Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Fachbereich: Ingenieurwissenschaften – Energie und Information
Studiengang: Computer Engineering

Bachelorarbeit

Entwicklung eines vollautomatisierten Embedded-Linux-Systems zur Ansteuerung und Auswertung eines Langzeittests

zur Erlangung des akademischen Grades

Bachelor of Engineering

von

Tim Nieter

05. Januar 2015 – 16. März 2015

- Nicht öffentlich -

Erstbetreuer: Prof. Dr. F. Bauernöppel
Zweitbetreuer: G. Schoel
Eingereicht am: 16. März 2015

Eidesstattliche Erklärung

Tim Nieter
Rudower Straße 95
12351 Berlin

Hiermit versichere ich, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Berlin, den 16. März 2015

Tim Nieter

(Unterschrift)

Sperrvermerk

Die nachfolgende Bachelorarbeit mit dem Titel „Entwicklung eines vollautomatisierten Embedded-Linux-Systems zur Ansteuerung und Auswertung eines Langzeittests“ enthält vertrauliche Daten der Pepperl+Fuchs GmbH. Veröffentlichungen oder Vervielfältigungen der Arbeit – auch nur auszugsweise – sind ohne ausdrückliche Genehmigung der Pepperl+Fuchs GmbH nicht gestattet. Die Arbeit ist lediglich den Korrektoren sowie den Mitgliedern der Prüfungskommission zugänglich zu machen.

Vorwort

Die Netze sind in einer vernetzten Welt von netzartiger Bedeutung. Ohne Netze ist

Die vorliegende Arbeit ist auf Basis des Latex-Templates zu [1] erstellt worden.

[1] T. Gockel. Form der wissenschaftlichen Ausarbeitung. Springer-Verlag, Heidelberg, 2008. Begleitende Materialien unter http://www.formbuch.de.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

SQL Structured Query Language . . . 3
ERM Entity-Relationship-Modell . . . 9
DUT Device Under Test . . . 5
GUI Graphical User Interface . . . 2
IDE Integrated Developement Environment . . . 2
SDK Software Developement Kit . . . 2

1 Einleitung

Wann immer Systeme in der realen Welt eingesetzt werden, sollen sie so zuverlässig, fehlerfrei und vorhersehbar wie möglich arbeiten. Gerade wenn diese Systeme an kritischen Punkten zum Einsatz kommen und über lange Zeiträume agieren, sind diese Eigenschaften besonders wichtig.
Um diesen Anforderungen gerecht zu werden, müssen alle Bauelemente eines solchen Systems diese Vorgaben erfüllen, denn die Zuverlässigkeit ist immer abhängig vom schwächsten Glied. Bei der Entwicklung eines Systems ist es somit entscheidend, alle Bauelemente vorher anhand der gegebenen Umstände zu qualifizieren. Dafür müssen die Grenzen ausgelotet werden, in denen sie zuverlässig betrieben werden können.

Eine dieser Grenzen ist die altersbedingte Änderung der Betriebsparameter, die sogenannte Degradation. Um das Degradationsverhalten eines Bauelementes bestimmen zu können, muss es über lange Zeiträume unter erschwerten Bedingungen betrieben und ausgewertet. Nur wenn ein Bauelement ein für die Anwendung akzeptables Degradationsverhalten aufweist, ist es für den Einsatz im Gesamtsystem geeignet.
Da ein hoher Einsatz von Ressourcen nötig ist, um jedes Bauelement individuell zu Prüfen, existieren automatisierte Teststände mit deren Hilfe der Aufwand minimiert werden soll.

Ein solcher automatisierter Teststand soll im Rahmen dieser Arbeit Entwickelt werden. Dies beinhaltet die Datenerfassung, die Datenauswertung und die Überwachung des Systems.

2 Grundlagen

Bei der Entwicklung des Teststandes kommen verschiedene Systeme, Schnittstellen und Konzepte zum Einsatz. Die Grundlagen dazu werden im folgenden Kapitel kurz erläutert.

2.1 Degradation

what do

2.2 QT

QT ist eine umfangreiche C++-Klassenbibliothek zur Gestaltung und Entwicklung von Anwendungen.Vor allem bei Applikationen mit grafischen Benutzeroberflächen (englisch: Graphical User Interface (GUI)) ist QT sehr beliebt.
Zusätzlich bringt QT eine große Auswahl an Tools und Modulen mit sich, welche die Programmierung erheblich erleichtern (z.B. Netzwerkprogrammierung, Datenbankanbindung, OpenGL, etc.).
Ein weiterer Vorteil ist die Plattformunabhängigkeit. So unterstützt QT aktuell (Version 5.4, 10. Dezember 2014) einen Großteil der aktuellen Betriebssystem wie Windows, Linux, Android, iOS und einige mehr.

Als Entwicklungsumgebung (englisch: Integrated Developement Environment (IDE)) dient der QT Creator (siehe Abbildung 2.2.1), welcher Teil des Software Developement Kit (SDK) von QT ist und sowohl auf Linux, Windows und auch Mac OS X zur Verfügung steht. Er kommt mit einem Debugger, einem integrierten grafischen GUI Designer und einem Texteditor mit Funktionen wie Syntax-Hervorhebung und automatischer Vervollständigung.
Es können alle gängigen Compiler verwendet werden und es besteht die Möglichkeit eigene Toolchains anzulegen.

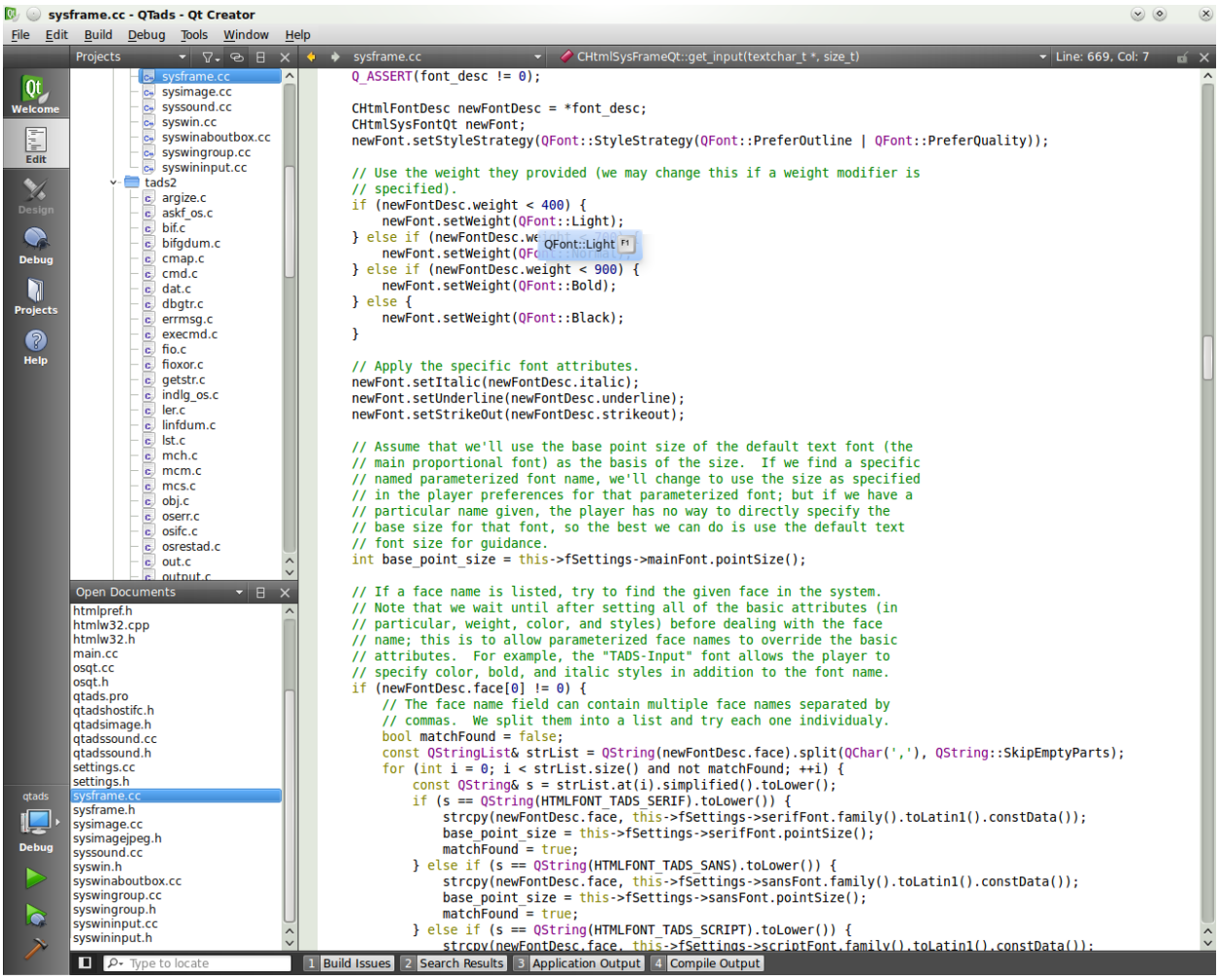

Abb. 2.2.1: QtCreator Version 2.0.1

2.3 RS232

Bei RS232 handelt es sich um eine serielle Schnittstelle zu Datenübertragung.

2.4 Datenbank

Für das Speichern der Messdaten und Betriebsparameter wird eine Structured Query Language (SQL) Datenbank verwendet. Bei der Wahl des Datenbankverwaltungssystems, standen mehrere Optionen zur Auswahl und die Entscheidung wurde zwischen SQLite und MySQL gefällt. Beide System haben ihre Vor- und Nachteile.

SQLite ist ein SQL Datenbankverwaltungssystem, welches ohne einen Server auskommt und operiert stattdessen in einer einzigen Datei. Es wird vor allem im Embedded Bereich eingesetzt, da kaum Konfigurationen oder Verwaltung notwendig ist. Deshalb eignet es sich ausgezeichnet für sich schnell weiterentwickelnde Applikationen.
Aufgrund dieser Eigenschaften existieren allerdings auch Nachteile. So unterstützt SQLite nur ein-

geschränkt mehrere Nutzer gleichzeitig. Da das gesamte Datenbanksystem in einer einzigen Datei zusammengefasst ist, können mehrere zur selben Zeit durchgeführte Schreibzugriffe nicht unterstützt werden. Denn die einzige Sicherstellung der Datenintegrität erfolgt durch das Betriebssystem.
Des Weiteren ist SQLite aufgrund des Ein-Datei-Systems nur eingeschränkt skalierbar. Bei einer größeren Datenmenge oder erhöhten Anzahl an Zugriffen ist es nicht möglich diese Datei auf mehrere Systeme zu separieren, um somit die Last gleichmäßig zu verteilen.

MySQL ist ein weiteres SQL Datenbankverwaltungssystem, welches allerdings auf einer Serverarchitektur beruht. Es ermöglicht die Verwaltung von Nutzern und Rechte. Außerdem ist das System gut hinsichtlich Performance und Größe zu skalieren. Zusätzlich bietet MySQL viele Möglichkeiten für Performanceanpassungen, wie z.B. Query-Caching.
Jedoch gibt es auch hier Nachteile. So ist die Konfiguration wesentlich schwerer und komplexer. Durch die Notwendigkeit eines Servers benötigt MySQL mehr Ressourcen auf dem Host-System.

Die Wahl fällt auf das relationale Datenbankverwaltungssystem MySQL. Auch wenn SQLite einige Vorteile vor allem im Embedded Bereich besitzt, ist die fehlende Unterstützen von mehreren Nutzern gleichzeitig ein Ausschlusskriterium.

[SSH10]

3 Anforderungsanalyse

Gegeben ist ein existierender Degradationsteststand. Die Bedienung des bestehenden Systems bringt jedoch ein hohes Maß an individuellen Anpassungen, einen großen zeitlichen Aufwand, geringe Benutzerfreundlichkeit und Transparenz mit sich. Aus diesen Gründen soll es hingegen der Benutzerfreundlichkeit, der Effizienz und der räumlichen Ausdehnung optimiert werden. Dazu ist es nötig das vorhandene System von Grund auf zu überarbeiten.

Folgende Anforderungen werden dabei an das neue System gestellt:

- Individuelle Parametrierung der Devices Under Test (DUTs)
- Automatische Erfassung der Messdaten
- Auswertung der Messdaten via Fernzugriff

Individuelle Parametrierung der DUTs
Immer 64 DUTs vom selben Typ befinden sich auf einem Testboard.

Autmatische Erfassung der Messdaten
Die Messdaten der DUTs sollen zyklisch erfasst werden. Dabei sollen die Intervalle zwischen den Messungen konfigurierbar sein.

4 Konzept

Im folgenden Kapitel wird auf die Erstellung des Konzeptes eingegangen.

4.1 Teststand

Das Gesamtsystem setzt sich aus drei Untersystemen zusammen:

- Mess-Slave
  - Übernimmt die lokale Ansteuerung der DUTs
  - Nimmt Messdaten auf und stellt sie zur Verfügung
- Mess-Master
  - Verwaltet angeschlossene Mess-Slaves
  - Speichert alle Messdaten
  - Bildet das Bindeglied zwischen Mess-Slave und dem PC
- PC Verwaltung
  - Parametriert die Mess-Slaves
  - Wertet Messdaten aus und stellt sie leserlich da

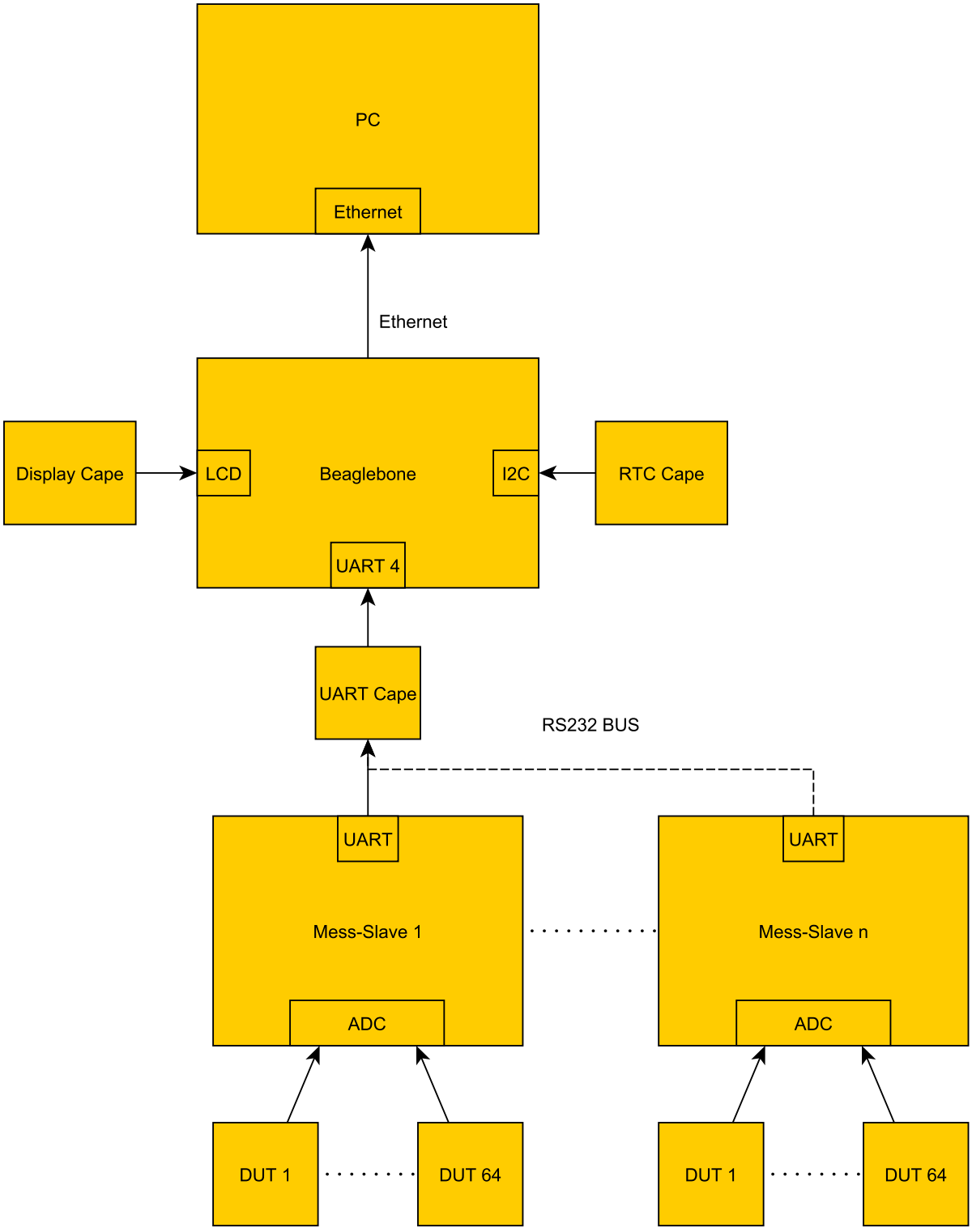

Abb. 4.1.1: Gesamt System

4.1.1 Mess-Slave

Das Herzstück des Mess-Slave bildet ein STM8 8-Bit Mikrocontroller der Firma STMicroelectronics. Auf jedem Mess-Slave sind 64 DUTs befestigt. In zyklischen Abständen werden die Messdaten der Prüfobjekte aufgenommen und über eine RS232-Schnittstelle zur Verfügung gestellt (siehe Abbildung **??**.
Dieses System war zum großen Teil bereits gegeben, so dass lediglich die Übertragung der RS232 Schnittstelle geregelt werden musste. Dazu wurde ein Protokoll für die Kommunikation entworfen (sieht Abschnitt 4.1.4).

4.1.2 Mess-Master

Als Mess-Master wird ein BeagleBone Black von Texas Instruments eingesetzt. Dabei handelt es sich um einen Einplatinen-Computer, der mit einem AM335x 1GHz ARM® Cortex-A8 Prozessor arbeitet. Das Betriebssystem ist eine Embedded Linux Debian Lösung.

4.1.3 PC Verwaltung

asdasad

4.1.4 RS232 Protokoll

Das Protokoll für die Kommunikation über die RS232 Schnittstelle ist nötig, um die Validität, Vollständigkeit und Zuverlässigkeit der Übertragungen sicherzustellen.

Folgende Kriterien sollen dabei gegeben sein:

- Adressierung individueller Kommunikationspartner
- Senden verschiedener Befehle
- Variable Größe der Daten
- Sicherstellung der Validität des Rahmens

Daraus ergibt sich folgendes Protokoll.

4.2 Datenbank

Aus den Anforderungen ergibt sich folgendes Entity-Relationship-Modell (ERM) (siehe Abbildung 4.2.1).

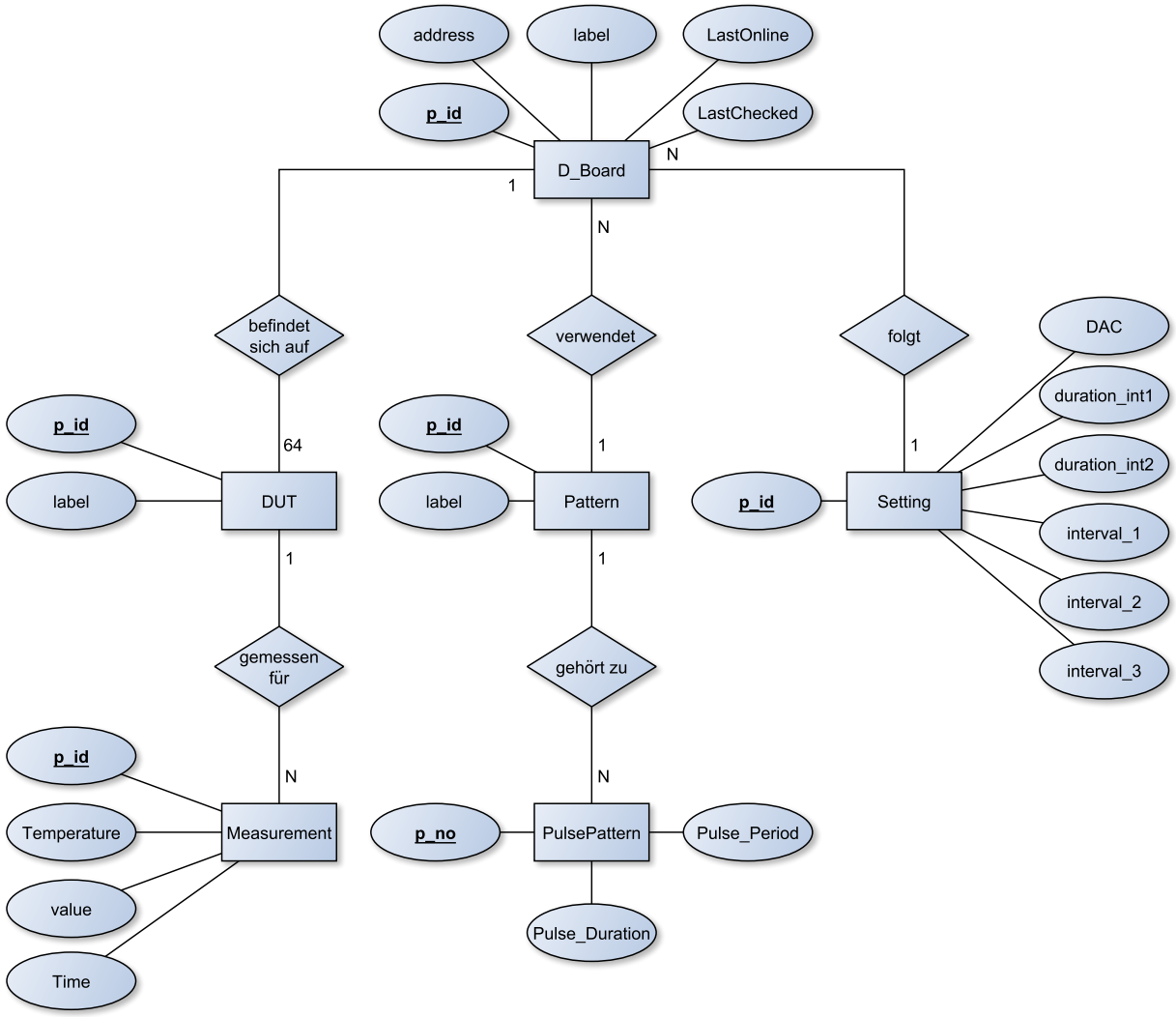

Abb. 4.2.1: Entity Relationship Modell

Die Datenbank muss folgende Daten für die Parameter der Mess-Slaves aufnehmen:

Parameter	Beschreibung
DAC	Vorverstärkung
duration_int1	Dauer der Zeit die Werte im 1.Interval aufgenommen werden in Tagen
duration_int2	Dauer der Zeit die Werte im 2.Interval aufgenommen werden in Tagen
interval_1	Abstand zwischen den Messungen im 1. Interval in Minuten
interval_2	Abstand zwischen den Messungen im 2. Interval in Minuten
interval_3	Abstand zwischen den Messungen nach dem 2. Interval in Minuten

Tab. 4.2.1: Tabelle Setting

5 Testen und Validieren

TEST TEST 123

Literaturverzeichnis

[SSH10] Saake, G. ; Sattler, K.U. ; Heuer, A.: *Datenbanken: Konzepte und Sprachen.* mitp, 2010 (Biber-Buch). – ISBN 9783826690570