*Alexandru PANĂ*

# DEVELOPMENT AND DESIGN OF A RETRACTABLE STABILIZATION SYSTEM FOR VIDEO CAMERAS ON AN AERIAL PLATFORM

***Abstract:*** *This article presents a simulation model in Matlab Simulink used to simulate the gimbal movement. The following section shows the gimbal design, implementation of the parameters and model in Matlab Simulink and comparative study between simulations determined data and experimental data.*

***Key words:*** *gimbal, system, design, Simulink, Eaglecad, simulation, open-loop, closed-loop.*

## INTRODUCTION

A Gimbal is mounted on an UAV (Unmanned Aerial Vehicle) in order to provide smooth and clear images from a video camera or LIDAR. The Gimbal cancels vibrations and side movement with the use of brushless motors and one or two IMU (Inertial Measurement Units). Another important feature is that the Gimbal is able to rotate freely from the body of the UAV. Fig. 1 shows an example of a standard Gimbal system.

**Fig 1** Example of a standard Gimbal system [1]

## 2. GIMBAL DESIGN

The gimbal is designed in CATIA and is made on a numerically controlled (CNC) machine. Engines and dampers can also be seen as a whole and are, respectively, pink and red. The Gimbal is designed to carry two cameras (thermal and video) or other tools with a maximum weight of about 700 grams.

During the design process I had to make more choices. First is the top of the gimbal, which connects the UAV and the gimbal. For more rigidity, the inner corners are beveled. After designing the majority of the upper part an engine connection must be made. To exclude unintended vibrations from the UAV, an additional platform is attached, which is connected via shock absorbers to the upper platform. The first drive is attached to the extra platform; this additional platform is designed to maintain its rigidity. The first actuator is responsible for the yaw, and then the arm attached to it is called the yaw arm. The arms attached to the second and third servomotors are referred to as the pitch arm and the roll arm, respectively.

The design of all of these arms is done collectively so that the position in fig. 2 is approximately the equilibrium position when, for example, a camera is attached to the roll arm. A mounting space for the MPU6050 sensor near the camera mounting point is left on the roll arm. The yaw arm saves space by inserting the two actuators attached to the arm. The pitch arm is made as easy as possible so that the gimbal can stand in its steady position. The roll arm has multiple slots to allow the cameras to balance and fit as accurately as possible.

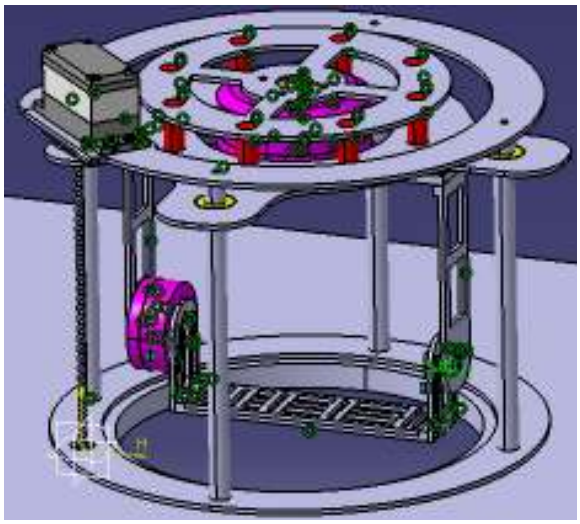

**Fig 2** 3D Design of the gimbal in CATIA

## 3. IMPLEMENTATION

The model contains several blocks, each one representing a separate subsystem. In fig. 3 is presented the complete model, after which each block of the model is treated in detail. The first part explains the model inputs, and the second part shows the implementation PWM converter. Lastly, the DC motor implementation is treated, along with the implementation gimbal block. At the end, the controller is detailed and explained.

There will be six inputs for the model. Three of them will be the angles of the gimbal, roll, pitch and yaw, and the other three will be the angles of the UAV. Fig. 4

presents an example of input for the model. The gimbal will generally receive constant values for the angles input, but there is also the possibility to make a trajectory for each individual angle. The UAV will generally receive input angles representing trajectories which mimic the flight of an UAV.

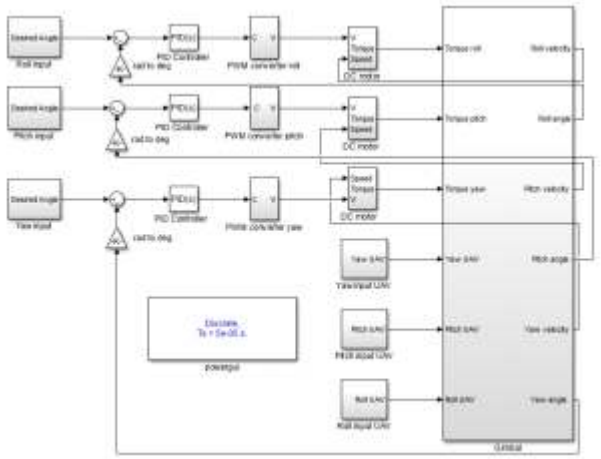

**Fig 3** Simulation of gimbal movement in Simulink


**Fig 4** Example of an input subsystem

Implementation of the equations in Matlab describes the PWM converter. The selection of the right equation is done using IF statements, in order to determine de corresponding PWM values. The roll, pitch and yaw of the gimbal represent the input function, while the output vector which contains the three corresponding PWM and phase values. The PWM converter block is shown in fig. 5.

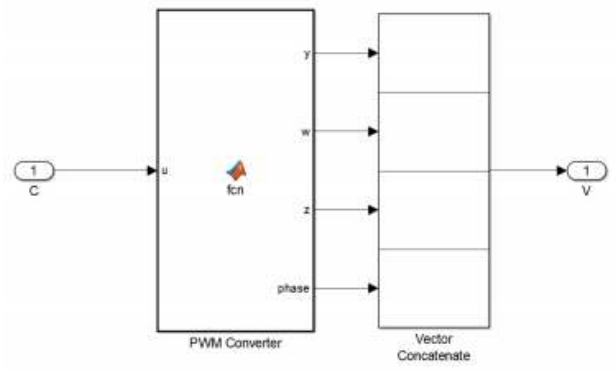


**Fig 5** PWM converter subsystem

In the beginning, the conversion of each output PWM signal is done to a voltage value. This is done by the CVS (Control Voltage Source) block in Matlab Simulink, with the initial amplitude of zero Volts DC. The negative connection is made to Ground and the positive connection is made to the DC motor for output voltage, while the input is PWM value (fig. 6).

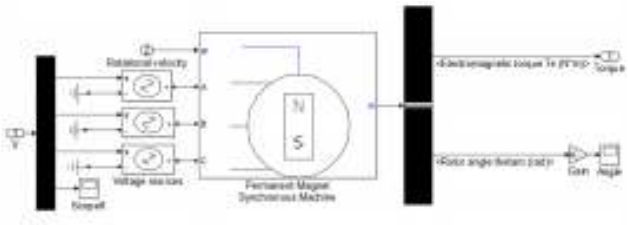

**Fig 6** The DC motor subsystem

Fig. 7 shows the Gimbal block implementation by dividing it into two separate subsystems. One is the Gimbal dynamics, and the other is the IMU block. The IMU has nine inputs, the angles of the UAV and Gimbal, plus the angular velocity of the Gimbal. It also has three outputs, which are the measured yaw, roll and pitch angles of the Gimbal suitable to gravity.

The Gimbal dynamics block inputs are the vectors of angles measured from IMU plus the angular velocities of the gimbal.

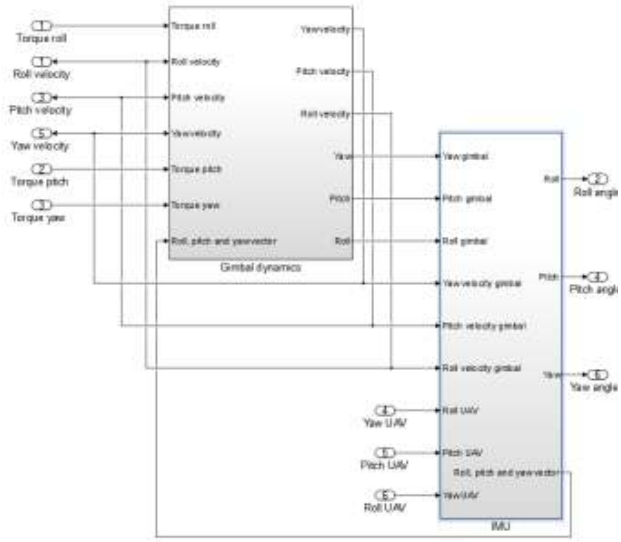


**Fig 7** Gimbal Subsystem, including the IMU and dynamics subsystem

Figure 8 shows the IMU subsystem. The Matlab function outputs the ideal measured angles, after which they are being transformed by adding some white noise in order to make a more realistic assumption.

Figure 9 shows the block which helps calculate the Gimbal dynamics. The simulation model uses a PID controller, which make the control of the gimbal in a position loop, Fig 3. This standard PID controller block from Simulink, chosen because of the limited time frame available for this project.

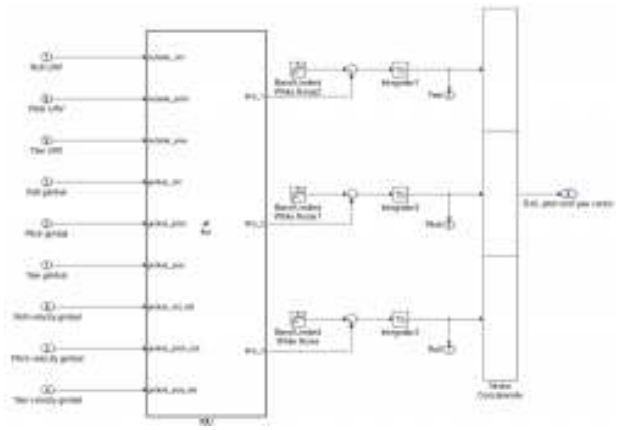

**Fig 8** IMU subsystem

It has a simple implementation, used only to control the Gimbal position loop. This kind of controller is proposed because it is the most common solution used by the current industry. Since the use of brushless motors, the values of dynamic and static friction are assumed to be inconsistent [3].

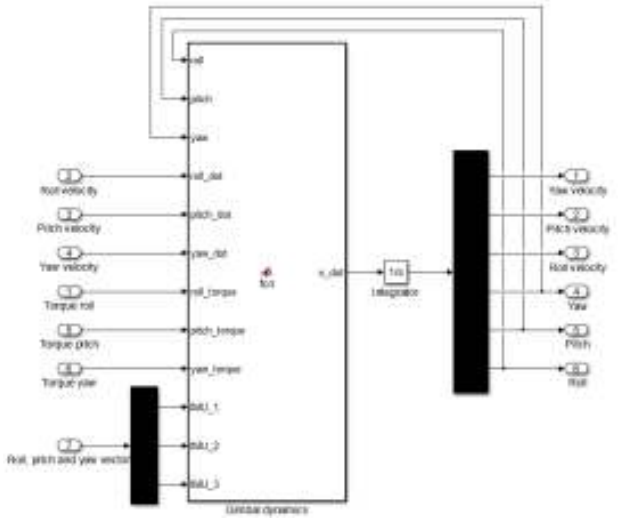

**Fig 9** The subsystem of the gimbal dynamics

## 4. ACQUIREMENT OF EXPERIMENTAL DATA

This chapter explains the acquisition procedure for the experimental data. There is also the problem of drifting for the yaw values, so the MPU6050 measurement data will be analyzed for drifting. Following this, there will be a test to control one single actuator in order to verify if the implemented algorithm works correctly on the Arduino. With the help of the Ziegler-Nichols PID tuning method, stable values for each P, I and D parameter will be further obtained.

### 4.1 Control of the actuators

The first step is to connect all the devices together on a breadboard using normal wires. This is mainly with the purpose of getting familiar with Arduino C++ coding, and testing the Arduino code. The next step is the downscaling of the ensemble to a compact PCB (Printed Circuit Board), designed with the help of Eaglecad.

Fig. 10 shows the layout of the design, which consists of a two layer PCB, with the blue lines representing the bottom side cabling and the red lines representing the top side cabling. Those cablings connect all the components and replace all the previous wires used on the breadboard. The design of the PCB includes connections to fit an Arduino, a PWM driver, three motor drivers, one MOSFET, two capacitors and communication with the MPU6050.

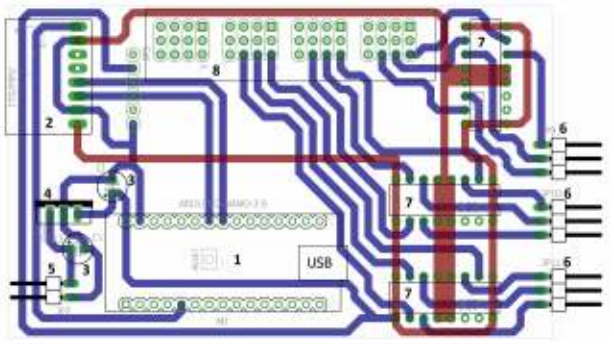


**Fig 10** PCB design

Fig. 10 also shows how all the components on the PCB are numbered: 1 is the Arduino, 2 is the MPU6050, 3 and 4 are the capacitors, 5 is a two-pin connector for the battery, 6 is a three-pin connector for an actuator, number 7 is the H-bridge controller, and number 8 the PWM shield.

#### 4.1.1 Yaw drift

The MPU6050 measurement data has to be analyzed for angle drifting. Firstly, the device has to be calibrated, in order to collect relevant data. All three axes have to be measured while the device is not moving. Figure 11 shows the results of the switching transient for the first approximately 1800 measurement cycles, which are equivalent to about 18 seconds of measurements. The rate of MPU6050 is 100 Hz, which means that 100 cycles should take place in 1 second. After the 18 seconds mentioned above, the roll, pitch and yaw values seem to stabilize.

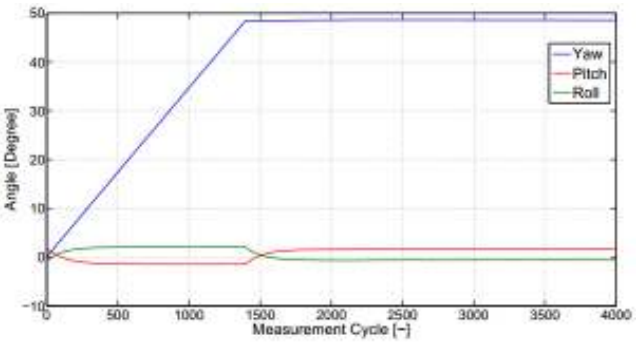


**Fig 11** Yaw, Roll and pitch with switching transient

Fig. 12 points out the drifting of yaw angle. This figure shows how the measurement data zero after the first twenty seconds of measurement cycles of the MPU 6050. The drifting has a slope equivalent to maximum

one degree per about 14000 measurements, or 140 seconds. Because the expected flight time is in the range of minutes, the drifting is considered to be too small in order to noticeably affect the video image from the cameras attached to the gimbal.

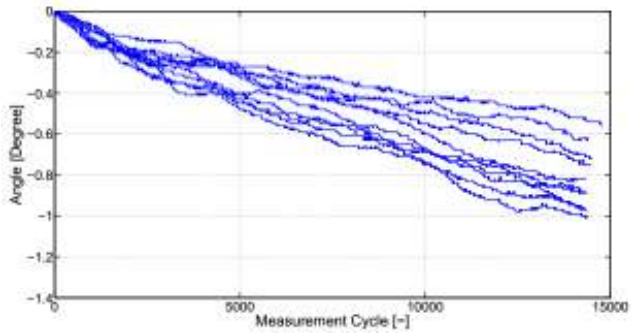


**Fig 12 S**tabilization of the MPU6050 (drifting yaw)

### 4.1.2 Single axis

The first test is conducted for a single axis operation in closed loop and open loop configurations. With the help of a test rig, the control algorithm on the Arduino is verified, before testing on the Gimbal. This testing rig is made up from a rigid base on which the motor is mounted, with a platform on top holding the breadboard and the MP6050.

While operating in open loop mode, the system behaved as expected. Inputting a ramp signal resulted in the expected movement of the Gimbal. Fig. 13 shows the open loop measurement with a decreasing ramp signal input. The results presents the system is working, so we can proceed to the implementation of the closed loop algorithm.

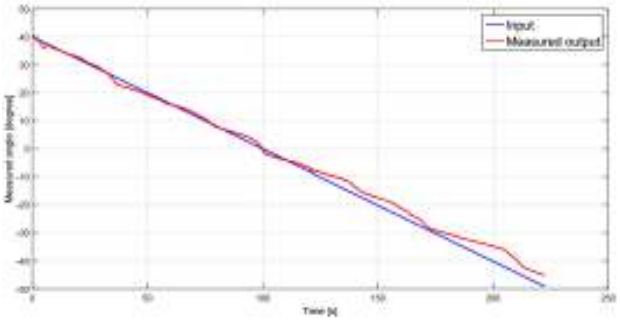


**Fig 13** Open-loop ramp down measurement, where the red line is the measured output and the blue line is the input signal

The closed loop algorithm was implemented on the Arduino without a controller. After testing with a single motor, the measurements show severe unstable behavior. For the purpose of correcting this problem, the use of a PID controller is proposed [3]. Fig. 14 shows the diagram for the control of a single axis. The notations $\Theta_{m,r}$ and $\Theta_m$ are used to represent the desired mechanical angle and the measured angle. The use of a PID controller ensures no occurrence of integral windup because of the anti-windup compensation [2].

The control algorithm is tested with a manually tuned controller, after which it is applied on the Gimbal using the same values. Fig. 15 shows the step responses of the roll angle, for the closed loop algorithm. The values for the P, D and I parameters are set to 0.15, 0.0025 and 9. The results show that the system mostly follows the step trajectory just as expected.

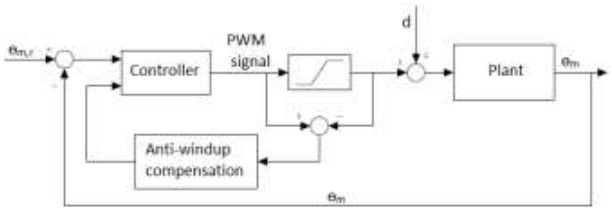


**Fig 14** Single axis control scheme

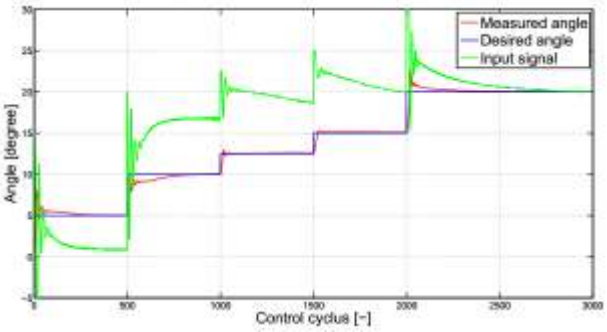


**Fig 15** Response of the system to the reference trajectory

### 4.1.3 Three axes

The control of all three axis of the Gimbal is similarly to that of one axis.

In order to avoid trial and error tuning, the Ziegler Nichols tuning method is used for the closed loop implementation. This method discards the integral and derivative parts and only applies a gain for the controller.

Therefore, if a disturbance is created in the loop, manually altering the position, the systems starts to oscillate. This gain represents the value $K_u$, at which the oscillations start to be constant with the period $P_u$. Knowing the $K_u$ and $P_u$, the values of the P, I and D parameters of the Ziegler Nichols method can be determined from Table 1.

*Table 1*

**Closed-loop calculations**

| | P | I | D |
|---|---|---|---|
| P | $\frac{K_u}{2}$ | | |
| PI | $\frac{K_u}{2.2}$ | $\frac{P}{\frac{P_u}{1.2}}$ | |
| PID | $\frac{K_u}{1.7}$ | $\frac{P}{\frac{P_u}{2}}$ | $\frac{PP_u}{8}$ |

All these are used to determine the right PWM values in order to set the yaw, pitch and roll axis as close to 0 degrees. After this is done, the Ziegler Nichols method for the closed loop tuning is used to determine the initial values of the PID parameters for the roll axis.

Fig. 16 shows how a change of set point determines a constant oscillation of the system. The values of $K_u$ and $P_u$ in the figure are 0.3 and respectively 0.11. Therefore, the corresponding values of the D, I and P parameters have initial values of 0.0024, 3.2086 and 0.1765. In the end these values can be manually tuned to achieve better results.

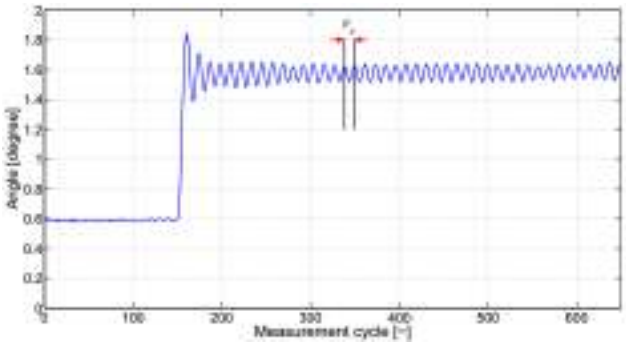


**Fig 16** Manually altering the position of the roll axis

Fig. 17 shows the results after tuning the pitch values, while keeping the set point constant and applying an external disturbance. The corresponding values of the D, I and P parameters have initial values of 0.2353, 1.5180 and 0.0091 for $P_u$ is 0.31 and $K_u$ is 0.4 seconds.

A similar procedure for the yaw axis 0.0099, 1.7647 and 0.2647 for the D, I and P parameters when $K_u$ is 0.45 and $P_u$ is 0.3 seconds.

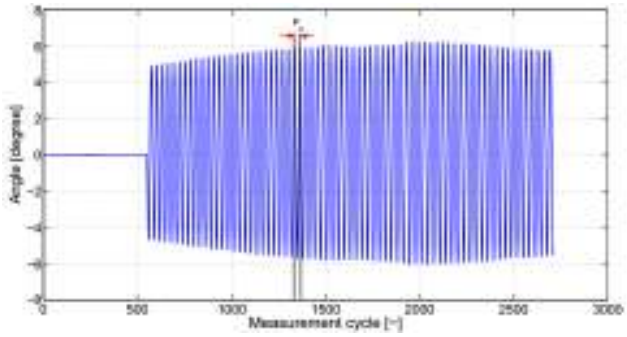


**Fig 17** Pitch axis (external disturbance)

A step response measurement for each axis is done separately, after implementing the initial values for all three PID controllers.

Figure 18 shows all three results in the same plot, even though they were done separately. We can observe that the roll, pitch and yaw axis still present a vibration. The yaw and roll have an especially significant overshoot.

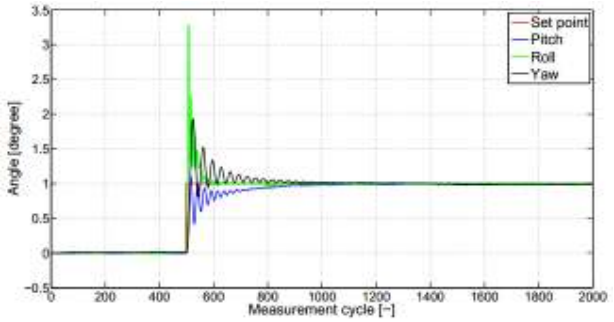


**Fig 18** One figure which combine step response of every individual axis

Even though manual tuning for each axis should show better results, no significant result is observed.

## 5. COMPARATIVE STUDY BETWEEN SIMULATION DETERMINED DATA AND EXPERIMENTAL DATA

This chapter shows the comparison of the open-loop results, and after that, the comparison of the closed-loop results.

### 5.1 Comparison of the open-loop data

The purpose of the straightforward comparison of the open-loop results, presented in this chapter, is to review the result of the simulated model related to the measured values, and, at the same time, to determine the equivalence of the two aspects.

Figure 19 and figure 20 show the rotation obtained based on the model, and the one determined from measurements, together with the actual required rotation. Due to the fact that differences may occur between the movement on the positive axis and the movement on the negative axis, the plotting of the two images was mandatory. It is obvious that the simulation values are an approximation of the actual measurement, especially when it comes to smaller angles.

On the other hand, when the angles are larger, differences occur between the measurement data and the simulation data, especially on the graphical representation on negative direction.

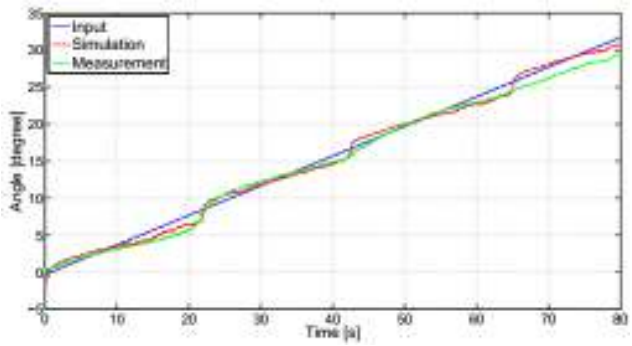


**Fig. 19** Gimbal system rotation on the positive axis - simulation and actual measurement

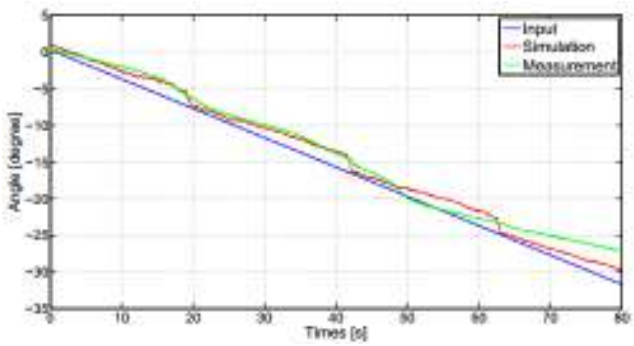


**Fig. 20** Gimbal system rotation on the negative axis - simulation and actual measurement

The two images emphasize that there are no major inconsistencies between the simulated model and the actual measurements. At this point, we have all the necessary data in order to compare the measurements with the closed-loop simulations.

### 5.2 Comparison of the closed-loop data

The measurements and simulations obtained in a closed loop are compared to the same PID controller. The PID controller values are shown in chapter 3; these were obtained by Ziegler-Nichols method, and they are as follows: 0.2647, 1.7647 and 0.0099. Please remember: for this position there is only one feedback reaction. Figure 21 shows the data generated by the simulation, as well as the ones obtained by measurements. Based on the open-loop simulation, it becomes obvious that the simulation data and the data obtained by measurements are approximately identical. But, in the case of larger angles, the differences between the simulation data and the measurement data are worthy of note.

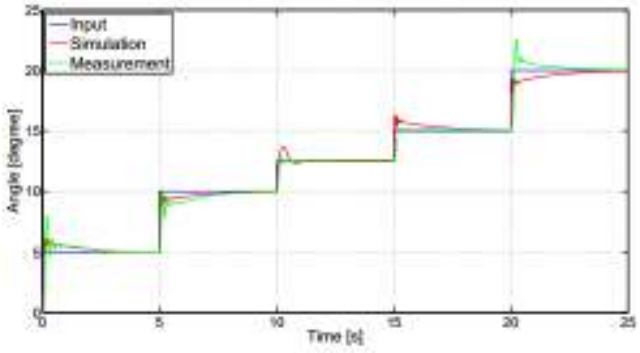


**Fig. 21** Gimbal system step-by-step rotation - simulation and actual measurement

If we only rely on the results, we might think that the gimbal system has a reasonable behavior. But, even if the results appear to be working, during the rotation of the upper part of the gimbal system, it can be observed a certain oscillation of the rotation motor. This behavior can only be seen for the rotation angle, and only when the platform is supposed to maintain a certain position relative to the environment.

## 6. CONCLUSIONS

The conclusions of this report are the design, model and control of a Gimbal system allowing full control of its features. The design was done without any major issues.

In the implementation phase, the need for more data regarding the usage of the actuators became clear. The Gimbal control was achieved with the help of a position feedback loop. Despite the simulation data not showing any instability the system presented some unexpected instability. The implementation of a PID controller was decided, following the current industry trend, tuned with the Ziegler-Nichols tuning method. After this, the Gimbal presented a much better trajectory following behavior, and also the measurement data turned out closer to the simulation data.

The goal of this paper has been largely achieved, even though further improvements to the Gimbal are certainly needed. The Gimbal full control is possible through programming on the Arduino, allowing a thorough understanding of the Gimbal behavior.

## REFERENCES

[1] Gimbal with three axis for UAV. http://www.hobbyking.com/hobbyking/store/__66150__FeiyuTech_G3_3_Axis_Brushless_Gimbal_for_Multi_Rotor_or_Aircraft.html. Accessed 11.2018.

[2] Goubej, M., Schlegel, M. (2014). *Robust PID control of electrical drive with compliant load*, Conference IFAC, World Congress, Cape Town, South Africa, Vol. 19.

[3] Baskin, M., Leblbicioglu, K. (2017). *Robust control for line-of-sight stabilization of a gimbal system*, Turkish Journal of Electrical Engineering & Computer Sciences, pp. 3839-3853.

**Author:**

**Eng. Alexandru PANĂ,** Student, University Politehnica, of Bucharest, Engineering Graphics and Design Department, E-mail: pana.alexandru@incas.ro