DESIGN AND TESTING OF AN ATTITUDE AND HEADING REFERENCE SYSTEM

İBRAHİM AHMET ALTAŞ

AUGUST 2024

ÇANKAYA UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF MECHATRONICS ENGINEERING

M.Sc. Thesis in

MECHATRONICS ENGINEERING

DESIGN AND TESTING OF AN ATTITUDE AND HEADING REFERENCE SYSTEM

İBRAHİM AHMET ALTAŞ

AUGUST 2024

ABSTRACT

DESIGN AND TESTING OF AN ATTITUDE AND HEADING REFERENCE SYSTEM

ALTAŞ, İbrahim Ahmet
M.Sc. in Mechatronics Engineering

Supervisor: Assoc. Prof. Dr. Halit ERGEZER
August 2024, 131 pages

AHRS (Attitude and Heading Reference System) is a system used to determine the orientation and position of a vehicle or object in space. It typically works by combining data from three-axis gyroscopes, accelerometers, and magnetometers. Gyroscopes measure the angular velocity of the vehicle, while accelerometers detect linear motion and the direction of gravity. Magnetometers, on the other hand, identify the direction of magnetic north. By integrating these sensor readings, the system can calculate the pitch, roll, and yaw angles, as well as the heading direction of the vehicle.

AHRS is widely used in various fields such as aircraft, UAVs (Unmanned Aerial Vehicles), marine vessels, autonomous systems, and virtual reality applications. In aircraft, AHRS is a part of flight control systems and provides pilots with crucial information about the vehicle's orientation in space. UAVs and autonomous vehicles rely on AHRS for stable and safe navigation and control. In marine vessels, AHRS enhances navigation safety by tracking the vessel's heading and tilt. Additionally, in virtual and augmented reality systems, AHRS allows for more natural and interactive experiences by tracking users' head movements. As an essential component of modern technology, AHRS plays a critical role in ensuring accurate and reliable orientation detection across different industries.

Keywords: Attitude and Heading Reference Systems, Unmanned Aerial Vehicles, MEMS Sensors, Sensor Fusion, Sensor Calibration.

ÖZET

DURUM VE İSTİKAMET REFERANS SİSTEMİNİN TASARIMI VE TEST EDİLMESİ

ALTAŞ, İbrahim Ahmet
Mekatronik Mühendisliği Yüksek Lisans

Danışman: Doç. Dr. Halit ERGEZER
Ağustos 2024, 131 sayfa

Durum ve istikamet referans sistemi, bir aracın veya cismin uzaydaki konumunu ve yönelimini belirleyen bir sistemdir. Genellikle, üç eksenli jiroskop, ivmeölçer ve manyetometre sensörlerini birleştirerek çalışır. Jiroskoplar aracın açısal hızını ölçerken, ivmeölçerler aracın doğrusal hareketini ve yerçekimi yönünü algılar. Manyetometreler ise manyetik kuzey yönünü tespit eder. Bu sensör verileri birleştirilerek, aracın eğim (pitch), yalpa (roll) ve sapma (yaw) açılarının yanı sıra baş yönü (heading) hesaplanır.

Durum ve istikamet referans sistemi, uçaklar, İHA'lar (İnsansız Hava Araçları), deniz araçları, otonom sistemler ve sanal gerçeklik uygulamaları gibi çeşitli alanlarda yaygın olarak kullanılmaktadır. Uçaklarda, Durum ve istikamet referans sistemi, uçuş kontrol sistemlerinin bir parçası olarak kullanılır ve pilotlara aracın uzaydaki konumuyla ilgili hayati bilgiler sağlar. İHA'lar ve otonom araçlar, navigasyon ve kontrol süreçlerinde Durum ve istikamet referans sistemine dayanarak stabil ve güvenli hareket sağlarlar. Deniz araçlarında ise, AHRS, geminin yönünü ve eğimini takip ederek seyir güvenliğini artırır. Ayrıca, sanal gerçeklik ve artırılmış gerçeklik sistemlerinde, kullanıcıların baş hareketlerini izleyerek daha doğal ve etkileşimli bir deneyim sunulmasına olanak tanır. Durum ve istikamet referans sistemi, modern teknolojinin önemli bir bileşeni olarak, farklı sektörlerde güvenli ve doğru yönelim tespiti için kritik bir rol oynamaktadır.

Anahtar Kelimeler: Açısal Konum ve Referans Sistemi, İnsansız Hava Araçları, MEMS Sensörler, Sensör Füzyonu, Sensör Kalibrasyonu

ACKNOWLEDGEMENT

First, I would like to thank Assoc. Prof. Dr. Halit ERGEZER, thank you for your supervision, guidance, suggestions, and encouragement throughout this research, contributions to my business life, the engineering perspective you taught me, and suggestions and help. I am grateful for your mentorship and what you have contributed to me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOLS

Roll Angle	: ϕ
Pitch Angle	: θ
Yaw Angle	: ψ

ABBREVIATIONS

UAV	: Unmanned aerial vehicles
AGV	: Autonomous Ground Vehicle
MUH	: Miniature Unmanned Helicopter
AHRS	: Attitude and Heading Reference System
IMU	: Inertial Measurement Unit
TRIAD	: Triaxial Attitude Determination
KF	: Kalman Filter
EKF	: Extended Kalman Filter
UKF	: Unscented Kalman Filter
DOF	: Degree of Freedom
RLG	: Ring Laser Gyroscopes
FOG	: Fiber Optics Gyroscopes
DTG	: Dynamically Tuned Gyroscopes
SQUID	: Superconducting Quantum Interference Device
MR	: Magneto-Resistive
ECEF	: Earth-Centered Earth-Fixed
NED	: North-East-Down
NEU	: North-East-Up
DCM	: Direction Cosine Matrix
DCT	: Direction Cosine Transformation
RMS	: Root Mean Square
RMSE	: Root Mean Square Error

CHAPTER I
INTRODUCTION

1.1 BACKGROUND

An Unmanned Aerial Vehicle (UAV) is defined as a powered aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or nonlethal payload. A UAV, commonly known as a drone, unpiloted aerial vehicle, or Remotely Piloted Aircraft (RPA), has its flight controlled autonomously by on-board computers or by the remote control of a pilot on the ground or in another vehicle [1].

UAVs have become prevalent daily and are essential in numerous applications, such as surveillance, mapping, cargo transport, security, and agriculture. However, for these UAVs to operate successfully, they require precise navigation and control. The accurate navigation attitude of UAVs is one of the parameters imported into the system design. Attitude [2] is an orientation of the UAV in 3D space.

Attitude and Heading Reference System (AHRS) [3] is a navigation system that produces attitude information, which is defined as Euler angles [4]. Euler angles can be listed as roll, pitch, and yaw angles. Sensor fusion is required using data from a 3-axis accelerometer, a 3-axis magnetometer, and a 3-axis gyroscope to obtain these angles. There are multiple steps to effectively achieve sensor fusion. The first step is stochastic error identification and modeling of sensor errors. The second step is linear and nonlinear filtering. Optimal filtering is a real-time capable data fusion that fulfills a mathematical optimality criterion. Rudolf E. Kalman invented a set of equations for the propagation and correction of the filter state vector and corresponding covariance matrix of a dynamic system [5].

Magnetometers are devices used for measuring magnetic fields. The primary purpose of a magnetometer is to detect magnetic variations accurately. For magnetometers, there are six types which are Fluxgate Magnetometers, Optically Pumped Magnetometers, SQUID Magnetometers, Hall Effect Magnetometers, Magneto-Resistive (MR) Magnetometers, and Lorentz Force Magnetometers [8].

Inertial Measurement Unit (IMU) [9] sensors ensure Unmanned Aerial Vehicles' accurate guidance and stable flight. Sensor technology is shrinking and, more importantly, becoming more accessible. In this context, MEMS IMU (Micro-Electromechanical Systems Inertial Measurement Units) sensors play a significant role in various application domains. These tiny devices are employed to measure motion, acceleration, and orientation.

1.2 LITERATURE SURVEY

When the literature was reviewed to obtain attitude information, it was seen that different methods were used. The most preferred methods are attitude estimators: Kalman Filter, Extended Kalman Filter (EKF), and Unscented Kalman Filter (UKF). Also, other methods, such as the Complementary Filter, Madgwick Filter, and Mahony Filter, do not work well as estimators. However, these methods are preferred in low-cost systems.

The attitude heading reference system is based on an unscented Kalman filter using the triaxial attitude determination (TRIAD) algorithm as the observation model designed in [10]. GPS/GNSS data can also be used to improve estimators' output.
The attitude estimation method for Miniature Unmanned Helicopter (MUH) is designed in [11]. The process is based on the Unscented Kalman Filter. Also, they added gyroscope bias to improve their estimators.

A two-step Extended Kalman Filter (EKF) algorithm is used in this study to estimate the orientation of an IMU. A 9-DOF device is used for this purpose, including a 6-DOF IMU with a three-axis gyroscope, a three-axis accelerometer, and a three-axis magnetometer [12]. Also, gyroscope bias should be added to improve filter performance.

The attitude estimation method based on information fusion of Complementary Filters is designed. Combined with analyzing the relation between attitude angular rate and attitude angle in different dynamic modes of UAV, concluding high and low

frequency, the attitude angle is calculated by a complementary filter with the attitude angular rate and acceleration information [13].

A complementary filter method based on multi-sensor fusion was proposed to solve the gyroscope drift problem, one of the biggest problems encountered when obtaining attitude information in unmanned aerial vehicles [14].

1.3 THESIS OBJECTIVE

As mentioned above, attitude estimation on UAVs is critical for autonomous vehicles. One of the most considerable difficulties in attitude estimation is the noise affecting the sensors. For example, high noise in the data received from the accelerometer, deviation of the gyroscope data over time, and environmental magnetic disturbances affecting the magnetometer data cause high deviations in the resulting roll angle, pitch angle and yaw angle values. In addition, high-performance sensor fusion of data from all these sensors is among the difficulties encountered.

In this thesis, as the first stage of the simulation process, 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer sensors were modeled in the AHRS system simulation environment, and the state estimator design was carried out. As the second stage of the simulation process, an ideal rotary wing UAV [15] modeling was carried out in the simulation environment. Then, the attitude information obtained from the designed AHRS model was compared with that obtained from the ideal rotary wing UAV model regarding roll, pitch and yaw angle.

To see how the simulation studies carried out in this thesis and the designed system will react in real life, a printed circuit board containing 3 3-axis accelerometers, a 3-axis gyroscope, and a 3-axis magnetometer was produced as an AHRS prototype. Sensor calibration was applied to each sensor to eliminate errors arising from the nature of these sensors. Soft & hard iron calibration [16] has also been used to minimize environmental magnetic distortions.

A rotary-wing UAV was designed to test the AHRS prototype in a simulation environment. This way, how the constructed AHRS methods react in a real system was examined. In addition, to test the AHRS prototype placed on this rotary wing UAV with higher accuracy, a 3-axis rotational test platform (3-DOF) has been designed with the high-accuracy encoders placed at the center of each axis.

1.4 THESIS OVERVIEW

The organization of the part of the thesis is as follows;

- Chapter 2, the Navigation Background Information section, provides general information about navigation systems.
- Chapter 3, Sensor Fusion Algorithms section, gives information about attitude estimators.
- Chapter 4, Sensor Error Modeling and Sensor Calibration Process section defines sensor errors and handles sensor calibration methods.
- Chapter 5, Attitude Estimator Design in Simulation Environment section, inspects simulations for different attitude estimators.
- Chapter 6, AHRS Experimental System Setup section, mentions how real sensor systems are established.
- Chapter 7, the AHRS Experimental System Test Setup Design section, provides information about how real systems are tested.
- Chapter 8, Performing Real System Test and Analysis of the Test Results section, handles the analysis of test and simulation outputs.
- Chapter 9, Conclusion and Future Works section, provides accomplished Works in this thesis and clarifies the outcomes of this thesis and future works.

CHAPTER II
NAVIGATION BACKGROUND INFORMATION

2.1 NAVIGATION REFERENCE FRAMES

Reference frames are essential in physics, engineering, and various other fields where the motion or orientation of objects needs to be precisely defined and measured. The object's motion must be compared with a recognized standard known as a reference frame to describe an object's position, velocity, and orientation. This reference frame consists of an origin that defines a position in space and three orthogonal unit vectors or axes arranged according to the right-hand rule. These three axes are typically called x, y, and z. Depending on the application type and desired outcomes, various reference frames [17] can be used to measure the motion of an object, including a sensor frame, a body frame, an Earth-Centered, Earth-Fixed (ECEF) frame, or a local North-East-Down (NED) frame. Detailed information about figure given in [45].

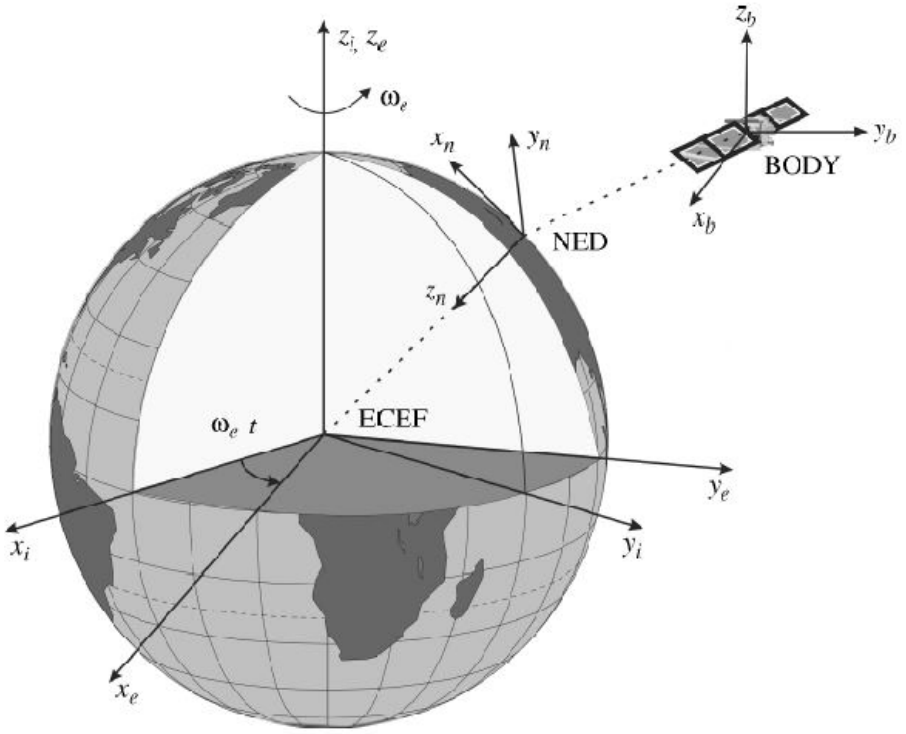


Figure 1: Body, NED and ECEF frames positioned to each other

2.1.1 Sensor Frame

A sensor frame is a reference frame tied to a specific sensor, such as a camera, accelerometer, magnetometer or gyroscope. It is beneficial when measuring an object's motion from the sensor's perspective. The sensor frame's origin and axes are typically defined relative to the sensor's position and orientation. Detailed information about figure given in [46].

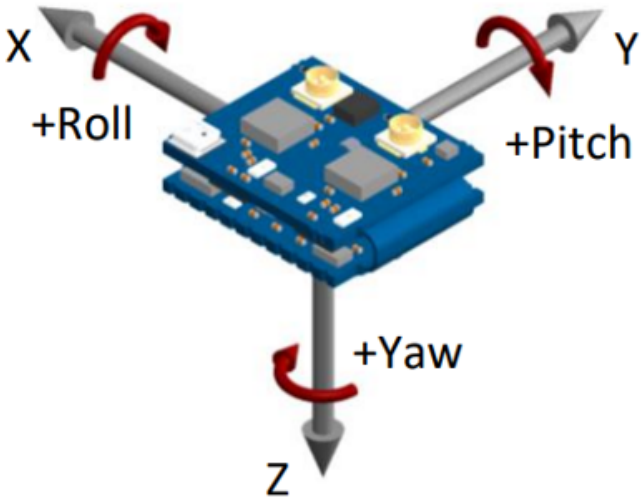


Figure 2: Sensor Frame and Euler Angles

2.1.2 Body Frame

The body frame is typically attached to the object of interest. It moves with the object and allows you to describe the object's motion relative to its own body. The origin is often located at the object's center of mass, and the axes are aligned with the object's principal axes. Detailed information about figure given in [47].

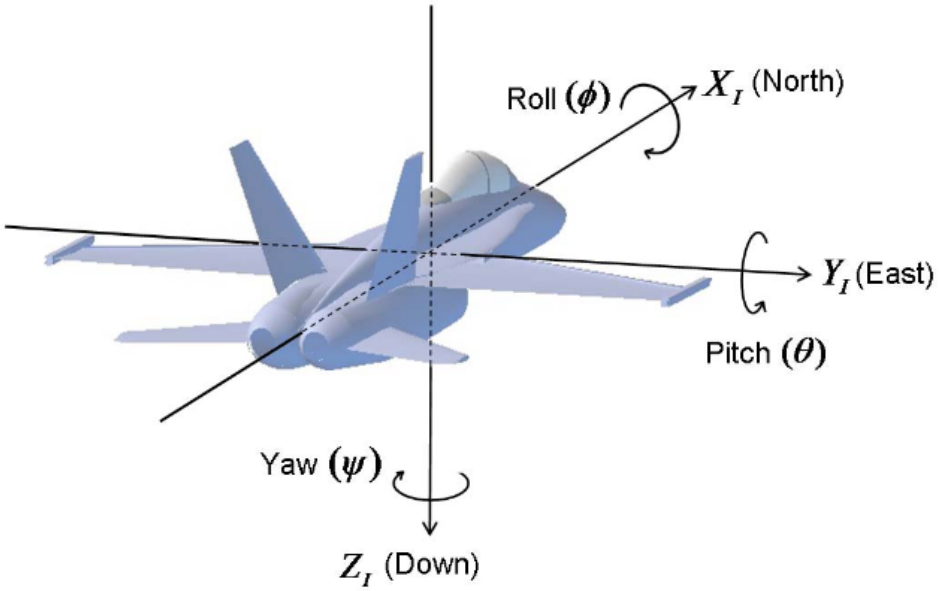


Figure 3: Body Frame and Euler Angles

2.1.3 Earth-Centered, Earth-Fixed (ECEF) Frame

The ECEF frame is a global reference frame used in applications like GPS navigation and satellite tracking. It originates at the center of the Earth, and the x, y, and z axes are fixed relative to the Earth's surface. This frame provides a global perspective for positioning and tracking objects on or above the Earth's surface. Detailed information about figure given in [48].

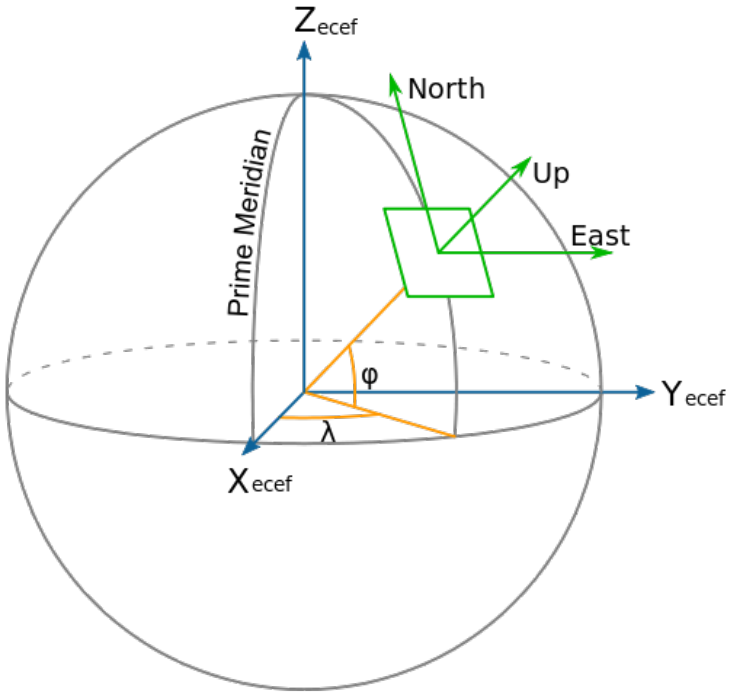


Figure 4: ECEF Frame according to NEU Frame

2.1.4 Local North-East-Down (NED) or Local North-East-Up (NEU) Frame

The NED or NEU frame is a local reference frame often used in aviation, navigation, and geodesy. It is defined with respect to a point on the Earth's surface and is well-suited for describing the motion of objects over relatively short distances. The origin is at the specified point, with the "North," "East," and "Down" axes or the "North," "East," and "Up" axes aligned with the local directions of the Earth's surface. Detailed information about figure given in [49].

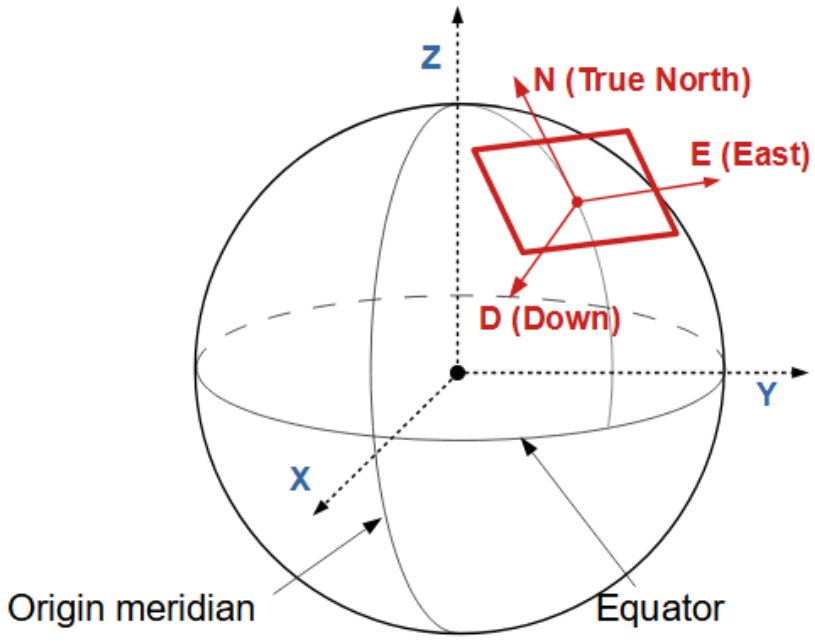


Figure 5: NED Frame according to ECEF Frame

2.1.5 Quaternions

Quaternions serve as a mathematical tool to precisely define and portray the orientation or rotation of an object within three-dimensional space. Unlike traditional methods like Euler angles, quaternions excel in navigation and robotics because they can sidestep issues like gimbal lock and offer seamless, efficient rotations. Essentially, they provide a reliable means to accurately track and manipulate the orientation of objects in a 3D environment.

Quaternions were introduced by Hamilton [18]. Quaternions can be seen as an extension of complex numbers. While a complex number can depict rotation in a two-

dimensional plane, a quaternion extends this concept to three dimensions, representing rotation around three axes instead of just one.

For representation of a quaternion, there are four values. One real component q_0 with three imaginary components are q_1 , q_2 , q_3 ,

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{2.1}$$

A quaternion is commonly represented as a normalized vector. Quaternions can be normalized, where ||q|| can be calculated as:

$$q_{norm} = \frac{q}{||q||} \tag{2.2}$$

$$||q|| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{2.3}$$

We can obtain quaternion from Euler angles, which are roll angle(φ), pitch angle (θ) and yaw angle (ψ).

$$q_0 = \cos\left(\frac{\varphi}{2}\right) * \cos\left(\frac{\theta}{2}\right) * \cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\varphi}{2}\right) * \sin\left(\frac{\theta}{2}\right) * \sin\left(\frac{\psi}{2}\right) \tag{2.4}$$

$$q_1 = \sin\left(\frac{\varphi}{2}\right) * \cos\left(\frac{\theta}{2}\right) * \cos\left(\frac{\psi}{2}\right) - \cos\left(\frac{\varphi}{2}\right) * \sin\left(\frac{\theta}{2}\right) * \sin\left(\frac{\psi}{2}\right) \tag{2.5}$$

$$q_2 = \cos\left(\frac{\varphi}{2}\right) * \sin\left(\frac{\theta}{2}\right) * \cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\varphi}{2}\right) * \cos\left(\frac{\theta}{2}\right) * \sin\left(\frac{\psi}{2}\right) \tag{2.6}$$

$$q_3 = \cos\left(\frac{\varphi}{2}\right) * \cos\left(\frac{\theta}{2}\right) * \sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\varphi}{2}\right) * \sin\left(\frac{\theta}{2}\right) * \cos\left(\frac{\psi}{2}\right) \tag{2.7}$$

For example, an attitude matrix is defined as $d = [0 \quad 0 \quad 90]$, which is roll, pitch and yaw angle, respectively. If we want to express this matrix in terms of quaternion, $0.7071 + 0i + 0j + 0.7071k$

2.1.6 Euler Angles

Euler angles were initially introduced by Leonhard Euler[18]. According to Euler theorem, any rotation in space can be described using three angles (φ, θ, ψ): roll, pitch and yaw.

Euler angles appear to be intuitive and straightforward to use. However, they have a limitation in literature which known as gimbal lock. This singularity results in the loss of one degree of freedom in a three-gimbal system. Gimbal lock occurs when two of the three gimbals align, causing one degree of freedom to be lost.

We can obtain Euler angles from quaternions, which are q_0 , q_1 , q_2 , q_3 . Detailed information about figure given in [50].

$$\varphi = \arctan\left(2 * \frac{q_0 * q_1 + q_2 * q_3}{1 - 2 * (q_1 * q_1 + q_2 * q_2)}\right) \tag{2.8}$$

$$\theta = \arcsin\left(2 * (q_0 * q_2 - q_3 * q_1)\right) \tag{2.9}$$

$$\psi = \arctan\left(2 * \frac{q_0 * q_3 + q_1 * q_2}{1 - 2 * (q_2 * q_2 + q_3 * q_3)}\right) \tag{2.10}$$

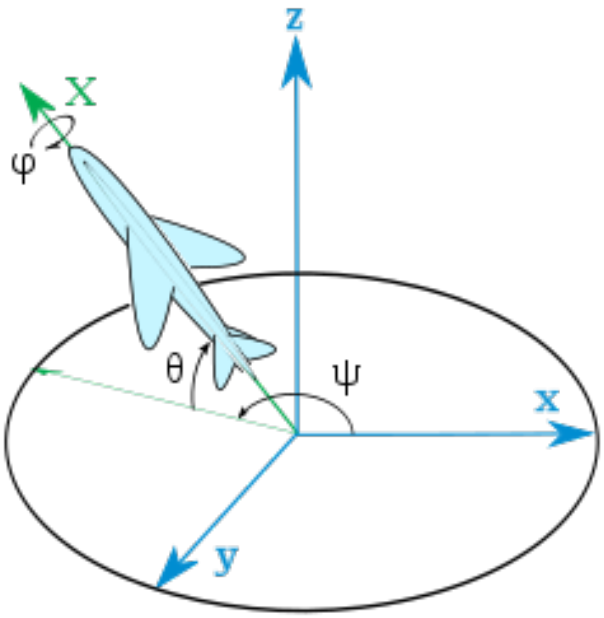


Figure 6: Euler Angles Representation

2.2 SENSORS USED IN NAVIGATION SYSTEMS

This study uses a MEMS accelerometer, a MEMS gyroscope, and a magneto-resistive magnetometer to design an AHRS system. An accelerometer is a sensor designed to measure and detect acceleration, which is the rate of change of velocity of an object in motion. Acceleration is typically expressed in units of meters per second squared (m/s^2) or gravitational force units (g), where 1 g is equal to the acceleration due to gravity at the Earth's surface (approximately 9.81 m/s^2) [18]. Detailed information about figure given in [51].

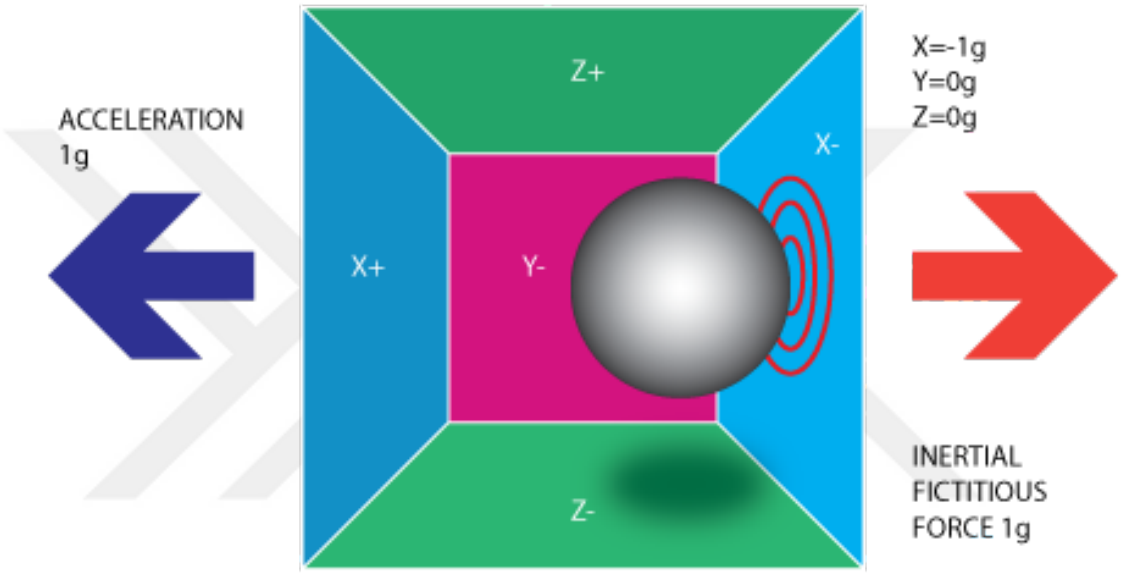


Figure 7: Accelerometer Sensor

A gyroscope is a device that measures and maintains orientation and angular velocity. It detects and quantifies rotational motion in various applications, such as navigation, robotics, and stabilization. Gyroscopes are vital for preserving motion objects' stability and control, especially when precise orientation is crucial [18]. Detailed information about figure given in [52].

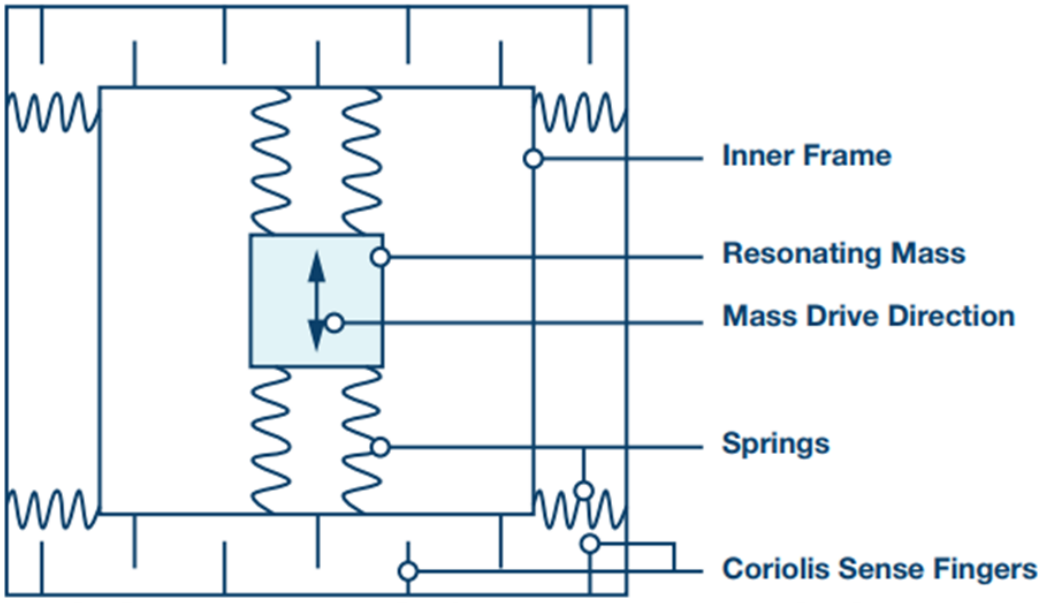


Figure 8: Gyroscope Sensor

A magnetometer is a device that measures the strength and direction of a magnetic field. It is used to detect and quantify the presence of magnetic fields, making it valuable in a wide range of applications, including navigation, geophysics, archaeology, and consumer electronics [18]. Electronic devices and metals degrade the magnetic field around them in various ways. These are divided into two: soft and *hard iron* effects.

Soft iron distortions occur when the magnetic field bends around materials that interact with it. When such materials are near a sensor, they can alter or weaken the magnetic field, reducing the compass's sensitivity or angular accuracy. These distortions do not generate strong magnetic fields or significantly disrupt nearby magnetic fields due to the soft iron's relative magnetic permeability. This results in uneven sensitivity to the Earth's geomagnetic field.

Common sources of soft iron distortion include materials such as cobalt ,iron, nickel, and their alloys, stainless steels. These materials are often used in various standard components, such as heat sinks, fasteners, hinges, connectors, batteries, spring steel dome switches, chip capacitors, and integrated circuits. The presence of these materials can alter the detected magnetic field, typically converting the spherical perceived field into an ellipsoid due to the distortion caused by the soft iron.

- Soft iron calibration is more difficult to calibrate than hard iron distortions. It requires more processing.

- Involves mapping the detected magnetic field and sensor orientation, followed by prior correction based on an ellipsoid model.

Hard iron distortion refers to the scaling of the observed magnetic field due to magnetized materials or rare earth magnetic materials. This distortion adds a constant additional magnetic field to the Earth's magnetic field, resulting in a consistent shift or displacement relative to the magnetic field source. These effects persist regardless of the location or orientation of the sensing platform.

Devices commonly responsible for hard iron distortion include motors, speakers and actuators.

Hard iron distortion;

- can be easily calibrated.
- requires only offset information according to the origin.
- is calibrated by subtracting the offset value directly from the measured data of the effected magnetometer.

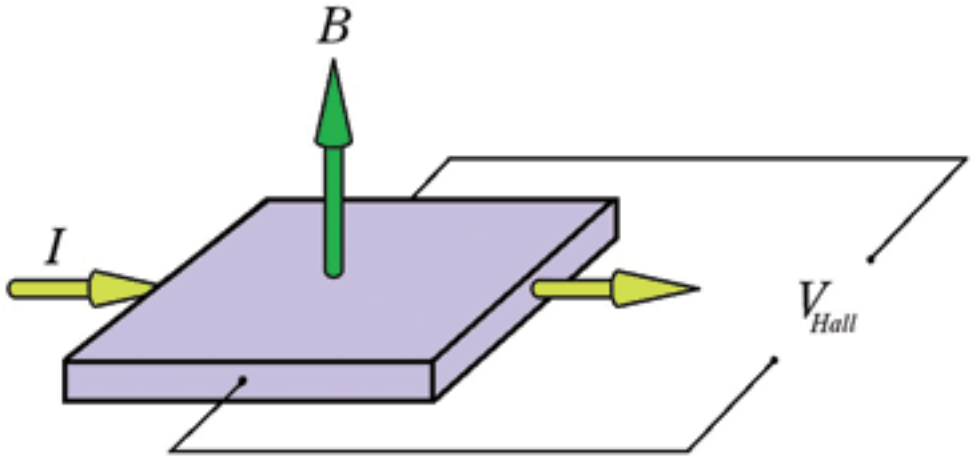


Figure 9: Magnetometer Sensor

Detailed information about figure given in [53].

2.3 DIRECTION COSINE MATRIX

The data received from the sensor is in the sensor frame. However, since the angles to be read on the vehicle are in the body frame, conversion is needed to switch between these frames.

A Direction Cosine Matrix (DCM) [18] is a mathematical representation used in various fields, including aerospace, robotics, and computer graphics. Its primary purpose is to describe the orientation of one coordinate frame with respect to another.

DCMs are particularly useful for representing the rotation between different reference frames in three-dimensional space. A Direction Cosine Matrix is typically a 3x3 orthogonal matrix that relates the unit vectors (usually, the x, y, and z axes) of one coordinate frame to the unit vectors of another. The matrix elements represent the direction cosines between the axes of the two frames. The direction cosines quantify how one set of axes is oriented relative to the other.

In aerospace and attitude determination, DCMs are used to describe the orientation of an object, such as an aircraft or a spacecraft, in three-dimensional space. By applying a DCM transformation, you can convert measurements taken in one reference frame into another frame or compute the attitude (orientation) of the object.

DCMs are essential in applications such as spacecraft attitude control, navigation, and robotics, where precise knowledge of orientation and coordinate frame transformations is

$$R_a^b = \begin{bmatrix} cos\,(\theta_{x^a,x^b}) & cos\,(\theta_{y^a,x^b}) & cos\,(\theta_{z^a,x^b}) \\ cos\,(\theta_{x^a,y^b}) & cos\,(\theta_{y^a,y^b}) & cos\,(\theta_{z^a,y^b}) \\ cos\,(\theta_{x^a,z^b}) & cos\,(\theta_{y^a,z^b}) & cos\,(\theta_{z^a,z^b}) \end{bmatrix} \tag{2.11}$$

where θ_{x^a,x^b} is the unsigned angle between the x_a axis and the x_b axis. This process effectively rotates the reference frame via the projection; it does *not* rotate the vector but views the same vector from a different frame.

The single-axis frame rotation transformations[19] are given by:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,(\theta) & sin\,(\theta) \\ 0 & -sin\,(\theta) & cos\,(\theta) \end{bmatrix} \tag{2.12}$$

$$R_y(\theta) = \begin{bmatrix} cos\,(\theta) & 0 & -sin\,(\theta) \\ 0 & 1 & 0 \\ sin\,(\theta) & 0 & cos\,(\theta) \end{bmatrix} \tag{2.13}$$

$$R_z(\theta) = \begin{bmatrix} cos\,(\theta) & sin\,(\theta) & 0 \\ -sin\,(\theta) & cos\,(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

CHAPTER III
SENSOR FUSION ALGORITHMS

As mentioned in Chapter II, three different sensors are used in AHRS. Roll, pitch and yaw angles are obtained by fusion of accelerometer and magnetometer sensors. At the same time, roll, pitch and yaw angles can be obtained only with the gyroscope. The importance given to these sensors must be adjusted depending on whether the system is moving or stationary. Therefore, these three-sensor data need to be fused. The Extended Kalman Filter, Unscented Kalman Filter and Complementary Filter were used for this. In this section, the algorithms used are explained. The Kalman Filter, which forms the basis of most of these filters, will be presented to illustrate other filters better.

Kalman filter is one of the most critical and standard estimation algorithms. The Kalman filter produces estimates of latent variables based on inaccurate and uncertain measurements. A Kalman filter design study was conducted considering the literature research carried out during AHRS studies. The most used types are Linear Kalman Filter [20], Extended Kalman Filter [21] and Unscented Kalman Filter[22]. The Kalman filter has two steps:

- **Prediction step**,

It predicts the next state and covariance of the system at time t, given its former state at time t-1.

$$\widehat{\boldsymbol{x_t}} = \boldsymbol{F} * \boldsymbol{x_{t-1}} + B * \boldsymbol{u_t} \tag{3.1}$$

$$\widehat{\boldsymbol{P_t}} = \boldsymbol{F} * \boldsymbol{P_{t-1}} * \boldsymbol{F^T} + Q_t \tag{3.2}$$

- **Correction step**,

It corrects the prediction with a series of measurements at time t.

$$\boldsymbol{v_t} = z_t - \boldsymbol{H} * \boldsymbol{x_t} \tag{3.3}$$

$$\boldsymbol{S_t} = \boldsymbol{H} * \boldsymbol{P_t} * \boldsymbol{H^T} + R \tag{3.4}$$

$$K_t = \boldsymbol{P_t} * \boldsymbol{H^T} * S_t^{-1} \tag{3.5}$$

$$\boldsymbol{x_t} = \boldsymbol{x_t} + K_t * \boldsymbol{v_t} \tag{3.6}$$

$$\boldsymbol{P_t} = \boldsymbol{P_t} - K_t * \boldsymbol{S_t} * K_t^T \tag{3.7}$$

Table 1: Variables in Kalman Filter.

Variable Name	Definition
$\widehat{\boldsymbol{P_t}}$	Estimated covariance of the state before seeing the measurements
$\boldsymbol{P_t}$	Estimated covariance of the state after seeing its measurements
$\boldsymbol{u_t}$	Control input vector that defines the system's expected behavior
B	Control input model
$\boldsymbol{H}$	Observation model that connects the predicted state and measurements
$\boldsymbol{v_t}$	Innovation or measurement residue.
$\boldsymbol{S_t}$	Measurement estimate covariance
K_t	Filter gain, or retention gain, tells you how much the predictions must be corrected.

Although the Kalman filter was examined above, it is not a suitable estimator for our problem because the Kalman filter is used in the solution of linear systems. However, since our system is nonlinear, EKF and UKF were used.

Within the scope of this study, 4 quaternions and 3 gyroscope bias error values are defined as the state vector of the state estimator, and it works as 7 states in total.

$$\boldsymbol{x_t} = [q_s\ q_x\ q_y\ q_z\ b_x\ b_y\ b_z] \tag{3.8}$$

Measurement Covariance Matrix

This matrix is the current measurement of the system and is based on the accelerometer and magnetometer. Initial values are chosen by calculating the estimated noise of the accelerometer and magnetometer.

$$\begin{bmatrix} \sigma_{A_x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{A_y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{A_z}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{M_x}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{M_y}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{M_z}^2 \end{bmatrix} \tag{3.9}$$

Table 2: Measurement Covariance Matrix Variables Definitions

Variable Name	Definition
σ_{A_x}	X-Axis Accelerometer Measurement Error
σ_{A_y}	Y-Axis Accelerometer Measurement Error
σ_{A_z}	Z-Axis Accelerometer Measurement Error
σ_{M_x}	X-Axis Magnetometer Measurement Error
σ_{M_y}	Y-Axis Magnetometer Measurement Error
σ_{M_z}	Z-Axis Magnetometer Measurement Error

Error Covariance Matrix

The 7x7 error covariance matrix contains the error in the measurements. The diagonal quaternions contain the gyroscope error shift.

$$\begin{bmatrix} \sigma_{q_s}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{q_x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{q_y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{q_z}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{b_x}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{b_y}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{b_z}^2 \end{bmatrix} \tag{3.10}$$

Table 3: Error Covariance Matrix Variables Definitions

Variable Name	Definition
σ_{q_s}	Quaternion Constant Term Component Error
σ_{q_x}	Quaternion X-Axis Component Error

Table 3 Continued

σ_{q_y}	Quaternion Y-Axis Component Error
σ_{q_z}	Quaternion Z-Axis Component Error
σ_{b_x}	X-Axis Gyroscope Error
σ_{b_y}	Y-Axis Gyroscope Error
σ_{b_z}	Z-Axis Gyroscope Error

The error covariance matrix is updated at each iteration and used to calculate the Kalman gain.

3.1 EXTENDED KALMAN FILTER

Extended Kalman filter has two steps:

- **Prediction step**,

It predicts the next state and covariance of the system at time t, given its former state at time t-1.

$$\widehat{\boldsymbol{x_t}} = \boldsymbol{f}(\boldsymbol{x_{t-1}}, \boldsymbol{u_t}) \tag{3.11}$$

$$\widehat{\boldsymbol{P_t}} = \boldsymbol{f}(\boldsymbol{x_{t-1}}, \boldsymbol{u_t}) * \boldsymbol{P_{t-1}} * \boldsymbol{F^T}(\boldsymbol{x_{t-1}}, \boldsymbol{u_t}) + Q_t \tag{3.12}$$

f is the nonlinear dynamic model,

- **Correction step**,

It corrects the prediction with a series of measurements at time t.

$$\boldsymbol{v_t} = z_t - \boldsymbol{h}(\boldsymbol{x_t}) \tag{3.13}$$

$$\mathbf{S_t} = \boldsymbol{H}(\boldsymbol{x_t}) * \widehat{\boldsymbol{P_t}} * \boldsymbol{H^T}(\boldsymbol{x_t}) + \mathbf{R_t} \tag{3.14}$$

$$K_t = \widehat{\mathbf{P}}_\mathbf{t} * \mathbf{H^T}(\mathbf{x_t}) * \boldsymbol{S_t^{-1}} \tag{3.15}$$

$$\boldsymbol{x_t} = \widehat{\boldsymbol{x_t}} + K_t * \boldsymbol{v_t} \tag{3.16}$$

$$\mathbf{P_t} = (\mathbf{I_{7x7}} - K_t * \boldsymbol{H}(\boldsymbol{x_t})) * \widehat{\boldsymbol{P_t}} \tag{3.17}$$

h is the nonlinear measurement model.

The matrices F and H are the Jacobians of f and h, respectively:

$$\mathbf{F}(\boldsymbol{x_{t-1}}, \boldsymbol{u_t}) = \frac{\boldsymbol{\partial f}(\boldsymbol{x_{t-1}}, \boldsymbol{u_t})}{\boldsymbol{\partial x}} \tag{3.18}$$

$$\mathbf{H}(\boldsymbol{x_t}) = \frac{\boldsymbol{\partial h}(\boldsymbol{x_t})}{\boldsymbol{\partial x}} \tag{3.19}$$

3.2 UNSCENTED KALMAN FILTER

UKF [22] handles EKF's zoom issues. The state distribution is again represented by a Gaussian Random Variable (GRV), but now it is specified using a minimum set selected from specific locations. These sample points fully capture the true mean and covariance of the GRV and, when propagated through the actual nonlinear system, accurately capture the posterior mean and covariance for any nonlinearity down to the 3rd order (Taylor series expansion). Unscented transformation (UT) is a method for calculating the statistics of a random variable that undergoes a nonlinear transformation.

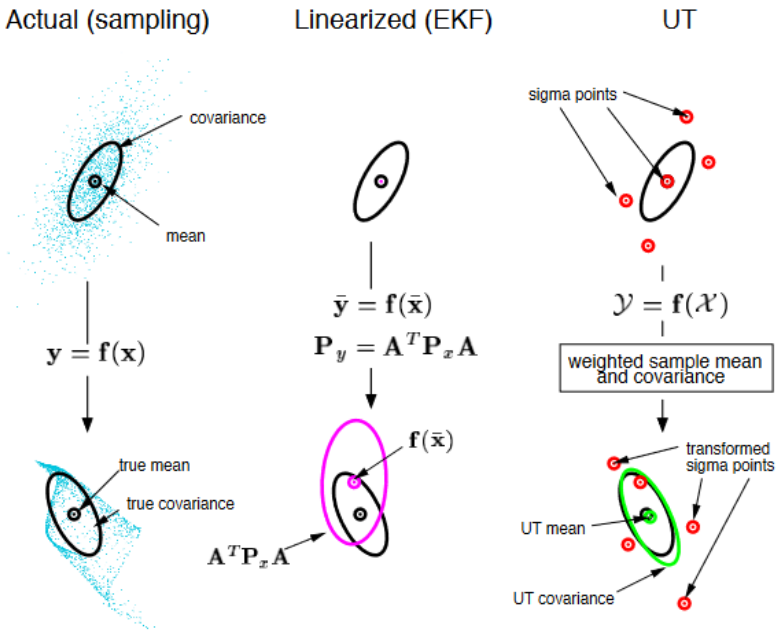


Figure 10: Unscented Transform Sampling vs EKF Sampling Scheme

For a 2-dimensional system:

- The left graph shows the true mean and covariance using Monte-Carlo sampling.
- The middle graph shows the results using a linearization approach as in EKF.

The graph on the right shows the performance of UT (only 5 sigma points are required).

➢ **Calculating Sigma Points**

The number of sigma points varies depending on the size of the system. The formula of the total sigma points number is 2N + 1.N specify dimension of the system.

$$\boldsymbol{X_0 = \mu} \tag{3.20}$$

$$\boldsymbol{X_i} = \boldsymbol{\mu} + \sqrt{((n+\tau) * \boldsymbol{\Sigma})} \tag{3.21}$$

$$\boldsymbol{X_i} = \boldsymbol{\mu} - \sqrt{((n+\tau) * \boldsymbol{\Sigma})} \tag{3.22}$$

Table 4: Sigma Point Formulas Variables Definitions

Variable Name	Definition
X	Sigma Matrix
$\boldsymbol{\mu}$	Gaussian Mean
n	Systems Dimension
τ	Scale Factor
Σ	Covariance Matrix

X indicates the sigma matrix. An important point to note is that each column **X** defines sigma points. Therefore, if we work in 3 dimensional system, the matrix size will be 3x7. So, for each dimension we have 7 sigma points.

τ represents the scaling factor indicating the distance from the mean at which we select our sigma points.

$$\boldsymbol{\Sigma = SS^T} \tag{3.23}$$

- **Calculating Sigma Points Weights**

Calculating the weight of the mean involves a different equation compared to the other sigma points. "τ" is the propagation parameter, and n is the dimensionality. The important part is that the sum of all weights is need to be equal to 1.

$$\omega_0 = \frac{\tau}{n + \tau} \tag{3.24}$$

$$\omega_i = \frac{\tau}{2 * (n + \tau)} \tag{3.25}$$

- **Calculating Mean and Covariance of Approximate Gauss**

$$\mu' = \Sigma_{i=0}^{2n} \omega_i * \boldsymbol{g}(\boldsymbol{X_i}) \tag{3.26}$$

$$\Sigma' = \omega_i * (\boldsymbol{g}(\boldsymbol{X_i}) - \mu')(\boldsymbol{g}(\boldsymbol{X_i}) - \mu')^T \tag{3.27}$$

Table 5: Mean and Covariance Formulas Variables Definitions

Variable Name	Definition
μ'	Predicted Mean
$\boldsymbol{\Sigma'}$	Predicted Covariance
ω	Sigma Points Weight
$\boldsymbol{g}$	Nonlinear Measurement Function
$\boldsymbol{X}$	Sigma Points Matrix
n	Number Sampling Points

- **Prediction step**

$$\mu' = \Sigma_{i=0}^{2n} \omega_i * \boldsymbol{g}(\boldsymbol{X_i}) \tag{3.28}$$

$$\Sigma' = \omega_i * (\boldsymbol{g}(\boldsymbol{X_i}) - \mu')(\boldsymbol{g}(\boldsymbol{X_i}) - \mu')^T + R_t \tag{3.29}$$

- **Correction step**

In this step, sigma points can be reconstructed because the estimated mean and variance values change. The following equations are used to transition from the state space to the measurement state space:

$$\boldsymbol{Z_i} = \boldsymbol{h}(\boldsymbol{X}) \tag{3.30}$$

$$\boldsymbol{z} = \Sigma_{i=0}^{2n} \omega_i * \boldsymbol{Z_i} \tag{3.31}$$

$$\boldsymbol{S} = \Sigma_{i=0}^{2n} \omega_i (\boldsymbol{Z_i} - \boldsymbol{z})(\boldsymbol{Z_i} - \boldsymbol{z})^T + Q \tag{3.32}$$

Table 6: UKF Algorithm Formulas Variables Definitions

Variable Name	Definition
$\boldsymbol{Z_i}$	Transformed sigma points in measurement area
$\boldsymbol{X_i}$	Sigma points matrix
z	Measurement vector
S	Covariance of measurement area
Q	Covariance of unpredictable noise
h	The function which maps sigma points to the measurement space

A crucial aspect here is that Z represents the measurement domain, which is the output from the sensor. Therefore, a function h is necessary to convert the state space into the measurement space, ensuring both are in equivalent units.

$$\mathbf{T} = \Sigma_{i=0}^{2n} \omega_i (\boldsymbol{X_i} - \boldsymbol{\mu'})(\boldsymbol{Z_i} - \boldsymbol{z})^T \tag{3.33}$$

$$K = \mathbf{T} * \mathbf{S^{-1}} \tag{3.34}$$

$$\boldsymbol{\mu} = \boldsymbol{\mu'} + K(\boldsymbol{Z_i} - \boldsymbol{z}) \tag{3.35}$$

$$\boldsymbol{\Sigma} = (\mathbf{I_{7x7}} - K * \mathbf{T}) * \boldsymbol{\Sigma'} \tag{3.36}$$

Table 7: UKF Algorithm Formulas Variables Definitions

Variable Name	Definition
T	Cross correlation matrix between state space and predicted space
S	Estimated covariance matrix
K	Kalman gain parameter

3.3 COMPLEMENTARY FILTER

The complementary filter [23] is easiest method for sensor fusion. It relies on the concept that one sensor can correct or offset the errors made by another sensor.

Gyroscopes tend to experience low-frequency drift, whereas accelerometers and magnetometers are affected by high-frequency noise. The complementary filter works by blending these two signals, effectively removing the gyroscope's drift and the accelerometer's noise.

Initially, the tilt angles calculated from the accelerometer readings are as follows:

$$\theta_m = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} = \begin{bmatrix} tan^{-1}\left(\frac{a_y}{a_x}\right) \\ tan^{-1}\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \\ 0 \end{bmatrix} \tag{3.37}$$

Only the roll, θ_x , and pitch, θ_y , angles are computed, yaw angle, θ_z is equal to zero. With a magnetometer sample, it becomes possible to compute the yaw angle. First, compensate the measured magnetic field using the tilt angles obtained from the accelerometer readings:

$$b = \begin{bmatrix} cos(\theta_x) & sin(\theta_x)\, sin(\theta_y) & sin(\theta_x)\, cos(\theta_y) \\ 0 & cos(\theta_y) & -sin(\theta_y) \\ -sin(\theta_x) & cos(\theta_x)\, sin(\theta_y) & cos(\theta_x)\, cos(\theta_y) \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \tag{3.38}$$

$$\begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} m_x\, cos(\theta_x) + m_y\, sin(\theta_x)\, sin(\theta_y) + m_z\, sin(\theta_x)\, cos(\theta_y) \\ m_y\, cos(\theta_y) - m_z\, sin(\theta_y) \\ -m_x\, sin(\theta_x) + m_y\, cos(\theta_x)\, sin\left(\theta_y\right) + m_z\, cos(\theta_x)\, cos(\theta_y) \end{bmatrix} \tag{3.39}$$

Then θ_z is obtained as:

$$\theta_z = \arctan\left(\frac{b_y}{b_x}\right) \tag{3.40}$$

Also, the attiude is estimated at t, but this time from a former attitude at $t-1$, and a given angular velocity, $\begin{bmatrix} w_x & w_y & w_z \end{bmatrix}^T$, in rad/s, using the most straightforward numerical integration:

$$\theta_w = \begin{bmatrix} \theta_{x_t} \\ \theta_{y_t} \\ \theta_{z_t} \end{bmatrix} = \begin{bmatrix} \theta_{x_{t-1}} + w_x \Delta_t \\ \theta_{y_{t-1}} + w_y \Delta_t \\ \theta_{z_{t-1}} + w_z \Delta_t \end{bmatrix} \tag{3.41}$$

Where Δ_t is the delta time between the current and former measurements, described as the sampling period or time step.

Finally, the estimations are merged using a complementary filter with a controlling parameter, α, in the range [0 1]

$$\theta = \alpha\theta_w + (1-\alpha)\theta_{am} \tag{3.42}$$

Where θ_w is the attitude which is estimated from the gyroscope, θ_{am} is the attitude which is derived from the magnetometer and the accelerometer sensor fusion, and α is the filter's gain.

The filter gain parameter must be a float value ranging between [0.0, 1.0]. When $\alpha = 0$, the attitude is estimated solely using data from the magnetometer and the accelerometer. When $\alpha = 1$, it is estimated exclusively with the gyroscope. The values within the range dictate how each estimation is weighted and integrated into the quaternion. [24].

CHAPTER IV
SENSOR ERROR MODELING AND SENSOR CALIBRATIONS

Sensor modeling plays a crucial role in navigation system design, as it involves the representation and simulation of sensors that gather data for the navigation process.

- Sensor models help designers evaluate the accuracy and precision of the navigation system. By accurately representing the behavior of sensors in various conditions, designers can assess how well the system can determine the position, orientation, and other relevant information.
- Before deploying a navigation system in real-world scenarios, it is essential to simulate and test its performance. Sensor models allow designers to create virtual environments to assess how the navigation system responds to different inputs and working conditions. These models aid in identifying potential issues and improving the system's robustness.
- Sensor models enable the simulation of sensor failures or malfunctions. These models help develop fault detection and diagnostics strategies, allowing the navigation system to recognize when a sensor provides unreliable or incorrect data and take appropriate corrective actions.
- Modern navigation systems often rely on a combination of different sensors, such as GPS, accelerometers, gyroscopes, magnetometers, and more. Sensor modeling facilitates the integration of these sensors by simulating their interactions. This integration is essential for achieving a more comprehensive and accurate navigation solution.
- Sensor models are essential for developing and refining algorithms that process sensor data. By understanding how sensors capture information, designers can create algorithms that effectively filter, fuse, and interpret the data to generate accurate navigation solutions.

4.1 ACCELEROMETER

The overall error model is given in Figure 11. There are two main types of error: Errors due to sensor production and errors due to sensor placement.

Sensor Bias Error is the output of the accelerometer when there is no movement, and Scale Factor Error is an error in the ratio of a change in the output signal to a change in the input signal to be measured. Detailed information is given in [25].

Misalignment Error is also known as non-orthogonality error. Mechanical components cannot be produced flawlessly and cannot be mounted perfectly. This results in non-orthogonality between gyroscope axes, and this non-orthogonality creates a scale factor effect on measurements. Detailed information is given in [25].

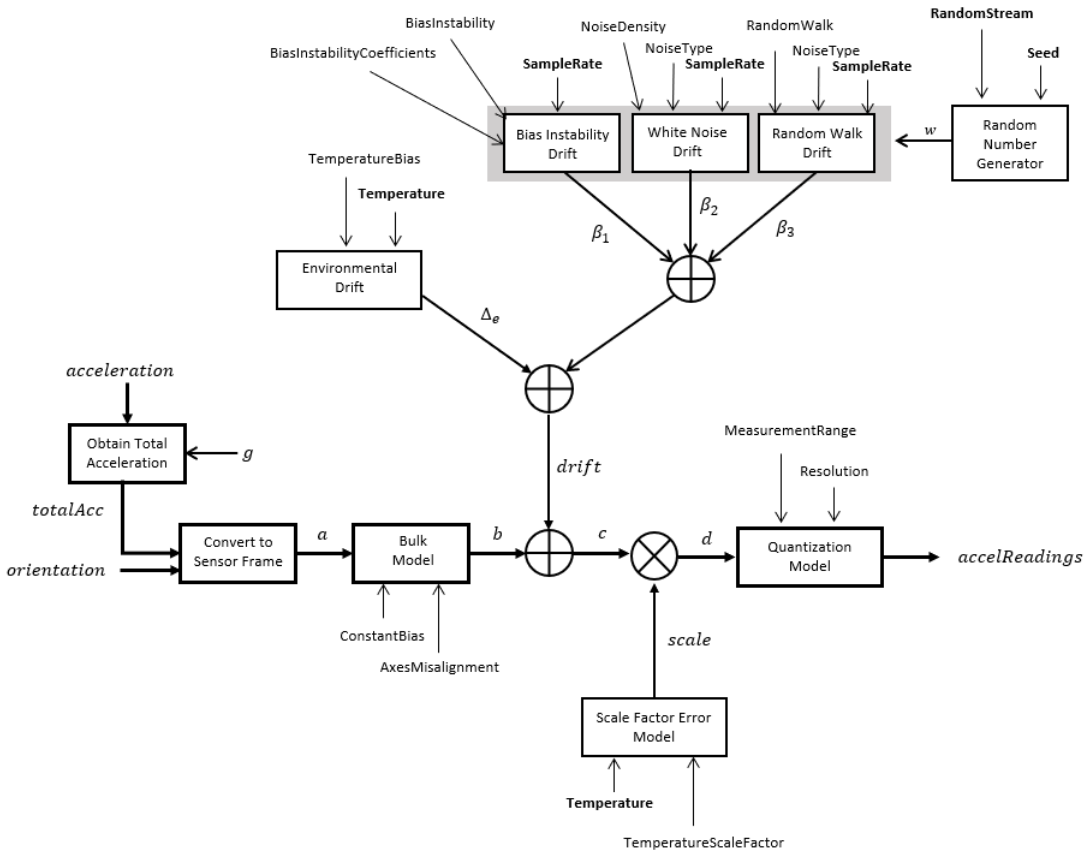


Figure 11: Accelerometer Sensor Error Model

Detailed information about figure given in [26].

4.1.1 The Mathematical Model

Due to the non-ideal nature of MEMS-type accelerometer sensors, error sources significantly impact their accuracy. If we aim to express these error sources mathematically for a three-axis, we can state:

$$\begin{bmatrix} \overline{a}_x \\ \overline{a}_y \\ \overline{a}_z \end{bmatrix} = \begin{bmatrix} (1 + S_x + \beta S_x)a_x + B_x + \beta B_x \\ (1 + S_y + \beta S_y)a_y + B_y + \beta B_y \\ (1 + S_z + \beta S_z)a_z + B_z + \beta B_z \end{bmatrix} \tag{4.1}$$

The variable represented by a_x, a_y, a_z in the formula are the x-axis, y-axis and z-axis values obtained from an ideal accelerometer sensor. The variable denoted by $\overline{a}_x, \overline{a}_y, \overline{a}_z$ the errors described affect the x-axis, y-axis and z-axis values obtained from the accelerometer sensor. The variable represented by S_x, S_y, S_z are the scale factor error parameters. The variable denoted by B_x, B_y, B_z are the measurement bias error parameters. $\beta S_x, \beta S_y, \beta S_z$ are the scale-factor error stability parameters, and $\beta B_x, \beta B_y, \beta B_z$ are the bias error stability parameters [25].

The $M_{xy}, M_{xz}, M_{yx}, M_{yz}, M_{zx}, M_{zy}$ are misalignment errors.

$$\begin{bmatrix} \overline{a}_x \\ \overline{a}_y \\ \overline{a}_z \end{bmatrix} = \begin{bmatrix} 1 + S_x + \beta S_x & M_{xy} & M_{xz} \\ M_{yx} & 1 + S_y + \beta S_y & M_{yz} \\ M_{zx} & M_{zy} & 1 + S_z + \beta S_z \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} B_x + \beta B_x \\ B_y + \beta B_y \\ B_z + \beta B_z \end{bmatrix} \tag{4.2}$$

The IMU block in MATLAB/Simulink has been used to model the accelerometer sensor. The IMU block uses the *accelparams* function [29]. The parameters and nominal values used to model the accelerometer are given in Table 8.

Table 8: MATLAB-accelparams Function Parameters Definitions and Values

Parameter	***Definition***	***Value***	***Unit***
Measurement Range	Maximum sensor reading	19.6	m/s^2

Table 8 Continued

Resolution	Resolution of sensor measurements	5.9e-4	$(m/s^2)/LSB$
Constant Bias	Constant sensor offset bias	0.1	m/s^2
Axes Misalignment	Sensor axes skew	1	%
Noise Density	Power spectral density of sensor noise	3.2e-4	$m/s^2/\sqrt{Hz}$
Bias Instability	Instability of the bias offset	5.1e-4	m/s^2
Random Walk	The integrated white noise of the sensor.	1.2e-4	$m/s^2 * \sqrt{Hz}$
Temperature Bias	Sensor bias from temperature	0.1	m/s^2/ºC
Temperature Scale Factor	Scale factor error from temperature	0.02	%/ºC

When there is no movement in the system, the ideal accelerometer gives us [0 0 9.81]$\frac{m}{s^2}$ acceleration in 3 axes.

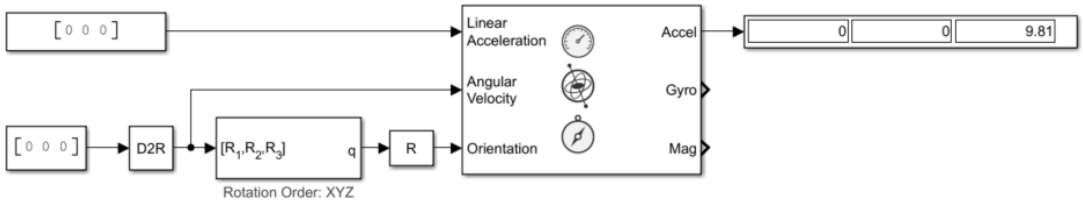


Figure 12: *Simple Accelerometer Simulink Model*

4.1.2 Sensor Calibration Process

Highly accurate outputs are expected from sensors used in industrial and defense industry applications, although these may vary depending on the application area. However, these sensors contain many error sources originating from design and production. Sensor error parameters must be found to make high-accurate measurements from sensors.

A sensor calibration algorithm is designed to reduce sensor errors. We need a ground truth accelerometer and a noncalibrated accelerometer to apply this algorithm. The data collected from these two sensors will be solved using the PSO method [27] and the objective function we will define.

- First, define 9 different data collection positions for sensor calibration. These points are selected according to the range interval at which we want to calibrate our sensor. These points are determined as follows:

Table 9: Data Collection Positions for Accelerometer Sensor Calibrations

	Roll Angle (°)	**Pitch Angle (°)**
1	0	0
2	0	25
3	0	-25
4	0	65
5	0	-65
6	25	0
7	-25	0
8	65	0
9	-65	0

The ground truth accelerometer and noncalibrated accelerometer data were collected at a 1-minute and 100 Hz output data rate for each point.

- Second, the Least Square Method [28] helps us predict outcomes based on existing datasets and identify any apparent anomalies in our data. The unconstrained optimization problem is defined as;

$$(\mathbf{M}_{\mathbf{acc}}^{*}, \mathbf{B}_{\mathbf{acc}}^{*}) = \text{argmin}\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{M}_{\mathbf{acc}}(\boldsymbol{a}_i - \mathbf{B}_{\mathbf{acc}}) - \overline{\boldsymbol{a}}\|^2}\right) \tag{4.3}$$

Table 10: Objective Function Formula Variables Definitions

Variable Name	**Definition**
$\boldsymbol{a_i}$	Uncalibrated accelerometer data
$\overline{\boldsymbol{a}}$	Ground truth accelerometer data
$\boldsymbol{n}$	Total data count
$\mathbf{M}_{\mathbf{acc}}$	3x3 scale factor and misalignment correction matrix
$\mathbf{B}_{\mathbf{acc}}$	1x3 bias correction matrix

- Lastly, PSO was used to solve the minimization problem. It gives a 3x3 scale error correction matrix and a 3x1 bias error correction matrix.

Simulation Results

In the MATLAB/Simulink environment, an ideal accelerometer model has been created to simulate the ground truth accelerometer, and a sensor error-added accelerometer model has been created to simulate the uncalibrated accelerometer.

The scale factor and misalignment error correction matrix have been found as follows;

$$\mathbf{M_{acc}} = \begin{bmatrix} 1.0006 & -0.0161 & 0.0256 \\ 0.0158 & 0.9920 & -0.0003 \\ 0.0323 & 0.0107 & 0.9686 \end{bmatrix}$$

$$\mathbf{B_{acc}} = [-0.0098 \quad -0.0133 \quad -0.0376]$$

Then, the matrix is applied to raw accelerometer data, and test data is generated to evaluate the sensor calibration method.

Table 11: Simulation Results Before and After Accelerometer Sensor Calibration

Ideal Angles (º)		**Uncalibrated Angles (º)**		**Calibrated Angles (º)**	
Pitch Angle	***Roll Angle***	***Pitch Angle***	***Roll Angle***	***Pitch Angle***	***Roll Angle***
0	0	1.39	1.38	-0.001	-0.02
-45	0	-42.96	1.43	-44.98	0.02
-30	0	-28.06	1.41	-30.00	0.001
-15	0	-13.27	1.39	-15.03	-0.01
0	15	1.39	15.98	-0.02	15.01
0	30	1.38	30.51	-0.01	30.01
0	45	1.38	43.99	-0.01	44.99

As a result, a pitch angle of 0.015 º RMSE and a roll angle of 0.014 º RMSE were obtained.

4.2 GYROSCOPE

The overall error model is given in Figure 13. There are two main types of errors: errors due to sensor production and errors due to sensor placement.

Sensor Bias Error is the gyroscope output when there is no movement; Scale Factor Error is an error in the ratio of a change in the output signal to a change in the input signal to be measured. Detailed information is given in [25].

Misalignment Error is also known as non-orthogonality error. Mechanical components cannot be produced flawlessly and cannot be mounted perfectly. This results in non-orthogonality between gyroscope axes, and this non-orthogonality creates a scale factor effect on measurements. Detailed information is given in [25].

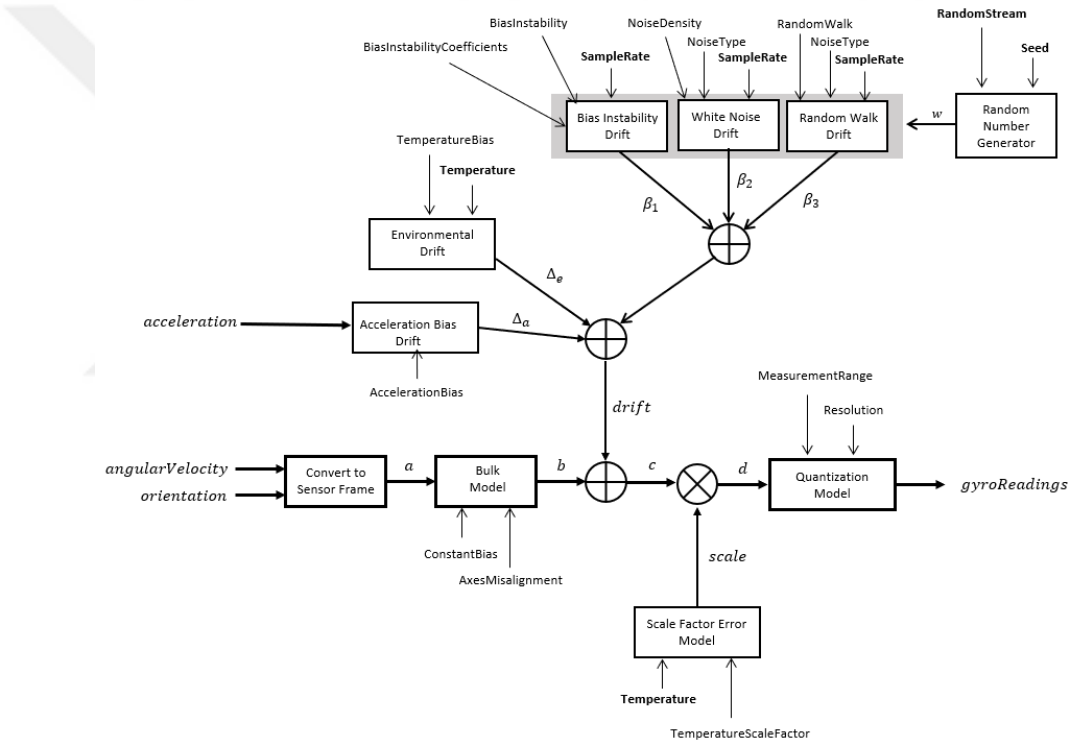


Figure 13: Gyroscope Sensor Error Model

Detailed information about figure given in [26].

4.2.1 The Mathematical Model

Due to the non-ideal nature of MEMS-type gyroscope sensors, error sources significantly impact their accuracy [25]. If we aim to express these error sources mathematically for a three-axis, we can state:

$$\begin{bmatrix}\overline{w}_x\\ \overline{w}_y\\ \overline{w}_z\end{bmatrix} = \begin{bmatrix}(1+S_x+\beta S_x)w_x + B_x + \beta B_x + B_{Gx}a_x + n_x\\ (1+S_y+\beta S_y)w_y + B_y + \beta B_y + B_{Gy}a_y + n_y\\ (1+S_z+\beta S_z)w_z + B_z + \beta B_z + B_{Gz}a_z + n_z\end{bmatrix} \quad (4.6)$$

Table 12: Gyroscope Mathematical Model Formula Variables Definitions

Variables	Definitions
w_x	Ideal Gyroscope Sensor X-Axis Value
w_y	Ideal Gyroscope Sensor Y-Axis Value
w_z	Ideal Gyroscope Sensor Z-Axis Value
$\overline{w}_x$	Affected Gyroscope Sensor X-Axis Value
$\overline{w}_y$	Affected Gyroscope Sensor Y-Axis Value
$\overline{w}_z$	Affected Gyroscope Sensor Z-Axis Value
S_x	X-Axis Scale Factor Error Parameter
S_y	Y-Axis Scale Factor Error Parameter
S_z	Z-Axis Scale Factor Error Parameter
B_x	X-Axis Measurement Bias Error
B_y	Y-Axis Measurement Bias Error
B_z	Z-Axis Measurement Bias Error
βS_x	X-Axis Scale Factor Error Stability Parameter
βS_y	Y-Axis Scale Factor Error Stability Parameter
βS_z	Z-Axis Scale Factor Error Stability Parameter
βB_x	X-Axis Bias Error Stability Parameter
βB_y	Y-Axis Bias Error Stability Parameter
βB_z	Z-Axis Bias Error Stability Parameter
B_{G_x}	X-Axis g-depended Bias Coefficient
B_{G_y}	Y-Axis g-depended Bias Coefficient
B_{G_z}	Z-Axis g-depended Bias Coefficient
n_x	X-Axis Random Sensor Noise
n_y	Y-Axis Random Sensor Noise
n_z	Z-Axis Random Sensor Noise

$$\begin{bmatrix}\overline{w}_x\\ \overline{w}_y\\ \overline{w}_z\end{bmatrix} = \begin{bmatrix}1+S_x+\beta S_x & M_{xy} & M_{xz}\\ M_{yx} & 1+S_y+\beta S_y & M_{yz}\\ M_{zx} & M_{zy} & 1+S_z+\beta S_z\end{bmatrix}\begin{bmatrix}w_x\\ w_y\\ w_z\end{bmatrix} + \begin{bmatrix}B_x+\beta B_x\\ B_y+\beta B_y\\ B_z+\beta B_z\end{bmatrix} + \begin{bmatrix}B_{Gx} & 0 & 0\\ 0 & B_{Gy} & 0\\ 0 & 0 & B_{Gz}\end{bmatrix}\begin{bmatrix}a_x\\ a_y\\ a_z\end{bmatrix} + \begin{bmatrix}n_x\\ n_y\\ n_z\end{bmatrix} \quad (4.7)$$

where M_{xy}, M_{xz}, M_{yx}, M_{yz}, M_{zx}, M_{zy} are misalignment errors. The IMU block in MATLAB/Simulink has been used to model the gyroscope sensor. The IMU block

uses the *gyroparams* function [30]. The parameters and nominal values used to model the gyroscope are given in Table 13.

Table 13: MATLAB-gyroparams Function Parameters Definitions and Values

Parameter	Definition	Value	Unit
MeasurementRange	Maximum sensor reading	4.32	rad/s^2
Resolution	Resolution of sensor measurements	5.9e-4	rad/LSB
ConstantBias	Constant sensor offset bias	0.1	rad/s
AxesMisalignment	Sensor axes skew	1	%
NoiseDensity	Power spectral density of sensor noise	3.2e-4	$rad/\sqrt{Hz}$
BiasInstability	Instability of the bias offset	5.1e-4	rad/s
RandomWalk	The integrated white noise of the sensor	1.2e-4	$rad * \sqrt{Hz}$
TemperatureBias	Sensor bias from temperature	0.1	$rad/ºC$
TemperatureScaleFactor	Scale factor error from temperature	0.02	$\%/ºC$

When there is no movement in the system, the ideal gyroscope gives us [0 0 0] rad/s angular velocity in 3 axes.

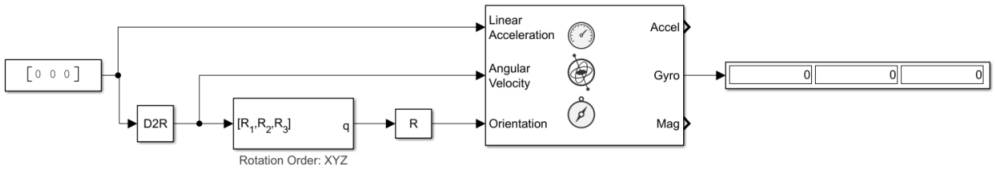


Figure 14: Simple Gyroscope Simulink Model

4.2.2 Sensor Calibration Process

The gyroscope is the most crucial sensor of AHRS systems if the platform on which AHRS is placed is moving. Therefore, calibrating the gyroscope will increase angle measurement accuracy in moving situations. However, unlike the accelerometer and magnetometer, the ground truth data for the gyroscope calibration cannot be collected in a stationary state. For this reason, the data has been collected by scanning

a particular angular path at constant velocity using our test rig. The time derivative of encoder readings is taken as ground truth. Then, the calibration coefficients are found by solving an optimization problem formulated as a least square sense. The Particle Swarm Optimization method has been preferred as a solver.

The procedure to obtain calibration coefficient:

- First, 6 different angular movement trajectories are defined for calibration. When determining these trajectories, the critical part is that the movement is applied in each axis. Additionally, the defined movements are completed at a constant angular speed. These trajectories are determined as follows:

Table 14: Data Collection Trajectories for Accelerometer Sensor Calibrations

Trajectory No	**Roll Angle Range (°)** [Start Angle, End Angle]	**Pitch Angle Range (°)** [Start Angle, End Angle]	**Yaw Angle Range (°)** [Start Angle, End Angle]
1	[0, 0]	[0, 0]	[0, 180]
2	[0, 0]	[0, 0]	[0, -180]
3	[0, 0]	[0, 180]	[0, 0]
4	[0, 0]	[0, -180]	[0, 0]
5	[0, 180]	[0, 0]	[0, 0]
6	[0, -180]	[0, 0]	[0, 0]

The ground truth and noncalibrated gyroscope data will be collected for each trajectory at a 1-minute and 100 Hz output data rate.

- Second, the Least Square Method calculates the error for each iteration. An unconstrained optimization problem is defined as;

$$\left(\mathbf{M}^{*}_{\mathbf{gyro}}, \mathbf{B}^{*}_{\mathbf{gyro}}\right) = \operatorname{argmin}\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left\|\mathbf{M}_{\mathbf{gyro}}(\boldsymbol{g}_i - \mathbf{B}_{\mathbf{gyro}}) - \overline{\boldsymbol{g}}\right\|^2}\right) \tag{4.8}$$

Table 15: Variables Definitions.

Variable Name	Definition
$\boldsymbol{g_i}$	Uncalibrated gyroscope data
$\bar{\boldsymbol{g}}$	Ground truth gyroscope data
n	Total data count
$\mathbf{M_{gyro}}$	3x3 scale factor and misalignment correction matrix
$\mathbf{B_{gyro}}$	1x3 bias correction matrix

- Lastly, the particle swarm optimization method solves the minimization problem. It gives a 3x3 scale error correction matrix and a 1x3 bias error correction matrix.

Simulation Results

In the MATLAB/Simulink environment, an ideal gyroscope model was created to simulate a ground truth gyroscope, and a sensor error added gyroscope model was developed to simulate an uncalibrated gyroscope.

The scale factor and misalignment error correction matrix have been found;

$$\mathbf{M_{gyro}} = \begin{bmatrix} 1.0003 & -0.0122 & -0.0124 \\ -0.0122 & 1.0002 & -0.0124 \\ -0.0125 & -0.0123 & 1.0001 \end{bmatrix}$$

Bias errors have been found as follows;

$$\mathbf{B_{gyro}} = [-0.0004 \quad -0.0004 \quad -0.0004]$$

Then, these matrices have been used on test data to evaluate the sensor calibration method.

Verifying the gyroscope sensor is done differently from the accelerometer and magnetometer due to how it works. Here, movements are completed in defined trajectories using calibrated and uncalibrated sensors for 1 hour. In this section, angle values were calculated using only simple integration without using any attitude estimator to calculate angle values. The uncalibrated accelerometer showed deviations up to 100 º due to error accumulation at the end of 1 hour.

Table 16: Simulation Results Before and After Gyroscope Sensor Calibration-1

	Roll Angle (º)	**Pitch Angle (º)**	**Yaw Angle (º)**
True Attitude	0	0	180
Calibrated Attitude	0.3657	0.8831	179.3234
Raw Attitude	105.4324	105.3668	283.0834

Table 17: Simulation Results Before and After Gyroscope Sensor Calibration-2

	Roll Angle (º)	**Pitch Angle (º)**	**Yaw Angle (º)**
True Attitude	0	0	-180
Calibrated Attitude	0.2013	1.0563	-180.4773
Raw Attitude	100.7511	101.0118	-76.7657

As a result, for yaw angle 0.677 º RMSE was obtained.

4.3 MAGNETOMETER

The overall error model is given in Figure 15. There are two main types of errors: Errors due to sensor production, errors due to sensor placement, and errors due to magnetic field changes.

Sensor offset error can be defined as the offset values that act and add to the magnetometer data. It creates the same effect as hard iron from environmental disturbances. Large deviations in the total magnetic field value are observed if not removed.

Scale factor error is the ratio of a change in the output signal to a change in the input signal to be measured. It proportionally distorts the values read on the axes, like the effect of soft iron due to environmental disturbances. Large deviations in the total magnetic field value are observed if not removed.

Misalignment Error is also defined as a non-orthogonality error. In ideal conditions, the magnetometer axes should be mounted vertically. However, mechanical parts cannot be produced perfectly and cannot be assembled perfectly. This imperfection causes a non-orthogonality between the magnetometer axes, which creates a scale factor effect on the measurements. Any movement in a Y-Axis causes changes in the other axes depending on the magnitude of the misalignment.

Magnetic distortions are defined as distortions in the lines of the magnetic field in which it is located, such as electronic devices, metals, and batteries. The soft and hard iron effects are examined in two parts. Detailed information about figure given in [26].

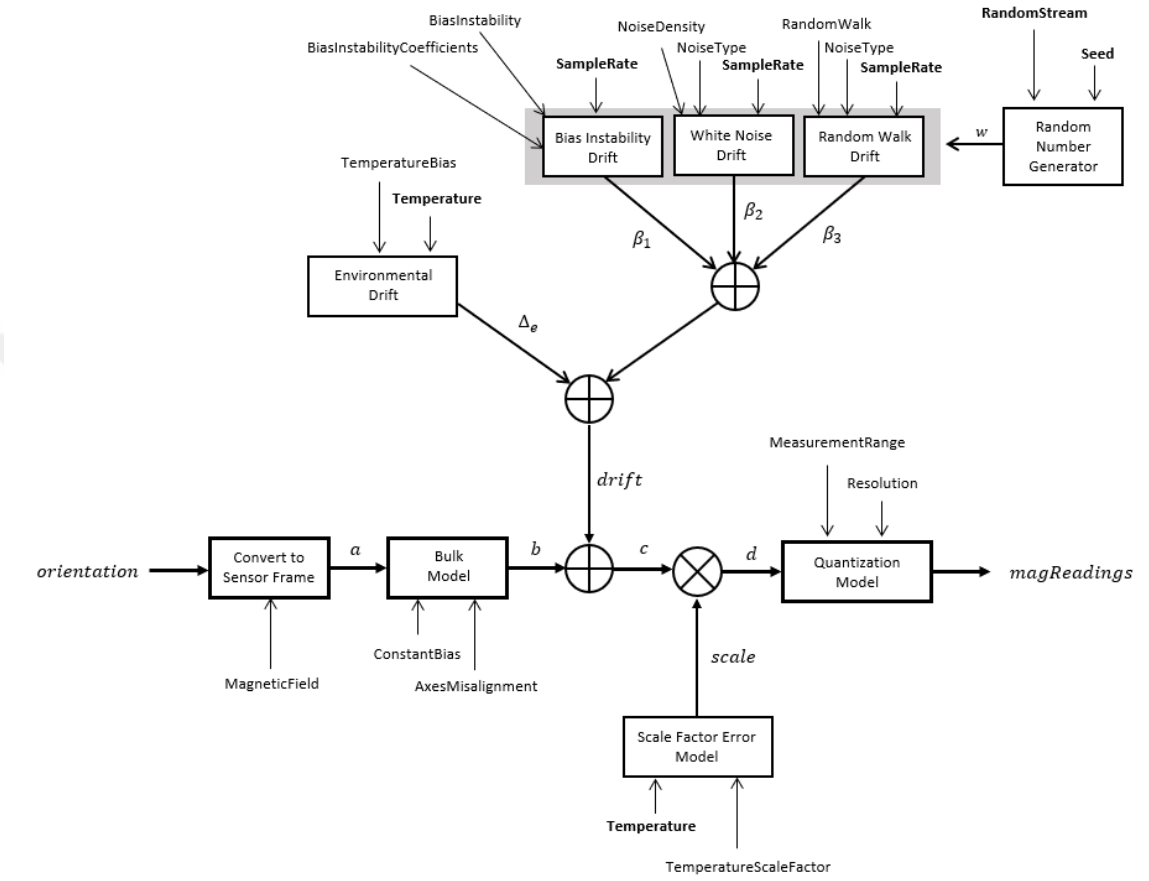


Figure 15: Magnetometer Sensor Error Model

4.3.1 Mathematical Error Model

Due to the non-ideal nature and react to magnetic field changes of MEMS-type magnetometer sensors, there exist error sources that significantly impact their accuracy.

If we aim to express these error sources mathematically[32], we can state:

$$H_m = C_{si}\, C_m\, C_{sf}\, C_{nor}\, H_e + O_{hi} + O_{sb} \tag{4.11}$$

$$H_m = C\, H_e + O_{hi} + O_{sb} \tag{4.12}$$

Table 18: Magnetometer Sensor Mathematical Model Formulas Variables Definitions

Variable Name	Definition
C_{si}	Soft Iron Effect
C_m	Misalignment Error
C_{sf}	Sensor Sensitivity
C_{nor}	Scale Factor Error
O_{hi}	Hard Iron Effect
O_{sb}	White noises and other insignificant noises

$C = C_{si}\, C_m\, C_{sf}\, C_{nor}$, expresses all errors and noises which effect the ideal magnetometer sensor. $H_e = [h_x\ h_y\ h_z]^T$, expresses ideal magnetometer sensor output. $H_m = [h_{mx}\ h_{my}\ h_{mz}]^T$ expresses 3-axis magnetometer sensor output.
In Equation (4.12), we neglect O_{sb} because of its insignificantly small value.
So, the equation turned.

$$H_m = C H_e + O_{hi} \tag{4.13}$$

Equation (3) can be corrected again to express the ideal magnetometer measurement result,

$$H_e = C^{-1}(H_m - O_{hi}) \tag{4.14}$$

$$H_e = M(H_m - O_{hi}) \tag{4.15}$$

$$M = C^{-1} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{4.16}$$

$$O_{hi} = [h_{11} \quad h_{12} \quad h_{13}] \tag{4.17}$$

Equations (4.16) and (4.17) are soft and hard impact correction matrices, respectively. Under ideal conditions, the magnetic field magnitude, H_e, in the environment where the magnetometer is located is constant.

The IMU block in MATLAB/Simulink has been used to model the magnetometer sensor. The IMU block uses the *magparams* function [31]. The parameters and nominal values used to model the magnetometer are given in Table 19.

Table 19: MATLAB-magparams Function Parameters Definitions and Values

Parameter	Definition	Value	Unit
MeasurementRange	Maximum sensor reading	800	uT
Resolution	Resolution of sensor measurements	5.9e-4	uT/LSB
ConstantBias	Constant sensor offset bias	0.1	uT
AxesMisalignment	Sensor axes skew	1	%
NoiseDensity	Power spectral density of sensor noise	3.2e-4	$uT/\sqrt{Hz}$
BiasInstability	Instability of the bias offset	5.1e-4	uT
RandomWalk	The integrated white noise of the sensor	1.2e-4	$uT * \sqrt{Hz}$
TemperatureBias	Sensor bias from temperature	0.1	$uT/$ºC
TemperatureScaleFactor	Scale factor error from temperature	0.02	%/ºC

In Ankara/Turkey according to NOAA[33] magnetic field values are;
in North Component 25.221 uT,
in East Component 2.747 uT,

Table 20: NOAA Ankara/TÜRKİYE Latitude and Longitude

Model Used:	WMM-2020
Latitude:	39° 55' 15" N
Longitude:	32° 51' 14" E
Elevation:	0.0 km Mean Sea Level

Table 21: NOAA Ankara/TÜRKİYE Magnetic Field Characteristics

Date	Declination	Inclination	Horizontal Intensity	North Comp	East Comp	Vertical Comp	Total Field
2024-02-16	6° 12' 58"	58° 9' 60"	25,370.9 nT	25,221.7 nT	2,747.1 nT	40,865.8 nT	48,100.9 nT
Change /year	0° 5' 7"/yr	0° 2' 54"/yr	3.5 nT/yr	-0.6 nT/yr	37.9 nT/yr	82.7 nT/yr	72.1 nT/yr
Uncertainty	0° 20'	0° 13'	128 nT	131 nT	94 nT	157 nT	145 nT

in Down Component 40.865 uT.

So, we can obtain those values from Simulink using an ideal magnetometer model.

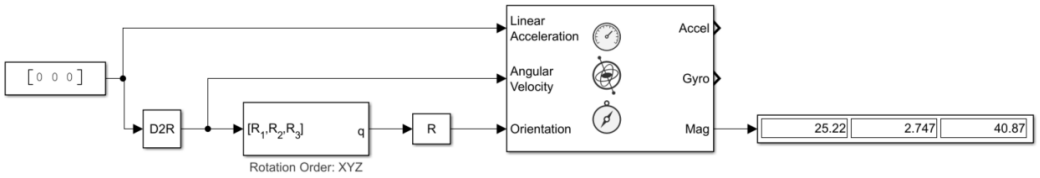


Figure 16: Simple Magnetometer Simulink Model

4.3.2 Sensor Calibration Process

Highly accurate magnetometers are essential for AHRS systems. However, magnetometers contain many error sources originating from design and production. Sensor error parameters must be found to make high-accurate measurements from sensors.

Sensor calibration algorithm has been designed to find sensor error parameters. The noncalibrated magnetometer should not contain any soft iron and hard iron effects. Only the sensor error should be on a magnetometer. Therefore, these effects should be removed before sensor calibration.

The data collected from these two sensors will be solved using the PSO method and the objective function we will define.

- First, define 12 different data collection positions for sensor calibration. These points are selected according to the range interval at which we want to calibrate our sensor. These points are determined as follows:

Table 22: Data Collection Positions for Magnetometer Sensor Calibrations

	Roll Angle (°)	**Pitch Angle (°)**	**Yaw Angle (°)**
1	30	0	0
2	30	0	60
3	30	0	120
4	30	0	180
5	30	0	240
6	30	0	300
7	-30	0	300
8	-30	0	240
9	-30	0	180
10	-30	0	120
11	-30	0	60
12	-30	0	0

The ground truth and noncalibrated magnetometer data will be collected for each point at a 1-minute and 100 Hz output data rate.

- Second, the Least Mean Square Method calculates the remaining error we create in the objective function. The objective function is defined as;

$$\left(\mathbf{M}^{*}_{\mathbf{mag}}, \mathbf{B}^{*}_{\mathbf{mag}}\right) = \text{argmin}\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left\|\mathbf{M}_{\mathbf{mag}}(\boldsymbol{m_i} - \mathbf{B}_{\mathbf{mag}}) - \bar{\boldsymbol{m}}\right\|^2}\right) \tag{4.18}$$

Table 23: Variables Definitions Eq 4.18

Variable Name	**Definition**
$\boldsymbol{m_i}$	Uncalibrated gyroscope data
$\bar{\boldsymbol{m}}$	Ground truth gyroscope data
n	Total data count
$\mathbf{M}_{\mathbf{mag}}$	3x3 scale factor and misalignment correction matrix
$\mathbf{B}_{\mathbf{mag}}$	1x3 bias correction matrix

- Lastly, the PSO method is used to solve the minimization problem. It gives a 3x3 scale error correction matrix and a 3x1 bias error correction matrix.

Simulation Results

In the MATLAB/Simulink environment, an ideal magnetometer model was created to simulate the ground truth magnetometer, and a sensor error-added magnetometer model was developed to simulate an uncalibrated magnetometer.

After the algorithm applied the scale factor and misalignment error correction matrix, find as follows;

$$\mathbf{M_{mag}} = \begin{bmatrix} 1.1183 & 0.0221 & 0.009 \\ -0.0237 & 1.1631 & -0.0037 \\ 0.0132 & -0.0230 & 1.1307 \end{bmatrix}$$

Bias errors are found as follows;

$$\mathbf{B_{mag}} = [-0.5222 \quad 0.8696 \quad -0.812]$$

Then, the matrix is used in raw magnetometer data and generates test data to evaluate the sensor calibration method.

Table 24: Simulation Results Before and After Magnetometer Sensor Calibration

Yaw Angle	Uncalibrated Yaw Angle º	Calibrated Yaw Angle º
0	1.56	0.02
60	58.14	60.02
120	123.21	119.98
180	178.56	180.01
240	242.18	239.99
300	297.67	300.03

As a result, for yaw angle 0.019 º RMSE error was obtained.

4.3.3 Soft and Hard Iron Calibration

Magnetometers can easily affect metals, electronic components or power sources. This soft and hard iron effect must be removed from the magnetometer data for high-accuracy yaw angle information. However, we cannot use an external ground truth sensor for this process, as we do to eliminate sensor calibration errors. Therefore, we must use only the magnetically affected data from the magnetometer sensor.

When we move an ideal magnetometer sensor only angularly around its center, we obtain a perfect sphere. However, when magnetically exposed magnetometer sensor outputs are examined similarly, we get an ellipsoid shape far from the ideal sphere. We need to follow a mathematically complex series of operations called ellipsoid fitting for this calibration.

- First, data should be collected on all axes so that the magnetometer makes only angular movements around its center.
- After, we provide all the collected data as input to the ellipsoid fitting method mentioned in Appendix B. For this calibration operation, the *magcal* function is used in MATLAB. It gives a 3x3 soft iron correction Matrix and a 3x1 hard iron correction Matrix.

Simulation

To verify the soft and hard iron calibration method with the ellipsoid fitting method in the simulation environment, we aimed to add certain distortions to the ideal magnetometer and then correct these distortions with the help of the *magcal* function.

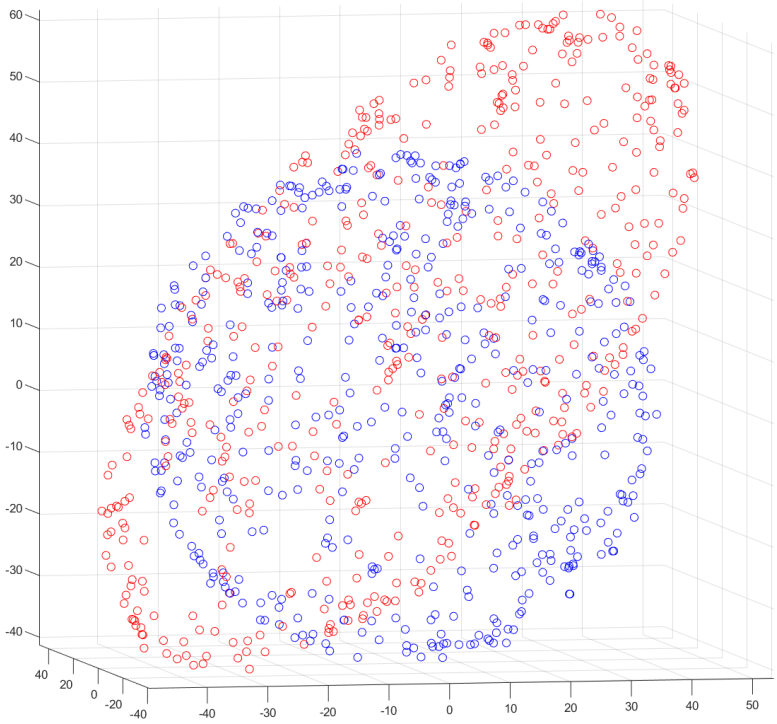


Figure 17: Calibrated and Uncalibrated Magnetometer Sensor Data

CHAPTER V
ATTITUDE ESTIMATOR DESIGN IN SIMULATION ENVIRONMENT

This section discusses the design of the attitude estimator in the simulation environment and the results' analysis. Error modeling and calibration of the sensors applied in this section are explained in detail in Chapter IV.

Attitude estimation for an unmanned aerial vehicle accelerometer, gyroscope and magnetometer sensors should be used in the AHRS we will design. In addition, GNSS data can also be added. Within the scope of this thesis, GNSS data is not used. When the literature is reviewed, the general idea in attitude estimators designed with the help of Kalman filters and without a support unit is to apply sensor fusion to the attitude information obtained from the accelerometer and magnetometer's internal sensor fusion and the attitude information obtained on the gyroscope.

For highly accurate attitude angles, all sensors must be calibrated. Additionally, soft & hard iron effect calibration must be performed for the magnetometer. The data of all sensors must be filtered with the help of digital filters before the attitude estimator to eliminate the effects of noise created by the system. A detailed system chart can be reviewed in Figure 18.

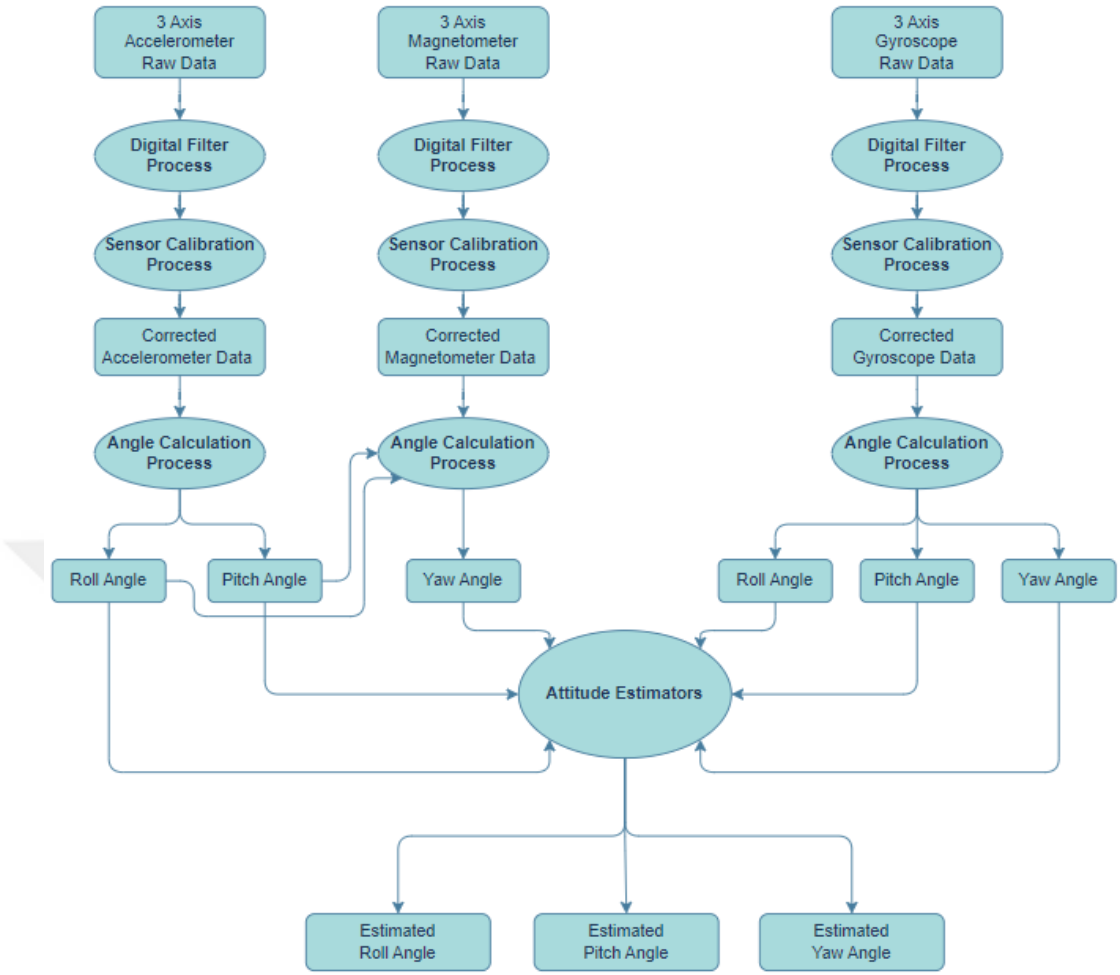


Figure 18: AHRS Algorithm Flowchart

The rotary wing unmanned aerial vehicle was modeled in the MATLAB/Simulink environment to design the attitude estimator and perform relevant verifications. While making this design, the attitude and altitude values on the desired axis can be adjusted with the help of PID controllers [34]. Error-free orientation information obtained with the help of the mathematical model of the unmanned aerial vehicle was used as a reference value in the design of the attitude estimator.

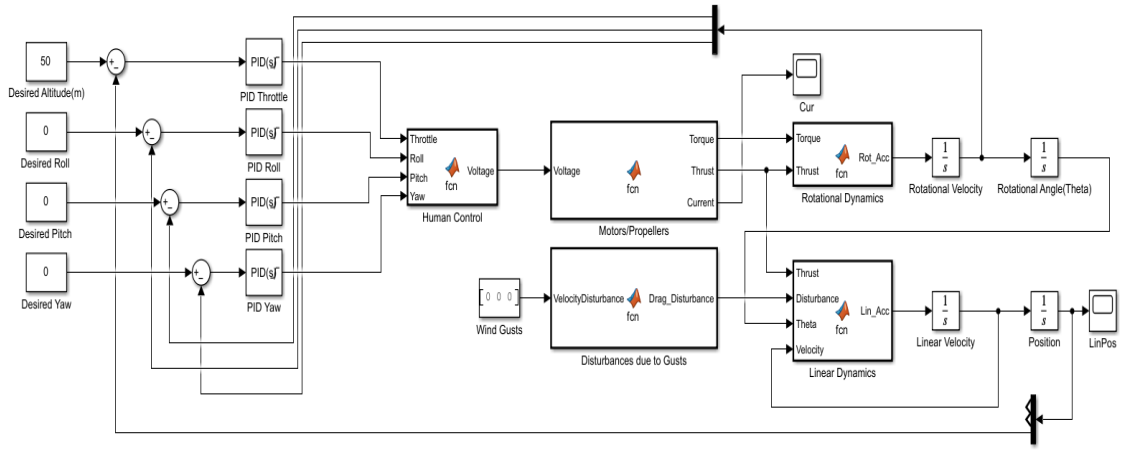


Figure 19: Quadcopter Simulink Model

5.1 ATTITUDE ESTIMATION with MATLAB/SIMULINK AHRS FUNCTION

The AHRS function in the MATLAB/Simulink library, which uses an EKF-based attitude estimator, was performed to compare the performance of the state estimators designed within the scope of this thesis. The time between the start and end of the simulation was calculated with the help of a timer to test the time cost of the filter. A total of two tests were performed: movement in one axis and movement in two axes.

Simulation 1

In this simulation,

- Movement is applied on a single axis around the pitch angle.
- Sensor data was generated at 1000 Hz.
- The total simulation time was determined as 60 seconds.
- In simulation, total processing time of 60001 data was obtained as 11.47 seconds.

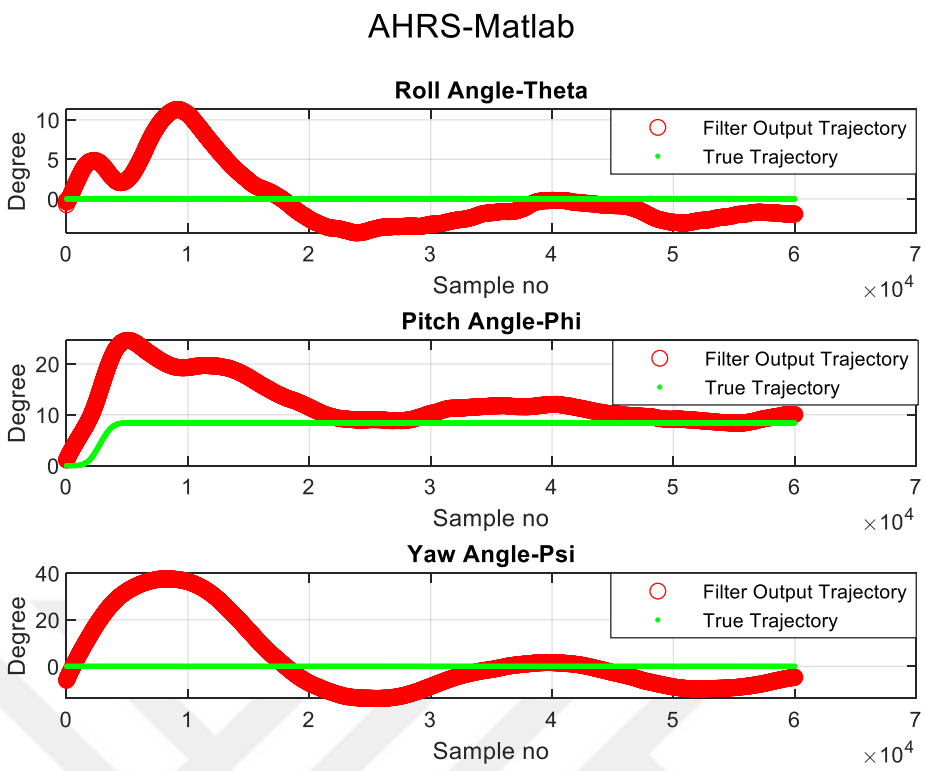


Figure 20: Single Axis Rotation Simulation Result for MATLAB-AHRS Attitude Estimator Function-Test 1

When the values obtained because of the state estimator are examined in Figure 20, it is seen that it approaches and moves away from the desired value more than once throughout the simulation. In addition, although there is no movement, large errors in the yaw angle are observed. In the last 10 seconds of the simulation, estimators give more stable values.

Table 25: Simulation Results for MATLAB-AHRS Attitude Estimator Function-Test 1

	True Value (°)	**Estimated Value After 60 Seconds (°)**	**Total RMS Error Across Entire Motion (°)**
Roll Angle	0	-1.91	3.91
Pitch Angle	8.37	10.04	6.29
Yaw Angle	0	-5.10	16.01

Table 25 is examined; the desired values achieved because of the simulation were achieved with higher performance for roll and pitch angles instead of yaw angles.

However, it was observed that the number of errors in the yaw angle was high. In addition, when the total RMSE amounts obtained throughout the entire movement are examined, it is observed that the state estimator works inefficiently.

Simulation 2

In this simulation,

- Movement is applied on two axes: around roll and pitch angle.
- Sensor data was generated at 1000 Hz.
- The total simulation time was determined as 60 seconds.
- In simulation, total processing time of 60001 data was obtained as 17.48 seconds.

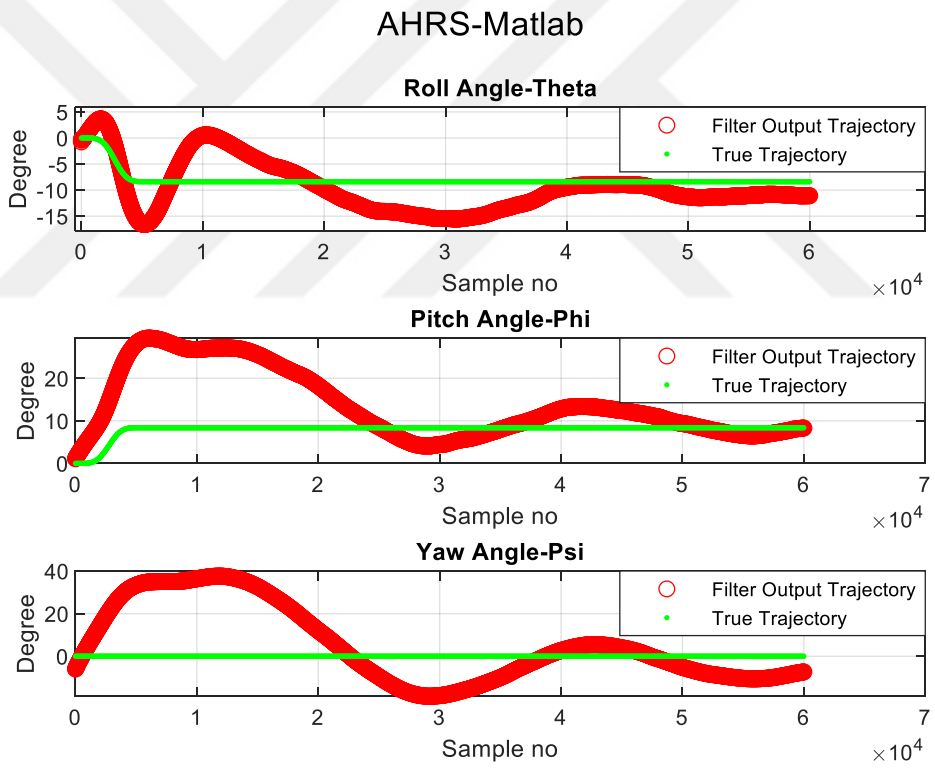


Figure 21: Dual Axis Rotation Simulation Result for MATLAB-AHRS Attitude Estimator Function-Test 2

Figure 21 shows that the oscillation continues throughout the simulation when movement is given in two axes.

Table 26: Simulation Results for MATLAB-AHRS Attitude Estimator Function-Test 2

	True Value (°)	**Estimated Value After 60 Second (°)**	**Total RMS Error Across Entire Motion (°)**
Roll Angle	-8.37	-11.09	4.44
Pitch Angle	8.37	7.95	9.77
Yaw Angle	0	-7.32	18.95

In Table 26, the desired values were achieved because the simulation achieved higher performance for roll and pitch angles instead of yaw angles. However, it was observed that the number of errors in the yaw angle was high. In addition, when the total RMSE amounts obtained throughout the entire movement are increased. Also, simulation time is increased according to Simulation 1.

5.2 ATTITUDE ESTIMATION WITH EXTENDED KALMAN FILTER

The 3-axis attitude information obtained from the accelerometer and magnetometer and the 3-axis attitude information obtained using the gyroscope are sent to the attitude estimator. While the attitude information from the gyroscope is used in the estimation step of the Kalman filter, the attitude information obtained from the accelerometer and magnetometer sensor is used in the correction step.

The designed EKF is applied to seven states. While four store attitude information as a quaternion, the remaining three are determined as offset values that eliminate the error added to the gyroscope over time. In addition, the error covariance matrices described in Section 3.1.2 were also applied to solve the problem.

Simulation 1

In this simulation,

- Movement is applied on a single axis around the pitch angle.
- Sensor data was generated at 1000 Hz for testing.
- The total simulation time was determined as 60 seconds.
- In simulation, total processing time of 60001 data was obtained as 5.48 seconds.

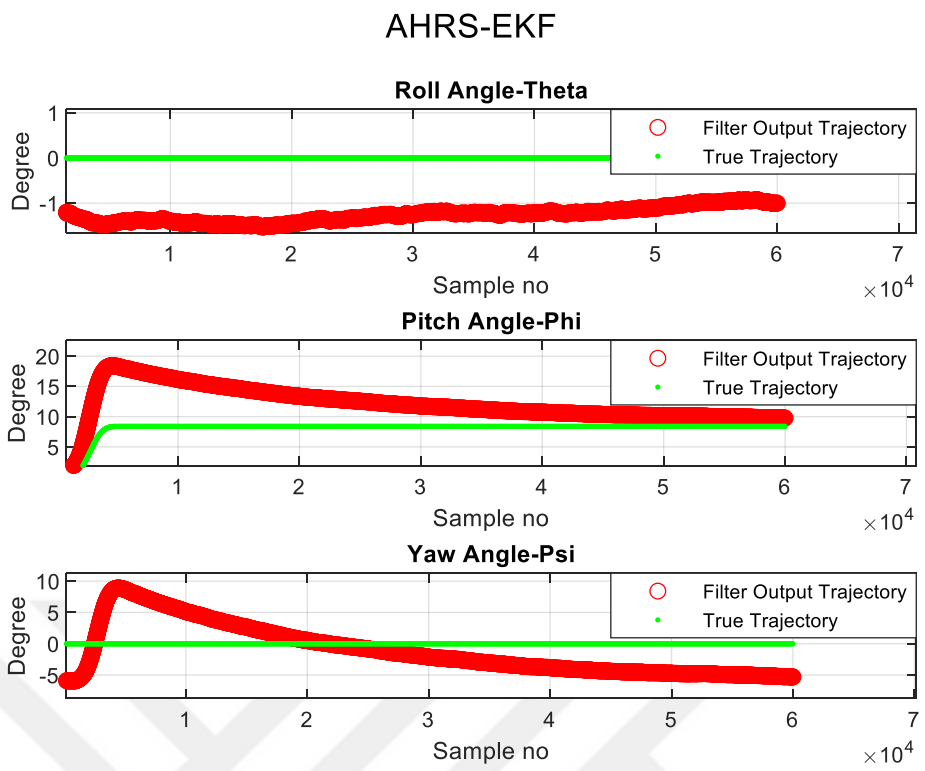


Figure 22: Single Axis Rotation Simulation Result for Designed EKF Attitude Estimator-Test 1

When the outputs of the designed filter are examined, the filter approaches the required value more stable. In addition, the data processing time is 50% shorter than the method described in Section 5.1.

Table 27: Simulation Results for Designed EKF Attitude Estimator Function-Test 1

	True Value (°)	**Estimated Value After 60 Second (°)**	**Total RMS Error Across Entire Motion (°)**
Roll Angle	0	-0.98	1.26
Pitch Angle	8.37	9.77	4.69
Yaw Angle	0	-5.30	4.20

When Table 27 is examined, the desired values were achieved because the simulation achieved higher performance for roll and pitch angles instead of yaw angles. However, it was observed that the number of errors in the yaw angle was high.

In addition, when the total RMSE amounts obtained throughout the entire movement are examined, it is more stable than the method described in Section 5.1.

Simulation 2

In this simulation,

- Movement is applied on two axes: around roll and pitch angle.
- Sensor data was generated at 1000 Hz.
- The total simulation time was determined as 60 seconds.
- In simulation, total processing time of 60001 data was obtained as 7.11 seconds.

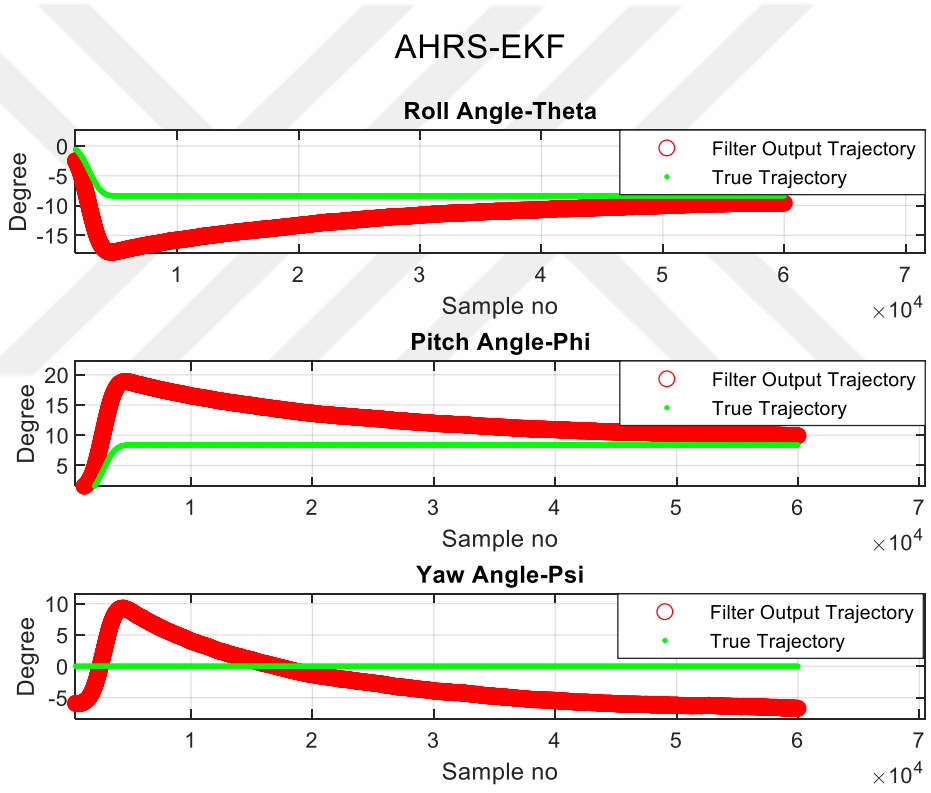


Figure 23: Dual Axis Rotation Simulation Result for Designed EKF Attitude Estimator-Test 2

When the filter outputs are examined, it is seen that the filter still works stable, and RMSE values are still reliable in two-axis simulation. In addition, the data processing time is not increased as the method mentioned in Section 5.1 Simulation 2.

Table 28: Simulation Results for Designed EKF Attitude Estimator Function-Test 2

	True Value (°)	**Estimated Value After 60 Second (°)**	**Total RMS Error Across Entire Motion (°)**
Roll Angle	-8.37	-9.59	4.49
Pitch Angle	8.37	9.35	4.93
Yaw Angle	0	-6.52	5.01

5.3 ATTITUDE ESTIMATION WITH UNSCENTED KALMAN FILTER

The designed UKF is applied to seven states. While four store attitude information as a quaternion, the remaining three are determined as offset values that eliminate the error added to the gyroscope over time. In addition, the error covariance matrices described in Section 3.1.2 were also applied to solve the problem.

Simulation 1

In this simulation,

- Movement is applied on the single axis around the pitch angle.
- Sensor data was generated at 1000 Hz for testing.
- The total simulation time was determined as 60 seconds.
- In simulation, total processing time of 60001 data was obtained as 62.87 seconds.

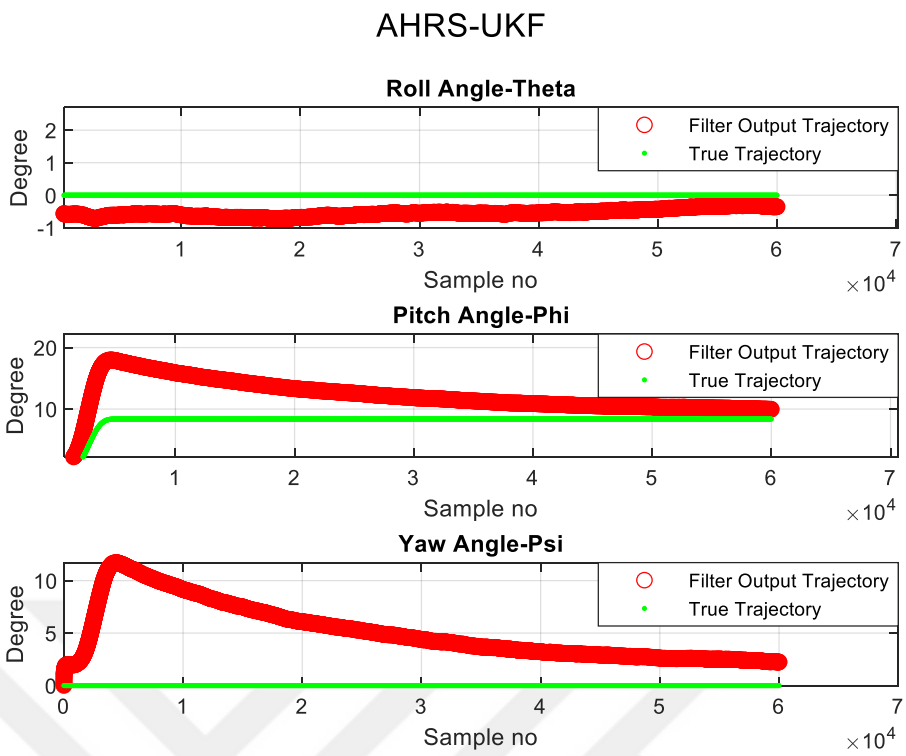


Figure 24: Single Axis Rotation Simulation Result for Designed UKF Attitude Estimator-Test 1

When the outputs of the designed filter are examined in Figure 24, the filter approaches the required value more stable. However, the data processing time is much longer than the method described in Section 5.1 and Section 5.2.

Table 29: Simulation Results for Designed UKF Attitude Estimator Function-Test 1

	True Value (°)	**Estimated Value After 60 Second (°)**	**Total RMS Error Across Entire Motion (°)**
Roll Angle	0	-0.35	0.55
Pitch Angle	8.37	9.96	4.63
Yaw Angle	0	1.24	4.85

When Table 29 is examined, the desired values were achieved with high accuracy. In addition, the high error in yaw angle observed in Section 5.1 and Section 5.2 is not observed in the UKF outputs. Also, the total RMSE is smaller than the EKF.

Simulation 2

In this simulation,

- Movement is applied on two axes around the roll and pitch angles.
- Sensor data was generated at 1000 Hz.
- The total simulation time was determined as 60 seconds.
- In simulation, total processing time of 60001 data was obtained as 73.52 seconds.

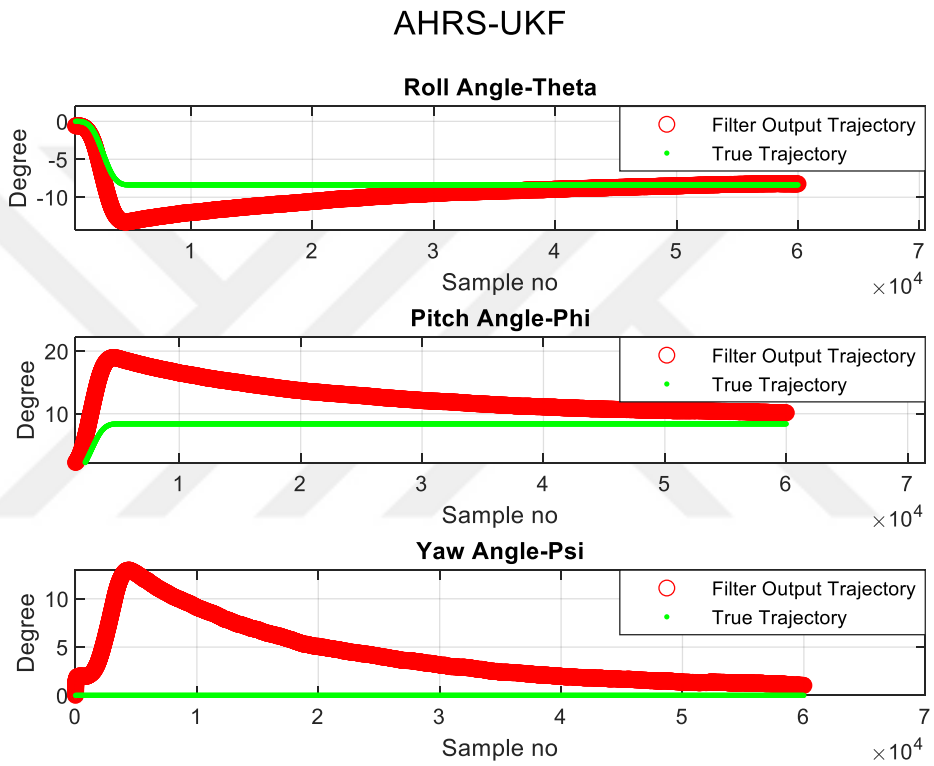


Figure 25: Dual Axis Rotation Simulation Result for Designed UKF Attitude Estimator-Test 2

When the filter outputs are examined, it is seen that the filter still works stable, and RMSE values are still reliable in two-axis simulation. In addition, the final values after simulation are almost the same as the required values.

Table 30: Simulation Results for Designed UKF Attitude Estimator Function-Test 2

	True Value (º)	**Estimated Value After 60 Second (º)**	**Total RMS Error Across Entire Motion (º)**
Roll Angle	-8.37	-8.37	2.07
Pitch Angle	8.37	10.14	5.01
Yaw Angle	0	0.29	4.54

In Table 30, the desired values were achieved at the end of the simulation with high accuracy in three axes. Also, the total RMSE is stable and smaller than the EKF. The major drawback of the UKF is the use of more resources because of their computational complexity.

5.4 ATTITUDE ESTIMATION WITH COMPLEMENTARY FILTER

The 3-axis attitude information obtained from the accelerometer and magnetometer and the 3-axis attitude information obtained using the gyroscope are sent to the attitude estimator. Attitude information from the gyroscope and attitude information obtained from the accelerometer and magnetometer sensor are fused with a gain value with interval [0,1].

Simulation 1

In this simulation,

- Movement is applied on the single axis around the roll angle.
- Sensor data was generated at 1000 Hz for testing.
- The total simulation time was determined as 60 seconds.
- In simulation, total processing time of 60001 data was obtained as 2.11 seconds.

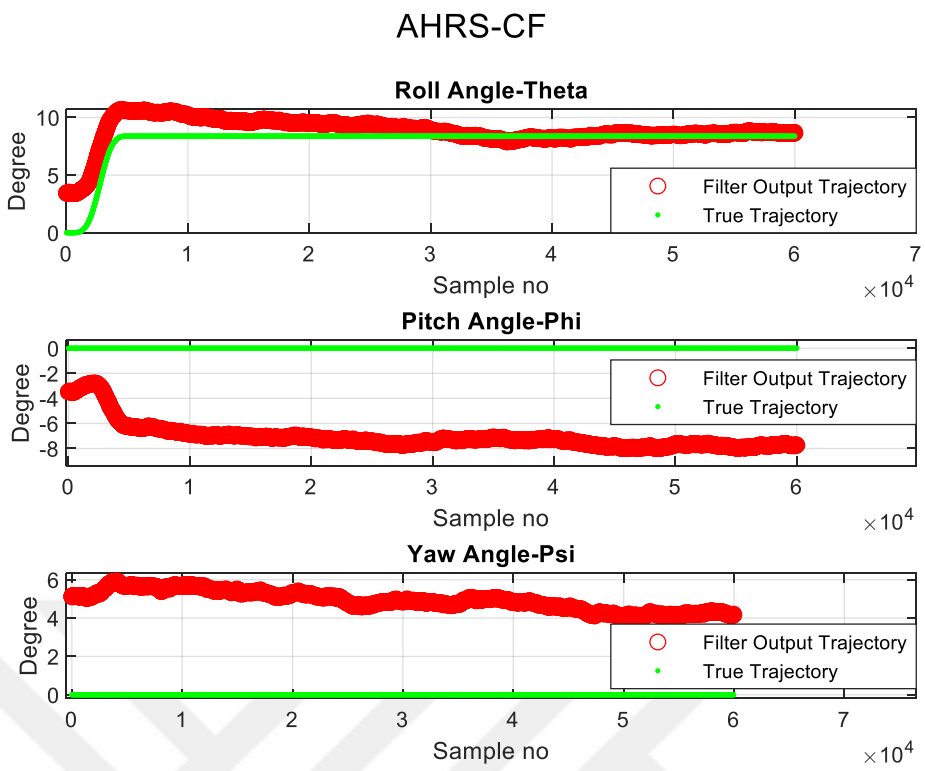


Figure 26: Single Axis Rotation Simulation Result for Designed CF Attitude Estimator-Test 1

When the values obtained from the state estimator are examined in Figure 26, the roll angle approaches smoothly throughout the simulation. In addition, pitch and yaw angles do not converge with the true value or roll angle.

The advantage of this method is processing time.CF is much faster than other estimators because of their simple algorithms

Table 31: Simulation Results for Designed CF Attitude Estimator Function-Test 1

	True Value (°)	**Estimated Value After 60 Second (°)**	**Total RMS Error Across Entire Motion (°)**
Roll Angle	8.37	8.67	1.27
Pitch Angle	0	-7.27	7.16
Yaw Angle	0	3.15	3.94

Table 31 is examined; the desired values achieved because of the simulation were achieved with higher performance for roll angle instead of pitch and yaw angle. In addition, when the total RMSE amounts obtained throughout the entire movement are examined, it is observed that the state estimator works inefficiently.

Simulation 2

In this simulation,

- Movement is applied on two axes: around roll and pitch angle.
- Sensor data was generated at 1000 Hz.
- The total simulation time was determined as 60 seconds.
- In simulation, total processing time of 60001 data was obtained as 2.13 seconds.

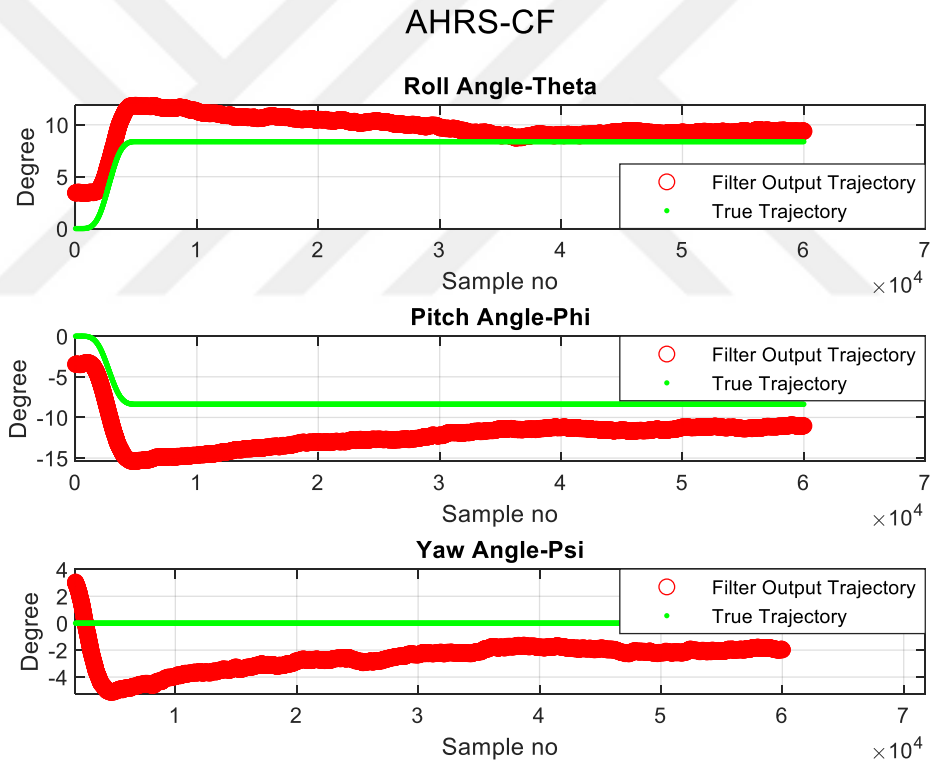


Figure 27: Dual Axis Rotation Simulation Result for Designed CF Attitude Estimator-Test 2

In Figure 27, it is seen that the roll angle approaches smoothly throughout the simulation. However, errors are more unstable on the axes of movement.

Table 32: Simulation Results for Designed CF Attitude Estimator Function-Test 2

	True Value (°)	**Estimated Value After 60 Second (°)**	**Total RMS Error Across Entire Motion (°)**
Roll Angle	8.37	9.34	1.88
Pitch Angle	-8.37	-11.03	4.30
Yaw Angle	0	-1.95	2.82

5.5 SIMULATION RESULTS

Within the scope of this thesis, four different attitude estimator models were studied. While one of them was used in MATLAB, "AHRS Function" as a reference for the study, the designs of the remaining three were developed within the scope of this study.

Single Axis Movement

Table 33: Single Axis Movement Simulation Results for MATLAB-EKF Attitude Estimator Results

MATLAB-EKF Results	**True Value**	**Est. Value**	**Total RMSE**
Roll Angle (°)	0	-1.91	3.91
Pitch Angle (°)	8.37	10.04	6.29
Yaw Angle (°)	0	-5.10	16.01

Table 34: Single Axis Movement Simulation Results for Designed EKF, UKF and CF Attitude Estimator Results

		EKF		***UKF***		***CF***	
	True Value	**Est. Value**	**Total RMSE**	**Est. Value**	**Total RMSE**	**Est. Value**	**Total RMSE**
Roll Angle (°)	0	-0.98	1.26	-0.35	0.55	8.67	1.27
Pitch Angle (°)	8.37	9.77	4.69	9.96	4.63	-7.27	7.16

Table 34 Continued

Yaw Angle (°)	0	-5.30	4.20	1.24	4.85	3.15	3.94

*Est. Value = Estimated Value

Table 34 shows,

- UKF is more accurate than other methods regarding the final estimated value. Also, the total RMSE along the simulation is smaller than that of other estimators.
- EKF is almost as accurate as UKF regarding axis movement—however, other axes estimated value errors to be higher. Also, the total RMSE along the simulation is almost the same as UKF.
- CF has more errors, according to other estimators, in estimated value and total RMSE.

Multiple Axis Movement

Table 35: Multiple Axis Movement Simulation Results for MATLAB-EKF Attitude Estimator Results

MATLAB-EKF Results	**True Value**	**Est. Value**	**Total RMSE**
Roll Angle (°)	-8.37	-11.09	4.44
Pitch Angle (°)	8.37	7.95	9.7
Yaw Angle (°)	0	-7.32	18.95

Table 36: Multiple Axis Movement Simulation Results for Designed EKF, UKF and CF Attitude Estimator Results

		EKF		***UKF***		***CF***	
	True Value	**Est. Value**	**Total RMSE**	**Est. Value**	**Total RMSE**	**Est. Value**	**Total RMSE**
Roll Angle (°)	-9.59	-9.59	4.49	-8.37	2.07	9.34	1.88
Pitch Angle (°)	9.35	9.35	4.93	10.14	5.01	-11.03	4.30

Table 36 Continued

Yaw Angle (°)	-6.52	-6.52	5.01	0.29	4.54	-1.95	2.82

Table 36 shows,

- UKF is more accurate than other methods regarding the final estimated value. Also, the total RMSE along the simulation is smaller than that of other estimators.
- EKF is almost as accurate as UKF regarding axis movement—however, other axes estimated value errors to be higher. Also, the total RMSE along the simulation is almost the same as UKF.
- CF has more errors, according to other estimators, in estimated value and total RMSE.

Process Time Comparison

Process times are examined for the appropriate filter selection. Processing time values in Table 37 are for single-axis movement.

Table 37: Process Time Comparison for Different Attitude Estimators

	MATLAB-EKF	***UKF***	**EKF**	**CF**
Process Time	11.47s	62.87s	5.48s	2.11s

Table 37 shows that UKF is slower than other estimators. CF is faster than other estimators.

CHAPTER VI
AHRS EXPERIMENTAL SYSTEM SETUP

6.1 HARDWARE DESIGN

To work with real sensors and achieve a more realistic system design, in addition to sensor modeling in MATLAB/SIMULINK, a printed circuit board (PCB) has been designed using Altium Designer.

6.1.1 System Components

6.1.1.1 Accelerometer

Analog Device - ADXL355B sensor is a 3-axis high-accuracy accelerometer. After the sensor calibration process mentioned in Section 4.1, static roll and pitch angles RMSE are under 0.1 °.

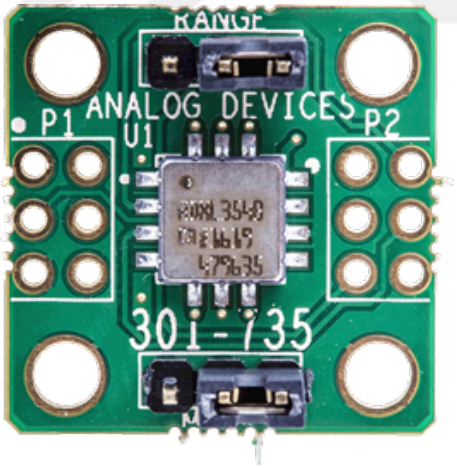


Figure 28: ADXL355 3-Axis Accelerometer Sensor

Specifications

Table 38: ADXL355 Specifications

	Values	**Units**
Output Full-Scale Range	±2, ±4, ±8	g
X, Y and Z-Axis Sensitivity	235,520 256,000 276,480 at +-2g	LSB/g
	117,760 128,000 138,240 at +-4g	
Cross Axis Sensitivity	1	%
Sensitivity Change due to Temperature	±0.01 btw −40°C to +125°C	%/°C
X, Y and Z-Axis 0 g Output	−75 ±25 +75	mg
NOISE (Spectral Density)	22.5 at ±2 g	µg/√Hz
Velocity Random Walk	5.3 in X-Y-Axis at 2g	mm/sec/√Hr
	7.7 in Z-Axis at 2g	
Price	$35	

6.1.1.2 Magnetometer

PNI - RM3100 sensor used as a magnetometer. RM3100 is a 3-axis high-accuracy magnetometer. Getting accurate navigation and position data depends on having accurate magnetic field measurements to begin with. Unlike traditional Hall-effect sensors, the RM3100 magneto-inductive sensor has over 23 times better resolution and 33 times less noise, resulting in far more precise data outputs [36]. After the sensor calibration process mentioned in Section 4.3, static yaw angles RMSE are under 0.1 °.

Figure 29: PNI-RM3100 3 Axis Magnetometer Sensor

Specifications

Table 39: RM3100 Specifications

	Values	**Units**
Sensitivity	50 26 13	nT
Resolution	40	µgauss
Field Range	± 11	gauss
Noise	30 20 15	nT
Price	$25	

6.1.1.3 Gyroscope

STMicroelectronics - A3G4250D sensor used is as a gyroscope. The A3G4250D is a low-power 3-axis angular rate sensor that provides unprecedented stability at zero-rate levels and sensitivity over temperature and time. It includes a sensing element and an IC interface capable of providing the measured angular rate to the external world through a standard SPI digital interface. An I2C-compatible interface is also available [37].

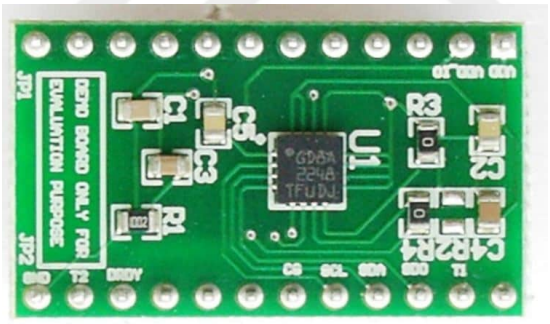


Figure 30: STMicroelectronics - A3G4250D 3-Axis Gyroscope Sensor

Specifications

Table 40: A3G4250D Specifications

	Values	**Units**
Dynamic Range	±245	°/sec
Initial Sensitivity	0.005	°/sec/LSB
Misalignment	±0.05	Degree
In-Run Bias Stability	8	°/hr

Table 40 Continued

Bias Temperature Coefficient	0.007	°/sec/°C
Linear Acceleration Effect on Bias	0.01	°/sec/g
Output Noise	0.075	°/sec rms
Sensor Resonant Frequency	65	kHz
Price	$15	

6.1.1.4 Processor

STMicroelectronics – STM32F411RCT6 chip used as a microcontroller. The STM32F411xC/xE devices utilize the high-performance Arm Cortex-M4 32-bit RISC core, capable of operating up to 100 MHz. This core includes a floating-point unit (FPU) that supports Arm's single-precision data-processing instructions and data types. Additionally, it incorporates a comprehensive set of DSP instructions and a memory protection unit (MPU), enhancing application security by controlling access rights to memory regions.

The STM32F411xC/xE is part of the STM32 Dynamic Efficiency product line, designed to balance power efficiency, performance, and integration. It introduces an innovative feature called Batch Acquisition Mode (BAM), which further reduces power consumption by enabling data batching. This mode optimizes power usage during periods of data collection, enhancing overall energy efficiency of the device. [38].

Figure 31: STMicroelectronics – STM32F411RCT6 Microcontroller

Memories:

- 256 Kb flash memory

- 128 Kb SRAM

Core:

- Arm 32-bit Cortex-M4 CPU with FPU,
- Adaptive real-time accelerator (ART Accelerator)
- Processing frequency up to 100 MHz, memory protection unit, 125 DMIPS/1.25 DMIPS/MHz

6.2 PCB Specifications

- The dimensions of the board are 5cm x 10cm.
- The PCB is designed as a double-layer board.
- One I2C port and one SPI port have been added to enable communication when connecting external sensors.
- It is designed with 2 USART ports to allow external connections for GPS/GNSS and, in desired cases, a telemetry module.
- An SWD (Serial Wire Debugger) output with a male header connector is used to program the processor and enable debugging.
- During the design, the magnetometer sensors and the driver chip are placed away from high-current paths and other components that draw current.
- Additional capacitors are placed to prevent potential noise on the power lines.

Table 41: Cost Analysis

	Price
ADXL-355B	$35
RM3100	$25
A3G4250D	$15
STM32F411	$15
Production	$5
Other	$5
Total Cost	$100

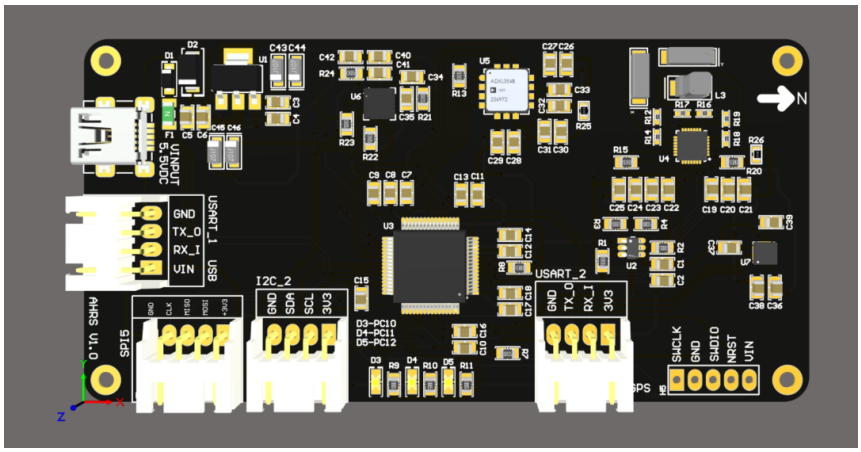


Figure 32: AHRS Printed Circuit Board 3D Model on Altium Designer

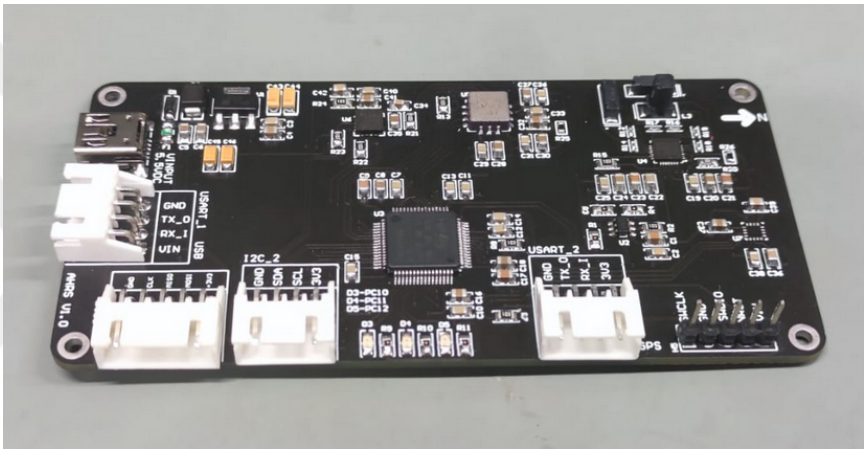


Figure 33: AHRS Printed Circuit Board

The arrow on the right side of the board specifies the system heading. When we place the AHRS system in the UAV or Autonomous Ground Vehicle (AGV), the direction of the device and the arrow mark must face the same direction.

6.3 EMBEDDED SOFTWARE DESIGN

Embedded software has been developed using the STM32 processor platform to process data from the accelerometers, magnetometers, and gyroscopes and used in the AHRS design. During the development of this software, various development tools such as STM32CubeIDE, STM32CubeProgrammer, and StmStudio have been employed.

When we want to test sensor data in the MATLAB environment or record data, this embedded software uses a predefined query-data protocol to send data to the computer environment via USART at 250Hz.

6.4 MATLAB APP DESIGN

A computer interface has been designed for this project. The aim of designing this interface is to access sensor data and test platform data easily. Also, an interface can handle showing real-time angles, soft and hard iron calibration, and sensor calibration procedures. This interface is suitable for long-term data collection. In this way, the data received from the sensor can be analyzed. The raw data sent at 250Hz in the AHRS system can produce meaningful attitude information at 150Hz. In addition, it instantly converts the raw data from the AHRS system's embedded software protocol into attitude information with the desired estimator. Also, it processes the angle information coming from the embedded software protocol of the test system as desired.

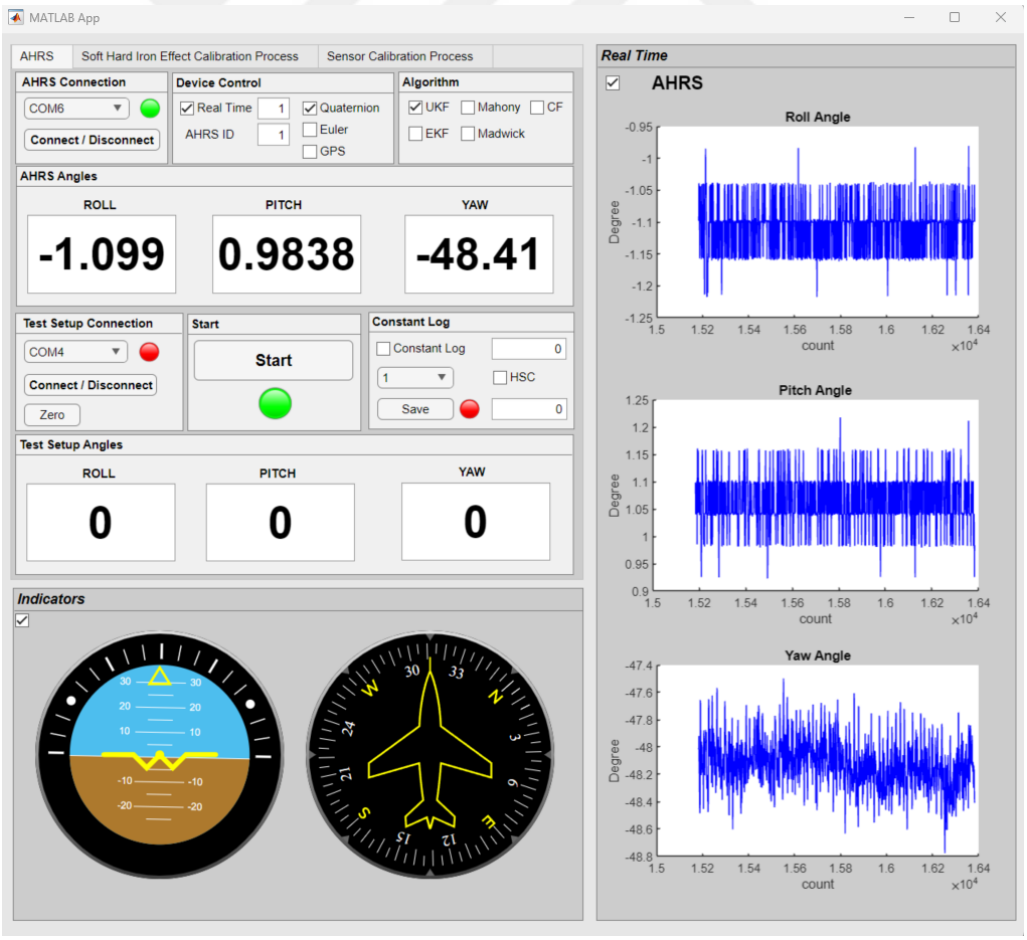


Figure 34: MATLAB Application

6.5 MECHANICAL DESIGN

A mechanical enclosure design has been created using AutoCAD Inventor to protect the designed PCB from external influences and to allow for its assembly onto a test platform. The production of the enclosure was completed using a 3D printer.
Mechanical Dimensions: 105 mm x 55 mm x 20 mm

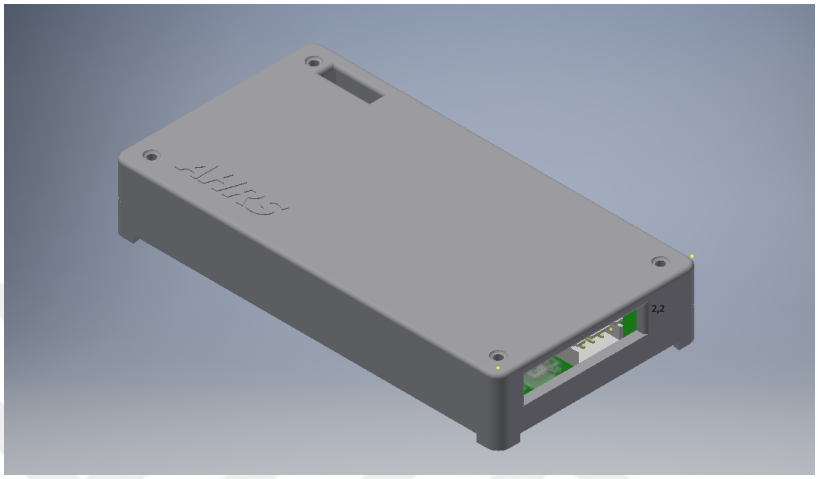


Figure 35: AHRS Printed Circuit Board Case 3D Model on AutoCAD Inventor

Figure 36: AHRS Printed Circuit Board Case

CHAPTER VII
AHRS EXPERIMENTAL SYSTEM TEST SETUP DESIGN

7.1 INTRODUCTION

A three-axis test system was designed to examine the AHRS system in real scenarios and conduct improvement studies. During the design process of this test system, a rotary wing UAV was designed to be positioned inside it. In this way, testing can be done in different environments with high vibration and magnetic interference, and a testing and development environment is provided in cleaner environments without a UAV.

In the test system's design, three high-accuracy encoders were placed at the center of each axis. To determine the accuracy values of these encoders, two different points, 180 ° apart, were determined, and measurements were taken during the test setup. As a result of these measurements, the error value of the encoders was found to be 0.3 °. Figure 37 shows the completed test platform.

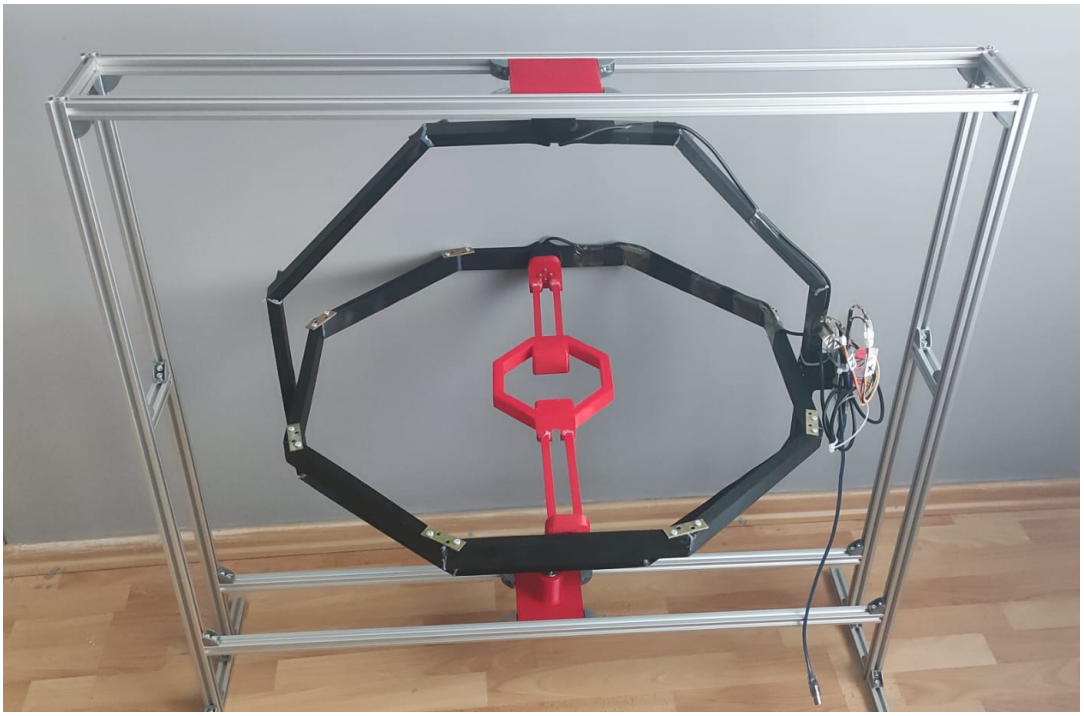


Figure 37: Designed Test Platform for AHRS System

7.2 MECHANICAL DESIGN

The design and technical drawings of the test system were completed using the AutoCAD Inventor program.

- When folded, the platform's dimensions are determined as 1000mm x 1000mm x 10mm.
- The maximum UAV size that can be placed inside the platform is 621mm x 621mm, including the wings.
- The size of the middle table designed to fix the UAV is determined to be 180mm.
- The platform can move freely in 3 axes.

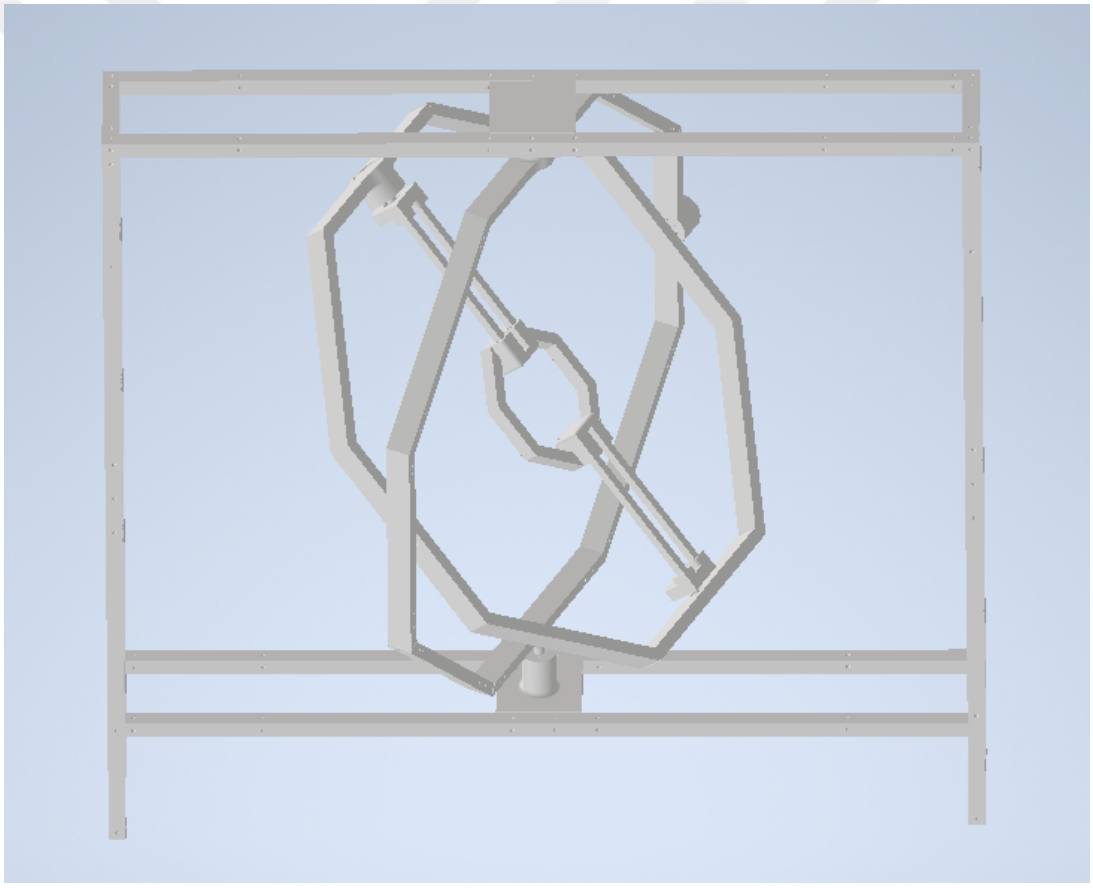


Figure 38: Test Platform 3D Model in AutoCAD Inventor

In Figure 39, the AHRS is positioned on the test platform. In this scenario, the AHRS card is powered by an external battery. Additionally, data is transferred via the wireless communication module. The reason for testing without an aircraft on the

platform is that the necessary calibrations are made and the tests are carried out in a relatively cleaner environment.

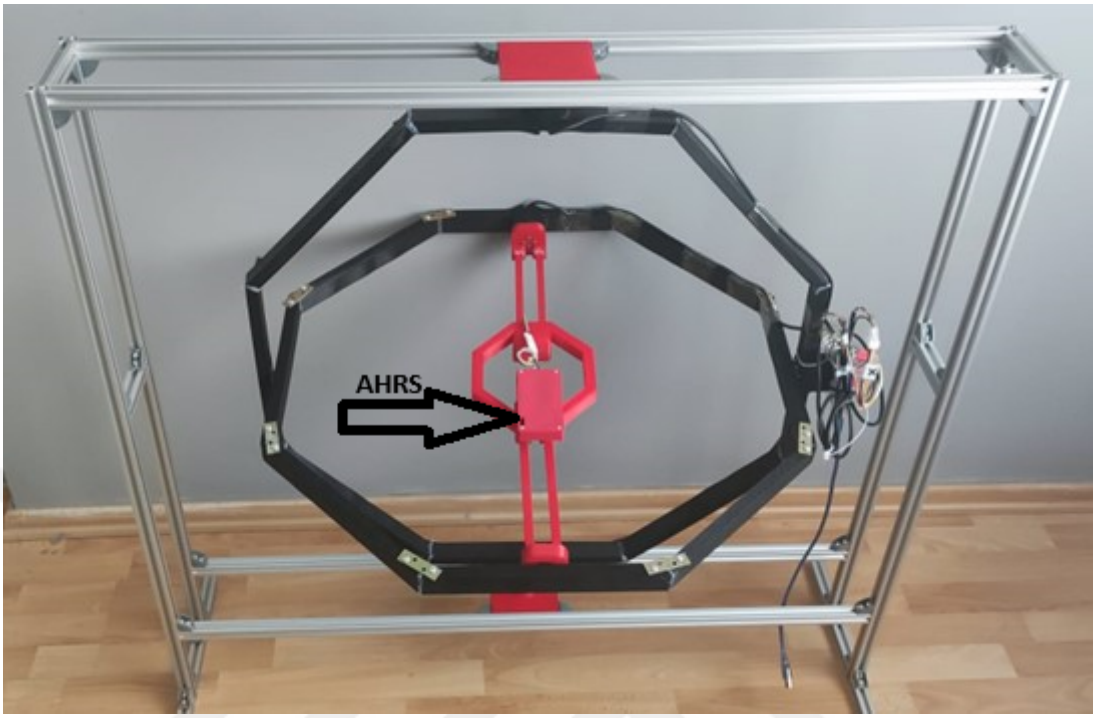


Figure 39: Test Platform with Only AHRS System

In Figure 40, AHRS is positioned on the UAV. In this scenario, the UAV is powered separately. An external power supply powers the AHRS. The main difficulty in this scenario is that the system is subject to high vibration and highly magnetically distorted.

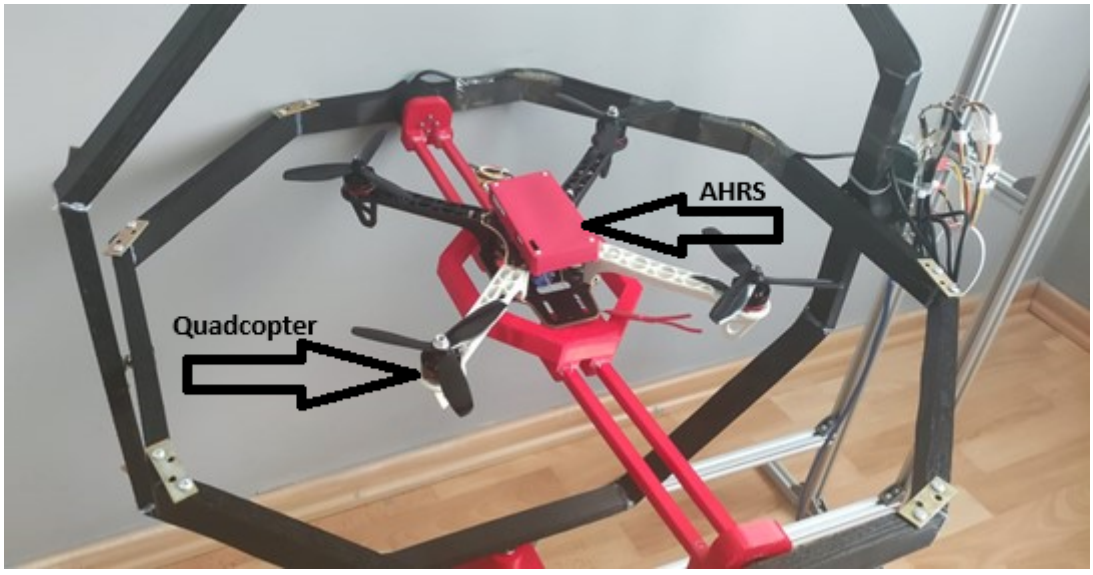


Figure 40: Quadcopter Carrying AHRS Fixed on Test Platform-Detailed

Figure 41 shows the control unit on the test platform, which processes encoder data and sends it via UART to the interface designed in MATLAB.

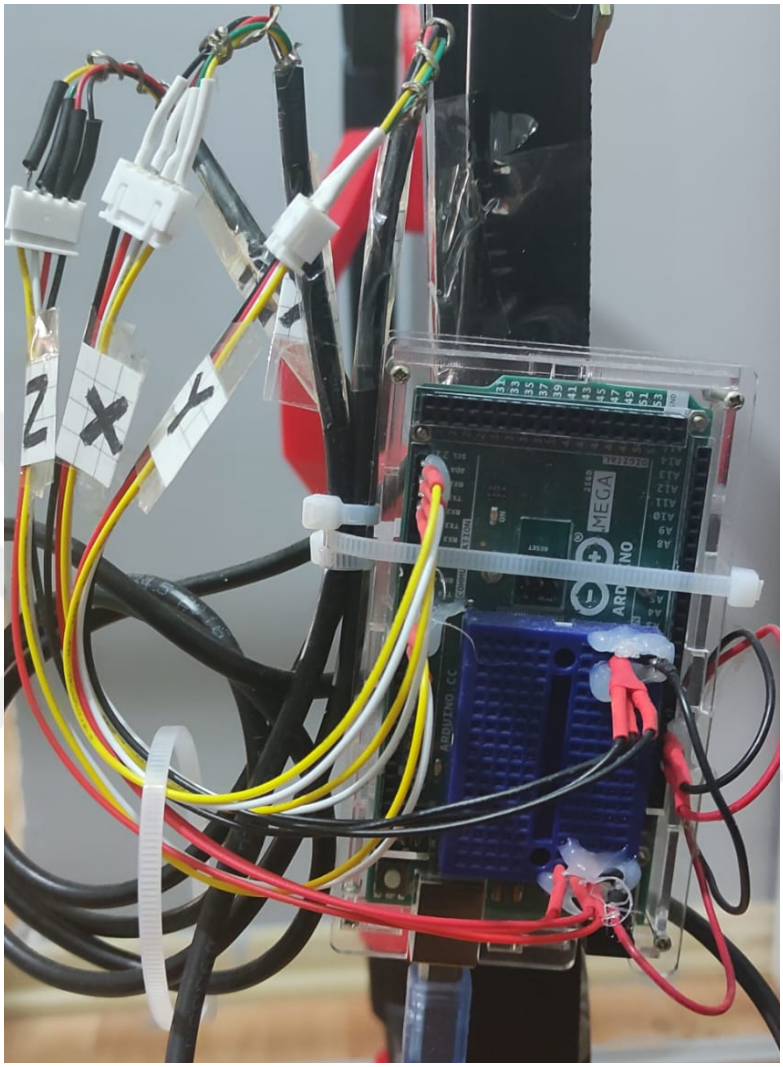


Figure 41: Test Platform Control Unit

7.3 HARDWARE DESIGN

7.3.1 Microcontroller

Arduino Mega[39] was used to control the system. The data of three encoders are read. Interrupt pins are used to collect data from all encoders simultaneously. The reason Arduino Mega is preferred is that it has 6 interrupt pins. With the help of these pins, encoder reading can be done without data loss.

Figure 42: Arduino Mega

7.3.2 Encoder

E6B2-CWZ6C 2000P/R 2M encoder [40] was used to collect real-time angle data. When the accuracy of the encoders is tested, an error value of 0.3 ° is obtained. This encoder is preferred because it has a more precise angle measurement ability with 2000 pulses per revolution. In addition, this encoder performs counting with the help of two channels with 90 degrees between phases. In this way, a more precise measurement can be achieved.

Figure 43: E6B2-CWZ6C 2000P/R 2M Encoder

7.3.3 Communication Module

HC-12 wireless communication module[41] was used to prevent cable confusion while receiving data on the platform. It makes working easier, especially in test scenarios with high-motion input.

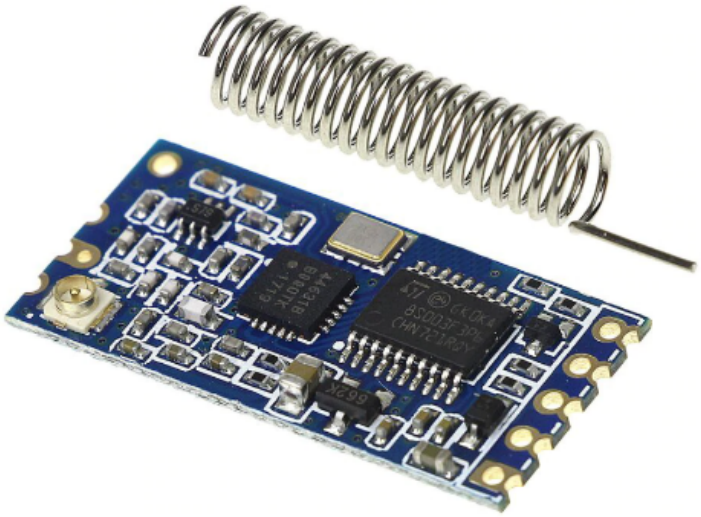


Figure 44: HC-12 Wireless Communication Module

As seen in the figure below, one communication module is placed on the USB-TTL module. In contrast, the other communication module is connected to the USART connector of the designed prototype AHRS card.

Figure 45: Wireless Communication Setup for Test Platform with AHRS System

7.4 SOFTWARE DESIGN

It aims to determine how much rotation movement is made with the help of 3 encoders located on the shafts of the designed test system. For this purpose, the data received from 2 separate channels of the encoder were classified as forward and reverse with the help of a counter. While this data was being received, it was collected precisely with the help of 6 interrupt functions, 2 for each encoder, so it only increases when there is movement. These data are given meaning with the help of the counted value in one full rotation. The operations described to make sense of this data and convert it into angles were carried out on Arduino. In addition, delta roll, delta pitch and delta yaw angles calculated at 200 Hz are sent to the outside world with the help of a protocol along with a timestamp via serial communication on Arduino.

CHAPTER VIII
PERFORMING REAL SYSTEM TESTS AND ANALYSIS OF TEST RESULTS

8.1 PERFORMANCE TESTS

The test system developed to test the designed AHRS system in real environments is mentioned in Chapter 7. In this section, verification tests of the designed AHRS system were carried out with the help of this test system.

A total of 10 tests were carried out in 7 different test scenarios to perform verifications. These tests are defined as only roll angle, only pitch angle, only yaw angle, both pitch and roll angle, both pitch and yaw angle, both roll and yaw angle and roll, pitch and yaw angle. While testing in these scenarios, the AHRS system is fixed to the test platform. Then, movement input was given to the system in predetermined scenarios. In addition, data from both the test system and the AHRS system are processed simultaneously with the application designed in the MATLAB environment and the roll, pitch and yaw angle are achieved. While processing this data, the UKF filter was used as a state estimator. The reason for this is that the UKF filter reached higher accuracy values compared to other filters in the studies carried out in Chapter 5.

Sensor calibrations of all sensors were performed to obtain accurate attitude information. In addition, soft and hard iron calibration was carried out to remove the magnetic effects affecting the magnetometer. Random rotation movements were given to the test platform with AHRS fixed in three axes to perform this calibration. With the help of the collected data, soft and hard effect calibration was completed, as described in Chapter 4.3.3.

In the AHRS system, the measurement range of roll and yaw angles is between ±180 º, and the measurement range of the pitch angle is between ± 90 º. The measurement range of the test system is between 0 º and 360 º for roll, pitch and yaw angle.

In this study, the angle values taken from the test system are accepted as correct values. In addition, the angle is defined as the correct angle value from the test system and the delta angle between the starting and ending angles.

In the studies conducted in this chapter, the dataset referred to as "True Trajectory" consists of real-time data read from the encoder sensor and the dataset referred to as "Filter Output Trajectory" consists of real-time data obtained from the AHRS.

8.1.1 Only Roll Angle Rotation Tests

8.1.1.1 Test-1

Test conditions were determined as follows;

- Motion input is given only in roll angle
- Data was collected at 150Hz

Figure 46 shows average oscillations of up to 0.5 º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data.

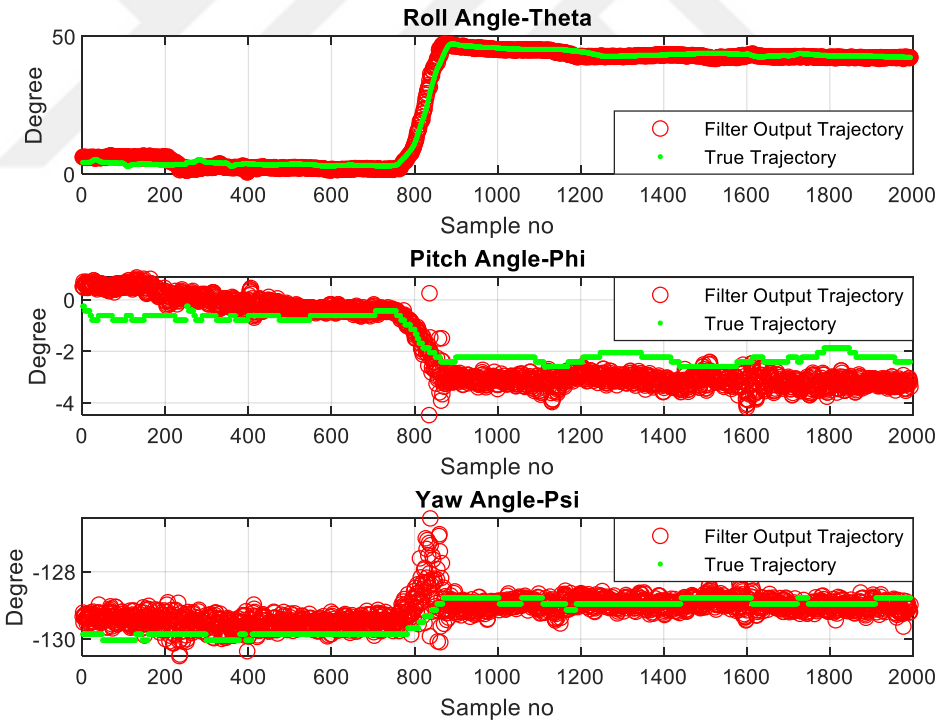


Figure 46: Encoder Data Along Roll Angle Rotation Movement

Table 42 shows each axis's exact delta rotation values for true trajectory and filter output. Even though rotation is given only in the roll angle, some small rotation values are observed in other axes.

Table 42: Encoder and UKF Delta Angles and Errors for Roll Angle Rotation

	Encoder (º)			UKF (º)			Error (º)
	Start	End	Δ	Start	End	Δ	
Roll	2.97	42.57	39.6	2.19	42.20	40.01	0.41
Pitch	-0.78	-2.41	1.63	-0.34	-2.92	2.58	0.95
Yaw	-129.85	-128.77	1.08	-129.53	-128.95	0.58	0.50

Table 42 shows the exact delta rotation-angle error values of each axis. The error in the roll angle given the rotation was 0.41 º. The error in pitch angle is 0.95 º, and the error in yaw angle is 0.5 º. The error amount in all axes remains below 1 º.

8.1.1.2 Test-2

Test conditions are the same with Test-1. Figure 47 shows average oscillations of up to 0.5 º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data except pitch angle. In pitch angle, delta-rotation error is bigger than other axes.

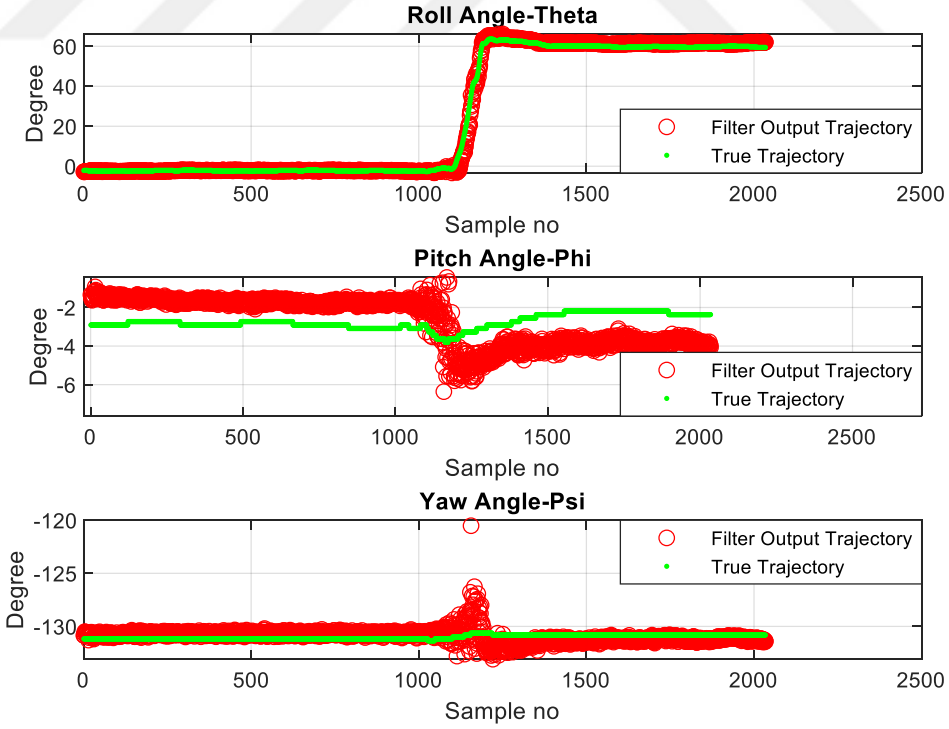


Figure 47: Encoder Data Along Roll Angle Rotation Movement

Table 43 shows each axis's exact delta rotation values for true trajectory and filter output. Even though rotation is given only in the roll angle, some small rotation values are observed in other axes.

Table 43: Encoder and UKF Delta Angles and Errors for Roll Angle Rotation

	Encoder (º)			**UKF (º)**			**Error (º)**
	Start	End	Δ	Start	End	Δ	
Roll	-2.12	60.15	62.27	-2.31	60.98	63.29	0.98
Pitch	-2.90	-2.18	0.72	-1.85	-3.45	1.6	0.88
Yaw	-131.17	-130.81	0.36	-130.69	-131.25	0.56	0.20

Table 43 shows the exact delta rotation-angle error values of each axis. The error in the roll angle, given the rotation, was 0.98 º. The error in pitch angle is 0.88 º and the error in yaw angle is 0.2 º. The error amount in all axes remains below 1 º.

In all test results obtained from the system where only roll motion is given as input, the attitude error is observed to be close to 1 º but remains less than 1 º.

8.1.2 Only Pitch Angle Rotation Tests

8.1.2.1 Test-1

Test conditions were determined as follows;

- Motion input is given only in pitch angle
- Data was collected at 150Hz

Figure 48 shows average oscillations of up to 0.5 º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data.

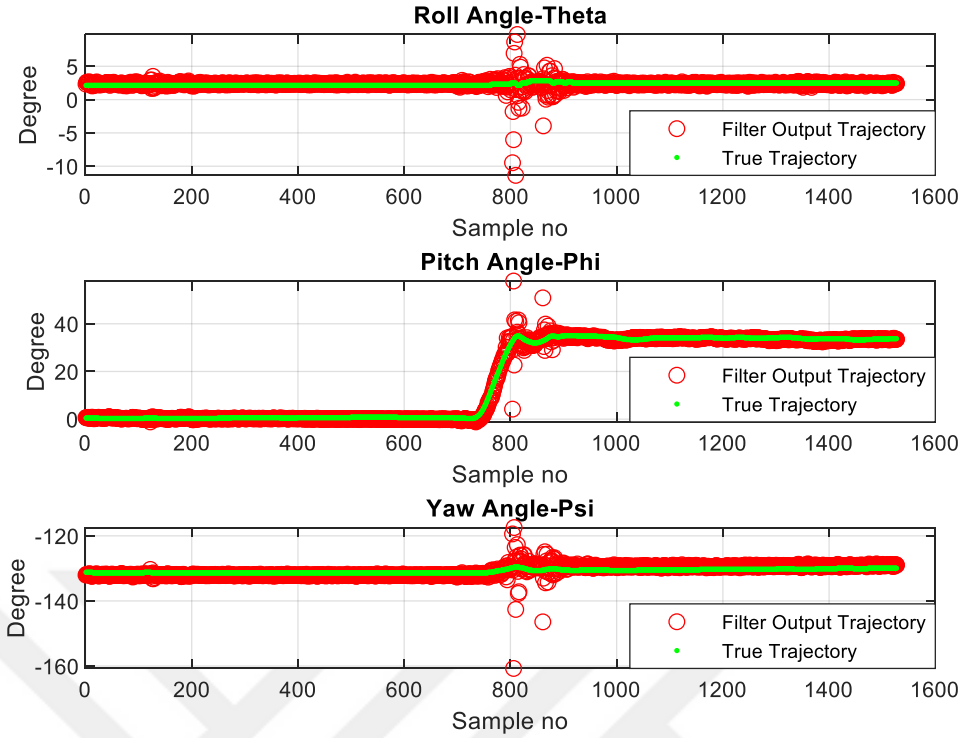


Figure 48: Encoder Data Along Pitch Angle Rotation Movement

When examining Figure 48, high oscillations are observed at the points where the movement starts. These oscillations need to be filtered out using a digital filter.

Table 44 shows each axis's exact delta rotation values for true trajectory and filter output. Even though rotation is given only in the pitch angle, some small rotation values are observed in other axes.

Table 44: Encoder and UKF Delta Angles and Errors for Pitch Angle Rotation

	Encoder (º)			**UKF (º)**			**Error (º)**
	Start	End	Δ	Start	End	Δ	
Roll	2.12	2.48	0.36	2.44	2.29	0.15	0.21
Pitch	0.28	33.40	33.12	0.57	33.77	33.20	0.08
Yaw	-131.38	-129.94	1.44	-131.94	-129.23	2.71	1.27

Table 44 shows the exact delta rotation-angle error values of each axis. The error in the pitch angle, given the rotation, was 0.08 º. The error in roll angle is 0.21 º and the error in yaw angle is 1.27 º. It is observed that the yaw angle error value is

above 1 º despite no motion input being given. However, the pitch angle error value, for which motion input is given, is below 1 º.

8.1.2.2 Test-2

Test conditions are the same with Test-1. Figure 49 shows average oscillations of up to 0.5 º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data except for the yaw angle.

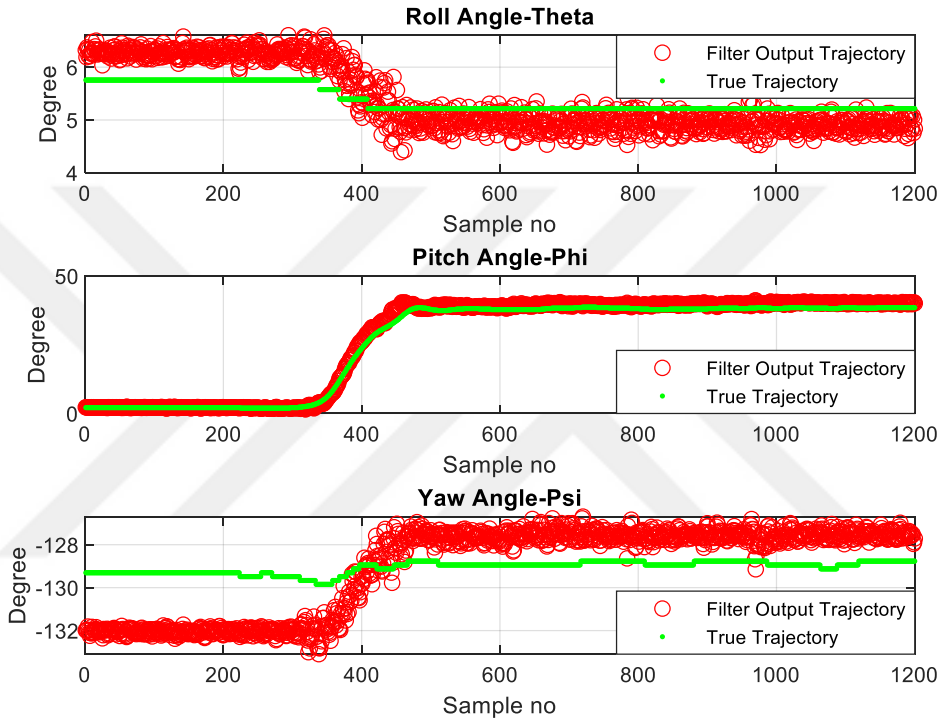


Figure 49: Encoder Data Along Pitch Angle Rotation Movement

Table 45 shows each axis's exact delta rotation values for true trajectory and filter output. Even though rotation is given only in the pitch angle, some small rotation values are observed in other axes.

Table 45: Encoder and UKF Delta Angles and Errors for Pitch Angle Rotation

	Encoder (º)			**UKF (º)**			**Error (º)**
	Start	End	Δ	Start	End	Δ	
Roll	5.75	5.21	0.54	6.22	5.01	1.21	0.67
Pitch	2.12	38.48	36.36	2.26	39.78	37.52	1.16

Table 45 Continued

Yaw	-129.30	-128.76	0.54	-132.08	-127.75	4.33	3.79

Table 45 shows the exact delta rotation-angle error values of each axis. The error in the pitch angle, given the rotation, was 1.16 º. The error in roll angle is 0.67 º and the error in yaw angle is 3.79 º.

Yaw angle errors of up to 4 º are observed despite no motion input. This error is due to the high-distortion magnetic environment.

8.1.3 Only Yaw Angle Rotation Tests

8.1.3.1 Test-1

Test conditions were determined as follows;

- Motion input is given only in the yaw angle
- Data was collected at 150Hz

Figure 50 shows average oscillations of up to 0.5 º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data.

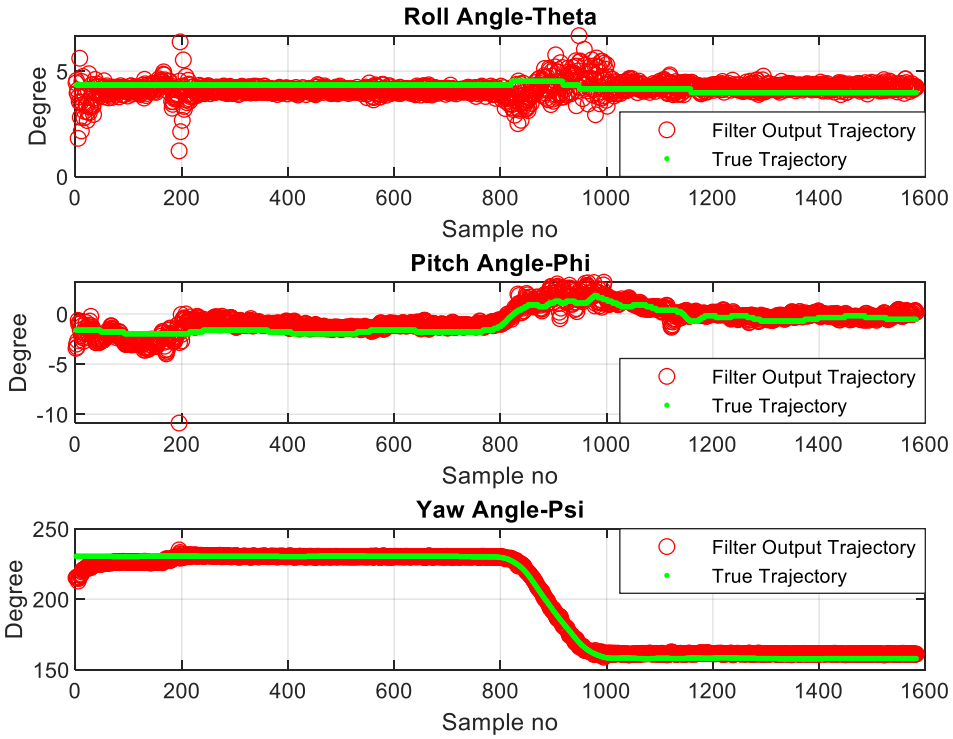


Figure 50: Encoder Data Along Pitch Angle Rotation Movement

When examining Figure 50, high oscillations are observed at the points where the movement starts. These oscillations need to be filtered out using a digital filter.

Table 46 shows each axis's exact delta rotation values for true trajectory and filter output. Even though rotation is given only in the yaw angle, some small rotation values are observed in other axes.

Table 46: Encoder and UKF Delta Angles and Errors for Yaw Angle Rotation

	Encoder (º)			**UKF (º)**			**Error (º)**
	Start	End	Δ	Start	End	Δ	
Roll	4.33	3.97	0.36	4.07	4.29	0.22	0.14
Pitch	-1.97	-0.71	1.26	-1.28	-0.17	1.11	0.15
Yaw	230.22	157.86	72.36	230.41	161.14	69.27	3.09

Table 46 shows each axis's exact delta rotation-angle error values. Given the rotation, the error in the yaw angle was 3.09 º. The error in roll angle is 0.14 º and the error in pitch angle is 0.15 º. Yaw angle errors of up to 3 º are observed. This error is due to the high-distortion magnetic environment.

8.1.3.2 Test-2

Test conditions are the same with Test-1. Figure 51 shows average oscillations of up to 0.7 º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data.

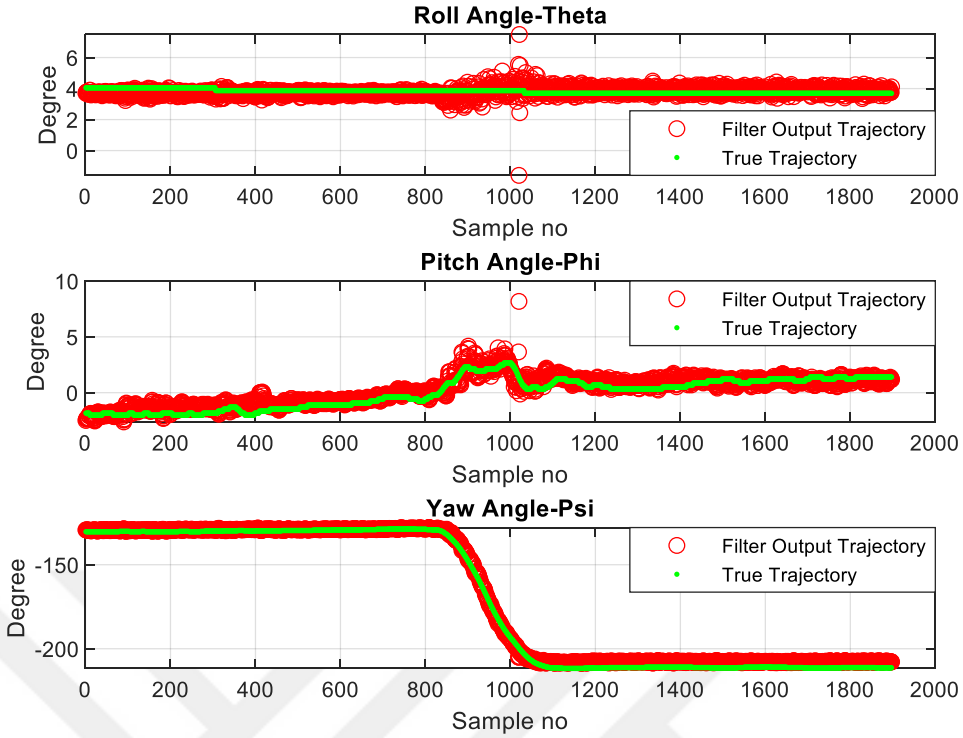


Figure 51: Encoder Data Along Yaw Angle Rotation Movement

Table 77 shows each axis's exact delta rotation values for true trajectory and filter output. Even though rotation is given only in the yaw angle, some small rotation values are observed in other axes.

Table 47: Encoder and UKF Delta Angles and Errors for Yaw Angle Rotation

	Encoder (º)			**UKF (º)**			**Error (º)**
	Start	End	Δ	Start	End	Δ	
Roll	4.05	3.69	0.36	3.94	3.86	0.08	0.28
Pitch	-1.83	1.40	0.43	-1.74	1.22	0.52	0.09
Yaw	-130.36	-211.36	81.00	-129.73	-207.66	77.93	3.07

Table 47 shows the exact delta rotation-angle error values of each axis. Given the rotation, the error in the yaw angle was 3.07 º. The error in roll angle is 0.28 º and the error in pitch angle is 0.09 º.

Yaw angle errors of up to 3 º are observed. This error is due to the high-distortion magnetic environment.

8.1.4 Roll-Pitch Angle Rotation Test

8.1.4.1 Test-1

Test conditions were determined as follows;

- Motion input is given in roll and pitch angle
- Data was collected at 150Hz

Figure 52 shows average oscillations of up to 0.5 º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data.

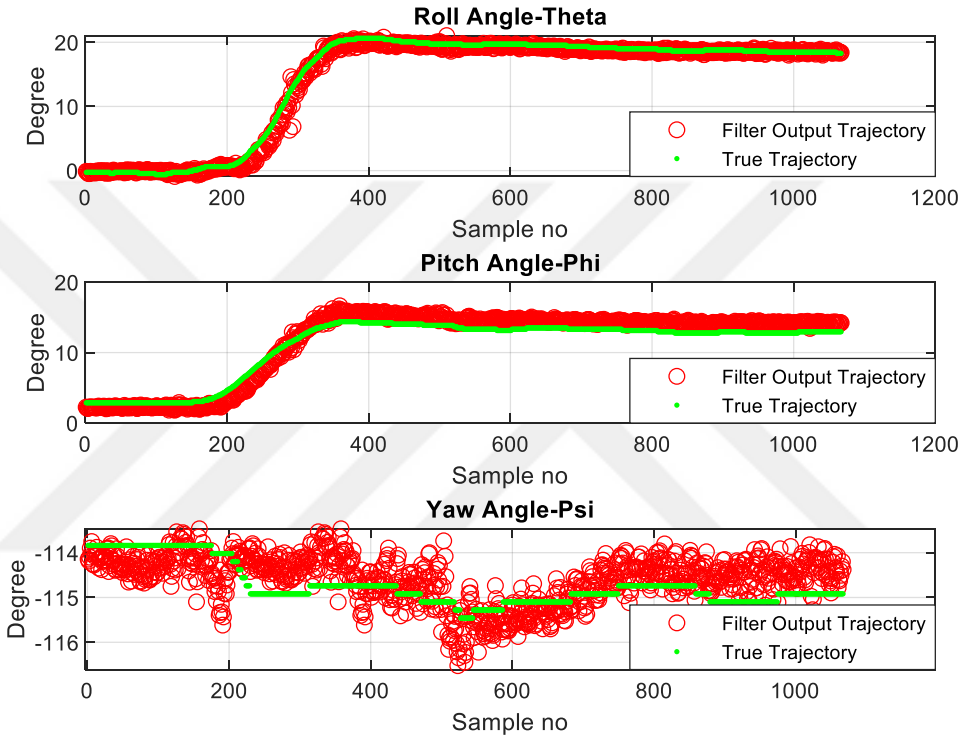


Figure 52: Encoder Data Along Roll Angle Rotation Movement

Table 48 shows each axis's exact delta rotation values for true trajectory and filter output.

Table 48: Encoder and UKF Delta Angles and Errors for Roll and Pitch Angle Rotations

	Encoder (º)			UKF (º)			Error (º)
	Start	End	Δ	Start	End	Δ	
Roll	-0.26	18.27	18.53	0.05	18.48	18.43	0.10
Pitch	2.87	12.95	10.08	2.44	13.87	11.43	1.35
Yaw	-113.83	-114.91	1.08	-114.23	-114.42	0.19	0.89

Table 48 shows the exact delta rotation-angle error values of each axis. The error in the roll angle is 0.1 º. The error in pitch angle is 1.35º and the error in yaw angle is 0.89 º.

8.1.5 Pitch-Yaw Angle Rotation Test

8.1.5.1 Test-1

Test conditions were determined as follows;

- Motion input is given in pitch and yaw angles
- Data was collected at 150Hz

Figure 53 shows average oscillations of up to 0.5º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data.

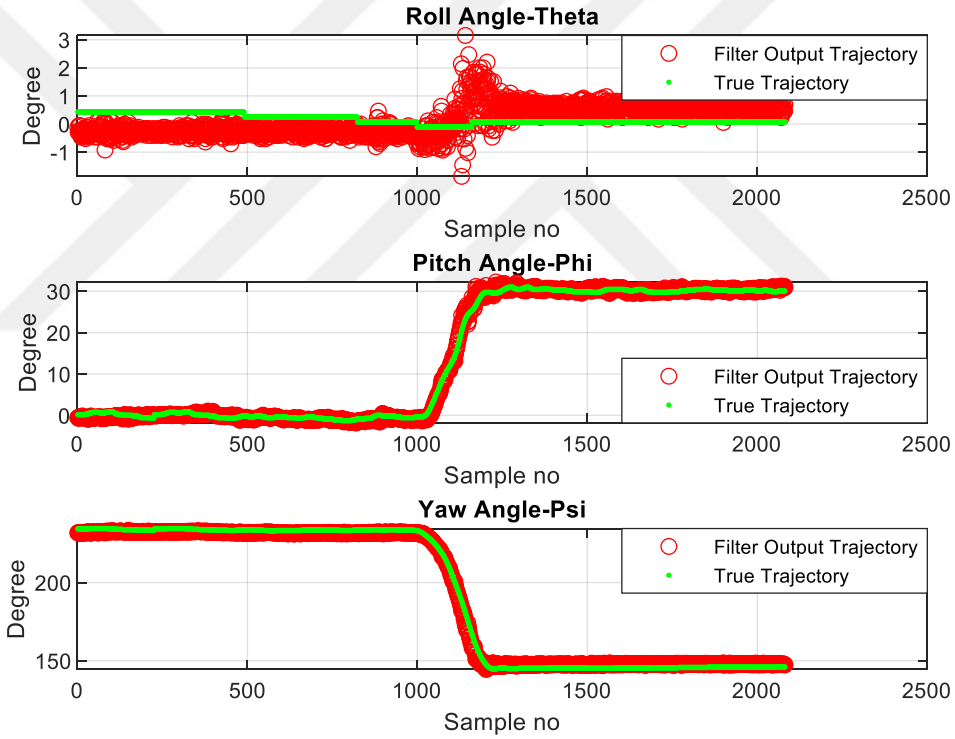


Figure 53: Encoder Data Along Roll Angle Rotation Movement

Table 49: Encoder and UKF Delta Angles and Errors for Pitch and Yaw Angle Rotations

	Encoder (º)			**UKF (º)**			**Error (º)**
	Start	End	Δ	Start	End	Δ	
Roll	0.43	0.07	0.36	-0.34	0.46	0.80	0.44
Pitch	0.18	30.24	30.06	-0.11	30.16	30.27	0.21

Table 49 Continued

Yaw	233.3	146	87.3	232.4	146.23	86.17	1.13

Table 49 shows the exact delta rotation-angle error values of each axis. The error in the roll angle is 0.44 º. The error in pitch angle is 0.21 º and the error in yaw angle is 1.13º.

8.1.6 Roll-Yaw Angle Rotation Test

8.1.6.1 Test-1

Test conditions were determined as follows;

- Motion input is given in roll and yaw angles
- Data was collected at 150Hz

Figure 54 shows average oscillations of up to 0.5º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data.

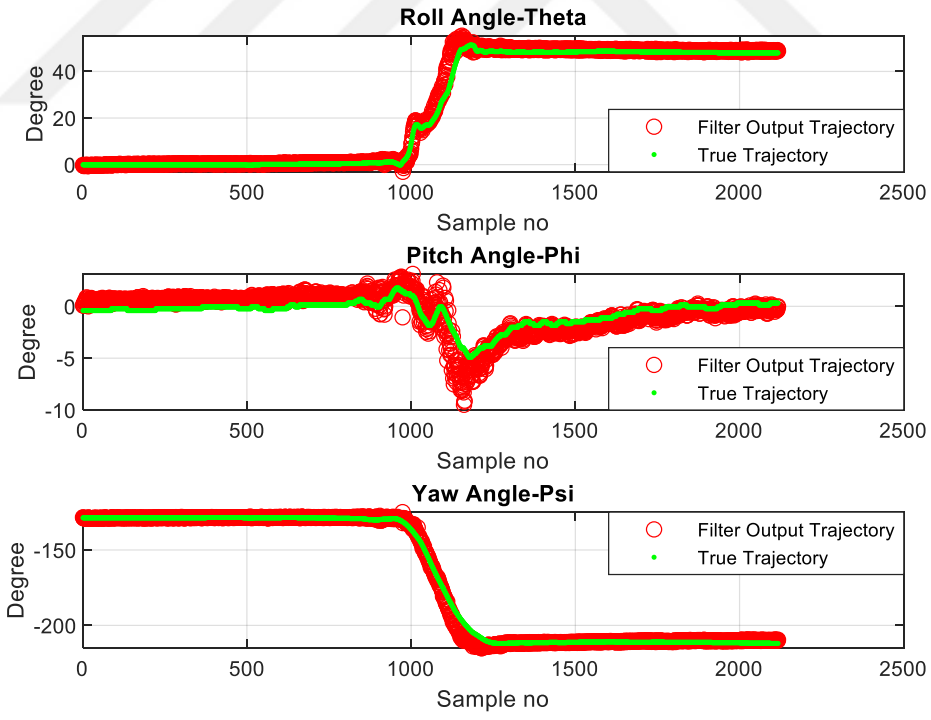


Figure 54: Encoder Data Along Roll Angle Rotation Movement

Table 50: Encoder and UKF Delta Angles and Errors for Roll and Yaw Angle Rotations

	Encoder (º)			UKF (º)			Error (º)
	Start	End	Δ	Start	End	Δ	
Roll	-0.07	48.16	48.23	-0.11	48.72	48.83	0.60
Pitch	-0.04	0.31	0.35	0.75	0.12	0.63	0.28
Yaw	-128.46	-211.08	82.62	-128.55	-210.88	82.33	0.29

Table 50 shows the exact delta rotation-angle error values of each axis. The error in the roll angle is 0.60º. The error in pitch angle is 0.28 º and the error in yaw angle is 0.29º.

8.1.7 Roll-Pitch-Yaw Angle Rotation Test

8.1.7.1 Test-1

Test conditions were determined as follows;

- Motion input is given in roll, pitch and yaw angles
- Data was collected at 150Hz

Figure 55 shows average oscillations of up to 0.5 º in the AHRS output. Additionally, it is seen that the filter output closely parallels the encoder data.

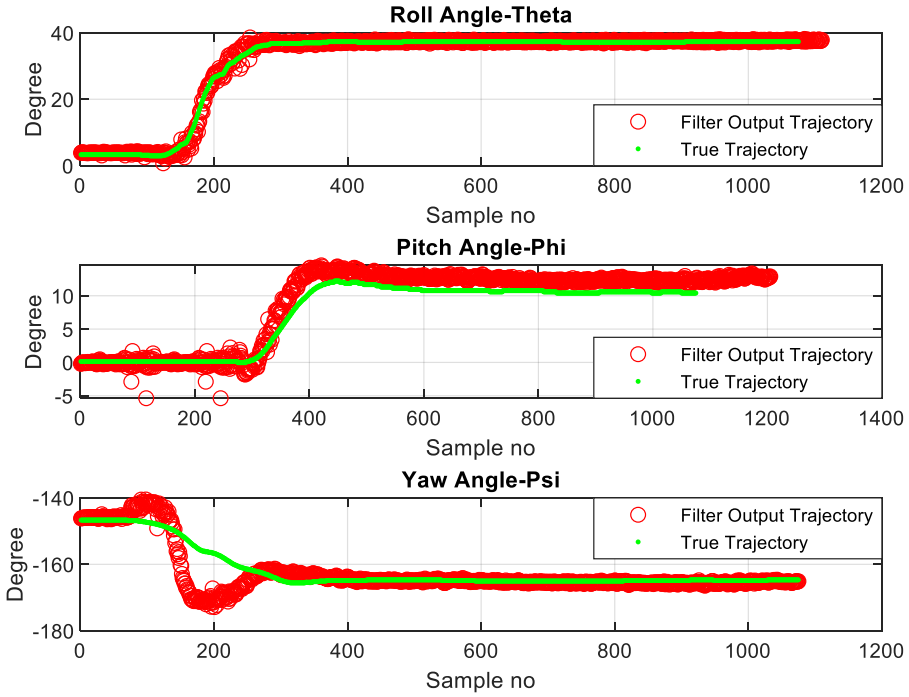


Figure 55: Encoder Data Along Roll Angle Rotation Movement

Table 51: Encoder and UKF Delta Angles and Errors for Roll, Pitch and Yaw Angle Rotations

	Encoder (º)			UKF (º)			Error (º)
	Start	End	Δ	Start	End	Δ	
Roll	3.32	37.34	34.02	3.83	37.66	33.83	0.19
Pitch	0.14	10.58	10.44	0.06	12.25	12.19	1.75
Yaw	-146.67	-164.67	18	-145.93	-165.47	19.54	1.54

Table 51 gives the exact delta rotation-angle error values of each axis. The error in the roll angle is 0.19 º. The error in pitch angle is 1.75 º and the error in yaw angle is 1.54º.

8.2 RESULTS

Table 52 lists all test results for roll, pitch and yaw angle as in degree. It is observed that the errors in the roll and pitch axes show similar characteristics. However, it was observed that the number of errors in the yaw angle was higher than the roll and pitch angle errors because the magnetic distortion of the environment could not be corrected well enough. In the designed AHRS setup, only the gyroscope sensor was used to support the accuracy of the yaw angle obtained from the magnetometer sensor data. However, other external units (GPS/GNSS) can improve this accuracy value.

The designed AHRS system was tested only on one axis, two, and three axes during the completed tests. For this reason, tests of the developed state estimator, the AHRS prototype card, and other sub-algorithms that comprise the AHRS system were carried out.

Table 52: Overall Results

Real System Test Results			
Test No	**Roll Angle Error(º)**	**Pitch Angle Error(º)**	**Yaw Angle Error(º)**
1	0.41	0.95	0.50
2	0.98	0.88	0.20
3	0.21	0.08	1.27
4	0.67	1.16	3.79
5	0.14	0.15	3.09

Table 52 Continued

6	0.28	0.09	3.07
7	0.1	1.35	0.89
8	0.44	0.21	1.13
9	0.60	0.28	0.29
10	0.19	1.75	1.54

Table 53 lists RMSE for roll, pitch and yaw angle as in degree.

Table 53: RMSE for UKF Estimator

	Roll Angle (º)	**Pitch Angle (º)**	**Yaw Angle (º)**
RMSE (º)	0.492	0.871	1.996

When the roll and pitch angle results are evaluated, it is observed that both error values remain below 0.9 degrees, while the yaw angle error remains below 2 degrees.

CHAPTER IX
CONCLUSIONS AND FUTURE WORKS

This thesis aims to design an AHRS system in which the attitude and orientation of unmanned aerial vehicles in 3D space are obtained. In addition, a test system design was carried out to construct the state estimators, which are the most essential part of the design of this system, compare different estimators, and carry out the necessary tests and improvements.

Some steps need to be followed to design the AHRS system. Sensor calibrations of the accelerometer, magnetometer and gyroscope sensors used in the system were carried out to obtain high-accuracy attitude information. In addition, soft and hard iron calibration was carried out before each use to eliminate magnetic distortions affecting the magnetometer sensor.

The attitude estimator, the main algorithm used to obtain attitude information, was designed by more than one attitude estimator. In these designs, estimators are designed to eliminate gyroscope drift errors. When comparing these estimators, KF is unsuitable for our problem because it works in linear systems. The UKF filter was observed to be the estimator with the highest accuracy value. However, the estimator runs the slowest processing time in the simulation environment. CF filter was observed to be the estimator with the lowest accuracy value. However, the estimator runs the fastest in terms of processing time in the simulation environment. UKF was preferred in this study because it is the estimator that works with the highest accuracy while obtaining attitude information at high speeds, such as 150 Hz, in the tests performed on the prototype system. This speed is considered sufficient within the scope of this study.

This thesis study used a 3-axis test system designed to test the AHRS system. There are multiple reasons for creating this system. These are using sensors in the AHRS system for sensor calibrations, supporting algorithm development studies on the attitude estimator, and allowing the necessary verification tests to be carried out.

In the tests performed with the 3-axis test system, the roll and pitch angle error values remain below 1 º, while the yaw angle error value remains below 2 º.

In future works, it is planned that the currently obtained accuracy values can be improved by rearranging the designed UKF filter with different state matrices.

REFERENCES

[1] Harvard University (2015), *Unmanned Aircraft Systems (Drones), Radcliffe Institute for Advanced Study*, https://rmas.fad.harvard.edu/unmanned-aircraft-systems-drones, DoA. 28.06.2024.

[2] RHUDY M., LARRABEE T., CHAO H., GU Y. and NAPOLITANO M. (2013), "UAV Attitude, Heading, and Wind Estimation using GPS/INS and an Air Data System", *AIAA Guidance, Navigation, and Control (GNC) Conference*, pp. 1–11, Boston, MA.

[3]VectorNav Technologies (2024), *Theory of AHRS Operation*, https://www.vectornav.com/resources/inertial-navigation-primer/theory-of-operation/theory-ahrs, DoA. 28.06.2024.

[4]WEISSTEIN E. (2024), *Euler Angles*, https://mathworld.wolfram.com/EulerAngles.html, DoA. 28.06.2024.

[5]Technical University of Munich (2024), *Data Fusion*, https://www.fsd.ed.tum.de/research/sensors-data-fusion-and-navigation/research-and-competence-areas/data-fusion/, DoA. 28.06.2024.

[6]Mouser Electronics (2024), *Choosing the Right Accelerometer*, https://www.mouser.com/pdfdocs/choosing-the-right-accelerometer-white-paper.pdf, DoA. 28.06.2024.

[7]Aeronsystems (2024), *Gyroscopes and Their Types*, https://aeronsystems.com/gyroscopes-and-their-types/, DoA. 28.06.2024.

[8] PNI Sensor Corporation (2024), *Understanding Magnetometers and Their Uses*, https://www.pnicorp.com/understanding-magnetometers-and-their-uses/, DoA. 28.06.2024.

[9] VectorNav Technologies (2024), *What is an Inertial Measurement Unit (IMU)?*, https://www.vectornav.com/resources/technical-documentation/technical-notes, DoA. 28.06. 2024.

[10] DE MARINA H. G., PEREDA F. J., GIRON-SIERRA J. M. and ESPINOSA F. (2012), "UAV Attitude Estimation Using Unscented Kalman Filter and TRIAD", *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4465-4474.

[11] PAN Y., SONG, P., LI K., ZHOU Y. and WANG Y. (2011), "Attitude Estimation of Miniature Unmanned Helicopter using Unscented Kalman Filter", *In*, *Proceedings 2011 International Conference on Transportation, Mechanical and Electrical Engineering (TMEE)*, pp. 1548-1551, IEEE, United States of America.

[12] FARAHAN S. B., MACHADO J. J. M., ALMEIDA F. G. de and TAVARES J. M. R. S. (2022), "9-DOF IMU-Based Attitude and Heading Estimation Using an Extended Kalman Filter with Bias Consideration", *Sensors*, vol. 22, no. 6, p. 3416.

[13] CAO D., QU Q., LI C. and HE C. (2009), "Research of Attitude Estimation of UAV Based on Information Fusion of Complementary Filter", *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, pp. 1290-1293, Seoul, Korea (South).

[14] CHE H., WANG G. and SHI C. (2020), "A Multi-sensor Fusion Algorithm for UAV Attitude Estimation", *In*, *CSAA/IET International Conference on Aircraft Utility Systems (AUS 2020)*, Online Conference, pp. 948-952, IET.

[15] PRIANDANA K., HAZIM M., WULANDARI and KUSUMOPUTRO B. (2020), "Development of Autonomous UAV Quadcopters using Pixhawk Controller and Its Flight Data Acquisition", *2020 International Conference on Computer Science and Its Application in Agriculture (ICOSICA)*, pp. 1-6, Bogor, Indonesia.

[16]VectorNav Technologies (2024), *Specs: Calibration*, https://www.vectornav.com/resources/inertial-navigation-primer/specifications--and--error-budgets/specs-hsicalibration, DoA. 28.06.2024.

[17]VectorNav Technologies (2024), *Math Fundamentals: Reference Frames*, https://www.vectornav.com/resources/inertial-navigation-primer/math-fundamentals/math-refframes, DoA. 28.06.2024.

[18] LÓPEZ REVUELTA Á. (2017), *Orientation Estimation and Movement Recognition using Low Cost Sensors* (Master's Thesis), Blekinge Institute of Technology, Karlskrona, Sweden.

[19] GOLDSTEIN H., POOLE C. and SAFKO J. (2002), *Classical Mechanics*, 3rd ed., pp. 142–144, Pearson, San Francisco.

[20] LI Q., LI R., JI K. and DAI W. (2015), "Kalman Filter and Its Application", *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pp. 74-77, Tianjin, China.

[21] WELCH G. and BISHOP G. (2006), *An Introduction to the Kalman Filter*, https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf, DoA. 28.06.2024.

[22] WAN E. A. and VAN DER MERWE R. (2000), "The unscented Kalman filter for nonlinear estimation", *In*, *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, Ed. Simon Haykin, pp. 153-158, IEEE, United States of America.

[23] VALENTI R.G., DRYANOVSKI I. and XIAO J. (2015), "Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs", *Sensors*, vol. 15, no. 8, pp. 19302-19330.

[24] BUSSON A. (2024), "*Complementary Filter — AHRS 1.0 documentation*", https://ahrs.readthedocs.io/en/latest/filters/complementary.html, DoA. 28.06.2024.

[25] UNSAL D. (2012), *Estimation of Deterministic and Stochastic IMU Error Parameters* (Master's Thesis), Middle East Technical University, Ankara, Turkey.

[26]MathWorks (2024), *IMUSensor System Object*, https://www.mathworks.com/help/nav/ref/imusensor-system-object.html, DoA. 28.06.2024.

[27] KENNEDY J. and EBERHART R. (1995), "Particle Swarm Optimization", *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol.4, pp. 1942-1948, Perth, Australia.

[28]SERT C. (2024), *Least Square Regression*, https://users.metu.edu.tr/csert/me310/me310_5_regression.pdf, DoA. 28.06.2024.

[29] MathWorks (2024), *accelParams*, https://www.mathworks.com/help/nav/ref/accelparams.html, DoA. 28.06.2024.

[30] MathWorks (2024), *gyroParams*, https://www.mathworks.com/help/nav/ref/gyroparams.html, DoA. 28.06.2024.

[31] MathWorks (2024), *magParams*, https://www.mathworks.com/help/nav/ref/magparams.html, DoA. 28.06.2024.

[32] LIVADA B., VUJIC S., RADIC D., UNKASEVIC T. and BANJAC Z. (2019), "Digital Magnetic Compass Integration with Stationary, Land-Based Electro-Optical Multi-Sensor Surveillance System", *Sensors,* Vol. 19, No. 19, Article Number 4331.

[33] National Centers for Environmental Information (NCEI) (2024), *Geomagnetic Field Calculators - Magnetic Field Calculators*, https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml#igrfwmm, DoA. 28.06.2024.

[34] KHANDAIT R., KUMAR V., BHURSE V., TIWARI V. and KHUBALKAR S. (2022), "Quadcopter Control using Different Controllers", *2022 International Conference on Intelligent Controller and Computing for Smart Power (ICICCSP)*, pp. 1-6. Hyderabad, India.

[35] Analog Devices (2024), *ADXL355: High g, Low Noise, Low Drift, Digital Accelerometer*, https://www.analog.com/en/products/adxl355.html, DoA. 28.06.2024.

[36] PNI Sensor Corporation (2024), *RM3100 Geomagnetic Sensor*, https://www.pnicorp.com/rm3100/, DoA. 28.06.2024.

[37] STMicroelectronics (2024), *A3G4250D: MEMS Motion Sensor, 3-Axis Digital Output Gyroscope*, https://www.st.com/en/mems-and-sensors/a3g4250d.html, DoA. 28.06.2024.

[38] STMicroelectronics (2024), *STM32F411RC: High-performance access line, ARM Cortex-M4 core with DSP and FPU, 512 Kbytes Flash, 100 MHz CPU, ART Accelerator, STM32F411RCT6*, https://www.st.com/en/microcontrollers-microprocessors/stm32f411rc.html, DoA. 28.06.2024.

[39]Arduino (2024), *Arduino Mega 2560 Rev3*, https://store.arduino.cc/collections/boards-modules/products/arduino-mega-2560-rev3?_pos=10&_fid=58420c1ee&_ss=c, DoA. 28.06.2024.

[40] Omron (2024), *E6B2-CWZ6C-2000P/R-2M Incremental Rotary Encoder*, https://industrial.omron.com.tr/tr/products/E6B2-CWZ6C-2000P-R-2M, DoA. 28.06.2024.

[41] HUGHES M. (2016), *Understanding and Implementing the HC-12 Wireless Transceiver Module*, https://www.allaboutcircuits.com/projects/understanding-and-implementing-the-hc-12-wireless-transceiver-module/, DoA. 28.06.2024.

[42]MathWorks (2024), *Least Square Fitting*, https://www.mathworks.com/help/curvefit/least-squares-fitting.html, DoA. 28.06.2024.

[43] YING X., YANG L., KONG J., HOU Y., GUAN S. and ZHA H. (2012), "Direct least square fitting of ellipsoids", *In*, *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 3228-3231, IEEE, Tsukuba, Japan.

[44] FITZGIBBON A., PILU M. and FISHER R. B. (1999), "Direct least square fitting of ellipses", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 476-480.

[45]OMERDIC E. (2017), *ExpertVisit UL Lecture 1.1*, https://excellabust.fer.hr/_download/repository/EXCELLABUST_-_ExpertVisit_UL_Lecture_1.1_Jan_2017.pdf, DoA. 28.06.2024.

[46] Inertial Sense (2024), *Coordinate Frames*, https://docs.inertialsense.com/user-manual/reference/coordinate_frames/, DoA. 28.06.2024.

[47] ESPINOSA J.R. (2016), *Runway detection from map, video and aircraft navigational data* (Master's Thesis), Mexican Navy Research and Development Institute, Mexico.

[48]DIRSIG (2023), *Coordinate Systems*, http://dirsig.cis.rit.edu/docs/new/coordinates.html, DoA. 28.06.2024.

[49] SBG Systems (2024), *Reference Coordinate Frames*, https://support.sbg-systems.com/sc/kb/latest/underlying-maths-conventions/reference-coordinate-frames, DoA. 28.06.2024.

[50]Wikipedia (2024), *Euler angles*, https://en.wikipedia.org/wiki/Euler_angles, DoA. 28.06.2024.

[51] CARROLL M. (2015), *Streaming Accelerometer Data to a Real-Time Dashboard*, https://www.pubnub.com/blog/streaming-accelerometer-data-realtime-dashboard/, DoA. 28.06.2024.

[52] WATSON J. (2016), *MEMS Gyroscope Provides Precision Inertial Sensing*, https://www.analog.com/en/resources/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html, DoA. 28.06.2024.

[53] Electronic Products (2012), *Magnetometer Basics for Mobile Phone Applications*, https://www.electronicproducts.com/magnetometer-basics-for-mobile-phone-applications/, DoA. 28.06.2024.

APPENDICES

APPENDIX A: DEFINITIONS FOR NAVIGATION SYSTEMS

Orientation

In navigation systems, 'orientation' pertains to the crucial process of ascertaining and upholding the accurate alignment or direction of a vehicle, vessel, aircraft, or individual concerning their immediate environment. This foundational element of navigation aids in comprehending one's current facing and intended trajectory.

True North

True North designates the direction pointing directly towards the geographic North Pole, serving as a pivotal reference for establishing orientation within navigation systems. Unlike magnetic north, true north remains constant across different locations and is crucial in conjunction with magnetic north for computing bearings and headings.

Magnetic North

Magnetic north denotes the direction a magnetic compass indicates, distinct from true north and susceptible to variation based on the local magnetic field's geographical location and characteristics. Navigational systems frequently necessitate adjusting the variance between magnetic north and true north to ensure accurate orientation and course plotting.

Heading

Heading describes the specific direction a vehicle or individual is moving or facing. This orientation is commonly quantified in degrees relative to true north (0°), progressing clockwise. GPS devices and gyrocompasses facilitate precision in determining heading, which provides reliable methods for ascertaining directional alignment.

Course

The course represents the intended direction of travel between two points, typically quantified in degrees relative to the true north. Navigation systems furnish essential guidance regarding the optimal course to pursue to reach a predetermined destination, offering vital information to facilitate efficient and accurate navigation.

Bearing

A bearing denotes the direction from one point to another, usually expressed in degrees relative to true north. It delineates the angle between the line connecting two points and the northward direction. Bearings establish the precise direction towards a particular location or waypoint, aiding navigation and route planning.

Attitude

Within aviation, marine, and specific land-based navigation systems, an attitude encompasses the spatial orientation of an object or vehicle across three dimensions. This orientation covers pitch, roll, and yaw, delineating the rotations of the aircraft or vessel about its respective axes. Ensuring the appropriate attitude is paramount for achieving stability and precise control during navigation.

APPENDIX B: AHRS DESIGN SUPPORTIVE WORKS

Particle Swarm Optimization Method

Particle Swarm Optimization (PSO) is a population-based, stochastic optimization algorithm inspired by the social behavior of birds flocking or fish schooling. It was developed by Dr. James Kennedy and Dr. Russell Eberhart in 1995 [27]. PSO solves optimization and search problems by simulating a group of particles' social behavior and movement within a multidimensional search space. This algorithm is particularly well-suited for continuous optimization problems.

The key idea behind PSO is to model the optimization process as a swarm of particles moving through the search space, with each particle representing a potential solution to the optimization problem. Each particle adjusts its position in the search space based on its own experience (personal best) and the experience of the swarm (global best). The following principles guide the movement of particles:

Position and Velocity: Each particle has a position and velocity in the search space. These parameters are adjusted over time to explore the solution space efficiently.

Personal Best (*pBest*): Each particle keeps track of its best-known position (solution) that has yielded the best fitness value.

Global Best (*gBest*): The swarm maintains the best-known solution across all particles in the population.

Social Interaction: Particles adjust their velocity and position based on their *pBest* and the *gBest*. This adjustment encourages particles to explore the search space efficiently by sharing information.

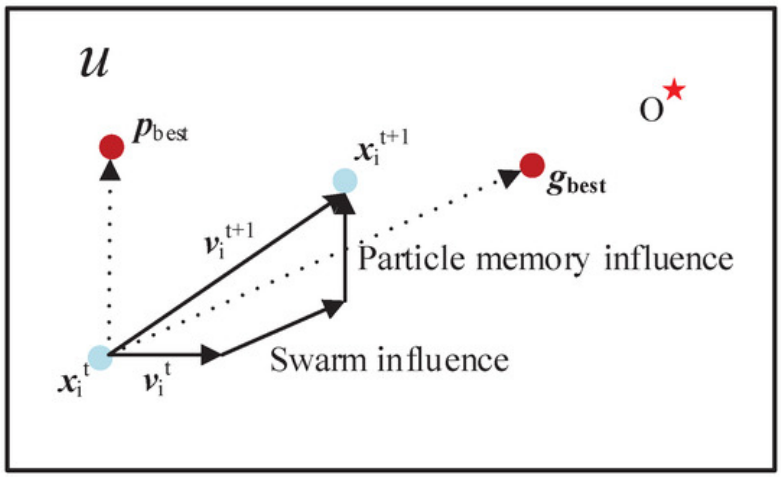
The PSO algorithm operates as follows:

- Initialize a population of particles with random positions and velocities within the search space.
- Evaluate the fitness of each particle's position.
- Update each particle's *pBest* if its current position is better than its previously recorded *pBest*.
- Determine the *gBest* position among all the particles in the population.

- Update the velocity and position of each particle based on its *pBest* and the *gBest*, using formulas that balance exploration and exploitation.
- Repeat steps 2-5 for a specified number of iterations or until a termination condition is met.

The algorithm converges to a solution as the particles adjust their positions over iterations.

PSO has been used in various optimization problems, such as function optimization, neural network training, image processing, and engineering design. It is appreciated for its simplicity and effectiveness in finding reasonable solutions in high-dimensional search spaces. However, PSO is not guaranteed to find the global optimum, as it may get stuck in local optima, and its performance can be sensitive to parameter settings. Researchers have developed variations and improvements of the PSO algorithm to address its limitations and adapt it to specific problem domains. Detailed information about figure given in [27].



Least Square Algorithm

The discrepancy between the observed response value and the predicted response value returned by the appropriate model for a given data point [42]. The equation to compute the vector of predicted responses is

$$\hat{y} = f(X, b) \tag{B.1}$$

where,

- $\hat{y}$ represents an n-by-1 vector of response estimates.
- f represents common form of the regression model.
- X represents design matrix which dimension n-by-m.
- b represents vector of fitted model coefficients which dimension m-by-1.

A least-squares fitting method computes model coefficients that minimize the sum of squared errors (SSE), also known as the residual sum of squares. For the ith data point, the residual r_i is calculated using the formula.

$$r_i = y_i - \hat{y}_i \tag{B.2}$$

where y_i is the *i*th observed response value and $\hat{y}_i$ is the *i*th fitted response value. The *SSE* is given by

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{B.3}$$

The discrepancy between the observed and actual values of a data point is termed the error. In practice, since the error cannot be directly measured, it is estimated using its residual[42].

Least-squares fitting methods perform best with datasets that lack significant outliers or random errors. Statistical outcomes, such as confidence intervals and prediction bounds, rely on the assumption that errors follow a normal distribution. Data fitting techniques typically rely on two key assumptions regarding errors in data that include random variations:

- Errors are assumed to exist solely in the response data and not in the predictor data.
- Errors are considered random and adhere to a normal distribution characterized by a zero mean and consistent variance.

Data fitting techniques typically assume errors follow a normal distribution because this distribution often provides a good approximation for many measured quantities. While the least-squares fitting method itself does not require normally

distributed errors to estimate parameters, it performs optimally with datasets lacking significant outliers or extreme random errors. The normal distribution is favored because extreme errors are rare within it. However, for statistical results like confidence and prediction intervals to hold validity, the assumption of normally distributed errors is typically necessary [42].

If the mean of the residuals is non-zero, it is important to assess whether the selection of the model or predictor variables is affecting the residuals.

Ellipsoid Fitting Algorithm

Fitting a quadratic surface to a set of points is essential in various applications like 3D pose estimation, reconstruction and object recognition. Previous methods can be divided into two main types: geometric and algebraic approaches. Algebraic methods employ a quadratic optimization model to achieve a non-iterative, singular solution, whereas geometric methods utilize iterative algorithms with non-linear models to achieve their objectives [43].

Conic Section and Classification

In the 2D system, the equation for a conic section can be expressed in the form:

$$ax^2 + by^2 + cxy + dx + ey + f = 0 \tag{B.4}$$

where a, b, c are not all zero. If $4ab - c^2$ value is bigger than 0, the equation describe an ellipse[43].

Conic Least-Square Fitting Approach

Given 2D points (x_i, y_i), we first collect data in the form of matrix D,

$$D = \begin{bmatrix} x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\ x_2^2 & y_2^2 & x_2y_2 & x_2 & y_2 & 1 \\ x_3^2 & y_3^2 & x_3y_3 & x_3 & y_3 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 1 \\ x_N^2 & y_N^2 & x_Ny_N & x_N & y_N & 1 \end{bmatrix} \tag{B.5}$$

The general least square solution for $\boldsymbol{a} = [a, b, c, d, e, f]^T$:

$$\min \Psi = || \boldsymbol{Da} ||^2 \tag{B.6}$$

s.t. $\boldsymbol{a^T a} = 1$.

So, we have,

$$\boldsymbol{Sa} = \lambda \boldsymbol{a}, \tag{B.7}$$

where $S = D^T D$. The eigenvector corresponding smallest eigenvalue of S is the solution of a [43].

Fitzgibbon's Ellipse-Specific Fitting Approach

We can freely scale the parameters, allowing us to integrate scaling into the constraint and impose the equality $4ab - c^2 = 1$. This quadratic constraint can be formulated in matrix form as:

$$\boldsymbol{a^T C a} = 1 \tag{B.8}$$

where,

$$\boldsymbol{C} = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.9}$$

The ellipse-specific fitting approach becomes:

$$\min \Psi = || \boldsymbol{Da} ||^2 \tag{B.10}$$

s.t. $\boldsymbol{a^T C a} = 1$.

We can obtain,

$$\boldsymbol{Sa} = \lambda \boldsymbol{Ca} \tag{B.11}$$

where $\boldsymbol{S} = \boldsymbol{D}^{\boldsymbol{T}}\boldsymbol{D}$. The unique elliptical solution corresponds to the generalized eigenvector associated with the single positive generalized eigenvalue of $\boldsymbol{S}$ and $\boldsymbol{C}$[44].

Quadratic Surface and Classification

In the 3D Cartesian coordinate system, a quadratic surface can be described by the general equation:

$$ax^2 + by^2 + cz^2 + dxy + eyz + fzx + gx + hy + iz + j = 0 \qquad \text{(B.12)}$$

There are 17 standard forms for quadratic surfaces. Examples include the cone, ellipsoid, cylinder, elliptic cone, elliptic hyperboloid elliptic cylinder, elliptic paraboloid, hyperbolic paraboloid, hyperbolic cylinder, paraboloid, and sphere.

Define:

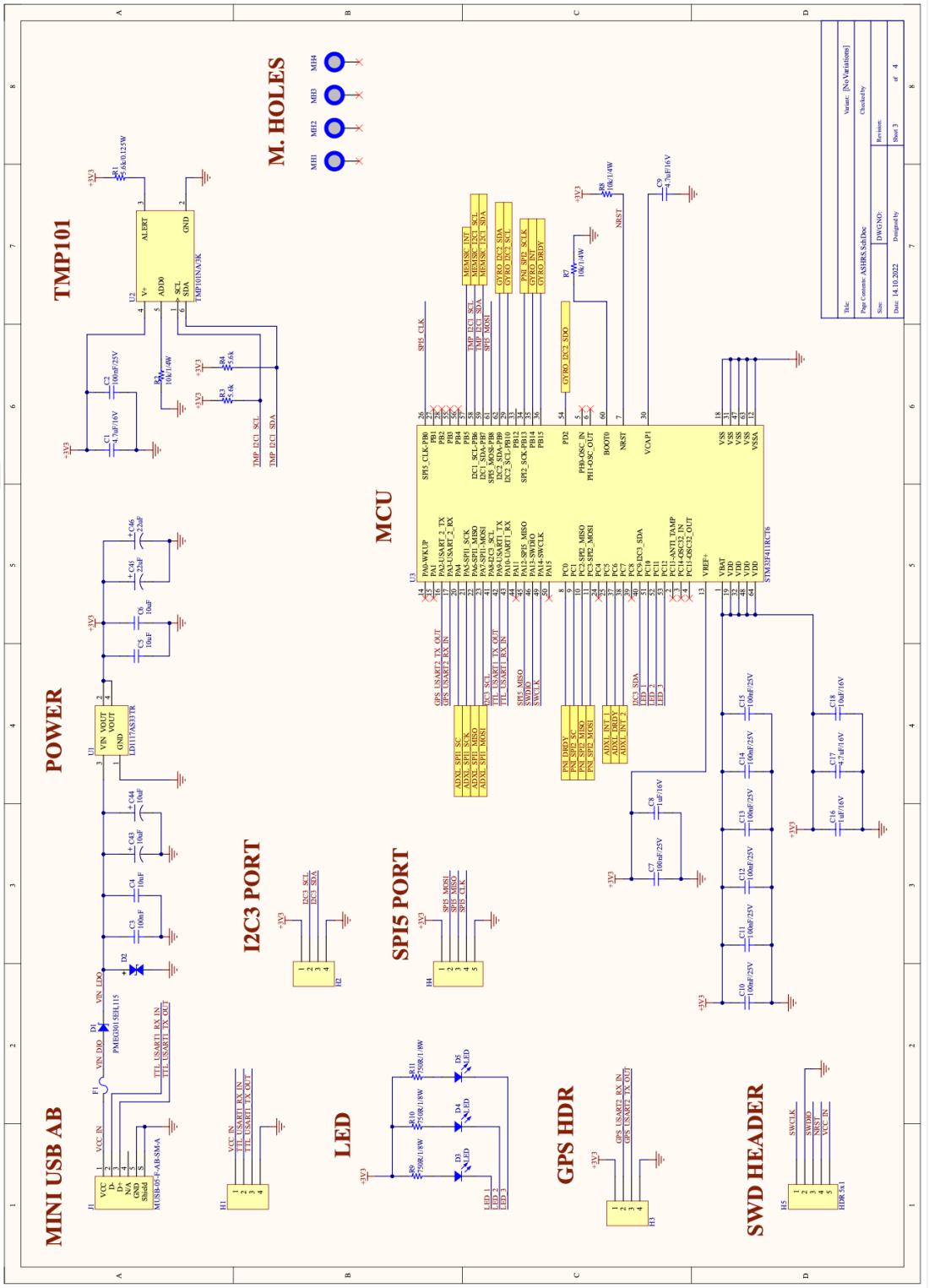
$$\mathbf{P} = \begin{bmatrix} a & d/2 & f/2 & g/2 \\ d/2 & b & e/2 & h/2 \\ f/2 & e/2 & c & i/2 \\ g/2 & h/2 & i/2 & j \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} a & d/2 & f/2 \\ d/2 & b & e/2 \\ f/2 & e/2 & c \end{bmatrix} \qquad \text{(B.13)}$$

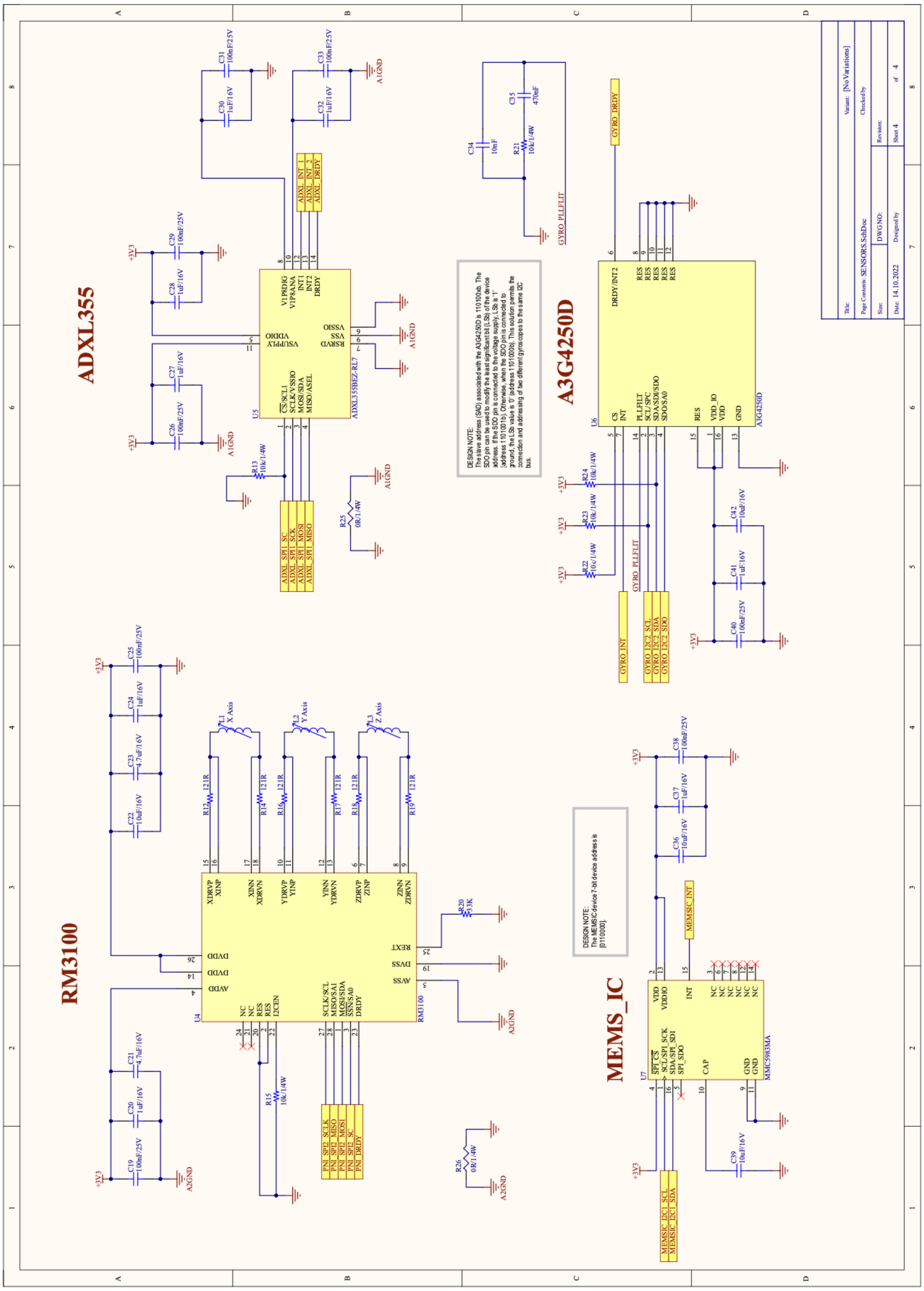
$\mathrm{P}_1 = \mathrm{rank}(\mathbf{P})$, $\mathrm{P}_2 = \mathrm{rank}(\mathbf{Q})$, $\Delta = \det(\mathbf{P})$, $\mathrm{k}_1, \mathrm{k}_2, \mathrm{k}_3$ are the eigenvalues of $\mathbf{Q}$, and define:

$$k = \begin{cases} 1 & if\ the\ signs\ of\ nonzero\ \mathrm{k}_1, \mathrm{k}_2, \mathrm{k}_3\ are\ same \\ 0 & otherwise \end{cases}$$

The constraints, such as the discriminants for quadratic surfaces or the necessary and sufficient conditions for a quadratic surface to be an ellipsoid, can be quite intricate. These often involve cubic or quartic constraints, making least squares problems with constraints higher than second degree particularly challenging to solve.

APPENDIX C: AHRS PCB SCHEMATIC DOCUMENTS




ADXL355
RM3100
A3G4250D
MEMS_IC

APPENDIX D: AHRS EXPERIMENTAL TEST PLATFORM

Main Code

```
#define X_Axis_A_PHASE 18
#define X_Axis_B_PHASE 19
#define Y_Axis_A_PHASE 20
#define Y_Axis_B_PHASE 21
#define Z_Axis_A_PHASE 2
#define Z_Axis_B_PHASE 3

unsigned int CCW_X = 0; //Assigned -X_Axis_flag_A
unsigned int CW_X = 0; //Assigned -X_Axis_flag_B
unsigned int CCW_Y = 0; //Assigned -Y_Axis_flag_A
unsigned int CW_Y = 0; //Assigned -Y_Axis_flag_B
unsigned int CCW_Z = 0; //Assigned -Z_Axis_flag_A
unsigned int CW_Z = 0; //Assigned -Z_Axis_flag_B

float countPerRevolution = 360.0 / 2000.0;

struct Angles{

float X_Axis_Roll_Angle;
float Y_Axis_Pitch_Angle;
float Z_Axis_Yaw_Angle;

};

float rollAngle  = 0.0, pitchAngle  = 0.0, yawAngle  = 0.0;
float Roll_Angle = 0.0, Pitch_Angle = 0.0, Yaw_Angle = 0.0;

Angles Diff_Angles ={0.0, 0.0, 0.0};

String incomingByte;
String resetCMD = "reset";

void setup()
{

pinMode(X_Axis_A_PHASE, INPUT);
pinMode(X_Axis_B_PHASE, INPUT);
pinMode(Y_Axis_A_PHASE, INPUT);
pinMode(Y_Axis_B_PHASE, INPUT);
pinMode(Z_Axis_A_PHASE, INPUT);
pinMode(Z_Axis_B_PHASE, INPUT);

Serial.begin(115200); //Serial Port Baudrate: 9600

attachInterrupt(digitalPinToInterrupt( X_Axis_A_PHASE), interrupt_X, RISING);
attachInterrupt(digitalPinToInterrupt( Y_Axis_A_PHASE), interrupt_Y, RISING);
attachInterrupt(digitalPinToInterrupt( Z_Axis_A_PHASE), interrupt_Z, RISING);
//Interrupt trigger mode: RISING

}

```

```
52. void loop()
53. {
54.
55. if (Serial.available() > 0) {
56. incomingByte = Serial.readString();
57. incomingByte.trim();
58.
59. if(incomingByte == resetCMD){
60. //Serial.println(incomingByte);
61. Reset();
62. incomingByte = "none";
63.
64. }
65. }
66.
67. Diff_Angles = Calc_Angles();
68. rollAngle  = rollAngle + Diff_Angles.X_Axis_Roll_Angle;
69. pitchAngle = pitchAngle + Diff_Angles.Y_Axis_Pitch_Angle;
70. yawAngle   = yawAngle + Diff_Angles.Z_Axis_Yaw_Angle;
71.
72. Roll_Angle  = Decide_Angle(rollAngle);
73. Pitch_Angle = Decide_Angle(pitchAngle);
74. Yaw_Angle   = Decide_Angle(yawAngle);
75.
76. Serial.print("SE;");Serial.print(Roll_Angle);Serial.print(";");Serial.print(Pitch_Angle);
77. Serial.print(";");Serial.print(Yaw_Angle);Serial.println(";FE");
78.
79. rollAngle = 0.0;
80. pitchAngle = 0.0;
81. yawAngle = 0.0;
82.
83. delay(10);
84. }
```

Angle Correction

```
1.  float Decide_Angle(float Angle)
2.  {
3.
4.  if(Angle >= 360.0){
5.  Angle = Angle - 360.0;
6.  }
7.  else if (Angle < 0.0){
8.  Angle = 360.0 + Angle;
9.  }
10. return Angle;
11.
12. }
```

Angle Calculation

```
1.  Angles Calc_Angles()
2.  {
3.
4.  float diff_roll_angle  = 0.0;
5.  float diff_pitch_angle = 0.0;
6.  float diff_yaw_angle   = 0.0;
7.
8.  float pos_Roll_Angle  = float(CCW_X) * countPerRevolution;
9.  float neg_Roll_Angle  = float(CW_X)  * countPerRevolution;
10. float pos_Pitch_Angle = float(CCW_Y) * countPerRevolution;
11. float neg_Pitch_Angle = float(CW_Y)  * countPerRevolution;
```

```
12. float pos_Yaw_Angle   = float(CCW_Z) * countPerRevolution;
13. float neg_Yaw_Angle   = float(CW_Z)  * countPerRevolution;
14.
15. diff_roll_angle  = pos_Roll_Angle  - neg_Roll_Angle;
16. diff_pitch_angle = pos_Pitch_Angle - neg_Pitch_Angle;
17. diff_yaw_angle   = pos_Yaw_Angle   - neg_Yaw_Angle;
18. Diff_Angles.X_Axis_Roll_Angle  = diff_roll_angle;
19. Diff_Angles.Y_Axis_Pitch_Angle = diff_pitch_angle;
20. Diff_Angles.Z_Axis_Yaw_Angle   = diff_yaw_angle;
21.
22. return Diff_Angles;
23. }
```

Reset Variables

```
1.  void Reset()
2.  {
3.
4.  CCW_X = 0.0;
5.  CW_X  = 0.0;
6.  CCW_Y = 0.0;
7.  CW_Y  = 0.0;
8.  CCW_Z = 0.0;
9.  CW_Z  = 0.0;
10. Roll_Angle  = 0.0;
11. Pitch_Angle = 0.0;
12. Yaw_Angle   = 0.0;
13.
14. }
```

X Axis Interrupt Function

```
1.  void interrupt_X()
2.  { // Interrupt function
3.
4.  char i;
5.  i = digitalRead( X_Axis_B_PHASE);
6.  if (i == 1)
7.  CCW_X += 1;
8.  else
9.  CW_X += 1;
10.
11. }
```

Y Axis Interrupt Function

```
1.  void interrupt_Y()
2.  {// Interrupt function
3.
4.  char i;
5.  i = digitalRead( Y_Axis_B_PHASE);
6.  if (i == 1)
7.  CCW_Y += 1;
8.  else
9.  CW_Y += 1;
10.
11. }
```

Z Axis Interrupt Function

```
1.  void interrupt_Z()
2.  {// Interrupt function
3.
```

```
4.  char i;
5.  i = digitalRead( Z_Axis_B_PHASE);
6.  if (i == 1)
7.  CCW_Z += 1;
8.  else
9.  CW_Z += 1;
10.
11. }
```