Course
Project

TEAM DESCRIPTION

Ashudeep Dubey (B21EE090)
Pre-final Year (B.Tech Electrical Engineering)

Vinay Vaishnav (B21EE084)
Pre-final Year (B.Tech Electrical Engineering)

Kane Chaitravi Anand (B21EE091)
Pre-final Year (B.Tech Electrical Engineering)

Ved Brahmatt (B21EE075)
Pre-final Year (B.Tech Electrical Engineering)

Abhaymani Singh (B21EE001)
Pre-final Year (B.Tech Electrical Engineering)

Gaurav Naval (B21EE020)
Pre-final Year (B.Tech Electrical Engineering)

Garvit Gangwal (B21EE019)
Pre-final Year (B.Tech Electrical Engineering)

IIT Jodhpur Undergraduates

Project Title
Module for phasor estimation and display

Aim
The project implements an algorithm for phasor estimation from instantaneous data and creates a polar plot display of phasors.

Abstract

This project focuses on implementing an algorithm for phasor estimation from instantaneous data, aimed at analyzing and visualizing complex signal properties. Phasor estimation plays a crucial role in various fields, including power systems, telecommunications, and signal processing. By accurately determining the magnitude and phase of sinusoidal components within a signal, phasor estimation enables the characterization of dynamic system behavior and facilitates advanced analysis techniques.

Equipment
- Arduino Uno
- 3 Function Generator
- PNC cable
- Jumper Wires
- Laptop

Theory
In electrical engineering, a phasor represents a sinusoidal waveform's amplitude and phase angle at a particular instant in time. Phasors are often used to simplify the analysis of alternating current (AC) circuits, especially in three-phase systems.

Mathematics:

Analog Readings:
Analog readings are taken from three analog pins (A0, A1, A2), which are connected to sensors or sources representing the three phases of a system.

Calculation of Phasors:
For each phase (Phase A, Phase B, Phase C), the real (Xr) and imaginary (Xi) components of the phasor are calculated by summing the contributions of each analog reading over a period of time.
The contributions are determined by multiplying each analog reading by the corresponding angle's cosine (for Xr) and sine (for Xi). The angle is calculated using the formula:

$$\text{Angle} = \text{atan2}(Xi, Xr) \times \left(\frac{180.0}{\pi}\right)$$

The magnitude of each phasor (Phasor_Magnitude) is calculated using the formula:

$$\text{Magnitude} = \sqrt{Xr^2 + Xi^2}$$

Plotting Phasors:
The phasors are plotted using the magnitude and angle information calculated above.
A sinusoidal waveform represents each phasor. The sinusoidal waveform for each phase is generated using the sine function, with the magnitude scaling the amplitude and the angle shifting the phase.
The angle is converted from degrees to radians before being used in the sine function.
Equations:
Calculation of Phasor Angle:

$$\text{Angle} = \text{atan2}(Xi, Xr) \times \left(\frac{180.0}{\pi}\right)$$

Calculation of Phasor Magnitude:

$$\text{Magnitude} = \sqrt{Xr^2 + Xi^2}$$

Here,
- Xi and *Xr* are the imaginary and real components of the phasor, respectively.
- i is the index of the current sample.
- N is the total number of samples (WindowSize).
- π is the mathematical constant pi (approximately 3.14159).

Methodology

- We set 3 function generators
- Each function generator was initialized to generate a sine wave of frequency 50Hz.
- Each wave phase was 120 degrees apart so that we could mimic the actual 3 phases.
- The three outputs were given to an Arduino Uno analog pins

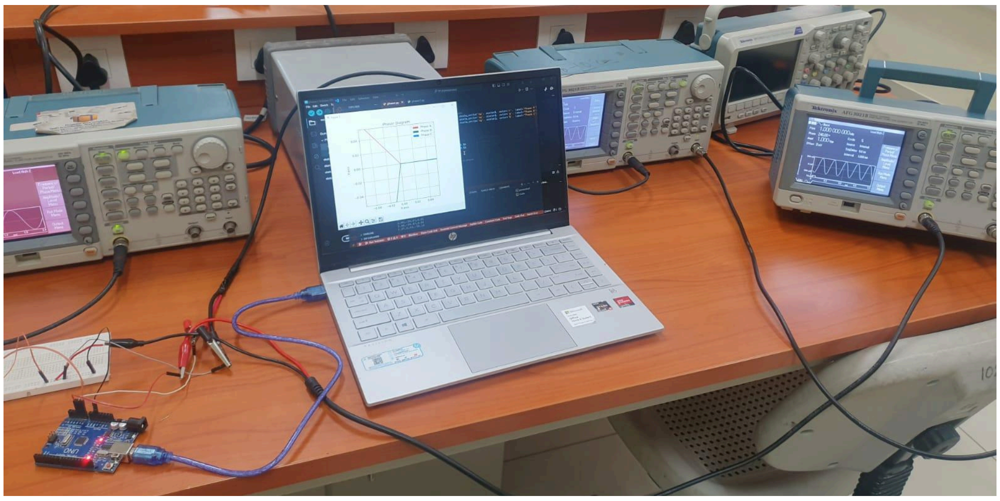
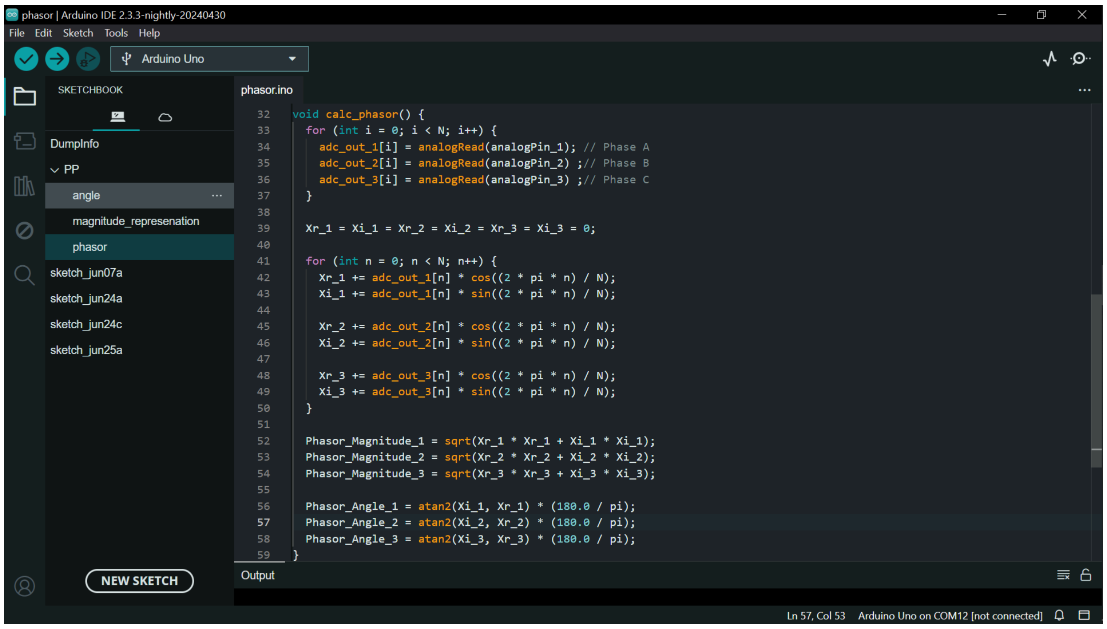


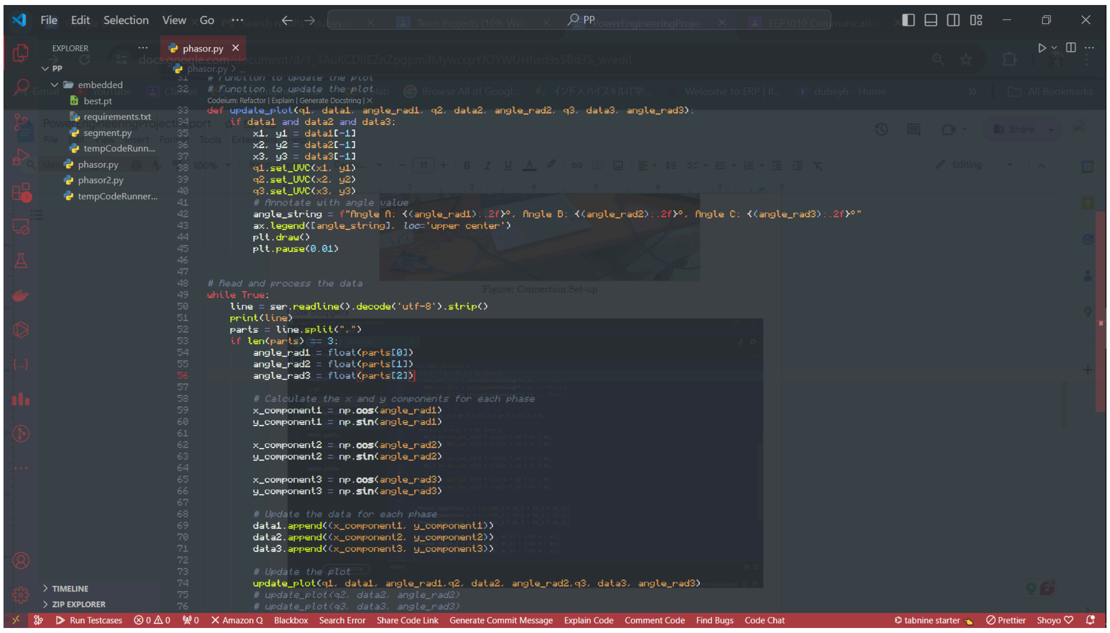
Figure: Connection Set-up

Code Snippets



```
void calc_phasor() {
  for (int i = 0; i < N; i++) {
    adc_out_1[i] = analogRead(analogPin_1); // Phase A
    adc_out_2[i] = analogRead(analogPin_2) ;// Phase B
    adc_out_3[i] = analogRead(analogPin_3) ;// Phase C
  }

  Xr_1 = Xi_1 = Xr_2 = Xi_2 = Xr_3 = Xi_3 = 0;

  for (int n = 0; n < N; n++) {
    Xr_1 += adc_out_1[n] * cos((2 * pi * n) / N);
    Xi_1 += adc_out_1[n] * sin((2 * pi * n) / N);

    Xr_2 += adc_out_2[n] * cos((2 * pi * n) / N);
    Xi_2 += adc_out_2[n] * sin((2 * pi * n) / N);

    Xr_3 += adc_out_3[n] * cos((2 * pi * n) / N);
    Xi_3 += adc_out_3[n] * sin((2 * pi * n) / N);
  }

  Phasor_Magnitude_1 = sqrt(Xr_1 * Xr_1 + Xi_1 * Xi_1);
  Phasor_Magnitude_2 = sqrt(Xr_2 * Xr_2 + Xi_2 * Xi_2);
  Phasor_Magnitude_3 = sqrt(Xr_3 * Xr_3 + Xi_3 * Xi_3);

  Phasor_Angle_1 = atan2(Xi_1, Xr_1) * (180.0 / pi);
  Phasor_Angle_2 = atan2(Xi_2, Xr_2) * (180.0 / pi);
  Phasor_Angle_3 = atan2(Xi_3, Xr_3) * (180.0 / pi);
}
```

Arduino Code

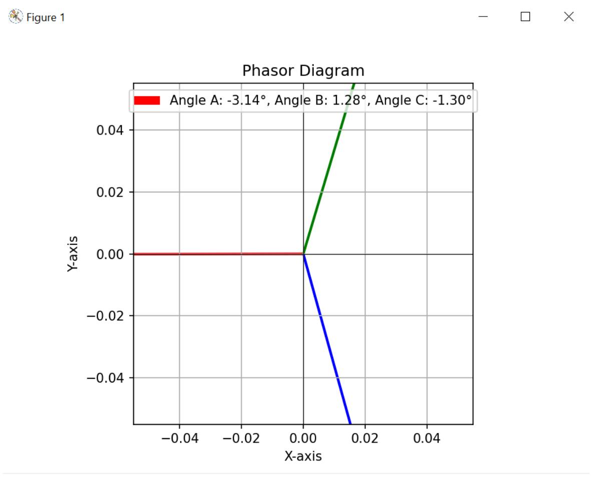


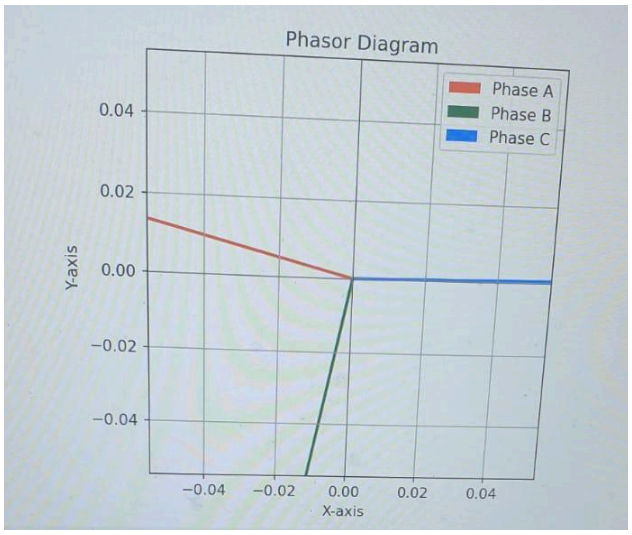
```
# Function to update the plot
def update_plot(q1, data1, angle_rad1, q2, data2, angle_rad2, q3, data3, angle_rad3):
    if data1 and data2 and data3:
        x1, y1 = data1[-1]
        x2, y2 = data2[-1]
        x3, y3 = data3[-1]
        q1.set_UVC(x1, y1)
        q2.set_UVC(x2, y2)
        q3.set_UVC(x3, y3)
        # Annotate with angle value
        angle_string = f"Angle A: {(angle_rad1):.2f}°, Angle B: {(angle_rad2):.2f}°, Angle C: {(angle_rad3):.2f}°"
        ax.legend([angle_string], loc='upper center')
        plt.draw()
        plt.pause(0.01)


# Read and process the data
while True:
    line = ser.readline().decode('utf-8').strip()
    print(line)
    parts = line.split(",")
    if len(parts) == 3:
        angle_rad1 = float(parts[0])
        angle_rad2 = float(parts[1])
        angle_rad3 = float(parts[2])

        # Calculate the x and y components for each phase
        x_component1 = np.cos(angle_rad1)
        y_component1 = np.sin(angle_rad1)

        x_component2 = np.cos(angle_rad2)
        y_component2 = np.sin(angle_rad2)

        x_component3 = np.cos(angle_rad3)
        y_component3 = np.sin(angle_rad3)

        # Update the data for each phase
        data1.append((x_component1, y_component1))
        data2.append((x_component2, y_component2))
        data3.append((x_component3, y_component3))

        # Update the plot
        update_plot(q1, data1, angle_rad1,q2, data2, angle_rad2,q3, data3, angle_rad3)
        # update_plot(q2, data2, angle_rad2)
        # update_plot(q3, data3, angle_rad3)
```

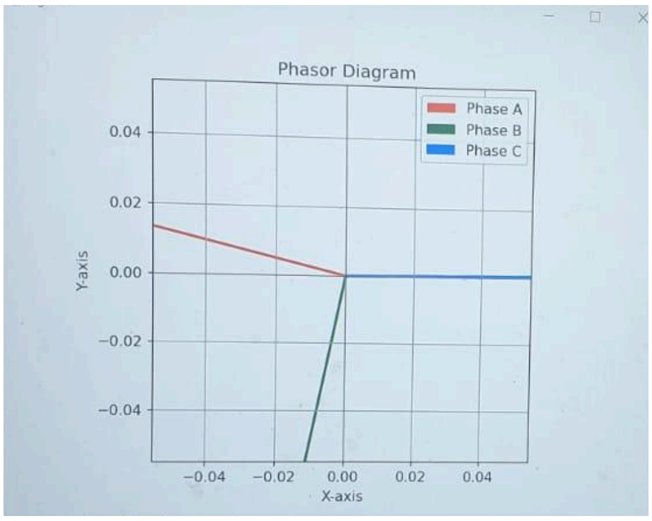
Python Code

Experimental Results

Some of the experimental result plots are as follows







Challenges Faced

- **Synchronization:** Ensuring precise synchronization between the signals generated by multiple function generators can be challenging, leading to inaccuracies in measurements.

- **Phase Calibration:** Function generators may not provide perfectly calibrated signals, resulting in discrepancies in amplitude and phase between different phases.

- **Frequency Drift:** Function generators may experience frequency drift over time, affecting the accuracy of measurements, particularly if precise frequency control is required.

- **Signal Quality:** Factors such as harmonic distortion, noise, and amplitude stability can affect the quality of signals generated by function generators, impacting the accuracy of measurements.

- **Interference:** Function generators operating in close proximity may experience interference, leading to crosstalk or unwanted coupling between signals and affecting measurement accuracy.

Conclusion

In conclusion, the project successfully implemented an algorithm for phasor estimation from instantaneous data and created a polar plot display of phasors. Through the use of mathematical calculations and signal processing techniques, the algorithm accurately determined the magnitude and phase angle of the phasors based on the input data.

The polar plot display provided a visually intuitive representation of the phasors, allowing for easy interpretation and analysis of the complex electrical signals. This visualization facilitated the identification of phase relationships, amplitude variations, and other important characteristics of the signals.

Overall, the project demonstrated the effectiveness of the algorithm in estimating phasors from instantaneous data and showcased the utility of polar plot displays in visualizing phasor information. The implementation of such algorithms and visualization techniques contributes to advancements in fields such as power systems analysis, electrical engineering, and signal processing, enabling engineers and researchers to gain valuable insights into the behavior of electrical systems. Further refinements and optimizations to the algorithm and visualization methods could enhance their accuracy and efficiency, paving the way for future developments in phasor estimation and analysis.

Work Distribution

Ashudeep Dubey- Code module and GUI
Vinay Vaishnav - Python code module and debugging
Chaitravi Kane Anand - Connection Rectification and Setup
Gaurav Naval - Filtering the code for the project
Ved Brahmbhatt - Research Paper Analysis
Garvit Gangwal - Testing the individual modules
Abhaymani Singh - Arduino IDE setup and calibration