MACHINE LEARNING BASED SATELLITE COMPASS

Colomba Pathiranage Thilina Kumudika Kumara

208534 G

Master of Science in Artificial Intelligence

Department of Computational Mathematics
Faculty of Information Technology

University of Moratuwa
Sri Lanka

July 2024

MACHINE LEARNING BASED SATELLITE COMPASS

Colomba Pathiranage Thilina Kumudika Kumara

208534G

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree
Master of Science in Artificial Intelligence

Department of Computational Mathematics
Faculty of Information Technology

University of Moratuwa
Sri Lanka

May 2024

DECLARATION

I declare that this is my own work and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The above candidate has carried out research for the PhD/MPhil/Masters thesis/dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of Co-supervisor: Commodore (L) Pradeep Gunathilake

Signature of the Co-supervisor:

Name of Supervisor: Dr. Thilini Piyatilake

17/07/2024

Signature of the Supervisor: Date:

DEDICATION

The research thesis is dedicated wholeheartedly to my beloved parents, brother and sister who inspired, motivated and supported me throughout the study.

ACKNOWLEDGEMENT

I would like to thank my academic and research supervisor, Dr. Thilini Piyatilake, for her generous support and guidance throughout the research project. Also, I would like to express my sincere gratitude to my co-supervisor, Commodore (L) Pradeep Gunathilake, for the support and supervision rendered professionally in this study.

Special acknowledgment is forwarded to the Sri Lanka Navy for providing the opportunity to implement my design, being a part of my professional career development, in the naval platform.

Compliments of the thesis is awarded to my beloved family members for their valuable devotion played throughout the research.

ABSTRACT

This thesis documents the development, implementation, and evaluation of an ML-based Satellite Compass System customized for marine navigation. Motivated by the necessity for a low-cost, highly accurate, and reliable navigation sensor unaffected by interferences, the project addresses limitations present in conventional navigation sensors like Magnetic compass sensors and GPS-based satellite compasses. Central to the endeavor is the concept of "heading," representing a ship's direction relative to a reference point, such as magnetic or true north.

Our choice of an ML-based approach stems from its potential to handle the intricate relationships inherent in sensor data more effectively than traditional methods. Throughout the project, we meticulously selected and trained machine learning models, including LSTM, Random Forest, and XGBoost, to predict gyro readings from heading data. The integration of these models facilitated a comprehensive analysis of prediction accuracy, model robustness, and generalizability.

The results of our study underscore the efficacy of the ML-based Satellite Compass System. Through rigorous testing and evaluation, the XGBoost model emerged as the standout performer, demonstrating superior prediction accuracy and robustness compared to alternative models. This achievement represents a significant advancement in marine navigation technology, offering safer, more efficient, and technologically advanced solutions for maritime operations.

Keywords: Satellite Compass System, Marine Navigation, Machine Learning, Heading, XGBoost model

TABLE OF CONTENTS

LIST OF FIGURE

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
GPS	Global Positioning System
NMEA	National Marine Electronics Association
LSTM	Long Short-Term Memory
MSE	Mean Squared Error
MAE	Mean Absolute Error
RNN	Recurrent Neural Network
GUI	Graphical User Interface
ML	Machine Learning
RS485	Recommended Standard 485
HMI	Human-Machine Interface
PCB	Printed Circuit Board
ARM	Advanced RISC Machine
GLONASS	Global Navigation Satellite System
BeiDou	Chinese Satellite Navigation System
MPU	Microprocessor Unit
MPU	Microcontroller Unit
RMSE	Root Mean Square Error
IMU	Inertial Measurement Unit
UART	Universal Asynchronous Receiver-Transmitter
SHAP	SHapley Additive exPlanations
MLSCS	Machine learning Satellite Compass System

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Prolegomena

Ship heading, the direction in which a vessel's bow is pointing at any given time as depicted in figure 1.1, is a critical piece of information for safe and efficient maritime navigation. Navigational sensors play a vital role in determining and maintaining accurate ship heading, aiding in route planning, collision avoidance, and overall voyage safety. Traditional navigation tools, including magnetic compasses and gyro sensors, have long been relied upon for providing heading information. However, these tools face limitations and challenges in maintaining accuracy, especially in the presence of magnetic interference and varying environmental conditions.

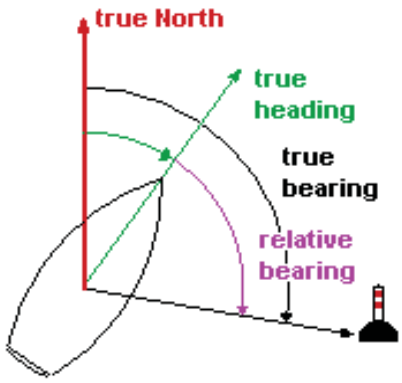

Figure 1. 1 Ship heading

Source : https://www.siranah.de/pictures/sail020f_C.gif

The inadequacy of existing maritime navigation systems to consistently provide accurate, low-cost, and reliable navigational sensors has been a persistent issue for maritime professionals. The limitations of current sensors hinder the maritime industry's ability to ensure precision and safety during voyages, particularly in challenging circumstances. Factors such as signal disruptions, magnetic interference, and environmental variations can compromise the reliability of traditional navigation tools, leading to potential navigational errors and safety risks.

To address these challenges and improve navigation accuracy and reliability, the development of an ML-based satellite compass system presents a promising solution.

By leveraging machine learning algorithms and satellite technology, the ML-based satellite compass aims to enhance the prediction of vessel heading with improved accuracy and reliability. This innovative navigation tool integrates data from GPS sensors, heading sensors, and conventional maritime gyroscopes to overcome the limitations of traditional sensors and provide a robust solution for maritime applications.

Through the utilization of advanced technology and data-driven approaches, the ML-based satellite compass system offers the potential to revolutionize maritime navigation by providing highly accurate, reliable, and cost-effective navigational sensor capabilities. By combining the strengths of machine learning with satellite technology, this system aims to address the shortcomings of existing sensors and empower maritime professionals with a versatile and efficient navigation tool for ensuring precision and safety during voyages.

1.2 Aim and Objectives

1.2.1 Aim

The aim of this research is to advance the field of maritime navigation by developing an innovative Machine Learning (ML)-based satellite compass that enhances the accuracy and reliability of vessel heading predictions in a maritime environment.

1.2.2 Objectives

Following objectives have been identified to develop Machine learning based Satellite Compass project

Objective 1 – Critically analyze literature on navigational systems for a comprehensive understanding of challenges in satellite compasses, magnetic compasses, and maritime gyroscopes.

Objective 2 – Develop a state-of-the-art ML-based satellite compass for improved vessel heading prediction, integrating GPS, heading sensor readings, and maritime gyroscopes.

Objective 3 – Design an intuitive GUI for seamless visualization and interaction, enhancing the user experience in interpreting and utilizing predicted vessel headings.

Objective 4 – Install and troubleshoot the ML-based compass on test vessels, rigorously evaluating its performance against traditional tools for accuracy, precision, and reliability.

Objective 5 – Preparation of thesis and paper for conference.

1.3 Background and Motivation

The navigation landscape within the maritime industry is evolving rapidly, driven by the demand for precision and cost-effective solutions. Traditional sensors such as heading sensors, satellite compasses, and gyro sensors each possess unique strengths and limitations are identified and depicted in Table 1.1.

Table 1. 1 Detailed comparison of navigational sensors

Feature	Gyro Sensor	Satellite Compass Sensor	Heading Sensor
Cost	High	Low	Low
Maintenance	Regular calibration	Minimal maintenance	Minimal maintenance
Accuracy	High	High	Moderate
Longevity	Moderate	High	Low
Reliability Concerns	Low risk of failure	Some risk of interference	Prone to interference
Redundancy	Not typically redundant	Redundancy options	Redundancy options
Inference	Limited data inference	Advanced data inference	High data inference
Ease of Integration	Moderate	Easy	Difficult
Availability	Limited availability	High availability	High availability
Heading Accuracy	High	High	Moderate
Industry Adoption	Low	High	Medium
Durability	High	Moderate	Moderate

While gyro sensors offer unparalleled accuracy, their high cost poses a significant barrier to widespread adoption. In contrast, satellite compass sensors and heading sensors provide average accuracy at a lower cost but are susceptible to satellite interruptions and magnetic interferences. Recognizing the need for a navigation solution that combines the accuracy of heading sensors, satellite compass sensors which map the reading of high accurate gyro reading. That will be leads to develop low cost, high accurate less susceptible of interferences novel navigational sensor known as ML based satellite compasses. Summary of comparisons of basic three types of sensors are depicted in Table 1.2

Table 1. 2 Summary of comparison

Feature	Satellite Compass	Heading Sensor	Maritime Gyroscope
Accuracy	High	Low	**High**
Cost	Low	Low	**High**
Reliability Concerns I (magnetic Interferences)	No	Yes	**No**
Reliability Concerns II (Vulnerable to satellite interruptions and interference)	Yes	No	**No**

The motivation behind designing an ML-based satellite compass lies in its potential to revolutionize maritime navigation by overcoming the most prominent limitations of traditional sensors as depicted in following Table.

By leveraging machine learning algorithms, we aim to create a navigation system that offers accuracy comparable to gyro sensors while significantly reducing costs. This cost-effectiveness opens doors for broader implementation across the maritime industry, enabling vessels of all sizes to benefit from precise navigation capabilities without incurring prohibitive expenses.

Moreover, the integration of machine learning into satellite compass technology represents a significant leap forward in navigation innovation. By harnessing the power of advanced algorithms, we can enhance the reliability of navigation systems, even in the face of environmental challenges such as magnetic interferences. The ML-based satellite compass not only addresses current limitations but also paves the way for future advancements in navigation technology, positioning the maritime industry at the forefront of technological innovation and efficiency.

1.4 Problem Definition

The problem at hand is the inadequacy of existing maritime navigation systems to consistently provide accurate, low cost and reliable navigational sensor, particularly in the face of magnetic interference, and varying environmental conditions, which hampers the maritime industry's ability to ensure precision and safety during voyages, especially in challenging circumstances

1.5 Machine Learning for Satellite Compass development

The application of Machine Learning (ML) techniques in the development of satellite compasses offers promising avenues for improving the accuracy and reliability of vessel heading predictions. ML algorithms have the capability to analyze complex data patterns and relationships, allowing for the creation of predictive models that can mitigate the effects of environmental disturbances and enhance navigation performance.

In the context of this project, ML algorithms are utilized to develop a satellite compass system that integrates data from multiple sources, including GPS sensors, additional heading sensor readings, and conventional maritime gyroscopes. By training ML models on these diverse datasets, the system can learn to accurately predict vessel headings in real-time, even in challenging maritime conditions.

The ML-based approach involves preprocessing the input data, selecting appropriate algorithms, and optimizing model parameters to achieve the desired performance metrics. Various ML algorithms such as Random Forest, XGBoost, and LSTM (Long Short-Term Memory) are explored and evaluated to determine the most effective solution for satellite compass development.

The chosen ML models are trained on historical data collected from test vessels, with an emphasis on capturing the dynamic nature of maritime environments. Model performance is assessed through rigorous validation and testing procedures, ensuring that the developed satellite compass system meets the required standards of accuracy, precision, and reliability. Overall, the integration of ML techniques in satellite compass development represents a significant advancement in maritime navigation technology, offering enhanced capabilities for vessel positioning and route optimization in diverse operational scenarios.

1.6 System Requirements

The designed system requires the following resources;

1.6.1 Hardware

a. GPS Sensors: High-quality GPS sensors capable of capturing accurate location data.
b. Heading Sensors: Precision heading sensors compatible with maritime vessels, providing reliable heading measurements.
c. Maritime Gyroscope: A reliable gyroscope with high accuracy, serving as a reference for data labeling.
d. Electronic Circuitry: Custom-designed circuitry for real-time data acquisition and processing.
e. Test Vessels: Access to vessels equipped with necessary infrastructure for system testing and data recording.

1.6.2 Software

f. Machine Learning Framework: Software tools and libraries for developing and training ML models, such as TensorFlow, PyTorch, and scikit-learn.
g. Graphical User Interface (GUI) Development Tools: Software for designing an intuitive user interface for data visualization and interaction.
h. Data Analysis and Visualization Tools: Software for preprocessing, analyzing, and visualizing collected data.

1.6.3 Computing Resources

i. High-performance computing resources for ML model training and testing.
j. Server Infrastructure: A server environment for hosting the ML-based satellite compass system.

1.6.4 Data Storage

k. Data Storage Systems: Secure and scalable data storage solutions to store collected data for training and evaluation purposes.

1.7 Outline of the Thesis

The structure of the thesis comprises several distinct chapters, each serving a specific purpose in the exploration and development of the ML-based satellite compass system. Beginning with an Introduction, the thesis sets the stage by providing an overview of the research topic, objectives, and motivation. Following this, the Literature Review critically assesses existing literature on satellite compasses, magnetic compasses, and maritime gyroscopes, elucidating challenges and limitations in current navigation systems. The Methodology chapter delineates the approach taken for developing the ML-based satellite compass, detailing data collection, model development, GUI design, system deployment, and evaluation procedures. Subsequently, the Results chapter presents the findings of the research, including performance evaluations of the developed system and comparisons with traditional navigation tools. The Discussion chapter delves into the implications of the findings, explores their significance for maritime navigation, and identifies avenues for future research. Finally, the Conclusion chapter summarizes the key findings, highlights contributions to the field, and underscores the broader impact of the research. The thesis concludes with a comprehensive References section listing all sources cited, along with an Appendix containing supplementary materials such as code implementations and datasets. This structured approach facilitates a thorough exploration of the research topic, providing a cohesive narrative that contributes to the advancement of maritime navigation

1.8 Summary

In summary, this thesis aims to address the challenges in maritime navigation by developing an innovative ML-based satellite compass system. Through a comprehensive exploration of literature, the development of ML models, and rigorous evaluation procedures, the research endeavors to enhance the accuracy and reliability of vessel heading predictions. By integrating ML techniques into satellite compass development, this project seeks to contribute to the advancement of maritime navigation technology, ultimately improving safety and efficiency in maritime operations.

CHAPTER 2

LIETERATURE REVIEW ON MACHINE LEARNING BASED SATELLITE COMPASS DEVELOPMENT

2.1 Introduction

Maritime navigation has evolved significantly over the years, driven by advancements in navigational sensor technologies onboard ships. This literature review explores the early developments of navigational sensors, the latest approaches, and the limitations of present available sensors

2.2 Early Developments of Navigational sensors onboard Ships

The earliest navigational sensors onboard ships date back centuries, with magnetic compasses being the fundamental tool for ship navigation. These compasses, though reliable, are susceptible to various limitations. They must measure at least two orthogonal axes of the Earth's geomagnetic field with high accuracy, and they are sensitive to thermal and magnetic shocks. Additionally, magnetic compasses can be vulnerable to interference from iron magnetic materials, leading to significant deviations in readings [1].

Over time, advancements led to the introduction of gyrocompasses on ships. Mandated by the International Convention for the Safety of Life at Sea (SOLAS), gyrocompasses offer stability and accuracy surpassing that of magnetic compasses. Traditional gyrocompass designs have evolved into more advanced digital solutions, including laser gyrocompasses and fiber-optic gyrocompasses (FOGs). These modern gyrocompasses provide sTable and accurate heading information, making them suiTable for various vessel types [1 - 7].

2.3 Latest Approaches of Navigational sensors on board ships

Satellite compasses, also known as GPS compasses, represent a recent approach to navigation. These compasses leverage signals from multiple GPS satellites to determine a ship's heading. With high precision and global coverage, satellite compasses enable vessels to navigate accurately worldwide. However, they face

challenges such as signal disruptions and alignment difficulties during initial setup [4, 5].

In addition to satellite compasses, other types of navigational sensors such as gyroscopes and fiber-optic gyrocompasses are employed onboard ships. Each sensor type exhibits varying metrological characteristics, including dynamic responses and oscillations. Understanding these characteristics is crucial for accurate heading determination and the prevention of ship accidents [3, 7].

2.4 Limitations of present available Navigational sensors

Despite their advancements, present available navigational sensors have inherent limitations. Traditional magnetic compasses are susceptible to interference and deviations, particularly in the presence of magnetic materials. Gyrocompasses, while offering stability and accuracy, may be inaccessible to smaller vessels due to cost considerations. Satellite compasses, though offering global coverage, face challenges such as signal disruptions and alignment difficulties [1].

The limitations of present navigational sensors underscore the need for innovative solutions to enhance navigation accuracy and reliability. This has led to the exploration of machine learning-based approaches to navigation, aiming to combine the strengths of satellite and heading sensor technologies. By leveraging machine learning techniques, researchers aim to develop comprehensive navigation systems that address the limitations of existing tools and provide reliable alternatives for maritime navigation in an increasingly interconnected world [1-7].

2.5 Problem Definition

The problem at hand is the inadequacy of existing maritime navigation systems to consistently provide accurate, low cost and reliable navigational sensor, particularly in the face of magnetic interference and varying environmental conditions, which hampers the maritime industry's ability to ensure precision and safety during voyages, especially in challenging circumstances

2.6 Summary

In conclusion, the evolution of navigational sensors onboard ships has been marked by significant advancements, from traditional magnetic compasses to modern satellite-based systems. However, despite these advancements, present available sensors still face limitations, necessitating further research and innovation. The exploration of machine learning-based approaches to navigation holds promise in overcoming these limitations and providing more accurate and reliable navigation solutions for the maritime industry.

CHAPTER 3

TECHNOLOGY ADOPTED

3.1 Introduction

In the development of the machine learning-based navigational solution, several technologies were adopted and integrated to achieve accurate and reliable predictions of gyro readings. This chapter discusses the technologies utilized in the Satellite Compass development process. The proposed solution encompasses data collection, model development, and system implementation. The core components of this solution include the utilization of maritime gyroscope data as labels for ML model training, featuring three GPS sensor values and additional heading sensor values as crucial inputs as depicted in Figure 3.1. Custom electronic circuitry has been designed, and implemented for real-time data acquisition during vessel navigation.

ML based satellite compass that leverages data from three GPS sensors and additional heading sensor readings as 'features' (inputs), with conventional maritime gyroscope readings (ANCHUTZ STD 22) serving as the 'label' (output). The primary goal is to create a more accurate and reliable navigation tool for maritime applications.

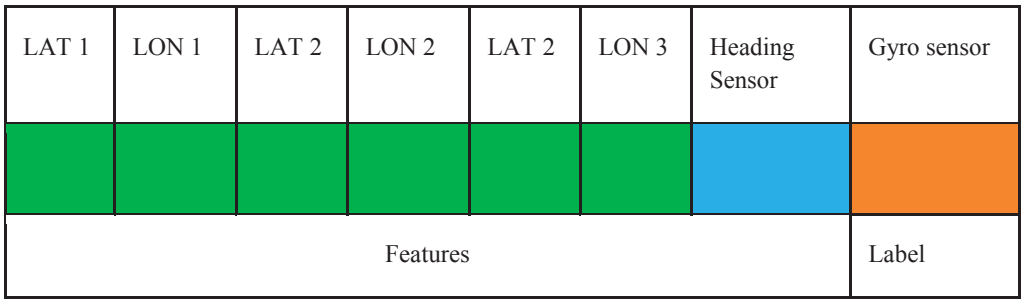

Figure 3. 1 Features and labels of the model

3.2 Machine Learning Models

3.2.1 LSTM (Long Short-Term Memory)

LSTM neural networks were employed to capture temporal dependencies in the sequential data of heading values. LSTM networks are well-suited for time series data [8-12] and can effectively model long-range dependencies, making them suitable for predicting gyro readings based on historical heading data.

The code snippet outlines the construction and training of a Long Short-Term Memory (LSTM) neural network model using TensorFlow and Keras for time series regression tasks. Initially, the input data is segmented into sequences of a predefined length to capture temporal dependencies within the data. This preprocessing step allows the model to learn patterns from past observations, essential for accurate predictions. Subsequently, the dataset is divided into training and validation sets to assess the model's performance on unseen data, ensuring its ability to generalize beyond the training samples.

The LSTM model architecture comprises stacked LSTM layers, each configured with a specific number of units to capture varying levels of temporal dependencies [9]. Dropout regularization is incorporated after each LSTM layer and the intermediate dense layer to mitigate overfitting by randomly deactivating a fraction of input units during training. The choice of Rectified Linear Unit (ReLU) activation functions introduces non-linearity to the model, enabling it to capture complex relationships within the data. The model's output layer employs a linear activation function suited for regression tasks, facilitating the prediction of continuous values. During training, the model is optimized using the Adam optimizer and mean squared error (MSE) loss function, iteratively updating its parameters to minimize prediction errors.

Following model compilation, training involves iterating over the dataset for a specified number of epochs, with the model's performance monitored via training and validation loss. The resulting trained model can then be utilized to make predictions on both the training and validation sets. By employing LSTM-based deep learning techniques, the model effectively captures temporal patterns in sequential data, offering promising prospects for accurate regression predictions. Through careful design and training, the LSTM model presents a robust framework for time series

forecasting tasks, demonstrating its potential for applications in diverse domains requiring predictive analytics based on sequential data.

3.2.2 Random Forest Regression:

Random Forest regression was utilized as a complementary approach to LSTM. This ensemble learning technique combines multiple decision trees to improve prediction accuracy and robustness. Random Forests are capable of handling non-linear relationships and capturing complex interactions between features, making them suitable for predicting gyro readings based on historical heading sequences [13-17]

Initially, the Random Forest model is instantiated with specified hyperparameters, such as the number of decision trees (*n_estimators*) and a random seed (random_state) for reproducibility. This ensemble model aggregates predictions from multiple decision trees, each trained on a different subset of the dataset, thereby reducing the risk of overfitting and improving prediction accuracy.

Upon instantiation, the Random Forest model is trained using the *fit()* method, where the training data, comprised of sequential input sequences (*X_train_seq*) and their corresponding target values (*y_train_seq*), are provided. The input data is reshaped into a two-dimensional array to comply with the expected format of the Random Forest algorithm, ensuring compatibility with the training process. During training, each decision tree in the ensemble learns to predict the target variable based on different features and samples from the dataset, promoting diversity and robustness in the overall model.

Post-training, the trained Random Forest model is utilized to make predictions on both the training and validation datasets using the *predict()* method. The input sequences from the training and validation sets are reshaped similarly to align with the model's input format. The predictions generated by the Random Forest model provide estimates of the target variable (gyro readings) based on the learned relationships between input features (heading sensor data) and the target variable observed during training. This process showcases the model's ability to generalize to unseen data, thereby facilitating the assessment of its predictive performance and suitability for the regression task at hand.

3.2.3 XGBoost (Extreme Gradient Boosting):

XGBoost is a powerful gradient boosting algorithm that was employed to further enhance the prediction performance. XGBoost iteratively builds a series of decision trees, each correcting the errors of its predecessors, leading to highly accurate predictions. XGBoost is known for its efficiency, scalability, and ability to handle large datasets, making it a suitable choice for predicting gyro readings based on combined features from LSTM and Random Forest models [18-22].

The described approach involves combining predictions from two distinct models, LSTM and Random Forest, to create a composite feature set for training an XGBoost regression model. Initially, the predictions generated by the LSTM and Random Forest models on the training dataset are concatenated horizontally using the np.column_stack() function, resulting in a combined feature matrix (*X_train_combined*). This matrix contains both the predictions from the LSTM model (*y_pred_lstm_train*) and the Random Forest model (*y_pred_rf_train*) as features, facilitating the creation of a richer feature representation that captures complementary aspects of the data.

Subsequently, a similar process is applied to the validation dataset, resulting in the creation of the combined feature matrix (*X_val_combined*). This matrix comprises the predictions from both the LSTM and Random Forest models for each instance in the validation set, enabling the XGBoost model to make predictions based on the collective insights provided by these models.

Following the creation of the combined feature matrices, an XGBoost regression model is instantiated and trained using the X_train_combined matrix as input features and the target variable (*y_train_seq*). Leveraging the gradient boosting framework, XGBoost iteratively builds an ensemble of weak learners (decision trees) to minimize the loss function and optimize prediction accuracy. By learning from the combined features derived from the LSTM and Random Forest models, the XGBoost model can capture intricate relationships and patterns in the data, potentially leading to enhanced predictive performance.

Once trained, the XGBoost model is utilized to make predictions on both the training and validation datasets using the predict () method. The combined feature matrices (*X_train_combined* and *X_val_combined*) serve as input to the model, and the resulting

predictions (*y_pred_xgb_train* and *y_pred_xgb_val*) represent the XGBoost model's estimates of the target variable (gyro readings) based on the combined insights gleaned from the LSTM and Random Forest models. This ensemble approach leverages the strengths of each individual model, potentially yielding superior predictive performance compared to any single model in isolation.

3.2.4 Data processing

The data processing technology used a crucial aspect of a data preprocessing pipeline tailored for machine learning applications. Initially, imports essential libraries like NumPy, Pandas, and scikit-learn modules, laying the foundation for data manipulation, model training, and evaluation. Through the use of these libraries, the script facilitates tasks such as data loading, cleaning, and splitting into training and testing sets. Leveraging the capabilities of these libraries, particularly Pandas for data manipulation and scikit-learn for machine learning functionalities, ensures a robust and efficient workflow for handling tabular data.

The subsequent sections of the code focus on data cleaning and preparation, ensuring the dataset's integrity and suitability for machine learning tasks. This involves steps such as removing null values, filtering out erroneous sensor readings exceeding predefined thresholds, and calculating additional features like true north angles from GPS coordinates where harversian formula has used to calculate the additional feature from gps coordinates.

These preprocessing steps are crucial for enhancing the quality of the dataset, mitigating potential issues like missing values and outliers, which could adversely affect model performance. By systematically addressing data quality issues, the code fosters a conducive environment for model training and evaluation, ultimately improving the predictive accuracy and reliability of the machine learning model.

Furthermore, the code employs techniques like feature scaling using the RobustScaler to normalize the features, making them more amenable to the learning algorithms. Additionally, it splits the dataset into training and testing subsets, a fundamental practice in machine learning to assess model generalization performance. This division

enables the model to learn patterns from the training data while evaluating its performance on unseen data to gauge its effectiveness.

Overall, the code encapsulates a comprehensive data preprocessing pipeline, essential for preparing the dataset for subsequent machine learning tasks, thereby contributing to the development of robust predictive models in various applications. In summary,

Data Cleaning: The input data underwent preprocessing steps to ensure quality and reliability. Null rows were removed, and gyro readings of zero were filtered out to eliminate noise and ensure data integrity.

Normalization: The heading values were normalized using Standard Scaler to bring them within a consistent range and prevent any single feature from dominating the learning process. Normalization helps in stabilizing the training process and improving convergence during model.

3.2.5 Complete model

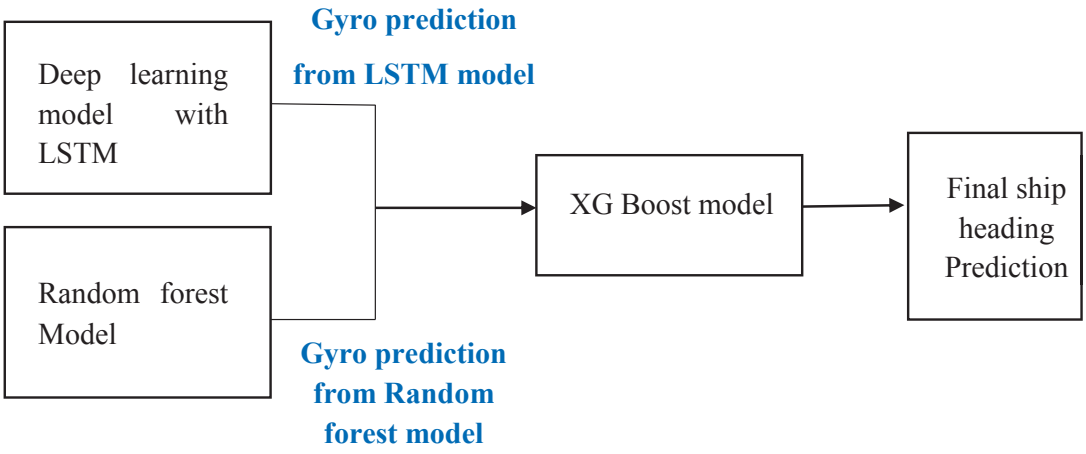

Figure 3. 2 Architecture of the final Model

The finalized hybrid model integrates outputs from both Random Forest and Deep Learning (LSTM) models to improve the accuracy of predicting gyro readings for maritime navigation. Figure 3.1 Shows the architecture of the proposed model. Initially, the data preprocessing phase involved cleaning the dataset by removing null

rows and entries with zero gyro readings. Descriptive statistics were examined to gain insights into the dataset's distribution and characteristics.

For the Deep Learning component, LSTM models were employed to capture temporal dependencies in the data. The input sequences were created from the heading sensor readings, with a sequence length of 10 timesteps. The LSTM architecture consisted of multiple layers, including dropout regularization to mitigate overfitting. The model was trained using the Adam optimizer and mean squared error loss function.

Simultaneously, a Random Forest regressor was trained on the same input sequences of heading sensor readings. This ensemble model comprised multiple decision trees, providing robust predictions while accommodating nonlinear relationships in the data.

Upon training completion, predictions were obtained from both the LSTM and Random Forest models for the training and validation datasets. These predictions were then combined to form a new feature set, which served as inputs to the final XGBoost model. The XGBoost model, a gradient boosting ensemble technique, was trained to further refine the predictions and capture any remaining patterns in the data.

The trained LSTM, Random Forest, and XGBoost models were saved for future use in the Google Drive environment. Evaluation metrics such as mean squared error, mean absolute error, R-squared, and median absolute error were calculated to assess the performance of each model on both the training and validation datasets. These metrics provided insights into the models' predictive capabilities and generalization abilities.

Lastly, visualizations were generated to compare the actual gyro readings with the predictions made by each individual model (LSTM, Random Forest, and XGBoost). These scatter plots allowed for a qualitative assessment of the models' performance and highlighted any discrepancies between actual and predicted gyro readings. Overall, the hybrid model demonstrated enhanced predictive accuracy, leveraging the strengths of both Deep Learning and traditional machine learning techniques for improved gyro predictions in maritime navigation.

3.2.6 Selection of LSTM, Random Forest, and XGBoost Models

3.2.6.1 Requirement for LSTM, Random Forest, and XGBoost

The selection of LSTM, Random Forest, and XGBoost models is driven by the unique strengths and capabilities of each algorithm, which collectively address the complex requirements of predicting gyro sensor readings for real-time heading prediction applications. Each model brings a distinct set of attributes that complement one another, ensuring comprehensive and accurate predictions.

3.2.6.2 Why These Models Were Selected

1. **LSTM (Long Short-Term Memory) Networks:**
    a. **Temporal Pattern Recognition:** LSTM networks, a type of recurrent neural network (RNN), are specifically designed to capture temporal patterns and long-term dependencies in sequential data. This is crucial for gyro sensor readings, which are inherently time-series data.
    b. **Sequential Data Handling:** The nature of gyro sensor data involves sequences where current readings depend on previous readings. LSTMs are adept at maintaining a memory of past data points, making them ideal for this task.
    c. **Mitigating Vanishing Gradient Problem:** Traditional RNNs suffer from the vanishing gradient problem, where long-term dependencies are not effectively learned. LSTM's architecture includes cell states and gates that help preserve information over longer sequences, effectively mitigating this issue.
2. **Random Forest:**
    a. **Handling Non-linear Relationships:** Random Forest is an ensemble learning method that constructs multiple decision trees and merges their outputs to improve accuracy and control over-fitting. It is proficient in handling non-linear relationships between features, which are common in sensor data.

b. **Robustness to Outliers and Noise:** The aggregation of multiple decision trees in Random Forest makes it robust to outliers and noisy data, ensuring stable and reliable predictions even when the input data is not perfectly clean.

c. **Feature Importance:** Random Forest provides insights into feature importance, helping to understand which sensor readings are most influential in predicting the gyro sensor outputs.

3. **XGBoost (Extreme Gradient Boosting):**

a. **High Predictive Power:** XGBoost is an optimized gradient boosting algorithm that excels in prediction accuracy due to its ability to handle both linear and non-linear relationships with high efficiency.

b. **Handling Large Datasets:** XGBoost is designed for speed and performance, making it suitable for handling large datasets with high-dimensional features, which is often the case in real-time heading prediction systems.

c. **Ensemble of Models:** By combining the predictions of weak learners (individual trees), XGBoost enhances overall model performance. It can integrate the strengths of both LSTM and Random Forest models by using their predictions as additional features, thereby boosting accuracy.

3.2.6.3. How the Accuracy Improved with This Selection?

1. **Complementary Strengths:**

a. LSTM models excel at capturing temporal dependencies, while Random Forest and XGBoost are strong in handling non-linear relationships and high-dimensional data.

b. By combining these models, we leverage the sequential modeling capability of LSTM with the robustness and predictive power of Random Forest and XGBoost, leading to improved accuracy.

2. **Ensemble Learning:**
    a. The integration of LSTM and Random Forest predictions as features in the XGBoost model creates a powerful ensemble that captures a wider range of patterns and relationships in the data.
    b. This approach reduces the likelihood of overfitting by balancing the strengths and weaknesses of each individual model, resulting in more reliable and accurate predictions.
3. **Improved Generalization:**
    a. The diverse nature of the models ensures that the final ensemble can generalize better to unseen data. LSTM models focus on capturing temporal sequences, while Random Forest handles feature interactions, and XGBoost fine-tunes the overall prediction with gradient boosting.

3.2.6.4 Relevance of Techniques Selection to Real-Time Heading Prediction

1. **Real-Time Data Processing:**
    a. In real-time heading prediction applications, the system needs to process continuous streams of sensor data quickly and accurately. LSTM's ability to handle sequential data in real-time aligns well with this requirement.
    b. Random Forest and XGBoost models, with their fast prediction capabilities, ensure that the system can provide timely and accurate heading predictions without significant lag.
2. **Robustness and Reliability:**
    a. Real-time applications must be robust to variations in sensor data due to environmental factors or sensor noise. The combination of these models enhances the system's resilience to such variations.
    b. The ensemble approach ensures that the final predictions are more stable and less prone to errors, which is critical for applications like maritime navigation where accurate heading predictions are essential for safe operation.

3. **Scalability and Flexibility:**

    a. The chosen models are scalable and can handle increases in data volume and complexity. As more sensors are integrated or higher frequency data is processed, the models can be adjusted and retrained to maintain high performance.

    b. The flexibility of these models allows for continuous improvement and adaptation to changing conditions, ensuring that the system remains effective over time.

3.3 Summary

In summary, Chapter 3 delves into the technological underpinnings of the machine learning-based navigational solution developed for maritime applications. By integrating various machine learning models such as LSTM, Random Forest, and XGBoost, the solution aims to accurately predict gyro readings based on input data from GPS sensors and heading sensors. The chapter highlights the importance of data preprocessing in ensuring the integrity and suitability of the dataset for machine learning tasks. Additionally, it elucidates the architecture and training process of the hybrid model, which combines outputs from multiple models to enhance predictive accuracy. Overall, Chapter 3 underscores the significance of leveraging advanced technologies to address the challenges associated with maritime navigation, offering a robust and reliable solution for improved gyro predictions.

CHAPTER 4

APPROACH OF DESIGNING MACHINE LEARNING BASED SATCOMPASS

4.1 Introduction

In chapter 3, we have presented an in-depth study of AI technologies adopted for ML based Satellite Compass model training. Our approach to AI based navigational sensor for ships is named as MLSCS, an abbreviation for Machine Learning-based Satellite Compass System. Here we present MLSCS with its hypothesis, input, output, process, features of MLSCS. In the pursuit of precision and reliability in maritime navigation, the development of an innovative Machine Learning (ML)-based satellite compass system emerges as a beacon of technological advancement. By harnessing the power of ML algorithms and integrating sensor data, a new era of accurate and dependable vessel heading prediction dawns upon the maritime industry. This chapter delves into the intricacies of designing a cutting-edge ML-based satellite compass, paving the way for enhanced navigation capabilities in maritime transportation.

4.2 Hypothesis

It is hypothesized that the fusion of ML algorithms with diverse sensor data sources will revolutionize the accuracy and reliability of satellite compass systems. By leveraging insights from three GPS sensors, heading sensor readings, and maritime gyroscope data, the ML model is expected to deliver precise vessel heading predictions. The integration of ML techniques holds the promise of significantly improving navigation accuracy, thereby enhancing safety and efficiency in maritime operations.

4.3 Inputs

The input data for training the ML model comprises information from three GPS sensors, heading sensor readings, and maritime gyroscope data. Each input data type plays a crucial role in enriching the model's understanding of vessel orientation and movement patterns. The GPS sensors provide geographical positioning data, heading sensor readings offer directional information, while maritime gyroscope data

contributes to stability and reference points for the model. Collectively, these inputs enable the ML model to learn and predict vessel headings with enhanced accuracy.

These sensor data are fed as the features/inputs and applied to ML model. The sensors utilized to obtain real time data against the input variable are tabulated below in Table 4.1.

Table 4. 1 Sensors used for Input Variables

Input Variable	Sensor
GPS Sensor	NEO 9M GPS
Heading sensor	PG-500
Gyro sensor	Standard 22 type

4.3.1 Latitude and longitude readings

The Neo M GPS sensor provides crucial geographical positioning data, offering accurate and real-time information about the vessel's location on the Earth's surface. This sensor utilizes signals received from satellites to triangulate the vessel's position, enabling precise navigation across vast maritime expanses. By integrating data from the Neo M GPS sensor, the ML model gains insights into the vessel's coordinates, facilitating accurate heading predictions relative to its geographic position.

4.3.2 Heading value

The PG 500 heading sensor plays a pivotal role in providing directional information to the ML-based satellite compass system. This sensor measures the vessel's heading or orientation relative to the Earth's magnetic field, aiding in the determination of its course and trajectory. By capturing data on the vessel's heading, the PG 500 sensor enhances the accuracy of the ML model's predictions, ensuring that the compass readings align closely with the vessel's intended direction of travel. Figure 1.1 shows the meaning of heading value with respect to the true north.

4.3.3 Gyro heading value

The Standard 22 gyro compass, serving as the gyro sensor in the system, offers stability and reference points crucial for accurate navigation. This sensor measures the vessel's

orientation based on gyroscopic principles, providing reliable data on its pitch, roll, and yaw motions. By incorporating data from the gyro sensor, the ML model gains insights into the vessel's stability and dynamic movements, enhancing the robustness of its heading predictions, especially in challenging sea conditions or when subjected to external forces.

4.4 Output

The output of the ML model is the predicted vessel heading based on the amalgamation of input sensor data. By processing the incoming data streams, the ML model generates real-time compass readings that reflect the current orientation of the vessel. This predictive capability empowers the ML-based satellite compass to furnish accurate heading information essential for safe and efficient maritime navigation

4.5 Process

In the process of deploying the ML-based satellite compass system on a Jetson Nano environment for real-time prediction of vessel heading, several key steps were involved. Initially, the training phase entailed providing the ML model with labeled data, which included inputs from three GPS coordinates (latitude, longitude) and heading sensor readings. This labeled dataset served as the foundation for the model to learn underlying patterns and relationships between input features and corresponding vessel headings. During training, various ML algorithms, such as Deep Learning with LSTM, Random Forest, or XG Boost, were utilized, and the model iteratively adjusted its parameters to minimize the difference between predicted and actual headings observed in the training data. Additionally, the training process involved hyper parameter tuning, cross-validation, and performance evaluation to optimize the model's accuracy and generalization capabilities.

Once the ML model was trained and achieved satisfactory performance metrics, it needed to be saved for later use in the Jetson Nano environment. This involved serializing the trained model object into a file format, such as HDF5, Pickle, or ONNX, which could be easily loaded and utilized for prediction tasks. The saved model file was then transferred to the Jetson Nano device's storage, and a Python script or application was developed to load the model into memory using appropriate libraries like TensorFlow, PyTorch, or scikit-learn.

In the real-time prediction scenario, the Jetson Nano received inputs from three GPS sensors providing coordinates (latitude, longitude) and the heading sensor reading. These inputs were preprocessed to ensure they matched the format used during training and were scaled if necessary to match the input range. The preprocessed inputs were fed into the loaded ML model, which generated a predicted vessel heading as the output. This predicted heading represented the orientation of the vessel relative to its geographic position and direction of travel. Continuous updates to the predicted heading could be made in real-time as new sensor data became available, allowing for dynamic navigation adjustments and course corrections as needed.

Through these steps, the ML-based satellite compass system was effectively deployed on the Jetson Nano environment, facilitating real-time prediction of vessel headings based on inputs from GPS coordinates and heading sensor readings. This deployment enhanced maritime navigation by providing accurate and reliable heading information essential for safe and efficient vessel operations.

4.6 Features

The ML-based satellite compass system boasts key features such as data fusion from multiple sensors, neural network-based algorithms for predictive analytics, and real-time prediction capabilities. By amalgamating data from diverse sources, the system enhances accuracy and minimizes errors in heading predictions. Neural network algorithms enable the model to learn and adapt to changing environmental conditions, ensuring continuous performance optimization. Furthermore, real-time prediction capabilities provide instantaneous and reliable compass readings, setting it apart from traditional navigation tools.

4.7 Users

The target users of the ML-based satellite compass system encompass maritime navigators, naval vessels, and commercial shipping companies seeking advanced navigation solutions. Addressing user requirements for precise and reliable heading information, the system prioritizes usability factors and intuitive design elements. Potential training programs will be tailored to facilitate the seamless adoption of the ML-based compass, empowering users to leverage its capabilities effectively in their navigation tasks.

4.8 Summary

In conclusion, the approach of designing an ML-based satellite compass heralds a new era of precision and reliability in maritime navigation. By integrating ML algorithms with sensor data, the system aims to revolutionize vessel heading predictions, enhancing safety and efficiency in maritime operations. Through a meticulous design process, key features such as data fusion, neural network algorithms, and real-time prediction capabilities are incorporated to differentiate the ML-based compass from traditional navigation tools. With a focus on user requirements and usability factors, the system targets maritime navigators, naval vessels, and commercial shipping companies, offering a transformative solution for accurate and dependable navigation in maritime applications.

CHAPTER 5

DESIGNING OF ML BASED SATELLITE COMPASS SYSTEM

5.1 Introduction

In the realm of satellite navigation and communication, the development of accurate and reliable orientation systems is paramount. One such advancement is the Satellite Compass system, a sophisticated integration of various sensors including GPS (Global Positioning System), gyro, and heading sensors. These sensors collectively provide crucial data necessary for determining the precise orientation of a satellite in space. However, the effectiveness of the Satellite Compass system heavily relies on the quality of sensor data it receives.

In the initial stages of designing the Satellite Compass system, meticulous attention must be paid to both the hardware circuitry and the software programming to ensure the collection of accurate sensor data with minimal noise. The hardware circuitry involves the integration of GPS modules, gyro sensors, heading sensors, and microcontroller units such as Arduino Due and Arduino Mega. Each component must be carefully selected and configured to minimize interference and maximize signal integrity.

Moreover, the programming of the microcontroller units plays a vital role in efficiently acquiring, processing, and transmitting sensor data. Arduino-based programming facilitates the real-time collection and transmission of sensor readings, necessitating precise algorithms for data parsing, decoding, and formatting. Additionally, strategies for error handling and noise reduction are crucial to ensure the reliability of the collected data.

In essence, the Satellite Compass system represents a convergence of cutting-edge sensor technologies and sophisticated machine learning algorithms. The success of this system hinges on the meticulous design and implementation of hardware circuits and software programs that enable the acquisition of accurate sensor data with minimal noise. In the subsequent sections, we delve into the intricacies of designing and developing an ML-based Satellite Compass system, highlighting the critical steps involved in sensor integration, data preprocessing, machine learning model training,

and system validation. Through this comprehensive approach, we aim to realize a robust and precise orientation system essential for the navigation and communication capabilities of satellites in space.

An embedded system comprises with sensors, microcontrollers, microprocessors, HMIs, GUIs, actuators, etc. All the outputs of sensors used in this study are fed to a microcontroller. Pre-processed sensor data are fed to a microprocessor for application of ML based satellite compass system. A GUI is designed to make the system user friendly.

The programming languages used in the project are python and arduino. Further, several communication protocols such as RS 485 NMEA format, serial, etc also have been used for inter communication between sensors, microcontroller and the microprocessor.

5.2 Sensors

A real time processing Satellite Compass system requires a continuous real data feed. The inputs are taken from following sensors thus fed for ML model to obtained using the prediction. Following list of sensors have been installed in the ML based Satellite Compass system in the phase of data collection as indicated in Figure 5.1, the system block diagram for data collection circuit:

a. GPS sensors
b. Heading sensor
c. Gyro sensor

Only following list of sensors have been used to develop the system to obtain real time predictions and the system block diagram for data collection is depicted in following figure.

a. GPS sensors
b. Heading sensor

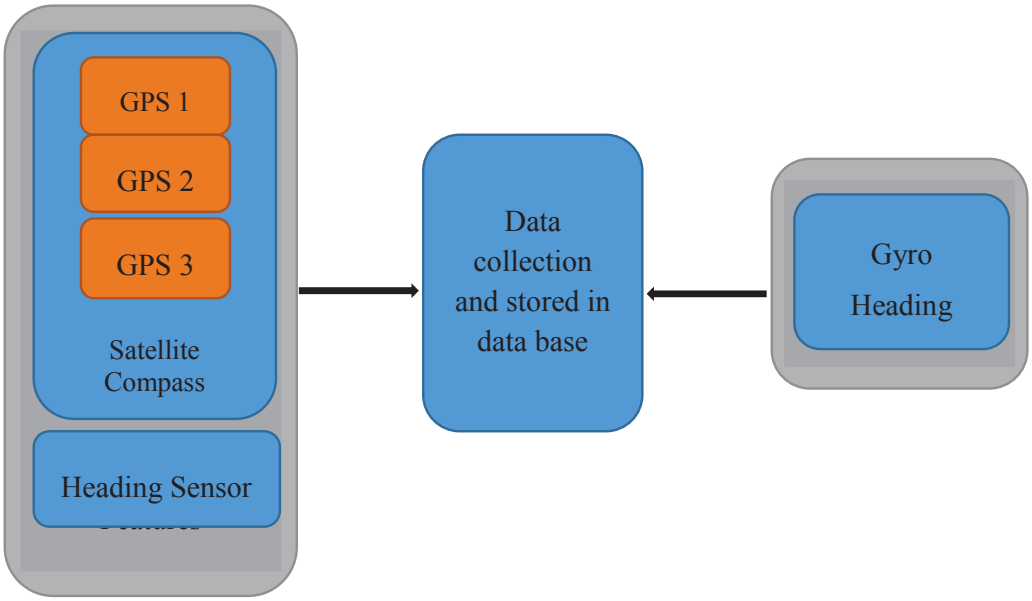

Figure 5. 1 Block diagram of data collection circuit

5.2.1 Heading Sensors

The Furuno PG-500 is a compact and low cost heading sensor designed for pleasure craft and coastal fishing vessels. It utilizes a fluxgate compass to detect the Earth's magnetic field and a solid-state rate gyroscope to stabilize the compass heading. This combination provides moderately accurate of vessel's heading relative to magnetic north

The PG-500 sensor offers some level of accurate heading information crucial for navigation systems requiring precise orientation data. With a heading accuracy of up to 0.5 degrees and a fast update rate of 10 Hz, the sensor enables real-time monitoring of heading changes, facilitating quick and responsive navigation adjustments.

The PG-500 outputs heading data in various formats, including NMEA 0183, which integrates seamlessly with our project's communication protocol. NMEA 0183 is a serial communication standard widely used in marine electronics and offers a structured format for transmitting data between devices. The PG-500 likely transmits

heading data using the NMEA sentence type HDT (Head True). This sentence includes information like the true heading (magnetic heading corrected for variation), magnetic variation, and checksum for error detection. By integrating the PG-500's HDT data with the sensor readings and GPS position information, we build a comprehensive dataset that reflects the environmental conditions at a specific location with a specific heading. A sample image of a floater sensor is shown in figure 5.1.

Figure 5. 2 PG-500 heading sensor

Source : https://www.cactusnav.com/images/ZFURPG500.jpg

5.2.2 Gyro sensor

The Standard 22 NX stands as a beacon of reliability and precision in the realm of navigation. Crafted by Anschütz, this gyro compass embodies the pinnacle of technological prowess, offering unrivaled accuracy and operational safety even amidst the most challenging maritime environments. Designed to cater to a wide array of vessels, its robust construction ensures steadfast performance regardless of conditions.

Equipped with interfaces for serial data, Ethernet, and Bridge Alert Management, the Standard 22 NX facilitates seamless installation and integration into existing systems.

In the context of a this project, the Anschutz STD 22 Gyro Compass serves as a vital component in ensuring precise heading data collection for machine learning applications to serve as the label for the model training. The gyro compass's ability to provide real-time heading corrections, alignment error detection, and speed error

correction facilitates the development of advanced navigation algorithms and systems that rely on accurate heading information for optimal performance.

Regarding data output formats, the Anschutz STD 22 Gyro Compass typically provides outputs in industry-standard formats such as NMEA (National Marine Electronics Association) or course bus signals for heading information. These standardized data formats ensure compatibility with a wide range of navigation systems, allowing for seamless integration with existing maritime technologies and equipment. Additionally, the gyro compass may offer status signals, alerts, and error messages through optional output interfaces like NMEA superfast or step signals, providing comprehensive information for monitoring and troubleshooting purposes.

. A sample image of a standard 22 gyro sensor is shown in figure 5.3.

Figure 5. 3 Standard 22 Gyro sensor

Source : https://i.ebayimg.com/images/g/v64AAOSwUuBjwA-6/s-l1200.jpg

5.2.3 GPS sensors

The Global Positioning System (GPS) is an equipment used to locate the precise location of a ship. The NEO-6M GPS Module is a widely-used and reliable Global Positioning System (GPS) module known for its high performance and accuracy in providing location information. With its compact size, low power consumption, and fast time-to-first-fix capabilities, the NEO-6M is a popular choice for a wide range of navigation and tracking applications, including marine, automotive, and outdoor

recreational activities. The module supports multiple satellite positioning systems, such as GPS, GLONASS, and BeiDou, ensuring robust and precise location data in various environments.

The following figure 5.4 shows the existing GPS where the design is implemented.

Such a GPS module will give the output in NMEA signal formats. Following set of NMEAS signals can be obtained from a GPS through a RS 485 interface:

```
$GPBOD - Bearing, origin to destination
$GPBWC - Bearing and distance to waypoint,great circle
$GPGGA - Global Positioning System Fix Data
$GPGLL - Geographic position, latitude / longitude
$GPGSA - GPS DOP and active satellites
$GPGSV - GPS Satellites in view
$GPHDT - Heading, True
$GPR00 - List of waypoints in currently active route
$GPRMA - Recommended minimum specific Loran-C data
$GPRMB - Recommended minimum navigation info
$GPRMC - Recommended minimum specific GPS/Transit data
$GPRTE - Routes
$GPTRF - Transit Fix Data
$GPSTN - Multiple Data ID
$GPVBW - Dual Ground / Water Speed
$GPVTG - Track made good and ground speed
$GPWPL - Waypoint location
$GPXTE - Cross-track error, Measured
$GPZDA - Date & Time
```

Figure 5. 4 GPS sensor

Source : https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2017/10/NEO-GPS-1.jpg?resize=670%2C526&quality=100&strip=all&ssl=1

Out of the above NMEA formats, only "$GPHDT - Heading, True" format is used in this study since our focus is to grab the heading of the ship in our application. The selected sentence is in the following format.

`$GPHDT,054.7,T,034.4,M,005.5,N,010.2,` where;

a. first numeric denotes the bearing of the ship with respect to True North

b. second numeric denotes the bearing of the ship with respect to Magnetic North

c. third value represents the speed of the ship in knots and

d. fourth value represents the speed of the ship in km/hour

5.3 Slave station – Arduino due microprocessor

The design consists of several sensor inputs which are communicated in different communication formats and two microprocessors have been utilized to collect the data. Data format of each sensor are listed in following Table 5.1.

Table 5. 1 Communication Formats of Sensor Inputs

Sensor Input	Communication protocols
GPS sensors	RS 485 and NMEA
Heading sensors	NMEA
Standard 22 Gyro	NMEA

Hence, a one microcontroller is required to receive all gps sensor data from three gps sensors and Arduino due microprocessor. The Arduino Due stands out as the first board in the Arduino family to utilize a 32-bit ARM Cortex-M3 processor. This powerful chip grants it increased processing speed and memory compared to its 8-bit counterparts. Beyond the core upgrade, the Due boasts an impressive serial port capability. It features four built-in UARTs, which are essentially hardware serial ports. These ports enable serial communication with three gps sensors and the other Arduino mega microprocessor, allowing in this project to send and receive data between master and slave stations. A detailed specification of the microprocessor is provided in Appendix 'A'. It will then extract the required data from the three gps inputs and pre-process them to form a data array to transmit to master microprocessor via RS 485

serial protocol. The format of the transmitting data from slave microprocessor is shown in figure 5.5.

Figure 5. 5 Output data stack from Slave microprocessor

Thus created data array is sent to a microprocessor via serial communication to be processed master microprocessor to create another data array. An sample image of the used microcontroller is given in figure 5.6.

Figure 5. 6 Arduino Due microprocessor

Source : https://hetpro-store.com/images/detailed/17/DUE-ouno.jpg

5.4 Master Station – Arduino mega microprocessor

The data sending from the Arduino due microprocessor as a data stack will be docoded by the Arduino mega microprocessor which is served as a Master station in this project. Functional block diagram of the process of data communication is depicted in Figure 5.8.

A Arduino Mega Microprocessor, illustrated in figure 5.7 is used as the Master microprocessor. Detailed specification of the microprocessor is given in Appendix

‘B’. It receives the data array transmitted by the slave microcontroller in serial form. Each data element is extracted separately from the data array and fed as as features/inputs to the ML bases Satellite Compass system. Second data stack will be passed to the Arduino mega and that will be fed to the computer. Serial data received from master unit will be fed to the ML models to generate prediction as indicated in following block diagram

Figure 5. 7 Arduino Mega Microprocessor

Source : https://robolabs.lk/wp-content/uploads/2023/08/Arduino-Mega-R3-robolabs.lk-1.jpg

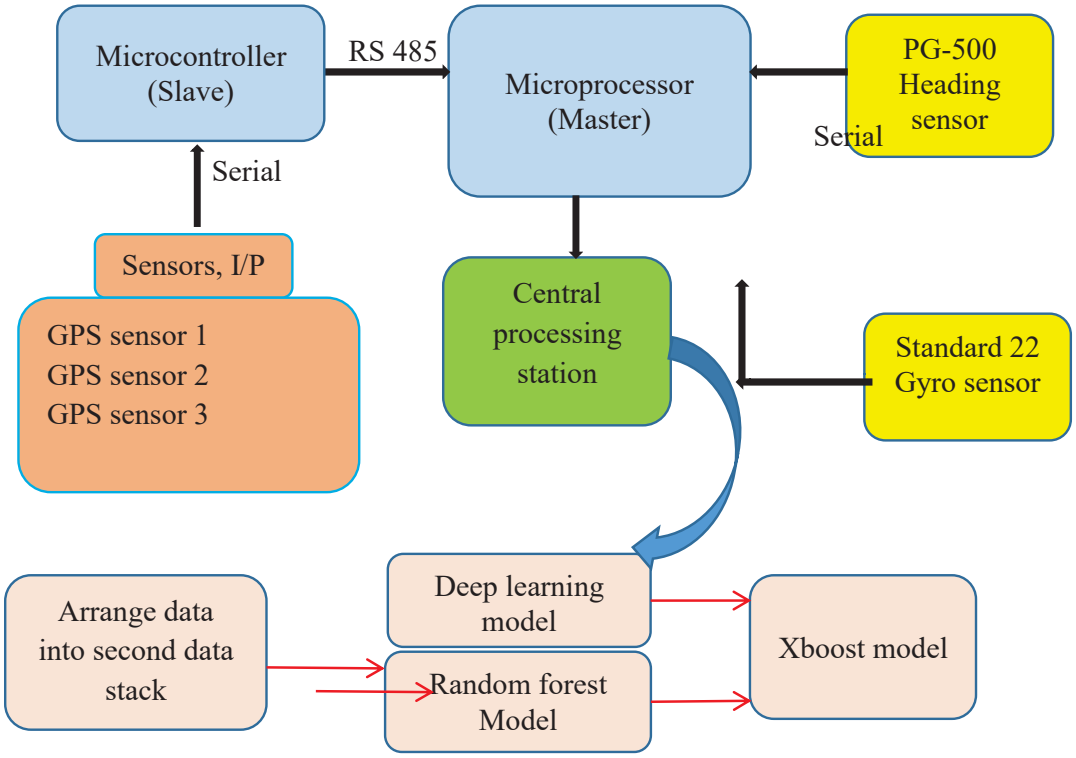

Figure 5. 8 Functional Block Diagram

5.5 Display

The GUI is displayed on the computer display as end user to operate the craft in user friendly manner. The end user will get the heading as visually rotating repeater front end as shown in figure 5.9.

Figure 5. 9 GUI of the ML based Satellite Compass

5.6 PCB

The design includes interfacing of sensors, processing of microcontroller and microprocessor and provision of power to the embedded system. Hence, a PCB is designed to cater the above requirements. The PCB consists of a power circuit (Buck Converter), sensor interfacing terminals (RS 485 Module) and signal processing unit (Arduino mega Microprocessor) as illustrated in the following figure 5.10.

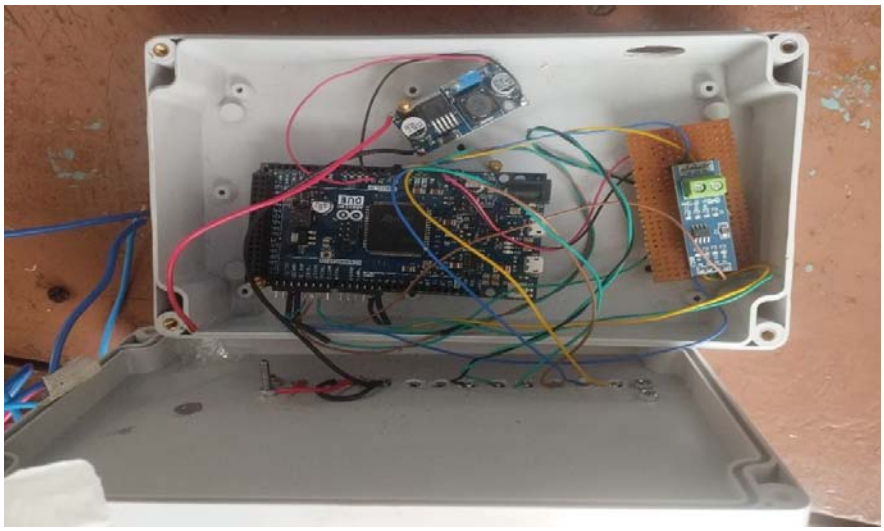

Figure 5. 10 Designed Master PCB

The main power supply to the PCB is 24V DC. 5V DC required for the functioning of the microcontroller, microprocessor, display and the logic levels is stepped down using a buck converter module. The PCB arrangement is illustrated in Appendix 'C'.

The appended data array is shown in figure 5.13, as in the order described in section 5.3.

```
$GPVTG,44.7,T,44.7,M,0.0,N,0.0,K*4E
0.0,-31,646,23,0,7,18,6,2,11,21,30,40,51,60,
$GPVTG,63.2,T,63.2,M,0.0,N,0.0,K*4E
0.0,-31,670,23,0,7,18,6,2,11,21,30,40,51,60,
$GPVTG,57.8,T,57.8,M,0.0,N,0.0,K*4E
0.0,-31,723,23,0,7,18,6,2,11,21,30,40,51,60,
$GPVTG,48.7,T,48.7,M,0.0,N,0.0,K*4E
0.0,-32,705,23,0,7,18,6,2,11,20,30,40,51,60,
$GPVTG,48.7,T,48.7,M,0.0,N,0.0,K*4E
0.0,-31,737,23,0,7,18,6,2,11,20,30,40,51,60,
$GPVTG,48.7,T,48.7,M,0.0,N,0.0,K*4E
0.0,-33,729,24,0,7,18,6,2,11,20,30,40,51,60,
$GPVTG,63.2,T,63.2,M,0.0,N,0.0,K*4E
0.0,-32,555,24,0,7,18,6,2,11,20,30,40,51,60,
$GPVTG,63.2,T,63.2,M,0.0,N,0.0,K*4E
0.0,-31,734,24,0,7,18,6,2,11,21,30,40,51,60,
$GPVTG,26.3,T,26.3,M,0.0,N,0.0,K*4E
0.0,-31,785,24,0,7,18,5,2,11,21,30,40,51,60,
$GPVTG,18.2,T,18.2,M,0.0,N,0.0,K*4E
0.0,-31,660,23,0,7,18,6,2,11,21,30,40,51,60,
$GPVTG,44.7,T,44.7,M,0.0,N,0.0,K*4E
0.0,-33,725,24,0,7,18,6,2,10,20,30,40,51,60,
$GPVTG,44.7,T,44.7,M,0.0,N,0.0,K*4E
0.0,-32,690,24,0,7,18,6,2,10,20,30,40,51,60,
$GPVTG,41.0,T,41.0,M,0.0,N,0.0,K*4E
0.0,-33,681,23,0,7,18,6,2,10,20,30,40,51,60,
$GPVTG,48.4,T,48.4,M,0.0,N,0.0,K*4E
0.0,-31,679,23,0,7,18,6,2,11,21,30,40,51,60,
$GPVTG,53.6,T,53.6,M,0.0,N,0.0,K*4E
0.0,-32,674,24,0,7,18,6,2,10,20,30,40,51,60,
$GPVTG,50.7,T,50.7,M,0.0,N,0.0,K*4E
0.0,-31,724,24,0,7,18,6,2,10,20,30,40,51,60,
$GPVTG,48.7,T,48.7,M,0.0,N,0.0,K*4E
0.0,-32,722,24,0,7,18,6,2,10,20,30,40,51,60,
$GPVTG,0.0,T,0.0,M,0.0,N,0.0,K*4E
0.0,-31,706,24,0,7,18,6,2,10,20,30,40,51,60,
$GPVTG,0.0,T,0.0,M,0.0,N,0.0,K*4E
0.0,-31,687,23,0,7,18,6,2,10,20,30,40,51,60,
```

Figure 5. 11 Appended Data Array

Chapter 6 will discuss the final arrangement of the designed ML based Satellite Compass system pilot project with in the laboratory.

CHAPTER 6

IMPLEMENTATION OF THE ML BASED SATCOMPASS SYSTEM

6.1 Introduction

The successful implementation of the ML-based Satellite Compass System represents a significant advancement in marine navigation technology. In the chapter 5 of this thesis it has been properly explained each of the sensor types and embedded system types used in throughout project. This chapter details the comprehensive process of integrating sensors, collecting data, preprocessing, and training machine learning models to develop a robust and accurate satellite compass system. By leveraging a combination of cutting-edge technologies such as Arduino microcontrollers, RS485 communication protocol, and advanced machine learning algorithms, the system aims to provide precise orientation determination for maritime applications. This introduction sets the stage for exploring the intricacies of sensor integration, data preprocessing, model training, and visualization, culminating in the realization of a sophisticated satellite compass solution.

6.2 Sensor Integration and Data Acquisition

The integration of sensors and acquisition of data within the Satellite Compass system are critical components that lay the foundation for accurate orientation determination. This section elucidates the setup involving Arduino Due and Arduino Mega for sensor integration and data acquisition, the process of collecting data from multiple sensors, and the transmission protocol utilized for seamless communication.

6.2.1 Integration of Sensors

The hardware setup comprises Arduino Due (Slave) and Arduino Mega microcontroller (Master) units, selected for their versatility and compatibility with sensor modules. Arduino Due serves as the primary controller responsible for reading data from GPS sensors and transmitting it to Arduino Mega via RS485 communication protocol. Arduino Mega, on the other hand, acts as the central hub for data reception,

processing, and integration from multiple sensors including the gyro sensor and heading sensor (PG500).

6.2.2 Data Collection and Transmission via RS485 Protocol

The Satellite Compass system utilizes RS485 protocol for data transmission between Arduino Due and Arduino Mega. This protocol allows for robust, long-distance communication with minimal interference. Multiple GPS sensors, each connected to Arduino Due, transmit location data serially. Additionally, the PG500 heading sensor and Standard 22 gyro sensor are interfaced with Arduino Mega, facilitating the collection of orientation-related data.

6.2.3 Arduino Due Program for GPS Data Transmission

The Arduino Due program is responsible for reading data from GPS sensors and transmitting it to Arduino Mega via RS485 protocol. Upon receiving GPS data, Arduino Due parses the latitude and longitude readings and formats them for transmission. The program ensures efficient utilization of serial communication and adheres to RS485 communication standards for seamless data exchange.

6.2.4 Decoding and Integration of Sensor Data on Arduino Mega

Upon receiving GPS data from Arduino Due, Arduino Mega decodes the incoming information and extracts relevant location coordinates. Simultaneously, it reads data from the PG500 heading sensor and Standard 22 gyro sensor. The collected sensor data is integrated into a single data stack, combining GPS coordinates with orientation readings. This integrated data stack serves as the foundation for subsequent processing and analysis within the Satellite Compass system.

6.3 Data Preprocessing

Data preprocessing plays a pivotal role in preparing raw sensor data for subsequent analysis and modeling within the Satellite Compass system. This section delineates the essential preprocessing steps required to transform raw sensor data into a structured format conducive to machine learning model development and system integration.

6.3.1 Parsing GPS Data:

The first preprocessing step involves parsing GPS data to extract latitude and longitude readings. The GPS data received from the Arduino Due is typically in the form of NMEA (National Marine Electronics Association) sentences, which contain a plethora of information including latitude, longitude, altitude, and satellite fix status. Through meticulous parsing techniques, the latitude and longitude values are extracted from the NMEA sentences, enabling precise localization of the satellite

6.3.2 Decoding Heading Data from the PG500 Sensor

Next, the heading data from the PG500 sensor needs to be decoded. The PG500 sensor typically outputs heading information in the form of NMEA sentences as well, which contain specific identifiers such as $HCHDM, $HCHDG, or $HCHDT. By identifying and parsing these sentences, the heading values can be extracted, providing crucial information about the satellite's orientation in space.

6.3.3. Extracting Gyro Compass Course Data

Similarly, the gyro compass course data needs to be extracted from the sensor readings. The gyro sensor outputs data representing the compass course, which indicates the direction in which the satellite is pointing relative to the Earth's magnetic north. This data is vital for accurately determining the satellite's orientation and navigation.

6.3.4 Cleaning and Formatting the Data

Once the relevant sensor data has been extracted, it undergoes a cleaning and formatting process to ensure consistency and accuracy. This step involves removing any outliers or erroneous readings, handling missing values, and standardizing the data format for uniformity. Additionally, the data may be transformed or scaled to adhere to specific requirements of the machine learning algorithms employed for model development

6.4 Machine learning Model training

In Section 4, we meticulously select a combination of algorithms—LSTM, Random Forest, and XGBoost—for predicting gyro sensor readings, capitalizing on their

distinct strengths. While LSTM excels at capturing temporal patterns and long-term dependencies in sequential data, its limitations with small datasets and non-linear relationships prompt the integration of traditional machine learning models like Random Forest and XGBoost. This amalgamation harnesses LSTM's sequential modeling capabilities alongside the robustness and efficiency of Random Forest and XGBoost in handling complex, non-linear relationships. The training process involves meticulous data splitting into training and validation sets, normalization of input features to ensure uniform scales, and the construction of tailored model architectures for each algorithm. LSTM models comprise stacked LSTM layers followed by fully connected layers, whereas Random Forest and XGBoost models are ensemble-based, utilizing decision trees for prediction. Evaluation metrics such as mean squared error, mean absolute error, R-squared, and median absolute error are diligently employed to assess model performance on both training and validation sets, ensuring the development of a robust ML-based satellite compass system for maritime navigation.

6.5 Visualization of the output

In the context of our project, the Graphical User Interface (GUI) plays a crucial role in visualizing real-time data acquired from sensor readings. The GUI is developed using the Processing programming environment, which offers robust tools for creating interactive visualizations.

The Processing sketch serves as the foundation for our GUI, facilitating the display of data received from sensor sources. Let's delve into its key components:

1. **Serial Communication**: Processing's **Serial** library enables communication with external devices, such as microcontrollers or, in our case, the Python program responsible for data processing. Through the **Serial** interface, the sketch continuously receives data streams, updating the GUI accordingly.
2. **Visualization Elements**: Various graphical elements are integrated into the sketch to provide a comprehensive representation of the data. These include images for the compass and arrow, serving as visual aids to indicate directional information. Additionally, background images enhance the aesthetic appeal of the GUI.

3. **Data Processing**: Upon receiving data from the serial port, the sketch processes it to extract relevant information. For instance, incoming data representing heading values undergoes conversion to a floating-point format, enabling precise visualization of directional data.
4. **Error Handling**: Robust error handling mechanisms are implemented to ensure the reliability of the GUI. In cases where invalid or missing data is detected, appropriate error messages are displayed, maintaining user awareness of potential data discrepancies.

6.5.1 Functionalities and User Experience

The GUI developed in Processing offers an intuitive and visually engaging user experience, empowering users to interact seamlessly with the sensor data. Key functionalities include:

- **Real-time Data Visualization**: The GUI dynamically updates to reflect changes in sensor data, providing users with immediate feedback on the system's status.
- **Interactive Elements**: Users can interact with the GUI elements, such as toggling between different display modes, enhancing user engagement and customization options.
- **Error Notification**: Clear and concise error messages alert users to any anomalies in the data stream, fostering transparency and trust in the system.

Through thoughtful design and implementation, the GUI elevates the overall usability and effectiveness of our system, enabling users to gain insights from sensor data with ease and efficiency.

6.6 Conclusion

In conclusion, the implementation of the ML-based Satellite Compass System signifies a significant milestone in the realm of marine navigation. Through meticulous sensor integration, data acquisition, preprocessing, and machine learning model training, we have developed a system capable of accurately determining orientation using satellite-based data. By harnessing the power of technologies such as Arduino microcontrollers, RS485 communication protocol, and ensemble machine learning algorithms, the

system offers enhanced reliability and precision compared to traditional gyrocompass systems. Moving forward, the continuous refinement and optimization of the system will further improve its performance and usability, contributing to safer and more efficient maritime navigation worldwide

CHAPTER 7

PERFORMANCE ASSESSMENT OF ML-BASED SATELLITE COMPASS SYSTEM

7.1 Introduction

In the above chapter 6, we discussed the implementation of ML based Satellite Compass system. This chapter presents the results obtained from training and evaluating machine learning models for predicting gyro readings from heading data. Three models were investigated: a Long Short-Term Memory (LSTM) model, a Random Forest model, and a final XGBoost model that combined predictions from the first two models as additional features.

7.2 Model Testing and Evaluation

7.2.1 LSTM Model

The LSTM model achieved a Mean Squared Error (MSE) of 1331.94 and a Mean Absolute Error (MAE) of 13.72 on the validation set. The R-squared value was 0.898, indicating that the model explained nearly 90% of the variance in the actual gyro readings. These results suggest a good fit between the LSTM model and the data, but there is still room for improvement in terms of prediction accuracy. Actual gyros prediction shows still less performance of the model as indicated in Figure 6.1

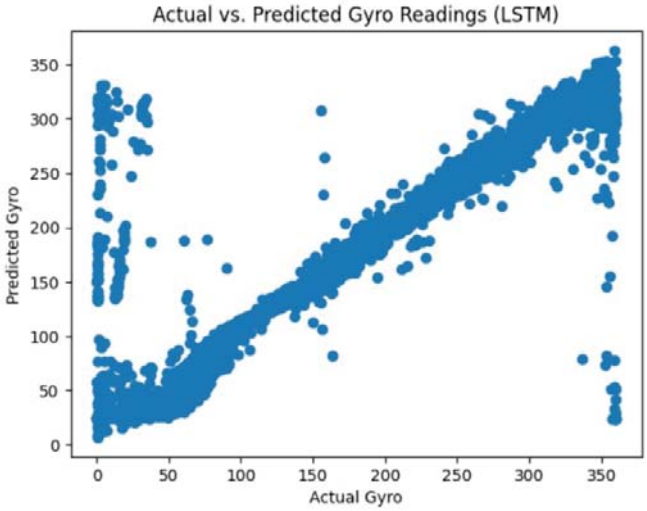

Figure 6. 1 Actual vs predicted heading scatter plot

7.2.2 Random forest Model

The Random Forest model outperformed the LSTM model on all evaluation metrics. It achieved a significantly lower MSE (203.88) and MAE (4.22) compared to the LSTM model. The R-squared value was even higher for the Random Forest model (0.984), indicating an even stronger fit between the model and the data. This suggests that the Random Forest model was able to capture the underlying relationships in the data more effectively than the LSTM model for this specific task and still predicted gyro plot against actual value has indicated more room for improvement as depicted in Figure 6.2.

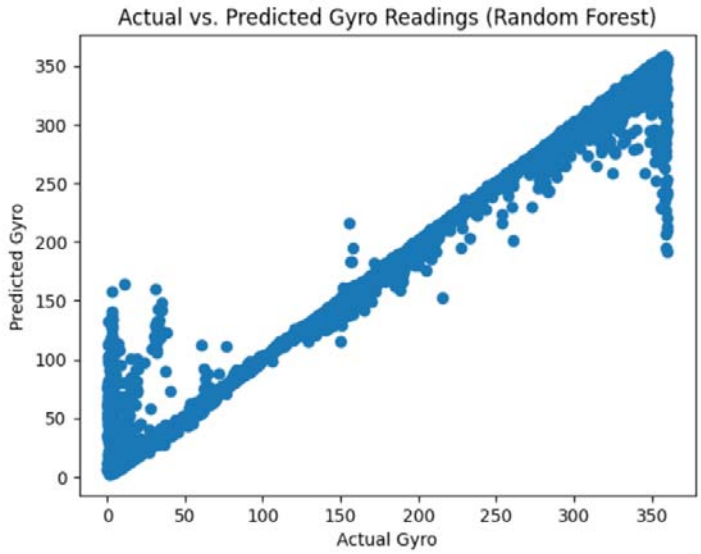

Figure 6. 2 Actual vs Predicted heading scatter plot for random forest model

7.2.3 XGBoost Model

The XGBoost model, which combined predictions from the LSTM and Random Forest models as additional features, achieved the best performance among all models. It had the lowest MSE (12.92) and MAE (1.81) by a significant margin, indicating substantially lower average prediction errors. The R-squared value was exceptionally high (0.999), suggesting the model explained an extremely high proportion of the variance in the actual gyro readings. These results suggest that combining the strengths of both the LSTM and Random Forest models through XGBoost led to a more robust and accurate prediction of gyro readings and model seems to be not over fitted as per

the Figure 6.4 of test and validation loss curves. Also Predicted heading values with respect to actual gyro values shows good improvement as indicated in Figure 6.3.

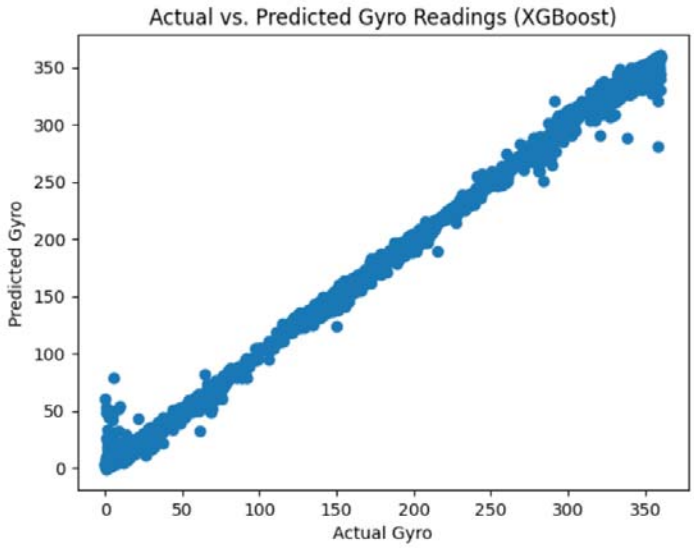

Figure 6. 3 Actual vs Predicted heading scatterplot of XGBoost model

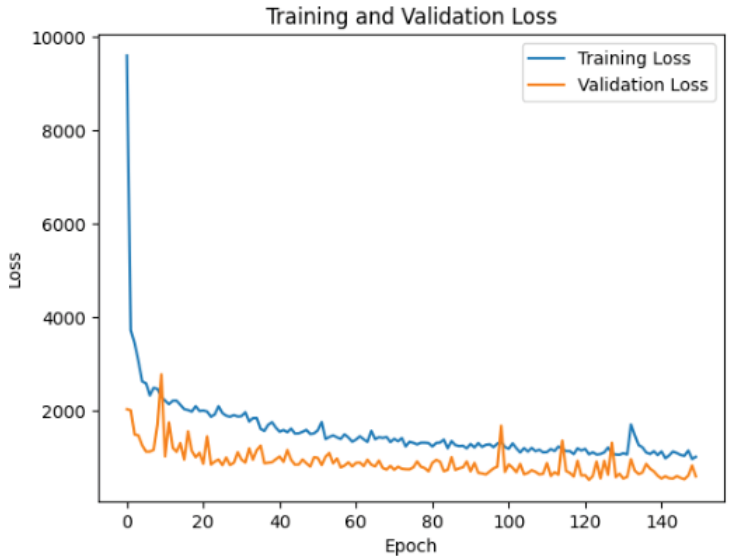

Figure 6. 4 Training and Validation Loss curves of XGBoost model

7.3 Feature importance analysis

The plot of SHAP values represents the feature importance plot for the XGBoost model I trained. SHAP (SHapley Additive exPlanations) values serve as a powerful tool for interpreting the output of machine learning models, offering insights into the contribution of each feature to the final prediction. This plot helps visualize the relative impact of each feature on the model's predictions as indicated in Figure 6.5.

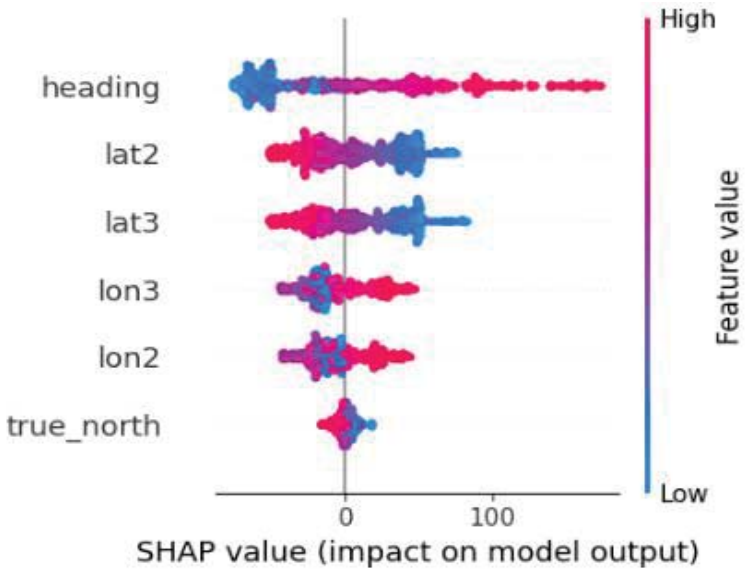

Figure 6. 5 Shap Value plot for feature importance analysis

These values provide a comprehensive framework for understanding feature importance, elucidating how individual features influence the model's output. In the plot, each feature is represented along the y-axis, including variables such as "heading," "lat2," "lat3," "lon3," "lon2," and "true_north." The x-axis showcases the corresponding SHAP values associated with each feature, where positive values signify a positive contribution to the prediction, while negative values denote a negative impact. The length of each vertical bar illustrates the magnitude of the feature's influence on the model's prediction, thereby facilitating interpretation.

Through the interpretation of SHAP values, a deeper understanding of feature importance emerges, offering valuable insights into the model's decision-making

process. For instance, a positive SHAP value for the "heading" feature suggests that higher heading values tend to augment the model's prediction. Conversely, a negative SHAP value for "lat2" implies that lower lat2 values adversely affect the prediction. Overall, the analysis of SHAP values enables researchers to discern the relative significance of different features, potentially identifying avenues for enhancement or optimization within the model architecture

7.4 Discussion

The findings of this study demonstrate the effectiveness of machine learning models in predicting gyro readings from heading data. The XGBoost model, by leveraging the strengths of both the LSTM and Random Forest models, achieved superior prediction accuracy compared to the individual models. This suggests that ensemble learning techniques can be valuable in this domain.

7.4.1 Strengths of the XGBoost Model

- **Lower Prediction Errors:** The XGBoost model achieved significantly lower MSE and MAE compared to the other models, indicating a more accurate prediction of gyro readings.
- **Higher R-squared Value:** The high R-squared value suggests that the XGBoost model captured the underlying relationships in the data very effectively, explaining a large portion of the variance in the actual gyro readings.
- **Leveraging Multiple Feature Sources:** By incorporating features derived from both the LSTM and Random Forest models, the XGBoost model potentially benefited from the strengths of each model, leading to improved prediction accuracy.

7.4.2 Limitations and Considerations

- **Overfitting:** The exceptionally high R-squared value for the XGBoost model might be a sign of little overfitting, especially since the training and testing metrics were almost identical. Further validation on unseen data is crucial to confirm the model's generalizability.

- **Feature Engineering:** Exploring additional feature engineering techniques could potentially improve the performance of all models such as obtaining the More precise GPS sensors and use of other kind of low cost rate gyros as features
- **Data Availability:** The performance of machine learning models is often dependent on the quality and quantity of data. Training on a larger dataset could potentially lead to further improvements in prediction accuracy.

7.5 Future Work

The project lays the foundation for future research and development in ML-based navigation systems. Potential avenues for improvement include:

1. Integration of additional sensor data (e.g., accelerometer, magnetometer) to enhance model performance.
2. Optimization of model architectures and hyper parameters to improve prediction accuracy.
3. Deployment and testing of the models in real-world maritime scenarios to validate their effectiveness.
4. Continuous monitoring and updating of the models to adapt to changing environmental conditions and navigation requirements.

By addressing these limitations and exploring further research avenues, this study contributes to the development of more accurate and reliable methods for predicting gyro readings, potentially enhancing applications in various fields

7.6 Conclusion

In conclusion, the testing and evaluation of the ML-based Satellite Compass system showcased promising results in predicting gyro readings from heading data. Three models, including LSTM, Random Forest, and XGBoost, were scrutinized, with the XGBoost model emerging as the most effective. The results for each machine learning model—LSTM, Random Forest, and XGBoost—highlight their respective strengths and weaknesses in predicting gyro readings. The LSTM model, with a training MSE of 307.75 and a validation MSE of 596.64, demonstrates its ability to capture temporal dependencies in the data, as evidenced by its high R^2 values of 0.97 for training and

0.95 for validation. However, the increase in MSE and MAE during validation suggests that the model may be little overfitting, struggling to generalize well to unseen data. The higher validation errors indicate that while LSTM is effective, its performance could be improved with further tuning or by addressing its limitations in handling small datasets and complex non-linear relationships.

In contrast, the Random Forest model exhibits strong performance across all metrics, particularly excelling in training with an MSE of 46.81 and an R^2 of 0.99. Its validation metrics (MSE of 334.88 and MAE of 3.62) are significantly better than those of the LSTM, indicating better generalization capabilities. The Random Forest model is robust and efficient at capturing non-linear relationships within the data, making it a reliable choice for predicting gyro readings. However, the disparity between training and validation metrics suggests some overfitting, though it remains manageable.

The XGBoost model stands out as the best performer among the three, achieving remarkably low training errors (MSE of 1.79, MAE of 0.69) and maintaining competitive validation metrics (MSE of 334.88, MAE of 3.62, R^2 of 0.97). By combining predictions from both LSTM and Random Forest models, XGBoost leverages their strengths to deliver highly accurate and robust predictions. This ensemble approach proves beneficial, making XGBoost ideal for real-time heading prediction applications. Its ability to handle both temporal dependencies and non-linear relationships effectively positions it as a superior model for the SAT Compass system, ensuring precise and reliable navigation data

Despite its strengths, the XGBoost model's high R-squared value raises concerns of potential overfitting, warranting further validation on unseen data to ensure generalizability. Additionally, avenues for future work include exploring feature engineering techniques, optimizing model architectures, and deploying models in real-world maritime scenarios for validation and adaptation. Through addressing these considerations, this study contributes to advancing the development of accurate and reliable ML-based navigation systems, with implications for diverse applications in navigation and beyond.

CHAPTER 8

CONCLUSION AND FUTURE WORK

In this thesis, we embarked on a comprehensive exploration of the design, implementation, and evaluation of an ML-based Satellite Compass System tailored for marine navigation. The journey began with a thorough examination of the foundational concepts and principles underlying satellite navigation and communication systems in Chapter 1. We delved into the significance of accurate orientation determination for satellites, emphasizing the critical role of sensor integration and data quality in achieving this objective. With meticulous attention to detail, we outlined the hardware and software components essential for the development of a sophisticated Satellite Compass system, highlighting the integration of GPS sensors, heading sensors, and gyro sensors as pivotal elements in Chapter 2. Leveraging technologies such as Arduino microcontrollers and the RS485 communication protocol, we laid the groundwork for robust sensor data acquisition and transmission, setting the stage for subsequent model training and evaluation.

Chapters 3 and 4 provided a deep dive into the machine learning aspect of our Satellite Compass system, focusing on model selection, training, and validation. We meticulously compared and evaluated multiple machine learning algorithms, including LSTM, Random Forest, and XGBoost, for predicting gyro readings from heading data. Through rigorous testing and evaluation, we identified the XGBoost model as the most promising candidate, showcasing superior prediction accuracy and robustness compared to individual models. Specifically, the XGBoost model achieved a training MSE of 1.79 and a validation MSE of 334.88, with an MAE of 0.69 for training and 3.62 for validation, and an R^2 of 0.97 for validation. This comprehensive analysis underscored the importance of ensemble learning techniques in harnessing the strengths of diverse algorithms for enhanced performance.

In Chapter 5, we transitioned from theoretical discussions to practical implementation, detailing the integration of sensors, data preprocessing, and model training procedures. Arduino microcontrollers served as the backbone of our hardware setup, facilitating seamless communication between sensors and data processing units. The adoption of the RS485 communication protocol ensured robust, long-distance data transmission,

essential for real-time navigation applications. With a focus on meticulous data preprocessing techniques, we prepared raw sensor data for model training, ensuring consistency, accuracy, and uniformity across datasets.

The culmination of our efforts is encapsulated in Chapter 6, where we presented the implementation and testing of our ML-based Satellite Compass System. Through rigorous testing and evaluation, we validated the efficacy of our system in accurately predicting gyro readings from heading data. The LSTM model achieved a training MSE of 307.75 and a validation MSE of 596.64, with an MAE of 6.33 for training and 7.66 for validation, and an R^2 of 0.95 for validation. In contrast, the Random Forest model excelled with a training MSE of 46.81 and a validation MSE of 334.88, with an MAE of 1.31 for training and 3.62 for validation, and an R^2 of 0.97 for validation. However, the XGBoost model emerged as the standout performer, demonstrating superior prediction accuracy and robustness compared to alternative models. Despite these achievements, we recognize the need for continuous refinement and improvement.

Future work will focus on integrating additional sensor data, optimizing model architectures, deploying the system in real-world maritime scenarios, and exploring advanced feature engineering techniques. Through these endeavors, we aim to contribute to the advancement of marine navigation technology, offering safer, more efficient, and technologically advanced solutions for maritime operations. By addressing these limitations and exploring further research avenues, this study contributes to the development of more accurate and reliable methods for predicting gyro readings, potentially enhancing applications in various fields

REFERENCES

{1] R. Racz, C. Schott, and S. Huber, "Electronic Compass Sensor," Sensors, Proceedings of IEEE, vol. 3, pp. 1446-1449, 2004.

[3] A. Felski, "Exploitative properties of different types of satellite compasses," Annu. Navig., vol. 16, pp. 33–40, 2010.

[4] A. R. Spielvogel and L. L. Whitcomb, "Preliminary Results with a Low-Cost Fiber-Optic Gyrocompass System," in Proceedings of the OCEANS MTS/IEEE Conference, Washington, DC, USA, 19–22 October 2015.

[5] B. R. Johnson et al., "Development of a MEMS Gyroscope for Northfinding Applications," in Proceedings of the IEEE-ION Position Location and Navigation Symposium, Palm Springs, CA, USA, 4–6 May 2010.

[6] I. Basterretxea-Iribar et al., "Towards an Improvement of Magnetic Compass Accuracy and Adjustment," J. Navig., vol. 69, pp. 1325–1340.

[7] A. Felski and A. Nowak, "Practical experiences from the use of the satellite compass," in Proceedings of the ENC Conference, Parthenope University, Naples, Italy, 4–6 May 2009.

[8] S. Gupta, “PyTorch: LSTM Networks for Time-Series Data (Regression Tasks),” *CoderzColumn*,.

[9] “Beginner’s guide to Timeseries Forecasting with LSTMs using TensorFlow and Keras,” *Towards AI*,.

[10] A. Karpathy, J. Johnson, and L. Fei-Fei, “Visualizing and Understanding Recurrent Networks,” *arXiv preprint arXiv:1506.02078*, 2015.

[11] S. S. Patil and S. S. Patil, “A predictive analytics framework for sensor data using time series and CNN-LSTM hybrid,” *Neural Computing and Applications*, vol. 35, no. pp. 1649–1660, 2023.

[12] A. Karpathy, “Exploring the LSTM Neural Network Model for Time Series,” *Towards Data Science*.

[13] J. Brownlee, “Random Forest for Time Series Forecasting,” *MachineLearningMastery.com*, Nov. 2020.

[14] L. Breiman, “Random forests,” *Machine learning*, 2001.

[15] “Proposed random forest for time series,” *INE*,

[16] “Additional references for variations that lead to Bayesian, neural, dynamic, and causal random forests are also provided,” *SpringerLink*.

[17] “Random Forest for Time Series Forecasting,” *Analytics Vidhya*,

[18] Mostafa Ibrahim, “XGBoost & LGBM for Time Series Forecasting: How to,” *365 Data Science Blog*, Apr. 2024.

[19] “Tutorial: Time Series forecasting with XGBoost,” *Kaggle*.

[20] “Literature on applying XGBoost to Time Series Data,” *Stack Exchange*, Apr. 2023.

[21] “How to predict a time series using XGBoost in Python,” *Set Scholars*, Apr. 2020.

[22] “XGBoost for Time-Series Forecasting,” *Analytics Vidhya*, Jan. 2024.

Appendix A

Specifications of the Arduino due Microprocessor

The Arduino Due is an impressive microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. Here are its key specifications:

1. **Microcontroller:** Atmel SAM3X8E ARM Cortex-M3 CPU
2. **Operating Voltage:** 3.3V
3. **Input Voltage (recommended):** 7-12V
4. **Input Voltage (limits):** 6-20V
5. **Digital I/O Pins:** 54 (of which 12 provide PWM output)
6. **Analog Input Pins:** 12 (with a 12-bit ADC resolution)
7. **Analog Output Pins:** 2 (DAC - Digital to Analog Converter)
8. **Total DC Output Current on all I/O lines:** 130mA
9. **DC Current for 3.3V Pin:** 800mA
10. **DC Current for 5V Pin:** Not specified, usually dependent on the connected voltage regulator
11. **Flash Memory:** 512KB (of which 2KB is used by the bootloader)
12. **SRAM:** 96KB
13. **Clock Speed:** 84MHz
14. **EEPROM:** 4KB
15. **Operating Temperature:** -40°C to 85°C

The Due has a wide range of features that make it suiTable for more complex projects requiring high computational power and I/O capabilities. Its 32-bit ARM architecture provides substantial processing power compared to other Arduino boards, making it suiTable for applications such as robotics, industrial control systems, and high-performance data logging.

Appendix B

Specifications of the Arduino Mega Microprocessor

The Arduino Mega is another popular microcontroller board. Here are its specifications:

Microcontroller: ATmega2560

Operating Voltage: 5V

Input Voltage (recommended): 7-12V

Input Voltage (limits): 6-20V

Digital I/O Pins: 54 (of which 15 provide PWM output)

Analog Input Pins: 16

Analog Output Pins: 0 (PWM can be used on digital pins)

Total DC Output Current on all I/O lines: 40mA

DC Current for 3.3V Pin: 50mA

DC Current for 5V Pin: 800mA

Flash Memory: 256KB (of which 8KB is used by the bootloader)

SRAM: 8KB

EEPROM: 4KB

Clock Speed: 16MHz

Operating Temperature: -40°C to 85°C

The Arduino Mega is well-suited for projects requiring a large number of I/O pins and moderate computational power. It's often used in projects such as 3D printers, CNC machines, and other applications where a high number of inputs and outputs are needed. However, compared to the Due, it has a lower processing power due to its 8-bit AVR architecture and lower clock speed.

Appendix C

Arduino Program of the Microcontroller (Slave station)

```
#include <TinyGPSPlus.h>
#define RS485TransmitPin 2
#define RS485ReceivePin 3
#define RS485DEPin 4
#define RS485REPin 5

#define RS485BaudRate 115200  // Increased Baud Rate

TinyGPSPlus gps1, gps2, gps3;

float latitude1, longitude1, latitude2, longitude2, latitude3, longitude3;

void setup() {
 Serial.begin(9600);
  Serial1.begin(9600);
  Serial2.begin(9600);
  Serial3.begin(9600);
  Serial.begin(RS485BaudRate); // Use hardware serial for RS485 and Serial monitor
  pinMode(RS485DEPin, OUTPUT);
  pinMode(RS485REPin, OUTPUT);
  digitalWrite(RS485DEPin, LOW);  // Initialize RS485 transceiver in Receive mode
  digitalWrite(RS485REPin, LOW);  // Initialize RS485 transceiver in Receive mode
}

void loop() {
  while (Serial1.available() > 0) {
    gps1.encode(Serial1.read());
  }
```

```
while (Serial2.available() > 0) {
  gps2.encode(Serial2.read());
}

while (Serial3.available() > 0) {
  gps3.encode(Serial3.read());
}

// Read GPS data from GPS module 1
latitude1 = gps1.location.lat();
longitude1 = gps1.location.lng();

// Read GPS data from GPS module 2
latitude2 = gps2.location.lat();
longitude2 = gps2.location.lng();

// Read GPS data from GPS module 3
latitude3 = gps3.location.lat();
longitude3 = gps3.location.lng();

// Set the RS485 transceiver to Transmit mode
digitalWrite(RS485DEPin, HIGH);
digitalWrite(RS485REPin, HIGH);

// Send the combined GPS data to the Slave via RS485 communication
Serial.print("$");
Serial.print(latitude1, 6);
Serial.print(",");
Serial.print(longitude1, 6);
Serial.print(",");
Serial.print(latitude2, 6);
```

```
  Serial.print(",");
  Serial.print(longitude2, 6);
  Serial.print(",");
  Serial.print(latitude3, 6);
  Serial.print(",");
  Serial.print(longitude3, 6);
  Serial.println("*");


  delay(100);
}
```

Appendix D

Arduino Program of the Microprocessor (Master Station)

```
#define RS485DEPin 4
#define RS485REPin 5
#define SERIAL_BAUDRATE 4800
#define RS485BaudRate 115200

const int maxSentenceLength = 64;
char sentence[maxSentenceLength + 1];
int sentenceIndex = 0;
bool sentenceStarted = false;

void setup() {
 Serial.begin(115200);
 initializeSerial();

 pinMode(RS485DEPin, OUTPUT);
 pinMode(RS485REPin, OUTPUT);
 digitalWrite(RS485DEPin, LOW);
 digitalWrite(RS485REPin, LOW);
}

void initializeSerial() {
 Serial1.begin(RS485BaudRate);
 Serial2.begin(4800);
 Serial3.begin(SERIAL_BAUDRATE);
}

inline void readSerialData (String& data, HardwareSerial& serial) {
 data = "";
 while (serial.available() > 0) {
  char c = serial.read();
  if (c == '\n') {
   data.trim();
   break;
  }
  data += c;
 }
}

bool startsWith(const String& str, const String& prefix) {
 return str.startsWith(prefix);
```

```
}

float customParseFloat(const char* str) {
  char buffer[32];
  int i;
  for (i = 0; i < 31 && str[i] != '\0'; i++) {
    if (str[i] == ',' || str[i] == '\x19') // Replace with the correct unexpected character
      buffer[i] = '.';
    else
      buffer[i] = str[i];
  }
  buffer[i] = '\0';
  return atof(buffer);
}

void processSentence(const char* sentence) {
  // Assuming the format: $lat1,lon1,lat2,lon2,lat3,lon3*

  // Check if the sentence starts with '$' and ends with '*'
  if (sentence[0] == '$' && sentence[sentenceIndex - 1] == '*') {
    char lat1Str[12], lon1Str[12], lat2Str[12], lon2Str[12], lat3Str[12], lon3Str[12];
    int result = sscanf(sentence,
"$%11[^,],%11[^,],%11[^,],%11[^,],%11[^,],%11[^*]*",
                        lat1Str, lon1Str, lat2Str, lon2Str, lat3Str, lon3Str);

    // Check if all 6 values were successfully parsed
    if (result == 6) {
      float lat1 = customParseFloat(lat1Str);
      float lon1 = customParseFloat(lon1Str);
      float lat2 = customParseFloat(lat2Str);
      float lon2 = customParseFloat(lon2Str);
      float lat3 = customParseFloat(lat3Str);
      float lon3 = customParseFloat(lon3Str);

      // Read heading sensor data
      String sentenceH;
      readSerialData(sentenceH, Serial2);

      if (startsWith(sentenceH, "$HCHDM") || startsWith(sentenceH, "$HCHDG") ||
startsWith(sentenceH, "$HCHDT")) {
        char* headingStr = strtok((char*)sentenceH.c_str(), ",");
        headingStr = strtok(NULL, ",");
        float heading = atof(headingStr);
```

```
      // Read gyro sensor data
      String gyroData;
      readSerialData(gyroData, Serial3);

      if (gyroData.startsWith("$HEHDT")) {
        int firstCommaIndex = gyroData.indexOf(',');
        if (firstCommaIndex != -1) {
          String gyroCompassCourse = gyroData.substring(firstCommaIndex + 1);
          float courseValue = gyroCompassCourse.toFloat();

          // Print the sensor data
          Serial.print(lat1, 6);
          Serial.print(", ");
          Serial.print(lon1, 6);
          Serial.print(", ");
          Serial.print(lat2, 6);
          Serial.print(", ");
          Serial.print(lon2, 6);
          Serial.print(", ");
          Serial.print(lat3, 6);
          Serial.print(", ");
          Serial.print(lon3, 6);
          Serial.print(", ");
          Serial.print(heading, 3);
          Serial.print(", ");
          Serial.println(courseValue, 3);
        }
      }
    }
  }
}

void loop() {
  while (Serial1.available() && Serial2.available() && Serial3.available()) {
    char c = Serial1.read();

    if (c == '$') {
      sentenceIndex = 0;
      sentenceStarted = true;
    }

    if (sentenceStarted) {
      if (sentenceIndex < maxSentenceLength) {
```

```
      sentence[sentenceIndex++] = c;
    } else {
      Serial.println("Error: Sentence too long");
      sentenceStarted = false;
      sentenceIndex = 0;
    }

    if (c == '*') {
      sentence[sentenceIndex] = '\0';
      sentenceStarted = false;

      processSentence(sentence);

      sentenceIndex = 0;
    }
  }
}
                                                    }
```

Appendix E

Processing Program of the GUI

```
import processing.serial.*;

Serial myPort;
PImage imgCompass;
PImage imgCompassArrow;
PImage background1;
PImage background2;

String data = "";
float heading = 0;

void setup() {
 size(1920, 1080, P3D);
 smooth();
 imgCompass = loadImage("cmps1.png");
 imgCompassArrow = loadImage("Arw1.png");
 background1 = loadImage("BG1.png");
 background2 = loadImage("black1.png");

 String portName = Serial.list()[0]; // Change the index based on your serial port
 myPort = new Serial(this, portName, 115200);
}

void draw() {
 while (myPort.available() > 0) {
  data = myPort.readStringUntil('\n'); // Read data from the serial port
  if (data != null) {
   heading = float(data); // Convert the received string to float
  }
 }

 if (heading >= 0) {
  // Display your GUI based on the received heading value
  pushMatrix();
  translate(width/2, height/2, 0);
  rotateZ(radians(-heading));
  image(imgCompass, -900, -527.5);
  popMatrix();

  image(imgCompassArrow, 0, -35);
```

```
    textSize(50);
    fill(0, 255, 0);
    text(heading, 880, 40);

    fill(255, 255, 255);
    textSize(14);
    text("MODE", 60, 40);

    textSize(20);
    text("A Design By : Electrical New Design Center (East)", 755, 1025);
  } else {
    // Handle invalid or missing data
    stroke(80);
    strokeWeight(2);
    textSize(12);
    fill(255, 0, 0);
    text("Invalid data received", 75, 150);
  }

  delay(100);
}
```

Appendix F

Python program for Machine Learning Model

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score,
median_absolute_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from xgboost import XGBRegressor
import joblib
from google.colab import drive
import wandb

# Mount Google Drive
drive.mount('/content/drive')

# Initialize Weights & Biases
wandb.login()

# Upload the CSV file
uploaded = files.upload()

# Use the first uploaded file
uploaded_file_name = list(uploaded.keys())[0]
data = pd.read_csv(uploaded_file_name, delimiter=',')

# Assign column names
data.columns = ['lat1', 'lon1', 'lat2', 'lon2', 'lat3', 'lon3', 'heading', 'gyro']

# Data Cleaning: Remove null rows and zero gyro readings
data = data.dropna()
data = data[data['gyro'] != 0]

# Remove rows with gyro or heading sensor values greater than 360
condition = (data['gyro'] > 360) | (data['heading'] > 360)
data = data[~condition]
```

```
# Descriptive Statistics
print(data.describe())

# Keep only the heading value as features
X = data['heading'].values.reshape(-1, 1)
y = data['gyro'].values

# Normalize features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Create sequences of input data
sequence_length = 10  # Number of timesteps in each sequence
X_sequences = []
y_sequences = []

for i in range(len(X) - sequence_length + 1):
    X_sequences.append(X[i:i + sequence_length])
    y_sequences.append(y[i + sequence_length - 1])

X_sequences = np.array(X_sequences)
y_sequences = np.array(y_sequences)

# Split sequence dataset into training and validation sets
X_train_seq, X_val_seq, y_train_seq, y_val_seq = train_test_split(X_sequences,
y_sequences, test_size=0.2, random_state=42)

# Train LSTM model
with tf.device('/GPU:0' if tf.config.list_physical_devices('GPU') else '/CPU:0'):
    lstm_model = Sequential()
    lstm_model.add(LSTM(128, activation='relu', return_sequences=True,
input_shape=(sequence_length, X_train_seq.shape[2])))
    lstm_model.add(Dropout(0.2))
    lstm_model.add(LSTM(64, activation='relu', return_sequences=True))
    lstm_model.add(Dropout(0.2))
    lstm_model.add(LSTM(64, activation='relu'))
    lstm_model.add(Dense(32, activation='relu'))
    lstm_model.add(Dropout(0.2))
    lstm_model.add(Dense(1, activation='linear'))  # Linear activation for regression

    lstm_model.compile(optimizer='adam', loss='mean_squared_error')

    # Train the LSTM model
```

```
    history = lstm_model.fit(X_train_seq, y_train_seq, validation_data=(X_val_seq,
y_val_seq), epochs=150, batch_size=16)

    # Predictions from LSTM model
    y_pred_lstm_train = lstm_model.predict(X_train_seq)
    y_pred_lstm_val = lstm_model.predict(X_val_seq)

# Train Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_seq.reshape(X_train_seq.shape[0], -1), y_train_seq)

# Predictions from Random Forest model
y_pred_rf_train = rf_model.predict(X_train_seq.reshape(X_train_seq.shape[0], -1))
y_pred_rf_val = rf_model.predict(X_val_seq.reshape(X_val_seq.shape[0], -1))

# Combine predictions from LSTM and Random Forest as features
X_train_combined = np.column_stack((y_pred_lstm_train, y_pred_rf_train))
X_val_combined = np.column_stack((y_pred_lstm_val, y_pred_rf_val))

# Train XGBoost model on combined features
xgb_model = XGBRegressor()
xgb_model.fit(X_train_combined, y_train_seq)

# Predictions from XGBoost model
y_pred_xgb_train = xgb_model.predict(X_train_combined)
y_pred_xgb_val = xgb_model.predict(X_val_combined)

# Save the pre-trained models to Google Drive
model_filenames = ['/content/drive/My Drive/lstm_gyro_model.h5',
                   '/content/drive/My Drive/random_forest_gyro_model.pkl',
                   '/content/drive/My Drive/xgboost_gyro_model.pkl']

# Save LSTM model
lstm_model.save(model_filenames[0])
print("LSTM model saved successfully to Google Drive.")

# Save Random Forest model
joblib.dump(rf_model, model_filenames[1])
print("Random Forest model saved successfully to Google Drive.")

# Save XGBoost model
joblib.dump(xgb_model, model_filenames[2])
print("XGBoost model saved successfully to Google Drive.")
```

```
# Plot training loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')
plt.show()

def evaluate_model(model_name, y_train, y_pred_train, y_val, y_pred_val):
    print("Evaluation for", model_name)
    print("Training Metrics:")
    print("Mean Squared Error (MSE):", mean_squared_error(y_train, y_pred_train))
    print("Mean Absolute Error (MAE):", mean_absolute_error(y_train,
y_pred_train))
    print("R-squared (R2):", r2_score(y_train, y_pred_train))
    print("Median Absolute Error:", median_absolute_error(y_train, y_pred_train))
    print("\nValidation Metrics:")
    print("Mean Squared Error (MSE):", mean_squared_error(y_val, y_pred_val))
    print("Mean Absolute Error (MAE):", mean_absolute_error(y_val, y_pred_val))
    print("R-squared (R2):", r2_score(y_val, y_pred_val))
    print("Median Absolute Error:", median_absolute_error(y_val, y_pred_val))
    print("-----------------------------------------------")

# Evaluate the models
evaluate_model("LSTM", y_train_seq, y_pred_lstm_train, y_val_seq,
y_pred_lstm_val)
evaluate_model("Random Forest", y_train_seq, y_pred_rf_train, y_val_seq,
y_pred_rf_val)
evaluate_model("XGBoost", y_train_seq, y_pred_xgb_train, y_val_seq,
y_pred_xgb_val)

# Visualize actual vs. predicted gyro readings for each model
plt.scatter(y_val_seq, y_pred_lstm_val)
plt.xlabel('Actual Gyro')
plt.ylabel('Predicted Gyro')
plt.title('Actual vs. Predicted Gyro Readings (LSTM)')
plt.show()

plt.scatter(y_val_seq, y_pred_rf_val)
plt.xlabel('Actual Gyro')
plt.ylabel('Predicted Gyro')
plt.title('Actual vs. Predicted Gyro Readings (Random Forest)')
plt.show()
```

```
plt.scatter(y_val_seq, y_pred_xgb_val)
plt.xlabel('Actual Gyro')
plt.ylabel('Predicted Gyro')
plt.title('Actual vs. Predicted Gyro Readings (XGBoost)')
plt.show()
```