Hochschule Magdeburg-Stendal
Fachbereich Ingenieurwissenschaften und Industriedesign (IWID)
Institut für Elektrotechnik

Masterarbeit

zur Erlangung des Grades eines "Master of Engineering"
im Studiengang Elektrotechnik-Gebäudesystemtechnik

Thema: **Entwicklung und Realisierung eines Prototyps zur Erfassung einzelner Produktionsabläufe und Protokollierung dieser in einer Datenbank**

Eingereicht von: **Eric Schmieder**

Angefertigt für: Krüger & Gothe GmbH

Matrikel: 20112139

Ausgabetermin: 23.01.2018

Abgabetermin: 19.06.2018

Schulischer Betreuer: Herr Prof. Dr.-Ing. Dieter Schwarzenau

Betrieblicher Betreuer: Herr Dipl.-Ing. Klaus Junge

....
1. Prüfer 2. Prüfer

I Kurzfassung

Bei der Produktion von elektronischen Geräten werden je nach Komplexität der Baugruppe verschiedene Arbeitsschritte vollzogen. Hierbei ist eine Rückverfolgbarkeit der einzelnen Arbeitsschritte Voraussetzung, um eine höchstmögliche Transparenz der Produktion zu erhalten. Diese ist einerseits für eine preisliche Bewertung und eine maximale Effizienz der Arbeitsschritte unabdingbar, andererseits muss beispielsweise bei möglichen qualitativen Mängeln jederzeit eine Aussage darüber getroffen werden können, welche Produktionsserie von defekten Bauteilen betroffen ist. Um die einzelnen Arbeitsschritte zu protokollieren, wurde ein Einplatinencomputer für verschiedenste Arbeitsplätze entwickelt. Gegenstand dieser Arbeit ist die Erstellung des Betriebssystems auf LINUX-Basis für den Einplatinencomputer, die Konstruktion einer MySQL-Datenbankstruktur und die Programmierung der Anwendungssoftware in Python. Nachfolgend werden die notwendigen Grundlagen zu dieser Thematik erläutert, die verwendete Hardware mit dazugehörigem Aufbau dargestellt und die einzelnen Softwareaspekte erklärt. Abschließend werden die Ergebnisse der Testmessungen von zwei verschiedenen Arbeitsplätzen veranschaulicht, analysiert und verschiedene Erweiterungsmöglichkeiten dargelegt.

II Abstract

In the production of electronic devices, various steps are performed depending on the complexity of the assembly. Traceability of the individual work steps is indispensable in order to obtain the highest possible transparency of production. On the one hand, this is necessary for a price evaluation and maximum efficiency of the work steps. On the other hand, for example, in the case of possible qualitative defects, it must be possible to make a statement about which production series is affected by defective components at any time. To log the individual work steps, a single-board computer was developed for a wide variety of workstations. The subject of this thesis is the creation of the LINUX-based operating system for the single-board computer, the construction of a MySQL database structure and the programming of the application software in Python. In the following, the necessary basics of this topic will be clear up, the used hardware and the corresponding structure will be shown and the individual software aspects will be explained. Finally, the results of the test measurements from two different workplaces are illustrated, analyzed and various expansion options presented.

III Abkürzungsverzeichnis

Abkürzung	Beschreibung
AD	Altium Designer
AOI	automated optical inspection
ARM	Advanced RISC Machine
ASCII	American Standard Code for Information Interchange
CISC	Complex Instruction Set Computing
CPU	central processing unit
DHCP	Dynamic Host Configuration Protocol
DMC	Data-Matrix-Codes
DNS	Domain Name System
EMS	Electronics Manufacturing Services
EVM	Evaluation module
GPL	General public License
GPU	Graphics Processing Unit
GS1	Global Standards One
HDD	Hard Disk Drive
IoT	Internet of Things
LAN	Local Area Network
MOSFET	Metall-Oxid-Halbleiter-Feldeffekttransistor
NTP	Network Time Protocol
OS	Operating System
PAP	Programmablaufplan
POST	Power On Self Test
RAM	Random-Access Memory
RFID	radio-frequency identification
RISC	Reduced Instruction Set Computing
SDK	Software Development Kits
SMD	Surface-mounted device
SoC	System on Chip
SPL	Secondary Program Loader
SQL	Structured Query Language
THT	Through Hole Technology
TI	Texas Instruments
UART	Universal Asynchronous Receiver Transmitter
UHF	Ultra-High-Frequency

Tabelle 1: Abkürzungstabelle

Inhaltsverzeichnis

1 Einleitung

Bei der modernen Produktion von elektronischen Geräten ist die Protokollierung der einzelnen Arbeitsschritte und damit deren Rückverfolgbarkeit eine wichtige Voraussetzung für die Sicherstellung der Qualität. Hierbei ist allerdings nicht nur die Tätigkeit und deren Dauer von Relevanz, sondern auch welche Bauteile in welchem Arbeitsschritt verwendet wurden.

Derzeit werden an verschiedenen Arbeitsplätzen in der Krüger & Gothe GmbH unterschiedliche Arbeitsschritte ausgeführt, z.B. Reparatur, Sichtkontrolle, Bestückung. Allerdings werden die einzelnen Produktionsabläufe bei einigen Prozessen entweder gar nicht, oder nicht ausreichend protokolliert. Erfolgt eine Protokollierung, wird diese in der Regel durch einen Personalcomputer, bestehend aus Rechner, Monitor, Tastatur, Maus realisiert. Durch diese Handhabung ergeben sich allerdings drei zentrale Probleme:

- ein großer Platzbedarf
- hoher Kostenaufwand für den Personalcomputer.
- Umständliche Auswertung der nur lokal gespeicherten Daten

Deshalb soll eine angepasste Lösung mit einem Einplatinencomputer entwickelt werden. Die Hardware-Entwicklung ist bereits abgeschlossen. Im nächsten Schritt ist ein Betriebssystem anzupassen und eine Anwendung zu programmieren, die mit dem Benutzer interagiert, sowie einzelne Arbeitsschritte und Fehler in einer Datenbank protokolliert.

Da der Einplatinencomputer auch für andere Anwendungsgebiete, beispielsweise verschiedene Messungen oder Testautomatisierungen, in der Krüger & Gothe GmbH eingesetzt werden soll, müssen das Betriebssystem universell verwendbar und das Anwendungsprogramm modular aufgebaut werden.

2 Betreuendes Unternehmen

Die Master-Arbeit wurde in Zusammenarbeit mit der Entwicklungsabteilung der Krüger & Gothe GmbH für die Zwecke des Unternehmens realisiert.

Abbildung 1: Hauptsitz der Krüger & Gothe GmbH

Bei der betreuenden Firma[1], die 1997 gegründet wurde, handelt es sich um einen Elektronikdienstleister, im engl. auch als Electronics Manufacturing Services (EMS) bezeichnet, mit Hauptsitz in Staßfurt und ca. 180 Mitarbeitern. Unter anderem werden folgende Dienstleistungen angeboten:

- Automatische Bestückung von SMD- (Surface-mounted device) und THT (Through Hole Technology) -Bauteilen für kleine, mittlere und große Stückzahlen,
- Handbestückung (Einzelplätze und Montagebänder),
- Prüfung mit AOI- (automated optical inspection) Systemen,
- In-Circuit-Test,
- Boundary-Scan,
- Funktionsprüfung,
- Gerätebau und Gerätemontage,

wobei die Kunden sich hauptsächlich auf die Marktsegmente Bahn-, Automobil- und Medizinindustrie sowie Luft- und Raumfahrttechnik verteilen.

[1] www.kug-ems.de/ueber-uns/ [Stand: 20.01.2018]

3 Grundlagen

In Kapitel 4 wird ein Überblick und eine kurze Erläuterung über den verwendeten Prototypen und seine Schnittstellen gegeben. In Kapitel 6 wiederum, wird auf die Vorgehensweise zur Generierung und Konfigurierung des Betriebssystems eingegangen. Anschließend werden in den beiden Punkten „Erstellung der MySQL-Datenbank“ und „Anwendungssoftware“ die Datenbankstruktur sowie das Benutzerprogramm erläutert. In den letzten beiden Kapiteln werden die beiden verwendeten Testarbeitsplätze beschrieben, die Ergebnisse der implementierten Rückverfolgbarkeitssoftware dargestellt und ein Fazit zum Projekt gegeben. Um das Zusammenspiel und die Eigenschaften verschiedener Bauteile und Algorithmen nachzuvollziehen zu können, wird in den folgenden Unterkapiteln die Rückverfolgbarkeit thematisiert sowie auf die Grundlagen der relevanten Hard- und Software eingegangen. Um den Umfang in Grenzen zu halten, wird für weiterführende Informationen auf die verwendeten Quellen verwiesen.

3.1 Rückverfolgbarkeit in Prozessabläufen

In dieser Arbeit liegt das Hauptaugenmerk bezüglich der Softwareentwicklung in der Bereitstellung eines Systems zur Rückverfolgbarkeit. Dazu werden in den folgenden Abschnitten die theoretischen Aspekte der Rückverfolgbarkeit erläutert.

3.1.1 Definition

Der Begriff Rückverfolgbarkeit, im engl. auch als Traceability bezeichnet, ist in der Norm „DIN EN ISO 9001“[2] definiert. Hier sieht die Norm vor, dass eine Organisation geeignete Maßnahmen ergreift, die für die Sicherstellung der Konformität von Produkten und Dienstleistungen notwendig sind. Das bedeutet, dass vom Wareneingang bis hin zum fertigen Endprodukt Auskunft darüber erteilt werden kann, in welcher Form das Produkt verarbeitet wurde und welche Prozesse dabei Verwendung fanden. Dies ermöglicht eine Transparenz über die gesamte Wertschöpfungskette.

[2] ISO 9001:2015 (Qualitätsmanagementsysteme – Anforderungen)

Des Weiteren wird die Rückverfolgbarkeit in zwei Bereiche, die vor- und rückwärtsgerichtete Verfolgung (im englischen auch als Downstream- und Upstream-Tracing bezeichnet werden), eingeteilt[3].

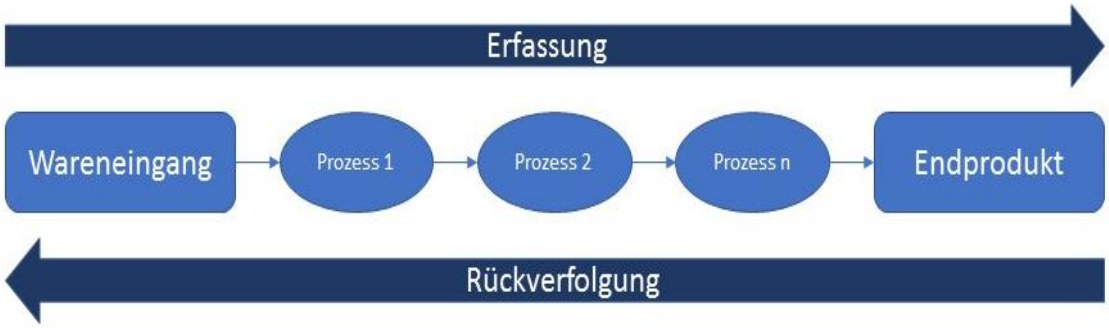

Abbildung 2: Vor- und Rückwärtsgerichtete Verfolgung

Wie in der Abbildung 2 dargestellt, erfolgt bei der vorwärtsgerichteten Verfolgung eine Erfassung beginnend beim Wareneingang, z.B. einer Leiterplatte, und erstreckt sich über die vollständige Prozesskette bis hin zum fertigen Endprodukt. Während des Herstellungszyklus werden alle relevanten Daten erfasst. Dies können unter anderem der prozessbedingte Arbeitsplatz, der ausführende Mitarbeiter oder verwendete Komponenten und Materialien sein.

Im Vergleich zum Upstream-Tracing, erfolgt die Verfolgung beim Downstream-Tracing vom Endprodukt zum Beginn der Prozesskette. Hierbei wird die Historie einer Baugruppe ermittelt und alle geleisteten Schritte können nachvollzogen werden.

[3] https://de.wikipedia.org/wiki/Rückverfolgbarkeit_(Produktionswirtschaft) [Stand 08.06.2018]

3.1.2 Identifikationstechniken und Detailtiefe

Um die verschiedensten Daten zur Rückverfolgbarkeit zu erfassen, ist die verwendete Technik hierbei ein zentraler Faktor. Eine zufriedenstellende Traceability ist nur dann zu gewährleisten, wenn jede betroffene Komponente durch eine eindeutige Identifikationsnummer gekennzeichnet wird. Um diese zu erfassen, können verschiedene Techniken eingesetzt werden. In Der Abbildung 3[4] sind die gängigsten Formen zur Markierung von Komponenten, Zubehör und Materialien dargestellt, die durch die GS1 (Global Standards One) standardisiert wurden.

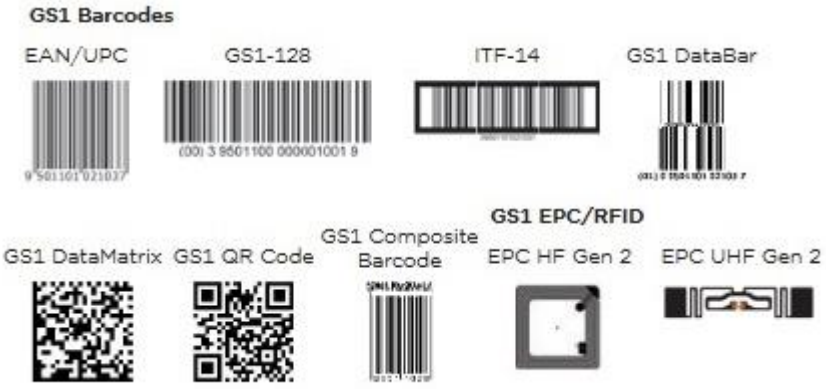

Abbildung 3: Kennzeichnungsvarianten

Hierbei ist eine Kategorisierung in die Sparten „optisch“ und „elektromagnetisch“ möglich, wobei sich diese nach der technischen Umsetzung zur Erfassung richtet. Weiterhin können die optischen Markierungen noch in Barcodes, gestapelte Codes und Matrixcodes unterschieden werden, welche bezüglich der Speicherkapazität von Informationen und in der Zuverlässigkeit im Ausleseprozess variieren[5].

[4] https://www.ecr.digital/wp_contents/uploads/2017/07/Rueckverfolgbarkeit_in_der_Lieferkette.pdf (Seite 14) [Stand: 12.06.2018]
[5] http://www.etikettenwissen.de/wiki/Barcode-Etikett [Stand: 12.06.2018]

Einhergehend mit den beiden Kategorien ergeben sich verschiedene Vor-[6] und Nachteile[7], wovon einige in der nachfolgenden Tabelle gegenübergestellt sind:

	Optische Erfassung	Elektromagnetische Erfassung
Datenübermittlung	Sichtkontakt notwendig	Ohne direkten Sichtkontakt
Fehlerquellen	Verschmutzung	Externe Störquellen
Datenänderung möglich?	Nein	Ja
Gleichzeitige Mehrfacherfassung	Nein	Ja
Kopierschutz möglich?	Nein	Ja
Maximaler Leseabstand	Ca. 0,5m	Bis zu 10m (UHF)
Preis	< 0,01€	>0,10€

Tabelle 2: Unterscheidung der Erfassungsarten

Neben der Auswahl der geeigneten Identifikationstechnik ist die Tiefe der Erfassung ebenso von höchster Relevanz. Hier muss eine Entscheidung darüber getroffen werden, in welchem Detail die Daten erfasst werden. Beispielsweise ist es in einem Prozess möglich, jedes verwendete Material einzeln oder nur die zugehörige Chargennummer zu erfassen. Mit einer steigenden Präzision im Erfassungsdetail, erhöht sich gleichermaßen die Aussagekraft der erfassten Datensätze. Durch das Markieren und Erfassen jeder einzelnen Komponente entsteht allerdings auch ein zusätzlicher Arbeitsaufwand.

[6] https://automation-insights.blog/2018/04/04/dmc-vs-rfid-in-manufacturing/
[7] Handbuch Fügen, Handhaben und Montieren, Seite 852

3.1.3 Vorteile von Traceability

Abhängig von der Tiefe der Rückverfolgbarkeit ergeben sich einerseits verschiedene Vorteile für das Upstream-Tracing. Hierbei kann beispielsweise eine Optimierung während der Produktionsprozesse erfolgen, um eine bestmögliche Auslastung der Kapazitäten zu erzielen. Andererseits lässt sich durch das Downstream-Tracing die Historie und Herkunft eines Produktes bestimmen, wodurch bei auftretenden Mängeln im Endprodukt weitere mögliche betroffene Geräte identifiziert und eine Ursache bestimmt werden kann.

Unabhängig von den genannten Punkten ergeben sich auch Vorteile bezüglich der Kategorie[8] der erfassten Daten, die sich wie folgt gliedern:

Mitarbeiter: Anhand einer eindeutigen Erfassung eines jeden Mitarbeiters in den einzelnen Arbeitsschritten, lässt sich durch eine Kalkulation der durchschnittlichen Bearbeitungsdauer eine genauere Kalkulation der Kosten jedes Prozesses.

Prüfungen: Bei der Prüfung von Baugruppen ist es beispielsweise möglich, neben dem erfassten Zeitpunkt und des Mitarbeiters, auch die Mess- und Prüfungsergebnisse aufzuzeichnen. Dadurch ist eine Sperrung der fehlerhaften Artikel über eine Datenbank möglich, damit diese nicht unbeabsichtigt im Warenausgang verarbeitet werden.

Prozesse: Bei der Verarbeitung in den einzelnen Prozessen ist eine Erfassung der verwendeten Konfigurationen möglich. Dies können bei einem EMS u.a. die Löttemperaturkurve, die verwendete Maschine oder das verwendete Wellenlötprogramm sein.

Material: Bei der Aufzeichnung der Materialien ist eine Erfassung jedes Lieferloses möglich. Hier ist eine detaillierte Kennzeichnung und Erfassung von Lieferung, Lieferant und/ oder Hersteller denkbar.

[8] https://www.kuttig.de/ems-dienstleistungen/service/rueckverfolgbarkeit.html [Stand: 12.06.2018]

3.2 Bauteile der Elektrotechnik

Nachfolgend werden einige elektronische Bauteile kurz erläutert, die in dem genutzten Hardwareprototypen verwendet werden. Eine ausführliche Beschreibung würde an dieser Stelle zu weit führen, deshalb wird hier auf die angegebenen Quellen verwiesen.

3.2.1 ARM-Prozessor

Im Allgemeinen unterscheidet man Prozessoren anhand ihrer Architektur, also zwischen CISC (Complex Instruction Set Computing)- und RISC (Reduced Instruction Set Computing)-Architekturen[9]. In gebräuchlichen Personalcomputern werden im Normalfall CISC-Prozessoren eingesetzt, da diese einen umfangreichen Befehlsumfang besitzen und sehr komplexe Adressierungsmöglichkeiten haben.

Eine heutzutage ebenfalls weit verbreitete Prozessorvariante, die auf der RISC-Architektur beruht, wird als Advanced RISC Machine (ARM) bezeichnet. Dieser findet beispielsweise in vielen Mobilgeräten Verwendung, in denen die ARM-Prozessoren als System on Chip (SoC) verbaut sind. Im Vergleich zur CISC-Architektur zeichnen sich diese durch einfachere Befehlssätze aus, wodurch diese Prozessoren mit einer geringeren CPU-Taktung betrieben werden können. Daraus resultiert der Vorteil, dass eine geringere Leistungsaufnahme gewährleistet werden kann, und eine niedrigere Wärmeentwicklung als bei der CISC-Architektur auftritt. Dadurch sind wiederum kleinere physikalische Abmessungen möglich. Durch diese und andere Vorteile, wie u.a. ein günstigerer Preis sowie einen CPU und eine GPU die bereits im SoC integriert sind, zeichnen sich Prozessoren der RISC-Architektur besonders für mobile Anwendungen und zukünftige Produkte im Bereich Internet of Things (IoT) aus.

[9] http://www.linux-magazin.de/Ausgaben/2013/06/ARM-Architektur [Stand: 05.04.2018]

3.2.2 Flash-Speicher

Auf dem Markt der Festspeicher ist schon seit langer Zeit ein Wandel im nachgefragten Typus zu erkennen. Hierbei werden die magnetischen Speicher (engl. Hard Disk Drive, kurz HDD) nach und nach von den elektronischen Speichern, auch Flash-Speicher[10] genannt, abgelöst. Diese speichern die Informationen in Halbleitern (MOSFETs) mithilfe des sogenannten Floating-Gates. Dieses bildet eine „Ladungsfalle“, in der die elektrische Ladung gespeichert und jederzeit wieder ausgelesen werden kann (binär „1“). Das Löschen erfolgt wiederum durch das Anlegen einer negativen Spannung (binär „0“).

Die elektronischen Speicher bieten gegenüber den magnetischen Speichern unter anderem folgende Vorteile:

- Geringere Abmaße bei gleicher bzw. größerer Speicherkapazität.,
- höhere Transferraten bei der Datenübertragung,
- schnellere Zugriffszeiten,
- keine Beeinflussung der Lese- und Schreibzugriffe durch externe Einwirkungen.

3.2.3 Random Access Memory

Um ein leistungsfähiges Betriebssystem einzusetzen, ist die Anbindung von einem externen Random Access Memory[11] (RAM) unabdingbar. Dieser wird auch als Arbeits- oder Hauptspeicher bezeichnet.

Bei der Verwendung von verschiedenen Programmen und deren Prozesse müssen unterschiedliche Informationen für den Prozessor zwischengespeichert werden. Da diese Daten auch jederzeit ohne Verzögerung abgerufen werden können, wird hierfür RAM verwendet. Dieser bietet durch seinen physikalischen Aufbau schnellere Zugriffszeiten als herkömmliche Massenspeicher, wie z.B. HDD.

[10] https://www.itwissen.info/Flash-Speicher-flash-memory.html [Stand: 09.04.2018]
[11] https://www.elektronik-kompendium.de/sites/com/0309191.htm [Stand: 09.04.2018]

3.3 Software

Nachfolgend werden verschiedene Programme und Dienste beschrieben, die für diese Arbeit von Relevanz sind und Verwendung fanden.

3.3.1 Altium Designer

Zur Erstellung des Schaltungs- und Platinenlayouts wurde das Programm Altium Designer (AD)[12] verwendet. Bei dem verwendeten Programm handelt es sich hinsichtlich der zur Verfügung gestellten Funktionen um eines der umfangreichsten Programme auf dem Markt für die Entwicklung und das Design elektrischer Schaltungen. Es zeichnet sich unter anderem durch folgende Funktionen aus:

- Schaltplanerfassung (Umfangreiche Konfiguration von verschiedenen Regeln, Implementierung von verschiedenen Bibliotheken und Datenbanken, Hinzufügen von Distributoren inklusive von Preis und Menge),
- Leiterplattenlayout (ebenso umfangreiche Implementierung von verschiedensten Regeln, Darstellung von 3D-Ansicht der designten Leiterplatte und der verwendeten Komponenten),
- Schaltungssimulation.

Besonders hervorzuheben sind hierbei die ersten beiden Punkte. Durch einen umfangreich konfigurierten Regelkatalog lassen sich viele Fehler beim Layout vermeiden und eine schnellere Projektumsetzung realisieren.

Des Weiteren können durch die 3D-Daten des Projektes sogenannte STEP-Dateien exportiert werden, die die einzelnen Maße der verwendeten Bauteile enthalten und somit die Erstellung von weiteren Bauteilen, wie Gehäuse oder Adapter, vereinfachen.

[12] https://de.wikipedia.org/wiki/Altium_Designer [Stand: 23.03.2018]

3.3.2 Virtuelle Maschine

Unter einer virtuellen Maschine[13] (VM) versteht man die Implementierung von einer virtuellen Computer-Hardware, in die sich jedes unterstützte Betriebssystem und Programm installieren und ausführen lässt.

Bei einer VM werden alle physischen Komponenten, wie CPU, Festplatte, Arbeitsspeicher, Netzwerk- und andere Hardware-Ressourcen simuliert. Gezielte Anfragen von Programmen/ Diensten an die jeweiligen Komponenten werden anschließend von der VM verarbeitet und an die betreffende Hardware weitergeleitet.

Der Einsatz von virtuellen Maschinen bietet eine Vielzahl von Vorteilen gegenüber einer gewöhnlichen Installation von Betriebssystemen und Software. Die Isolation der installierten Komponenten gewährleistet, dass das primäre Betriebssystem durch fehlerhafte Anwendungen nicht beeinträchtigt oder geschädigt werden kann. Systemadministratoren können außerdem die Vorteile von virtuellen Umgebungen für einfache Backups, neue Entwicklungen und Zusatzdienste, wie z.B. Datenbanken, nutzen.

3.3.3 Eingebettete Systeme

In den nachfolgenden Unterpunkten werden verschiedene Aspekte von Eingebetteten Systemen beschrieben. Diese Informationen sind bei der Erstellung des Betriebssystems im Kapitel 6von großer Relevanz.

3.3.3.1 *Linux*

Das heute mit weitem Abstand verbreitetste Betriebssystem (engl. Operating System, kurz OS) ist unbestreitbar Linux[14]. Bei diesem handelt es sich um ein Unix-ähnliches Mehrbenutzerbetriebssystem, das unter der freien General public License (GPL)[15] veröffentlicht wird. Es besticht unter anderem durch eine komplexe Verwaltung von Benutzerrechten und eine vollständige Kontrolle über das System und seine Prozesse, was bei anderen Betriebssystemen selten in dieser Form der Fall ist.

[13] https://www.edv-lehrgang.de/virtuelle-maschine-mit-virtueller-pc/ [Stand: 13.05.2018]
[14] Linux – Das umfassende Handbuch (S. 25 f.)
[15] https://www.gnu.org/gnu/linux-and-gnu.en.html [Stand: 11.04.2018]

Linux findet Anwendung in Bereichen von Desktop-Systemen, Server-Anwendungen und der Automobilindustrie. Die derzeit wohl verbreitetsten Varianten von Linux, auch Distributionen genannt, werden aber nach wie vor in der Konsumentenelektronik eingesetzt. Dazu zählt Android, das heute auf ca. 75% der Smartphones in Deutschland[16] anzutreffen ist. Ein weiteres sehr verbreitetes Anwendungsgebiet ist die Implementierung von Linux in sogenannten Eingebetteten Systemen. Dies können verschiedenste elektronische Steuerungen und Regelungen in der Industrie (Produktionsautomatisierung) oder in Heimanwendungen (Router, Smart_TV's) sein. Durch die zukünftigen Veränderungen die mit der „Industrie 4.0" kommen, werden noch viele weitere Anwendungsgebiete folgen.

3.3.3.2 *Bootvorgang Embedded Linux*

Das nachfolgende Kapitel gibt einen Überblick über die einzelnen Schritte beim Booten von Embedded Linux. Für weitere Informationen sei an dieser Stelle auf die unten aufgeführte Quelle verwiesen[17].

Der Aufbau von Linux und dessen verschiedene Distributionen gliedert sich vereinfacht dargestellt wie folgt:

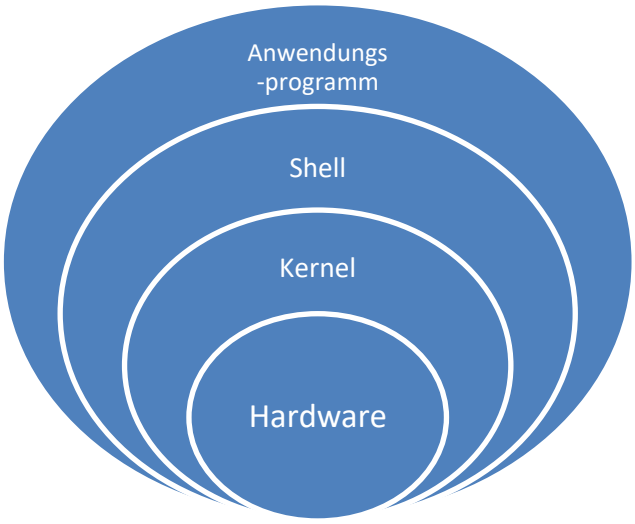

Abbildung 4: Aufbau Linux

[16] https://de.statista.com/statistik/ [Stand 11.04.2018]
[17] https://www.ibm.com/developerworks/library/l-linuxboot/ [Stand 11.04.2017]

Hierbei sind die einzelnen Bereiche miteinander verknüpft:

Hardware: Die notwendigen Bauteile zum Starten des Linux-Systems, z.B. SoC und die dazugehörige Spannungsversorgung.

Kernel: Dieser bildet das Herzstück jedes Linux-Betriebssystems. Er bildet die Verbindung zur Kommunikation zwischen der Hardware und der Shell. Diese wird wiederum durch die Geräte-Treiber realisiert.

Shell: Diese dient im Linux als Oberfläche/ Schnittstelle zwischen dem Kernel und dem Benutzer.

Anwendungsprogramme: Hierbei kommen verschiedene Dienste und Programme zum Einsatz, um die Aufgaben des Benutzers zu unterstützen bzw. zu vereinfachen.

Die allgemeine Bootsequenz variiert zwar marginal zwischen den jeweils eingesetzten SoC, ist aber bei allen Embedded-Linux-Systemen im Ablauf ähnlich und in der Abbildung 5[18] dargestellt:

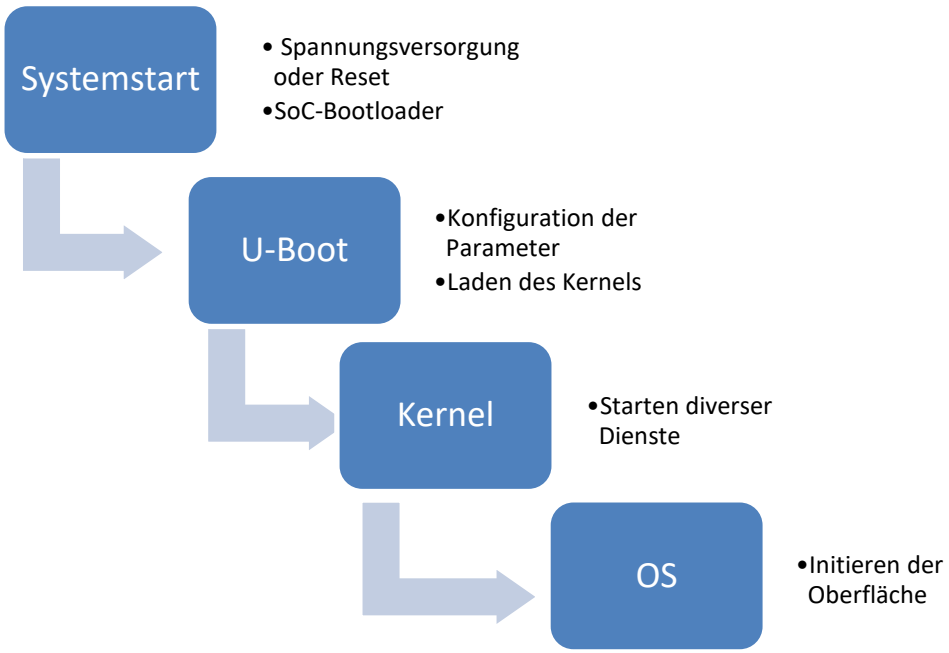

Abbildung 5: Bootprozess

[18] https://www.suse.com/de-de/documentation/sles11/singlehtml/book_sle_admin/cha.boot.html [Stand: 08.05.2018]

Systemstart: Hierbei handelt es sich um den ersten Schritt des Bootens. Dazu wird ein sogenannter POST (Power On Self Test) ausgeführt. Hierbei werden verschiedene Parameter (Spannungsversorgung, notwendige Komponenten) überprüft, die für einen erfolgreichen Systemstart notwendig sind. Des Weiteren wird die Boot-Reihenfolge des Systems ermittelt.

U-Boot: Universeller Bootloader[19] für viele verschiedene Systemarten wie beispielsweise ARM und x86. Dieser ist in zwei Teilabschnitte untergliedert, da das U-Boot (5-10 MB) nicht in den internen Speicher des SoC (128 kB) geladen werden könnte:

- SPL: (Secondary Program Loader, bei TI auch MLO genannt) Minimalistischer Bootloader, der in den RAM des SoC geladen werden kann. Dieser enthält einfache Befehle und Informationen zur Initialisierung des RAM, des Speichers und der seriellen Schnittstellen.
- U-boot.img: Nach der Initialisierung des RAM, kann das vollständige U-Boot (U-Boot.img) in diesen geladen und vom SoC verarbeitet werden. In dieser Datei sind u.a. verschiedene Befehle, Einstellungen und Treiber enthalten.

Kernel: Beim Ausführen des Kernels erfolgt sein Start in zwei Teilen. Zuerst wird der Kernel als komprimierte Datei (u-boot.img) in den Speicher geladen. In diesem Zustand können nur rudimentäre Funktionen, z.B. Speichermanagment, ausgeführt werden. Im Anschluss wird der zweite Teil des U-boot geladen (uImage). Bei diesem werden die jeweiligen Systemprozesse gestartet.

OS: Beim letzten Punkt des Bootvorganges wird das System an den Benutzer übergeben. Sollte eine grafische Benutzeroberfläche installiert sein, wird dieses ausgeführt.

[19] http://processors.wiki.ti.com/index.php/The_Boot_Process [Stand: 22.05.2018]

3.3.4 Python

Bei der Umsetzung der Endanwendersoftware wurde die Sprache Python verwendet, bei der es sich um eine höhere, universelle, und dynamisch interpretierte Programmiersprache handelt[20]. Python bietet im Vergleich zu anderen gängigen Programmiersprachen, wie beispielsweise C++, Pascal, Java, verschiedene Vorteile:

- Schnelle Umsetzbarkeit von Programmen möglich.
- Verschiedene Prinzipien der Softwareentwicklung sind umsetzbar:
    - Dont-Repeat-Yourself
    - Code-ReUse
    - Keep-It-Short-and-Simple
- Interpreter-Sprachen sind plattformunabhängig einsetzbar.
- Automatisierte Speicherverwaltung integriert.
- Eine Typenbindung ist auf Objekte beschränkt.
- Intuitiver und einfach verständlicher Programmcode aufgrund der Programmiersyntax, bspw. Werden Programmschleifen[21] durch Einrückungen gekennzeichnet).
- Open-Source und kostenfrei verfügbar.
- Umfangreiche Standardbibliothek enthalten.

Allerdings gehen mit diesen Vorteilen auch Nachteile einher. Interpreter-Programmiersprachen übersetzen zuerst den Quellcode in Byte-Code und führen diesen im Anschluss aus. Hieraus ergibt sich eine längere Ausführungszeit im Vergleich zu Compiler-Sprachen. Dies ist in den meisten Endanwendungen allerdings irrelevant, da es sich hierbei selten um zeitkritische Systeme (z.B. Sicherheitssysteme in Fahrzeugen) handelt.

[20] https://de.wikipedia.org/wiki/Python_(Programmiersprache) [Stand: 03.04.2018]
[21] https://www.python-kurs.eu/python3_bloecke.php [Stand: 12.05.2018]

3.3.5 MySQL

Bei MySQL handelt es sich um eines der meist genutzten Open-Source-Datenbankverwaltungssysteme[22]. Es basiert auf der Programmiersprache SQL (Structured Query Language), bei der es sich um eine Syntax für Datenbanken handelt, um Datensätze zu selektieren, diese zu bearbeiten, hinzuzufügen oder diese zu löschen.

Grundsätzlich besteht dieses System immer aus einem Server und mehreren Clients, die Anfragen an ihn und seine Datenbank richten. Auf dem Server können auch mehrere Datenbanken erzeugt werden. Diese bestehen wiederum aus Tabellen.

MySQL stellt auf dem Server für jede Datenbank einen separaten Ordner, in dem die Daten und die jeweilige dazugehörige Struktur der Tabelle in einer „.sql“-Datei hinterlegt ist. Durch die jeweilige verwendete Speichertechnik (Engine) kann ein effizientes Speichern von großen Datenmengen realisiert und ein performantes Abfragen der gespeicherten Daten zugesichert werden.

[22] https://www.php-einfach.de/mysql-tutorial/ [Stand: 08.05.2018]

4 Softwareanforderungen

Neben dem Betriebssystem für den Prototypen sollte außerdem eine Datenbank, zur Erfassung und Protokollierung der Daten, erstellt sowie eine Endanwendersoftware zur Rückverfolgbarkeit entwickelt werden.

In den nächsten beiden Unterkapiteln werden sowohl die Zielvorgaben an das Anwendungsprogramm, als auch der entworfene Programmablaufplan, erläutert. Hierdurch wird bereits ein Einblick in die Struktur des Algorithmus sowie in die benötigten Zubehörkomponenten gegeben.

4.1 Anforderungen an das Anwendungsprogramm

Die zu programmierende Endanwendersoftware dient in erster Linie zur Rückverfolgbarkeit von verschiedenen Produktionsschritten im Unternehmen. Durch diese Daten erlangt man einerseits Aufschluss über die Einhaltung der Reihenfolge von einzelnen Produktionsschritten. Andererseits kann auch eine exakte Aussage darüber getroffen werden, in welcher Geschwindigkeit Prozesse vollzogen werden und wie diese finanziell im Wert einer Dienstleistung zu beziffern sind.

Als Anforderungen an die zu programmierende Software ergaben sich die nachfolgend aufgeführten Punkte:

- Modularer Aufbau, um eine Verwendung an verschiedenen Arbeitsplätzen/ Prozessen zu gewährleisten (in dieser Arbeit begrenzt auf zwei Arbeitsschritte),
- Verwendung eines Handscanners um bspw. folgende Daten zu erfassen:
    - Seriennummer und Zubehörmaterial, falls diese im Arbeitsschritt verwendet werden, bspw. Handbestückung,

- Erfassung der Mitarbeiteridentifikationsnummer mithilfe eines RFID-Lesegerätes.
- Informationsausgabe über ein Display und akustische Rückmeldung im Fehlerfall.
- Protokollierung in einer Datenbankstruktur, welcher Mitarbeiter welche Daten erfasst hat und zu welchem Zeitpunkt er einen Produktionsschritt durchgeführt hat.

4.2 Softwareentwurf

In der Abbildung 6 auf der Seite 19 ist der Programmablaufplan (PAP) dargestellt. Dieser gibt ein Grundverständnis wie, und in welcher Reihenfolge, die Anforderungen aus dem Kapitel 4.1 im Programm umgesetzt werden. Hierbei ist allerdings zu beachten, dass dieser den Programmablauf nur im allgemeinen wiederspiegelt. Explizite Programmabläufe können je nach Arbeitsplatz/ Prozess abweichen und werden hier nicht dargestellt.

Zu Beginn des Programmes wird der Mitarbeiter durch eine RFID-Transponderkarte erfasst und die ausgelesene Identifikationsnummer soll über ein Display ausgegeben werden.

Nachdem die Identität des Mitarbeiters erfasst wurde, befindet sich das Programm in einer Endlosschleife. In dieser werden zu Beginn Informationen über die Leiterplatte aufgenommen. Die Erfassung kann beispielsweise durch das Einscannen eines Strichcodes oder eines Data-Matrix-Codes (DMC) erfolgen. Als Rückmeldung für den Mitarbeiter wird der eingescannte Code am Display angezeigt.

Nachdem die eingescannten Informationen im Programm verarbeitet wurden, werden diese zusammen mit einem Zeitstempel und der Mitarbeiter-ID in die Datenbank geschrieben. Im Anschluss wird geprüft, ob eine Abbruchbedingung der Endlosschleife vorliegt. Sollte dies der Fall sein, wird das Programm beendet. Andernfalls beginnt die Schleife wieder von vorn mit dem Einscannen der Leiterplatte.

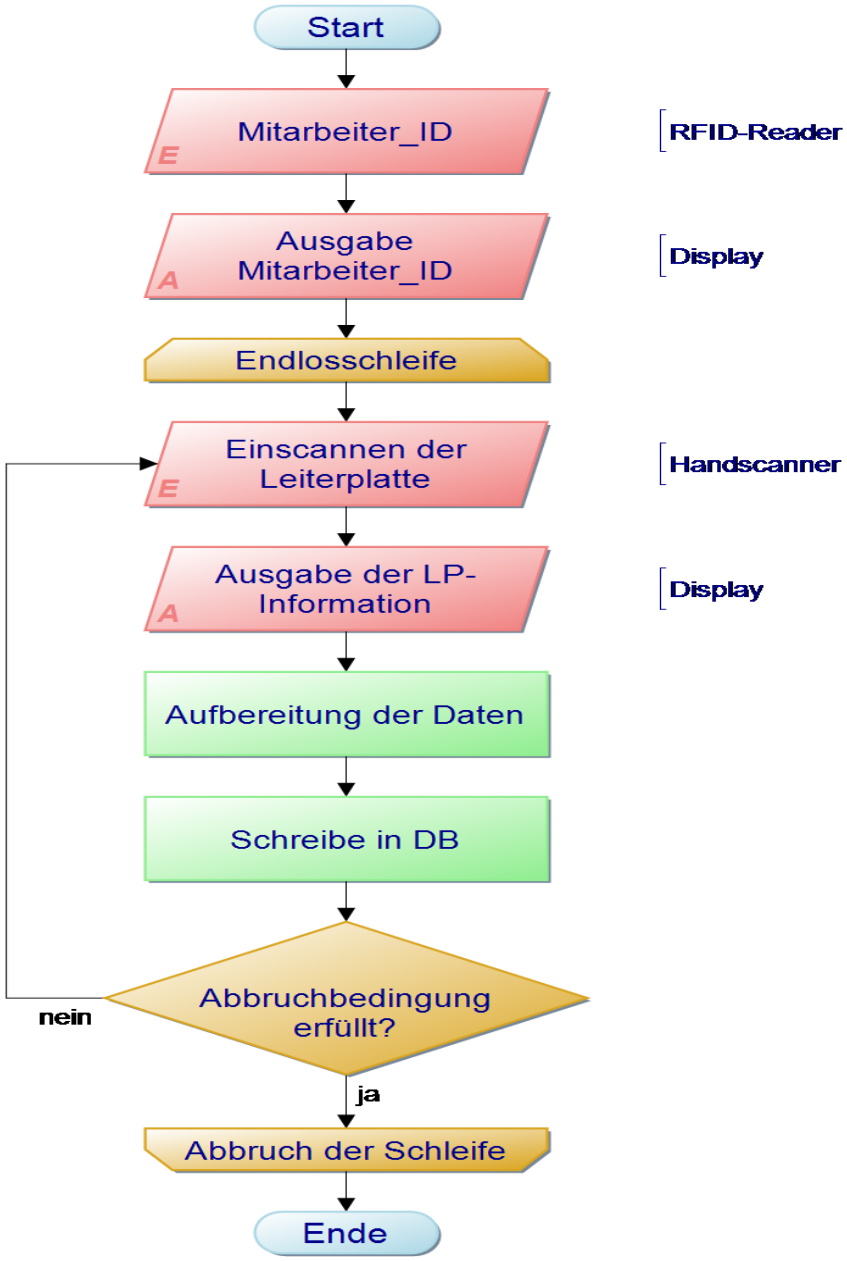

Abbildung 6: Allgemeiner Programmablaufplan

.

5 Elektronische Bestandteile des Prototyps

Im nachfolgenden Kapitel wird auf den Prototypen, die eingesetzten Bauteile sowie auf das Layout des verwendeten Einplatinencomputers eingegangen. Diese Aspekte sind für das Verständnis der nachfolgenden Kapitel 6 und Kapitel 9 notwendig, werden aber aufgrund ihrer Komplexität nicht in aller Einzelheit erläutert. Hierfür wird bereits an dieser Stelle auf die verschiedenen Datenblätter und den Schaltplan[23] im Anhang verwiesen.

Bei der Auswahl der Komponenten und dem Layout wurde sich an der Entwicklungsleiterplatte (EVM) „TMDXEVM3358“ der Firma Texas Instruments (TI) orientiert. Neben einigen Konfigurationen wurden die Abmaße und die Positionen der einzelnen Anschlüsse an die eines „RaspberryPi 2“[24] angepasst. Ein Gehäuse kann so kostengünstig erworben und muss nicht separat entworfen werden. In der nachfolgenden Abbildung ist der verwendete Prototyp dargestellt:

Abbildung 7: Raspberry PI

[23] Anhang: MA\Hardware\Leiterplatte\Schaltplan_KuG_Trace.pdf
[24] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ [Stand 08.05.2018]

5.1 Markante Prototypenbauteile

Nachfolgend wird auf die wichtigsten Bauteile der Schaltung eingegangen. Passive Komponenten wie Widerstände und Kondensatoren werden hier nicht separat hervorgehoben. Diese wurden einerseits in Anlehnung an die Schaltungsvorschläge aus dem jeweiligen Handbuch der Bauteile, sowie andererseits nach dem Lagerbestand des betreuenden Unternehmens ausgewählt.

Die Bauteilbezeichnungen werden nachfolgend um eine Zahl in eckigen Klammern ergänzt. Diese repräsentiert die Platzierung des Bauteils in den Abbildungen im Kapitel 5.2.

5.1.1 Spannungsversorgung

Die Spannungsversorgung kann auf zweierlei Wegen erfolgen. Zum einen durch ein externes Netzteil (5V), dass über die Hohl-Power-Buchse [3] mit der Leiterplatte verbunden ist. Andererseits lässt sich der Prototyp auch über einen Micro-USB-Konnektor mit einer Spannung von 5V versorgen.

5.1.2 Prozessor, Arbeitsspeicher und Flash-Speicher

Als Prozessor wurde ein „AM3358ZCZ100“[25] [1] der Firma TI verwendet. Bei diesem handelt es sich um einen ARM-Cortex-A8 RISC-Prozessor mit einer Taktung von 1 GHz.

Der Arbeitsspeicher[26] (DDR3L SDRAM, 4 Gbit) [2] wurde entsprechend des Handbuchs des AM3358 ausgewählt. In diesem werden DDR3-Module empfohlen, mit denen eine vollständige Funktionalität gewährleistet wird. Im Zusammenspiel mit dem AM3358 arbeitet der Arbeitsspeicher mit einer Frequenz von 400 MHz.

Der Flashspeicher[27] [10] von der Firma Micron dient als nichtflüchtiger Speicher und hat eine Kapazität von 16 Gigabyte.

[25] Anhang: MA\Hardware\Bauteile\Prozessor_AM3358\Datasheet_AM3358.pdf
[26] Anhang: MA\Hardware\Bauteile\DDR3L_MT41K256M16TW107_ITP\Datasheet_MT41K256M16.pdf
[27] Anhang: MA\Hardware\Bauteile\Flash_MTFC4GLDDQ-4M_IT\Datasheet_MTFC4GLDDQ-IT.pdf

5.1.3 Bootkonfiguration

Bei dem verwendeten Prozessor wird die Bootreihenfolge der einzelnen Schnittstellen (SD-Karte, UART, Serial etc.) über sogenannte „Pullup"- bzw. „Pulldown"-Widerstände [12] konfiguriert. Bei dem eingesetzten Einplatinencomputer wurde über die Widerstände folgende Bootreihenfolge realisiert (Siehe AM3358 TRM[28] Tabelle 26-7 und Schaltplan.pdf):

1. MMC1 (Flashspeicher) → 2. MMC0 (SD-Karte) → 3. UART0 (DEBUG-Schnittstelle)

5.1.4 Schnittstellen

In den nächsten Unterkapiteln werden die Schnittstellen des Prototypen erläutert.

5.1.4.1 *SD-Karte*

Als ein Medium zum Booten des Prozessors kann eine MicroSD-Karte [11] verwendet werden. Die Speicherkapazität kann hierbei maximal 64 Gigabyte betragen.

5.1.4.2 *RJ45 und WLAN-Modul*

Die Realisierung einer Netzwerkverbindung ist sowohl kabelgebunden über eine RJ45-Buchse [5], als auch mit dem WLAN-Modul WL1801[29] [8] möglich. Hierdurch ist eine maximale Flexibilität je nach Anwendungsgebiet gewährleistet.

5.1.4.3 *USB*

Bei dem Prototyp kamen zwei verschiedene USB-Schnittstellen zum Einsatz. Der A-Konnektor [4] kann zur Kommunikation mit anderen Geräten eingesetzt werden. Außerdem ist ein Micro-USB-Anschluss [7] vorhanden. Dieser kann zur Spannungsversorgung als auch zur Kommunikation mit anderen Geräten eingesetzt werden.

5.1.4.4 *Summer*

Zur akustischen Rückmeldung an den Bediener wurde ein Summer [6] eingesetzt.

[28] Anhang: MA\Hardware\Bauteile\Prozessor_AM3358\TRM_AM3358.pdf
[29] Anhang: MA\Hardware\Bauteile\Wifi-Modul_WL1801\WL1801_HIG.pdf

5.1.5 Externe Peripherie

In den nachstehenden Unterkapiteln wird die verwendete externe Peripherie kurz veranschaulicht.

5.1.5.1 *LC-Display*

In der Abbildung 8 ist das verwendete LC-Display L2032[30] abgebildet. Dieses verfügt über zwei Zeilen mit jeweils 20 Zeichen und kommuniziert über eine parallele Verbindung.

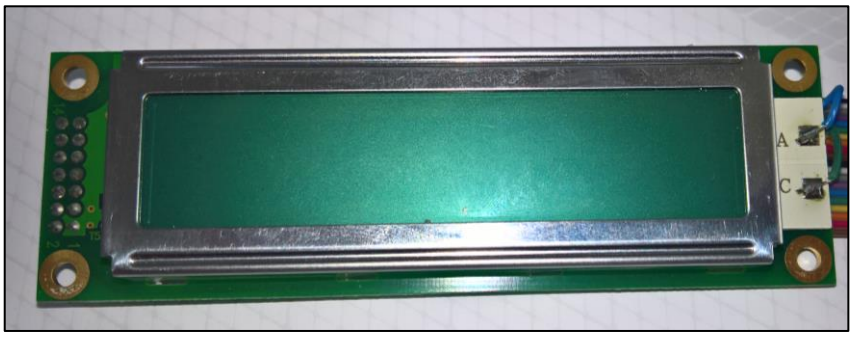

Abbildung 8: Seiko L2032

5.1.5.2 *RFID-Reader*

Zur Erfassung der RFID-Kennziffer vom jeweiligen bearbeitenden Mitarbeiter wurde ein externes RFID-Reader[31] der Firma Parallax eingesetzt. In der eingesetzten Variante wird zur Kommunikation eine serielle Verbindung verwendet.

5.1.5.3 *Handscanner*

Die Leiterplatten werden durch das Einscannen eines aufgebrachten Data-Matrix-Codes (DMC) durch den Mitarbeiter erfasst. Hierfür wird ein kabelloser Handscanner D8530 der Firma Datalogic verwendet. Der Datenaustausch wird hierbei über einen USB-Anschluss realisiert.

[30] Anhang: MA\Hardware\Bauteile\LCD_L2032\Datasheet_L2032.pdf
[31] Anhang: MA\Hardware\Bauteile\RFID-Reader_28140\Datasheet_28142.pdf

5.2 Bauteilplatzierung

In den beiden nachfolgenden Abbildungen ist einmal das generierte 3D-Bild der Leiterplatte in „TOP“- und „BOTTOM“-Ansicht dargestellt:

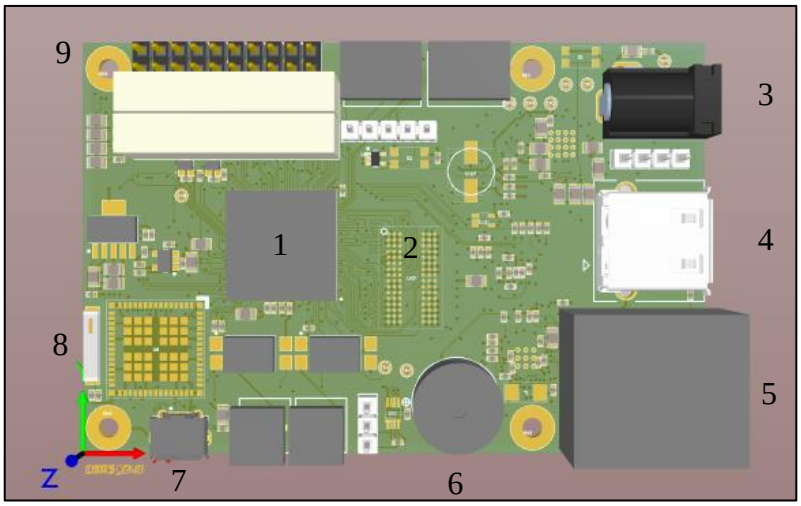

Abbildung 9: TOP-Ansicht

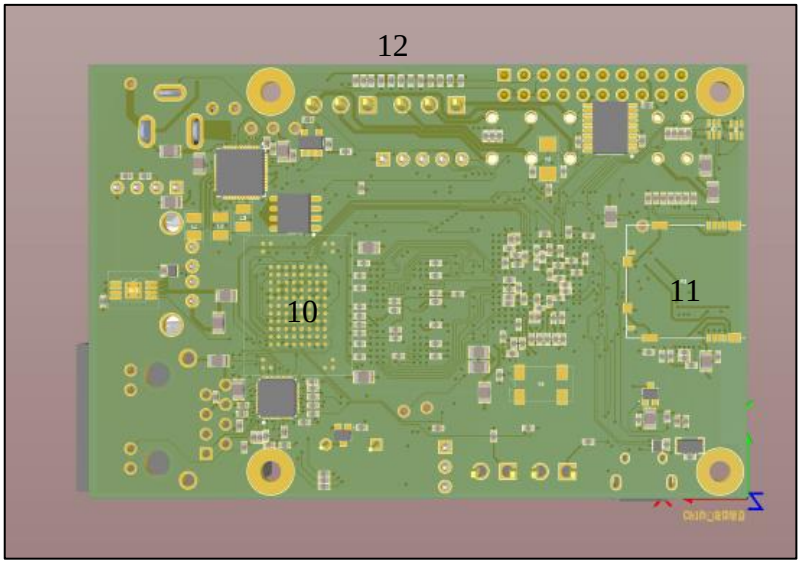

Abbildung 10: BOTTOM-Ansicht

Hier ist ein allgemeiner Überblick über die Anordnung der eingesetzten Bauteile und den Aufbau der Leiterplatte gegeben. Dazu muss allerdings erwähnt werden, dass nicht für alle Komponenten ein 3D-Modell verfügbar war. Für eine Übersicht des Aufbaus ist die Darstellung allerdings vollkommen ausreichend.

6 Konfiguration und Generierung des Betriebssystems

Da sich die Entwicklung, Konfiguration und Generierung von U-Boot und Betriebssystems sehr komplex gestalten kann, werden sogenannte Software Development Kits (SDK) von den verschiedenen Herstellern für ihre Produkte angeboten. Diese haben in erster Linie zwei Hauptaufgaben. Einerseits die vollständige Bereitstellung der benötigten Dateien zur Konfiguration und Generierung der U-Boot-Binärdateien und des Betriebssystems. Auf der anderen Seite müssen diese Dateien konfiguriert und kompiliert werden, was durch die SDK's vereinfacht wird.

Wie im vorherigen Kapitel 5.1.2 bereits beschrieben wurde, ist als SoC ein AM3358 der Firma TI verwendet worden. Als SDK kam das SDK AM335x in der Version 04.01.00.06 zum Einsatz.

Da das SDK allerdings nur für das Betriebssystem Ubuntu angeboten wird, wurde dieses in der Version 16.04, auch Xenial genannt, als virtuelle Maschine mithilfe des Programmes VirtualBox[32] in der Version 5.2.8 installiert.

In den nachfolgenden Unterkapiteln werden die grundlegenden Schritte beschrieben, die zur Konfiguration und Generierung der benötigten Softwarekomponenten nötig sind. Auf etwaige verwendete Linux-Befehle kann allerdings nicht detailliert eingegangen werden, da dies über den Rahmen dieser Arbeit hinausgehen würde. Um eine grobe Übersicht verschiedenster Linux-Befehle und deren Beschreibung zu erhalten, ist die Zusammenfassung „Linux auf einem Blatt"[33] im Anhang[34] eingefügt.

[32] https://www.virtualbox.org/ [Stand: 03.03.2018]
[33] http://helmbold.de/artikel/Linux-auf-einem-Blatt.pdf [Stand: 08.05.2018]
[34] Anhang: MA\Dokumente\Linux_Befehlübersicht.pdf

6.1 Konfiguration der U-Boot-Dateien

Bevor das U-Boot generiert wird, müssen essentielle Dateien an die verwendeten Komponenten angepasst werden. In diesem Falle sind hiervon die Dateien „ddr_defs.h“ und „board.c“ betroffen.

6.1.1 ddr_defs.h

In „ddr_defs.h“[35] werden die einzelnen Parameter für den RAM konfiguriert. Einerseits dient hierzu das „AM335x DDR3 Timing Configuration Tool“[36] von TI. Mit diesem Werkzeug werden anhand der Parameter des Datenblattes verschiedene Werte für die „ddr_defs.h“ kalkuliert. Die restlichen fehlenden Attribute der Datei werden dem Datenblatt entnommen.

Nachfolgend ist ein Auszug aus der „ddr_defs.h“ mit den verwendeten Parametern dargestellt:

```
/* Micron MT41K256M16HA-107 MA*/
#define MT41K256M16HA107_EMIF_READ_LATENCY      0x100007
#define MT41K256M16HA107_EMIF_TIM1              0x0AAAD4DB
#define MT41K256M16HA107_EMIF_TIM2              0x286B7FDA
#define MT41K256M16HA107_EMIF_TIM3              0x501F867F
#define MT41K256M16HA107_EMIF_SDCFG             0x61C05332
#define MT41K256M16HA107_EMIF_SDREF             0xC30
#define MT41K256M16HA107_ZQ_CFG                 0x50074BE4
#define MT41K256M16HA107_RATIO                  0x80
#define MT41K256M16HA107_INVERT_CLKOUT          0x0
#define MT41K256M16HA107_RD_DQS                 0x38
#define MT41K256M16HA107_WR_DQS                 0x44
#define MT41K256M16HA107_PHY_WR_DATA            0x7D
#define MT41K256M16HA107_PHY_FIFO_WE            0x94
#define MT41K256M16HA107_IOCTRL_VALUE           0x18B
```

[35] Anhang: MA\Software\Linux\ddr_defs.h
[36] http://processors.wiki.ti.com/index.php/Sitara_Linux_Training:_Tuning_the_DDR3_Timings_on_BeagleBoneBlack [Stand: 05.05.2018]

6.1.2 board.c

Die “board.c”-Datei[37] bildet den Hauptteil vom U-Boot. Hier werden alle verwendeten Funktionen definiert und integriert. Neben vielen Bibliotheken wird auch die Datei “ddr_defs.h” in der Zeile 17 geladen.

Um die selbst ermittelten Variablen der Datei „ddr_defs.h“ in der “board.c” zu verwenden, muss der jeweilige Variablenname in den verschiedenen Funktionsblöcken eingesetzt werden. Dies ist exemplarisch im nachfolgenden Ausschnitt der “board.c” dargestellt:

```
static const struct ddr_data ddr3_data = {
        .datardsratio0 = MT41K256M16HA107_RD_DQS,
        .datawdsratio0 = MT41K256M16HA107_WR_DQS,
        .datafwsratio0 = MT41K256M16HA107_PHY_FIFO_WE,
        .datawrsratio0 = MT41K256M16HA107_PHY_WR_DATA,
}
;
```

6.2 Generierung des U-Boot

Das U-Boot wird nach der Bearbeitung der Dateien erstellt. Nachfolgend werden die dazu benötigten Befehle in ihrer Reihenfolge der Ausführung kurz erläutert. Die Ausgabeinformationen der Befehle werden aufgrund ihres Umfanges nicht mit aufgeführt, können im Anhang[38] aber eingesehen werden:

```
eric@eric-VirtualBox:~$ cd /opt/ti-processor-sdk-linux-am335x-evm-04.01.00.06/board-support/u-boot-2017.01/
eric@eric-VirtualBox:/opt/ti-processor-sdk-linux-am335x-evm-04.01.00.06/board-support/u-boot-2017.01$ sudo make clean
```

Durch den Befehl „make clean“ werden alle vorher erstellten Konfigurationsdateien entfernt, und generierte U-Boot-Dateien gelöscht.

[37] Anhang: MA\Software\Linux\board.c
[38] Anhang: MA\Software\Linux\Create_linux_log.txt

```
eric@eric-VirtualBox:/opt/ti-processor-sdk-linux-am335x-evm-04.01.00.06/board-support/u-boot-2017.01$
sudo make am335x_evm_config
```

Nach dem Ausführen von „make am335x_evm_config“ werden Grundeinstellungen vom Evaluation-Board übernommen. Da der Prototyp und das EVM sich nur gering in Aufbau und Pinkonfiguration unterscheiden, kann ein Großteil der Einstellungen übernommen werden.

```
eric@eric-VirtualBox:/opt/ti-processor-sdk-linux-am335x-evm-04.01.00.06/board-support/u-boot-2017.01$
sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

Durch „menuconfig“ können, nach der Vorkonfigurierung der Parameter, mithilfe einer grafischen Oberfläche einzelne U-Boot-Einstellungen geändert werden. Hierbei wurden allerdings im Anschluss nur Schnittstellen deaktiviert, die nicht verwendet werden und unterstützende Informationen zur etwaigen Fehlersuche (DEBUG) aktiviert.

```
eric@eric-VirtualBox:/opt/ti-processor-sdk-linux-am335x-evm-04.01.00.06/board-support/u-boot-2017.01$
sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j2
```

Durch ein einfaches „make“ werden das U-Boot und alle benötigten Hilfsdateien generiert. Durch die Endung „-j2“ wird hierbei nur die Anzahl der zu nutzenden Kerne des CPU’s zur Kompilierung bestimmt.

6.3 Erstellung einer bootfähigen SD-Karte mit Linux

Zur Erstellung einer bootfähigen SD-Karte wird vom SDK für den AM3358 ein shell-Skript (create-sdcard.sh) zur Verfügung gestellt. In der Abbildung 11 ist der allgemeine Ablauf zur Erstellung eines bootfähigen Mediums schematisch dargestellt:

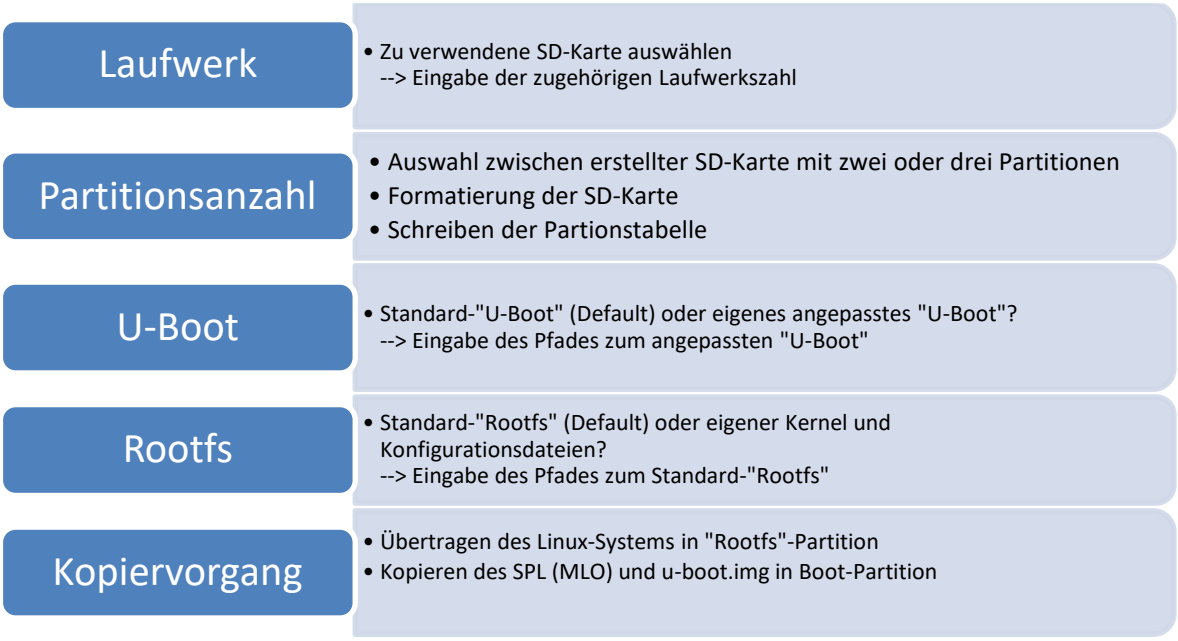

Abbildung 11: Ablauf zur SD-Karten-Erstellung

Im ersten Schritt wird die zu verwendende SD-Karte ausgewählt. Im Anschluss wird die gewünschte Partitionszahl festgelegt. In den meisten Fällen sind hierbei zwei Partitionen zu wählen, eine zum Booten und eine für das Betriebssystem. Im Falle, dass verschiedene Werkzeuge des SDK's auf der SD-Karte installiert werden sollen, ist wiederum die Option mit drei Partitionen zu wählen.

In den nächsten beiden Schritten werden das zu verwendende U-Boot und Linux-System ausgewählt. Anschließend werden alle notwendigen Daten auf die Partitionen kopiert. Hierdurch entsteht eine SD-Karte mit einer Partition „Boot" (MLO, u-boot.img etc.) und einer Partition „rootfs" mit dem Betriebssystem.

7 Testszenario der Software

Zur Erprobung des erstellten Betriebssystems, der verwendeten Datenbankstruktur und der Anwendungssoftware wurden zwei miteinander verbundene Prozesse ausgewählt, bei denen derzeit keine Erfassung der Produktionsschritte erfolgt. In den nächsten beiden Unterkapiteln werden die verwendete Baugruppe, die beiden Arbeitsplätze und die dazugehörigen Prozesse kurz erläutert, um ein besseres Verständnis für den umgesetzten Algorithmus in Kapitel „Anwendungssoftware“ zu erlangen.

7.1 Verwendete Baugruppe

Bei der im Versuchszeitraum verwendeten Baugruppe handelt es sich um ein Produkt aus der Automobilbranche. Dieses dient in Fahrzeugen zur Steuerung von Kraftstoffpumpen und wird im Unternehmen in verschiedenen Varianten hergestellt. In der Abbildung 12 ist eine Variante der Baugruppe im Nutzenverbund dargestellt. Auf dem Nutzenrand ist der DMC aufgedruckt (siehe roter Kreis), der mittels Handscanner erfasst wird.

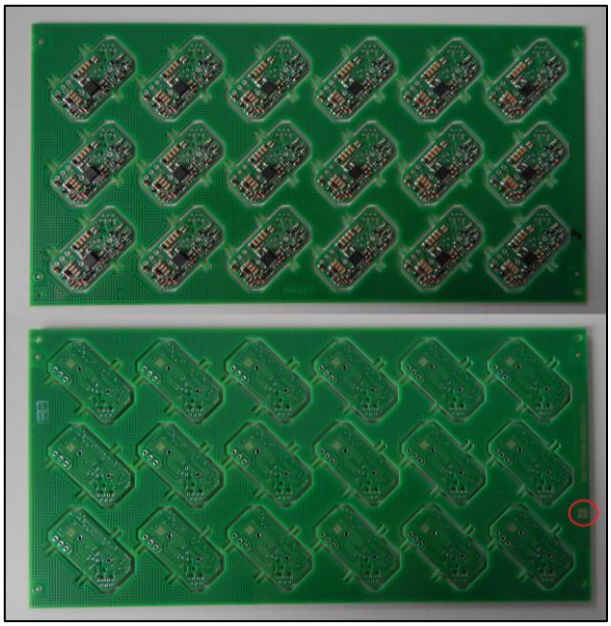

Abbildung 12: Baugruppe im Nutzen

7.2 Zugehörige Arbeitsplätze

In den nächsten beiden Unterkapiteln werden die beiden Arbeitsplätze kurz vorgestellt, die für die Testphase ausgewählt wurden. Bei diesen beiden Arbeitsplätzen erfolgt keine Rückverfolgbarkeit wie sie im Kapitel 3.1 beschrieben wurde. Es werden lediglich die DMC's der Baugruppen im Nutzenverbund in eine Excel-Tabelle eingescannt.

7.2.1 Handbestückung des Hall-Sensors

Nachdem die Baugruppe SMD-bestückt wurde, wird per Hand ein Hall-Sensor in die Baugruppe gesteckt. Hierbei wird jede Charge von Magnetfeldsensoren auf der Verpackung mit einem DMC versehen.

7.2.2 Selektivlötung des Hall-Sensors

Nach der durchgeführten Handmontage werden die Hallsensoren in einer separaten Anlage (siehe Abbildung 13) selektiv gelötet.

Abbildung 13:Selektivlötanlage

8 Erstellung der MySQL-Datenbank

Die einzelnen Arbeitsschritte sollen mithilfe des erstellten Algorithmus erfasst und protokolliert werden. Dazu werden diese nach der Akquirierung in einer Datenbank abgespeichert (Kapitel 4). Im folgenden Kapitel wird die Syntax zur Erstellung der einzelnen Tabellen erläutert und ihre Relation zueinander dargestellt.

8.1 Erstellung der Tabellen

Arbeitsplatztabelle:

Mit dem nachfolgenden SQL-Befehl wird in der Datenbank eine Tabelle mit der Bezeichnung „Arbeitsplatz“ erstellt, die aus den beiden Spalten „Arbplz_ID“ (Primärschlüssel) und „Bezeichnung“ besteht:

```
CREATE TABLE `Arbeitsplatz` (
  `Arbplz_ID` tinyint(4) unsigned NOT NULL auto_increment,
  `Bezeichnung` varchar(20) NOT NULL,
  PRIMARY KEY  (`Arbplz_ID`)
) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

Die weiteren Konfigurationen („ENGINE=MyISAM“, „CHARSET=utf8“) dienen zur Einstellung der Tabellen-Engine und der verwendeten Kodierung. Hierdurch wird der Speicherplatzbedarf optimiert und ein Durchsuchen der Tabelle beschleunigt.

Trace-Tabelle:

In der auf der folgenden Seite dargestellten Syntax wird die Tabelle „Trace“ erstellt. Diese besteht aus den Spalten „Index“ (Primärschlüssel), „Arbplz_ID“, „Zeit“, „Typ“, „BG“, „Bemerkung“ und „Material“. Hierbei wurde auch diese Tabelle mit verschiedenen Parametern so konfiguriert, z.B. „ENGINE=MyISAM“, dass ein Optimum an Speicherbedarf und Leistung erreicht wird.

```
CREATE TABLE `Trace` (
  `Index` int(11) unsigned NOT NULL,
  `Arbplz_ID` mediumint(8) default NULL,
  `Zeit` datetime NOT NULL,
  `Typ` varchar(30) default NULL,
  `BG` varchar(30) default NULL,
  `Bemerkung` varchar(45) default NULL,
  `Material` varchar(45) default NULL,
  PRIMARY KEY  (`Index`)
) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

8.2 Relation der Tabellen

Die beiden für die Datenbank erzeugten Tabellen stehen in Relation zueinander. Dies bedeutet, dass Informationen aus einer Tabelle in einer anderen verwendet werden können.

In der Abbildung 14 sind die beiden erzeugten Tabellen „Trace“ und „Arbeitsplatz“ mit Ihren Spalten gegenübergestellt. In diesem Falle wurden die beiden Tabellen über den Eintrag „Arbplz_ID“ miteinander verknüpft:

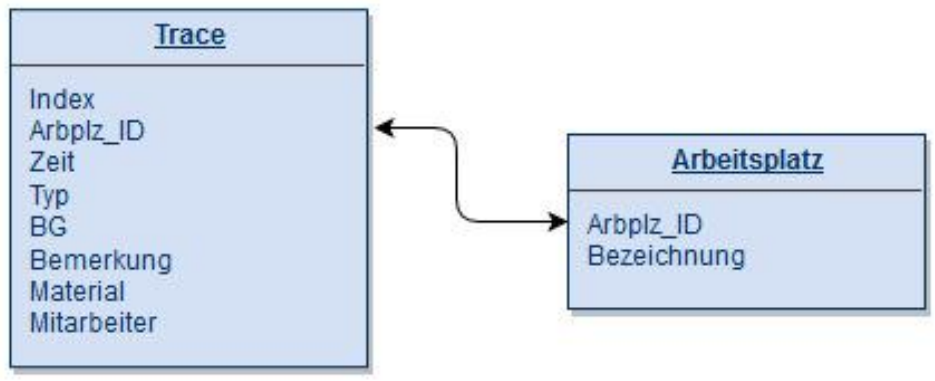

Abbildung 14: Relation der Tabellen in der Datenbank

Hierbei kann beim Hinzufügen eines neuen Datenbankeintrages in der Tabelle „Trace“ ein enormer Speicherplatz eingespart werden, da nur die zugehörige Arbeitsplatz-ID (Integer), anstelle der vollständigen Arbeitsplatzbezeichnung (String), eingetragen werden muss

9 Anwendungssoftware

In den nächsten Unterkapiteln wird das mit der Programmiersprache Python erstellte Anwendungsprogramm und seine Bestandteile erläutert. Hierbei werden allerdings nur schematische Darstellungen zur Erläuterung der Funktionsweise verwendet. Für den vollständigen Quellcode wird auf den Anhang[39] verwiesen.

9.1 Verwendete Bibliotheken

Nachfolgend sind die verwendeten Python-Bibliotheken aufgeführt und ihre Funktionen erläutert:

```
import os, sys, serial
import MySQLdb, MySQLdb.cursors
from configparser import ConfigParser
from time import localtime, strftime
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.UART as UART

UART.setup("UART4")
ser=serial.Serial(port="/dev/ttyO4", baudrate=2400, bytesize=8, parity='N', stopbits=1, timeout=1)
```

1. **os, sys:** Diese beiden Bibliotheken dienen dazu, verschiedene Systemfunktionen zu nutzen oder auch um Systeminformationen auszulesen
2. **serial:** Die Routine „serial“ wird verwendet, um eine serielle Kommunikation zu ermöglichen. In den nachfolgenden zwei Zeilen werden die zugehörigen Parameter für die Kommunikation festgelegt.
3. **MySQLdb, MySQLdb.cursors:** Hierbei handelt es sich um zwei Routinen, die zur Kommunikation mit einer MySQL-Datenbank dienen. Dadurch ist es möglich, jeden MySQL-Befehl von einem externen Programm in einer Datenbank auszuführen
4. **time:** Findet Verwendung, um Zeitfunktionen zu nutzen oder auch Datum- und Uhrzeitinformationen zu erhalten.
5. **Adafruit.BBIO.GPIO (UART):** Die beiden „Adafruit“-Bibliotheken werden verwendet, um Pins zu konfigurieren und auszulesen („High“ oder „Low“).

[39] Anhang: MA\Software\Python\trace_main.py

9.2 Verwendete Funktionen

In den nächsten Unterkapiteln werden die verschiedenen Unterfunktionen der Anwendungssoftware beschrieben, und ihre Funktionsweise erläutert.

9.2.1 lcd_init() und lcd_text(zeile,text)

Nachdem das Display eingeschaltet oder zurückgesetzt wird, kann es nicht augenblicklich verwendet werden. Bevor Informationen übertragen und angezeigt werden können, muss das Display zuerst initialisiert werden. Bei der Initialisierung werden verschiedene hexadezimale Werte in einer Sequenz, analog zum Handbuch des Displaytreibers[40], an das Gerät im 4-Bit-Modus übertragen. Dadurch werden verschiedene notwendige Register und Einstellungen konfiguriert. Die einzelnen Initialisierungsparameter sind im dargestellten Prozessplan auf der folgenden Seite in ihrer zu übermittelnden Reihenfolge dargestellt.

Die ersten vier übermittelten Variablen dienen zum zurücksetzten aller Register, um einen definierten Ausgangszustand für die Initialisierung zu gewährleisten. Im Anschluss wird das Display für die jeweiligen Bedürfnisse konfiguriert. In der programmierten Init-Prozedur wird die Display-Beleuchtung eingeschaltet. Danach werden mit dem nächsten Befehl etwaige angezeigte Informationen aus dem Display gelöscht. Mit dem letzten Informationssatz wird die Schreibrichtung des Cursors vorgegeben.

Zu den einzelnen hexadezimalen Parametern sind zugehörige Zeitwerte angegeben. Hierbei handelt es sich um Wartezeiten, die nach der Übermittlung der einzelnen Variablen mindestens eingehalten werden müssen um die Verarbeitung der Befehle zu gewährleisten, da die Initialisierung sonst fehlschlagen könnte. An einigen Stellen im Quellcode wurden höhere Wartezeiten gewählt, da mit den Wartezeitangaben des Herstellers gelegentlich Fehler bei der Initialisierung auftraten.

[40] Anhang: MA\Hardware\Bauteile\LCD_L2032\Datasheet_L2032.pdf

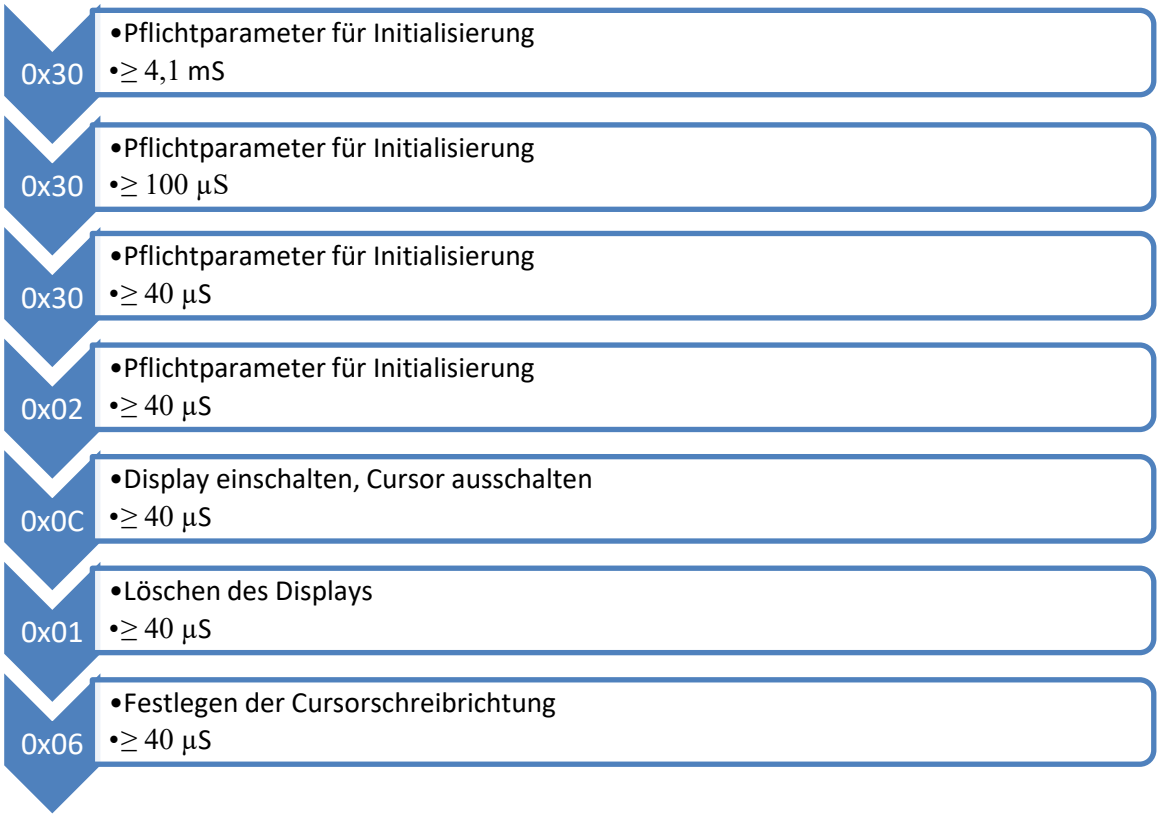

Abbildung 15: Initialisierungsprozess LC-Display

Nachdem das Display initialisiert wurde, können Informationen mit der Funktion lcd_txt() über das Display ausgegeben werden. Die Übertragung erfolgt hierbei ebenso wie bei der Initialisierung, im 4-Bit-Modus. Beim Aufruf der Funktion erfolgt eine Übergabe von zwei Variablen („zeile“ und „text“), anhand derer der übergebene String in die jeweilige Zeile geschrieben wird. Dies geschieht bei jeder Textausgabe von einer definierten Position 0;0 (Zeile 1, Zeichen1). Außerdem werden je nach Länge des Strings „text“ die ungenutzten Zeichen der Displayzeile mit „_“ aufgefüllt.

```
def lcd_text(zeile,text):
    lcd_byte(0x80, LCD_CMD)  # Setzt Cursor an Position (0,0)
    if zeile == 1:
        lcd_byte(LCD_ZEILE_1, LCD_CMD)
    if zeile == 2:
        lcd_byte(LCD_ZEILE_2, LCD_CMD)
    message=text.ljust(LCD_LAENGE,'_')
    for i in range(LCD_LAENGE):
        lcd_byte(ord(message[i]),LCD_CHR)
```

9.2.2 codescan()

Um den Handscanner mit dem dargestellten Algorithmus verwenden zu können, muss dieser konfiguriert werden. Hierbei ist es einerseits notwendig, dass der Handscanner als Tastatur (Human Interface Device) konfiguriert ist. Dadurch werden die Informationen Analog zu einer Scan-Code-Tabelle übertragen[41]. Außerdem muss automatisch am Ende jeder Übertragung ein Carriage-Return-Line-Feed (CR-LF) hinzufügt werden. Die Konfiguration erfolgt hierbei über eine Software des Herstellers.

In der folgenden Abbildung ist der Programmcode dargestellt, mit dem die Informationen vom Handscanner erfasst werden:

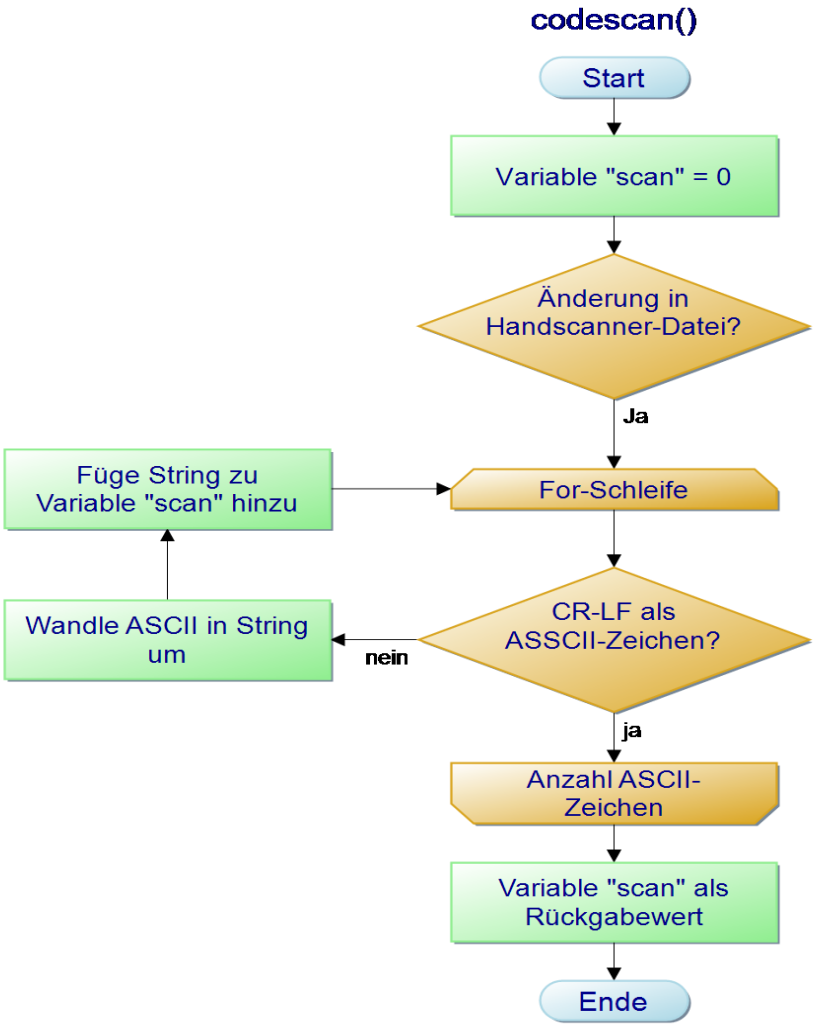

Abbildung 16: Programmablaufplan codescan()

[41] https://www.tu-chemnitz.de/informatik/RA/lehre/mop/dos_stuff/scancodes.html [Stand: 25.05.201]

Zu Beginn der Funktion wird eine Variable initiiert („scan“). Bei jedem Einscannen von Informationen, werden diese automatisch in Form von ASCII-Zeichen in eine Datei geschrieben. Sobald eine Änderung in dieser detektiert wird, beginnt eine For-Schleife über die Anzahl an Werten in dieser Datei. In jedem Durchlauf der For-Schleife wird der jeweilige ASCII-Code mit der Liste „scancodes“ abgeglichen.

```
scancodes = {
# ASCII-Zeichensatz
0: None, 1: u'ESC', 2: u'1', 3: u'2', 4: u'3', 5: u'4', 6: u'5', 7: u'6', 8: u'7', 9: u'8',
10: u'9', 11: u'0', 12: u'-', 13: u'=', 14: u'BKSP', 15: u'TAB', 16: u'Q', 17: u'W', 18: u'E', 19: u'R',
20: u'T', 21: u'Y', 22: u'U', 23: u'I', 24: u'O', 25: u'P', 26: u'[', 27: u']', 28: u'CRLF', 29: u'LCTRL',
30: u'A', 31: u'S', 32: u'D', 33: u'F', 34: u'G', 35: u'H', 36: u'J', 37: u'K', 38: u'L', 39: u';',
40: u'"', 41: u'`', 42: u'LSHFT', 43: u'\\', 44: u'Z', 45: u'X', 46: u'C', 47: u'V', 48: u'B', 49: u'N',
50: u'M', 51: u',', 52: u'.', 53: u'/', 54: u'RSHFT', 56: u'LALT', 100: u'RALT'
}
```

Nachdem der jeweilige Wert aus der Liste extrahiert wurde, wird er der Variable „scan“ hinzugefügt. Sollte kein passender Wert in der Tabelle vorhanden sein, wird der String „UNKNOWN“ hinzugefügt.

Sobald am Ende der Übertragung ein CR-LF erkannt wird, wird die Schleife beendet und die Variable „scan“ wird an das Hauptprogramm als String übergeben.

9.2.3 read_config_db()

Mithilfe der Funktion read_config_db() werden die Konfigurations- und die Zugangsinformationen zur Datenbank aus der Datei „config.ini“ eingelesen. Daraus resultiert der Vorteil, dass die verschiedenen Einstellungen für den Datenbankzugang zentral bearbeitet, verwaltet und abgerufen werden können.

9.2.4 sql_write(db_config, ap_id)

In der Abbildung 17 auf der folgenden Seite ist eine Veranschaulichung des Algorithmus zur Übertragung der akquirierten Werte in die Datenbank dargestellt.

In erster Linie werden die erfassten Daten in eine Textdatei geschrieben. Sollte temporär keine Verbindung zur Datenbank bestehen, sind die erworbenen Daten auf dem Gerät gespeichert. Im Anschluss werden die Datensätze anhand des Dateinamens mit einem entsprechenden Vermerk in die Datenbank geschrieben.

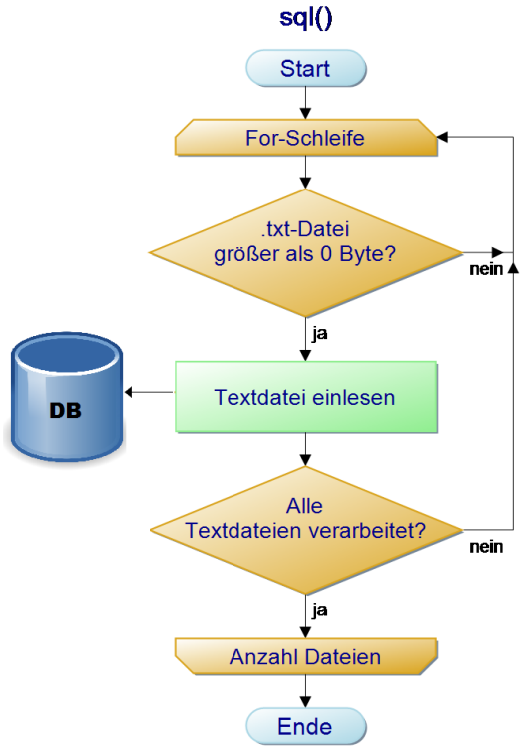

Abbildung 17: Programmablaufplan sql_write()

9.2.5 sql_select(db_config, bg)

Durch den Aufruf der Funktion „sql_select“ wird überprüft, ob die übergebene Baugruppennummer bereits vom „Arbeitsplatz 1“ bearbeitet wurde und liefert das Ergebnis der Datenbankabfrage zurück.

9.2.6 rfid_scan()

Die Funktion „rfid_scan“ findet Anwendung, um den Zugangskontrolltransponder des Mitarbeiters zu erfassen. Durch den verwendeten RFID-Reader kann die Identifikationsnummer des Transponders in Form von zehn hexadezimalen Zeichen eingelesen werden. Im zentralen Zutrittskontrollsystem (Hersteller Honeywell) des Unternehmens wird diese Zeichenfolge allerdings in einem konvertierten dezimalen Format, auch Zugangskontrollnummer (ZK-Nr.) genannt, abgespeichert. Um die Transponder-ID mit der des Zutrittskontrollsystems abgleichen zu können, muss diese vor der Übertragung in die Datenbank in die dezimale Form umgewandelt werden. Die in der Funktion „rfid_scan()“ angewandte Konvertierung ist anhand eines Beispiels von hexadezimalen Zeichen (010FEBDCAB) exemplarisch in der Tabelle 3 dargestellt:

RFID-HEX: 010FEBDCAB

1. Schritt	01				0F				EB				DC				AB			
2. Schritt	00	00	00	01	00	00	11	11	11	10	10	11	11	01	11	00	10	10	10	11
3. Schritt	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	1	0	1	1	1	0	0	1	1	1	0	1	1	1	1	0	1	0	1
Bit-Wert	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
4. Schritt	08		00		15		00		13		07		03		11		13		05	

Tabelle 3: Hex-Konvertierung

Nachfolgend werden die einzelnen Schritte der Syntax erläutert:

1. Schritt:

Im ersten Schritt werden die zehn hexadezimalen Zeichen aufgegliedert. Dadurch ergibt sich eine Liste mit fünf Elementen, die jeweils aus zwei hexadezimalen Zeichen bestehen.

2. Schritt:

Jedes hexadezimale Element der Liste wird in zwei neue Elemente aufgeteilt und diese werden jeweils in Binärzeichen (vier Bit) konvertiert.

3. Schritt:

Im 3. Schritt erfolgt eine Spiegelung jeder einzelnen Gruppe von Binärzeichen.

4. Schritt:

Im letzten Schritt werden die einzelnen Binärzeichengruppen anhand ihrer binären Wertigkeit in eine dezimale Schreibweise umgewandelt. Dadurch ergibt sich in dem dargestellten Beispiel die ZK-Nr: 08001500130703111305.

9.2.7 trace_main

Im Hauptprogramm werden die bisherigen beschriebenen Funktionen verwendet, um die Erfassung der einzelnen Datensätze zu realisieren. Die Unterprogramme sind in den nächsten Seiten an der betreffenden Stelle zur Verdeutlichung in Klammern geschrieben. Der dazu verwendete Quellcode wird in den beiden nachfolgenden Abbildungen in Form eines Programmablaufplans beschrieben. Diese beiden sind als zusammenhängend zu betrachten, und wurden nur zur Beschreibung und Übersichtlichkeit unterteilt.

Zum Programmstart wird die Datei „config.ini“ mit den Konfigurationseinstellungen für den Verbindungsaufbau zur Datenbank in eine Variable eingelesen (read_config_db)

Im Verzeichnis des Hauptprogrammes befindet sich eine Textdatei, in der die Daten bis zur Übertragung in die Datenbank zwischengespeichert werden. Über die Dateibennenung (z.B. ap2.txt) dieser Textdatei, wird im Punkt „Einlesen des Arbeitsplatzes“ die Arbeitsplatzidentifikationsnummer erfasst. Diese ist identisch mit der ID aus der Datenbanktabelle „Arbeitsplatz“.

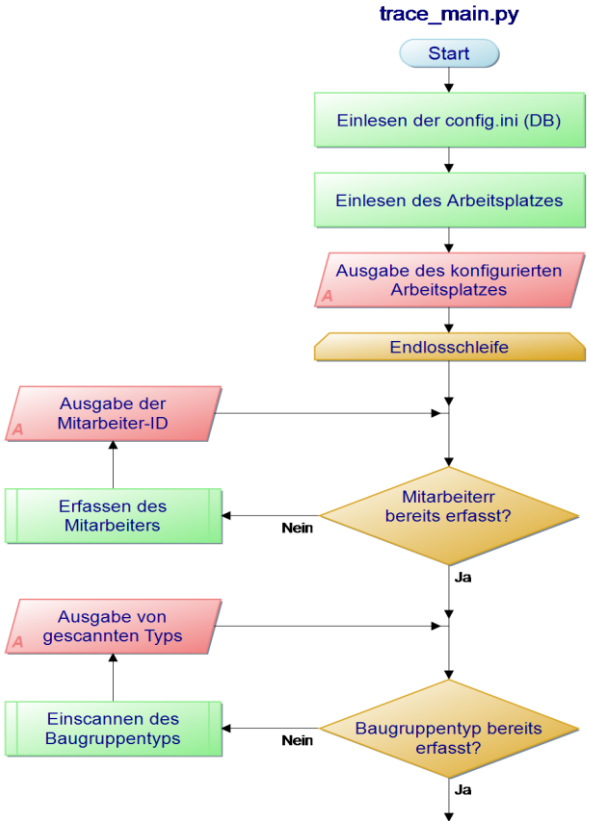

Abbildung 18: Programmablaufplan Trace_main (Teil 1)

Nach dem Einlesen der Konfigurationen befindet sich das Programm in einer Endlosschleife, in der die Datensätze erfasst werden. Zu Beginn dieser Prozedur wird überprüft, ob der ausführende Mitarbeiter (rfid_scan) und der zu bearbeitende Baugruppentyp (codescan()) bereits in einer Variable abgespeichert wurde. Sollte eine der Variablen noch keine Informationen enthalten, wird der Mitarbeiter zum Hinzufügen der fehlenden Informationen aufgefordert. Zur Überprüfung der eingegebenen Daten, werden diese im Anschluss über das Display angezeigt.

Nachdem sichergestellt wurde, dass der Baugruppentyp und die Mitarbeiter-ID in einer Variable vorhanden sind, wird in Abhängigkeit vom Arbeitsplatz noch eine Überprüfung durchgeführt, ob das verwendete Material bereits eingescannt wurde. Sollte dies nicht der Fall sein, wird der Mitarbeiter hier ebenfalls zur Erfassung des Materials aufgefordert.

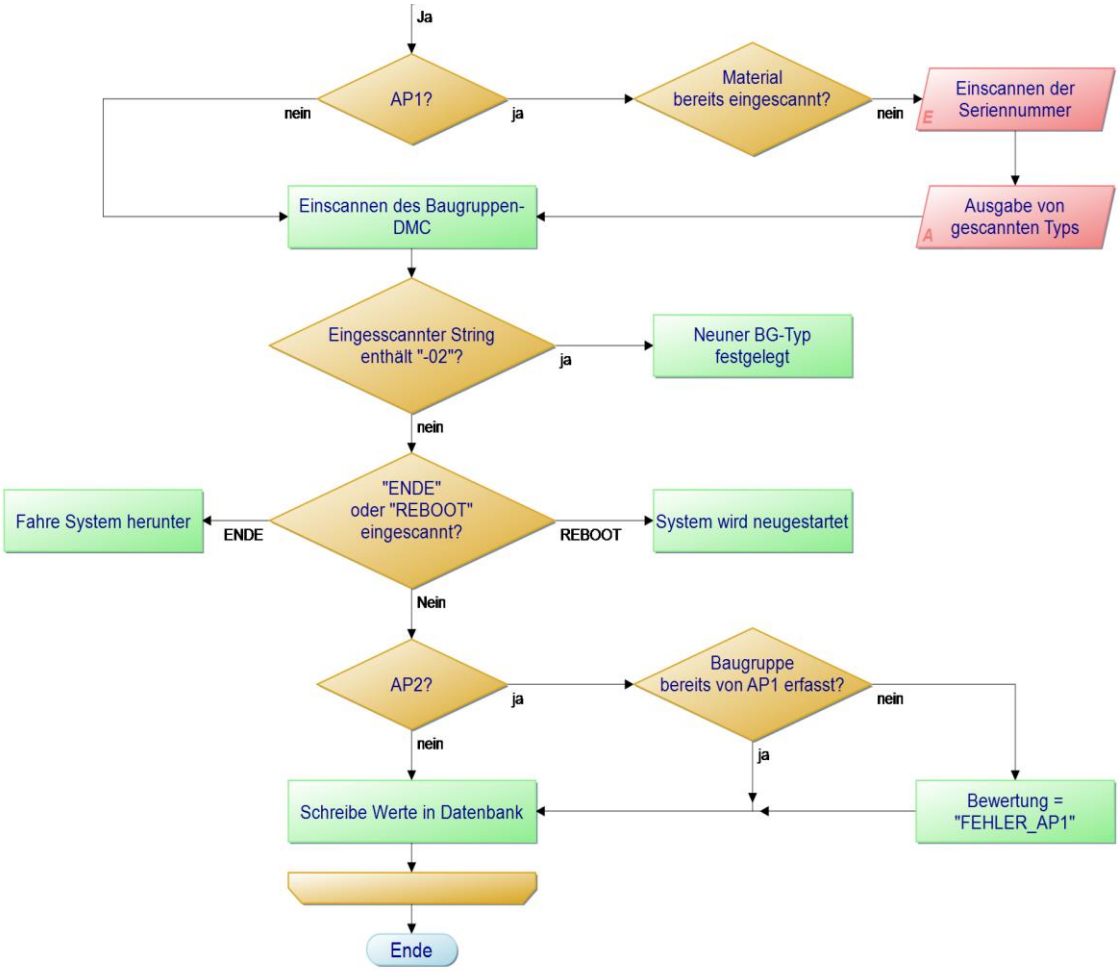

Abbildung 19: Programmablaufplan Trace_main (Teil 2)

Nach diesen einzelnen Schritten erfolgt die Erfassung des DMC's der einzelnen Baugruppen. Sobald eine Baugruppe eingescannt wurde, wird überprüft welche Information im String enthalten ist, entweder Baugruppennummer, Baugruppentyp, „ENDE" oder „REBOOT". Wurde der String „ENDE" eingelesen, wird die Aufzeichnung beendet und das System wird heruntergefahren. Wurde wiederum der String „REBOOT" erfasst, wird ein Neustart des Systems durchgeführt.

Als weitere Variante besteht die Möglichkeit, dass ein Baugruppentyp erfasst wurde. Da alle Nummern der verschiedenen Varianten vom Aufbau identisch sind (0xxxxxxxxx-02xx), erfolgt eine Überprüfung des Strings ab der zehnten Position und abzüglich der letzten beiden Zeichen (Überprüfung auf „-02“ im String; Zeile 370). Sollte hierbei festgestellt werden, dass ein Baugruppentyp eingescannt wurde, wird die Variable „Typ“ mit dem neuen Baugruppentyp überschrieben. Sollte es sich bei dem eingescannten String nicht um einen Baugruppentyp handeln, handelt es sich um eine Baugruppennummer und der erfasste String wird der Variable „bg“ (Baugruppe) zugeordnet.

Nachdem die Baugruppe erfasst wurde, erfolgt am „Arbeitsplatz 2“ noch eine Überprüfung, ob die Baugruppe auch am vorhergegangenen Arbeitsplatz bearbeitet wurde. Hierfür wird die erfasste Baugruppennummer der Funktion „sql_select“ übergeben. Sollte diese den Wert „-1“ zurückliefern, resultiert daraus das die Baugruppe noch nicht am vorhergegangenen Arbeitsplatz bearbeitet wurde und die Variable „Bewertung“ erhält die Information „FEHLER_AP1“.

Abschließend werden alle akquirierten Datensätze zeilenweise in folgender Form in die Textdatei geschrieben:

> *Zeitstempel [JJJJ-MM-TT hh:mm:ss], Baugruppentyp, Baugruppennummer, Bewertung, Material, Mitarbeiter*

Nachdem dieser Vorgang abgeschlossen ist, beginnt die Endlosschleife von vorne und die nächste Baugruppe kann erfasst werden.

10 Testaufbau und Resultate

Der Testlauf für die Software wurde im Oktober 2017 durchgeführt. In diesem Kapitel wird die Konfiguration des Testaufbaus, die erfassten Daten sowie prägnante Datensätze erläutert.

10.1 Prototypenaufbau und Konfiguration

An den beiden Arbeitsplätzen wurde der beschriebene Prototyp aus dem Kapitel 5 mit zugehörigen externen Peripherien verwendet. Der vollständige Aufbau ist exemplarisch am Arbeitsplatz der Selektivlötung in der Abbildung 20 dargestellten:

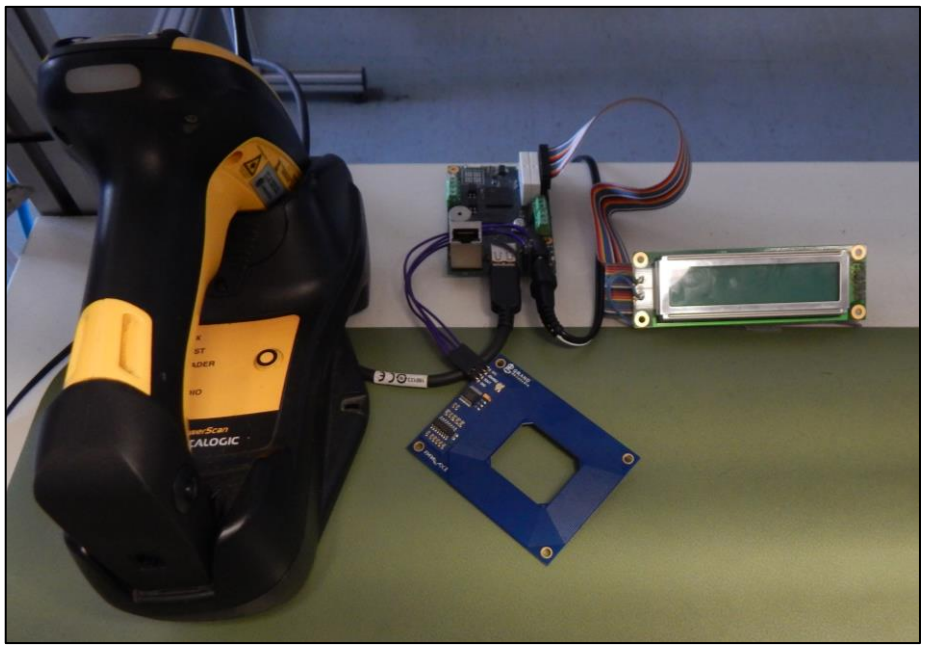

Abbildung 20: Nachbildung des Testaufbaus

Im generierten Betriebssystem (Kapitel 6.3) wurden die grundlegenden Einstellungen (Systemsprache, Uhrzeit) bei der Erstinbetriebnahme vorgenommen. Für eine Verbindung in das Unternehmensnetzwerk wurde die Einstellung für den RJ45-Konnektor geändert. Hierbei wurden die Etherneteinstellungen von einer statischen IP-Adresse zu einer Serverseitigen Konfiguration mittels DHCP (Dynamic Host Configuration Protocol) konfiguriert.

Des Weiteren wurde der Cron-Dienst[42] konfiguriert. Bei diesem handelt es sich um einen Prozess, der dauerhaft die Uhrzeit überwacht und je nach Konfiguration beliebige Befehle ausführen kann.

Nachfolgend sind die beiden Befehle dargestellt, die bei jedem Systemstart durch den Cron-Dienst ausgeführt werden:

```
eric@traceability:~# crontab -e

#
@reboot sudo /usr/bin/python /home/eric/Desktop/python/trace_main.py
@reboot sudo /usr/bin/python /home/eric/Desktop/python/systemuhr.py
```

Der erste Eintrag ist dafür zuständig, dass das Anwendungsprogramm nach dem Systemstart automatisch auszuführt wird. Der zweite Eintrag ist für die automatische Synchronisation der Systemuhrzeit zuständig. Dies würde regulär durch einen Abgleich mit einem NTP-Server (Network Time Protocol) erfolgen, ist aber durch die Unternehmens-Firewall blockiert. Deshalb wurde eine Script-Datei[43] erstellt, die eine Abfrage der aktuellen mitteleuropäischen Zeit an einen DNS-Server (Domain Name System) sendet und im Anschluss die Systemzeit konfiguriert.

Als weiteres Hilfsmittel wurde eine Übersicht[44] mit verschiedenen DMC's vorbereitet. Diese enthielten einerseits alle Variantennummern der verschiedenen Baugruppen, die an den beiden Arbeitsplätzen verwendet werden. Andererseits wurden noch die beiden Strings „ENDE“ und „REBOOT“ als DMC hinterlegt, um verschiedene Systemfunktionen im Hauptprogramm ausführen zu können. Somit ist eine einfache Eingabe von Informationen möglich, ohne weitere Verwendung von zusätzlicher externer Peripherie.

[42] http://www.good-tutorials.de/linux/administration/cronjobs-unter-linux-einrichten-141.html [Stand: 01.06.2018]
[43] Anhang: MA\Software\Linux\systemuhr
[44] Anhang: MA\Dokumente\Varianten_Befehle_DMC.pdf

10.2 Akquirierte Messergebnisse

In der Tabelle 4 sind die erfassten Werte aus dem Testzeitraum vom Oktober 2017 dargestellt. Diese Daten stellen allerdings nur einen Teil des vollständigen Datensatzes dar, sind aber für eine Erläuterung der Messwerte vollkommen ausreichend. Für den vollständigen Datensatz wird an dieser Stelle auf den Anhang[45] verwiesen.

Index	Arbplz_ID	Zeit	Typ	Baugruppe	Bemerkung	Material	Mitarbeiter
5087	1	2017-10-12 6:00:50	0481800116-0200	0818510374360011311170157		0KGA381802363-0200:57:570100006996817000 9RALT	0800263076 9807591777
5088	2	2017-10-12 6:02:03	0481800116-0200	0818510374360011311170158			0800495877 3998118165
5089	1	2017-10-12 6:03:30	0481800116-0200	0818510374360011311170158		0KGA381802363-0200:57:570100006996817000 9RALT	0800263076 9807591777
5090	2	2017-10-12 6:04:59	0481800116-0200	0818510374360011311170158			0800495877 3998118165
5091	1	2017-10-12 6:06:50	0481800116-0200	0818510374360011311170158		0KGA381802363-0200:57:570100006996817000 9RALT	0800263076 9807591777
5092	2	2017-10-12 6:07:45	0481800116-0200	0818510374360011311170158			0800495877 3998118165
5093	1	2017-10-12 6:09:07	0481800116-0200	0818510374360011311170158		0KGA381802363-0200:57:570100006996817000 9RALT	0800263076 9807591777
5094	2	2017-10-12 6:10:39	0481800116-0200	0818510374360011311170158			0800495877 3998118165
5095	1	2017-10-12 6:33:57	0481800116-0200	0818510374360011311170158		0KGA381802363-0200:57:570100006996817000 9RALT	0800017345 1297712671
5096	2	2017-10-12 6:34:52	0481800116-0200	0818510374360011311170159			0800191080 0085388185
5097	1	2017-10-12 6:34:57	0481800116-0200	0818510374360011311170159		0KGA381802363-0200:57:570100006996817000 9RALT	0800017345 1297712671
5098	2	2017-10-12 6:35:02	0481800116-0200	0818510374360011311170159			0800191080 0085388185
5099	1	2017-10-12 6:35:07	0481800116-0200	0818510374360011311170159		0KGA381802363-0200:57:570100006996817000 9RALT	0800017345 1297712671
5100	2	2017-10-12 6:35:12	0481800116-0200	0818510374360011311170159			0800191080 0085388185
5101	1	2017-10-12 6:35:16	0481800116-0200	0818510374360011311170159		0KGA381802363-0200:57:570100006996817000 9RALT	0800017345 1297712671
5102	2	2017-10-12 6:36:52	0481800116-0200	0818510374360011311170159			0800191080 0085388185
5103	1	2017-10-12 6:38:13	0481800116-0200	0818510374360011311170159		0KGA381802363-0200:57:570100006996817000 9RALT	0800017345 1297712671
5104	2	2017-10-12 6:39:38	0481800116-0200	0818510374360011311170160			0800191080 0085388185
5105	1	2017-10-12 6:41:08	0481800116-0200	0818510374360011311170160		0KGA381802363-0200:57:570100006996817000 9RALT	0800017345 1297712671
5106	2	2017-10-12 6:42:41	0481800116-0200	0818510374360011311170160			0800191080 0085388185

Tabelle 4: Auszug aus der Datenbank

[45] Anhang: MA\Dokumente\Trace_DB.csv

Anhand der dargestellten Daten erhält man einen Einblick, in welcher Form die Daten in der Datenbank hinterlegt werden. Hierbei ist jedem erfassten Datensatz der jeweilige Arbeitsplatz anhand der Datenbanktabelle „Arbeitsplatz“ zugeordnet. Außerdem erfolgt über einen Zeitstempel, bestehend aus Datum und Uhrzeit (JJJJ-MM-TT hh:mm:ss), eine exakte zeitliche Zuordnung zu jedem Datensatz. Des Weiteren wird die Variante der bearbeiteten Baugruppe und ihr DMC erfasst. In den letzten beiden Spalten wird die Artikelnummer des verwendeten Materials und die konvertierte Mitarbeiternummer erfasst.

Im Abschnitt 9.2.5 wurde eine Teilsyntax des Hauptprogrammes beschrieben. Hierbei bestand eine Teilfunktion darin, Baugruppen zu markieren, die bei dem vorhergegangenen Arbeitsschritt nicht erfasst wurden. In der Tabelle 5 ist ein Eintrag im Bemerkungsfeld durch diese Teilfunktion des Hauptprogrammes dargestellt:

Index	Arbplz_ID	Zeit	Typ	Baugruppe	Bemerkung	Material	Mitarbeiter
646	2	2017-10-02 19:29:08	0481800116-0200	50271303007-12042017-0010			0800191080 0085388185
647	1	2017-10-02 19:32:25	0481800116-0200	50271303007-12042017-0010		0200:57:57:57: 57:57:57:57007	0800263076 9807591777
648	2	2017-10-02 19:33:13	0481800116-0200	50271303007-12042017-0010			0800191080 0085388185
649	2	2017-10-02 19:36:56	0481800116-0200	50271303007-12042017-0011	FEHLER_AP1		0800191080 0085388185
650	1	2017-10-02 19:37:37	0481800116-0200	50271303007-12042017-0012		0200:57:57:57: 57:57:57:57007	0800263076 9807591777
651	2	2017-10-02 19:38:11	0481800116-0200	50271303007-12042017-0012			0800191080 0085388185
652	1	2017-10-02 19:50:52	0481800116-0200	50271303007-12042017-0012		0200:57:57:57: 57:57:57:57007	0800263076 9807591777

Tabelle 5: Fehlereintrag in Datenbank

Für diesen Eintrag können mehrere Ursachen verantwortlich sein. Einerseits ist es möglich, dass die Baugruppe noch nicht am Arbeitsplatz „Handbestückung“ bearbeitet wurde und fälschlicherweise weitergegeben wurde. Andererseits ist es möglich, dass am Arbeitsplatz „Handbestückung“ ein Erfassen der Baugruppe versäumt wurde.

Als weitere prägnante Teilfunktion des Anwendungsprogrammes wurde im Unterkapitel „codescan()“ der Algorithmus beschrieben, mit dessen Hilfe Informationen vom Handscanner an das Hauptprogramm übertragen werden. In der Tabelle 6 sind Datensätze veranschaulicht, bei denen die Einträge in der Spalte „Material“ unvollständig sind:

Index	Arbplz_ID	Zeit	Typ	Baugruppe	Bemerkung	Material	Mitarbeiter
11231	2	2017-10-24 10:33:27	0481800005-0209	0018-50271303007-04072017-0004			0800017345 1297712671
11232	1	2017-10-24 10:33:32	0481800005-0209	0018-50271303007-04072017-0004		0KG114101254-0200UNKNOWN:57UNKNOWN:57UNKNOWN:	0800189448 3896061175
11233	2	2017-10-24 10:33:37	0481800005-0209	0018-50271303007-04072017-0004			0800017345 1297712671
11234	1	2017-10-24 10:34:10	0481800005-0209	0018-50271303007-04072017-0005		0KG114101254-0200UNKNOWN:57UNKNOWN:57UNKNOWN:	0800189448 3896061175
11235	2	2017-10-24 10:35:30	0481800005-0209	0018-50271303007-04072017-0005			0800017345 1297712671

Tabelle 6: Unvollständige Einträge in Spalte „Material“

Dies äußert sich durch den Teilstring „UNKNOWN“. Dieser Eintrag wird dadurch hervorgerufen, dass ein eingescanntes Symbol nicht in der Vergleichsliste „scancodes“ vorhanden ist.

11 Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, ein modulares Softwaresystem für einen Hardwareprototypen zu entwerfen, um an jedem Prozessschritt der Produktion eine Rückverfolgbarkeit gewährleisten zu können. Hierbei war die Aufgabe in drei Themengebiete unterteilt:

- Erstellung eines Betriebssystems zur Verwendung des Prototypens.
- Programmierung einer Anwendungssoftware mit der, in Verbindung mit dem Prototyp, einzelne Produktionsschritte erfasst werden können.
- Erzeugung einer Datenbankstruktur in der die Daten zur Rückverfolgbarkeit hinterlegt und abgerufen werden können.

Zur Erstellung des Betriebssystems wurde ein Embedded-Linux als Betriebssystem gewählt, das durch eine hohe Leistungsfähigkeit und verschiedenste Konfigurationsmöglichkeiten besticht. Außerdem konnte durch das Betriebssystem eine Kommunikation mit verschiedenen Schnittstellen und Peripherien der Hardware realisiert werden.

Bei der Umsetzung der Anwendungssoftware konnte ein effizientes und modulares Programm zur Rückverfolgbarkeit in der Programmiersprache Python erstellt werden. Außerdem ist durch die Verwendung von dieser Programmiersprache ein Einsatz der Hardware und seiner Ressourcen und Schnittstellen, z.B. Relaissteuerung oder Optokoppler, jederzeit möglich. Dadurch wären beispielsweise Automatisierungen in der Mess- und Prüftechnik denkbar.

Bei der dritten Teilaufgabe, dem Entwurf und der Implementierung der Datenbank, wurde für den ersten Testdurchlauf eine MySQL-Datenbank mit zwei Tabellen erstellt. Durch die Verwendung von MySQL ist es möglich, große Mengen an Messdaten effizient zu erfassen und diese ebenso auszuwerten.

Besonders interessant ist die Betrachtung der erfassten Datensätze, die an den beiden ausgewählten Arbeitsplätzen akquiriert werden konnten. Durch die Menge an Datensätzen (ca. 15000 im Testzeitraum) kann die Möglichkeit zur umfangreichen Analyse und Auswertung der Prozesse und der dazugehörigen Arbeitsschritte gewährleistet werden. Dadurch ist eine Optimierung der Prozessabläufe, eine genauere Preiskalkulation der Baugruppen, als auch eine genauere Zuordnung der betroffenen Baugruppen im Fehlerfall, möglich.

Derzeit werden im betreuenden Unternehmen alle Prozessabläufe analysiert, bei denen keine oder nur eine mangelhafte Prozessaufzeichnung durchgeführt wird. Dadurch soll ein Überblick gegeben werden, welche Prozesse betroffen sind und welche Informationen an diesen erfasst werden müssten. Bis zum Ende des dritten Quartals 2018 sollen kontinuierlich alle betroffenen Prozesse mit der Software, angepasst auf die jeweiligen Arbeitsplätze, ausgestattet werden.

Vor dem großflächigen Einsatz der Software im Unternehmen müssen noch einige Optimierungen durchgeführt werden. In erster Linie müssen weitere Maßnahmen zur Prävention von Fehleingaben getroffen werden. Hierbei wird beispielsweise derzeit ein Algorithmus implementiert, durch den automatisch erkannt wird, ob ein Baugruppentyp, ein Material oder eine Baugruppennummer eingescannt wurde. Des Weiteren muss die Funktion codescan() und ihre Liste „scancodes“ um fehlende Symbole erweitert werden. Dadurch lassen sich unvollständige Einträge in der Datenbank vermeiden.

Als weitere Idee zur Optimierung, ist eine Erweiterung der Datenbank denkbar. Hierbei wäre ein Hinzufügen von weiteren Tabellen sinnvoll, beispielsweise eine Typen- oder eine Materialtabelle. Dadurch könnte durch die Einsparung von redundanten Daten der Speicherbedarf und die Effizienz bei Datenbankabfragen verbessert werden. Zusätzlich wäre ein externes Programm zur Datengenerierung für andere Abteilungen sinnvoll, um relevanten Daten zur Analyse aus der Datenbank zu beziehen und diese auszuwerten.

Bezüglich der Hardware werden derzeit verschiedene Ansätze überprüft, um das zweizeilige Display durch ein Touchdisplay mit bis zu einer Größe von 10“ auszutauschen. Daraus würde der Vorteil resultieren, dass eine umfangreichere Menge an Informationen dargestellt werden kann. Außerdem würde sich durch eine weitere Möglichkeit zur Informationseingabe anbieten, und eine Kommunikation mit der Software erleichtern.

IV Tabellenverzeichnis

V Abbildungsverzeichnis

VI Quellenverzeichnis

[1] www.kug-ems.de/ueber-uns/ [Stand: 20.01.2018]

[2] ISO 9001:2015 (Qualitätsmanagementsysteme – Anforderungen)

[3] https://de.wikipedia.org/wiki/Rückverfolgbarkeit_(Produktionswirtschaft) [Stand 08.06.2018]

[4] https://www.ecr.digital/wp_contents/uploads/2017/07/Rueckverfolgbarkeit_in_der_Lieferkette.pdf (Seite 14) [Stand: 12.06.2018]

[5] http://www.etikettenwissen.de/wiki/Barcode-Etikett [Stand: 12.06.2018]

[6] https://automation-insights.blog/2018/04/04/dmc-vs-rfid-in-manufacturing/

[7] Günter Spur; (2014, 1. Auflage): Handbuch Fügen, Handhaben und Montieren, Carl Hanser Verlag

[8] https://www.kuttig.de/ems-dienstleistungen/service/rueckverfolgbarkeit.html [Stand: 12.06.2018]

[9] http://www.linux-magazin.de/Ausgaben/2013/06/ARM-Architektur [Stand: 05.04.2018]

[10] https://www.itwissen.info/Flash-Speicher-flash-memory.html [Stand: 09.04.2018]

[11] https://www.elektronik-kompendium.de/sites/com/0309191.htm [Stand: 09.04.2018]

[12] https://de.wikipedia.org/wiki/Altium_Designer [Stand: 23.03.2018]

[13] https://www.edv-lehrgang.de/virtuelle-maschine-mit-virtueller-pc/ [Stand: 13.05.2018]

[14] Michael Kofler; (2014, 1. Auflage): Linux – Das umfassende Handbuch, Rheinwerk Computing

[15] https://www.gnu.org/gnu/linux-and-gnu.en.html [Stand: 11.04.2018]

[16] https://de.statista.com/statistik/ [Stand 11.04.2018]

[17] https://www.ibm.com/developerworks/library/l-linuxboot/ [Stand 11.04.2017]

[18] https://www.suse.com/de-de/documentation/sles11/singlehtml/book_sle_admin/cha.boot.html [Stand: 08.05.2018]

[19] http://processors.wiki.ti.com/index.php/The_Boot_Process [Stand: 22.05.2018]

[20] https://de.wikipedia.org/wiki/Python_(Programmiersprache) [Stand: 03.04.2018]

[21] https://www.python-kurs.eu/python3_bloecke.php [Stand: 12.05.2018]

[22] https://www.php-einfach.de/mysql-tutorial/ [Stand: 08.05.2018]

[24] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ (Stand 08.05.2018)

[32] https://www.virtualbox.org/ [Stand: 03.03.2018]

[33] http://helmbold.de/artikel/Linux-auf-einem-Blatt.pdf [Stand: 08.05.2018]

[36] http://processors.wiki.ti.com/index.php/Sitara_Linux_Training:_Tuning_the_DDR3 Timings_on_BeagleBoneBlack [Stand: 05.05.2018]

[41] https://www.tu-chemnitz.de/informatik/RA/lehre/mop/dos_stuff/scancodes.html [Stand: 25.05.201]

[42] http://www.good-tutorials.de/linux/administration/cronjobs-unter-linux-einrichten-141.html [Stand: 01.06.2018]

VII Anlagenverzeichnis

Die folgenden Dateien sind auf der beigefügten CD vorhanden:

- Masterarbeit_20112139.pdf
- MA\Hardware\Leiterplatte\Schaltplan_KuG_Trace.pdf
- MA\Hardware\Bauteile\Prozessor_AM3358\Datasheet_AM3358.pdf
- MA\Hardware\Bauteile\DDR3L_MT41K256M16TW107_ITP\Datasheet_MT41K256M16.pdf
- MA\Hardware\Bauteile\Flash_MTFC4GLDDQ-4M_IT\Datasheet_MTFC4GLDDQ-IT.pdf
- MA\Hardware\Bauteile\Prozessor_AM3358\TRM_AM3358.pdf
- MA\Hardware\Bauteile\Wifi-Modul_WL1801\WL1801_HIG.pdf
- MA\Hardware\Bauteile\LCD_L2032\Datasheet_L2032.pdf
- MA\Hardware\Bauteile\RFID-Reader_28140\Datasheet_28142.pdf
- MA\Dokumente\Linux_Befehlübersicht.pdf
- MA\Software\Linux\ddr_defs.h
- MA\Software\Linux\board.c
- MA\Software\Linux\Create_linux_log.txt
- MA\Software\Python\trace_main.py
- MA\Hardware\Bauteile\LCD_L2032\Datasheet_L2032.pdf
- MA\Software\Linux\systemuhr
- MA\Dokumente\Varianten_Befehle_DMC.pdf
- MA\Dokumente\Trace_DB.csv

VIII Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Hohendodeleben, 18.06.2018 .....
Ort, Datum Unterschrift des Verfassers

Danksagung

Diese Arbeit möchte ich meinen Vater und meinen Opa widmen, die die Fertigstellung dieser Masterarbeit nicht mehr miterleben konnten.

Ich möchte in dieser Arbeit noch all jenen Danken, die durch ihre fachliche oder auch persönliche Unterstützung zu dieser Masterarbeit beigetragen haben. Mein Dank gilt vor allem meiner Familie. Diese haben mich zu jeder Zeit in allen Vorhaben bekräftigt und mir Kraft gegeben.

Ein großes Dankeschön natürlich auch an Prof. Dr. Dieter Schwarzenau, der diese Abschlussarbeit erst ermöglichte und mich bei der Erstellung mit konstruktiver Kritik unterstützte.

Bezugnehmend auf die Krüger & Gothe GmbH möchte ich mich bei allen Kollegen bedanken, die mich während der Anfertigung dieser Arbeit unterstützt haben. Besonders hervorzuheben ist hier einerseits Klaus Junge. Andererseits ist an dieser Stelle Marcus Weirauch zu nennen, mit dem die Zusammenarbeit immer eine Freude ist.

Das größte Dank an dieser Stelle möchte ich meiner Lebensgefährtin Christiane Brodrück aussprechen, die mich nicht nur im Korrekturlesen dieser Arbeit unterstützte, sondern in jeder Sekunde mit aufbauenden Worten, Verständnis und Rückhalt für mich da war und ohne die ich diese Arbeit nicht bewältigt hätte.