Estimation of Open & Short Circuit Fault Detected Distance in the Underground Transmission Cable Using Arduino & GSM Module

By
Md. Rashidur Rahman
16221007
Md. Arafat Ullah
16221036
Nazmul Hasan Sakib
17321018
Johaor Kowsar Arnab
16321164

A thesis submitted to the Department of Electrical and Electronic Engineering in partial fulfillment of the requirements for the degree of
Bachelor of Science in Electrical and Electronic Engineering.

Electrical and Electronic Engineering
Brac University
June, 2021

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material that has been accepted or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help

Student's Full Name & Signature:

Md. Rashidur Rahman 16221007	Mohammad Arafat Ullah 16221036
Nazmul Hasan Sakib 17321018	Johaor Kowsar Arnab 16321164

Approval

The thesis titled "Estimation of Open & Short Circuit Fault Detected Distance in the Underground Transmission Cable Using Arduino & GSM Module" submitted by

1. Md. Rashidur Rahman (16221007)
2. Mohammad Arafat Ullah (16221036)
3. Nazmul Hasan Sakib (17321018)
4. Johaor Kowsar Arnab**(**16321164**)**

Of Summer 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering on 09-06-2021.

Examining Committee:

Supervisor:

_____

Dr. AKM Abdul Malek Azad
Professor, Department of Electrical and Electronic Engineering
Brac University

Program Coordinator:
(Member)

_____

Dr. Abu S. M. Mohsin
Assistant Professor, Department of Electrical and Electronic Engineering
Brac University

Departmental Head:
(chair)

_____

Md. Mosaddequr Rahman
Professor and Chairperson, Department of Electrical and Electronic Engineering
Brac University

Abstract

The purpose of this paper is to develop an IOT based prototype that can detect both the open and short circuit fault location for underground cables by using an Arduino and GSM module. Identification of Short circuit faults can be done by using the concept of OHM'S law. For this purpose, we use the row scanning technique in a three-phase system that continuously scans the three-phase system. We divide the cable into multiple segments where every segment has a particular known resistance value. Depending on the fault location there will be a change in cable voltage which will be detected by the microcontroller and it will display the fault on the LCD and send an SMS through GSM. For open-circuit fault detection, we use a capacitor for each segment of the cable which has a particular time constant for each capacitor. When an open circuit fault occurs in a particular segment, the time constant will change, which will be detected by the Arduino and display the corresponding fault location.

Keywords: short circuit, open circuit, IOT, GSM, OHM'S law, time constant, Arduino

Dedication

We would like to devote our research to our parents who have always inspired us to aspire for achievement and have taken us so far with their never-ending encouragement at all times. We would also like to dedicate this paper to our supervisor for his guidance and support in achieving the final work performance.

Acknowledgment

We would like to thank our supervisor Dr. A K M Abdul Malek Azad, Professor, Dept. of Electrical & Electronic Engineering (EEE), BRAC University, for his motivation, support, guidance, and feedback for the successful completion of the thesis. We would like to acknowledge The Control & Research Centre (CARC) for motivation and guidance in the completion of this project. And last but not the least, we would like to thank our parents, family members and friends for their motivation, guidance, support and sacrifices so that we could finish this thesis without any problems.

Table of Contents

List of Table

List of Figures

List of Acronyms

GSM	Global System for Mobile Communications
IOT	Internet of Things
TC	Time Constant
DC	Direct Current
RC	Resistor Capacitor
LCD	Liquid Crystal Display

Chapter 1

Introduction

1.1 Motivation of work

Underground cable systems have a lot of advantages over overhead cables which is why it is more preferred nowadays. Recently plans have been made to replace overhead electrical cables with underground cables in the next decade or so [1]. While fault occurrence in the underground cable is rare, it is also very hard to detect. In the past, there have been many research papers suggesting the use of various methods like the Murray loop test, Varley loop test or implementation of ohm's law to find the short circuit fault but they have many confinements associated with them [2]. This is where our project aims to minimize the confinements and enhance the already existing projects for better performance. Also, the second part of our project aims to locate the open circuit fault for the three-phase system which is IOT based that has not been previously done.

1.2 Literature review

Most of the underground cable faults occurs due to mechanical failure, degradation of cable sheath, moisture in insulation etc. Several theses, books, journals and articles composed and published by researchers worldwide provide the required knowledge about the already available methods and technological tools. Acknowledges the fact that faults in underground cable lines significantly affect the cable resistance, eventually leading to fatal issues like voltage breakdowns [3]. Therefore testing the underground cables for faults prior to the commencement of their function is sincerely urged.

The research paper titled 'Underground Cable Fault Distance Conveyed over GSM' composed by Nikhil Kumar Sain, Rajesh Kajla[4], discusses a method for locating faults in underground cable lines with the help of microcontrollers. The outcome of this method is the determination of the distance (In kilometers) of the fault from the base station using the theory of the commonly known Ohm's law. The approach is an occurrence of a fault in cable lines adversely affects resistance and voltage in the lines, followed by a variation in current in the lines as well. The cable is represented by resistors, while DC voltage is supplied at one end. Afterward, an analog to voltage converter and a microcontroller performs relevant calculations to detect variations in voltage, which indicate fault. The fault distance is then shown on an LCD screen. Similarly, another paper titled 'Underground Cable Fault Detection using Raspberry Pi and Arduino' prepared by R.K Raghul Mansingh, R.Rajesh, S.Ramasubramani, G.Ramkumar[5] also aims to detect faults in underground cable lines using a different method- the concept of 'Current Transformer' theory. Just like the previous paper, cable faults will cause variations in the voltage depending on the length of the fault in the cable. Voltage variations are triggered by a signal conditioner here and a microcontroller is used as before to perform calculations and display the fault distance on-screen using IOT devices.

Our project uses the basic concept of OHM’S law meaning when the dc voltage is applied at the feeder end of the system through a series of cable lines the current will vary when a fault is generated depending upon the location of the fault. When the current changes the voltage will change proportionally since the length of the cable is known when a fault occurs at any section of the cable we can find the distance where the fault has occurred according to the voltage drop across that segment of the cable. This data is fed to the ADC pin of a microcontroller which is pre-programmed to process the data and find the fault and sends a message through the GSM

module and the exact location through GPS and also which will be displayed in the LCD display.

1.3 Problem Statement

In the underground system, cables are generally directly buried or in ducts. So there is very little chance for the cable to be damaged. But nevertheless, there is always a chance for the cables to get damaged due to various reasons such as insulation failure due to chemical reaction with atmospheric agents, Mechanical damage during transportation or installation. Since the cable is buried underneath the ground visual inspection for locating the fault is not practical. The faults in cables can be classified into three categories-

1. Open circuit fault
2. Short circuit fault
3. Earth fault.

In this paper, we shall be discussing Open and Short circuit faults. Open circuit faults arise due to the breaking of the conductor of a cable which can happen due to excessive mechanical stress or degradation of cable over time where Short circuit faults occur due to an excessive amount of current flowing through the cable when two or more conductors come into physical contact with one another [2]. The Short circuit fault can be detected by applying ohm's law to check for the variations of the voltage in the cable [6]. For the open circuit, we use capacitive time constant measurement to check for the open circuit fault. This project aims to implement the above-mentioned methods using an Arduino to accurately find the type and location of the fault and transmit the data using a GSM module.

1.4 Objective

The project aims to estimate open and short circuit fault distances in the underground transmission cable along with achieving the following purposes -

1. To identify underground problems effortlessly
2. To increase more system efficiency compared to conventional detection approaches
3. Detecting the fault with the precise location
4. To ensure facile maintenance of the system

1.5 Thesis organization

The rest of the dissertation is as follows:

Chapter 2: Short Circuit Fault Detection and Implementation

In this chapter, we have discussed the short circuit fault in underground cables and the fault detection principles of the system. Further, we have presented the circuit diagram for short circuit fault detection and its implementation in both software and hardware.

Chapter 3: Open Circuit Fault Detection and Implementation

In this chapter, we have discussed the open circuit fault in underground cables and the fault detection principles of the system. Further, we have presented the circuit diagram for open circuit fault detection and its implementation.

Chapter 4: Results

In this chapter, the results of software and hardware implementation is presented and analyzed.

Chapter 5: Conclusions and Future Works

In this chapter, the main results of our project are summarized, and some concluding remarks and potential directions for future work scope have been given.

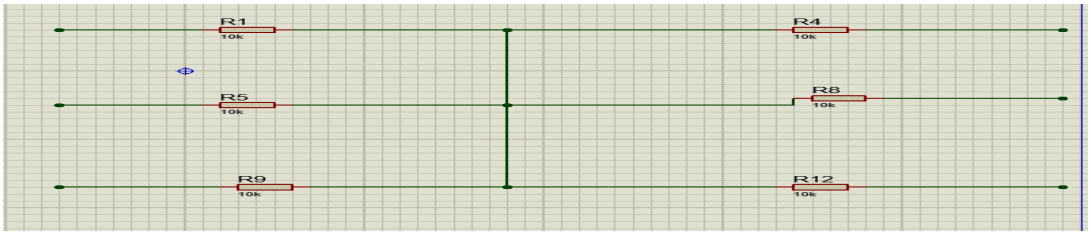
Chapter 2

Short circuit fault detection and implementation

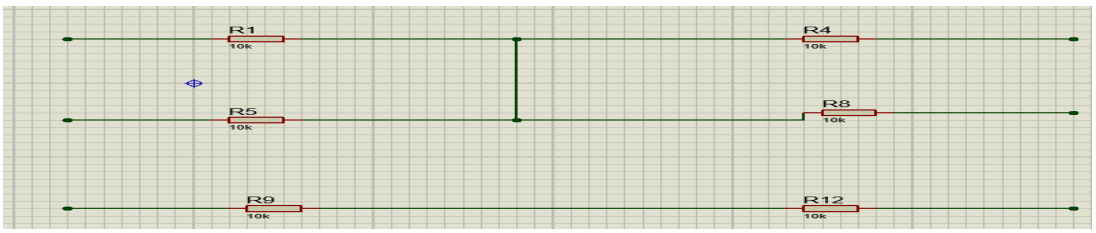
Short Circuit fault can be defined as the amount of excessive current flowing through a conductor. This is the most common type of fault in the power failure and also the most dangerous as excessive amounts of current will lead to overheating of the conductor and can cause severe damage to the cables. Short circuit faults are often associated with the term shunt faults as they occur in parallel branches of a power system. This type of fault occurs due to insulation failure. Short circuit faults can be classified in many categories such as-

1. line to ground fault
2. double line to ground fault
3. line to line fault
4. 3-phase fault [7].

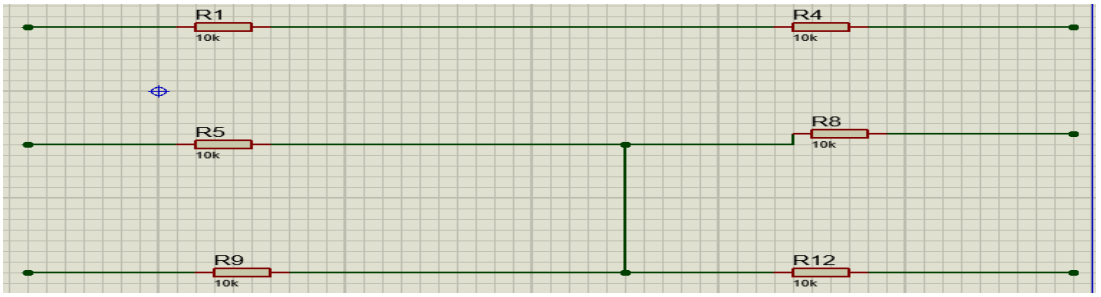
For the sake of our project, we will mainly focus on Phase to Phase fault detection using an Arduino. For fault detection, the concept we are going to use is Ohm's law.



L-L-L



L-L



L-L

Figure 1: L-L-L and L-L short circuit fault

Principle of OHM'S Law:

OHM'S law states that the current flowing through an electrical circuit is directly proportional to the applied voltage and inversely proportional to the resistance of the material.[8]

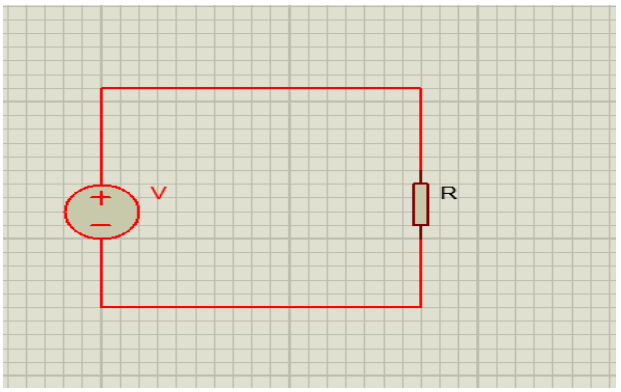


Figure 2: DC circuit connected to a resistor.

So in mathematical terms, the relation between voltage and current can be deduced as:

V=IR

Or I=V/R [8]

Where,

V= Applied voltage

I= The flowing current through the electrical circuit

R= The resistance of the material

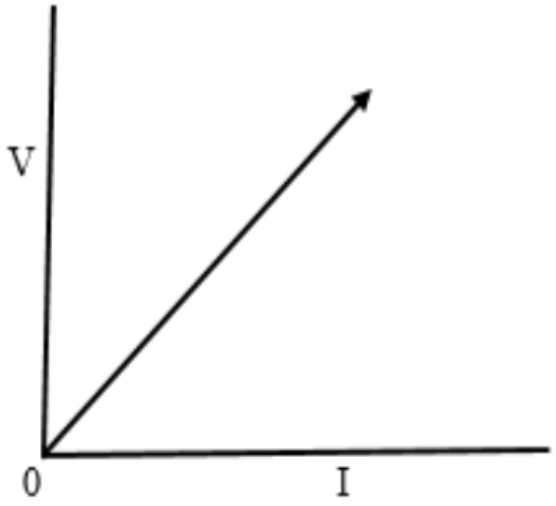


Figure 3: V vs I graph

So since voltage varies with resistance we can calculate these parameters and check for the pre-calculated values of these parameters to find the fault distance location.

2.1 Voltage divider circuit implementation in proteus:

In the circuit below, we used a 5v voltage source across a three-phase system and measured the resistors' voltage. When a fault will be generated by using a switch, the voltmeter in proteus will detect the change in the voltage and display it. For example, when we made a fault at switch1 the voltage changed to 3.3v at the voltmeter due to the voltage divider rule.

Likewise, if we made fault at switch number S6, S11, S12 we got these voltages respectively 4v, 4.3v, and 4.4v. Then we will convert this voltage into a digital value [9].

To convert the analog output into the digital output we use the following form:

Digital output = (2^n *Analogue input voltage) / reference voltage

[10]

Where n= number of bits in ADC converter.

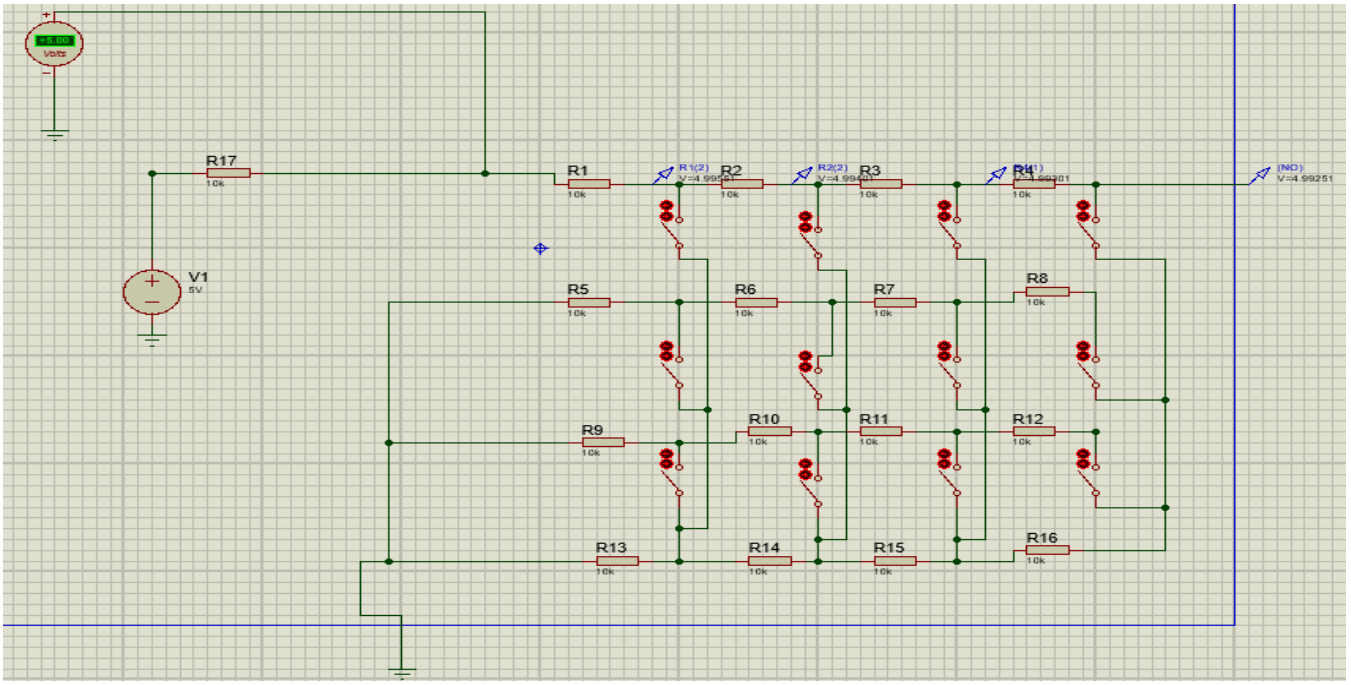


Figure 4: voltage divider law in a 3-phase system.[9]

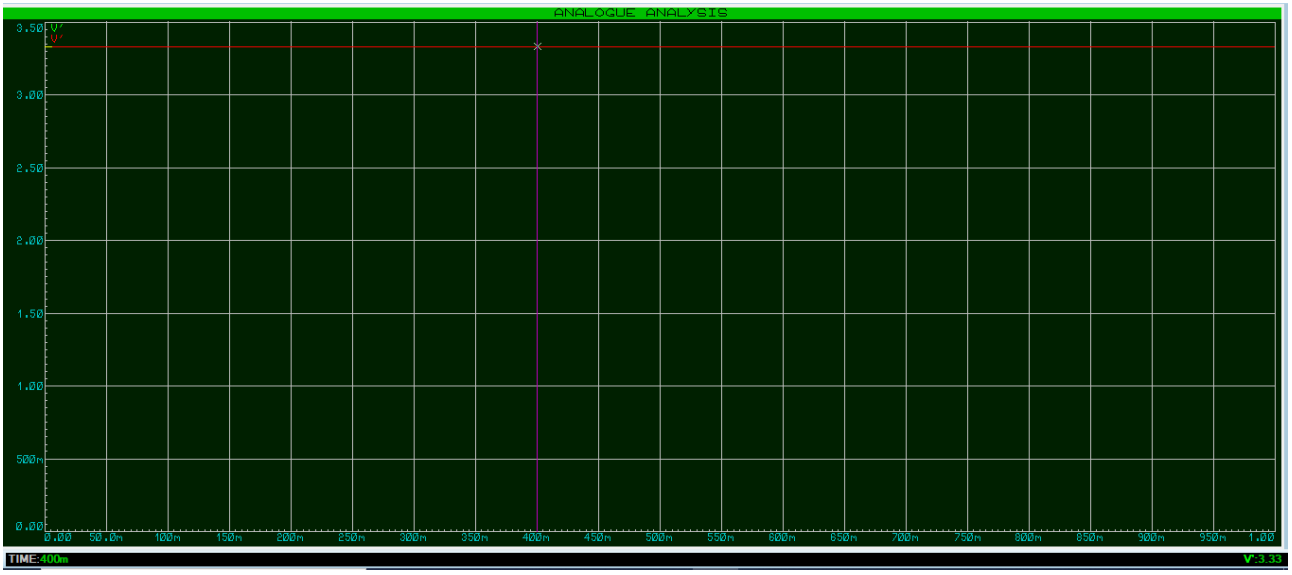


Figure 5: Graph of the voltage drop due to the fault trigged at switch 1

The following voltmeter data is presented in tabular form:

Fault Switch	voltage across the series resistor(v)	ADC output	Distance at which fault occurs
S1	3.3	666	2 KM
S6	4	800	4KM
S11	4.3	860	6KM
S12	4.4	888	8KM

Table 1: Voltmeter Data form Short Circuit Fault and Its Corresponding Location

2.2 Algorithm

Step1: initialize the ports and pin. Initialize the timer, Array, GSM, LCD, ADC functions.

Step2: Initiate infinite while loop. Set relay 1 by setting pin 0.0 high.

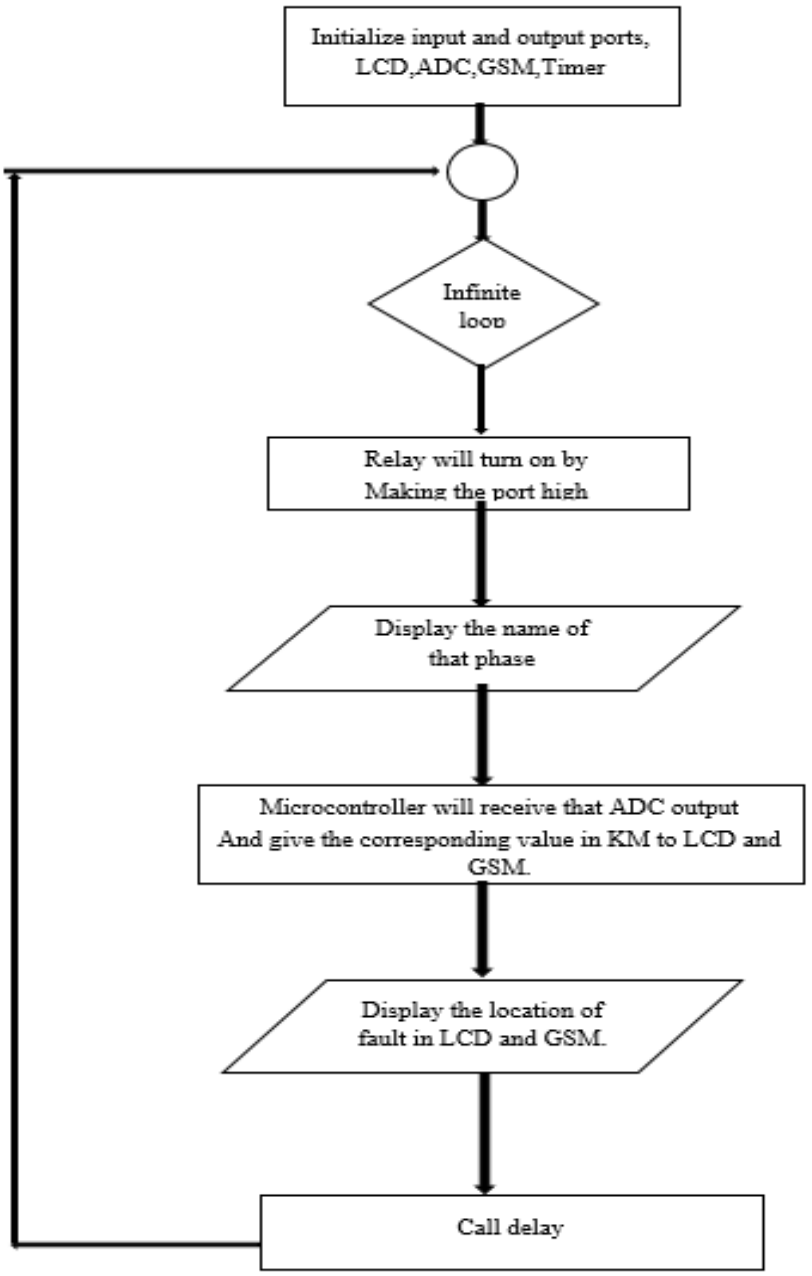
Step3: Display the current status of the phase on LCD and GSM.

Step4: Call ADC function upon ADC output reading through the fault location.

Step5: call delay function.

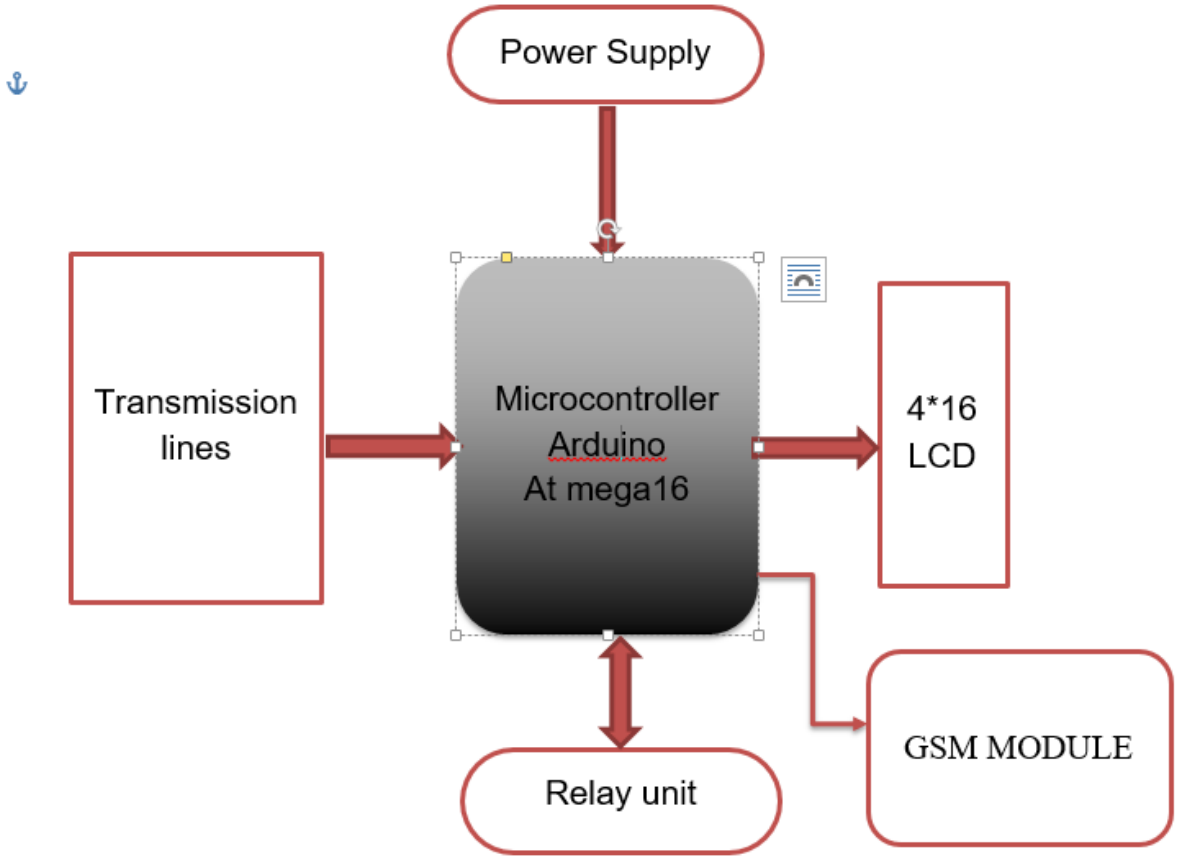
Step6: Repeat step3 to step5 for the remaining two phases.

2.3 Flow Chart



Flow chart for short-circuit fault detection.

Block Diagram:



Block diagram for short-circuit fault detection

2.4 Circuit Diagram for short circuit fault detection

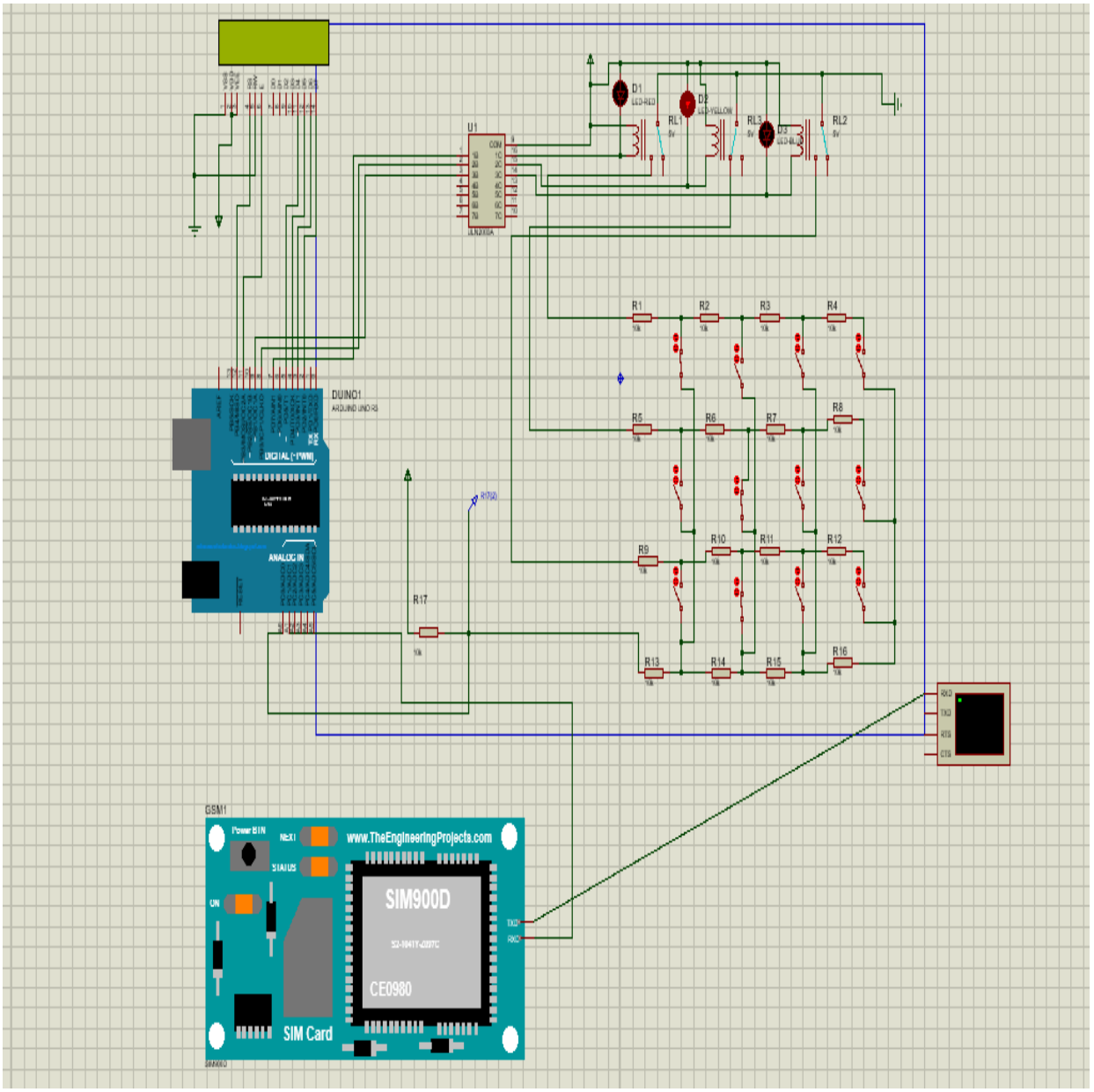


Figure 6: circuit diagram for short-circuit fault detection.

A 5V DC supply is connected to the feeder portion of the circuit, which supplies 5V to R, Y and B phases.12 sets of resistors R1 to R4, R5-R8, and R9-R12 are assembled in series to the corresponding R, Y and B phases. An LCD is connected to the Arduino in pins 12, 11, 5, 4, 3, 2 and a GSM module is connected to the A1 pin of the Arduino. A relay driver ULN2003A controls the 3 sets of relays to pins 7, 6,5 of the Arduino.

2.5 Circuit Working Principle

This project uses the simple concept of OHM'S law meaning when the dc voltage is applied at the feeder end of the system through a series of cable lines the current will vary when a fault is generated depending upon the location of the fault. When the current changes the voltage will change accordingly. Since the length of the cable is known when a fault occurs at any section of the cable we can find the distance where the fault has occurred according to the voltage drop across that segment of the cable. This data is fed to the ADC pin of a microcontroller which is pre-programmed to process the data. After that, It will find the fault and exact location. Finally, it will send a message through the GSM module and also display the fault location on the LCD. [11]

The Proposed system can be divided into four parts.

1. DC power supply source
2. Cable Section
3. Control Section
4. Display Section.

The first section which is the power section consists of a 220 V Ac source that is stepped down to 12 V Ac using a step-down transformer then rectified through a bridge circuit for converting it into DC voltage then the ripples are removed using a filter capacitor. Finally, the Voltage regulator is used to providing a 12V and 5 V constant dc supply for the other sections.

The cable section requires 5V DC as input. This section consists of cables, resistors, and switches. For a three-phase system, 3 sets of cables are used each phase having equivalent resistance of 40 K ohms. Each of the phases is divided into 4 sections denoting equal distance

so that each section of the cable has 10 K ohms resistance. Another additional cable section is included at the bottom of this section to generate the fault when a switch is pressed. Each span of the cable has 8 KM in length so each segment has 2 KM in length. Note that these values of length are arbitrary and can be changed.[6]

The control section includes an Arduino and a relay driver. The Arduino is pre-programmed such that it controls the relay driver. The relay driver is used as the bridge, a connection between the microcontroller and the main cable section. The relay driver controls the relay and when the relays have tripped that phase (any one of R, Y and B) gets activated for fault detection assessment. The relay driver works in a way that operates row scanning from which the data is fed to the ADC pin of Arduino.

The display section shows the output. Here an LCD panel is used to show the current situation of the cables if they have any faults or not. The pre-programmed Arduino process the data after that it will show the current status on the screen of the LCD. In case there is no fault it will show “NF” and in case there is/are any faults it will show the location and distance of the fault in the respective R, Y and B phases. Also, a GSM module is used to send a text when a fault occurs to show the location and distance of the fault in the respective R, Y and B phases.

Chapter 3

Open Circuit Fault detection and implementation

Open circuit fault arises when a portion of the conductor breaks causing the current to stop flowing through the wire. This can be due to cable damage during transport or installation.

These types of faults are referred to as series faults because the fault occurs in series. These types of faults are generally hard to pinpoint unlike short circuit faults because an open conductor does not carry any current so it is hard to find the exact location of the fault. Open circuit faults can be classified into 3 categories:

1. One open conductor fault (When only one conductor is broken)
2. Two open conductor fault (When two conductors are open)
3. Three open conductor fault (When all three conductors are open)

[7]

Given below are picture representation of the above listen open circuit faults:

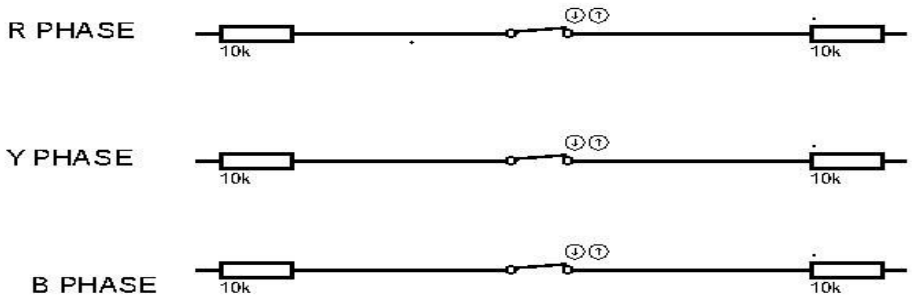


Figure 7: Three-phase cable under normal condition

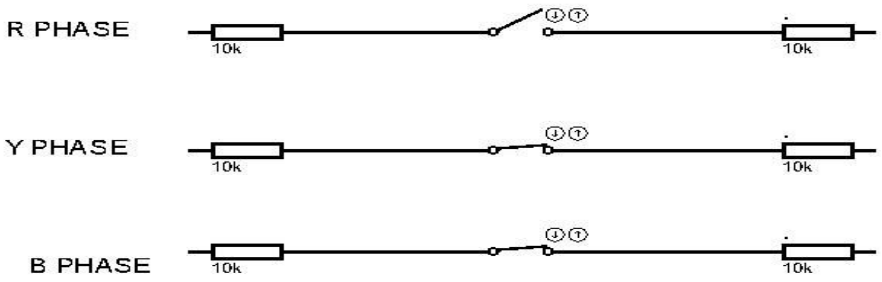


Figure 8:One open conductor fault

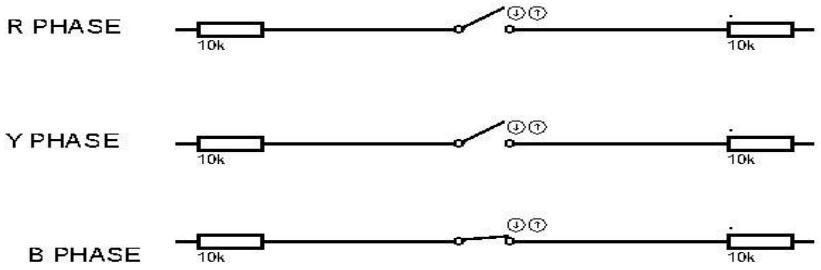


Figure 9 :Two open conductor fault

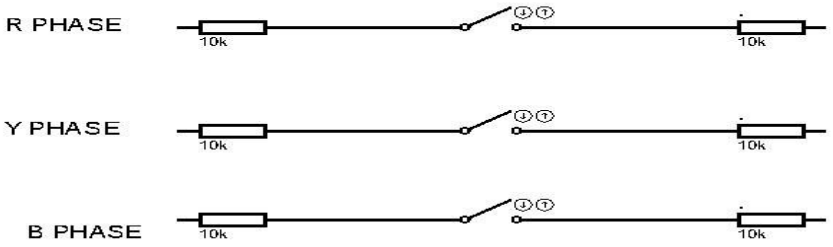


Figure 10: Three open conductor fault

3.1 Explanation of the actual circuit and the working principle

As mentioned above open circuit faults are much difficult to find. To resolve this issue we can use the simple concept of a capacitor time constant to pinpoint the open circuit fault location. If we loop at the equivalent circuit of a transmission line we can analyze it and find that it is mainly comprised of series resistors and inductors and shunt capacitors and conductors (Reciprocal of resistors). For calculation's sake, we ignore the series inductance. We can put the damaged cable under test and perform an RC time constant measuring test. Here the

series resistor will act as the resistor through which the shunt capacitor charges through our microcontroller and the shunt conductance can act like the discharging capacitor through which the capacitor discharges. By measuring and analyzing the time constant we can pinpoint the exact fault location as we had previously done on short circuit fault detection. The principle is the same we divide the cable into multiple sections consisting of resistors and capacitors. Since their values are known we can easily measure the time constant through the charging-discharging cycle with the microcontroller. Since all the time constant measurement is pre-calculated by the microcontroller, we can use conditional statements programmed into the microcontroller to find the fault distance location. If a conductor is open then it will never reach 63.7 percent of its source voltage [12] In other words, it will surpass its characterized Time constant value $\tau=RC$ which the microcontroller can sense and pinpoint the fault location.

Below is a representation of an equivalent circuit of a transmission line:

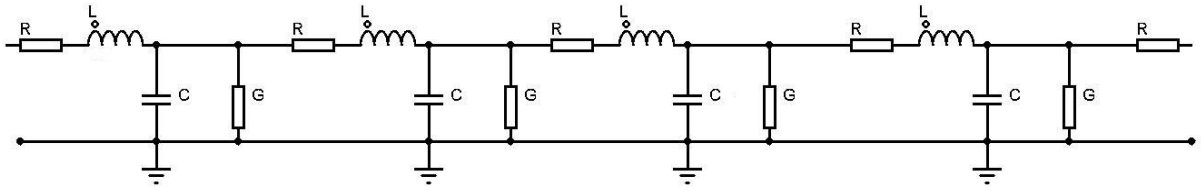


Figure 11: Equivalent circuit of a medium Transmission line With R, L and C parameters.[2]

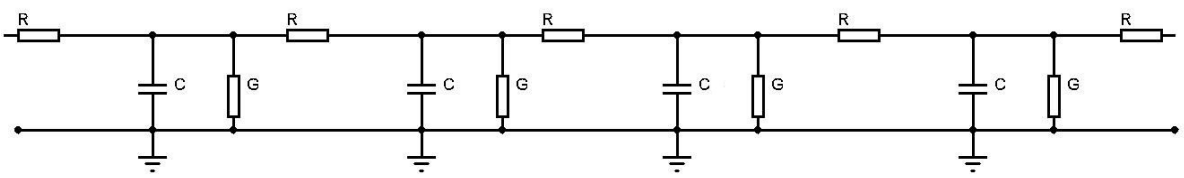


Figure 12:Equivalent circuit of a transmission line ignoring Inductance

3.2 Algorithm

Step 1: Initialize LCD, Charging and discharging pins, declare timers, ADC, LCD function.

Step 2: Make charge pin output and set it as Low, Initialize Serial baud rate.

Step 3: Make charge pin High, Start infinite while loop, and measure time constants of each segment capacitor with Analog pins.

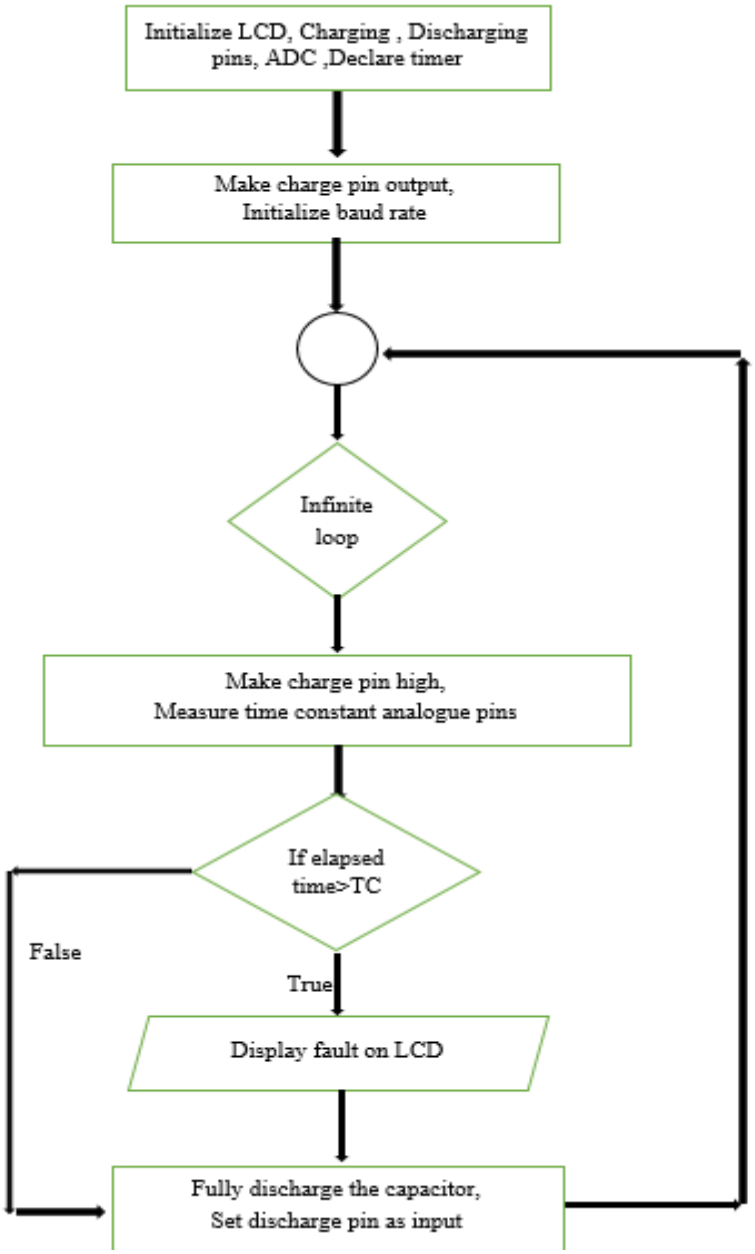
Step 4: Enter conditional if statement to check if the elapsed time (Time constant variable) Has exceeded the actual time constant of that particular capacitor.

Step 5: Set the charging pin to LOW and make discharge pin output.

Step 6: Drain the capacitor through the discharge pin till it fully discharges.

Step 7: Repeat steps 3 to 6.

3.3 Flowchart



Flow chart for open circuit fault detection.

3.4 Concept of RC Time Constant and Simulation:

RC time constant is the product of equivalent resistance and capacitance in a given circuit.

RC time constant can also be defined as the time it takes for a capacitor to reach 63.2% of

The supply voltage of that circuit. The voltage across the capacitor as a function of time can be found by the following Equation: Vc(t)=Vs(1-e^-t/RC)[13]

An example simulation of RC time constant graph of a DC circuit is using LTSPICE given below:

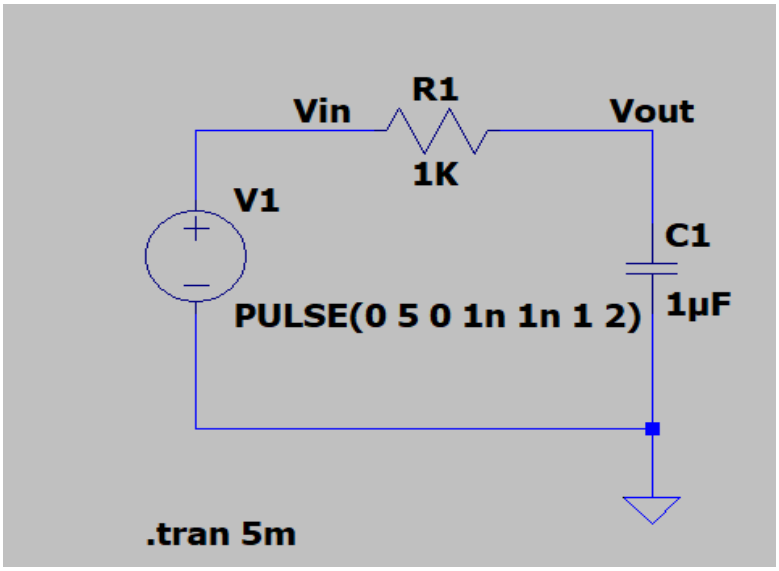


Figure 13:Basic RC circuit in LTSPICE

Here,

R=1000 and C=10^-6

So, T=R*C= 1000*10^-6=1ms

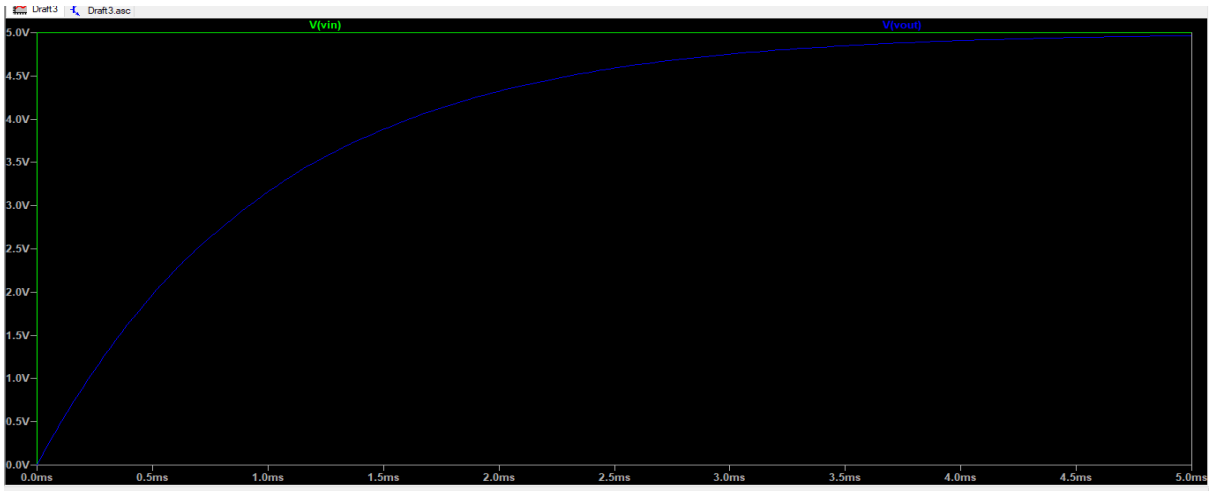


Figure 14 :Charging cycle of the simulated circuit

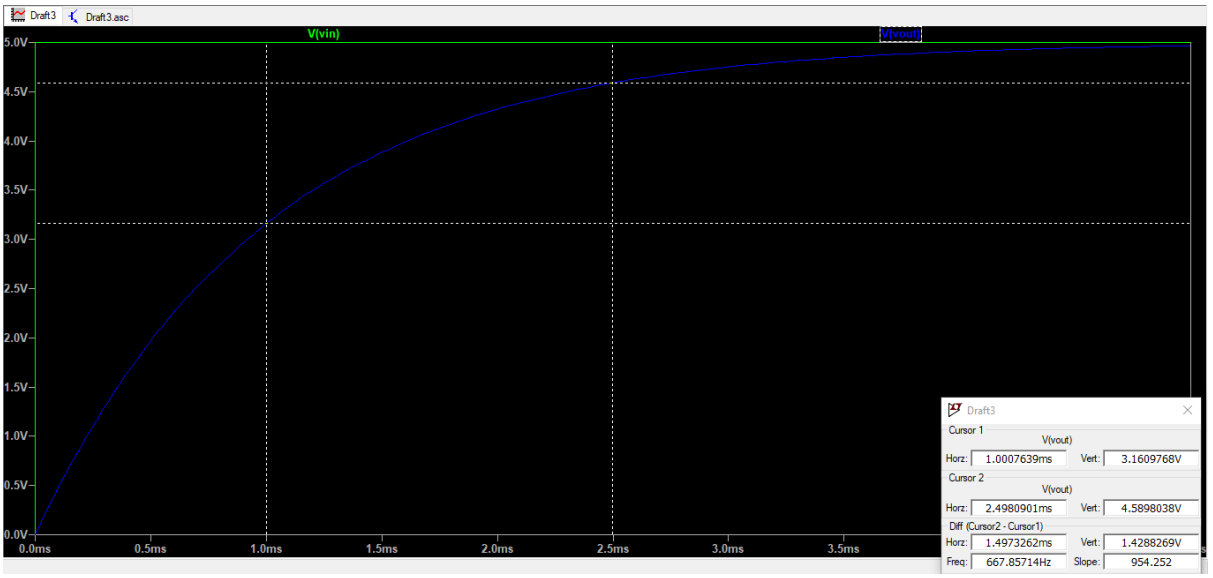


Figure 15: The voltage and time constant value indicated in graph

RC charging voltage Vc(t)=5*(63.2/100)= 3.16 V

Time constant T=RC= 1ms

3.5 RC Time Constant Measurement with Arduino and Fault Detection:

Below is an illustration of how the time constant of an RC circuit can be measured with the timer function millis() which returns the number of milliseconds passed since the Arduino board began running the current program.

In the following circuit R1=10K ohms R2=220 Ohms and C= 100uF

Req=R1+R2=10220 Ohms

Time constant = ReqC =(10*10^3*100*10^-6) = 1.022 Sec or 1022 millisecond

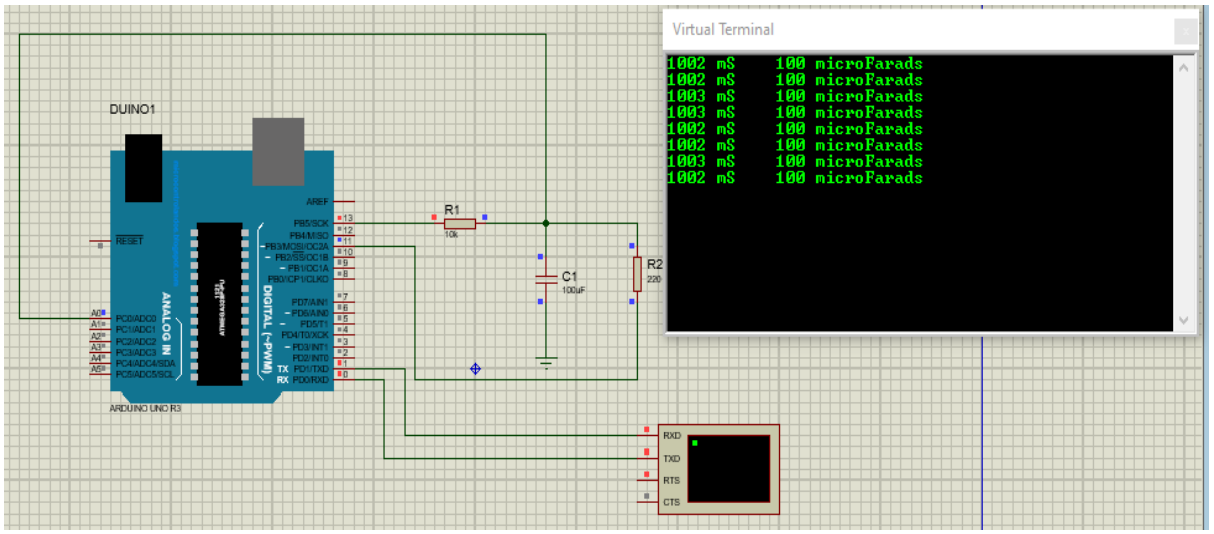


Figure 16:circuit diagram for measuring RC time constant with Arduino

Single Line 2 KM fault detection with Arduino:

The fault detection with Arduino using the principle of measuring time constant with an LCD panel to show the fault location is implemented below:

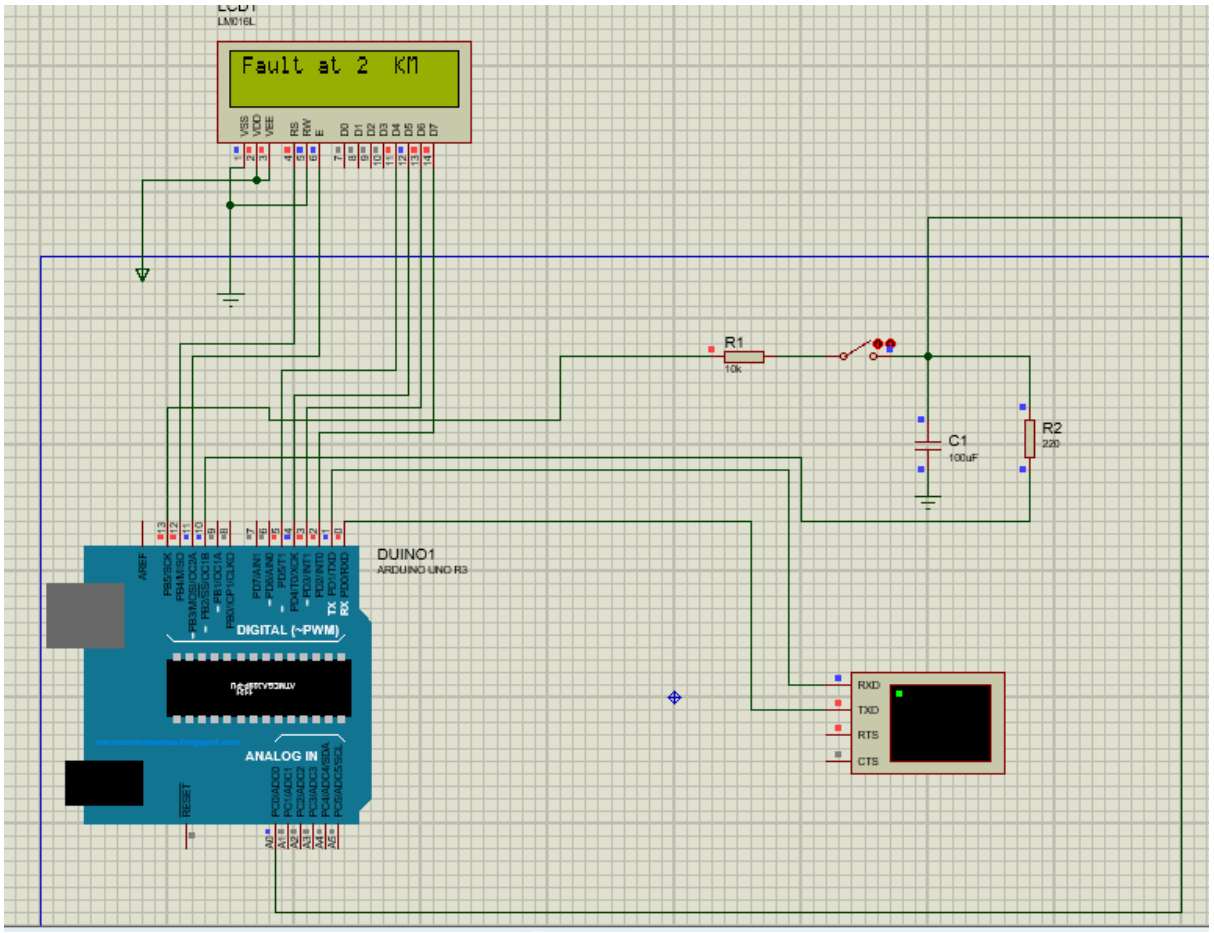


Figure 17:2KM open circuit fault detection with Arduino.

Chapter 4

Result

4.1 SHORT CIRCUIT FAULT DETECTION:

In this chapter, we will discuss the results of underground cable short circuit fault detection we have found from both software and hardware implementation.

I. **SOFTWARE IMPLEMENTATION:**

For short circuit fault detection, we have used proteus software for simulation. After successfully simulating the circuit in proteus, we have found all the possible outcomes of short circuit fault. From there we are showing some of the cases below,

No-fault condition: at first the circuit is at no fault condition where all the switches are open. So the LCD is showing no fault (NF) for all three phases and the virtual terminal (VT) is not showing any messages.

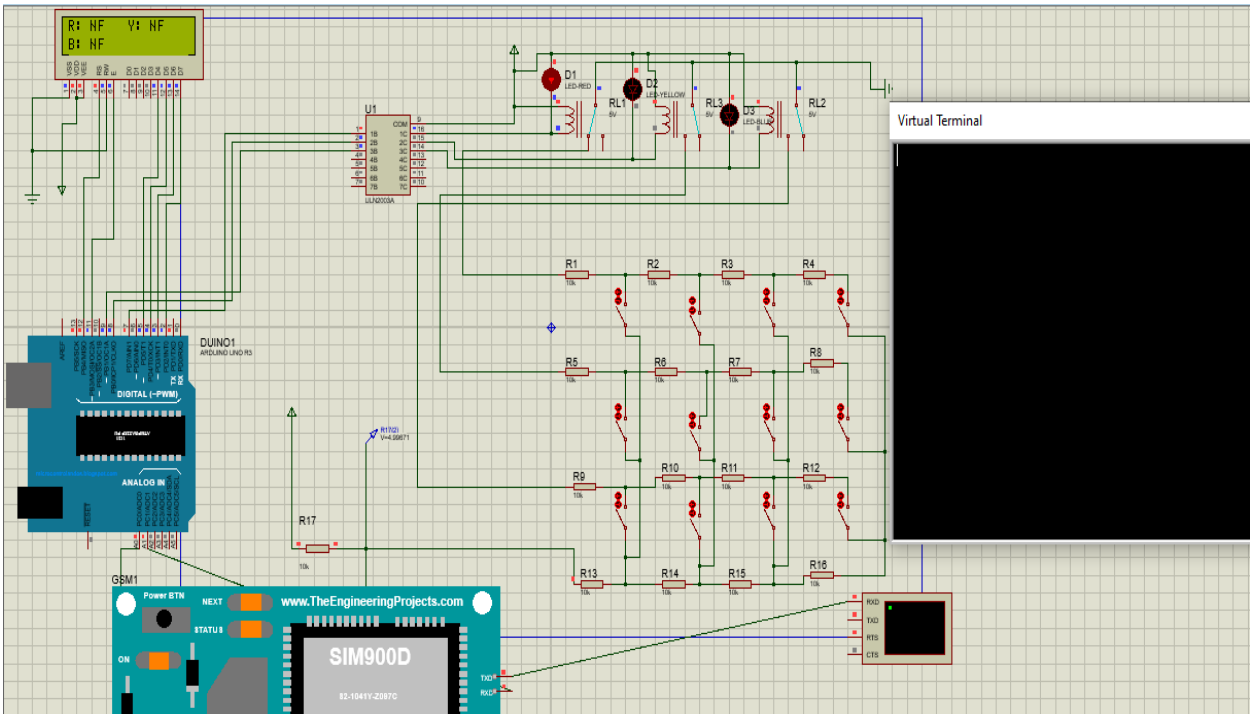


Figure 18: circuit at no fault condition

Fault at 2km in R, Y & B phase: in figure 19, we can see that short circuit fault detected at 2km distance as the switches are closed between R1 & R2, R5 & R6 and R9 & R10 respectively. The voltage changes accordingly to 3.33 Volts which is converted to ADC and sensed by the Arduino which locates the 2 KM fault in the corresponding line. Therefore, the LCD is showing 2km fault at R, Y & B lines respectively and also the virtual terminal is showing a message of the faults.

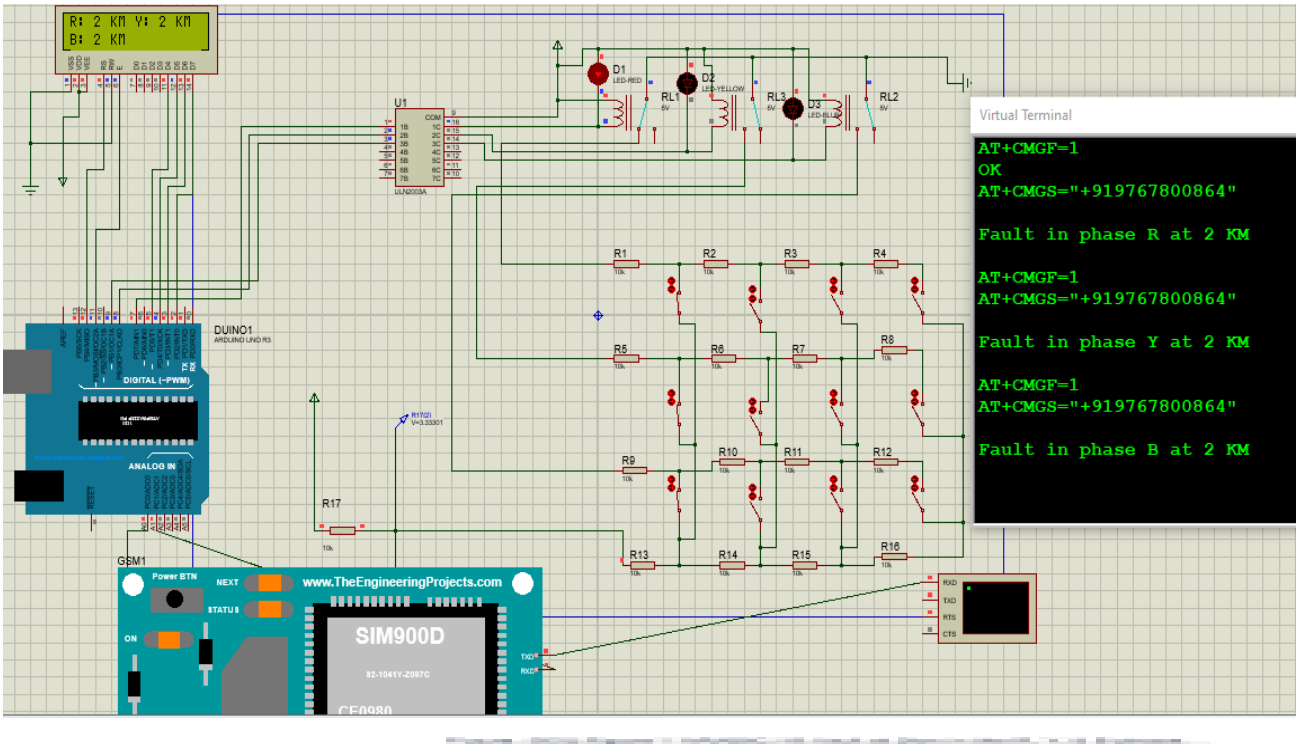


Figure 19*: Fault at 2km in R, Y & B phase*

Fault at 4km in R, Y & B phase: in Figure 20, we can see that short circuit fault detected at 4km distance in R, Y & B phase as the switches are closed between R2 & R3, R6 & R7 and R10 & R11 respectively. Since the above-mentioned switches are closed, the rest of the following resistors R3, R4, R7, R8, R11 and R12 are redundant. The voltage changes accordingly to 4 Volts which is converted to ADC and sensed by the Arduino which locates the 4 KM fault in the corresponding phases. As a result, the fault is detected at 4km in each of the three phases and the faults are shown on the LCD monitor and the Virtual Terminal respectively.

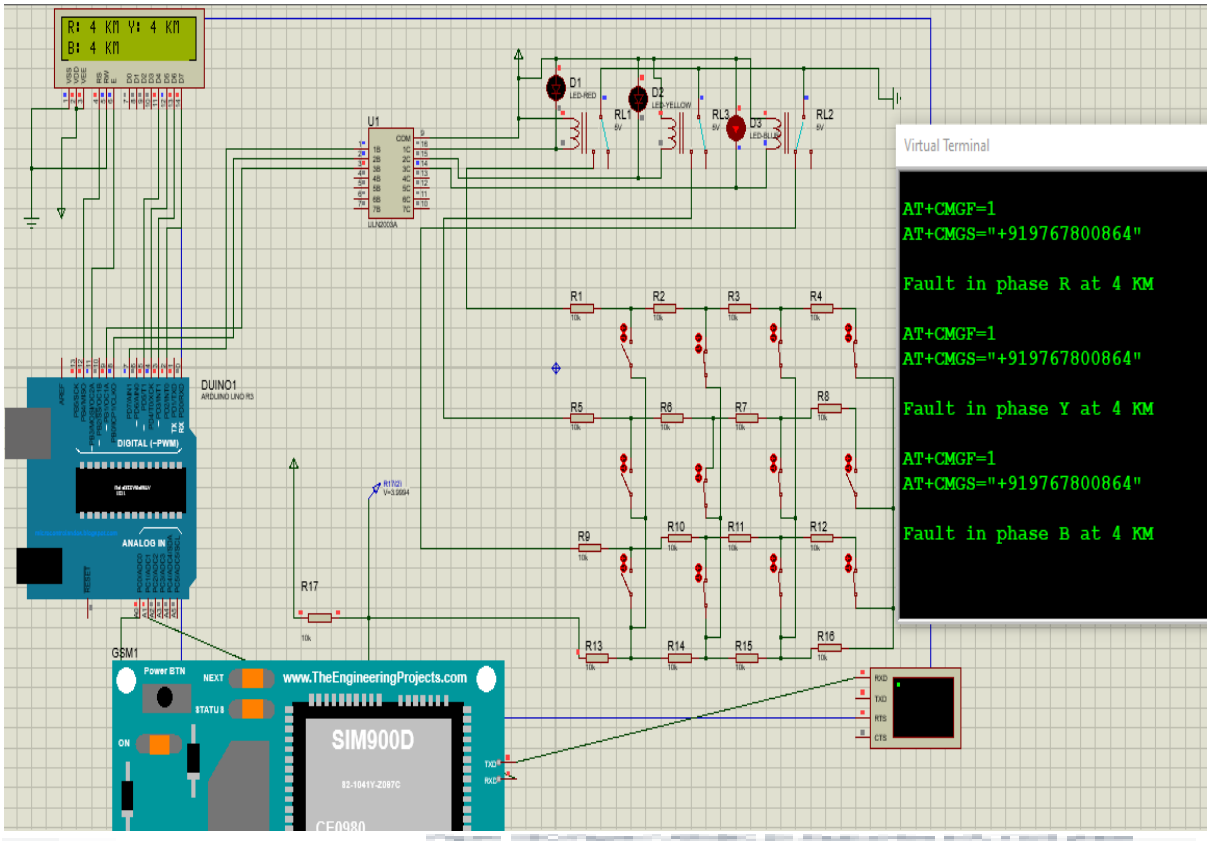


Figure20 : **Fault at 4km in R, Y & B phase**

Fault at a different location: from figure 21, we can see that short circuit fault detected at 4km, 6km & 8km in the R, Y & B phase respectively. The voltage changes accordingly to 3.33,4 and 4.3 volts respectively in R, Y and B phase which is converted to ADC and sensed by the Arduino which locates the 2,4 and 6 KM fault in the corresponding phases. The faults are shown on the LCD monitor and the Virtual terminal is showing messages of fault detection sent from the GSM module.

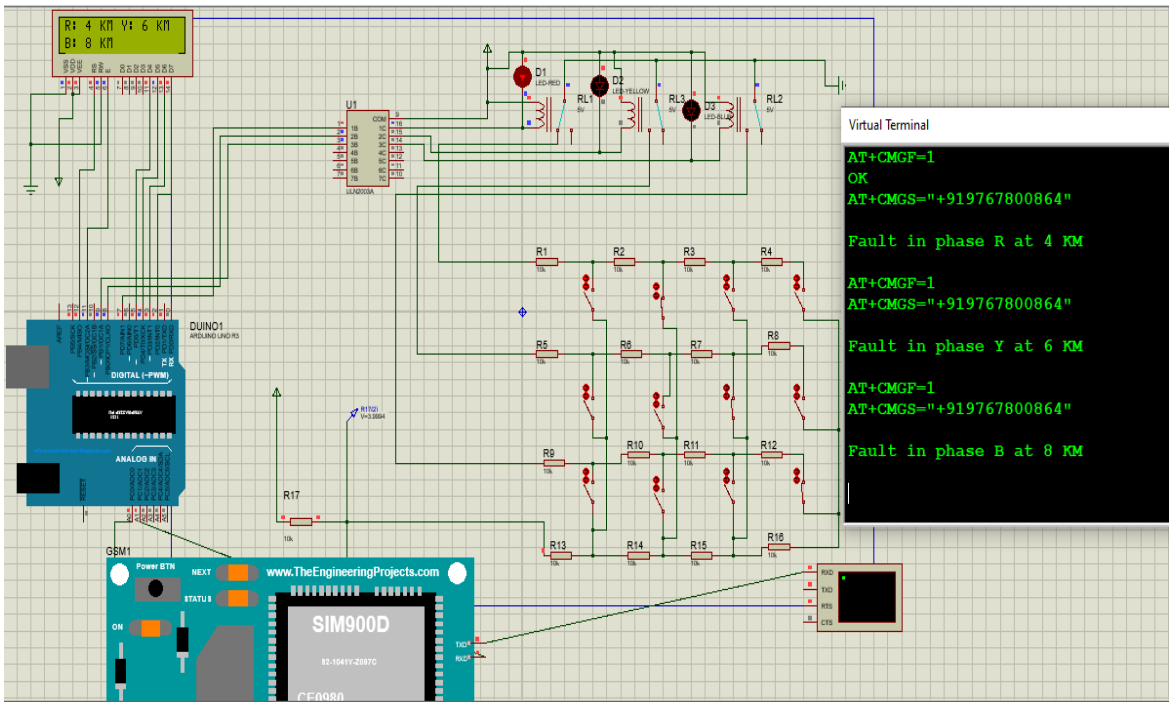


Figure 21: Fault at 4, 6 & 8km in R, Y & B phase respectively.

Fault in R & B phase: from figure 22, we can see that short circuit fault detected in R & B phase at 4km and 2km distance respectively as the switches are closed between R2 & R3 and R9 & R10..The voltage changes accordingly to 4 and 3.33 volts respectively in R and B phase which is converted to ADC and sensed by the Arduino which locates the 4 and 2 KM fault in the corresponding phases. We can see the results are shown on the LCD showing short circuit fault at 4km & 2km in the R and B phase respectively and the Virtual Terminal is showing the message sent by the GSM module. Here, as you can see in the figure below it is not showing the fault of the R line at 8 km distance and also the fault of the B line at 8 km distance. It is because the system will not show more than one fault in a single line at a time. After solving the first fault, the second fault(8km) will show on the LCD.

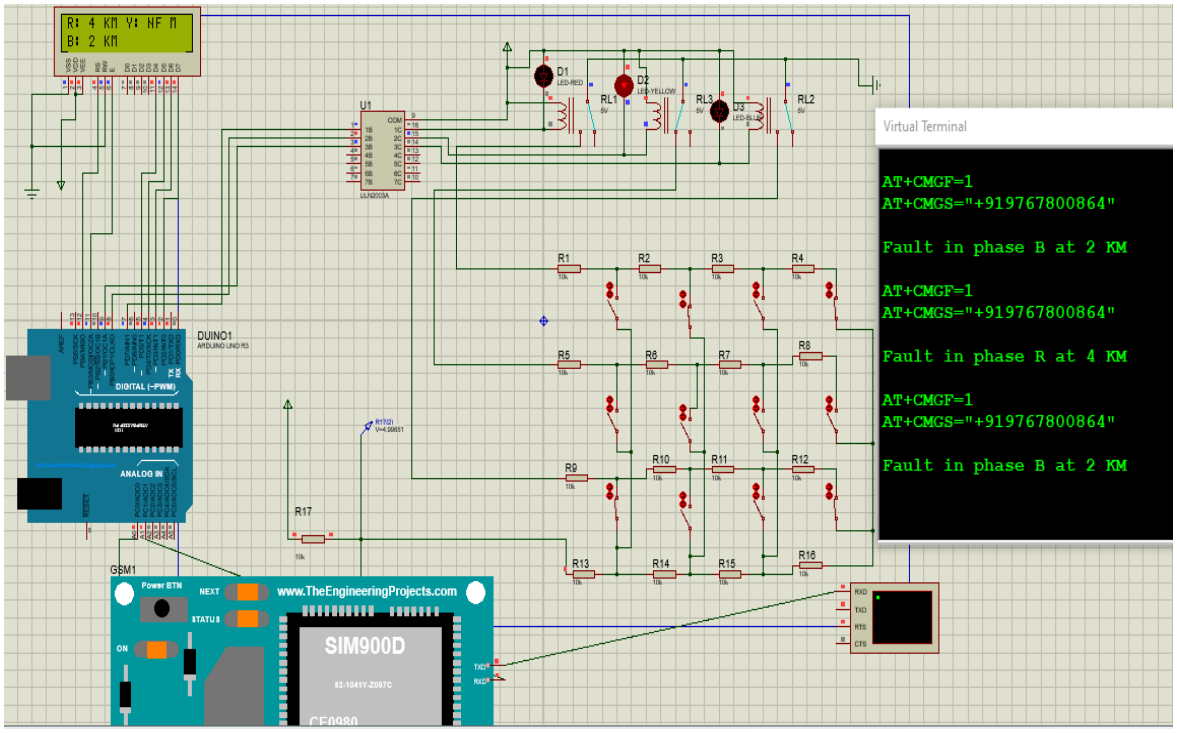


Figure 22: Fault in R & B phase at 4km, 2km respectively.

Hardware Implementation

We have used microcontroller Arduino Atmega16, Power supply, Transmission line, Relay unit, and GSM module as core components for implementing short circuit fault detection. Assembling these components, we have found all the possible outcomes of short circuit fault. From there we are showing some of the cases below –

No-fault condition:

The circuit shown in Figure 23 shows that not a single fault is detected since all the switches are off here. The LCD in the picture shows that there are no faults in any of the three phases: R, Y and B.

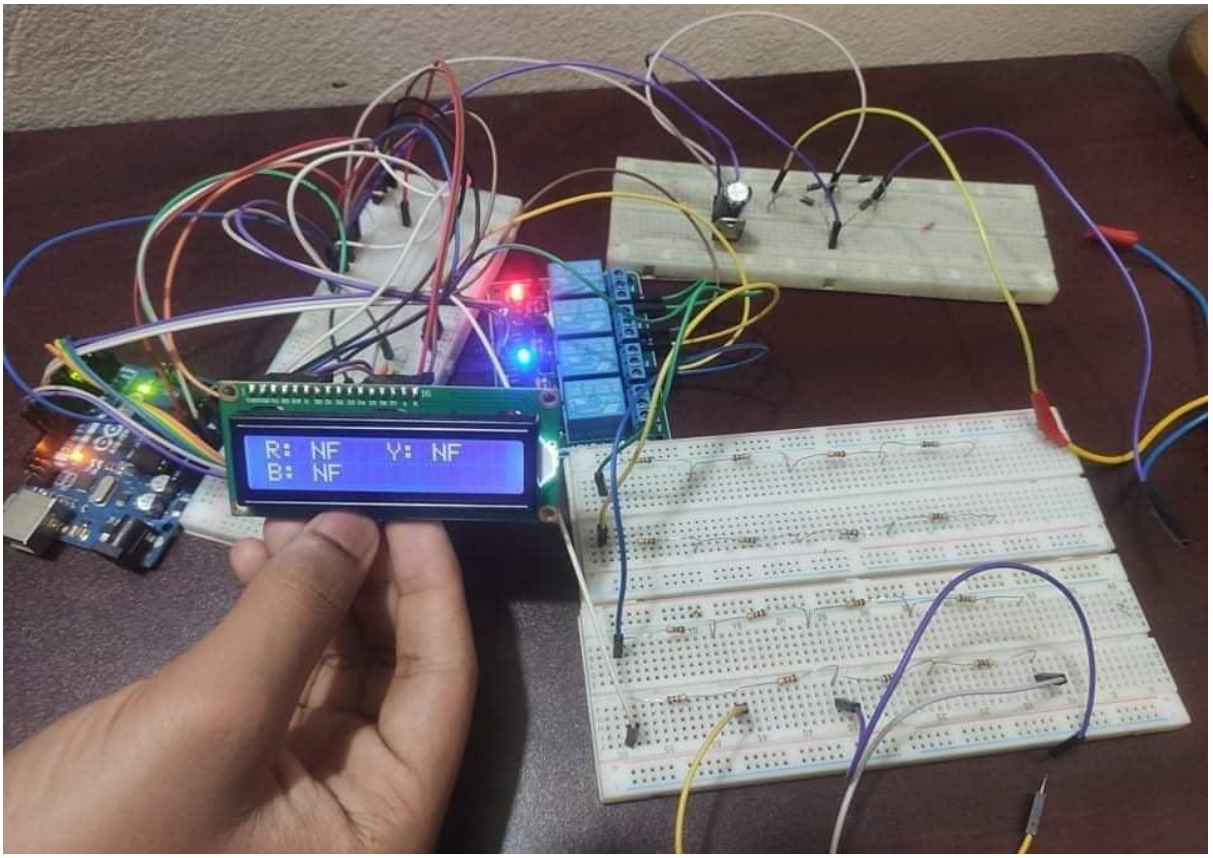


Figure 23:No-Fault Condition

Fault at 6 km in B phase:

In figure 24, we can see that a short circuit fault has been detected in phase B at 6 km when the switches are closed between R9 & R10., The LCD display displays a fault at 6km in the B phase. The voltage across the A_0 pin changes to 1.74 volts and the micro-controller converted this voltage into ADC value . After that micro-controller will show that fault through a pre-programmed set of instructions.

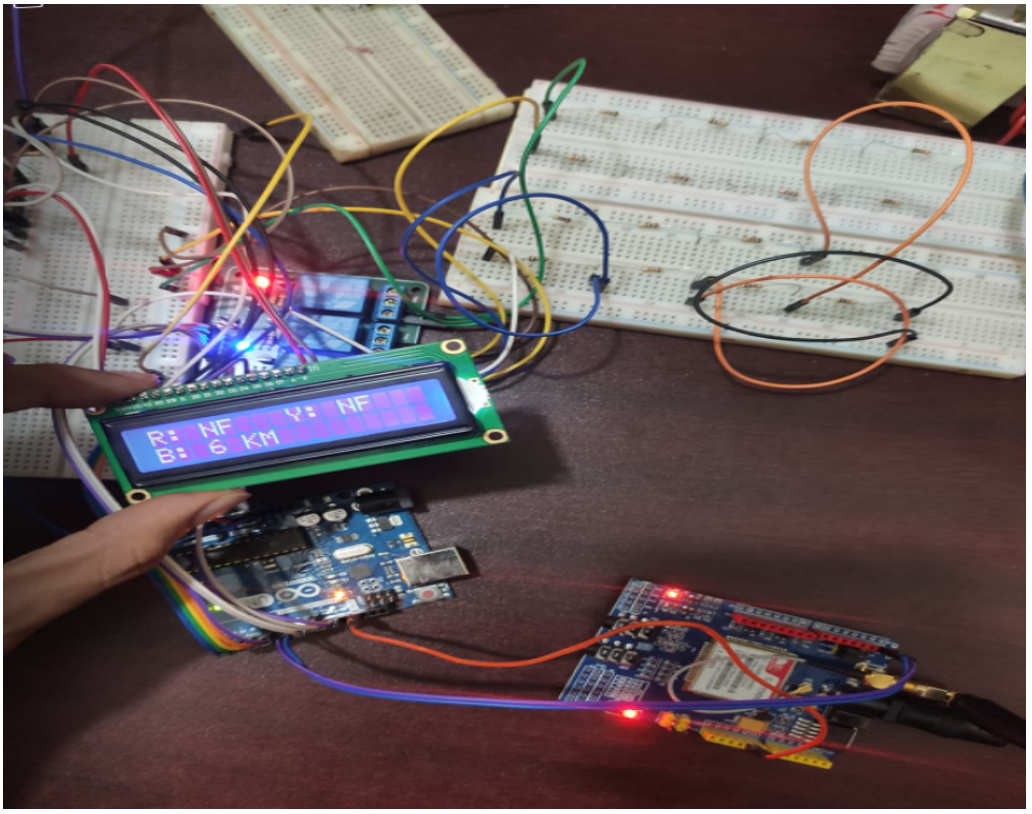


Figure 24: Fault at 6 km in B phase

Fault at 4 km in Y phase:

In figure 25, we can see another result similar to the previous one, except this time the short circuit fault has been detected in phase Y at 4 km when the switches are closed between R5 & R6., the LCD is displaying a fault at 4km in the Y phase. The voltage across the A_0 pin changes to 1.42 volts and the micro-controller converted this voltage into ADC Value . After that micro-controller will show that fault through a pre-programmed set of instructions.

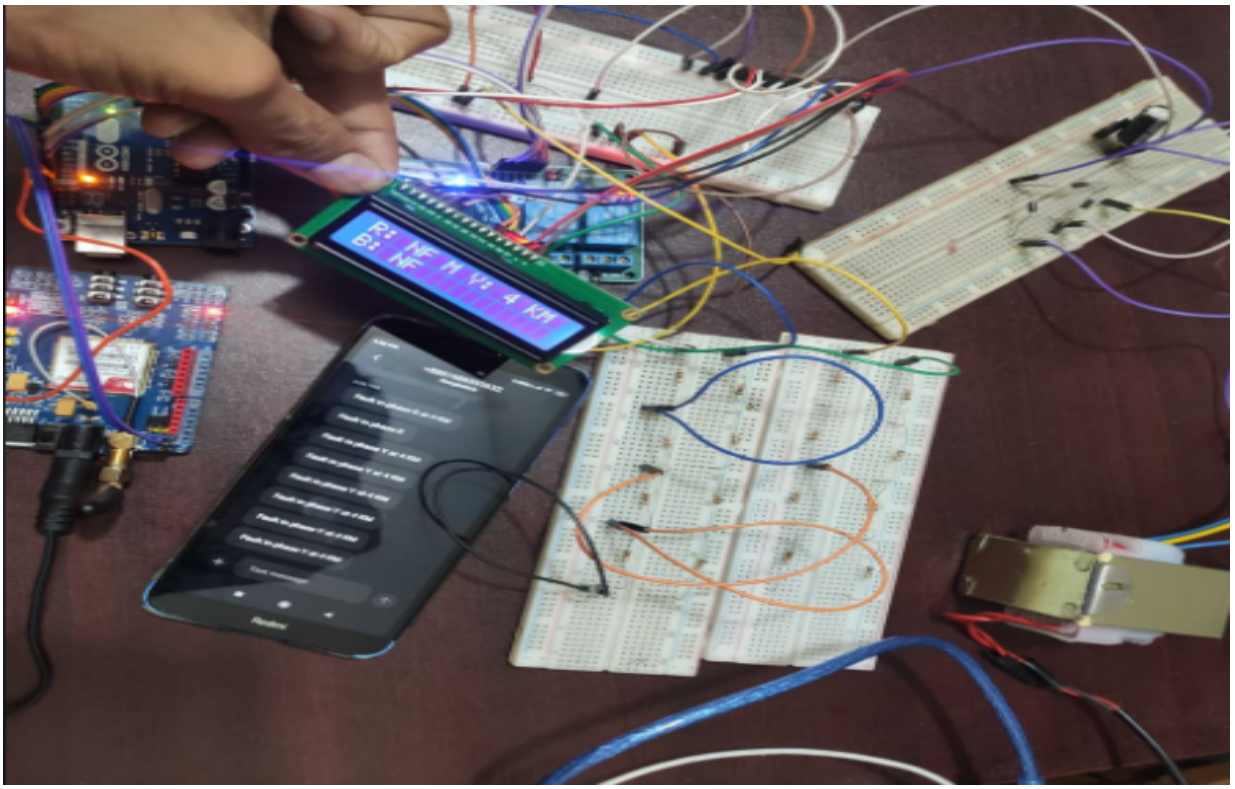


Figure 25: Fault at 4 km in Y phase

Fault at 8 km in B phase:

In figure 26, The LCD shows yet another fault in the B phase. However, this time the fault occurs at a different distance at 8km. when the switches are closed between R11 & R12,

The voltage across the A_0 pin changes to 2.03 volts and the micro-controller converted this voltage into ADC Value. Then micro-controller will show that fault through a pre-programmed set of instructions.

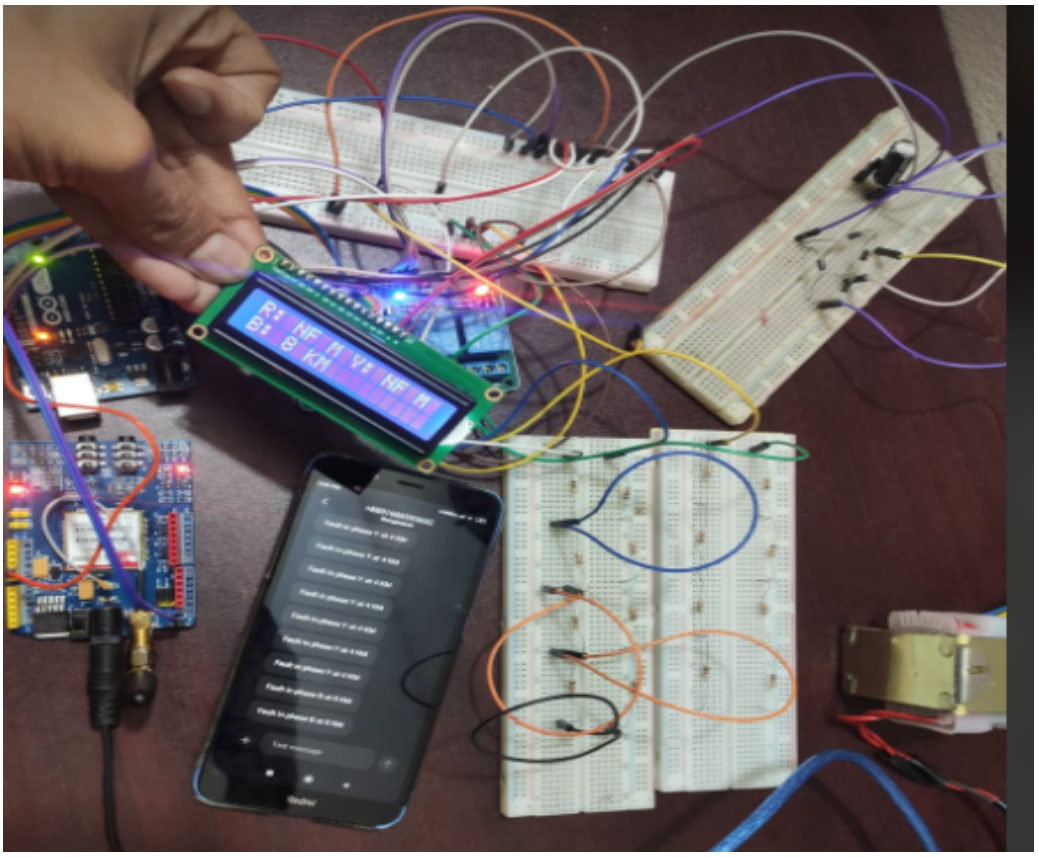


Figure 26: Fault at 8 km in B phase

Fault at 2km in R phase: from figure 27, we can have a clear view that short circuit fault detected at 2km in the R phase as the switches are closed between R1 & R2 and there is no fault Y and B phase. Due to this, the voltage across the A0 pin changes to 0.78 volts and the micro-controller converted this voltage into ADC Value. Then micro-controller will show that fault through a pre-programmed set of instructions.

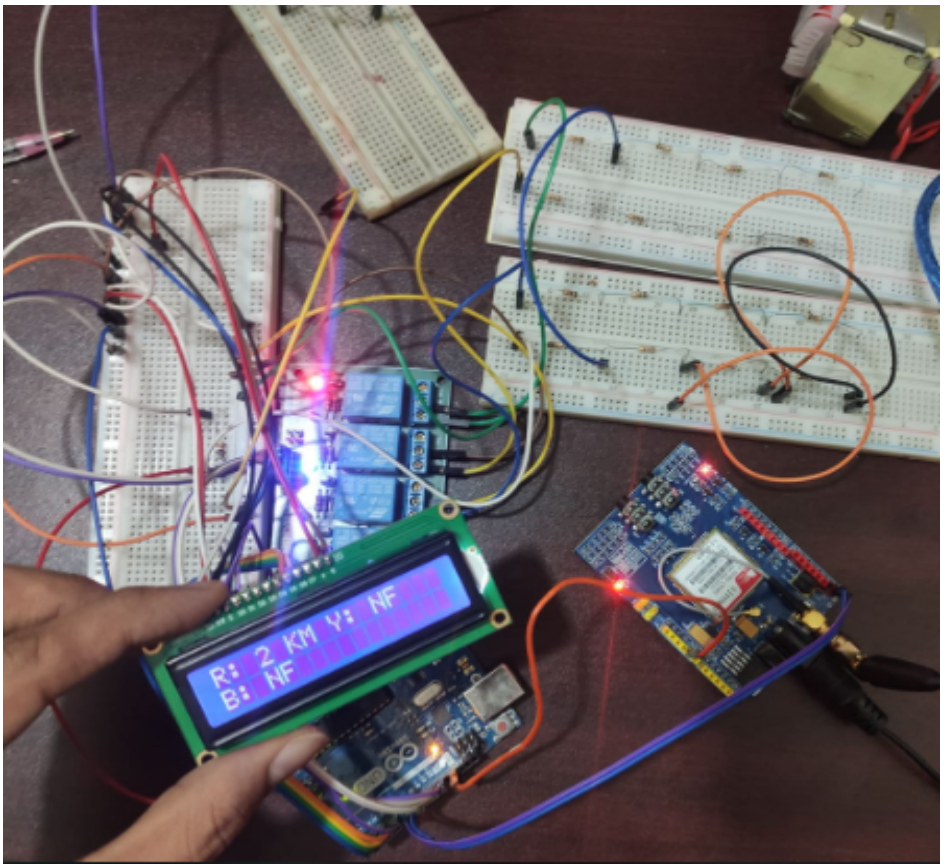


Figure 27: Fault at 2km in R phase

Fault at 2km in R, 6 km in Y & 4 km in B phase: from figure 28, we can see that short circuit fault detected at 2km, 6km & 4km in the R, Y & B phase respectively. The faults are shown on the LCD monitor

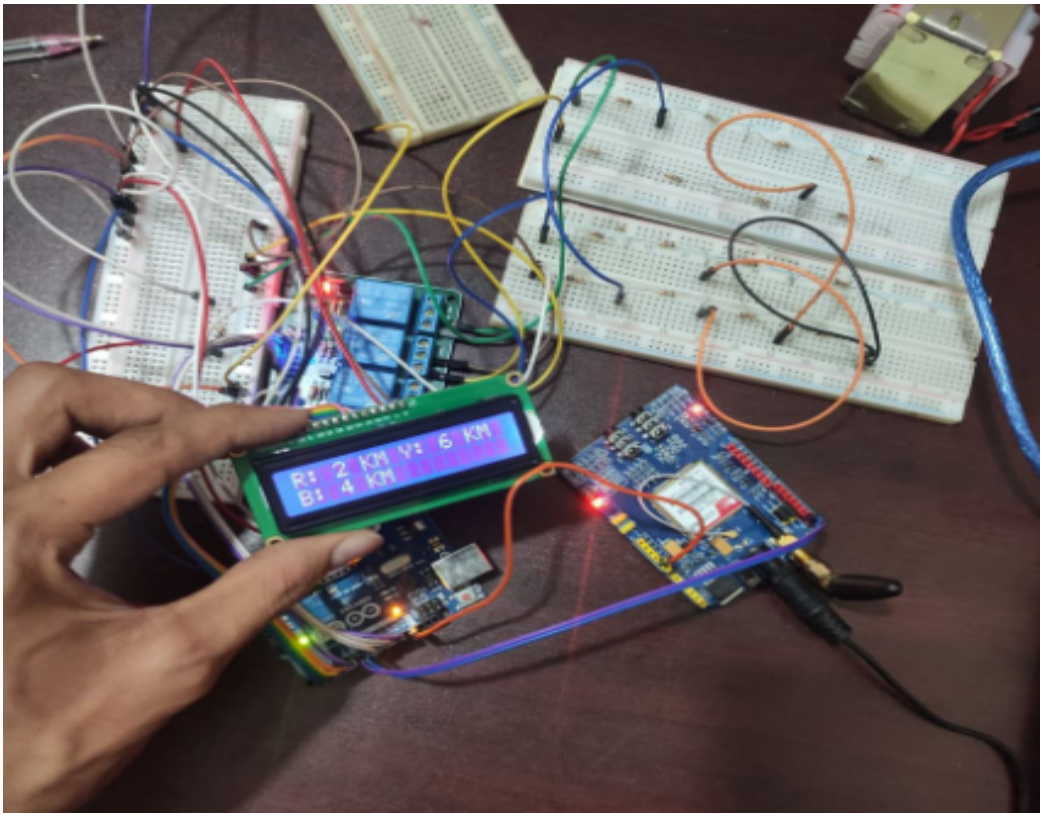


Figure 28: Fault at 2km in R, 6 km in Y & 4 km in the B phase

Fault at 2km in R, 6 km in Y & 8 km in B phase: from figure 29, we can have a clear view that short circuit fault detected at 2km, 6km & 8km in the R, Y & B phase respectively. The faults are shown on the LCD monitor.

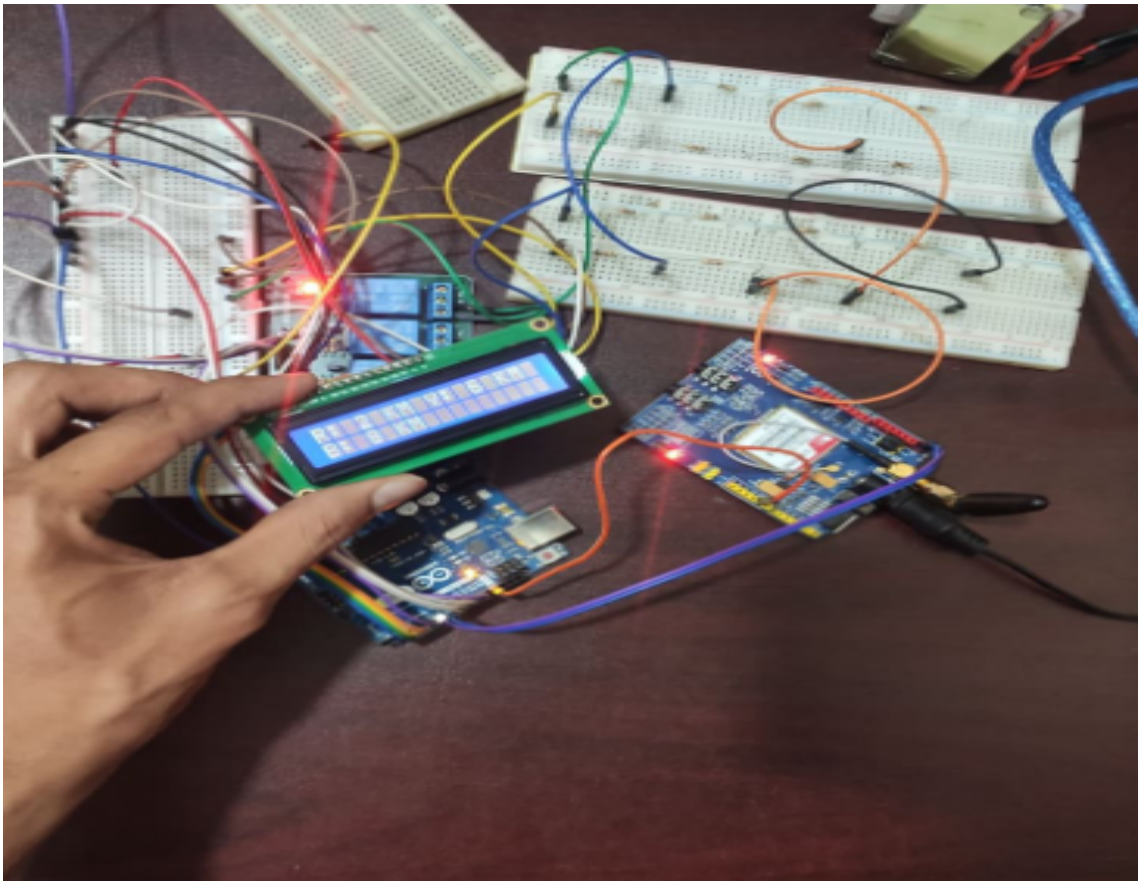


Figure 29: Fault at 2km in R, 6 km in Y & 8 km in the B phase

The following data table is the comparison of voltmeter data between Proteus simulation and hardware implementation results:

Fault Switch	Voltage across the series resistor(v) (Proteus simulation)	Voltage across the series resistor(v) (hardware implementation)	Distance at which fault occurred (KM)
S1	3.3	0.78	2
S6	4	1.42	4
S11	4.3	1.78	6
S12	4.4	2.03	8

Table 2 : Comparison of voltmeter data between Proteus simulation and hardware results.

From Table 2 we can see that there is a difference in voltmeter data between theoretical values used in the Arduino code and the data measured in the hardware prototype.

4.2 Open circuit fault simulations and results

Open circuit fault detection for the 3-phase system:

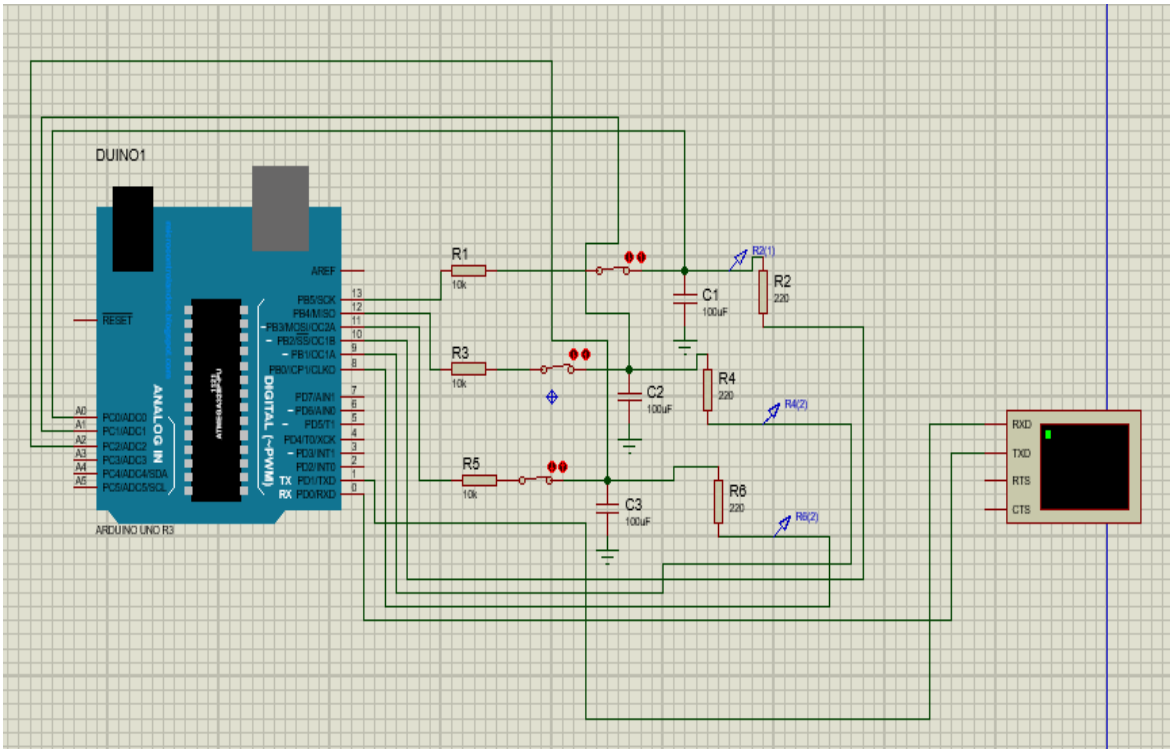


Figure 30:3 Phase fault detection circuit at no-fault condition.

A row scanning technique is used to scan each phase R, Y and B at a time. The Arduino measures the Time Constant across each resistor. If a switch is opened then the capacitor will exceed its time constant and Arduino will immediately sense that and will show which phase is faulty since it is pre-programmed.

When switch-1(Top switch) is opened the Arduino detects that the R phase exceeds its characteristic time constant of 1005 ms. So, the Arduino which is pre-programmed shows the

corresponding R phase fault that is shown on Virtual Terminal R phase fault is shown on the Virtual Terminal

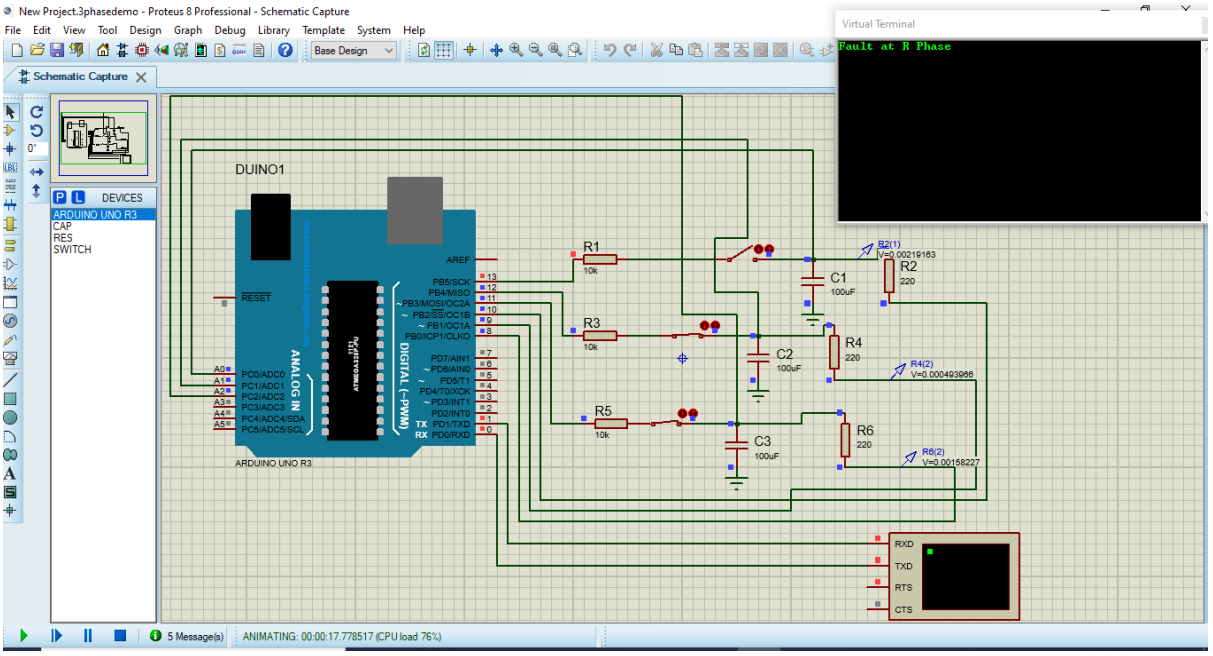


Figure 31: Open circuit Fault at R phase

When switch-2(Middle Switch) is opened Y the Arduino detects that the R phase exceeds its characteristic time constant of 1005 ms. So, the Arduino which is pre-programmed shows the corresponding Y phase fault that is shown on Virtual Terminal.

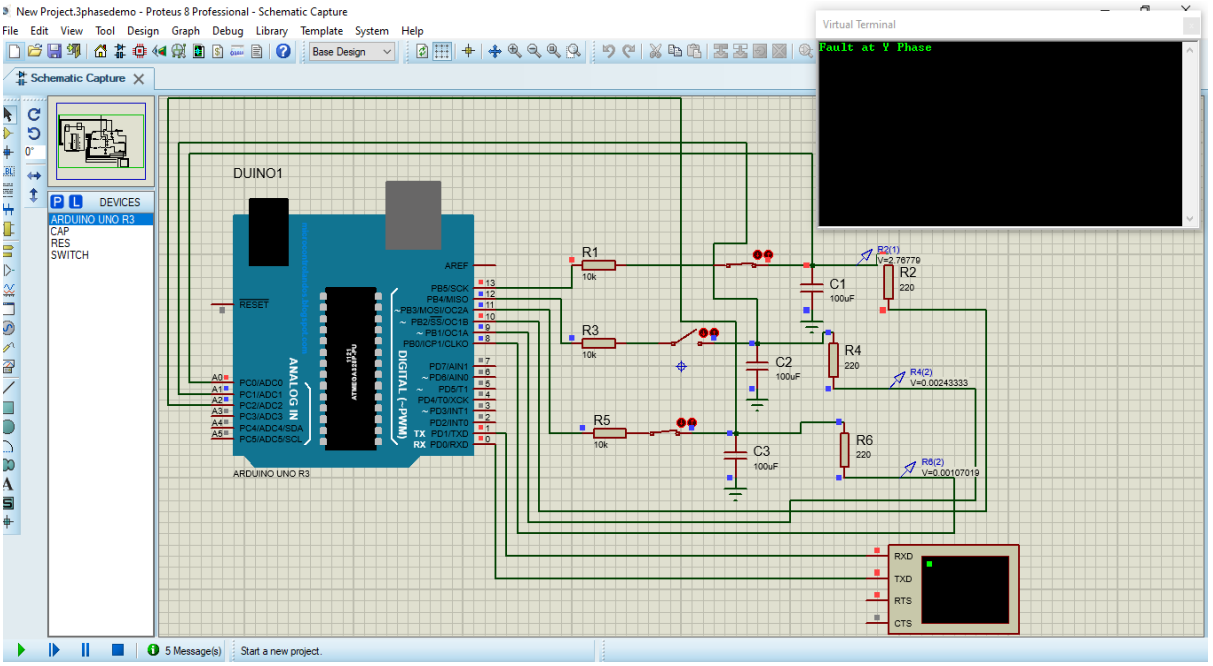


Figure 32: Open circuit Fault at Y Phase

When switch-3(Bottom Switch) is opened the Arduino detects that the B phase exceeds its characteristic time constant of 1005 ms. So, the Arduino which is pre-programmed shows the corresponding B phase fault that is shown on the Virtual Terminal R phase fault is shown on the Virtual Terminal.

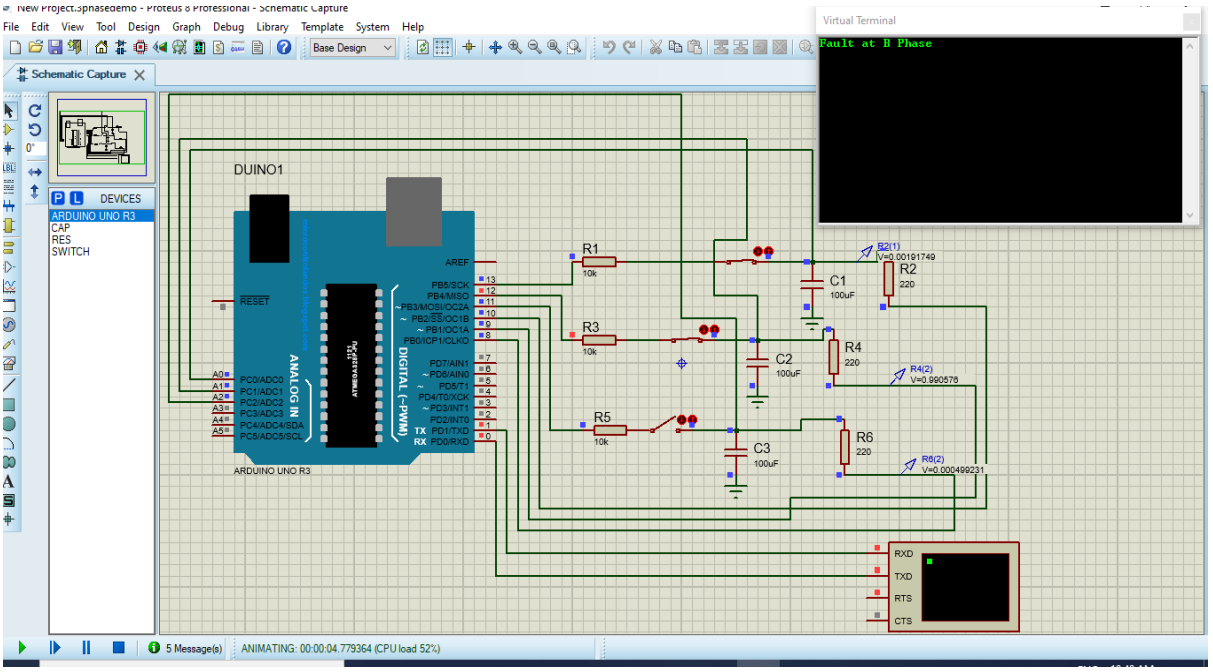


Figure 33: Open circuit Fault at B Phase

When switch-1 and switch-2 both are opened the Arduino detects that the R and Y phase exceeds their characteristic time constant of 1005 ms. So, the Arduino which is pre-programmed shows the corresponding R and Y phase fault that is shown on Virtual Terminal.

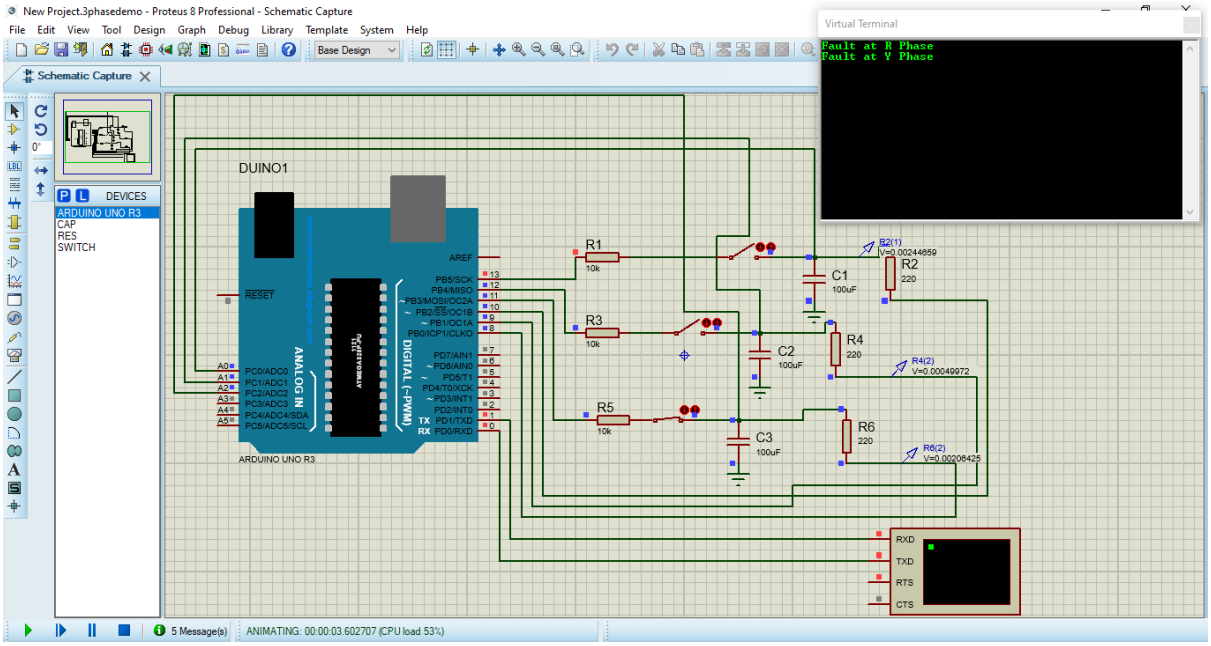


Figure 34 : Open circuit fault at R and Y phase.

When switch-1 and switch-3 both are opened the Arduino detects that the R and B phase exceeds their characteristic time constant of 1005 ms. So, the Arduino which is pre-programmed shows the corresponding R and B phase fault that is shown in Virtual Terminal.

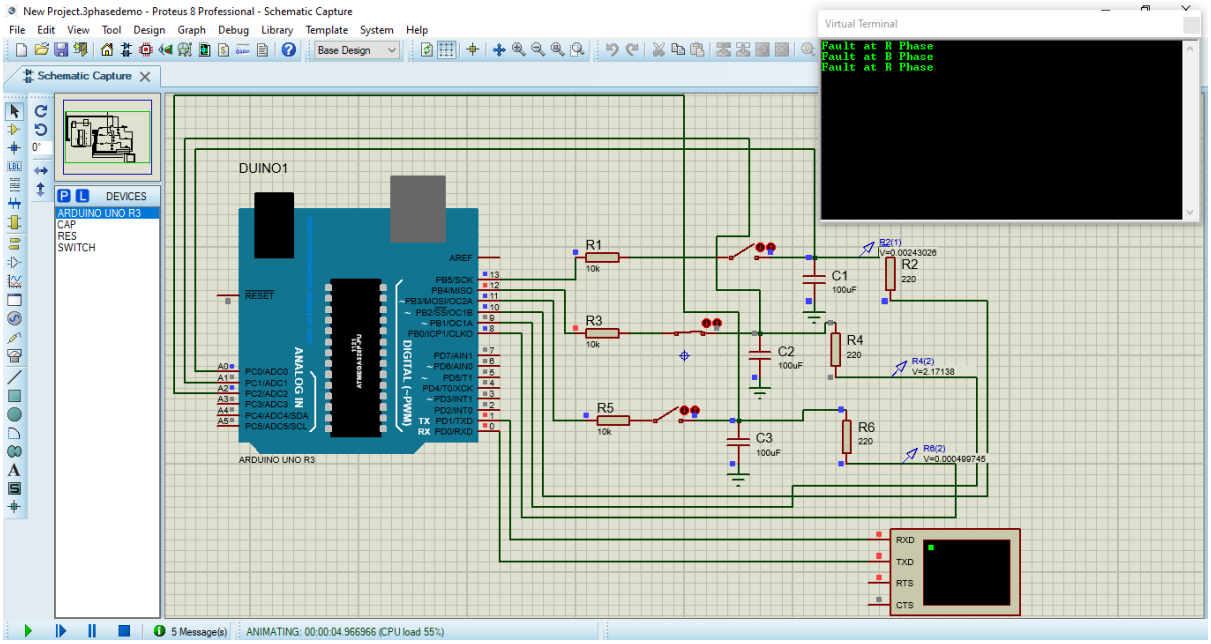


Figure 35: Open circuit fault at R and B phase

When switch-2 and switch-3 both are opened the Arduino detects that the Y and B both phase exceeds their characteristic time constant of 1005 ms. So, the Arduino which is pre-programmed shows the corresponding Y and B phase fault that is shown in Virtual Terminal.

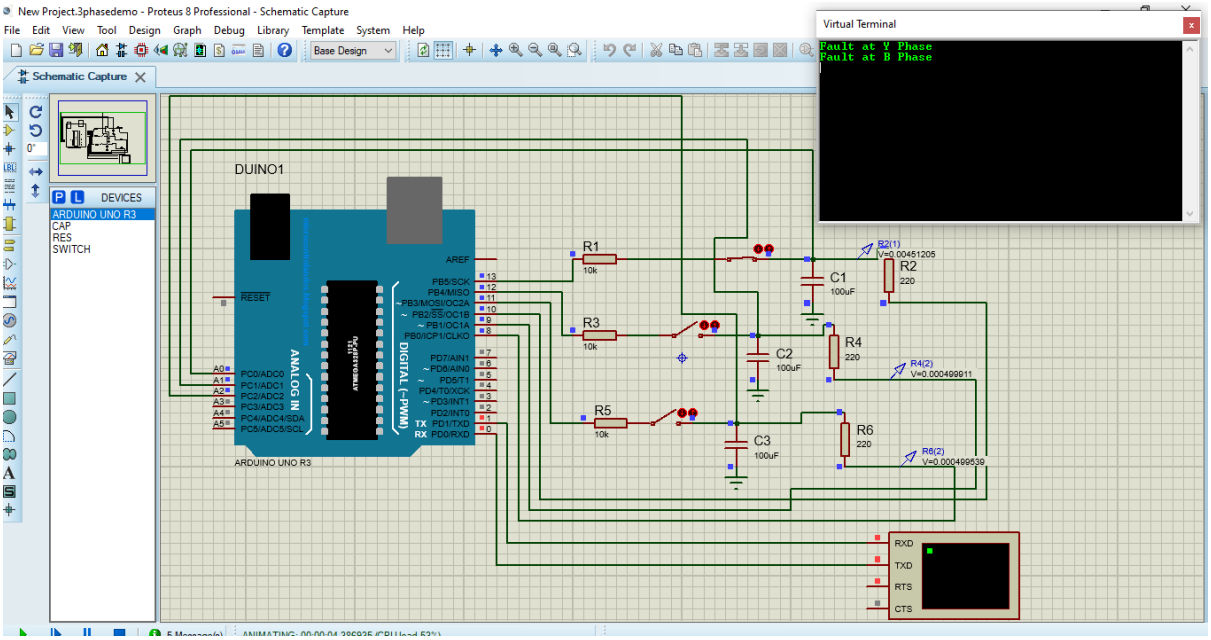


Figure 36: Open circuit fault at Y and B phase

When all three switches are opened both are opened the Arduino detects that the R, Y and B all phase exceeds their characteristic time constant of 1005 ms. So, the Arduino which is pre-programmed shows the corresponding R, Y and B phase fault that is shown on Virtual Terminal.

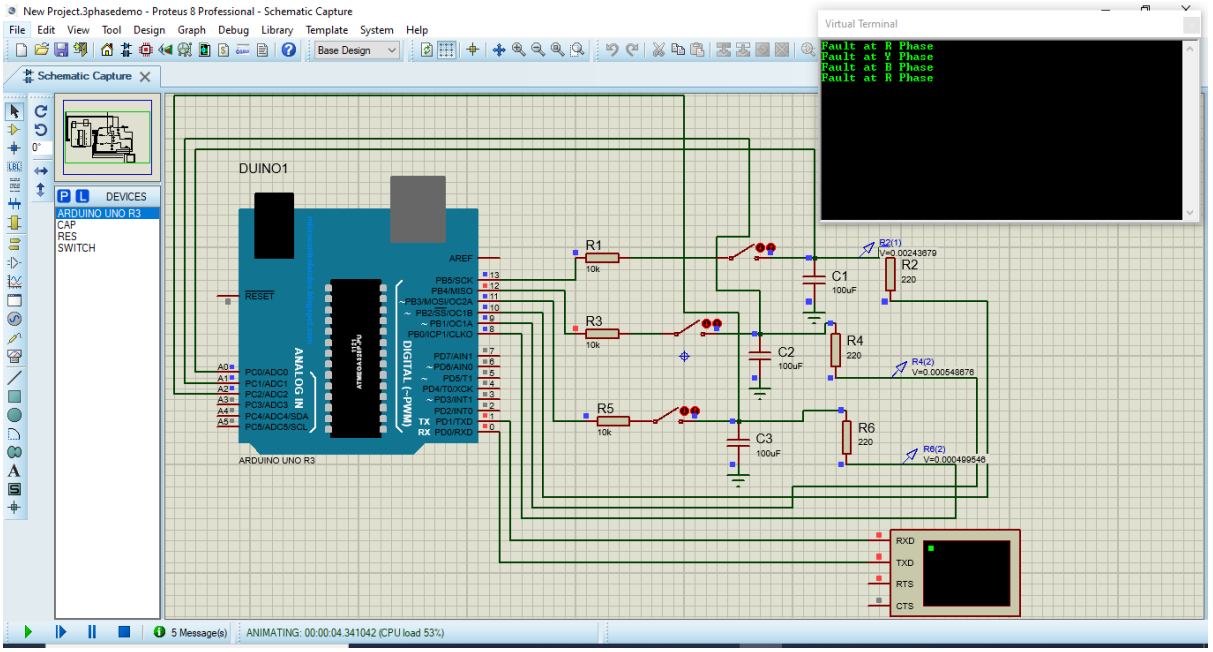


Figure 37: Open circuit fault at R, Y and B phase

Single-Phase 6 KM fault detection circuit with LCD Display:

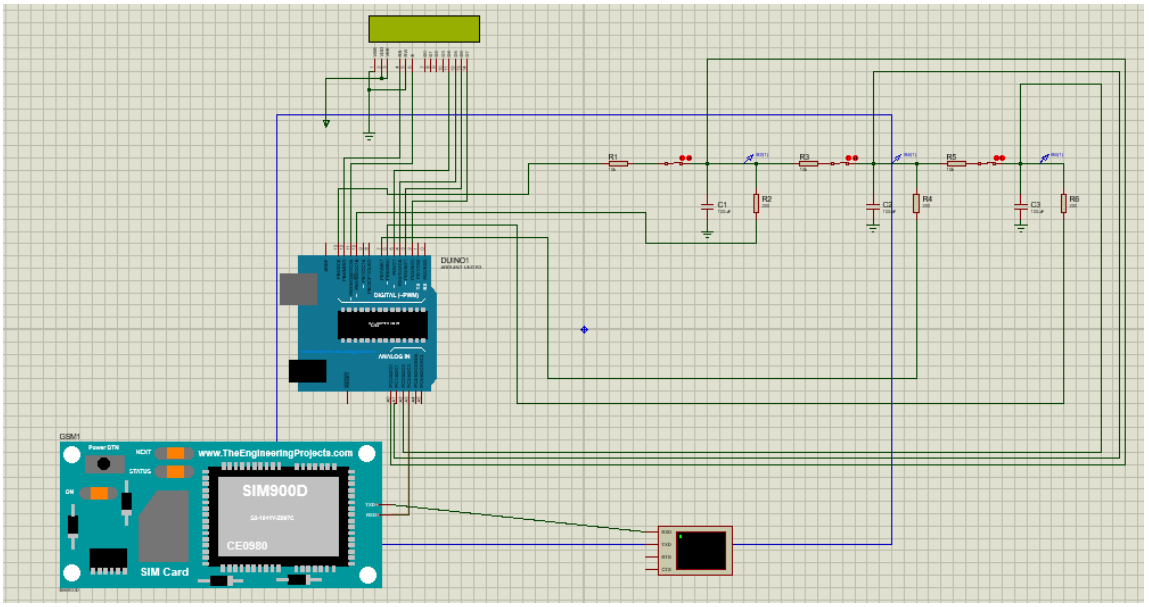


Figure 38: Single-phase 6 KM circuit at No-Fault condition

This circuit works on the same principle as a three-phase row scanning circuit which is the monetization of the time constant. The only difference is it shows the exact fault location instead of the defective phase by measuring the Time constant of each segment capacitor and checking if it exceeds the time constant or not. If TC is exceeded depending on the capacitor location it will show the fault location on the LCD.

.

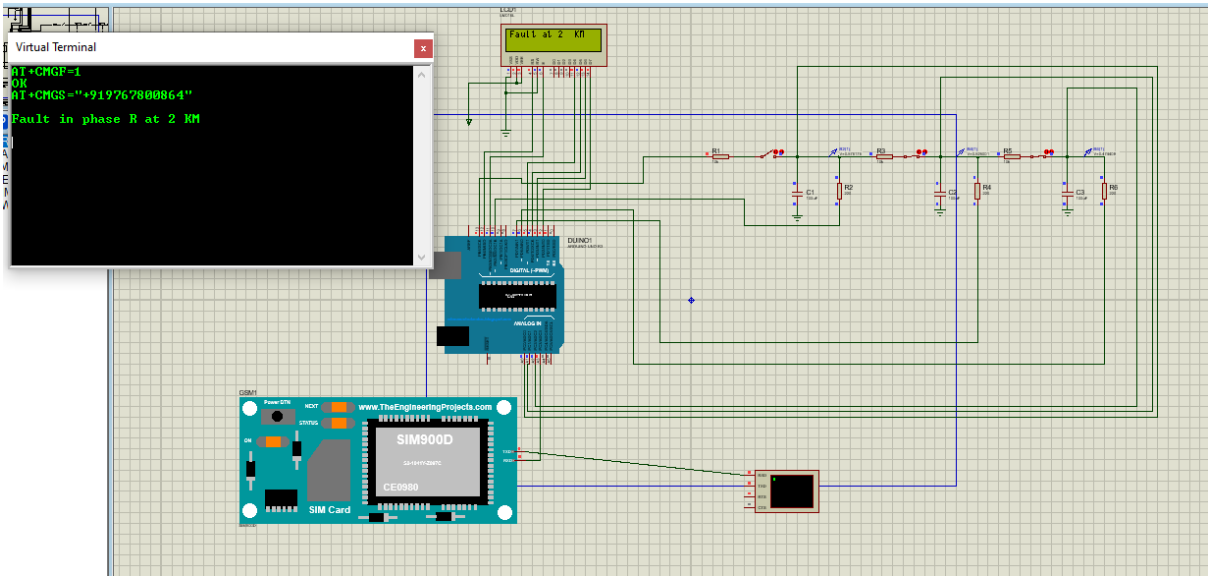


Figure 39 : Single-phase open circuit 2 KM fault

When switch-1(Left switch) is opened due to the opening of that portion of the circuit, the time constant exceeds its original characteristic time constant. So, the Arduino which has been pre-programmed detects this and shows the corresponding 2 KM fault is shown in the LCD and Sends a message through a GSM Module.

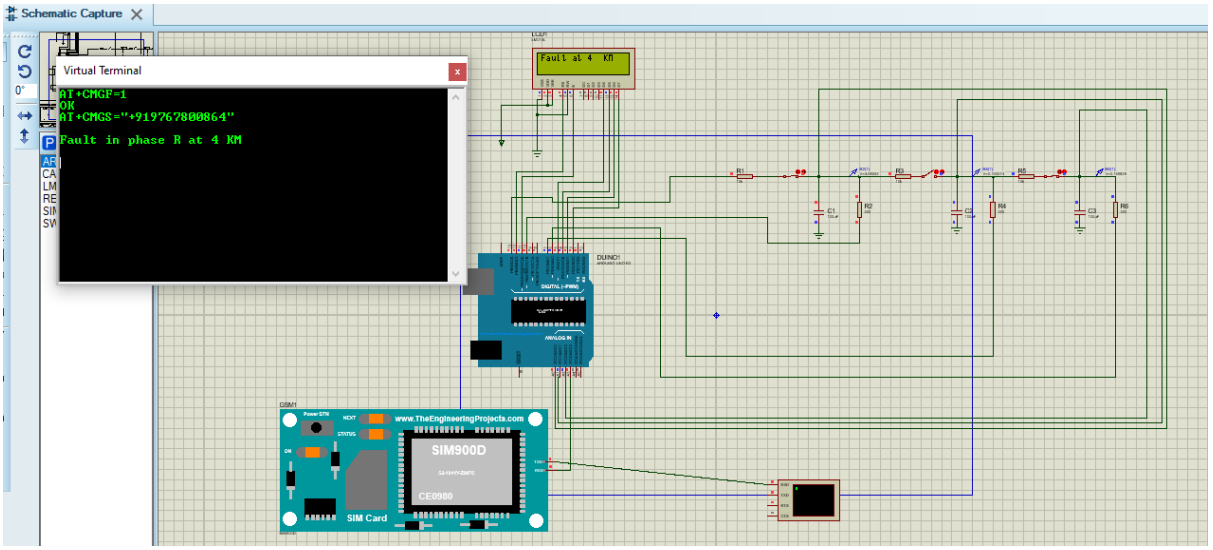


Figure 40: Single-phase open circuit 4 KM fault

When switch-2(Middle switch) is opened due to the opening of that portion of the circuit, the time constant exceeds its original characteristic time constant. So, the Arduino which has been pre-programmed detects this and shows the corresponding 4 KM fault is shown in the LCD and Sends a message through a GSM Module.

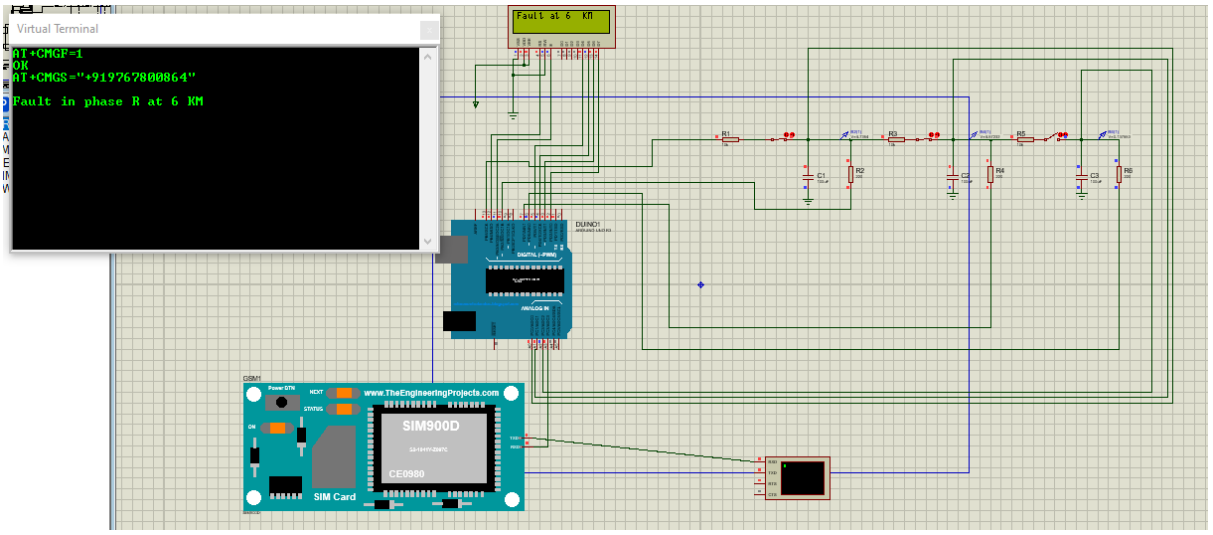


Figure 41: Single-phase open circuit 6 KM fault

When switch-3(Right switch) is opened due to the opening of that portion of the circuit, the time constant exceeds its original characteristic time constant. So, the Arduino which has been pre-programmed detects this and shows the corresponding 6 KM fault is shown in the LCD and Sends a message through a GSM Module.

Chapter 5

Conclusion

The main goal of our project was to develop an Arduino and GSM-based prototype to detect distances of the open and short circuit faults in the underground transmission cables, using a fairly convenient procedure. Our project proposes a possibility of isolating fault locations in the cable lines underground that is not visible to the bare eye otherwise. It will help to deal with hazardous situations like chemical damaging of underground cables due to chemical reaction with atmospheric reagents, or mechanical damaging while installing the cables and so on. By the end of our extensive research, we successfully proposed a design to carry out the fault location detection only in the case of 'short circuits', while keeping the 'open circuit' segment under consideration for future work. To detect the location of faulty underground cables in the case of short circuits using none other than the simple and very familiar concept of Ohm's Law and the row scanning technique in a three-phase system. Ohm's Law ensured that variations in voltage and current due to short circuits somewhere in the underground cable lines caused changes in the cable resistances. Our proposal included a microcontroller afterward to detect the corresponding variations, and displayed the fault in an LCD screen, sending a notification through GSM [14]. This proposal will undoubtedly help pave the way towards developing a nation with extensive installation and usage of underground cables as an alternative to overhead cables, thus allowing citizens to reap all the advantages of the former. As explained throughout the paper, it will do so by effectively aiding the process of detecting fault type and location using a much more convenient method than previously proposed methods like the Murray loop test [15], Varley loop test [16] and so on. Our project minimizes the limitations associated with these and many other past proposed methods, therefore encouraging increased use of underground cable lines instead of overhead cable lines.

5.1 Future Scope

In this project, we have estimated the location of short circuit faults and open circuit faults in the underground cable from the base station using Proteus simulation and Arduino. Although we have implemented the short circuit part in the hardware version as short circuit faults are more pernicious to the power line components, in the future, we are planning to detect earth faults as well and implement an open circuit in the hardware version.

Reference

[1] T. F. Express, “Overhead electric cables to go underground in five years,” *The Financial Express*. https://thefinancialexpress.com.bd/national/overhead-electric-cables-to-go- (accessed May 20, 2021).

[2] “Principles of Power Systems V.K Mehta.pdf.” https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxiZWUzYjJrMTF8Z3g6MjNiZDc2Y2E3YzRkZWFjOQ (accessed May 20, 2021).

[3] “IJREE010201.pdf.” Accessed: May 21, 2021. [Online]. Available: http://researchscript.com/wp-content/uploads/2014/12/IJREE010201.pdf

[4] N. Chourasia, K. Phule, R. Kamble, N. Marathe, and S. Fulkar, “Internet of Things (IOT) Based Underground Cable Fault Detector using ATmega Microcontroller,” vol. 4, no. 3, p. 6, 2019.

[5] R. K. R. Mansingh, R. Rajesh, S. Ramasubramani, and G. Ramkumar, “Underground Cable Fault Detection using Raspberry Pi and Arduino,” vol. 5, no. 4, p. 3, 2017.

[6] I. Samuel, A. Olajube, A. Awelewa, and U. Bassey, “ARDUINO MICROCONTROLLER BASED UNDERGROUND CABLE FAULT DISTANCE LOCATOR,” vol. 10, pp. 10–13, Mar. 2019.

[7] C. L. Wadhwa, *Electrical Power Systems*. New Age International, 2006.

[8] “Ohm’s law,” *Wikipedia*. Jan. 01, 2021. Accessed: May 20, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Ohm%27s_law&oldid=997572212

[9] “Design and fabrication of underground fault distance locator using arduino and GSM.” https://ieeexplore.ieee.org/abstract/document/8279001/ (accessed May 20, 2021).

[10] “ADC calculator-Analog to digital converter calculator,formula.” https://www.rfwireless-world.com/calculators/n-bit-ADC-calculator.html (accessed May 20, 2021).

[11] S. Desai, K. Walunjkar, V. Bhoyar, P. Maske, and S. Singh, “Underground Cable Fault Detection Using,” vol. 8, no. 10, p. 4.

[12] S. Kumar and K. Shradha, “OPEN CIRCUIT & SHORT CIRCUIT CABLE FAULT DETECTION USING ARDUINO,” vol. 6, no. 1, p. 8, 2018.

[13] “RC time constant,” *Wikipedia*. Dec. 07, 2020. Accessed: May 21, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=RC_time_constant&oldid=992825137

[14] “Estimation of Open & Short Circuit Fault Distances in the Underground Cable using Arduino & GSM Module.” https://1library.net/document/zxl1vwnz-estimation-short-circuit-fault-distances-underground-arduino-module.html (accessed May 21, 2021).

[15] “Murray loop test for locating underground cable fault | Electricalunits.com.” http://www.electricalunits.com/murray-loop-test-underground-cable-fault/?fbclid=IwAR1at2XXDfkv5TUmQo3G7vudiYvj2pn2JKDF4xUq_qaZ6Mmo6UtqCdTGcSA (accessed May 21, 2021).

[16] S. Chauhan, “Varly loop Test For finding Location Of Faults In Underground Cables,” *Electrical idea*, Jun. 04, 2018. http://www.electricalidea.com/varly-loop-test-for-finding-location-of-faults-in-underground-cables/ (accessed May 21, 2021).

Appendix

Short circuit Fault detection code:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(7, 8);
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
// define phase control pins
int phase[3] = {6, 9, 10};
//*************************************************************
int distance(int inputVoltage) {
if (inputVoltage >= 440 && inputVoltage < 500) {
return 8;
}
else if (inputVoltage >= 350 && inputVoltage < 420) {
return 6;
}
else if (inputVoltage >= 250 && inputVoltage < 330) {
return 4;
}
else if (inputVoltage >= 130 && inputVoltage < 190) {
return 2;
}
else return 0 ;

}
int red =0;
int yellow =0;
```

```
int blue =0;

//**************************
void setup()
{
  // put your setup code here, to run once:
  mySerial.begin(19200);     //Setting the baud rate of GSM module
  Serial.begin(19200);      // setting the baud rate of serial mopnitor (arduino)
  delay(100);
// set up the LCD's number of columns and rows:
lcd.begin(16, 2);
// set pin mode for phase relays
for (int j = 0; j < 3; j++) {
pinMode(phase[j], OUTPUT);
}

}
void loop()
{
digitalWrite(phase[0], HIGH);
delay(1000);
int dist1 = distance(analogRead(A0));
if (dist1 == 0) {
lcd.setCursor(0, 0);
lcd.write("R: ");
lcd.setCursor(3, 0);
lcd.write("NF ");
red = 0;
}
else {
lcd.setCursor(0, 0);
lcd.write("R: ");
lcd.setCursor(3, 0);
lcd.print(dist1);
lcd.setCursor(4, 0);
lcd.write(" KM");
red = dist1;
}
digitalWrite(phase[0], LOW);
```

```
//================================================
digitalWrite(phase[1], HIGH);
delay(1000);
int dist2 = distance(analogRead(A0));
if (dist2 == 0) {
lcd.setCursor(8, 0);
lcd.write("Y: ");
lcd.setCursor(11, 0);
lcd.write("NF ");
yellow = 0;
}
else {
lcd.setCursor(8, 0);
lcd.write("Y: ");
lcd.setCursor(11, 0);
lcd.print(dist2);
lcd.setCursor(12, 0);
lcd.write(" KM");
yellow = dist2;
}
digitalWrite(phase[1], LOW);
//================================================
digitalWrite(phase[2], HIGH);
delay(1000);
int dist3 = distance(analogRead(A0));
if (dist3 == 0) {
lcd.setCursor(0, 1);
lcd.write("B: ");
lcd.setCursor(3, 1);
lcd.write("NF ");
blue = 0;
}
else {
  lcd.setCursor(0, 1);
lcd.write("B: ");
lcd.setCursor(3, 1);
lcd.print(dist3);
lcd.setCursor(4, 1);
lcd.write(" KM");
```

```
blue = dist3;
}
digitalWrite(phase[2], LOW);
if (Serial.available()>0)
    Serial.read();
  {
    //red
    if(red==0){
      //RecieveMessage();
    }
else if(red==2)
{
  SendMessage1();
  }
  else if(red==4)
{
  SendMessage2();
  }
  else if(red==6)
{
  SendMessage3();
  }
else if(red==8)
{
  SendMessage4();
  }

  //yellow

  if(yellow==0){
      //RecieveMessage();
    }
else if(yellow==2)
{
  SendMessage5();
  }
  else if(yellow==4)
{
  SendMessage6();
```

```
  }
  else if(yellow==6)
{
  SendMessage7();
  }
else if(yellow==8)
{
  SendMessage8();
  }

  //blue

  if(blue==0){
      //RecieveMessage();
    }
else if(blue==2)
{
  SendMessage9();
  }
  else if(blue==4)
{
  SendMessage10();
  }
  else if(blue==6)
{
  SendMessage11();
  }
else if(blue==8)
{
  SendMessage12();
  }


  }

  if (mySerial.available()>0)
    Serial.write(mySerial.read());
}

void SendMessage1()
```

```
{
 mySerial.println("AT+CMGF=1");
 delay(100);
 mySerial.println("AT+CMGS=\"+8801711442192\"\r");
 delay(100);
 mySerial.println("Fault in phase R at 2 KM");
 delay(100);
 mySerial.println((char)26);
 delay(100);
}
void SendMessage2()
{
 mySerial.println("AT+CMGF=1");
 delay(100);
 mySerial.println("AT+CMGS=\"+8801711442192\"\r");
 delay(100);
 mySerial.println("Fault in phase R at 4 KM");
 delay(100);
 mySerial.println((char)26);
 delay(100);
}
void SendMessage3()
{
 mySerial.println("AT+CMGF=1");
 delay(100);
 mySerial.println("AT+CMGS=\"+8801711442192\"\r");
 delay(100);
 mySerial.println("Fault in phase R at 6 KM");
 delay(100);
 mySerial.println((char)26);
 delay(100);
}
void SendMessage4()
{
 mySerial.println("AT+CMGF=1");
 delay(100);
 mySerial.println("AT+CMGS=\"+8801711442192\"\r");
 delay(100);
 mySerial.println("Fault in phase R at 8 KM");
```

```
  delay(100);
  mySerial.println((char)26);
  delay(100);
}
void SendMessage5()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+8801711442192\"\r");
  delay(100);
  mySerial.println("Fault in phase Y at 2 KM");
  delay(100);
  mySerial.println((char)26);
  delay(100);
}
void SendMessage6()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"8801711442192\"\r");
  delay(100);
  mySerial.println("Fault in phase Y at 4 KM");
  delay(100);
  mySerial.println((char)26);
  delay(100);
}
void SendMessage7()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+8801711442192\"\r");
  delay(100);
  mySerial.println("Fault in phase Y at 6 KM");
  delay(100);
  mySerial.println((char)26);
  delay(100);
}
void SendMessage8()
{
```

```
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+8801711442192\"\r");
  delay(100);
  mySerial.println("Fault in phase Y at 8 KM");
  delay(100);
  mySerial.println((char)26);
  delay(100);
}
void SendMessage9()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+8801711442192\"\r");
  delay(100);
  mySerial.println("Fault in phase B at 2 KM");
  delay(100);
  mySerial.println((char)26);
  delay(100);
}
void SendMessage10()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+8801711442192\"\r");
  delay(100);
  mySerial.println("Fault in phase B at 4 KM");
  delay(100);
  mySerial.println((char)26);
  delay(100);
}
void SendMessage11()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+8801711442192\"\r");
  delay(100);
  mySerial.println("Fault in phase B at 6 KM");
  delay(100);
```

```
  mySerial.println((char)26);
  delay(100);
}
void SendMessage12()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+8801711442192\"\r");
  delay(100);
  mySerial.println("Fault in phase B at 8 KM");
  delay(100);
  mySerial.println((char)26);
  delay(100);
}

void RecieveMessage()
{
  mySerial.println("AT+CNMI=2,2,0,0,0");
  delay(100);
}
```

Open Circuit fault detection(Three phase):

```
#define analogPin0      0
#define analogPin1      1
#define analogPin2      2
#define chargePin1      13
#define chargePin2      12
#define chargePin3      11
#define dischargePin1    10
#define dischargePin2     9
#define dischargePin3     8
int phase[3] = {13, 12, 11};




unsigned long startTime0;
unsigned long elapsedTime0;
unsigned long startTime1;
```

```
unsigned long elapsedTime1;
unsigned long startTime2;
unsigned long elapsedTime2;


void setup(){
  for (int j = 0; j < 3; j++) {
pinMode(phase[j], OUTPUT);
}

  pinMode(chargePin1, OUTPUT);
  digitalWrite(chargePin1, LOW);

  pinMode(chargePin2, OUTPUT);
  digitalWrite(chargePin2, LOW);

  pinMode(chargePin3, OUTPUT);
  digitalWrite(chargePin3, LOW);

  Serial.begin(9600);
}

void loop()
{
digitalWrite(phase[0], HIGH);
//delay(1000);
digitalWrite(chargePin1, HIGH);
  startTime0 = millis();
   while(analogRead(analogPin0) < 648){
elapsedTime0= millis() - startTime0;
//Serial.println(elapsedTime0);


if(elapsedTime0>=1005)
{
  Serial.println("Fault at R Phase");
  break;
  }
```

```
  }



  digitalWrite(chargePin1, LOW);

  pinMode(dischargePin1, OUTPUT);

  digitalWrite(dischargePin1, LOW);

  while(analogRead(analogPin0) > 0){
  }

  pinMode(dischargePin1, INPUT);

  digitalWrite(phase[0], LOW);
  digitalWrite(phase[1], HIGH);
  digitalWrite(chargePin2, HIGH);
  startTime1 = millis();
   while(analogRead(analogPin1) < 648)
   {
elapsedTime1= millis() - startTime1;

 if(elapsedTime1>=1005)
{
  Serial.println("Fault at Y Phase");
  break;
  }
//Serial.println(elapsedTime1);
   }
digitalWrite(chargePin2, LOW);

  pinMode(dischargePin2, OUTPUT);

  digitalWrite(dischargePin2, LOW);

  while(analogRead(analogPin1) > 0){
```

```
  }

  pinMode(dischargePin2, INPUT);


  digitalWrite(phase[1], LOW);
  digitalWrite(phase[2], HIGH);
  digitalWrite(chargePin3, HIGH);
  startTime2 = millis();
   while(analogRead(analogPin2) < 648)
   {
elapsedTime2= millis() - startTime2;
if(elapsedTime2>=1005)
{
  Serial.println("Fault at B Phase");
  break;
  }
//Serial.println(elapsedTime2);
   }
digitalWrite(chargePin3, LOW);

  pinMode(dischargePin3, OUTPUT);

  digitalWrite(dischargePin3, LOW);

  while(analogRead(analogPin2) > 0){
  }

  pinMode(dischargePin3, INPUT);

  digitalWrite(phase[2], LOW);

}
```

Open Circuit fault detection(Single phase 6 KM):

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(11, A3);
#define analogPin0      0
#define analogPin1      1
#define analogPin2      2
#define chargePin      13
#define dischargePin   10
#define dischargePin    7
#define dischargePin    6
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

                    // F formatter tells compliler it's a floating point value
```

```
unsigned long startTime0;
unsigned long elapsedTime0;
unsigned long startTime1;
unsigned long elapsedTime1;
unsigned long startTime2;
unsigned long elapsedTime2;
int red =0;



void setup(){
  lcd.begin(16, 2);

  pinMode(chargePin, OUTPUT);     // set chargePin to output

  digitalWrite(chargePin, LOW);
mySerial.begin(9600);
  Serial.begin(9600);

}

void loop(){

  digitalWrite(chargePin, HIGH);  // set chargePin HIGH and capacitor charging

  startTime0 = millis();
  startTime1 = millis();
  startTime2 = millis();

  while(analogRead(analogPin0) < 648)
  {

  elapsedTime0= millis() - startTime0;
  //Serial.println(elapsedTime0);
if(elapsedTime0>=2200)
{
  Serial.println("Fault at 2 KM");
  lcd.setCursor(0, 0);
```

```
lcd.write("Fault at ");
lcd.setCursor(9, 0);
lcd.print("2 ");
lcd.setCursor(11, 0);
lcd.write(" KM");
SendMessage0();
  return;
  }
  }
  while(analogRead(analogPin1) < 648)
  {

  elapsedTime1= millis() - startTime1;
  //Serial.println(elapsedTime1);
  if(elapsedTime1>=5000)
{
  Serial.println("Fault at 4 KM");
  lcd.setCursor(0, 0);
lcd.write("Fault at ");
lcd.setCursor(9, 0);
lcd.print("4 ");
lcd.setCursor(11, 0);
lcd.write(" KM");
SendMessage1();
  return;
  }

  }
  while(analogRead(analogPin2) < 648)
  {

  elapsedTime2= millis() - startTime2;
  //Serial.println(elapsedTime2);
  if(elapsedTime2>=6100)
{
  Serial.println("Fault at 6 KM");
  lcd.setCursor(0, 0);
lcd.write("Fault at ");
lcd.setCursor(9, 0);
```

```
lcd.print("6 ");
lcd.setCursor(11, 0);
lcd.write(" KM");
SendMessage2();
 return;
 }

 }



 digitalWrite(chargePin, LOW);             // set charge pin to  LOW

 pinMode(dischargePin, OUTPUT);            // set discharge pin to output

 digitalWrite(dischargePin, LOW);          // set discharge pin LOW

 while(analogRead(analogPin0) > 0){        // wait until capacitor is completely discharged

 }

 pinMode(dischargePin, INPUT);

if (Serial.available()>0)
   Serial.read();
 {
  //red
  if(red==0){
    //RecieveMessage();
  }
  else if(red==2)
{
 SendMessage0();
 }
 else if(red==4)
{
 SendMessage1();
 }
 else if(red==6)
```

```
{
  SendMessage2();
  }
  }
  if (mySerial.available()>0)
    Serial.write(mySerial.read());
}

void SendMessage0()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+919767800864\"\r");
  delay(100);
  mySerial.println("Fault in phase R at 2 KM");
  delay(100);
  mySerial.println((char)26);
  delay(100);
}
void SendMessage1()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+919767800864\"\r");
  delay(100);
  mySerial.println("Fault in phase R at 4 KM");
  delay(100);
  mySerial.println((char)26);
  delay(100);
}
void SendMessage2()
{
  mySerial.println("AT+CMGF=1");
  delay(100);
  mySerial.println("AT+CMGS=\"+919767800864\"\r");
  delay(100);
  mySerial.println("Fault in phase R at 6 KM");
  delay(100);
  mySerial.println((char)26);
```

```
  delay(100);
}
```