Ground Console and Analysis for ANALOG MINI MAGNETOMETER

Submitted by,

RAGHUVEER V - 20211COM0030

Under the guidance of,

Mr. Mohamed Shakir

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER ENGINEERING

At

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Internship report **“Ground Console and Analysis for ANALOG MINI MAGNETOMETER”** being submitted by “RAGHUVEER V” bearing roll number “20211COM0030” in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Engineering is a bonafide work carried out under my supervision.

Mr. Mohamed Shakir
Assistant Professor - PSCS
Presidency University

Dr. Gopal Krishna Shyam
Professor & HoD-PSCS
Presidency University

Dr. Mydhili K Nair
Associate Dean-PSCS
Presidency University

Dr. Sameeruddin Khan
Pro Vice Chancellor
Engineering
Dean –PSCS / PSIS
Presidency University

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

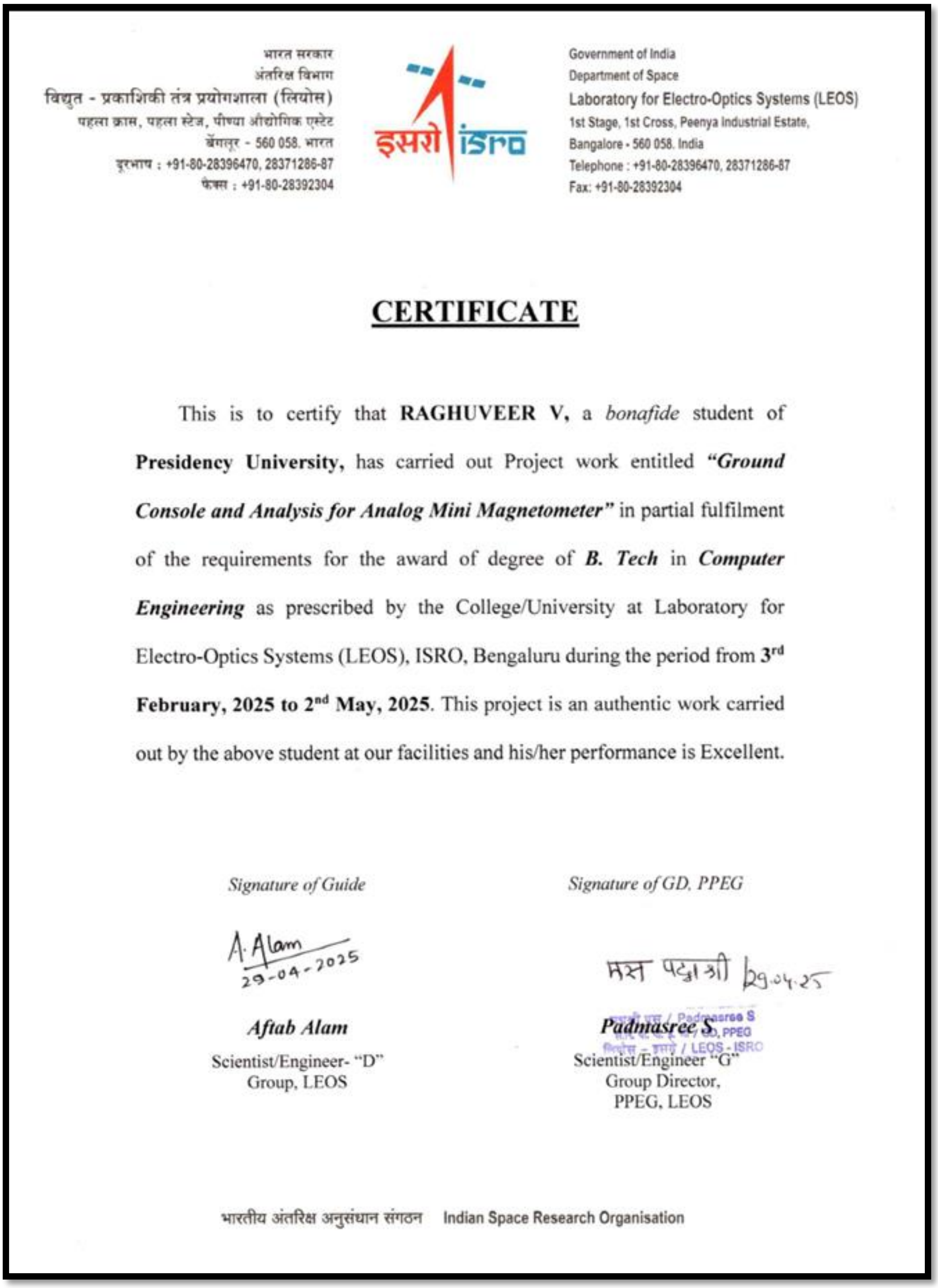
DECLARATION

I hereby declare that the work, which is being presented in the project report entitled “**Ground Console and Analysis for ANALOG MINI MAGNETOMETER”** in partial fulfilment for the award of Degree of **Bachelor of Technology** in **Computer Engineering**, is a record of our own investigations carried under the guidance of **Mr. Mohamed Shakir, Assistant Professor , Presidency School of Computer Science and Engineering , Presidency University, Bengaluru.**

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

RAGHUVEER V (20211COM0030)

INTERNSHIP COMPLETION CERTIFICATE



भारत सरकार
अंतरिक्ष विभाग
विद्युत - प्रकाशिकी तंत्र प्रयोगशाला (लियोस)
पहला क्रास, पहला स्टेज, पीण्या औद्योगिक एस्टेट
बेंगलूर - 560 058. भारत
दूरभाष : +91-80-28396470, 28371286-87
फैक्स : +91-80-28392304

Government of India
Department of Space
Laboratory for Electro-Optics Systems (LEOS)
1st Stage, 1st Cross, Peenya Industrial Estate,
Bangalore - 560 058. India
Telephone : +91-80-28396470, 28371286-87
Fax: +91-80-28392304

CERTIFICATE

This is to certify that **RAGHUVEER V,** a *bonafide* student of **Presidency University,** has carried out Project work entitled ***"Ground Console and Analysis for Analog Mini Magnetometer"*** in partial fulfilment of the requirements for the award of degree of ***B. Tech*** in ***Computer Engineering*** as prescribed by the College/University at Laboratory for Electro-Optics Systems (LEOS), ISRO, Bengaluru during the period from **3rd February, 2025 to 2nd May, 2025**. This project is an authentic work carried out by the above student at our facilities and his/her performance is Excellent.

Signature of Guide

A. Alam
29-04-2025

Aftab Alam
Scientist/Engineer- "D"
Group, LEOS

Signature of GD, PPEG

[illegible] 29.04.25

Padmasree S
GD, PPEG
LEOS - ISRO

Padmasree S
Scientist/Engineer "G"
Group Director,
PPEG, LEOS

भारतीय अंतरिक्ष अनुसंधान संगठन   Indian Space Research Organisation

ABSTRACT

Magnetometers are specialized tools employed for the determination of magnetic field strength, direction, and changes. Magnetometers are used in a number of applications, such as in space exploration, geophysical research, and navigation systems. The present project concerns the application of an Analog Mini Magnetometer (AMM) as a component of ISRO's Gaganyaan mission for the determination of magnetic fields in space. The primary aim is to develop a real-time dashboard and analysis system to read and interpret the magnetic field data recorded by the AMM sensor. The system makes use of an Arduino microcontroller for communication with the AMM sensor, taking magnetic field readings and processing and plotting them in real-time on a Python-based dashboard. The data is also saved into an Excel file for analysis. This live monitoring system not only shows the live visualization of the changing magnetic field in real time but also involves data analysis methods in sophisticated ways, like regression modeling and noise elimination. Regression procedures are applied for the determination of the best fit curve of data, and analysis of noise is conducted to get the influence of the environment over measurements. This project illustrates the use of a Fluxgate Magnetometer, a vector magnetometer, that provides a complete 3D picture of the magnetic field components. Measuring all three magnetic field components (X, Y, and Z), the Fluxgate Magnetometer provides precise and detailed information, essential for space missions where magnetic field measurements are critical to navigation and scientific experiments. The findings of this project will help in the creation of more effective real-time monitoring systems for space applications, which ultimately will assist in the success of future missions such as Gaganyaan.

ACKNOWLEDGEMENTS

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC - Engineering and Dean, Presidency School of Computer Science and Engineering & Presidency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili K Nair,** Presidency School of Computer Science and Engineering, Presidency University, and **Dr. Gopal Krishna Shyam**, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide and reviewer **Mr. Mohamed Shakir**, Assistant Professor Presidency School of Computer Science and Engineering, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 Internship Coordinator **Mr. Md Ziaur Rahman and Dr. Sampath A K,** department Project Coordinators **Mr. Mohamed Shakir** and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

RAGHUVEER V (20211COM0030)

LIST OF TABLES

Sl. No.	**Table Name**	**Table Caption**	**Page No.**
1	Table 1.1	Study of existing methods/technology	5

LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page No.
1	Figure 1.1	LEOS Campus	1
2	Figure 1.2	ISRO Logo	2
3	Figure 1.3	Flux gate Magnetometer	4
4	Figure 4.1	Architecture Diagram	9
5	Figure 6.1	Flow Chart of Main GUI	16
6	Figure 6.2	Flow Chart of Noise Accumulation	17
7	Figure 6.3	Flow Chart of Current stimuli Test	18
8	Figure 6.4	Flowchart of Analysis	19
9	Figure 6.5	Stimuli Excel sheet	20
10	Figure 6.6	Real Time Data Excel sheet	20
11	Figure 6.7	Analysis Excel sheet	21
12	Figure 9.1.1	Real Time Dashboard	27
13	Figure 9.1.2	Serial port Data collection	28
14	Figure 9.1.3	Real-Time Plotting	30
15	Figure 9.1.4	Excel Data Export and Recording	31
16	Figure 9.2.1	Real-Time Noise Accumulation with Zoomable Plot	37
17	Figure 9.2.2	Real-Time Noise Accumulation Zoomed out	37
18	Figure	Real-Time Noise Accumulation Zoomed in	38

	9.2.3		
19	Figure 9.3.1	Current stimuli GUI	39
20	Figure 9.3.2	Current stimuli Serial Data Processing	40
21	Figure 9.3.3	Current stimuli Result Excel Sheet	42
22	Figure 9.4.1	Data collection and user input	44
23	Figure 9.4.2	Linear Regression	45
24	Figure 9.4.3.1	Final Intercept	46
25	Figure 9.4.3.2	Final Slope	46
26	Figure 9.4.4	Polynomial Regression Analysis Excel Sheet	47
27	Figure 9.4.5	Computed Values Excel Sheet	48
28	Figure 9.4.6	Final Output Excel Sheet	49
29	Fig 10.1	Main GUI	62
30	Fig 10.2	Serial port reading and real time plotting	62
31	Fig 10.3	Serial Data accumulation	62
32	Fig 10.4	Real time Serial Data storage	62
33	Fig 10.5	Real time Noise accumulation with zoomable plot	62
34	Fig 10.6	Zoomed out Noise plot	62
35	Fig 10.7	Zoomed in Noise plot	63
36	Fig 10.8	Current Stimuli Test GUI	63
37	Fig 10.9	Current stimuli test processing	63
38	Fig 10.10	Current Stimuli Test Result Excel sheet	63
39	Fig 10.11	Linear Regression	63
40	Fig 10.12	Analysis Console Output	63

41	Fig 10.13	Intercept Plot	64
42	Fig 10.14	Slope Plot	64
43	Fig 10.15	M and C Values Excel sheet	64
44	Fig 10.16	path for the plots	64
45	Fig 10.17	Computed Values	64
46	Fig 10.18	Final output of Analysis (Differences)	64

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**v**
	ACKNOWLEDGEMENT	**vi**
	LIST OF TABLES	**vii**
	LIST OF FIGURES	**viii**
1.	**INTRODUCTION**	**1**
	1.1 Introduction to Company	1
	1.1.1 Introduction to LEOS	1
	1.1.2 Introduction to ISRO	2
	1.2 Introduction to Magnetometer	3
	1.2.1 Types of Magnetometer	3
	1.2.2 Flux Gate Magnetometer	3
	1.2.3 The application of Flux Gate Magnetometer	4
2.	**LITERATURE REVIEW**	**5**
3.	**RESEARCH GAPS OF EXISTING METHODS**	**7**
4.	**PROPOSED METHODOLOGY**	**9**
5.	**OBJECTIVES**	**12**
6.	**SYSTEM DESIGN AND IMPLEMENTATION**	**13**
7.	**TIMELINE FOR EXECUTION OF THE PROJECT**	**23**
8.	**OUTCOMES**	**24**
9.	**RESULTS AND DISCUSSIONS**	**26**
10.	**CONCLUSION**	**50**
	REFERENCES	**51**
	APPENDIX-A	**52**
	APPENDIX-B	**62**
	APPENDIX-C	**65**

CHAPTER-1
INTRODUCTION TO COMPANY
LABORATORY FOR ELECTRO-OPTICS SYSTEMS

Figure 1.1 LEOS

Laboratory for Electro-Optics Systems(LEOS) is a research lab belonging to Indian Space Research Organization. It involves in design and development of optics and sensor modules that can be deployed either onboard satellite or with the launch vehicle.

LEOS was established in 1993, Laboratory for Electro-Optics Systems was established at the same place where the first Indian satellite Aryabhata was fabricated in 1975, namely Bangalore. The laboratory has developed sensors for tracking Earth and Stars for the satellites which were launched when the space research was ushering in India. Satellites like Aryabhata, Bhaskara, Apple, IRS and INSAT-2 have been equipped with the sensors developed by this laboratory. The lab has also participated in the India's first Moon mission, Chandrayaan-1. They have an instrument in ISRO's mission Chandrayan-3 and Aditya-L1 . The Laboratory for Electro-Optics Systems (LEOS) is responsible for design, development and production of electro-optic sensors and optics for spacecraft use. Sensor system includes earth sensors, star sensors, sun sensors, magnetic sensors, fiber optic gyro, temperature sensors and processing electronics. Optics system includes optics for remote sensing cameras, radiometers, star sensor optics, optical filter, optical masks, optical coatings, IR detectors and MEMS based inclinometer.

LEOS is actively involved in the development of new technologies for present / future satellites. This includes development active pixel sensor star tracker, Charge Coupled Device (CCD) based star tracker, Fiber Optics Gyro, Optical inter satellite link, high resolution camera optics, optical coatings and MEMS devices (magnetometer, accelerometer etc.).

INDIAN SPACE RESEARCH ORGANIZATION

Fig 1.2 ISRO LOGO

The **Indian Space Research Organization (ISRO)** is the national space agency of India. It operates as the primary research and development arm of the Department of Space, which is directly overseen by the Prime Minister of India, while the Chairman of ISRO also acts as the executive of DOS. ISRO is primarily responsible for performing tasks related to space-based operations, space exploration, international space cooperation and the development of related technologies. ISRO is one of the six government space agencies in the world that possesses full launch capabilities, can deploy cryogenic engines, can launch extraterrestrial missions and operate a large fleet of artificial satellites. ISRO is one of the four government space agencies to have soft landing capabilities.

ISRO built India's first satellite, Aryabhata, which was launched by the Soviet space agency Interkosmos in 1975.In 1980, ISRO launched satellite RS-1 onboard SLV-3, making India the seventh country to be capable of undertaking orbital launches. SLV-3 was followed by ASLV, which was subsequently succeeded by the development of many medium-lift launch vehicles, rocket engines, satellite systems and networks enabling the agency to launch hundreds of domestic and foreign satellites and various deep space missions for space exploration.
ISRO has the world's largest constellation of remote-sensing satellites and operates the GAGAN and IRNSS (NavIC) satellite navigation systems. It has sent three missions to the Moon and one to Mars.

ISRO's programs have played a significant role in the socio-economic development of India and have supported both civilian and military domains in various aspects including disaster management, telemedicine and navigation and reconnaissance missions. ISRO's spin-off technologies also have founded many crucial innovations for India's engineering and medical industries.

INTRODUCTION

What is a Magnetometer?

A magnetometer is a device that measures the strength of the magnetic field or magnetic dipole moment. A magnetometer is an instrument with a sensor that measures the magnetic flux density. Magnetometers determine the direction, strength, or relative variation in the magnetic field at a certain location.

Types of magnetometers:

1. Vector magnetometers
2. Scalar magnetometers

Vector Magnetometers

Vector magnetometers measure the flux density value in a specific direction in three-dimensional space. One such type is a fluxgate magnetometer. It measures the strength of the component of the earth's field by positioning the sensor in the direction of the desired component.

Flux Gate Magnetometer

A **fluxgate magnetometer** is typically a **vector magnetometer**, meaning it measures the **three components** of the magnetic field (usually the xx, yy, and zz components in a 3D coordinate system). This allows it to detect the **direction** as well as the **magnitude** of the magnetic field in space.

In contrast, a **scalar magnetometer** only measures the **magnitude** of the magnetic field, without providing information about its direction. Scalar magnetometers are less common in precise applications compared to vector magnetometers like fluxgates, which offer more detailed and comprehensive data about the magnetic field.

So, a fluxgate magnetometer can provide a full **vector representation** of the magnetic field, making it more versatile in applications where both the strength and orientation of the field are important.

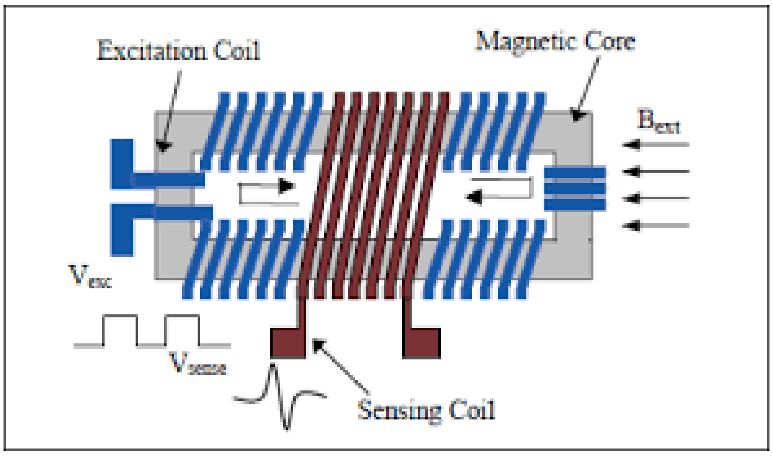


Figure 1.3 Flux Gate Magnetometer

Applications of Flux Gate Magnetometer

- Spacecraft Navigation
- Space weather and Planetary Magnetospheres
- Geomagnetic Field Measurement
- Geophysical Exploration
- UXO detection

CHAPTER-2

LITERATURE SURVEY

Magnetometer sensors, according to sensor literature, have a central application in spaceflight, such as navigation and science measurements, in missions like Gaganyaan. A few of the usual noise reduction approaches, like Kalman filtering and moving averages, are most frequently utilized in order to refine sensor accuracy, although higher-end machine learning solutions are less well-known. Regression analysis, especially linear and polynomial models, is also commonly used to fit sensor data, but more sophisticated models may be beneficial for real-world usage. Python tools such as Tkinter, PySerial, and openpyxl are good for data logging and real-time acquisition, with extensions to sensor systems. Signal processing methods, including Fourier and wavelet transforms, are valuable for filtering magnetometer data, although real-time implementation and incorporation into machine learning is less explored.

Table 1.1: Study of existing methods/technology

Author Name	**Title**	**Introduction**	**Limitations**
[1] J.C. Choi, J.W. Lee	Sensor Data Analysis Using Regression and Machine Learning	Discusses advanced regression techniques and machine learning models for sensor data analysis, particularly in noisy environments.	Limited application to real-time processing and specific sensor types like magnetometers.
[2] J.P. Bristow, P.C. Ford	An Introduction to Magnetometer	Provides an overview of magnetometers, their working principles, and applications in various fields including space missions.	Primarily focused on the fundamentals; lacks real-time processing examples

[3] R.F. McQueen et al.	Application of Magnetometers in Space Missions	Discusses the various uses of magnetometers in space, focusing on their critical role in space missions and magnetic field measurement in space environments.	Limited focus on data processing and analysis, as well as specific sensor configurations for missions like Gaganyaan.
[4] S.M. Nadkarni, A.K. Choudhary	Noise Reduction in Sensor Measurements	Explores methods for reducing noise in sensor measurements, which is crucial for accurate data acquisition in scientific applications.	Does not cover real-time implementation or specific applications to space missions.
[5] S. Sharma, P.K. Gupta	Regression Analysis Techniques in Sensor Data	Introduces regression analysis techniques used in sensor data to perform fitting, noise reduction, and to analyze the sensor performance.	Focuses on theoretical approaches, not directly applicable to real-time data collection and analysis.
[6] R.C. Jain et al.	Use of Python for Data Logging and Analysis	Describes the use of Python in logging and analyzing sensor data, with examples of real-time data logging and analysis tools for scientific applications.	Limited discussion on advanced techniques like machine learning for noise reduction and regression.
[7] M.J. Brown, L.P. Ramos	Real-Time Data Acquisition and Analysis with Python	Discusses how Python can be used for real-time data acquisition and analysis, focusing on the application of Python libraries for sensor data analysis.	Does not cover specific techniques for handling magnetometer data or space mission requirements.

[8] S.K. Gupta et al.	Signal Processing in Magnetometer Data	Focuses on signal processing methods for magnetometer data, including noise filtering, calibration, and analysis of data from space-based sensors.	Limited real-time processing and analysis, focusing more on offline data analysis methods.
[9] J.H. Shun, R.K. Prakash	Multi-Channel Magnetometer Calibration	Explores techniques for the calibration of multi-channel magnetometers used in space missions to ensure accurate magnetic field measurement.	Does not address data analysis techniques or real-time processing in space missions.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

Magnetic field data from sensors like the AMM are vital for spacecraft stability in missions like ISRO's Gaganyaan, but current methods are limited by manual processes and lack real-time automation.

1. Utilization of Multimeter for Field Voltage

- Simple multimeters are usually applied to measure the analog voltage representing the strength of the magnetic field. Although this technique is easy and common, it has the following drawbacks:
- Lack of accuracy in highly dynamic magnetic environments.
- No time resolution – measurements are pointwise and not continuous.
- No storing or logging – one must write down values manually, which is inefficient and prone to error.

2. Digital Oscilloscope for Noise Analysis

- Oscilloscopes are often employed to monitor noise in the signal. But:
- They give only a visual display, without automatic measurement of noise levels.
- Interpretation by hand is required to determine the nature and effect of the noise.
- Not ideal for long-duration monitoring or logging, particularly in real-time.

3. Power Supply as a Constraint

- Most configurations employ lab-based or fixed power supplies which:
- Are not portable and need lab facilities.
- Do not incorporate sensor feedback for adaptive powering.
- Cannot replicate in-flight field conditions as seen on space missions.

4. Data Reading Manually

- Manual data acquisition is reading values from analog instruments or indicators:
- Time-consuming and susceptible to human error.
- Not ideal for real-time decision-making.
- No ability to correlate data with environmental or time-dependent events.

5. Analysis using MS Excel

- Although Excel is popularly used for plotting and simple analysis:
- It does not support advanced statistical capabilities necessary for regression, noise modelling, and outlier detection.
- Manual data entry or import is necessary, which is a limitation in supporting real-time functionality.
- Inadequate for automatic or ongoing data processing.

CHAPTER-4

PROPOSED METHODOLOGY

The system was designed based on an **MVC (Model-View-Controller)** architecture to ensure separation of concerns.

- User Interface (View): The real-time dashboard and data visualization were designed using Python's Matplotlib and Tkinter libraries. The graphical interface provides a seamless way to monitor magnetic field data, visualize sensor readings, and interact with the system.
- Backend Logic (Controller): The backend is responsible for reading data from the Arduino sensor via PySerial, processing the incoming magnetic field data, and managing the communication between the hardware and the visualization system. The data analysis, including regression and noise filtering, is handled by Python scripts, ensuring efficient data processing.
- Data Management (Model): Data storage and logging are managed using Excel (via the openpyxl Python library) for recording sensor data in real time. The data is stored in a structured format to enable easy retrieval for further analysis.

ARCHITECTURE

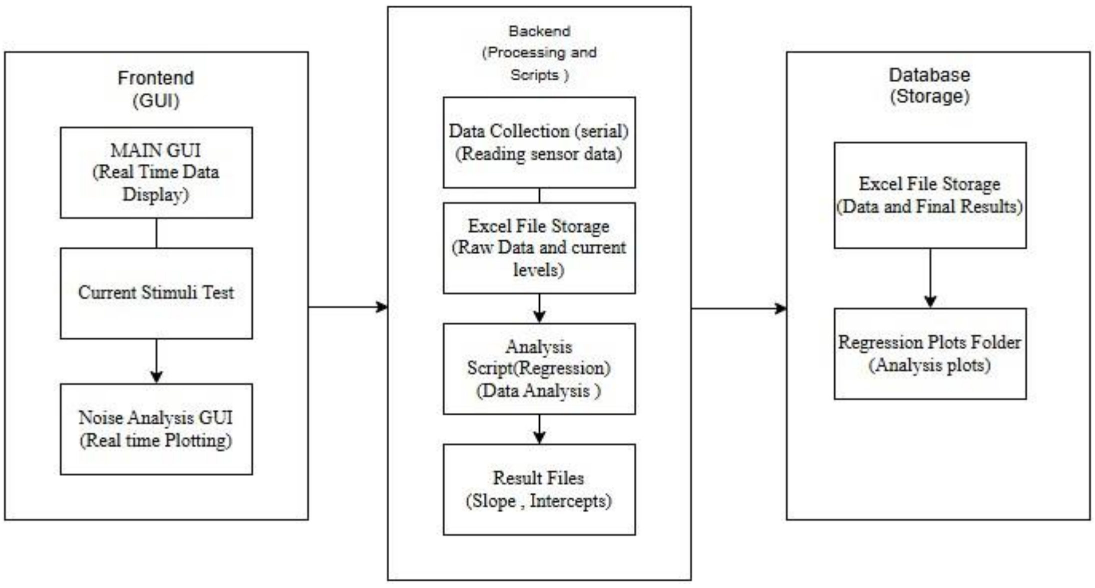


Figure 4.1: Architecture Diagram

MODULES

The application is divided into the following modules:

- **Sensor Data Acquisition:**
  This module handles the collection of magnetic field data from the Analog Mini Magnetometer (AMM) using an Arduino. The sensor reads the magnetic field in real-time and sends the data to the Python backend using PySerial for further processing.

- **Data Processing:**
  Once the data is received, it is processed in real-time to filter noise and perform regression analysis to identify trends and calculate errors. The noise reduction algorithms (like Kalman Filtering or Moving Averages) are applied to ensure that only clean data is used for analysis.

- **Data Visualization:**
  The real-time magnetic field data is then visualized using Python's Matplotlib and Tkinter. The dashboard displays live graphs of magnetic field strength, orientation, and other key parameters in real-time. This module provides mission control operators with an easy-to-read interface for monitoring satellite status.

- **Data Logging:**
  The collected and processed data is logged into an Excel file using the openpyxl Python library. This allows for offline storage of sensor data and makes it easy to review and analyze later.

  - ➢ **Data Storage:** Excel File (Local Storage): The data is logged to an Excel file, ensuring that each data point is saved with a timestamp for future analysis. This enables offline functionality and a reliable record of the sensor data.

  - ➢ **Regression Analysis Results (Model Storage):** The results of the regression analysis and any corrective measures (based on noise reduction) are saved in a separate Excel sheet, providing clear insights into the performance of the satellite's orientation and trajectory.

- **Satellite Orientation and Navigation:**
  Based on the real-time data, the system aids in the orientation and navigation of the satellite by calculating the deviations in magnetic field strength and adjusting the satellite's movement accordingly. The system's continuous feedback ensures that the satellite maintains its trajectory and orientation within the desired parameters.

CHAPTER-5

OBJECTIVES

1. Graphical User Interface (GUI)

The primary objective under GUI development is to provide an intuitive and efficient front-end interface that supports real-time interaction with the Analog Mini Magnetometer data.

- **Data Acquisition**
  Develop a robust communication link between the Arduino (connected to the AMM sensor) and the Python environment using PySerial. This will facilitate seamless and continuous acquisition of real-time magnetic field values.
- **Real-Time Plotting**
  Implement dynamic plotting of magnetic field data using libraries such as matplotlib or plotly to visualize magnetic field fluctuations in real-time. This will aid in observing sensor response under various conditions.
- **Noise Acquisition**
  Capture and visualize noise patterns by analyzing signal deviations from expected magnetic field trends. This will involve integrating basic filters or statistical methods (e.g., moving average, standard deviation) in real-time to differentiate noise from the true signal.

2. Data Analysis

This section focuses on post-processing of the acquired data for performance evaluation and calibration.

- **Temperature Calibration**
  Study the correlation between magnetic field readings and ambient temperature. This involves recording field values across a range of temperatures and understanding how temperature variations affect the sensor’s accuracy.
- **Input: Field at Different Temperatures**
  Collect datasets by varying the temperature in a controlled manner and recording the corresponding magnetic field measurements from the AMM sensor.
- **Curve Fitting to Derive Field-Temperature Equation**
  Apply regression analysis (e.g., linear, polynomial) to model the relationship between temperature and magnetic field strength. This model will help calibrate the AMM sensor and compensate for temperature-induced errors.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

System Design:

Hardware:

1. **Microcontroller: Arduino Mega 2560**

- **Model:** Arduino Mega 2560
- **Purpose:** Acts as the central controller to interface between the analog sensors (such as the magnetometer) and the computer for data collection.
- **Features:**
- **Microcontroller:** ATmega2560
- **Digital I/O Pins:** 54
- **Analog Input Pins:** 16 (for sensor input)
- **Communication Ports:**
- 4 UART (Serial Communication)
- SPI, I2C for communication with external devices
- **Power Supply:** Can be powered via USB or external power supply (9-12V)
- **Interface:** Connects to the computer via USB (for serial communication)
- **Role:** The Arduino Mega reads data from the sensors and sends it to the computer through a serial connection, where the Python code processes the data.

2. **Analog Mini Magnetometer (Sensor)**

- **Model:** Analog Mini Magnetometer
- **Purpose:** Measures magnetic field strength and orientation (used in the system for collecting sensor data).
- **Features:**
- **Measurement Range:** ±1.3 to ±8 Gauss (dependent on sensor configuration)
- **Output Type:** Analog output (typically a varying voltage)
- **Communication:** Analog signal output that is fed into the Arduino’s analog input pins.
- **Sensitivity:** The sensitivity varies based on the orientation and strength of the magnetic field.
- **Power Consumption:** Low power consumption, typically operating at 3.3V to 5V.

- **Role:** The magnetometer measures the magnetic field and outputs the corresponding analog voltage. The Arduino Mega reads this analog voltage and sends the data via the serial port to the computer.

Software:

1. Frontend:

- **Platform:** Desktop Application using Python's Tkinter for GUI.
- **UI Components:** Tkinter widgets (Buttons, Labels, Text boxes, Scrolled Text Areas, Plot Canvas).
- **Libraries Used:**
  - Tkinter for user interface.
  - Matplotlib for live and static plotting.
  - ScrolledText for displaying sensor values.
  - Threading for background tasks.
  - Openpyxl for Excel manipulation.

2. Backend:

- **Database:**
  - Excel files used as structured data storage using openpyxl and pandas.
  - File-based storage with proper sheet segregation for reference data, computed values, differences, regression results, and plots.
- **Business Logic:**
  - Python classes and functions for:
  - Real-time serial data reading.
  - Signal conversion from sensor input.
  - Regression calculations.
  - Noise accumulation and polynomial modeling.
  - GUI interactions.
  - Data visualization and storage.

3. Key Features:

Role-specific functionality across scripts:

- **Current_stimuli_test.py:** Handles fixed current stimuli testing and result recording.
- **Noise_accumulation.py:** Performs real-time noise accumulation and visualization with zooming.
- **Main GUI:** Central hub to collect data, trigger other tools, and visualize live plots.
- **Analysis.py:** Performs regression analysis, polynomial modeling, and error computations.

Interactive GUI for:

- Reading sensor data.
- Monitoring real-time plots.
- Navigating to noise and current testing tools.
- Opening generated Excel reports.
- Modular plotting using matplotlib embedded in GUI.
- Real-time responsiveness using multi-threading.

User Flow:

1. Real-Time Data Collection (Main GUI):

- User starts the main GUI.
- Clicks "Start Process" to begin sensor data collection.
- Data is displayed in real-time and saved to Excel.
- Upon clicking "Stop Process", averages are computed and stored.

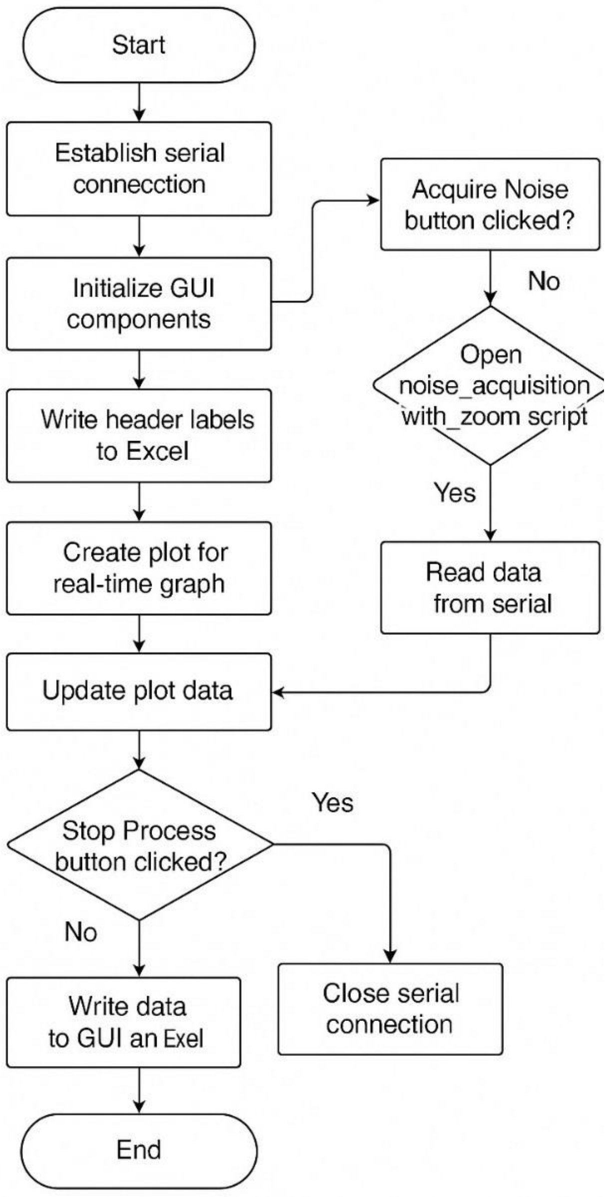

Start
Establish serial connecction
Initialize GUI components
Write header labels to Excel
Create plot for real-time graph
Update plot data
Acquire Noise button clicked?
No
Open noise_acquisition with_zoom script
Yes
Read data from serial
Stop Process button clicked?
Yes
No
Write data to GUI an Exel
Close serial connection
End

Figure 6.1 Flow chart of Main GUI

2. Noise Accumulation:

- User launches noise module.
- Data from the serial port is plotted in real time.
- Zooming is enabled for deeper analysis.
- Data collection stops when "Close" is pressed.

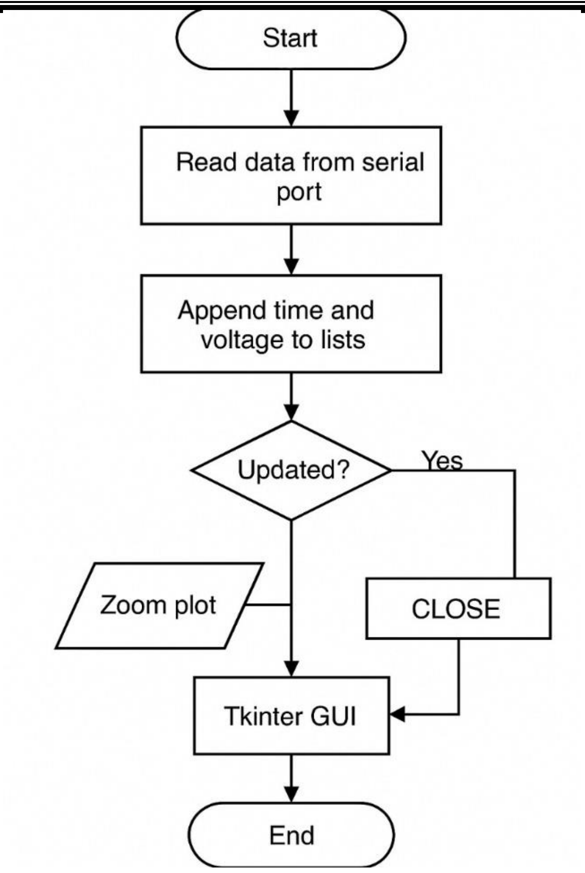


Figure 6.2 Flow chart of Noise Accumulation

3. Current Stimuli Test:

- User selects a current level (-4mA to +4mA) using buttons.
- Voltage values are recorded and averaged per level.
- Upon pressing "STOP", calculations are made:
- Average voltages.
- Difference from 0mA.
- Alternate difference from mirror currents.
- Data is saved into stimuli_test.xlsx.

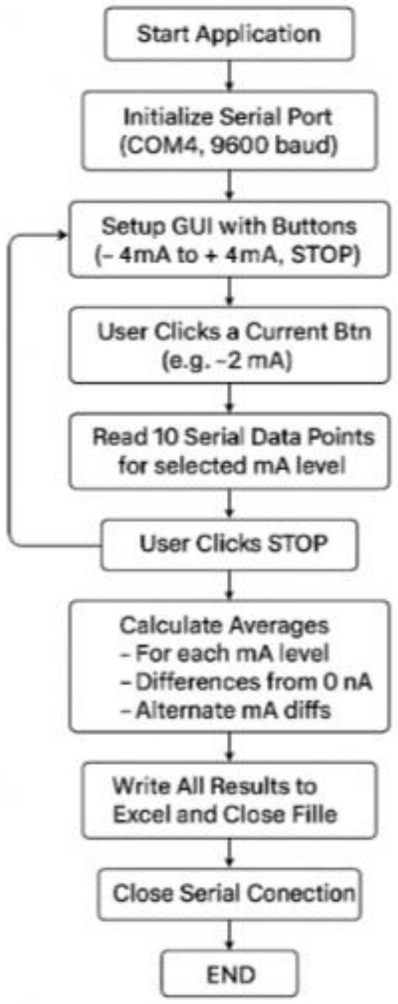

Start Application
Initialize Serial Port (COM4, 9600 baud)
Setup GUI with Buttons (- 4mA to + 4mA, STOP)
User Clicks a Current Btn (e.g. -2 mA)
Read 10 Serial Data Points for selected mA level
User Clicks STOP
Calculate Averages - For each mA level - Differences from 0 nA - Alternate mA diffs
Write All Results to Excel and Close Fille
Close Serial Conection
END

Figure 6.3 Flowchart of Current Stimuli Test

4. Analysis Module:

- User provides slope, intercept, and a reference temperature column.
- Performs:
- Linear regression per column.
- Polynomial regression for slope and intercept.
- Error difference computation.
- Saves:
- All regression plots.
- Final computed values.
- Differences with overall max error.

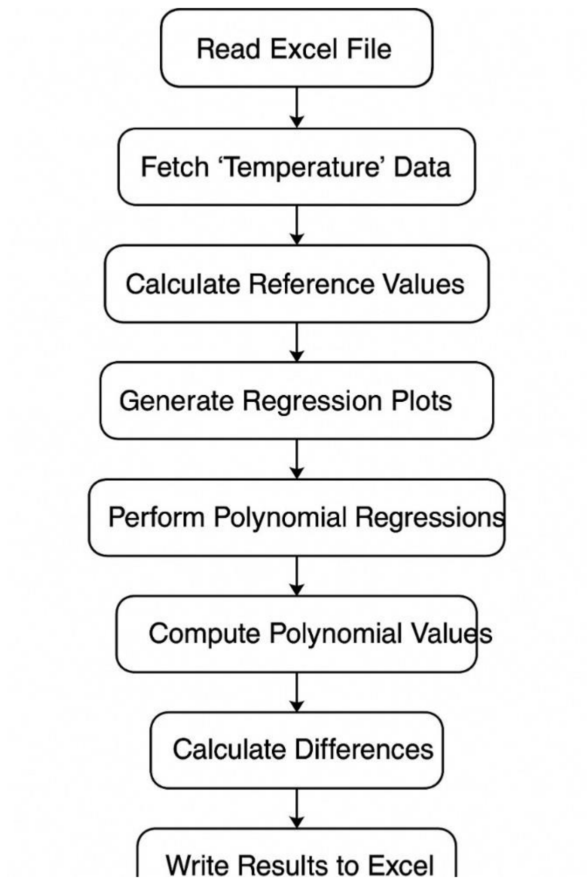
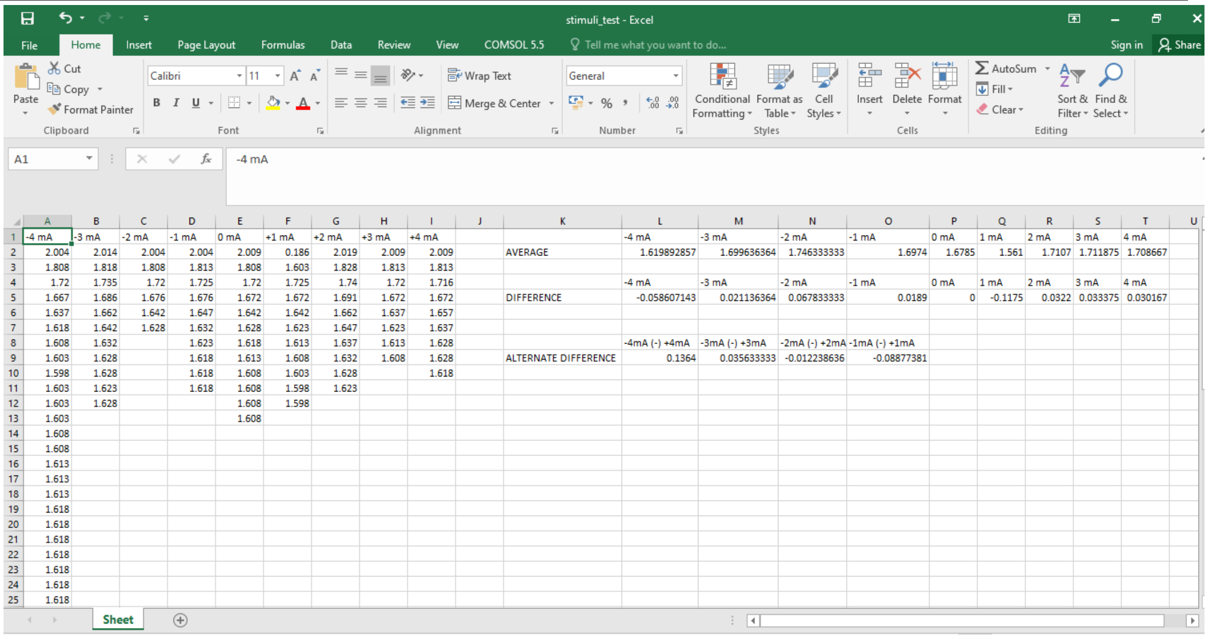

Read Excel File
Fetch 'Temperature' Data
Calculate Reference Values
Generate Regression Plots
Perform Polynomial Regressions
Compute Polynomial Values
Calculate Differences
Write Results to Excel

Figure 6.4 Flowchart of Analysis

Data Collection (Excel Sheets):

1. For Stimuli Test (stimuli_test.xlsx):

- Columns for each current level (-4 to +4 mA).
- Rows:
- AVERAGE
- DIFFERENCE (from 0 mA)
- ALTERNATE DIFFERENCE (e.g., +1 vs -1 mA)



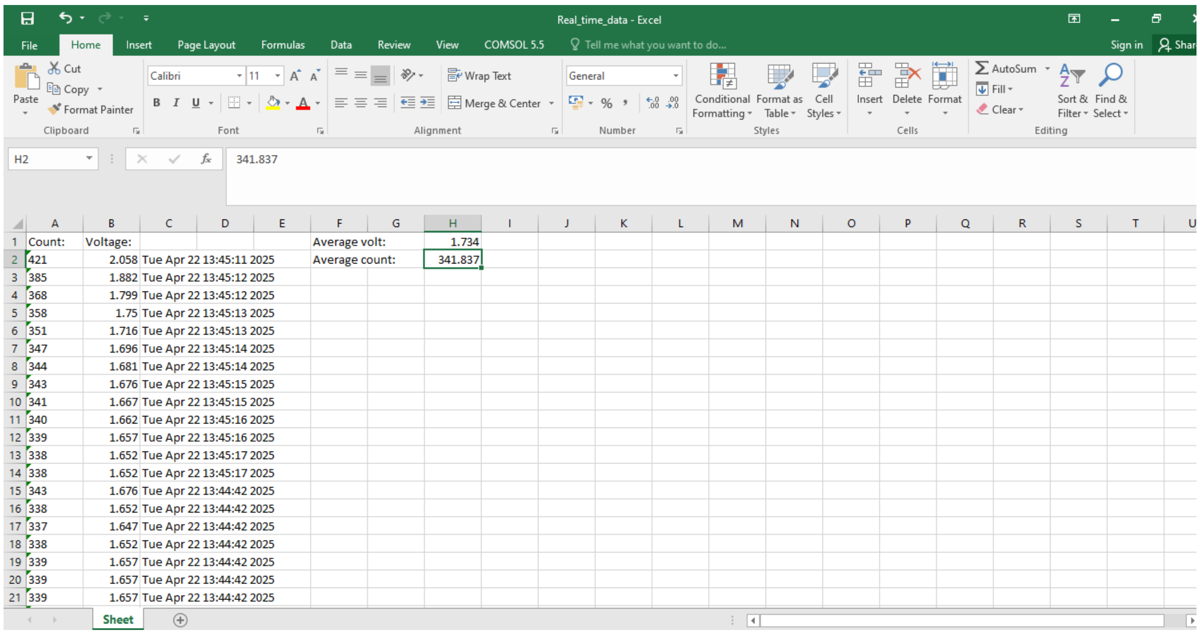
	A	B	C	D	E	F	G	H	I
1	-4 mA	-3 mA	-2 mA	-1 mA	0 mA	+1 mA	+2 mA	+3 mA	+4 mA
2	2.004	2.014	2.004	2.004	2.009	0.186	2.019	2.009	2.009
3	1.808	1.818	1.808	1.813	1.808	1.603	1.828	1.813	1.813
4	1.72	1.735	1.72	1.725	1.72	1.725	1.74	1.72	1.716
5	1.667	1.686	1.676	1.676	1.672	1.672	1.691	1.672	1.672
6	1.637	1.662	1.642	1.647	1.642	1.642	1.662	1.637	1.657
7	1.618	1.642	1.628	1.632	1.628	1.623	1.647	1.623	1.637
8	1.608	1.632		1.623	1.618	1.613	1.637	1.613	1.628
9	1.603	1.628		1.618	1.613	1.608	1.632	1.608	1.628
10	1.598	1.628		1.618	1.608	1.603	1.628		1.618
11	1.603	1.623		1.618	1.608	1.598	1.623		
12	1.603	1.628			1.608	1.598			
13	1.603				1.608				
14	1.608								
15	1.608								
16	1.613								
17	1.613								
18	1.613								
19	1.618								
20	1.618								
21	1.618								
22	1.618								
23	1.618								
24	1.618								
25	1.618								

	K	L	M	N	O	P	Q	R	S	T
1		-4 mA	-3 mA	-2 mA	-1 mA	0 mA	1 mA	2 mA	3 mA	4 mA
2	AVERAGE	1.619892857	1.699636364	1.746333333	1.6974	1.6785	1.561	1.7107	1.711875	1.708667
3										
4		-4 mA	-3 mA	-2 mA	-1 mA	0 mA	1 mA	2 mA	3 mA	4 mA
5	DIFFERENCE	-0.058607143	0.021136364	0.067833333	0.0189	0	-0.1175	0.0322	0.033375	0.030167
6										
7										
8		-4mA (-) +4mA	-3mA (-) +3mA	-2mA (-) +2mA	-1mA (-) +1mA					
9	ALTERNATE DIFFERENCE	0.1364	0.035633333	-0.012238636	-0.08877381					

Figure 6.5 Stimuli Excel sheet

2. For Noise Data (Real_time_data.xlsx):

Columns: Count, Average Count, Voltage, Average Voltage, Timestamp

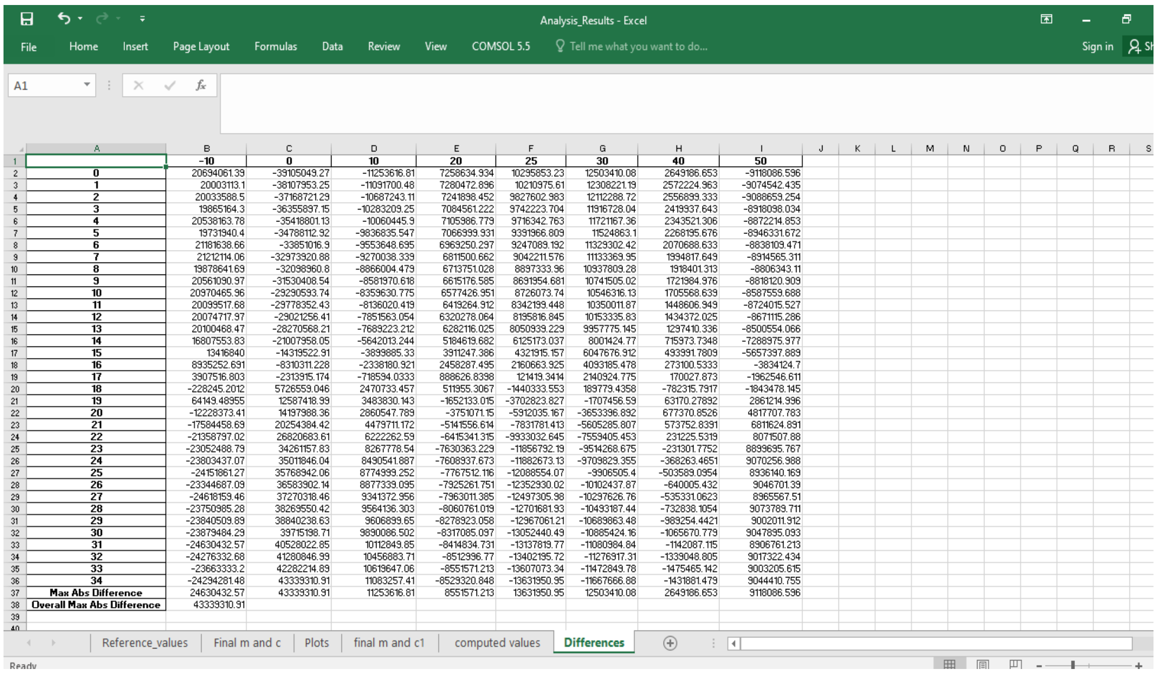


	A	B	C	F	H
1	Count:	Voltage:		Average volt:	1.734
2	421	2.058	Tue Apr 22 13:45:11 2025	Average count:	341.837
3	385	1.882	Tue Apr 22 13:45:12 2025		
4	368	1.799	Tue Apr 22 13:45:12 2025		
5	358	1.75	Tue Apr 22 13:45:13 2025		
6	351	1.716	Tue Apr 22 13:45:13 2025		
7	347	1.696	Tue Apr 22 13:45:14 2025		
8	344	1.681	Tue Apr 22 13:45:14 2025		
9	343	1.676	Tue Apr 22 13:45:15 2025		
10	341	1.667	Tue Apr 22 13:45:15 2025		
11	340	1.662	Tue Apr 22 13:45:16 2025		
12	339	1.657	Tue Apr 22 13:45:16 2025		
13	338	1.652	Tue Apr 22 13:45:17 2025		
14	338	1.652	Tue Apr 22 13:45:17 2025		
15	343	1.676	Tue Apr 22 13:44:42 2025		
16	338	1.652	Tue Apr 22 13:44:42 2025		
17	337	1.647	Tue Apr 22 13:44:42 2025		
18	338	1.652	Tue Apr 22 13:44:42 2025		
19	339	1.657	Tue Apr 22 13:44:42 2025		
20	339	1.657	Tue Apr 22 13:44:42 2025		
21	339	1.657	Tue Apr 22 13:44:42 2025		

Figure 6.6 Real Time Data Excel sheet

3. For Analysis (Analysis_Results.xlsx):

- Sheet: Reference values → Reference slope line values.
- Sheet: Final m and c → Slope and intercept per column.
- Sheet: Plots → File paths of regression plots.
- Sheet: Computed values → Final Y = mx + c for each column.
- Sheet: Differences → Differences from reference with max values included.



	-10	0	10	20	25	30	40	50
0	20694061.39	-39105049.27	-11253616.81	7258634.934	10295853.23	12503410.08	2649186.653	-9118086.596
1	20003113.1	-38107953.25	-11091700.48	7280472.896	10210975.61	12308221.19	2572224.963	-9074542.435
2	20033588.5	-37168721.29	-10687243.11	7241898.452	9827602.983	12112288.72	2556899.333	-9088659.254
3	19865164.3	-36355897.15	-10283209.25	7084561.222	9742223.704	11916728.04	2419937.643	-8918098.034
4	20538163.78	-35418801.13	-10060445.9	7105986.779	9716342.763	11721167.36	2343521.306	-8872214.853
5	19731940.4	-34788112.92	-9836835.547	7066999.931	9391966.809	11524863.1	2268195.676	-8946331.672
6	21181638.66	-33851016.9	-9553648.695	6969250.297	9247089.192	11329302.42	2070688.633	-8838109.471
7	21212114.06	-32973920.88	-9270038.339	6811500.662	9042211.576	11133369.95	1994817.649	-8914565.311
8	19878641.69	-32098960.8	-8866004.479	6713751.028	8897333.96	10937809.28	1918401.313	-8806343.11
9	20561090.97	-31530408.54	-8581970.618	6615176.585	8691954.681	10741505.02	1721984.976	-8818120.909
10	20970465.96	-29290593.74	-8359630.775	6577426.951	8726073.74	10546316.13	1705568.639	-8587559.688
11	20099517.68	-29778352.43	-8136020.419	6419264.912	8342199.448	10350011.87	1448606.949	-8724015.527
12	20074717.97	-29021256.41	-7851563.054	6320278.064	8195816.845	10153335.83	1434372.025	-8671115.286
13	20100468.47	-28270568.21	-7689223.212	6282116.025	8050939.229	9957775.145	1297410.336	-8500554.066
14	16807553.83	-21007958.05	-5642013.244	5184619.682	6125173.037	8001424.77	715973.7348	-7288975.977
15	13416840	-14319522.91	-3899885.33	3911247.386	4321915.157	6047676.912	493991.7809	-5657397.889
16	8935252.691	-8310311.228	-2338180.921	2458287.495	2160663.925	4093185.478	273100.5333	-3834124.7
17	3907516.803	-2313915.174	-718594.0333	888626.8398	121419.3414	2140924.775	170027.873	-1962546.611
18	-228245.2012	5726559.046	2470733.457	511955.3067	-1440333.553	189779.4358	-782315.7917	-1843478.145
19	64149.48955	12587418.99	3483830.143	-1652133.015	-3702823.827	-1707456.59	63170.27892	2861214.996
20	-12228373.41	14197988.36	2860547.789	-3751071.15	-5912035.167	-3653396.892	677370.8526	4817707.783
21	-17584458.69	20254384.42	4479711.172	-5141556.614	-7831781.413	-5605285.807	573752.8391	6811624.891
22	-21358797.02	26820683.61	6222262.59	-6415341.315	-9933032.645	-7559405.453	231225.5319	8071507.88
23	-23052488.79	34261157.83	8267778.54	-7630363.229	-11856792.19	-9514268.675	-231301.7752	8899695.767
24	-23803437.07	35011846.04	8490541.887	-7608937.673	-11882673.13	-9709829.355	-368263.4651	9070256.988
25	-24151861.27	35768942.06	8774999.252	-7767512.116	-12088554.07	-9906505.4	-503589.0954	8936140.169
26	-23344687.09	36583902.14	8877339.095	-7925261.751	-12352930.02	-10102437.87	-640005.432	9046701.39
27	-24618159.46	37270318.46	9341372.956	-7963011.385	-12497305.98	-10297626.76	-535331.0623	8965567.51
28	-23750985.28	38269550.42	9564136.303	-8060761.019	-12701681.93	-10493187.44	-732838.1054	9073789.711
29	-23840509.89	38840238.63	9606899.65	-8278923.058	-12967061.21	-10689863.48	-989254.4421	9002011.912
30	-23879484.29	39715198.71	9890086.502	-8317085.097	-13052440.49	-10885424.16	-1065670.779	9047895.093
31	-24630432.57	40528022.85	10112849.85	-8414834.731	-13137819.77	-11080984.84	-1142087.115	8906761.213
32	-24276332.68	41280846.99	10456883.71	-8512996.77	-13402195.72	-11276917.31	-1339048.805	9017322.434
33	-23663333.2	42282214.89	10619647.06	-8551571.213	-13607073.34	-11472849.78	-1475465.142	9003205.615
34	-24294281.48	43339310.91	11083257.41	-8529320.848	-13631950.95	-11667666.88	-1431881.479	9044410.755
Max Abs Difference	24630432.57	43339310.91	11253616.81	8551571.213	13631950.95	12503410.08	2649186.653	9118086.596
Overall Max Abs Difference	43339310.91							

Figure 6.7 Analysis Excel sheet

Implementation Plan

1. Environment Setup:

- Python 3.10+
- Required libraries: tkinter, matplotlib, NumPy, pandas, openpyxl, serial, os, threading

2. UI Design:

- Use Tkinter to create interactive windows with:
- Scrollable text areas
- Buttons for operations
- Embedded plots

3. Core Functionalities:

- Implement separate scripts:
- Real-time voltage plotting
- Stimuli testing with Excel export
- Full regression and error analysis

4. Testing and Debugging:

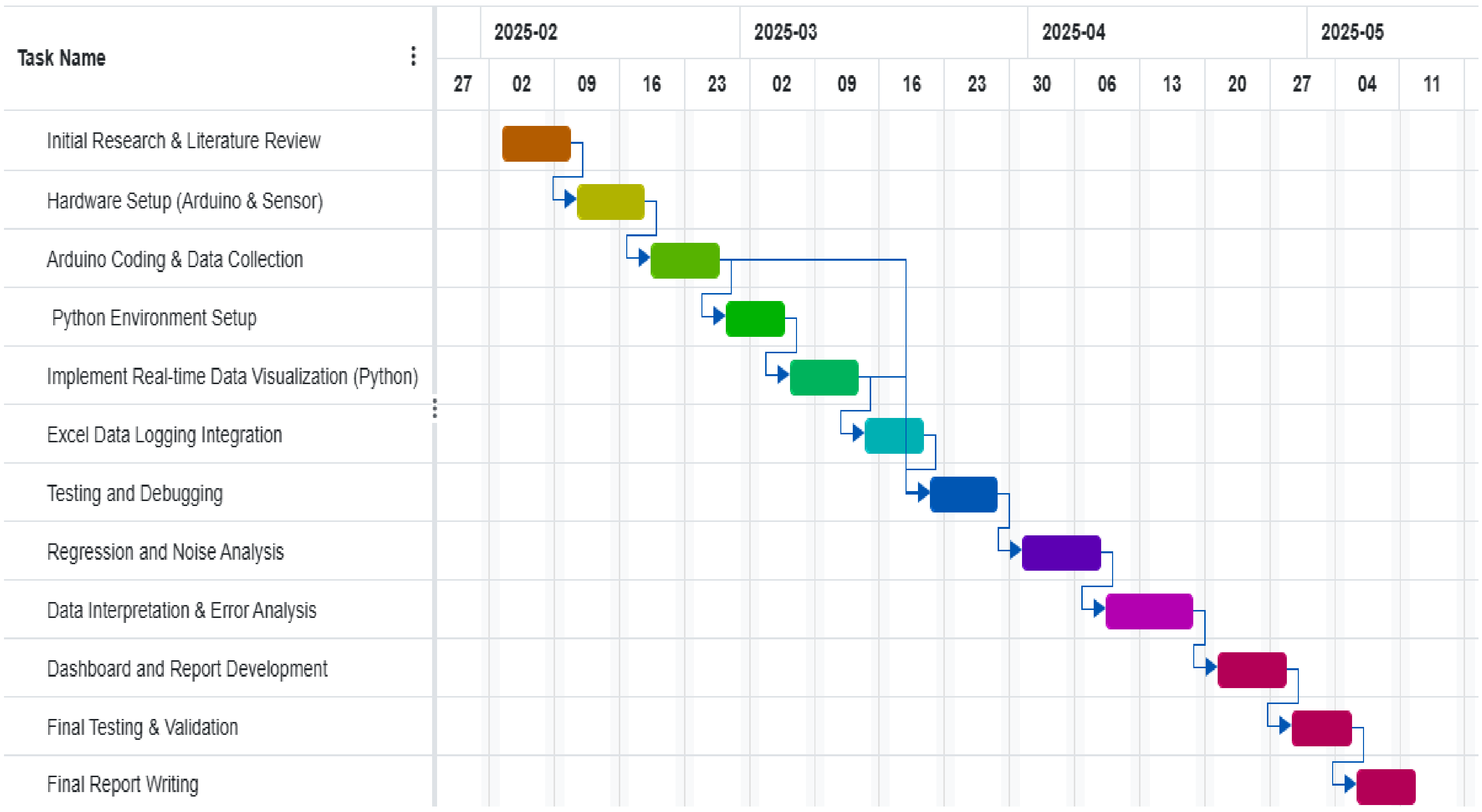
- Simulate serial data for offline testing
- Validate Excel data structure
- Test UI responsiveness with multiple interactions

5. Deployment:

- Package scripts with PyInstaller for standalone use
- Organize folders for plots, data, and logs

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)



CHAPTER-8

OUTCOMES

1. **Improved Satellite Orientation and Navigation:** Through the use of magnetic field data gathered from the Analog Mini Magnetometer (AMM) and processed via real-time analysis, the satellite will be able to accurately establish its orientation in space. This enables accurate satellite alignment, which is vital for mission success. Magnetic field measurements guarantee that the satellite is correctly oriented, avoiding any drift or directional errors that may jeopardize mission goals.

2. **Optimum Satellite Motion and Path Control:** The real-time logging and analysis of data enable accurate control over the movement of the satellite through its orbit. Through the constant observation of the magnetic field fluctuations, the system can adjust the satellite's orbital modifications in fine detail to keep it on course. Such is particularly important in space missions such as Gaganyaan, where slight deviations can translate into serious operational problems. Detection and measurement of small magnetic variations ensure seamless satellite navigation.

3. **Real-time Decision Making for Orbital Corrections:** Through regression analysis of the data, the system can identify deviations or anomalies in the satellite's trajectory. With this real-time functionality, mission control can make well-informed decisions regarding adjustments or corrections that may be needed. The data-driven method minimizes the possibility of errors and maximizes the satellite's capacity to stay on course without outside interference, further adding to the success of the mission.

4. **Enhanced Mission Safety and Reliability:** The onboard integration of the Fluxgate magnetometer data enhances the safety and reliability of the mission. With constant monitoring of the magnetic field and realignment of the satellite's orientation and motion, the risk of satellite failure from misalignment or trajectory faults is eliminated. This ensures the mission is safe, reliable, and efficient from launch to completion.

5. **Simplified Mission Monitoring and Control:** The real-time logging system and dashboard give mission control a readily accessible picture of the status of the satellite. Ground-based operators can easily gauge the condition of the satellite by observing the magnetic field data and movement of the satellite in real-time and respond accordingly to any problems. This process of monitoring, streamlined as it is, makes the overall management of the satellite more efficient, and the mission operations smoother and less susceptible to human errors.

CHAPTER-9

RESULTS AND DISCUSSIONS

MAIN_GUI .PY

1. RealTimeOutputGUI (Real-Time Data Acquisition and Display)

Functionality:

- The RealTimeOutputGUI class provides a graphical user interface (GUI) for real-time data acquisition from a sensor via serial communication. It displays the count and voltage data, updates these readings in real-time, and generates a live voltage vs. time graph.
- The GUI also allows users to start and stop data collection, view the average readings, and export data to an Excel file.

Key Points:

- **Data Collection:** The application collects real-time data from a sensor over a serial connection and displays both count and voltage readings in the GUI.
- **Graphing:** Real-time updates are shown on a voltage vs. time graph using matplotlib, which is embedded in the Tkinter window.
- **Data Export:** Users can export the collected data to an Excel sheet for further analysis.
- **Noise and Stimuli Tests:** Additional functionalities like acquiring noise and performing current stimuli tests are available through buttons in the interface.
- **Strengths:**
- **Real-Time Feedback:** The data is updated in real-time, providing immediate feedback to users through text fields and graphs.
- **Interactive Visualization:** The integration of matplotlib for live plotting makes it easy for users to monitor data trends visually.
- **Excel Export:** The ability to export data to Excel is an essential feature, allowing users to save the readings for later analysis or reporting.
- **Ease of Use:** The GUI is user-friendly, with clearly labelled buttons and text fields, making it accessible even to users with minimal technical knowledge.

Improvements:

- **Error Handling:** The application could benefit from enhanced error handling, especially when dealing with serial communication issues or file writing errors.
- **User Input Validation:** The program should include validation checks for the inputs where applicable, such as ensuring the correct serial port is selected or handling unexpected data formats.
- **UI Responsiveness:** For a smoother user experience, the interface could be made more responsive, particularly when handling large amounts of data or when performing long-running tasks.
- **Advanced Plotting:** The plot could be enhanced with features like zooming, panning, and interactive tooltips to better analyse the data.

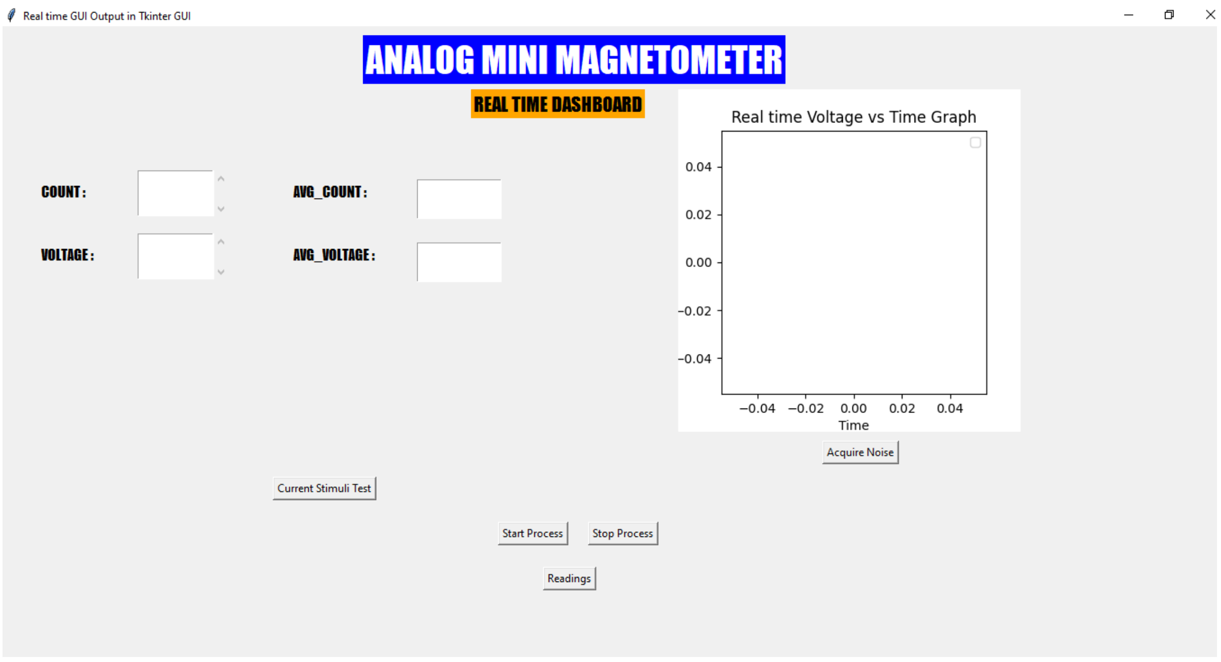


Figure 9.1.1 Real Time Dashboard

2. Serial Data Collection and Processing

Functionality:

- Data is collected in real-time from the serial port (COM6) using the serial library. The program continuously reads data from the sensor, processes it into usable count and voltage values, and updates the display in the GUI.
- Each reading is stored temporarily in lists (count_array, value_array, time_array)

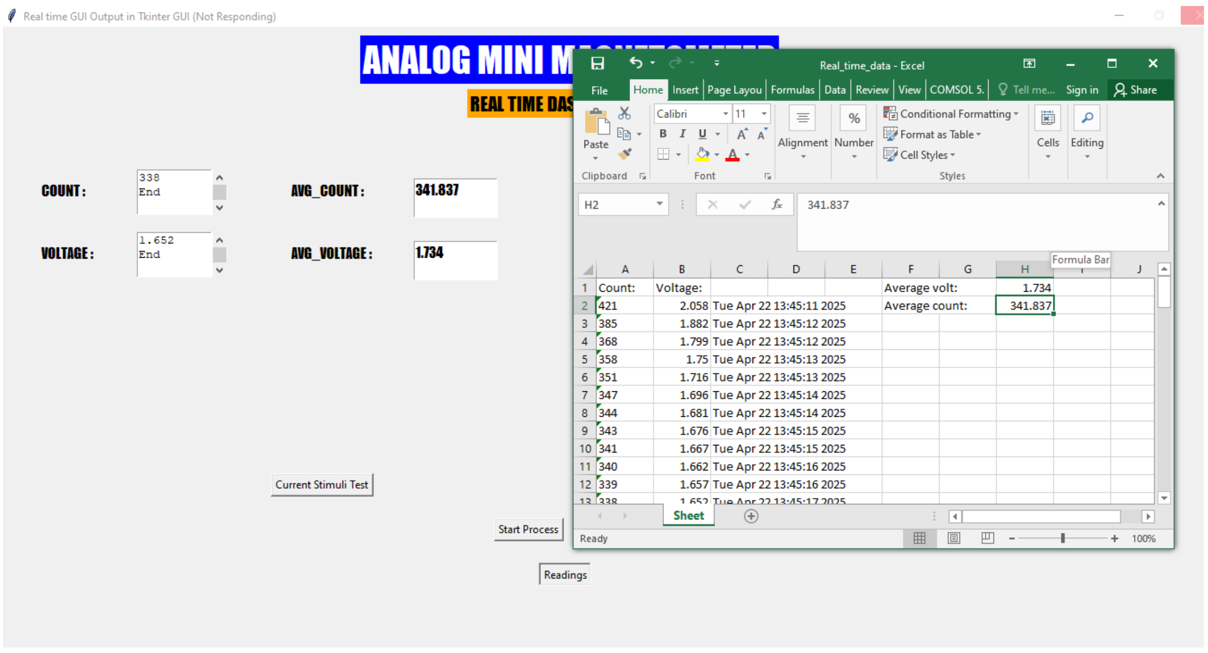
and later used to calculate averages and generate the voltage vs. time plot.

Key Points:

- **Data Conversion:** The raw sensor readings (count) are converted into voltage values using a scaling factor based on the sensor's specifications.
- **Real-Time Data Management:** The sensor data is managed in arrays to track the count and voltage, which are used for plotting and generating statistical values.
- **Multithreading:** Data collection is run in a separate thread to prevent freezing the GUI during long data collection sessions.
- **Strengths:**
- **Continuous Data Collection:** The system continuously collects data in the background while allowing users to interact with the GUI, providing a smooth experience.
- **Data Processing Efficiency:** The system processes and converts sensor data efficiently, ensuring minimal delay between data collection and GUI updates.
- **Threading for Performance:** The use of threading allows real-time data collection without blocking the main GUI thread, ensuring the interface remains responsive.

Improvements:

- **Serial Communication Error Handling:** Improved handling of errors like disconnections or communication issues with the sensor would increase system reliability.
- **Data Integrity Checks:** Implement checks to ensure the integrity of the data received from the sensor, such as verifying the data format or filtering out anomalies.



COUNT : 338 End

VOLTAGE : 1.652 End

AVG_COUNT : 341.837

AVG_VOLTAGE : 1.734

H2 | 341.837

	A	B	C	D	E	F	G	H
1	Count:	Voltage:				Average volt:		1.734
2	421	2.058	Tue Apr 22 13:45:11 2025			Average count:		341.837
3	385	1.882	Tue Apr 22 13:45:12 2025					
4	368	1.799	Tue Apr 22 13:45:12 2025					
5	358	1.75	Tue Apr 22 13:45:13 2025					
6	351	1.716	Tue Apr 22 13:45:13 2025					
7	347	1.696	Tue Apr 22 13:45:14 2025					
8	344	1.681	Tue Apr 22 13:45:14 2025					
9	343	1.676	Tue Apr 22 13:45:15 2025					
10	341	1.667	Tue Apr 22 13:45:15 2025					
11	340	1.662	Tue Apr 22 13:45:16 2025					
12	339	1.657	Tue Apr 22 13:45:16 2025					
13	338	1.652	Tue Apr 22 13:45:17 2025					

Figure 9.1.2 Serial port Data collection

3. Graphical Representation of Data (Real-Time Plotting)

Functionality:

- A live plot is updated in real-time to display the voltage against time, allowing users to visually track the data as it is collected.
- The graph is generated using matplotlib and embedded into the Tkinter window with FigureCanvasTkAgg.

Key Points:

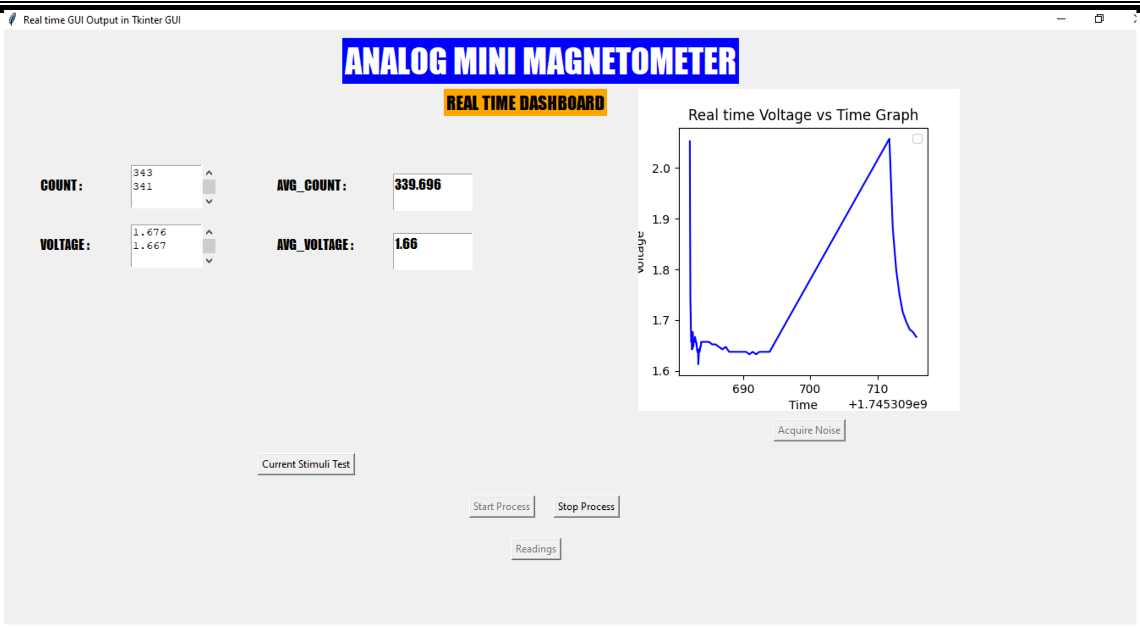
- **Live Graph Updates:** The plot is updated every 100ms using the update() method, allowing users to track changes in the data as they happen.
- **Plot Customization:** The plot includes labels for the x and y axes, and the title is updated dynamically to reflect the real-time nature of the data collection.
- **Graphical Feedback:** The real-time plot provides immediate feedback on the data trends, such as voltage fluctuations over time.

Strengths:

- **Real-Time Visualization:** The live plot offers an intuitive way to visualize the sensor data, making it easier to detect trends or anomalies in the collected data.
- **Customization and Clarity:** The plot is well-labeled, making it easy to interpret the data being visualized.

Improvements:

- **Interactive Plot Features:** Adding zooming, panning, or tooltips for better data exploration would enhance the plot's functionality.
- **Plot Exporting:** Allow users to save or export the plot as an image (e.g., PNG or SVG) for inclusion in reports or presentations.



COUNT: 343 341

AVG_COUNT: 339.696

VOLTAGE: 1.676 1.667

AVG_VOLTAGE: 1.66

Figure 9.1.3 Real-Time Plotting

4. Excel Data Export and Recording

Functionality:

- The program saves the collected data, including count and voltage values, into an Excel file (Real_time_data.xlsx), updating it in real-time.
- The averages for voltage and count are calculated and written into the Excel sheet once the data collection stops.

Key Points:

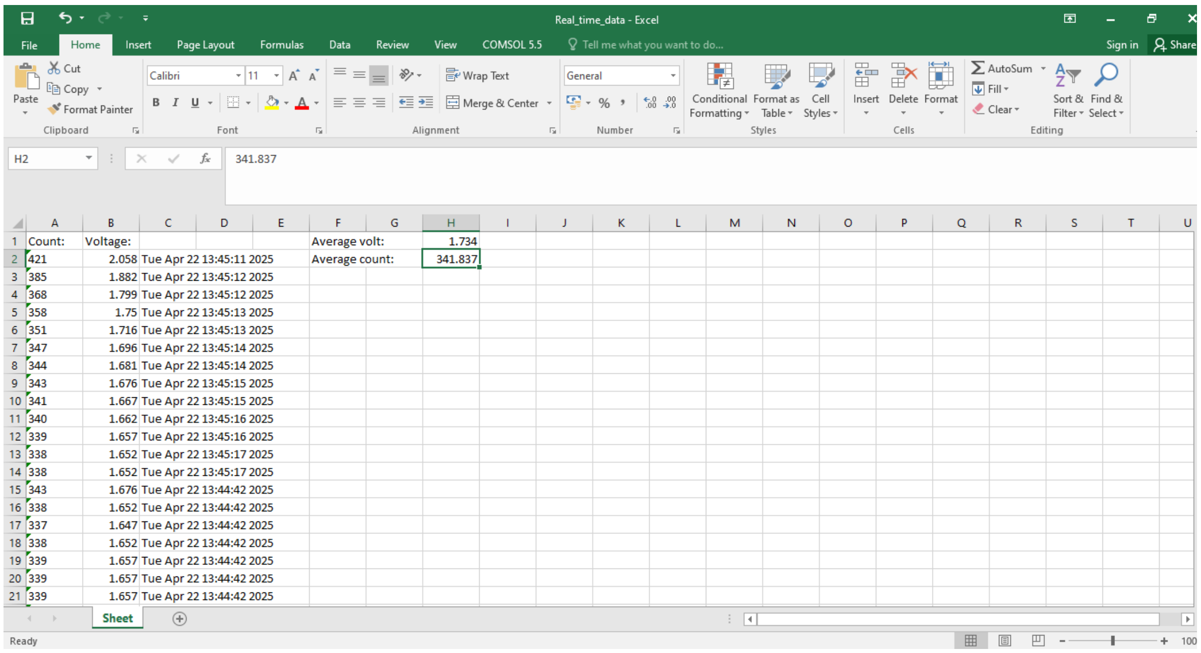
- **Data Recording:** Each sensor reading is saved in a new row in the Excel file, making it easy to store and review data after the collection period.
- **Averages Calculation:** Once data collection stops, the program calculates the average voltage and count and writes these values into the appropriate Excel cells.
- **Excel Integration:** The program uses the openpyxl library to interact with Excel, providing a reliable way to manage and save data.

Strengths:

- **Automatic Data Logging:** Data is logged continuously, ensuring no data is lost during the collection process.
- **Structured Output:** The use of Excel to store data provides a structured format that can be easily analyzed later.

Improvements:

- **Excel Error Handling:** The program could benefit from additional error handling for situations where the Excel file cannot be accessed or written to.
- **More Advanced Reporting:** The Excel output could be improved by including additional summaries, charts, or statistical analyses of the data.



Real_time_data - Excel

H2 341.837

	A	B	C	D	E	F	G	H
1	Count:	Voltage:				Average volt:		1.734
2	421	2.058	Tue Apr 22 13:45:11 2025			Average count:		341.837
3	385	1.882	Tue Apr 22 13:45:12 2025					
4	368	1.799	Tue Apr 22 13:45:12 2025					
5	358	1.75	Tue Apr 22 13:45:13 2025					
6	351	1.716	Tue Apr 22 13:45:13 2025					
7	347	1.696	Tue Apr 22 13:45:14 2025					
8	344	1.681	Tue Apr 22 13:45:14 2025					
9	343	1.676	Tue Apr 22 13:45:15 2025					
10	341	1.667	Tue Apr 22 13:45:15 2025					
11	340	1.662	Tue Apr 22 13:45:16 2025					
12	339	1.657	Tue Apr 22 13:45:16 2025					
13	338	1.652	Tue Apr 22 13:45:17 2025					
14	338	1.652	Tue Apr 22 13:45:17 2025					
15	343	1.676	Tue Apr 22 13:44:42 2025					
16	338	1.652	Tue Apr 22 13:44:42 2025					
17	337	1.647	Tue Apr 22 13:44:42 2025					
18	338	1.652	Tue Apr 22 13:44:42 2025					
19	339	1.657	Tue Apr 22 13:44:42 2025					
20	339	1.657	Tue Apr 22 13:44:42 2025					
21	339	1.657	Tue Apr 22 13:44:42 2025					

Sheet

Ready

Figure 9.1.4 Excel Data Export and Recording

5. Button Functionality (Control and Testing)

Functionality:

- The GUI provides multiple buttons for controlling the data collection process, including starting and stopping the collection, viewing readings, acquiring noise data, and performing current stimuli tests.

Key Points:

- **Start and Stop Controls:** The user can start or stop the data collection, with the appropriate buttons being enabled or disabled based on the current state of the program.
- **Noise and Stimuli Tests:** Users can initiate tests such as noise accumulation or current stimuli, which trigger corresponding actions (e.g., running another Python script).

Strengths:

- **Clear Control Structure:** The buttons provide clear control over the program, allowing users to start, stop, and initiate tests with a single click.
- **Interactive Testing:** The inclusion of testing buttons (noise and stimuli) allows for interactive experiments, providing a flexible tool for different scenarios.

Improvements:

- **Button State Management:** Improve button state management to prevent users from clicking on disabled buttons by accident, which could cause errors.
- **Enhanced Feedback:** Add visual feedback or progress indicators to show when actions (e.g., noise acquisition or stimuli tests) are in progress.

NOISE_ACCUMULATION.PY

1. Real-Time Noise Accumulation with Zoomable Plot

Functionality:

- The application collects real-time noise data from a sensor connected via a serial port and visualizes it on a live plot. The plot updates continuously, displaying the voltage data over time.
- Users can zoom in and out on the plot using the mouse scroll wheel for a more detailed view of specific data points.
- The program includes a stop button that allows the user to close the application.

Key Points:

- **Data Collection:** Real-time data is read from the serial port (COM6) using the serial library. Each reading represents the noise value from the sensor, converted into voltage and plotted against time.
- **Live Plotting:** The plot is updated in real time using matplotlib, and the data points are dynamically appended to the x_data and y_data lists.
- **Zooming:** Users can zoom in and out of the plot using the scroll wheel. This functionality is implemented by adjusting the x and y axis limits based on the mouse cursor's position, providing an interactive way to inspect the data.

Strengths:

- **Real-Time Data Visualization:** The live plotting feature provides immediate feedback to users, showing how noise values change over time.
- **Interactive Zooming:** The zoom functionality allows users to explore the data in greater detail, making it easier to identify trends or anomalies.
- **Efficient Data Collection:** The program collects and processes sensor data efficiently using threading, ensuring the GUI remains responsive during continuous data collection.

Improvements:

- **Error Handling:** The program could benefit from additional error handling for scenarios where the sensor data is corrupted, or if there are issues with the serial communication.
- **GUI Enhancements:** The user interface could be improved by adding more informative labels, tooltips, or status messages, making it more intuitive for users.
- **Data Export:** Allowing users to export the collected data to a file (such as a CSV or Excel file) could be useful for further analysis or reporting.

2. Serial Data Collection and Processing

Functionality:

- The program reads data from a sensor connected via the serial port (COM6) and converts the raw sensor count values into voltage values using a scaling factor.
- The data is then appended to x_data (representing time) and y_data (representing the voltage values) for plotting.
- The read_serial_data() function runs continuously in a separate thread to handle real-time data collection without blocking the main GUI thread.

Key Points:

- **Data Conversion:** Raw count values are converted into voltage using the formula voltage = (count * 5.0 / 1023.0), which scales the sensor data to a voltage range of 0 to 5V.
- **Threading for Efficiency:** Data collection occurs in a background thread (read_serial_data()) to prevent blocking the main GUI thread and ensure the interface remains responsive.
- **Continuous Data Collection:** The data collection runs in an infinite loop, ensuring that new data is continuously appended to the plot as it is received from the sensor.

Strengths:

- **Seamless Data Collection:** The use of threading ensures that the program can collect data continuously without freezing the GUI.
- **Accurate Data Processing:** The conversion from sensor counts to voltage is handled correctly, providing meaningful data for visualization.
- **Real-Time Plot Updates:** The live updates ensure that users can view the latest data as soon as it's available.

Improvements:

- **Serial Communication Robustness:** Implement error handling for issues with the serial connection, such as disconnection or timeout.
- **Data Integrity Checks:** Add validation for the incoming data to ensure it is correctly formatted and within expected ranges.

3. Zooming and User Interaction

Functionality:

- The zoom functionality allows users to zoom in and out of the plot by scrolling with the mouse wheel. The zoom is centered around the mouse cursor's current position on the plot, providing an intuitive and interactive way to explore the data in more detail.
- The zoom functionality is implemented using the scroll_event in matplotlib, which updates the x and y axis limits based on the zoom direction (in or out).

Key Points:

- **Zoom Centered Around Cursor:** The zoom is not a uniform scaling of the plot; instead, it focuses on the area around the cursor. This feature allows users to zoom into specific regions of the plot.
- **Interactive Plot:** Users can interact with the plot by zooming in on specific areas to inspect the data in greater detail, which is particularly useful when analyzing noise data or looking for patterns in the readings.

Strengths:

- **Intuitive User Interaction:** The zoom functionality provides a natural way for users to explore the data. By using the mouse scroll wheel, the plot can be zoomed in and out with ease.
- **Focus on Data Regions of Interest:** Centering the zoom on the mouse cursor allows users to zoom in on specific data points or trends, improving data exploration.

Improvements:

- **Zoom Limits:** It might be helpful to set a limit on how far the user can zoom in or out to prevent the plot from becoming too cluttered or too sparse.
- **Additional Zoom Controls:** Adding buttons or keyboard shortcuts for zooming could provide alternative methods of zooming for users who prefer not to use the mouse wheel.

4. Graphical Representation of Data (Real-Time Plotting)

Functionality:

The real-time data is visualized using a line plot in matplotlib, with the x-axis representing time and the y-axis representing the voltage values. As new data is collected, the plot is updated dynamically to reflect the latest data points.

Key Points:

- **Dynamic Plot Updates:** The plot is updated continuously by modifying the x_data and y_data lists and adjusting the plot's data series accordingly.
- **Real-Time Feedback:** As the sensor data is received, the plot updates in real-time, providing users with immediate feedback on the noise levels.

Strengths:

- **Clear Data Representation:** The plot provides a clear visual representation of the sensor data over time, allowing users to easily track changes in voltage.
- **Live Updates:** The plot updates in real-time, ensuring that users can monitor data as it is collected without delay.

Improvements:

- **Plot Customization:** The plot could be enhanced with additional features, such as different colors or line styles for various data series or the inclusion of grid lines for better readability.

- **Interactive Features:** Additional interactive plot features, such as tooltips or data point highlighting, could improve user experience and make data analysis easier.

5. Application Control (Stop Button)

Functionality:

- The application includes a stop button that allows users to close the program and stop data collection. The stop_button_clicked() function is responsible for closing the serial connection and destroying the Tkinter window.

Key Points:

- **Stop Button for Control:** The stop button provides a simple and effective way for users to stop the data collection and exit the application.
- **Resource Cleanup:** When the stop button is clicked, the serial connection is closed, and the Tkinter window is destroyed, ensuring that resources are properly released.

Strengths:

- **Simple Control Mechanism:** The stop button provides a clear and easy way to terminate the data collection and exit the program.
- **Resource Management:** Closing the serial connection and cleaning up the GUI resources helps prevent memory leaks and other issues.

Improvements:

- **Confirmation Dialog:** Adding a confirmation dialog before closing the application could prevent users from accidentally exiting the program.
- **Graceful Shutdown:** The program could benefit from additional error handling to ensure that the serial connection and other resources are always properly released, even in the case of an unexpected error.

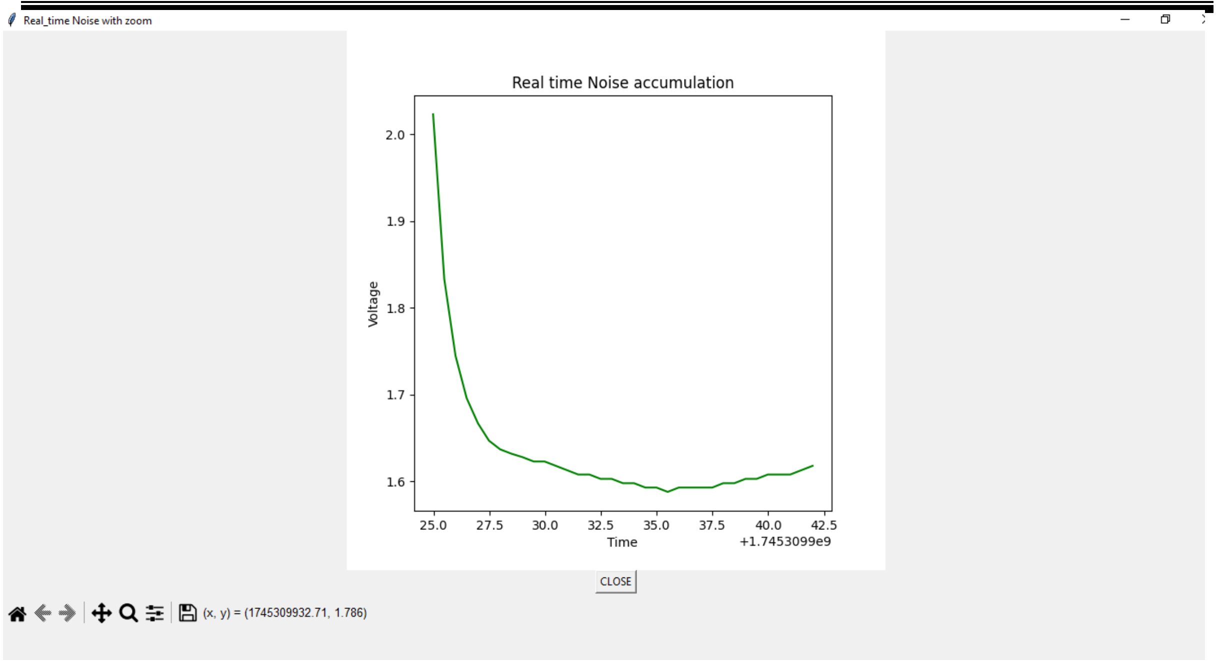

Real_time Noise with zoom
Real time Noise accumulation
Voltage
2.0
1.9
1.8
1.7
1.6
25.0
27.5
30.0
32.5
35.0
37.5
40.0
42.5
Time
+1.7453099e9
CLOSE
(x, y) = (1745309932.71, 1.786)

Figure 9.2.1 Real-Time Noise Accumulation with Zoomable Plot

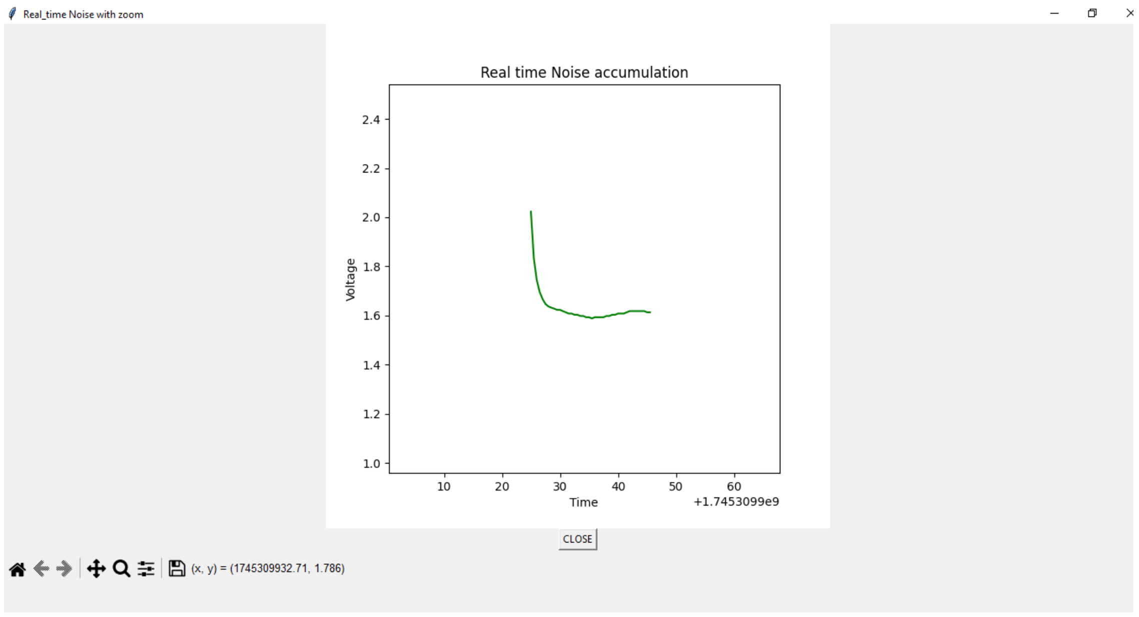

Real_time Noise with zoom
Real time Noise accumulation
Voltage
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
10
20
30
40
50
60
Time
+1.7453099e9
CLOSE
(x, y) = (1745309932.71, 1.786)

Figure 9.2.2 Real-Time Noise Accumulation Zoomed out

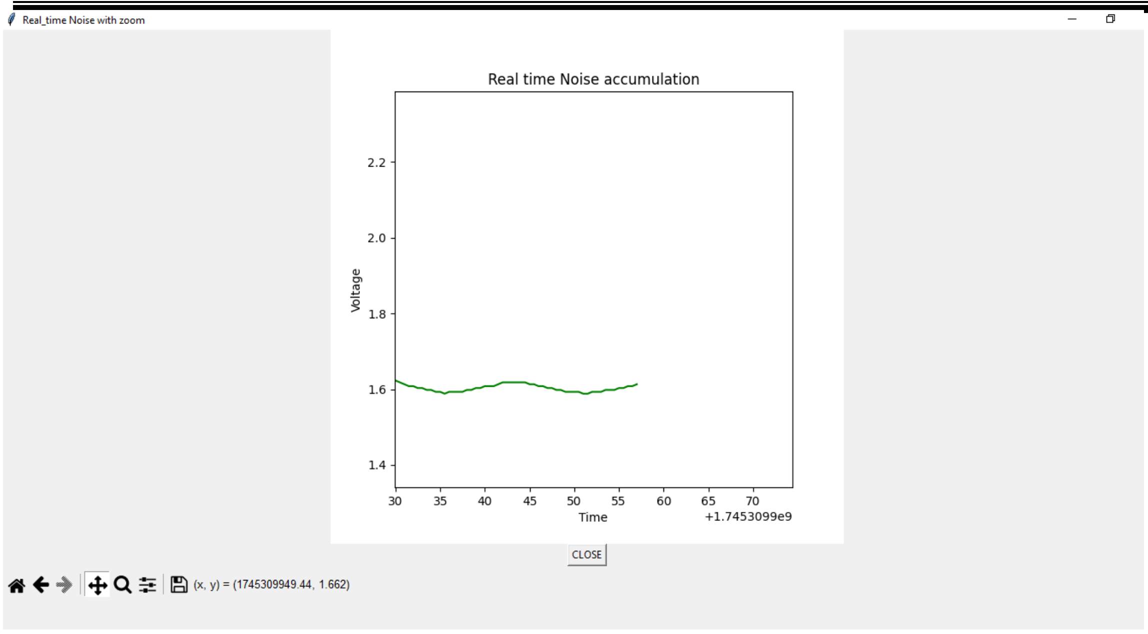

Real_time Noise with zoom
Real time Noise accumulation
Voltage
Time
+1.7453099e9
CLOSE
(x, y) = (1745309949.44, 1.662)

Figure 9.2.3 Real-Time Noise Accumulation Zoomed in

CURRENT_STIMULI_TEST .PY

1. **Real-Time Current Stimuli Collection with GUI Interface**

Functionality:

- The application collects real-time voltage data from a sensor for various current stimuli, ranging from -4 mA to +4 mA.
- Users can select a specific stimulus level through the GUI, and the system begins recording corresponding voltage values immediately.
- Once the stop button is clicked, it calculates the average voltages, differences from baseline (0 mA), and symmetrical differences between opposing current levels.
- The results are saved in an Excel file for further analysis.

Key Points:

- Stimuli levels are selected using dedicated buttons.
- Each level has a separate display for its average value.
- Data is stored and structured in an Excel file, including average, difference from 0 mA, and alternate difference values.

Strengths:

- The GUI layout simplifies user interaction, guiding the user step-by-step through testing.
- Data collection and processing are tightly integrated, reducing the need for manual intervention.
- Excel output allows immediate review and long-term storage.

Improvements:

- A live plot during data collection would help users visualize voltage changes in real time.
- GUI feedback during data collection could be improved with visual indicators like a loading bar or status color.
- The system could allow users to name or select the save location for the Excel output.

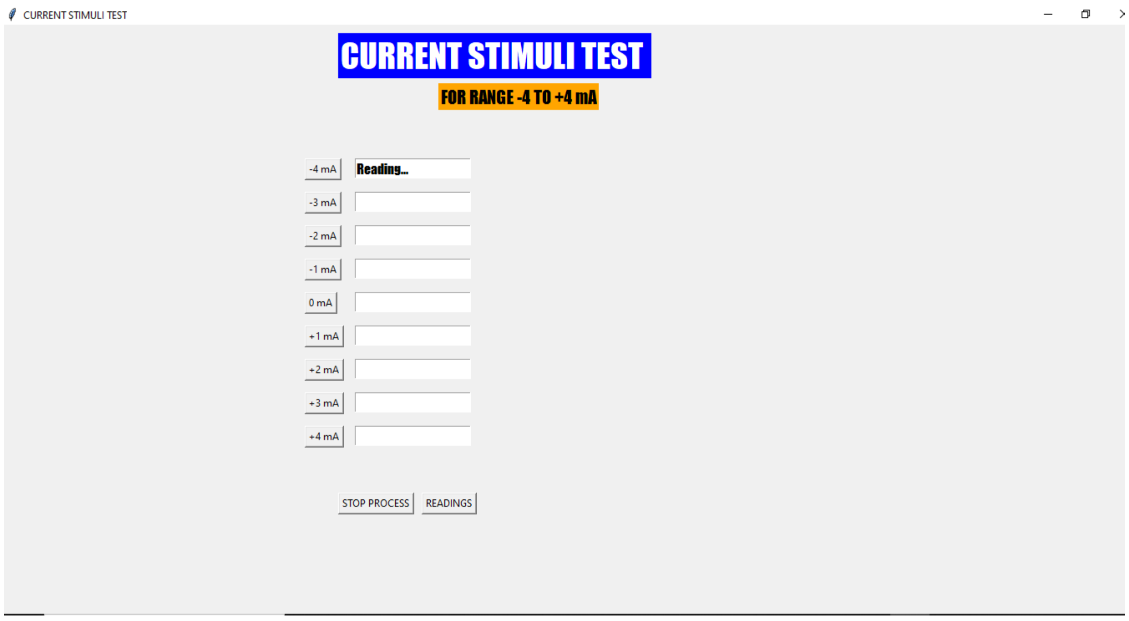


Figure 9.3.1 Current stimuli GUI

2. Serial Data Collection and Processing

Functionality:

- Voltage readings are acquired via a serial port (COM6) and converted from raw digital values into voltage.
- Depending on the selected current level, the values are stored in corresponding arrays and written to an Excel file.
- Data collection is handled in a separate thread to keep the GUI responsive.

Key Points:

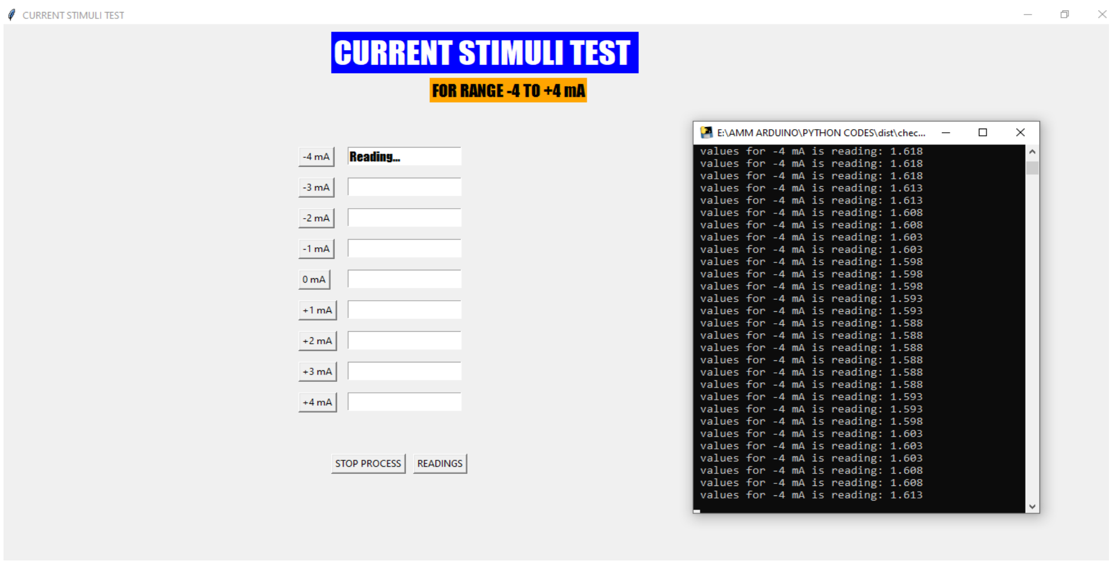
- Each current level has its own buffer for collecting values.
- Conversion from digital count to voltage is handled in real time.
- Excel updates happen continuously during data acquisition.

Strengths:

- Efficient use of threading prevents the interface from freezing.
- Collected data is directly written to Excel in an organized format.
- The architecture supports clean and systematic testing for each current level.

Improvements:

- Better error handling could help detect serial communication issues or unexpected data formats.
- Time stamping the readings would add more context for future analysis.
- Data validation could ensure noise or erroneous values are filtered out before storage.



```
values for -4 mA is reading: 1.618
values for -4 mA is reading: 1.618
values for -4 mA is reading: 1.618
values for -4 mA is reading: 1.613
values for -4 mA is reading: 1.613
values for -4 mA is reading: 1.608
values for -4 mA is reading: 1.608
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.598
values for -4 mA is reading: 1.598
values for -4 mA is reading: 1.598
values for -4 mA is reading: 1.593
values for -4 mA is reading: 1.593
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.593
values for -4 mA is reading: 1.593
values for -4 mA is reading: 1.598
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.608
values for -4 mA is reading: 1.608
values for -4 mA is reading: 1.613
```

Figure 9.3.2 Current stimuli Serial Data Processing

3. GUI Interaction and Stimuli Navigation

Functionality:

- The GUI is designed with buttons for each stimulus level from -4 to +4 mA.
- When a button is pressed, data collection starts automatically for that level.

- Average voltage readings are displayed in real-time inside text boxes adjacent to each button.

Key Points:

- Current selection is visually reinforced by displaying "Reading..." in the respective field.
- Only one current level can be tested at a time, minimizing confusion.

Strengths:

- The layout is straightforward and well-structured.
- The design encourages methodical testing by isolating each current step.
- Users can clearly see which level is active and monitor feedback from the sensor.

Improvements:

- Button highlighting or color changes during active reading would enhance clarity.
- A message log area showing past actions would help users keep track of their testing steps.
- Disabling other buttons during active collection can prevent accidental inputs.

4. Excel-Based Output and Calculations

Functionality:

- Voltage readings and calculated values are automatically stored in an Excel file.
- The Excel file includes raw values, average voltages for each stimulus, differences from the baseline (0 mA), and symmetrical comparisons between negative and positive currents.

Key Points:

- Average values are stored in a designated row for easy identification.
- A separate row is used for showing the difference from 0 mA.
- Another row is used to compare symmetrical stimulus pairs (e.g., -1 mA vs +1 mA).

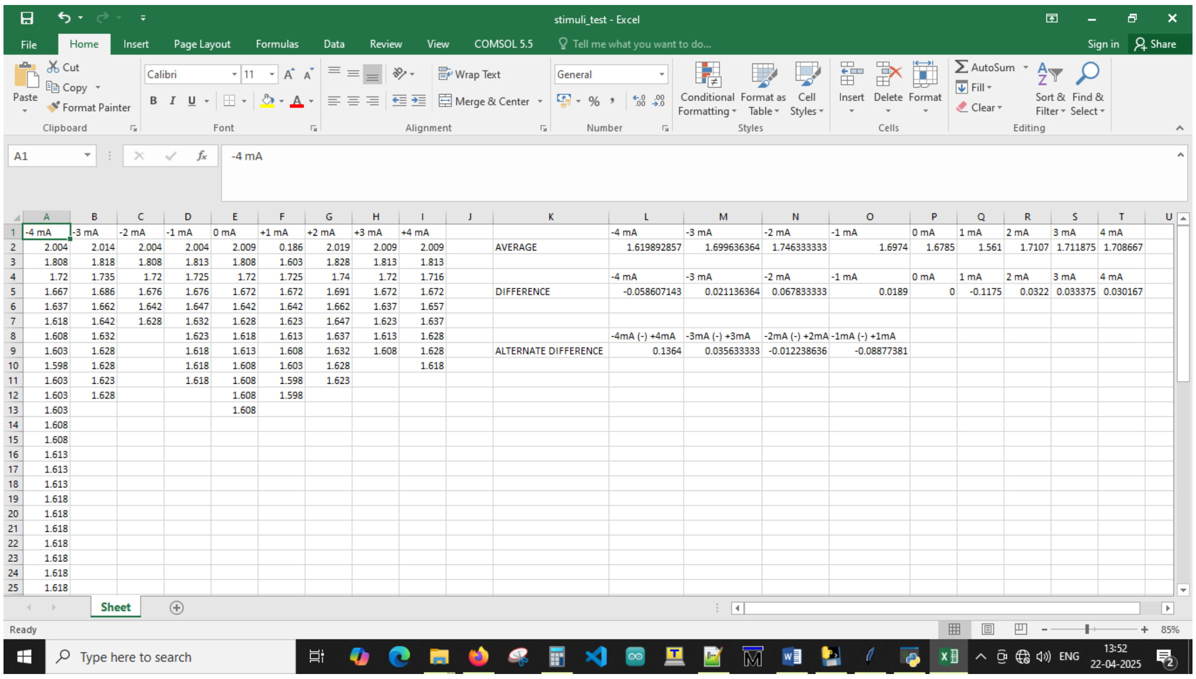
Strengths:

- The structured Excel layout supports easy interpretation of data.
- Automatic calculations save time and reduce errors from manual computation.

- The file is updated in real time, so no data is lost even if the session ends unexpectedly.

Improvements:

- Including charts or graphs in the Excel output would help visualize the trends.
- Allowing users to select custom file paths or formats could make the tool more flexible.
- File versioning (timestamp-based filenames) could help track multiple sessions.



A1 -4 mA

	A	B	C	D	E	F	G	H	I
1	-4 mA	-3 mA	-2 mA	-1 mA	0 mA	+1 mA	+2 mA	+3 mA	+4 mA
2	2.004	2.014	2.004	2.004	2.009	0.186	2.019	2.009	2.009
3	1.808	1.818	1.808	1.813	1.808	1.603	1.828	1.813	1.813
4	1.72	1.735	1.72	1.725	1.72	1.725	1.74	1.72	1.716
5	1.667	1.686	1.676	1.676	1.672	1.672	1.691	1.672	1.672
6	1.637	1.662	1.642	1.647	1.642	1.642	1.662	1.637	1.657
7	1.618	1.642	1.628	1.632	1.628	1.623	1.647	1.623	1.637
8	1.608	1.632		1.623	1.618	1.613	1.637	1.613	1.628
9	1.603	1.628		1.618	1.613	1.608	1.632	1.608	1.628
10	1.598	1.628		1.618	1.608	1.603	1.628		1.618
11	1.603	1.623		1.618	1.608	1.598	1.623		
12	1.603	1.628			1.608	1.598			
13	1.603				1.608				
14	1.608								
15	1.608								
16	1.613								
17	1.613								
18	1.613								
19	1.618								
20	1.618								
21	1.618								
22	1.618								
23	1.618								
24	1.618								
25	1.618								

	K	L	M	N	O	P	Q	R	S	T
1		-4 mA	-3 mA	-2 mA	-1 mA	0 mA	1 mA	2 mA	3 mA	4 mA
2	AVERAGE	1.619892857	1.699636364	1.746333333	1.6974	1.6785	1.561	1.7107	1.711875	1.708667
3										
4		-4 mA	-3 mA	-2 mA	-1 mA	0 mA	1 mA	2 mA	3 mA	4 mA
5	DIFFERENCE	-0.058607143	0.021136364	0.067833333	0.0189	0	-0.1175	0.0322	0.033375	0.030167
6										
7										
8		-4mA (-) +4mA	-3mA (-) +3mA	-2mA (-) +2mA	-1mA (-) +1mA					
9	ALTERNATE DIFFERENCE	0.1364	0.035633333	-0.012238636	-0.08877381					

Figure 9.3.1 Current stimuli Result Excel Sheet

5. Application Control (Stop Button)

Functionality:

- The stop button ends the current test session, triggers calculations, updates the Excel sheet, and closes the serial connection.
- It resets the interface for the next test cycle.

Key Points:

- The stop button also ensures that the data from the last test is finalized and saved.

- The serial connection is properly closed, preventing resource leaks or errors.

Strengths:

- Provides a single, clear way to complete a test and save results.
- Helps maintain system stability by releasing hardware resources.
- Ensures results are not lost even if the session ends prematurely.

Improvements:

- A confirmation popup before stopping could prevent accidental clicks.
- Error handling could be improved to guarantee the stop function executes cleanly in all cases.
- Optionally saving a session log would help users refer back to their actions during the test.

ANALYSIS .PY

1. Data Extraction and User Input

Functionality:

- The program reads temperature-dependent data from an Excel sheet and extracts numerical values for analysis.
- Users input initial slope and intercept values, along with a specific temperature column for baseline comparison.

Key Points:

- Non-numeric data is filtered to maintain clean and accurate datasets.
- The chosen temperature column is validated against the dataset to ensure it exists.

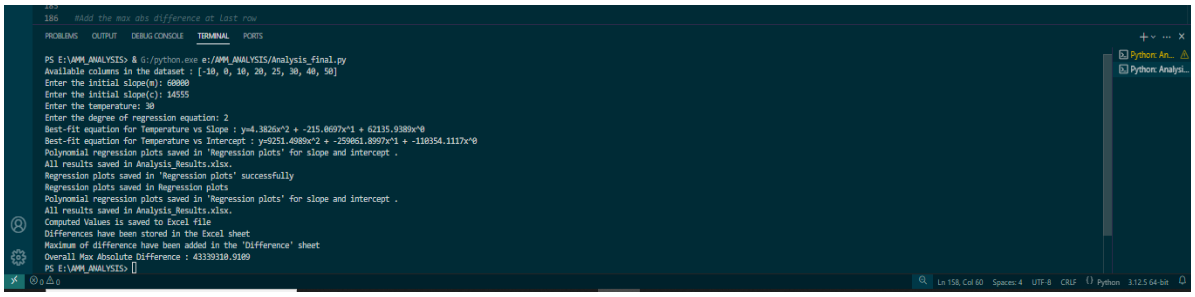
Strengths:

- The input validation ensures data integrity and avoids processing errors.
- Initial user-provided parameters allow for flexible reference selection depending on experiment requirements.

Improvements:

- A dropdown menu for column selection instead of manual input would reduce errors and improve user experience.
- Adding default values or suggestions for slope and intercept could help guide the user

through the setup process.



```
#Add the max abs difference at last row
```

```
PS E:\AMM_ANALYSIS> & G:/python.exe e:/AMM_ANALYSIS/Analysis_final.py
Available columns in the dataset : [-10, 0, 10, 20, 25, 30, 40, 50]
Enter the initial slope(m): 60000
Enter the initial slope(c): 14555
Enter the temperature: 30
Enter the degree of regression equation: 2
Best-fit equation for Temperature vs Slope : y=4.3826x^2 + -215.0697x^1 + 62135.9389x^0
Best-fit equation for Temperature vs Intercept : y=9251.4989x^2 + -259061.8997x^1 + -110354.1117x^0
Polynomial regression plots saved in 'Regression plots' for slope and intercept .
All results saved in Analysis_Results.xlsx.
Regression plots saved in 'Regression plots' successfully
Regression plots saved in Regression plots
Polynomial regression plots saved in 'Regression plots' for slope and intercept .
All results saved in Analysis_Results.xlsx.
Computed Values is saved to Excel file
Differences have been stored in the Excel sheet
Maximum of difference have been added in the 'Difference' sheet
Overall Max Absolute Difference : 43339310.9109
PS E:\AMM_ANALYSIS> 
```

Figure 9.4.1 Data collection and user input

2. Linear Regression and Visualization

Functionality:

- A linear regression is performed using the reference temperature column and user-provided slope and intercept.
- The resulting regression is plotted and saved in a dedicated folder.

Key Points:

- Regression lines are plotted for both the reference and all other temperature columns.
- Each plot includes data points and a best-fit line, along with titles and axis labels.

Strengths:

- Visualizing the data helps validate the fit of the regression model across all temperatures.
- Each plot is saved as an image, allowing easy documentation and presentation.

Improvements:

- Displaying plots within the application interface would allow real-time feedback.
- Including a summary of residuals or goodness-of-fit values on the plots would enhance interpretability.

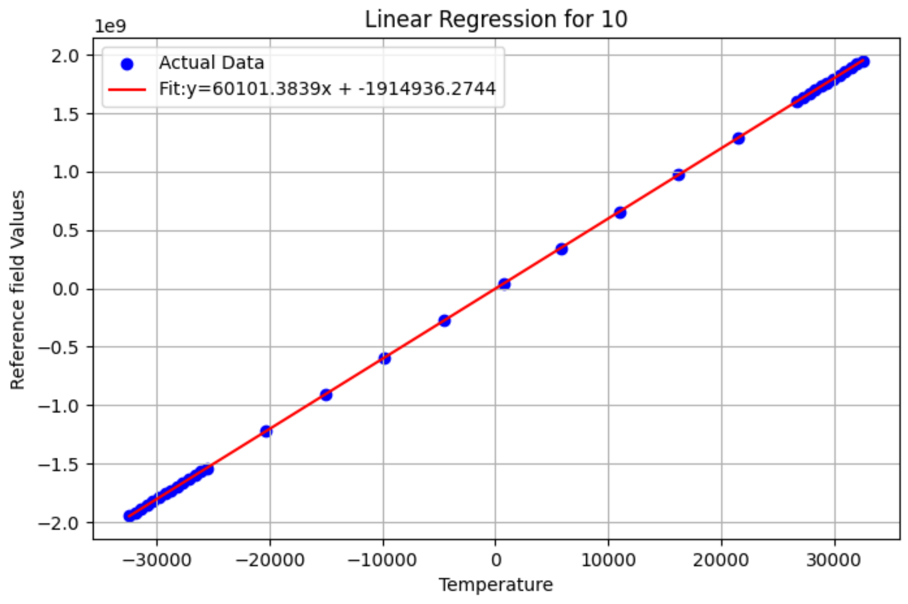


Figure 9.4.2 Linear Regression

3. Final Slope and Intercept Calculation

Functionality:

- For each temperature column, the program calculates the best-fit slope and intercept using the reference as a baseline.
- These values are stored and later used for polynomial regression analysis.

Key Points:

- Slope and intercept values are compiled and sorted based on temperature.
- All results are written into an organized Excel sheet for further use.

Strengths:

- The structured output provides a clear mapping between temperature and regression parameters.
- Automatically sorting the data improves readability and consistency.

Improvements:

- Adding statistical confidence intervals or standard deviations would provide more insights into the stability of the regression.
- Including flags for poor fits could alert users to questionable data points.

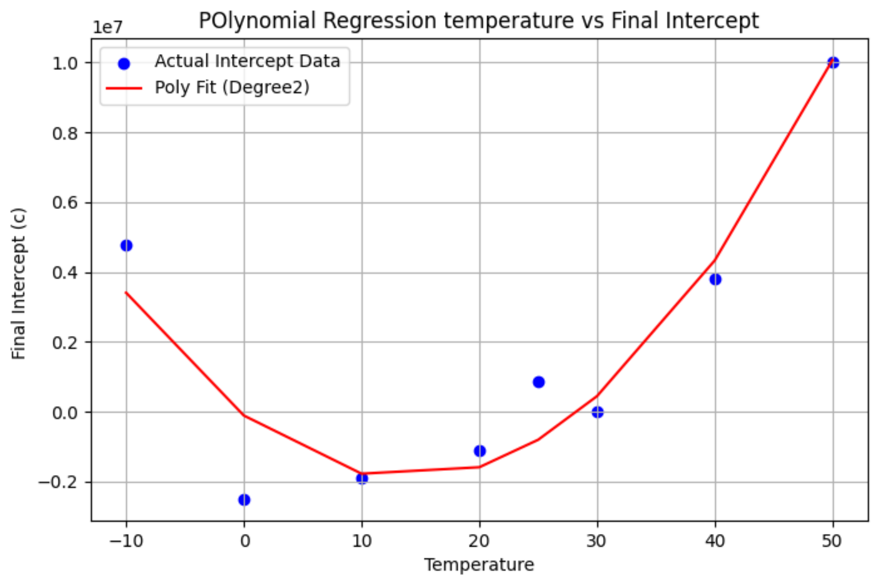


Figure 9.4.3.1 Final Intercept

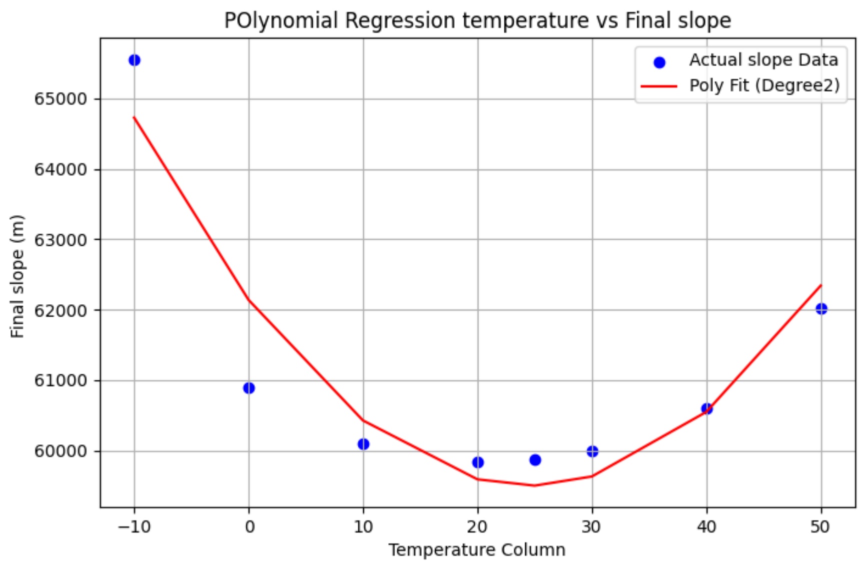


Figure 9.4.3.2 Final Slope

4. Polynomial Regression Analysis

Functionality:

- Polynomial regression is used to model how slope and intercept values vary with temperature.
- Users input the desired degree of the polynomial for more flexibility in fitting.

Key Points:

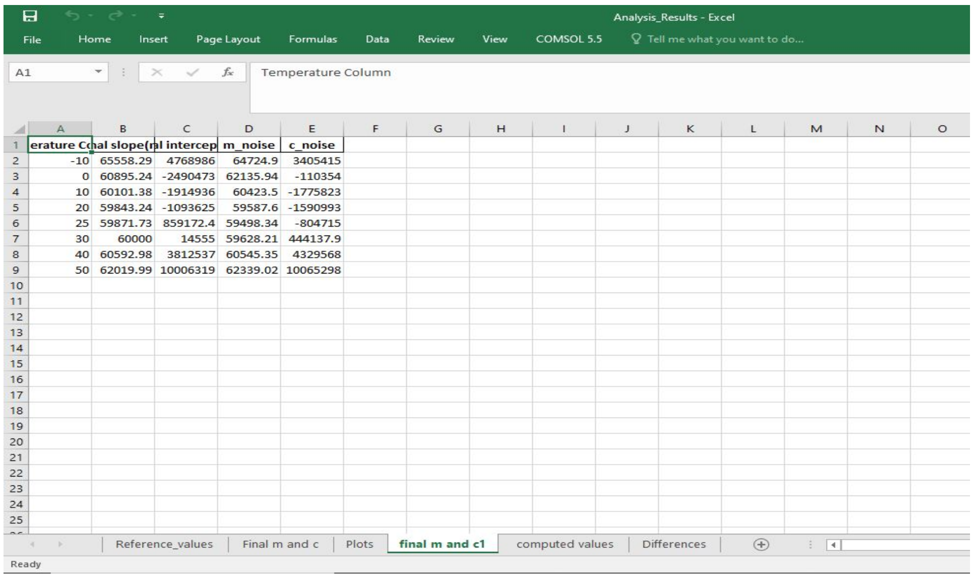
- Best-fit equations for both slope and intercept are printed and plotted.
- The equations are also stored in the Excel file along with updated m and c values across the temperature range.

Strengths:

- Polynomial regression helps capture the non-linear behaviour of regression parameters with temperature.
- Users have control over the model complexity through the polynomial degree input.

Improvements:

- Visualizing residuals or errors for the polynomial fits would help evaluate overfitting or underfitting.
- Automatically suggesting a best-fit degree based on error metrics would guide less experienced users.



Temperature Column

	A	B	C	D	E
1	erature Co	al slope(n	al intercep	m_noise	c_noise
2	-10	65558.29	4768986	64724.9	3405415
3	0	60895.24	-2490473	62135.94	-110354
4	10	60101.38	-1914936	60423.5	-1775823
5	20	59843.24	-1093625	59587.6	-1590993
6	25	59871.73	859172.4	59498.34	-804715
7	30	60000	14555	59628.21	444137.9
8	40	60592.98	3812537	60545.35	4329568
9	50	62019.99	10006319	62339.02	10065298

Figure 9.4.4 Polynomial Regression Analysis Excel Sheet

5. Computed Values and Difference Analysis

Functionality:

- The script uses the polynomial regression models to compute expected Y-values for each temperature dataset.
- It then calculates the differences between computed and reference values, identifying

the maximum absolute error.

Key Points:

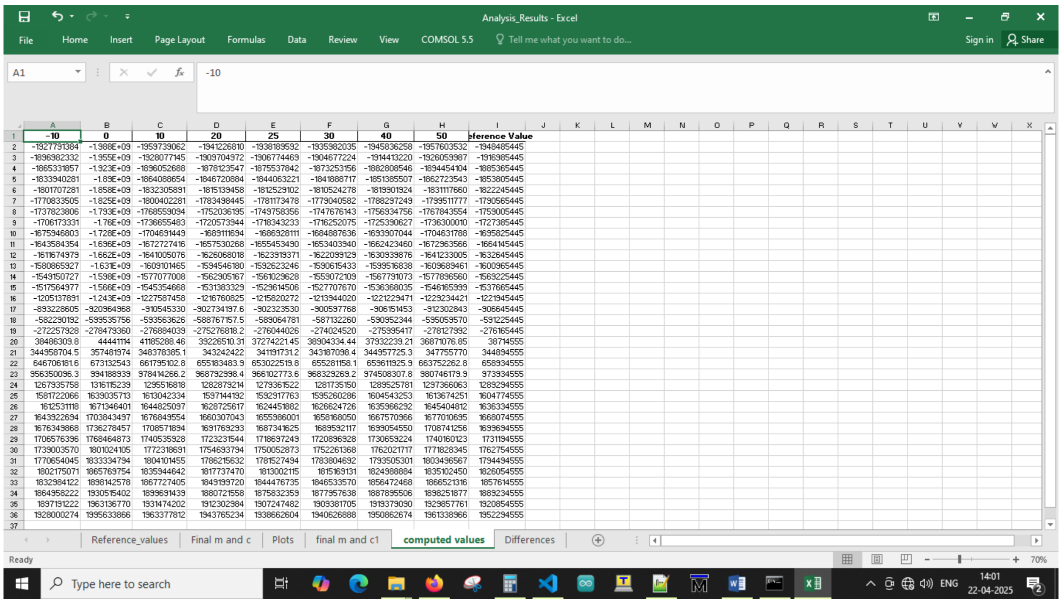
- Computed values and differences are written into separate sheets in the Excel file.
- The maximum difference per column and overall maximum difference are clearly reported.

Strengths:

- This analysis helps validate how well the model generalizes to different temperature datasets.
- Including the overall maximum difference allows users to assess the model's worst-case performance.

Improvements:

- Adding color-coded heatmaps for difference values in the Excel file could make discrepancies more visible.
- Including percentage differences or normalized error values would enhance interpretability.



	A	B	C	D	E	F	G	H	I
1	-10	0	10	20	25	30	40	50	eference Value
2	-1927791384	-1.988E+09	-1959739062	-1941226810	-1938189592	-1935982035	-1945836258	-1957603532	-1948485445
3	-1896982332	-1.955E+09	-1928077145	-1909704972	-1906774469	-1904677224	-1914413220	-1926059987	-1916985445
4	-1865331857	-1.923E+09	-1896052688	-1878123547	-1875537842	-1873253156	-1882808546	-1894454104	-1885365445
5	-1833940281	-1.89E+09	-1864088654	-1846720884	-1844063221	-1841888717	-1851385507	-1862723543	-1853805445
6	-1801707281	-1.858E+09	-1832305891	-1815139458	-1812529102	-1810524278	-1819901924	-1831117660	-1822245445
7	-1770833505	-1.825E+09	-1800402281	-1783498445	-1781173478	-1779040582	-1788297249	-1799511777	-1790565445
8	-1737823806	-1.793E+09	-1768559094	-1752036195	-1749758356	-1747676143	-1756934756	-1767843554	-1759005445
9	-1706173331	-1.76E+09	-1736655483	-1720573944	-1718343233	-1716252075	-1725390627	-1736300010	-1727385445
10	-1675946803	-1.728E+09	-1704691449	-1689111694	-1686928111	-1684887636	-1693907044	-1704631788	-1695825445
11	-1643584354	-1.696E+09	-1672727416	-1657530268	-1655453490	-1653403940	-1662423460	-1672963566	-1664145445
12	-1611674979	-1.662E+09	-1641005076	-1626068018	-1623919371	-1622099129	-1630939876	-1641233005	-1632645445
13	-1580865927	-1.631E+09	-1609101465	-1594546180	-1592623246	-1590615433	-1599516838	-1609689461	-1600965445
14	-1549150727	-1.598E+09	-1577077008	-1562905167	-1561029628	-1559072109	-1567791073	-1577896560	-1569225445
15	-1517564977	-1.566E+09	-1545354668	-1531383329	-1529614506	-1527707670	-1536368035	-1546165999	-1537665445
16	-1205137891	-1.243E+09	-1227587458	-1216760825	-1215820272	-1213944020	-1221229471	-1229234421	-1221945445
17	-893228605	-920964968	-910545330	-902734197.6	-902323530	-900597768	-906151453	-912302843	-906645445
18	-582290192	-599535756	-593563626	-588767157.5	-589064781	-587132260	-590952344	-595059570	-591225445
19	-272257928	-278479360	-276884039	-275276818.2	-276044026	-274024520	-275995417	-278127992	-276165445
20	38486309.8	44441114	41185288.46	39226510.31	37274221.45	38904334.44	37932239.21	36871076.85	38714555
21	344958704.5	357481974	348378385.1	343242422	341191731.2	343187098.4	344957725.3	347755770	344894555
22	646706181.6	673132543	661795102.8	655183483.9	653022519.8	655281158.1	659611925.9	663752262.8	658934555
23	956350096.3	994188939	978414266.2	968792998.4	966102773.6	968329269.2	974508307.8	980746179.9	973934555
24	1267935758	1316115239	1295516818	1282879214	1279361522	1281735150	1289525781	1297366063	1289294555
25	1581722066	1639035713	1613042334	1597144192	1592917763	1595260286	1604543253	1613674251	1604774555
26	1612531118	1671346401	1644825097	1628725617	1624451882	1626624726	1635966292	1645404812	1636334555
27	1643922694	1703843497	1676849554	1660307043	1655986001	1658168050	1667570966	1677010695	1668074555
28	1676349868	1736278457	1708571894	1691769293	1687341625	1689592117	1699054550	1708741256	1699694555
29	1706576396	1768464873	1740535928	1723231544	1718697249	1720896928	1730659224	1740160123	1731194555
30	1739003570	1801024105	1772318691	1754693794	1750052873	1752261368	1762021717	1771828345	1762754555
31	1770654045	1833334794	1804101455	1786215632	1781527494	1783804692	1793505301	1803496567	1794494555
32	1802175071	1865769754	1835944642	1817737470	1813002115	1815169131	1824988884	1835102450	1826054555
33	1832984122	1898142578	1867727405	1849199720	1844476735	1846533570	1856472468	1866521316	1857614555
34	1864958222	1930515402	1899691439	1880721558	1875832359	1877957638	1887895506	1898251877	1889234555
35	1897191222	1963136770	1931474202	1912302984	1907247482	1909381705	1919379090	1929857761	1920854555
36	1928000274	1995633866	1963377812	1943765234	1938662604	1940626888	1950862674	1961338966	1952294555

Figure 9.4.5 Computed Values Excel Sheet

6. Output and File Management

Functionality:

- All results including plots, computed values, differences, and regression parameters are saved in an Excel file and image directory.

Key Points:

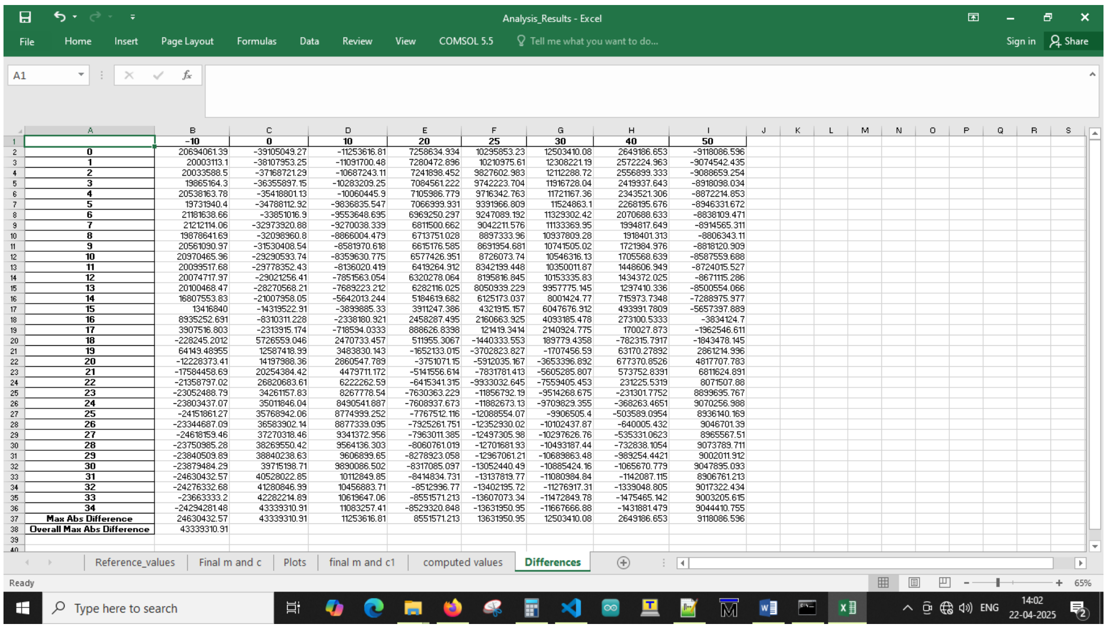
- The file structure includes separate sheets for reference values, computed data, regression plots, and difference metrics.
- Regression plots are stored in a dedicated folder, simplifying access and organization.

Strengths:

- Output files are well-organized and comprehensive, making further analysis or reporting straightforward.
- Saving plots separately ensures reproducibility and visual traceability of results.

Improvements:

- Allowing users to specify a custom save path or file name would increase flexibility.
- Including a readme or metadata sheet in the Excel file to explain column meanings could assist future users or collaborators.



	-10	0	10	20	25	30	40	50
0	20694061.39	-39105049.27	-11253616.81	7258634.934	10295853.23	12503410.08	2649186.653	-9118086.596
1	20003113.1	-38107953.25	-11091700.48	7280472.896	10210975.61	12308221.19	2572224.963	-9074542.435
2	20033588.5	-37168721.29	-10687243.11	7241898.452	9827602.983	12112288.72	2556899.333	-9088659.254
3	19865164.3	-36355897.15	-10283209.25	7084561.222	9742223.704	11916728.04	2419937.643	-8918098.034
4	20538163.78	-35418801.13	-10060445.9	7105986.779	9716342.763	11721167.36	2343521.306	-8872214.853
5	19731940.4	-34788112.92	-9836835.547	7066999.931	9391966.809	11524863.1	2268195.676	-8946331.672
6	21181638.66	-33851016.9	-9553648.695	6969250.297	9247089.192	11329302.42	2070688.633	-8838109.471
7	21212114.06	-32973920.88	-9270038.339	6811500.662	9042211.576	11133369.95	1994817.649	-8914565.311
8	19878641.69	-32098960.8	-8866004.479	6713751.028	8897333.96	10937809.28	1918401.313	-8806343.11
9	20561090.97	-31530408.54	-8581970.618	6615176.585	8691954.681	10741505.02	1721984.976	-8818120.909
10	20970465.96	-29290593.74	-8359630.775	6577426.951	8726073.74	10546316.13	1705568.639	-8587559.688
11	20099517.68	-29778352.43	-8136020.419	6419264.912	8342199.448	10350011.87	1448606.949	-8724015.527
12	20074717.97	-29021256.41	-7851563.054	6320278.064	8195816.845	10153335.83	1434372.025	-8671115.286
13	20100468.47	-28270568.21	-7689223.212	6282116.025	8050939.229	9957775.145	1297410.336	-8500554.066
14	16807553.83	-21007958.05	-5642013.244	5184619.682	6125173.037	8001424.77	715973.7348	-7288975.977
15	13416840	-14319522.91	-3899885.33	3911247.386	4321915.157	6047676.912	493991.7809	-5657397.889
16	8935252.691	-8310311.228	-2338180.921	2458287.495	2160663.925	4093185.478	273100.5333	-3834124.7
17	3907516.803	-2313915.174	-718594.0333	888626.8398	121419.3414	2140924.775	170027.873	-1962546.611
18	-228245.2012	5726559.046	2470733.457	511955.3067	-1440333.553	189779.4358	-782315.7917	-1843478.145
19	64149.48955	12587418.99	3483830.143	-1652133.015	-3702823.827	-1707456.59	63170.27892	2861214.996
20	-12228373.41	14197988.36	2860547.789	-3751071.15	-5912035.167	-3653396.892	677370.8526	4817707.783
21	-17584458.69	20254384.42	4479711.172	-5141556.614	-7831781.413	-5605285.807	573752.8391	6811624.891
22	-21358797.02	26820683.61	6222262.59	-6415341.315	-9933032.645	-7559405.453	231225.5319	8071507.88
23	-23052488.79	34261157.83	8267778.54	-7630363.229	-11856792.19	-9514268.675	-231301.7752	8899695.767
24	-23803437.07	35011846.04	8490541.887	-7608937.673	-11882673.13	-9709829.355	-368263.4651	9070256.988
25	-24151861.27	35768942.06	8774999.252	-7767512.116	-12088554.07	-9906505.4	-503589.0954	8936140.169
26	-23344687.09	36583902.14	8877339.095	-7925261.751	-12352930.02	-10102437.87	-640005.432	9046701.39
27	-24618159.46	37270318.46	9341372.956	-7963011.385	-12497305.98	-10297626.76	-535331.0623	8965567.51
28	-23750985.28	38269550.42	9564136.303	-8060761.019	-12701681.93	-10493187.44	-732838.1054	9073789.711
29	-23840509.89	38840238.63	9606899.65	-8278923.058	-12967061.21	-10689863.48	-989254.4421	9002011.912
30	-23879484.29	39715198.71	9890086.502	-8317085.097	-13052440.49	-10885424.16	-1065670.779	9047895.093
31	-24630432.57	40528022.85	10112849.85	-8414834.731	-13137819.77	-11080984.84	-1142087.115	8906761.213
32	-24276332.68	41280846.99	10456883.71	-8512996.77	-13402195.72	-11276917.31	-1339048.805	9017322.434
33	-23663333.2	42282214.89	10619647.06	-8551571.213	-13607073.34	-11472849.78	-1475465.142	9003205.615
34	-24294281.48	43339310.91	11083257.41	-8529320.848	-13631950.95	-11667666.88	-1431881.479	9044410.755
Max Abs Difference	24630432.57	43339310.91	11253616.81	8551571.213	13631950.95	12503410.08	2649186.653	9118086.596
Overall Max Abs Difference	43339310.91							

Figure 9.4.6 Final Output Excel Sheet

CHAPTER-10

CONCLUSION

We successfully designed a real-time dashboard and analysis platform for tracking magnetic field data from the Analog Mini Magnetometer (AMM), which is a key sensor for the Gaganyaan mission. Interfacing the AMM with an Arduino and using Python for data acquisition, real-time visualization, and analysis, we were able to show the potential of processing magnetic field measurements with high accuracy.

The system efficiently provides constant monitoring of the satellite's orientation and travel in orbit. Utilizing regression modeling and real-time data analysis, the system can identify minimal changes in the magnetic field for proper satellite orientation and error-free trajectory. It is done using accurate data-based corrections and orbital adjustments based on magnetometer data. The incorporation of noise filtering methods, including moving average filters and regression-based smoothing, increases data integrity and provides reliable performance in the hostile space environment.

Moreover, the real-time dashboard offers mission control instant access to the satellite's magnetic field data, allowing quick decision-making for corrective maneuvers as and when needed. The high-resolution magnetic measurements coupled with automatic data logging ensures that the satellite's path and orientation are maintained at all times with minimal manual intervention.

In summary, this system greatly enhances the efficiency, safety, and reliability of satellite operations by providing accurate control over its trajectory and orientation in space. The real-time interpretation and analysis of magnetometer data are critical for accurate mission navigation and orbital control, ultimately supporting the success of the Gaganyaan mission and further advancing future space exploration missions.

REFERENCES

[1] J. C. Choi, J. W. Lee, “Sensor Data Analysis Using Regression and Machine Learning,” *International Journal of Sensor Data Science*, vol. 8, no. 1, pp. 45–52, 2020.

[2] S. K. Gupta, A. R. Menon, “Signal Processing Techniques for Magnetometer Data in Space Applications,” *IEEE Sensors Journal*, vol. 19, no. 10, pp. 3820–3827, May 2019.

[3] M. J. Brown, L. P. Ramos, “Real-Time Data Acquisition and Analysis Using Python,” *Journal of Data Acquisition and Processing*, vol. 12, no. 4, pp. 203–210, Dec. 2018.

[4] N. D. Patel, A. G. Desai, “Machine Learning Algorithms for Real-Time Sensor Data Processing,” *Journal of Intelligent Systems*, vol. 26, no. 2, pp. 98–107, 2019.

[5] T. Y. Zhang, L. Q. Zhao, “Deep Learning Models for Magnetometer Signal Classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 6, pp. 4150–4158, June 2020.

[6] E. M. Thompson, B. L. George, “Magnetometer-Based Satellite Attitude Estimation,” *Advances in Space Research*, vol. 65, no. 3, pp. 911–918, 2020.

[7] R. V. Menon, K. H. Das, “Low-Cost Magnetometer Integration for CubeSat Missions,” *Acta Astronautica*, vol. 180, pp. 1–8, 2021.

[8] M. R. Iqbal, S. V. Thomas, “Python-Based Real-Time Sensor Monitoring Systems,” *Journal of Sensor Networks and Applications*, vol. 9, no. 3, pp. 221–229, 2019.

[9] H. K. Bose, S. K. Yadav, “Real-Time Embedded Data Systems with Python,” *International Journal of Embedded Systems*, vol. 13, no. 1, pp. 54–62, 2021.

[10] K. J. Walters, L. C. Martin, “Regression-Based Prediction Models in Sensor Networks,” *Journal of Environmental Informatics*, vol. 28, no. 2, pp. 134–141, 2022.

[11] A. R. Vyas, P. J. Mehta, “Noise Filtering Methods in Magnetic Field Sensor Data,” *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–8, 2023.

[12] J. H. Shun, R. K. Prakash, “Multi-Channel Magnetometer Calibration Techniques,” *Review of Scientific Instruments*, vol. 94, no. 1, Article ID 015106, Jan. 2023.

APPENDIX-A

PSEUDOCODE

MAIN GUI:

```
BEGIN PROGRAM

1. IMPORT NECESSARY MODULES
   - Tkinter for GUI
   - Serial for communication
   - Threading for real-time data handling
   - Time for timestamps
   - Openpyxl for Excel operations
   - OS and subprocess for launching other scripts
   - Matplotlib for embedding real-time plots

2. DEFINE CLASS: RealTimeOutputGUI
   FUNCTION: __init__(self, root)
     - Open serial port (COM6, 9600 baud rate)
     - Configure main GUI window (title, size)
     - Create labels for headings and data fields
     - Add scrolled text and text boxes for:
         • Count
         • Average Count
         • Voltage
         • Average Voltage
     - Add GUI buttons:
         • Start Process
         • Stop Process
         • Readings (Excel viewer)
         • Acquire Noise (launch noise plotting script)
         • Current Test (launch another script)
```

```
- Initialize:
        • Lists to store readings (value_array, count_array, time_array)
        • Excel workbook and worksheet
        • Average value placeholders
        • Plot for real-time voltage vs. time using Matplotlib
        • Canvas to display plot inside GUI

3. FUNCTION: update_plot(self)
   - Update x and y data for plot
   - Adjust plot limits and redraw canvas

4. FUNCTION: update(self)
   - Call update_plot() every 100 milliseconds
   - Loop using root.after()

5. FUNCTION: stop_button_clicked(self)
   - Flush and close the canvas
   - Calculate and display average voltage and count
   - Print summary in console (count, average, sum, number of readings)
   - Close serial connection
   - Enable 'Readings' and 'Noise' buttons
   - Update Excel sheet with final average values

6. FUNCTION: start_data_collection(self)
   - Open serial port if closed
   - Clear previous data lists
   - Start update loop and thread for sensor data collection
   - Disable start, readings, and noise buttons

7. FUNCTION: collect_sensor_data(self)
   - WHILE True:
       IF serial data is available:
           - Read and decode the sensor count
```

```
    - Convert count to voltage
        - Update GUI text boxes with count and voltage
        - Append data to lists (x_data, y_data, value_array, count_array)
        - Write count, voltage, and timestamp to Excel sheet
        - Auto-scroll text boxes and update GUI
        - Increment Excel row
        - Save workbook

8. FUNCTION: excel_button_clicked(self)
   - Open "Real_time_data.xlsx" using default app

9. FUNCTION: noise_button_clicked(self)
   - Close serial port
   - Launch external script "noise_accumulation_with_zoom.py"

10. FUNCTION: current_stimuli_button_clicked(self)
    - Close serial port
    - Launch external script "replicate.py"

11. START MAIN PROGRAM
   - Create root Tkinter window
   - Instantiate RealTimeOutputGUI class
   - Run root.mainloop()

END PROGRAM
```

```
NOISE ACCUMULATION
BEGIN PROGRAM
1. INITIALIZATION
   - Import necessary modules:
       • Tkinter for GUI
       • Matplotlib for plotting
       • Serial for communication
       • NumPy and threading for processing
   - Define empty lists for x_data (time) and y_data (voltage)
   - Set SERIAL_PORT to 'COM6' and BAUD_RATE to 9600

2. OPEN SERIAL PORT
   TRY to open serial connection
       IF fails, print error message

3. FUNCTION: create_plot()
   - Create a Matplotlib figure and axis
   - Set plot title: "Real time Noise accumulation"
   - Set x-axis as Time and y-axis as Voltage
   - Initialize an empty line plot for dynamic data
   - RETURN the figure, axis, and line plot object

4. FUNCTION: update_plot(line, ax)
   - Update x and y data of the line plot with global lists
   - Recalculate plot limits and scale view to fit new data
   - Redraw canvas to reflect changes

5. FUNCTION: read_serial_data()
   - WHILE True:
       IF data is available from serial:
           - Read a line, decode and strip it
           - Convert the digital value to voltage
           - Append current time to x_data
```

```
        - Append voltage to y_data
6. FUNCTION: zoom_with_scroll(event, ax)
   - Get mouse coordinates and current axis limits
   - Define zoom factor (10%)
   - IF mouse scroll up:
      • Zoom in centered at cursor
   - ELSE IF mouse scroll down:
      • Zoom out centered at cursor
   - Redraw canvas to apply zoom

7. FUNCTION: stop_button_clicked()
   - Close serial port
   - Destroy the GUI window

8. GUI SETUP
   - Create root window titled "Real_time Noise with zoom"
   - Call create_plot() and embed plot in Tkinter window
   - Add "CLOSE" button to stop data collection and exit
   - Add Matplotlib toolbar for zoom, pan, and reset view

9. EVENT BINDINGS
   - Bind mouse scroll to zoom_with_scroll() on plot

10. START THREAD
    - Start background thread to continuously read serial data

11. FUNCTION: update()
    - Call update_plot() every 100 milliseconds
    - Schedule itself again using root.after()

12. MAIN LOOP
    - Start update loop
    - Run Tkinter mainloop
```

```
END PROGRAM
CURRENT STIMULI TEST:

BEGIN PROGRAM
1. INITIALIZATION
   - Import required libraries (GUI, Serial, Excel, Plotting, Threading)
   - Initialize serial port on COM6 (9600 baud rate)
   - Create GUI window with title "CURRENT STIMULI TEST" and set dimensions

2. EXCEL SETUP
   - Create new Excel workbook and activate worksheet
   - Label columns for current levels from -4 mA to +4 mA in 3 different rows
   - Label rows: AVERAGE, DIFFERENCE, ALTERNATE DIFFERENCE

3. DATA STRUCTURES
   - Create empty lists to store voltage readings for each current level
   - Initialize average variables for each current level
   - Initialize GUI display fields (Text Areas) to show live average values

4. GUI COMPONENTS
   - Add title labels and descriptions
   - Create buttons for current levels (-4 mA to +4 mA)
       -> Each button triggers:
          - Set selected current
          - Reset iterator
          - Display "Reading..." in text field
          - Start data collection thread
   - Create a STOP button to end current session and compute statistics

5. DATA COLLECTION THREAD
   START THREAD
   LOOP while serial port is open:
       IF new data is available:
           - Read line from serial port
```

```
        - Convert digital value to voltage
        - Append to correct current level list
        - Write voltage to Excel sheet
        - Increment iterator
        - Save Excel file
  END THREAD

6. STOP BUTTON FUNCTION
  - Clear all "Reading..." text fields
  - Determine which current level was last selected
  - Calculate average voltage from the list
  - Display average in GUI and save in Excel
  - Calculate difference from 0 mA average and store in Excel
  - Calculate alternate difference between mirrored currents (e.g., -1 vs +1) and store in Excel
  - Save workbook as "stimuli_test.xlsx"
  - Close the serial port

7. ADDITIONAL FUNCTION BUTTONS
  - Excel button: Opens the saved Excel file
  - Noise button: Closes serial port and runs noise analysis script
  - Current Stimuli button: Closes serial port and restarts current GUI script

8. MAIN FUNCTION
  - Create root GUI window
  - Initialize RealTimeOutputGUI class
  - Start GUI mainloop

END PROGRAM
```

ANALYSIS:

```
BEGIN PROGRAM

1. IMPORT REQUIRED LIBRARIES
   - NumPy and Pandas for data handling
   - Matplotlib for plotting
   - OS for directory and file handling

2. READ DATA
   - Load 'data.xlsx' file into a Pandas DataFrame
   - Drop non-numeric and NaN values
   - Ensure only numeric columns are used

3. USER INPUT
   - Get slope (m) and intercept (c) values from user
   - Get column number for reference temperature
   - Validate the selected column

4. COMPUTE REFERENCE VALUES
   - Extract temperature column as X
   - Calculate reference Y values using y = mx + c
   - Store in a new DataFrame (reference_df)

5. SAVE FIRST REGRESSION PLOT
   - Create "Regression plots" directory if not exists
   - Plot scatter and line graph for selected temperature
   - Save the plot image in the folder

6. INITIALIZE RESULT STORAGE
   - Store slope and intercept values for selected temperature
   - Prepare to store plots for all columns

7. FOR EACH COLUMN IN THE DATAFRAME (excluding selected):
   - Perform linear regression: Fit reference Y with column X
   - Store calculated slope (m) and intercept (c)
   - Generate and save regression plots for each column

8. SORT RESULTS BY TEMPERATURE (Column name)
   - Sort list of m and c by temperature order (numerical)

9. SAVE INITIAL RESULTS TO EXCEL
   - Save reference values
   - Save slope and intercept (m and c)
   - Save list of plot paths

10. PERFORM POLYNOMIAL REGRESSION
   - Ask user for the degree of polynomial
   - Fit polynomial for:
      a. Temperature vs slope
      b. Temperature vs intercept
   - Display regression equations
   - Evaluate fitted values (m_noise and c_noise)
   - Add these to the final results DataFrame

11. UPDATE EXCEL FILE WITH POLYFIT RESULTS
   - Replace "final m and c" sheet with updated polynomial results

12. GENERATE POLYFIT PLOTS
   - Plot: Temperature vs slope (fitted)
   - Plot: Temperature vs intercept (fitted)
   - Save both plots in regression plot folder

13. COMPUTE Y VALUES USING FITTED m AND c
   - For each temperature column:
      • Compute y = m_noise * x + c_noise
      • Store results in new DataFrame
```

```
14. CALCULATE DIFFERENCES FROM REFERENCE VALUES
    - Subtract reference values from computed values
    - Store difference values in a new DataFrame
    - Compute max absolute difference per column
    - Add row for max and overall maximum difference

15. SAVE FINAL RESULTS TO EXCEL
    - Save computed Y values to "computed values" sheet
    - Save difference DataFrame to "Differences" sheet

16. DISPLAY SUMMARY TO USER
    - Show paths to plots and final Excel file
    - Print overall max absolute difference

END PROGRAM
```

APPENDIX-B

SCREENSHOTS

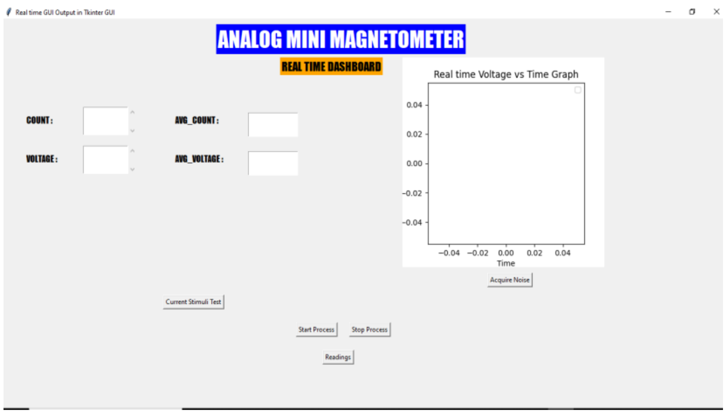


Figure 10.1 Main GUI

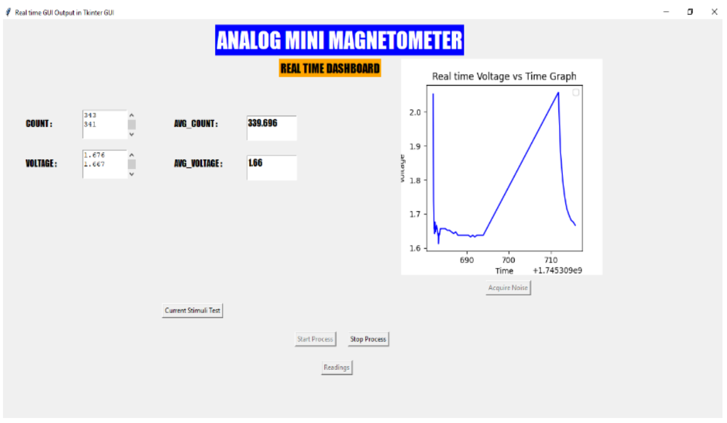
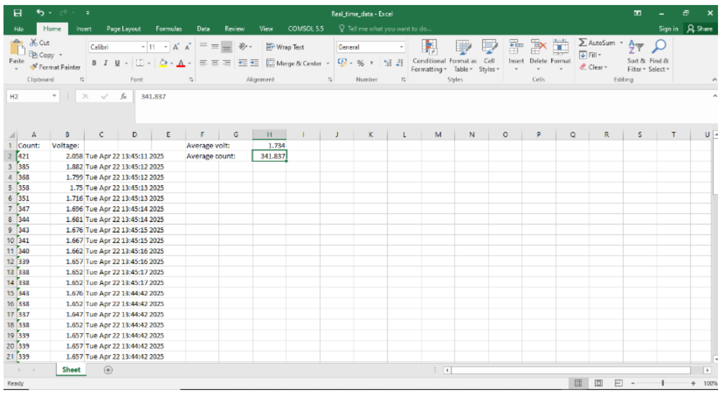


Figure 10.2 Serial port reading and real time plotting



Real_time_data - Excel

H2 341.837

	A	B	C	D	E	F	G	H
1	Count:	Voltage:				Average volt:		1.734
2	421	2.058	Tue Apr 22 13:45:11 2025			Average count:		341.837
3	385	1.882	Tue Apr 22 13:45:12 2025					
4	368	1.799	Tue Apr 22 13:45:12 2025					
5	358	1.75	Tue Apr 22 13:45:13 2025					
6	351	1.716	Tue Apr 22 13:45:13 2025					
7	347	1.696	Tue Apr 22 13:45:14 2025					
8	344	1.681	Tue Apr 22 13:45:14 2025					
9	343	1.676	Tue Apr 22 13:45:15 2025					
10	341	1.667	Tue Apr 22 13:45:15 2025					
11	340	1.662	Tue Apr 22 13:45:16 2025					
12	339	1.657	Tue Apr 22 13:45:16 2025					
13	338	1.652	Tue Apr 22 13:45:17 2025					
14	338	1.652	Tue Apr 22 13:45:17 2025					
15	343	1.676	Tue Apr 22 13:44:42 2025					
16	338	1.652	Tue Apr 22 13:44:42 2025					
17	337	1.647	Tue Apr 22 13:44:42 2025					
18	338	1.652	Tue Apr 22 13:44:42 2025					
19	339	1.657	Tue Apr 22 13:44:42 2025					
20	339	1.657	Tue Apr 22 13:44:42 2025					
21	339	1.657	Tue Apr 22 13:44:42 2025					

Figure 10.3 Serial Data accumulation

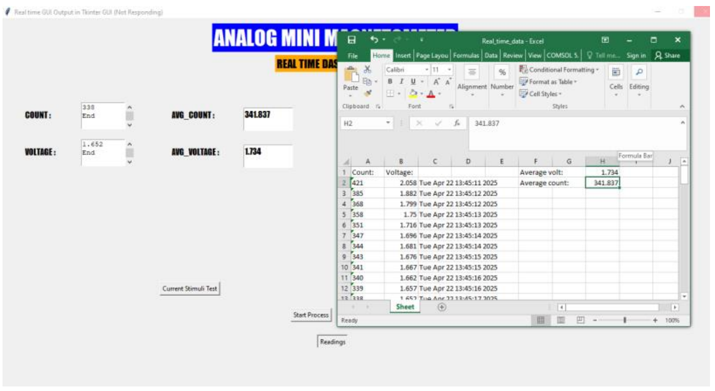


Figure 10.4 Real time Serial Data storage

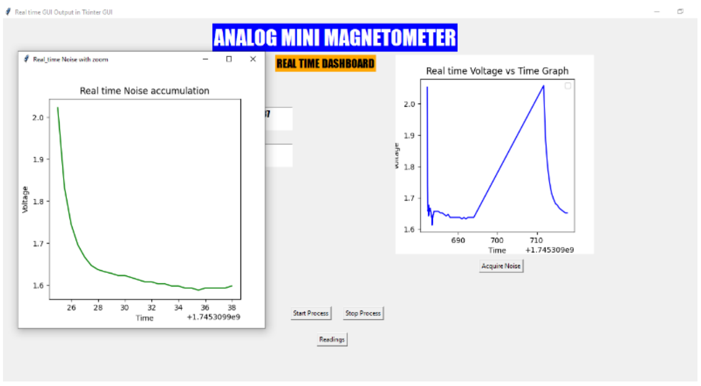


Figure 10.5 Real time Noise accumulation with zoomable plot

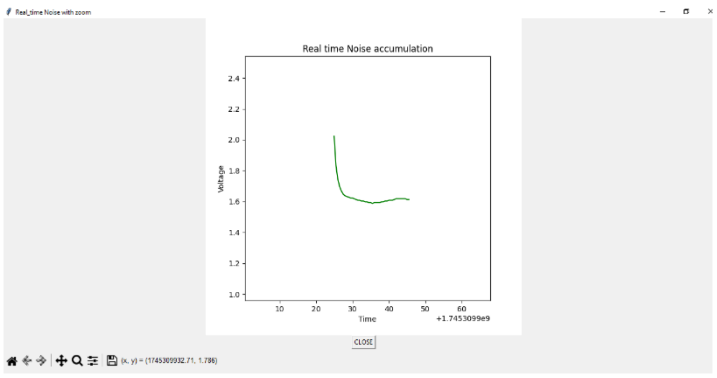

Real_time Noise with zoom
Real time Noise accumulation
Voltage
Time
+1.7453099e9
CLOSE
(x, y) = (1745309932.71, 1.786)

Figure 10.6 Zoomed out Noise Plot

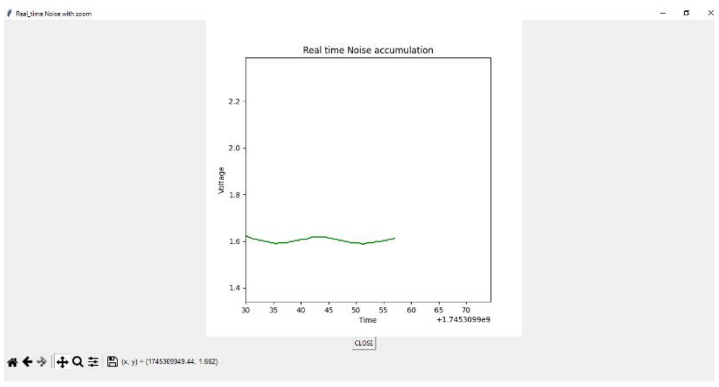
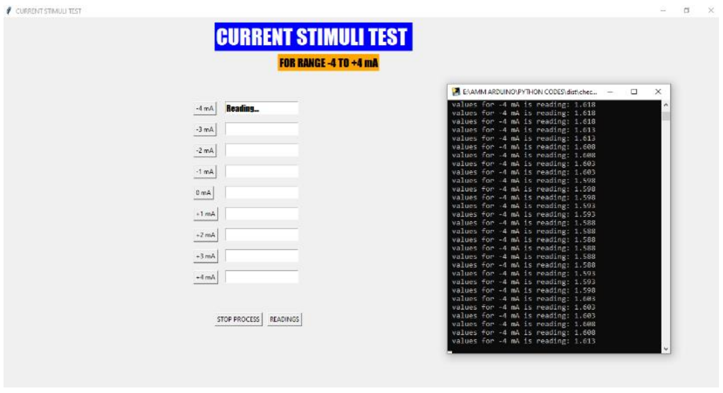
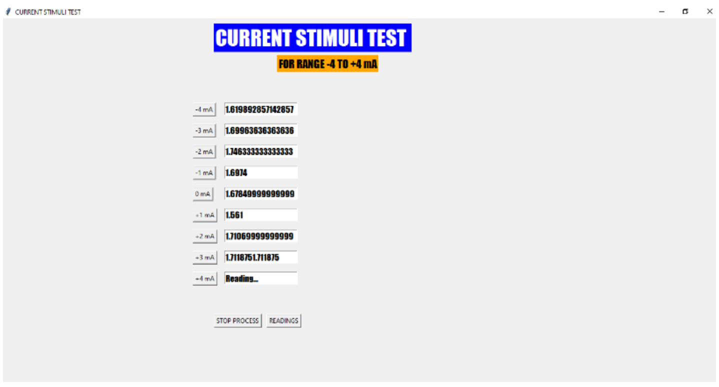


Figure 10.7 Zoomed in Noise plot

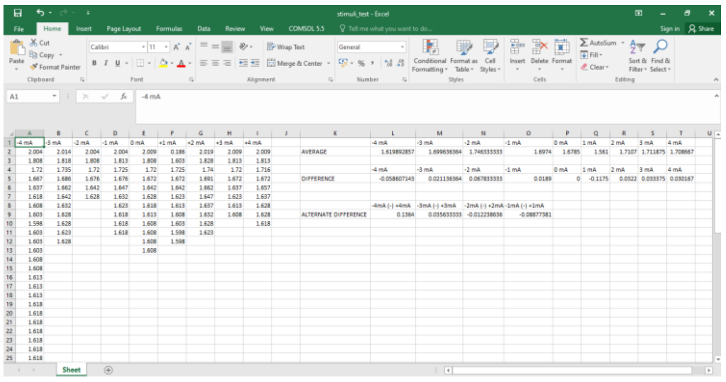


```
values for -4 mA is reading: 1.618
values for -4 mA is reading: 1.618
values for -4 mA is reading: 1.618
values for -4 mA is reading: 1.613
values for -4 mA is reading: 1.613
values for -4 mA is reading: 1.608
values for -4 mA is reading: 1.608
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.598
values for -4 mA is reading: 1.598
values for -4 mA is reading: 1.598
values for -4 mA is reading: 1.593
values for -4 mA is reading: 1.593
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.588
values for -4 mA is reading: 1.593
values for -4 mA is reading: 1.593
values for -4 mA is reading: 1.598
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.603
values for -4 mA is reading: 1.608
values for -4 mA is reading: 1.608
values for -4 mA is reading: 1.613
```

Figure 10.8 Current Stimuli Test GUI



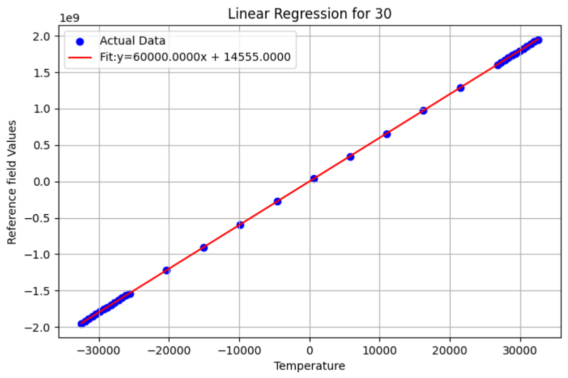
-4 mA	1.619892857142857
-3 mA	1.69963636363636
-2 mA	1.746333333333333
-1 mA	1.6974
0 mA	1.67849999999999
+1 mA	1.561
+2 mA	1.71069999999999
+3 mA	1.7118751.711875
+4 mA	Reading...

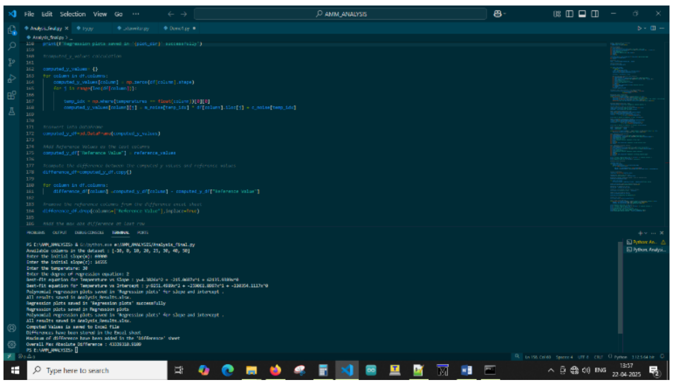


	A	B	C	D	E	F	G	H	I
1	-4 mA	-3 mA	-2 mA	-1 mA	0 mA	+1 mA	+2 mA	+3 mA	+4 mA
2	2.004	2.014	2.004	2.004	2.009	0.186	2.019	2.009	2.009
3	1.808	1.818	1.808	1.813	1.808	1.603	1.828	1.813	1.813
4	1.72	1.735	1.72	1.725	1.72	1.725	1.74	1.72	1.716
5	1.667	1.686	1.676	1.676	1.672	1.672	1.691	1.672	1.672
6	1.637	1.662	1.642	1.647	1.642	1.642	1.662	1.637	1.657
7	1.618	1.642	1.628	1.632	1.628	1.623	1.647	1.623	1.637
8	1.608	1.632		1.623	1.618	1.613	1.637	1.613	1.628
9	1.603	1.628		1.618	1.613	1.608	1.632	1.608	1.628
10	1.598	1.628		1.618	1.608	1.603	1.628		1.618
11	1.603	1.623		1.618	1.608	1.598	1.623		
12	1.603	1.628			1.608	1.598			
13	1.603				1.608				
14	1.608								
15	1.608								
16	1.613								
17	1.613								
18	1.613								
19	1.618								
20	1.618								
21	1.618								
22	1.618								
23	1.618								
24	1.618								
25	1.618								

	K	L	M	N	O	P	Q	R	S	T
1		-4 mA	-3 mA	-2 mA	-1 mA	0 mA	1 mA	2 mA	3 mA	4 mA
2	AVERAGE	1.619892857	1.699636364	1.746333333	1.6974	1.6785	1.561	1.7107	1.711875	1.708667
3										
4		-4 mA	-3 mA	-2 mA	-1 mA	0 mA	1 mA	2 mA	3 mA	4 mA
5	DIFFERENCE	-0.058607143	0.021136364	0.067833333	0.0189	0	-0.1175	0.0322	0.033375	0.030167
6										
7										
8		-4mA (-) +4mA	-3mA (-) +3mA	-2mA (-) +2mA	-1mA (-) +1mA					
9	ALTERNATE DIFFERENCE	0.1364	0.035633333	-0.012238636	-0.08877381					

Figure 10.9 Current stimuli test processing Figure 10.10 Current Stimuli Test Result Excel sheet





```
computed_y_df=pd.DataFrame(computed_y_values)

# Add Reference Values as the last column
computed_y_df["Reference Value"] = reference_values

# Compute the difference between the computed y values and reference values
difference_df=computed_y_df.copy()

for column in df.columns:
    difference_df[column] =computed_y_df[column] - computed_y_df["Reference Value"]

# Remove the reference column from the difference excel sheet
difference_df.drop(columns=["Reference Value"],inplace=True)

# Add the max abs difference at last row
```

```
PS E:\AMM_ANALYSIS> & C:/python.exe e:/AMM_ANALYSIS/Analysis_final.py
Available columns in the dataset : [-30, 0, 10, 20, 25, 30, 40, 50]
Enter the initial slope(m): 60000
Enter the initial slope(c): 14555
Enter the temperature: 30
Enter the degree of regression equation: 2
Best-fit equation for Temperature vs Slope : y=4.3826x^2 + -215.0607x^1 + 62135.9380x^0
Best-fit equation for Temperature vs Intercept : y=9151.4580x^2 + -259061.8867x^1 + -130354.1117x^0
Polynomial regression plots saved in 'Regression plots' for slope and intercept .
All results saved in Analysis_Results.xlsx.
Regression plots saved in 'Regression plots' successfully
Regression plots saved in Regression plots
Polynomial regression plots saved in 'Regression plots' for slope and intercept .
All results saved in Analysis_Results.xlsx.
Computed Values is saved to Excel file
Differences have been stored in the Excel sheet
Maximum of difference have been added in the 'Difference' sheet
Overall Max Absolute Difference : 43330310.9109
PS E:\AMM_ANALYSIS> []
```

Figure 10.11 Linear Regression Figure 10.12 Analysis Console Output

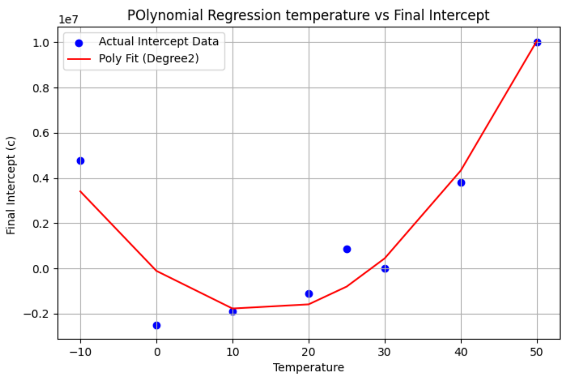


Figure 10.13 Intercept Plot

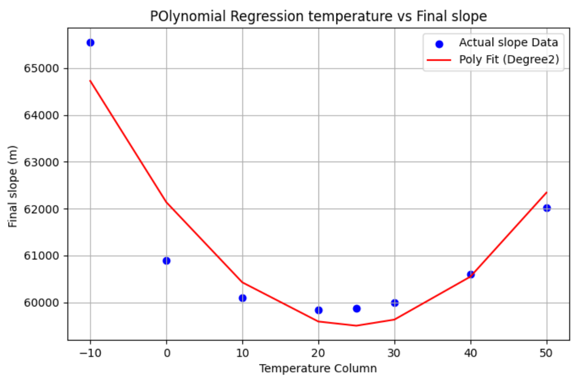
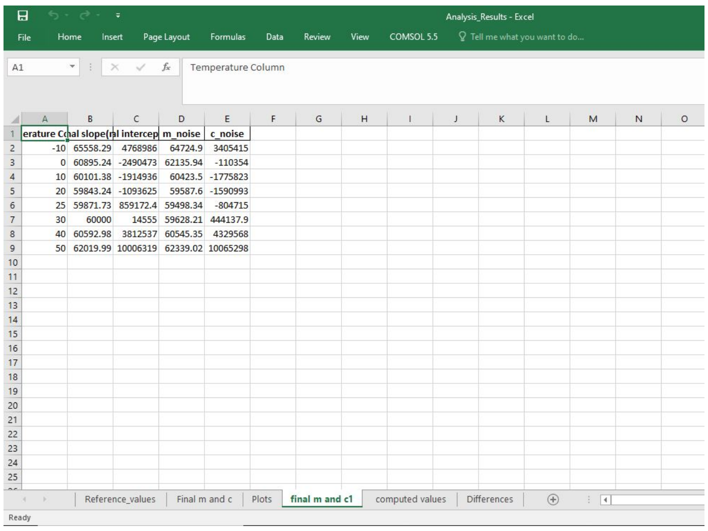


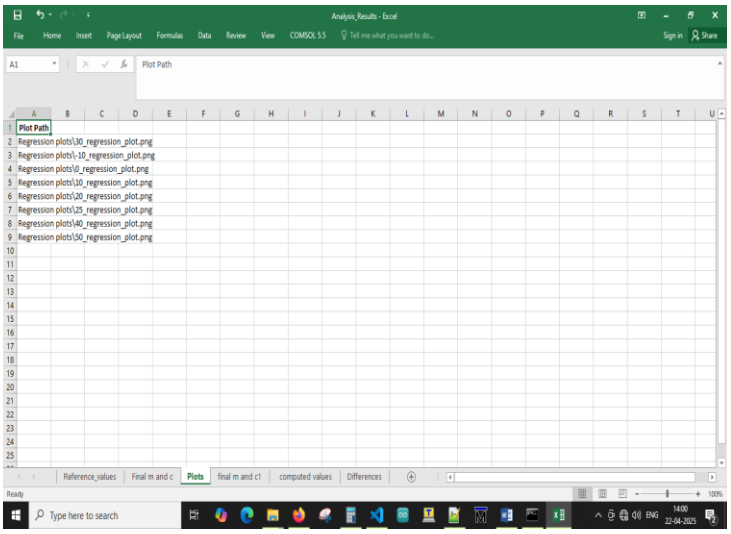
Figure 10.14 Slope Plot



Temperature Column

	A	B	C	D	E
1	erature Co	al slope(m	l intercep	m_noise	c_noise
2	-10	65558.29	4768986	64724.9	3405415
3	0	60895.24	-2490473	62135.94	-110354
4	10	60101.38	-1914936	60423.5	-1775823
5	20	59843.24	-1093625	59587.6	-1590993
6	25	59871.73	859172.4	59498.34	-804715
7	30	60000	14555	59628.21	444137.9
8	40	60592.98	3812537	60545.35	4329568
9	50	62019.99	10006319	62339.02	10065298

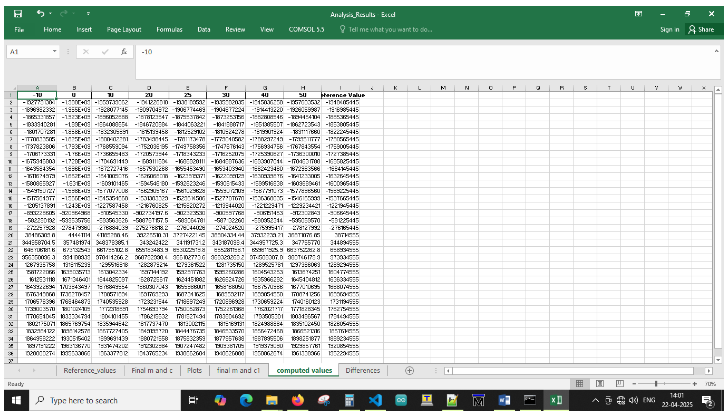
Figure 10.15 M and C Values Excel sheet



Plot Path

	A
1	Plot Path
2	Regression plots\30_regression_plot.png
3	Regression plots\-10_regression_plot.png
4	Regression plots\0_regression_plot.png
5	Regression plots\10_regression_plot.png
6	Regression plots\20_regression_plot.png
7	Regression plots\25_regression_plot.png
8	Regression plots\40_regression_plot.png
9	Regression plots\50_regression_plot.png

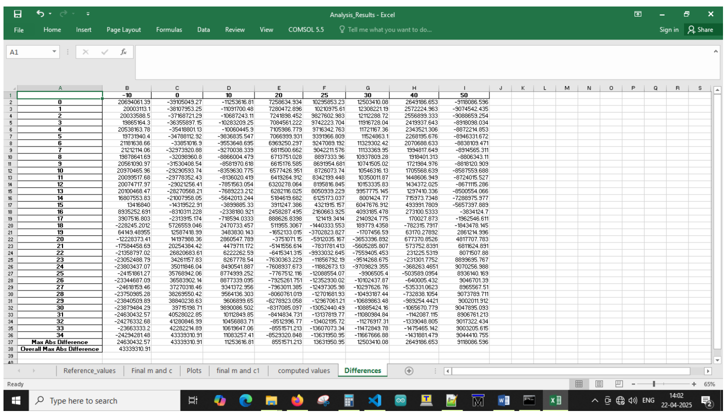
Figure 10.16 path for the plots



-10

	A	B	C	D	E	F	G	H	I
1	-10	0	10	20	25	30	40	50	eference Value
2	-1927791384	-1.988E+09	-1959739062	-1941226810	-1938189592	-1935982035	-1945836258	-1957603532	-1948485445
3	-1896982332	-1.955E+09	-1928077145	-1909704972	-1906774469	-1904677224	-1914413220	-1926059987	-1916985445
4	-1865331857	-1.923E+09	-1896052688	-1878123547	-1875537842	-1873253156	-1882808546	-1894454104	-1885365445
5	-1833940281	-1.89E+09	-1864088654	-1846720884	-1844063221	-1841888717	-1851385507	-1862723543	-1853805445
6	-1801707281	-1.858E+09	-1832305891	-1815139458	-1812529102	-1810524278	-1819901924	-1831117660	-1822245445
7	-1770833505	-1.825E+09	-1800402281	-1783498445	-1781173478	-1779040582	-1788297249	-1799511777	-1790565445
8	-1737823806	-1.793E+09	-1768559094	-1752036195	-1749758356	-1747676143	-1756934756	-1767843554	-1759005445
9	-1706173331	-1.76E+09	-1736655483	-1720573944	-1718343233	-1716252075	-1725390627	-1736300010	-1727385445
10	-1675946803	-1.728E+09	-1704691449	-1689111694	-1686928111	-1684887636	-1693907044	-1704631788	-1695825445
11	-1643584354	-1.696E+09	-1672727416	-1657530268	-1655453490	-1653403940	-1662423460	-1672963566	-1664145445
12	-1611674979	-1.662E+09	-1641005076	-1626068018	-1623919371	-1622099129	-1630939876	-1641233005	-1632645445
13	-1580865927	-1.631E+09	-1609101465	-1594546180	-1592623246	-1590615433	-1599516838	-1609689461	-1600965445
14	-1549150727	-1.598E+09	-1577077008	-1562905167	-1561029628	-1559072109	-1567791073	-1577896560	-1569225445
15	-1517564977	-1.566E+09	-1545354668	-1531383329	-1529614506	-1527707670	-1536368035	-1546165999	-1537665445
16	-1205137891	-1.243E+09	-1227587458	-1216760825	-1215820272	-1213944020	-1221229471	-1229234421	-1221945445
17	-893228605	-920964968	-910545330	-902734197.6	-902323530	-900597768	-906151453	-912302843	-906645445
18	-582290192	-599535756	-593563626	-588767157.5	-589064781	-587132260	-590952344	-595059570	-591225445
19	-272257928	-278479360	-276884039	-275276818.2	-276044026	-274024520	-275995417	-278127992	-276165445
20	38486309.8	44441114	41185288.46	39226510.31	37274221.45	38904334.44	37932239.21	36871076.85	38714555
21	344958704.5	357481974	348378385.1	343242422	341191731.2	343187098.4	344957725.3	347755770	344894555
22	646706181.6	673132543	661795102.8	655183483.9	653022519.8	655281158.1	659611925.9	663752262.8	658934555
23	956350096.3	994188939	978414266.2	968792998.4	966102773.6	968329269.2	974508307.8	980746179.9	973934555
24	1267935758	1316115239	1295516818	1282879214	1279361522	1281735150	1289525781	1297366063	1289294555
25	1581722066	1639035713	1613042334	1597144192	1592917763	1595260286	1604543253	1613674251	1604774555
26	1612531118	1671346401	1644825097	1628725617	1624451882	1626624726	1635966292	1645404812	1636334555
27	1643922694	1703843497	1676849554	1660307043	1655986001	1658168050	1667570966	1677010695	1668074555
28	1676349868	1736278457	1708571894	1691769293	1687341625	1689592117	1699054550	1708741256	1699694555
29	1706576396	1768464873	1740535928	1723231544	1718697249	1720896928	1730659224	1740160123	1731194555
30	1739003570	1801024105	1772318691	1754693794	1750052873	1752261368	1762021717	1771828345	1762754555
31	1770654045	1833334794	1804101455	1786215632	1781527494	1783804692	1793505301	1803496567	1794494555
32	1802175071	1865769754	1835944642	1817737470	1813002115	1815169131	1824988884	1835102450	1826054555
33	1832984122	1898142578	1867727405	1849199720	1844476735	1846533570	1856472468	1866521316	1857614555
34	1864958222	1930515402	1899691439	1880721558	1875832359	1877957638	1887895506	1898251877	1889234555
35	1897191222	1963136770	1931474202	1912302984	1907247482	1909381705	1919379090	1929857761	1920854555
36	1928000274	1995633866	1963377812	1943765234	1938662604	1940626888	1950862674	1961338966	1952294555

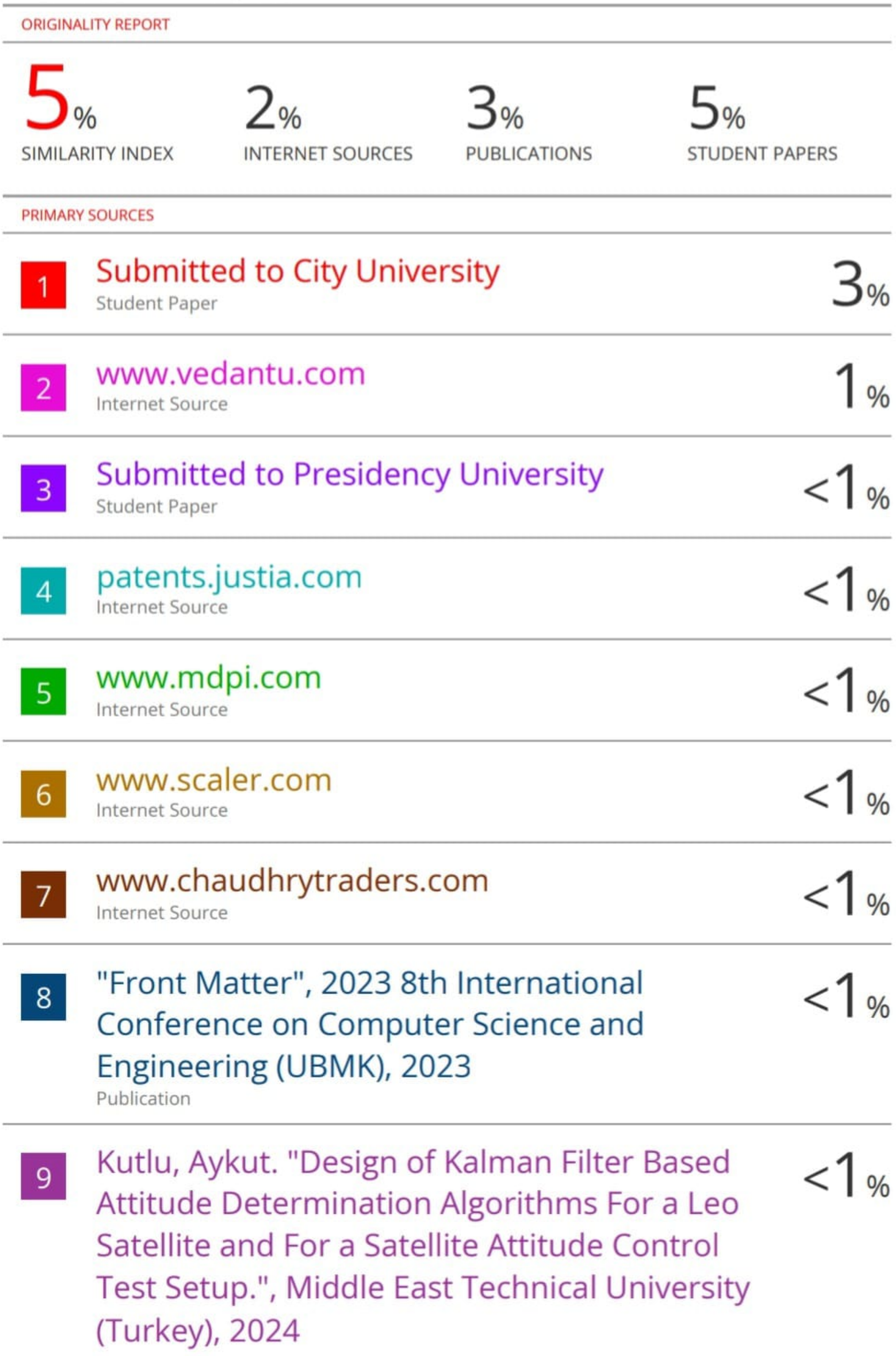
Figure 10.17 Computed Value



	A	B	C	D	E	F	G	H	I
1		-10	0	10	20	25	30	40	50
2	0	20694061.39	-39105049.27	-11253616.81	7258634.934	10295853.23	12503410.08	2649186.653	-9118086.596
3	1	20003113.1	-38107953.25	-11091700.48	7280472.896	10210975.61	12308221.19	2572224.963	-9074542.435
4	2	20033588.5	-37168721.29	-10687243.11	7241898.452	9827602.983	12112288.72	2556899.333	-9088659.254
5	3	19865164.3	-36355897.15	-10283209.25	7084561.222	9742223.704	11916728.04	2419937.643	-8918098.034
6	4	20538163.78	-35418801.13	-10060445.9	7105986.779	9716342.763	11721167.36	2343521.306	-8872214.853
7	5	19731940.4	-34788112.92	-9836835.547	7066999.931	9391966.809	11524863.1	2268195.676	-8946331.672
8	6	21181638.66	-33851016.9	-9553648.695	6969250.297	9247089.192	11329302.42	2070688.633	-8838109.471
9	7	21212114.06	-32973920.88	-9270038.339	6811500.662	9042211.576	11133369.95	1994817.649	-8914565.311
10	8	19878641.69	-32098960.8	-8866004.479	6713751.028	8897333.96	10937809.28	1918401.313	-8806343.11
11	9	20561090.97	-31530408.54	-8581970.618	6615176.585	8691954.681	10741505.02	1721984.976	-8818120.909
12	10	20970465.96	-29290593.74	-8359630.775	6577426.951	8726073.74	10546316.13	1705568.639	-8587559.688
13	11	20099517.68	-29778352.43	-8136020.419	6419264.912	8342199.448	10350011.87	1448606.949	-8724015.527
14	12	20074717.97	-29021256.41	-7851563.054	6320278.064	8195816.845	10153335.83	1434372.025	-8671115.286
15	13	20100468.47	-28270568.21	-7689223.212	6282116.025	8050939.229	9957775.145	1297410.336	-8500554.066
16	14	16807553.83	-21007958.05	-5642013.244	5184619.682	6125173.037	8001424.77	715973.7348	-7288975.977
17	15	13416840	-14319522.91	-3899885.33	3911247.386	4321915.157	6047676.912	493991.7809	-5657397.889
18	16	8935252.691	-8310311.228	-2338180.921	2458287.495	2160663.925	4093185.478	273100.5333	-3834124.7
19	17	3907516.803	-2313915.174	-718594.0333	888626.8398	121419.3414	2140924.775	170027.873	-1962546.611
20	18	-228245.2012	5726559.046	2470733.457	511955.3067	-1440333.553	189779.4358	-782315.7917	-1843478.145
21	19	64149.48955	12587418.99	3483830.143	-1652133.015	-3702823.827	-1707456.59	63170.27892	2861214.996
22	20	-12228373.41	14197988.36	2860547.789	-3751071.15	-5912035.167	-3653396.892	677370.8526	4817707.783
23	21	-17584458.69	20254384.42	4479711.172	-5141556.614	-7831781.413	-5605285.807	573752.8391	6811624.891
24	22	-21358797.02	26820683.61	6222262.59	-6415341.315	-9933032.645	-7559405.453	231225.5319	8071507.88
25	23	-23052488.79	34261157.83	8267778.54	-7630363.229	-11856792.19	-9514268.675	-231301.7752	8899695.767
26	24	-23803437.07	35011846.04	8490541.887	-7608937.673	-11882673.13	-9709829.355	-368263.4651	9070256.988
27	25	-24151861.27	35768942.06	8774999.252	-7767512.116	-12088554.07	-9906505.4	-503589.0954	8936140.169
28	26	-23344687.09	36583902.14	8877339.095	-7925261.751	-12352930.02	-10102437.87	-640005.432	9046701.39
29	27	-24618159.46	37270318.46	9341372.956	-7963011.385	-12497305.98	-10297626.76	-535331.0623	8965567.51
30	28	-23750985.28	38269550.42	9564136.303	-8060761.019	-12701681.93	-10493187.44	-732838.1054	9073789.711
31	29	-23840509.89	38840238.63	9606899.65	-8278923.058	-12967061.21	-10689863.48	-989254.4421	9002011.912
32	30	-23879484.29	39715198.71	9890086.502	-8317085.097	-13052440.49	-10885424.16	-1065670.779	9047895.093
33	31	-24630432.57	40528022.85	10112849.85	-8414834.731	-13137819.77	-11080984.84	-1142087.115	8906761.213
34	32	-24276332.68	41280846.99	10456883.71	-8512996.77	-13402195.72	-11276917.31	-1339048.805	9017322.434
35	33	-23663333.2	42282214.89	10619647.06	-8551571.213	-13607073.34	-11472849.78	-1475465.142	9003205.615
36	34	-24294281.48	43339310.91	11083257.41	-8529320.848	-13631950.95	-11667666.88	-1431881.479	9044410.755
37	Max Abs Difference	24630432.57	43339310.91	11253616.81	8551571.213	13631950.95	12503410.08	2649186.653	9118086.596
38	Overall Max Abs Difference	43339310.91							

Figure 10.18 Final output of Analysis (Differences)

APPENDIX-C

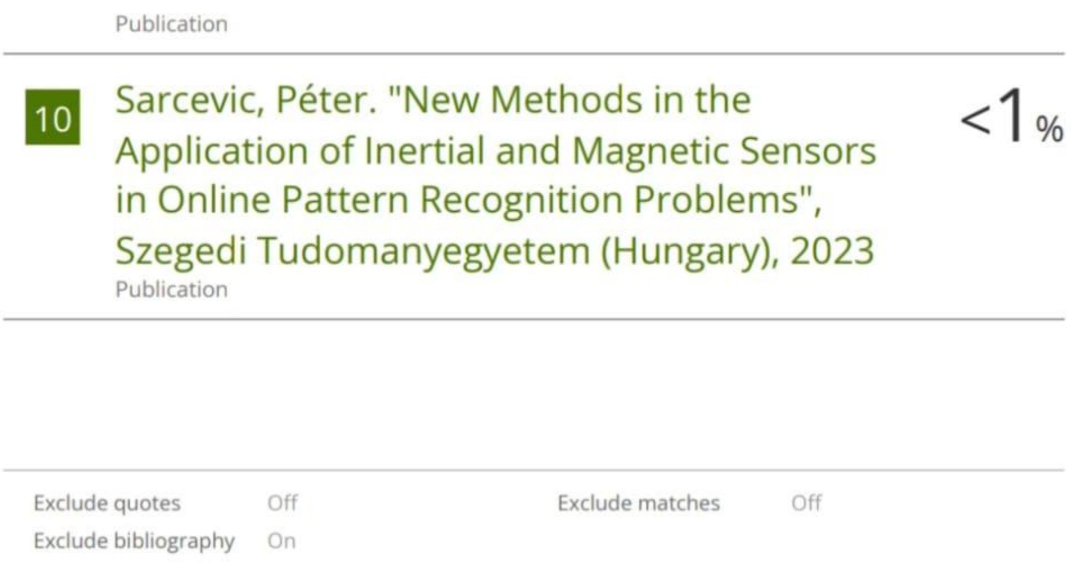


ORIGINALITY REPORT

5%	2%	3%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to City University Student Paper	3%
2	www.vedantu.com Internet Source	1%
3	Submitted to Presidency University Student Paper	<1%
4	patents.justia.com Internet Source	<1%
5	www.mdpi.com Internet Source	<1%
6	www.scaler.com Internet Source	<1%
7	www.chaudhrytraders.com Internet Source	<1%
8	"Front Matter", 2023 8th International Conference on Computer Science and Engineering (UBMK), 2023 Publication	<1%
9	Kutlu, Aykut. "Design of Kalman Filter Based Attitude Determination Algorithms For a Leo Satellite and For a Satellite Attitude Control Test Setup.", Middle East Technical University (Turkey), 2024	<1%



Publication

10	Sarcevic, Péter. "New Methods in the Application of Inertial and Magnetic Sensors in Online Pattern Recognition Problems", Szegedi Tudomanyegyetem (Hungary), 2023 Publication	<1 %

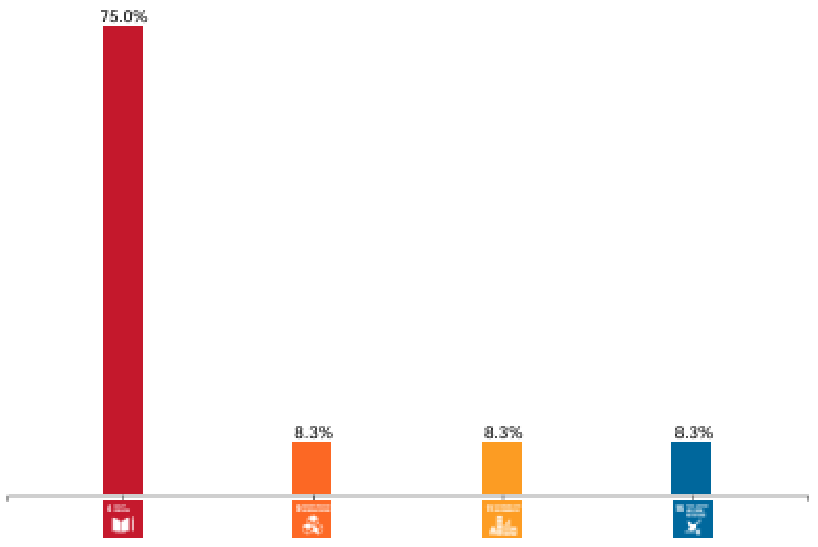
Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

SUSTAINABLE DEVELOPMENT GOAL

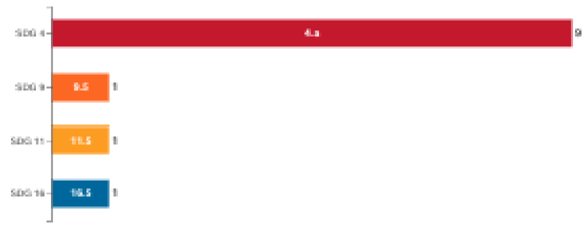
SDG Report - Real time Dashboard and Analysis for ANALOG MINI MAGNETOMETER

This SDG mapping has been made with the JRC SDG Mapper. The main slide shows the SDGs detected (by ranking). A second slide provides granular information at the level of the detected SDG targets. The SDG mapper can be accessed just with ECAS login at https://knowsdgs.jrc.ec.europa.eu/sdgmapper. Basic instructions for use are found here https://knowsdgs.jrc.ec.europa.eu/sdgmapper#learn.

Relevant SDGs



SDG Targets


SDG 4
4.a
SDG 9
SDG 11
SDG 16