Dedication

We would like to express our gratitude to our parents, our families our friends and our colleagues who always encourages us to excellence and delivering the best, and they were in all stages of the project.

They have always been a beacon for us in the way of our lives until we got to this success. This achievement and coming achievement is meaningless without their presence beside us contactable.

We express our love and gratitude and deep respect for them.

Acknowledgments

First of all, we would like to thank Allah, then our thanks go to our parents, supervisor Dr. Iyad Hashlamoun, and teaching and administrative staff at Palestine Polytechnic University in general and Mechanical Engineer Department in particular for their great patience and work with the project team and their assistance to get this work.

Last but not least we would like to thank our friends, colleagues and the rest of the family for their everlasting love and support.

Abstract

The stabilization of the camera during recording videos is one of the most challenges which facing any photographer whether he is professional or beginner.

This project aims to compensate the vibration from photographer's body which acts on the recorded video by stabilize the camera in two-axes to get smooth record of video regardless of photographer's movement.

To achieve the project goal, there are two motor to stabilize the camera, one motor for each axis. The gyroscope and accelerometer which placed on the base of the camera gives the feedback of the current position of the camera to the control unit.

الملخّص

استقرار الكاميرا والحفاظ على ثباتيتها أثناء تسجيل الفيديو هو واحد من أكثر التحديات التي تواجه أي مصور سواء كان هذا المصور محترفاً أو من المصورين المبتدئين.

يهدف هذا المشروع إلى التخلص من الاهتزاز في الفيديو والحفاظ على ثباتية الكاميرا حول محورين وعزلها عن اهتزاز جسم المصور من أجل الحصول على تصوير فيديو سلس خالٍ من الاهتزازات .

لتحقيق هدف هذا المشروع, يوجد محركان للحفاظ على استقرار الكاميرا, كل محرك يحافظ على وضعية الكاميرا حول محور معين, ويوجد حساس مثبت على قاعدة الكاميرا يزود المتحكم بوضعية الكاميرا الحالية.

List of Contents

List of Figures

List of Tables

List of Symbols

symbol	Meaning
Θ_i	Current angle of body (i)
L_i	Length of body i
h_i	Height of body i
b_i	Width of body i
${}^{j}R_i$	Rotation matrix between frame j and frame i
${}^{j}d_i$	Distance between frame j and frame i
m_i	Mass of body i
$I_{ixx}, I_{ixy}, I_{ixz}$ $I_{iyx}, I_{iyy}, I_{iyz}$ $I_{izx}, I_{izy}, I_{izz}$	Mass moment of inertia for body i about its center of mass
ω_i	Angular velocity of body i
Φ	Angular velocity of camera
$(\varphi_x, \varphi_y, \varphi_z)$	Components of angular velocity for the camera
D	Inertial-type matrix
H	Velocity coupling vector
G	Gravitational vector
Q	Reaction torques

Chapter 1
Introduction

The stabilizer is a device that is used to keep the camera stable against the external vibration and disturbances.

1.1) Problem Definition

One of the most common challenges that faces the photographers is taking pictures or video recordings in moving states. Maintaining the stability of the camera within a specific position needs a "professional" photographer. Therefore, to keep the position of the camera fixed "stable" even the body of the photographer vibrates, i.e., helps the photographers in stabilizing the position of the camera, photographers need to use stabilizers for their cameras while recording videos.

Using different types of camera-support arms to increase the stability of the camera is ineffective solution. The following pictures show some difficulties that can face the photographers when using these support-arms while taking pictures.

Figure 1.1: Support Arm

Figure 1.2: Support Arm

Any person can use this device (the stabilizer) to get a clear and smooth record of video, regardless of his vibration of body. It is also not heavy, which means that the photographer can move freely. This also implies that recording a clear video will not need a professional photographer.

As a mechatronics system, the camera must be stabilized automatically and intelligently, so the system needs sensors to act some measurements (angle of rotation, attitude, etc.), a controller with desired driver to drive the motors with suitable torque and synergetic integration between them. Using one motor for each axis makes the camera stable in two-axes. The stabilizer is used

easily, which means that the clear photos and smooth videos will not need to be taken by a professional photographer when this device is used.

1.2) Related Work

Some techno-companies such as Feiyu Tech, Glidecam were provide a solution of the problem mentioned in Section 1.1. By design models and controllers of specific type of cameras to get rid of the vibrations. Some of these products are shown in Figure 1.3. [1]

This project is following the way of the solution which created by these companies, by designing a model and controller to solve the problem.

Figure 1.3: Some Products of Companies (Feiyu Tech, Glidecam)

1.3) Motivation

Any beginner photographer wants to record video confront his lack of skills in recording as a blocker from record or publish his recorded video. So the project team aims to build the stabilizer to help anyone to record his events perfectly with no high skills. Also to provide the product to the local market with reasonable price, the cost of the products which available in the market is up to 2000 $.

1.4) Timeframe Table

(Table 1.1) shows the timeframe of the work and steps that are take place in the first semester. On the other hand, (Table 1.2) shows the timeframe of the work in the second semester.

Table 1.1: Timeframe of the First Semester

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Problem Identification	■	■												
Finding Solution			■	■										
CAD Design					■	■	■							
Kinematic and Dynamic Equations								■	■	■	■	■		
Simulation and Detection of Errors												■	■	■
Specifying Components														■

Table 1.2: Timeframe of the Second Semester

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Purchase of the necessary electronic parts	■	■												
Checking the control system on prototype			■	■	■	■	■	■	■	■	■	■		
Turning and painting the parts										■	■	■		
Assembling and checking the system												■	■	■

1.5) Chapters Overview

This report consists of five chapters including this chapter. The rest of this report is organized as follow:

Chapter 2:
This chapter talks about mechatronics design approach of the stabilizer. It includes recognition of the need, conceptual design and specification, modular solution of the need, sensors and actuators, detailed mathematical model which discuss kinematics with its two branches forward and inverse and discuss the dynamics of the system.

Chapter 3:
This chapter discusses the material, mechanical and electrical component selection for the project.

Chapter 4:
This chapter is talks about software programming and controller design.

Chapter 5:
This chapter explains the project parts and their assembly.

Chapter 6:
This chapter discusses the experimental results of using the stabilizer while recording video.

Chapter 7:
This chapter discusses conclusion and the future work.

Chapter 2
Mechatronics Design

This chapter explains the construction of the stabilizer, detail of its mechanical parts and the dynamic model as well as forward and inverse kinematics of the stabilizer.

2.1) Recognition of the Need

Any beginner photographer suffers from vibration of the camera which affect on the record of video. This is usually due to the fact that the photographer cannot modify stand of the camera vibration and sudden movements during recording, which affects the quality of the recorded video negatively. Thus, it is a fact that any person who uses the camera for video recording hopes to get clear videos easily.

To make the camera stable while taking a photo or recording a video, a device to hold the camera and to allow the photographer to move freely and to get clear photo or video is needed. The previous solution, which is using the camera support arms, is difficult to be applied, these arms limit the photographer's movement.

So the need is to compensate against the noise which appear on the recorded videos.

2.2) Conceptual Design and Functional Specifications

We design a device that consists of three bodies that are connected with each other by motors as shown in Figure 2.1.

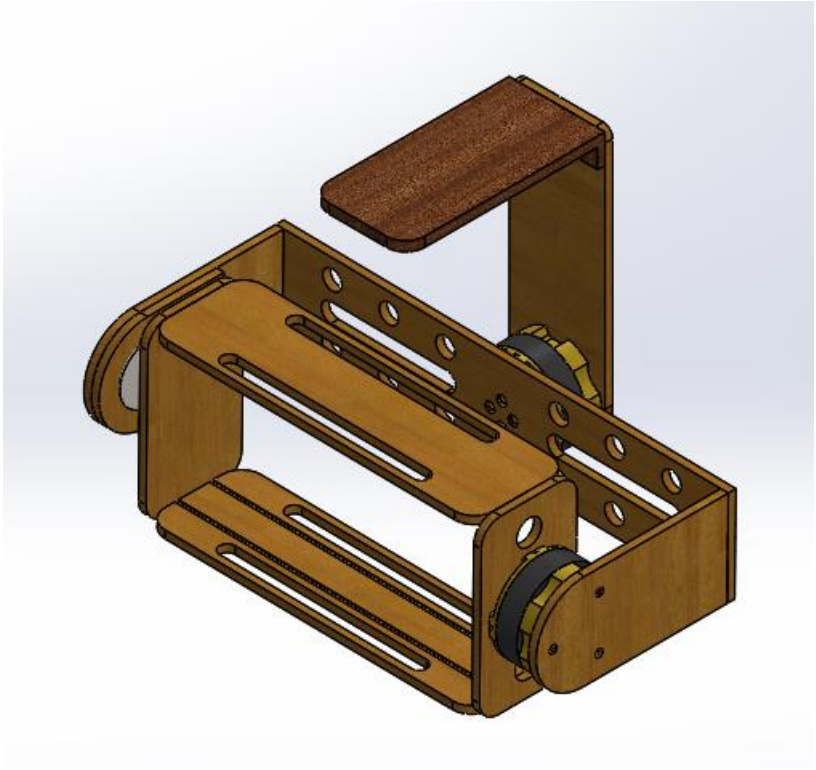


Figure 2.1: The Stabilizer

The main feature of this design is portability, so it should be light. In addition, the energy source should be built in it.

2.3) Modular Mathematical Modeling

The principle of elimination of the noise is described as below:

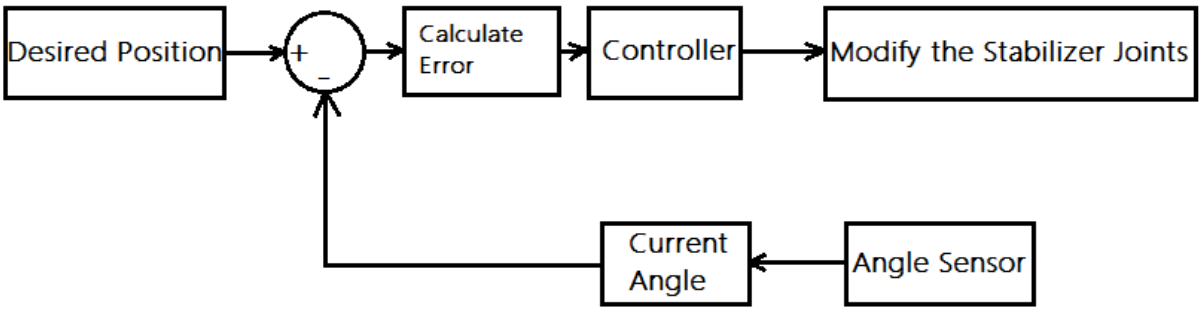


Figure 2.2: Modular Mathematical Model (calculate the error of wrong)

We aim to eliminate of the frequency under 2 Hz. And amplitude range [-10 to +10] cm.

2.4) Sensors and Actuators

This section describes the desired sensors and the actuators to match the stabilizer requirements.

2.4.1) Sensors

Mpu6050 (Accelerometer and gyroscope sensor) is used to measure the angle of the camera in two directions and to give the attitude feedback to the controller about the current position. It is shown in Figure 2.3

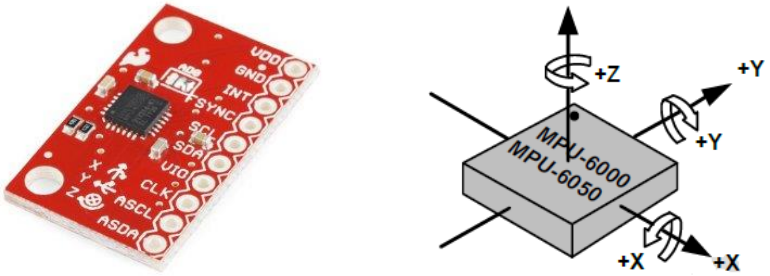


Figure 2.3: Mpu6050 Angle Sensor

2.4.2) Actuators

Since we trend to use light material and the stabilizer design is symmetry, the stabilizer did not need motors with high torque. So we choose three-phase brushless DC motors, namely, (BGM 4108-130). A top view of this motor is shown in Figure 3.2.

Figure 2.4: Three-Phase Brushless DC Motor

This motor is often use for stabilizers. Since it has high speed and high accuracy. The magnetic field of the motor rotates it to reach the needed position, then the same magnetic field locks it from slipping.

The motor specifications are shown in Table2.1. [2]

Table 2.1: Specifications of BGM 4108-130 Three-Phase Brushless DC Motor

Turns	130 turns
Cooper Wire(mm)	0.15
Poles	22
Slots	24
Weight	90g
Motor Size (mm)	Φ46*25
Ri (Ω)	17.0 ohm

2.5) Mathematical Model

This section explains the kinematics and dynamics of our stabilizer.
The stabilizer consists of three parts (part one, part two and part three) as shown in Figures 2.5 and 2.6. These parts are connected with each other by two revolute joints. Each of them is driven by a brushless DC-motor. The mounting point of the camera is located on part three. The box on part one contains the electronic and control circuits.

Part one might be held by hand or fixed during tests.
The direction of the joints is indicated by the double arrows.
The dimensions l_1, l_2, h_1, b_2, h_3 are in Figures 2.5 and 2.6, mass, mass moment of inertia and center of mass for each body are given by the Computer Aided Design (CAD) models of the stabilizer.

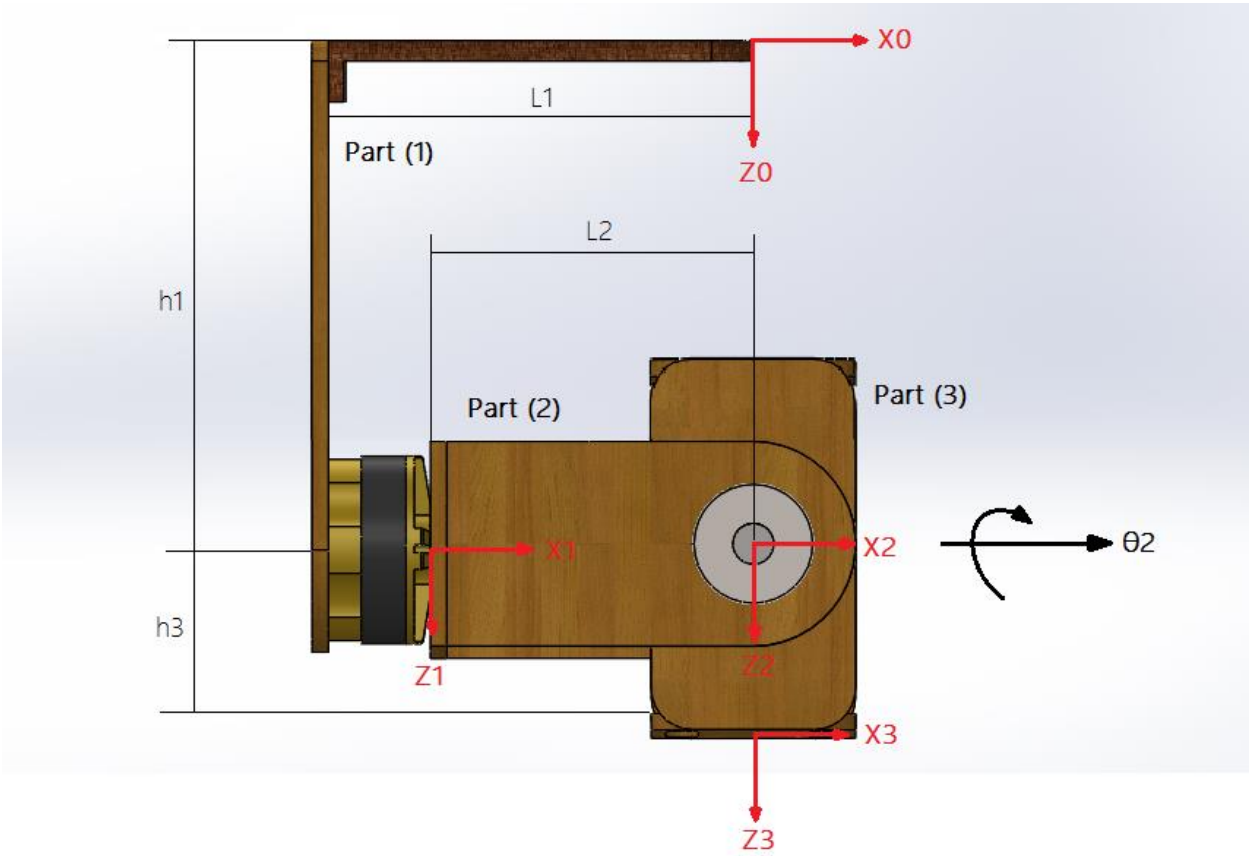


Figure 2.5: Side View of the Stabilizer

l_1=105 mm.
l_2=80 mm.
h_1=125 mm.
b_2=180 mm.
h_3=40 mm.

And Table 2.2 shows the stabilizer properties which taken from SolidWorks CAD model.

Table 2.2: The stabilizer properties which taken from SolidWorks CAD model.

Part / Property	Unit	Part one	Part two	Part three
Mass	[gram]	56	82.2	86.5
Mass moment of Inertia about x-axis (I_x)	[gram*mm^2]	(0.00, 0.86, 0.51)	(1.00, -0.01, -0.04)	(1.00, -0.01, 0.00)
Mass moment of Inertia about y-axis (I_y)	[gram*mm^2]	(0.00, -0.51, 0.86)	(-0.04, 0.00, -1.00)	(0.01, 1.00, -0.00)
Mass moment of Inertia about z-axis (I_z)	[gram*mm^2]	(1.00, 0.00, 0.00)	(0.01, 1.00, -0.00)	(0.00, 0.00, 1.00)
Center of mass along x-axis	[mm]	-25	-34.12	-26.23
Center of mass along y-axis	[mm]	-38.48	-124.89	-124.98
Center of mass along z-axis	[mm]	26.77	58.44	108

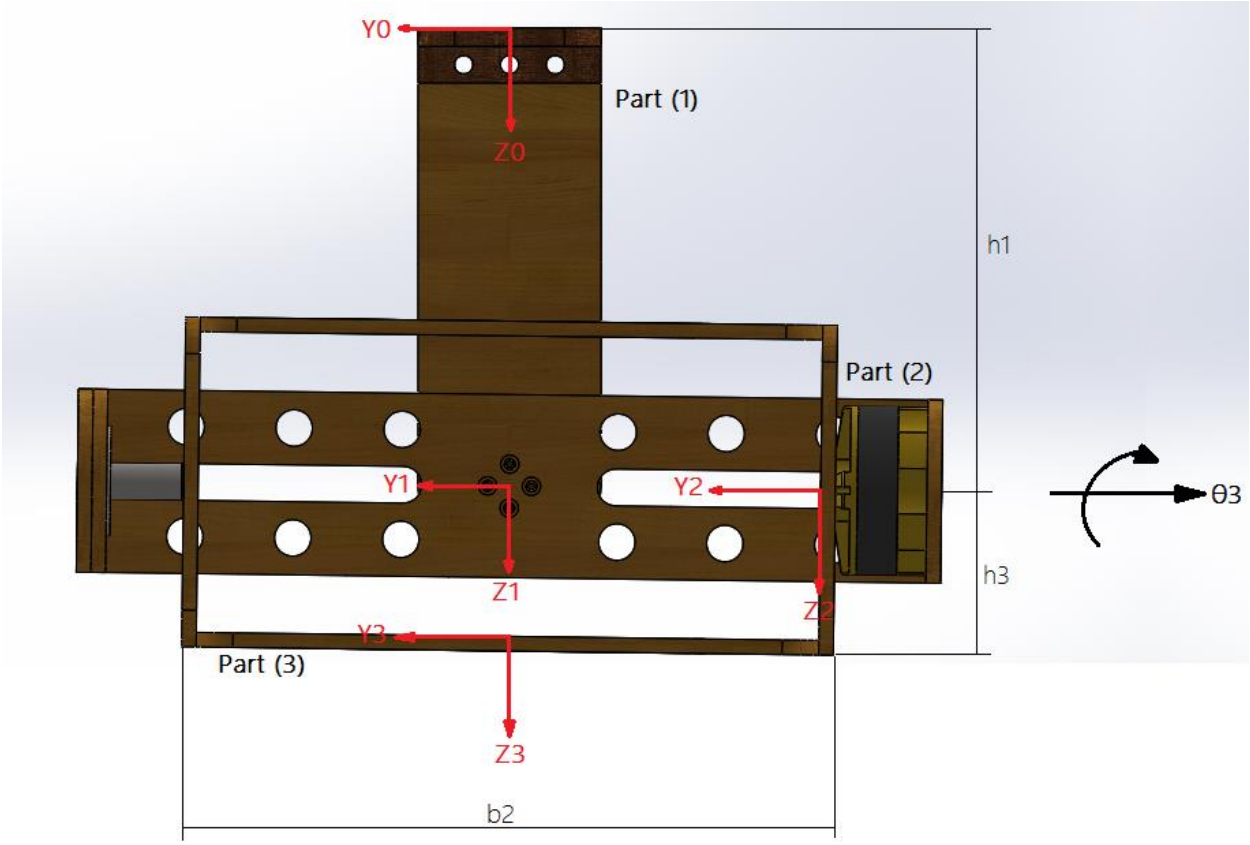


Figure 2.6: Front View of the Stabilizer

2.5.1) Stabilizer's Kinematics

The kinematics is the description of the motion system regardless of the force that acts to the system. We used the kinematics in the derivation of the system dynamics (see Section 2.5.2).

The two-axes camera stabilizer (2-ACS) (see Figures 2.5 and 2.6) consists only of two revolute joints (R-joints). Each revolute joint has one degree of freedom (DOF). This gives a two DOF, roll and pitch (θ_2, θ_3), respectively.

The kinematics can be divided into forward and inverse kinematics as shown below: - [3]

2.5.1.1) Forward Kinematics

Forward kinematics of a Multi-Body System (MBS) is used to find the position and the velocity of the end effector due to change in joints angles.

The 2-ACS is classified as serial robot. The camera is the end effector. Serial robots are often described by using the Denavit-Hartenberg method (D-H method). This method places the coordinate systems of each joint and use a set of rules to find forward kinematics of the MBS. Each frame is described by double arrow on it with the frame number. [4]

Two-Axes transformation matrices

By using the parameters of the stabilizer's parts l_1, l_2, h_1, h_3 and b_2 in Figures 2.5 and 2.6, the transformation between the bodies is described as follow:

1) There is no rotation between base frame (0) and frame (1) about z-axis, so the rotation matrix Between the base frame (0) and frame (1) ($^0R_1$) that is shown in Figures 2.5 and 2.6 is [5]

$$^0R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

The distance between the coordinate origins of base frame (0) and frame (1), ($^0d_1$), is

$$^0d_1 = \begin{bmatrix} l_1 \\ 0 \\ h_1 \end{bmatrix} \tag{2.2}$$

2) The rotation matrix between frame (1) and frame (2), ($^1R_2$), that is shown in Figures 2.5 and 2.6. is

$$^1R_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_2) & -\sin(\theta_2) \\ 0 & \sin(\theta_2) & \cos(\theta_2) \end{bmatrix} \tag{2.3}$$

The distance between the coordinate origins of base frame (1) and frame (2), ($^1d_2$), is

$$^1d_2 = \begin{bmatrix} l_2 \\ -0.5b_2\cos(\theta_2) \\ -0.5b_2\sin(\theta_2) \end{bmatrix} \tag{2.4}$$

3) The rotation matrix between the frame of body (2) and the frame of body (3), (${}^{2}R_{3}$), is

$$ {}^{2}R_{3} = \begin{bmatrix} \cos(\theta_3) & 0 & \sin(\theta_3) \\ 0 & 1 & 0 \\ -\sin(\theta_3) & 0 & \cos(\theta_3) \end{bmatrix} \tag{2.5} $$

and the distance between the coordinate origins (${}^{2}d_{3}$) is

$$ {}^{2}d_{3} = \begin{bmatrix} h_3 \sin(\theta_3) \\ 0.5b_2 \\ h_3 \cos(\theta_3) \end{bmatrix} \tag{2.6} $$

4) The total transformation between the frame of body (3) and the base frame (0), (${}^{0}T_{3}$), is

$$ {}^{0}T_{3} = \begin{bmatrix} {}^{0}R_{3} & {}^{0}d_{3} \\ 0 & 1 \end{bmatrix} = {}^{0}T_{1}\,{}^{1}T_{2}\,{}^{2}T_{3} \tag{2.7} $$

The rotation matrix between the base frame (0) and the frame of body (3), (${}^{0}R_{3}$), is

$$ {}^{0}R_{3} = {}^{0}R_{1}\,{}^{1}R_{2}\,{}^{2}R_{3} \tag{2.8} $$

$$ {}^{0}R_{3} = \begin{bmatrix} \cos(\theta_3) & 0 & \sin(\theta_3) \\ \sin(\theta_2)\sin(\theta_3) & \cos(\theta_2) & -\cos(\theta_3)\sin(\theta_2) \\ -\cos(\theta_2)\sin(\theta_3) & \sin(\theta_2) & \cos(\theta_2)\cos(\theta_2) \end{bmatrix} \quad . \tag{2.9} $$

And the distance between the origin of the base frame (0) and the origin of the frame of body (3), (${}^{0}d_{3}$) , from the transformation matrix (${}^{0}T_{3}$) is

$$ {}^{0}d_{3} = \begin{bmatrix} l_2 - l_1 + h_3 \sin(\theta_3) \\ -h_3 \cos(\theta_3)\sin(\theta_2) \\ h1 + h_3 \cos(\theta_2)\cos(\theta_3) \end{bmatrix} \tag{2.10} $$

Another transformation is used in the inverse kinematics in Section 2.5.1.2 which is the rotation matrix between body (3) and the ground.

$$ {}^{3}R_{ground} = \begin{bmatrix} \cos(\alpha_y) & 0 & \sin(\alpha_y) \\ \sin(\alpha_x)\sin(\alpha_y) & \cos(\alpha_x) & -\cos(\alpha_y)\sin(\alpha_x) \\ -\cos(\alpha_x)\sin(\alpha_y) & \sin(\alpha_x) & \cos(\alpha_x)\cos(\alpha_x) \end{bmatrix} \tag{2.11} $$

Where (α_x , α_y) are the angles of camera on body (3) which are read from an angle sensor attached on body (3). It is assumed that frame (3) is parallel to the angle sensor frame.

Forward Angular Velocity

The relationship between the local angular velocities (ω_2, ω_3) of the joints and the camera angular velocity (${}^{3}_{3}\phi$) can be expressed using Jacobian-Matrix J as

$$ {}^{3}_{3}\phi = \begin{bmatrix} \varphi_x \\ \varphi_y \\ \varphi_Z \end{bmatrix} = J \begin{bmatrix} \omega_2 \\ \omega_3 \end{bmatrix} \tag{2.12} $$

where (ω_2, ω_3) are the angular velocity of body two and angular velocity of body three respectively.

The Jacobian J can be viewed as the angular velocity directions of the joints presented in the coordinate of camera's frame and because the 2-ACS consists only of R-joints it can be written as

$$ \begin{aligned} J &= [\,{}^{3}\theta_2 \quad {}^{3}\theta_3] = [\,{}^{3}R_2\,{}^{2}\theta_2 \quad {}^{3}\theta_3] \\ &= [\,{}^{2}R_3^{T}\,{}^{2}\theta_2 \quad {}^{3}\theta_3] \end{aligned} \tag{2.13} $$

where

$$ {}^{2}\theta_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.14} $$

$$ {}^{3}\theta_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{2.15} $$

which are the directions of the joints, by substituting Equations (2.5), (2.14) and (2.15) into (2.13), and then sub in Equation (2.12) we get

$$ \begin{bmatrix} \varphi_{roll} \\ \varphi_{pitch} \\ \varphi_{yaw} \end{bmatrix} = \begin{bmatrix} \cos(\theta_3) & 0 & 0 \\ 0 & 1 & 0 \\ \sin(\theta_3) & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_2 \\ \omega_3 \\ 0 \end{bmatrix} \tag{2.16} $$

2.5.1.2) Inverse Kinematics

Inverse kinematics is used to calculate the joints angles when the end effector position is known. Inverse kinematics in the 2-ACS application is used to know the joints angles (θ_2, θ_3), which are explained in Figure 2.5 and Figure 2.6, from the orientation of the camera.

Angular Position

The goal is to change the attitude of the camera relative to the ground. This will result in a control error between the desired attitude and angle sensor current attitude. This can be presented as a new rotation matrix in the reference of the camera frame.

$$ {}^{3}R_{\mathbf{e}} = \begin{bmatrix} \cos(\varepsilon_3) & 0 & \sin(\varepsilon_3) \\ \sin(\varepsilon_2)\sin(\varepsilon_3) & \cos(\varepsilon_2) & -\cos(\varepsilon_3)\sin(\varepsilon_2) \\ -\cos(\varepsilon_2)\sin(\varepsilon_3) & \sin(\varepsilon_2) & \cos(\varepsilon_2)\cos(\varepsilon_2) \end{bmatrix} \tag{2.17} $$

where $\varepsilon_2, \varepsilon_3$ are the errors $(\varepsilon = \alpha_{des} - \alpha)$ of the roll and pitch in the camera reference frame where are (e) denotes an imagined error coordinate frame.

The total rotation matrix between the error frame (e) and the 2-ACS base frame (0) is

$$ {}^{0}R_{\mathbf{e}}(\Theta, \varepsilon) = {}^{0}R_{3}(\Theta)\,{}^{3}R_{\mathbf{e}}(\varepsilon) \tag{2.18} $$

where Θ are the current angles of joints (θ_2, θ_3). A new joint configuration Θ_{new} is chosen such that.

$$ {}^{0}R_{e}(\Theta_{new}) = {}^{0}R_{e}(\Theta, \varepsilon) \tag{2.19} $$

Which means that coordinate frame (3) becomes in the same frame as (e) that means the desired attitude is reached.

Equation (2.19) together with Equation (2.18) give

$$ {}^{0}R_{\mathbf{e}}(\Theta_{new}) = {}^{1}R_{3}(\Theta)\,{}^{3}R_{e}(\varepsilon) \tag{2.20} $$

If the right hand side of (2.20) is expressed as

$$ {}^{1}R_{3}(\Theta)\,{}^{3}R_{\mathbf{e}}(\varepsilon) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.21} $$

r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32} and r_{33} are found in appendix A.

The new joint parameters are: -

$$ \theta_{2new} = \sin^{-1}(r_{32}) \tag{2.22} $$

$$ \theta_{3new} = \tan^{-1}(-r_{31}, r_{33}) \tag{2.23} $$

These angels will be the angular position reference signals for the individual joints.

2.5.2) Dynamics

The dynamics of a 2-ACS describe its motion taking into consideration the effect of external forces, such as gravity and torque from the joint actuators. It is presented as a system of differential equations obtained using either Newton-Euler equations which use Newton's second law of motion, or Lagrange equations [5]. In our project we use Lagrange equations to find the dynamic equations.

In 2-ACS (Figures 2.5 and 2.6) we have three bodies, the first one is the base, the second body rotates about the x-axis (roll), and the last rotate about y-axis (pitch) and the equation of motion is:

$$D(q)\ddot{q} + H(q,\dot{q})\dot{q} + G(q) = Q \tag{2.24}$$

where

D is Inertia matrix.
H is Coriolis matrix.
G is Gravitational matrix.
Q is Torque.
q is the angle.
$\dot{q}$ is the angular velocity.

The dynamic model matrices are found in appendix B.

Chapter 3
Material and Component Selection

This chapter discusses the material selection of the stabilizer parts, mechanical components and electrical components which are added to sensors and actuators that were explained in Section 2.4.

3.1) Material

The project team choose the wood, the cheapest and most abundant material. Although it is easy to be machined to form the parts, it is sensitive to the environmental conditions.

3.2) Electronic Component

The project team choose these components in the control system of the stabilizer to make synergetic integration among them to get the promising function.

3.2.1) Controller

Arduino Mega 2560, shown in Figure 3.1, and its specifications are shown in Table 3.1

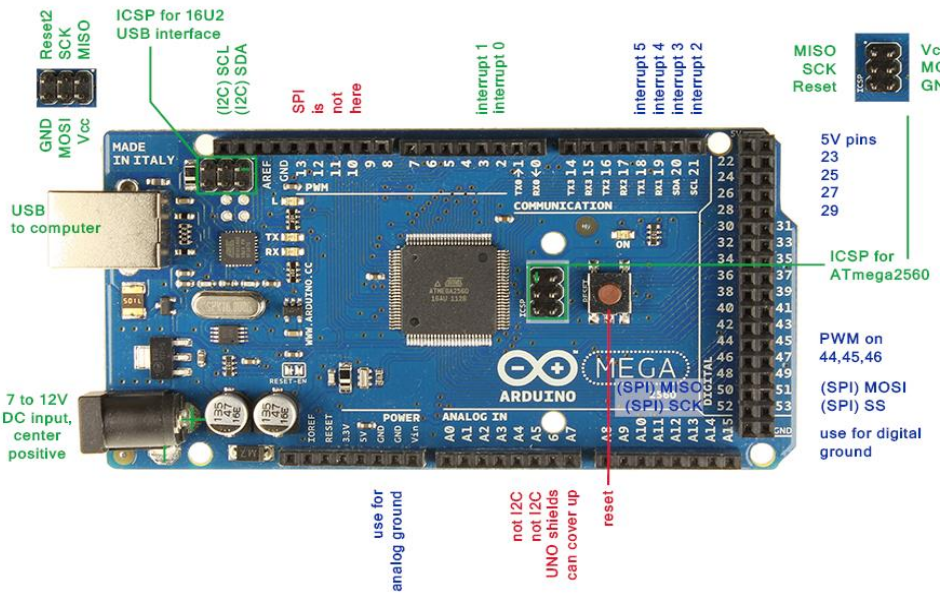


Figure 3.1: Arduino Mega 2560

Table 3.1: Arduino Mega 2560 Specifications [6]

Microcontroller	AT mega 2560
Operating Voltage	5 volts
Input Voltage	7-12 volt
Digital I/O Pins	54 (of which 14 provides PWM)
Analog Input Pins	16
Dc Current Per I/O Pin	40 mA
Dc Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

This controller is available in the market and it contains suitable inputs and outputs to control the stabilizer. Its specifications have encouraged the project team to choose it. The specifications include:

1) Its suitable size and weight to carry on the handle of the stabilizer.
2) It is compatible with Matlab, and programming can be done effectively through it.
3) The power supply source for this controller can be a movable battery, and it is suitable for this mobile project.
4) Open source codes which are available for many applications and can be downloaded from the company's website.
5) The available shields for many components which allow the connection of modules easily.
6) The modularity and the last important thing is that it does not need a dedicated programmer as it can be programmed through the same cable that is used for the PC connection.
7) Other advantages include the upgradability.[6]

3.2.2) Motors Drivers

L298n dual H-bridge driver is used to control the motors speed and direction. It is shown in Figure 3.2

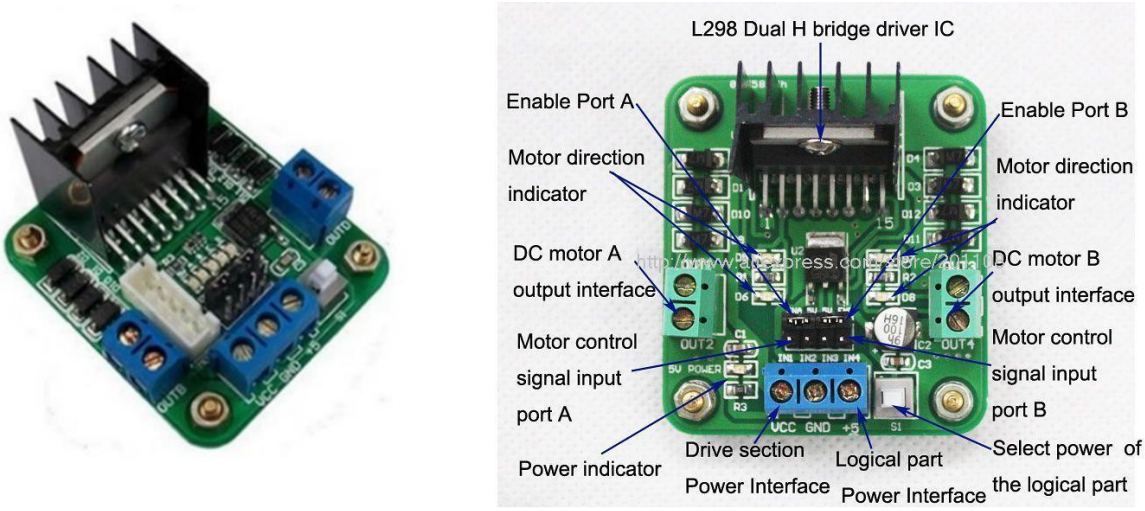


Figure 3.2: L298n Dual H-bridge Driver

Motor driver specifications shown in Table 3.2.

Table 3.2: L298n Dual H-bridge Driver Specifications

Logical voltage: 5V Drive voltage: 5V-35V
Logical current: 0-36 mA Drive current: 2A (MAX single bridge)
Max power: 25W
Dimensions: 43 x 43 x 26 mm
Weight: 26 g

3.2.3) Power Source

The used power source is Lithium recharged batteries, 5v Li-ion Battery Pack because it is suitable for the device specifications and can drive the motors, sensors and control unit.

3.2.4) Joystick

Used from the user to control the angles of the camera. It is shown in Figure 3.3.

Figure 3.3: Joystick

3.3) Mechanical Components

The project team use these mechanical components to accomplish the stabilizer model to achieve its duty.

3.3.1) Connecting Pin

It is used to connect part three of the stabilizer with part two without friction. Its length is determined to make the center of mass of the camera passes through the joints to decrease the torque on the motor and make the system statically stable. It is shown in Figure 3.4.

Figure 3.4: Connecting Pin

3.3.2) Ball Bearing

NSK 6200 ZZCM NS7S: it is used to connect the rod from part (3) of the stabilizer with part (2) with no friction to allow the motor to balance the camera about the x-axis. It is shown in figure 3.5.

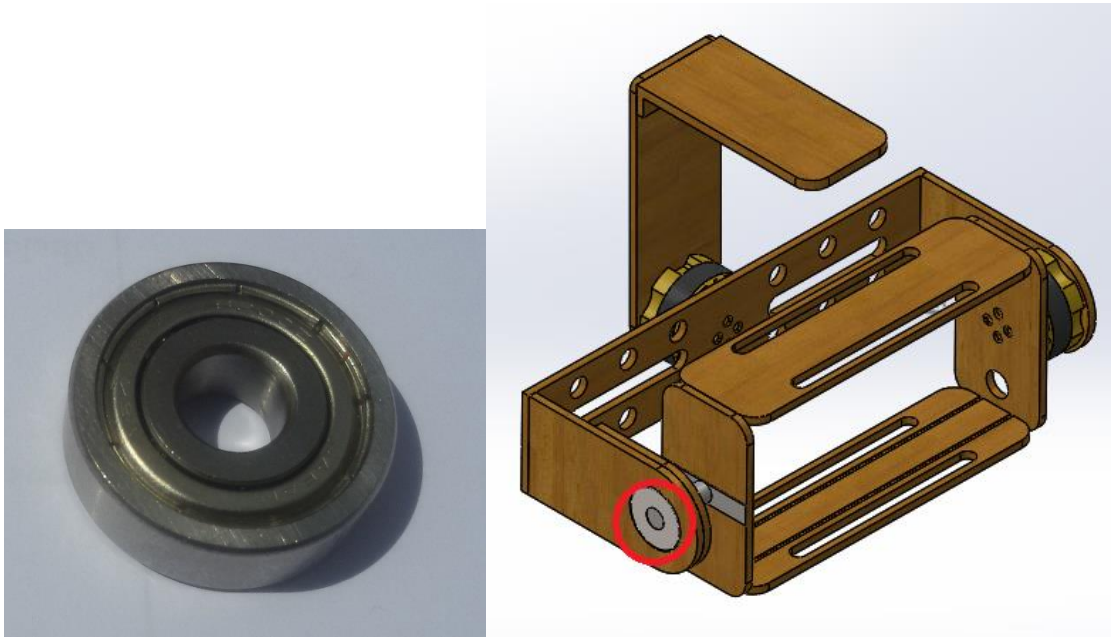
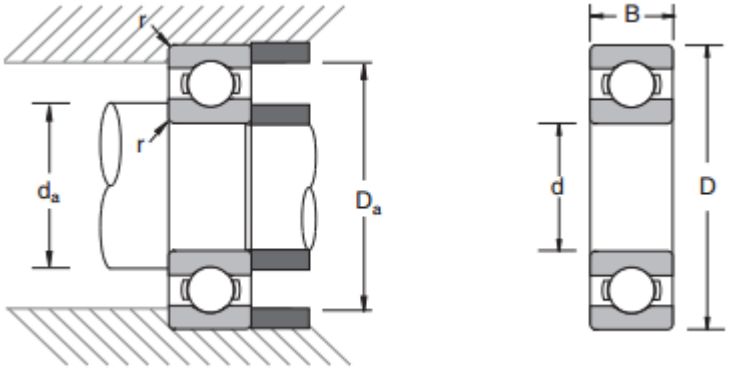


Figure 3.5: NSK 6200 ZZCM NS7S Ball Bearing

Its specifications are shown below in Table 3.3. [7]

Table 3.3: NSK 6200 ZZCM NS7S Ball Bearing Specifications



Bearing Dimension	d [mm]	10
	D [mm]	30
	B [mm]	9
Shoulder Diameters	r [mm]	0.61
	da [mm]	min 0.5
		max 0.63
	Da [mm]	max 0.984
Load Ratings	C_r [lbs.]	1150
	C_{ar} [lbs.]	538
Factor	/	13.2
Limiting Speed (1000) RPM	Grease	24
	Oil	30
Bearing Weight	lbs.	0.07

Chapter 4
Software and Controller Design

Now we will select the suitable controller and prepare the code for microcontroller to achieve the purposes of this project.

4.1) The Basic Work Principle of the System

The project system has two modes, automatic and manual. This Figure shows the operation sequences in every mode.

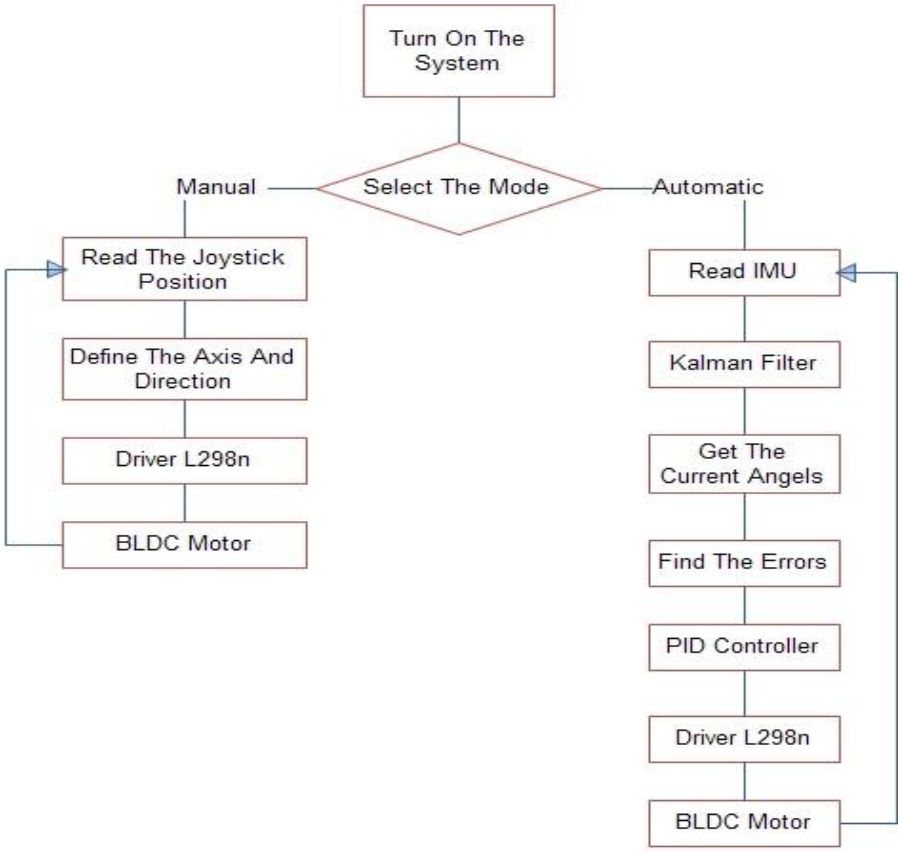


Figure 4.1: Basic work principle of the system

In automatic mode, the controller will maintain the stability of the camera at angles (0, 0) for the roll and pitch angles. But in manual mode, the user can change the position of camera using a joystick. The user can use a switch to turn the system ON/OFF and select the mode.

4.2) Controller Design

We select the Proportional-Integral-Derivative (PID) controller for some reasons: -

1) PID controller depends on three parameters (Kp, Ki, Kd), and we can change the values of these parameters to get a suitable response (settling time, over shoot, steady state error). The output of the controller is expressed as

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (4.1)$$

Where: -

Kp is proportional gain.
Ki is integral gain.
Kd is derivative gain.
t is the time.
u (t) is the controller output.
e (t) is the error signal.

2) We can use this controller for this prototype, by change the value of PID parameters.

3) To use the PID controller we need only to know the effects of PID parameters on the response: -

The effects of increasing PID controller parameters independently are shown in Table 4.1. [8]

Table 4.1: Effects of Increasing PID Controller Parameters Independently

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
Kp	Decrease	Increase	Small change	Decrease	Degrade
Ki	Decrease	Increase	Increase	Eliminate	Degrade
Kd	Minor change	Decrease	Decrease	No effect in theory	Improve if Kd small

We built two PID controllers one for the roll and the other one for the pitch. To find the suitable value for the PID parameters that achieve the specification of the project we use two ways: using the simulating program MAPLE and the experimental trial and error way.

4.2.1) MAPLE Program Simulation

Maple allows the user to draw a model with its physical properties and design a controller to give a response of the system. We use the frequency as an input and check if the controller achieves the specification or not. So we draw the prototype with real dimension and weights and build a PID controller for each motor, but we use a brush DC motor because there is no model for brushless

DC motor and we face a problem to build a model based on mathematical model for brushless DC motor. The schematic diagrams are shown in Figure 4.2 and Figure 4.3

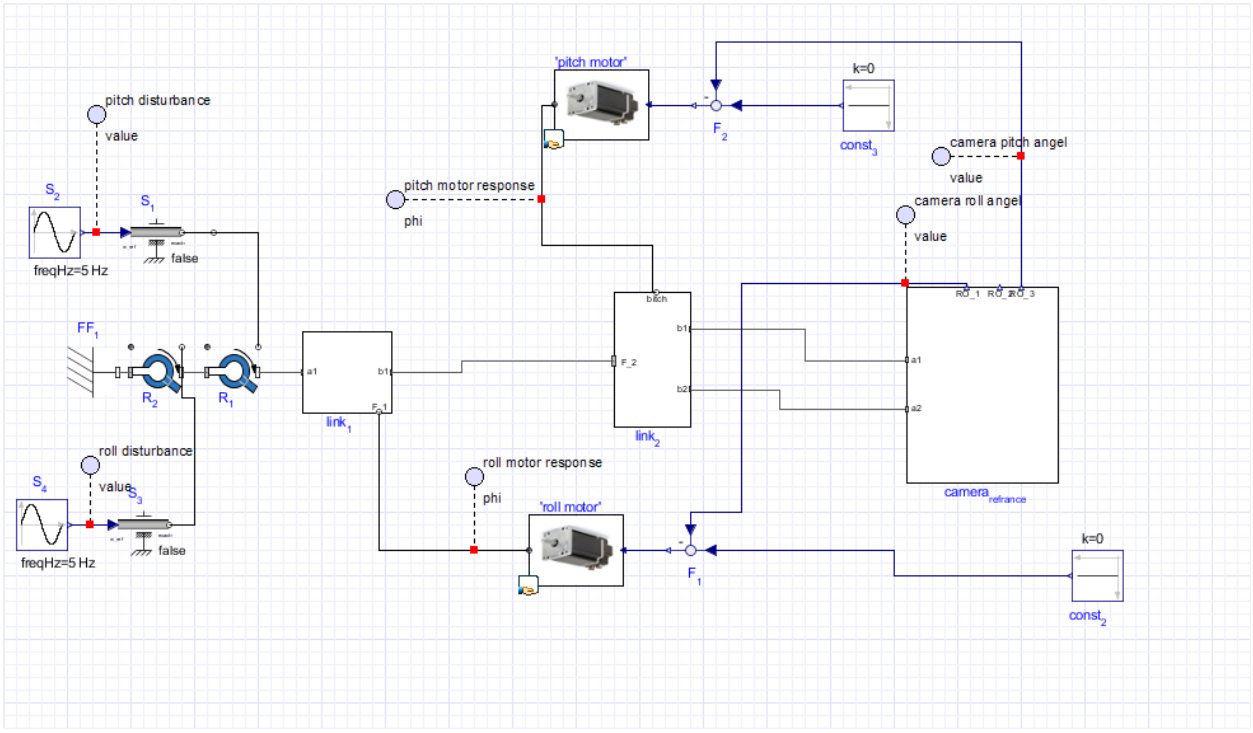


Figure 4.2: Blocks Model of the Stabilizer on Maple

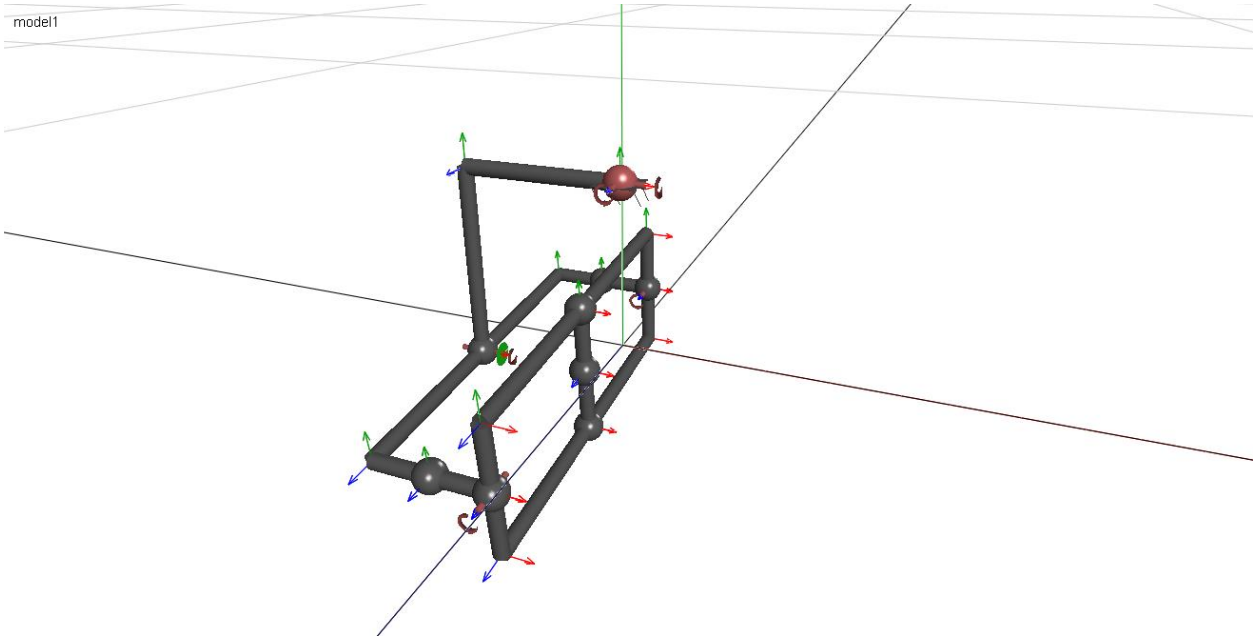


Figure 4.3: 3D Model of the Stabilizer on Maple

.

To find the suitable parameters for the PID controller we add a torque on the base body one as a disturbance in the pitch and the roll directions, the frequency of the torque is 10 Hz and the amplitude is 1.5 radian.

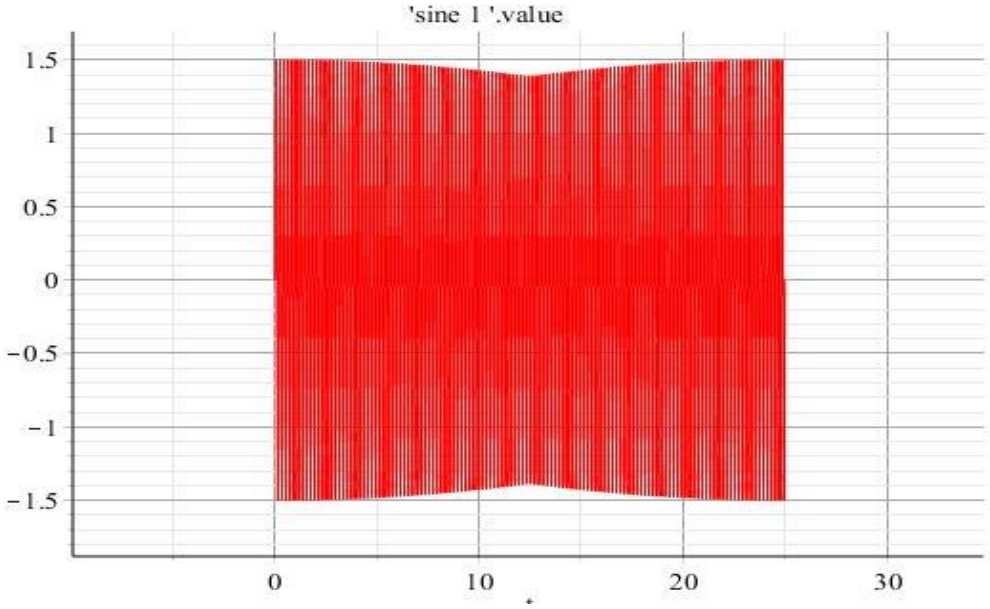


Figure 4.4: Torque Disturbance in Pitch and Roll Direction

Then we change the parameters values of the PID controller to have a suitable response. We get the response in Figures 4.5 and 4.6 when the PID parameters of the PID Roll are (KP=20, KI=2, KD=1) and the parameters of the PID Pitch are (KP=25, KI=4, KD=0.5).

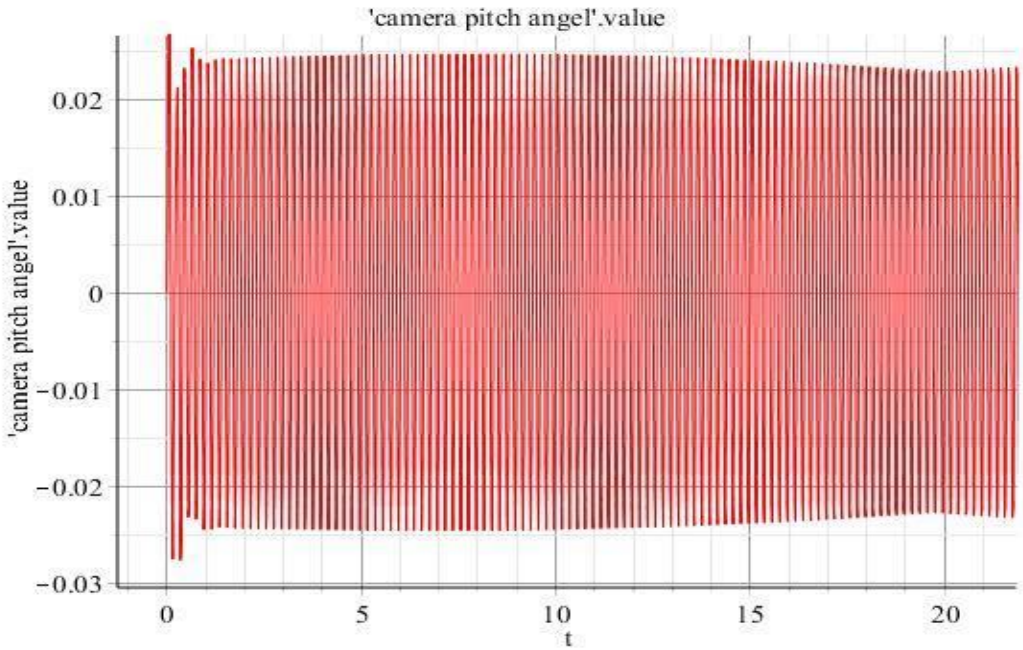


Figure 4.5: Camera Angle Response about Pitch

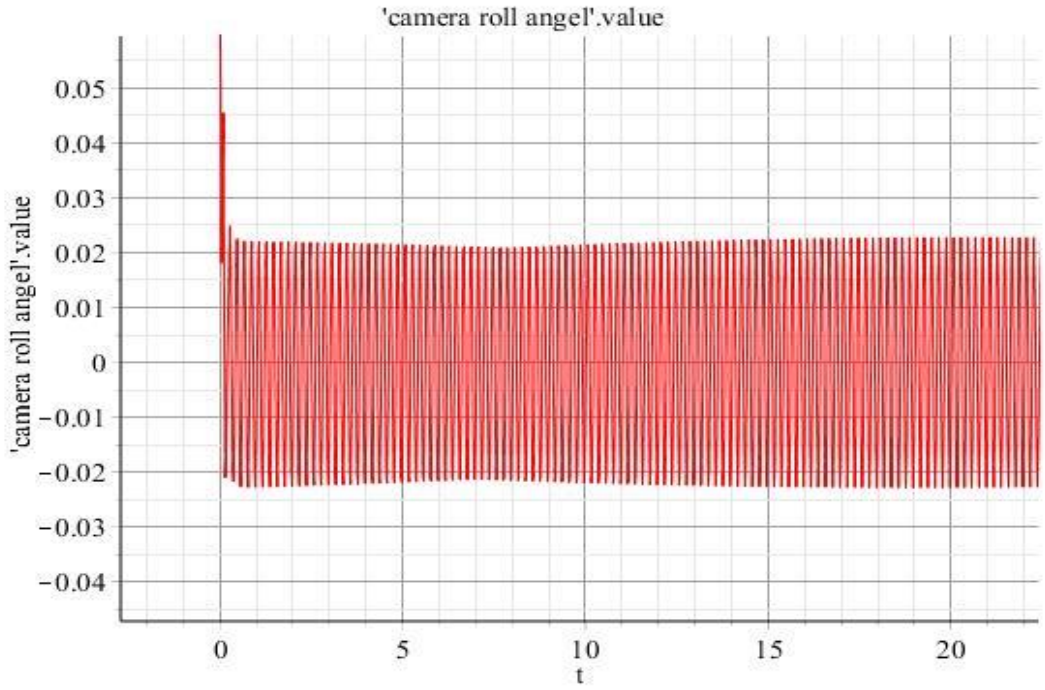


Figure 4.6: Camera Angle Response about Roll

4.2.2) Matlab Simulation

After Simulation on Maple we went to build the model on Matlab. The system is multi-input multi-output (MIMO), so we linearize the system.

Dynamic Model Linearization

The dynamic model is non-linear so we face a problem to design a controller for it, we use a Tylor series to linearize the dynamic model at the operating point.
We use Matlab to linearize the model.

State-Space Model

After linearization of the system, we find the state space model for this sates variables. The state-space model of the system is expressed as follows:

$$\begin{aligned} x_1 &= \theta_2 \\ x_2 &= \dot{\theta}_2 \\ x_3 &= \theta_3 \\ x_4 &= \dot{\theta}_3 \end{aligned} \tag{2.4}$$

Where:
x_1, x_2, x_3 and x_4 are the state variables.
θ_i is the angle of the body i.
$\dot{\theta}_i$ is the angular velocity of the body i.

The state space model of the system is expressed as follows:

$$\dot{x} = A\vec{x} + B\vec{u}$$
$$y = C\vec{x} + D\vec{u} \tag{4.3}$$

Where:
x is the state vector.
y is the output vector.
u is the input vector.
A is the system matrix.
B is the input matrix.
C is the output matrix.
D is the feedforward matrix.

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ \ddot{\theta}_2 \\ x_4 \\ \ddot{\theta}_3 \end{bmatrix} \tag{4.4}$$

Controller design

The state-space model is explaining that the system is a multi-input multi-output. we select a regulator as a controller for the system to achieve the design requirements. And we select the design requirements:

$\xi = 1, \quad f = 2\ Hz$

where:

ξ is the damping ratio.

f is the frequency.

We use Matlab to find the regulator matrix.

Simulink model

The Simulink model of the stabilizer is shown in Figure 4.7.

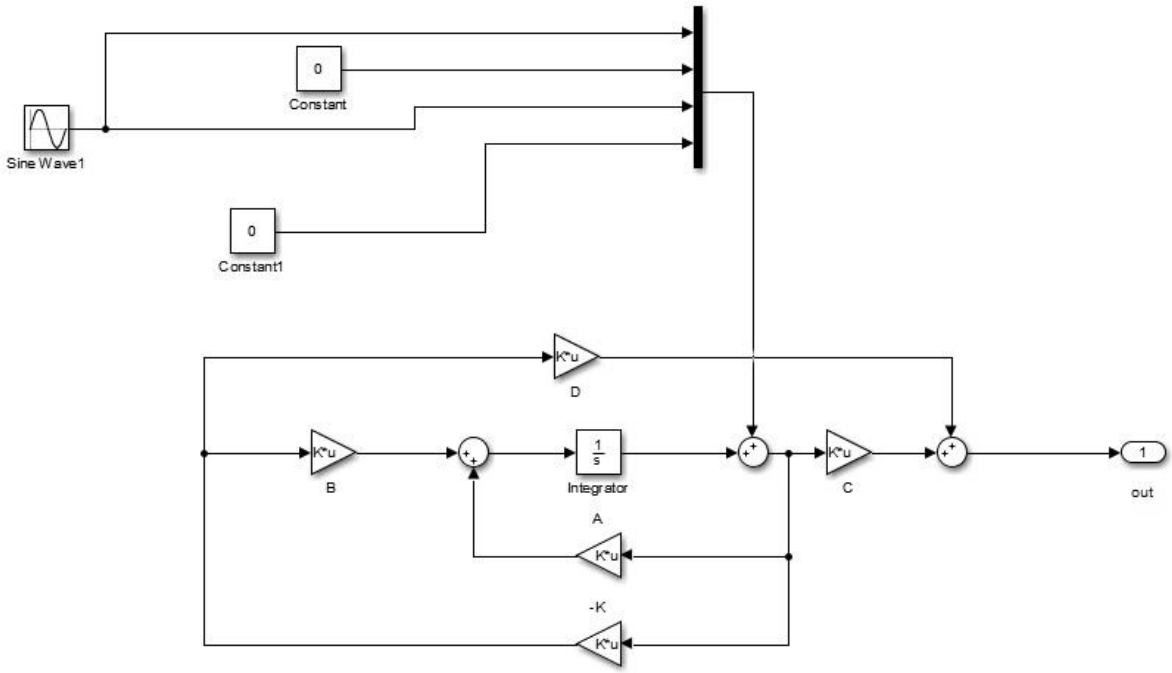


Figure 4.7: The Simulink model of the stabilizer

The disturbance which act on the model in both directions (Pitch and Roll) is shown in Figure 4.8.

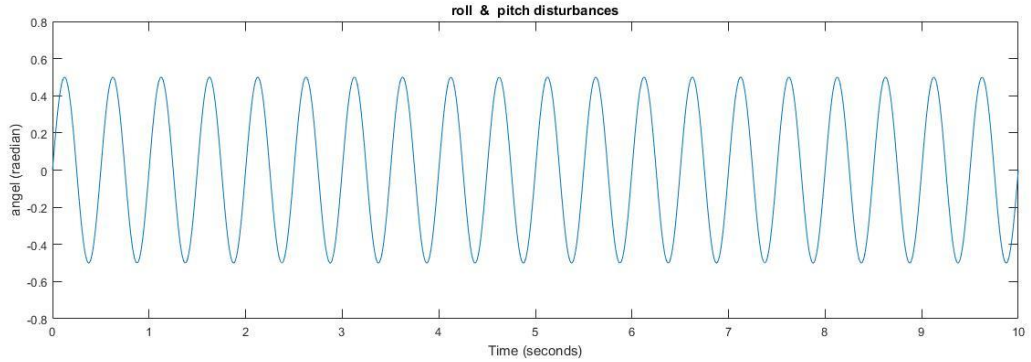


Figure 4.8: The disturbance which act on the model in both directions (Pitch and Roll)

The response of the system about pitch angle is shown in Figure 4.9.

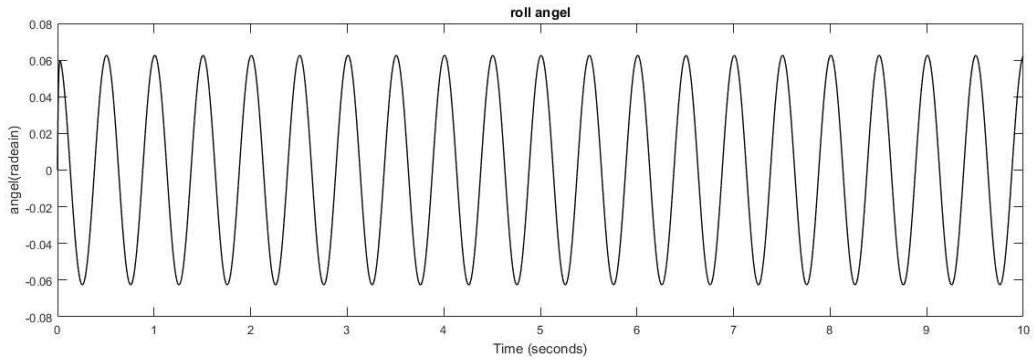


Figure 4.9: The response of the system about pitch angle

And the response about roll angle is shown in Figure 4.10

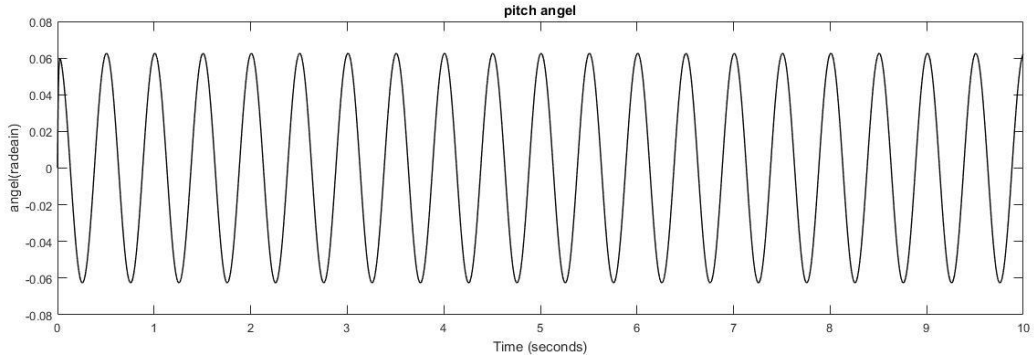


Figure 4.10: The response of the system about pitch angle

Matlab code founded in appendix C.

4.2.3) Experimental Part

In this part we examine the validation of PID parameters, by using the prototype to record a video then watch the video. And depending on video quality we decide any required changes on the controller's parameters by using the information in Table 4.1.

4.3) Brushless DC Motor Control

The project contains two of three-phase Brushless DC Motors (BLDC), to control these motors, the controller gives every phase sine waves and at the same time the phase between waves is 120 degrees as shown in Figure 4.11.

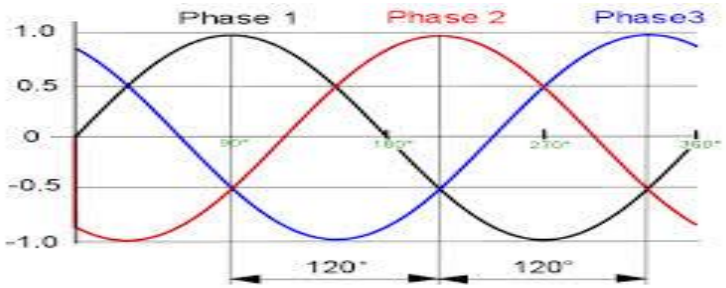


Figure 4.11: Input sine waves to control the BLDC Motor

The used controller (Arduino Mega 2560) has a PWM pins with 8bits timers. The range of output for the PWM is from 0 to 255 level, so we build a sequences by PWM to control the BLDC motor.

The Arduino code is in appendix D.

Chapter 5
Parts of the Stabilizer and Assembly

5.1) Parts

This section explains the parts of the stabilizer and their dimensions. All parts are shaped from wood plate of 5mm thickness.

Note: all units which used in dimensions are in millimeter (mm).

1) Base and Cup of Camera Holder

This part is the base which hold the camera, also it is the cup of the camera to make the system symmetrical. So the stabilizer contains two pieces of this part, the slots which are shown in the part are designed to add the camera fixture easily It is shown in Figure 5.1.

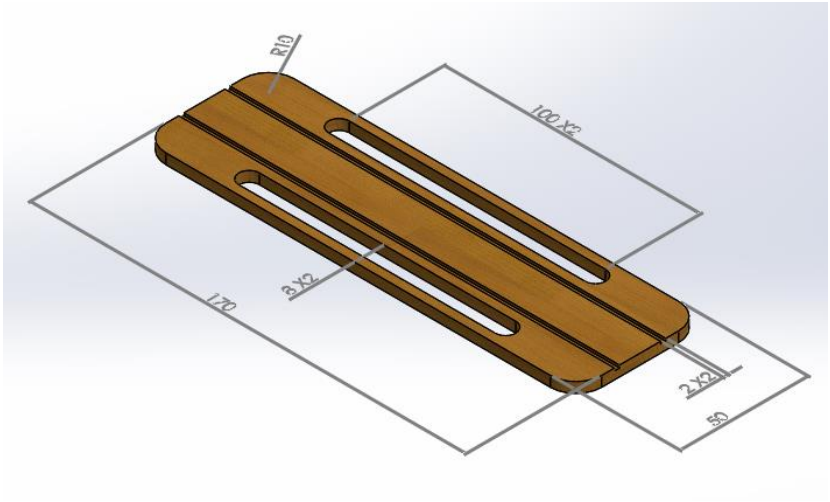


Figure 5.1: Base and Cup

2) Left Side of Camera Holder

This part contains four holes of 3mm diameter each for screws to fix the motor with the part, and two symmetrical holes to decrease the mass of the stabilizer. It is shown in Figure 5.2.

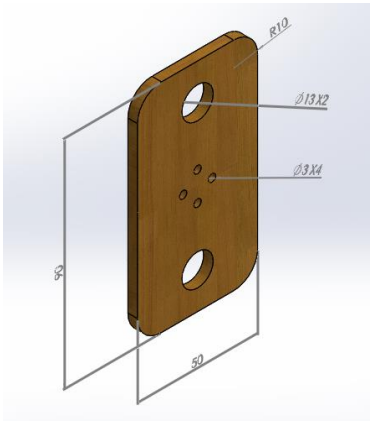


Figure 5.2: Left Side of Camera Holder

3) Right Side of Camera Holder

This part has a connecting pin to support the camera holder by connecting the pin with a ball bearing in right support of the camera holder. It is shown in Figure 5.3.

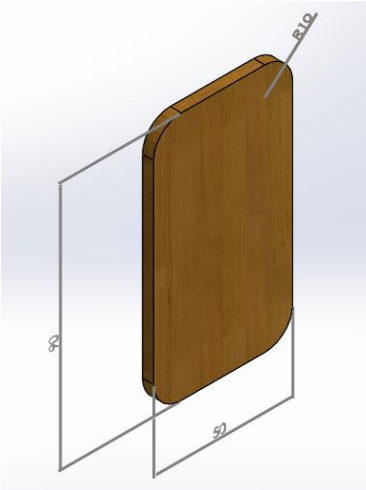


Figure 5.3: Right Side of Camera Holder

4) Right Support of the Camera Holder

This part is used as a support of camera holder. It connects the camera holder with the main holder. It contains a hole for the ball bearing. It is shown in Figure 5.4.

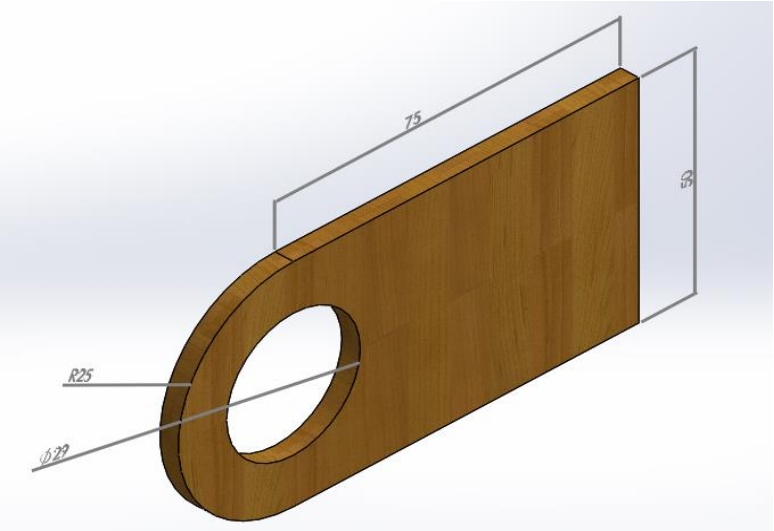


Figure 5.4: Right Support of the Camera Holder

5) Assistant Part for Right Support of the Camera Holder

Because the width of the ball bearing is 10 mm and the stabilizer is made of wood of 5mm width, this part is added to assist the right support of the camera holder to contain the ball bearing efficiently. It is shown in Figure 5.5.

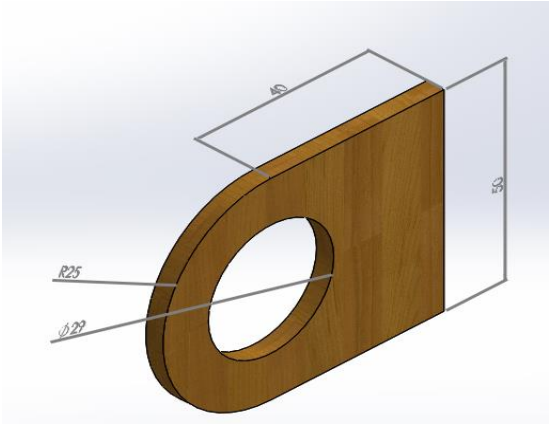


Figure 5.5: Assistant Part for Right Support of the Camera Holder

6) Left Support of the Camera Holder

This part holds the motor which stabilizes the camera about Y-axis. It contains three holes of 2mm diameter each for screws which are used to fix the motor and one pocket of 8mm diameter to prevent the motor shaft from lock. It is shown in Figure 5.6.

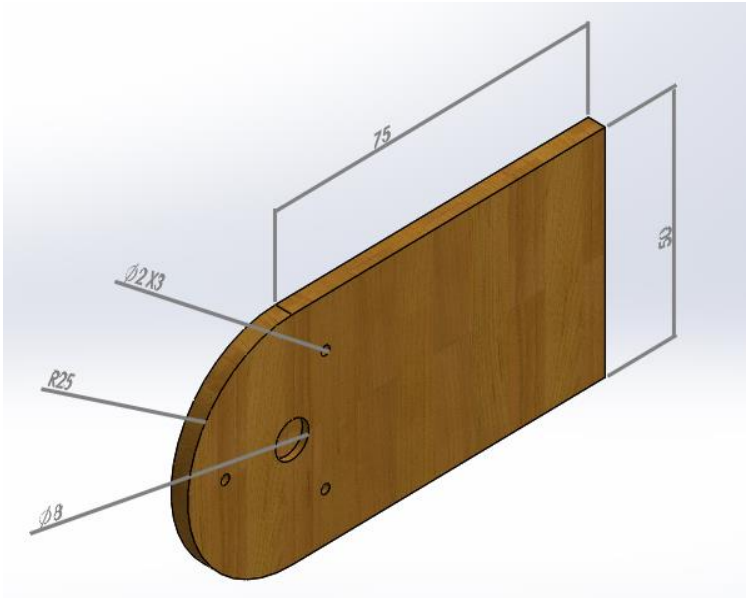


Figure 5.6: Left Support of the Camera Holder

7) Primary Holder

This part holds the right and left supports of the camera holder. It is connected with the motor to stabilize the camera about X-axis.as shown in Figure 5.7. The four holes of 3mm diameter are for the screws to fix the motor, the other twelve holes of 10mm diameter and the slots are just to remove mass from stabilizer while maintaining the model symmetry.

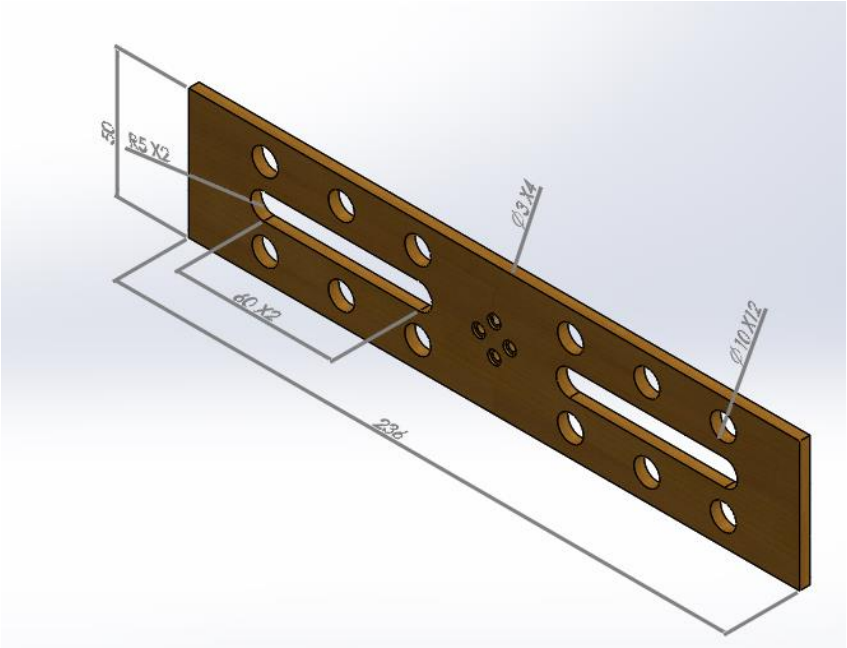


Figure 5.7: Primary Holder

8) Main Holder

This part holds the motor which stabilizes the camera about X-axis, also it holds the box which contains the control card and drivers. It contains three holes of 2mm diameter for screws of the motor. The other side contains the same pocket with 8mm diameter to prevent the motor shaft from lock. And it contains three holes of 5mm diameter to fix it with the stabilizer handle with screws .it is shown in Figure 5.8.

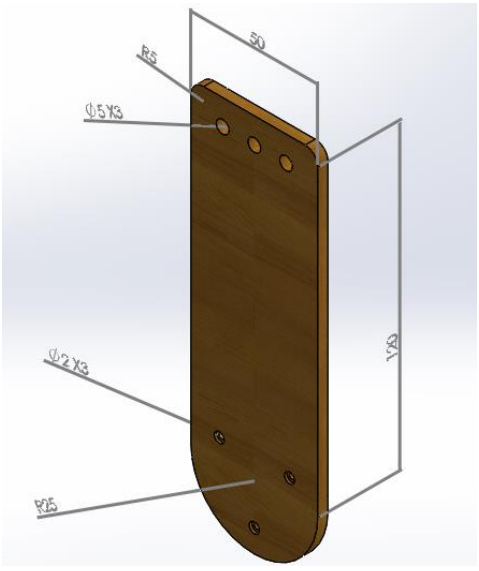


Figure 5.8: Main Holder

9) Handle of the Stabilizer

This part is used to handle the stabilizer, also it holds the joystick on it. it contains three holes of 5mm diameter to connect it with the main holder with screws. It is shown in Figure 5.9.

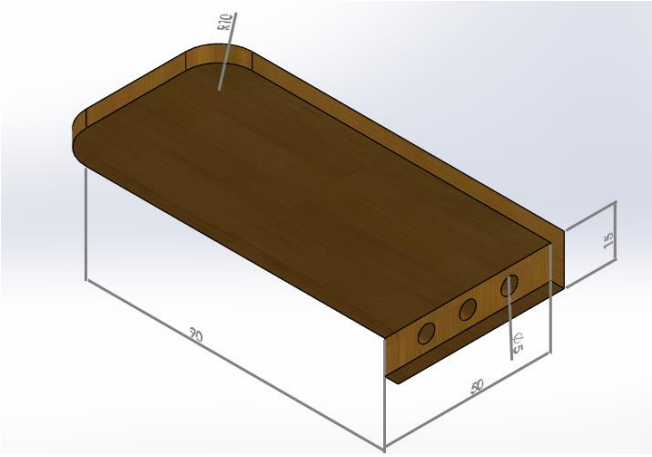


Figure 5.9: Handle of the Stabilizer

5.2) Assembly

This section explains the assembly of the part of the stabilizer.

A) Assembly Catalog of the Stabilizer

The Figure 5.10 explains the exploded view of the stabilizer.

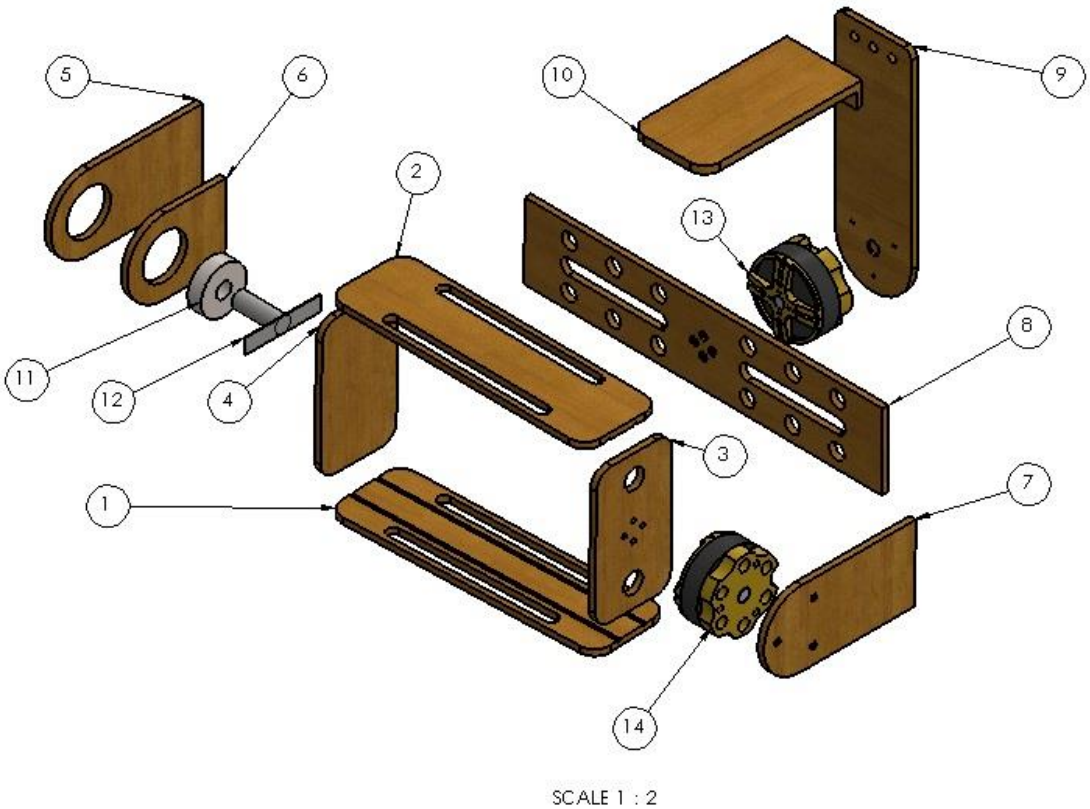


Figure 5.10: Exploded View of the Stabilizer

The numbered parts which are shown in Figure 5.10 are explained in Table 5.1.

Table 5.1: The Stabilizer Parts in Exploded View

1)	Base of Camera Holder
2)	Cup of Camera Holder
3)	Left Side of Camera Holder
4)	Right Side of Camera Holder
5)	Right Support of the Camera Holder
6)	Assistant Part for Right Support of the Camera Holder
7)	Left Support of the Camera Holder
8)	Primary Holder
9)	Main Holder
10)	Handle of the Stabilizer
11)	NSK 6200 ZZCM NS7S Ball Bearing
12)	Connecting Pin
13)	BGM 4108-130 - 3-phase brushless DC motors - Roll
14)	BGM 4108-130 - 3-phase brushless DC motors - Pitch

B) Camera Holder

This assembled part consists of base, cup, left side and right side of camera holder. After it is assembled practically, it is shown in Figure 5.11.

Figure 5.11: Camera Holder

C) Primary Holder with Supports

This assembled part consists of right and left supports of the camera holder, assistant part for right support of the camera holder and the primary holder. After it is assembled practically, it is shown in Figure 5.12.

Figure 5.12: Primary Holder with Supports

D) The Stabilizer

This is the stabilizer after it has been assembled practically. It is shown in Figure 5.13.

Figure 5.13: The Stabilizer After it had been Assembled Practically

E) Wiring Diagram

Figure 5.14 shows the wiring diagram between the electronic components of the stabilizer.

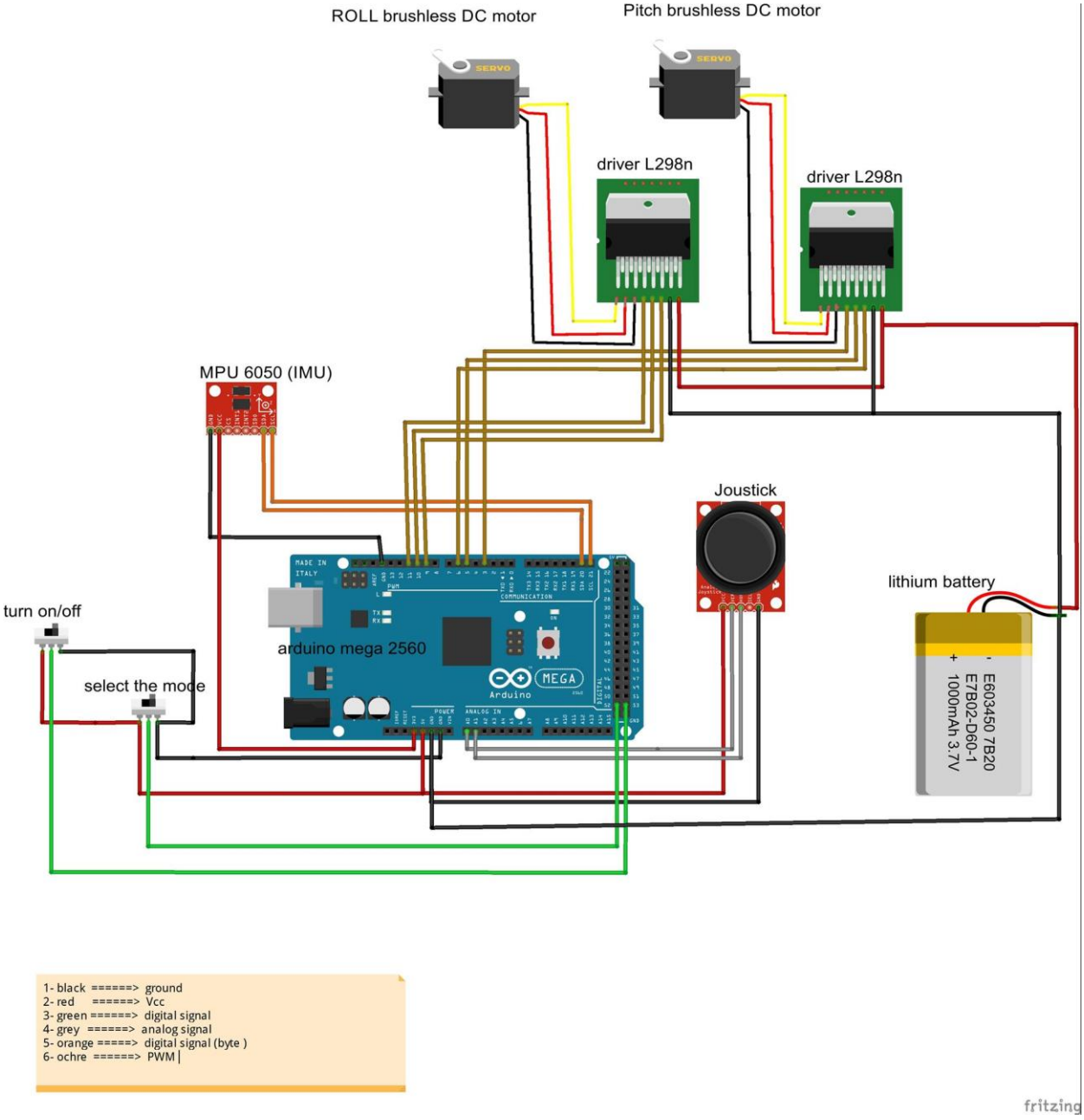


Figure 5.14: Wiring Diagram of the Stabilizer

Chapter 6
Experimental Results

This chapter explains the experimental results which taken from the controller while recording a video.

First we use a kalman filter to get the angels from the sensor, we get the following result.
And the Figure 6.1 explains the error on pitch angle when we use a kalman filter to get the angels.

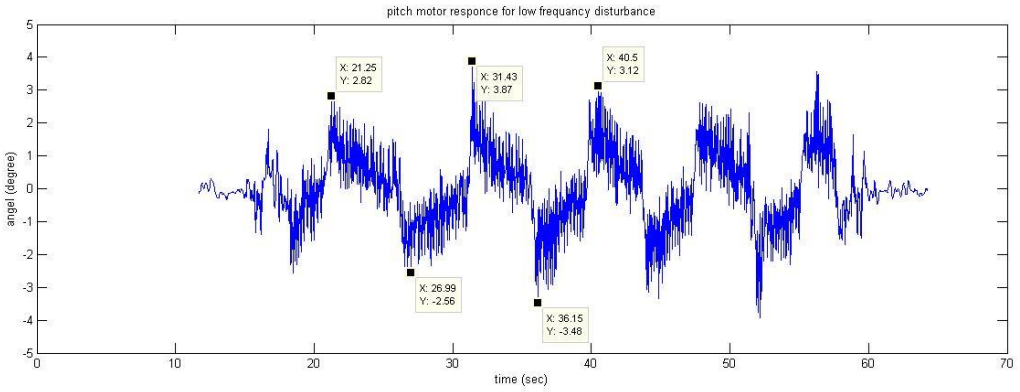


Figure 6.1: The error on pitch angle when we use a kalman filter to get the angels.

We get a right angels but when we move the angels come with noise in high frequency so we use a digital motion process algorithm that compatible with the sensor which depending on external interrupt.

After we complete the assembly of the model with the controller and put the parameters of the PID controller to be for roll (KP= 0.2, KI= 0.1, KD= 0.05) and for pitch (KP= 1, KI= 0.5, KD= 0.01), then we check the stabilizer by walking in a flat way.
We have these results.

And the Figure 6.2 explains the error on pitch angle while recording the video.

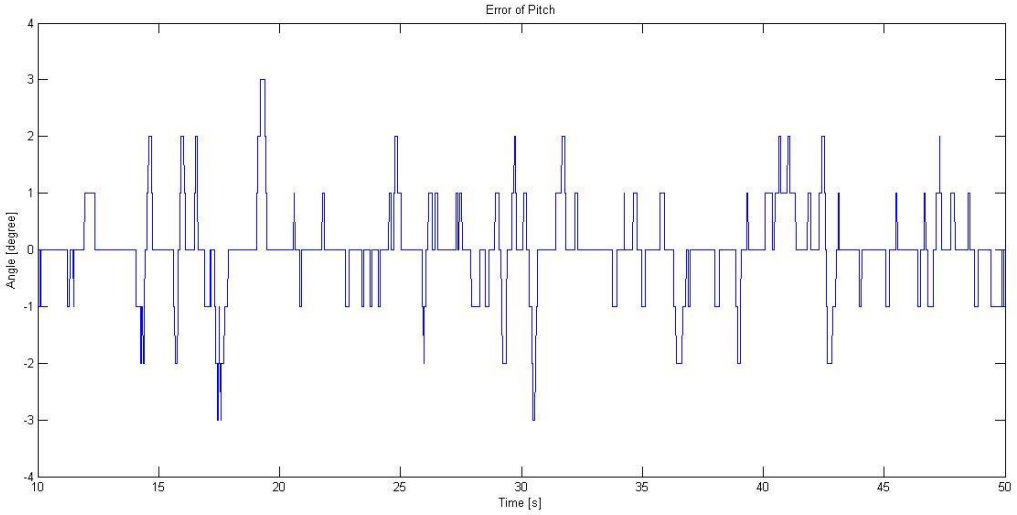


Figure 6.2 The error on pitch angle while recording the video.

The Figure 6.3 explains the output of the controller on pitch angle while recording the video.

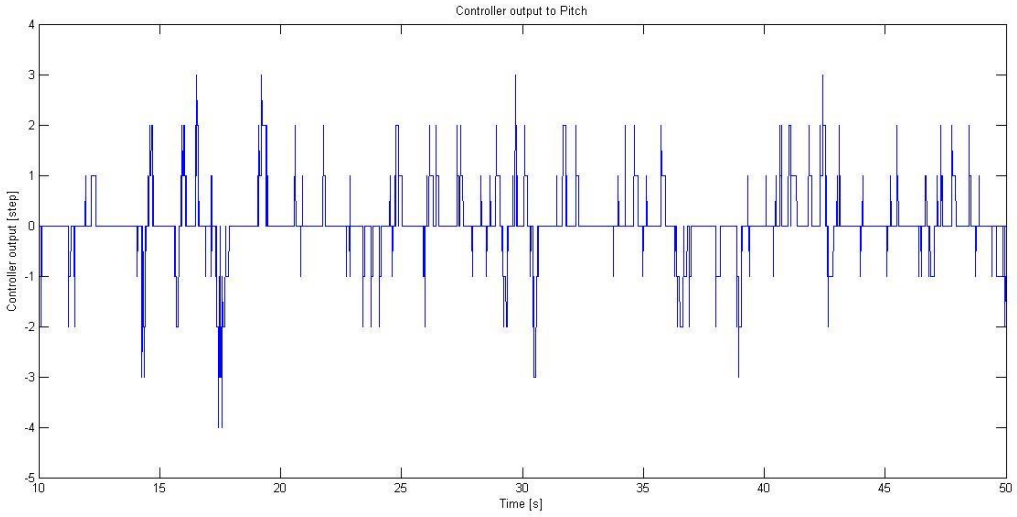


Figure 6.3: The controller output on pitch angle while recording the video.

And the Figure 6.4 explains the error on roll while recording the video.

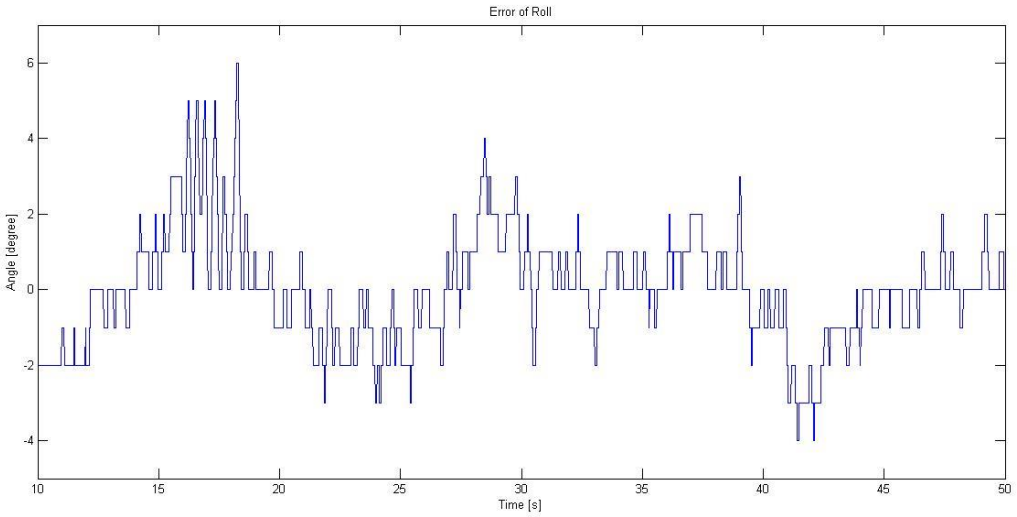


Figure 6.4: The error on the roll angle while recording the video.

The Figure 6.5 explains controller output to the roll motor while recording the video.

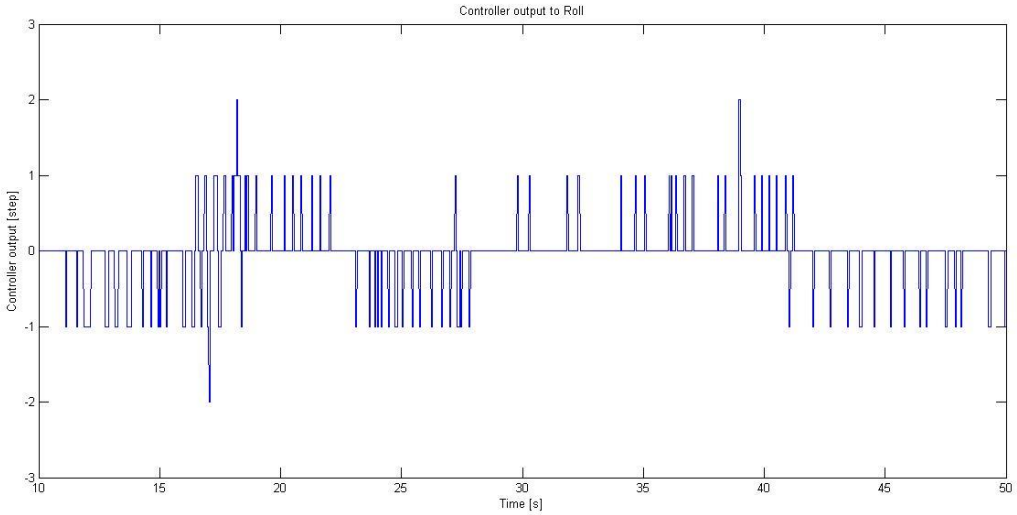


Figure 6.5: The controller output on roll while recording the video.

Chapter 7
Conclusion and Future Work

7.1) Conclusion

The project aims to build and provide the market with a local product of the stabilizer, with lower cost and good performance in compare with universal product.

The stabilizer had been built with PID controller, and we make experiments for the stabilizer and the results are good but it is lower than our expectations.

7.2) Future Work

1) Develop the methodology of controller programming.
2) Develop the project to be three-axes stabilizer.

Appendixes

Appendix A: The total rotation matrix between the error frame (e) and the 2-ACS base frame (0).

$$ {}^{1}R_{3}\left(\Theta\right){}^{3}R_{\mathbf{e}}\left(\varepsilon\right)=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

$$ \boldsymbol{r}_{11}=\cos(\boldsymbol{\varepsilon}_{3})\cos(\theta_{3})-\cos\left(\boldsymbol{\varepsilon}_{2}\right)\sin\left(\boldsymbol{\varepsilon}_{3}\right)\sin(\theta_{3}) $$

$$ \boldsymbol{r}_{12}=\sin\left(\boldsymbol{\varepsilon}_{2}\right)\sin(\theta_{3}) $$

$$ \boldsymbol{r}_{13}=\sin(\boldsymbol{\varepsilon}_{3})\cos(\theta_{3})+\cos\left(\boldsymbol{\varepsilon}_{2}\right)\cos\left(\boldsymbol{\varepsilon}_{3}\right)\sin(\theta_{3}) $$

$$ \boldsymbol{r}_{21}=\cos\left(\theta_{2}\right)\sin\left(\boldsymbol{\varepsilon}_{2}\right)\sin\left(\boldsymbol{\varepsilon}_{3}\right)+\cos\left(\boldsymbol{\varepsilon}_{3}\right)\sin\left(\theta_{2}\right)\sin(\theta_{3})+\cos\left(\boldsymbol{\varepsilon}_{2}\right)\cos\left(\theta_{3}\right)\sin\left(\boldsymbol{\varepsilon}_{3}\right)\sin\left(\theta_{2}\right) $$

$$ \boldsymbol{r}_{22}=\cos(\boldsymbol{\varepsilon}_{2})\cos(\theta_{2})-\cos\left(\theta_{3}\right)\sin\left(\theta_{2}\right)\sin\left(\boldsymbol{\varepsilon}_{2}\right) $$

$$ \boldsymbol{r}_{23}=\sin\left(\boldsymbol{\varepsilon}_{3}\right)\sin\left(\theta_{2}\right)\sin\left(\theta_{3}\right)-\cos\left(\boldsymbol{\varepsilon}_{3}\right)\cos\left(\theta_{2}\right)\sin(\boldsymbol{\varepsilon}_{2})-\cos\left(\boldsymbol{\varepsilon}_{2}\right)\cos\left(\boldsymbol{\varepsilon}_{3}\right)\cos\left(\theta_{3}\right)\sin\left(\theta_{2}\right) $$

$$ \boldsymbol{r}_{31}=\sin\left(\boldsymbol{\varepsilon}_{2}\right)\sin\left(\boldsymbol{\varepsilon}_{3}\right)\sin\left(\theta_{2}\right)-\cos\left(\boldsymbol{\varepsilon}_{3}\right)\cos\left(\theta_{2}\right)\sin(\theta_{3})-\cos\left(\boldsymbol{\varepsilon}_{2}\right)\cos\left(\theta_{2}\right)\cos\left(\theta_{3}\right)\sin\left(\boldsymbol{\varepsilon}_{3}\right) $$

$$ \boldsymbol{r}_{32}=\cos(\boldsymbol{\varepsilon}_{2})\sin(\theta_{2})+\cos\left(\theta_{2}\right)\cos\left(\theta_{3}\right)\sin\left(\boldsymbol{\varepsilon}_{2}\right) $$

$$ \boldsymbol{r}_{33}=\cos(\varepsilon_{2})\cos(\varepsilon_{3})\cos(\theta_{3})\cos(\theta_{2})-\cos(\theta_{2})\sin(\varepsilon_{2})\sin(\theta_{3})-\cos(\varepsilon_{3})\sin(\varepsilon_{2})\sin(\theta_{2}) $$

Appendix B: Dynamics Matrices

1) D matrix = [D1; D2; D3] (Inertia Matrix)

$$D_1 = [0.5 I_{2yy}\ cos(\theta_2)\ cos(\theta_2) - 0.5\, I_{2zy}\ cos(\theta_2)\ sin(\theta_2) - 0.5 I_{2yz}\ sin(\theta_2)\ cos(\theta_2) + 0.5\, I_{2zz}\ sin(\theta_2)\ sin(\theta_2)\ ,\ 0,\ 0]$$

$$\begin{aligned} D_2 = [0, & 0.25\, m_2\ b_2\ sin(\theta_2)\ b_2\ sin(\theta_2) + 0.25\, m_2\ b_2\ cos(\theta_2)\ b_2\ cos(\theta_2) \\ & + m_3\ h_3\ cos(\theta_3)\ cos(\theta_2)\ h_3\ cos(\theta_3) cos(\theta_2) + m_3\ h_3\ sin(\theta_2) cos(\theta_3)\ h_3\ sin(\theta_2)\ cos(\theta_3) \\ & + 0.5 sin(\theta_3)\ sin(\theta_3)\ I_{3xx} - 0.5 sin(\theta_3)\ sin(\theta_2)\ cos(\theta_3)\ I_{3yx} + 0.5\ sin(\theta_3)\ cos(\theta_3)\ cos(\theta_2)\ I_{3zx} \\ & - 0.5\ sin(\theta_2)\ cos(\theta_3)\ sin(\theta_3)\ I_{3xy} + 0.5 sin(\theta_2)\ cos(\theta_3)\ sin(\theta_2)\ cos(\theta_3)\ I_{3yy} \\ & - 0.5 sin(\theta_2)\ cos(\theta_3)\ cos(\theta_3)\ cos(\theta_2)\ I_{3zy} + 0.5\, I_{3xz}\ cos(\theta_2)\ cos(\theta_3)\ sin(\theta_3) \\ & - 0.5\, I_{3yz}\ cos(\theta_2)\ cos(\theta_3)\ sin(\theta_2)\ cos(\theta_3) + 0.5 I_{3zz}\ cos(\theta_2)\ cos(\theta_3)\ cos(\theta_3)\ cos(\theta_2)\ ,\ 0] \end{aligned}$$

$$\begin{aligned} D_3 = [0, & 0, cos(\theta_3)\ h_3\ m_3\ h_3\ sin(\theta_3)\ sin(\theta_2) \\ & + h_3\ m_3 \begin{pmatrix} sin(\theta_3)\ sin(\theta_2)\ h_3\ sin(\theta_3)\ sin(\theta_2) + sin(\theta_3)\ cos(\theta_2)\ h_3\ sin(\theta_3)\ cos(\theta_2) \\ + cos(\theta_3)\ h_3\ cos(\theta_3) \end{pmatrix}] \end{aligned}$$

2) H vector = [H1; H2; H3] (Coriolis Vector)

$$H1=\begin{bmatrix} -cos\left(\theta_2\right)\ I_{2yy}\ sin\left(\theta_2\right)-0.5cos\left(\theta_2\right)\ cos\left(\theta_2\right)\ I_{2zy} \\ +0.5\ sin\left(\theta_2\right)\ sin\left(\theta_2\right)I_{2zy}+0.5sin\left(\theta_2\right)\ sin\left(\theta_2\right)\ I_{2yz} \\ -0.5\ cos\left(\theta_2\right)\ cos\left(\theta_2\right)\ I_{2yz}+sin\left(\theta_2\right)\ I_{2zz}\ cos\left(\theta_2\right) \end{bmatrix}$$

$$\begin{aligned} H\ 2=[\ & \theta_{2\_dot}\theta_{2\_dot}\ (-0.25\ I_{3yx}\ cos\left(\theta_2\right)\ cos\left(\theta_3\right)\ sin\left(\theta_3\right)-0.25\ I_{3zx}\ cos\left(\theta_3\right)\ sin\left(\theta_2\right)\ sin\left(\theta_3\right) \\ & -0.25\ I_{3xy}\ cos\left(\theta_2\right)\ cos\left(\theta_3\right)\ sin\left(\theta_3\right)+0.5\ I_{3yy}sin\left(\theta_2\right)cos\left(\theta_3\right)cos\left(\theta_3\right)cos\left(\theta_2\right)+0.25\ I_{3zy}cos\left(\theta_3\right)sin\left(\theta_2\right) \\ & -0.25I_{3zy}cos\left(\theta_3\right)cos\left(\theta_3\right)\ cos\left(\theta_2\right)cos\left(\theta_2\right)-0.25\ I_{3zy}cos\left(\theta_3\right)sin\left(\theta_2\right)-0.25I_{3zy}cos\left(\theta_3\right)cos\left(\theta_3\right)\ cos\left(\theta_2\right)\ cos\left(\theta_2\right) \\ & -0.25\ sin\left(\theta_3\right)\ cos\left(\theta_3\right)\ sin\left(\theta_2\right)\ I_{3xz}-0.25cos\left(\theta_2\right)cos\left(\theta_2\right)\ cos\left(\theta_3\right)\ cos\left(\theta_3\right)\ I_{3yz} \\ & +0.25\ sin\left(\theta_2\right)\ sin\left(\theta_2\right)cos\left(\theta_3\right)cos\left(\theta_3\right)I_{3yz}-0.5cos\left(\theta_3\right)cos\left(\theta_3\right)\ cos\left(\theta_2\right)\ I_{3zz}\ sin\left(\theta_2\right)) \\ & +\theta_{3\_dot}\ \theta_{2\_dot}\ (-2\ cos\left(\theta_2\right)cos\left(\theta_2\right)cos\left(\theta_3\right)\ h_3h_3\ m_3\ sin\left(\theta_3\right)-2\ cos\left(\theta_3\right)\ sin\left(\theta_2\right)\ sin\left(\theta_2\right)h_3\ h_3\ m_3\ sin\left(\theta_3\right)+I_{3xx}\ sin\left(\theta_3\right) \\ & +0.5I_{3yx}\ sin\left(\theta_2\right)\ sin\left(\theta_3\right)sin\left(\theta_3\right)-0.5\ I_{3yx}\ sin\left(\theta_2\right)\ cos\left(\theta_3\right)\ cos\left(\theta_3\right)-0.5\ I_{3zx}\ sin\left(\theta_3\right)\ sin\left(\theta_3\right)cos\left(\theta_2\right) \\ & +0.5\ I_{3zx}\ cos\left(\theta_3\right)\ cos\left(\theta_3\right)cos\left(\theta_2\right)+0.5\ I_{3xy}\ sin\left(\theta_2\right)\ sin\left(\theta_3\right)sin\left(\theta_3\right)-0.5\ I_{3xy}\ sin\left(\theta_2\right)\ cos\left(\theta_3\right)cos\left(\theta_3\right) \\ & -I_{3yy}\ sin\left(\theta_2\right)sin\left(\theta_2\right)cos\left(\theta_3\right)\ sin\left(\theta_3\right)+I_{3zy}cos\left(\theta_3\right)\ cos\left(\theta_2\right)\ sin\left(\theta_2\right)\ sin\left(\theta_3\right)+0.5cos\left(\theta_3\right)cos\left(\theta 3\right)\ cos\left(\theta_2\right)\ I_{3xz} \\ & -0.5\ sin\left(\theta_3\right)sin\left(\theta_3\right)cos\left(\theta_2\right)\ I_{3xz}+sin\left(\theta_2\right)cos\left(\theta_3\right)\ cos\left(\theta_2\right)I_{3yz}\ sin\left(\theta_3\right)-cos\left(\theta_3\right)\ cos\left(\theta_2\right)\ cos\left(\theta_2\right)\ I_{3zz}\ sin\left(\theta_3\right))] \end{aligned}$$

$$H3=\begin{bmatrix} \theta_{2\_dot}\ \theta_{2\_dot} & \begin{pmatrix} cos\left(\theta_2\right)cos\left(\theta_2\right)cos\left(\theta_3\right)h_3h_3\ m_3\ sin\left(\theta_3\right)+cos\left(\theta_3\right)sin\left(\theta_2\right)sin\left(\theta_2\right)h_3h_3\ m_3\ sin\left(\theta_3\right) \\ -0.5I_{3xx}sin\left(\theta_3\right)cos\left(\theta_3\right)-0.25I_{3yx}sin\left(\theta_2\right)sin\left(\theta_3\right)sin\left(\theta_3\right)+0.25I_{3yx}sin\left(\theta_2\right)cos\left(\theta_3\right)cos\left(\theta_3\right) \\ +0.25I_{3zx}sin\left(\theta_3\right)sin\left(\theta_3\right)cos\left(\theta_2\right)-0.25I_{3zx}cos\left(\theta_3\right)cos\left(\theta_3\right)cos\left(\theta_2\right)-0.25I_{3xy}sin\left(\theta_2\right)sin\left(\theta_3\right)sin\left(\theta_3\right) \\ +0.25I_{3xy}sin\left(\theta_2\right)cos\left(\theta_3\right)cos\left(\theta_3\right)+0.5I_{3yy}sin\left(\theta_2\right)\ sin\left(\theta_2\right)cos\left(\theta_3\right)sin\left(\theta_3\right)-0.5I\ 3zycos\left(\theta 3\right)cos\left(\theta_2\right)sin\left(\theta_2\right)\ sin\left(\theta_3\right) \\ -0.25cos\left(\theta_3\right)cos\left(\theta_3\right)\ cos\left(\theta_2\right)I_{3xz}+0.25sin\left(\theta_3\right)sin\left(\theta_3\right)\ cos\left(\theta_2\right)I_{3xz}-0.5sin\left(\theta_2\right)cos\left(\theta_3\right)cos\left(\theta_2\right)I_{3yz}\ sin\left(\theta_3\right) \\ +0.5cos\left(\theta_3\right)cos\left(\theta_2\right)cos\left(\theta_2\right)I_{3zz}sin\left(\theta_3\right) \end{pmatrix} \\ +0.5\ \theta_{3\_dot} & \theta_{3\_dot}\left(2\ h_3\ sin\left(\theta_3\right)\ sin\left(\theta_2\right)sin\left(\theta_2\right)cos\left(\theta_3\right)+2\ h_3\ sin\left(\theta_3\right)\ cos\left(\theta_2\right)cos\left(\theta_2\right)\ cos\left(\theta_3\right)-2h_3cos\left(\theta_3\right)sin\left(\theta_3\right)\right)m_3h_3 \end{bmatrix}$$

3) G vector = [G1; G2; G3] (Gravitational Vector)

$$G_1 = 0\ ;$$

$$G_2 = m_2\ \left(-9.81 sin\left(\theta_3\right)\ sin\left(\theta_2\right)\ sin\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right) + 9.81 cos\left(\theta_2\right)\ sin\left(\theta_{2\_imu}\right) - 9.81\right) b_2 sin\left(\theta_2\right)$$
$$-0.5\ m_2 \begin{pmatrix} 9.81 sin\left(\theta_3\right)\ cos\left(\theta_2\right)\ sin\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right) + 9.81 sin\left(\theta_2\right)\ sin\left(\theta_{2\_imu}\right) \\ +9.81 cos\left(\theta_3\right)\ cos\left(\theta_2\right)\ cos\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right) \end{pmatrix} b_2\ cos\left(\theta_2\right)$$

$$G_3 = -m_3\left(\ -9.81 sin\left(\theta_3\right) sin\left(\theta_2\right) sin\left(\theta_{3\_imu}\right) cos\left(\theta_{2\_imu}\right) + 9.81 cos\left(\theta_2\right) sin\left(\theta_{2\_imu}\right) - 9.81\ \right) h_3 cos\left(\theta_3\right) cos\left(\theta_2\right) - sin\left(\theta_2\right)$$
$$-m_3\left(\ 9.81 sin\left(\theta_3\right) cos\left(\theta_2\right) sin\left(\theta_{3\_imu}\right) cos\left(\theta_{2\_imu}\right) + 9.81 sin\left(\theta_2\right) sin\left(\theta_{2\_imu}\right) + 9.81 cos\left(\theta_3\right) cos\left(\theta_2\right) cos\left(\theta_{3\_imu}\right) cos\left(\theta_{2\_imu}\right)\right) h_3 sin\left(\theta_3\right) cos\left(\theta_2\right)$$
$$-m_3\left(\ -9.81 sin\left(\theta_3\right)\ sin\left(\theta_2\right)\ sin\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right) + 9.81 cos\left(\theta_2\right)\ sin\left(\theta_{2\_imu}\right) - 9.81\ \right) h_3\ cos\left(\theta_3\right)\ cos\left(\theta_2\right)$$
$$-m_3\left(\ 9.81 sin\left(\theta_3\right)\ cos\left(\theta_2\right)\ sin\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right) + 9.81 sin\left(\theta_2\right)\ sin\left(\theta_{2\_imu}\right) + 9.81 cos\left(\theta_3\right)\ cos\left(\theta_2\right)\ cos\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right)\right) h_3\ sin\left(\theta_2\right)\ cos\left(\theta_3\right)$$
$$+m_3\ \left(-9.81 cos\left(\theta_3\right)\ sin\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right) + 9.81 sin\left(\theta_3\right)\ cos\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right)\right) h_3 cos\left(\theta_3\right)$$
$$+m_3\ \left(-9.81 sin\left(\theta_3\right) sin\left(\theta_2\right) sin\left(\theta_{3\_imu}\right) cos\left(\theta_{2\_imu}\right) + 9.81 cos\left(\theta_2\right)\ sin\left(\theta_{2\_imu}\right) - 9.81\ \right) h_3 sin\left(\theta_3\right)\ sin\left(\theta_2\right)$$
$$-m_3\ \left(\ 9.81 sin\left(\theta_3\right)\ cos\left(\theta_2\right)\ sin\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right) + 9.81 sin\left(\theta_2\right)\ sin\left(\theta_{2\_imu}\right) + 9.81 cos\left(\theta_3\right)\ cos\left(\theta_2\right)\ cos\left(\theta_{3\_imu}\right)\ cos\left(\theta_{2\_imu}\right)\right)\ h_3\ sin\left(\theta_3\right)\ cos\left(\theta_2\right)$$

Appendix C: Matlab Code

%d1_0: The distance vector between the coordinate origins of base frame (0) and frame (1).

```
function result = d1_0(l1,h1)
result=[l1;0; h1];
```

%d2_0: The distance vector between the coordinate origins of base frame (0) and frame (2).

```
function result=d2_0(l1,l2,b2,h1,theta2);
result=[l2+l1;-0.5 *cos(theta2) *b2;-0.5 *sin(theta2) *b2 + h1];
```

%d2_1: The distance vector between the coordinate origins of frame (1) and frame (2).

```
function result=d2_1(theta2,l2,b2)
result=[l2;-0.5*b2*cos(theta2);-0.5*b2*sin(theta2)];
```

%d3_0: The distance vector between the coordinate origins of base frame (0) and frame (3).

```
function result=d3_0(theta2,theta3,l1,h1,l2,h3)
result=[l2-l1+h3*sin(theta3);-
h3*cos(theta3)*sin(theta2);h1+h3*cos(theta2)*cos(theta3)];
```

%d3_1: The distance vector between the coordinate origins of frame (1) and frame (3).

```
function result=d3_1(theta2,theta3,l2,h3)
result=[l2 + h3*sin(theta3)
;-h3*cos(theta3)*sin(theta2);h3*cos(theta2)*cos(theta3)];
```

%d3_2: The distance vector between the coordinate origins of frame (2) and frame (3).

```
function result=d3_2(theta3,h3 ,b2 )
result=[h3*sin(theta3);0.5*b2;h3*cos(theta3)];
```

%I1:mass moment matrix for body one about center of mass .

```
function result=I1(I1xx,I1xy,I1xz,I1yx,I1yy,I1yz,I1zx,I1zy,I1zz);
result=[I1xx I1xy I1xz;I1yx I1yy I1yz;I1zx I1zy I1zz];
```

%I2: mass moment matrix for body two about center of mass .

```
function result=I2(I2xx,I2xy,I2xz,I2yx,I2yy,I2yz,I2zx,I2zy,I2zz);
result=[I2xx I2xy I2xz;I2yx I2yy I2yz;I2zx I2zy I2zz];
```

%I3: mass moment matrix for body three about center of mass .

```
function result=I3(I3xx,I3xy,I3xz,I3yx,I3yy,I3yz,I3zx,I3zy,I3zz);
result=[I3xx I3xy I3xz;I3yx I3yy I3yz;I3zx I3zy I3zz];
```

%%% jacobian matrices

%J1 for d1_0

```
function result=J1();
result=[0 0 0;0 0 0;0 0 0];
```

%J2 for d2_0

```
function result=J2(theta2,b2);
```

```
result=[0 0 0;0 0.5* sin(theta2)* b2 0;0 -0.5* cos(theta2)* b2 0];
```

%J3 for d3_0

```
function result=J3(theta3,h3,theta2);
result=[0 0 cos(theta3)* h3;0 -cos(theta3) *cos(theta2) *h3  sin(theta2)
*sin(theta3) *h3;0 -sin(theta2) *cos(theta3)* h3  -sin(theta3)* cos(theta2)
*h3]
```

%%% mass moment matrix from the frame zero to bodies frame

%I0_1:

```
function result=I0_1(I1xx,I1xy,I1xz,I1yx,I1yy,I1yz,I1zx,I1zy,I1zz);
result=[I1xx I1xy I1xz;I1yx I1yy I1yz;I1zx I1zy I1zz];
```

%I0_2:

```
function result=I0_2(I2xx,I2xy,I2xz,I2yx,I2yy,I2yz,I2zx,I2zy,I2zz,theta2);
result= [I2xx , I2xy *cos(conj(theta2)) - I2xz* sin(conj(theta2)) , I2xy
*sin(conj(theta2)) + I2xz *cos(conj(theta2)) ; cos(theta2) *I2yx -
sin(theta2)* I2zx ,(cos(theta2)* I2yy - sin(theta2)* I2zy) *cos(conj(theta2))
- (cos(theta2)* I2yz - sin(theta2)* I2zz) *sin(conj(theta2)) , (cos(theta2)*
I2yy - sin(theta2)* I2zy) *sin(conj(theta2)) + (cos(theta2) *I2yz -
sin(theta2) *I2zz) *cos(conj(theta2)); sin(theta2) *I2yx + cos(theta2)* I2zx
,  (sin(theta2)* I2yy + cos(theta2)* I2zy) *cos(conj(theta2)) - (sin(theta2)*
I2yz + cos(theta2) *I2zz) *sin(conj(theta2)) ,  (sin(theta2)* I2yy +
cos(theta2)* I2zy) *sin(conj(theta2))  + (sin(theta2)* I2yz + cos(theta2)
*I2zz)* cos(conj(theta2))]
```

%I0_3:

```
function
result=I0_3(I3xx,I3xy,I3xz,I3yx,I3yy,I3yz,I3zx,I3zy,I3zz,theta2,theta3);
result= [(cos(conj(theta3))* I3xx + z2 *I3yx - z1* I3zx) *cos(theta3)+ z4
*sin(theta3) *sin(theta2) - z3 *sin(theta3) *cos(theta2) , z4 *cos(theta2) +
z3 *sin(theta2) ,(cos(conj(theta3))* I3xx + z2* I3yx - z1 *I3zx)
*sin(theta3)- z4 *sin(theta2) *cos(theta3) + z3 *cos(theta3)* cos(theta2)
;(cos(conj(theta2))* I3yx + sin(conj(theta2))*I3zx)* cos(theta3)+ z6
*sin(theta3)* sin(theta2) - z5 *sin(theta3) *cos(theta2) , z6* cos(theta2) +
z5 *sin(theta2) , (cos(conj(theta2))* I3yx + sin(conj(theta2))* I3zx)
*sin(theta3) - z6* sin(theta2) *cos(theta3) + z5*cos(theta3)
*cos(theta2);(sin(conj(theta3))* I3xx - z8 *I3yx + z7* I3zx) *cos(theta3) +
z10 *sin(theta3) *sin(theta2) - z9 *sin(theta3)* cos(theta2) , z10
*cos(theta2) + z9* sin(theta2) ,(sin(conj(theta3)) *I3xx - z8 *I3yx + z7*
I3zx) *sin(theta3)- z10* sin(theta2) *cos(theta3) + z9*cos(theta3)*
cos(theta2)]
```

%r1: rotation matrix for body one

```
function result=r1();
result=[0 0 0;0 0 0;0 0 0]
```

%r2: rotation matrix for body two

```
function result=r2();
result=[0 0 0;1 0 0;0 0 0]
```

%r3: rotation matrix for body three

```
function result=r3();
result=[0 0 0;0 0 0;0 1 0]
```

%R1_0: rotation matrix for frame one refer to frame zero

```
function result =R1_0();
result=[1 0 0;0 1 0;0 0 1];
```

%R2_1: rotation matrix for frame two refer to frame one

```
function result=R2_1(theta2)
result=[1 0 0;0 cos(theta2) -sin(theta2);0 sin(theta2) cos(theta2)];
```

%R3_2: rotation matrix for frame three refer to frame two

```
function result=R3_2(theta3)
result=[cos(theta3) 0 sin(theta3);0 1 0;-sin(theta3) 0 cos(theta3)];
```

%R3_0: rotation matrix for frame three refer to frame zero

```
function result0=R3_0(theta2,theta3)
result1=[cos(theta3),0,sin(theta3)];
result2=[ sin(theta2)*sin(theta3), cos(theta2), -cos(theta3)*sin(theta2)];
result3=[ -cos(theta2)*sin(theta3), sin(theta2),  cos(theta2)*cos(theta3)];
result0=[result1;result2;result3];
```

%%% dynamic model

%D_matrix : inertia type matrix

```
function result=Dmatrix(theta2,theta3, I2yy,  I2yz  , I2zy , I2zz,  I3xx,
I3xy  ,I3xz , I3yx  , I3yy,  I3yz , I3zx,  I3zy , I3zz, m2, m3,b2,h3 )
t1=(J1()'*m1*J1())+(0.5*r1()'*I0_1(I1xx,I1xy,I1xz,I1yx,I1yy,I1yz,I1zx,I1zy,I1
zz)*r1())
t2=(J2(theta2,b2)'*m2*J2(theta2,b2))+(0.5*r2()'*I0_2(I2xx,I2xy,I2xz,I2yx,I2yy
,I2yz,I2zx,I2zy,I2zz,theta2)*r2())
t3=(J3(theta3,h3,theta2)'*m3*J3(theta3,h3,theta2))+(0.5*r3()'*I0_3(I3xx,I3xy,
I3xz,I3yx,I3yy,I3yz,I3zx,I3zy,I3zz,theta2,theta3)*r3())
result=t1+t2+t3;
```

% D_inverse: inverse matrix for a inertia type matrix

```
function result=invD(theta2,theta3, I2yy,  I2yz  , I2zy , I2zz,  I3xx,  I3xy
,I3xz , I3yx  , I3yy,  I3yz , I3zx,  I3zy , I3zz, m2, m3,b2,h3 );
result1= [1/(0.5* cos(theta2)* cos(theta2)*  I2yy-0.5*sin(theta2)
*cos(theta2)* I2zy-0.5*cos(theta2)* sin(theta2) *I2yz+0.5*sin(theta2)*
sin(theta2) *I2zz),0,0];
result2=[0 ,1/(0.25*sin(theta2)*sin(theta2)*b2*b2*m2+0.25* cos(theta2)*
cos(theta2) * b2 * b2*  m2 + cos(theta2)*cos(theta2)*  cos(theta3) *
cos(theta3)* h3* h3 * m3+cos(theta3) *cos(theta3)* sin(theta2)* sin(theta2) *
h3 * h3  *m3 + 0.5* I3xx *sin(theta3)*sin(theta3)-0.5* I3yx *cos(theta3)
*sin(theta2) *sin(theta3)+0.5*I3zx* cos(theta2) *cos(theta3)* sin(theta3)-
0.5*I3xy*cos(theta3)*sin(theta2)*sin(theta3)+0.5*
I3yy*cos(theta3)*cos(theta3)*sin(theta2)*sin(theta2)-0.5* I3zy* cos(theta2)
*cos(theta3)*cos(theta3)*sin(theta2)+0.5* sin(theta3) *cos(theta3)
*cos(theta2)*I3xz-0.5*cos(theta3) *cos(theta3)*sin(theta2)*cos(theta2)*
I3yz+0.5*cos(theta2) *cos(theta2)*cos(theta3)*cos(theta3) *I3zz),0];
result3=[0,0,1/(h3*h3*(sin(theta3)
*(sin(theta3)*cos(theta2)*cos(theta2)+sin(theta3)* sin(theta3)* sin(theta2)
*sin(theta2) + cos(theta3) *cos(theta3))*m3))];
result=[result1;result2;result3];
```

%H_Matrix : Coriolis matrix

```
function result=Hmatrix(theta2,theta2_dot,theta3, theta3_dot)

v=Dmatrix(theta2,theta3, I2yy,  I2yz  , I2zy , I2zz,  I3xx,  I3xy  ,I3xz ,
I3yx  , I3yy,  I3yz , I3zx,  I3zy , I3zz, m2, m3,b2,h3 )
x=[0;theta2;theta3]
c=[0;theta2_dot;theta3_dot]

for i=1:1:3
for j=1:1:3
    for k=1:1:3
       r1=(diff(v(i,j),x(k,1))-(0.5*diff(v(j,k),x(i,1))))*c(j,1)*c(k,1)
       h(k)=sum(r1);
    end
    Hnn(j)=sum(h);
end
y(i)=sum(Hnn)
end
result=[y(1);y(2);y(3)]
```

%G_matrix: gravitational matrix

```
function  result=Gvector(theta2,theta3,theta2_imu,theta3_imu)

grav0=(R3_0(theta2,theta3)*[-
cos(theta2_imu)*sin(theta3_imu);sin(theta2_imu);cos(theta2_imu)*cos(theta3_im
u)]*9.81).';
grav1=m1*grav0*J1();
grav2=m2*grav0*J2(theta2,b2);
grav3=m3*grav0*J3(theta3,h3,theta2);
grav=[grav1(1,1)+grav1(1,2)+grav1(1,3);grav2(1,1)+grav2(1,2)+grav2(1,3);grav3
(1,1)+grav3(1,2)+grav3(1,3)];
```

%%% linearization code

%Parameters: Variables which used in the model.

```
global   l2 b2 h3  I3xx I3xy I3xz I3yx I3yy I3yz I3zx I3zy I3zz I2xx I2xy
I2xz I2yx I2yy I2yz I2zx I2zy I2zz m2 m3
l2=0.045
b2=0.082
h3=0.024
I3xx=0.3
I3xy=0
I3xz=0.04
I3yx=0
I3yy=0
I3yz=0
I3zx=0.2
I3zy=0
I3zz=0.5
I2xx=0
I2xy=0
I2xz=0
I2yx=0
I2yy=0.6
I2yz=0.2
I2zx=0
I2zy=0.3
I2zz=0.2
m2=0.03572
m3=0.02405

v=invD(theta2,theta3, I2yy,  I2yz  , I2zy , I2zz,  I3xx,  I3xy  ,I3xz , I3yx
, I3yy,  I3yz , I3zx,  I3zy , I3zz, m2, m3,b2,h3 )
z=Gvector(theta2,theta3,theta2_imu,theta3_imu)
c=Hmatrix(theta2,theta2_dot,theta3, theta3_dot)
t=[0;t2;t3]
b=v*(t-z-c)
%%%%to find matrix A
A21=diff(b(2),theta2)
A22=diff(b(2),theta2_dot)
A23=diff(b(2),theta3)
A24=diff(b(2),theta3_dot)
A41=diff(b(3),theta2)
A42=diff(b(3),theta2_dot)
A43=diff(b(3),theta3)
A44=diff(b(3),theta3_dot)

theta3_imu=0
theta2_imu=0
theta2=0
theta3=0
theta2_dot=0
theta3_dot=0

%%%to find matrix B
b21=diff(b(2),t2)
b22=diff(b(2),t3)
b41=diff(b(3),t2)
b42=diff(b(3),t3)
```

```
%%%state space model

A=[0 1 0 0;0.1149 0 -0.0276 0;0 0 0 1;0 0 0 0]
B=[0 0;3.998 0;0 0;0 500]
C=[1 0 0 0;0 0 1 0]
 D=[0 0;0 0]
 %%% design requirements  zeta=1  f=2 w=2*pi*f
 %%% to find the regulator matrix
 P=[-80 -80.01 -80.02 -80.03]
 K=place(A,B,P)
```

Appendix D: Arduino code

1) PID controller:

```
double pid_controller_roll (double error, double error_previous)
{
double result1=0;
  ///******* calculat the time ******
delta_time = (micros() - last_process) / 1000000.0;
last_process = micros();
//delta_time = 0.000001;

  //*********** PID *****
  KP = (P * error );
  KI += I * error * delta_time;
  KD = ((error - error_previous) / delta_time)* D;

  result1 = KP + KD + KI;
    return result1;
}

double pid_controller_pitch(double error2 , double error_previous2)
{
  double result=0;
  ///******* calculat the time ******
delta_time2 = (micros() - last_process2) / 1000000.0;
last_process2 = micros();
//delta_time = 0.000001;
  //*********** PID *****
  KP2 = (P2 * error2 );
  KI2 += I2 * error2 * delta_time2;
  KD2 = ((error2 - error_previous2) / delta_time2) * D2;

  result = KP2 + KD2+ KI2;

          return result;

}
```

2) Control Code of Brushless DC Motor

```
void move_motor_roll(int sign_roll, int step_roll) {

  for(int i=0;i<step_roll;i++) {

  if(direction_roll * sign_roll < 0 ) {

  direction_a_roll = direction_a_roll * -1;
  direction_b_roll = direction_b_roll * -1;
  direction_c_roll = direction_c_roll * -1;
  direction_roll = direction_roll * -1;

  }

  if(position_a_roll + direction_a_roll > 255 | position_a_roll + direction_a_roll < 0) direction_a_roll = direction_a_roll * -1;
  if(position_b_roll + direction_b_roll > 255 | position_b_roll + direction_b_roll < 0) direction_b_roll = direction_b_roll * -1;
  if(position_c_roll + direction_c_roll > 255 | position_c_roll + direction_b_roll< 0) direction_c_roll = direction_c_roll * -1;

  position_a_roll = position_a_roll + direction_a_roll;
  position_b_roll= position_b_roll + direction_b_roll;
  position_c_roll = position_c_roll + direction_c_roll;

  p1 = SinusValues[position_a_roll];
  p2 = SinusValues[position_b_roll];
  p3 = SinusValues[position_c_roll];

  analogWrite(ph1,p1);
  analogWrite(ph2,p2);
  analogWrite(ph3,p3);
  //delay(1);
  }

}
```

```
void move_motor_pitch(int sign_pitch, int step_pitch) {

  for(int i=0;i<step_pitch;i++) {

  if(direction_pitch * sign_pitch < 0 ) {

  direction_a_pitch = direction_a_pitch * -1;
  direction_b_pitch = direction_b_pitch* -1;
  direction_c_pitch = direction_c_pitch * -1;
  direction_pitch = direction_pitch* -1;

  }

  if(position_a_pitch + direction_a_pitch > 255 | position_a_pitch + direction_a_pitch < 0) direction_a_pitch = direction_a_pitch * -1;
  if(position_b_pitch + direction_b_pitch > 255 | position_b_pitch + direction_b_pitch < 0) direction_b_pitch = direction_b_pitch* -1;
  if(position_c_pitch + direction_c_pitch > 255 | position_c_pitch+ direction_b_pitch < 0) direction_c_pitch = direction_c_pitch * -1;

  position_a_pitch = position_a_pitch + direction_a_pitch;
  position_b_pitch = position_b_pitch + direction_b_pitch;
  position_c_pitch = position_c_pitch + direction_c_pitch;

  p4 = SinusValues2[position_a_pitch];
  p5 = SinusValues2[position_b_pitch];
  p6 = SinusValues2[position_c_pitch];

  analogWrite(ph4,p4);
  analogWrite(ph5,p5);
  analogWrite(ph6,p6);

  }

}
```

3) The Main Page

```
#define ph1 5
#define ph2 6
#define ph3 7

#define ph4 11
#define ph5 10
#define ph6 9
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
   #include "Wire.h"
#endif
MPU6050 mpu;
// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };

*********              INTERRUPT DETECTION ROUTINE            ********

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
   mpuInterrupt = true;
}
uint8_t i2cData[14]; // Buffer for I2C data
const int SinusValues[256];
```

```
void calcSinusArray(uint8_t maxPWM, uint8_t *array)
{
  for(int i=0; i<N_SIN; i++)
  {
array[i] = 128 -sin(2.0 * i / N_SIN * 3.14159265) * maxPWM / 2.0;
  }
}

// ********* Motor roll *****************

 static uint32_t position_a_roll = 0;
static uint32_t position_b_roll = 170;
static uint32_t position_c_roll = 170;
int direction_a_roll = 1;
int direction_b_roll = 1;
int direction_c_roll = -1;
int direction_roll = 1;
int p1,p2,p3;

//********** Motor pitch******************
static uint32_t position_a_pitch = 0;
static uint32_t position_b_pitch = 170;
static uint32_t position_c_pitch = 170;
int direction_a_pitch = 1;
int direction_b_pitch = 1;
int direction_c_pitch= -1;
int direction_pitch = 1;
int p4,p5,p6;


//**********roll  pid parameters  ***************
double setpoint = 0;
double current_position = 0;
double output = 0;
long last_process = 0;
float delta_time = 0;
double error = 0;
double error_previous = 0;

double
P = 1.5,
I = 0.0005,
D = 0.002;
double
```

```
KP = 0,
KI = 0,
KD = 0;
double result1=0;




//************* pitch pid parameters  ******************
//VARIABELS
double setpoint2 =0;
double current_position2 = 0;
double output2 = 0;
long last_process2 = 0;
float delta_time2 = 0;
double error2 = 0;
double error_previous2 = 0;

double
P2 = 0.5,
I2 = 1,
D2 = 1;
double
KP2 = 0,
KI2 = 0,
KD2 = 0;

void setup() {
void setup() {
   // join I2C bus (I2Cdev library doesn't do this automatically)
   #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
      Wire.begin();
      TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
   #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
      Fastwire::setup(400, true);
   #endif

   Serial.begin(115200);
   while (!Serial); // wait for Leonardo enumeration, others continue immediately

   // initialize device
   Serial.println(F("Initializing I2C devices..."));
   mpu.initialize();
```

```
    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));

    // wait for ready
    Serial.println(F("\nSend any character to begin DMP programming and demo: "));
    while (Serial.available() && Serial.read()); // empty buffer
    while (!Serial.available());                 // wait for data
    while (Serial.available() && Serial.read()); // empty buffer again

    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

    // supply your own gyro offsets here, scaled for min sensitivity
    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

    // make sure it worked (returns 0 if so)
    if (devStatus == 0) {
        // turn on the DMP, now that it's ready
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        // enable Arduino interrupt detection
        Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();

        // set our DMP Ready flag so the main loop() function knows it's okay to use it
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;

        packetSize = mpu.dmpGetFIFOPacketSize();
    }

  pinMode(ph1, OUTPUT);
  pinMode(ph2, OUTPUT);
  pinMode(ph3, OUTPUT);
  pinMode(ph4, OUTPUT);
  pinMode(ph5, OUTPUT);
```

```
  pinMode(ph6, OUTPUT);
    Serial.begin(115200);
setup_MPU();
setpoint = 0;
setpoint2 = 0;
calcSinusArray(255, SinusValues);
}




void loop() {

if (!dmpReady) return;

while (!mpuInterrupt && fifoCount < packetSize) {
   current_position = ypr[1] * 180/M_PI;
   current_position2 = ypr[2] * 180/M_PI;

  //****** error ******
  error = setpoint - current_position;
  error2 = setpoint2 - current_position2;

  //********** pid controller   *******

output = pid_controller_roll(error, error_previous);
output2 = pid_controller_pitch(error2, error_previous2);
error_previous = error;
error_previous2 = error2;


if (current_position != 0){
if(output > 0 ){
  move_motor_roll(1, abs(output)) ;

  }
 else {
move_motor_roll(-1, abs(output)) ;
  }
}


if (current_position2 != 0){
if(output2 > 0 ){
```

```
 move_motor_pitch(1, abs(output2)) ;

 }
else {
   move_motor_pitch(-1, abs(output2)) ;
 }
}
  // reset interrupt flag and get INT_STATUS byte
  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();

  // get current FIFO count
  fifoCount = mpu.getFIFOCount();

  // check for overflow (this should never happen unless our code is too inefficient)
  if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
     // reset so we can continue cleanly
     mpu.resetFIFO();
     Serial.println(F("FIFO overflow!"));

  // otherwise, check for DMP data ready interrupt (this should happen frequently)
  } else if (mpuIntStatus & 0x02) {
     // wait for correct available data length, should be a VERY short wait
     while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

     // read a packet from FIFO
     mpu.getFIFOBytes(fifoBuffer, packetSize);

     // track FIFO count here in case there is > 1 packet available
     // (this lets us immediately read more without waiting for an interrupt)
     fifoCount -= packetSize;

     #ifdef OUTPUT_READABLE_YAWPITCHROLL
        // display Euler angles in degrees
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

     #endif

}
}
```

References

[1] Wikipedia, San Diego, California, USA,2014. Available at https://en.wikipedia.org/wiki/Gimbal_(company). Last visit date 06/07/2016.

[2] Dys, Beijing exhibition center,China,2016. Available at http://www.dys.hk/ProductShow.asp?ID=19 . Last visit at 06/07/2016.

[3] Theory of Machines and Mechanisms , Joseph Edward Shigley and John Joseph Iicker,Jr, McGraw-Hill Book Company,Auckland Bogota Guatmala Hamburg Lisbon, 1981.

[4] R N Jazar. Theory of Applied Robotics: Kinematics, Dynamics, and Control. Springer, 2007.

[5] Robot Modeling and Control, Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, JOHN WILEY & SONS, INC,New York,1985.

[6] Arduino web site, https://store.arduino.cc/ . Last visit at 06/07/2016.

[7] NSK Company, Table of Contents Ball Bearings,pp 11. Available at www.bearing.co.il/1-AM7B.pdf . Last visit at 06/07/2016.

[8] Wikipedia. Available at https://en.wikipedia.org/wiki/PID_controller . Last visit at 06/07/2016