Wrocław University
of Science and Technology

# Intermediate project
# The self-balance device Gimbal

**Author** : Nguyen Viet
**Faculty**: Embedded Robotics (AER)
**Department** : Department of Electronics
**Instructor** : Ph.D Witold Paluszyński
**Date** : 28 January 2022

**ABSTRACT**

The control of a self-balancing device has been studied for many years and, despite several achievements, there are still open issues. Gimbal device is one of the famous applications of self-balancing control theory, it is the device help to keep balance an object by control its three orientation Yaw, Pitch and Roll. The aim of this project to build a simple Gimbal by implementing the Kalman Filter to control orientation from low-cost servo motors and IMU sensor which contain accelerameter and gyroscope. The project also describe the raw data process as well as the mathematical calculation to apply Kalman Filter properly.

## 1 Introduction:

A self-stabilizing platform ( Gimbal ) is a pivoted support that permits rotation of an object about an axis. A set of three gimbals, one mounted on the other with orthogonal pivot axes, may be used to allow an object mounted on the innermost gimbal to remain independent of the rotation of its support. The popular application of gimbal is a supporting accessory for camera where the camera men need to catch the stable frame while doing a lot of movement and there may be shake when holding the camera.

The gimbal of this project consists of three main parts:

- Servo motor: 3 MG996R servo motor for the 3-axis control of the object frame.
- Target object: The object need to be kept balance.
- The base part: the place attach IMU sensors.

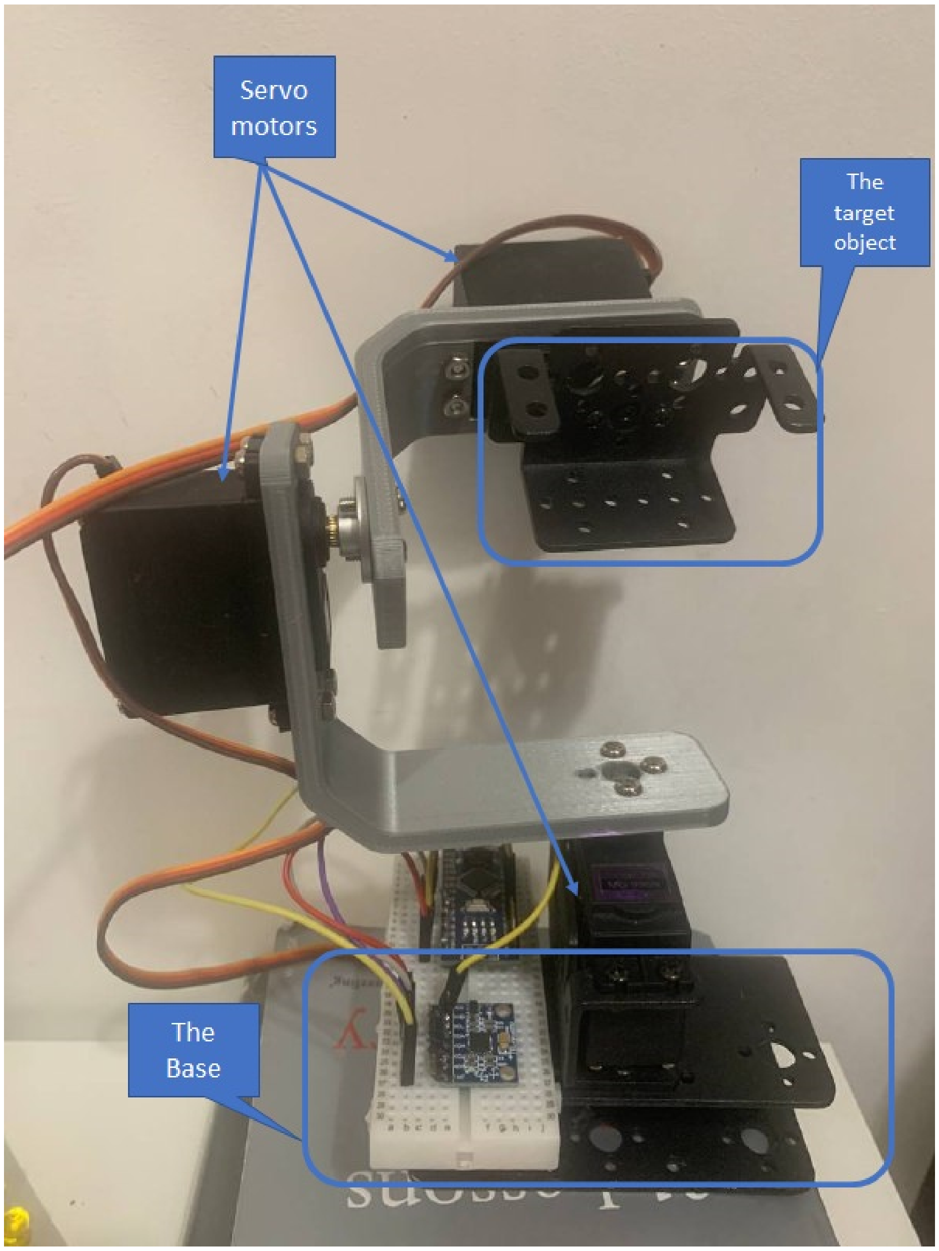

Figure 1: The device Gimbal

The device with the main function is to balance the target object part under any impact to the body part by control 3 servo motors corresponding to 3 orientation ( Yaw, Pitch and Roll) axis. In order to do that, a close loop feedback control system from IMU sensor signal (Accelerometer , Gyroscope) and Fusion method (Kalman filter) will be designed and implemented.

## 2 Assumption:

- All the noise in the system (process noise and measurement noise) is assumed additive white Gaussian noise.
- The rotate angle of each orientation axis only in range 0 $\rightarrow 180^o$

## 3 Hardware set up:

The hardware consist of:

- 3 of MG996R servo motors
- Buck Converter DC/DC
- Arduino Mini
- IMU LPS25HB 6D0F
- Battery Li-on 18500mA 7.4V

The following figure show the connection between them:

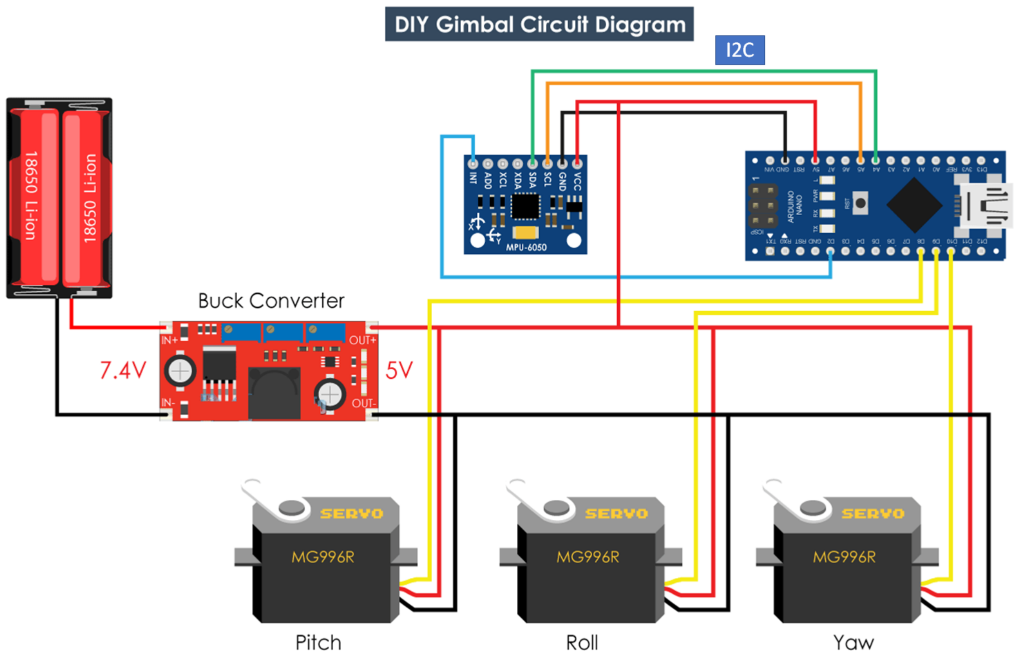

Figure 2: The circuit diagram of device

The IMU sensor use and $I^2C$ interface and therefore can be easily combined with and Arduino or Raspberry Pi. It will be mounted tightly on the base part of the device.

The servo motor MG996R are controlled by Pulse Width Modulation signal from Arduino controller. They are powered through Buck Conventer DC/DC to ensure stability and a guaranteed 5V power supplier.

## 4 Proposed method and implementation:

When working with a sensor unit containing a camera and an IMU, several reference coordinate systems have to be presented. The four major coordinate systems are depicted in following Figure:

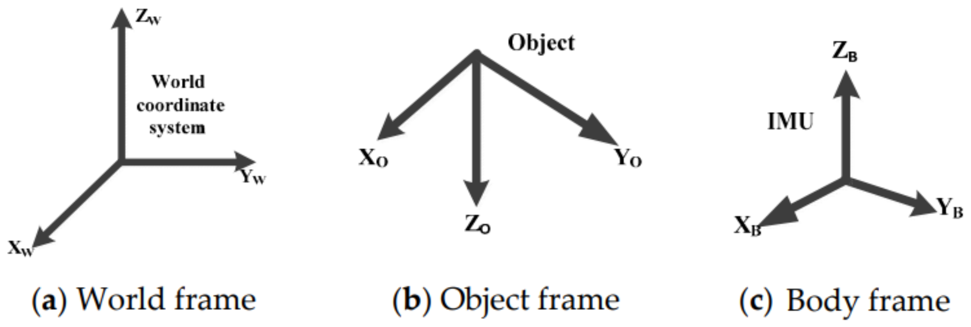

Figure 3: The coordinate type of system

Reference frame for the system

- Global frame/world frame (w): This frame aids the user to navigate and determine the pose estimation in relative to IMU and camera frames
- IMU/body frame (b): This frame is attached to the IMU (accelerometer and gyroscope) on the mobile robot
- Object coordinate frame (o): the frame of the object which need to balance and three servo motors are used to control coordinate of the object.

The orientation angle Yaw, Pitch and Roll of a coordinate system is showed in Figure (4). There is always a set of angle Yaw, Pitch and Roll where we can apply to transform a coordinate to another coordinate ( noticed that the order of rotating angle is so important, it means the same set of orientation but different order will lead to different coordinate).

At the rest state of the device, the world frame, body frame and object frame are the same. Assume we haven't built any control, when we rotate the device to a certain position, it equivalents with applying an orientation set to the body frame, in this case the body frame and object frame also change with respect to world frame but the object frame doesn't change with body frame. The task is to balance the target object, it means to using servo motors transfer the object frame to same as world frame. In order to solve that, we need to know the set orientation angle and apply for the object frame by servo motor. And we can extract this orientation set by data from IMU sensor.

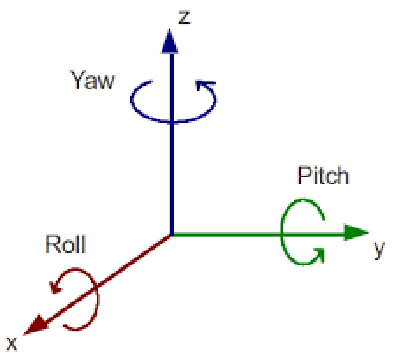

Figure 4: Orientation of a frame

The aim of this part to design a control system where we can obtain the measurement orientation (Yaw, Pitch and Roll angle in the body frame) of device with high accuracy by applying Kalman Filter and sensor IMU. The sensor IMU consist of 3-dof Gyroscope and 3-dof Accelerameter:

**The gyroscope** measures angular rates of change around each axis with respect to the body frame. Assuming noise-free measurements, we could integrate these raw measurements to get the angular position relative to the body frame. Unfortunately, as usual we will have a fair amount of noise present in the gyroscope readings. If we were to integrate this, we would add up all the small errors due to noise, which then would accumulate over time and become unbounded in magnitude. This is commonly known as **"gyroscopic drift"**. In order to compensate that drift, we can use data from Accelerometer to calculate orientation angle and using Kalman filter to fusion two of this data.

Regarding to **the Accelerometer**, when the sensor IMU in the rest, it measures solely gravitational acceleration. So that, by resolving forces using the basic trigonometry [5], we can obtain an estimate of the pitch and roll using raw data as following:

$$\begin{aligned}\hat{\phi}_{Acc} &= \arctan(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}).\frac{180}{\pi} \\ \hat{\theta}_{Acc} &= \arctan(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}).\frac{180}{\pi}\end{aligned} \quad (1)$$

Where $A_x$, $A_y$ and $A_z$ is denoted for the raw acceleration measurements along each axis ($\frac{m}{s^2}$). And $\hat{\phi}_{Acc}$ and $\hat{\theta}_{Acc}$ indicate respectively an estimate of the roll and pitch angle which will be used in the Kalman Filter process.

**Raw data pre-processing:**
Before going to Kalman filter process, we need to consider about the raw data from sensor. We need to low-pass filter the raw data before using them, since there will be additive, high-frequency noise overlayed on the sign. Both sensors will have an offset (bias), which means at rest they will not read exactly zero – even though in theory they should. For calculate these errors we can do 200 readings for all outputs while the sensor is in flat still position. Then we sum them and divide them by 200. As we are holding the sensor in flat still position, the expected output values should be 0. So, with this calculation we can get the average error the sensor makes. This process need to be done only once time and the error will be use as initial value of system while coding.

**Kalman filter:**
The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations.The time update equations can also be thought of as *predictor equations*, which will be calculate from signal of Gyroscope Measurements , while the measurement update equations can be thought of as *corrector equations or fusion* which will be calculated by signal of Accelerometer Measurements.

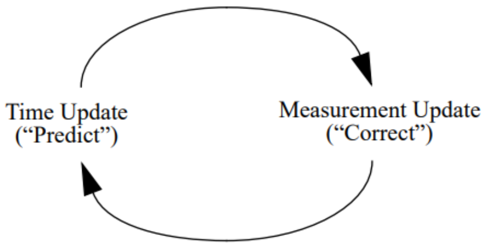

Figure 5: The process of Kalman Filter

In the process Time update, the filter uses measurements from a Gyroscope sensor and underlying system dynamics to predict the future state. This is the best we can do without a measurement update and the estimated state would drift over time. This drift is directly proportional to the amount of error in the process model.In the measurement update, the filter uses measurements from Accelerometer sensor to correct for errors in predicted state.

The articles [1] and [2] show detail Mathematical Formulation of Kalman Filter with raw data from IMU. In order to implement algebra calculation above in C++, I use the library [3].

Finally, we will obtain the orientation angle with respect to *world frame*, which will use to control the motor. In order to visualize and test the result after implementing, we can use application Process [6] by sending above calculated signal from Arduino to computer via Serial Communication, through that, we can see how well the implemented Kalman Filter is. The experiment can be showed as following:

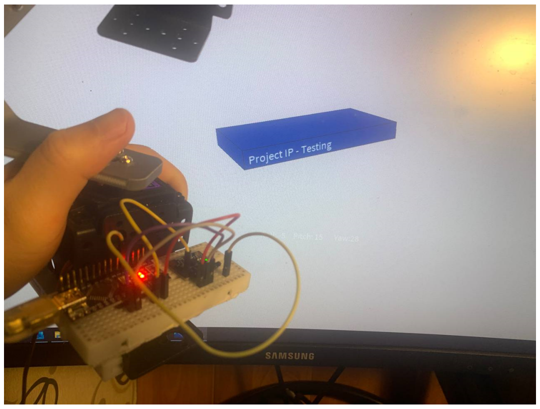

Figure 6: The experiment to confirm the quality of Kalman Filter

Consequently, after receiving good result from sensor by Fusion method, we will map the values of the Yaw, Pitch and Roll, from $-90$ to $+90$ degrees, into values from 0 to 180 which are used for driving the servos.

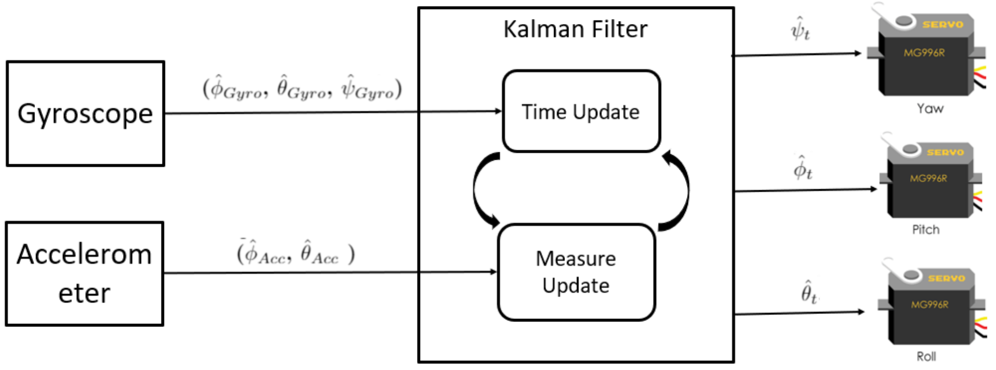

Figure 7: The diagram of signal process

## 5 Result:

The project has successed in implementing an close-loop feedback control system from IMU sensor signal (Accelerometer , Gyroscope) and Fusion method (Kalman filter) to build a self balance device Gimbal. The Gimbal can able to estimate attitude of its self and control servo motor to balance the target object. However, the speed respond of the servo motor is not really smooth,the result is acceptable due to the quality restriction of the servo motors as well as IMU. To improve the project, in addition to improving the quality of the devices, we should consider developing closed-loop controllers for the motor itself.

## 6 Further work:

An advance idea for the device is to attach a camera on the target object part and the camera keep tracking the face under also the impact to the body part. In order to do that, theoretically, we need to apply an orientation set $S_{cam} = (\psi_{cam}, \theta_{cam}, \phi_{cam})$ ( in the object frame) to transform the object frame to another frame where the position of the face is in the middle of the camera. The following way can show the algorithm to calculate $S_{cam}$ :

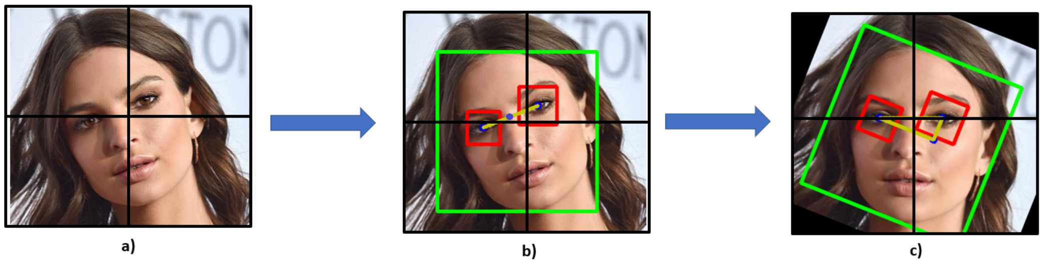

Figure 8: The process of alignment

When we move the face as the Figure(8a), by image processing using *OpenCV*[4] we can detect the location of the middle point between two eyes with origin coordinate is the center of the camera ( Z-axis is from camera to front), and the task is to bring the origin coordinate to the this middle point as Figure(8c).

We can obtain the location of that middle points $(x_m, y_m)$ from location of two eyes as Figure(9) by following:

$$\begin{cases} x_m = \frac{x_l + x_r}{2} \\ y_m = \frac{x_l + x_r}{2} \end{cases} \tag{2}$$

So that with any set of $(x_m,\ y_m,\ \theta)$ we can compute a set $S_{cam}$. As we mentioned above, $S_{cam}$ is the orientation in the object frame, we need to apply the mathematical transformation to obtain the new set in the body frame, so that we can combine with the data from IMU and then control the servo motor.

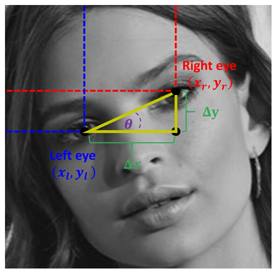

Figure 9: The location obtained from image process

## References

[1] Mathematical Model of an IMU ( https://nitinjsanket.github.io/tutorials/attitudeest/kf)

[2] An Introduction to the Kalman Filter:

https://perso.crans.org/club-krobot/doc/kalman.pdf

[3] MatrixMath for C/C++ (https://www.arduino.cc/reference/en/libraries/matrixmath/)

[4] Face Alignment for Face Recognition in Python within OpenCV:

https://sefiks.com/2020/02/23/face-alignment-for-face-recognition-in-python-within-opencv/

[5] Tilt Sensing Using a Three-Axis Accelerometer p.11 - p.13

https://www.nxp.com/files-static/sensors/doc/app_note/AN3461.pdf

[6] Application Processing (https://processing.org/)