BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Projekt-Labor: Automatisierungstechnik

Development and validation of a SoC algorithm for solarbatterychargecontroller Mppt-1210-hus by Libre.solar with lead acid WS2122

handed in

Emile Schons 903606

EDV.Nr.:123 456

of the faculty VII – Elektrotechnik –
of Berlin University of Applied Sciences and Technology handed in Projektarbeit
to finalize the Projekt-Labor
Master of Engineering (M.Eng.)
im Studiengang
Energie und Automatisierungstechnik

Tag der Abgabe October 25, 2021

Gutachter
Gutachter1 Berlin University of Applied Sciences and Technology

Contents

List of Figures

Introduction

The git source code repository of this document [Schons 2021] is available at [1]
(4d) (mostly offline)

1.1 why Kalman, why EKF?

[Espedal et al. 2021] Current Trends for State-of-Charge (SoC) Estimation in
Lithium-Ion Battery Electric Vehicles.
The kalman filter is a great tool to base a SoC algorithms on, as it is clearly defined by mathematical theories more than by a arbitrary algorithm designed by a programmer. One needs to use a extend kalman filter as the problem to solve is non linear, TODO explain why it is non linear.
Other methods are based on auto regressive exogenous (ARX) model or a combination of it and the Kalman Filter. TODO reference to resources.

[1] `https://github.com/mulles/Doc_Projekt-Labor_SOC`

Implementation

Link to source code. [1] The algo is a fork of a kalman-soc from person xxx of the company okra-solar.
Some adjustments have been made to improve or adapt it to libre.solar use case and code base.
Changelog:
-adoption of libre solar code style guide
-change of energy counting to coulomb counting
-use of float instead of integer in order to guarantee correct calculations, with no overflow.
-
-exact changes are traceable with the commits to the fork.

2.1 Hardware Setup for validation

TODO Migrate from the other document.
TODO What resolution is necessary.

2.2 EKF Implementation

date: (4d)
A good step by step guide to implement the extended kalman filter is [Rzepka et al. 2021].
To design an extended kalman filter algorithm three steps are necessary: 1.The control-input model and the observation model need to be well designed. 2.The constants inside the models expressed as matrices need to be defined, which is as well known as parametrization of the model. 3.The model needs to be implemented as code running on embedded systems.
The key to success for a good SoC is a good definition of both models and a good parametrization. The implementation is independent of the use case and depends on the skills of the programmer, in his hands lies the optimization of the code. In case no FPU is available on the micro-controller unit (MCU) fix point arithmetic might be an option.
Particularly challenging is the parametrization, which is typically done offline [2] and requires quite advanced equipment. The Adaptive extend Kalman Filter (AEKF) is an option, where the parameters are determined online, but still AEKF needs to be initialized with parameters not necessary fitting all batteries of a given chemistry, because of different capacities or tolerances and specific designs. TODO It needs to be researched if it only affects the convergence time or if convergence is not reached at all? A further advantage of an adaptive filter is that the state of health (SOH) can be determined along the side. When simple observation and control-input models are used as in the okra kalman-soc algorithm, the parametrization might be much easier? TODO to check if the initial parameters, in this case OCV in function of SoC data can be used generally or only on the

[1] `https://github.com/mulles/kalman-soc`

[2] not generated by the micro controller unit (MCU) the algorithm is deployed to, thus not on the system running the final algorithm. Offline means on more advanced hardware, which is not available in the final product. See `https://en.wikipedia.org/wiki/Online_model`

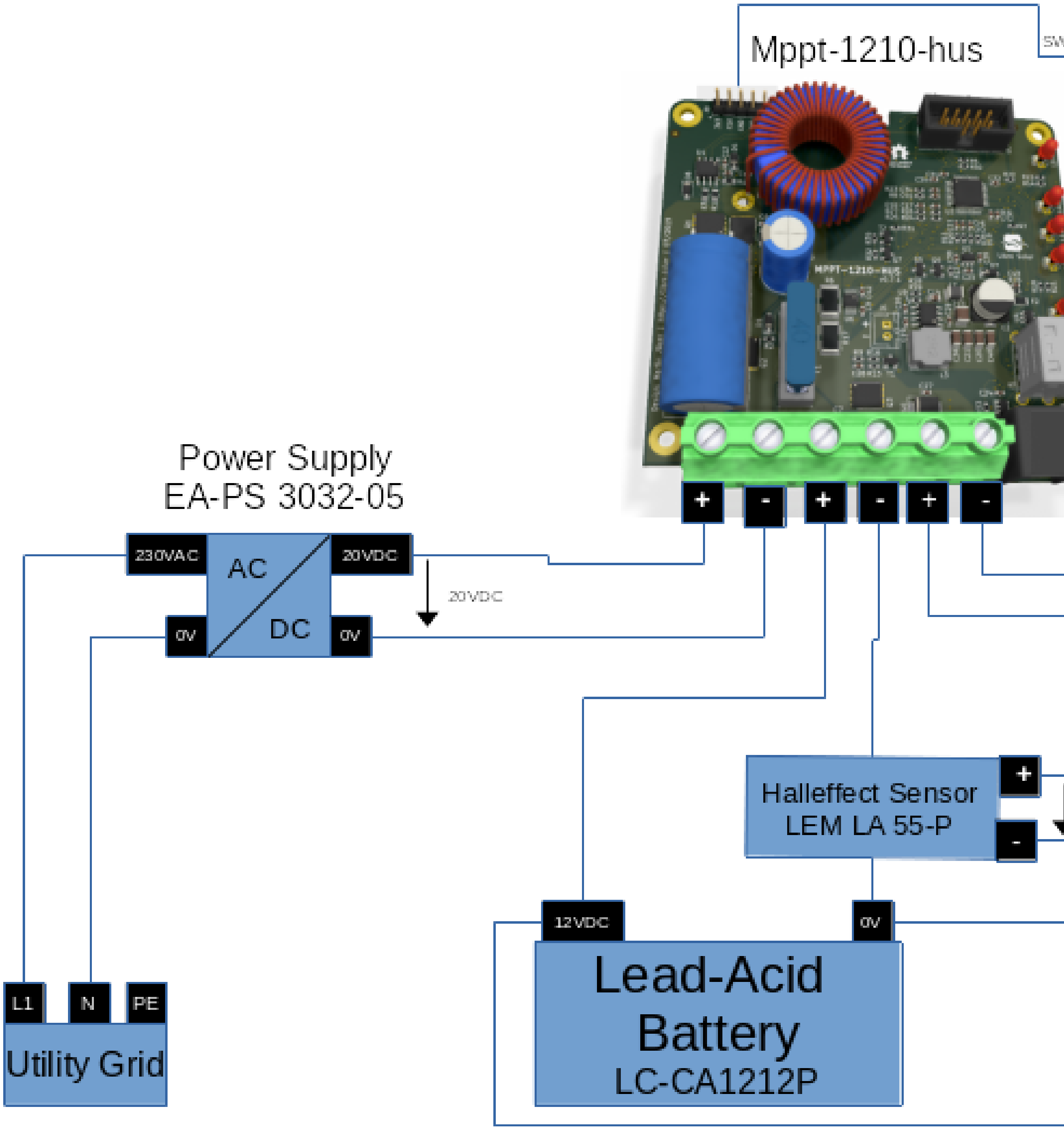

Figure 2.1: Setup to verify SoC Algorithm

batteries, the data has been generated with.

Citations concerning the on/off-line parametrization

Offline parametrization:
'As a result, many experimental pretests investigating the effects of the internal and external conditions of a battery on its parameters are required, since the accuracy of state estimation depends on the quality of the information regarding battery parameter changes'
'Therefore, some tests data must be available in advance to find the parameters of these models.' [Hussein und Batarseh 2011]

Gregory Plett says:
'Two possible approaches:

First, an algorithm might somehow adapt the parameter values of the model during operation to match presently observed current–voltage behaviors; but, this **must be done very carefully to avoid making the model unstable or physically nonmeaningful**.

Alternately, a set of models could be pre-computed at different feasible aging points and the model from this set that most closely predicts presently observed current–voltage dynamics could be selected from the set. This second approach guarantees stable and physically meaningful models since all models in the pre-computed set meet these criteria. We propose such an approach here.' —https://www.sciencedirect.com/science/article/abs/pii/S2352152X18301385?viaIn order to have online first algo, the parametrization needs to be done online!.
As the OCV curve is non-linear the kalman filter can not be used.
state transition and observation models

2.2.1 Definition Mathematics Equations

TODO Make a diagram where you can see the input output and steps

Most general form of state observer equations:
$x_{k+1} = Ax_k + Bu_k$ (state observer equation)
$y_k = Cx_k + dDu_k$ (output equation)

$u_k \equiv i_k$ a control vector (input), defined as the measurement of current through the battery at time k.
$y_k \equiv v_k$ an observation (or measurement), defined as a voltage measurement of the battery at time k.
$A = I$ identity matrix f.i $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$B = \begin{bmatrix} -\frac{\Delta t}{Q_C} & 0 \\ 0 & 1 \end{bmatrix}$ is **the control-input model**, deduced from $f(\boldsymbol{x}_k, \boldsymbol{i}_k, \Delta t) = x_k - \frac{1}{Q_C} \int_0^{\Delta t} i_k \, dk$
$C = [0, -R_1, -R_2, ..., M] = H(x_k)$ and $H(x_k)$is the **observation model** , which maps the state space into the observed space and TODO understand the relationship to $h(x_k)$
$D = R_0$

Most general form of Kalman Filter equations:

$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k$ (State Space equation)
$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k$ (Output equation)

$\hat{x}_{k+1} = (A - BK)\,\hat{x}_k + L\,(y_k - \hat{y}_k)$ (here u_k seems missing)

Predicted variables $\hat{y}_k$ and $\hat{x}_k$ are commonly denoted by a "hat" to distinguish them from y_k and $x(k)$ of the physical system. As the state of charge (SoC) denoted as $\hat{x}_k = SoC_k$ cannot be measured directly it is always a predicted variable, opposed to $y(k)$, which is the measured circuit voltage. Consequently $\hat{y}_k = v_k$ is the predicted circuit voltage also known as measurable output:
$\hat{y}_k = (C - DK)\,\hat{x}_k$
Input to the extend kalman filter (EKF) is current and voltage measurement and the period of time between these measurements. The initialization of the EKF outputs the initial estimate state of charge $x_{k|k=0}$ another input to the EKF.
System's dynamic model
The $f()$ function is defined as the $f(\boldsymbol{x}_k, \boldsymbol{i}_k, \Delta t) = x_k - \frac{1}{Q_C}\int_0^{\Delta t} i_k\,dk$ or more discrete $f(x_k, i_k, \Delta t) = x_k - \frac{\Delta t}{Q_C} i_k$, thus current measurement i_k The $h(\boldsymbol{x}_k)$ function is defined as the OCV lookup table. A general OCV lookup table for the battery chemistry can be used or for better results a specific OCV lookup established by offline measurements for the given battery should be used. To further improve the OCV prediction a correction of it can be performed by a equivalent circuit model (ECM) of the battery feeded by the current measurement used to predict the SoC: $v_k = D\,OCV(z_k) + Cx_k + Di_k$ with $C = [0, -R_1, -R_2, ..., M]$ and $D = R_0$
-Measurement equation, input (measured voltage, OCV lookup table, current if the correction with a Enhanced Self-Correcting (ESC) Cell Model /ECM) -> output SOC) equation should be use standard letters: filterpy, wikipedia, gregoryPlett, Step by Step Guide
After having defined the observation and control-input model as matrices and described their meaning in case of SoC estimation we proceed to the functioning of the EKF, which is typically divided into to steps, Predict and Update.
In the **predict** step a future state of charge estimate $\hat{x}_{k+1}$ is predicted based on the current state of charge estimate $\hat{x}_k$ and the current i_k during the period Δt (between k and $k+1$) by calculating $\mathbf{\hat{x}_{k+1}} = \mathbf{Ax_k} + \mathbf{Bu_k}$ Moreover an estimate of the covariance $P_{k+1}x$ is calculated based on noise covariances Q_k and current P_k (wiki: a measure of the estimated uncertainty of the prediction of the system's state) $\mathbf{P}_{k+1} = \mathbf{B}_k\mathbf{P}_k\mathbf{B}_k^\mathsf{T} + \mathbf{Q}_k$
In the **update** step
$\mathbf{K}_k = \mathbf{P}_{k+1}\mathbf{H}_k^\mathsf{T}(H_k\mathbf{P}_{k+1}\mathbf{H}_k^{\mathsf{T}+\mathbf{R}_k)^{-1}}$ is the kalman gain, which weights whether the SoC based on the measurement of the circuit voltage v_k is more trusted than the SoC prediction based on current i_k
$\mathbf{\hat{x}}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)(\mathbf{\hat{x}}_{k+1}) + (\mathbf{K}_k)(\mathbf{H}_k\mathbf{\hat{x}}_k + \mathbf{v}_k)$ TODO update $\hat{x}_k$ because one can not now want one want to calculate

$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\,\mathbf{P}_k$ update of the covariance TODO is H_k the same as OCV?

$\mathbf{R_k}$ the covariance of the observation noise

Most general equation of extended kalman filter EKF:

$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k$ (state transition model)
$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k$ (observation model)
f -> predicted state from the previous estimate
h -> compute the predicted measurement from the predicted state

Predict
$\hat{\boldsymbol{x}}_{k|k-1} = f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k)$
$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_k^\top + \boldsymbol{Q}_k$

Update
$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})$
$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^\top + \boldsymbol{R}_k$
$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^\top \boldsymbol{S}_k^{-1}$
$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k$
$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k|k-1}$

2.2.2 Kalman-SoC

A Fork of Okra-Solar Algorithm.
Features:
-Works without the input of a Equivalent Circuit Model (ECM) specific to the physical battery, which would need to be parameterized doing advanced measurements during charging and discharging of the battery.
-Inputs: Current and Voltage Measurements and OCV Lookup Table
-Outputs: SoC in %Wh
TODO Discuss the Code Snippets? But Some Code in Appendix or reference Github Repo only?

Validation

2 Arten von Datensätze:
Vergleich alter Algo neuer Algo, mit Daten von einem Device. X mit Zeit? Vllt schafft man dies im gleiche Schritt wie man die Zeiträume verbessert? Y Achse in 100
Titel SoC Libre.solar Algo vs Kalman-soc auf Datensatz von Device xy im Zeitraum xx. Kein Titel in der Graphik selber.
Mit Matplotlib da ist wsl mehr Hilfe verfügbar? Schnell mal mit Plotly versuchen.
-Prozent Abweichung dazwischen. Warum ist er besser? Scheint schwer zu diskutieren.
-Meine Daten?
-Neue Daten generieren? Der Prozess ist ja eigentlich recht automatisiert.
It is difficult to say that how much the kalman-soc algo is better, but sure is that both are different and that there are many cases where it is sure that libre.solar performs poorly, show examples timeframes.
Discuss results. (4d)

3.1 Graphs with Matplotlib or plotly

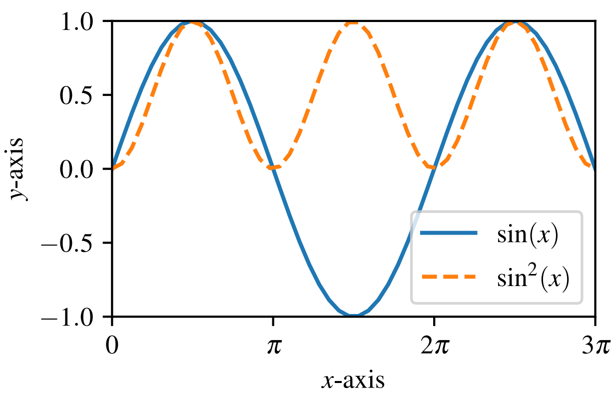

Figure 3.1: A plot created with PythonTeX and Matplotlib

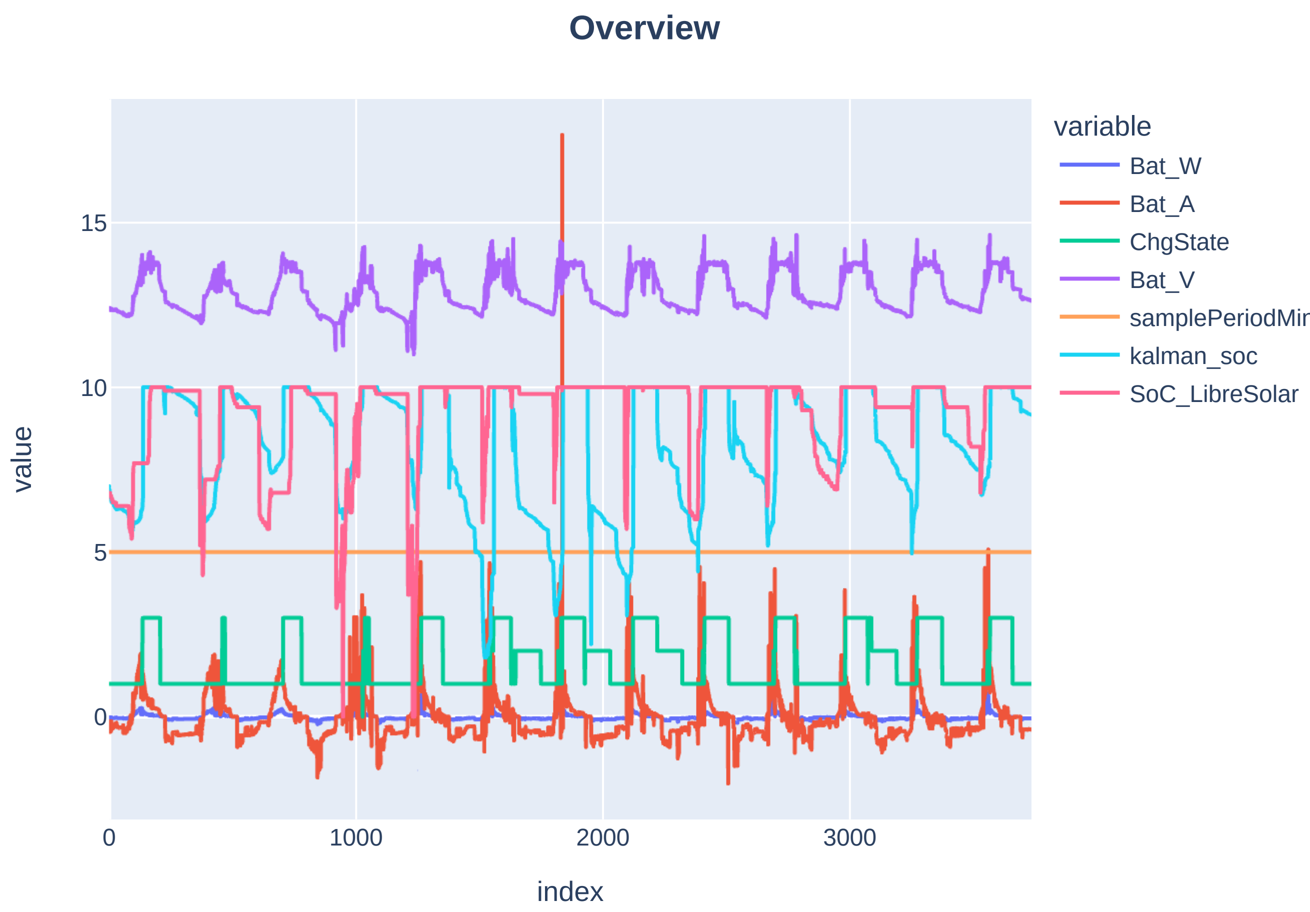

Figure 3.2: Overview of data

Figure 3.3: Overview of data

Outlook

Promising articles and/or methods for online parameter estimation:

online parameter estimation from the ARX model tran2017state
[Wang et al. 2021] xia2018online

Appendix

Stylefile

Die Styledatei für diese Abschlussarbeit ist `bhtThesis.sty`, die in der Archivdatei vorliegt. Diese muss von LaTeX auffindbar sein, muss also in einem LaTeX bekannten Ordner liegen:

- Ubuntu-Linux: `$HOME/texmf/tex/latex/bhtThesis/bhtThesis.sty`
- MikTeX: `c:\localtexmf\tex\latex\bhtThesis/bhtThesis.sty`

Beispieldokument

Dieses Dokument befindet sich im Unterordner `tryout` des zip-files. Sie können diese Dateien in einen Ordner kopieren, in dem Sie schliesslich arbeiten werden. Die Dateien sind die folgenden

- `abstract_de.tex` Kurzfassung in deutscher Sprache
- `abstract_en.tex` Kurzfassung in englischer Sprache
- `anhang.tex` der Anhang
- `bhtThesis.bib` beinhaltet die zu zitierenden Literaturstellen und wird von bibTeXausgewertet
- `main.pdf` ist die Ausgabendatei mit der Druckvorlage
- `main.tex` beinhaltet das Hauptdokument
- `makefile` realsiert das automatische mehrfache Übersetzen, hierfür muss `make` auf dem System installiert sein.
- `myapalike.bst` beinhaltet die Formatierung für das Literaturverzeichnis
- `personalMacros.tex` kann einzelne, persönliche Macros beinhalten, die das Schreiben erleichtern
- `titelseiten.tex` realisiert alle Seiten bis zum Beginn des ersten Abschnittes
- Ordner `pictures`
  - `BHT-Logo-Basis.eps`
  - `BHT-Logo-Basis.pdf`
- Ordner `kapitel1`
  - `ch1.tex` Quelltext des Kapitel 1
  - Ordner `pictures`
    - `schaltbild.pdf`

- Ordner `kapitel2`
  - `ch2.tex` Quelltext des Kapitel 2
  - Ordner `pictures`
    - leer

Bibliography

[Espedal et al. 2021] Espedal, I. B., Jinasena, A., Burheim, O. S., und Lamb, J. J., Current trends for state-of-charge (soc) estimation in lithium-ion battery electric vehicles. *Energies*, 14(11):3284.

[Hussein und Batarseh 2011] Hussein, A. A.-H. und Batarseh, I., An overview of generic battery models. In *2011 IEEE Power and Energy Society General Meeting*, Seiten 1–6. IEEE.

[Rzepka et al. 2021] Rzepka, B., Bischof, S., und Blank, T., Implementing an extended kalman filter for soc estimation of a li-ion battery with hysteresis: A step-by-step guide. *Energies*, 14(13):3733.

[Schons 2021] Schons, E., Development and validation of a SoC algorithm for solarbatterychargecontroller Mppt-1210-hus by Libre.solar with lead acid.

[Wang et al. 2021] Wang, Y., Huang, F., Pan, B., Li, Y., und Liu, B., Augmented system model-based online collaborative determination of lead–acid battery states for energy management of vehicles. *Measurement and Control*, 54(1-2):88–101.