

CyberGrid Guardian: Implementing a Distance Protection Algorithm for Transmission Lines Using Java

Dissertation submitted to Shri Ramdeobaba College of Engineering and
Management, Nagpur in partial fulfillment of requirement for the award of
degree of

Bachelor of Technology

In

ELECTRICAL ENGINEERING

By

**Atharva Sandip Joshi
Taniya Sanjay Jaiswal
Jayesh Santosh Dharmale
Rohan Bhagchand Kukreja**

Guide

Dr. Gaurav Goyal



Electrical Engineering

**Shri Ramdeobaba College of Engineering and Management,
Nagpur 440013**

An Autonomous Institute affiliated to Rashtrasant Tukadoji Maharaj,
Nagpur University Nagpur

October 2023

CyberGrid Guardian: Implementing a Distance Protection Algorithm for Transmission Lines Using Java

Dissertation submitted to Shri Ramdeobaba College of Engineering and
Management, Nagpur in partial fulfillment of requirement for the award of
degree of

Bachelor of Technology

In

ELECTRICAL ENGINEERING

By

**Atharva Sandip Joshi
Taniya Sanjay Jaiswal
Jayesh Santosh Dharmale
Rohan Bhagchand Kukreja**

Guide

Dr. Gaurav Goyal



Electrical Engineering

**Shri Ramdeobaba College of Engineering and Management,
Nagpur 440013**

(An Autonomous Institute affiliated to Rashtrasant Tukadoji Maharaj Nagpur
University Nagpur)

October 2023

**Shri Ramdeobaba College of Engineering and Management,
Nagpur 440013**

(An Autonomous Institute affiliated to Rashtrasant Tukadoji Maharaj Nagpur
University Nagpur)

Department of Electrical Engineering

CERTIFICATE

This is to certify that the Thesis on “**CyberGrid Guardian: Implementing a Distance Protection Algorithm for Transmission Lines Using Java**” is a **bonafide** work of **Atharva Sandip Joshi** submitted to Shri Ramdeobaba College of Engineering and Management Nagpur, in partial fulfillment of Final Year Project during the academic year 2023-2024.

I would also like to acknowledge the collaboration of my fellow group members, **Taniya Sanjay Jaiswal, Jayesh Santosh Dharmale, and Rohan Bhagchand Kukreja**, who provided valuable input and support throughout the project. Their contributions in discussions have been appreciated.

Date: 17/11/2023

Place: Nagpur (M.H), IN

Dr. Gaurav Goyal

Project Guide

Department of Electrical Engineering

Dr. V.A. Huchche

Project Co-Ordinator

Department of Electrical Engineering

Dr. Sanjay B. Bodkhe

H.O.D.

Department of Electrical Engineering

Dr. R.S. Pande

Principal

Shri Ramdeobaba College of Engineering
and Management

DECLARATION

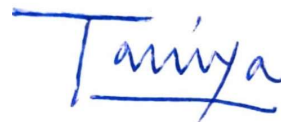
I, Hereby declare that the thesis titled “**CyberGrid Guardian: Implementing a Distance Protection Algorithm for Transmission Lines Using Java**” submitted herein, has been carried out in the Department of Electrical Engineering of Shri Ramdeobaba College of Engineering and Management, Nagpur. The work is original and has not been submitted earlier as a whole or part for the award of any degree/diploma at this or other Institution/University

Date: 17/11/2023

Place: Nagpur (M.H), IN



Atharva Sandip Joshi
Group Leader
Department of Electrical Engineering



Taniya Sanjay Jaiswal
Group Member
Department of Electrical Engineering



Jayesh Santosh Dharmale
Group Member
Department of Electrical Engineering



Rohan Bhagchand Kukreja
Group Member
Department of Electrical Engineering

APPROVAL SHEET

This Thesis entitled “**CyberGrid Guardian: Implementing a Distance Protection Algorithm for Transmission Lines Using Java**” at Shri Ramdeobaba College of Engineering and Management, Prepared and submitted by Atharva Sandip Joshi(31), Taniya Jaiswal(57), Jayesh Dharmale(71), Rohan Kukreja(83) is approved for the degree of Electrical Engineering.

Date: 17/11/2023

Place: Nagpur (M.H), IN

Name & Signature of Supervisor

Name & Signature of Examiner(s)

Dr. Gaurav Goyal

Name & Signature of RRC Members

Name & Signature of H.O.D

1. Dr. Uday B. Mujumdar

Dr. Sanjay B. Bodkhe

2. Dr. Sonali Rangari

Acknowledgement

We would like to express our sincere gratitude to the numerous individuals of the **Department of Electrical Engineering (RCOEM)** who have supported us throughout the journey of completing this final year engineering project thesis. Their invaluable contributions, guidance, and encouragement have been instrumental in making this endeavor a reality.

I extend my deepest appreciation to my **guide, Dr. Gaurav Goyal**, for his unwavering support and mentorship.

I am also indebted to the faculty members of the Shri Ramdeobaba College of Engineering and Management, Nagpur for their dedication to fostering an environment of learning and research. Their teachings have not only enriched my knowledge but have also inspired me to explore new horizons in engineering.

In addition, Special thanks **Dr. U.B. Mujumdar** and **Dr. S.C. Rangari** for their constant feedback and contributions to the improvement of my project. Their insights and guidance have been crucial in refining this work.

In conclusion, this project represents the collective effort of many, and I am deeply thankful for the encouragement, guidance, and inspiration I have received. Their support has been integral to the successful completion of this engineering project thesis.

Abstract

Transmission Lines are an important part of the power systems. It has various components like generators, transformers, circuit breakers, isolators, bus bars, and many more. Transmission lines have high power transfer capacity and are hence prone to various faults. Therefore, a power system protection scheme is necessary. The faulty section is isolated from the healthy section by the protection system because there are chances of power system equipment getting damaged due to the fault conditions. Relays are used where it is necessary to control a circuit by an independent low-power signal, or where several circuits must be controlled by one signal. A numerical relay is a computer-based system with software-based protection algorithms for the detection of electrical faults in power systems. Nowadays relays are simulated in a simple assembly language. In this project, the relay is simulated using high level language “Java”. Many algorithms or techniques are used for the simulation of relays such as the Mann-Morrison algorithm, Three sampling techniques, DFT (Discrete Fourier transform), etc. This report presents a time comparison for fault detection between Mann Morrison Technique and the Three Sampling Technique. Also, it contains the results for implementing the above algorithms for distance relay using Arduino/ microcontroller.

Index

Cover Page.....	ii
Certificate.....	iii
Declaration.....	iv
Approval Sheet.....	v
Acknowledgements.....	vi
Abstract.....	vii
List of Figures.....	ix
List of Tables.....	x
1. Chapter 1 : Introduction	x
1.1 Introduction	1
1.2 Transmission Line	3
1.3 Relays and Types of Relays	5
1.4 Objective and Motivation	8
1.5 Conclusion	9
2. Chapter 2 : Distance Protection Algorithm	10
2.1 Introduction	10
2.2 Mann Morrison Algorithm	11
2.3 Conclusion	15
3. Chapter 3 : Implementation of Distance Protection Algorithm using Java	17
3.1 Introduction	17
3.2 Features of software	17
3.3 Methodology of program algorithm	19
3.4 Time complexity	20
3.5 Database : MySQL	23
3.6 Results	27
4. Chapter 4 : Hardware Implementation of Distance Protection Algorithms ...	32
4.1 Introduction	32
4.2 Microcontroller	32
4.3 Design of Hardware Model	34
4.4 Data collection through serial communication	38
4.5 Data collection through ESP32 Module	40
4.6 Conclusion	43
5. Chapter 5 : Conclusion and Future Scope	44
5.1 Introduction	44
5.2 Conclusion	44
5.3 Future Scope	44
6. Appendices	46
7. References.....	59
8. Project submission	61
9. Personal Details.....	62

List of figures

1. Chapter 1: Introduction	1
1.1 Single line diagram of a Transmission Line	4
1.2 A typical relay circuit	5
1.3 Classification of Numerical Relays	8
2. Chapter 2 : Distance Protection Algorithms	9
2.1 Mann Morrison Algorithm working	15
3. Chapter 3 : Implementation of Distance Protection Algorithms Using Java ...	17
3.1 Methodology of program algorithm	19
3.2 Linear Time Complexity.....	23
3.3 New table is created for each file	26
3.4 Structure of table in database	26
3.5 Data in table	27
3.6 GUI of the developed software	27
3.7 No Over Current fault detected using Mann Morrison Tech.	28
3.8 Graph showing no over current fault.....	28
3.9 Uploading custom CSV file	29
3.10 Over Voltage fault detected using Mann Morrison Tech.	29
3.11 Updated database with fault value calculated using Mann Morrison	30
3.12 Graph showing over voltage fault in blue line	30
3.13 Output of the algorithm in console	31
4. Chapter 4 : Hardware Implementation of Distance Protection Algorithms	32
4.1 Arduino UNO	33
4.2 The Hardware setup for the project	34
4.3 Pin Diagram of Atmega 32 Microcontroller	35
4.4 Block diagram of circuit	36
4.5 Block diagram of Data Acquisition System	38
4.6 Output of Arduino UNO showing different parameters	40
4.7 ESP32 Module	40
4.8 Output of ESP32 Module on google spreadsheet	39

List of Tables

2.	Chapter 2 : Distance Protection Algorithms	10
2.1	Formulae used in Mann Morrison Algorithm	14

Chapter 1

Introduction

1.1 Introduction

Transmission lines are a critical component of the power system, consisting of several interconnected components like generators, transformers, circuit breakers, isolators, and bus bars, among others. These lines are capable of transferring high levels of power, making them vulnerable to various faults. Therefore, a robust power system protection scheme is crucial to safeguard the equipment from damage under fault conditions.

Numerical relays are computer-based systems that use software-based protection algorithms to detect electrical faults in power systems. These relays are now simulated in high-level language like “Java” and SQL queries, which offer several advantages over traditional programming languages. Various algorithms and techniques, including the Mann-Morrison algorithm, Three Sample Technique, and the Discrete Fourier transform (DFT), are used to simulate relays. These techniques are used to detect and isolate faults in the transmission lines, enabling the protection system to act quickly and prevent any damage to the equipment.

The field of digital protection has seen the development of numerous algorithms proposed in the literature. These algorithms are designed to be highly efficient, provided the assumptions made in their development hold in practice. Simple yet effective algorithms like the Mann- Morrison algorithm, the Three Sampling Algorithm, and the Discrete Fourier Transform (DFT) are commonly used in digital protection systems. Despite their simplicity, these algorithms are capable of effectively detecting and isolating faults in transmission lines. Overall, the use of these algorithms in digital protection systems has significantly improved the reliability and efficiency of power systems. This paper consists of the Java program required for the implementation of the Mann Morrison Algorithm and Three Sampling Algorithm.

In utility and industrial electric power transmission and distribution systems, a numerical relay is a computer-based system with software-based protection algorithms for the detection of electrical faults. Such relays are also termed microprocessor-type protective relays. Numerical protection relays are digital systems in constant communication with substation automation systems through menu-driven interfaces. They have configurable binary inputs, outputs, and programmable logic. They monitor, measure, and record electrical values, faults and disturbances, and events. The primary motivation for undertaking this research is the fact that, in current times, numerical relays are often simulated using assembly language. Specifically, distance protection relays are a critical component of transmission lines, and they are often programmed using assembly language. Assembly language is a low-level language hence time-consuming to write and requires in-depth knowledge of the underlying hardware.

However, there is a notable lack of literature on the simulation of relays using high-level language like Java and Python. These languages offer several advantages over assembly language. For instance, they provide more intuitive and user-friendly syntax, which makes program cleaner and simple. Additionally, these languages provide access to libraries that enables in implementing features like graph plotting, maintaining databases. To address this gap in the literature, we have attempted to develop a distance protection scheme using Java and MySQL, which is a widely-used, programming language. By leveraging the power and flexibility of Java, we aim to demonstrate that it is possible to simulate numerical relays using a high-level programming language. This will provide a more accessible and efficient approach to relay simulation, allowing for faster and more accurate analysis of transmission lines and power systems. Overall, this research represents an important step toward in the development of more advanced and accessible numerical relays. By leveraging the advantages of high-level object-oriented programming languages such as Java, we hope to contribute to the ongoing effort to improve the safety and reliability of power systems around the world.

1.2 Transmission Line

A transmission line is an electrical circuit used to transfer power or signals from one location to another. It is primarily used for transmitting high voltage electrical power from power plants to distribution substations, where the voltage is stepped down for local distribution. There are several types of transmission lines, including:

- Overhead lines: These are the most common type of transmission lines and consist of conductors supported on towers or poles. They are used for high voltage power transmission over long distances.
- Underground lines: These are used in urban areas or for short distances where overhead lines are not feasible. They consist of insulated cables buried underground.
- Submarine lines: These are used for transmitting power between islands or across bodies of water. They are similar to underground lines, but the cables are designed to withstand the harsh underwater environment.
- HVDC lines: These are high-voltage direct current lines that are used for long-distance power transmission. They are more efficient than AC lines for transmitting power over long distances and can be used for interconnecting power grids.
- Microstrip lines: These are used in high-frequency circuits for transmitting signals. They consist of a thin conductive strip on a substrate and are used in applications such as antennas, radar, and satellite communication. Overall, transmission lines play a critical role in delivering power and signals over long distances and are essential for modern communication and energy systems. Transmission lines can be classified based on their lengths in kilometers(km) into three main categories:
 - Short transmission lines: These are transmission lines with a length up to 80 km, and they are usually used to connect nearby substations to distribute electricity to consumers.
 - Medium transmission lines: These are transmission lines with a length ranging from 80 km to 250 km, and they are typically used to connect power plants to

local distribution networks or to transmit electricity over relatively short distances.

- Long transmission lines: These are transmission lines with a length greater than 250 km, and they are typically used to transmit power over very long distances, such as between different regions or countries. It's worth noting that the classification of transmission lines based on their length is not strictly defined, and there may be variations in the range of each category depending on the context or application.

The single line diagram of a long transmission line as shown in Fig.1.1 where distance protection is a necessary component. Distance protection is a type of protection scheme used in power transmission systems to detect and isolate faults on transmission lines. The distance protection scheme operates by measuring the impedance of the transmission line from the point where the protection relay is installed to the location of the fault. The impedance is the total opposition offered by the transmission line to the flow of electrical current. If a fault occurs on the transmission line, the impedance of the line changes and the distance relay compares the measured impedance with a predetermined set of impedance characteristics. If the measured impedance falls within the set of impedance characteristics, the distance relay sends a tripping signal to the circuit breaker to isolate the faulted section of the transmission line.

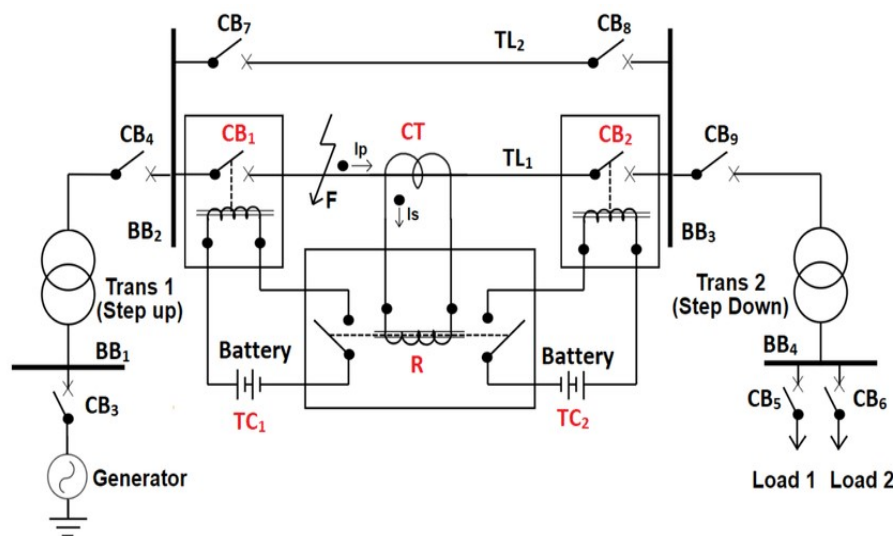


Fig 1.1: Single line diagram of a transmission line

Distance protection is widely used because it is simple, reliable, and cost-effective. It can detect faults accurately even in situations where the fault location is not precisely known. The distance protection scheme can be set to provide different levels of protection depending on the distance from the relay to the fault.

1.3 Relay and Types of Relays

Relays are essential components in electrical systems that help to control and protect the circuits they are attached to. They work by using a small electrical signal to trigger a switch, which can then control a larger electrical load. There are several different types of relays available, each with its unique design and purpose. One of the most common types of relays is the electromagnetic relay.

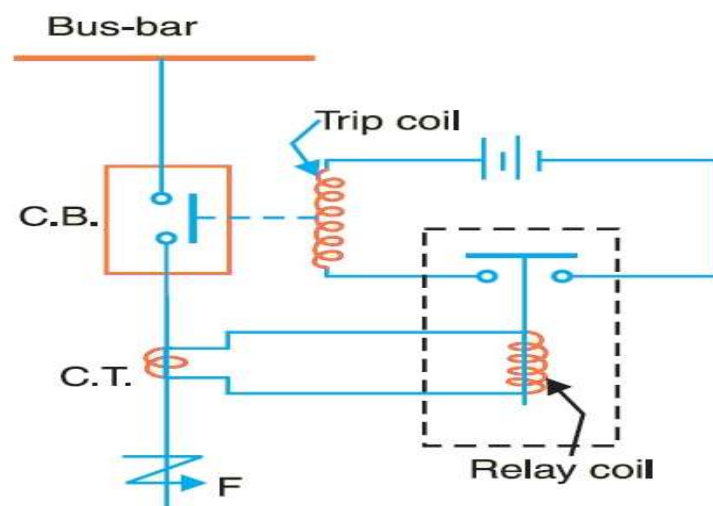


Fig 1.2 : Typical Relay Circuit

These relays consist of a coil of wire that produces a magnetic field when an electrical current passes through it. This magnetic field, in turn, attracts an iron or steel armature, which is connected to a set of contacts. When the armature is attracted, it closes the contacts, completing an electrical circuit. Electromagnetic relays are reliable and durable, making them ideal for use in a wide range of applications, from automotive to aerospace. Another type of relay is the solid-state relay. Solid-state relays use semiconductor devices, such as thyristors or transistors, to switch electrical loads. These relays have no moving parts, making them extremely reliable and long-lasting. They also have fast switching

speeds, which make them ideal for applications that require high- frequency switching, such as in power electronics.

There are also differential relays, which are used to detect faults in electrical systems. These relays compare the current flowing into a circuit with the current flowing out of it. If there is a difference between these two currents, it indicates that there is a fault in the circuit. Differential relays are commonly used in power systems to detect faults in transmission lines, generators, and transformers. Other types of relays include thermal relays, which use a bimetallic strip to detect changes in temperature, and reed relays, which use a magnet to activate a set of contacts. Each type of relay has its own unique strengths and weaknesses, making it suitable for particular applications. In conclusion, relays play a critical role in electrical systems, enabling the control and protection of circuits. There are several different types of relays, each designed to meet specific requirements, from high-frequency switching to fault detection. Choosing the right type of relay for a particular application requires careful consideration of factors such as reliability, durability, and switching speed, as well as the specific needs of the system being controlled or protected. Numerical relays can be classified based on their applications, functionality, and operating principles. Here are some common classifications of numerical relays:

- Application-based classification: Feeder protection relays, Transformer protection relays, Motor protection relays, Generator protection relays, Busbar protection relays, Line differential protection relays, Distance protection relays.
- Functionality-based classification: Overcurrent relays, Earth fault relays, Directional relays, Differential relays, Distance relays, Synchronizing relays, Breaker failure relays, Under/Overvoltage relays.
- Operating principle-based classification: Microprocessor-based relays, Static relays, Electromechanical relays, Hybrid relays.

Each type of numerical relay has its own specific features, advantages, and limitations, and the choice of the right relay depends on the particular application requirements. Distance relays are a type of protective relay that are widely used in power systems to detect faults in transmission lines. These relays

work by measuring the impedance of the line and comparing it to a pre-set threshold. If the impedance exceeds the threshold, the relay will activate and send a trip signal to the circuit breaker, disconnecting the faulty section of the line from the rest of the system. One of the primary advantages of distance relays is their ability to detect faults accurately and quickly over long distances. They are also relatively simple to install and maintain, making them a popular choice for transmission line protection.

There are several different types of distance relays, each with its own unique design and capabilities. For example, mho relays are a popular type of distance relay that are often used in high-speed protection applications. These relays use a circular impedance characteristic, which allows them to operate quickly and accurately, even in the presence of power swings or other disturbances. Another type of distance relay is the quadrilateral characteristic relay, which is commonly used in applications where high accuracy and selectivity are required. These relays use a four-sided characteristic, which allows them to provide accurate distance measurements even in complex power systems with multiple transmission lines and sources.

Despite their many advantages, distance relays are not without their limitations. For example, they can be affected by changes in load or system topology, which can cause false trips or delays in fault detection. To address these issues, newer generations of distance relays are incorporating advanced algorithms and machine learning techniques to improve their accuracy and resilience to external disturbances. Overall, distance relays are an essential component of modern power systems, enabling the safe and reliable transmission of electricity over long distances. With ongoing advancements in technology and engineering, these relays are set to become even more sophisticated and effective in the years to come.

Types of Numerical Relay Based on Protection Scheme:

- Overcurrent / Earth fault relay. Differential relay.
- Directional relay. Under / Overvoltage relay. Distance relay.

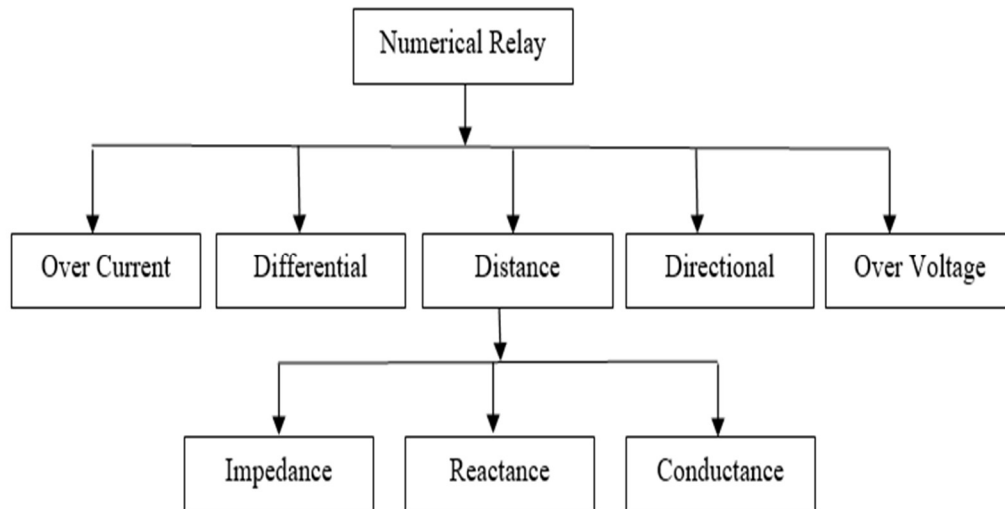


Fig 1.3 : Classification of Numerical relays

1.4 Objective and Motivation

The protection of transmission lines from any malfunction leading to short circuit or fires or burns in long transmission lines is of great importance, and an important role is played by distance protection. For this purpose, distance relays and distance protection algorithms are used. It is aimed in this dissertation to simulate a Numerical Relay in Transmission Line by using algorithms like Mann Morrison and Three Sample Technique. To accomplish this objective, an Assembly Language is used to program and devise the Numerical Relay, particularly the distance relay. However, there is no relevant information or formal literature found regarding the Simulation of Numerical Relay using Java. Furthermore, there is no relevant data or literature work available that shows the comparison of the Algorithms or techniques used in fault detection and simulation.

The developed software **maintains a database using MySQL** that stores information including but not limited to **instantaneous voltage, instantaneous current, overvoltage and overcurrent values, time of fault, and reason of fault** it also **enables user to plot graphs, upload custom CSV files for analysis**. It provides the user a interactive and **easy to operate Graphical User Interface (GUI)**.

1.5 Conclusion

Transmission lines play a crucial role in power transmission and their protection from accidents like short circuits, fires, rise in voltage or current etc. is a major concern. Relays act as a switch and are used in protection of long head transmission lines. Various types of relays based on their functionality are there in which for long head transmission line distance relays are one of the effective and useful ones.

Chapter 2

Distance Protection Algorithm

2.1 Introduction

Distance protection algorithms are a set of mathematical equations and algorithms used in distance protection schemes implemented in distance relays. These algorithms enable the distance relay to accurately determine the location of a fault in the transmission line by measuring the impedance between the relay and the fault. There are several distance protection algorithms commonly used in distance relays, including the impedance-based algorithm, the reactance-based algorithm, and the apparent impedance algorithm. Each algorithm has its own unique characteristics and advantages, depending on the specific requirements of the application. The impedance-based algorithm is the most commonly used distance protection algorithm, and it works by comparing the measured impedance of the transmission line to a pre-set impedance characteristic curve. If the measured impedance exceeds the curve, the distance relay will send a trip signal to the circuit breaker, isolating the faulty section of the line.

The reactance-based algorithm is similar to the impedance-based algorithm, but it calculates the reactance of the transmission line instead of its impedance. This algorithm is particularly useful in situations where the line is heavily loaded, as it can compensate for the effects of load impedance on the distance measurement. The apparent impedance algorithm is a more advanced algorithm that takes into account the effects of the fault current on the impedance measurement. By compensating for the effects of fault current, this algorithm can provide more accurate distance measurements, reducing the risk of false trips or delayed fault detection. Overall, the choice of distance protection algorithm depends on a range of factors, including the specific application, the characteristics of the transmission line, and the performance requirements of the system. By implementing the appropriate distance protection algorithm, distance relays can provide reliable and effective protection against faults in transmission lines, ensuring the safe and reliable operation of the power system. Various distance protection algorithms used in transmission line protection.

Some of the most common algorithms are:

- Impedance-based distance protection: This algorithm measures the impedance of the transmission line to detect and locate faults. The impedance is calculated from the measured voltage and current phasors using complex arithmetic. The algorithm compares the calculated impedance with a set of impedance characteristics to determine the fault location.
- Reactance-based distance protection: This algorithm is based on the measurement of the reactance of the transmission line. The reactance is calculated from the measured voltage and current phasors and is used to determine the fault location.
- Mho distance protection: This algorithm uses a mho characteristic to detect and locate faults. The mho characteristic is a circle on the impedance plane with the center at the relay location and the radius set to the reach of the protection zone. When a fault occurs within the protection zone, the measured impedance falls within the mho characteristic, and the relay operates to isolate the fault.
- Quadrilateral distance protection: This algorithm uses a quadrilateral characteristic on the impedance plane to detect and locate faults. The quadrilateral characteristic is a polygon with four vertices that define the protection zone. When a fault occurs within the protection zone, the measured impedance falls within the quadrilateral characteristic, and the relay operates to isolate the fault.
- Advanced distance protection: This algorithm uses advanced techniques such as digital signal processing, wavelet transforms, and artificial intelligence to improve the accuracy and reliability of the distance protection scheme. These techniques can enhance fault detection and location accuracy and reduce the risk of false tripping.

2.2 Mann Morrison Algorithm

The Mann-Morrison algorithm is a specific implementation of the mho distance protection algorithm. It is named after its developers, Robert Mann and R.J. Morrison, who published a paper on the algorithm in 1971. The Mann-Morrison algorithm uses a circular mho characteristic on the impedance plane to detect

and locate faults on transmission lines. The mho characteristic is defined by its center at the relay location and the radius of the circle, which is set to the reach of the protection zone. When a fault occurs within the protection zone, the measured impedance falls within the mho characteristic, and the relay operates to isolate the fault.

The algorithm operates in the following steps:

- Calculate the measured impedance using the voltage and current phasors measured at the relay location.
- Convert the measured impedance to the impedance plane using complex arithmetic.
- Check if the measured impedance falls within the mho characteristic. If it does, then the relay sends a tripping signal to the circuit breaker to isolate the fault.

The Mann-Morrison algorithm is widely used in transmission line protection due to its simplicity and reliability. It is particularly useful in protecting long transmission lines where accurate fault location is not critical, and fast fault detection is more important. Thus, voltage samples collected by the relay are assumed to belong to a signals of the type:

Thus, voltage samples collected by the relay are assumed to belong to a signals of the type:

$$v(tk) = Vm \sin(\omega tk + \theta v) \dots \dots \dots (2.1)$$

Where we assume Vm , ω and θv to be constant. Further we assume that the frequency ' ω ' is known. Thus in the above equations we have two unknowns, namely the peak value of the signal and its phase angle. Hence, we need one more equation to solve for the unknowns. Let us generate another equation by taking derivative of the first equation giving:

$$v'(tk) = \omega Vm \cos(\omega tk + \theta v) \dots \dots \dots (2.2)$$

from which we get:

$$V \frac{v'(tk)}{\omega} = Vm \cos(\omega tk + \theta v) \dots \dots \dots (2.3)$$

Combining Eqs. (2.1) and (2.3), we get:

$$v^2 m = (v(tk))^2 + \left(\frac{v'(tk)}{\omega} \right)^2 \dots \dots \dots (2.4)$$

Hence, we can find the amplitude of voltage signal as:

$$Vm = \sqrt{(v(tk))^2 + \left(\frac{v'(tk)}{\omega} \right)^2} \dots \dots \dots (2.5)$$

Similarly, we can write for the current signal as:

$$i(tk) = Im \sin(\omega tk + \theta i) \dots \dots \dots (2.6)$$

Differentiating Eq (2.6) at instant tk we get:

$$i'(tk) = \omega Im \cos(\omega tk + \theta i) \dots \dots \dots (2.7)$$

Combining Eqs (2.6) and (2.7) we get:

$$Im^2 = (i(tk))^2 + \left(\frac{i'(tk)}{\omega} \right)^2 \dots \dots \dots (2.8)$$

Hence, We can find the amplitude of current signal as:

$$Im = \sqrt{(i(tk))^2 + \left(\frac{i'(tk)}{\omega} \right)^2} \dots \dots \dots (2.9)$$

Dividing Eq (2.1) by Eq (2.3) we get:

$$\theta_v = \tan^{-1} \left(\frac{v(tk)}{\left(\frac{v'(tk)}{\omega} \right)} \right) - \omega t_k \dots \dots \dots (2.10)$$

The Derivatives can be computed numerically using the central difference formula as :

$$V'(tk) = \frac{V_{(k+1)} - V_{(k-1)}}{2 \cdot \Delta t} \dots \dots \dots (2.11)$$

$$I'(tk) = \frac{I_{(k+1)} - I_{(k-1)}}{2 \cdot \Delta t} \dots \dots \dots (2.12)$$

Thus Computation of one derivate requires three samples, we have to keep estimating the peak and the phase angle and hence the phasor.

Model of voltage signal	$v(tk) = Vm \sin(\omega tk + \theta_v)$
Model of current	$i(tk) = Im \sin(\omega tk + \theta_i)$
Vm	$Vm = \sqrt{(v_{(tk)})^2 + \left(\frac{v'_{(tk)}}{\omega} \right)^2}$ $V'(tk) = \frac{V_{(k+1)} - V_{(k-1)}}{2 \cdot \Delta t}$
θ_v	$\tan^{-1} \left(\frac{v(tk)}{\left(\frac{v'(tk)}{\omega} \right)} \right) - \omega t_k$
Im	$\sqrt{(i_{(tk)})^2 + \left(\frac{i'_{(tk)}}{\omega} \right)^2}$ $I'(tk) = \frac{I_{(k+1)} - I_{(k-1)}}{2 \cdot \Delta t}$
θ_i	$\tan^{-1} \left(\frac{v(tk)}{\left(\frac{v'(tk)}{\omega} \right)} \right) - \omega t_k$

Table 2.1: Formulae derived by Mann Morrison

The Mann-Williamson technique (also known as the "Mann-Morrison" technique) is a commonly used method for analyzing power system faults and designing protective relays for power system protection. The technique is based on the concept of fault impedance, which is the ratio of the voltage and current at a fault location. The fault impedance varies depending on the location and type of fault and can be used to determine the optimal settings for protective relays. The formulae for Mann-Morrison algorithm are shown in Table 1.1 above. Voltage samples collected by the relay are assumed to belong to a signal of the type $v(tk) = V_m \sin(\omega tk + \theta_v)$, where we assume V_m , ω and θ_v to be constant. Current samples collected by the relay are assumed to belong to a signal of the type $i(tk) = I_m \sin(\omega tk + \theta_i)$, where we assume I_m , ω and θ_i to be constant and k represents the index of the sample. V_m is the maximum voltage and I_m is the maximum current while θ_v and θ_i are the angles in degrees. In this algorithm, we model the signal in a very simplistic manner. It is assumed that the signal can be represented as a pure sinusoid whose amplitude as well as frequency is constant during the period under consideration as shown in Figure

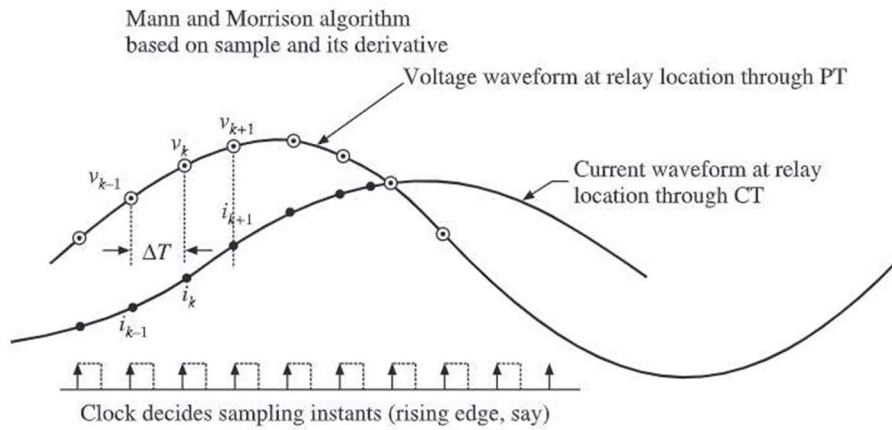


Fig 2.1 Mann Morrison Algorithm

2.3 Conclusion

After studying and comparing the Mann-Morrison and Three Sampling techniques for fault detection in power transmission lines, it can be concluded that both techniques have their strengths and weaknesses. The Mann-Morrison technique is more accurate in detecting faults near the relaying point, but it is

less accurate in detecting faults at the far end of the transmission line. On the other hand, the Three Sampling technique is more accurate in detecting faults at the far end of the transmission line, but it requires more hardware and is more complicated to implement. Therefore, the choice of technique depends on the specific requirements of the transmission system and the priorities of the stakeholders. Further research can be conducted to improve the accuracy of fault detection techniques and to optimize their implementation in power transmission systems.

Chapter 3

Implementation of Distance Protection Algorithm Using Java.

3.1 Introduction

The motive of this project is to address the lack of significant literature available regarding the simulation of distance relays and the implementation of distance protection algorithms using Java. Currently, these tasks are accomplished using assembly language, which has several disadvantages. By contrast, Java offers several advantages over assembly language, including smoother and more efficient processes, less time consumption, and quicker fault detection, leading to enhanced equipment protection. Therefore, the implementation of distance protection algorithms using Java has been discussed in this dissertation. It is worth noting that the implementation of distance protection algorithms using Java has not been widely explored in literature, making this dissertation a valuable contribution to the field. The results of this study could be used as a basis for further research aimed at improving the accuracy and efficiency of distance protection algorithms in power transmission systems.

3.2 Features of Software

The software "CyberGrid Guardian" represents a numerical relay system meticulously developed for the final year project. It serves as a comprehensive and robust solution designed to detect faults in transmission systems in real-time, providing a user-friendly interactive graphical user interface (GUI) for efficient control and analysis. The software deploys sophisticated fault detection techniques, including the Mann-Morrison and Three Sample methods, empowering users to make critical decisions promptly.

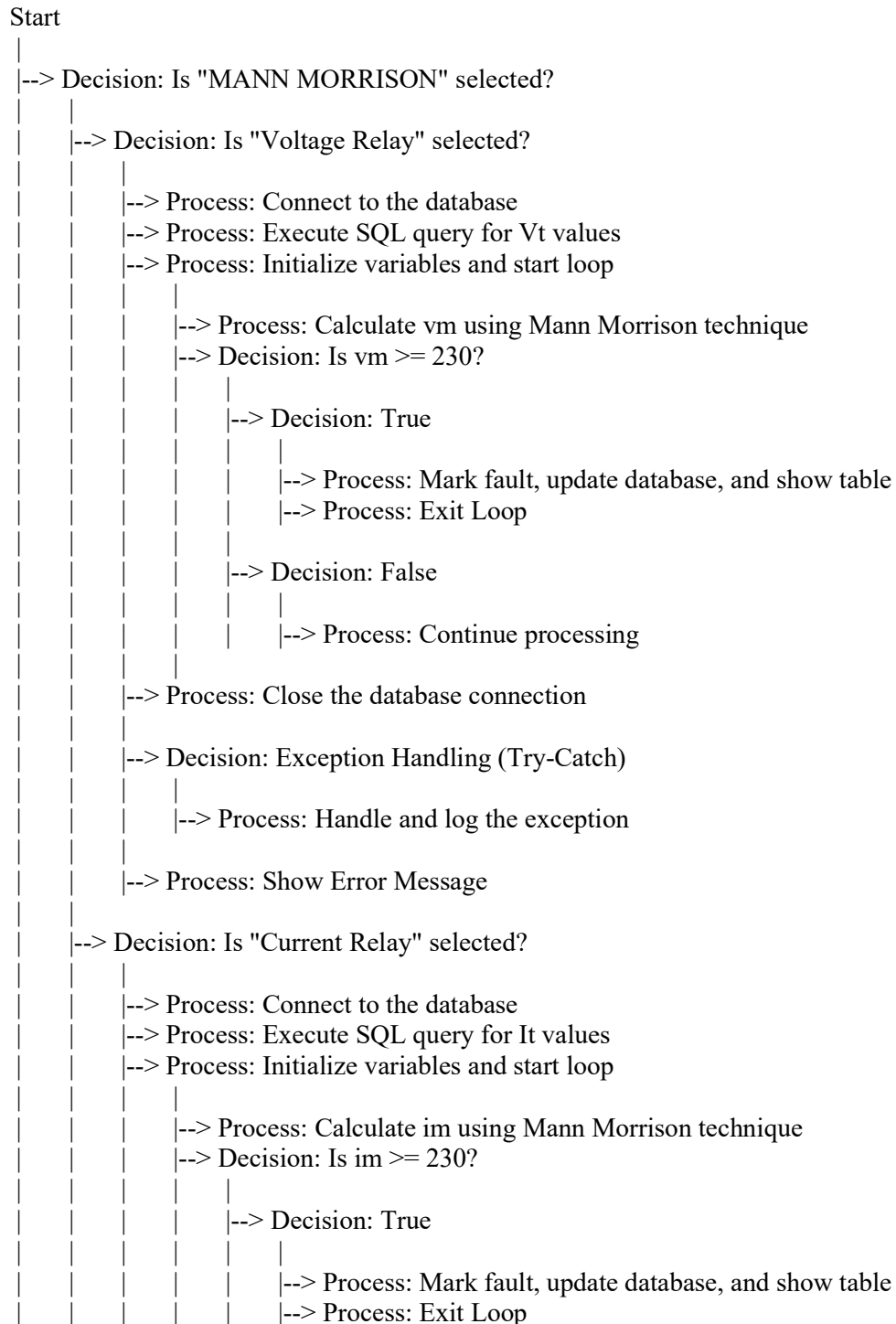
The salient features of "CyberGrid Guardian" are as follows:

1. **Realtime Fault Detection:** The software offers two distinct fault detection techniques: Mann-Morrison and Three Sample. Users can seamlessly switch between these techniques to adapt to varying system conditions, ensuring versatile fault detection capabilities.

2. **Graphical User Interface(GUI):** The software boasts an interactive GUI, enhancing user-friendliness and allowing seamless navigation and control. Users can easily select between fault detection techniques, namely Mann-Morrison and Three-Sample, tailoring the system to specific needs.
3. **Relay Selection:** The software allows users to choose between two fundamental relay types: Overvoltage Relay and Overcurrent Relay. This flexibility ensures adaptability to diverse network requirements and operational scenarios.
4. **Graphical Data Visualization:** Users can generate and visualize graphical representations of relay data. This feature aids in the rapid interpretation of the system's performance and the identification of anomalies.
5. **CSV Data Analysis:** The software supports the analysis of collected data by allowing users to upload CSV files. This feature simplifies data management and extends the software's usability, especially in research and post-fault analysis.
6. **SQL Database Integration:** The core of "CyberGrid Guardian" is its integrated SQL database. This database stores a comprehensive set of information pertaining to processed data, including:
 - a. **Instantaneous Voltage (Vt):** Real-time voltage measurements.
 - b. **Instantaneous Current (It):** Real-time current measurements.
 - c. **Voltage (Vm) and Current (Im) Calculations:** The results obtained from the chosen fault detection techniques.
 - d. **Sampling Time:** The time at which data points were sampled.
 - e. **Cause of Fault:** A critical piece of information that pinpoints the root cause of the fault.
 - f. **Time of Fault:** The exact timestamp when the fault occurred.

7. **Real-Time Data Processing:** A pivotal feature of "CyberGrid Guardian" is its ability to operate on real-time data collected from the microcontroller-based hardware. This capability ensures that the software can promptly respond to live data streams, offering timely fault detection and protection for the transmission system.

3.3 Methodology of the program algorithm



Common time complexities:

- i. **$O(1)$ - Constant Time:** An algorithm with constant time complexity always takes the same amount of time to execute, regardless of the input size. Accessing an element in an array by index is an example of $O(1)$ complexity.
- ii. **$O(\log n)$ - Logarithmic Time:** Algorithms with logarithmic time complexity divide the input size in half at each step. Binary search is a classic example of $O(\log n)$ complexity.
- iii. **$O(n)$ - Linear Time:** An algorithm with linear time complexity takes time proportional to the input size. Scanning an array or a list of elements is typically $O(n)$.
- iv. **$O(n \log n)$ - Linearithmic Time:** This complexity often appears in efficient sorting algorithms like Merge Sort and QuickSort. It's faster than quadratic time but slower than linear.
- v. **$O(n^2)$ - Quadratic Time:** Algorithms with quadratic time complexity have a runtime that grows with the square of the input size. Nested loops iterating over the input data are a common cause of $O(n^2)$ complexity.
- vi. **$O(2^n)$ - Exponential Time:** Exponential time complexity implies that the algorithm's running time doubles with each increase in input size. Many combinatorial problems have exponential time complexity.
- vii. **$O(n!)$ - Factorial Time:** Factorial time complexity means that the algorithm's running time grows factorially with the input size. This is highly inefficient and is often found in brute-force solutions for combinatorial problems.

Time Complexity of the program:

1. **Loop Iterations:** The primary time-consuming part of the code is the loop that processes the data from the database. The loop iterates until a fault is detected or all data points are processed.
 - In the "Voltage Relay" section, it loops through the "Vt" values.
 - In the "Current Relay" section, it loops through the "It" values. The number of iterations depends on the data size and when a fault is detected.
2. **Database Operations:** The program performs database operations, including executing SQL queries, fetching results, and updating records. The time complexity of these operations is dependent on the database system and the number of records.
3. **Mathematical Operations:** The mathematical calculations performed within the loop, including square root, addition, and exponentiation, are generally constant time operations.

The overall time complexity of this program is dominated by the loop iterations and the database operations, making it dependent on the data size and the behavior of the loop. **In the worst case, if the loop processes all data points, the time complexity can be considered $O(N)$, where N is the number of data points.**

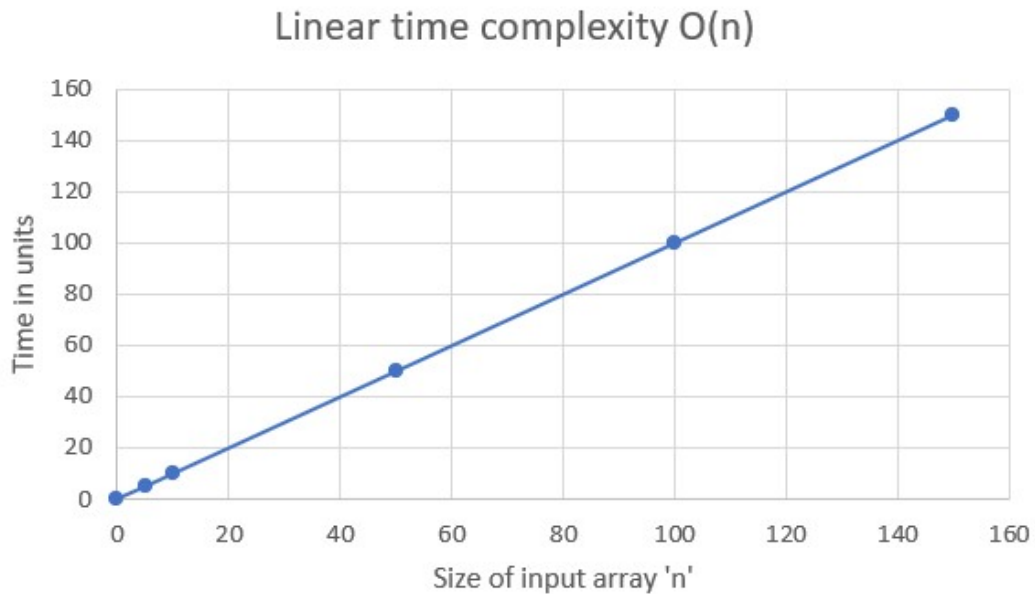


Fig 3.2 : Linear Time Complexity

3.5 Database : MySQL

Features of MySQL?

MySQL is one of the most popular and widely used relational database management systems (RDBMS) in the world. It offers a comprehensive set of features and functionalities to manage data effectively. This report provides an overview of key concepts related to MySQL database management, including Data Definition Language (DDL), Data Manipulation Language (DML), Transaction Control Language (TCL), and the ACID properties.

Data Definition Language (DDL)

DDL is a subset of SQL (Structured Query Language) that focuses on defining and managing the structure of the database. MySQL provides a range of DDL commands for creating, altering, and dropping database objects. Some common DDL commands in MySQL include:

- i. **CREATE:** Used to create new database objects, such as tables, indexes, or views.
- ii. **ALTER:** Enables modifications to the structure of existing database objects, like adding or removing columns from a table.

- iii. **DROP:** Deletes database objects, such as tables, indexes, or databases themselves.
- iv. **TRUNCATE:** Removes all data from a table but retains the table's structure.
- v. **RENAME:** Allows renaming of database objects.

DDL ensures the consistency and integrity of the database structure.

Data Manipulation Language (DML)

DML deals with manipulating data stored in the database. MySQL provides various DML commands to retrieve, insert, update, and delete data. Common DML commands include:

- i. **SELECT:** Retrieves data from one or more tables, allowing for filtering and sorting.
- ii. **INSERT:** Adds new records into a table.
- iii. **UPDATE:** Modifies existing records in a table.
- iv. **DELETE:** Removes records from a table based on specified criteria.

DML ensures data retrieval and manipulation, enabling users to interact with the database's content.

Transaction Control Language (TCL)

TCL is responsible for managing database transactions, ensuring data integrity, consistency, and reliability. MySQL supports the following TCL commands:

- i. **COMMIT:** Confirms all changes made during the current transaction, making them permanent.

- ii. **ROLLBACK:** Reverts any changes made during the current transaction, returning the database to its previous state.
- iii. **SAVEPOINT:** Sets a savepoint within a transaction to which you can later roll back.

TCL commands are essential for ensuring the reliability of database transactions and maintaining data consistency.

ACID Properties

ACID is an acronym that represents a set of properties that guarantee the reliability of database transactions. MySQL, as an RDBMS, adheres to these ACID properties:

- i. **Atomicity:** Each transaction is treated as a single, indivisible unit. It is either fully completed or fully undone (rolled back) if an error occurs during execution.
- ii. **Consistency:** Transactions bring the database from one consistent state to another. It ensures that data constraints and integrity rules are not violated during the transaction.
- iii. **Isolation:** Multiple concurrent transactions do not interfere with each other. Each transaction sees a consistent snapshot of the data, even in a multi-user environment.
- iv. **Durability:** Once a transaction is committed, its changes are permanent and persist even in the face of system failures.

The ACID properties are fundamental to maintaining data reliability and are crucial in applications where data accuracy and consistency are paramount.

Database structure :

Each time a new CSV file is uploaded the program will create a new table for storing the data so that every file/table is isolated and the values do not interfere with each other.

```
mysql> show tables;
+-----+
| Tables_in_cybergrid |
+-----+
| algo                 |
| readings             |
+-----+
```

Fig 3.1 New table is created for each file

MySQL query to create table in database :

```
CREATE TABLE your_table_name (
  column1_name data_type constraints,
  column2_name data_type constraints,
  columnN_name data_type constraints
);
```

Table Structure :

Each table in the database will have nine columns namely “row_id, sampling Time, Vt, It, time, Vm, Im, Voltage Relay and Current Relay(cause of fault)”.

Field	Type	Null	Key	Default	Extra
row_id	int	NO	PRI	NULL	auto_increment
sampTime	double	YES		NULL	
Vt	double	YES		NULL	
It	double	YES		NULL	
time	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
Vm	double	YES		NULL	
Im	double	YES		NULL	
voltageRelay	tinyint(1)	YES		0	
currentRelay	tinyint(1)	YES		0	

Fig 3.2 Structure of tables in database

MySQL query to insert data into table.

```
INSERT INTO table_name (column1, column2,
column3, ...)
VALUES (value1, value2, value3, ...);
```

row_id	sampTime	Vt	It	time	Vm	In	voltageRelay	currentRelay
1	0	39.9390808633939	-38.6370330515627	2023-10-28 13:45:13	229.9935	39	0	0
2	0.00002	41.3614609449631	-38.5712224967526	2023-10-28 13:45:13	229.9935	39	0	0
3	0.00004	42.7822081468764	-38.5038892161228	2023-10-28 13:45:14	229.9935	39	0	0
4	0.00006	44.2012663804668	-38.4350358678759	2023-10-28 13:45:14	229.9935	39	0	0
5	0.00008	45.6185796237453	-38.3646651702243	2023-10-28 13:45:14	229.9935	39	0	0
6	0.0001	47.0340919236117	-38.2927799012827	2023-10-28 13:45:14	229.9935	39	0	0
7	0.00012	48.4477473980643	-38.2193828989582	2023-10-28 13:45:14	229.9935	39	0	0
8	0.00014	49.8594902384054	-38.144477060839	2023-10-28 13:45:14	229.9935	39	0	0
9	0.00016	51.269264711445	-38.0680653440793	2023-10-28 13:45:14	229.9935	39	0	0
10	0.00018	52.6770151617009	-37.9901507652827	2023-10-28 13:45:14	229.9935	39	0	0

Fig 3.3 Data in table

MySQL query to view/select data from table

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

3.6 Results

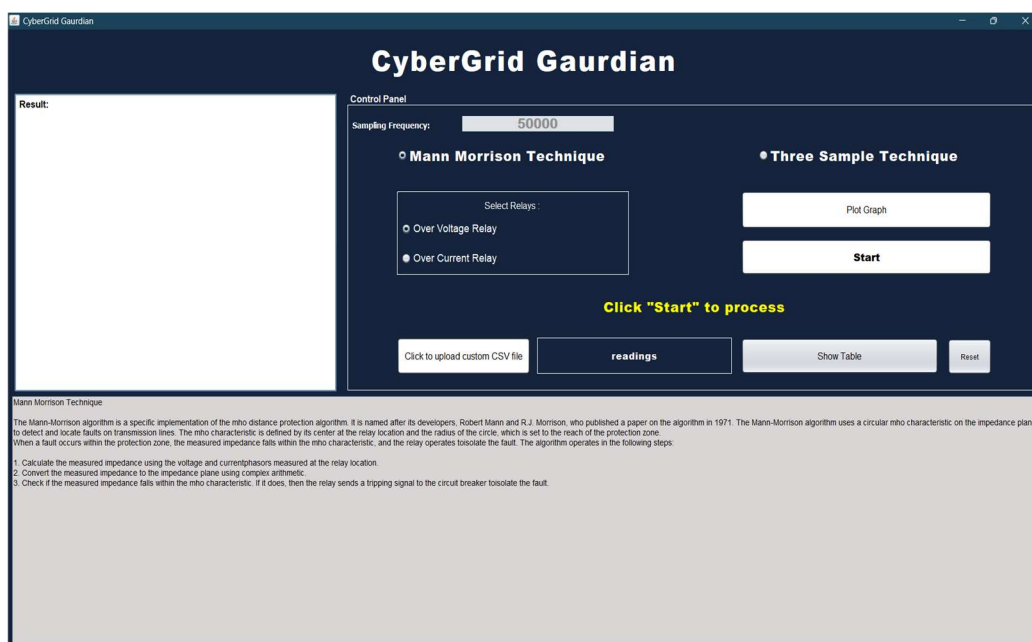


Fig 3.4 GUI of the software developed

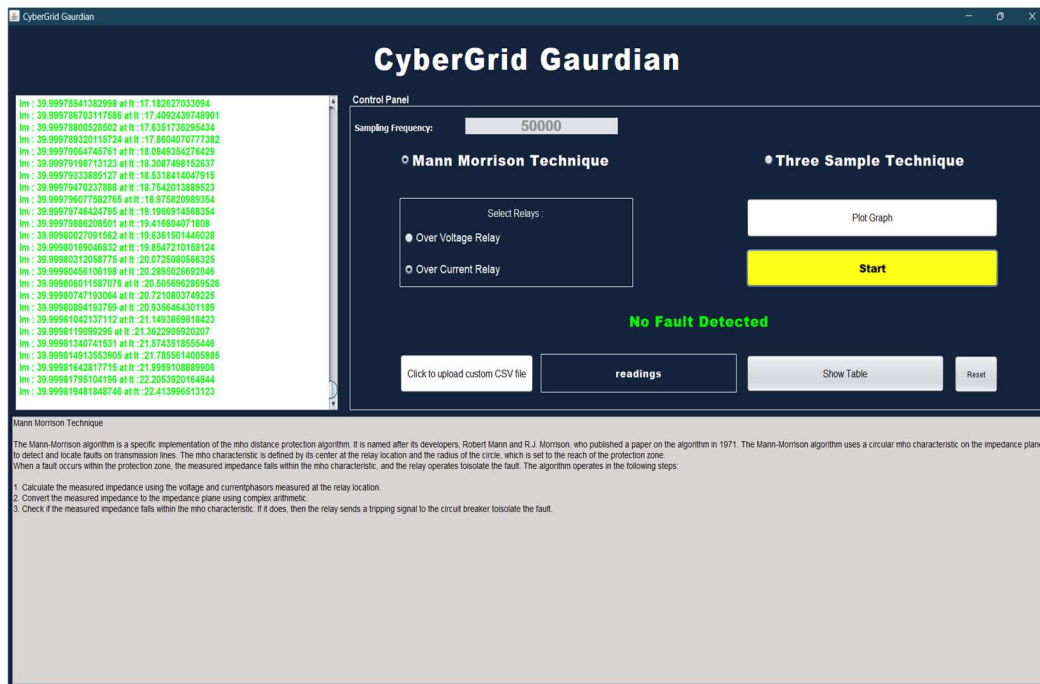


Fig 3.5 No Over Current Fault detected with Mann Morrison Tech.

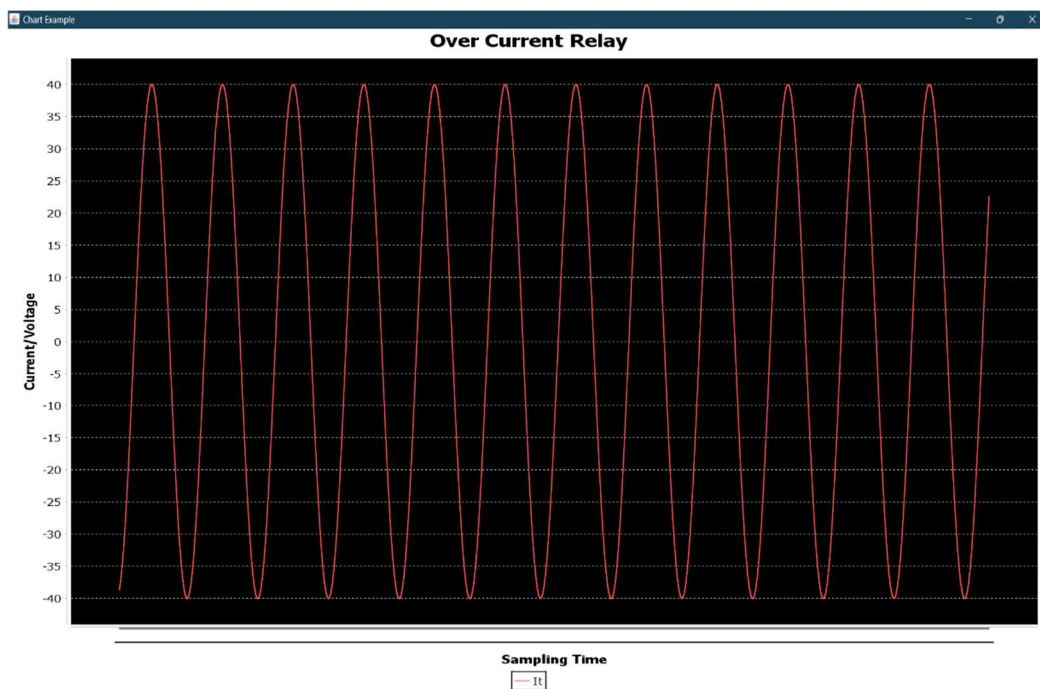


Fig 3.6 Graph showing No Over Current fault

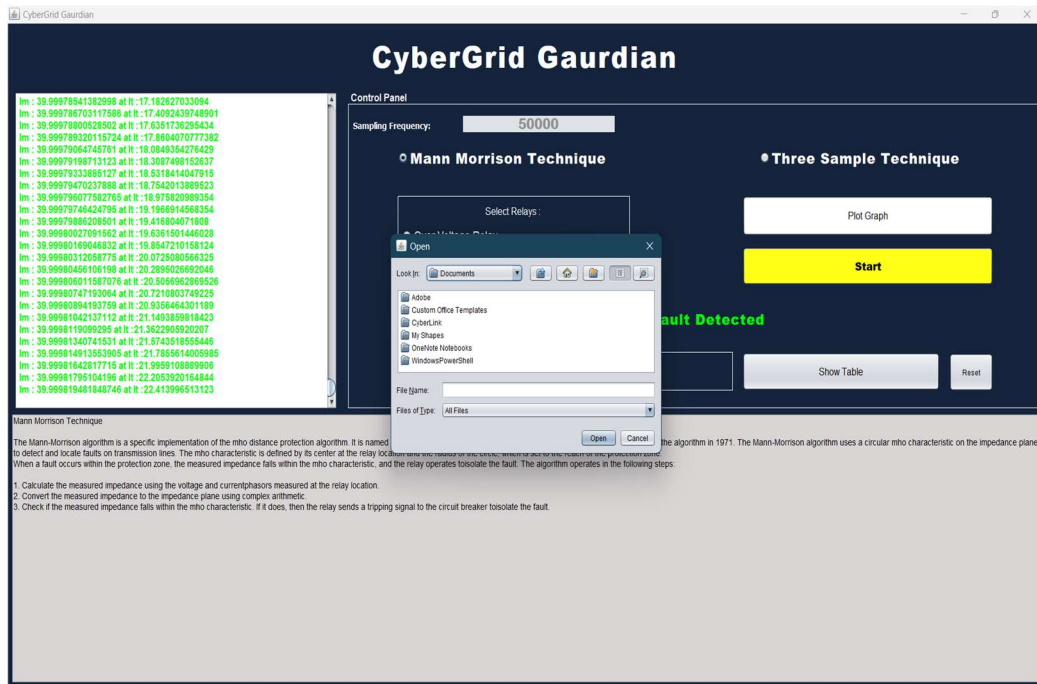


Fig 3.7 Uploading custom CSV file

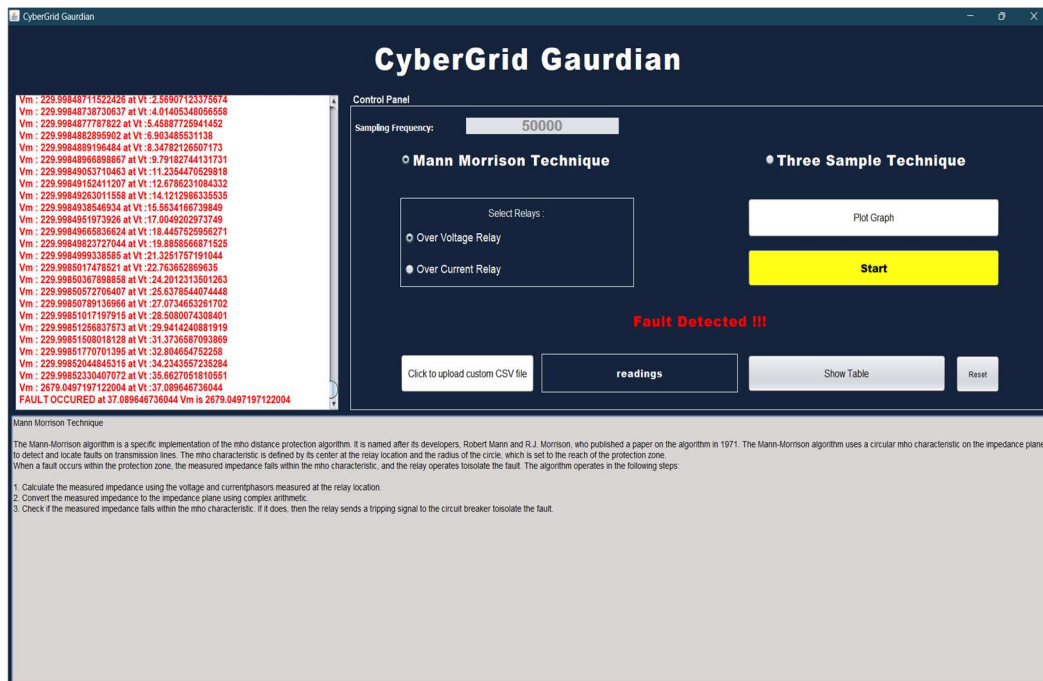


Fig 3.8 Over Voltage Fault Detected with Mann Morrison Tech.

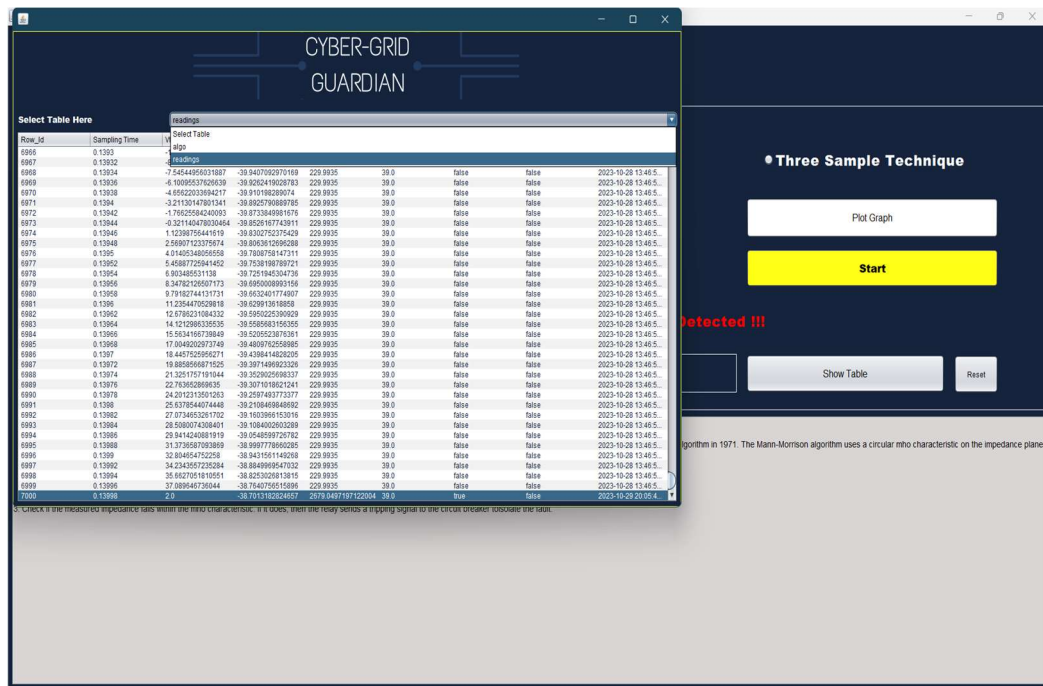


Fig 3.9 Updated database with detected fault

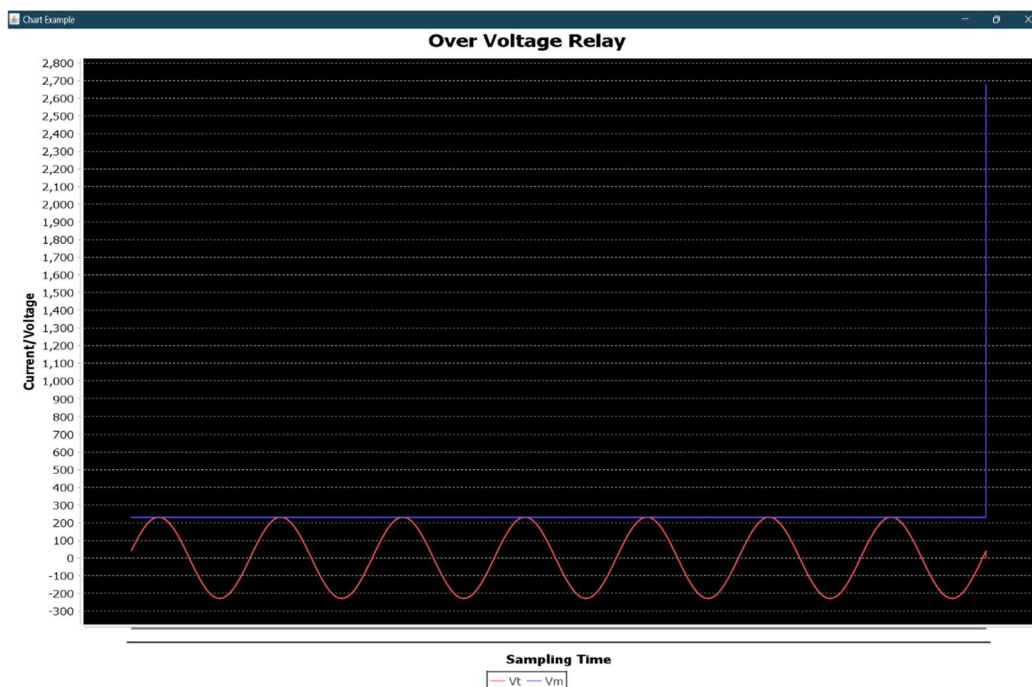


Fig 3.10 Graph showing over voltage fault in blue line

The screenshot shows the NetBeans IDE interface with the 'Output' window open. The output displays the execution of a Java program named 'FaultDetectionSystem'. The program iterates through a series of calculations, updating 'next2', 'prev2', and 'curr2' values. The output is as follows:

```
next2 26.326560597709
Final Answer 39.999852485226356
prev2 26.515277082278
curr2 26.326560597709
next2 26.1368241596063
Final Answer 39.99985084533189
prev2 26.326560597709
curr2 26.1368241596063
next2 25.8460558844389
Final Answer 39.99984920822105
prev2 26.1368241596063
curr2 25.8460558844389
next2 25.7542433034123
Final Answer 39.99984754113578
prev2 25.8460558844389
curr2 25.7542433034123
next2 25.5614539881685
Final Answer 39.999845943347545
prev2 25.7542433034123
curr2 25.5614539881685
next2 25.3674355504895
Final Answer 39.99984431611929
prev2 25.5614539881685
curr2 25.3674355504895
next2 25.1720156419956
Final Answer 39.999842692479785
prev2 25.3674355504895
curr2 25.1720156419956
next2 24.9770019538426
Final Answer 39.999841073316466
prev2 25.1720156419956
curr2 24.9770019538426
next2 24.7802022164202
Final Answer 39.99983945827396
prev2 24.9770019538426
curr2 24.7802022164202
next2 24.5824241990466
Final Answer 39.99983784779868
prev2 24.7802022164202
curr2 24.5824241990466
next2 24.38367570946537
Final Answer 39.999836242115437
prev2 24.5824241990466
```

Fig 3.11 Output of algorithm in console

Chapter 4

Hardware Implementation of Distance Protection Algorithm

4.1 Introduction

After successfully developing the code for distance protection algorithms, the next step is to implement the code on a hardware circuit. The hardware circuit comprises two primary components: the Arduino Uno and the microcontroller (Atmega 32a). The Arduino Uno is a microcontroller board that is designed to facilitate the creation of digital devices and interactive objects that can sense and control the physical world. It is a popular platform for developing projects that involve electronic sensors and actuators, as it provides a user-friendly programming environment and a wide range of pre-built libraries and tools. The Atmega 32a microcontroller is a low-power, high-performance microcontroller that is widely used in a variety of applications, including industrial control systems, automotive electronics, and consumer electronics. It offers a range of features, including on-chip flash memory, multiple communication interfaces, and various peripherals, making it ideal for implementing complex algorithms such as distance protection algorithms. The integration of the Python code and the hardware circuit will allow for real-time testing and evaluation of the distance protection algorithms in a practical setting.

4.2 Micro-Controllers

Arduino UNO is a microcontroller board based on the ATmega328P. It comprises 14 digital input/output pins, including six that can be used as pulse-width modulation (PWM) outputs, six analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an In-Circuit Serial Programming (ICSP) header, and a reset button. The board is self-sufficient to support the microcontroller, and it can be connected to a computer with a USB cable or powered with an AC-to-DC adapter or battery. The microcontroller board can be used to interface with different sensors, motors, and other electronic components to build a functional system.

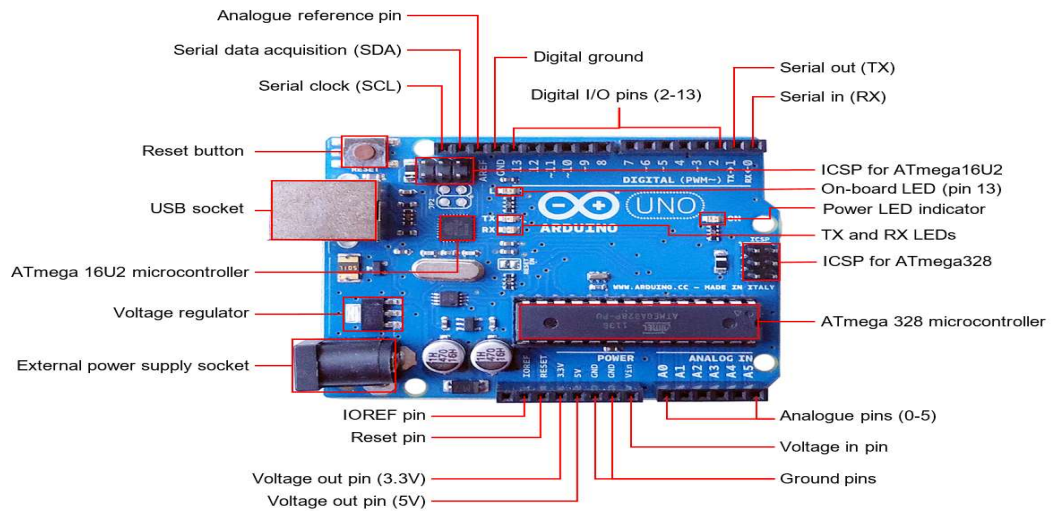


Fig 4.1 Arduino UNO

Arduino UNO is a microcontroller board based on the ATmega328P microcontroller. It is one of the most popular and widely used development boards in the Arduino family, owing to its compact design, ease of use, and versatility. Here are some key features of the Arduino UNO:

1. **Microcontroller:** The ATmega328P microcontroller is a high-performance, low-power 8-bit AVR microcontroller that provides 32KB of flash memory, 2KB of SRAM, and 1KB of EEPROM. Also in Input/Output the board has 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, and an ICSP header.
2. **Compatibility:** The Arduino UNO is compatible with a wide range of sensors, actuators, and other devices, making it ideal for a variety of projects.
3. **Programming:** The board can be programmed using the Arduino programming language, which is based on C/C++. The Arduino IDE (Integrated Development Environment) is a user-friendly platform that simplifies the process of programming the board.
4. **Power:** The board can be powered via the USB connection or an external power supply (7-12V DC). It also has a built-in voltage regulator, which allows it to accept a wide range of input voltages.
5. **Shields:** The Arduino UNO is designed to be compatible with various expansion shields that allow you to add functionality to the board, such as WiFi, Ethernet, LCD displays, and motor control.

The Arduino UNO is a versatile and easy-to-use microcontroller board that is perfect for beginners and advanced users alike. With its open- source platform and wide range of compatible devices and shields, it is an ideal choice for building all sorts of electronic projects, from simple to complex.

4.3 Design of Hardware Model

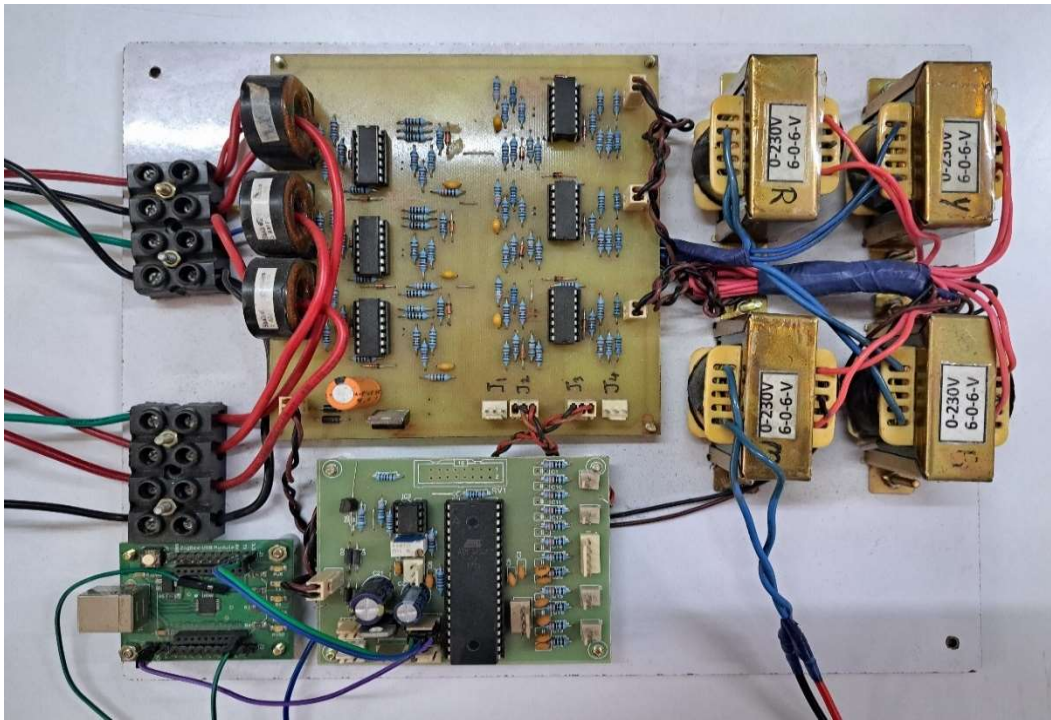


Fig.4.2: The Hardware setup for the project.

Components of Hardware Circuits:

- Potential Transformers (4 in Nos.): Voltage transformers (VT), also called potential transformers (PT), are a parallel-connected type of instrument transformer. A potential transformer (P.T.) is an instrument transformer which is used for the protection and measurement purposes in the power systems. A potential transformer is mainly used to measure high alternating voltage in a power system.
- Current Transformers (3 in Nos.): A Current Transformer (CT) is used to measure the current of another circuit. CTs are used worldwide to monitor high-voltage lines across national power grids. A CT is designed to produce an alternating current in its secondary winding that is proportional to the current that it is measuring in its primary. A current transformer is designed to maintain

an accurate ratio between the currents in its primary and secondary circuits over a defined range.

- **Signal Conditioning Unit:** A signal conditioner is a device that modifies raw analog output signals produced by sensors and provides the essential circuitry between the sensor and the data acquisition system. These modifications convert these various signals into signals that are compatible with process monitoring and control devices.
- **Microcontroller (Atmega 32):** The Atmega32 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega32 achieves throughput approaching 1 MIPS per MHz allowing the system designed to optimize power consumption versus processing speed.
- **Load:** Electricity needs something to do, called a load, in order to be useful in a circuit. Without a load, the circuit won't do anything.
- **Java:** Java is used for calculations and simulation of Mann Morrison Algorithm & Three Sample Algorithm.

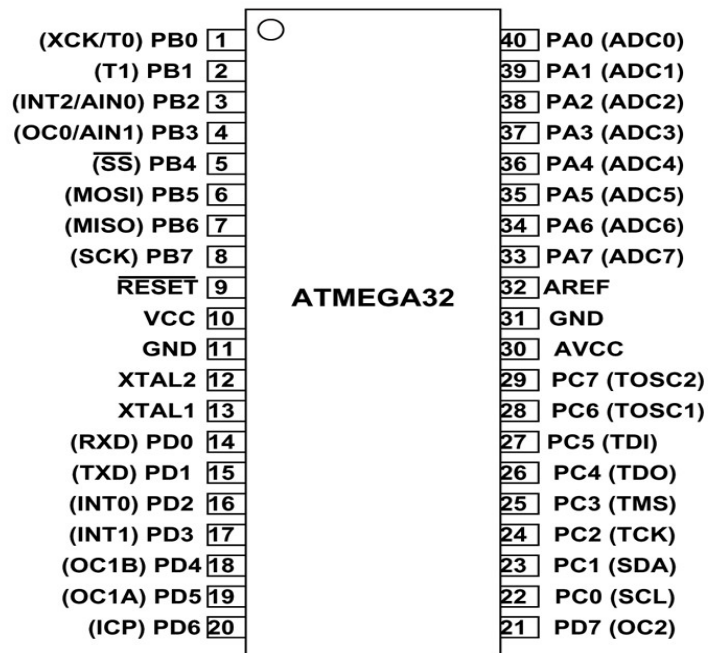


Fig 4.3 : Pin Diagram of ATmega 32 Microcontroller

ATmega32 is an 8-bit microcontroller from the AVR family of microcontrollers, developed by Atmel Corporation, now a subsidiary of Microchip Technology. It is a versatile and powerful microcontroller with a wide range of applications in

various Industries including consumer electronics, industrial control systems, and automotive electronics.

The ATmega32 has 32K Bytes of in-system programmable Flash memory, 2K Bytes of SRAM, and 1K Byte of EEPROM. It operates on a wide voltage range from 1.8V to 5.5V and has a clock speed of up to 16 MHz. The microcontroller also has 32 I/O pins, which can be configured as input or output. One of the key features of the ATmega32 is its advanced interrupt handling capability, which allows for fast and efficient response to external events. The microcontroller also has a sophisticated power management system that enables it to operate on low power consumption, making it suitable for battery- powered applications.

The ATmega32 also has a range of communication interfaces, including two USARTs (Universal Synchronous/Asynchronous Receiver/Transmitter), a SPI (Serial Peripheral Interface), and a two-wire I2C (Inter-Integrated Circuit) bus. These interfaces allow the microcontroller to communicate with other devices in a variety of ways.

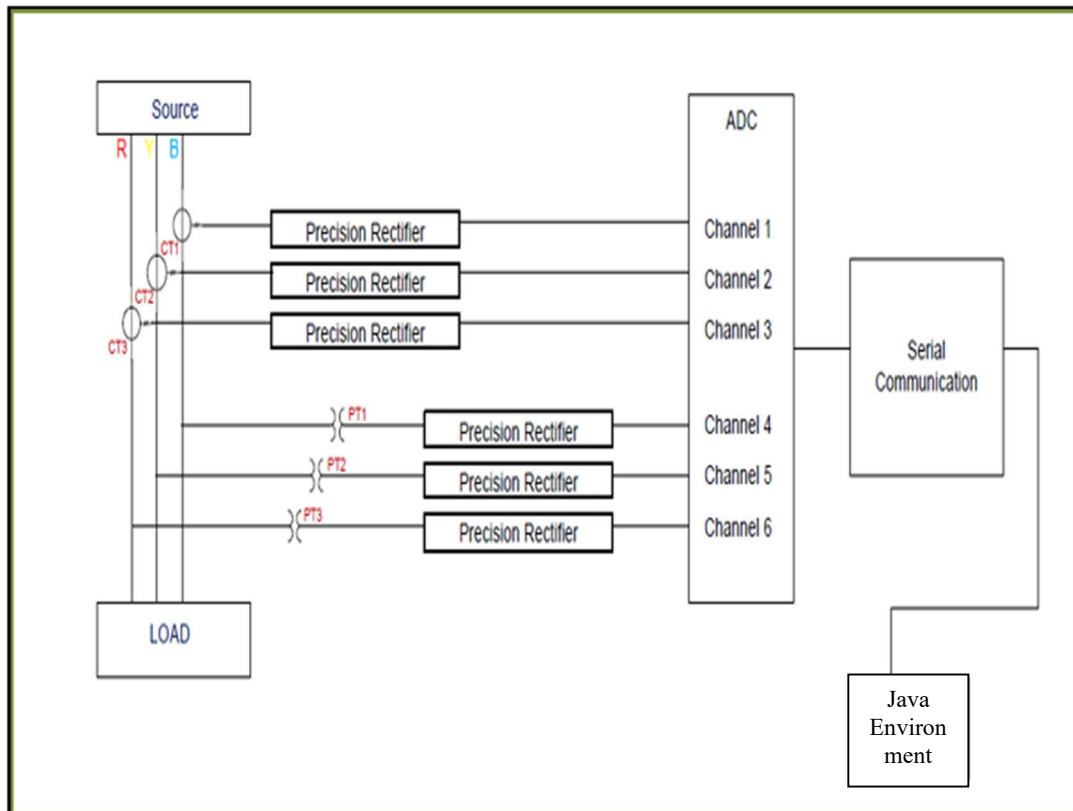


Fig 4.4 : Block diagram of circuit

Programming the ATmega32 is relatively straightforward, thanks to the AVR Studio Integrated Development Environment (IDE) and the AVR Instruction Set, which includes a broad range of instructions optimized for efficient code execution. Additionally, the microcontroller supports in-system programming (ISP) and in-circuit programming (ICP), which allows for programming and debugging without removing the microcontroller from the target board.

The ATmega32 is a powerful and versatile microcontroller with a wide range of features and capabilities, making it an excellent choice for a broad range of applications. At the heart of the Atmega32A's pin diagram are its 40 input/output (I/O) pins, which are organized into four ports labeled A, B, C, and D. These ports are used to interface with a variety of external devices and peripherals, such as sensors, actuators, and displays. Additionally, the microcontroller features a range of other pins and functions, including power supply pins, clock pins, and programming pins. The power supply pins provide the necessary voltage and current to operate the microcontroller, while the clock pins are used to synchronize the operation of the microcontroller with other components in the system.

The programming pins are used to program and debug the microcontroller, allowing developers to upload code and monitor its execution. Overall, the pin diagram of the Atmega32A is a crucial component of its design, providing a clear and concise overview of its various pins and functions. By understanding the pin diagram, engineers and developers can design and implement complex embedded systems and applications with ease, leveraging the power and flexibility of this versatile microcontroller.

Data Acquisition System:

Data acquisition is the process of sampling of real-world analog signals and conversion of the resulting samples into digital numeric values that can be manipulated by a computer. The system which performs data acquisition is called data acquisition system (DAS). The main components of DAS are the signal conditioner (analog input subsystem) and the analog interface.

1. Signal conditioner(Analog Input Substation): Signal conditioner (also called analog input subsystem) is necessary to make the signals from the transducers compatible with the analog interface.

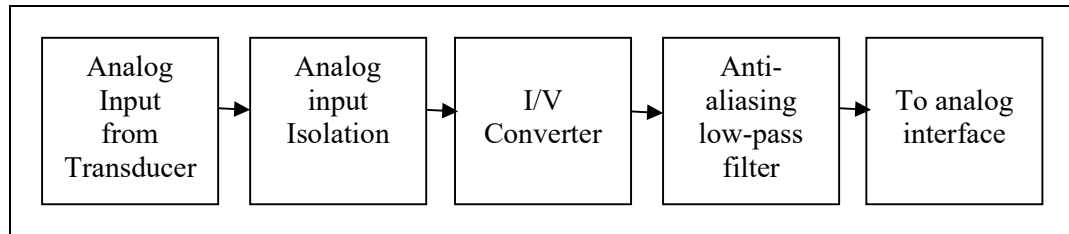


Fig 4.5 : Block diagram of Data Acquisition System

2. Aliasing: Post-fault power system signals contain dc offset and harmonic components, in addition to the major fundamental frequency component. In order to convert analog signals to sequences of numbers, an appropriate sampling rate should be used, because high- frequency components which might be present in the signal, could be incorrectly interpreted as components of lower frequencies.
3. Sampling: Sampling is the process of converting a continuous time signal, such as a current or voltage, to a discrete time signal. The selected sampling rate should be as high as is practical taking into account the capabilities of the A/D converter and the processor. Modern numerical relays use sampling rate that are as high as 96 samples per cycle. (Sampling frequency is 9.6 KHZ used).

Analog Interface: The analog interface makes the signal compatible with the processor. The outputs of the signal conditioner are applied to the analog interface which includes sample and hold (S/H) circuits, analog multiplexers and analog to digital (A/D) converters.

4.4 Data Collection through serial communication

Arduino and Java can work together in a variety of ways, allowing you to leverage the strengths of both platforms to create powerful applications. One way to use Arduino and Java together is by using the RXTX library of JAVA to communicate with an Arduino board. RXTX Library allows Java to send and receive data to and from an Arduino board over a serial connection. This can be

useful for controlling the Arduino from a Java program, or for sending data from the Arduino to the software for processing. Here, we have implemented the Mann-Morrison algorithm for single phase system.

What is single phase circuit?

A single-phase circuit is an electrical system that utilizes a single alternating current (AC) waveform to distribute power. It is commonly used in residential and light commercial applications where the power requirements are relatively low. In a single-phase circuit, the AC voltage wave alternates between positive and negative values in a regular pattern, with a frequency of typically 50 (in India) or 60 Hertz (Hz) (in USA).

The power is delivered using two wires, which are typically color-coded as black and white, or red and black. Single-phase circuits are used to power smaller appliances and devices, such as fans, refrigerators, and lighting fixtures. They are also used for smaller motors, such as those found in air conditioning units and pumps.

Compared to three-phase circuits, single-phase circuits have a lower power output and are less efficient. They can experience voltage fluctuations and dips, which can affect the performance of the connected devices. However, they are simpler and less expensive to install and maintain, making them a practical choice for many applications.

For real time data accumulation using Arduino UNO, Arduino IDE is used. The program to stream the data over a serial cable is dumped into Arduino using the IDE. The software receive data from micro-controller through the serial port using RXTX Library. Once we get the data the required parameters are calculated and the Mann Morrison Algorithm is applied on it. The output of this simulation contains the value of resistance (R), capacitance (X), inductance(L) and impedance (Z). Although, in this project we have simulated the algorithm for single phase, it can be simulated for three phase as well. For simulating the algorithm for three phase, the values of all the three phases of voltage and current is to be dumped in Arduino and passed to the software. Fig 4.2 shows

the output of Mann Morrison algorithm using Arduino UNO through serial cable.

```
02 02 2D 02 2D 00 02 2D 00 8C 02 2D 00 8C 00 3A 02 2D 00 8C 00 3A 02 2D 00 8C 00 3A 2D B3 02 2D 00 8C 00 3A 2D B3 00 2D 00 8C 00 3A 2D B3 00 26 02 2D 00 8C 00 3A 2D B3
00 26 00 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 02 2D 00 8C 00 3A
2D B3 00 26 00 4F 00 26 02 88 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94
00 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D
43 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02
88 00 94 00 3A 2D 43 00 7A 00 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02
02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 2D 00 8C 00 3A
2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 02 2D 00 8C 00 3A 2D B3
00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 6E 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 6E 02 2D 00 8C 00 3A
2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 6E 00 3A 2D 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 6E 00 3A 2D
02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 6E 00 3A 2D A8 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00
68 02 21 00 6E 00 3A 2D A8 00 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 6E 00 3A 2D A8 00 3E 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02
88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 6E 00 3A 2D A8 00 3E 00 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 6E 00 3A 2D A8 00 3E 00
3C 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D 43 00 7A 00 88 00 68 02 21 00 6E 00 3A 2D A8 00 3E 00 3C 00 02 2D 00 8C 00 3A 2D B3 00 26 00 4F 00 26 02 88 00 94 00 3A 2D
```

Fig 4.6 Output of Arduino showing different parameters through serial cable

4.5 Data Collection through ESP32 Module

In recent years, the integration of Internet of Things (IoT) devices in various applications has become increasingly prevalent. The ESP32 microcontroller, developed by Espressif Systems, is a versatile platform known for its robust Wi-Fi and Bluetooth capabilities. This section explores the process of collecting data using an ESP32 module and feeding that data into a Google Spreadsheet for real-time monitoring and analysis.

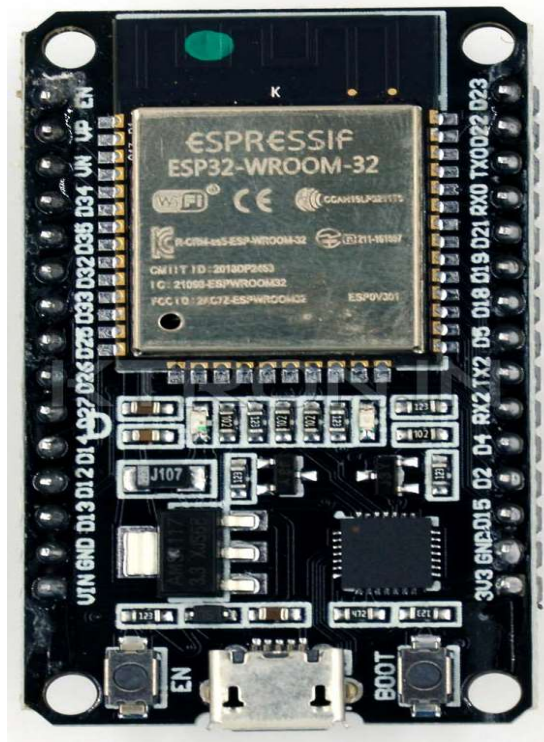


Fig: 4.7 : ESP32 Module

ESP32 Microcontroller

The ESP32 microcontroller is a powerful and flexible platform that offers numerous features:

- **Dual-Core Processor:** The ESP32 is equipped with two CPU cores, providing the ability to perform multitasking and handle complex tasks efficiently.
- **Wi-Fi Connectivity:** The built-in Wi-Fi module allows the ESP32 to connect to local Wi-Fi networks, making it suitable for IoT applications requiring internet connectivity.
- **Bluetooth Connectivity:** The integrated Bluetooth module enables communication with other Bluetooth devices and peripherals.
- **GPIO Pins:** The ESP32 provides a range of General-Purpose Input/Output (GPIO) pins that can be used to interface with sensors, actuators, and other components.
- **Analog-to-Digital Converters (ADC):** The ESP32 features multiple ADC channels, allowing it to read analog sensor data.

Programming

The ESP32 microcontroller is programmed to read data from the sensors periodically. The data can be collected at predefined intervals, ensuring accurate and up-to-date information. The ESP32's firmware should be written to interface with the sensors, read the data, and store it in variables.

Data Formatting

The collected data is formatted into a suitable structure. In most cases, a common format is CSV (Comma-Separated Values) as it is easily interpretable by spreadsheet software. The ESP32 can convert the collected data into a CSV format, which separates the values with commas and adds line breaks for different data entries.

Feeding Data into Google Spreadsheet : Google Sheets Integration

To enable data transmission to Google Sheets, the following steps are required:

- **Google Developer Console:** You need to create a project on the Google Developer Console and enable the Google Sheets API for your project.
- **OAuth 2.0 Authentication:** Set up OAuth 2.0 authentication for your project, allowing the ESP32 to access the Google Sheets API.

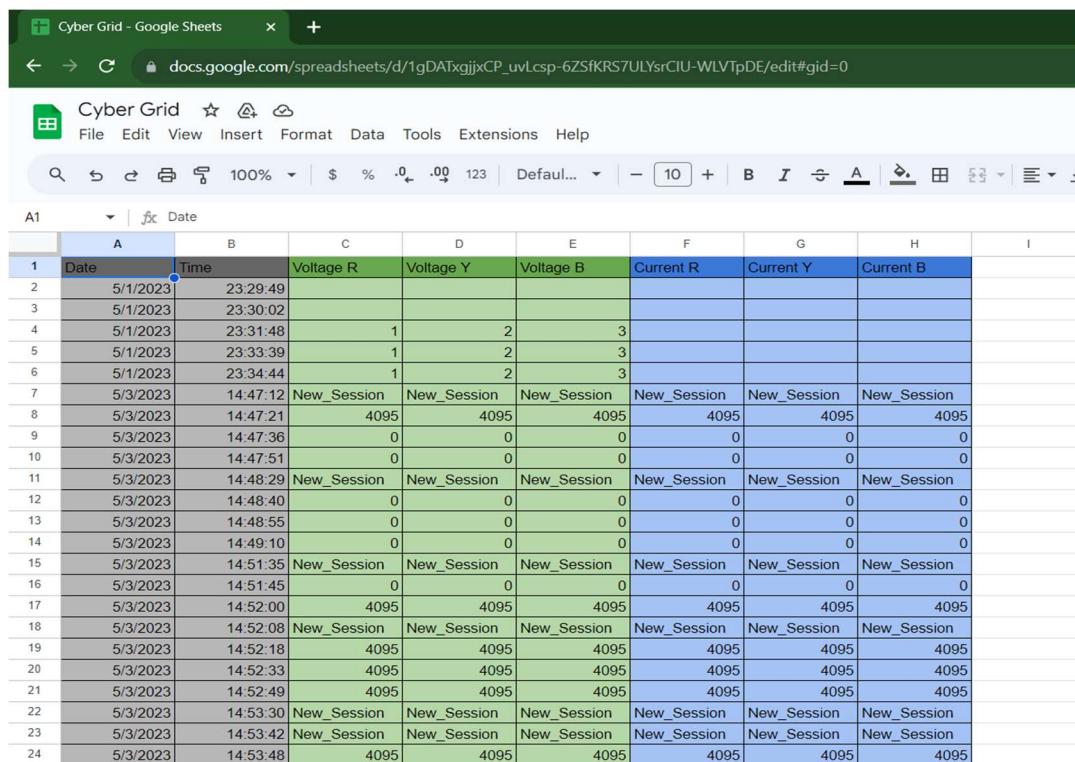
- **Access to the Spreadsheet:** Share the target Google Spreadsheet with the Google Service Account email to grant access for writing data.

Data Upload

The ESP32 sends POST requests to Google Sheets using the API key and OAuth 2.0 authentication. The POST request includes the formatted data, which is then appended to the spreadsheet. This process can be automated to occur at predefined intervals for real-time data tracking.

Benefits and Applications

- **Real-Time Monitoring:** This system provides real-time data updates, making it suitable for applications where constant monitoring and immediate access to data are essential.
- **Remote Data Collection:** With the ESP32's Wi-Fi capabilities, data can be collected from remote locations and transmitted to a central Google Spreadsheet.
- **Data Analysis:** The collected data can be easily analyzed, visualized, and shared using Google Sheets' built-in tools, making it useful for a wide range of applications, from environmental monitoring to industrial automation.



	A	B	C	D	E	F	G	H	I
	Date	Time	Voltage R	Voltage Y	Voltage B	Current R	Current Y	Current B	
1	5/1/2023	23:29:49							
2	5/1/2023	23:30:02							
3	5/1/2023	23:31:48	1	2	3				
4	5/1/2023	23:33:39	1	2	3				
5	5/1/2023	23:34:44	1	2	3				
6	5/3/2023	14:47:12	New_Session	New_Session	New_Session	New_Session	New_Session	New_Session	
7	5/3/2023	14:47:21	4095	4095	4095	4095	4095	4095	
8	5/3/2023	14:47:36	0	0	0	0	0	0	
9	5/3/2023	14:47:51	0	0	0	0	0	0	
10	5/3/2023	14:48:29	New_Session	New_Session	New_Session	New_Session	New_Session	New_Session	
11	5/3/2023	14:48:40	0	0	0	0	0	0	
12	5/3/2023	14:48:55	0	0	0	0	0	0	
13	5/3/2023	14:49:10	0	0	0	0	0	0	
14	5/3/2023	14:51:35	New_Session	New_Session	New_Session	New_Session	New_Session	New_Session	
15	5/3/2023	14:51:45	0	0	0	0	0	0	
16	5/3/2023	14:52:00	4095	4095	4095	4095	4095	4095	
17	5/3/2023	14:52:08	New_Session	New_Session	New_Session	New_Session	New_Session	New_Session	
18	5/3/2023	14:52:18	4095	4095	4095	4095	4095	4095	
19	5/3/2023	14:52:33	4095	4095	4095	4095	4095	4095	
20	5/3/2023	14:52:49	4095	4095	4095	4095	4095	4095	
21	5/3/2023	14:53:30	New_Session	New_Session	New_Session	New_Session	New_Session	New_Session	
22	5/3/2023	14:53:42	New_Session	New_Session	New_Session	New_Session	New_Session	New_Session	
23	5/3/2023	14:53:48	4095	4095	4095	4095	4095	4095	

Fig 4.8 Output of ESP32 Module

4.6 Conclusion

The integration of an ESP32 microcontroller with Google Sheets for data collection and real-time monitoring is a powerful solution for IoT applications. It offers the flexibility to collect data from various sensors and send it to a centralized spreadsheet, making it accessible for analysis and decision-making. This system can be applied to a diverse range of applications where data tracking and analysis are crucial.

Chapter 5

Conclusion and Future Scope

5.1 Introduction

In the previous stage, a study was conducted on transmission lines, relays, Python programming, programming in Python, and distance protection algorithms. In the concluding part of the study, the methodology was successfully implemented on a single-phase circuit, which was comprised of a hardware circuit with a microcontroller as its main component. The implementation involved the direct extraction of data from the hardware circuit. The study was concluded by demonstrating the successful implementation of the methodology.

5.2 Conclusion

In summary, the "CyberGrid Guardian" software represents a sophisticated and versatile numerical relay system, integrating advanced fault detection techniques, data analysis tools, and a robust SQL database. Its ability to work with both real-time and historical data positions it as a valuable asset in the realm of power system protection, aiding engineers, operators, and inspection teams in maintaining the reliability and safety of transmission systems. This project showcases a profound understanding of power system engineering, real-time data processing, and fault detection methodologies, and offers a practical and effective solution for the power industry.

5.3 Future Scope

The model described above can be extended to three-phase systems, where data can be extracted from a three-phase circuit. The code developed for the Mann Morrison algorithm and the Three- Sample Technique can be applied to the hardware circuit to create a system that can obtain real-time data and convert it from hex format to an integer format.

By implementing distance protection algorithms in real-time, this system can safeguard long head transmission lines from any potential damages caused by short circuits, fires, surges, or equipment failures. Furthermore, the

implementation of distance protection algorithms using Java programming language offers several advantages, such as improved speed, accuracy, and flexibility in algorithm design. The utilization of real-time data enables the system to respond quickly and effectively to any potential threats to the transmission lines, ensuring that any damage or disruption to the power supply is minimized. Additionally, the ability to extract data from the hardware circuit enables the system to continuously monitor the transmission lines, providing an early warning system that can detect potential faults before they develop into major issues. Overall, the extension of this model to three-phase systems represents a significant step forward in the development of effective and efficient distance protection schemes. By utilizing advanced algorithms and real-time data processing techniques, this system can play a crucial role in ensuring the reliability and stability of power transmission systems, even in the face of unexpected disruptions and failures.

=====**=====

6. Appendices

1. Code snippet for Mann Morrison Voltage and Current Relay

```
static Connection conn;
static Statement st;
String attachment_path;
public static String tableName = "readings";

// MANN MORRISON TECHNIQUE + VOLTAGE RELAY SELECTED
if(mann.isSelected() && voltRelay.isSelected()){
ta.append("MANN MORRISON TECHNIQUE + VOLTAGE RELAY SELECTED \n
CONNECTED DATABASE: "+ tableName +"\n Processing Started\n");
boolean isfault = false;
//double f = Integer.parseInt(sampleFreq.getText());
//double deltaT = 1/f;
final double omega = 314.159265;
try{
    Class.forName("java.sql.Driver");
    Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost/cyberGrid","root","
root");

    st = conn.createStatement();
    String stmt="SELECT Vt FROM " + tableName + " ";
    st.executeQuery(stmt);
    String sql=stmt;
    ResultSet rs = st.executeQuery(sql);
    ta.append("Connection Established\n");
    rs.next();
    double prev, curr, nxt;
    prev = rs.getDouble("Vt");
    rs.next();
    curr = rs.getDouble("Vt");
    rs.next();
    nxt = rs.getDouble("Vt");
    rs.next();
    while(isfault != true){
        double diff = (nxt - prev);
        double vDash1 = diff/(0.00004);
        double vDash = vDash1/omega;
        double vmsq = Math.pow(vDash, 2);
        double vtsq = Math.pow(curr, 2);
        double vm = Math.sqrt(addition);
        System.out.println("Final Answer "+vm);
        ta.append("Vm : "+vm+ " at Vt : " + curr + "\n");

        if(vm >= 230.00){
            isfault = true;
            ta.append("FAULT OCCURED at"+curr+"Vm is"+vm);
            ta.setForeground(Color.red);
            lbl1.setText("Fault Detected !!!");
            lbl1.setForeground(Color.RED);
            lbl1.setVisible(true);

            String updateFaultSql = "UPDATE " + tableName +
" SET voltageRelay = true, time = CURRENT_TIMESTAMP, Vm =" + vm +
"WHERE Vt =" + nxt;
```



```

        st.executeUpdate(updateFaultSql);
        new table().setVisible(true);
        break;
    }
    prev = curr;
    curr = nxt;
    nxt = rs.getDouble("Vt");
    rs.next();
    ta.setForeground(Color.green);
    lbl1.setText("No Fault Detected");
    lbl1.setForeground(Color.GREEN);
    lbl1.setVisible(true);
}
}catch(Exception e){
    System.out.print(e);
}
}

// MANN MORRISON TECHNIQUE + CURRENT RELAY SELECTED

else if(mann.isSelected() && currentRelay.isSelected()){
ta.append("MANN MORRISON TECHNIQUE + CURRENT RELAY SELECTED \n
CONNECTED DATABASE: "+ tableName + "\n Processing Started\n");
boolean isfault = false;
//double f = Integer.parseInt(sampleFreq.getText());
//double deltaT = 1/f;
final double omega = 314.159265;
try{
    Class.forName("java.sql.Driver");
    Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost/cyberGrid","root","
root");
    st = conn.createStatement();
    String stmt="SELECT It FROM "+ tableName + " ";
    st.executeQuery(stmt);
    String sql=stmt;
    ResultSet rs = st.executeQuery(sql);
    ta.append("Connection Established\n");
    rs.next();
    double prev, curr, nxt;
    prev = rs.getDouble("It");
    rs.next();
    curr = rs.getDouble("It");
    rs.next();
    nxt = rs.getDouble("It");
    rs.next();
    while(isfault != true){
        double diff = (nxt - prev);
        double iDash1 = diff/(0.00004);
        double iDash = iDash1/omega;
        double imsq = Math.pow(iDash, 2);
        double itsq = Math.pow(curr, 2);
        double im = Math.sqrt(imsq + itsq);
        System.out.println("Final Answer "+im);
        ta.append("Im : "+im+ " at It:"+curr+ "\n");

        if(im >= 40.00){
            isfault = true;

```

```

        ta.append("FAULT OCCURED at"+curr+"Im is"+im);
        ta.setForeground(Color.red);
        lbl1.setText("Fault Detected !!!");
        lbl1.setForeground(Color.RED);
        lbl1.setVisible(true);

        String updateFaultSql = "UPDATE " + tableName + "      SET
currentRelay = true, time = CURRENT_TIMESTAMP, Im =" + im + "WHERE It
=" + nxt;

        st.executeUpdate(updateFaultSql);
        new table().setVisible(true);
        break;
    }
    prev = curr;
    curr = nxt;
    nxt = rs.getDouble("It");
    rs.next();
    ta.setForeground(Color.green);
    lbl1.setText("No Fault Detected");
    lbl1.setForeground(Color.GREEN);
    lbl1.setVisible(true);
}
}catch(Exception e){
    System.out.print(e);
}
}

```

2. Code snippet for uploading CSV file upload

```

JFileChooser file = new JFileChooser();
file.showOpenDialog(null);
File f = file.getSelectedFile();
String filename = f.getAbsolutePath();
if(f != null){
    String fileNameWithExtension = f.getName();
    int dotIndex = fileNameWithExtension.lastIndexOf('.');
    String files = (dotIndex == -1) ? fileNameWithExtension :
        fileNameWithExtension.substring(0, dotIndex);
    tn.setText(files.toLowerCase());
    tableName = files.toLowerCase();

    try{
        Class.forName("java.sql.Driver");
        conn =
DriverManager.getConnection("jdbc:mysql://localhost/cyberGrid","root","
root");

        st = conn.createStatement();
        String createTable = "CREATE TABLE IF NOT EXISTS " +
            tableName + " (row_id INT AUTO_INCREMENT PRIMARY
KEY,\n" +

                "sampTime DOUBLE, Vt DOUBLE, It DOUBLE, \n" +
                "time TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,\n" +
                "Vm DOUBLE, Im DOUBLE,\n" +
                "voltageRelay TINYINT(1) DEFAULT 0,\n" +
                "currentRelay TINYINT(1) DEFAULT 0\n" +
                ");";
        st.execute(createTable);
    }
}

```

```

        }catch(Exception e){
            JOptionPane.showMessageDialog(null, e);
        }
    }

    try {
        BufferedReader br = new BufferedReader(new
        FileReader(filename));
        String line;
        while((line = br.readLine())!= null){
            String[] value = line.split(",");
            String sql = "INSERT INTO "+ tableName + " (sampTime,
Vt, It)
VALUES('"+value[1]+'','"+value[2]+'','"+value[3]+'');";
            PreparedStatement pst = conn.prepareStatement(sql);
            pst.executeUpdate();
        }
        br.close();
    }catch(Exception e ) {
        JOptionPane.showMessageDialog(null, e);
    }

    lbl1.setText("Suuccessful Action..");

```

3. Code snippet for plotting graph

```

if(voltRelay.isSelected()){
    try{
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/cyberGrid", "root",
"root");
        st = conn.createStatement();
        String stmt = "SELECT sampTime, Vt, Vm\n" +
            "FROM " + tableName + "\n" +
            "WHERE sampTime <= (\n" +
            "    SELECT MIN(sampTime)\n" +
            "    FROM " + tableName + "\n" +
            "    WHERE voltageRelay = true\n" +
            ")";
        ResultSet rs = st.executeQuery(stmt);
        if (!rs.next()) {
            // The result set is empty, so change the query
            stmt = "SELECT sampTime, Vt from " + tableName;
            rs = st.executeQuery(stmt);
        }
        JDBCCategoryDataset dataset = new JDBCCategoryDataset(conn,
stmt);
        JFreeChart chart = createChart("Over Voltage Relay",
"Sampling
Time", "Current/Voltage",
dataset);
        ChartPanel chartPanel = new ChartPanel(chart);

        // Create and display the main frame
        JFrame frame = new JFrame("Chart Example");

```

```

        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(800, 600);
        frame.setBackground(Color.BLACK);
        frame.setForeground(Color.WHITE);
        frame.getContentPane().add(chartPanel);
        frame.setVisible(true);
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}
else if(currRelay.isSelected()){
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn =
DriverManager.getConnection("jdbc:mysql://localhost/cyberGrid", "root",
"root");

        st = conn.createStatement();
        String stmt = "SELECT sampTime, It, Im\n" +
                        "FROM " + tableName+ "\n" +
                        "WHERE sampTime <= (\n" +
                        "    SELECT MAX(sampTime)\n" +
                        "    FROM "+ tableName + "\n" +
                        "    WHERE currentRelay = true\n" +
                        ")";
        ResultSet rs = st.executeQuery(stmt);
        if (!rs.next()) {
            // The result set is empty, so change the query
            stmt = "SELECT sampTime, It from " + tableName;
            rs = st.executeQuery(stmt);
        }

        JBCCategoryDataset dataset = new
JBCCategoryDataset(conn, stmt);
JFreeChart chart = createChart("Over Current Relay","Sampling
Time","Current/Voltage",dataset);
ChartPanel chartPanel = new ChartPanel(chart);

// Create and display the main frame
JFrame frame = new JFrame("Chart Example");
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
frame.setSize(800, 600);
frame.setBackground(Color.BLACK);
frame.setForeground(Color.WHITE);

// Add the chart panel to the frame
frame.getContentPane().add(chartPanel);
frame.setVisible(true);
}catch (Exception e) {
    JOptionPane.showMessageDialog(null, e);
}
}
}

```

4. Code snippet for CSV file upload to software

```

JFileChooser file = new JFileChooser();
file.showOpenDialog(null);
File f = file.getSelectedFile();
String filename = f.getAbsolutePath();

```

```

        if (f != null) { // Check if a file was
selected
            String fileNameWithExtension =
f.getName();
            int dotIndex =
fileNameWithExtension.lastIndexOf('.');
            String files = (dotIndex == -1) ?
fileNameWithExtension :
fileNameWithExtension.substring(0, dotIndex);
            //System.out.println("File Name without
Extension: " + files);
            tn.setText(files.toLowerCase());
            tableName = files.toLowerCase();
            try {
                Class.forName("java.sql.Driver");
                conn =
DriverManager.getConnection("jdbc:mysql://localhost/c
yberGrid","root","root");
                st = conn.createStatement();
                String createTable = "CREATE
TABLE IF NOT EXISTS "+ tableName + " ( row_id INT
AUTO_INCREMENT PRIMARY KEY,\n" +
                                "
sampTime DOUBLE,\n" +
                                "      Vt
DOUBLE,\n" +
                                "      It
DOUBLE,\n" +
                                "      time
TIMESTAMP DEFAULT CURRENT_TIMESTAMP,\n" +
                                "      Vm
DOUBLE,\n" +
                                "      Im
DOUBLE,\n" +
                                "
voltageRelay TINYINT(1) DEFAULT 0,\n" +
                                "
currentRelay TINYINT(1) DEFAULT 0\n" +
                                ");";
                st.execute(createTable);
            } catch(Exception e){

JOptionPane.showMessageDialog(null, e);
        }
    }

    try {

```

```

        BufferedReader br = new
BufferedReader(new FileReader(filename));
        String line;
        while((line = br.readLine())!= null){
            String[] value = line.split(",");
            System.out.println("Value 1
:"+value[0]+"Value 2 :"+value[1]+"Value 3
:"+value[2]);
            String sql = "INSERT INTO "+
tableName +" (sampTime, Vt, It)
VALUES('"+value[1]+"', '"+value[2]+"', '"+value[3]+"')";
            PreparedStatement pst =
conn.prepareStatement(sql);
            pst.executeUpdate();
        }
        br.close();
    } catch (Exception e ) {
        JOptionPane.showMessageDialog(null, e);
    }

    lbl1.setText("File Successfully uploaded");

```

5. Code snippet for ESP3 Module

```

//-----WiFi Libraries-----
#include <WiFi.h>
#include <WiFiClientSecure.h>

//-----Definations of Pins-----
#define ON_Board_LED 2 //--> Defining an On Board
LED, used for indicators when the process of
connecting to a wifi router
#define V_R 34          //--> Voltage R Phase input
#define V_Y 35          //--> Voltage Y Phase input
#define V_B 32          //--> Voltage B Phase input
#define I_R 33          //--> Current R Phase input
#define I_Y 25          //--> Current Y Phase input
#define I_B 26          //--> Current B Phase input

const char *ssid = "Atharva";
//--> wifi name or SSID.
const char *password = "zbfakrw6tq";
//-->wifi password.
const char *host = "script.google.com";
//--> Host
const int httpsPort = 443;
//-->httpsPort

```

```

String GAS_ID =
"AKfycbyFx49DIY565fNVfkfRcwamUz4b_lAx59_a407--
eewRfpH1JYrQcYLJDeyI3UGfW3BQ"; //--> spreadsheet
script ID
//
https://script.google.com/macros/s/AKfycbyFx49DIY565f
NVfkfRcwamUz4b_lAx59_a407--
eewRfpH1JYrQcYLJDeyI3UGfW3BQ/exec

int Values[6];
WiFiClientSecure client;

void setup()
{
    Serial.begin(115200);
    WiFi.begin(ssid, password); //--> Connect to your
WiFi router
    Serial.println("");

    //-----OnBoard LED-----
    pinMode(ON_Board_LED, OUTPUT);
    digitalWrite(ON_Board_LED, HIGH);

    //-----Waiting for Connection-----
    Serial.print("Connecting");
    while (WiFi.status() != WL_CONNECTED)
    {
        //-----Make the On Board Flashing LED
on the process of connecting to the wifi router.
        Serial.print(".");
        digitalWrite(ON_Board_LED, LOW);
        delay(250);
        digitalWrite(ON_Board_LED, HIGH);
        delay(250);
    }
    //-----
    digitalWrite(ON_Board_LED, HIGH); //--> Turn off
the On Board LED when it is connected to the wifi
router.
    Serial.println("");
    Serial.print("Successfully connected to : ");
    Serial.println(ssid);
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.println();

    client.setInsecure();

```

```

        pinMode(V_R, INPUT);
        pinMode(V_Y, INPUT);
        pinMode(V_B, INPUT);
        pinMode(I_R, INPUT);
        pinMode(I_Y, INPUT);
        pinMode(I_B, INPUT);

        newGame();
    }

    void loop()
    {
        Values[0] = analogRead(V_R);
        Values[1] = analogRead(V_Y);
        Values[2] = analogRead(V_B);
        Values[3] = analogRead(I_R);
        Values[4] = analogRead(I_Y);
        Values[5] = analogRead(I_B);

        sendData();
    }

    void newGame()
    {
        Serial.println("*****");
        Serial.println("NEW SESSION STARTED");
        Serial.print("connecting to ");
        Serial.println(host);

        //-----Connect to Google host---
        ----
        if (!client.connect(host, httpsPort))
        {
            Serial.println("connection failed");
            return;
        }

        //-----Processing data and sending
        data-----

        String url = "/macros/s/" + GAS_ID +
        "/exec?Voltage_R=" + "New_Session" + "&Voltage_Y=" +
        "New_Session" + "&Voltage_B=" + "New_Session" +
        "&Current_R=" + "New_Session" + "&Current_Y=" +
        "New_Session" + "&Current_B=" + "New_Session";
        Serial.print("requesting URL: ");
        Serial.println(url);
    }

```



```

        client.print(String("GET ") + url + "
HTTP/1.1\r\n" +
                    "Host: " + host + "\r\n" +
                    "User-Agent:
BuildFailureDetectorESP8266\r\n" +
                    "Connection: close\r\n\r\n");

        Serial.println("Data sent");
        //-----Checking whether the data was sent
        successfully or not-----

        while (client.connected())
        {
            String line = client.readStringUntil('\n');
            if (line == "\r")
            {
                Serial.println("Headers Received");
                break;
            }
        }

        Serial.println("Closing Connection");
        Serial.println("*****");
        Serial.println();
    }

    void sendData()
    {

        Serial.println("*****");

        Serial.println(Values[0]);
        Serial.println(Values[1]);
        Serial.println(Values[2]);
        Serial.println(Values[3]);
        Serial.println(Values[4]);
        Serial.println(Values[5]);

        Serial.print("connecting to ");
        Serial.println(host);

        //-----Connect to Google host-----
-
        if (!client.connect(host, httpsPort))
        {
            Serial.println("connection failed");
            return;
        }
    }

```

```

//-----Processing data and sending data---
----

String url = "/macros/s/" + GAS_ID +
"/exec?Voltage_R=" + String(Values[0]) +
"&Voltage_Y=" + String(Values[1]) + "&Voltage_B=" +
String(Values[2]) + "&Current_R=" + String(Values[3])
+ "&Current_Y=" + String(Values[4]) + "&Current_B=" +
String(Values[5]);
Serial.print("requesting URL: ");
Serial.println(url);

client.print(String("GET ") + url + "
HTTP/1.1\r\n" +
"Host: " + host + "\r\n" +
"User-Agent:
BuildFailureDetectorESP8266\r\n" +
"Connection: close\r\n\r\n");

Serial.println("Data sent");
//-----Checking whether the data was
sent successfully or not-----

while (client.connected())
{
String line = client.readStringUntil('\n');
if (line == "\r")
{
Serial.println("Headers Received");
break;
}
}

Serial.println("Closing Connection");
Serial.println("*****");
Serial.println();
//-----
}

```

6. Code snippet for API integration

```

function doGet(e) {
Logger.log( JSON.stringify(e) );
var result = 'Ok';
if (e.parameter == 'undefined') {
result = 'No Parameters';
}
else {

```

```

var sheet_id = '1gDATxgjxxCP_uvLcsp-6ZSfKRS7ULYsrCIU-
WLVtpDE'; // Spreadsheet ID
var sheet =
SpreadsheetApp.openById(sheet_id).getActiveSheet();
var newRow = sheet.getLastRow() + 1;
var rowData = [];
var Curr_Date = new Date();
rowData[0] = Curr_Date; // Date in column A
var Curr_Time = Utilities.formatDate(Curr_Date,
"Asia/Kolkata", 'HH:mm:ss');
rowData[1] = Curr_Time; // Time in column B
for (var param in e.parameter) {
Logger.log('In for loop, param=' + param);
var value = stripQuotes(e.parameter[param]);
Logger.log(param + ':' + e.parameter[param]);
switch (param) {
//-----GUN-1-----
-----
case 'Voltage_R':
rowData[2] = value; // Hit by to GUN-1 in column C
result = ' ,Voltage R Phase Entered in Column C';
break;
case 'Voltage_Y':
rowData[3] = value; // Hit to by GUN-1 in column D
result += ' ,Voltage Y Phase Entered in Column D';
break;
//-----GUN-2-----
-----
case 'Voltage_B':
rowData[4] = value; // Hit to by GUN-2 in column D
result += ' ,Voltage B Phase Entered in Column E';
break;
case 'Current_R':
rowData[5] = value; // Hit to by GUN-2 in column D
result += ' ,Current R Phase Entered in Column F';
break;
//-----GUN-3-----
-----
case 'Current_Y':
rowData[6] = value; // Hit to by GUN-3 in column D
result += ' ,Current Y Phase Entered in Column G';
break;
case 'Current_B':
rowData[7] = value; // Hit to by GUN-3 in column D
result += ' ,Current B Phase Entered in Column H';
break;
//-----
-----

```

```
default:
result = "unsupported parameter";
}
}
Logger.log(JSON.stringify(rowData));
var newRange = sheet.getRange(newRow, 1, 1,
rowData.length);
newRange.setValues([rowData]);
}
return ContentService.createTextOutput(result);
}
function stripQuotes( value ) {
return value.replace(/^[\'"]|[\']$/g, "");
}
```

7. References/ Citations

Book

- M. Abdel-Salam, R. Kamel, K. Sayed, and M. Khalaf, "Analysis of Overcurrent Numerical-Relays for Protection of a Stand-alone PV System," Electrical Engineering Department, Faculty of Engineering, Assiut University, Assiut, Egypt; Electrical Engineering 52 Department, College of Engineering and Petroleum, Kuwait University, Kuwait; and Electrical Engineering Department, Faculty of Engineering, Sohag University, Sohag, Egypt.
- B. Kasztenny and D. Finney, "Fundamentals of Distance Protection," General Electric Multilin, [Online]. Available: <https://www.gegridsolutions.com/multilin/notes/fundamentals-of-distance-protection.pdf>.

Conference Paper /Journal Paper

- B. Shah, V. K. Sood, O. Saad, and V. Ramachandran, "Mho Relay for Protection of Series Compensated Line," in IEEE Transactions on Power Delivery, vol. 7, no. 3, pp. 1363- 1370, July 1992.
- A.L.Sayeth Saabith, T. Vinothraj, and MMM. Fareez, "Popular Python Libraries and Their Application Domains," in 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), pp. 1-6, December 2020.
- A.N Jayanthi, P. Rini Emaculate, V. Shanmugapriya, and V. Sarmila, "Fault Detection in Transmission Line Distribution System," in 2022 6th International Conference on Power, Control and Computing Technologies (ICPC2T), pp. 135-139, March 2022.
- E. A. Awada, "Numerical Overcurrent Relay: A Digitizing Element Testing Automation and Simulation Based on Wavelet Transform," in 2021 8th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 1155-1160, April 2021.
- K. K. Joshi and M. R. Hans, "Development of Mho Type Distance Relay for Protection of Long Transmission Line using Matlab/Simulink Environment," in 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 978-982, March 2019.
- S. S. R., M. N., P. V., S. C. Biradar, and T. V., "Study and Analysis of Modern Numerical Relay Compared to Electromechanical Relay for Transmission of Power," in 2020 International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 213-217, December 2020.
- L. K. Sarma, C. N. V. M. S. Teja, M. N. Chaitanya, S. R. Teja, and B. S. Gargeya, "Development of Numerical Relay for Power System Protection Laboratory," in 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 870- 873, May 2018.
- Abdelmoumene and H. Bentarzi, "A Review on Protective Relays' Developments and Trends," in 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 1916-1920, Apr. 2019.

- S. M. Velasco-Gómez, S. M. Pérez-Londoño and J. J. Mora-Flórez, "A Qualitative Comparison of Distance-Based Protection Approaches for Active Distribution Networks," in IEEE Access, vol. 8, pp. 172012-172026, 2020.

Software & Libraries

- JFreeChart – A free and open source library for java to generate static graphs.
- NetBeans 8.2 – Netbeans is a free Integrated Development Environment (IDE) provided by Apache.
- MySQL – A Relational Database Management System(RDBMS).
- Arduino – Arduino IDE

8. Project Submission

Software part of this project is made open source on GitHub and can be forked.

Link: <https://github.com/atharva20-coder/cyberGrid>

9. Personal Details



Dr. Gaurav Goyal
Assistant Professor
Project Guide



Atharva Sandip Joshi
Email id: joshias_1@rknc.edu
Contact No. : 7000523270



Taniya Sanjay Jaiswal
Email id:
jaiswalts@rknc.edu
Contact No.:
8975362347



Jayesh Santosh Dharmale
Email id:
dharmalejs@rknc.edu
Contact No.: 8421731681



Rohan Bhagchand Kukreja
Email id:
kukrejarb@rknc.edu
Contact No.:
9511681521

