

Designing Software for User-Friendly Class-S Power Quality Analyzer based on Raspberry Pi

EE3L11: Bachelor Graduation Project

Hüsna Yıldız

Marten de Vries

Emirhan Öztürkoglu



Designing Software for User-Friendly Class-S Power Quality Analyzer based on Raspberry Pi

by

Hüsna Yıldız
Marten de Vries
Emirhan Öztürkoglu

Student Name	Student Number
Hüsna Yıldız	5026938
Marten de Vries	4868625
Emirhan Öztürkoglu	5054133

Internal proposer and group supervisor: Simona Renzaglia
Project supervisors: Prof. Dr. Peter Palensky & Dr. Aleksandra Lekić
Practical supervisor: Ing. Remko Koornneef
Project Duration: September, 2023 - January, 2024
Faculty: Faculty Electrical Engineering, Mathematics and Computer Science
Delft

Preface

This thesis is part of our bachelor thesis at the Faculty of Electrical Engineering of the Technical University of Delft. The term 'Power Quality' never really crossed our minds when thinking about a thesis topic. But when we started to look into it, we found that there were immense possibilities and a lot to discover. Writing this thesis made us think about the fun, but also the challenging times during this project. The countless hours of research and coding sometimes seemed endless and boring. But we are delighted and proud of our final result.

Before you read this thesis, we would like to thank some people who played a big part in this thesis project. First of all, we would like to thank Eline, Jorik and Luke, the members of the hardware group, for their hard work and dedication. We had a lot of fun working together and learned a lot from each other. We would like to thank Dr. I.E. Lager for his guidance during the bachelor and graduation process. We would like to thank Dr. A. Lekić for her incredible help, ability to solve problems and the time she spent with us. We would also like to thank Ing. R Koornneef, who insists we call him 'Remko', for his guidance and recommendations over the past few weeks. Finally, we would like to thank Ms. S. Renzaglia for her input, feedback and countless hours of hands-on help during the project. We faced many challenges during the thesis project, and the guidance of the aforementioned individuals was truly our signal in the noise.

Hüsna Yıldız
Marten de Vries
Emirhan Öztürkoglu
Delft, January 2024

Abstract

This thesis describes the design of a low-cost class-S power quality analyzer based on a Raspberry Pi 4. Capable of detecting power frequency, magnitude, voltage dips and swells, harmonics, and total harmonic distortion, the project consists of four main components: the communication protocol, the interface, the algorithms for power quality parameters, and the database. Each with its subdivisions. The communication module, which uses the I2C protocol, was chosen for its high sampling rates and built-in acknowledgement system, ensuring robust and fast operation. The interface module features a secure login process with two-step verification, a graphical user interface for real-time monitoring, and integration with algorithms for parameter calculation. The algorithm module includes Fast Fourier Transform for harmonic detection, zero crossing method for power factor and frequency, and peak detection for voltage dips and swells. The database, powered by MariaDB on Raspberry Pi 4, securely manages the received data with restricted access for increased security, allowing remote access only from specified IP addresses.

Contents

Preface	i
Abstract	ii
Nomenclature	1
1 Introduction	2
1.1 Problem Description	2
1.2 State-of-the-art Analysis	2
1.3 Overview of the Project	3
1.3.1 General Strategy and Approach	3
1.3.2 Task Division	3
1.3.3 Structure of thesis	3
2 Program of Requirements	5
2.1 Overall project requirements	5
2.1.1 System requirements	5
2.2 Functional requirements	6
2.2.1 Data storage	6
2.2.2 Data Communication protocol	6
2.2.3 Power quality parameter calculators	6
2.2.4 Interface and Authentication system	7
3 Design	8
3.1 Data Communication Protocol	8
3.2 Data Storage Unit	9
3.3 Interface	9
3.4 Power quality parameter calculators	9
3.5 Integration	9
3.6 Monitoring speed	9
4 Data Communication Protocol	10
4.0.1 Consideration between UART, SPI and I2C protocols.	10
4.1 Inter-Integrated Circuit Data communication protocol	11
4.2 I2C implementation	12
4.2.1 Preparations	12
4.2.2 PiGPIOd	13
4.2.3 ADC modes	13
4.3 Binary to numerical conversion	14
4.4 I2C performance	14
5 Database	16
5.1 Database and connection	16
5.1.1 Database	16
5.2 Security Layers and Accessibility	16
5.2.1 Physical Security	17
5.2.2 User Privileges	17
5.2.3 Encryption	17
5.2.4 Accessibility	18
5.2.5 Remote access using a VPN server	19
5.3 Data Management	19
5.3.1 phpMyAdmin	19

6	Interface	21
6.1	Requirements Considerations	21
6.2	Implementation Considerations	21
6.3	Usage	23
6.4	GUI	23
6.5	Final Product	23
7	Modules	24
7.1	Frequency calculation	24
7.1.1	Importance of frequency deviation	24
7.1.2	Calculations methods	24
7.2	Power Factor	26
7.2.1	Power factor determination	27
7.3	Voltage Sags and Swells	28
7.3.1	V_{RMS} algorithm using live data	29
7.3.2	V_{RMS} algorithm using sampled data	30
7.3.3	Peak detection algorithm	32
7.4	Harmonics	33
7.5	Total Harmonic Distortion	35
8	System Integration and Testing	37
8.1	Raspberry Pi and database Integration	37
8.1.1	Raspberry pi implementations	37
8.1.2	Rasbperry pi testing	37
8.2	Interface Implementations	38
8.2.1	Interface testing	38
9	Risk Assessment	39
9.1	Introduction to Risk Assessment	39
9.2	Scale Determination	39
9.3	Risk Identification	40
9.3.1	Risk Assessment Raspberry Pi	40
9.3.2	Risk Assessment Communication Protocol I2C	40
9.3.3	Risk Assessment Interface Authentication	41
9.4	Risk Analysis	41
10	Conclusion	43
11	Discussion	44
11.1	Requirements fulfillment evaluation	44
11.1.1	Data acquisition	44
11.1.2	Database	44
11.1.3	Calculation modules	44
11.1.4	Interface	44
11.2	Future Improvements	45
11.2.1	Database writing	45
11.2.2	Database security	45
11.2.3	Implementation of physical storage	45
11.2.4	Remote Accessibility	45
	References	46
A	Data Communication Protocol	50
A.1	FULL I2C frame	51
A.2	Clock stretching	52
A.3	Setup different modes	52
A.4	Manual Mode schematic	53
A.5	Binary to numerical conversion	54
B	Sampled plots	55

C	System Integration	58
C.1	User interface after integration with real data	59
C.2	User interface after integration with generated data	61
D	Login screen and GUI	66
E	Database	68
F	Modules	71
G	Tables	78
H	Code	79
H.1	Plotting ADC values	79
H.1.1	Voltage	79
H.1.2	Current	82
H.2	Sampling and database integration	85
H.2.1	Voltage sampling with database integration	85
H.2.2	Current sampling with database integration	88
H.3	Power Quality Parameters	92
H.4	Signal Generator	98
H.5	GUI	100
H.6	GUI for Login Process	113
H.7	Logging in GUI	119
H.8	Account creation GUI	120
H.9	Verification GUI before opening Interface that monitors power quality	122
H.10	Verification GUI before changing password	123
H.11	Enter email GUI before changing password	124
H.12	User changes his password on this GUI	125

Nomenclature

Abbreviations

Abbreviation	Definition
ADC	Analog to digital converter
API	Application Programming Interface
CLK	Serial Clock
CSV	Comma Separated Values
FFT	Fast Fourier Transform
GUI	Graphical User Interface
I2C	Inter-Integrated Circuit
MVC	Model-View-Controller
PQQ	Power Quality Quantizer
RMS	Root Mean Square
SDA	Serial Data
SMTP	Simple Mail Transfer Protocol
SPI	Serial Peripheral Interface
SQL	Structured query language
THD	Total Harmonic Distortion
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
UI	User Interface

Introduction

"Scientia potentia est", as is said by Sir Francis Bacon in his work *Meditationes Sacrae* from 1597. The phrase has the following meaning: Knowledge is power. So what if you know your power? No, not power in the literal sense, but your grid power! Bad power quality can manifest itself in different ways. Knowing your grid power can be achieved by building a power quality analyzer, which is exactly what this thesis is about.

1.1. Problem Description

Power quality is a crucial consideration in modern power systems as it can impact both consumers and utilities. The definition of power quality could be better described by examining power quality disturbances. A power quality disturbance is generally accepted as: 'Any change in the power (voltage, current, or frequency) that deviates from the normal operation of electrical equipment' [1]. There are several phenomena causing disturbances in the quality of power. Examples of these phenomena, defined by IEEE Power and Energy Society, are the following: Harmonics, voltage fluctuations, voltage sags/swells and interruptions, voltage imbalance, spikes, transients, power-frequency variations and low power factor [2]. Power quality analysis is important as these disturbances to power quality can result in damage to electronics, data corruption, flickering lights, increased energy consumption, discomfort, financial losses, and potential safety concerns. Voltage dips, transients, and short interruptions are recognized as primary power quality issues that result in significant financial losses [3]. More and more households are implementing solar photovoltaic (PV) systems to generate renewable energy and be environmentally friendly. The advancement in renewable energy also creates technical challenges that have an impact on the power quality. Voltage and frequency fluctuations and harmonics are power quality issues occurring with the integration of renewable energy [4]. Including the trend in renewable energy sources, it is important to know about power quality. Current power quality analysers that are available on the market are very expensive (1500-3000). Furthermore, the devices may not always be suitable for average consumers who want to know about their power quality.

1.2. State-of-the-art Analysis

There are several power quality analyzers developed in the past years, with each having different purposes. For instance, some devices are developed exclusively for academic laboratory applications [5], whereas some are developed for specific purposes like a power quality assessment tool for an electric motor [6]. There are also power quality analyzers which are used in smart-grid smart home applications, these calculate the cost of electricity in real time [7]. Other power quality analyzers only included the voltage signal as input, not taking the current into account [8] [9]. The installation of the power analyzer mainly has to be done in the electric panel of the house [10] [7]. No study or existing power quality analyzers includes the option to analyse the power consumption of a device by simply connecting the device plug to the socket of the analyzer. The existing power quality analyzers are mainly used for big industry applications using three-phase systems. The analyzers also make use of microcomputers, differing from Raspberry Pi's [8] to NanoPi Neo Air ARM's [10]. There are also some power quality analyzers which do not use any microcomputer [6] [9]. Although some power quality analyzers are

low-cost, they are either not available for everyone [5], or only take the voltage into account [8], or do not have an installation method that anyone can perform without technical knowledge [10]. The aim of this project is to make a combination of the above-named analyzers. The aim of this graduation project is to create a Raspberry Pi-base one-phase power quality analyzer that is able to store data in a remotely accessible database and has a user-friendly interface. The device should have a plug-and-play system and meet class-S requirements. These requirements will be discussed in depth in section 2.

1.3. Overview of the Project

The primary objective of this bachelor thesis is to design, build and test a cost-effective power quality meter capable of detecting the power frequency, power magnitude, voltage sags and swells, harmonics, and total harmonic distortion. The power quality analyzer should be capable of determining the power consumption of other devices, which is made possible by using a socket. Simultaneously, it is desired to have effective communication between the application and the users, achieved through a user-friendly and intuitive graphical user interface. This way anyone should be able to use the interface and have a general understanding of the power quality in their house or small business. Furthermore, it is important to ensure the device is safe and secure enough for users to trust its privacy features.

1.3.1. General Strategy and Approach

To fulfil the specified goals, the 'Power Quality Quantizer' (PQQ) has been developed, built and validated. The PQQ connects to the AC mains through a standard wall outlet. The first step involves the transformation of the voltage signal, achieved through a transformer to reduce the voltage. Subsequently, the signal is filtered and amplified to achieve the desired range before being processed by an analog-to-digital converter (ADC). If the user wants to know the consumption of a device, this device can be plugged into the PQQ. In this configuration, current from the AC mains is measured using a current sensor, and the device's power consumption is monitored. Both the voltage and current sensors are passed into the ADC, and the converted signals are transmitted to the Raspberry Pi, through I2C communication. The Raspberry Pi processes these digital signals and transmits them into the database, which is also established on the Raspberry Pi. The interface, connected to this database presents the data from the current and voltage sensors as well as the quality quantifiers to the users. Moreover, a server is established on the Raspberry Pi to enable remote access from devices on the same local network.

1.3.2. Task Division

Two subgroups were formed in order to realise the above-mentioned objective. Subgroup 1 was responsible for the development of the hardware part, which is the part from the mains input to the ADC. Subgroup 2 had the responsibility for everything between the output of the ADC and the general user interface. This also included the database and the power quality parameter calculation.

Software

This thesis describes the design process, the methods and the results gathered by using them. The software part of the power quality analyzer plays a crucial role in extracting meaningful insight from the captured electrical signals by the hardware. One could also say that the software is the brain of a power quality analyzer, performing tasks such as processing the data obtained from the hardware by means of an Analog-to-Digital converter. As the brains of the analyzer, its primary objective was to implement algorithms that can identify and quantify several quantifiers such as frequency, power factor, voltage sags/swells, harmonics, and total harmonic distortion. The software is also responsible for storing, managing, and visualizing the data. Storing and managing the data is done in a database, whereas the visualization is realized by designing an user friendly interface. The interface provides the means for the users to interact with and understand the data. The software part can be seen as a bridge between the hardware and the user. It determines the product's usability by the designed and developed functions.

1.3.3. Structure of thesis

This report is organized as follows: Chapter 2 covers the specific requirements that guide the design of the Power Quality Analyzer. In chapter 3 the complete system design is given, and also in short how

the requirements described in chapter 2 will be met. The following chapter 4 takes a deep dive into the module responsible for the data transfer between hard- and software. The next chapter 5 describes several aspects regarding the storage and management of the data acquired. Chapter 6 describes the interface design process in detail, also outlining the login process. The next chapter 7 delves into the designed algorithms that are performing power quality parameter calculations. The power quality parameters included in this chapter are the following: Frequency calculation, Power factor calculation, Voltage sags and swells detection, Harmonics and the total Harmonic Distortion. Chapter 8 highlights the crucial phase of testing and validation. It details the methodologies, scenarios, and criteria employed to ensure the reliability and functionality of the developed system. The following chapter 9, explains the likelihood of potential risks that can occur and the impact if they occur, by the power quality quantizer. The conclusion drawn from the research will be served in chapter 10. The last chapter 11 opens avenues for further exploration, reflection, and future development. It provides a platform for discussing challenges encountered, lessons learned, and potential enhancements or extensions to the power quality analyzer.

2

Program of Requirements

A power quality analyzer based on Raspberry Pi was developed during the Bachelor's Graduation Project. Subgroup 2 primarily worked on the software part of the project. The main objective is to read out the ADC, process the signal coming into the Raspberry Pi, and show the results of the processed signal on the graphical user interface. The goal is to extract the power quality parameters and determine the power quality of the incoming signal. The data obtained from the ADC is stored in a database, and specific algorithms have been written for signal processing to quantify the power quality. The results of these algorithms are also stored in the database. An interface is implemented to display the results of the algorithms to the user. This section outlines the requirements for the software part of the project.

2.1. Overall project requirements

This bachelor's project aims to develop a reliable and inexpensive solution for measuring the power quality of a network. The product must be able to measure power frequency, power magnitude, voltage dips and swells, harmonics and total harmonic distortion. It must be an embedded device with a single board computer suitable for homes (single phase) and small businesses. The consumer must be able to view real-time data and access stored data on power quality anomalies to gain an understanding of the power quality in their homes or small businesses. PQQ helps the user to gain insight into their power quality. This product does not improve the power quality nor does it compensate for the distortions. It simply collects data on the power quality parameter and displays it to the consumer.

2.1.1. System requirements

1. The system must be able to measure the power consumption of at least 1 connected device.
2. The system must be capable of separately and accurately recording the voltage variations over time, regardless of whether the connected devices are consuming a high or low amount of electrical power.
3. The cost of assembling the entire system must be within the range of 250.
4. The system must present waveform data and update every second to monitor the power quality.
5. The system must have the capability to store a full month's worth of power quality data.
6. The design of the system must be modular. The system components must be organized into distinct, interchangeable modules, and each module must be replaceable without affecting the functionality of the entire system.
7. The embedded device for the system must be based on a Raspberry Pi 4.
8. The system must meet the specifications of a Class S power quality meter, see table 2.1 for the specifications.

Table 2.1: Class S requirements according to IEC-61000-4-30 [IEC-61000-4-30]

Parameter	Uncertainty	Measuring range
Frequency	± 50 mHz	42.5 Hz to 57.5 Hz
Magnitude of the supply	$\pm 0.5 \% U_{din}$	20 % to 120 % U_{din}
Dips and swells	Amplitude $\pm 1\% U_{din}$ Duration ± 1 cycle or ± 2 cycles	N/A
Unbalance	$\pm 0.3\%$	1% to 5% of u_2 (if implemented same for u_0)
Voltage harmonics	200% of IEC 61000-4-7 Class 2	10% to 100% of Class 3 of IEC61000-2-4
Current	$\pm 2\%$	10% FS to 100% FS

2.2. Functional requirements

2.2.1. Data storage

The aim is to have a data storage system, that is safe, fast and big enough for this project. It should be well designed, with a clear overview.

Functional requirements Database

1. The database must store data in a structured and organized manner.
2. The system must allow the modules to input new data into the database.
3. The database has to be secure.
4. The user must be able to access the database locally and remotely.
5. The system must support efficient and accurate retrieval of data.

2.2.2. Data Communication protocol

This module aims to have a fast and reliable data transfer between the ADC and the Raspberry Pi. This module must give a good representation of the physical waveforms as this is the bridge between hardware and software.

1. Sampling must be done continuously.
2. The sampling must be done at a sampling rate above 10kHz.
3. The data communication protocol must include robust mechanisms to mitigate the effects of noise and ensure the resilience of the system. Specifically, it must be designed to handle potential errors, such as bad bits, without causing the entire system to fail.
4. The data communication protocol must be able to sample voltage and current at the same time.
5. The module must convert the binary data into numerical values accurately.

2.2.3. Power quality parameter calculators

The power quality parameters are the heart of this project. These values are what determine if the power quality is good or bad. These should have well-defined requirements.

1. The parameter calculators must calculate the chosen class S parameters.
2. The parameter calculators must meet the class S accuracy
3. The parameter calculators must be able to calculate the required parameter within the time instances given before it should be displayed on the interface.
4. The system must detect and analyse harmonics in the power signal, providing detailed information on harmonics content of the current and voltage signal.
5. The analyzer must calculate and display the Total Harmonic Distortion (THD) of the current and voltage signal, indicating the overall level of harmonic distortion.
6. The analyzer must monitor and display the frequency of the voltage signal, and track accurately the variations of frequency in decimal.

7. The analyzer must calculate the power factor of the electrical system, providing information on the phase relationship between voltage and current signals.
8. Implement an algorithm to detect and report voltage sags and swells, it must indicate the encountered deviations from the normal voltage levels based on IEEE Std 1159-2019.
9. Users must have the capability to remotely access and control the power quality analyzer.

2.2.4. Interface and Authentication system

The aim is to have a secure and user-friendly interface. For the security layer, a user authentication system is developed. The purpose of this document is to outline the functional and non-functional requirements of the interface and the user authentication system.

Functional requirements

Overall Functional requirements GUI

1. Programming language that suits the developers the most must be used for interface development.
2. To ensure broad accessibility and user reach, the system must be designed for cross-platform compatibility.
3. A basic layer of security must be implemented
4. Two-step verification must be implemented to increase security.
5. The interface must be easy to use and it must have a professional look.

Functional requirements login process

1. Users must make an account to make use of our application, for this, the user provides a valid and unique email and password which are stored in our database.
2. Passwords must be hidden during typing to prevent visibility.
3. Users must log in before having access to the interface.
4. The user is given an option to go back to the login screen during the login process.
5. A two-step verification must be implemented when logging in or changing the password.
6. Users can change their password if forgotten or for other reasons.
7. Only a valid email address known in the database receives a verification code if the user creates an account or wants to change the password, to ensure authorized access.
8. After the user enters the right verification code, users can enter the application or change the password.
9. The user must be able to change the password, this must be updated in the database,

Functional requirements Interface

1. The interface has an intuitive navigation control, which allows the user to open different pages easily to different pages of the interface.
2. Every page must contain relevant information about the power quality parameters.
3. The user must be able to see the current and voltage signals that are updated every second.
4. The user must see the numerical values for power factor, power magnitude, voltage sags and swells, VRMS, total harmonic distortion, and frequency deviation. Those parameters are very second updated.
5. An information page must be implemented that gives a summary of the application and who to contact when they encounter problems with the application.
6. The graphs must be updated every second.

3

Design

To achieve the desired requirements stated in chapter 2, a complete system design has been made. This design had to be robust, accurate, and reliable. This design aims to ensure that the system can withstand varying conditions, deliver accurate and precise results and operate consistently with a high level of reliability. The current and voltage signals are sampled using ADC (analogue to digital converter). The digital values obtained from the ADC represent the sampled current and voltage signals and those values are stored in a database. These values are processed by the power quality parameter calculators and the results of those algorithms are displayed on the interface. The input must be fetched continuously from the ADC, and the output must display the calculated values with the least delay of seconds. Between these in- and outputs, the calculations and signal processing must be done with the accuracy of class-S analyzers. This will be met by having a sampling rate of at least 10kHz and robust algorithms. The interface shows the live data and the power quality parameters clearly and simplistically. Figure 3.1 shows a schematic overview of the system.

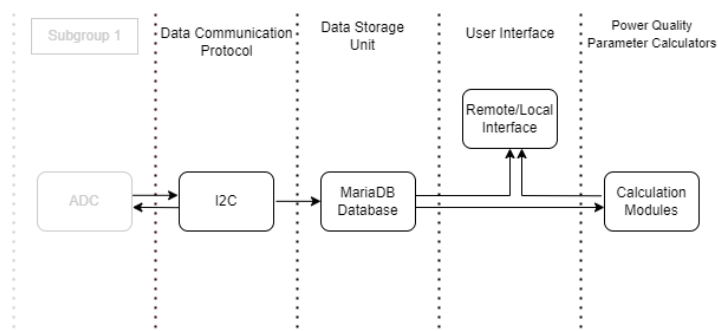


Figure 3.1: A schematic overview of the design

As shown in figure 3.1, there are a few different modules involved in this system. First, the ADC will communicate with a Raspberry Pi via a data communication protocol. This needs to be done at a high enough sampling rate and with as much resolution as possible. This data is then stored in a database running on the Raspberry Pi and accessible remotely over the same network. This data can be retrieved by the various parameter modules. The parameter modules calculate the power quality parameters with the fetched data and write the results back to the database. When each parameter is ready, all the information must be sent to the interface to be displayed to the user. This chapter gives a brief description of the different modules. Later in this thesis the modules will be explained in more detail.

3.1. Data Communication Protocol

The data communication protocol collects samples from the ADC. It must be a reliable and accurate data acquisition with as few errors as possible. The data communication protocol should be compatible

with the selected ADC and Raspberry Pi. The chosen data communication protocol is the I2C (Inter-Integrated Communication) protocol. This protocol was chosen because it has a high enough theoretical sampling rate for this use case and has a built-in acknowledgement system as an additional layer of robustness. This module will be discussed further in chapter 4.

3.2. Data Storage Unit

To share data between the different modules, a data storage unit must be designed. This could be in the form of a CSV (Comma-Separated Values) file, multiple arrays within the code, or an accessible database. The design choice is made to use a database running on MariaDB. This choice is made because creating a database would provide a clean and organized place to store large amounts of data. The database can be programmed using SQL (Structured Query Language), which is accessible in Python and C code. MariaDB is accessible and widely used on Raspberry Pi. Also, this database can be accessed remotely on the same local network. This makes it a suitable solution for this use case. This module is discussed in detail in the 5 chapter.

3.3. Interface

Since this power quality analyzer is designed for homeowners and small businesses, it is critical to design a way for any user to understand the quality of their power. This will be done by implementing a remote user interface. The interface will be written in Python using the PyQt5 extension module. This user interface must be easy to use and display the different power quality parameters. It will also have the ability to display the raw voltage and current data. The interface needs to be accessible only to known users. To achieve this, a login screen will be implemented with an email and password login that uses a verification method to verify the user. In this way, a layer of security is created between the data and the interface. This module will be discussed in detail in chapter 6.

3.4. Power quality parameter calculators

To give an accurate description of one's power quality, the power quality parameters should be calculated and updated every second. To achieve this, different power quality calculation modules will be built. The data will be fetched from the database and the calculation of every parameter will be done. These calculations will be done on the remote device the interface is running on. This was because the calculation modules don't take processing power from the Raspberry Pi which is already running the database and fetching the data. These parameter calculators must be efficient to at least match the speed of the incoming data and have accurate outputs. These parameters will be discussed in depth in chapter 7.

3.5. Integration

To give an accurate description of the power quality, the power quality parameters must be calculated and updated every second. To achieve this, several power quality calculation modules are built. The data is retrieved from the database and the calculation of each parameter is performed. These calculations are done on the remote device where the interface is running. This is because the calculation modules don't take any processing power away from the Raspberry Pi, which is already running the database and fetching the data. These parameter calculators must be efficient to at least match the speed of the incoming data and have accurate outputs. These parameters are discussed in detail in the 7 chapter.

3.6. Monitoring speed

One of the requirements is to update the PQQ analyzer values every second. This, rather than achieving true real-time monitoring, is a practical balance. This choice provides a trade-off between data accuracy and computational efficiency, allowing for more accurate insights without overtaxing system resources. By choosing to update every second, the system efficiently manages database loads and provides a dynamic and responsive user experience. This choice recognizes the challenges of continuous real-time monitoring while maintaining a consistent and practical approach to meet user expectations and application needs.

4

Data Communication Protocol

The data communication protocol is a crucial part of the systems as it acts as a bridge between hardware and software. To transfer the ADCs data, a data communication protocol must be established. This communication protocol must be able to sample continuously. It must be fast enough to gather at 10kHz. It should include robustness and lastly, it must be compatible with the Raspberry Pi and be able to sample from two devices at the same time. Three well-known protocols that can be used in this use case as they are supported by the Raspberry Pi are 'UART', 'SPI', and 'I2C'. All of these have their advantages and disadvantages, which we will briefly discuss. Given the functional requirements listed in 2.2.2, we will choose the appropriate communication protocol.

4.0.1. Consideration between UART, SPI and I2C protocols.

UART (Universal Asynchronous Receiver/Transmitter) is a two-wire (Transmit line and Receive line) data communication protocol that is widely used, and has asynchronous communication options with half- and full-duplex capabilities [11] [12] [13]. However, the main disadvantages are that it has synchronization issues, is unsuitable for high-speed applications and has no built-in error check [14]. This makes it not suitable for our use case.

Serial peripheral interface, or 'SPI', uses 4 wires; Master out Slave in, Master in Slave out, Serial clock, Slave select/Chip select. The disadvantages of this protocol are that it has no built-in error checking, it could have complex protocol configurations, and it has the highest power consumption of the three. Some advantages are its high data transfer speed, full duplex capabilities, and synchronous characteristics [15] [16] [17].

The 'I2C' (Inter-Integrated Circuit) protocol is a two-wire 'SDA' (Serial Data) and 'SCL' (Serial Clock) data communication protocol with high-speed data transfer, bi-directional communication with a built-in acknowledgement mechanism. The disadvantages are reported troubles with clock synchronization and the possibility of complex protocols [18] [19] [20].

The most important factors for this specific case are high speed and reliable data communication. Since UART is typically used at lower speeds and has no built-in error-checking mechanism, it is the first to be eliminated from the design considerations. SPI achieves the highest sampling rate but has no built-in error detection, and I2C has a lower sampling rate than SPI but is high enough for this use case and has a built-in acknowledgement system. So the SPI and I2C options would be suitable for this project. Finally, in collaboration with the hardware group, an ADC using the I2C protocol was selected. This ADC was chosen for its accuracy and speed.

4.1. Inter-Integrated Circuit Data communication protocol

The I2C data communication protocol is a two-wire protocol that can achieve data transfer using a clock line and a bidirectional data line in a master-slave architecture. The clock is generated by the master; thus, the master, mostly, chooses the data transfer speed. There is a way for the slave to reduce the transfer speed, this will be discussed at the end of the section. The communication is done via a so-called 'bus' via serial data transfer. This bus can be used by multiple slaves as these can be distinguished by using different slave addresses, but as this is serial transfer, this limits the amount of data transferred per device. This protocol has a built-in acknowledgement system, which is crucial in communication. After every data byte, the receiver will send an acknowledgement or Not acknowledged bit.

This acknowledgement bit can also be sent to indicate the stopping of sending data. This gives the acknowledgement bit an important role for this protocol. It is critical to note that these bits only indicate if the full data byte is transferred; this will be a strong foundation for error detection, but it does not provide certainty that the received byte is the actual data byte. In certain cases, a third wire can be used in this protocol as the Busy/Not Ready pin, which will be pulled high while data is being transferred and pulled low when the device is idling.

Now that all wires are explained, how does the protocol transmit data? Figure 4.1 shows the basic functionalities of the protocol. The CLK line is a clock line that has a pre-determined baud rate. This baud rate is set by the master, which, in this case, is the Raspberry Pi. The SDA holds the binary data. The SDA line is sampled at every rising edge of the CLK line. Before every I2C frame, a 'start' signal must be given. Similarly, a 'stop' signal must be given after every end of an I2C frame.

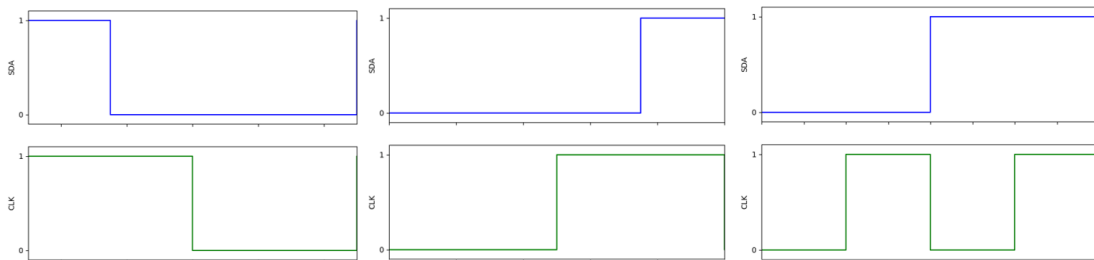


Figure 4.1: Different fundamentals of the I2C protocol. Start condition: right, Stop condition: Middle, Writing a zero and then a 1: left

Now, before the intricacies of the I2C communication protocol can be discussed, the caveat that was skipped over at the beginning of this section can be discussed. The baud rate, as said before, is set by the master. However, when the master wants to read something from the slave, the slave can stall the clock given by the master and have more time to prepare the data. The slave is allowed to hold down the clock to stall the rising edge and thus the reading. The master is required to read back the clock signal after releasing it to the high state and wait until the line has gone high. This phenomenon is called clock stretching and is used by the slave when the sampling speed set by the master is too high. Figure A.2 shows the clock stretching phenomenon between 15 and 25 ms. The clock should be high but is kept low by the slave. This behaviour can be problematic when the master does not detect the clock stretching as the master will continue with its functions while the slave is lagging. The Raspberry Pi 4 does have functionalities to deal with the clock stretching but these have been reported to not be reliable so it's important to stay under the limitations of the ADC.

Now that it's established how the I2C communication protocol interacts, an implementation can be designed for the desired use case. This particular ADC has its specific I2C communication that has to be set up by providing the correct I2C frames. The specific ADC is made for its educational board. Thus, the driver package provided is not suitable for this specific use. Thus, a separate way to communicate had to be made. Luckily, an I2C package that uses C code and specifies exactly what I2C bits it sends can be found. These bits can be matched up with the device's datasheet [21]. Two important steps are to read and write to a specific register. Using the datasheet, a specific set of instructions can be given to

write to a register; these instructions are shown in figures 4.2. They include Sending a start condition, providing the 7-bit slave device address, and providing a write bit. If everything went as expected, the slave would provide a bit of acknowledgement. After this, the master can provide an opcode that tells the slave what operation it wants to do. This is then provided again by acknowledgement. Now, the master should tell the slave what register it wants to write to; after this is acknowledged, the master should provide the data it wants to write to the register. This will be followed by an acknowledgement from the slave; whereafter, the master can send a stop communication.

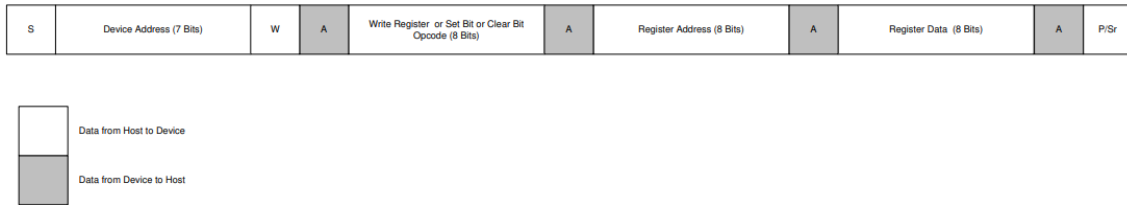


Figure 4.2: The chosen ADC I2C frame for writing a single register

A similar procedure happens when reading a single register. This is shown in figure 4.3. The master starts by writing the start, followed by the slave device address and a write bit. The slave will provide an acknowledgement bit. Now, the master can send the opcode for a single register read. When this is acknowledged, the master can send the register it wants to read the contents of. After another acknowledgement, the master should send a stop bit and start a new I2C frame. This starts as usual with the start bit and the device address, but this time, it sends a read bit, and the slave will acknowledge and provide the single register's content. The master can send an acknowledgement followed by a stop condition.



Figure 4.3: The chosen ADC I2C frame for reading a single register

Appendix A.1 shows a written-out I2C frame that was used in the project.

4.2. I2C implementation

Now that it is known how I2C works and how it should be used to communicate with the ADC. During the design and implementation of the I2C protocol, multiple challenges were met which needed their own deviations and solutions. The chosen ADC works with I2C but its source code is made for its own educational board. As this board has its own layout and coding program the supplied source code can not be used on the Raspberry Pi. Thus the programming of the ADC has to be done from scratch. To do this C-code is used as the first iteration with Python was too slow for this use case. Since Python is an interpreted language it needs multiple steps before the code can be run. C code can immediately be run by the CPU and thus is much faster.

4.2.1. Preparations

To use the GPIO pins on the Raspberry Pi for I2C communication the configuration has to be set for the correct SDA and SCL lines. Additionally, as the project is going to use two ADCs, two different buses should be made so they can communicate independently. For this, it's crucial that both ADCs have different addresses. Two busses are created; bus 1 connected to ADC with address 0x18 and bus 4 connected to ADC with address 0x1f. Now that the Raspberry Pi can see the connected ADCs the communication could begin.

4.2.2. PiGPIOD

To communicate with the ADC an open source package was found that could help with the communication via C code. This was the PiGPIOD package [22]. PiGPIOD stands for Raspberry Pi GPIO daemon. A daemon is a general service that runs in the background of the Raspberry Pi. It's a package of functions that can be used when the daemon program is running in the background. As this daemon has the I2C functionalities with similar I2C frames as described in 4.1 it is suitable for this use case. This daemon has to be initialized at the start of your use in the program and has to be closed at the end. One of the main reasons for the choice of this daemon is its detailed description. The functions in the daemon are explained clearly and the exact bits are described so that they can be matched up with the ADCs datasheet.

4.2.3. ADC modes

Using the ADC datasheet [21], the device can be set to different modes by toggling on and off different registers and writing different values in them. For this use case, the device is set to 'Manual Mode'. One of the considerations made was to use High precision mode (HP-mode). This mode allowed the ADC to use its internal accumulator to accumulate 16 12-bit words into one 16-bit word. This would increase the accuracy significantly but would slow down the sampling rate by 16 times. As the (not overclocked) Raspberry Pi can achieve a baud rate of about 400K bits per second and a perfect data transfer would take about 18 (16 data bits and two acknowledgement bits) bits the maximum achievable sampling rate would be about 1.4k. This would be too low for this use case and the added accuracy would have no effect. Even after overclocking the Raspberry Pi to a 1.000K baud rate, which would increase the possibility of clock synchronization errors, the maximum achievable sampling rate would be 3.5k. Again, too low.

Thus the decision is made to use the manual mode of the ADC. This manual mode is suitable for this use case as it allows the external processor, the Raspberry Pi, to directly request and control when and how the data is being sampled. The first program, for the voltage sampling, was built using the 'single channel single-ended' mode as only one signal was coming into the ADC and this should be referenced against the ground.

The Manual mode is set up by following the provided instructions given in the datasheet. A schematic overview of this is shown in appendix A.3. When the device is set to manual mode, which can be checked by looking at the operational mode status register, the device can sample data using specific instructions in a loop. A schematic overview is given in appendix A.4.

Now, using these two I2Cframes described in 4.1, the ADC can be set in the correct mode and is primed for use. The bus should be read out continuously as the ADC converts the analogue signal into digital values. This is done by using a third I2C frame where the master starts the communication with a start bit, followed by the device address and a read bit; after this is acknowledged by the slave, it will send the first byte, if received correctly the master will send an acknowledgement. This will continue until the set amount of bytes is sent. When this is the case, the master will send an acknowledge bit followed by a stop bit. A schematic overview is shown in 4.4.

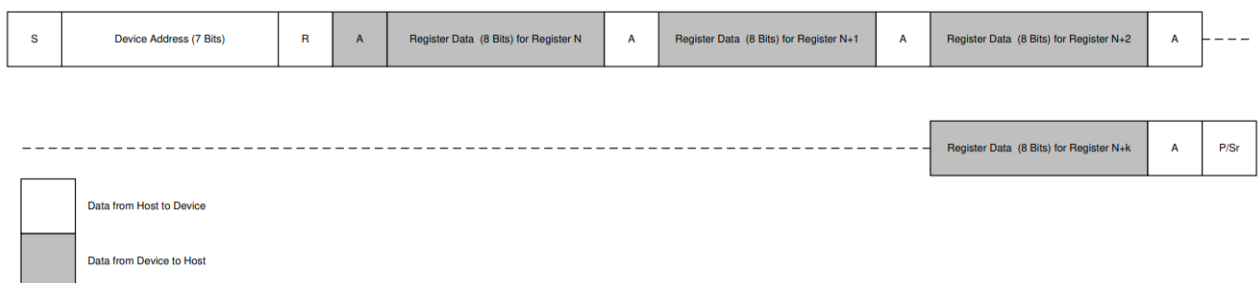


Figure 4.4: The chosen ADC I2Cframe for reading continuously

As the design used two ADCs where one samples the voltage and the other the current this process

had to be done twice. However for the current the hardware group built two circuits where one was more accurate at low current and the second one at high current. Both these signals have to be read continuously and depending on the amplitude of the current, one of the two will be displayed in the interface. To achieve this, the second ADC was set to 'two-channel single ended' mode. This meant that first the most- and least significant byte of channel 0 were sampled, and then the most- and least significant byte of channel 1. While increasing the accuracy because of the switching in hardware, the sampling rate was halved.

4.3. Binary to numerical conversion

As the ADC is a 12-bit ADC but it sends information by bytes there will be 4 bits with information that is not regarded. The value for these bits can be changed by writing a byte in the 'DOUT FORMAT CFG' register in the ADC. An overview of the different configurations is shown in 4.1 During the

Table 4.1: Overview of possible output configurations

Value (hexadecimal)	Configuration
0x00	12-bit conversion followed by 0000 (b) (default)
0x01	12-bit conversion result followed by 3-bit channel ID
0x02	12-bit conversion result followed by 3-bit channel ID followed by DATA VALID bit
0x03	12-bit conversion result followed by 0000 (b)

designing of the communication, this was set to 0x02 to have access to the 'DATA VALID' bit. This bit would be 0 in the following two cases:

1. If the entry in the data buffer is not filled after the last start of the sequence
2. If the I2Cmaster tries to read more than 16 entries from the data buffer

This was an important tool while debugging to check the correct use of the daemon and the communication with the ADC. For ease, after debugging, this has been set to the default value of 0x00.

The most significant byte will be sent first, after an acknowledgement the least significant byte will be sent. These two will have to be concatenated and the 4 least significant bits will have to be deleted. As the ADC is a linear quantizer the converted number can be divided by the maximum number of a 12-bit signal. Then this value, which is always less than or equal to 1, can be multiplied with the maximum input range. This will give a numerical value of the sample. A schematic overview of the calculation is shown in figure A.5.

4.4. I2C performance

The program has been written and the I2C connection has been tested. The results have been plotted in figure 4.5. As can be seen, 21 periods have been sampled in 3200 sampled values. This means the program has sampled 6400 binary numbers. Equations 4.1 and 4.2 show a theoretical estimation of the achievable sampling rate. Note that the baud rate of the Raspberry Pi was set to 400000.

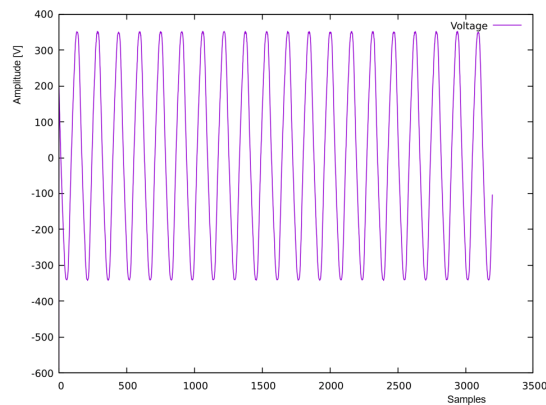


Figure 4.5: The voltage graph

$$n_{samples}/s = \frac{f_{grid}}{n_{periods}} \cdot n_{samples} \quad (4.1)$$

$$\frac{50}{21} \cdot 6400 = 15.2 \cdot 10^3 \quad (4.2)$$

3200 Samples was not an arbitrary number, rather this was done because when reading out the ADC for more than 8k samples (4k values) the ADC would act unpredictably. This behaviour is shown in figure 4.6. Not that the offset is wrong as it's calculated with the mean of all samples. And ass 2k samples are mostly at the lower range this offset is calculated wrong.

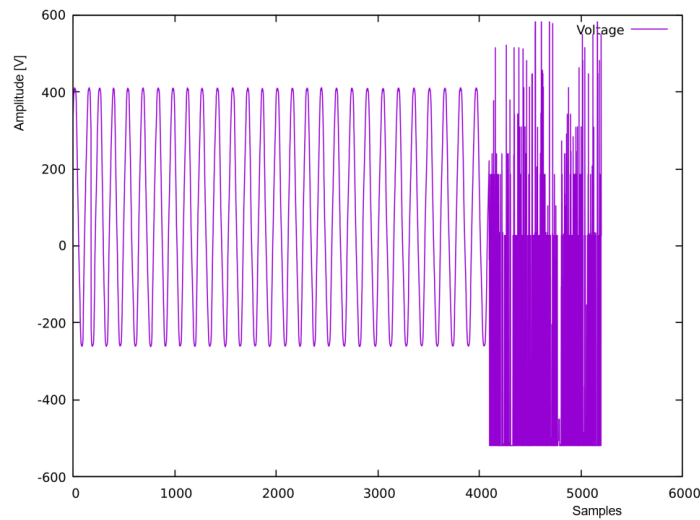


Figure 4.6: The voltage graph with more than 4k samples

The same program is built for the current ADC. These figures are shown in Appendix B.

5

Database

5.1. Database and connection

The database plays a critical role in structuring and organizing data. It must facilitate user input to add new information, implement a secure communication protocol, provide both local and remote access, and support efficient data retrieval to ensure a comprehensive and reliable data management system. These criteria underscore the importance of a robust and accessible database in achieving the project's goals. This section outlines the choices that were made and the decisions that were made during the design process of the database on Raspberry Pi.

5.1.1. Database

Several database management systems can be installed on Raspberry Pi. One is SQLite, a lightweight and embedded relational database that works well for smaller applications and projects. However, it does not support client-server network connections, which means it is not remotely accessible, a requirement for this project. Another option is PostgreSQL, an open source object-relational database. It is often used for applications that need to handle large amounts of data. However, it is not recommended for smaller projects such as a power quality analyzer. Another option considered was the NoSQL database MongoDB, a document-oriented database known for its flexibility and scalability. However, the lack of a SQL language in the database made MongoDB unsuitable for this project. Since the group lacked experience in creating and managing databases, it would not be helpful to choose a database that is not widely used. Therefore, MariaDB was chosen. The following section explains why MariaDB was chosen.

MariaDB was chosen over other options for several reasons. MariaDB is designed to be lightweight and resource efficient, which makes it well suited for running on the Raspberry Pi. Since the communication protocol also runs on the Raspberry Pi, which is heavy, it was necessary to have a database software that was resource efficient. The second reason for using MariaDB is that it is an open source database management system using MySQL, which means it is available to everyone. This statement helps to fulfill one of the requirements of the project, which is to be a low-cost power quality analyzer. The third reason is that MariaDB's compatibility with time series data is advantageous for efficiently handling the chronological data points inherent in power quality measurements. The strong community support and extensive documentation that comes with MariaDB are also reasons to choose MariaDB. MariaDB's security features, such as SSL/TLS encryption and user access controls, ensure that power quality data is protected from unauthorized access. Compatibility with Python and C, which were used as programming languages during the project, was also a reason to choose MariaDB. Finally, the scalability of MariaDB allows for the potential growth of data sets over time. During the literature review conducted prior to developing the database, it was observed that most articles developing similar products used MariaDB [23] [24] [25].

5.2. Security Layers and Accessibility

One of the requirements of the database is to have a secure database. Securing a MariaDB database running on a Raspberry Pi involves implementing several layers of security to protect the data and the

database system. The different layers of security that were implemented are the following: physical security, Raspberry Pi password, user privileges, and firewall configuration. Some options could not be applied, such as VPN connection and encryption.

5.2.1. Physical Security

The user is responsible for the physical security aspect. It is important to protect Raspberry Pi from unauthorized access and interference. To limit physical access to the device, it should be stored in a secure and controlled environment. Enforcing proper physical security helps prevent potential threats from unauthorized individuals having direct physical contact with the Raspberry Pi, reducing the risk of tampering, data theft, or any malicious activity that could compromise the integrity and confidentiality of the system. Customers can increase the overall resilience of their Raspberry Pi (and therefore the database) to threats.

5.2.2. User Privileges

User privileges in the database are critical to maintaining a secure and well-controlled environment. It is very important to give the database users as few privileges as possible. Since in this project there will be only one user at a time, user privileges do not need to be considered. But if the owner wants to add more users, it is possible to give those users the least amount of privileges possible.

Privileges are permissions that allow or restrict the ability of other users to perform certain operations on database objects such as tables, views, procedures, and more. This also means that it is better to avoid using root users for routine tasks and instead create separate user accounts with specific and limited privileges. Since there is also no limit to the number of databases, this would increase the security of the MariaDB server by creating different databases for storing different data. Using this method, organizations or home users could prevent data leakage and reduce the risk of unauthorized access to the server (to all data). This can be achieved by ensuring that each user has the essential permissions required for their specific use of the case.

5.2.3. Encryption

Securing sensitive data is critical in modern database management, and MariaDB provides two types of encryption methods to secure data at different stages. These encryption methods play a critical role in protecting data-at-rest and data-in-transit, ensuring that unauthorized access and potential threats are mitigated. Encryption could not be used in this project because we were using the community version of MariaDB, which does not support encryption plug-ins available only in the enterprise version. It was also not possible to obtain the Enterprise version of MariaDB through the university. However, since security is a part of this project, the implementation of encryption will be discussed.

Data-at-Rest Encryption

Data-at-rest encryption is a critical security feature in MariaDB that protects stored data from being accessed by unauthorized users, ensuring the privacy and security of sensitive information when it is not actively in use. This encryption method is particularly relevant in scenarios where the physical security of storage media, such as hard disks or solid-state drives, cannot be guaranteed. MariaDB provides strong mechanisms for implementing data-at-rest encryption at various levels within the database.

MariaDB's data-at-rest encryption requires the use of a key management and encryption plugin. These plugins are responsible for managing encryption keys and encrypting and decrypting data. MariaDB supports multiple encryption keys, each identified by a 32-bit integer. If the plugin supports key rotation, encryption keys can be rotated to create a new version of the key. The key management options supported by MariaDB are: the file key management plugin shipped with MariaDB, the AWS key management plugin, and the eperi Gateway for Databases [26].

MariaDB has two possible storage engines that can be used for data-at-rest encryption. One is the InnoDB storage engine, which supports encryption for tables and is also the default engine for MariaDB. When InnoDB encryption is enabled, it encrypts data as it is written to the server and decrypts data as it is read from the file system[27]. The other engine option is the Aria storage engine. MariaDB provides data encryption for both user-created tables and internal on-disk temporary tables using the

Aria storage engine, ensuring exclusive access through MariaDB [28].

Before knowing that key management and InnoDB encryption were only available in the Enterprise edition, an attempt was made to install the File Key Management plugin. The File Key Management plugin was a simple and easy-to-install third-party tool that was ideal for this project. It was able to generate random keys in hexadecimal and store them in an encrypted file. However, due to the unavailability of the File Key Management plugin and the InnoDB encryption engine, it could not be developed further.

Encrypting Data in Transit

By default, MariaDB transfers data between the server and clients without using encryption, which is considered acceptable in scenarios where the server and client are on the same host or within secure networks. However, when the server and client are on different networks or in environments with heightened security risks, the lack of encryption raises security concerns. To maintain confidentiality and integrity, it is critical to secure data in transit through encryption. Encryption minimizes the risk of interception, leakage, or unauthorized access.

To address this issue, MariaDB enables encryption of data in transit between the server and clients by implementing the Transport Layer Security (TLS) protocol, which is supported by MariaDB. Certificate verification is how TLS authenticates its connections by verifying that it is talking to who it says it is [29]. VPN also uses the TLS protocol, so creating a VPN connection ensures a private and secure connection when accessing the Raspberry Pi over the public network.

The database is set up to be remotely accessible on the same network and is not accessible via the web. Therefore, there was no need to set up data-in-transit encryption because no data is sent over the Internet and the local network (university network) is secure enough. Data-in-transit encryption could be considered as a security layer for further expansion of the project in the future, when web access may be developed.

5.2.4. Accessibility

Another requirement of this project was to be able to access the database, and therefore the Raspberry Pi, locally and remotely. During this project, a user was created for remote access, which was granted all privileges. The user has to choose his own password, which is used to connect to the database remotely. By granting privileges to remote users, the privileges can also be granted to a specific IP address (device). This also increases the security of the server.

An effective firewall configuration on the Raspberry Pi is critical to securing the MariaDB database. By default, MariaDB uses port 3306 for communication, and restricting access to this port helps prevent unauthorized access. By implementing firewall rules, specific IP addresses can be allowed to connect to the MariaDB service, reducing the risk of unauthorized access and potential attacks. By default, port 3306 is disabled on the Raspberry Pi, making it impossible to connect to the database remotely. The first step in making the database remotely accessible was to enable port 3306. The bind address is a configuration parameter that determines the network interface to which the server is bound. The server will listen for incoming database connections on the specified network interface.

By default, the bind address is set to 127.0.0.1, which is the address of localhost, in this case the Raspberry Pi. This bind address has been changed to 0.0.0.0 because 0.0.0.0 means that the Raspberry Pi will listen on all available network interfaces. The bind address parameter is critical for controlling network accessibility and security. For example, you can increase security by setting the bind address to the local host. During the design phase, the server was created in a secure environment within the Tu Delft network. This allowed the bind address to be set to 0.0.0.0. For future use cases, the bind address needs to be changed to the specific IP address of the device or network interface that wants to connect to the server to increase security. After correcting the bind address and firewall configuration, it was possible to remotely connect to the server on the same network. The connection to the server was made in Python using the 'PyMySQL' library.

In Python, basic connection parameters such as 'host', 'username', 'password' and 'database name' are built in. The 'host' is the host address of the server. The username and password are created specifically for remote access by granting the appropriate privileges. The database name depends on the use of the code. If the user wants to access the server locally (via the Raspberry Pi), the hostname has to be changed to "localhost". Also, the username and password must be changed to the username and password of the local user account or the root account.

5.2.5. Remote access using a VPN server

By setting up a VPN connection, it would be possible to connect to the server from anywhere. After installing any VPN server software available for use on Raspberry Pi, the VPN server needs to be configured with encryption settings, user accounts, and network parameters. Since the Raspberry Pi will be behind a router, port forwarding must be set up for VPN traffic. VPN enhances security with strong encryption that protects data exchanged between remote devices. To further strengthen the overall security posture of remote access, authentication measures such as usernames, passwords, or advanced methods such as digital certificates ensure that only authorized individuals can establish secure connections [30].

An additional layer of security provided by VPN is the concept of tunneling, which creates a secure channel for data transmission. An example study where remote access was established by building a tunnel from the Raspberry Pi server to the client [31]. The VPN connection could not be established at first because the research group did not have permission to port forward on the university network. However, one member of the group was able to set up a VPN server at home using a Raspberry Pi. Port forwarding was set up on the home WiFi router to create a receiving point for the web connection. Figures E.2 E.3 confirm that a connection was successfully established on both Windows and Android operating systems.

OpenVPN was chosen over Wireguard because of its compatibility with a wider range of operating systems, including Windows, MacOS, Android, and iOS. While Wireguard is faster, it is currently only supported on Linux. OpenVPN was chosen for its availability on multiple platforms. To save time installing software, generating encryption keys, adding firewall ports, and setting up a static IP address for the Pi, a script called PiVPN was installed from GitHub. This tool simplifies the process of setting up a VPN and minimizes the risk of error. This was particularly beneficial for the group as they had no previous experience.

Running a script directly from a URL is generally not recommended as it can provide an easy entry point for unauthorized access to your Raspberry Pi, potentially causing significant damage. However, this script was used and verified by several websites that provide step-by-step instructions on how to create a VPN server on the Raspberry Pi [32][33]. Since the group was able to create a VPN connection to the Raspberry Pi at home and connect to it E.2, this feature can be added to the Raspberry Pi used during the project in future development stages.

5.3. Data Management

The requirement to store data in a structured and organized manner is met by storing the data in appropriate tables in MariaDB. The data in the tables is purged after a specified amount of time by a procedure function. The procedure function removes all the data stored in the table. By implementing this function, the data storage is not overloaded with data for a long time. It is possible to set the interval for clearing the table. The choice of the interval depends on the modules themselves. The data stored is also determined by the modules themselves.

5.3.1. phpMyAdmin

PhpMyAdmin, a freely available software tool coded in PHP, is designed to facilitate MySQL administration through web interfaces. It includes a wide range of features for managing MySQL and MariaDB databases. From basic tasks such as managing databases, tables, columns, relations and indexes to more advanced operations such as managing users and privileges, phpMyAdmin provides an intuitive user interface. At the same time, it retains the flexibility to execute SQL statements directly, providing

a comprehensive solution for database administration [34]. The availability of administrative functions in phpMyAdmin depends on the access privileges granted to it on the MySQL server. When you set up phpMyAdmin on your local test system, it will be granted root privileges, allowing access to all databases and enabling all administrative functions. However, your Internet Service Provider (ISP) will only grant you phpMyAdmin access to your own databases [35].

Since the group lacked prior experience and knowledge in creating and managing databases, phpMyAdmin was a useful tool to simplify the process of creating and managing databases. PhpMyAdmin made it easier to analyze and edit the data sent to and stored in the database. The figure E.1 shows the user interface of the software tool used. As you can see, the tool really provides a clear and simple overview of the databases.

6

Interface

The interface includes logging in viewing the raw data and calculating the power quality parameters of the incoming signal. The power quality can only be viewed when a known user logs in. This chapter aims to provide a comprehensive overview of the considerations, implementations, final product features, and usage aspects of the interface design process.

6.1. Requirements Considerations

The choice of programming language for the interface development was a crucial decision. In line with the groups expertise, Python was chosen for its versatility and extensive library support.

Ensuring cross-platform compatibility was a key requirement. PyQts cross-platform support allowed the interface to be created on a Windows laptop and then run on a Raspberry Pi running a Linux system.

Creating an interface that is not only easy to use but also has a professional aesthetic has been a key focus. The aim is to instil confidence in our users, ensuring that they not only find value in our product, but also perceive it as a reliable and trustworthy solution. According to [36], a professional-looking interface is integral to achieving this goal.

Emphasising the importance of security software is crucial, especially given our responsibility to protect user information and ensure the protection of our system infrastructure.

Our project aims to monitor power quality with separate pages, updating power quality parameter values every second. This frequent updating ensures per second monitoring, a fundamental aspect for power quality analyzers. The interface displays power factor, power magnitude, voltage sags and swells, VRMS, total harmonic distortion, and frequency deviation for our class S power quality analyzer. Additionally, incoming data is visually presented through plots to give users insight into the supplied signal.

The interface is clear, with separate pages for updating different power quality parameters. A welcoming home page presents our identity with a logo. An overview menu, together with clear buttons, simplifies navigation and makes our application easy to use. An information page is available, detailing our identity and providing insights into the purpose of our product.

6.2. Implementation Considerations

Choice of programming language and GUI library

A user-friendly interface with a professional appearance was a key consideration. For each GUI library in Python, an evaluation was conducted to assess its support for real-time plotting, cross-platform compatibility, learning curve, pros, cons, and suitable use cases, see table G.1. The analysis led to the conclusion that PyQt5 stands out as the optimal choice. PyQt is recommended due to its comprehensive features, real-time capabilities, professional appearance, and cross-platform support. It proves to be

particularly well-suited for the development of complex desktop applications involving real-time data visualization [37].

Documentation and development tool and Platform compatibility

Based on the table G.1, we selected only the GUI libraries that have cross-platform support. The other requirement, which led to the selection of PyQt, is that it is well documented. In addition, the Qt Designer tool was used for the interface layout, allowing a 'what-you-see-is-what-you-get' design approach and saving valuable development time [38].

Two-Step Verification and Secure Verification Code Transmission

To ensure security, 2-step verification is implemented, which adds an extra layer of security during the login process and requires users to provide additional authentication. Another requirement is to securely send verification codes to known users. This is done by using an external email service (Brevo) to securely send verification codes, which increases the overall security of the authentication process. Most external email programs are paid services. Brevo has a free version. It has a multi-layered data security framework in place to ensure that its products, applications and user data are secure. The company is also ISO 27001:2013 certified, which ensures that it meets security standards [39].

Login Process Architecture

The goal is to establish an access control mechanism to enhance security. The architecture of the login process is built upon the Model-View-Controller (MVC) design pattern. A design pattern is a reusable solution to a commonly occurring problem in software design. Design patterns are not code or algorithms but rather high-level descriptions that guide the overall architecture and interaction of software components.

MVC stands for Model-View-Controller, this is an architectural design pattern in which each of these components must be separated from each other. The 'Model' manages how data from the database is presented to the user, the 'View' is responsible for displaying that data on the screen without knowing what this data means, and the 'Controller' allows the 'View' and 'Model' to interact with each other. When users make requests via the 'View', the 'Controller' ensures that the appropriate action is taken. This architectural pattern is implemented in the following way. Model: The database-related operations and functions within the classes used, the database interactions such as query, insert and update user data are part of the model. The data handling operations are handled by the functions that check login, check if the email is in the database and insert and update in the database. View: The interface layouts are created using the Qt designer, which generates the GUI files. The visual elements and user interaction elements such as buttons, input fields and labels are part of the 'View'. Controller: The controller manages user input and responds by coordinating with the 'Model' and updating the 'View' accordingly. Event handlers are responsible for processing user input, making decisions based on user actions, and updating the 'Model' or triggering changes to the 'View'. The MVC design pattern ensures a clean separation of these parts. The 'Model' handles data, the 'View' manages the user interface, and the controller facilitates communication between the 'Model' and the 'View' based on user interaction [40].

Database Interaction

To ensure the secure handling of user data, database operations such as query, insert, and update were encapsulated within the 'Model' component of the MVC pattern. This architecture maintains a clear distinction between the database operations and other aspects of the application.

Email Service Selection

For sending verification code to the emails of the user, Brevo is used as an external email service, because Google Gmail API discontinued non-visible email support. The chosen service allows sending emails without requiring explicit authentication, aligning with the user experience requirements. However, a challenge arises as emails sent through Brevo end up in the spam box due to potential trust issues with the SMTP server. Despite successful email delivery, there's a need to tackle spam box placement, which may involve adjusting email content, enhancing email authentication, and seeking specific recommendations from Brevo. For now, a text has been added to the interface informing the user that the email

may end up in the spam box and that it needs to be checked. The implemented 'smtplib' code handles the connection to Brevo's server, initiates TLS for security, logs in with email credentials, and sends the email.

6.3. Usage

The login process is restricted to registered users. To create an account the user must provide their email address and set a password, with a confirmation step to ensure that the user enters the intended password. This email-password pair is stored in a database with columns for email and password, with matching email and password in the same row. Account creation is only allowed for unknown email addresses. During login, the email and password are checked against the database and two-step verification is implemented, requiring the user to enter the correct code sent to their email. The application will then open. Only known users can change their password through a Forgot Password screen, where their email is checked to see if it exists in the database, and after entering the correct verification code, the user can change to a new password. Once the password is changed, the database is updated with the new password corresponding to the user's email. The user is given the option to go back to the login screen during the login process, see figures D.1 and D.2 for the visualisation of the login process. The interface also interacts with the user. When the user enters an invalid password, email, or verification code, an error message appears to notify the user that an issue has occurred.

6.4. GUI

Finally, after logging in, the application of our power quality analyzer is opened. Our application is called Power Quality Quantizer, which is shortened to PQQ, and with a bit of fun it becomes PQ^2 , which is used to make the logo. The home page opens when the application is launched. In figure D.3 the interface is shown. The layout is divided into the left menu container and the main body container. The left menu container contains navigation buttons for Home, Current and voltage signals, Harmonics present in signals, Results and Information. The right part of the menu is the main body container, which is a stacked widget that displays different pages based on user selection, including real-time plots of current and voltage signals, harmonics analysis and results of power quality parameters. The interface uses standard colours. Real-time plots are implemented using the pyqtgraph library. The application allows the user to seamlessly navigate between sections providing detailed information on power quality analysis and results.

6.5. Final Product

In Figure D.1 and D.2, the layouts of the login process are depicted. In Figure D.1, the login screen facilitates user login. After pressing login, the verification page opens, prompting the user to click on the 'Send Code' button for the verification code to be sent to their email. Upon entering the correct verification code, as shown in D.3, the user is directed to the homepage. Account creation is possible by selecting 'Don't have an account? Create Account' button on the login page, leading to the sign up page on the right layout in Figure D.1. For password recovery, clicking on the 'Forget Password' button on the login page initiates a page where the user enters their email, illustrated in Figure D.2 on the left layout. If the email is recognized, the verification page opens, and upon entering the correct verification code, the user can set a new password, as shown on the right page in Figure D.2. Clicking 'Save' updates the database, allowing the user to return to the login screen. Error handling is implemented to notify the user in case of incorrect or unintended inputs.

In Figure C.3, a graph depicts the supplied voltage, updating every second. Figure C.4 illustrates the Current signal page, displaying the updated current information every second. Figure C.5 shows the Harmonics page, presenting harmonics in a bar graph that is updated every second. Lastly, Figure C.6 exhibits the Result page, where power quality parameters and their values are shown and updated every second.

7

Modules

When considering the specified requirements for the power quality analyzer, a critical aspect is the accurate operation of the parameter algorithms. These algorithms must not only calculate the selected Class S parameters, but also meet Class S accuracy standards. In addition, the algorithms are expected to perform these calculations within specified time frames to ensure timely and seamless presentation on the interface. This underscores the importance of accuracy, reliability and efficiency in the analysis of power quality parameters.

7.1. Frequency calculation

The second module discussed in this chapter is the frequency calculation. An accurate calculation of the instantaneous frequency must be made to determine if there is a deviation in the 50 Hz grid.

7.1.1. Importance of frequency deviation

The first question that can be answered is why one should even bother calculating the frequency of the grid. The grid frequency is crucial as it measures the balance between the supply and demand of power. If the demand is lower than the supply the power generators will have less resistance resulting in a higher frequency. If the demand is higher than the supply the generators will have to exert more torque resulting in a lower frequency. This frequency was set to 50 Hz in the late 1800's [41] as it fit the specifications at that time.

Now that the grid frequency was established all devices were made to work in Europe at 50Hz. But what happens if this frequency deviates from this mean? A deviation in frequency can affect the efficiency of devices, damage them, and even trigger protection within devices.

7.1.2. Calculations methods

Now that the importance of frequency calculation is established an accurate and reliable calculation method has to be determined. For this project, two calculation methods have been implemented:

- Fast Fourier Transform Method
- Zero-crossing Method

Both these methods have been developed and have been tested with simulated data.

Fast Fourier Transform

The Fast Fourier Transform (FFT) converts the time domain into the frequency domain. Thus this seems like a great way of determining the frequency of the incoming signal. It is suitable for this use case as its calculation is done efficiently in Python. The equation used in the Python module is shown in equations 7.1 and 7.2.

$$A_k = \sum_{m=0}^{n-1} a_m e^{\{-2\pi i \frac{mk}{n}\}} \quad k = 0, \dots, n-1 \quad (7.1)$$

$$a_m = e^{\{2\pi i f m \Delta t\}} \quad (7.2)$$

These formulas calculate the complex exponential at a certain sampling interval and frequency components. This approach is great as it can differentiate between the signal and the noise frequencies. However, this approach only returns discrete frequency values. This is a problem as we already know the frequency will be around 50Hz but we want to know a more specific frequency. To improve this estimation the peak frequencies and amplitudes are interpolated to end up with a more accurate frequency estimation.

Zero-crossing method

The zero-crossing method is a method where every time instance at a zero-crossing is determined. In the ideal case, the time between zero-crossings will be half the period. Using this information the frequency of the signal can be determined. A schematic representation of this method is shown in figure 7.1

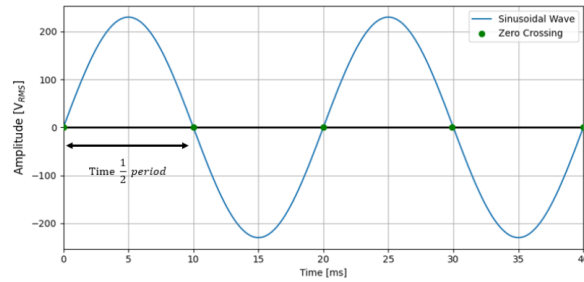


Figure 7.1: Using the zero crossing method to determine the frequency based on half a period

This is a straightforward concept but not that robust because if the calculation is based on a signal with a phase shift or a different frequency this error will be multiplied by two and assumed for the whole signal. The first implementation to make the algorithm more robust is to calculate the frequency based on all half-periods and averaging them per second. This will give a more accurate output of the frequency per second. One of the main disadvantages of the zero-crossing method is its behaviour with noise. In this instance, there could be multiple zero crossings detected in half a period. To make the algorithm robust to this behaviour after all zero crossings have been detected a look at the samples can be done and samples that are too close to each other should be discarded. But how does one know when to start looking for new zero crossings again? To answer this an assumption has to be made. This assumption is that it is expected to find a frequency at around 50 hertz. For the Dutch grid, this assumption is in place as there are many regulations stating that the grid operators maintain the stability of the grid around 50 Hz [42]. Next to this assumption, it can be argued that one-fourth of the period would be the ideal time to start looking for a new zero crossing as this is the furthest away from two zero crossings. Now these two factors can be used to calculate the best amount of samples to wait for a new zero crossing. This concept is shown in figure 7.2.

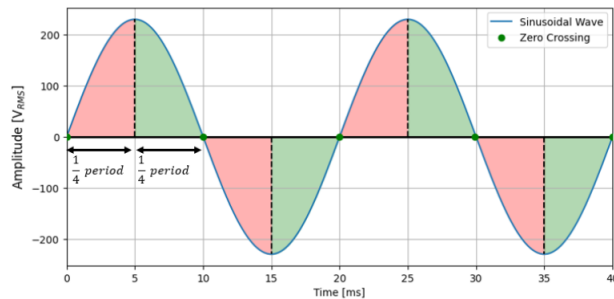


Figure 7.2: Representation of the window for zero crossing

Performance

Both methods work in theory, but which one gives the most accurate solutions and is the most robust for this use case? To answer this question both methods are tested with simulated data. A sinusoid with a known frequency is created and white noise with different standard deviations is added. The calculated frequency is compared to the actual frequency and displayed in the correlation plot in figure 7.3.

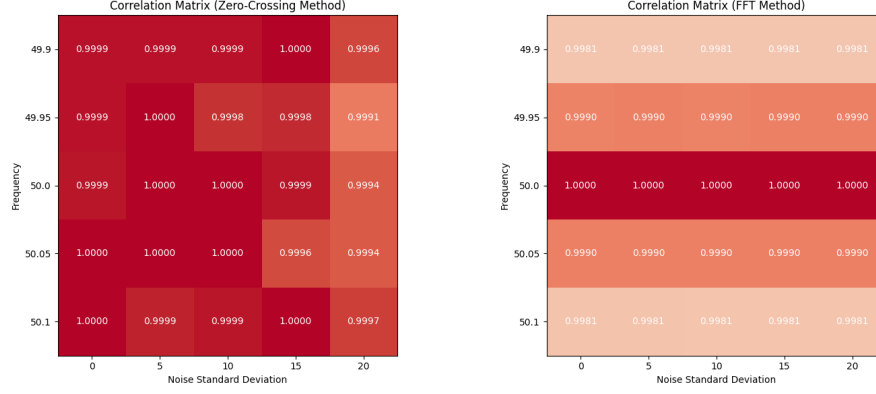


Figure 7.3: The performance of the FFT and zero-crossing method

As can be seen in 7.3 the zero crossing method performs well but degrades when the standard deviation of the noise increases. This is as expected due to the 'random' factor of the white noise. As can be seen, by the performance of the FFT, the performance is overall slightly worse but is not affected by noise. The choice has been made to use the zero crossing method as this system ought to be robust enough for the use case and has a higher accuracy.

7.2. Power Factor

The power factor of an AC power system is the ratio of the real power consumed by the load to the apparent power transmitted in the circuit. Or as described by many [43] [44] [45], "The power factor is the cosine of the phase difference between the voltage and the current". The power factor is a unitless quantity and defines how efficiently electrical power is being utilized by the connected load. It also indicates whether the current is lagging, leading or in phase with the voltage by computing equation 7.3. The power factor is a range between -1 and 1. The term 'leading power factor' refers to a situation where the current leads the voltage resulting in a negative power factor, indicating a capacitive load. Conversely, 'lagging power factor' refers to a situation where the current lags behind the voltage resulting in a positive power factor, indicating an inductive load. In a purely resistive load, there is no phase difference between the voltage and current signal which leads to a power factor of unity [44]. A load with a power factor of 1 (maximum) results in the most efficient system loading. Where a power factor lower than 1, meaning that active power is being consumed at a fixed voltage, leads to increased current and higher ohmic losses. Households and small commercial users usually have a power factor close to 1, whereas heavy-industrial loads rather have a low power factor [45]. The power factor can have a negative value. This occurs when the device, typically the load, generates real power that then flows back to the source [43].

$$pf = \frac{P}{S} = \cos(\theta_v - \theta_i) \quad (7.3)$$

A low power factor leads to increased current flow through the system, which in turn increases power loss as heat is dissipated in the conductors. This unnecessary energy loss not only wastes energy but can also increase utility bills and overall energy consumption. Another consequence of having a low power factor is that the electricity efficiency is being compromised. A lower power factor means that the reactive power is using more of the electrical capacity of the system, which leads to less capacity being available for active/real power that is responsible for doing the useful work. This condition may reduce

the lifespan and performance of equipment like transformers and motors. Low power factor also could be the cause of voltage drops which in turn affects the operation of equipment and voltage regulation. Knowledge about the power factor makes it possible to consider taking measures that can increase the power factor. Improvement of the power factor can enhance system efficiency, reduce energy costs and extend equipment operating life.

7.2.1. Power factor determination

The algorithm that has been designed sets up a system to keep track of power factors in real-time but is also compatible with data in array format. To start with, it obtains the data from the MariaDB database that processes the live voltage and current data.

After successfully retrieving the data, the next step is to pass the data to the function that is responsible for tracking and identifying the zero crossings in the voltage and current data. These functions almost operate in the same way, having a small difference. The zero crossings are identified by the functions that are searching for a sign change in the data. The function starts looking for a sign change after waiting for a short time period, which is defined as starting time. The starting time was implemented during the testing stage of the algorithm to prevent the function from falsely detecting the starting point as a zero crossing. The starting time of the current zero crossing tracker is different to the voltage zero crossing tracker, as the current signal is expected to have a phase shift. To improve the accuracy of the algorithm and reduce the effect of noise, a sliding window was designed. The first zero crossing is being saved and updated after every period to estimate the location of the next zero crossing in the signals. The sliding window is being shifted with a frequency of 49 and 51 Hz from the last zero crossing to create boundaries for the region where the next zero crossing is estimated to be. A buffer of 1 Hz was chosen to limit the detection of false zero crossings as a result of noise because a difference in the range of 1 Hz does not affect the accuracy of the power factor value that much. The time difference between the first and last zero crossings are being computed and the result is being checked if it indeed has a difference between 49 and 51 Hz since the last zero crossing. A demonstration of the sliding window is being shown in figure 7.4. Valid zero crossings are stored in an array, and the function ensures that the last valid crossing is returned, along with the time of the previous zero crossing.

$$\cos(\theta) = 2\pi * \frac{\Delta t}{T} \quad (7.4)$$

The zero crossings are then being used to calculate the time difference between them. The time difference that is being determined in the previous step will be used in the formula as described in 7.4. The Δt in the formula is the time difference between the zero crossings, and T is the time of one period, which corresponds to 20ms. As can be seen the phase difference in time is first being converted to radians. With the phase difference obtained in radians, the power factor is calculated by taking the cosine of the phase difference. This process is repeated for every period. The power factor values are then saved in an array. The array will save up to 100 power factor values, from which it then will compute the mean value. The mean power factor values will be updated on the interface continuously after the recording of 50 power factors. Taking the mean of 50 power factors is done to minimize the effect of noise and maximize the accuracy of the algorithm. It is limited to take the mean of 50 power factors to ensure that the power factor is updated every second.

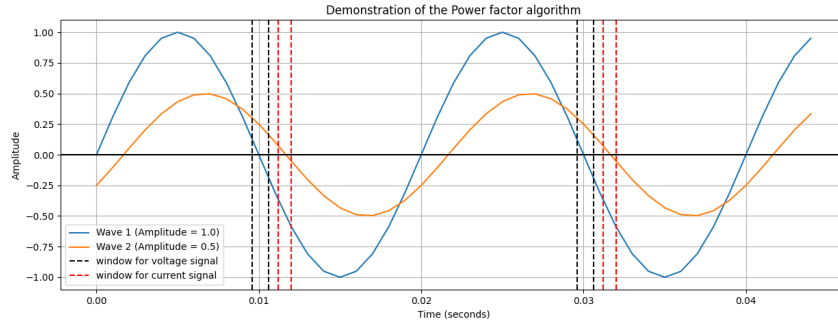


Figure 7.4: Demonstration of the power factor algorithm

Simulation testing

Figure 7.4 displays the operation of a sliding window used to detect zero crossings after each period. The voltage wave is simulated without a phase delay, while the current wave is simulated with a delay of $\pi/4$. In ideal circumstances, the phase difference should be approximately $\cos(\pi/4) = 0.7071$. After running the code and calculating the mean of 100 power factors, a power factor of exactly 0.7071 was observed. The algorithm is also tested using a noisy signal as shown in figures F.2 and F.3. This was performed for 1000 power factor values, resulting in The power factor ranging from 0.7134 to 0.7280. The accuracy of the power factor estimation was increased by taking the mean of 100 values. By using a larger sample size to calculate the mean, accuracy can be further improved. During testing with noise, the sliding window boundaries were reduced to 0.5 Hz. It was observed that this also led to a higher level of accuracy. The table below displays the results of tests conducted for various phase shift values. From the table can be observed that the designed power factor calculation algorithm behaves as expected, with the maximum difference from the expected power factor being 0.015. This difference can be explained by the fact that noise could have occurred in the window itself since the window has a buffer of 2 Hz (1 Hz on each side). During the testing stage, one issue encountered is the algorithm's inability to detect a 180-degree phase shift. The algorithm detects a value of 1 instead of -1. It is worth noting that a phase shift of 180 is a rare occurrence and was therefore not taken into consideration.

Table 7.1: The testing results of the power factor algorithm

Lagging Phase shift (in degrees)	Expected power factor	Obtained power factor
90	0	0.0037 - 0.0150
60	0.5	0.4887 - 0.4937
45	0.7071	0.7107 - 0.7205
30	0.866	0.867 - 0.872
11.25	0.9807	0.9821 -0.9833
Leading Phase shift (in degrees)	Expected power factor	Obtained Power Factor
90	0	0.0037 - 0.0150
60	-0.5	-0.4887 - -0.4937
45	-0.7071	-0.7051 - -0.6961
30	-0.866	-0.861 - -0.8567
11.25	-0.9807	-0.9801 - -0.976

7.3. Voltage Sags and Swells

Voltage sags and swells are widely acknowledged as the most problematic disturbances that are capable of affecting commercial, industrial, and household electrical loads. Even brief periods of voltage sags can cause irreversible damage to sensitive equipment and significant economic losses due to unexpected interruptions in industrial production processes [46]. A voltage sag is characterized as a short-term RMS fluctuation of 0.1-0.9 p.u. Whereas, swells are fluctuations of 1.1-1.8 p.u. With both having a duration between 0.5 cycle and 1 min [47].

Voltage sag can cause serious problems to sensitive loads because these loads often drop off-line due to

voltage sag. As a result, some industrial facilities experience production outage that results in economic losses [48]. The duration of voltage sags is often reliant on the speed of fault clearing. The sag duration refers to the period in which the voltage is below the specified threshold, typically lasting less than one second. As per the guidelines set by IEEE Std. [2], voltage sag can be categorised into three types based on their duration: i) Instantaneous (0.5-30 cycles), ii) Momentary (30 cycles-3 seconds), and iii) Temporary (3 seconds-1 minute). Voltage sag occurs at nearly all locations in the power system, and avoiding them is only possible to a certain extent. Voltage sag is caused by system faults, transformer energisation, or heavy load switching. Mitigating the frequency and severity of voltage sag beyond good engineering practice can incur significant costs to the customer.

Several methods have been developed to detect and characterize voltage sags and swells. These methods include various approaches such as waveform envelope, discrete wavelet transform, missing voltage, threshold RMS voltage, Direct-quadrature-zero transformation, numerical matrix and peak detector[49]. Existing methodologies leverage either instantaneous or RMS measurements of single or three-phase voltage waveforms. During this project, a peak detection and an RMS algorithm were developed.

The peak detection algorithm is essential for power quality analysis due to its ability to identify events such as sudden spikes or dips in voltage that signal voltage sags and swells. This capability enables the timely direction of abrupt changes in the voltage signal, allowing to address potential issues, as described above, associated with these disturbances. Peak detection algorithms are also well known for their advantage in their fast response time [50]. They can quickly identify sudden changes in voltage levels. Peak detection algorithms also bring in the benefit of their simplicity of implementation. Their uncomplicated design contributes to computational efficiency, enabling real-time monitoring and rapid analysis of power quality events. The advantages of the simplicity are easier integration into existing systems and enhancement of the algorithm's practical applicability in diverse settings, ranging from households, and smart grids to industrial environments. Several studies have implemented the peak detection algorithm in different manners[47] [51] [52]. The peak detection algorithm detected during this project is explained in subsection 7.3.3.

The Root Mean Square (RMS) algorithm plays a crucial role in power quality analysis for several reasons. Firstly, RMS algorithms provide a comprehensive measure of the overall magnitude of the voltage waveform. Unlike scenarios where voltage sags and swells might not be characterized by a single peak or trough, RMS values provide a representation of the waveform's magnitude over a specified period. The RMS algorithm is inclusive, capturing nuanced variations that may occur during disturbances by considering the entirety of the waveform. RMS algorithms take both positive and negative variations into account, providing a more balanced and complete assessment. RMS values are less sensitive to fluctuations caused by specific transient events, providing a more reliable indication of the ongoing impact on the system. Additionally, aligning RMS algorithms with established power quality standards is advantageous. Many standards define limits based on RMS values, making the use of RMS algorithms not only reliable but also compliant with power quality assessment standards. The Root Mean Square method to detect voltage sags and swell have been used in several studies, and also have been mentioned as a method to detect these distortions in these and several other studies [53] [47] [49] [54] [51]. The upcoming subsections will explain the RMS algorithm that has been implied during this project.

7.3.1. V_{RMS} algorithm using live data

The designed RMS voltage algorithm creates a real-time monitoring system for AC voltage signals, primarily focused on detecting voltage swell and sag events. The algorithm uses simulated live voltage data employing a sliding window approach to calculate the Root Mean Square (RMS) value of the voltage signal. The RMS of every window is being computed. This calculation is a crucial metric for quantifying the amplitude of the varying AC voltage signal, providing insights into its quality. The RMS value is obtained by using the formula 7.5.

$$V_{RMS} = \sqrt{\frac{1}{N} * \sum_{i=1}^N x_i^2} \quad (7.5)$$

In this formula, N is the number of samples within the sliding window, where the window size is determined by the samples per period variable. The sample per period variable is calculated by dividing the sample rate by the signal frequency. Each voltage sample within the period is represented by x_i . The formula involves squaring each voltage sample, summing up these squared values, dividing the sum by the number of samples, and finally, taking the square root of the result. The RMS value provides a measure of the effective or "root" power in the AC signal, accounting for both positive and negative excursions [47].

The algorithm employs a dynamic approach using a sliding window to calculate the Root Mean Square (RMS) value of live AC voltage data. The script maintains a window of samples, which is updated in real-time as new voltage data is received. The RMS value is calculated when the number of samples in the window completes a full period of the AC signal. The sliding window allows for continuous monitoring, making it possible to detect voltage swell and sag events based on the specified thresholds (1.1 - as described in 7.3. To be more precise, the swell threshold that has been used equals $1.1 * 230$, whereas the sag threshold used equals $0.9 * 230$). The sliding window ensures adaptability to evolving signal characteristics, providing a real-time assessment of the voltage data.

Furthermore, the algorithm logs the data associated with the voltage events into the dynamically created tables within the database. Each table represents one voltage event that has been detected. The system uses SQL queries to insert voltage data with timestamps, ensuring a complete record of each event's duration and characteristics. It also includes logic to handle the end of voltage swell and sag events, saving the related data to the database to capture the entirety of each event.

7.3.2. V_{RMS} algorithm using sampled data

The RMS voltage algorithm had to be adjusted, due to the changes made at the integration 3.5 part of all the modules. The sampled values from the live data are not instantly fed to the algorithms. To be able to take exactly one period, the algorithm starts by looking for a zero crossing or the value closest to zero. That point is being used as the starting boundary for the window. The length of the array input including all the data and the frequency estimation 7.1 algorithm are used to calculate the number of samples in one period. Which is obtained by dividing the length of the array by the estimated frequency. This value is then being used to determine the end boundary of the window, by adding the samples per period by the starting boundary. All the samples within this window are passed to the equation 7.5 that computes the RMS value for that period. With the RMS value obtained, sags and swells can be detected for every period. After every period the RMS value is being compared to the threshold values defined for sags and swells. The threshold values used in 7.3.1, are used again. If a distortion is detected, it will be shown to the consumer on the interface by plotting the magnitude of the event. The magnitude and the moment of the event will be saved to the corresponding table in the database. The power magnitude of the voltage signal is obtained by calculating the mean of V_{RMS} values.

Simulation result

Figure 7.5 displays the outcomes obtained from the virtual testing of the V_{RMS} method on a simulated live AC signal which corresponds to the input of a Dutch household. The algorithm behaves predictably in response to voltage fluctuations. The V_{RMS} decreases during a sag and increases during a swell. All data about these events are recorded in separate tables in the database, complete with a column that includes the timestamp denoting the moment of occurrence.

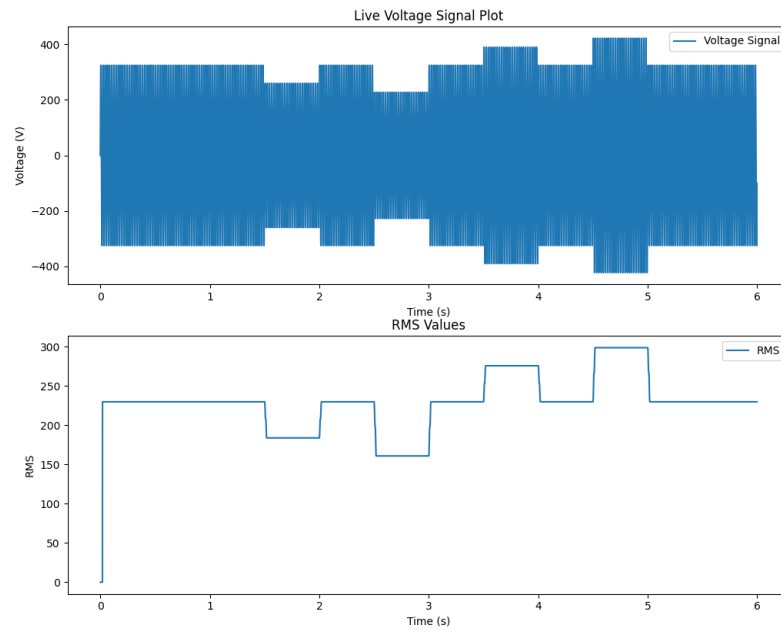


Figure 7.5: A 6 seconds simulation of an AC signal using the V_{RMS} algorithm

Figure F.1 presents the algorithm's initiation, where V_{RMS} values are assigned a period after one cycle. This observation aligns with the expectation since the sliding window operates on one cycle at a time; accordingly, the V_{RMS} is calculated from that cycle as per equation 7.5. During the initial period, the system reports a 'Voltage sag detected' message if the V_{RMS} value falls below the sag threshold. Since this occurs in the first period due to the sliding window, the message is falsely reported. However, this behaviour is expected, as the V_{RMS} value is obtained after the period. To prevent this error, the V_{RMS} values will only be recorded after the first period.

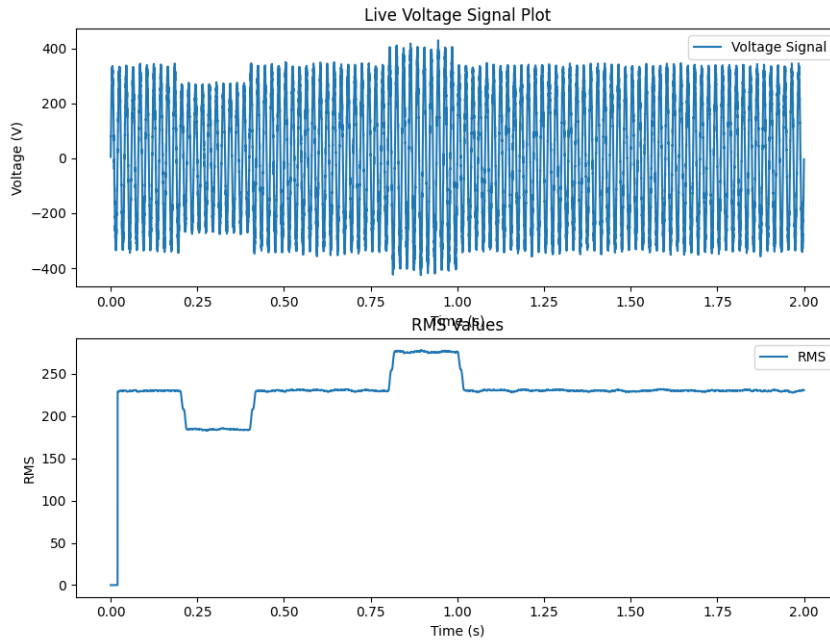


Figure 7.6: Simulation of the RMS algorithm using a noisy input signal

From the graph 7.6, it is evident that noise has a negligible impact on the RMS calculation. The noisy signal that has been used during testing is shown in figures F.13 F.14. The data depicted in the graph illustrates that the RMS value remains relatively stable despite the presence of noise. This indicates that the RMS calculation is robust enough to filter out the noise, ensuring the accuracy of the resulting RMS value. The consistent trend observed in the graph reinforces the conclusion that the noise does not exert a significant influence on the RMS calculation, affirming the reliability of the calculated values.

7.3.3. Peak detection algorithm

The designed peak detection algorithm uses a real-time monitoring system for detecting voltage swell and sag events in AC electrical systems. The algorithm first starts by establishing a connection to the MySQL database, configuring essential parameters 5.2. Voltage thresholds, compliant with IEEE standards (as described in section 7.3) and expressed as percentages relative to a nominal AC voltage of 230V, are defined to serve as criteria for identifying swell and sag events.

The core of the methodology involves continuous monitoring of the AC voltage signal in real-time. Voltage data is processed periodically, and voltage peaks are detected using the *find_peaks* function in Python. The algorithm calculates the percentage deviation from the nominal voltage based on identified peaks, facilitating the immediate detection of voltage swell and sag events. The maximum and minimum peaks are taken from each period. In reality, there is no perfect sinusoidal signal input. The signal may be noisy, which must be considered when analyzing it. By specifying the maximum and minimum peaks, the effect of noise on the calculation of the deviation from the nominal voltage is minimized. This is because only two values are considered, resulting in a more accurate measurement. A sliding window is also used for this algorithm but in a different form. The sliding window approach in this algorithm resets the peak values after each period. This has been done to prevent using the bigger maximum or smaller minimum peak from the periods before. The formula used to calculate the deviation is 7.6. The calculated percentage deviation provides a measure of how much the measured voltage deviates from the expected nominal voltage, expressed as a percentage of the nominal voltage. This metric is then used in the module to determine whether a voltage swell or sag event has occurred based on predefined threshold values.

$$\text{Percentage Deviation} = \frac{(|\text{Max Peak} - \text{Nominal Voltage}|, |\text{Min Peak} - \text{Nominal Voltage}|)}{\text{Nominal Voltage}} \times 100 \quad (7.6)$$

The module uses a database-driven approach to efficiently track events. It dynamically creates tables in MySQL to store detailed information, such as timestamps and voltage values, upon event detection. This methodology ensures a well-organized record of voltage events, facilitating subsequent analysis. The system provides real-time feedback by printing deviation values. Details of voltage events that exceed predefined thresholds are recorded in the MySQL database.

Simulation result

The algorithm is tested with a 230V, 50 Hz virtual AC signal that has 2 sags and 2 swells in it that have been simulated for 6 seconds. This is a simulation under ideal circumstances. Figure 7.7 shows the simulation result. The Red dots are the peaks that are being taken. Both positive and negative peaks are taken into account because a sag/swell can have a duration of 0.5 cycles. This means that such a distortion can start at both sides, so both sides are considered in the algorithm.

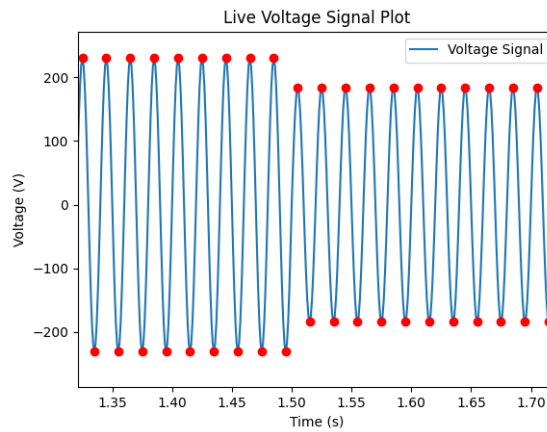


Figure 7.7: Peak detection algorithm on a clean voltage signal

The algorithm is also tested by using a noisy AC input signal F.13. It was observed that the maximum and minimum peaks could be influenced by noise. High or low peaks caused by noise that exceed the threshold value, would falsely detect a distortion. That was the main reason that the V_{RMS} algorithm is chosen to be used over the peak detection algorithm.

7.4. Harmonics

Another important set of requirements concerns the system's harmonic analysis capabilities. The analyzer must be able to detect and analyze harmonics in both current and voltage signals.

Harmonics have been identified as a concern for EU networks, as they can cause significant damage. It leads to excessive heating, premature ageing, failure of customer equipment and resonance. Harmonics cause additional losses in loads, causing loads to operate at a lower rate than their actual ratings, reducing energy efficiency [55]. Harmonics are described as follows (IEEE 1159-1995): "Harmonics are sinusoidal voltages or currents having frequencies that are integer multiples of the frequency at which the supply system is designed to operate" [56].

Normally, when a linear, time-invariant electrical load is connected, it draws a sinusoidal current that matches the frequency of the supplied voltage signal. Harmonics are caused by non-linear loads, when such a load is connected it draws a current signal that may not be strictly sinusoidal. This drawn

current signal then distorts the supplied voltage signal [57].

Harmonics have several negative effects on the grid, significantly impacting power quality. It is very important to know what kind of harmonics are present and how much they affect the current and voltage signals. So, the measurement of harmonics is essential in the design of a power quality analyzer [58].

The Fourier Transform is the most popular method of analysing the harmonics in a signal [59]. To detect the current and voltage harmonics and their severity, a harmonic detection algorithm is written in Python. For harmonic analysis, Numpy or the Scipy library in Python could be used, Scipy is used library. The SciPy module for FFT is a more comprehensive superset of the Numpy [60].

The algorithm retrieves the incoming current and voltage data from the MySQL database. The FFT transforms the electrical signal from the time domain to the frequency domain, which extracts all the frequencies present in the signal.

The Fourier Transform outputs a double-sided spectrum, spanning from negative infinity to positive infinity. This spectrum consists of both positive and negative frequencies, with amplitudes in the double-sided spectrum being half of the peak amplitude in the time domain. In digital data acquisition, particularly when utilizing a FFT, only the positive frequency range is typically displayed and processed. Real-valued signals do not include negative frequencies, so in a double-sided spectrum, the negative frequencies lack useful information and mirror the positive frequencies. So we move from a double-sided spectrum to a one-sided spectrum, which contains only the positive frequencies [61] [62]. After obtaining the frequency spectrum, small peaks appeared around zero. To extract only the relevant information, a threshold was applied to isolate and distinguish the relevant peaks from the less significant ones. This step is crucial for the extraction of frequencies and amplitudes using the peak function of the Scipy library, eliminating unnecessary peaks. The threshold was set by playing with the signal and observing the point at which unwanted small peaks were effectively filtered out.

The grid frequency fluctuates around 50 Hz rather than being a fixed integer value. Applying the FFT produced integer values for the frequency components. To extract these non-integer frequencies with greater precision, the resolution of the frequency axis in the frequency spectrum was improved by using the interpolation method. This mathematical technique estimates values between known data points by constructing a function that approximates the behaviour of the data set, we were able to achieve with this method frequencies specified to decimal places rather than integers. In signal processing, interpolation is useful for estimating values between discrete samples [63]. Given the sinusoidal input signal, quadratic interpolation is particularly well-suited for capturing the smooth and periodic nature inherent in such signals [64].

More specifically we used the quadratic interpolation of the SciPy interpolation function. Quadratic interpolation constructs an approximating quadratic function that fits the sample data. This mathematical technique estimates values by assuming the changing points between samples can be represented by a quadratic curve. In this case, the SciPy function for quadratic interpolation was used to calculate frequency values to decimal places rather than integers. Interpolation enhances the frequency resolution beyond what the FFT can provide alone. It allows a more accurate representation of the true fluctuating grid frequency, which occurs at non-integer values between FFT bins. The higher resolution from quadratic interpolation via SciPy captured the decimal-valued frequencies present in greater detail.

The amplitude was normalized by the signal length to obtain accurate amplitudes. The algorithm's main objective is to generate arrays of valid harmonic frequencies and their respective amplitudes. The described process in this subchapter accomplishes this goal. The resulting frequency and amplitude arrays are stored in the database. Total Harmonic Distortion (THD) analysis utilizes this information to quantify the signal distortion.

Simulation

A sinusoidal signal with an amplitude of 327.25 V and a frequency of 50.05 Hz was generated in order to test the harmonics algorithm. The signal was enriched by including harmonics up to the seventh harmonics and random Gaussian noise. By sampling the signal at a frequency of 10 000 Hz. All expected harmonics were successfully extracted, as shown in Figure 7.8. In this figure, the first plot shows the generated signal and the second plot shows the frequency spectrum of this signal.

Signals with varying frequencies, amplitudes, additional harmonics, and noise were tested, and the outcomes aligned with expectations.

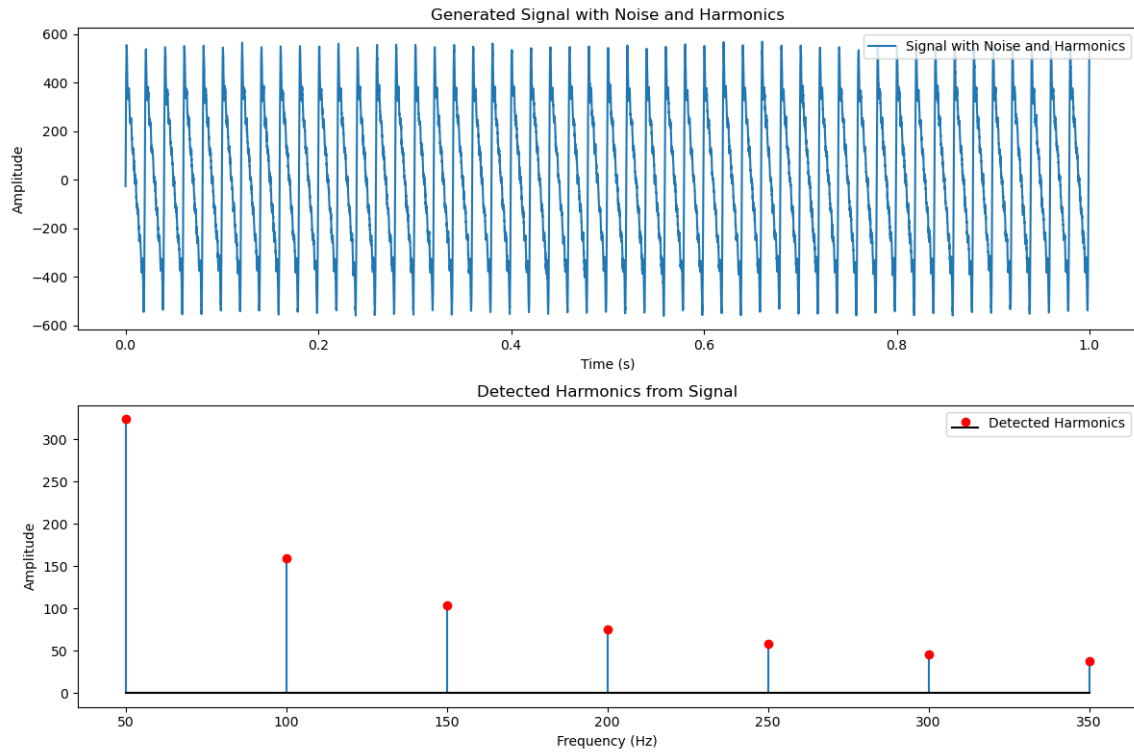


Figure 7.8: 50.05 Hz signal with an amplitude of 327.25[V], noise and second till the seventh harmonics is added the signal

```
[ 49.9904999  99.98099981 150.02149021 200.01199012 250.05248052
 300.04298043 350.13346133] [324.21072871 159.87146717 104.68240212  75.76518993  58.37226538
 46.43415388  37.68703699]

Process finished with exit code 0
```

Figure 7.9: In Python all existing harmonics are extracted from the signal that is generated in 7.8

7.5. Total Harmonic Distortion

Total Harmonic Distortion (THD) is used to measure the harmonic distortion introduced into an electrical signal or system by non-linear devices. It quantifies how much harmonics deviate from the ideal sinusoidal waveform. THD is expressed as a percentage and it is commonly used in power systems and signal processing to evaluate the quality of electrical signals, to evaluate how pure the signal is. It is calculated by comparing the root mean square (RMS) of the harmonic components to the RMS of the

fundamental frequency. The formula is often written like this [65]:

$$\text{THD (\%)} = \frac{\sqrt{H_2^2 + H_3^2 + \dots + H_n^2}}{H_1} \times 100 \quad (7.7)$$

Low THD means that the system output is similar to the system input, with minimal distortion. A high THD indicates that the output signal is highly distorted than the input signal [66]. The THD algorithm retrieves the array of amplitudes from the harmonics algorithm discussed in 7.4. The first element of this array is the amplitude corresponding to the fundamental frequency. The RMS values of the elements are first calculated with

$$RMS = \frac{\text{elements from amplitude array}}{\sqrt{2}} \quad (7.8)$$

Since the elements in the array from the database are not in RMS. After this equation, 7.7 is applied. As THD is expressed as a percentage, the value is multiplied by 100. This value is displayed on the interface of our power quality analyzer to show the user how distorted their signal is.

Simulation

The simulated signal in 7.4 is used to test the THD algorithm. In figure 7.10, the first array shows all the detected frequencies and the second array shows all the corresponding amplitudes of the detected frequencies using the harmonics algorithm described in section 7.4. The amplitude array is used to calculate the THD, for this particular signal the THD is 68.5%, which means that the signal is very distorted.

```
Array of detected frequencies: [ 49.9904999  99.98099981 150.02149021 200.01199012 250.05248052
300.04298043 350.08347083]
Array of the detected amplitudes corresponding to the frequencies:
[323.92069492 160.02931472 104.41751259 75.86179892 58.56086923
46.21281714 37.46411939]
THD: 68.50189075303177%
```

Figure 7.10: THD result with simulated signal

System Integration and Testing

When all components of the project were finished, they had to come together as was designed in figure 3.1. The I2C communication has to write the sampled data to the database, the remote interface has to fetch data from the database and implement the calculation modules, and the modules have to use the fetched data to run the calculations. To achieve this, implementations had to be done on the Raspberry Pi and on the remote interface.

8.1. Raspberry Pi and database Integration

8.1.1. Raspberry pi implementations

The Raspberry Pi implements the I2C communication and has to write the samples to the database. To do this continuously, the program must sample data while also writing it to the database simultaneously. This can only be achieved if the two processes work in parallel. To achieve this, multiprocessing was implemented in the C code. This way after the samples have been acquired and the samples are converted from binary values to voltage values these values get written to the database simultaneously. The samples get a timestamp when they get written to the database, using the timestamp interface can sort one second of data and loop through it.

8.1.2. Raspberry pi testing

After all implementations are done the system is tested. The multiprocessing works but immediately a problem occurred. The writing to the database is significantly slower than the I2C sampling. This meant that even though multiprocessing is used the system has to wait for all values to be written in the database. If this had not been the case the system would acquire data faster than it can write it away to the database leading to a build-up of data. A further problem was that a timestamp was given to the sample when it was written into the database. At first sight, this looks like a solid system but if writing to the database is slow then the timestamps on the data are slow as well. This means that the timestamps connected to the data are not representative of when the data is acquired. These two integration problems have a major impact on the system's performance. Because now there is a big discontinuity between two I2C frames that fetch data the fetched data is stretched out. A schematic overview of the problem is shown in figure 8.1.

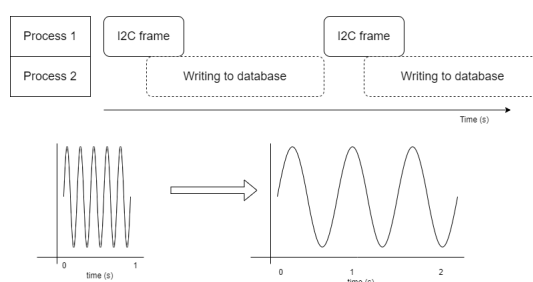


Figure 8.1: Schematic overview of the timing problem and its result

This behaviour is shown in appendix C.1. Because the power quality modules are designed to have 10k samples and some use the assumption that the frequency is around 50Hz, these do not achieve the wanted accuracy. Only the RMS voltage calculation module works with decent accuracy. As the performance of the calculation modules is dependent on the performance of the data acquisition the performance is only calculated with real data. These performances can be found in section 8.2.1

8.2. Interface Implementations

After the interface is designed on a remote device it should be connected to the database and the power quality parameter modules. The database is connected using SQL code. During the login process, the given email and password are stored in the database. If the same email is presented at the login screen the password can be checked correctly. For the interface, the connection is made and a loop is built that loops every second. Now that there is data flowing in the interface a separate file was built with all calculation modules. These modules were written in their own function that had the voltage or/and current data as their input. The interface fetched data from the database and passed it through the calculation modules. They, in return, calculate their respective parameter and pass them back to the interface which displays them on the 'Results' page.

8.2.1. Interface testing

The implementation of the login screen was tested by creating a new table in the database and checking if the usernames and passwords were stored correctly. Figure 8.2 shows three entries in the username and password database. Note that these usernames and passwords are no longer in use. When one of these three usernames was presented to the login screen the username ID was found and the given password was matched with the password in the database. When correct, the user was led into the two-step verification process and the interface.

	UserID	EMAIL	password
☐ Wijzigen Kopiëren Verwijderen	1	husna[REDACTED]@hotmail.com	7777
☐ Wijzigen Kopiëren Verwijderen	2	marten[REDACTED]@live.nl	0000
☐ Wijzigen Kopiëren Verwijderen	3	xeline[REDACTED]@gmail.com	1234

Figure 8.2: Overview of the username and password page

To validate the working of the interface implementations a signal generator was built in Python that acted as the Raspberry Pi. Every second, it wrote a known voltage and current signal to the database. This signal was a sinusoid with a known frequency, with noise and a phase shift between the voltage and current. This way we could validate the implementation in a controlled environment. As expected the interface managed to fetch the data from the database, display it on the voltage and current page and calculate the power quality parameters. It also displayed the harmonics with their respective amplitudes.

9

Risk Assessment

9.1. Introduction to Risk Assessment

The purpose of risk management is to identify potential risks that may occur, their likelihood and impact if they occur. Risk management is carried out during the requirements gathering process and throughout the development of the software, the risk management is reviewed and updated as new risks are identified. Risk management provides visibility of threats to project success, it also reduces the impact of uncertainty on the project and helps to produce robust software [67].

Risk is assessed as a combination of probability and magnitude of consequence if the risk occurs. Risk is defined as the potential for a scenario to occur, along with the associated consequences. The following formula 9.1 is used to calculate risk:

$$Risk = Probability \cdot Impact \quad (9.1)$$

Probability refers to the likelihood of a particular scenario occurring, while consequence refers to the severity associated with that scenario. A scenario refers to a specific set of circumstances that harm the software being developed, and these scenarios are used to evaluate the potential consequences [67].

Risk assessment is not carried out in this chapter with the aim of eliminating or mitigating all risks, but to provide an insight into what type of risk may occur and, if it does occur, how it may be resolved. However, our design decisions are based on the risks with the highest likelihood and impact. In this chapter, the risks are considered specifically when the equipment is installed in a small business.

9.2. Scale Determination

To assess risks, both ordinal and numeric scales are utilized. Numeric scales use numerical values to represent the severity of a certain parameter. On the other hand, an ordinal scale uses labels to indicate the order of parameters. Risk assessment is highly subjective, making it challenging to assign a precise numerical value. It isn't easy to substantiate such numerical assignments, hence using ordinal scales is useful in those cases. Ordinal scales are simpler to understand, and they quickly convey the importance of a parameter. Given the uncertainties and subjective nature inherent in risk assessment, an ordinal scale is used. However, for assessing risks based on a formula, a numerical scale is necessary. Two different scales are used. The numbers used for the numerical scale are highly subjective and depend on factors related to the considered scenario.

To assess the likelihood scale the following scale is used: [very likely, likely, possible, unlikely, very unlikely]. For assessing severity, we will use the following scale: [negligible, minor, moderate, significant, severe]. In the table, the risk assessment matrices are shown. In this table, for each possible situation, the likelihood and severity are intersected, identifying the associated risk.

To assess the risk itself using the formula, the probability is expressed between 0 and 1. Therefore,

the probability scale from 0.0 to 1.0 is used. A score of 0.0 signifies no probability that a certain scenario will happen, while 1.0 indicates a 100% likelihood. For impact, A scale from 1 to 5 is used. A score of 1 denotes shallow impact, while 5 signifies a very high impact on the system if a certain scenario occurs. A 1 to 5 scale is used because it allows for quick comprehension of severity levels associated with each risk, and people intuitively grasp the concept of such a scale. This scale provides sufficient detail without being overly complex.

The two ordinal scales are integrated with the numerical scales, each comprising five instances. Within the likelihood scale, instances such as 'Very Likely' correspond to a probability of 1.0, 'Likely' to a probability of 0.8, 'Possible' to a likelihood of 0.4, and 'Unlikely' to a probability of 0.2. Concurrently, the ordinal scale representing severity aligns with the numerical scale as follows: 'Negligible' corresponds to an impact of 1, 'Minor' to an impact of 2, 'Moderate' to an impact of 3, 'Significant' to an impact of 4, and 'Severe' to an impact of 5.

Severity levels indicate how much a scenario affects the system. Severe impact means the system is completely down, making the product inaccessible. Significant impact is for data breaches or spreading false information due to security issues. Moderate impact causes noticeable but manageable disruptions. Minor impact means small disturbances and the system stays mostly unaffected. Negligible impact means minimal effect, ensuring the system operates smoothly for an uninterrupted user experience.

9.3. Risk Identification

During the requirements gathering process, potential risks were identified and requirements were defined based on these risks. Below are all the identified scenarios that were encountered during the development process of the software development concerning small business and home environments. This device is intended for use in small businesses. However, it can also be used in households.

9.3.1. Risk Assessment Raspberry Pi

Attack on Raspberry Pi

Attackers could exploit vulnerabilities in the operating system or software running on Raspberry Pi 4. Keeping Raspberry Pi's software up to date, and following the rules for securing Linux-based systems.

Network exposure

If Raspberry Pi is connected to a network, it may be exposed to external threats such as network-based attacks. Configuring firewalls, disabling unnecessary services and using network segmentation to limit exposure.

Unauthorized Access and physical access

Unauthorized users gaining access to the Raspberry Pi may manipulate the I2C communication or extract sensitive data. Physical access to the Raspberry Pi may lead to tampering with the I2C communication, potentially disrupting the ADC readings.

Man-in-the-Middle Attacks

Man-in-the-middle attacks on the I2C communication can inject malicious commands. The attacker positions themselves between the Raspberry Pi (master) and the ADC (slave) on the I2C bus. The attacker listens to the I2C communication, gaining insight into the data being exchanged. The attacker may modify the data in transit, the attacker might inject their own commands into the communication.

9.3.2. Risk Assessment Communication Protocol I2C

I²C bus snooping

Attackers could perform bus snooping on the I2C bus, intercepting and analyzing communication between the Raspberry Pi and the ADC.

Clock Signal Manipulation

The manipulation of the clock signal of I²C can distort the communication between Raspberry Pi and the ADC.

9.3.3. Risk Assessment Interface Authentication

Accountability

SQL injection

It is possible that SQL injections can happen. If this happens, the database can be destroyed and so all the data stored in the database will be lost. The user can enter an SQL statement in the input fields where we ask for email and password. If the attacker fills in a SQL statement, unknowingly this will run on our database, and destroy our database or steal information.

Lack of Rate Limiting

Lack of rate limiting happens when there is no restriction on the number of login attempts, a user can make within a specific period. This lack of control opens up the application to attacks. An attacker repeatedly can try different combinations of usernames and passwords until they gain unauthorized access to an account. Without rate limiting, attackers can automate these attempts, trying thousands of combinations in a short period.

External Service Dependency

The application relies on an external SMTP service, Brevo, which introduces dependency. Any security vulnerabilities and disruptions in the service of Brevo could impact the application.

Email Spoofing

Attackers can send emails that appear to come from the application's domain, this is called spoofing. This can happen, for example, when data is leaked from Brevo.

9.4. Risk Analysis

In tables 9.1, 9.2 and 9.3 the risk assessment matrices are shown. All the identified risks have a low to medium risk to happen but if it happens then they have a severe to significant impact.

In the event of a possible attack on the Raspberry Pi, there's a 60% chance with a severe impact, resulting in a risk score of 3.0. Network exposure carries a similar 60% probability and a significant impact, yielding a risk score of 2.4. Unauthorized access, though unlikely (20% probability), poses severe consequences, with a risk score of 1.0. Man-in-the-middle attacks, very unlikely (20% probability), still present a significant risk, with a score of 0.8. Security measures, software updates, and vigilance are key mitigations.

Two scenarios for I2Cbus snooping and clock signal manipulation both show a very unlikely 20% probability and a significant impact, resulting in a risk score of 0.8. Robust security measures, communication monitoring, and prevention of unauthorized access are essential for mitigating these risks in small business and home environments.

In evaluating interface authentication risks, the likelihood of SQL Injection, rated as likely with a severe impact (4.0), emphasizes the need for robust defenses against database compromise. Additionally, the risk of Lack of Rate Limiting, also likely with significant consequences (3.2), underscores the importance of implementing stringent controls to prevent unauthorized access attempts. Addressing External Service Dependency risks, categorized as likely with a significant impact (3.2), requires a focused approach to fortify dependencies and enhance system reliability. Proactive measures in these areas are crucial for maintaining authentication security.

Risk Assessment Matrix for Raspberry Pi:

Scenario	Likelihood	Severity	Probability	Impact	Risk
Attack on Raspberry Pi	Possible	Severe	0.6	5	3.0
Network Exposure	Possible	Significant	0.6	4	2.4
Unauthorized Access & Physical Access	Unlikely	Severe	0.2	5	1.0
Man-in-the-Middle Attacks	Very unlikely	Significant	0.2	4	0.8

Table 9.1: Risk Assessment for Raspberry Pi

Risk Assessment Matrix for Communication Protocol I2C:

Scenario	Likelihood	Severity	Probability	Impact	Risk
I2C Bus Snooping	Very unlikely	Significant	0.2	4	0.8
Clock Signal Manipulation	Very unlikely	Significant	0.2	4	0.8

Table 9.2: Risk Assessment for Communication Protocol I2C

Risk Assessment Matrix for Interface Authentication:

Scenario	Likelihood	Severity	Probability	Impact	Risk
SQL Injection	Likely	Severe	0.8	5	4.0
Lack of Rate Limiting	Likely	Significant	0.8	4	3.2
External Service Dependency	Likely	Significant	0.8	4	3.2
Man-in-the-Middle Attacks	Possible	Significant	0.6	4	2.4
Email Spoofing	Possible	Significant	0.6	4	2.4

Table 9.3: Risk Assessment for Interface Authentication

10

Conclusion

In conclusion, this thesis presents the development and design of the Power Quality Analyzer. This device has successfully achieved low cost, user-friendliness, and the capability to monitor power quality in electrical systems. The thesis provides insight into the thought process employed to meet all the specified requirements. Now, the question that arises is: have we met all our requirements?

The main requirements of the project were to capture the waveforms presented by Subgroup 1, store them in a database, perform calculations on them, and display them on a user interface. These main requirements were met. A communication protocol between hardware and software was created, a MariaDB database was built to store the data systematically, six power quality calculation modules were created, and an easy-to-use interface was designed and built to display all the data.

However, not every requirement was met. One of the requirements was to sample data continuously. This requirement was not met because there were problems with the database writing speed. This meant that although sampling was fast enough and could be done continuously, writing the data to the database took more processing time than sampling. This created a discontinuity between waves and made the data in the database unrepresentative of the incoming waves. This had a major impact on the accuracy of the power quality calculation modules, which expected a wave without discontinuities and a higher sampling rate.

To validate our other requirements, a signal generator was built that could write to the database at a sufficient rate. All components were tested against this generator and their accuracy was validated.

In conclusion, although all of our components were designed, built, and met their respective requirements, the integration of the entire system did not meet the requirements of a Class S power quality analyzer. This thesis describes the design process and the decisions made to reach the final goal. The discussion provided suggestions on how to enhance the developed power quality analyzer to meet the class-S requirements.

11

Discussion

After the implementation and the testing of the overall system we can look back at our requirements and see if we met them and what could be improved.

11.1. Requirements fulfillment evaluation

11.1.1. Data acquisition

The data acquisition is the module where the most improvements can be made. As can be seen in 4.1 the sampling is fast enough to meet the requirement of a 10k sampling rate. Also looking at the plots made there are no big error samples. However, the big problem is the integration between sampling and the database. Writing to the database is significantly slower than data acquisition. This combined with the fact that the data sampling cannot sample a full second of data at the same time causes a big discontinuity between I2C frames. This combined with the wave stretching caused by labeling the samples with the time they are written in the database gives a skewed representation of the grid voltage and current.

11.1.2. Database

The database that has been set up on the Raspberry Pi is able to store data in a structured and organized manner. However, this could be improved by making it possible to write to multiple databases, which could make it more organized. But, this would cost more CPU usage and that was something that could not be afforded since there was already not that much CPU to work with. The database receives data from the modules and saves them to the tables created for that specific module. The security of the database can be improved further by making use of data-at rest encryption. It was managed to access the database remotely on the same local network. Remote access via the web could only be realised on the Raspberry Pi at home, by creating a VPN server on the Raspberry Pi.

11.1.3. Calculation modules

As can be seen in F all modules are designed and work with the generated data. The peak-detection algorithm was found to not be as robust as was preferred. Due to time limitations, the peak-detection was not improved further, as the VRMS algorithm was the more robust option.

11.1.4. Interface

The implemented system meets the specified requirements for both the login process and the power quality monitoring interface. A robust security layer with two-step verification has been built into the software. Nevertheless, there is a problem that company emails enter the spam folder. To address this issue, we plan to improve the email detection system to ensure that our communications are consistently identified as safe. The user interface has a highly professional look and feel with intuitive navigation controls that provide an intuitive experience. In addition, the interface effectively displays real-time power quality parameters that are updated every second. Users can create an account, log in, and change their password.

11.2. Future Improvements

11.2.1. Database writing

Finding a faster way to store the data would greatly improve the quality of this project. Another solution would be to sample a second of data by saving the samples in arrays within the program itself. Then write them all to the database at the same time. This way you could have one second of data, then write to the database for 5 seconds and gather a new second of data. This lowers the resolution and thus is a worse way to represent the grid but would work with our interface and calculation modules. Writing to a database could be made faster by writing to a database on the cloud, instead of on the Raspberry Pi. However, this is a consideration that should be made, since there is no free database cloud server that can be used for such a project. Another way to be able to write faster to the database is to have a Raspberry Pi with a bigger RAM.

11.2.2. Database security

The database could be made more secure by implementing data-at-rest encryption. This was not possible to integrate during this project, because it was on the enterprise version of MariaDB, which is only available with a subscription. Due to university restrictions on port forwarding on the university network, remote access via the web using VPN could only be tested at home on another Raspberry Pi. Making the Raspberry Pi remotely accessible would improve the quality of this project.

11.2.3. Implementation of physical storage

As Raspberry Pi's are known to crash once in a while it would be an improvement to connect a physical hard drive and write the contents of the database to reduce the risk of losing all accumulated data.

11.2.4. Remote Accessibility

It was managed to create a VPN server on the Raspberry Pi, which was only used for testing, making it possible to connect to it via the web. It would have been a great addition to this project to test this on the actual Raspberry Pi used during the project. The VPN protocol (Open VPN) that has been used, was found to be slow. In the future, it would be better to choose a faster protocol that is also available on widely used operating systems.

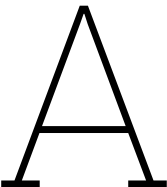
References

- [1] J.G. Dougherty and W.L. Stebbins. “Power quality: a utility and industry perspective”. In: 1997 IEEE Annual Textile, Fiber and Film Industry Technical Conference. 1997, 5. 10 pp.-. DOI: 10.1109/TEXCON.1997.598528.
- [2] “IEEE Recommended Practice for Monitoring Electric Power Quality”. In: IEEE Std 1159-2019 (Revision of IEEE Std 1159-2009) (2019), pp. 1–98. DOI: 10.1109/IEEESTD.2019.8796486.
- [3] S. Bhattacharyya, J. M. A. Myrzik, and W. L. Kling. “Consequences of poor power quality - an overview”. In: 2007 42nd International Universities Power Engineering Conference. 2007, pp. 651–656. DOI: 10.1109/UPEC.2007.4469025.
- [4] Xiaodong Liang. “Emerging Power Quality Challenges Due to Integration of Renewable Energy Sources”. In: IEEE Transactions on Industry Applications 53.2 (2017), pp. 855–866. DOI: 10.1109/TIA.2016.2626253.
- [5] R Janani et al. “Low-cost Power Quality analyzer for academic applications”. In: Student Journal of Electrical and Electronics Engineering 1.1 (2015).
- [6] Yogesh V. Mangalekar and R. T. Ugale. “Electric motor power quality assessment using LabVIEW based smart power analyzer”. In: 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS). 2015, pp. 318–323. DOI: 10.1109/RAICS.2015.7488435.
- [7] Florin-Ciprian Argatu et al. “Power Quality Analyzer for Smart Grid-Smart Home Applications”. In: 2019 54th International Universities Power Engineering Conference (UPEC). 2019, pp. 1–4. DOI: 10.1109/UPEC.2019.8893501.
- [8] Duan Medved et al. “Project design of the electric power quality analyzer using an open-source platform”. In: 2018 International IEEE Conference and Workshop in Óbuda on Electrical and Power Engineering (CANDO-EPE). 2018, pp. 000169–000172. DOI: 10.1109/CANDO-EPE.2018.8601160.
- [9] Miguel A. Oliván et al. “A Computer-Based IEC 61850 Sampled Values Analyzer for Parallel Power Quality Analysis”. In: IEEE Transactions on Instrumentation and Measurement 71 (2022), pp. 1–8. DOI: 10.1109/TIM.2022.3175044.
- [10] Eduardo Viciania et al. “OpenZmeter: An Efficient Low-Cost Energy Smart Meter and Power Quality Analyzer”. In: Sustainability 10.11 (2018). ISSN: 2071-1050. DOI: 10.3390/su10114038. URL: <https://www.mdpi.com/2071-1050/10/11/4038>.
- [11] KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART) User Guide. Texas Instruments. November 2010.
- [12] Eric Peña and Mary Grace Legaspi. “Uart: A hardware communication protocol understanding universal asynchronous receiver/transmitter”. In: Visit Analog 54.4 (2020).
- [13] Mihai Daraban et al. “Protocol over UART for Real-Time Applications”. In: 2021 IEEE 27th International Symposium for Design and Technology in Electronic Packaging (SIITME). 2021, pp. 85–88. DOI: 10.1109/SIITME53254.2021.9663605.
- [14] Jinrong Chen and Shengqiu Huang. “Analysis and Comparison of UART, SPI and I2C”. In: 2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA). 2023, pp. 272–276. DOI: 10.1109/EEBDA56825.2023.10090677.
- [15] Serial Peripheral Interface (SPI). Microchip. 2003.
- [16] Frederic Leens. “An introduction to I2C and SPI protocols”. In: IEEE Instrumentation Measurement Magazine 12.1 (2009), pp. 8–13. DOI: 10.1109/MIM.2009.4762946.
- [17] Xiao Chun Tian et al. “Design and Implementation of SPI Communication Based-On FPGA”. In: vol. 291. Advanced Materials Research. Trans Tech Publications Ltd, Sept. 2011, pp. 2658–2661. DOI: 10.4028/www.scientific.net/AMR.291-294.2658.

- [18] AN10216-01 I2C MANUAL. Philips Semiconductors. March 2003.
- [19] A Basic Guide to I2C. Texas Instruments. November 2022.
- [20] Leandro Trujillo, Osamu Saotome, and Johnny Öberg. “Dependable I2C Communication with a FPGA”. In: Proceedings of the 7th Brazilian Technology Symposium (BTSym’21). Ed. by Yuzo Iano et al. Cham: Springer International Publishing, 2022, pp. 383–395.
- [21] ADS7142-Q1 Automotive, 2-Channel, 12-Bit, 140-kSPS, I2C-Compatible ADC With Programmable Threshold and Host Wake-Up Features. Texas Instruments. NOVEMBER 2018–REVISED OCTOBER 2019.
- [22] pigpio library description. <https://abyz.me.uk/rpi/pigpio/index.html>. Accessed: 11-12-2023.
- [23] David Trillo-Montero et al. “Design and Development of a Relational Database Management System (RDBMS) with Open Source Tools for the Processing of Data Monitored in a Set of Photovoltaic (PV) Plants”. In: Applied Sciences 13.3 (2023). ISSN: 2076-3417. DOI: 10.3390/app13031357. URL: <https://www.mdpi.com/2076-3417/13/3/1357>.
- [24] André Eugênio Lazzaretti et al. “A Monitoring System for Online Fault Detection and Classification in Photovoltaic Plants”. In: Sensors 20.17 (2020). ISSN: 1424-8220. DOI: 10.3390/s20174688. URL: <https://www.mdpi.com/1424-8220/20/17/4688>.
- [25] J Galvão et al. “CPS, Sustainability and Efficiency in an Aquaponic Greenhouse”. In: Renewable Energy and Power Quality Journal 20 (2022), pp. 210–215.
- [26] MariaDB Corporation. MariaDB File Key Management Encryption Plugin. Accessed on: Date of Access. 2023. URL: <https://mariadb.com/kb/en/file-key-management-encryption-plugin/>.
- [27] MariaDB Corporation. MariaDB InnoDB Encryption Overview. Accessed on: Date of Access. 2023. URL: <https://mariadb.com/kb/en/innodb-encryption-overview/>.
- [28] MariaDB Corporation. MariaDB Aria Encryption Overview. Accessed on: Date of Access. 2023. URL: <https://mariadb.com/kb/en/aria-encryption-overview/>.
- [29] MariaDB Corporation. MariaDB Secure Connections Overview. Accessed on: Date of Access. 2023. URL: <https://mariadb.com/kb/en/secure-connections-overview/>.
- [30] Satoshi Uda and Mikifumi Shikida. “Challenges of Deploying PKI Based Client Digital Certification”. In: Proceedings of the 2016 ACM SIGUCCS Annual Conference. SIGUCCS ’16. Denver, Colorado, USA: Association for Computing Machinery, 2016, pp. 55–60. ISBN: 9781450340953. DOI: 10.1145/2974927.2974938. URL: <https://doi-org.tudelft.idm.oclc.org/10.1145/2974927.2974938>.
- [31] Dimas Febriyan Priambodo, Amiruddin, and Nanang Trianto. “Hardening a Work from Home Network with Wireguard and Suricata”. In: 2021 International Conference on Computer Science and Engineering (IC2SE). Vol. 1. 2021, pp. 1–4. DOI: 10.1109/IC2SE52832.2021.9791983.
- [32] Pi My Life Up. Raspberry Pi VPN Server. Accessed: 01-01-2024. 2024. URL: <https://pimylifeup.com/raspberry-pi-vpn-server/>.
- [33] Whitson Gordon. How to Create a VPN Server with Raspberry Pi. Accessed: 01-01-2024. 2024. URL: <https://www.pcmag.com/how-to/how-to-create-a-vpn-server-with-raspberry-pi>.
- [34] phpMyAdmin Development Team. phpMyAdmin. Accessed on: Date of Access. 2023. URL: <https://www.phpmyadmin.net/#:~:text=phpMyAdmin%20is%20a%20free%20software%20tool%20written%20in,wide%20range%20of%20operations%20on%20MySQL%20and%20MariaDB..>
- [35] Michael Kofler. “phpMyAdmin”. In: The Definitive Guide to MySQL5 (2005), pp. 87–116.
- [36] Anthony Vance, Paul Benjamin Lowry, and Dennis Eggett. “Increasing Accountability Through User-Interface Design Artifacts: A New Approach to Addressing the Problem of Access-Policy Violations”. In: MIS Quarterly 39.2 (2015), pp. 345–366. ISSN: 02767783, 21629730. URL: <https://www.jstor.org/stable/26628357> (visited on 01/14/2024).
- [37] Qasim Uddin. Python GUI : Difference between tkinter, PyQt, and Kivy. 2023. URL: <https://medium.com/@qasim.coder/python-gui-smackdown-unleashing-the-power-of-tkinter-pyqt-and-kivy-e7b05d0e862>.

- [38] The Qt Company Ltd. Qt Designer Manual. 2023. URL: <https://doc.qt.io/qt-6/qtdesigner-manual.html>.
- [39] Brevo. Security Privacy: Protecting Your Data. 2024. URL: <https://www.brevo.com/plus/security/>.
- [40] Peilun Ouyang et al. "Design of Intelligent Drilling System Software Framework and Data Architecture Based on MVC Pattern". In: 2019 Chinese Control Conference (CCC). 2019, pp. 7075–7078. DOI: 10.23919/ChiCC.2019.8866460.
- [41] E.L. Owen. "The origins of 60-Hz as a power frequency". In: IEEE Industry Applications Magazine 3.6 (1997), pp. 8–14. DOI: 10.1109/2943.628099.
- [42] Netbeheer Nederland. "Power-Generating Modules compliance verification". In: (2021).
- [43] Braham Ferreira Wim van der Merwe. The principles of electronic and electromagnetic power conversion: a systems approach. Wiley-IEEE Press, 2013. ISBN: 9781118798850.
- [44] Matthew Sadiku Charles K Alexander. Fundamentals of Electric Circuits. 6. McGraw-Hill, 2017. ISBN: 978-0-07-802822-9.
- [45] Pieter Schavemaker and Lou Sluis. Electrical Power System Essentials, 2nd edition. July 2017. ISBN: 978-1-118-80347-9.
- [46] S.M. Deckmann and A.A. Ferrira. "About voltage sags and swells analysis". In: 10th International Conference on Harmonics and Quality of Power. Proceedings (Cat. No.02EX630). Vol. 1. 2002, 144–148 vol.1. DOI: 10.1109/ICHQP.2002.1221423.
- [47] Raj Naidoo and Pragasen Pillay. "A New Method of Voltage Sag and Swell Detection". In: IEEE Transactions on Power Delivery 22.2 (2007), pp. 1056–1063. DOI: 10.1109/TPWRD.2007.893185.
- [48] Vima P Mali, RL Chakrasali, and KS Aprameya. "A technical investigation of voltage sag". In: American Journal of Engineering Research (AJER) 4.10 (2015), pp. 60–68.
- [49] Alvaro Furlani Bastos et al. "Accurate Identification of Point-on-Wave Inception and Recovery Instants of Voltage Sags and Swells". In: IEEE Transactions on Power Delivery 34.2 (2019), pp. 551–560. DOI: 10.1109/TPWRD.2018.2876682.
- [50] Padmanabh Thakur and AK Singh. "A novel way to quantify the magnitude of voltage sag". In: Electrical Engineering 95 (2013), pp. 331–340.
- [51] Muhamad Mansor and Nasrudin Abdul Rahim. "Voltage sag detection - A survey". In: 2009 International Conference for Technical Postgraduates (TECHPOS). 2009, pp. 1–6. DOI: 10.1109/TECHPOS.2009.5412088.
- [52] C. Venkatesh, D. V. S. S. Siva Sarma, and M. Sydulu. "Mitigation of Voltage Sag/Swell Using Peak Detector Based Pulse Width Modulation Switched Autotransformer". In: Electric Power Components and Systems 39.11 (2011), pp. 1117–1133. DOI: 10.1080/15325008.2011.559188.
- [53] K. Uma Rao P. Meena and Ravishankar Deekshit. "A Simple Method For Real-Time Detection Of Voltage Sags and Swells in Practical Loads". In: EPE Journal 21.3 (2011), pp. 33–40. DOI: 10.1080/09398368.2011.11463801.
- [54] Vijay G Neve, Pallavi V Pullawar, and Chetan H Kidile. "Power quality issues & detection methods for voltage sag and swell". In: International Journal of Research in Biosciences, Agriculture and Technology (2020), pp. 71–79.
- [55] S. Bhattacharyya, J. M. A. Myrzik, and W. L. Kling. "Consequences of poor power quality - an overview". In: 2007 42nd International Universities Power Engineering Conference. 2007, pp. 651–656. DOI: 10.1109/UPEC.2007.4469025.
- [56] A. Rash. "Harmonics-what are they, how to measure them and how to solve the problem (in connection with standards IEEE 1159-1995 and IEEE 519-1992)". In: Proceedings of 19th Convention of Electrical and Electronics Engineers in Israel. 1996, pp. 83–86. DOI: 10.1109/EEIS.1996.566898.
- [57] J.C. Das. Power System Harmonics and Passive Filter Designs. Wiley-IEEE Press, March 2015. ISBN: 978-1-118-86162-2.
- [58] P. S. Sanjan et al. "Enhancement of Power Quality in Domestic Loads Using Harmonic Filters". In: IEEE Access 8 (2020), pp. 197730–197744. DOI: 10.1109/ACCESS.2020.3034734.

- [59] W.A. Maslowski. “Harmonics in power systems”. In: Proceedings:Electrical Electronics Insulation Conference and Electrical Manufacturing Coil Winding Conference. 1995, pp. 37–44. DOI: 10.1109/EEIC.1995.482334.
- [60] The Numpy community. Discrete Fourier Transform (numpy.fft). 2022. URL: <https://numpy.org/doc/stable/reference/routines.fft.html>.
- [61] Michael Cerna and Audrey F. Harvey. The Fundamentals of FFT-Based Signal Analysis and Measurement. 2020. URL: https://www.sjsu.edu/people/burford.furman/docs/me120/FFT_tutorial_NI.pdf.
- [62] Siemens. What is the Fourier Transform? 2019. URL: <https://community.sw.siemens.com/s/article/what-is-the-fourier-transform>.
- [63] J.L. Gonzalez M. Gasior. “https://mgasior.web.cern.ch/pap/FFT_resolution.pdf”. In: EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH ORGANISATION EUROPEENNE POUR LA RECHERCHE NUCLEAIRE. AB-Note-2004-021 BDI. 2004, pp. 37–44.
- [64] François Georges Germain Kurt James Werner. “Sinusoidal Parameter Estimation Using Quadratic Interpolation around Power-Scaled Magnitude Spectrum Peaks”. In: 2016, pp. 186–191. DOI: <https://doi.org/10.3390/app6100306>.
- [65] K. Sajjad, G. Kalsoom, and A. Mughal. “Simplified THD measurement and analysis for electronic power inverters”. In: 2015 12th International Bhurban Conference on Applied Sciences and Technology (IBCAST). 2015, pp. 186–191. DOI: 10.1109/IBCAST.2015.7058502.
- [66] Siemens. What is Total Harmonic Distortion (THD). 2020. URL: <https://community.sw.siemens.com/s/article/what-is-total-harmonic-distortion-thd>.
- [67] Terje Aven. “Risk assessment and risk management: Review of recent advances on their foundation”. In: European Journal of Operational Research 253.1 (2016), pp. 1–13. ISSN: 0377-2217. DOI: <https://doi.org/10.1016/j.ejor.2015.12.023>. URL: <https://www.sciencedirect.com/science/article/pii/S0377221715011479>.



Data Communication Protocol

A.1. Full I2C frame

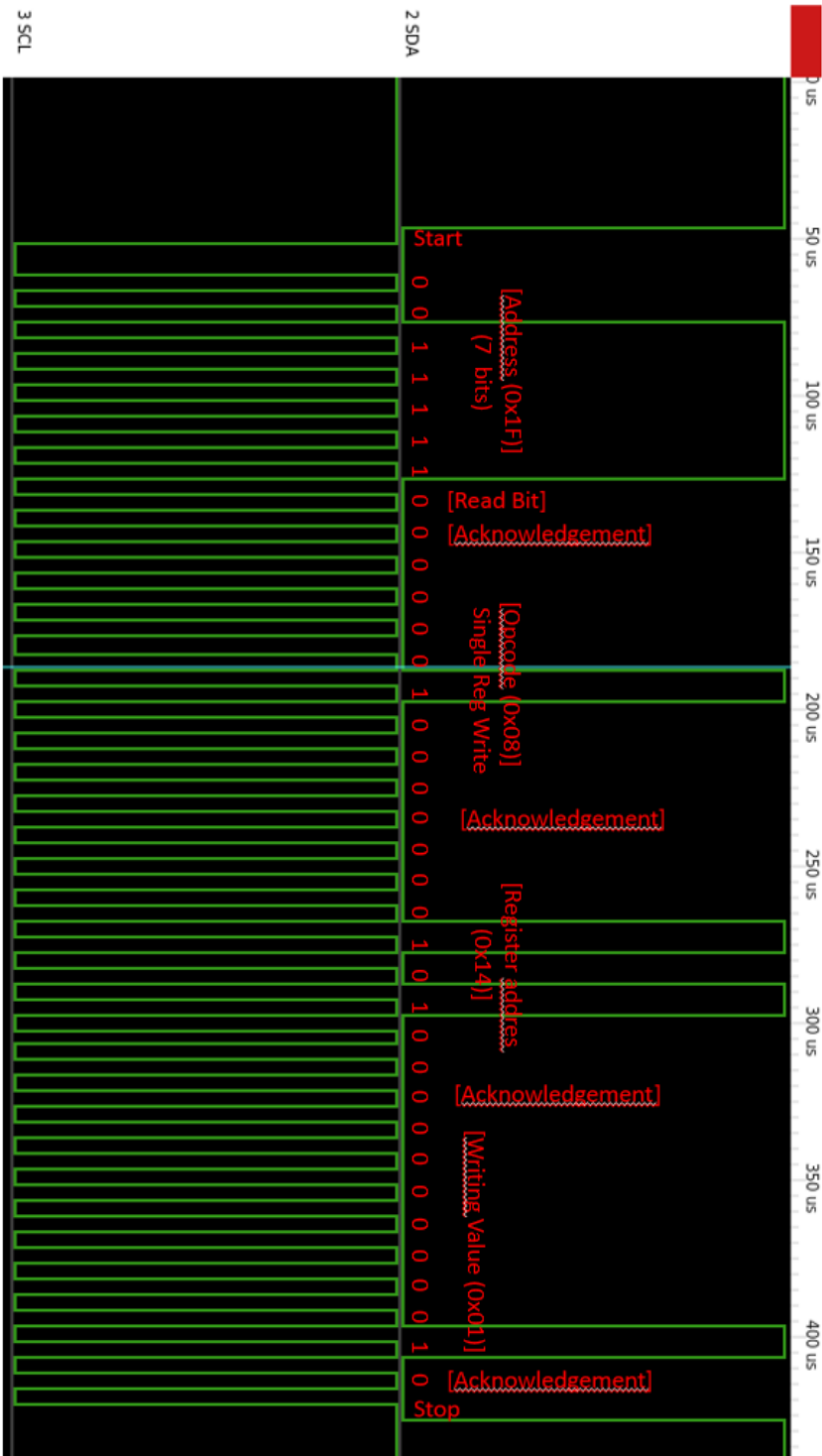


Figure A.1: A written out I2C frame

A.2. Clock stretching

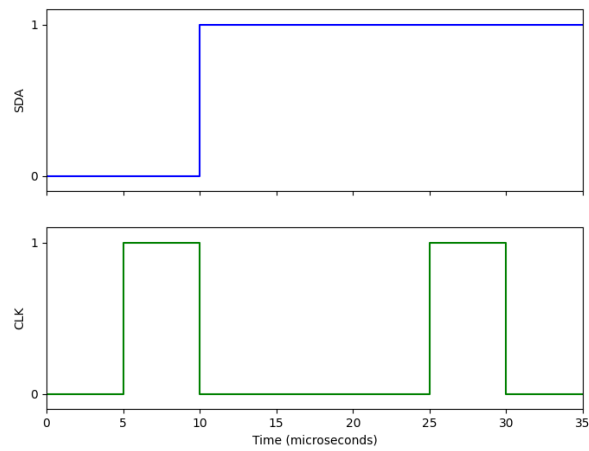


Figure A.2: Clock stretching

A.3. Setup different modes

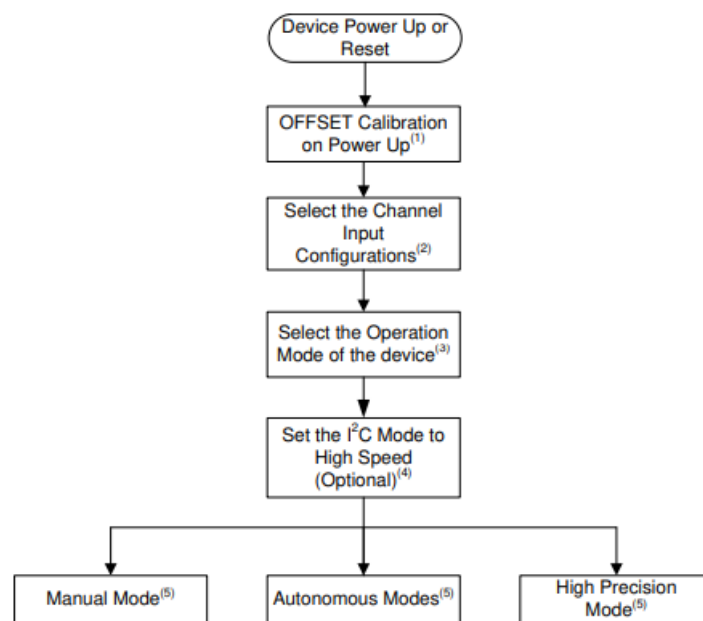


Figure A.3: Overview of steps to setup Manual mode

A.4. Manual Mode schematic

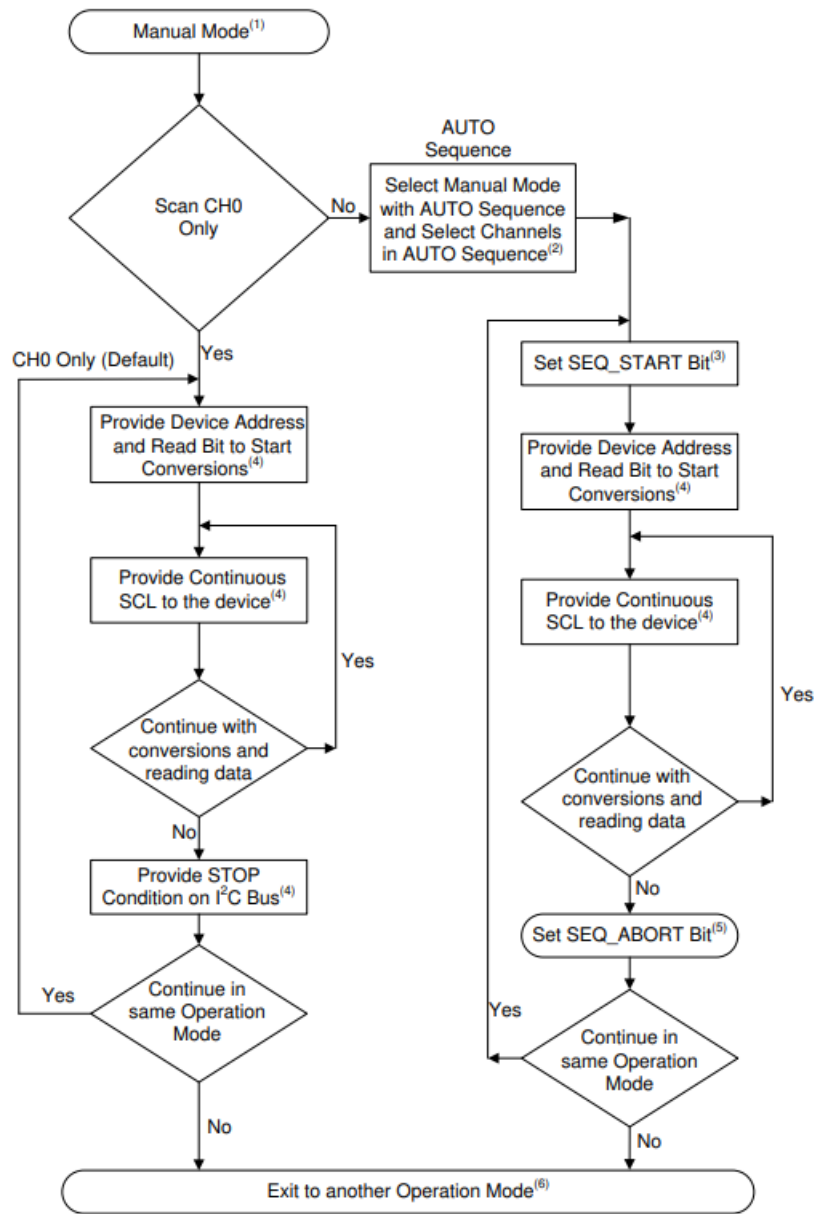


Figure A.4: Overview of steps sample data in Manual mode

A.5. Binary to numerical conversion

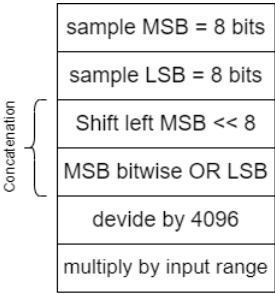


Figure A.5: Schematic overview of the binary to numerical conversion

B

Sampled plots

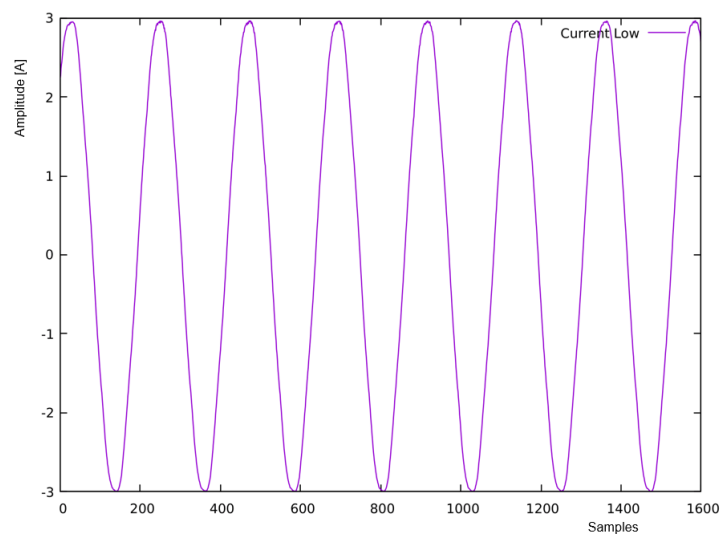


Figure B.1: The current through an industrial worklight (Low current circuit)

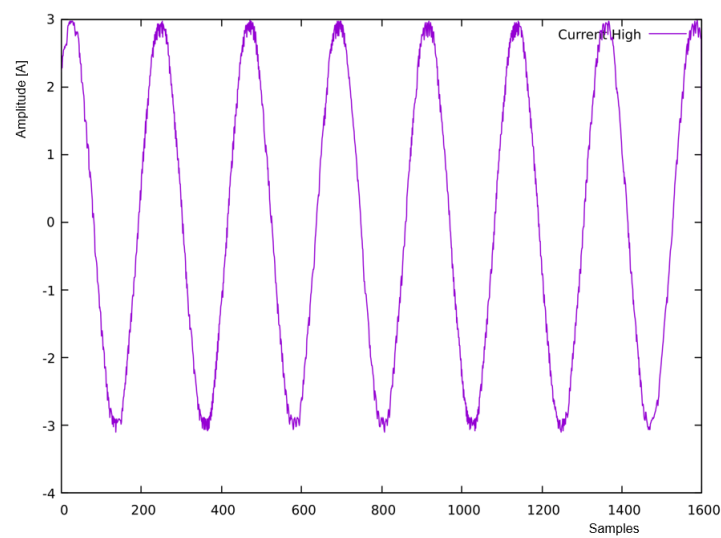


Figure B.2: The current through an industrial worklight (High current circuit)

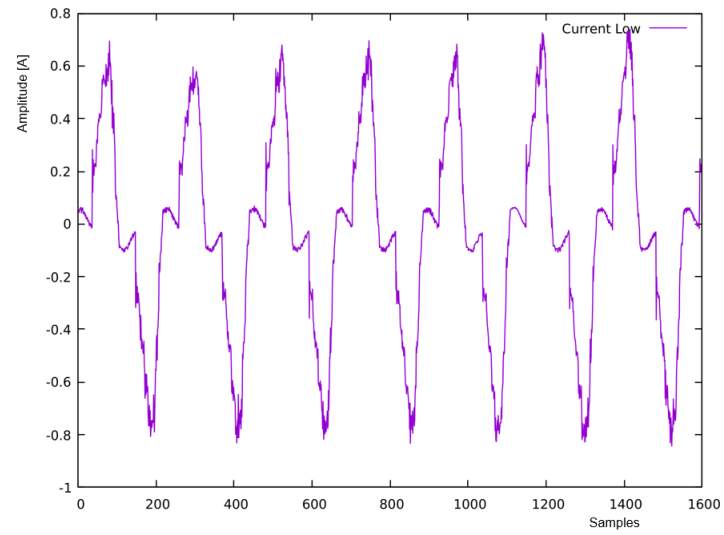


Figure B.3: The current through a laptop charger (Low current circuit)

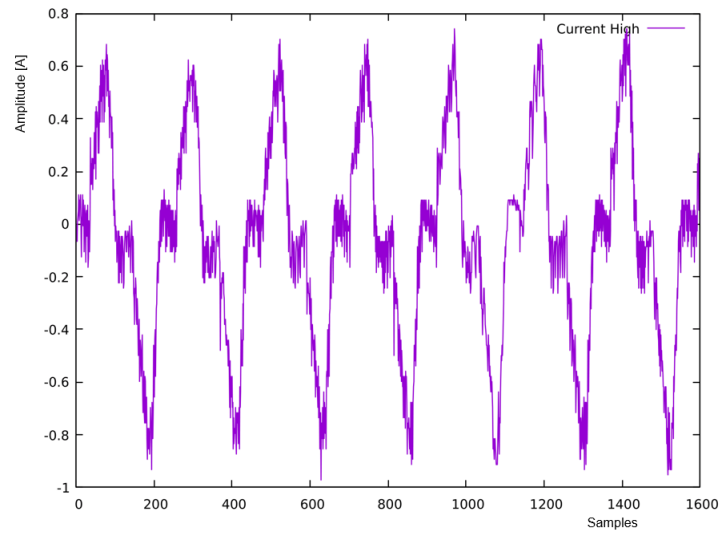


Figure B.4: The current through a laptop charger (High current circuit)

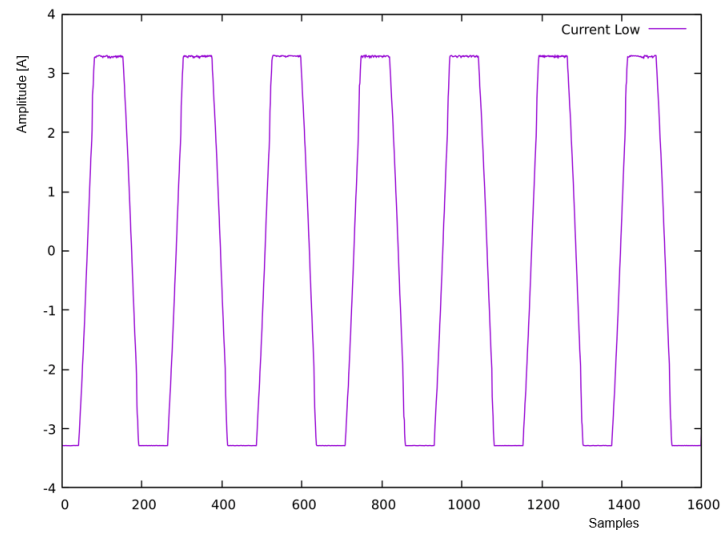


Figure B.5: The current through a power brick with two worklights and two laptop chargers(Low current circuit)

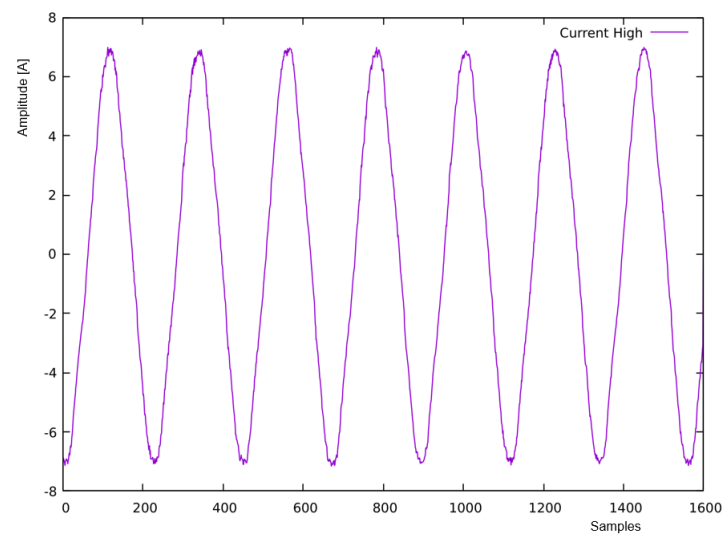
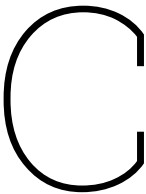


Figure B.6: The current through a power brick with two worklights and two laptop chargers(High current circuit)



System Integration

C.1. User interface after integration with real data

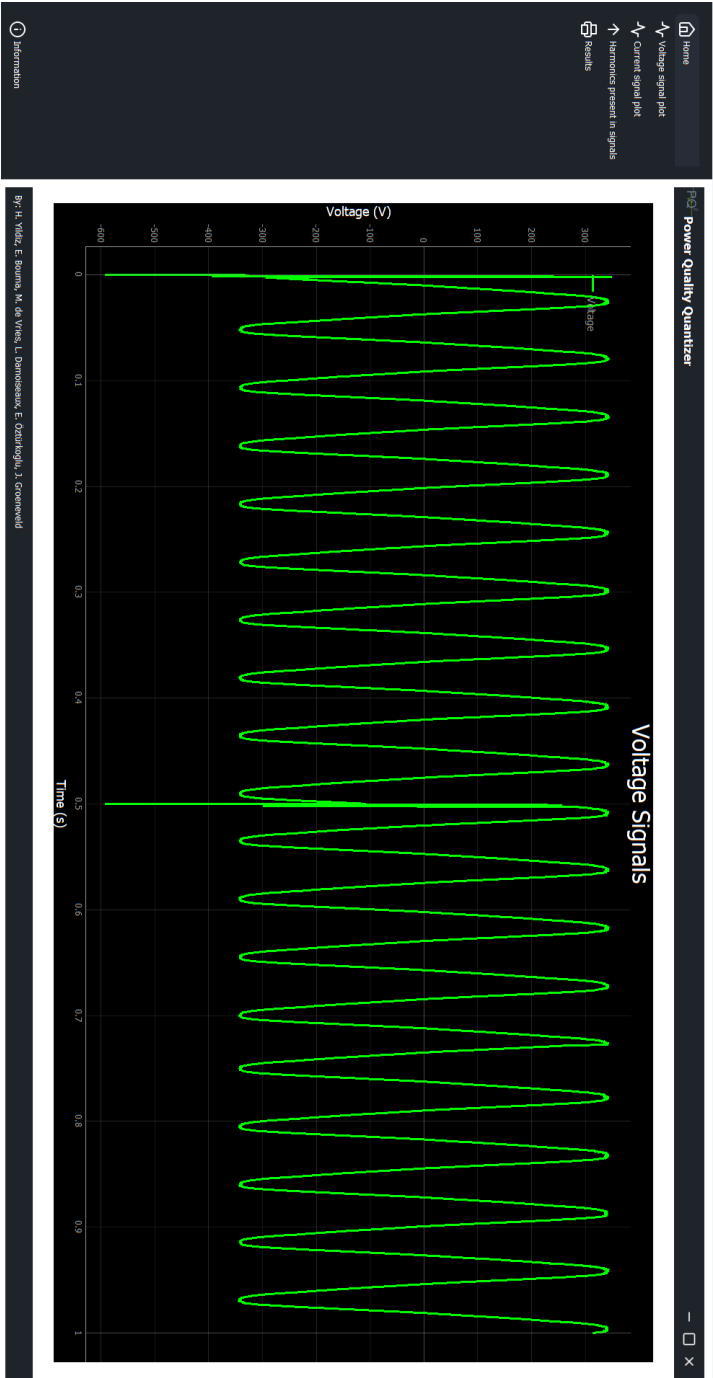


Figure C.1: The voltage plot with real sampled data

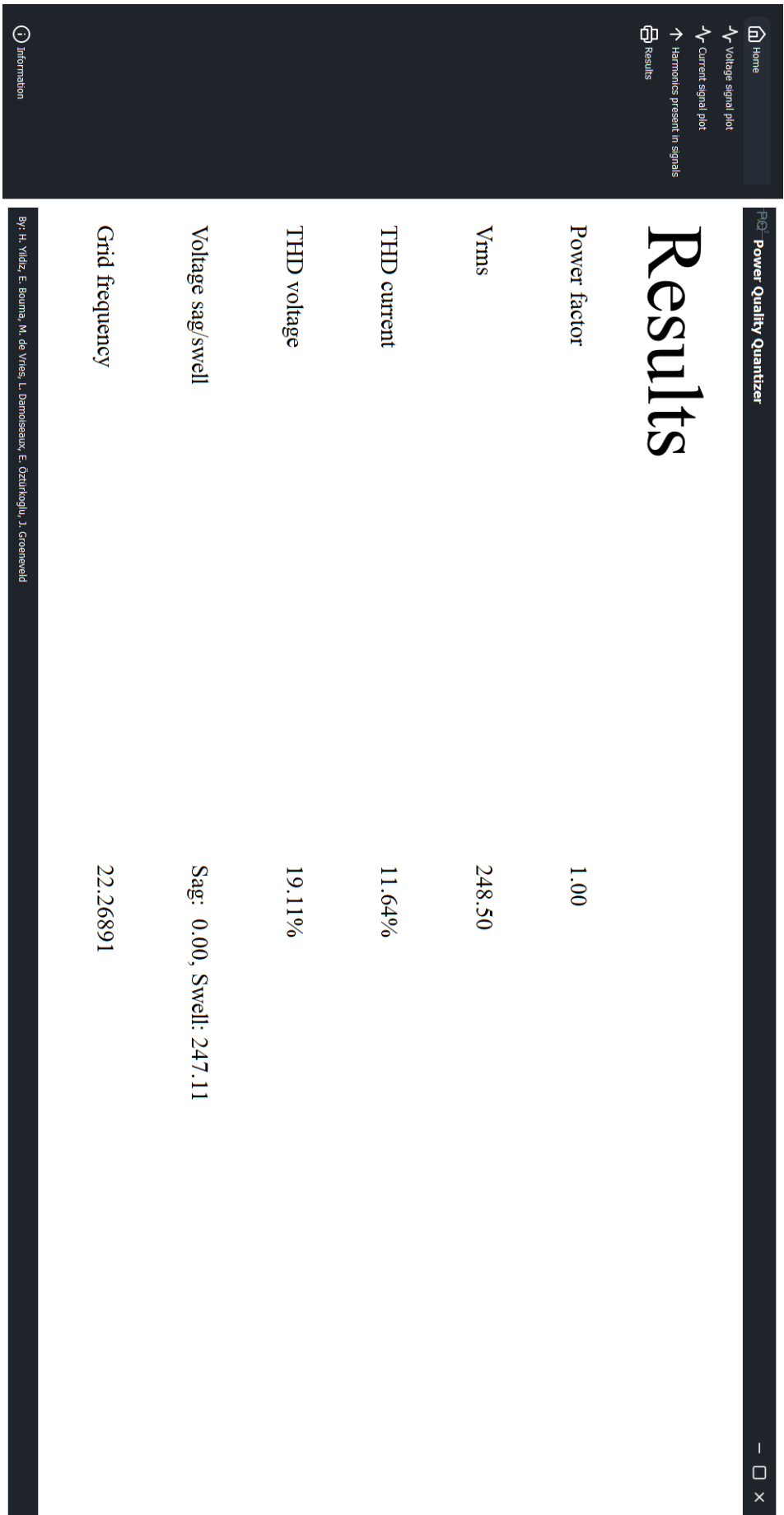


Figure C.2: The results page with real sampled data

C.2. User interface after integration with generated data

Table C.1: Parameters of the signal generator signals

Parameter	Values
frequency	50.25 Hz
Grid Amplitude	$325 V_p / 229.80 V_{RMS}$
Voltage harmonic	2^{nd} harmonic with amplitude 50V
Current amplitude	3.0 A
Phase difference	$\frac{\pi}{2}$
Sags/Swells	None

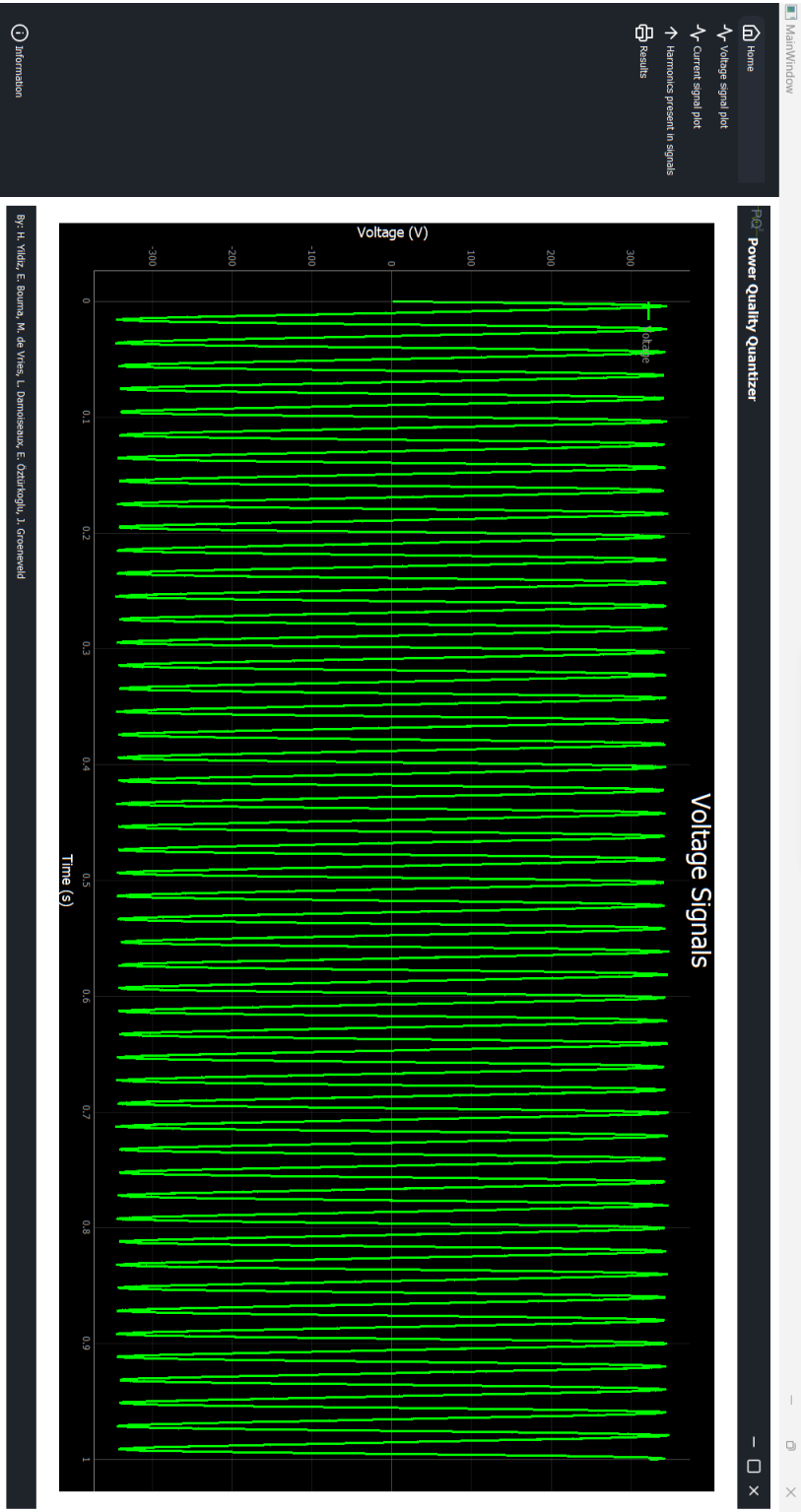


Figure C.3: The voltage plot with generated sampled data

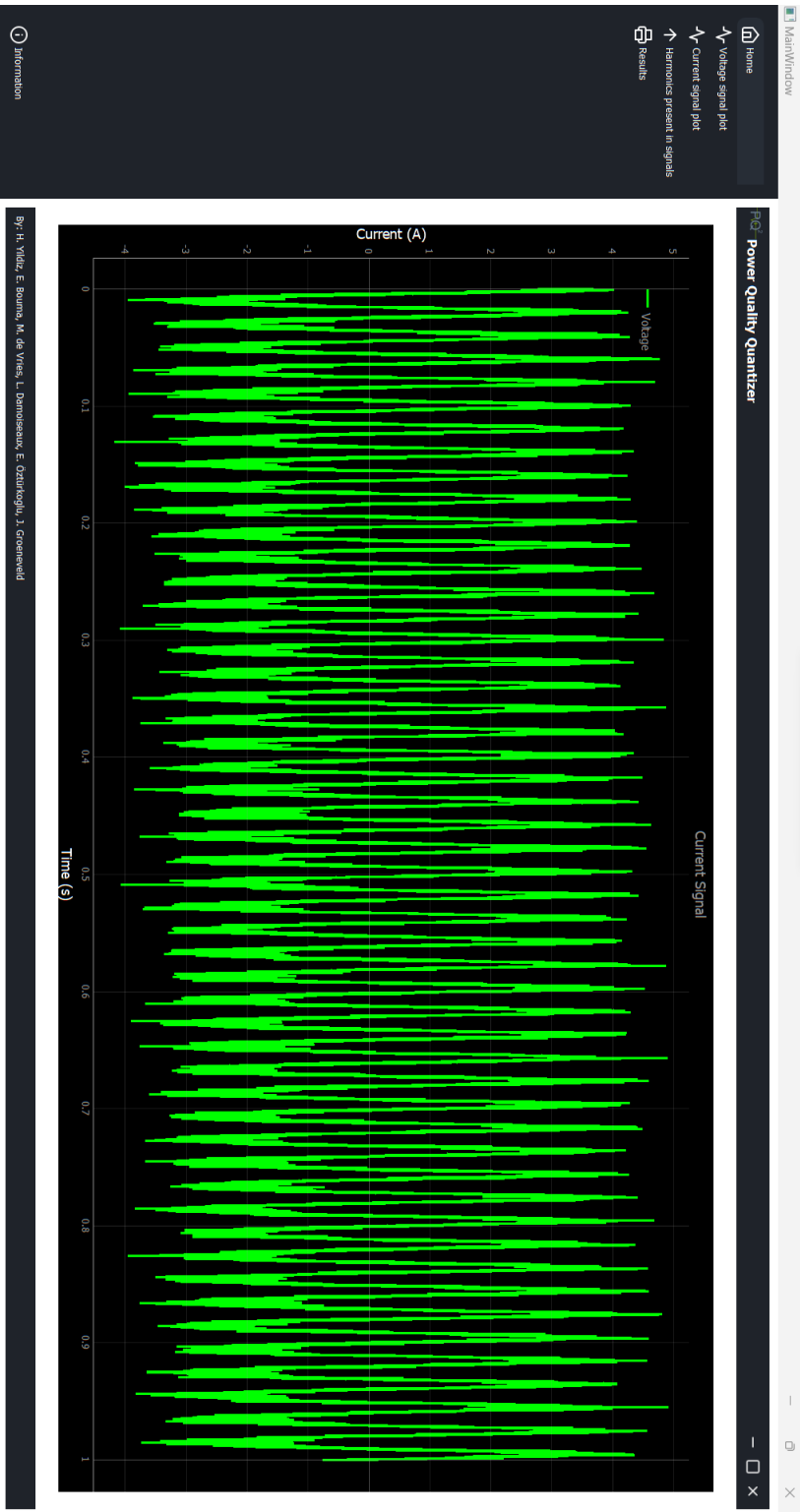


Figure C.4: The current plot with generated sampled data

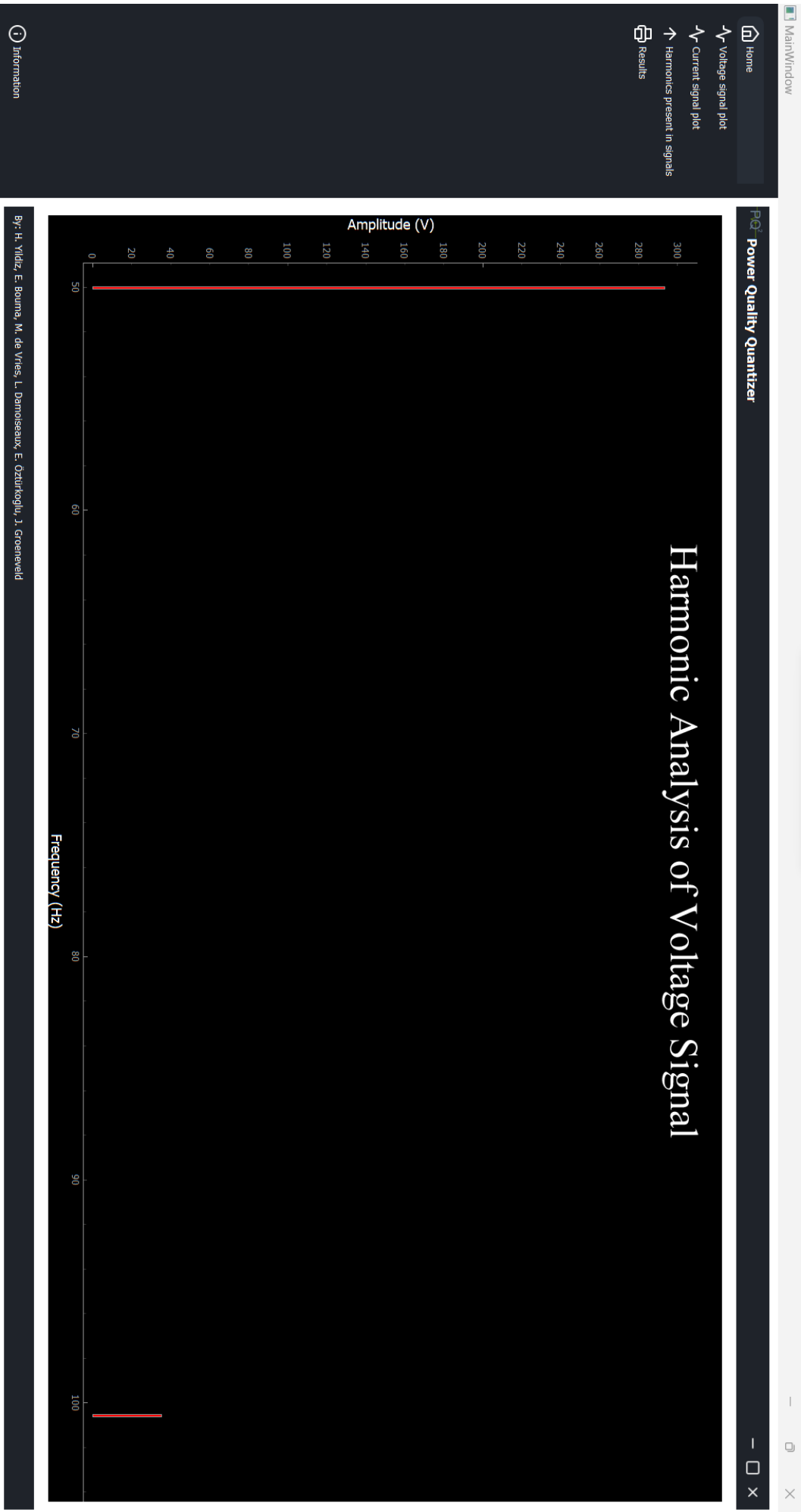


Figure C.5: The harmonics plot with generated sampled data



Figure C.6: The results page with generated sampled data

D

Login screen and GUI

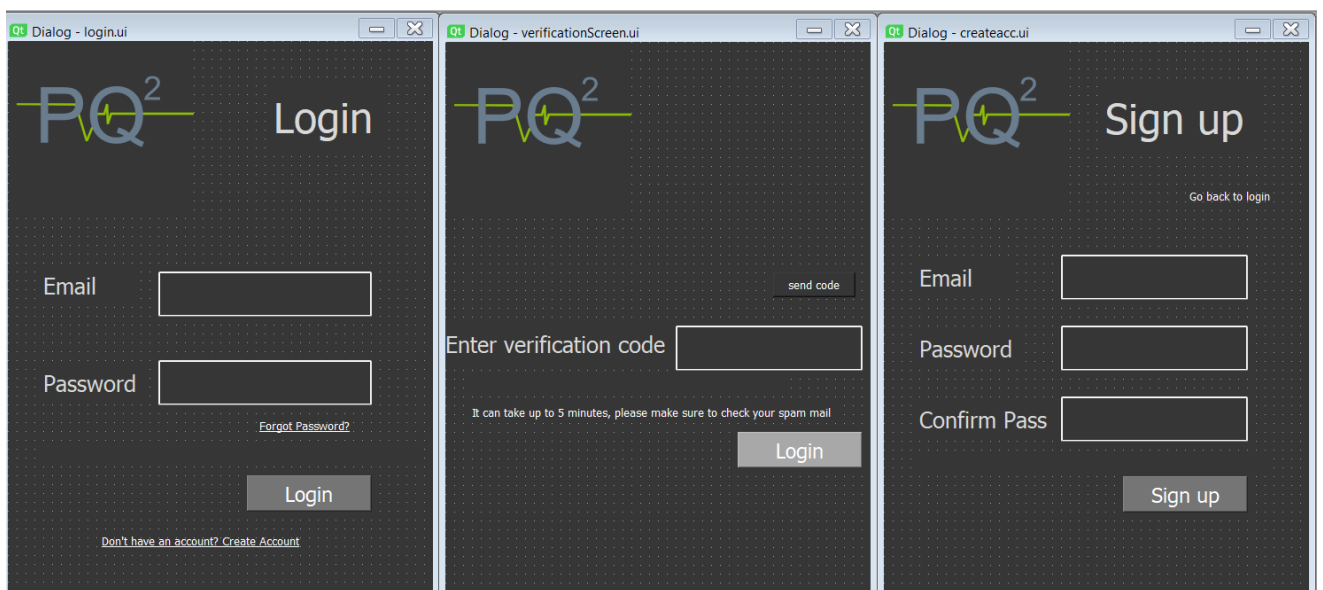


Figure D.1: Layouts of the login process. The user enters the email and password, if correct the verification screen pops up. If the user has no account, this is made on the signup page

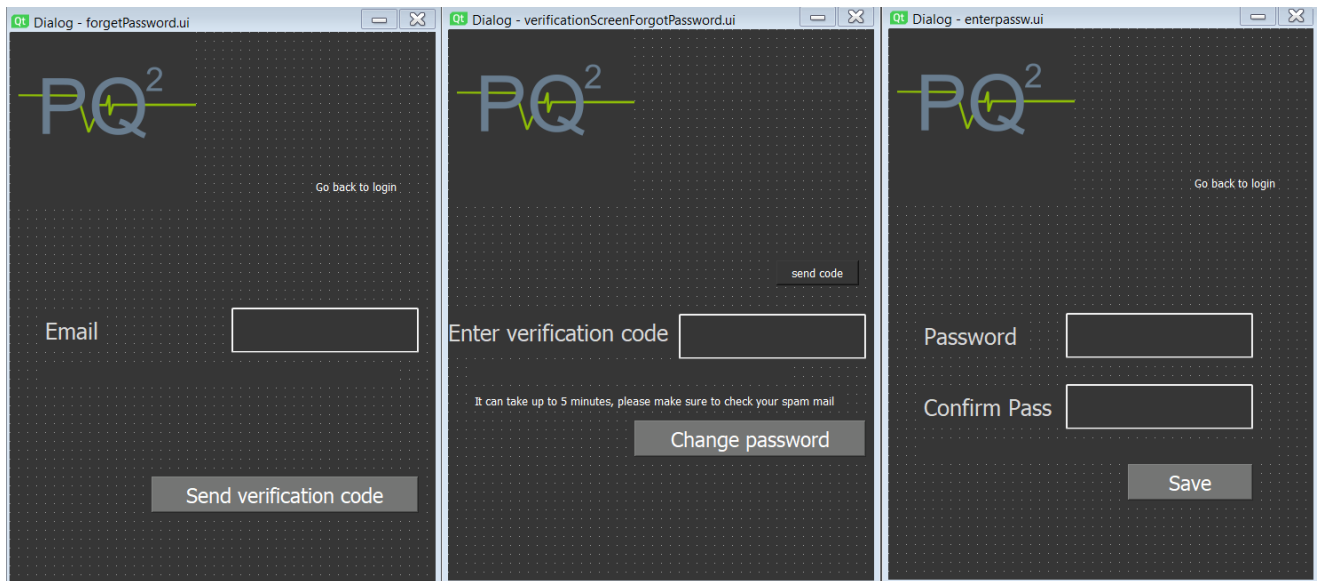


Figure D.2: Layouts of the pages when a user forgets his/her password. First, the email is entered, on the left screen. On the middle screen, a verification code is sent. On the right screen, the password is changed

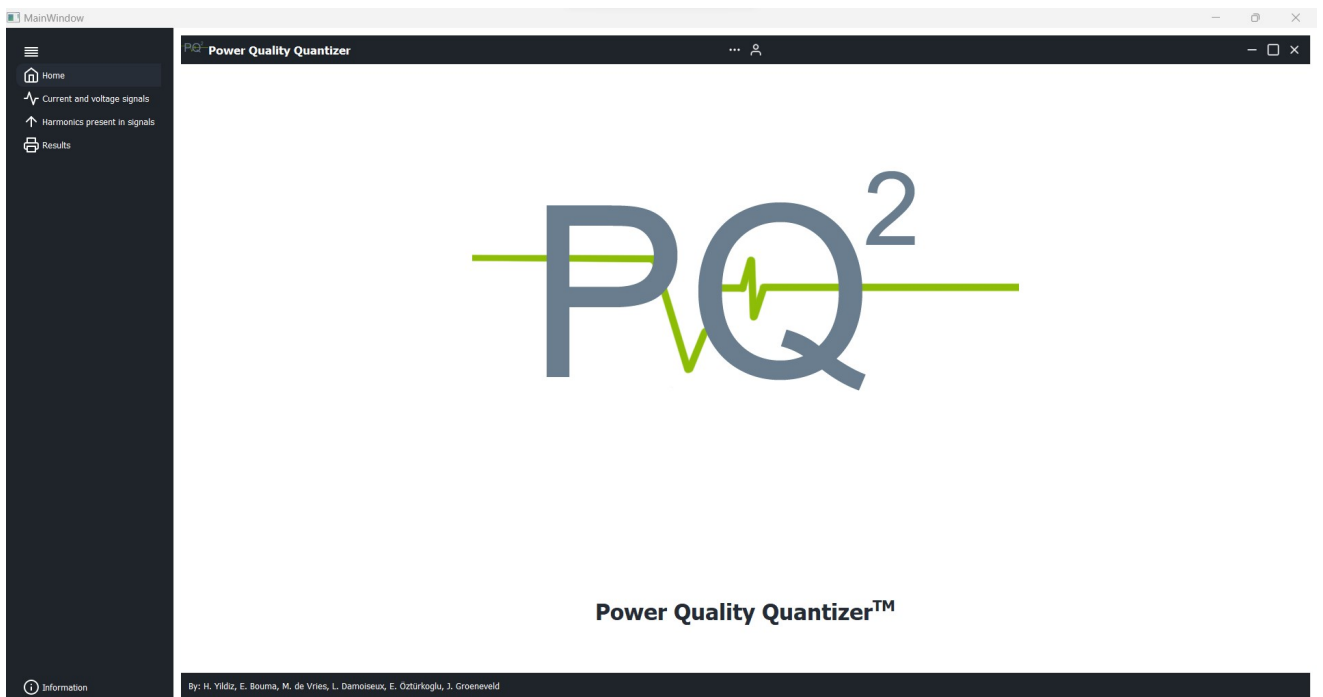


Figure D.3: Layouts of the homepage. The left menu bar is for navigation through the GUI and the right side is for displaying the pages

E

Database

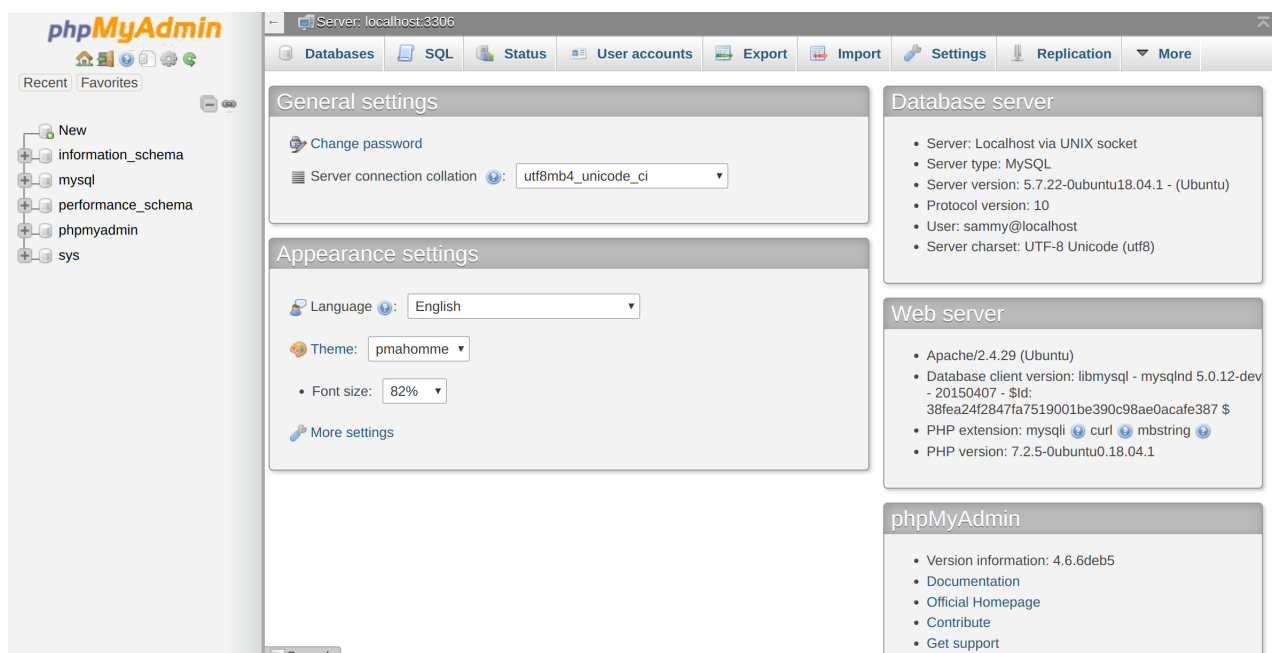


Figure E.1: phpMyAdmin interface (not the actual interface from the designed database during this project)

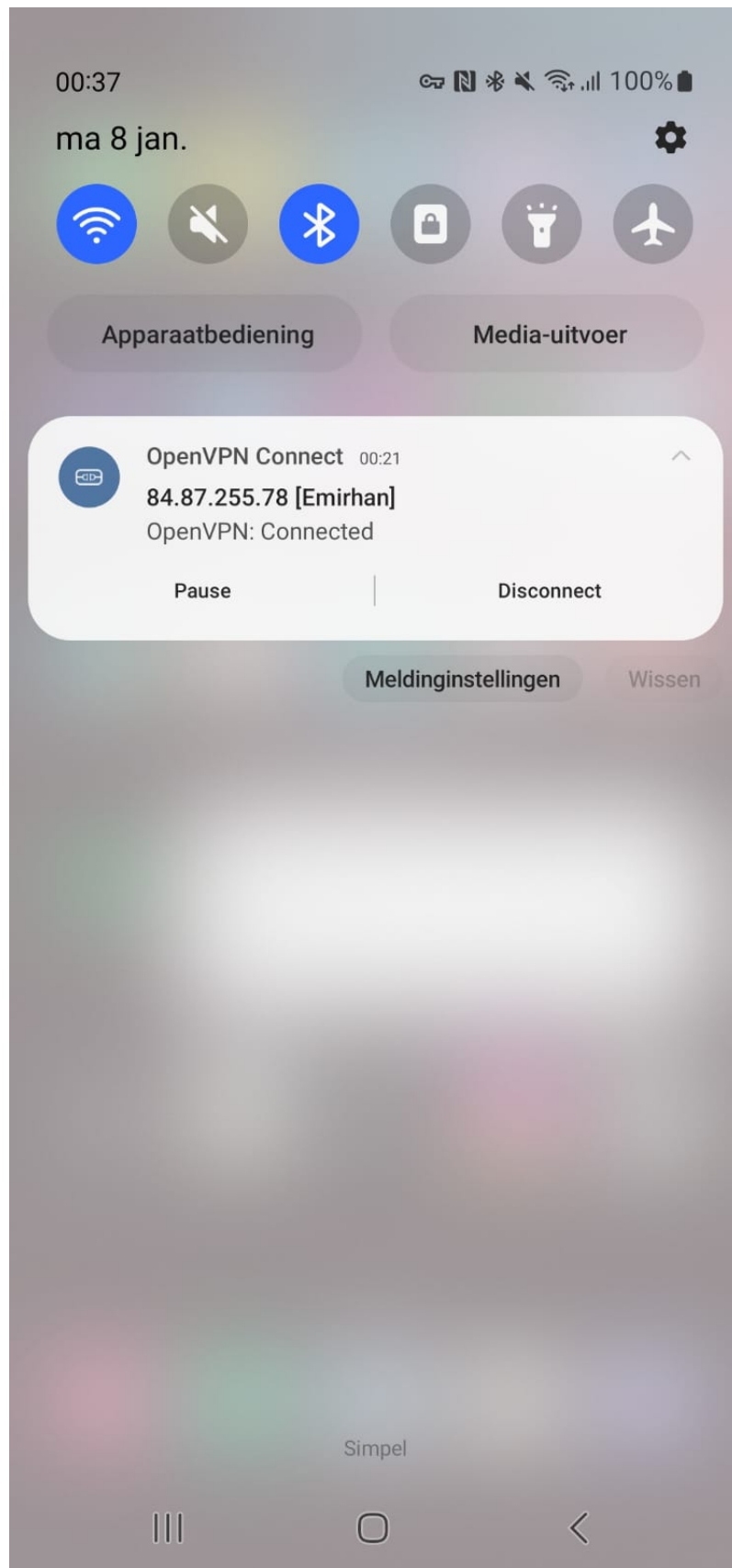


Figure E.2: VPN connection to the VPN server on the Raspberry Pi using an Android phone

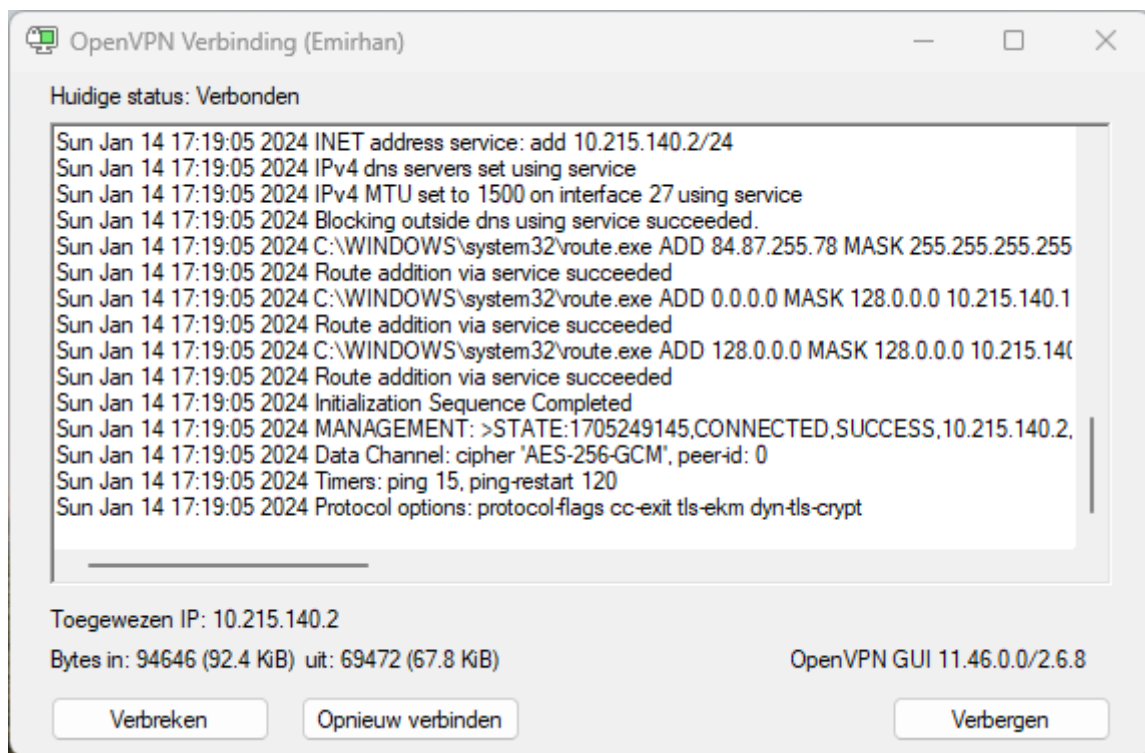


Figure E.3: VPN connection established to the Raspberry Pi via a laptop running on windows

F

Modules

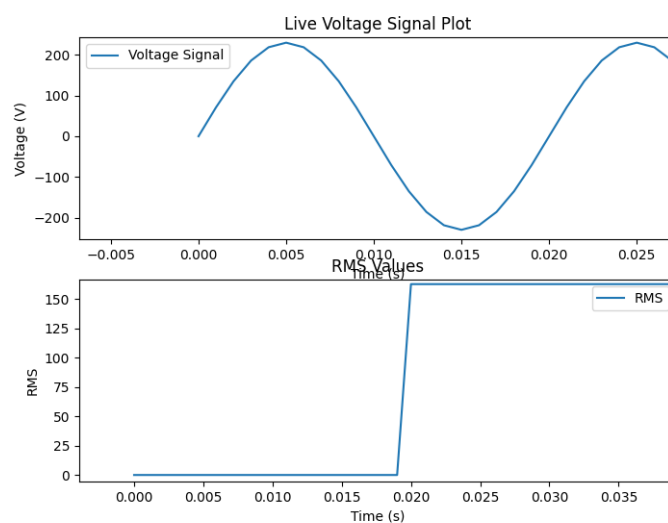


Figure F.1: The starting moment in the VRMS algorithm

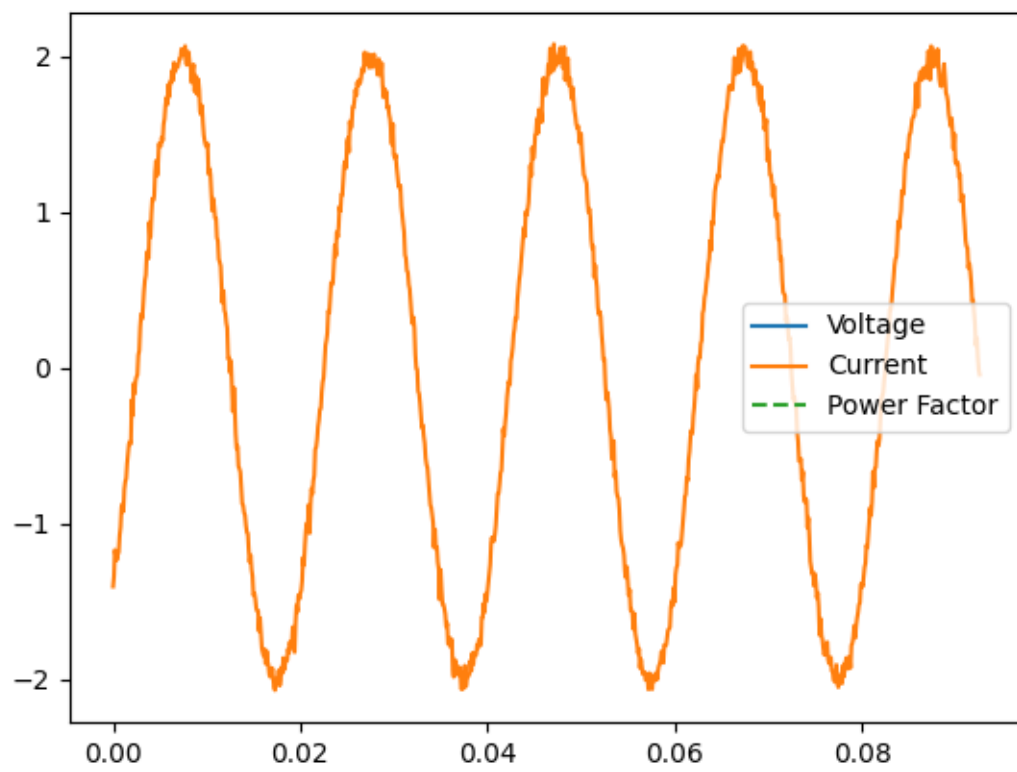


Figure F.2: Current signal that has been used for power factor testing

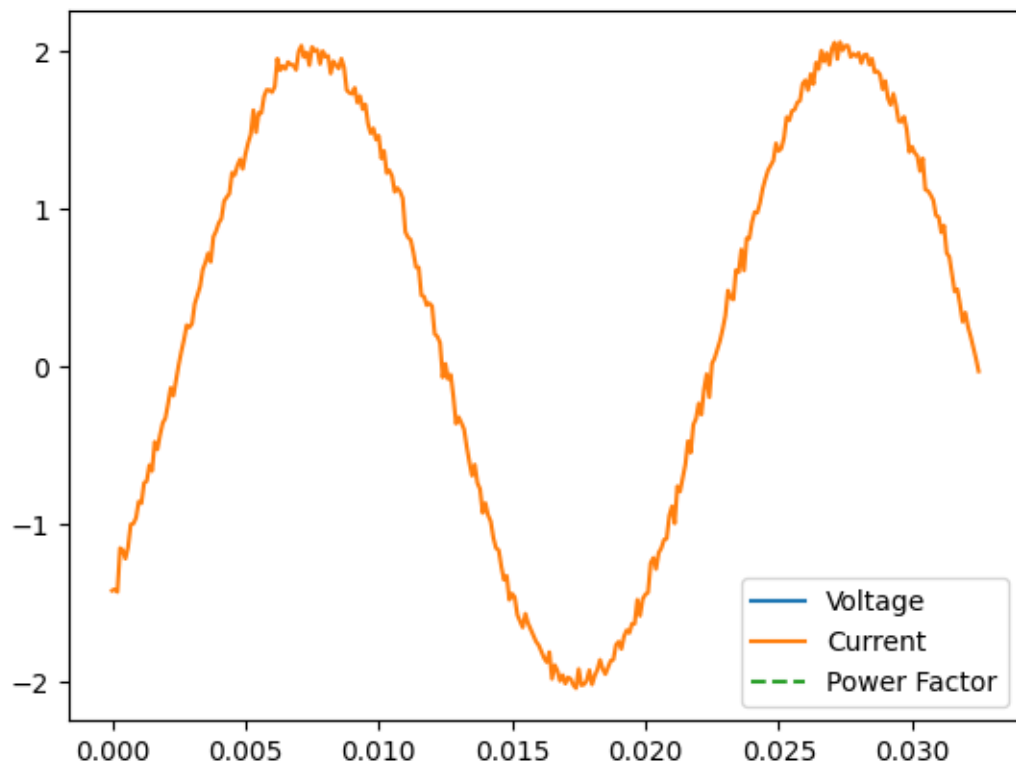


Figure F.3: Current signal that has been used for power factor testing

```
Mean Power Factor: 0.006280911870549467
Mean Power Factor: 0.0037668111992224256
Mean Power Factor: 0.010046483736529981
Mean Power Factor: 0.010676558223866837
Mean Power Factor: 0.01318879897754452
Mean Power Factor: 0.015074684467355721
Mean Power Factor: 0.009423227723427798
Mean Power Factor: 0.004396886298388972
Mean Power Factor: 0.010049582987361106
Mean Power Factor: 0.01067717819639451
```

Figure F.4: 90 Degree lagging current wave

```
Mean Power Factor: 0.4887439011172857
Mean Power Factor: 0.4920553781600527
Mean Power Factor: 0.49371219856163906
Mean Power Factor: 0.48990196197923674
Mean Power Factor: 0.48382313756038414
Mean Power Factor: 0.48816066902916516
Mean Power Factor: 0.4898454674155232
Mean Power Factor: 0.49363560011725016
Mean Power Factor: 0.49204586950576834
Mean Power Factor: 0.493703237885378
```

Figure F.5: 60 Degree lagging current wave

```
Mean Power Factor: 0.7135240432521818
Mean Power Factor: 0.7165834214241922
Mean Power Factor: 0.7174722691548113
Mean Power Factor: 0.7191484418494948
Mean Power Factor: 0.7187133340264836
Mean Power Factor: 0.7157102278756595
Mean Power Factor: 0.7152586028704512
Mean Power Factor: 0.717429982112674
Mean Power Factor: 0.7205041514666604
Mean Power Factor: 0.7107872572593742
```

Figure F.6: 45 Degree lagging current wave

```
Mean Power Factor: 0.8723809224265302
Mean Power Factor: 0.8727092044565687
Mean Power Factor: 0.8716724698327316
Mean Power Factor: 0.8727095116652344
Mean Power Factor: 0.8717932358694149
Mean Power Factor: 0.8711363475314232
Mean Power Factor: 0.8708420605091272
Mean Power Factor: 0.8674354402268964
Mean Power Factor: 0.8723293411014714
Mean Power Factor: 0.8726230478449376
```

Figure F.7: 30 Degree lagging current wave

```
Mean Power Factor: 0.9833113536055521
Mean Power Factor: 0.982555608203339
Mean Power Factor: 0.9825945973692225
Mean Power Factor: 0.9821928734588813
Mean Power Factor: 0.9828106409638789
Mean Power Factor: 0.982879780199795
Mean Power Factor: 0.982566583503845
Mean Power Factor: 0.9831734823128051
Mean Power Factor: 0.983331042158648
Mean Power Factor: 0.9830762226070432
```

Figure F.8: 11.25 Degree lagging current wave

```
Mean Power Factor: -0.4887439011172857
Mean Power Factor: -0.4920553781600527
Mean Power Factor: -0.49371219856163906
Mean Power Factor: -0.48990196197923674
Mean Power Factor: -0.48382313756038414
Mean Power Factor: -0.48816066902916516
Mean Power Factor: -0.4898454674155232
Mean Power Factor: -0.49363560011725016
Mean Power Factor: -0.49204586950576834
Mean Power Factor: -0.493703237885378
```

Figure F.9: 60 Degree leading current wave

```
Mean Power Factor: -0.7006662439842734
Mean Power Factor: -0.6983292352848833
Mean Power Factor: -0.7015402725601544
Mean Power Factor: -0.6974939328677136
Mean Power Factor: -0.696126841045315
Mean Power Factor: -0.6979865508407549
Mean Power Factor: -0.6988884925775313
Mean Power Factor: -0.7051210306669247
Mean Power Factor: -0.697932060339448
Mean Power Factor: -0.6974403462475156
```

Figure F.10: 45 Degree leading current wave

```

Mean Power Factor: -0.8607984855261388
Mean Power Factor: -0.8610264700607216
Mean Power Factor: -0.8585518021983214
Mean Power Factor: -0.8591404439135636
Mean Power Factor: -0.8567770885383946
Mean Power Factor: -0.8588804082030971
Mean Power Factor: -0.8585018072147844
Mean Power Factor: -0.8608491452251973
Mean Power Factor: -0.8575676040576501
Mean Power Factor: -0.8604872092314118

```

Figure F.11: 30 Degree leading current wave

```

Mean Power Factor: -0.9776349089574957
Mean Power Factor: -0.9799167000646297
Mean Power Factor: -0.9787122253612287
Mean Power Factor: -0.9786347991854185
Mean Power Factor: -0.9779670289468398
Mean Power Factor: -0.9767049122698083
Mean Power Factor: -0.9774111879796807
Mean Power Factor: -0.9801326179212246
Mean Power Factor: -0.9769404695809973
Mean Power Factor: -0.9777239390000616

```

Figure F.12: 11.25 Degree leading current wave

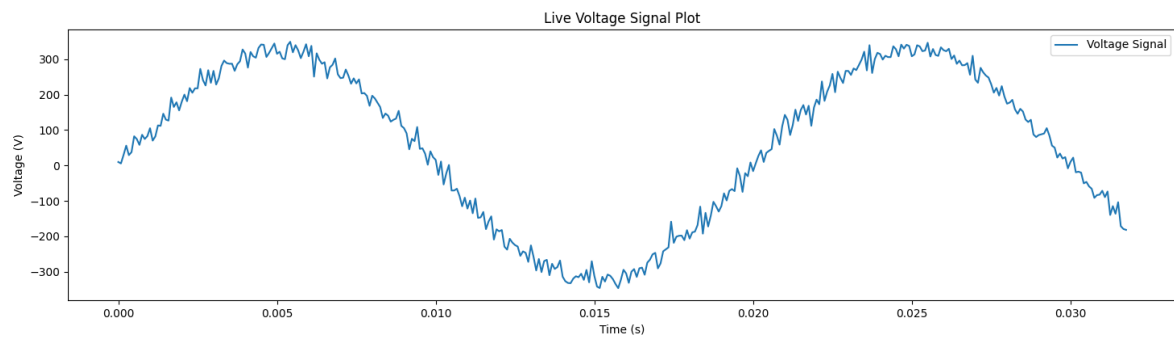


Figure F.13: Closer look to the noisy signal that has been used for testing the VRMS algorithm

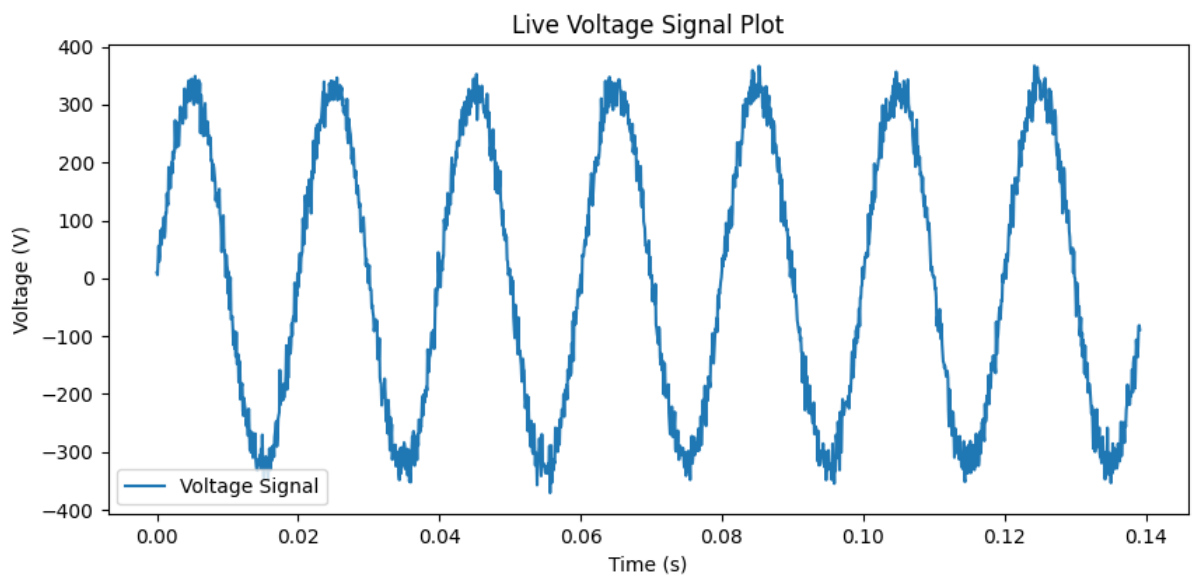


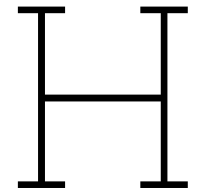
Figure F.14: An extended look to the noisy signal used for testing VRMS algorithm

G

Tables

Library	Real-Time Support	Platforms	Learning Curve	Pros	Cons	Suitability
TKinter	Possible	Multiple	Simple	- No extra installation - Simple to learn - Available on multiple platforms	- Limited widgets for advanced interfaces - May not provide modern interfaces - Difficult to debug	Simple and small-scale applications; may not meet complex requirements
PyQt	Yes	Cross-platform	Steeper	- Wide range of widgets - Real-time plotting with Matplotlib - Professional appearance - Well-documented	- Steeper learning curve - Separate installation required	Complex desktop applications with real-time data visualization and calculations
Kivy	Yes	Cross-platform	Easier	- Designed for cross-platform development - Suitable for touch-based interfaces - Relatively easier to learn	- Default style may be less suitable for complex desktop applications - Less commonly used for desktop applications	Touch-based or mobile platforms; easier learning curve
wxPython	Depends on Implementation	Cross-platform	Time-consuming	- Native look and feel - Wide range of widgets	- Time-consuming learning curve - Separate installation required	Applications requiring native platform look and feel with a variety of widgets
PySimpleGUI	Possible (with additional libraries)	Raspberry Pi	Easy	- Easy to learn - Ideal for prototyping simple interfaces	- Limited features - Better for simpler applications	Rapid development of simple interfaces
Libav	Not recommended	Multiple	Not specified	- Designed for multimedia applications - Supports multiple platforms	- Overkill for simple GUIs	Multimedia applications; may be complex for power quality measurement
PyForms	Possible (with optimization)	Cross-platform	Moderate	- Designed for quick development - Encourages a modular approach	- User-friendly, but may take time	Cross-platform development with a learning curve for complex applications
PySide6	Yes	Cross-platform	Moderate	- Cross-platform development - Integration with Qt Designer for UI design	- May take time to learn for complex applications	Cross-platform applications, especially if familiar with Qt
Wx	Not recommended	Cross-platform	Limited community	- Designed for easy GUI creation - Supports cross-platform development	- Limited community and documentation	Small to medium-sized projects prioritizing simplicity
PyGUI	Varies	Multiple	Limited community	- Quick and straightforward GUI development - Supports multiple platforms	- Limited community and resources	Small to medium projects requiring simple and rapid GUI development

Table G.1: Comparison of GUI Libraries



Code

H.1. Plotting ADC values

H.1.1. Voltage

```
1 // *****
2 // EE3L11: Bachelor Graduation Project
3 // SUBGROUP: SOFTWARE
4 // MEMBERS: Emirhan ztrkoglu, Marten de Vries, Hsna Yildiz
5 //
6 // BY Marten de Vries
7 // DATE: 18/12/2023
8 // DESCRIPTION: This code uses the ADS7142-Q1 on GPIO pins 9
9 // and 10. It does the startup for the ADC, samples and plots
10 // the outcome.
11 // *****
12
13 #include <pigpio.h>
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include <string.h>
17 #include <time.h>
18 #include <math.h>
19 #include <unistd.h>
20 #include <mariadb/mysql.h>
21
22 #define SDA 10
23 #define SCL 9
24 #define BUSY_READY_PIN 17
25 #define sample_number 6399
26
27 #define I2C_DEVICE 0x1F
28 #define SINGLE_REGISTER_WRITE 0x08
29 #define SINGLE_REGISTER_READ 0x10
30
31 int read_Register(unsigned handle, int ADC_register) {
32     i2cWriteByteData(handle, SINGLE_REGISTER_READ, ADC_register);
33     int value = i2cReadByte(handle);
34     return value;
35 }
36
37 double convertToVoltage(uint16_t inputValue, double maxOutputValue) {
38     return (((double)inputValue / ((1 << 12) - 1)) * maxOutputValue); //-448.8
39 }
40
41 void ADC_startup(unsigned handle) {
42     //GENERAL SETUP
43     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0A17); //Write WKEY for Reset
44     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0114); // Reset
45     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0115); // Offset Calibration
46     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0017); // Set WKEY back to 0x00
```

```

47  i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0024); // Single channel, Single
    ended
48  i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x001C); // Manual mode
49  i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x041C); // Manual mode with AUTO
    sequence(set 00 for standard manual mode)
50  i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0120); // Auto sequencing for
    Manual mode CH0
51  printf("Startup completed \n");
52 }
53
54 int plot(double *data, double offset) {
55     /* MAIN PLOT FUNCTION */
56     // Plot the data using gnuplot
57     FILE *gnuplotPipe = popen("gnuplot -persistent", "w");
58
59     if (gnuplotPipe == NULL) {
60         fprintf(stderr, "Error opening gnuplot pipe.\n");
61     }
62
63     // Plotting commands
64     fprintf(gnuplotPipe, "plot '-' with lines title 'Voltage'\n\n");
65
66     // Send data to gnuplot
67     for (int i = 0; i < sample_number/2; ++i) {
68         fprintf(gnuplotPipe, "%f\n", (data[i]-offset));
69     }
70
71     // End of data
72     fprintf(gnuplotPipe, "e\n");
73
74     // Close gnuplot pipe
75     fclose(gnuplotPipe);
76 }
77
78 int main() {
79
80     if (gpioInitialise() < 0) {
81         fprintf(stderr, "Error initializing Pigpio library.\n");
82         return EXIT_FAILURE;
83     }
84
85     MYSQL *conn;
86     conn = mysql_init(NULL);
87
88     if (conn == NULL) {
89         fprintf(stderr, "Error initializing MySQL.\n");
90         gpioTerminate();
91         return EXIT_FAILURE;
92     }
93
94     if (mysql_real_connect(conn, "localhost", "root", "Subgroup2", "LIVE_DATA", 0, NULL
, 0) == NULL) {
95         fprintf(stderr, "Failed to connect to database: Error: %s\n", mysql_error(conn)
);
96         mysql_close(conn);
97         gpioTerminate();
98         return EXIT_FAILURE;
99     }
100
101     if (mysql_query(conn, "CREATE TABLE IF NOT EXISTS Voltage_Data (id INT
AUTO_INCREMENT PRIMARY KEY, Value FLOAT, timestamp_col DATETIME(6) DEFAULT
CURRENT_TIMESTAMP(6))" ) != 0) {
102         fprintf(stderr, "Failed to create table: Error: %s\n", mysql_error(conn));
103         mysql_close(conn);
104         gpioTerminate();
105         return EXIT_FAILURE;
106     }
107
108     // Set GPIO modes for I2C
109     gpioSetMode(BUSY_READY_PIN, PI_INPUT);
110

```

```

111     int handle = i2cOpen(4, I2C_DEVICE, 0); //OPENING BUS 4
112
113     if (handle < 0) {
114         fprintf(stderr, "Error opening I2C handle.\n");
115         mysql_close(conn);
116         gpioTerminate();
117         return EXIT_FAILURE;
118     }
119
120     char z;
121     char ADC_VALUE[sample_number + 1];
122     char temp;
123     int j = 0;
124     double data[sample_number];
125     double sum = 0;
126     double offset = 0;
127
128
129
130     while(1){
131         ADC_startup(handle);
132         i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x011E); //SEQ_START
133         i2cReadDevice(handle, ADC_VALUE, sample_number);
134         i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x011F); //SEQ_STOP
135
136         for(int i = 1; i < sample_number; i++){
137             if (i % 2 == 0) {
138                 temp = ADC_VALUE[i];
139                 j = i / 2;
140             } else if (i % 2 != 0) {
141                 int concat = ((int)temp << 8) | ADC_VALUE[i];
142                 data[j] = convertToVoltage(concat >> 4,1100);
143                 sum += data[j];
144             }
145         }
146
147         offset = sum/j;
148         printf("offset = %f \n");
149         printf("In closeup\n");
150
151         // Close I2C handle
152         i2cClose(handle);
153
154         // Insert data into the database
155         for (int i = 0; i < sample_number/2; i++) {
156             int temp = data[i];
157
158             // Use placeholders in the query
159             char query[100];
160             sprintf(query, "INSERT INTO Voltage_Data (Value, timestamp_col) VALUES (%f,
CURRENT_TIMESTAMP(6))", temp);
161             // Execute the query
162             if (mysql_query(conn, query) != 0) {
163                 fprintf(stderr, "Failed to insert data: Error: %s\n", mysql_error(conn)
);
164
165                 mysql_close(conn);
166                 gpioTerminate();
167                 return EXIT_FAILURE;
168             }
169             // Release resources
170             plot(data, offset);
171             i2cClose(handle);
172             gpioTerminate();
173             mysql_close(conn);
174             return 0;
175             void exit();
176         }
177     }

```

H.1.2. Current

```

1 // *****
2 // EE3L11: Bachelor Graduation Project
3 // SUBGROUP: SOFTWARE
4 // MEMBERS: Emirhan ztrkoglu, Marten de Vries, Hsna Yildiz
5 //
6 // BY Marten de Vries
7 // DATE: 18/12/2023
8 // DESCRIPTION: This code uses the ADS7142-Q1 on GPIO pins 2
9 // and 3. It does the startup for the ADC, samples and plots
10 // the outcome. It runs on two-channel single ended.
11 // *****
12
13
14 #include <pigpio.h>
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include <math.h>
19 #include <unistd.h>
20 #include <mariadb/mysql.h>
21
22 #define SDA 2
23 #define SCL 3
24 #define BUSY_READY_PIN 17
25 #define sample_number 6400
26
27
28 // user emir, pass Trabzon
29
30 #define I2C_DEVICE 0x18 // I2C address
31 #define SINGLE_REGISTER_WRITE 0x08 // Single register write opcode
32 #define SINGLE_REGISTER_READ 0x10 // Single register read opcode
33
34 int read_Register(unsigned handle, int ADC_register){
35     i2cWriteByteData(handle, SINGLE_REGISTER_READ, ADC_register);
36     int value = i2cReadByte(handle);
37     return value;
38 }
39
40 double convertToCurrent_Low(uint16_t inputValue, double maxOutputValue) {
41     return (((double)inputValue / ((1 << 12) - 1)) * maxOutputValue - ((maxOutputValue/2)
42 )/1.03);
43 }
44
45 double convertToCurrent_High(uint16_t inputValue, double maxOutputValue) {
46     return (((double)inputValue / ((1 << 12) - 1)) * maxOutputValue) - ((maxOutputValue
47 /2))/1.02);
48 }
49
50 void ADC_startup(unsigned handle){
51     //GENERAL SETUP
52     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0A17); //Write WKEY for Reset
53     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0114); // Reset
54     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0115); // Offset Calibration
55     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0017); // Set WKEY back to 0x00
56     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0024); // Single channel, Single
57     ended
58     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x001C); // Manual mode
59     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x041C); // Manual mode with AUTO
60     sequence(set 00 for standard manual mode)
61     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0320); // Auto sequencing for
62     Manual mode CH0 and CH1
63     printf("Startup completed\n");
64 }
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

65  /* MAIN PLOT FUNCTION */
66  // Plot the data using gnuplot
67  FILE *gnuplotPipe = popen("gnuplot -persistent", "w");
68
69  if (gnuplotPipe == NULL) {
70      fprintf(stderr, "Error opening gnuplot pipe.\n");
71  }
72
73  // Plotting commands
74  fprintf(gnuplotPipe, "plot '-' with lines title 'Current High'\n");
75
76  // Send data to gnuplot
77  for (int i = 0; i < (sample_number/4); ++i) {
78      fprintf(gnuplotPipe, "%f\n", data[i]);
79  }
80
81  // End of data
82  fprintf(gnuplotPipe, "e\n");
83
84  // Close gnuplot pipe
85  fclose(gnuplotPipe);
86  }
87
88  /* PLOTTER FUNCTION */
89  int plot_low(double *data){
90      /* MAIN PLOT FUNCTION */
91      // Plot the data using gnuplot
92      FILE *gnuplotPipe = popen("gnuplot -persistent", "w");
93
94      if (gnuplotPipe == NULL) {
95          fprintf(stderr, "Error opening gnuplot pipe.\n");
96      }
97
98      // Plotting commands
99      fprintf(gnuplotPipe, "plot '-' with lines title 'Current Low'\n");
100
101      // Send data to gnuplot
102      for (int i = 0; i < (sample_number/4); ++i) {
103          fprintf(gnuplotPipe, "%f\n", data[i]);
104      }
105
106      // End of data
107      fprintf(gnuplotPipe, "e\n");
108
109      // Close gnuplot pipe
110      fclose(gnuplotPipe);
111  }
112
113  int main(){
114      gpioInitialise();
115      if (gpioInitialise() < 0) {
116          fprintf(stderr, "Error initializing Pigpio library.\n");
117      }
118
119      MYSQL *conn;
120      conn = mysql_init(NULL);
121
122      if (conn == NULL) {
123          fprintf(stderr, "Error initializing MySQL.\n");
124          gpioTerminate();
125          return EXIT_FAILURE;
126      }
127
128      if (mysql_real_connect(conn, "localhost", "root", "Subgroup2", "LIVE_DATA", 0, NULL, 0) == NULL) {
129          fprintf(stderr, "Failed to connect to database: Error: %s\n", mysql_error(conn));
130      };
131      mysql_close(conn);
132      gpioTerminate();
133      return EXIT_FAILURE;
134  }

```

```

134
135     if (mysql_query(conn, "CREATE TABLE IF NOT EXISTS Current_Data_High (id INT
136         AUTO_INCREMENT PRIMARY KEY, Value FLOAT, timestamp_col DATETIME(6) DEFAULT
137         CURRENT_TIMESTAMP(6))" ) != 0) {
138         fprintf(stderr, "Failed to create table: Error: %s\n", mysql_error(conn));
139         mysql_close(conn);
140         gpioTerminate();
141         return EXIT_FAILURE;
142     }
143
144     if (mysql_query(conn, "CREATE TABLE IF NOT EXISTS Current_Data_Low (id INT
145         AUTO_INCREMENT PRIMARY KEY, Value FLOAT, timestamp_col DATETIME(6) DEFAULT
146         CURRENT_TIMESTAMP(6))" ) != 0) {
147         fprintf(stderr, "Failed to create table: Error: %s\n", mysql_error(conn));
148         mysql_close(conn);
149         gpioTerminate();
150         return EXIT_FAILURE;
151     }
152
153     // Set GPIO modes for I2C
154     gpioSetMode(SDA, PI_ALTO); // SDA
155     gpioSetMode(SCL, PI_ALTO); // SCL
156     gpioSetMode(BUSY_READY_PIN, PI_INPUT); //BUSY/NRDY
157
158     int handle = i2cOpen(1, I2C_DEVICE, 0);
159     char z;
160     char ADC_VALUE[sample_number];
161     char temp; //Value data i even
162     int j = 0;
163     int m = 0;
164     double Current_High[(sample_number/2)];
165     double Current_Low[(sample_number/2)];
166
167     while(1){
168         ADC_startup(handle);
169         i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x011E); //SEQ_START
170         i2cReadDevice(handle, ADC_VALUE, sample_number);
171         i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x011F); //SEQ_STOP
172
173         for(int i = 0; i < sample_number-1; i++){
174             if(i%2 == 0){
175                 temp = ADC_VALUE[i];
176                 j = i/2;
177             }
178             else if(i%2 != 0){
179                 if (j%2==0){
180                     m = j/2;
181                     int concat = ((int)temp << 8) | ADC_VALUE[i];
182                     Current_Low[m] = convertToCurrent_Low(concat >> 4, 6.79602);
183                 }
184                 else if(j%2==1){
185                     int concat = ((int)temp << 8) | ADC_VALUE[i];
186                     Current_High[m] = convertToCurrent_High(concat >> 4, 80.652);
187                 }
188             }
189         }
190
191         // Insert Current_high into the database
192         for (int i = 0; i < m; i++) {
193             int temp = Current_High[i];
194
195             // Use placeholders in the query
196             char query[100];
197             sprintf(query, "INSERT INTO Current_Data_High (Value, timestamp_col) VALUES (%f,
198                 CURRENT_TIMESTAMP(6))", temp);
199             // Execute the query
200             if (mysql_query(conn, query) != 0) {
201                 fprintf(stderr, "Failed to insert data: Error: %s\n", mysql_error(conn));
202                 mysql_close(conn);
203                 gpioTerminate();
204                 return EXIT_FAILURE;
205             }
206         }
207     }

```

```

200     }
201 }
202
203 // Insert Current_Low into the database
204 for (int i = 0; i < m; i++) {
205     int temp = Current_Low[i];
206
207     // Use placeholders in the query
208     char query[100];
209     sprintf(query, "INSERT INTO Current_Data_Low (Value, timestamp_col) VALUES (%f,
CURRENT_TIMESTAMP(6))", temp);
210     // Execute the query
211     if (mysql_query(conn, query) != 0) {
212         fprintf(stderr, "Failed to insert data: Error: %s\n", mysql_error(conn));
213         mysql_close(conn);
214         gpioTerminate();
215         return EXIT_FAILURE;
216     }
217 }
218
219 // Close I2C handle
220 printf("in closeup \n");
221 plot_high(Current_High);
222 plot_low(Current_Low);
223 i2cClose(handle);
224
225 // Release resources
226 gpioTerminate();
227 void exit();
228 return(0);
229 }
230 }

```

H.2. Sampling and database integration

H.2.1. Voltage sampling with database integration

```

1 // *****
2 // EE3L11: Bachelor Graduation Project
3 // SUBGROUP: SOFTWARE
4 // MEMBERS: Emirhan ztrkoglu, Marten de Vries, Hsna Yildiz
5 //
6 // BY Marten de Vries
7 // DATE: 8/1/2024
8 // DESCRIPTION: This code uses the ADS7142-Q1 on GPIO pins 9
9 // and 10. It does the startup for the ADC, samples data and
10 // writes it to the database. It does this by making use of
11 // multiprocessing.
12 // *****
13
14
15 #include <pigpio.h>
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include <string.h>
19 #include <time.h>
20 #include <math.h>
21 #include <unistd.h>
22 #include <mariadb/mysql.h>
23 #include <sys/ipc.h>
24 #include <sys/shm.h>
25 #include <sys/types.h>
26 #include <sys/wait.h>
27
28
29 #define SDA 10
30 #define SCL 9
31 #define BUSY_READY_PIN 17
32 #define sample_number 3199
33
34 #define I2C_DEVICE 0x1F

```

```

35 #define SINGLE_REGISTER_WRITE 0x08
36 #define SINGLE_REGISTER_READ 0x10
37
38 int read_Register(unsigned handle, int ADC_register) {
39     i2cWriteByteData(handle, SINGLE_REGISTER_READ, ADC_register);
40     int value = i2cReadByte(handle);
41     return value;
42 }
43
44 double convertToVoltage(uint16_t inputValue, double maxOutputValue) {
45     return (((double)inputValue / ((1 << 12) - 1)) * maxOutputValue));
46 }
47
48 void ADC_startup(unsigned handle) {
49     //GENERAL SETUP
50     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0A17); //Write WKEY for Reset
51     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0114); // Reset
52     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0115); // Offset Calibration
53     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0017); // Set WKEY back to 0x00
54     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0024); // Single channel, Single
55     ended
56     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x001C); // Manual mode
57     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x041C); // Manual mode with AUTO
58     sequence(set 00 for standard manual mode)
59     i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x0120); // Auto sequencing for
60     Manual mode CH0
61 }
62
63 int main() {
64
65     MYSQL *conn;
66     conn = mysql_init(NULL);
67
68     if (conn == NULL) {
69         fprintf(stderr, "Error initializing MySQL.\n");
70         gpioTerminate();
71         return EXIT_FAILURE;
72     }
73
74
75     char z;
76     char ADC_VALUE[sample_number + 1];
77     char temp;
78     int j = 0;
79     double temp_storage[(sample_number/2)-1];
80
81     double sum = 0;
82     double offset = 0;
83     int fd[2];
84     int counter = 1;
85
86     if (pipe(fd) == -1){
87         perror("Problem starting pipe");
88         return EXIT_FAILURE;
89     }
90
91
92
93     while(1){
94         int pid = fork();
95         if (pid < 0) {
96             fprintf(stderr, "Failed to fork.\n");
97             mysql_close(conn);
98             gpioTerminate();
99             return EXIT_FAILURE;
100         }
101         if(pid == 0){
102             if (mysql_real_connect(conn, "localhost", "root", "Subgroup2", "LIVE_DATA", 0,

```

```

103     NULL, 0) == NULL) {
104         fprintf(stderr, "Failed to connect to database: Error: %s\n", mysql_error(conn)
105     );
106         mysql_close(conn);
107         gpioTerminate();
108         return EXIT_FAILURE;
109     }
110
111     if (mysql_query(conn, "CREATE TABLE IF NOT EXISTS Voltage_Data (id INT
112     AUTO_INCREMENT PRIMARY KEY, Value FLOAT, timestamp_col DATETIME(6) DEFAULT
113     CURRENT_TIMESTAMP(6))" ) != 0) {
114         fprintf(stderr, "Failed to create table: Error: %s\n", mysql_error(conn));
115         mysql_close(conn);
116         gpioTerminate();
117         return EXIT_FAILURE;
118     }
119
120     //CHILD PROCESS (PUTTING IN DATABASE
121     double child_storage[(sample_number/2)-1];
122     if(read(fd[0], child_storage, sizeof(temp_storage)) == -1){
123         perror("Problem reading the pipe");
124         return EXIT_FAILURE;
125     }
126
127     // Begin the transaction
128     if (mysql_query(conn, "BEGIN") != 0) {
129         fprintf(stderr, "Failed to start transaction: %s\n", mysql_error(conn));
130         mysql_close(conn);
131         gpioTerminate();
132         return EXIT_FAILURE;
133     }
134
135     for (int i = 0; i < (sample_number / 2) - 1; i++) {
136         double temp = child_storage[i];
137         // Use placeholders in the query
138         char query[100];
139         sprintf(query, "INSERT INTO Voltage_Data (Value, timestamp_col) VALUES (%f,
140         CURRENT_TIMESTAMP(6))", temp);
141         // Execute the query
142         if (mysql_query(conn, query) != 0) {
143             fprintf(stderr, "Failed to insert data: Error: %s\n", mysql_error(conn));
144             // Rollback the transaction on error
145             mysql_query(conn, "ROLLBACK");
146             mysql_close(conn);
147             gpioTerminate();
148             return EXIT_FAILURE;
149         }
150     }
151
152     // Commit the transaction
153     if (mysql_query(conn, "COMMIT") != 0) {
154         fprintf(stderr, "Failed to commit transaction: %s\n", mysql_error(conn));
155         mysql_close(conn);
156         gpioTerminate();
157         return EXIT_FAILURE;
158     }
159
160     mysql_close(conn);
161     exit(EXIT_SUCCESS);
162 } else {
163     printf("Amount of times we looped: %d \n", counter);
164     //PARENT PROCESS PUTTING IN PIPELINE
165
166     if (gpioInitialise() < 0) {
167         fprintf(stderr, "Error initializing Pigpio library.\n");
168         return EXIT_FAILURE;
169     }
170
171     // Set GPIO modes for I2C
172     gpioSetMode(BUSY_READY_PIN, PI_INPUT);
173
174     int handle = i2cOpen(4, I2C_DEVICE, 0); //OPENING BUS 4
175
176     if (handle < 0) {

```

```

169     fprintf(stderr, "Error opening I2C handle.\n");
170     mysql_close(conn);
171     gpioTerminate();
172     return EXIT_FAILURE;
173 }
174 ADC_startup(handle);
175 i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x011E); //SEQ_START
176 i2cReadDevice(handle, ADC_VALUE, sample_number);
177 i2cWriteWordData(handle, SINGLE_REGISTER_WRITE, 0x011F); //SEQ_STOP
178 i2cClose(handle);
179 //gpioTerminate();
180 for(int i = 1; i < sample_number; i++){
181     if (i % 2 == 0) {
182         temp = ADC_VALUE[i];
183         j = i / 2;
184     } else if (i % 2 != 0) {
185         int concat = ((int)temp << 8) | ADC_VALUE[i];
186         temp_storage[j] = convertToVoltage(concat >> 4,1100);
187     }
188 }
189 if(write(fd[1], temp_storage, sizeof(temp_storage)) == -1){
190     perror("Problem writing to pipe");
191     return EXIT_FAILURE;
192 }
193 //close(fd[1]);
194 wait(NULL);
195 counter += 1;
196 }
197 }
198 }

```

H.2.2. Current sampling with database integration

```

1 // *****
2 // EE3L11: Bachelor Graduation Project
3 // SUBGROUP: SOFTWARE
4 // MEMBERS: Emirhan ztrkogl, Marten de Vries, Hsna Yildiz
5 //
6 // BY Marten de Vries
7 // DATE: 8/1/2024
8 // DESCRIPTION: This code uses the ADS7142-Q1 on GPIO pins 2
9 // and 3. It does the startup for the ADC, samples data and
10 // writes it to the database. It does this by making use of
11 // multiprocessing. This code only works if the pigpiod daemon
12 // is running on your device.
13 // *****
14
15
16 #include <pigpiod_if2.h>
17 #include <pigpio.h>
18 #include <stdio.h>
19 #include <stdlib.h>
20 #include <string.h>
21 #include <time.h>
22 #include <math.h>
23 #include <unistd.h>
24 #include <mariadb/mysql.h>
25 #include <sys/ipc.h>
26 #include <sys/shm.h>
27 #include <sys/types.h>
28 #include <sys/wait.h>
29
30
31 #define SDA 2
32 #define SCL 3
33 #define BUSY_READY_PIN 17
34 #define sample_number 6400
35
36
37 #define I2C_DEVICE 0x18 // I2C address
38 #define SINGLE_REGISTER_WRITE 0x08 // Single register write opcode

```

```

39 #define SINGLE_REGISTER_READ 0x10    // Single register read opcode
40
41 int read_Register(unsigned handle, int ADC_register) {
42     i2cWriteByteData(handle, SINGLE_REGISTER_READ, ADC_register);
43     int value = i2cReadByte(handle);
44     return value;
45 }
46
47 double convertToCurrent_Low(uint16_t inputValue, double maxOutputValue) {
48     return (((double)inputValue / ((1 << 12) - 1)) * maxOutputValue);
49 }
50
51 double convertToCurrent_High(uint16_t inputValue, double maxOutputValue) {
52     return (((double)inputValue / ((1 << 12) - 1)) * maxOutputValue);
53 }
54
55 void ADC_startup(unsigned pi, unsigned handle) {
56     //GENERAL SETUP
57     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x0A17); //Write WKEY for
58     //Reset
59     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x0114); // Reset
60     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x0115); // Offset Calibration
61     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x0017); // Set WKEY back to 0
62     //x00
63     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x0024); // Single channel,
64     //Single ended
65     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x001C); // Manual mode
66     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x041C); // Manual mode with
67     //AUTO sequence(set 00 for standard manual mode)
68     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x0120); // Auto sequencing
69     //for Manual mode CH0
70 }
71
72 int main() {
73     int pi = pigpio_start(NULL, NULL);
74
75     MYSQL *conn;
76     conn = mysql_init(NULL);
77
78     if (conn == NULL) {
79         fprintf(stderr, "Error initializing MySQL.\n");
80         gpioTerminate();
81         return EXIT_FAILURE;
82     }
83
84     char z;
85     char ADC_VALUE[sample_number + 1];
86     char temp;
87     int j = 0;
88     int m = 0;
89     double temp_storage[(sample_number/4)];
90     double Current_High[(sample_number/4)];
91     double Current_Low[(sample_number/4)];
92
93     double sum = 0;
94     double offset = 0;
95     int fd[2];
96     int fd_2[2];
97     int counter = 1;
98
99     if (pipe(fd) == -1){
100         perror("Problem starting pipe");
101         return EXIT_FAILURE;
102     }
103
104     if (pipe(fd_2) == -1){
105         perror("Problem starting pipe");
106         return EXIT_FAILURE;
107     }

```

```

105     }
106
107     while(1){
108         int pid = fork();
109         if (pid < 0) {
110             fprintf(stderr, "Failed to fork.\n");
111             mysql_close(conn);
112             pigpio_stop(pi);
113             return EXIT_FAILURE;
114         }
115         if(pid == 0){
116             if (mysql_real_connect(conn, "localhost", "root", "Subgroup2", "LIVE_DATA", 0,
117 NULL, 0) == NULL) {
118                 fprintf(stderr, "Failed to connect to database: Error: %s\n", mysql_error(conn)
119 );
120                 mysql_close(conn);
121                 pigpio_stop(pi);
122                 return EXIT_FAILURE;
123             }
124             //WRITING CURRENT LOW
125             if (mysql_query(conn, "CREATE TABLE IF NOT EXISTS Current_Data_Low (id INT
126 AUTO_INCREMENT PRIMARY KEY, Value FLOAT, timestamp_col DATETIME(6) DEFAULT
127 CURRENT_TIMESTAMP(6))" ) != 0) {
128                 fprintf(stderr, "Failed to create table: Error: %s\n", mysql_error(conn));
129                 mysql_close(conn);
130                 pigpio_stop(pi);
131                 return EXIT_FAILURE;
132             }
133             //CHILD PROCESS (PUTTING IN DATABASE
134             //close(fd[1]); //close writing
135             double child_storage_Low[(sample_number/4)];
136             if(read(fd[0], child_storage_Low, sizeof(temp_storage)) == -1){
137                 perror("Problem reading the pipe");
138                 return EXIT_FAILURE;
139             }
140             // Begin the transaction
141             if (mysql_query(conn, "BEGIN") != 0) {
142                 fprintf(stderr, "Failed to start transaction: %s\n", mysql_error(conn));
143                 mysql_close(conn);
144                 pigpio_stop(pi);
145                 return EXIT_FAILURE;
146             }
147
148             for (int i = 0; i < (sample_number / 4); i++) {
149                 double temp = child_storage_Low[i];
150                 // Use placeholders in the query
151                 char query[100];
152                 sprintf(query, "INSERT INTO Current_Data_Low (Value, timestamp_col) VALUES (%f,
153 CURRENT_TIMESTAMP(6))", temp);
154                 // Execute the query
155                 if (mysql_query(conn, query) != 0) {
156                     fprintf(stderr, "Failed to insert data: Error: %s\n", mysql_error(conn));
157                     // Rollback the transaction on error
158                     mysql_query(conn, "ROLLBACK");
159                     mysql_close(conn);
160                     pigpio_stop(pi);
161                     return EXIT_FAILURE;
162                 }
163             }
164
165             // Commit the transaction
166             if (mysql_query(conn, "COMMIT") != 0) {
167                 fprintf(stderr, "Failed to commit transaction: %s\n", mysql_error(conn));
168                 mysql_close(conn);
169                 pigpio_stop(pi);
170                 return EXIT_FAILURE;
171             }
172             //WRITING CURRENT HIGH
173             if (mysql_query(conn, "CREATE TABLE IF NOT EXISTS Current_Data_High (id INT
174 AUTO_INCREMENT PRIMARY KEY, Value FLOAT, timestamp_col DATETIME(6) DEFAULT
175 CURRENT_TIMESTAMP(6))" ) != 0) {

```

```

169     fprintf(stderr, "Failed to create table: Error: %s\n", mysql_error(conn));
170     mysql_close(conn);
171     pigpio_stop(pi);
172     return EXIT_FAILURE;
173 }
174 //CHILD PROCESS (PUTTING IN DATABASE
175 //close(fd[1]); //close writing
176 double child_storage_High[(sample_number/4)];
177 if(read(fd_2[0], child_storage_High, sizeof(temp_storage)) == -1){
178     perror("Problem reading the pipe");
179     return EXIT_FAILURE;
180 }
181 // Begin the transaction
182 if (mysql_query(conn, "BEGIN") != 0) {
183     fprintf(stderr, "Failed to start transaction: %s\n", mysql_error(conn));
184     mysql_close(conn);
185     pigpio_stop(pi);
186     return EXIT_FAILURE;
187 }
188
189 for (int i = 0; i < (sample_number / 4); i++) {
190     double temp = child_storage_High[i];
191     // Use placeholders in the query
192     char query[100];
193     sprintf(query, "INSERT INTO Current_Data_High (Value, timestamp_col) VALUES (%f
, CURRENT_TIMESTAMP(6))", temp);
194     // Execute the query
195     if (mysql_query(conn, query) != 0) {
196         fprintf(stderr, "Failed to insert data: Error: %s\n", mysql_error(conn));
197         // Rollback the transaction on error
198         mysql_query(conn, "ROLLBACK");
199         mysql_close(conn);
200         pigpio_stop(pi);
201         return EXIT_FAILURE;
202     }
203 }
204
205 // Commit the transaction
206 if (mysql_query(conn, "COMMIT") != 0) {
207     fprintf(stderr, "Failed to commit transaction: %s\n", mysql_error(conn));
208     mysql_close(conn);
209     pigpio_stop(pi);
210     return EXIT_FAILURE;
211 }
212
213 mysql_close(conn);
214 exit(EXIT_SUCCESS);
215 } else {
216     printf("Amount of times we looped: %d \n", counter);
217     //PARENT PROCESS PUTTING IN PIPELINE
218     //close(fd[0]); //close reading
219
220     if (pigpio_start(NULL, NULL) < 0) {
221         fprintf(stderr, "Error initializing Pigpio library.\n");
222         return EXIT_FAILURE;
223     }
224
225     // Set GPIO modes for I2C
226     //gpioSetMode(BUSY_READY_PIN, PI_INPUT);
227
228     int handle = i2c_open(pi, 1, I2C_DEVICE, 0); //OPENING BUS 1
229
230     if (handle < 0) {
231         fprintf(stderr, "Error opening I2C handle.\n");
232         mysql_close(conn);
233         pigpio_stop(pi);
234         return EXIT_FAILURE;
235     }
236
237     ADC_startup(pi, handle);
238     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x011E); //SEQ_START

```

```

239     i2c_read_device(pi, handle, ADC_VALUE, sample_number);
240     i2c_write_word_data(pi, handle, SINGLE_REGISTER_WRITE, 0x011F); //SEQ_STOP
241     //i2c_close(pi, handle);
242     //pigpio_stop(pi);
243     for(int i = 0; i < sample_number-1; i++){
244         if(i%2 == 0){
245             temp = ADC_VALUE[i];
246             j = i/2;
247         }
248         else if(i%2 != 0){
249             if (j%2==0){
250                 m = j/2;
251                 int concat = ((int)temp << 8) | ADC_VALUE[i];
252                 Current_Low[m] = convertToCurrent_Low(concat >> 4, 6.79602);
253             }
254             else if(j%2==1){
255                 int concat = ((int)temp << 8) | ADC_VALUE[i];
256                 Current_High[m] = convertToCurrent_High(concat >> 4, 80.652);
257             }
258         }
259     }
260     if(write(fd[1], Current_Low, sizeof(Current_Low)) == -1){
261         perror("Problem writing to pipe1");
262         return EXIT_FAILURE;
263     }
264     if(write(fd_2[1], Current_High, sizeof(Current_High)) == -1){
265         perror("Problem writing to pipe2");
266         return EXIT_FAILURE;
267     }
268     //close(fd[1]);
269     wait(NULL);
270     counter += 1;
271 }
272
273 }
274 }

```

H.3. Power Quality Parameters

```

1  # #####
2  # EE3L11: Bachelor Graduation Project
3  # SUBGROUP: SOFTWARE
4  # MEMBERS: Emirhan Üztürkolu, Marten de Vries, Hüsna Yildiz
5  #
6  # BY Emirhan Üztürkolu, Marten de Vries, Hüsna Yildiz
7  # DATE: 8/1/2024
8  # Description: This is a file with all the power quality parameter calculators.
9  # The data is fetched from the database and fed to the different function.
10 # They return their values.
11 # #####
12
13
14 # Importing packages
15 import numpy as np
16 import matplotlib.pyplot as plt
17 from scipy.fft import fft, fftfreq
18 from scipy.signal import find_peaks, butter, lfilter
19 from scipy.interpolate import interp1d
20 import datetime
21 from datetime import datetime, timedelta
22 import pymysql
23
24 timestamps_Voltage = []
25 values_Voltage = []
26
27 timestamps_Current_Low = []
28 timestamps_Current_High = []
29
30 values_Current_Low = []
31 values_Current_High = []

```

```

32
33 host = '145.94.62.207'
34 user = "remote_software"
35 password = "remote_database"
36 database = "LIVE_DATA"
37 connection = pymysql.connect(
38     host=host,
39     user=user,
40     password=password,
41     db=database
42 )
43 cursor = connection.cursor()
44 timerange = timedelta(seconds=5)
45 start_time = datetime.now() - timedelta(seconds=5)
46 end_time = start_time + timerange
47
48 query = f"SELECT Value, timestamp_col FROM Voltage_Data WHERE timestamp_col >= '{
49     start_time}' AND timestamp_col < '{end_time}'"
50 cursor.execute(query)
51 rows = cursor.fetchall()
52
53 for row in rows:
54     timestamps_Voltage.append(row[1])
55     values_Voltage.append(row[0])
56
57 query = f"SELECT Value, timestamp_col FROM Current_Data_Low WHERE timestamp_col >= '{
58     start_time}' AND timestamp_col < '{end_time}'"
59 cursor.execute(query)
60 rows = cursor.fetchall()
61
62 for row in rows:
63     timestamps_Current_Low.append(row[1])
64     values_Current_Low.append(row[0])
65
66 query = f"SELECT Value, timestamp_col FROM Current_Data_High WHERE timestamp_col >= '{
67     start_time}' AND timestamp_col < '{end_time}'"
68 cursor.execute(query)
69 rows = cursor.fetchall()
70
71 for row in rows:
72     timestamps_Current_High.append(row[1])
73     values_Current_High.append(row[0])
74
75 print("Length of voltage values: ",len(values_Voltage), " Length of Current Low: ", len
76     (values_Current_Low), " en length of Current High :", len(values_Current_High))
77 voltage_values_numpy = np.array(values_Voltage)
78 current_values_low_numpy = np.array(values_Current_Low)
79 current_values_high_numpy = np.array(values_Current_High)
80
81 # for no error i added this piece
82 if len(values_Current_High) > 0:
83     max_current_High = np.amax(values_Current_High)
84 else:
85     max_current_High = 0 # or any default value you want
86
87 mean_voltage = np.mean(values_Voltage)
88 # Biased voltage values
89 voltage_signal_with_bias = voltage_values_numpy - mean_voltage
90 if max_current_High > 4 * np.sqrt(2):
91     current_signal_with_bias = current_values_high_numpy - (np.amax(
92         current_values_high_numpy) / 2) * 1.02
93
94 else :
95     # current_signal_with_bias = current_values_low_numpy - (np.amax(
96         current_values_low_numpy)/2)*1.03
97     if len(current_values_low_numpy) > 0:
98         max_current_High = np.amax(current_values_low_numpy)
99         current_signal_with_bias = current_values_low_numpy - (max_current_High /
100             2) * 1.03
101     else:
102         max_current_High = 0 # or any default value you want
103         current_signal_with_bias = np.array([]) # or any default array you want

```

```

96
97 #
98     #####
99
100 # Frequency estimation algorithm
101
102 def estimate_frequency_zero_crossing(signal):
103     f = 50
104     sample_rate = len(signal)
105     window_size = sample_rate / (f * 4)
106     zero_crossings = np.where(np.diff(np.sign(signal)))[0]
107     valid_crossings = []
108
109     prev_crossing = None
110
111     for crossing in zero_crossings:
112         if prev_crossing is None or crossing - prev_crossing > window_size:
113             # Keep the first sample of the zero crossing
114             valid_crossings.append(crossing)
115             prev_crossing = crossing
116
117     # Calculate time for a full period (time between consecutive zero crossings)
118     time_between_crossings = (2 * np.diff(valid_crossings)) / sample_rate
119
120     # Average time for a full period
121     if len(time_between_crossings) > 0:
122         average_time_for_full_period = np.mean(time_between_crossings)
123         estimated_frequency = 1 / average_time_for_full_period
124     else:
125         estimated_frequency = 0
126
127     return estimated_frequency
128
129 #
130     #####
131
132 # Power factor calculation algorithm
133
134 def estimate_zero_crossing_voltage(voltage, frequency, v_prev_crossing):
135     sample_rate = len(voltage)
136     window_size_start = 1 / (frequency + 0.5)
137     window_size_end = 1 / (frequency - 0.5)
138     # looking for a change in the sign of the wave to determine the zero crossing
139     zero_crossings = np.where(np.diff(np.sign(voltage)))[0]
140     # valid crossings are being saved in this array
141     valid_crossings = []
142     # will be used to return the last valid crossing
143     last_valid_crossing = None
144     # is implemented to prevent that the start of the function is being detected as the
145     # first zero crossing
146     start_time = 0.001
147     # updated = False
148     # v_prev_crossing = None
149
150     for v_crossing in zero_crossings:
151         v_crossing_time = v_crossing / sample_rate
152         # sliding window that has been implemented and the start time that has been
153         # included for prevention
154         if v_crossing_time >= start_time and (
155             v_prev_crossing is None or (window_size_start < (v_crossing_time -
156             v_prev_crossing) < window_size_end)):
157             # Keep the first sample of the zero crossing
158             valid_crossings.append(v_crossing_time)
159             # the zero crossing is being set to previous crossing to be able to use the
160             # sliding window for pre estimation of the place of next zero crossing
161             v_prev_crossing = v_crossing_time
162             # updated = True
163
164     # if updated and valid_crossings:

```

```

157     # if there is a valid crossing it is being returned as well as the previous zero
158     crossing
159     if valid_crossings:
160         last_valid_crossing = valid_crossings[-1]
161     return last_valid_crossing, v_prev_crossing
162
163 def estimate_zero_crossing_current(current, frequency, c_prev_crossing):
164
165     window_size_start = 1 / (frequency + 0.5)
166     window_size_end = 1 / (frequency - 0.5)
167     sample_rate = len(current)
168     zero_crossings = np.where(np.diff(np.sign(current)))[0]
169     valid_crossings = []
170     last_valid_crossing = None
171     # updated = False
172     # c_prev_crossing = None
173     start_time = 0.006 # starting time different since is has a phase shift
174
175     for c_crossing in zero_crossings:
176         c_crossing_time = c_crossing / sample_rate
177         if c_crossing_time >= start_time and (
178             c_prev_crossing is None or window_size_start < (c_crossing_time -
179 c_prev_crossing) < window_size_end):
180             valid_crossings.append(c_crossing_time)
181             c_prev_crossing = c_crossing_time
182             # updated = True
183
184     if valid_crossings:
185         last_valid_crossing = valid_crossings[-1]
186         print("last_valid_crossing:", last_valid_crossing)
187     return last_valid_crossing, c_prev_crossing
188
189 def calculate_power_factor(voltage, current, frequency):
190     # function to calculate the phase difference in radians
191     power_factors = np.array([])
192     mean_pfs = np.array([])
193     current_zero_crossings = []
194     voltage_zero_crossings = []
195     c_prev_crossing = None
196     v_prev_crossing = None
197     voltage_zero_crossing, v_prev_crossing = estimate_zero_crossing_voltage(voltage,
198 frequency, v_prev_crossing)
199     #print("v_crossing:", voltage_zero_crossing)
200     current_zero_crossing, c_prev_crossing = estimate_zero_crossing_current(current,
201 frequency, c_prev_crossing)
202     #print(("current_zero_crossing: ", current_zero_crossing))
203     if voltage_zero_crossing is not None:
204         voltage_zero_crossings.append(voltage_zero_crossing)
205         print("array:", voltage_zero_crossings)
206     if current_zero_crossing is not None:
207         current_zero_crossings.append(current_zero_crossing)
208         print("array:", current_zero_crossings)
209
210     if voltage_zero_crossings and current_zero_crossings:
211         phase_difference = 2 * np.pi * (current_zero_crossings[-1] -
212 voltage_zero_crossings[-1]) * frequency
213         power_factor = np.cos(phase_difference)
214         print(power_factor)
215         voltage_zero_crossings = []
216         current_zero_crossings = []
217         power_factors = np.append(power_factors, power_factor)
218     #if len(power_factors) >= 3:
219     #    mean_pf = np.mean(power_factors[-3:0])
220     #    mean_pfs = np.append(mean_pfs, mean_pf)
221     #    last_mean = mean_pfs[-1]
222     power_factors = []
223     # power factor is being calculated by using the phase difference
224     return power_factor

```

```

223
224
225 # #####
226 # Algorithm for the detection of voltage swells and sags
227
228 def calculate_rms(voltage, frequency):
229     sample_rate = len(voltage)
230     rms_values = []
231     samples_per_period = int(sample_rate / frequency)
232     #print(samples_per_period)
233
234     start_index = 0
235
236     while start_index < sample_rate:
237         # Find the index of the closest value to zero
238         closest_to_zero_index = np.abs(voltage[start_index:]).argmin()
239         zero_crossing = start_index + closest_to_zero_index
240
241         if zero_crossing < sample_rate:
242             start_index = zero_crossing
243             end_index = start_index + samples_per_period
244
245             if end_index <= sample_rate:
246                 window = voltage[start_index:end_index]
247                 #print(window)
248                 rms = np.sqrt(np.mean(np.square(window)))
249                 rms_values.append(rms)
250                 #print(rms)
251                 # Move to the next period
252                 start_index = end_index
253             else:
254                 break
255         else:
256             break
257
258     return rms, rms_values
259
260
261 def voltage_magnitude(signal, frequency):
262     rms, rms_values = calculate_rms(signal, frequency)
263     rms_values_np = np.array(rms_values)
264     voltage_magnitude = np.mean(rms_values_np)
265     return voltage_magnitude
266
267
268 def sag_check(voltage, frequency):
269     rms, rms_values = calculate_rms(voltage, frequency)
270     sag_threshold = 230 * 0.9 # 0.9
271     sag_data = np.array([])
272     sag_event_active = False
273     sag_counter = 0
274     max_sag = 0
275
276     if rms > sag_threshold:
277         sag_event_active = True
278         max_swell = rms
279
280     return max_sag
281
282
283 def swell_check(voltage, frequency):
284     rms, rms_values = calculate_rms(voltage, frequency)
285     swell_threshold = 230 * 1.1
286     swell_data = np.array([])
287
288     max_swell = 0
289
290     if rms > swell_threshold:
291         # print("i am here", rms)

```

```

292     swell_data = np.append(swell_data, rms)
293     # print(swell_data)
294     max_swell = rms
295
296     return max_swell
297
298
299 #
300 #####
301 # Harmonics
302 def apply_low_pass_filter(signal, order=4):
303     sample_rate = len(signal)
304     cutoff_freq = 4000
305     nyquist = 0.5 * sample_rate
306     normal_cutoff = cutoff_freq / nyquist
307     b, a = butter(order, normal_cutoff, btype='low', analog=False)
308     filtered_signal = lfilter(b, a, signal)
309     return filtered_signal
310
311 def detect_harmonics(signal, num_peaks):
312     target_frequency = estimate_frequency_zero_crossing(signal)
313     sample_rate = len(signal)
314     n = len(signal)
315     frequencies = fftfreq(n, d=1 / sample_rate)
316     fft_values = fft(signal)
317
318     positive_frequencies = frequencies[:n // 2]
319     positive_fft_values = np.abs(fft_values[:n // 2])
320
321     # Adjust the threshold based on your signal characteristics
322     threshold = 0.1 * np.max(positive_fft_values)
323
324     peaks, _ = find_peaks(positive_fft_values, height=threshold)
325
326     # Use interpolation for more precise frequency estimation
327     f_interp = interp1d(positive_frequencies, positive_fft_values, kind='quadratic',
328                        fill_value="extrapolate")
329     interpolated_frequencies = np.linspace(positive_frequencies[0],
330     positive_frequencies[-1], 10 * n)
331     interpolated_fft_values = f_interp(interpolated_frequencies)
332
333     # Find peaks in the interpolated spectrum
334     interpolated_peaks, _ = find_peaks(interpolated_fft_values, height=threshold)
335
336     # Sort peaks by their proximity to the target frequency
337     sorted_peaks = sorted(interpolated_peaks, key=lambda x: abs(
338     interpolated_frequencies[x] - target_frequency))
339
340     # Select the first 'num_peaks' peaks
341     selected_peaks = sorted_peaks[:num_peaks]
342
343     detected_frequencies = interpolated_frequencies[selected_peaks]
344
345     # Normalize amplitude by signal length
346     detected_amplitudes = 2 * f_interp(detected_frequencies) / n
347     print(detected_frequencies, detected_amplitudes)
348     return detected_frequencies, detected_amplitudes
349
350 #####
351 # THD ALGORITHM
352 def total_harmonic_distortion(array_of_amplitudes_harmonics):
353     # The first amplitude of the array is the ground frequency amplitude,
354     # and each index represents about which harmonics we talk about
355     new_arr = []
356     numerator_thd_eq = 0
357
358     # The elements in array_of_amplitudes is converted into rms values

```

```

356     for elem in array_of_amplitudes_harmonics:
357         new_elem = elem / np.sqrt(2)
358         new_arr.append(new_elem)
359
360     # The index defines to which harmonic component it belongs, the first element in the
361     # array is the ground frequency
362     ground_freq_amp = new_arr[0]
363     for amplitude in new_arr[1:]:
364         numerator_thd_eq += amplitude ** 2
365
366     # Now we can apply the THD equation to find the THD
367     THD = np.sqrt(numerator_thd_eq) / ground_freq_amp
368
369     # THD is expressed in percentages, so therefore we multiply with 100
370     return THD * 100

```

H.4. Signal Generator

```

1  # #####
2  # EE3L11: Bachelor Graduation Project
3  # GROUP 1:
4  # SUBGROUP: SOFTWARE
5  # MEMBERS: Emirhan Öztürkolu, Marten de Vries, Hüsna Yildiz
6  #
7  # BY Marten de Vries
8  # DATE: 8/1/2024
9  # Description: This program generates a noisy sine wave and stores it
10 # a MySQL database. This program can be used to validate the interface
11 # and the power quality calculation methods.
12 # #####
13
14 # Importing packages
15 import numpy as np
16 import matplotlib.pyplot as plt
17 from scipy.fft import fft, fftfreq
18 from scipy.signal import find_peaks, butter, lfilter
19 from scipy.interpolate import interp1d
20 import datetime
21 from datetime import datetime, timedelta
22 import time
23 import pymysql
24
25 # Define the sample rate
26 sample_rate = 12000
27 count = 0
28 def generate_volt_with_noise(frequency, noise_std, phase):
29     duration = 1
30     amplitude = 325.0
31     t = np.arange(0, duration, 1/sample_rate)
32     sinusoid = amplitude * np.sin(2 * np.pi * frequency * t + phase)
33
34     for i in range(2, 2 + 1):
35         harmonic_freq = i * 50.25
36         harmonic_amplitude = 50
37         sinusoid += harmonic_amplitude * np.sin(2 * np.pi * harmonic_freq * t + phase)
38
39     # Add white noise
40     noise = np.random.normal(0, noise_std, len(t))
41     signal_with_noise = sinusoid + noise
42     return t, signal_with_noise
43
44 def generate_current_with_noise(frequency, noise_std, phase):
45     duration = 1
46     amplitude = 3.0
47     t = np.arange(0, duration, 1/sample_rate)
48     sinusoid = amplitude * np.sin(2 * np.pi * frequency * t + phase)
49     for i in range(2, 2 + 1):
50         harmonic_freq = i * 50.25
51         harmonic_amplitude = 0.5
52         sinusoid += harmonic_amplitude * np.sin(2 * np.pi * harmonic_freq * t + phase)

```

```

53     # Add white noise
54     noise = np.random.normal(0, noise_std, len(t))
55     signal_with_noise = sinusoid + noise
56     return t, signal_with_noise
57
58 host = '192.168.25.34'#'145.94.62.207'
59 user = "remote_software"
60 password = "remote_database"
61 database = "LIVE_DATA"
62 connection = pymysql.connect(
63     host=host,
64     user=user,
65     password=password,
66     db=database
67 )
68 cursor = connection.cursor()
69 cursor.execute("DROP TABLE IF EXISTS Generated_voltage")
70 cursor.execute("DROP TABLE IF EXISTS Generated_current")
71 # Corrected table creation SQL statement
72 cursor.execute("CREATE TABLE IF NOT EXISTS Generated_voltage (id INT AUTO_INCREMENT
73     PRIMARY KEY, Value FLOAT, timestamp_col FLOAT)")
73 cursor.execute("CREATE TABLE IF NOT EXISTS Generated_current (id INT AUTO_INCREMENT
74     PRIMARY KEY, Value FLOAT, timestamp_col FLOAT)")
74
75 #while(True):
76 #cursor.execute("TRUNCATE TABLE Generated_voltage")
77 #cursor.execute("TRUNCATE TABLE Generated_current")
78 start_time = time.monotonic()
79 voltage_time, voltage_values = generate_volt_with_noise(50.25, 3, 0)
80 voltage_time = voltage_time + count
81
82 current_time, current_values = generate_current_with_noise(50.25, 0.5, np.pi/2)
83 current_time = current_time + count
84 # Corrected SQL query for inserting data
85 query = "INSERT INTO Generated_voltage (Value, timestamp_col) VALUES (%s, %s)"
86 data_voltage = list(zip(voltage_values, voltage_time))
87 cursor.executemany(query, data_voltage)
88
89 query = "INSERT INTO Generated_current (Value, timestamp_col) VALUES (%s, %s)"
90 data_current = list(zip(current_values, current_time))
91 cursor.executemany(query, data_current)
92 # Commit changes and close the connection
93 connection.commit()
94 end_time = time.monotonic()
95
96 time.sleep(max(0,1-(end_time-start_time)))
97 print(count)
98 count = count
99 connection.close()

```

H.5. GUI

```

1  #####
2  # EE3L11 : Bachelor Graduation Project
3  # GROUP 1:
4  # SUBGROUP : SOFTWARE
5  # MEMBERS : Emirhan Öztürkoglu , Marten de Vries , Hüsna Yildiz
6  #
7  # Hüsna Yildiz
8  # DATE : 12/1/2024
9  # Description : This is the Graphical User Interface,
10 # on this interface the power quality parameters are displayed
11 #####
12 from PyQt5.QtCore import QTimer
13 import numpy as np
14 from pyqtgraph import PlotWidget
15 from PyQt5 import QtCore, QtGui, QtWidgets
16 import pyqtgraph as pg
17 import pymysql
18 from datetime import datetime, timedelta
19
20
21
22 from allAlgorithms import estimate_frequency_zero_crossing
23 from allAlgorithms import calculate_power_factor, voltage_magnitude, detect_harmonics
24 from allAlgorithms import total_harmonic_distortion, sag_check, swell_check
25
26 class Ui_MainWindow(object):
27     def setupUi(self, MainWindow):
28         MainWindow.setObjectName("MainWindow")
29         MainWindow.resize(1112, 1052)
30         MainWindow.setStyleSheet("
31 {
32     border: none;
33     background-color: transparent;
34     padding: 0;
35     margin: 0;
36     color: #fff;
37 }
38 #centralwidget{
39     background-color: #1f232a;
40 }
41 #leftMenuSubContainer{
42     background-color: #1f232a;
43 }
44 #leftMenuSubContainer QPushButton{
45     text-align: left;
46     padding: 5px 10px;
47     border-top-left-radius: 10px;
48     border-bottom-left-radius: 10px;
49 }
50
51 #centerMenuSubContainer{
52     background-color: #2c313c;
53 }
54 #frame{
55     background-color: #16191d;
56     border-radius: 10px;
57 }
58 #mainBodyContainer{
59     background-color: rgb(168, 165, 115);
60 }
61
62 ")
63         self.centralwidget = QtWidgets.QWidget(MainWindow)
64         self.centralwidget.setStyleSheet("")
65         self.centralwidget.setObjectName("centralwidget")
66         self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
67         self.gridLayout.setContentsMargins(0, 0, 0, 0)
68         self.gridLayout.setObjectName("gridLayout")
69         self.leftMenuContainer = QtWidgets.QWidget(self.centralwidget)

```

```

70     self.leftMenuContainer.setMaximumSize(QtCore.QSize(16777215, 16777215))
71     self.leftMenuContainer.setObjectName("leftMenuContainer")
72     self.verticalLayout = QtWidgets.QVBoxLayout(self.leftMenuContainer)
73     self.verticalLayout.setObjectName("verticalLayout")
74     self.centreMenuContainer = QtWidgets.QWidget(self.leftMenuContainer)
75     self.centreMenuContainer.setMinimumSize(QtCore.QSize(200, 0))
76     self.centreMenuContainer.setObjectName("centreMenuContainer")
77     self.verticalLayout_5 = QtWidgets.QVBoxLayout(self.centreMenuContainer)
78     self.verticalLayout_5.setContentsMargins(0, 0, 0, 0)
79     self.verticalLayout_5.setSpacing(0)
80     self.verticalLayout_5.setObjectName("verticalLayout_5")
81     self.verticalLayout.addWidget(self.centreMenuContainer)
82     self.leftMenuSubContainer = QtWidgets.QWidget(self.leftMenuContainer)
83     self.leftMenuSubContainer.setStyleSheet("")
84     self.leftMenuSubContainer.setObjectName("leftMenuSubContainer")
85     self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.leftMenuSubContainer)
86     self.verticalLayout_2.setContentsMargins(5, 0, 0, 0)
87     self.verticalLayout_2.setSpacing(0)
88     self.verticalLayout_2.setObjectName("verticalLayout_2")
89     self.frame_2 = QtWidgets.QFrame(self.leftMenuSubContainer)
90     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.
    QSizePolicy.Preferred)
91     sizePolicy.setHorizontalStretch(0)
92     sizePolicy.setVerticalStretch(0)
93     sizePolicy.setHeightForWidth(self.frame_2.sizePolicy().hasHeightForWidth())
94     self.frame_2.setSizePolicy(sizePolicy)
95     self.frame_2.setFrameShape(QtWidgets.QFrame.StyledPanel)
96     self.frame_2.setFrameShadow(QtWidgets.QFrame.Raised)
97     self.frame_2.setObjectName("frame_2")
98     self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.frame_2)
99     self.verticalLayout_3.setContentsMargins(0, 0, 0, 0)
100    self.verticalLayout_3.setSpacing(0)
101    self.verticalLayout_3.setObjectName("verticalLayout_3")
102    self.homeBtn = QtWidgets.QPushButton(self.frame_2)
103    self.homeBtn.setStyleSheet("\n
104    \"background-color: rgb(39, 45, 54);")
105    icon = QtGui.QIcon()
106    icon.addPixmap(QtGui.QPixmap("feather/home.svg"), QtGui.QIcon.Normal, QtGui.
    QIcon.Off)
107    self.homeBtn.setIcon(icon)
108    self.homeBtn.setIconSize(QtCore.QSize(24, 24))
109    self.homeBtn.setObjectName("homeBtn")
110    self.verticalLayout_3.addWidget(self.homeBtn)
111    self.dataBtn = QtWidgets.QPushButton(self.frame_2)
112    self.dataBtn.setStyleSheet("")
113    icon1 = QtGui.QIcon()
114    icon1.addPixmap(QtGui.QPixmap("feather/activity.svg"), QtGui.QIcon.Normal,
    QtGui.QIcon.Off)
115    self.dataBtn.setIcon(icon1)
116    self.dataBtn.setIconSize(QtCore.QSize(24, 24))
117    self.dataBtn.setObjectName("dataBtn")
118    self.verticalLayout_3.addWidget(self.dataBtn)
119    # currentBtn
120    self.currentBtn = QtWidgets.QPushButton(self.frame_2)
121    self.currentBtn.setStyleSheet("")
122    icon1 = QtGui.QIcon()
123    icon1.addPixmap(QtGui.QPixmap("feather/activity.svg"), QtGui.QIcon.Normal,
    QtGui.QIcon.Off)
124    self.currentBtn.setIcon(icon1)
125    self.currentBtn.setIconSize(QtCore.QSize(24, 24))
126    self.currentBtn.setObjectName("currentBtn")
127    self.verticalLayout_3.addWidget(self.currentBtn)
128    self.reportsBtn = QtWidgets.QPushButton(self.frame_2)
129    self.reportsBtn.setStyleSheet("")
130    icon2 = QtGui.QIcon()
131    icon2.addPixmap(QtGui.QPixmap("feather/arrow-up.svg"), QtGui.QIcon.Normal,
    QtGui.QIcon.Off)
132    self.reportsBtn.setIcon(icon2)
133    self.reportsBtn.setIconSize(QtCore.QSize(24, 24))
134    self.reportsBtn.setObjectName("reportsBtn")
135    self.verticalLayout_3.addWidget(self.reportsBtn)

```

```

136         self.verticalLayout_2.addWidget(self.frame_2)
137         self.resultsBtn = QtWidgets.QPushButton(self.leftMenuSubContainer)
138         icon3 = QtGui.QIcon()
139         icon3.addPixmap(QtGui.QPixmap("feather/printer.svg"), QtGui.QIcon.Normal, QtGui
        .QIcon.Off)
140         self.resultsBtn.setIcon(icon3)
141         self.resultsBtn.setIconSize(QtCore.QSize(24, 24))
142         self.resultsBtn.setObjectName("resultsBtn")
143         self.verticalLayout_2.addWidget(self.resultsBtn)
144         spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
        QtWidgets.QSizePolicy.Expanding)
145         self.verticalLayout_2.addItem(spacerItem)
146         self.frame_3 = QtWidgets.QFrame(self.leftMenuSubContainer)
147         self.frame_3 setFrameShape(QtWidgets.QFrame.StyledPanel)
148         self.frame_3 setFrameShadow(QtWidgets.QFrame.Raised)
149         self.frame_3.setObjectName("frame_3")
150         self.verticalLayout_4 = QtWidgets.QVBoxLayout(self.frame_3)
151         self.verticalLayout_4.setContentsMargins(0, 0, 0, 0)
152         self.verticalLayout_4.setSpacing(0)
153         self.verticalLayout_4.setObjectName("verticalLayout_4")
154         self.informationBtn = QtWidgets.QPushButton(self.frame_3)
155         self.informationBtn.setStyleSheet("")
156         icon4 = QtGui.QIcon()
157         icon4.addPixmap(QtGui.QPixmap("feather/info.svg"), QtGui.QIcon.Normal, QtGui
        .QIcon.Off)
158         self.informationBtn.setIcon(icon4)
159         self.informationBtn.setIconSize(QtCore.QSize(24, 24))
160         self.informationBtn.setObjectName("informationBtn")
161         self.verticalLayout_4.addWidget(self.informationBtn)
162         self.verticalLayout_2.addWidget(self.frame_3)
163         self.verticalLayout.addWidget(self.leftMenuSubContainer)
164         self.gridLayout.addWidget(self.leftMenuContainer, 0, 0, 1, 1)
165         self.mainBodyContainer = QtWidgets.QWidget(self.centralwidget)
166         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.
        QSizePolicy.Preferred)
167         sizePolicy.setHorizontalStretch(0)
168         sizePolicy.setVerticalStretch(0)
169         sizePolicy.setHeightForWidth(self.mainBodyContainer.sizePolicy().
        hasHeightForWidth())
170         self.mainBodyContainer.setSizePolicy(sizePolicy)
171         self.mainBodyContainer.setStyleSheet("background-color: rgb(255, 255, 255);")
172         self.mainBodyContainer.setObjectName("mainBodyContainer")
173         self.verticalLayout_10 = QtWidgets.QVBoxLayout(self.mainBodyContainer)
174         self.verticalLayout_10.setObjectName("verticalLayout_10")
175         self.headerContainer = QtWidgets.QWidget(self.mainBodyContainer)
176         self.headerContainer.setMinimumSize(QtCore.QSize(0, 0))
177         self.headerContainer.setStyleSheet("background-color: rgb(31, 35, 42);")
178         self.headerContainer.setObjectName("headerContainer")
179         self.horizontalLayout_4 = QtWidgets.QHBoxLayout(self.headerContainer)
180         self.horizontalLayout_4.setContentsMargins(0, 0, 0, 0)
181         self.horizontalLayout_4.setSpacing(0)
182         self.horizontalLayout_4.setObjectName("horizontalLayout_4")
183         self.frame_4 = QtWidgets.QFrame(self.headerContainer)
184         self.frame_4 setFrameShape(QtWidgets.QFrame.StyledPanel)
185         self.frame_4 setFrameShadow(QtWidgets.QFrame.Raised)
186         self.frame_4.setObjectName("frame_4")
187         self.horizontalLayout_5 = QtWidgets.QHBoxLayout(self.frame_4)
188         self.horizontalLayout_5.setContentsMargins(0, 0, 0, 0)
189         self.horizontalLayout_5.setSpacing(0)
190         self.horizontalLayout_5.setObjectName("horizontalLayout_5")
191         self.label_5 = QtWidgets.QLabel(self.frame_4)
192         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.
        QSizePolicy.Preferred)
193         sizePolicy.setHorizontalStretch(0)
194         sizePolicy.setVerticalStretch(0)
195         sizePolicy.setHeightForWidth(self.label_5.sizePolicy().hasHeightForWidth())
196         self.label_5.setSizePolicy(sizePolicy)
197         self.label_5.setMaximumSize(QtCore.QSize(40, 40))
198         self.label_5.setLayoutDirection(QtCore.Qt.LeftToRight)
199         self.label_5.setText("")
200         self.label_5.setPixmap(QtGui.QPixmap("logo.png"))

```

```

201     self.label_5.setScaledContents(True)
202     self.label_5.setAlignment(QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
203     self.label_5.setObjectName("label_5")
204     self.horizontalLayout_5.addWidget(self.label_5)
205     self.label_6 = QtWidgets.QLabel(self.frame_4)
206     font = QtGui.QFont()
207     font.setPointSize(10)
208     font.setBold(True)
209     font.setWeight(75)
210     self.label_6.setFont(font)
211     self.label_6.setObjectName("label_6")
212     self.horizontalLayout_5.addWidget(self.label_6)
213     self.horizontalLayout_4.addWidget(self.frame_4, 0, QtCore.Qt.AlignLeft)
214     self.frame_6 = QtWidgets.QFrame(self.headerContainer)
215     self.frame_6.setFrameShape(QtWidgets.QFrame.StyledPanel)
216     self.frame_6.setFrameShadow(QtWidgets.QFrame.Raised)
217     self.frame_6.setObjectName("frame_6")
218     self.horizontalLayout_3 = QtWidgets.QHBoxLayout(self.frame_6)
219     self.horizontalLayout_3.setContentsMargins(-1, 11, -1, 11)
220     self.horizontalLayout_3.setSpacing(7)
221     self.horizontalLayout_3.setObjectName("horizontalLayout_3")
222     self.minimizeWindowBtn = QtWidgets.QPushButton(self.frame_6)
223     self.minimizeWindowBtn.setText("")
224     icon5 = QtGui.QIcon()
225     icon5.addPixmap(QtGui.QPixmap("feather/minus.svg"), QtGui.QIcon.Normal, QtGui.
QIcon.Off)
226     self.minimizeWindowBtn.setIcon(icon5)
227     self.minimizeWindowBtn.setIconSize(QtCore.QSize(20, 20))
228     self.minimizeWindowBtn.setObjectName("minimizeWindowBtn")
229     self.horizontalLayout_3.addWidget(self.minimizeWindowBtn)
230     self.maximizeWindowBtn = QtWidgets.QPushButton(self.frame_6)
231     self.maximizeWindowBtn.setText("")
232     icon6 = QtGui.QIcon()
233     icon6.addPixmap(QtGui.QPixmap("feather/square.svg"), QtGui.QIcon.Normal, QtGui.
QIcon.Off)
234     self.maximizeWindowBtn.setIcon(icon6)
235     self.maximizeWindowBtn.setIconSize(QtCore.QSize(20, 20))
236     self.maximizeWindowBtn.setObjectName("maximizeWindowBtn")
237     self.horizontalLayout_3.addWidget(self.maximizeWindowBtn)
238     self.closeWindowBtn = QtWidgets.QPushButton(self.frame_6)
239     self.closeWindowBtn.setText("")
240     icon7 = QtGui.QIcon()
241     icon7.addPixmap(QtGui.QPixmap("feather/x.svg"), QtGui.QIcon.Normal, QtGui.QIcon
.Off)
242     self.closeWindowBtn.setIcon(icon7)
243     self.closeWindowBtn.setIconSize(QtCore.QSize(20, 20))
244     self.closeWindowBtn.setObjectName("closeWindowBtn")
245     self.horizontalLayout_3.addWidget(self.closeWindowBtn)
246     self.horizontalLayout_4.addWidget(self.frame_6, 0, QtCore.Qt.AlignRight)
247     self.verticalLayout_10.addWidget(self.headerContainer, 0, QtCore.Qt.AlignTop)
248     self.stackedWidget = QtWidgets.QStackedWidget(self.mainBodyContainer)
249     self.stackedWidget.setObjectName("stackedWidget")
250     self.homePage = QtWidgets.QWidget()
251     self.homePage.setObjectName("homePage")
252     self.verticalLayout_8 = QtWidgets.QVBoxLayout(self.homePage)
253     self.verticalLayout_8.setObjectName("verticalLayout_8")
254     self.frame = QtWidgets.QFrame(self.homePage)
255     self.frame.setStyleSheet("")
256     self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
257     self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
258     self.frame.setObjectName("frame")
259     self.verticalLayout_11 = QtWidgets.QVBoxLayout(self.frame)
260     self.verticalLayout_11.setObjectName("verticalLayout_11")
261     self.label = QtWidgets.QLabel(self.frame)
262     self.label.setText("")
263     self.label.setPixmap(QtGui.QPixmap("logo.png"))
264     self.label.setObjectName("label")
265     self.verticalLayout_11.addWidget(self.label, 0, QtCore.Qt.AlignHCenter)
266     self.homePageTitle = QtWidgets.QTextEdit(self.frame)
267     self.homePageTitle.setObjectName("homePageTitle")
268     self.verticalLayout_11.addWidget(self.homePageTitle)

```

```

269     self.verticalLayout_8.addWidget(self.frame)
270     self.stackedWidget.addWidget(self.homePage)
271     self.currentVoltagePage = QtWidgets.QWidget()
272     self.currentVoltagePage.setObjectName("currentVoltagePage")
273     self.verticalLayout_6 = QtWidgets.QVBoxLayout(self.currentVoltagePage)
274     self.verticalLayout_6.setObjectName("verticalLayout_6")
275     self.frame_7 = QtWidgets.QFrame(self.currentVoltagePage)
276     self.frame_7.setFrameShape(QtWidgets.QFrame.StyledPanel)
277     self.frame_7.setFrameShadow(QtWidgets.QFrame.Raised)
278     self.frame_7.setObjectName("frame_7")
279     self.verticalLayout_12 = QtWidgets.QVBoxLayout(self.frame_7)
280     self.verticalLayout_12.setObjectName("verticalLayout_12")
281     self.graphWidget = PlotWidget(self.frame_7)
282     self.graphWidget.setObjectName("graphWidget")
283     self.verticalLayout_12.addWidget(self.graphWidget)
284     self.graphWidget.setTitle("Voltage Signal")
285     self.graphWidget.setTitle("Voltage Signals", color="w", size="20pt")
286     styles = {"color": "white", "font-size": "18px"}
287     self.graphWidget.setLabel("left", "Voltage", units="V", **styles)
288     self.graphWidget.setLabel("bottom", "Time", units="s", **styles)
289     self.graphWidget.addLegend()
290     self.graphWidget.showGrid(x=True, y=True)
291     self.plot_incoming_data_voltage()
292     self.timer = QTimer()
293     self.timer.timeout.connect(self.plot_incoming_data_voltage)
294     self.timer.start(1000)
295     self.verticalLayout_6.addWidget(self.frame_7)
296     self.stackedWidget.addWidget(self.currentVoltagePage)
297     self.currentPage = QtWidgets.QWidget()
298     self.currentPage.setObjectName("currentSignalPlotPage")
299     self.verticalLayout_14 = QtWidgets.QVBoxLayout(self.currentPage)
300     self.verticalLayout_14.setObjectName("verticalLayout_14")
301     self.frame_current = QtWidgets.QFrame(self.currentPage)
302     self.frame_current.setFrameShape(QtWidgets.QFrame.StyledPanel)
303     self.frame_current.setFrameShadow(QtWidgets.QFrame.Raised)
304     self.frame_current.setObjectName("frame_current")
305     self.verticalLayout_15 = QtWidgets.QVBoxLayout(self.frame_current)
306     self.verticalLayout_15.setObjectName("verticalLayout_15")
307     self.graphWidget_current = PlotWidget(self.frame_current)
308     self.graphWidget_current.setObjectName("graphWidget_current")
309     self.verticalLayout_15.addWidget(self.graphWidget_current)
310     self.graphWidget_current.setTitle("Current Signal")
311     self.graphWidget_current.setLabel("left", "Current", units="A", **styles)
312     self.graphWidget_current.setLabel("bottom", "Time", units="s", **styles)
313     self.graphWidget_current.addLegend()
314     self.graphWidget_current.showGrid(x=True, y=True)
315     self.plot_incoming_data_current()
316     self.timer = QTimer()
317     self.timer.timeout.connect(self.plot_incoming_data_current)
318     self.timer.start(1000)
319     self.verticalLayout_14.addWidget(self.frame_current)
320     self.stackedWidget.addWidget(self.currentPage)
321     self.harmonicsPage = QtWidgets.QWidget()
322     self.harmonicsPage.setObjectName("harmonicsPage")
323     self.verticalLayout_7 = QtWidgets.QVBoxLayout(self.harmonicsPage)
324     self.verticalLayout_7.setObjectName("verticalLayout_7")
325     self.graphicsView = PlotWidget(self.harmonicsPage)
326     self.graphicsView.setObjectName("graphicsView")
327     self.verticalLayout_7.addWidget(self.graphicsView)
328     self.graphicsView.setTitle("Real-time Harmonic Analysis of Current and Voltage
Signals", color="w", size="20pt")
329     styles = {"color": "white", "font-size": "18px"}
330     self.graphicsView.setLabel("left", "Amplitude", units="V", **styles)
331     self.graphicsView.setLabel("bottom", "Frequency", units="Hz", **styles)
332     self.initialize_harmonics_plot()
333     self.timer_harmonics = QTimer()
334     self.timer_harmonics.timeout.connect(self.initialize_harmonics_plot)
335     self.timer_harmonics.start(1000)
336     self.stackedWidget.addWidget(self.harmonicsPage)
337     self.resultsPage = QtWidgets.QWidget()
338     self.resultsPage.setObjectName("resultsPage")

```

```

339     self.verticalLayout_13 = QtWidgets.QVBoxLayout(self.resultsPage)
340     self.verticalLayout_13.setObjectName("verticalLayout_13")
341     self.widget = QtWidgets.QWidget(self.resultsPage)
342     self.widget.setObjectName("widget")
343     self.gridLayout_2 = QtWidgets.QGridLayout(self.widget)
344     self.gridLayout_2.setObjectName("gridLayout_2")
345     self.thdNum = QtWidgets.QLabel(self.widget)
346     self.thdNum.setStyleSheet("color: rgb(0, 0, 0);\n"
347                               "font: 18pt \"Times New Roman\";")
348     self.thdNum.setObjectName("thdNum")
349     self.gridLayout_2.addWidget(self.thdNum, 5, 0, 1, 1, QtCore.Qt.AlignLeft)
350     self.thdVoltageValue = QtWidgets.QLabel(self.widget)
351     self.thdVoltageValue.setStyleSheet("color: rgb(0, 0, 0);\n"
352                                       "font: 18pt \"Times New Roman\";")
353     self.thdVoltageValue.setObjectName("thdVoltageValue")
354     self.gridLayout_2.addWidget(self.thdVoltageValue, 6, 1, 1, 1)
355     self.gridFreqValue = QtWidgets.QLabel(self.widget)
356     self.gridFreqValue.setStyleSheet("color: rgb(0, 0, 0);\n"
357                                     "font: 18pt \"Times New Roman\";")
358     self.gridFreqValue.setObjectName("gridFreqValue")
359     self.gridLayout_2.addWidget(self.gridFreqValue, 8, 1, 1, 1)
360     self.powerFactorNum = QtWidgets.QLabel(self.widget)
361     self.powerFactorNum.setStyleSheet("color: rgb(0, 0, 0);\n"
362                                      "font: 18pt \"Times New Roman\";")
363     self.powerFactorNum.setObjectName("powerFactorNum")
364     self.gridLayout_2.addWidget(self.powerFactorNum, 2, 0, 1, 1, QtCore.Qt.
AlignLeft)
365     self.vrmValue = QtWidgets.QLabel(self.widget)
366     self.vrmValue.setStyleSheet("color: rgb(0, 0, 0);\n"
367                                "font: 18pt \"Times New Roman\";")
368     self.vrmValue.setObjectName("vrmValue")
369     self.gridLayout_2.addWidget(self.vrmValue, 3, 1, 1, 1)
370     self.thdCurrentValue = QtWidgets.QLabel(self.widget)
371     self.thdCurrentValue.setStyleSheet("color: rgb(0, 0, 0);\n"
372                                       "font: 18pt \"Times New Roman\";")
373     self.thdCurrentValue.setObjectName("thdCurrentValue")
374     self.gridLayout_2.addWidget(self.thdCurrentValue, 5, 1, 1, 1)
375     self.voltageSagSwellValue = QtWidgets.QLabel(self.widget)
376     self.voltageSagSwellValue.setStyleSheet("color: rgb(0, 0, 0);\n"
377                                             "font: 18pt \"Times New Roman\";")
378     self.voltageSagSwellValue.setObjectName("voltageSagSwellValue")
379     self.gridLayout_2.addWidget(self.voltageSagSwellValue, 7, 1, 1, 1)
380     self.powerFactorValue = QtWidgets.QLabel(self.widget)
381     self.powerFactorValue.setStyleSheet("color: rgb(0, 0, 0);\n"
382                                        "font: 18pt \"Times New Roman\";")
383     self.powerFactorValue.setObjectName("powerFactorValue")
384     self.gridLayout_2.addWidget(self.powerFactorValue, 2, 1, 1, 1)
385     self.thdVoltage = QtWidgets.QLabel(self.widget)
386     self.thdVoltage.setStyleSheet("color: rgb(0, 0, 0);\n"
387                                  "font: 18pt \"Times New Roman\";")
388     self.thdVoltage.setObjectName("thdVoltage")
389     self.gridLayout_2.addWidget(self.thdVoltage, 6, 0, 1, 1, QtCore.Qt.AlignLeft)
390     self.voltageSagSwellNum = QtWidgets.QLabel(self.widget)
391     self.voltageSagSwellNum.setStyleSheet("color: rgb(0, 0, 0);\n"
392                                          "font: 18pt \"Times New Roman\";\n"
393                                          "")
394     self.voltageSagSwellNum.setObjectName("voltageSagSwellNum")
395     self.gridLayout_2.addWidget(self.voltageSagSwellNum, 7, 0, 1, 1, QtCore.Qt.
AlignLeft)
396     self.gridFreqNum = QtWidgets.QLabel(self.widget)
397     self.gridFreqNum.setStyleSheet("color: rgb(0, 0, 0);\n"
398                                   "font: 18pt \"Times New Roman\";")
399     self.gridFreqNum.setObjectName("gridFreqNum")
400     self.gridLayout_2.addWidget(self.gridFreqNum, 8, 0, 1, 1, QtCore.Qt.AlignLeft)
401     self.vrmsNum = QtWidgets.QLabel(self.widget)
402     self.vrmsNum.setStyleSheet("color: rgb(0, 0, 0);\n"
403                               "font: 18pt \"Times New Roman\";")
404     self.vrmsNum.setObjectName("vrmsNum")
405     self.gridLayout_2.addWidget(self.vrmsNum, 3, 0, 1, 1, QtCore.Qt.AlignLeft)
406     self.titleResultPage = QtWidgets.QLabel(self.widget)
407     self.titleResultPage.setStyleSheet("color: rgb(0, 0, 0);\n"

```

```

408                                     "font: 75 60pt \\"Times New Roman\\";")
409     self.titleResultPage.setObjectName("titleResultPage")
410     self.gridLayout_2.addWidget(self.titleResultPage, 0, 0, 1, 1)
411
412     _translate = QtCore.QCoreApplication.translate
413     # QTimer to update the values every second
414     self.timer = QTimer()
415     self.timer.timeout.connect(self.update_power_factor)
416     self.timer.timeout.connect(self.update_thd_voltage)
417     self.timer.timeout.connect(self.update_voltage_sag_swell)
418     self.timer.timeout.connect(self.update_thd_current)
419     self.timer.timeout.connect(self.update_vrm)
420     self.timer.timeout.connect(self.update_grid_freq)
421     self.timer.start(1000)
422     self.verticalLayout_13.addWidget(self.widget)
423     self.stackedWidget.addWidget(self.resultsPage)
424     ##### Information page
425     self.informationPage = QtWidgets.QWidget()
426     self.informationPage.setObjectName("informationPage")
427     self.verticalLayout_9 = QtWidgets.QVBoxLayout(self.informationPage)
428     self.verticalLayout_9.setObjectName("verticalLayout_9")
429     self.informationText = QtWidgets.QTextEdit(self.informationPage)
430     self.informationText.setObjectName("informationText")
431     self.verticalLayout_9.addWidget(self.informationText)
432     self.stackedWidget.addWidget(self.informationPage)
433     self.verticalLayout_10.addWidget(self.stackedWidget)
434     self.mainBodyContent = QtWidgets.QWidget(self.mainBodyContainer)
435     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.
436     QSizePolicy.Expanding)
437     sizePolicy.setHorizontalStretch(0)
438     sizePolicy.setVerticalStretch(0)
439     sizePolicy.setHeightForWidth(self.mainBodyContent.sizePolicy().
440     hasHeightForWidth())
441     self.mainBodyContent.setSizePolicy(sizePolicy)
442     self.mainBodyContent.setStyleSheet("background-color: rgb(31, 35, 42);")
443     self.mainBodyContent.setObjectName("mainBodyContent")
444
445     self.homeBtn.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.
446     homePage))
447     self.dataBtn.clicked.connect(lambda : self.stackedWidget.setCurrentWidget(self.
448     currentVoltagePage))
449     #added new button
450     self.currentBtn.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(
451     self.currentPage))
452     self.reportsBtn.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(
453     self.harmonicsPage))
454     self.informationBtn.clicked.connect(lambda: self.stackedWidget.setCurrentWidget
455     (self.informationPage))
456     self.resultsBtn.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(
457     self.resultsPage))
458
459     self.horizontalLayout_7 = QtWidgets.QHBoxLayout(self.mainBodyContent)
460     self.horizontalLayout_7.setContentsMargins(-1, -1, 0, -1)
461     self.horizontalLayout_7.setSpacing(7)
462     self.horizontalLayout_7.setObjectName("horizontalLayout_7")
463     self.label_7 = QtWidgets.QLabel(self.mainBodyContent)
464     self.label_7.setObjectName("label_7")
465     self.horizontalLayout_7.addWidget(self.label_7)
466     self.verticalLayout_10.addWidget(self.mainBodyContent, 0, QtCore.Qt.AlignBottom
467     )
468
469     self.gridLayout.addWidget(self.mainBodyContainer, 0, 1, 1, 1)
470     MainWindow.setCentralWidget(self.centralWidget)
471
472     self.retranslateUi(MainWindow)
473     self.stackedWidget.setCurrentIndex(0)
474     self.closeWindowBtn.clicked.connect(MainWindow.close)
475     self.maximizeWindowBtn.clicked.connect(MainWindow.showMaximized)
476     self.minimizeWindowBtn.clicked.connect(MainWindow.showMinimized)
477     QtCore.QMetaObject.connectSlotsByName(MainWindow)

```

```

470
471
472 def voltage_signal(self):
473     timestamps_Voltage = []
474     values_Voltage = []
475
476     host = '145.94.62.207'
477     user = "remote_software"
478     password = "remote_database"
479     database = "LIVE_DATA"
480     connection = pymysql.connect(
481         host=host,
482         user=user,
483         password=password,
484         db=database
485     )
486     cursor = connection.cursor()
487     timerange = timedelta(seconds=1)
488     start_time = datetime.now() - timedelta(seconds=1)
489     end_time = start_time + timerange
490
491     query = f"SELECT Value, timestamp_col FROM Voltage_Data WHERE timestamp_col >=
'{start_time}' AND timestamp_col < '{end_time}'"
492     cursor.execute(query)
493     rows = cursor.fetchall()
494
495     for row in rows:
496         timestamps_Voltage.append(row[1])
497         values_Voltage.append(row[0])
498     print(len(values_Voltage))
499     print(values_Voltage)
500     voltage_values_numpy = np.array(values_Voltage)
501     # Mean of the signal
502     mean_voltage = np.mean(values_Voltage)
503     # Biased voltage values
504     voltage_signal_with_bias = voltage_values_numpy - mean_voltage
505     return voltage_signal_with_bias
506
507
508 def current_signal(self):
509     timestamps_Current_Low = []
510     timestamps_Current_High = []
511
512     values_Current_Low = []
513     values_Current_High = []
514
515     host = '145.94.62.207'
516     user = "remote_software"
517     password = "remote_database"
518     database = "LIVE_DATA"
519     connection = pymysql.connect(
520         host=host,
521         user=user,
522         password=password,
523         db=database
524     )
525     cursor = connection.cursor()
526     timerange = timedelta(seconds=1)
527     start_time = datetime.now() - timedelta(seconds=1)
528     end_time = start_time + timerange
529
530     query = f"SELECT Value, timestamp_col FROM Current_Data_Low WHERE timestamp_col
>= '{start_time}' AND timestamp_col < '{end_time}'"
531     cursor.execute(query)
532     rows = cursor.fetchall()
533
534     for row in rows:
535         timestamps_Current_Low.append(row[1])
536         values_Current_Low.append(row[0])
537
538     query = f"SELECT Value, timestamp_col FROM Current_Data_High WHERE

```

```

timestamp_col >= '{start_time}' AND timestamp_col < '{end_time}''
539     cursor.execute(query)
540     rows = cursor.fetchall()
541
542     for row in rows:
543         timestamps_Current_High.append(row[1])
544         values_Current_High.append(row[0])
545     print(" Length of Current Low: ", len(values_Current_Low), " en length of
Current High :",
546           len(values_Current_High))
547     current_values_low_numpy = np.array(values_Current_Low)
548     current_values_high_numpy = np.array(values_Current_High)
549
550     # for no error i added this piece
551     if len(values_Current_High) > 0:
552         max_current_High = np.amax(values_Current_High)
553     else:
554         max_current_High = 0 # or any default value you want
555
556     if max_current_High > 4 * np.sqrt(2):
557         current_signal_with_bias = current_values_high_numpy - (np.amax(
current_values_high_numpy) / 2) * 1.02
558
559     else:
560         # current_signal_with_bias = current_values_low_numpy - (np.amax(
current_values_low_numpy)/2)*1.03
561         if len(current_values_low_numpy) > 0:
562             max_current_High = np.amax(current_values_low_numpy)
563             current_signal_with_bias = current_values_low_numpy - (max_current_High
/ 2) * 1.03
564         else:
565             max_current_High = 0 # or any default value you want
566             current_signal_with_bias = np.array([])
567     return current_signal_with_bias
568
569
570 def plot_incoming_data_current(self):
571     timestamps_Current_Low = []
572     timestamps_Current_High = []
573
574     values_Current_Low = []
575     values_Current_High = []
576
577     host = '145.94.62.207'
578     user = "remote_software"
579     password = "remote_database"
580     database = "LIVE_DATA"
581     connection = pymysql.connect(
582         host=host,
583         user=user,
584         password=password,
585         db=database
586     )
587     cursor = connection.cursor()
588     timerange = timedelta(seconds=1)
589     start_time = datetime.now() - timedelta(seconds=1)
590     end_time = start_time + timerange
591
592     query = f"SELECT Value, timestamp_col FROM Current_Data_Low WHERE timestamp_col
>= '{start_time}' AND timestamp_col < '{end_time}''
593     cursor.execute(query)
594     rows = cursor.fetchall()
595
596     for row in rows:
597         timestamps_Current_Low.append(row[1])
598         values_Current_Low.append(row[0])
599
600     query = f"SELECT Value, timestamp_col FROM Current_Data_High WHERE
timestamp_col >= '{start_time}' AND timestamp_col < '{end_time}''
601     cursor.execute(query)
602     rows = cursor.fetchall()

```

```

603
604     for row in rows:
605         timestamps_Current_High.append(row[1])
606         values_Current_High.append(row[0])
607     print(" Length of Current Low: ", len(values_Current_Low), " en length of
Current High :", len(values_Current_High))
608     current_values_low_numpy = np.array(values_Current_Low)
609     current_values_high_numpy = np.array(values_Current_High)
610
611     # for no error i added this piece
612     if len(values_Current_High) > 0:
613         max_current_High = np.amax(values_Current_High)
614     else:
615         max_current_High = 0 # or any default value you want
616
617     if max_current_High > 4 * np.sqrt(2):
618         current_signal_with_bias = current_values_high_numpy - (np.amax(
current_values_high_numpy) / 2) * 1.02
619
620     else:
621         # current_signal_with_bias = current_values_low_numpy - (np.amax(
current_values_low_numpy)/2)*1.03
622         if len(current_values_low_numpy) > 0:
623             max_current_High = np.amax(current_values_low_numpy)
624             current_signal_with_bias = current_values_low_numpy - (max_current_High
/ 2) * 1.03
625         else:
626             max_current_High = 0 # or any default value you want
627             current_signal_with_bias = np.array([])
628
629     ### plotting
630     self.graphWidget_current.clear()
631     plottime = np.linspace(0, 1, len(current_signal_with_bias))
632     # Plot the voltage signal
633     pen_current = pg.mkPen(color=(0, 255, 0), width=3)
634     self.graphWidget_current.plot(plottime, current_signal_with_bias, pen=
pen_current, name="Voltage")
635
636     pen = pg.mkPen(color=(255, 0, 0), width=3)
637     pen2 = pg.mkPen(color=(0, 255, 0), width=3)
638
639     def plot_incoming_data_voltage(self):
640         timestamps_Voltage = []
641         values_Voltage = []
642
643         host = '145.94.62.207'
644         user = "remote_software"
645         password = "remote_database"
646         database = "LIVE_DATA"
647         connection = pymysql.connect(
648             host=host,
649             user=user,
650             password=password,
651             db=database
652         )
653         cursor = connection.cursor()
654         timerange = timedelta(seconds=1)
655         start_time = datetime.now() - timedelta(seconds=1)
656         end_time = start_time + timerange
657
658         query = f"SELECT Value, timestamp_col FROM Voltage_Data WHERE timestamp_col
>= '{start_time}' AND timestamp_col < '{end_time}'"
659         cursor.execute(query)
660         rows = cursor.fetchall()
661
662         for row in rows:
663             timestamps_Voltage.append(row[1])
664             values_Voltage.append(row[0])
665         print(len(values_Voltage))
666         print(values_Voltage)
667         voltage_values_numpy = np.array(values_Voltage)

```

```

668         # Mean of the signal
669         mean_voltage = np.mean(values_Voltage)
670         # Biased voltage values
671         voltage_signal_with_bias = voltage_values_numpy - mean_voltage
672         # clear previous plot to plot again each 1 second
673         self.graphWidget.clear()
674         plottime = np.linspace(0, 1, len(values_Voltage))
675         # Plot the voltage signal
676         pen_voltage = pg.mkPen(color=(0, 255, 0), width=3)
677         self.graphWidget.plot(plottime, voltage_signal_with_bias, pen=pen_voltage,
name="Voltage")

678
679         pen = pg.mkPen(color=(255, 0, 0), width=3)
680         pen2 = pg.mkPen(color=(0, 255, 0), width=3)
681
682     def initialize_harmonics_plot(self):
683         # voltage signal THD
684         # Initialize or update your harmonics data here
685         voltage_signal = self.voltage_signal()
686         current_signal = self.current_signal() #current THD
687         detected_frequencies, detected_amplitudes = detect_harmonics(voltage_signal,
100)
688         self.harmonics_data_freq = detected_frequencies # freq data harmonics
689         self.harmonics_data_amp = detected_amplitudes # amplitude data harmonics
690
691         # Create a bar graph item
692         self.harmonics_plot = pg.BarGraphItem(
693             x=self.harmonics_data_freq,
694             height=self.harmonics_data_amp,
695             width=0.1, # Adjust the width of the bars as needed
696             brush='r' # Color of the bars (replace with your desired color)
697         )
698         self.graphicsView.addItem(self.harmonics_plot)
699
700
701     def retranslateUi(self, MainWindow):
702         _translate = QtCore.QCoreApplication.translate
703         MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
704         self.homeBtn.setToolTip(_translate("MainWindow", "Home"))
705         self.homeBtn.setText(_translate("MainWindow", "Home"))
706         self.dataBtn.setToolTip(_translate("MainWindow", "<html><head/><body><p align
=\"justify\"><span style=\" color:#000000;\">Data Analysis</span></p></body></html>
"))
707         self.dataBtn.setText(_translate("MainWindow", "Voltage signal plot"))
708         self.reportsBtn.setToolTip(_translate("MainWindow", "<html><head/><body><p><
span style=\" color:#000000;\">View reports</span></p></body></html>"))
709         self.currentBtn.setText(_translate("MainWindow", "Current signal plot"))
710         self.reportsBtn.setText(_translate("MainWindow", "Harmonics present in signals"
))
711         self.resultsBtn.setText(_translate("MainWindow", "Results"))
712         self.informationBtn.setToolTip(_translate("MainWindow", "<html><head/><body><p
><span style=\" color:#000000;\">Information about application</span></p></body></
html>"))
713         self.informationBtn.setText(_translate("MainWindow", "Information"))
714         self.label_6.setText(_translate("MainWindow", "Power Quality Quantizer"))
715         self.homePageTitle.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC
\"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:7.8pt; font-
weight:400; font-style:normal;\n\">\n"
"<p align=\"center\" style=\" margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;><span style=\" font-size
:20pt; font-weight:600; color:#262c35;\">Power Quality Quantizer</span><span style
=\" font-size:20pt; font-weight:600; color:#262c35; vertical-align:super;\">TM</
span></p></body></html>"))
716
717         self.thdNum.setText(_translate("MainWindow", "THD current"))
718         self.thdVoltageValue.setText(_translate("MainWindow", "None"))
719         self.gridFreqValue.setText(_translate("MainWindow", "None"))
720         self.powerFactorNum.setText(_translate("MainWindow", "Power factor"))
721         self.vrmValue.setText(_translate("MainWindow", "None"))

```

```

725     self.thdCurrentValue.setText(_translate("MainWindow", "None"))
726     self.voltageSagSwellValue.setText(_translate("MainWindow", "None"))
727     self.powerFactorValue.setText(_translate("MainWindow", "None"))
728     self.thdVoltage.setText(_translate("MainWindow", "THD voltage"))
729     self.voltageSagSwellNum.setText(_translate("MainWindow", "Voltage sag/swell"))
730     self.gridFreqNum.setText(_translate("MainWindow", "Grid frequency"))
731     self.vrmsNum.setText(_translate("MainWindow", "Vrms"))
732     self.titleResultPage.setText(_translate("MainWindow", "Results"))
733     self.informationText.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC
    \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
734 "<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
735 "p, li { white-space: pre-wrap; }\n"
736 "</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:7.8pt; font-
    weight:400; font-style:normal;\n"
737 "<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
    block-indent:0; text-indent:0px;\n"
738 "<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0
    px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:8pt; color
    :#262c35;\n"
739 "<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
    block-indent:0; text-indent:0px;\n"
740 "<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0
    px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:11pt; color
    :#262c35;\n"
741 "<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
    block-indent:0; text-indent:0px;\n"
742 "<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0
    px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:11pt; color
    :#262c35;\n"
743 "<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
    block-indent:0; text-indent:0px;\n"
744 "<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0
    px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:11pt; color
    :#262c35;\n"
745 "<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
    block-indent:0; text-indent:0px;\n"
746     self.label_7.setText(_translate("MainWindow", "By: H. Yildiz, E. Bouma, M. de
    Vries, L. Damoiseaux, E. Öztürkoglu, J. Groeneveld"))
747
748
749     def update_thd_voltage(self):
750         #fetch the signal from the database -> this is the voltage signal actually from
        the voltage function
751         voltage_signal = self.voltage_signal()
752         detected_frequencies, detected_amplitudes = detect_harmonics(voltage_signal,
        20)
753         thd_percentage_voltage = total_harmonic_distortion(detected_amplitudes)
754         self.thdVoltageValue.setText(f"{thd_percentage_voltage:.2f}%")
755
756     def update_thd_current(self):

```

```

757     #fetch the signal from the database -> this is the voltage signal actually from
       the voltage function
758     current_signal = self.current_signal()
759     detected_frequencies, detected_amplitudes = detect_harmonics(current_signal,
20)
760     thd_percentage_current = total_harmonic_distortion(detected_amplitudes)
761     self.thdCurrentValue.setText(f"{thd_percentage_current:.2f}%")
762
763
764     def update_power_factor(self):
765         # Power Factor is calculated
766         voltage_signal = self.voltage_signal()
767         current_signal = self.current_signal()
768         estimated_freq = estimate_frequency_zero_crossing(voltage_signal)
769         pf = calculate_power_factor(voltage_signal, current_signal, estimated_freq)
770         self.powerFactorValue.setText(f"{pf:.2f}")
771
772     def update_vrm(self):
773         # VRMS is calculated
774         voltage_signal = self.voltage_signal()
775         estimated_freq = estimate_frequency_zero_crossing(voltage_signal)
776         voltage_magnitude_func = voltage_magnitude(voltage_signal, estimated_freq)
777         self.vrmValue.setText(f"{voltage_magnitude_func:.2f}")
778
779
780     def update_grid_freq(self):
781         # Grid frequency is detected
782         voltage_signal = self.voltage_signal()
783         estimated_freq = estimate_frequency_zero_crossing(voltage_signal)
784         self.gridFreqValue.setText(f"{estimated_freq:.5f}")
785
786     def update_voltage_sag_swell(self):
787         # Voltage sag and swell are detected
788         voltage_signal = self.voltage_signal()
789         estimated_freq = estimate_frequency_zero_crossing(voltage_signal)
790         sag = sag_check(voltage_signal, estimated_freq)
791         swell = swell_check(voltage_signal, estimated_freq)
792         self.voltageSagSwellValue.setText(f"Sag: {sag:.2f}, Swell: {swell:.2f}")
793
794 if __name__ == "__main__":
795     import sys
796
797     app = QtWidgets.QApplication(sys.argv)
798     MainWindow = QtWidgets.QMainWindow()
799     ui = Ui_MainWindow()
800     ui.setupUi(MainWindow)
801     MainWindow.show()
802     sys.exit(app.exec_())

```

H.6. GUI for Login Process

```

1  #####
2  # EE3L11 : Bachelor Graduation Project
3  # GROUP 1:
4  # SUBGROUP : SOFTWARE
5  # MEMBERS : Emirhan Öztürkoglu , Marten de Vries , Hüsna Yildiz
6  #
7  # Hüsna Yildiz
8  # DATE : 17/12/2024
9  # Description : This is the Graphical User Interface for the
10 # login process
11 #####
12 import os.path
13 import sys
14 from PyQt5 import QtWidgets
15 from PyQt5.QtWidgets import QDialog, QApplication
16 from PyQt5.uic import loadUi
17 from interface import Ui_MainWindow
18 from PyQt5.QtWidgets import QMainWindow
19 import pymysql
20 import random
21 import smtplib
22 from email.mime.text import MIMEText
23 from email.mime.multipart import MIMEMultipart
24
25
26
27 # host = "10.42.0.1"
28 host = '145.94.62.207'
29 user = "remote_software"
30 password = "remote_database"
31 database = "uss_psn"
32 connection = pymysql.connect(
33     host=host,
34     user=user,
35     password=password,
36     db=database
37 )
38 cursor = connection.cursor()
39
40 class Login(QDialog):
41     def __init__(self):
42         super(Login,self).__init__()
43         loadUi("login.ui",self)
44         self.loginbutton.clicked.connect(self.loginfunction)
45         self.password.setEchoMode(QtWidgets.QLineEdit.Password)
46         self.createaccbutton.clicked.connect(self.gotocreate)
47         self.forgotPswBtn.clicked.connect(self.open_forget_password_screen)
48
49     def loginfunction(self):
50         email = self.email.text()
51         password = self.password.text()
52
53         if self.verify_login(email, password, cursor):
54             verification = Verification(email)
55             widget.addWidget(verification)
56             widget.setCurrentIndex(widget.currentIndex() + 1)
57         else:
58             self.error.setText("Please input correct username and or password.")
59
60     def gotocreate(self):
61         createacc>CreateAcc()
62         widget.addWidget(createacc)
63         widget.setCurrentIndex(widget.currentIndex()+1)
64
65     def verify_login(self, EMAIL, password, cursor):
66         # Retrieve user information based on the provided username and password
67         select_query = "SELECT UserID FROM user_data WHERE EMAIL = %s AND password = %s"
68         cursor.execute(select_query, (EMAIL, password))

```

```

69
70     # Fetch the result
71     user_data = cursor.fetchone()
72
73     if user_data:
74         state=True
75         user_id = user_data[0]
76         print(f"Login successful. User ID: {user_id}")
77     else:
78         state=False
79         print("Incorrect email or password. Login failed.")
80     return state
81
82     def open_forget_password_screen(self):
83         forgetPassword = ForgotPasswordScreen()
84         widget.addWidget(forgetPassword)
85         widget.setCurrentIndex(widget.currentIndex() + 1)
86
87
88 class CreateAcc(QDialog):
89     def __init__(self):
90         super(CreateAcc,self).__init__()
91         loadUi("createacc.ui",self)
92         self.signupbutton.clicked.connect(self.createaccfunction)
93         self.goBackBtn.clicked.connect(self.go_back_to_login_screen)
94         self.password.setEchoMode(QtWidgets.QLineEdit.Password)
95         self.confirmpass.setEchoMode(QtWidgets.QLineEdit.Password)
96
97     def is_email_not_in_database(self, EMAIL, cursor):
98         # Check if the email already exists in the database
99         check_query = "SELECT UserID FROM user_data WHERE EMAIL = %s"
100         cursor.execute(check_query, (EMAIL,))
101
102         # Fetch the result
103         result = cursor.fetchone()
104
105         # Return True if the email is not in the database, False otherwise
106         return result is None
107
108     def insert_user_credentials(self, EMAIL, password, cursor, connection):
109         insert_query = "INSERT INTO user_data (EMAIL, password) VALUES (%s, %s)"
110         cursor.execute(insert_query, (EMAIL, password))
111         connection.commit()
112         print(f"User with email {EMAIL} successfully inserted")
113
114     def data_base_fill_in_email_passw(self, EMAIL, password):
115         EMAIL = self.email.text()
116         # Check whether email is not in the database, if not then insert into db
117         if self.is_email_not_in_database(EMAIL, cursor):
118             self.insert_user_credentials(EMAIL, password, cursor, connection)
119         # If it is in the db, then give error about that the account exist
120
121     def createaccfunction(self):
122         EMAIL = self.email.text()
123         if self.password.text()==self.confirmpass.text():
124             self.data_base_fill_in_email_passw(EMAIL, self.password.text())
125             password=self.password.text()
126             print("Successfully created acc with email: ", EMAIL, "and password: ",
password)
127             login=Login()
128             widget.addWidget(login)
129             widget.setCurrentIndex(widget.currentIndex()+1)
130         else:
131             self.error.setText("Passwords do not match")
132
133     def go_back_to_login_screen(self):
134         goback = Login()
135         # Show the interface dialog
136         widget.addWidget(goback)
137         widget.setCurrentIndex(widget.currentIndex() + 1)
138

```

```

139
140 class Verification(QDialog):
141     def __init__(self, email):
142         super(Verification, self).__init__()
143         loadUi("verificationScreen.ui", self)
144         # self.sendCodeBtn.clicked.connect(self.send_verification_code_function)
145         self.email = email
146         self.sendCodeBtn.clicked.connect(self.sending_mail)
147         self.verificationCode.setEchoMode(QtWidgets.QLineEdit.Password)
148         self.loginBtn.clicked.connect(self.open_interface)
149         self.generated_verification_code = None
150         self.interface_dialog = None # Store interface_dialog as an instance variable
151
152     def sending_mail(self):
153         # Enter verification code
154         self.generated_verification_code = random.randint(1000, 100000)
155         # Replace with your SMTP server details
156         smtp_server = 'smtp-relay.brevo.com'
157         smtp_port = 587 # Replace with the appropriate port
158
159         # Replace with your email credentials
160         smtp_username = 'bappowerquality@gmail.com'
161         smtp_password = '6yZvKIhUCkQA8pqg'
162
163         # Replace with your email content
164         from_email = 'bappowerquality@gmail.com'
165         to_email = self.email
166         subject = 'Verification code Power Quality Quantizer'
167         body = f'The verification code is: {self.generated_verification_code}'
168
169         # Create the MIMEText object for the email body
170         email_message = MIMEText(body, 'plain')
171         email_message['From'] = from_email
172         email_message['To'] = to_email
173         email_message['Subject'] = subject
174         email_message.attach(MIMEText(body, 'plain'))
175
176         try:
177             # Connect to the SMTP server
178             server = smtplib.SMTP(smtp_server, smtp_port)
179
180             # Start TLS for security
181             server.starttls()
182
183             # Log in to the SMTP server
184             server.login(smtp_username, smtp_password)
185
186             # Send the email
187             server.sendmail(from_email, to_email, email_message.as_string())
188
189             print("Email sent successfully.")
190         except Exception as e:
191             print(f"Error sending email: {str(e)}")
192         finally:
193             # Close the connection
194             server.quit()
195
196     # def send_verification_code_function(self):
197     #     self.generated_verification_code = random.randint(1000, 100000)
198     #     print(self.generated_verification_code)
199
200     def create_open_interface_page(self):
201         return (
202             self.generated_verification_code is not None
203             and int(self.verificationCode.text()) == self.generated_verification_code
204         )
205
206     def open_interface(self):
207         if self.create_open_interface_page():
208             print("You successfully entered the interface of PQQ safely!!")
209             # Instantiate the InterfaceDialog class and store it as an instance

```

```

variable
210     self.interface_dialog = InterfaceDialog()
211     # Show the interface dialog
212     self.interface_dialog.show()
213 else:
214     self.error.setText("Please input correct verification code")
215
216
217 class InterfaceDialog(QMainWindow):
218     def __init__(self):
219         super(InterfaceDialog, self).__init__()
220         self.ui = Ui_MainWindow()
221         self.ui.setupUi(self)
222
223
224 class ForgotPasswordScreen(QDialog):
225     def __init__(self):
226         super(ForgotPasswordScreen, self).__init__()
227         loadUi("forgotPassword.ui", self)
228         self.sendVerifCodeBtn.clicked.connect(self.send_verification_code)
229         # self.password.setEchoMode(QtWidgets.QLineEdit.Password)
230         # self.createaccbutton.clicked.connect(self.gotocreate)
231         # self.forgotPswBtn.clicked.connect(self.open_forget_password_screen)
232
233     def send_verification_code(self):
234         email = self.email.text()
235         if self.check_if_email_exist_in_db():
236             # password = self.password.text()
237             verification = Verification2(email)
238             widget.addWidget(verification)
239             widget.setCurrentIndex(widget.currentIndex() + 1)
240             # send_email_verif = Verification(self.email)
241             # send_email_verif.sending_mail()
242             # if send_email_verif.create_open_interface_page():
243
244     def check_if_email_exist_in_db(self):
245         email = self.email.text()
246         # Check whether email is in the database
247         create_acc_instance = CreateAcc()
248         if not create_acc_instance.is_email_not_in_database(email, cursor):
249             print("email is found in the database")
250             return True
251         else:
252             print("email not found")
253             return False
254
255
256 class Verification2(QDialog):
257     def __init__(self, email):
258         super(Verification2, self).__init__()
259         loadUi("verificationScreenForgotPassword.ui", self)
260         # self.sendCodeBtn.clicked.connect(self.send_verification_code_function)
261         self.email = email
262         self.sendCodeBtn.clicked.connect(self.sending_mail)
263         self.verificationCode.setEchoMode(QtWidgets.QLineEdit.Password)
264         self.loginBtn.clicked.connect(self.open_password_screen)
265         self.generated_verification_code = None
266         self.interface_dialog = None # Store interface_dialog as an instance variable
267
268     def sending_mail(self):
269         # Enter verification code
270         self.generated_verification_code = random.randint(1000, 100000)
271         # Replace with your SMTP server details
272         smtp_server = 'smtp-relay.brevo.com'
273         smtp_port = 587 # Replace with the appropriate port
274
275         # Replace with your email credentials
276         smtp_username = 'bappowerquality@gmail.com'
277         smtp_password = '6yZvKIhUCkQA8pqg'
278
279         # Replace with your email content

```

```

280     from_email = 'bappowerquality@gmail.com'
281     to_email = self.email
282     subject = 'Verification code Power Quality Quantizer'
283     body = f'The verification code is: {self.generated_verification_code}'
284
285     # Create the MIMEText object for the email body
286     email_message = MIMEMultipart()
287     email_message['From'] = from_email
288     email_message['To'] = to_email
289     email_message['Subject'] = subject
290     email_message.attach(MIMEText(body, 'plain'))
291
292     try:
293         # Connect to the SMTP server
294         server = smtplib.SMTP(smtp_server, smtp_port)
295
296         # Start TLS for security
297         server.starttls()
298
299         # Log in to the SMTP server
300         server.login(smtp_username, smtp_password)
301
302         # Send the email
303         server.sendmail(from_email, to_email, email_message.as_string())
304
305         print("Email sent successfully.")
306     except Exception as e:
307         print(f"Error sending email: {str(e)}")
308     finally:
309         # Close the connection
310         server.quit()
311
312     def open_password_screen(self):
313         save_password_screen = SavePassword(self.email)
314         widget.addWidget(save_password_screen)
315         widget.setCurrentIndex(widget.currentIndex() + 1)
316
317 class SavePassword(QDialog):
318     def __init__(self, email):
319         super(SavePassword, self).__init__()
320         loadUi("enterpassw.ui", self)
321         self.email = email
322         self.signupbutton.clicked.connect(self.update_in_db)
323         self.goBackBtn.clicked.connect(self.go_back_to_login_screen)
324         self.password.setEchoMode(QtWidgets.QLineEdit.Password)
325         self.confirmpass.setEchoMode(QtWidgets.QLineEdit.Password)
326
327     def update_in_db(self):
328         """If the confirm_passw == passw, then update in the db"""
329         if self.password.text() == self.confirmpass.text():
330             self.update_password_in_database(self.password.text(), cursor)
331             password = self.password.text()
332             print("Successfully updated acc with email: ", self.email, "and password: ",
333                   password)
334             login = Login()
335             widget.addWidget(login)
336             widget.setCurrentIndex(widget.currentIndex() + 1)
337         else:
338             self.error.setText("Passwords do not match")
339
340     def go_back_to_login_screen(self):
341         goback = Login()
342         widget.addWidget(goback)
343         widget.setCurrentIndex(widget.currentIndex() + 1)
344
345     def update_password_in_database(self, password, cursor):
346         # query = f"UPDATE password FROM user_data WHERE EMAIL == '{self.email}'"
347         # user_data = our table name??
348         query = "UPDATE user_data SET password = %s WHERE EMAIL = %s"

```

```
350         cursor.execute(query, (password, self.email))
351         connection.commit()
352
353 app=QApplication(sys.argv)
354 mainwindow=Login()
355 widget=QtWidgets.QStackedWidget()
356 widget.addWidget(mainwindow)
357 widget.setFixedWidth(480)
358 widget.setFixedHeight(620)
359 widget.show()
360 app.exec_()
```

H.7. Logging in GUI

```

1  #####
2  # EE3L11 : Bachelor Graduation Project
3  # GROUP 1:
4  # SUBGROUP : SOFTWARE
5  # MEMBERS : Emirhan Öztürkoglu , Marten de Vries , Hüsna Yildiz
6  #
7  # Hüsna Yildiz
8  # DATE : 1/12/2023
9  # Description : This is the page where the user logs in
10 #####
11
12 from PyQt5 import QtCore, QtGui, QtWidgets
13
14
15 class Ui_Dialog(object):
16     def setupUi(self, Dialog):
17         Dialog.setObjectName("Dialog")
18         Dialog.resize(480, 620)
19         Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
20         self.label = QtWidgets.QLabel(Dialog)
21         self.label.setGeometry(QtCore.QRect(300, 50, 121, 71))
22         self.label.setStyleSheet("font-size:28pt; \n"
23 "color: rgb(216, 216, 216);")
24         self.label.setObjectName("label")
25         self.label_2 = QtWidgets.QLabel(Dialog)
26         self.label_2.setGeometry(QtCore.QRect(40, 260, 101, 31))
27         self.label_2.setStyleSheet("font-size:15pt; \n"
28 "color: rgb(216, 216, 216);")
29         self.label_2.setObjectName("label_2")
30         self.email = QtWidgets.QLineEdit(Dialog)
31         self.email.setGeometry(QtCore.QRect(170, 260, 241, 51))
32         self.email.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
33         self.email.setObjectName("email")
34         self.password = QtWidgets.QLineEdit(Dialog)
35         self.password.setGeometry(QtCore.QRect(170, 360, 241, 51))
36         self.password.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
37         self.password.setText("")
38         self.password.setObjectName("password")
39         self.label_3 = QtWidgets.QLabel(Dialog)
40         self.label_3.setGeometry(QtCore.QRect(40, 370, 111, 31))
41         self.label_3.setStyleSheet("font-size:15pt; \n"
42 "color: rgb(216, 216, 216);")
43         self.label_3.setObjectName("label_3")
44         self.loginbutton = QtWidgets.QPushButton(Dialog)
45         self.loginbutton.setGeometry(QtCore.QRect(270, 490, 141, 41))
46         self.loginbutton.setStyleSheet("background-color: rgb(116, 117, 116); font-size
:14pt; color:rgb(255, 255, 255)")
47         self.loginbutton.setObjectName("loginbutton")
48         self.createaccbutton = QtWidgets.QPushButton(Dialog)
49         self.createaccbutton.setGeometry(QtCore.QRect(22, 550, 391, 28))
50         self.createaccbutton.setStyleSheet("color:rgb(255, 255, 255);\n"
51 "text-decoration: underline;\n"
52 "font: 8pt \"MS Shell Dlg 2\";\n"
53 "border: none;\n"
54 "background-color: transparent;\n"
55 "background: transparent;\n"
56 "padding: 0;\n"
57 "margin: 0;\n"
58 "color: #fff;")
59         self.createaccbutton.setObjectName("createaccbutton")
60         self.label_5 = QtWidgets.QLabel(Dialog)
61         self.label_5.setGeometry(QtCore.QRect(10, 0, 200, 200))
62         self.label_5.setText("")
63         self.label_5.setPixmap(QtGui.QPixmap("logo.png"))
64         self.label_5.setScaledContents(True)
65         self.label_5.setObjectName("label_5")
66         self.error = QtWidgets.QLabel(Dialog)
67         self.error.setGeometry(QtCore.QRect(40, 460, 241, 20))
68         self.error.setStyleSheet("color: rgb(255, 0, 0)")

```

```

69         self.error.setText("")
70         self.error.setObjectName("error")
71         self.forgotPswBtn = QtWidgets.QPushButton(Dialog)
72         self.forgotPswBtn.setGeometry(QtCore.QRect(260, 420, 151, 28))
73         self.forgotPswBtn.setStyleSheet("color:rgb(255, 255, 255);\n"
74 "text-decoration: underline;\n"
75 "font: 8pt \"MS Shell Dlg 2\";\n"
76 "border: none;\n"
77 "background-color: transparent;\n"
78 "background: transparent;\n"
79 "padding: 0;\n"
80 "margin: 0;\n"
81 "color: #fff;")
82         self.forgotPswBtn.setObjectName("forgotPswBtn")
83         self.label_5.raise_()
84         self.label.raise_()
85         self.label_2.raise_()
86         self.email.raise_()
87         self.password.raise_()
88         self.label_3.raise_()
89         self.loginbutton.raise_()
90         self.createaccbutton.raise_()
91         self.error.raise_()
92         self.forgotPswBtn.raise_()
93
94         self.retranslateUi(Dialog)
95         QtCore.QMetaObject.connectSlotsByName(Dialog)
96
97     def retranslateUi(self, Dialog):
98         _translate = QtCore.QCoreApplication.translate
99         Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
100        self.label.setText(_translate("Dialog", "Login"))
101        self.label_2.setText(_translate("Dialog", "Email"))
102        self.label_3.setText(_translate("Dialog", "Password"))
103        self.loginbutton.setText(_translate("Dialog", "Login"))
104        self.createaccbutton.setText(_translate("Dialog", "Don't have an account?
Create Account"))
105        self.forgotPswBtn.setText(_translate("Dialog", "Forgot Password?"))

```

H.8. Account creation GUI

```

1  #####
2  # EE3L11 : Bachelor Graduation Project
3  # GROUP 1:
4  # SUBGROUP : SOFTWARE
5  # MEMBERS : Emirhan Öztürkoglu , Marten de Vries , Hüsna Yildiz
6  #
7  # Hüsna Yildiz
8  # DATE : 1/12/2023
9  # Description : This is the page where the user makes an account
10 #####
11 from PyQt5 import QtCore, QtGui, QtWidgets
12
13
14 class Ui_Dialog(object):
15     def setupUi(self, Dialog):
16         Dialog.setObjectName("Dialog")
17         Dialog.resize(480, 620)
18         Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
19         self.label = QtWidgets.QLabel(Dialog)
20         self.label.setGeometry(QtCore.QRect(250, 50, 171, 71))
21         self.label.setStyleSheet("font-size:28pt;\n"
22 "color: rgb(216, 216, 216);")
23         self.label.setObjectName("label")
24         self.label_2 = QtWidgets.QLabel(Dialog)
25         self.label_2.setGeometry(QtCore.QRect(40, 250, 101, 31))
26         self.label_2.setStyleSheet("font-size:15pt; color: rgb(216, 216, 216);")
27         self.label_2.setObjectName("label_2")
28         self.email = QtWidgets.QLineEdit(Dialog)
29         self.email.setGeometry(QtCore.QRect(200, 240, 211, 51))

```

```

30     self.email.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
31     self.email.setObjectName("email")
32     self.password = QtWidgets.QLineEdit(Dialog)
33     self.password.setGeometry(QtCore.QRect(200, 320, 211, 51))
34     self.password.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
35     self.password.setText("")
36     self.password.setObjectName("password")
37     self.label_3 = QtWidgets.QLabel(Dialog)
38     self.label_3.setGeometry(QtCore.QRect(40, 330, 111, 31))
39     self.label_3.setStyleSheet("font-size:15pt; color: rgb(216, 216, 216);")
40     self.label_3.setObjectName("label_3")
41     self.signupbutton = QtWidgets.QPushButton(Dialog)
42     self.signupbutton.setGeometry(QtCore.QRect(270, 490, 141, 41))
43     self.signupbutton.setStyleSheet("background-color: rgb(116, 117, 116); font-
size:14pt; color:rgb(255, 255, 255)")
44     self.signupbutton.setObjectName("signupbutton")
45     self.confirmpass = QtWidgets.QLineEdit(Dialog)
46     self.confirmpass.setGeometry(QtCore.QRect(200, 400, 211, 51))
47     self.confirmpass.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
48     self.confirmpass.setText("")
49     self.confirmpass.setObjectName("confirmpass")
50     self.label_4 = QtWidgets.QLabel(Dialog)
51     self.label_4.setGeometry(QtCore.QRect(40, 410, 151, 31))
52     self.label_4.setStyleSheet("font-size:15pt; color: rgb(216, 216, 216);")
53     self.label_4.setObjectName("label_4")
54     self.label_5 = QtWidgets.QLabel(Dialog)
55     self.label_5.setGeometry(QtCore.QRect(10, 0, 200, 200))
56     self.label_5.setText("")
57     self.label_5.setPixmap(QtGui.QPixmap("logo.png"))
58     self.label_5.setScaledContents(True)
59     self.label_5.setObjectName("label_5")
60     self.error = QtWidgets.QLabel(Dialog)
61     self.error.setGeometry(QtCore.QRect(40, 470, 391, 16))
62     self.error.setStyleSheet("color: rgb(255, 0, 0)")
63     self.error.setText("")
64     self.error.setObjectName("error")
65     self.goBackBtn = QtWidgets.QPushButton(Dialog)
66     self.goBackBtn.setGeometry(QtCore.QRect(330, 160, 121, 28))
67     self.goBackBtn.setStyleSheet("color: rgb(255, 255, 255);\n"
68 "border: none;\n"
69 "background-color: transparent;\n"
70 "background: transparent;\n"
71 "padding: 0;\n"
72 "margin: 0;\n"
73 "color: #fff;\n"
74 "")
75     self.goBackBtn.setObjectName("goBackBtn")
76     self.label_5.raise_()
77     self.label.raise_()
78     self.label_2.raise_()
79     self.email.raise_()
80     self.password.raise_()
81     self.label_3.raise_()
82     self.signupbutton.raise_()
83     self.confirmpass.raise_()
84     self.label_4.raise_()
85     self.error.raise_()
86     self.goBackBtn.raise_()
87
88     self.retranslateUi(Dialog)
89     QtCore.QMetaObject.connectSlotsByName(Dialog)
90
91     def retranslateUi(self, Dialog):
92         _translate = QtCore.QCoreApplication.translate
93         Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
94         self.label.setText(_translate("Dialog", "Sign up"))
95         self.label_2.setText(_translate("Dialog", "Email"))
96         self.label_3.setText(_translate("Dialog", "Password"))
97         self.signupbutton.setText(_translate("Dialog", "Sign up"))
98         self.label_4.setText(_translate("Dialog", "Confirm Pass"))
99         self.goBackBtn.setText(_translate("Dialog", "Go back to login"))

```

H.9. Verification GUI before opening Interface that monitors power quality

```

1  #####
2  # EE3L11 : Bachelor Graduation Project
3  # GROUP 1:
4  # SUBGROUP : SOFTWARE
5  # MEMBERS : Emirhan Öztürkoglu , Marten de Vries , Hüsna Yildiz
6  #
7  # Hüsna Yildiz
8  # DATE : 1/12/2023
9  # Description : This is the page where the user enters his verification code for
10 # entering the interface
11 #####
12
13 from PyQt5 import QtCore, QtGui, QtWidgets
14
15
16 class Ui_Dialog(object):
17     def setupUi(self, Dialog):
18         Dialog.setObjectName("Dialog")
19         Dialog.resize(480, 620)
20         Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
21         self.verificatiOnCode = QtWidgets.QLineEdit(Dialog)
22         self.verificatiOnCode.setGeometry(QtCore.QRect(260, 320, 211, 51))
23         self.verificatiOnCode.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
24         self.verificatiOnCode.setText("")
25         self.verificatiOnCode.setObjectName("verificatiOnCode")
26         self.label_3 = QtWidgets.QLabel(Dialog)
27         self.label_3.setGeometry(QtCore.QRect(0, 320, 251, 41))
28         self.label_3.setStyleSheet("font-size:15pt; color: rgb(216, 216, 216);")
29         self.label_3.setObjectName("label_3")
30         self.loginBtn = QtWidgets.QPushButton(Dialog)
31         self.loginBtn.setGeometry(QtCore.QRect(330, 440, 141, 41))
32         self.loginBtn.setStyleSheet("background-color: rgb(167, 168, 167); font-size:14
33 pt; color:rgb(255, 255, 255)")
34         self.loginBtn.setObjectName("loginBtn")
35         self.label_5 = QtWidgets.QLabel(Dialog)
36         self.label_5.setGeometry(QtCore.QRect(10, 0, 200, 200))
37         self.label_5.setText("")
38         self.label_5.setPixmap(QtGui.QPixmap("logo.png"))
39         self.label_5.setScaledContents(True)
40         self.label_5.setObjectName("label_5")
41         self.sendCodeBtn = QtWidgets.QPushButton(Dialog)
42         self.sendCodeBtn.setGeometry(QtCore.QRect(370, 260, 93, 28))
43         self.sendCodeBtn.setStyleSheet("color:rgb(255, 255, 255)")
44         self.sendCodeBtn.setObjectName("sendCodeBtn")
45         self.error = QtWidgets.QLabel(Dialog)
46         self.error.setGeometry(QtCore.QRect(30, 380, 351, 16))
47         self.error.setStyleSheet("color: rgb(255, 0, 0)")
48         self.error.setText("")
49         self.error.setObjectName("error")
50         self.label = QtWidgets.QLabel(Dialog)
51         self.label.setGeometry(QtCore.QRect(30, 410, 411, 16))
52         self.label.setStyleSheet("color: rgb(255, 255, 255)")
53         self.label.setObjectName("label")
54         self.label_3.raise_()
55         self.label_5.raise_()
56         self.verificatiOnCode.raise_()
57         self.loginBtn.raise_()
58         self.sendCodeBtn.raise_()
59         self.error.raise_()
60         self.label.raise_()
61
62         self.retranslateUi(Dialog)
63         QtCore.QMetaObject.connectSlotsByName(Dialog)
64
65     def retranslateUi(self, Dialog):
66         _translate = QtCore.QCoreApplication.translate
67         Dialog.setWindowTitle(_translate("Dialog", "Dialog"))

```

```

67     self.label_3.setText(_translate("Dialog", "Enter verification code"))
68     self.loginBtn.setText(_translate("Dialog", "Login"))
69     self.sendCodeBtn.setText(_translate("Dialog", "send code"))
70     self.label.setText(_translate("Dialog", "It can take up to 5 minutes, please
make sure to check your spam mail"))

```

H.10. Verification GUI before changing password

```

1  #####
2  # EE3L11 : Bachelor Graduation Project
3  # GROUP 1:
4  # SUBGROUP : SOFTWARE
5  # MEMBERS : Emirhan Öztürkoglu , Marten de Vries , Hüsna Yildiz
6  #
7  # Hüsna Yildiz
8  # DATE : 1/12/2023
9  # Description : This is the page where the user enters his verification code before he
is allowed to change his password
10 #####
11 from PyQt5 import QtCore, QtGui, QtWidgets
12
13
14 class Ui_Dialog(object):
15     def setupUi(self, Dialog):
16         Dialog.setObjectName("Dialog")
17         Dialog.resize(480, 620)
18         Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
19         self.verificationCode = QtWidgets.QLineEdit(Dialog)
20         self.verificationCode.setGeometry(QtCore.QRect(260, 320, 211, 51))
21         self.verificationCode.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
22         self.verificationCode.setText("")
23         self.verificationCode.setObjectName("verificationCode")
24         self.label_3 = QtWidgets.QLabel(Dialog)
25         self.label_3.setGeometry(QtCore.QRect(0, 320, 251, 41))
26         self.label_3.setStyleSheet("font-size:15pt; color: rgb(216, 216, 216);")
27         self.label_3.setObjectName("label_3")
28         self.loginBtn = QtWidgets.QPushButton(Dialog)
29         self.loginBtn.setGeometry(QtCore.QRect(210, 440, 261, 41))
30         self.loginBtn.setStyleSheet("font-size:14pt; color:rgb(255, 255, 255);\n"
"background-color: rgb(116, 117, 116);")
31         self.loginBtn.setObjectName("loginBtn")
32         self.label_5 = QtWidgets.QLabel(Dialog)
33         self.label_5.setGeometry(QtCore.QRect(10, 0, 200, 200))
34         self.label_5.setText("")
35         self.label_5.setPixmap(QtGui.QPixmap("logo.png"))
36         self.label_5.setScaledContents(True)
37         self.label_5.setObjectName("label_5")
38         self.sendCodeBtn = QtWidgets.QPushButton(Dialog)
39         self.sendCodeBtn.setGeometry(QtCore.QRect(370, 260, 93, 28))
40         self.sendCodeBtn.setStyleSheet("color:rgb(255, 255, 255)")
41         self.sendCodeBtn.setObjectName("sendCodeBtn")
42         self.error = QtWidgets.QLabel(Dialog)
43         self.error.setGeometry(QtCore.QRect(30, 380, 351, 16))
44         self.error.setStyleSheet("color: rgb(255, 0, 0)")
45         self.error.setText("")
46         self.error.setObjectName("error")
47         self.label = QtWidgets.QLabel(Dialog)
48         self.label.setGeometry(QtCore.QRect(30, 410, 411, 16))
49         self.label.setStyleSheet("color: rgb(255, 255, 255)")
50         self.label.setObjectName("label")
51         self.label_3.raise_()
52         self.label_5.raise_()
53         self.verificationCode.raise_()
54         self.loginBtn.raise_()
55         self.sendCodeBtn.raise_()
56         self.error.raise_()
57         self.label.raise_()
58
59
60     self.retranslateUi(Dialog)
61     QtCore.QMetaObject.connectSlotsByName(Dialog)

```

```

62
63     def retranslateUi(self, Dialog):
64         _translate = QtCore.QCoreApplication.translate
65         Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
66         self.label_3.setText(_translate("Dialog", "Enter verification code"))
67         self.loginBtn.setText(_translate("Dialog", "Change password"))
68         self.sendCodeBtn.setText(_translate("Dialog", "send code"))
69         self.label.setText(_translate("Dialog", "It can take up to 5 minutes, please
make sure to check your spam mail"))

```

H.11. Enter email GUI before changing password

```

1  #####
2  # EE3L11 : Bachelor Graduation Project
3  # GROUP 1:
4  # SUBGROUP : SOFTWARE
5  # MEMBERS : Emirhan Öztürkoglu , Marten de Vries , Hüsna Yildiz
6  #
7  # Hüsna Yildiz
8  # DATE : 12/12/2023
9  # Description : This is the page where the user enters his email before changing his
password
10 #####
11 from PyQt5 import QtCore, QtGui, QtWidgets
12
13
14 class Ui_Dialog(object):
15     def setupUi(self, Dialog):
16         Dialog.setObjectName("Dialog")
17         Dialog.resize(480, 620)
18         Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
19         self.emailBtn = QtWidgets.QLabel(Dialog)
20         self.emailBtn.setGeometry(QtCore.QRect(40, 320, 61, 31))
21         self.emailBtn.setStyleSheet("font-size:15pt; color: rgb(216, 216, 216);")
22         self.emailBtn.setObjectName("emailBtn")
23         self.email = QtWidgets.QLineEdit(Dialog)
24         self.email.setGeometry(QtCore.QRect(250, 310, 211, 51))
25         self.email.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
26         self.email.setObjectName("email")
27         self.sendVerifCodeBtn = QtWidgets.QPushButton(Dialog)
28         self.sendVerifCodeBtn.setGeometry(QtCore.QRect(160, 500, 301, 41))
29         self.sendVerifCodeBtn.setStyleSheet("background-color: rgb(116, 117, 116); font
-size:14pt; color:rgb(255, 255, 255)")
30         self.sendVerifCodeBtn.setObjectName("sendVerifCodeBtn")
31         self.label_5 = QtWidgets.QLabel(Dialog)
32         self.label_5.setGeometry(QtCore.QRect(10, 0, 200, 200))
33         self.label_5.setText("")
34         self.label_5.setPixmap(QtGui.QPixmap("logo.png"))
35         self.label_5.setScaledContents(True)
36         self.label_5.setObjectName("label_5")
37         self.error = QtWidgets.QLabel(Dialog)
38         self.error.setGeometry(QtCore.QRect(40, 380, 391, 16))
39         self.error.setStyleSheet("color: rgb(255, 0, 0)")
40         self.error.setText("")
41         self.error.setObjectName("error")
42         self.goBackBtn = QtWidgets.QPushButton(Dialog)
43         self.goBackBtn.setGeometry(QtCore.QRect(330, 160, 121, 28))
44         self.goBackBtn.setStyleSheet("color: rgb(255, 255, 255);\n"
"font: 75 8pt \\"MS Shell Dlg 2\\";\n"
"border: none;\n"
"background-color: transparent;\n"
"background: transparent;\n"
"padding: 0;\n"
"margin: 0;\n"
"color: #fff;\n"
"")
45         self.goBackBtn.setObjectName("goBackBtn")
46         self.label_5.raise_()
47         self.emailBtn.raise_()
48         self.error.raise_()
49         self.goBackBtn.raise_()

```

```

57         self.sendVerifCodeBtn.raise_()
58         self.error.raise_()
59         self.goBackBtn.raise_()
60
61         self.retranslateUi(Dialog)
62         QtCore.QMetaObject.connectSlotsByName(Dialog)
63
64     def retranslateUi(self, Dialog):
65         _translate = QtCore.QCoreApplication.translate
66         Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
67         self.emailBtn.setText(_translate("Dialog", "Email"))
68         self.sendVerifCodeBtn.setText(_translate("Dialog", "Send verification code"))
69         self.goBackBtn.setText(_translate("Dialog", "Go back to login"))

```

H.12. User changes his password on this GUI

```

1  #####
2  # EE3L11 : Bachelor Graduation Project
3  # GROUP 1:
4  # SUBGROUP : SOFTWARE
5  # MEMBERS : Emirhan Öztürkoglu , Marten de Vries , Hüsna Yildiz
6  #
7  # Hüsna Yildiz
8  # DATE : 1/12/2023
9  # Description : The user changes his password on this page
10 #####
11 from PyQt5 import QtCore, QtGui, QtWidgets
12
13
14 class Ui_Dialog(object):
15     def setupUi(self, Dialog):
16         Dialog.setObjectName("Dialog")
17         Dialog.resize(480, 620)
18         Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
19         self.password = QtWidgets.QLineEdit(Dialog)
20         self.password.setGeometry(QtCore.QRect(200, 320, 211, 51))
21         self.password.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
22         self.password.setText("")
23         self.password.setObjectName("password")
24         self.label_3 = QtWidgets.QLabel(Dialog)
25         self.label_3.setGeometry(QtCore.QRect(40, 330, 111, 31))
26         self.label_3.setStyleSheet("font-size:15pt; color: rgb(216, 216, 216);")
27         self.label_3.setObjectName("label_3")
28         self.signupbutton = QtWidgets.QPushButton(Dialog)
29         self.signupbutton.setGeometry(QtCore.QRect(270, 490, 141, 41))
30         self.signupbutton.setStyleSheet("background-color: rgb(116, 117, 116); font-size:14pt; color:rgb(255, 255, 255)")
31         self.signupbutton.setObjectName("signupbutton")
32         self.confirmpass = QtWidgets.QLineEdit(Dialog)
33         self.confirmpass.setGeometry(QtCore.QRect(200, 400, 211, 51))
34         self.confirmpass.setStyleSheet("font-size:14pt; color:rgb(243, 243, 243)")
35         self.confirmpass.setText("")
36         self.confirmpass.setObjectName("confirmpass")
37         self.label_4 = QtWidgets.QLabel(Dialog)
38         self.label_4.setGeometry(QtCore.QRect(40, 410, 151, 31))
39         self.label_4.setStyleSheet("font-size:15pt; color: rgb(216, 216, 216);")
40         self.label_4.setObjectName("label_4")
41         self.label_5 = QtWidgets.QLabel(Dialog)
42         self.label_5.setGeometry(QtCore.QRect(10, 0, 200, 200))
43         self.label_5.setText("")
44         self.label_5.setPixmap(QtGui.QPixmap("logo.png"))
45         self.label_5.setScaledContents(True)
46         self.label_5.setObjectName("label_5")
47         self.error = QtWidgets.QLabel(Dialog)
48         self.error.setGeometry(QtCore.QRect(40, 470, 391, 16))
49         self.error.setStyleSheet("color: rgb(255, 0, 0)")
50         self.error.setText("")
51         self.error.setObjectName("error")
52         self.goBackBtn = QtWidgets.QPushButton(Dialog)
53         self.goBackBtn.setGeometry(QtCore.QRect(330, 160, 121, 28))

```

```
54         self.goBackBtn.setStyleSheet("color: rgb(255, 255, 255);\n"
55 "border: none;\n"
56 "background-color: transparent;\n"
57 "background: transparent;\n"
58 "padding: 0;\n"
59 "margin: 0;\n"
60 "color: #fff;\n"
61 "")
62         self.goBackBtn.setObjectName("goBackBtn")
63         self.label_5.raise_()
64         self.password.raise_()
65         self.label_3.raise_()
66         self.signupbutton.raise_()
67         self.confirmpass.raise_()
68         self.label_4.raise_()
69         self.error.raise_()
70         self.goBackBtn.raise_()
71
72         self.retranslateUi(Dialog)
73         QtCore.QMetaObject.connectSlotsByName(Dialog)
74
75     def retranslateUi(self, Dialog):
76         _translate = QtCore.QCoreApplication.translate
77         Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
78         self.label_3.setText(_translate("Dialog", "Password"))
79         self.signupbutton.setText(_translate("Dialog", "Save"))
80         self.label_4.setText(_translate("Dialog", "Confirm Pass"))
81         self.goBackBtn.setText(_translate("Dialog", "Go back to login"))
```