

Automatisierung von Laborequipment mit MATLAB – eine objektorientierte Lösung

Hauke Nannen, Heiko Zatocil

Automatisierung von Laborequipment mit MATLAB - eine objektorientierte Lösung

SIEMENS

1 Status quo

2 Sollzustand

3 Lösung

3.1 Lösungsaufbau in Schichten

3.2 Konzeptüberblick

3.3 Kommunikationsklasse

3.4 Geräteklasse

3.5 Anwendungsbeispiel

4 Zusammenfassung

Automatisierung von Laborequipment mit MATLAB - eine objektorientierte Lösung

SIEMENS

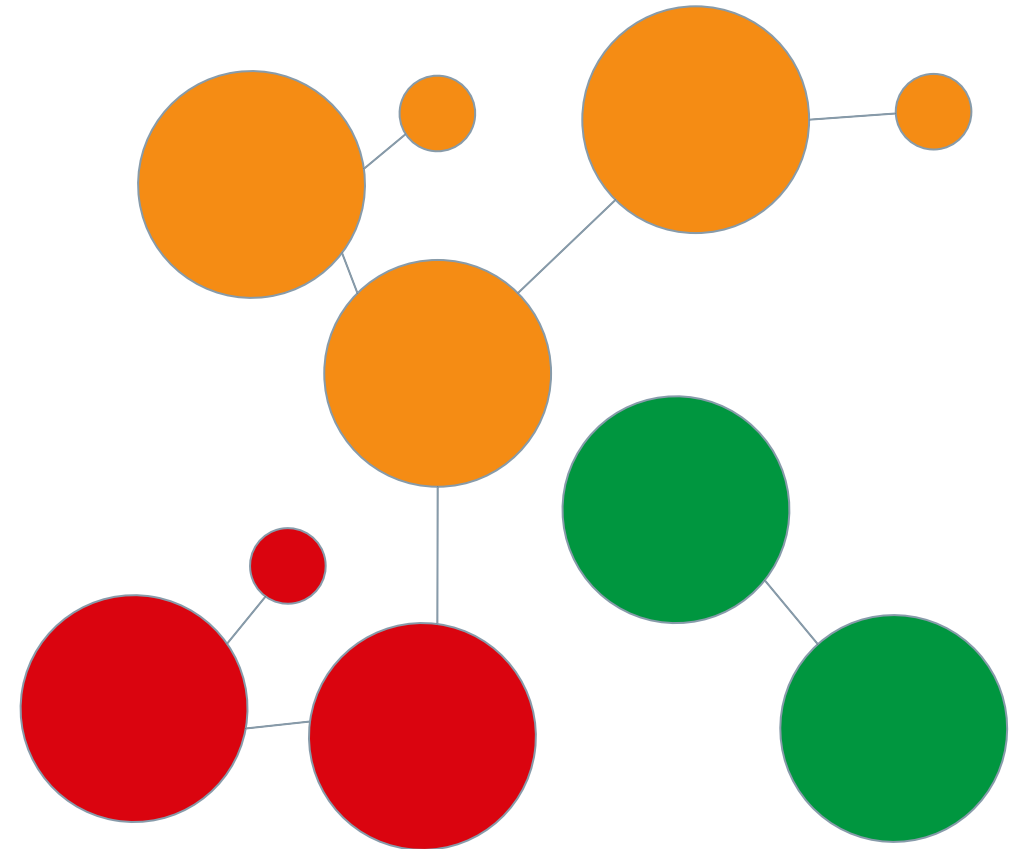
1	Status quo
2	Sollzustand
3	Lösung
3.1	Lösungsaufbau in Schichten
3.2	Konzeptüberblick
3.3	Kommunikationsklasse
3.4	Gerätekasse
3.5	Anwendungsbeispiel
4	Zusammenfassung

Status quo – Situation ohne Lösung für Laborautomatisierung

- Automatisierung in Insellösungen
- Wiederverwendung nur bei exakt identischem Aufbau
- Kernkommunikation wird vielfach neu implementiert
- „lessons learned“ nur bei einer Person
- Jeder User nutzt unterschiedliche Toolboxes
- Immer nur Gutpfad implementiert
→ Fehlerbehandlung und Robustheit sind fraglich
- Heterogener Aufbau des Codes lässt den Aufbau einer Bibliothek kaum zu
- Wiederverwendbarkeit und Wartbarkeit sind zweifelhaft

→ Automatisierung wird aufgrund des initialen Zeitaufwands gescheut

Heterogene, spezifische Einzelimplementierungen



1 Status quo

2 Sollzustand

3 Lösung

3.1 Lösungsaufbau in Schichten

3.2 Konzeptüberblick

3.3 Kommunikationsklasse

3.4 Geräteklasse

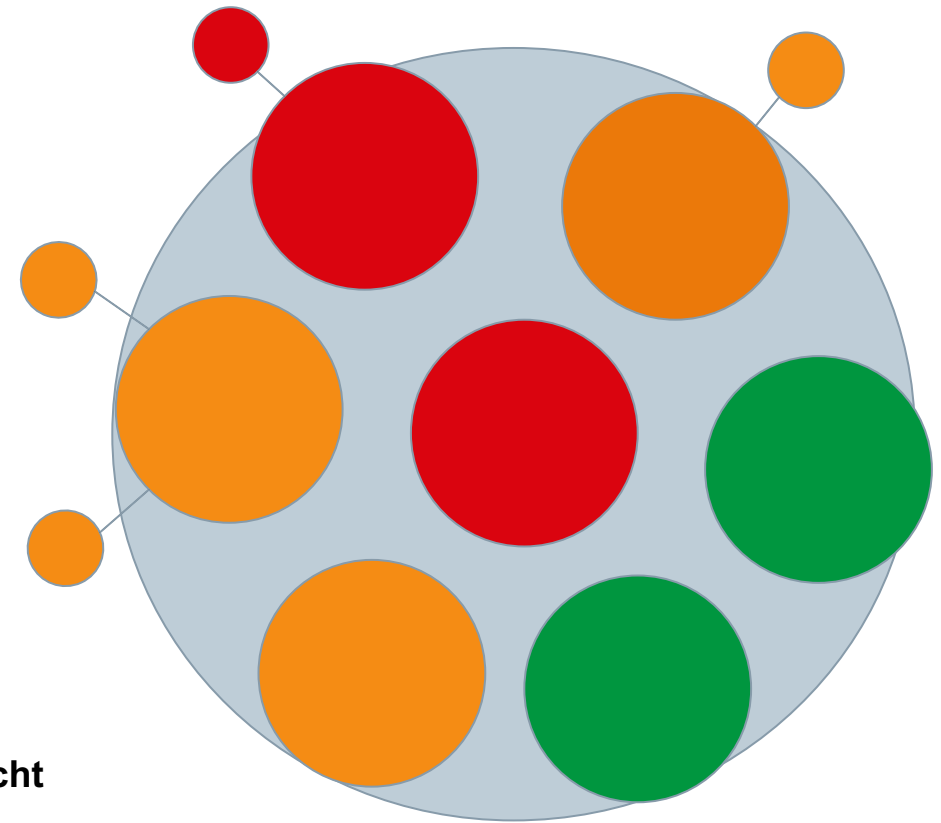
3.5 Anwendungsbeispiel

4 Zusammenfassung

Sollzustand – Anforderungen an ein Lösungskonzept

- Alle Einzellösungen basieren auf zentraler Bibliothek
 - Grundfunktionen sind in Bibliothek abgebildet
 - Verwendung von Funktionen mit intuitiver Schnittstelle in der die Gerätebedienschritte mittels Funktionen nachgebildet werden
 - Fehlerroutrinen (z.B. Verbindungsverlust) sind implementiert und einfach handhabbar
 - Geräte-/HW-Schnittstelle ist flexibel austauschbar
 - Implementierung ist (möglichst) unabhängig von Toolboxnutzung des jeweiligen Users
 - Änderungen in der Bibliothek sind dokumentiert und nachvollziehbar
 - Inkrementelle Ergänzung von Funktionalität im Bedarfsfall ermöglicht ständig steigenden Funktionsumfang
- **Es entsteht eine wartbare, umfassende Messgeräteautomatisierung, die eine schnelle und robuste Umsetzung neuer Messaufgaben ermöglicht**

Messgeräteautomatisierung auf Bibliothekslösung



Automatisierung von Laborequipment mit MATLAB - eine objektorientierte Lösung

SIEMENS

1 Status quo

2 Sollzustand

3 Lösung

3.1 Lösungsaufbau in Schichten

3.2 Konzeptüberblick

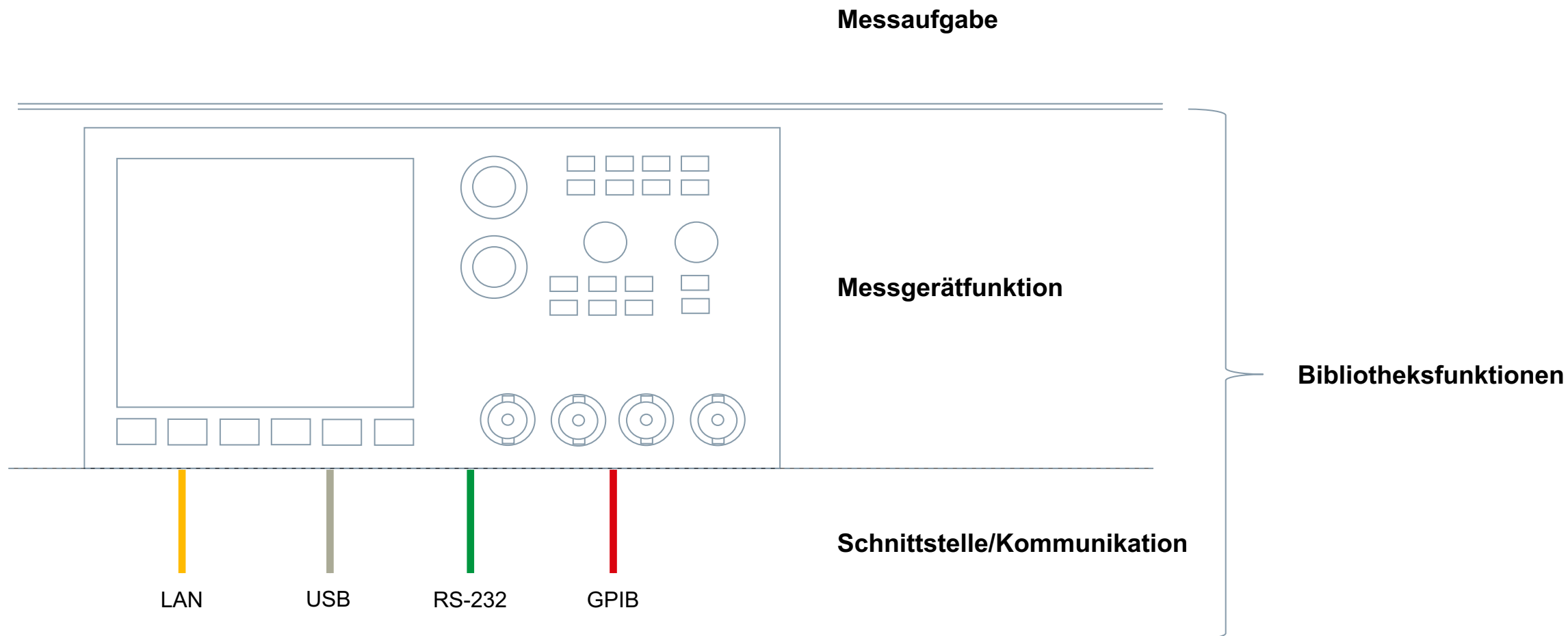
3.3 Kommunikationsklasse

3.4 Geräteklasse

3.5 Anwendungsbeispiel

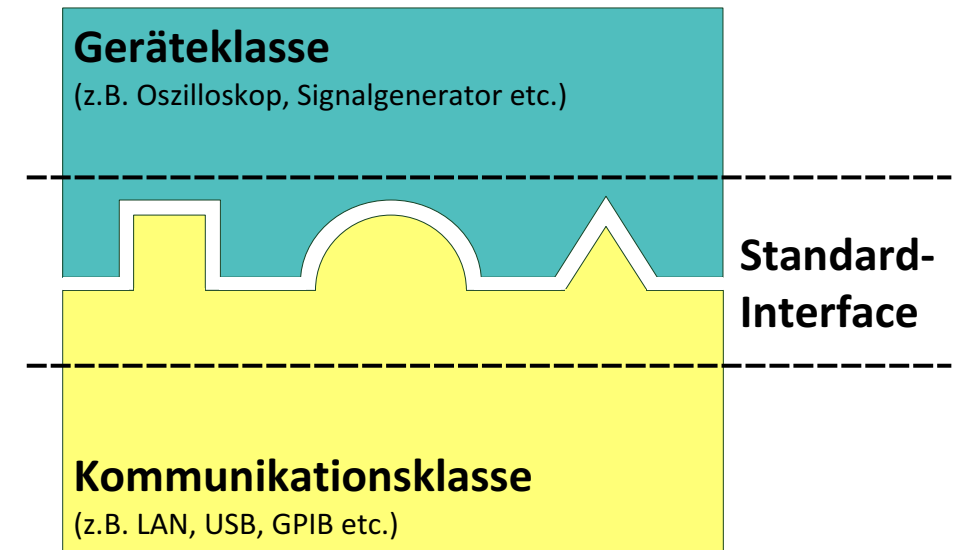
4 Zusammenfassung

Lösungsaufbau in verschiedenen Schichten



Lösung – Überblick über das Konzept

- Lösung ist eine objektorientierte Bibliothek
- Gegliedert in Geräte- und Kommunikationsklassen
- Standardisiertes Interface ermöglicht nachträglichen Austausch der Schnittstelle
- Geräteklassen bilden die Funktionalität des Gerätes ab
- Geräteklasse bildet nur die Funktionalität ab, nicht die Kommunikation
- Ein Standard-Interface für alle Kommunikationsklassen
- Flexibilität bei der Wahl der physikalischen Schnittstelle durch separate Kommunikationsklasse
- Verwaltet werden die Geräte- und Kommunikationsklassen in einem GIT-Projekt d.h. Änderungen sind nachvollziehbar und alte Stände sind wiederherstellbar



Lösung – Kommunikationsklasse als Basis

- Kommunikationsklasse fasst alle zur Kommunikation nötigen Routinen zusammen
- Die Struktur der Kommunikationsklasse ist in einer abstrakten Klasse *connection* definiert
- Verschiedene HW-Schnittstellen (z.B. Seriell, LAN, GPIB) als Ausprägung der Klasse *connection*
- Kernbestandteile sind die Funktionen
 - *connect*: Sorgt für den Verbindungsaufbau zu einem Gerät. Je nach Schnittstelle werden die zur Verbindung nötigen Parameter übergeben
 - *disconnect*: Beendet die Verbindung zu einem Gerät
 - *cmd_get*: Sendet einen Datenstrom an ein Gerät und wartet auf eine Antwort
 - *cmd_set*: Sendet einen Datenstrom an ein Gerät
- Beispielaufruf für eine Spannungsquelle:

```
%           IP-Adresse      Port
con_Supply = tcp_pnet('192.168.100.35', 4000); % Initialisierung der Verbindungsklasse
con_Supply.connect(); % Verbindungsaufbau zum Device
%           Befehl
Return      = con_Supply.cmd_get('SOURce:VOLTage?'); % Abfragen eines Parameters des Geräts
```

Lösung – Geräteklasse als Struktur für Gerätefunktionalität

- Geräteklassen fassen die Funktionen eines Gerätes zusammen
- Bedienungsschritte am realen Gerät werden in Funktionen abgebildet
- Für Geräte mit gleicher Funktionalität (z.B. Spannungsquellen) werden abstrakte Klassen definiert die die Kernfunktionalitäten umfassen
→ sehr einfacher Austausch von Geräten möglich
- Kommunikation basiert auf Kommunikationsobjekt, das dem Objekt bei der Instanziierung übergeben wird
- Einfaches Funktionsinterface ermöglicht intuitive Umsetzung einer Automatisierung

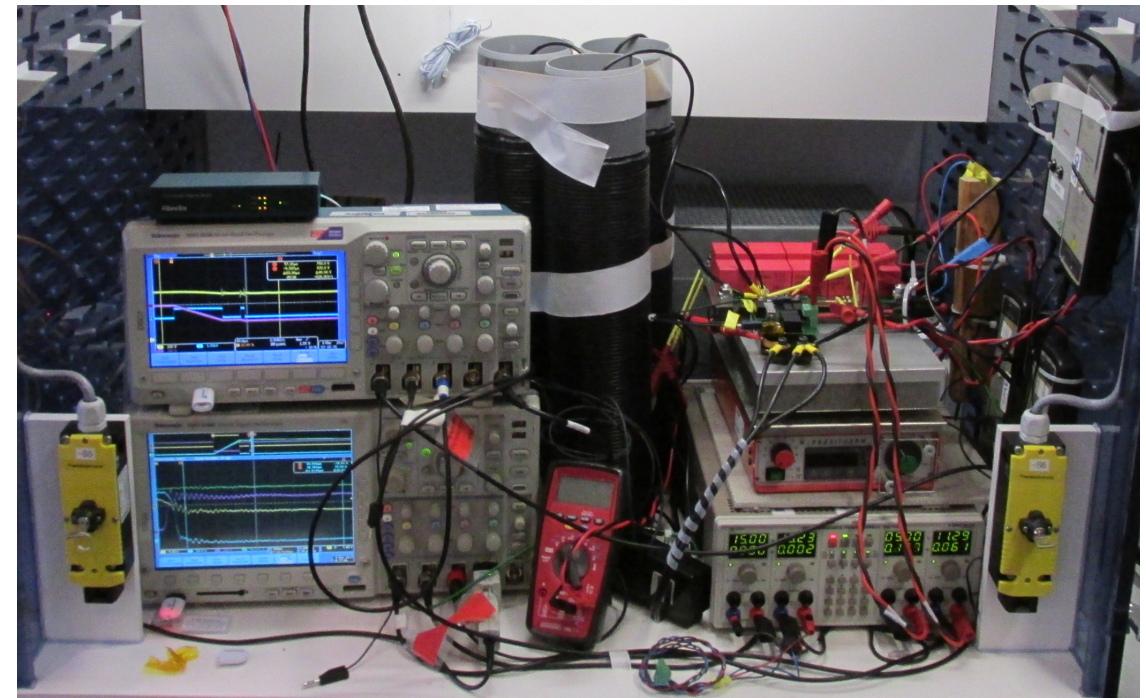
```
%                               IP-Adresse      Port
con_Supply = tcp_pnet('192.168.100.35', 4000); % Initialisierung der Verbindungsklasse (TCP-Verbindung)
Supply     = TDKLamdaGEN3300(con_Supply);      % Initialisierung des Devices(Labornetzteil)
Supply.Set_U(10);                             % Einstellen der Ausgangsspannung des Labornetzteils
```

```
classdef Source < handle
    properties
        Manufacturer
        Model
        SN
        Revision
    end

    methods (Abstract)
        ok = Get_Err(obj)
        ret = Get_U(obj)
        ret = Set_U(obj, voltage)
        ret = Get_I(obj)
        ret = Set_I(obj, current)
        [...]
    end
end
```

Lösung – Einführung in das Anwendungsbeispiel „Doppelpulsversuch“

- Ziel des Versuchsaufbaus:
Schalten mit einem Leistungshalbleiter unter bestimmten Bedingungen (Spannung, Strom, Temperatur), um dessen Schaltverhalten/Schaltverluste charakterisieren zu können
- Vielzahl von Geräten beteiligt: Spannungsquellen, Signalgenerator, Oszilloskope, Heizplatte, Temperaturmessung
- Hohe Wiederholrate bei unterschiedlichen Betriebspunkten macht Automatisierung besonders attraktiv
- Direkte Auswertung ermöglicht unmittelbare Messdatenverifikation



Lösung – Ablauf des Anwendungsbeispiels „Doppelpulsversuch“



```
con_NT      = tcp_pnet('192.168.100.35', 4000);           % Aufbau der Verbindung zur Spannungsquelle
NT          = TDKLamdaGEN3300(con_NT);                  % Instanziierung der Spannungsquelle

con_Multimeter = serial_matlab('COM5');                  % Aufbau der Verbindung zum Multimeter
Multimeter    = Fluke189(con_Multimeter);               % Instanziierung des Multimeters

[...]
```

Lösung – Ablauf des Anwendungsbeispiels „Doppelpulsversuch“



```
OZ.Set_AquState('ON');  
OZ.Set_AquMode('SAMPLE');  
OZ.Set_AquStopAfter('SEQUENCE');  
OZ.Set_Trigger('EDGE','AC','AUX','RISe',1);  
OZ.Set_RecPoints(10e6);
```

% Oszilloskop auf "run" schalten
% Akquisitionsmodus einstellen
% Wahl des "Single-Shot"-Modus
% Trigger auf 1 Volt steigende Flanke
% Einstellen der Record-Length auf einen ausreichend hohen Wert

[...]

Lösung – Ablauf des Anwendungsbeispiels „Doppelpulsversuch“



```
%% Zuschalten der DC-Spannung
NT.Set_U(10);
NT.Set_I(1);
NT.Set_Output_Enable();
fprintf('Spannung angelegt\n');
```

[...]

```
% Setzen der Ausgangssollspannung auf 10 V
% Setzen des Ausgangsmaximalstroms auf 1 A
% Aktivieren des Outputs
```

Lösung – Ablauf des Anwendungsbeispiels „Doppelpulsversuch“

Geräteinitialisierung

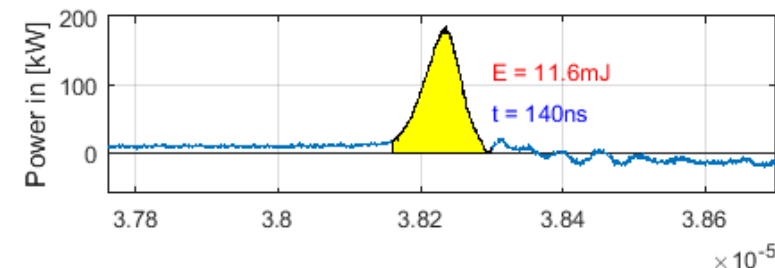
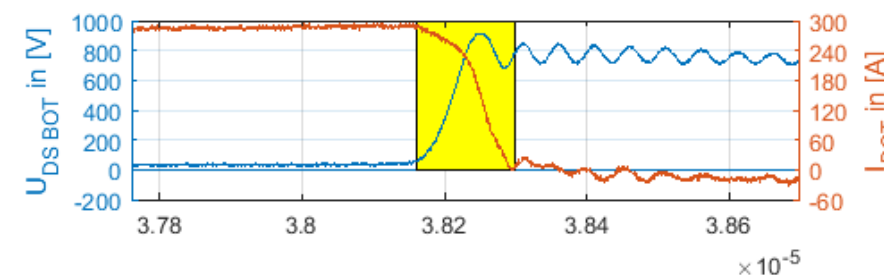
Geräte
parametrierenMessspannung
einstellenMessung starten &
auswerten

```

Signalgenerator.Trigger(1); % Auslösen des Triggers

%% Laden der Daten aus dem Oszilloskop
[Meas.UDS.x, Meas.UDS.y, Meas.UDS.xUnit, Meas.UDS.yUnit] =
    OZ.readwaveform_range('channel1',TimeRead.Start,TimeRead.End);
fprintf('Signal UDS ausgelesen\n');
[...]

%% Auswerten der Daten
[...]
```



Automatisierung von Laborequipment mit MATLAB - eine objektorientierte Lösung

SIEMENS

1 Status quo

2 Sollzustand

3 Lösung

3.1 Lösungsaufbau in Schichten

3.2 Konzeptüberblick

3.3 Kommunikationsklasse

3.4 Geräteklasse

3.5 Anwendungsbeispiel

4 Zusammenfassung

Zusammenfassung

- Objektorientiertes Bibliothekskonzept mit
 - Kommunikationsklassen
 - Geräteklassen
 - Intuitive Schnittstelle zur einfachen Anwendung
 - Sukzessiver Ausbau der Kommunikations-/Geräteklassen ermöglicht Automatisierung ohne hohen Initialaufwand
 - Zunehmende Verwendungsdauer sorgt für stetig steigenden Funktionsumfang
 - Versionierung mit GIT sorgt für Nachverfolgbarkeit von Informationen und geringen Administrationsaufwand
 - Direkte Messungsauswertung in MATLAB ermöglicht umgehende Validierung des Messaufbaus
- Messautomatisierung ist mit massiv reduziertem Aufwand möglich und die Ergebnisqualität steigt deutlich**

Thank you very much for your attention



Hauke Nannen
R&D Engineer
PD TI AT 5

Gleiwitzer Str. 555
90443 Nürnberg

Telefon: +49 911 895 3025

E-Mail:
hauke.nannen@siemens.com

Dr. Heiko Zatocil
R&D Engineer
PD TI AT 5

Gleiwitzer Str. 555
90443 Nürnberg

Telefon: +49 162 365 8492

E-Mail:
heiko.zatocil@siemens.com