Anomaly Detection with Industrial PC-Based Energy Measurement Framework

Faculty of Computer Science and Engineering Science
Cologne University of Applied Sciences

Submitted by:

Bhagyashree Popat Zaware	(11160161)
Eduardo Guandulain Juárez	(11158980)
Muhammad Munam Uddin Farooqui	(11160281)
Nnamdi Ogbonnaya Odii	(11160333)
Rohit Arunachalam Gopinath	(11158214)
Yuganshu Wadhwa	(11158887)

Supervisors:
Prof. Michael Freiburg
Prof. Felix Hackelöer

Gummersbach, 09.07.2024

Technology
Arts Sciences
TH Köln

Abstract

In the following paper, a thorough investigation into various anomalies within power systems was conducted. An experiment using a BECKHOFF Industrial PC running TwinCAT 3, equipped with measurement I/O modules capable of capturing high-frequency data was set up. Voltages and currents were measured from various sources, including electrical servo drives and signal generators. Based on these measurements, an attempt was made to analyze the voltage and current signals in real-time. As a following step, different algorithms were developed to recognize features or characteristics representing anomalous behavior.

Contents

List of Tables

List of Figures

1 Introduction

Electric energy plays a vital role in today's society, driving social and economic development in nations. As demand for this essential resource continues to surge, so does the complexity of the systems responsible for delivering it. The continuous structural, operational, and administrative changes are taking place in the power sector in these modern times due to digitalization and liberalization of the energy market, which is a major cause of complicatedness currently being experienced in the industry [1]. The adoption of advanced control systems, smart meters, digital twins, smart grids, IoT, Cloud solutions and distributed energy resources (DERs) are giving rise to some anomalous behaviors in the generation, transmission, and distribution networks [2]. To deploy these innovative technologies to build smart and robust electricity grids that achieve the desired efficiency, security and reliability, the development of methods that can detect and handling the inherent uncertainties in real-time is imperative [3].

In the context of power system networks, any departure or abnormality from the expected or typical behavior within the system is referred to as an "anomaly". Unusual power consumption patterns, fake data injection attacks, communication system failures, abrupt changes in load or generation, and huge measurement errors in voltages and currents are but a few examples of how these anomalies can appear [4]. Power system anomalies could be pointers to foreseeable problems, dangers, or inefficiencies that should be quickly identified and fixed to maintain the grid's dependability, effectiveness, and security. These anomalies, electrical faults or outliers can be caused by numerous factors such as equipment damage, environmental and/or human factors [5]. Whatever the cause is, it is essential to identify and categorize these anomalies to preserve steady operations, avoid disturbances, and protect against cyberattacks or equipment failures [6].

For the sake of modeling, these anomalies can be identified as power failures, voltage and frequency variations, harmonic distortions, transients and surges, voltage unbalance, and power quality issues. These faults can disrupt essential services, industrial processes, and daily life, resulting in significant economic losses, equipment damage and death of humans. Real-time identification and mitigation of these anomalies are therefore essential to minimize downtime, prevent equipment damage, and maintain uninterrupted power supply to consumers. Addressing these challenges is pivotal for advancing the state-of-the-art in power system monitoring, control, and protection.

Considering that anomaly detection in power systems could improve system efficiency and reliability, it has attracted a lot of attention lately. Previous studies have concentrated on several anomaly detection-related topics, including harmonic distortion, transient events, voltage imbalance, power quality problems, and fluctuations in voltage and frequency. The use of machine learning techniques for anomaly detection in power systems has been the subject of several studies. For instance, a support vector machine (SVM)-based technique to identify voltage disruptions in distribution networks was presented by Baghaee et al [7]. The authors of [8] and [9] employed deep learning algorithms to detect power quality faults in smart grids in real-time.

Additionally, hardware and software systems created especially for energy systems monitoring as well as evaluation have been the focus of research efforts. For example, the TwinCAT 3 SoftPLC- equipped BECKHOFF Industrial PC provides a stable and adaptable framework for real-time data processing and acquisition in industrial settings. Through the integration of sophisticated analytics capabilities with high-frequency measurement modules, these platforms provide the early detection and remediation of anomalies within power systems.

Even with these developments, successful anomaly detection in industrial power systems still requires sophisticated frameworks that smoothly combine hardware and software components. This study fills this gap by putting forth a novel framework that combines the powerful analytics techniques offered by Beckhoff's TwinCAT 3 SoftPLC platform with machine learning algorithm in real-time. By offering a useful method for real-time anomaly identification in industrial power systems, this study adds to the body of information already in existence. Through the utilization of innovative technologies and techniques, the suggested framework provides an invaluable instrument for enhancing the dependability and effectiveness of industrial power systems.

The rest of this study is organized in the following sections. Section 2 provides a summary of the related works we reviewed during the process. Section 3 outlines requirements engineering for the entire research project. The experimental setup, including the data acquisition, hardware setup and TwinCAT 3 setup are detailed in section 4. In section 5, various power system anomalies of interest are highlighted. Different communication approaches and comparative analysis between various communication approaches are discussed in section 6. Finally, the anomaly detection algorithm is presented in section 7.

2 State of the Art and Literature Review

In recent years, there has been a significantly growing interest in the application of advanced technologies such as big data analytics, machine learning, and artificial intelligence to Anomaly detection in power systems. These technologies can be used to process the vast amounts of data generated by power plants and to extract valuable insights that can help improve the performance and reliability of power systems. In the following table, it is possible to visualize several studies that have investigated the application of these technologies to Anomaly detection in power systems.

Authors	Title	Main Findings
Ugur Halden, Umit Cali	Anomaly Detection in Power Markets and Systems [10]	Investigation and discussion of the methodologies and strategies used to detect abnormalities in power systems and markets, using various methods such as AI-based advanced analytics.
C. Chahla, H. Snoussi, L. Merghem, M. Esseghir	A deep learning approach for anomaly detection and prediction in power consumption data [11]	A deep learning approach for anomaly detection and prediction in power consumption data. A combination of clustering-based methods with the prediction-based ones to learn typical behavior scenarios and to predict the power consumption of the next hour.
Xinlin Wanga, Sung-Hoon Ahn	Real-time prediction and anomaly detection of electrical load in a residential community [12]	Presentation of a new anomaly detection framework featuring a one-step-ahead load predictor and a rule-engine-based load anomaly detector. The predictor combines various models to mitigate over or under-fitting issues in real-time pre- diction. Meanwhile, the anomaly detector operates independently, referencing normal power usage patterns to identify anomalies like electrical theft or leakage.
Rizvi, Syed Muhammad Hur et al.	Data-driven short-term voltage stability assessment using convolutional neural networks considering data anomalies and localization [13]	The paper proposes a time-series deep learning framework using 1D-CNN for real-time monitoring of short-term voltage stability, incorporating fast voltage collapse detection and severity quantification.
Ali, Mairaj, et al.	Detecting and monitoring of voltage and frequency variation and underground cable fault location using check point method [14]	The authors proposed a method for monitoring voltage, phase, and frequency variations in power grid synchronization to prevent failures. It utilizes the windows comparator method for voltage variations and the zero-cross detector method for frequency monitoring.

Authors	Title	Main Findings
Ten, Woei, Hong, Chen-Ching Liu Chee- Junho and	Anomaly detection for cyber-security of the substations [15]	The paper addresses cybersecurity issues in power system substations by proposing an anomaly inference algorithm for early detection of cyber-intrusions, considering simultaneous attacks across multiple substations.
Asefi, Sajjad, et al.	Anomaly detection and classification in power system state estimation: Combining model-based and data-driven methods [16]	Addresses challenges in power system state estimation due to various anomalies, including measurement errors and false data injection at- attacks. It introduces a new algorithm combining analytical and machine learning approaches to detect, classify, and identify the origin of anomalies. Its accuracy and effectiveness were evaluated on the IEEE 14 bus test system.
Yen, Soo Wan, et al.	Effect of smart meter data collection frequency in an early detection of shorter-duration voltage anomalies in smart grids [17]	The study investigates the role of smart meters in improving smart grid operations by enabling real-time high frequency data collection and ana- lysis, demonstrating effective detection of short- duration voltage anomalies.
Wadi, Mo- hammed, and Wisam Elmasry	An anomaly- based technique for fault detection in power system networks [5]	The paper presents an anomaly-based technique for fault detection in electrical power systems, utilizing One-Class Support Vector Ma- chine and Principal Component Analysis models, and demonstrating its effectiveness through ROC curve analyses.
Mozaffari, Mahsa, Keval Doshi, and Yasin Yilmaz	Real-time detection and classification of power quality disturbances [18]	The paper presents a real-time detection and classification method for power quality disturbances in power delivery systems, utilizing sequential, multivariate, non-parametric, and supervised learning from training data.

3 Requirement Engineering

3.1 Basic Engineering

In the introduction chapter, it is discussed how the use of modern technology has made power system networks more complicated, which also makes maintenance of system security and reliability difficult. Power outages and voltage fluctuations are examples of anomalies that can impair vital services and endanger human life. Before implementing real-time detection techniques, it is necessary to address the requirements and systematic approach. This ensures satisfaction of stakeholders' expectations and specifications which aligns with the primary goal of requirement engineering.

3.1.1 Key IEEE standards

1. **IEEE 1159: Recommended Practice for Monitoring Electric Power Quality**

This standard provides definitions and nomenclature for several types of events that might affect the quality of power. It serves as a comprehensive guide to ensure consistency in description and classification of power quality occurrences. It characterizes abnormal behavior based on instantaneous rms voltage levels with respect to the normal rms voltage of a signal - this is of major significance for us in further implementation.

2. **IEEE 1564: Guide for Voltage Sag Indices**

This standard specifies indices and characteristics for assessing voltage sags and swells in electrical power and supply systems. It also outlines the methodologies for calculating these indices in 50/60 Hz power systems. Particularly, it states that for 50 Hz signals, the rms voltage calculations need to be performed on a 10-cycle window. The rms is calculated for one cycle, updated every half cycle

3. **IEEE 519: Recommended Practices and Requirements for Harmonic Control in Electrical Power Systems**

This standard outlines the procedures for analyzing the frequency behavior of electrical signals using Fast Fourier Transform (FFT). It defines Total Harmonic Distortion (THD) as a measure of signal distortion due to harmonics. According to IEEE 519-2014, a THD of 8% is considered an acceptable threshold. The use of FFT allows for the detection of primary frequencies, which are essential for ensuring that the system operates within its design parameters. Identifying anomalies in these frequencies is crucial for maintaining the overall performance and reliability of the power system.

3.1.2 Connection to Hardware and Software Selection

Deploying power quality monitoring and anomaly detection systems in compliance with IEEE standards requires careful hardware and software selection. This includes:

Continuous Data Acquisition: Hardware must be able to continually monitor voltage and current parameters to collect real-time data on power quality events.

Data Processing and Transmission: The system should facilitate the easy transmission of data to centrally located processing units for analysis. Thus, reliable communication is important.

In addition to this, software needs to be able to manage enormous volumes of data, evaluate it instantly, and generate actionable insights.

3.2 Detailed Engineering of Hardware Components

3.2.1 Beckhoff's Industrial PC

Many automation operations, including data acquisition, image processing, interconnection of plant components, control of machines, processes, and logistical systems, revolve around Beckhoff Industrial PCs. Hardware PLCs are progressively being replaced by PC-based control technology, which offers superior scalability and flexibility for conventional control tasks.

Also, scalable, and ultra-compact industrial PCs such as the Beckhoff's IPC used in this experiment, offer a multitude of installation options for the control cabinet along with the highest possible computing capability in the most compact conceivable size. It is ideal for communication, control, and visualization—into the cloud, for instance. The PCs are particularly ideal for usage in Industry 4.0 applications, such as an Internet of Things gateway, because of their remarkable capacity for computation in relation to their size.

We believe that the BECKHOFF IPC has been provided for this application due to its numerous advantages over other PLCs, including high performance, dependable I/O module support, adaptable architecture, strong real-time processing with TwinCAT 3, and integrated analytics capabilities. Because of these characteristics, BECKHOFF Industrial PCs are especially well-suited to meet the demands of an energy measuring and anomaly identification framework, providing dependable, efficient, and scalable industrial systems to manage energy.

However, there is a limitation in the storage capacity of the provided IPC for data logging and analysis. A higher storage capacity is therefore highly advised for future experiments as the current version placed some limitations on the executed experiment.

3.2.2 I/O Modules

Two communication modules were used in this experiment namely: the EK1100 EtherCAT Coupler and the EL3783 input/output module.

EK1100: This EtherCAT Coupler is the link between the EtherCAT protocol at the fieldbus level and the EtherCAT Terminals. The coupler converts the passing telegrams from Ethernet 100BASE-TX to E-bus signal representation. A station usually consists of a coupler and several EtherCAT Terminals (up to 65,535) which are automatically detected and individually displayed in the process image. It usually has a 2x RJ45 socket for connecting the coupler to the internet. The coupler provides a maximum of 5V and 2A to all connected EtherCAT Terminals, which are required for communication.

EL3783: It is a power monitoring terminal designed for state monitoring of 3-phase Ac voltage systems. For each phase, voltages and currents are sampled as instantaneous values. True RMS values or power calculations as well as complex user-specific can be calculated using the voltage and current characteristics. It works based on the EtherCAT oversampling principle with a temporary resolution of up to 50 microseconds and passed on to the control system. The oversampling function enables the terminal to measure signal values at shorter intervals than the cycle time of the control system.

I/O modules were used in this experiment to ensure that the signals generated using the Analog Discovery 2 tool were properly transmitted to the IPC for further analysis.

3.2.3 Analog Discovery Tool

Digilent's Analogue Discovery 2 is a multifunctional instrument which allows users to generate, record, measure, visualize, and operate an extensive range of mixed signal circuits. It provides feasibility to work with analogue and digital circuits in almost any setting. Below are a few of the properties of these tools that make it suitable for this application.

- The tool is equipped with dual-channel function generators and oscilloscopes which gives it the capability to generate precise waveforms and signals that can mimic various electrical anomalies such as voltage spikes, sags, and harmonic distortions.
- High sampling rate and resolution: The device offers a sampling rate of up to 100 MS/s (mega-samples per second) and 14-bit resolution. To precisely capture the subtleties of electrical anomalies, high sample rates and resolution are essential, and this device has such functionality.
- The device also provides a wide range of measurement capabilities. Features like an oscilloscope, logic analyzer, spectrum analyzer, and network analyzer come with Analogue Discovery 2. A thorough investigation of the generated anomalies becomes possible using these analyzing tools. With the aid of these instruments, it is possible to examine various aspects of the signals, such as their frequency composition, timing characteristics, and general waveform integrity, ensuring an accurate evaluation of the variances.

- Analog Discovery 2 provides flexible and configurable software interface: The device uses the WaveForms software, which provides a user-friendly interface for configuring, controlling, and visualizing measurements. Due to the flexibility and ease of use provided by the WaveForms software, quick setup and modification of signal parameters are made easy. This enables users to easily configure the software to generate specific types of anomalies and adjust their characteristics, making the tool highly adaptable to various research needs.

3.3 Detailed engineering of software components

3.3.1 TwinCAT 3 and PyADS for data collection

Besides the Waveform software which is used for Analog Discovery 2 device to ensure that different forms of signals are generated, the TwinCAT 3 controls the Beckhoff's IPC and the I/O modules. Below are some properties of the TwinCAT 3 which makes it suitable for this application:

- It provides real-time control and data processing.
- The TwinCAT 3 Analytics library has the necessary tools required for data analysis, feature extraction and pattern recognition.
- The software provides various interfaces through API which provides choice for analysis of data collected from IPC. One such API provided by the TwinCAT 3 is PyADS which provides an interface for integration with Python
- The TwinCAT 3 development environment is free and can be installed on development PC, but TwinCAT 3 analytics and real time communication modules need licenses [19].

3.3.2 Detection algorithms using Python language as tool

Python is a desirable choice for implementing anomaly detection algorithms due to its comprehensive libraries like NumPy, pandas, and scikit-learn, which facilitate efficient data handling, preprocessing, and machine learning. Its ease of use and readability accelerate the development and deployment of detection models that adhere to IEEE 1159 and 1564 standards. Python's integration capabilities with real-time data processing frameworks and visualization tools like Matplotlib and Plotly enable continuous monitoring, accurate detection, and analysis of voltage sags, swells, and other power quality anomalies, ensuring reliable and standardized power quality management.

4 Experimental Setup

4.1 Data Source

Analog Discovery 2, utilizing the Wave Forms software, was employed to generate a variety of signals. This device can produce voltage signals across different frequencies, with a maximum amplitude of 5V. By adjusting the software settings, precise control over the signal parameters is achieved, making it a versatile tool for testing and experimentation.

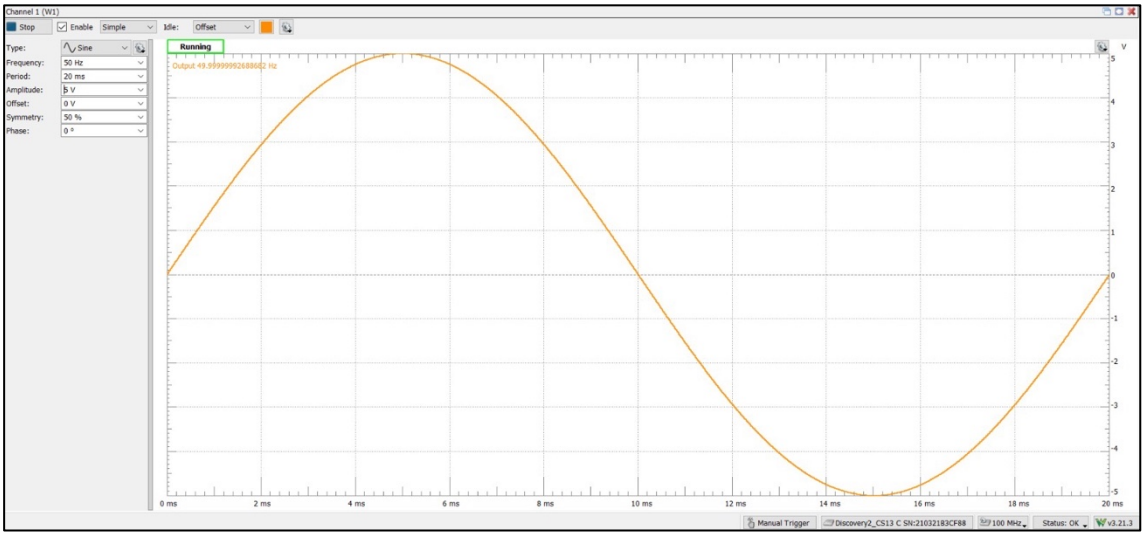

Figure 1 Waveform Software

Using Beckhoff's Industrial PC and the oversampling module EL3783, the voltage signal could be measured in real time. A simple resistor configuration was designed to measure both voltage and current, simulating real-life scenarios of power anomalies in the line. This setup allows for accurate monitoring and analysis of electrical disturbances, providing valuable insights for testing and diagnostics.

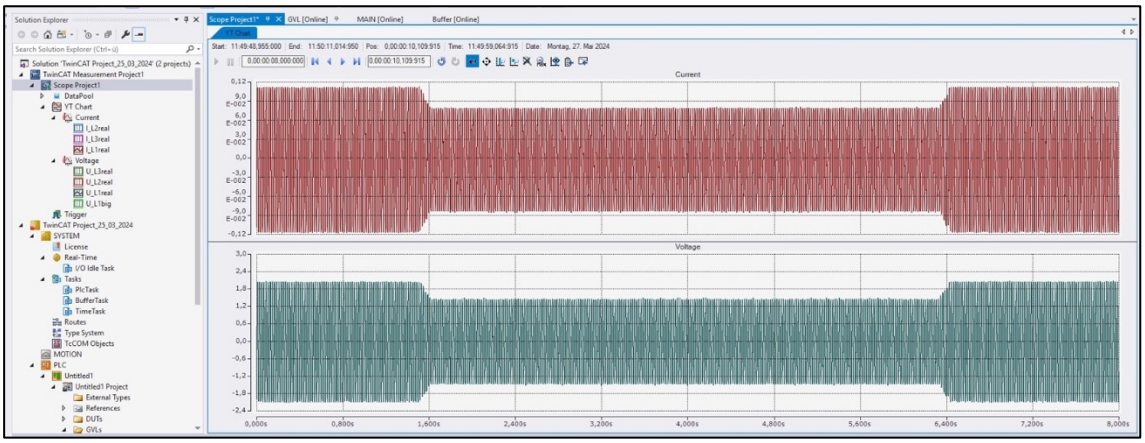

Figure 2 Data Acquisition in TwinCAT 3

4.2 Hardware Infrastructure

The hardware devices which were put together in the laboratory for this experiment are shown in the figure below:

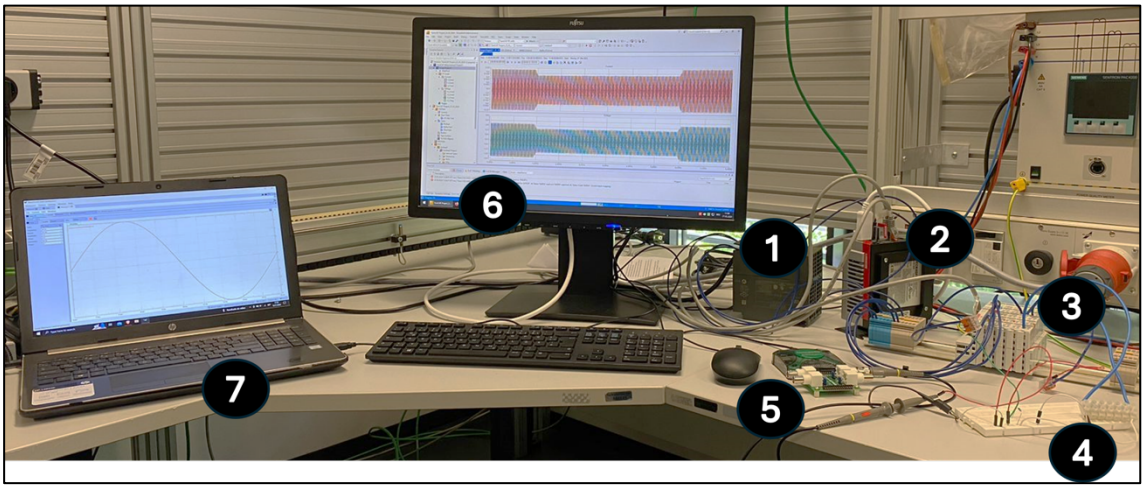

Figure 3 Hardware Infrastructure Setup

The devices, as numbered, include:

1. 24V DC power supply unit (PSU)
2. Beckhoff's Industrial personal computer (IPC) hosting TwinCAT 3 SoftPLC
3. Beckhoff's I/O terminals (EK 1100, EL 3783)
4. 220 ohms resistive load on a protoboard
5. Digilent Analog Discovery 2
6. Computer monitor
7. Laptop hosting Waveform - Digilent's proprietary software

The PSU supplies 24V DC to both the IPC and the input-output terminals. Beckhoff's proprietary software, TwinCAT 3, runs on the IPC which maintains send-and-receive communication with the I/O modules via its Ethernet interfaces. The I/O terminal, Ek 1100 EtherCAT Coupler is the link between the EtherCAT protocol at fieldbus level and the EtherCAT terminals. The Coupler converts the passing telegrams from Ethernet 100BASE-TX to E-bus signal representation. Any number of EtherCAT terminals connected to this coupler are automatically detected and individually displayed in the process image. The connected modules are supplied with the current required for communication from the supplied system voltage. The EL 3783 EtherCAT terminal is a power monitoring I/O terminal used for state monitoring of a 3-phase AC voltage system. Through the oversampling principle with a temporal resolution of up to 50 microseconds, the terminal can measure at significantly shorter intervals than the cycle time of the control system. With the processing power available in this module, true RMS, or power

calculations as well as complex user-specific algorithms can be calculated via the voltage and current characteristics.

To achieve electrical power calculations using this setup, we have provided an electrical load to the system since we are using Digilent's Analog Discovery device as our signal source. We have deployed a 220 ohm purely resistive load in this experiment. This configuration enables us to simulate and mimic many of the anomalies common in the power system networks. The proprietary software, Waveforms provided by Digilent is used to drive the operations of the Analog Discovery tool. Within the limit of the operational capacity of the tool, we generated signals which were sent to the I/O modules and then monitored in the TwinCAT 3 platform.

4.3 TwinCAT 3 Setup

A new PLC project is set up in TwinCAT 3 and required libraries are added in the References.

I/O Devices:

Scan for the I/O Devices from TwinCAT 3. It will scan and show all the I/O devices connected with the PLC in our project like EL1008 (EtherCAT 8-channel digital input Terminal), EL3443(EtherCAT 3-channel analog input Terminal), EL3783(EtherCAT 3-channel analog input oversampling terminal) and EL9001(Ethernet card).

Variable Declaration:

The variables can be declared in the following places:

- Declaration part of a programming object
- GVL editor

In our program we have declared the variables of current and voltages of all three lines in GVL. Refer Annex 9.3 for variable declaration.

Main PLC Program:

The main PLC program is written in POUs to read the data from PLC as shown in Annex 9.4, which is cyclically fetched by PLC from oversampling module in the form of arrays. A new task is created, and the main program is linked with the task and Task cyclic tick is set at 5ms.

Connect the variables for voltage and current values and set the Oversampling factor to 100. Now build the project, activate configuration, login to the PLC, download the program to PLC and Run the program.

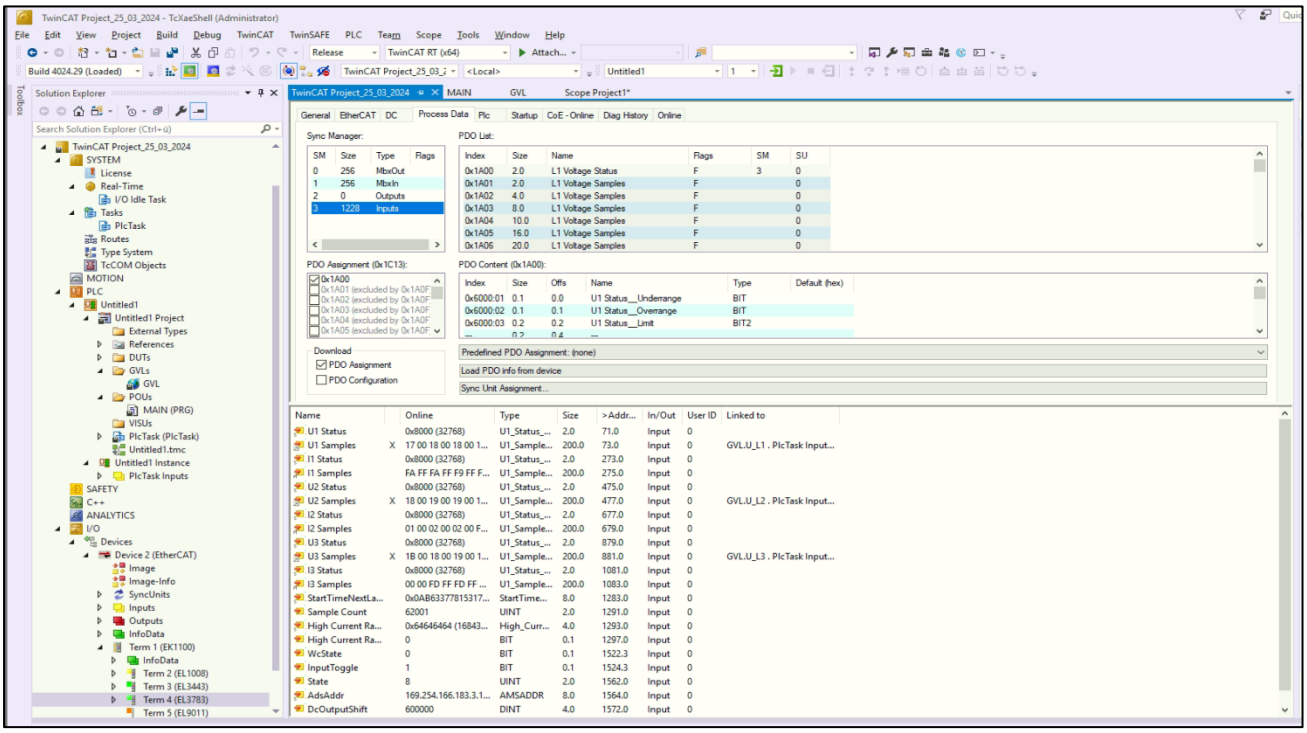

Figure 4 TwinCAT 3 Project Overview

Clock Synchronization:

The distributed clocks in the EtherCAT devices in our project are synchronized so that the digital inputs can be acquired synchronously. DC devices like EL 3783 have their own system time which allows full functions of distributed clocks. The distributed clocks are synchronized between the EtherCAT devices. A reference clock is specified for synchronization.

Timestamping:

Because of real time communication, Timestamping is also necessary for real time data before sending it to Python anomaly detection code. Timestamping is based on the system clock as shown in Annex 9.5.

Data Visualization:

For Data Visualization in TwinCAT 3 in real time a new TwinCAT 3 Measurement Project is created, with a new YT scope project. Scope is added and variables of voltage and current are linked with the scope to visualize in real time as shown in Figure 2.

5 Power Anomalies

Various power anomalies are often found in electrical systems. Understanding these anomalies plays a vital role in identifying potential risks, implementing preventive measures, and ensuring efficient power delivery.

In a broad sense, power system anomalies are classified into two categories namely: the transient behavior variations and the frequency behavior variations. The classification was made based on the type of signal information required to detect the anomalies. The classification also aids the development of the detection algorithm, where certain anomalies using the same information can be grouped together and multiple anomalies can be detected.

5.1 Transient Behavior Variations

Imbalances or disruptions in the electricity system that are transient in nature usually last for a certain period. The stability and functionality of the electrical grid might be significantly impacted by these anomalies, which can arise from a variety of sources. Since such variations are usually time-dependent, the signals must be analyzed continuously in time and for small time frames.

Some common types of transient power system anomalies are discussed below.

5.1.1 Voltage Sag

A voltage sag is alternatively referred to as a voltage dip. It is a temporary reduction of the Root Means Square (RMS) value with respect to the nominal value. The RMS method is one of the most widely used approaches for determining whether and for how long a voltage sag has occurred, as it relies on monitoring the RMS value over time. There could be various causes of the voltage sag and the most common ones are motor starting currents, faults on the power line, sudden increase in load, etc.

One of the methods to detect voltage sag is through threshold value. This method compares the measured voltage with a predefined threshold (e.g., 90% of nominal voltage). If the voltage falls below the threshold for a specific duration, a sag is detected. The IEEE 1159-2019 standard further categorizes this anomaly into instantaneous, momentary, temporary, or long-term sag

In our setup which lasted for a total period of 120 seconds, the voltage sag (1.32V) spanned a duration of 30 seconds while the normal operational voltage (1.65V) before and after the anomaly lasted for the remainder of the test period. Figure 5 illustrates a voltage sag.

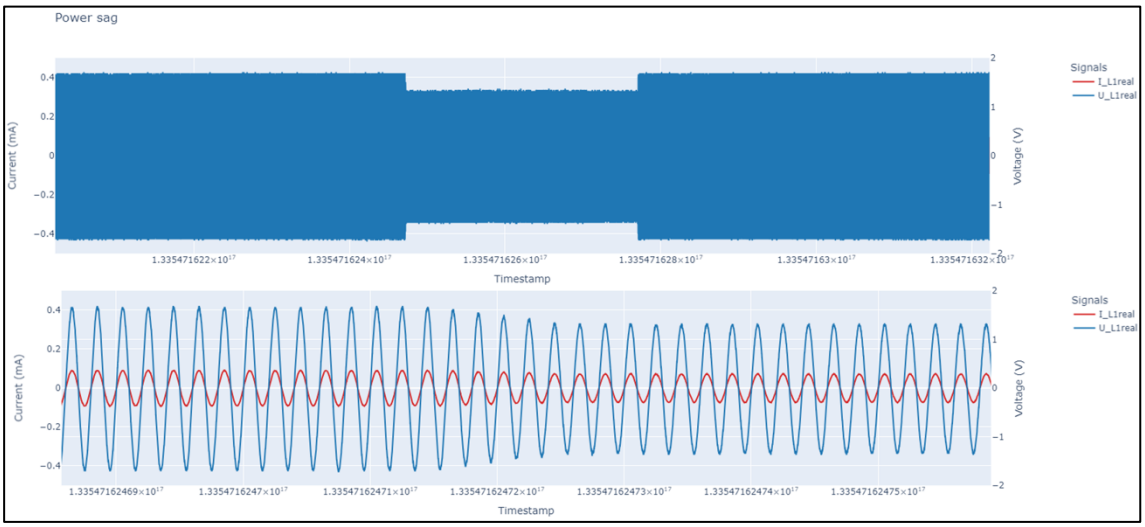

Figure 5 Voltage Sag

Under-voltage is a form of voltage sag categorized under the long duration variation by the IEEE standard and is defined to last for a typical duration of more than 1 minute. Certain events in the power system can result in under-voltage such as switching on a load or switching off a capacitor bank. Overloaded circuits can also cause this fault.

To describe this anomaly using our experimental setup, the normal operational conditions lasted for 25 seconds before the fault and the same duration after it was rectified. The fault took a duration of 70 seconds as can be seen in Figure 6.

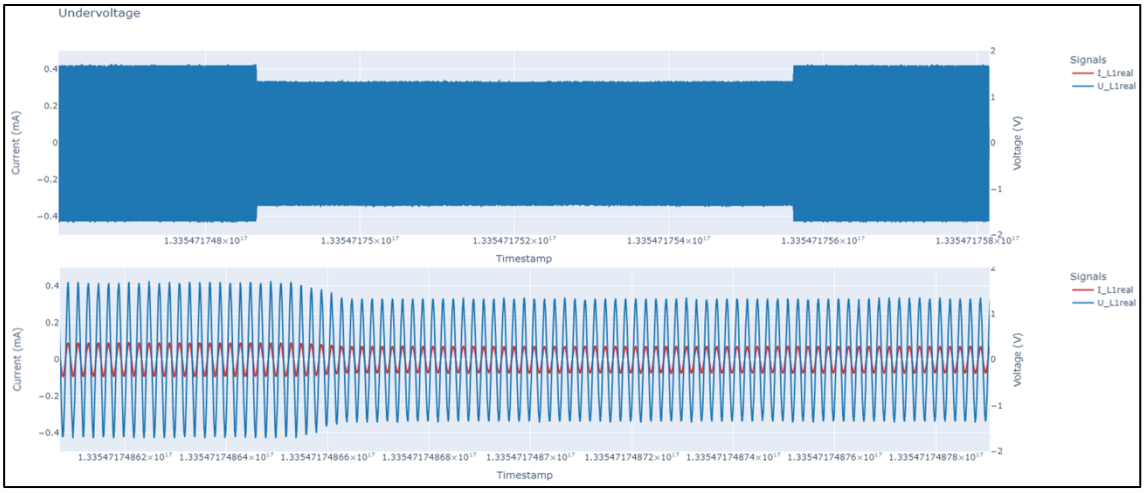

Figure 6 Undervoltage

5.1.2 Voltage Swell

A voltage swell, also known as a voltage surge, is a sudden increase in the rms-voltage value of an electric power supply with respect to its nominal value. Like detection of the voltage sag, the RMS voltage value is calculated and compared to a threshold. According to the IEEE standard, the RMS value surpassing 110% of nominal voltage for a specific duration is considered a voltage swell. [20]

Further classifications of voltage swell include instantaneous, momentary, temporary, and long-term. Correspondingly, their typical voltage magnitudes vary between 1.8 p.u. and 1.1 p.u. [21]. Unlike longer voltage surges, which have lower voltages, voltage spikes typically last only a few milliseconds. While they may not always result in equipment failure due to their short duration, voltage spikes pose a significant risk to sensitive equipment, and in many cases, they can cause catastrophic damage.

From our experimental setup, the Figure 7 shows a typical voltage swell.

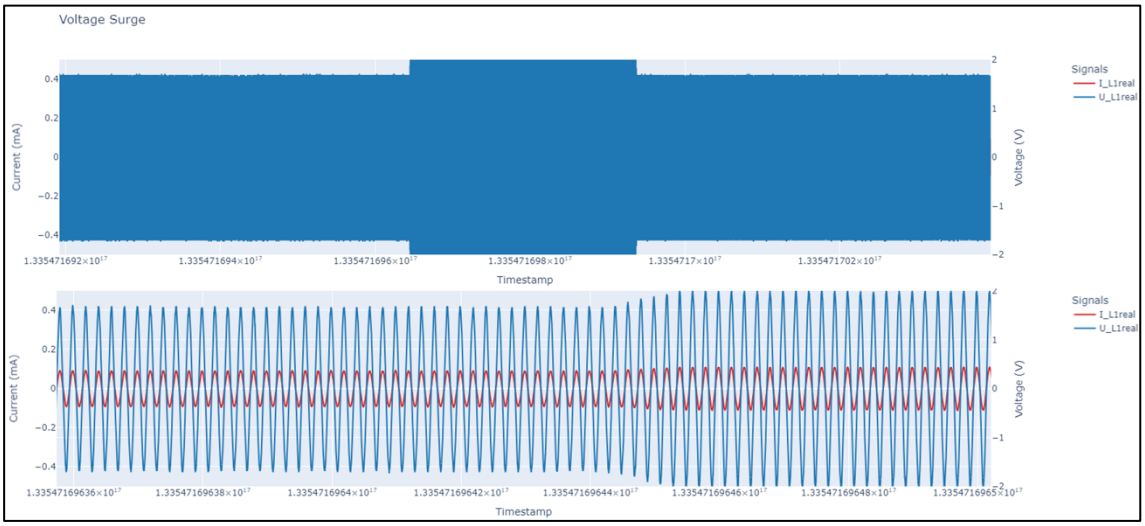

Figure 7 Voltage Swell

IEEE has classified a voltage swell whose duration is greater than 1 minute as overvoltage. Certain events that can cause over-voltage in a power system include, but are not limited to, load switching, variations in the reactive compensation on the system like switching on a capacitor bank, poor system voltage regulation capabilities and incorrect tap setting on transformers. Using our setup to model this anomaly, the fault, 1.98V against nominal value of 1.65V, took a period of 90seconds out of the total test duration of 120 seconds. The Figure 8 demonstrates an overvoltage fault.

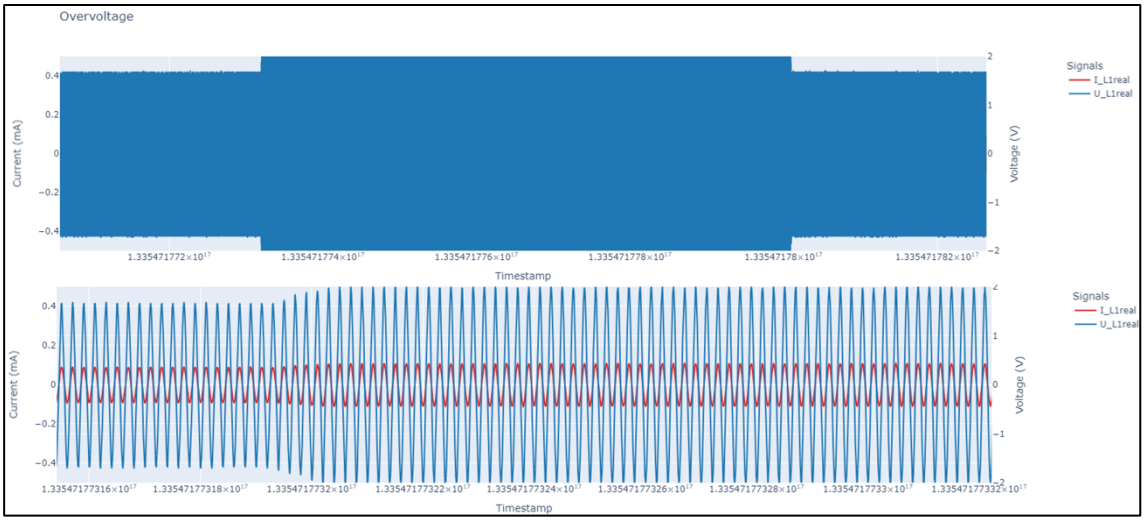

Figure 8 Overvoltage

5.1.3 Power Failure / Interruptions

A power outage, known by various names such as power cut, power failure, blackout, or power loss, refers to the disruption of electrical supply to end-users within the power grid network. Power outages are typically classified into three distinct phenomena based on their duration and impact:

• A transient fault is a brief loss of power, often around 10 seconds, caused by a fault on a power line, such as a short circuit or flashover. Power is automatically restored once the fault is rectified.

• A brownout is characterized by a drop in voltage within the electrical power system. The term "brownout" originates from the dimming experienced by incandescent lighting when voltage decreases. Brownouts can lead to equipment malfunctions or improper operation.

• A blackout represents the most severe form of outage, involving the complete loss of power to a specific area for an extended period, often lasting hours or even days.

Outages can vary in duration, ranging from a few minutes to several weeks, depending on the nature of the blackout and the configuration of the electrical network. These categories delineate the diverse nature and consequences of power disruptions within the electrical infrastructure.

Using the hardware infrastructure setup, voltage and current signals were produced at the output voltage value of 1.65V at normal operational conditions which spanned a period of 15seconds. The loss of signal which depicted power failure lasted for one-and-a half minutes before restoration of the signal. This is in line with [21] which defines various categories and typical characteristics of power system electromagnetic phenomena [22]. Under the long duration variation category, power failure which is referred to as sustained interruption is defined for a typical duration and voltage magnitude of more than 1 minute and 0.0 p.u. respectively. The Figure 9 illustrates this anomaly as described.

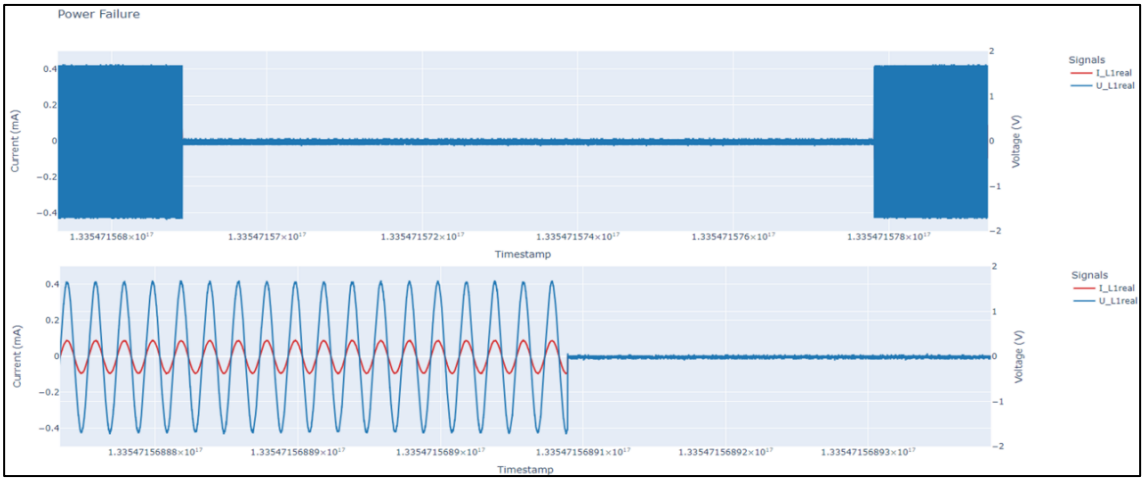

Figure 9 Power Failure / Interruption

5.2 Frequency Behavior Variations

In power systems, frequency behavior describes the changes in the operating frequency of the system caused by imbalances between the load and the generation. Understanding and controlling the frequency behaviors in electrical systems is key to maintaining system balance and stability over time.

Some frequency-based power system anomalies are discussed below.

5.2.1 Electrical Signal Noise

According to [21], unwanted electrical signals with a wideband spectral content of less than 200 kHz that are superimposed on the voltage or current in phase conductors or signal lines of the power system are referred to as noise. Power electronic devices, control circuits, arcing equipment, loads with solid-state rectifiers, and switching power supply can all contribute to noise in power systems. The source of the noise and the properties of the system determine the noise's frequency range and amplitude level. Typically, noise constitutes less than 1% of the voltage magnitude. To demonstrate electrical line noise, we superimposed a 10kHz noise signal on our 50Hz pure sine voltage signal at steady state as shown in the diagram below.

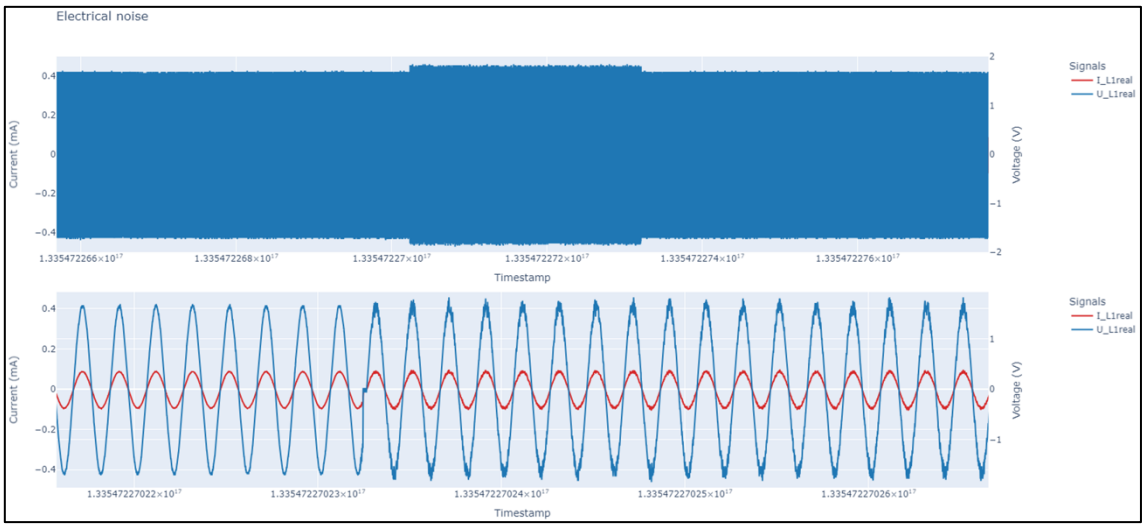

Figure 10 Electrical Signal Noise

5.2.2 Signal Frequency Variation

The generators on the system have a direct relationship between their rotational speed and the power system frequency. The frequency at any given time is determined by how well the load and available generation are balanced. Tiny variations in frequency happen as this dynamic balance shifts. The characteristics of the load and the generation system's reaction to variations in load determine the magnitude and duration of the frequency shift. Frequency changes that exceed the acceptable bounds for the power system's regular steady-state operation are typically the result of large-scale load disconnections, large-scale generation sources falling offline or problems in the bulk power transmission system. To model this fault, we switched the operational frequency from 50Hz to 60Hz for a period of 8seconds, according to the Standard. Power frequency variation is shown in Figure 11.

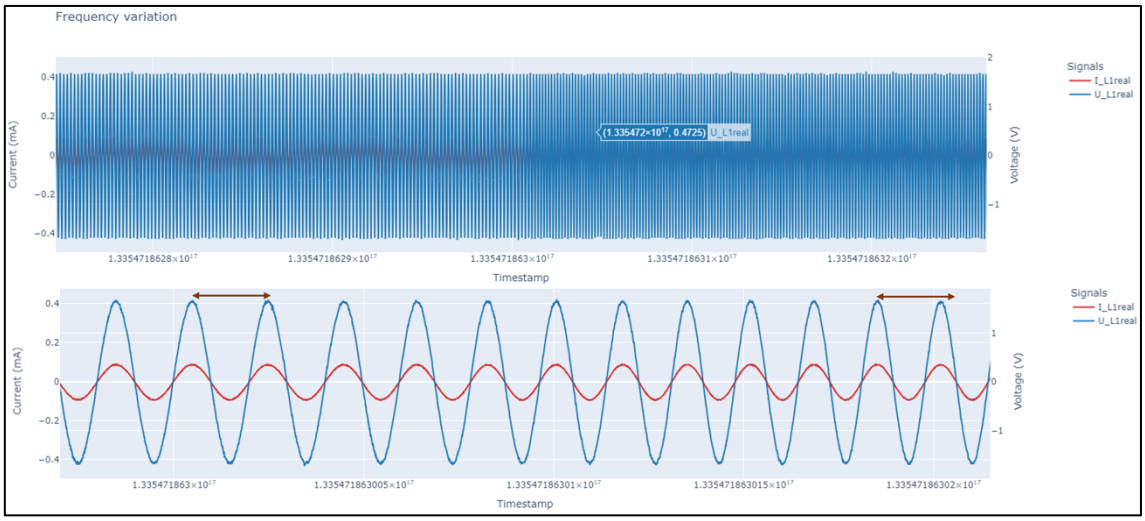

Figure 11 Signal Frequency Variation

6 Communication

To detect anomalies in electrical grids, this part focuses on creating real-time communication between a Python algorithm and a Beckhoff TwinCAT 3 system. The TwinCAT 3 oversampling module collects signal samples at an extremely high rate using oversampling module, which must be sent in real-time to the Python environment so that anomaly detection algorithms may be used. A specific communication channel is needed to bridge the gap between the Python analytic algorithms and the oversampling package, TwinCAT 3, to accomplish this smooth data flow. We then investigated several communication approaches to provide this real-time data flow and guarantee efficient anomaly detection capabilities to protect the stability and dependability of the electrical grid.

Potential communication channels for integration of OT data for real-time analysis:

1. **ADS:** This protocol, exclusive to Beckhoff, facilitates efficient data exchange for industrial automation uses. With their intuitive interface, Python modules like PyADS facilitate communication. The real-time performance features and limitations of ADS in this context will be examined in more detail.
2. **Buffer for accessing the data samples:** The establishment and transmission of a buffer are essential components in this method for storing high-frequency signal samples generated by the oversampling module. A data buffer must be configured within TwinCAT 3 to facilitate this process. Subsequently, the Python algorithm can utilize PyADS to regularly retrieve and access the data retained in the buffer. Subsequent investigations will center on the buffer management system within TwinCAT 3, exploring potential challenges related to buffer size and data latency.
3. **ADS notifications:** TwinCAT 3 enables the activation of notifications based on designated events, such as the availability of new data. An in-depth study will assess the viability of utilizing ADS notifications for the transmission of signal samples.
4. **MQTT communication with TF6701:** TwinCAT 3's TF6701 library provides real-time MQTT protocol data exchange features. Because of its lightweight design, the MQTT protocol makes communication possible even in low-resource contexts.

6.1 ADS Communication

The ADS (Automation Device Specification) protocol is an open and free protocol provided in the TwinCAT 3 system by Beckhoff Automation. It is a transport layer protocol and supports data exchange between various software modules from any point in TwinCAT 3. For example, communication between PLC and numerical control(NC) within the TwinCAT 3 environment. Through this communication is possible with any tool at any point within TwinCAT 3.

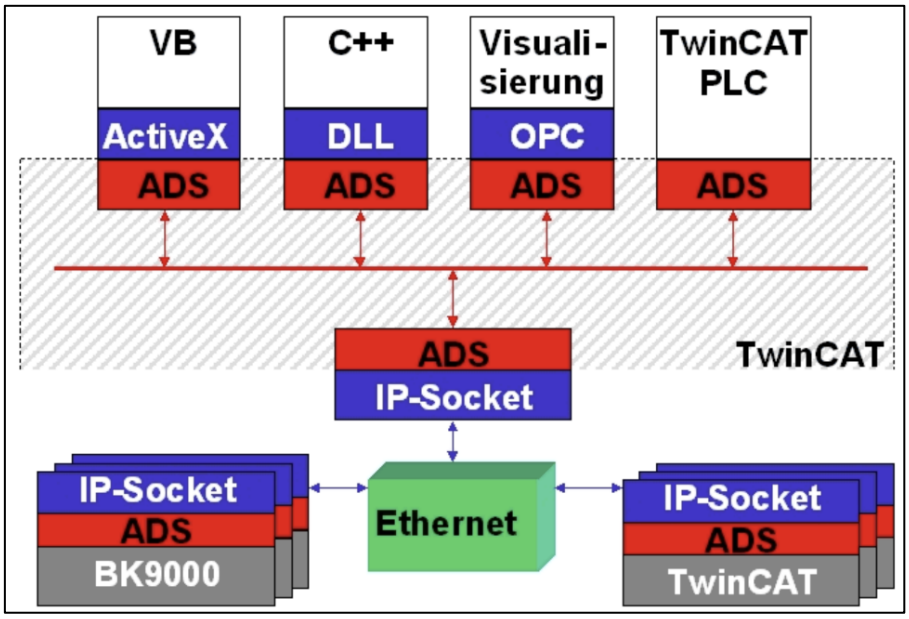

Figure 12 ADS Communication [1]

ADS also makes communication possible with other PCs and devices within the networked system. The ADS protocol is based on TCP/IP in this data exchange, and it is possible to reach all data from any point in the network. ADS interface is supported by TwinCAT 3 'message router' which allows server and client programs to exchange commands and data.

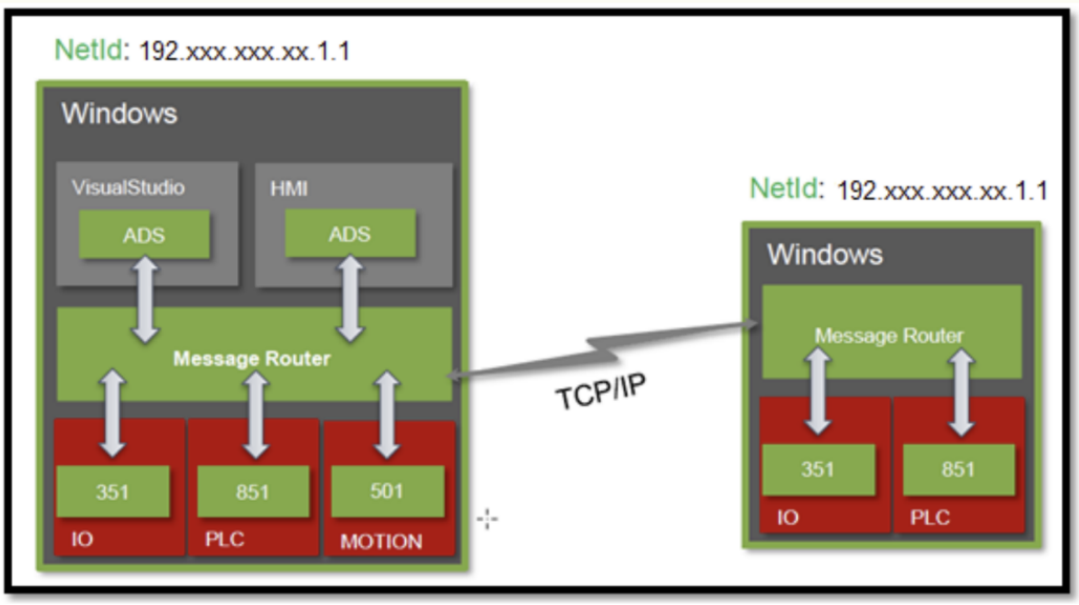

Figure 13 ADS Communication via TCP/IP [2]

[1] https://infosys.beckhoff.com/english.php?content=../content/1033/cx8095_hw/1610551947.html&id=

[2] https://www.plccoder.com/communicating-between-beckhoff-controllers-part-2-ads/

The structure of ADS communication is described below:

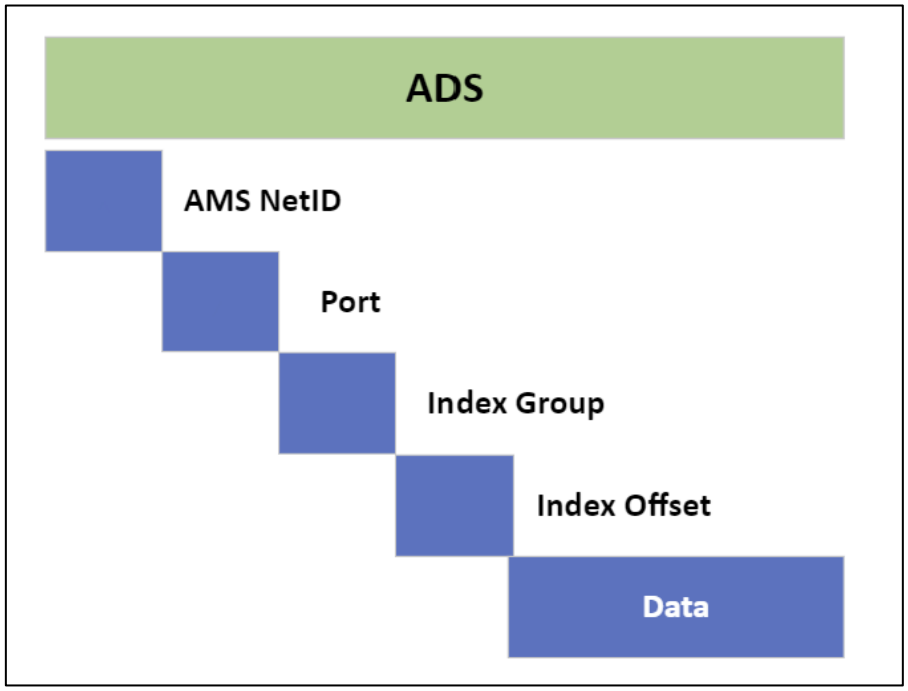

Figure 14 ADS basic structure

1. AMSNetID: The AMSNetID describes the TwinCAT 3 system to be addressed. This is taken from the Mac address of the first Ethernet port (X001). For example, in our experimental setup the AMSNetID is 169.254.166.183.1.1
2. Port number: The port number distinguishes each ADS device. For example, 501 is NC and 851 is PLC runtime1.
3. Index Group: The index group distinguishes different data within a port.
4. Index Offset: Specifies the offset from which byte is to be read or written.
5. Index Group: Specifies the length of the data in bytes to be read or written.
6. Len: Specifies the length of the data in bytes to be read or written.
7. TCP port number: The TCP port number for the ADS protocol is 48898 or 0xBF02.

6.1.1 PyADS

For communication on the python side, we used PyADS, a Python wrapper for Twin CAT 3's ADS library to communicate with TwinCAT 3 devices using the Python programming language.

Setup

Installation:

```
$ pip install pyads
```

Quick Start:

ADS uses its own address system named AMSNetID to identify devices. The assignment of a device to an AMSNetID happens via routing.

```
1. import pyads
2. AMSNETID = "169.254.166.183.1.1"
3. plc = pyads.Connection(AMSNETID, pyads.PORT_TC3PLC1)
4. plc.open()
5. plc.close()
```

Data flow

The oversampling module in the TwinCAT 3 system is intentionally designed to capture signal values at a rapid rate of 50 microseconds per sample. This meticulous data acquisition capability enables a thorough analysis of the electrical grid's performance.

The received signal samples are then stored in a structured array within the TwinCAT 3 software. Given that this array can hold up to 100 samples, it can gather data for up to 5 milliseconds (100 samples * 50 microseconds/sample = 5 ms)

This time frame is equivalent to the Programmable Logic Controller's (PLC) 5 millisecond execution cycle.

To streamline the process of transferring the collected signal samples to the Python environment for anomaly detection, the Automation Device Service (ADS) protocol is utilized. We then utilize pyADS library to initiate connections with the TwinCAT 3 PLC through the ADS protocol. Through the implementation of pyADS functions, the Python program can retrieve the sample array contained in the memory of the TwinCAT 3 PLC, facilitating the extraction of essential signal data.

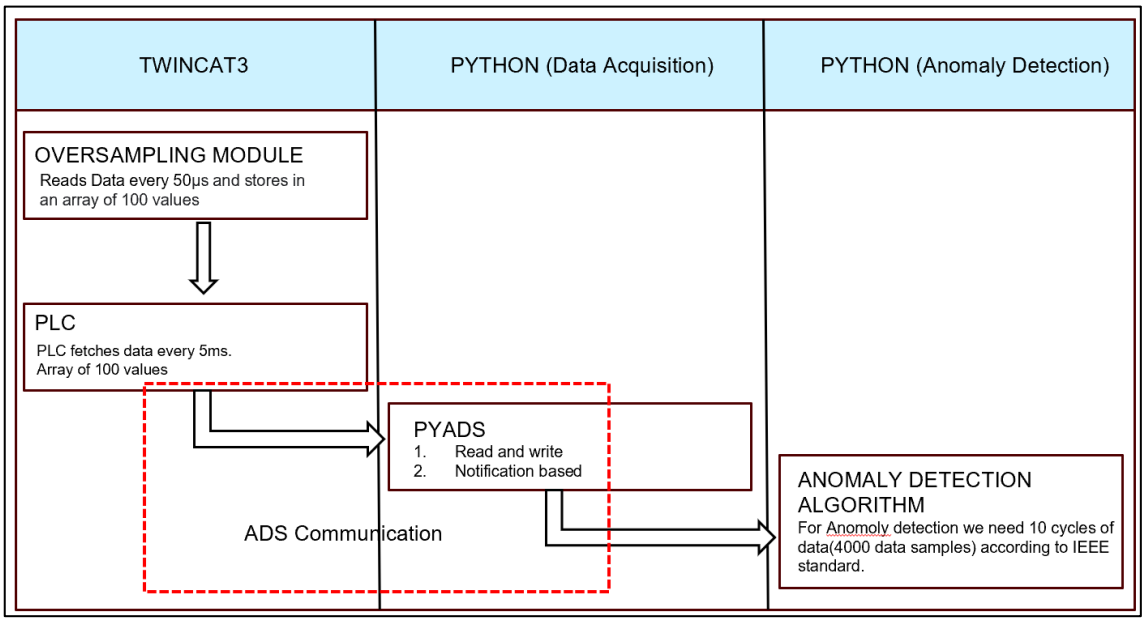

Figure 15 Dataflow when using PyADS

6.1.2 Data acquisition via PyADS: Read and Write

The first methodology employed was the usage of pyADS in the Python programming environment to read signal data samples from a TwinCAT 3 at regular time intervals.

Setup

In the prior section on pyADS setup, it was explained that the program makes use of the pyADS library to establish communication with TwinCAT 3 PLC via the ADS protocol. Upon successful connection, the functions in pyADS enable the Python program to retrieve the complete sample array from a specified memory location within TwinCAT 3 PLC. This memory location corresponds to the 100-sample array generated by the oversampling module. This can be done by using direct functions available Connection.read_by_name() and Connection.write_by_name() but alternatively custom function can also be used to read and write array data at once [23].

```
1. def read_int(plcInput, variable):
 2.       var= plcInput.read_by_name(variable, pyads.PLCTYPE_INT*100)
 3.       print(f"For variable {variable}:")
 4.       print(var)
 5.       return var
 6.
 7. def read_real(plcInput, variable):
 8.       var= plcInput.read_by_name(variable, pyads.PLCTYPE_REAL*100))
 9.       print(f"For variable {variable}:")
10.       print(var)
11.       return var
12.
13. read_int(plcInput = plc, variable = "MAIN.I_L1")
14. read_real(plcInput = plc, variable = "MAIN.I_L1real")
```

Observations:

Inconsistency in Data Acquisition Times: Upon examination, discrepancies were noted in the timestamps linked to the collected data samples. These discrepancies suggested that there were instances of data packets being dropped while acquiring data through pyADS. This is evident from timestamps highlighted in Figure 16 and from Figure 17.

time	
DT#2024-05-27-14:27:21T#710ms	[22, 27, 33, 40, 42, 43, 50, 56, 60, 63, 69, 72, 79, 83, 84, 87, 92, 99, 103, 103, 110, 117, 118, 123, 124, 133, 138, 142,
DT#2024-05-27-14:27:21T#830ms	[312, 316, 318, 316, 314, 314, 316, 318, 318, 314, 311, 311, 314, 314, 311, 309, 307, 310, 311, 306, 303, 302, 303, 3
DT#2024-05-27-14:27:21T#970ms	[-345, -348, -349, -349, -348, -344, -343, -343, -342, -339, -338, -332, -327, -325, -326, -327, -324, -319, -314, -316, -
DT#2024-05-27-14:27:22T#120ms	[-31, -33, -38, -45, -52, -53, -57, -66, -71, -72, -73, -80, -88, -90, -93, -99, -104, -110, -114, -117, -120, -124, -131, -134
DT#2024-05-27-14:27:22T#250ms	[-25, -30, -34, -36, -44, -50, -53, -54, -59, -67, -73, -77, -80, -83, -93, -95, -95, -103, -108, -112, -117, -118, -127, -130,
DT#2024-05-27-14:27:22T#400ms	[-22, -25, -32, -41, -44, -50, -52, -56, -62, -71, -71, -76, -79, -86, -92, -93, -98, -103, -110, -114, -114, -119, -123, -132,
DT#2024-05-27-14:27:22T#530ms	[-345, -350, -348, -346, -346, -349, -346, -340, -339, -339, -336, -333, -329, -329, -326, -323, -320, -319, -321, -317, -
DT#2024-05-27-14:27:22T#680ms	[-345, -350, -348, -346, -346, -349, -346, -340, -339, -339, -336, -333, -329, -329, -326, -323, -320, -319, -321, -317, -
DT#2024-05-27-14:27:22T#810ms	[-345, -350, -348, -346, -346, -349, -346, -340, -339, -339, -336, -333, -329, -329, -326, -323, -320, -319, -321, -317, -
DT#2024-05-27-14:27:22T#950ms	[-345, -350, -348, -346, -346, -349, -346, -340, -339, -339, -336, -333, -329, -329, -326, -323, -320, -319, -321, -317, -

Figure 16 Direct data read via pyADS

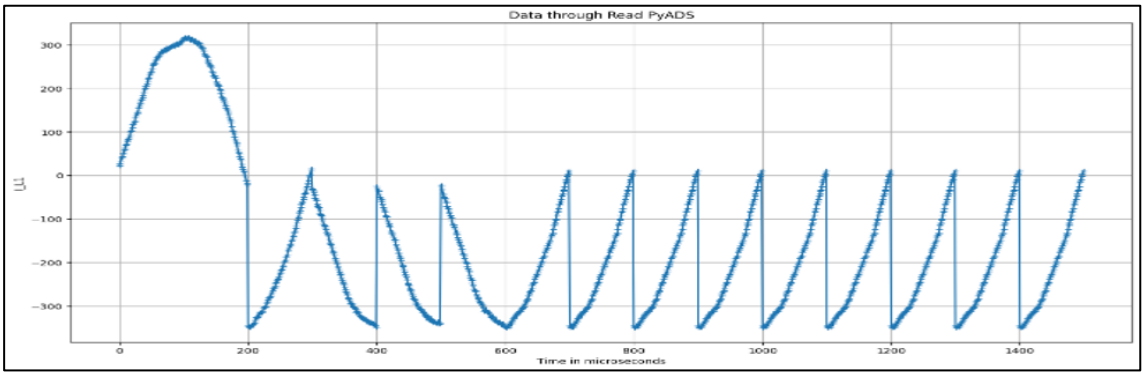

Figure 17 Data Read by PyADS

Investigation results

A careful examination of the available pyADS materials and records could not yield a plausible reason for this behavior. Based on basic communication principles and the features of Python execution, the following might be relevant factors:

1. Python Interpreter Overhead: The computational load imposed by the Python interpreter may cause a small amount of latency while retrieving data. Even while these delays would not be very noticeable in most situations, they could result in data packet omissions in situations when data collecting occurs frequently.
2. Improper pyADS Implementation: While the pyADS library makes communication easier, there can be room for improvement in the current data gathering logic by making better use of pyADS functionalities. Further study may be necessary to improve the Python program's data transmission methods.

6.1.3 Buffering and logging

1. Memory Ring Buffer in Python:

A First-In-First-Out (FIFO) memory ring buffer was implemented within the Python program to temporarily store the acquired signal data. This approach was chosen due to its suitability for time-series data analysis, ensuring the order of data points is preserved for accurate anomaly detection algorithms. While this implementation demonstrated initial success, data loss was still encountered. Further investigation is required to identify and address the root cause of this data loss within the Python environment. [24]

2. Fast logging with data buffer in TwinCAT 3:

TwinCAT 3 also provides examples of data buffer setup in TwinCAT 3 [25] in milliseconds interval, where the data is first collected in a buffer before it is transferred. Theoretically, these buffers can be of different sizes according to requirements and several buffers must be created in which the data samples are combined to avoid a gap in the write process. An example of the implementation is given in the appendix.

This implementation was also evaluated in the TwinCAT 3 environment during our research, but the implementation was not successful since our data to PLC was already in arrays and at an interval of 5ms.

3. Data Buffer in TwinCAT 3:

Setup

To mitigate issues with data loss, a data buffer was directly set up into the TwinCAT 3 environment. To collect signal information, this buffer was implemented in a dedicated PLC task and a loop. The first configuration, as shown in Figure 18, was designed to handle a maximum of two data cycles for preliminary testing.

The initial idea was to increase the buffer capacity after testing successfully as anomaly detection algorithms require data from 10 cycles. Nevertheless, even though this approach seemed logical, the execution of the data buffer did not meet expectations. Examination showed that the reason was a mistake in the TwinCAT 3 task's loop structure, causing improper data storage and duplicated data packets. As one becomes more acquainted with TwinCAT 3 programming, you may be able to address these setup issues.

Observations

The implementation of the TwinCAT 3 data buffer (Figure 18) uncovered a significant problem: repetitive cycles present in the collected signal. The repeated pattern, as seen in the signal, indicates a data processing error stemming from the PLC task that controls the buffer.

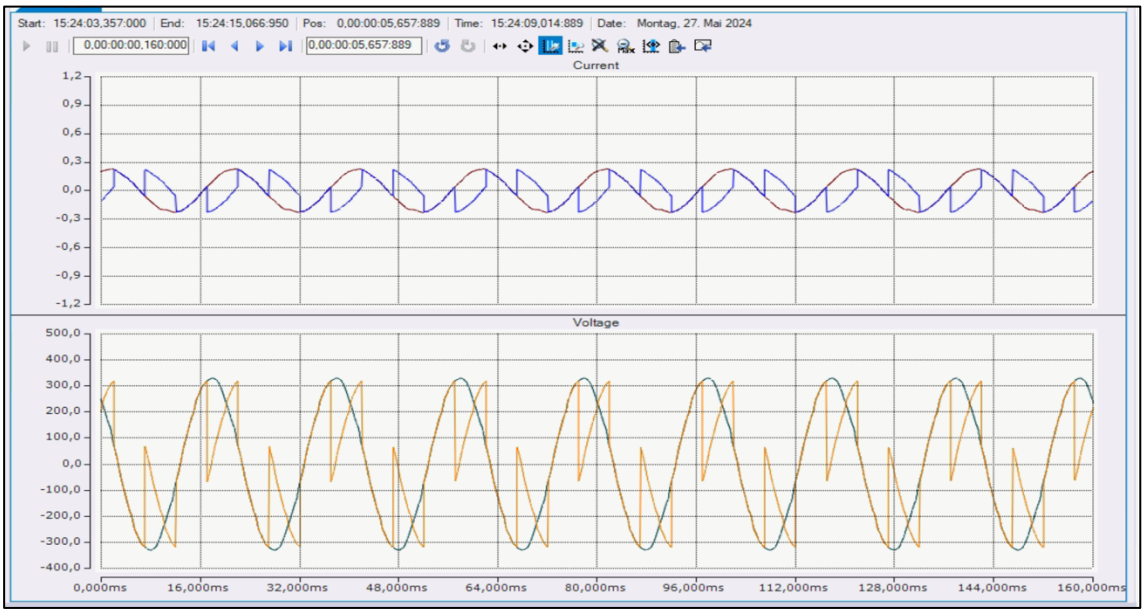

Figure 18 Data acquisition with PyADS read using Buffer

Investigation results

Potential reasons for this error could be due to errors in the loop's configuration logic in PLC task since buffer overflow situations were already addressed. Correcting these setup mistakes in the TwinCAT 3 system is crucial to ensure the correct data filling in buffer and retrieval of data at a later stage.

6.1.4 ADS Notification

ADS notifications are distinct from conventional polling methods as they utilize an event-driven communication approach. Notifications are generated and sent to the connected client based on data changes in specific PLC variables. This method removes the necessity for constant data requests from the client, reducing unnecessary network traffic and communication overhead.

Operational mechanism:

The following steps summarize the capabilities of ADS notifications as shown in Fig. 19: Notification Setup: The specified variables are registered for ADS notifications within the PLC program. With this setup, a notification will be sent out, if their values changes. Data Change and Notification Trigger: An ADS notification is automatically delivered to the Python program whenever the value of a monitored variable in the PLC changes. Data Reception and Processing: The notice with the new data value is sent to the Python program. After then, this figure may be analyzed in real time to look for anomalies.

Why ADS notifications:

ADS notifications have lower latency when compared to polling. The Python program quickly receives alerts when data changes occur, allowing for rapid response to important events in

the electrical grid. This enhanced ability to respond quickly is crucial for detecting irregularities promptly.

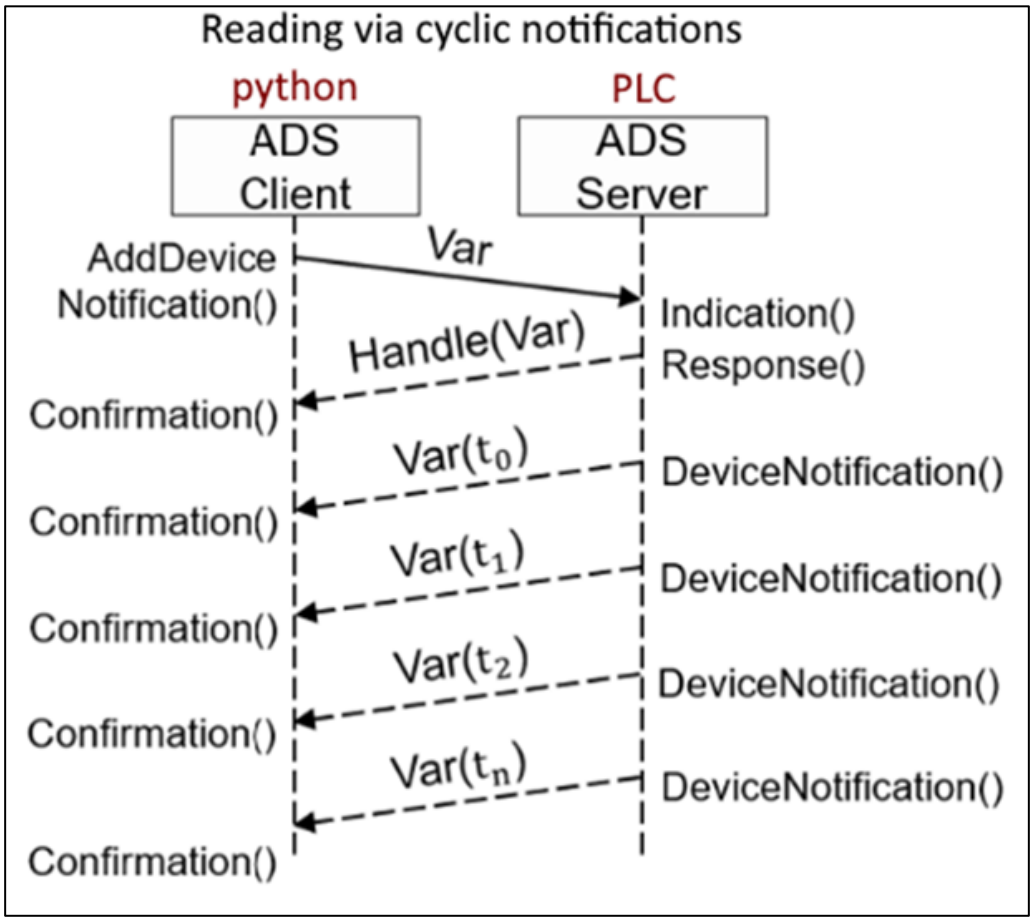

Figure 19 ADS Notification

Setup

```
1. def mycallback(notification, data):
2.
3.     # Append the received value to the array defined outside this function
4.     global values
5.     data_type = tags[data]
6.     handle, timestamp, value = plc.parse_notification(notification, data_type)
7.     values.append(value)
8.
9. # Specify the size and type of the notification attribute
10. attr = pyads.NotificationAttrib(sizeof(c_float))
11.
12. # Add a device notification for the variable we want to monitor
13. handle, user_handle = plc.add_device_notification('GVL.U_L1real', attr, mycallback)
14.
15. # Remove the device notification using the returned user_handle
16. plc.del_device_notification(handle, user_handle)
```

Observations

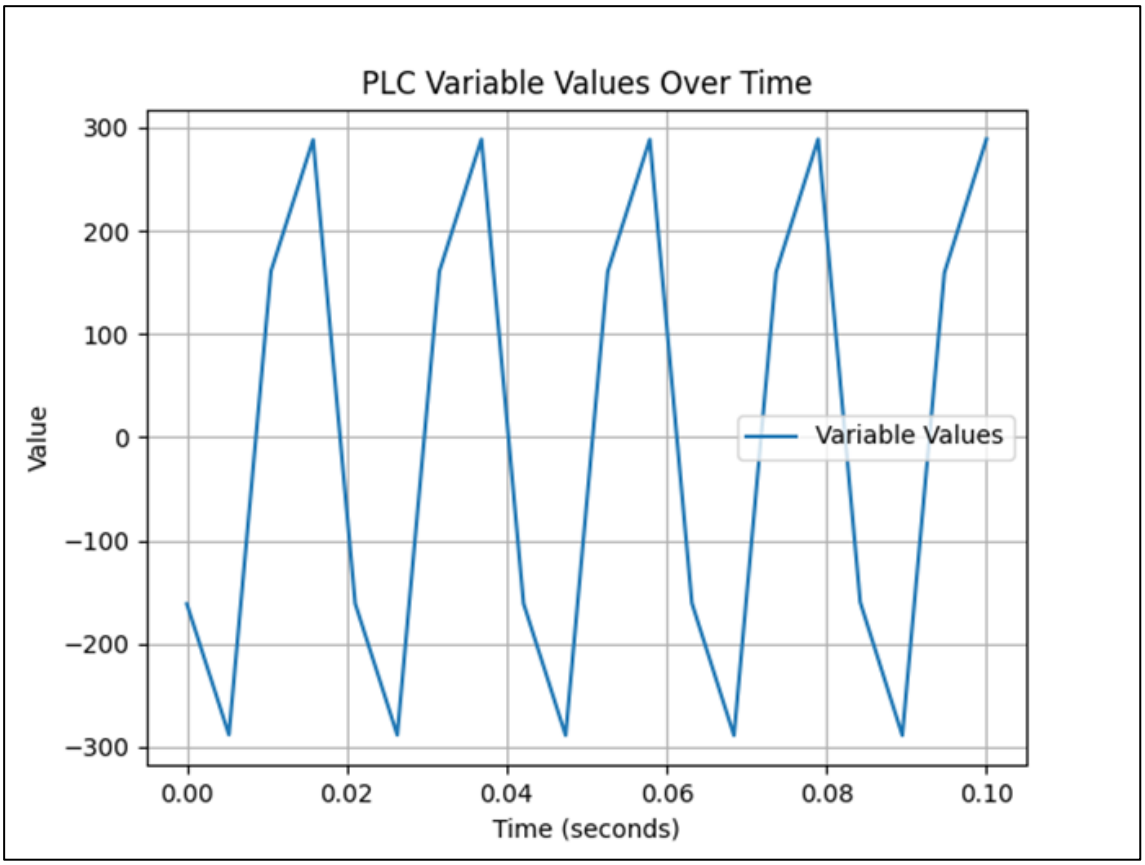

Figure 20 Data acquisition by ADS Notification

ADS notifications performance was a big step up from data buffer approach. It is evident from fig. 20 that the data is captured in streamlined manner but still there are missed data points which could be due to missed notifications.

Investigation results

Even though ADS notifications aid in network congestion, there is still a possibility that some notifications are missed. This is because, received data are added to an array (values) in the mycallback method. But the method may fail to provide notifications for rapidly changing data if its processing or analysis takes a lengthy period – this seems to be the case in our scenario. Delays could be minimized by adjusting mycallback's processing mechanism.

6.2 Comparative analysis of various communication approaches for integration between TwinCAT 3 and pyADS:

This part investigates two main methods for communicating between TwinCAT 3 and PyADS in the case study: using ADS notifications and utilizing data buffers on the TwinCAT 3 PLC.

Benefits of receiving ADS notifications:

1. Decreased Network Traffic: ADS notifications provide a major advantage by sending data only when there is a change in the monitored variable in the PLC. This reduces extra network traffic compared to methods like data buffering or FIFO memory rings that need the complete data stream to be transferred regularly. Efficiency is especially important in industrial network environments with limited resources.
2. Reduced latency: Reduced latency and enhanced responsiveness are achieved through the event-driven approach of ADS notifications, resulting in faster performance than data buffer or FIFO memory ring setups. The Python program quickly gets alerts whenever there are changes in data in the PLC, allowing faster response times for anomaly detection algorithms.
3. Scalability and Efficiency: ADS notifications have built-in scalability and can effectively manage many data sources and clients in a network. This makes the communication architecture more streamlined and decreases the processing burden in the PLC as opposed to handling individual data buffers for each data source.

Benefits of utilizing TwinCAT 3 Data Buffer:

1. Data Storage and Loss Prevention: A data buffer serves as temporary storage for captured signal data on the PLC. This has the potential to reduce the occurrence of data loss situations caused by network interruptions or temporary delays in processing within the Python environment.
2. Logical approach: An effective approach for obtaining real-time data is using a well-configured data buffer in TwinCAT 3, which should satisfy the requirements for real-time data acquisition theoretically. Data points are stored in the buffer every PLC cycle and then transferred to the Python program at scheduled intervals. This method is very direct and easy to understand.

Final word on selection considerations for future work:

The most effective approach for establishing communication between TwinCAT 3 and pyADS will be determined by a combination of factors that were considered during the testing of different methods outlined in the preceding section.

1. Bandwidth: Network bandwidth constraints may necessitate the use of ADS notifications due to their ability to minimize network traffic.
2. Latency: When dealing with high-frequency data, reducing latency is essential. In these situations, ADS notifications are preferred because they have lower latency characteristics
3. TwinCAT 3 expertise: Limited proficiency in TwinCAT 3 development might make it seem that it is easier to implement and maintain ADS notifications compared to rectifying configuration errors in the data buffer setup.

4. Data loss: In cases where anomaly detection algorithms require zero data loss tolerance, utilizing a data buffer would be preferred despite configuration challenges. However, it goes without saying that addressing identified configuration issues in loop is crucial to ensure the reliable functioning of the data buffer.

7 Anomaly Detection

As stated earlier, signals emulating anomalous situations in electrical systems are generated using the Analog Discovery 2 signal generator and are acquired using Beckhoff's EL3783 oversampling module. These can then be viewed, analyzed, and exported using Beckhoff's TwinCAT 3 software. To detect the before mentioned anomalies, the data representing different anomalies is exported (as .csv files) and utilized (the signal data generated from the signal generator was scaled to the magnitude $\pm$330 V to emulate real-life). After analyzing the data and the present literature about anomaly detection, an algorithm was developed based on the existing standards.

7.1 Detection of Transient Behavior Variations

To analyze the change in signal behavior with respect to time, the Root Mean Square (RMS) of the signal was chosen to be an appropriate characteristic. According to [20], the RMS voltage can be calculated as,

$$V_{rms} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} V_k^2}$$

where N – Number of samples per cycle

V_k – Sampled voltage waveform

Following the standard, for a signal with a typical frequency of 50 Hz, the RMS voltage is calculated for every 1 cycle and updated every ½ cycle for a group of 10 cycles.

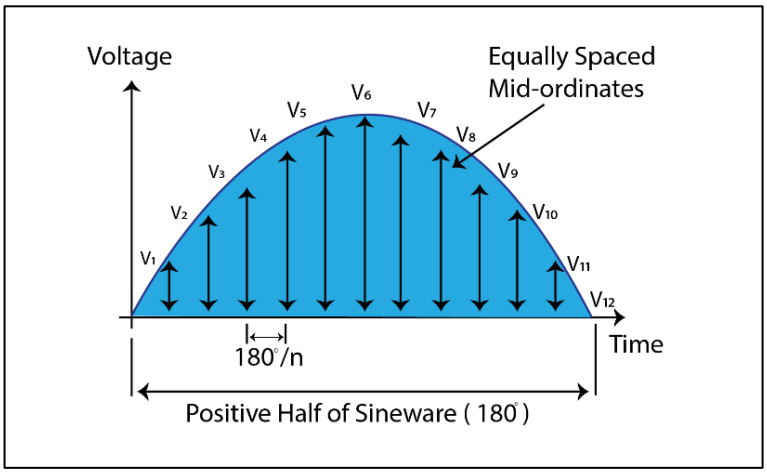

Figure 21 Graphical Representation of RMS Calculation[3]

In this way, the variations in RMS voltage, and consequently the variations in the normal signal can be detected over time. Various anomalies dealing with RMS variations, typical per unit

[3] https://thelinuxforum.com/articles/546-average-value-of-ac-waveform

(p.u.) RMS magnitudes, and their classification according to the duration are provided in [21]. These are also considered as the basis of the development of the detection algorithm.

S. No	Categories	Typical Duration (Cycles)	Typical Duration (Seconds)	Typical Voltage Magnitude
	Instantaneous			
1.1	Sag	0.5–30 cycles	0.01 - 0.6 sec.	0.1–0.9 pu
1.2	Swell	0.5–30 cycles	0.01 - 0.6 sec.	>1.1 pu
	Momentary			
2.1	Interruption	0.5 cycles – 3s	0.01 – 3 sec.	< 0.1 pu
2.2	Sag	30 cycles – 3 s	0.6 – 3 sec.	0.1–0.9 pu
2.3	Swell	30 cycles – 3 s	0.6 – 3 sec.	>1.1 pu
	Temporary			
3.1	Interruption		>3 sec. – 1 min.	< 0.1 pu
3.2	Sag		>3 sec. – 1 min.	0.1–0.9 pu
3.3	Swell		>3 sec. – 1 min.	>1.1 pu
	Long Duration RMS			
4.1	Power Failure		> 1 min.	<0.1 pu
4.2	Under - Voltage		> 1 min.	0.1–0.9 pu
4.3	Over - Voltage		> 1 min.	>1.1 pu

Table 1 Anomaly Detection as per IEEE 1159-2014

Since the transient behavior is analyzed over a total of 10 cycles at a time, during the detection of anomalies, it was simulated that not the complete data, but rather only 10 cycles are treated at once. The developed algorithm is then capable of detecting the beginning and the end of anomalous transient behavior and classifying them based on their durations.

```
10 cycles received at: 2024-03-12 11:14:48.909000
Sag Start: 2024-03-12 11:14:49.079000
Sag End: 2024-03-12 11:14:49.099001
Interruption Start: 2024-03-12 11:14:49.099001

**************************************************
Sag Duration: 0:00:00.020001
Retained Voltage: 23.655662404047128
Sag Type: Instantaneous
**************************************************

10 cycles received at: 2024-03-12 11:14:49.108999

10 cycles received at: 2024-03-12 11:14:49.309000

10 cycles received at: 2024-03-12 11:14:49.509001
```

```
10 cycles received at: 2024-03-12 11:16:18.108999
Sag Start: 2024-03-12 11:16:18.118999
Interruption End: 2024-03-12 11:16:18.118999

**************************************************
Interruption Duration: 0:01:29.019998
Interruption Type: Long Term Sustained Interruption
**************************************************
Sag End: 2024-03-12 11:16:18.139000

**************************************************
Sag Duration: 0:00:00.020001
Retained Voltage: 85.36153729954698
Sag Type: Instantaneous
**************************************************
```

Figure 22 Power Failure Output

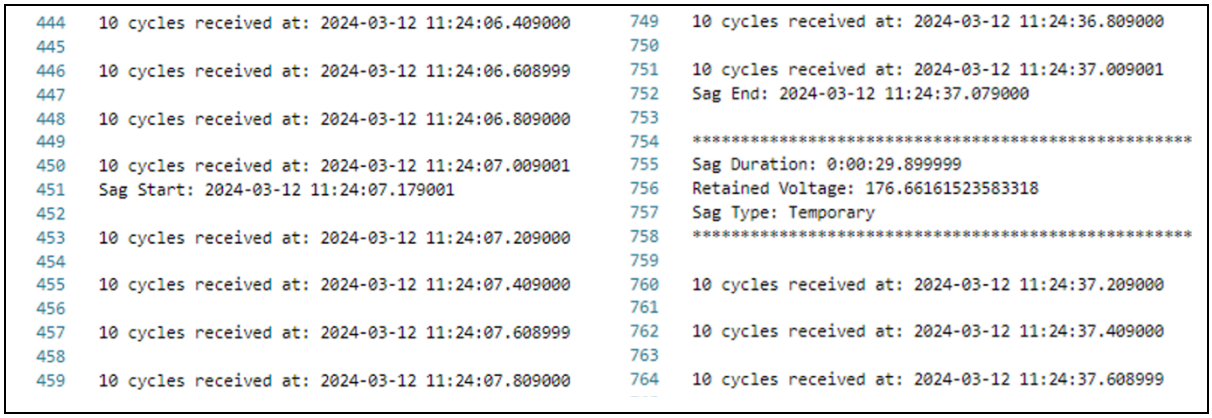

Figure 23 Voltage Sag Output

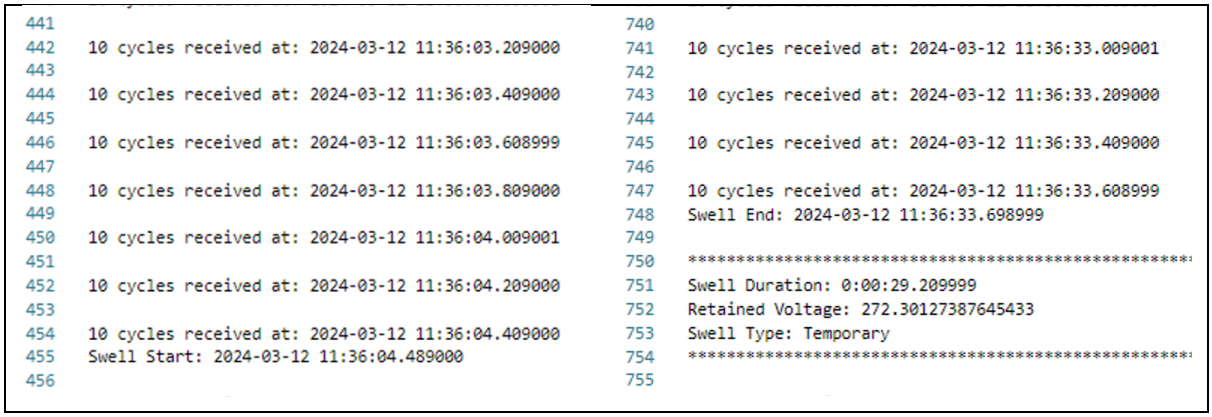

Figure 24 Voltage Swell Output

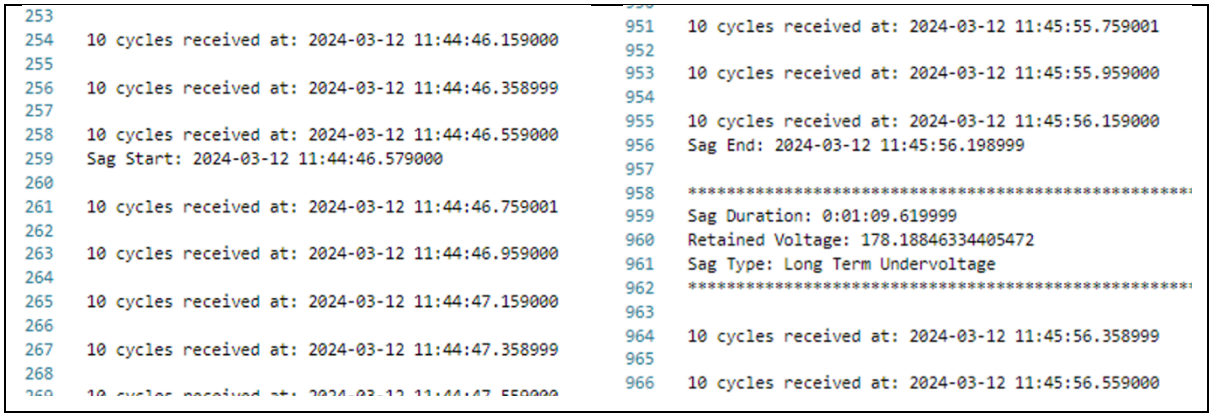

Figure 25 Under voltage Output

```
10 cycles received at: 2024-03-12 11:48:51.159000

10 cycles received at: 2024-03-12 11:48:51.358999

10 cycles received at: 2024-03-12 11:48:51.559000

10 cycles received at: 2024-03-12 11:48:51.759001

10 cycles received at: 2024-03-12 11:48:51.959000
Swell Start: 2024-03-12 11:48:51.989000

10 cycles received at: 2024-03-12 11:48:52.159000

10 cycles received at: 2024-03-12 11:48:52.358999

10 cycles received at: 2024-03-12 11:48:52.559000
```

```
10 cycles received at: 2024-03-12 11:50:00.358999

10 cycles received at: 2024-03-12 11:50:00.559000

10 cycles received at: 2024-03-12 11:50:00.759001

10 cycles received at: 2024-03-12 11:50:00.959000
Swell End: 2024-03-12 11:50:00.959000

**************************************************
Swell Duration: 0:01:08.970000
Retained Voltage: 271.64832539698625
Swell Type: Long Term Overvoltage
**************************************************

10 cycles received at: 2024-03-12 11:50:01.159000
```

Figure 26 Over voltage Output

7.2 Detection of Frequency Behavior Variations

To deal with frequencies present in the signal, Fourier, or Fast Fourier Transform (FFT) analysis can be performed. As stated earlier, frequency behavior of a signal can be considered as an indicator of the overall quality and the “goodness” of a signal. For this purpose, the Total Harmonic Distortion (THD) Percentage of the voltage signal can be a useful indicator. Considering that the algorithm deals with only 10 cycles at a time, the THD% of the received signal can be calculated at once using the formula,

$$THD = \sqrt{\frac{\sum_{k=2}^{N/2} X_k^2}{X_1}}$$

where N – Total Number of Harmonic present in the signal

X_k – Magnitude of kth Harmonic Component

X_1 – Magnitude of Prime Harmonic Frequency

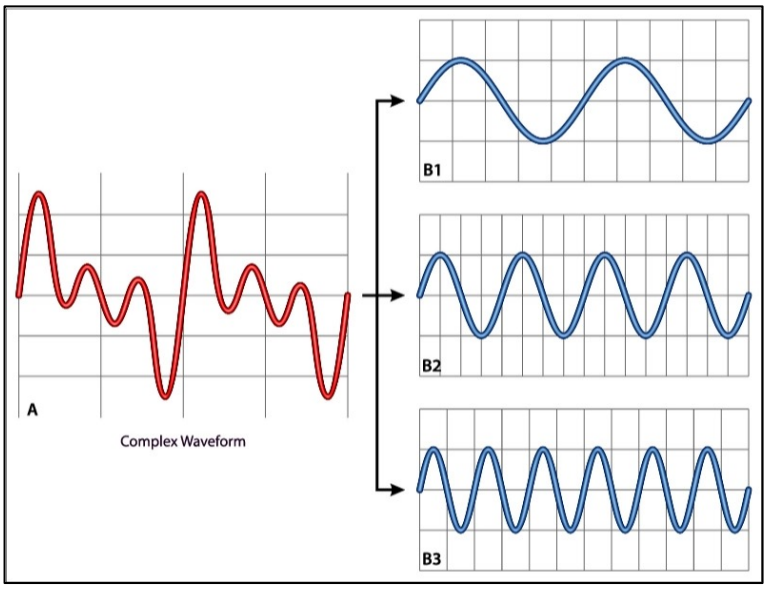

Figure 27 Composite Signal Wave with Multiple Harmonics

According to the IEEE 519-2014, a THD% value of 8% is also considered as an ideal threshold, hence 8% was used while developing the algorithm. This makes the algorithm capable of detecting the poor quality of the voltage signal in situations like the presence of electrical noise.

```
747
748   10 cycles received at: 2024-03-12 13:12:11.459000
749
750   10 cycles received at: 2024-03-12 13:12:11.659000
751
752   10 cycles received at: 2024-03-12 13:12:11.858999
753
754   10 cycles received at: 2024-03-12 13:12:12.059000
755
756   ************************************************
757   THD Threshold Violated
758   THD: 94.23051475930751 %
759   ************************************************
760
761   10 cycles received at: 2024-03-12 13:12:12.259001
762
763   10 cycles received at: 2024-03-12 13:12:12.459000
764
765   10 cycles received at: 2024-03-12 13:12:12.659000
766
```

Figure 28 Total Harmonic Distortion Output

Performing FFT of the signal also helps the algorithm to detect the prime harmonic frequency of the signal, making it capable of detecting the anomalous behavior when there is a change in the signal frequency.

```
564   10 cycles received at: 2024-03-12 12:03:49.358999
565
566   10 cycles received at: 2024-03-12 12:03:49.559000
567
568   10 cycles received at: 2024-03-12 12:03:49.759001
569
570   10 cycles received at: 2024-03-12 12:03:49.959000
571
572   ************************************************
573   Prime Harmonic Variation
574   Dominant Frequency: 55.0
575   ************************************************
576
577   10 cycles received at: 2024-03-12 12:03:50.159000
578
579   ************************************************
580   Prime Harmonic Variation
581   Dominant Frequency: 60.0
582   ************************************************
583
```

Figure 29 Prime Harmonic Variation Output

Therefore, using already established standards as its base, the developed algorithm can work with the generated data and detect anomalous situations correctly.

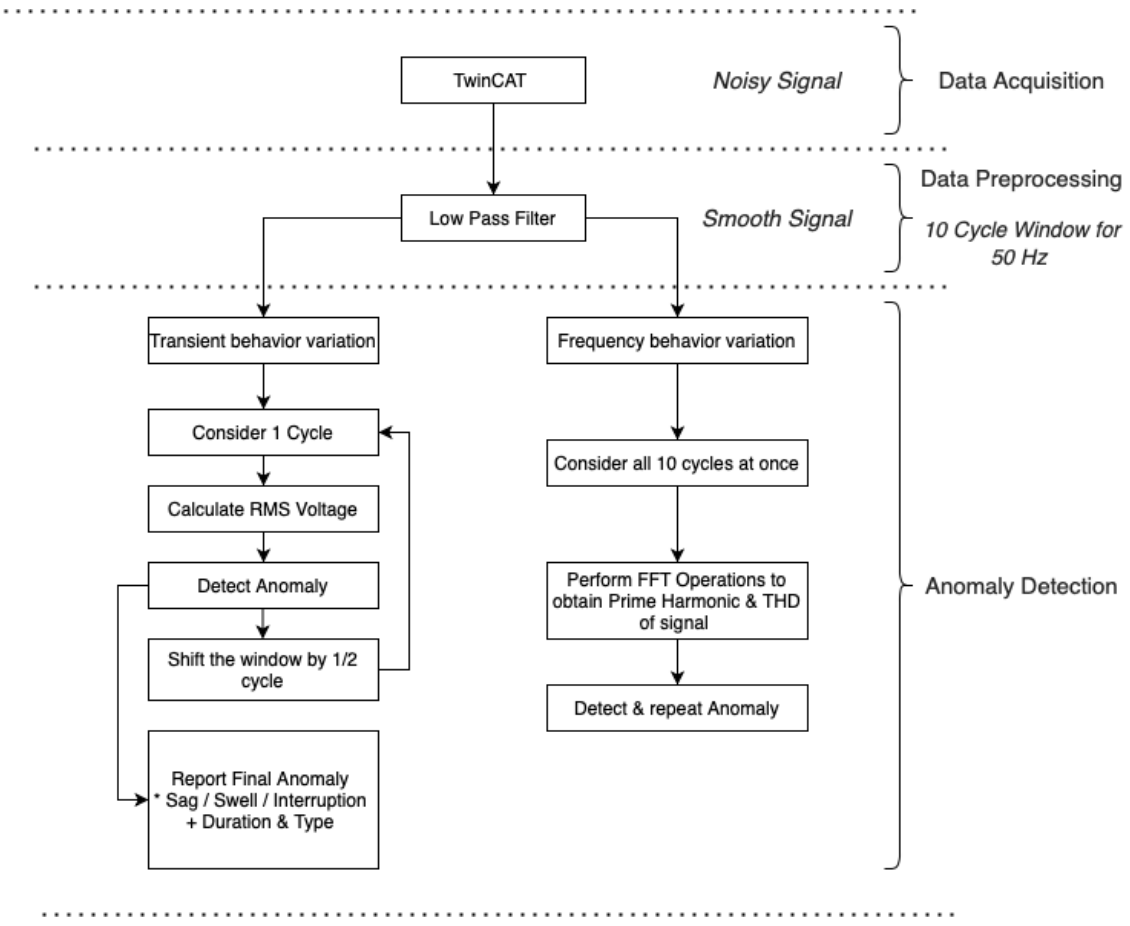

Figure 30 Algorithm Flow for Anomaly Detection

8 Conclusion and Future Work

In recent years, significant advancements have been made in the development of predictive maintenance methods for power systems. The study conducted over the last few months has provided valuable insights, not only identifying the range of anomalies detectable within power systems but also contributing to the development of both the infrastructure and software necessary for this task.

The use of the Analog Discovery Tool for anomaly simulation emerged as a straightforward yet effective method for replicating real-life anomaly scenarios in the absence of equipment capable of measuring phenomena such as frequency variations or electrical noise in transmission lines. Despite these limitations, it was possible to generate and measure such signals on a Beckhoff PLC setup, utilizing a resistor as the load.

However, merely reading the generated signals proved insufficient for the development of an anomaly detection algorithm. The analysis highlighted that the time taken to read and process these values is critical. Consequently, several strategies were pursued to mitigate this issue, ranging from the creation of data buffers to the exploration of various communication protocols for signal transmission.

Despite the substantial effort dedicated to researching and implementing new data transmission methods, considerable potential for enhancement remains. Exploring alternative communication protocols, for example, MQTT, could facilitate the efficient transmission of measured data from TwinCAT3 to a broker for subsequent anomaly analysis. This area offers significant scope for further research and improvement, underscoring the ongoing need for innovation in predictive maintenance technology within power systems.

In summary, while integrating TwinCAT 3 with anomaly detection algorithms for high frequency data acquisition, Data buffer is preferred over ADS communication as it is not hard real-time and ADS loses data packets as Python environment is slow.

Future work should focus on addressing communication challenges in the configuration of data buffer or optimizing ADS notifications for real time communication, exploring alternative protocols like MQTT to enhance efficiency and reliability.

9 Annex

9.1 RMS Voltage Calculation

As stated earlier, to use the RMS voltage to analyze the transient behavior of the signal, 10 cycles are to be considered at a time instance. The main source of data, i.e., the EL3783 oversampling module samples the data at a frequency of $20\ kHz$. Furthermore, the sampled voltage signal has a frequency of $50\ Hz$.

Therefore, for a period of $1\ second$, 50 cycles of the voltage signal can be represented by 20,000 data points.

$$50\ cycles \rightarrow 20000\ values$$

$$1\ cycle \rightarrow 400\ values$$

$$10\ cycles \rightarrow 4000\ values$$

Therefore, 4000 data values (10 cycles) were considered at a time to analyze the transient behavior. Starting from the beginning of these 4000 values, a window of 400 values (1 cycle) was used to calculate the RMS voltage. Then the window was shifted by 200 values (1/2 cycle) and the next 400 values were used to calculate RMS voltage again. In this way, RMS calculated for 1 cycle over 10 cycles by updating every ½ cycle.

```python
def rms_variation(V_array, T_array, anomaly_meta_info):

    # Beginning to end: 4000 values
    # Shifting the window by every 200 values
    for i in np.arange(start=0, stop=len(V_array), step=200):

        # taking 400 values at a time to calculate RMS
        start = i
        end = i + 400

        # RMS calculation
        V_rms = np.sqrt(np.mean(np.square(V_array[start:end])))
```

Figure 31 RMS calculation in algorithm

Using the calculated RMS voltage and the provided corresponding timestamps form the data, the exact time instance of the beginning and the ending of an anomaly could be detected. This, consequently, helps to detect the duration of the anomaly. Using the duration and the type of anomaly, classifications could be made based on the standards.

```python
# reporting detected sag
if (anomaly_meta_info['sag_start'] != None) and (anomaly_meta_info['sag_end'] != None):
    sag_duration = anomaly_meta_info['sag_end'] - anomaly_meta_info['sag_start']

    print("\n*********************************************************************************")
    print(f"Sag Duration: {sag_duration}")
    print(f"Retained Voltage: {anomaly_meta_info['min_sag_value']}")

    if (sag_duration <= datetime.timedelta(seconds=0.599853)): print(f"Sag Type: Instantaneous")
    elif (sag_duration > datetime.timedelta(seconds=0.599853)) and (sag_duration <= datetime.timedelta(seconds=3)): print(f"Sag Type: Momentary")
    elif (sag_duration > datetime.timedelta(seconds=3)) and (sag_duration <= datetime.timedelta(minutes=1)): print(f"Sag Type: Temporary")
    elif (sag_duration > datetime.timedelta(minutes=1)): print(f"Sag Type: Long Term Undervoltage")
    print("*********************************************************************************")

    anomaly_meta_info['sag_start'] = None
    anomaly_meta_info['sag_end'] = None
    anomaly_meta_info['min_sag_value'] = 1000000.
```

Figure 32 Example: Sag classification in algorithm

9.2 THD Calculation

Since detecting variations in the frequency behavior occurs once over all 4000 values (10 cycles), the entire window can be used at once and variations from the threshold can be detected.

To calculate the THD % of the signal, Fast Fourier Transform (FFT) is performed which results in the determination of all the present harmonics and their corresponding maximum amplitudes in the signal. Using this obtained information, THD of the signa and the present prime harmonic can be detected.

```python
def thd_variation(signal, anomaly_meta_info, fs=20000):

    # Taking FFT of the signal
    X = fft(signal)
    N = len(X)
    n = np.arange(N)
    T = N/fs

    # Considering only half due to FFT
    freq = (n/T)[1:int(N/2)]
    freq_amplitudes = np.abs(X)[1:int(N/2)]

    # Sorting frequencies based on detected amplitudes
    sorted_indices = np.argsort(-freq_amplitudes)

    # Getting first frequency as dominant
    dominant_freq = freq[sorted_indices[0]]
    other_freq = freq[sorted_indices[1:]]

    anomaly_meta_info['dominant_freq'] = dominant_freq


    # Squaring amplitudes other than prime frequency
    sq_sum = np.sum(np.square(freq_amplitudes))

    # Squaring amplitude of prime frequency
    sq_prime = np.square(np.max(freq_amplitudes))
    sq_harmonics = sq_sum - sq_prime

    # Calculating THD
    thd = 100 * np.sqrt(sq_harmonics / sq_prime)

    anomaly_meta_info['thd'] = thd
```

Figure 33 THD Calculation in Algorithm

9.3 Variable declaration in TwinCAT 3

```
VAR_GLOBAL CONSTANT

      n         : INT := 100

END_VAR

VAR_GLOBAL


      U_L1 AT %I* : ARRAY[0..(n-1)] OF INT;
      U_L2 AT %I* : ARRAY[0...(n-1)] OF INT;
      U_L3 AT %I* : ARRAY[0..(n-1)] OF INT;

      U_L1real AT %I* : ARRAY[0..(n-1)] OF REAL;
      U_L2real AT %I* : ARRAY[0..(n-1)] OF REAL;
      U_L3real AT %I* : ARRAY[0..(n-1)] OF REAL;

      (*

      I_L1 AT %I* : ARRAY[0..(n-1)] OF INT;
      I_L2 AT %I* : ARRAY[0..(n-1)] OF INT;
      I_L3 AT %I* : ARRAY[0..(n-1)] OF INT;

      I_L1real AT %I* : ARRAY[0..(n-1)] OF REAL;
      I_L2real AT %I* : ARRAY[0..(n-1)] OF REAL;
      I_L3real AT %I* : ARRAY[0..(n-1)] OF REAL;

      P_L1 AT %I* : ARRAY[0..(n-1)] OF REAL;
      P_L2 AT %I* : ARRAY[0..(n-1)] OF REAL;
      P_L3 AT %I* : ARRAY[0..(n-1)] OF REAL;
      *)

END_VAR
```

9.4 PLC Program in TwinCAT 3

```
1. FOR i:= 0 TO (n-1) DO
2.
3.        U_L1real[i] := INT_TO_REAL(U_L1[i]) * 0.0225;
4.        U_L2real[i] := INT_TO_REAL(U_L2[i]) * 0.0225;
5.        U_L3real[i] := INT_TO_REAL(U_L3[i]) * 0.0225;
6.
7.        I_L1real[i] := INT_TO_REAL(I_L1[i]) * 0.000056 * 12;
8.        I_L2real[i] := INT_TO_REAL(I_L2[i]) * 0.000056 * 12;
9.        I_L3real[i] := INT_TO_REAL(I_L3[i]) * 0.000056 * 12;
10.
11.       P_L1[i] := U_L1real[i] * I_L1real[i];
12.       P_L2[i] := U_L2real[i] * I_L2real[i];
13.       P_L3[i] := U_L3real[i] * I_L3real[i]
14.
15. END_FOR
```

9.5 Timestamping in TwinCAT 3

```
1. //fbLocalTime(bEnable := TRUE);
2. IF NOT bInit
3. THEN
4.     dtBufferDT := SYSTEMTIME_TO_DT(fbLocalTime.systemTime);
5.     IF fbLocalTime.bValid
6.     THEN
7.         bInit := TRUE;
8.     END_IF
9. ELSE
10.     nTime := nTime + 1;
11.     tBufferTime := UINT_TO_TIME(nTime*10);
12.     IF tBufferTime = T#1S
13.     THEN
14.         //Add a second to your system time
15.         ntime := 0;
16.         nCalcBuffer := DT_TO_UDINT(dtBufferDT)+1;
17.         dtBufferDT := UDINT_TO_DT(nCalcBuffer);
18.         sLogTime := DT_TO_STRING(dtBufferDT);
19.         sLogTimeWithMs := sLogTime;
20.     ELSE
21.         //Add ms string time-stamp
22.         sMs := TIME_TO_STRING(tBufferTime);
23.         sLogTimeWithMs := CONCAT(sLogTime,sMs);
24.     END_IF
25. END_IF
```

9.6 Fast logging with data buffer in TwinCAT 3

Data sample

```
TYPE ST_Data :
STRUCT
    Timestamp        : LINT;
    fAM              : LREAL;
    fPeak            : LREAL;
    fPulse           : LREAL;
    fSawtooth        : LREAL;
    fSine            : LREAL;
    fSquare          : LREAL;
    fStairs          : LREAL;
    fTriangular      : LREAL;
END_STRUCT
END_TYPE
```

Extract from the function block FB_Record_tbl_Signals

("State Machine" => State: Recording)

```
bRecording := TRUE;
 //Fill buffer
stData[nWriteBufferIndex, nWriteIndex].Timestamp := nTimestamp;
stData[nWriteBufferIndex, nWriteIndex].fAM := fAM;
stData[nWriteBufferIndex, nWriteIndex].fPeak := fPeak;
stData[nWriteBufferIndex, nWriteIndex].fPulse := fPulse;
stData[nWriteBufferIndex, nWriteIndex].fSawtooth := fSawtooth;
stData[nWriteBufferIndex, nWriteIndex].fSine := fSine;
stData[nWriteBufferIndex, nWriteIndex].fSquare := fSquare;
stData[nWriteBufferIndex, nWriteIndex].fStairs := fStairs;
stData[nWriteBufferIndex, nWriteIndex].fTriangular := fTriangular;
//Set buffer index
nWriteIndex:=nWriteIndex+1;
IF nWriteIndex = 100 THEN
            nWriteIndex := 0;
             aWriteSQL[nWriteBufferIndex]:= TRUE;
            nWriteBufferIndex := nWriteBufferIndex + 1;
            IF nWriteBufferIndex = 20 THEN
                    nWriteBufferIndex := 0;
            END_IF
           IF aWriteSQL[nWriteBufferIndex] THEN
                  nState := 255;
                 RETURN;
          END_IF
       END_IF
```

9.7 Buffer in TwinCAT 3 using big array

```
VAR CONSTANT
      n                              : INT := 100; // oversampling factor
      samples             : INT := 4000;
END_VAR


FOR j:= 0 TO (samples - 1) BY 100 DO

      FOR i := 0 TO (n-1) DO

                //Spannung[V] (0.0225V/digit)
                U_L1real[i] := INT_TO_REAL(U_L1[i]) * 0.0225;
                U_L2real[i] := INT_TO_REAL(U_L2[i]) * 0.0225;
                U_L3real[i] := INT_TO_REAL(U_L3[i]) * 0.0225;


                //Big array

                 U_L1big[j + i] := U_L1real[i];
                 U_L2big[j + i] := U_L2real[i];
                 U_L3big[j + i] := U_L3real[i];


                //Strom[A] für FESTEN 1A Messbereich (56µA/digit)
                I_L1real[i] := INT_TO_REAL(I_L1[i]) * 0.000056 * 12;
                I_L2real[i] := INT_TO_REAL(I_L2[i]) * 0.000056 * 12;
                I_L3real[i] := INT_TO_REAL(I_L3[i]) * 0.000056 * 12;


                //Big array I

                 I_L1big[j + i] := I_L1real[i];
                 I_L2big[j + i] := I_L2real[i];
                 I_L3big[j + i] := I_L3real[i];

                //Momentanleistung[VA]
                P_L1[i] := U_L1real[i] * I_L1real[i];
                P_L2[i] := U_L2real[i] * I_L2real[i];
                P_L3[i] := U_L3real[i] * I_L3real[i];


                // Big   Momentanleistung[VA]
                P_L1big[j + i] := U_L1big[j + i] * I_L1big[j + i];
                P_L2big[j + i] := U_L2big[j + i] * I_L2big[j + i];
                P_L3big[j + i] := U_L3big[j + i] * I_L3big[j + i];

      END_FOR

END_FOR
```

9.8 Buffer in TwinCAT 3 using 1 Task

```
1. FOR i := 0 TO (n-1) DO
 2.       //Spannung[V] (0.0225V/digit)
 3.       U_L1real[i] := INT_TO_REAL(U_L1[i]) * 0.0225;
 4.       //Strom[A] für FESTEN 1A Messbereich (56µA/digit) ( *12 wegen Faktor Stromwandler)
 5.       //I_L1real[i] := INT_TO_REAL(I_L1[i]) * 0.000056 * 12;
 6.
 7.     IF (insertIndex + i) <= samples THEN
 8.         U_L1big[insertIndex + i] := U_L1real[i];
 9.     END_IF
10.
11.       IF i = (n-1) THEN
12.                 insertIndex := insertIndex + 100;
13.       END_IF
14.
15.     // Reset or wrap the index if it reaches the buffer limit
16.     IF (insertIndex + i) >= (samples - 1) THEN
17.         insertIndex := 0;
18.     END_IF
19.
20. END_FOR
```

10 Bibliography

[1] K. Morison, L. Wang and P. Kundur, "Power system security assessment," *IEEE power and energy magazine,* vol. 2, p. 30–39, 2004.

[2] M. L. Di Silvestre, S. Favuzza, E. R. Sanseverino and G. Zizzo, "How Decarbonization, Digitalization and Decentralization are changing key power infrastructures," *Renewable and Sustainable Energy Reviews,* vol. 93, p. 483–498, 2018.

[3] G. Anagnostou, F. Boem, S. Kuenzel, B. C. Pal and T. Parisini, "Observer-based anomaly detection of synchronous generators for power systems monitoring," *IEEE Transactions on Power Systems,* vol. 33, p. 4228–4237, 2018.

[4] S. Asefi, M. Mitrovic, D. Ćetenović, V. Levi, E. Gryazina and V. Terzija, "Power system anomaly detection and classification utilizing WLS-EKF state estimation and machine learning," *arXiv preprint arXiv:2209.12629,* 2022.

[5] M. Wadi and W. Elmasry, "An anomaly-based technique for fault detection in power system networks," in *2021 International Conference on Electric Power Engineering–Palestine (ICEPE-P)*, 2021.

[6] A. Prasad, J. B. Edward and K. Ravi, "A review on fault classification methodologies in power transmission systems: Part—I," *Journal of electrical systems and information technology,* vol. 5, p. 48–60, 2018.

[7] H. R. Baghaee, D. Mlakić, S. Nikolovski and T. Dragicević, "Support vector machine-based islanding and grid fault detection in active distribution networks," *IEEE Journal of Emerging and Selected Topics in Power Electronics,* vol. 8, p. 2385–2403, 2019.

[8] A.-D. Gonzalez-Abreu, M. Delgado-Prieto, R.-A. Osornio-Rios, J.-J. Saucedo-Dorantes and R.-d.-J. Romero-Troncoso, "A novel deep learning-based diagnosis method applied to power quality disturbances," *Energies,* vol. 14, p. 2839, 2021.

[9] R. A. de Oliveira and M. H. J. Bollen, "Deep learning for power quality," *Electric Power Systems Research,* vol. 214, p. 108887, 2023.

[10] U. Halden, U. Cali and F. O. C. S. D'Arcoy, Anomaly Detection in Power Markets and Systems, University of Stavanger, 2022.

[11] C. Chahla, H. Snoussi and L. e. a. Merghem, "A deep learning approach for anomaly detection and prediction in power consumption data.," *Energy Efficiency 13, 1633–1651,* 2020.

[12] S.-H. A. Xinlin Wang, "Real-time prediction and anomaly detection of electrical load in a residential community," *Applied Energy,* 2020.

[13] S. M. H. Rizvi, S. K. Sadanandan and A. K. Srivastava, "Data-driven short-term voltage stability assessment using convolutional neural networks considering data anomalies and localization," *IEEE Access,* vol. 9, p. 128345–128358, 2021.

[14] M. Ali, A. Ateem, Z. A. Akbar and M. A. Bashir, "Detecting and monitoring of voltage and frequency variation and under ground cable fault location using check point method," in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2018.

[15] C.-W. Ten, J. Hong and C.-C. Liu, "Anomaly detection for cybersecurity of the substations," *IEEE Transactions on Smart Grid,* vol. 2, p. 865–873, 2011.

[16] S. Asefi, M. Mitrovic, D. Ćetenović, V. Levi, E. Gryazina and V. Terzija, "Anomaly detection and classification in power system state estimation: Combining model-based and data-driven methods," *Sustainable Energy, Grids and Networks,* vol. 35, p. 101116, 2023.

[17] S. W. Yen, S. Morris, M. A. G. Ezra and T. J. Huat, "Effect of smart meter data collection frequency in an early detection of shorter-duration voltage anomalies in smart grids," *International journal of electrical power & energy systems,* vol. 109, p. 1–8, 2019.

[18] M. Mozaffari, K. Doshi and Y. Yilmaz, "Real-time detection and classification of power quality disturbances," *Sensors,* vol. 22, p. 7958, 2022.

[19] "twincat-3 licensing," [Online]. Available: https://www.beckhoff.com/en-en/products/automation/twincat/twincat-3-licensing/.

[20] IEEE, *IEEE Guide for Voltage Sag Indices,* IEEE Piscataway, NJ, USA, 2014, p. 1–59.

[21] "IEEE Recommended Practice for Monitoring Electric Power Quality," *IEEE Std 1159-2019 (Revision of IEEE Std 1159-2009),* pp. 1-98, 2019.

[22] "IEEE Recommended Practice for Monitoring Electric Power Quality," *IEEE Std 1159-1995,* pp. 1-80, 1995.

[23] S. Lehmann, "pyADS Docs: Read write values via pyADS," 2024. [Online]. Available: https://pyads.readthedocs.io/en/latest/documentation/connection.html#read-and-write-by-name.

[24] Beckhoff, "FB_MemRingBuffer," [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_utilities/35010187.html&id=.

[25] Beckhoff, "Fast logging with data buffer," [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tf6420_tc3_database_server/6263315851.html&id=.

[26] H. Wickham, R Packages, 1st ed., O'Reilly Media, Inc., 2015.

[27] S. Soltan, D. Mazauric and G. Zussman, "Analysis of Failures in Power Grids," *IEEE Transactions on Control of Network Systems,* vol. 4, pp. 288-300, 2017.

[28] M. Jahromi, S. Meshksar, E. Farjah and M. Zolghadri, "Voltage Sag State Estimation For Power Distribution Systems Using Kalman Filter," in *2007 IEEE International Symposium on Industrial Electronics*, 2007.

[29] GridsMini S. Thomas, John D. McDonaldK, Power Systems SCADA and Smart Grids, 1 ed., 6000 Broken Sound Parkway NW, Suite 300: Taylor & Francis Group, 2015.