Sudan University of Science and Technology

College of Graduate Studies

Implementation of Denoising MEMS Accelerometer Using Wavelet Transformation

تطبيق مرشح التحويل المويجي على جهاز قياس التسارع الإلكتروميكانيكي الدقيق

A Thesis Submitted in Partial Fulfillment of the Requirements of the Degree of M.Sc. in Aeronautical Engineering (AVIONICS)

Prepared By:

Mohammed Abd Elmonim Mohammed Yousif

Supervisor:

Dr. Omer Abdel Razag Sharif Abubaker

MARCH 2019

الآية

بسم الله الرحمن الرحيم

قال تعالي:

(وَمَا أُوتِيتُم مِّن الْعِلْمِ إِلاَّ قَلِيلاً)

صدق الله العظيم

الآية (85) سورة الإسراء

ACKNOWLEDGEMENT

An endless appreciation cannot fulfill my research supervisor Dr. Omer Abdelrazag Sharif for his help throughout this thesis writing, he provides me with the strength to complete this thesis, I thank him very much for his endless support, guidance, encouragement, wise advice along with his precious time he gave to me, hoping him all the best of success in his life.

Many thanks also go to Dr. Sakher who shows us the principles of successful research work, also for his encouragement and support. Also sincerest gratitude goes to my all lectures throughout the whole year of my master study. For their endless knowledge that they never hesitate to answer all my questions, queries to help me understand advanced aeronautical engineering sciences.

Also not to forget my colleague in the master class for their cooperation and the endless knowledge sharing throughout the whole year of our class studies hoping them all the best in their future studies and work.

Also many thanks go to my superiors in SCAA Eng: Hisham Dawod Airworthiness Director for the facilitation he gave to allow me complete my class studies successfully, and also Eng: DAFFALLA, for his advices and help. Also Engineers: WAHIB, ATIF, SULAF and OMER for their help, support, encouragement and cooperation during class studies.

Before all most thanks and gratitude be to ALLAH, who without his approvingly nothing can be done.

DEDICATION

This thesis is dedicated to:

*My **Parents** who gave me the inspiration to achieve my goals*

To

*My **Brother** and **Sister** other part of mine*

To

*MY **Wife** that never stop encourage me in depressed times*

To

*My **Kids** for making my life meaningful and enjoyable*

TO

*My **All Teachers** who lights up the way for me through whole years of my studies*

To

*My **Colleagues***

To

*My **Family***

To them all I dedicate this work hoping them all the best and peace be upon

GOD BLESS US ALL

ABSTRACT

The MEMS accelerometer is an important sensor to measure the object's motion state (acceleration), and it is a key device in inertial systems such as, vibration measurement and navigation systems. MEMS accelerometer has the advantages of high precision and large measurement range, but it suffers from noise problems which results in negative consequences, because of double integration process to calculate the distance which results in erroneous measurements especially when integrated in IMU or INS systems**,** that may jeopardize safety.

In this research wavelet transformation method evaluated against Kalman and Kalman-wavelet combination methods, multi test scenarios were examined to select the outperformed method to be implemented using MPU-6050 sensor**,** and Arduino DUE development board**.** From the above mentioned multi test scenarios it concludes that the developed Wavelet transformation-based denoising algorithm outperforms Kalman and Kalman-Wavelet combination filtering methods by a percentage of 82% and 80% respectively so that it was the optimum method to be implemented as hypothesized in this research.

Finally it was implemented successfully in the single microcontroller platform (Arduino DUE) but with certain limitation in the decomposition level and the type of the mother wavelet used in addition to the number of signal samples.

المستخلص

يعتبر جهاز قياس التسارع الإلكتروميكانيكي الدقيق أحد المكونات الهامة لأجهزة القصور الذاتى وأنظمة الملاحة، حيث تمتاز بالدقة العالية ومجال القياس الواسع. يعانى مقياس التسارع الإلكتروميكانيكي الدقيق من تأثره بالضجيج الذاتي و البيئي والذي يؤدى إلى حدوث أخطاء قياس تراكمية نسبة لعملية التكامل المزدوج لحساب قيمة المسافة من التسارع.

فى هذا البحث تم إستخدام طريقة التحويل المويجي كمرشح لضجيج متحسس التسارع الإلكتروميكانيكي الدقيق (ام–بى- يو 6050) ومقارنتها مع مرشح "كالمان". لتقييم أداء المرشح المقترح والمرشحات الأخرى، تم تصميم عدد من التجارب الحاسوبية بإستخدام برنامج الماتلاب والتي أوضحت أن طريقة التحويل المويجى تفوقت على طرق الترشيح الأ خرى مثل كالمان والطريقة المركبة كالمان-التحويل المويجى بنسبة 82% و 80% على التوالي. أيضاً تم تطبيق خوارزمية التحويل المويجى لإزالة الضجيج بنجاح فى المتحكم الدقيق "أردوينو ديو" مع بعض المحددات، المتمثلة فى مستويات التفكيك ونوع دالة التحويل المويجى الأم بالأضافة الى عدد عينات الإشارة.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACC.	Accelerometer
AREF	Analog REFrence
ARM	Advanced RISC Machine
COTS	Commercial off the shelves
CPU	Central processing unit
DAC	Digital to analogue convertor
DC	Direct current
DLPF	Digital Low pass Filter
DMA	Direct Memory Access
DMP	Digital motion processing
GPS	Global positioning system
I2C	A communication standard
IDE	Integrated Development Environment
INS	Inertial Navigation System
KF	Kalman Filter
MEMS	Micro-Electro-Mechanical systems
PWM	Pulse width modulation
RISC	Reduced Instruction Set Computers
RADAR	RAdio Detection And Ranging
SCL	Serial clock
SDA	Serial data
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
STFT	Short Time Fourier Transform
SD	Standard Deviation

TWI	Two Wire Interface
UART	Universal Asynchronous Receive Transmit
UAV	Un-manned Aerial Vehicles
VC	Variable Capacitive
WF	Wavelet-based Filter

LIST OF SYMBOLS

$\boldsymbol{\varepsilon_A}$	Permittivity
$\boldsymbol{\varepsilon_\circ}$	Initial Permittivity
$\boldsymbol{\varepsilon}$	Permittivity
$\mathbf{A}$	Area of the electrodes
B	Optional control input n*l matrix
$\boldsymbol{C_\circ}$	Parallel-plate capacitance
d	Distance between capacitor plates
C_1	Formed Capacitor 1
C_2	Formed Capacitor 2
x_1	Displacement 1
x_2	Displacement 2
$\boldsymbol{\Delta C}$	Change in capacitance
k_s	Spring constant
$\boldsymbol{K_k}$	Kalman gain
$\mathbf{x}$	State value
$\boldsymbol{\Re^n}$	Real numbers
w_k	Random variable
v_k	Random variable
R	Measurement noise covariance
Q	Process noise covariance
A	Matrix n*n
z_k	Measurement Matrix
H	Measurement Matrix m*n
P_k	Estimation Error Covariance
$\boldsymbol{\tau}$	Translation

s	Scale parameters
$\boldsymbol{\psi}$ (t)	The transforming function
$\emptyset(\mathbf{t})$	Scale function
$\boldsymbol{\Psi}(\mathbf{t})$	Wavelet function
λ	Threshold
$\boldsymbol{\sigma}$	Standard deviation
n	Signal sample

CHAPTER ONE

INTRODUCTION

CHAPTER ONE
INTRODUCTION

1.1. Preface

MEMS stand for micro-electro-mechanical-systems. Hence, they are devices in the "micro" scale, in which one or more of their dimensions are in the micrometer range. The "electro" part indicates that they use electric power, for example for actuation and detection, "Mechanical" means these devices rely on some kind of mechanical, actions. The word "system" refers to the fact that they functioned, with in systems not stand alone [1].

The accelerometer is an important sensor to present the object's motion state (acceleration), and it is a key device in inertial systems such as, vibration measurement and navigation systems, and has the advantages of high precision and large measurement range. Therefore, this accelerometer has acquired much attention, especially in the aeronautical field. One of major problems that affect accelerometer performance is noise hence; the noise reduction is highly required.

Fabrication of accelerometers by MEMS technology is now an emerging technology due to its endless benefits such as light weight, less power consumption, high accuracy. MEMS have been used in many wide ranges of applications such as industrial, medical, military, inertia systems and Un-manned Aerial Vehicles (UAV) [2].

There are many types of MEMS –Based accelerometer such as: resistive, capacitive and piezoelectric. The Variable Capacitive (VC), MEMS accelerometers are high sensitivity devices used to measure constant acceleration this research concern on study the VC accelerometer. It is worthy to note that, the accelerometer that gives the highest accurate measurement may not be the smallest size, while one with the lowest noise may not be the lowest power. Although MEMS is small size, less power consumption, it is susceptible to white Gaussian noise, offset or drift.

1.2. Problem Statement

The MEMS accelerometer signal contaminated by noise during signal acquisition, due to sensor's physical characteristics, circuit devices and complex environmental factors which are degrading the accuracy and performance this results in negative consequences, because of double integration process to calculate the distance from the MEMS accelerometer signal which results in erroneous measurements, especially when integrated in IMU or INS systems, that may jeopardize safety. This research trying to solve this noise problem using Wavelet transformation method as well as implementation of the resultant filter in a single microcontroller/microprocessor (Arduino DUE).

1.3. Research Objective

a. Denoise MEMS Accelerometer (MPU 6050) by using wavelet transformation method.
b. Evaluate and validate the wavelet transformation method by, comparison with Kalman filter technique.
c. Implement the optimum method into the single microcontroller /microprocessor Arduino DUE.

1.4. Hypothesis

i. Wavelet transformation can be utilized as a denoising method for MEMS-based accelerometer; therefore the accelerometer performance can be improved noticeably.
ii. The wavelet transformation-based denoising algorithm can be implemented in a single microprocessor/microcontroller platform with acceptable performance.

1.5. Research Questions

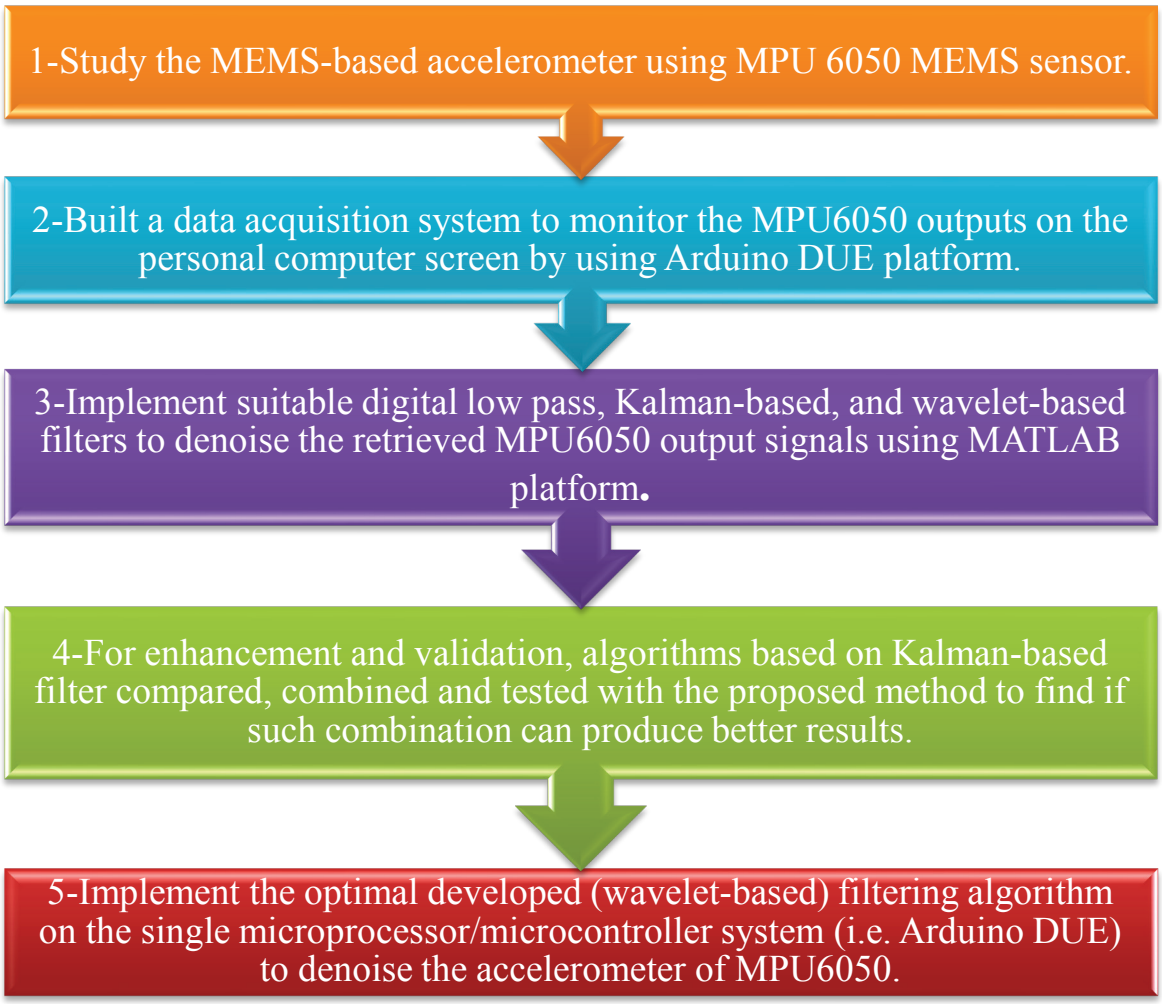
i. To what extend noise level can be reduced in MEMS accelerometer by applying wavelet?

ii. If the proposed wavelet algorithm combined with other denoising methods shall it produce better results?

iii. Can the proposed denoising wavelet-based algorithm be implemented in a single microprocessor/microcontroller platform?

1.6. Research Methodology

To achieve the research objective, the following method is followed:



1.7. Scope and limitation

The scope of this research is limited to the usage of the single microcontroller (Arduino DUE) and the MEMS accelerometer sensor MPU-6050. And the use of wavelet transformation method as filtering technique in addition to and Kalman filter for validation and comparison using MATLAB platform and Arduino IDE for implementation purposes.

1.8. Thesis Outlines

Including this chapter, the thesis contains five chapters. In Chapter Two brief representations of the utilized techniques on denoising MEMS sensor such as low pass filter, Kalman filter and Wavelet transformation. In Chapter Three, the connection of the Arduino-based MEMS acquisition system was presented where the required software was developed. Moreover, the proposed denoising MEMS technique based on Wavelet was also elaborated. In Chapter Four, the obtained results were depicted and discussed, where in Chapter Five the conclusion and recommendations were drawn.

CHAPTER TWO
LITRITURE REVIEW AND THEORTICAL BACKGROUND

CHAPTER TWO
LITRITURE REVIEW AND THEORTICAL BACKGROUND

2.1. Introduction

In this chapter literature review presents some previous studies related to the usage of wavelet technique in different MEMS applications also a brief theoretical background will be given, describing the MEMS accelerometers technology in general in addition to the denoising techniques used throughout this research to reduce it's noise effects which degrades the performance and hence the output data that may lead to negative consequences for all system using MEMS accelerometer.

2.2. Literature Review

Wavelet transforms have been successfully applied for de-noising, classification, recognition, compression, and other applications. It can decompose the signal to a frequency component in local time. By using this characteristic, the wavelet denoising method shrinks the signal noise by eliminating the frequency component which contains only noise. Furthermore, denoising quality is also improved by wavelet thresholding techniques. These are the schemes which are used to remove the noise in wavelet transform domain by using thresholding operator.

Woo *et al*. (2010) have proposed the wavelet transform in MEMS-GPS to reduce the signal distortion while improving the SNR of the signal. By applying this technique, it was proved that the INS signal could be improved and the overall navigation performance was also enhanced. Unfortunately, applying the wavelet transform in such complex system cause algorithm complexity.

Elkhidir T.A, *et al*. (2011) has proposed a wavelet multi-resolution analysis (WMRA) to solve MEMS-based INS performance which is affected by stochastic complex errors. Two threshold denoising functions were proposed to rise above the limitations of hard and soft threshold denoising, The results showed that the proposed threshold denoising functions compared with soft and hard thresholding denoising methods made the denoising better, and enhances the signal to noise ratio [9].

Recently, LI *et al*. (2014) have addressed the random noise that affects the precision of MEMS gyroscope. The Analysis of noise characteristic of gyroscope by using Allan variance method was used. The performed static and dynamic tests showed an effective elimination for random noise MEMS gyroscopes and improve the stability characteristics of the signal [10].

In addition EDU *et al.*, (2014) aimed to implement and validate procedures by proposing new algorithms that receive data from inertial navigation systems and provide accurate navigation information. In the first phase, an implementation of a real-time evaluation criterion with the intention of achieving real-time data from an accelerometer was done. It is well known that most errors in the detection of position, velocity and attitude in inertial navigation occur due to difficult numerical integration of noise. The main goal of the study was to propose a signal processing algorithm, based on the Wavelet filter, used along with a criterion for evaluating and updating the Wavelet's optimal level of decomposition, for reducing the noise component. A numerical simulations was performed using signals received from an accelerometer and analyzed the numerical results to see whether the improved Wavelet (proposed method) can be used to achieve more precise information on a vehicle. The results show great improvements [11].

ADOCHIEI, *et al*.(2015), have proposed a new software procedure for processing data from an inertial navigation system boarded on board a moving

vehicle , in order to achieve accurate navigation information on the displacement of the vehicle in terms of position, speed, acceleration and direction., a real-time evaluation criterion was implemented to achieving real-time data from an accelerometer. A signal processing algorithm, based on Wavelet filter was developed, to achieve accurate information from inertial sensors [12].

Most recently AMMAR, *et al* (2017),uses Kalman filters as a proposed noise reduction technique for MPU-6050 gyroscope signal the produced results was compared with the low pass filter, the proposed Kalman filter was performing [13].

2.3. MEMS Accelerometer Concept

An accelerometer is an electromechanical device that measures acceleration forces. These forces may be static, like the constant force of gravity pulling at our feet, or they could be Dynamic - caused by moving or vibrating the accelerometer. There are many types of accelerometers developed and reported in the literature. The vast majority is based on piezoelectric crystals, but they are too big. Scientist tried to develop something smaller, that could increase applicability and started searching in the field of microelectronics, and finally they developed MEMS accelerometer.

The first MEMS accelerometer was designed in 1979 at Stanford University, but it took over 15 years before such devices became accepted mainstream products for large volume applications In the 1990s MEMS accelerometers revolutionized the automotive-airbag system industry [3]. Since then they have enabled unique features and applications ranging from hard-disk protection on laptops …etc. More recently, used in industrial applications Such as aviation with advancements in MEMS the estimate of vehicle position and attitude has become affordable and increasingly accurate as sensors have become smaller and cheaper. Micro machined accelerometers are a highly enabling technology.

They provide lower power, compact and robust sensing in addition to the removal of the most mechanical complexity of conventional platform systems. Multiple sensors are often combined to provide multi-axis sensing and more accurate data [3]. Such combination in the inertial navigation system (INS) which combined with gyroscopes to provide the angular velocity.

There are many different ways to make an accelerometer. Some accelerometers use the piezoelectric effect - they contain microscopic crystal structures that get stressed by accelerative forces, which cause a voltage to be generated. Another way is by sensing changes in capacitance .This thesis is focused on the later one.

Capacitive interfaces have several attractive features. In most micromachining technologies no or minimal additional processing is needed. Capacitors can operate both as sensors and actuators. They have excellent sensitivity and the transduction mechanism is intrinsically insensitive to temperature. Capacitive sensing is independent of the base material and relies on the variation of capacitance when the geometry of a capacitor is changing. Neglecting the fringing effect near the edges, the parallel-plate capacitance is:

$$C_\circ = \varepsilon_\circ \varepsilon \frac{A}{d} = \varepsilon_A \frac{1}{d} \qquad (2.1)$$

where: $\varepsilon_A = \varepsilon_\circ \varepsilon A$ and (A) is the area of the electrodes, (d) the distance between them and (ε) the permittivity of the material separating them. A change in any of these parameters will be measured as a change of capacitance and variation of each of the three variables has been used in MEMS sensing. For example, whereas chemical or humidity sensor may be based on a change of (ε) accelerometers have been based on a change in (d) or in (A). If the dielectric in the capacitor is air, capacitive sensing is essentially independent of temperature but contrary to piezo resistivity, capacitive sensing requires complex readout electronics. Still the sensitivity of the method can be very large and, for example,

Analog Device used for his range of accelerometer a comb capacitor having a suspended electrode with varying gap. Measurement showed that the integrated electronic circuit could resolve a change of the gap distance of only 20 *Pico meter*, a mere 1/5th of the silicon inter-atomic distance [3].

Typical MEMS accelerometer is composed of movable proof mass with plates that is attached through a mechanical suspension system to a reference frame, Proof mass is attached through springs (k_s=spring constant) at substrate as shown in Figure 2.1. Movable plates and fixed outer plates represent capacitors. The deflection of proof mass is measured using the capacitance difference. The free-space (air) capacitances between the Movable plate and two stationary outer plates C_1 and C_2 are functions of the corresponding Displacements x_1 and x_2:

$$\begin{cases} C_1 = \varepsilon_A \ \frac{1}{x_1} = \ \varepsilon_A \ \frac{1}{d+x} = C_\circ - \Delta C \\ C_2 = \varepsilon_A \ \frac{1}{x_2} = \ \varepsilon_A \ \frac{1}{d-x} = C_\circ + \Delta C \end{cases} \tag{2.2}$$

If the acceleration is zero, the capacitances C_1 and C_2 are equal because $x_1 = x_2$. The proof mass displacement x results due to acceleration. If x = 0, the capacitance difference is found to be:

$$C_2 \ - \ C_1 \ = 2\Delta C = 2\varepsilon_A \ \frac{x}{d^2-x^2} \tag{2.3}$$

Measuring *C*, the displacement *x* can be found by solving the nonlinear algebraic equation:

$$\Delta C x^2 + \ \varepsilon_A x - \Delta C d^2 = 0 \tag{2.4}$$

This equation can be simplified. For small displacements, the term $\Delta C x^2$ is negligible. Thus, $\Delta C x^2$ can be omitted. Then, from:

$$x \approx \ (d^2/\epsilon_A)\Delta C = d(^{\Delta C}/_{C_\circ}) \tag{2.5}$$

So it is concluded that the displacement is approximately proportional to the capacitance difference ΔC [3].

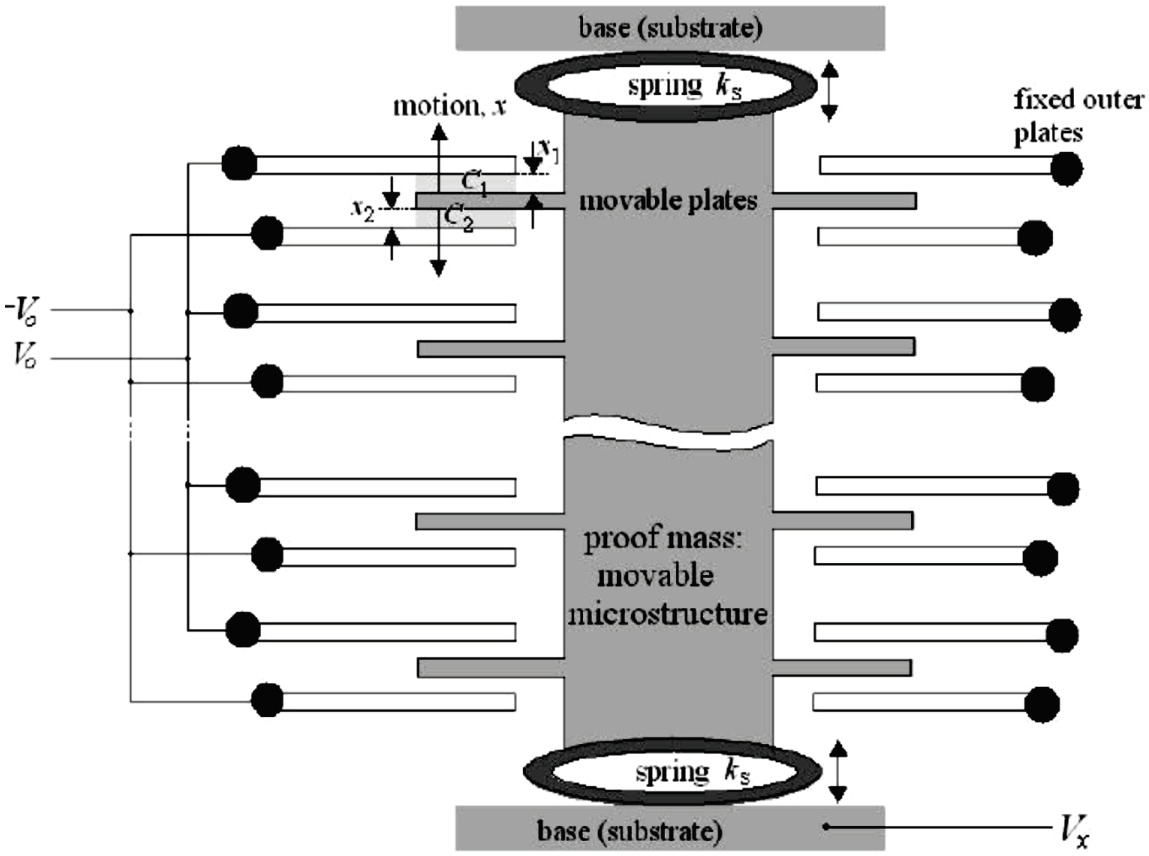


Figure (2.1): Accelerometer Structure [3].

2.4. MEMS benefits and limitations

With the development of microelectronics and integrated circuit technology, the use of MEMS inertial sensor technology with its small size, light weight, low cost (from patch fabrication), enhanced performance, reliability and excellent product characteristics, are widely used in automotive, aviation, aerospace and weapons guidance and other military and civilian fields. But a substantial increase in computing complexity when integrated with other systems such as INS is a one of the limitations.

Also accelerometer performance is affected with error sources, these sources can be recognized in three forms: White Gaussian noise (thermal noise), Offset, and Drift as shown in Figure 2.2.

***Noise*:** sources in the MEMS sensor systems generally are summarized as:

- ❖ Noise from the environment such as temperature fluctuations, and gravity.
- ❖ Unwanted movements, for example, shocks, vibrations.
- ❖ Noise from the hardware such as electrical noise and mechanical thermal noise.

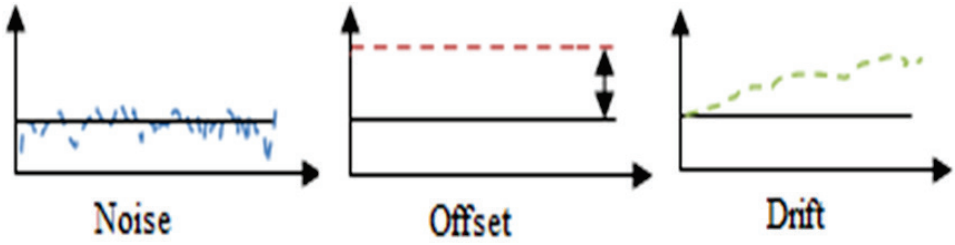


Figure (2.2): Noise Types at MEMS which are: Noise, Offset and Drift.

Offset: is also called zero error or bias error. It is characterized that the output of the sensor is constant when there is no measured value. This implies a shift or displacement of the zero point. It is a common error that exists with most of the measured sensor signals.

Drift: means that the output changes with time when there is no change of the input signal. Drift can be caused by vibration, shock, aging, and temperature variations. Temperature drift is generated due to the temperature changes. This phenomenon is well-known with the MEMS gyroscope [4].

Accumulation error can also be regarded as a drift due to numerical integration during the position calculation. That is integration of the acceleration to get the velocity, and the integration of the velocity to get the position. The problem is that while acquiring the noisy signal due to error sources explained above, these integrations will increase noise levels at the output, it looks like amplifying them and here is the potential error source in INS navigation information.

2.5. Denoising Techniques

In the following, the used denoising techniques in this research were briefly highlighted. These techniques are Digital Low pass Filter (DLPF), Kalman Filter (KF) and Wavelet-based Filter (WF). For the purpose of implementation, those methods were firstly implemented and tested on MATLAB platform for comparison; to ensure right implementation for individual and combination methods, and verification. Then, an implementation of the best method has been applied to a single microcontroller i.e. Arduino DUE as a test platform.

2.5.1. Digital Low pass Filter (DLPF) Technique

A low pass filter is simply to pass appropriate certain frequency below a cut-off frequency. The normalized cut-off frequency is a parameter with a value from zero to one. A low-pass filter was applied to the MEMS accelerometer for noise reduction embedded into the MPU 6050. Normally the output of the MEMS is divided into signals of different frequencies. The idea behind is, that something known about the acceleration that would be measured. So that filtering out signals with a higher frequency can be filtered out easily using the configurable DLPF provided into the MPU 6050. The MPU 6050 is set up for applying the DLPF; typical values are between 0 and 6 to the last 3 bits of register DLPF_CFG i.e. the values from 00000000 to 00000110. [14]

Table 2.1 shows DLPF response to different bit setting it is shows how the DLPF reacts to the bits set. The delay factor should be considered with respect to the used hardware, which comes from the need to buffer values internally by the DLPF used for dividing the signal into different frequencies. In this thesis, the default DLPF setting i.e. DLPF_CFG is set to (2).

It is worthy to note that these filtered data is considered as the original measurement data throughout this thesis unless other consideration is stated explicitly.

Table (2.1): The value of DLPF configuration for bandwidth and delay [14].

DLPF_CFG	Accelerometer (Fs=1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs(kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

2.5.2. Kalman Filter Technique

The name filter comes out from the process of finding the “best estimate” from noisy data amounts to “filtering out” the noise. However a Kalman filter also doesn’t just clean up the data measurements, but also projects these measurements onto the state estimate.

A Kalman filter is an optimal estimator - i.e. infers parameters of interest from indirect, inaccurate and uncertain observations. It is recursive so that new measurements can be processed as they arrive. Optimal in the sense that if all noise is Gaussian, the Kalman filter minimizes the mean square error of the estimated parameters. If the noise is not Gaussian Given only the mean and standard deviation of noise, the Kalman filter is the best linear estimator. Non-linear estimators may be better [5]. The Kalman filter technique gives good results in practice due to optimality and structure, convenient form for online

real time processing, and also it is easy to formulate and implement given a basic understanding.

For application purposes, Kalman filter is used in determination of planet orbit parameters from limited earth observations. Also it is used in tracking targets - e.g. aircraft, missiles using RADAR, and recently it is used in Robot Localization and Map building from range of sensors. [5].

The Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation:

$$X_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{2.6}$$

With a measurement $z_k \in \Re^m$ that is:

$$z_k = Hx_k + v_k \tag{2.7}$$

where w_k and v_k are random variables represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions:

$$p(w) \sim N(0, Q) \tag{2.8}$$

$$p(v) \sim N(0, R) \tag{2.9}$$

In practice, the *process noise covariance* Q and *measurement noise covariance* R matrices might change with each time step or measurement, however here we assume they are constant. The $n \times n$ matrix A in the difference equation (2.6) relates the state at the previous time step $k-1$ to the state at the current step k, in the absence of either a driving function or process noise. Note that in practice A might change with each time step, but here we assume it is constant. The $n \times l$ matrix B relates the optional control input $u \in \Re^l$ to the state x. The $m \times n$ matrix H in the measurement equation (2.7) relates the state to the measurement z_k. In practice H might change with each time step or measurement, but here we assume it is constant [6].

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems as shown below in Figure (2.3).

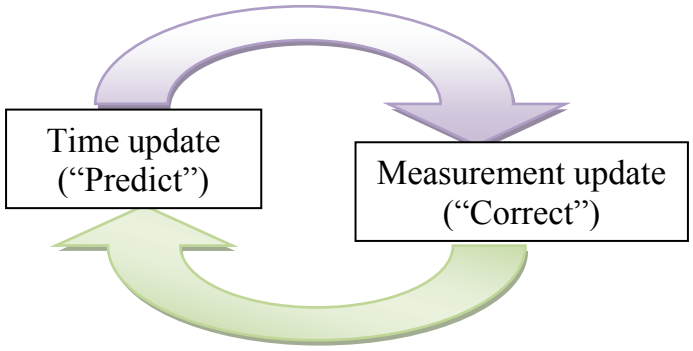


Figure (2.3): The Kalman filter cycle [6].

The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time. The discrete Kalman filter for the time and measurement updates equations are presented below:

$$\bar{\hat{x}}_k = A\hat{x}_{k-1} + Bu_{k-1} \tag{2.10}$$

$$\bar{P}_k = AP_{k-1}A^T + Q \tag{2.11}$$

It can be noticed that the time update equations (2.10) and (2.11) above project the state and covariance estimates forward from time step k-1 to step k.

A and B matrices are from (2.6), while Q matrix is from (2.8).

Discrete Kalman filter measurement update equations are given by:

$$K_k = \bar{P}_k H^T (H \bar{P}_k H^T + R)^{-1} \quad (2.12)$$

$$\hat{x}_k = \bar{x}_k + K_k (z_k - H \bar{x}_k) \quad (2.13)$$

$$P_k = (I - K_k H)\bar{P}_k \quad (2.14)$$

The first task during the measurement update is to compute the Kalman gain, K_k . The next step is to actually measure the process to obtain z_k, and then to generate an a posteriori state estimate by incorporating the measurement as in (2.13). The final step is to obtain an a posteriori error covariance estimate via (2.14).

After each time and measurement update pair, the process is repeated with the previous a posteriori estimates used to project or predict the new a priori estimates. This recursive nature is one of the very appealing features of the Kalman filter; it makes practical implementations much more feasible. The Kalman filter instead recursively conditions the current estimate on all of the past measurements. Figure (2.4) offers a complete picture of the operation of the filter [6].

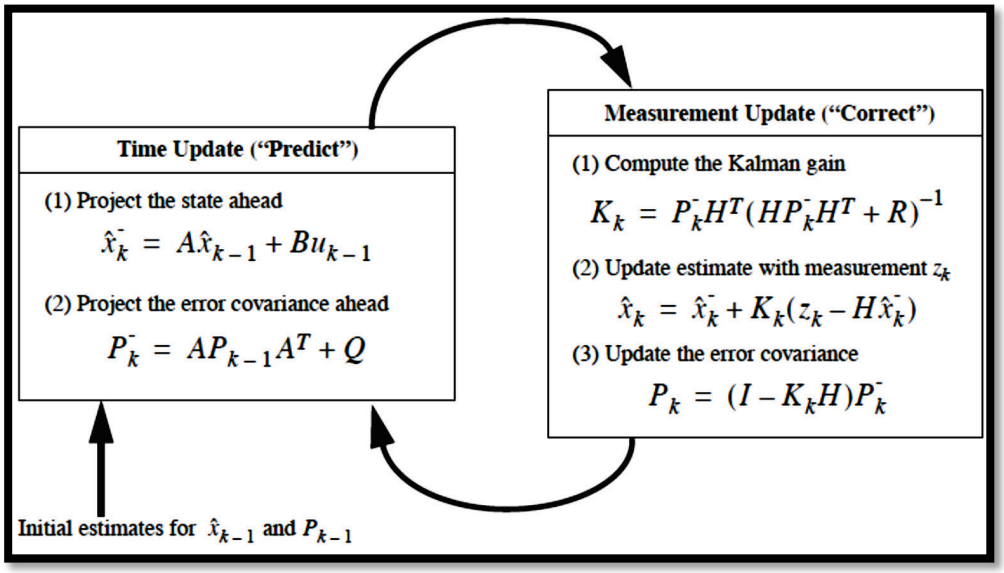


Figure (2.4): The operation of the Kalman filter overview [6].

Figure (2.4) gives a complete view of Kalman filter operation, combining the diagram of Figure (2.3) with the equations from (2.10) to (2.14). In the actual implementation of the filter, the measurement noise covariance R is usually measured prior to operation of the filter. Measuring the measurement error covariance R is generally practical (possible) because it is the process can be measured anyway (while operating the filter) so that some off-line sample measurements should be taken in order to determine the variance of the measurement noise. The determinant of the process noise covariance Q is generally more difficult as typically do not have the ability to directly observe the process that can be estimated. Sometimes a relatively simple (poor) process model can produce acceptable results if enough uncertainty injected into the process via the selection of Q. Certainly in this case hoping that the process measurements are reliable. In either case, whether or not having a rational basis for choosing the parameters, often times superior filter performance (statistically speaking) can be obtained by tuning the filter parameters Q and R. The tuning is usually performed off-line, frequently with the help of another (distinct) Kalman filter in a process generally referred to as system identification under conditions where Q and R .are in fact constant, both the estimation error covariance $P_{k,}$ and the Kalman gain K_k will stabilize quickly and then remain constant. If this is the case, these parameters can be pre-computed by either running the filter off-line, or for example by determining the steady-state value of P_k It is frequently the case however that the measurement error (in particular) does not remain constant the process noise Q is sometimes changed dynamically during filter operation , becoming Q_k in order to adjust to different dynamics [6].

2.5.3. Wavelet Transform Technique

Wavelet transform is a signal transform technique popularly used in several areas such as image processing and audio signal processing. The wavelet

analysis is based on a windowing technique with variable-sized windows as shown in Figure 2.5. The wavelet transform applies the wide window (long time intervals) to low frequency and the narrow window (short time intervals) to high frequency.

Wavelet transform can be defined as: "The wavelet transform is a tool that cuts up data, functions or operators into different frequency components, and then studies each component with a resolution matched to its scale".

Continuous wavelet transform is given by the following equation:

$$CWT_x^{\psi}(\tau, s) = \psi_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^* \left(\frac{t-\tau}{s}\right) dt \qquad (2.15)$$

Where: τ = translation, **s** =scale parameters, ψ (t) = the transforming function, and it is called the mother wavelet. $x(t)$ = continuous signal

The term mother wavelet gets its name due to two important properties of the wavelet analysis as explained below:

- **The term mother wavelet** implies that the functions with different region of support that are used in the transformation process are derived from one main function, or the mother wavelet.

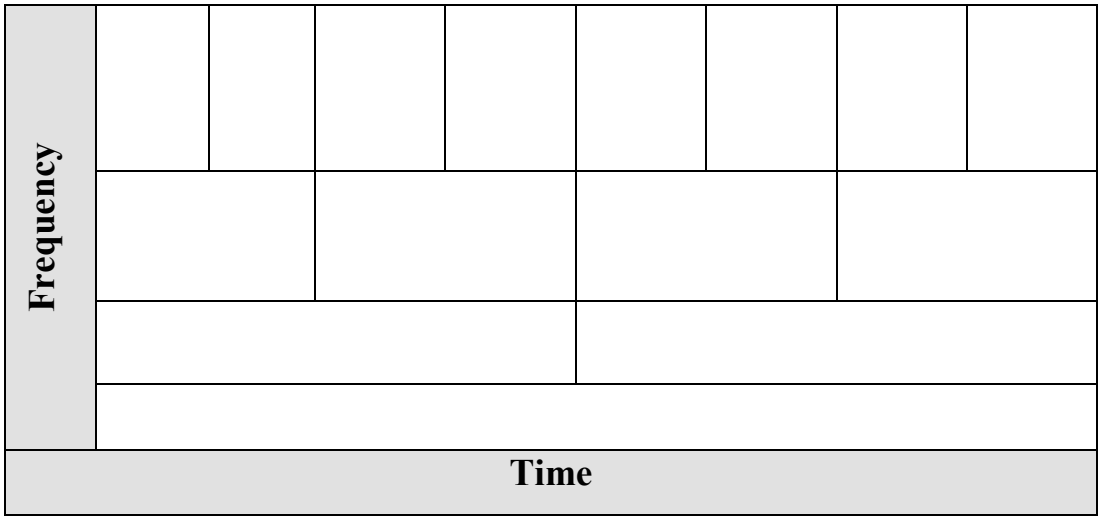


Figure (2.5): Wavelet multi resolution [7].

- **The term translation** is used in the same sense as it was used in the Short Time Fourier Transform (STFT); it is related to the location of the window, as the window is shifted through the signal. This term, obviously, corresponds to time information in the transform domain. Instead of a frequency parameter as in STFT, a scale parameter which is defined as (1/frequency) is used.
- **The parameter scale** in the wavelet analysis is similar to the scale used in maps. As in the case of maps, high scales correspond to a non-detailed global view (of the signal), and low scales correspond to a detailed view. Similarly, in terms of frequency, low frequencies (high scales) correspond to a global information of a signal (that usually spans the entire signal), whereas high frequencies (low scales) correspond to a detailed information of a hidden pattern in the signal (that usually lasts a relatively short time) [7].

Discrete time wavelet transform is executed as formula (2.16) and (2.17)

$$\emptyset(t) = \sum_k g_0\,[k]\emptyset(2t-k) \tag{2.16}$$

$$\Psi(t) = \sum_k g_1\,[k]\emptyset(2t-k) \tag{2.17}$$

Where $\emptyset$ (t) is called the scale function and $\Psi\,(t)$ is called the wavelet function. Each g_0 and g_1 refers to the wavelet coefficient [8].

Wavelet function can be any function that satisfies the relationship of formula (2.16) and (2.17). The scale function of upper level can be expressed as the convolution of the scale function and the wavelet function of lower level. It means that the low-frequency area can be decomposed to the high-frequency area and low-frequency area. Such relationship is shown in Figure (2.6).

Wavelet transform is progressed stage by stage. When the signal is divided into low-frequency waves, it requires twice the amount of data. In addition, the

lowest possible decomposable frequency area matches the Direct Current (DC) value of the Fourier transform calculated using the entire data.

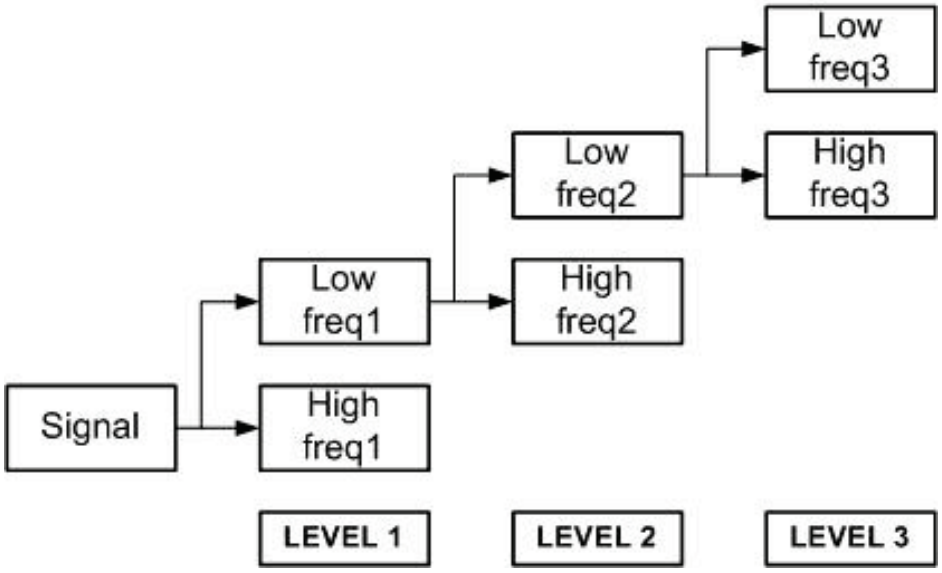


Figure (2.6): Wavelet decomposition concept [8].

Wavelet denoising technique was developed to reduce various noises included in the signal. Among its variations, the most representative method is the wavelet shrinkage technique, which is one of the thresholding techniques. Wavelet thresholding technique is a signal estimation technique that exploits the capabilities of wavelet transform for signal denoising. It removes the noise by eliminating coefficients that are insignificant relative to some threshold. Researchers have developed various techniques for choosing denoising parameters and so far there is no best universal threshold determination technique. Wavelet thresholding technique assumes that the magnitude of the actual signal is greater than the noise level, and the noise is white noise [8].

The wavelet thresholding technique, however, reduces the noise level with almost no distortion for the sudden change in signal. The result of the thresholding technique has almost no distortion and accurate, so that it sits right on the actual signal almost the same. Therefore, it can be used by preprocessing

filter for the inertial sensor signal and overcomes the shortages of the existing low-pass filter. The thresholding technique is classified into the hard thresholding and soft thresholding operation.

$$T_{\lambda}^{hard} = \begin{cases} x & if\ |x| \geq \lambda \\ 0 & otherwise \end{cases} \quad (2.18)$$

$$T_{\lambda}^{soft} = \begin{cases} (x - sign\,(x)\lambda\,) & if\ |x| \geq \lambda \\ 0 & otherwise \end{cases} \quad (2.19)$$

The formula (2.18) shows the hard thresholding function and the formula (2.19) shows the soft thresholding function, which are illustrated in Figure 2.7. (***x)*** is the wavelet coefficient. The wavelet coefficient means that the magnitude of a certain frequency component. Thus, wavelet coefficients in the band of noise become zero. Among the two methods, the soft thresholding technique is known to have better performance than the hard thresholding technique. One of the important elements influencing the performance in thresholding technique is how the standard value of (***λ***) is set. Generally, it is determined by the formula (2.20). Here, (***σ***) is the standard deviance of the signal, and (***n***) is the number of signal samples.

$$\boldsymbol{\lambda = \sqrt{(2\ log\ n)\ \sigma}} \quad (2.20)$$

The value determined by the formula, however, does not lead to the optimal result, so the appropriate **(*λ*)** must be determined through experimentation.

The thresholding algorithm is executed as follows:

- ***First***, the wavelet transform is executed for the signal to acquire the wavelet coefficients.
- ***Second***, the thresholding operation is executed for each wavelet coefficient. Then, the original coefficients are replaced to the coefficients from the result of the thresholding operation.
- ***Finally***, inverse transform is carried out [8].

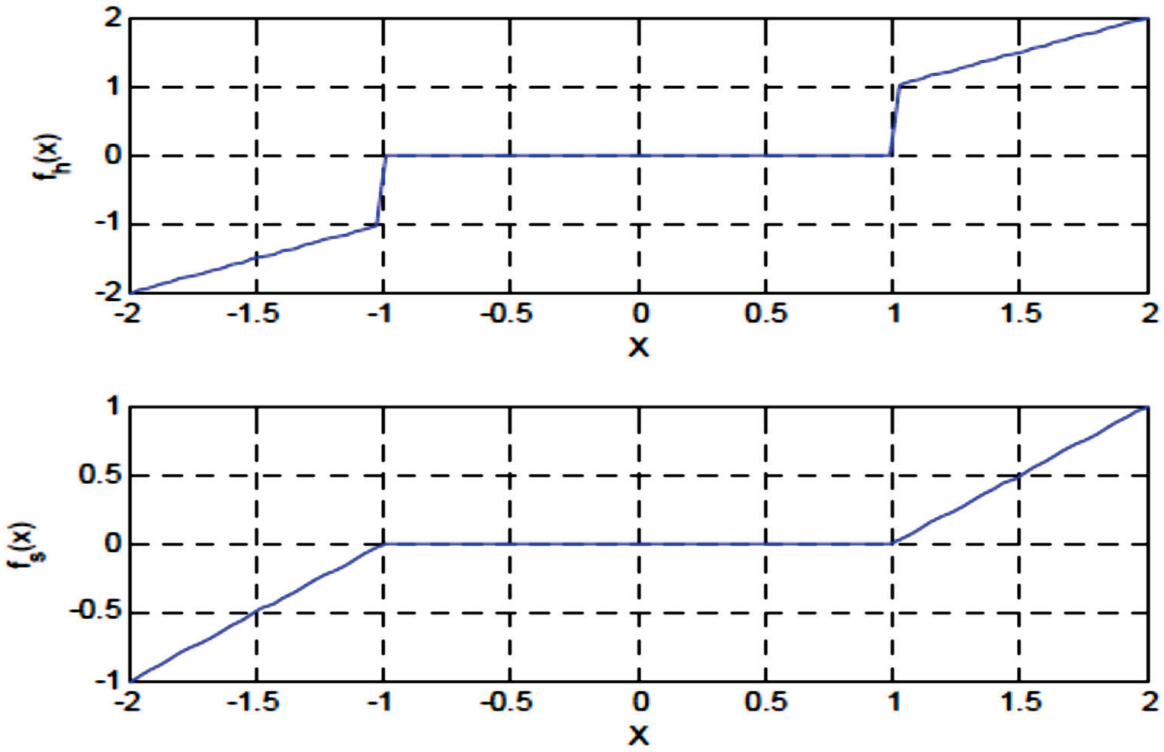


Figure (2.7): Hard and soft thresholding functions [8].

CHAPTER THREE
METHODOLOGY

CHAPTER THREE
METHODOLOGY

3.1. Introduction

In this chapter, the proposed wavelet transformation denoising technique is discussed. To evaluate the proposed technique and compare it with the previous filtering techniques, a test bed is designed and developed using the following commercial of the shelves (COTS) components: MPU-6050 (motion sensor unit), and the Arduino DUE board (processing unit) interconnected and programmed to process the raw data comes from the sensor (Figure 3.1).

The methods for MEMS denoising such as DLPF, Kalman Filter and Wavelet-based are simulated in the MATLAB environment to determine which is more convenient to the MPU-6050 by addressing different parameters in these techniques. Moreover, the resultant wavelet based technique is implemented in the composed system to show a real demonstration as can be seen in Chapter Four. The following paragraphs give a brief description of the proposed wavelet-based technique as well as the system's components and interconnection.

3.2. The MEMS MPU-6050 Sensor

For the purpose of this study the sensor MPU-6050 (see Figure (3.2) and Table (3.1)) is used. It is an integrated 6-axis motion processor solution that eliminates the package-level gyroscope and accelerometer cross-axis misalignment associated with discrete solutions.

The devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an onboard Digital Motion Processor™ (DMP™) capable of processing complex 9-axis sensor fusion algorithms using the field-proven and proprietary Motion Fusion™ engine. MPU-6050's integrated 9-axis Motion Fusion algorithms access external magnetometers or other sensors

through an auxiliary master I^2C bus, allowing the devices to gather a full set of sensor data without intervention from the system processor. The devices are in the 4x4x0.9 mm frame. The MPU-6050 supports communications at up to 400 kHz and has a Vcc pin that defines its interface voltage level [14].

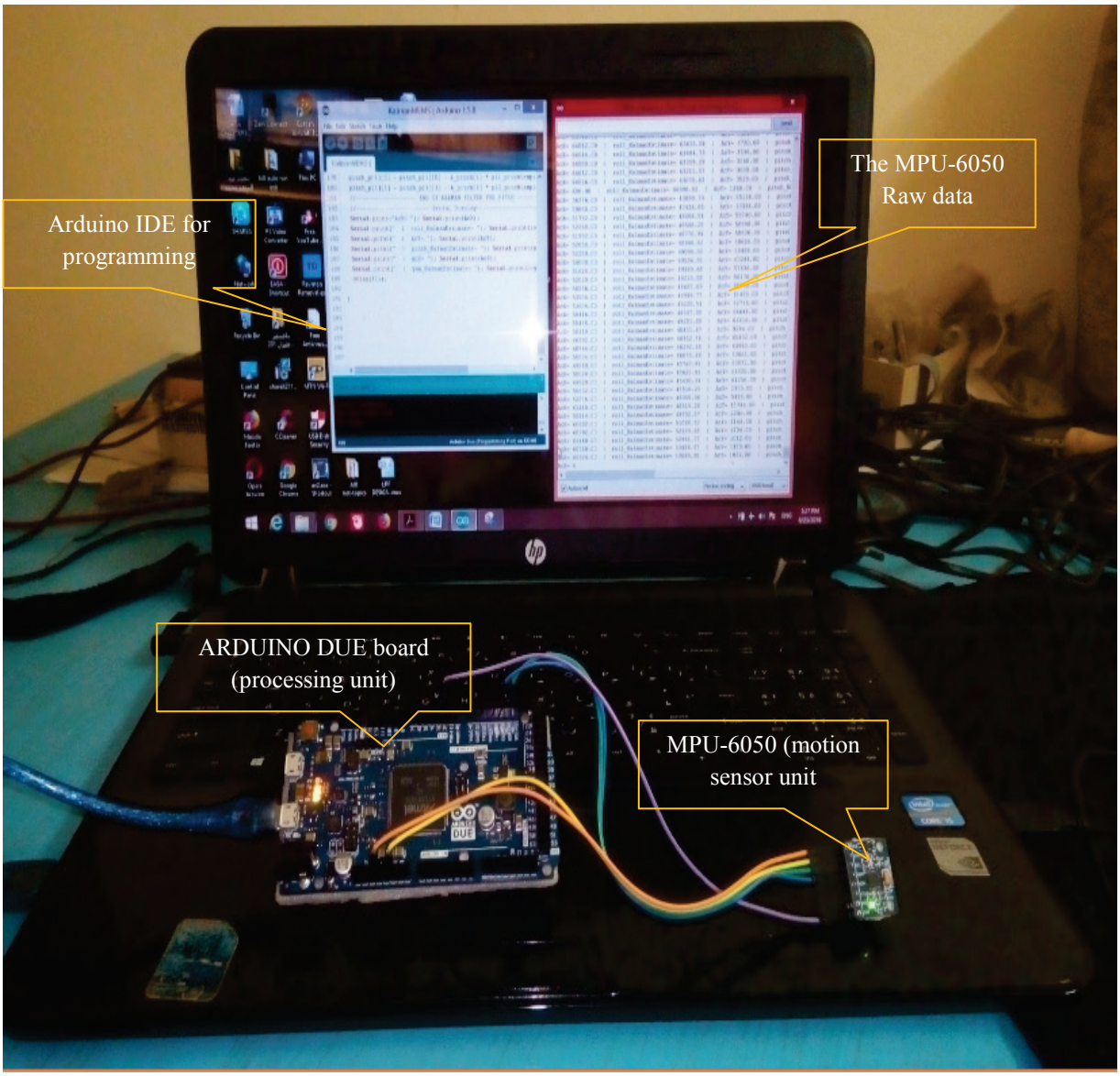
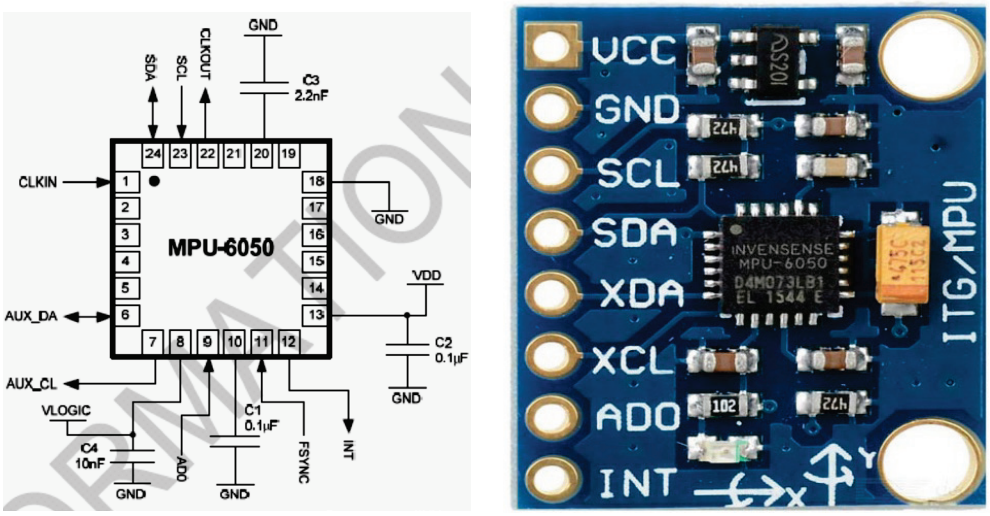


Figure (3.1): Embedded Test platform MEMS-based Accelerometer



(A) (B)

Figure (3.2): (A) The InvenSense MPU-6050 MEMS Sensor (B) MPU-6050 typical interface Board [15].

Table (3.1): Pin layout and signal description of the MPU-6050

Pin No.	PIN Name	Description
13	VCC	Power supply voltage and Digital I/O supply voltage
18	GND	Power supply ground
23	SCL	I^2C serial clock
24	SDA	I^2C serial data
6	XDA (AUX_DA)	I^2C master serial data, for connecting to external sensors
7	XCL (AUX_CL)	I^2C Master serial clock, for connecting to external sensors
9	ADO	I^2C Slave Address LSB
12	INT	Interrupt digital output

3.3. Arduino DUE Board

The Arduino DUE is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit Advanced RISC Machine (ARM) core microcontroller. It has 54 digital input/output pins (of which 12 can be used as Pulse Width Modulation (PWM) outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.

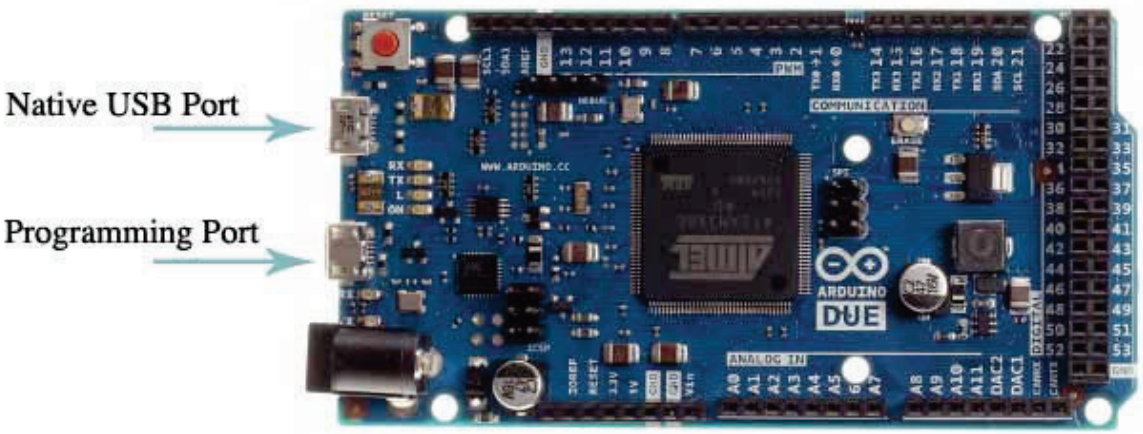
The board contains everything needed to support the microcontroller; it is connected to the computer with a USB cable or by powers it with an AC-to-DC adapter or battery to get started. The Atmel AT91SAM3X8E has 512 KB (2 blocks of 256 KB) of flash memory for storing code. The boot loader is pre-burned in factory from Atmel and is stored in a dedicated ROM memory. The available SRAM is 96 KB in two contiguous banks of 64 KB and 32 KB. All the available memory (Flash, RAM and ROM) can be accessed directly as a flat addressing space.

The DUE is compatible with all Arduino shields that work at 3.3V and are compliant with the 1.0 Arduino pin-out as follow:

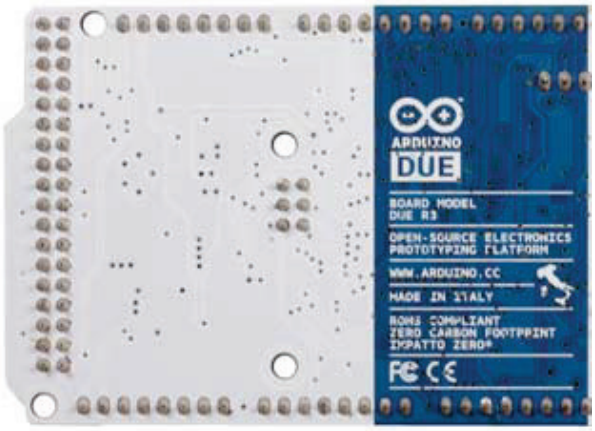
- TWI: SDA and SCL pins that are near to the AREF pin.
- The IOREF pin which allows an attached shield with the proper configuration to adapt to the voltage provided by the board. This enables shield compatibility with a 3.3V board like the DUE and AVR-based boards which operate at 5V.
- An unconnected pin, reserved for future use.
- The DUE has a 32-bit ARM core that can outperform typical 8-bit microcontroller boards.

The most significant differences are:

- A 32-bit core, that allows operations on 4 bytes wide data within a single CPU clock.
- CPU Clock at 84Mhz.
- 96 KBytes of SRAM.
- 512 KBytes of Flash memory for code.
- a DMA controller, that can relieve the CPU from doing memory intensive tasks [16].



(A)



(B)

Figure (3.3): (A) front, and (B) back side of Arduino DUE Board [16]

3.4. Development of denoising MEMS test platform

For the purpose of this thesis, a test bed for the proposed wavelet based technique as well as the other denoising techniques. The test bed development can be implemented in many stages which include: sub-systems interconnections and communications for data acquisition and system demonstration as follow:

i. Sub-systems interconnection

The connection between Arduino DUE Board and the MPU-6050 is shown in the Figure (3.4), and the pins connections are shown in the Table (3.2).

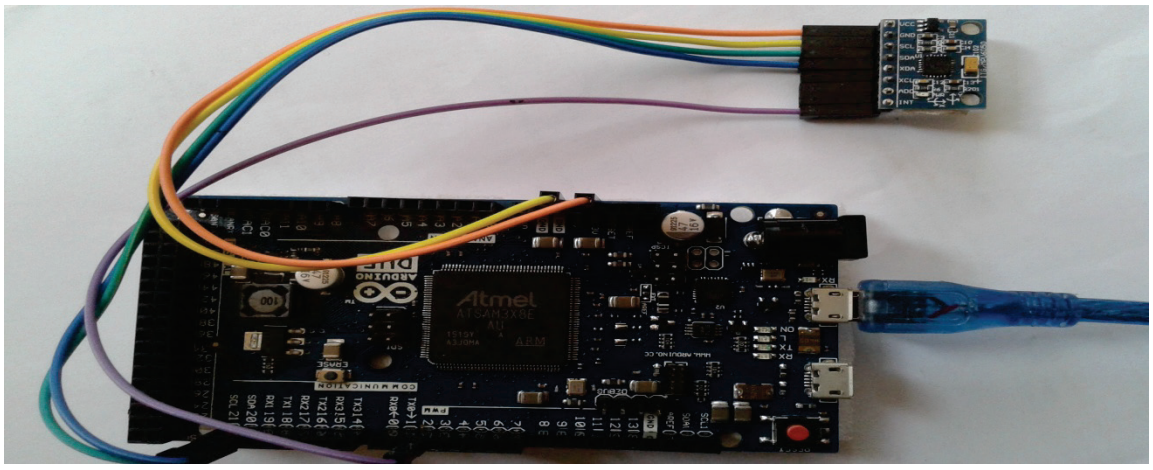


Figure (3.4): The Connection of Arduino DUE Board with MPU-6050

The MPU-6050 board uses the communication standard I^2C to communicate to other devices. The I^2C connection uses a two wire topology with a line for a clock signal (SCL) and a line for the data signal.(SDA) Using only two wires limits the I^2C to being a half-duplex communication line. And Because of the Arduino DUE is 32 bit architecture and its 84 MHz clock speed, the Arduino DUE board was chosen. Its 32-bit architecture is useful when considering the MEMS MPU-6050 (16) bit data output for each of its accelerometer and gyroscope outputs [15].

Table (3.2): Pin connection Arduino DUE VS MPU-6050

Arduino DUE	MPU-6050
VCC 3.3	VCC
GND	GND
SCL (pin 21)	SCL
SDA (pin 20)	SDA
PWM (pin 2)	INT

ii. Installation of I^2C dev & MPU-6050 Libraries

If the code was to be written from scratch, it would take ages and there would be a lot of reverse engineering required to make good use of the module's proprietary Digital Motion Processing (DMP) engine because InvenSense intentionally released minimal data on its MPU-6050. The good thing is that the hard work has already been done for by; Jeff Rowberg who wrote some Arduino libraries to obtain the accelerometer / gyro data and handle all the calculations, they are available as a zip file from the (github) website source cited in reference [18].

Once unzipped, the Arduino folder was found within it and copied the two folders "I^2Cdev" and "MPU6050" over to the Arduino "libraries" folder in the following directory: C:\Program Files (x86)\Arduino\libraries.

Then the Arduino IDE was opened and in the examples section MPU6050_DMP6 was found within MPU-6050, it was then opened, Arduino was plugged in, the appropriate com port was selected and the sketch was uploaded. In the serial window, a baud rate of 9600 was selected. It was prompted that the MPU-6050 connection is successful. The data collection was

tested by typing any key in the text bar and pressing enter, the data then started showing up as in figure (3.5).

The sensor MPU-6050 was connected with an Arduino DUE board and suitable codes were uploaded, then the output data at the three axes X, Y and Z sensing MEMS were collected through MATLAB and arduino IDE. In figure (3.6) shows Matlab program flowchart for MEMS MPU-6050 data i.e. acceleration and gyroscope data acquisition and applying denoising techniques i.e. Kalman and Wavelet was shown,

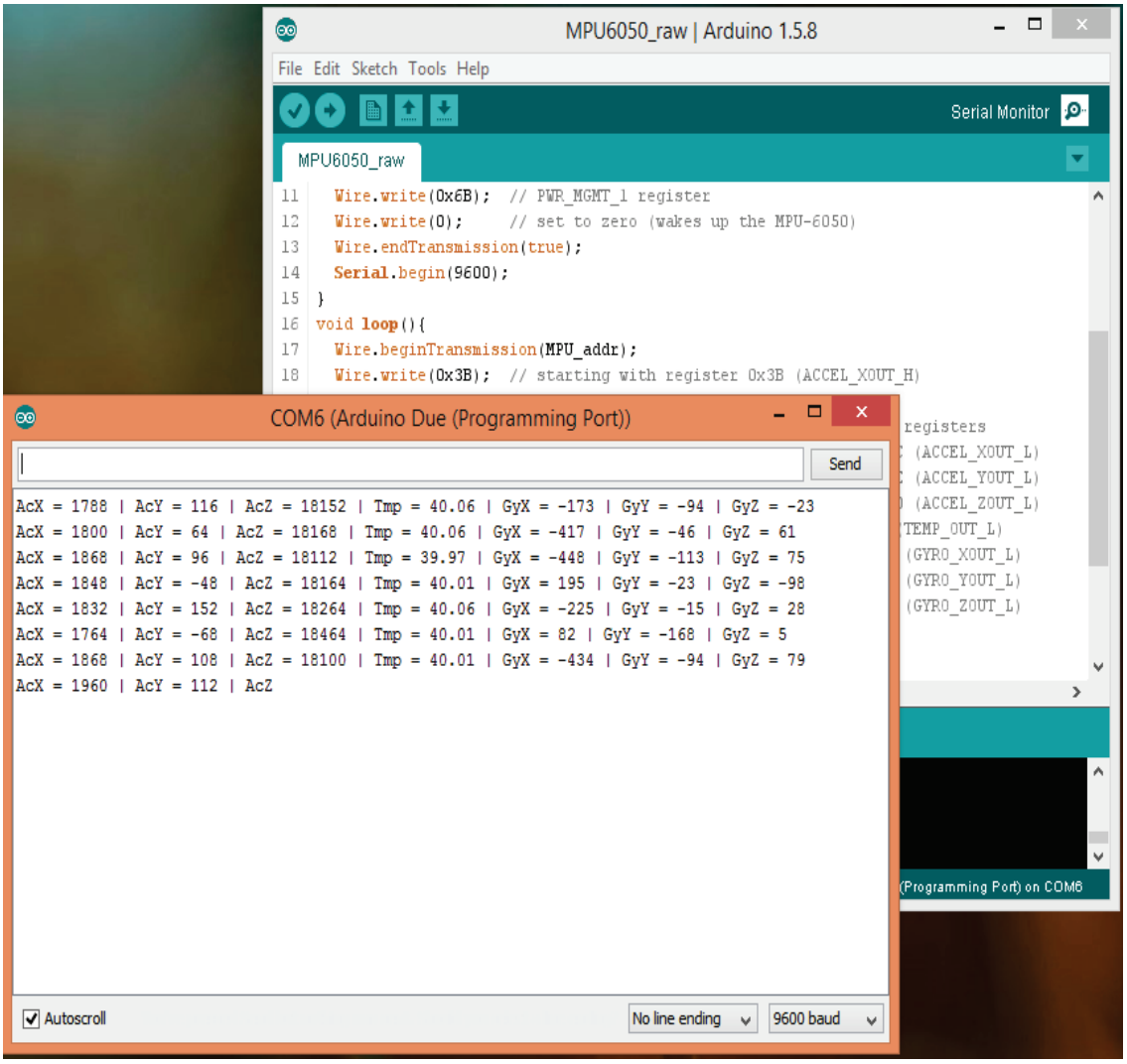


Figure (3.5): The MPU-6050 raw data at the serial monitor of Arduino IDE.

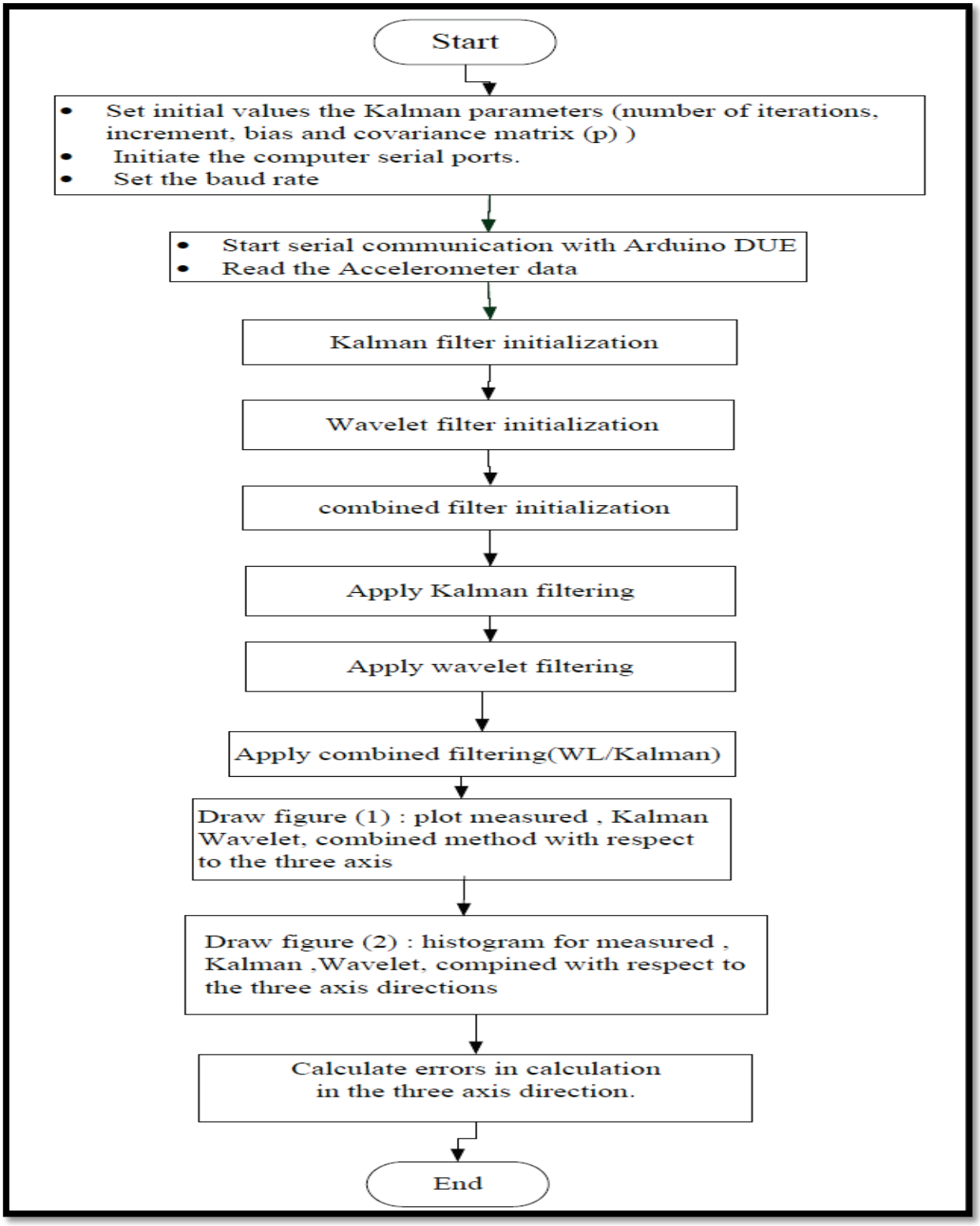


Figure (3.6): MATLAB program flowchart of accelerator data acquisition and applying denoising techniques**.**

iii. Development of MATLAB Simulation Test Bed

The used denoising techniques in this thesis were simulated in MATLAB platform to explore the performance for each one, those are:, Kalman filter (KF) and Wavelet-based filter (WF) in addition to Kalman–wavelet filter combination

For the purpose of wavelet denoising the MATLAB function ***"wden( )"*** is used. It performs an automatic denoising process of a one-dimensional signal using wavelets with certain parameters. These parameters were formed in different proposed test scenarios in order to find the optimum denoised signal. The most important parameter is the threshold which can be selected based on:

- Stein's Unbiased Estimate of Risk (quadratic loss function) which gets an estimate of the risk for a particular threshold value (λ). Minimizing the risks in (λ) and gives a selection of the threshold value. This method of selection rule is expressed as ***'rigrsure'***.
- The other selection criteria is one that uses a fixed-form threshold yielding minimax performance multiplied by a small factor proportional to ***log(length(X))***. This one is expressed as ***'sqtwolog'***.
- The third selection criteria use a fixed threshold chosen to yield minimax performance for mean square error against an ideal procedure. This minimax principle is used in statistics in order to design estimators. Since the denoised signal can be assimilated to the estimator of the unknown regression function, the minimax estimator is the one that realizes the minimum of the maximum mean square error obtained for the worst function in a given set. This criterion is expressed as ***'minimaxi'***.
- The forth one is ***'heursure'***. It is a mixture of the two options ***'rigrsure'***, and ***'sqtwolog'***, which is already addressed in the previous methods.

In addition, the thresholding criteria can be: soft or hard (***'s' or 'h'***). Also in these scenarios, a scale parameter is used **'scal'**. It defines multiplicative threshold rescaling with the following options: **'*one*'** for no rescaling. ***'sln'*** for rescaling using a single estimation of level noise based on first-level coefficients. And the last option is ***'mln'*** for rescaling using level-dependent estimation of level noise. In addition, the Wavelet decomposition is performed at **level (N)** for this thesis level (5) will be used as default where ***'wname'*** is a character vector containing the name of the desired orthogonal wavelet function, therefore , a complete wavelet denoising function in MATLAB syntax can be given by:

$$X_D = wden(X,TPTR,SORH,SCAL,N,'wname')$$

which returns a denoised version X_D of input signal X obtained by thresholding the wavelet coefficients. For generality, a wavelet family Daubechies wavelets ***dbN*** **(db 4/8)** will be used where it covers most of the wavelet properties as can be grasped from Appendix (D) [19].

Based on the above options a set of twelve test scenarios will be proposed to investigate denoising response. The good scenarios out the twelve will be selected to compose the best scenario. To explore the effect of decomposition level on the wavelet denoised acceleration, another two levels will be investigated i.e. Level (3), and level (7) as well as level (5).

The results of these proposed scenarios will be presented using the denoised signal comparable to the measured one, their histograms, and basic statistical measurements such as average and standard deviation (SD) in a tabulation manner.

iv. Implementation of Denoising techniques at Arduino DUE:

Finally, the outperformed denoising method by the running simulation will be implemented in Arduino DUE / MPU-6050 test platform to evaluate its performance in a real time at single microcontroller platform.

CHAPTER FOUR

RESULTS AND DISCUSSION

CHAPTER FOUR

RUSULTS AND DISCUSSION

4.1. Introduction

In this Chapter, the simulation scenarios of denoising the MPU-6050 acceleration measures using Kalman, Wavelet and combination of Kalman-Wavelet were conducted. Moreover, for more exploration on Wavelet denoising technique, additional scenarios were addressed. The obtained results shown the optimal denoising method which will be implemented on the proposed test bed of single microcontroller i.e. Arduino DUE.

4.2. Results of Denoising Technique Using Kalman Filter

By using the developed Kalman filter denoising algorithm which compiled in MATLAB based program at appendix (A), Figure (4.1) shows the measured and estimated Z-Axis, X-Axis and Y-Axis from MPU-6050 and Kalman filter respectively.

It can be noticed that the Kalman denoising filter achieves an improvement on the acceleration measurements where the contaminated noise was partially filtered. More precisely, the performance of the Kalman filter in denoising the X-Axis measures is performed its impact on the Y-Axis and Z-Axis by reading the standard deviation of the denoised measures as **25.3119** mg, **46.4524** mg and **51.2433** mg respectively as in Table (4.1).

Moreover, the distribution of the measured and denoised Kalman estimated one was shown in Figure (4.2). The Z-Axis and Y-Axis measures are more vulnerable the noise – more likely to the Gaussian distribution- at MPU-6050 than the X-Axis which is more responsive to the Kalman estimator as can be grasped from Figure (4.2).

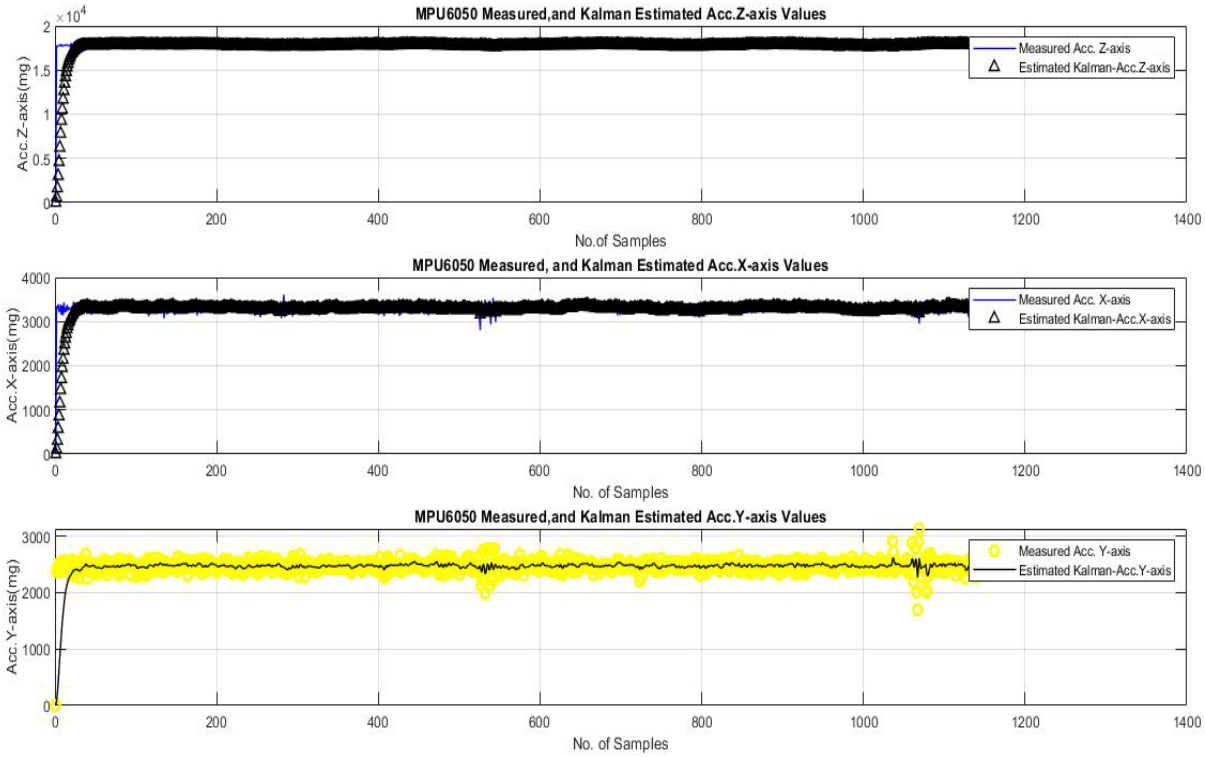


Figure (4.1): Result of MPU-6050 accelerometer measured and denoised signal using Kalman estimator.

Table (4.1): Results of Measured and Kalman Estimated Values

Acceleration Values	**Average (*mg*)**		**Standard Deviation (SD)(*mg*)**	
	Measured	Kalman Filtered	Measured	Kalman Filtered
Z-Axis	1.8116e+04	1.8093e+04	117.3916	51.2433
X-Axis	2.3047e+03	2.3036e+03	167.9994	25.3119
Y-Axis	932.1569	938.7714	229.3587	46.4524

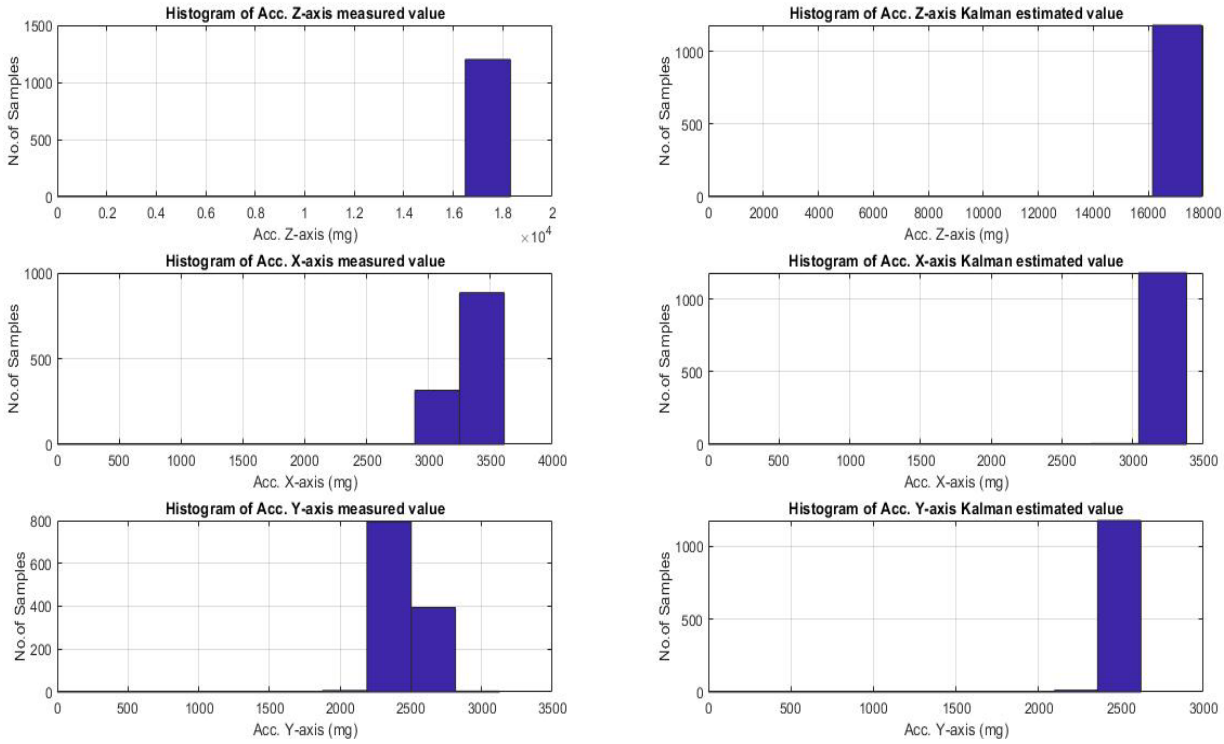


Figure (4.2): Histogram of MPU-6050 accelerometer measured and denoised signals using Kalman estimator

4.3. Results of Denoising Technique Using Wavelet and Kalman-Wavelet Combination:

In this subsection, a set of different test scenarios were established to explore the optimum performance of these different denoising techniques for the MPU - 6050 Accelerometer as well as recording the optimal parameters of the best filtering technique which will be utilized in the implementation. The following are twelve MATLAB syntax and their related results figures as follow:

4.3.1. Test Scenario No (1):

```
%---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,'modwtsqtwolog','h','mln',5,'db8');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,'modwtsqtwolog','h','mln',5,'db8');
 %---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,'minimaxi','h','one',5,'db8');
 X-Axis _KLWLESTM = wden(X-Axis_KALMAN,'minimaxi','h','one',5,'db8');
 %---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE,'rigrsure','h','one',5,'db8');
 Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,'rigrsure','h','one',5,'db8');
```

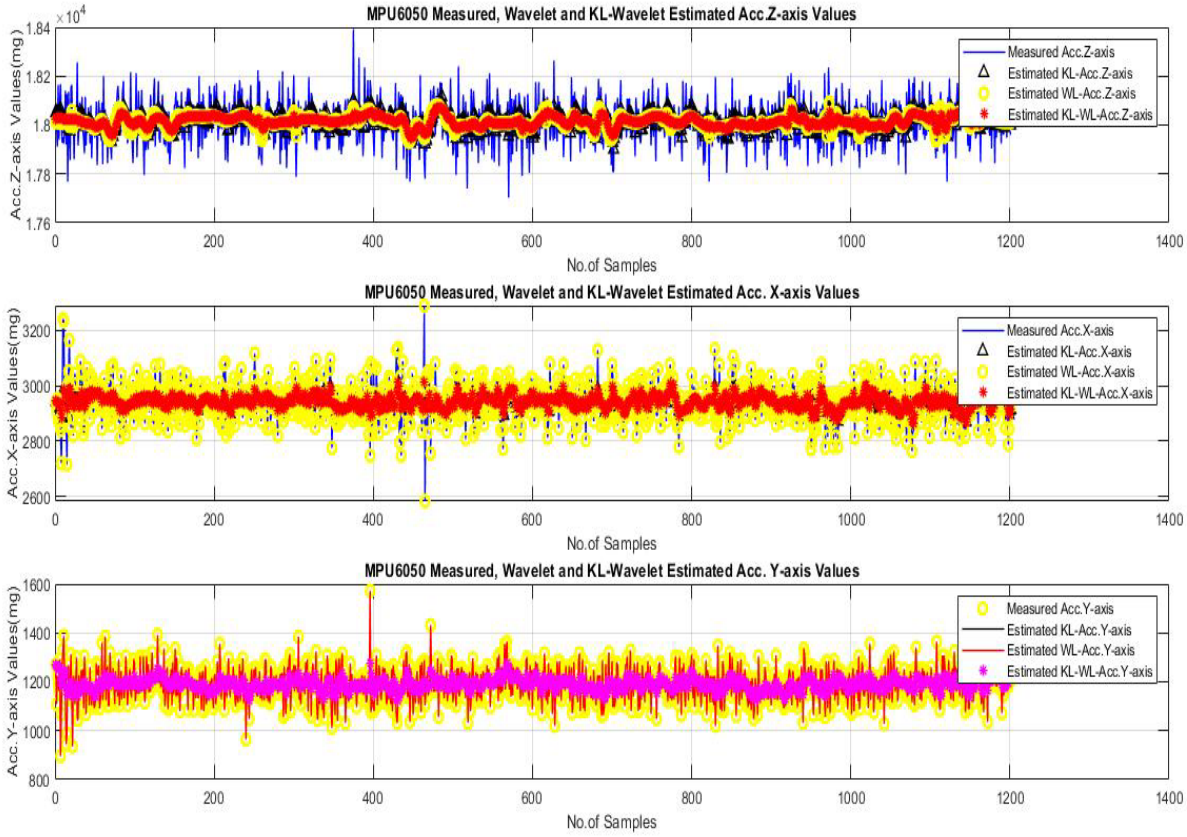


Figure (4.3): Results of test scenario No (1)

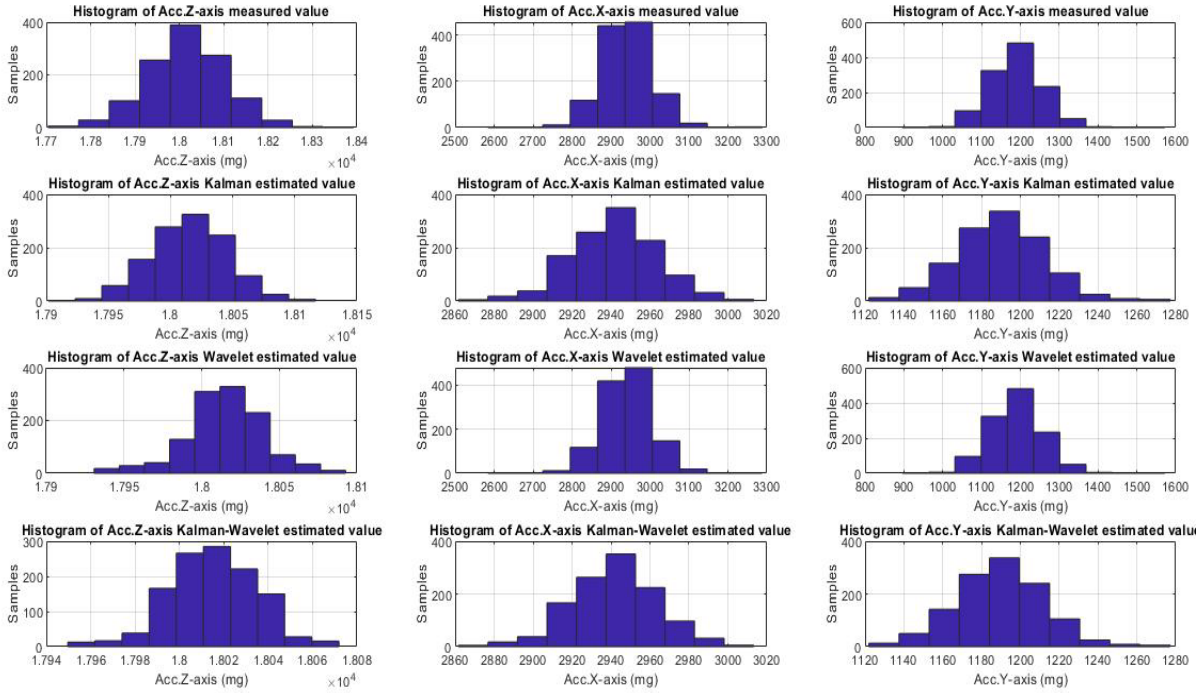


Figure (4.4): Histogram of Test scenario No (1)

4.3.2. Test Scenario No (2):

```
%---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,'modwtsqtwolog','s','mln',5,'db8');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,'modwtsqtwolog','s','mln',5,'db8');
 %---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,'minimaxi','h','sln',5,'db8');
 X-Axis _KLWLESTM = wden(X-Axis_KALMAN,'minimaxi','h','sln',5,'db8');
 %---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE,'rigrsure','h','sln',5,'db8');
 Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,'rigrsure','h','sln',5,'db8');
```

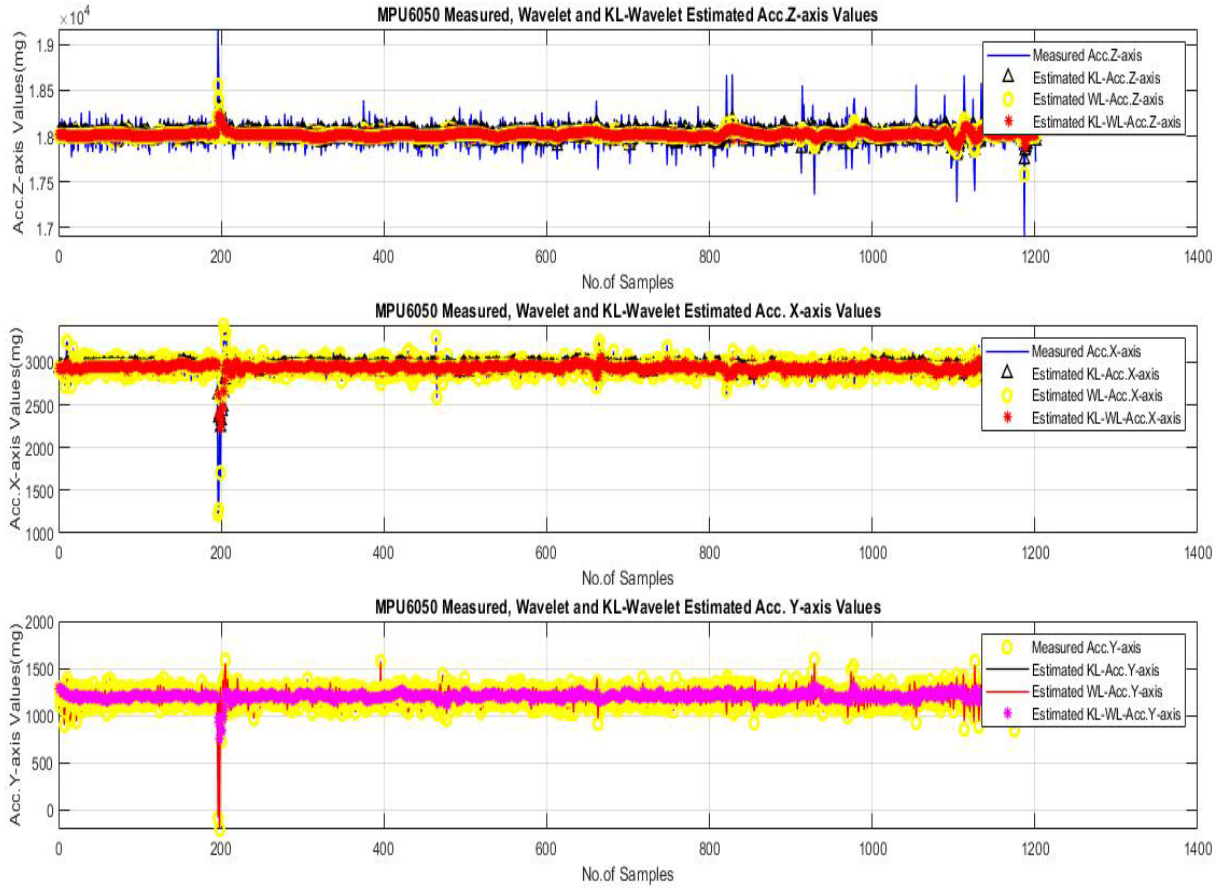


Figure (4.5): Results of test scenario No (2)

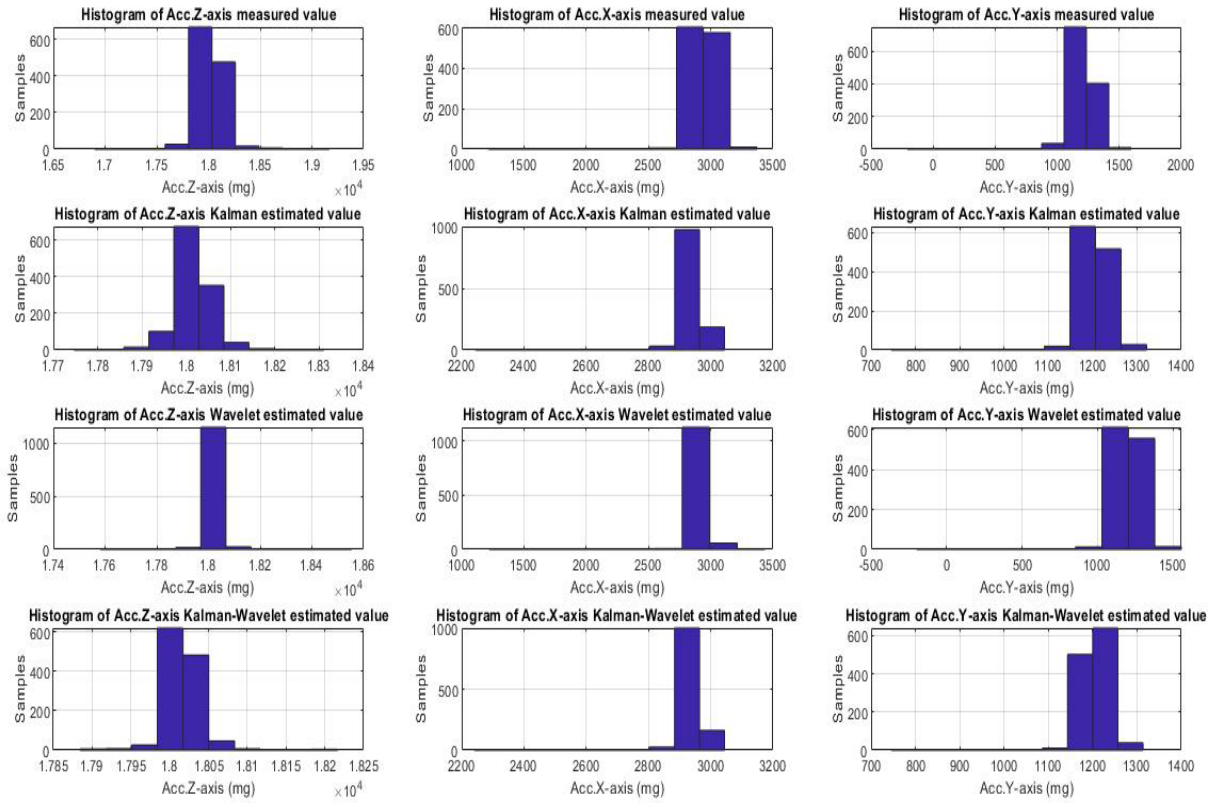


Figure (4.6): Histogram of Test scenario No (2)

4.3.3. Test Scenario No (3):

```
%---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,'minimaxi','h','mln',5,'db8');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' minimaxi ','h','mln',5,'db8');
%---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' modwtsqtwolog ','s','mln',5,'db8');
X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' modwtsqtwolog ','s','mln',5,'db8');
%---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE,'rigrsure','h','mln',5,'db8');
Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,'rigrsure','h','mln',5,'db8');
```

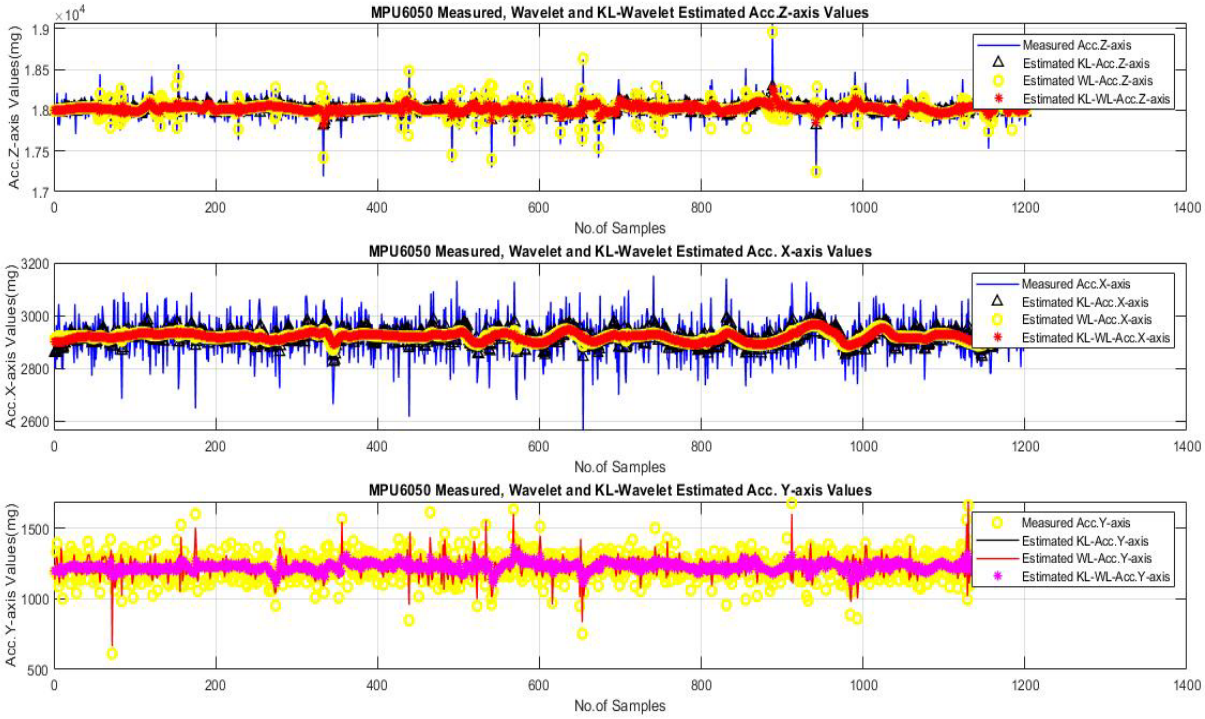


Figure (4.7): Results of test scenario No (3)

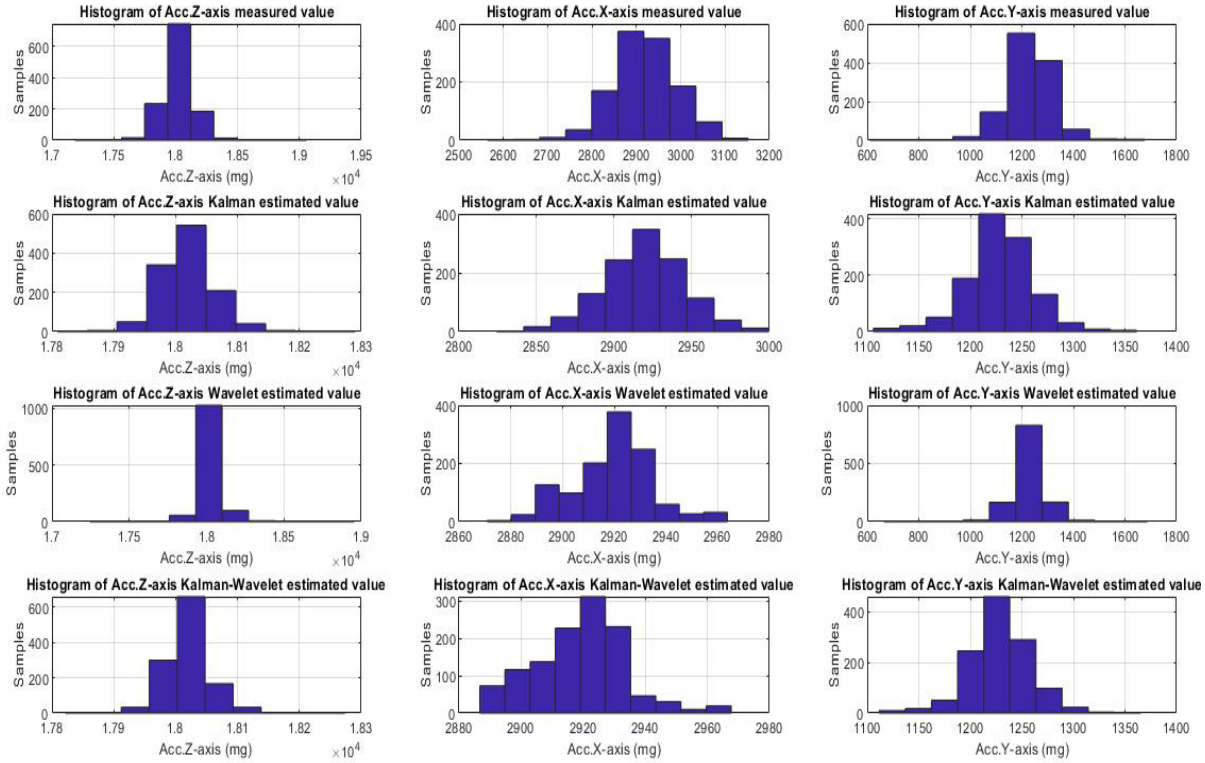


Figure (4.8):Histogram of Test scenario No (3)

4.3.4 Test Scenario No (4):

```
%---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,'minimaxi','s','one',5,'db8');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' minimaxi ','s','one',5,'db8');
%---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' modwtsqtwolog ','h','mln',5,'db8');
X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' modwtsqtwolog ','h','mln',5,'db8');
%---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE,'rigrsure','s','one',5,'db8');
Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,'rigrsure','s','one',5,'db8');
```

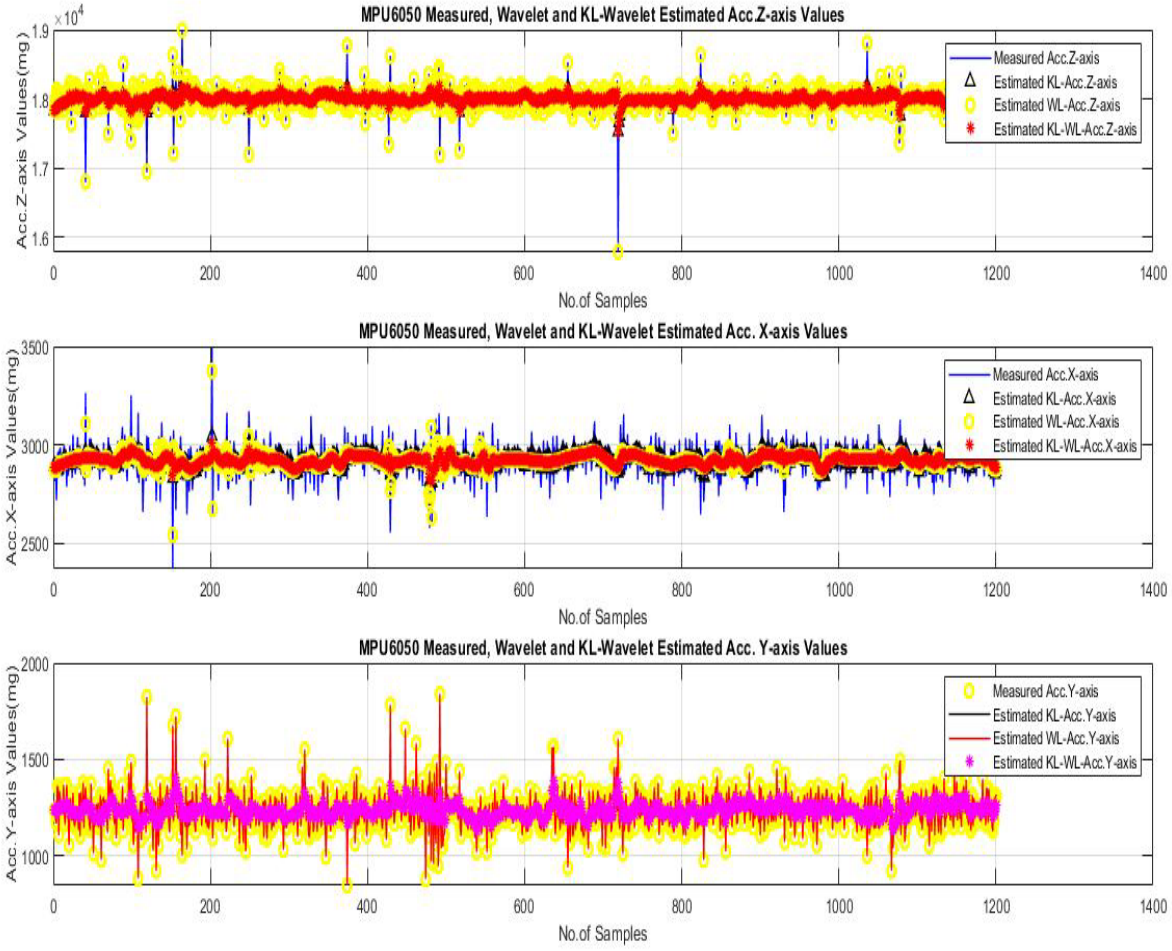


Figure (4.9): Results of test scenario No (4)

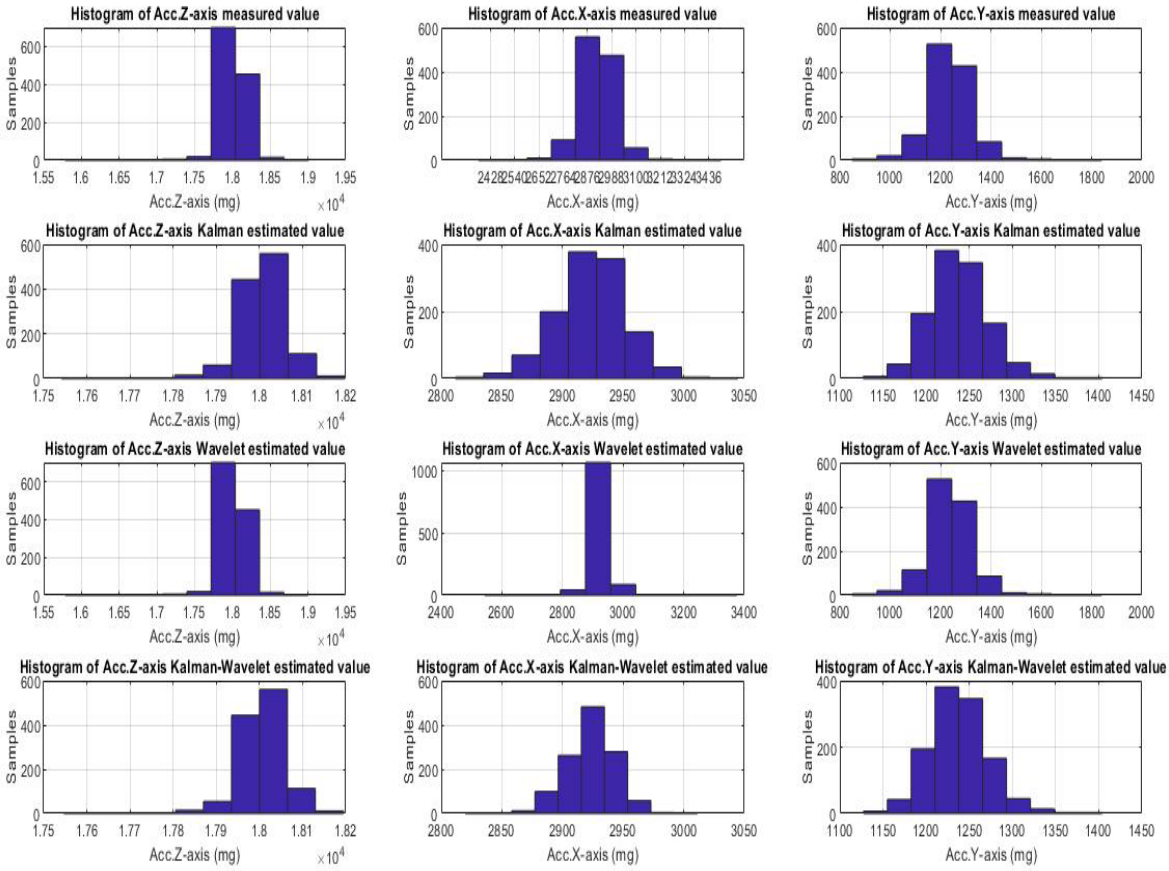


Figure (4.10):Histogram of Test scenario No (4)

4.3.5 Test Scenario No (5):

```
%---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,' rigrsure ','s','sln',5,'db8');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' rigrsure ','s','sln',5,'db8');
%---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' minimaxi ','s','sln',5,'db8');
X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' minimaxi ','s','sln',5,'db8');
%---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE, 'modwtsqtwolog ','h','mln',5,'db8');
Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,' modwtsqtwolog ','h','mln',5,'db8');
```

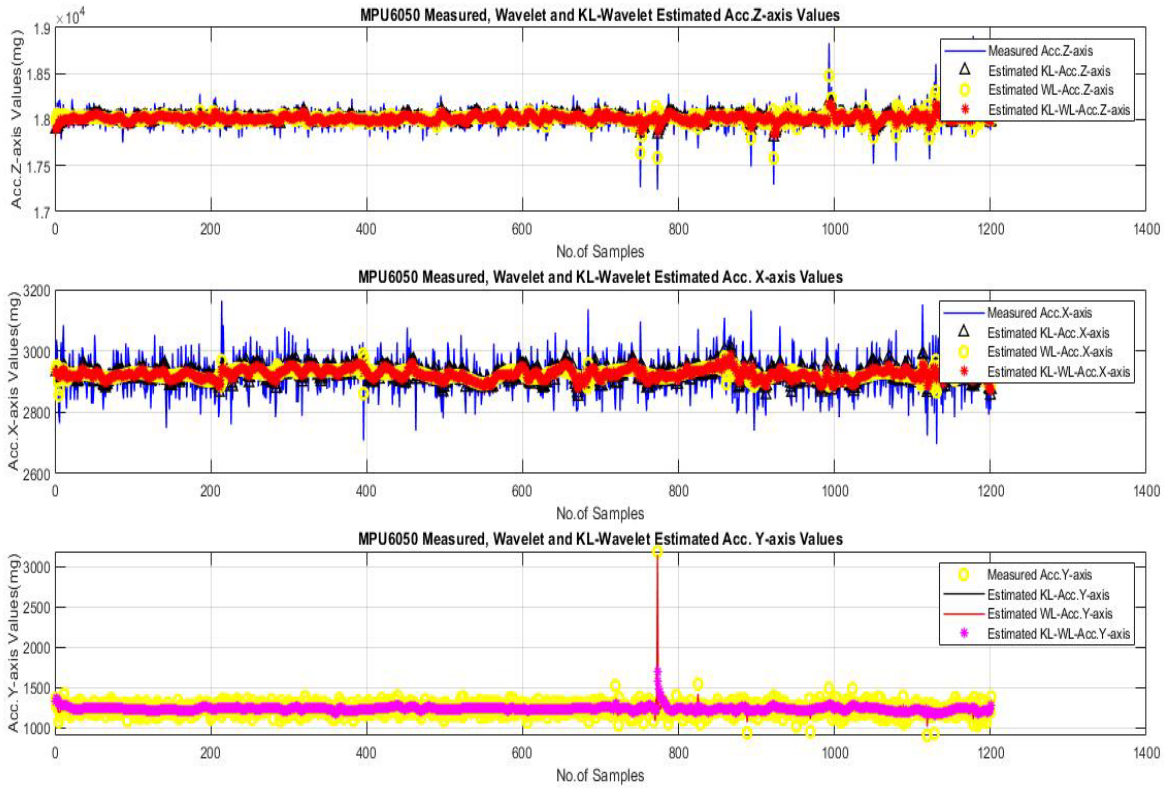


Figure (4.11): Results of test scenario No (5)

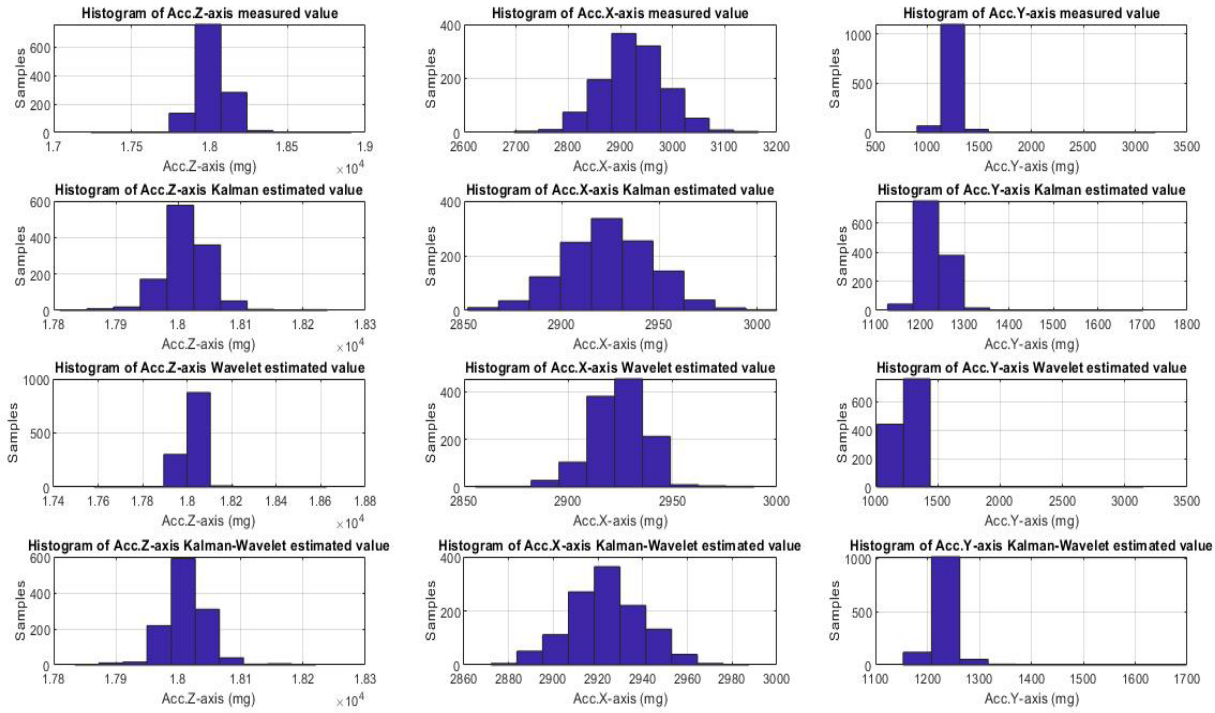


Figure (4.12): Histogram of Test scenario No (5)

4.3.6 Test scenario No (6) :

```
    %---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,' rigrsure ','s','mln',5,'db8');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' rigrsure ','s','mln',5,'db8');
    %---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' minimaxi ','s','mln',5,'db8');
 X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' minimaxi ','s','mln',5,'db8');
    %---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE, 'modwtsqtwolog ','s','mln',5,'db8');
 Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,' modwtsqtwolog ','s','mln',5,'db8');
```

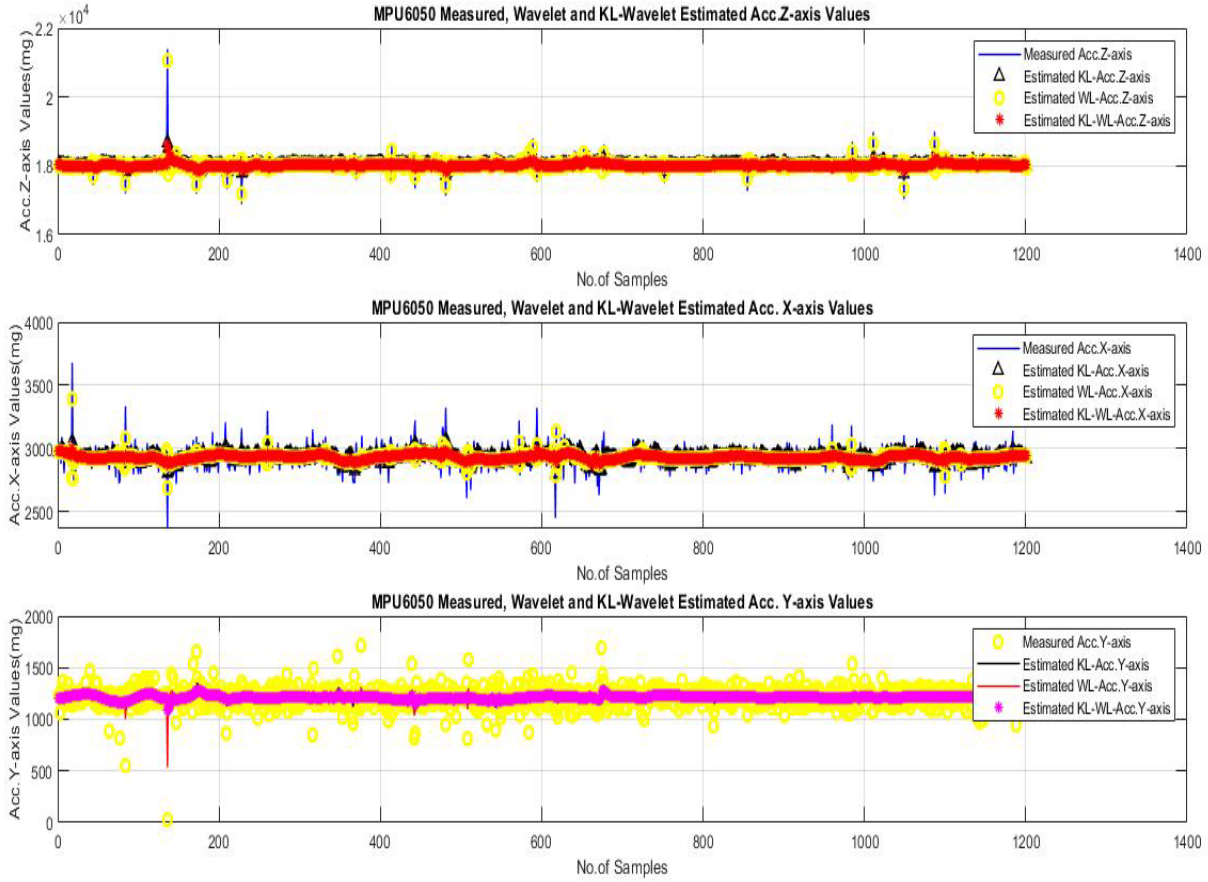


Figure (4.13): Results of test scenario No (6)

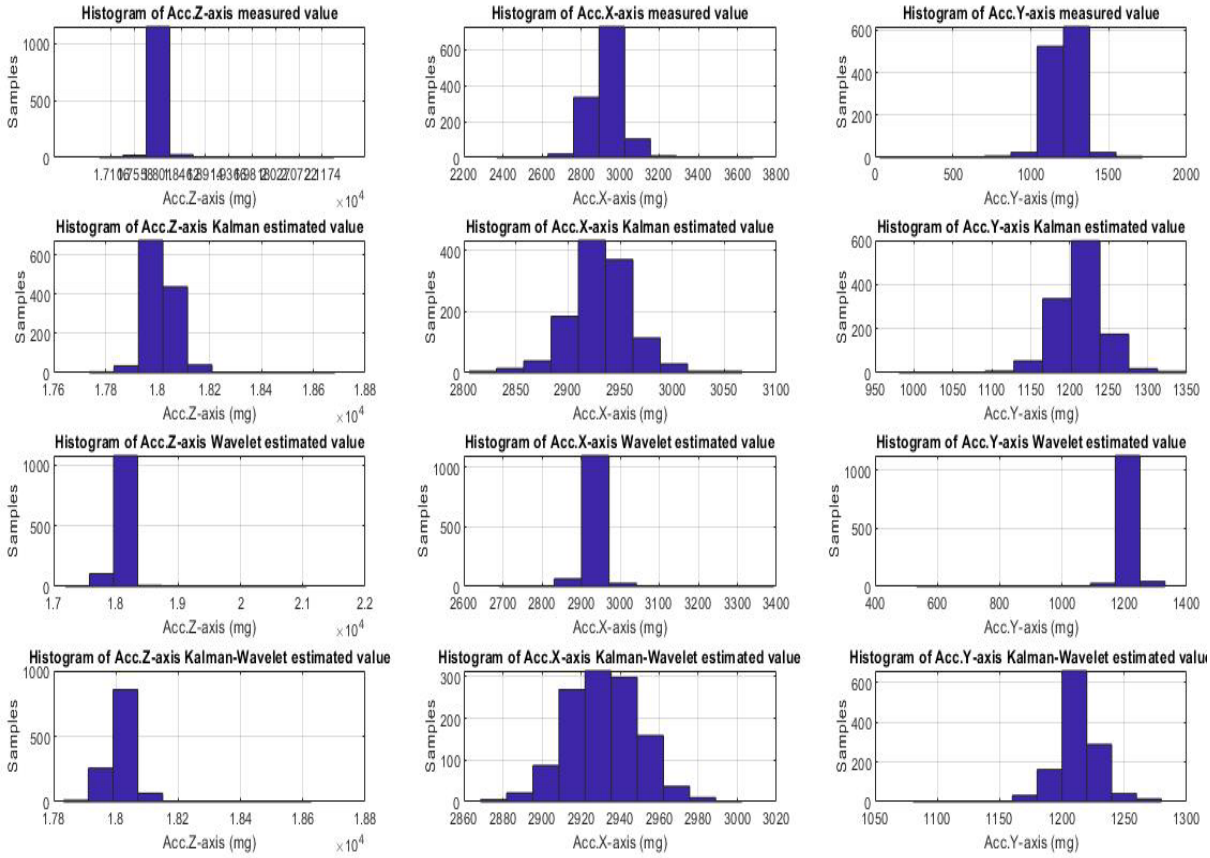


Figure (4.14): Histogram of Test scenario No (6)

4.3.7 Test Scenario No (7):

```
    %---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,' modwtsqtwolog ','s','mln',5,'db4');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' modwtsqtwolog ','s','mln',5,'db4');
     %---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' minimaxi ','s','one',5,'db4');
 X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' minimaxi ','s','one',5,'db4');
     %---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE, ' rigrsure ','s','one',5,'db4');
 Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,' rigrsure ','s','one',5,'db4');
```

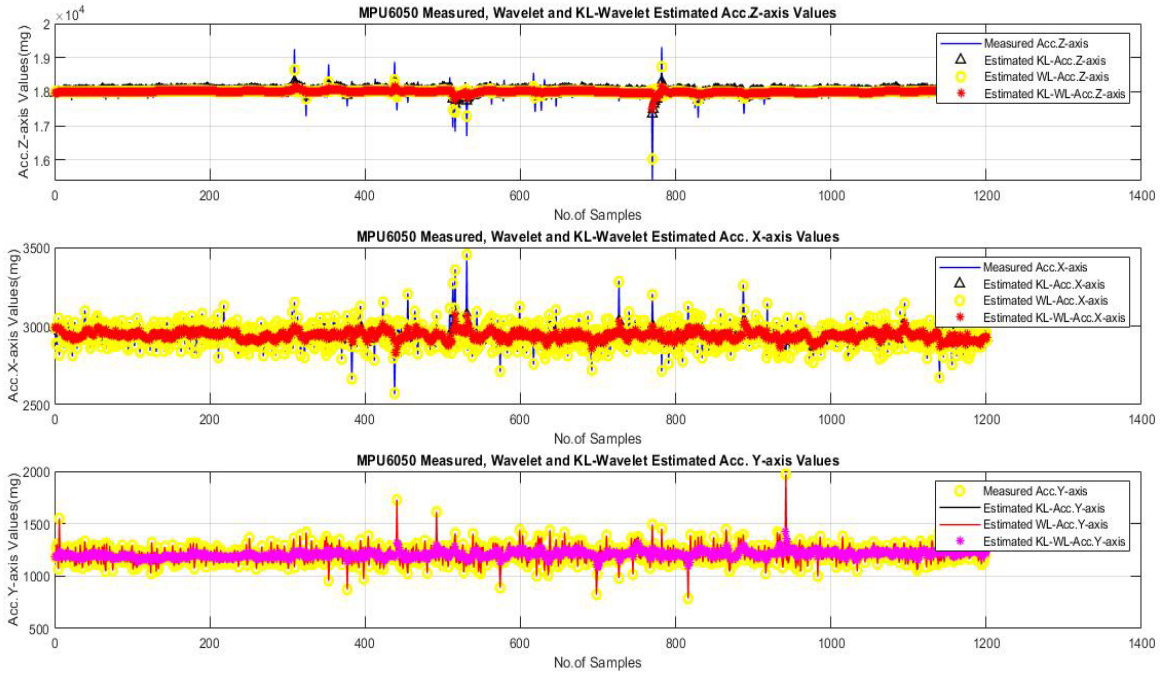


Figure (4.15): Results of test scenario No (7)

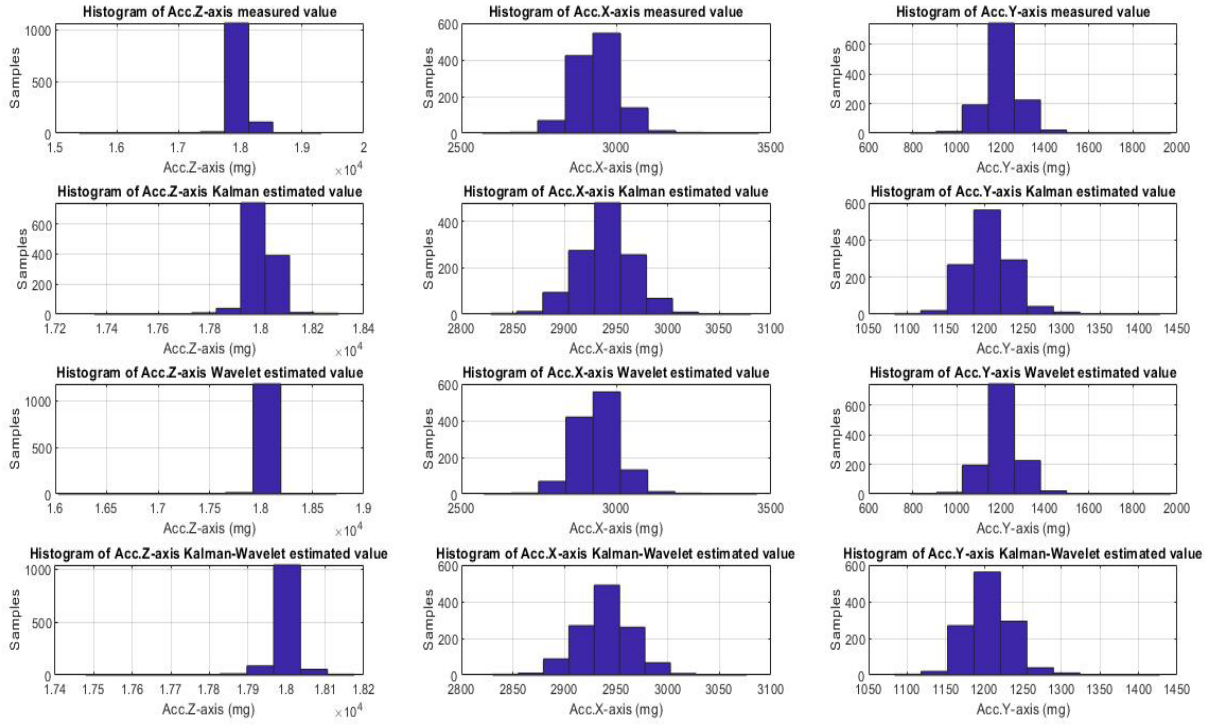


Figure (4.16): Histogram of Test scenario No (7)

4.3.8 Test Scenario No (8):

```
%---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,' modwtsqtwolog ','h','mln',5,'db4');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' modwtsqtwolog ','h','mln',5,'db4');
%---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' minimaxi ','s','sln',5,'db4');
X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' minimaxi ','s','sln',5,'db4');
%---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE, ' rigrsure ','s','sln',5,'db4');
Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,' rigrsure ','s','sln',5,'db4');
```

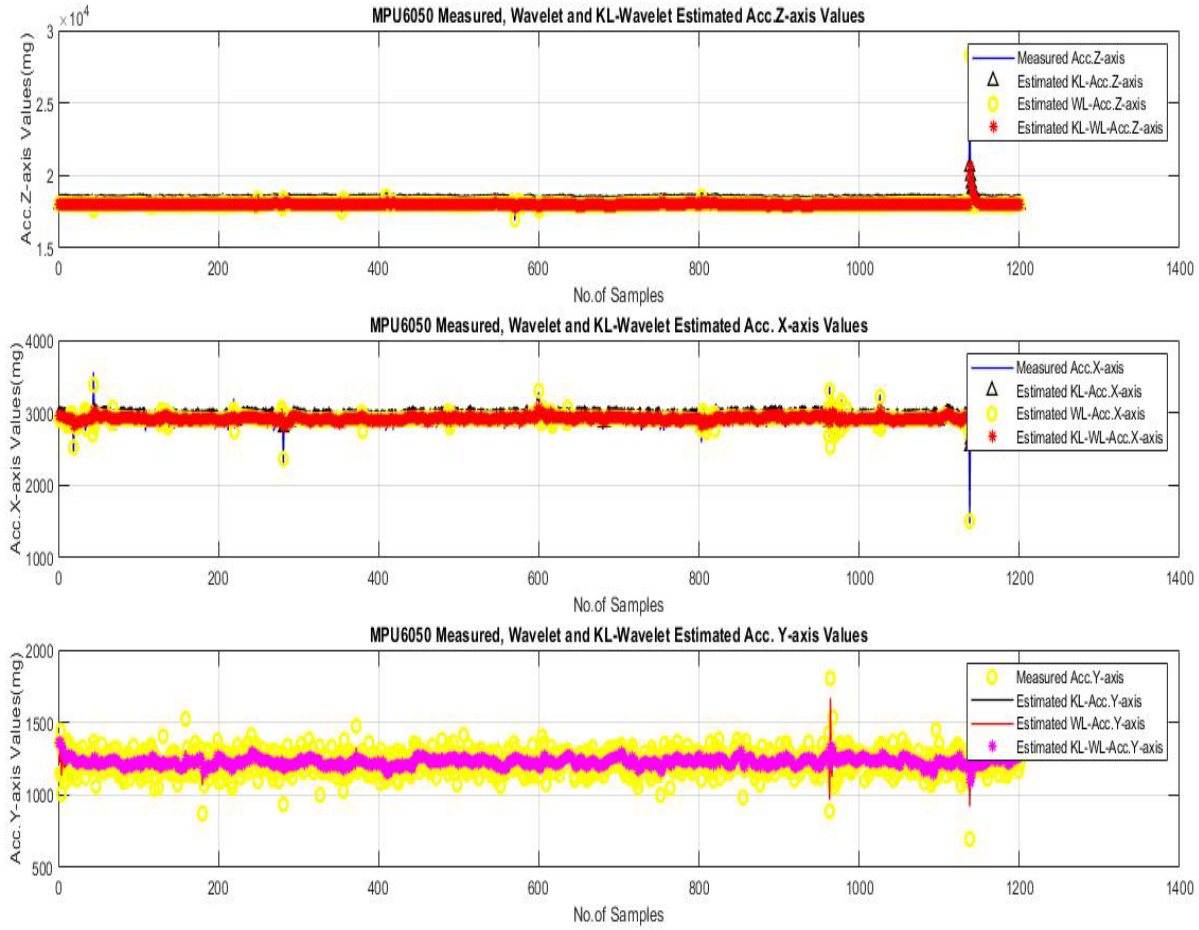


Figure (4.17): Results of test scenario No (8).

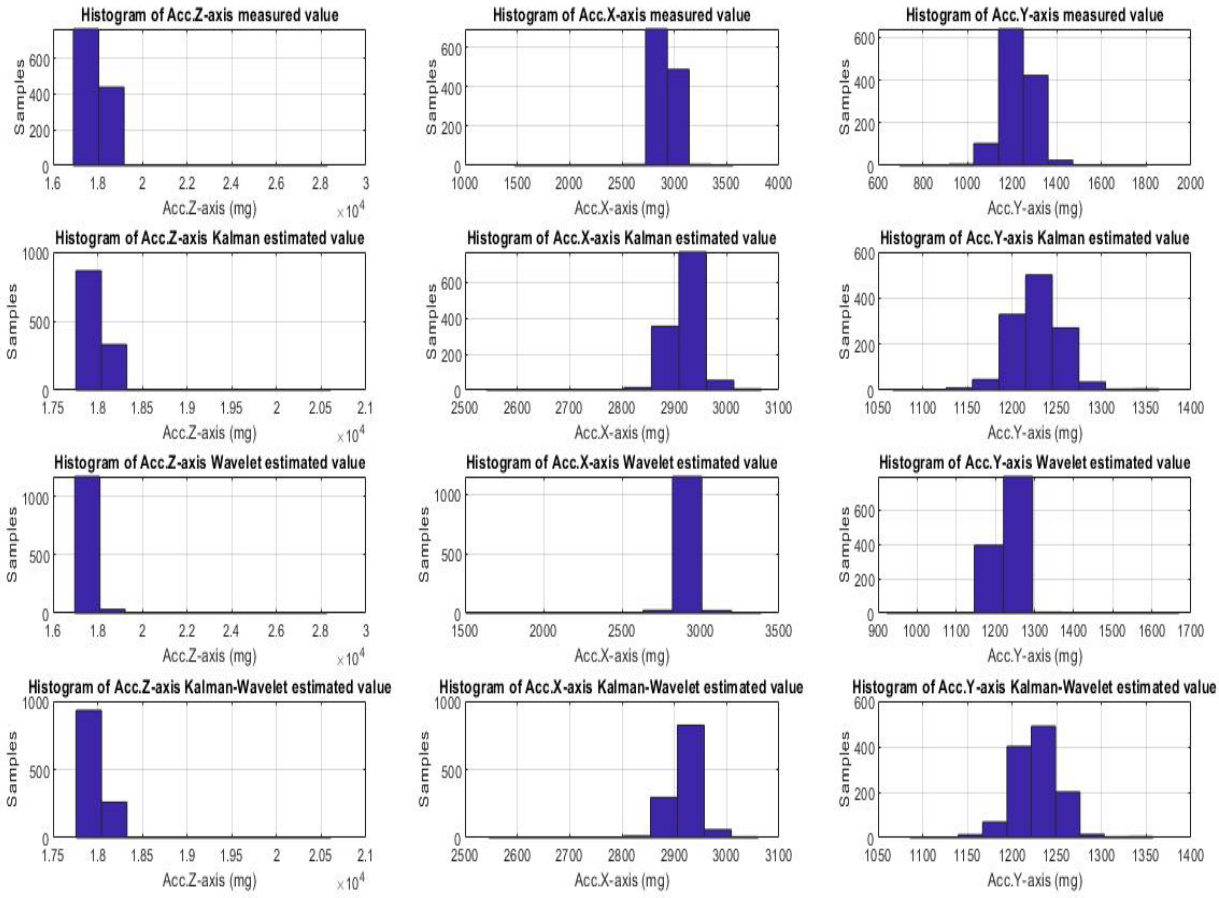


Figure (4.18): Histogram of Test scenario No (8).

4.3.9 Test Scenario No (9) :

```
    %---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,' minimaxi ','s','mln',5,'db4');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' minimaxi ','s','mln',5,'db4');
     %---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' modwtsqtwolog ','s','mln',5,'db4');
 X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' modwtsqtwolog ','s','mln',5,'db4');
     %---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE, ' rigrsure ','s','mln',5,'db4');
 Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,' rigrsure ','s','mln',5,'db4');
```

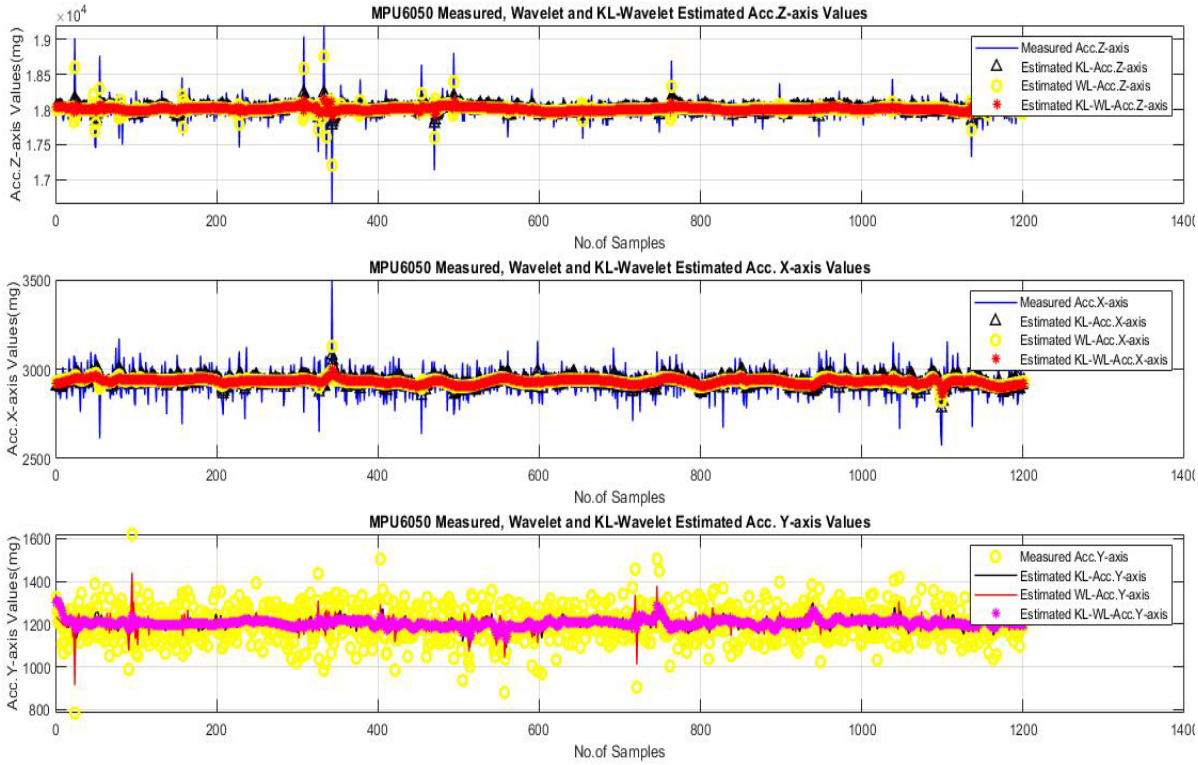


Figure (4.19): Results of test scenario No (9)

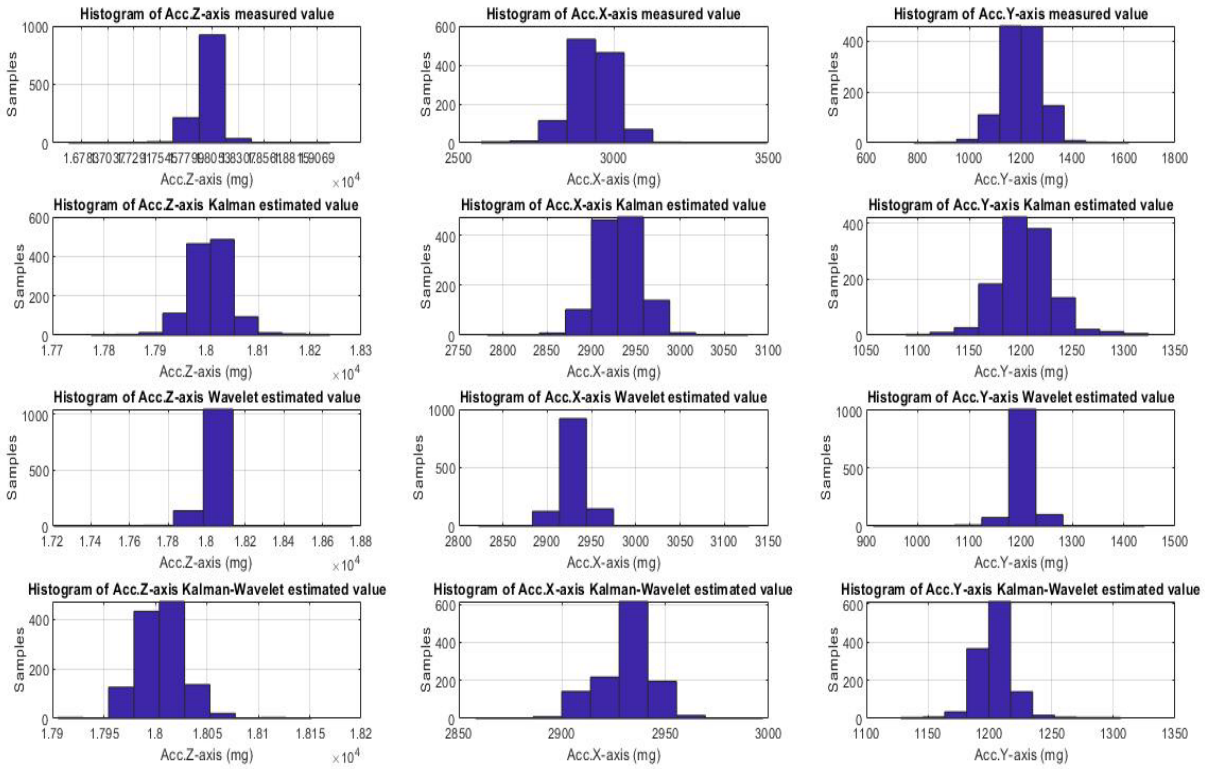


Figure (4.20): Histogram of Test scenario No (9)

4.3.10 Test Scenario No (10):

```
    %---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,' minimaxi ','h','one',5,'db4');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' minimaxi ','h','one',5,'db4');
     %---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' modwtsqtwolog ','h','mln',5,'db4');
 X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' modwtsqtwolog ','h','mln',5,'db4');
     %---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE, ' rigrsure ','h','one',5,'db4');
 Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,' rigrsure ','h','one',5,'db4');
```

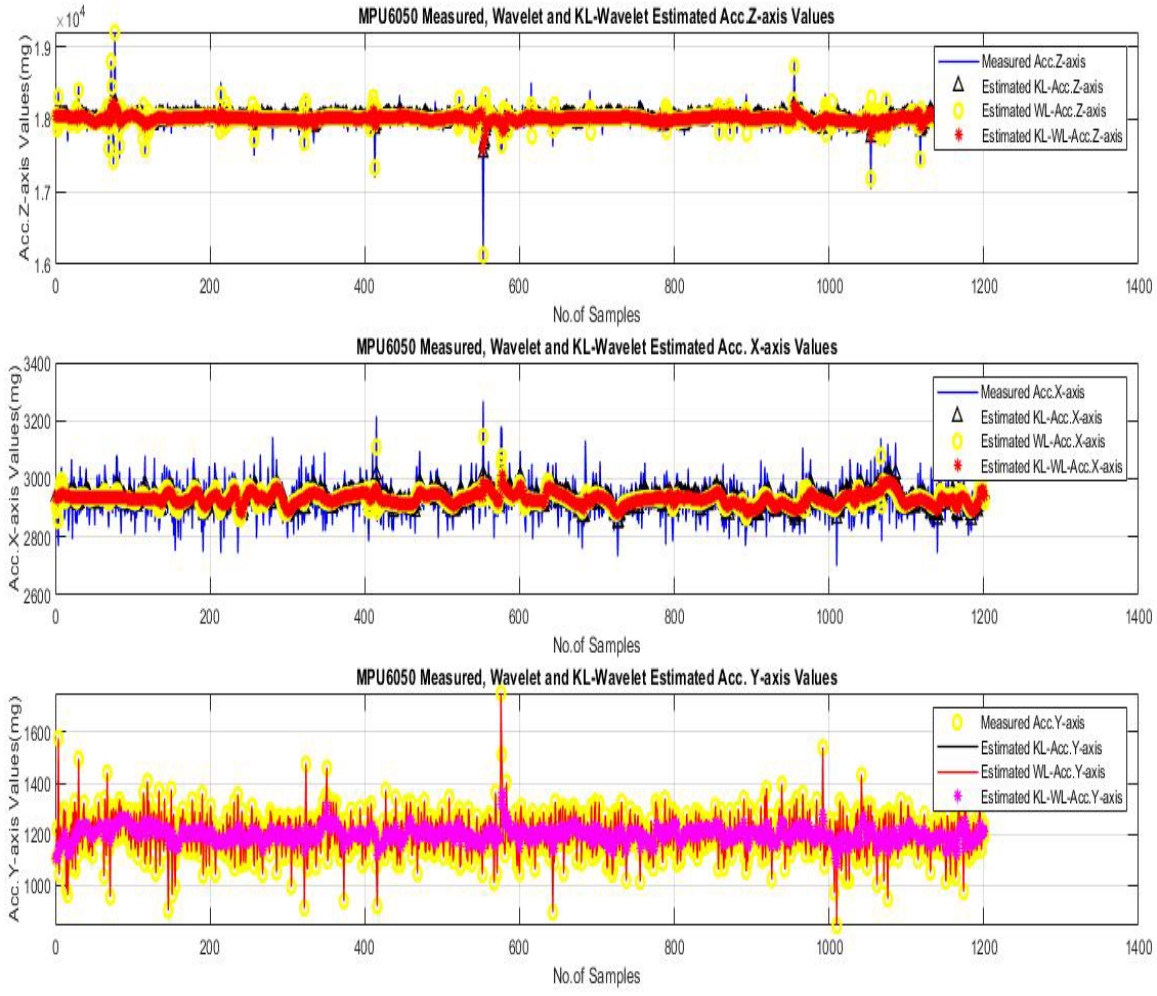


Figure (4.21): Results of test scenario No (10)

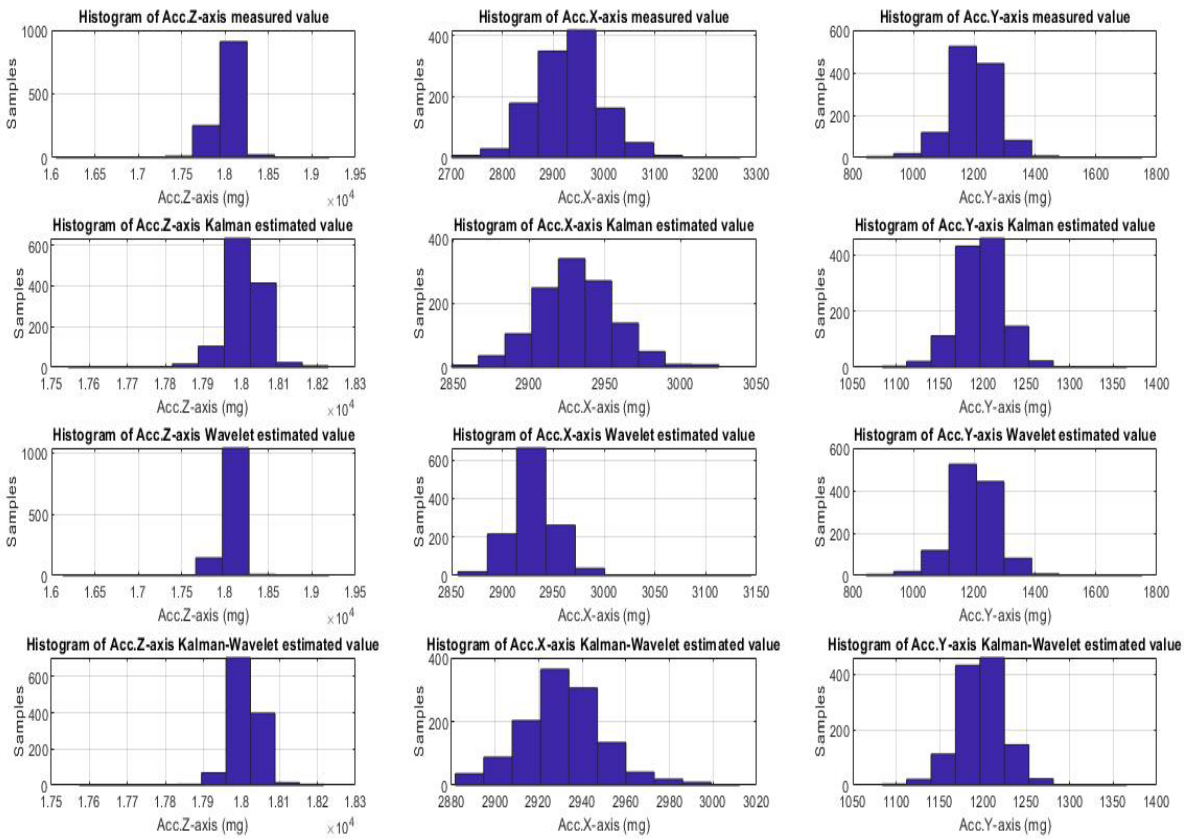


Figure (4.22): Histogram of Test scenario No (10).

4.3.11 Test Scenario No (11) :

```
%---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,' rigrsure ','h','sln',5,'db4');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' rigrsure ','h','sln',5,'db4');
%---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' minimaxi ','h','sln',5,'db4');
X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' minimaxi ','h','sln',5,'db4');
%---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE, 'modwtsqtwolog ','s','mln',5,'db4');
Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,' modwtsqtwolog ','s','mln',5,'db4');
```

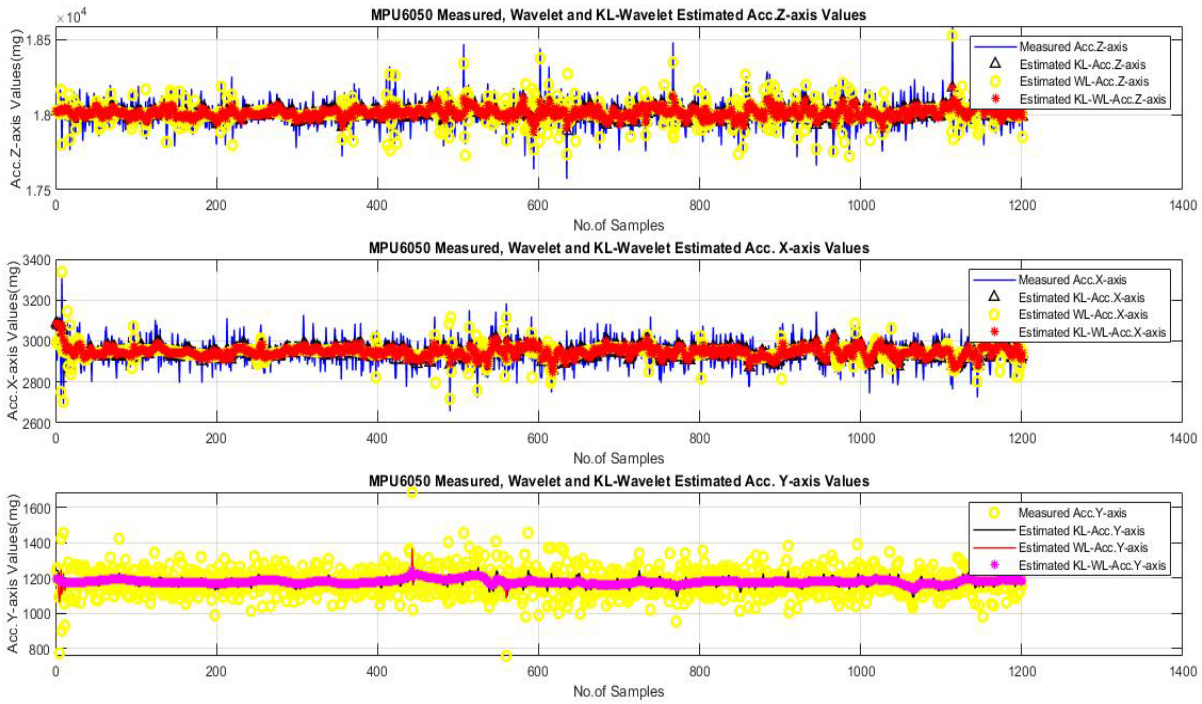


Figure (4.23): Results of test scenario No (11)

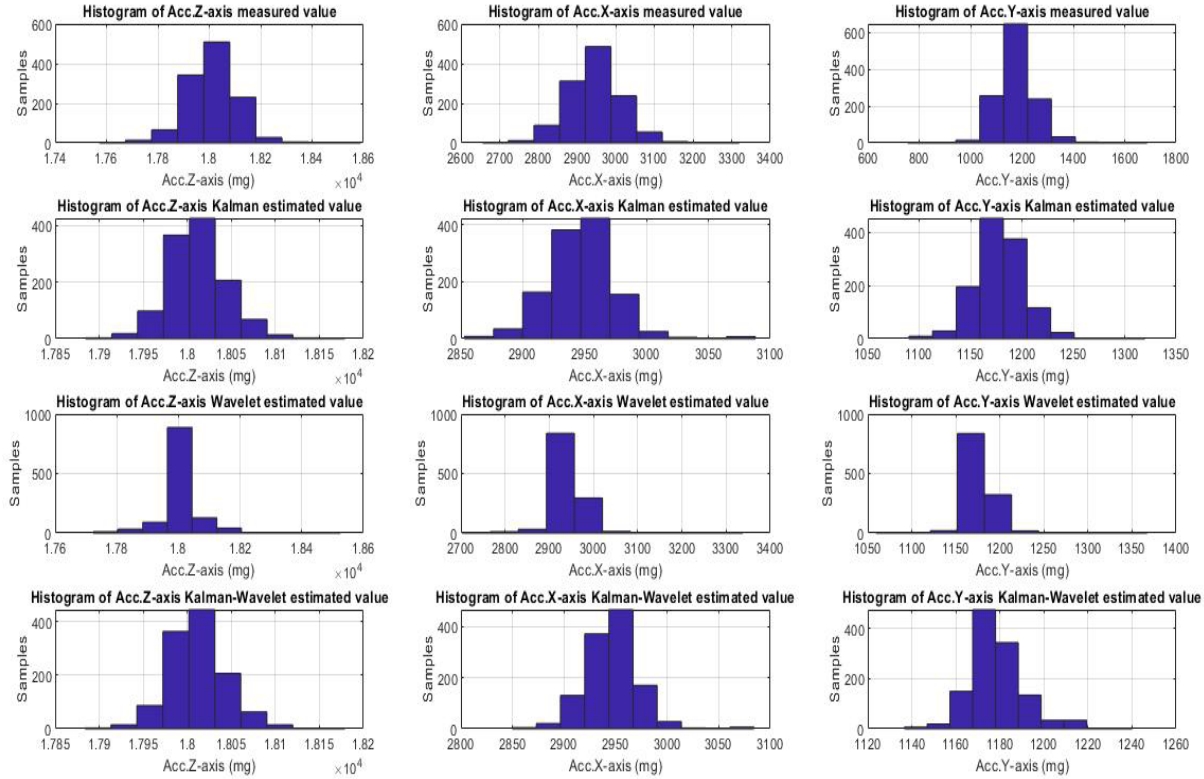


Figure (4.24): Histogram of Test scenario No (11)

4.3.12 Test Scenario No (12) :

```
    %---------- Z-Axis  ----------------------------
Z-Axis _WAVELET = wden(Z-Axis_MEASURE,' minimaxi ','h','mln',5,'db4');
Z-Axis _KLWLESTM = wden(Z-Axis_KALMAN,' minimaxi ','h','mln',5,'db4');
    %---------- X-Axis  ----------------------------
X-Axis _WAVELET= wden(X-Axis_MEASURE,' rigrsure ','h','mln',5,'db4');
 X-Axis _KLWLESTM = wden(X-Axis_KALMAN,' rigrsure ','h','mln',5,'db4');
    %---------- Y-Axis ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE, ' modwtsqtwolog ','h','mln',5,'db4');
 Y-Axis _KLWLESTM = wden(Y-Axis _KALMAN,' modwtsqtwolog ','h','mln',5,'db4');
```

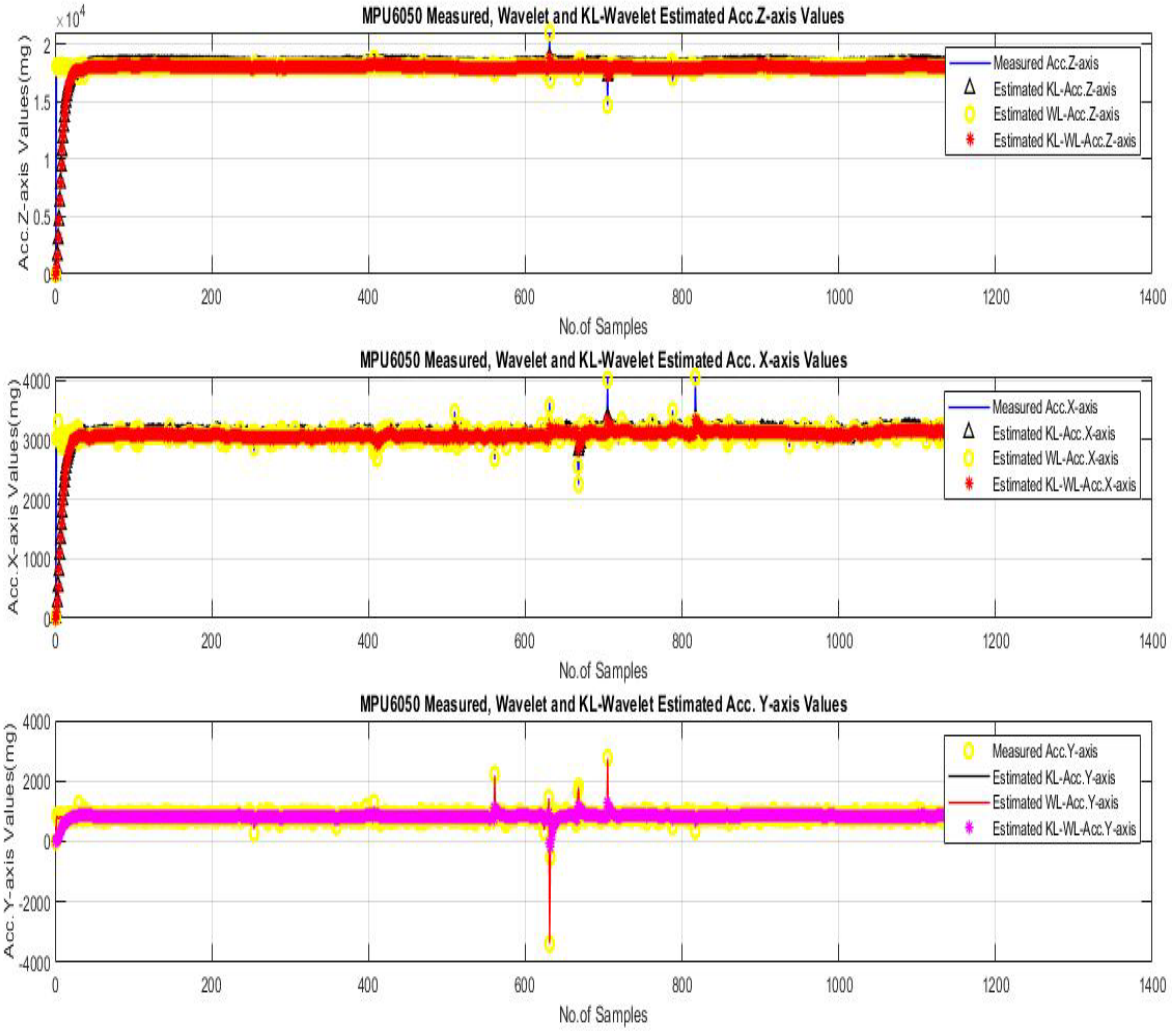


Figure (4.25): Results of test scenario No (12)

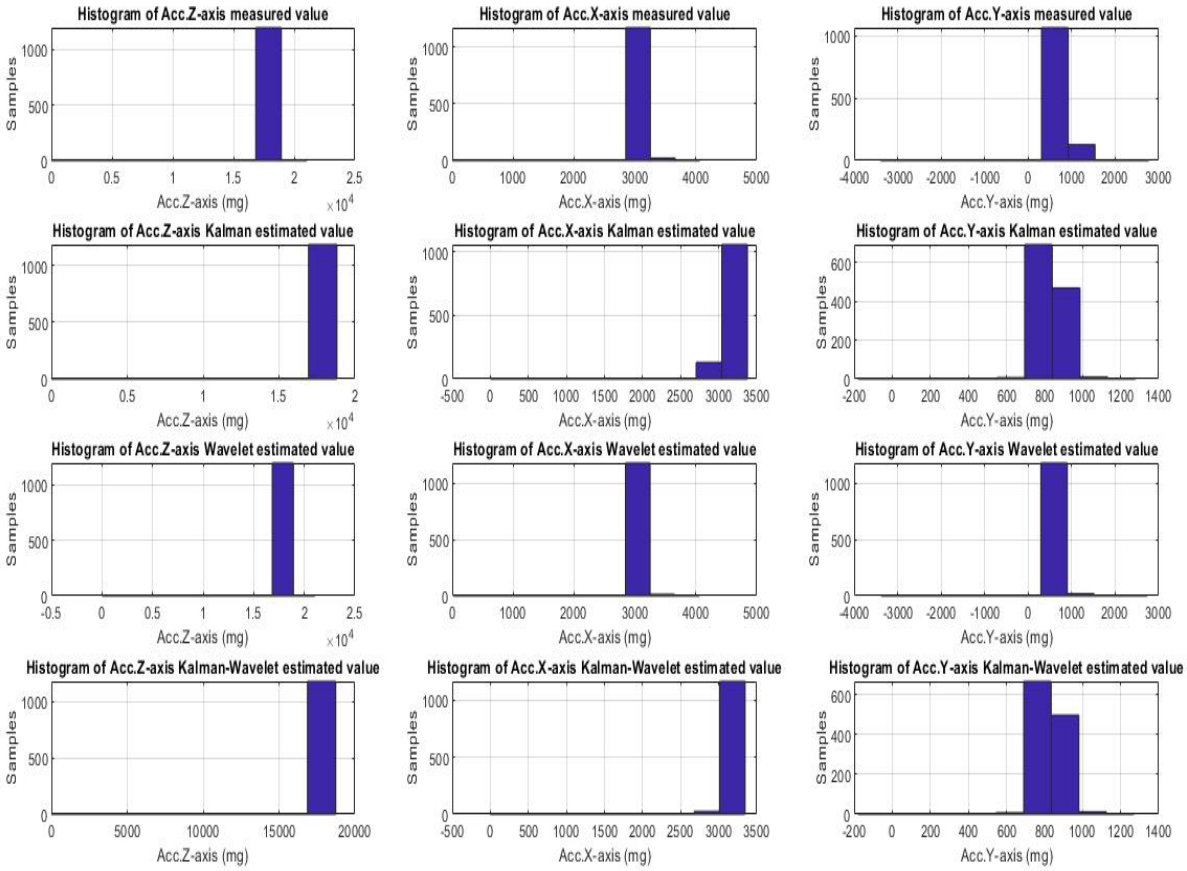


Figure (4.26): Histogram of Test scenario No (12)

Table (4.2): Measured and denoised Z-Axis Acc. Values (mg)

		Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario 7	Scenario 8
Average	Measured	1.8015e+04	1.8016e+04	1.8019e+04	1.8008e+04	1.8013e+04	1.8016e+04	1.8000e+04	1.7991e+04
	Kalman	1.8015e+04	1.8016e+04	1.8020e+04	1.8007e+04	1.8013e+04	1.8016e+04	1.8000e+04	1.7991e+04
	Wavelet	1.8015e+04	1.8016e+04	1.8019e+04	1.8008e+04	1.8013e+04	1.8016e+04	1.8000e+04	1.7991e+04
	KL-WL	1.8015e+04	1.8016e+04	1.8019e+04	1.8007e+04	1.8013e+04	1.8016e+04	1.8000e+04	1.7991e+04
Standard Deviation	Measured	90.0619	130.6928	131.4272	161.3535	110.9238	171.4919	161.1546	90.8594
	Kalman	30.5058	45.0615	44.1732	52.9173	39.0096	60.0122	57.8505	30.3540
	Wavelet	**24.9628**	35.6573	86.0612	159.8563	45.6025	117.9382	77.7398	37.4578
	KL-WL	**19.6329**	24.7399	36.8162	51.8213	35.3274	46.6679	39.0708	21.2804
		Scenario 9	Scenario 10	Scenario 11	Scenario 12				
Average	Measured	1.8006e+04	1.7999e+04	1.8011e+04	1.8005e+04				
	Kalman	1.8006e+04	1.7999e+04	1.8012e+04	1.7898e+04				
	Wavelet	1.8006e+04	1.7999e+04	1.8011e+04	1.8005e+04				
	KL-WL	1.8006e+04	1.7999e+04	1.8012e+04	1.7898e+04				
Standard Deviation	Measured	138.1918	92.2531	99.0601	550.2822				
	Kalman	44.0213	34.7395	32.9341	1.2182e+03				
	Wavelet	57.4366	92.2529	63.5892	543.5075				
	KL-WL	23.1871	34.7371	31.6331	1.2178e+03				

Table (4.3): Measured and denoised X-Axis Acc. Values (mg)

		Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario 7	Scenario 8
Average	Measured	2.9417e+03	2.9363e+03	2.9194e+03	2.9234e+03	2.9238e+03	2.9314e+03	2.9401e+03	3.1267e+03
	Kalman	2.9417e+03	2.9362e+03	2.9192e+03	2.9236e+03	2.9238e+03	2.9316e+03	2.9402e+03	3.1273e+03
	Wavelet	2.9417e+03	2.9363e+03	2.9194e+03	2.9234e+03	2.9237e+03	2.9315e+03	2.9401e+03	3.1267e+03
	KL-WL	2.9417e+03	2.9362e+03	2.9192e+03	2.9236e+03	2.9238e+03	2.9316e+03	2.9402e+03	3.1273e+03
Standard Deviation	Measured	65.7549	108.3089	72.7658	84.0197	62.2188	84.2590	74.4105	65.1058
	Kalman	21.9972	50.1009	26.3223	28.5536	22.8504	30.5496	27.4004	23.8680
	Wavelet	65.7546	90.9331	15.4391	35.1660	**13.2289**	28.5863	72.7215	13.5629
	KL-WL	21.9921	49.3000	14.7196	20.3973	16.4462	17.5532	26.2638	16.8093
		Scenario 9	**Scenario 10**	**Scenario 11**	**Scenario 12**				
Average	Measured	2.9314e+03	3.1287e+03	2.9461e+03	3.0889e+03				
	Kalman	2.9310e+03	3.1292e+03	2.9469e+03	3.0706e+03				
	Wavelet	2.9314e+03	3.1287e+03	2.9461e+03	3.0890e+03				
	KL-WL	2.9310e+03	3.1292e+03	2.9469e+03	3.0707e+03				
Standard Deviation	Measured	72.4491	62.2311	68.2490	132.0780				
	Kalman	25.3895	24.3718	26.9381	215.5714				
	Wavelet	15.0006	19.7543	34.1313	119.4301				
	KL-WL	**13.2268**	17.5099	25.6076	214.7704				

Table (4.4): Measured and denoised Y-Axis Acc. Values (mg)

		Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario 7	Scenario 8
Average	Measured	1.1892e+03	1.2021e+03	1.2286e+03	1.2373e+03	1.2325e+03	1.2126e+03	1.2070e+03	884.8127
	Kalman	1.1896e+03	1.2026e+03	1.2285e+03	1.2378e+03	1.2332e+03	1.2128e+03	1.2067e+03	884.1776
	Wavelet	1.1892e+03	1.2022e+03	1.2286e+03	1.2373e+03	1.2325e+03	1.2126e+03	1.2070e+03	884.7070
	KL-WL	1.1896e+03	1.2026e+03	1.2285e+03	1.2378e+03	1.2332e+03	1.2128e+03	1.2067e+03	884.1698
Standard Deviation	Measured	68.0322	98.8379	90.4858	94.3858	87.3764	96.5058	80.9599	61.5626
	Kalman	22.7028	37.1582	32.9917	35.0036	33.1327	32.3956	30.0627	23.8777
	Wavelet	68.0321	83.7285	66.9669	94.2358	61.1793	29.2096	80.7178	16.3103
	KL-WL	22.7028	36.4268	29.7955	34.9028	28.9394	17.1223	29.9677	21.9558
		Scenario 9	**Scenario 10**	**Scenario 11**	**Scenario 12**				
Average	Measured	1.2034e+03	874.4280	1.1779e+03	831.9634				
	Kalman	1.2041e+03	874.0832	1.1785e+03	827.6808				
	Wavelet	1.2036e+03	874.4280	1.1779e+03	831.9634				
	KL-WL	1.2043e+03	874.0832	1.1785e+03	827.6808				
Standard Deviation	Measured	75.3218	61.4240	72.8087	173.5715				
	Kalman	27.0917	21.6720	23.9566	88.5013				
	Wavelet	26.8785	61.4240	**14.5696**	155.6583				
	KL-WL	16.2235	21.6720	**12.0108**	86.1857				

4.4. Discussion

From the above Figures and Tables the following can be deduced:

1- During the test it is shown that no single denoising technique suitable for all acceleration components , therefore:
   - For ***Z-Axis*** measurement can be denoised with minimum SD of ***19.6329*** mg as shown in Table (4.2) in scenario ***No. 1*** when Kalman-Wavelet combination filtering technique was used. The second good result was achieved by using Wavelet filter with SD of ***24.9628*** mg.
   - For ***X-Axis*** the best result of SD achieved is ***13.2268*** mg also when using the Kalman-Wavelet combination as in scenario ***No 9*** at Table (4.3) However, denoising using Wavelet filter achieves much more closer result of ***13.2289*** mg SD value as in scenario ***No 5***.
   - For ***Y-Axis*** the best result of SD achieved is ***12.0108*** mg also when using the Kalman-Wavelet combination as in scenario ***No 11*** at Table (4.4) However, denoising using wavelet filter achieves much closer result of **14.5696** mg SD value as in scenario ***No 11***.

2- There is no much differences in performance of Wavelet and Kalman-Wavelet denoising techniques, where the differences in SD Values between Wavelet and Kalman-Wavelet combination are as follows:

   For Z-Axis: ***5.3299*** mg more for wavelet.

   For X-Axis: ***0.0021*** mg more for wavelet.

   For Y-Axis: ***2.5588*** mg more for wavelet.

3- For more investigation the outperformed scenario composed out of the good results achieved from previous test scenarios and in sake of optimization by studying the effect of decomposition level on the proposed denosing technique, different decomposition levels (3) and (7) were examined.

Therefore, the MATLAB syntax to achieve this study is:

```
%---------- Z-Axis ----------------------------
Z-Axis_WAVELET = wden(Z-Axis_MEASURE ,'modwtsqtwolog','h','mln',5/3/7,'db8');
Z-Axis_KLWLESTM = wden(Z-Axis_KALMAN,'modwtsqtwolog','h','mln', 5/3/7,'db8');
%---------- X-AXIS  ----------------------------
X-Axis _WAVELET= wden(X-Axis _MEASURE,'minimaxi','s','sln', 5/3/7,'db8');
X-Axis _KLWLESTM = wden(X-Axis _KALMAN,'minimaxi','s','sln', 5/3/7,'db8');
%---------- Y-Axis  ----------------------------
Y-Axis _WAVELET = wden(Y-Axis _MEASURE, 'modwtsqtwolog','s','mln', 5/3/7,'db4');
Y-Axis  _KLWLESTM = wden(Y-Axis_KALMAN, 'modwtsqtwolog','s','mln', 5/3/7,'db4');
```

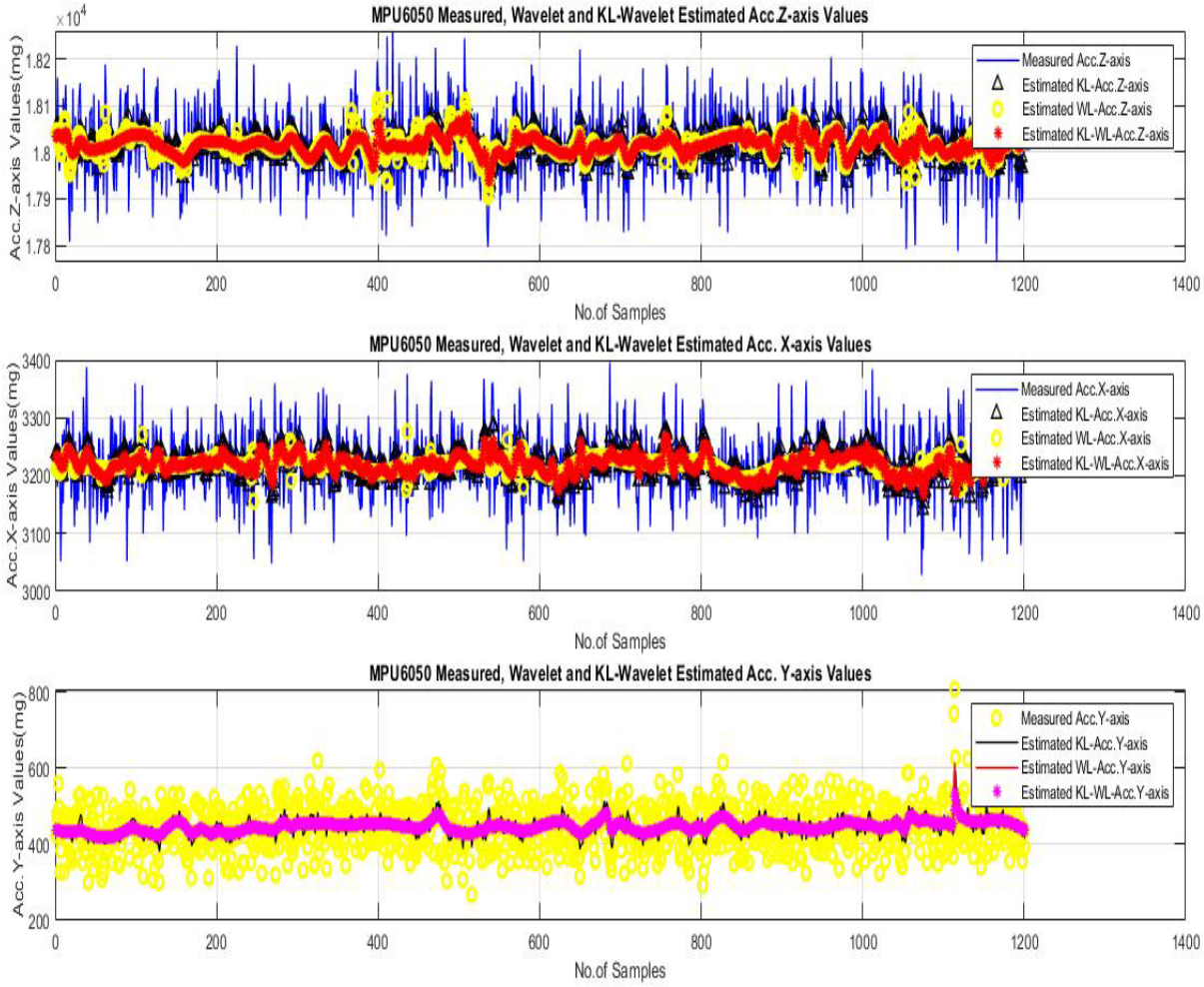


Figure (4.27): Results of outperformed test scenario with level (5)

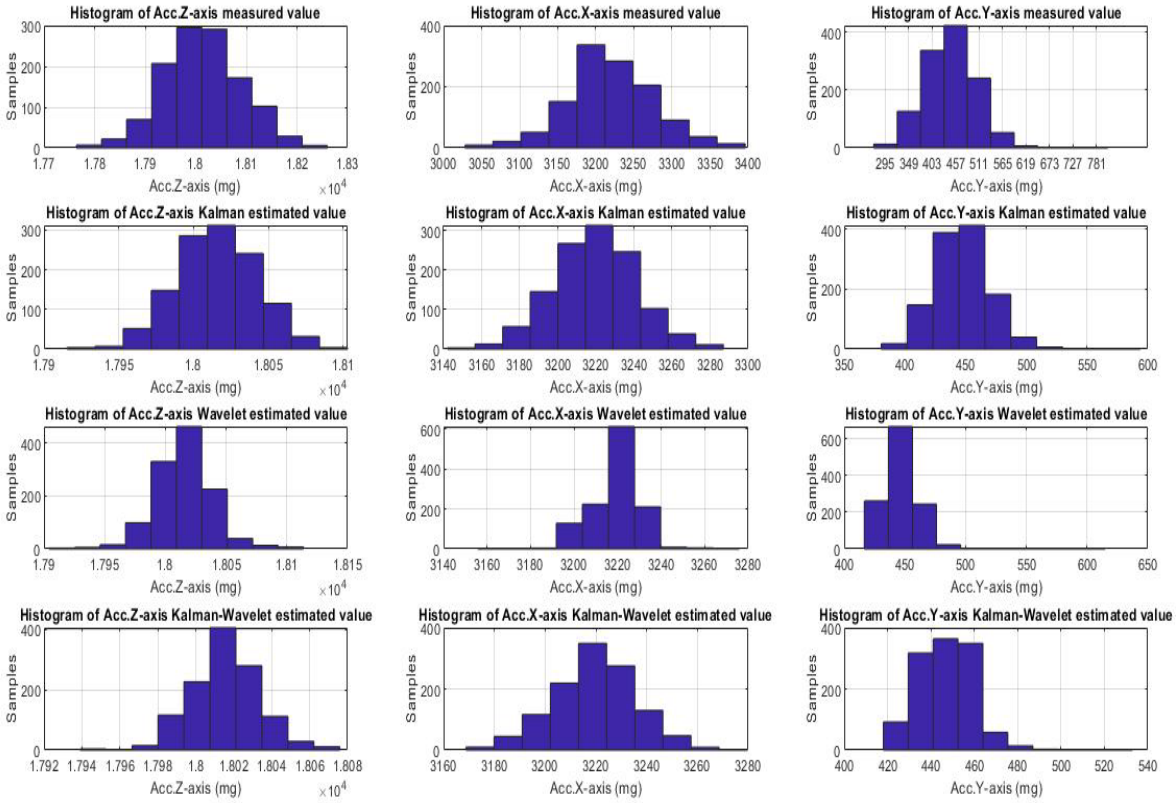


Figure (4.28): Histogram of the outperformed test scenario with level (5)

4.3.14 Best Scenario with level-3:

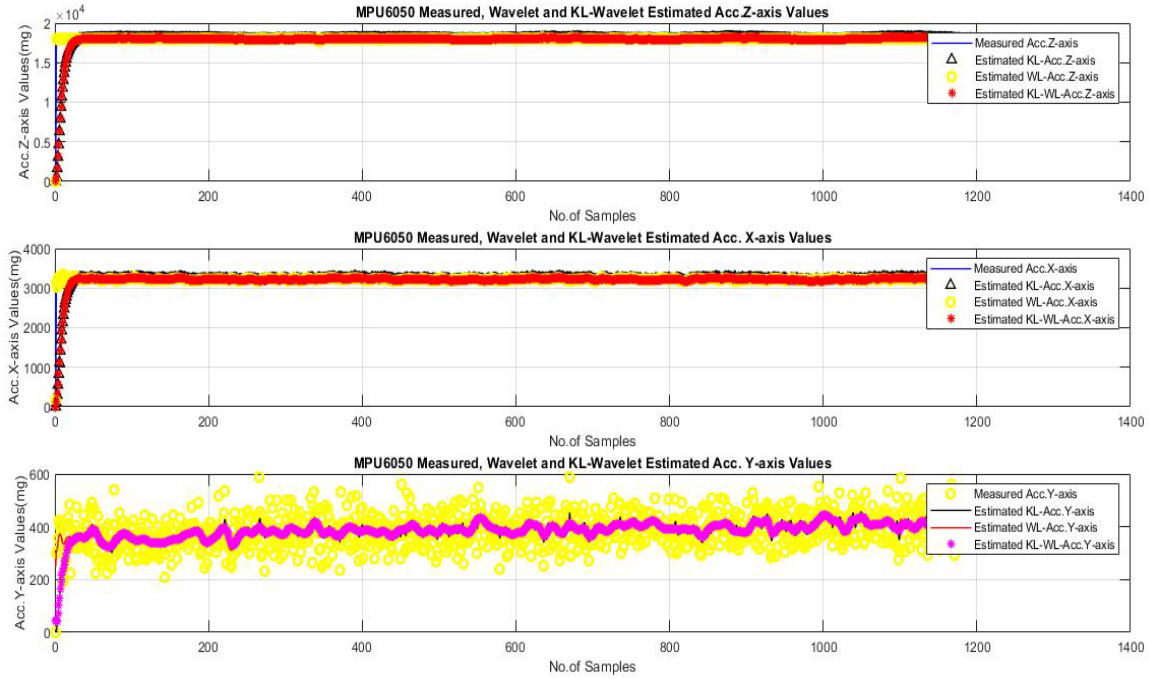


Figure (4.29): Results of outperformed test Scenario with level (3).

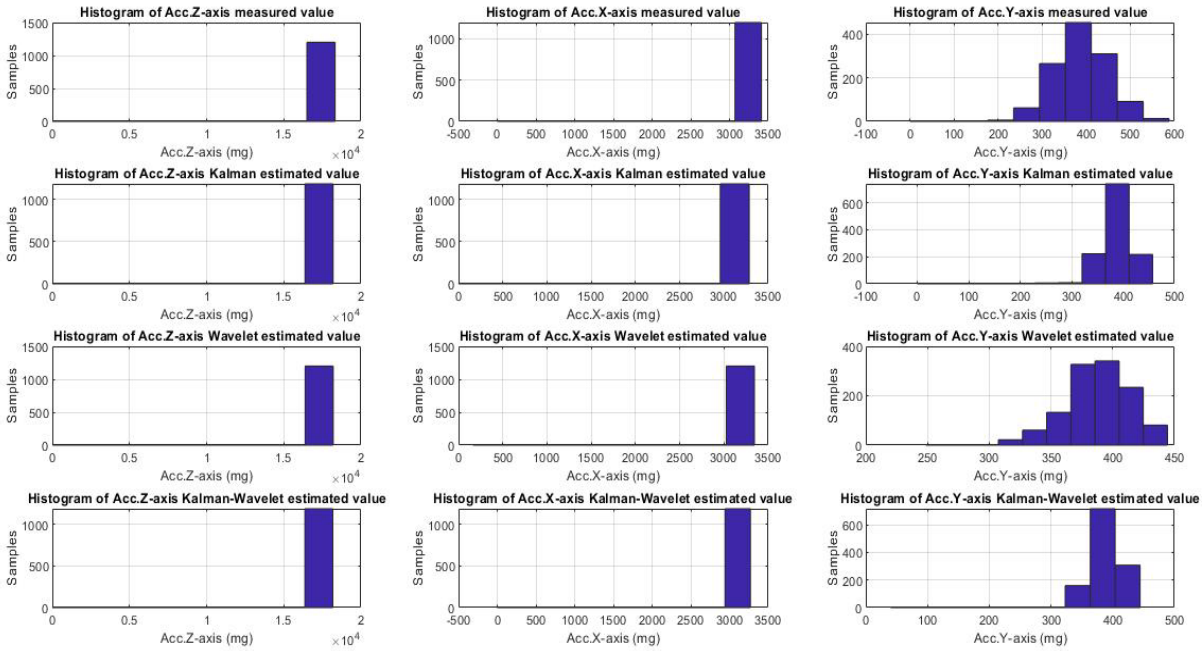


Figure (4.30): Histogram of the outperformed test scenario with level (3)

4.3.15 Best Scenario with level (7):

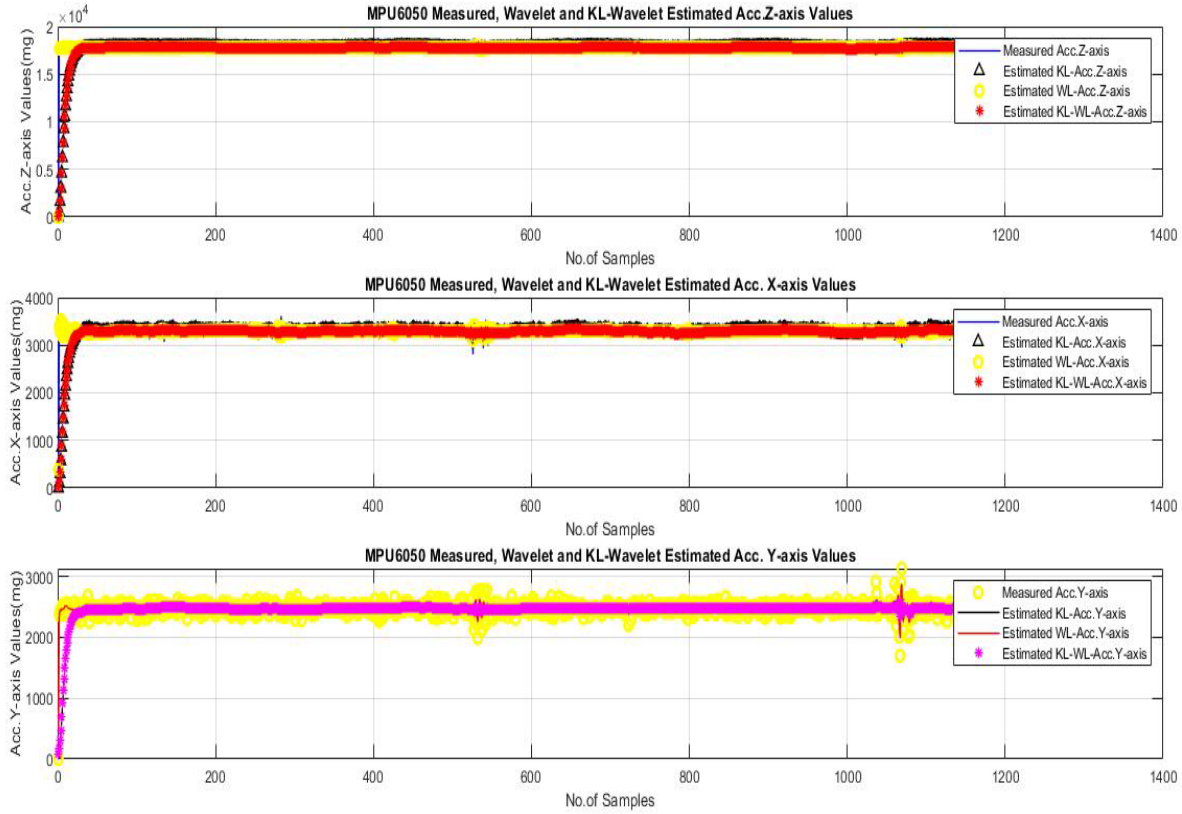


Figure (4.31): Results of the outperformed test Scenario with level (7).

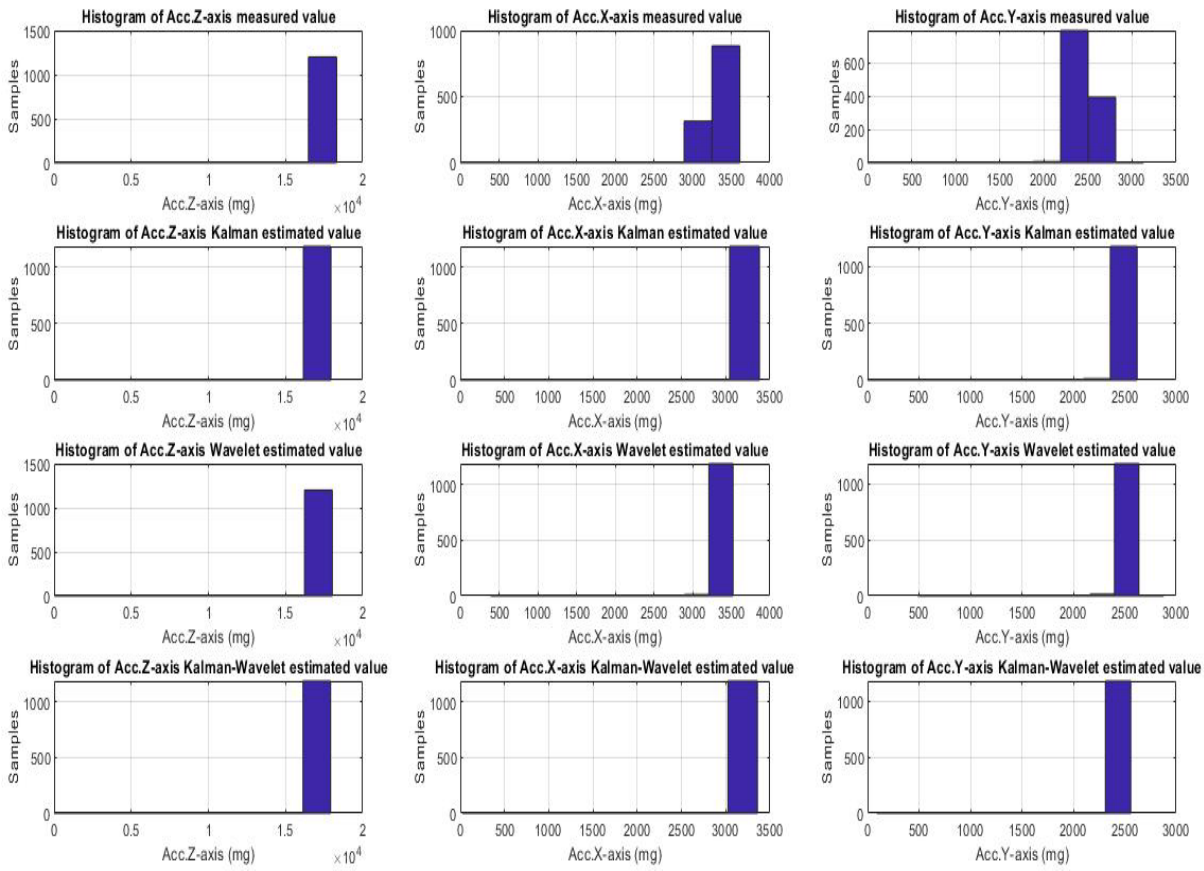


Figure (4.32): Histogram of the outperformed test scenario with level (7)

Table (4.5): Measured and denoised Z-Axis Acc values (mg) for different levels

		The Best with level 3	The Best with level 5	The Best with level 7
Average	Measured	1.7997e+04	1.8015e+04	1.8011e+04
	Kalman	1.7891e+04	1.8015e+04	1.8011e+04
	Wavelet	1.7997e+04	1.8015e+04	1.8011e+04
	KL-WL	1.7891e+04	1.8015e+04	1.8011e+04
Standard Deviation	Measured	526.6086	76.4318	82.9548
	Kalman	1.2187e+03	26.9104	27.8210
	Wavelet	520.6915	23.7922	34.2105
	KL-WL	1.2186e+03	**17.6218**	**19.3377**

Table (4.6): Measured and denoised X-Axis Values (mg) for different levels

		The Best with level 3	The Best with level 5	The Best with level 7
Average	Measured	3.2196e+03	3.2192e+03	3.2277e+03
	Kalman	3.2004e+03	3.2195e+03	3.2272e+03
	Wavelet	3.2195e+03	3.2191e+03	3.2272e+03
	KL-WL	3.2004e+03	3.2195e+03	3.2270e+03
Standard Deviation	Measured	109.2333	57.0178	59.0542
	Kalman	218.1690	21.8523	21.0616
	Wavelet	90.8034	11.2185	**6.1812**
	KL-WL	217.4975	16.0418	13.2828

Table (4.7): Measured and denoised Y-Axis Values (mg) for different levels

		The Best with level 3	The Best with level 5	The Best with level 7
Average	Measured	387.5171	447.1341	462.9076
	Kalman	385.3590	447.0573	463.5414
	Wavelet	387.5171	447.1341	462.9076
	KL-WL	385.3590	447.0573	463.5414
Standard Deviation	Measured	60.7062	58.9058	58.1949
	Kalman	37.6392	23.0989	21.8267
	Wavelet	25.8810	14.5565	**7.7808**
	KL-WL	34.7099	12.9925	8.0856

4- The result of the Decomposition level change to level (3) and (7) respectively shown on Tables (4.5, 4.6, and 4.7). From these Tables it is quite evident that a very good result achieved at level 7 for ***X-Axis*** and ***Y-Axis*** with only Wavelet filter with SD of ***6.1812, 7.7808*** mg respectively. For Z-Axis a good result achieved was in Kalman-Wavelet filter combination with level (5) by a difference in SD of ***1.7159*** mg. from wavelet filter alone.

5- From the figure (4.32) it is noticed that, when uncontaminated measured signal is acquisitioned all filters becomes neutral.

6- From the test scenarios it is seen that wavelet decomposition levels has considerable effect on the results.

7- The overall result achieved is that the optimum method for denoising MPU 6050 accelerometer is the wavelet denoising method for X-Axis and Y-Axis with level (7) but for Z-Axis the optimum is Kalman-wavelet combination level (5).

8- To avoid complexity, in the implementation for Z-Axis , wavelet filtering only with decomposition level (5) will be used with a negligible effect as follows:

For Z-Axis ----------'modwtsqtwolog','h','mln', **5**,'db8').
For X-Axis ----------('minimaxi','s','sln',**7**,'db8').
For Y-Axis ----------('modwtsqtwolog','s','mln',**7**,'db4').

9- When the proposed technique implemented in Arduino DUE as shown in Figure (4.33); unfortunately available Wavelet Arduino library did not supports decomposition level higher than level (2). In addition, only small size of signals samples i.e. 64 samples can be processed by Arduino DUE. Although of these limitations, the obtained results by the implemented Wavelet filter are accepted in terms of real time applications, costs and simplicity.

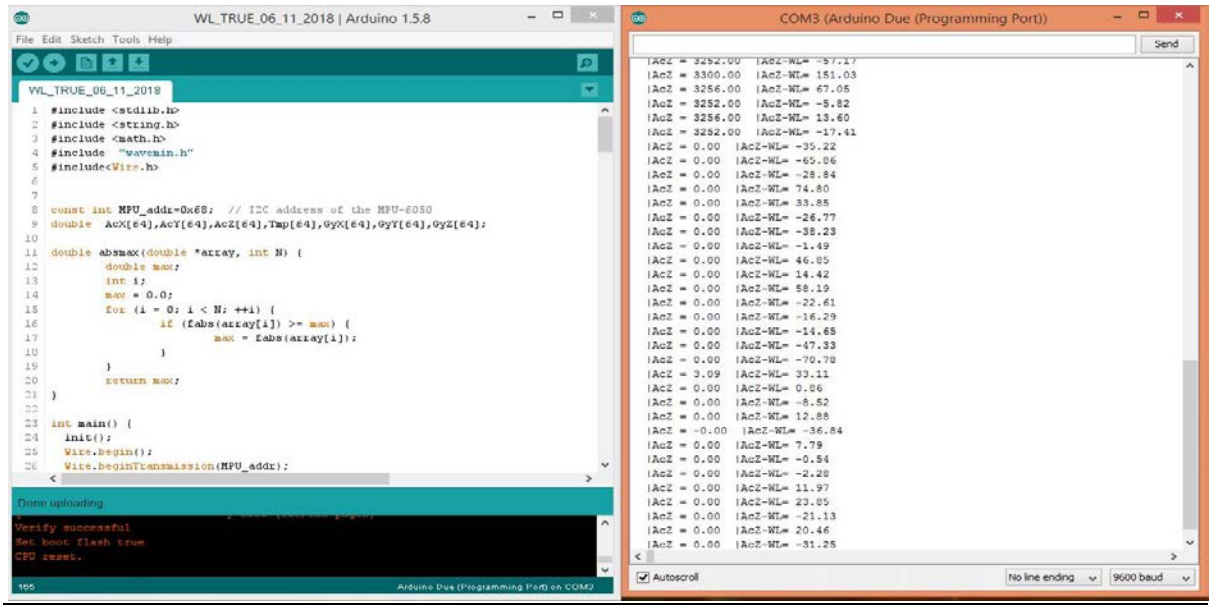


Figure (4.33): screenshot of Arduino DUE wavelet implementation and results

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The objective of this research was to improve the MEMS-based accelerometer system performance using Wavelet transformation and implement the developed algorithm in a single microcontroller Arduino DUE as a test bed. The performance of the simulated Wavelet denoising filter was examined for the acquisitioned MPU-6050 Accelerometer data. In addition, for comparison with Kalman based denoising filter was achieved. Therefore, the parameters of the Wavelet denoising filter were recognized and implemented in the proposed test bed; however the following conclusions can be drawn:

- No single denoising technique such as Kalman and Wavelet are suitable for all acceleration components. For X-Axis and Y-Axis, as example, it is suitable to use Wavelet filter where the Z-Axis can be denoised with a combination of Kalman-Wavelet.
- In spite of a single microcontroller limitation and available Arduino DUE library, it was recognized that a noticeable improvement achieved by Wavelet transformation.
- Only two decomposition levels can be implemented with Daubechies wavelets in Arduino DUE.
- Only 'No rescaling' option i.e. 'one' can be implemented in Arduino DUE where the other rescaling options such as "*sln"* and "*mln"* are not available in Arduino library.

Finally the developed wavelet transformation-based denoising algorithm implemented in a single microcontroller platform Arduino DUE with certain

limitation mentioned above , although the overall results concludes that Wavelet transformation outperformed Kalman and Kalman-Wavelet combination .

5.2 Recommendation and Future Work

It is recommended to use the Wavelet as an optimum denoising technique for MEMS based sensors, for practical application such as UAV, INS, IMU and all other MEMS applications. Moreover, when implementing a good consideration should be made to hardware and software limitations.

This research can be continued with the following future work suggestions:

- Re-inspect these methods with gyro signals.
- Make tests and comparisons against mechanical accelerometer for validation.
- Implement Wavelet denoising methods into a real MEMS based INS, UAV systems with the consideration for limitation mentioned above.
- Explore the implementation of Wavelet denoising in other platforms such as Raspberry Pi.
- Develop an Arduino suitable Wavelet library to include the most thresholding selection techniques and higher levels of decomposition.

REFERENCES

[1] Mohammad, I.,Younis, “MEMS Linear and Nonlinear Statics and Dynamic”, Department of Mechanical Engineering State University of New York. Binghamton, NY, USA, ISSN 1389-2134 .ISBN 978-1-4419-6019-1 Springer New York Dordrecht Heidelberg London, Springe Science, Business Media, LLC 2011.

[2] Prime Faraday Technology Watch “An Introduction to MEMS ” , Wolfson School of Mechanical and Manufacturing Engineering Loughborough University, Loughborough, Leics LE11 3TU ISBN 1-84402-020-7 , January 2002.

[3] Matej Andrejaši˘c ,MEMS Accelerometer Seminar , University of Ljubljana,Faculty for mathematics and physics, Department of physics,march 2008.

[4] Jiaying Du, “signal processing for MEMS sensor based motion analysis system”, Mälardalen University Press Licentiate Theses No. 228, 2016.ISBN 978-91-7485-256-1ISSN 1651-9256Printed by Arkitektkopia, Västerås, Sweden.

[5] Lindsay Kleeman, “Understanding and Applying Kalman Filtering”, Department of Electrical and Computer Systems Engineering, Monash University, Clayton.

[6] Greg Welch , Gary Bishop “An Introduction to the Kalman Filter”, University of North Carolina at Chapel Hill Department of Computer Science Chapel Hill, NC 27599-3175.

[7] Robi Polikar, “The Wavelet Tutorial”, Second Edition, Ames,IA,1994.

[8] Woo, C. K., Ho, C. K., and Gook, C. P., “Wavelet Denoising Technique for Improvement of the Low Cost MEMS-GPS Integrated System”

,International Symposium on GPS/GNSS, Taipei, Taiwan. October 26-28, 2010.

[9] Elkhidir T. Y., and Shuhimi M., Musa T. A., Satti. A., "Pre-Filtering Low-Cost Inertial Measuring Unit using a Wavelet Thresholding De-noising for IMU/GPS Integration", Sept. 27-29, 2011–Shah Alam, MALAYSIA .

[10] Li, Z.P., Fan, Q.J., Chang, L.M., Yang, X., "Improved wavelet threshold denoising method for MEMS Gyroscope" IEEE 11th International Conference on Control & Automation (ICCA), Taichung, Taiwan, June 18-20, 2014.

[11] Edu, I.R. Adochiei, F.C., Obreja, R. Rotaru, C. Grigorie, T. L., "New Tuning Method of the Wavelet Function for Inertial Sensor Signals Denoising". Advances in Applied and Pure Mathematics Journal, pp. 153-157. ISBN: 978-1-61804-240-8. Published on 07/ 07/ 2014.

[12] Adochiei, F., Obreja ,R. Ggrigorie, T. L.," Inertial Sensor Signals Denoising with Wavelet Transform", INCAS BULLETIN, volume 7, issue 1/ 2015, pp. 57 – 64 ISSN 2066 – 8201.

[13] AMMAR, HIBA, and SHIHAB,"Digital Signal Processing for MEMS Sensor Based Motion Analysis System", B.Sc. thesis, Aeronautical Engineering Department, Sudan University for science and technology (SUST), October 2017.

[14] InvenSense Inc., "MPU-6000 and MPU-6050 Register Map and Descriptions", Revision 4.0, Document Number: RM-MPU-6000A-00 Revision: 4.0 Release Date: 03/09/2012.

[15] InvenSense Inc., "MPU-6000 and MPU-6050 Product Specification", Revision 3.2, Document Number: PS-MPU-6000A-00, Release Date: 11/16/2011.

[16] Arduino, "Arduino Due data sheet" Arduino Due Board doc No 102312.

[17] Christoffer, Lundager, Nedergaard, “Gait Sensor- For Monitoring Movement before fall” AALBORG University 7th semester project January 2016.

[18] Jeff Rowberg ‘https://github.com/jrowberg/i2cdevlib/zipball/master.’ Sep 2017

[19] MATLAB 2017 Package Documentations.

APPENDICES

APPENDIX (A)
Denoising MPU-6050 using Kalman, Wavelet, and combination MATLAB programme

```
%#########################################################################

clear ALL
close all
clc

yrpNO=50; % number of iterations
dt=0.01;  % incriment
rate=0.0; %the gyro measurement over time gyro/sec
bias=0.0; % the amount the gyro has drift
p=zeros(2,2); % covariance matrix intalized by zeros
%-----------------------------------------
arduinoDUE=serial('COM3','BaudRate',9600);
fopen(arduinoDUE);
data=fscanf(arduinoDUE); % data reading ( string )

%--------------  Accellerometer, Temperature and Gyro indeces---
Ac_indx=strfind(data,'Ac'); % locating Ac at data string
Tmp_indx=strfind(data,'Tmp');
Gy_indx=strfind(data,'Gy');
%-----------------------------------------------------------------
AcX=(data(1,(Ac_indx(1)+6): (Ac_indx(2)-4))); % selecting
accelrometer readings from the data string
AcX= str2double(AcX); % convert string to number with 4bytes
(double)

AcY=(data(1,(Ac_indx(2)+6): (Ac_indx(3)-4)));
AcY= str2double(AcY);

AcZ=(data(1,(Ac_indx(3)+6): (Tmp_indx(1)-4)));
AcZ= str2double(AcZ);
yaw_MEASURE= AcZ;
roll_MEASURE= AcX;
pitch_MEASURE= AcY;
%----------------wavelet Initialization-------------------------
yaw_WAVELET=AcZ;
roll_WAVELET=AcX;
pitch_WAVELET=AcY;
%-------------Kalman Initialization-----------------------------
yaw_TRUErate= rate;
yaw_EST= yaw_MEASURE;
```

```
yaw_KALMAN=yaw_MEASURE;
yaw_bias=bias;
yaw_p=p;
%-----------------------------------------------
roll_TRUErate= rate;
roll_EST= roll_MEASURE;
roll_KALMAN=roll_MEASURE;
roll_bias=bias;
roll_p=p;
%-----------------------------------------------
pitch_TRUErate= rate;
pitch_EST= pitch_MEASURE;
pitch_KALMAN=pitch_MEASURE;
pitch_bias=bias;
pitch_p=p;
%-------------Combination Initialization-----------------------

yaw_KLWLESTM=  yaw_MEASURE;
roll_KLWLESTM=  roll_MEASURE;
pitch_KLWLESTM= pitch_MEASURE;
for i = drange(1:yrpNO) % yaw ,roll , pitch numbers of itration
    %---------- Z-axis -----------------------------
    data=fscanf(arduinoDUE);
%--------------  Accellerometer, Temperature and Gyro indices---

    Ac_indx=strfind(data,'Ac');
    Tmp_indx=strfind(data,'Tmp');
    Gy_indx=strfind(data,'Gy');
    %--------------------------------------
    AcX=(data(1,(Ac_indx(1)+6): (Ac_indx(2)-4)));
    AcX= str2double(AcX); % convert string to number with 4bytes
(double)

    AcY=(data(1,(Ac_indx(2)+6): (Ac_indx(3)-4)));
    AcY= str2double(AcY);

    AcZ=(data(1,(Ac_indx(3)+6): (Tmp_indx(1)-4)));
    AcZ= str2double(AcZ);
 %----------------Kalman denoising ----------------------------
            ------------- Z-axis -------------------
    yaw_MEASURE =[yaw_MEASURE AcZ];
    [yaw_TRUErate, yaw_EST, yaw_bias
,yaw_p]=Kalman_Acc(dt,yaw_TRUErate,yaw_bias,yaw_p,AcZ, yaw_EST);
    yaw_KALMAN=[yaw_KALMAN yaw_EST];
    %---------- X-axis -----------------------------
    roll_MEASURE=[roll_MEASURE AcX];
```

```
[roll_TRUErate roll_EST roll_bias
roll_p]=Kalman_Acc(dt,roll_TRUErate,roll_bias,roll_p,AcX,
roll_EST);
    roll_KALMAN=[roll_KALMAN roll_EST];
    %---------- Y-axis -----------------------------
    pitch_MEASURE=[pitch_MEASURE AcY];
    [pitch_TRUErate pitch_EST pitch_bias
pitch_p]=Kalman_Acc(dt,pitch_TRUErate,pitch_bias,pitch_p, AcY,
pitch_EST);
    pitch_KALMAN=[pitch_KALMAN pitch_EST];
end

fclose(arduinoDUE);

%------------------------wavelet denoising -------------------
%lev = 5;
%---------- Z-axis -----------------------------
yaw_WAVELET =
wden(yaw_MEASURE,'modwtsqtwolog','h','mln',5,'db4');
yaw_KLWLESTM =
wden(yaw_KALMAN,'modwtsqtwolog','h','mln',5,'db4');
%---------- X-axis -----------------------------
roll_WAVELET = wden(roll_MEASURE,'heursure','h','mln',5,'db4');

roll_KLWLESTM = wden(roll_KALMAN,'heursure','h','mln',5,'db4');
%---------- Y-axis -----------------------------
pitch_WAVELET =
wden(pitch_MEASURE,'heursure','h','mln',5,'db4');
pitch_KLWLESTM =
wden(pitch_KALMAN,'heursure','h','one',5,'db4');
%================================================================
i =1:yrpNO+1;
figure (1)
subplot(3,1,1)
plot(i,yaw_MEASURE,'-b','linewidth',1);
hold on
plot(i,yaw_KALMAN,'^k','linewidth',1);
hold on
plot(i,yaw_WAVELET,'oy','linewidth',2);
hold on
plot(i,yaw_KLWLESTM,'*r','linewidth',1)
hold on
xlabel('No. of Samples');
ylabel('MPU6050 Acc. Z-axis Values');
title('MPU6050 Measured, Wavelet and KL-Wavelet Estimated Acc.
Z-axis Values');
legend('Measured Acc. Z-axis      ',...
```

```
       'Estimated KL-Acc. Z-axis   ',...
       'Estimated WL-Acc. Z-axis   ',...
       'Estimated KL-WL-Acc. Z-axis ')
grid on;

subplot(3,1,2)
plot(i,roll_MEASURE,'-b','linewidth',1);
hold on
plot(i,roll_KALMAN,'^k','linewidth',1);
hold on
plot(i,roll_WAVELET,'oy','linewidth',2);
hold on
plot(i,roll_KLWLESTM,'*r','linewidth',1);
hold on
xlabel('No. of Samples');
ylabel('MPU6050 Acc. X-axis Values');
title('MPU6050 Measured, Wavelet and KL-Wavelet Estimated Acc.
X-axis Values');
legend('Measured Acc. X-axis      ',...
'Estimated KL-Acc. X-axis   ',...
        'Estimated WL-Acc. X-axis   ',...
        'Estimated KL-WL-Acc. X-axis  ');
grid on;
subplot(3,1,3)
plot(i,pitch_MEASURE,'oy','linewidth',2);
hold on
plot(i,pitch_KALMAN,'-k','linewidth',1);
hold on
plot(i,pitch_WAVELET,'-r','linewidth',1);
hold on
plot(i,pitch_KLWLESTM,'*m','linewidth',1)
hold on
grid on;
xlabel('No. of Samples');
ylabel('MPU6050 Y-axis Values');
title('MPU6050 Measured, Wavelet and KL-Wavelet Estimated Acc.
Pitch Values');
legend('Measured Acc. Y-axis       ',...
       'Estimated KL-Acc. Y-axis   ',...
       'Estimated WL-Acc. Y-axis   ',...
       'Estimated KL-WL-Acc. Y-axis ');
figure (2)
subplot (2,2,1);
hist(yaw_MEASURE);
title (' Z-axis measured value');
subplot (2,2,2);
hist(yaw_KALMAN);
```

```
title (' Y-axis kalman estimate value');
subplot (2,2,3);
hist(yaw_WAVELET);
title (' Y-axis wavelet estimate value');
subplot (2,2,4);
hist(yaw_KLWLESTM);
title (' Y-axis kalman-wavelet combination estimate value');
figure (3)
subplot (2,2,1);
hist(roll_MEASURE);
title (' X-axis measured value');
subplot (2,2,2);
hist(roll_KALMAN);
title (' X-axis kalman estimate value');
subplot (2,2,3);
hist(roll_WAVELET);
title (' X-axis wavelet estimate value');
subplot (2,2,4);
hist(roll_KLWLESTM);
title (' X-axis kalman-wavelet combination estimate value');
figure (4)
subplot (2,2,1);
hist(pitch_MEASURE);
title (' Y-axis measured value');
subplot (2,2,2);
hist(pitch_KALMAN);
title (' Y-axis kalman estimate value');
subplot (2,2,3);
hist(pitch_WAVELET);
title (' Y-axis wavelet estimate value');
subplot (2,2,4);
hist(pitch_KLWLESTM);
title (' Y-axis kalman-wavelet combination estimate value');

%#################### ERROR Calculations ########################
                          # (1) Z-axis
% (1.0) Z-axis measured values:
yaw_MEASURED_value_STNDEV  = std (yaw_MEASURE)
yaw_MEASURED_value_average = mean(yaw_MEASURE)
YAW_MEASURED_value_offset = ['Z-axis offset value_MEASURED = ',
num2str(yaw_MEASURED_value_average),'+-',
num2str(yaw_MEASURED_value_STNDEV)];
disp (YAW_MEASURED_value_offset)
% (1.1) Z-axis kalman estimate error calculation and offset:
yaw_KALMAN_STNDEV  = std (yaw_KALMAN)
yaw_KALMAN_average = mean (yaw_KALMAN)
```

```
YAW_KL_offset = [' Z-axis offset value_KALMAN = ',
num2str(yaw_KALMAN_average),'+-', num2str(yaw_KALMAN_STNDEV)];
disp (YAW_KL_offset)
% (1.2) Z-axis wavelet estimate error calculation and offset:
yaw_WAVLETEST_STNDEV  = std (yaw_WAVELET)  % yaw standard
deviation for wavelet estimate
yaw_WAVLETEST_average = mean (yaw_WAVELET) % yaw mean value for
wavelet estimate
YAW_WL_offset         = [' Z-axis offset value_wavelet = ',
num2str(yaw_WAVLETEST_average),'+-',
num2str(yaw_WAVLETEST_STNDEV)];
disp (YAW_WL_offset)
% (1.3) Z-axis wavelet and kalman comination estimate error
calculation and offset:
yaw_KLWLESTM_STNDEV  = std (yaw_KLWLESTM)
yaw_KLWLESTM_average = mean (yaw_KLWLESTM)
YAW_KLWLCOM_offset   = [ ' Z-axis offset value_KLWLCOM = ',
num2str(yaw_KLWLESTM_average),'+-',
num2str(yaw_KLWLESTM_STNDEV)];
disp (YAW_KLWLCOM_offset)

%##########################(2)X-axis ############################
% (2.0) X-axis measured values:
roll_MEASURED_value_STNDEV  = std (roll_MEASURE)
roll_MEASURED_value_average = mean(roll_MEASURE)
roll_MEASURED_value_offset = [ ' X-axis offset value_MEASURED =
', num2str(roll_MEASURED_value_average),'+-',
num2str(roll_MEASURED_value_STNDEV)];
disp (roll_MEASURED_value_offset)

% (2.1) X-axis KALMAN estimate error calculation and offset:
roll_KALMAN_STNDEV  = std (roll_KALMAN)
roll_KALMAN_average = mean (roll_KALMAN)
ROLL_KL_offset      = [ ' X-axis offset value_kalman = ',
num2str(roll_KALMAN_average),'+-', num2str(roll_KALMAN_STNDEV)];
disp (ROLL_KL_offset)

% (2.2) X-axis wavelet estimate error calculation and offset:
roll_WAVLETEST_STNDEV  = std (roll_WAVELET)  % X-axis standard
deviation for wavelet estimate
roll_WAVLETEST_average = mean (roll_WAVELET) % X-axis mean value
for wavelet estimate
ROLL_WL_offset = [ ' X-axis offset value_wavelet = ',
num2str(roll_WAVLETEST_average),'+-',
num2str(roll_WAVLETEST_STNDEV)];
disp (ROLL_WL_offset)
```

```
% (2.3) X-axis wavelet and kalman comination estimate error
calculation and offset:
roll_KLWLESTM_STNDEV  = std (roll_KLWLESTM)
roll_KLWLESTM_average = mean (roll_KLWLESTM)
ROLL_KLWLCOM_offset   = [ ' X-axis offset value_KLWLCOM = ',
num2str(roll_KLWLESTM_average),'+-',
num2str(roll_KLWLESTM_STNDEV)];
disp (ROLL_KLWLCOM_offset)

%########################## (3) Y-axis ##########################
% (3.0) Y-axis measured values
pitch_MEASURED_value_STNDEV  = std (pitch_MEASURE)
pitch_MEASURED_value_average = mean(pitch_MEASURE)
pitch_MEASURED_value_offset = [ ' Y-axis offset value_MEASURED =
', num2str(pitch_MEASURED_value_average),'+-',
num2str(pitch_MEASURED_value_STNDEV)];
disp (pitch_MEASURED_value_offset)
% (3.1) Y-axis KALMAN estimate error calculation and offset:
pitch_KALMAN_STNDEV  = std (pitch_KALMAN)
pitch_KALMAN_average = mean (pitch_KALMAN)
PITCH_KL_offset = [ ' Y-axis offset value_kalman = ',
num2str(pitch_KALMAN_average),'+-',
num2str(pitch_KALMAN_STNDEV)];
disp (PITCH_KL_offset)
% (3.2) Y-axis wavelet estimate error calculation and offset :
pitch_WAVLETEST_STNDEV  = std (pitch_WAVELET)  % Y-axis standard
deviation for wavelet estimate
pitch_WAVLETEST_average = mean (pitch_WAVELET) % Y-axis mean
value for wavelet estimate
PITCH_WL_offset = [ ' Y-axis offset value_wavelet = ',
num2str(pitch_WAVLETEST_average),'+-',
num2str(pitch_WAVLETEST_STNDEV)];
disp (PITCH_WL_offset)
% (3.3) Y-axis wavelet and kalman comination estimate error
calculation and offset :
pitch_KLWLESTM_STNDEV  = std (pitch_KLWLESTM)
pitch_KLWLESTM_average = mean (pitch_KLWLESTM)
PITCH_KLWLCOM_offset = [ ' Y-axis offset value_KLWLCOM = ',
num2str(pitch_KLWLESTM_average),'+-',
num2str(pitch_KLWLESTM_STNDEV)];
disp (PITCH_KLWLCOM_offset)

%#############################################################
```

//End Of The Program

APPENDIX (B)

Arduino DUE Implementation Wavelet Program

```
#include <stdlib.h>

#include <string.h>

#include <math.h>

#include  "wavemin.h"

#include<Wire.h>

const int MPU_addr=0x68;  // I2C address of the MPU-6050

double  AcX[64],AcY[64],AcZ[64],Tmp[64],GyX[64],GyY[64],GyZ[64];

double absmax(double *array, int N) {
        double max;
        int i;
        max = 0.0;
        for (i = 0; i < N; ++i) {
                if (fabs(array[i]) >= max) {
                        max = fabs(array[i]);
                }
        }
        return max;
}

int main() {
  init();
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
```

```
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  Serial.begin(9600);
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
  //l++++++++++++++++++++++++++++++   (Z axis)  ++++++++++++++++++++++++++++++++++++++++++++++
{
wave_object obj;
wt_object wt;
double *inp, *out, *diff;
int N, i, J;
char *name = "db4";
            obj = wave_init(name);// Initialize the wavelet
            N = 64;  //Length of Signal
            inp = (double*)malloc(sizeof(double)* N); //Input signal
            out = (double*)malloc(sizeof(double)* N);
            diff = (double*)malloc(sizeof(double)* N);
            //wmean = mean(temp, N);
for (i = 0; i < N; ++i) {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
```

```
 AcZ[i]=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
 }
     J = 2; //Decomposition Levels
          wt = wt_init(obj, "dwt", N, J);// Initialize the wavelet transform object
          setDWTExtension(wt, "sym");// Options are "per" and "sym". Symmetric is the default option
          setWTConv(wt, "direct");       dwt(wt, AcZ);// Perform DWT for AcZ
           for (i = 0; i < wt->outlength; ++i)
{
           Serial.print("  |AcZ = ");
           Serial.print(AcZ[i]);
           Serial.print("  |AcZ-WL= ");
                    Serial.println(wt->output[i]);
          }
}
//+++++++++++++++++++++++++++++++++  (X-axis)++++++++++++++++++++++++++++++++++++++++++
{
wave_object obj;
wt_object wt;
double *inp, *out, *diff;
int N, i, J;
char *name = "db4";
          obj = wave_init(name);// Initialize the wavelet
          N = 64;  //Length of Signal
          inp = (double*)malloc(sizeof(double)* N); //Input signal
     out = (double*)malloc(sizeof(double)* N);
```

```
        diff = (double*)malloc(sizeof(double)* N);
        for (i = 0; i < N; ++i) {

 Wire.beginTransmission(MPU_addr);
 Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
 Wire.endTransmission(false);
 Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
 AcX[i]=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
 }
        J = 2; //Decomposition Levels
        wt = wt_init(obj, "dwt", N, J);// Initialize the wavelet transform object
        setDWTExtension(wt, "sym");// Options are "per" and "sym". Symmetric is the default option
        setWTConv(wt, "direct");
//*******************************************************************************
        dwt(wt, AcX);// Perform DWT for AcX
    //dwt(wt, AcY);// Perform DWT for AcY
    //dwt(wt, AcZ);// Perform DWT for AcZ
    for (i = 0; i < wt->outlength; ++i) {
         Serial.print("AcX= ");
         Serial.print(AcX[i]);
         Serial.print("  |AcX-WL= ");
                Serial.print(wt->output[i]);
        }
}
//+++++++++++++++++++++  (y _axis ) ++++++++++++++++++++++++++++++++++++++++++++++
```

```
{
wave_object obj;
wt_object wt;
double *inp, *out, *diff;
int N, i, J;
char *name = "db4";
        obj = wave_init(name);// Initialize the wavelet
        N = 64;  //Length of Signal
        inp = (double*)malloc(sizeof(double)* N); //Input signal
      out = (double*)malloc(sizeof(double)* N);
        diff = (double*)malloc(sizeof(double)* N);
        //wmean = mean(temp, N);
for (i = 0; i < N; ++i) {
 Wire.beginTransmission(MPU_addr);
 Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
 Wire.endTransmission(false);
 Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
  AcY[i]=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
 }
        J = 2; //Decomposition Levels
        wt = wt_init(obj, "dwt", N, J);// Initialize the wavelet transform object
        setDWTExtension(wt, "sym");// Options are "per" and "sym". Symmetric is the default option
        setWTConv(wt, "direct");
//*************************************************************************
        dwt(wt, AcX);// Perform DWT for AcX
```

```
      //dwt(wt, AcY);// Perform DWT for AcY
      //dwt(wt, AcZ);// Perform DWT for AcZ
      for (i = 0; i < wt->outlength; ++i) {
            // Serial.print("AcX= ");
            // Serial.print(AcX[i]);
            // Serial.print("  |AcX-WL= ");
                  Serial.print(wt->output[i]);
             Serial.print("  |AcY= ");
             Serial.print(AcY[i]);
             Serial.print("  |AcY-WL= ");

            }
}
//*****************************************************************************
            wave_object obj;
wt_object wt;
double *inp, *out, *diff;
int N, i, J;
            idwt(wt, out);// Perform IDWT (if needed)
            //Test Reconstruction;
            for (i = 0; i < wt->siglength; ++i) {
                  diff[i] = out[i] - inp[i];
            }
            printf("\n MAX %g \n", absmax(diff, wt->siglength)); // If Reconstruction succeeded then the
output should be a small value.
            wt_summary(wt);// Prints the full summary.
```

```
        wave_free(obj);
        wt_free(wt);
        free(inp);
        free(out);
        free(diff);
Serial.flush();
        return 0;


}
```

APPENDIX (C)

Summary of Wavelet Families and Associated Properties

	morl	mexh	meyr	haar	*db*N	symN	coifN	biorNr.Nd
"Crude"	•	•						
Infinitely regular	•	•	•					
Compactly supported orthogonal				•	•	•	•	
Compactly supported biothogonal								•
Symmetry	•	•	•	•				•
Asymmetry					•			
Near symmetry						•	•	
Arbitrary number of vanishing moments					•	•	•	•
Vanishing moments for ϕ							•	
Arbitrary regularity					•	•	•	•
Existence of ϕ			•	•	•	•	•	•
Orthogonal analysis			•	•	•	•	•	
Biorthogonal analysis			•	•	•	•	•	•
Exact reconstruction			•	•	•	•	•	•
FIR filters				•	•	•	•	•
Continuous transform	•	•	•	•	•	•	•	•
Discrete transform			•	•	•	•	•	•
Fast algorithm				•	•	•	•	•
Explicit expression	•	•		•				for splines