



One-degree-of-freedom PID controlled Helicopter

PDE 2420 – Control Systems

Abdelati Zelbane M00374639
Eduardo Abend M00375571

Payam Rahmdel
May 2013

Table of Contents

1. Introduction.....	3
2. Description of Design and Build of Rig	4
3. Programming.....	9
3.1 Arduino	9
3.2 Matlab/Simulink	10
4. Testing.....	11
4.1 Proportional (P).....	12
4.2 Proportional-Derivative (PD)	14
4.3 Proportional-Integral-Derivative (PID).....	15
5. Conclusion	17

1. Introduction

This report will deal with a one-degree-of-freedom helicopter since its inception up to its final tuned closed-loop PID (Proportional-Integral-Derivative) control. This document aims to show and explain the process of designing, building, programming, testing and finally choosing a suitable PID controller for the overall designed system. Since PID controllers are extensively used throughout the industry, the aim is to demonstrate the different aspects and influence on a system of all three gains, namely proportional, integral and derivative gains. These gains will be referred to as K_p , K_i and K_d respectively.

There are different methods that can be chosen to determine the right values of the controller's parameters. Herein, an attempt to use Ziegler-Nichols method has been made. For reasons later in the report mentioned, our group was unable to use that method. Therefore we followed a trial-and-error approach in order to get precise values for the above mentioned controller. Throughout this experiment, the use of screenshots as well as pictures is made to aid the understanding of the impact of the controller's parameters on the one-degree-of-freedom helicopter.

The components used in this experiment are as follows: a DYS 2830-11 motor, a 20A Electronic Speed Controller (ESC), a 10x4.5 propeller, a 2200mAh 11.1V lithium polymer battery (for reasons later mentioned, this battery has been replaced by a 12V 750Ah car battery) and an AMS22S non-contacting analog rotary position sensor. Together with these components, the use of an Arduino (microprocessor) and Matlab/Simulink software is also made.

2. Description of Design and Build of Rig

The first approach towards the build of the rig was to construct it smaller than the actual finished design. Still in the early stages of the creation process, it became clear that the type of motor used is quite powerful and presents risks if it is not mounted and encaged correctly. Due to the described safety issues, the build of a cowling around the propeller used was made by the health and safety expert Neil (Spike) Melton (Fig.1), who has many years of experience in the area. Once the design of the cowling has been

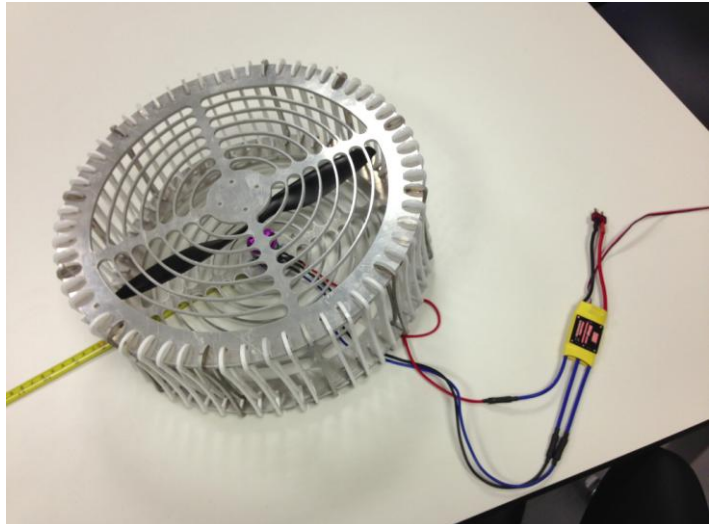


Figure 1. Cowling implemented due to Health and Safety measures

finished and sent to us, adjustments on the computer-aided design (CAD) files have been made in order to keep an overall balance – both practical and aesthetical. For this project, the software used to create and modify such CAD files was SolidWorks.

Still in the design part, a choice has been made of combining the constraints and robustness a physical and dynamic system demands with a more artistic approach making use of organic, natural and round shapes. Although the implementation of such irregular shapes makes the overall designing process more challenging, the end result was of paramount importance to us. The build of the rig has been firstly made in medium density fibreboard (MDF) to have a feel for the overall size and functionality of the same as well as having a cheaper version prototype where alterations could be made without substantive costs.



Figure 2. Spike cutting the pieces in aluminium

Before the designing of the whole structure has been made, the approach taken by our group was to discuss many possible designs, sketch them and then decide how to proceed. Since some time has been invested *before* the start of the parts' design, future modifications have been avoided.

Since this is a dynamic project, all the care has been taken to ensure the stability and

safety of the main body. This care led to the design printed in MDF to be checked and authorised by Spike, without any further modifications needed. Once approved, the

design has been sent to a water-jet cutter (Fig. 2) where it was made in aluminium. Since to weld aluminium plates require much more heat than steel, for example, and does not allow any posterior disassemblies, the approach taken in the construction of the project was to use bolts and nuts. This approach to design aided the assembly process, considering the overall assembly procedure in order to place every single component the simplest way possible.

Following the health and safety guidelines, it became apparent the importance of having a structure that could be easily mounted and fixed to a standard desk used in the

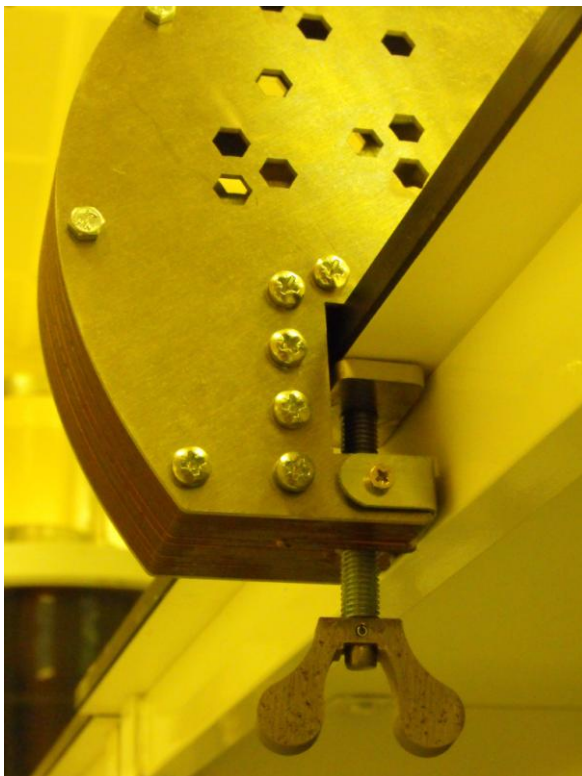


Figure 3. Clamping mechanism

university's laboratories, where extensive tests have been made. Furthermore, considering the overall weight of the system and the forces acting upon it during the performance process, a strong and robust clamping mechanism was necessary.

Moreover, in order to add strength to the structure and to help the work done by the clamp, a consideration has been made to distribute the overall weight in some strategic points, e.g., the “neck” of the structure. To achieve that all the connection parts between the two external aluminium support structures have been carefully designed (Fig. 4).

These connections have been chosen to be in solid MDF to keep the structure as light as possible without compromising on the strength and safety of the overall structure. These decisions have been applied to get the most of the one-degree-of-freedom helicopter during the extensive testing process.

As the pictures shows (Fig. 5 and 6), at the top, a mechanism was designed to connect efficiently the angular sensor to the actual lever arm of the helicopter without overloading the angular sensor’s axis. The mounting kit for the angular sensor is designed in a way that allows us to adjust the sensor itself in order to select the starting point of values to be read, making easy every adjustment whenever needed just by simply

unscrewing a couple of tiny bolts. The structure of the lever arm could be described

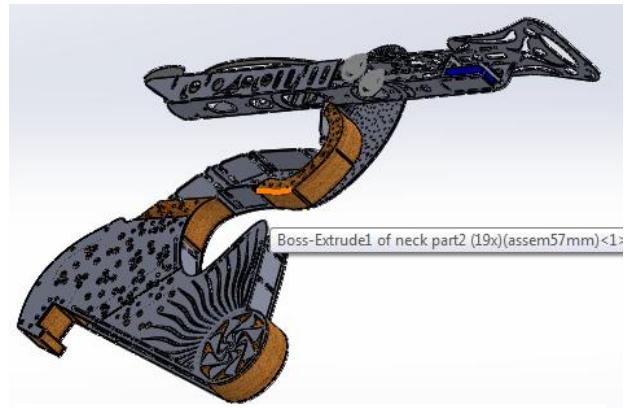


Figure 4. Assembly deliberately open to show internal wooden parts

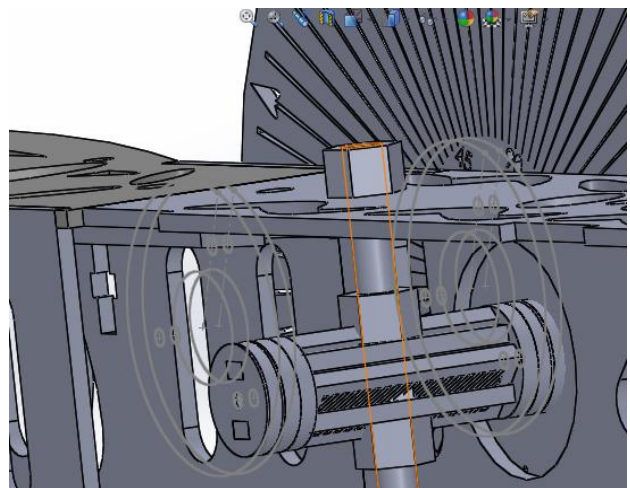


Figure 5. Important connection part between the lever arm and the support

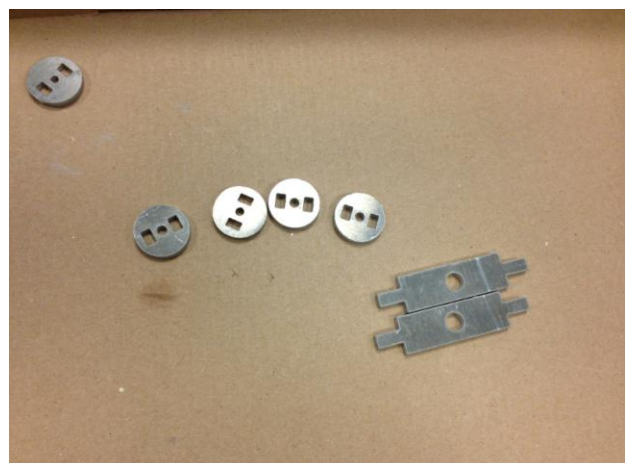


Figure 6. Actual print out in aluminium of the important connecting parts

as a long box, having the width, height and length respectively 50mm, 50mm and 960mm (Fig. 7). This volume of free space all along the lever arm was chosen to fit all the cables and wiring connections. Furthermore, it is important to mention that the design of the arm, once it was sent to be cut in aluminium, presented a small challenge. The original CAD files (Fig. 8), made in MDF, were all fine. In aluminium, we had to split the whole arm in two main separate parts. This was a constraint we had to work around. The root of the issue was related to material limitation,



Figure 7. Internal picture of the lever arm where all the cables are hidden

where we have been asked to fit all our cad files into a 600mm by 600mm aluminium sheet. In the original MDF cuts, this limitation was nonexistent, since the large laser cutter machine can deal with much larger

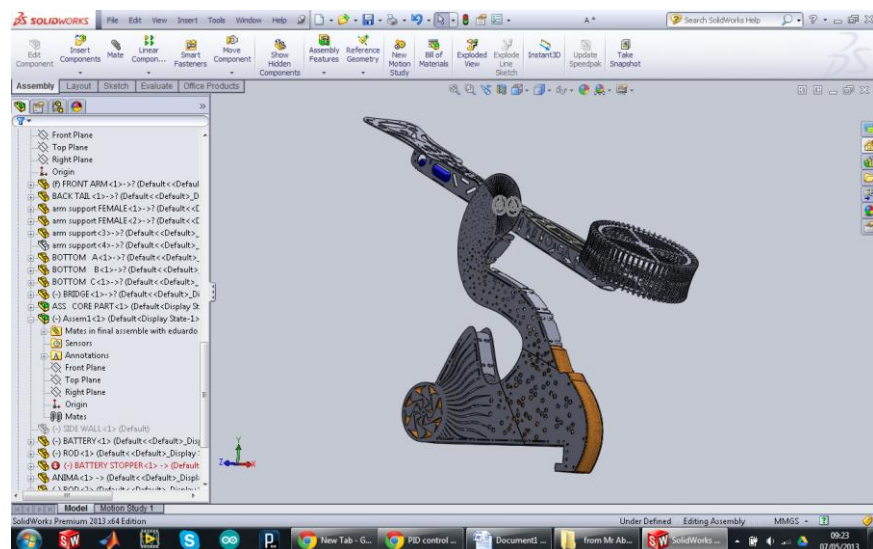
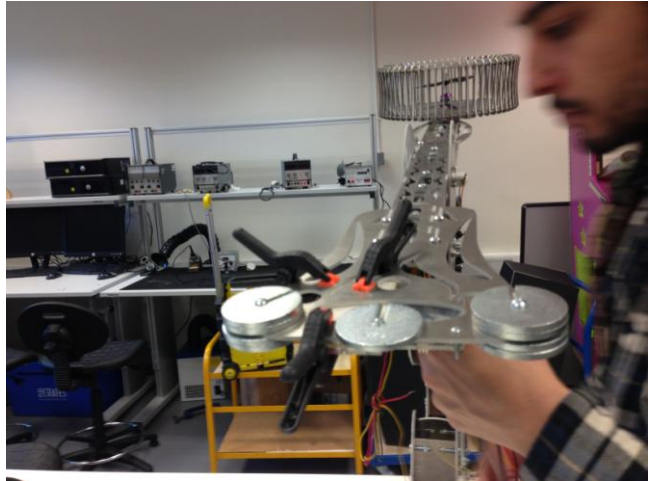


Figure 8. Solidworks rendering

dimensions. Since the overall arm length is greater than the given size, our group had to come up with a solution to such a split. After that, our group ran some test to see the gravity of the problem. It has been proven during the tests that the joining of the two parts of the arm needed extra strength because the stress of the whole lever arm was concentrated exactly there. Our approach to the solution of the problem was to design two small aluminium plates that were bolted to the internal parts of the arm

structure. The effect of this operation demonstrated good motion of the arm with very little friction.

The weight in the lever arm plays a very important role in having a stable system. It is crucial to understand the narrow balance between the cowl and its counterweight. Having too much weight on the cowl side means that even at full power, the motor would not even be able to come off the ground. On the other hand, having too much counterweight, means that the lever arm is not able to



come back to its original position. The design implemented here, therefore, makes use of an adjustable counterweight, using small 100 grams discs bolted to the lever arm (Fig. 9).

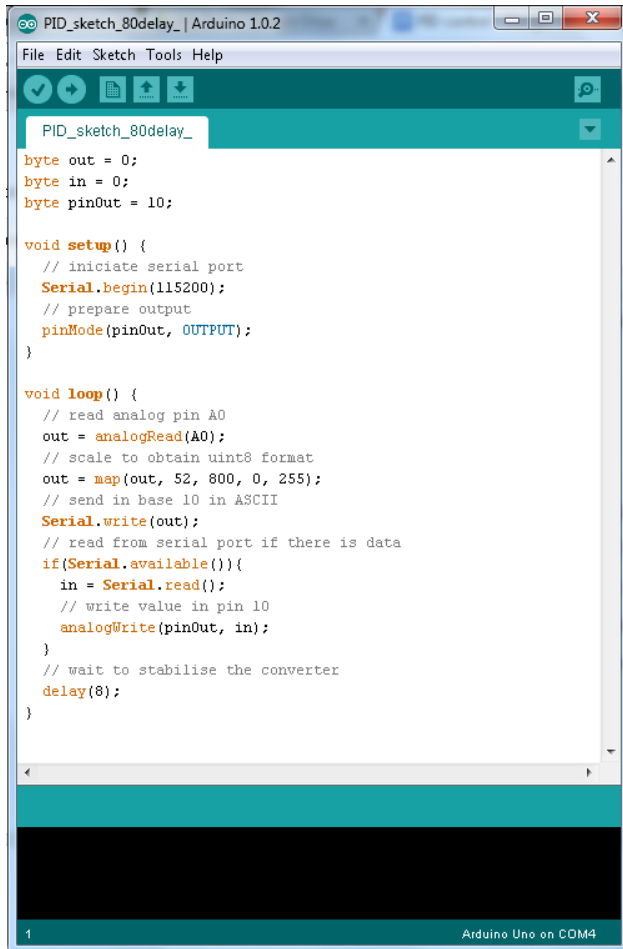
Figure 9. One hundred grams discs bolted to the lever arm to provide overall balance

A problem encountered by our group, and definitely worth mentioning, was in the very brief of the project. The components used herein were sourced by the university. After extensive tests, our group found out that the batteries used for this project were not strong enough to provide the stable current needed to fine tune the PID control values. Since the motor (DYS 2830-11) used is quite powerful and therefore drains high currents (with short bursts of up to 25A) the batteries (2200mAh 11.1V lithium polymer) would run for approximately 10-15 minutes from full charge to completely discharged. The response of the system to certain PID values found with a recently charged battery would largely differ from the response obtained after a very short period of time, making it virtually impossible to tune the system. There were two possible solutions: either to order a smaller dc motor or to get an alternative power supply. Choosing the later, the battery first ordered has been replaced by a 12V 750Ah car battery. The replacement of the battery caused some late disruptions. Since it was an unforeseen problem, and therefore not implemented and considered into the design of the structure, it affected the above mentioned delicate balance. Furthermore, the

mere weight of the cables used to connect the car battery to the structure was, again, another challenge to overcome.

3. Programming

3.1 Arduino

A screenshot of the Arduino IDE interface. The title bar reads "PID_sketch_80delay_ | Arduino 1.0.2". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening, saving, and running. The main text area contains the following code:

```
PID_sketch_80delay_  
  
byte out = 0;  
byte in = 0;  
byte pinOut = 10;  
  
void setup() {  
  // initiate serial port  
  Serial.begin(115200);  
  // prepare output  
  pinMode(pinOut, OUTPUT);  
}  
  
void loop() {  
  // read analog pin A0  
  out = analogRead(A0);  
  // scale to obtain uint8 format  
  out = map(out, 52, 800, 0, 255);  
  // send in base 10 in ASCII  
  Serial.write(out);  
  // read from serial port if there is data  
  if(Serial.available()){  
    in = Serial.read();  
    // write value in pin 10  
    analogWrite(pinOut, in);  
  }  
  // wait to stabilise the converter  
  delay(8);  
}
```

The status bar at the bottom indicates "1" and "Arduino Uno on COM4".

Figure 10. Arduino code implemented

The Arduino code (Fig. 10) starts with defining variables and assigning the pulse width modulation (PWM) pin to number 10. The speed used for serial communication is 115200 baud rate, which is the fastest possible. Since the project demands a real-time response, speed is a very important aspect. Further on, the code prepares the pin 10 to be an output. Inside the loop, the code reads the analog pin A0 which is physically connected to the AMS22S analog rotary position sensor. This sensor is basically a potentiometer which has a slightly reduced angle (of about 340 degrees). After reading the position

sensor's value, the code maps the value so to obtain an uint8 (unsigned integer 8 bits) format. The code then sends the value through the serial port. The next line of code checks to see if there is data available. If there is, the code writes this value - that is in PWM - to the assigned pin number 10. At the end of the code, a short delay is used to stabilise the converter. As it was above mentioned, in this project, speed is very important. The approach taken by our group was to reduce this end of code delay the most, having the lowest possible value before problems occurred. Our group encountered problems with 6 milliseconds or less in the duration of the delay. The

3.2 Matlab/Simulink

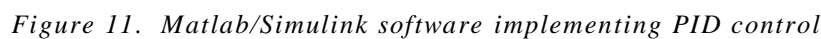
[illegible]

Figure 12. Matlab/Simulink code with graph and slide control

PID block, where all three values can be adjusted. The data right after the PID block

can be seen on the graph in the **turquoise** colour. The value is converted into unsigned integer 8 bits format which is then sent to the Arduino through the serial port. On the left bottom corner of the Simulink program, a serial communications block can be found. Inside this block, many parameters can be adjusted such as communications port and baud rate. The data coming from the Arduino using the same serial parameters is received in Simulink and converted into double precision type. This value, which is the position sensor's readings, can be seen in the **yellow** colour (Fig. 10) on the graph. The value of the position sensor is fed back into the system, making it a closed-loop one. The purpose of this value is to correct errors in the system. Due to difference in magnitude of the values graphed, mathematical blocks have been used in order to obtain meaningful values in the same graph (Fig. 11).

4. Testing

Here we want to remind the reader that the proportional–integral–derivative controller is a generic widely used control system with closed-loop (feedback). The

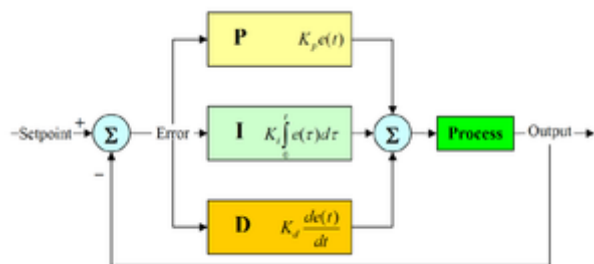


Figure 13. PID control blocks

PID controller calculates the "error" value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by feeding the signal back in a closed-loop.

Once having the hardware and software both running correctly, the first approach taken was to verify the response of the overall system in respect of time. Following the theory on control systems, the approach chosen to get the desirable values was the Ziegler–Nichols tuning method. This method focuses on setting firstly the Ki and Kd to zero and setting Kp to 1. This makes the PID control to behave as if there was no PID at all (Fig. 13), since the only effect on the input would be a proportional gain of 1. Any value fed through a gain (multiplication proportional to input) of 1, says basic Arithmetic, is 1.

Furthermore, following Ziegler-Nichols method, the value of the proportional parameter should be increased until reaching a marginally stable position - where the system is on the borderline of stability, constantly oscillating. That is where some problems were encountered. Following a very methodical engineering approach, our group increased the values of K_p with regular intervals. In the search for a marginally stable system, once the value of K_p has been increased to 1.6, it could be noticed that the system was still stable, although having a tremendously slow settling time. The next number for K_p on our tests up - 1.7 - made the lever arm in the system to reach a point of saturation, hitting the clamped structure on the lowest physical position. This physical encountered problem resulted in our group not being able to obtain the value of K_p when the system is considered marginally stable. Therefore, our group was unable to use Ziegler-Nichols method and instead used a trial-and-error approach to obtain the desired values of K_p , K_i and K_d .

4.1 Proportional (P)

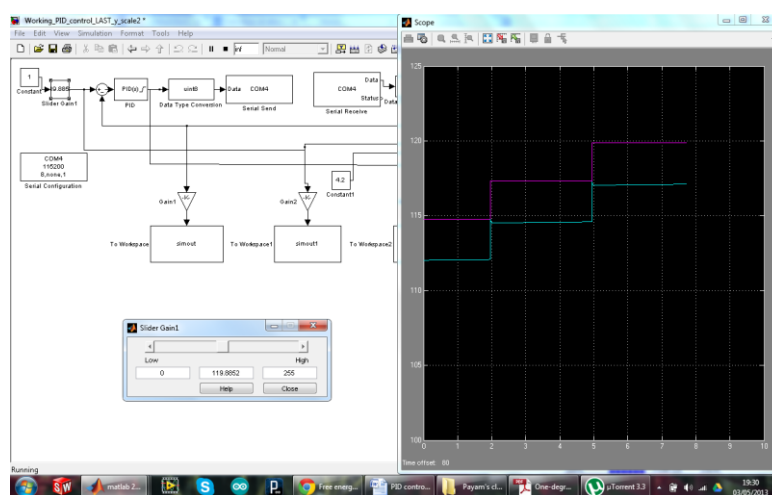


Figure 14. Proportional gain

Here the graph (Fig. 14) shows the rise of the response once setting a power of 150 units (in the slide as well as in the y axis scale), which is the set point. Without the other two terms there is no control of the effective response of the system since the K_p

parameter acts as an amplifier and it just gives this sort of “jump” straight to the set point corresponding to the rising process by the arm. In this particular case there is no

overshoot since the turquoise line, which is the response after the influence of PID, is still under the desired value.

By setting the K_p to 1 and the K_i and K_d to 0, we obtained a clear ‘‘jump’’ of the output that is proportional to the current error value. The proportional response can be adjusted by multiplying the error with the constant K_p gain. We noticed that if the proportional gain is too high, the system becomes unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive (or sensitive) controller. If the proportional gain is too low, the control action was too small when responding to system disturbances. All this can be described as an overshoot where the graph corresponding to the response of our system, overpasses the set point (magenta colour in the graph), resulting in an unstable response.

Our group noticed that this type of response physically corresponds to harmonic oscillations of the arm. When the K_p value is increased, the response is faster, meaning that the system would take less time to reach the set point (improved rising time), but the instability gets greater ending up with faster oscillations. Contrariwise if the proportional value is smaller, the rising time increases and the oscillations become slower.

As clearly the data show (Fig. 15), the more we increase K_p term, the faster is our model to reach the position set by applying a full voltage to our driving propeller. Hence we can say that the rising time is inversely proportional to K_p in absence of K_i and K_d

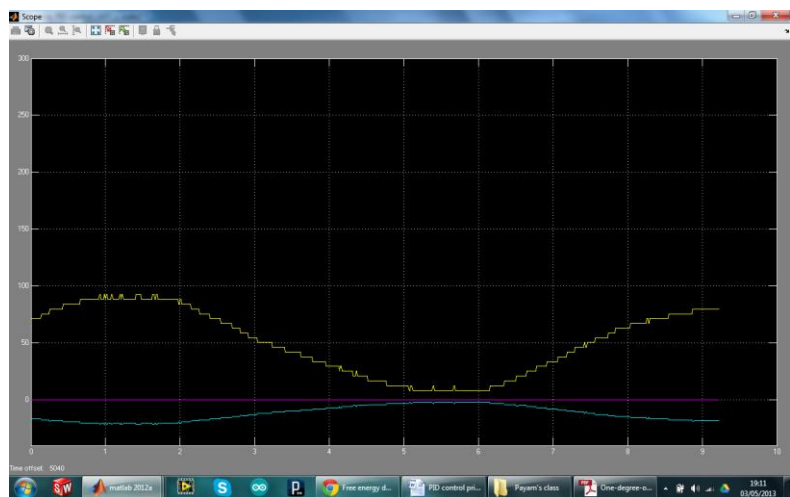


Figure 15 Settling time of a system

parameters; considering the size and the proportion of our model and hence the weight of the structure mounted, this observation can confirm the theory that for a small value of K_p , you have slow rising time with a large error continuously fed back forcing the system to oscillate and then settle after long time.

In this particular screenshot, you can notice the system response is following the yellow path in the opposite direction: that means the error, or more precisely, the variation detected from the sensor is then sent back as an feedback causing this oscillation, typical of an unstable or marginally stable system. In our case, we obtained almost the same track of the two lines setting different values each time for K_p .

4.2 Proportional-Derivative (PD)

After the above observations of the response of a system with only a proportional term, here, the introduction of a second term is made, the derivative term

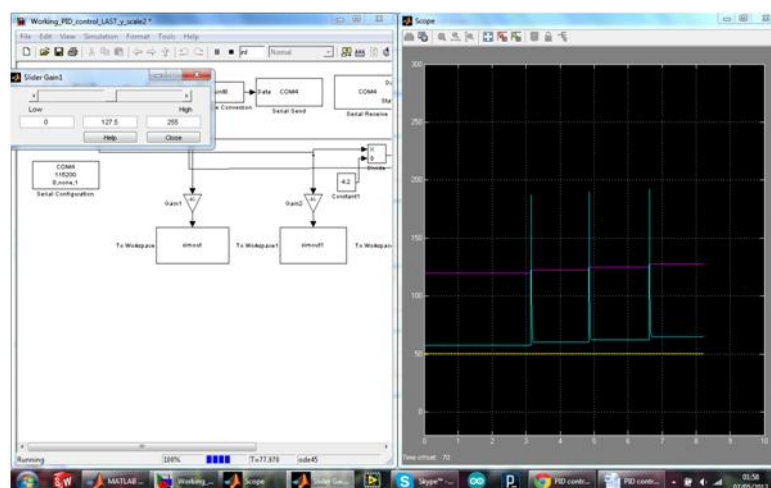


Figure 16. Overshoot observed with a large value of K_d

overshoot noticed by the turquoise colour (value after the PID block) in response to the reference point (magenta colour). The value of K_d used here is 0.5, which is five hundred times larger than the desired value. This term meant that although it is helpful in reducing the settling time when compared to an only-proportional system, when the value is too big, a tremendous overshoot is to be observed.

We noticed that the K_d term (when added to the proportional term) accelerates the movement of the process towards set point (rise time) and reduces or almost eliminates the residual steady-state error that occurs with a proportional controller only. However, since the integral term is responding to accumulated errors from the

past, it can cause the present value to overshoot the set point value (cross over the set point and then create a deviation in the other direction).

The contribution given by the term K_p is proportional to both the magnitude of the error and the duration of the error. Summing the instantaneous error over time gives the accumulated offset that should have been corrected previously. Then, the accumulated error is amplified by multiplying it with the proportional gain and added to the controller output. The magnitude of the contribution of the proportional term to the overall control action is determined by the integral gain.

4.3 Proportional-Integral-Derivative (PID)

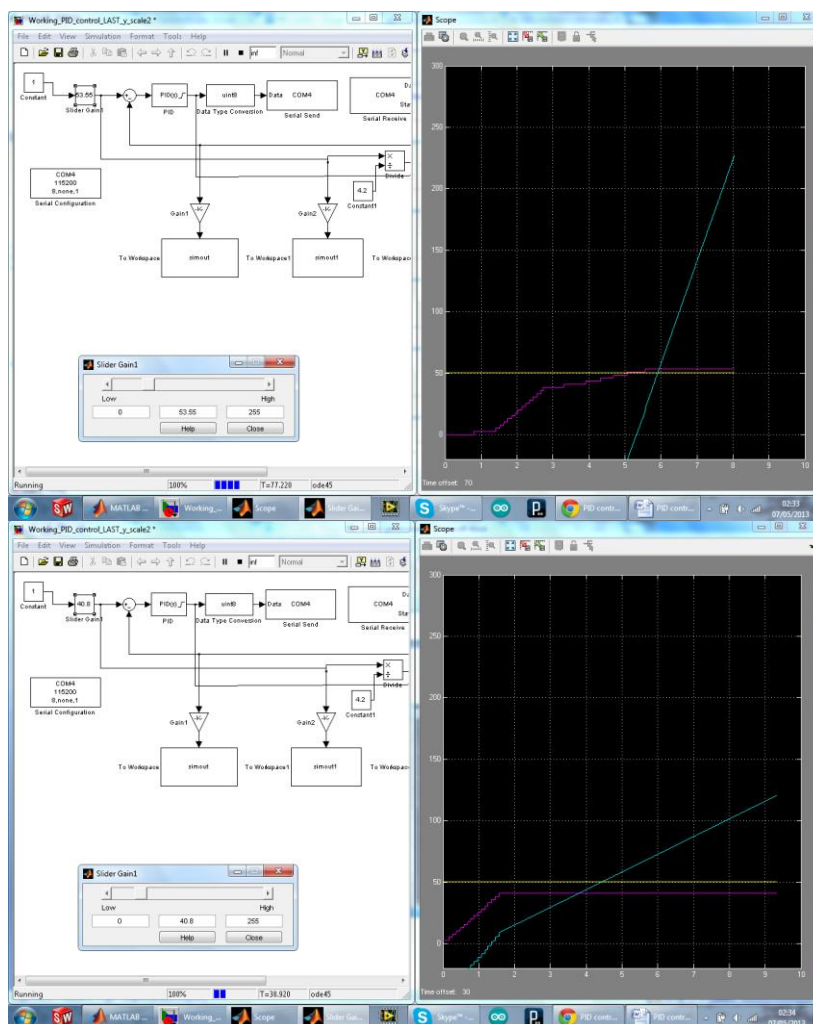


Figure 17. Response of the systems with K_i set to 2 and 0.5 correspondently

Here we start our implementation with the K_i parameter and the following graph (Fig. 17) clearly shows the slope we get once introducing a value of K_i to 2. In order to understand how this parameter affects our response we started with this value and then reducing it massively to have a clear visualization of the affect on the graph. To achieve that we set $K_i=0.5$ and the result

curiously as we expected

improved the response in terms of slope, suggesting we had to implement a smaller value in order to have a response that doesn't either have a positive or negative slope, but constant trend in order to follow the desired value.

In this particular case (Fig. 18), it can be clearly noticed the overshoot of the

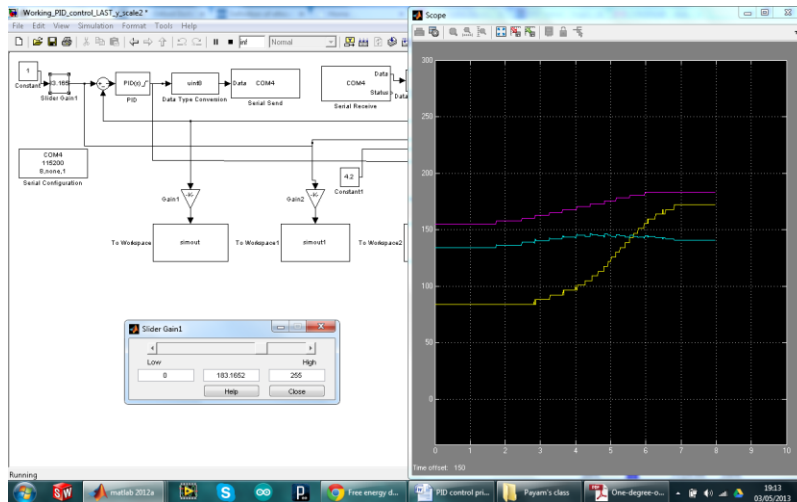


Figure 18. Overshoot of the system's response

system, in this particular case, with the implementation of the Ki term equal to 0.0002, having all the other PID parameters Kp and Kd set to the proper values 0.96 and 0.001

respectively,

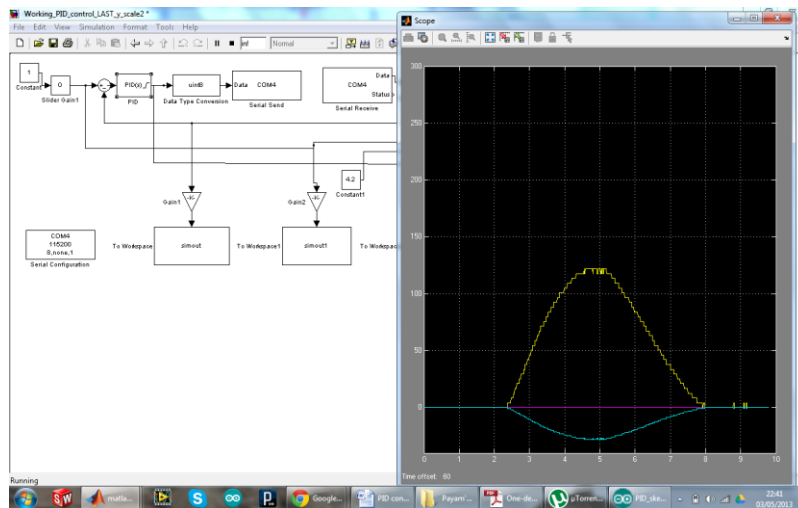


Figure 19. Settling time related to feedback loop

you can notice the massive error followed by the overshoot and at 9/6 seconds (1.5 sec) there are visible 3 tiny oscillations before the response settles down (Fig. 19).

Because the contribution from the integral term is proportional to both the magnitude of the error and the duration of the error, in our case the integral term adds

response which then settles down following the value chosen, with an incredibly reduction of steady state error and also with a faster settling time. In fact, in the X axis is shown the time and 1sec = 6 units, hence the

overshoots and settles down in 8 units of time ($8/6 \text{ sec} = 1.33 \text{ sec}$). When the system overshoots, the top value reached is at 5 units time, which corresponds to $\frac{5}{6} = 0.83$ seconds. In the same fraction of time

more stability in terms of faster rising time, faster settling time with a massively reduced steady state error.

5. Conclusion

After long days of testing the final tuned values are $K_p=0.96$ $K_i=0.0002$ and $K_d=0.001$ (Fig. 20). We can conclude saying that, when designing a model in order to study its response with the implementation of the PID controller, it is very important to take in consideration many aspects of the system itself such as the overall weight, the physical movement the lever arm is able to perform, the type of sensor used to detect the variation from the desired position, the size in relation to the motor used and the power supply. As mentioned before, the weight balance between the cowl and

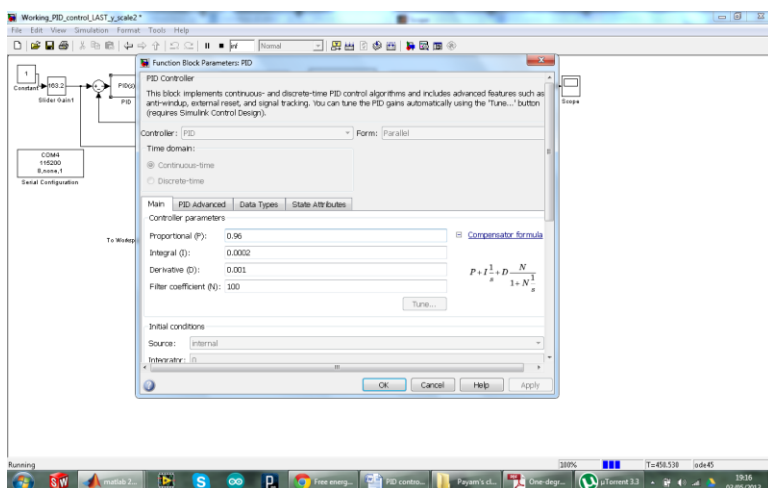


Figure 20. PID values fine-tuned

its counterweight is crucial. An advice to anyone attempting to construct a similar model would be to implement adjustable weights, since unforeseen changes might affect the overall weight of the system.

Furthermore, the freedom of movement the lever arm has is also important. A wider range of movement provides the system with better analysis. In addition, the angular position sensor should be able to detect the slightest variation of angle, since a fast response to this movement is necessary to a good operating system. Moreover, the size of the structure in relation to the motor and the power supply used are so important, that they can represent the failure or success of a functional system. Problems regarding the power source have been discussed above, and to summarise, in order to tune the PID control

values, a stable power supply providing constant amperage is imperative to the functionality of the system.

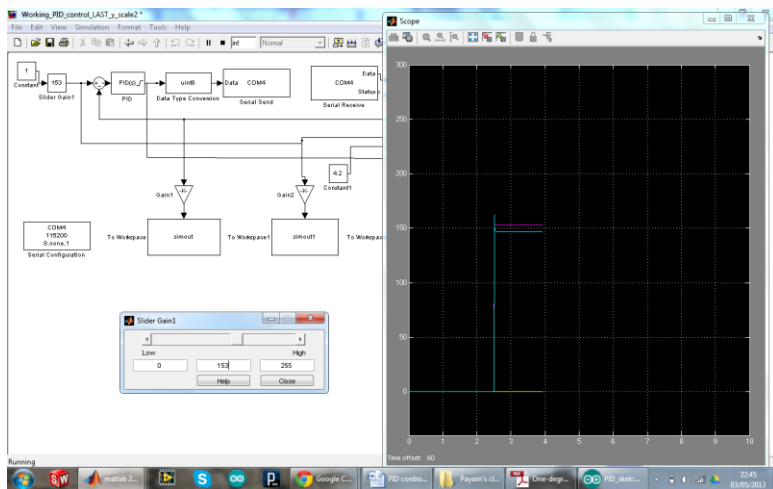


Figure 21. Good system response with desirable overshoot

system to a minimum, such as reducing the end-of-code delay in Arduino and implementing the fastest possible serial communication.

Through our research, it has been physically proven that these PID parameters make our system stable (Fig. 21), even in the presence of a small external force applied to it. It is important to mention, though, that these external forces have to be in a certain range since a too large disturbance in magnitude could overcome the job done by PID controller resulting in an indefinitely unstable system.

Since this project makes use of a real-time communication, the speed of this communication plays a strong part in it. All care has been taken to reduce the inevitable lag of the