Agile Hardware-Entwicklung

Dipl. Ing. Mario Blunk

Buchfinkenweg 3 / 99097 Erfurt / Germany
http://www.blunk-electronic.de

2022-09-26

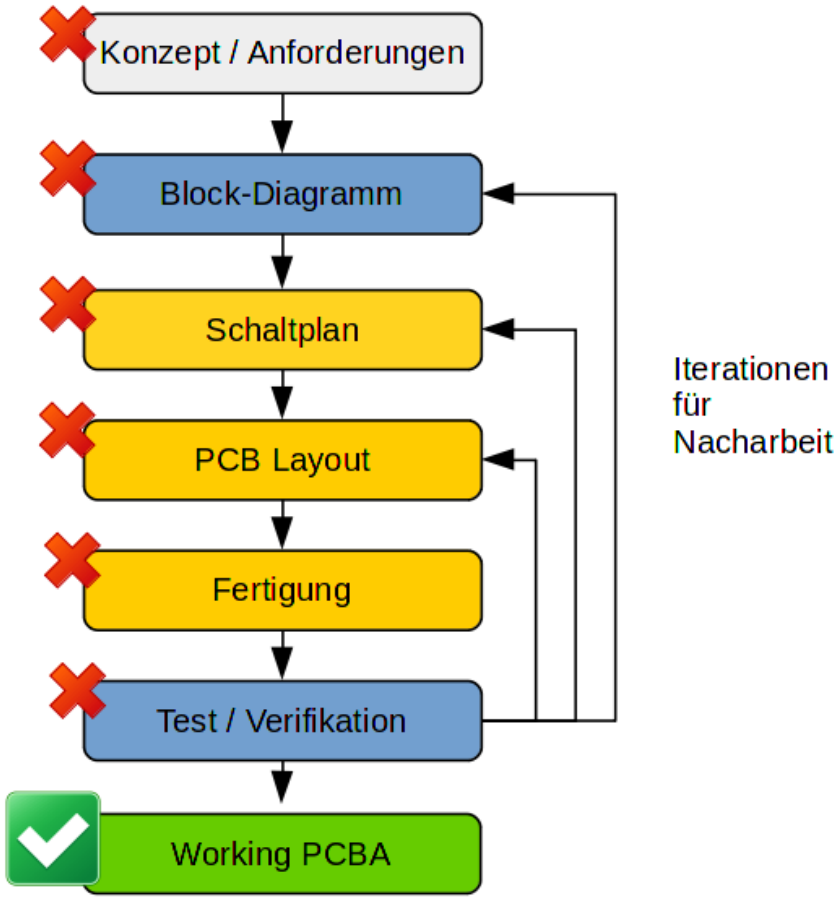
Abstract

Agile Hardware-Entwicklung bedeutet Arbeiten im Team. Während die klassische Entwicklung von Elektronik oft auf dem Tisch eines Entwicklers stattfand, werden mit dieser Methode Teilbereiche auf mehrere Mitspieler verteilt. Die für das Endprodukt relevanten Schaltungsteile bleiben vom Stadium des Prototypen bis ins Finale erhalten. Agile Hardware-Entwicklung wird angewendet, wenn Anforderungen an das Produkt unscharf sind, sich oft ändern und somit das Pflegen von Lastenheften keinen Sinn mehr macht. Dieses Seminar erläutert die Grundlagen agiler Prozesse, die einzuhaltenden Regeln für die Mitspieler, benötigte Werkzeuge und die praktische Anwendung anhand einfacher Beispiele.

Überblick

1. Prozess der HW-Entwicklung im allgemeinen
2. Vergleich Wasserfall-Modell mit agiler Methode
3. Anwendung, Nutzen und Vorteile
4. Zeitpläne und Aufwand
5. Regeln und Richtlinien (Style Guides und Clean Code)
6. Schnittstellen zwischen Modulen
7. Werkzeuge (CAM-Prozessor, ERP, Versionskontrolle, ...)
8. Beispiele und praktische Übungen in bereitgestellter Entwicklungsumgebung

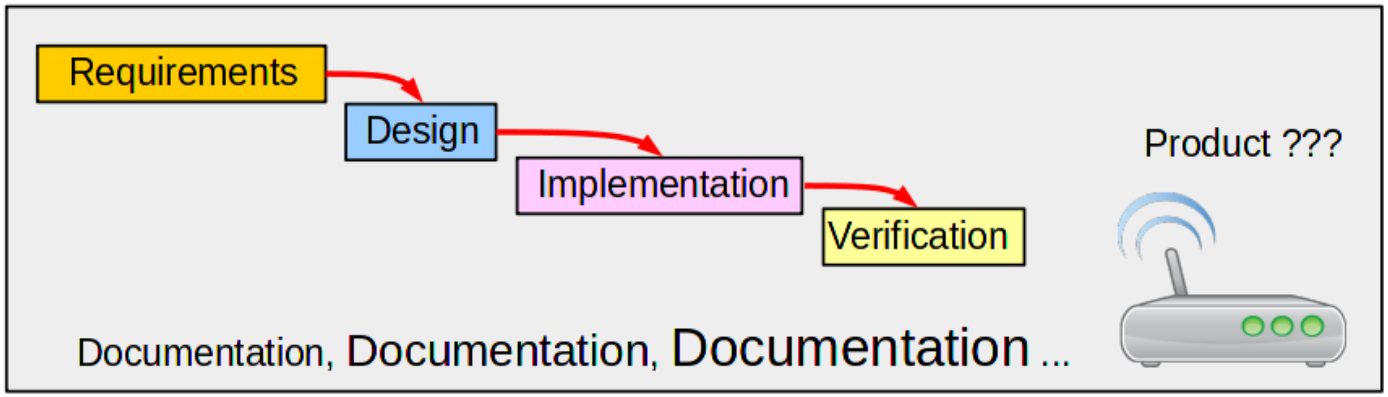
Prozess der HW-Entwicklung



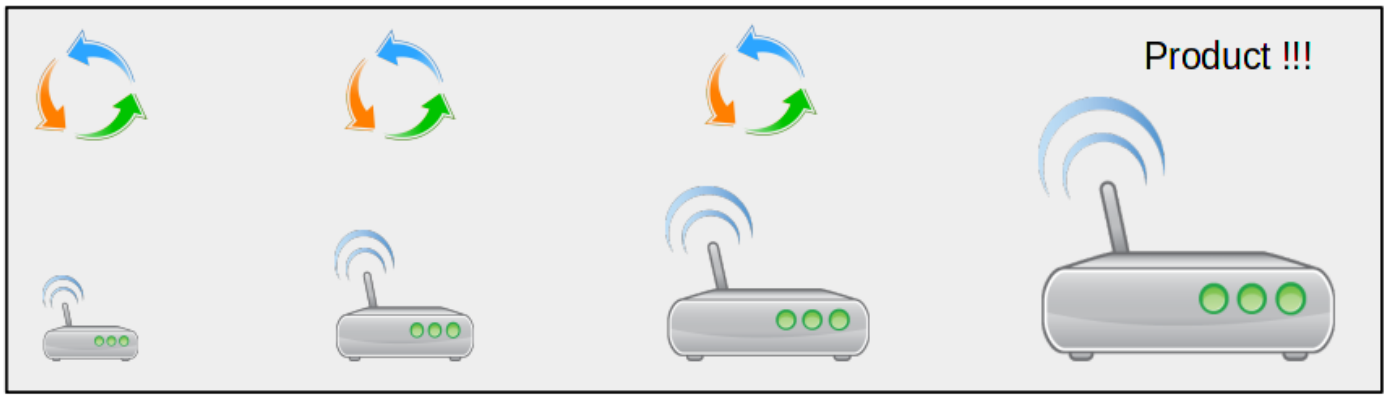
Dokumentation, Firmware, Logiksynthese, Materialwirtschaft ?

Vergleich Wasserfall-Modell mit agiler Methode 1.

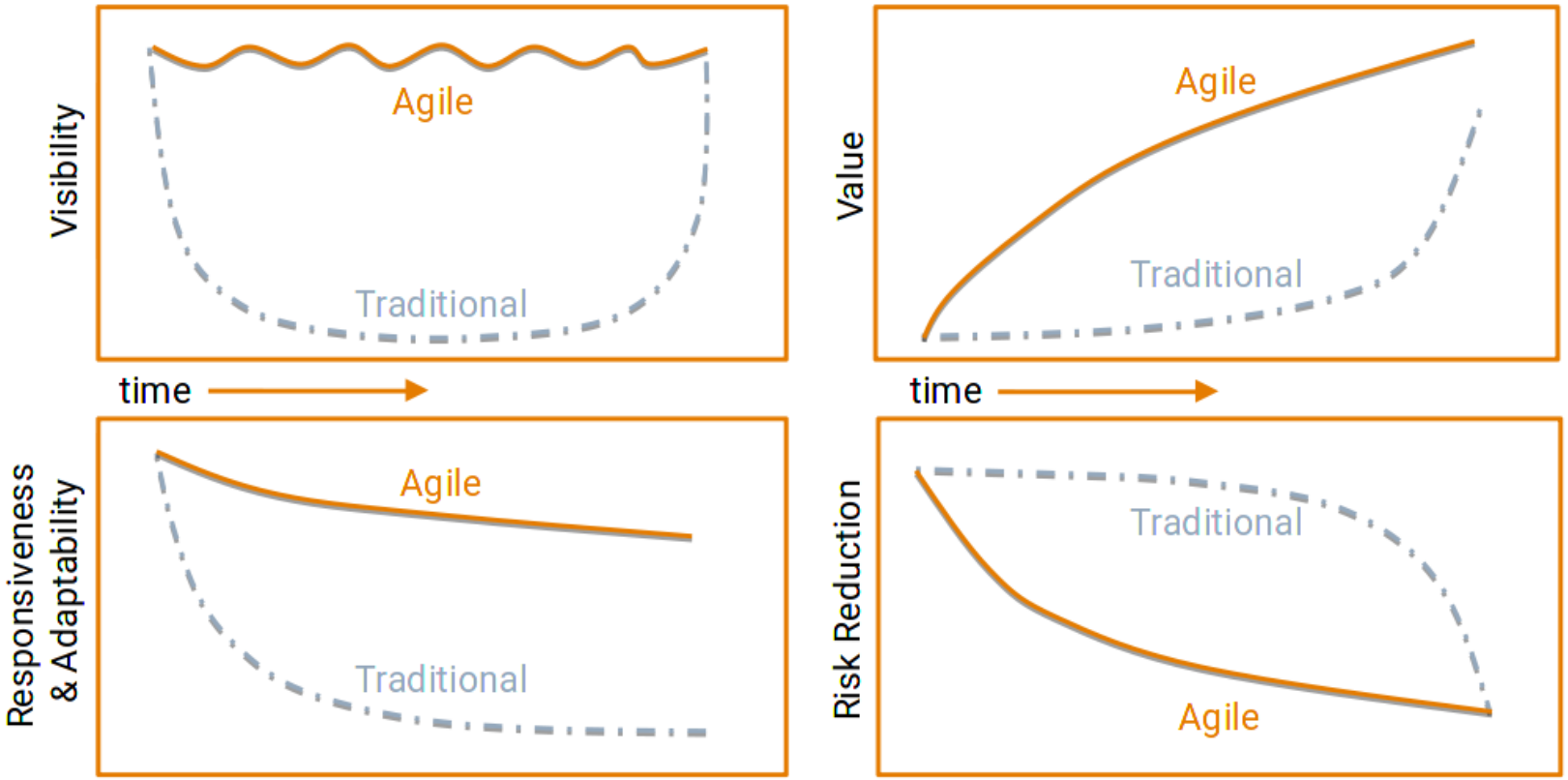
Wasserfall:



agile Entwicklung:

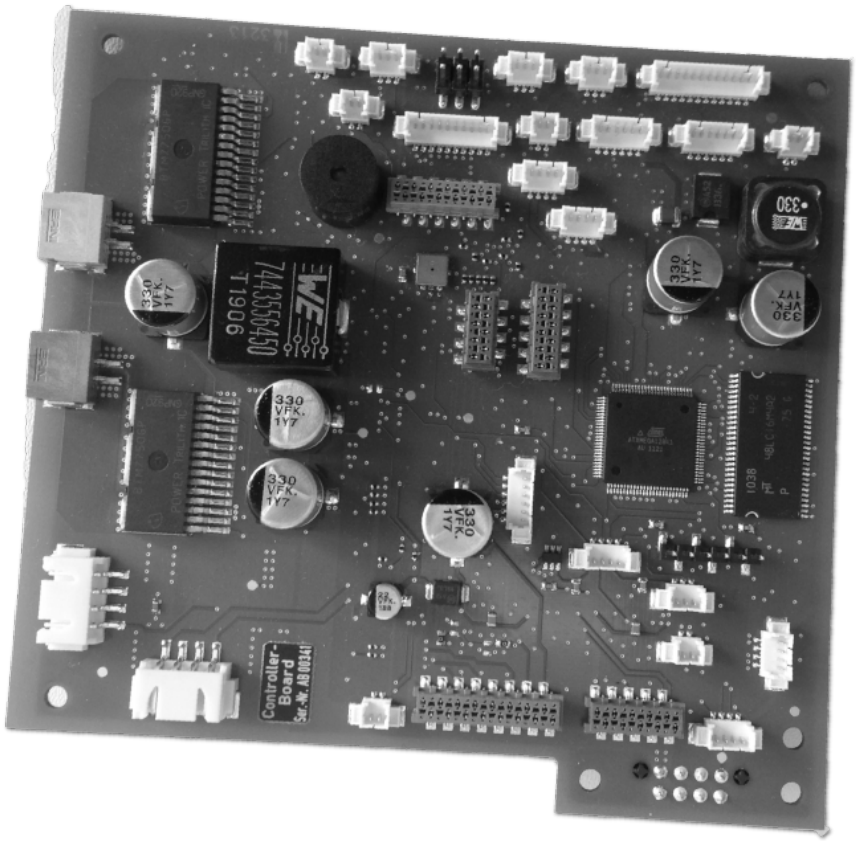


Vergleich Wasserfall-Modell mit agiler Methode 2.



Quelle: https://www.versionone.com/agile-101/agile-software-development-benefits/

Ein klassisches Projekt 1.



Ein klassisches Projekt 2.

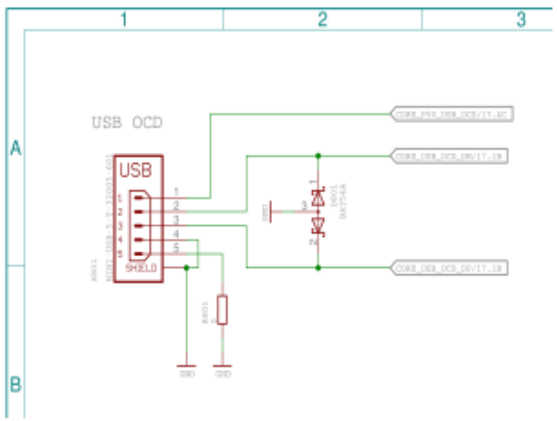
- Erfahrenes Team (2 HW-Entwickler, 2 FW-Entwickler, 1 Tester)
- Sehr gute Bedingungen (Resourcen, Kommunikation)

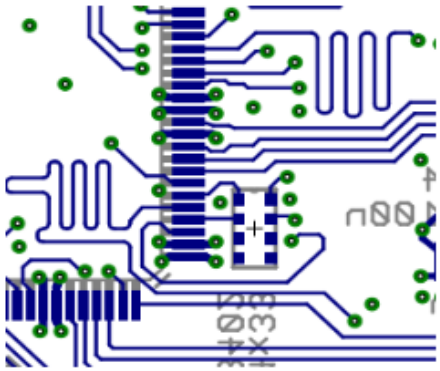
	Dauer der Projektphase	Gesamtdauer
Lastenheft	1 Monat	1 Monat
Entwicklung und Fertigung Null-Serie	6 Monate	7 Monate
SW-Integration und erstes erfolgreiches Ende-Zu-Ende Szenario	6 Monate	**13 Monate**

- Bereits nach 3 Monaten erste Change Requests – für Null-Serie aber bereits zu spät
- Folgerevisionen behoben nicht alle Fehler, verursachten aber neue

Kosten der HW-Entwicklung 1.

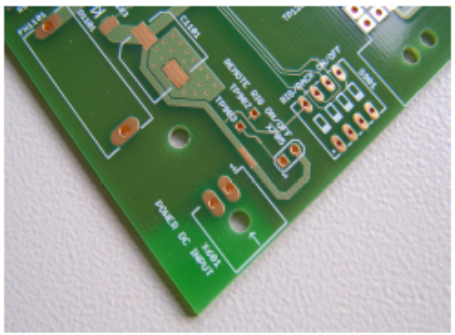
moderat





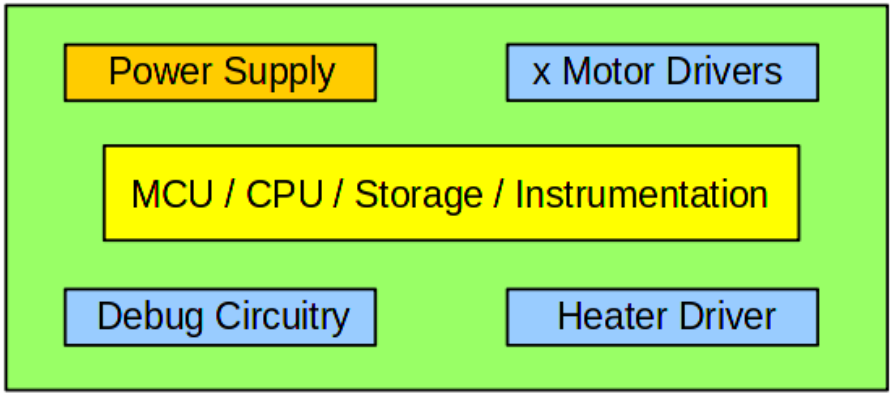
sehr hoch !

akzeptabel

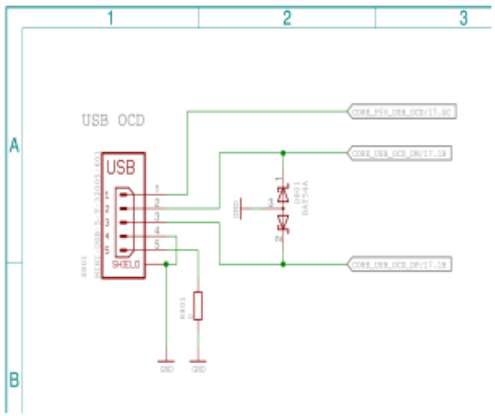


akzeptabel

Kosten der HW-Entwicklung 2.



EIN PCB für alles !

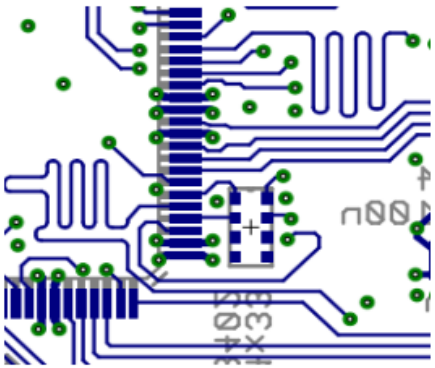


Alle Wünsche müssen in EIN Design !!!

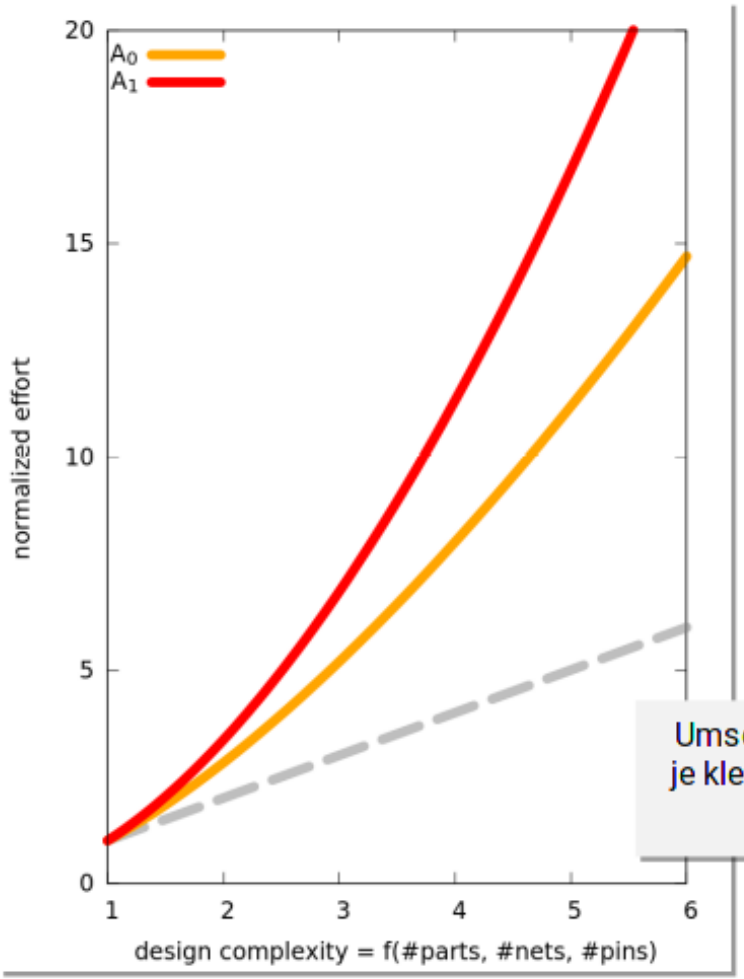
→ träge
→ teuer
→ zeitintensiv !

- Erweiterungen
- Änderungen
- Fehlersuche

Artwork !



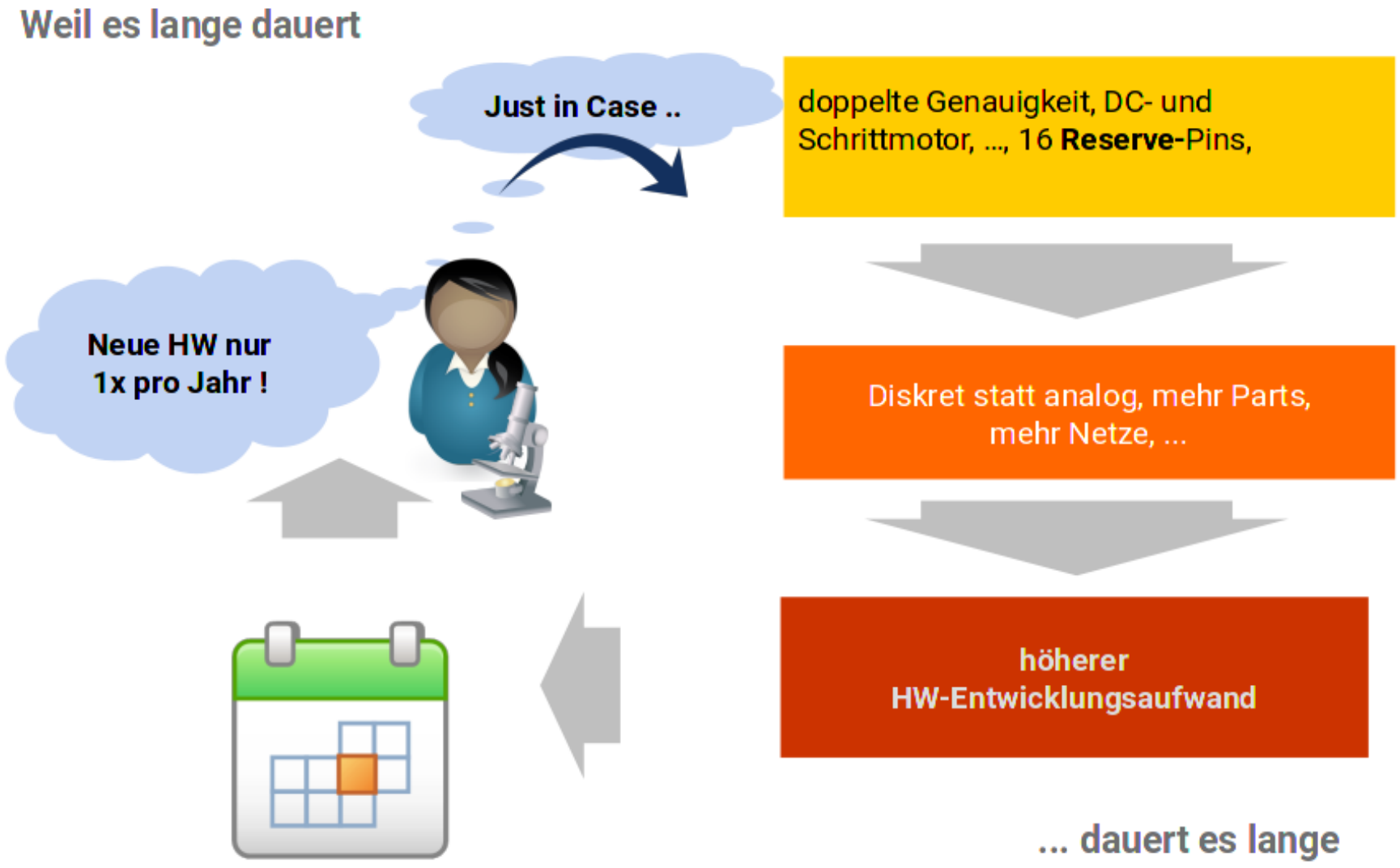
Kosten der Komplexität



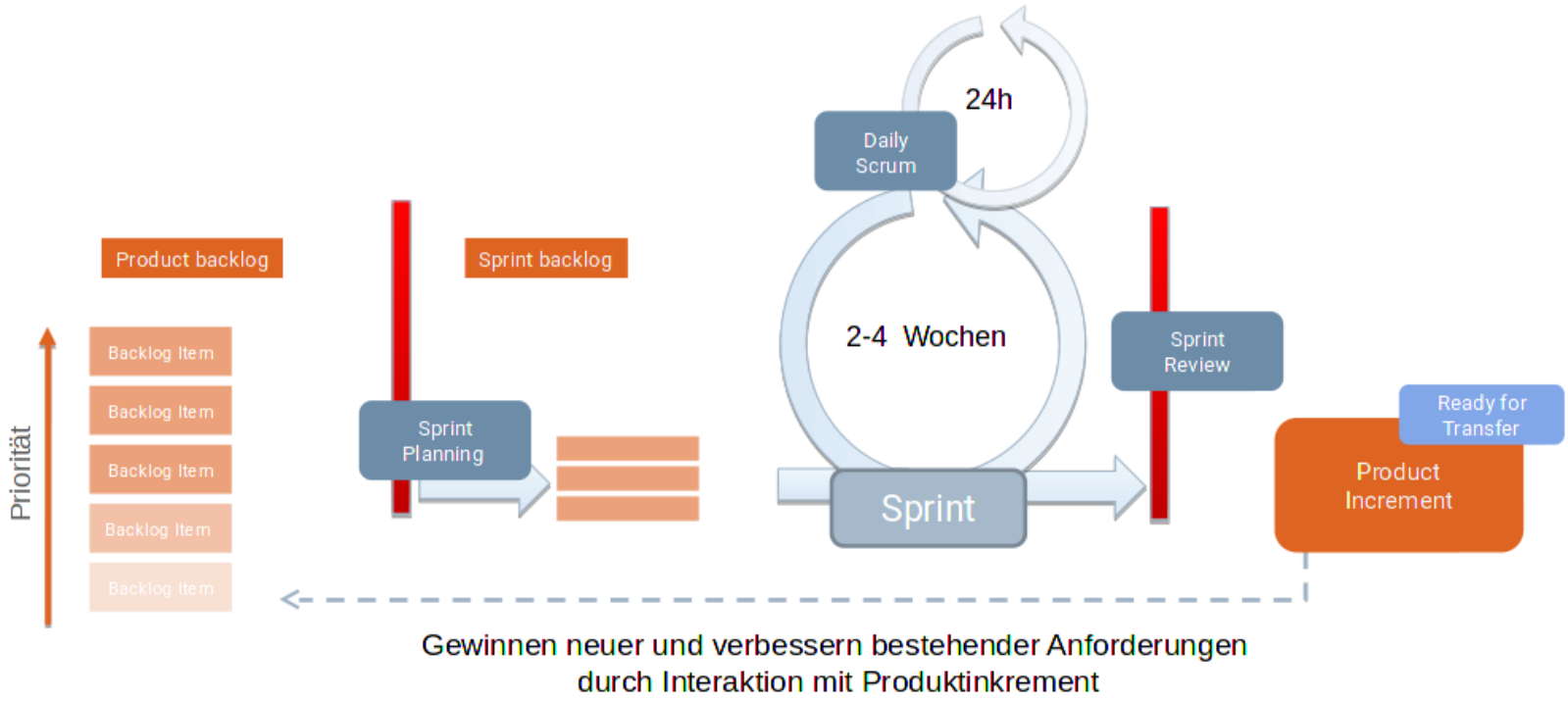
Umso dramatischer je kleiner die Fläche: $A_1 < A_0$

Design	# Parts	# Nets	# Pins
Very Complex	>1000	>1000	>2000
Complex	750	544	2250
Moderate	315	383	1162
Simple	121	101	350
Very Simple	<100	<70	<200

Weil es lange dauert ...



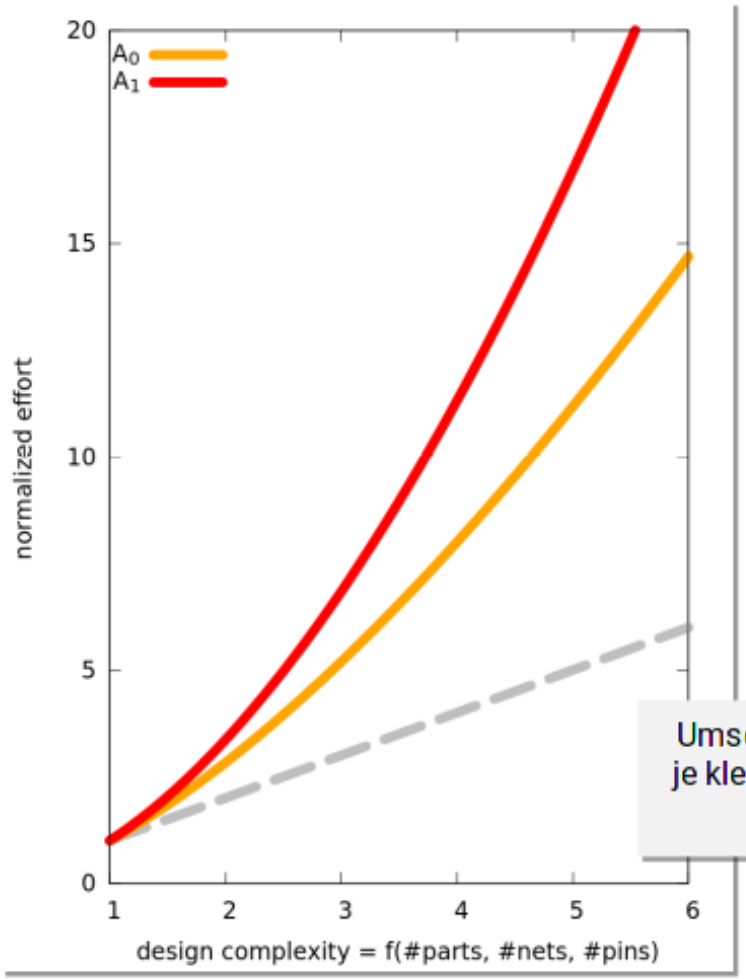
Scrum Framework



1. Ursprung in der SW-Entwicklung
2. Wir übertragen es auf HW-Entwicklung !

Agile Hardware

Kosten der Komplexität

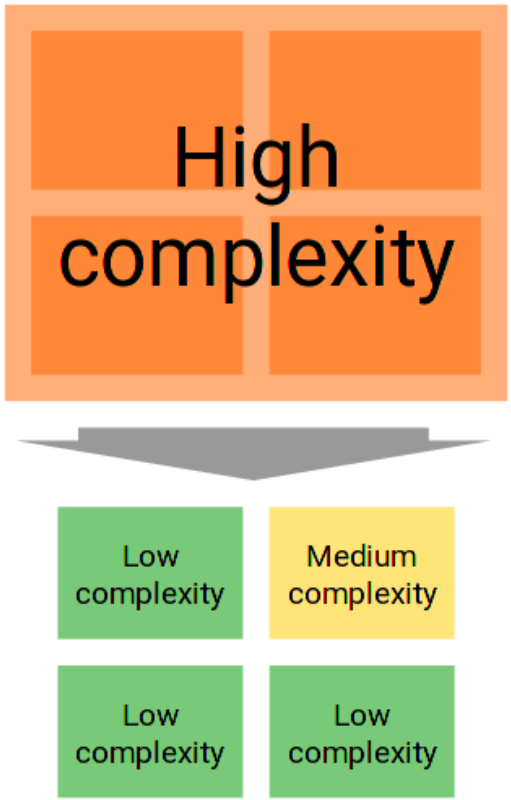


Design	# Parts	# Nets	# Pins
Very Complex	>1000	>1000	>2000
Complex	750	544	2250
Moderate	315	383	1162
Simple	121	101	350
Very Simple	<100	<70	<200

Umso dramatischer je kleiner die Fläche: $A_1 < A_0$

Agile Hardware

Aufbrechen in einfache und beherrschbare Teile

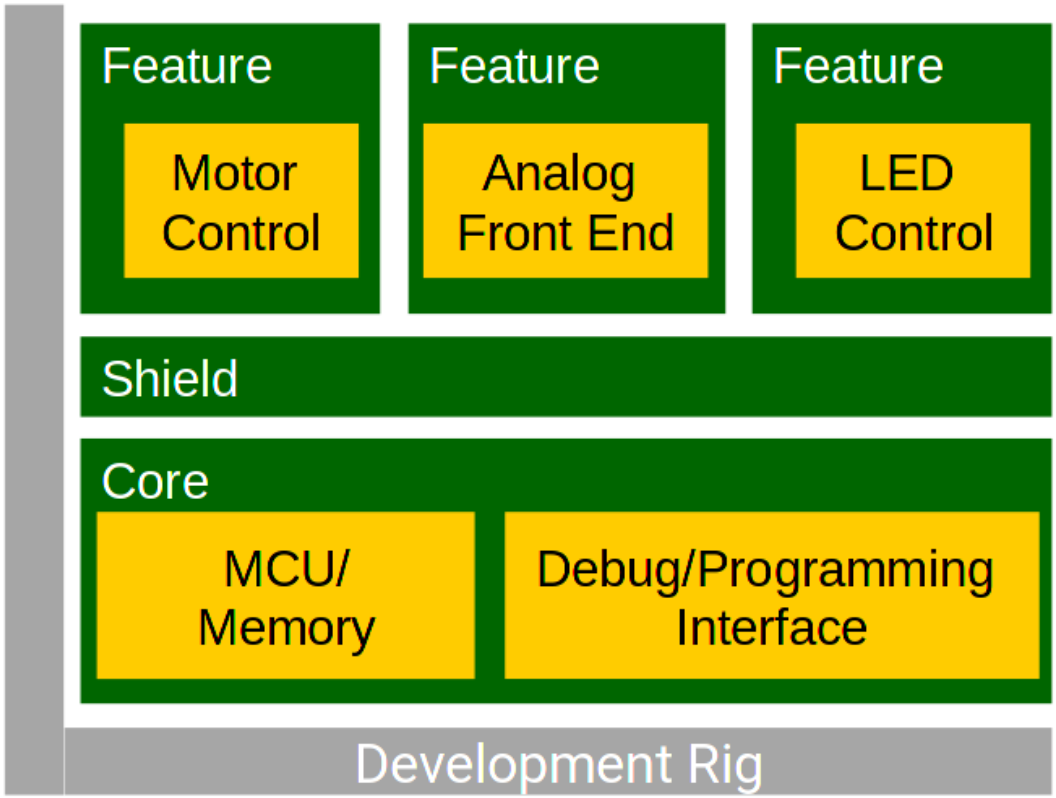


Achtung !

Kritische Schaltungsteile werden NICHT zerlegt !

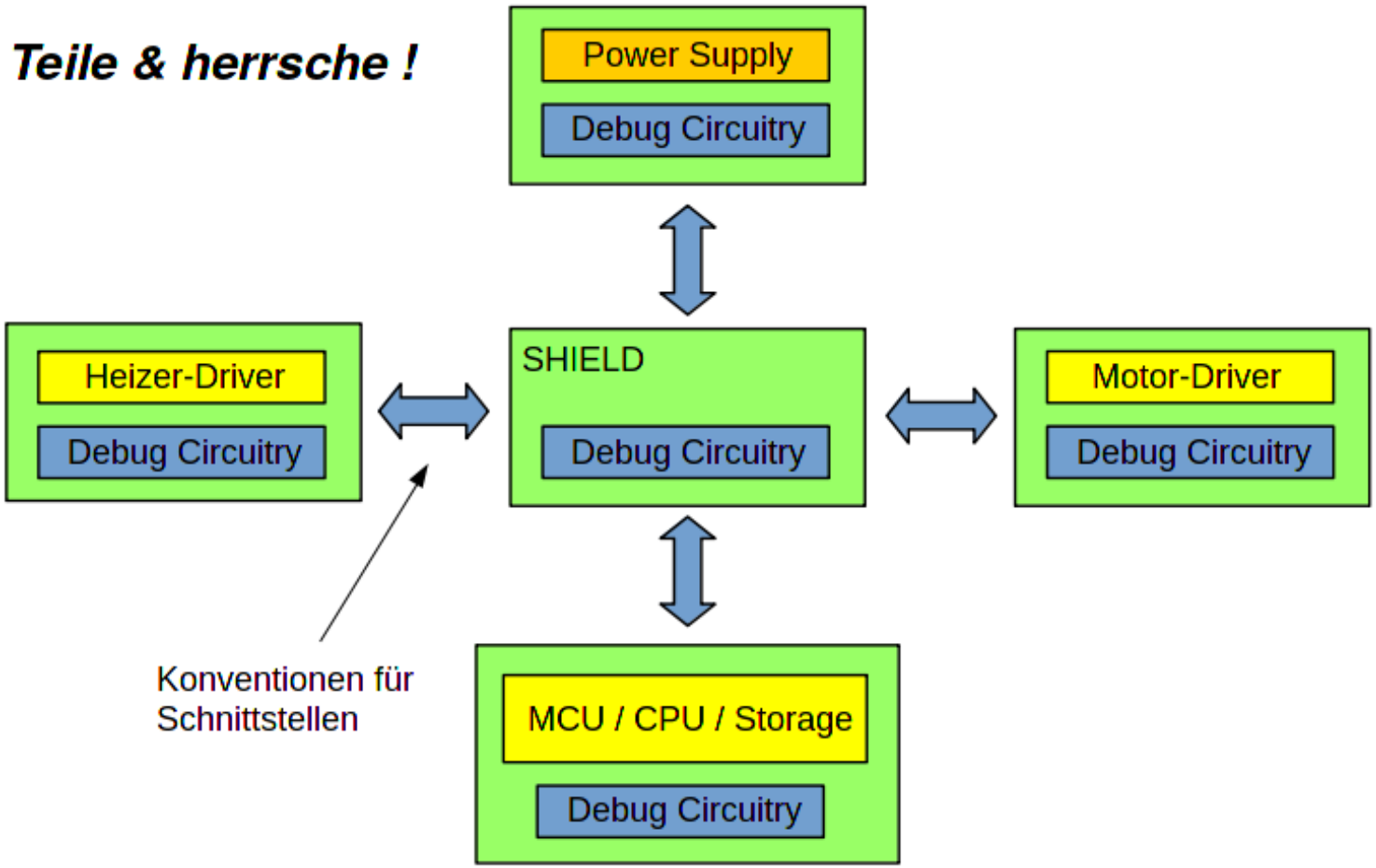
Agile Hardware

Rig-Design 1a



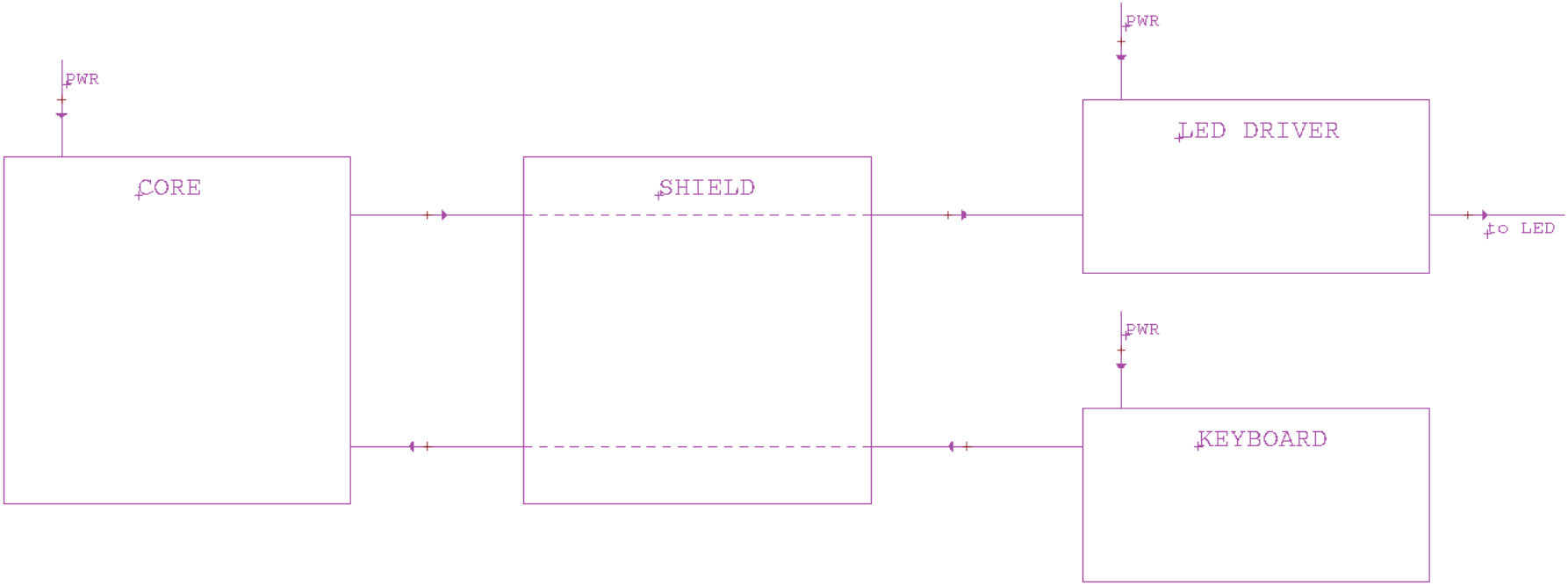
Agile Hardware

Rig-Design 1b



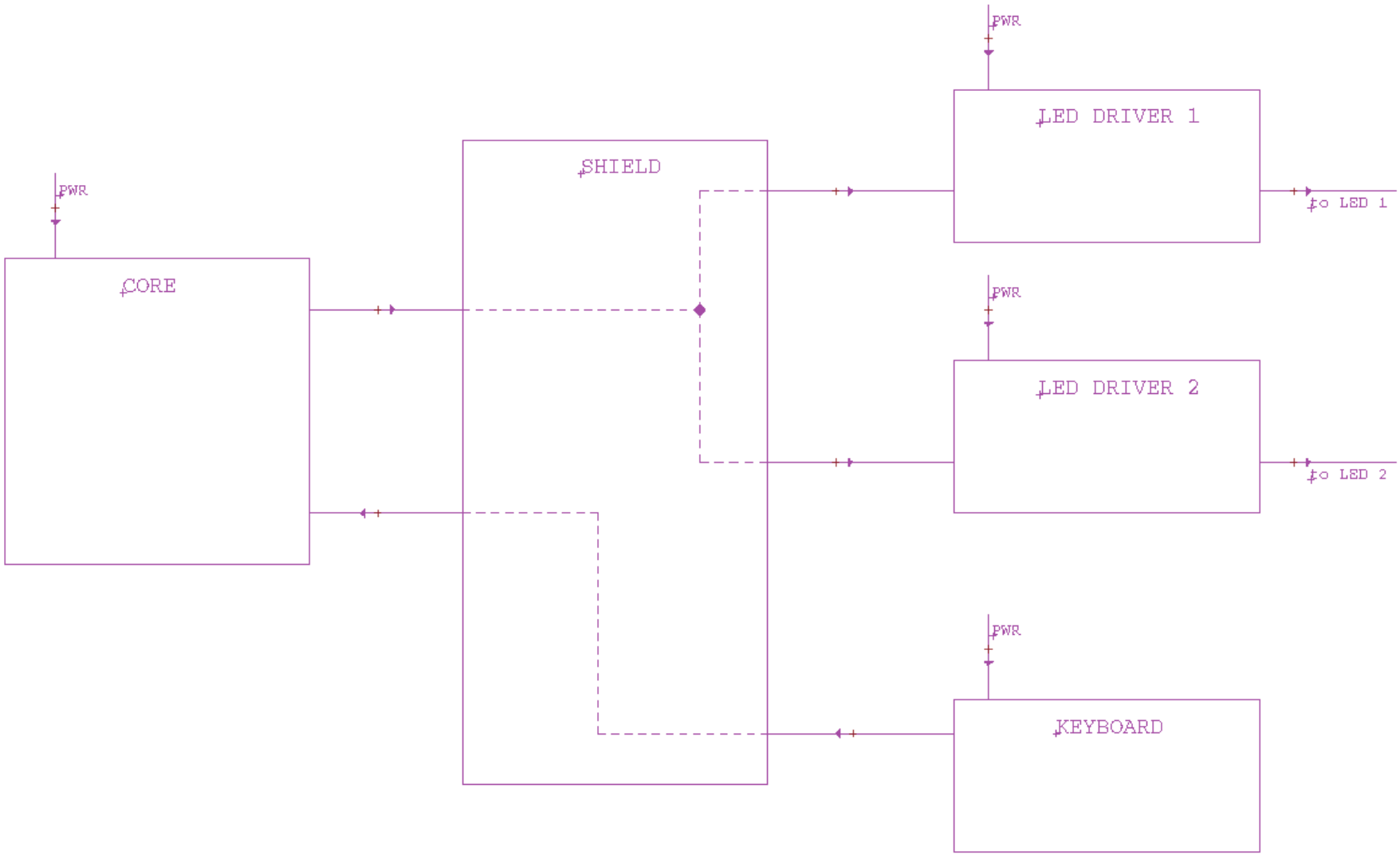
Agile Hardware

Rig-Design Blockschaltbild A



Agile Hardware

Rig-Design Blockschaltbild B



Agile Hardware

Vorteile und Möglichkeiten 1.

1. parallele Entwicklung
2. 2 Tage / Schaltplan
3. 3 Tage / PCB Layout
4. parallele Fertigung
5. großflächige PCBs erleichtern Entflechtung
6. niedrige Anforderungen an Fertigung (Preis !)
7. Verwendung des Autorouters spart Zeit (ca. 25%)
8. Wiederverwendbarkeit von Schaltungsteilen und PCBA
9. kurzfristige Änderungen möglich
10. frühzeitige Materialbeschaffung je PCB
11. paralleler Designcheck und Testgenerierung

Agile Hardware

Vorteile und Möglichkeiten 2. Autorouter

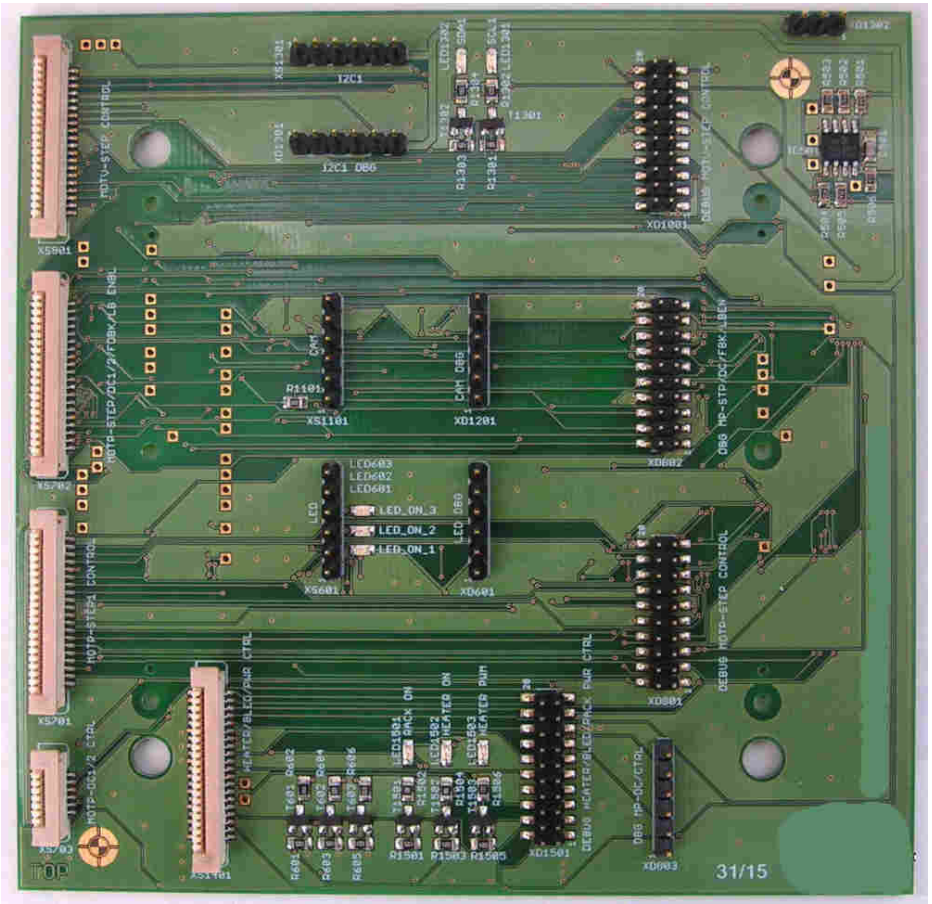
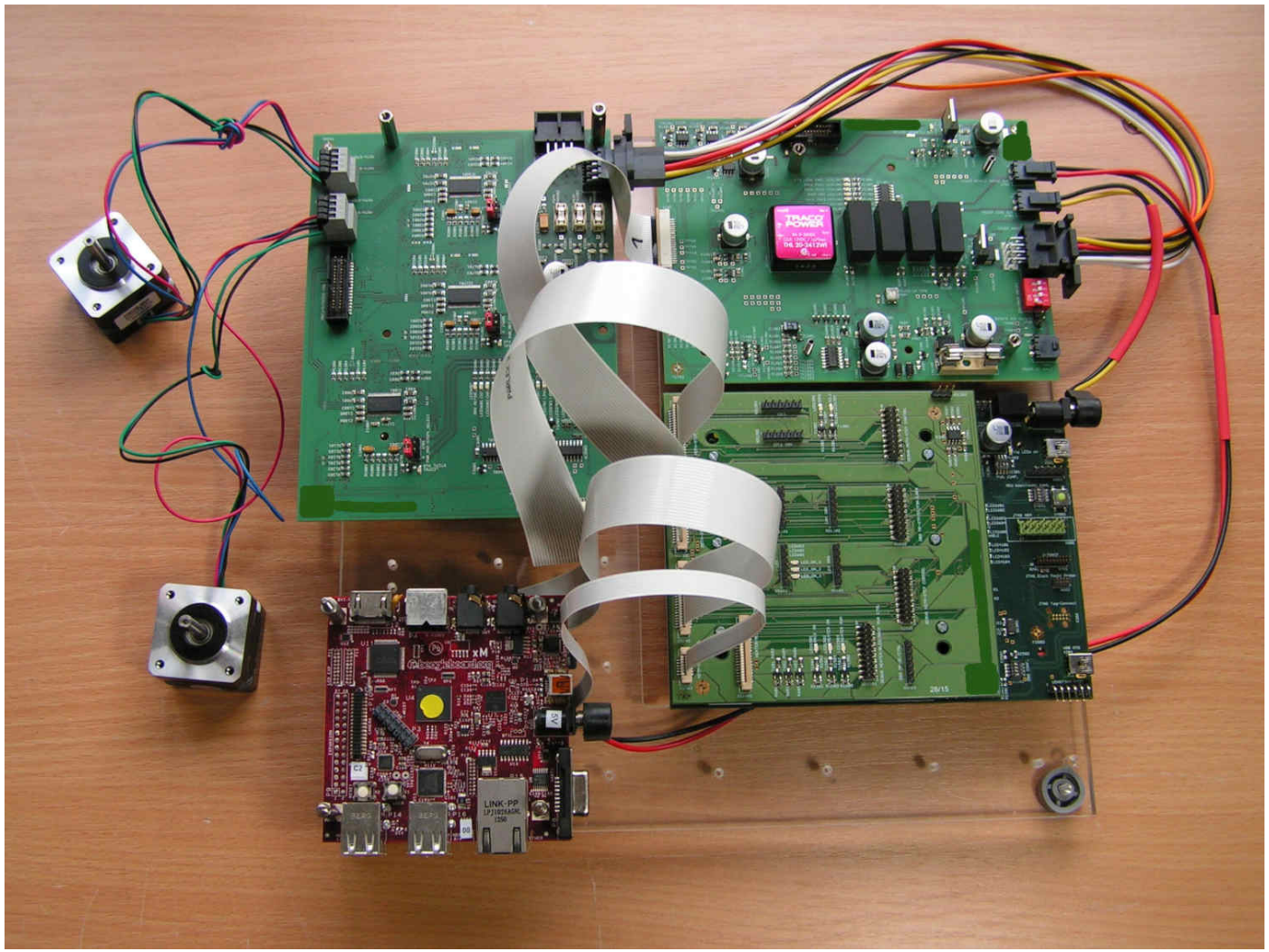


Figure 1: Shield

Agile Hardware

Beispiel eines Rigs



Anwendung der agilen Methode 1.

Wann anzuwenden ?

1. unscharfe, häufig wechselnde Anforderungen
2. Dokumentation und Verwaltung ufert aus (Lasten/Pflichtenhefte)
3. kurze Entwicklungszeiten (Sprint 2 bis 4 Wochen) gefordert

Vorteile:

1. Projekt läßt sich besser steuern gegenüber Wasserfallmodell
2. Rückmeldung vom Kunden nach einem Sprint
3. weniger Überraschungen bei Auslieferung des Produktes
4. SW bleibt konstant von Prototyp bis Auslieferung
5. für SW/FW-Entwicklung bleiben Ports von MCU, FPGA, CPLD konstant

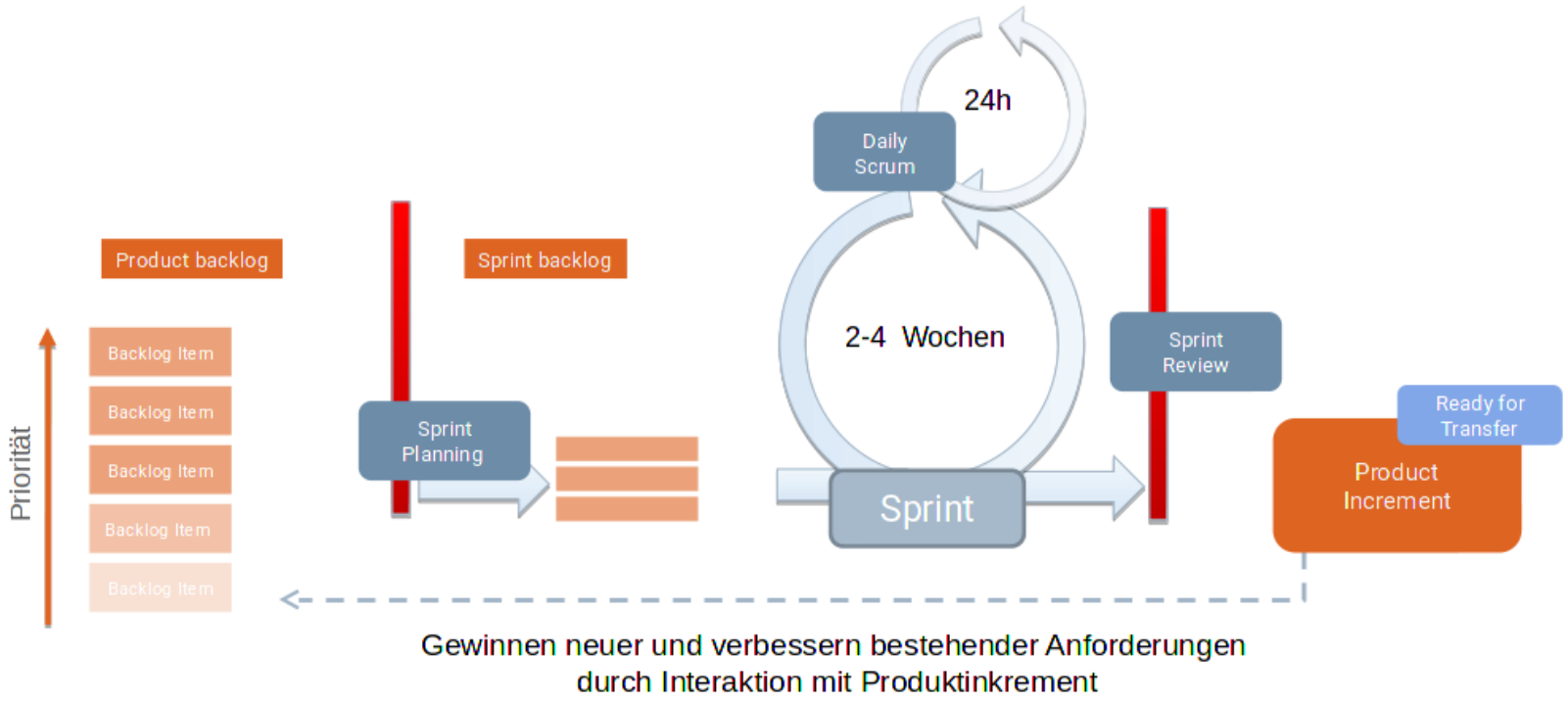
Anwendung der agilen Methode 2.

Zusammenfassung für Manager

1. Das agile Entwicklungsmodell ist keine Wunderwaffe !
2. Projekte mit häufig wechselnden oder unscharfen Anforderungen lassen sich damit sehr gut steuern. $\rightarrow$ weniger böse Überraschungen
3. Agile Methoden machen bei klar beschriebenden Anforderungen keinen Sinn. $\rightarrow$ Wasserfallmodell anwenden

Vorgehensmodell

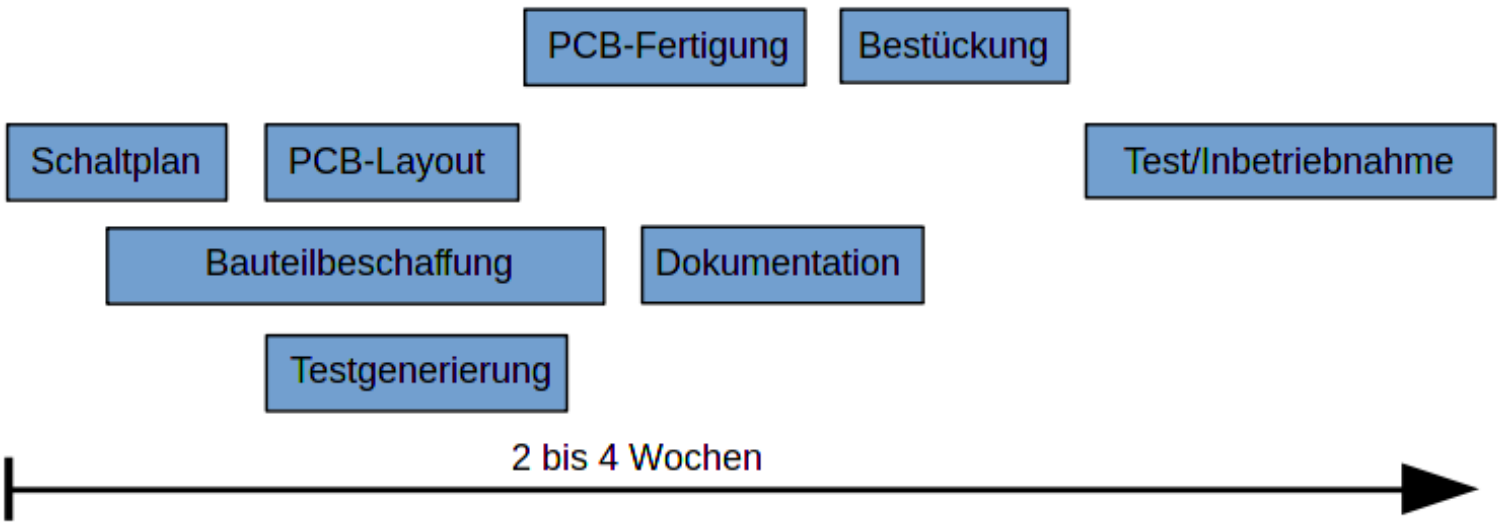
ein Sprint 1.



Vorgehensmodell

ein Sprint 2.

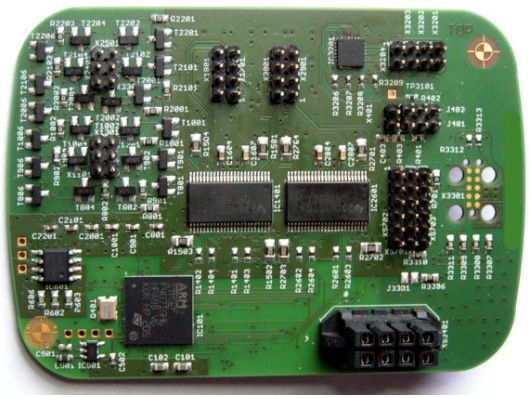
1. Design neuer Rig-PCBAs: Ein Feature / eine Funktion pro Modul
2. Modifikation oder Überarbeitung bestehender Module
3. Herstellung und Inbetriebnahme der einzlnen Rig-PCBAs



Vorgehensmodell

Ableitung des Produktlayouts aus Rig-Design (nach N Sprints)

1. einzelne PCBA werden im Verbund getestet
2. Schaltpläne vereinen (merge)
3. Schaltplan reduzieren auf produktrelevante Schaltungsteile
4. Layout für Produkt erstellen $\rightarrow$ "darf lange dauern"



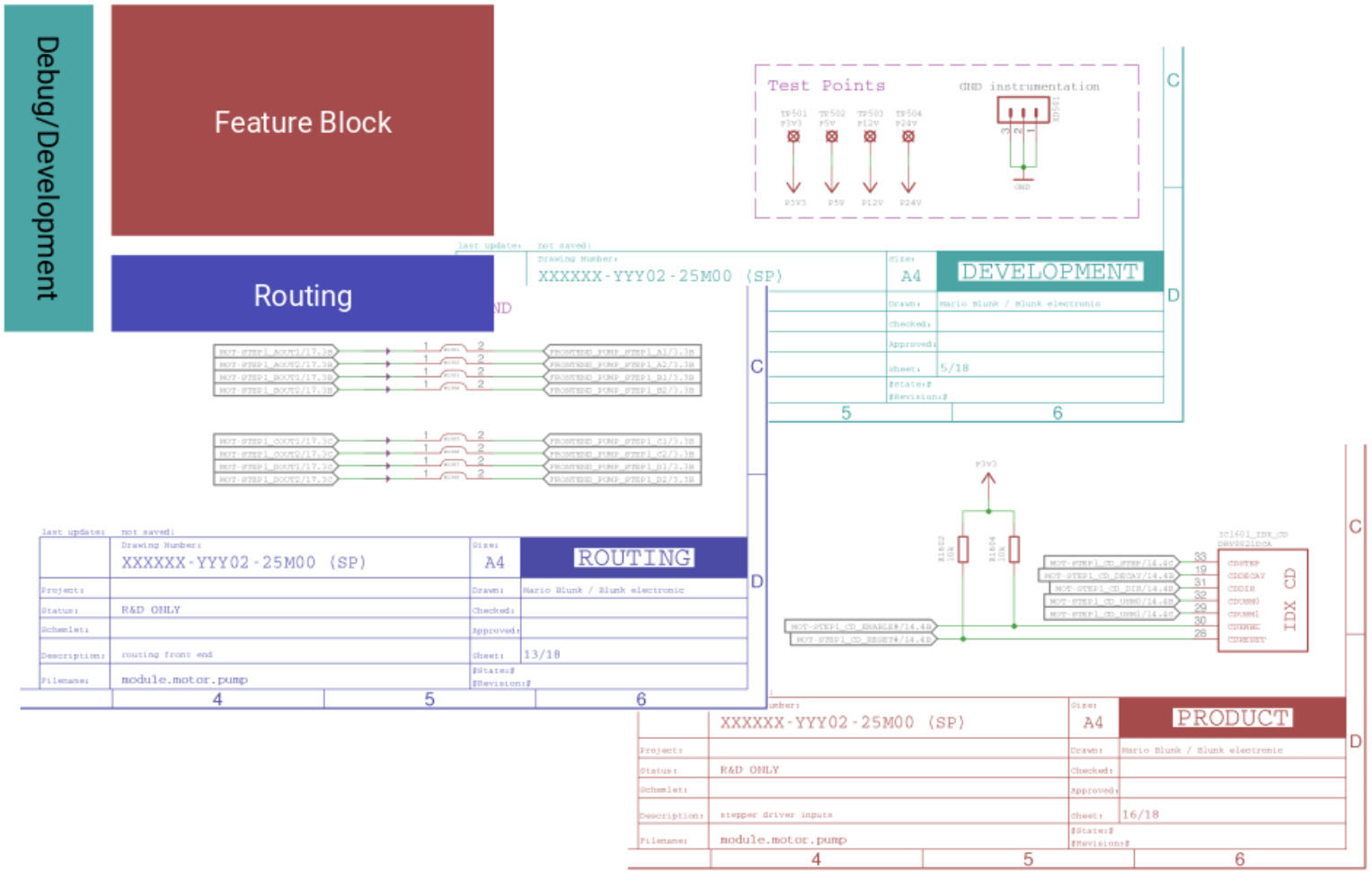
Agile HW - Werkzeuge

CAE- Werkzeuge müssen automatisierbar sein (skriptbar):

1. Löschen von Schaltplanseiten, Bauteilen und Netzen
2. Umbenennen von Netzen
3. Feststellung Verstöße gegen Konventionen
4. Ergebnis von ERC/DRC in ASCII-Datei
5. Erzeugung von Materiallisten, CAM-Daten
6. Designdaten in Klartext, ASCII, XML (nicht binär !)
7. Dokumentation in Klartext (Sphinx, Latex)
8. automatisierte Testgenerierung (z.B. Boundary-Scan)
9. Versionskontrolle (z.B. mit Git)
10. erprobt mit Autodesk-EAGLE
11. CAE System ET (in Entwicklung)
    https://github.com/Blunk-electronic/ET

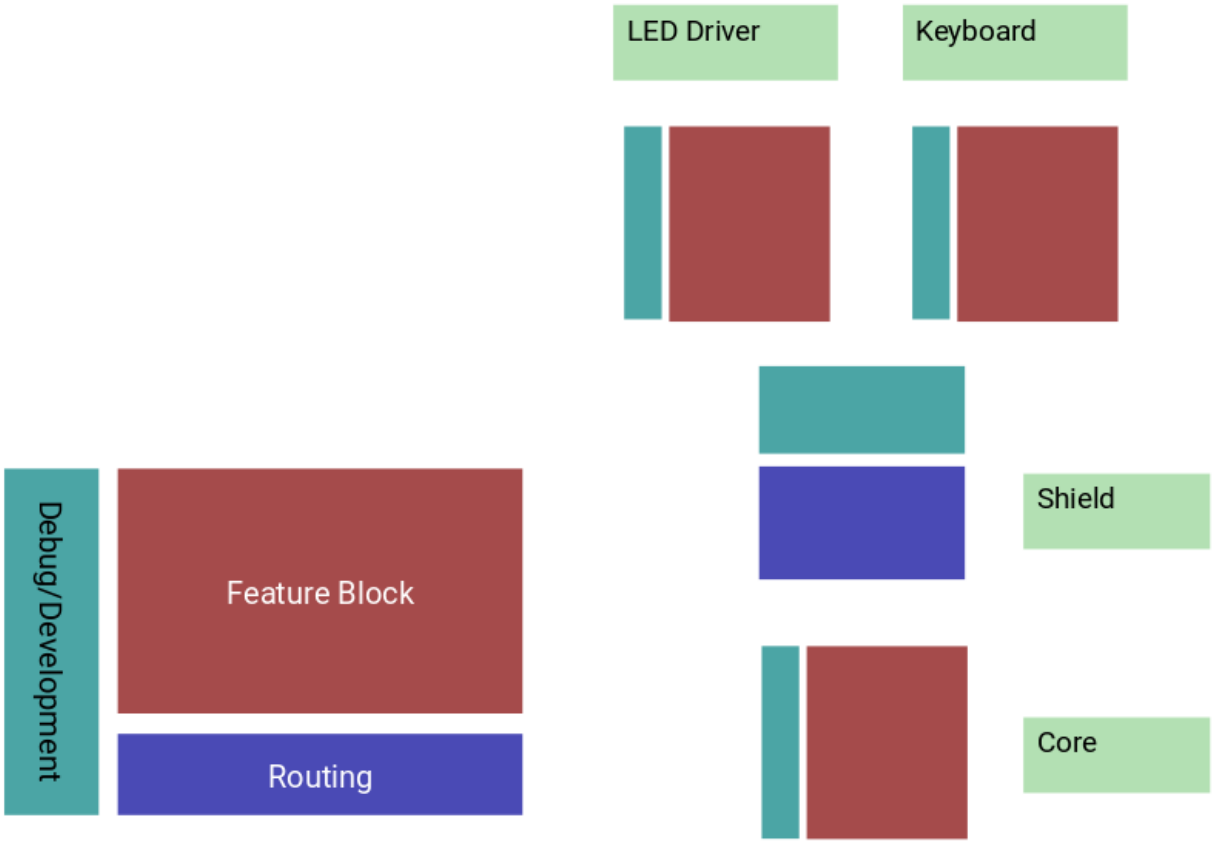
Agile HW - Entwicklung

Struktur der Schaltpläne 1.



Agile HW - Entwicklung

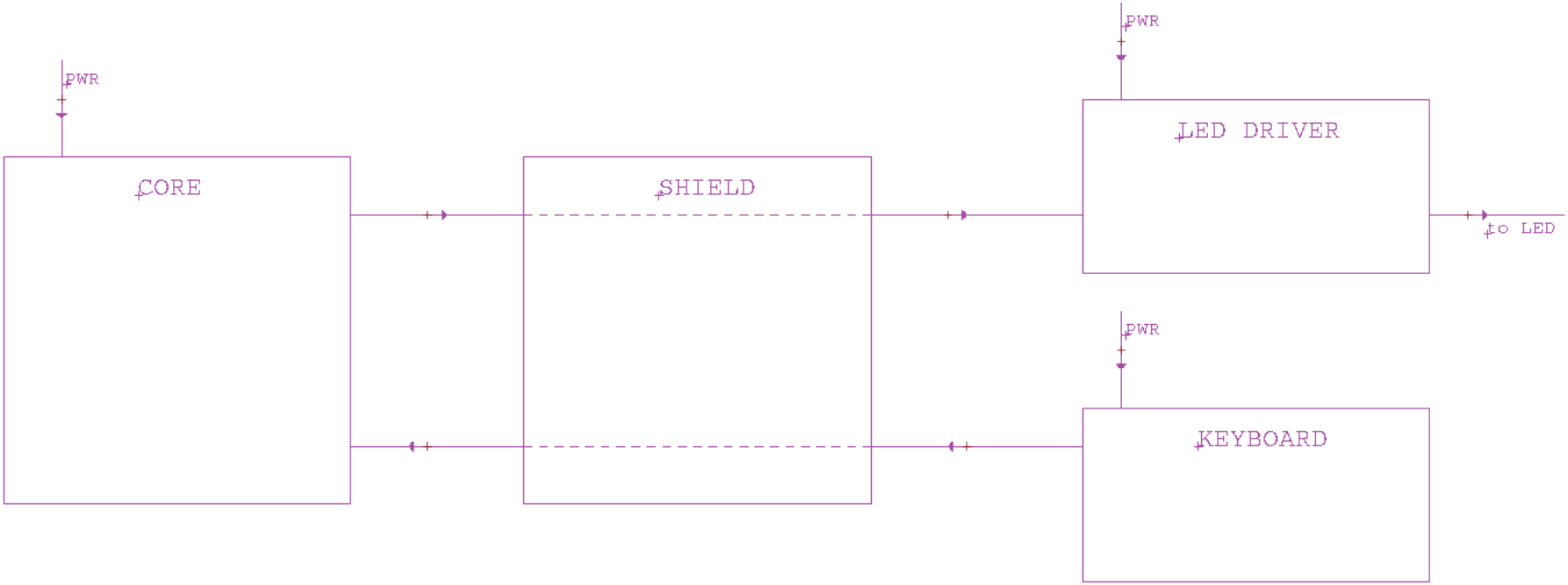
Struktur der Schaltpläne 2.



Schaltpläne:
http://www.blunk-electronic.de/pdf/agile_HW/demo/

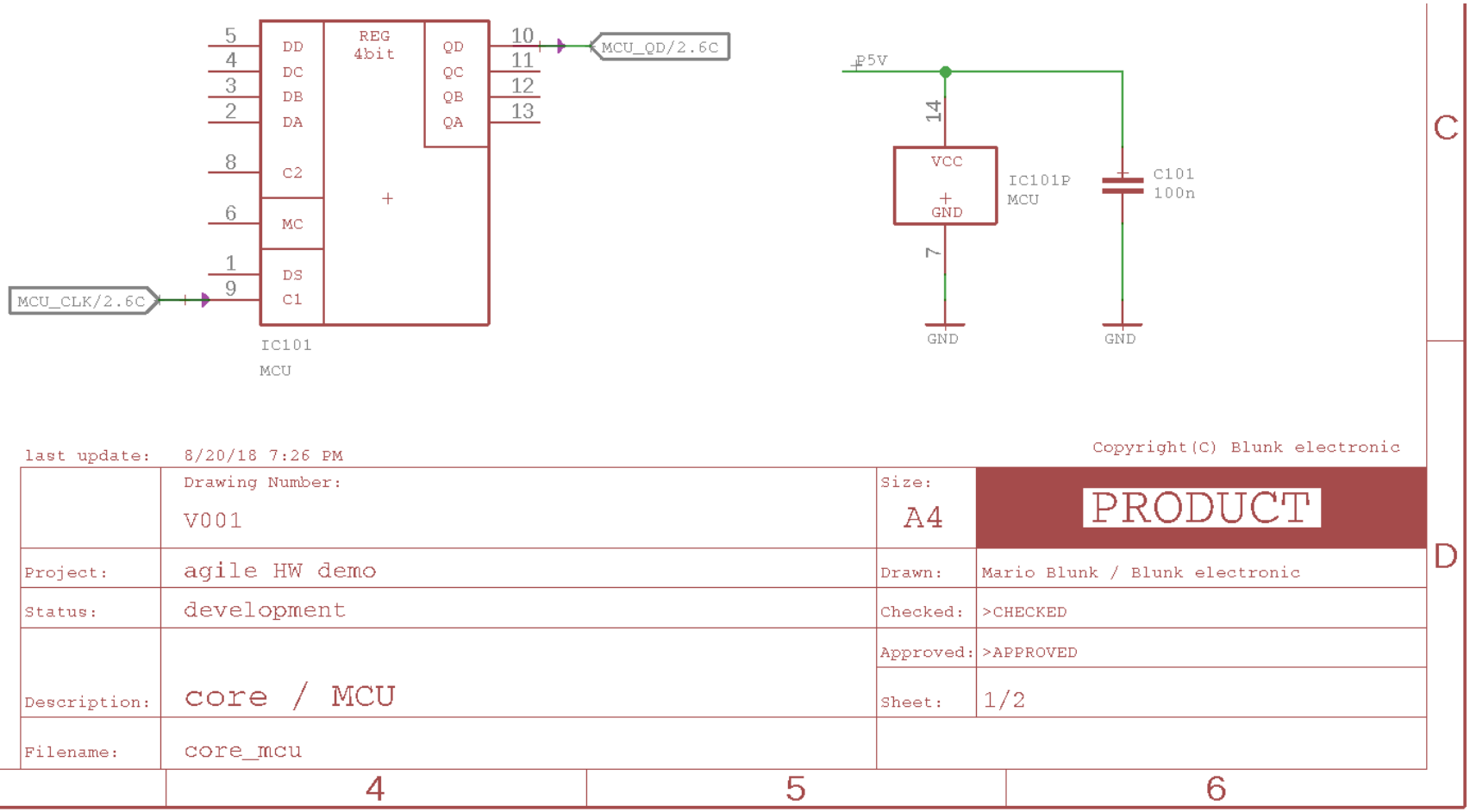
Agile HW - Entwicklung

Blockschaltbild eines Demo-Projektes



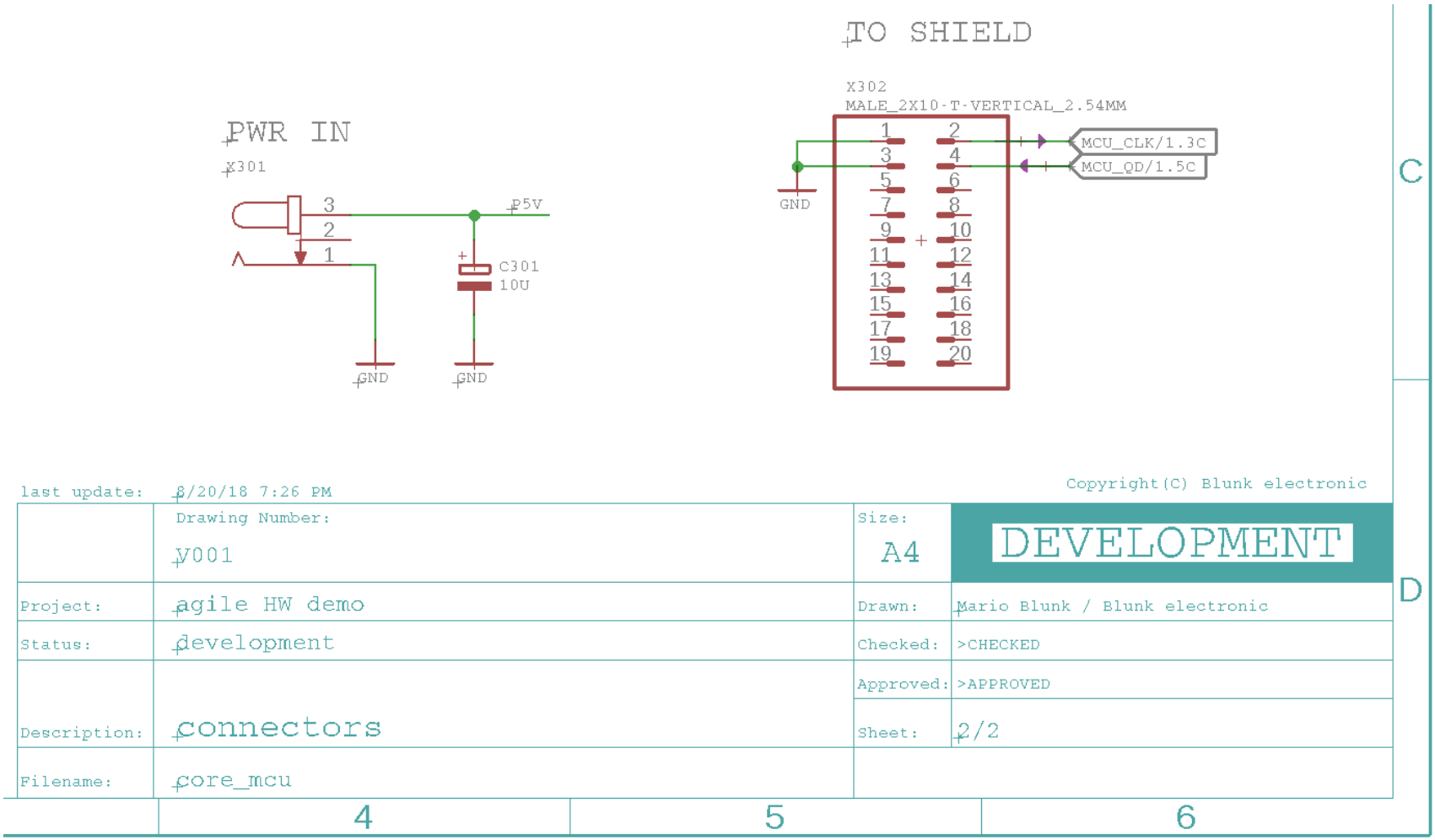
Demo Projekt

Core Modul - produktrelevante Seite



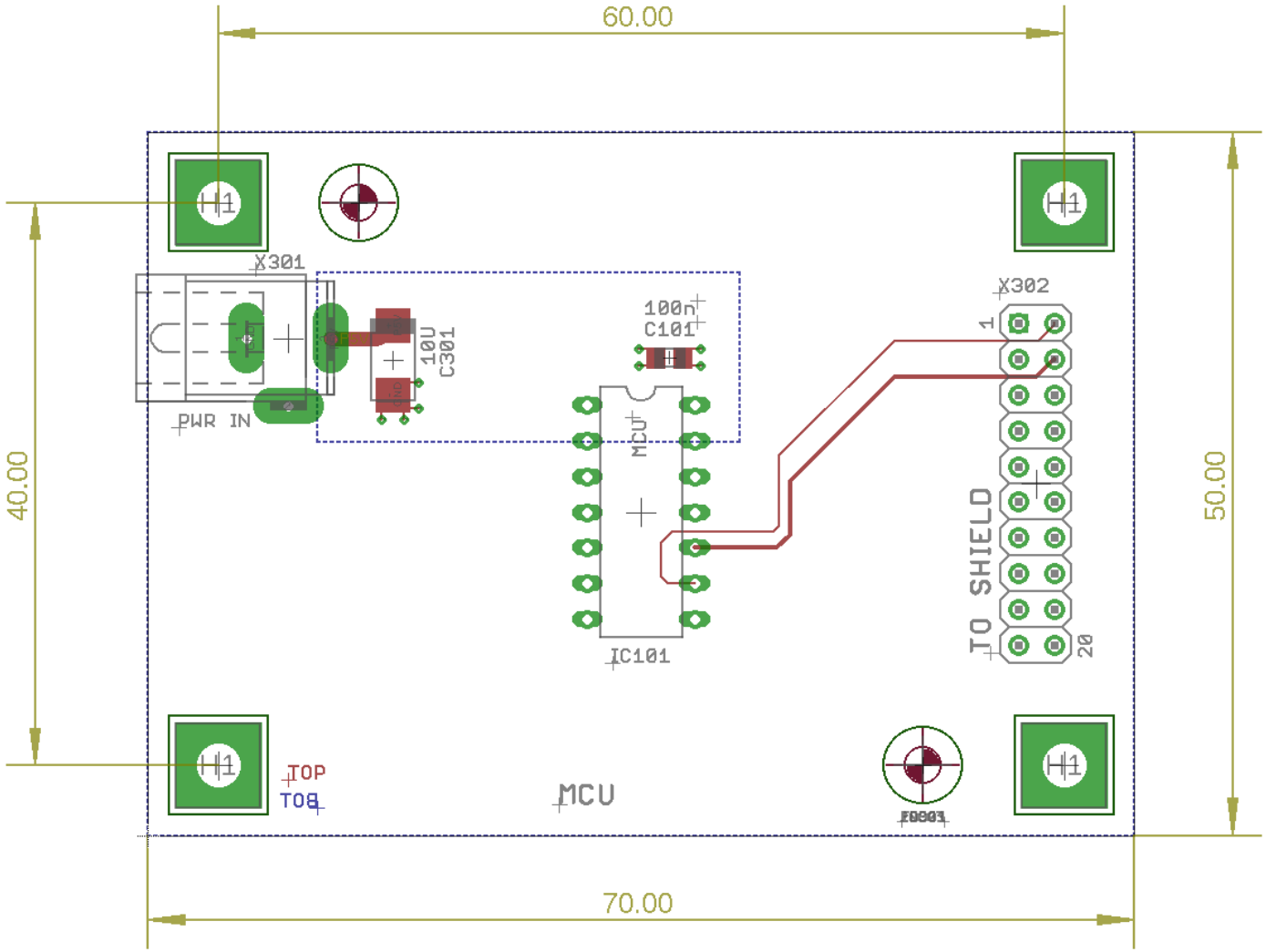
Demo Projekt

Core Modul - Seite für Entwicklung



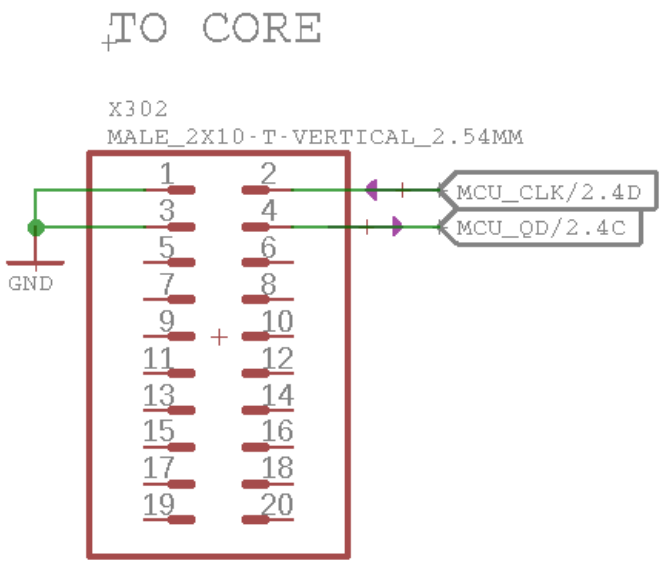
Demo Projekt

Core Modul - PCB Layout



Demo Projekt

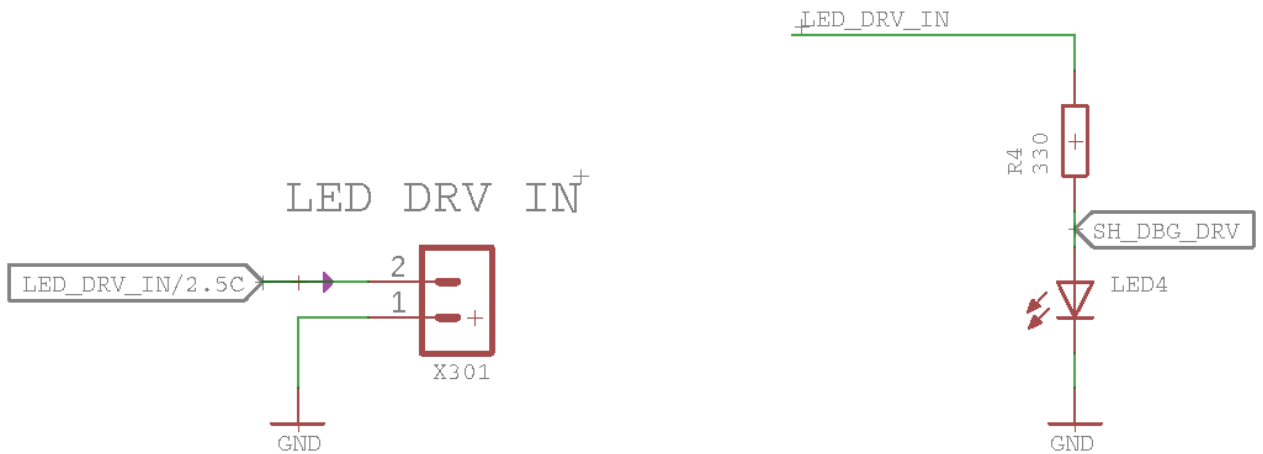
Shield Modul – Seite für Entwicklung 1.



last update: 8/20/18 7:26 PM			
	Drawing Number: V001	Size: A4	DE
Project: agile HW demo		Drawn:	Mario Blunk

Demo Projekt

Shield Modul – Seite für Entwicklung 2.



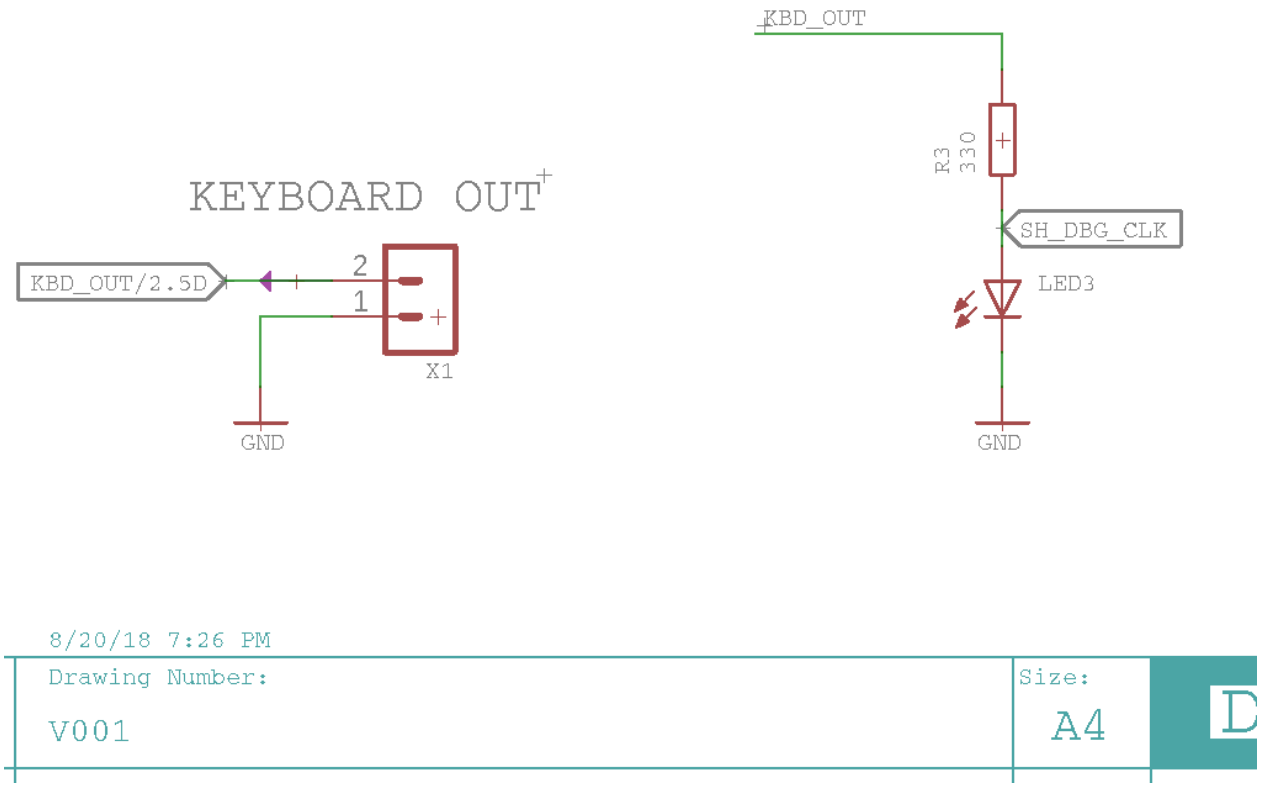
8/20/18 7:26 PM

Copyright (C) Blu

Drawing Number:	Size:	DEVELOPM
V001	A4	
agile HW demo	Drawn:	Mario Blunk / Blunk electro

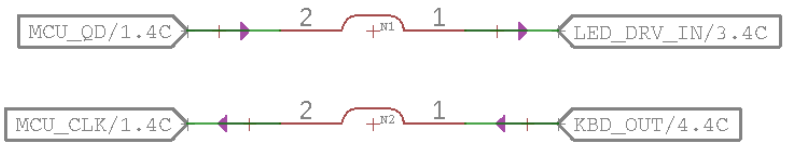
Demo Projekt

Shield Modul – Seite für Entwicklung 3.



Demo Projekt

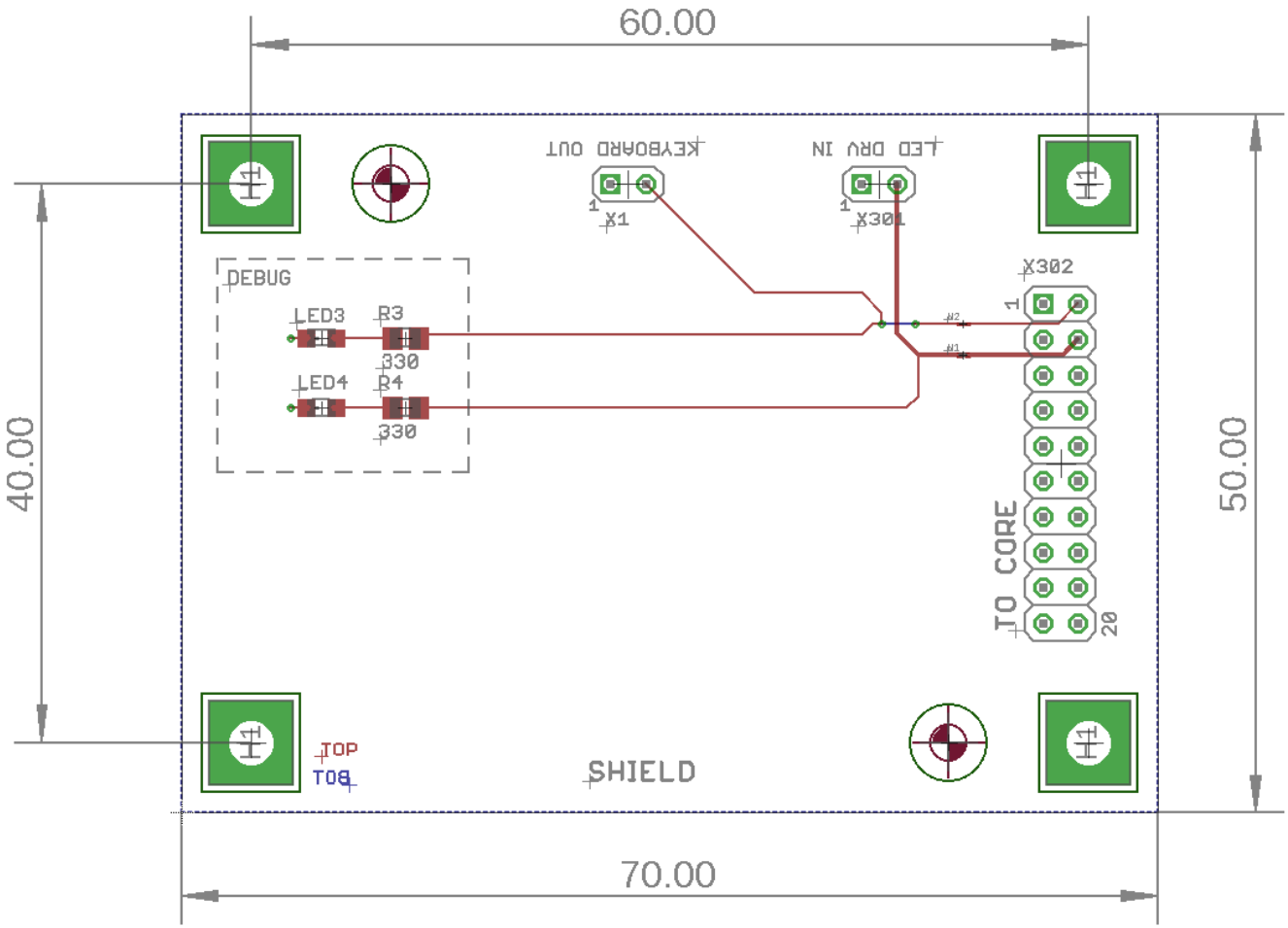
Shield Modul – Seite für Entwicklung 4.



last update: 8/20/18 7:26 PM			
	Drawing Number: V001	Size: A4	ROUTING
Project:	agile HW demo	Drawn:	Mario Blunk / Blunk electronic

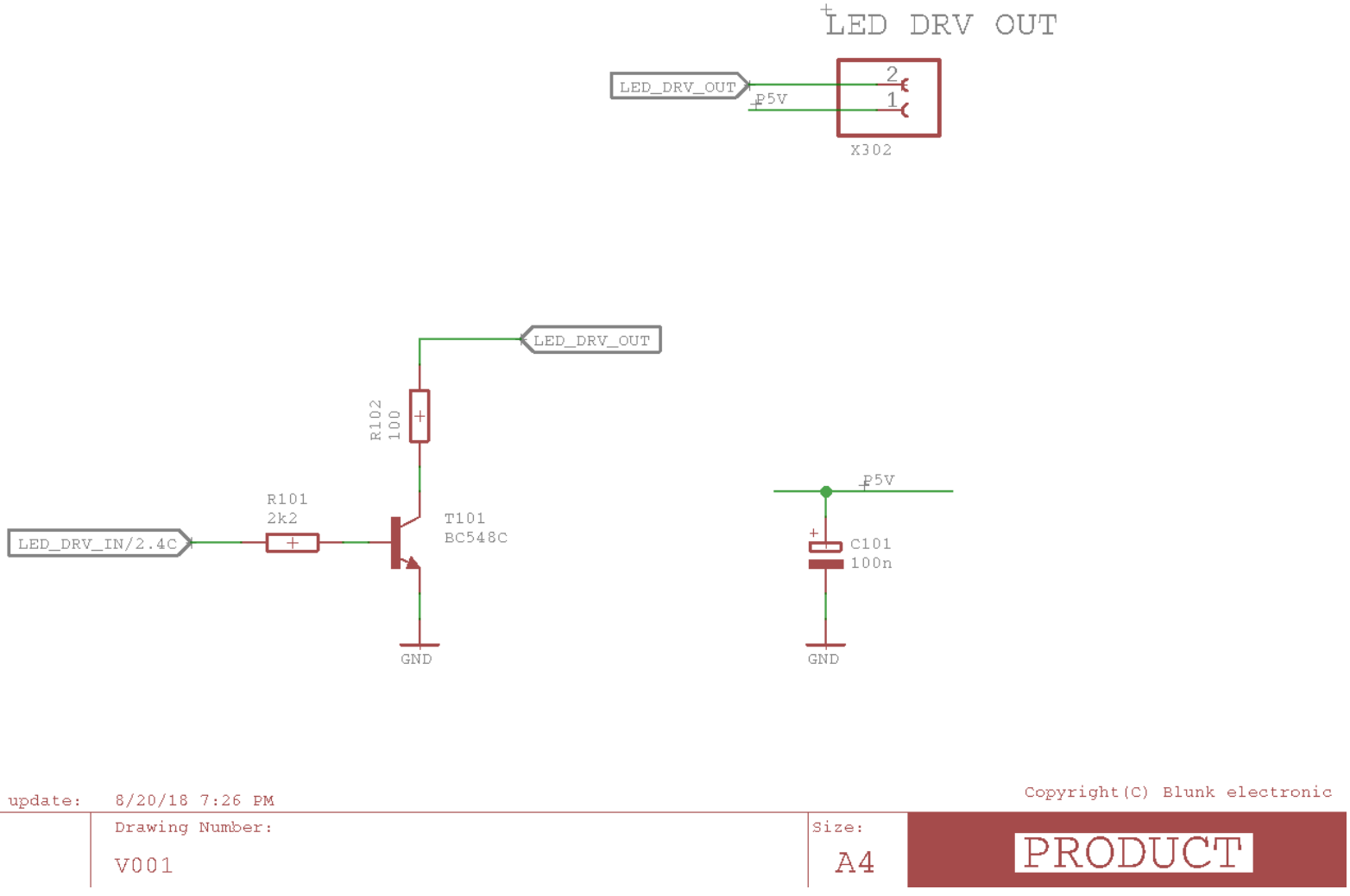
Demo Projekt

Shield Modul – PCB Layout



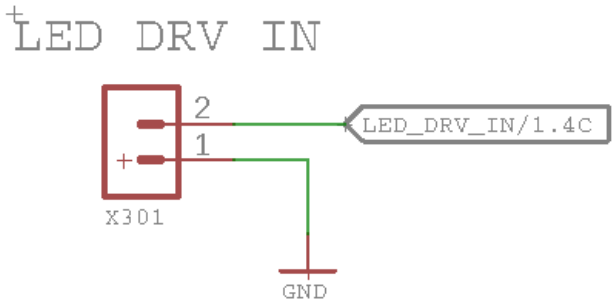
Demo Projekt

LED-Treiber Modul – produktrelevante Seite



Demo Projekt

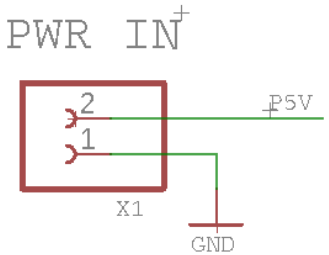
LED-Treiber Modul – Seite für Entwicklung 1.



last update: 8/20/18 7:26 PM			Co
	Drawing Number: V001	Size: A4	DEV
Project:	agile HW demo	Drawn:	Mario Blunk

Demo Projekt

LED-Treiber Modul – Seite für Entwicklung 2.



last update: 8/20/18 7:26 PM

Co

Drawing Number:

V001

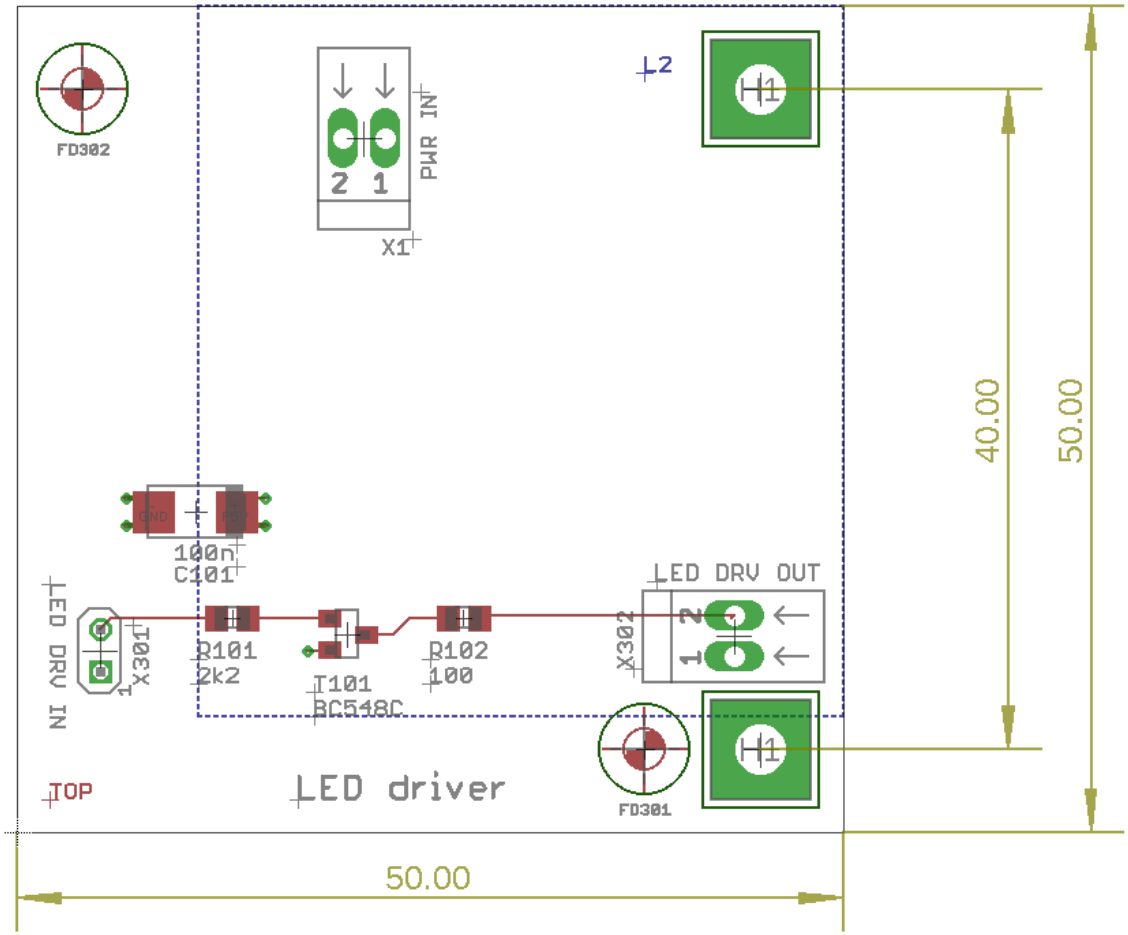
Size:

A4

DEV

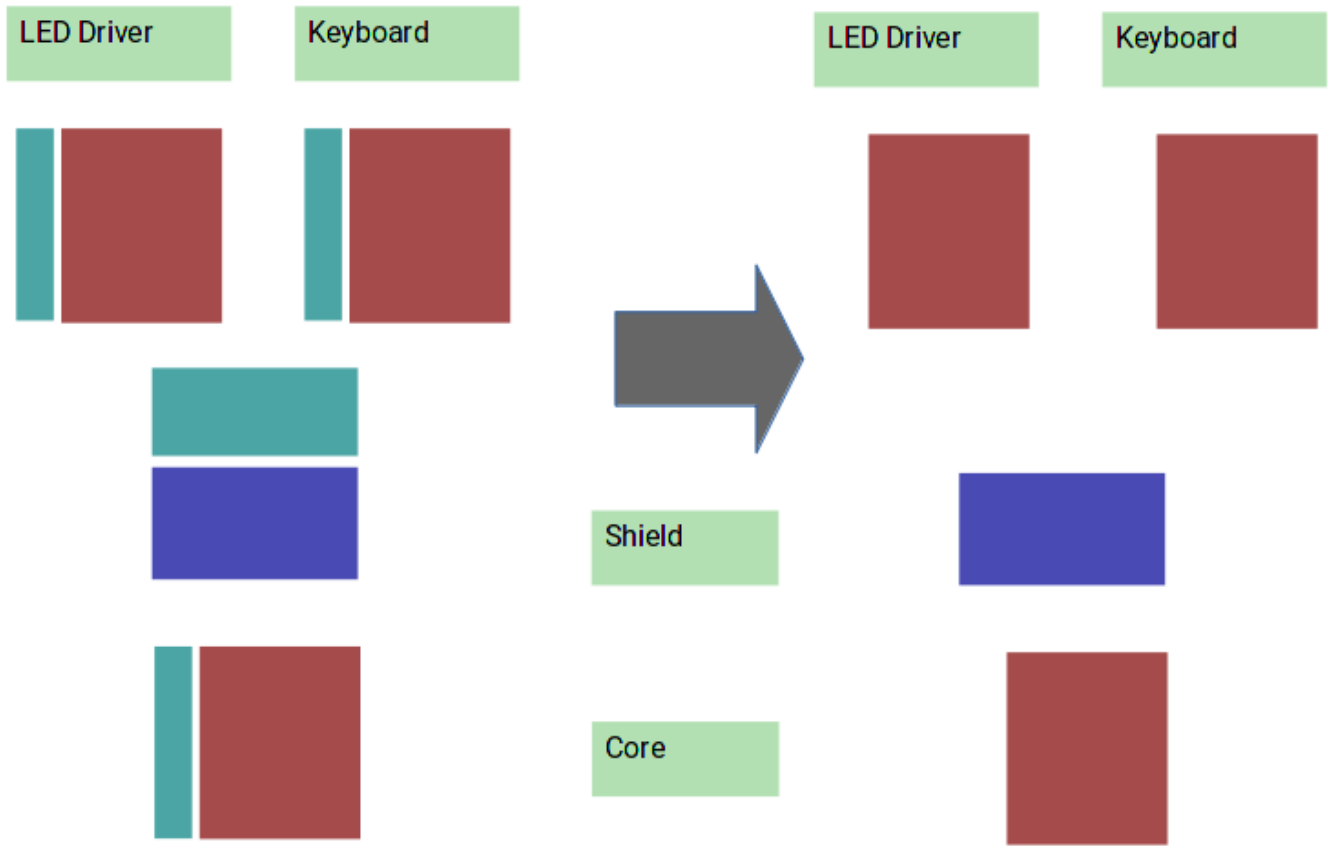
Demo Projekt

LED-Treiber Modul – PCB Layout



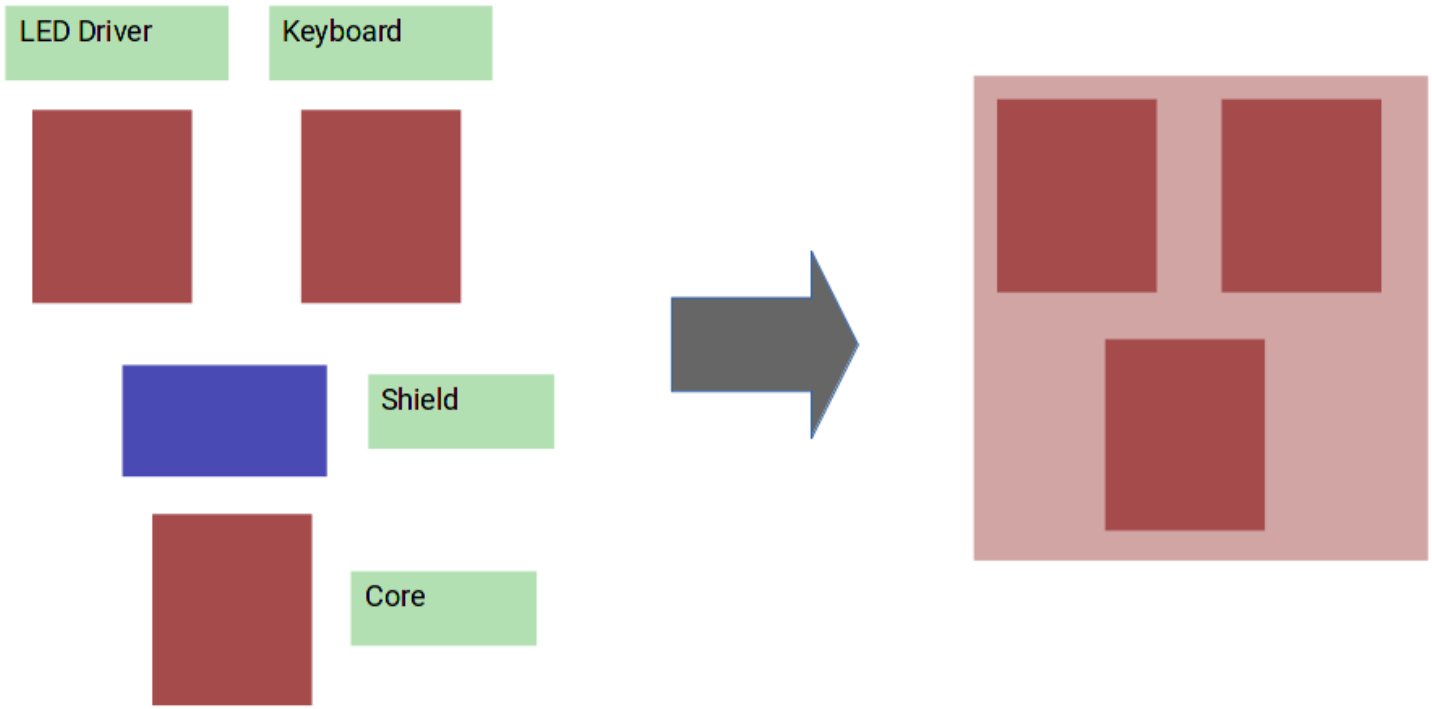
Demo Projekt

Schaltpläne vereinen und reduzieren 1.



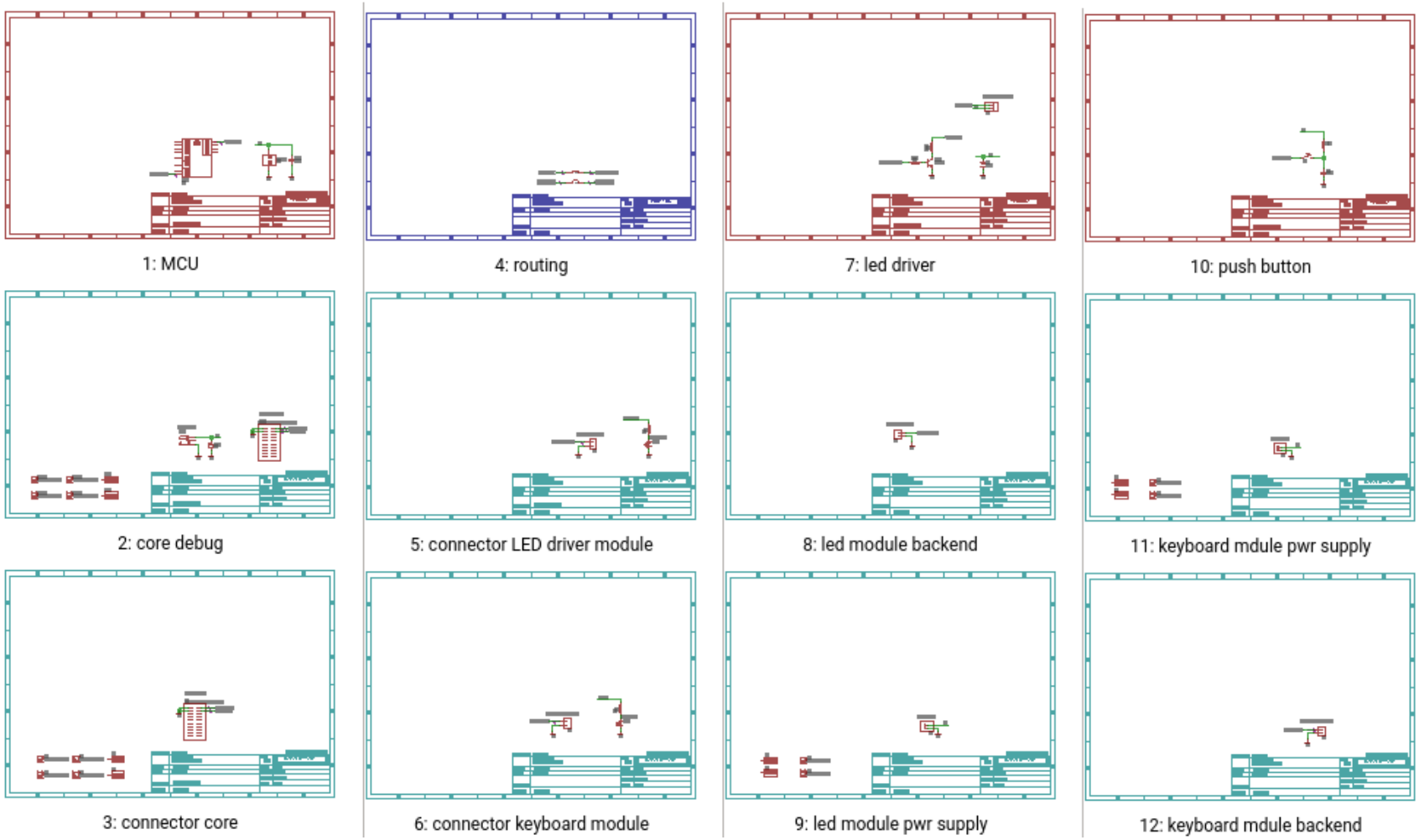
Demo Projekt

Schaltpläne vereinen und reduzieren 2.



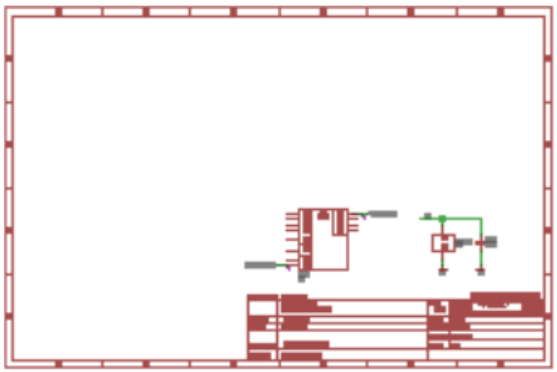
Demo Projekt

Schaltpläne vereinen und reduzieren 3.

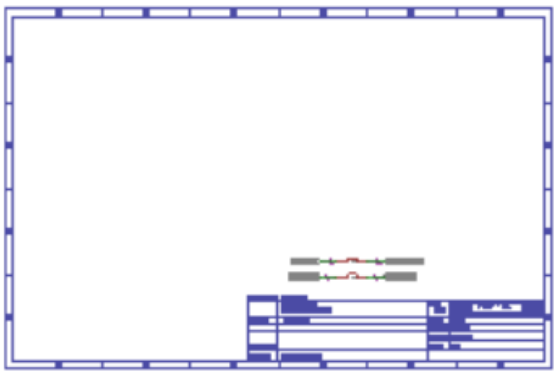


Demo Projekt

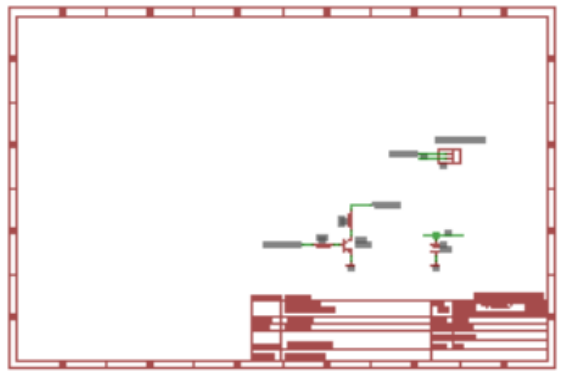
Schaltpläne vereinen und reduzieren 4.



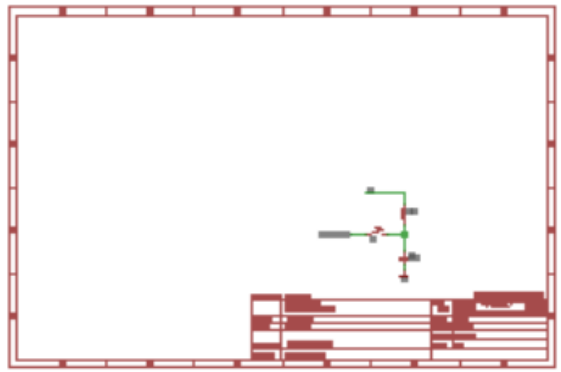
1: MCU



2: routing



3: led driver

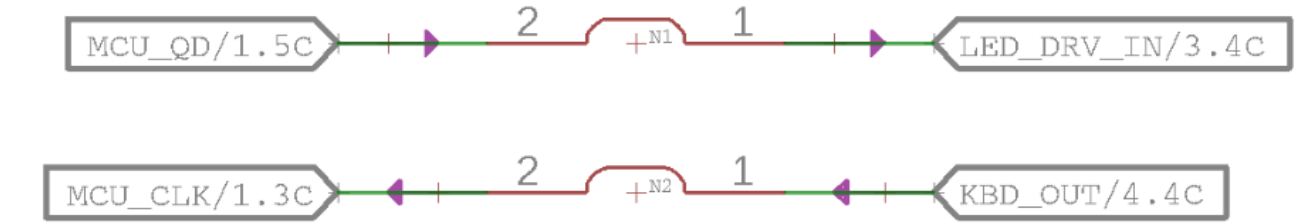


4: push button

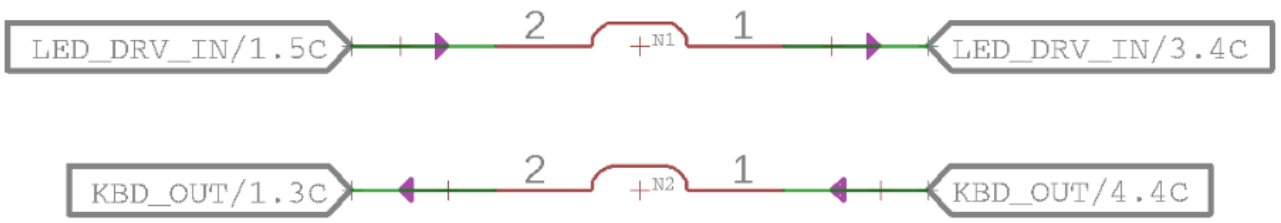
Demo Projekt

Schaltpläne vereinen und reduzieren 5.

vorher:

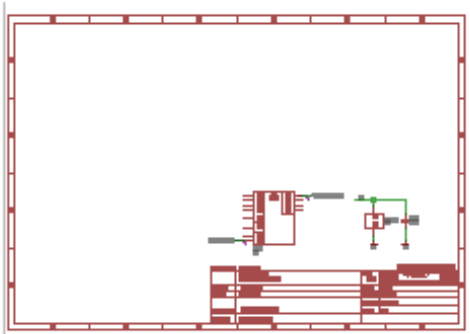


nachher:

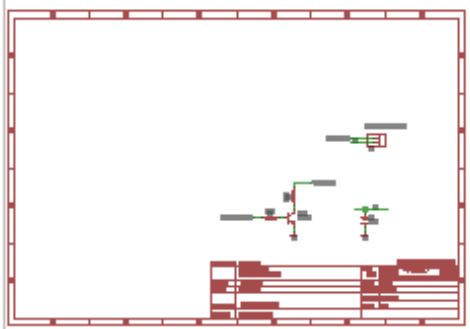


Demo Projekt

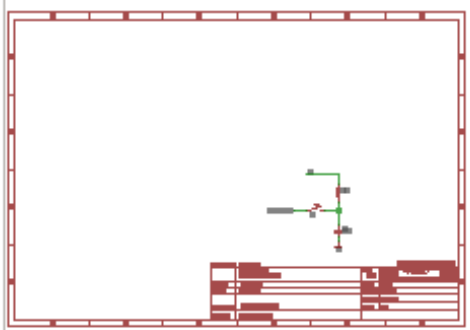
Schaltpläne vereinen und reduzieren 6.



1: MCU



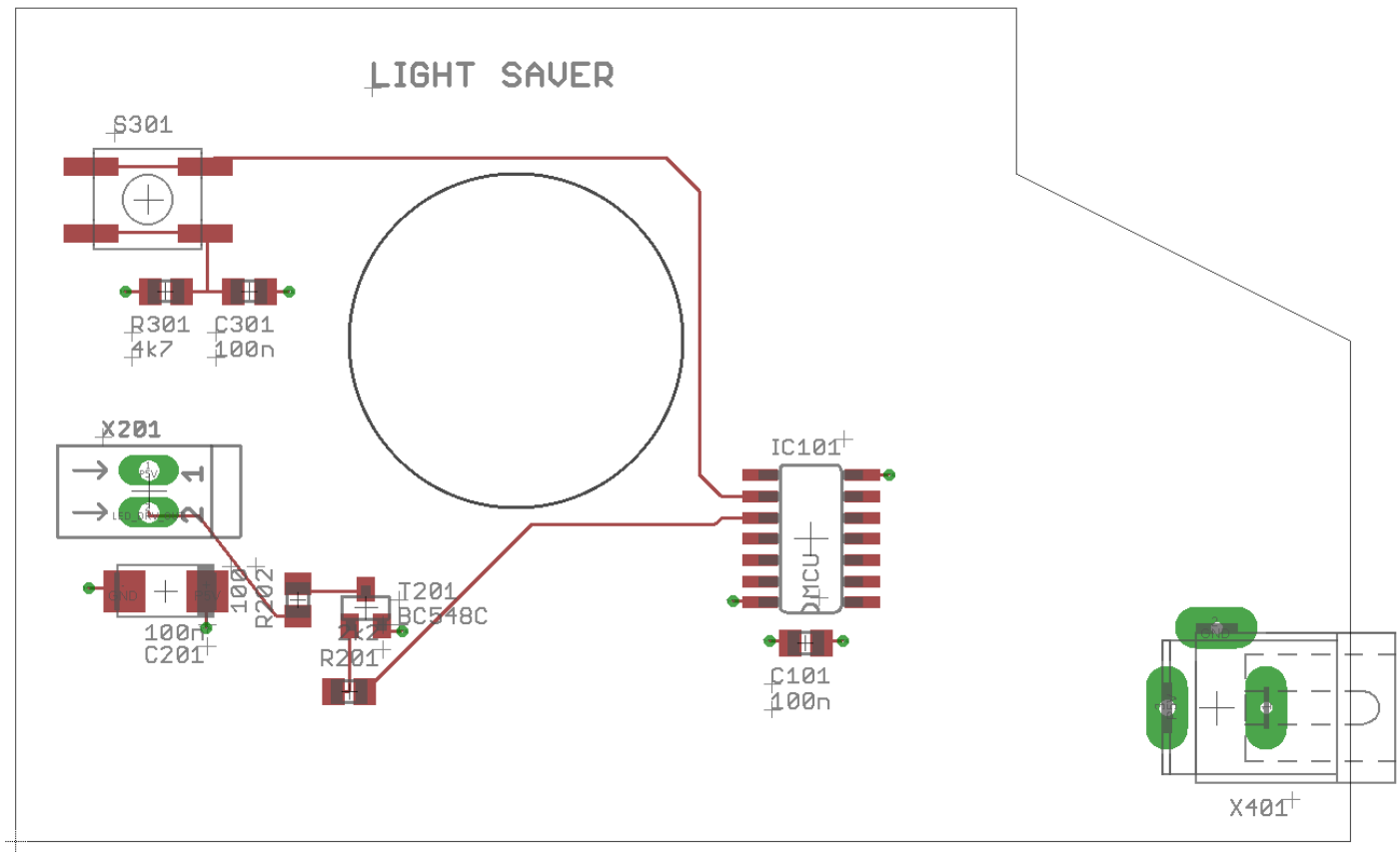
2: led driver



3: push button

Demo Projekt

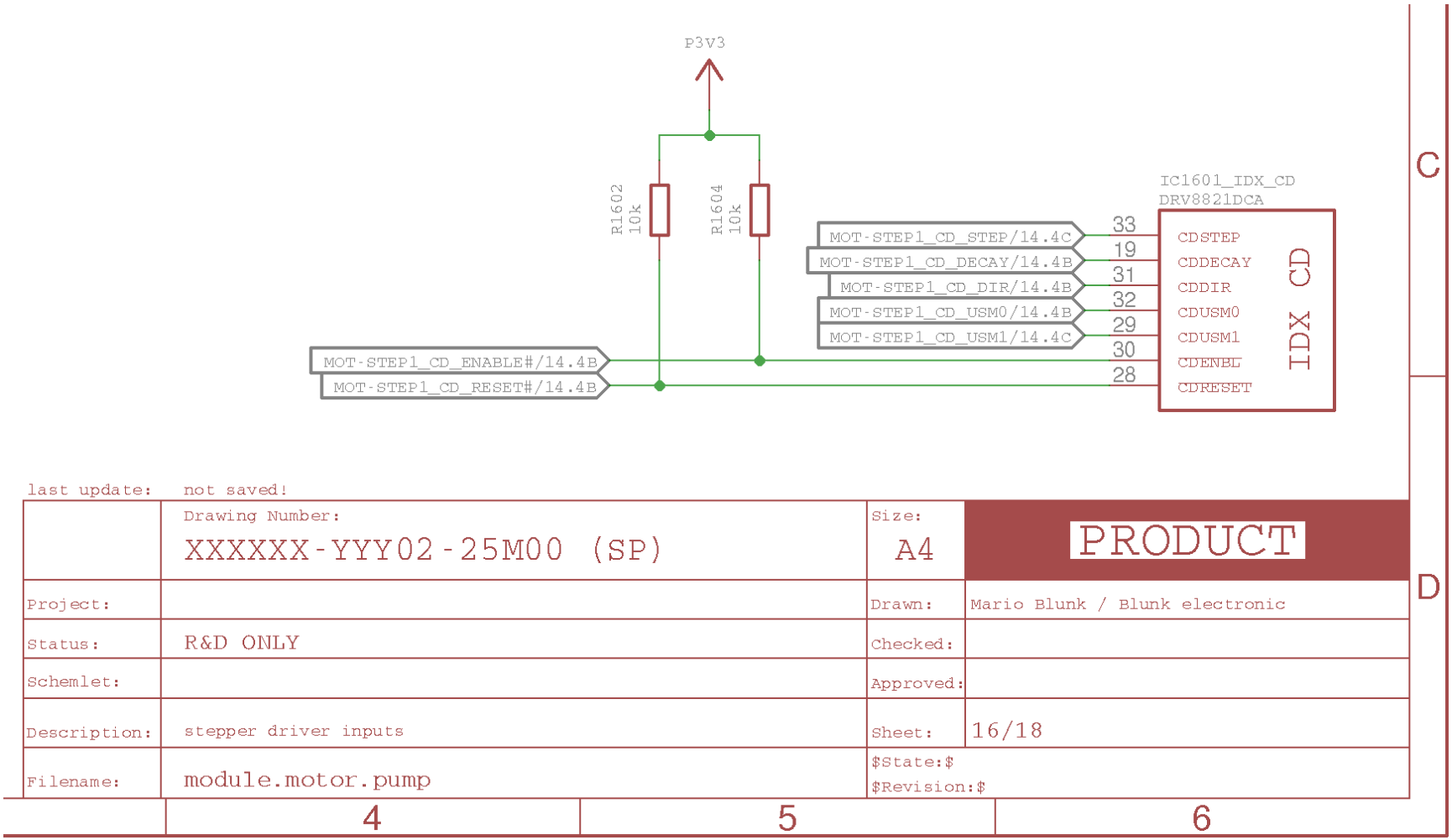
Finales PCB Layout des Produktes



1. Lesbarkeit
2. Übersicht
3. zeitsparende Bearbeitung
4. Wiederverwendung von Schaltungsteilen
5. Modularisierung

Konventionen und Spielregeln

Prefixe in Netznamen 1.



Konventionen und Spielregeln

Prefixe in Netznamen 2.

Template:

1. MODULENAME_FUNCTION_COUNT
2. MODULENAME_BLOCKNAME_FUNCTION_COUNT

Beispiele:

1. PWR_VREG_IN, PWR_VREG_OUT, PWR_VREG_ADJ
2. CPU_JTAG_TCK, CPU_JTAG_TMS
3. KBD_IN, KBD_OUT
4. LED_DRV_IN, LED_DRV_ON_OFF
5. CPU_GPIO_1, CPU_GPIO_2
6. KBD-1_SW_1, KBD-2_SW_1,
7. MOT-DRV-1_RL_1, MOT-DRV-2_RL_1,

In globalen, rig-weiten Netzen wie z.B. Betriebsspannungen und GND keine Prefixe verwenden.

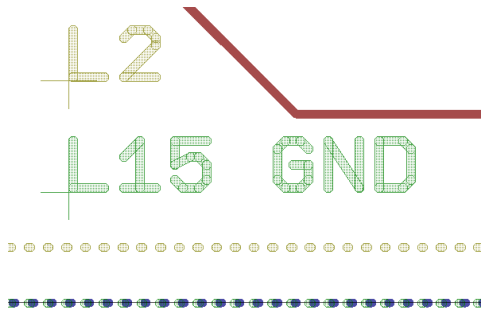
1. P3V3, P12V, N12V
2. GND
3. Konsens über Verwendung von Signallagen

Achtung ! Nicht nicht über Module hinweg verteilen:

1. Analog Masse (AGND)
2. sensible analoge Signale
3. SI-kritische Signale

Konventionen und Spielregeln

Nummer der Signallagen



1. Texte (auch in Zwischenlagen) helfen Entwicklung und Fertigung
2. Zählweise beachten (laut IPC von oben/TOP nach unten/BOTTOM) !

Konventionen und Spielregeln

Nutzen der korrekten Benennung von Bauteilen und Netzen

1. Vereinfachung im Layout
2. verbesserte Lesbarkeit
3. Vorbereitung zur Modularisierung
4. Anwendung von Skripten zum Check oder Modifizieren des Designs
5. Design-Checks mit externen Werkzeugen (Linting)

Achtung !

1. keine anonymen Netznamen wie N$1701 verwenden !
2. vermeide Sonderzeichen (μ , ö, ä, ...)

Konventionen und Spielregeln

Prefixe von Bauteilnamen

ANT	ANTENNA
B	BUZZER
BAT	BATTERY
C	CAPACITOR
CA	CAPACITOR_ADJUSTABLE
D	DIODE
DPH	DIODE_PHOTO
DI	DIAC
DIS	DISPLAY
F	FUSE
HS	HEATSINK
IC	INTEGRATED_CIRCUIT
J	JUMPER
JD	JUMPER (for development)
K	RELAY
KP	KEYPAD
L	INDUCTOR
LA	INDUCTOR_ADJUSTABLE
LS	LOUDSPEAKER
LED	LIGHT_EMMITTING_DIODE
LDA	LIGHT_EMMITTING_DIODE_ARRAY
M	MOTOR
MIC	MICROPHONE
N	NETCHANGER
OC	OPTOCOUPLER
Q	QUARTZ
R	RESISTOR
RA	RESISTOR_ADJUSTABLE
RN	RESISTOR_NETWORK
RP	POTENTIOMETER
RPH	RESISTOR_PHOTO
S	SWITCH
T	TRANSISTOR
TP	TRANSISTOR_PHOTO
TF	TRANSFORMER
TPT	TESTPOINT
TH	THYRISTOR
THP	THYRISTOR_PHOTO
TR	TRIAC
TUB	TUBE
X	CONNECTOR
XD	CONNECTOR

Beispiele IC1, T1, R1, TP1, RN1

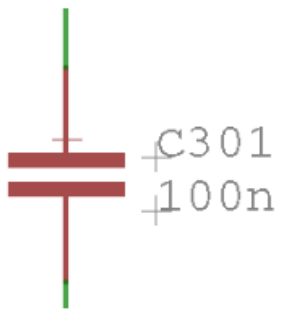
Konventionen und Spielregeln

Werte von Bauteilen

m	MILLIOHM
R	OHM
k	KILOOHM
M	MEGAOHM
G	GIGAOHM
p	PICOFARAD
n	NANOFARAD
u	MICROFARAD
m	MILLIFARAD
F	FARAD
n	NANOHENRY
u	MICROHENRY
m	MILLIHENRY
H	HENRY
V	VOLT
m	MILLIAMPERE
A	AMPERE
k	KILOHERTZ
M	MEGAHERTZ
G	GIGAHERTZ

Beispiele 4k7, 4n7, 2A5, 3V3

Die Bauteilart wie Widerstand, IC, LED ...
ergibt sich aus Schaltplansymbol und
Name:



Konventionen und Spielregeln

sonstige

1. Belegung Steckverbinder
2. LED Helligkeiten [3]
3. Kennzeichnung Schaltplanseiten (Produkt, Entwicklung, Routing)
4. ISO Datumsformat YYYY-MM-DD
5. Lagenverwendung (Layout)

Automatisierte Erstellung von CAM-Daten

1. es geht um: Plotdaten (Gerber), Bohrdaten
2. Materiallisten (BOM), Pick & Place
3. Prozess-Sicherheit
4. Verwendung quelloffener Dokumentation
5. am besten ASCII, *.csv $\rightarrow$ maschinell erstellbar und lesbar
6. *.xls, *.docx, u.s.w. nicht geeignet weil nicht quelloffen $\rightarrow$ maschinell NICHT erstellbar und lesbar
7. Zeitersparnis

Automatisierte Erstellung von CAM-Daten

Materiallisten (BOM) 1.

Gegenwärtiger Zustand	Sollzustand
Informationen zu Bestellnummern, Datenblättern, Preisen, Abnahmemengen, Produktions-Status, Lieferanten, Herstellern nicht oder chaotisch in EAGLE-Bibliotheken vorhanden	einheitliches Schema zum Zugreifen auf diese informationen
Wenn sich Bestellnummern, Datenblätter ... auf Seiten der Hersteller oder Lieferanten ändern, müssen EAGLE-Bibliotheken aktualisiert werden.	Nur EIN Primärschlüssel zu einer externen Material-Datenbank.
Material- und Bestelllisten müssen mühsam von Hand in Excel-Tabellen geschrieben und aktualisiert werden.	Material- und Bestelllisten werden maschinell innerhalb von Sekunden geschrieben und aktualisiert.
Leiterplatten-Bestücker kann Excel-Materiallisten nicht maschinell lesen – muß also die Tabellen von Hand in Fertigungssystem übertragen.	Bestücker kann Materiallisten maschinell in sein Fertiungssystem einlesen.
Warenwirtschaftsystem kostet Lizenz und bindet an Hersteller	KEINE Lizenzen oder Gebühren, KEINE Bindung weil Quellcode offen

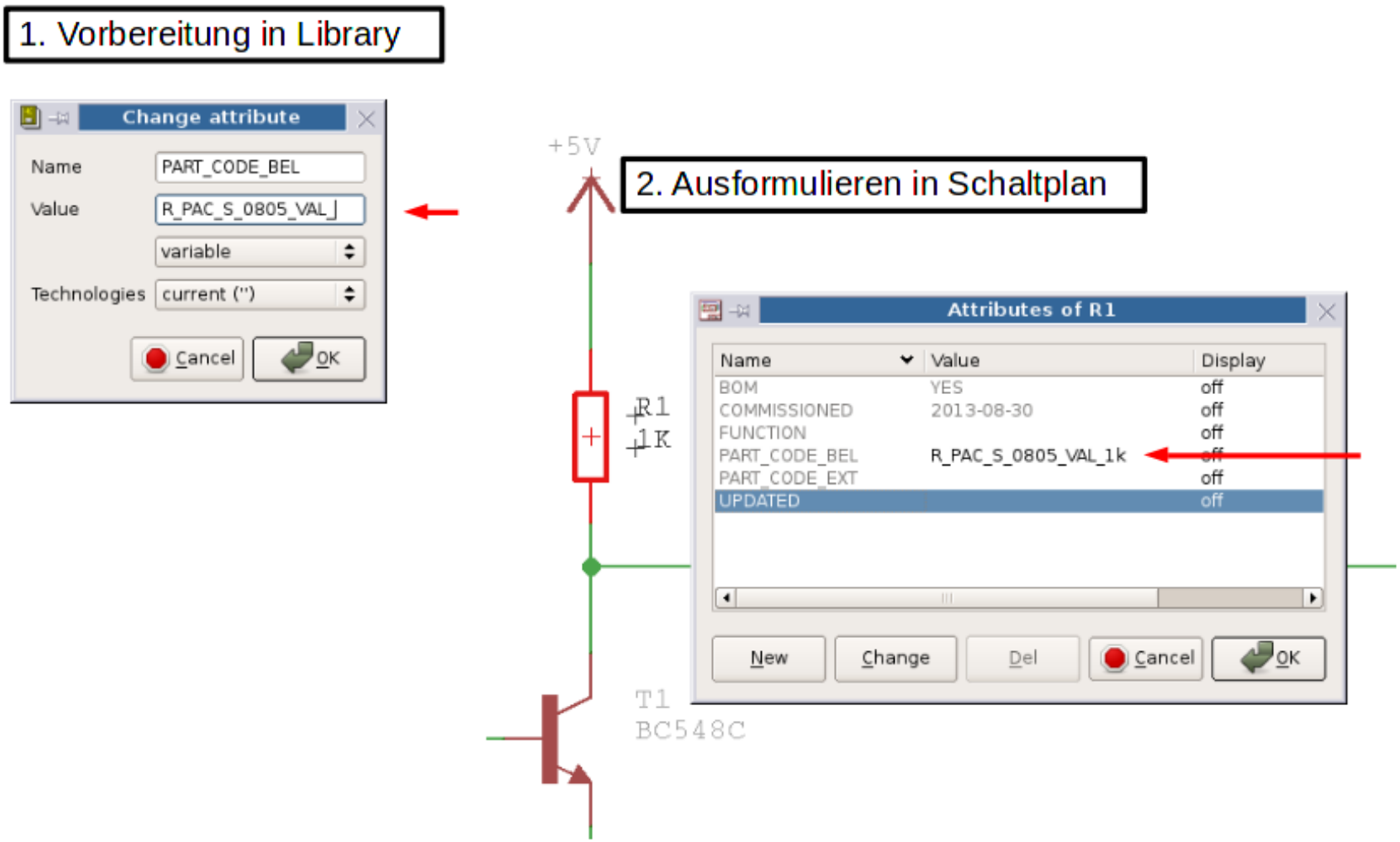
Automatisierte Erstellung von CAM-Daten

Materiallisten (BOM) 2.

1. Ein CAE-System ist kein ERP-System !
2. Nur EIN Materialschlüssel (MS) ist die Schnittstelle zum ERP-System.
3. MS in Bibliothek des CAE-Systems vorbereiten
4. MS im Schaltplan ausformulieren
5. Teileliste (TS) exportieren als *.csv
6. TS mit ERP-System prozessieren
7. ERP-System liefert Bestellummern, Datenblätter, Preise, u.s.w.

Automatisierte Erstellung von CAM-Daten

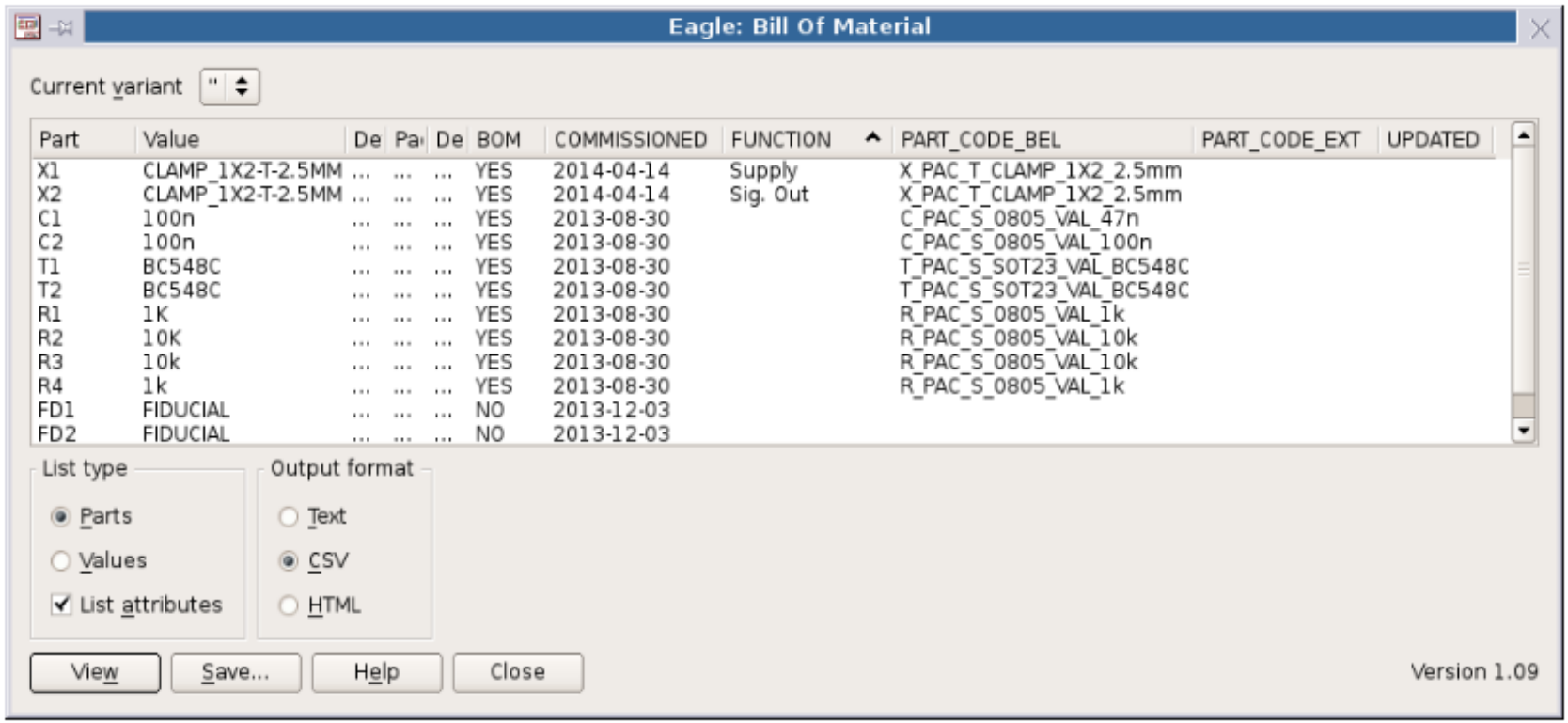
Materiallisten (BOM) 3.



Automatisierte Erstellung von CAM-Daten

Materiallisten (BOM) 4.

Exportieren der Teileliste (”vorab” BOM)



Automatisierte Erstellung von CAM-Daten

Materiallisten (BOM) 5.

Prozessierung mit ERP-System[1]

transceiver_m1.csv - LibreOffice Calc

File Edit View Insert Format Tools Data Window Help

Liberation Sans 10

A3 board:

	A	B	C	D	E	F	G
1	BILL OF MATERIAL						
2	----------						
3	board:	transceiver_m1.csv					
4							
5	created by STOCK MANAGER	V007					
6	contact:	www.blunk-electronic.de					
7	date:	2015-06-04 10:28:27	(YYYY-MM-DD HH:MM:SS)				
8							
9	part count SMD:	36					
10	part count THT:	8					
11	part count TOTAL:	44					
12							
13	POS.	QTY	PART_NAME	PART_CODE_BEL	PART_ID	PRICE_NET_MIN	PRICE_NET_MAX
14							
15	1	1	C501	C_PAC_S_0805_VAL_100n	13	0.01	0.04
16	2	1	IC501	IC_PAC_S_SOIC8_VAL_24LC256-I/SN	56	0.63	0.87
17	3	2	LED601, LED1302	LED_PAC_S_0805_VAL_red	99	0.46	0.46
18	4	1	LED602	LED_PAC_S_0805_VAL_green	98	0.23	0.23
19	5	2	LED603, LED1301	LED_PAC_S_0805_VAL_blue	97	0.46	0.46
20	6	6	R501, R502, R503, R504, R505, R506	R_PAC_S_0805_VAL_4k7	133	0.24	0.24
21	7	5	R601, R603, R605, R1301, R1303	R_PAC_S_0805_VAL_1M	147	0.05	0.05
22	8	2	R602, R1304	R_PAC_S_0805_VAL_1k5	150	0.04	0.04
23	9	1	R604	R_PAC_S_0805_VAL_330	158	0.03	0.03
24	10	2	R606, R1302	R_PAC_S_0805_VAL_820	316	0.06	0.06
25	11	1	R1101	R_PAC_S_0805_VAL_0	138	0.01	0.01
26	12	5	T601, T602, T603, T1301, T1302	T_PAC_S_SOT23_VAL_DMG3420U	202	0.60	1.25
27	13	7	XD601, XD803, XD1201, XD1301, XS601, XS1101, XS1301	X_PAC_T_MALE_1X6_VERTICAL_GRID_2.54mm	267	3.15	3.15
28	14	3	XD801, XD802, XD1001	X_PAC_S_MALE_2X10_VERTICAL_GRID_2mm	229	4.23	4.23
29	15	1	XD1302	X_PAC_T_MALE_3X1_VERTICAL	264	0.08	0.08
30	16	3	XS701, XS702, XS901	X_PAC_S_FEMALE_1x20_VERTICAL_GRID_1mm	219	3.54	3.54
31	17	1	XS703	X_PAC_S_FEMALE_1x6_VERTICAL_GRID_1mm	222	0.50	0.50
32	----------	----------	----------	----------	----------	----------	----------
33						14.32	15.24
34	END OF BOM						

Sheet1

Sheet 1 / 1 Default Sum=0 100%

Achtung !

Automatisierte Erstellung von CAM-Daten

Aufbereitung für Fertigung 1.

Gegenwärtiger Zustand	Sollzustand
Daten für Plot/Gerber, Bohrungen, BOM, … werden mühsam in ein CAM-Archiv zusammengetragen	nur EIN Vorgang nötig
Design-Daten (sch/brd/lbr) weichen vom Stand im CAM-Archiv ab.	Design-Daten werden synchron mit Erstellung des CAM-Archives archiviert.
PCB-Hersteller und Bestücker muß CAM-Daten deuten weil kein einheitliches Übergabe-Format besteht	CAM-Daten werden immer auf gleiche Weise archiviert. Zulieferer müssen nur einmal „angelernt“ werden.

Automatisierte Erstellung von CAM-Daten

Aufbereitung für Fertigung 2.

1. Bediener startet Shell-Skript (Diese sendet Anweisungen an EAGLE.)
   (Flexibilität, Automatisierung, Reproduzierbarkeit ...)
2. EAGLE exportiert Daten.
3. Shell-Skript führt weitere Aktionen aus (Zip, Datumstempel, PDF erzeugen ...)

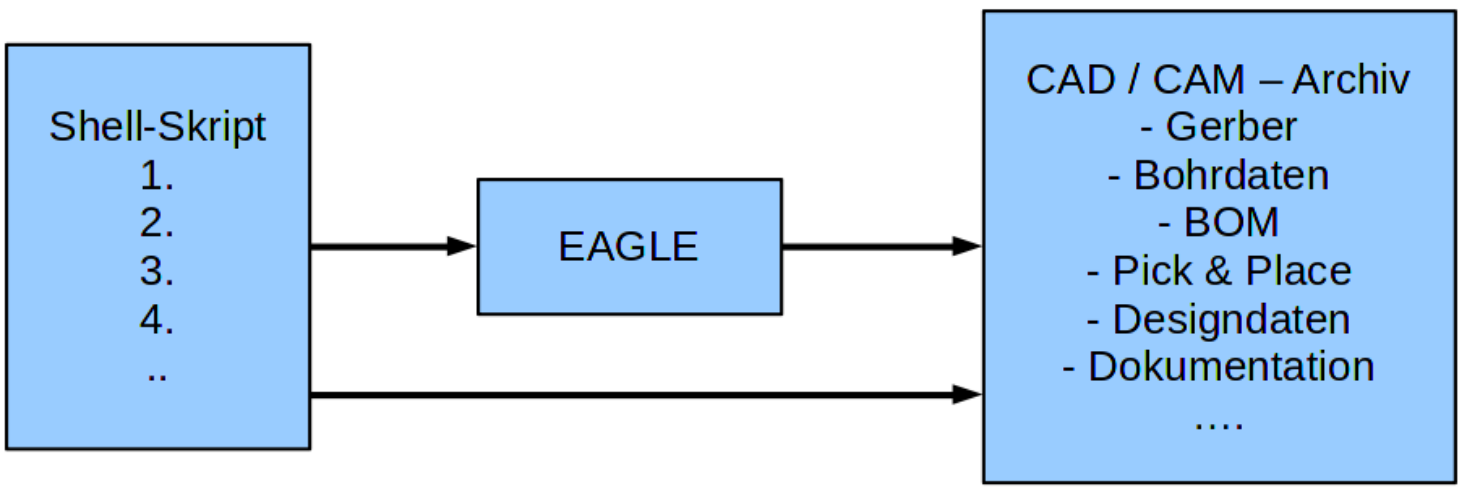


Figure 2: erweiterter CAM-Prozessor[2]

Automatisierte Erstellung von CAM-Daten

Aufbereitung für Fertigung 3.

```
$ cad/projects/eagle/BE/training/sqw> mkcam sqw.brd 2
CAM file generator version 005

board file given : sqw.brd
layer count      : 2

generating CAM files in directory 'cam/sqw_CAM_2014-08-11_1144' ...
- layer 1 (top) ...
- layer 16 (bottom) ...
- silk screen ...
- stop mask ...
- cream mask ...
- drills ...
- Drills Documentation ...
- plated millings ...
- outline ...
- documentation
dimensions, measures, document ...
dimension, tplace, tvalues, tdoc, document
dimension, bplace, bvalues, bdoc, document
dimension, bplace, borigin, bnames, bdoc, document …
netlist ..
partlist ..
pick & place …
converting gerber files to pdf …
...
```

Das Ergebnis: CAD / CAM Archiv !

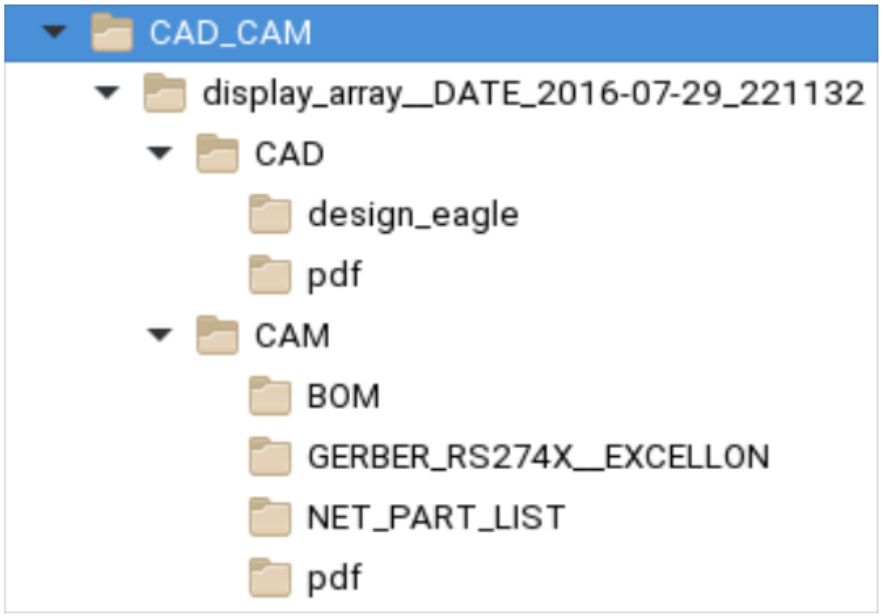


Figure 3: erweiterter CAM-Prozessor[2]

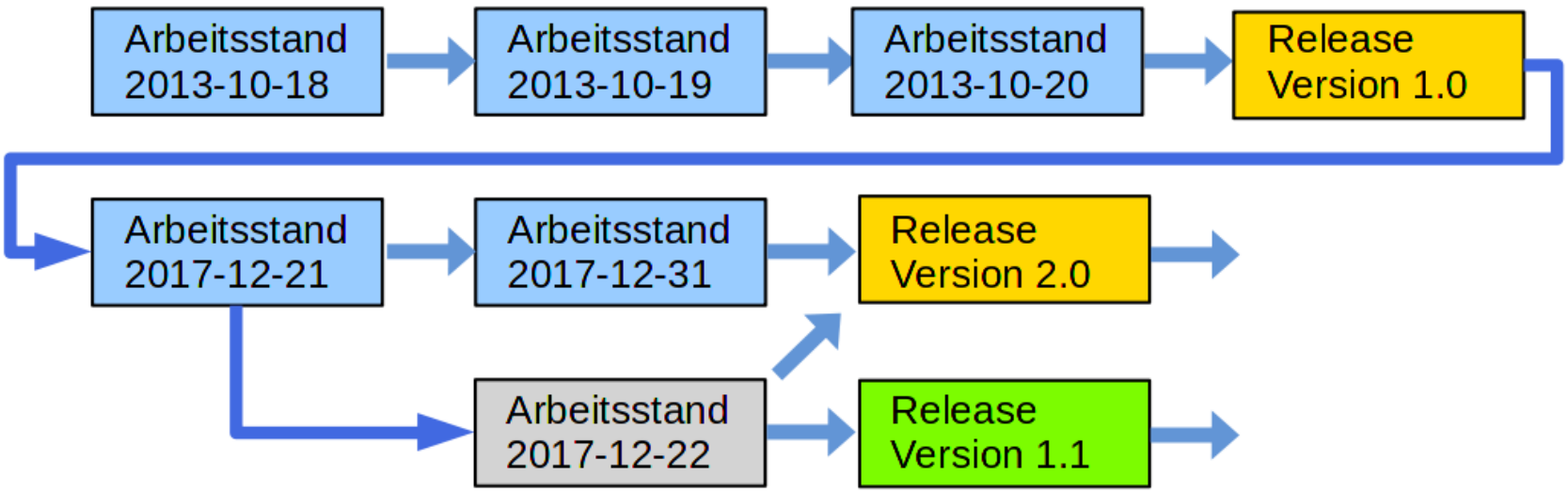
Versionskontrolle 1.

Warum ?

1. erleichterte Fehlersuche/Debugging
2. Rückverfolgung von Änderungen
3. Archivierung von Konstruktionsdaten (CAD), Quellcode, Dokumentation, Fertigungsdaten (CAM) ...
4. Release, Versionierung (V1.0, V2.2, u.s.w.)
5. für Zertifizierungen (ISO 9001, IEC 61508 / 61511 / 62061, DO-178B, MIL-STD-882-E, ...)

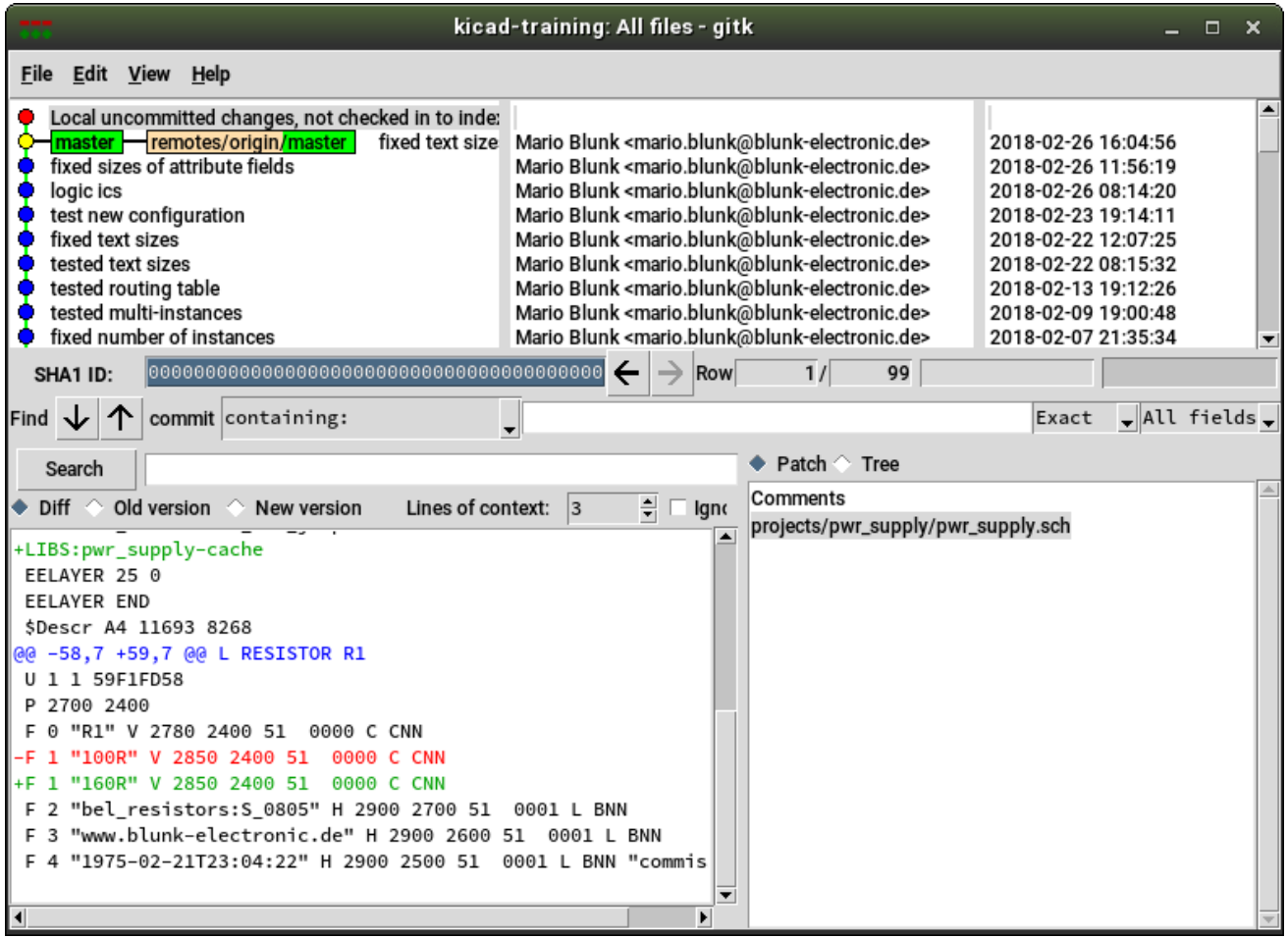
http://www.blunk-electronic.de/pdf/git_einfuehrung.pdf
http://www.blunk-electronic.de/pdf/git_training_teil_1.pdf

Versionskontrolle 2.



http://www.blunk-electronic.de/pdf/git_einfuehrung.pdf
http://www.blunk-electronic.de/pdf/git_training_teil_1.pdf

Versionskontrolle 3.



http://www.blunk-electronic.de/pdf/git_einfuehrung.pdf
http://www.blunk-electronic.de/pdf/git_training_teil_1.pdf

Literaturquellen I

[1] Mario Blunk / Blunk electronic *Ein schlankes ERP-System - OpenSource*.
http://www.blunk-electronic.de/products/sw/stock_manager/doc/Stock_Manager_de.pdf

[2] Mario Blunk / Blunk electronic *Der erweiterte EAGLE CAM-Prozessor*.
https://github.com/Blunk-electronic/eagle_CAM_processor.git

[3] Mario Blunk / Blunk electronic *Angleichung der Helligkeit von Status LEDs*.
http://www.blunk-electronic.de/pdf/LED_brightness_adjustment.pdf

[4] Mario Blunk / Blunk electronic *Zuverlässigkeit in der Elektronik*.
http://www.blunk-electronic.de/pdf/zuverlaessigkeit.pdf

[5] Mario Blunk / Blunk electronic *Testverfahren der Elektronik*.
http://www.blunk-electronic.de/pdf/testverfahren_der_elektronik.pdf

[6] Mario Blunk / Blunk electronic *Design Checklist*.
http://www.blunk-electronic.de/pdf/Design_Checklist_en.pdf

[7] Mario Blunk / Blunk electronic *Boundary Scan Training Teil 1*.
http://www.blunk-electronic.de/pdf/bst_teil_1.pdf

[8] Mario Blunk / Blunk electronic *Boundary Scan Training Teil 2*.
http://www.blunk-electronic.de/pdf/bst_teil_2.pdf

[9] Walt Kester, James Bryant, Mike Byrne / Analog Devices *Grounding Data Converters and Solving the Mystery of "AGND" and "DGND"*.

[10] Joachim Franz *EMV Störungssicherer Aufbau elektronischer Schaltungen*. Vieweg+Teubner 2011

[11] Alan Rich *Shielding and Guarding*. Analog Dialogue 17-1 1983

[12] Mark I. Montrose *Printed Circuit Board Design Techniques for EMC Compliance*. IEEE 2000

Literaturquellen II

[13] Howard Johnson / Marin Graham *High-Speed Signal Propagation, Advanced Black Magic*. Prentice Hall PTR 2012