Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Alexander Bokow

Software - Entwicklung für den Leistungsmesser ADE9000 für das Elektrotechnik - Labor

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Alexander Bokow

Software - Entwicklung für den Leistungsmesser ADE9000 für das Elektrotechnik - Labor

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung

im Studiengang Mechatronik
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer/in: Prof. Dr. -Ing. Hans - Joachim Beyer
Zweitprüfer/in: Dipl. -Ing. Reinhard Breuer

Abgabedatum: 25.09.20

Alexander Bokow

Thema der Bachelorthesis

Software - Entwicklung für den Leistungsmesser ADE9000
für das Elektrotechnik - Labor

Stichworte

Leistungsmesser, ADE9000 Board, Spannungsmessung, Strommessung, Leistung, Datenspeicherung, Arduino MKR ZERO, Programmcode

Kurzzusammenfassung

Diese Bachelorthesis umfasst die Entwicklung der Steuerungssoftware für den Leistungsmesser ADE9000. Der Leistungsmesser setzt sich aus verschiedenen Komponenten zusammen, welche spezielle Konfigurationen für den Betrieb erfordern. Diese Komponenten werden näher betrachtet und mit Hilfe der Aufstellung unterschiedlicher Varianten erfolgt die Endauswahl und Konfiguration der Bestandteile des Leistungsmessers. In diesem Zusammenhang wird die erforderliche Software für die Konfigurierung und Programmierung vorgestellt und das Programm zur Steuerung des Gesamtsystems näher erläutert.

Alexander Bokow

Title of the paper

Software development for the ADE9000 power meter for the electrical engineering laboratory

Keywords

Power meter, ADE9000 Board, Voltage measurement, Current measurement, power, Data storage, Arduino MKR ZERO, Program code

Abstract

This bachelor thesis covers the development of the control software for the power meter ADE9000. The power meter is made up of various components which require special configurations for operation. These components are examined more closely and the final selection and configuration of the components of the power meter takes place with the help of the list of different variants. In this context, the software required for configuration and programming is presented and the program for controlling the entire system is explained in more detail.

Inhaltsverzeichnis

1 Einleitung

Der Bereich der Elektrotechnik bildet ein großes Einsatzgebiet. Besonders der Anwendungsbereich von Mikrocontroller – basierten technischen Geräten hat in den vergangenen Jahren immer weiter zugenommen. Auch der zunehmende Einsatz von elektrischen Anlagen setzt immer größer werdende Grundkenntnisse im Bereich der Elektrotechnik voraus, um den grundlegenden Aufbau und die Wirkungsweise der verschiedenen beteiligten elektronischen Komponenten verstehen zu können. Um diesen Anforderungen gerecht zu werden, absolvieren die Studierenden des Studiengangs Medientechnik der HAW Hamburg im Labor für elektrische Anlagen und Wandler regelmäßige vorlesungsbegleitende Praktika. Einige Laborübungen erfordern hierbei Leistungsmessgeräte, um die Spannungen, Ströme und Leistungen verschiedener Komponenten in unterschiedlichen Betriebszuständen messen zu können. Auf dem Markt gibt es ein großes Angebot von Leistungsmessgeräten. Betrachtet man den Umstand, dass für die Laborübungen 15 Laborarbeitsplätze mit Messgeräten ausgestattet werden müssen, können für die Anschaffung dieser Geräte jedoch schnell Summen zwischen 60.000 EUR und 90.000 EUR erreicht werden. Diese Anschaffungskosten dürfen außerdem ein gewisses zur Verfügung gestelltes Budget nicht überschreiten. In einer bereits durchgeführten Bachelorthesis wurde daher die Entwicklung eines 3 – Phasen – Leistungsmessers vorgestellt, um auch diese Aspekte besser berücksichtigen zu können. Ein entscheidender Vorteil ist dabei die Möglichkeit, die erforderlichen Komponenten nach den eigenen Anforderungen zusammenstellen zu können und nicht auf eine auf dem Markt erhältliche Komplettlösung zurückgreifen zu müssen. Dieser Ansatz bietet daher auch die Möglichkeit, den Kostenfaktor bzw. das maximale Budget zu berücksichtigen. Für den Leistungsmesser wurde dabei ein Board mit dem Chip ADE7758 verwendet. Das erforderliche Arduino Board (Mikrocontroller) wurde mit der Entwicklungsumgebung Arduino programmiert. Durch nicht ausreichende Genauigkeiten, welche während der Testphase ermittelt wurden, kommt dieses Leistungsmessgerät jedoch nicht für die Ausrüstung des Labors in Frage. Es musste nach einer anderen Lösung gesucht werden und es wurde ein Leistungsmessgerät auf Basis des EVAL – ADE9000 – Shields entwickelt. Es handelt sich hierbei um ein Evaluationskit des Halbleiterherstellers Analog Devices. Als die Idee entstand, das Elektrotechnik – Labor im Department Medientechnik mit diesen Messgeräten auszustatten, war der erforderliche Mikrocontroller zur Steuerung des ADE9000 allerdings nicht mehr erhältlich. Das ADE9000 – Shield war für das 32 – Bit Board Arduino Zero ausgelegt. Es wurde ein neues Layout für die Platine entworfen, um das verfügbare 32 – Bit Board MKR ZERO verwenden zu können. Die angepasste Platine wurde anschließend von der Dunst tronic GmbH hergestellt. Das auf Basis dieser Platine entwickelte Messgerät wird von einem Programm gesteuert, welches wie die Vorgängerversion mit der Entwicklungsumgebung Arduino programmiert wurde. Dieses Leistungsmessgerät bildet die Grundlage für die weiteren Betrachtungen in dieser Bachelorthesis.

2 Leistungsmessgeräte auf dem Markt

Das folgende Kapitel gibt einen Überblick über vorhandene Leistungsmessgeräte auf dem Markt. **Tabelle 1** zeigt zunächst eine Auswahl von Leistungsmessgeräten nach den Merkmalen *Anzahl der Phasen*, *Spannungsmessbereich*, *Strommessbereich*, *Auflösung* und *Preis*. Anschließend werden zwei Leistungsmesser mit Ihren jeweiligen technischen Daten aufgeführt. Diese Messgeräte werden beispielhaft vorgestellt, um die Ausstattungsmerkmale in verschiedenen Preissegmenten miteinander vergleichen zu können.

2.1 Überblick

Messgerät	Phasen (Anzahl)	Messbereich (Spannung)	Messbereich (Strom)	Auflösung		Preis
				Spannung	Strom	
GPM - 8213 	1	1 ... 700 V	1 ... 25 A	0,1 V	0,1 A	757,12 €
PCE - 360 	3	50 ... 600 V	3 ... 999,9 A	0,1 V	0,1 A	1.449,88 €
PeakTech 4145 	3	1 ... 1000 V	1 ... 300 A 1 ... 3000 A	0,1 V	0,1 A 1 A	2.067,72 €
Fluke 1736 	3	0,1 ... 1000 V	1 ... 150 A 10 ... 1500 A (mit Stromzange i17xx-flex 1500, 30 cm)	0,1 V	0,1 A	3.729,95 €
MAVOWATT 30 	3	1 ... 600 V	0,1 ... 6000 A	0,1 V	0,1 A	3.860,00 €
Norma 4000 	3	0,3 ... 1000 V	0,03 mA ... 20 A	0,1 V	0,1 V	8.070,15 €

Tabelle 1: *Leistungsmessgeräte im Überblick* [1] [2] [3] [4] [5] [6]

2.2 Leistungsmesser PeakTech 4145

Dieses 3 – Phasen Leistungsmessgerät besitzt eine TFT – Anzeige und einen Datenlogger. Das Messgerät bietet die detaillierte Darstellung von Strömen, Spannungen und Frequenzen, Oberschwingungen, Schein-, Blind-, und Wirkleistungen. Über ein Menü sind Funktionen wie z.B. Wellenformanzeige, Balkendiagramm, Trend – Anzeige, Ereignisliste und Vektorgrafik aufrufbar. Hiermit werden umfangreiche Analysefunktionen für den Industrie- und Service – Bereich zur Verfügung gestellt. Das Gerät ist außerdem mit einem internen Datenlogger zur Aufnahme von Ereignissen über einen Zeitraum von 2 Stunden bis zu 7 Tagen ausgestattet. Die Aufnahmeraten können von 1 Sekunde bis zu 60 Minuten gewählt werden. Ein integrierter LAN – Anschluss ermöglicht die Fernsteuerung über ein Netzwerk. Ein USB – Host dient zur Datenübertragung auf USB – Speichermedien. Das Messgerät bietet außerdem eine Screenshot – Funktion für USB – Speichermedien. Zum umfangreichen Zubehör gehören 5 x Prüfleitungen (3m), 4 x flexible Stromsensoren bis 3000 A, 5 x Krokodilklemmen, DC – Netzteil, Software – CD, Bedienungsanleitung und eine Tragetasche mit Schultergurt. Dieses Leistungsmessgerät eignet sich aufgrund der kompakten Abmessungen auch gut für den mobilen Einsatz.

Allgemeine technische Daten	
Anzahl der Phasen	3
Spannungseingangsbereich	1 bis 1000 V
Stromeingangsbereich	1 bis 3000 A
Anzeige	Farbe, 142 mm 320 × 240 Pixel
Speicher (für Messdaten und Screenshots)	8 Gbyte (TF)
Speicher für Einstellungen	serienmäßig
Schnittstelle	USB - Host LAN
PeakTech Power View PC - Software	Fernsteuerung des Messgerätes, zum Datenaustausch und für die Analyse
Netzanschluss	90 bis 264 V AC (50 / 60 Hz) AC - Adapter
Akku	7.2 V 3.8 Ah > 7 h Arbeitszeit

Tabelle 2: *Technische Daten PeakTech 4145* [7]

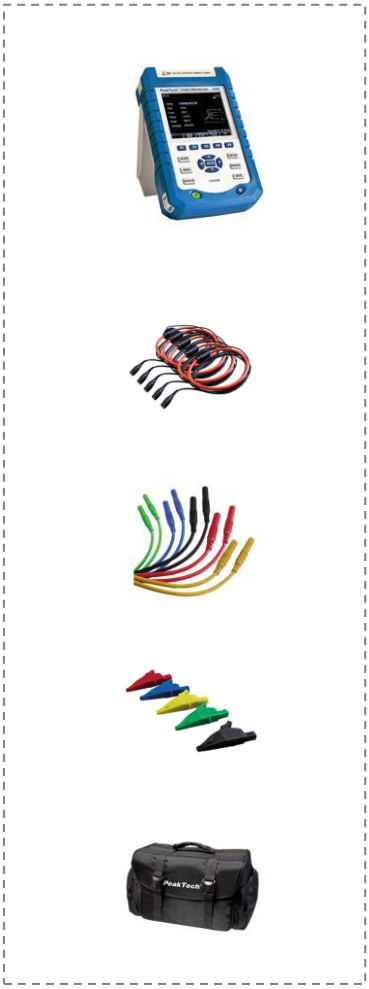


Abbildung 1: *PeakTech 4145 mit Zubehör* [3]

2.3 Leistungsmesser Norma 4000

Der 3 – Phasen Leistungsmesser Norma 4000 besitzt ein Farbdisplay und verfügt über eine Rekorderfunktion. Das Messgerät bietet außerdem die Funktionen Oberschwingungsanalyse, Oszilloskopmodus und Vektordiagrammanzeige. Die Leistungswerte Schein-, Blind-, Wirkleistung und Phasenwinkel können detailliert dargestellt werden. 4 MB Onboard – Speicher (erweiterbar auf 128 MB) stehen für die zu speichernden Messwerte zur Verfügung. Es können bis zu 15 Anwenderkonfigurationen in einen nichtflüchtigen Speicher gespeichert und später wieder geladen werden. Das Leistungsmessgerät kann schnell und einfach mit einem PC verbunden werden. RS232 und USB sind standardmäßig im Lieferumfang enthalten, IEEE 488, Ethernet oder USB 2.0 sind optional erhältlich. Wahlweise kann ein Drucker über einen externen Wandler angeschlossen werden. Für den Datenaustausch, die Analyse und Berichterstellung ist die Software NormaView im Lieferumfang enthalten.

Allgemeine technische Daten	
Anzahl der Phasen	3
Grundgenauigkeit	0,2 %, 0,1 % oder 0,03 % (je nach Eingangsmodul)
Abtastrate	0,33 MHz oder 1 MHz (je nach Eingangsmodul)
Spannungseingangsbereich	0,3 bis 1000 V
Stromeingangsbereich (direkt, nicht über einen Shunt)	0,03 mA bis 20 A (je nach Eingangsmodul)
Anzeige	Farbe, 144 mm 320 x 240 Pixel
Speicher für Messdaten	4 MB
Speicher für Einstellungen	serienmäßig
RS232 - Schnittstelle	serienmäßig
IEEE 488 GPIB - Schnittstelle Ethernet	optional
NormaView PC - Software	zum Datenaustausch, für Analyse und Berichterstellung
Netzanschluss	85 bis 264 V AC (50 bis 60 Hz) 100 bis 260 V DC Europastecker mit Schalter

Tabelle 3: *Technische Daten Norma 4000* [8]

Abbildung 2: *Norma 4000* [10]

3 Anforderungen an die Neuentwicklung

Der Überblick über die Leistungsmessgeräte auf dem Markt lässt erkennen, dass kein Leistungsmesser aus Kostengründen oder nicht ausreichender Ausstattung für die Ausrüstung des Elektrotechnik – Labors in Frage kommt. Wie schon in der Einleitung erwähnt, müssen 15 Laborarbeitsplätze je ein Messgerät erhalten. Der Leistungsmesser muss über 3 Phasen verfügen. Das Leistungsmessgerät *PeakTech 4145* würde auch den geforderten Spannungsmessbereich erfüllen, aber bei 15 Geräten würden Gesamtkosten von ca. 30.000 EUR anfallen. Das *Norma 4000* bietet eine sehr gute Ausstattung, aber der Preis ist zu hoch. Das folgende Kapitel zeigt daher die Anforderungen an die Neuentwicklung eines Leistungsmessers und dessen Komponenten für den Einsatz im Elektrotechnik – Labor. Zunächst werden die allgemeinen Anforderungen erläutert, welche an das Messgerät gestellt werden. Danach werden die Randbedingungen aufgeführt, welche von den Räumlichkeiten des Elektrotechnik – Labors gegeben sind. Die Anordnung der Bedienelemente des Leistungsmessers wird zusätzlich erläutert. Abschließend werden die Anforderungen an das Menü, die Displaysteuerung und die Datenübertragung zum PC aufgeführt.

3.1 allgemeine Anforderungen

Der Leistungsmesser muss für die Inbetriebnahme einen Schalter besitzen. Für die Laborversuche ist es erforderlich, dass die Messgeräte über 3 Phasen verfügen. Dadurch können gleichzeitig mehrere Messungen durchgeführt werden. Im 1 – Phasen – Betrieb sollten nur die Messwerte der jeweiligen Phase angezeigt werden. Vorhandene Leistungsmessgeräte auf dem Markt bieten nicht die Möglichkeit, die Messwerte aller Phasen auf einer Seite anzeigen zu lassen. Dies sollte jedoch möglich sein. Die Messwerte müssen dabei einen Genauigkeitsbereich von zwei Nachkommastellen besitzen. Der Leistungsmesser sollte die Leistungswerte Schein-, Blind-, Wirkleistung und den Leistungsfaktor darstellen können. Alle Messwerte müssen softwaremäßig berechnet werden. Der Spannungsmessbereich sollte auch Spannungen kleiner 32 V unterstützen. Ein wichtiges Merkmal für die Durchführung der Laborversuche ist die Option, Messdaten aufzunehmen und diese speichern zu können. Die Messdaten müssen dabei in einem Format vorliegen, das eine einfache weitere Bearbeitung der Daten ermöglicht. Es muss eine Schnittstelle vorhanden sein, welche den Datenaustausch mit dem PC im Labortisch ermöglicht. Ein Interface muss die Steuerung des Leistungsmessers ermöglichen. Über ein Menü, das auf einem Touch Display dargestellt wird, müssen die unterschiedlichen Betriebsmodi des Leistungsmessers auswählbar sein. Die Menüführung des Displays sollte dabei die einfache Auswahl der Phasen ermöglichen. Somit können die benötigten Messwerte angezeigt werden. Die Anzeige soll hierbei numerische Daten darstellen. Das Messgerät muss mit einer Kalibrierungsfunktion ausgestattet sein, um eine Kalibrierung der Hardware durchführen zu können. Eine weitere Anforderung ist ein möglichst kleiner Gesamtpreis aller beteiligten Komponenten bei trotzdem guter Gesamtleistung. Dies stellt in diesem Zusammenhang auch ein Hauptkriterium für die Neuentwicklung des hier beschriebe-

nen Leistungsmessgerätes dar, damit das zur Verfügung stehende Budget nicht überschritten wird. Dabei liegen die Gesamtkosten für 15 Leistungsmesser bei ca. 15.000 EUR.

3.2 Anordnung der Bedienelemente und Randbedingungen

Die Leistungsmessgeräte müssen so in die einzelnen Laborplätze integriert werden, dass die zur Verfügung stehende Fläche auf dem Labortisch nicht beeinträchtigt wird. Dadurch kann die Tischoberfläche gut für die zu messenden Komponenten genutzt werden. Das Board des Leistungsmessers befindet sich dabei in einem Gehäuse, das im oberen Teil des Labortisches platziert wird. **Abbildung 3** zeigt einen Labortisch im Labor für elektrische Anlagen und Wandler des Departments Medientechnik. Der rote Pfeil zeigt auf den Bereich, in dem das Leistungsmessgerät unterzubringen ist. Das Gehäuse des Leistungsmessers wird hinter dem Frontpanel platziert. In einem Rahmen befindet sich das Display für die Bedienung und für das Anzeigen der Messwerte. Hierbei kommt ein Touch Display zur Anwendung. Die Ein– und Ausgänge der 3 Phasen sind von der Frontseite erreichbar. Weitere erforderliche Bedienelemente werden ebenfalls auf der Frontseite untergebracht. **Abbildung 4** zeigt das Gehäuse des Leistungsmessers, in dem sich das ADE9000 Board und das Board MKR ZERO (Mikrocontroller) für die Steuerung befinden und **Abbildung 5** zeigt das Touch Display für die Bedienung und für das Anzeigen der Messwerte.

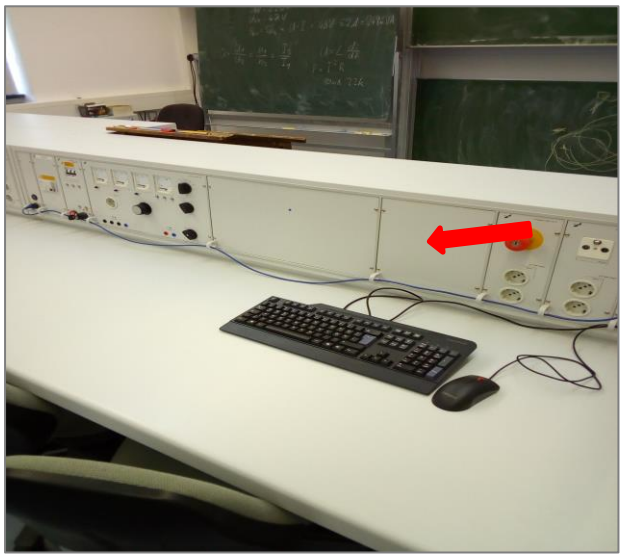


Abbildung 3: *Labortisch im Elektrotechnik - Labor*

Abbildung 4: *Leistungsmesser mit Gehäuse*

Abbildung 5: *Touch Display mit Rahmen*

3.3 Anforderung an das Menü und die Displaysteuerung

Das Menü muss es ermöglichen, die entsprechende Phase für das Anzeigen der Messwerte auswählen zu können. Außerdem müssen die Spannungen und Ströme aller Phasen auf einer Seite darstellbar sein. Die Leistungswerte der verschiedenen Phasen müssen über das Menü aufgerufen werden können. Das Menü muss eine einfache Bedienung des Leistungsmessers ermöglichen. Die Displaysteuerung muss über die Touchfunktion und eine alternative Steuerungsmöglichkeit erfolgen können.

3.4 Anforderung an die Datenübertragung zum PC

Es muss eine Möglichkeit geben, dass der PC die serielle Datenübertragung vom MKR ZERO Board zum Display mitlesen kann. Hiermit können die Messwerte auf dem PC aufgenommen werden. Es muss die Möglichkeit geben, mit einem Adapter die serielle Schnittstelle des Leistungsmessers mit der USB – Schnittstelle des PCs verbinden zu können.

4 Entwicklungsumgebung und Softwarevarianten

Der Schwerpunkt dieser Bachelorthesis liegt in der Erstellung der Software für die Steuerung des Leistungsmessers. Die erforderliche Software läuft dabei auf dem Arduino Board MKR ZERO. Dieses Board setzt für die Programmierung die Arduino – Entwicklungsumgebung voraus. Daher erfolgt zunächst ein Überblick über diese Programmierumgebung. Anschließend wird ein Überblick über die Entwicklungsumgebung des Touch Displays gegeben. Das Touch Display erfordert eine eigene Entwicklungsumgebung für die Erstellung der Oberfläche. Danach erfolgt eine Übersicht über die verschiedenen Realisierungsvarianten der Programme für die Steuerung des Leistungsmessers. Mit der Bewertung der Realisierungsvarianten wird anschließend die beste Lösung ausgewählt. Zuletzt wird auf die Datenübermittlung der Messwerte zum PC eingegangen.

4.1 Entwicklungsumgebung für die Steuerung des Leistungsmessers

Bei der Arduino – Entwicklungsumgebung handelt es sich um eine Java – Anwendung für die Programmierung von Mikrocontrollern. Diese Software lässt sich auf den Plattformen *Windows*, *Linux* und *macOS* installieren. Die Programmierung erfolgt dabei in einer C – bzw. C++ – ähnlichen Programmiersprache. Die Übertragung eines Programms auf den Mikrocontroller erfolgt über die USB – Schnittstelle. Ein Vorteil der Entwicklungsumgebung besteht darin, dass diese kostenlos erhältlich und quelloffen ist. Das Einbinden von Bibliotheken in ein Programm ermöglicht die Nutzung von Funktionalitäten, ohne dass diese in einem Programm codiert werden müssen. Über den Menüpunkt *Werkzeuge* und die Option *Bibliotheken* verwalten, können Bibliotheken der Entwicklungsumgebung hinzugefügt werden. Dies erfordert allerdings eine bestehende Internetverbindung, damit die entsprechende Bibliothek heruntergeladen werden kann. Die **Abbildung 6** zeigt die Oberfläche der Entwicklungsumgebung (*Arduino – IDE*) nach dem Starten.

Abbildung 6: *Arduino – IDE*

Ein Programm, welches mit der *Arduino – IDE* erstellt wird, enthält immer die Funktionen setup() und loop(). Die setup() – Funktion wird nur einmalig aufgerufen und enthält z.B. die Definitionen der Pins des verwendeten Arduino Boards, die als Eingang oder Ausgang genutzt werden sollen. Die loop() – Funktion wird immer wieder durchlaufen, solange das Arduino Board mit Strom versorgt wird. In dieser Funktion befindet sich das auszuführende Programm. Für das Hochladen eines Programms auf den Mikrocontroller ist es zunächst erforderlich, dass dieser in der *Arduino – IDE* ausgewählt wird. In der *Menüleiste* muss hierfür der Menüpunkt *Werkzeuge* ausgewählt werden. Anschließend ist über die Option *Board* das zu verwendende Board auszuwählen. Die **Abbildung 7** zeigt den ausgewählten Menüpunkt *Werkzeuge*. Der rote Pfeil zeigt auf das ausgewählte Board *Arduino MKR ZERO*.

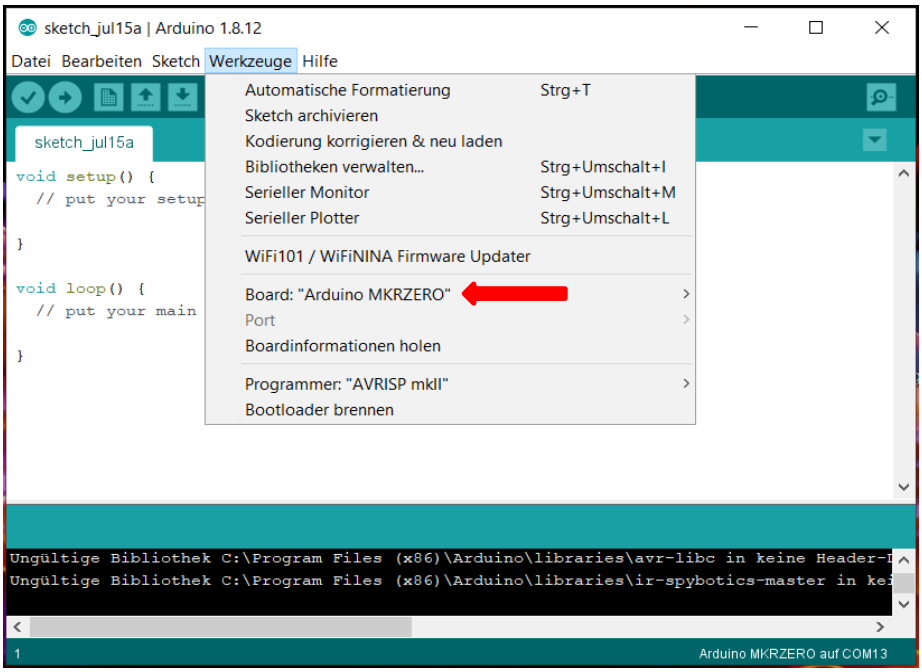


Abbildung 7: *Arduino – IDE (Auswahl des Boards / Mikrocontroller)*

Das Hochladen eines Programms auf den Mikrocontroller erfolgt über den Menüpunkt *Sketch* und die Option *Hochladen*. **Abbildung 8** zeigt den ausgewählten Menüpunkt *Sketch*. Der rote Pfeil zeigt auf die Option *Hochladen*. Auf weitere Details bzgl. der *Arduino – IDE* kann an dieser Stelle nicht eingegangen werden, weil der Schwerpunkt dieser Arbeit in der Erstellung des Programms für die Steuerung des Leistungsmessers liegt.

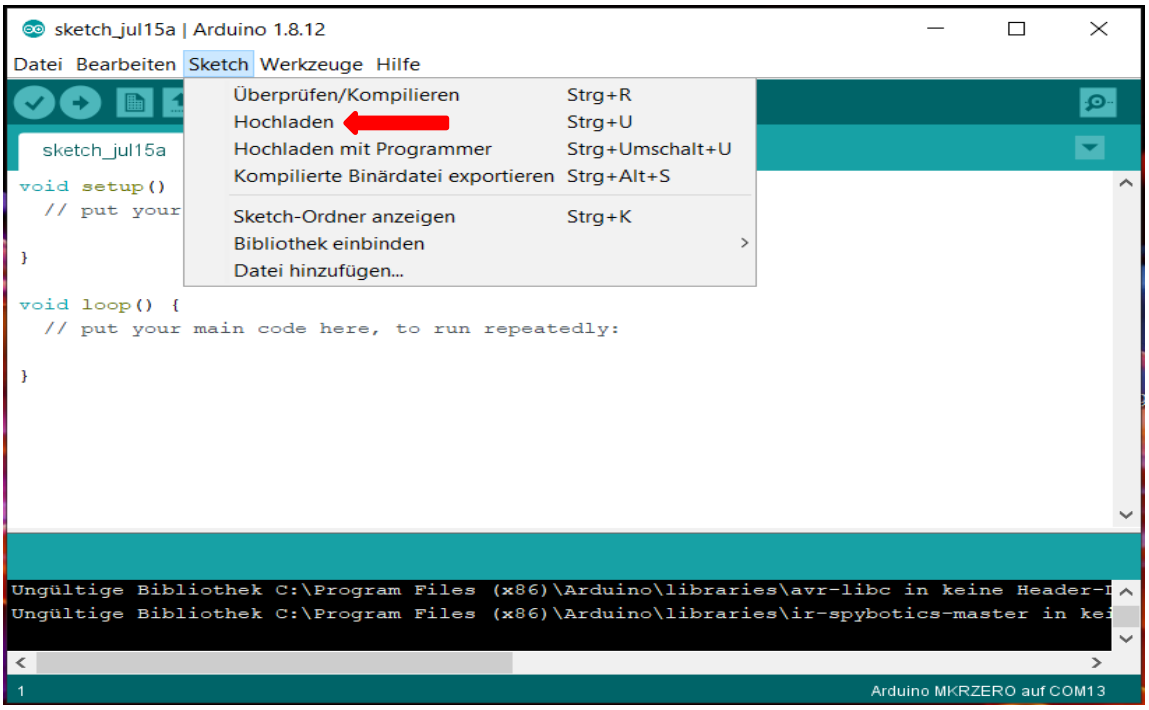


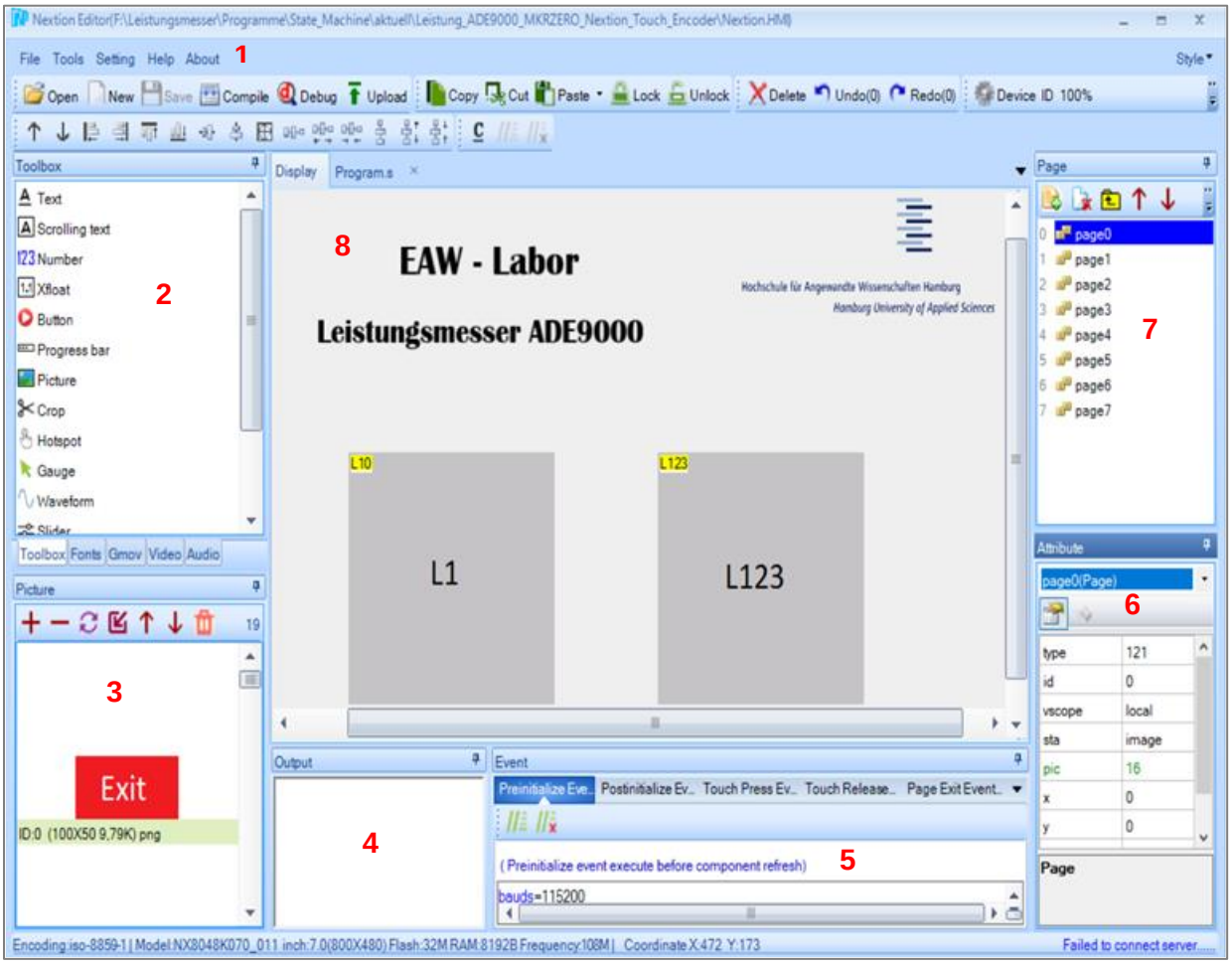
Abbildung 8: *Arduino – IDE (Auswahl der Option Hochladen)*

4.2 Entwicklungsumgebung für die Programmierung des Touch Displays

Für die Darstellung der Messwerte des Leistungsmessers kommt im endgültigen Entwurf ein *Nextion Touch Display* zur Anwendung. Das Display ist in unterschiedlichen Ausführungen erhältlich. Das für den Leistungsmesser gewählte Touch Display der Enhanced K Serie hat eine Größe von 7 Zoll und eine Auflösung von 800 x 480 Pixel. Das Touch Display besitzt einen Mikrocontroller mit der Bezeichnung *ARM Cortex M3* mit einer Taktfrequenz von 108 MHz. Das Display beinhaltet einen 32 MB großen Flash Speicher. Hier werden die Schriftarten und Bilder gespeichert, die für die Darstellung einer Oberfläche genutzt werden sollen. Das Touch Display ist außerdem mit einem 8192 Byte großen SRAM Speicher für die Speicherung von Variablen während der Laufzeit eines Programms ausgestattet. **Abbildungen 9** u. **10** zeigen das zum Einsatz kommende Nextion Touch Display mit der Bezeichnung *NX8048K070_011R*. Für die Erstellung einer Menüoberfläche ist zunächst eine Konfiguration des Touch Displays über eine kostenlos erhältliche Entwicklungsumgebung erforderlich. Dabei handelt es sich um den *Nextion Editor*, der die Erstellung von grafischen Benutzeroberflächen ermöglicht. Nach dem Starten des Programms erhält man die Oberfläche wie in **Abbildung 11** dargestellt.

Abbildung 9: *Nextion Touch Display (Vorderseite)*

Abbildung 10: *Nextion Touch Display (Rückseite)*



1 ≙ Hauptmenü
2 ≙ Toolboxbereich
3 ≙ Ressourcenbereich
4 ≙ Ausgabe
5 ≙ Benutzerereigniscode
6 ≙ Attributbereich
7 ≙ Seitenbereich
8 ≙ Display- / Programmbereich

Abbildung 11: *Nextion Editor (Programmoberfläche)*

Um eine Oberfläche für ein *Nextion* Display zu erstellen, muss zunächst im *Hauptmenü* der Menüpunkt *File* und anschließend die Option *New* gewählt werden. Jetzt kann ein Name für die zu erstellende Datei vergeben werden. Der Dateiname erhält automatisch die Endung *.HMI*. Außerdem kann der Speicherort gewählt werden. Im nächsten sich öffnenden Fenster ist das Display auszuwählen, welches zum Einsatz kommen soll. Es muss aus den Kategorien *Basic*, *Enhanced* oder *Intelligent* das entsprechende Modell gewählt werden. Das für den Leistungsmesser verwendete Display mit der Bezeichnung *NX8048K070_011R* kann unter der Kategorie *Enhanced* ausgewählt werden. Anschließend muss unter der Option *Display* die Ausrichtung des Displays für die Darstellung eingestellt werden. Es können die Ausrichtungen 0°, 90°, 180° oder 270° gewählt werden. Außerdem kann eine Einstellung der Zeichenkodierung vorgenommen werden. Die Standardeinstellung ist *iso-8859-1*. Unter der Kodierung versteht man dabei die Zuordnung eines Wertes auf ein bestimmtes Zeichen. In einer 1 Byte ASCII – Kodierung entspricht beispielsweise der Hexadezimalwert 0x41 dem Zeichen „A“. Der Wert 0x41 kann bei einer anderen Kodierung einem anderen Zeichen entsprechen. Da Computersysteme und Mikrocontroller numerische Werte für die Darstellung von Zeichen verwenden, ist es daher wichtig, die vom jeweiligen System verwendete Kodierung zu kennen. Unter der Option *Project* kann die Vergabe eines Passwortes erfolgen. Hiermit kann das gesamte Projekt, das mit dem *Nextion Editor* erstellt wird, mit einem Passwort geschützt werden. Wenn bereits ein Passwort vergeben wurde, muss dieses zunächst eingegeben werden, bevor die Vergabe eines neuen Passwortes erfolgen kann. Falls ein Passwort verloren geht, kann das mit dem *Nextion Editor* erstellte Projekt nicht wiederhergestellt werden. Für die Displays der *Intelligent* – Serie kann die Speichergröße für das Speichern von Bildern in Bytes angegeben werden. Nach Vornahme dieser Einstellungen kann mit dem Entwurf einer Oberfläche für das *Nextion* Display begonnen werden. Aus dem *Toolboxbereich* können jetzt Komponenten ausgewählt werden, die in die zu erstellende Oberfläche integriert werden sollen. Um numerische Werte darzustellen, können im *Toolboxbereich* die Komponenten *Number* oder *Xfloat* gewählt werden. Nach Auswahl der jeweiligen Komponente wird am oberen linken Rand im *Displaybereich* ein Feld erzeugt. Das erstellte Feld kann anschließend mit der Maus an eine gewünschte Position verschoben werden. Es kann aber auch über die Eingabe einer X- bzw. Y – Koordinate eine genaue Position bestimmt werden. Die zu ändernden Werte (Attribute) einer Komponente werden im *Attributbereich* angezeigt. Für die Darstellung von Fließkommazahlen muss die Komponente *Xfloat* im *Toolboxbereich* ausgewählt werden. Im Attribut *ws1* kann die Anzahl der Nachkommastellen eingegeben werden. Im Attribut *ws0* kann die Anzahl der Stellen vor dem Komma angegeben werden. Für die Darstellung einer Zahl oder eines Zeichens muss eine Schriftart (Font) erzeugt werden. Hierzu muss im Hauptmenü der Eintrag *Tools* gewählt werden. Über die Option *Font Generator* kann anschließend eine Schriftart erzeugt werden. Zu beachten ist hierbei das Feld *Encoding*, in dem die zu verwendende Zeichenkodierung gewählt werden muss. Die Zeichenkodierung muss mit der Kodierung übereinstimmen, die bereits bei der

Displayeinrichtung gewählt wurde. Um z.B. ein Zahlenfeld zu beschriften, kann im *Toolboxbereich* die Komponente *Text* gewählt werden. Das erzeugte Feld erscheint anschließend wieder in der oberen linken Ecke im *Displaybereich*. Im Attribut *txt* im *Attributbereich* kann jetzt der gewünschte Text eingegeben werden. Das Textfeld kann anschließend wieder positioniert werden. Die Schriftfarbe der erstellten Text- bzw. Zahlenfelder kann außerdem geändert werden. Über das Attribut *bco* kann die Hintergrundfarbe eines Feldes geändert werden. Über das Attribut *pco* kann die Farbe der Schrift (Font) angepasst werden. Die erzeugten Schriftarten (Fonts) können jeweils im Attribut *font* einer Komponente der jeweiligen Komponente zugewiesen werden. Durch diese Möglichkeit können z.B. die verschiedenen Elemente einer Menüoberfläche in unterschiedlichen Schriftgrößen oder Schriftarten dargestellt werden. Die verschiedenen Elemente einer darzustellenden Oberfläche können auf unterschiedlichen Seiten (pages) angeordnet werden. Hierzu können im *Seitenbereich* weitere Seiten angelegt werden. Für die Erstellung einer neuen Seite muss im *Seitenbereich* die Option *Add* gewählt werden. Die erzeugte Seite erscheint anschließend im *Seitenbereich*. Zu beachten ist die automatische Nummerierung der Seiten. Die erste Seite erhält den Index *0* und erscheint im Seitenbereich als *page0*. Um einer Oberfläche einen Button hinzuzufügen, muss im *Toolboxbereich* die Komponente *Button* ausgewählt werden. Beim Hinzufügen muss das Fenster *Benutzerereigniscode* beachtet werden. Bei Benutzung eines Buttons kann über die Optionen *Touch Press Event* und *Touch Release Event* das Verhalten gesteuert werden. Ein Ereignis kann unmittelbar nach dem Drücken oder aber nach dem Loslassen ausgelöst werden. Um z.B. nach dem Loslassen eines Buttons auf eine andere Seite zu gelangen, muss die Option *Touch Release Event* im Fenster Benutzerereigniscode markiert werden. Anschließend muss im sich darunter befindenden Textfeld der Befehl für das Wechseln der Seite hinzugefügt werden. Um beispielsweise auf die zweite Seite einer Oberfläche zu gelangen, muss dort der Befehl *page 1* eingegeben werden. Würde man den Befehl zum Wechseln einer Seite unter der Option *Touch Press Event* schreiben, dann würde das Wechseln der Seite unmittelbar beim Drücken des Buttons erfolgen. Das entspricht in den meisten Fällen aber nicht dem gewünschten Verhalten. Jede erzeugte Komponente erhält automatisch eine *ID* und einen Namen. Das Attribut *id* wird beim Hinzufügen einer Komponente automatisch vergeben und kann nicht geändert werden. Der Komponentenname kann über das Attribut *objname* geändert werden. Die Attribute *id* und *objname* sind außerdem für die Referenzierung von Komponenten aus dem Arduino – Programm erforderlich, welches über den Mikrocontroller mit dem Display kommunizieren soll. Um die Kommunikation mit einem Programm zu ermöglichen, muss die jeweilige Komponente im *Displaybereich* ausgewählt werden und anschließend ist im Fenster *Benutzerereigniscode* der Eintrag *Send Component ID* zu markieren. Auf die spezielle Menüoberfläche, die für den endgültigen Entwurf des Leistungsmessers genutzt wird und weitere Funktionen des *Nextion Editors*, wird in den *Abschnitten 4.3* und *5.1* näher eingegangen.

4.3 Realisierungsvarianten für die Steuerung

4.3.1 Variante 1

Die Variante 1 des Leistungsmessers setzt sich aus den Komponenten *Board* / *Platine*, *MKR ZERO* Board (Mikrocontroller*)* und *Display* zusammen. Bei dieser Variante werden alle Messwerte auf einer Seite angezeigt und es besteht die Möglichkeit, die gemessenen Werte in einer Textdatei zu speichern. Das Display verfügt nicht über eine Touch – Funktion. Es kann daher keine Steuerung des Leistungsmessers über das Display erfolgen. Dieser Entwurf sieht außerdem keine spezielle Menüoberfläche für das Display vor, weil auch keine weiteren Eingabegeräte wie z.B. ein Tastenfeld oder ein Encoder für die Steuerung des Leistungsmessers zum Einsatz kommen. Das verwendete Display ist ein Nextion LCD – Display der Größe 5“. Das Display besitzt einen 16 MB großen FLASH – Speicher für das Speichern von Schriften und Bildern. Außerdem ist es mit einem RAM – Speicher mit einer Größe von 3584 Byte zur Speicherung von Variablen ausgestattet. Das Layout für die Darstellung der Messwerte wurde mit dem *Nextion Editor* erstellt und anschließend in den Speicher des Displays geladen. **Abbildung 12** zeigt das Layout und die möglichen Messwerte der einzelnen Phasen (L1, L2 u. L3). Spannungen, Ströme, Scheinleistungen, Wirkleistungen, Blindleistungen und Leistungsfaktoren der einzelnen Phasen werden jeweils untereinander dargestellt. Ein Arduino – Programm, das sich im Speicher des Mikrocontrollers befindet, berechnet die Messwerte in den gewünschten Einheiten. Auf die für die Berechnung erforderlichen Werte kann über Register des Boards zugegriffen werden. Diese können über das Steuerungsprogramm ausgelesen werden. Jede Phase des Leistungsmessers wird dabei als eine eigene Funktion im Programm abgebildet. Die Messungen erfolgen über das Board bzw. dessen Komponenten. Die Messwerte werden dann über das Programm kontinuierlich in die erstellten Felder / Variablen des Nextion – Displays übertragen. Es werden bei dieser Variante also immer alle Messwerte auf dem Display angezeigt, auch wenn z.B. nur im 1 – Phasen – Betrieb gearbeitet wird. Das Speichern der Messwerte erfolgt mit einem Programm, das während der Laufzeit kontinuierlich die Werte in eine Textdatei schreibt. Dieses Programm kann optional gestartet werden.

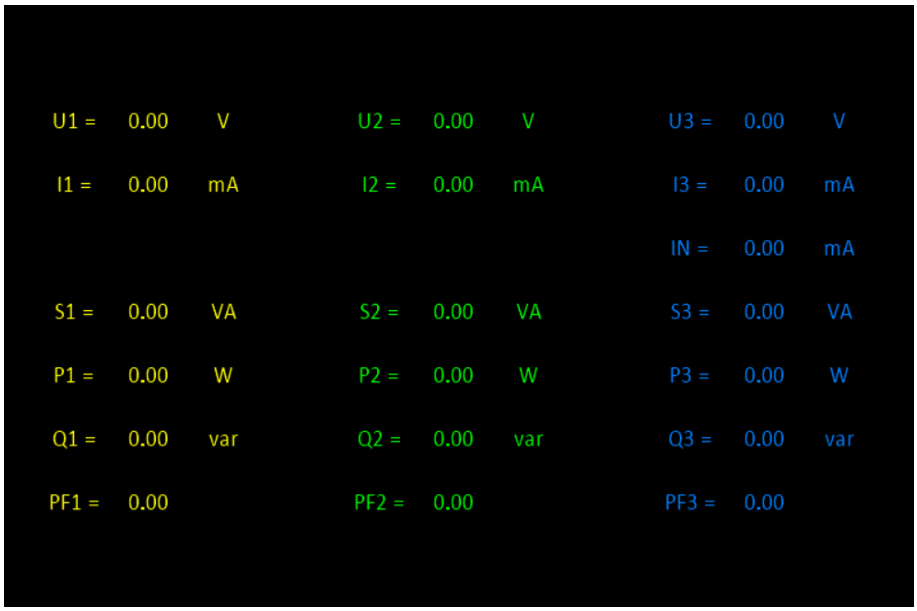


Abbildung 12: *Nextion Display (Layout Variante 1)*

4.3.2 Variante 2

Die Variante 2 des Leistungsmessers setzt sich aus den Komponenten *Board* / *Platine*, *MKR ZERO* Board (Mikrocontroller*)*, *Touch Display* und einer *Erweiterungskarte* zusammen. Bei dieser Variante können die verschiedenen Phasen des Leistungsmessers über das Touch Display aufgerufen werden. Es kann hierüber also eine Steuerung des Messgerätes erfolgen. Beim Display handelt es sich um das im *Abschnitt 4.2* erläuterte Nextion Touch Display. Die Menüführung für das Display wurde auch bei dieser Variante mit dem *Nextion Editor* erstellt und anschließend in den Speicher des Displays geladen. Die **Abbildung 13** zeigt die Hauptseite der erstellten Oberfläche. Nach dem Drücken eines Buttons (L1, L2 oder L3) gelangt man jeweils auf eine andere Seite innerhalb der Oberfläche. Wird z.B. der Button L1 gedrückt, dann werden anschließend die Messwerte Spannung, Strom, Scheinleistung, Wirkleistung, Blindleistung und Leistungsfaktor dieser Phase angezeigt (**Abbildung 14**). Nach dem Drücken des Exit Buttons gelangt man wieder auf die Hauptseite. Eine Besonderheit der Variante 2 besteht in einer Erweiterungskarte, die ergänzend für das Touch Display zum Einsatz kommt. Diese Karte besitzt Taster und ermöglicht dadurch die Bedienung der Menüoberfläche, ohne das Touch Display zu berühren. **Abbildung 15** zeigt die Erweiterungskarte (Expansion Board). Die Karte besitzt 6 Taster mit den Beschriftungen *Esc*, *Enter*, *Left*, *Right*, *Up* und *Down*. Das Expansion Board erfordert allerdings eine Anschlussmöglichkeit des Touch Displays, um eine Verbindung zwischen diesen Komponenten herstellen zu können. Die Nextion Displays der Enhanced Serie sind hierfür mit 8 universellen Ein- / Ausgangsleitungen (*GPIO*) ausgestattet. Über ein Flexkabel kann somit eine Verbindung zwischen Expansion Board und Touch Display

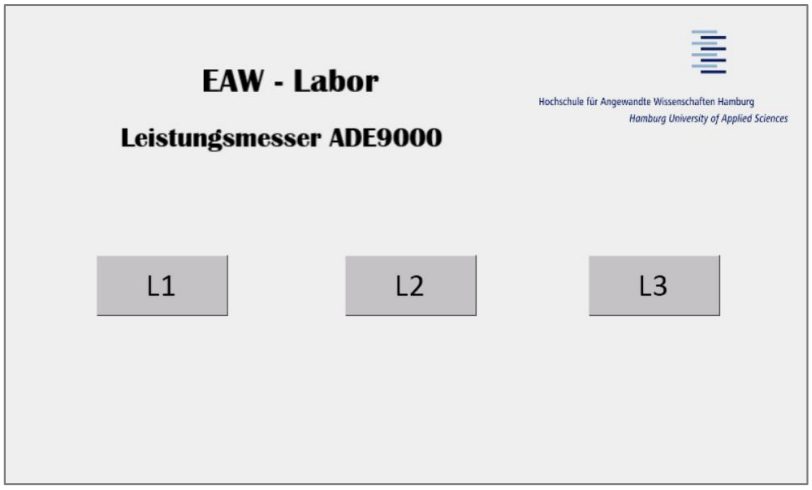


Abbildung 13: *Hauptseite (Variante 2)*

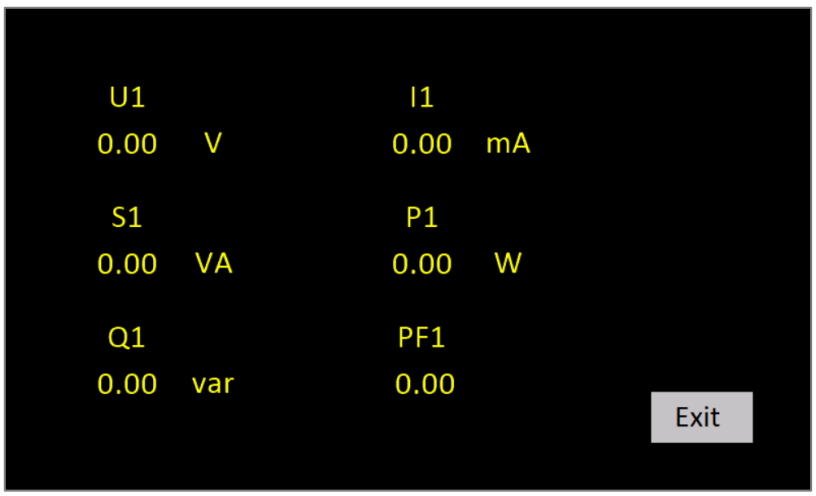


Abbildung 14: *Ansicht Phase L1 (Variante 2)*

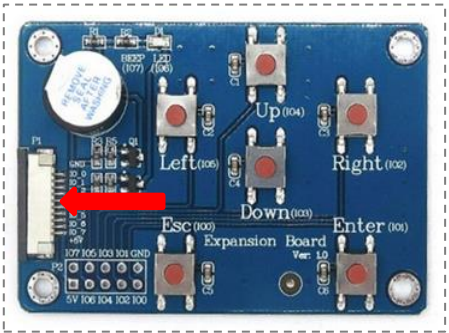


Abbildung 15*: Erweiterungskarte (Expansion Board)* [11]

Abbildung 16: *Flexkabel* [11]

Abbildung 17: *Flexkabel am Nextion Display (Rückseite)*

hergestellt werden. Die Stromversorgung erfolgt dabei über das Flexkabel (**Abbildung 16**). Die Betriebsspannung beträgt 5 V. Die roten Pfeile in den **Abbildungen 15** u. **17** zeigen die Positionen, an denen das Kabel am Expansion Board und Nextion Touch Display angeschlossen wird. Das Expansion Board besitzt außerdem einen Buzzer. Dieser kann bei Bedarf das Drücken eines Tasters durch einen Ton signalisieren. Um die Erweiterungskarte in die Menüführung des Displays integrieren zu können, müssen allerdings beim Erstellen einer Oberfläche mit dem *Nextion Editor* einige Einstellungen vorgenommen werden. Für jeden Button auf einer Oberfläche, der einem bestimmten Taster des Expansion Boards zugeordnet werden soll, muss im *Nextion Editor* ein *Hotspot* erstellt werden. Hierzu muss im *Toolboxbereich* des Editors die Komponente *Hotspot* ausgewählt werden. Der *Hotspot* muss anschließend über dem jeweiligen Button im *Displaybereich* positioniert werden. Der *Hotspot* ist später auf der Oberfläche nicht sichtbar, aber dient der Zuordnung eines Tasters des *Expansion Boards* zu einer bestimmten Seite, die sonst durch Drücken eines Buttons geöffnet wird. Im Fenster *Benutzerereigniscode* des Editors muss hierfür im Textfeld die Anweisung zum Wechseln auf eine bestimmte Seite ergänzt werden. Auf die Codeanweisung zum Wechseln einer Seite wurde im *Abschnitt 4.2* näher eingegangen. Auf der Hauptseite muss im Fenster *Benutzerereigniscode* die Option *Preinitialize* ausgewählt werden und anschließend muss im Textfeld eine Anweisung ergänzt werden. Für die Menüführung der hier beschriebenen Variante 2 steht dort z.B. die folgende Anweisung: *cfgpio 5,1,L3*

Die Zahl *5* gibt die *ID* des zu verwendenden Tasters an. Jeder Taster des *Expansion Boards* ist einer bestimmten *ID* zugeordnet und kann dadurch einer Aktion zugewiesen werden. Die *ID 5* referenziert den Taster *left*. Die *IDs* der Taster sind auf dem *Expansion Bord* neben den Beschriftungen der Taster in Klammern aufgeführt. Die darauffolgende Zahl *1* dient der *STATE* – Konfiguration. Die *1* steht für die Verbindung des *I/O* – Pins mit dem Eingang einer Komponente. Die letzte Angabe *L3* der Anweisung ist der vergebene Name für das Attribut *objname* des erstellten *Hotspots*. Dieser *Hotspot* wurde zuvor über dem Button L1 positioniert. In diesem *Hotspot* verweist die Anweisung *page 1* im Textfenster des Fensters *Benutzerereigniscode* auf die Seite zum Anzeigen der Leistungsdaten der Phase L1. Es wird also nach dem Drücken des Tasters left oder

nach dem Drücken des Buttons L1 die entsprechende Seite geöffnet. Die anderen beiden *Hotspots* sind so konfiguriert, dass nach dem Drücken des Tasters Up bzw. nach dem Drücken des Buttons L2 die entsprechende Seite geöffnet wird und nach dem Drücken des Tasters Right oder des Buttons L3 die Leistungsdaten der Phase L3 angezeigt werden. Der Taster Esc entspricht dem Verhalten des Buttons Exit. Somit kann das Nextion Display auch ohne Nutzung des Touchscreens bedient werden. Auch bei dieser Variante können die Messwerte in einer Textdatei gespeichert werden.

4.3.3 Variante 3

Die Variante 3 des Leistungsmessers setzt sich aus den Komponenten *Board* / *Platine*, *MKR ZERO* Board (Mikrocontroller), *Touch Display* und einem *Drehencoder* zusammen. Auch bei dieser Variante können die verschiedenen Phasen des Leistungsmessers über das Touch Display aufgerufen werden. Das bei dieser Variante verwendete Display ist das Nextion Touch Display, das auch bei der Variante 2 zum Einsatz kommt. Die Menüführung für das Display wurde auch bei dieser Variante mit dem *Nextion Editor* erstellt und anschließend in den Speicher des Displays geladen. Die bei Variante 3 zur Verfügung stehende Menüoberfläche bietet jedoch größere Auswahlmöglichkeiten. Auf der Hauptseite kann zunächst die Auswahl erfolgen, ob im 1 – Phasen – Betrieb oder Mehrphasen – Betrieb gearbeitet werden soll. **Abbildung 18** zeigt die Hauptseite der Oberfläche. Nach dem Drücken des Buttons L1 gelangt man auf die Seite der Messwerte Spannung, Strom, Scheinleistung, Wirkleistung, Blindleistung und Leistungsfaktor der Phase L1 (**Abbildung 19**). Nach dem Drücken des Exit Buttons gelangt man wieder auf die Hauptseite (**Abbildung 18**). Nach Drücken des Buttons L123

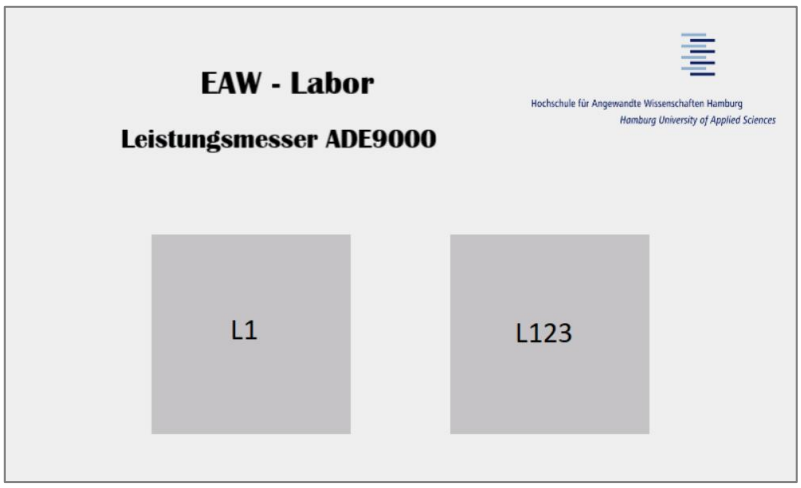


Abbildung 18: *Hauptseite (Variante 3)*

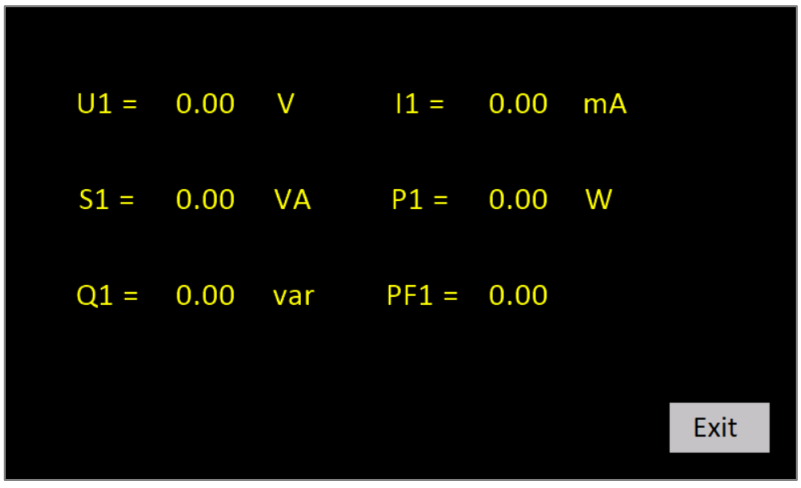


Abbildung 19: *Ansicht Phase L1 (Variante 3)*

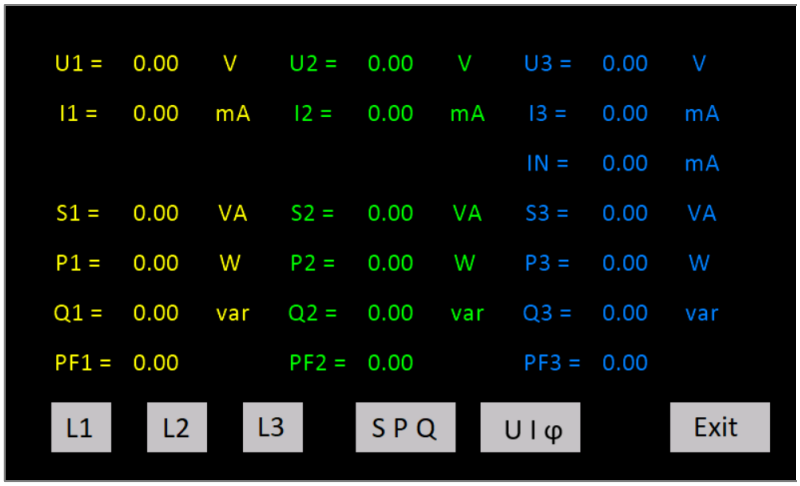


Abbildung 20: *Ansicht Mehrphasen - Betrieb (Variante 3)*

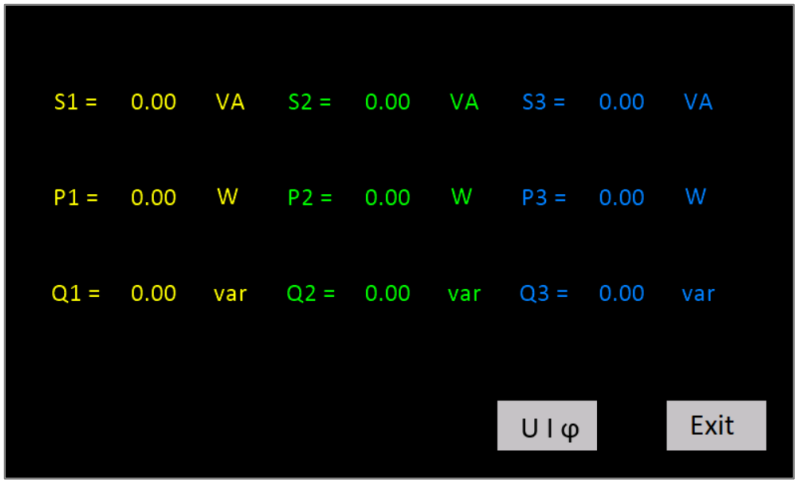


Abbildung 21: *Ansicht Leistungsmesswerte (Variante 3)*

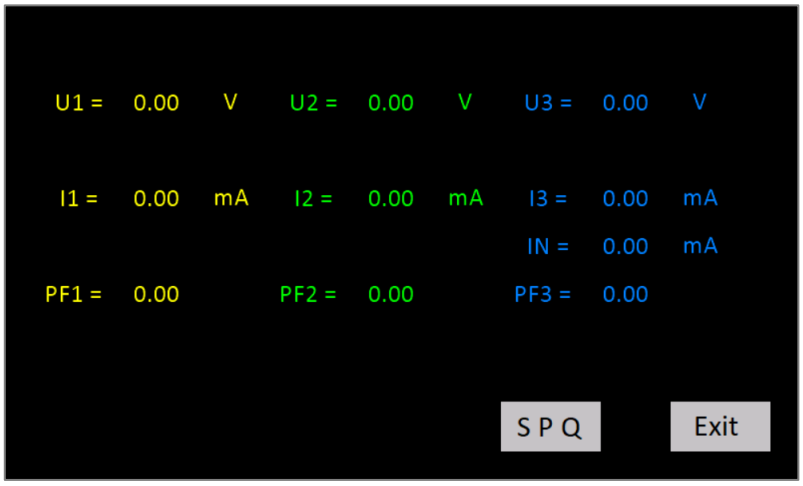


Abbildung 22: *Ansicht der Messwerte Spannung, Strom u. Leistungsfaktor (Variante 3)*

gelangt man auf eine Übersichtsseite für den Mehrphasen – Betrieb mit allen Messwerten der Phasen L1, L2 und L3. Auf dieser Seite besteht zusätzlich die Möglichkeit, die Messwerte der einzelnen Phasen aufzurufen (**Abbildung 20**). Drückt man auf der jeweiligen Seite einer Phasenansicht den Exit Button, gelangt man wieder auf die Übersichtsseite für den Mehrphasen – Betrieb (**Abbildung 20**). Außerdem gelangt man über den Button SPQ auf eine Seite mit den Messwerten Scheinleistung, Wirkleistung und Blindleistung aller Phasen (**Abbildung 21**). Nach dem Drücken des Exit Buttons gelangt man von dort wieder auf die Übersichtsseite für den Mehrphasen – Betrieb (**Abbildung 20**). Über den Button UIφ gelangt man auf eine Seite mit den Messwerten Spannung, Strom und Leistungsfaktor aller Phasen (**Abbildung 22**). Nach dem Drücken des Exit Buttons erreicht man von dort wieder die Ansicht für den Mehrphasen – Betrieb (**Abbildung 20**). Zwischen den Ansichten der Leistungsmesswerte (**Abbildung 21**) und der Messwerte Spannung, Strom und Leistungsfaktor (**Abbildung 22**) kann man

Abbildung 23: *Drehencoder (Rotary Encoder KY – 040)* [12]

nach Drücken des entsprechenden Buttons hin und her wechseln. Von der Ansicht für den Mehrphasen – Betrieb **(Abbildung 20**) erreicht man durch Drücken des Exit Buttons wieder die Hauptseite (**Abbildung 18**). Eine Besonderheit der Variante 3 besteht in einem Drehencoder (**Abbildung 23**), der als eine Alternative für das Touch Display, die Navigation durch die Menüoberfläche ermöglicht. Der Drehencoder ermöglicht durch Drehen und Drücken die Bedienung der Menüoberfläche, ohne das Touch Display zu berühren. Bei einer Drehung des Encoders wird der entsprechende Button der Menüoberfläche in roter Farbe dargestellt, um die aktuelle Position im Menü erkennen zu können. Dies erfolgt durch speziellen Programmcode im Steuerungsprogramm des Leistungsmessers. Wird der Switch des Encoders gedrückt, entspricht die Reaktion dem Drücken eines Buttons bei der Nutzung des Touch Displays. Der Drehencoder *KY – 040* besitzt die fünf Pins CLK (Clock), DT (Data), SW (Switch), + und GND (Ground). Die Spannungsversorgung des Drehencoders erfolgt über den + Pin. Dieser ist mit dem 3,3 V Ausgangspin des MKR ZERO Boards verbunden. Der GND – Pin ist mit dem GND – Eingangspin des MKR ZERO verbunden. Die Pins CLK und DT sind jeweils mit einem digitalen Pin des MKR ZERO verbunden. Der Pin CLK ist mit einem Interrupt – fähigen Pin des MKR ZERO Boards verbunden. Diese Konfiguration ist erforderlich, damit im Falle des Steuerungsprogramms für den Leistungsmesser eine Aktion beim Drehen des Encoders ausgelöst werden kann. Da kontinuierlich Messwerte über den seriellen Port des MKR ZERO an das Display gesendet werden, muss dieses Senden der Werte beim Drehen des Encoders für eine kurze Zeit unterbrochen werden, damit die spezielle Menüführung ermöglicht werden kann. Hierfür wird der Interrupt – fähige Pin benötigt. Mit Hilfe der beiden Ausgänge (CLK u. DT) kann die Drehrichtung des Encoders im Programm ermittelt werden. Bei einer Drehung des Encoders um einen Schritt, wechseln die Ausgänge CLK und DT auf den gegenteiligen Wert (von HIGH auf LOW und von LOW auf HIGH). Durch die Anzahl der HIGH / LOW – Wechsel können die Schritte gezählt werden. Der SW – Pin dient der Auswertung des Signals beim Drücken des Encoders. Alle Encoderausgänge werden mit einem Pull – up – Widerstand betrieben. Die drei Widerstände werden auf der Rückseite des Encoders verlötet. Es können aber auch die vom MKR ZERO Board bereitgestellten internen Widerstände verwendet werden. In diesem Fall müssen keine Widerstände verlötet werden. Wird der Encoder nicht gedrückt, dann liegt am entsprechenden Eingangspin vom MKR ZERO Board ein HIGH – Pegel (logische 1) an, weil ein Stromfluss stattfindet. Beim Drücken des Encoders bewirkt der Pull – up – Widerstand einen LOW – Pegel (logische 0) am entsprechenden Eingangspin des MKR ZERO Boards. Durch das Drücken des Encoderswitches wird ein Kontakt geschlossen und der Strom nimmt den Weg des geringsten Widerstandes über den GND – Pin. Daher findet in dieser Situ-

ation über den SW – Pin kein Stromfluss statt und dieses Verhalten kann im Programm als LOW – Pegel entsprechend ausgewertet werden. Dieses Verhalten erfolgt spiegelbildlich beim Drehen des Encoders. Der Drehencoder kann nach links oder rechts gedreht werden und ermöglicht somit die Navigation nach links oder rechts innerhalb der Menüoberfläche. Eine Besonderheit der Variante 3 besteht außerdem im Programm für die Steuerung des Leistungsmessers. Die loop() – Funktion des Arduino – Programms entspricht der Struktur einer State Machine (Zustandsautomat). Diese Struktur ermöglicht die Aufteilung der Menüoberfläche in verschiedene Zustände (States). Außerdem kann die Menüoberfläche durch ein solches Design übersichtlich im Programm dargestellt werden. Durch das Drücken eines Buttons innerhalb der Menüoberfläche erfolgt der Wechsel in einen anderen Zustand im Programm. Hiermit kann außerdem sichergestellt werden, dass nur der im jeweiligen Zustand definierte Programmcode zur Laufzeit ausgeführt wird. Auch die Variante 3 des Leistungsmessers ermöglicht das Speichern der Messwerte in einer Textdatei.

4.4 Bewertung der Realisierungsvarianten und Auswahl der besten Lösung

Tabelle 4 zeigt eine Übersicht der verschiedenen vorgestellten Realisierungsvarianten mit den Merkmalen, die für die Bewertung und die Auswahl der besten Lösung entscheidend sind.

Realisierungsvarianten (Anzeige und Steuerung)			
Teilfunktion	Variante 1	Variante 2	Variante 3
Anzeigen der Messwerte (Display)	Nextion Display (NX8048T050_011)	Nextion Display (NX8048K070_011R)	Nextion Display (NX8048K070_011R)
Displaygröße	5,0 Zoll	7,0 Zoll	7,0 Zoll
Touch Display	nein	ja	ja
Menüoberfläche	nein	ja	ja
Gesamtansicht Messwerte (Phase L1, L2 u. L3)	ja	nein	ja
Menüführung mit Touch - Funktion	nein	ja	ja
Menüführung (Alternative)	nein	Erweiterungskarte (Expansion Board)	Drehencoder (Rotary Encoder) KY - 040

Tabelle 4: *Übersicht Realisierungsvarianten*

Ein Nachteil der *Variante 1* ist die fehlende Touch – Funktion des Displays. Da für diese Variante keine alternative Steuerungsmöglichkeit vorgesehen ist, kann keine spezielle Menüoberfläche realisiert werden. Da also kein Wechseln zwischen verschiedenen Ansichten erfolgen kann, müssen alle Messwerte auf einer Seite dargestellt werden. Wenn nur im 1 – Phasen – Betrieb gearbeitet wird, kann eine solche Darstellung die Übersichtlichkeit verschlechtern. Die *Variante 2* bietet in diesem Zusammenhang schon mehr Möglichkeiten. Das Display bietet einen größeren sichtbaren Bereich und verfügt über eine Touch – Funktion. Diese Variante ermöglicht daher die Nutzung einer speziellen Menüoberfläche. Diese Oberfläche ist jedoch in der Komplexität begrenzt, weil die als alternative Steuerungsmöglichkeit genutzte Erweiterungskarte hier Grenzen setzt. Eine Anforderung an das Leistungsmessgerät ist die Darstellungsmöglichkeit einer speziellen Ansicht innerhalb der Menüoberfläche für die Messwerte aller Phasen. Die *Variante 2* bietet diese Möglichkeit nicht. Die Phasen L1, L2 und L3 können hier nur jeweils getrennt voneinander dargestellt werden. Die *Variante 3* bietet die meisten Möglichkeiten bezogen auf den Funktionsumfang. Es können verschiedene Ansichten gewählt werden und die alternative Steuerungsmöglichkeit mittels Drehencoder ist komfortabel. Diese Variante bietet auch Flexibilität im Hinblick auf Anpassungen der Menüoberfläche. Die Steuerung des Drehencoders ist unabhängig von der Komplexität der Oberfläche und kann an diese angepasst werden. Die *Variante 3* wurde daher als beste Lösung ausgewählt.

Tabelle 5 zeigt die Auswahl der besten Lösung nach dem *Punktebewertungsverfahren* (VDI 2225).

	Bewertungskriterien				
Realisierungsvarianten (Anzeige und Steuerung)	Display	Menüoberfläche / Funktionsumfang	alternative Menüführung	Punkt-summe	Rang-folge
Variante 1	6	5	0	11	3
Variante 2	10	5	5	20	2
Variante 3	10	10	10	30	1

Tabelle 5: *Punktebewertungsverfahren (Realisierungsvarianten)*

0 … 10 Punkte möglich

4.5 Datenübermittlung zum PC

Für die Übertragung der Messwerte zum PC wird ein Programm verwendet, das mit der Programmiersprache *Processing* erstellt wurde. *Processing* stellt eine integrierte Entwicklungsumgebung zur Verfügung. *Processing* kann kostenlos aus dem Internet heruntergeladen werden und wird in einem quelloffenen Projekt entwickelt. Der grundlegende Aufbau eines Programms, das mit *Processing* entworfen wird, ähnelt einem Arduino – Programm. Allerdings gibt es einen Unterschied. Die loop() – Funktion, die in einem Arduino – Programm den eigentlichen Programmcode enthält, hat in der Programmiersprache *Processing* den Namen draw(). Die Bezeichnung der kontinuierlich auszuführenden Funktion mit dem Namen draw() lässt die eigentlichen Schwerpunkte dieser Programmiersprache erkennen. Diese liegen in den Bereichen Grafik, Simulation und Animation. Aber auch eine Aufgabe wie im vorliegenden Fall, kann mit *Processing* relativ einfach gelöst werden. Ein Vorteil beim Einsatz von *Processing* besteht dabei in der Möglichkeit, den erstellten Programmcode in eine ausführbare Anwendung umwandeln zu können. Zum Aufzeichnen der Messwerte muss dann nur die erstellte Anwendung gestartet werden. Um aus einem Programmcode eine ausführbare Anwendung zu erstellen, muss in *Processing* der Menüpunkt *Datei* ausgewählt werden. Anschließend muss unter der Option *Exportieren* das Betriebssystem ausgewählt werden, für das eine ausführbare Anwendung erstellt werden soll. *Processing* bietet die Möglichkeit, ausführbare Anwendungen für die Plattformen *Windows*, *Mac OS X* und *Linux* zu erstellen. Die Arduino – Entwicklungsumgebung bietet nicht eine solche Möglichkeit. Der Schwerpunkt liegt hierbei in der Erstellung von Programmen für die Steuerung von Mikrocontrollern. Das Programm für das Speichern der Messwerte wurde so entworfen, dass die Messwerte fortlaufend in einer Textdatei gespeichert werden. Nach dem Beenden von Messungen bzw. nach dem Schließen der Anwendung für das Aufzeichnen der Messdaten, kann die Textdatei eingesehen und weiterverarbeitet werden. Das Arduino – Programm für die Steuerung des Leistungsmessers enthält allerdings speziellen Code, mit dem die Speicherung der Messwerte über die externe Anwendung ermöglicht wird. Die Messwerte der einzelnen Phasen des Leistungsmessers werden über das Arduino – Programm kontinuierlich in einer Variablen vom Typ String gespeichert. Durch den zyklischen Programmablauf wird der Inhalt der Variablen kontinuierlich aktualisiert. Die mit *Processing* erstellte Anwendung für das Speichern der Messwerte liest fortlaufend den Datenstrom mit, der sich bei der Ausführung des Arduino – Programms auf dem seriellen Port des Mikrocontrollers ergibt. Hiermit kann der im Arduino – Programm erstellte String mit den Messwerten mitgelesen und in die Textdatei geschrieben werden. Für die Aufzeichnung der Messwerte muss der Mikrocontroller des Leistungsmessers über die USB – Schnittstelle mit dem PC verbunden werden. Dann kann die mit *Processing* erstellte Anwendung den Datenstrom mitlesen. Das *Processing* Programm zur Speicherung der Messdaten und die erforderlichen Anpassungen im Arduino – Programm werden im *Kapitel 6* (Datenübertragung zum PC) näher erläutert.

5 Realisierung des gewählten Lösungsansatzes

Im folgenden Kapitel werden die einzelnen Komponenten der ausgewählten Lösung (*Variante 3*) und das Arduino – Programm für die Steuerung des Leistungsmessers näher betrachtet. Im letzten Abschnitt wird auf die Kalibrierung und das Testen des Messgerätes eingegangen.

5.1 Leistungsmesser mit Touch Display und Drehencoder (Displaysteuerung)

Die Hauptkomponente des Leistungsmessers ist das Board bzw. die Platine, welche auf einem EVAL – ADE9000 – Shield basiert. Es handelt sich dabei um ein Evaluationskit des Halbleiterherstellers Analog Devices. Wie schon in der Einleitung angesprochen, wurde auf Basis des Evaluationskits ein neues Layout für die Platine entworfen, um das MKR ZERO Board für die Steuerung des Leistungsmessers verwenden zu können. Der Sockel musste für die Verwendung des MKR ZERO angepasst werden, weil dieses Board andere Abmessungen als das Arduino ZERO besitzt, welches ursprünglich zum Einsatz kommen sollte. Die Hauptbestandteile des ADE9000 Boards wurden aber nicht verändert. Dieses Board enthält 7 unabhängige Hochleistungs – Analog / Digital – Wandler (ADCs) mit einer Größe von jeweils 24 Bit. Die Daten der ADCs werden jeweils als 32 Bit Wert gespeichert. Integriert in das Bord ist ein digitaler Signalprozessorkern, der ADE9000. Dieser Prozessor bildet das Kernstück des Boards. Das Board besitzt 3 Phaseneingänge (L1, L2 u. L3). An diese Eingänge können die zu messenden Eingangsspannungen angeschlossen werden. Das Board besitzt drei Ausgänge für die Phasen L1, L2 u. L3. An diese Ausgänge können Lasten angeschlossen werden, für welche die entsprechenden Strom u. Leistungswerte ermittelt werden sollen. Das ADE9000 Board wird mit einem Steckernetzteil mit einer Spannung von 5 V versorgt. Diese Versorgungsspannung wird für den Betrieb des MKR ZERO benötigt. Der Signalprozessor des Boards und die weiteren beteiligten Komponenten benötigen eine Betriebsspannung von 3,3 V. Das MKR ZERO Board stellt diese Spannung über den VCC (3,3 V) – Pin zur Verfügung. Das ADE9000 Board besitzt Stromwandler, um die Stromstärke der jeweiligen Phase messen zu können. Es werden hierbei Stromwandler vom Typ *ZMCT103C* verwendet. Diese Stromwandler haben ein Windungszahlverhältnis von 1000:1. Der maximal empfohlene Eingangsstrom beträgt bei diesen Wandlern 5 A. Um die Messgenauigkeit zu erhöhen, wenn eine kleine Eingangsspannung an den Phaseneingängen anliegt, verfügt jede Phase des Boards über eine Filterschaltung, die über ein Relais im Steuerungsprogramm

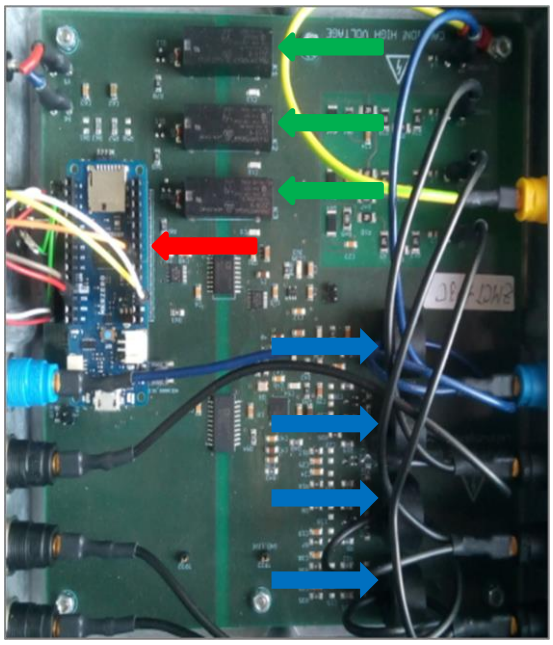


Abbildung 24: *ADE9000 Board mit MKR ZERO*

aktiviert werden kann. Damit können Messungen bei einer Eingangsspannung von 23 V (Wechselstrom) erfolgen. Ansonsten ist die Messgenauigkeit für eine Eingangsspannung von 230 V (Wechselstrom) ausgelegt. Diese Option wurde bei der Entwicklung des neuen Layouts für das Board hinzugefügt, um spezielle Messungen im Labor für elektrische Anlagen und Wandler durchführen zu können. **Abbildung 24** zeigt das ADE9000 Board. Der rote Pfeil zeigt das MKR ZERO Board in eingebauter Position. Die grünen Pfeile zeigen auf die Relais der Phasen L1, L2 u. L3, mit denen die Messgenauigkeit an den Spannungsbereich angepasst werden kann. Die blauen Pfeile zeigen auf die vier Stromwandler zum Messen der Stromstärke der jeweiligen Phase. Es kommen vier Stromwandler zum Einsatz, weil auch die Stromstärke des Neutralleiters gemessen werden kann. **Abbildung 25** zeigt die Frontseite des Leistungsmessergehäuses. Der rote Pfeil zeigt auf den Anschluss für das Steckernetzteil zur Stromversorgung des ADE9000 Boards bzw. des MKR ZERO. Der grüne Pfeil zeigt auf den Anschluss für das Nextion Touch Display. Dieser Anschluss kombiniert die Leitungen + (5 V), - (GND), TX (Transmit) und RX (Receive). Das Touch Display wird also über diesen Anschluss mit Strom versorgt. Die Datenleitungen TX und RX dienen der Kommunikation zwischen dem Mikrocontroller und dem Touch Display. Die blauen Pfeile markieren die Eingänge der Phasen L1, L2, L3 und den Anschluss für den Neutralleiter. **Abbildung 26** zeigt die Rückseite des Leistungsmessergehäuses. Die blauen Pfeile markieren die Anschlüsse der Phasen L1, L2 u. L3, für welche eine Messung einer angeschlossenen Last erfolgen soll. Außerdem befindet sich auf der Rückseite der Anschluss für den Neutralleiter. Der grüne Pfeil zeigt auf den PE – Anschluss. Da es sich bei dem Leistungsmessergehäuse um ein Metallgehäuse handelt, muss durch diese Schutzerdung verhindert werden, dass im Falle der Berührung eines spannungsführenden Leiters mit dem Gehäuse (z.B. bei einem Defekt) ein Stromfluss durch den menschlichen Körper stattfinden kann oder dieser Stromfluss zumindest minimiert wird.

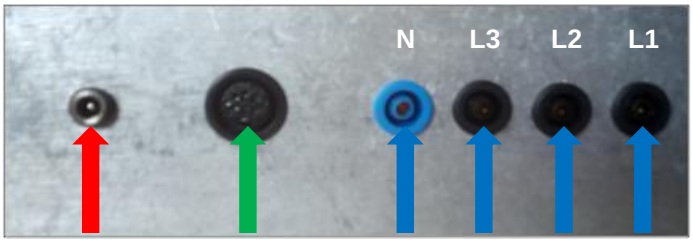


Abbildung 25: *Frontseite des Leistungsmessergehäuses*

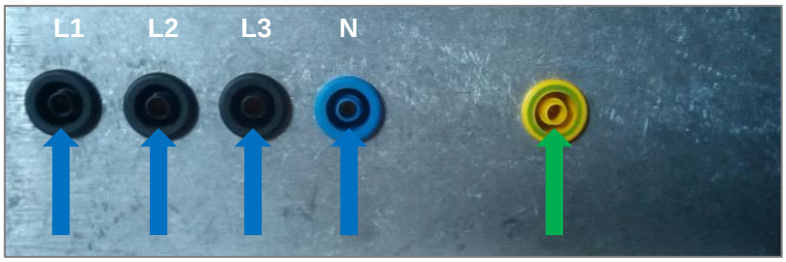


Abbildung 26: *Rückseite des Leistungsmessergehäuses*

Abbildung 27: *MKR ZERO* [12]

Für die Steuerung des Leistungsmessers kommt das Arduino Board *MKR ZERO* (**Abbildung 27**) zum Einsatz. Dieses Board verfügt über die gleiche Leistung wie das Arduino ZERO, ist jedoch in seinen Abmessungen deutlich kleiner. Auch das *MKR ZERO* unterstützt 32 Bit Anwendungen. Das Board bietet außerdem Multimedia – Eigenschaften. Ein integrierter SD – Stecker mit dedizierter SPI – Schnittstelle ermöglicht das Abspielen von Musikdateien ohne den Einsatz von zusätzlicher Hardware. Dieses Feature spielt allerdings bei der Verwendung des Leistungsmessers keine Rolle. Das Board *MKR ZERO* besitzt 22 digitale I / O – Pins. 12 Pins unterstützen das PWM – Verfahren. Das *MKR ZERO* bietet 7 Analogeingänge und einen analogen Ausgang. Die Stromversorgung kann über die Micro – USB – Schnittstelle oder den VIN – Pin erfolgen. Bei Verwendung des VIN – Pins kann die Versorgungsspannung 5 V bis maximal 6 V betragen. In diesem Fall wird die Stromversorgung über die USB – Schnittstelle deaktiviert. Es besteht jedoch ein Unterschied im Vergleich zu den meisten Arduino – Boards. Die maximale Spannung an den I / O – Pins darf nicht größer als 3,3 V sein. Die meisten Arduino Boards unterstützen eine Spannung von 5 V an den Ein– und Ausgängen. Bei der Stromversorgung über die integrierte USB – Schnittstelle oder bei Nutzung des VIN – Pins wird automatisch eine Spannung von 3,3 V an den Ausgängen zur Verfügung gestellt. Bei der Nutzung eines Pins als ein Eingang muss daher beachtet werden, dass die am jeweiligen Pin anliegende Spannung nicht größer als 3,3 V ist. Bei Überschreiten dieser Spannung besteht sonst die Gefahr, dass das *MKR ZERO* Board beschädigt wird. Der 5 V – Pin des *MKR ZERO* liefert eine Ausgangsspannung von 5 V bei Nutzung der USB – Schnittstelle oder des VIN – Pins für die Stromversorgung. Beim Leistungsmesser wird über den VIN – Pin die Stromversorgung des *MKR ZERO* über das Steckernetzteil sicher-

Microcontroller	SAMD21 Cortex-M0+ 32bit low power ARM MCU
Board Power Supply (USB/VIN)	5V
Supported Battery(*)	Li-Po single cell, 3.7V, 700mAh minimum
DC Current for 3.3V Pin	600mA
DC Current for 5V Pin	600mA
Circuit Operating Voltage	3.3V
Digital I/O Pins	22
PWM Pins	12 (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, A3 - or 18 -, A4 -or 19)
UART	1
SPI	1
I2C	1
Analog Input Pins	7 (ADC 8/10/12 bit)
Analog Output Pins	1 (DAC 10 bit)
External Interrupts	8 (0, 1, 4, 5, 6, 7, 8, A1 -or 16-, A2 - or 17)
DC Current per I/O Pin	7 mA
Flash Memory	256 KB
Flash Memory for Bootloader	8 KB
SRAM	32 KB
EEPROM	no
Clock Speed	32.768 kHz (RTC), 48 MHz

Tabelle 6: *Technische Daten (MKR ZERO)* [12]

gestellt. Der VCC – Pin des *MKR ZERO* liefert eine Ausgangsspannung von 3,3 V über den integrierten Spannungsregler. Diese Spannung steht an diesem Pin unabhängig von der gewählten Stromquelle (USB, VIN oder Akku) zur Verfügung. Eine Besonderheit des *MKR ZERO* besteht in der Anschlussmöglichkeit eines einzelligen Lipo – Akkus. Der Lipo – Akku kann über den integrierten Anschluss aufgeladen werden. Allerdings sollte der Akku eine minimale Kapazität von 700 mAh haben. Ein spezieller Chip des *MKR ZERO* regelt das Ladeverhalten. Ein angeschlossener Lipo – Akku wird mit einem Ladestrom von 350 mAh geladen. Der integrierte Chip unterstützt eine Ladezeit von bis zu 4 Stunden. Danach geht er automatisch in den sleep mode. Daher können pro Ladezyklus maximal 1400 mAh geladen werden. Bei Nutzung eines Lipo – Akkus beträgt die Ausgangsspannung am 5 V – Pin ca. 3,7 V. Für den Betrieb des Leistungsmessers wird die Akku Funktion allerdings nicht genutzt, weil in diesem Fall kein Akku zum Einsatz kommt. Das *MKR ZERO* Board besitzt 8 Pins, die als Interrupt – Pins genutzt werden können. Außerdem besteht die Möglichkeit, Hardware über die integrierte SPI- und I2C – Schnittstelle zu nutzen. **Abbildung 28** zeigt das Pinout – Diagramm des *MKR ZERO*.

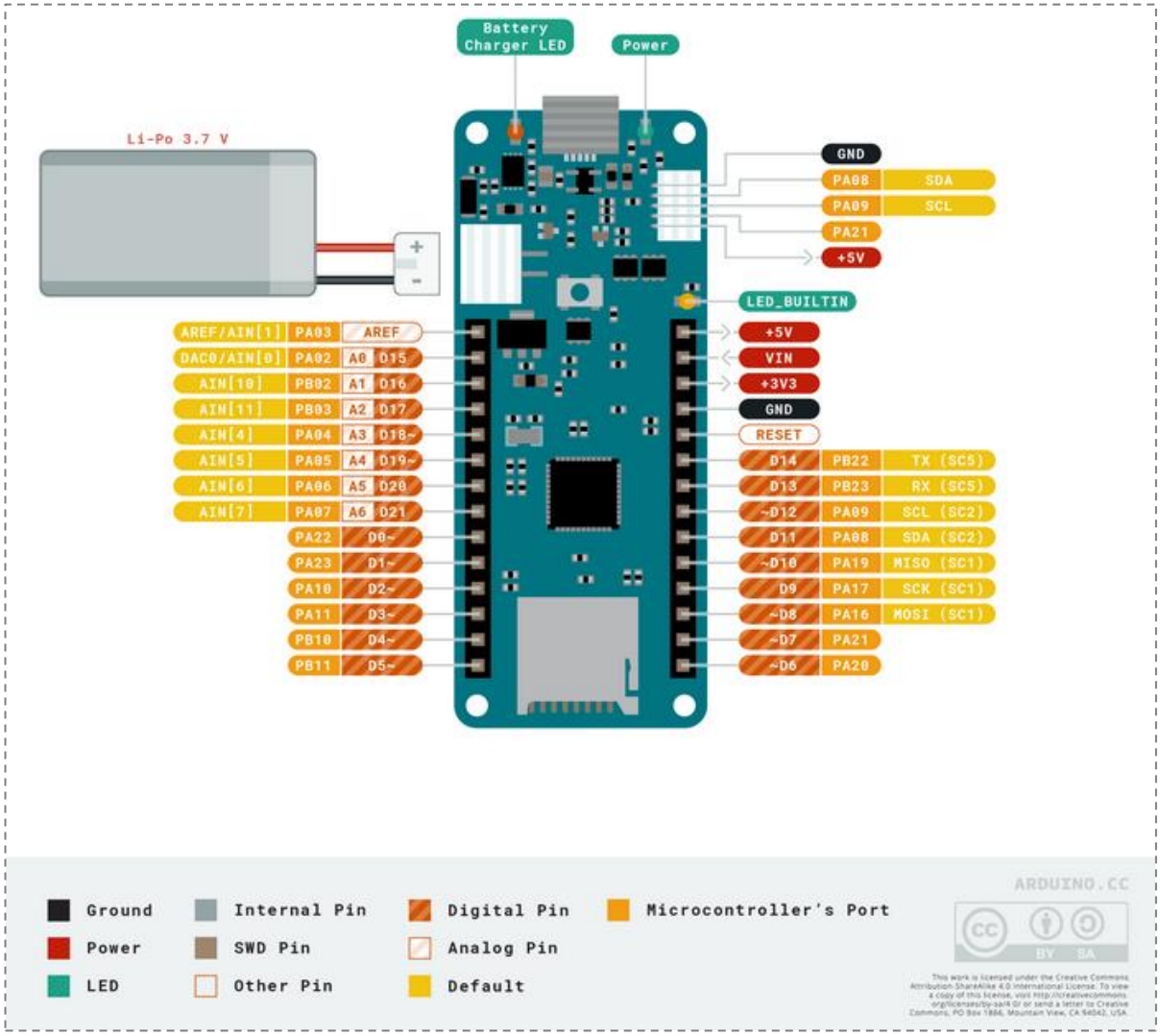


Abbildung 28: *Pinout – Diagramm MKR ZERO* [12]

Tabelle 7 zeigt die Pinbelegung des MKR ZERO für den Betrieb des Leistungsmessers.

Pinbelegung (MKR ZERO)		
Pin	Funktion / Pin	Bedeutung
VIN	Stromversorgung (MKR ZERO)	Betrieb des MKR ZERO wird ermöglicht
VCC	Stromversorgung (ADE9000 Board u. Rotary Encoder)	Betrieb des ADE9000 Boards u. des Rotary Encoders wird ermöglicht
D1	Schalten von Relais 1	Messgenauigkeit (Phase L1)
D2	Schalten von Relais 2	Messgenauigkeit (Phase L2)
D3	Schalten von Relais 3	Messgenauigkeit (Phase L3)
D4	Power Mode PM1 Pin	Einstellen des Betriebsmodus (ADE9000)
D5	Hardware Reset (ADE9000)	Kalibrierung (ADE9000 Board)
D6	CS (SPI - Schnittstelle)	Datenaustausch zwischen ADE9000 und MKR ZERO
D8	MOSI (SPI - Schnittstelle)	
D9	SCK (SPI - Schnittstelle)	
D10	MISO (SPI - Schnittstelle)	
D7	I2C - Schnittstelle	Hilfspin bei Erweiterung der Schnittstelle
D11	SDA (I2C - Schnittstelle)	Zugriff (EEPROM ADE9000 Board)
D12	SCL (I2C - Schnittstelle)	
D13	Kommunikation mit dem Touch Display (RX - Pin) (serieller Port)	Daten empfangen
D14	Kommunikation mit dem Touch Display (TX - Pin) (serieller Port)	Daten senden
D16	Switch - Pin (SW) (Rotary Encoder)	Signalübertragung
D17	Pin A (CLK) (Rotary Encoder)	Signalübertragung
D20	Pin B (DATA) (Rotary Encoder)	Signalübertragung

Tabelle 7: *Pinbelegung (MKR ZERO)*

Im Folgenden wird noch einmal auf die in **Tabelle 7** aufgeführte Pinbelegung des MKR ZERO eingegangen. Der VIN Eingang des MKR ZERO dient der Stromversorgung des Boards. Dieser Eingang ist mit dem Anschluss für das Steckernetzteil (5 V) verbunden. Der VCC (3,3 V) Ausgang am MKR ZERO liefert eine Ausgangsspannung von 3,3 V und dient der Stromversorgung für das ADE9000 Board und dem Drehencoder. Die Pins 1, 2 und 3 sind als Ausgänge konfiguriert und ermöglichen das Schalten des jeweiligen Relais der 3 Phasen über das Steuerungsprogramm. Hiermit kann die Messgenauigkeit an den Spannungsbereich angepasst werden. Der Pin 4 ist als Ausgang konfiguriert und wird für die Einstellung des Betriebsmodus des ADE9000 Boards benötigt. Das ADE9000 Board stellt für die Wahl des Betriebsmodus die Pins PM1 u. PM0 zur Verfügung. Das MKR ZERO Board ist über den Pin 4 mit dem Pin PM1 des ADE9000 Boards verbunden. Zum Pin PM0 des ADE9000 Boards besteht keine Verbindung. Diese Verbindung ist nicht erforderlich, weil beim Schreiben eines Low – Pegels auf den Pin PM1 unabhängig vom Pin PM0 der Betriebsmodus *Normal mode* mit allen zur Verfügung stehenden Funktionen des ADE9000 eingestellt wird. Diese Einstellung wird für den Betrieb des Leistungsmessers benötigt. Der Pin PM0 des ADE9000 Boards muss daher nicht über das Programm angesprochen werden. Wenn man zusätzlich den Pin PM0 mit dem MKR ZERO Board verbinden würde und auf beide Eingänge (PM0 u. PM1) des ADE9000 Boards einen High – Pegel schreiben würde, dann könnte man das ADE9000 Board hiermit in den Betriebsmodus *idle*, also *Leerlauf*, versetzen. Dann stehen aber keine Funktionen für den Betrieb zur Verfügung. Der Pin 5 ist als Ausgang konfiguriert und kann zum Durchführen eines Hardware Reset für das ADE9000 Board über das Steuerungsprogramm genutzt werden. Die Pins 6, 8, 9 und 10 stellen die Verbindung zwischen ADE9000 Board und MKR ZERO über die SPI – Schnittstelle her. Über diese Schnittstelle erfolgt der Zugriff auf die Register des ADE9000 Prozessors für die Berechnung der Messwerte im Steuerungsprogramm. Über die Pins 7, 11 und 12 erfolgt die Konfiguration der I2C – Schnittstelle des MKR ZERO. Hierüber kann auf das EEPROM des ADE9000 Boards zugegriffen werden. Diese Verbindung wird jedoch nicht für den Betrieb des Leistungsmessers benötigt und wurde als Option hinzugefügt. Die Pins 13 und 14 ermöglichen die Kommunikation zwischen dem MKR ZERO Board und dem Touch Display über den seriellen Port des MKR ZERO. Der Pin 13 (RX – Pin) wird für das Empfangen von Daten des Touch Displays benötigt. Wird ein Button der Menüoberfläche auf dem Touch Display gedrückt, dann muss dieses Verhalten vom MKR ZERO bzw. vom Steuerungsprogramm entsprechend ausgewertet werden. Der Pin 14 (TX – Pin) wird für das Senden von Daten an das Display benötigt. Hiermit wird die Darstellung der Messwerte auf dem Touch Display ermöglicht und es werden hierüber die Befehle zum Aufrufen der verschiedenen Seiten der Oberfläche an das Display gesendet. Die Pins 16, 17 und 20 sind als Eingänge konfiguriert und dienen der Auswertung des Drehencoders über das MKR ZERO Board bzw. das Steuerungsprogramm für den Leistungsmesser. Das Drücken und Drehen am Encoder muss vom Programm entsprechend ausgewertet werden, um die Navigation mit dem Rotary Encoder zu

ermöglichen.

Für die Darstellung der grafischen Oberfläche des Leistungsmessers kommt ein Nextion Touch Display zum Einsatz, das im *Abschnitt 4.2* (Entwicklungsumgebung für die Programmierung des Touch Displays) näher beschrieben wurde. Im Folgenden wird als Ergänzung zum *Abschnitt 4.2* die mit dem Nextion Editor erstellte Menüoberfläche für das Touch Display näher betrachtet. Für diese spezielle Oberfläche müssen im Nextion Editor alle Objekte erstellt werden, die später auf dem Display dargestellt werden sollen. Außerdem wird mit Hilfe der einzelnen Elemente und Einstellungen die Kommunikation zwischen Steuerungsprogramm und Touch Display ermöglicht. Hiermit kann der Mikrocontroller über das Programm die entsprechenden Daten zum Touch Display senden. Auch in der anderen Richtung muss eine Referenzierung erfolgen. Das Drücken eines Buttons innerhalb der Menüoberfläche muss vom Steuerungsprogramm ausgewertet werden können, um eine entsprechende Aktion auszulösen. **Abbildung 29** zeigt die Hauptseite der Menüoberfläche im Nextion Editor. Am rechten Rand des Bildausschnitts werden die Ansichten für den *Seitenbereich* (Page) und den *Attributbereich* (Attribute) angezeigt. Der Button L1 wird im *Displaybereich* blau umrandet dargestellt, weil dieser mit der Maus markiert wurde.

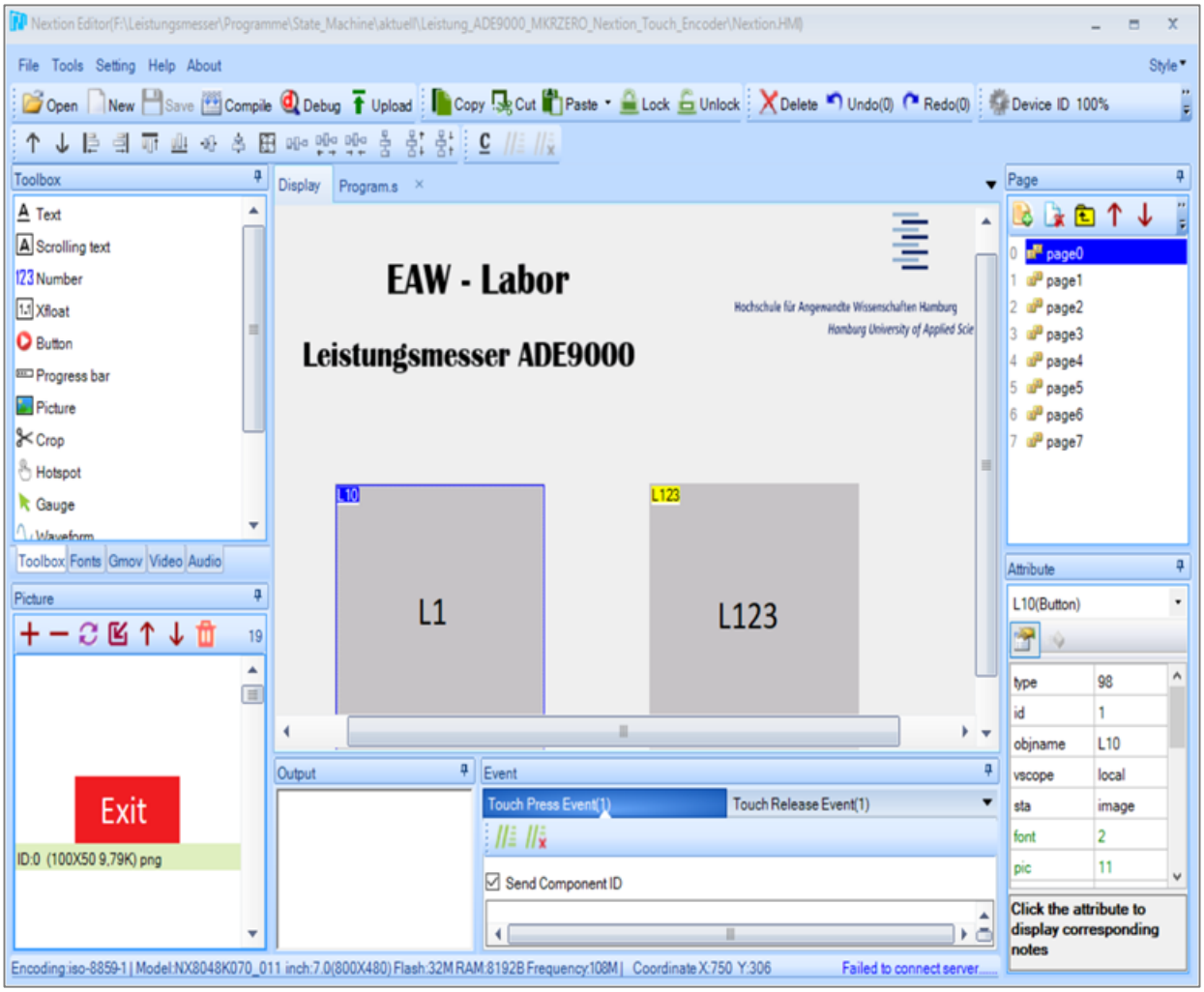


Abbildung 29: *Hauptseite der Menüoberfläche im Nextion Editor*

Abbildung 30 zeigt vergrößert den *Seitenbereich* und *Attributbereich*. Nach dem Auswählen einer *Komponente* aus dem *Toolboxbereich* des Nextion Editors werden die zugehörigen Attribute im *Attributbereich* angezeigt. Für den markierten Button L1 werden die zugehörigen Attribute im *Attributbereich* angezeigt, wie in **Abbildung 30** dargestellt. Auf die Vorgehensweise für das Erstellen einer Oberfläche wurde im *Abschnitt 4.2* (Entwicklungsumgebung für die Programmierung des Touch Displays) näher eingegangen. Für die Referenzierung im Arduino Programm zur Steuerung des Leistungsmessers sind die Attribute *id* und *objname* von großer Bedeutung. Der Wert für das Attribut *id* wird beim Erstellen einer Komponente automatisch vom Nextion Editer vergeben und kann nicht geändert werden. Der Name für das Attribut *objname* kann selber vergeben werden. Dazu muss mit der Maus das Feld rechts von *objname* markiert werden. In diesem Fall wurde der Name L10 vergeben. Im Fenster für den *Seitenbereich* wird durch eine blaue Markierung die Seite hervorgehoben, auf der sich die aktuell ausgewählte Komponente befindet. In diesem Fall ist es die Seite 0 (page0). Der Button L1 befindet sich also auf Seite 0 der Menüoberfläche. Die in **Abbildung 29** dargestellte Hauptseite hat daher den Index 0. Dieser Index muss zusammen mit den Inhalten der Attribute *objname* und *id* im Steuerungsprogramm aufgeführt werden, um eine Referenzierung zwischen Touch Display und Steuerungsprogramm herstellen zu können. Im Programm muss der Button L1 in folgender Weise deklariert werden: *NexButton L10 = NexButton(0, 1, "L10");*

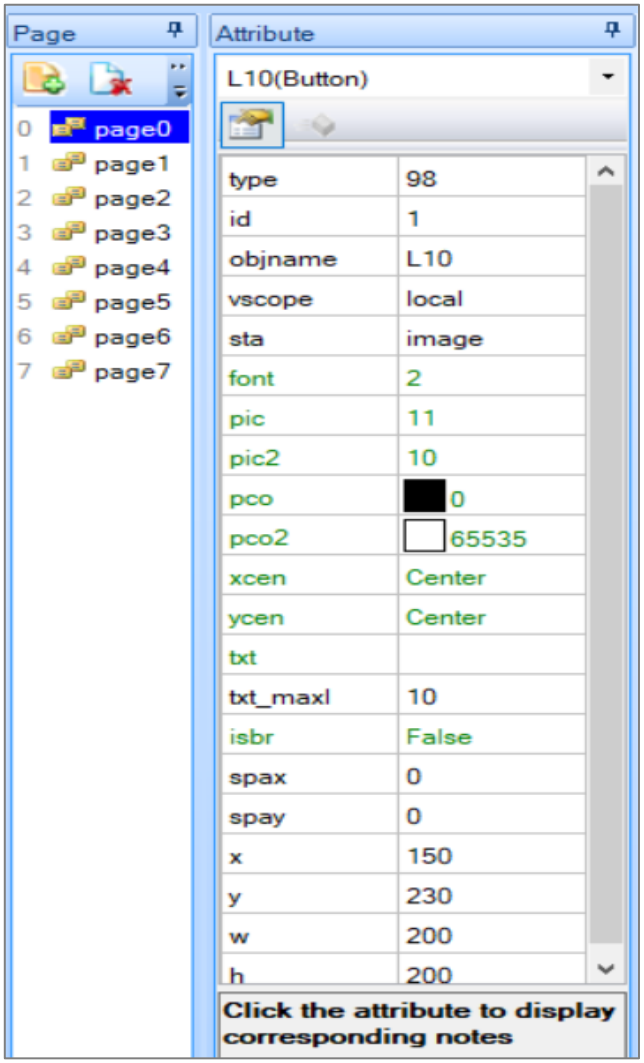


Abbildung 30: *Bildausschnitt (Seitenbereich u. Attributbereich im Nextion Editor)*

Die Angabe *NexButton* ist erforderlich, weil es sich bei dem Objekt um einen Button handelt. Die Bezeichnung *L10* ist der vergebene Variablenname im Programm, über welchen der Button referenziert werden kann. Die Zahl *0* des Klammerausdrucks gibt die Seite an, auf welcher sich der Button befindet. Die Zahl *1* ist der Wert des Attributs *id*. Die Angabe *"L10"* des Klammerausdrucks ist der vergebene Name für das Attribut *objname*, wie in **Abbildung 30** dargestellt. Diese Angaben müssen für jeden Button im Steuerungsprogramm vorgenommen werden, wenn nach dem

Drücken eines Buttons im Programm eine bestimmte Aktion ausgelöst werden soll. Um die Kommunikation zwischen einem Button und dem Steuerungsprogramm zu ermöglichen, muss außerdem eine weitere Einstellung im Nextion Editor vorgenommen werden. In der Ansicht *Benutzerereigniscode* (Event) muss der Reiter *Touch Press Event(1)* mit der Maus markiert werden. Jetzt muss die Option *Send Component ID* markiert werden. Mit Hilfe dieser Option wird beim Drücken eines Buttons die zugehörige ID an das Programm übertragen und es kann entsprechend eine Aktion ausgeführt werden. **Abbildung 31** zeigt den unteren Bildausschnitt des Nextion Editors. Diese *Event* – Ansicht erscheint am unteren Rand im Nextion Editor, nachdem ein erstellter Button mit der Maus markiert wurde.

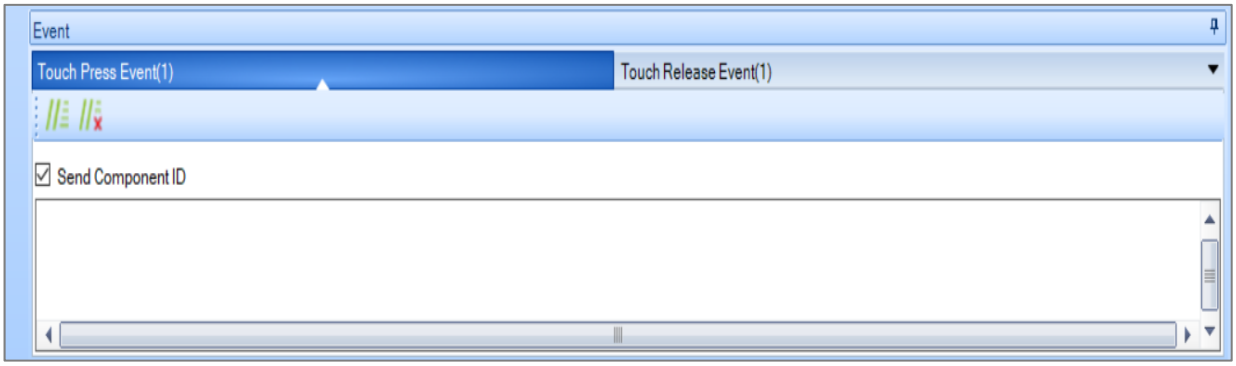


Abbildung 31: *Bildausschnitt (Touch Press Event(1) im Nextion Editor)*

Es wäre auch möglich, den Reiter *Touch Release Event(1)* auszuwählen. Nach Auswahl dieses Reiters erscheint auch die Option *Send Component ID* und kann markiert werden. Das damit verbundene Verhalten beim Drücken eines Buttons ist dann aber anders und wird so in den meisten Fällen nicht gewünscht. Die ID des entsprechenden Buttons wird in diesem Fall erst nach Loslassen des Buttons an das Programm gesendet und würde eine Verzögerung der auszuführenden Aktion des Programms bewirken. Um eine Verbindung zwischen Button und Programm herzustellen, müssen im Steuerungsprogramm für den Leistungsmesser weitere Definitionen erfolgen. Außerdem muss eine spezielle Bibliothek (Library) in das Programm eingebunden werden, um die erforderlichen Definitionen für das Nextion Touch Display verwenden zu können bzw. damit die Hardware (Display) über das Programm angesprochen werden kann. Die Variablennamen für die Buttons der Menüoberfläche werden in einem speziellen Array gespeichert und über Pointer (Zeiger) erfolgt die Referenzierung auf die Buttons. Die verwendete Bibliothek stellt diese Funktionen zur Verfügung. Für jeden Button erfolgt die Definition einer speziellen Funktion im Programm. Für den Button L1 ist die folgende Codeangabe im Programm erforderlich:

```
void L10_Press(void *ptr)
{
  vL10Button = true;
}
```

Die Anweisung *Press* gibt an, dass durch Drücken des Buttons eine Aktion im Programm ausgelöst werden kann. Beim Drücken des Buttons soll der in geschweiften Klammern aufgeführte Code ausgeführt werden. In diesem Fall wird der im Programm erstellten *Variablen* mit dem Namen *vL10Button* der Wert *true* zugewiesen. Hiermit kann im Programm erkannt werden, ob der entsprechende Button, in diesem Fall L1, gedrückt wurde. Für alle weiteren Buttons der Menüoberfläche muss spiegelbildlich im Steuerungsprogramm die oben aufgeführte Funktion erstellt werden, damit über den jeweiligen Button eine Aktion ausgelöst werden kann. Die Verbindung des Buttons L1 der Menüoberfläche und des Steuerungsprogramms wird über die folgende Funktion im Programm hergestellt: *L10.attachPush(L10_Press, &L10);*
Die Funktion *attachPush()* kann durch das Einbinden der speziellen Bibliothek genutzt werden. Die Bezeichnung *L10_Press* des Klammerausdrucks bezieht sich auf die zuvor genannte Funktion, in der die auszuführende Aktion beim Drücken des Buttons L1 festgelegt wird und über die Angabe *&L10*, referenziert ein Pointer (Zeiger) den erstellten Button über die Variable L10. Für alle Buttons muss spiegelbildlich die *attachPush()* – Funktion im Programm erstellt werden. Im Nextion Editor kann das Verhalten des Touch Displays auch nur über die mit dem Editor erstellte Oberfläche gesteuert werden. Um z.B. nach dem Drücken des Buttons L1 auf die Seite 1 der Menüoberfläche zu wechseln, müsste in der Ansicht *Benutzerereigniscode* (Event) der Reiter *Touch Release Event(1)* ausgewählt werden. Dort müsste dann der Befehl *page 1* zum Wechseln der Seite ergänzt werden. **Abbildung 32** zeigt den Bildausschnitt der Ansicht *Benutzerereigniscode* (Event) für diese Situation. Wenn das Drücken eines Buttons also nicht vom Arduino Programm ausgewertet werden soll, dann reicht diese Anweisung für das Wechseln auf die entspre-

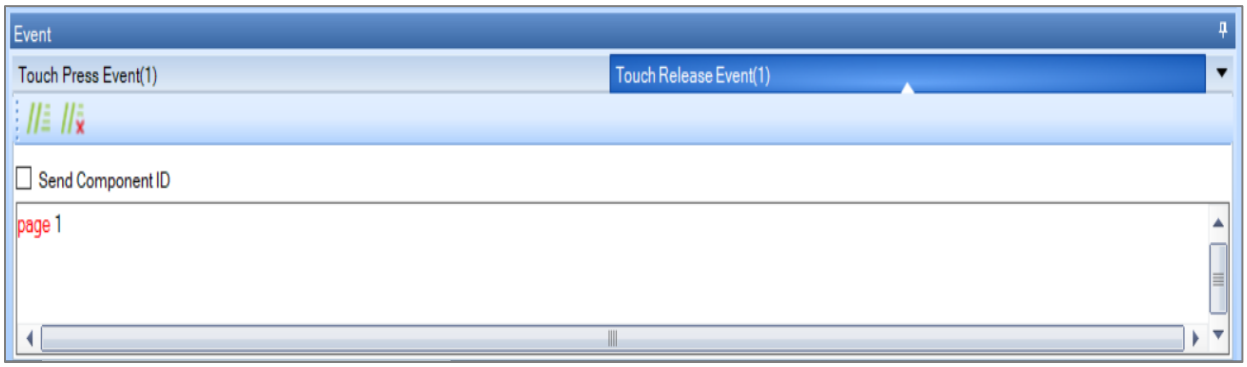


Abbildung 32: *Bildausschnitt (Touch Release Event(1) im Nextion Editor)*

chende Seite. Es muss dann nicht die Option *Send Component ID* markiert werden. Das Arduino Programm für die Steuerung des Leistungsmessers entspricht aber vom Aufbau einem Zustandsautomaten. Daher muss in diesem Fall das Drücken eines Buttons vom Programm ausgewertet werden, um innerhalb des Programms entsprechend in einen anderen Zustand zu wechseln. Für jeden Button kann im Nextion Editor ein spezielles Layout erstellt werden. Über das Attribut *txt* kann die Beschriftung für einen Button festgelegt werden. Dazu muss mit der Maus das Feld rechts vom Eintrag *txt* im *Attributbereich* markiert werden. Danach kann eine Bezeichnung einge-

geben werden, welche danach auf der Oberfläche des jeweiligen Buttons angezeigt wird. Es besteht aber auch die Möglichkeit, für einen Button eine Bilddatei zu erzeugen. Dieses Image kann z.B. als PNG – Datei gespeichert werden und für die Darstellung des Buttons verwendet werden. Bei der grafischen Erstellung des Layouts für einen Button muss allerdings beachtet werden, dass dieser die gleichen Abmessungen wie der im Nextion Editor definierte Bereich hat. Über die Attribute *w* und *h* (Breite u. Höhe) im *Attributbereich* wird die Größe festgelegt, die der Button auf der Seite einnimmt. Die Größe des erstellten Bildes in Pixeln muss also mit den Werten der Attribute *w* und *h* übereinstimmen. Nach dem Erstellen der Images müssen diese in den Speicher des Touch Displays geladen werden, damit die Bilder für die grafische Darstellung der Buttons verwendet werden können. Bevor die Bilder in den Speicher des Displays geladen werden können, müssen diese zunächst im Nextion Editor der Menüoberfläche hinzugefügt werden. Im *Ressourcenbereich* des Editors muss hierfür der Reiter *Picture* ausgewählt werden und anschließend kann über das + – Symbol das erstellte Bild zum Projekt hinzugefügt werden. **Abbildung 33** zeigt den Bildausschnitt der Ansicht *Ressourcenbereich* für den gewählten Reiter *Picture*. Für alle hinzugefügten Images wird vom Nextion Editor eine fortlaufende ID erstellt. Über diese ID kann ein Image vom Steuerungsprogramm referenziert werden. Die erstellten Bilder können anschließend über den *Attributbereich* einem ausgewählten Button hinzugefügt werden. Über das Attribut *pic* kann jetzt das erstellte Bild für die Darstellung des Buttons ausgewählt werden, wenn dieser nicht gedrückt wird. Über das Attribut *pic2* wird das Bild ausgewählt, das beim Drücken des Buttons erscheinen soll. Über das Attribut *pco* kann die Schriftfarbe für den nicht gedrückten Button eingestellt werden und über das Attribut *pco2* die Schriftfarbe für den gedrückten Button. Über das Attribut *font* kann jedem Button eine individuelle Schrift zugewiesen werden. Für die Menüoberfläche des Leistungsmessers wurde nach der beschriebenen Vorgehensweise für jeden Zustand eines Buttons ein Bild erstellt. Da eine Steuerung des Leistungsmessers auch über das Programm möglich sein muss, z.B. wenn der Drehencoder für die Navigation verwendet wird, ist es mit dieser Methode möglich, die aktuelle Encoderposition im Menü farblich hervorzuheben. Es muss in diesem Fall vom Steuerungsprogramm ein Befehl an das Display gesendet werden, mit dem an der aktuellen Encoderposition der entsprechende Button in einer anderen Farbe dargestellt wird. Bei der Bedienung des Leistungsmessers über das Touch Display erfolgt die Hervorhebung eines gedrückten Buttons nach der gleichen Vorgehensweise. Durch diese einheitliche Steuerung des Verhaltens wird außerdem sichergestellt, dass beim Wechseln der Eingabe-

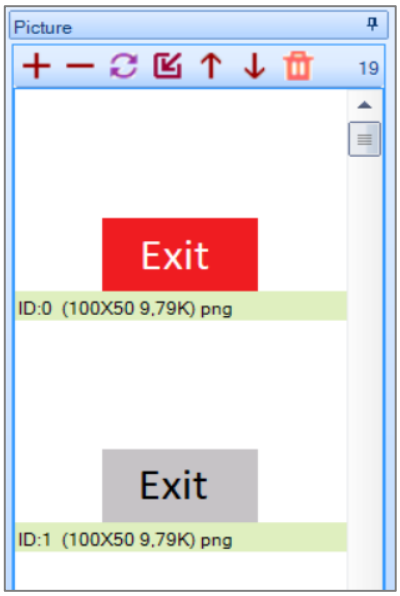


Abbildung 33: *Bildausschnitt (Ansicht Ressourcenbereich / Picture im Nextion Editor)*

methode zwischen Touch Display und Drehencoder im Steuerungsprogramm keine undefinierten Zustände ausgelöst werden können und somit die Steuerung und die farbliche Hervorhebung der Buttons synchron abläuft. Für die Darstellung der Messwerte auf der Menüoberfläche des Displays wird die Komponente *Xfloat* des Nextion Editors verwendet. Die Bezeichnungen der Felder für die Messwertdarstellung (U1, I1, S1 usw.) erfolgen über die Komponente *Text*. **Abbildung 34** zeigt die Ansicht für die Phase L1 im Nextion Editor.

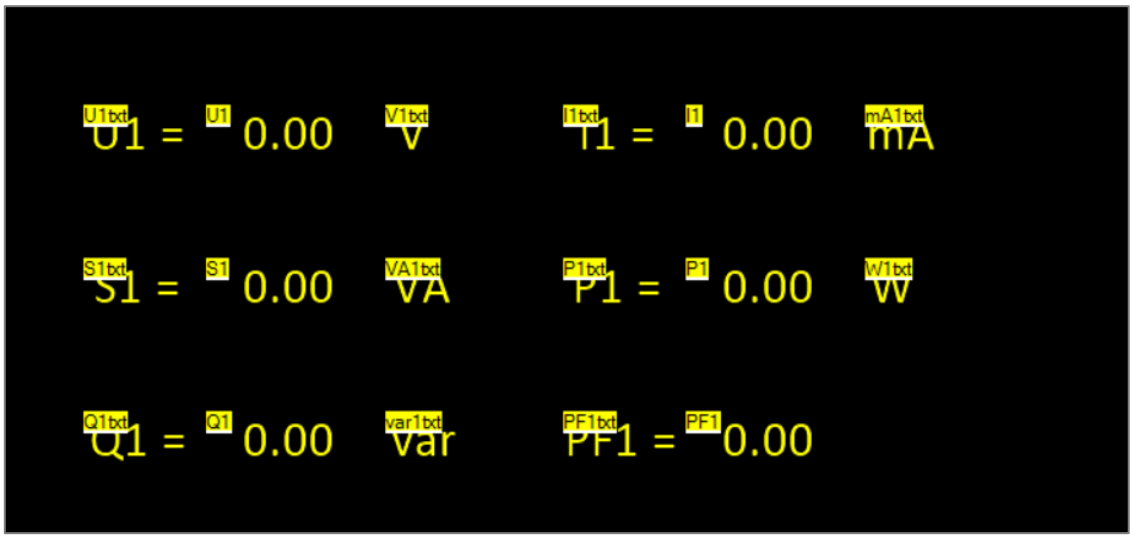


Abbildung 34: *Ansicht Phase L1 im Nextion Editor*

In der Abbildung ist erkennbar, dass auch für die Textkomponenten und die Komponenten zur Darstellung eines Messwertes, das Attribut *objname* vergeben wird. Für das Feld U1 wurde z.B. der Name *U1txt* und für das Feld für die Darstellung des zugehörigen Messwertes die Bezeichnung *U1* gewählt. Um die Messwerte über das Programm in die entsprechenden Felder des Touch Displays schreiben zu können, muss auch im Programm eine Definition der entsprechenden Felder vorgenommen werden. Der Code für die Herstellung einer Verknüpfung zwischen Steuerungsprogramm und Touch Display für das Zahlenfeld U1 sieht beispielsweise folgendermaßen aus: *NexNumber U1_L10 = NexNumber(1, 1, "U1");*
Die Angabe *NexNumber* ist erforderlich, weil es sich bei der erzeugten Komponente im Nextion Editor um ein Zahlenfeld handelt. Die Bezeichnung *U1_L10* ist der vergebene *Variablenname* im Programm, über welchen das Feld referenziert werden kann. Die erste Zahl *1* des Klammerausdrucks gibt die Seite an, auf welcher sich der Button innerhalb der Menüoberfläche befindet. Die darauffolgende Zahl *1* ist der Wert des Attributs *id*. Die Angabe *"U1"* des Klammerausdrucks ist der vergebene Name für das Attribut *objname* der erzeugten Komponente für die Darstellung des Messwertes. Alle Zahlenfelder der Menüoberfläche müssen nach dieser Vorgehensweise im Steuerungsprogramm definiert werden. Um einen Wert in ein Zahlenfeld zu schreiben, wird eine Funktion verwendet, auf welche über die einzubindende Bibliothek (Library) zur Nutzung des Displays bzw. der Codeanweisungen, zugegriffen werden kann. Um z.B. in das oben beschriebene

Zahlenfeld U1 einen Wert zu schreiben, muss folgende Codeanweisung im Steuerungsprogramm stehen: *U1_L10.setValue(U1 * 100);*

Die Funktion *setValue()* ist die Funktion, welche über die Bibliothek zur Verfügung gestellt wird. Die Variable mit dem Namen *U1* des Klammerausdrucks ist die Variable des Programms, welche den entsprechenden Wert enthält, der in das Feld *U1* geschrieben werden soll. Der Wert muss mit dem Faktor *100* multipliziert werden, damit der Wert in der gewünschten Einheit (V) im Feld angezeigt wird. Alle Zahlenfelder werden nach dieser Vorgehensweise entsprechend über das Programm angesprochen und es können somit kontinuierlich die dargestellten Messwerte aktualisiert werden.

Um die mit dem Nextion Editor erstellte Menüoberfläche verwenden zu können, muss diese in den Speicher des Touch Displays geladen werden. Das Display besitzt hierfür auf der Rückseite einen Steckplatz für eine Micro SD – Karte (**Abbildung 35**). Eine im FAT32 – Format formatierte SD – Karte mit einer maximalen Speicherkapazität von 32 GB muss verwendet werden, um die erstellte Oberfläche auf das Display übertragen zu können. Im Nextion Editor muss im *Hauptmenü* der Menüpunkt *File* und anschließend die Option *TFT file output* gewählt werden. Nach der Auswahl des Zielverzeichnisses wird die Datei *Nextion.tft* erstellt. Diese Datei muss, falls ein anderes Zielverzeichnis gewählt wurde, auf die formatierte SD – Karte kopiert werden und danach kann mit dem Übertragen der Datei begonnen werden. Für das Hochladen der Datei muss sich das Nextion Touch Display zunächst im ausgeschalteten Zustand befinden. Nach dem Einstecken der SD – Karte und dem Einschalten des Displays beginnt der Hochladevorgang. Eine Anzeige auf dem Display zeigt den aktuellen Status dieses Vorgangs. Wenn das Übertragen der Datei abgeschlossen wurde, muss das Touch Display ausgeschaltet und die SD – Karte entnommen werden. Nach dem erneuten Einschalten des Displays kann die erstellte Menüoberfläche verwendet werden.

Abbildung 35: *SD – Karten – Slot (Nextion Touch Display)*

Im Folgenden werden die wichtigsten Code – Anweisungen des Arduino – Programms für die Steuerung des Leistungsmessers erklärt. Das Programm setzt sich aus 3 Teilen zusammen. Im ersten Teil werden die erforderlichen Bibliotheken (Libraries) und Header – Dateien, welche für die Nutzung der Hardware erforderlich sind, in das Steuerungsprogramm eingebunden. Außerdem erfolgt die Deklaration der Variablen, die für die Berechnung der Messwerte und der Auswertung der Drehencoder – Position, verwendet werden. Zusätzlich erfolgt die Definition einzelner Pins, welche für den Betrieb des ADE9000 – Boards und des Drehencoders, erforderlich sind.

```
#define PinButton 16
#define PinA      17
#define PinB      20
```

Abbildung 36: *Pin – Definitionen für den Drehencoder*

Abbildung 36 zeigt die Pin – Definitionen für die Verwendung des Drehencoders. Der Zustandsautomat (State Machine), welcher die verschiedenen aufrufbaren Seiten der Menüoberfläche des Leistungsmessers enthält, wird am Ende des ersten Teils definiert. **Abbildung 37** zeigt den Programmausschnitt für das Einbinden der erforderlichen Bibliotheken. Die Bibliothek (*Enhanced – Nextion – Library – master*), welche die Kommunikation des Nextion Displays mit dem Mikrocontroller ermöglicht, ist nicht im Funktionsumfang der Arduino – Entwicklungsumgebung enthalten und muss daher aus dem Internet heruntergeladen werden. Mit dem Befehl *#include "Nextion.h"* wird die Bibliothek in das Steuerungsprogramm eingebunden. Diese Bibliothek enthält aber nur die Einstellungen für die Nutzung von VAR – basierten Mikrocontrollern wie z.B. Arduino UNO, MEGA2560. Der MKR ZERO gehört aber zu der Kategorie der SAMD – basierten Mikrocontroller. Daher müssen in der Bibliothek einige Anpassungen vorgenommen werden, damit auch der MKR ZERO mit dem Nextion – Display kommunizieren kann. In der Datei *NexConfig.h* der heruntergeladenen Bibliothek müssen Änderungen vorgenommen werden. Um das Nextion – Display über den seriellen Port des Mikrocontrollers ansprechen zu können, darf die Codezeile *#define nexSerial Serial* nicht auskommentiert sein. Für das Board MKR ZERO muss die folgende Änderung vorgenommen werden: *#define nexSerial Serial1*. Hiermit wird der hardwareseitige serielle Port des Mikrocontrollers für die Kommunikation aktiviert. Im Falle des Boards MKR ZERO ist es die Einstellung *Serial1*. Die Codezeile *#define NEX_SOFTWARE_SERIAL* muss auskommentiert werden. In der Datei *NexUpload.cpp* muss die Codezeile *#include <SoftwareSerial.h>* auskommentiert werden. In der Datei *NexHardware.cpp* muss die Funktion *bool nexInit(...)* angepasst werden. Die Codezeile *nexSerial.begin(...)* muss mit der gewünschten Baudrate für die Kommunikation ergänzt / angepasst werden, z.B. in: *nexSerial.begin(115200)*. Für den Datenaustausch zwischen MKR ZERO und ADE9000 Board über die SPI – Schnittstelle ist das Einbinden einer weiteren Bibliothek erforderlich. Der Befehl *#include <SPI.h>* ermöglicht die Kommunikation mit Geräten, die eine SPI – Schnittstelle besitzen, wobei der MKR ZERO

```
#include "Nextion.h"
#include <SPI.h>
#include <ADE9000RegMap.h>
#include <ADE9000API.h>
```

Abbildung 37*: Einbinden der erforderlichen Bibliotheken (Nextion Display u. ADE9000 Board)*

```
struct ActivePowerRegs powerRegs;
struct CurrentRMSRegs curntRMSRegs;
struct VoltageRMSRegs vltgRMSRegs;
```

Abbildung 38: *Structure – Deklarationen (Speicherung der Registerdaten des ADE9000)*

dabei das Master – Gerät darstellt. Über die Befehle *#include <ADE9000RegMap.h>* und *#include <ADE9000API.h>* werden spezielle Header – Dateien in das Programm eingebunden, welche die Nutzung des ADE9000 – Boards über das Steuerungsprogramm ermöglichen. Für die Speicherung der Daten der Register Leistung (Power), Strom (Current) und Spannung (Voltage) des ADE9000 wird die in **Abbildung 38** gezeigte Deklaration im Programm vorgenommen. Für die Berechnung der Messwerte der Phasen L1, L2 u. L3 des ADE9000 Boards erfolgt die Deklaration von Variablen für jede Phase des Boards. Die Phasen L1, L2 u. L3 werden im Programm durch die Funktionen Phase_A(), Phase_B() u. Phase_C() abgebildet. In diesen Funktionen erfolgen die Berechnungen der einzelnen Messwerte der jeweiligen Phase. Zusätzlich gibt es die Funktion Phase_N() im Steuerungsprogramm. In dieser Funktion wird der Messwert für die Stromstärke des Neutralleiters berechnet. Die Rohdaten der Messwerte Strom, Spannung u. Wirkleistung werden in den oben genannten Funktionen der Phasen jeweils in einer Variablen gespeichert. **Abbildung 39** zeigt die Deklaration dieser Variablen im Programm. Das Speichern der Rohdaten in den angelegten Variablen erfolgt durch Auslesen der jeweiligen Register des ADE9000. Für die Werte Strom, Spannung u. Leistung der Funktion Phase_A() erfolgt das Auslesen der Register wie in **Abbildung 40** dargestellt. Über entsprechenden Code erfolgt das Auslesen der jeweiligen Register des ADE9000 in den Funktionen Phase_B(), Phase_C() u. Phase_N(). Mit diesen Registerwerten werden die eigentlichen Messwerte der einzelnen Phasen berechnet. Die Berechnung dieser Messwerte (Strom, Spannung u. Wirkleistung) erfolgt, indem die Werte der Variablen (**Abbildung 39**) durch einen konstanten Wert geteilt werden. Diese Konstanten wurden jeweils experimentell ermittelt, indem die Ergebnisse für die Berechnungen der Messwerte Strom (I), Spannung (U) u. Wirkleistung (P) mit einem Multimeter überprüft wurden. Für jede Berechnungsfunktion (Phase_A(), Phase_B(), Phase_C() u. Phase_N()) erfolgen für die Speicherung der Messwerte Variablendeklarationen im Programm. Für die Speicherung der Messwerte der einzelnen Phasen erfolgt die Deklaration einer jeweiligen Variablen vom Typ double. Die Speicherung des Messwertes *Spannung* der jeweiligen Phase erfolgt in den Variablen *U1*, *U2* u. *U3*. Die Speicherung des Messwertes *Strom* der jeweiligen

```
double AI = 0;
double AV = 0;
double AW = 0;
double BI = 0;
double BV = 0;
double BW = 0;
double CI = 0;
double CV = 0;
double CW = 0;
double NI = 0;
```

Abbildung 39: *Variablendeklaration (Speicherung der ausgelesenen Register des ADE9000)*

```
AI = ade9000.SPI_Read_32(ADDR_AIRMS);
ade9000.ReadVoltageRMSRegs(&vltgRMSRegs);
ade9000.ReadActivePowerRegs(&powerRegs);
AV = vltgRMSRegs.VoltageRMSReg_A;
AW = powerRegs.ActivePowerReg_A;
```

Abbildung 40: *Auslesen u. Speichern der Registerdaten (Funktion Phase_A())*

Phase erfolgt in den Variablen *I1*, *I2*, *I3* u. *IN*. Die Speicherung des Messwertes *Scheinleistung* der jeweiligen Phase erfolgt in den Variablen *S1*, *S2* u. *S3*. Die Speicherung des Messwertes *Wirkleistung* der jeweiligen Phase erfolgt in den Variablen *P1*, *P2* u. *P3*. Die Speicherung des Messwertes *Blindleistung* der jeweiligen Phase erfolgt in den Variablen *Q1*, *Q2* u. *Q3*. Der *Leistungsfaktor* der jeweiligen Phase wird in den Variablen *PF1*, *PF2* u. *PF3* gespeichert. Mit den Formeln (**Abbildung 41**) werden die Messwerte der Phase L1 berechnet. Die Berechnung der Messwerte erfolgt spiegelbildlich für die Phasen L2 u. L3. Die erforderlichen Deklarationen für die einzelnen erstellten Buttons der Menüoberfläche des Nextion Displays und die zugehörigen Funktionen befinden sich in der Datei *NextionDisplayButtons.h*. Diese wird mit dem Befehl *#include "NextionDisplayButtons.h"* in das Steuerungsprogramm eingebunden. Die erforderlichen Deklarationen für die einzelnen erstellten Felder der Menüoberfläche des Nextion Displays und die Funktionen zum Senden der Messwerte an das Display befinden sich in der Datei *NextionDisplayValues.h.* Diese wird mit dem Befehl *#include "NextionDisplayValues.h"* in das Steuerungsprogramm eingebunden. Die Deklarationen und Funktionen für die Kommunikation zwischen Nextion Display und Steuerungsprogramm befinden sich in den genannten beiden Dateien, um das Programm übersichtlicher gestalten zu können. Das Steuerungsprogramm entspricht der Struktur einer State Machine (Zustandsautomat). Im ersten Teil des Programms erfolgt daher die Deklaration des Zustandsautomaten. Mit der folgenden Anweisung wird der Zustandsautomat mit den möglichen Zuständen deklariert:

```
U1 = AV / 92879; (Spannung)
I1 = AI / 762; (Strom)
S1 = (U1 * I1) / 1000; (Scheinleistung)
P1 = AW / 528; (Wirkleistung)
Q1 = sqrt(S1 * S1 - P1 * P1); (Blindleistung)
PF1 = P1 / S1; (Leistungsfaktor)
```

Abbildung 41: *Berechnung der Messwerte für die Phase L1*

enum State_Machine {PAGE0 = 0, PAGE1 = 1, PAGE2 = 2, PAGE3 = 3, PAGE4 = 4, PAGE5 = 5, PAGE6 = 6, PAGE7 = 7};

Die Ausdrücke in geschweiften Klammern sind die verschiedenen Zustände bzw. die aufrufbaren Seiten der Menüoberfläche des Automaten. Über diesen Zustandsautomaten erfolgt die Steuerung des Leistungsmessers. Man kann in der Deklaration des Automaten erkennen, dass die Menüoberfläche aus insgesamt 8 Seiten besteht. Den zweiten Teil des Steuerungsprogramms stellt die setup() – Funktion dar. Diese Funktion wird nur einmalig aufgerufen und enthält die Definitionen der Pins des MKR ZERO, die als Eingang oder Ausgang genutzt werden sollen. Mit dem Aufruf der Funktion *nexInit()* erfolgt die Konfiguration für die serielle Kommunikation zwischen dem MKR ZERO und dem Nextion Touch Display. **Abbildung 42** zeigt die Definitionen der verwendeten Pins für die Nutzung des Drehencoders. Wie im *Abschnitt 4.3.3* (Variante 3) beschrieben, werden alle En-

```
pinMode(PinButton, INPUT_PULLUP);
pinMode(PinA, INPUT_PULLUP);
pinMode(PinB, INPUT_PULLUP);
```

Abbildung 42: *Definitionen der Pins für den Drehencoder*

coderausgänge mit einem Pull – up – Widerstand betrieben. Hierfür können aber auch die vom MKR ZERO bereitgestellten Widerstände genutzt werden. Dann müssen am Drehencoder keine speziellen Widerstände verlötet werden. **Abbildung 42** zeigt die Definitionen für die Verwendung der internen Widerstände. Um den Drehencoder für die Navigation innerhalb der Menüoberfläche verwenden zu können, muss mit einem Pin des Encoders ein Interrupt im Programm ausgelöst werden. Da während des Betrieb des Leistungsmessers fortlaufend Messwerte über den seriellen Port zum Nextion Display gesendet werden, muss das Senden der Werte für eine kurze Zeit unterbrochen werden, damit über den seriellen Port der Befehl für die farbliche Hervorhebung der aktuellen Encoderposition (roter Button) an das Display gesendet werden kann. Ansonsten würde das kontinuierliche Senden der Messwerte die Verwendung des Encoders verzögern oder blockieren. Aus diesem Grund ist PinA (CLK) des Encoders mit einem Interrupt – fähigen Pin des MKR ZERO Boards verbunden. Über den folgenden Befehl wird der Interrupt für den Drehencoder definiert:

attachInterrupt(PinA, isrEncoderPosition, FALLING);

Über den PinA (CLK) des Encoders wird bei einer fallenden Flanke der Interrupt ausgelöst. Über PinA erfolgt die Auswertung, ob der Encoder gedreht wurde. Wenn dies der Fall ist, wird die Funktion *isrEncoderPosition()* im Programm aufgerufen und alles Weitere wird für diesen kurzen Moment im Programm gestoppt. In der Funktion *isrEncoderPosition()* wird die aktuelle Position des Encoders ermittelt, das heisst, wenn am Encoder gedreht wird, werden über diese Funktion die aktuellen Schritte gezählt. **Abbildung 43** zeigt die definierten Pins für das ADE9000 Board. Die Pins sind als Ausgänge konfiguriert. Durch Schreiben eines LOW – Pegels auf dem Pin *PinPM1*, wird für den ADE9000 der PSM0 mode (Normal mode) eingestellt. Dieser Betriebsmodus ermöglicht die Nutzung aller Funktionen des ADE9000. Durch Schreiben eines High – Pegels auf dem Pin *PinReset*, wird für den ADE9000 ein Reset ausgeführt. **Abbildung 44** zeigt die Setup – Konfiguration der Pins für die Relais 1, 2 u. 3, mit welchen die Messgenauigkeit der Phasen L1, L2 u. L3 an den Spannungsbereich angepasst werden kann. Die Pins 1, 2 u. 3 sind als Ausgänge konfiguriert und es wird jeweils ein LOW – Pegel auf diese Ausgänge geschrieben. Dadurch werden die Relais 1, 2 u. 3 nicht geschaltet. **Abbildung 45** zeigt die Konfiguration für die Nutzung der SPI – Schnittstelle und den Aufruf einer Funktion zur Initialisierung der Register des

```
pinMode(PinPM1, OUTPUT);
digitalWrite(PinPM1, LOW);

pinMode(PinReset, OUTPUT);
digitalWrite(PinReset, HIGH);
```

Abbildung 43: *Setup – Konfiguration (ADE9000)*

```
pinMode(1, OUTPUT);
digitalWrite(1, LOW);
pinMode(2, OUTPUT);
digitalWrite(2, LOW);
pinMode(3, OUTPUT);
digitalWrite(3, LOW);
```

Abbildung 44: *Setup – Konfiguration (Relais 1, 2 u. 3)*

```
ade9000.SPI_Init(SPI_SPEED, PinCS);
ade9000.SetupADE9000();
ade9000.SPI_Write_16(ADDR_RUN, 0x1);
```

Abbildung 45: *Setup – Konfiguration (SPI - Schnittstelle u. ADE9000)*

ADE9000. Die SPI – Schnittstelle wird für den Datenaustausch zwischen MKR ZERO und ADE9000 verwendet.

```
switch (state)
{
  case PAGE1:
    setValuesPage1();
    break;
  case PAGE2:
    setValuesPage2();
    break;
  case PAGE3:
    setValuesPage3();
    break;
  case PAGE4:
    setValuesPage4();
    break;
  case PAGE5:
    setValuesPage5();
    break;
  case PAGE6:
    setValuesPage6();
    break;
  case PAGE7:
    setValuesPage7();
    break;
}
```

Abbildung 46: *Switch – Anweisung (Darstellung der Messwerte der Menüoberfläche)*

Den dritten Teil des Steuerungsprogramms stellt die loop() – Funktion dar. Diese Funktion enthält das eigentliche Programm und wird zyklisch durchlaufen. Mit dem folgenden Befehl wird am Anfang der loop() – Funktion eine Variable deklariert, die den vorgegebenen Zustand für den Zustandsautomaten speichert:

static char state = PAGE0;

Der Anfangszustand, den der Zustandsautomat beim Programmstart einnimmt, ist *PAGE0*. Diese Seite stellt die Hauptseite der Menüoberfläche dar. Die Funktionen Phase_A(), Phase_B(), Phase_C() u. Phase_N() für die Berechnung der Messwerte der Phasen L1, L2 u. L3 des Leistungsmessers werden nacheinander aufgerufen. Durch den zyklischen Programmablauf werden durch den Aufruf dieser Funktionen immer die aktuellen Messwerte berechnet. **Abbildung 46** zeigt eine *switch* – Anweisung. In Abhängigkeit vom jeweiligen Zustand, in dem sich der Zustandsautomat befindet, wird eine Funktion aufgerufen, welche die für die aktuell gewählte Seite der Menüoberfläche erforderlichen Messwerte, an das Nextion Display sendet. Mit dem Befehl *nexLoop(nex_listen_list);* wird überprüft, ob ein Button innerhalb der Menüoberfläche des Nextion Touch Displays gedrückt wurde. Wenn ein Button gedrückt wurde, dann wird der entsprechende definierte Code ausgeführt und eine Aktion ausgelöst. Über If – Abfragen im Programm wird die farbliche Hervorhebung der Buttons in Abhängigkeit zu der aktuellen Encoderposition geregelt. **Abbildung 47** zeigt die If – Abfragen. Am Anfang erfolgt die Prüfung, ob sich die aktuelle Encoderposition durch das Drehen am Encoder verändert hat. Wenn sich die aktuelle Position verändert hat, wird in Abhängigkeit des aktuellen Zustands, in welchem sich der Zustandsautomat befindet, eine Funktion aufgerufen, mit der die farbliche Hervorhebung der Buttons gesteuert wird. In diesen Funktionen werden Befehle verwendet, mit denen die im Nextion Display gespeicherten

```
if (checkEncoderPositionChanged() == true)
{
  if (state == PAGE0)
  {
    updateNextionButtonPictures_MainMenue();
  }
  else  if (state == PAGE2)
  {
    updateNextionButtonPictures_L123();
  }
  else if (state == PAGE6)
  {
    updateNextionButtonPictures_SPQ();
  }
  else if (state == PAGE7)
  {
    updateNextionButtonPictures_UIPF();
  }
  else
  {
    updateNextionButtonPictures_ExitOnly();
  }
}
```

Abbildung 47*: If – Abfragen (Hervorhebung der Buttons in Abhängigkeit der Encoderposition)*

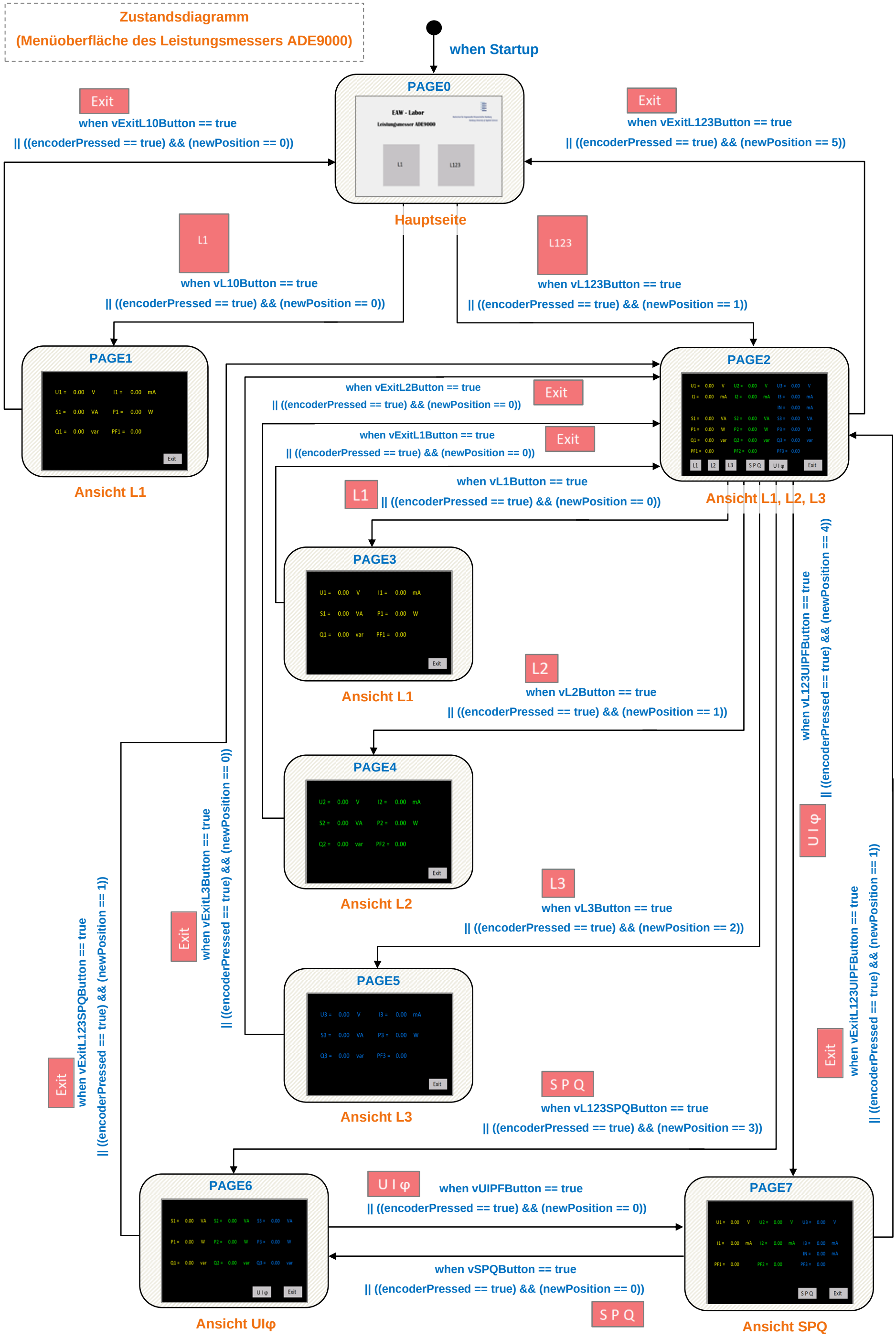
```
void updateNextionButtonPictures_MainMenue()
{
  if (newPosition == 0)
  {
    sendCommand("pic 150,230,10");
    sendCommand("pic 450,230,13");
  }
  else
  {
    sendCommand("pic 450,230,12");
    sendCommand("pic 150,230,11");
  }
}
```

Abbildung 48*: Funktion updateNextionButtonPictures_MainMenue()*

Bilder der Buttons an der definierten Position der aktuellen Seite der Menüoberfläche erscheinen. Hiermit wird die Visualisierung der Drehencoderposition ermöglicht. Anhand der Funktion *updateNextionButtonPictures_MainMenue()* soll die grundsätzliche Funktionsweise erläutert werden. **Abbildung 48** zeigt den Programmcode dieser Funktion. Wenn auf der Hauptseite (PAGE0) der Menüoberfläche eine Drehung am Encoder festgestellt wird und die im Programm ermittelte Position für die Drehung einer 0 entspricht, werden mit den Befehlen *sendCommand("pic 150,230,10");* und *sendCommand("pic 450,230,13");* die entsprechenden Bilder der Buttons dargestellt. Die Funktion *sendCommand()* ist Bestandteil der im Programm eingebundenen Bibliothek für die Verwendung des Nextion Displays und kann daher genutzt werden. Die ersten beiden Zahlen des Klammerausdrucks geben die X- u. Y – Koordinate an, an der das Bild (pic) positioniert werden soll. Die letzte Zahl des Klammerausdrucks ist die ID des jeweiligen Bildes. Diese ID wird beim Hinzufügen eines Bildes im Nextion Editor automatisch vergeben. Durch die Angabe der entsprechenden ID kann somit auf ein erstelltes Bild zugegriffen werden. Mit der Zahl 10 des ersten sendCommand() – Befehls wird das Bild des roten Buttons für die Auswahl der Phase L1 an der entsprechenden Stelle positioniert und mit der Zahl 13 des zweiten sendCommand() – Befehls wird der andere Button der Hauptseite (Button L123) in grauer Farbe an der angegebenen Position dargestellt. Die Befehle des else – Ausdrucks werden ausgeführt, wenn am Encoder nochmals gedreht wurde und die interne Position daher nicht mehr den Wert 0 hat. In diesem Fall wird mit den sendCommand() – Anweisungen der Button L123 in roter Farbe dargestellt und der Button L1 in grauer Farbe. Mit Hilfe dieser Funktionen kann der Drehencoder für die Navigation in der Menüoberfläche verwendet werden. Die anderen Funktionen der If – Abfragen (**Abbildung 47**) verwenden das gleiche Prinzip für die Visualisierung. Die Steuerung des Zustandsautomaten erfolgt mit Hilfe einer switch – Struktur. Hiermit können in Abhängigkeit des aktuellen Zustands die definierten Aktionen ausgeführt werden. Der vorgegebene Anfangszustand ist *PAGE0*. Diese Seite stellt die Hauptseite der Menüoberfläche dar und wird daher nach dem Einschalten des Leistungsmessers angezeigt. In Abhängigkeit des gewählten Buttons auf dieser Seite, wechselt der Zustandsautomat durch den entsprechenden Befehl in einen anderen Zustand. Dieser Zustandswechsel erfolgt entweder durch Drücken eines Buttons auf der Menüoberfläche des Touch Displays oder durch Drücken des switches am Drehencoder. Durch den Vergleich der aktuellen Encoderposition mit den vordefinierten Positionen innerhalb der Zustände, kann durch Drücken des Switches am Encoder eine Auswahl getroffen werden und das Verhalten entspricht dem Drücken eines Buttons.

Zustandsdiagramm

(Menüoberfläche des Leistungsmessers ADE9000)



5.2 Kalibrierung und Testen

Um die Genauigkeit des Leistungsmessers gewährleisten zu können und für die Durchführung eventueller Kalibrierungen, wird die Messgenauigkeit der Phasen L1, L2 u. L3 mit Hilfe einer Messreihe überprüft. Hierzu wird der Leistungsmesser im Labor für elektrische Anlagen und Wandler an das Drehstromnetz angeschlossen. Jeder Ausgang (Leiter) des Drehstromanschlusses des Labortisches wird mit jeweils einem Multimeter verbunden und über den Neutralleiter wird der jeweilige Stromkreis geschlossen (Sollwert Spannungen). Zusätzlich werden die Eingänge des Leistungsmessers über jeweils ein Multimeter mit den Ausgängen des Drehstromanschlusses des Labortisches verbunden (Sollwert Ströme). Die Ausgänge des Leistungsmessers werden mit jeweils einer Last verbunden und die Ausgänge der jeweiligen Last werden mit dem Neutralleiteranschluss des Leistungsmessers verbunden. Für die Messung des Neutralleiters des Leistungsmessers wird der zweite Neutralleiteranschluss des Messgerätes über ein zusätzliches Multimeter mit dem Neutralleitereingang des Drehstromanschlusses verbunden und somit wird der Stromkreis geschlossen. Mit dieser Schaltung werden zwei Messreihen aufgezeichnet. Für jede Messreihe wird eine andere Last in die Schaltung integriert. Nach jeder Messung können mit Hilfe dieser Schaltung die *Sollwerte* (Messwerte der Multimeter) mit den *Istwerten* (Messwerte des Leistungsmessers) verglichen werden. Für die Prüfung der Messgenauigkeit des Leistungsmessers im *Spannungsbereich* wird in die Schaltung ein Bauteil mit 3 Halogenlampen (48W, G9) als Last integriert. **Abbildung 49** zeigt den Aufbau der Schaltung.

Abbildung 49: *Prüfung der Messgenauigkeit im Spannungsbereich (Schaltungsaufbau)*

Für die Durchführung der Messungen wird der Drehstromanschluss von Tisch 8 des EAW – Labors verwendet. Die Messreihe beginnt mit einer Spannung von 230 V. Danach wird die Spannung schrittweise heruntergeregelt. **Tabelle 8** zeigt die *Sollwerte* und **Tabelle 9** die *Istwerte* der Messreihe.

Sollwerte						
U1	U2	U3	I1	I2	I3	IN
230,6 V	228,6 V	228,9 V	219,9 mA	221,4 mA	224,3 mA	4,0 mA
200,6 V	198,0 V	198,5 V	203,0 mA	204,5 mA	207,7 mA	4,1 mA
150,2 V	148,8 V	148,7 V	173,5 mA	175,0 mA	178,1 mA	4,1 mA
100,7 V	98,8 V	98,9 V	139,2 mA	139,8 mA	143,3 mA	3,9 mA
50,6 V	49,6 V	49,2 V	95,9 mA	96,3 mA	100,5 mA	4,5 mA
30,0 V	28,9 V	28,9 V	74,0 mA	73,8 mA	79,1 mA	5,0 mA
20,1 V	19,3 V	19,0 V	61,7 mA	61,4 mA	67,2 mA	5,5 mA
10,03 V	9,49 V	9,23 V	46,9 mA	46,5 mA	51,5 mA	5,1 mA
5,76 V	5,32 V	4,77 V	38,0 mA	36,6 mA	38,1 mA	2,6 mA
2,23 V	1,96 V	1,81 V	21,7 mA	19,6 mA	19,6 mA	0,9 mA
1,51 V	0,96 V	0,94 V	16,2 mA	10,8 mA	11 mA	0,7 mA
0,75 V	0,50 V	0,53 V	8,8 mA	5,8 mA	6,4 mA	0,6 mA

Tabelle 8: Sollwerte (Messwerte der Multimeter)

Istwerte						
U1	U2	U3	I1	I2	I3	IN
230,0 V	227,1 V	229,9 V	219,6 mA	220,5 mA	223,5 mA	4,6 mA
198,8 V	196,5 V	199,4 V	202,5 mA	204,3 mA	207,5 mA	4,6 mA
148,8 V	147,4 V	148,8 V	173,2 mA	174,4 mA	177,7 mA	4,5 mA
99,9 V	98,2 V	99,4 V	139,1 mA	139,5 mA	143,2 mA	4,4 mA
50,1 V	49,2 V	49,3 V	95,8 mA	96,2 mA	100,4 mA	4,6 mA
29,8 V	28,8 V	29,0 V	74,1 mA	73,8 mA	79,2 mA	5,5 mA
19,9 V	19,2 V	19,1 V	61,8 mA	61,5 mA	67,2 mA	5,9 mA
9,93 V	9,39 V	9,22 V	47,1 mA	46,6 mA	51,6 mA	5,5 mA
5,83 V	5,21 V	4,67 V	38,2 mA	36,8 mA	38,3 mA	3,4 mA
2,20 V	1,81 V	1,72 V	22,1 mA	19,9 mA	19,8 mA	0 mA
1,42 V	0,87 V	0,87 V	16,3 mA	11,1 mA	11,2 mA	0 mA
0,71 V	0,44 V	0,50 V	9,2 mA	6,2 mA	6,9 mA	0 mA

Tabelle 9: Istwerte (Messwerte der Phasen L1, L2 u. L3 des Leistungsmessers)

Die Messungen ergeben, dass ab einer Spannung kleiner 2 V Ungenauigkeiten bei den *Istwerten* der Messwerte *U1*, *U2* u. *U3* entstehen. In diesem Bereich sind die vom Leistungsmesser ermittelten Messwerte ungenau und zeigen eine Abweichung von ca. 5 % zu den *Sollwerten* der Multimeter. Für diesen Bereich müsste, wenn möglich, noch eine Verbesserung der Kalibrierung vorgenommen werden.

Für die Prüfung der Messgenauigkeit des Leistungsmessers im *Strombereich* erfolgt die Durchführung einer neuen Messreihe. Als Last für die Leiter *L1*, *L2* u. *L3* kommt jeweils ein 50 Ω Widerstand zum Einsatz. **Abbildung 50** zeigt die verwendeten Widerstände. Die restliche Schaltung bleibt unverändert.

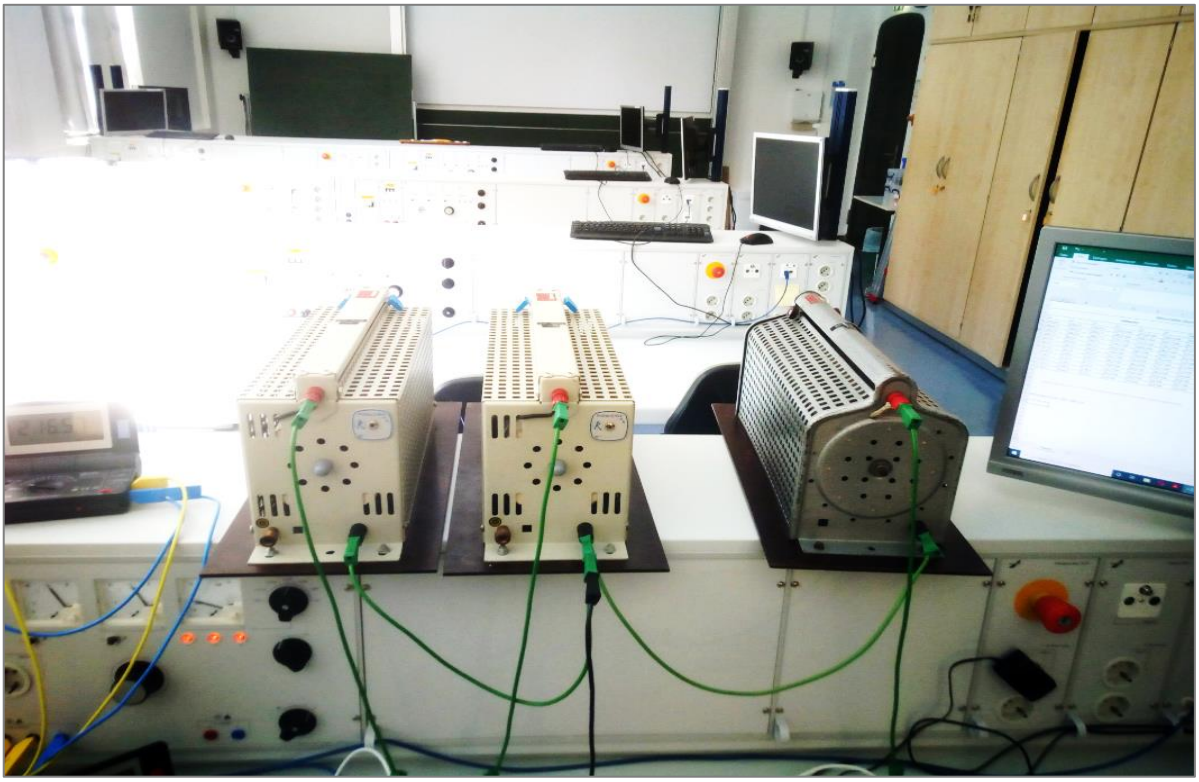


Abbildung 50: *Widerstände für die Prüfung der Messgenauigkeit im Strombereich*

Die Messreihe beginnt mit einem Anfangsstrom von 4 A. Danach wird der Strom schrittweise heruntergeregelt. **Tabelle 10** zeigt die *Sollwerte* und **Tabelle 11** die *Istwerte* der Messreihe.

Sollwerte						
U1	U2	U3	I1	I2	I3	IN
219,4 V	216,1 V	216,9 V	4,06 A	4,05 A	4,14 A	128,3 mA
111,8 V	109,8 V	110,5 V	2,05 A	2,03 A	2,08 A	57,8 mA
54,7 V	53,6 V	53,7 V	1,00 A	0,99 A	1,00 A	23,1 mA
27,5 V	26,6 V	26,7 V	0,50 A	0,49 A	0,50 A	13,4 mA
11,7 V	10,7 V	10,9 V	0,21 A	0,2 A	0,20 A	9,5 mA
6,1 V	5,5 V	5,0 V	0,11 A	0,10 A	0,09 A	10,3 mA
3,0 V	2,6 V	2,3 V	0,055 A	0,048 A	0,043 A	7,1 mA

Tabelle 10: Sollwerte (Messwerte der Multimeter)

Istwerte						
U1	U2	U3	I1	I2	I3	IN
217,5 V	214,7 V	217,3 V	4,05 A	4,02 A	4,12 A	127,8 mA
110,3 V	108,3 V	110,1 V	2,05 A	2,03 A	2,08 A	57,9 mA
54,2 V	53,1 V	53,4 V	1,00 A	0,99 A	1,00 A	23,2 mA
27,2 V	26,4 V	26,6 V	0,50 A	0,49 A	0,50 A	13,4 mA
11,4 V	10,6 V	10,8 V	0,21 A	0,2 A	0,20 A	9,7 mA
5,9 V	5,4 V	5,0 V	0,11 A	0,10 A	0,09 A	10,6 mA
2,94 V	2,54 V	2,30 V	0,054 A	0,048 A	0,043 A	7,3 mA

Tabelle 11: Istwerte (Messwerte der Phasen L1, L2 u. L3 des Leistungsmessers)

Die Messungen ergeben im *Strombereich* keine nennenswerten Ungenauigkeiten zwischen *Sollwerten* und *Istwerten*. Für diesen Bereich liefern die Messwerte der einzelnen Phasen des Leistungsmessers eine hohe Genauigkeit.

6 Datenübertragung zum PC

```
String U1string = String(U1);
String U2string = String(U2);
String U3string = String(U3);
String I1string = String(I1);
String I2string = String(I2);
String I3string = String(I3);
String S1string = String(S1);
String S2string = String(S2);
String S3string = String(S3);
String P1string = String(P1);
String P2string = String(P2);
String P3string = String(P3);
String Q1string = String(Q1);
String Q2string = String(Q2);
String Q3string = String(Q3);
String PF1string = String(PF1);
String PF2string = String(PF2);
String PF3string = String(PF3);
```

Abbildung 51: *Variablendeklaration (Speicherung der Messwerte als String)*

Für die Übertragung der Messwerte zum PC bzw. für das Speichern dieser Werte in einer Textdatei, wird ein Programm verwendet, das mit der Programmiersprache *Processing* erstellt wurde. Auf die Programmiersprache *Processing* wurde im Abschnitt 4.5 (Datenübermittlung zum PC) näher eingegangen. Die Speicherung der Messwerte mit Hilfe des Programms erfordert allerdings speziellen Code im Steuerungsprogramm. In der Funktion *normalizeDataPhaseABC()* werden die Inhalte der Variablen, in denen die Messwerte der einzelnen Phasen gespeichert werden (U1, U2, U3, I1, I2, …), in den Datentyp *String* umgewandelt. Es wird dabei jeder Messwert in einer erzeugten Variablen vom Typ *String* gespeichert. **Abbildung 51** zeigt die Deklaration der einzelnen Variablen mit anschließender Zuweisung des jeweiligen Messwertes als *String*. Anschließend wird für jede Phase (L1, L2 u. L3) des Leistungsmessers in der Funktion eine weitere Variable vom Typ *String* erstellt, in der die zuvor erstellten jeweiligen Strings der Messwerte gespeichert werden. **Abbildung 52** zeigt die Variablendeklaration für die Phase L1 (Variable retL1) mit anschließender Zuweisung der erstellten Variablen aus **Abbildung 51**. Für die Phasen L2 u. L3 erfolgt spiegelbildlich die Deklaration der Variablen retL2 u. retL3. Mit einer weiteren Variablen werden die erstellten Strings der einzelnen

```
String retL1 = String(String("Phase L1  ") + String("U1: ") + U1string + String(" V") + String(" ") + String("I1: ") +
I1string + String(" mA") + String(" ") + String("S1: ") + S1string + String(" VA") + String(" ") + String("P1: ") +
P1string + String(" W") + String(" ") + String("Q1: ") + Q1string + String(" var") + String(" ") + String("PF1: ") +
PF1string);
```

Abbildung 52: *Variablendeklaration (Speicherung aller Messwerte der Phase L1 als String)*

Phasen als Gesamtstring als Rückgabewert an das Steuerungsprogramm übergeben. Die folgenden Codeanweisungen zeigen die Deklaration der Variablen für den Gesamtstring mit anschließender Zuweisung der erstellten Variablen mit den Messwerten der Phasen und die Rückgabe des Gesamtstrings ret an das Steuerungsprogramm:

String ret = retL1 + " | " + retL2 + " | " + retL3;

return ret;

Die Funktion *normalizeDataPhaseABC()* wird in der *loop()* – Funktion des Steuerungsprogramms wie folgt aufgerufen:

data = normalizeDataPhaseABC(U1, U2, U3, I1, I2, I3, S1, S2, S3, P1, P2, P3, Q1, Q2, Q3, PF1, PF2, PF3);

Die oben genannten Variablen mit den Messwerten der einzelnen Phasen werden der Funktion beim Aufruf übergeben und der Rückgabewert *ret* der Funktion wird der Variablen *data* zugewiesen. Die Variable *data* ist vom Datentyp *String* und wurde zuvor im Steuerungsprogramm deklariert. Mit der Codeanweisung *Serial.println(data);* wird der String mit den Messwerten aller Phasen (L1, L2 u. L3) des Leistungsmessers über den seriellen Port des MKR ZERO Boards ausgegeben. Durch den zyklischen Programmablauf wird dieser String mit den Messwerten fortlaufend aktualisiert. Die Codeanweisung *Serial.println(data);* hat im Steuerungsprogramm eine große Bedeutung. Diese Anweisung ermöglicht dem mit der Programmiersprache *Processing* erstellten Programm das Mitlesen des Datenstroms auf dem seriellen Port des MKR ZERO. Für das Schreiben dieses Strings wurde daher die oben erläuterte Funktion erstellt. **Abbildung 53** zeigt den Programmcode des *Processing* Programms für das Speichern der Messwerte. Mit der Codeangabe *String messDaten;* erfolgt die Deklaration einer Variablen vom Typ String. In dieser Variablen wird der auf dem seriellen Port ausgegebene Datenstrom gespeichert. Mit der Anweisung *messDaten = mySerial.readString();* wird der mit dem Befehl *Serial.println(data);* auf dem seriellen Port ausgegebene Datenstrom mitgelesen und der Variablen *messDaten* zugewiesen. In der setup() – Funktion des Programms muss mit der Codeanweisung *mySerial = new Serial(this, "COM13", 9600);* der Port bekanntgegeben werden, an dem das MKR ZERO Board mit dem PC verbunden ist. In diesem Fall ist es "COM13". Der vergebene Port wird in der Arduino – Entwicklungsumgebung im Menüpunkt *Werkzeuge* unter der Option *Port* angezeigt. Die Zahl *9600* gibt die Baudrate an und sollte mit dem im Programm eingestellten Wert übereinstimmen. Das MKR ZERO Board muss also für das Speichern der Messwerte über die USB – Schnittstelle mit dem

PC verbunden werden. Ansonsten ist das Mitlesen des Datenstroms nicht möglich. Mit der Codeanweisung *output = createWriter("Messdaten.txt");* wird der Dateiname für die Zieldatei vergeben, in der die Messwerte gespeichert werden. In diesem Fall wurde der Dateiname *"Messdaten.txt"* gewählt. Wird kein spezielles Verzeichnis angegeben, dann wird die Datei in dem Verzeichnis erstellt, in welchem sich das *Processing* Programm befindet. Die Funktion draw() des Programms wird kontinuierlich ausgeführt und schreibt den erstellten String mit den Messwerten in die Zieldatei. Diese Funktion ist in einem Arduino – Programm die loop() – Funktion. Das Programm für das Speichern der Messwerte kann durch einen Doppelklick gestartet werden. Nach dem Beenden des Programms kann die erstellte Datei zur weiteren Bearbeitung geöffnet werden.

```
import processing.serial.*;
Serial mySerial;
PrintWriter output;
String messDaten;

void setup() {
  println(Serial.list());
  mySerial = new Serial(this, "COM13", 9600);
  output = createWriter("Messdaten.txt");
  mySerial.bufferUntil('\n');
}
void draw() {
  if (messDaten != null) {
    output.println(messDaten);
  }
}

void serialEvent(Serial mySerial) {
  messDaten = mySerial.readString();
}
```

Abbildung 53: *Programm für das Speichern der Messwerte*

7 Zusammenfassung

In dieser Bachelorthesis wurde die Steuerungssoftware für ein Leistungsmessgerät entwickelt, das aus verschiedenen Komponenten zusammengestellt wurde und an die speziellen Anforderungen des Labors für elektrische Anlagen und Wandler des Departments Medientechnik der HAW Hamburg angepasst wurde. Es wurden in diesem Zusammenhang verschiedene Leistungsmessgeräte auf dem Markt näher betrachtet und begründet, warum im vorliegenden Fall nicht auf eine fertige erhältliche Lösung bzw. ein auf dem Markt angebotenes Messgerät zurückgegriffen werden kann. Mit dem Aufstellen verschiedener Varianten wurden mögliche Ausführungen des Leistungsmessers im Hinblick auf den Funktionsumfang miteinander verglichen. Die beteiligten Komponenten wurden näher betrachtet und die endgültige Version des Leistungsmessers wurde vorgestellt. Hierbei wurde der Funktionsumfang des Messgerätes erklärt. Die erforderlichen Konfigurationen der einzelnen Komponenten und die hierfür benötigte spezielle Software wurden erläutert. Das Programm für die Steuerung des Leistungsmessers wurde in den Hauptbestandteilen vorgestellt und anhand von Codeauszügen erklärt. Die Genauigkeit des Leistungsmessers wurde mit Hilfe von Testmessungen überprüft und bewertet.

8 Schlussfolgerung / Ausblick

Messungen haben ergeben, dass der Leistungsmesser im Vergleich mit auf dem Markt erhältlichen Messgeräten der höheren Preisklasse gute Messergebnisse liefert. Im 230 V – Bereich ergeben sich bei den Messwerten nur geringe Abweichungen (kleiner 0,5 %). Diese nur sehr kleinen Abweichungen zeigen, dass die Konstanten für die Berechnungen der Messwerte im Steuerungsprogramm des Leistungsmessers sehr gut bestimmt wurden. Im Spannungsbereich unterhalb von 2 V liegen die Abweichungen in einem Bereich von 5 %. In diesem Bereich könnte eine Anpassung des Messbereichs an den Spannungsbereich mit Hilfe der verbauten Relais bzw. mit den damit umschaltbaren Spannungsteilern vorgenommen werden. Hierdurch würden auch für diesen Messbereich sehr genaue Messwerte ermittelt. Der Funktionsumfang der Menüoberfläche des Touch Displays könnte für zukünftige Messungen noch erweitert werden. Eine Erweiterung wäre z.B. die grafische Darstellung der Messwerte. Hierfür müssten dann aber auch Erweiterungen am Steuerungsprogramm vorgenommen werden. Außerdem könnte die Darstellung des Leistungsfaktors innerhalb der Menüoberfläche erweitert werden. Die aktuelle Menüoberfläche zeigt den Leistungsfaktor als cos(φ) – Wert. Eine Auswahlmöglichkeit innerhalb der Menüoberfläche könnte die Option bieten, den Leistungsfaktor als Phasenwinkel, also in Grad anzuzeigen.

9 Literaturverzeichnis

[1] „www.reichelt.de,“ 10 07 2020. [Online]. Available: https://www.reichelt.de/netzanalysator-1-phase-gpm-8213-p266864.html?PROVID=2788&gclid=EAIaIQobChMI2Mvks8fw6wIVT-btCh2_hgXiEAYYASABEgI-y_D_BwE&&r=1. [Zugriff am 10 07 2020].

[2] „www.pce-instruments.com,“ 10 07 2020. [Online]. Available: https://www.pce-instruments.com/deutsch/regeltechnik/leistungsmessgeraet/leistungsmessgeraet-pce-instruments-leistungsmessgeraet-pce-360-det_2198852.htm. [Zugriff am 10 07 2020].

[3] „www.reichelt.de,“ 10 07 2020. [Online]. Available: https://www.reichelt.de/netzanalysator-komplettset-3-phasen-peaktech-4145-p146015.html?PROVID=2788&gclid=EAIaIQobChMI-uaUldLw6wIVVObtCh10igXfEAAYASAAEgJKkPD_BwE&&r=1. [Zugriff am 10 07 2020].

[4] „www.messgeraete-chemnitz.de,“ 10 07 2020. [Online]. Available: https://www.messgeraete-chemnitz.de/messtechnik/netzanalysatoren/1086/fluke-1736/eus-energie-logger-netzanalysator-power-logger-1736?gclid=EAIaIQobChMIlMjuutPw6wIVCuntCh20twfiEAYYASABEgItZvD_BwE. [Zugriff am 10 07 2020].

[5] „http://www.mercateo.com,“ 10 07 2020. [Online]. Available: http://www.mercateo.com/kw/mavowatt(20)30/mavowatt_30.html. [Zugriff am 10 07 2020].

[6] „www.instrumart.com,“ 10 07 2020. [Online]. Available: https://www.instrumart.com/products/43063/fluke-norma-40005000-power-analyzers. [Zugriff am 10 07 2020].

[7] „www.peaktech.de,“ 14 07 2020. [Online]. Available: https://www.peaktech.de/produktdetails/kategorie/leistungs-analysator/produkt/peaktech-4145.html. [Zugriff am 14 07 2020].

[8] „www.fluke.com,“ 14 07 2020. [Online]. Available: https://www.fluke.com/de-de/produkt/elektrische-pruefungen/netzqualitaet/norma-4000. [Zugriff am 14 07 2020].

[9] „store.arduino.cc,“ 17 09 2020. [Online]. Available: https://store.arduino.cc/arduino-mkr-zero-i2s-bus-sd-for-sound-music-digital-audio-data.

[10] „www.techedu.com,“ 14 07 2020. [Online]. Available: https://www.techedu.com/Power-Quality-Analyzers/Fluke-N4K-1PP54/. [Zugriff am 14 07 2020].

[11] „www.antratek.de,“ 14 07 2020. [Online]. Available: https://www.antratek.de/expansion-board-for-nextion-enhanced-

display?gclid=EAIaIQobChMI3diu8Inx6wIVGOJ3Ch1K6AxREAQYASABEgLwdvD_BwE. [Zugriff am 14 07 2020].

[12] „www.real.de,“ 15 07 2020. [Online]. Available: https://www.real.de/product/351867072/?kwd=&source=pla&sid=32844101&gclid=EAIaIQobChMIiZPa0Izx6wIV1entCh0ZTAXhEAQYGSABEgL7NPD_BwE. [Zugriff am 15 07 2020].

[13] „https://funduino.de,“ 15 07 2020. [Online]. Available: https://funduino.de/nr-32-der-rotary-encoder-ky-040.

[14] „https://nextion.tech,“ 16 07 2020. [Online]. Available: https://nextion.tech/editor_guide/.

[15] „https://nextion.tech,“ 16 07 2020. [Online]. Available: https://nextion.tech/instruction-set/.

[16] „https://nextion.tech,“ 16 07 2020. [Online]. Available: https://nextion.tech/datasheets/nx8048k070/.

[17] „www.boecker-systemelektronik.de,“ 16 07 2020. [Online]. Available: https://www.boecker-systemelektronik.de/Seite-/-Kategorie-1/Grafische-Oberflaechen-fuer-Mikrocontroller-Anwendungen-mit-Nextion-Displays.

[18] „https://de.wikipedia.org,“ 20 07 2020. [Online]. Available: https://de.wikipedia.org/wiki/Schutzleiter.

[19] „www.analog.com,“ 20 07 2020. [Online]. Available: https://www.analog.com/media/en/technical-documentation/user-guides/ADE9000-UG-1098.pdf

Margolis, M: Arduino – Kochbuch. 1. Aufl. Beijing: O'Reilly, 2012

10 Abbildungsverzeichnis

11 Tabellenverzeichnis

Eidesstattliche Erklärung

Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangabe kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Hamburg, 25.09.20 [redacted]

Ort, Datum Unterschrift