ENTWICKLEREFFIZIENZ

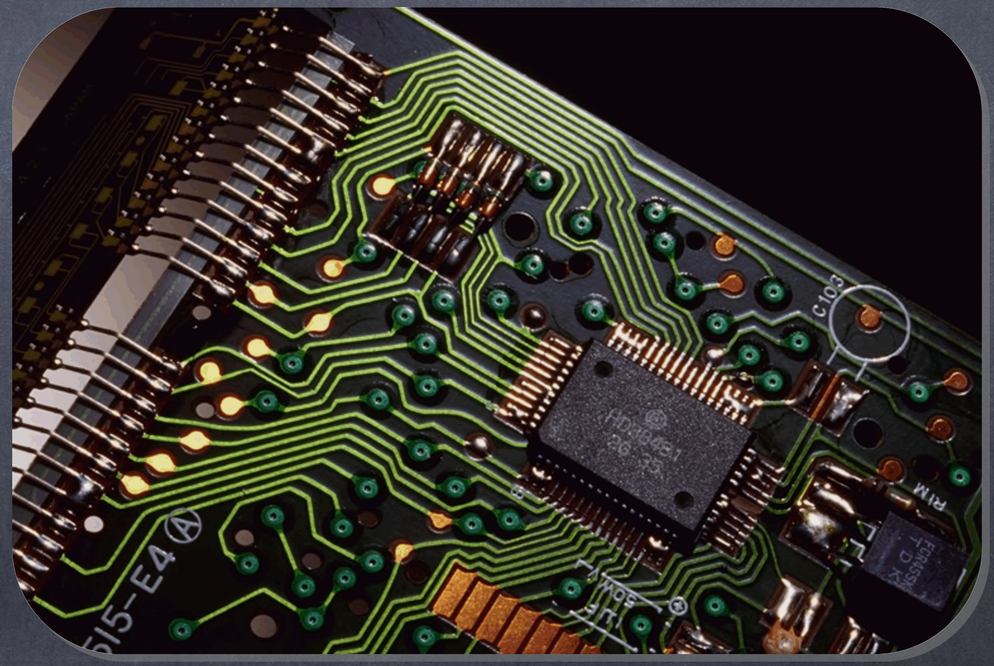
SOFTWARE
ARCHITEKTUR
FÜR
EINGEBETTETE SYSTEME

Elektronik & Informatik OWL e.G.

EIOWL.DE

Eingebettete Systeme

EINGEBETTETE SYSTEME
SIND
TECHNISCHE GERÄTE
MIT
MICROCONTROLLERN.



Technische Geräte

Internet of things

Eingebettete Systeme

BESTANDTEILE:

- GEHÄUSE
- ELEKTRONIK (HARDWARE)
- COMPUTERPROGRAMM (SOFTWARE)

Software

IN DIESER PRÄSENTATION
GEHT ES UM DIE

SOFTWARE ARCHITEKTUR

Definition

- Die Gesamtsoftware besteht aus Komponenten
- Die Architektur beschreibt

  die Aufgaben der Komponenten

  und die Beziehungen der Komponenten zueinander
- Bauplan der Gesamtsoftware

Erwartungen

SOFTWARE MUSS

- TESTBAR
- WARTBAR
- WIEDERVERWENDBAR

SEIN.

Testbarkeit

- Das Produkt entspricht nachweisbar den Anforderungen.
- Eine ausreichende Testtiefe ist erreicht worden.
- Es gibt eine überschaubare Anzahl von Testfällen.

Wartbarkeit

- Änderungen können in kurzer Zeit durchgeführt werden.
- Änderungen führen nicht zu Problemen in anderen Programmteilen.

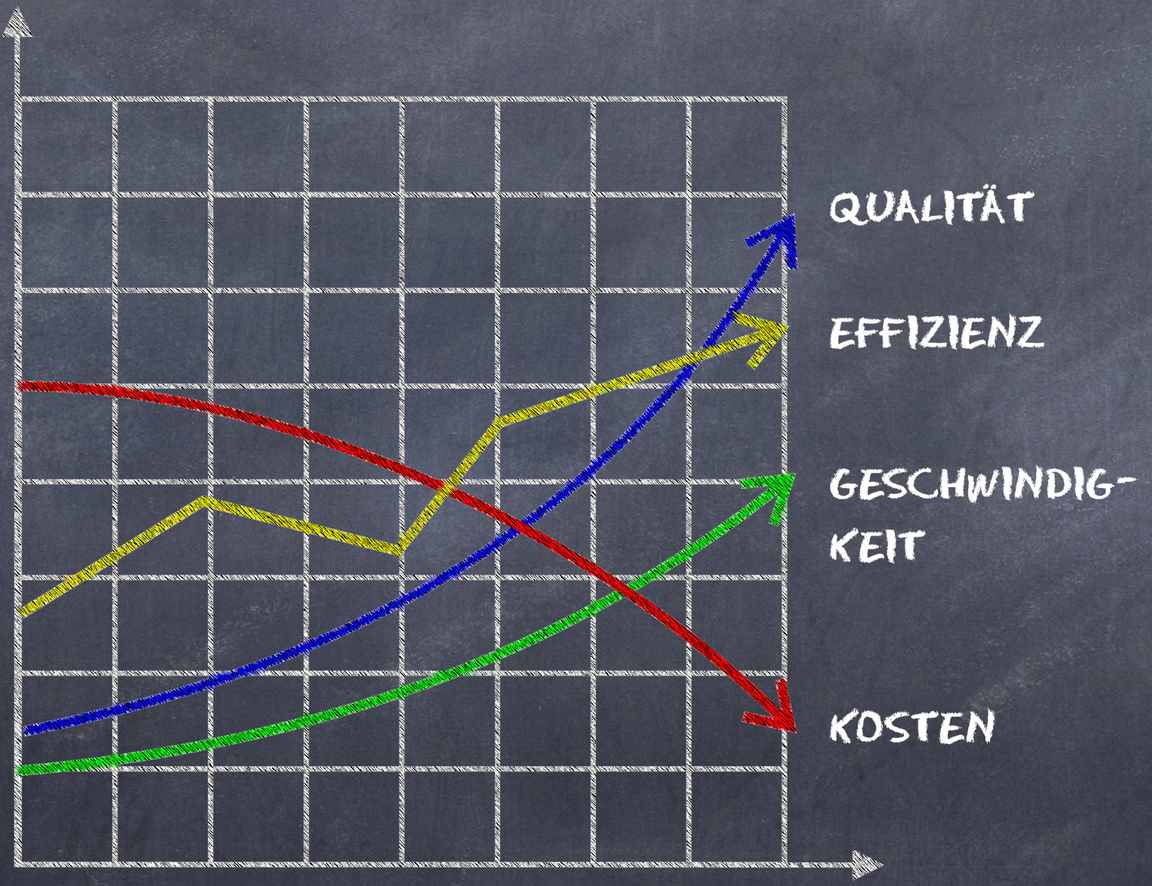
Wiederverwendbarkeit

Teile der Software können in

- Nachfolgeprodukten
- anderen Produkten

verwendet werden.

Primärziel Effizienz



Latein agilis = beweglich

WIESO
BEWEGLICH ?

Anforderungen

WAHRSCHEINLICH SIND NICHT ALLE ANFORDERUNGEN ZU PROJEKTBEGINN BEKANNT

VIELE AUSNAHMEN ODER SONDERFÄLLE WERDEN ERST WÄHREND DER ENTWICKLUNG ENTDECKT.

Lernen

ERKENNTNISSE AUS DEN PROTOTYPENPHASEN

TRAGEN ZUR VERBESSERUNG

DES PRODUKTES

UND ZUR WETTBEWERBSFÄHIGKEIT BEI.

Ideen

Auch für den Kunden dreht sich die Welt weiter.

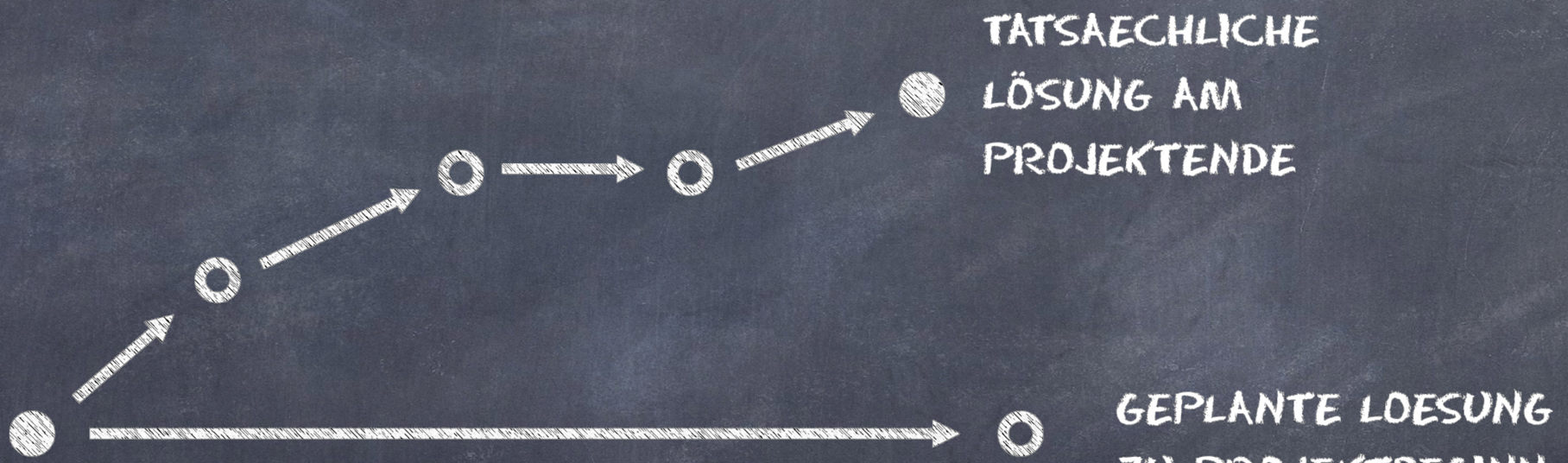
Auch er kommt erwartungsgemäß mit neuen Ideen.

Updates

NEUE UND BESSERE PRODUKTE FOLGEN.

?

Projektverlauf



PRODUKTENTWICKLUNG IST EHER EINE EXPEDITION!

Änderungssong

WIR ÄNDERN MORGEN, WIR ÄNDERN HEUT'
WIR ÄNDERN WÜTEND UND ERFREUT,
WIR ÄNDERN OHNE ZU VERZAGEN,
AN ALLEN SIEBEN WOCHENTAGEN.

WIR ÄNDERN TEILS AUS PURER LUST
MIT VORSATZ TEILS, TEILS UNBEWUSST.
WIR ÄNDERN GUT UND AUCH BEDINGT,
WEIL ÄNDERN IMMER ARBEIT BRINGT.

WIR ÄNDERN RESIGNIERT UND STILL
WIE HINZ UND KUNZ ES HABEN WILL;
DIE ALTEN ÄNDERN UND DIE JUNGEN,
WIR ÄNDERN SELBST DIE ÄNDERUNGEN.

WIR ÄNDERN, WAS MAN ÄNDERN KANN
UND STEHEN DABEI UNSERN MANN.
IST EIN PLAN AUCH GUT GELUNGEN,
BESTIMMT VERTRÄGT ER ÄNDERUNGEN.

WIR ÄNDERN DESHALB FRÜH UND SPÄT
ALLES, WAS ZU ÄNDERN GEHT.
WIR ÄNDERN HEUT UND JEDER ZEIT,
ZUM DENKEN BLEIBT UNS WENIG ZEIT.

UND WENN WIR DANN GENUG GEÄNDERT,
DANN HABEN WIR UNS AUCH VERÄNDERT,
DENN DURCH DIE EWIGE ÄNDEREI
GEHT UNSER LEBEN SCHNELL VORBEI.

UND STEHN WIR DANN AM HIMMELSTOR,
DER ALTE PETRUS STEHT DAVOR;
DANN IST'S SOWEIT, JETZT BLEIBT'S DABEI
VORBEI IST'S MIT DER ÄNDEREI.

GEZ. ÄNDERMANN
ÄNDERUNGEN VORBEHALTEN!

QUELLE: INTERNET, VERFASSER UNBEKANNT

Schichtenmodell

Driver

REGISTERZUGRIFFE AUF DIE SCHNITTSTELLEN

- PORTS
- ALTERNATIVE PORTFUNKTIONEN
  AD, PWM, INPUT CAPTURE, UART, SPI, CAN...
- INTERNE SCHNITTSTELLEN
  FLASH, GPU, WATCHDOG...

Applikation

FUNKTIONALITÄT

ALGORITHMEN

IST SELBST AUCH GESCHICHTET

HAL

BINDEGLIED ZWISCHEN HARDWARE UND APPLIKATION

DIE APPLIKATION WEISS NICHT WO DIE INFORMATIONEN HERKOMMEN ODER HINGEHEN.

BEISPIEL: ANALOG-DIGITAL-EINGANG ODER CAN-BUS

Betriebssystem

INITIALISIERUNGEN

AUFRUFE

Hilfsfunktionen

KLASSISCHE FUNKTION

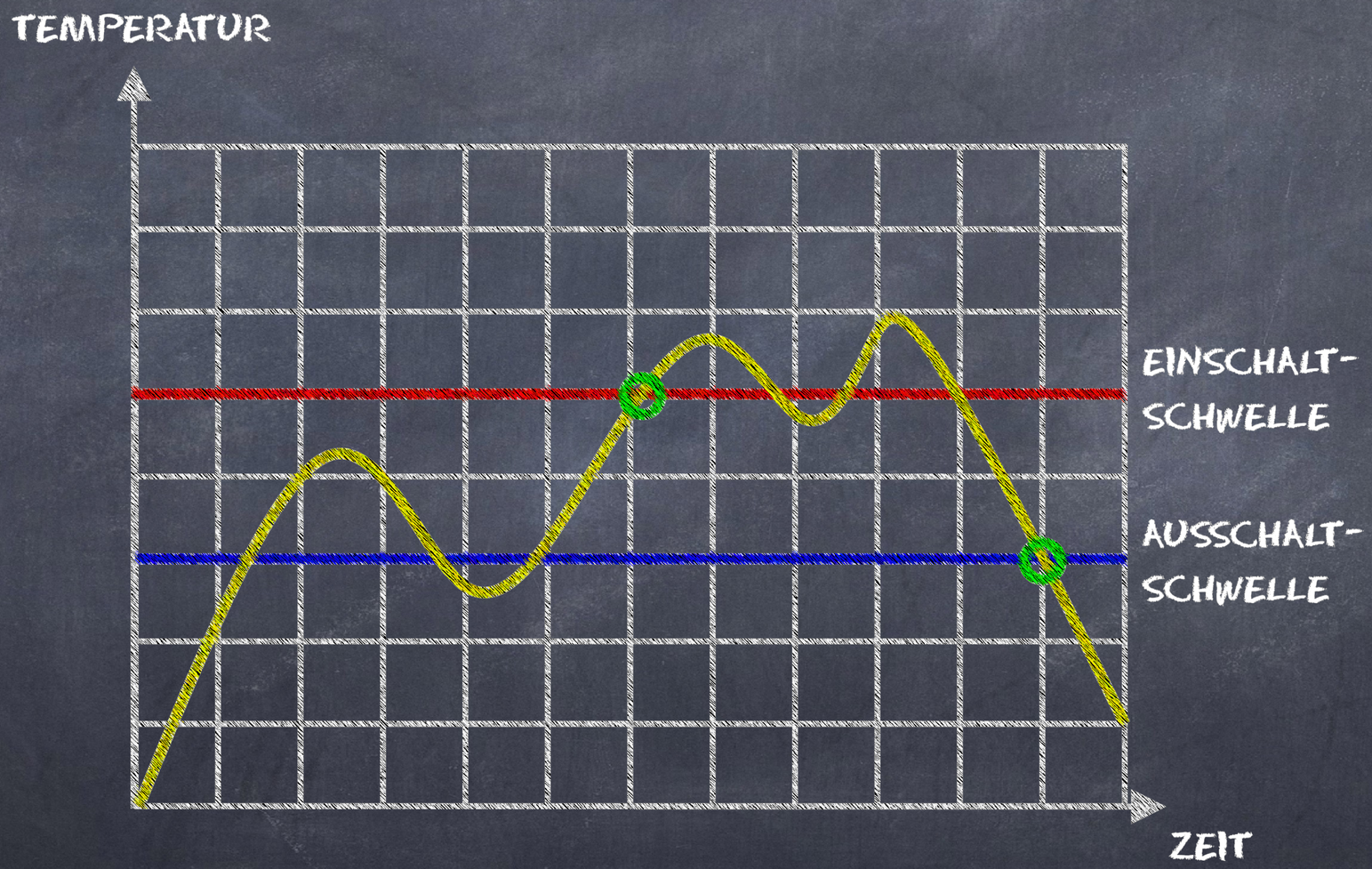
Z.B.: MATHE-FUNKTIONEN

EINHEITEN UMRECHNUNGEN

Regeln

- Die höheren Schichten beobachten die darunterliegenden
- Set Funktionen vermeiden
- Layer Bridging ist erlaubt

Beispiel Lüfter



Beispiel Lüfter

DigitalOutMain();

DRV OUT

HAL OUT

GetPortState();

FanMain();

APPLICATION

GetFanState();

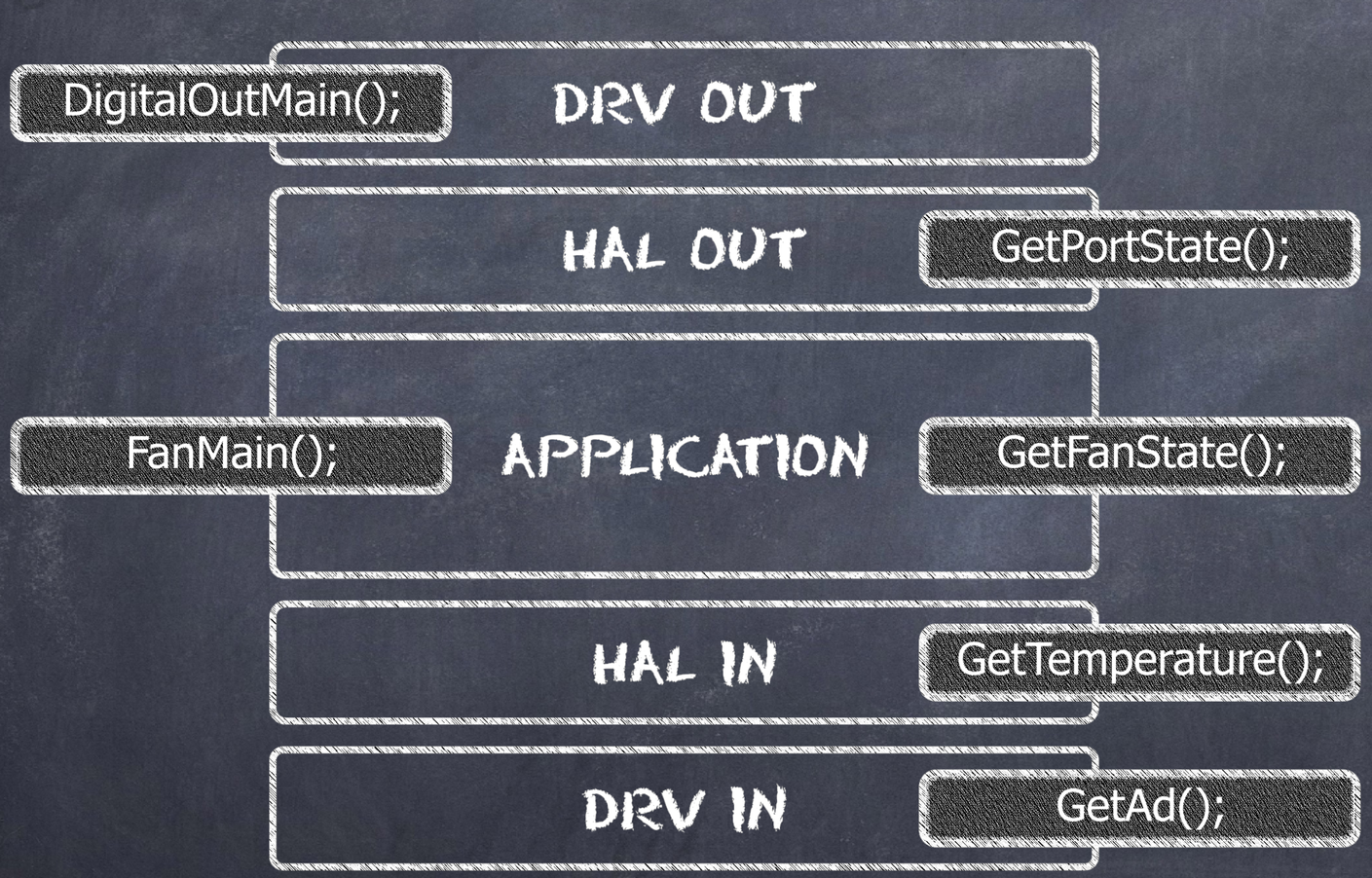
HAL IN

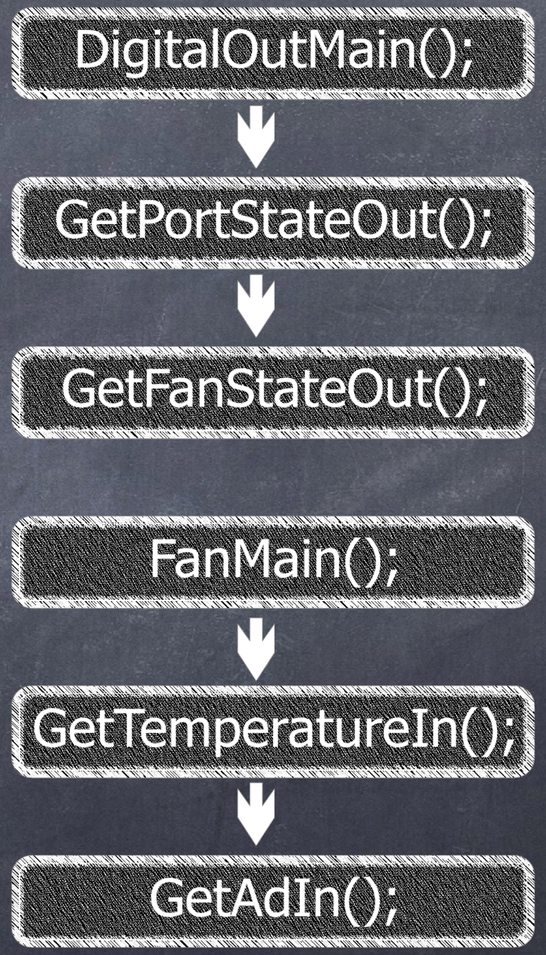
GetTemperature();

DRV IN

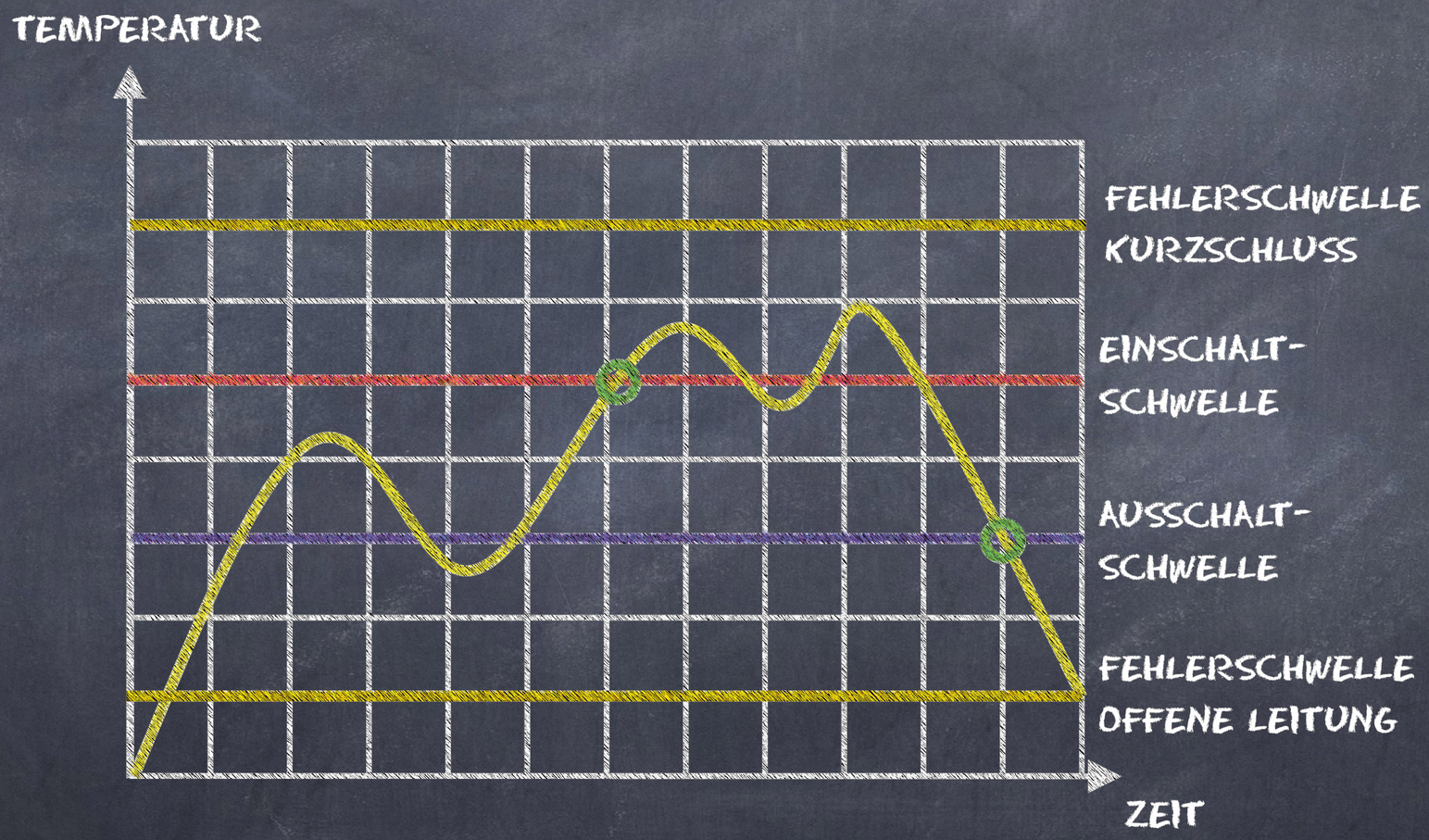
GetAd();

Beispiel Lüfter



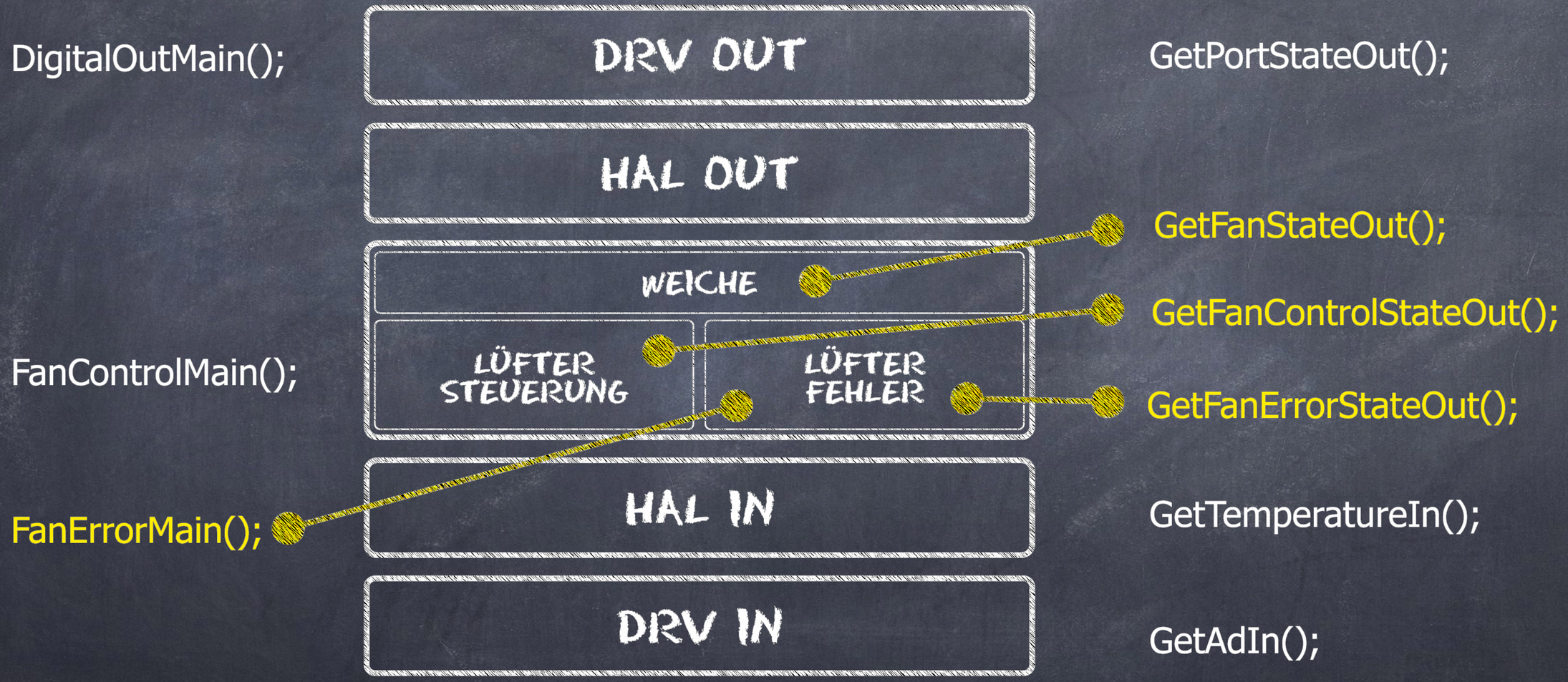


Nichtfunktion Fehler



Nichtfunktion

Nichtfunktion



Nichtfunktion

DigitalOutMain();

GetPortStateOut();

GetFanStateOut();

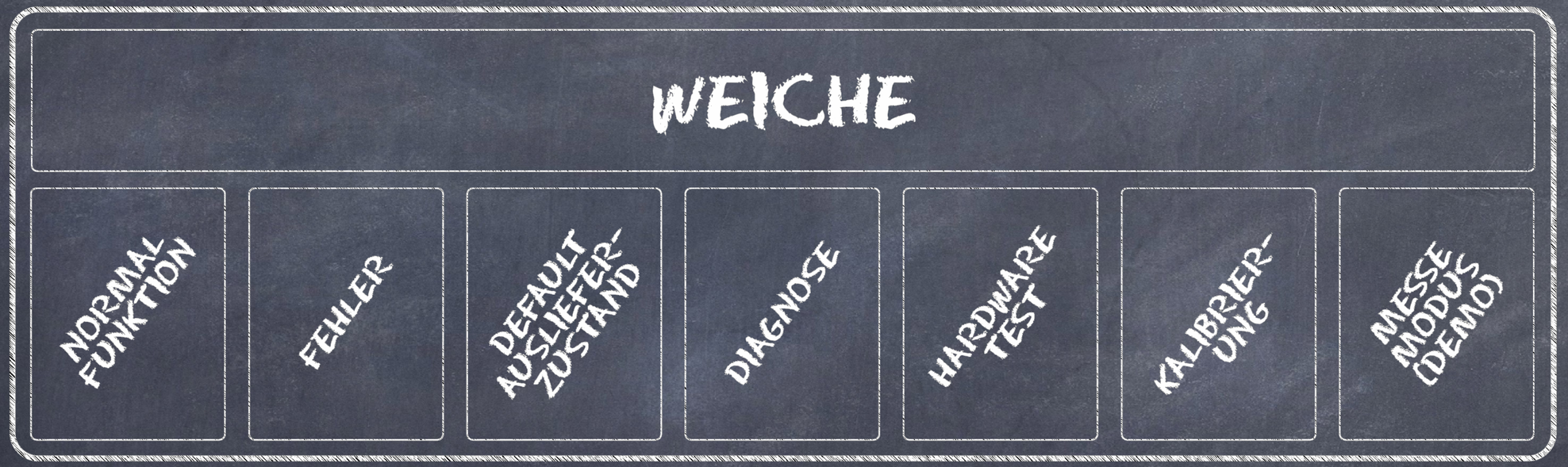
GetFanControlStateOut(); GetFanErrorStateOut();

FanControlMain(); FanErrorMain();

GetTemperatureIn();

GetAdIn();

Betriebsarten

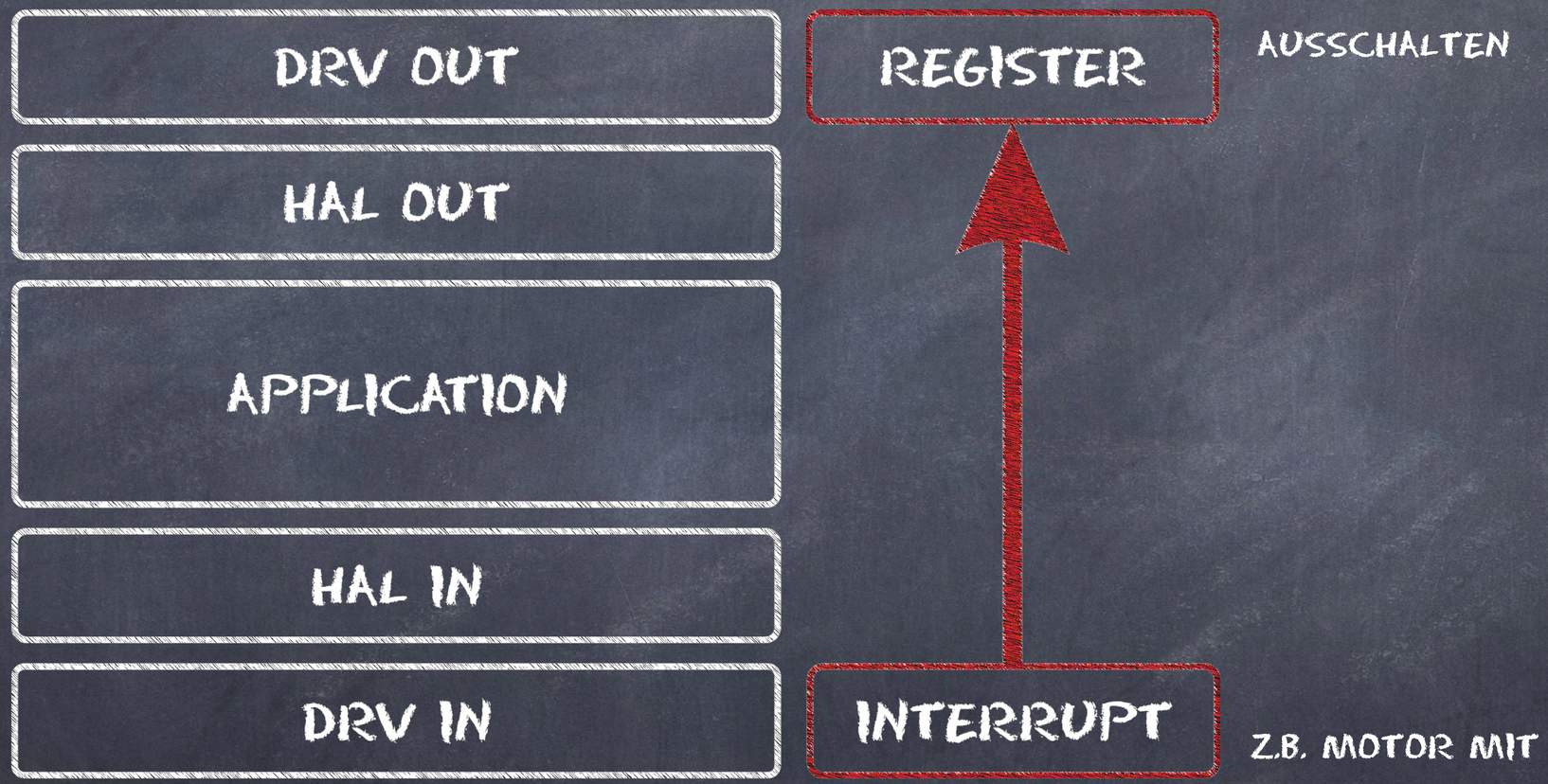


JE NACH BETRIEBSART ENTSCHEIDET DIE WEICHE WELCHER SOLLWERT GÜLTIG IST.

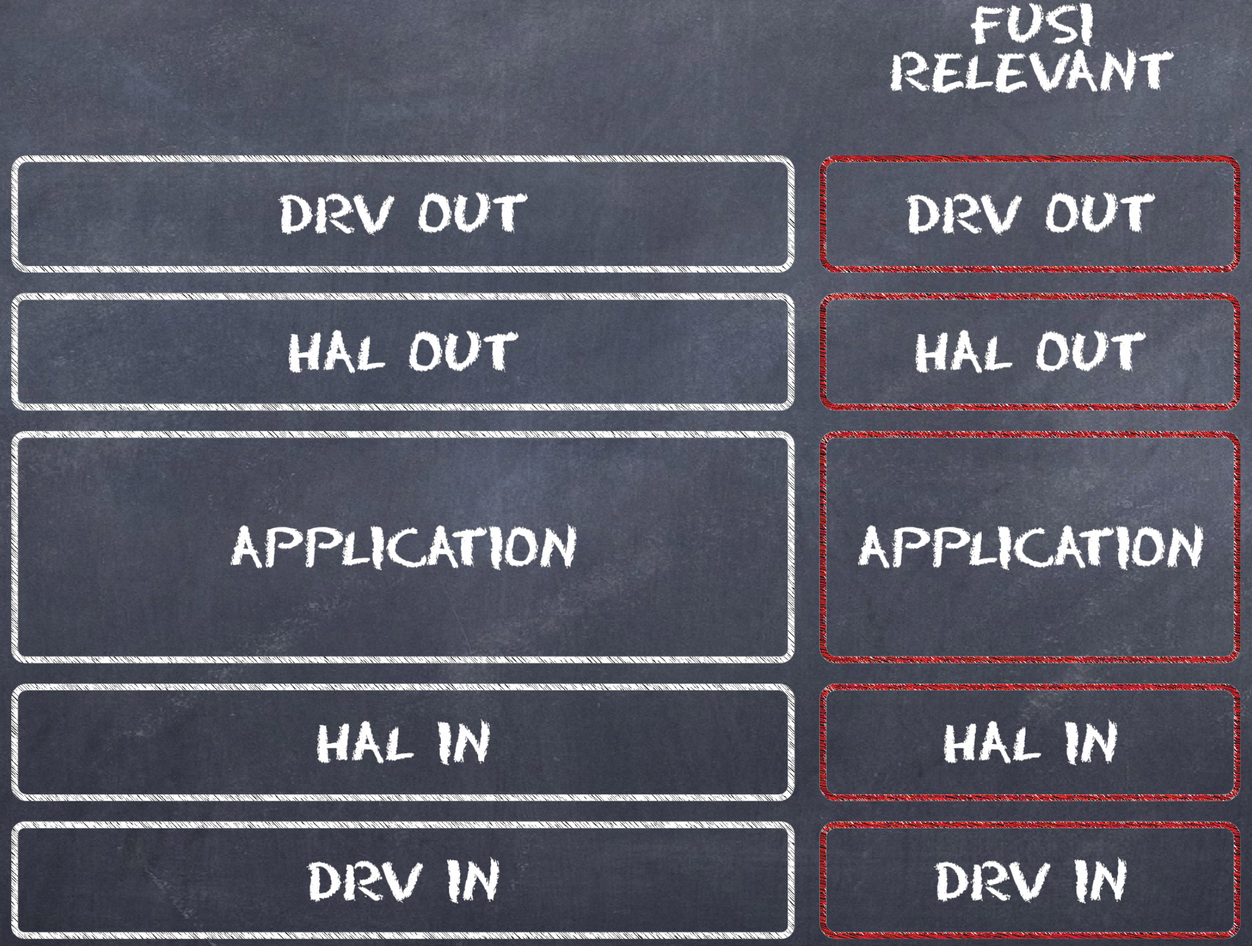
Varianten


WEICHE
NORMAL FUNKTION VARIANTE 1
NORMAL FUNKTION VARIANTE 2
NORMAL FUNKTION VARIANTE 3
FEHLER VARIANTE 1
FEHLER VARIANTE 2
DEFAULT AUSLIEFER-ZUSTAND
DIAGNOSE
HARDWARE TEST
...
MESSE DEMO MOSUS

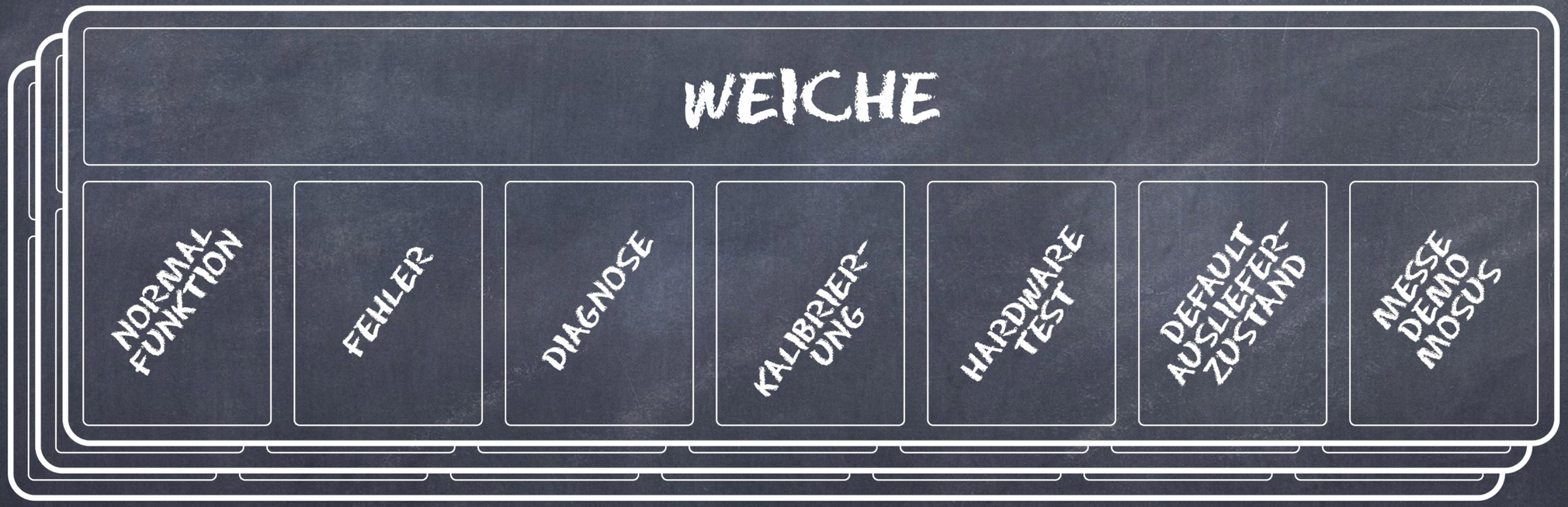
Bypass


DRV OUT
HAL OUT
APPLICATION
HAL IN
DRV IN
REGISTER
INTERRUPT
SOFORT
AUSSCHALTEN
Z.B. MOTOR MIT
KURZSCHLUSS

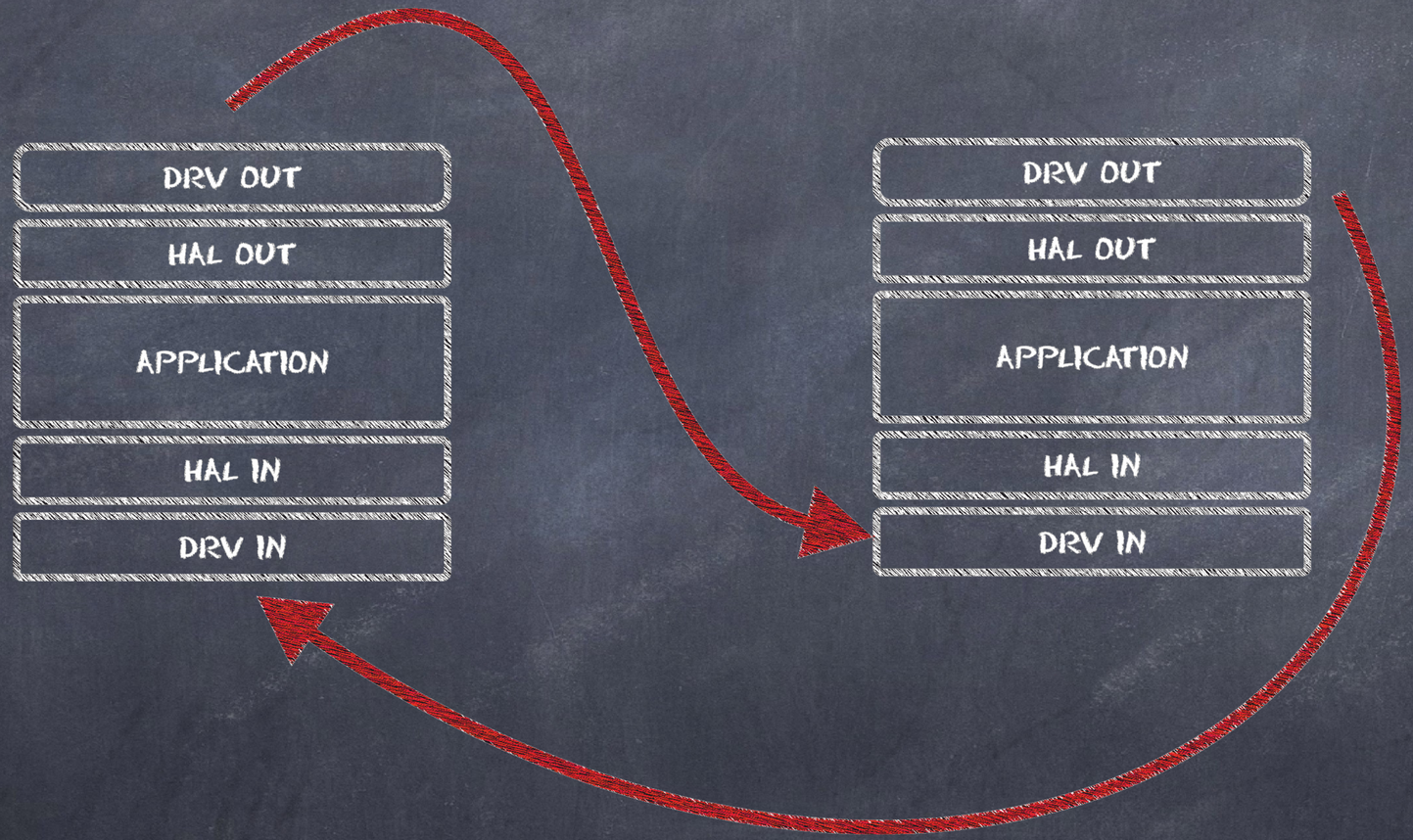
Funktionale Sicherheit



Unabhängige Funktionen



Verteilte Systeme



Zusammenfassung

EINFACH		✓		✓
GEKAPSELTE KOMPONENTEN	✓		✓	✓
OPTIMAL ENTFLOCHTEN	✓	✓		✓
WENIGE ABHÄNGIGKEITEN	✓	✓	✓	✓
STARKE HIRARCHIE			✓	✓
KEINE KONKURIERENDEN ZUGRIFFE		✓		✓

Zusammenfassung

- ☑ TESTBAR
  (ALS KOMPONENTE UND IM MODELL)
- ☑ WARTBAR
- ☑ WIEDERVERWENDBAR
- ☑ AGIL
- ☑ VITAL
- ☑ EFFIZIENT

Elektronik & Informatik OWL e.G.

EIOWL.DE

Danke

THOMAS
LACHTRUP

Elektronik & Informatik OWL e.G.

EIOWL.DE

coming soon

Entwicklereffizienz

- Embedded C
  Programmieren und Simulieren
  mit Visual Studio

- Best Practice

- Software Engineering Barcamp

Thomas Lachtrup

Elektronik & Informatik OWL e.G.

EIOWL.DE