REPUBLIC OF TURKEY
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

POSITION AND SPEED CONTROL OF THE BALANCED SYSTEM ACTUATED BY BLDC MOTORS WITH GUI IN LABVIEW

HUZEYFE DOĞRUKAN
MASTER OF SCIENCE

ADANA 2019

REPUBLIC OF TURKEY
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

POSITION AND SPEED CONTROL OF THE BALANCED SYSTEM ACTUATED BY BLDC MOTORS WITH GUI IN LABVIEW

Huzeyfe DOĞRUKAN
MASTER OF SCIENCE

SUPERVISOR
Assoc. Prof. Dr. Lütfü SARIBULUT

ADANA 2019

POSITION AND SPEED CONTROL OF THE BALANCED SYSTEM ACTUATED BY BLDC MOTORS WITH GUI IN LABVIEW

submitted by **Huzeyfe DOĞRUKAN** in partial fulfillment of the requirements for the degree of **Master of Science in Department of Electrical and Electronic Engineering, Adana Alparslan Türkeş Science and Technology University** by,

Assoc. Prof. Dr. Gözde Baydemir
PEŞİNT
Graduate School Director

Assoc.Prof. Dr. Mahmud Yusuf
TANRIKULU
Department Chair

Assoc. Prof. Dr. Lütfü
SARIBULUT
Supervisor

Thesis Committee

Assoc. Prof. Dr. Lütfü SARIBULUT
Adana Alparslan Türkeş Science and Technology University

Assoc. Prof. Dr. Önder
TUTSOY
Adana Alparslan Türkeş Science
and Technology University

Dr. Ercan
AVŞAR
Çukurova University

Thesis Defense Date

13/06/2019

I hereby declare that all information in this thesis has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all information that is not original to this work.

Hüzeyfe DOĞRUKAN

ABSTRACT

POSITION AND SPEED CONTROL OF THE BALANCED SYSTEM ACTUATED BY BLDC MOTORS WITH GUI IN LABVIEW

Hüzeyfe DOĞRUKAN
Department of Electrical and Electronic Engineering
Supervisor: Assoc. Prof. Dr. Lütfü SARIBULUT
June 2019, 39 pages

Brushless DC motors (BLDC) are preferred in today's defense industry, industrial applications and electric cars due to their advantages such as low maintenance requirements, high speed revolutions, low electrical noise and high efficiency compared to brushed motors. In addition to these fields, BLDC motors are widely used in rotating propeller flight equipment and in autonomous flights, the balance regulation of the device is provided by the data obtained from acceleration and xyz-coordinate sensors.

In this thesis, the negative effects of external noise on xyz-coordinate data obtained from Gyro-sensor have been investigated and it has been tried to reduce these effects by using various digital filters. In this study, an equilibrium system using two brushless dc motors was designed and the data obtained from the acceleration sensors were transferred to the computer via Arduino controller. LABVIEW program tried to control the balance system by processing these data. In addition, the system was turned into an experimental set and the performance was observed according to the proportional-integral-derivative (PID) controller parameters.

Keywords: Brushless DC Motor, BLDC, PID, Arduino, LABVIEW

ÖZET

BLDC MOTORLARDAN OLUŞAN DENGE SİSTEMİNİN KONUM VE HIZ KONTROLÜNÜN LABVIEW PROGRAMINDA GUI İLE GERÇEKLEŞTİRİLMESİ

Hüzeyfe DOĞRUKAN
Elektrik Elektronik Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Lütfü SARIBULUT
Haziran 2019, 39 sayfa

Fırçasız Doğru Akım Motorlarının (FDAM) bakım gereksinimlerinin az olması, yüksek devir sağlamaları, elektriksel gürültülerinin düşük olması, fırçalı motorlara göre verimlerinin yüksek olması gibi avantajlarından dolayı günümüz savunma sanayisinde, endüstriyel uygulamalarda ve elektrikli arabalarda tercih edilmektedir. Bu alanların yanı sıra FDAM'ler, döner pervaneli uçuş donanımlarında çokça kullanılmakta olup, otonom uçuşlarda cihazın denge düzenlemesi ivme ve xyz-koordinat sensörlerinden alınan verilerle sağlanmaktadır.

Bu tez çalışmada dış etkenli gürültülerin, Gyro sensöründen alınan xyz-koordinat verilerinin üzerindeki olumsuz etkileri incelenmiş olup, çeşitli dijital filtreler kullanılarak bu etkileri azaltılmaya çalışılmıştır. Tez kapsamında, iki FDAM'in kullanıldığı bir denge sistemi tasarlanmış, Arduino denetleyicisi aracılığı ile ivme sensörlerinden alınan veriler bilgisayara aktarılmıştır. LABVIEW programı ile bu veriler işlenerek denge sistemi kontrol edilmeye çalışılmıştır. Ayrıca, sistem deney seti haline getirilerek Oransal-İntegral-Türevsel (PID) Denetleyici parametrelerine göre performansı gözlemlenmiştir.

Anahtar Kelimeler: Fırçasız DC motor, FDAM, PID, Arduino, LABVIEW

Dedication

I would like to dedicate my thesis to my family.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Assoc. Prof. Dr. Lütfü SARIBULUT for his support during my graduate studies.

I would like to thank my dear mother Mrs. Dürdane DOĞRUKAN, who I owe everything and Mr. Hasan DOĞRUKAN for doing his best to grow up without feeling the lack of a father, even though he has been abroad for years.

I would like to thank my esteemed sister Sule ÇALTA and Prof. Dr. Meting ÇALTA who hosted me in their homes during my university years and raised me as their children.

I would like to thank my workplace ENERJİSA for its financial and moral support during my graduate studies.

I would like to thank my gratitude to my brothers and sisters, who are proud and valuable members of my family.

The meaning of my life I would like to thank my wife YASEMİN DOĞRUKAN and my dear daughter Hanne DOĞRUKAN.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

ACCELY : Axis Y Acceleration
ACCEL Z : Axis Z Acceleration
ACCELX : Axis X Acceleration
BLDC : Brushless Direct Current
DAC : Digital to Analog
EMF : Electromagnetic Force
ESC : Electronic Speed Control
FPGA : Field Programmable Gate Array
GND : Ground
GPS : Global Positioning System
GYRO : Gyroscope
IMU : Inertia Measurement Unit
MATLAB : Matrix Laboratory
MOSFET : Metal-Oxide-Semiconductor Field-Effect Transistor
PI : Proportional Integral
PIC : Peripheral Interface Controller
PID : Proportional Integral Derivative
PWM : Pulse Wide Modulation
RMS : Root Mean Square

1. INTRODUCTION

1.1 Scope of Thesis

A balanced system actuating by two BLDC motors was installed to observe the efficiency of several digital filters used for determining the xyz-coordinates of balance-bar in this thesis. PI and PID controllers for BLDC motors were provided by using LABVIEW. Also, a user interface was design by using GUI of LABVIEW. It allows the user to change the controller parameters at any position of the scale. It is easy to see what kind of responses the motors give to the variable speed and position situations by using user interface.

An acceleration sensor (MPU6050), potentiometer and gyro-sensor were used for obtaining the position of the scales in the power circuit of balanced system. The acceleration sensor gives the position information as an input parameter to PID control. Both the potentiometer or gyro-sensor can be used to obtain the xyz-coordinates of the balance stick and they can be selected by using GUI for which type of sensor is used in the application. BLDC motors are controlled by using ESC without hall sensors. The outputs of PI, PID and sensors are also given graphically on the display of GUI.

2. LITERATURE REVIEW

In the literature research, the studies are generally made on the design performance and drivers of BLDC motors and there are few academic articles about the control of the system. A BLDC motor design for use in a satellite application for position control for low orbit satellites was examined and performance parameters such as torque per unit weight, torque per unit volume, moment of inertia contribution and power efficiency were discussed [1]. As a result, the overall performance of the engine was discussed and the different advantages were expressed. A driver circuit performing closed loop speed control of a 3 phase BLDC motor was made [2]. Motor commutation was realized with MC33035 integrated circuit and MC33039 integrated circuit, which was an electronic tachometer, measured the speed of BLDC motor and experimental results were obtained and comparisons were made.

Basic concepts related to BLDC motors were examined and the structure of the motor, sensor types and operating principles were given [3]. DSPIC30f module used for motor speed control was examined and 'P'I and 'PID' controllers were designed with the help of experimentally obtained open conversion results. The switching and position, the torque and speed control algorithms were compared [4]. It was tried to reveal the differences in the behavior of the position controller and the changes of the parameters. Finally, the results of the experimental setup were presented.

The control of BLDC motors with and without sensors was analyzed on MATLAB / Simulink platform and the simulation results were shown under reference speed and variable load [5]. BLDC motor speed and position control was actualized in the real time [6]. The general operating principles and parameters of the BLDC motor were controlled by running the combination of MATLAB/GUI and Simulink modules and by entering the values to the interface program. Position and current control systems for brushless motor were designed and operated in a series of FGPA [7]. Experimental results were obtained by controlling brushless direct current motors in the wing drive system of a guided missile. Both sensored and sensorless controls of BLDC motor were studied. Contrast EMF method was used as a sensorless control method. ATMEL microprocessor was used for a real-time control. The results of simulation and real time experiments were given.

3. MATERIALS AND METHODS

3.1 Brushless Direct Current Motors

BLDC rapidly increase the popularity in parallel with the technological developments in the industrial and daily fields. While the first popular applications were hard disks, nowadays, in the small and medium sized unmanned aerial vehicles, medical industry, the electric automotive industry are increasing day by day. The most important advantages of BLDC motors are given below.

- Better speed versus torque characteristics,
- High dynamic response,
- High efficiency,
- Long operating life,
- Noiseless operation,
- Higher speed ranges,
- Low maintenance [1]

3.2 Construction

BLDC motors are typical of synchronous motor. So the magnetic field is generated by the stator and the rotor rotates at the same frequency [2].

3.3 Stator Structure

Most of the BLDC motors stator windings are connected in star form. Types of stator windings are trapezoidal and sinusoidal motors. These different windings mean different Electromotor Force (EMF). As seen in figure 3.1 and figure 3.2, EMF waveforms are different from each other.

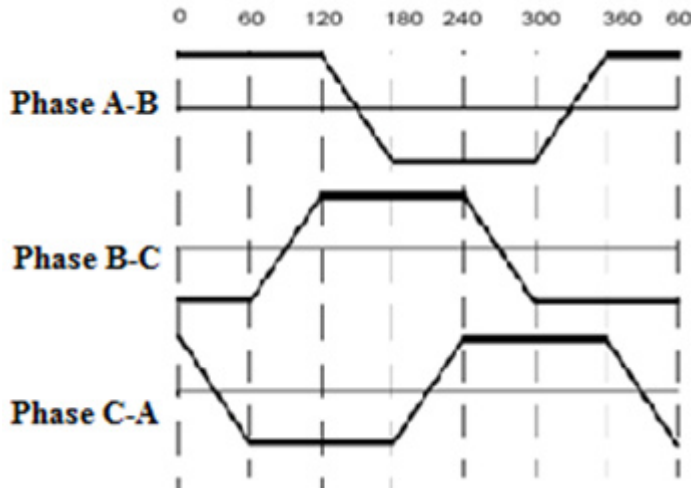

Figure 3.1 Trapezoidal Back EMF [2]

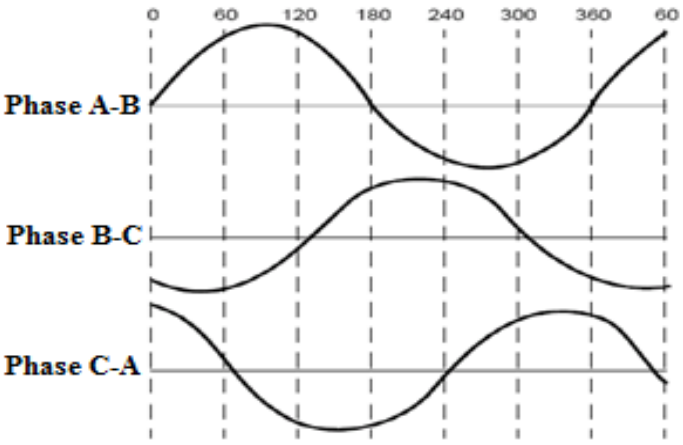

Figure 3.2 Sinusoidal Back EMF [2]

3.4 Rotor Structure

The number of poles can vary between 2 and 8 in motor made from permanent magnet. Figure 3.3 shows cross sections of different arrangements of magnets in a rotor.

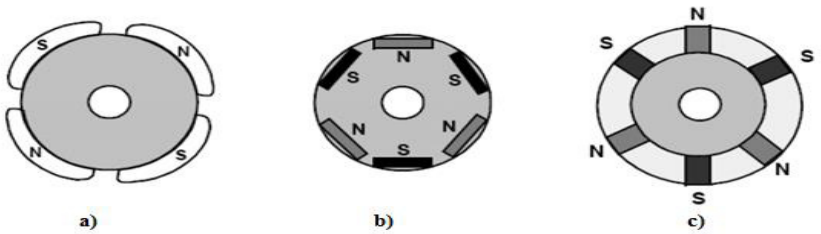

Figure 3.3 Rotor Magnet Cross Section a) Magnet on the periphery, b) rectangular magnets embedded in the rotor, c) rectangular magnets inserted into the rotor core

3.5 BLDC Equivalent Circuit

The equivalent circuit of the BLDC motor and the drive circuit are shown in Figure 3.4. Resistance, inductance and reverse EMF are the basic elements of the equivalent circuit. The drive circuit is the commonly used H-bridge method in DC motors. The basic element of the bridge is the switching elements. Depending on the motor power and the switching speed, this element can be a transistor or MOSFET.

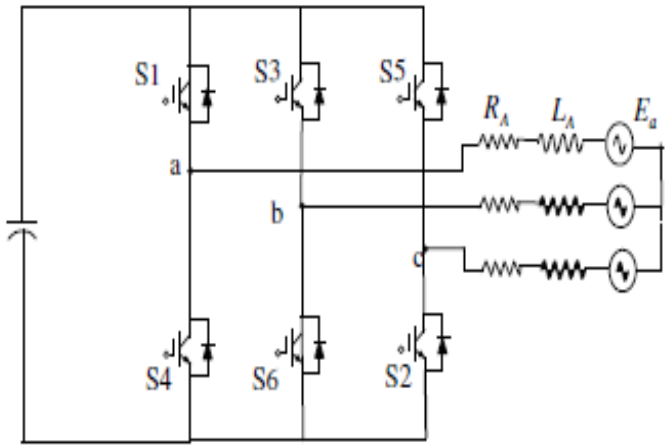

Figure 3.4 H-Bridge and BLDC equivalent Circuit [2]

3.6 H-Bridge Motor Driver and BLDC Commutation

The motor accelerates or decelerates according to the ON or OFF speed of the active switching elements. When the PWM method is used for low torque high speeds, the High-Low method can be used in applications requiring high torque. The sequence to be monitored for driving is given in the Table 3.1 switching elements are divided into two groups as High and Low side. For example, when switches ‘S1’ and ‘S2’ are ON, other switches are in OFF, the current flows through the coil ‘a’ and the motor advances one step. Figure 3.5 shows the commutation sequence of a three-phase BLDC motor driver circuit for counterclockwise rotation.

Three hall sensors ‘a’, ‘b’, and ‘c’ are mounted on the stator at 120° intervals, while the three phase windings are in a star form. For every 60° rotation, one of the hall sensors changes its state; it takes six steps to complete a whole electrical cycle. In synchronous mode, the phase current switching updates every 60°. For each step, there is one motor terminal driven high, another motor terminal driven low, with the third one left floating. Individual drive controls

for the high and low drivers permit high drive, low drive, and floating drive at each motor terminal. [8]

Table 3.1. Position of the motor according to the ON / OFF status of the switching elements used in brushless DC motors

One Cycle State	Output					
	Low Side Switch			High Side Switch		
	S4	S6	S2	S1	S3	S5
A	OFF	OFF	ON	ON	OFF	OFF
B	OFF	OFF	ON	OFF	ON	OFF
C	ON	OFF	OFF	OFF	ON	OFF
D	ON	OFF	OFF	OFF	OFF	ON
E	OFF	ON	OFF	OFF	OFF	ON
F	OFF	ON	OFF	ON	OFF	OFF

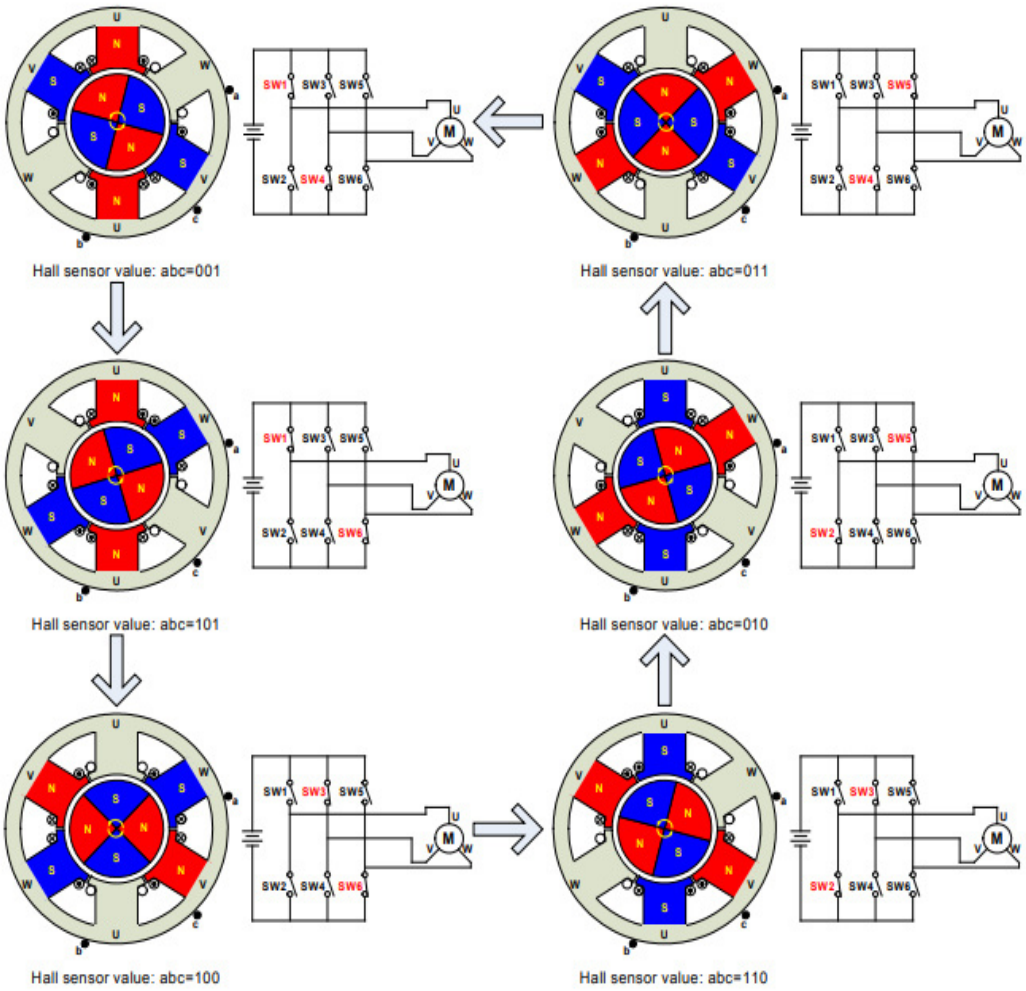

Figure 3.5 Three-Phase BLDC Motor Commutation Sequence [8]

3.7 Electronic Speed Control

ESC is a device that controls the electric motor. There are special ESCs for brushless dc motors. It is built of switching elements and accommodates the H-bridge with varying power. Thanks to an external potentiometer and PWM circuit, a brushless dc motor can be easily controlled.

3.8 Control Circuits

Brushless DC motors are three-phase and require two switching elements to control each output. Figure 3.6 shows one phase control circuit. The brushless dc motor can be controlled when three of these circuits are made. Figure 3.7 shows three phase control circuit. The A- and A + terminals are connected to the PWM outputs of the Arduino. These inserts serve as control ends. The U1 and U2 elements are photo diode that provides electrical isolation between the H-bridge and the Arduino. Metal Oxide Semiconductor Field Effect Transistor (MOSFET) was used as the switching element. Depending on the power of the brushless DC motor, the switching element is selected and can be used in the transistor for a low-power motor.

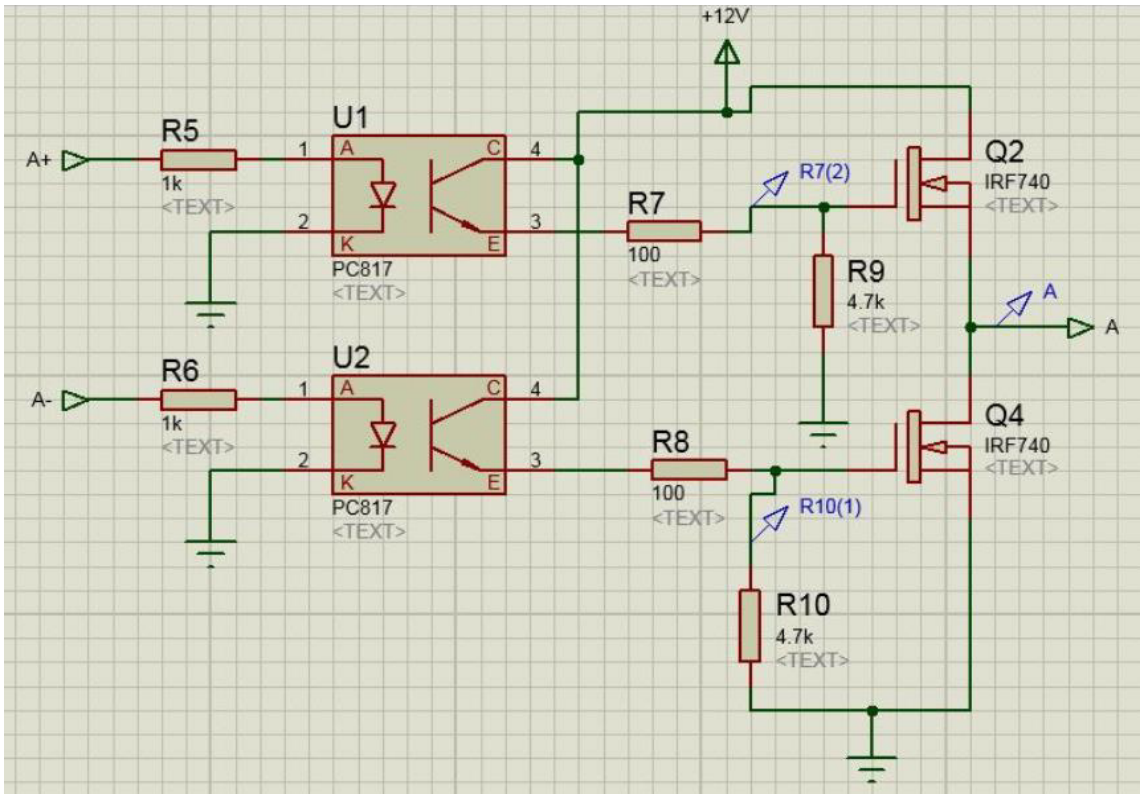

Figure 3.6 BLDC One-Phase Control Circuit

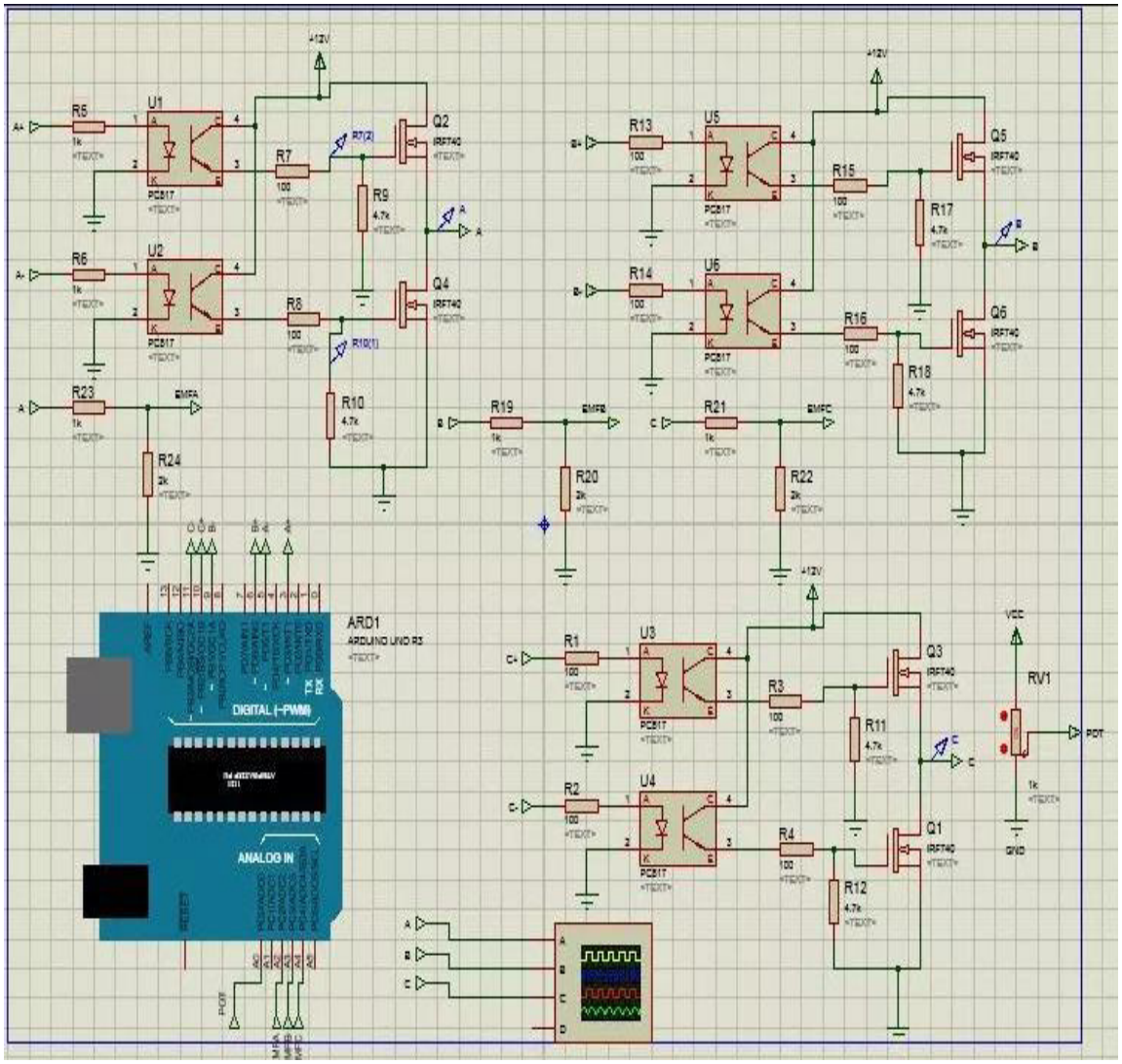

Figure 3.7 BLDC Three-Phase Control Circuit

Arduino programming enables switching according to the order in Table 3.1 ,1 or 0 BLDC motor control can be made if Arduino digital outputs are available. In motor drive circuits, when the switching elements such as relay, transistor or MOSFET are energized, reverse EMF is generated on the circuit. A diode or photo diode is placed in the parallel of these switching elements in order to avoid this effect.

3.9 Speed Control Method

ESC circuit always expects a reference value from the user. This increases or decreases the duty cycle value of the PWM signal. It will generate a reference signal regardless of how many bits of analog input such as Arduino or PIC will be used. In automatic controls and closed loop designs, sensors are usually position or speed sensors.

Figure 3.8 shows, a reference signal between the 5V (VCC) and 0V (GND) connected to the circuit can be adjusted from 0-5V at the intermediate end. A potentiometer is connected to the analog input and provides reference duty cycle to the PWM outputs controlling the H-bridge circuit.

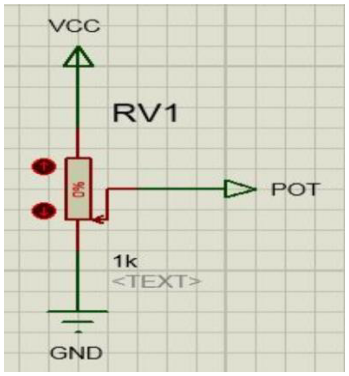

Figure 3.8 Speed Control Method

BLDC motors have speed control circuits for industrial or hobby purposes. Circuit in figure 3.9 is produced in very small sizes with different current values. Electronic speed control (ESC), which is produced by Simon, which has a capacity of 12 V, 20A, is controlled by PWM. It provides + 5V output to control the ESC circuit without a microprocessor. This allows you to set up a PWM generator circuit and control the BLDC motor speed. The frequency of the ESC is 490 Hz with duty cycle the speed of the motors can be adjusted.

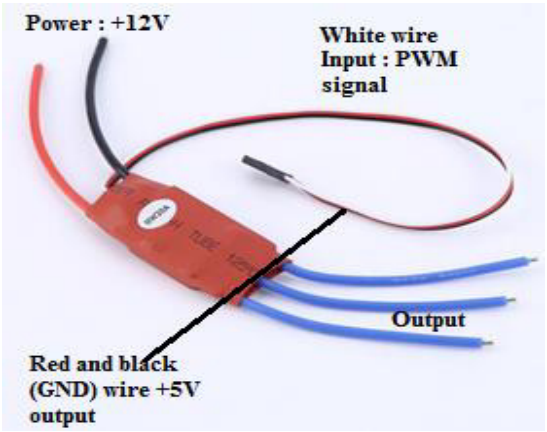

Figure 3.9. ESC module input and output wire

3.10 Arduino Board

The Arduino is a microcontroller board based on the ATmega328. Arduino is an open-source hardware and software. It has a programmable board with input and output pins. It has varies library such as Global Positioning System (GPS), Ethernet, Motor Drives, MPU6050.

Figure 3.10 shows that have 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

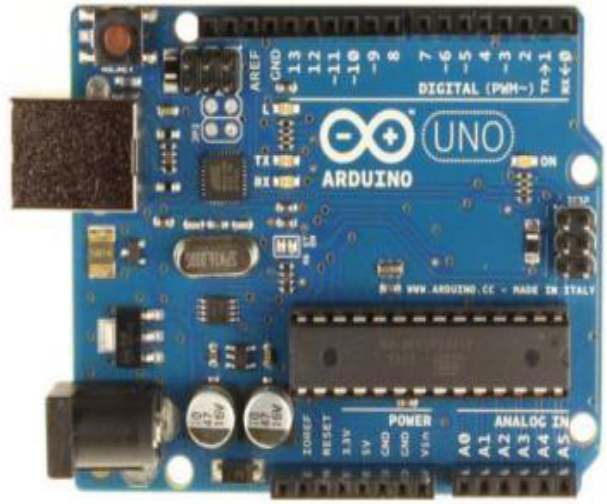

Figure 3.10. Arduino UNO

3.11 MPU-6050 Gyro Sensor

MPU6050 is an IMU sensor board for use in multi-rotor and robotics projects with gyro, angular accelerometer and temperature sensor. The card, which provides information via the I2C communication protocol, has a 3-5 V operating range. With 16 bit resolution, it provides sensitive data for many applications. It has 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor and Temperature sensor in package. It has I2C bus interface to communicate with the microcontrollers. [14]

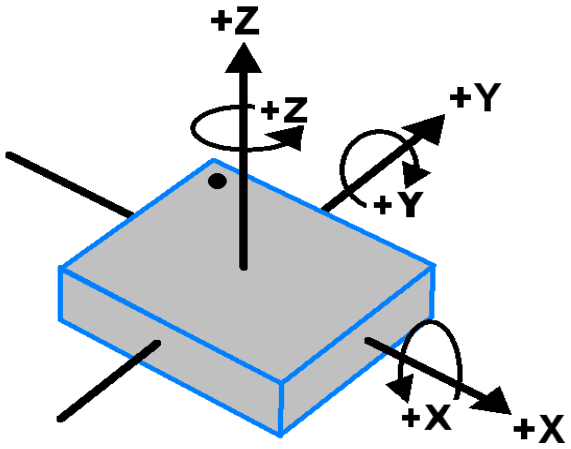

Figure 3.11 Gyro Sensor Positions [14]

3.12 Arduino and MPU6050

The MPU6050 can be easily integrated into the Arduino board. The Arduino I2C has SDA and SCL pins with communication pins, so it can be connected directly. It does not need any crossing as in serial communication. Software Arduino 'wire.h' communicates with other I2C based cards.

3.13 Arduino wire.h library

The MPU6050 acceleration sensor I2C is based on using a 'wire.h' library, Arduino and I2C communication sensors like MPU6050 can communicate in serial. The SCL and SDA pins in the Arduino circuit can communicate with sensors. The functions and descriptions of the library are described below. Basically, I2C protocol is started on the master device. For MPU6050 this address is specified in datasheet as 0x068. The acceleration sensor transmits X, Y, Z accelerometer position, X, Y, Z gyro and temperature information to the master unit at the sampling rate.

- Wire.begin(address) **// Begin communication.**
- Wire.beginTransmission() **//The data register of the slave device is written.**
- Wire.endTransmission() **// Stop commination.**
- Wire.write () **// Write message**

3.14 Connection MPU6050 with Arduino

The first is connected to the VCC pin of the 3.3V or 5V MPU6050 sensor provided by the Arduino circuit for the MPU6050 supply, then the SCL and SDA pins required for the I2C protocol are mutually connected between Arduino and MPU6050. Figure 3.11 shows the circuit diagram of the connection. In this study, Arduino Uno is used and Analog pin 4 (A4), analog pin 5 (A5), I2C communication pins. If the MPU6050 is required to be used in the Arduino code, INT pin can be control with digital pin 2 (D2).

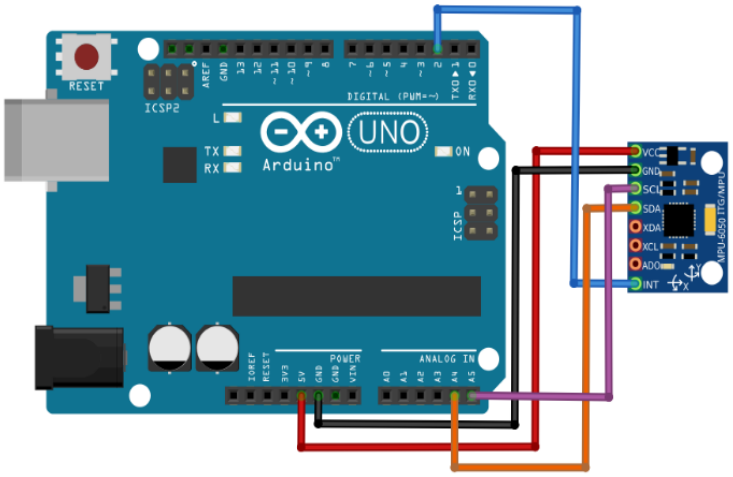

Figure 3.12. Arduino and MPU6050 connection

Position data can be obtained from 3 axes for the position. Since the balance arm moves in 'X' axis, 'X' axis is used in this study.

However, there are lines for all other axes in the code. It is activated in the desired case. In the same way, the temperature sensor is in passive state in the code.

3.15 LABVIEW in History

National Instrument, founded in Austin, Texas, USA in 1976, started its first work with General Purpose Interface Bus-IEEE488 cards. The Data Acquisition (DAQ) card manufacturer for MAC machines began looking for ways to reduce the time required for programming in 1983 and was released in 1986 as DOS-based Macintosh compatible [10].

3.16 Arduino Control with LabVIEW Software

The position of the two BLDC motors on the balance bar is measured by a gyro sensor or analog potentiometer. Figure 3.13 shows the basic elements of the project. MPU6050 or potentiometer is sensors used as position information. When receiving information from the 'A1' potentiometer from the analog inputs of the Arduino, the 'A0' gyro sensor is connected. After the user decides which sensor to select, the measurement results are evaluated in the program created using LABVIEW. MPU6050 sensor uses communication with the I2C. As is known, I2C is a serial communication method. As LABVIEW and Arduino uses serial communication protocol with LabVIEW software and position sensor. Therefore, two Arduino were used. As shown in Figure 3.13. , this Arduino is shown to the right of the MPU6050.

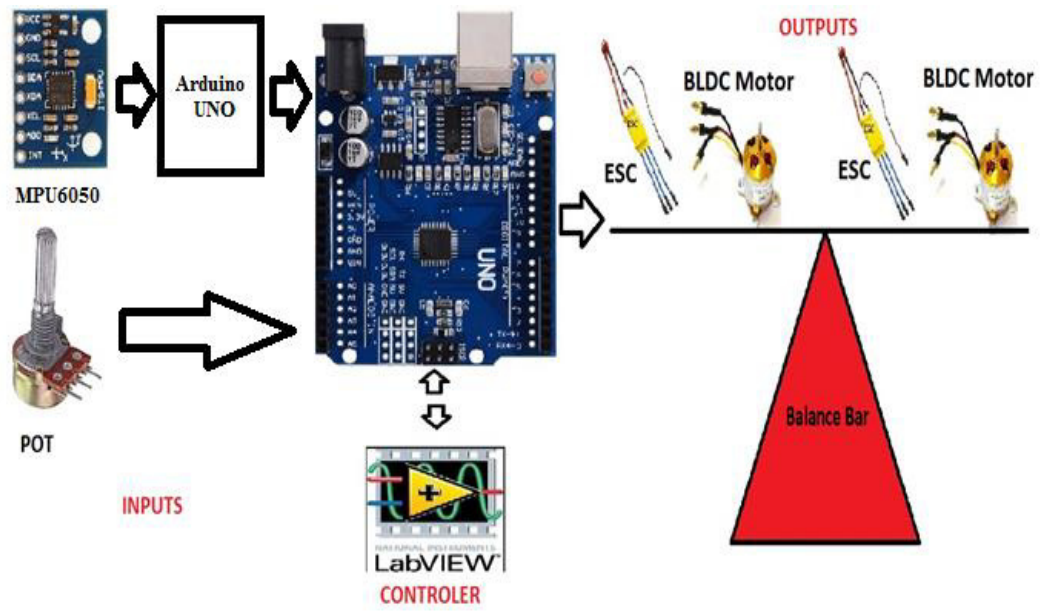

Figure 3.13 Basic Block Diagram

3.17 Digital to Analog Conversion

MPU6050 is I2C based serial communication. Since serial communication with computer is used in the study, position information is converted to analog information by means of a low pass filter and transmitted to the other Arduino.MPU6050 measures the angle of the balance bar in degrees. With the software included in the Arduino, this angular converts it into an 8 bit

data. For example, when it is -90 degrees 0, it will be +90 degrees 255. The simplest way to convert analog data to digital data circuit (DAC) is to design an R-C circuit.

As shown in figure 3.14, the digital information from the gyro sensor is sent to the Arduino digital pin. The digital to analog converter circuit has transformed from PWM signal to an analog signal. For example, when the balance bar is at 0 degrees, the Gyro sensor will produce a value of 127. This value will be 490 Hz frequency, and the amplitude of PWM between 0 and 5 volts will be 2.5 RMS volts.

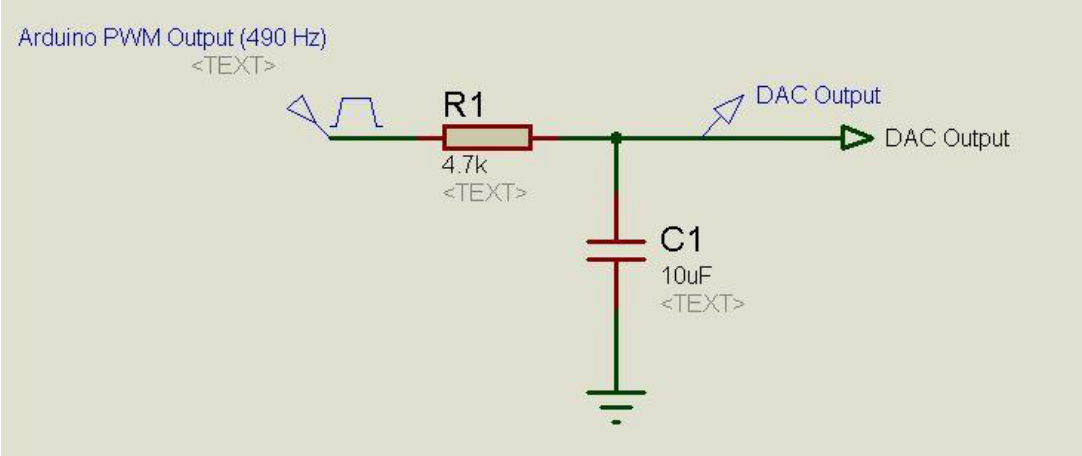

Figure 3.14 Digital to Analog Circuit

Figure 3.14 shows the output signal of the Gyro sensor. This PWM signal is the input of the circuit.

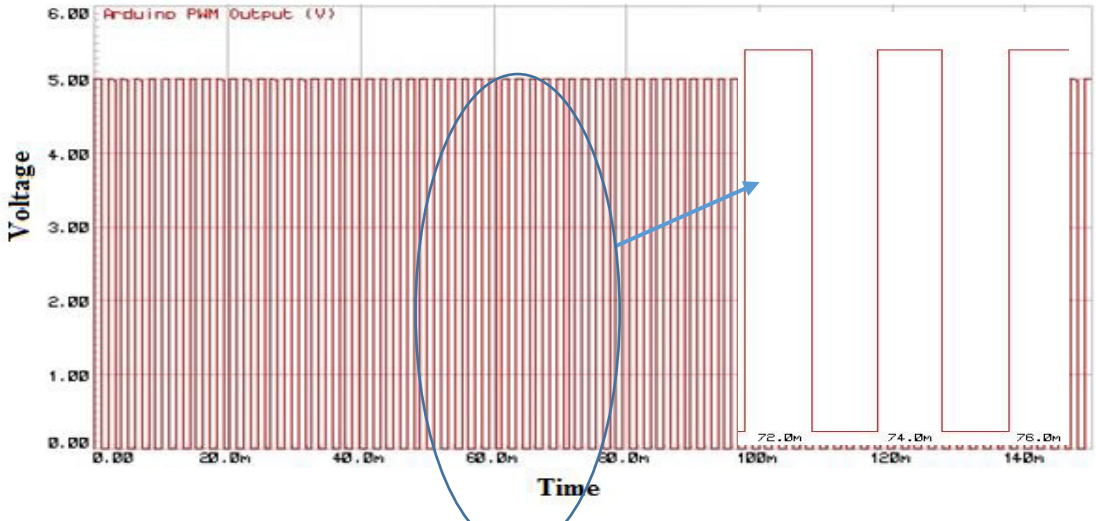

Figure 3.15 Arduino 490Hz Square Wave Output Signal

PWM input of the Low-pass filter is now an analog signal. This signal is the input signal of the feedback control.

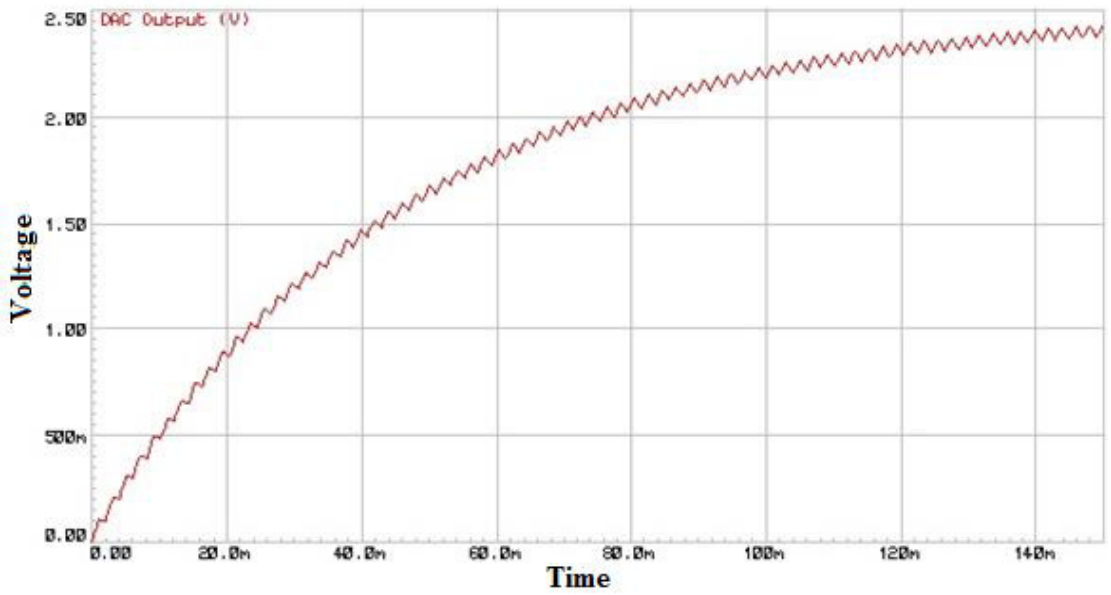

Figure 3.16 R-C circuit output

The 0-5V PWM signal with the DAC circuit is converted to 0-5V analog signal. The duty cycle value of the square wave is 125. So rms value is 2.5 volts. DAC circuit is seen in 140 milliseconds has reached approximately 2.5 V value.

4. RESULTS AND DISCUSSIONS

4.1 Pitch Roll and Yaw

Planes, rockets, unmanned aerial vehicles have three dimensions. These are pitch roll and yaw (Figure 4.1). MPU6050 sensors include accelerometer and gyro information. This data should be converted to pitch roll and yaw information in a 3D plane.
In the project, feedback control was performed using only the pitch axis. Since the motion bar only moves up and down, it is not necessary to use other axes.
Pitch, Roll and Yaw are the rotations about;

- **Roll,** the X axis (between -180 and 180 deg).
- **Pitch,** the Y axis (between -90 and 90 deg).
- **Yaw**, the Z axis (between -180 and 180).

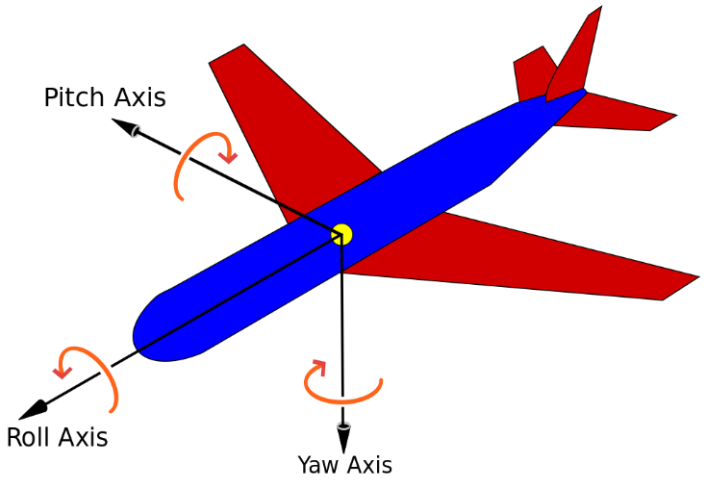

Figure 4.1 Pitch, Roll and Yaw

BLDC motors in the balance bar used in this study move on the Roll axis. In the following equations, the acceleration information for the axes is abbreviated as 'Accel'. In the first equation [11], the Roll axis is calculated by the acceleration data of the X, Y and Z axes. In this way, + / 180 degree change in X axis is calculated.

$$Roll = \tan(AccelY / \sqrt{AccelX^2 + AccelZ^2}) \quad \text{Rad/s} \quad (1)$$

4.2 MPU5060 Measurement Data

The serial communication screen of the Arduino software displays information from the MPU6050 sensor. Arduino serial communication screen 'roll' axis data are analyzed when a lot of jump is observed. Even if there is no movement in the balance bar, it is observed that the measured values change continuously. Figure 4.1 shows the values of the 'roll' axis for different positions of the balance bar. On the graph X-axis, 8-bit data are measured on the Y-axis of 350 samples. Although there is no change in the position of the red circles on the graph, splashes are seen.

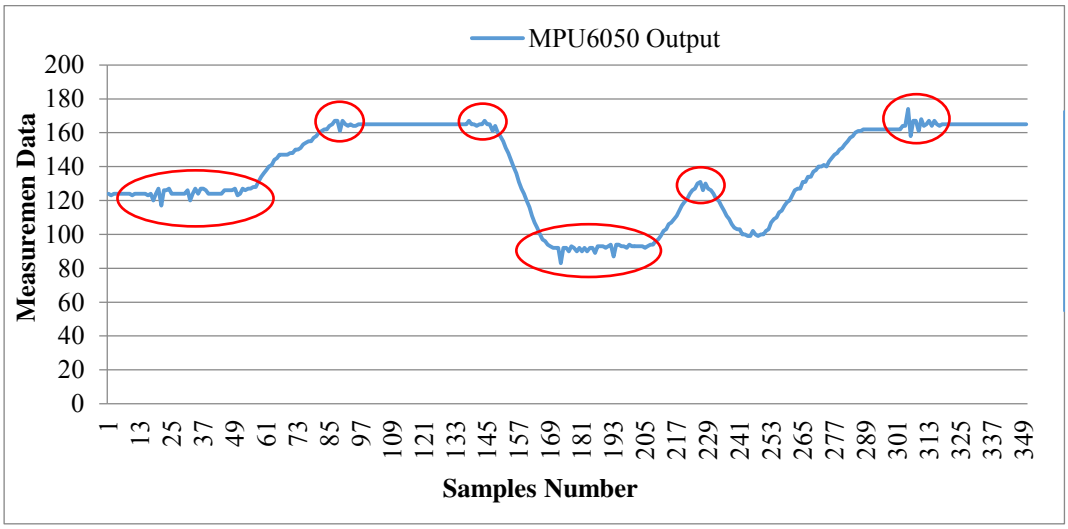

Figure 4.1 Measurement Roll Axis Data

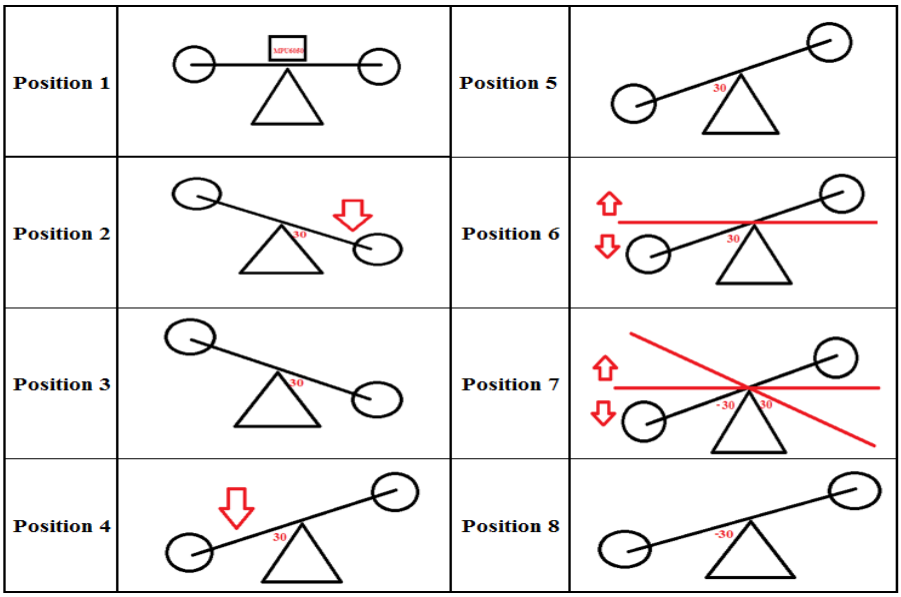

Figure 4.2 Balance Bar Position

When the balance bar moves more-less slowly-rapidly, when the 350 pieces of data containing the position information of the balance bar are divided into 8 sections, the change in the roll axis of any position is examined in Figure 4.2.

Position 1: Balance bar is in the fixed position at 0 degrees; the data of the roll axis data is not stable.

Position 2: Balance bar moves slowly downwards, the data of the roll axis is variable. The angle of the balance bar is about +30 degrees.

Position 3: Balance bar is fixed at +30 degrees.

Position 4: Balance bar is quickly set to -30 degrees from +30 degrees.

Position 5: Balance bar is kept constant in the -30 degree position.

Position 6: Balance bar is set to -30 degrees from 0 to 30 degrees and then again to -30 degrees.

Position 7: Balance bar has been brought from -30 degrees to +30 degrees slowly.

Position 8: Balance Bar is fixed at -30 degrees.

Analysis of the Roll axis from the MPU6050 sensor showed no favorable condition for PID control. Therefore, it was decided to design a filter. Researches and case studies have shown that Kalman and Complementary methods are commonly used methods for location studies. These two methods are examined in detail in the following sections.

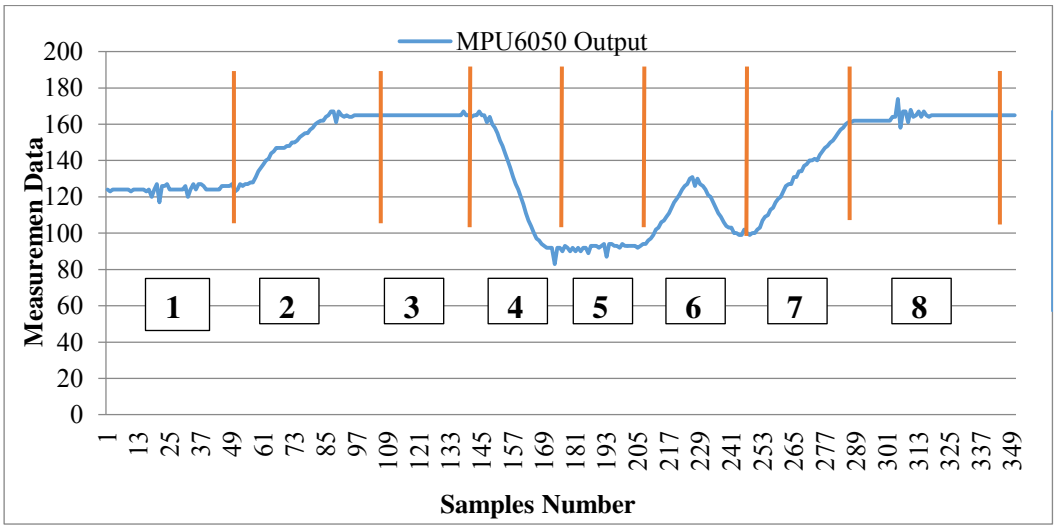

Figure 4.3 Roll axis and Position of Balance Bar

In Figure 4.2, the data measured by the MPU6050 sensor on the graph is divided into 8 different sections. These 8 different cases are examined in detail in figure 4.4. In the summary, the balance bar was brought from 0 degrees to +30 and -30 degrees at different speeds and kept constant at -30 degrees.

4.3 MPU5060 Measurement Data and Filter Design

The complementary filter was first designed to reduce splashes in the roll axis data from the MPU6050 sensor. The filter is software designed in Arduino code. It has been observed that the Complementary filter balance bar is moving, reducing the jumping in the 2-4-6 and 8 regions marked in the graph in Figure 4.2. However, there is still error in the absence of movement and sudden transitions. It is seen that spikes on the graph in figure 4.5 are marked.

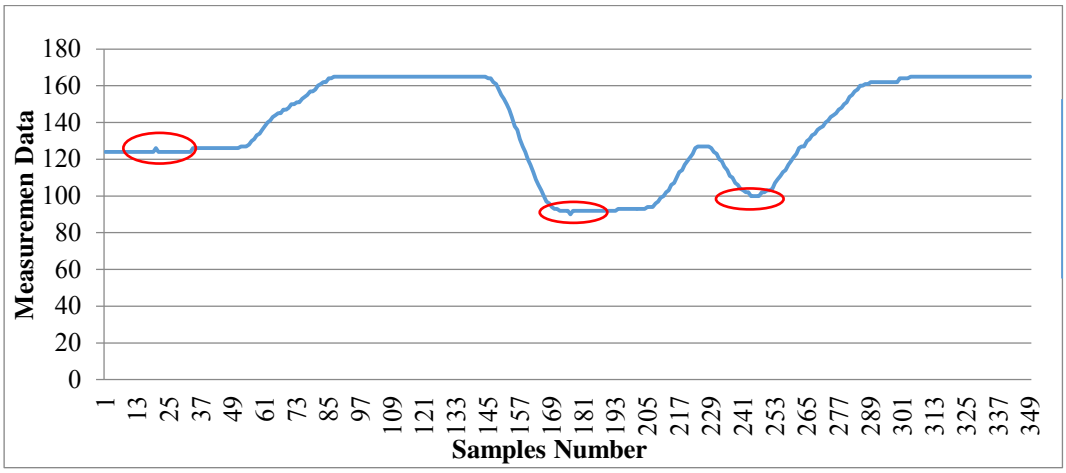

Figure 4.4 Complementary Filter Output

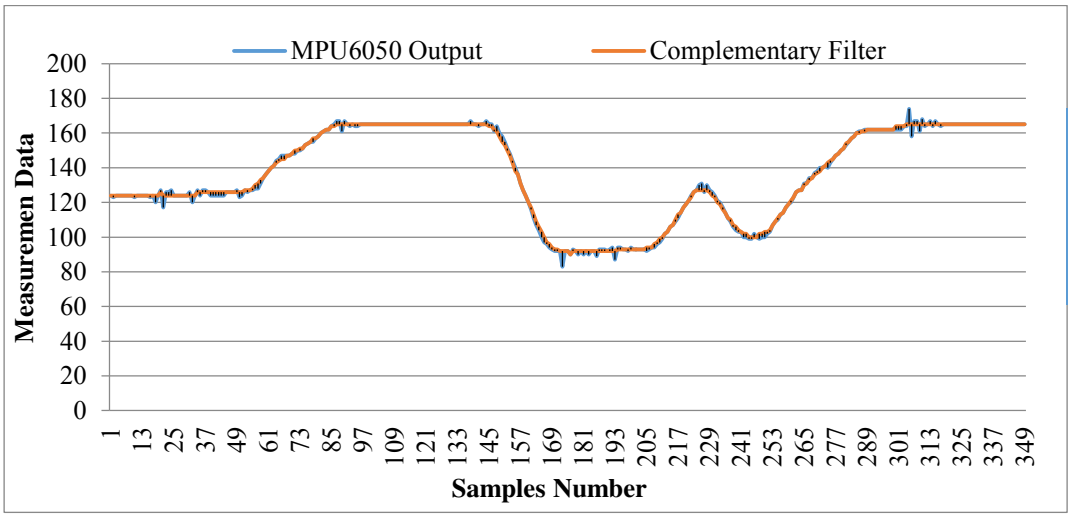

Figure 4.5 Complementary Filter Output and MPU6050 Output

Figure 4.6 shows MPU6050 sensor output and the situation after the filter is examined. In particular, the number of samples in the range 0-73 and 169-205 yields stable data for feedback control.

4.4 Kalman Filter

Kalman filter has a more complex structure than complimentary filter. First, the maximum and minimum values of the axis to be designed for the Kalman filter are assigned as the max and min value of the filter. For example, the roll axis changes range from -180 degrees to +180 degrees. If the measured value is not within this range, the angle of stay is calculated using the calculated roll axis angle in section 3.1.

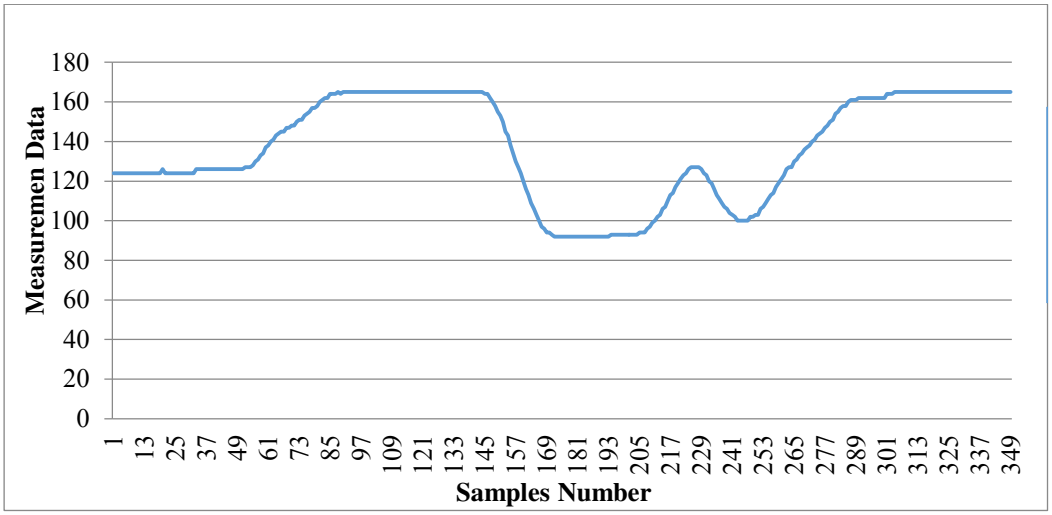

Figure 4.6 Kalman Filter Output

If the measured value is between -180 and +180 degrees, the new angle value is calculated with the variance of the gyro information by the roll angle. In figure 4.7, it is seen that the data from the MPU6050 sensor decreases the spills after the filter is filtered and the data obtained at the filter output change steadily.

In the figure 4.8, it is seen that the Kalman filter avoid the jumps in the data at intervals between 13-61, 169-205, 301-325 where the balance bar is fixed. Likewise, the sample number 205-265, where the equilibrium bar changes its direction, appears to destroy the splashes of the Kalman filter.

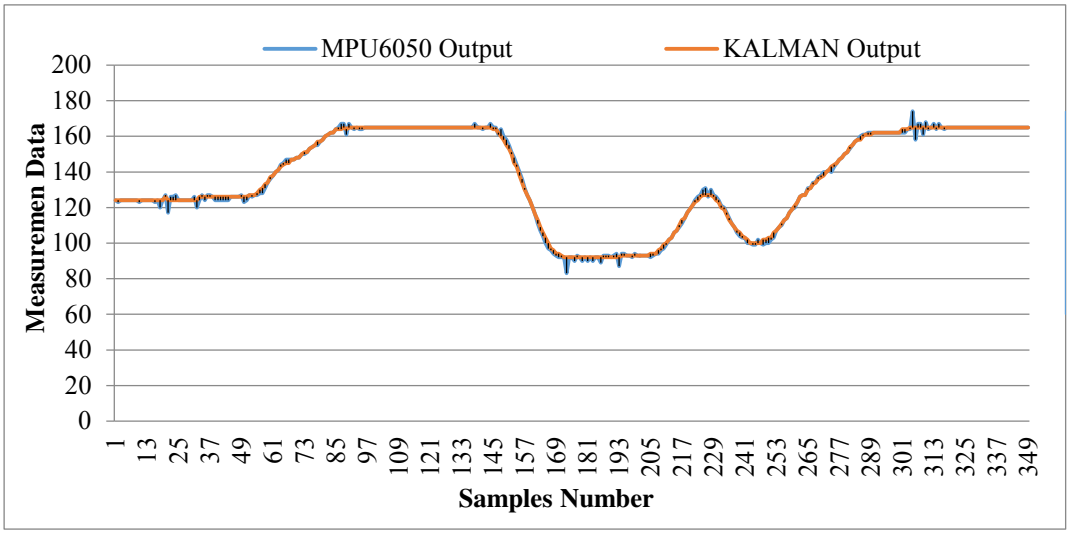

Figure 4.7 Kalman Filter and MPU6050 Output

4.5 A New Approach Mean Value Filter

Analyzing the data obtained from two popular filter MPU6050 sensors and observing the results, a new method was developed at this stage of the thesis. This method is the average value filter, which records the data received from the MPU6050 sensor as an array and takes the average value of each three data and makes the first term of the array.

4.6 Mean Value Filter Algorithm

First, the unfiltered raw data from the MPU6050 sensor is assigned to an array. This process is constantly renewed. Then, when you take the values from this array and averages the desired value through a for loop, the position data remains in a set. For example, in the case of a constant state, the desired state is always the same value, if it is not, there is an error.

The second value of the filter is the arithmetic average of the second value of the array and its newly assigned value six. This process is repeated each time a new value is received. Table 3.2 shows which values are used to calculate the first 5 value of the Mean Value filter.

Table 3.2. Method for calculating the Mean Value Filter output with data from the MPU6050 sensor

Mean Filter Output 1	**Mean Filter Output 2**	**Mean Filter Output 3**	**Mean Filter Output 4**	**Mean Filter Output 5**
Value 1	**Value 2**	**Value 3**	**Value 4**	**Value 5**
Value 2	Value 3	Value 4	Value 5	Value 6
Value 3	Value 4	Value 5	Value 6	Value 7
Value 4	Value 5	Value 6	Value 7	Value 8
Value 5	Value 6	Value 7	Value 8	Value 9
The first value of the filter: The arithmetic mean of these five values	**The second value of the filter:** The arithmetic mean of these five values	**The third value of the filter:** The arithmetic mean of these five values	**The fourth value of the filter:** The arithmetic mean of these five values	**The fifth value of the filter:** The arithmetic mean of these five values

Figure 4.9 examines the performance of the 'Mean Value Filter' as successful as the 'Kalman' and 'Complementary' filters. It is simple algorithm and preferable in applications where very precise controls are not performed. Although the balance bar is constant between the first 50 sampling numbers and 157-205, there are still minor vibrations in the mean value filter output.

When the average of all three information received from the MPU6050 sensor is taken and assigned to the first value of the average value filter, it appears that the jumps continue at the points indicated by the red circle in Figure 4.9.

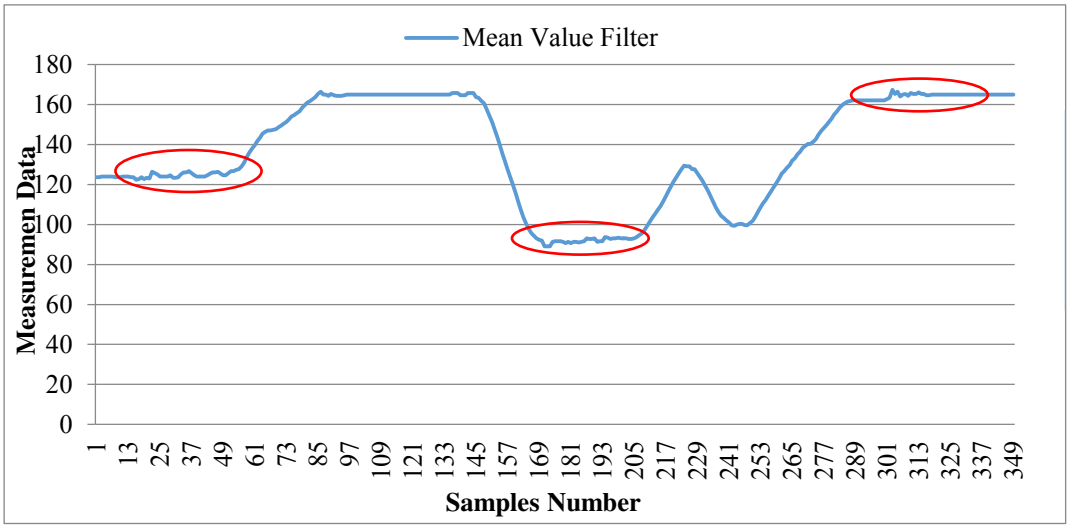

Figure 4.8 Mean value filter the average of three values

In figure 4.10, the average of each 10 value come from the MPU6050 sensor is taken and the first value of the average value filter is occurred. It is observed that the jumps are reduced when the average value of each 10 value is not equal to three values and when the mean value is assigned to the first value of the filter.

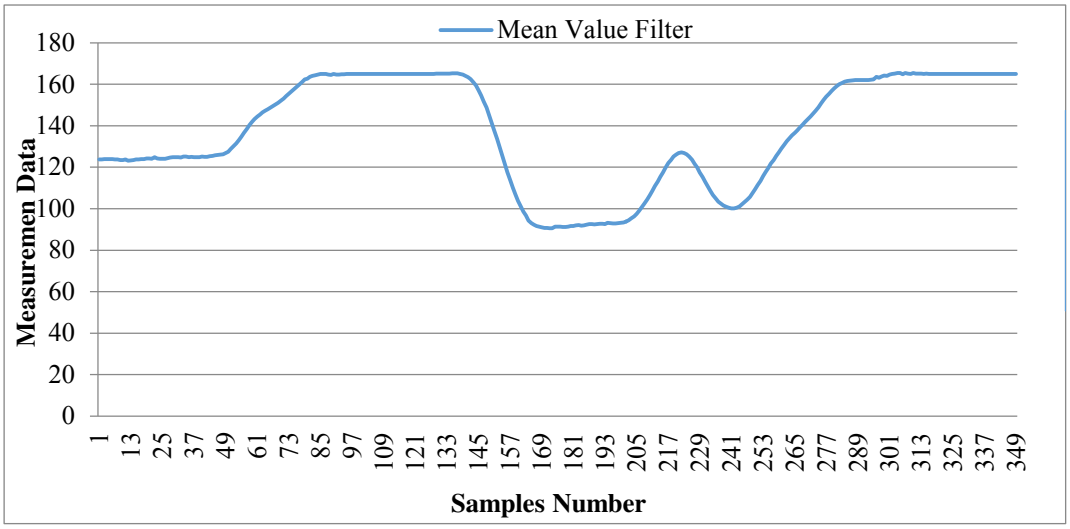

Figure 4.9 Mean value filter the average of ten values

Although the number of points to be averaged, the jumps goes, but this time the deviations are observed to be increased. Figure 4.11 shows the average value filter output and the MPU6050 sensor output when the take three values is averaged and the first value of the average value

In cases where the balance bar is stationary, the deviation in the position is small, although the jumps are high.

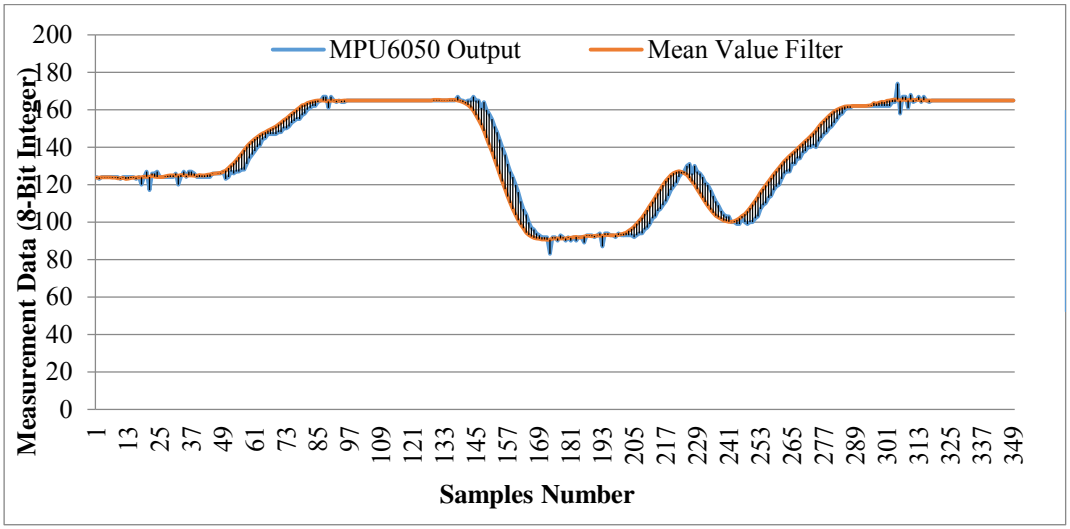

Figure 4.10 MPU6050 Output and Mean Value Filter Use of ten samples

As shown in figure 4.12, splashes continue when the filter output and position are in the same phase.

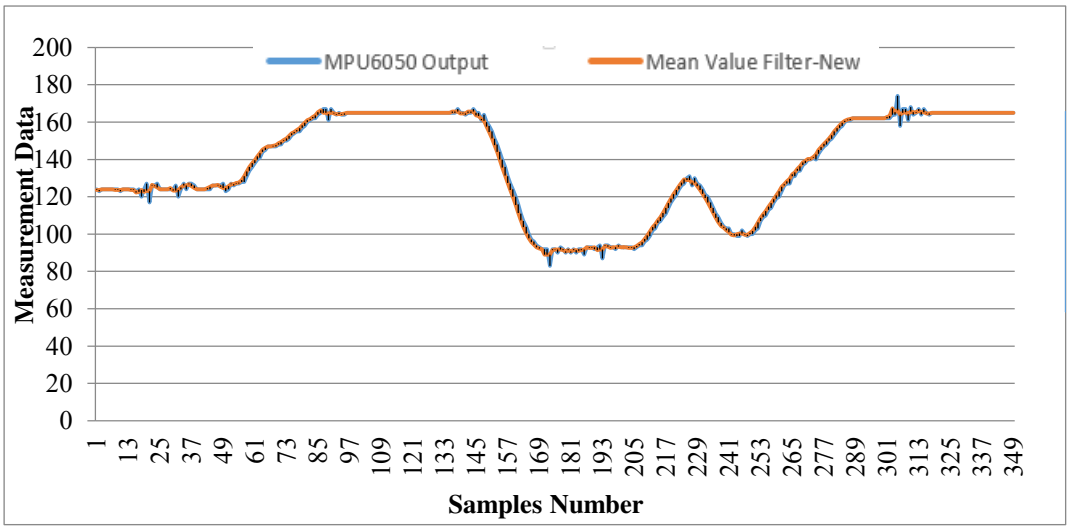

Figure 4.11 MPU6050 Output and Mean Value Filter Use of three samples

Especially in areas where there is no movement, the jumps are not prevented and the motors do not run stably. In cases where phase shift is not very important for the preferred applications, that is to say, very high precision is not required; more than three values can be averaged for the filter.

Table 3.3: New Method for calculating the Mean Value Filter output with data from the MPU6050 sensor

Mean Value 1	**Mean Value 2**	**Mean Value 3**	**Mean Value 4**	**Mean Value 5**
Value 1		**Value 3**		**Value 5**
Value 2		Value 4		Value 6
Value 3		Value 5		Value 7
Value 4		Value 6		Value 8
Value 5		Value 7		Value 9
The first value of the filter: The arithmetic mean of these five values	**Value 2**	**The third value of the filter:** The arithmetic mean of these five values	**Value 3**	**The fifth value of the filter:** The arithmetic mean of these five values

The new method was decided to take the first five values and use a full empty method. For example; the first value of the data will be the average of the first five values of the raw data and the second value will be the measured value. Table 3.3 shows how to account for the first five values.

Figure 4.13 shows that the new method eliminates splashes between the sampling numbers 1-65, 169-225 and 295-323, where the balance bar remains stable, and there is no shift in position. However, the new method follows the vibrations in the raw data, as shown in the zoom construction part.

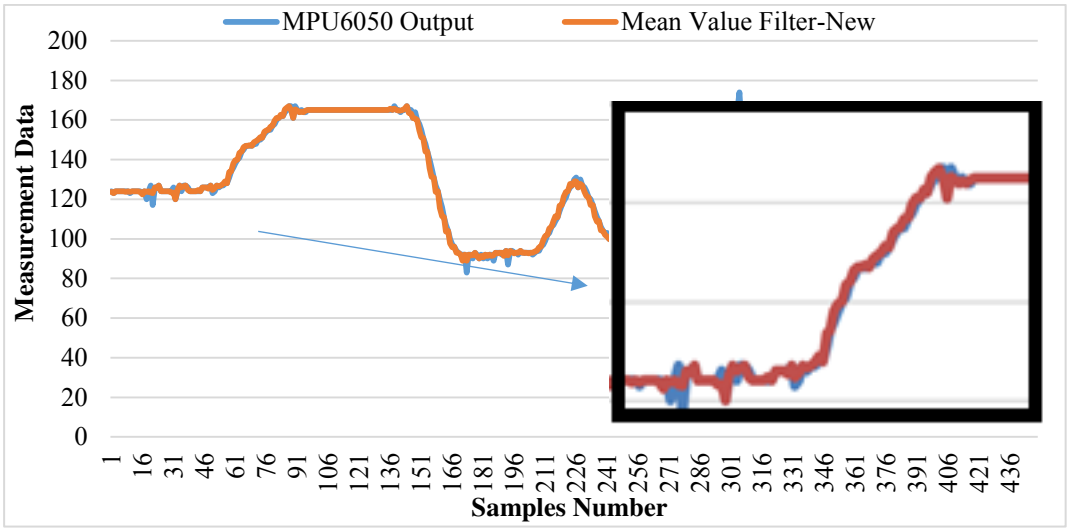

Figure 4.12 MPU6050 Output and New Method Mean Value Filter Use of three samples

4.7 Kalman, Complimentary and Mean value Filter Comparison

BLDC motors can be controlled quickly and at the desired angle three different filters are examined in detail. Although the mean value filter algorithm provides simplicity in terms of its applicability, it is observed that when the balance bar is fixed and the average value of the first three values is assigned to the first value and the mean value of 10 is not determined, it is observed that the jumps are filtered but there is a deviation in the position angle.

The simplest algorithm of the complimentary filter has shown that the filter is simpler than the Kalman filter that there is little deviation in the position angle, which is more effective than other filters It is observed that Kalman filter has a more complex algorithm than the other two filters but it is suitable to be preferred in applications requiring high accuracy and high accuracy output.

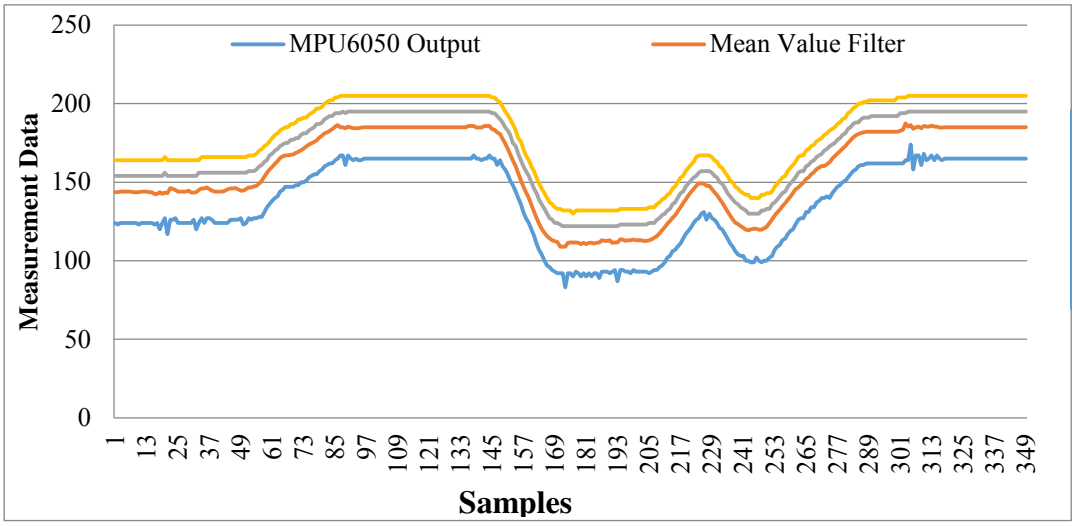

Figure 4.13 Kalman, Complimentary and Mean value Filter Comparison

Figure 4.14 shows the output of the three filters and the graphs of the MPU6050 sensor output. When the first 61 values and 169-205 are examined in the graph, it is seen that Kalman and Complimentary filter is sensitive and low compared to Mean Value filter. It is observed that the mean value filter splashes between the values of 289-313

4.8 PID Control and LabVIEW

PID is one of the most commonly used control algorithm due its ease of use and minimal required knowledge of the system or plant to be controlled. The PID control continuously monitors the output element in a model designed for a set parameter. For example, in a motor speed control, the desired speed is 1000 rpm and the PID control measures the speed of the motor and increases the pulse width of the PWM signal, which is the input parameter of the motor drive below 1000 rpm. Likewise, PWM reduces the pulse width above 1000 rpm.

4.9 Feedback System

Feedback circuits are used in most of the circuits used in the industry or military field in daily life. In figure 4.15, ‘R’ is the desired status in the feedback control circuit, ‘Y’ is the output ‘U’ is the layer in which the error is eliminated, ‘e’ is the error element of the system. Depending on the area to be used, the fault elements can be designed very precisely. If you design a robot in the healthcare industry, there must be a minimum error in every movement the robot makes.

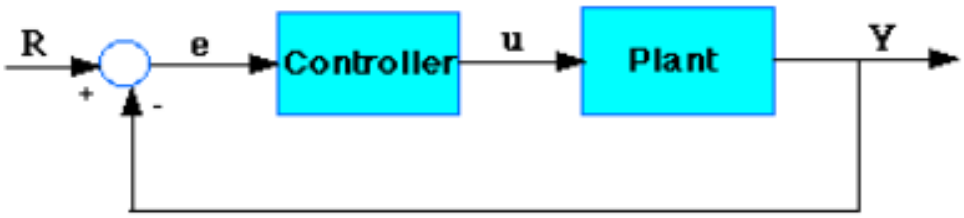

Figure 4.14 Feedback Control Scheme

4.10 PID Controllers

The PID control has three basic elements. These are proportional (P), integral (I) and derivative (D) controllers. Proportional control (P) generates design output depending on the amount of error and Kp coefficient and increases static, dynamic accuracy. Integral control (I) generates design output depending on the amount of error and Ki coefficient, and increases static accuracy [12]. Differential control (D) adjusts the design output and increases the dynamic response and improves it, depending on the rate of change of the error and the coefficient Kd. Table 3.4 shows the response of the system as a result of the change of P, I and D parameters. The error is reduced while the proportional controller and the integral controller generate overshoot on the system. As a result of such fine adjustments, the control parameters of the system will be obtained [13].

Mathematicians can express mathematically as follows;

$$P = Kp(error) \quad (2)$$

$$I = Ki\int error.dt \quad (3)$$

$$D = Kd(de/dt) \quad (4)$$

Table 3.4 System response as a result of changing PID parameters [9]

CONTROLLER	RISE TIME	OVERSHOOT	SETTLING TIME	ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	Small Change

4.11 PID Control with LABVIEW

Closed loop design with LABVIEW can be done easily with control and simulation library. PID clicked on Control and Simulation library PID functions shown in figure 4.16 are reached. After selecting the PID from the function palette in the block diagram, controllers and indicators are added. PID library has VIs used for various applications. In this thesis, PID.vi is used.

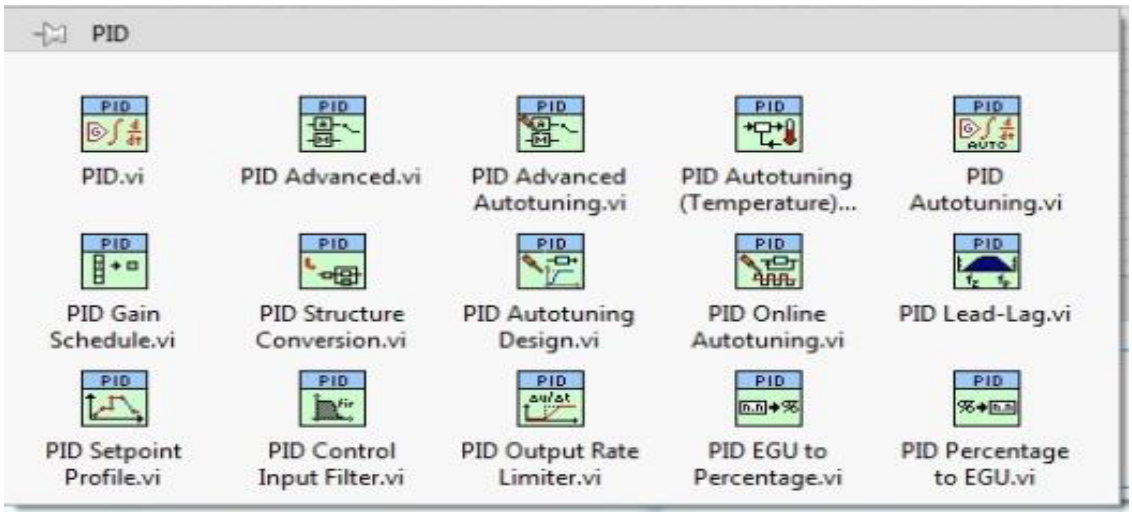

Figure 4.15 PID Functions

As shown in figure 4.17, the key elements of the PID function placed in a while loop are the 'process variable', 'set point', and 'PID gains' and 'output' feedback control output is the difference between process value and set point is the system error. Process value can be a temperature sensor, water level sensor, pressure sensor or position sensor used in this project.

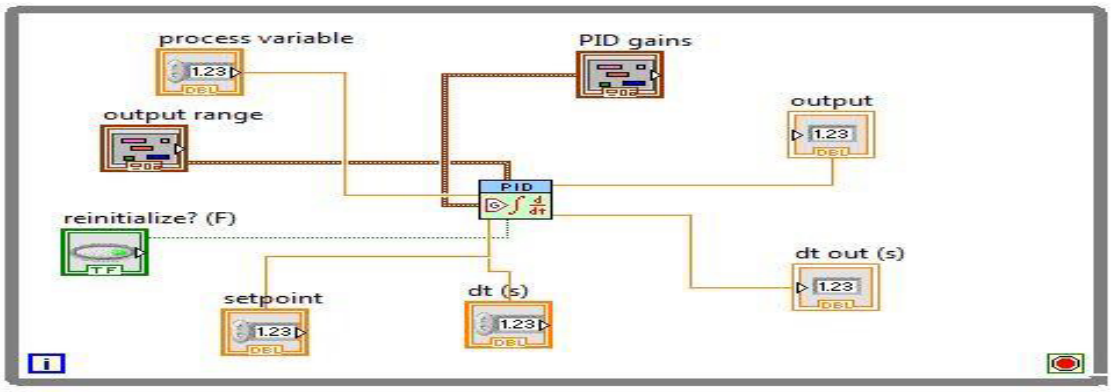

Figure 4.16 PID Control Block Diagram

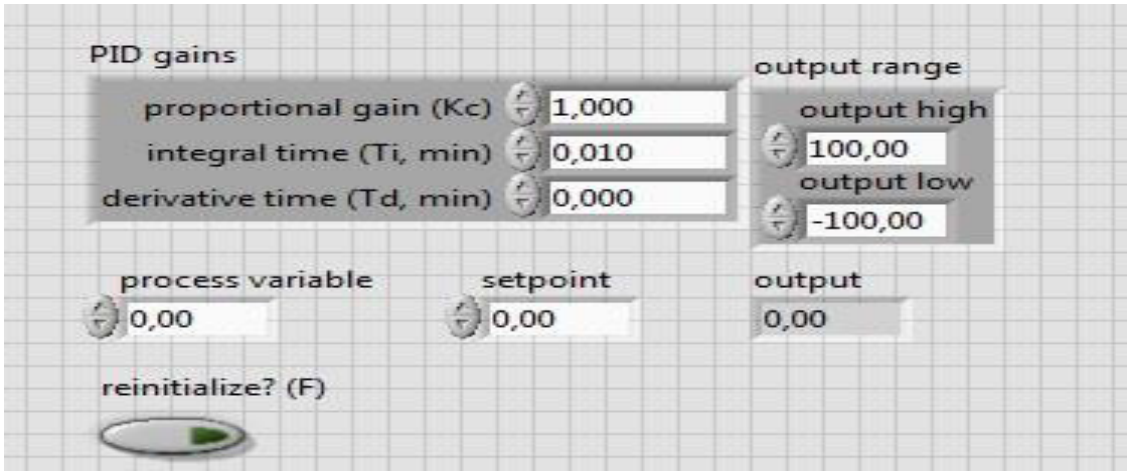

Figure 4.17 PID Control Front Panel

When the controllers and indicators in figure 4.17 are added in the block diagram, the user interface in the front panel is created in figure 4.18. Design input PID gain consists of three basic controllers. Kc, Ti, and Td are the input parameters of the controllers P, I, and D. By means of the up-down arrows next to the controller's icon, or manually, the values can be changed.

The desired set point is controlled in the same way. Output is an indication of the output of the system. Likewise, the maximum and minimum ratio of the output element can be changed by the user. The 'reinitialize' button is set to ten when the PID parameters are to be checked automatically.

4.12 User Interface

The LabVIEW user interface is the computer software shown in Figure 7.5, which consists of five sections, in which the P-I-D parameters, the desired position and the position sensor selection are made.

Section 1:

This section is used to select USB port for serial communication.The com port address to which the Arduino is connected can be seen from the point of connection after the computer-features and device manager steps are followed.

Section 2:

This section where the sensor to be received is selected to use MPU6050 or adjustable resistance.

Section 3:

This section that there is display bars where position information of both motors are taken instantly and set point controller where to determine the position of the motor.

Section 4:

PID parameters are set manually or automatically.

Section 5:

The position of the right and left BLDC motors in the set point and balance bar is planned to be seen instantly.

The block diagram seen in figure 4.19 is the part of the program running in a ‘while loop’ where the main body of the program is made. Basically it consists of four parts. The first part is the communication band rate and com-port where the communication entered is started and the pins to be controlled are adjusted. The second part is the section where the right and left motor drives are controlled by PWM input. The third part is the section where PID control is included in the program. The response of the motors to any parameter entered by the manually set PID parameters can be calculated. In both cases, both motors can be independently controlled via automatic feedback.

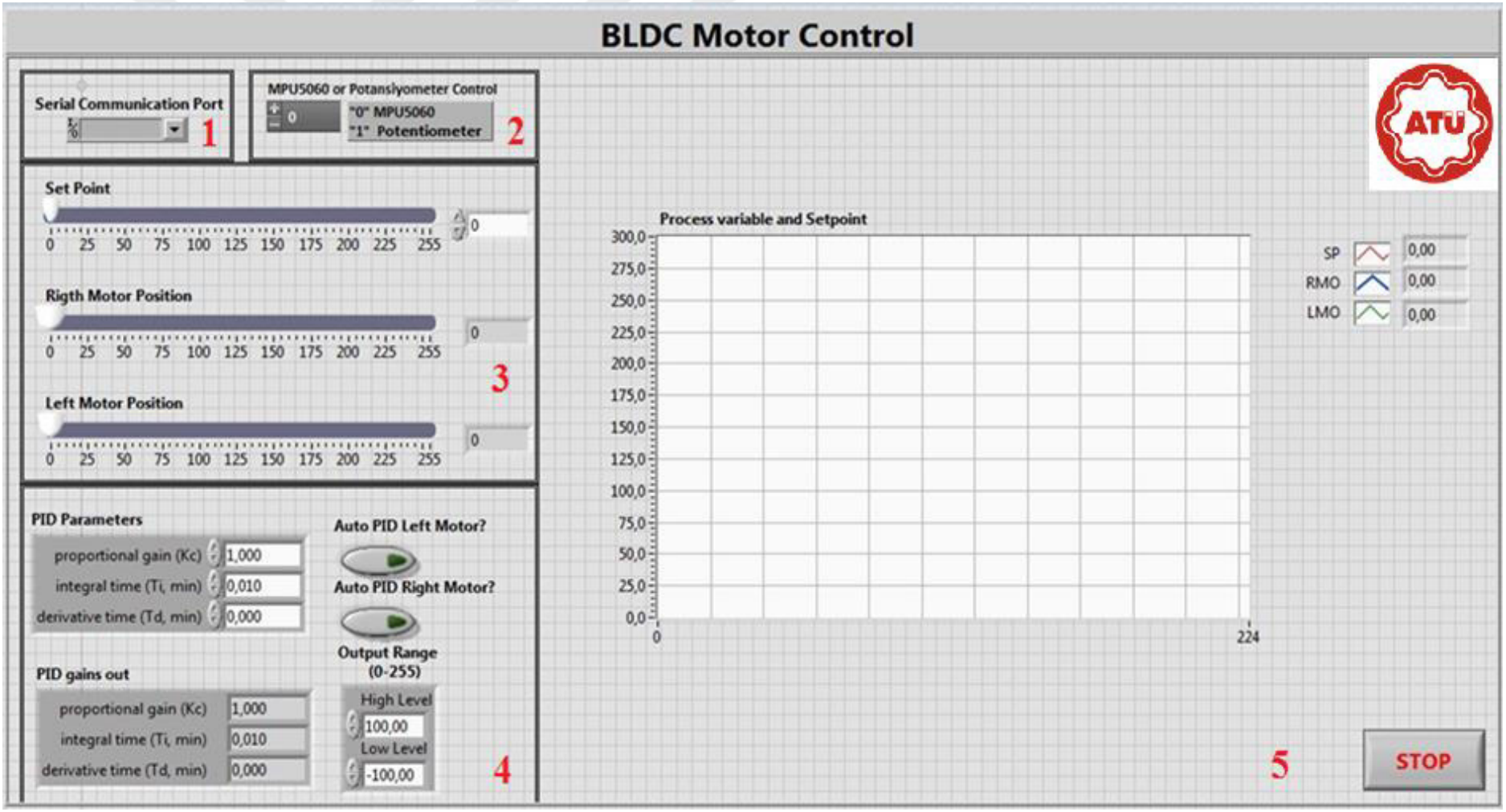

Figure 4.18 BLDC motor control user interface

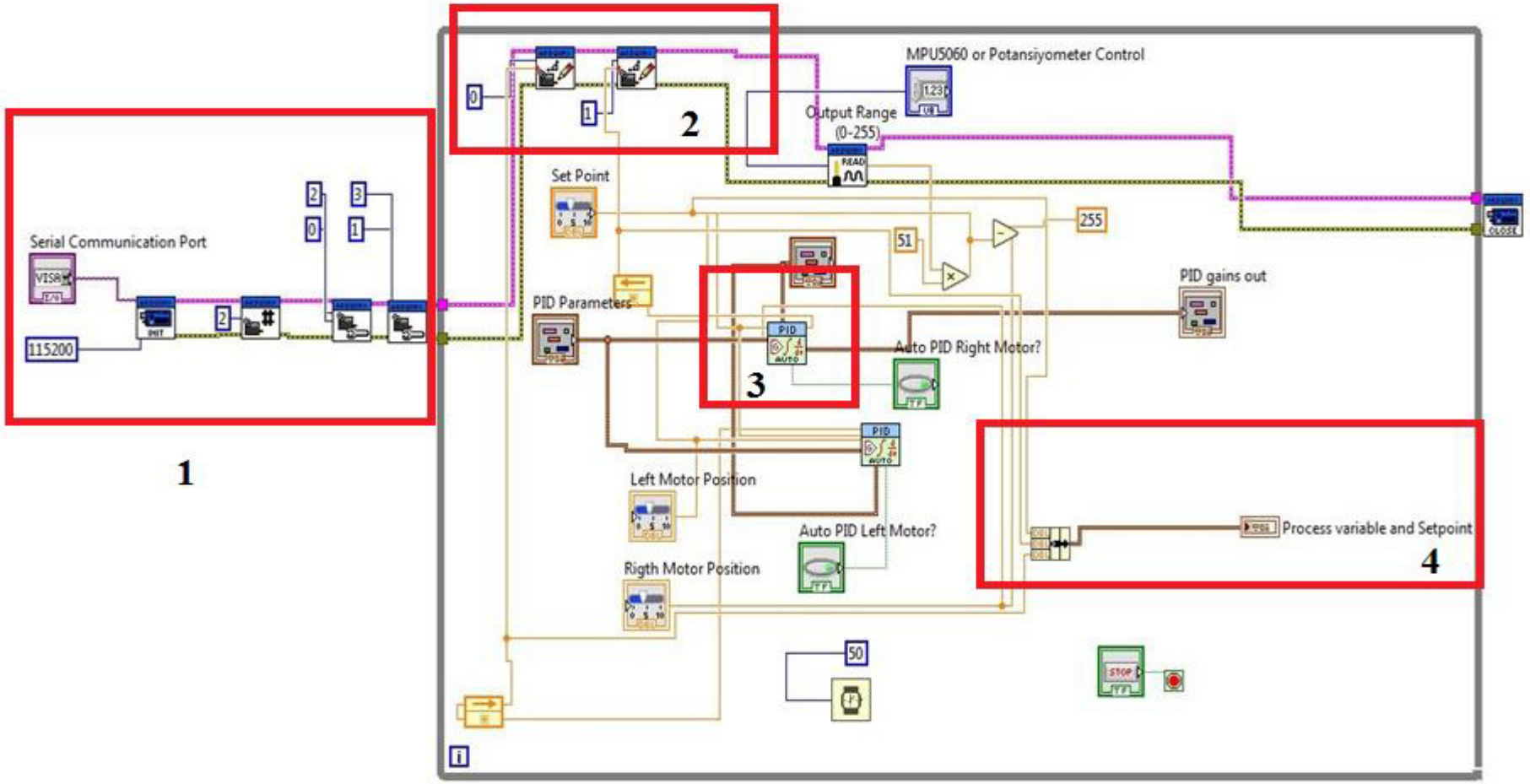

Figure 4.19 BLDC motor control block diagram

4.13 Experimental Work

The user interface is created with LABVIEW and the required program is written and two BLDC motors are placed on a plastic chassis for the experimental set. The MPU6050 gyro sensor and additional potentiometer are mounted in the middle of the plastic chassis. The position information from the first Arduino and the MPU6050 and potentiometer were converted into analog signals and sent to the Arduino 2. It provides communication with LABVIEW. Drivers of BLDC motors are ESC 1 and ESC 2. The control terminals of the drives are connected to pin 2 and pin 3 from the PWM outputs of the Arduino 2.

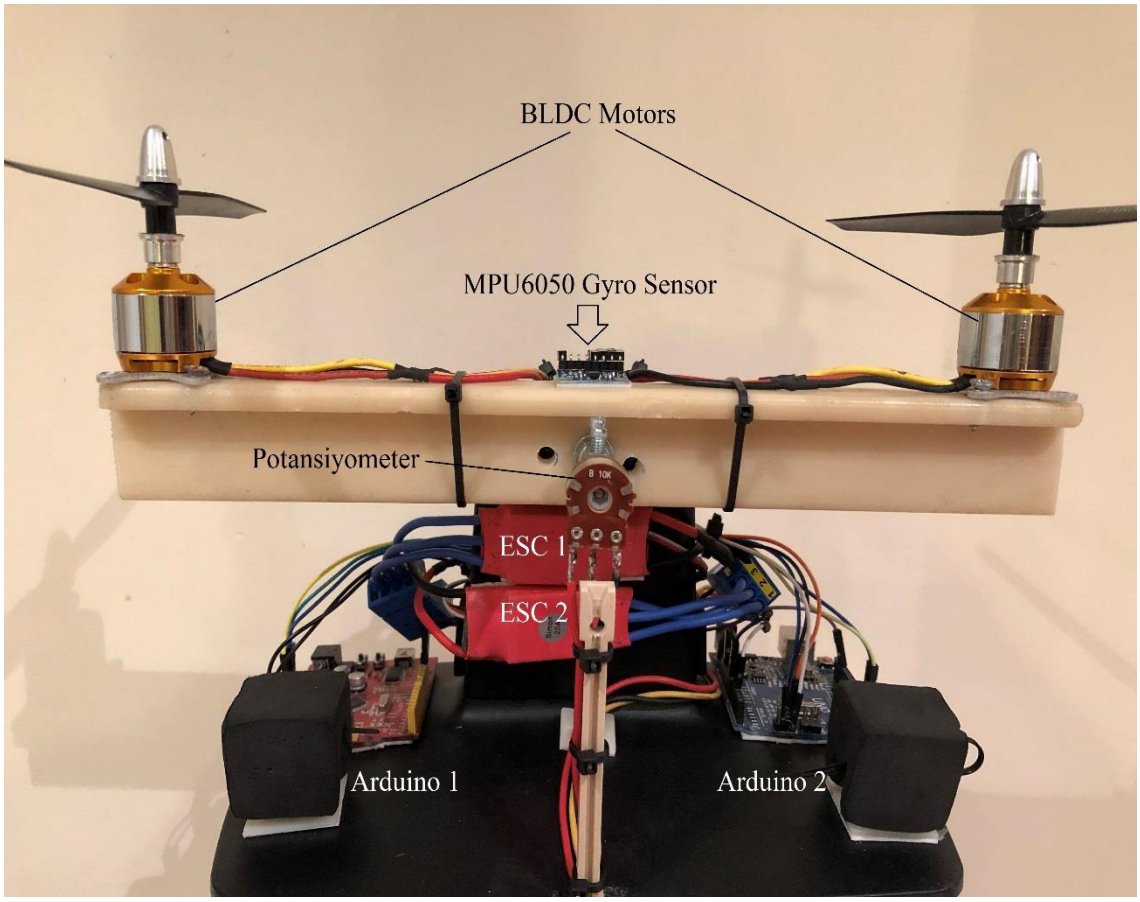

Figure 4.20 Balance Bar

The instances in which the above test kit stabilizes itself according to different P-I-D parameters can be seen from the video shared in references 16.

5.CONCLUSION

BLDC motors are widely used in rotating propeller flight equipment and in autonomous flights, the balance regulation of the device is provided by the data obtained from acceleration and xyz-coordinate sensors. Also, users will be able to choose these engines for different applications after analyzing the engine's characteristic and response.

In this study, it was observed that the sensors used in rotary wing flight equipment were affected negatively by external influences and especially the negative effects on xyz-coordinate data obtained from Gyro sensor were investigated. Within the scope of this thesis, these effects were tried to be reduced by using various digital filters and then a new filter was designed.In the case studies, the results of the proposed method and other methods were compared and it was observed that the proposed method yielded very close results with the most commonly used Kalman Filtering method. When the algorithms speeds of these two methods are compared with microcontrollers, it is seen that the proposed method is performed much faster than Kalman Filter.

The experimental system developed within the scope of the thesis has been turned into a training set and also the user interface is designed by using LABVIEW. It has added basic PI and PID controllers to the interface for system control, and the controller can be automatically discarded by both the user and the feature used in the interface. In the future, it is planned to add four BLDC motors to the system, Fuzzy and Neural Network control algorithms on the user interface, and the communication between them will be made with the new technology LORA communication protocol.

REFERENCES

[1] Çağan, N. (2015). Design of an outer-rotor brushless dc motor for control moment gyroscope applications. Master's thesis. Middle East Technical University. Ankara.

[2] Kurdoğlu, A. (2007).Brushless dc motor speed control circuit design. Master's thesis. Istanbul Technical University. İstanbul.

[3] Ulu, B. (2011). Brushless Dırect Current Motor (Bldc) Speed Control Master's thesis. İnönü University. Malatya.

[4] Gödekoğlu, H. (2007). Design of Brushless DC motor position controller Master's thesis. İstanbul Technical University. İstanbul.

[5] Topal, M. (2019). Control and analysis the speed of a brushless dc motor using three different methods Master's thesis. Gazi University. Ankara.

[6] Karaoğlu, S. (2015). Brushless dc motor training toolkit and interface Master's thesis. Marmara University. İstanbul.

[7] Uygur, S. (2015). An FPGA Based Bldc Motor Control System Master's thesis. Middle East Technical University. Ankara.

[8] Jian, Z.Y, Yu. (2011).The Future of Analog IC Technology Brushless DC Motor Fundamentals application note ,

[9] Electronicwings Platform (2019, Feb 15). MPU6050 (Gyroscope + Accelerometer + Temperature) Sensor Module, from http://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module

[10] Ünsaçar, F. Eşme, E.(2011). Graphic Programming Language LABVIEW, Editor, Cognition: (pp. 1-10). Ankara : Seçkin Yayıncılık.

[11] Pedley, M. (2013).Tilt Sensing using a three-axis accelerometer application note, from https://cache.freescale.com/files/sensors/doc/app_note/AN3461.pdf

[12] National Instruments. (2012). PID Control, from http://www.ni.com/tutorial/6440/en/

[13] Ömürlü, V, Feedback control systems , from http://www.yildiz.edu.tr/~omurlu/CF/OKI/12.pdf

[14]National Instruments. (2001). PID Control Toolset user manual, from http://experimentationlab.berkeley.edu/sites/default/files/PID_Manual.pdf

[15] Özdemir, Ş. (2013). Control of Brushless Dc Motor Used in Unmanned Aırcraft Master's thesis. Marmara University. İstanbul.

[16] Experimental work video link: https://www.youtube.com/watch?v=YnEhakzmeQw

APPENDIX 1

Arduino and MPU6050 Connection

```
#include <Wire.h>
#include <Kalman.h>
#define RESTRICT_PITCH
Kalman kalmanX; // Create the Kalman instances
Kalman kalmanY;
/* IMU Data */
double accX, accY, accZ;
double gyroX, gyroY, gyroZ;
int16_t tempRaw;
double gyroXangle, gyroYangle; // Angle calculate using the gyro only
double compAngleX, compAngleY; // Calculated angle using a complementary filter
double kalAngleX, kalAngleY; // Calculated angle using a Kalman filter
uint32_t timer;
uint8_t i2cData[14]; // Buffer for I2C data

// TODO: Make calibration routine

void setup() {
  Serial.begin(9600);
  Wire.begin();
#if ARDUINO >= 157
  Wire.setClock(400000UL); // Set I2C frequency to 400kHz
#else
  TWBR = ((F_CPU / 400000UL) - 16) / 2; // Set I2C frequency to 400kHz
#endif
  i2cData[0] = 7; // Set the sample rate to 1000Hz - 8kHz/(7+1) = 1000Hz
  i2cData[1] = 0x00; // Disable FSYNC and set 260 Hz Acc filtering, 256 Hz Gyro filtering, 8 KHz sampling
  i2cData[2] = 0x00; // Set Gyro Full Scale Range to ±250deg/s
  i2cData[3] = 0x00; // Set Accelerometer Full Scale Range to ±2g
  while (i2cWrite(0x19, i2cData, 4, false)); // Write to all four registers at once
  while (i2cWrite(0x6B, 0x01, true)); // PLL with X axis gyroscope reference and disable sleep mode

  while (i2cRead(0x75, i2cData, 1));
  if (i2cData[0] != 0x68) { // Read "WHO_AM_I" register
    Serial.print(F("Error reading sensor"));
    while (1);
  }

  delay(100); // Wait for sensor to stabilize

  /* Set kalman and gyro starting angle */
  while (i2cRead(0x3B, i2cData, 6));
  accX = (int16_t)((i2cData[0] << 8) | i2cData[1]);
  accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
  accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);

  // Source: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf eq. 25 and eq. 26
  // atan2 outputs the value of -? to ? (radians) - see http://en.wikipedia.org/wiki/Atan2
  // It is then converted from radians to degrees
#ifdef RESTRICT_PITCH // Eq. 25 and 26
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
#else // Eq. 28 and 29
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif

  kalmanX.setAngle(roll); // Set starting angle
  kalmanY.setAngle(pitch);
  gyroXangle = roll;
  gyroYangle = pitch;
  compAngleX = roll;
  compAngleY = pitch;

  timer = micros();
}

void loop() {
  /* Update all the values */
  while (i2cRead(0x3B, i2cData, 14));
  accX = (int16_t)((i2cData[0] << 8) | i2cData[1]);
  accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
  accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);
  tempRaw = (int16_t)((i2cData[6] << 8) | i2cData[7]);
  gyroX = (int16_t)((i2cData[8] << 8) | i2cData[9]);
  gyroY = (int16_t)((i2cData[10] << 8) | i2cData[11]);
  gyroZ = (int16_t)((i2cData[12] << 8) | i2cData[13]);;

  double dt = (double)(micros() - timer) / 1000000; // Calculate delta time
  timer = micros();
```

```
 // Source: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf eq. 25 and eq. 26
 // atan2 outputs the value of -? to ? (radians) - see http://en.wikipedia.org/wiki/Atan2
 // It is then converted from radians to degrees
#ifdef RESTRICT_PITCH // Eq. 25 and 26
 double roll  = atan2(accY, accZ) * RAD_TO_DEG;
 double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
#else // Eq. 28 and 29
 double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
 double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif

 double gyroXrate = gyroX / 131.0; // Convert to deg/s
 double gyroYrate = gyroY / 131.0; // Convert to deg/s

#ifdef RESTRICT_PITCH
 // This fixes the transition problem when the accelerometer angle jumps between -180 and 180 degrees
 if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90)) {
   kalmanX.setAngle(roll);
   compAngleX = roll;
   kalAngleX = roll;
   gyroXangle = roll;
 } else
   kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); // Calculate the angle using a Kalman filter

 if (abs(kalAngleX) > 90)
   gyroYrate = -gyroYrate; // Invert rate, so it fits the restriced accelerometer reading
 kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
#else
 // This fixes the transition problem when the accelerometer angle jumps between -180 and 180 degrees
 if ((pitch < -90 && kalAngleY > 90) || (pitch > 90 && kalAngleY < -90)) {
   kalmanY.setAngle(pitch);
   compAngleY = pitch;
   kalAngleY = pitch;
   gyroYangle = pitch;
 } else
   kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt); // Calculate the angle using a Kalman filter

 if (abs(kalAngleY) > 90)
   gyroXrate = -gyroXrate; // Invert rate, so it fits the restriced accelerometer reading
 kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); // Calculate the angle using a Kalman filter
#endif

 gyroXangle += gyroXrate * dt; // Calculate gyro angle without any filter
 gyroYangle += gyroYrate * dt;
 //gyroXangle += kalmanX.getRate() * dt; // Calculate gyro angle using the unbiased rate
 //gyroYangle += kalmanY.getRate() * dt;

 compAngleX = 0.93 * (compAngleX + gyroXrate * dt) + 0.07 * roll; // Calculate the angle using a Complimentary filter
 compAngleY = 0.93 * (compAngleY + gyroYrate * dt) + 0.07 * pitch;

 // Reset the gyro angle when it has drifted too much
 if (gyroXangle < -180 || gyroXangle > 180)
   gyroXangle = kalAngleX;
 if (gyroYangle < -180 || gyroYangle > 180)
   gyroYangle = kalAngleY;

 /* Print Data */
#if 0 // Set to 1 to activate
 Serial.print(accX); Serial.print("\t");
 Serial.print(accY); Serial.print("\t");
 Serial.print(accZ); Serial.print("\t");

 Serial.print(gyroX); Serial.print("\t");
 Serial.print(gyroY); Serial.print("\t");
 Serial.print(gyroZ); Serial.print("\t");

 Serial.print("\t");
#endif
//Serial.print("\n");
 //Serial.print(roll); Serial.print("\t");
//  Serial.print(gyroXangle); Serial.print("\t");
//  Serial.print(compAngleX); Serial.print("\t");
//  Serial.print(kalAngleX); Serial.print("\t");
  // int valroll = map(roll,-90,+90,0,255);
  // int valgyro = map(gyroXangle,-90,+90,0,255);
//   int valcomp = map(compAngleX,-90,+90,0,255);
   int valanglex = map(kalAngleX,-90,+90,0,255);
   //int valdigital = map(kalAngleX,-90,+90,0,255);
 //  Serial.print(valroll);Serial.print("\t");
 //   Serial.print(valgyro);Serial.print("\t");
```

```
//    Serial.print(valcomp);Serial.print("\t");
 //    Serial.print(valanglex);Serial.print("\t");
//  Serial.print("\n");

// Serial.print(pitch); Serial.print("\t");
//  Serial.print(gyroYangle); Serial.print("\t");
// Serial.print(compAngleY); Serial.print("\t");
//  Serial.print(kalAngleY); Serial.print("\t");
// int val = map(kalAngleX,-90,+90,0,255);
// Serial.print(val);Serial.print("\t");
analogWrite(9,valanglex);
// Serial.print(kalAngleY);
 //Serial.print("\t");
//Serial.print("\n");
#if 0 // Set to 1 to print the temperature
 Serial.print("\t");

 double temperature = (double)tempRaw / 340.0 + 36.53;
 Serial.print(temperature); Serial.print("\t");
#endif

// Serial.print("\r\n");
 delay(2);
}
//i2c communication code

const uint8_t IMUAddress = 0x68; // AD0 is logic low on the PCB
const uint16_t I2C_TIMEOUT = 1000; // Used to check for errors in I2C communication

uint8_t i2cWrite(uint8_t registerAddress, uint8_t data, bool sendStop) {
 return i2cWrite(registerAddress, &data, 1, sendStop); // Returns 0 on success
}

uint8_t i2cWrite(uint8_t registerAddress, uint8_t *data, uint8_t length, bool sendStop) {
 Wire.beginTransmission(IMUAddress);
 Wire.write(registerAddress);
 Wire.write(data, length);
 uint8_t rcode = Wire.endTransmission(sendStop); // Returns 0 on success
 if (rcode) {
  Serial.print(F("i2cWrite failed: "));
  Serial.println(rcode);
 }
 return rcode; // See: http://arduino.cc/en/Reference/WireEndTransmission
}

uint8_t i2cRead(uint8_t registerAddress, uint8_t *data, uint8_t nbytes) {
 uint32_t timeOutTimer;
 Wire.beginTransmission(IMUAddress);
 Wire.write(registerAddress);
 uint8_t rcode = Wire.endTransmission(false); // Don't release the bus
 if (rcode) {
  Serial.print(F("i2cRead failed: "));
  Serial.println(rcode);
  return rcode; // See: http://arduino.cc/en/Reference/WireEndTransmission
 }
 Wire.requestFrom(IMUAddress, nbytes, (uint8_t)true); // Send a repeated start and then release the bus after reading
 for (uint8_t i = 0; i < nbytes; i++) {
  if (Wire.available())
   data[i] = Wire.read();
  else {
   timeOutTimer = micros();
   while (((micros() - timeOutTimer) < I2C_TIMEOUT) && !Wire.available());
   if (Wire.available())
    data[i] = Wire.read();
   else {
    Serial.println(F("i2cRead timeout"));
    return 5; // This error value is not already taken by end Transmission
   }
  }
 }
 return 0; // Success
}
```