Indian Institute of Technology Jodhpur

॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

EEL3060: Project Report

"Phasor estimation and display"

Group Members:

Anupama Birman (B21EE008)

Jyoti Dhayal (B21EE029)

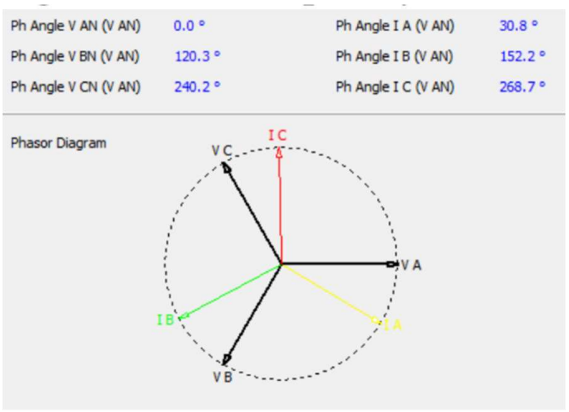
Aman Tripathi(B21EE005)

Akshit Singh (B21EE004)

Karanam Preethi(B21EE032)

Problem Statement: Module for Phasor Estimation and Display

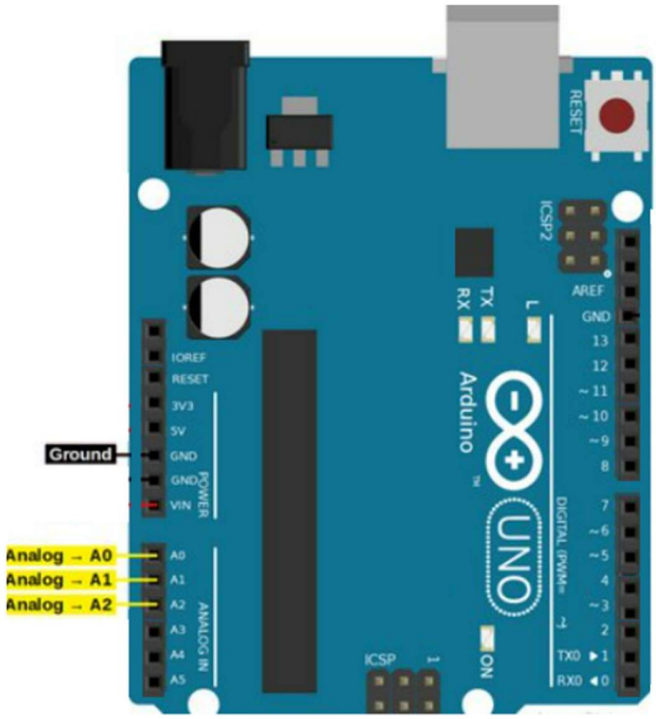
The project requires the implementation of an algorithm for phasor estimation from instantaneous data and the creation of a polar plot display of phasors. The algorithm should be implemented on either a Raspberry Pi or an Arduino Uno platform. The provided Arduino code handles the phasor estimation, while the Python script manages the display of phasors using Matplotlib.



Arduino Implementation:

The Arduino code implements the algorithm for phasor estimation. It samples analog data from three different phases, computes the real and imaginary components of each phasor using Discrete Fourier Transform (DFT), and then calculates the magnitude and phase angle of each phasor. The calculated values are then outputted via serial communication.

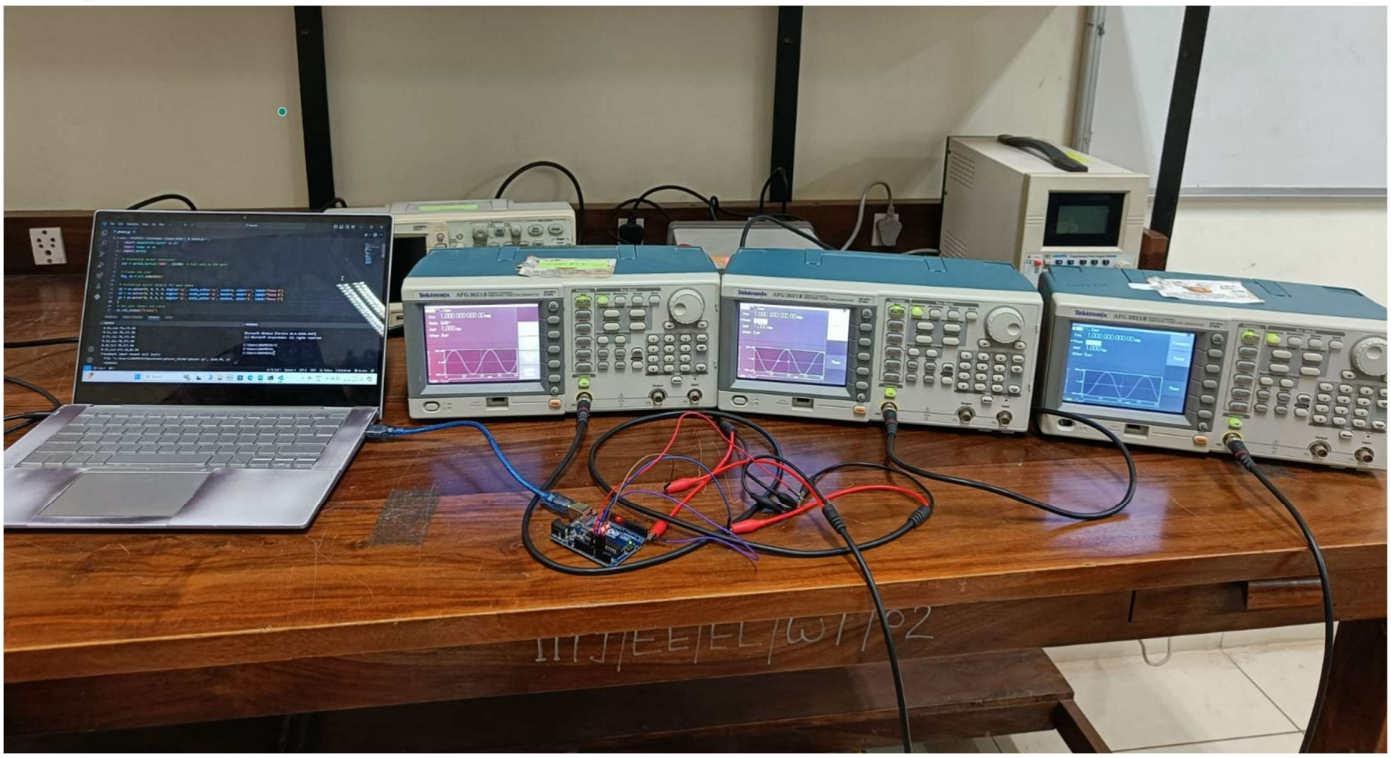
Arduino Pin Connections Used in Project

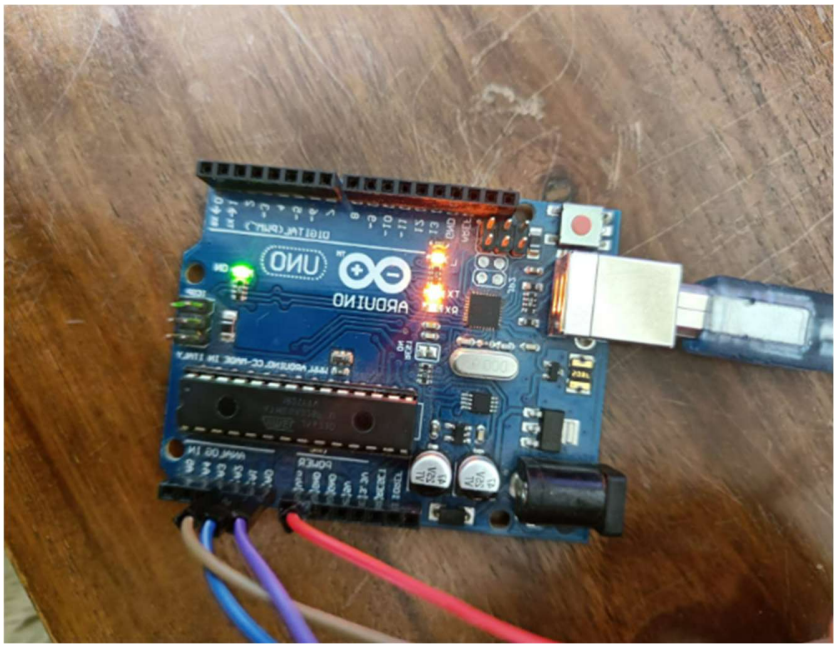


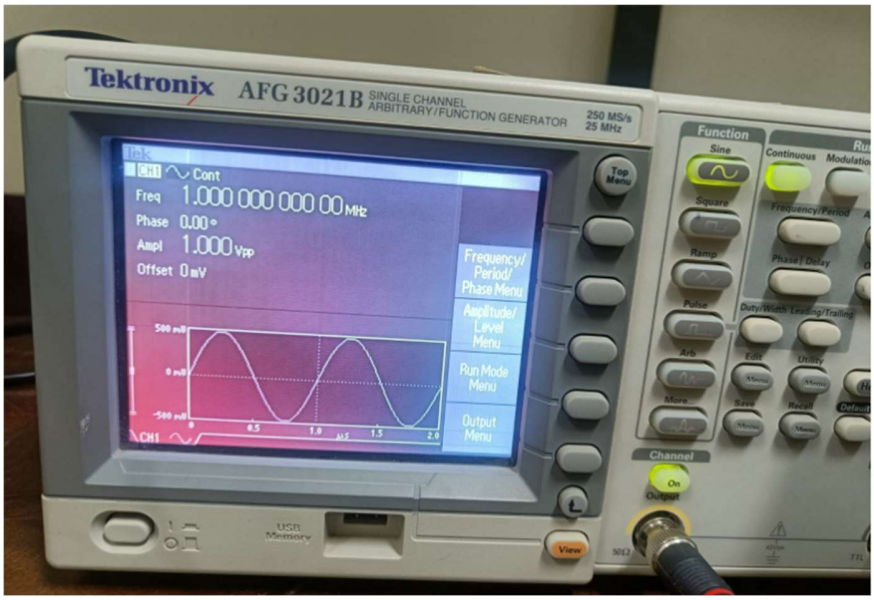
Python Script for Display:

The Python script receives the phasor data over serial communication from the Arduino board. It then parses the data, calculates the x and y components of each phasor, and updates the polar plot using Matplotlib. The plot dynamically displays the phasors' magnitudes and angles, providing a visual representation of the electrical system's state.

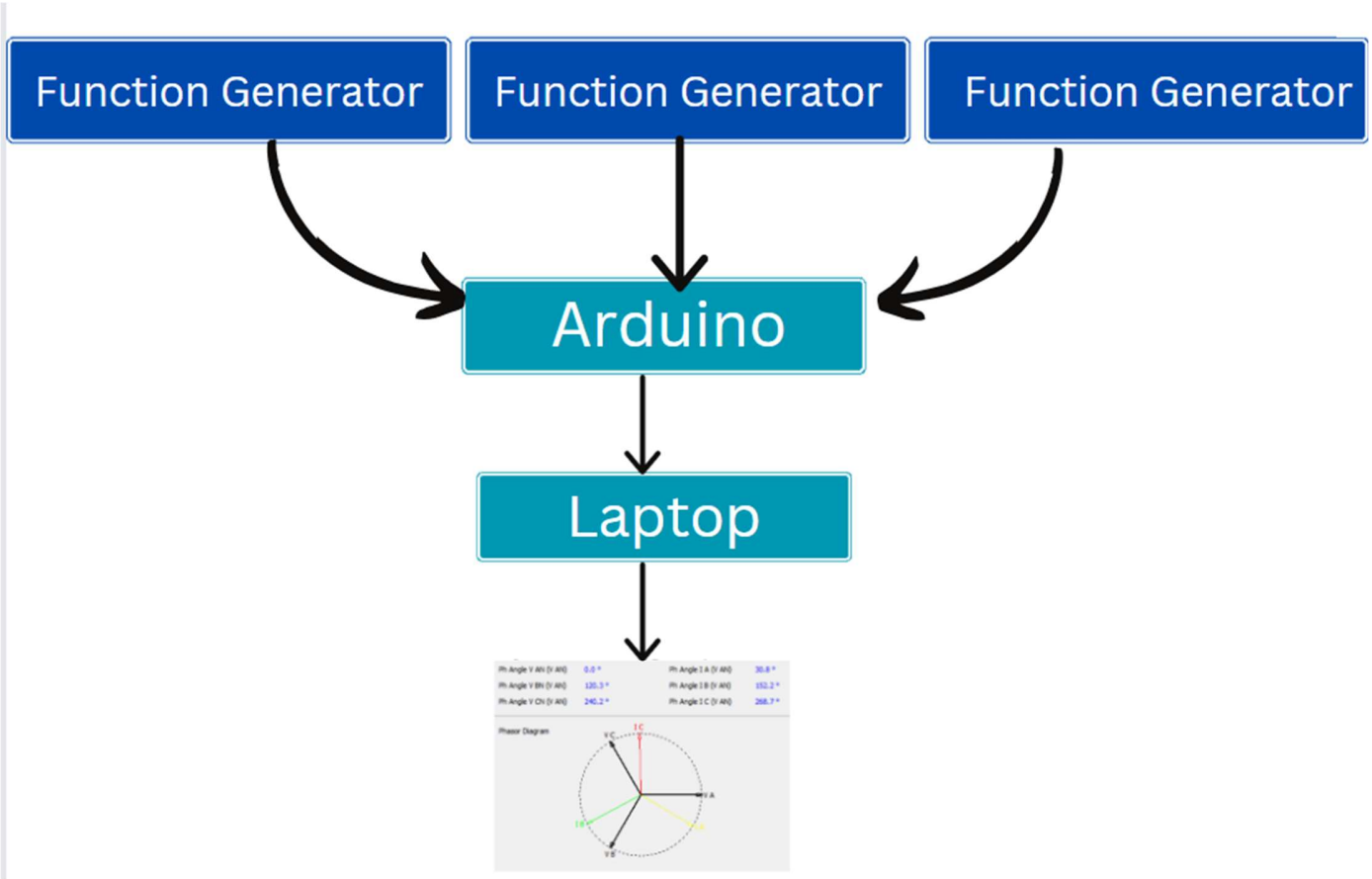
Setup:







Flow Diagram for Phasor Estimation and Display



Implementation Details:

Arduino Code:

- Utilizes analog Read to sample data from three different phases.
- Implements DFT to compute the real and imaginary components of each phasor.
- Calculates magnitude and phase angle for each phasor.
- Outputs phasor data via serial communication.

Python Script:

- Establishes a serial connection with the Arduino board.
- Receives phasor data and parses it into magnitude and phase angle components.
- Converts polar coordinates to Cartesian coordinates for plotting.
- Updates the polar plot in real-time to visualize phasors' states.

Code Explanation:

Arduino Code explanation

Variable Declarations:

- **WindowSize:** Defines the size of the window for data sampling.
- **N:** Sets the number of samples to WindowSize.
- **pi:** Defines the value of pi for later calculations.
- **Arrays adc_out_1, adc_out_2, and adc_out_3:** Store analog data samples for each phase.
- Variables for real (Xr) and imaginary (Xi) components of each phase's phasor, phasor magnitudes (Phasor_Magnitude), and phasor angles (Phasor_Angle).
- **Constants analogPin_1, analogPin_2, and analogPin_3:**
- Define the analog pins used for sampling data from each phase.

Setup Function (void setup()):

- Initializes serial communication with a baud rate of 115200.
- Sets the analog reference to DEFAULT (5V).

Loop Function (void loop()):

- Calls calc_phasor() to calculate phasors.
- Calls plot_phasors() to output phasor angles.
- Delays for 1 second before the next iteration.

Phasor Calculation Function (void calc_phasor()):

- Samples analog data from each phase and stores it in the corresponding arrays.
- Initializes real and imaginary components to zero.
- Uses a loop to compute the real and imaginary components for each phase using Discrete Fourier Transform (DFT) equations.
- Calculates phasor magnitudes and angles using the computed real and imaginary components.

Phasor Plotting Function (void plot_phasors()):

- Loops through each sample in the window.
- Calculates samples for each phase using the phasor magnitudes and angles.
- Outputs the phasor angles of each phase via serial communication

Python script Code explanation

The Python script creates a phasor diagram using Matplotlib to visualize data received from the serial connection with Arduino.

Imports:

- **matplotlib.pyplot as plt**: Importing the Matplotlib library's pyplot module and aliasing it as plt.
- **numpy as np**: Importing the NumPy library and aliasing it as np.
- **serial**: Importing the serial module for serial communication.

Serial Connection Initialization:

- It initializes a serial connection with a device connected to the COM3 port at a baud rate of 115200.

Plot Initialization:

- It creates a plot using Matplotlib's subplots() function, returning a figure (fig) and an axis (ax) object.

Quiver Initialization:

- Quiver objects (q1, q2, q3) are initialized for each phase (Phase A, Phase B, Phase C) on the plot.

Setting Plot Labels and Title:

- X and Y labels and a title are set for the plot.
- Grid lines are enabled.
- Axes are set to be equal and square.
- Legends for each phase are enabled.

Data Initialization:

Empty lists (data1, data2, data3) are initialized to store data for each phase.

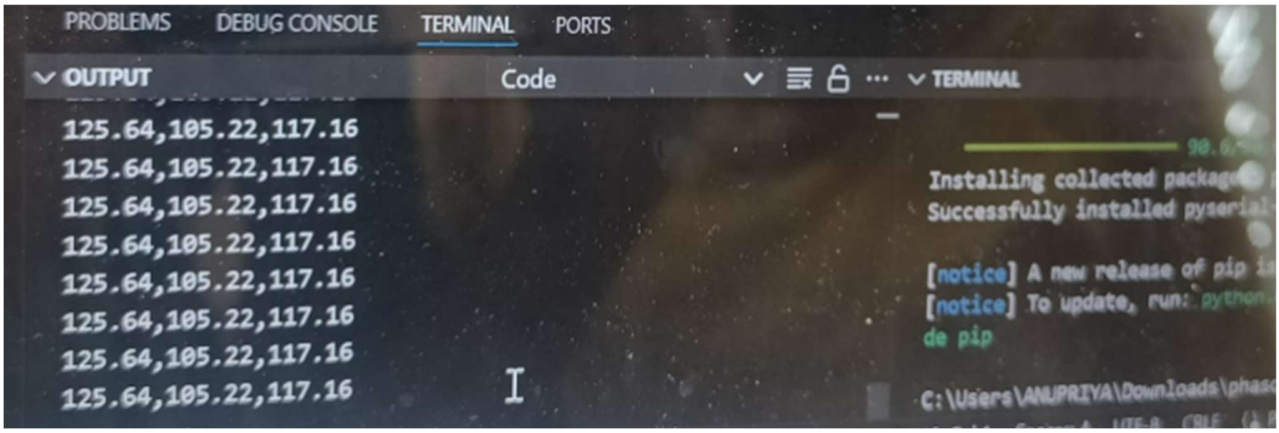
Update Plot Function:

- update_plot() function updates the quiver plot with new data.
- It takes the quiver object (q), the data list (data), and the angle in radians (angle_rad) as inputs.
- It sets the magnitude and direction of the quiver arrows based on the latest data.
- It annotates the plot with the current angle value.
- It then updates the plot.

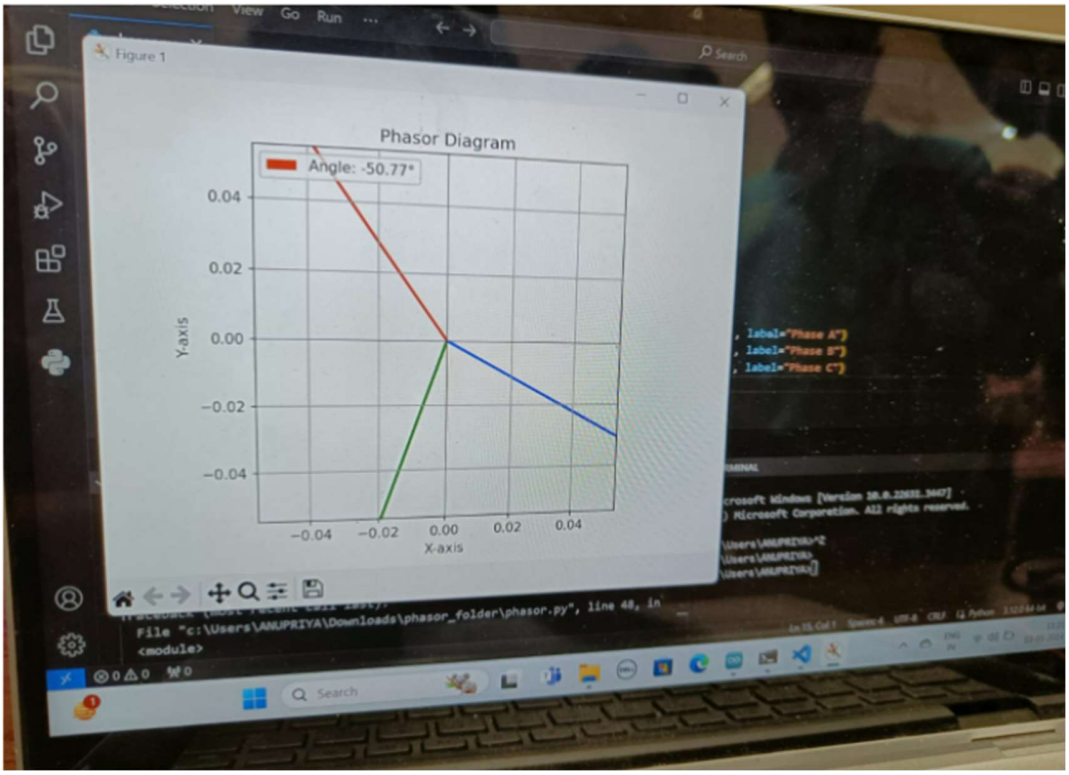
Reading and Processing Data:

- A while loop continuously reads data from the serial connection.
- Each line of data received is decoded from bytes to a string and stripped of whitespace.

- The string is split into parts using commas as delimiters.
- If the length of parts is 3, it indicates that valid data for all three phases has been received.
- The angles in radians for each phase are extracted from parts.



- The x and y components of each phase are calculated using trigonometric functions (np.cos() and np.sin()).
- The x and y components are appended to the respective data lists (data1, data2, data3).
- The plot is updated for each phase using the update_plot() function.



Thank You