Slider 2

Otahal, Holecek, Jevtic,

Sprint 2

Sprintdokumentation

31.03.-14.04.2016

Inhaltsverzeichnis

[1 Sprintbericht 3](#_Toc448404466)

[1.1 Tasks 3](#_Toc448404467)

[2 Konzept der GUI fertigstellen 5](#_Toc448404468)

[2.1 Allgemein 5](#_Toc448404469)

[2.2 Programm Aufbau 5](#_Toc448404470)

[2.3 Durchführung 5](#_Toc448404471)

[2.4 Fertiggestellte GUI 6](#_Toc448404472)

[3 Laser in Betrieb nehmen 7](#_Toc448404473)

[3.1 Allgemeines 7](#_Toc448404474)

[4 Logik der GUI mittels Python 8](#_Toc448404475)

[4.1 Prototyp 8](#_Toc448404476)

[5 Graphische Darstellung von Testdaten 13](#_Toc448404477)

[5.1 PyQT Designer Konvertierung 13](#_Toc448404478)

[6 Webserver einrichten 15](#_Toc448404479)

[6.1 Einleitung 15](#_Toc448404480)

[7 Erstellung bzw. Überarbeitung eines Webinterfaces 15](#_Toc448404481)

[7.1 Einleitung 15](#_Toc448404482)

[7.2 Durchführung 15](#_Toc448404483)

[8 Burndown Charts 18](#_Toc448404484)

[8.1 Sprint Burndown Chart 18](#_Toc448404485)

[8.2 Product Burndown Chart 18](#_Toc448404486)

# Sprintbericht

## Tasks

### Von Sprint übernommen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Name | Zugewiesen | Akzeptanzkriterium | anfängl.  Schätz. | verstri-chen | verblei-  bend | Status |
| 1269 | Laser in Betrieb nehmen | MO | Laser muss laufen | 2 hrs |  |  | fertiggestellt |
| 1319 | Konzept der GUI fertigstellen | MO | GUI muss den Anforderungen entsprechen | 3 hrs |  |  | fertiggestellt |
| 1321 | Logik der GUI mittels Python | MO | GUI in Python lauffähig | 4 hrs |  |  | fertiggestellt |
| 1282 | Grafische Darstellung von Testdaten | RH | GUI soll mittels den Testdaten angezeigt werden | 6 hrs |  |  | fertiggestellt |
| 1309 | Laser mit Beaglebone verbinden | RH | Tesstdaten bekommen | 3 hrs |  |  | Nicht fertiggestellt |
| 1273 | Webserver einrichten | BJ | Zugriff auf Server möglich | 4 hrs |  |  | fertiggestellt |
| 1333 | Erstellung bzw. Überarbeitung des Webinterfaces | BJ | Webinterface wird angezeigt |  |  |  | fertiggestellt |

### Neu hinzugefügt

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Name | Zugewiesen | Akzeptanzkriterium | anfängl.  Schätz. | verstri-chen | verblei-  bend | Status |
| 1271 | Serielle Kommunikation herstellen | MO | Testdaten |  |  |  | Nicht fertiggestellt |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# Konzept der GUI fertigstellen

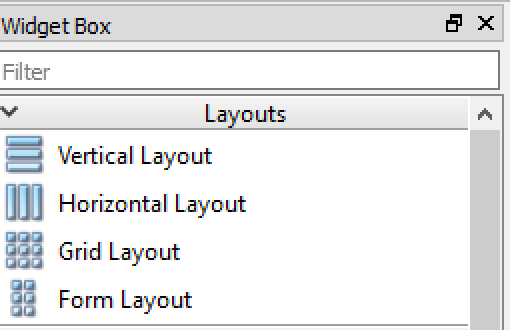
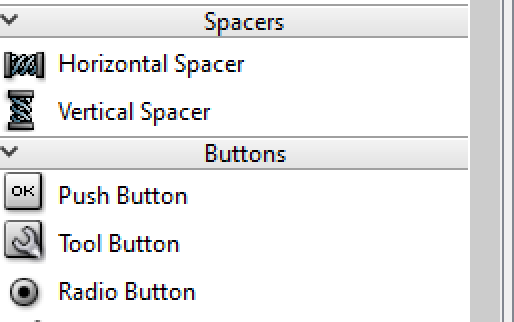
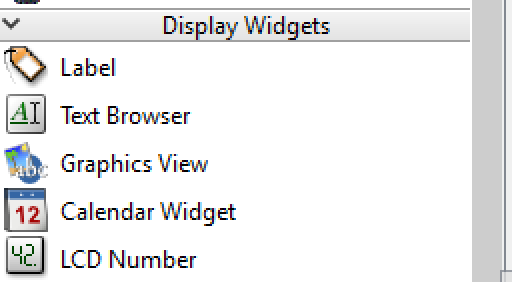
## Allgemein

PyQt ist eine der am häufig verwendeten Python Bindigs für die Qt-Cross-Plattform.

Für dieses Fenster wird die Version PyQt 4 verwendet.

## Programm Aufbau

Aus der Widget-Box kann man verschiedene Widgets auswählen (je nachdem welche man benötigt) und diese dann ins Fenster hineinziehen und anschließend beliebig bearbeiten.

Man kann ebenfalls verschiedene Widgets in ein Layout (z.B Vertical Layout) hineinpacken um dann das Fenster besser zu gliedern.

Diese Aufteilung hat mehrere Vorteile um die Widgets besser zu gliedern beziehungsweise aufzuteilen.

Für unser Projekt haben das Widget Graphics View und Label öfter verwendet.

Sowie Date/Time Edit um das aktuelle Datum und Uhrzeit anzuzeigen.

Weiterst haben wir uns überlegt, einen QR Code im Fenster anzuzeigen, welches man mittels einem QR-Code Scanner fotografieren kann und anschließend an die Website gelangt, in der man den Laser steuern kann.

## Durchführung

### Graphics View

Um in PyQt eine Graphics View richtig anzuzeigen muss man im Ressource Browser die Images bekannt geben.

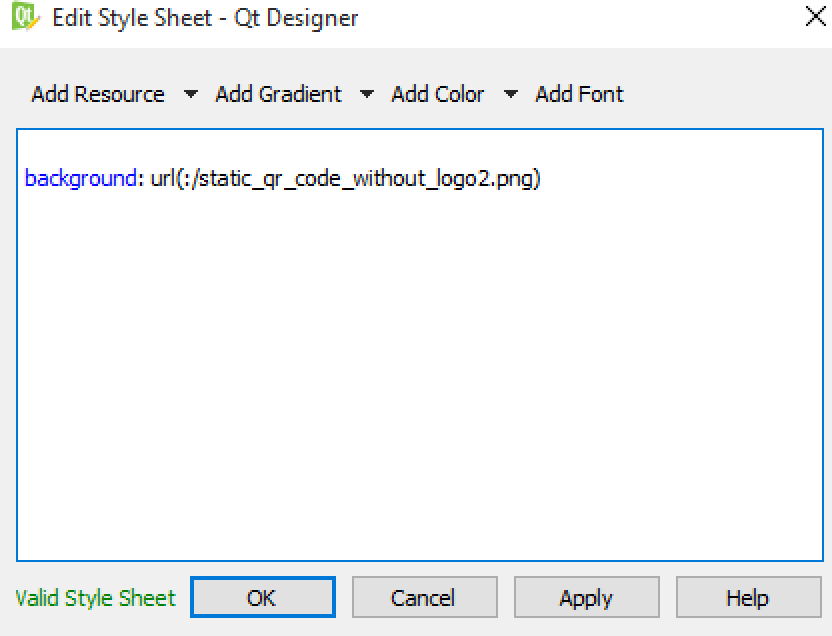
Ressource Browser -> Edit Ressources -> Prefix/Path -> Add Files

Anschließend wählt man das gewollte Bild aus, welches aber ein .png als Dateiformat sein muss!

Darauffolgend muss man:

Rechtsklick -> Change styleSheet

Wie im Bild unten angezeigt wird die background url des Bildes angegeben.

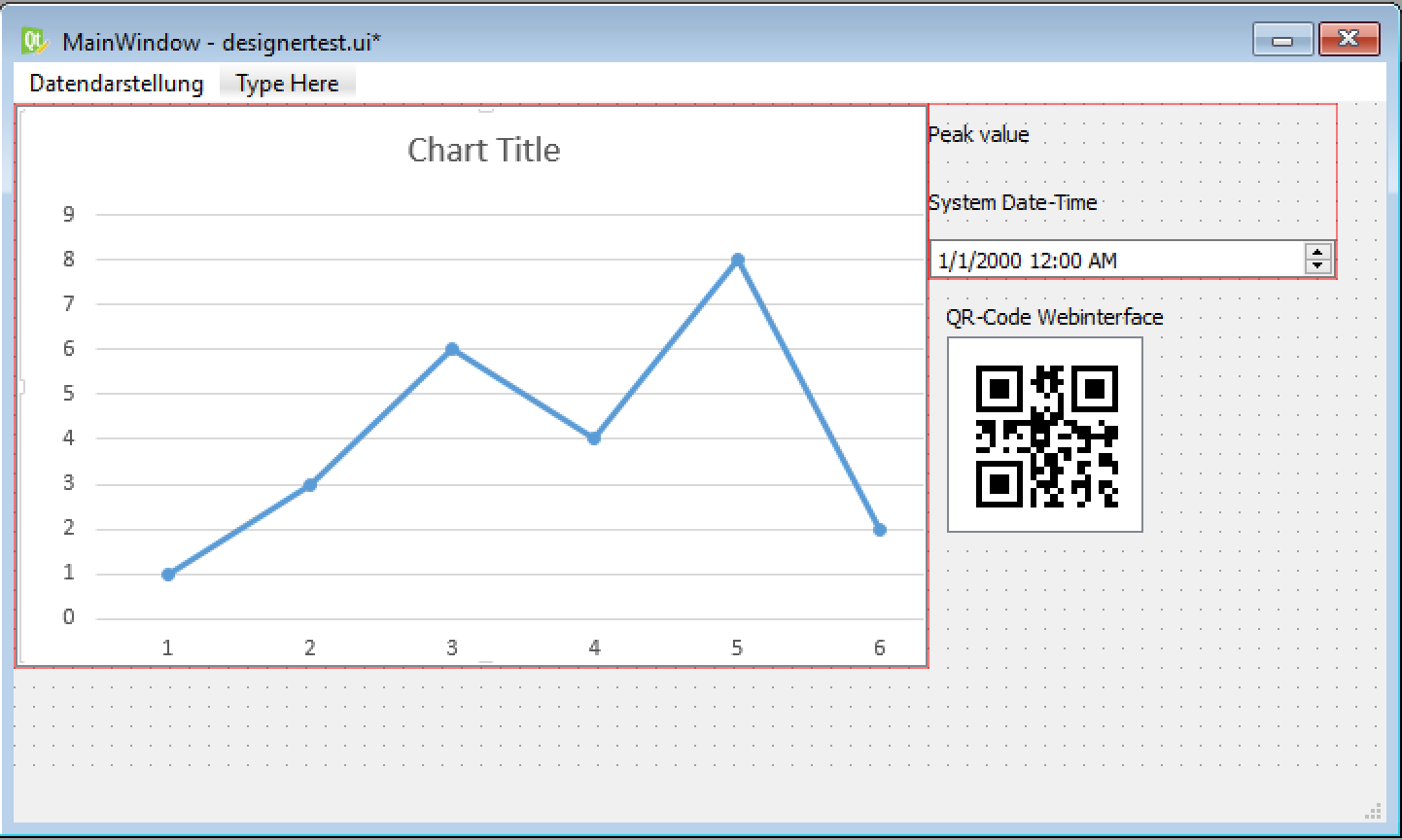


Im Anschluss wird das Bild in der Graphics View angezeigt.

### Date/Time Edit

Um das aktuelle Datum abzurufen wird das Date/Time Edit Widget verwendet

## Fertiggestellte GUI



# Laser in Betrieb nehmen

## Allgemeines

Das LDM41/42 ist ein Laser-Distanzmessgerät, welches Entfernungen im Bereich von 0,1 m bis über 100 m punktgenau misst. Durch den roten Lasermesspunkt ist das Messziel eindeutig zu identifizieren. Die Reichweite ist abhängig vom Reexionsvermögen und der Oberflächenbeschaffenheit des Messziels.

Das Gerät arbeitet auf Basis der Phasenvergleichsmessung. Dabei wird hochfrequent moduliertes Laserlicht ausgesendet. Das vom Messobjekt diffus reflektierte und phasenverschobene Licht wird mit dem Referenzsignal verglichen. Aus dem Betrag der Phasenverschiebung lässt sich die Distanz millimetergenau bestimmen.

Das Auslösen einer Distanzmessung kann auf verschiedene Arten erfolgen:

* Senden eines Kommandos mittels PC oder einer anderen Steuereinheit
* entsprechende Parametrierung des Autostartkommandos und Anlegen der Versorgungsspannung
* durch externe Triggerung (im Fremdtrigger-Mode)
* Nutzung der Autostart-Trigger-Funktion.

Besondere Merkmale sind:

* Betrieb im extremen Außentemperaturbereich mit hoher Genauigkeit und Reichweite möglich.
* großer Betriebsspannungsbereich 10 V= bis 30 V= aus dem KFZ-Bord netz, einem Industrie-Gleichspannungsnetz oder einem Gleichspannungsnetzteil.
* geringe, konstante Leistungsaufnahme <1,5 W (ohne IAlarm).
* Reichweite bis 30 m für Distanzmessungen, mit zusätzlichen Reaktoren auf dem Zielobjekt über 100 m möglich (in Abhängigkeit von der Re ektivität und den Umgebungsbedingungen).
* einfaches Anvisieren des Zieles durch einen sichtbaren Laserstrahl.
* Eingabe der Befehle für die Messfunktionen und Ausgabe der Messwerte über einen PC oder Laptop mit RS232-Schnittstelle möglich.
* getrennte Programmierung von Schaltausgang und Analogausgang.
* Signalisieren der Distanzüber- und Unterschreitung am Schaltausgang mit einstellbarer Grenze.
* Messwertanzeige in Meter, Dezimeter, Zentimeter, Feet, Inch, u.a. durch freie Skalierung.
* Fernauslösung einer Messung von einer externen Triggereinrichtung möglich.

# Logik der GUI mittels Python

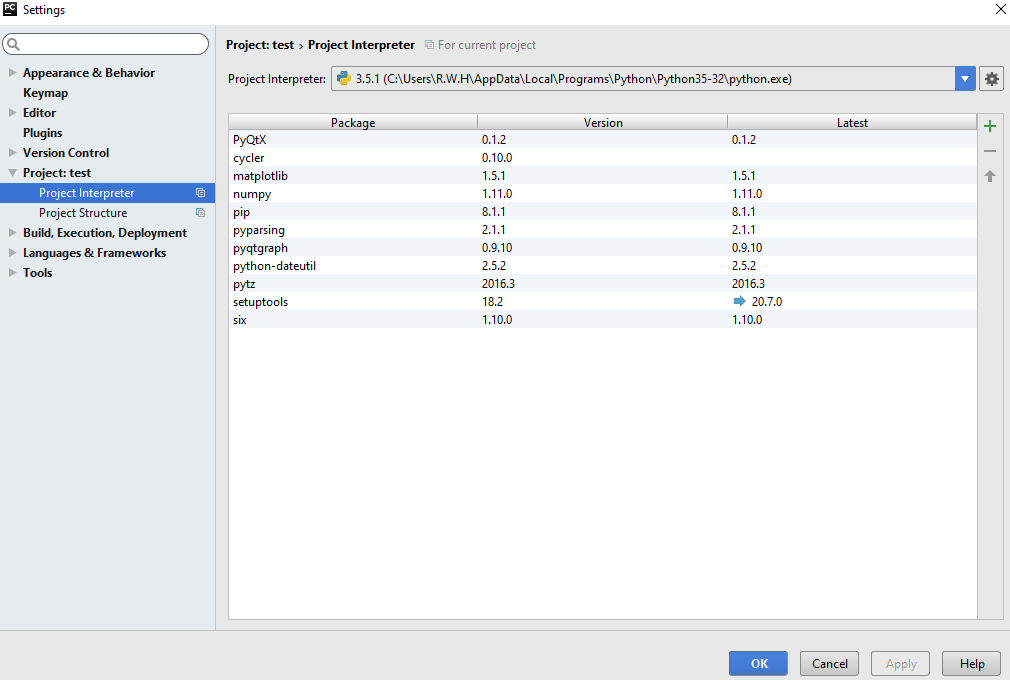
## Prototyp

Der erste Gedanke der zur Konzipierung der GUI einfiel war sie in python(Pycharm) selber zu realisieren. Dies wurde mit den Modulen Tkinter (GUI) und Matplotlib(Graph) ermöglicht. Als IDE wurde eine Windows x64 version von Netbeans Pycharm verwendet.

### Installieren der Module

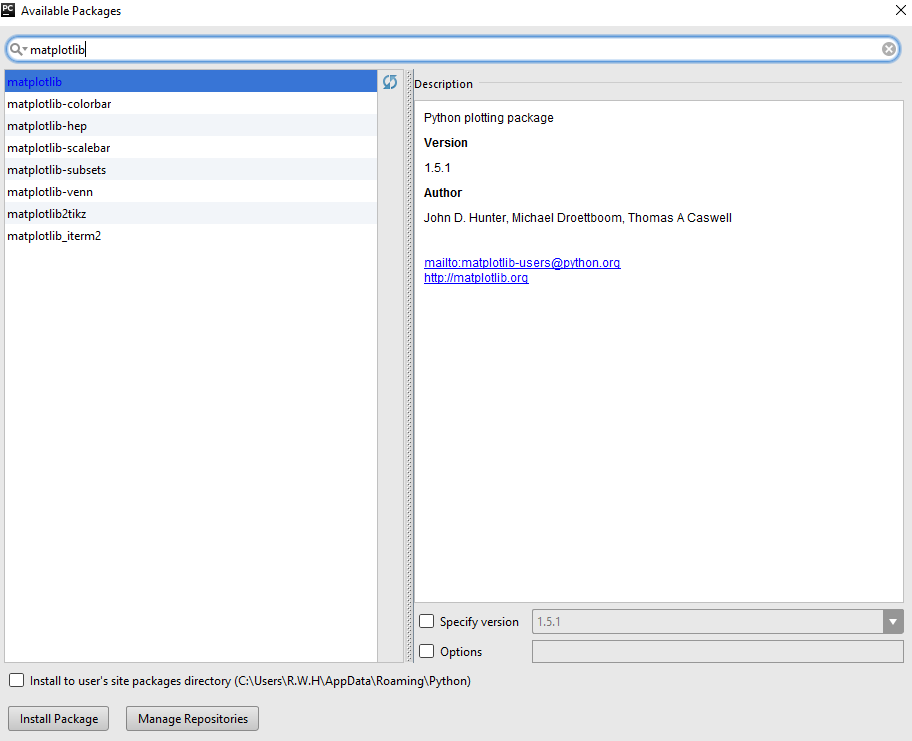
Tkinter ist bei Python 2.7 per default dabei und wird mit den Befehl *from tkinter import \** in eine Klasse importiert. Wenn man nicht den Konstruktor von Tkinter direkt aufrufen will wenn man ihm eine Variable zuweist, kann man den Befehl *import tkinter as varname* verwenden*.*

Das Modul Matplotlib hingegen muss mittels pip noch extra installiert werden. In Pycharm wird die installation über pip mit einer GUI erleichtert. Diese wird in *File -> Settings -> Project -> Project Interpreter* gefunden. Zu der Suche der verschiedenen Module gelangt man über das Betätigen des *+ Buttons.*



Nach der Betätigung erfolgt die Öffnung des folgenden Fensters.

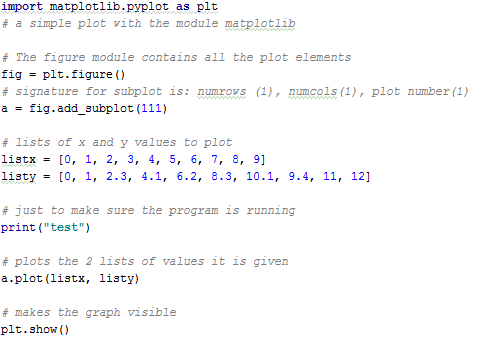
In diesem können mittels des Suchfeldes alle möglichen Module gefunden und mit betätigung des *Install Package* Buttons dann installiert werden. In unserem fall sind während der Installation keine Fehler aufgetreten.



Danach kann matplotlib in einer beliebigen Klasse mit dem Befehl *import matplotlib* verwendet werden.

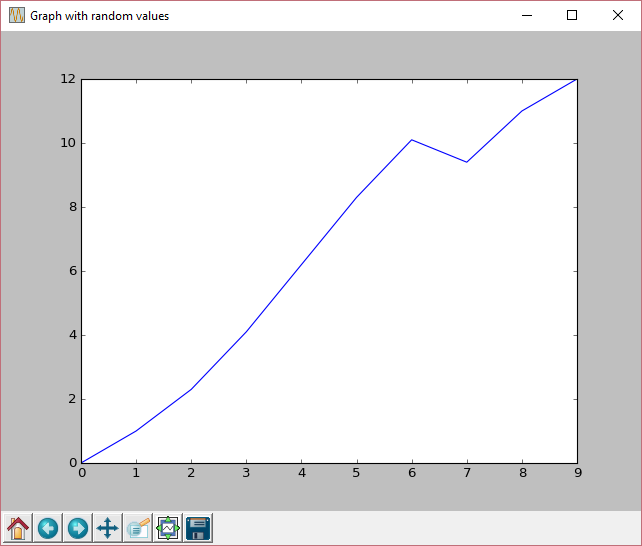
### Matlplotlib Probeklasse

In der folgenden gut dokumentierten Klasse wird gezeigt wie man ein einfaches Programm mittels matplotlib erstellt:



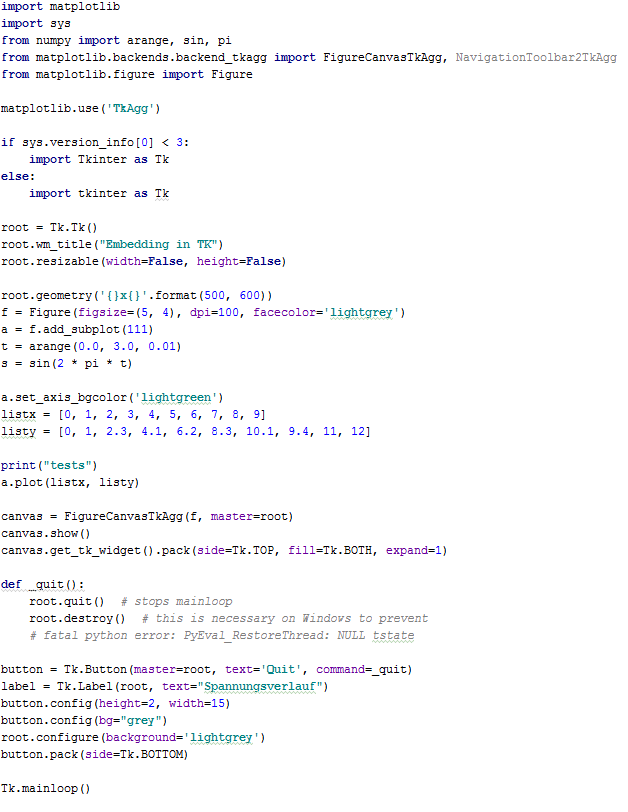
Die ausgegebene GUI sieht wie gefolgt aus:

Die Buttons im unteren Teil des Fensters sind Standardmäßig bei einem Plot dabei.

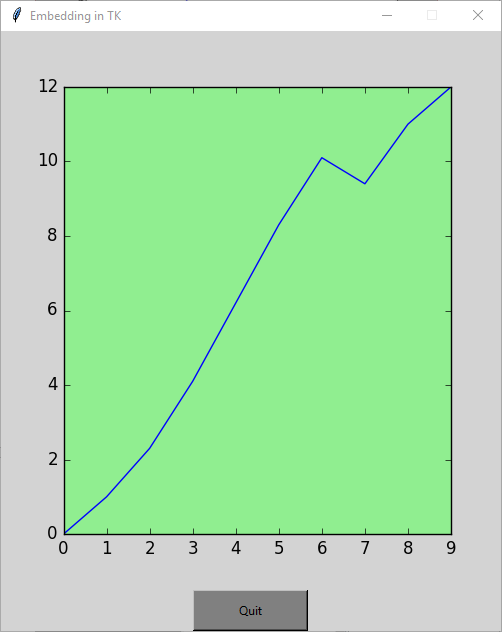


### Embedden in TKinter

Die nächste Aufgabe war die veraltet aussehen Buttons mit selbst erstellten zu ersetzen. Nach einer Recherche hat sich die Folgende Lösung ergeben:



Die Ausgabe dieses Codes sieht wie gefolgt aus:



Die Buttons im unteren Teil wurden durch einen Button des Moduls Tkinter ersetzt. Eigentlich wird gehört der ganze südliche Teil zu Tkinter.

# Graphische Darstellung von Testdaten

## PyQT Designer Konvertierung

Nach der Erstellung der GUI mit Verwendung von PyQT Designer wird ein .ui File ausgegeben. Da das .ui File nur im Designer Programm geöffnet und verändert werden kann, wird es in ein Python (.py) File umgewandelt.

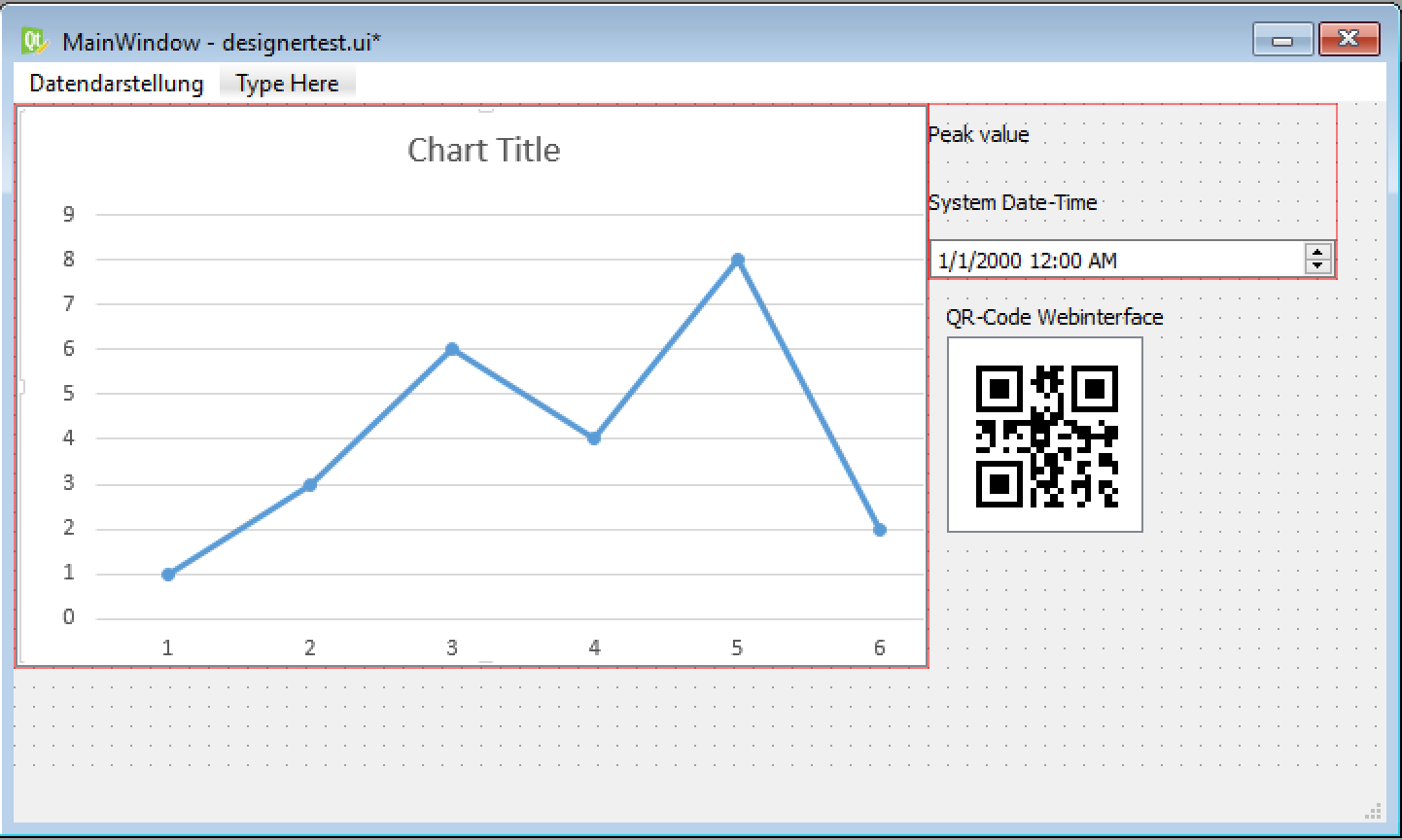
Dies erfolgt ein Programm namens pyuic und ist Teil des Python Site-Package. Nach mehreren gescheiterten Versuchen das Programm in Windows auszuführen hat sich das Projektteam für eine Virtualisierte Umgebung entschieden. Diese ist ein Ubuntu Desktop 14.04. Da auf dieser Ubuntu Version Python 3 installiert war wurde die Python Version 2.7 eingefügt. Nach Öffnung der Linux CLI wird der Befehl *apt-get install python-qt4*

eingegeben um pyuic4 zu installieren. Damit pyuic4 ein .ui File konvertiert muss der Befehl

*pyuic4 –o pyqtfile.ui pythonfile.py* eingegeben werden.

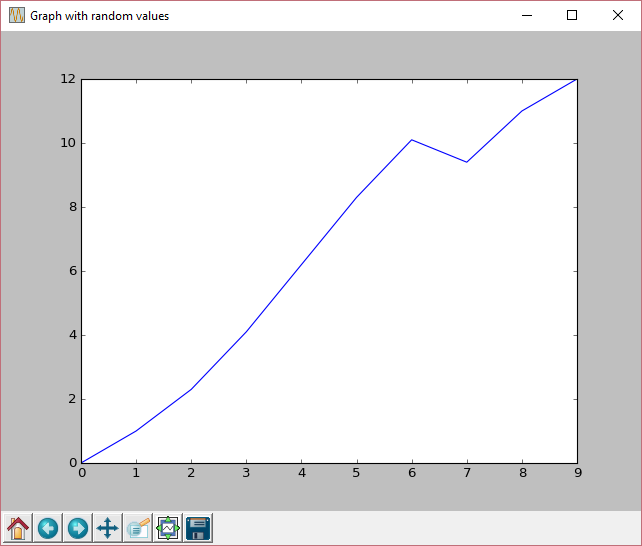
Um Implementierungsprobleme bei der Testung des von pyuic ausgegebenen Files zu beheben wurde Netbeans Pycharm auch in der VM installiert. Dies erfolgt über das Herunterladen des Files von der Offiziellen Netbeans Webseite und das darauffolgende Entpacken und Ausführen der pycharm.sh Datei.

Nach der Ausführung des konvertierten Programms wird das folgende Fenster angezeigt:



Da die eigentliche Darstellung der GUI nicht in einer VM sondern auf dem Beaglebone erfolgen soll wurde ein anderes Programm dafür entwickelt. Dies ist das vorhergenannte Programm namens matplotlib Probeklasse. Um die GUI des Beaglebones zu starten muss man sich zuerst als Default User in einen Putty Session anmelden (Wobei man X11 Port Forwarding aktivieren muss).

Nach der Anmeldung wird der Befehl *startlxde* ausgeführt um ein Lightweight Desktop im Host System geöffnet. Eine Voraussetzung des Hostsystems ist das ein XMing Server auf diesen vorhanden sein muss. Im jetzt geöffneten Beaglebone Desktop wird das Python File hineinkopiert und ausgeführt. Dadurch erscheint das Folgende Fenster:



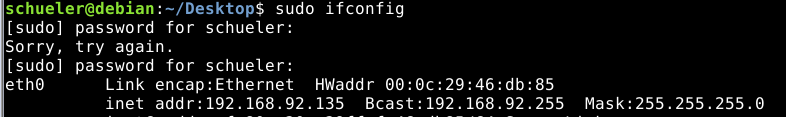
# Webserver einrichten

## Einleitung

Ziel ist es, die Verbindung von Beagle Bone zu einem Linux Betriebssystem (z.B: Debian) herzustellen. Auf dem Debian System läuft schon ein bereits konfigurierter Webserver. Das Debian System ist der Webserver.

Um eine Verbindung zum Beagle Bon herstellen zu können, wird ein Webserver benötigt. Die Daten werden von Beagle Bon zum Webserver übertragen. Der Webserver verwaltet die Webschnittstelle bzw. er zeigt die Daten auf einer Webseite (nämlich auf das Web Interface) an.

Der Webserver hat eine IP-Adresse, welche lautet: 192.168.92.135.



# Erstellung bzw. Überarbeitung eines Webinterfaces

## Einleitung

Das Web Interface ist eine grafische Benutzeroberfläche. Das Web Interface (Webschnittstelle) ist die Schnittstelle des Webservers, die dafür verwendet wird, um eine Verbindung zum Beagle Bon herzustellen. Die Verbindung wird aber simuliert. Man erstellt sozusagen eine Verbindung zur SQLite-Datenbank. Diese Datenbank heißt Beagle Bon. Beagle Bon ist die simulierte SQLite-Datenbank.

Das Web Interface kann auch die Laserkanonen vom dem Hardware-Teil steuern. Die Laserkanonen können gestartet und gestoppt werden. So können die Laserkanonen über das Web Interface simuliert und getestet werden.

## Durchführung

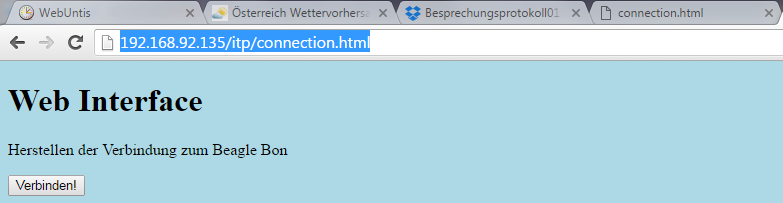
Folgende Programmiersprachen werden angewandt:

* HTML
* JavaScript
* PHP

Um den Webserver erreichen zu können, soll folgende URL im Browser eingegeben werden:

<http://192.168.92.135/itp/connection.html>

Wenn man diese URL eingibt, wird eine Webseite angezeigt. Es gibt einen Button. Wenn man auf ihn klickt, gelangt man auf dieser Webseite.



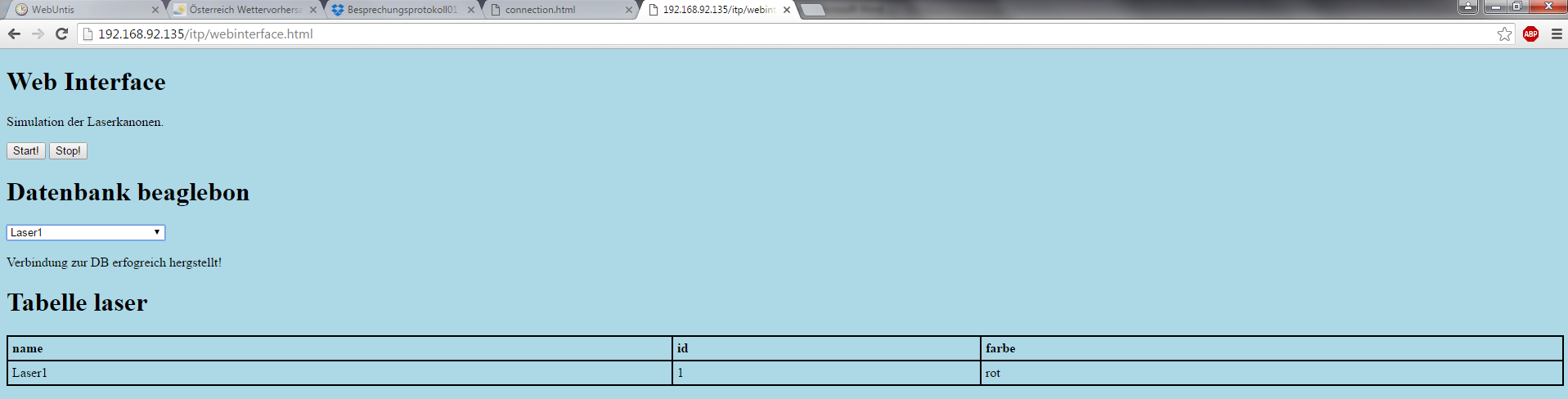
Wenn man auf den Button „Verbinden“ klickt, gelangt man auf dieser Webseite.

Die simulierte Datenbank heißt BeagleBone. In der Liste kann auswählen, welchen Laser man steuern möchte. Die Voraussetzung ist dafür, dass man auf den START-Button schon gedrückt hat. Laser1, Laser2 sind die Tabellen aus der Datenbank. Die einzelnen Arten von Lasern können gesteuert werden.

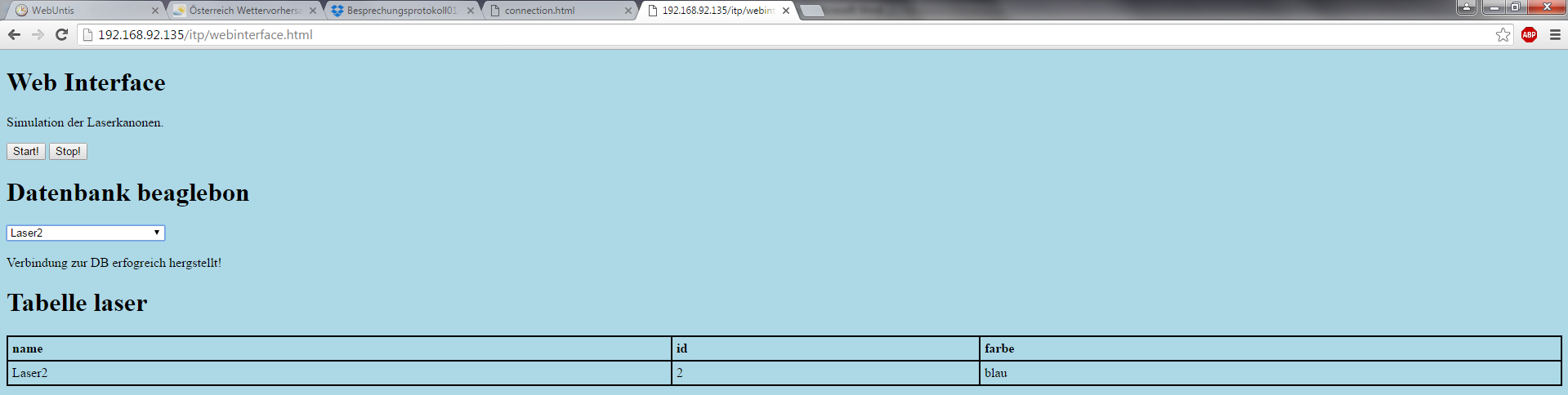


Wenn man aus der Liste, die Tabelle Laser1 auswählt, dann werden die Informationen bzw. die Daten von den einzelnen Lasern angezeigt. Ein Laser hat zum Beispiel einen Namen („Laser1“, eine Id (1) und eine Farbe („rot“). Jeder andere Laser hat eine andere Eigenschafte (z.B: einen anderen Namen oder eine andere Farbe).

Laser 1:

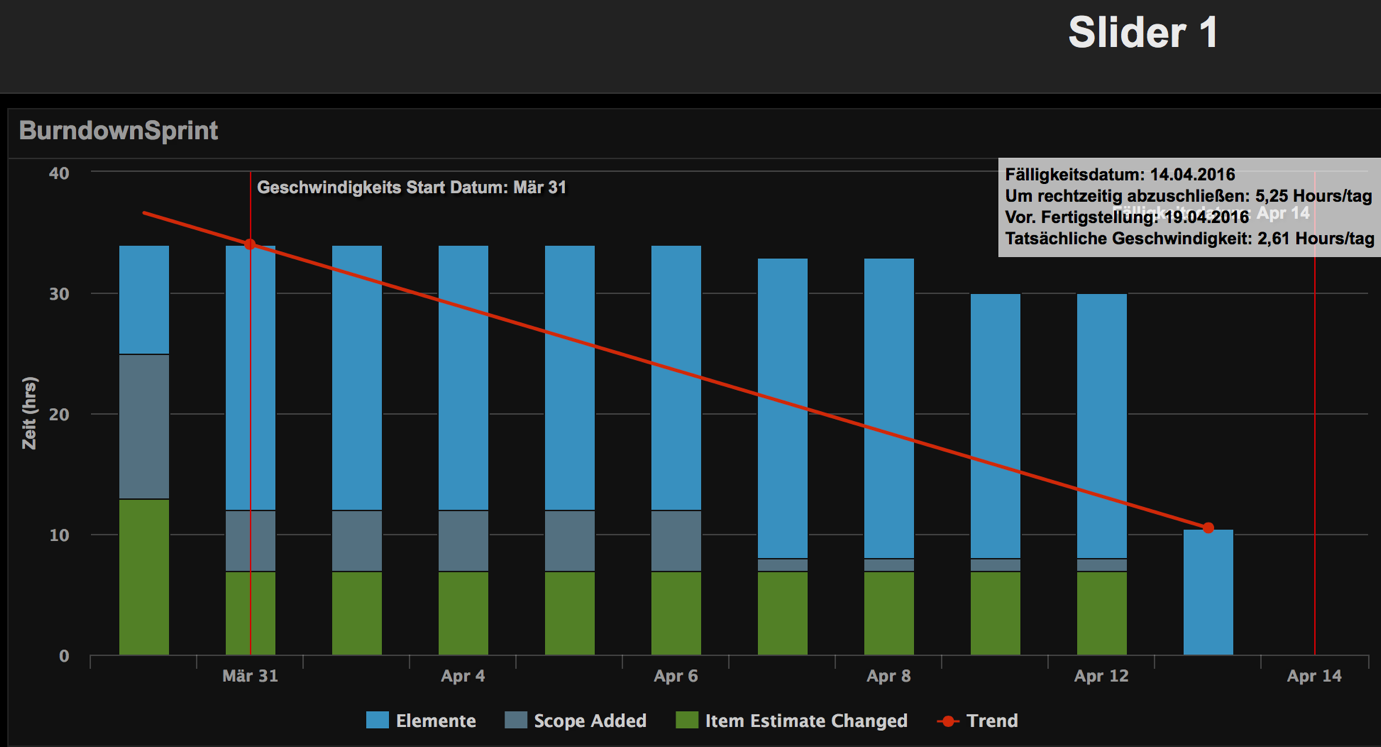


Laser2:

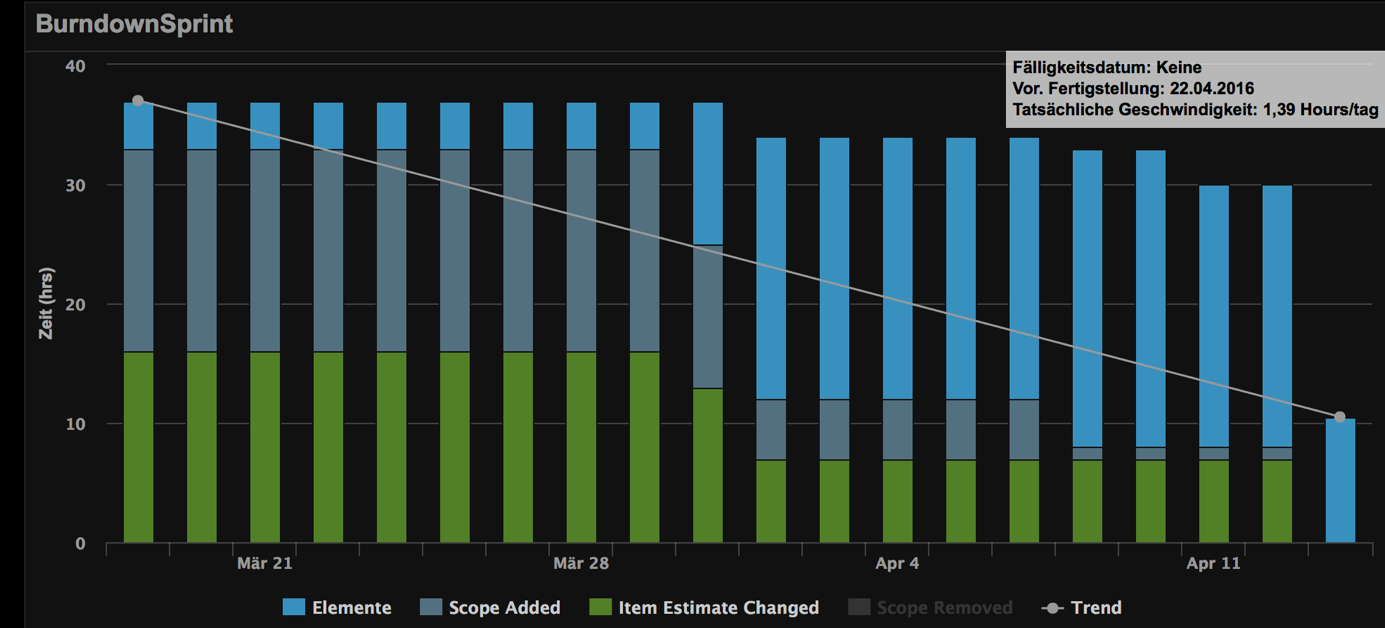


# Burndown Charts

## Sprint Burndown Chart



## Product Burndown Chart



# SQLite Datenbank

## Warum SQLite

Unser Projektteam hat sich für das DBMS SQLite wegen den Folgenden punkten entschieden:

* SQLite benötigt viel weniger Ressourcen als andere herkömmliche DBMS. Dies ist wichtig da der Beagle Bone nicht viel Rechenleistung zur Verfügung stellen kann.
* Es ist File basiert (die ganze Datenbank besteht aus 1. File) und ist damit sehr Portable.
* Es ist zum Testen von Daten geeignet da es nicht zusätzliche Prozesse für diesen Vorgang benötigt.

## Installation

In unserem Fall wird die SQLite Version 3 verwendet. Um den Download der .tar.gz Datei zu initialisieren wird der Befehl ***wget http://www.sqlite.org/sqlite-autoconf-3070603.tar.gz*** verwendet. Darauffolgend wird die jetzt heruntergeladene Datei mit dem tar Befehl

***tar xvfz sqlite-autoconf-3070603.tar.gz*** die Datei entpackt. Danach wird in eines der extrahierten Verzeichnisse mit dem Befehl ***cd sqlite-autoconf-3070603*** gewechselt. Hier befindet sich dann die Datei die mit dem ***./configure*** Befehl in ein make fähiges gemacht wird. Mit den Befehlen ***make*** und darauffolgend ***make install*** wird dann SQLite3 installiert.

Nachdem die ***make*** Befehle ausgeführt werden erschient ein Output der besagt dass die SQLite3 Binaries im Verzeichnis ***/usr/local/bin*** installiert werden.

## Erstellen einer SQLite3 Datenbank

Für das Projekt wird eine Sehr simple Datenbank erstellt die die Peak Werte der Differenz der Beiden Laser speichert. Die Datenbank wird mit den Befehl ***sqlite3 beaglebone*** erstellt.

Es wird dann ein Table mit dem namen ***PEAK*** mit dem Datentypen ***INTEGER*** mit dem Befehl ***create table PEAK(value INTEGER)*** erstellt.

Die Tabelle wird dann den folgenden Testdaten gefüllt: ***Insert into PEAK value(5)***