

# Studentische Projekte

## Displayansteuerung mit i2C Bus und Simulink Embedded Coder



Konstantin Lutz hat in seiner Bachelorarbeit eine Displayansteuerung implementiert für unsere universelle Hardwareplattform in der Leistungselektronik. Das Display kann nun für weitere Projekte komfortabel in die Simulink-Umgebung eingebunden werden. Ohne sich um die Details kümmern zu müssen können die einzelnen Zeilen ganz einfach über Strings beschrieben werden. Als Schnittstelle wird der i2C Bus verwendet. Das Display kann sowohl im Stand-Alone als auch im Monitor&Tune Modus beschrieben werden. Im Video erklärt Hr. Lutz die Details zum Projekt und wie das Display angesprochen werden kann.

Die vollständige Dokumentation zum Download:

[ansteuerung\\_eines\\_lcd-displays\\_mit\\_matlab\\_konstantin\\_lutz\\_2021.pdf](#)

und die Simulink Demo-Files:

[htwg\\_i2c\\_display\\_simulink.zip](#)

## Ansteuerung eines BLDC Motors mit Visualisierung



Auch im „Corona-Semester“ haben wir in der Vorlesung Leistungselektronik unsere Projekte realisiert. Allerdings lag der Fokus jetzt auf der Projektierung, Simulation und virtuellen Inbetriebnahme. Die Projektgruppe mit Christoph Casagrande, Johannes Huober und Lucas Flender hat in ihrem Projekt gezeigt wie sich ein BLDC - Motor ansteuern lässt mit einer Visualisierung der aktuellen Rotorposition auf einem externen PC. Nachfolgender Link zeigt das Projekt in einem kleinen Videobeitrag. [Video](#)

## Ein mit Simulink programmier- und regelbarer Buck-Boost Converter



Trifolant sitzt auf der Unterseite der Platine

zu lassen.

Dokumentation zum Download.

Lukas Hölsch hat in seiner Projektarbeit zum ersten mal im Labor für Leistungselektronik ein TI Launchpad eingesetzt zur Programmierung und Regelung eines Buck-Boost Converters. Das Besondere daran ist, dass die Programmierung in Simulink rein graphisch erfolgt. Das bedeutet es sind keinerlei Programmierkenntnisse erforderlich. Weiterhin erlaubt die Anbindung der Schaltung an Simulink eine Variation aller Parameter im laufenden Betrieb. Somit ist es auf einfache Weise möglich Regelerparameter anzupassen und sich die Ergebnisse direkt visualisieren

[projektarbeit\\_lukas\\_hoelsch.pdf](#)

## 100W Klasse D Audio Verstärker



Im Projektlabor Leistungselektronik haben wir auch dieses Semester eine 100W Audioendstufe entwickelt. Das Ergebnis von Fabian Sernatinger und Dennis Rutkowski kann sich hören lassen. Die erstmals eingesetzten Snubber über den Leistungshalbleitern haben einen deutlich positiven Effekt auf das Klangerlebnis. Die Endstufe generiert die Taktsignale mit Hilfe zweier analogen Referenzspannungen im T/2 Versatz. Dadurch können aufwendige Filterschaltungen zur Dämpfung des Trägersignals entfallen. Überblick zum Projekt

[le\\_rutkowskisernatinger\\_praesentation1.pdf](#)

, [Inbetriebnahmevideo](#)

## Aktuelle Traktorgeneration



Auch dieses Semester gab es eine neue Version des vollelektrischen Traktors. Der Fokus liegt zunehmend auf Fahrspaß womit die Studierenden dem Motor immer mehr Drehmoment abverlangen. So konnte dieses Semester der Abzug noch einmal um 50% gesteigert werden. Die Obergrenze ist aber noch nicht erreicht. Ich bin gespannt ob wir in den kommenden Jahren noch schneller beschleunigen.

## Wechselrichter



Peter Schmidt hat in seiner Bachelorarbeit einen Sinuswechselrichter aufgebaut der zukünftig im Praktikum zu den elektrischen Maschinen zum Einsatz kommen wird. Der Wechselrichter kann sowohl im Stand-Alone als auch im Grid-Betrieb zur Energieeinspeisung in das öffentliche Netz verwendet werden. Die Nachfolgearbeit wird die Frequenzerkennung verbessern und den Wechselrichter in einem kompakten Design aufbauen.

Dokumentation:

netzgefuehrter\_sinuswechselrichter\_schmidt.pdf

## Elektr. Trettraktor



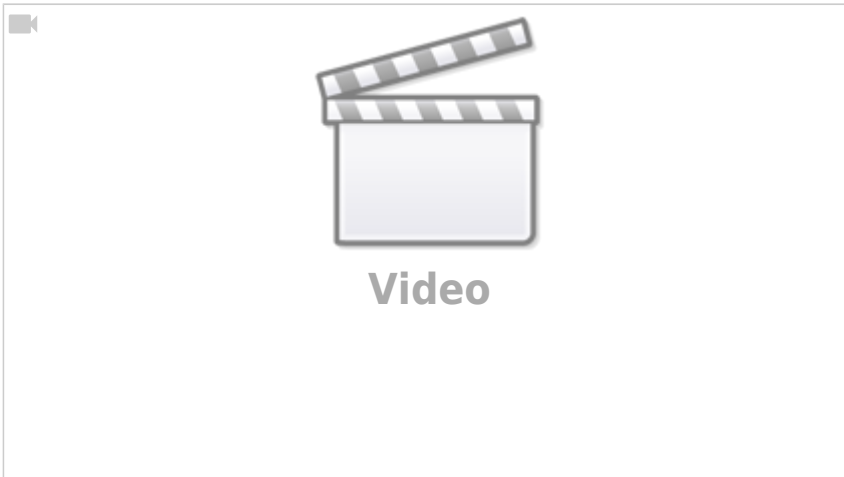
Wer ist als Kind nicht gerne mit seinem Trettraktor oder dem Dreirad unterwegs gewesen? Wieviel besser wäre es dann erst mit einer elektrisch angetriebenen Version oder einer elektrischen Unterstützung? Im Sommersemester 2017 haben wir zum ersten mal für die lange Nacht der Wissenschaft die Traktoren elektrifiziert.

Heute gibt es bereits mehrere Versionen: Als Pedelec, als Elektrofahrzeug, mit Gleichstrommotor oder BLDC Motor.

Die Traktoren werden ständig weiterentwickelt und stehen für Projekt- und Abschlussarbeiten zur Verfügung:

- Elektrische Lenkung
- Kollisionsvermeidung über Sensoren
- Autonom fahrende Traktoren
- Induktive Ladeplatte

Die Systeme können auch im Rahmen des Projektlabors der Vorlesung Leistungselektronik aufgebaut werden.

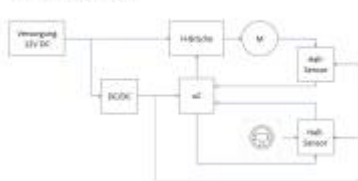


## Steer-ByWire

Sollen unsere Traktoren bald autonom fahren können müssen noch einige Hardwareanforderungen dazu umgesetzt werden. Grundvoraussetzung ist natürlich die Möglichkeit die Fahrtrichtung elektronisch vorzugeben. Das bedeutet der Lenkwinkel / Fahrerwunsch muss elektronisch umgesetzt werden. Im PAE7 Praktikum haben sich Studenten der Aufgabe angenommen und einen ersten Umsetzungsvorschlag ausgearbeitet. Es geht hier nicht darum die perfekte Lösung für eine Fahrzeugintegration zu finden, sondern mit einfachen und vorhandenen Mitteln die Funktion darzustellen. Unsere Standardkomponente im Praktikum, welche auch hier zum Einsatz kommen:

- DC- Getriebemotor (Aus Akkuschauber)
- H-Brücke aus vorhandenem Praktikumsversuch
- XC866 Trainingsboard

Blockschaltbild

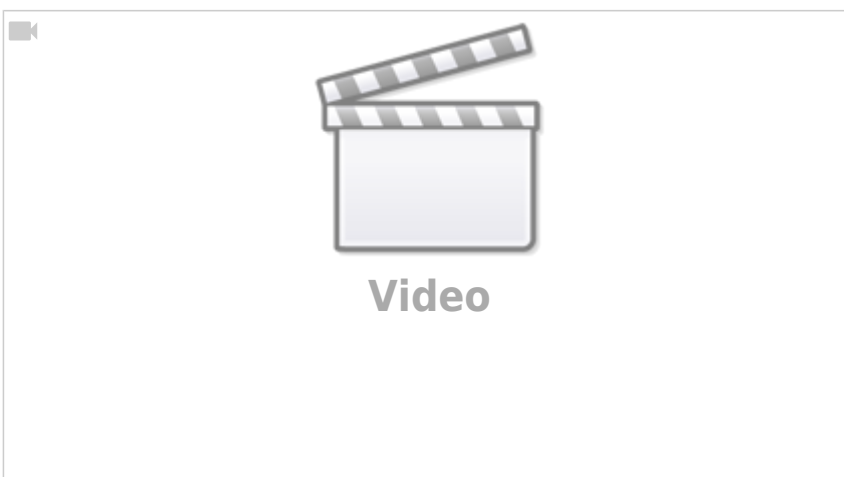


Die Lageerfassung des Lenkrads erfolgt über einen Hallsensor. Über einen zweiten Hallsensor wird die aktuelle Position der Lenkachse ermittelt. Die Software zur elektischen Lenkung hat nur eine Aufgabe: Der Fehlerwinkel zwischen Sollwinkel und Istwinkel muss ausgeglichen bzw. möglichst klein gehalten werden.

Das Projekt hat super funktioniert und es konnte gezeigt werden, dass auch mit einfachen Mitteln eine elektrische Lenkung oder ein Steer by Wire System aufgebaut werden kann. Für den Einsatz im Traktor müssen wir allerdings nach einem geeigneten Motor ausschau halten, da:

- Das hohe Haltemomente des Getriebes die Regelung ggf. sehr „nervös“ macht
- Das Getriebeispiel der Akkuschaubereinheit sehr hoch ist

Das Ergebnis kann sich sehen lassen:

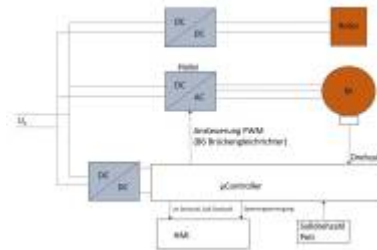


## Fahrzeuggenerator als Motor



Der Drehstromgenerator in modernen Fahrzeugen ist schon lange keine permanentterregte Gleichstrommaschine mehr, wie noch weithin angenommen. Besonders Fahrzeuge im Premiumsegment haben einen Leistungsverbrauch im Bereich von 3-4kW auf der 12V- Seite mit steigender Tendenz. Für die Versorgung des 12V Bordnetz kommen heute Drehstromgeneratoren zum Einsatz. Genauer gesagt handelt es sich hierbei um fremderregte Synchronmaschinen.

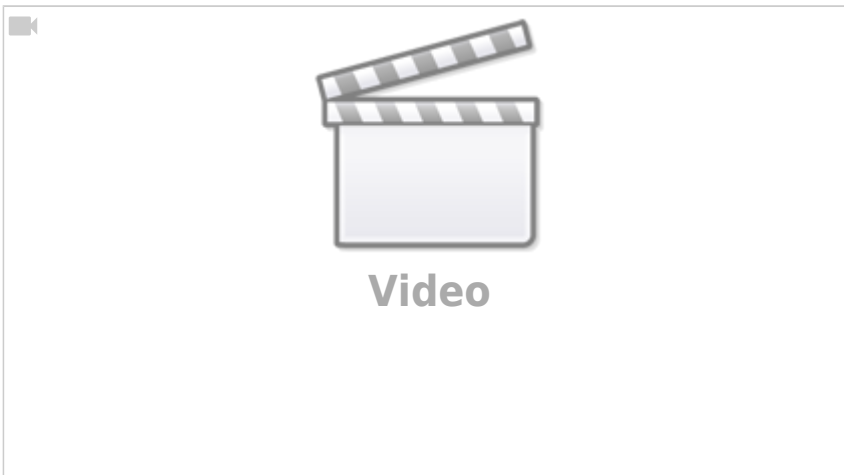
Die Regelung der Ausgangsspannung bzw. des Ausgangsstromes erfolgt über einen Linearregler welcher den Erregerstrom entsprechend beeinflusst. Stabilisiert werden die Bordnetze über einer



oder zwei Fahrzeugbattereien.

Im ersten Schritt muss

das hintere Lagerschild und die Gleichrichterioden entfernt werden um an die einzelnen Wickelabgriffe zu kommen. Danach gilt es herauszufinden wie die Wicklung verschalten ist. Typischerweise sind die Maschinen im Dreieck verschalten. Da meist keine Informationen zur Verschaltung verfügbar sind kann über eine Spannungsmessung (Drehen von Hand +Erregerfeld anlegen) der Wicklungssinn ermittelt werden.



## TI Launchpad F28069

<https://www.ti.com/tool/LAUNCHXL-F28069M#technicaldocuments>

Das Launchpad F28069 bietet für die Leistungselektronik und elektrische Maschinen einen schnellen Einstieg zur Steuerung und Regelung leistungselektronischer Schaltungen.

Zusätzlich ist es möglich die Hardware über Simulink zu programmieren, womit nur wenig Kontakt zum C-Code notwendig wird. Die grafische Programmierung über Simulink ist in der Fahrzeugentwicklung und Autosar-Systemen weit verbreitet und bietet auch ohne tiefes Wissen über die Hardware eine effiziente Programmierung.

Wie immer ist die erste Hürde (bis die Hardware mit Simulink kommuniziert) eine der Größten.

## Vorbereitende Maßnahmen:

- Launchpad F28069 organisieren
- In Matlab die Pakete installieren: Embedded Coder Support Package for Texas Instruments C2000 Processors
- Notwendige Third Party Software wie CCS (Code Composer Studio) installieren

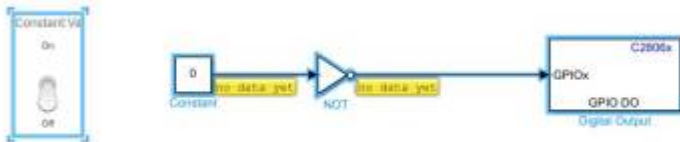
Anbei meine Fallstricke bei der Inbetriebnahme:

### Mit welchen LEDs kann ich spielen?

LED Pins:

- GPIO34 D9 (Rot) (Enable Boot)
- GPIO39 D10 (Blau)

Zum Spielen bietet sich D10 an (Achtung, Ausgabe ist invertiert) ....

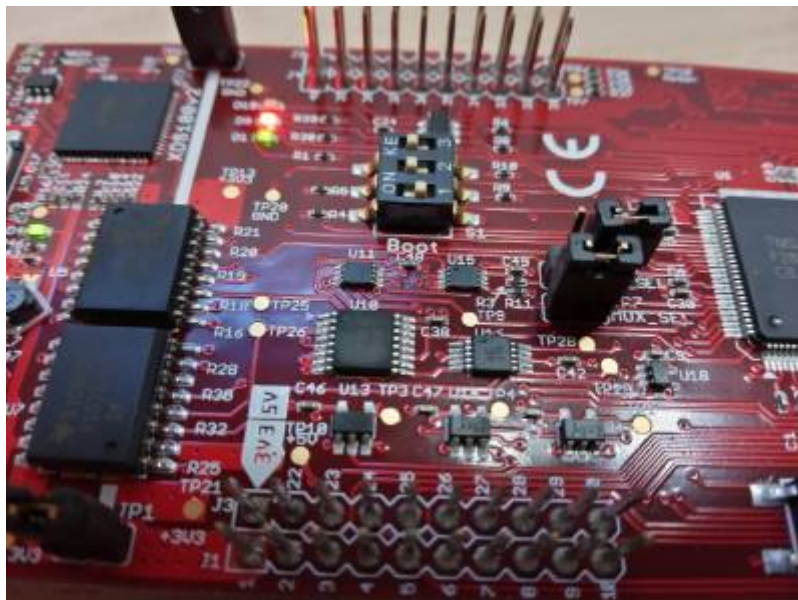


firstc2000\_led\_toggle.zip

Im Simulinkbeispiel lässt sich über den Schalter die LED D10 aktivieren und deaktivieren. Dazu muss in Simulink unter Hardware- Run on Hardware - Monitor&Tune gewählt werden.

Falls der Download scheitert:

- Unter COM-Port im Gerätemanager muss XDS100 Class USB verfügbar sein, siehe <https://e2e.ti.com/support/microcontrollers/c2000/f/171/t/670519>
- Achtung: Schalter der Boot-Mode Switches beachten. Download aus Simulink mit Modus 2 Emulation Boot, siehe Launchpad Manual
- In Simulink muss der richtige COM-Port gewählt werden, siehe <https://de.mathworks.com/help/supportpkg/texasinstrumentsc2000/ug/external-mode.html>

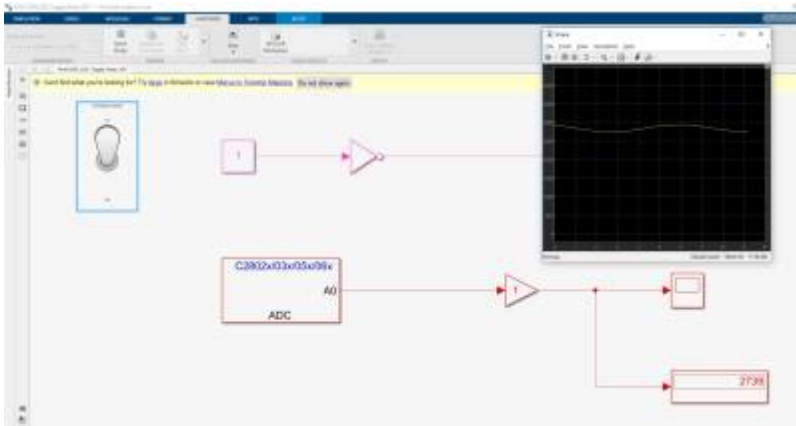


## Daten vom ADC lesen

Über den Block ADC lässt sich der Analog- Digitalwandler konfigurieren und einlesen.

**To Do:** Wo ist der Unterschied zum Block AIOX?





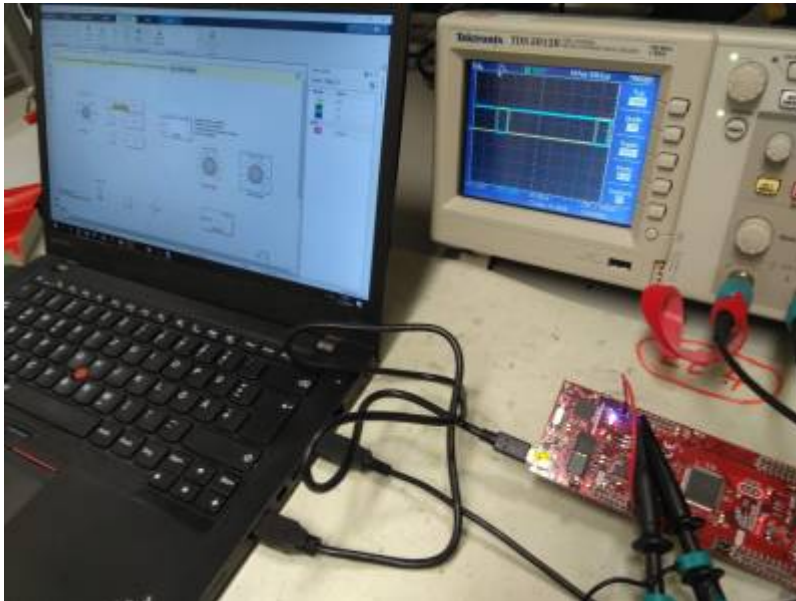
firstc2000\_led\_toggle\_read\_adc.zip

## PWM Ausgabe

Die nächste Herausforderung besteht darin eine PWM zu generieren. Idealerweise gleich zusammen mit der invertierten Ausgabe zur Ansteuerung einer Halbbrücke.

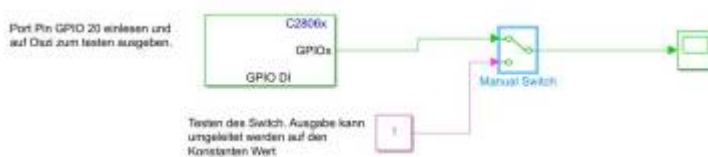
Folgendes Setup ist eine Erweiterung der bisherigen Versuch zur PWM Ausgabe.

Über die Dashboard-Potis lässt sich die Einschaltdauer ändern sowie die Totzeiten für die steigende und fallende Flanke.



firstc2000\_led\_toggle\_read\_adc\_calcvalue\_epwm.zip

## Digitalport einlesen



Signalquelle im Monitor&Tune Modus direkt ändern lassen, siehe

firstc2000\_led\_toggle\_read\_adc\_calcvalue\_epwm\_readdigital.zip

Der erste Versuch direkt einen Interrupt Routine auszulösen über den GPIO Pin 20, siehe unten ging leider schief. Die neue Strategie ist daher erst zu testen einen Digitalen Eingang auszulesen und damit auch zu prüfen ob die Logik-Pegel erkannt werden. Erzeugt werden die Impulse vom GPIO 0 welcher ein in Simulink einstellbares Rechtecksignal ausgibt. Erfreulich ist, dass sich die

## Interrupt



Das Launchpad besitzt eine Reihe an interruptfähigen Pins. In der Pin-Übersicht des Quick-Start-Guide mit (!) gekennzeichnet. Das folgende Beispiel löst durch einen Flankenwechsel am GPIO20 einen Interrupt aus. In der ISR wird zuerst Port 20 eingelesen, dann die LED entsprechend dem Port-Pin gesetzt. Achtung, das LED Signal ist invertiert. Zum testen lassen wir uns am GPIO0 ein Rechtecksignal erzeugen. Damit der Interrupt ausgelöst wird muss eine Drahtbrücke von GPIO 0 auf 20 gesetzt werden.

Wichtig ist, dass in Matlab unter Hardware-Settings → Target hardware resources → External Interrupt → der XINT3 ausgewählt wird für GPIO 20. Leider lassen sich nur drei externe Interruptquellen anwählen.

firstc2000\_led\_toggle\_read\_adc\_calvalue\_epwm\_interrupt.zip

## Stand-Alone

Um eine Stand-Alone Applikation (ohne Simulinkanbindung) zu generieren müssen folgende Änderungen vorgenommen werden.

- In Simulink unter Hardware Implementation „Boot From Flash“ aktivieren. Danach das Programm mit „Build, Deploy and Start“ auf den Controller flashen. Das Programm starte nach erfolgreichem Flashen auf dem Controller.



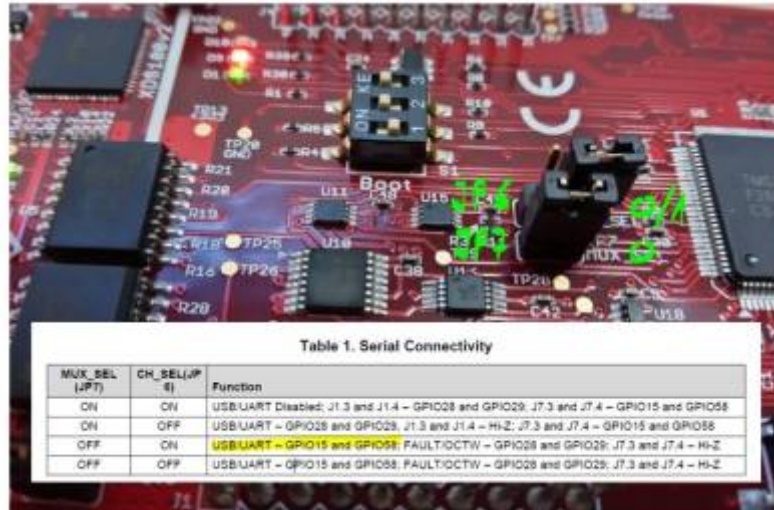
- Auf dem Launchpad die Dipschalter (Mäuseklavier) auf die Stellung „Get Mode“ setzen: ON ON OFF. Damit wird die richtige Bootkonfiguration gesetzt, damit nach einem Spannungsreset das Programm geladen wird. Dies ist der letzte Schritt bevor die Schaltung sich nicht mehr aus der USB Verbindung versorgt. (Ausgangszustand: OFF ON ON)
- Jetzt können wir das Launchpad über die Zwischenkreisspannung (Batterie oder ähnliches versorgen). Dazu auf der Leiterkarte JP4 setzen. Auf dem **Launchpad** die Jumper JP1 und JP2 entfernen.

## Monitoring über Serielle Schnittstelle

In vielen Fällen stößt die Monitor&Tune Funktion an Ihre Grenzen. Insbesondere wenn es darum geht Interrupts mit hoher Wiederholrate zu übertragen. Hier ist es sinnvoller den µC ohne Tune-Funktion laufen zu lassen und sich die Werte über die serielle Schnittstelle an den PC (beliebiges Interface) übertragen zu lassen. Besonders einfach geht das natürlich wieder mit Simulink.

Vorgehensweise:

1. Code für µC in Simulink erstellen und über den Button Build, Deploy & Start auf den µC übertragen. Im Code müssen natürlich die zu übertragenden Signale an die Serielle Schnittstelle übergeben werden. (Siehe Beispielcode). Dieser Schritt muss bei jedem Power-Up des µC wiederholt werden. Außer, es werden die Jumper-Settings für den Stand-Alone Betrieb gesetzt, siehe oben.
2. Das Serial Communication Interface A ist für den Monitor&Tune Modus reserviert. Zur Übertragung zu einem Host-PC wird das SCI B Modul verwendet. Da der FTDI Serial-To-USB nur einen Kanal beinhaltet müssen wir die Jumper-Settings auf dem Board verändern entsprechend Tabelle 1 im Manual. JP 7 = offen, JP 6 = gesetzt/offen.
3. Simulink als Host und Block zum lesen/schreiben der Seriellen Schnittstelle öffnen. (Achtung, ggf. muss die COMport Nummer angepasst werden).

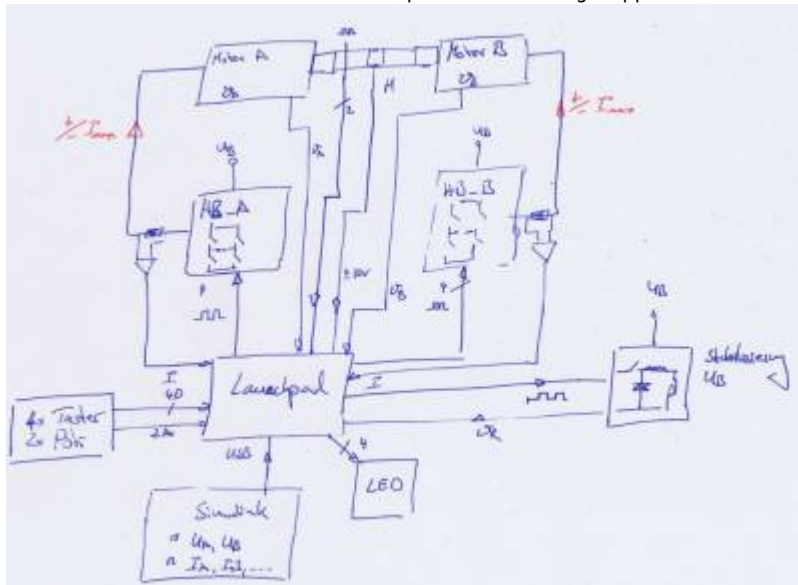


Demo-Download:

host-read.zip

## DC- Motor/Generator Prüfstand

Mit Hilfe von Simulink und dem TI-Launchpad werden zwei gekoppelte Gleichstrommaschinen getrennt voneinander angesteuert.



Hardware:

- Drehzahl / Drehmomentmessung an der Welle
- Strommessung (positiv, negativ) Motor A/B
- Spannung stellen Motor A/B
- Strom stellen Motor A/B (positiv, negativ), im Generatorbetrieb (Strom negativ) muss Leistung in Wärme umgesetzt werden
- Temperatur Motor A/B

Funktionsblöcke in Simulink:

- Konstante Spannung Motor (positiv, negativ) A/B (Strombegrenzung auf +/- I<sub>max</sub>)
- Konstanter Motorstrom (positiv, negativ) A/B (Spannungsbegrenzung auf +/- U<sub>max</sub>)
- Konstantes Drehmoment (positiv, negativ) A/B (Strombegrenzung)
- Lastsimulation, es können vorgegebene Drehmomentverläufe (z.B. Fahrprofile) abgefahren werden
- Konstante Drehzahl A/B (Spannungsbegrenzung, Strombegrenzung)
- Konstante Position A/B (Spannungsbegrenzung, Strombegrenzung)

Versuchsdurchführung:

- Messung der Kennlinie n(M) Motor A, Parameter: Ausgangsspannung

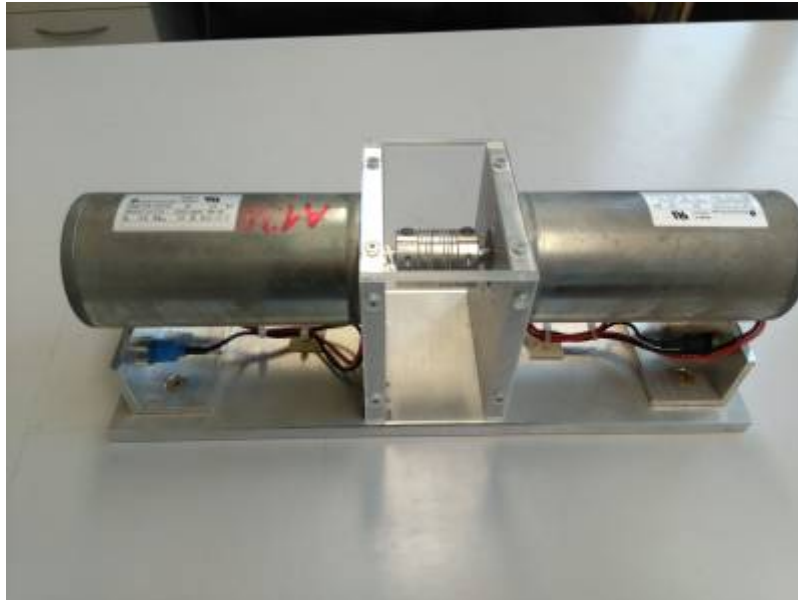


- Messung der Kennlinie  $I(M)$  Motor A, Parameter: Ausgangsspannung
- Wirkungsgrad Motor A
- Hochlaufzeitzeit unter Last

## Hardware-Blöcke

Parameter:

- $U_{\max} = 30V$
- $I_{\max} = 20A$

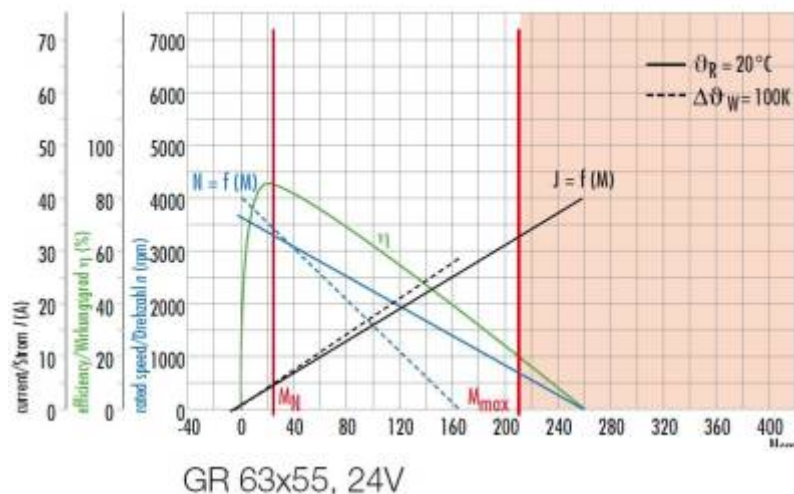


Das Bild zeigt den aktuellen Stand der gekoppelten Motoren. [Datenblatt](#)

Es fehlen noch der Encoder und die Drehmomentmessung.

Wellendurchmesser: 8mm, Technische Zeichnung

[gr-63x55-8844201270.pdf](#)



GR 63x55, 24V

Aus Datenblatt: Dunkermotoren.de Kenndaten eines Motors am Bemessungspunkt: Spannung: 24V Strom: 4,9A Drehzahl: 3350 1/min Drehmoment: 27Ncm Leistung 94,7W

Weitere Kenndaten: Leerlaufdrehzahl: 3650 1/min Leerlaufstrom: 0,4A Anlaufmoment (24V): 260Ncm Anlaufstrom 40A Entmagnetisierungsstrom 30A

Hardwarebegrenzung auf 20A ergibt ein maximales Drehmoment der Motoren von 128Ncm. Sicherheitsreserve für Impulsbelastung: Drehmomentmessung bis 300Ncm.

Nachfolgend ein Teil der notwendigen Hardwarekomponenten:

## Leistungswiderstand

Der Widerstand wird verwendet damit im Genertorbetrieb die erzeugte Energie in Wärme umgesetzt werden kann, da unsere Netzteile im Allgemeinen nicht rückspeisefähig sind.

Beispiel für einzusetzenden Lastwiderstand:

<https://de.farnell.com/arcol/hs100-r10-j/drahtwiderstand-0r1-5-axial/dp/2478150>

Der Widerstand darf nicht zu groß gewählt werden, da ansonst nur für hohe Drehzahlen (große U<sub>ind</sub>) ein ausreichend großer Strom fließen kann.

Der Widerstand muss zusätzlich auf einem Kühlkörper montiert werden und wird mit einem Temperatursensor ausgestattet.

Die Ansteuerung des Widerstandes erfolgt über PWM (min. 50kHz) über das Launchpad, geregelt wird auf konstante Versorgungsspannung.

Der Leistungswiderstand wird in der ersten Version der Schaltung nicht weiter verfolgt, das stets zwei gekoppelte Motoren verwendet werden. Die Gefahr eines ungewollten Rückspeisens entsteht aufgrund der Verluste in der Leistungselektronik und den Motoren nicht.

## Spannungsversorgung

Die Motoren müssen jeweils separat mit einer Spannung  $-U_{\max} < U < +U_{\max}$  angesteuert werden können.

Der maximale Laststrom wird auf  $\pm I_{\max}$  begrenzt.

Die Ansteuerung der Spannungsregelung erfolgt über das Launchpad, geregelt wird auf konstante Ausgangsspannung.

## Drehmomentsensor

Das an der Welle anliegende Drehmoment wird mittels einem Drehmomentsensor erfasst.

Analoger Messbereich  $\pm 10V$  für z.B.  $M_{\max} = 1Nm$

<https://www.burster.de/de/drehmomentsensoren/p/detail/8661/>

Nach den ersten Anfragen bei Herstellen von Drehmomentsensoren ist klar, dass sich die Messwellen im Bereich von  $> 2000€$  abspielen → viel zu teuer.

Wir werden daher den Antriebsmotor frei lagern und über einen Hebelarm die Kraft auf einen Hebelarm messen.

## Encoder

Zur Drehzahl und Drehrichtungserkennung muss ein Encoder zum Einsatz kommen.

Der Encoder RE30/500 ist mit 5V Impulsausgängen A, B und einem Indexausgang ausgestattet.

## Pinbelegung Launchpad Hardware V01

Funktion	Analog / Digital	Anpassung	Launchpad	Launchpad Pin	Schaltplanlabel	Sonstiges, Schutzbeschaltung
PWM Out Motor A HB_A_Links_HS HalbBrücke_A_Links_HighSide	D	Logikpegel Treiber	1PWM_AH EPWM1A	40	M1B1H_IN	Todzeiteinstellung für jede Halbbrücken im $\mu C$ ermöglichen
PWM Out Motor A HB_A_Links_LS	D	Logikpegel Treiber	1PWM_AL EPWM1B	39	M1B1L_IN	
PWM Out Motor A HB_A_Rechts_HS	D	Logikpegel Treiber	1PWM_BH EPWM2A	38	M1B2H_IN	
PWM Out Motor A HB_A_Rechts_LS	D	Logikpegel Treiber	1PWM_BL EPWM2B	37	M1B2L_IN	
PWM Out Motor A* HB_A*_HS	D	Logikpegel Treiber	1PWM_CH EPWM3A	36	M1B3H_IN	Vorhalt für BLDC Ansteuerung Motorport A
PWM Out Motor A* HB_A*_LS	D	Logikpegel Treiber	1PWM_CL EPWM3B	35	M1B3L_IN	Vorhalt für BLDC Ansteuerung Motorport A
PWM Out Motor B HB_B_Links_HS	D	Logikpegel Treiber	2PWM_AH EPWM4A	80	M2B1H_IN	
PWM Out Motor B HB_B_Links_LS	D	Logikpegel Treiber	2PWM_AL EPWM4B	79	M2B1L_IN	
PWM Out Motor B HB_B_Rechts_HS	D	Logikpegel Treiber	2PWM_BH EPWM5A	78	M2B2H_IN	
PWM Out Motor B HB_B_Rechts_LS	D	Logikpegel Treiber	2PWM_BL EPWM5B	77	M2B2L_IN	
PWM Out Motor B* HB_B*_HS	D	Logikpegel Treiber	2PWM_CH EPWM6A	76	M2B3H_IN	Vorhalt für BLDC Ansteuerung Motorport A

Funktion	Analog / Digital	Anpassung	Launchpad	Launchpad Pin	Schaltplanlabel	Sonstiges, Schutzbeschaltung
PWM Out Motor B* HB_B*_LS	D	Logikpegel Treiber	2PWM_CL EPWM6B	75	M2B3L_IN	Vorhalt für BLDC Ansteuerung Motorport A
Strommessung Motorstrom A Messbereich +/- 20A	A	ACS712	ADCINA0	27	Strom_M1_33	
Strommessung Motorstrom B Messbereich +/- 20A	A	ACS712	ADCINB6	6	Strom_M2_33	
Versorgungsspannung +15V	A		ADCINB2	26	+15V_33	Überwachung der Versorgungsspannung,
Batteriespannung	A		ADCINA2	25	Vbat*_33	
Versorgungsspannung +3,3V	A		ADCINB1	24	+3,3V	
Drehmomentmessung I Kraftsensor 1	A	Spannungsteiler	ADCINA6	2	Moment_Pos_33	
Drehmomentmessung II Kraftsensor 1	A	Spannungsteiler	ADCINA7	23	Moment_Neg_33	
Encoder Pulse	D Interrupt 1	Logikpegel anpassen	GPIO19 ECAP1	19	Encoder-HallA_Interrupt_33	Jumper BLDC Hall- Sensoren, Wird auch als ECAP1 benutzt
Encoder Drehrichtung Rechts	D	Logikpegel anpassen	GPIO44	18	Rechts_33	
Encoder Drehrichtung Links	D	Logikpegel anpassen	GPIO14	47	Links_33	
Encoder Index	D Interrupt 2	Logikpegel anpassen	GPIO25 ECAP2	54	Index_HallB_Interrupt_33	Jumper
Vorhalt Hallsensor BLDC	D Interrupt 3	Logikpegel anpassen	GPIO26 ECAP3	58	HallC_33	BLDC Hall-Sensoren
Bafang Speed-Signal	D	Logikpegel anpassen	GPIO16	15	Speed_33	
Taster 1 für manuelle Ansteuerung	D	Pull-Up	GPIO12	5	T1	
Taster 2 für manuelle Ansteuerung	D	Pull-Up	GPIO22	8	T2	
Taster 3 für manuelle Ansteuerung	D	Pull-Up	GPIO13	34	T3	Achtung: Liegt auf Enable-Boot. Ggf. problematisch
Potentiometer A	A		ADCINB7	63	Poti 1	Manuelle Drehzahlstellung Motor A (frei Programmierbar in Simulink)
Potentiometer B	A		ADCINB4	64	Poti 2	
Temperatur Motor A	A	LM355 mit Impedanzwandler	ADCINA5	65	Temperatur1_33	
Temperatur Motor B	A	LM355 mit Impedanzwandler	ADCINB5	66	Temperatur2_33	
LED-Anzeige	D		GPIO51	12	LED_Rot	
LED-Anzeige	D		GPIO27	59	LED_Grün	
Display i2C, SCL	D		SCL GPIO 33	9	SCL	
Display i2C, SDA	D		SDA GPIO 32	10	SDA	
Analoge Erweiterung 1	A		ADCINB0	28		
Analoge Erweiterung 2	A		ADCINA1	29		



Aktualisiert 05.04.2020:

Die aktuelle Schaltung mit PCB zum Download in der ersten Version:

[dcdc\\_v01\\_update.zip](#)

Das Layout zur Inbetriebnahme der Schaltung in Simulink ermöglicht einen schnellen Test aller IOs.



Folgende Tests können durchgeführt werden:

- LEDs aktivieren
- Externe Schalterposition einlesen
- Pegel der Hall und Encodersignale einlesen
- Zeit zwischen zwei Hall-Flanken ermitteln
- Test aller Halbbrücken
- Analog-Werte einlesen und umwandeln

[Download der Inbetriebnahme-Software](#)

[inbetriebnahme\\_v05.zip](#)

Fazit und Erkenntnisse während des Aufbaus der Inbetriebnahmesoftware:

- Compiler und Flash-Vorgang dauert sehr lange.
- Dashboard Anzeigeelemente funktionieren leider nicht. Zur Anzeige stehen nur Display und Scopes zur Verfügung
- Das eCap Modul zur Messung der Periodendauer (Encoder) läuft nur alleine. Eine gleichzeitige Portabfrage über GPIO DI Pin ist nicht möglich

## Pinbelegung Launchpad Hardware V02

Die Umsetzung der ersten Hardware V01 hat nahezu problemlos funktioniert. Für die Hardware der zweiten Generation werden folgende Änderungen vorgenommen:

Nr.	Änderungen zu V01
1	Falsches Label zur Spannungsmessung 3,3V, daher keine Verbindung auf dem PCB
2	3x Zusätzliche Stromsensoren zur Messung der Phasenströme Motor A
3	Wechsel auf IR2110 Treiber mit SD Eingang, da die PWM ansonsten nicht deaktiviert werden kann. PWM 0% bedeutet die Low-Side Mosfets sind dauerhaft geschlossen womit kein Freilauf möglich ist. SD Pin deaktiviert die Brücken.
4	Ausgabe der Encoder Impulse über Jumper optional auf Hall C. Damit kann später auf Position geregelt werden. Simulink erlaubt keine doppelte Pinfunktion z.B. eCAP und gleichzeitig einen Interrupt
5	Neuer Drehmomentsensor mit 50N, da aktuelle Version mit 10N zu gering
6	Taster 1 und 2 werden auf ADC-Ports gesetzt, damit mehr GPIO freiwerden
7	Analoge Erweiterungsports entfallen

Pinbelegung für V02

Funktion	Analog / Digital	Anpassung	Launchpad	Launchpad Pin	Schaltplanlabel	Sonstiges, Schutzbeschaltung
PWM Out Motor A HB_A_Links_HS HalbBrücke_A_Links_HighSide	D	Logikpegel Treiber	1PWM_AH EPWM1A	40	M1B1H_IN	Todzeiteinstellung für jede Halbbrücken im µC ermöglichen
PWM Out Motor A HB_A_Links_LS	D	Logikpegel Treiber	1PWM_AL EPWM1B	39	M1B1L_IN	
<b>SD A Links</b>		Logikpegel Treiber	<b>GPIO50</b>	<b>13</b>	<b>M1B1_SD</b>	<b>Motor 1 Brücke 1 SD</b>
PWM Out Motor A HB_A_Rechts_HS	D	Logikpegel Treiber	1PWM_BH EPWM2A	38	M1B2H_IN	
PWM Out Motor A HB_A_Rechts_LS	D	Logikpegel Treiber	1PWM_BL EPWM2B	37	M1B2L_IN	
<b>SD A Rechts</b>		Logikpegel Treiber	<b>GPIO24</b>	<b>55</b>	<b>M1B2_SD</b>	<b>Motor 1 Brücke 2 SD</b>
PWM Out Motor A* HB_A*_HS	D	Logikpegel Treiber	1PWM_CH EPWM3A	36	M1B3H_IN	Vorhalt für BLDC Ansteuerung Motorport A
PWM Out Motor A* HB_A*_LS	D	Logikpegel Treiber	1PWM_CL EPWM3B	35	M1B3L_IN	Vorhalt für BLDC Ansteuerung Motorport A
<b>SD A*</b>		Logikpegel Treiber	<b>GPIO52</b>	<b>53</b>	<b>M1B3_SD</b>	<b>Motor 1 Brücke 3 SD</b>
PWM Out Motor B HB_B_Links_HS	D	Logikpegel Treiber	2PWM_AH EPWM4A	80	M2B1H_IN	
PWM Out Motor B HB_B_Links_LS	D	Logikpegel Treiber	2PWM_AL EPWM4B	79	M2B1L_IN	
<b>SD B Links</b>	D	Logikpegel Treiber	<b>GPIO53</b>	<b>52</b>	<b>B2B1_SD</b>	<b>Motor 2 Brücke 1 SD</b>
PWM Out Motor B HB_B_Rechts_HS	D	Logikpegel Treiber	2PWM_BH EPWM5A	78	M2B2H_IN	
PWM Out Motor B HB_B_Rechts_LS	D	Logikpegel Treiber	2PWM_BL EPWM5B	77	M2B2L_IN	
<b>SD B Rechts</b>	D	Logikpegel	<b>GPIO12</b>	<b>5</b>	<b>M2B2_SD</b>	<b>Motor 2 Brücke 2 SD</b>
PWM Out Motor B* HB_B*_HS	D	Logikpegel Treiber	2PWM_CH EPWM6A	76	M2B3H_IN	Vorhalt für BLDC Ansteuerung Motorport A
PWM Out Motor B* HB_B*_LS	D	Logikpegel Treiber	2PWM_CL EPWM6B	75	M2B3L_IN	Vorhalt für BLDC Ansteuerung Motorport A
<b>SD B*</b>	D	Logikpegel Treiber	<b>GPIO22</b>	<b>8</b>	<b>M2B3_SD</b>	<b>Motor 2 Brücke 3 SD</b>
Strommessung Motorstrom A Messbereich +/- 20A	A	ACS712	ADCINA0	27	Strom_M1_33	



Funktion	Analog / Digital	Anpassung	Launchpad	Launchpad Pin	Schaltplanlabel	Sonstiges, Schutzbeschaltung
<b>Strommessung Motor 1 Brücke 1</b>	A	ACS712	<b>ADCINA3</b>	<b>67</b>	<b>Strom_M1_B1_33</b>	<b>Phasenstrom Motor 1 B1</b>
<b>Strommessung Motor 1 Brücke 2</b>	A	ACS712	<b>ADCINB3</b>	<b>68</b>	<b>Strom_M1_B2_33</b>	<b>Phasenstrom Motor 1 B2</b>
<b>Strommessung Motor 1 Brücke 3</b>	A	ACS712	<b>ADCINA4</b>	<b>69</b>	<b>Strom_M1_B3_33</b>	<b>Phasenstrom Motor 1 B3</b>
Strommessung Motorstrom B Messbereich +/- 20A	A	ACS712	ADCINB6	6	Strom_M2_33	
Versorgungsspannung +15V	A		ADCINB2	26	+15V_33	Überwachung der Versorgungsspannung,
Batteriespannung	A		ADCINA2	25	Vbat*_33	
Versorgungsspannung +3,3V	A		ADCINB1	24	+3,3V	
Drehmomentmessung I Kraftsensor 1	A	Spannungsteiler	ADCINA6	2	Moment_Pos_33	
Drehmomentmessung II Kraftsensor 1	A	Spannungsteiler	ADCINA7	23	Moment_Neg_33	
Encoder Pulse ECPA Modul	D Interrupt 1	Logikpegel anpassen	GPIO19 ECAP1	19	Encoder-HallA_Interrupt_33	Jumper BLDC A Hall-Sensor oder ECAP1 Modul
Encoder Drehrichtung Rechts	D	Logikpegel anpassen	GPIO44	18	Rechts_33	
Encoder Drehrichtung Links	D	Logikpegel anpassen	GPIO14	47	Links_33	
Encoder Index	D Interrupt 2	Logikpegel anpassen	GPIO25 ECAP2	54	Index_HallB_Interrupt_33	Jumper BLDC B Hall-Sensor oder Encoder Index
<b>Vorhalt Hallsensor BLDC</b>	D Interrupt 3	Logikpegel anpassen	GPIO26 ECAP3	58	<del>HallC_33</del> <b>Encoder-HallC_Interrupt_33</b>	<b>Jumper BLDC Hall-Sensoren oder Encodersignal zur Positionsbestimmung</b>
Bafang Speed-Signal	D	Logikpegel anpassen	GPIO16	15	Speed_33	
<b>Taster 1 für manuelle Ansteuerung</b>	D	Pull-Up	GPIO12 <b>ADCINB0</b>	<b>5 28</b>	T1	<b>Keine GPIO mehr übrig. Lösung über ADC Ports</b>
<b>Taster 2 für manuelle Ansteuerung</b>	D	Pull-Up	GPIO22 <b>ADCINA1</b>	<b>8 29</b>	T2	<b>Keine GPIO mehr übrig. Lösung über ADC Ports</b>
Taster 3 für manuelle Ansteuerung	D	Pull-Up	GPIO13	34	T3	
Potentiometer A	A		ADCINB7	63	Poti 1	Manuelle Drehzahlstellung Motor A (frei Programmierbar in Simulink)
Potentiometer B	A		ADCINB4	64	Poti 2	
Temperatur Motor A	A	LM355 mit Impedanzwandler	ADCINA5	65	Temperatur1_33	
Temperatur Motor B	A	LM355 mit Impedanzwandler	ADCINB5	66	Temperatur2_33	
LED-Anzeige	D		GPIO51	12	LED_Rot	
LED-Anzeige	D		GPIO27	59	LED_Grün	
Display i2C, SCL	D		SCL GPIO 33	9	SCL	
Display i2C, SDA	D		SDA GPIO 32	10	SDA	
Analoge Erweiterung 1	A		ADCINB0	28		
Analoge Erweiterung 2	A		ADCINA1	29		

Hardware Version DCDC\_V02 zum Download:

dcdc\_v02.zip

Leider habe ich es in KiCad nicht geschafft alle Projektdateien umzubenennen da ansonsten ständig die Footprints verlorengehen.

Daher bleibt die Bezeichnung der Schaltpläne auf V01. KiCad ist hier leider nicht sehr benutzerfreundlich ..... (oder ich kenne mich zu wenig aus)

Schaltplan als pdf:

dcdc\_v02.pdf

Software zur Inbetriebnahme der DCDC\_V02 Hardware:

inbetriebnahme\_v07\_4\_dcdc02.zip