/board/base/Benutzerhinweise_STMNucleo32-ESBoardRev005_3.pdf

[3] STMicroelectronics; RM0351 Reference manual
../platform/bsp/doc/chip/rm0351-stm32l47xxx-stm32l48xxx-stm32l49xxx-and-stm32l4axxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

[4] Dirk Müller: Entwurf von SW für eingebette Systeme, Vorlesung TU Chemniz WS 2010/11.
http://docplayer.org/1450489-Entwurf-von-software-fuer-eingebettete-systeme.html

Viel Erfolg!