

MACHINE CONDITION MONITORING USING A PREDICTIVE MAINTENANCE

NURUL IZZAH BINTI AZMI



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

MACHINE CONDITION MONITORING USING A PREDICTIVE MAINTENANCE

NURUL IZZAH BINTI AZMI

**This report is submitted in partial fulfilment of the requirements
for the degree of Bachelor of Electronic Engineering with Honours**

**Faculty of Electronic and Computer Technology Engineering
Universiti Teknikal Malaysia Melaka**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

**BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II**

Tajuk Projek : Machine Condition Monitoring Using A Predictive Maintenance
Sesi Pengajian : 2022/2023

Saya NURUL IZZAH BINTI AZMI mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

SULIT*

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

TERHAD*

(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.)

TIDAK TERHAD

Disahkan oleh:


(TANDATANGAN PENULIS)


(COP DAN TANDATANGAN PENYELIA)

Alamat Tetap: Lot 97, Kg Kuala Sungai Baru, Batu 13, Jalan Klang, 47100, Puchong, Selangor

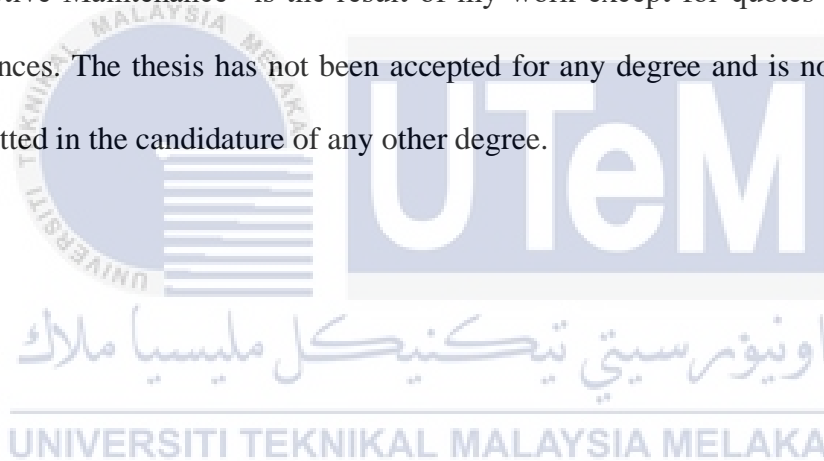
Tarikh : 24 Januari 2024

KHAIRUN NISA BINTI KHAMIL (Ph.D)
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER (FTKEK)
UNIVERSITI TEKNIKAL MALAYSIA MELAKA (UTeM)
HANG TUAH JAYA, DURIAN TUNGGAL,
76100, MELAKA

Tarikh : 24 January 2023

DECLARATION

I declare that this report entitled “Machine Condition Monitoring System Using Predictive Maintenance” is the result of my work except for quotes as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in the candidature of any other degree.



Signature :

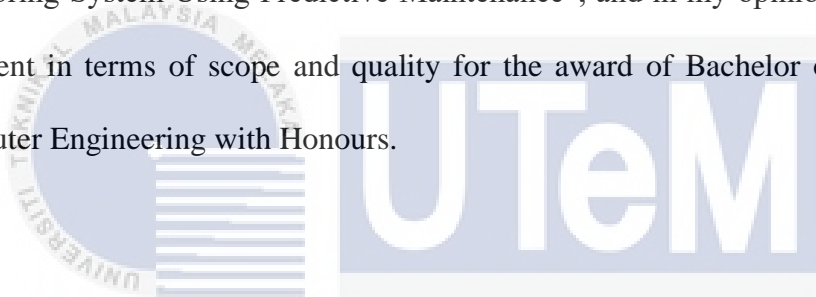
A handwritten signature in black ink, appearing to be 'Nurul Izzah Binti Azmi', is written over the dotted line for the signature field.

Author : ..NURUL IZZAH BINTI AZMI..

Date : ..24 JANUARI 2024.....

APPROVAL

I hereby declare that I have checked this thesis entitled “Machine Condition Monitoring System Using Predictive Maintenance”, and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic / Computer Engineering with Honours.



Signature : 

Supervisor Name : Khairun Nisa binti Khamil

Date : 24 January 2024

DEDICATION

I would like to dedicate the success of this project research, especially to my supervisor which is Dr. Khairun Nisa binti Khamil. This thesis will be dedicated to her because I want to thank her for all the sacrifices that she made for me while I have been studying at this university. Secondly, this dedication is given to my father which is Azmi bin Esa who always being for me when I needed and helps me to accomplish the project. Next, I would like to express my gratitude to assistant lab Asecs, Mr. Hairul and my friends gave a lot of help while completing this bachelor's degree project.

ABSTRACT

Monitoring the condition of machines plays a vital role in guaranteeing the best possible performance and lifespan of industrial equipment. This research proposes a predictive maintenance system for monitoring machine conditions and enhancing industrial equipment performance and longevity. Leveraging the Raspberry Pi, sensors, and Node-RED, the system utilizes various sensors to collect real-time data on motor aspects like temperature and vibration. The Raspberry Pi serves as the central unit for data collection, processing, and analysis. Utilizing Node-RED's visual programming, a comprehensive monitoring and predictive maintenance plan is developed. The system employs advanced data analysis, including machine learning, to identify patterns and anomalies in sensor data, allowing for proactive fault detection aligning with predictive maintenance strategies. The comparison between two motors is analyzed by comparing the vibration and temperature variation. Each motor produces different vibration levels during operation and the differences in the motor's age, design, and maintenance can affect motor efficiency. The effectiveness of the proposed system is demonstrated through experiments conducted on a real motor setup.

ABSTRAK

Penyelidikan ini mencadangkan sistem penyelenggaraan meramal untuk memantau keadaan mesin dan meningkatkan prestasi serta jangka hayat peralatan industri. Dengan menggunakan Raspberry Pi, sensor, dan Node-RED, sistem ini menggunakan pelbagai sensor untuk mengumpul data secara langsung mengenai aspek motor seperti suhu dan getaran. Raspberry Pi berfungsi sebagai unit pusat untuk pengumpulan, dan analisis data. Dengan menggunakan pemrograman visual Node-RED, satu cadangan pemantauan meramal yang menyeluruh dijalankan. Sistem ini menggunakan analisis data canggih, termasuk pembelajaran mesin, untuk mengenal pasti corak dan anomali dalam data sensor, membolehkan pengesanan ralat secara proaktif sejajar dengan strategi penyelenggaraan meramal. Perbandingan antara dua motor dianalisis dengan membandingkan variasi getaran dan suhu. Setiap motor menghasilkan tahap getaran yang berbeza semasa operasi dan perbezaan dalam usia, reka bentuk, dan penyelenggaraan motor boleh memberi kesan kepada kecekapan motor. Keberkesanan sistem yang dicadangkan ini ditunjukkan melalui eksperimen yang dijalankan pada suatu susunan motor sebenar.

ACKNOWLEDGEMENTS

I would like to start by thanking Allah and sending blessings to Prophet Muhammad (S.A.W). I am thankful to God for giving me the strength and patience to finish my final year project and thesis on time. I want to express my sincere gratitude to everyone who supported and contributed to the completion of this project. First and foremost, I deeply appreciate my supervisor, Dr. Khairun Nisa Binti Khamil, for her guidance, expertise, and unwavering support throughout this journey. Her valuable insights and feedback played a vital role in shaping this research.

I also want to thank the participants who generously dedicated their time and shared their knowledge, enabling me to collect the necessary data for this study. Special thanks to Cik Hairol for allowing me to use the Lab Research for my project. Their cooperation and willingness to contribute were extremely valuable.

Lastly, I want to express my gratitude to my family and friends for their constant encouragement and understanding throughout the challenges and successes of this project. Their unwavering belief in my abilities has been a great source of motivation.

TABLE OF CONTENTS

Declaration	
Approval	
Dedication	
Abstract	i
Abstrak	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures	ix
List of Tables	xiii
List of Symbols and Abbreviations	xv
List of Appendices	xvi
CHAPTER 1 INTRODUCTION	1
1.1 Background	2
1.2 Problem Statement	4
1.3 Project Objectives	5
1.4 Project Scope and Project Impact	5

1.5	Significance of Study	7
1.6	Propose Solutions for the Project.	7
1.7	Structure of the Thesis	9
CHAPTER 2 BACKGROUND STUDY		11
2.1	Background Literature Review	11
2.2	Condition Monitoring (CM)	12
2.3	Vibration and Temperature Data	17
2.4	Node-RED Visual Programming Tool	22
2.5	Predictive Maintenance using MATLAB apps.	24
2.6	Comparison between technique used and parameters considered on literature.	25
CHAPTER 3 METHODOLOGY		28
3.1	General Methodology	29
3.2	Flowchart	29
3.2.1	Flowchart of Integration Circuit	33
3.3	Modern tools for enhanced project performance	35
3.3.1	Circuit integration between Raspberry Pi, ADXL345 accelerometer sensor, and DS18B20 temperature digital sensor module	35
3.3.2	Placement of sensors at motors	37
3.3.3	Node-RED development tool	38
3.3.3.1	Node-RED applications	39

3.3.3.2	Installation of Nodejs and Node-RED on Raspberry Pi 3, Model b	42
3.3.3.3	Overview of Node-RED Dashboard of the project	43
3.3.3.4	Functional overview of Node-RED Flow	44
3.3.4	3D-printing of prototype	47
3.3.5	Python code for ADXL345 accelerometer sensor and DS18B20 temperature sensor	48
3.3.6	MATLAB for data analysis and predictive maintenance	49
CHAPTER 4 RESULTS AND DISCUSSION		52
4.1	Collection data for two motors	52
4.2	Detection of motor abnormalities in vibration and temperature levels	53
4.3	A monitoring system using Node-RED Dashboard	55
4.4	A predictive analytic using MATLAB software.	56
4.4.1	Levenberg-Marquardt Train Performance	56
4.4.1.1	DS18B20 data for 3 days at Motor 1	56
4.4.1.2	ADXL345_1 data for 3 days at Motor 1	63
4.4.1.3	ADXL345_2 data for 3 days at Motor 1	66
4.4.1.4	DS18B20 data for 3 days at Motor 2	69
4.4.1.5	ADXL345_1 data for 3 days at Motor 2	74
4.4.1.6	ADXL345_2 data for 3 days at Motor 2	77
4.4.2	Scaled Conjugate Gradient Train Performance	80

4.4.2.1 DS18B20 data for 3 data at Motor 1	80
4.4.2.2 ADXL345_1 data at Motor 1	86
4.4.2.3 ADXL345_2 data at Motor 1	89
4.4.2.4 DS18B20 data at Motor 2	92
4.4.2.5 ADXL345_1 data at Motor 2	97
4.4.2.6 ADXL345_2 data at Motor 2	100
4.4.3 Comparison network training using different training algorithms for each motor.	103
4.5 Relationship with Sustainable Development Goal (SDG).	107
CHAPTER 5 CONCLUSION AND FUTURE WORKS	109
5.1 Future works	110
REFERENCES	112
APPENDICES	120
APPENDIX A: Neural Network Training for ADXL345 sensors using Levenberg-Marquardt algorithm.	120
APPENDIX B: Neural Network Training for ADXL345 sensors using Scaled Conjugate Gradient algorithm.	122
APPENDIX C: Comparison of Actual and Predicted Offset of ADXL345_1 and ADXL345_2 using Levenberg-Marquardt algorithm.	123
APPENDIX D: Comparison of Actual and Predicted Offset of ADXL345_1 and ADXL345_2 using Scaled Conjugate Gradient algorithm.	124

APPENDIX E: Regression graph for Training, Validation, and Testing for DS18B20 sensor and ADXL345 sensors using Levenberg-Marquardt algorithm at Motor 1	125
APPENDIX F: Regression graph for Training, Validation, and Testing for DS18B20 sensor and ADXL345 sensors using Levenberg-Marquardt algorithm at Motor 2	126
APPENDIX G: Regression graph for Training, Validation, and Testing for DS18B20 sensor and ADXL345 sensors using Scaled Conjugate Gradient algorithm at Motor 1	127
APPENDIX H: Regression graph for Training, Validation, and Testing for DS18B20 sensor and ADXL345 sensors using Scaled Conjugate Gradient algorithm at Motor 2	128
APPENDIX I: Graph for Comparison of Actual and Predicted Offset for DS18B20 sensor at Motor 2	129
APPENDIX J: MATLAB code for Levenberg-Marquardt and Scaled Conjugate Gradient algorithm.	130
APPENDIX K: Python code for ADXL345 accelerometer sensor	132
APPENDIX L: Python code for DS18B20 temperature sensor	134

LIST OF FIGURES

Figure 3.1: Flowchart to achieve objectives.	31
Figure 3.2: Flowchart of integration circuit	33
Figure 3.3: Circuit connection between Raspberry Pi, ADXL345 accelerometer sensor, and DS18B20 temperature sensor.....	36
Figure 3.4: Sensor placement at Motor 1	37
Figure 3.5: Sensor placement at Motor 2	38
Figure 3.6: Overview of Node-RED Library	41
Figure 3.7: Command for installing Node-RED in Raspberry Pi.....	42
Figure 3.8: Interface of Nore-RED	43
Figure 3.9: Node-RED Flow	43
Figure 3.10: Node-RED dashboard visualization.	45
Figure 3.11: Node-RED flow for ADXL345 accelerometer sensor.	46
Figure 3.12: Node-RED flow for DS18B20 temperature sensor.	47
Figure 3.13: Ender-3 3D printer.....	48
Figure 3.14: Network pane in MATLAB for NAR network	50
Figure 3.15: Training Progress in Training pane	51
Figure 3.16: Model Summary of training algorithm.....	51
Figure 4.1: Data from ADXL345 sensor and DS18B20 sensor displayed in Node-RED Dashboard	53

Figure 4.2: Node-RED dashboard for ADXL345 sensor's data	54
Figure 4.3: Node-RED dashboard for DS18B20 sensor's data	55
Figure 4.4: Neural Network Training for Levenberg-Marquardt algorithm that data taken for three days at Motor 1.	57
Figure 4.5: Validation performance for DS18B20 sensor using LM algorithm at Motor 1.....	59
Figure 4.6: Regression graph for DS18B20 sensor using LM algorithm at Motor 1	60
Figure 4.7: Graph OF Mean Absolute Error (MAE) between actual and predicted offset for DS18B20 sensor using LM algorithm at Motor 1.....	61
Figure 4.8: Graph for comparison between actual and predicted offset using LM algorithm at Motor 1	62
Figure 4.9: Validation performance for ADXL345_1 using LM algorithm at Motor 1	64
Figure 4.10: Regression graph for ADXL345_1 sensor using LM algorithm at Motor 1.....	65
Figure 4.11: Validation performance for ADXL345_2 sensor using LM algorithm at Motor 1.....	67
Figure 4.12: Regression graph for ADXL345_2 sensor using LM algorithm at Motor 1.....	68
Figure 4.13: Neural Network Training for Levenberg-Marquardt algorithm that data taken for three days at Motor 2.	70
Figure 4.14: Validation performance for DS18B20 sensor using LM algorithm at Motor 2.....	71
Figure 4.15: Regression graph for DS18B20 sensor using LM algorithm at Motor 2	72
Figure 4.16: Graph OF Mean Absolute Error (MAE) between actual and predicted offset for DS18B20 sensor using LM algorithm at Motor 2.....	73
Figure 4.17: Validation performance for ADXL345_1 using LM algorithm at Motor 2	75

Figure 4.18: Regression graph for ADXL345_1 sensor using LM algorithm at Motor 2.....	76
Figure 4.19: Validation performance for ADXL345_2 using LM algorithm at Motor 2	78
Figure 4.20: Regression graph for ADXL345_2 sensor using LM algorithm at Motor 2.....	79
Figure 4.21: Neural Network Training for Scaled Conjugate Gradient algorithm that data taken for three days at Motor 1.	81
Figure 4.22: Validation performance for DS18B20 sensor using SCG algorithm at Motor 1.....	82
Figure 4.23: Regression graph for DS18B20 sensor using SCG algorithm at Motor 1	83
Figure 4.24: Graph for comparison between actual and predicted offset using SCG algorithm at Motor 1	84
Figure 4.25: Graph of Mean Absolute Error (MAE) between actual and predicted offset for DS18B20 sensor using SCG algorithm at Motor 1	85
Figure 4.26: Validation performance for ADXL345_1 sensor using SCG algorithm at Motor 1.....	87
Figure 4.27: Regression graph for ADXL345_1 sensor using SCG algorithm at Motor 1.....	88
Figure 4.28: Validation performance for ADXL345_2 sensor using SCG algorithm at Motor 1.....	90
Figure 4.29: Regression graph for ADXL345_2 sensor using SCG algorithm at Motor 1.....	91
Figure 4.30: Neural Network Training for Scaled Conjugate Gradient algorithm that data taken for three days at Motor 2.	93
Figure 4.31: Validation performance for DS18B20 sensor using SCG algorithm at Motor 2.....	94
Figure 4.32: Regression graph for DS18B20 sensor using SCG algorithm at Motor 2	95

Figure 4.33: Graph of Mean Absolute Error (MAE) between actual and predicted offset for DS18B20 sensor using SCG algorithm at Motor 2.....	96
Figure 4.34: Validation performance for ADXL345_1 sensor using SCG algorithm at Motor 2.....	98
Figure 4.35: Regression graph for ADXL345_1 sensor using SCG algorithm at Motor 2.....	99
Figure 4.36: Validation performance for ADXL345_2 sensor using SCG algorithm at Motor 2.....	101
Figure 4.37: Regression graph for DS18B20 sensor using SCG algorithm at Motor 2.....	102



LIST OF TABLES

Table 2. 1: Comparison on Literature	26
Table 4.1: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of DS18B20 sensor using LM algorithm at Motor 1	63
Table 4.2: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_1 sensor using LM algorithm at Motor 1	66
Table 4.3: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_2 sensor using LM algorithm at Motor 1	69
Table 4.4: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of DS18B20 sensor using LM algorithm at Motor 2	74
Table 4.5: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_1 using LM algorithm at Motor 2	76
Table 4.6: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_2 sensor using LM algorithm at Motor 2	80
Table 4.7: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of DS18B20 sensor using SCG algorithm at Motor 1	86
Table 4.8: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_1 sensor using SCG algorithm at Motor	89
Table 4.9: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_2 sensor using SCG algorithm at Motor 1	92
Table 4.10: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of DS18B20 sensor using SCG algorithm at Motor 2.....	97
Table 4.11: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_1 sensor using SCG algorithm at Motor 2 ..	100

Table 4.12: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_2 sensor using SCG algorithm at Motor 2 .. 103

Table 4.13: Table comparison between using LM algorithm and SCG algorithm.. 105

Table 4.14: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of sensors at Motor 1 and Motor 2 107



LIST OF SYMBOLS AND ABBREVIATIONS

For examples:

LM	:	Levenberg-Marquardt
SCG	:	Scaled Conjugate Gradient
CM	:	Condition Monitoring
CNN	:	Convolutional Neural Networks
LSTM	:	Long Short-Term Memory
ID	:	Induced Draft
PCA	:	Principal Component Analysis
FFT	:	Fast Fourier Transform
AI	:	Artificial Intelligence
PV	:	Photovoltaic
FIS	:	Fuzzy Inference System
DC	:	Direct Current
IoT	:	Industrial of Thing
MEMS	:	Micro Electromechanical Systems
ILFE	:	In-Line Fiber Etalon
DFT	:	Discrete Fourier Transform
WSN	:	Wireless Sensor Network
PDA	:	Personal Digital Assitant

LIST OF APPENDICES

Appendix A: Neural Network training for ADXL345 sensors using Levenberg-Marquardt algorithm.	120
Appendix B: Neural Network Training for ADXL345 sensors using Scaled Conjugate Gradient algorithm.	122
Appendix C: Comparison of Actual and Predicted Offset of ADXL345_1 and ADXL345_2 using Levenberg-Marquardt algorithm.	123
Appendix D: Comparison of Actual and Predicted Offset of ADXL345_1 and ADXL345_2 using Scaled Conjugate Gradient algorithm.	124
Appendix E: Regression graph for Training, Validation, and Testing for DS18B20 sensor and ADXL345 sensors using Levenberg-Marquardt algorithm at Motor 1	125
Appendix F: Regression graph for Training, Validation, and Testing for DS18B20 sensor and ADXL345 sensors using Scaled Conjugate Gradient algorithm at Motor 2	126
Appendix G: Regression graph for Training, Validation, and Testing for DS18B20 sensor and ADXL345 sensors using Scaled Conjugate Gradient algorithm at Motor 1	127

Appendix H: Regression graph for Training, Validation, and Testing for DS18B20 sensor and ADXL345 sensors using Scaled Conjugate Gradient algorithm at Motor 2	128
Appendix I: Graph for Comparison of Actual and Predicted Offset for DS18B20 sensor at Motor 2	129
Appendix J: MATLAB code for Levenberg-Marquardt and Scaled Conjugate Gradient algorithm.	130
Appendix K: Python code for ADXL345 accelerometer sensor	132
Appendix L: Python code for DS18B20 temperature sensor	134



CHAPTER 1

INTRODUCTION



Generally, the main purpose of project is to develop a condition monitoring system utilizing predictive maintenance techniques. The system integrated with Raspberry Pi microprocessor, acceleration sensor, a temperature sensor, Node-RED, and MATLAB for data analysis and visualization. The objective was to enable real-time monitoring of equipment health and anticipate potential failures for proactive maintenance actions.

Nowadays, stepper motors are among the machines that are widely utilized in various fields including industrial and domestic applications that require effective detection of their status. Due to their robustness and affordability, stepping motors are frequently used as power sources in a variety of industrial applications. Yet, industrial

processes can halt when motor failures arise because of the characteristics of stepping motors.

The productivity of stepper motor can be increased by decreasing the fault circumstances. From the operation of running motor, predictive maintenance is used to take the preemptive actions. Hence, the breakdown or machine failure can be detected early and save the cost of maintenance. Thus, the aim of this study is to develop a system where it can monitor the stepper motor from being faulty while running and can give predictive information before its breakdown to meet the specification required by Texas Instruments. In the developed system, a Node-RED software will be used in the optimization process to display the data directly to a web component such as gauge or table. At the end, the data then is analyzed using MATLAB software for predictive maintenance.

1.1 Background

Machine condition monitoring system is crucial in industrial settings to ensure equipment performs well and lasts long. Stepper motors are widely used in industries because they have precise control and positioning abilities. Nevertheless, like other machine, stepper motors can wear out, deteriorate, and possibly fail over time. To overcome these challenges, predictive maintenance techniques has been a growing interest in industrial. Predictive maintenance aims to find and fix potential issues before they cause big problems. By constantly checking the condition of stepper motors, organizations can spot early signs of wear, performance issues, or upcoming failures. This proactive approach enables timely maintenance actions like lubrication, repairs, or component replacements, leading to less downtime and improved productivity. One study proposes a predictive maintenance platform for stepper

motors that monitors the running status of the motor and collects and analyzes load value and prediction [1].

Setting up a system to monitor the condition of stepper motors requires combining different technologies and approaches. This involves using sensors to real-time data on important motor measurements like temperature, vibration, current, and position. The gathered data is then analyzed using advanced methods like machine learning algorithms to detect patterns, irregularities, and possible failure scenarios unique to stepper motors [2].

The Raspberry Pi is a small and affordable computer that can be used as the main processing unit for the machine condition monitoring system. It can effectively collect, process, and analyze sensor data, giving real-time information about the motor's condition. Additionally, user-friendly tools like Node-RED allow for easy creation of monitoring workflows, data visualization, and triggering maintenance actions based on preset limits or predictive models [3].

Implementing a predictive maintenance system for stepper motors in industries brings many advantages. It helps organizations optimize maintenance schedules, reduce unplanned downtime, and extend the lifespan of critical equipment. By proactively addressing potential issues, organizations can cut maintenance costs, enhance overall operational efficiency, and ensure the reliability and safety of their production processes.

In this research, we propose developing a machine condition monitoring system that uses predictive maintenance techniques for stepper motors in industrial applications. The system aims to provide real-time monitoring, detect faults early, and

offer practical insights for maintenance staff, enabling informed decision-making and efficient resource allocation. Through experiments and data analysis, we will evaluate the system's effectiveness and feasibility, demonstrating its potential to improve the reliability and performance of stepper motors in industrial environments.

1.2 Problem Statement

The dependability and performance of a system with a low life-cycle cost have drawn a lot of attention in the engineering field. Machines are operating in more complicated contexts with greater unpredictability as industrial applications expand, raising the chance of system failure. The industry also faces a significant challenge due to the absence of effective and proactive condition monitoring methods for stepper motors. Without immediate knowledge about the motor's health and performance, it becomes challenging to foresee and mitigate potential failures. Therefore, predictive maintenance is necessary to prevent losses in maintenance and motor breakdown to ensure the operation of machinery is long lasting and durable. Hence, there is a pressing need to create a dependable and efficient predictive maintenance system tailored specifically for stepper to avoid complete failure. The potential for using condition monitoring and an open-source visual programming tool to identify issues is enormous.

Motor defect detection can be done using a visual programming tool. Stepper motor defects need to be fixed right away to prevent losses. By using a visual programming tool such as Node-RED, it is such a fantastic maintenance technique. If a problem develops, the motor may continue operating and result in winding, core, and other failures. So, defects can be avoided by keeping an eye on motor output numbers and turning off the power before the motor breaks down [4]. In order to plan maintenance

and prevent significant failure, which would be costly in terms of plant repairs and lost production time, they are especially prevalent with large induction and synchronous machines. To improve risk management in many industry applications, continuous in-service monitoring is significantly beneficial for bearing failures detection in their incipient stage [5]. Another problem statement of this project is that a rotating machine also can be a failure. This can be attributed to be one of the serious causes of breakdown in machines where they operate at high or low rotational speeds [6]. Thus, this project aims to develop a machine condition monitoring system where it can monitor the performance of the motor over time, analyzing the collected data, and using this information to approach predictive maintenance. The measured data can then be displayed on the Node-RED dashboard and can be saved in the Excel file. From the collected data in Excel, predictive maintenance can be created using the MATLAB software. Predictive maintenance is needed to minimize downtime and reduce maintenance costs while maximizing equipment availability and performance.

1.3 Project Objectives

- i. To examine and detect various types of motor abnormalities such as excessive vibration, and temperature levels.
- ii. To develop a monitoring system through the Node-RED visual programming tool using a Raspberry Pi as a microcontroller.
- iii. To develop predictive maintenance using historical data from collected sensor data.

1.4 Project Scope and Project Impact

The scope of this project is to design and develop a monitoring system for a stepper motor that runs in a tape and reel machine. The stepper motor's name is

VEXTA Stepping Motor PH569M-NBA, and it is already obsolete from the OEM. Hence, a monitoring system is developed to monitor the motor's condition to give a warning or message on the Node-RED dashboard at the same time can make a predictive analytic before motor may be faulty or overheating and take proactive maintenance actions. The data is then saved in Excel file and utilize MATLAB software for data analytic to create a prediction model using machine learning algorithms in MATLAB. The project aims to achieve the following objectives:

- i. **Hardware Integration:** Integrate Raspberry Pi as the central processing unit and connect various sensors to collect real-time data on motor performance parameters, including temperature, vibration, current, and power consumption.
- ii. **Software Development:** Utilize Node-RED, a visual programming tool, to develop a comprehensive monitoring and predictive maintenance workflow. Design and implement the necessary software modules to capture, process, and analyze sensor data.
- iii. **Data Analytics:** Apply advanced data analytics techniques, including machine learning algorithms, to identify patterns and anomalies in the sensor data. Develop algorithms for predictive maintenance, enabling early detection of degradation, wear and tear, or potential faults.
- iv. **Visualization and Reporting:** Create a user-friendly dashboard or interface to visualize the real-time and historical data related to machine condition. Generate reports and provide actionable insights for maintenance personnel.
- v. **Experimental Validation:** Conduct experiments on a real-world motor setup to validate the effectiveness and accuracy of the proposed system.

Evaluate the system's performance in accurately monitoring the motor's condition, detecting abnormalities, and providing useful insights for maintenance personnel.

- vi. Documentation and Presentation: Prepare comprehensive documentation, including system design, implementation details, and user manuals. Present the project findings, methodologies, and outcomes in a clear and concise manner.

1.5 Significance of Study

This project is of great significance to the industry as it improves equipment reliability, reduces costs, enhances operational efficiency and safety, enables data-driven decision making, and prepares for the future of industrial practices. By implementing a condition monitoring system using predictive maintenance techniques, it allows for real-time monitoring, early detection of faults, and proactive maintenance measures. This leads to reduced downtime, optimized maintenance efforts, and increased productivity. With access to data-driven insights, industry professionals can make well-informed decisions regarding maintenance, resource allocation, and equipment optimization. Moreover, the project aligns with the industry's shift towards automation and smart manufacturing, ensuring competitiveness and sustainability in the ever-changing market landscape. In summary, this project makes a substantial contribution to enhancing industry performance, safety, and long-term success.

1.6 Propose Solutions for the Project.

This project presents a comprehensive overview of the proposed solutions to address the identified challenges within the project's scope. Identified the critical

importance of enhancing motor condition monitoring and maintenance practices, the proposed solutions aim to integrate advanced technologies, data analytics, and predictive maintenance to optimize motor performance, reliability, and longevity. A detailed explanation of the proposed solution's components, methodologies, and potential impact in advancing the project's objectives and addressing the identified problem is described:

- i. Sensor integration: Integrate the ADXL345 accelerometer sensor and DS18B20 temperature sensor to detect anomalies and temperature variations in the machine's movement, providing a mechanical issue and overheating or cooling inefficiencies which may indicate operational problems.
- ii. Data acquisition and processing: Utilize Raspberry Pi as the central processing unit to gather data from the integrated sensors, facilitating a real-time monitoring and analysis of machine conditions. Implement a Node-RED to develop a visual flow-based programming interface, enabling seamless integration and communication between Raspberry Pi, sensors, and data visualization tools.
- iii. Data analysis and predictive maintenance: Utilize MATLAB for advanced data analysis, predictive maintenance modelling, and visualization, using its powerful toolboxes and capabilities to optimize machine performance and reliability.
- iv. Continuous monitoring and optimization: Implement Node-RED Dashboard to generate real-time monitoring based on the analyzed data, establish a feedback loop mechanism. To continuously monitor, evaluate,

and optimize the sensor's data based on the machine's operational data and performance metrics.

1.7 Structure of the Thesis

The thesis follows a structured approach, comprising an introduction to provide an overview of the project, a comprehensive literature review to analyze existing research and theories, a methodology section outlining the research design and data collection methods, a results and discussion section to present and analyze the findings, and a conclusion and recommendation section that summarizes the key findings and offers insights for future projects and research.

This dissertation is organized as follows:

- Chapter 2 discusses previous work and research related to machine condition monitoring and predictive maintenance. The review focuses on the use of sensors, data analytics, and machine learning algorithms to detect patterns, anomalies, and potential faults in machinery.
- Chapter 3 discusses the approach and techniques employed to achieve the research objectives. It provides a detailed description of the steps and procedures followed in implementing the proposed machine condition monitoring system using predictive maintenance.
- Chapter 4 discusses the contributions of Node-RED monitoring and MATLAB in data management and predictive maintenance. Data processing using MATLAB software for predictive maintenance to build a predictive model.
- Chapter 5 summarize the key findings and contributions of the research, emphasizing the significance in the field of motor condition monitoring. It

will provide closure to the thesis, reaffirming its relevance and setting the groundwork for advancements and innovations in motor monitoring technologies.



CHAPTER 2

BACKGROUND STUDY



The literature review section of this study examines previous research and studies related to machine condition monitoring and predictive maintenance. It explores various approaches and techniques employed in monitoring the performance and health of industrial equipment. The review focuses on the use of sensors, data analytics, and machine learning algorithms to detect patterns, anomalies, and potential faults in machinery. By analyzing existing literature, this study aims to identify gaps and contribute to the existing knowledge by proposing a novel system for machine condition monitoring using predictive maintenance.

2.1 Background Literature Review

In recent years, there has been increasing interest in the utilization of condition monitoring and predictive maintenance as valuable approaches to enhance asset

management efficiency and minimize operational expenses. These practices have proven the importance in diverse domains such as structural health monitoring, industrial equipment maintenance, and environmental monitoring. Moreover, the advancement of microcomputer technology has introduced the Raspberry Pi as an economical and adaptable platform for the creation of vibration monitoring systems. The objective of this literature review is to offer a comprehensive summary of the current research on condition monitoring methods that incorporate predictive maintenance in diverse industries. Through an examination of numerous articles, this review seeks to uncover the primary techniques, technologies, and applications utilized in predictive maintenance for effective condition monitoring.

2.2 Condition Monitoring (CM)

Previous study from Kumari Sarita et al. (2021), had reported the principle of component analysis technique for early fault detection. The paper highlighted an unsupervised statistical algorithm based on principal component analysis (PCA) for the predictive maintenance of industrial induced draft (ID) fan. However, the paper only concentrates on three identical ID fans that are monitored together using the proposed technique which is PCA technique and fast fourier transform (FFT) technique. Therefore, my study will emphasize on using this paper method in using the proposed technique for early prediction of the faulty part to help in forecasting the maintenance schedule for the equipment before breakdown [4].

Studies from research Grace, R. K., & Subhasri, V. P. (2022), had presented a cloud enabled predictive maintenance tool for induction motor. The paper highlights are to predict the failure of an induction motor and to schedule the preventive maintenance. However, the paper only concentrates on utilizing predictive maintenance to anticipate

failures in induction motors. Therefore, my study will target on using this paper method by using cloud storage advancements that enable data collection from various sources including sensors, to monitor motor performance [7].

Ekkawach Noyjeen et al. (2021) presented a monitoring parameters of three-phase induction motor using IoT. The research emphasized the design of IoT technology to monitor and diagnose the performance of a three-phase induction motor and recording critical operating parameters. However, the paper only focuses on a design that utilizes IoT to collect and analyze critical motor parameters such as voltage, current, temperature, and vibration. The data is stored in the cloud and can be accessed through web pages and displayed on smartphones using the MIT application. Therefore, my study will emphasize on using this paper approach where fault alerts can be notified timely and the availability of historical data for predictive maintenance, resulting in reduced downtime and cost savings [8].

Next, a number of researchers have sought to propose that load fault diagnosis in induction motor using Artificial Intelligence algorithm. The paper highlighted developments of a machine learning strategy based on algorithms in order to learn the characteristics from vibration signal's frequency distribution. However, the paper only focuses on ANN-based fault diagnosis system for Induction Motors since there is not enough data for artificial intelligence (AI) to identify problems. Therefore, my study will focus on using this paper method to create fault detection techniques where several parameters are measure such as temperature, voltage, and current [9].

Research such as that conducted by Guilherme Beraldi Lucas et al. (2006) stated that sensor applied to bearing fault detection in three-phase induction motors. The paper highlighted to present the state of the art of the past five years concerning the

sensing techniques based on current, vibration, and infra-red analysis, which are characterized as promising tools to perform bearing fault detection. However, the paper only focuses on the bearing fault models for current, vibration, and infrared sensors. Therefore, my study will targets on using this paper method in using the current and vibration frequencies of the bearing fault mathematical modeling for these techniques [10].

In 2022, G Aragón González et al. published a paper which they reported a paper about remote control and monitoring of a hydraulic machine. The paper underlines a deal with the testing of a low-cost electronic system based on an open-source programmable board, designed for remote control and monitoring of an industrial machine. Even so, the paper only focused on designing an electronic system based on an open-source programmable board that was tested for remote control and monitoring of a fluid power machine. Therefore, my study will focus on using this paper method on creating a code to process the information received from sensors and send it to user interface to make logical decisions and process the information from the control orders to govern the stepper motor [11].

In the previous years, a number of researchers have sought to propose that non-contact condition monitoring of electrical drive. The paper highlighted a non-contact technology for conditional monitoring of an intelligent electric drive. The study is relevant to the field of smart manufacturing, which involves a continuous reduction in operating costs and the development of intelligent control systems for the condition monitoring of equipment. However, the paper only highlights on implementation using universal software, which permits to reduce production costs for condition

monitoring. Therefore, my study will be using this paper method for condition monitoring of electric motors [12].

Research study that conducted by N. Dehbashi et al. (2020) report a study of IoT Based Condition Monitoring and Control of Induction Motor Using Raspberry Pi. The paper highlighted the exploration use IoT in the control and monitoring of electric motors, which leads to the formation of smart drives. The data is collected by the Raspberry Pi and sent to the server, where it is stored in the database. However, the paper only focuses on save the process data and sent back to the Raspberry Pi. Therefore, my study will engage on using this paper method by using Raspberry Pi board as processor and measure temperature for detecting failures [13].

Next, a report by Hong-Chan Chang et al. presented that proactive operation condition monitoring system of high-voltage motors based on CNN and LSTM. The paper highlighted to develop an active monitoring system for high-voltage motor operation status based on a convolutional neural network and long short-term memory neural network. However, the paper only concentrates on predict the high-voltage motor operation by subtracting the maximum error value from the international threshold value. Therefore, my study will be using this paper method in determine the maximum error between the predicted and actual operation status [14].

Other studies related to this project is reported by Agam Gugaliya et al. (2022) that presented availability improvement of induction motors through condition monitoring. The paper highlighted to improve the availability of induction motors by reducing repair/replacement time and increasing the efficiency of maintenance. However, the paper only emphasizes on use fault diagnostic technique to find the fault at any point in P-F interval (potential failure to functional failure). Therefore, my study will be

using this paper method to monitor the motor's condition because early detection of potential failure may give maintenance managers sufficient time to arrange the necessary equipment for system maintenance [15].

Furthermore, study from Keyur V. Surti et al. (2018) stated that availability improvement of induction motors through condition monitoring. The paper highlighted to improve the availability of induction motors by reducing repair/replacement time and increasing the efficiency of maintenance. However, the paper only focuses on validate with real voltage and current signals acquired using laboratory experimental setup for a healthy and two unhealthy bearings. Therefore, my study will be using this paper method to differentiate between the unhealthy and healthy of motor condition [16].

Ramakrishnan Raman et al. (2023) presented that smart industrial motor monitoring with IoT enabled Photovoltaic system. The paper highlighted a new idea for combining a photovoltaic (PV) system with an induction motor (IM), together with IoT based monitoring technologies. However, the paper only focus on integrates the power-producing capabilities of the PV system with monitoring sensors provided by the IoT to deal with the drawbacks of conventional power sources. Therefore, my study will be using this paper method to have a real-time information on the motor's temperature and vibration is gathered by the IoT based monitoring devices [17].

Other than that, studies by Om Prakash Choudhary et al. (2021), presented that IoT enabled condition monitoring of low-voltage motors using Fuzzy Inference system. The paper highlighted is to develop a real-time condition monitoring prototype of a low voltage industrial motor. However, the paper only concentrate on using Zigbee module for sending data wirelessly to the remote end using IoT and then fed from

Thingspeak server to the Fuzzy Inference System (FIS) to estimate the health of the motor. Therefore, my study will focus on using this paper method to monitor vital parameters, temperature and vibration using suitable sensors connected to microcontroller [18].

Many studies conducted by Zayneb Bousselmi et al. (2021) reported that wireless IoT approach for testing is situ motor's axis vibration monitoring. The paper highlights is to develop a workbench for remote control of rotating machines that can predict the maintenance of a direct current (DC) motor in an efficient way. However, the paper only focuses on utilizing Allan's variance technique to accomplish detection, definition, and localization of vibration signatures. Therefore, my study will emphasize on using this paper method by carrying out a large set of measurements of the acceleration signal, derived from MEMS accelerometer sensor placed on the rod axis [19].

Research that has been conducted by Michal Markiewicz et al. (2019) reported that predictive maintenance of induction motors using ultra-low power wireless sensors and compressed recurrent neural networks. The paper highlighted a novel architecture for machinery monitoring in real-world applications. However, the paper only focuses on architecture involves moving the processing to the sensors themselves, which is made possible by using compressed recurrent neural networks. Therefore, my study will focus on using this paper method to process data locally and wirelessly send only a single packet with the probability that the machine is working incorrectly [20].

2.3 Vibration and Temperature Data

Other studies related to this project is reported by L. Magadan et al. (2022) that presented a low-cost industrial IoT system for wireless monitoring of electric motors condition. The paper highlighted the design, implementation, and testing of a low-cost

Industrial of Things (IoT) system designed to monitor electric motors in real-time. However, the paper only focusses on collecting vibration data from electric motors. Therefore, my study will emphasize on using this paper method where the proposed system of this paper can be used to monitor of any rotary machine with similar accuracy to monitor devices in a professional way but in a low-cost [21].

Furthermore, study from Dileep Kumar et al. (2022) reported the triaxial bearing vibration dataset of induction motor under varying load conditions. The paper highlighted is discussion on triaxial vibration data for moto bearing faults detection and identification. However, this paper only focusses on the monitoring of rotating machines in industries through the analysis of vibrations. It presents findings on the detection and identification of motor bearing faults using triaxial vibration data collected using a MEMS accelerometer and the National Instruments myRIO board. The dataset includes different bearing conditions and load conditions, offering a means to evaluate the performance of novel approaches for effective bearing fault diagnosis. Therefore, my study will focus on using this paper method where the collected data is stored in separated values CSV files or Excel file [22].

Studies conducted by Ioan Szabo et al. (2021), a vibration and temperature sensor network solutions: case study for industry 4.0 had been reported. The study is about a new predictive procedure for industrial ventilation installation and industrial testing equipment. However, the paper only highlighted the procedure involves digital signal processing of temperature and vibration sensor data to assess the operating regime and technical condition of the equipment. The tests conducted on ventilation devices and industrial gearbox testing equipment successfully identified faulty bearings through data analysis. Therefore, my study will concentrate on using this paper method where

a real-time data can be accessed through a cloud platform, and the solution is cost-effective due to the utilization of open-source projects [23].

Research study that conducted by Khandelwal et al. (2022) report a study on sensor-based vibration analysis of motor using MATLAB software. The paper highlighted a real-time graph monitoring of engine vibration and analyze to identify faults such as wear errors. However, this paper only focusses on using Arduino as microcontroller and MATLAB software for real-time monitoring. Therefore, my study will highlight on using this paper method where various parameters were measured by using various sensors that fitted to the motor as well can improved the efficiency of motor and using MATLAB software for data preprocessing [24].

In previous studies of J.M. Corres et al. (2006), different parameters have been used to be related to vibration of motor. The study is about vibration monitoring in electrical engines using an in-line fiber etalon. The paper highlighted a proposed sensor that has been optimized to achieve minimum detectable acceleration amplitude of 0.05 g in the frequency range of interest, although the preliminary scheme can be customized to the specific motion system. However, the paper only highlighted on new applications to electric machines condition monitoring (CM) using an ILFE fiber optic sensor. therefore, my study will focus on using this paper method where the sensor can do a predictive maintenance not only includes mechanical defects, because due to high sensitivity that can also be obtained in the low frequency range [5].

Previous study from A. Holovatyy et al. (2017) had proposed the development of a system monitoring vibration accelerations based on the Raspberry Pi microcomputer and the ADXL345 accelerometer. The paper highlighted the structure and the algorithm of functioning of the system for monitoring. However, the paper only

emphasizes on analyzing the vibration of acceleration spectrum which operates in a real-time mode. Therefore, my study will focus on using this paper method to reads data from the accelerometer, processes them which convert the vibration acceleration signal obtained from the sensor from the time domain to the frequency domain by using software [25].

Marek Iwaniec et al. (2017) had reported the development of vibration spectrum analyzer using the Raspberry Pi microcomputer and 3-axis digital MEMS accelerometer ADXL345. The paper highlighted about spectrum analyzer of vibration accelerations using the Raspberry Pi microcomputer and 3-axis digital MEMS ADXL345 accelerometer has been developed. However, this paper only concentrates on data acquiring using software and processing from the acceleration sensor, the conversion of the vibration acceleration signals from time domain into frequency domain using discrete fourier transform (DFT), graph plotting of the vibration accelerations and their spectra. Therefore, my study will focus on using this paper method for development of automated monitoring and control system the single-board microprocessor [26].

Other researcher also makes a study about development of fault diagnosis unit for induction motor using digital signal processor and MEMS accelerometer written by Vishwanath hegde and Maruthi G. S. (2019). The paper highlighted the development of a portable fault diagnosis unit for Induction Motors (IM) using a digital signal processor and a MEMS accelerometer. However, the paper only emphasized on detection various electrical and mechanical faults in IMs, enabling continuous condition monitoring. Therefore, my study will focus on using this paper method involves capturing vibration signals using an accelerometer, converting them to

voltage signals, and performing spectral analysis through FFT on a digital signal processor [27].

Agusmian Partogi Ompusunggu et al. (2021) presented a condition monitoring of critical industrial assets using high performing low-cost MEMS accelerometers. The paper highlighted experiences in setting up two different remote vibration monitoring systems using low-cost MEMS accelerometers available on the market in two different industrial settings. However, the paper only focusses on demonstrations on the use of low-cost MEMS accelerometer for long-term condition monitoring of critical assets (induction motors as an expander machine) in two different industrial settings. Therefore, my study will highlight on using this paper method where the low-cost MEMS accelerometers are being used for monitoring the assets [28].

Research study that conducted by Chhaya Devi R. Sahu et al. (2021), report a study on integration of machine learning and IoT system for monitoring parameters and optimizing farming. The paper's approach is in hydroponic farming, leveraging Internet of Things (IoT) and Machine Learning technologies. Central to this system is the DS18B20 temperature sensor, interfaced with an ESP8266 Wi-fi module for real-time monitoring. Through the ThingSpeak IoT platform, the sensor's data contributes to optimizing plant growth conditions, enhancing resource utilization and fostering sustainable agriculture practices in both urban and rural settings. However, this paper only focuses on storing all data from various sensors in a text file which is converted into a CSV file for machine learning algorithm and also this text file is sent to Thingspeak server for monitoring purpose which aggregate, visualize and analyze live data streams in the cloud. Therefore, my study will focus on using this paper method

to utilize a DS18B20 temperature sensor to measure the temperature of motor which it has an accuracy of about $\pm 5^{\circ}\text{C}$ [29].

2.4 Node-RED Visual Programming Tool

Other than that, the study that related to this project is from Susmita Banerjee et al. (2020) entitled IoT instrumental food and grain warehouse traceability system for farmers. This paper highlighted about the proposal of an IoT enabled monitoring system for remote areas in India to reduce food losses and increase food safety. However, this paper only focusses on system monitors warehouse parameters such as temperature, humidity, CO, motion, vibration, and smoke, which are critical for preserving grains. Therefore, my study will focus on using this paper approach where the data from sensors is collected by the ESP32 WiFi module and transmitted to the Node-red dashboard through an MQTT broker. Multiple IoT nodes installed at various locations at the motor provide information about the condition through mobile SMS and email notifications [30].

Research that conducted by Bhavana et al. (2023), stated that high security alert system for industrial atmospheric parameters. The highlighted topic of this study is the proposal of a cost and power-efficient real-time monitoring system for industrial processes. The system utilizes sensors to measure parameters such as temperature, gas, smoke, and fire, and transmits the data to a microprocessor and mobile device via a Raspberry Pi module and GSM module. However, the paper only focus on the system incorporates Wireless Sensor Network (WSN) and IoT technologies for wireless communication and cloud upload, enabling remote monitoring and reducing the need for human intervention. Therefore, my study will focus on using this paper method where the collected data display on node-red dashboard for monitoring [31].

A number of researchers have reported that monitoring system for elderly health care using smart band, Raspberry Pi, and Node-RED. The topic highlighted in this study is the development of a system for monitoring the health conditions of multiple elderly individuals and sending alerts during abnormal conditions based on heart rate values. Nevertheless, the paper only concentrates on the limitation of storing user data in the vendor's cloud and allows simultaneous data analysis and alerts for nurses, doctors, and family members. Therefore, my study will focus on using this paper method to give notification via any platform to users where in this paper, a smart band is used to monitor vital signs and human activity. The system includes a local database on Raspberry Pi and a monitoring system on the Node-RED platform to display health data. When an abnormal heart rate is detected, notifications are sent via Telegram to multiple registered receivers [32].

Other researchers have made a study that related to this project about environmental parameter monitoring system based on NodeMCU ESP8266, MQTT and Node-RED by Macheso et al. (2022). This paper highlighted about the utilization of IoT in smart environmental monitoring. However, the paper only focusses on using the system communication protocols like MQTT and end sensor nodes to connect to the internet. The central units consist of the ESP8266 NodeMCU microcontroller board, DHT22 sensor, and Raspberry Pi with an MQTT broker. Therefore, my study will focus on using this paper method where the system collects environmental parameters such as air temperature and humidity through sensing and visualizes the sensor data on a Node-Red Dashboard [33].

Recent studies from A. Sinan Cabuk (2022) have reported that experimental IoT study on fault detection and preventive apparatus using Node-RED ship's main engine

cooling water pump motor. The paper highlighted the thermal, vibration and current data of the 7.5 kW 3-phase induction motor in the cooling pump used for the cooling system of this ship's main engine were analyzed and the data received were monitored by node-RED. However, the paper only focusses on an architecture that receives instantaneous data from sensors, transfers them to the internet via electronic circuit, and transfers them to users via the dashboard and stores these data on MySQL. Therefore, my study will emphasize on using this paper method in analyzing data using node-RED with the IoT sensor data [34].

A number of authors from Alaa Abdulhady Jaber et al. (2014), have reported on the state of the art in research into the condition monitoring of industrial machinery. The paper highlighted the basic concepts of condition monitoring and introduced the necessary background information about the various condition monitoring technologies used for different types of machines. However, the paper only focusses on a remote monitoring system for a rotating machine that can be run based on smartphone or PAD (personal digital assistant). Therefore, my study will highlight on using this paper method where the developers put the capability of informing the concerned user if a fault appears in the remotely monitored machine using Node-RED dashboard [35].

2.5 Predictive Maintenance using MATLAB apps.

In 2021, Mohamed Hayouni et al. published a paper which they reported a paper about wireless IoT approach for testing in situ motor's axis vibration monitoring. The paper underlines a development of workbench for remote control of rotating machines to make a predictive maintenance of a DC motor in efficient way. However, the paper only prioritize on the reliability of sensor systems and the proposed experimental

bench is tested through extensive measurements using a MEMS accelerometer sensor placed on the motor's rod axis. Various vibration signals with different characteristics are injected to assess the devices' reliability. The successful detection, definition, and localization of vibration signatures are achieved using Allan's variance technique. Therefore, my study will focus on using this paper method where a large set of measurements of the acceleration signal is derived from the MEMS accelerometer sensor that placed on the motor and use software for predictive analytic [36].

2.6 Comparison between technique used and parameters considered on literature.

Table 2. 1 represents the comparison on different techniques for prediction of fault on motor in the literature.

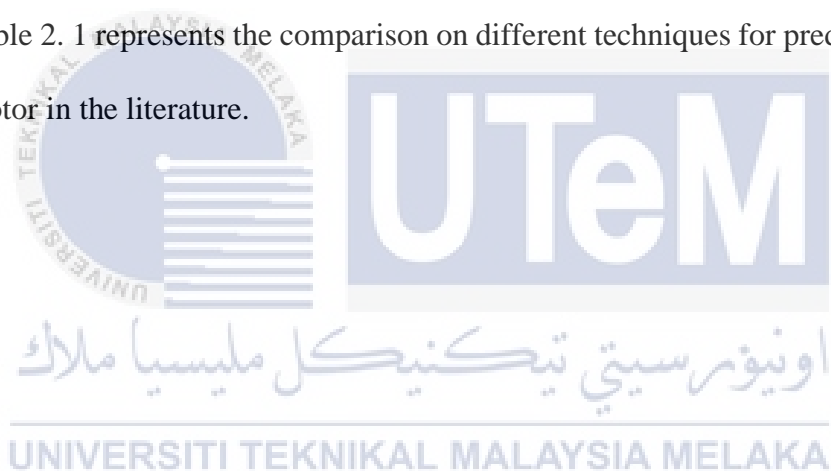


Table 2. 1: Comparison on Literature

No	Technique Used	Parameters Considered	Efficiency
1.	IoT with art of cloud storage [7]	Vibration, voltage, current, and speed	90% of prediction from machine failure
2.	Using a non-contact technology [12]	Current, voltage, and rotational speed	Accuracy is high
3.	IoT with PWM [13]	Temperature, current, and voltage signals	Accuracy is high
4.	IoT with CNN and LSTM [14]	Voltage, current, vibration, and temperature signals	92% of accuracy
5.	IoT with Condition Based Maintenance (CBM) [15]	Vibration	92.3% of accuracy
6.	IoT with K-Nearest Neighbors Based Classifier (KNN) [16]	Current and voltage signals	90.48% of accuracy
7.	IoT with Photovoltaic (PV) and Induction Motor (IM) [17]	Temperature, vibration, and power consumption	15% went up for efficiency

8.	IoT with Fuzzy Inference System (FIS) [18]	Temperature, vibration, and speed	91% of accuracy
9.	IoT with Allan's variance technique [19]	Acceleration Signal	Accuracy is high
10.	IoT with RNN [20]	Accuracy, F1-Score	91.97% of accuracy
11.	Condition monitoring technique [28]	Vibration	Accuracy is medium
12.	IoT with Naïve Bayes classifier [36]	Vibration	93.2% of accuracy-Load 100% of accuracy-Nonload.

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 3

METHODOLOGY



The methodology section of this study outlines the approach and techniques employed to achieve the research objectives. It provides a detailed description of the steps and procedures followed in implementing the proposed machine condition monitoring system using predictive maintenance. The section begins by explaining the hardware integration, including the use of the Raspberry Pi as the central processing unit and the connection of various sensors to collect real-time data. Next, the software development process is discussed, highlighting the utilization of Node-RED as a visual programming tool and the development of software modules for data capture, processing, and analysis. The section further explores the application of advanced data analytics techniques, such as machine learning algorithms, to identify patterns and anomalies in the sensor data. The methodology section concludes by describing the experimental validation process, where real-world motor setups are utilized to

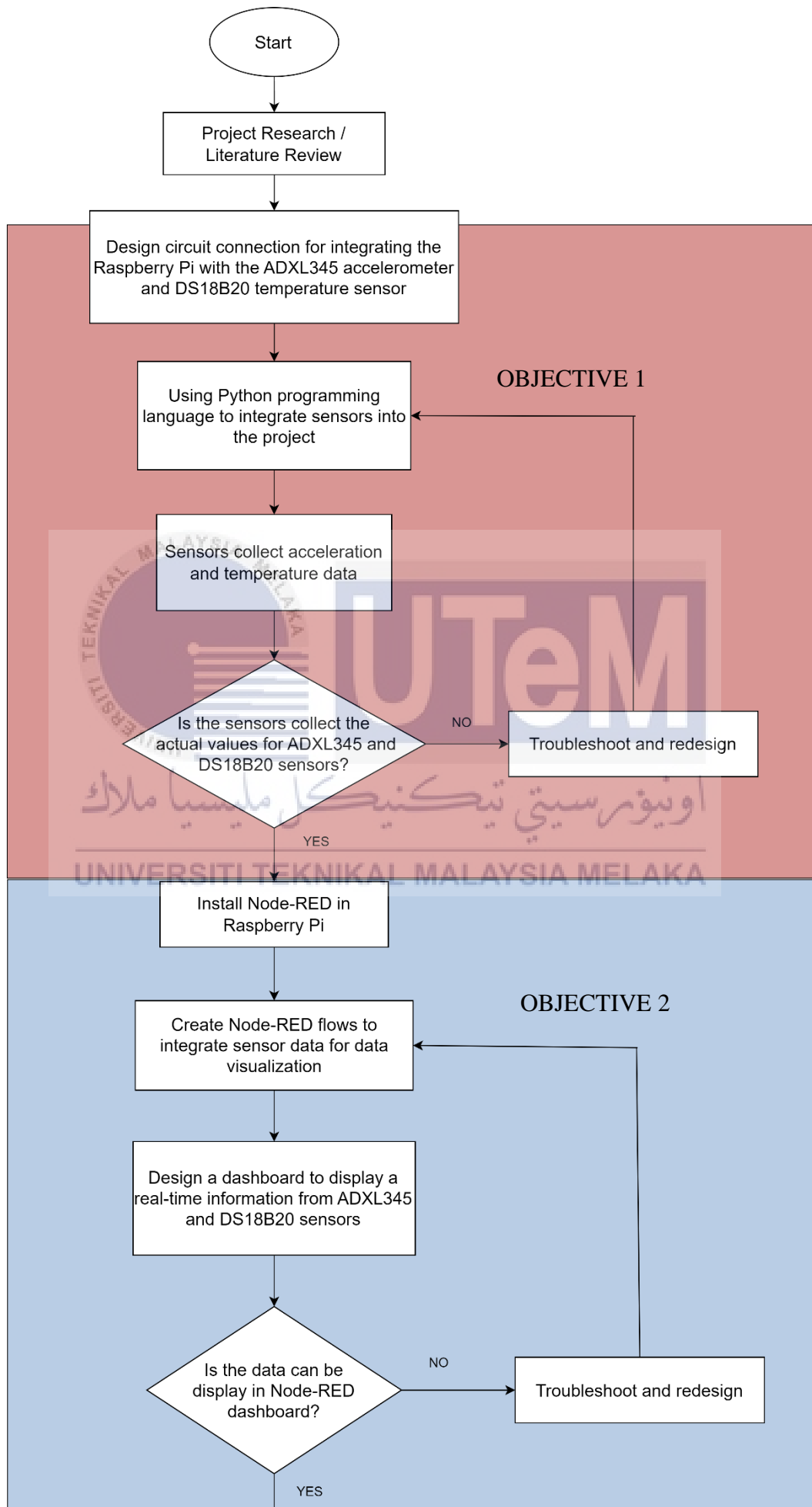
evaluate the system's performance and effectiveness in accurately monitoring the motor's condition and providing valuable insights for maintenance personnel.

3.1 General Methodology

The general methodology for this project involves several key steps. Firstly, a literature review will be conducted to gather relevant information from journals, articles, and other materials related to the project. Following that, measurements will be taken at the Lab Research facility where the machine's motor is located to investigate the suspected source of vibration at Texas Instrumental Sdn. Bhd. for prototype measurement design. Subsequently, a simulation will be performed based on the data obtained from the parameter measurements. The analysis of the collected data will be presented, focusing on the detection of motor abnormalities, and proposing possible solutions. Finally, a comprehensive report summarizing the entire study will be written upon the project's completion.

3.2 Flowchart

Figure 3.1 shows the flowchart for PSM. The flowchart outlines a comprehensive approach to address motor abnormalities and deliver actionable insights for maintenance and optimization purposes through problem analysis, feasibility assessment, method selection, component selection, project design, code integration, prototype fabrication, testing, and data analysis.



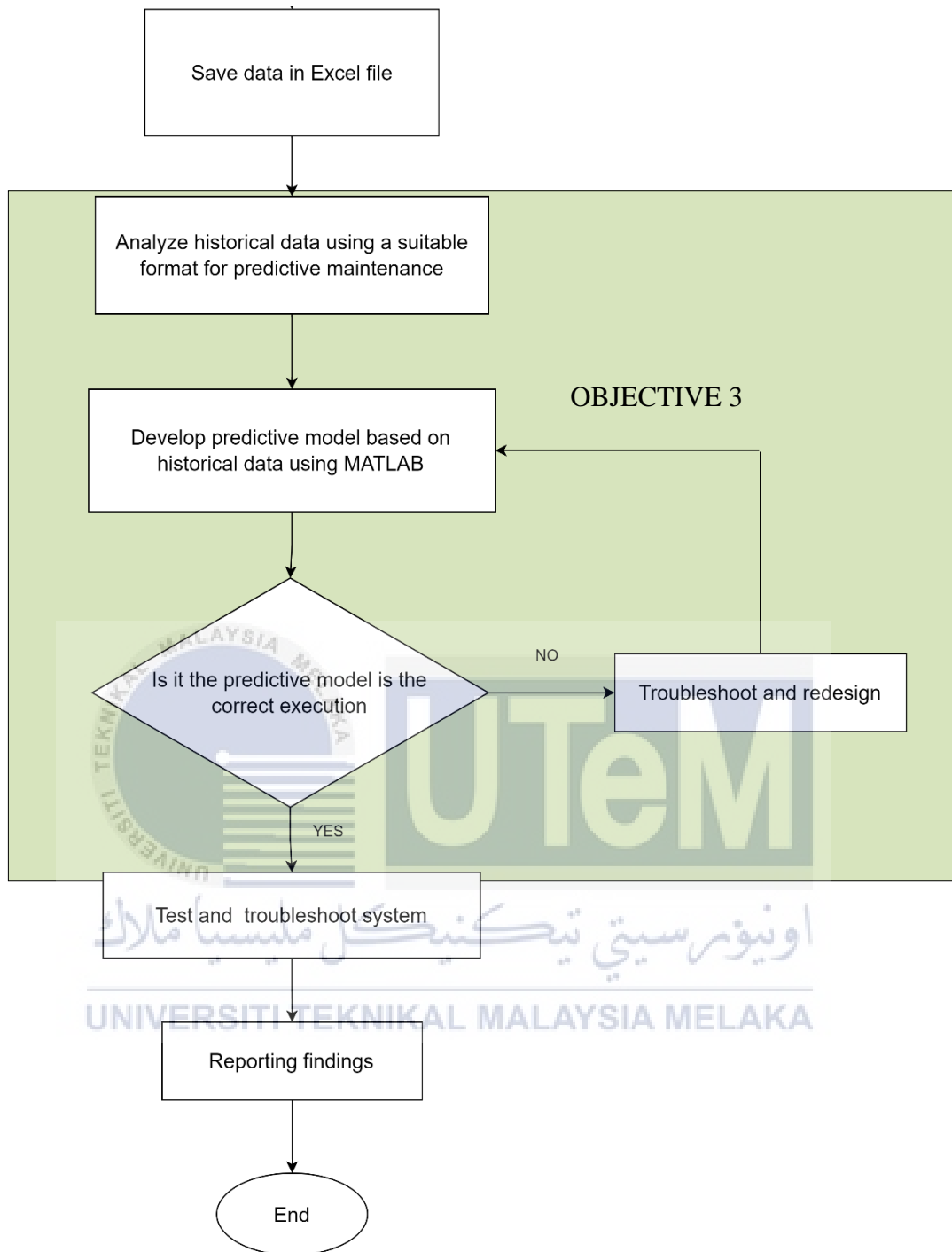


Figure 3.1: Flowchart to achieve objectives.

Figure 3.1 illustrates the flowchart to achieve the objectives. The flowchart delineates the organized sequence of the project, commencing with a comprehensive literature review to amass pertinent information. It subsequently advances to crafting circuit connections, wherein the Raspberry Pi is seamlessly integrated with the ADXL345 accelerometer and DS18B20 temperature sensor using Python programming. The sensors adeptly gather acceleration and temperature data, prompting a validation step to ensure the precision of data acquisition.

The flow unfolds through phases of troubleshooting and redesign, accommodating adjustments as required. Subsequently, Node-RED is installed on the Raspberry Pi, and flows are constructed to amalgamate sensor data for real-time visualization. A purposeful dashboard is devised to exhibit information from the ADXL345 and DS18B20 sensors. A validation check ensures the successful display of data in the Node-RED dashboard, instigating further troubleshooting and redesign endeavors if necessary.

Following this stage, the data is stored in an Excel file for meticulous historical analysis. The historical data undergoes scrutiny utilizing an appropriate format for predictive maintenance, culminating in the development of a predictive model using MATLAB. Validation ensues to guarantee the accurate execution of the predictive model, with additional troubleshooting and redesign considerations as deemed necessary.

The flowchart then advances to the testing and troubleshooting of the entire system, ultimately culminating in the reporting of findings and the project's conclusion. Each step is meticulously detailed to ensure a methodical and comprehensive approach to the project's evolution and implementation.

3.2.1 Flowchart of Integration Circuit

Figure 3.2 shows the flowchart of integration circuit between Raspberry Pi, ADXL345 sensor, and DS18B20 temperature data.

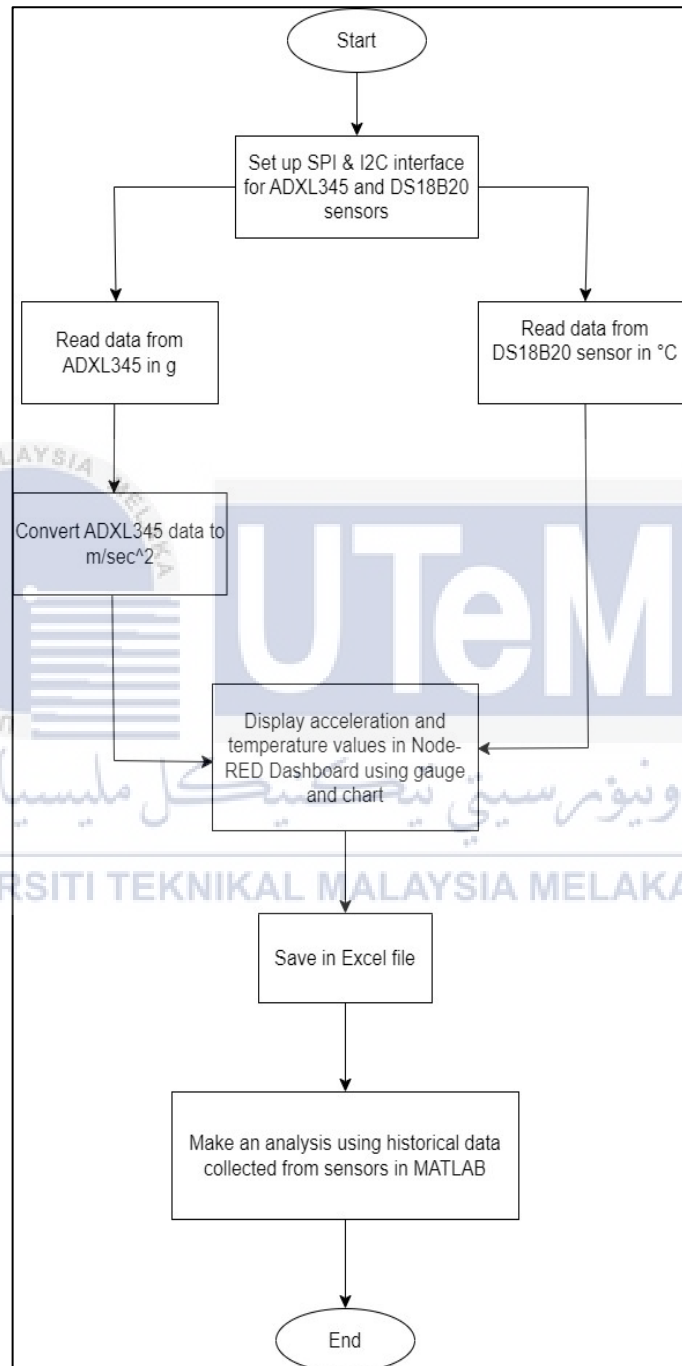


Figure 3.2: Flowchart of integration circuit

The flowchart illustrates the seamless integration of various components, including the Raspberry Pi, ADXL345 sensor, DS18B20 sensor, Node-RED dashboard, and Excel file, in order to establish a robust and comprehensive system for predictive maintenance. The Raspberry Pi serves as the central hub, orchestrating the communication and data flow between the different elements. Through the I2C protocol, the Raspberry Pi interacts with the ADXL345 sensor to capture acceleration data, while the DS18B20 sensor provides temperature measurements.

Moving forward in the flowchart, the acquired data enters the processing and visualization stage. Here, the Raspberry Pi employs suitable algorithms to process the sensor readings and formats them for display on the Node-RED dashboard. This real-time visualization empowers users to monitor and analyze the sensor data effectively, providing valuable insights into the performance and condition of the monitored system. Simultaneously, the collected data is stored in an Excel file, serving as a valuable repository for historical analysis. This historical data facilitates predictive maintenance by enabling the identification of patterns, trends, and abnormalities that can inform timely maintenance interventions.

By integrating these components, the flowchart showcases a holistic approach to predictive maintenance, encompassing data acquisition, processing, visualization, and storage. This comprehensive system empowers users to monitor the health and performance of the system in real-time while leveraging historical data to identify potential issues and optimize maintenance activities. The seamless integration of the Raspberry Pi, ADXL345 sensor, DS18B20 sensor, Node-RED dashboard, and Excel file underscores the power of technology in enabling efficient and proactive maintenance practices, ultimately leading to improved operational efficiency and

reduced downtime. The data is then processed using MATLAB software for predictive maintenance.

3.3 Modern tools for enhanced project performance

The modern tools used for this project to enhance project performance encompass a diverse range of technologies and methodologies aimed at improving efficiency, collaboration, and overall project outcomes.

3.3.1 Circuit integration between Raspberry Pi, ADXL345 accelerometer sensor, and DS18B20 temperature digital sensor module

The integration of sensors with computational platforms like the Raspberry Pi represents a fundamental integration of hardware and software methodologies. The discussion delves into the intricacies of connecting two distinct sensors: the ADXL345 accelerometer and the DS18B20 temperature sensor. Referring to Figure 3.3, a circuit integration between a Raspberry Pi, ADXL345 accelerometer sensor, and DS18B20 temperature digital sensor module, was set up by connecting the ADXL345's VCC pin to the Raspberry Pi's 3.3V output, its GND pin to the Pi's ground, and its SDA and SCL pins to the respective I2C GPIO pins on the Pi, typically GPIO2 for SDA and GPIO3 for SCL. For the DS18B20, connect its VCC pin to the Pi's 3.3V, the GND pin to ground, and its data pin to a GPIO pin, like GPIO4. Crucially, to ensure reliable communication on the 1-Wire bus for the DS18B20, integrate a 4.7k Ω pull-up resistor between its data pin and VCC. Once the physical connections are established, the I2C and 1-Wire interfaces, were enable by configure the Raspberry Pi's software settings via 'raspi-config'. Additionally, relevant libraries such as 'w1thermsensor' was installed to facilitate data reading from the DS18B20 and ADXL345 sensors.

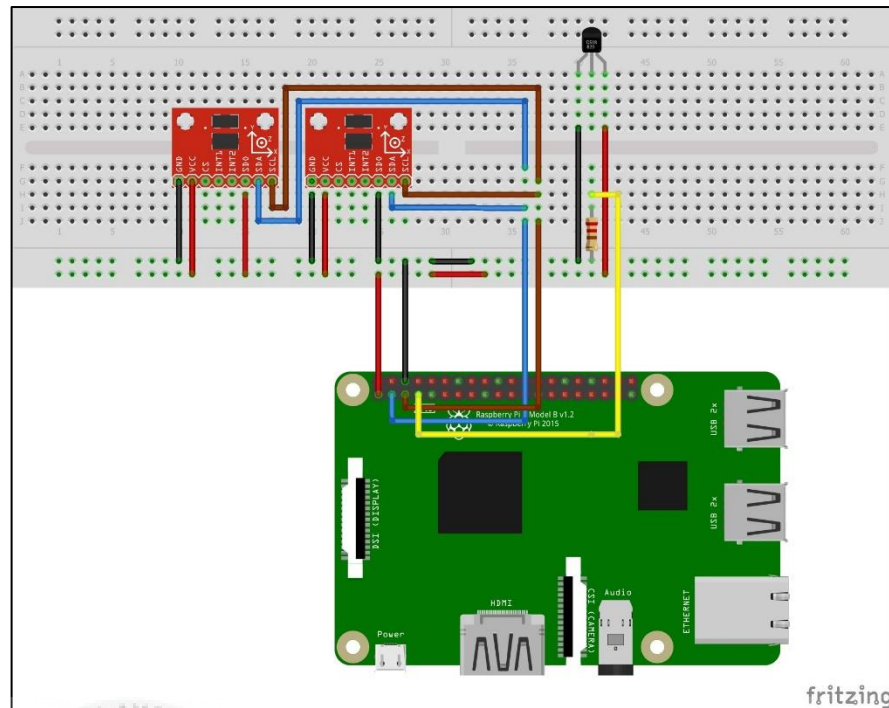


Figure 3.3: Circuit connection between Raspberry Pi, ADXL345 accelerometer sensor, and DS18B20 temperature sensor

After completion of circuit integration, the software development begins by write file in terminal of Raspberry Pi. In the exploration of sensor integration with the Raspberry Pi, Python emerges as an instrumental tool, bridging the hardware and software domains with precision. Consider the ADXL345 accelerometer: Python, with the aid of the ``adafruit-circuitpython-adxl34x`` library, facilitates detailed configuration, encompassing aspects such as the I2C address and data rate. Likewise, the DS18B20 temperature sensor finds its integration with Python, courtesy of the ``w1thermsensor`` library, ensuring calibrated temperature readings. As these sensors begin their data distribution, Python's role becomes important—it serves as the medium for data acquisition, interpretation, and analysis. Whether recognizing dynamics with the ADXL345 or conducting ambient temperature assessments via the DS18B20, Python orchestrates these operations with commendable efficiency. Furthermore, the integration concludes in a user-friendly interface; a concise

command within the Raspberry Pi's terminal initiates the Python script, thereby enabling a comprehensive exploration of the acquired data. This symbiotic combination of hardware interfacing and software orchestration underscores the complexity and promise capability in contemporary sensor integrations with the Raspberry Pi platform.

3.3.2 Placement of sensors at motors

In this project, a test rig comprising of a three-phase induction motor manufactured in year 2020 and ED-5311 Auto Driving Unit motor manufactured in year 2007 was utilized. Figure 3.4 shows the setup used for bench marking the selected low-cost MEMS accelerometers (ADXL345) and DS18B20 temperature sensor. On this gearbox, the three sensors are mounted on the same side of the motor housing for three days of collection data, with the same measuring axis orientation. The sensors are mounted on the motor for four hours straight and the data is saved for further analysis.

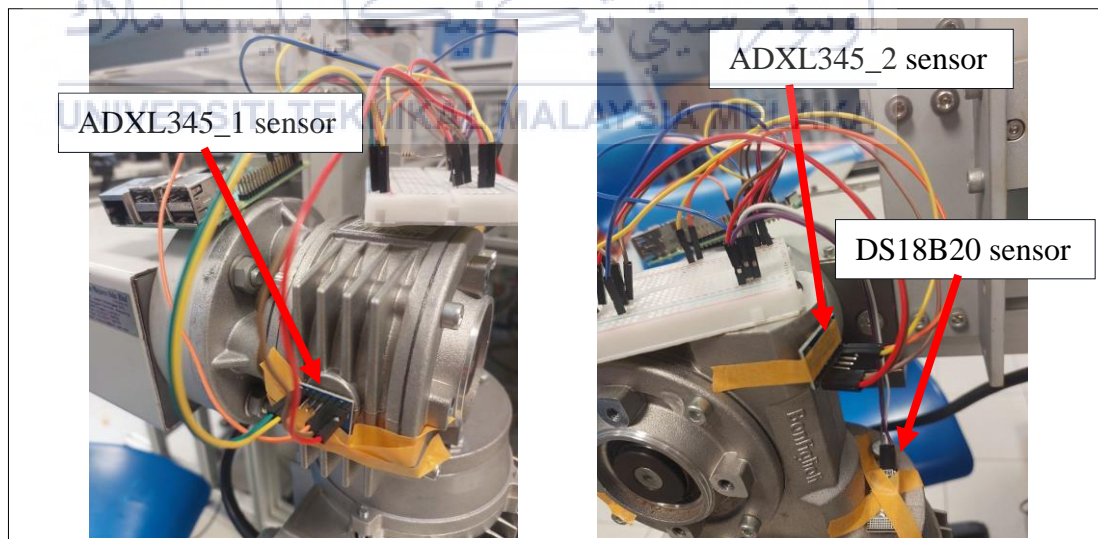


Figure 3.4: Sensor placement at Motor 1

Figure 3.5 shows the sensor placement on Motor 2. The data is collected for three days for a comparison between data collection on Motor 1 and Motor 2. The vibration

signals and temperature signal of the three sensors were acquired for four days. Then, the data measured by the sensors is then displayed in Node-RED dashboard. From there, can be seen vibration and temperature data that have been collected for four hours over three days.

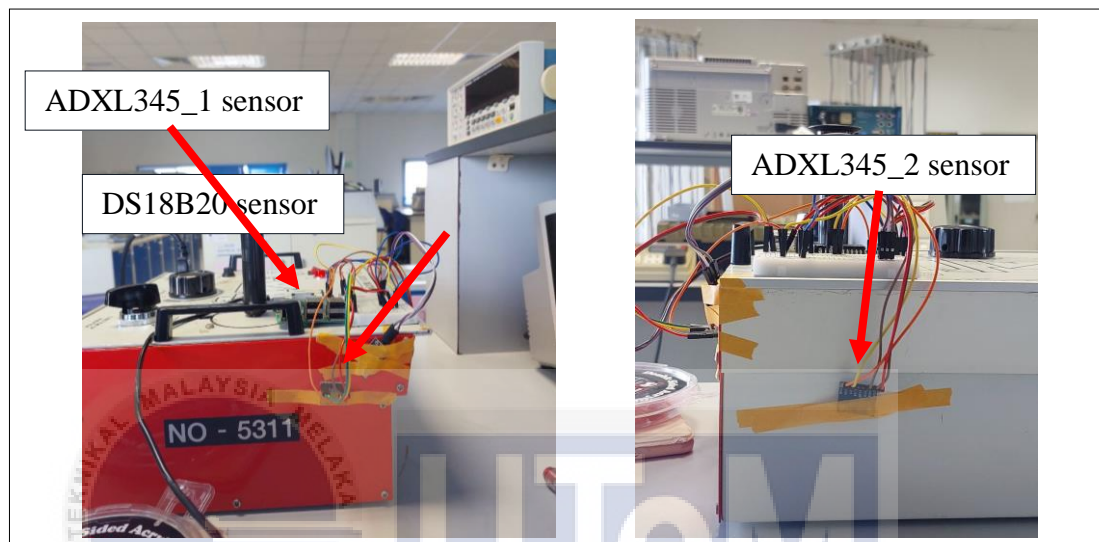


Figure 3.5: Sensor placement at Motor 2

3.3.3 Node-RED development tool

Node-RED is an open-source development environment, ideal for IoT systems. It enables the creation of data streams by connecting hardware and software components. Node-RED is a free JavaScript-based tool, built on Node.js platform, offering a graphical web-based editor for creating flows. Within this system, nodes are represented with specific icons. Users can interact by dragging, dropping, and connecting nodes or by importing JavaScript scripts [37]. Node-RED simplifies data processing and allows for easy compilation of logic and transfer of processed data to various systems or immediate display. It provides a Dashboard interface that eliminates the need for HTML and CSS knowledge when creating visually appealing

interfaces. The platform offers a wide range of nodes with different functions and uses JSON objects to store created processes.

Node-RED is flexible, facilitates real-time application development, and is excellent for prototyping. It provides the industry with the opportunity to build and test prototypes before committing significant resources to innovation. Node-RED's ease of use and cost-effectiveness set it apart from other software options like TIA Portal and Sysmac Studio, which have steeper learning curves and expensive licenses [38].

3.3.3.1 Node-RED applications

Node-RED Dashboard stands out as an exceptional option for presenting data collected from a Raspberry Pi, ADXL345 sensor, and DS18B20 digital temperature sensor module due to its user-friendly interface, adaptability, and wide compatibility. What sets Node-RED Dashboard apart is its intuitive design, which allows users with various levels of programming proficiency to effortlessly create and personalize interactive dashboards without extensive coding knowledge. Moreover, Node-RED Dashboard boasts remarkable integration capabilities, supporting an extensive range of protocols and interfaces. This compatibility extends to the ADXL345 accelerometer sensor, and DS18B20 digital temperature sensor module connected to the Raspberry Pi, enabling straightforward data collection and seamless visualization on the dashboard.

Real-time data visualization is a key feature of Node-RED Dashboard, empowering users to monitor and analyze sensor data in the moment. By leveraging its responsive and dynamic widgets, users can view data collected from the ADXL345 accelerometer sensor such as acceleration data and DS18B20 temperature sensor module such as

temperature data in real time. This instantaneous feedback proves invaluable for making prompt assessments and informed decisions based on the sensor data at hand. Node-RED Dashboard further impresses with its diverse set of customizable widgets, including gauges, charts, sliders, and switches. These widgets can be tailored to meet specific requirements, resulting in visually engaging and meaningful representations of the sensor data. Users have the freedom to configure these widgets according to their preferences, thus enhancing the interpretation and analysis of the presented information [39].

Additionally, the versatility of Node-RED Dashboard allows for easy expansion and integration of additional sensors or devices as needed. Its comprehensive library of available Node-RED nodes and flows offers ample options for enhancing the capabilities of the dashboard as shown in Figure 3.6. Consequently, the dashboard can effortlessly accommodate the integration of new sensors or devices without significant modifications to the existing setup. Node-RED Dashboard provides users with multiple deployment options, ensuring flexibility in choosing the most suitable method based on project requirements and scalability needs. Whether running the dashboard locally on the Raspberry Pi or hosting it on a remote server, Node-RED Dashboard facilitates a seamless deployment experience [40].

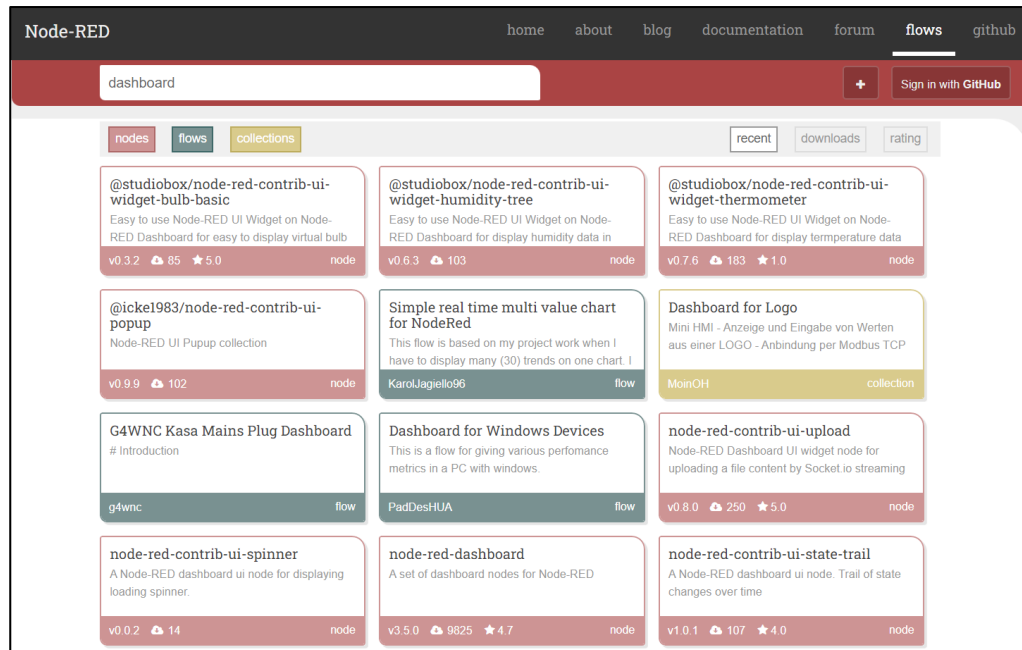


Figure 3.6: Overview of Node-RED Library

Furthermore, the thriving community surrounding Node-RED contributes to its appeal, as developers and enthusiasts actively provide support, share resources, tutorials, and examples. This collaborative community fosters knowledge sharing, troubleshooting, and innovation, ultimately enhancing the overall experience of utilizing Node-RED Dashboard. Considering these remarkable attributes, it becomes evident that Node-RED Dashboard is an outstanding choice for presenting data from a Raspberry Pi, ADXL345 sensor, and DS18B20 temperature sensor. Its user-friendly nature, adaptability, real-time data visualization capabilities, extensive integration options, and the supportive community surrounding it make Node-RED Dashboard a powerful tool for effectively visualizing and monitoring sensor data. With Node-RED Dashboard, users can derive valuable insights, make informed decisions, and drive innovation in their projects [41].

3.3.3.2 Installation of Nodejs and Node-RED on Raspberry Pi 3, Model b

Raspberry Pi is a single-board computer, size of credit card, developed in the United Kingdom by the Raspberry Pi Foundation. Before installing the Node.js module and Node-RED editor, it is necessary that Raspberry Pi has an operating system installed, Raspbian or NOOBS, which can be found on the official Raspberry Pi web page [42].

The command for installing Node.js module and Node-RED editor is as shown in Figure 3.7

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
```

Figure 3.7: Command for installing Node-RED in Raspberry Pi

Node-RED runs from the terminal with command: `node-red-start`. Accessing over the browser by entering the IP address of Raspberry Pi and the 1880 port, for example: `http://<ip.address.>:1880`, then a Node-RED interface will be displayed, confirming that the installation was successful. The appearance of the Node-RED is shown in

Figure 3.8. Pressing Ctrl + C stops the execution of the tool, and Node-RED is switched off by the command: node-red-stop.

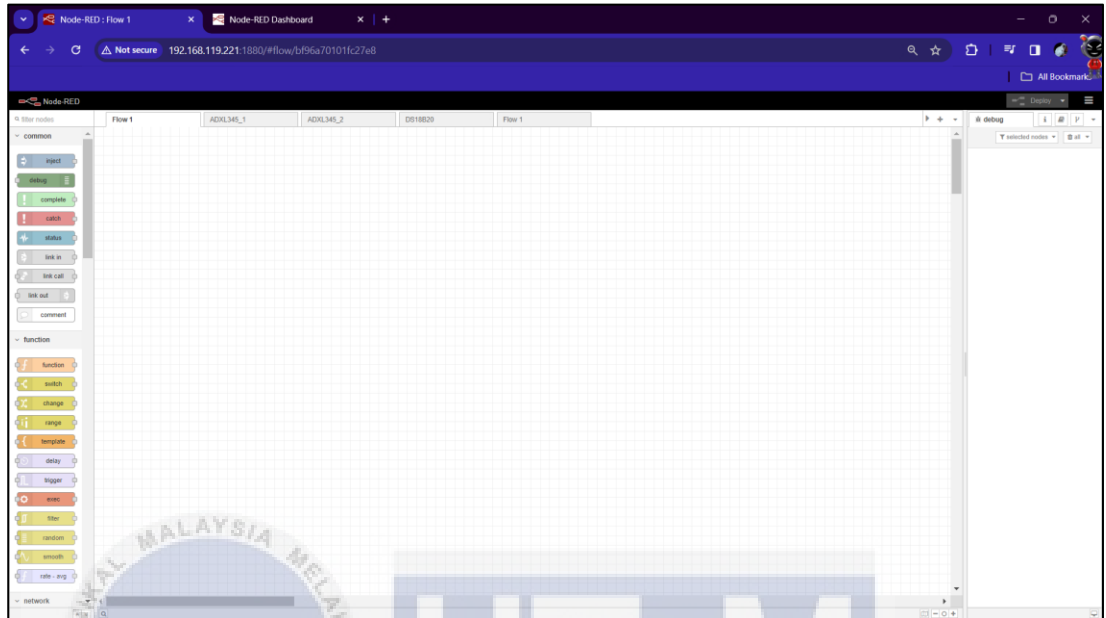


Figure 3.8: Interface of Nore-RED

3.3.3.3 Overview of Node-RED Dashboard of the project

Figure 3.9 shows the visual representation of the application's logic, consist of interconnected nodes and code block that perform specific functions and tasks.

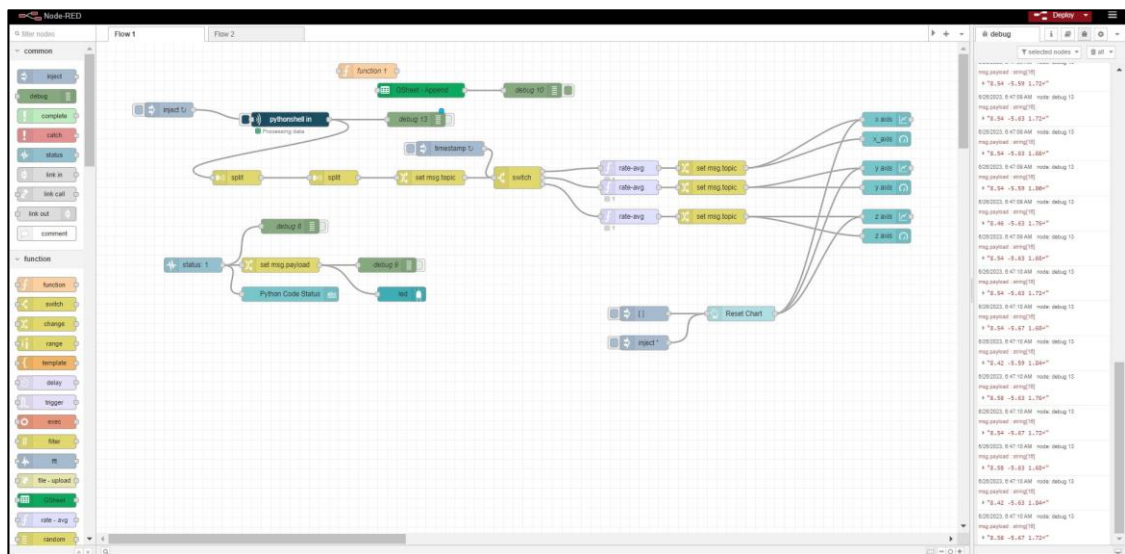


Figure 3.9: Node-RED Flow

The construction of a flow in Node-RED involves utilizing a user-friendly drag-and-drop interface, which simplifies the process of assembling nodes into configurations that yield functional programs. Node-RED offers a range of pre-installed nodes, and additional third-party nodes can be installed. Similar to Arduino's third-party libraries, Node-RED allows for the installation of such nodes. The name Node-RED derives from its foundation on Node.js, a lightweight JavaScript framework. This framework enables efficient and speedy execution, making it ideal for creating applications that require nimble performance, even on low-cost hardware like the Raspberry Pi. Being an open-source platform, Node-RED benefits from a large community of contributors. Its longstanding presence and stability have attracted users from hobbyists to major corporations, further validating its widespread adoption and usage [43].

3.3.3.4 Functional overview of Node-RED Flow

Based on the theoretical background presented above, a condition monitoring application have been developed and its environment using Node-RED. Various sensors record the motor's operating data (vibration) and changes in its environment (temperature). This data is displayed in real-time on an interface via a Node-RED Dashboard. Taking the advantages of the features provided by the Node-RED Dashboard, a simple and easy-to be use interface as shown in Figure 3.10 and related topic, using real-time important data, and manipulating some of the selected parameters on the interface.

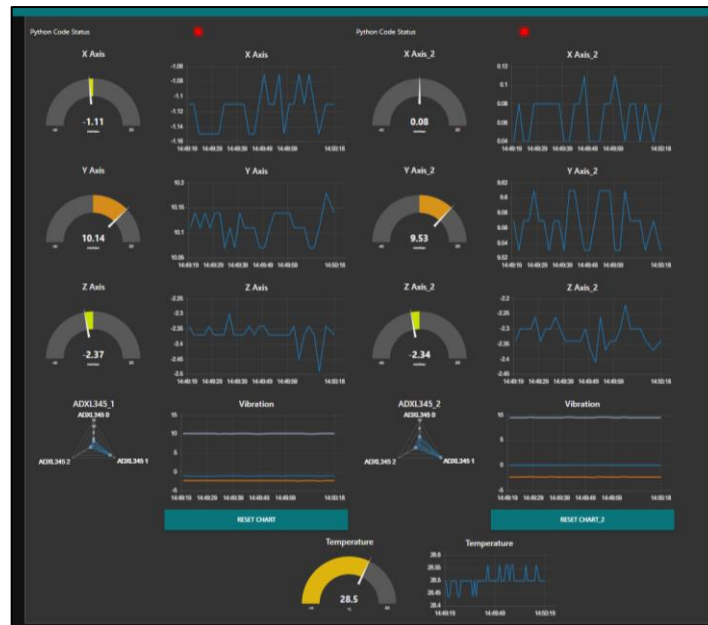


Figure 3.10: Node-RED dashboard visualization.

In order to have an adequate amount of IoT sensor data, a python code is used to simulate the operation of a general industrial motor and its environment. Using the code, the sensors is created using JSON (JavaScript Object Notation) objects and give them a start value. Figure 3.11 shows several nodes were used to execute the reading from ADXL345 sensor to Node-RED Dashboard. Pythonshell node is used to execute a python script from Node-RED. Input to the node will become the argument for the python script, output of the script will be sent to output of the node. The node connected to output node of Pythonshell is Split node, where it is utilized to break down a message containing raw accelerometer data typically received as a single measure or array into individual components such as acceleration values along the X, Y, and Z axes. The sensor sends a message containing space-separated acceleration values like “X Y Z”, the Split node can separate them into individual X, Y, and Z values. Next is the change node, where its role is to change the type of values from a string to numbers for mathematical calculations, for example add units (e.g., “mm/sec”) for clarity. Switch node is to routes messages to different outputs based on

certain conditions. The data read from sensors will be sent to three different processing nodes for X, Y, and Z values. Next is for Rate-avg node where its purpose is to calculate a rolling average of incoming values over a specified time window and it is to smooth out fluctuations and provides a more stable representation of sensor readings [44]. The period is set to 500 milliseconds for a cleaner visualization of readings. The gauge and chart node were used to display sensor's data in real-time as shown in Figure 3.10.

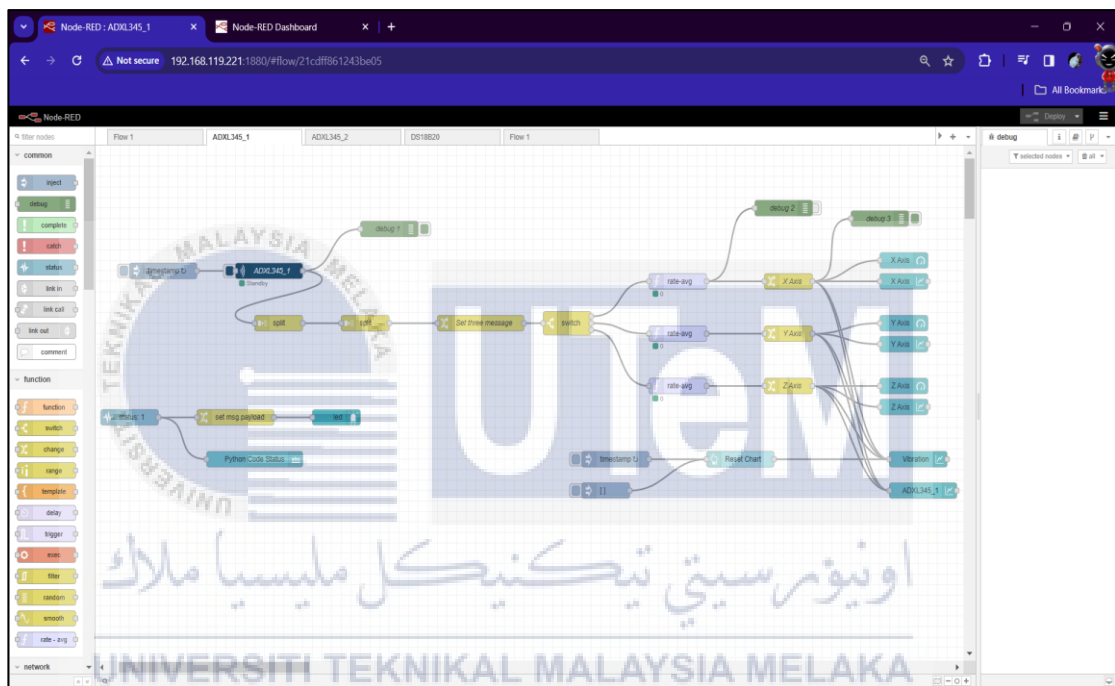


Figure 3.11: Node-RED flow for ADXL345 accelerometer sensor.

Figure 3.12 illustrates the flow of Node-RED for DS18B20 temperature sensor integrate with Raspberry Pi 3. Several nodes also were used to perform a logic function which is Inject, DS18B20, and Function nodes. *Node-red-contrib-sensors-ds18b20* is specifically designed for DS18B20 sensors that work with Raspberry Pi, hence, the node is used to configure and read the sensor's data in a python code. For every detected sensor, the system provides an `msg` object with the topic corresponding to the sensor's ID and the temperature as its payload. The node's topic is adjusted to the device ID, ensuring a singular reading for that specific sensor. When the "Array"

option is enabled or when ``msg.array`` is true, the data is presented as an array within ``msg.payload``. Additionally, the timestamp is consistently embedded as ``msg.timestamp``. A straightforward workflow involves triggering an input node with a blank message, directing it to the sensor node, and observing the resultant output via a debug node.

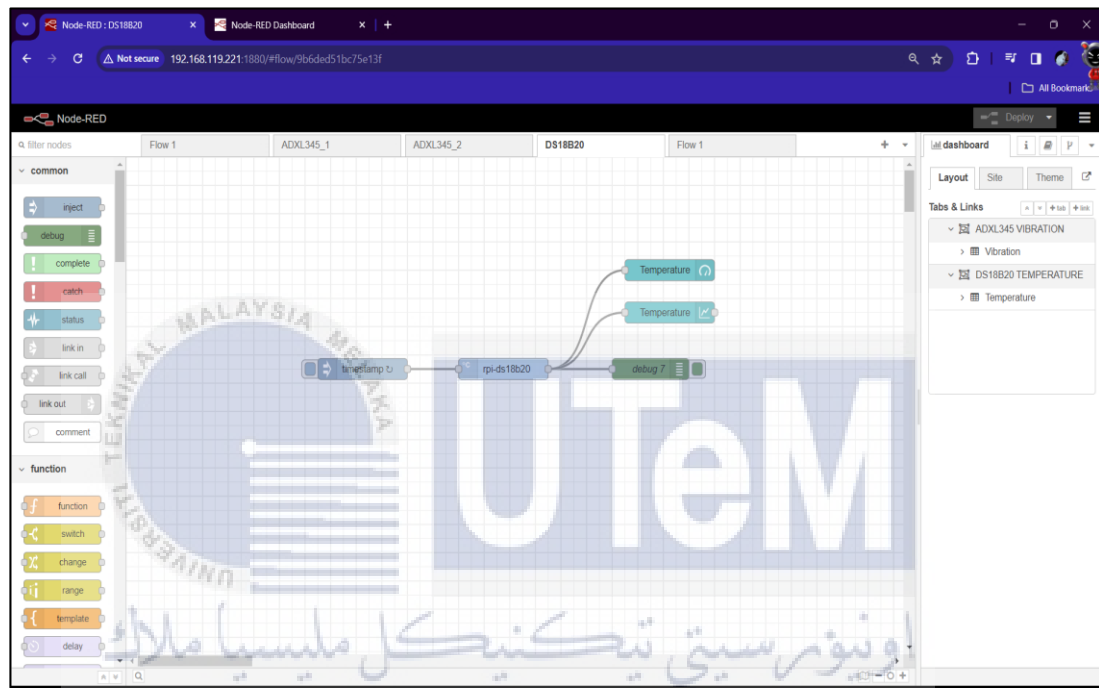


Figure 3.12: Node-RED flow for DS18B20 temperature sensor.

3.3.4 3D-printing of prototype

Figure 3.13 shows the 3D-print Ender-3 that was used in printing the prototype for the project. Ender-3 is an FDM 3D printer, an open-frame all-metal 3D printer. It prints parts by melting and extruding a thermoplastic filament through a heated single extruder equipped with a Bowden feeder system. The extruder module has been designed for printing with generic 1.75mm filaments, whatever the brand [45]. There are total of 9 parts of the prototype to be printed using Ender-3 as shown in Figure 3.13. Before slicing the 3D model, Onshape modelling software is used to generate

and designing the prototype. Then, the file is saved in stl file to open in Cura Ultimaker software. This free and open-source software can give a control over the printing settings, it features ongoing upgrades and provides over 400 settings to fine-tune print model to get optimum printing results.



Figure 3.13: Ender-3 3D printer

3.3.5 Python code for ADXL345 accelerometer sensor and DS18B20 temperature sensor

The Python programming code that was generated to integrate ADXL345 accelerometer sensor with Raspberry Pi microcontroller can be referred in Appendix K. The Python program code is written using Nano text editor and saved in the terminal of Raspberry Pi. The file is saved as `adxl345.py` and `adxl345_2.py` in order to differentiate the file from other typed of files. The Python script also can be run directly from the terminal by using `'python3 adxl345.py'`. The script will be executed inside the terminal. To stop the execution, simply pressed CTRL + C. Additionally, the Python code enable the integration between a Raspberry Pi and ADXL345 accelerometer sensor. Upon initialization, the code configures the sensor's

measurement mode and sets the acceleration range. A function, `read_acceleration`, is employed to extract raw acceleration data from the sensor registers, convert the combined bytes into a signed 16-bit value, and subsequently transform it into acceleration values in m/s^2 units. Within the main program loop, acceleration data along the X, Y, and Z axes are continuously retrieved, printed, and displayed, with a 2-second delay between each iteration. Exception handling ensures a smooth exit from the program upon receiving a keyboard interrupt.

In Appendix L shows the Python programming code write that was generated for DS18B20 temperature sensor integrate with Raspberry Pi microcontroller. In order to execute the file is similar to what mentioned above. The file is saved as `ds18b20.py` in the terminal of Raspberry Pi. For this python script, it utilizes the `1wthermsensor` library to continuously read temperature data from a DS18B20 sensor. Upon execution, the `read_temperature()` function initializes the sensor and retrieves the current temperature in Celsius. The main loop then prints this temperature value with two decimal places, followed by a one-second delay before the next reading. If the user interrupts the script, it gracefully stops the temperature readings.

3.3.6 MATLAB for data analysis and predictive maintenance

After the collecting data from sensors, the MATLAB software is utilized to analyse the data for data analytics and processing. The Neural Net Time Series apps is used to visualize, and train dynamic neural networks using the collected data. In Neural Net Time Series app, it provides a built-in training algorithms to train neural network. There are three type of training algorithm available but for this project, only two training algorithms that were used which is Levenberg-Marquardt and Scaled conjugate gradient backpropagation. Both algorithms have their own advantages and

disadvantages, the main difference lies in their approach. The LM algorithm adjusts the self-scaling parameter to improve convergence, while the SCG algorithm uses a conjugate gradient method to find the optimal solution. In terms of performance, the LM algorithm has been shown to have quadratic convergence under certain conditions, while the SCG algorithm has been found to have lower false alarm rate in flood forecasting [46]. Figure 3.14 illustrates the interface of the Neural Net Time Series app in MATLAB. At Train can select either Levenberg-Marquardt or Scaled Conjugate Gradient.

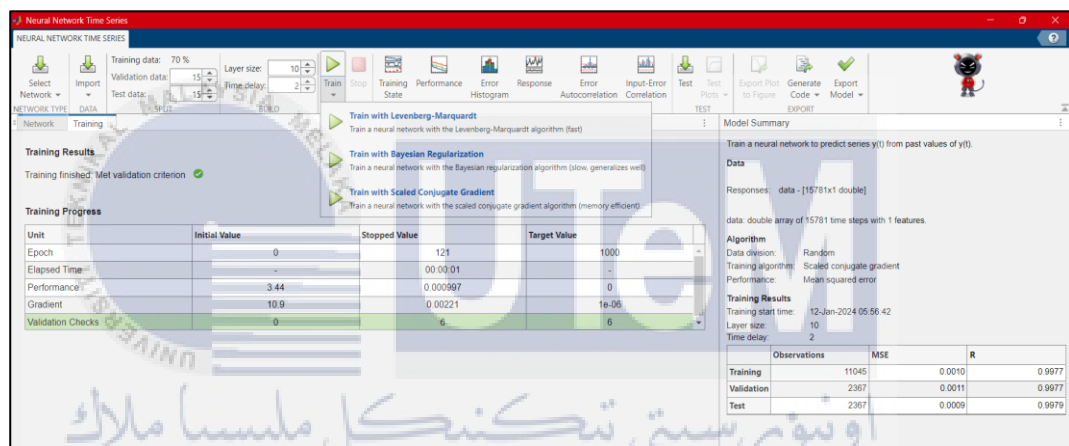


Figure 3.14: Network pane in MATLAB for NAR network

From the select network, there are three different kinds of time series problem. Hence for this project, only one series is involved which is each data for sensors is taken only one reading. It is called as NAR Network where the future values of a time series are predicted only from the past values of that series, this is called as a nonlinear autoregressive, or NAR. After that, to train the network, select Train > Train with Levenberg-Marquardt / Scaled Conjugate Gradient, then it will show the Training pane as shown in Figure 3.15. Training continues until one of the stopping criteria is met. In this example for data DS18B20 temperature sensor, training continues until the validation error increases consecutively for six iterations.

Training Results			
Training finished: Met validation criterion ✔			
Training Progress			
Unit	Initial Value	Stopped Value	Target Value
Epoch	0	15	1000
Elapsed Time	-	00:00:00	-
Performance	7.83	0.00097	0
Gradient	11.4	0.000699	1e-07
Mu	0.001	1e-07	1e+10
Validation Checks	0	6	6

Figure 3.15: Training Progress in Training pane

Figure 3.16 shows the Model Summary that contains information about the training algorithm and the training results for each data set. In order to analyse the results, the plot is generated, there are several graphs that can be plotted including Training State, Performance, Error Histogram, Response, Error Autocorrelation, and Input-Error Correlation.

Model Summary			
Train a neural network to predict series $y(t)$ from past values of $y(t)$.			
Data			
Responses: data - [4880x1 double]			
data: double array of 4880 time steps with 1 features.			
Algorithm			
Data division:	Random		
Training algorithm:	Levenberg-Marquardt		
Performance:	Mean squared error		
Training Results			
Training start time:	08-Jan-2024 05:07:59		
Layer size:	10		
Time delay:	2		
	Observations	MSE	R
Training	3414	9.7244e-04	0.9980
Validation	732	8.6880e-04	0.9981
Test	732	9.5613e-04	0.9980

Figure 3.16: Model Summary of training algorithm

Other than plotting the graphs directly from there, the MATLAB code is generated to reproduce the previous step from the command line. Then, the trained network and results is exported to Workspace. Figure shows the MATLAB code that was generated using the Neural Net Time Series app. The code is then modified to customize the training process. The related program code can be referred in Appendix J.

CHAPTER 4

RESULTS AND DISCUSSION



This chapter serves as an important junction in thesis journey, where we meticulously dissect the essence of our research. Here, readers will find a blend of succinct summaries and in-depth analyses of our primary findings. Leveraging a combination of visuals and narrative, we not only present our data but also contextualize it within the broader academic framework. This chapter aims to bridge the gap between raw results and meaningful insights, offering readers a comprehensive understanding of the study's implications, significance, and potential avenues for future exploration.

4.1 Collection data for two motors

For this project, the data is collected from two different motors which is located at Automation Lab and Industrial Electronic Lab. The data is taken for four hours in three

days. The sensors are mounted on the motors to collect vibration and temperature data. Each data is then analyzed by using MATLAB software. Figure 4.1 shows the Node-RED Dashboard that display the reading taken from the ADXL345 sensors and DS18B20 sensor. The data from ADXL345 is display by axes, X, Y, and Z meanwhile the data taken from DS18B20 is display in Celsius, using gauge and charts.

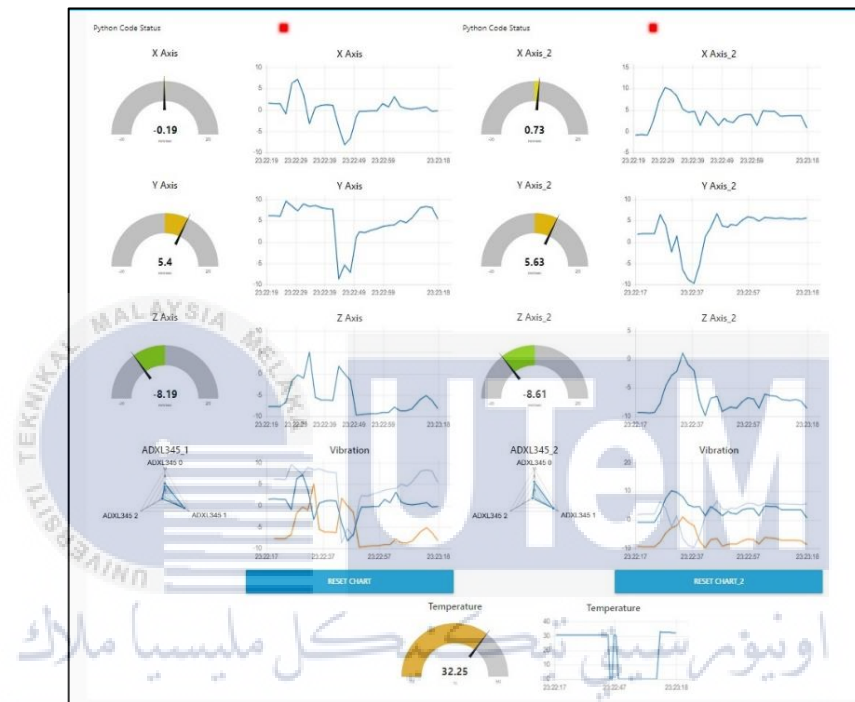


Figure 4.1: Data from ADXL345 sensor and DS18B20 sensor displayed in Node-RED Dashboard

The data from both sensors have been collected and visualized using gauge and charts in Node-RED dashboard. The collected data is saved in CSV file for further analysis. The MATLAB software is utilized to make a predictive analytic using Neural Net Time Series app.

4.2 Detection of motor abnormalities in vibration and temperature levels

This project addresses a real-world industrial challenge faced by Texas Instruments: implementing machine condition monitoring for predictive maintenance through vibration and temperature level detection of motors. Due to the strict

regulations and standards within the industrial environment, data collection was limited to two motors available at faculty for vibration and temperature signal analysis. Figure 4.2 displays the collected data from both ADXL345 sensors in Node-RED dashboard. The data is plotted as an acceleration-over-time graph, offering a clear visualization of motor vibrations. The dashboard gauge showcases the readings of the ADXL345's triple-axis accelerometer (X, Y, and Z axes) in real-time. Two ADXL345 sensors were utilized in this project. To enable the I2C communication protocol, each sensor's SPA pin had to be connected to either ground or VCC on the Raspberry Pi. This configuration allowed differentiation between sensor 1 and sensor 2, enabling separate data readouts for each motor and comprehensive vibration analysis.

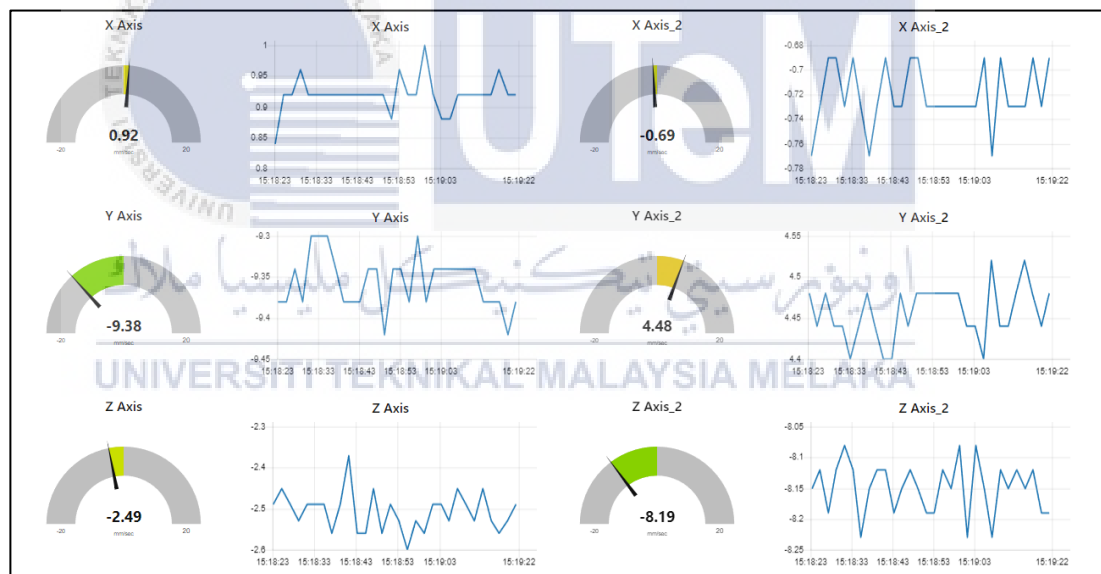


Figure 4.2: Node-RED dashboard for ADXL345 sensor's data

Figure 4.3 depicts a dashboard gauge and chart displaying real-time temperature data collected from the motors using DS18B20 sensors. This sensor utilizes a simple one-wire protocol, requiring only one data pin for connection to a Raspberry Pi data logger. Additional pins provide ground and power supply (VCC). Temperatures from both

motors are monitored simultaneously, enabling side-by-side comparisons and insightful analysis.

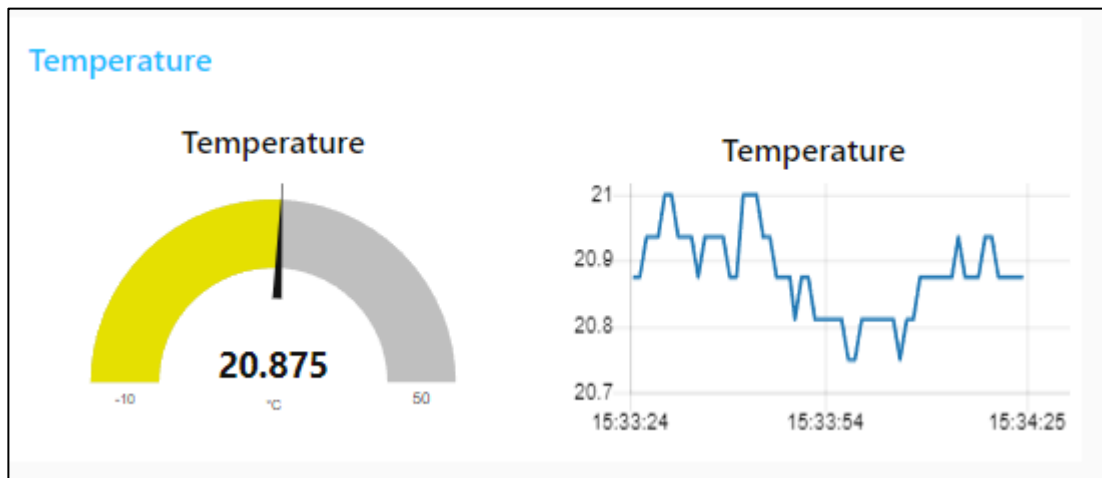


Figure 4.3: Node-RED dashboard for DS18B20 sensor's data

4.3 A monitoring system using Node-RED Dashboard

For this project, where vibration and temperature data from motors are measured, a Node-RED dashboard is employed for real-time monitoring. As the sensors capture data, it is instantly displayed on the Node-RED Dashboard, providing users with immediate insights into environmental conditions. The ADXL345 sensor contributes precise three-axis acceleration data, supplying valuable details about movement and vibrations. Simultaneously, the DS18B20 sensor ensures accurate temperature measurements. Node-RED, known for its user-friendly visual programming, seamlessly integrates these sensor inputs, enabling users to effortlessly monitor and analyze the data. The Raspberry Pi acts as a reliable and compact computing platform, ensuring the monitoring system's dependability. Through the Node-RED dashboard, users can visually engage with the sensor data, fostering efficient monitoring and decision-making. This cohesive solution showcases the effectiveness of combining open-source technologies to create adaptable and user-friendly monitoring systems.

4.4 A predictive analytic using MATLAB software.

The collected data for four days from motors one is manufactured in 2007 while the other motor is manufactured in 2020 is analyzed to make a predictive analytic. The MATLAB software is utilized for data processing. The data is analyzed for each sensor and then a comparison is made between Motor 1 and Motor 2. In this analysis, there are two methods in Neural Net Time Series app are used to make a comparison between those method to choose which are the best train performance. Here, the Levenberg-Marquardt and Scaled Conjugate Gradient is used. The data for vibration and temperature is analyzed separately for both motors using these methods.

4.4.1 Levenberg-Marquardt Train Performance

After completing collect data from the sensor for three hours each day, the data then is analyzed using two methods which are Levenberg-Marquardt and Scaled Conjugate Gradient techniques. The Levenberg-Marquardt training algorithm is an optimization method used to solve nonlinear least squares problems and it uses a self-scaling parameter to control its performance [46].

4.4.1.1 DS18B20 data for 3 days at Motor 1

The historical data collected from DS18B20 temperature sensor mounted on Motor 1 and Motor 2 is analyzed using Levenberg-Marquardt algorithm in MATLAB. Purpose of using this algorithm is to build a model to predict future temperature. A neural network is chosen to map the past temperature values to the predicted future temperature. The Levenberg-Marquardt algorithm can be used to optimize the model's parameter by minimizing the difference between the predicted and actual temperatures in the training data. Other than that, it also is a type of machine learning algorithm that train the neural network. Firstly, the collected data for each day is saved in CSV file

and analyzed. Figure 4.4 illustrates the neural network training data using Levenberg-Marquardt algorithm.

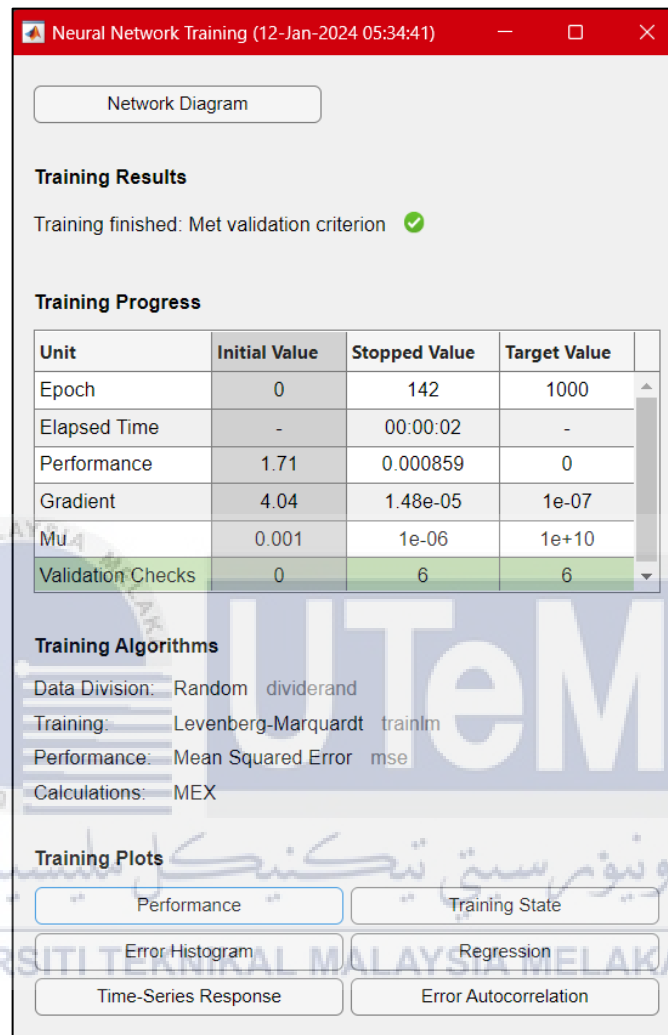


Figure 4.4: Neural Network Training for Levenberg-Marquardt algorithm that data taken for three days at Motor 1.

As can be seen from Figure 4.4, the results of training program shows that the network was able to learn the training data very well. This is because the performance of the network improved significantly from the beginning to the end of the training. From observation and analyzation of the data, the network was trained for 142 epochs, where it represents one complete pass through the entire training dataset. The network was trained for 142 epochs, so it means that it repeatedly learned from the training

data 48 times. The values for gradient also decreased significantly, which indicates that the network is no longer train anything new from the training data. As for the mu values, it is also decreased, from 0.001 to 1e-06, which indicates that the algorithm is becoming more confident in its results.

As illustrates in Figure 4.5, can be seen that training and validation errors decreased until the highlighted epoch which is 136. The graph plots the Mean Squared Error (MSE) on a logarithmic scale against the number of epochs. From the graph, both training and validation curves start with high MSE values, shows that model's initial have a poor performance in predicting data. During the early epochs, both curves decrease rapidly, showing significant improvement in the model's ability to learn the historical data patterns. The training curve continues to decrease and eventually plateaus, showing that the model has optimized the parameters for the training data. The best validation performance is marked and achieved at epoch 136 with an MSE of 0.0012498. The validation curve follows closely with the training curve but remains higher. This model indicates that it is not overfitting the training data and can be seen generally.

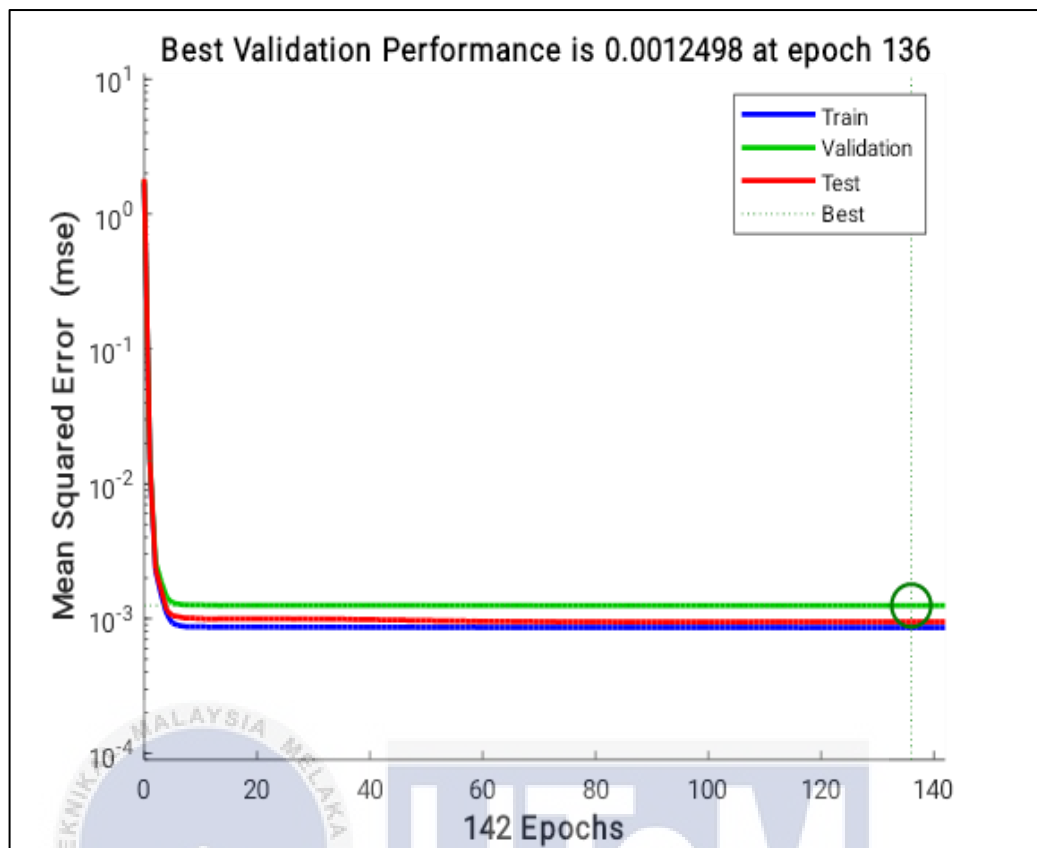


Figure 4.5: Validation performance for DS18B20 sensor using LM algorithm at Motor 1

As shown in Figure 4.6, the graph represents data collected from a DS18B20 sensor over three days. The regression signal represents a scatter plot of data points in circle and a fit solid blue line. The circles represent the actual data points, and the solid blue line is the fit achieved using this algorithm. The data points are closely aligned with the fit line, indicating a strong positive linear relationship between the 'Target' and 'Output'. The correlation coefficient is 0.99789, which is very closed to 1, indicating an extremely strong positive linear correlation between the two variables. The 'Target' axis represents the desired future temperature that want to predict while the 'Output' axis represents the temperature measurements by the DS18B20 sensor itself. The straight line is the slope regression line where from the graph, the slope is positive. It means that the output temperature increases as the target temperature increases.

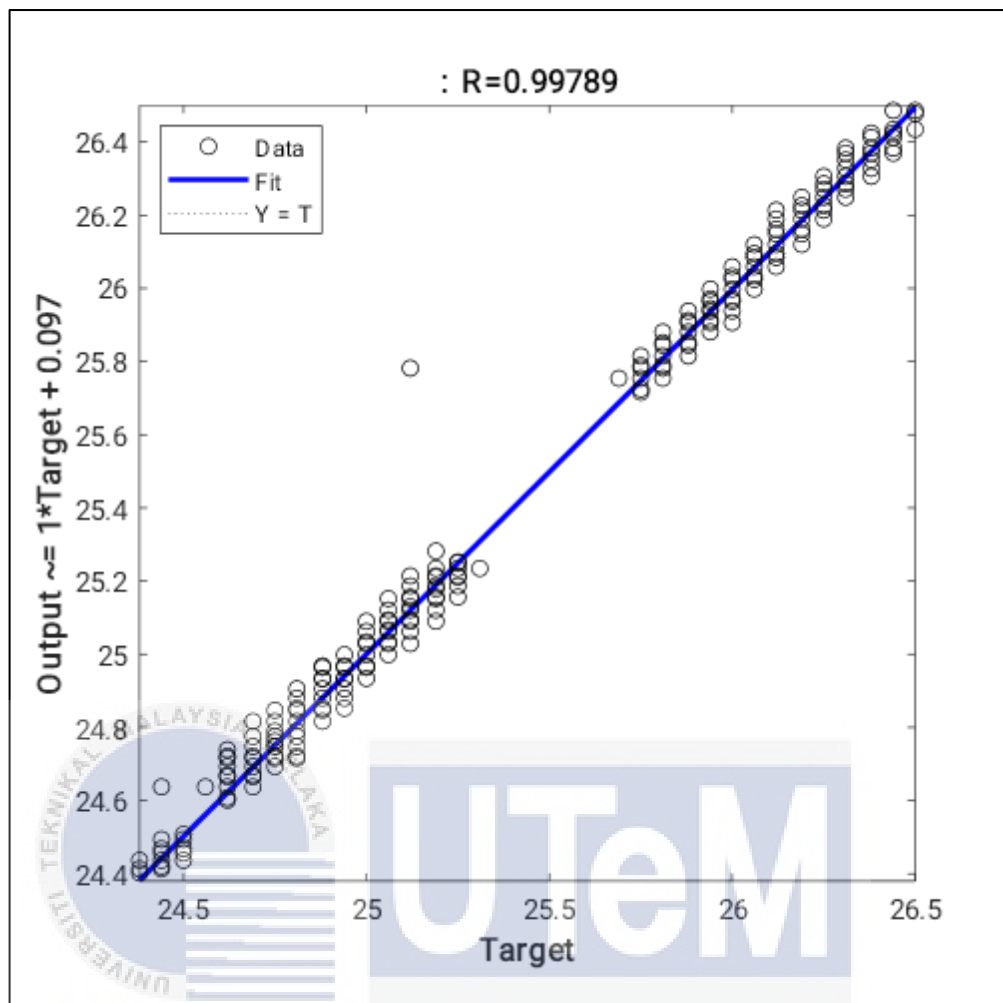


Figure 4.6: Regression graph for DS18B20 sensor using LM algorithm at Motor 1

Figure 4.7 depicts the Mean Absolute Error (MAE) between actual and predicted offset over time. The MSE is utilized to measure the average magnitude of errors in a set of predictions. From the graph, can be seen that MAE have a lower range of number which indicates that it is better because of it have a small average error. The MAE is calculated as the average of the absolute differences between the predicted and actual values of temperature data. Furthermore, the Levenberg-Marquardt algorithm is typically used for curve-fitting problems, so in this context, it has been used to minimize the error between observed and predicted temperature readings from the

sensor over three days. There are two prominent spikes in MAE around the 1900 and 9000 mark on the time axis, signalling a large error at that specific point in time.

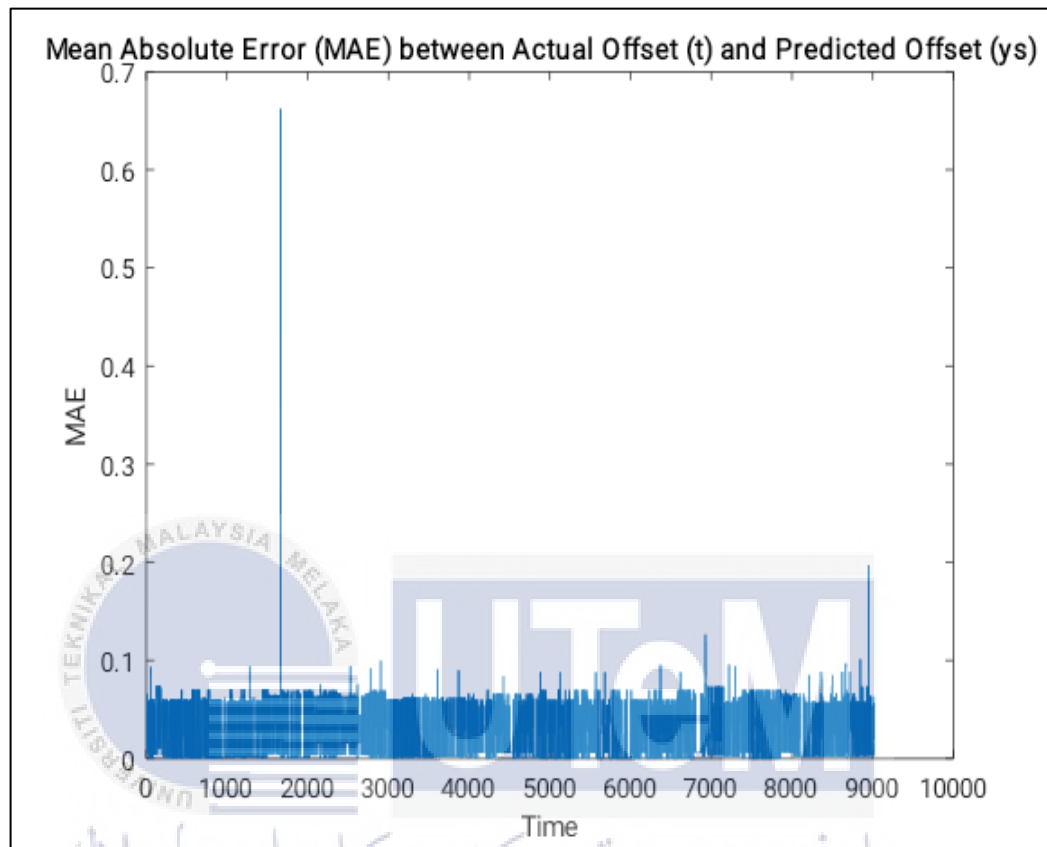


Figure 4.7: Graph OF Mean Absolute Error (MAE) between actual and predicted offset for DS18B20 sensor using LM algorithm at Motor 1

Figure 4.8 illustrates the graph of a comparison between actual and predicted offset over time. The actual offset starts at approximately 26.5 mm and have been dropped around 1700 seconds, stabilizing near 25.2 mm for the remainder of the time observed. In contrast, the predicted offset remains relatively constant throughout, with minor fluctuations but not mirroring the sharp decline of the actual offset. These generated graphs are utilizing sensors data at Motor 1 for three days.

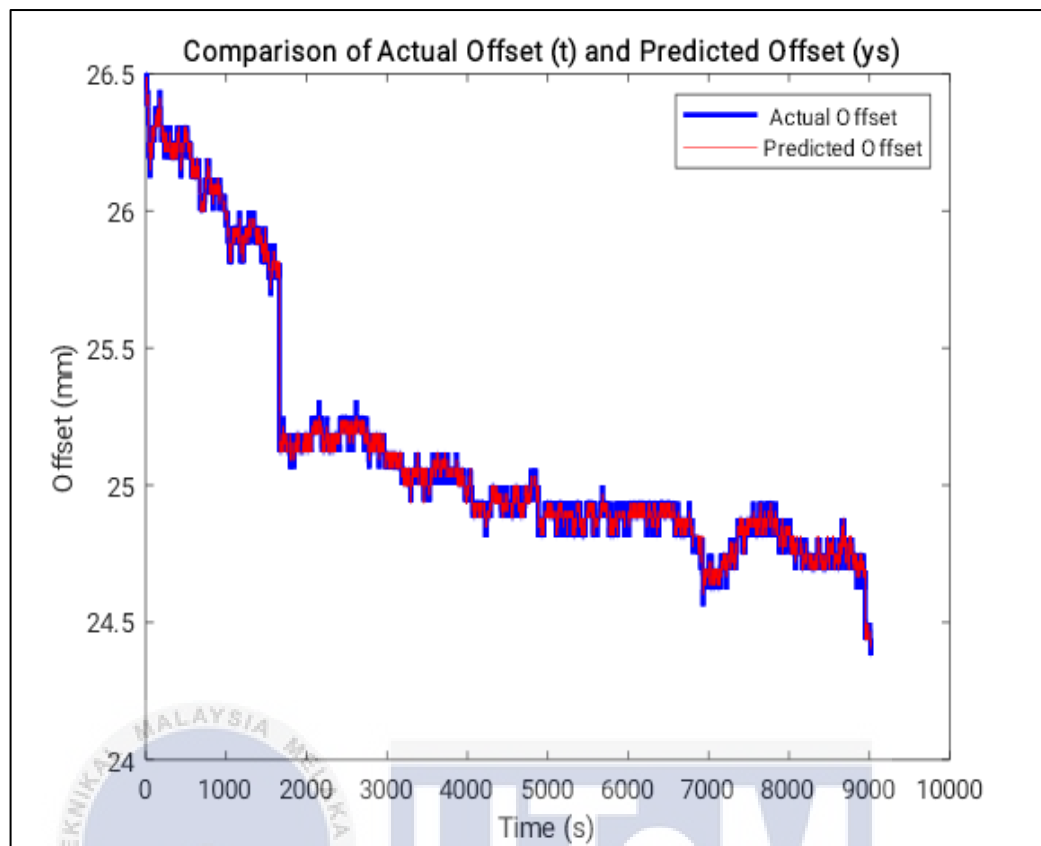


Figure 4.8: Graph for comparison between actual and predicted offset using LM algorithm at Motor 1

Table 4.1 shows the Mean Square Error (MSE) and regression values for training, validation, and testing phases. The MSE measures the average squared difference between the predicted and actual values while the R value measures the correlation between the predicted and actual values. The MSE values are quite low for all three phases, indicating a good model fit for a better performance with the minimal error between the predicted and actual outcomes. The R values are very close to 1 for all three phases as well, suggesting a strong correlation between the predicted and actual outcomes, indicating an excellent fit of the model. The graph can be referred to Appendix E.

Table 4.1: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of DS18B20 sensor using LM algorithm at Motor 1

	MSE	R
Training	0.000902	0.99804
Validation	0.0018	0.99715
Testing	0.00108	0.99794

4.4.1.2 ADXL345_1 data for 3 days at Motor 1

Data taken from ADXL345 sensors also been analyzed using the same method which is LM algorithm. The data is combined together for three days to make data processing become easy. For ADXL345 sensor at Motor 1, from observation and analyzation of the data, the network was trained for 51 epochs, where it represents one complete pass through the entire training dataset. The network was trained for 51 epochs, so it means that it repeatedly learned from the training data 51 times. The values for gradient also decreased significantly, which indicates that the network is no longer train anything new from the training data. This data explanation can be referred in Appendix A

Figure 4.9 depicts the Mean Squared Error (MSE) in relation to the number of epochs for predictive analytics model during its training, validation, and testing phases. The MSE is plotted on a logarithmic scale. As the number of epochs increases, the error for training, validation, ad testing decreases significantly and then stabilizes. The best validation performance is indicated by the lowest mean squared error that was marked on the graph with a circle at epoch 45 with an MSE of 2.4494. All three lines-Train, Validation, Test start with high MSE values but rapidly decreases as epochs increase.

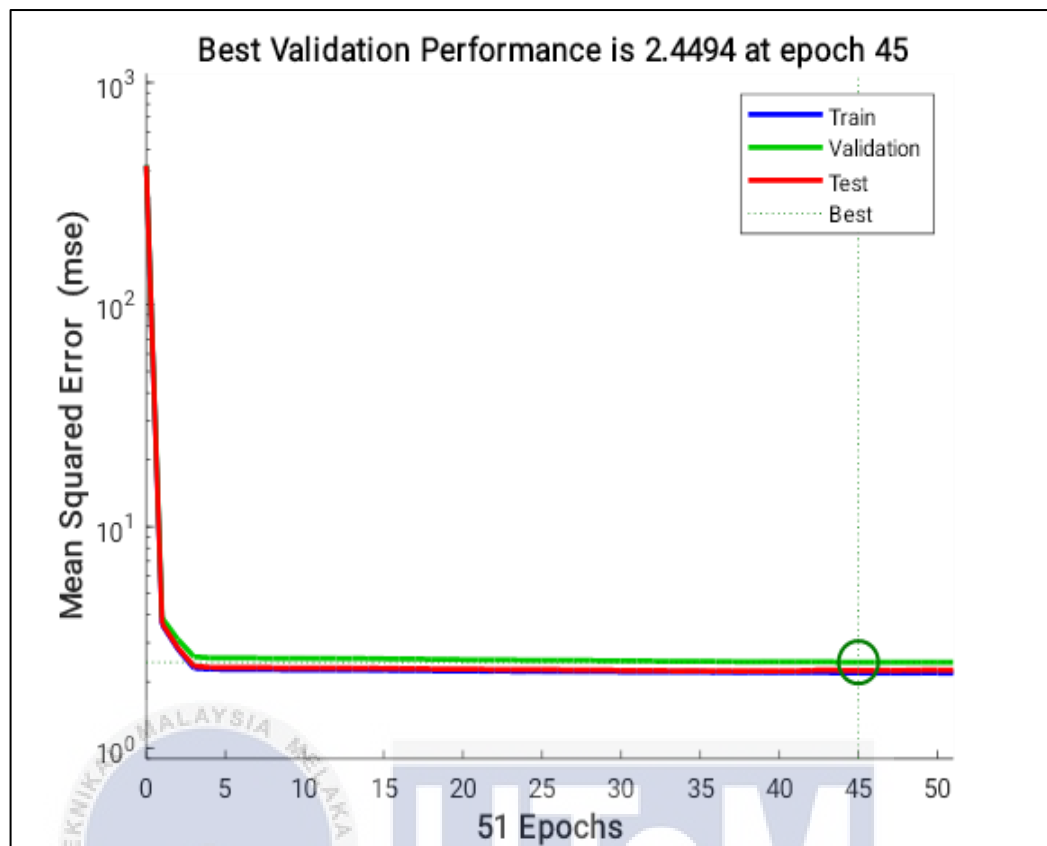


Figure 4.9: Validation performance for ADXL345_1 using LM algorithm at Motor 1

Figure 4.10 illustrates the regression analysis for using the LM algorithm. It shows the relationship between target and output variables. The R-squared value of 0.96771 indicates a very strong positive correlation, meaning that the model explains 96.71% of the variability in the dependent variable by the independent variable. In terms of predictive analytics, this high R-squared value suggest that the model has excellent predictive power and accuracy. Therefore, the model can be used to predict future values of the dependent variable with a high degree of accuracy.

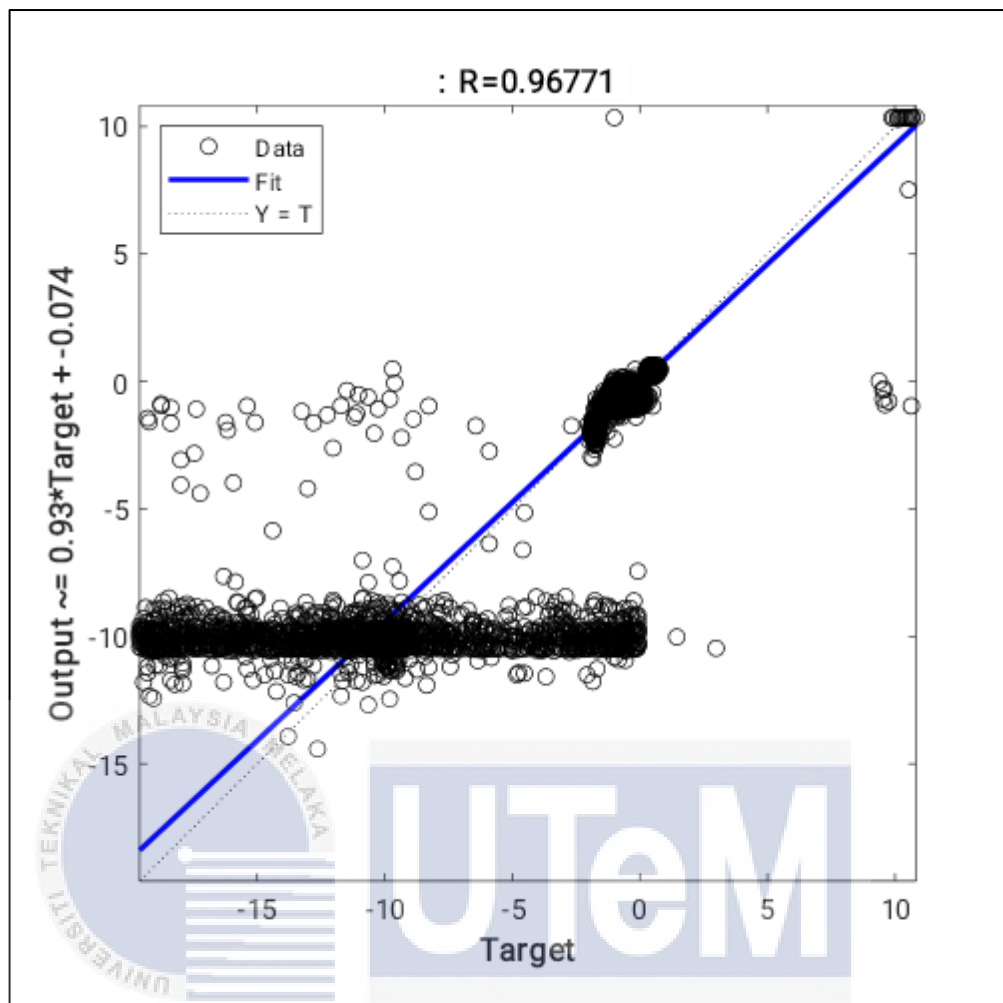


Figure 4.10: Regression graph for ADXL345_1 sensor using LM algorithm at Motor 1

Table 4.2 shows the values for training, validation, and testing phases of a model. It can be observed that the model has a lower MSE values indicates it is a better fit of the model to the data. The training phase has an MSE of 1.84, indicating a relatively good fit while the validation phase has a higher MSE of 2.92, suggesting that the model didn't perform as well on unseen data. The R values for each phase is listed in the table and it is shown that all R values close to 1, indicating a strong positive correlation. The related graph can be referred in Appendix E.

Table 4.2: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_1 sensor using LM algorithm at Motor 1

	MSE	R
Training	1.84	0.96829
Validation	2.92	0.96471
Testing	2.4	0.96802

4.4.1.3 ADXL345_2 data for 3 days at Motor 1

The second ADXL345 sensor also been analyzed using LM algorithm, although it is the same sensor, but the placement of sensors is different and the reading also different. Hence, for both ADXL345 sensors, the reading is record and analyzed using the same training algorithm. For this second ADXL345, from observation and analyzation of the data, the network was trained for 15 epochs, where it much lower than the first ADXL345 sensor. The network was trained for 15 epochs, so it means that it repeatedly learned from the training data 15 times. The values for gradient also decreased significantly, which indicates that the network is no longer train anything new from the training data. This data explanation can be referred in Appendix A.

Figure 4. 11 depicts the best validation performance which indicated by the lowest mean squared error (MSE), was achieved at epoch 9 with an MSE of 5.5183. The graph depicts MSE over 15 epochs for training, validation, and testing phases. The x-axis is labeled “15 epochs” and shows the increments of five from 0 to 15. The MSE decreases sharply during the initial epochs and then stabilizes. The best validation performance, marked by a circle on the graph, is 5.5183 at epoch 9. This indicates that at this point, the model has achieved its lowest validation error and is likely to be the most generalized version of the model before it starts overfitting. Overfitting is a modelling error that occurs when a statistical model fits too closely to a particular set

of data and may therefore fail to fit to additional data or predict future observations reliably.

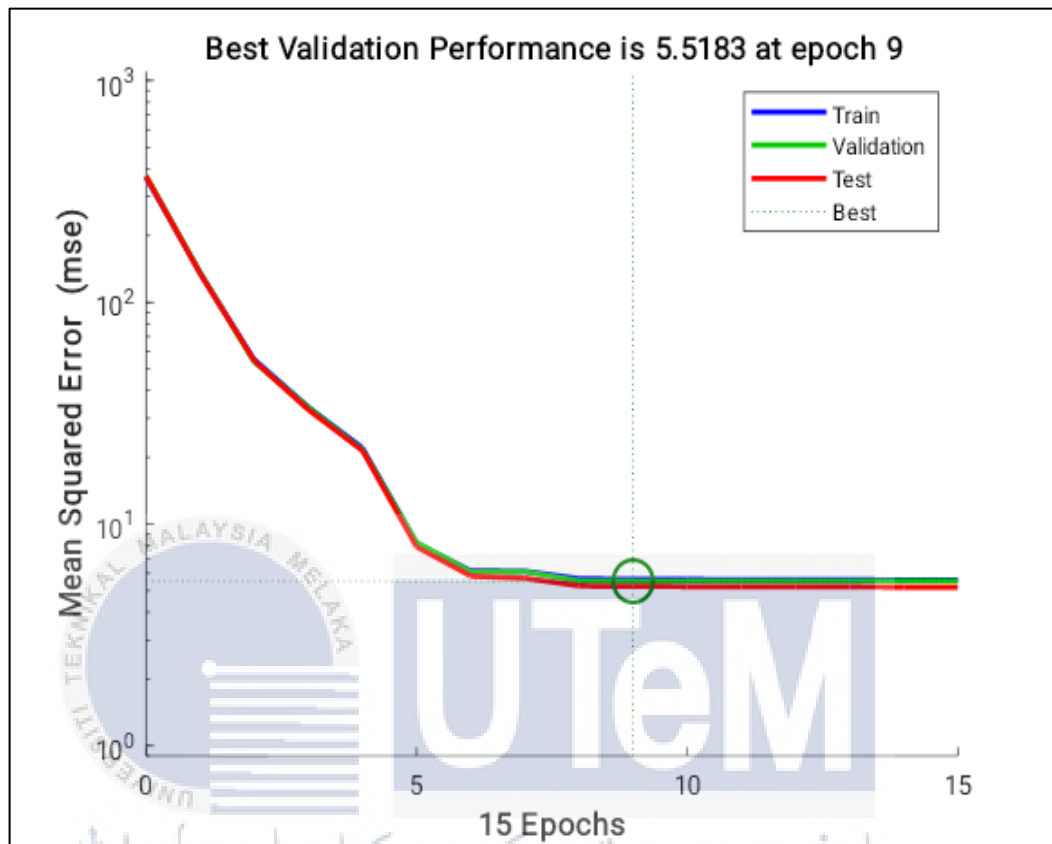


Figure 4.11: Validation performance for ADXL345_2 sensor using LM algorithm at Motor 1

Figure 4.12 illustrates the regression analysis using the LM algorithm for second ADXL345 sensor. The relationship between the LM algorithm and predictive analytics is demonstrated through the fit of the model to the data points. The R values, 0.9293 indicates a strong positive correlation, meaning that as the target variable increases, the output also increases. In terms of performance, the high R-value suggests that the LM algorithm is performing well in predicting outcomes based on the input data.

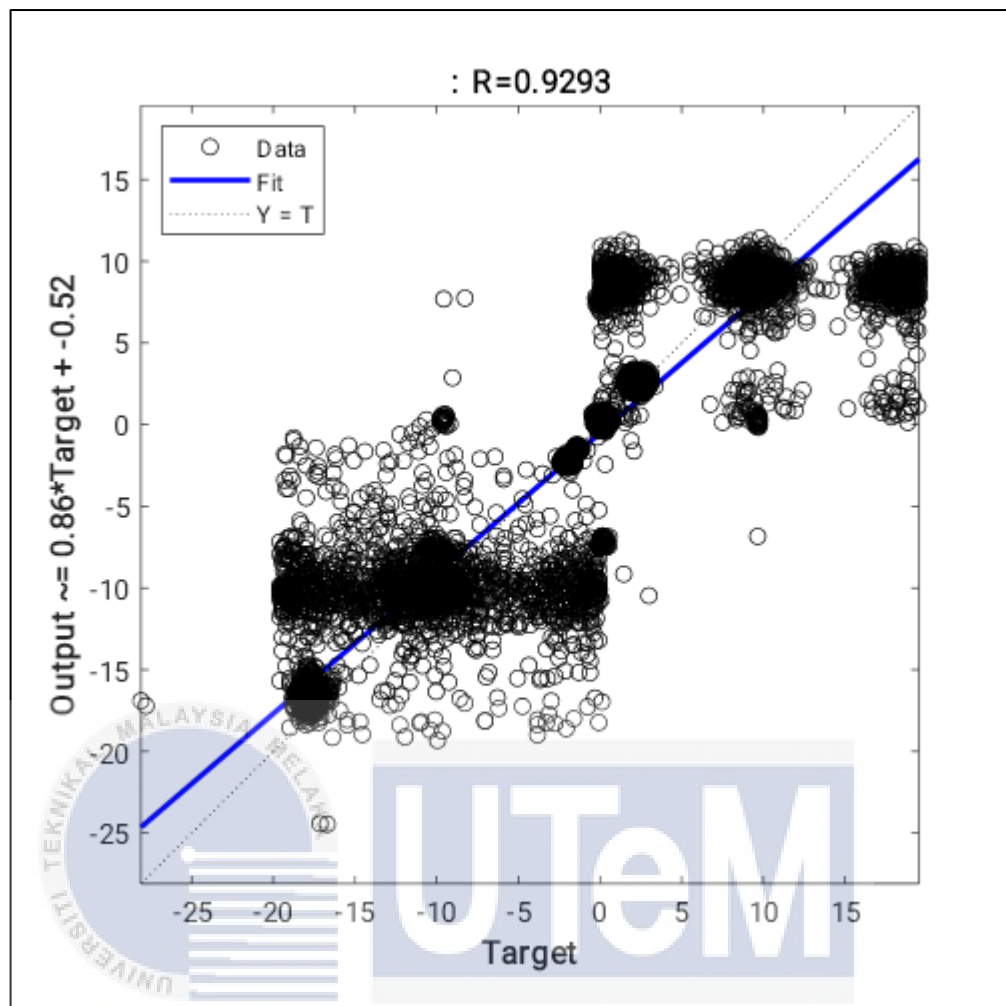


Figure 4.12: Regression graph for ADXL345_2 sensor using LM algorithm at Motor 1

Next as for the Table 4.3, the MSE and regression values for three phases is listed using the LM algorithm. The MSE values are 7.29, 5.31 and 4.51 for training, validation, and testing respectively. Meanwhile the R values are 0.92822, 0.93114, and 0.93259 respectively. The MSE values were decreased from training to testing which indicates that the model is learning effectively. The lowest MSE in testing suggests that the model has generalized well from its training data. As for the R values, all values are closer to 1 which indicates a strong positive relationship between predicted and actual values in all phases. In conclusion, as the values for MSE are decreasing, and R values are high, the LM algorithm is becoming effective. The high R values signify

that predictions made by this model can be highly trusted due to strong correlations with actual outcomes. The regression plot for each phase can be referred in Appendix E.

Table 4.3: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_2 sensor using LM algorithm at Motor 1

	MSE	R
Training	7.29	0.92822
Validation	5.31	0.93114
Testing	4.51	0.93259

4.4.1.4 DS18B20 data for 3 days at Motor 2

The readings from DS18B20 sensor at Motor 2 also been recorded and analyzed using the same training algorithm which is Levenberg-Marquardt algorithm. Figure 4.13 illustrates the neural network training algorithm. From observation and analysis of the data, the network was trained for 22 epochs, where it represents one complete pass through the entire training dataset. The network was trained for 22 epochs, so it means that it repeatedly learned from the training data 22 times. The values for gradient also decreased significantly, which indicates that the network is no longer train anything new from the training data. As for the mu values, it is also decreased, from 0.001 to 1e-05, which indicates that the algorithm is becoming more confident in its results.

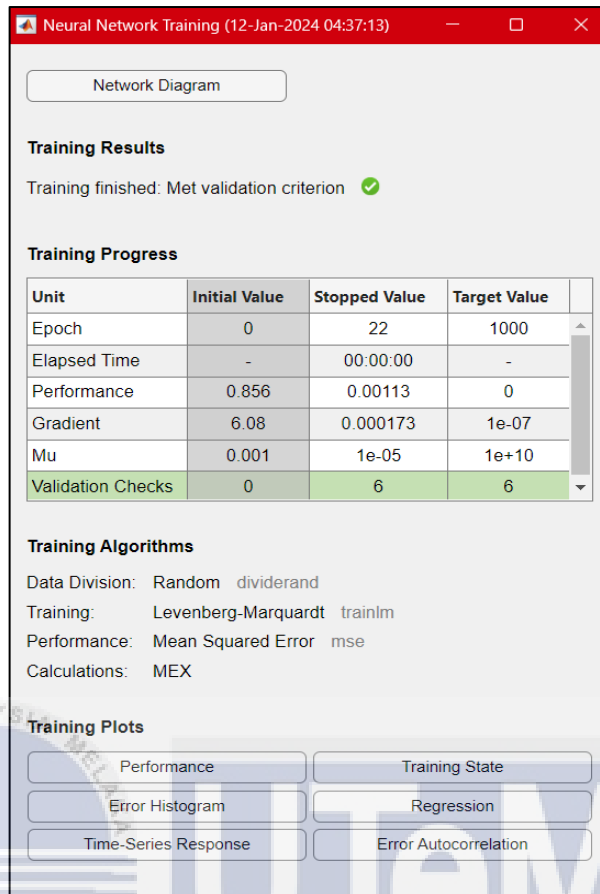


Figure 4.13: Neural Network Training for Levenberg-Marquardt algorithm that data taken for three days at Motor 2.

Figure 4.14 depicts the Mean Squared Error (MSE) over 16 epochs for training, validation, and testing data from a DS18B20 sensor. The Levenberg-Marquardt algorithm is using for curve-fitting problem, so in this context, it is used to minimize the error between the observed and predicted temperature readings from the DS18B20 sensor on day 3. From the graph, it can be seen that validation performance initially increases rapidly as the network learns from the data, but it starts to decrease until the highlighted epoch as the network was trained. Moreover, the best validation performance is 0.00060587, and it was achieved at epoch 16. This indicates that the network performed best on unseen data at the end of epoch 16.

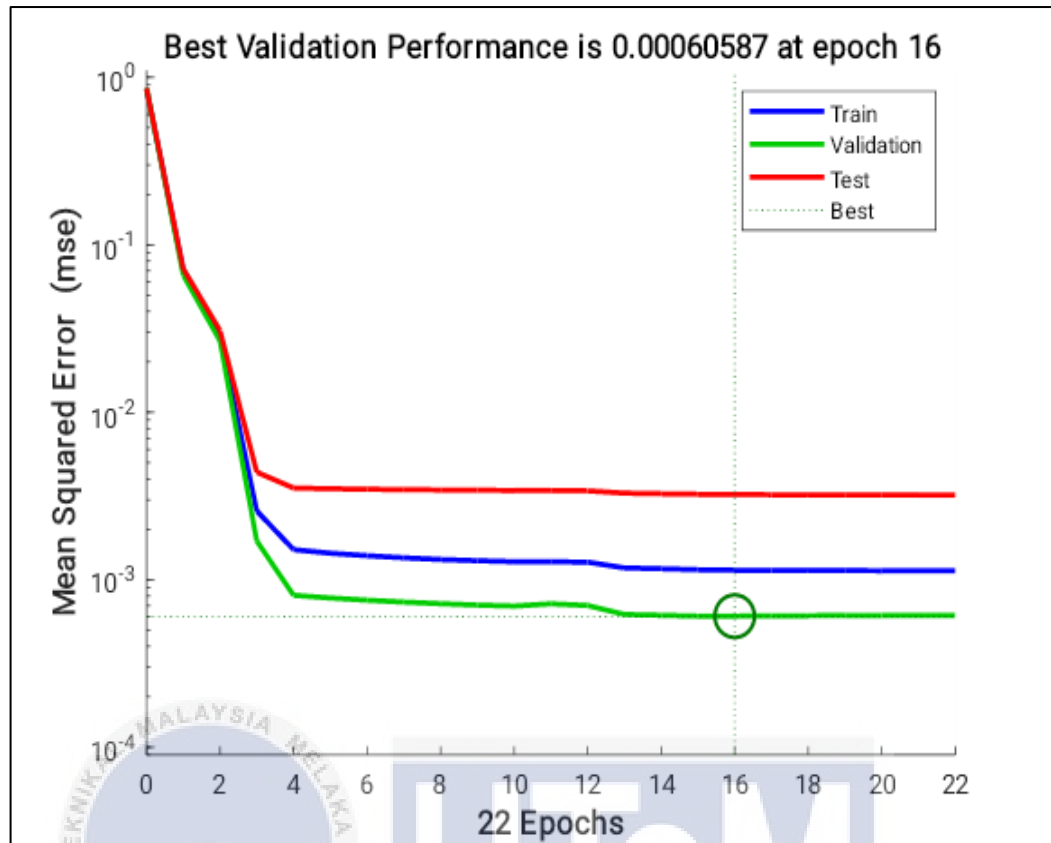


Figure 4.14: Validation performance for DS18B20 sensor using LM algorithm at Motor 2

Figure 4.15 illustrates the scatter plot that shows the relationship between two variables for regression graph for DS18B20 sensor for three days. It can be concluded that the regression line has a strong linear relationship where the R value of 0.9947 indicates a very strong linear relationship between the target and output variables. However, the presence of more scatter shows that the relationship might not be perfectly linear, and there could be some other factors influencing the output. Despite of having more scatter, the model can still predict the output values with very high accuracy, given a target value. The R value shows that the model explains 99.95% of the variance in the output variable.

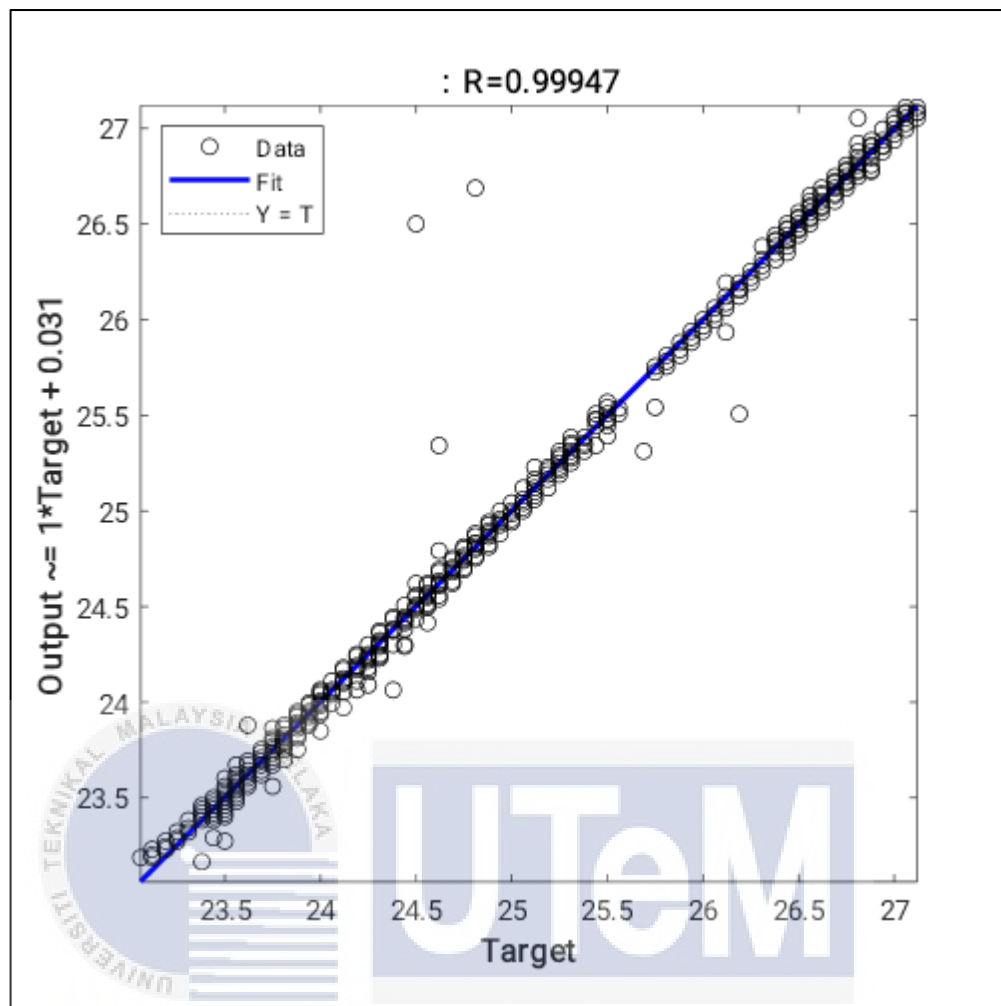


Figure 4.15: Regression graph for DS18B20 sensor using LM algorithm at Motor 2

A comparison between actual and predicted offset over time of the training network. The x-axis represents time in seconds (s), ranging from 0 to 12000, while the y-axis represents offset in millimeters (mm), ranging from approximately 23 to 27.5. Both lines show similar trends but have noticeable differences in their values at various points on the graph. There are three sections on the graph where both lines rise sharply, plateau, then fall sharply. Initially, both the actual and predicted offsets are closely aligned, but discrepancies become apparent as time progresses. Around 2000s, there is a sharp increase in both values but the predicted offset lags slightly behind the actual offset. Between 4000s to 8000s, the predicted offset is consistently higher than

the actual one. After 8000s, both values drop sharply with the predicted offset again lagging slightly. The related graph can be referred in Appendix C.

Figure 4.16 depicts the Mean Absolute Error (MAE) between the actual and predicted offset over time. The MAE values are mostly concentrated around 0, indicating a good model performance for those instances. However, there are noticeable spikes at certain points in time around 4000, and 9000, where the MAE nearly 2. This indicates that significant errors between the predicted and actual offset at these specific times,

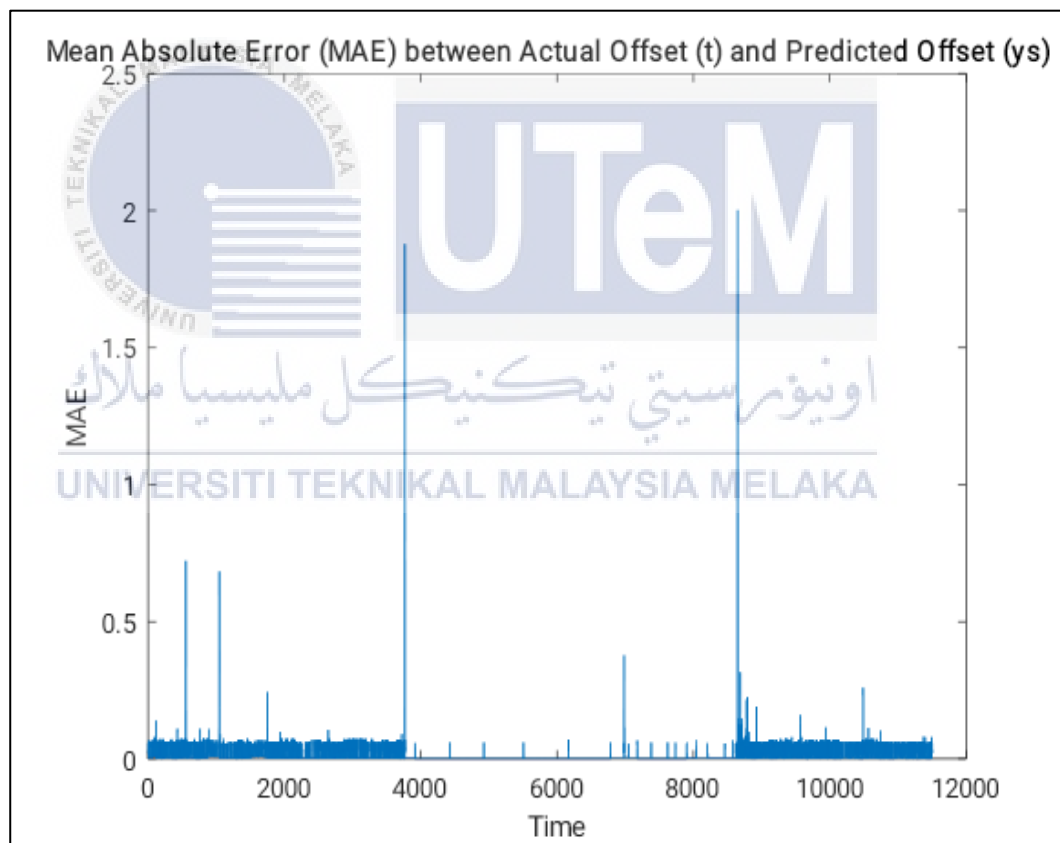


Figure 4.16: Graph OF Mean Absolute Error (MAE) between actual and predicted offset for DS18B20 sensor using LM algorithm at Motor 2

Table 4.4 shows the Mean Square Error (MSE) and regression values for training, validation, and testing phases. The MSE values are low for all three phases, indicating

a good fit of the model to the data. The R values are very close to 1 for all three phases as well, indicating a strong correlation between the predicted and actual outcomes. The graph can be referred to Appendix F.

Table 4.4: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of DS18B20 sensor using LM algorithm at Motor 2

	MSE	R
Training	0.0014	0.99955
Validation	0.000736	0.99976
Testing	0.00445	0.99877

4.4.1.5 ADXL345_1 data for 3 days at Motor 2

For motor 2 that is produced at year 2007, the data from ADXL345 sensors also being collected to make a predictive analytic using Levenberg-Marquardt algorithm. From the training results of neural network, the training stopped at 13 epochs, with a performance of 2.28. The gradient is at 0.534, indicating the rate of changes in error with respect to the weights and biases in the network. The low mu value suggests that adjustments made during training were small, leading to a more refined model fit. Initially, the mu values are at 0.01 and stopped at an extremely small value of $1e+10$ indicating precision in adjustments during training. The neural network of training results for ADXL345 sensors can be referred in Appendix A.

Figure 4. 17 depicts the validation performance of the model. The data from ADXL345 sensor is used as input to make a prediction about future data points or trends. The graph has shown the MSE error over 13 epochs. As for this sensor, the best validation performance is 2.4891 at epoch 7 which have been marked by a circle on the graph. This indicates that at this point, the mode has achieved its lowest

validation error. Initially, the MSE values is increases rapidly as the network learns from the data, but later it starts to decrease.

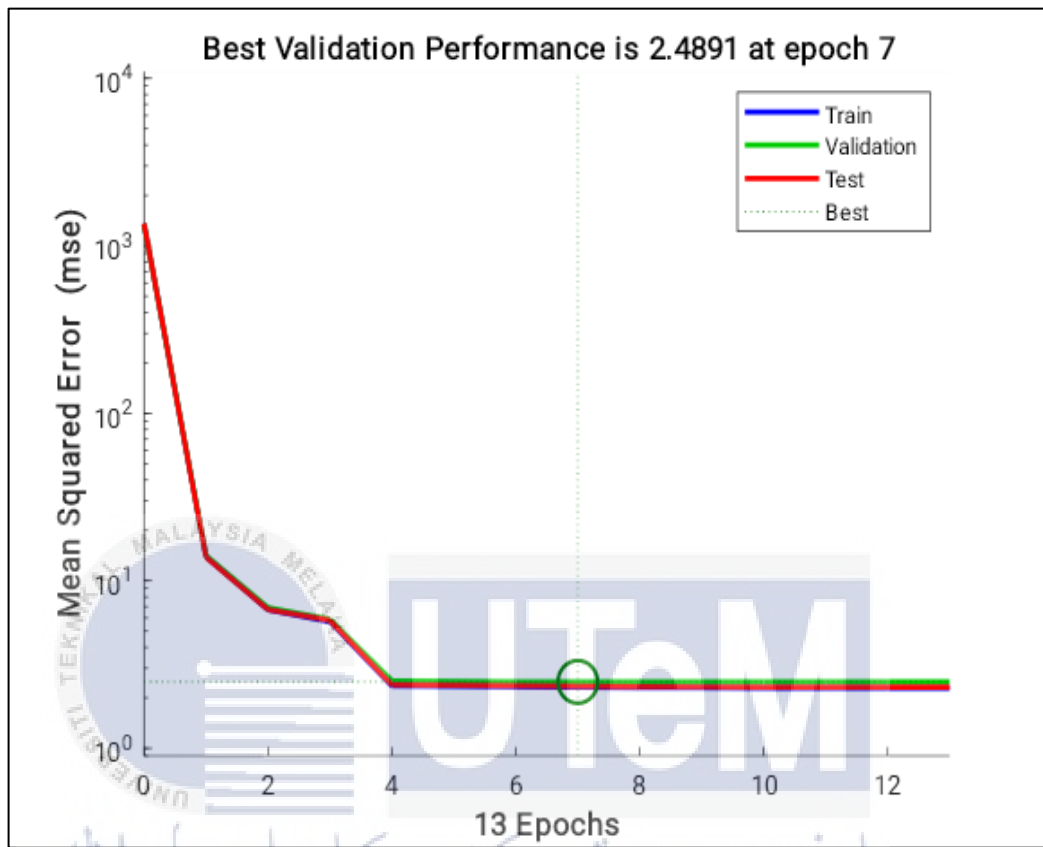


Figure 4.17: Validation performance for ADXL345_1 using LM algorithm at Motor 2

Figure 4.18 illustrates the regression plot for data taken from ADXL345_1 at Motor 2. The scatter plot shows the relationship between the target and output variables, with a high correlation coefficient of $R=0.94135$, indicating a strong linear relationship. The blue line represents the fit of the LM algorithm to the data, while the dotted line represents perfect prediction. As for the predictive analytics using LM algorithm, the relationship would involve using ADXL345 sensor's data from Motor 2 as input to make predictions about future data points or trends. To conclude with, the LM algorithm is a good fit for the data and has an excellent predictive power and efficiency.

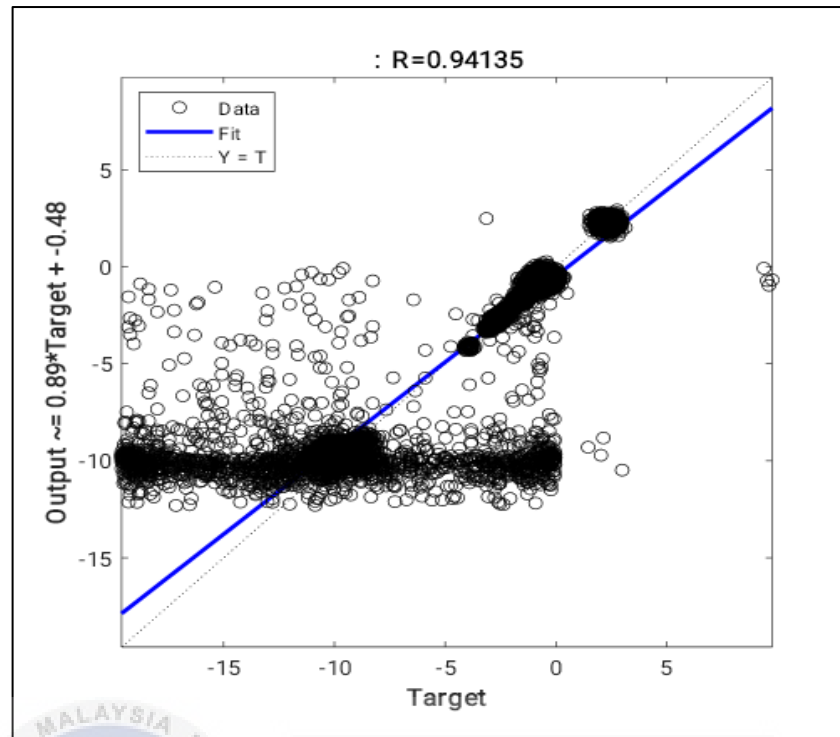


Figure 4.18: Regression graph for ADXL345_1 sensor using LM algorithm at Motor 2

Table 4.5 shows the Mean Square Error (MSE) and regression values for training, validation, and testing phases. The R values are very close to 1 for all three phases, indicating a strong correlation between the predicted and actual outcomes. However, there is an increase in MSE during validation which suggests that the model might be overfitting to the training data. The regression graph for each phase can be referred to Appendix F.

Table 4.5: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_1 using LM algorithm at Motor 2

	MSE	R
Training	1.73	0.94199
Validation	3.01	0.93787
Testing	2.45	0.94188

4.4.1.6 ADXL345_2 data for 3 days at Motor 2

The neural network training algorithm for ADXL345_2 is also being analyzed. From observation and analyzation of the data, the network was trained for 107 epochs, where it represents one complete pass through the entire training dataset. The network was trained for 107 epochs, so it means that it repeatedly learned from the training data 107 times. The values for gradient also decreased significantly, which indicates that the network is no longer train anything new from the training data. As for the mu values, it is also decreased, from 0.001 to 0.0001, which indicates that the algorithm is becoming more confident in its results. The neural network training results can be referred to Appendix A.

Figure 4.19 illustrates the Mean Squared Error (MSE) across 107 epochs for training, algorithm, validation, and testing data obtained from a DS18B20 sensor. Employing the Levenberg-Marquardt algorithm for curve-fitting, it aims to minimize the disparity between observed and predicted temperature readings from the DS18B20 sensor for three "days." The graph depicts a rapid initial increase in validation performance as the network learns from the data, followed by a subsequent decline until the highlighted epoch during the training phase. Notably, the optimal validation performance, reaching 0.50183, was attained at epoch 101. This suggests that the network demonstrated its highest performance on unseen data towards the conclusion of epoch 101.

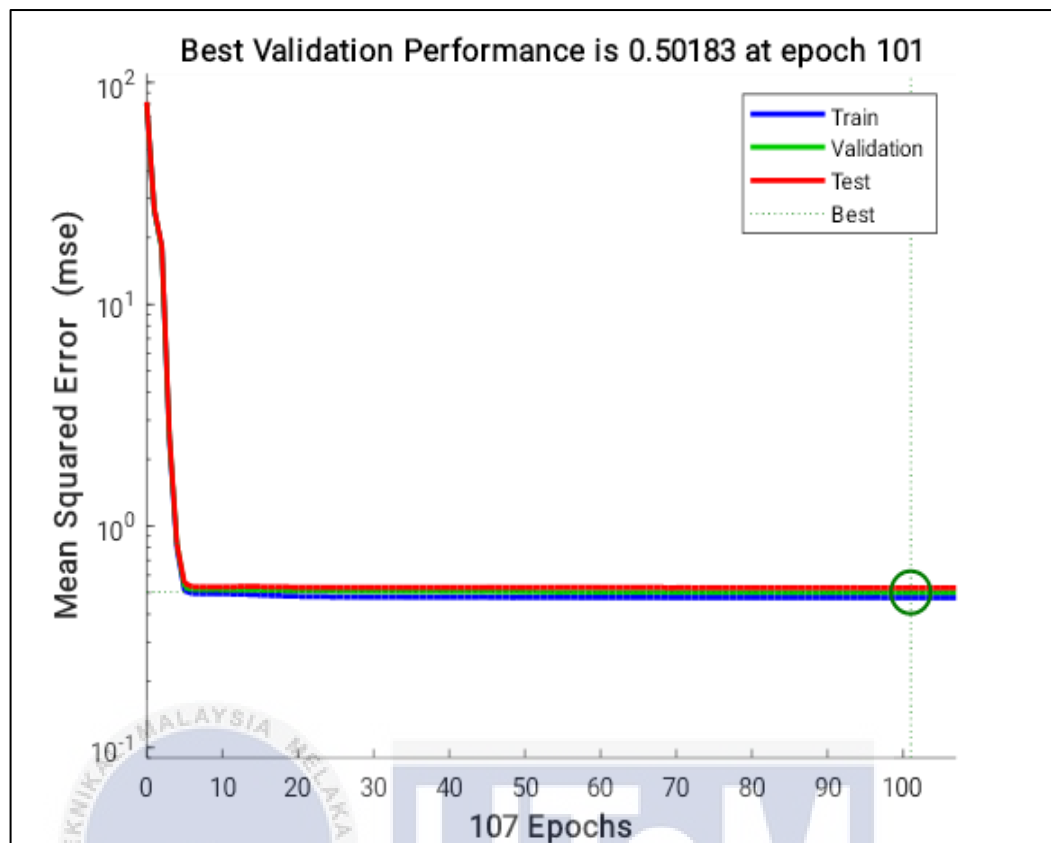


Figure 4.19: Validation performance for ADXL345_2 using LM algorithm at Motor 2

Meanwhile, Figure 4.20 illustrates a regression graph spanning three days, incorporating data from the ADXL345 sensor. The x-axis denotes the actual temperature readings retrieved from the ADXL345 sensor, while the y-axis represents temperature values derived through the scaled conjugate gradient algorithm. Individual data points are depicted as circles, and a linear fit, represented by the blue line, emphasizes a robust positive correlation with an R-value of 0.98858. This indicates a proportional increase in output values as the target values rise, signifying a strong positive relationship. In the realm of predictive analytics, the elevated correlation coefficient implies that the model exhibits high accuracy in forecasting future data points based on historical data.

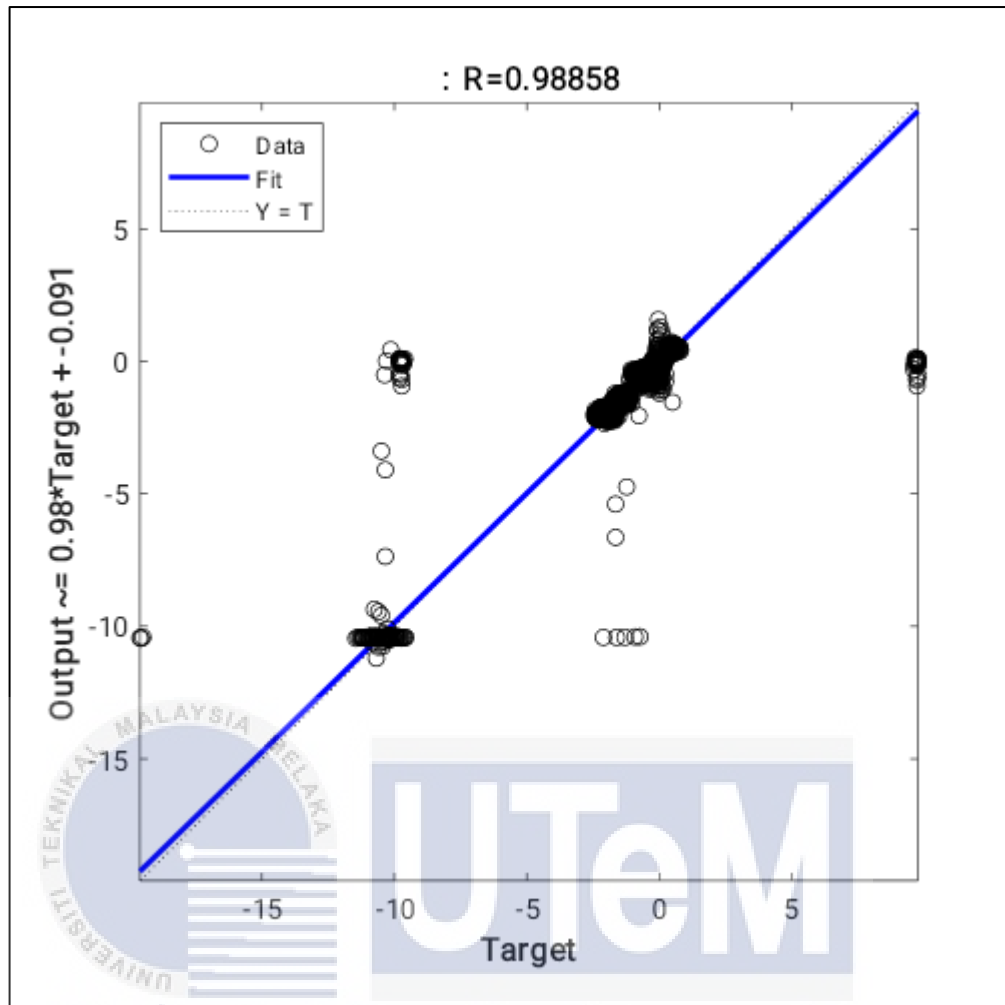


Figure 4.20: Regression graph for ADXL345_2 sensor using LM algorithm at Motor 2

Table 4.6 presents the Mean Square Error (MSE) and regression values during the training, validation, and testing phases. MSE quantifies the average squared difference between predicted and actual values, while the R-value gauges the correlation between these values. The consistently low MSE values across all phases signify a well-fitted model, demonstrating superior performance with minimal discrepancies between predicted and actual outcomes. Similarly, the R-values, approaching 1 for each phase, underscore a robust correlation between predicted and actual outcomes, affirming the model's excellent fit. The graph can be referred in Appendix F.

Table 4.6: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_2 sensor using LM algorithm at Motor 2

	MSE	R
Training	0.411	0.98884
Validation	0.479	0.98838
Testing	0.614	0.98756

4.4.2 Scaled Conjugate Gradient Train Performance

Other than using the Levenberg-Marquardt training algorithm, the Scaled Conjugate Gradient training algorithm is also being used to analyze and process data from sensors. This is because to make a comparison between two methods, which one is the best one to use for data analysis. The Scaled Conjugate Gradient technique is a gradient-based method that is commonly used for training artificial neural networks.

4.4.2.1 DS18B20 data for 3 data at Motor 1

Figure 4.21 depicts the neural network from data of DS18B20 sensor using Scaled Conjugate Gradient algorithm. From observation and analysis of the data, the network was trained for 62 epochs, where it represents one complete pass through the entire training dataset. Initially, the network was trained for 62 epochs, while the target was set to 1000 epochs, so it means that it repeatedly learned from the training data 62 times. As for the performance, the initial value was 0.992 and it improved significantly to a stopped value of 0.00205, with a target value of 0. The values for gradient also decreased significantly, which indicates that the network is no longer train anything new from the training data. Where it started at a value of 4.31 and reduced to 0.00298, approaching towards a very small target value (1e-06).

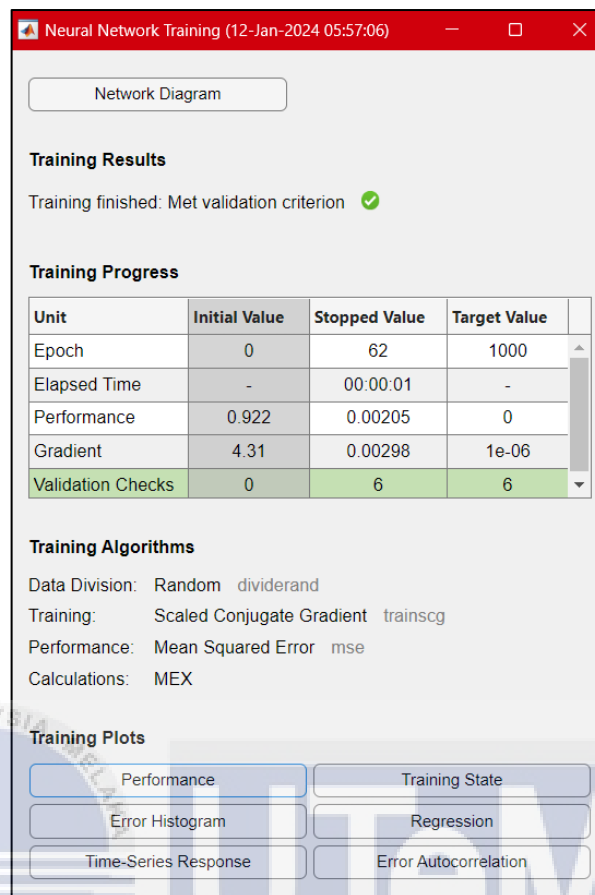


Figure 4.21: Neural Network Training for Scaled Conjugate Gradient algorithm that data taken for three days at Motor 1.

Figure 4.22 depicts the Mean Squared Error (MSE) over 62 epochs for training, validation, and testing data from a DS18B20 sensor. The Levenberg-Marquardt algorithm is used for curve-fitting problems, so in this context, it is used to minimize the error between the observed and predicted temperature readings from the DS18B20 sensor on day 3. From the graph, it can be seen that validation performance initially increases rapidly as the network learns from the data, but it starts to decrease until the highlighted epoch as the network was trained. Moreover, the best validation performance is 0.0017247, and it was achieved at epoch 56. This indicates that the network performed best on unseen data at the end of epoch 56.

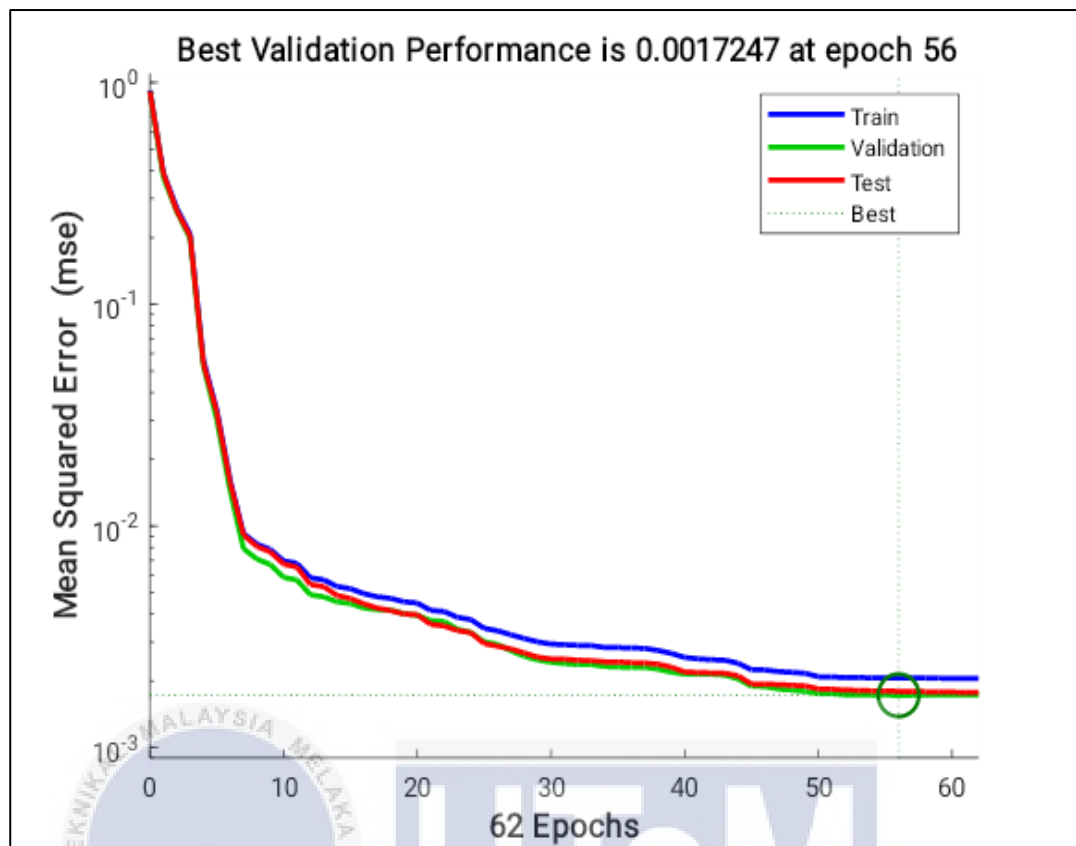


Figure 4.22: Validation performance for DS18B20 sensor using SCG algorithm at Motor 1

Meanwhile Figure 4.23 represents regression graph over three days for data from DS18B20 sensors too. The x-axis represents the actual temperature readings from the DS18B20 sensor. While the y-axis indicates the temperature values obtained through the scaled conjugate gradient algorithm. The circles represent individual data points, and the blue line is a linear fit to this data, indicating a strong positive correlation with an R-values of 0.99552. It means that as the target values increase, the output values also increase proportionally. In predictive analytics, this high correlation coefficient suggests that the model is highly accurate in predicting future data points based on historical data.

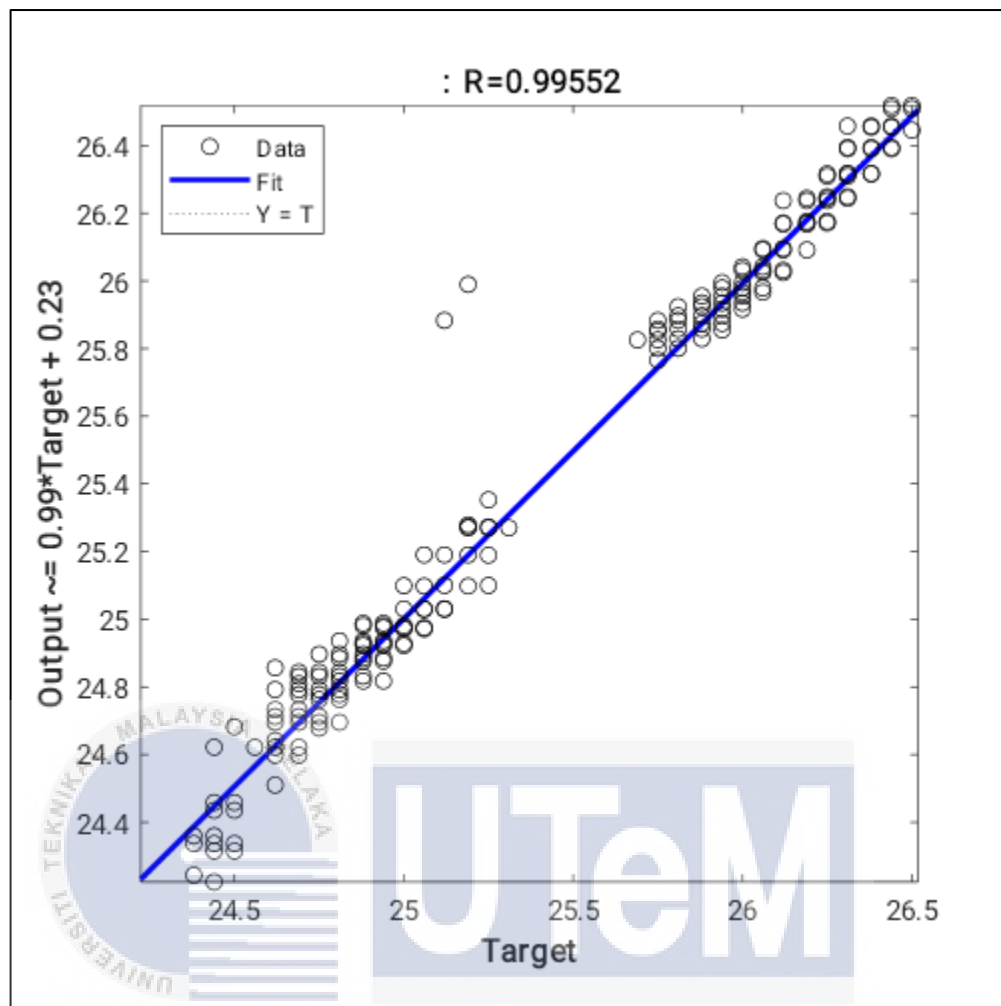


Figure 4.23: Regression graph for DS18B20 sensor using SCG algorithm at Motor 1

Figure 4.24 illustrates the graph of a comparison between actual and predicted offset over time. The actual offset is depicted in red and exhibits a sharp decline at 26.5mm before stabilizing at 25.2 mm and then dropping again. In contrast, the predicted offset, shown in blue, closely follows the actual trend but with slight deviations with time observation. The predicted offset remains relatively constant throughout, with minor fluctuations but not mirroring the sharp decline of the actual offset. Furthermore, this type of graph is crucial for evaluating the performance of predictive models like SCG. It allows analysts to visually assess how accurately a model's predictions align with actual outcomes over a specified or under certain

conditions. The closer these two lines are throughout the graph, the more accurate and reliable the predictive model is considered to be. These generated graphs are utilizing sensors data at Motor 1 for three days.

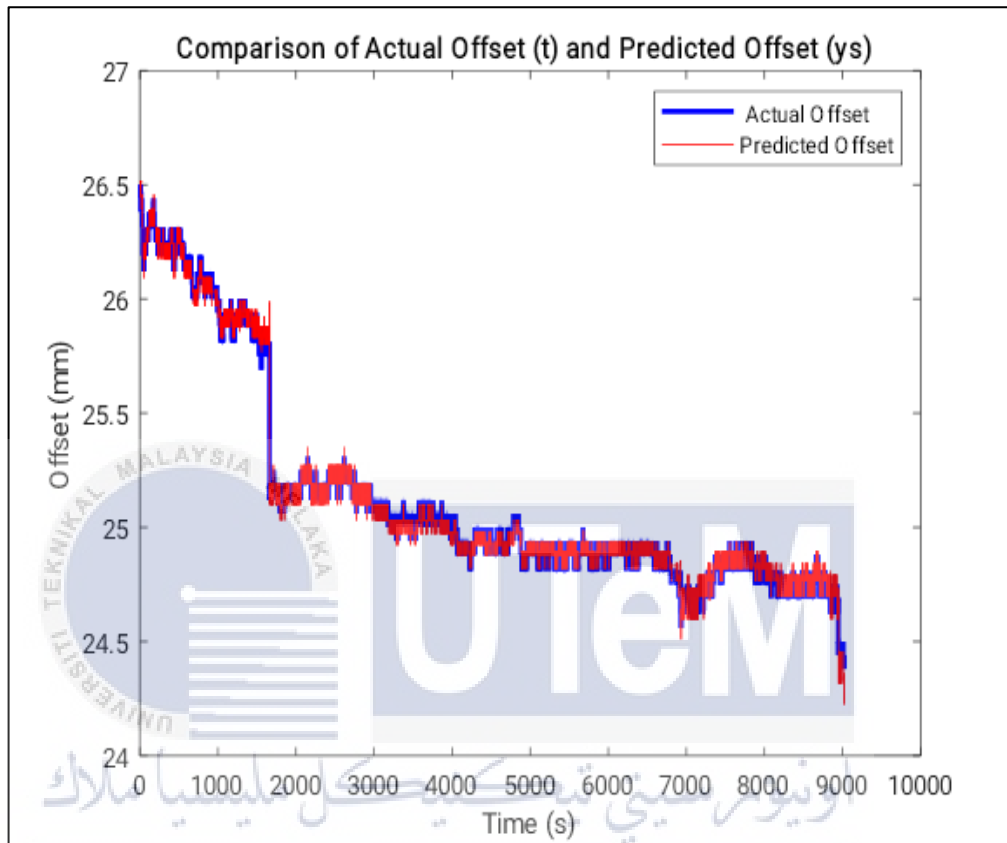


Figure 4.24: Graph for comparison between actual and predicted offset using SCG algorithm at Motor 1

Figure 4.25 depicts the Mean Absolute Error (MAE) using the SCG algorithm over a time period. The MAE is a measure used to quantify how accurate predictions are, it measures the average magnitude of errors between predicted and observed values, without considering their direction. A blue line on the graph depicts fluctuations in MAE over time. In predictive analytics, a lower MAE indicates more accurate predictions. Moreover, in this graph, for most of the time periods, the MAE is relatively low but there are spikes indicating instances where the predicted offsets were significantly different from actual offsets. This could imply that while SCG

algorithm is generally effective, there might be certain conditions or data patterns where its performance decreases.

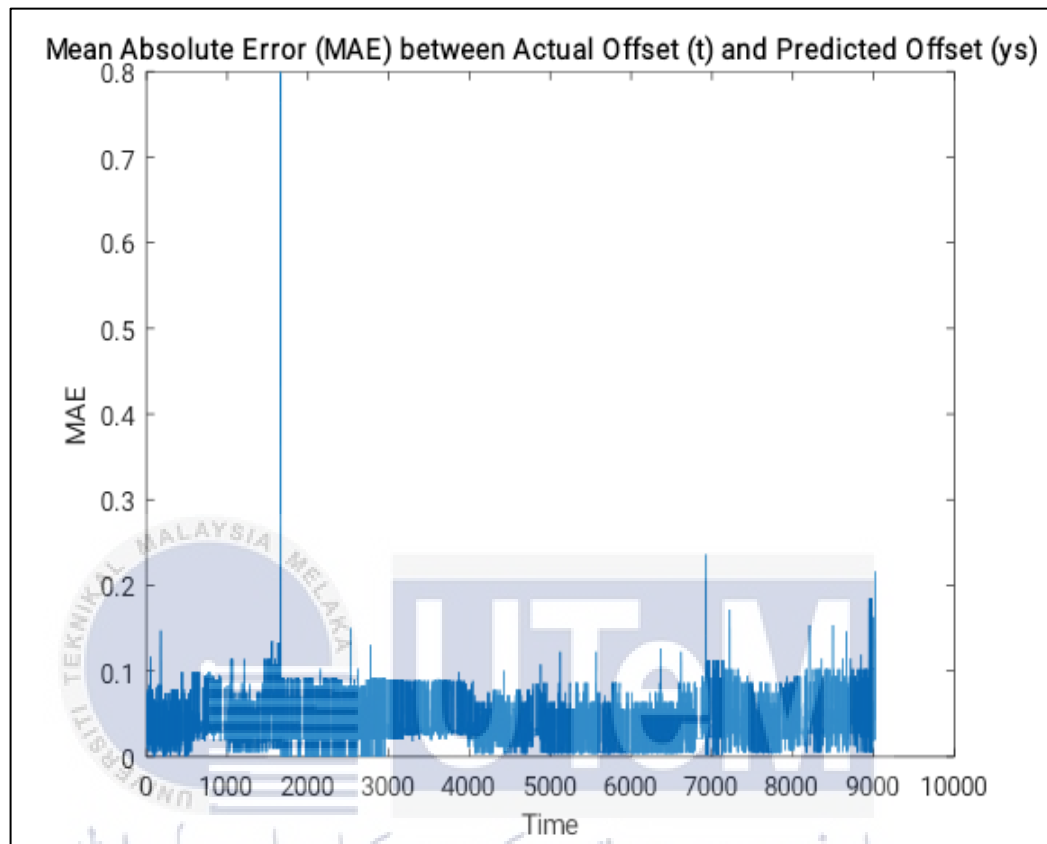


Figure 4.25: Graph of Mean Absolute Error (MAE) between actual and predicted offset for DS18B20 sensor using SCG algorithm at Motor 1

Table 4.7 shows the Mean Square Error (MSE) and regression values for training, validation, and testing phases. The MSE values are low for all three phases, indicating a good fit of the model to the data. The R values are very close to 1 for all three phases as well, indicating a strong correlation between the predicted and actual outcomes. The graph can be referred to Appendix G.

Table 4.7: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of DS18B20 sensor using SCG algorithm at Motor 1

	MSE	R
Training	0.00252	0.99531
Validation	0.00176	0.99609
Testing	0.00192	0.99595

4.4.2.2 ADXL345_1 data at Motor 1

The data acquired from ADXL345 sensors underwent analysis using the SCG algorithm. To facilitate data processing, the information from three days was consolidated. For Motor 1's ADXL345 sensor, the neural network underwent training for 56 epochs, representing complete passes through the training dataset. The decreasing gradient values signify diminishing learning from the training data. Additional details about this data illustrations can be found in Appendix B.

The accompanying Figure 4.26 illustrates the Mean Squared Error (MSE) concerning the number of epochs in the predictive analytics model's training, validation, and testing phases. The logarithmic scale depicts MSE, revealing a substantial decrease and subsequent stabilization as the epochs increase. The epoch 56 marks the best validation performance, denoted by the lowest MSE of 2.5909. The MSE values for training, validation, and testing start high but consistently decline with increasing epochs.

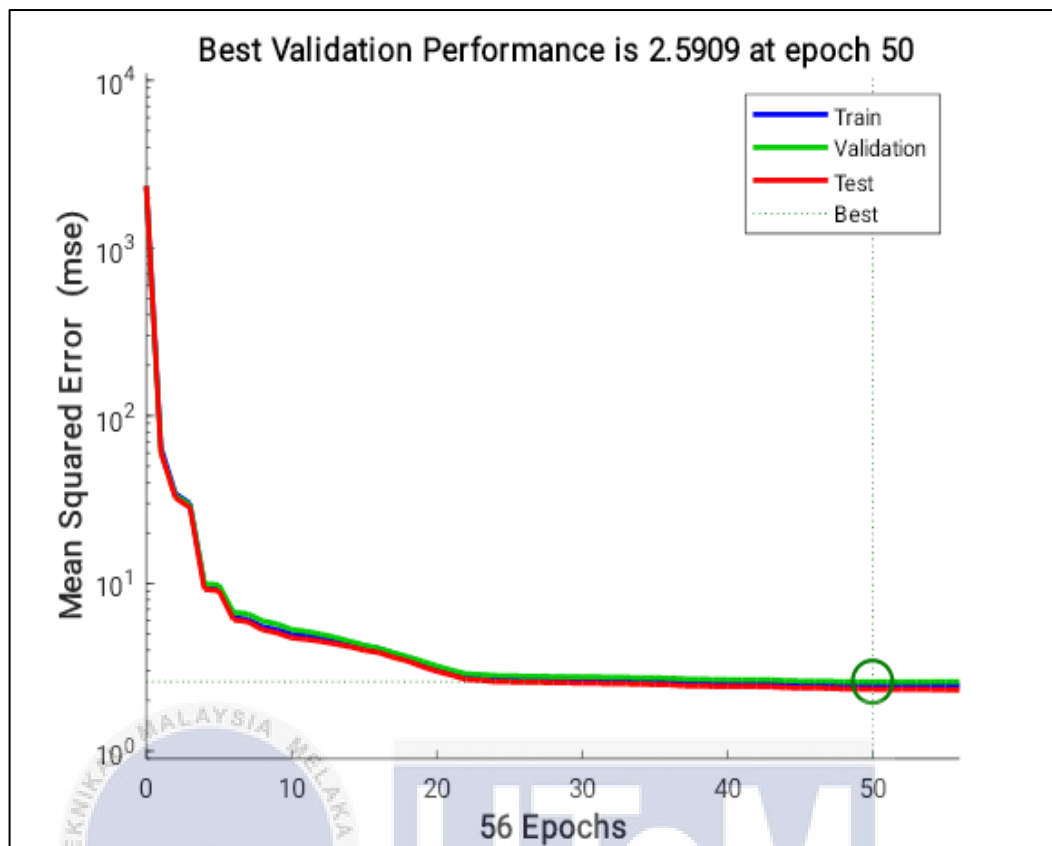


Figure 4.26: Validation performance for ADXL345_1 sensor using SCG algorithm at Motor 1

Figure 4.27 visualizes the regression analysis conducted with the SCG algorithm, portraying the connection between target and output variables. The R-squared value of 0.96443 signifies a remarkably robust positive correlation, signifying that the model elucidates 96.44% of the variability in the dependent variable through the independent variable. In the context of predictive analytics, this elevated R-squared value implies the model's outstanding predictive capability and accuracy. Consequently, the model is well-suited for accurately predicting future values of the dependent variable.

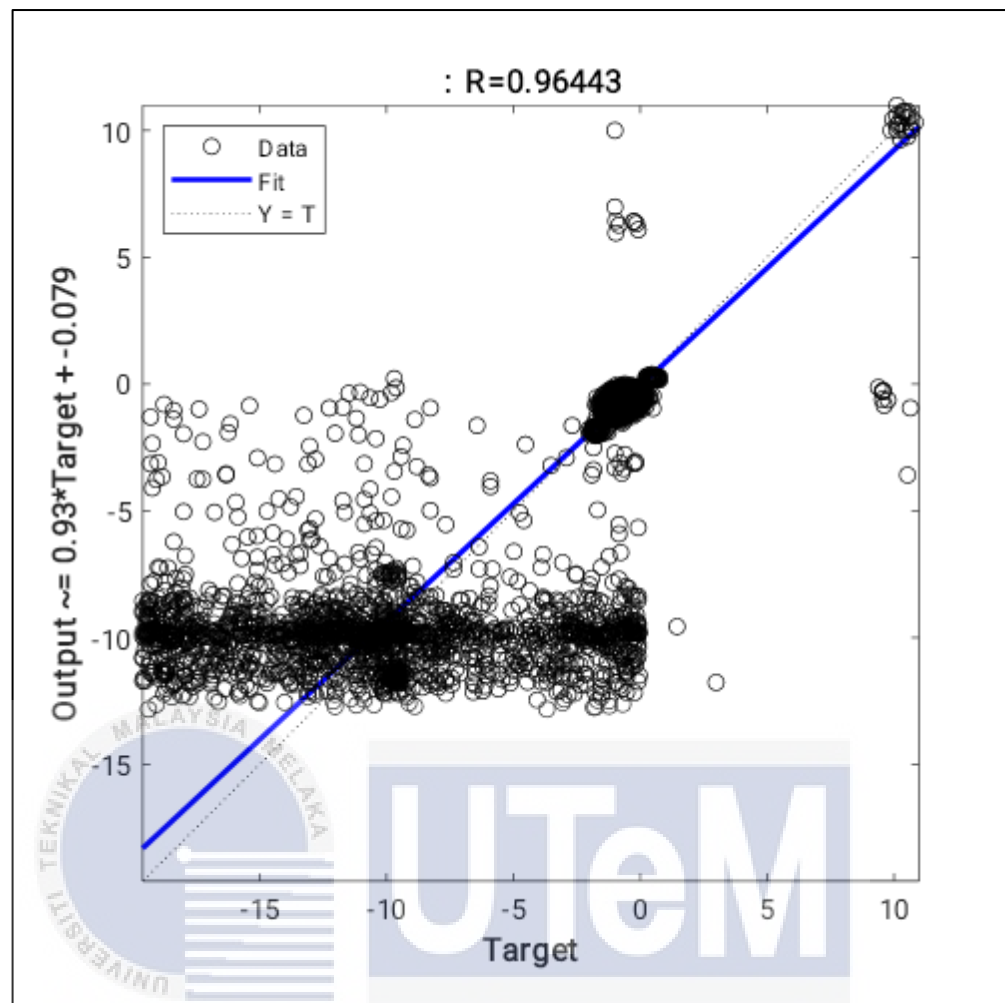


Figure 4.27: Regression graph for ADXL345_1 sensor using SCG algorithm at Motor 1

Table 4.8 presents the values corresponding to the training, validation, and testing phases of the model. Notably, lower MSE values across all phases signify a superior fit of the model to the data. The training phase exhibits an MSE of 2.57, indicating a relatively commendable fit. Conversely, the validation phase displays a higher MSE of 3.52, implying that the model didn't perform as effectively on unseen data. The R values for each phase are included in the table, and it's evident that all R values are close to 1, denoting a robust positive correlation. The related graph can refer in Appendix G.

Table 4.8: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_1 sensor using SCG algorithm at Motor

	MSE	R
Training	2.57	0.96463
Validation	3.52	0.9621
Testing	2.05	0.96578

4.4.2.3 ADXL345_2 data at Motor 1

The second ADXL345 sensor was also analyzed using the SCG algorithm. Despite being the same sensor, variations in sensor placement and readings occurred. Consequently, both ADXL345 sensors' readings were recorded and analyzed using the same training algorithm. For this second ADXL345 sensor, upon observing and analyzing the data, the network underwent training for 102 epochs, surpassing the number for the first ADXL345 sensor. The gradient values experienced a significant decrease, signifying that the network ceased learning anything new from the training data. Further details on this data are available in Appendix B.

The accompanying Figure 4.28 illustrates the best validation performance, identified by the lowest mean squared error (MSE), achieved at epoch 96 with an MSE of 6.12. The graph portrays the MSE trends over 102 epochs for the training, validation, and testing phases. The x-axis is labeled "102 epochs," with increments of ten from 0 to 100. The MSE sharply decreases in the initial epochs and subsequently stabilizes. The circled point on the graph represents the best validation performance, which is 6.12 at epoch 96. This signifies that at this juncture, the model has attained its lowest validation error and is likely the most generalized version before the risk of overfitting begins.

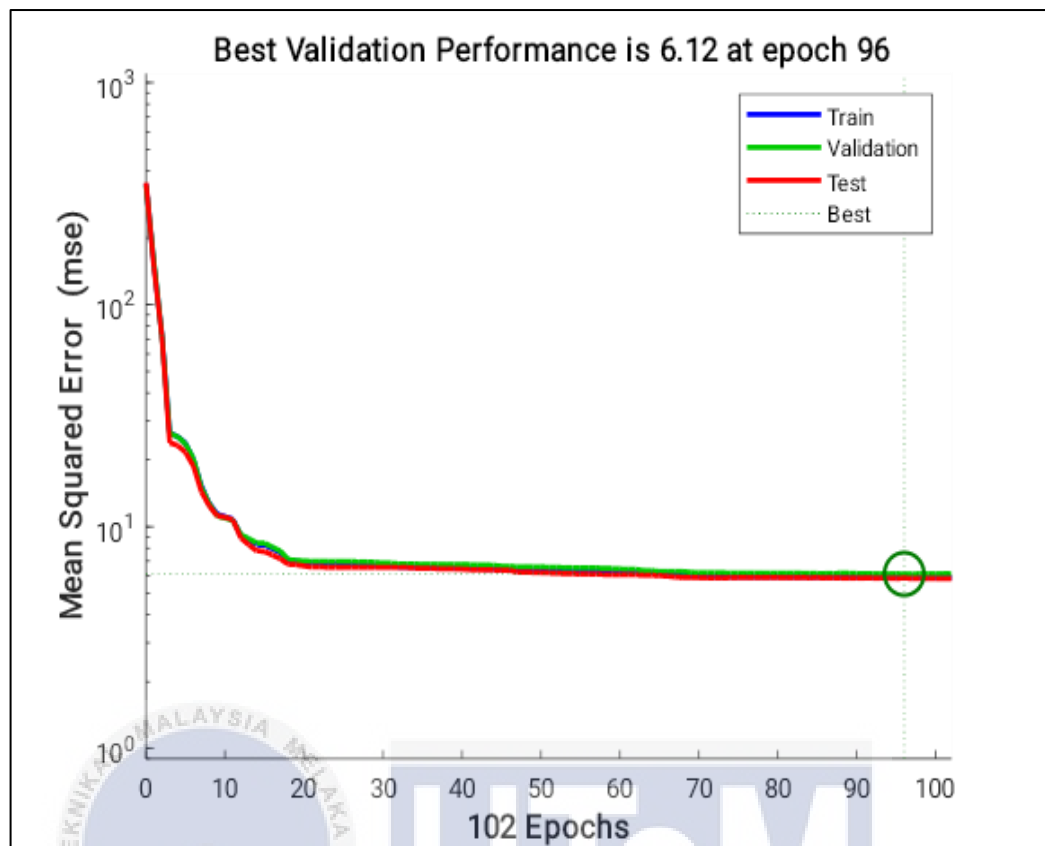


Figure 4.28: Validation performance for ADXL345_2 sensor using SCG algorithm at Motor 1

Figure 4.29 depicts the regression analysis employing the SCG algorithm for the second ADXL345 sensor. It showcases the model's alignment with the data points, illustrating the interplay between the LM algorithm and predictive analytics. The R-value of 0.9244 signifies a robust positive correlation, indicating that as the target variable rises, the output similarly increases. In terms of performance, the elevated R-value implies that the SCG algorithm excels in predicting outcomes based on the provided input data.

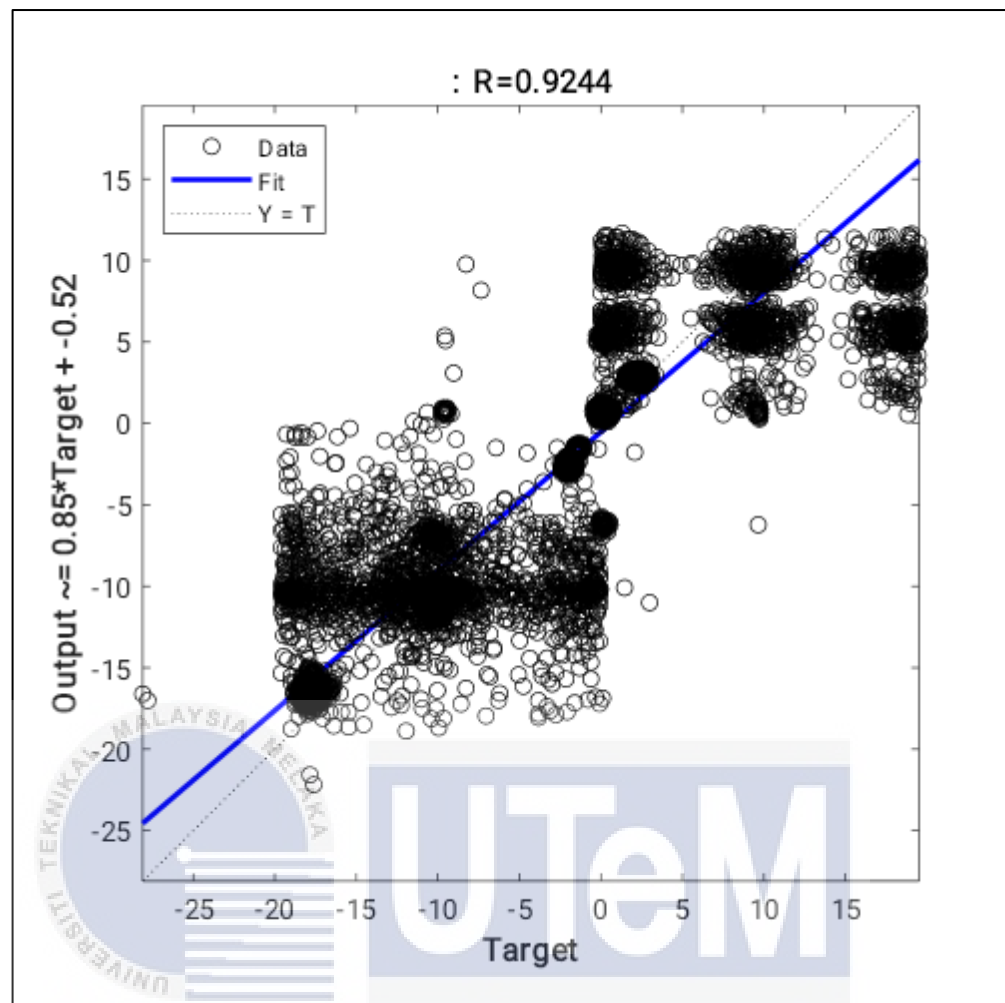


Figure 4.29: Regression graph for ADXL345_2 sensor using SCG algorithm at Motor 1

In the Table 4.9, the LM algorithm's MSE and regression values for the three phases are presented. The MSE values are 6.09, 7.48, and 5.00 for training, validation, and testing, respectively. Notably, the MSE values for validation show an increase, whereas for testing, there is a decrease. The lowest MSE in testing indicates that the model has generalized effectively from its training data. Regarding the R values, all are close to 1, signifying a robust positive relationship between predicted and actual values in all phases. In conclusion, the effectiveness of the SCG algorithm is evident, as indicated by the high R values. These high R values affirm the model's reliability

in producing trustworthy predictions, showcasing strong correlations with actual outcomes. The regression plots for each phase can be found in the Appendix G.

Table 4.9: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_2 sensor using SCG algorithm at Motor 1

	MSE	R
Training	2.57	0.96463
Validation	3.52	0.9621
Testing	2.05	0.96578

4.4.2.4 DS18B20 data at Motor 2

After completing analyze data for Motor 1, the sensors data for Motor 2 is then analyzed using the same technique which is Scaled Conjugate Gradient training algorithm. The purpose of data processing is to make a predictive analysis between two motors over three days of testing. Figure 4. 30 illustrates the results of training a neural network using the SCG algorithm. The training stopped at 73 epochs, with a target of reaching 1000 epochs. There were initially zero checks and it stopped after performing six checks.

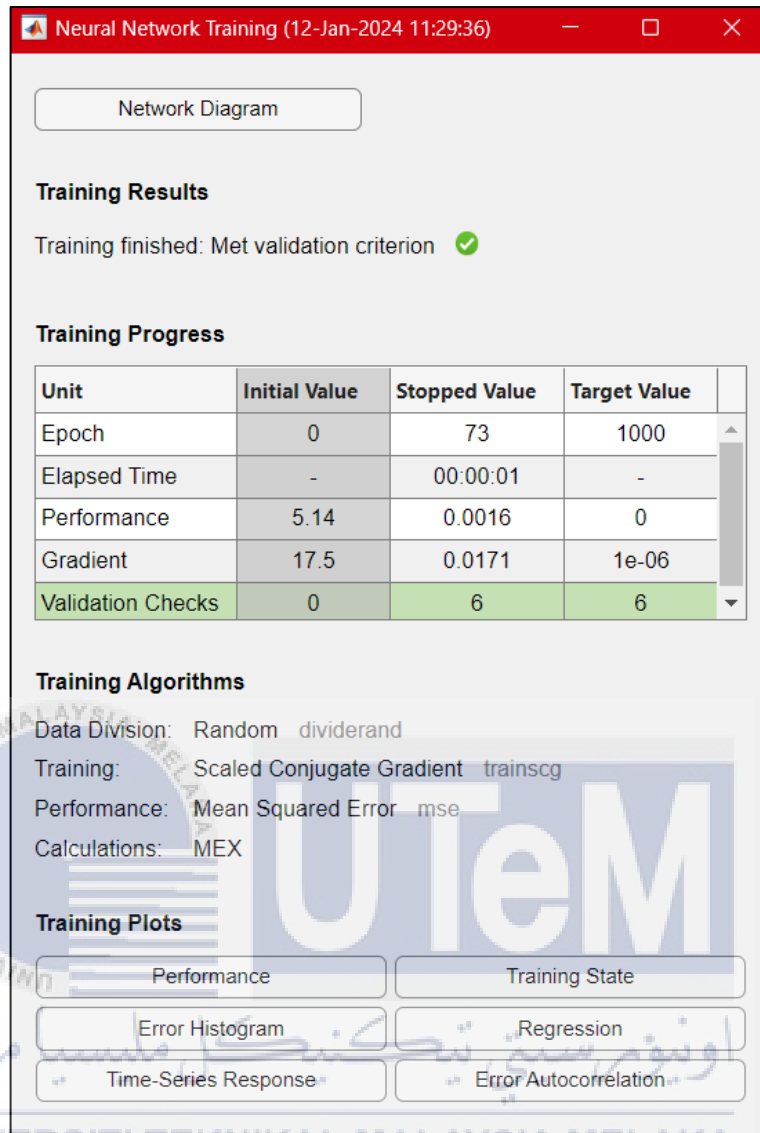


Figure 4. 30: Neural Network Training for Scaled Conjugate Gradient algorithm that data taken for three days at Motor 2.

Figure 4.31 illustrates the Mean Squared Error (MSE) of a model trained on data taken from DS18B20 sensor over 73 epochs. Can be seen that the MSE for training, validation, and testing datasets are plotted. The best validation performance, marked by a dotted line, is achieved at epoch 67 with an MSE of 0.0035358.

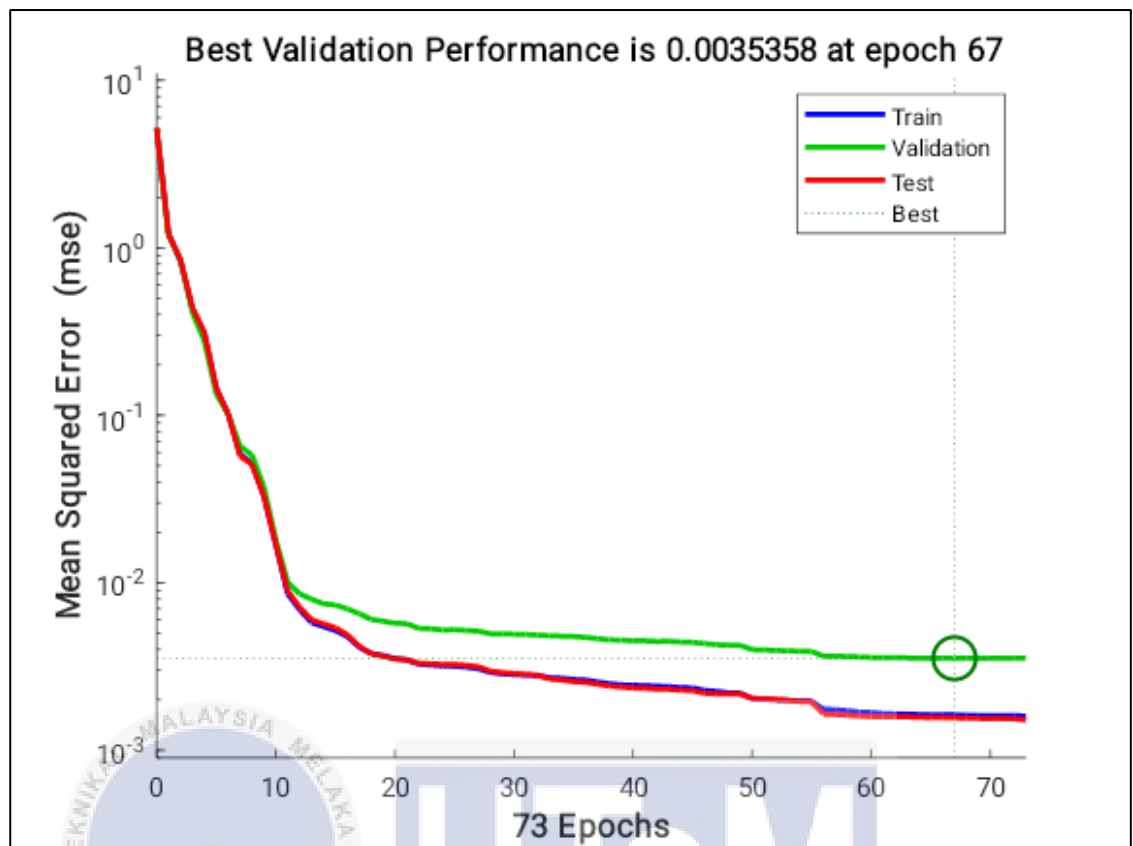


Figure 4.31: Validation performance for DS18B20 sensor using SCG algorithm at Motor 2

Next is for regression graph that can be seen at Figure 4.32. The blue line represents the model's predictions, R , from graph can be seen that R value is close to 1, indicates that a very strong positive linear relationship between the predicted and actual values. This is because, the closed the points are to the blue line, the better the model's predictions.

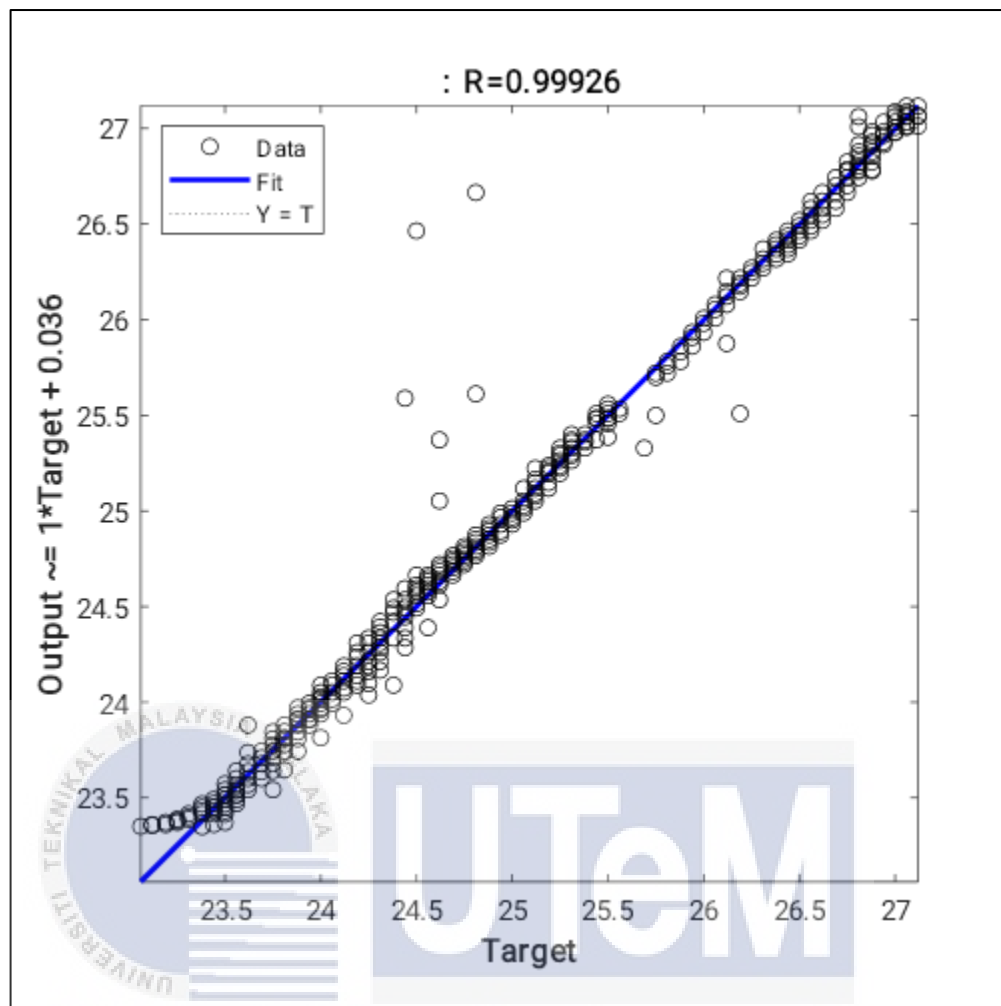


Figure 4.32: Regression graph for DS18B20 sensor using SCG algorithm at Motor 2

The predicted analytics involved here would be utilizing historical data to forecast future offset values. In this case, it appears that the predictive model was not entirely accurate, as there are noticeable discrepancies between the actual and predicted offsets, especially from 4000s to 8700s time mark. This can be referred to Appendix I.

Next, Figure 4.33 shows graph of the Mean Absolute Error (MAE). This graph indicates the average magnitude of errors between predicted and actual offsets over time. The spike in MAE shows the predicted model's performance deviated significantly from the actual data, indicating a need for model refinement to improve

accuracy. From the graph, can observed that there are several spikes in MAE visible on the graph, with two prominent ones reaching close to a MAE of 1.8 around times 3800 and 2 around 8800 respectively. Most of the errors are clustered near the bottom, indicating a low mean absolute error for most of the predictions.

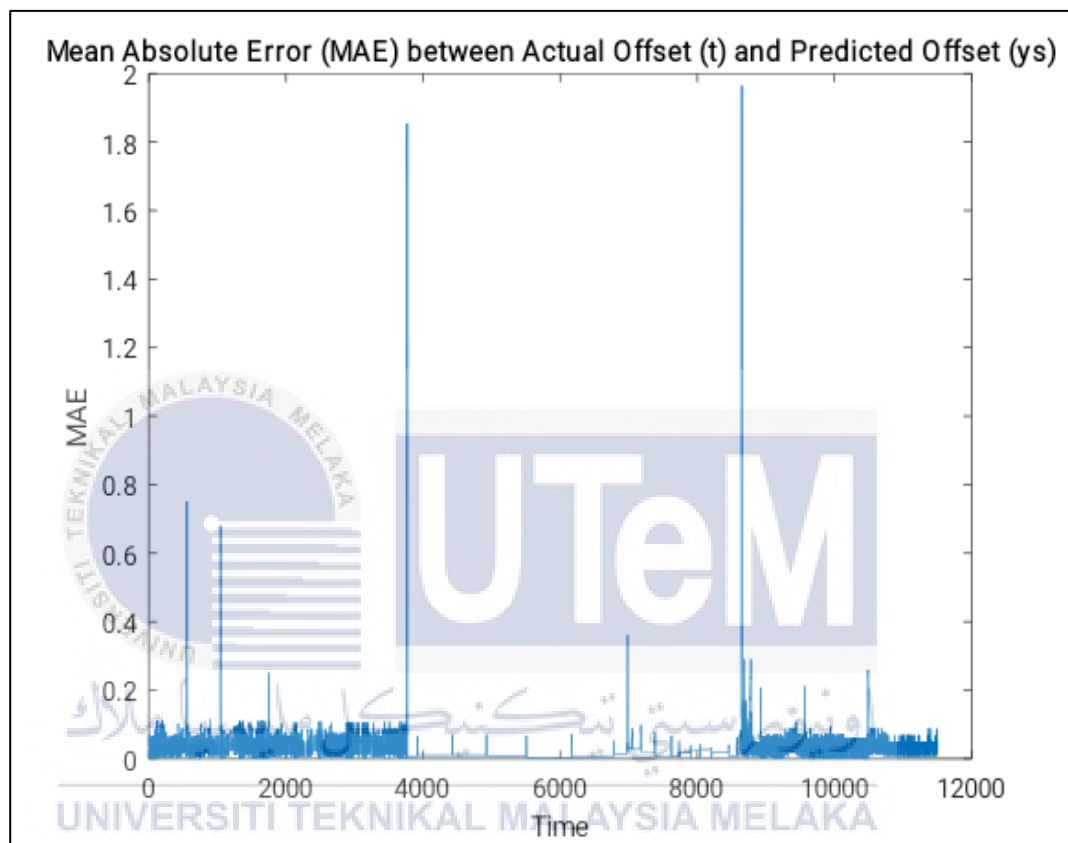


Figure 4.33: Graph of Mean Absolute Error (MAE) between actual and predicted offset for DS18B20 sensor using SCG algorithm at Motor 2

Table 4.10 shows the Mean Square Error (MSE) and regression values for training, validation, and testing phases. The MSE values are low for all three phases, indicating a good fit of the model to the data. The R values are very close to 1 for all three phases as well, indicating a strong correlation between the predicted and actual outcomes. The graph can be referred to Appendix H.

Table 4.10: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of DS18B20 sensor using SCG algorithm at Motor 2

	MSE	R
Training	0.00185	0.99937
Validation	0.00452	0.9986
Testing	0.00135	0.99939

4.4.2.5 ADXL345_1 data at Motor 2

For motor 2 that is produced at year 2007, the data from ADXL345 sensors also being collected to make a predictive analytic using Scaled Conjugate Gradient algorithm. From the training results of neural network, the training stopped at 92 epochs, with a performance of 2.4. The gradient is at 0.174, indicating the rate of changes in error with respect to the weights and biases in the network. The neural network of training results for ADXL345 sensors can be referred in Appendix B.

Figure 4.34 illustrates the validation performance of the model using data from the ADXL345 sensor as input for predicting future data points or trends. Displaying the MSE error over 92 epochs, the graph highlights the best validation performance at epoch 86, marked by a circle. This signifies the point where the model has attained its lowest validation error. Initially, the MSE values increase rapidly as the network learns from the data, but subsequently, they start to decrease.

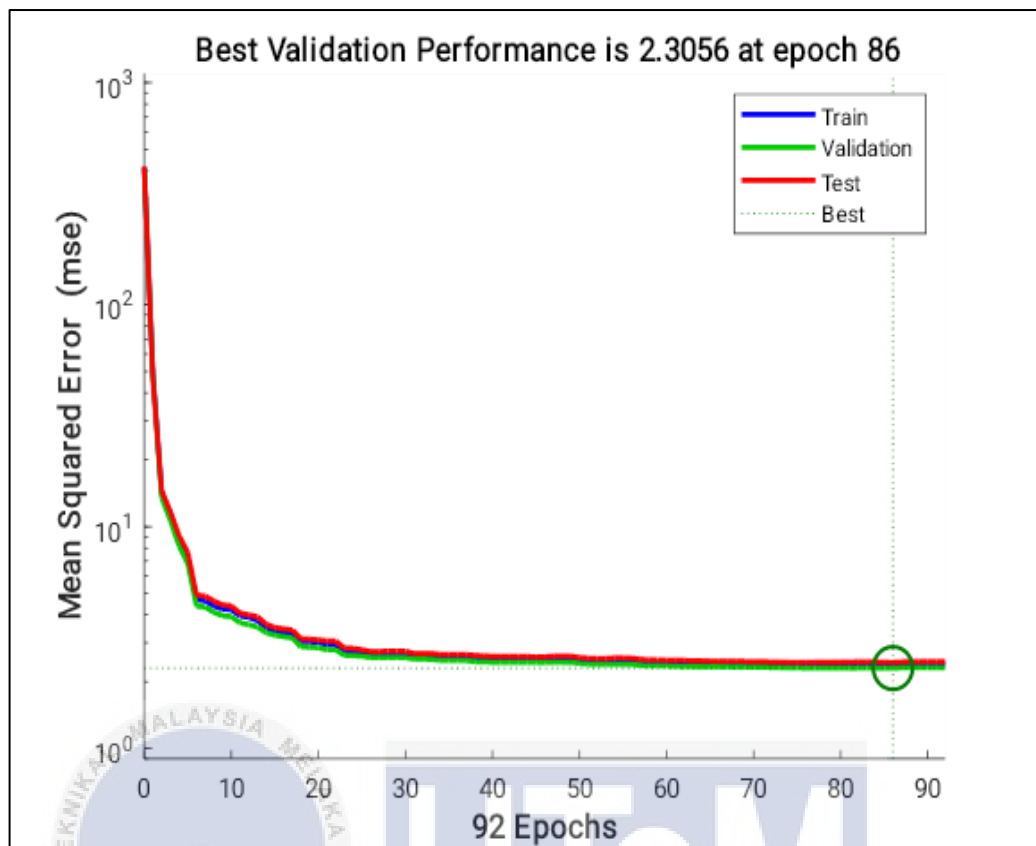


Figure 4.34: Validation performance for ADXL345_1 sensor using SCG algorithm at Motor 2

Figure 4.35 presents the regression plot for the data obtained from ADXL345_1 at Motor 2. This scatter plot visually displays the relationship between the target and output variables, showcasing a robust linear correlation with an R-value of 0.93978. The blue line represents the fit of the SCG algorithm to the data, while the dotted line signifies perfect prediction. In the realm of predictive analytics using the SCG algorithm, this relationship involves utilizing the data from the ADXL345 sensor at Motor 2 as input for predicting future data points or trends. In conclusion, the SCG algorithm proves to be a fitting model for the data, demonstrating excellent predictive power and efficiency.

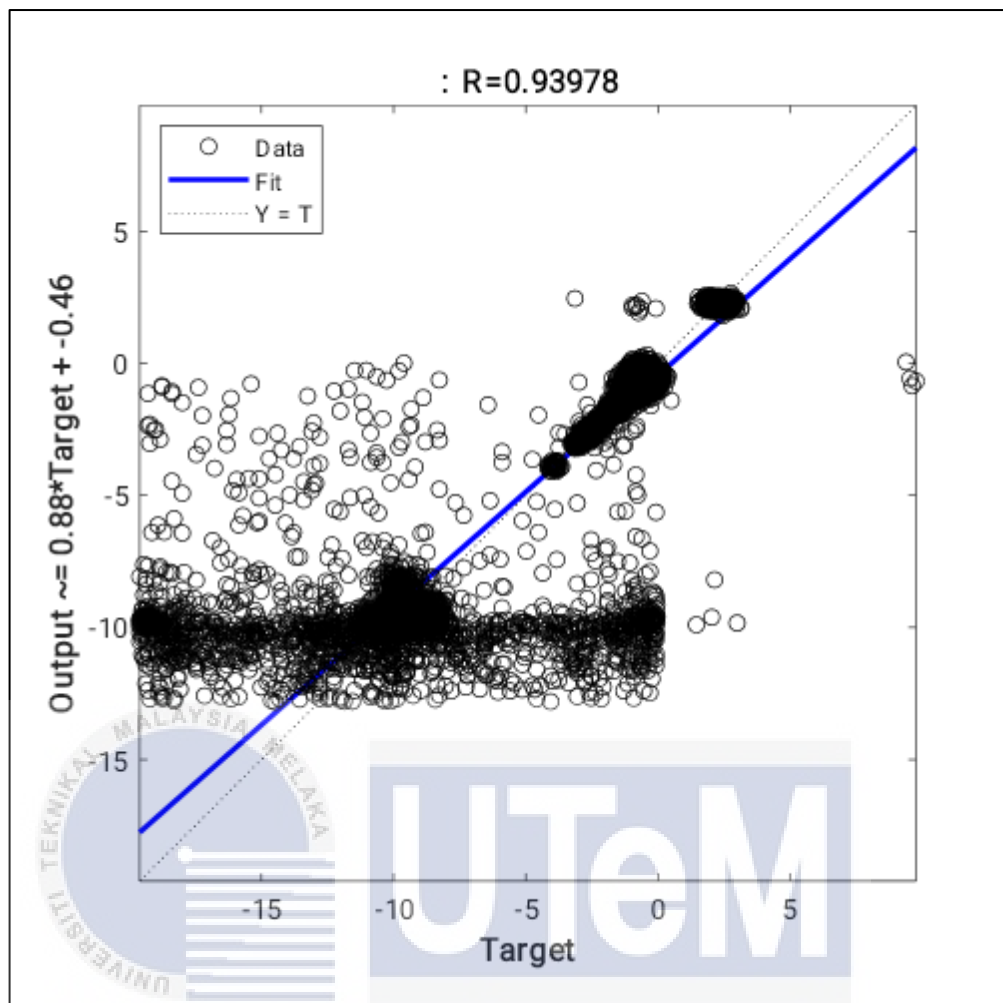


Figure 4.35: Regression graph for ADXL345_1 sensor using SCG algorithm at Motor 2

Table 4.11 provides information on the Mean Square Error (MSE) and regression values during the training, validation, and testing phases. The R values are consistently close to 1 across all three phases, signifying a strong correlation between the predicted and actual outcomes. However, there is a notable increase in MSE during the testing phase, suggesting a potential issue of overfitting to the training data. For a detailed visualization of the regression graphs for each phase, please refer to the Appendix H.

Table 4.11: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_1 sensor using SCG algorithm at Motor 2

	MSE	R
Training	2.84	0.93955
Validation	2.02	0.9413
Testing	2.92	0.9394

4.4.2.6 ADXL345_2 data at Motor 2

The neural network training algorithm for ADXL345_2 is under scrutiny. Upon observing and analyzing the data, the network underwent training for 106 epochs, signifying a complete pass through the entire training dataset. With each epoch representing a repeated learning cycle from the training data, the gradient values exhibited a significant decrease, indicating that the network ceased learning anything new from the training data. Detailed results of the neural network training can be found in Appendix B.

The accompanying Figure 4.36 illustrates the Mean Squared Error (MSE) across 106 epochs, encompassing training, algorithm, validation, and testing data from an ADXL345 sensor. Utilizing the Scaled Conjugate Gradient algorithm for curve-fitting, the objective is to minimize the disparity between observed and predicted temperature readings from the ADXL345 sensor over three days. The graph reflects an initial rapid increase in validation performance during the network's learning phase, succeeded by a subsequent decline until the highlighted epoch. Notably, the optimal validation performance, reaching 0.53908, was achieved at epoch 100, indicating the network's peak performance on unseen data toward the conclusion of this epoch.

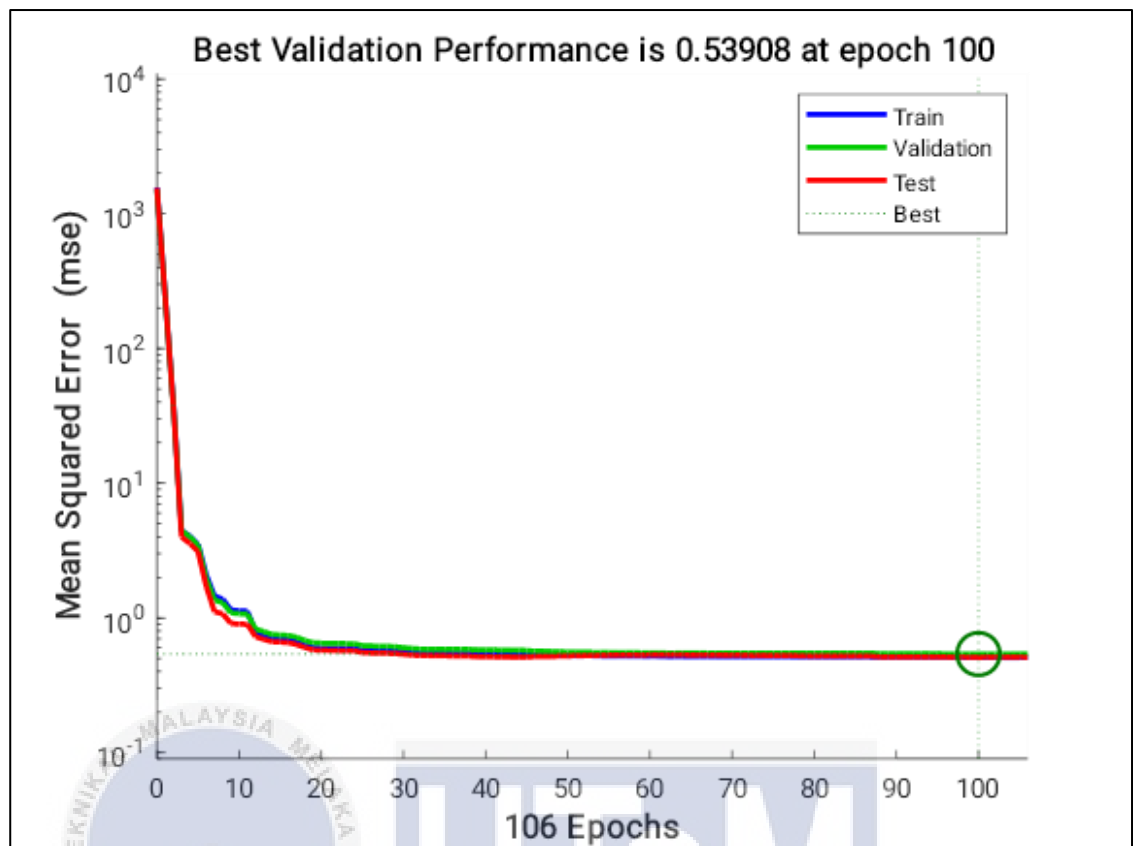


Figure 4.36: Validation performance for ADXL345_2 sensor using SCG algorithm at Motor 2

In the Figure 4.37, a regression graph spanning three days showcases data from the ADXL345 sensor. The x-axis illustrates actual temperature readings retrieved from the ADXL345 sensor, while the y-axis displays temperature values obtained through the scaled conjugate gradient algorithm. Each data point is represented by a circle, and the blue line indicates a linear fit, highlighting a robust positive correlation with an R-value of 0.98793. This suggests that output values increase proportionally with rising target values, demonstrating a strong positive relationship. Within the realm of predictive analytics, the elevated correlation coefficient indicates that the model possesses high accuracy in forecasting future data points based on historical data.

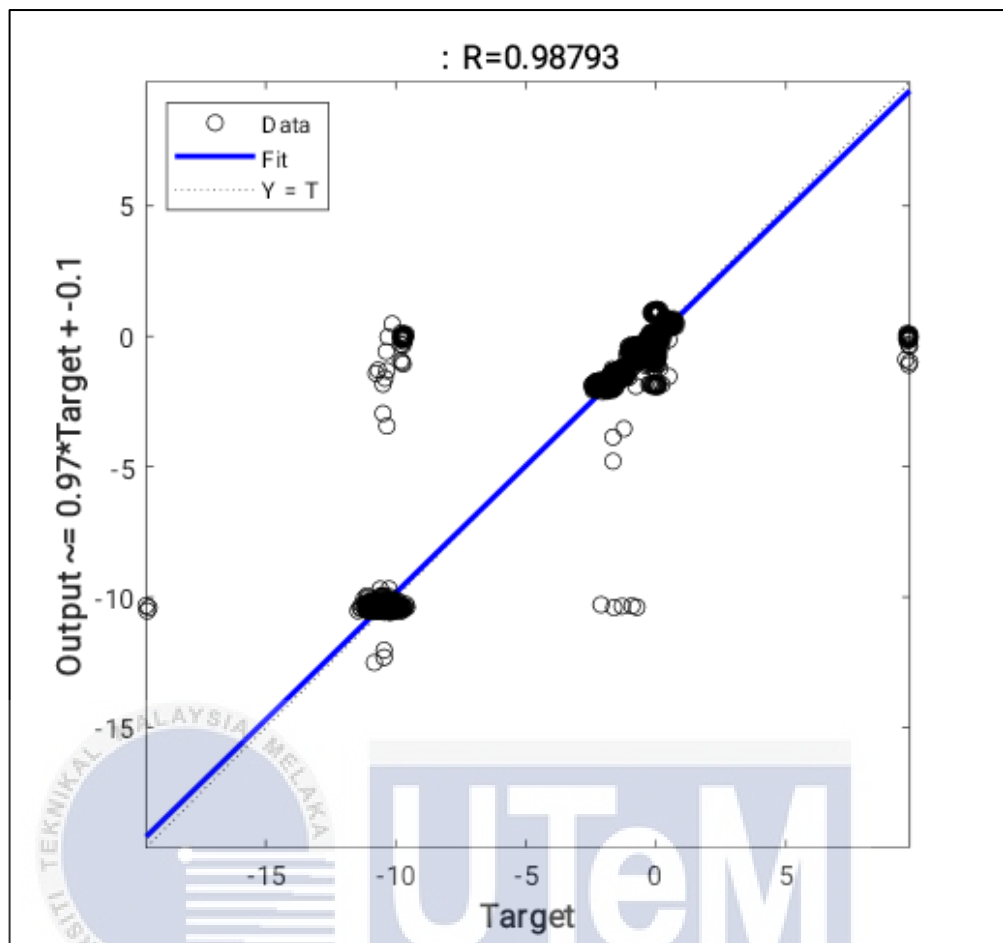


Figure 4.37: Regression graph for DS18B20 sensor using SCG algorithm at Motor 2

Table 4.12 displays the Mean Square Error (MSE) and regression values throughout the training, validation, and testing phases. MSE measures the average squared difference between predicted and actual values, while the R-value assesses the correlation between these values. The consistently low MSE values across all phases indicate a well-fitted model, showcasing superior performance with minimal disparities between predicted and actual outcomes. Similarly, the R-values, nearing 1 for each phase, emphasize a robust correlation between predicted and actual outcomes, affirming the model's excellent fit. The corresponding graph is available in the Appendix H.

Table 4.12: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of ADXL345_2 sensor using SCG algorithm at Motor 2

	MSE	R
Training	0.371	0.98808
Validation	0.72	0.98712
Testing	0.371	0.988

4.4.3 Comparison network training using different training algorithms for each motor.

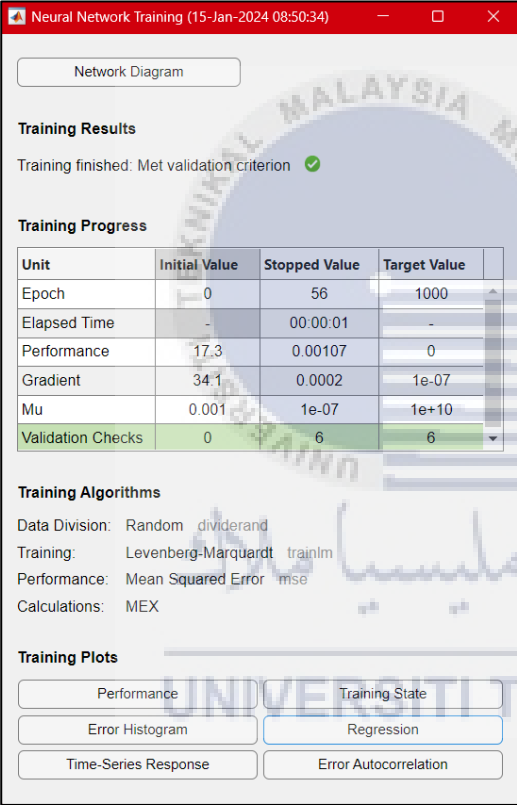

After all sensor's data been analysed with different technique, then the comparison is made between using LM algorithm and SCG algorithm by combining all data from Motor 1 and Motor 2 to train. This action is taken to observe the training performance for both motors when using different algorithm. From the observation, can be seen that Levenberg-Marquardt algorithm offered the best accuracy in terms of neural network training, validation performance and R obtained as shown in Table 4.13. This can be proved with several training results to compare both algorithms. First, the neural network training results using LM algorithm shows that the training stopped at 56 epochs while training for SCG algorithm, stopped at 190 epochs. LM algorithm shows that it is the faster as it requires fewer epochs to converge. This makes LM algorithm is more efficient in the specific instance because it achieved the desired performance in less time and fewer iterations.

Other than that, in the context of the LM algorithm, the best validation performance is 0.00088256 was achieved at epoch 50. This means that the model achieved the best performance on the validation dataset at epoch 50. However, in the context of SCG algorithm, the best validation performance is 0.0015796 was achieved at epoch 184. This means that the model achieved the best performance on the validation dataset at

epoch 184. This indicates LM algorithm has achieved the lowest value of the validation performance during the training process, which the data is trained well. Furthermore, LM algorithm also shown that it has the higher values for R-squared which is 0.9992 near to 1 compared to SCG algorithm which is 0.99895. Both have the best values of R-squared but LM algorithm much higher than SCG algorithm. The higher the values, the better the model fit to the data.



Table 4.13: Table comparison between using LM algorithm and SCG algorithm.

Levenberg-Marquardt Algorithm	Scaled Conjugate Gradient Algorithm																																																				
 <p>Neural Network Training (15-Jan-2024 08:50:34)</p> <p>Network Diagram</p> <p>Training Results Training finished: Met validation criterion ✓</p> <p>Training Progress</p> <table border="1"> <thead> <tr> <th>Unit</th> <th>Initial Value</th> <th>Stopped Value</th> <th>Target Value</th> </tr> </thead> <tbody> <tr> <td>Epoch</td> <td>0</td> <td>56</td> <td>1000</td> </tr> <tr> <td>Elapsed Time</td> <td>-</td> <td>00:00:01</td> <td>-</td> </tr> <tr> <td>Performance</td> <td>17.3</td> <td>0.00107</td> <td>0</td> </tr> <tr> <td>Gradient</td> <td>34.1</td> <td>0.0002</td> <td>1e-07</td> </tr> <tr> <td>Mu</td> <td>0.001</td> <td>1e-07</td> <td>1e+10</td> </tr> <tr> <td>Validation Checks</td> <td>0</td> <td>6</td> <td>6</td> </tr> </tbody> </table> <p>Training Algorithms Data Division: Random dividerand Training: Levenberg-Marquardt trainlm Performance: Mean Squared Error mse Calculations: MEX</p> <p>Training Plots</p> <p>Performance Training State Error Histogram Regression Time-Series Response Error Autocorrelation</p>	Unit	Initial Value	Stopped Value	Target Value	Epoch	0	56	1000	Elapsed Time	-	00:00:01	-	Performance	17.3	0.00107	0	Gradient	34.1	0.0002	1e-07	Mu	0.001	1e-07	1e+10	Validation Checks	0	6	6	 <p>Neural Network Training (15-Jan-2024 09:06:20)</p> <p>Network Diagram</p> <p>Training Results Training finished: Met validation criterion ✓</p> <p>Training Progress</p> <table border="1"> <thead> <tr> <th>Unit</th> <th>Initial Value</th> <th>Stopped Value</th> <th>Target Value</th> </tr> </thead> <tbody> <tr> <td>Epoch</td> <td>0</td> <td>190</td> <td>1000</td> </tr> <tr> <td>Elapsed Time</td> <td>-</td> <td>00:04:11</td> <td>-</td> </tr> <tr> <td>Performance</td> <td>3.76</td> <td>0.00132</td> <td>0</td> </tr> <tr> <td>Gradient</td> <td>17.8</td> <td>0.00292</td> <td>1e-06</td> </tr> <tr> <td>Validation Checks</td> <td>0</td> <td>6</td> <td>6</td> </tr> </tbody> </table> <p>Training Algorithms Data Division: Random dividerand Training: Scaled Conjugate Gradient trainscg Performance: Mean Squared Error mse Calculations: MEX</p> <p>Training Plots</p> <p>Performance Training State Error Histogram Regression Time-Series Response Error Autocorrelation</p>	Unit	Initial Value	Stopped Value	Target Value	Epoch	0	190	1000	Elapsed Time	-	00:04:11	-	Performance	3.76	0.00132	0	Gradient	17.8	0.00292	1e-06	Validation Checks	0	6	6
Unit	Initial Value	Stopped Value	Target Value																																																		
Epoch	0	56	1000																																																		
Elapsed Time	-	00:00:01	-																																																		
Performance	17.3	0.00107	0																																																		
Gradient	34.1	0.0002	1e-07																																																		
Mu	0.001	1e-07	1e+10																																																		
Validation Checks	0	6	6																																																		
Unit	Initial Value	Stopped Value	Target Value																																																		
Epoch	0	190	1000																																																		
Elapsed Time	-	00:04:11	-																																																		
Performance	3.76	0.00132	0																																																		
Gradient	17.8	0.00292	1e-06																																																		
Validation Checks	0	6	6																																																		

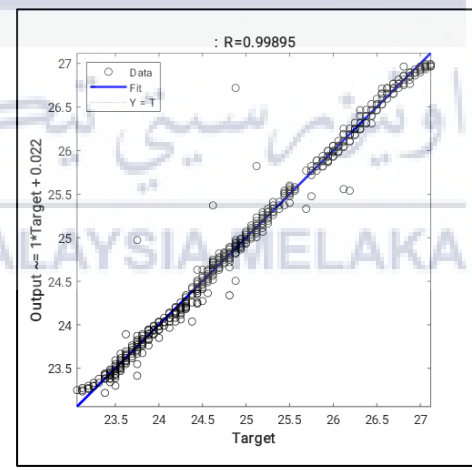
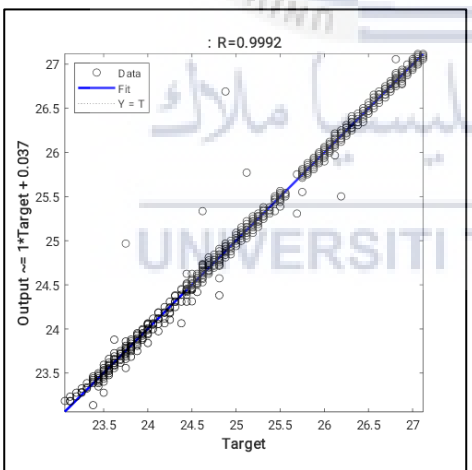
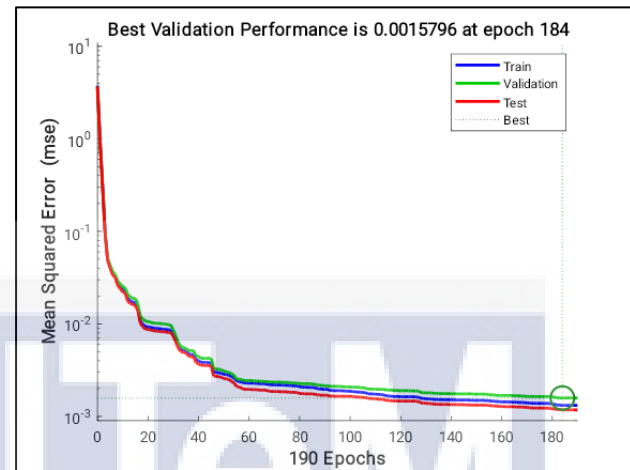
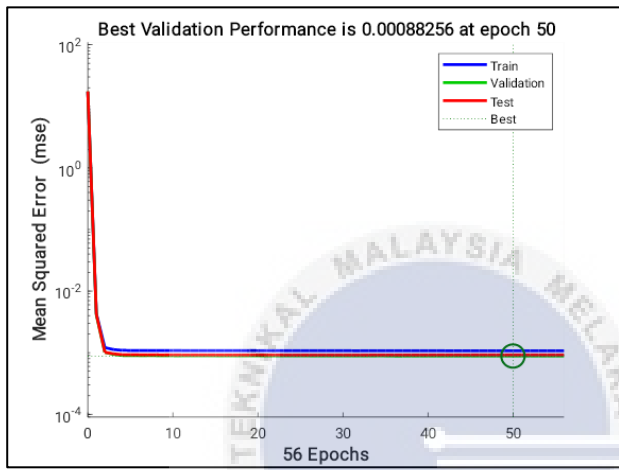


Table 4.14 represents the Mean Square Error (MSE) and regression values for training, validation, and testing phases of two algorithms: Levenberg-Marquardt and Scaled Conjugate Gradient algorithms. In terms of predictive analytics, LM algorithm has the lower MSE values across all three phases compared to SCG algorithm, indicating a better performance in terms of error rate. The R values for LM algorithm also are slightly higher than those for SCG algorithm, suggesting that LM also performs better in terms of regression. In conclusion, LM algorithm is better than the SCG algorithm for predictive analytics and performance.

Table 4.14: The Mean Square Error (MSE) and Regression (R) values for the Training, Validation and Testing of sensors at Motor 1 and Motor 2

	Levenberg-Marquardt Algorithm		Scaled Conjugate Gradient Algorithm	
	MSE	R	MSE	R
Training	0.0012	0.99915	0.0022	0.99898
Validation	0.00256	0.99932	0.00136	0.99871
Testing	0.000675	0.99929	0.00104	0.99907

4.5 Relationship with Sustainable Development Goal (SDG).

The machine condition monitoring project aligns with Sustainable Development Goal (SDG) No. 9, which emphasizes the need to build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation. By focusing on predictive maintenance techniques, the project contributes to the resilience of industrial infrastructure by proactively addressing potential faults and minimizing disruptions. It also promotes sustainable industrialization by ensuring optimal performance and longevity of machinery, thereby enhancing efficiency in industrial processes. The incorporation of innovative technologies, such as Raspberry Pi,

sensors, and Node-RED, showcases a commitment to fostering innovation in the field of industrial maintenance. Overall, the project exemplifies a comprehensive approach to advancing SDG No. 9 through its emphasis on resilient infrastructure, sustainable industrial practices, and technological innovation.

The machine condition monitoring project using predictive maintenance contributes to Sustainable Development Goal (SDG) No. 12, which focuses on ensuring sustainable consumption and production patterns. By implementing predictive maintenance techniques, the project aims to prolong the lifespan of industrial machinery and optimize their performance. This approach aligns with SDG No. 12 by promoting sustainable practices in industrial production. Through the proactive identification of potential faults, the project minimizes unnecessary resource consumption associated with reactive maintenance and unplanned downtime. By fostering efficient and sustainable consumption of resources, the project contributes to the broader objective of achieving sustainable production patterns outlined in SDG No. 12.

CHAPTER 5

CONCLUSION AND FUTURE WORKS



In conclusion, there are various methods to detect motor abnormalities such as vibration analysis, energy monitoring, and sensor data-based predictive maintenance using ADXL345 and DS18B20 sensors. . By integrating advanced sensors, including ADXL345 for vibration detection and DS18B20 for temperature monitoring, the system has provided accurate and timely insights into the motor's condition, enabling proactive interventions, and minimizing the risk of catastrophic failures. These methods can help detect potential issues or faults in the motor, which can help minimize downtime and costly repairs.

In this study, Node-RED has been used to develop a real-time data processing and visualization. Node-RED is a powerful open-source tool for visual programming that can be used to build Internet of Things (IoT) applications. This innovative solution

facilitates real-time data acquisition, processing, and visualization, empowering maintenance teams with a user-friendly interface to monitor motor conditions and make informed decisions effectively. The monitoring system collect data from sensors and use that data to make a prediction about future performance.

Additionally, predictive maintenance is developed using historical data from collected sensor's data. By leveraging historical data, sensor data, and advanced analytics, predictive maintenance aims to identify potential failures or issues before they occur, allowing for proactive maintenance actions and minimizing downtime and costly repairs. The data analytics and machine learning techniques had been used to analyse patterns, identifies potential failure modes, and optimizes maintenance schedule.

Hence, by combining the methods mentioned above, it is possible to develop a system that can detect motor abnormalities, develop a monitoring system using the Node-RED visual programming tool with a Raspberry Pi as a microcontroller, and develop predictive maintenance using historical data from collected sensor data. This can help ensure the efficient operation of the motor and minimize downtime and costly repairs.

5.1 Future works

The motor condition monitoring project, utilizing predictive maintenance through components like Raspberry Pi, ADXL345, DS18B20, Node-RED, and MATLAB, presents promising avenues for future enhancements. One of the future works which can be made is include integrating advanced sensors for comprehensive motor data capture and exploring wireless sensor deployment. For example, current sensors, or acoustic sensors to capture more comprehensive data about the motor's condition.

Next is, cloud platform integration which offers centralized data management and remote monitoring capabilities, enhancing accessibility and responsiveness where maintenance teams can monitor motor conditions from anywhere and receive alerts on their devices.

After that another future works can be implemented is in term of user interface and visualization where it focuses on intuitive dashboards and trend visualization tools, empowering maintenance teams with user-friendly manner. Continuous optimization of predictive algorithms, which a through safety assessment and implement an additional safety feature, such as emergency shutdown mechanisms or redundant monitoring systems. A compliance with safety regulations, scalability, and integration with existing systems also can ensure a holistic approach to motor maintenance. Moreover, emphasizing training and skill development for maintenance teams on utilizing the enhances system capabilities effectively can keep abreast of emerging technologies and best practices in for optimizing motor performance, reliability, and safety.

REFERENCES

- [1] Z. Liu, T. Yuan, X. Zhou, X. Yuan, Y. Wang and X. Zhang, "Research on Predictive Maintenance Technology of Stepping Motor Based on Load Value Analysis," *In 2020 Chinese Automation Congress (CAC)*, pp. 7122-7125, 2020.
- [2] B. S. Kumar, A. Aravindraj, T. A. S. S. Priya and S. Nihanth, "A Machine Learning Model for Predictive Maintenance of a Stepper Motor Using Digital Simulation Data," *In Proceedings of Third International Conference on Sustainable Expert Systems: ICSES 2022*, 2023.
- [3] N. Dehbashi, S. M. Hosseimi and A. Yazdian-Varjani, "IoT Based Condition Monitoring and Control of Induction Motor Using Raspberry Pi," *In 2022 13th Power Electronics, Drive Systems, and Technologies Conference (PEDSTC)*, pp. 134-138, 2022.
- [4] K. Saritaa, R. Devarapallia and S. Kumarc, "Principal component analysis technique," *Journal of Intelligence & Fuzzy Systems*, no. 1064-1246, 2021.

- [5] M. J. Corres, J. Bravo, F. J. Arregui and R. I. Matias, "Vibration monitoring in electrical engines using an in-line fiber etalon," *Sensors and Actuators A: Physical*, vol. 132, no. 2, pp. 506-515, 2006.
- [6] A. A. J. a. r. Bicker, "The State of the Art in Research into the Condition Monitoring of Industrial Machinery," *International Journal of Current Engineering and Technology*, vol. 4, p. 3, 2014.
- [7] V. P. Subhasri, K. Grace, Subhasri and V. P., "Cloud Enabled Predictive Maintenance Tool for," *In 2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, pp. 832-835, 2022.
- [8] E. Noyjeen, C. Tanita, N. Panthasarn, P. Chansri and J. Pukkham, "Monitoring Parameters of Three-Phase Induction," *9th International Electrical Engineering Congress (iEECON)*, pp. 483-486, 2021.
- [9] B. R. Kumar, M. Gowrisankar, S. Ramana, P. D. Aakash and S. Aravind, "Load Fault Diagnosis in Induction Motor using Artificial Intelligence Algorithm," *IEEE Explore*, no. 22068098, 2022.
- [10] G. B. Lucas, B. A. d. Castro, P. J. A. Serni, R. R. Riehl and A. L. Andreoli, "Sensors Applied to Bearing Fault Detection in Three-Phase Induction Motors," *MCPI*, vol. 1, no. 10, p. 40, 2021.
- [11] G. A. Gonzalez, B. Santiago, B. Godinez, P. B. Martinez, O. D. Roman, A. L. Galicia and J. M. Garay, "Remote control and monitoring of a hydraulic machine," *Journal of Physics: Conference Series*, no. 012009, 2022.
- [12] D. Topolsky, I. Topolskaya, V. Mitin, N. Topolsky, N. Yumagulov and T. Gonenko, "Non-Contact Condition Monitoring of Electrical Drive," *In 2020*

- Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBREIT)*, pp. 436-439, 2020.
- [13] N. Dehbashi, M. S. Hosseini and A. Yazdian-Varjani, "IoT Based Condition Monitoring and Control of Induction Motor Using Raspberry Pi," *In 2022 13th Power Electronics, Drive Systems, and Technologies Conference (PEDSTC)*, pp. 134-138, 2022.
- [14] H.-C. Chang, Y.-M. Jheng, S.-W. Sun, J.-W. Shen, C.-C. Kuo, Y.-C. Wang and T.-Y. Tsai, "Proactive Operation Condition Monitoring System of High-Voltage Motors Based on CNN and LSTM," *In 2021 9th International Conference on Orange Technology (ICOT)*, pp. 1-4, 2021.
- [15] A. Gugaliya, G. Singh, V. Shah, M. Tamboli, A. Bandekar and P. Baviskar, "Availability Improvement of Induction Motors through Condition Monitoring," *In 2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, pp. 1-5, 2022.
- [16] K. V. Surti and A. C. Naik, "Bearing Condition Monitoring of Induction Motor Based on Discrete Wavelet Transform & K-nearest Neighbor," *In 2018 3rd International Conference for Convergence in Technology (I2CT)*, pp. 1-5, 2018.
- [17] R. Raman and K. Naikade, "Smart Industrial Motor Monitoring with IoT-Enabled Photovoltaic System," *In 2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pp. 53-57, 2023.
- [18] O. Prakash Choudhary, S. Jaiswal and S. Das, "IoT Enabled Condition Monitoring of Low Voltage Motors using Fuzzy Inference System," *In 2021*

- IEEE 5th International Conference on Condition Assessment Techniques in Electrical Systems (CATCON)*, no. DOI: 10.1109/CATCON52335.2021.9670486, pp. 132-137, 2021.
- [19] M. Hayouni, Z. Bousselmi, T.-H. Vuong, F. Choubani and J. David, "Wireless IoT Approach for Testing in situ Motor's Axis Vibration Monitoring," *In 2021 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 92-97, 2021.
- [20] K. Liliana, M. Markiewicz, M. Wielgosz, M. Bochenski, W. Tabaczynski, T. Konieczny and L. Kowalczyk, "Predictive maintenance of induction motors using ultra-low power wireless sensors and compressed recurrent neural networks," *EEE Access*, vol. 7, pp. 178891-178902, 2019.
- [21] L. Magadan, F. J. Suarez, J. C. Granda and D. F. Garcia, "Low-Cost Industrial IoT System for Wireless Monitoring of Electric," *Mobile Networks and Applications*, pp. 1-10, 2022.
- [22] D. Kumar, S. Mehran, S. Shaikh, M. Z., M. Hussain, B. S. Chowdhry and T. Hussain, "Triaxial bearing vibration dataset of induction motor under varying load conditions," *Data in Brief* 42, p. 108315, 2022.
- [23] I. Szabo, A. A. Tulbure and D. Fleseriu, "Vibration and temperature sensor network solutions:," *IEEE 6th International Conference on Computing, Communication and Automation (ICCCA)*, pp. 678-682, 2021.
- [24] V. Khandelwal, P. Ramtekkar, M. Chauhan, Y. Bhute and R. Kouthekar, "Sensor Based Vibration Analysis of Motor Using," *10th International Conference on Emerging Trends in Engineering and Technology-Signal and Information Processing (ICETET-SIP-22)*, pp. 1-4, 2022.

- [25] A. Holovatyy, V. Teslyuk, M. Iwaniec and M. Mashevskaya, "Development of A System For Monitoring Vibration Accelerations Based On the Raspberry Pi Microcomputer and the ADXL345 Accelerometer," *Восточно-Европейский журнал передовых технологий*, vol. 6, no. 9, pp. 52-62, 2017.
- [26] M. Iwaniec, A. Holovatyy, V. Teslyuk, K. Kolesnyk and M. Mashevskaya, "Development of Vibration Spectrum Analyzer Using the Raspberry Pi Microcomputer and 3-Axis Digital MEMS Accelerometer ADXL345," *XIIIth International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, pp. 25-29, 2017.
- [27] V. Hedge and G. S. Maruthi, "Development of Fault Diagnosis Unit for Induction Motor using Digital Signal Processor and MEMS Accelerometer," *IEEE 4th International Conference on Condition Assessment Techniques in Electrical Systems (CATCON)*, pp. 1-5, 2019.
- [28] A. P. Ompusunggu, K. Eryilmaz and K. Janssen, "Condition monitoring of critical industrial assets using high performing low-cost MEMS accelerometers," *Procedia CIRP*, vol. 104, pp. 1389-1394, 2021.
- [29] C. D. R. Sahu, A. I. Mukadam, S. D. Das and S. Das, "Integration of Machine Learning and IoT System for Monitoring Different Parameters and Optimizing farming," *In 2021 International Conference on Intelligent Technologies*, pp. 1-5, 2021.
- [30] S. Banerjee, A. K. Saini, H. Nigam and VijayV., "IoT Instrumented Food and Grain Warehouse Traceability System for Farmers," *international*

conference on artificial intelligence and signal processing (AISP), pp. 1-4, 2020.

- [31] N. Bhavana, M. R. Ranga, A. G. Krishna, N. Manisha and D. Rithin, "High Security Alert System for Industrial Atmospheric Parameters," *7th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1157-1161, 2023.
- [32] Y. Ardiyanto, K. Adi, K. T. Putra and P. Utomo, "Monitoring System for Elderly Health Care Using Smart Band, Raspberry Pi, and Node-Red," *2nd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, pp. 252-257, 2022.
- [33] P. S. Macheso, T. D. Manda, A. G. Meela, J. S. Mlatho, G. T. Taulo and B. M. Mame, "Environmental Parameter Monitoring System Based on NodeMCU ESP8266, MQTT and Node-RED," *International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1-4, 2022.
- [34] Cabuk and A. Sinan, "Experimental IoT study on fault detection and preventive apparatus using Node-RED ship's main engine cooling water pump motor," *Engineering Failure Analysis*, p. 106310, 2022.
- [35] Bicker, A. A. Jaber and Robert, "The State of the Art in Research into the Condition Monitoring of Industrial," *International Journal of Current Engineering and Technology*, vol. 4, no. 3, 2014.
- [36] M. Hayouni, Z. Bousselmi, T. H. Vuong, F. Choubani and J. David, "Wireless IoT Approach for Testing in situ Motor's Axis Vibration Monitoring," *International Wireless Communications and Mobile Computing (IWCMC)*, pp. 92-97, 2021.

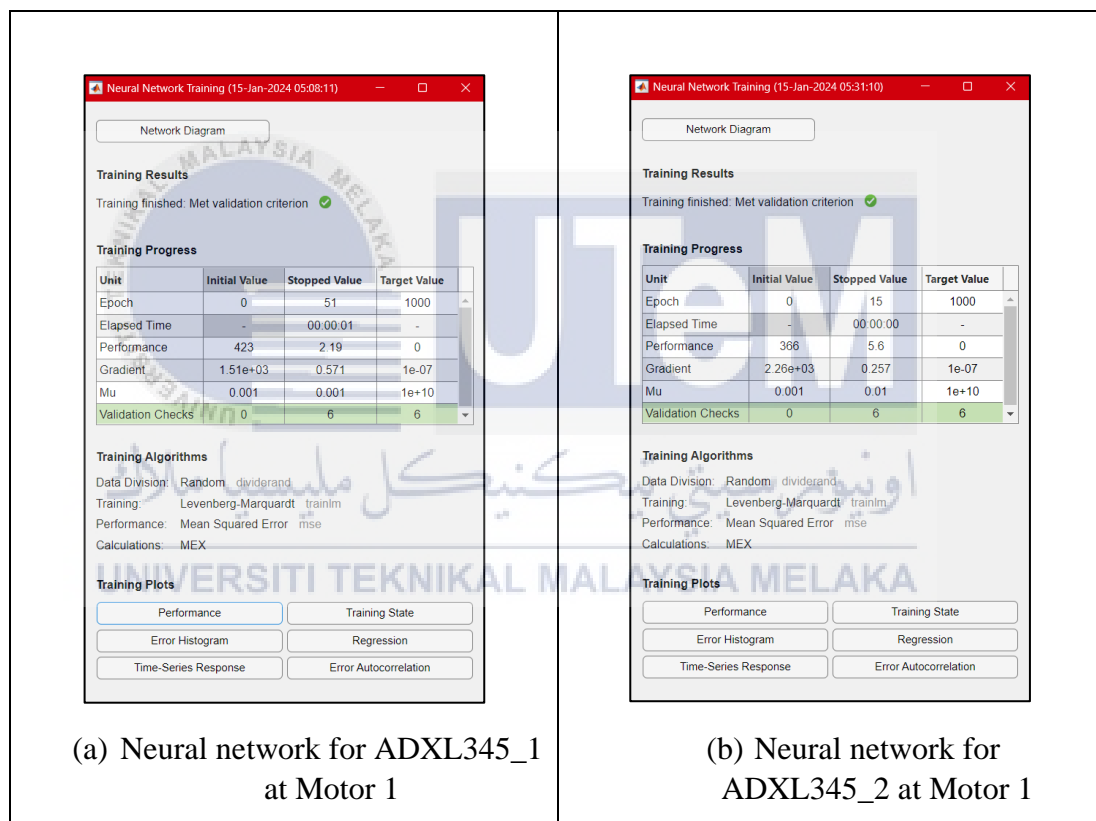
- [37] M. Lekic and G. Gardasevic, "IoT sensor integration to Node-RED platform.," *17th International Symposium Infoteh-Jahorina (Infoteh)*, pp. 1-5, 2018.
- [38] K. Ferencz and J. Domokos, "Using Node-RED platform in an industrial environment," *Jubileumi Kandó Konferencia*, pp. 52-63, 2019.
- [39] "Node-RED," OpenJS Foundation, [Online]. Available: <https://nodered.org/>. [Accessed 28 June 2023].
- [40] "Node-RED Library," OpenJS Foundation, [Online]. Available: <https://flows.nodered.org/>. [Accessed 28 June 2023].
- [41] "Node-RED Dashboard," OpenJS Foundation, [Online]. Available: <https://flows.nodered.org/node/node-red-dashboard>. [Accessed 28 June 2023].
- [42] "Raspberry Pi," Official Web Site, [Online]. Available: [https://www.raspberrypi.org/..](https://www.raspberrypi.org/)
- [43] "Introduction To Node-RED: Examples And Documentation," Tech Explorations, [Online]. Available: <https://techexplorations.com/guides/esp32/node-red-esp32-project/1-introduction-to-node-red/>. [Accessed 29 June 2023].
- [44] "node-red-contrib-rate-avg," Node-RED, [Online]. Available: <https://flows.nodered.org/node/node-red-contrib-rate-avg>.
- [45] "Creality Ender 3 Review: Specs, Upgrades, Software and More," 24 August 2020.
- [46] E. Boos, D. S. Goncalves and F. S. V. Bazan, "Levenberg-Marquardt method with Singular Scaling and applications," vol. 2305.07755.

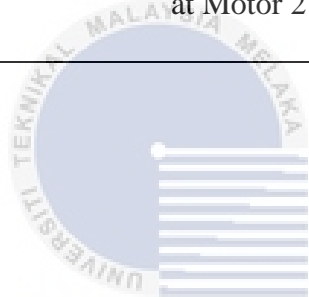
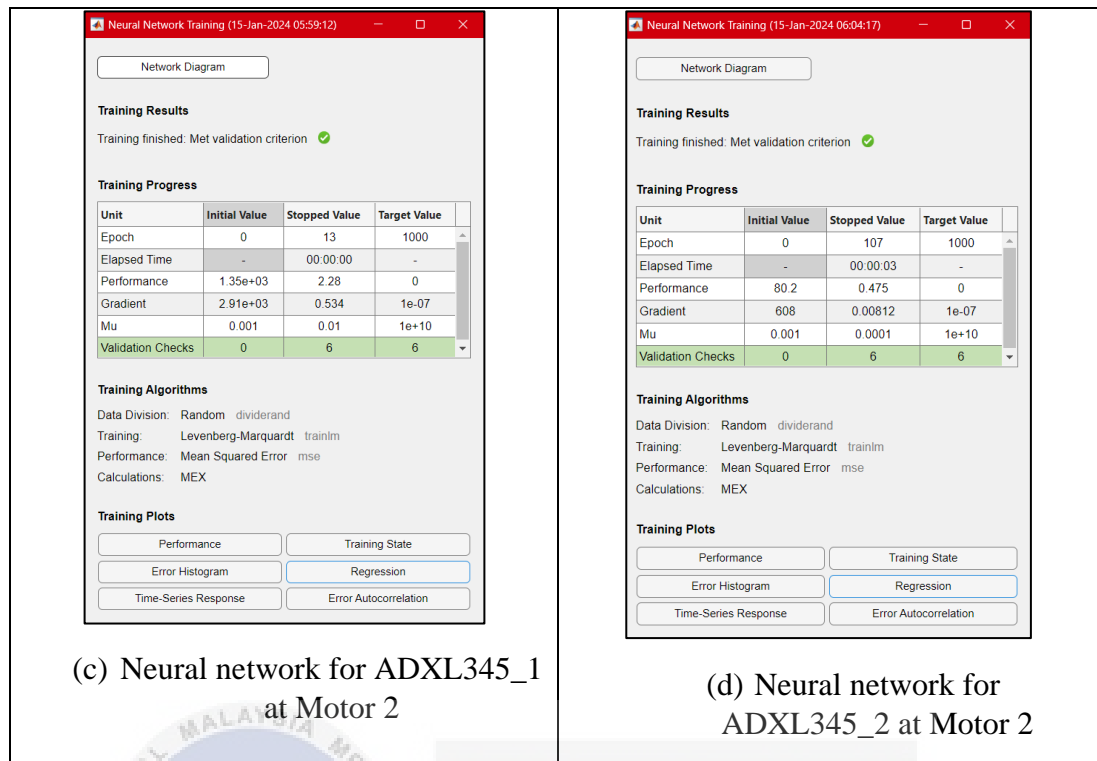
- [47] E. Boos, D. S. Goncalves and F. S. V. Bazan, "Levenberg-Marquardt method with Singular Scaling and applications," vol. 2305.07755.



APPENDICES

APPENDIX A: Neural Network Training for ADXL345 sensors using Levenberg-Marquardt algorithm.





اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

APPENDIX B: Neural Network Training for ADXL345 sensors using Scaled Conjugate Gradient algorithm.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	56	1000
Elapsed Time	-	00:00:01	-
Performance	2.34e+03	2.46	0
Gradient	4.23e+03	0.649	1e-06
Validation Checks	0	6	6

Training Algorithms
 Data Division: Random dividerand
 Training: Scaled Conjugate Gradient trainscg
 Performance: Mean Squared Error mse
 Calculations: MEX

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	102	1000
Elapsed Time	-	00:00:02	-
Performance	350	5.9	0
Gradient	1.86e+03	2.58	1e-06
Validation Checks	0	6	6

Training Algorithms
 Data Division: Random dividerand
 Training: Scaled Conjugate Gradient trainscg
 Performance: Mean Squared Error mse
 Calculations: MEX

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	92	1000
Elapsed Time	-	00:00:02	-
Performance	414	2.4	0
Gradient	1.6e+03	0.174	1e-06
Validation Checks	0	6	6

Training Algorithms
 Data Division: Random dividerand
 Training: Scaled Conjugate Gradient trainscg
 Performance: Mean Squared Error mse
 Calculations: MEX

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	106	1000
Elapsed Time	-	00:03:05	-
Performance	1.56e+03	0.507	0
Gradient	2.79e+03	0.628	1e-06
Validation Checks	0	6	6

Training Algorithms
 Data Division: Random dividerand
 Training: Scaled Conjugate Gradient trainscg
 Performance: Mean Squared Error mse
 Calculations: MEX

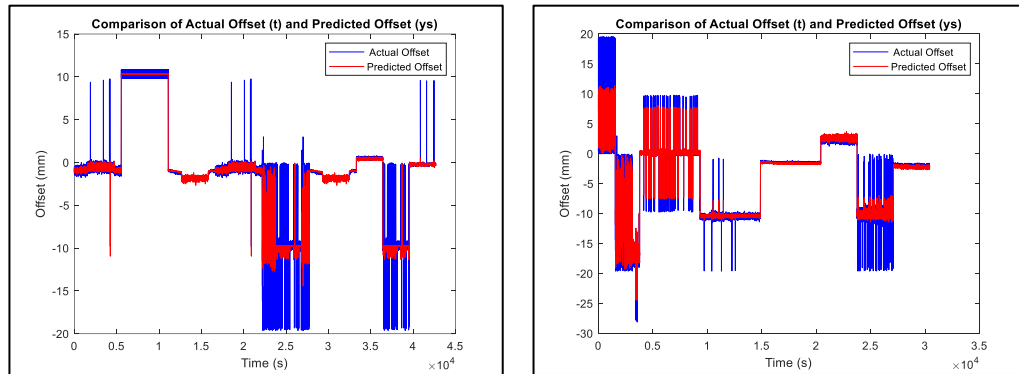
(a) Neural network for ADXL345_1 at Motor 1

(b) Neural network for ADXL345_2 at Motor 1

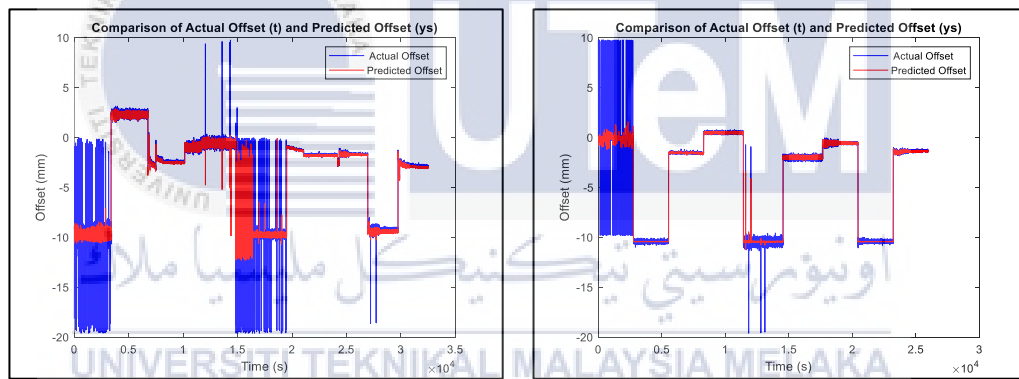
(c) Neural network for ADXL345_1 at Motor 2

(d) Neural network for ADXL345_2 at Motor 2

**APPENDIX C: Comparison of Actual and Predicted Offset of
ADXL345_1 and ADXL345_2 using Levenberg-Marquardt algorithm.**

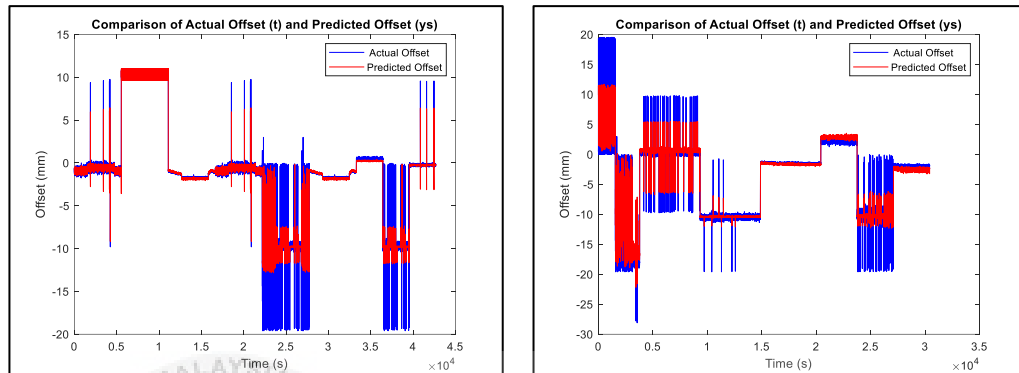


(a) Graph for ADXL345_1 at Motor 1 and Motor 2

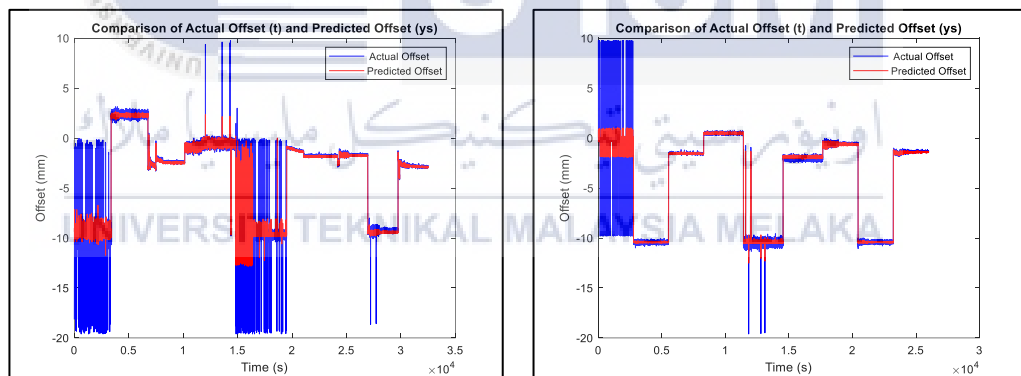


(b) Graph for ADXL345_2 at Motor 1 and Motor 2

**APPENDIX D: Comparison of Actual and Predicted Offset of
ADXL345_1 and ADXL345_2 using Scaled Conjugate Gradient
algorithm.**



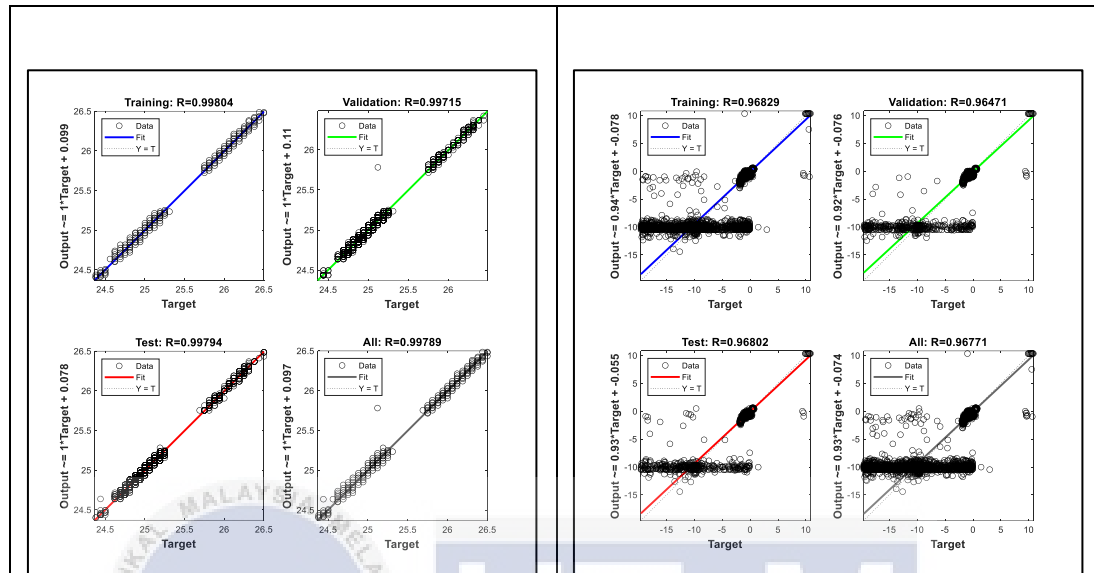
(a) Graph for ADXL345_1 at Motor 1 and Motor 2



(b) Graph for ADXL345_2 at Motor 1 and Motor 2

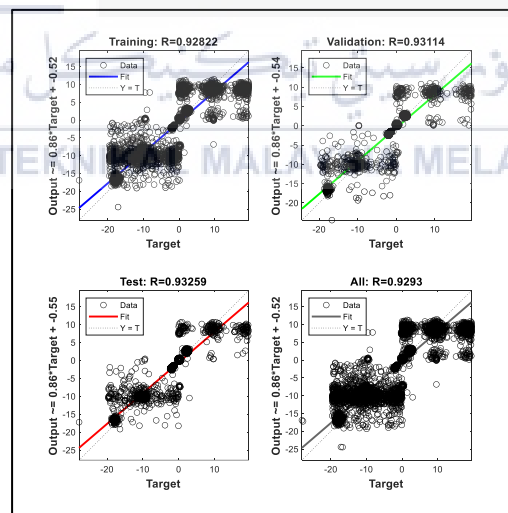
**APPENDIX E: Regression graph for Training, Validation, and Testing
for DS18B20 sensor and ADXL345 sensors using Levenberg-Marquardt**

algorithm at Motor 1



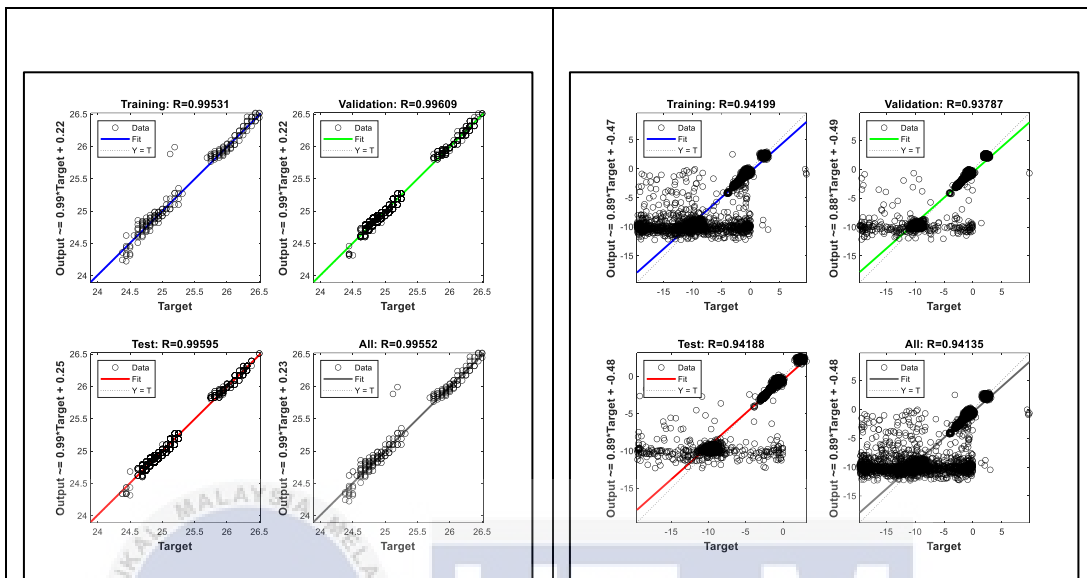
(a) Regression graph for DS18B20 sensor

(b) Regression graph for ADXL345_1 sensor



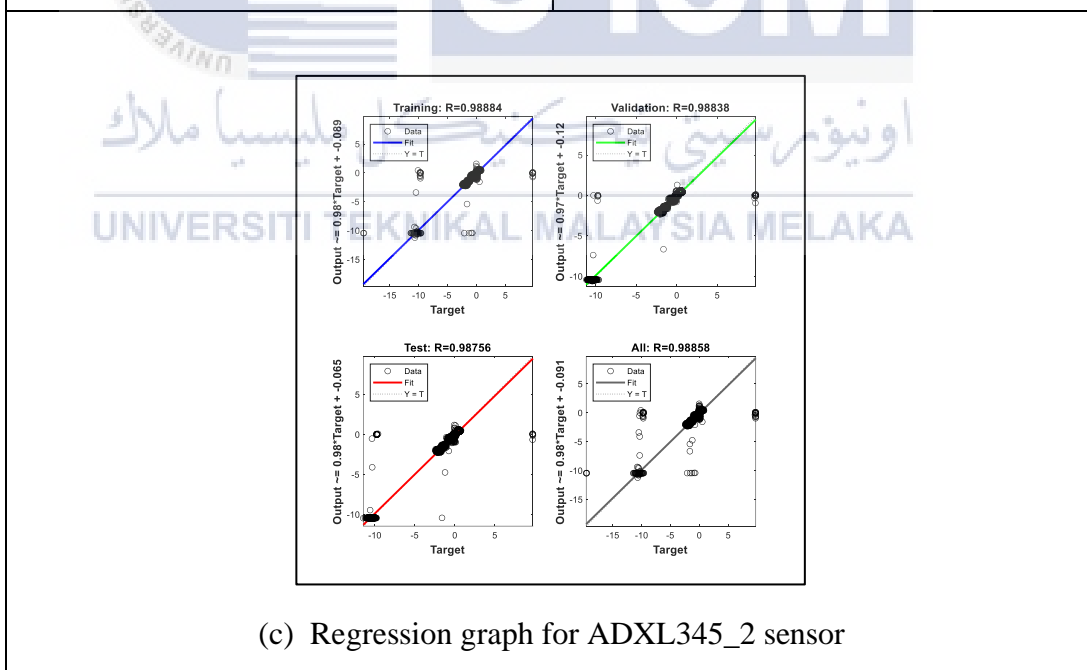
(c) Regression graph for ADXL345_2 sensor

APPENDIX F: Regression graph for Training, Validation, and Testing for DS18B20 sensor and ADXL345 sensors using Levenberg-Marquardt algorithm at Motor 2



(a) Regression graph for DS18B20 sensor

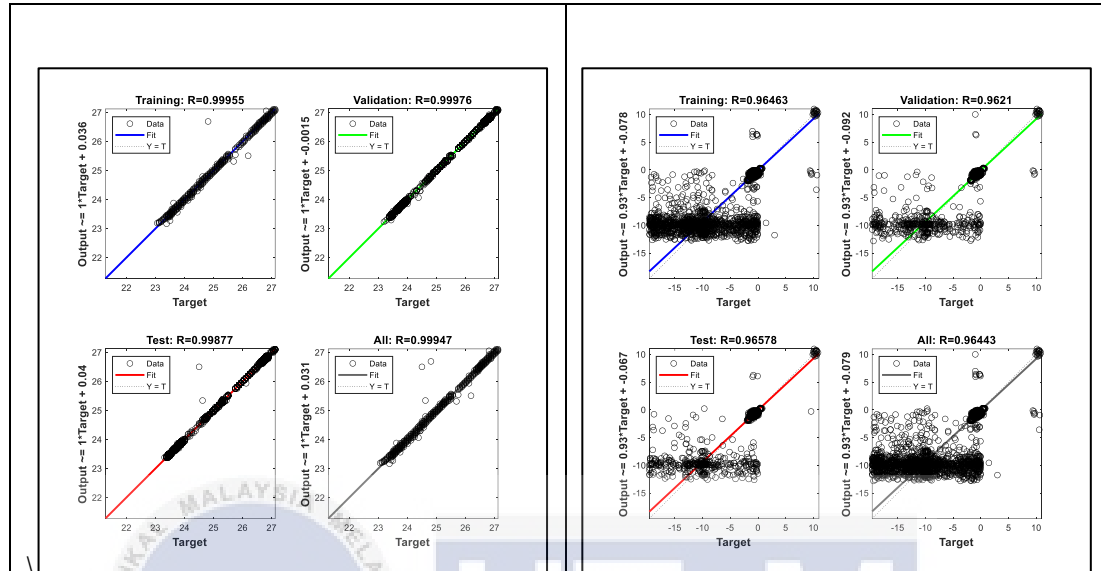
(b) Regression graph for ADXL345_1 sensor



(c) Regression graph for ADXL345_2 sensor

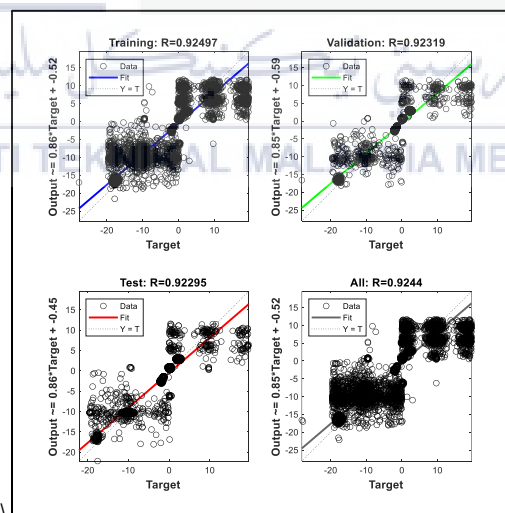
**APPENDIX G: Regression graph for Training, Validation, and Testing
for DS18B20 sensor and ADXL345 sensors using Scaled Conjugate
Gradient algorithm at Motor 1**

Gradient algorithm at Motor 1



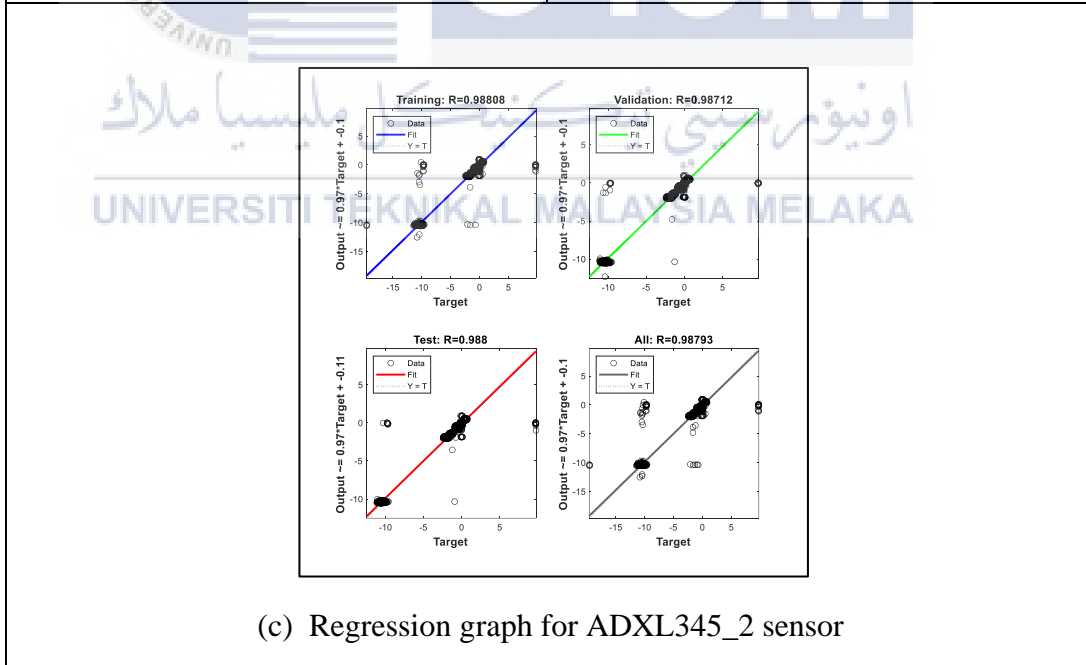
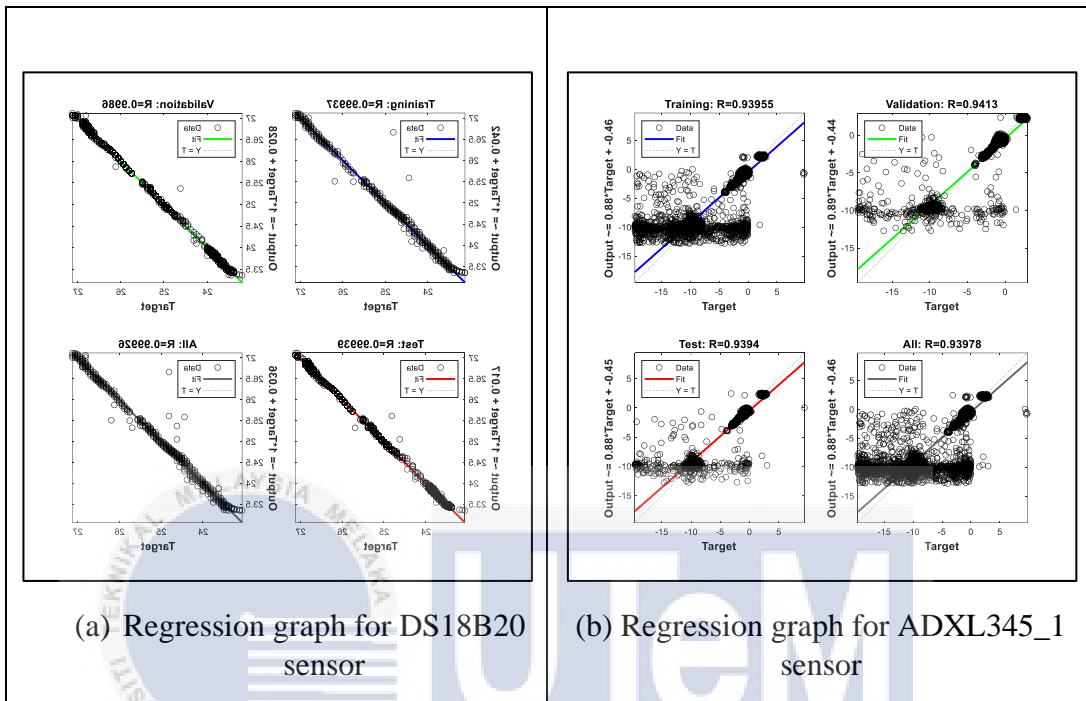
(a) Regression graph for DS18B20 sensor

(b) Regression graph for ADXL345_1 sensor

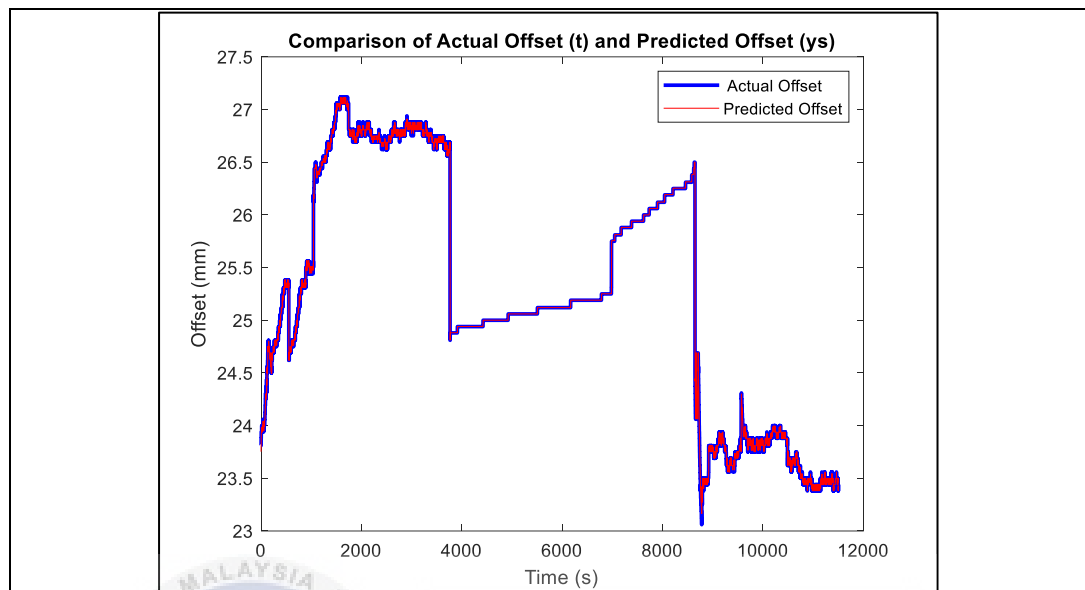


(c) Regression graph for ADXL345_2 sensor

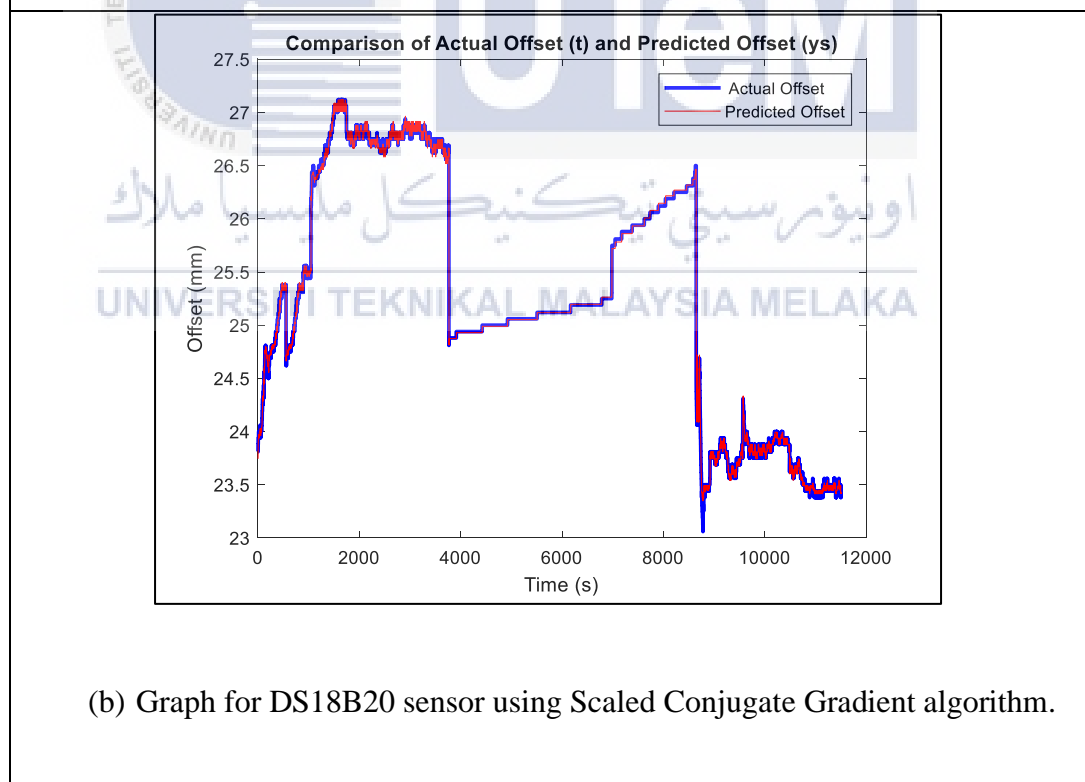
**APPENDIX H: Regression graph for Training, Validation, and Testing
for DS18B20 sensor and ADXL345 sensors using Scaled Conjugate
Gradient algorithm at Motor 2**



**APPENDIX I: Graph for Comparison of Actual and Predicted Offset
for DS18B20 sensor at Motor 2**



(a) Graph for DS18B20 sensor using Levenberg-Marquardt algorithm.



(b) Graph for DS18B20 sensor using Scaled Conjugate Gradient algorithm.

APPENDIX J: MATLAB code for Levenberg-Marquardt and Scaled Conjugate Gradient algorithm.

```

% Solve an Autoregression Time-Series Problem with a NAR Neural
Network
% Script generated by Neural Time Series app
% Created 08-Jan-2024 04:58:53
%
% This script assumes this variable is defined:
%
% data - feedback time series.

T = tonndata(data,false,false);

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm'; % Levenberg-Marquardt backpropagation.

% Create a Nonlinear Autoregressive Network
feedbackDelays = 1:2;
hiddenLayerSize = 10;
net = narnet(feedbackDelays,hiddenLayerSize,'open',trainFcn);

% Prepare the Data for Training and Simulation
% The function PREPARETS prepares timeseries data for a particular
network,
% shifting time by the minimum amount to fill input states and layer
% states. Using PREPARETS allows you to keep your original time series
data
% unchanged, while easily customizing it for networks with differing
% numbers of delays, with open loop or closed loop feedback modes.
[x,xi,ai,t] = preparets(net,{}, {},T);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t,xi,ai);

% Test the Network
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

```

```

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotresponse(t,y)
%figure, ploterrcorr(e)
%figure, plotinerrcorr(x,e)

% Closed Loop Network
% Use this network to do multi-step prediction.
% The function CLOSELOOP replaces the feedback input with a direct
% connection from the output layer.
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
view(netc)
[xc,xic,aic,tc] = preparets(netc,{}, {},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc)

% Step-Ahead Prediction Network
% For some applications it helps to get the prediction a timestep
early.
% The original network returns predicted y(t+1) at the same time it is
% given y(t+1). For some applications such as decision making, it would
% help to have predicted y(t+1) once y(t) is available, but before the
% actual y(t+1) occurs. The network can be made to return its output a
% timestep early by removing one delay so that its minimal tap delay is
now
% 0 instead of 1. The new network returns the same outputs as the
original
% network, but outputs are shifted left one timestep.
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
[xs,xis,ais,ts] = preparets(nets,{}, {},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)

```

APPENDIX K: Python code for ADXL345 accelerometer sensor

```

import smbus

import time

# ADXL345 registers

POWER_CTL = 0x2D

DATA_FORMAT = 0x31

DATA_X0 = 0x32

DATA_X1 = 0x33

DATA_Y0 = 0x34

DATA_Y1 = 0x35

DATA_Z0 = 0x36

DATA_Z1 = 0x37

# Raspberry Pi configuration

bus = smbus.SMBus(1) # Use I2C bus 1
address = 0x53 # ADXL345 address

# Initialize ADXL345

bus.write_byte_data(address, POWER_CTL, 0x08) # Enable measurement mode

bus.write_byte_data(address, DATA_FORMAT, 0x0B) # Set range to +/- 16g

# Read acceleration data

def read_acceleration(reg):

    # Read lower and upper bytes

    lower_byte = bus.read_byte_data(address, reg)

    upper_byte = bus.read_byte_data(address, reg + 1)

    value = -(65536 - value)

```

```

# Combine bytes into a signed 16-bit value

value = (upper_byte << 8) + lower_byte

if value & 0x8000: # Check for negative value

    value = -(65536 - value)

# Convert to acceleration in g

acceleration = value * 0.00390625 # 1 LSB = 0.00390625g

# Convert to acceleration in m/s^2

acceleration_m_s2 = acceleration * 9.8

return acceleration_m_s2

# Main program loop
try:
while True:
    # Read acceleration data
    x = read_acceleration(DATAX0)
    y = read_acceleration(DATAY0)

    z = read_acceleration(DATAZ0)

    # Print the values
    print("{:.2f} {:.2f} {:.2f}".format(x, y, z))

    # Wait for a while
    time.sleep(2)

except KeyboardInterrupt:

    pass

# Disable ADXL345

bus.write_byte_data(address, POWER_CTL, 0x00)

```

APPENDIX L: Python code for DS18B20 temperature sensor

```
from w1thermsensor import W1ThermSensor

import time

def read_temperature():

    # Initialize the sensor

    sensor = W1ThermSensor()

    # Get the temperature in Celsius

    temperature = sensor.get_temperature()

    return temperature

if __name__ == "__main__":

    try:

        while True:

            temp = read_temperature()

            print(f"{temp:.2f}")

            time.sleep(1) # Wait for 1 second before the next reading

    except KeyboardInterrupt:
```

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA