CRANFIELD UNIVERSITY

Joris René Duran

Flight Desk Control Demonstrator

School of Aerospace, Transport and Manufacturing
MSc Aerospace Vehicle Design

Msc
Academic Year 2017 - 2018

Supervisor: Dr James F Whidborne
August 2018

CRANFIELD UNIVERSITY

School of Aerospace, Transport and Manufacturing
MSc Aerospace Vehicle Design

Msc

Academic Year 2017 - 2018

Joris René Duran

Flight Desk Control Demonstrator

Supervisor: Dr James F Whidborne
August 2018

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science in Aerospace Vehicle Design

This thesis covers the part of the assessment concerned with the IRP Project. Readers must be aware that the work contained within is not necessarily 100% correct and caution should be exercised if the thesis or the data it contains is being used for future work. If in doubt, please refer to the supervisor named in the thesis, or the Department of Aerospace Technology.

Abstract

The aim of a control system is to obtain a desired output response according to an input command. This can be achieved by knowing a model of the system with an open-loop control. However, an accurate model can be difficult to obtain. With a closed-loop control system, the controller determines the input signal of the process by using the measurement of the output. The most used method in the industry world involves PID correction.

The concept of feedback control and the choice of the three gains (Proportional, Integrator, Derivative) for a simple PID controller can be quite hard for students to conceptualize and understand their effectiveness. The aim of this project is to develop a simple feedback system for aerospace students to understand the nature of feedback control, the choice and the influence of the PID terms. The system consists of a demonstrator for the control of the pitch angle of a simple aerofoil by means of a regulated flap.

This document focuses on the process to design a fully working demonstrator including the design of the demonstrator, its building and the programming of the GUI (Graphical User Interface). The first step is to create an aerodynamic model of the system.

Once a reliable model is obtained, a structural layout is suggested, based on existing wind tunnel design. The wind tunnel design is critical because the geometry has a direct impact on the loads acting on the aerofoil and it must satisfy aerodynamic requirements. The wind tunnel must create favourable aerodynamic conditions to make an easier control of the aerofoil by its flap. Then, the demonstrator is built using laser cutting and 3D printing.

The PID controller is implemented into an Arduino board programmed in C++ connected via Bluetooth to the GUI on a computer programmed in JAVA. It is possible to plot and save the output of the demonstrator as well as send new settings to the controller.

The demonstrator will be assessed, and several PID settings are suggested.

Keywords: PID controller, trailing edge control surface, Arduino GUI, Pitot Tube, JAVA, C++

Acknowledgment

I would first like to thank my thesis advisor Dr James F WHIDBORNE of the SATM School of Engineering at Cranfield University for his assistance.

I would also like to thank Mr. Barry Walker for his help during the demonstrator building and for taking the time to machine the different parts.

I would like to sincerely acknowledge the PHD students Martin Alejandro CARRIZALES RODRIGUEZ and Alessandro PONTILLO for their support and cooperation, it was a real pleasure to work with them, I am gratefully indebted for their valuable comments on this project.

Finally, I must express my profound gratitude to my parents, my brother Geoffrey DURAN and Cécile REINERI for providing me with unfailing support and continuous encouragement throughout this year of study. This accomplishment would not have been possible without them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

No table of figures entries found.

NOMENCLATURE

g	Gravitational constant $9.81m.s^2$
t	Time
$e(t)$	Error between the system output and the command
$r(t)$	Command
$y(t)$	System output
$u(t)$	Modified error thanks to a controller
K_p	Proportional coefficient
K_i/T_i	Integral coefficient (parallel/ideal form)
K_d/T_d	Derivative coefficient (parallel/ideal form)
$C(s)$	PID controller
N	Derivative filter divider coefficient
f_c	cut-off frequency at $-3D_B$
T	Sampling time
K_u	Ultimate proportional coefficient (system unstable/marginal stable)
T_u	Oscillating period when K_u is applied
a	Acceleration
D	Diameter
F	Force
R	Electric resistance
V	Velocity
S	Surface
A	Area
L	Length
p	Pressure
ρ	Density
λ	Friction factor from the Moody's diagram
R_e	Reynold's number
μ	Air viscosity
v	Air velocity
c	Aerofoil's chord length
M_a	Mach number
$(a_1)_T$	Inviscid lift-curve slope
$(a_1)_v$	Viscous lift-curve slope
C_l	Lift coefficient
α	Angle of attack

δ	Flap deflection
$(a_2)_v$	Rate of change of lift coefficient with control deflection in compressible flow
m_0	Rate of change of pitching moment coefficient with a plain control deflection in compressible flow
I	Aerofoil assembly inertia
υ	Viscosity coefficient
o_s	Overshoot
ξ	Damping factor
t_p	Peak time
t_s	Settling time
t_r	Rise time
ω_n	Natural damped frequency
ω_d	Natural frequency
s	Complex number
C	Correction constant
e_{ss}	Steady-state error
$T(s)$	Transfer function
$S(s)$	Sensitivity function
G	Gain
φ	Phase

LIST OF ABREVIATIONS

V	Velocity
IRP	Individual Research Project
HMI	Human Machine Interface
GUI	Graphical User Interface
ZN	Ziegler-Nichols
CC	Cohen Coon
CHR	Chien, Hrones and Reswick
ITAE	Integral of Time multiplied by Absolute Errors
AMIGO	Approximate M-constrained integral gain Optimisation
ADC	Analog Digital Converter
MEMS	Micro-Mechanical Systems
IMU	Inertial Measure Units
ESC	Electronic Speed Controller
PWM	Pulse Width Modulation
CFD	Computational Fluid Dynamics
CAD	Computational Aided Design
AoA	Angle of Attack
CG	Centre of Gravity
PDB	Power Distributor Board
IDE	Integrated Development Environment
UML	Unified Modeling Language
XML	Extensible Markup Language
SISO	Single Input Single Output
OpAmp	Operational Amplifier

1 Introduction

“The aim of a control system is to obtain a desired response for a given system. This can be done with an open-loop control system, where the controller determines the input signal to the process on the basis of the reference signal only, or with a closed-loop control system, where the controller determines the input signal to the process by using also the measurement of the output” [1]. PID correction is widely used in the industry. However, the concept of feedback control and the choice of the three gains (Proportional, Integrator, Derivative) for a simple PID controllers can be quite hard for students to conceptualize and to understand their effectiveness.

The aim of this project is to develop a simple feedback system for aerospace students to understand the nature of feedback control and the choice and influence of the PID tuning terms. The system will consist of a demonstrator for the control of the pitch of a simple aerofoil by means of a regulated flap.

The main objectives consist of:

- Create an aerodynamic model of the system
- Build the benchtop
- Program the GUI interface

The main tasks are:

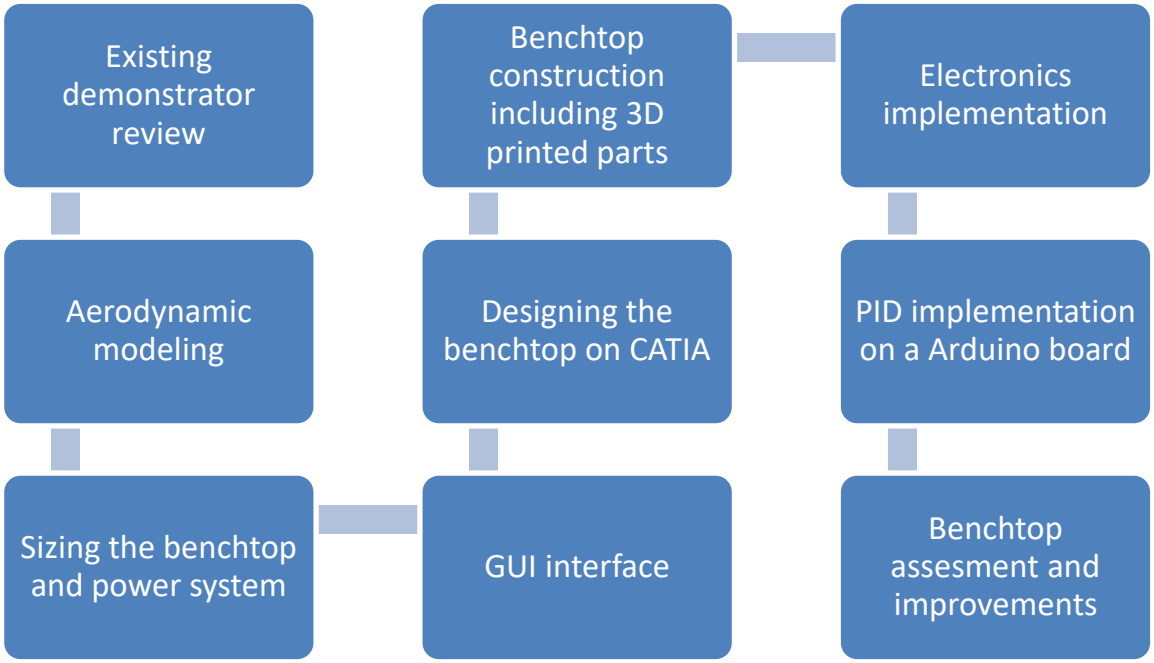
- Reviewing available demonstrator systems aimed at aerospace students
- Create a simple aerodynamic model of the system (wing + one trailing edge flap) and a related simulation on Matlab
- Designing the system experimental benchtop
- Designing the electronics to implement a PID controller on an Arduino® board and interface to the experimental benchtop
- Designing a GUI interface with an Arduino® board in Java, where it is possible to test different PID configurations, visualize responses and compare PID performances
- Test and evaluate system

1.1 Initial assumptions

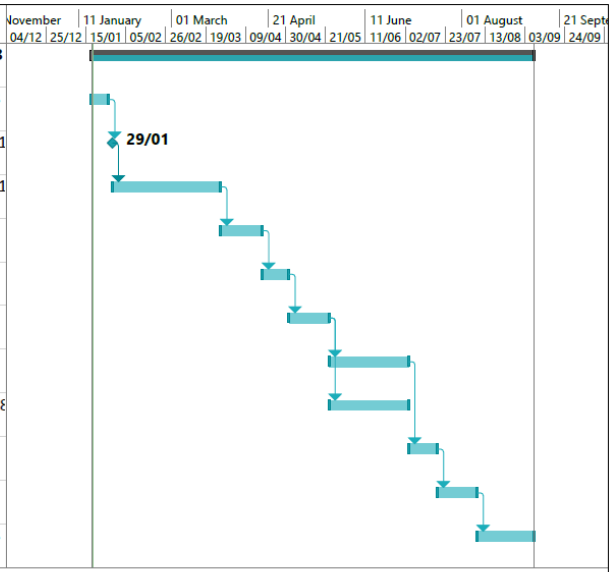
The following assumptions are established to make calculation feasible and due to some missing data:

- Rigid body is assumed for the entire structure, thus no aeroelasticity effects are considered
- Vibration effects are neglected
- Mechanical friction inside bearings is negligible
- The aerofoil is width enough to neglect side effects
- The time response of the servomotor is negligible compare to the time response of the system
- The inlet air flow is supposed to be uniform
- PID settings focus on reference tracking

1.2 Project Organisation



ID	Task Mode	Task Name	Duration	Start	Finish
1		**IRP**	**167 days**	**Thu 18/01/1**	**Fri 07/09/18**
2		First Research	7 days	Thu 18/01/1	Fri 26/01/18
3		IRP Abstract	0 days	Mon 29/01/1	Mon 29/01/1
4		Literature review	41 days	Mon 29/01/1	Mon 26/03/1
5		Simple aerodynamic modeling	16 days	Tue 27/03/18	Tue 17/04/18
6		Designing the benchtop on CATIA	10 days	Wed 18/04/18	Tue 01/05/18
7		Sizing the benchtop and power supply	15 days	Wed 02/05/18	Tue 22/05/18
8		Benchtop construction including 3D printing	30 days	Wed 23/05/18	Tue 03/07/18
9		GUI interface	30 days	Wed 23/05/1	Tue 03/07/18
10		Electronics implementation	11 days	Wed 04/07/18	Wed 18/07/18
11		PID implementation on Arduino board	15 days	Thu 19/07/18	Wed 08/08/18
12		Benchtop assesment and improvements	22 days	Thu 09/08/18	Fri 07/09/18



Project: IRP Gantt Chart
Date: Fri 19/01/18

Task		Inactive Summary		External Tasks	
Split		Manual Task		External Milestone	
Milestone		Duration-only		Deadline	
Summary		Manual Summary Rollup		Progress	
Project Summary		Manual Summary		Manual Progress	
Inactive Task		Start-only			
Inactive Milestone		Finish-only			

Page 1

1.3 Constraints

Some project constraints are present during the realisation of the tasks:

- Geometry: The wind tunnel must be transportable and feasible
- Data input: Many of the tasks can be done in parallel, especially the programming part
- Calculation: The calculation method must be accurate enough and easy to compute
- Material: Some parts must be purchased in Asia and the delivery time can be long
- Budget: The expenditure must be monitoring all the project long

The code implemented into the microcontroller cannot exceed the microcontroller capacity in terms of size and calculation.

The demonstrator must be flexible for future uses.

The servomotor and sensor dynamics are neglected for the sake of the controller design.
The decision is made by the author to use a C++ code on Arduino® to implement the PID correction. The GUI programmed in JAVA must be user-friendly. VBA code on Excel© are used to compute the loads and plot diagrams. The code was widely used to carry out the work presented in this thesis and make the calculation faster and flexible.

2 Basics of PID control

2.1 Structure of PID control

PID control is used to keep a process variable as close as possible to a desired value into a closed-loop control system [1]. A system with a feedback can be depicted as follow:

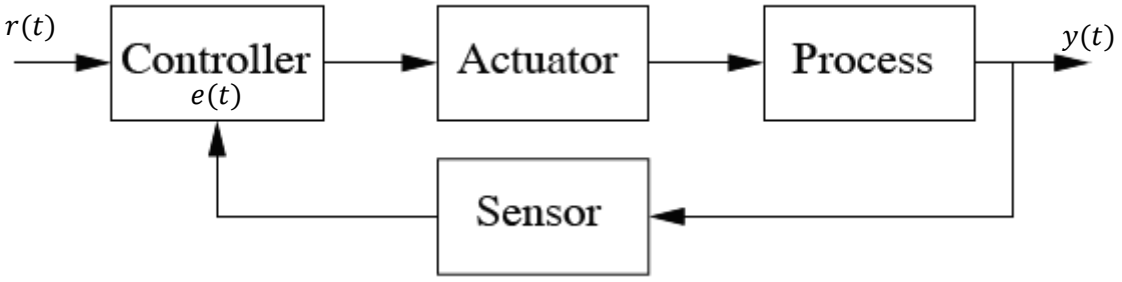


Figure 2-1 Typical components of a feedback control loop [1]

In this configuration, a command is sent to the microcontroller that controls the actuator. The system state changes according to the process model and the evolution of the system is monitored by the sensor. The microcontroller uses the difference between the command and the current state of the system to adjust the actuator. It is at this moment the PID is involved.

$$e(t) = r(t) - y(t) \tag{2-1}$$

A PID control is the application of a combination of three terms.

2.2 Proportional action

The first term is the Proportional. It gives a proportional action according to the current control error:

$$u(t) = K_p e(t) = K_p(r(t) - y(t)) \tag{2-2}$$

Where K_p is the proportional gain.

The proportional action allows a better precision and time response. It is possible to use only a pure proportional controller but it cannot suppress a steady-state error and too high values of K_p can lead to oscillations [1].

2.3 Integral action

The second term is the Integral. It gives an integration of the control error by the time and its action is based on the past values of the control error.

$$u(t) = K_i \int_0^t e(t)d\tau \tag{2-3}$$

Where K_i is the integral gain.

The integral term leads to a suppression of the steady-state-error. This acts like an automatic reset. A proportional action in conjunction to an integral action solves the main issue of a pure proportional control. However, the phenomenon of integrator windup appears and needs to be considered into the microcontroller. Indeed, the integration action has no limit and so can saturate the microcontroller or reach the actuator limits. The system acts in open-loop and the control error decreases slower than expected. Then, the system can reach the setpoint but the microcontroller is still saturate this leads to a generally large overshoot and settling time [1]. Several options exist whether the nonlinearity of the control is considered or not:

- "by limiting or smoothing the setpoint changes and/or by detuning the controller" [1] However, this method leads to a significant reduction in terms of performance
- Integrator clamping: the integral term is limited to a predefined value or switched off according certain conditions. Typically, when "the integration is stopped when the control variable saturates and the control error and the control variable have the same sign"[1]. In case of preloading, two predefined values of the integral action limit the integral action. Those two values need to be carefully chosen.
- Back calculation: "consists of recomputing the integral term when the controller saturates" [1]. Due to its greater complexity for the microcontroller, this method is not considered for the flight desk control demonstrator. This "methodology provides the capability to influence the transient response through the tuning of the tracking time constant" [1].
- A combination of the method depicted above

2.4 Derivative action

The last term is the Derivative, action is based on the predicted future values of the control error.

$$u(t) = K_d \frac{de(t)}{dt} \tag{2-4}$$

Where K_d is the derivative gain.

The derivative action is called anticipatory control, or rate action, or pre-act due to its capacity to predict an incorrect trend of the control error. By counteracting it, it has a great potentiality to improve the control performance. However, the derivative action amplifies the measured noise of the manipulated variable. This can be solved by simply filtering the derivative action thanks to a low-pass filter (at least first order). This is another parameter to tune because it has an impact on the control performance [1].

2.5 PID structure

The combination of the three actions can be done in the ideal (non-interacting) or serie (interacting) or in parallel form. The parallel form is preferred because it is more concise and easier to tune [1]. The PID structure in Laplace notation with the derivative action's low pass filter:

Ideal form:

$$C(s) = K_p \left(1 + \frac{1}{T_i s} + \frac{T_d s}{\frac{T_d}{N} s + 1} \right) \tag{2-5}$$

Serie form

$$C(s) = K_p \left(1 + \frac{1}{T_i s} \right) \left(\frac{T_d s + 1}{\frac{T_d}{N} s + 1} \right) \tag{2-6}$$

Parallel form

$$C(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{\frac{K_d}{N} s + 1} \tag{2-7}$$

Where N is between 8 and 16 in the majority of practical case [1] and:

$$K_d = K_p T_d \tag{2-8}$$

$$K_i = \frac{K_p}{T_i} \tag{2-9}$$

2.6 Setpoint Weighting

Achieving great performances in setpoint following and load disturbance rejection is difficult to obtain with a PID control. For example, a great load disturbance rejection can be achieved thanks to a high-gain controller but such a configuration leads to oscillating setpoint step response. A solution is a two degrees of freedom control architecture: a combination of feedforward and feedback control laws [1]. In the case of Weighting, the setpoint signal is achieved as follow:

$$C(s) = K_p\left(\beta + \frac{1}{T_i s} + \frac{T_d s}{\frac{T_d}{N}s + 1}\right) \tag{2-10}$$

Where β is a value between 0 and 1.

This aspect of the PID control is not investigated in this document. However, a future research on this subject using the demonstrator can be led.

2.7 Microcontroller implementation

The previous work assumes a continuous time that is not correct in the case of microcontroller where the time is discretised. The PID control in discretised time for a parallel form becomes:

$$u(t_k) = K_p e(t_k) + K_i \Delta t \sum_{i=1}^{k} e(t_i) + \frac{K_d}{\Delta t}(e(t_k) - e(t_{k-1})) \tag{2-11}$$

This formula is implemented into the microcontroller of the flight desk control demonstrator. The windup is removed by clamping when the control variable saturates, and the control error and the control variable have the same sign. The output of the PID is also limited regarding the servomotor available range of angle.

Applying the z-transform to a first order low-pass filter in Laplace notation shows directly how to program such a filter digitally:

$$F(s) = \frac{1}{1 + \frac{s}{2\pi f_c}} \rightarrow F(z) = \frac{1 - e^{-2\pi f_c T}}{1 - z^{-1}e^{-2\pi f_c T}} \rightarrow y(t_k) = y(t_{k-1})e^{-2\pi f_c T} + x(t_k)(1 - e^{-2\pi f_c T}) \tag{2-12}$$

Where $f_c$is the cut-off at $-3D_B$, T the sampling time. This will be applied on the derivative term but also on the analogic data input read by the microcontroller.

2.8 PID tuning

A PID control is ruled by 3 major gains:

- K_p, proportional gain
- K_i, Integral gain
- K_d, derivative gain

The tuning of the PID parameters is a key point in the overall controller design. The tuning should be done according to the control specifications:

- Setpoint following
- Load disturbance rejection
- Robustness
- Life-span of the actuator
- Cost

Many practical tuning rules exist to help the operator to tune the controller.

2.8.1 Ziegler-Nichols (ZN) Method

It is the most popular method and it is based on the open-loop step response. It relies on two parameters L and T obtained by drawing a tangent line at the inflexion point as shown below [2]. K_u is the proportional ultimate gain at which the system becomes unstable and T_u is the corresponding oscillating period. The coefficient K_p, K_i and K_d from the PID parallel form without the derivative low-pass filter are calculated as follow:

Table 2-1 PID controller parameters in Ziegler-Nichols step response method [2]

Controller parameter	K_p	K_i	K_d
P Controller	$0.5K_u$		
PI Controller	$0.4K_u$	$\frac{0.5K_u}{T_u}$	
PID Controller	$0.6K_u$	$\frac{1.2K_u}{T_u}$	$0.075K_uT_u$

2.8.2 Cohen Coon (CC) Method

This method is a more complex version of the Ziegler-Nichols method. It is based on a first order system with a pure delay.

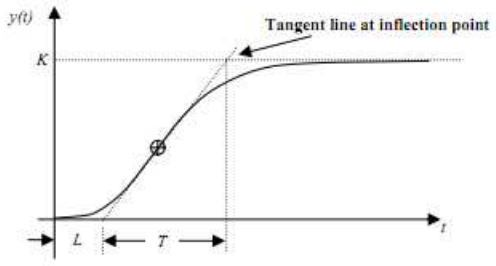


Figure 2-2 Response curve for Cohen Coon method [2]

The coefficient K_p, K_i and K_d from the PID parallel form without the derivative low-pass filter are calculated as follow:

Table 2-2 PID controller parameters in Cohen Coon step response method [2]

Controller parameter	K_p	K_i	K_d
PID Controller	$\frac{T}{K \times L}\left(\frac{16T+30}{12T}\right)$	$\frac{L\left(32+\frac{6L}{T}\right)}{13+\frac{8L}{T}}$	$\frac{4L}{11+\frac{2L}{T}}$

2.8.3 Chien, Hrones and Reswick (CHR) Method

This is a modified version of Ziegler-Nichols method and "provides a better way to select a compensator for process control applications. In process industry, controller parameters are often tuned according to CHR recommendation" [2]. The coefficients T, L and K are obtained regarding Figure 2-2.The coefficient K_p, T_i and T_d from the PID ideal form without the derivative low-pass filter are calculated as follow:

Table 2-3 PID controller parameters in CHR step response method [2]

Overshoot	0%			20%		
Controller parameter	K_p	T_i	T_d	K_p	T_i	T_d
P	$\frac{0.3T}{K \times L}$			$\frac{0.7T}{K \times L}$		
PI	$\frac{0.35T}{K \times L}$	$1.2T$		$\frac{0.6T}{K \times L}$	T	
PID	$\frac{0.6T}{K \times L}$	T	$0.5L$	$\frac{0.95T}{K \times L}$	$1.4T$	$0.47L$

2.8.4 ITAE and AMIGO tuning Methods

This method is a more complex version of the Ziegler-Nichols method. The coefficients T, L and K are obtained regarding Figure 2-2. The coefficient K_p, T_i and T_d from the PID parallel form without the derivative low-pass filter are calculated as follow:

Table 2-4 PID controller parameters in ITAE step response method [2]

Controller parameter	K_p	T_i	T_d
PID Controller ITAE	$\frac{0.965}{K\left(\frac{T}{L}\right)^{0.855}}$	$\frac{T}{\left(0.796-\frac{0.147L}{T}\right)}$	$0.308T\left(\frac{L}{T}\right)^{0.929}$
PID Controller AMIGO	$\frac{1}{K}\left(0.2+\frac{0.45T}{L}\right)$	$\frac{0.4L+0.8T}{L+0.1T}L$	$\frac{0.4L\times T}{0.3L+T}$

2.9 Conclusion

This section has covered fundamental concepts of PID controllers. Several tuning methods have been depicted. For the sake of the flight desk control demonstrator only the equation from 2.7 is used. The windup is removed by clamping. The output of the PID is also limited regarding the servomotor available range of angle. No setpoint weighting is implemented at this point. Different PID configurations and tunings are tried during the demonstrator assessment. To conclude, the following tables summarised the P, I, D actions and the advantages and drawbacks of each correction.

Table 2-5 P, I, D actions [2]

Close Loop Response	Overshoot	Settling Time	Steady-State Error	Rise Time	Stability
Increasing K_p	Increase	Small Increase	Decrease	Decrease	Decrease
Increasing K_i	Increase	Increase	Large Decrease	Decrease	Increase
Increasing K_d	Increase	Decrease	Minor Change	Minor Change	Decrease

Table 2-6 Correction comparison [1]

Close Loop Response	Advantages	Drawbacks
P	Simple	Steady-state error, oscillations
PI	Simple, widely used, no steady-state error	Efficient for first order process, windup
PID	Significant possible improvements	Filter, windup, complex

3 Main variable sensor investigation

The aim of this section is to review the options to measure the angle of attack of the aerofoil. The solution must respect some criteria:

- Easy to implement, the solution does not make the system hyperstatic
- Easy to calibrate and to guaranty the measures
- Does not disturb the system or at least the disturbance is easy to quantify
- Can be used with an Arduino® board
- Cost effective

3.1 Potentiometer

An easy way to measure the angle of attack of the aerofoil is fixing a potentiometer on the rotational axis of the aerofoil.

A potentiometer can be defined as an electro-mechanical transducer converting a rotary or a linear motion into a change of electronical resistance [3].

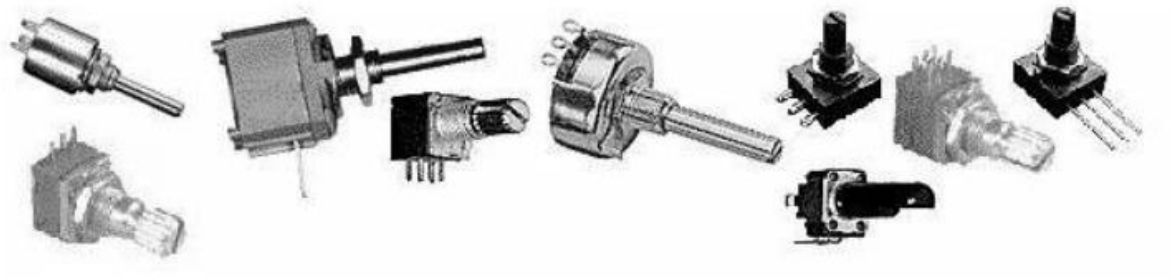
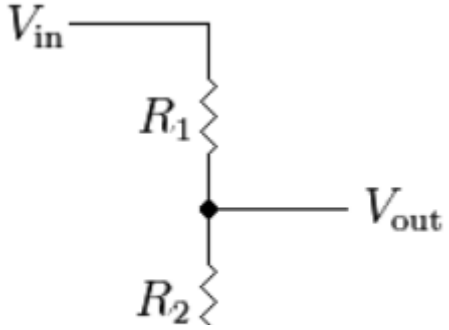


Figure 3-1 Potentiometer examples [3]

It is then possible to measure a tension with a microcontroller's analogic input by applying a tension divider.



$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in} \qquad (3\text{-}1)$$

Figure 3-2 Tension divider schema

The potentiometer total resistance must be great enough to limit the current through the component, but a too great value leads to high interferences sensibility. A value of $10k\Omega$ is typical.

The tension V_{out} is read by the microcontroller thanks to an ADC. The ADC of an Arduino® is generally $10\ bits$ [4]. It means 1024 possible values. With the typical $1.1V$ internal reference [4], the precision is equal to $1.07mV$. If the rotation is $74.8°$ from an extreme to the other, the precision in degree is $0.075°$. The potentiometer is assumed to be linear.

The variation of resistance is created by sliding a piece of metal on a conductive material. The conductive material is crucial, and several types exist:

- Carbon
- Cermet
- Conductive plastic
- Wire wound

The conductive plastic is particularly recommended for this project because it offers high quality potentiometers in terms of mechanical friction, life-span, noise and precision [3].

A good candidate for the flight desk control demonstrator is a VISHAY® Model 157.

Figure 3-3 VISHAY® Model 157 [5]

However, using a potentiometer leads to a hyperstatic configuration and creates a torque due to the friction inside the potentiometer. Even if the value is available in the datasheet, the hyperstatic configuration creates non-linearity in the model and new constraints. The hyperstatic configuration can be removed using a flexible joint between the potentiometer and the aerofoil axis.

3.2 Hall effect sensor

The hall effect is a contactless technology and should be ideal for the sake of the flight desk control demonstrator. This phenomenon was discovered by Dr. Edwin Hall in 1879 and greatly develop during the1950s with the advent of semiconducting material. “The Hall element is constructed from a thin sheet of conductive material with output connections perpendicular to the direction of current flow. When subjected to a magnetic field, it responds with an output voltage proportional to the magnetic field strength” [6]. It is a transducer. As the output voltage is about μV, additional electronics is needed to achieve a useful signal. The combination of a Hall element and the associated electronics forms a Hall effect sensor [6]. This technology offers many advantages:

- Contactless and no moving part
- Life-span
- Works even in stationary input (zero speed)
- Great repeatability

The Hall effect principle relies on a potential difference across the output generated by the Lorentz force disturbing the current distribution through a thin metal sheet when a perpendicular magnetic field is present [6].

Tension due to Lorentz Force:

$$V = I \times B \tag{3-2}$$

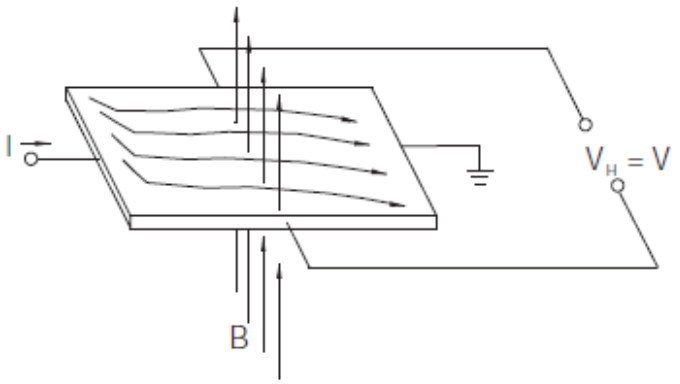


Figure 3-4 Hall effect principle [6]

An easy way to use a Hall element is as a ratiometric linear sensor. “The ratiometric output voltage is set by the supply voltage and varies in proportion to the strength of the magnetic field” [6]. An analogic Hall effect sensor is generally connected to an operational amplifier with an internal pull-down resistor. The value of the pull-down resistor is such that the current rating of the analogic output is low [6].

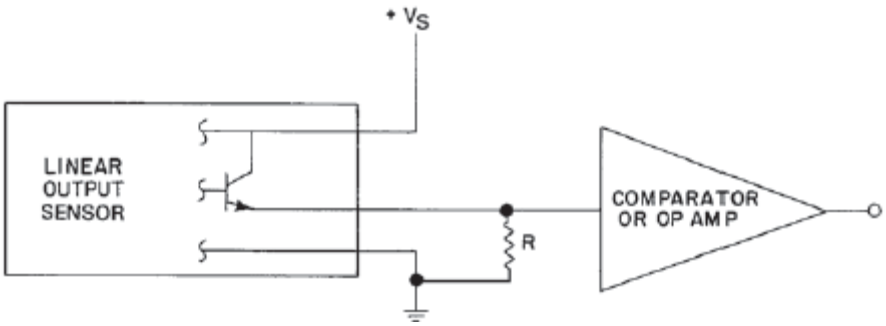


Figure 3-5 Analogic Hall effect sensor [6]

The idea chosen for measuring the angle of the system is to use a bipolar slide-by mode with a ring magnet. “A ring magnet is a disk shaped piece of magnetic material with pole pairs magnetized around its circumference” [6].With a two pole pairs ring magnet the rotational motion results in a sine wave shaped. For a small range of degree, typically between $\pm 30°$, the results can be assimilated to a linear function.

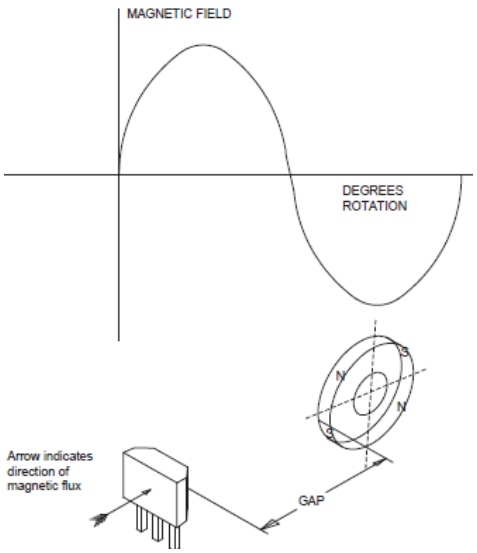


Figure 3-6 Bipolar slide-by mode with a ring magnet [6]

This solution removed the hyperstatic problem. A good candidate can be the SS495A1* High Accuracy from Honeywell®. The precision is worse than the previous potentiometer. However, the main issues are to guaranty the calibration and the geometry of the final assembly. The other issue is from the possible noise.

Figure 3-7 SS495A1 [7]

3.3 MPU6050 gyroscopes and accelerometers

Accelerometers belong to the category of Micro-Mechanical Systems (MEMS). They are devices able to measure the acceleration. The acceleration measurement is generally achieved by capacitive plates: half of them are fixed while the others are attached to springs. Thus, this system moves internally according to the forces acting on the accelerometer and the acceleration can be determined from the change of capacitance. Another but similar way to measure the acceleration inside MEMS is using piezoelectric materials. The piezoelectric material creates a potential difference when its crystal structures are deformed [8].

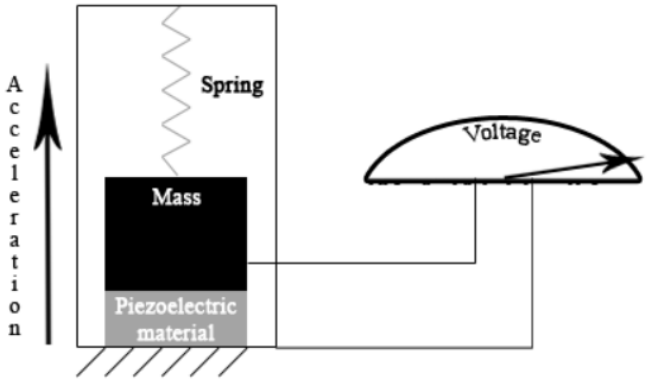
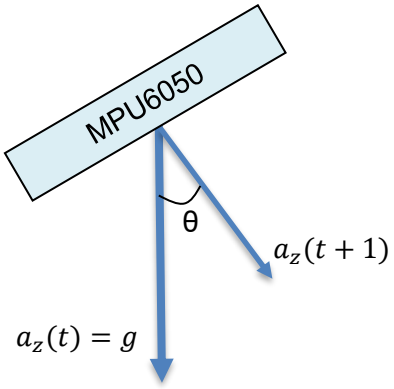


Figure 3-8 Acceleration measuring by piezoelectric material [8]

It is possible to get the angular velocity by integrating the acceleration with time. By integrating once again, it is possible to get the angular position.

The downward gravitational acceleration on Earth can be assumed to be constant and equals to $9.81 m.s^{-2}$. It is also possible to get directly the angular position using trigonometry.



$$\theta = \operatorname{atan}\left(\frac{a_z(t+1)}{g}\right) \qquad (3\text{-}3)$$

Figure 3-9 Acceleration due to the Earth gravitational force on z axis

An IMU such as the MPU6050 is an Inertial Measure Units combining accelerometers, gyroscopes and sometimes, magnetometers. The gyroscopes (angular velocity) values are from the accelerometers.

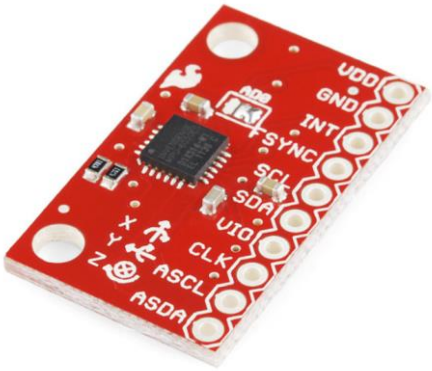


Figure 3-10 MPU6050 [8]

The MPU6050 combined a 3-axis accelerometer and a 3-axis gyroscope. It can be power by a $3.3V$ power supply and it is possible to communicate with it using I2C or SPI. So it is not an analogic device and a better precision is expected (Linear acceleration sensitivity $0.1°.s^{-1}.g^{-1}$) [8].

Once calibrated, the MPU6050 is easy to implement and every adjustment can be done from the program inside the microcontroller. However, the gyroscopes values drift due to the integration and they can't be used to get the angular position. A gyroscope gives quick and good results only for a short time period. Because of the high noise, values from the accelerometer can't be used as well. An accelerometer is very precise and reliable, but the data need to be filtered.

A solution is to use a sensor fusion algorithm such as a Kalman filter. This kind of filter tries to minimise the estimation error covariance matrix by means of stochastic filtering and needs high computational performances [9]. Another solution is a complementary filter. It "allows fusion of independent measurements of the same signal with different spectral characteristics" [9]. The objective is to use the quick results from the gyroscopes and removes the drift phenomenon by using the data from the accelerometers.

The complementary filter principle can be depicted as follow considering the measurement from the accelerometer x with a disturbance of mostly high frequency μ_1 and the measurement from the gyroscope x with a disturbance of mostly low frequency μ_2.

$$y_1 = x + \mu_1 \quad (3\text{-}4) \qquad y_2 = x + \mu_2 \quad (3\text{-}5)$$

Assuming two transfer functions $F_1(s)$ and $F_2(s)$ where $F_1(s)$ is low-pass filter, $F_2(s)$ is a high-pass filter and:

$$F_1(s) + F_2(s) = 1 \quad (3\text{-}6)$$

The final measurement $\hat{X}(s)$ is:

$$\hat{X}(s) = F_1(s)Y_1(s) + F_2(s)Y_2(s) = X(s) + F_1(s)\mu_1 + F_2(s)\mu_2 \quad (3\text{-}7)$$

Where the real value $X(s)$ is not deteriorated but the disturbances are filtered. A "complementary filters are especially well suited for fusing low bandwidth position measurements with high bandwidth rate measurements" [9].

Typical values $F_1(s) = 0.98$ and $F_2(s) = 0.02$ are commonly used value with a MPU6050.

Finally, using a MPU6050 removes the hyperstatic issue, the implementation and reliability issues. However, the code inside the microcontroller is a bit more complex and the sensor still must be calibrated. Another issue comes from the wires to connect the sensor. In fact, they will hang down and disturb the system.

3.4 Conclusion

The aim of this section has been to determine the best technology and sensor to measure the pitch angle of the aerofoil regarding the constraints in terms of simplicity, computational performances and reliability.

Table 3-1 Sensor and technology comparison

	Precision	Hardware complexity	Software complexity	Noise
VISHAY Model 157	$0.075°$	Low but Hyperstatic	Very Low	Medium without filter
S495A1	$\geq 0.075°$	Medium but accurate calibration impossible	Low	High
MPU6050	$> 0.1°$	Low	Calibration, I2C, Filter	Medium once filtered

According to this section the MPU6050 seems to be the best option because it is easy to implement, and all the difficulties can be treated in the code. It is also a very interesting option because of the notion involved (Kalman filter, I2C, complementary filter). However, by adding a flexible joint between the potentiometer and the pivot of the aerofoil, the hyperstatic issue is removed and the potentiometer becomes the best solution. An additional digital filter, see 2.42.7 can be implemented at the potentiometer output to smooth the results.

4 Demonstrator review

This section aims to give example of previous desk demonstrator showing PID actions and focuses on the configuration of each demonstrator.

- How the control is achieved?
- How the user can interact with the system?
- What are the possibilities with such a demonstrator?

4.1 Inverted pendulum

An interesting way to demonstrate the effectiveness of PID controls can be achieved using a basic inverted pendulum mounted to a carriage. The gravitational force tends to attract a mass to the ground. The system is unstable. The aim of the system is to control the carriage motion to maintain and balance the mass at a certain angular position. The dynamics is easy to model and compute. This control problem is similar to those involved in rocket or missile stabilisation [10].

Figure 4-1 Inverted Pendulum [10]

The stability is achieved by the carriage motion. It must always move to balance the pendulum. On the system shown on Figure 4-1, the carriage is controlled by a DC motor connected to an integral tachometer, the angle of the rod is measured by the mean of a potentiometer. The position of the carriage is tracked by a multiturn potentiometer. A control console allows the user to connect the components to use the demonstrator in analogic or digital mode and to adjust the different gains of the controller.

This system is 25 years olds (1993) and the technology to control a system are far more advanced and cheaper. The system has no GUI and the possibility to reuse the demonstrator for another study is limited.

4.2 2 DOF Helicopter

Commercial demonstrators exist and the 2 DOF Helicopter from QANSER® is one of them. This demonstrator is composed of two rotors and allows to understand and test control laws for vehicles such as helicopter or underwater vehicle.

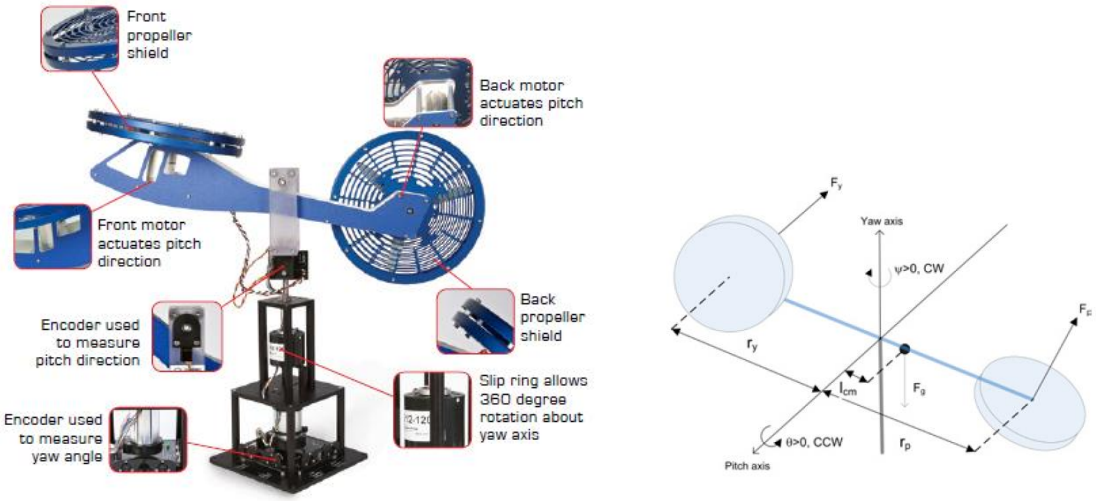


Figure 4-2 2 DOF Helicopter QANSER® [11]

The two rotors are actuated by two DC motors. It is a 2-DOF demonstrator because the rotors are perpendicularly mounted, and the moving part can rotate on the yaw and roll axis. The angle from each axis is measured thanks to high-resolution encoders. This configuration reminds the main rotor and the anti-torque tail rotor of a traditional helicopter. A real-time control software using Matlab/Simulink is provided [11].

This system is a far more advanced demonstrator and allows the user to perform high-level studies in many fields. The dynamic model is also more complex.

4.3 Conclusion

These two systems give examples how to think the flight desk control demonstrator and to create a reliable and durable benchtop. The author is particularly aware regarding the demonstrator safety, the sensors and the user-friendliness of the GUI. The flexibility and the robustness of the demonstrator are also critical. The demonstrator must come with precise documentation and easy to access information.

5 Wind tunnel design

The demonstrator is a wind tunnel to guaranty the validity of the aerodynamic model and isolate the system. In fact, the aim of the demonstrator is not to investigate aerodynamic subjects but to demonstrate the actions of a PID correction. So, a wind tunnel tends to create ideal aerodynamic conditions to only focus on the control.

5.1 Analogic power sizing

Figure 5-1 The motor with its propeller

Wind tunnels are usually huge devices, several meters long. The inside turbine must be powerful enough to create sometime hypersonic airflows. A first step in the design process is to size the turbine and the analogic power system. This gives a great idea of the wind tunnel size.

A constraint of the demonstrator is to be transportable and usable indoor. So, the benchtop must be as small as possible and the loud must be limited. The Author is particularly aware that the demonstrator will be used during class and must not make the students uncomfortable.

The motor used is a brushless motor. This technology is very common nowadays, especially for aero models because of their high efficiency and weight/power ratio. The chosen propulsion system is a T-MOTOR® MT2814 with a APC® $12" \times 3.8$ propeller.

Table 5-1 T-MOTOR® MT2814 with a APC® $12" \times 3.8$ at Design characteristic 65%

Kv	Tension	Max current	Max Power	Design Power	Design current	Design RPM	Design Thrust
$770Kv$	$11.1V$	$30A$	$430W$	$104.34W$	$9.4A$	$5091rpm$	$750g$

The brushless is controlled by a LYNXMOTION® $30A$ ESC with $5V$ BEC. The BEC is used to power the command circuit. The ESC is controlled using PWM signal from the Arduino. An imposing dissipater from an old PC is mounted with thermal paste directly on the aluminium dissipater of the ESC to avoid it to burn because of the high current drawn by the motor.

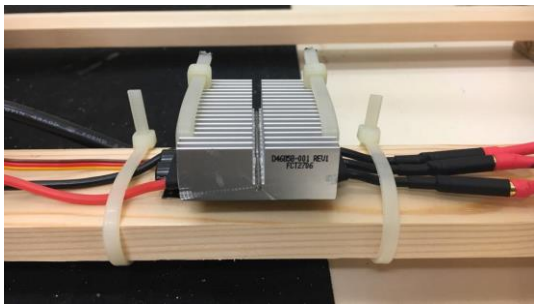


Figure 5-2 The ESC with dissipater

The power supply is an old modified $300W$ PC power supply. The useless wires are removed, only the most powerful $12V$ and $5V$ output are used. The $-12V$ is also used to power an operational amplifier. A switch and a power LED indicator are added to turn on the device. All the soldering and sections are secured with shrink tubes and glue. The power supply is powerful enough to power the motor and features great characteristics (short-circuit, overheating, current rate limit)

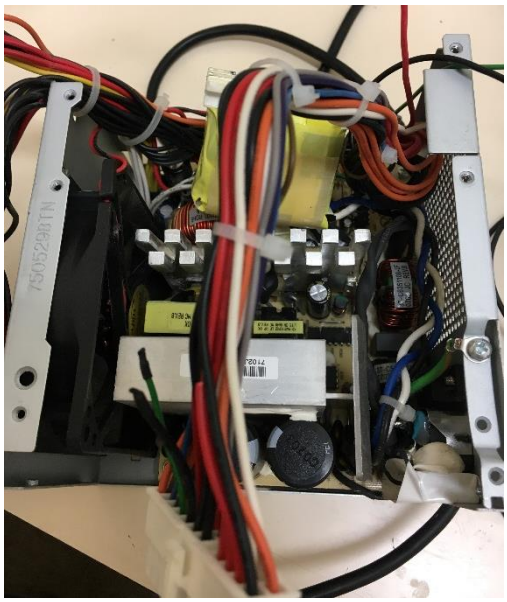


Figure 5-3 PC power supply

5.2 Wind Tunnel layout

5.3 General

The wind tunnel is designed following low speed wind tunnel for aerofoil aerodynamic analyses and micro wind turbine studies rules. The following method deals with closed-loop wind tunnel and it is adapted by the Author to fit with an open-loop design. Compare to a closed-loop design depicted in Figure 2-1, an open-loop wind tunnel is only composed of a nozzle inlet, test chamber and a diffuser outlet. The open-loop design is more affordable and easy to buy but generates high noise and the airflow quality can be a bit lower without additional screens than a closed-loop design [12].

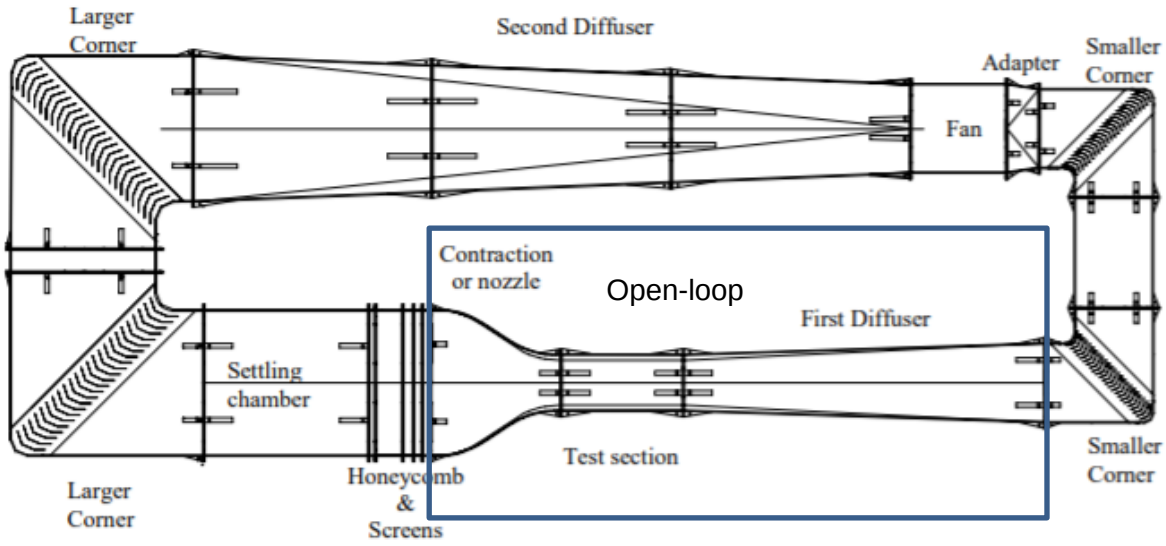


Figure 5-4 Closed-loop wind tunnel [12]

The design procedure is the following:

- Defining the test chamber section and desired flow velocity
- Designing the wind tunnel component using criteria
- Computing the wind tunnel components pressure losses
- Verifying the turbine defined previously match the airflow velocity desired inside the test chamber

The wind tunnel has a square section.

5.4 Test chamber geometry

The test chamber section is a $a = 25\ cm$ side square and the airflow velocity is $V_a = 10\ m.s^{-1}$. These are the most important criteria and they rule the wind tunnel geometry. The Hydraulic diameter D_h is given by:

$$D_h = 2\sqrt{\frac{A}{\pi}} \tag{5-1}$$

Where A is the test chamber section $D_h = 0.28m$.

The test chamber length is between 0.5 and 3 times the hydraulic diameter [12]. A value of 1.5 is chosen to keep the wind tunnel total length low. $L_{test\ chamber} = 0.42m$. The test chamber sharp edges are smoothed with glue. The airflow needs usually 0.5 time the hydraulic diameter to become uniform. A too long test chamber generally enlarge the boundary layer thickness [12].

5.5 Nozzle inlet geometry

The aim of the nozzle is to concentrate and accelerate the airflow following the Venturi's Law. For an incompressible flow:

$$S_1 V_1 = S_2 V_2 \tag{5-2}$$

Where S is the airflow section and V the airflow velocity. The nozzle outlet section is the inlet test chamber section. The Nozzle inlet section must be as large as possible to keep the losses down. The ratio between the inlet and the outlet cross section is generally between 6 and 10 [12]. The nozzle length chosen is $0.55m$ to get a ratio of 6.
The nozzle's silhouette is given by fifth order Bell-Metha polynomials [12]:

$$y = a\xi^5 + b\xi^4 + c\xi^3 + d\xi^2 + e\xi + f \tag{5-3}$$

Where $\xi = \frac{X}{L}$ and $0 \leq X \leq L$ and L is the nozzle length and y is the half cross-section side-length at X.

The Bell-Metha polynomial coefficients are imposed:

- $\xi = 0$ at $y = y_0$ where y_0 is the half cross-section side-length at the nozzle inlet
- $\xi = 1$ at $y = y_1$ where y_1 is the half cross-section side-length at the nozzle outlet
- $\frac{dy}{d\xi} = 0$ at $\xi = 0$
- $\frac{dy}{d\xi} = 0$ at $\xi = 1$
- $\frac{d^2y}{d\xi^2} = 0$ at $\xi = 0$
- $\frac{d^2y}{d\xi^2} = 0$ at $\xi = 1$

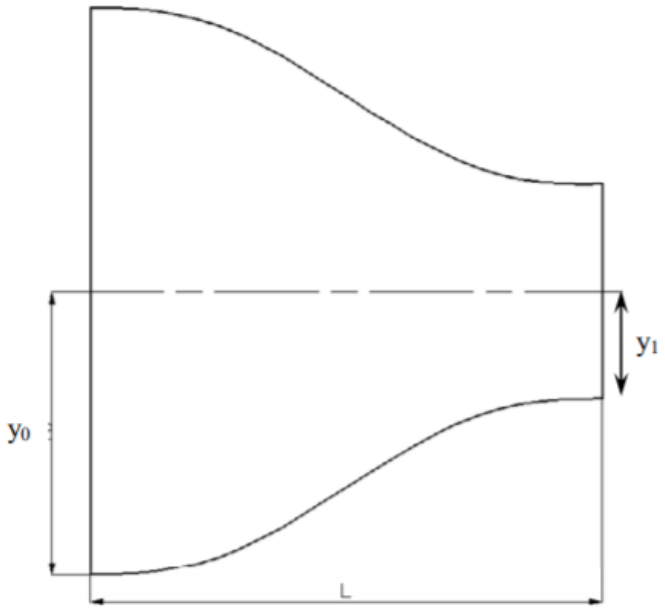


Figure 5-5 Nozzle shape [12]

y_0 is generally given by:

$$\frac{L}{2y_0} \cong 1 \qquad (5\text{-}4)$$

If this ratio is inferior to 0.667 experiments show an airflow detachment at the nozzle outlet and a value greater than 1.79 increases the boundary layer thickness [12]. This gives $y_0 = 0.31\ m$.

Table 5-2 Bell-Metha polynomial coefficients

a	b	c	d	e	f
-1.08	2.71	-1.81	0	0	0.31

The final shape is given below, the curve is approximated to a straight line to be easy to manufacture:

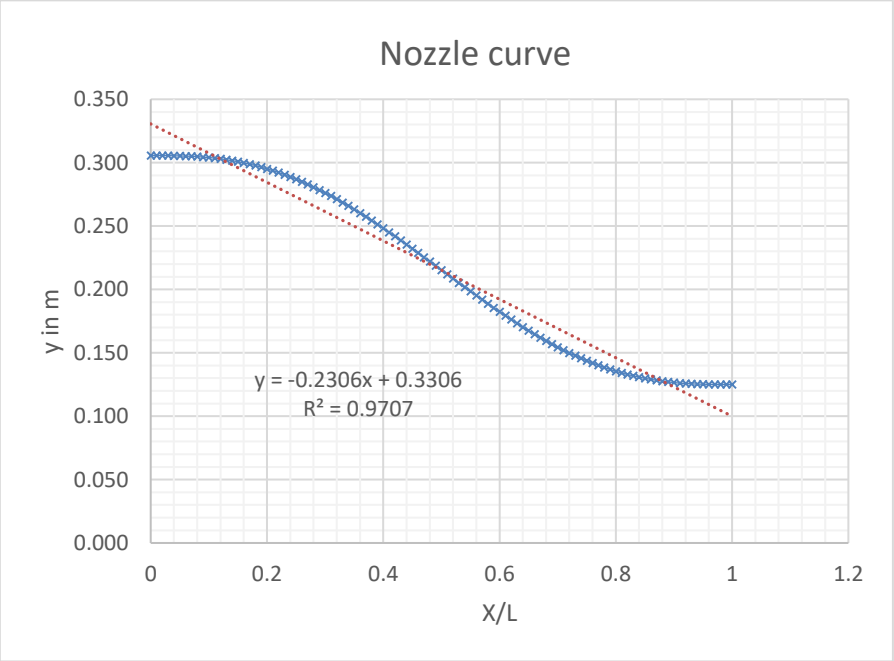


Figure 5-6 Nozzle curve

No screen or honey comb is used at the nozzle inlet to keep the design simple and to keep the losses down. The airflow is low enough to justify small turbulences.

5.6 Diffuser Geometry

The inlet cross-section is governed by the test chamber cross-section and the outlet by the fan dimension.$D_{fan} = 0.3\ m$. The outlet section is given by:

$$l = \frac{D_{fan}}{2}\sqrt{\pi} \tag{5-5}$$

$l = 0.27\ m$. The ratio between the inlet and the outlet is generally between 2 and 3. A greater ratio leads to irregular flow velocities and smaller value increases the wind tunnel dimensions. The length of the diffuser is $0.55\ m$ and the ratio is 1.17 which is acceptable for sake of the project.

The equivalent cone expansion angle is generally below $6°$ and given by [12]:

$$\theta = arctan\left(\frac{\sqrt{A_r - 1}}{2\left(\frac{L}{D_h}\right)}\right) \tag{5-6}$$

$\theta = 6°$ with the mentioned dimensions. The total length of the wind tunnel is $1.52m$.

5.7 Wind tunnel losses calculation

It is now possible to quantify the losses inside the wind tunnel. The aim is to combine these data with the pressure drop created by the turbine and check if the airflow velocity of $10\ m.s^{-1}$ is reached. The calculation is divided into:

- Regular losses through the wind tunnel due to air friction. The wind tunnel is assumed to be a constant-area section of $0.25\ m$ side-length (test chamber) for this calculation. The result is expected to be higher than the reality but still negligible compare to the screen losses.
- Singular losses due to the nozzle and the diffuser
- Singular losses through the screen, it is located at the outlet of the diffuser to protect the user from the spinning propeller. No other screen is required because of the low speed of the airflow and to keep the losses down.

Table 5-3 Air Characteristics

Viscosity μ	Density ρ	Velocity V
$1.85 \times 10^{-5}\ Kg.m^{-1}.s^{-1}$	$1.225\ Kg.m^{-3}$	$10.57\ m.s^{-1}$

The velocity in Table 5-3 is directly the results from this iterative work. A VBA code was used to make the process quicker, see 11Appendix B.

The regular pressure loss Δp_r is equal to:

$$\frac{\Delta p_r}{\rho} = \lambda \frac{L v^2}{2D} \tag{5-7}$$

Where L is the total length of the wind tunnel and D the equivalent diameter. The friction factor is determined with the Moody's diagram.

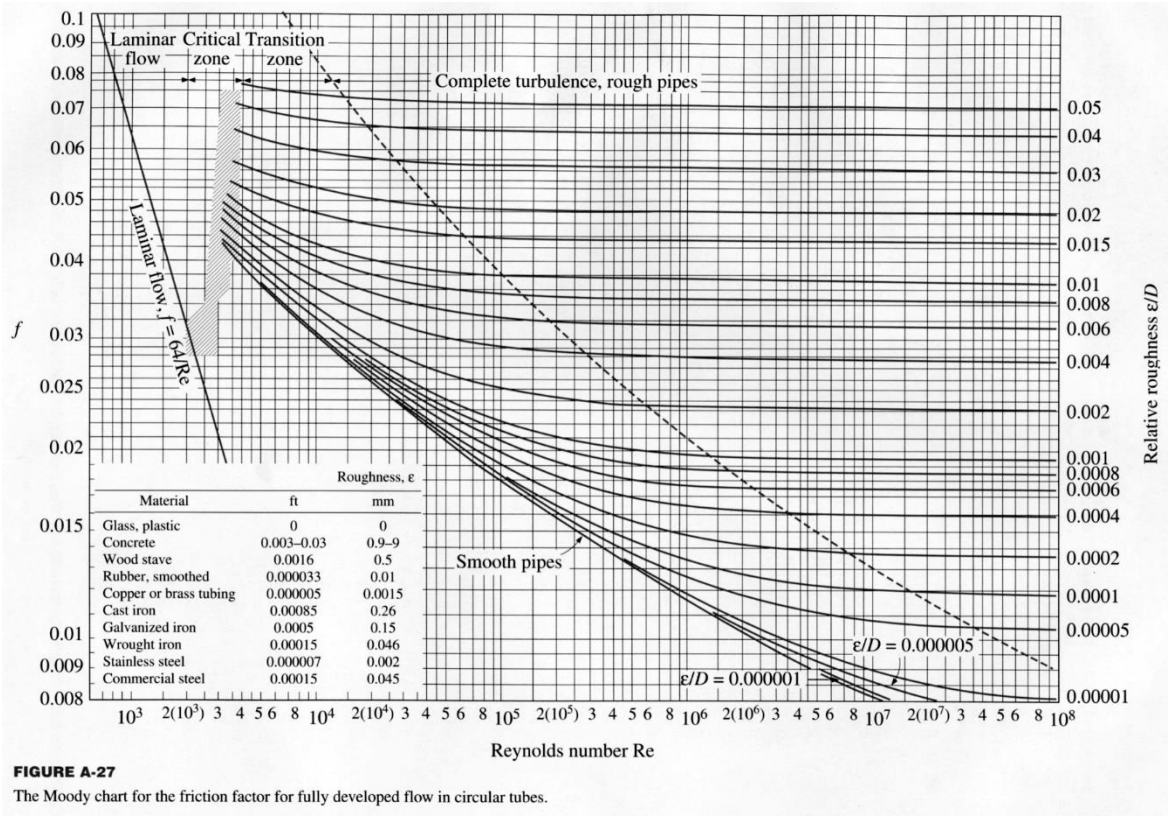


FIGURE A-27
The Moody chart for the friction factor for fully developed flow in circular tubes.

Figure 5-7 Moody's Diagram

The Reynold's number is given by:

$$R_e = \frac{\rho D v}{\mu} \tag{5-8}$$

$R_e = 175\,000$. For common plastic, the rugosity ε is equal to $0.0015mm$. The friction factor is then $\lambda = 0.017$ according to the Moody's diagram. The flow is fully turbulent. It now possible to get the regular loss through the wind tunnel $\Delta p_r = 7.08Pa$.

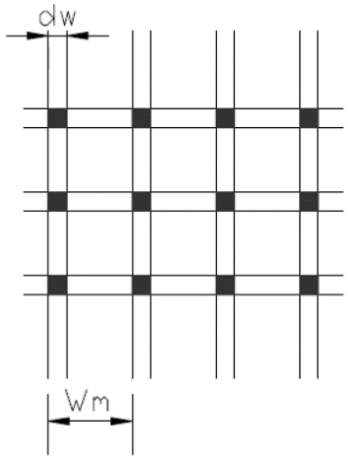
The singular losses Δp_s are given by the general equation:

$$\Delta p_s = K \frac{\rho v^2}{2} \tag{5-9}$$

Where K is a coefficient depending on the singularity. The calculation for each singularity is given in the table below:

Table 5-4 Singularity losses

	Nozzle [13]	Diffuser [13]	Screen [12]
K Formula	$\left(\frac{S_2}{S_c} - 1\right)^2 \sin(\alpha)$	$\left(\frac{S_1}{S_2} - 1\right)^2 \sin(\alpha)$	$\frac{1.3 S_o}{S_t} + \left(\frac{S_t}{S_o} - 1\right)^2$
K Calculation	$\frac{S_2}{S_c} = 1.1$ $\alpha = 0.47\ rad$	$\frac{S_1}{S_2} = 0.86$ $\alpha = 6°$	See below
K	0.004	0.002	0.59
Δp_s in Pa	2.99	0.15	40.4



A traditional porosity for a screen is between 0.58 and 0.8. For $d_w = 1\ mm$ and $W_m = 6\ mm$, the porosity is equal to 0.77, the open aera is equal to $S_o = (W_m - d_w)^2 = 25\ mm^2$ and the total area is $S_t = W_m^2 = 36\ mm^2$.

The total amount of loss through the wind tunnel is $\Delta p = 50.6 Pa$.

The next step is to check the airflow velocity inside the test chamber.

Figure 5-8 Screen mesh [12]

The pressure drops created by the turbine $\Delta p_{turbine}$ can be calculated as follow:

$$\Delta p_{turbine} = \frac{g T_h}{A_p} \tag{5-10}$$

Where T_h is thrust from the turbine and A_p the Propeller area. $\Delta p_{turbine} = 100.8 Pa$.

The next step is to remove the losses through the wind tunnel from the pressure drop created by the turbine: $\Delta p_f = \Delta p_{turbine} - \Delta p = 50.2Pa$

Making the following assumptions: incompressible flow, steady state flow and uniform flow on streamline from the turbine inlet ($v = 0$) to the turbine outlet ($p = 0$), the Bernoulli's theorem allows the Author to get the airflow velocity right behind the turbine.

$$\frac{v^2}{2} + g * z + \frac{p}{\rho} = constant \tag{5-11}$$

$v = \sqrt{\frac{2\Delta p_f}{\rho}}$. The final step is to consider the Venturi's effect due to the diffuser.

$$v_{test\ chamber} = v \times \frac{S_2}{S_1} = 10.57m.s^{-1}$$

As expected the airflow velocity inside the test chamber is about $10m.s^{-1}$ at the design point. This result is checked thanks to a CFD analysis.

5.8 Wind Tunnel CFD Analysis

5.8.1 Geometry

A 2D CFD analysis using ANSYS Fluent® is led to confirm the analytic results. A geometry is created following the previous calculated dimensions.

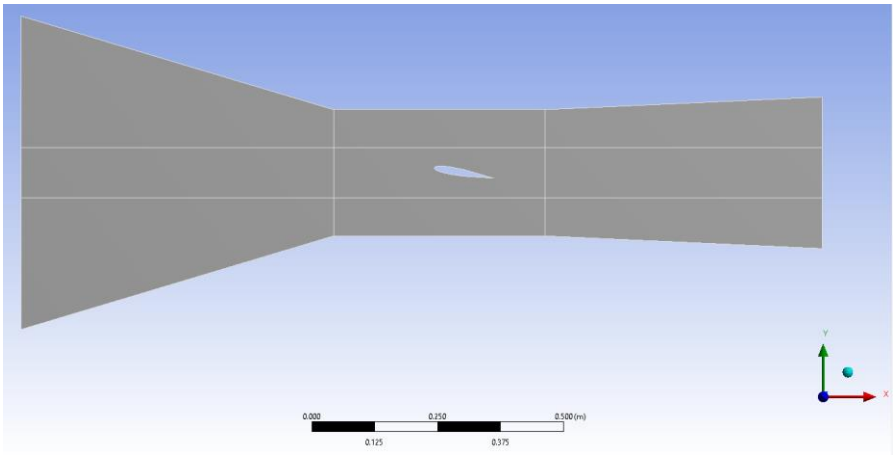


Figure 5-9 Geometry

The geometry is divided onto several parts to better control the mesh. The aerofoil is cut out from the geometry with an angle of attack of $10°$. This allows to check the model later by comparing the lift coefficient obtained with NASA's data.

5.8.2 Model $k-w$ Transition SST

The general settings of the model are:

- Pressure-based
- Steady analysis
- 2D analysis
- No gravity effect(useless)
- Energy equation turned on

The standard k-ω model in ANSYS Fluent is based on the Wilcox $k-\omega$ model, which incorporates modifications for Low-Reynolds number effects, compressibility, and shear flow spreading. The standard $k-\omega$ model is an empirical model based on model transport equations for the turbulence kinetic energy (k) and the specific dissipation rate (ω) [14].

The model used during the simulation is: “The transition SST model [, it] is based on the coupling of the SST $k-\omega$ transport equations with two other transport equations, one for the intermittency and one for the transition onset criteria, in terms of momentum-thickness Reynolds number” [14].

The model is suitable for this kind of analysis because it can be very accurate especially for the boundary-layer transition. This is a key point of the aerodynamic model developed later. The transition SST model is able to resolve the flow behaviour near a wall especially the viscous sub-layer. The model has a near-wall model approach and a very fine mesh is needed close to the wall [14].

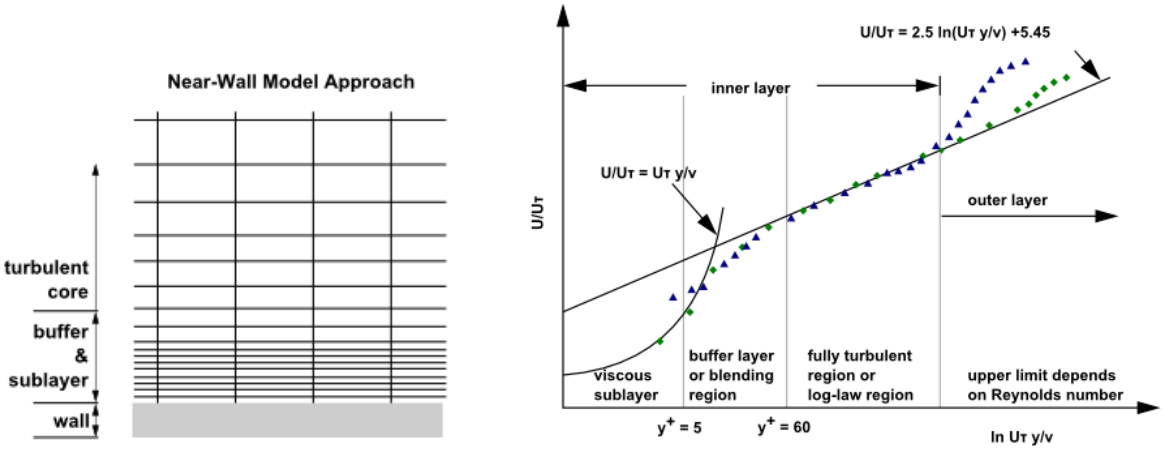


Figure 5-10 Near-wall model approach [14]

The y_+ criterion is expected to be inferior to 1 (up to 5) with the transition SST model [14].

The validation is performed in post-processing by plotting the values of y_+ along all wall surfaces of the model, checking whether it stays between critical value. If so, the size of mesh close to the wall is fine enough.

5.8.3 Mesh

Finding the right mesh is an iterative process. The mesh is structured with square cell and bias except around the aerofoil. The final mesh is shown below:

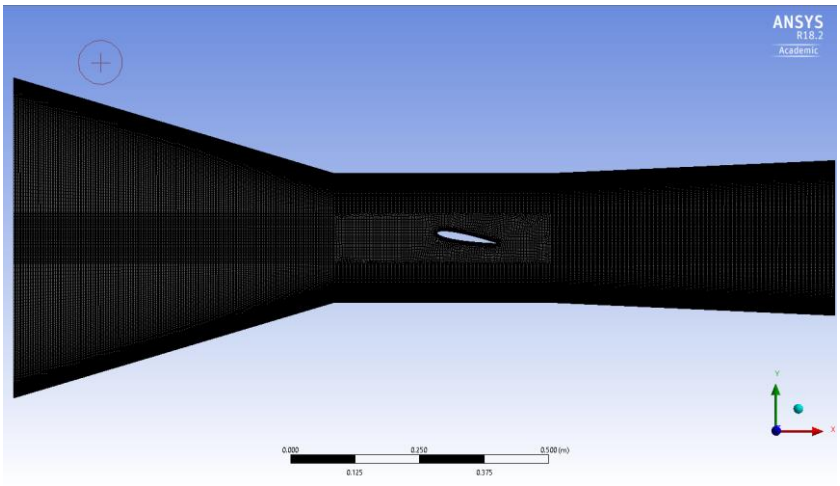


Figure 5-11 Overall mesh

With a very fine mesh near to the aerofoil to solve the viscous layer and have an accurate estimation of the boundary transition layer.

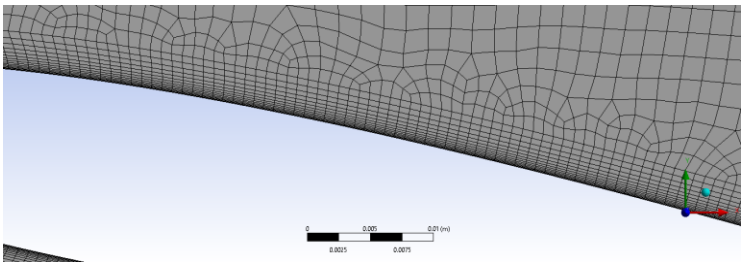


Figure 5-12 Mesh near to the aerofoil

An inflation is used around the aerofoil to control the near-wall size mesh. The mesh quality report given by Fluent® shows the results below:

- 276 255 elements and 277 727 nodes
- Minimum orthogonal quality 3.3×10^{-2}
- Maximum aspect ratio 1.6×10^{2}

For the sake of the project, these results are acceptable.

5.8.4 Results

The model conditions are:

- Pressure-inlet equals to pressure drop created by the turbine minus the pressure losses
- Pressure outlet equals to 0
- The initial values are computed from the outlet and the reference value is the aerofoil chord length for the dimensionless coefficient
- The analysis is stopped when all the parameters converge with a residual set at 10^{-4}

The CFD software allows to display the pressure distribution inside the wind tunnel and the streamline. The aim of this analysis is also to check if there are reversal flows inside the wind tunnel. The results are obtained after the calculation is fully converged.

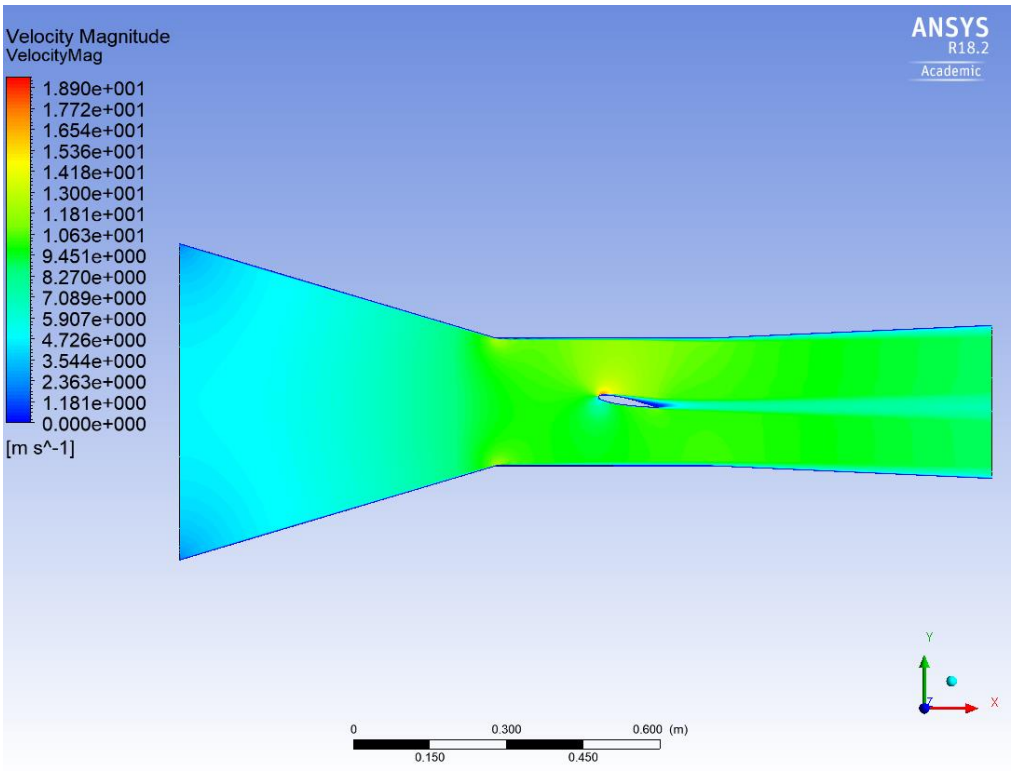


Figure 5-13 Velocity distribution inside the Wind tunnel

It is possible to see there is no reverse flow inside the wind tunnel and the velocity is uniform at the inlet of the test chamber. However, the low clearance between the aerofoil and the top creates an overpressure above the aerofoil.

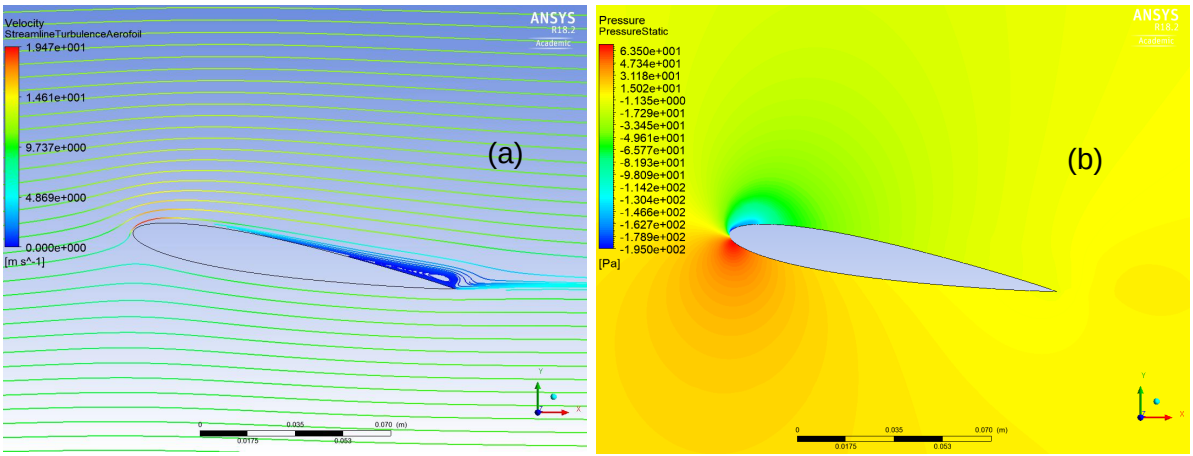


Figure 5-14 Streamlines (a) and pressure distribution around the aerofoil (b)

The airflow around the aerofoil is not perturbated and the boundary transition location is clearly visible a few millimetres after the front edge of the aerofoil: $X_t = 10\%$ of the chord. The lift coefficient is equals to $C_l = 1.11$ for $\alpha = 10°$. The airflow velocity before the aerofoil inside the test chamber is shown below:

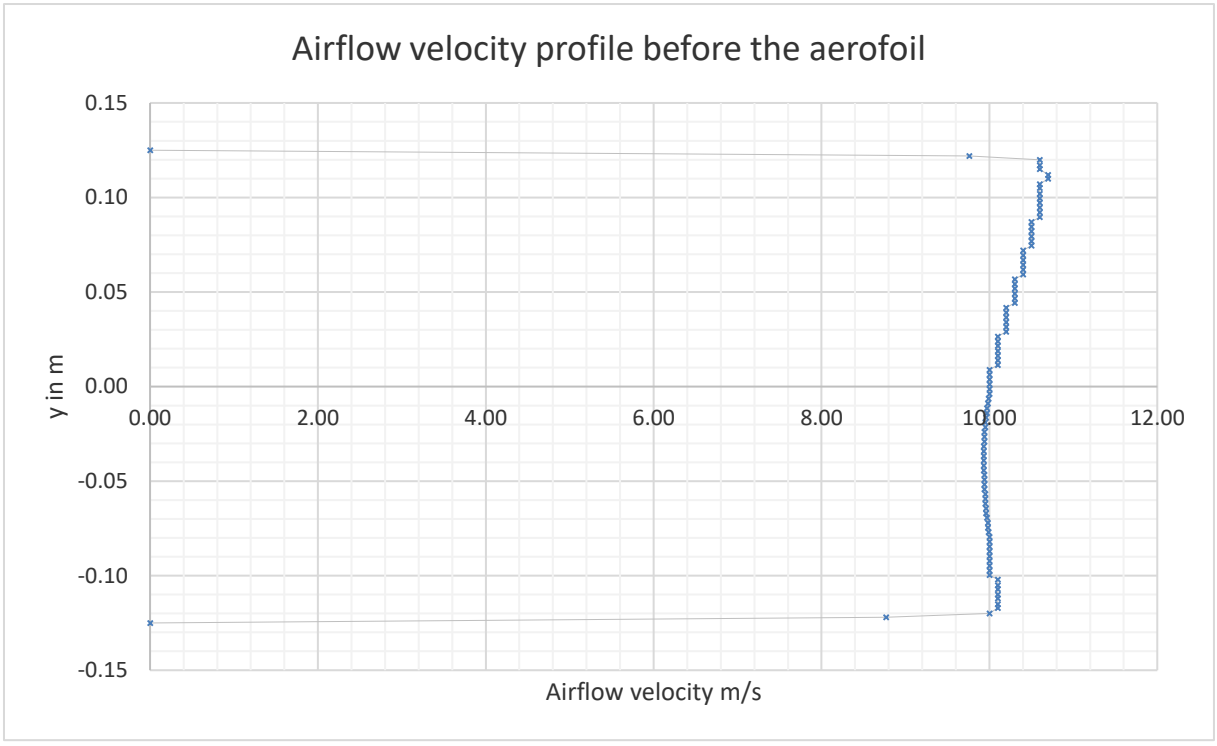


Figure 5-15 Airflow velocity profile before the aerofoil

The average airflow velocity is $v = 10.15 m.s^{-1}$.

5.8.5 Validation

The y_+ around the aerofoil and along the test chamber all are plotted below:

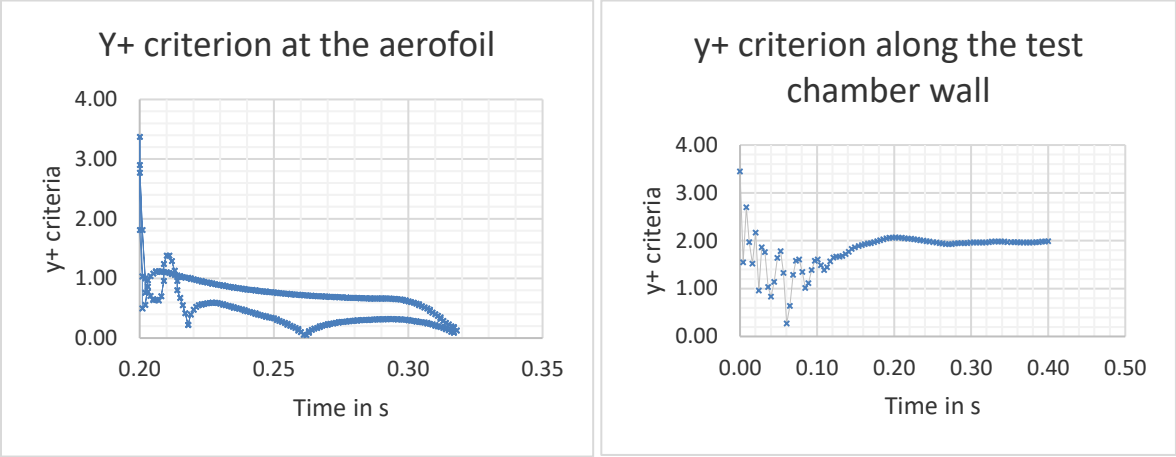


Figure 5-16 y_+ criterion at the aerofoil and along the test chamber wall

The validation criteria are summarized in the table below:

Table 5-5 Validation criteria

	Boundary layer transition location	y_+ criterion	Lift coefficient C_l $\alpha = 10°$	Test chamber velocity
Target Value	9%	< 5	1.00	$10.57m.^{-1}$
Obtained value	10%	$Max\ 2$	1.11	$10.15m.^{-1}$
Relative gap	1%	none	10%	4%

The CFD analysis and the wind tunnel geometry is validated despite the overpressure above the aerofoil. This will be threated later in the system model. The next step is to create the aerodynamic model of the aerofoil.

5.9 Wind Tunnel CAD model

A CAD model of the wind tunnel is created using CATIA® V5 following the dimensions calculated previously. A carriage is also drawn to transport the wind tunnel. The system is laid on a support.

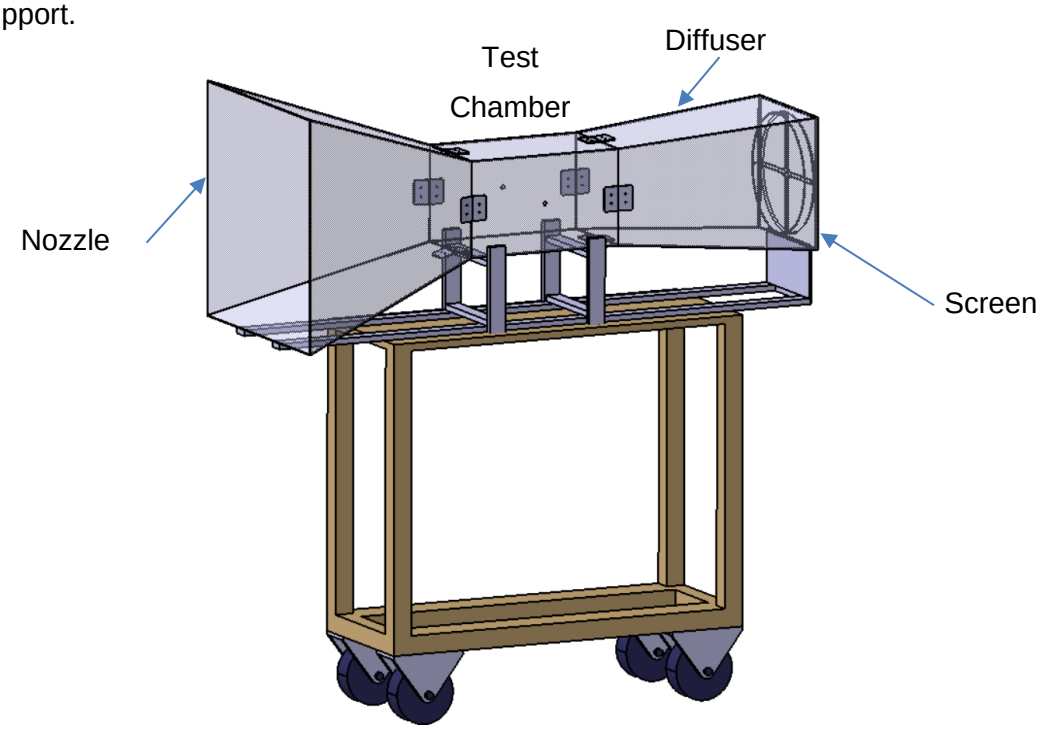


Figure 5-17 Wind tunnel CAD model

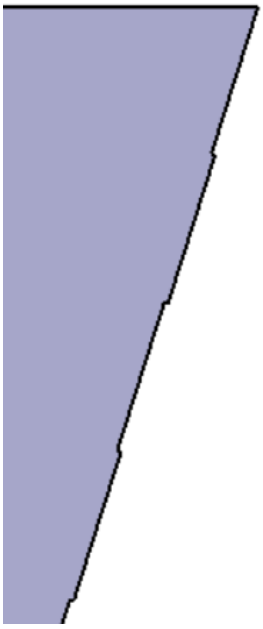


Figure 5-18 Panels nicks

The two holes on both sides of the test chamber fit bearings to allow the free rotation of the aerofoil. The overall architecture is straightforward to make the building process easier. The wind tunnel is divided into three parts: the nozzle, the test chamber and the diffuser. Each part is connected to the other by 4 joint parts. The Author is aware of the building process and the wind tunnel is designed with simple shapes to be obtained with a minimum of tools. The building material cost is also considered.

The Cranfield workshop have a laser-cut machine and a 3D printer. Those two machines allow to create a bit more complex shapes, especially for the joints parts. The panels constituting the nozzle, the test chamber and the diffuser must fit the laser-cut machine maximum dimensions. The nicks on the panels edges make the building process easier and the structure stiffer.

5.10 Wind tunnel Manufacturing

The panels from the nozzle, the test chamber and the diffuser are cut out from $3mm$ transparent polycarbonate sheets using the laser-cut machine. The panels of each section of the wind tunnel are then glued together. The motor support is cut out from doubled $3mm$ wood sheets.

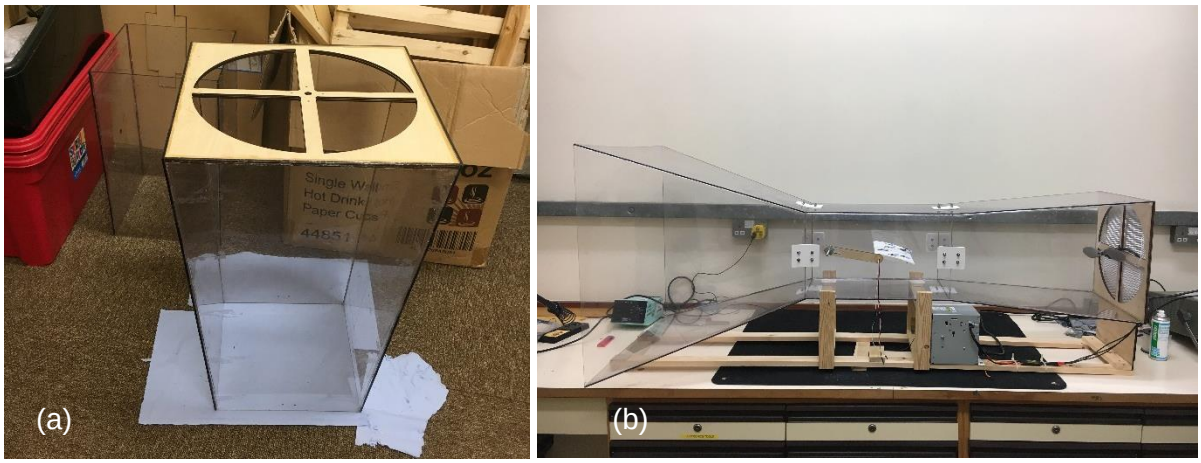


Figure 5-19 Diffuser gluing (a) and result (b)

The joint parts are 3D printed and make the gluing process easier. They are $5mm$ thick and screwed.

The support is made of wood plate and assemble with triangles and screws. Special locations are implemented to fit the power supply and the electronics. The wind tunnel leans on dumper to reducer the noise. The motor is screwed at the back of the wind tunnel and a grid (screen) protect the users.

The overall mass of the benchtop is low and can be handle by one person. The structure is quite stiff. Some other pieces of wood are used to complete the assembly to implement the measuring tools. The aerofoil inside the wind tunnel is fairly easy to access and it is possible to remove it for maintenance purpose.

Figure 5-20 Sections assembly and gluing

5.11 Conclusion

The geometry of the wind tunnel has been validated using CFD. Once the wind tunnel built, the next step of the project is to create an aerodynamic model of the aerofoil. The wind tunnel has a good looking and the transparent panels allow the user to see the aerofoil easily. The loud is acceptable. A good improvement should be an easier access to the aerofoil.

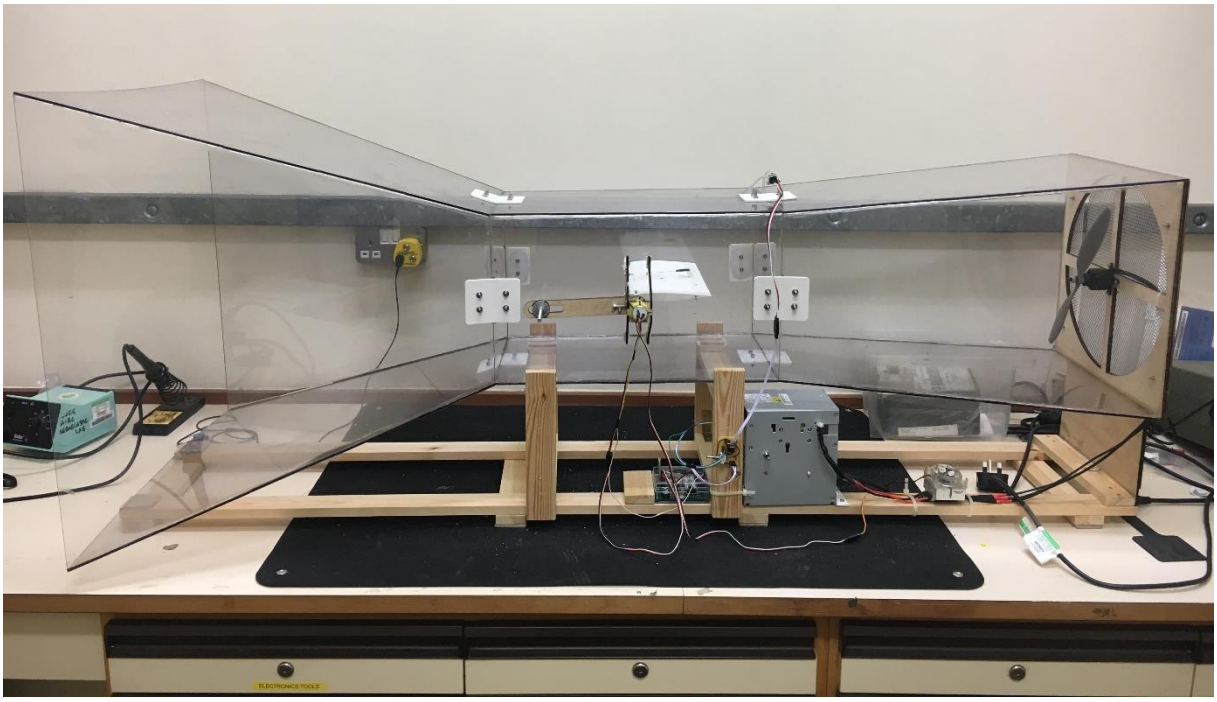


Figure 5-21 Flight Desk Control Demonstrator

6 Aerofoil analysis

6.1 General

The profile chosen for the aerofoil is a NACA 0012. It is a symmetric profile. The equation for 4 digits profile is:

$$y_t = \frac{t}{0.2}\left[0.2969\sqrt{\frac{x}{c}} - 0.1260\left(\frac{x}{c}\right) - 0.3516\left(\frac{x}{c}\right)^2 + 0.2843\left(\frac{x}{c}\right)^3 - 0.1015\left(\frac{x}{c}\right)^4\right] \quad (6\text{-}1)$$

Where c is the chord length, t is the thickness of the profile in percentage of the chord length and y_t.is the half thickness of the profile at x. $0 \leq x \leq c$.

Figure 6-1 NACA 0012 profile [15]

The chord length is $c = 120mm$ and the span wise is $s = 240mm$. There is then no side effects. The aerodynamic model is based on ESDU Datasheets. The trailing edge device is 20% of the chord length.

A key parameter for this analysis is the boundary layer transition location x_{tr}. It is approximated and extrapolated from a range of AoA using [15]. $x_{tr} = 34\%$ of the chord.

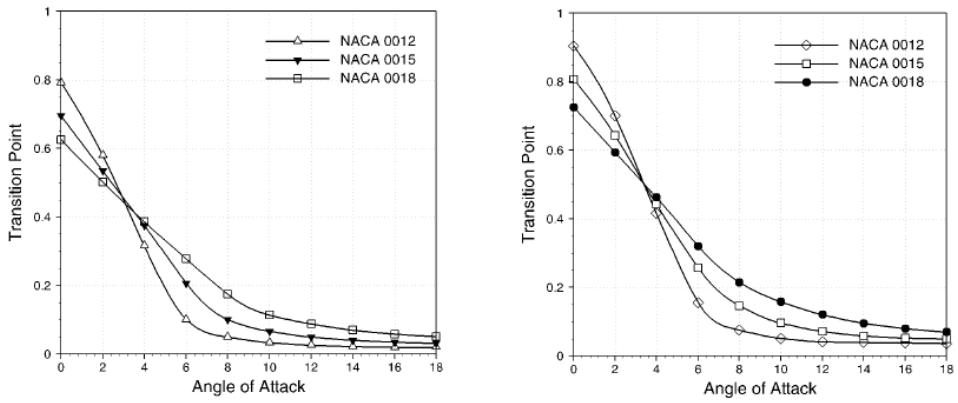


Figure 6-2 Variation of the transition point with the angle of attack over the NACA 0012, NACA 00115, and NACA 0018 airfoil profiles for the Reynolds numbers of (a) 5×10^5 and (b) 2×10^5 [15]

6.2 Aerodynamic characteristics of the aerofoil

6.2.1 Domain of validity and accuracy

The results in this section are validated for 2D compressible or incompressible inviscid (except if mentioned otherwise) airflow at subcritical M_aMach numbers.

$$M_a = \frac{V}{a} \tag{6-2}$$

Where V is the airflow velocity and a is the speed of sound in air. Here $M_a = 0.031$. The domain of validity and the accuracy of the model are given below. The model can be applied for this project. The Reynold's number with the chord length as reference is $R_e = 83\,300$.

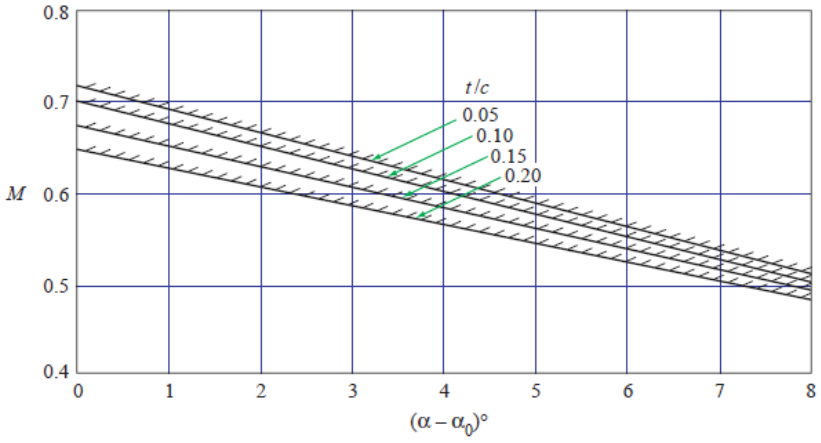


Figure 6-3 Domain of validity

-

Figure No.	*Parameter*	*Maximum Error* (±)
1	a_{1i}	0.0005
2	α_{0i}	3 per cent $(z_{c6}/z_{c4} \leq 0.5)$ 10 per cent $(z_{c6}/z_{c4} > 0.5)$
3	C_{m0i}	5 per cent
4	x_{ai}/c	0.001

Figure 6-4 Accuracy in incompressible flow [16]

6.2.2 Geometry useful parameters

Some geometric parameters are needed to compute the characteristics of the aerofoil.

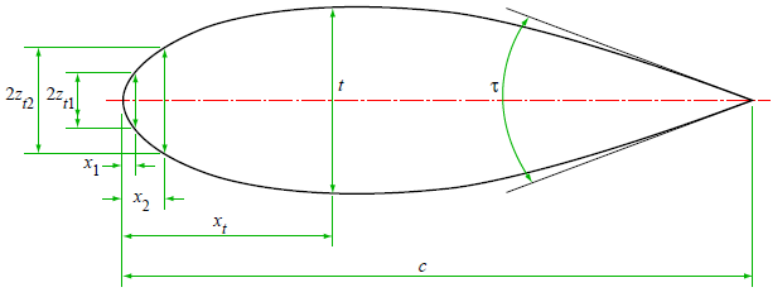


Figure 6-5 Thickness distribution [16]

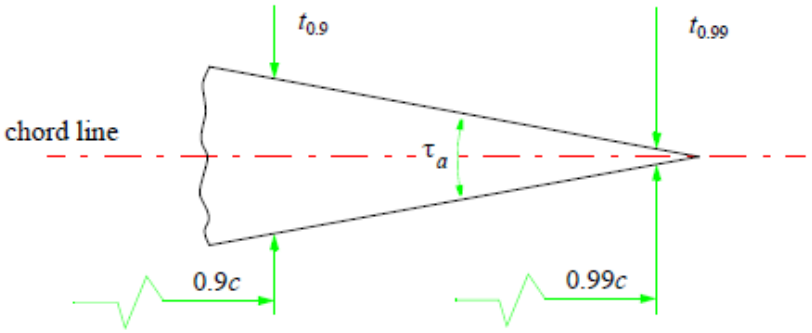


Figure 6-6 Trailing edge angle [17]

Table 6-1 Useful parameters value

Thickness equation					
Dimension	Value	unity	Dimension	Value	unity
x/c	0.90	mm	t0.9	0.00	mm
x/c	0.99	mm	t0.99	0.00	mm
x1/c	0.005	mm	zc1	0.00	mm
x2/c	0.05	mm	zc2	0.00	mm
x3/c	0.20	mm	zc3	0.00	mm
x4/c	0.50	mm	zc4	0.00	mm
x5/c	0.90	mm	zc5	0.00	mm
x6/c	0.92	mm	zc6	0.00	mm

The trailing edge angle τ is given by [17]:

$$\tan\left(\frac{\tau}{2}\right) = \frac{t_{0.9} - t_{0.99}}{0.18c} \tag{6-3}$$

$$\tau = 7.52°$$

The two coefficients $F_1 = 0.80$ and $F_3 = 0.26$ are obtained from [16] figure 1 and figure 4 respectively.

6.2.3 Aerodynamic characteristics

It is known that the moment at the aerodynamic centre for a symmetric profile is equals to zero no matter the AoA. This special point is very useful to lead an aerodynamic analysis of an aerofoil. The aerodynamic centre is given by [16]:

$$\frac{\mathrm{x_{ai}}}{c} = \frac{1}{4}\left(1 + F_3\frac{t}{c}\right) \tag{6-4}$$

$$\mathrm{x_{ai}} = 0.26$$

This is a common value, close to $\frac{1}{4}$, which is commonly used for a symmetric aerofoil analysis. [18]. The location of the aerodynamic centre is fundamental for an aerofoil analysis.
The inviscid lift-curve slope $(a_1)_T$ is given by [16]:

$$(a_1)_T = 0.10967\left(1 + F_1\frac{t}{c}\right) \tag{6-5}$$

$$(a_1)_T = 0.12\ deg^{-1}$$

The viscous lift-curve slope $(a_1)_v$ is given by [17]:

$$(a_1)_v = \left[\frac{(a_1)_v}{(a_1)_T}\right]_{sym} \times (a_1)_T \tag{6-6}$$

Where $\left[\frac{(a_1)_v}{(a_1)_T}\right]_{sym} = 0.95$ by extrapolating the data from [17] figure 4 and 5.

$$(a_1)_v = 0.11\ deg^{-1}$$

According to [19], “Within the linear range of the lift-incidence curve and over the range of control deflection for which the increment of lift is linear with control deflection, the lift coefficient C_l of a two-dimensional aerofoil at an angle of incidence α and a control surface deflection δ is given by” at the aerodynamic centre:

$$C_l = (a_1)_v \alpha + (a_2)_{0v} \delta \tag{6-7}$$

This equation is applicable to a range of δ between $\pm 15°$ with an accuracy of $\pm 15\%$ where the gap between the aerofoil and the flap is sealed.

The rate of change of lift coefficient with control deflection in compressible flow $(a_2)_v$ is given by [17]:

$$(a_2)_v = \left[\frac{(a_2)_{0v}}{(a_2)_{0T}}\right] \times (a_2)_{0T} \tag{6-8}$$

Where $\left[\frac{(a_2)_{0v}}{(a_2)_{0T}}\right] = 0.90$ from [19] figure 2 and $(a_2)_{0T} = 3.65\ rad^{-1}$ from [19] figure 1.

$$(a_2)_{0v} = 3.44\ rad^{-1}$$

In a similar way, the rate of change of pitching moment coefficient with a plain control deflection in compressible flow m_0 is given by [20] at the aerodynamic centre:

$$m_0 = \mathrm{m_{0T}} \left(\frac{m_0}{m_{0T}}\right) \tag{6-9}$$

Where $\left(\frac{m_0}{m_{0T}}\right) = 0.90$ from [20] figure 2 and $m_{0T} = 0.64\ rad^{-1}$ from [20] figure 1.

$$m_0 = 0.53\ rad^{-1}$$

6.3 Static aerodynamic equilibrium

Once the aerodynamic modification due to the flap deflection known, it is possible to compute the static aerodynamic equilibrium between the AoA of the aerofoil and the flap deflection. It is a one DOF system and the calculation is led at the aerofoil pivot liaison between the wing and the wind tunnel.

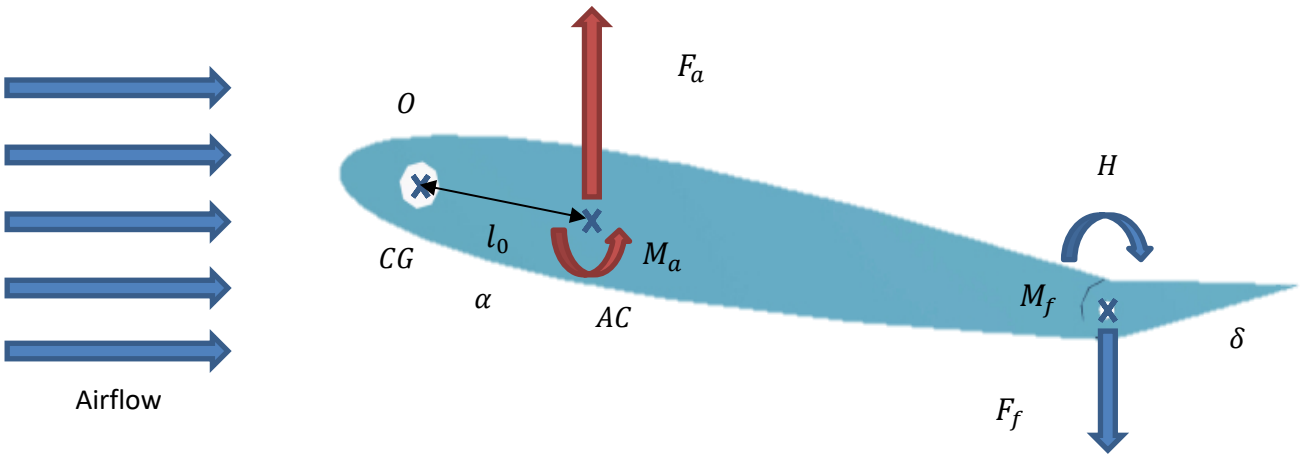


Figure 6-7 Aerofoil static aerodynamic equilibrium

The centre of gravity of the aerofoil is located at the pivot location thanks to a crank fixed to the aerofoil axis. This crank is here to balance the aerofoil to get a CG at the right location at any time. The system is then independent from the gravitational loads.

A lot of dimensionless data are used in this analysis, the variable $q = 68.4\ Kg.m^{-1}.s^{-2}$ is useful to get the dimensionless moment of inertia from the inertia $I = 1544\ Kg.mm^2$ of the aerofoil computed with CATIA® V5:

$$q = \frac{1}{2}\rho V^2 \qquad (6\text{-}10)$$

By applying the fundamental principle of dynamics in the plan at the pivot location O:

$$\frac{I}{qS}\ddot{\alpha} = -\alpha l_0 (a_1)_v + \delta l_0 (a_2)_0 + \delta c m_0 - \frac{v}{qS}\dot{\alpha} \qquad (6\text{-}11)$$

Where v is the viscous friction inside the bearings of the pivot and due to the potentiometer internal friction. $S = s \times c = 28\,571\ mm^2$ is the reference area of the aerofoil. $l_0 = 20.68mm$ is the distance between AC and O.

In static $\ddot{\alpha} = 0$ and $\dot{\alpha} = 0$, the equation (6-11) becomes:

$$\frac{\delta}{\alpha} = \frac{l_0(a_1)_v}{l_0(a_2)_0 + cm_0} = \frac{\partial \delta}{\partial \alpha} \tag{6-12}$$

$$\frac{\partial \delta}{\partial \alpha} = 1.01 \frac{deg}{deg}$$

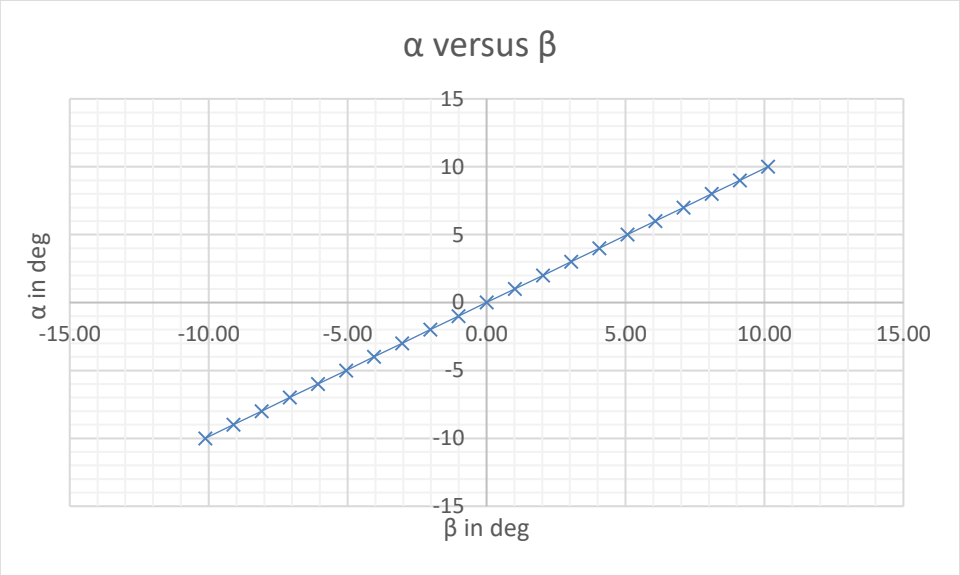


Figure 6-8 α versus β in static aerodynamic equilibrium

This is this value of $\frac{\partial \delta}{\partial \alpha}$ that is going be implemented in the microcontroller to control the flap by the servomotor according to the angle-command in open-loop.

$$\delta \cong \alpha \tag{6-13}$$

This value of $\frac{\partial \delta}{\partial \alpha}$ is also very convenient because it means an AoA of α is reached when $\delta \cong \alpha$.

6.4 Model validation

The aim of this section is to validate the static equilibrium described above using JAVAFOIL®. The software is set up with the parameters of the analysis. The input of the analysis is the AoA α and the flap defection δ where $\delta \cong \alpha$ following Figure 6-8. A range of α between $-10°$ and $10°$ is used. Beyond these limits the stall occurs, and the model is not validated anymore. A script is written to make the process automatic 11Appendix E.

The comparison between the analytic model and the results from JAVAFOIL® are shown below:

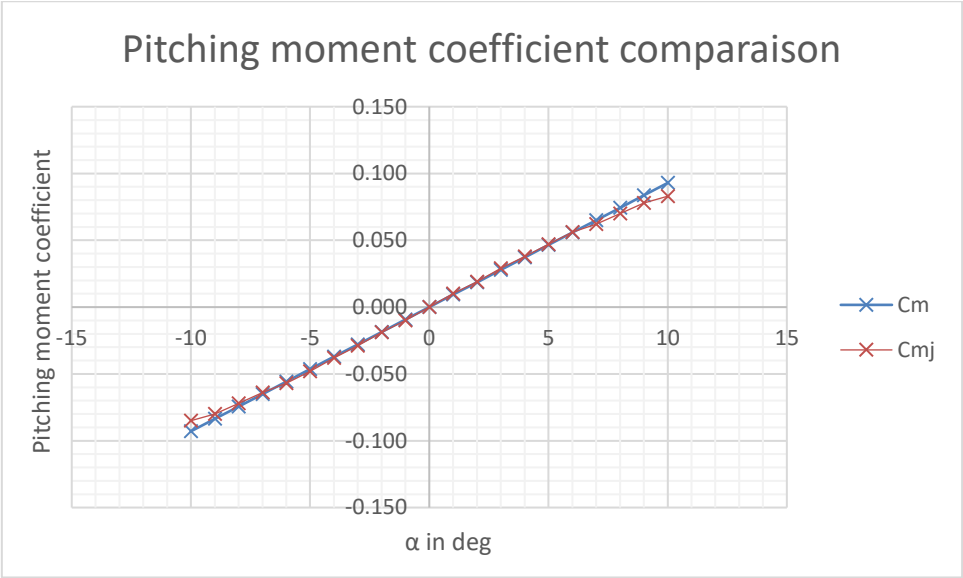


Figure 6-9 Pitching moment coefficient comparison

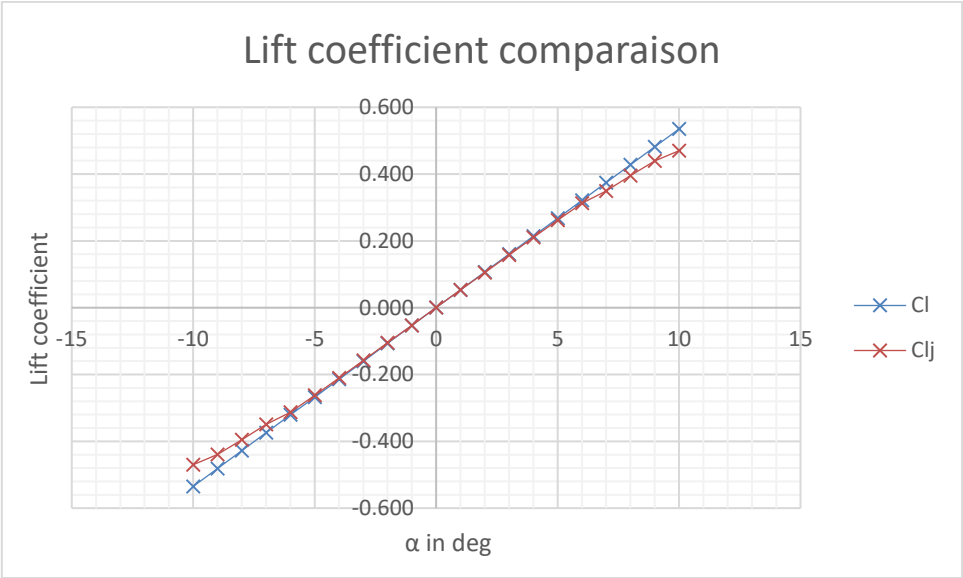


Figure 6-10 Lift moment coefficient comparison

Finally, the theoretical model is close to the results from JAVAFOIL®, the relative gap is below 10% for both curves. The model is so validated and can be used for the theoretical model of the demonstrator later in MATLAB®.

6.5 Aerofoil CAD Model

A CAD model of the aerofoil is created using CATIA® V5 following the dimensions calculated previously. The crank is designed to locate the aerofoil assembly CG at the pivot location. The aerofoil is divided into three parts:

- The aerofoil itself is divided into two parts to limit the height of the part during the printing process
- The flap is also divided into two parts to limit the height of the part during the printing process. The flap features an emplacement for the horn.
- A shield to close the servomotor emplacement. The servomotor is indeed directly integrated inside the aerofoil design and accessible by removing the shield. The shield is mounted to the aerofoil thanks to a screw. An oblong allows the servomotor arm to move.

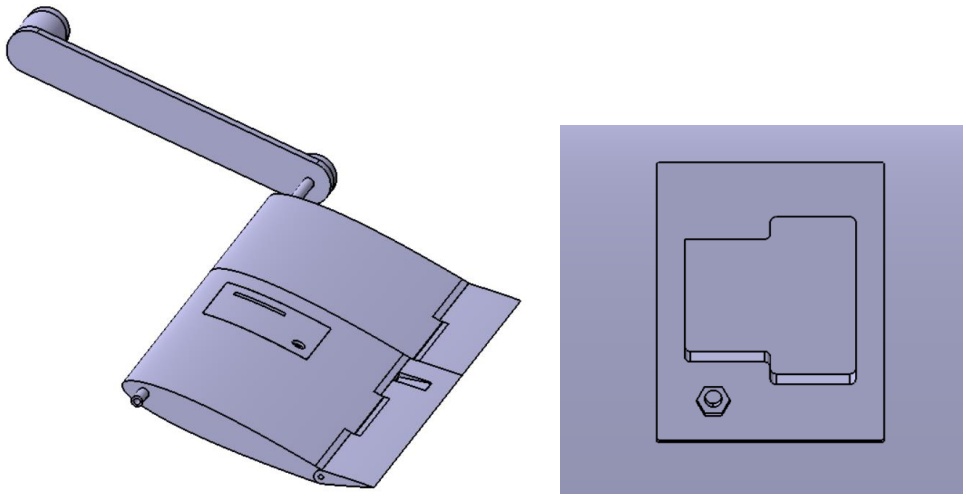


Figure 6-11 Aerofoil CAD Model

The joint between the aerofoil and the flap must be as small as possible but still must allow the rotation. The flap deflection is superior to $\pm 15°$ and mechanically limited by the flap geometry.

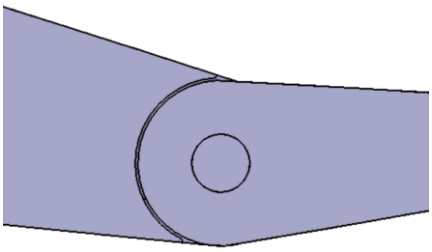


Figure 6-12 Joint between the aerofoil and the flap

6.6 Aerofoil Manufacturing

The aerofoil's parts are 3D printed and glued together. The crank is cut out from $3mm$ wood sheets. The axis between the flap and the aerofoil is a tube made of steel with a diameter of $1.5mm$. The axis for the pivot liaison is an aluminium tube with an external diameter of $6mm$ and an internal diameter of $4mm$. The wires from the servomotor go inside this tube. The aerofoil is very light because it is totally empty inside.

The servomotor is glued inside the aerofoil. This is obviously not the best solution but a test with double-sided tape was not conclusive.

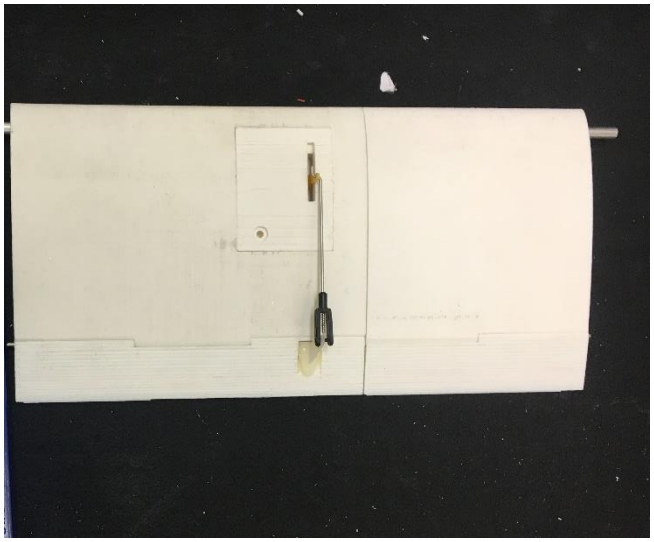


Figure 6-13 Aerofoil assembly

The crank is simply fixed by friction at the end of the aluminium tube and the rotational motion is transmitted by a flat part. A heavy screw at the end of the crank can be adjusted to balance the aerofoil.

The link between the servomotor and the horn is adjusted and the assembly is implemented inside the wind tunnel. The assembly and the 3D parts are quite fragile and must be manipulated carefully.

6.7 Conclusion

The analysis of the aerofoil gives a reliable aerodynamic model. This model is used for the theoretical approach of the demonstrator and its assessment. The aerofoil is implemented inside the wind tunnel and balanced. The pivot liaison has low friction thanks to the bearings. The next step is to program the GUI and the microcontroller.

Table 6-2 Aerofoil geometry characteristics

q	S Wing area	I Wing assembly inertia	Length AC-O l_0	c chord
$68.4\ Kg.m^{-1}.s^{-2}$	$28\ 571\ mm^2$	$1544\ Kg.mm^2$	$20.68\ mm$	$119.05\ mm$

Table 6-3 Aerofoil aerodynamic characteristics

$(a_1)_v$ lift coefficient slope	rate of change of pitching moment m_0	rate of change of lift coefficient $(a_2)_0$	$\frac{\partial \delta}{\partial \alpha}$
$0.11\ deg^{-1}$	$0.53\ rad^{-1}$	$3.44\ rad^{-1}$	$1.01\frac{deg}{deg}$

The inertia I is obtained from CATIA® V5 where:

$$I = I_{yy} + I_{xy} + I_{zy} \tag{6-14}$$

7 Electronics and PDB

7.1 General

The Author have made the decision to design a PDB to fit all the electronics on. This can be achieved for cheap and offers many advantages including a reliable circuit, a professional project looking, a size optimisation and a lot of time saved. The electronics were first tested onto a breadboard according to the following schematic:

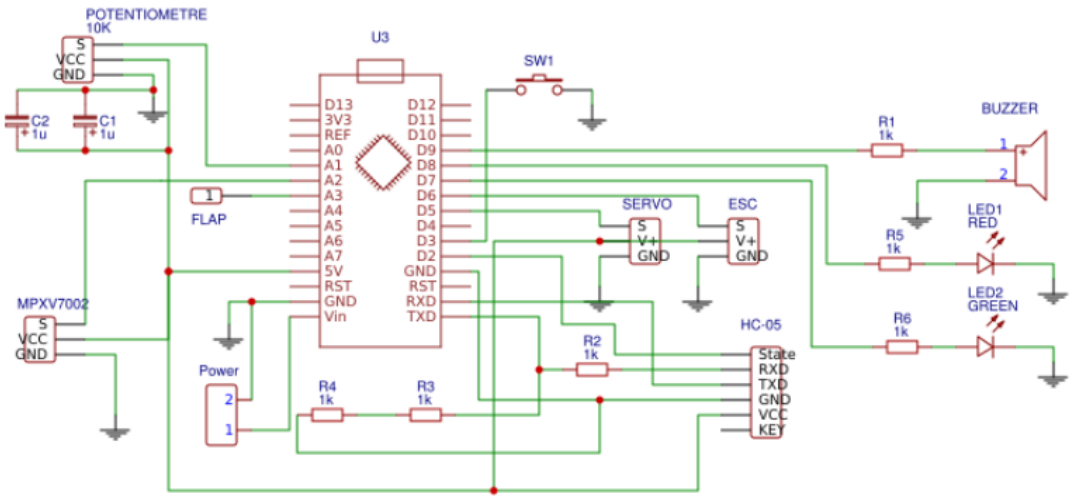


Figure 7-1 Electronic circuit

The circuit has emplacements for:

- Female headers for an Arduino® Nano with the microcontroller ATMEL® 328p $5.0V$ and $16MH_z$ and for the Bluetooth module HC-05
- Male headers to power the board from an external power supply, to read the flap actuator position (if needed), to read the analogic data input from the potentiometer and the pitot tube (MPXV7002)
- 2 capacitors $1\mu F$ in parallel to limit the noise from the ESC BEC.
- A voltage divider with three $1k\Omega$ resistance for the RX pin of the Bluetooth because it only accepts $3.3V$ level
- A buzzer HMB1275-12B with its $1k\Omega$ resistance to limit the current
- A green LED with its $1k\Omega$ resistance to limit the current
- A red LED with its $1k\Omega$ resistance to limit the current
- A push button on the pin interrupt D3 of the Arduino® to set the angle zero of the demonstrator

7.2 The Pitot tube (MPXV7002)

According to the MPXV7002 datasheet [21] (pitot tube) the output voltage is within the range between $0.5V$ and $4.5V$ where the middle of this range equals to the sensor output at rest. However, the internal reference of $1.1V$ of the Arduino® is used and the voltage output of the sensor needs to be adapted. A circuit is created including a dual operational amplifiers LM328p [22].

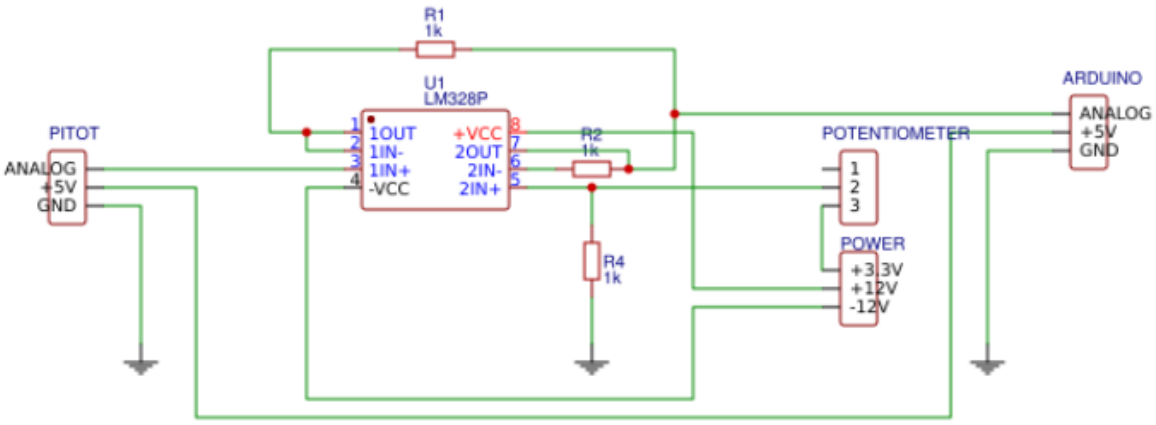


Figure 7-2 Pitot tube output adaptor circuit

The first OpAmp is used as a voltage follower because of the high impedance of the sensor output. Indeed, any current drawn from the sensor will cause a voltage drop and so a basic voltage divider cannot be used to reduce the voltage. Moreover, a voltage divider will reduce the data reading accuracy. The solution to offset the tension is to use another OpAmp as a differential amplifier. A potential of $3.3V$ from the demonstrator power supply is used. The value of the sensor output will be subtracted to $3.3V$ and then read by the microcontroller.

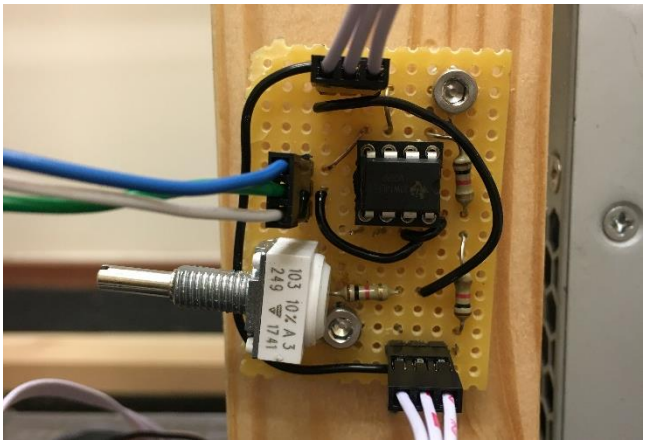


Figure 7-3 Pitot tube voltage adaptor circuit

The dual operational amplifiers LM328p is powered by the demonstrator power supply with $\pm 12V$ while all the other components are powered by the $5V$ BEC from the ESC.

A potentiometer of precision allows the user to trim the differential output to fit with the allowable range of readable tension by the microcontroller.

The calculation of the pressure induced by the airflow motion is given in [21]:

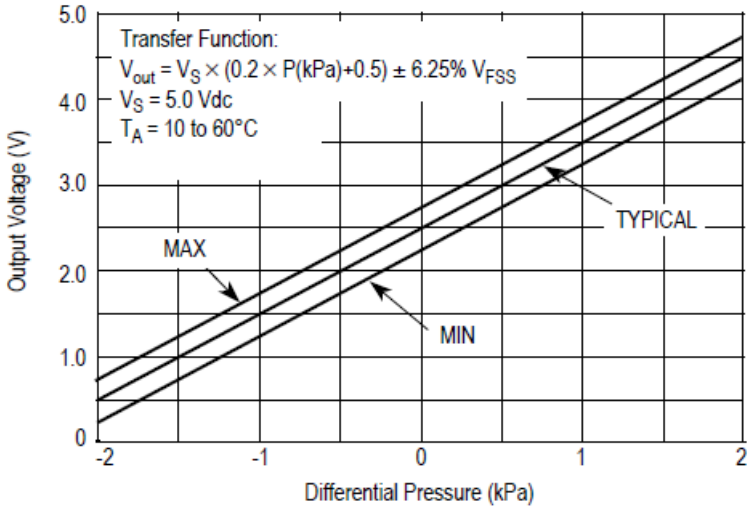


Figure 7-4 Analogic output voltage from the pitot tube [21]

The airflow velocity V can be calculated from the difference of pressure Δp using the Bernoulli's equation (5-11):

$$V = \sqrt{2\frac{\Delta p}{\rho}} \qquad (7\text{-}1)$$

The pitot is strategically placed into the wind tunnel to not disturb the airflow according to the CFD analysis. The pitot tube is located behind the aerofoil and on a side to not disturb the system. The sensor is located as far as possible from the panels to avoid the boundary layer.

After adjustment the airflow velocity measured by the Pitot tube is close to the theoretical value. The value from the sensor increases up to $9.5\, m.s^{-1}$ at 55% of the turbine power and then decreases to $8.5 m.s^{-1}$ at 65% of the turbine power. This can be explained by the thickening of the boundary layer or local reverse flows when the airflow velocity increases. However, the Pitot tube cannot be located closer to the middle of the wind tunnel because of the perturbations induced by the aerofoil. A value of the airflow velocity above $10\, m.s^{-1}$ at 65% is expected.

7.3 PDB Design

The PDB is then designed according to Figure 7-1. The pins' name and the connection as well as the Cranfield University's logo are directly printed onto the PDB.

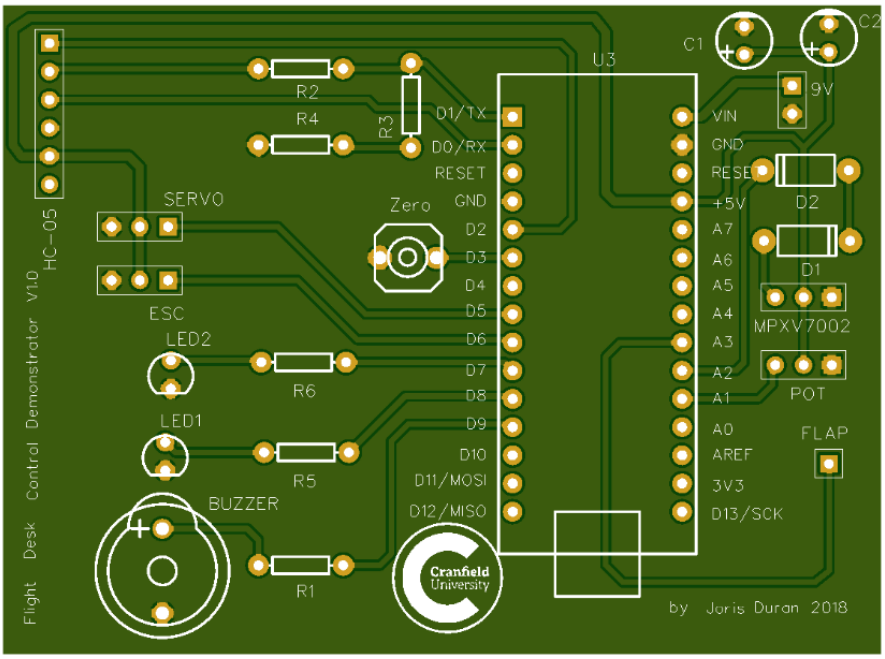


Figure 7-5 Demonstrator's PDB

The final step is to solder the components onto the PDB.

Table 7-1 PDB characteristics

Dimension	Thickness	Copper Weight	Material
$79\ mm \times 59\ mm$	$1.6\ mm$	$1.0\ oz.$	FR4-Standard Tg 140C

7.4 Conclusion

The printed PDB brings the project to a next level. The electronics is more reliable, and the look is more professional. The dimensions of the PDB are optimal. Unfortunately, a last-minute issue required the creation of a new circuit for the Pitot tube and this one is not integrated into the main PDB.

8 Microcontroller programming

8.1 General

This section aims to describe the code within the microcontroller and how it interacts with the GUI. The totality of the code is not explained but only the key functions:

- How the PID control is implemented
- How the communication with the GUI works

The code is programmed in C++ using the Arduino® IDE. It is divided into several files to better organise the program. The main loop and the initial settings are written in a *.ino* file which is the main Arduino file written in Arduino language (derivated from C++). The related functions and functionality are written in separated files in C++: the header *.h* files with the prototype functions and the main variables, and the *.cpp* files including the definition of the functions and needed variables.

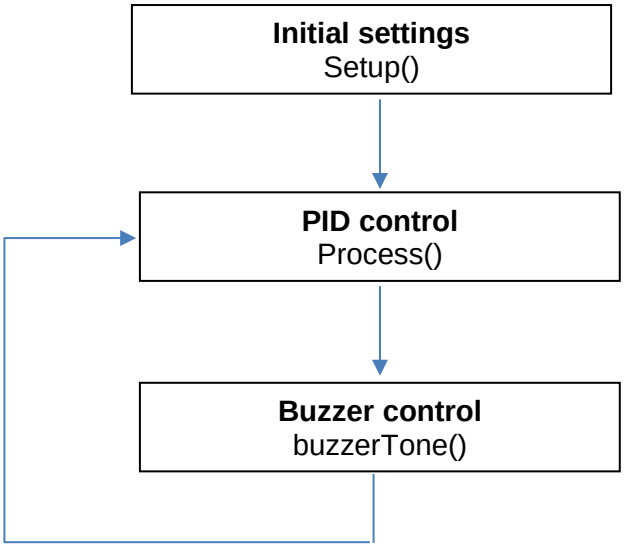


Figure 8-1 Microcontroller code architecture

Two important features of the code are the control of the buzzer/LEDs and the control of the turbine. Both are non-blocking function. The buzzer is enable with a Boolean variable during a certain amount of time. The board sends a ramp to the turbine to avoid pic of current and putting the power supply in security mode.

Another key point is the fact that the microcontroller only deals with integers from the ADC and the servomotor is directly controlled using the duty cycle of the PWM signal (neutral at 1500). However, the integers used by the microcontroller are converted into the real values with two decimals just before being sent to the GUI.

8.2 PID initialisation

The PID calculation is performed within the microcontroller main loop. The PID control is the Author version of the free to use and modify PID Arduino® library from Brett BEAUREGARD. The PID control are first initialised:

```
PID PIDsettings(&Input, &OutputServo, &Setpoint, 0, 0, 0, DIRECT);
```

Hard links are created between the variable *Input* from the potentiometer analogic signal, the command to servomotor *OutputServo* and the command *Setpoint* thanks to pointers The PID terms are first initialised at 0, the system works in open-loop. The PID output is not reversed compare to the input.

```
PIDsettings.SetWeightFilter(0.1);
PIDsettings.SetSampleTime(SAMPLE_TIME);
PIDsettings.SetOutputLimits(flapMillM15, flapMillP15);
PIDsettings.SetInputLimits(angleMillM15, angleMillP15);
PIDsettings.SetSetpointLimits(flapMillM15, flapMillP15);
```

The different limits are defined. The PID output limits are adjusted to be added to the command angle according to the aerodynamic model. The filter on the derivative term is set to 10% wich represents $f_c = 8.38H_z$ with a sampling time of $T = 2ms$. The filter percentage is obtained following this formula:

$$filter\% = \left(1 - e^{-2\pi T f_c}\right) \times 100 \tag{8-1}$$

The reference angles are calculated as follow:

```
flapMillP15 = 1000 + 1000*(90+3*LIMITE_ANGLE)/180;
flapMillM15 = 1000 + 1000*(90-3*LIMITE_ANGLE)/180;
angleMillP15 = zeroAngle + 1024*LIMITE_ANGLE/74.8;
angleMillM15 = zeroAngle - 1024*LIMITE_ANGLE/74.8;
```

The angle sent to the servomotor is three times the command angle to get the flap at the right angle. This is a real advantage because the servomotor is more accurate with this bigger range of angles. The *zeroAngle* variable can be set manually using the push button on the board. $74.8°$ is the maximum angle from the potentiometer to get an analogic output signal between 0 and $1.1V$.

8.3 PID Loop

Once the data input from the potentiometer are read, the PID terms are computed using the line:

```
Input = 0.988*Input + 0.012*analogRead(POTENTIOMETRE);
PIDsettings.Compute();
```

The analogic data from the potentiometer are also digitally filtered to smooth the curve. The values presented above show a smooth curve without creating a delay compare to the non-filtered data input. Indeed, the value read by the microcontroller are not continuous because of the ADC (digitalisation) and this phenomenon can perturbate the PID control especially the derivative term. The explanation below describes the *compute()* function.

```
if(!inAuto) return false;
   unsigned long now = millis();
   unsigned long timeChange = now - lastTime;
   if(timeChange >= SampleTime)
   {
```

The working variables are computed:

```
double input = PID::mapf(*myInput, inMin, inMax, setMin, setMax);
double error = *mySetpoint - input;
double dInput = input - lastInput;
```

And the PID term are calculated. The proportional term is calculated on the error. kp is equal to the Proportional coefficient, ki is equal to the Integral coefficient time the sampling time and kd is equal to the Derivative coefficient divided by the sampling time:

```
outputSum += ki * error;
if (ki == 0 && pOnE) outputSum= 0;
     else if(outputSum + *mySetpoint > outMax) outputSum= outMax - *mySetpoint;
     else if(outputSum + *mySetpoint < outMin) outputSum= outMin - *mySetpoint;
```

The *outputSum* is the total Integral term. It can be removed by setting the Integral coefficient to zero. The windup phenomenon is removed here by clamping. The other terms are added:

```
double output;
if(pOnE) output = kp * error;
derivative = (1-weightFilter)*derivative + weightFilter*kd*dInput;
output += outputSum - derivative;
```

The PID output is finally limited by the user-defined limits and the variables for the next loop are stored:

```
    if(output + *mySetpoint > outMax) output = outMax;
    else if(output + *mySetpoint < outMin) output = outMin;
    else *myOutput = output + *mySetpoint;
    lastInput = input;
    lastTime = now;
    return true;
  }
  else return false;
}
```

The function *compute()* is now over and the command to the flap is sent by the following lines:

```
flapActuator.writeMicroseconds(OutputServo);
```

The command to the flap is already constrained within the PID output calculation.

8.4 Data sending loop

The data sending loop is included within the main loop. The BAUD rate for the serial communication is set at 115 200.

The communication between the microcontroller and the GUI is possible via two buffers:

```
byte dataSent[BUFFER_LENGTH_SENT];
byte dataReceived[BUFFER_LENGTH_RECEIVED];
```

The two buffers are basically two lists of bytes. They are filled with a known order with the right data. When a float value is sent, it is first multiplied by 100 by *decimalDivider()* to only keep two decimals and then it is divided into to two bytes by *byteDivider()*. The least significant byte is still signed. A basic operation transforms it into an unsigned byte when a buffer is received. The float multiplied by 100 is reconstituted as follow:

```
int serialReadData(int Pos) {
   return ( dataReceived[Pos] << 8) | ( dataReceived[Pos + 1] & 0xFF );
 }
```

The value threated by the microcontroller are converted into the real value with two decimals just before being sent:

```
void serialSendData() {
  if (Serial || digitalRead(BT_STATE) == HIGH) {
    int inputSent = decimalDivider( mapf(Input, angleMillM15, angleMillP15, -LIMITE_ANGLE, LIMITE_ANGLE) );
    int outputSent = decimalDivider( mapf(OutputServo, flapMillM15, flapMillP15, -LIMITE_ANGLE, LIMITE_ANGLE) );
    int airVelocitySent = decimalDivider( sqrt(mapf(airVelocity, 896, 0, 0, 1796)) );
    int setpointSent = decimalDivider( mapf(Setpoint, flapMillM15, flapMillP15, -LIMITE_ANGLE, LIMITE_ANGLE) );
```

Then, the buffer is filled and sent via the serial port:

```
    dataSent[INPUT_READ] = byte( byteDivider(inputSent) );
    dataSent[INPUT_READ + 1] = byte(inputSent);
    dataSent[OUTPUT_PID] = byte( byteDivider(outputSent) );
    dataSent[OUTPUT_PID + 1] = byte(outputSent);

    dataSent[AIR_VELOCITY] = byte( byteDivider(airVelocitySent) );
    dataSent[AIR_VELOCITY + 1] = byte(airVelocitySent);
    dataSent[PERIOD] = byte( byteDivider(PERIOD_UART) );
    dataSent[PERIOD + 1] = byte(PERIOD_UART);

    dataSent[ECHO_SETPOINT] = byte( byteDivider(setpointSent) );
    dataSent[ECHO_SETPOINT + 1] = byte(setpointSent);

    Serial.write( dataSent, BUFFER_LENGTH_SENT );
  } else {
    pflagTone = true;
  }
}
```

The communication loop sends data every $25ms$.

8.5 Turbine Control

The turbine's motor is controlled via the ESC within a range of value between 0% and 100%. It is not possible to send directly a high value to the ESC because this could create a pic of current and put the power supply into safety mode. The solution is to send a ramp to the ESC.

```
void setMotorFanSpeed() {
 currentMillisMotor = millis();
 if (motorFanCommand == 0) {
    motorFanSpeed = motorFanCommand;
    motorFan.writeMicroseconds(1000 + 10*motorFanSpeed);
 } else if (motorFanCommand > motorFanSpeed) {
    motorFanSpeed++;
    motorFan.writeMicroseconds(1000 + 10*motorFanSpeed);
 } else if (motorFanCommand < motorFanSpeed) {
    motorFanSpeed--;
    motorFan.writeMicroseconds(1000 + 10*motorFanSpeed);
 }
}
```

The code above works thanks to two global variables:

- *motorFanSpeed* is the current percentage sent to the ESC
- *motorFanCommand* is the user-defined percentage sent via the GUI

The percentage is then converted into the duty cycle of the PWM signal in microsecond.

```
if ( (millis() - currentMillisMotor >= 100) && !(motorFanCommand == motorFanSpeed) )
setMotorFanSpeed();
```

The *setMotorFanSpeed()* function is called in the main loop every $100\ ms$ and only if the current percentage sent to the ESC is different from the percentage sent via the GUI. The current percentage sent to the ESC can only be increased or decreased by 1 in percentage. A ramp is so created.

8.6 Buzzer Control

It is very useful to have a feedback when data are received by the demonstrator or when the board is powered up. That is why a buzzer and two LED are implemented. It is then easy to debug the Demonstrator.

- The red LED is turned on when the board is powered up
- The green LED blinks when data are received
- The buzzer rings when data are received, when no data are sent and when the board is powered up

```
void buzzerTone() {
  if (pflagTone && millis() - currentMillisTone >= BUZZER_TIME) {
    currentMillisTone = millis();
    pflagTone = false;
    digitalWrite(PIN_BUZZER, HIGH);
    digitalWrite(PIN_LED_GREEN, HIGH);
  } else if (!pflagTone && millis() - currentMillisTone >= BUZZER_TIME) {
    digitalWrite(PIN_BUZZER, LOW);
    digitalWrite(PIN_LED_GREEN, LOW);
  }
}
```

The buzzer and the green LED are enabled when *pflagTone* is equal to *true*. They are disabled after $300\ ms$.

8.7 Conclusion

The code inside the microcontroller is constituted by two loops:

- The main loop running at $2ms$ including the PID control, it is the translation of the literature review into the microcontroller
- The communication loop running at $25ms$

The function *millis()* is widely used because it allows the program to master the time. The entire code has not been explained but only the interesting parts, especially how the PID control is implemented and the communication. The code has also been optimised for controlling the turbine and the buzzer.

9 GUI programming

9.1 General

The GUI is the most important feature of the demonstrator. It must be:

- Reliable
- Easy to use
- Flexible
- As elegant as possible

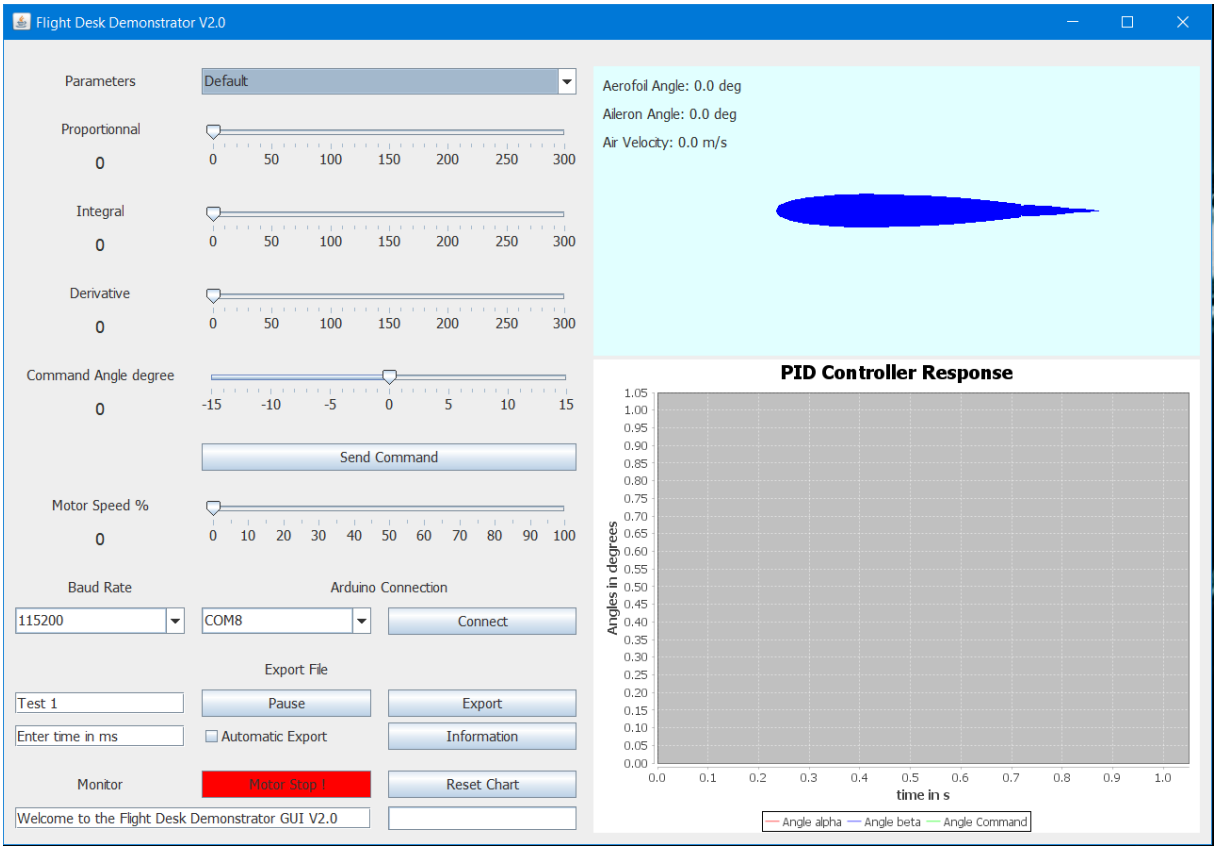


Figure 9-1 Demonstrator's GUI

The GUI can be divided into 5 parts:

- The real time animation and data measured from the board.
- The chart displaying the command angle, the AoA of the aerofoil and the flap angle
- The control panel for setting the PID, the motor speed or the command angle
- The Arduino connection panel for choosing the right COM port and the right BAUD rate
- The monitor panel for exporting the data or resetting the chart

The button *Information* displays key data about the demonstrator and how to use it. The button *Motor Stop!* allows the user to quickly stop the turbine. The GUI features a progress bar and a text bar to give feedback to the user.

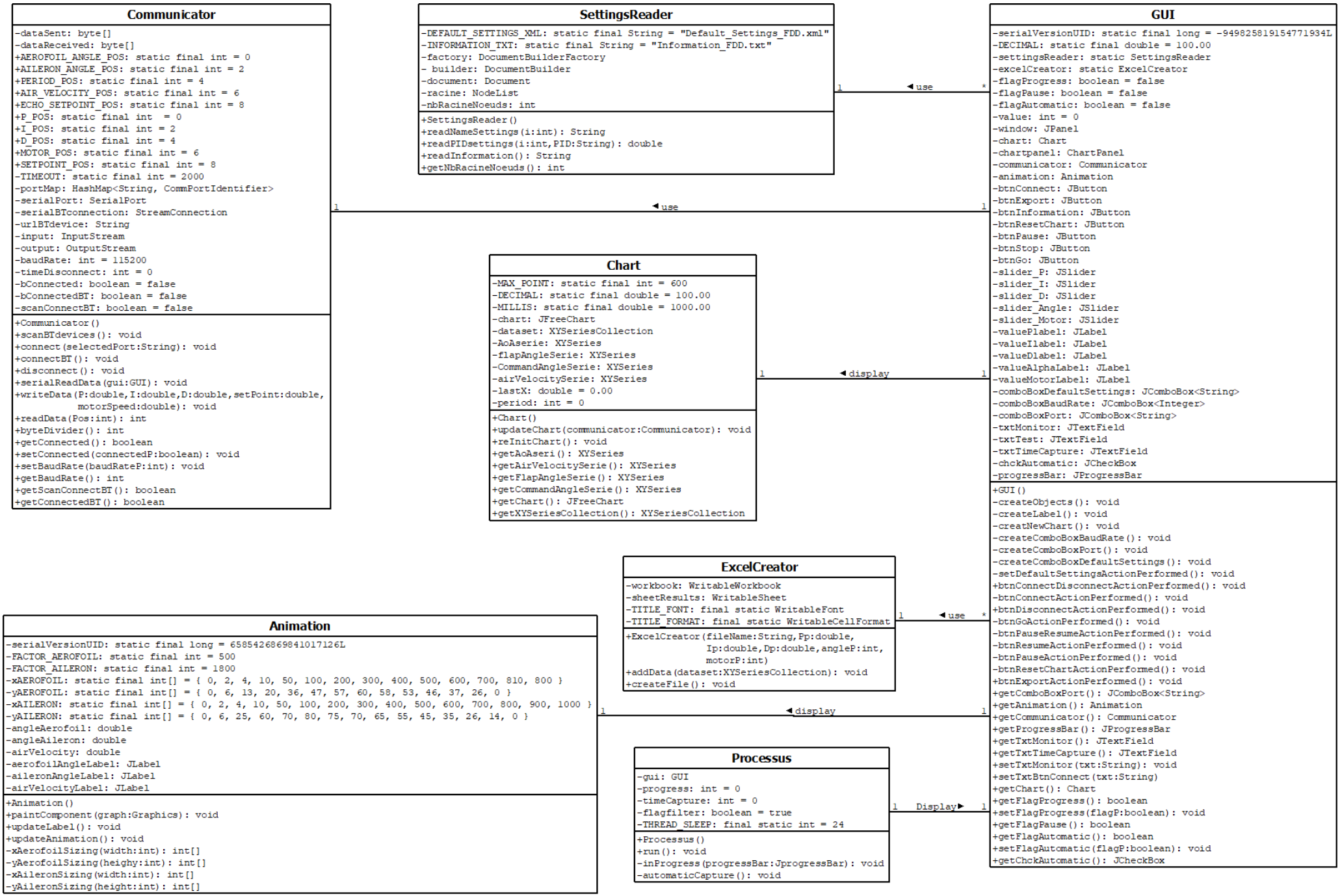
The GUI is programmed in Java® using the Eclipse® IDE. It is very common to use a java GUI for controlling an Arduino® board. Many libraries exist to simplify the programming process. The program can easily be converted into a runnable file and exported on several computers.

This section aims to explain how the GUI works. Only the key features are going to be described. The way the data are sent from the GUI or received from the microcontroller is obviously the same that method implemented within the microcontroller. This section focuses on:

- How the GUI works
- How the GUI communicates with the board
- How the chart is updated
- How the settings are stored and exported

The relations between the classes are depicted in the UML diagram shown on the next page. Each class has attributes and functions. The UML schema was created using DIA®.

The communication between the Arduino® and the computer is performed through a serial port and an USB connection. It is possible to select the Baud rate and the right COM port. Thanks to this option, the GUI is flexible and another board can be used for the demonstrator.


Communicator
-dataSent: byte[]
-dataReceived: byte[]
+AEROFOIL_ANGLE_POS: static final int = 0
+AILERON_ANGLE_POS: static final int = 2
+PERIOD_POS: static final int = 4
+AIR_VELOCITY_POS: static final int = 6
+ECHO_SETPOINT_POS: static final int = 8
+P_POS: static final int = 0
+I_POS: static final int = 2
+D_POS: static final int = 4
+MOTOR_POS: static final int = 6
+SETPOINT_POS: static final int = 8
-TIMEOUT: static final int = 2000
-portMap: HashMap<String, CommPortIdentifier>
-serialPort: SerialPort
-serialBTconnection: StreamConnection
-urlBTdevice: String
-input: InputStream
-output: OutputStream
-baudRate: int = 115200
-timeDisconnect: int = 0
-bConnected: boolean = false
-bConnectedBT: boolean = false
-scanConnectBT: boolean = false
+Communicator()
+scanBTdevices(): void
+connect(selectedPort:String): void
+connectBT(): void
+disconnect(): void
+serialReadData(gui:GUI): void
+writeData(P:double,I:double,D:double,setPoint:double,
motorSpeed:double): void
+readData(Pos:int): int
+byteDivider(): int
+getConnected(): boolean
+setConnected(connectedP:boolean): void
+setBaudRate(baudRateP:int): void
+getBaudRate(): int
+getScanConnectBT(): boolean
+getConnectedBT(): boolean
SettingsReader
-DEFAULT_SETTINGS_XML: static final String = "Default_Settings_FDD.xml"
-INFORMATION_TXT: static final String = "Information_FDD.txt"
-factory: DocumentBuilderFactory
- builder: DocumentBuilder
-document: Document
-racine: NodeList
-nbRacineNoeuds: int
+SettingsReader()
+readNameSettings(i:int): String
+readPIDsettings(i:int,PID:String): double
+readInformation(): String
+getNbRacineNoeuds(): int
GUI
-serialVersionUID: static final long = -9498258191547719 34L
-DECIMAL: static final double = 100.00
-settingsReader: static SettingsReader
-excelCreator: static ExcelCreator
-flagProgress: boolean = false
-flagPause: boolean = false
-flagAutomatic: boolean = false
-value: int = 0
-window: JPanel
-chart: Chart
-chartpanel: ChartPanel
-communicator: Communicator
-animation: Animation
-btnConnect: JButton
-btnExport: JButton
-btnInformation: JButton
-btnResetChart: JButton
-btnPause: JButton
-btnStop: JButton
-btnGo: JButton
-slider_P: JSlider
-slider_I: JSlider
-slider_D: JSlider
-slider_Angle: JSlider
-slider_Motor: JSlider
-valuePlabel: JLabel
-valueIlabel: JLabel
-valueDlabel: JLabel
-valueAlphaLabel: JLabel
-valueMotorLabel: JLabel
-comboBoxDefaultSettings: JComboBox<String>
-comboBoxBaudRate: JComboBox<Integer>
-comboBoxPort: JComboBox<String>
-txtMonitor: JTextField
-txtTest: JTextField
-txtTimeCapture: JTextField
-chckAutomatic: JCheckBox
-progressBar: JProgressBar
+GUI()
-createObjects(): void
-createLabel(): void
-creatNewChart(): void
-createComboBoxBaudRate(): void
-createComboBoxPort(): void
-createComboBoxDefaultSettings(): void
-setDefaultSettingsActionPerformed(): void
+btnConnectDisconnectActionPerformed(): void
-btnConnectActionPerformed(): void
+btnDisconnectActionPerformed(): void
-btnGoActionPerformed(): void
-btnPauseResumeActionPerformed(): void
-btnResumeActionPerformed(): void
-btnPauseActionPerformed(): void
-btnResetChartActionPerformed(): void
+btnExportActionPerformed(): void
+getComboBoxPort(): JComboBox<String>
+getAnimation(): Animation
+getCommunicator(): Communicator
+getProgressBar(): JProgressBar
+getTxtMonitor(): JTextField
+getTxtTimeCapture(): JTextField
+setTxtMonitor(txt:String): void
+setTxtBtnConnect(txt:String)
+getChart(): Chart
+getFlagProgress(): boolean
+setFlagProgress(flagP:boolean): void
+getFlagPause(): boolean
+getFlagAutomatic(): boolean
+setFlagAutomatic(flagP:boolean): void
+getChckAutomatic(): JCheckBox
Chart
-MAX_POINT: static final int = 600
-DECIMAL: static final double = 100.00
-MILLIS: static final double = 1000.00
-chart: JFreeChart
-dataset: XYSeriesCollection
-AoAserie: XYSeries
-flapAngleSerie: XYSeries
-CommandAngleSerie: XYSeries
-airVelocitySerie: XYSeries
-lastX: double = 0.00
-period: int = 0
+Chart()
+updateChart(communicator:Communicator): void
+reInitChart(): void
+getAoAseri(): XYSeries
+getAirVelocitySerie(): XYSeries
+getFlapAngleSerie(): XYSeries
+getCommandAngleSerie(): XYSeries
+getChart(): JFreeChart
+getXYSeriesCollection(): XYSeriesCollection
ExcelCreator
-workbook: WritableWorkbook
-sheetResults: WritableSheet
-TITLE_FONT: final static WritableFont
-TITLE_FORMAT: final static WritableCellFormat
+ExcelCreator(fileName:String,Pp:double,
Ip:double,Dp:double,angleP:int,
motorP:int)
+addData(dataset:XYSeriesCollection): void
+createFile(): void
Animation
-serialVersionUID: static final long = 6585426869841017126L
-FACTOR_AEROFOIL: static final int = 500
-FACTOR_AILERON: static final int = 1800
-xAEROFOIL: static final int[] = { 0, 2, 4, 10, 50, 100, 200, 300, 400, 500, 600, 700, 810, 800 }
-yAEROFOIL: static final int[] = { 0, 6, 13, 20, 36, 47, 57, 60, 58, 53, 46, 37, 26, 0 }
-xAILERON: static final int[] = { 0, 2, 4, 10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 }
-yAILERON: static final int[] = { 0, 6, 25, 60, 70, 80, 75, 70, 65, 55, 45, 35, 26, 14, 0 }
-angleAerofoil: double
-angleAileron: double
-airVelocity: double
-aerofoilAngleLabel: JLabel
-aileronAngleLabel: JLabel
-airVelocityLabel: JLabel
+Animation()
+paintComponent(graph:Graphics): void
+updateLabel(): void
+updateAnimation(): void
-xAerofoilSizing(width:int): int[]
-yAerofoilSizing(heighy:int): int[]
-xAileronSizing(width:int): int[]
-yAileronSizing(height:int): int[]
Processus
-gui: GUI
-progress: int = 0
-timeCapture: int = 0
-flagfilter: boolean = true
-THREAD_SLEEP: final static int = 24
+Processus()
+run(): void
-inProgress(progressBar:JprogressBar): void
-automaticCapture(): void
use
use
use
display
display
Display
1
*

9.2 Main process

The GUI works almost like the code within the Arduino® board. The class *Processus* extends from the class *Thread*, the latter can be considered as a loop using the function *run().*

```
public void run() {
    while (true) {
      try {
        //System.out.println();
        if (gui.getFlagProgress()) inProgress(gui.getProgressBar());
        if (gui.getFlagAutomatic() ) automaticCapture();
        if (gui.getCommunicator().getConnected()) {
          if (gui.getFlagPause() == false && !flagFilter) {
              gui.getCommunicator().serialReadData(gui);
              gui.getChart().updateChart(gui.getCommunicator());
              gui.getAnimation().updateAnimation(gui.getChart());
          }
        } else {
          if (gui.getCommunicator().getScanConnectBT() == false)
gui.getCommunicator().searchForPorts(gui);
          flagFilter = true;
        }
        Processus.sleep(THREAD_SLEEP);
      } catch (InterruptedException | IOException e) {
        gui.btnConnectDisconnectActionPerformed();
        e.printStackTrace();
      }
    }
  }
```

The loop checks first if the progress bar must be enabled and if the user wants an automatic capture. Then, if the board is connected and not taking a break, the program reads the incoming data and update the chart and the animation. In case of the board is not connected, the program is always looking for a serial port.

The Boolean *flagfilter* is used to start the updating loop (the animation and the chart) only when the progress bar reaches 100%. The first values from the Arduino® can be corrupted and they are not considered this way.

9.3 Data receiving

The buffer sent by the board is received within the communicator object. The length of the buffer is checked before considering the data.

```
public void serialReadData(GUI gui) throws IOException {
   if (input.available() == 0) {
     timeDisconnect++;
   } else if (input.available() == dataReceived.length) {
     input.read(dataReceived, 0, dataReceived.length);
     timeDisconnect = 0;
   } else if (input.available() % dataReceived.length == 0) {
     input.skip(input.available() - dataReceived.length);
     input.read(dataReceived, 0, dataReceived.length);
     timeDisconnect = 0;
   } else {
     input.skip(input.available());
     timeDisconnect = 0;
   }
   if (timeDisconnect > 50) gui.btnConnectDisconnectActionPerformed();
 }
```

The length of the buffer expected is $dataReceived.length = 10$. The buffer acts as a stack. If two buffers from the board are received closely, the buffer length is equal to 20 but the data are still correct. In this case, only the last data received are kept and the buffer is cleared.

If nothing is received while the board is connected, a counter *timeDisconnect* is incremented. The counter is reset each time something is received but above 50 attempts, the board is disconnected if nothing is received. This a protection in case of the board is not disconnected properly.

9.4 Data filtering

The data are always checked before being used to update the GUI. This also allows a better experience for the user. The data are threated within the chart object.

```
public void updateChart(Communicator communicator) {
   double periodReceived = communicator.readData(Communicator.PERIOD_POS);
   double setPoint = communicator.readData(Communicator.ECHO_SETPOINT_POS) / DECIMAL;
   double yAerofoil = communicator.readData(Communicator.AEROFOIL_ANGLE_POS) / DECIMAL;
   double yFlap = communicator.readData(Communicator.AILERON_ANGLE_POS) / DECIMAL;
   double airVelocity = communicator.readData(Communicator.AIR_VELOCITY_POS) / DECIMAL;


   if (period == 0 && periodReceived > 0 && periodReceived < 1000) period = (int) periodReceived;
   if (periodReceived == period && period > 0) {
     lastX = period/MILLIS + lastX
     AoAserie.add(lastX, yAerofoil);
     flapAngleSerie.add(lastX, yFlap);
     CommandAngleSerie.add(lastX, setPoint);
     airVelocitySerie.add(lastX, airVelocity);
   } else if (periodReceived != 0) {
     lastX = period/MILLIS + lastX;
   }
 }
```

The chart is constituted with three curves:

- The command angle
- The AoA of the aerofoil
- The flap angle

The *setPoint* received is the one sent by the user using the GUI. This allow to easily debug the Arduino®. Indeed, if the set point (command angle) is different from the one sent previously, it means something is going wrong through the connection.

The microcontroller always sends the *period* ($25ms$). This is used to check the data: the chart is updated only when the *period* received is equal to the previous one. The right value of the *period* is taken once. A value of period superior to 0 and inferior to $1000ms$ is expected.

The chart is not updated If nothing is received while the board seems connected. However, if corrupted data are received the abscises are still updated to always keep the right timeline since the data are always received every $25ms$.

9.5 Bluetooth connection

The GUI can be connected using an USB port or a Bluetooth connection. Both are serial communication protocols. The Bluetooth connection is more complicated because of the intermediaries. The Arduino® board must first send the data to the Bluetooth module via its serial port. The Bluetooth module HC-05 must be known (appaired) and connected to send the data to the computer. The Bluetooth connection makes the demonstrator more convenient to use.

The Bluetooth module HC-05 is as easy to pair as a Bluetooth speaker. The password is 1234 for most HC-05. The name of the HC-05 module mounted on the board is “FLIGHT_DESK_DEMONSTRATOR”. The program will always be looking for this name before trying a connection. It means in case of demonstrator’s Bluetooth module dysfunction; the new Bluetooth module must be renamed.

```
public void scanBTdevices() throws InterruptedException, BluetoothStateException {
      if ( LocalDevice.isPowerOn() && (urlBTdevice == null) ) {
        scanConnectBT = true;
    LocalDevice.getLocalDevice().getDiscoveryAgent().startInquiry(DiscoveryAgent.GIAC,
new DiscoveryListener() {
      public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
            try {
          if (btDevice.getFriendlyName(false).matches("FLIGHT_DESK_DEMONSTRATOR"))
{
              urlBTdevice = "btspp://" + btDevice.getBluetoothAddress() +
":1;authenticate=false;encrypt=false;master=falsel";
              scanConnectBT = false;
          }
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
```

It possible to see that the Bluetooth module is threated as server since a *http* request is sent.

The code above looks into the known devices of the computer to find the Bluetooth module “FLIGHT_DESK_DEMONSTRATOR”. The research is launched only if the Bluetooth on the computer is turned on and if the HC-05’s address (url) has not been collected yet.

The process to find the Bluetooth module needs time. The *Thread* (loop) is stopped as long as the connection is not established. If the time connection is superior to 10 seconds, an error occurs, and the loop is broken.

```
while (scanConnectBT) {
      Thread.sleep(250);
       }
```

9.6 Data export

The GUI offers two ways to export the chart into an Excel® file. The first option is the manual option by using the button “Export”. The export includes only the data displayed into the chart. The second way to export data is the automatic export. The automatic export works only if the box “automatic export” is checked and if a value in milliseconds is entered inside the text box. The automatic export starts when a new command angle is sent by using the button “Send Command”. At the end of the user-defined time, the data displayed into the chart are exported into an Excel® file.

The Excel® file can be named before the export and it is located where the GUI application is located.

The following information are included into the exported file:

- Time in second
- The AoA of the aerofoil
- The flap angle
- The command angle
- The airflow velocity
- The PID settings
- The motor speed in percentage

The curves drawn into the chart are based on three lists with a maximum size set to 600 points.

The lists are written into the Excel® file as follow:

```
public void addData(XYSeriesCollection dataset) throws RowsExceededException, WriteException {


    int starLine = 4;
    for(int i = 0; i < dataset.getSeriesCount(); i++) {
      for(int j = 0; j < dataset.getSeries(i).getItemCount(); j++) {
        sheetResults.addCell( new Number(2*i, starLine+j, (double) dataset.getSeries(i).getX(j)) );
        sheetResults.addCell( new Number(2*i+1, starLine+j, (double) dataset.getSeries(i).getY(j)) );
      }
    }
 }
```

The values are taken from the object chart.

9.7 Information reading

The button “Information” opens a message box displaying a text with information regarding the demonstrator and how to use it. The text is not saved in the source code of the GUI but within a text file compressed into the *.jar* file. The *.jar* file is basically the java application’s extension. In fact, it is possible to have directly access to the text file by opening the *.jar* with a file compressor software such as WinRar®.

The *.jar* is a compressed file including the code source, the libraries and the useful files such as pictures or text files. This type of file is read by JAVA® to open the application.

It is possible to modify the text file “Information_FDD” to add new information by opening it directly inside the *.jar* and saving the modifications. The GUI is then flexible. According to the code reading the text file, a line break stops the reading:

```
public String readInformation () {
   try{
     InputStream flux = getClass().getResourceAsStream("/" + INFORMATION_TXT);
     InputStreamReader lecture = new InputStreamReader(flux);
     BufferedReader buff = new BufferedReader(lecture);
     String ligne;
     String text = "";
     while ( (ligne = buff.readLine())!=null ){
       text = text + "\n" + ligne;
     }
     buff.close();
     return text;
  } catch (Exception e) {
    e.printStackTrace();
     return null;
    }
  }
```

9.8 Default settings storage

The default PID settings are a combination of PID values obtained according to the calculation methods depicted in 2.8. The own settings of the Author are also included. It is then possible to select a setting with the combo box. They are automatically sent once selected. This GUI feature allows the user to quickly show the system response according to special settings.

The defaults PID settings are not store within the code source of the GUI but into an XML file. It very common to store basic data using this type of file. An XML file is a text-based file with a rigorous format. It is so readable by a machine or a human.

It is also possible to modify it following the same way of the previous text file by opening the *.jar* file. The file is named *Default_Settings_FDD.*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- DTD -->
<!DOCTYPE repertory [
        <!ELEMENT repertory (settings*)>
        <!ELEMENT settings (p, i, d)>
        <!ELEMENT p (#PCDATA)>
        <!ELEMENT i (#PCDATA)>
        <!ELEMENT d (#PCDATA)>
]>
<!-- Corps -->
<repertory>
        <settings name="Default">
                <p>1.00</p>
                <i>0</i>
                <d>0</d>
        </settings>
        <settings name="Other 1">
                <p>0</p>
                <i>1.00</i>
                <d>0</d>
        </settings>
</repertory>
```

It is possible to add/remove a setting by copy-pasting the block in red and by giving a new name at the tag *settings name*. It is another example of the flexibility of the demonstrator.

9.9 Conclusion

This section aims to explain the features of the GUI. The entire code has not been explained but only the interesting parts especially the communication between the computer and the demonstrator. The communication can be performed using Bluetooth or a basic USB cable.

The communication process includes safety checks regarding the incoming data. These features allow the user to use the demonstrator continuously without any annoying interruption because of a communication error.

Another key feature of the GUI is the possibility of post-processing by exporting the data directly into an Excel® file. All the data displayed into the chart are saved but also the parameters. The export can be achieved manually or automatically by entering a certain amount of time in milliseconds.

The GUI is also scalable. It is possible to add new default settings or information for the user by modifying the text file or the XML file directly in the *.jar* file.

10 Flight Desk Control Demonstrator assessment

10.1 General

The aim of this part is to compare the theoretical model using Matlab® with the real system behaviour. The performance criteria in temporal analysis will be compare:

- Stability
- Steady-state error e_{ss}
- Settling time at where the output remains 2% of its final value t_s
- Overshoot o_s in percentage
- Peak time t_p when the output is at its greatest value
- Rise time t_r when the output increases from 10% to 90% of its final value

Several correction methods depicted in 2.8 are will be tested on the demonstrator. The first step is to create a model of the system. Using equation (**6**-**11**) in Laplace notation and:

$$\frac{I}{qS} = I_q = 0.790\ s^2.mm.deg^{-1}$$

$$l_0(a_2)_0 + cm_0 = B = 2.33\ mm.deg^{-1}$$

$$\frac{v}{qS} = v_q = 0.8\ s.mm.d^{-1}$$

$$l_0(a_1)_v = A = 2.36\ mm.deg^{-1}$$

A typical second order system is obtained [23]:

$$F(s) = \frac{\alpha}{\delta} = \frac{\frac{B}{I_q}}{s^2 + \frac{v_q}{I_q}s + \frac{A}{I_q}} \qquad (10\text{-}1)$$

The value of v_q is difficult to obtain. It is obtained step by step with the demonstrator itself to get acceptable theoretical response compare to the reality. This point will be discussed later. The meaning of the term comes from the friction inside the bearings, the potentiometer and the deformation of the flexible joint. Due to the complexity to determine theoretically this value, it is assumed and correlated with the experimental data.

10.2 Second Order system and theoretical performances

The general second order system form is:

$$F(s) = \frac{K\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \quad (10\text{-}2)$$

Where:

- ω_n is the natural damped frequency and the natural frequency is $\omega_d = \sqrt{1-\xi^2}$
- ξ is the damping factor $0 \leq \xi \leq 1$
- K is a constant

The different value of the coefficients can be obtained by identification:

Table 10-1 Second order Model coefficients' value

Model Coefficient		
Dimension	Value	unity
ωn	1.73	rad.s-1
ξ	0.29	
SQRT(1-ξ^2)	0.96	
K	0.99	
ωd	1.65	rad.s-1
σ=ξωn	0.51	rad.s-1

It is possible to calculate the theoretical performances using [23] in open-loop:

Overshoot $o_s$in percentage:

$$o_s = 100e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}} \quad (10\text{-}3)$$

Peak time t_p:

$$t_p = \frac{\pi}{\omega_d} \quad (10\text{-}4)$$

Settling time at 2% t_s:

$$t_s \cong \frac{4}{\sigma} \quad (10\text{-}5)$$

Rise time t_r for $0.3 \leq \xi \leq 0.8$:

$$t_r \cong \frac{2.16\xi + 0.60}{\omega_n} \qquad (10\text{-}6)$$

The results from theoretical model in open-loop are given below:

Table 10-2 Performances from the theoretical model in open-loop

Performances		
Dimension	Value	unity
os	38.17	%
ess	0.00	deg
Stability	Yes	
tp	1.82	s
ts	7.90	s
tr	0.71	s

A continuous model in Matlab® is used. The servomotor model is also included in the general model as a first order system. However, the time constant T is very small compare to the others and it can be neglected. The windup phenomenon due to the integrator is removed by clamping within the PID block. The angle $\Delta\delta$ calculated from the PID is added the angle δ from the static analysis (aerodynamic model). The PID output is saturated according to the system geometry $\pm 15°$. The angle δ of the flap and the angle α after the model from the aerofoil are monitored.

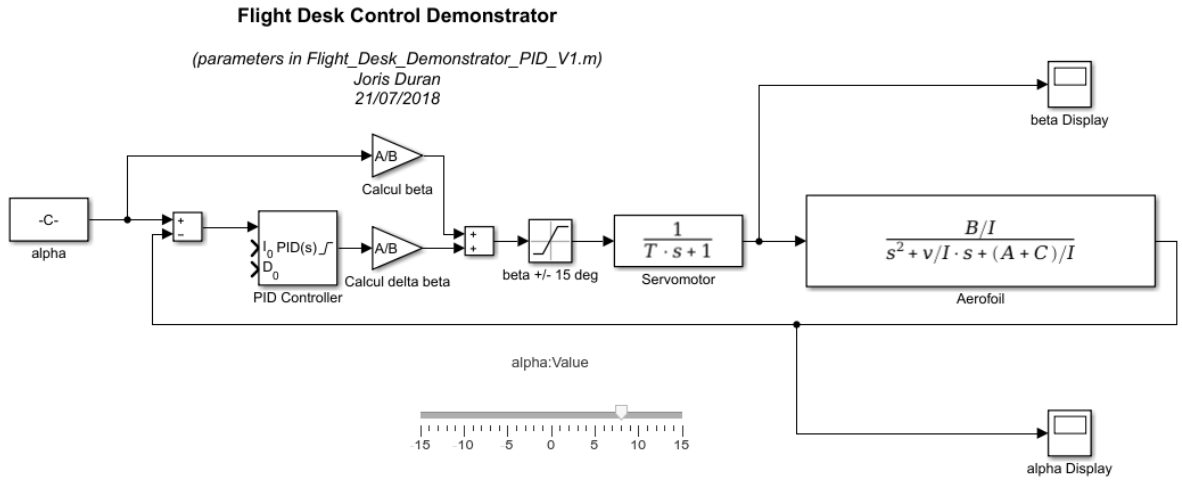


Figure 10-1 Model on Matlab®

Where $\frac{A}{B} \cong 1$ is the $\frac{\partial\delta}{\partial\alpha}$ from the aerodynamic model in static.

The variables are initialised following the code in 11Appendix H. The variable C will be explain later. It is equal to zero for the moment and aims to explain a difference between the theoretical model and the reality.

The theoretical system response is given below for a command angle equal to 8°.

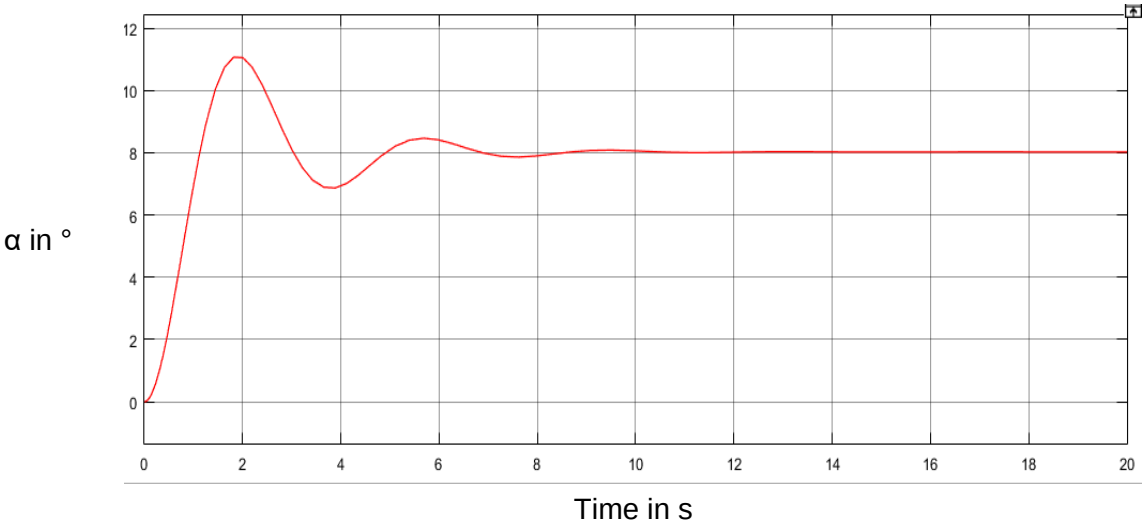


Figure 10-2 System response: AoA α, command angle 8°

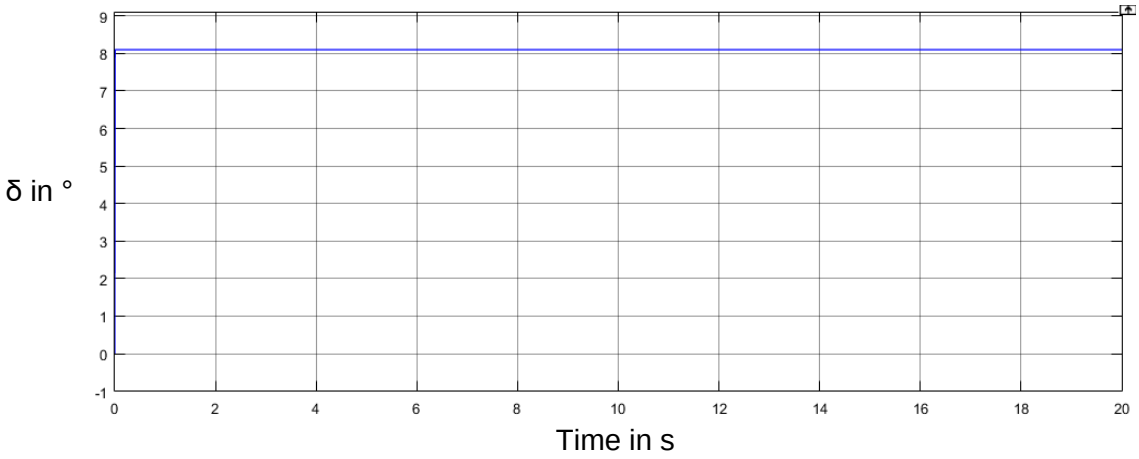


Figure 10-3 System response: flap angle δ, command angle 8°

The theoretical model in open-loop is:

$$F(s) = \frac{2.96}{s^2 + 1.01s + 2.98}$$

The next step is to get a model by identification from the real system and to compare the performances and investigate where these differences come from.

10.3 Experimental data: second order identification in Open-loop

10.3.1 Second order identification

After adjusting the demonstrator, the tests are led, and it is possible to see below an example of the system response for a command angle of $8°$ and $-8°$.

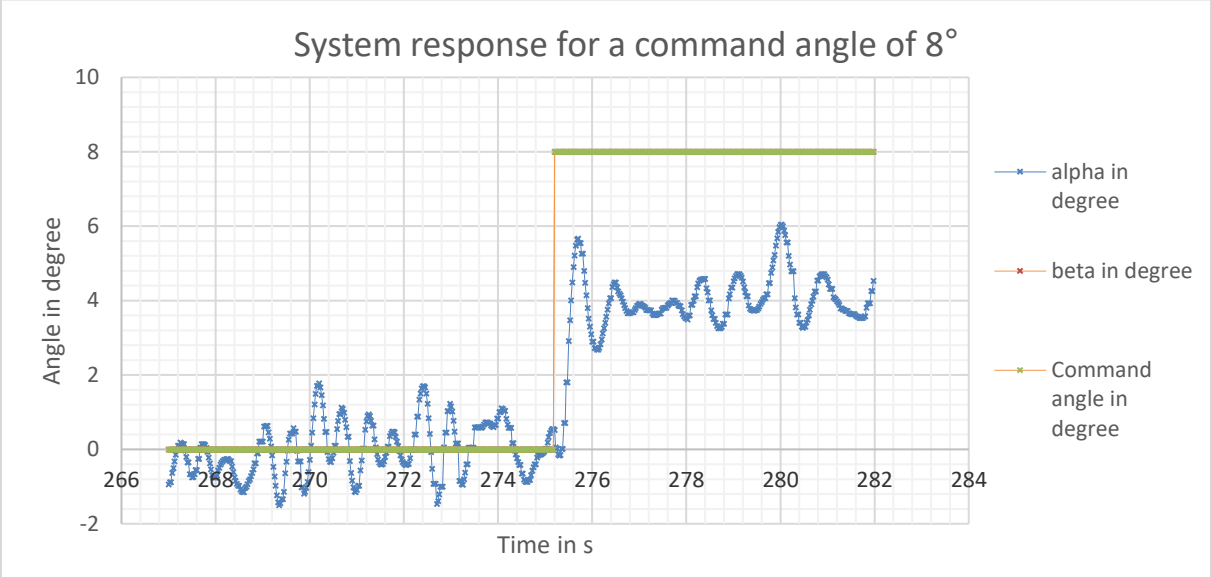


Figure 10-4 Example of system response

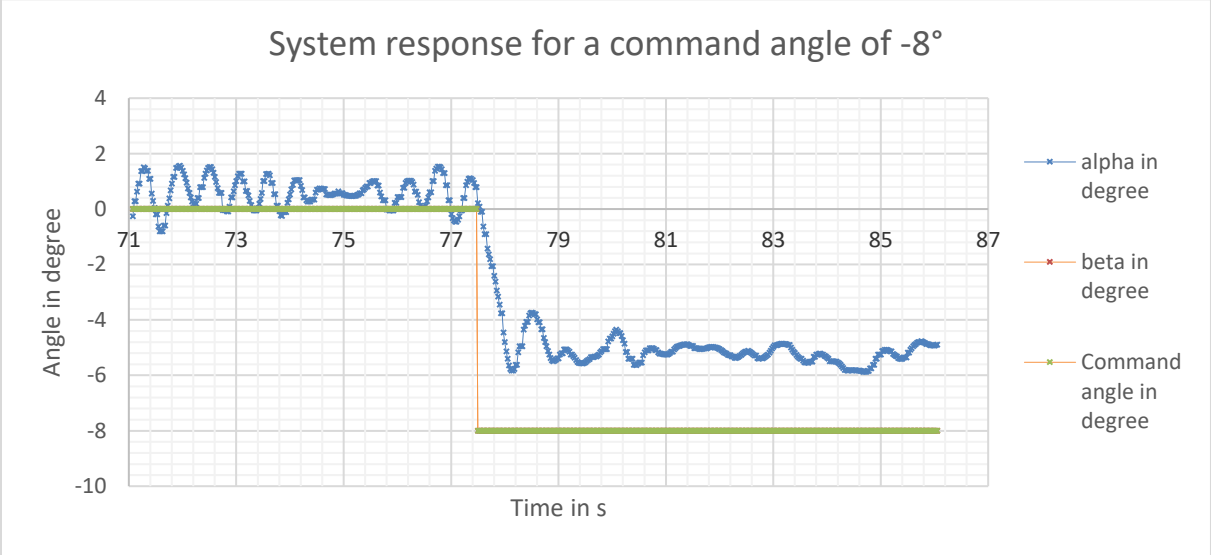


Figure 10-5 Another example of system response

It is possible to notice the response is symmetric as expected.

The values of $8°$ is chosen because it is not the model limit. Indeed, beyond $10°$ the aerofoil stalls. These values are also very convenient because it is easier to catch the response system.

It is possible to notice the response is symmetric as expected. However, the aerofoil is perturbated and oscillates even at the neutral AoA where the aerofoil should be perfectly stable. The reasons of the perturbations are explained during the comparison between the theoretical and the real system response. This part focuses on the system response identification to a second order system.

The system response is still readable, and it is possible to notice a high steady-state error. The model is obviously not complete. Other disturbances have been neglected and those can lead to such a high steady-state error. They are investigated later during the model comparison with the real system.

Based on the system response to a command angle of $8°$, the time response shape gives a typical second order system.

$$F(s) = \frac{K\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \qquad (10\text{-}7)$$

Table 10-3 Second order Model coefficients' value

Model Coefficient		
Dimension	Value	unity
ωn	9.48	rad.s-1
ξ	0.27	
SQRT(1-ξ^2)	0.96	
K	0.50	
ωd	9.12	rad.s-1
σ=ξωn	2.59	rad.s-1

Using the same equation as the theoretical model, the performances are:

Table 10-4 Performances from the empirical model in open-loop

Performances		
Dimension	Value	unity
os	41.00	%
ess	4.00	deg
Stability	Yes	
tp	0.33	s
ts	1.54	s
tr	0.13	s

10.3.2 Empirical model characteristics

The empirical model in open-loop is:

$$F(s) = \frac{44.94}{s^2 + 5.12s + 89.87}$$

Thanks to the empirical model, it is possible to further characterize the system and to find approaches to tune the PID terms. The frequential analysis of the system gives useful data for the correction process.

The poles are given by:

$$s_{1,2} = -\sigma \pm j\omega_d \tag{10-8}$$

Then, $s_1 = -2.56 + j9.13$ and, $s_2 = -2.56 - j9.13$. they are plotted on the s-plane:

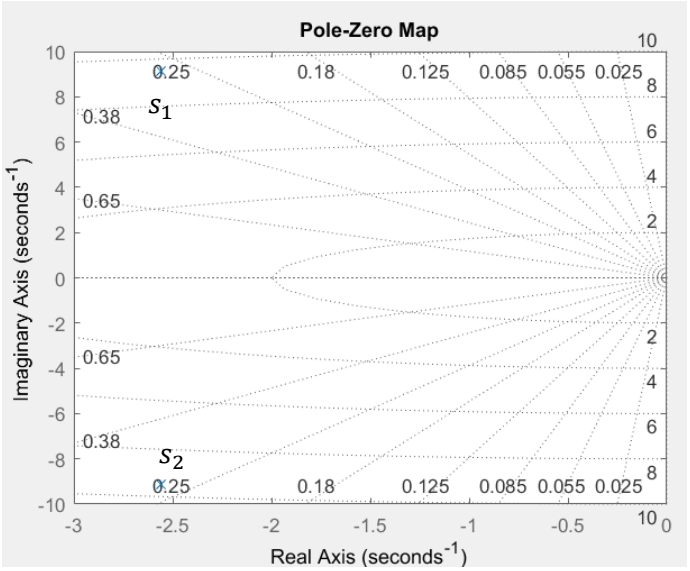


Figure 10-6 Poles on the s-plane

Showing the poles and the zeros of the system gives useful data about how the system reacts. The s-plane shows directly the phase and the magnitude of pole/zero and whether they satisfy the performances constraints. This diagram is also useful to lead a root locus analysis to find an adequate correction for the system.

It is interesting to plot de Bode diagram of the $F(s)$. The Bode diagram of the empirical system is given below using Matlab®. Bode diagrams plot the frequency response of: the magnitude (in dB) and the phase (in °) of the system response as a function of frequency. They are used to analyse the system properties.

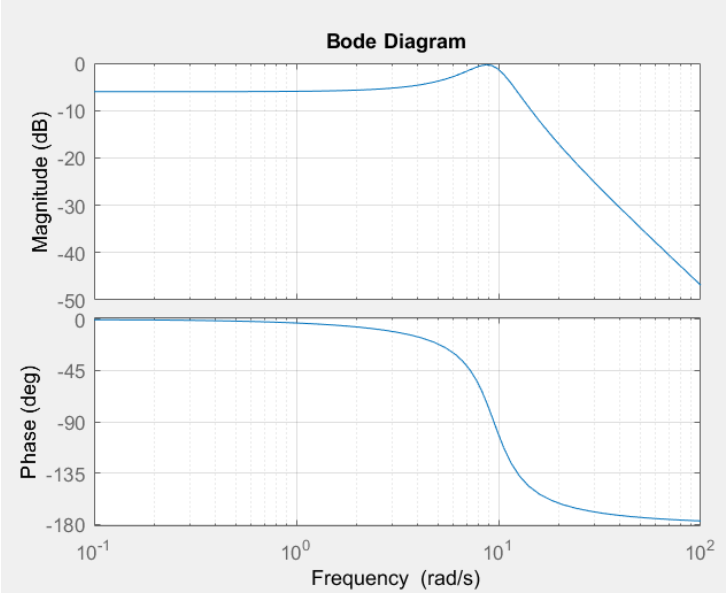


Figure 10-7 Bode diagram of the empirical system

The Nyquist plot can be also used to assess of the stability of the system. The Nyquist plots the frequency response of a dynamic system model. The system is stable according to the Nyquist stability criterion. This is a particular case because the system has no pole or zero with positive real part: the curve is on the right side of the point $(-1, j0)$ when $\omega \in [0, +\infty[$.

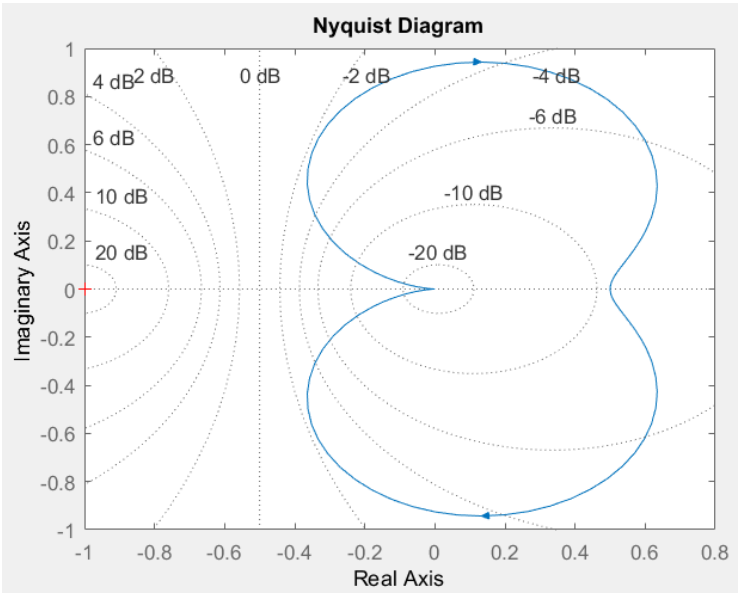


Figure 10-8 Nyquist Plot of the empirical system

The Gain Margin GM and the Phase Margin PM can be extracted from those two diagrams:

- $GM = +\infty$ in this case. It is the difference between the gain at $\omega_{180°}$ and $0\ dB$.
- $PM = +\infty$ in this case. It is the difference between the phase at ω_c at $0\ dB$ and $-180°$

Those two values must be positive to get a stable system. An infinite GM is normal for a second order system. The infinite margins have no real physical meaning. The system is so expected to be very stable even in closed-loop with high correction gain. A phase margin is usually between $40°$ and $60°$. A gain margin is usually between $3.5dB$ and $15dB$ [23].

It is possible to get closed-loop frequency response from the open-loop response thanks to the Nichol chart. Each point on the plot corresponds to a certain frequency.

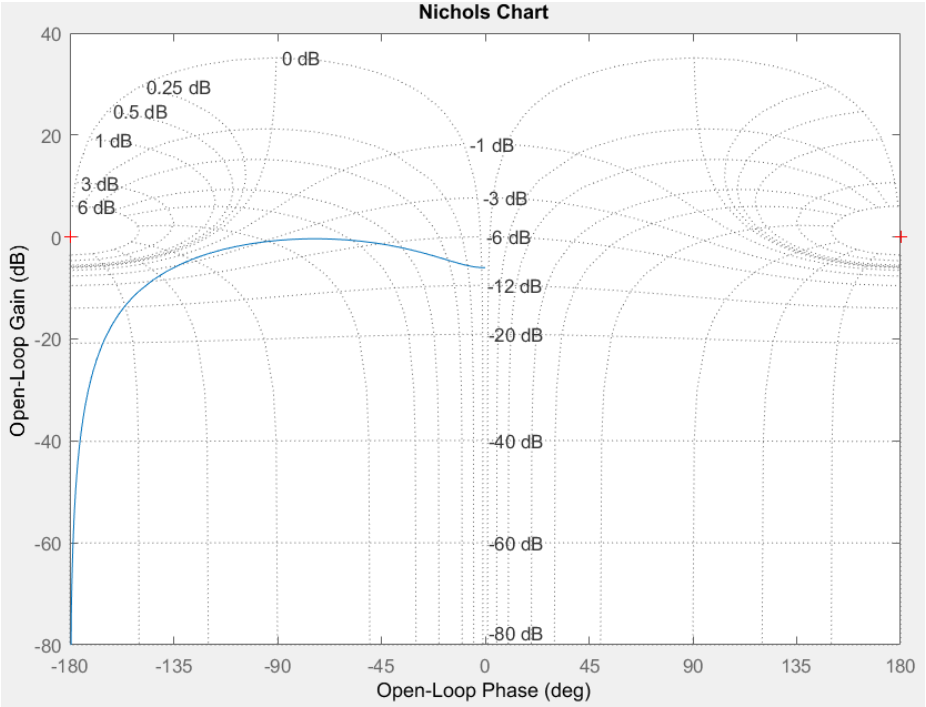


Figure 10-9 Nichol chart

The open loop properties can be read on the axis and the close loop properties on the isogain and the isophase curves.

10.3.3 Comparison between empirical and theoretical model

The performances between the theoretical and empirical model are summarised below:

Table 10-5 Performances comparison between the empirical and theoretical model

Performances comparison				
Dimension	unity	Theoretical values	Empirical values	Comparison in %
os	%	38.17	41.00	7.4
ess	deg	0.00	4.00	50
Stability		Yes	Yes	Idem
tp	s	1.82	0.33	81.8
ts	s	7.90	1.54	80.4
tr	s	0.71	0.13	82.4

The values from the theoretical model are far away from the empirical model. The theoretical model is slower and has no error steady-state error. It seems it is not complete, and the inertia term is over considered.

However, the aerofoil is perturbated and oscillates even at the neutral AoA where the aerofoil should be perfectly stable. In fact, the wind tunnel is obviously not perfect in term of geometry. The turbine is quite close to the aerofoil, so any small variation from the turbine creates perturbations. A duct around the propeller should reduce the perturbation. The presence of the Pitot tube as well as head screws inside the wind tunnel create perturbations. Finally, the aerofoil itself and its rotational axis create disturbances. The side-effect is also a source of perturbation. All these phenomena have been neglected during the analysis to get a simple model to work with. It is one of the wind tunnel design achievement to keep these perturbations as low as possible.

The system shows a high steady-state error. The model is obviously not complete. Other disturbances have been neglected those can lead to such a high steady-state error, especially:

- The dangling wires from the servomotor inside the aerofoil
- The flexible joint that can act as rotational spring
- The crank cannot be adjusted to balance perfectly the aerofoil
- The overpressure above/below the aerofoil when the aerofoil's AoA increases/decreases due to the low clearance between the aerofoil and the wind tunnel top/bottom panel of the test chamber

The phenomena can lead to a torque against the aerofoil rotation increasing with the AoA. It is possible to complete the theoretical model by adding a term depending on AoA α to the righ side of the equation (6-11).

$$\frac{I}{qS}\ddot{\alpha} = -\alpha l_0(a_1)_v + C\alpha + \delta l_0(a_2)_0 + \delta cm_0 - \frac{v}{qS}\dot{\alpha} \qquad (10\text{-}9)$$

Where C is the sum of the imperfections increasing with the AoA. An empirical value of:

$$C = 2.35\ N.mm.deg^{-1}.q^{-1}.S^{-1}$$

An important source of error comes from the airflow velocity. The velocity used for the theoretical model leans on the motor datasheet. The motor and the propeller can have a different behaviour inside the wind tunnel. The Pitot tube cannot be used to measure this velocity accurately with the electronics involved and there are too much noises even with a digital filter. However, this value was checked using CFD and the datasheet is usually reliable.

Another source of error comes from the aerofoil assembly inertia. Indeed, the value is given by Catia® by applying usual density on the different materials. However, these densities can differ from the CAD model and the software assumes a perfectly homogenous material. This is obviously not the case with a 3D printed aerofoil.

As shown previously, some data used for the theoretical model were checked using CFD:

- The different aerodynamic coefficient
- The aerofoil dimensions
- The airflow velocity and viscosity
- The aerodynamic centre location

The parameters likely to be wrong at this moment are:

- The aerofoil inertia I assembly
- The viscous friction v inside the pivot liaison, it was already an empirical data
- The constant C

An attempt is led to find whether changing these coefficients can give a better reliable model and then create a semi-empirical model.

The final values are:

- Aerofoil assembly Inertia $I = 100\ Kg.mm^2$
- Viscous friction coefficient $\upsilon = 0.3\ N.mm.deg^{-1}.s^{-1}.q^{-1}.S^{-1}$
- $C = 2.35\ N.mm.deg^{-1}.q^{-1}.S^{-1}$

The response and the performances are given below:

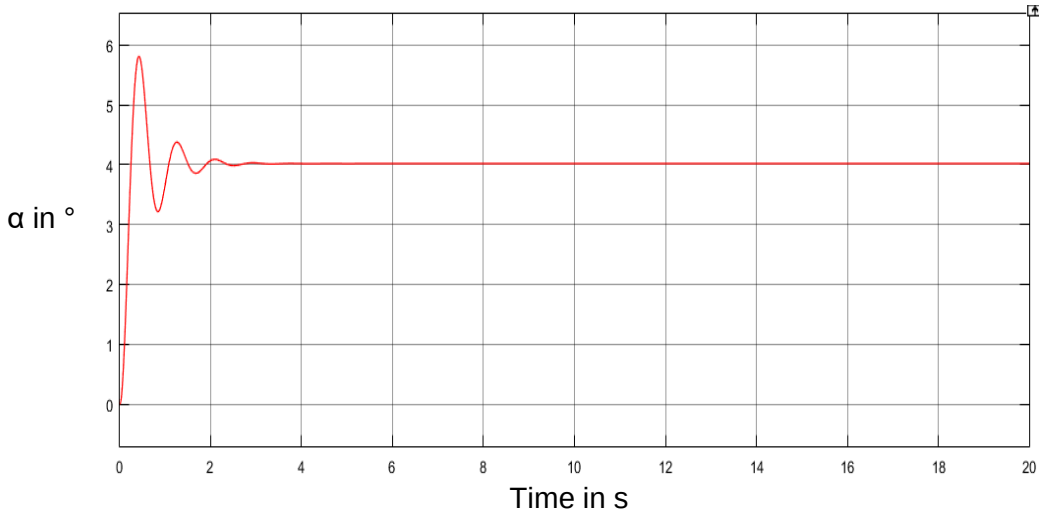


Figure 10-10 System response: AoA α, command angle 8°

Table 10-6 Performances comparison between the empirical and theoretical model

Performances comparison				
Dimension	unity	Semi-theoretical values	Empirical values	Comparison in %
os	%	36.48	41.00	12.4
ess	deg	4.00	4.00	0
Stability		Yes	Yes	Idem
tp	s	0.34	0.34	0.1
ts	s	1.36	1.54	13.2
tr	s	0.13	0.13	4.5

The performances parameters are significantly closer to the empirical model and the overall response shape looks closer to the reality. This second analysis allows the Author to have a better idea of the different sources of error and to not only lean on theoretical data especially from CAD software when it is already built with approximations.

10.4 Closed-loop tuning methods comparison

10.5 General

This section aims to give example of K controllers keeping in mind the following criteria [23]:

- Closed-loop stability
- Decreased sensitivity to system variations
- Steady-state error equal to zero
- Dynamical response
- Robust stability

The characteristics of the open-loop have been established. It is now possible to implement the feedback control with a PID correction. The majority of SISO system including a feedback loop can be depicted as follow:

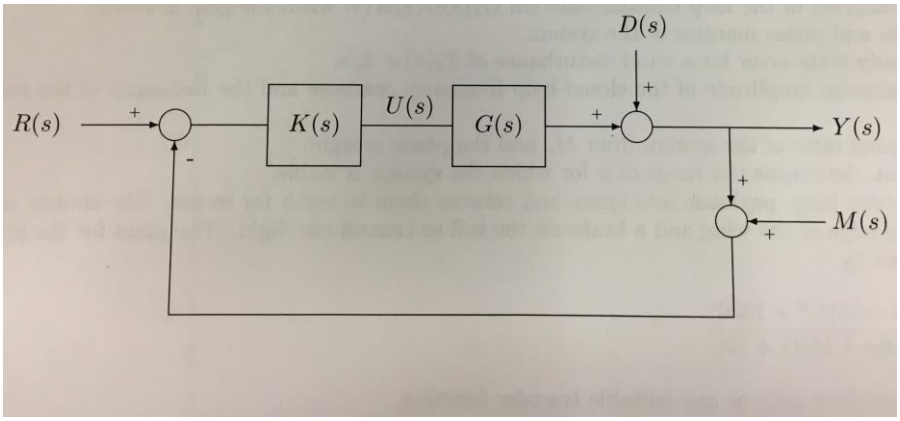


Figure 10-11 General SISO system [23]

Where $R(s)$ is the reference, $K(s)$ is the controller with the correction, $G(s)$ is the plant (the model), $Y(s)$ is the system output and $D(s)$ is a disturbance signal. $M(s)$ is the measurement error and can be neglected because it is very low.

According to the schematic, it is possible to define the transfer function $T(s)$ in closed-loop:

$$T(s) = \frac{G(s)K(s)}{1 + G(s)K(s)} \qquad (10\text{-}10)$$

And the sensitivity function $S(s)$:

$$S(s) = \frac{1}{1 + G(s)K(s)} \tag{10-11}$$

Where:

$$S(s) + T(s) = 1 \tag{10-12}$$

The sensitivity function is defined as the sensitivity of the system to variation in some parameters. A well-controlled system should be insensitive to parameters variation [23].

The transfer function shows the system capacity to track the input reference. Both terms $S(s)$ and $T(s)$ can be influenced by the controller and must be as small as possible [23].

However, the equation (10-12) means the controller can only get a compromise between a good reference tracking and a good disturbance rejection. In practice, the $|D(s)|$ spectrum is expected to show higher frequency than the $|R(s)|$ spectrum. So that the conflict between $T(s)$ and $S(s)$ can be resolved by designing in the frequency domain. $|T(s)|$ can be kept small at high frequency and $|S(s)|$ small at low frequency [23].

It is worth to keep in mind the conflict between $T(s)$ and $S(s)$ because it is now known the aerofoil inside the wind tunnel is perturbated. On Figure 10-4 such a periodic perturbation is clearly visible. The calculated frequency is $\omega_p = 8.38\, rad.s^{-1}$. $|T(s)|$ is then expected to be small for $\omega > \omega_p$ and $|S(s)|$ small for $\omega < \omega_p$.

Another issue pointed out from Figure 10-4 is the steady-state error. It is obvious that it should be equal to zero and this can be achieved using the Integral term of the PID control.

Several controllers will be tested in this section including, P, PI, PD and PID controller. The previous empirical model is used to tune the PID terms.

10.5.1 Ziegler-Nichols PID tuning for the demonstrator

The Ziegler-Nichols tuning method involved empirical tests directly on the benchtop. A feedback loop is implemented within the controller with only a proportional correction. The gain K_p is increased step by step up to such a value where the system becomes unstable or marginal-stable. This value noted K_u and the period of the oscillation is noted T_u. According to 2.8.1, it is possible to tune the PID term to get a balance behaviour between reference tracking and disturbance rejection. The Ziegler-Nichols tuning method is widely used in the industry and many variants exist.

Following this method, the values of K_u and T_u are:

- $K_u = 0.67$
- $T_u = 0.58\ s$

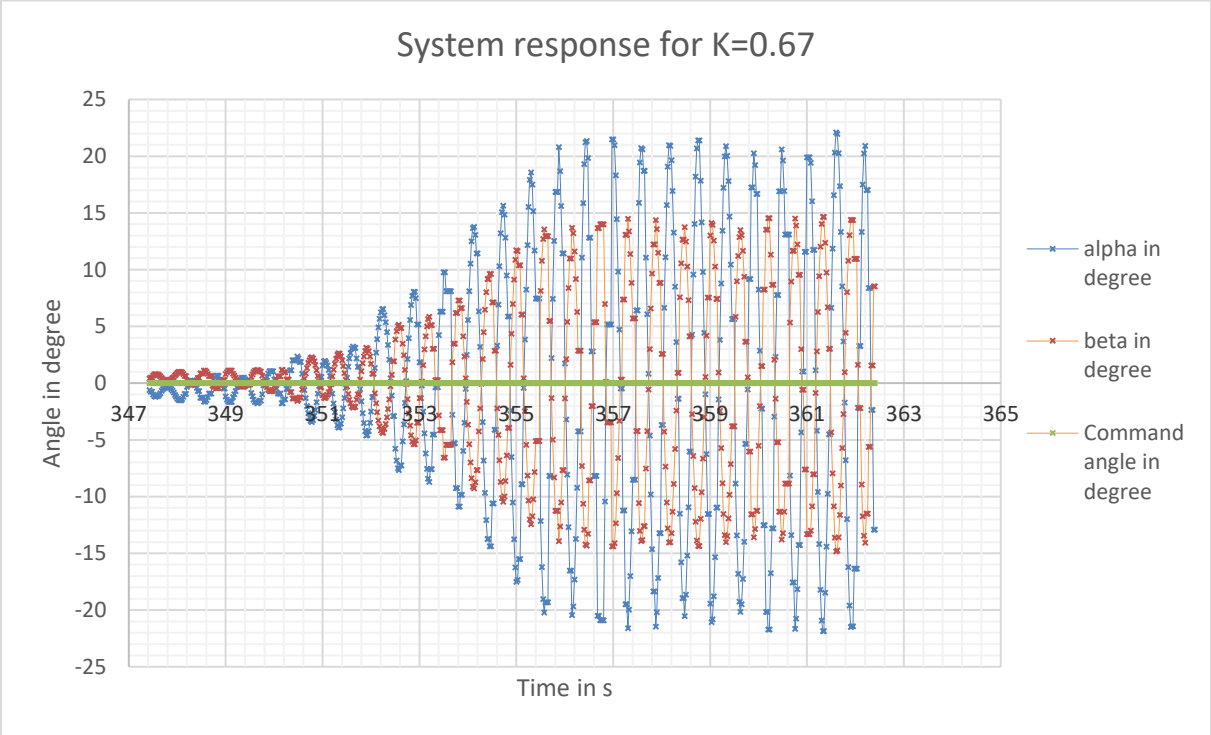


Figure 10-12 Ziegler-Nichols tuning method

The PID terms are calculated and summarised below:

Table 10-7 Ziegler-Nichols settings

Controller parameters	Kp	Ki	Kd
P	0.34		
PI	0.27	0.58	
PID	0.40	1.39	0.003

The different PID settings from the Ziegler-Nichols tuning method are tested. The performances from each setting are summarised in Table 10-8.

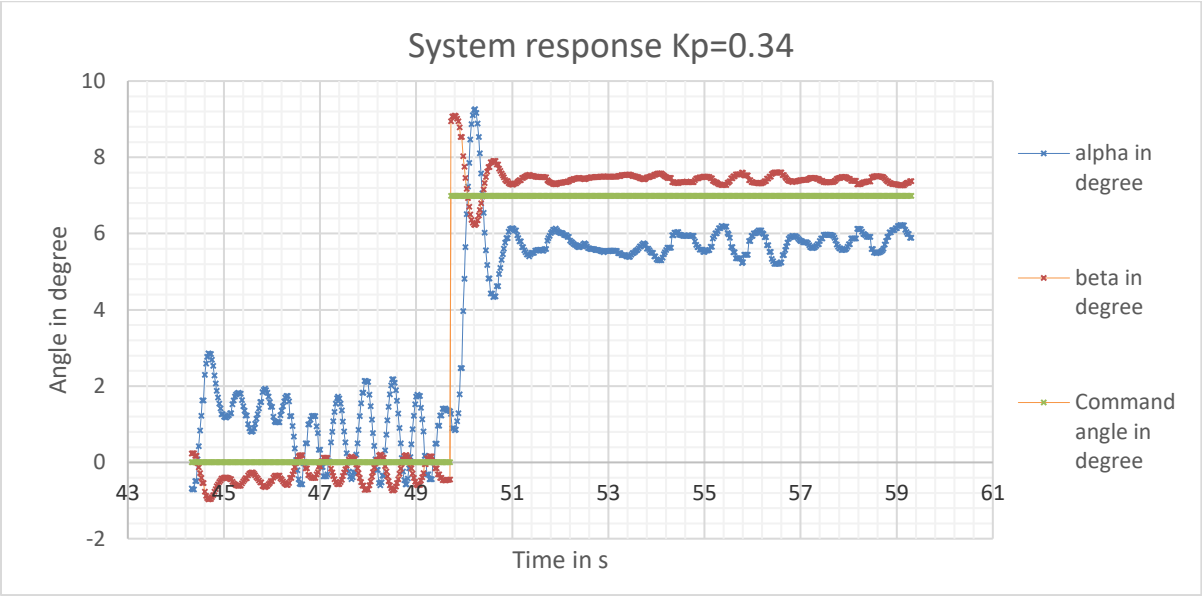


Figure 10-13 System response from Ziegler-Nichols tuning Kp=0.34

The proportional correction shows a high overshoot and a steady-state error. However, the time response is better compare to the open-loop response. It is possible to notice the proportional controller reduces the steady-state but cannot make it equal to zero.

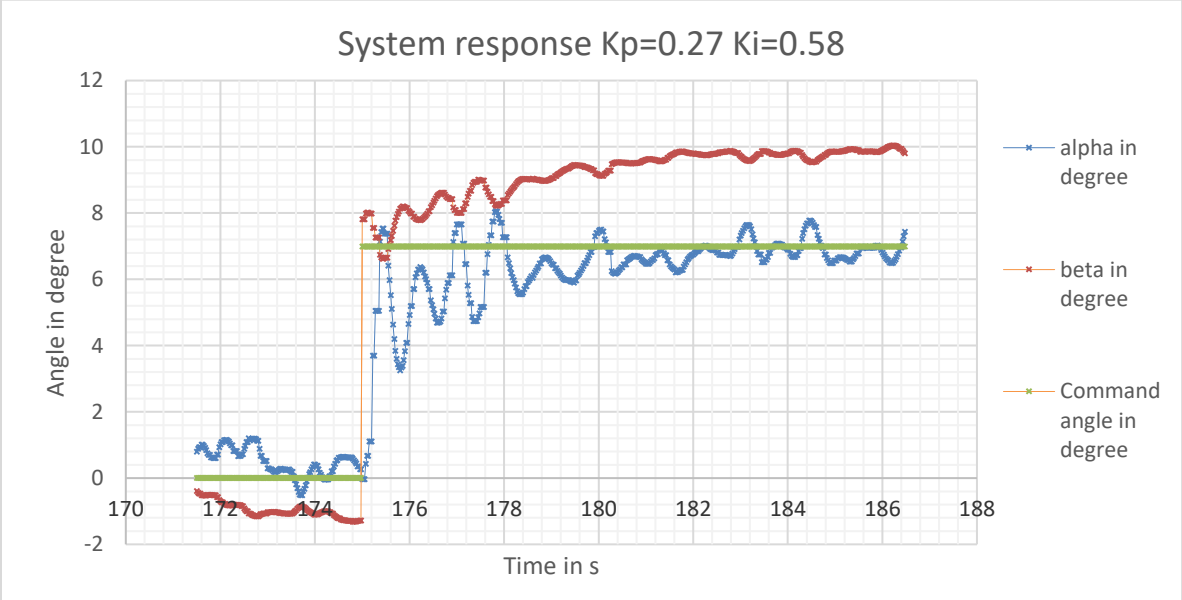


Figure 10-14 System response from Ziegler-Nichols tuning Kp=0.27 Ki=0.58

The PI controller shows a reduced overshoot and a steady-state error equal to zero. However, the transient response is longer.

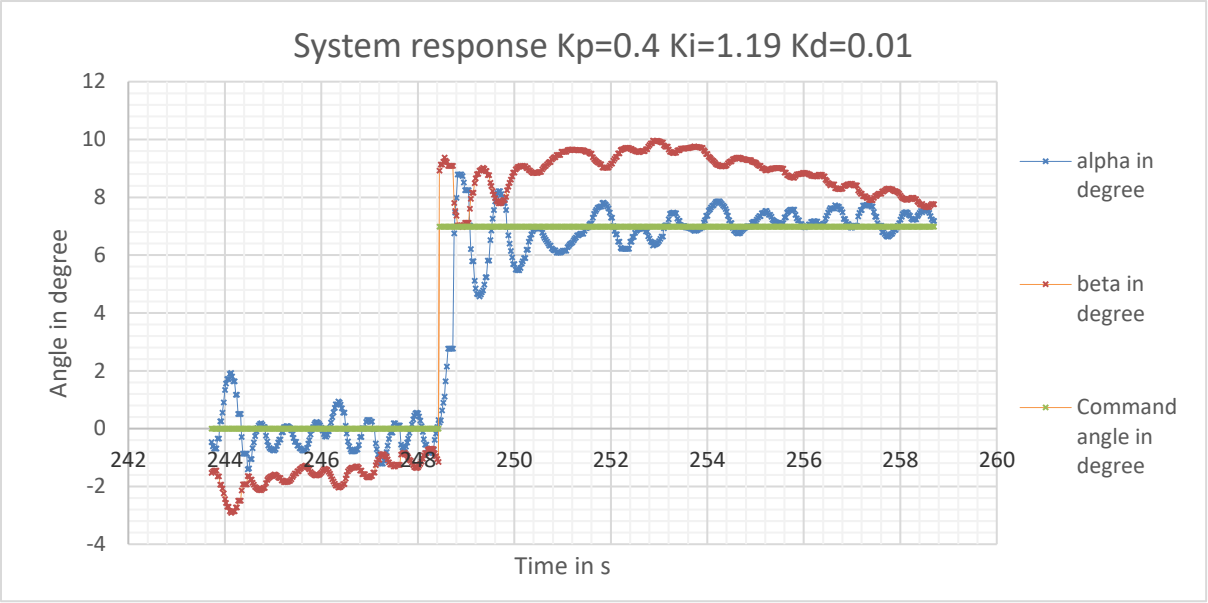


Figure 10-15 System response from Ziegler-Nichols tuning Kp=0.4 Ki=1.19 Kd=0.01

The PID controller is clearly superior to the previous settings. The overshoot is limited thanks to the derivative term and the steady-state error is null thanks to the integral term.

Table 10-8 System response performances from Ziegler-Nichols tuning

Controller parameters	P	PI	PID	unity
os	56.95	17.57	24.86	%
ess	1.10	0.00	0.00	deg
Stability	Yes	Yes	Yes	
tp	0.41	0.42	0.50	s
ts	2.89	4.96	1.43	s
tr	0.12	0.19	0.21	s

The Ziegler-Nichols tuning method offers interesting results. Indeed, the controller has a real impact on the system response. A proportional correction is not sufficient to get an acceptable value of steady-state error. Moreover, the proportional term cannot be as great as possible as expected because of the perturbations. In fact, the system is not a perfect second-order system and the perturbations are too hard to quantify.

10.5.2 P tuning for the demonstrator

According to the system Bode diagram in open-loop in 10.3.2, the phase margin $PM = +\infty$. Despite there is no real physical meaning, it is possible to assume the system is very stable and the final output value can be reached faster. A traditional phase margin is $PM = 45°$. The value of K_p can be determined to reach this phase margin. The gain margin GM is still equal to $+\infty$ because the phase never goes below $-180°$.

This value is found graphically using the Bode diagram in open-loop in Figure 10-7.

The value of $\omega_{45°}$ to get $PM = 45°$ and $\varphi = 135°$ is $12.5\, rad.s^{-1}$.

The gain at $\omega_{45°} = 12.5\, rad.s^{-1}$ is $G_{\omega_{45°}} = -6.5\, dB$ and should be equal to zero to get $PM = 45°$. In open-loop, the gain curve in a Bode diagram can be easily translated upward by multiplying the open-loop transfer function by K_p. Then, the value of K_p is:

$$K_p = 10^{\frac{-G_{\omega_{45°}}}{20}} \qquad (10\text{-}13)$$

$$K_p = 2.11$$

This value is superior to K_u from the Ziegler-Nichols tuning. The system will be unstable. An attempt with this value is done without any success.

A proportional correction is clearly insufficient to control the aerofoil. A proportional controller can only make the system faster or unstable. The steady-state error cannot be equal to zero and the overshoot is too large. Moreover, the sensitivity function shows a peak near the main perturbation frequency ω_p.

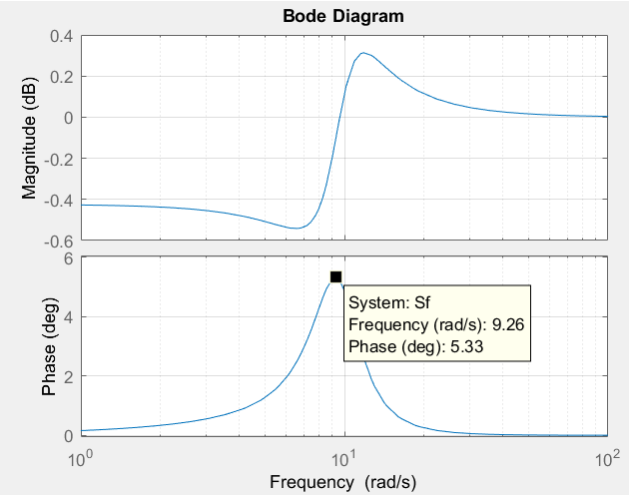


Figure 10-16 Sensitivity function Bode diagram ($K_p = 0.1$)

10.5.3 PI tuning for the demonstrator

The integral term will introduce a phase delay within the system response. This delay can increase the system instability. It is possible to notice the integral term has almost no significative impact on frequencies superior to $\frac{10*K_p}{K_i}$ or $\frac{10}{T_i}$.

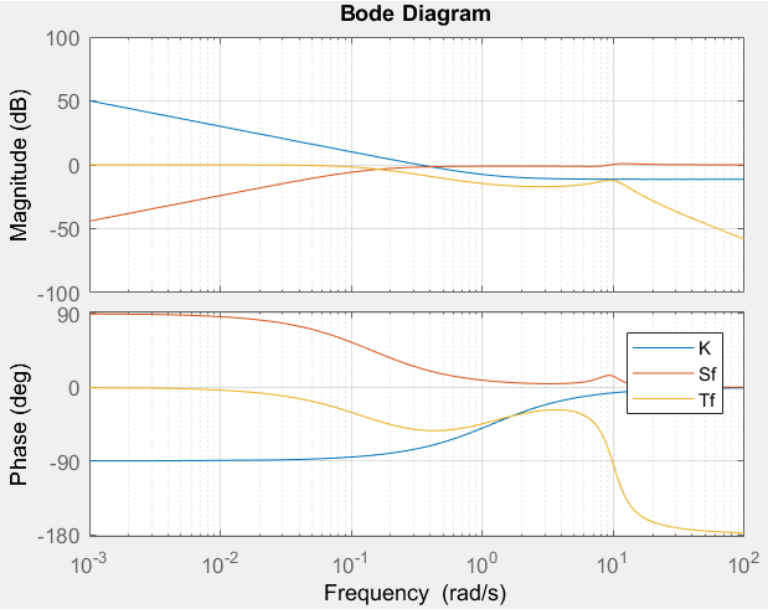


Figure 10-17 Integral term Bode diagram

A value of $K_p = 0.27$ from the Ziegler-Nichols tuning method is chosen and $K_i = 0.32$ from:

$$K_i = \frac{10K_p}{\omega_p} \qquad (10\text{-}14)$$

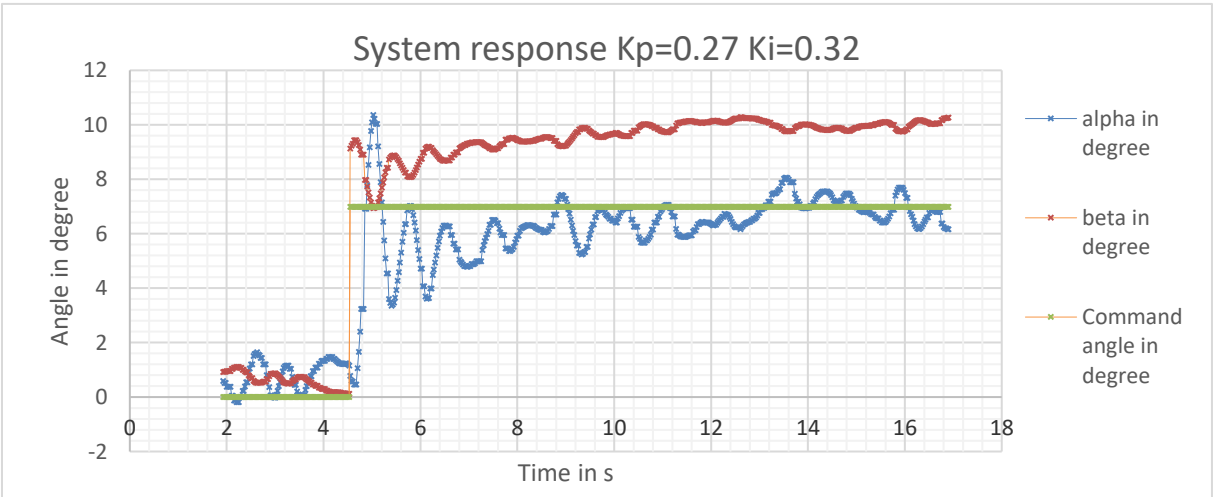


Figure 10-18 System response with a PI controller tuned using Bode diagrams

Table 10-9 System response performances PI controller

Performances		
Dimension	Value	unity
os	46.00	%
ess	0.00	deg
Stability	Yes	
tp	0.31	s
ts	4.35	s
tr	0.11	s

This correction offers a steady-state error null but a higher overshoot regarding the PI controller from the Ziegler-Nichols PI setting. The response seems quicker, but the settling time is not improved. Despite the interesting performances of the PI controller in term of steady-state response, this controller is limited to reduce the overshoot and to improve the settling time.

10.5.4 PD tuning for the demonstrator

A PD correction can increase the system stability. A value of $K_p = 0.4$ from the Ziegler-Nichols tuning method is chosen. Then a value of $K_d = 0.2$ is chosen to remove the peak of the sensitivity function:

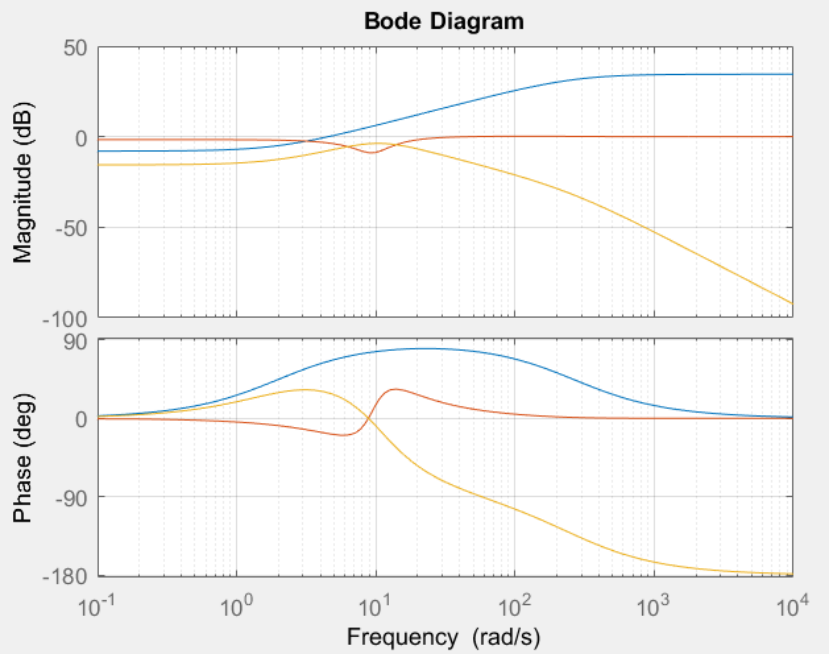


Figure 10-19 Derivative term Bode diagram

The peak of the sensitivity is clearly removed thanks to the PD controller. The system should be more stable and less sensitive to the perturbations.

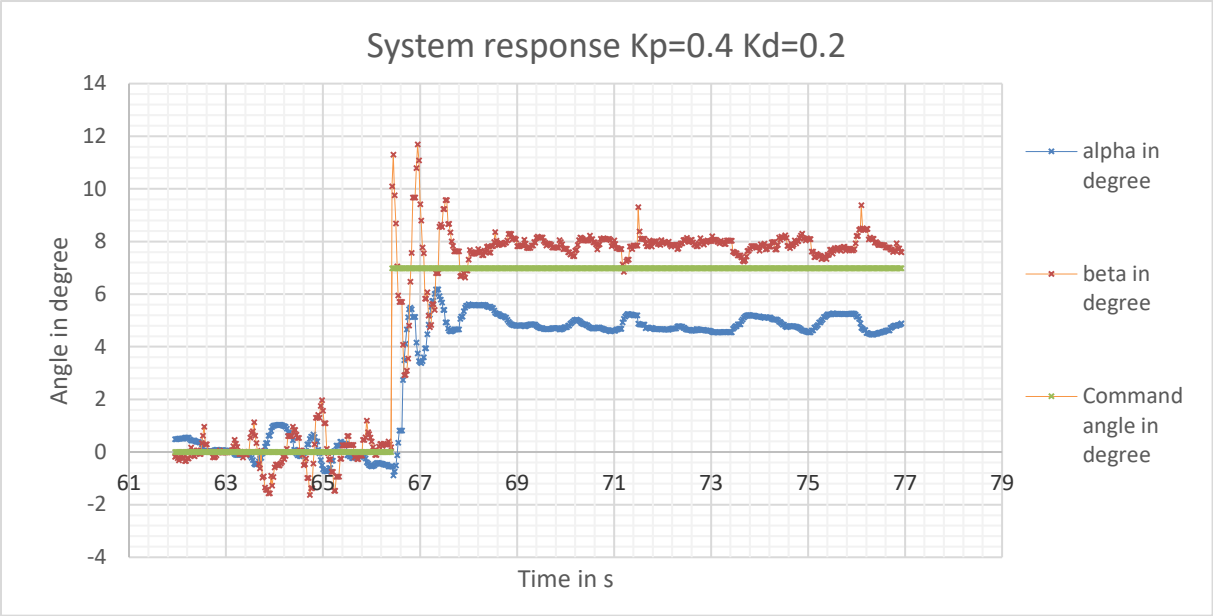


Figure 10-20 System response with a PD controller tuned using Bode diagrams

The system response shows interesting features regardless the steady-state error such as a short transient response and less oscillations due to the perturbations. It is worth to introduce the derivative term within the controller.

Table 10-10 System response performances PD controller

Performances		
Dimension	Value	unity
os	30.75	%
ess	2.35	deg
Stability	Yes	
tp	0.28	s
ts	0.95	s
tr	0.11	s

The oscillations from the perturbations are reduced by $\approx 50\%$ thanks to the derivative term because it is a lead phase term. The stability is then increased in this case.

The value of the filter is fixed during all tests. The value was determined empirically to get a smooth derivative term and the minimum of delay. It is important to notice that the filter coefficient $\frac{1}{N}$ in Matlab® is equal to $\frac{K_d}{N}$ introduced in 2.5. The value found is $N = 10.54$ which is typical for a filter coefficient value. It means in Matlab® $N = 52.68$ with $K_d = 0.2$.

10.5.5 PID tuning for the demonstrator

The aim of the following study is to get the optimal system response using the PID tuning tool of Matlab®. The controller must implement a PID control to get the advantages of each term.

The following theoretical response is obtained using the empirical model.

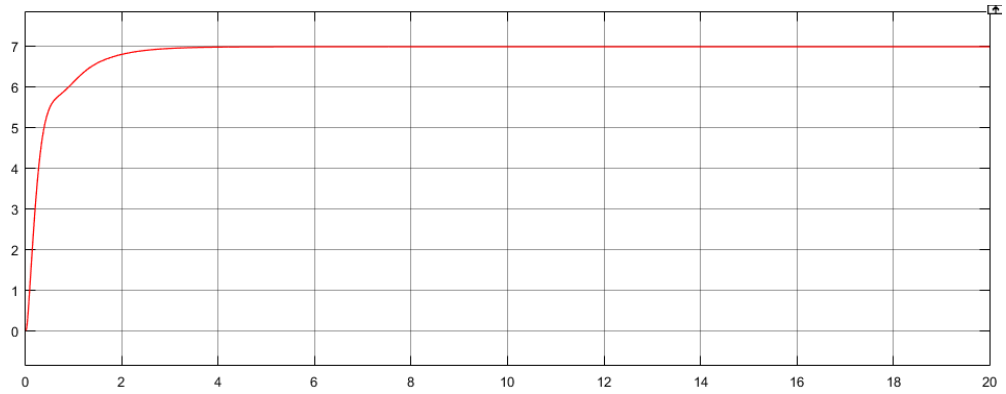


Figure 10-21 Theoretical system response Kp=0.1 Ki=2.5 Kd=0.2

The overshoot and the steady-state error should equal to zero. No oscillation.

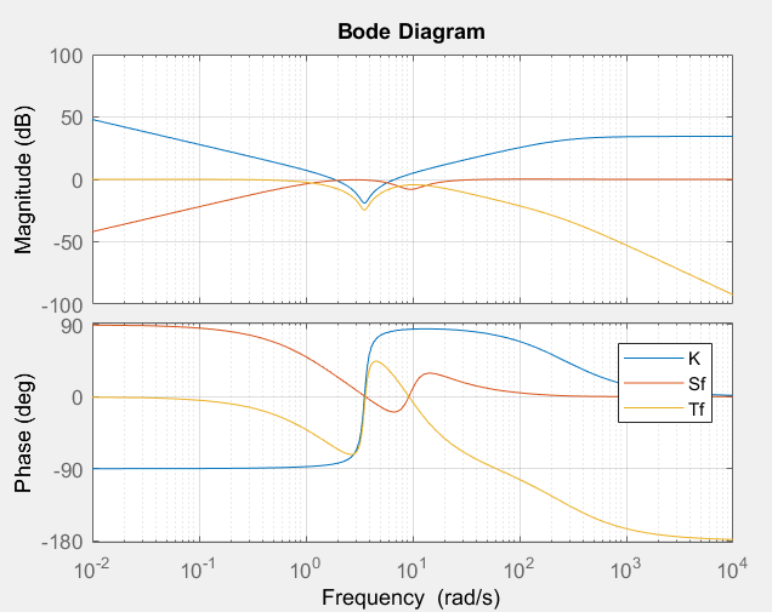


Figure 10-22 PID Bode diagram

The peak of the sensitive function is removed thanks to the derivative term according to the Bode diagram.

The real system response is given below:

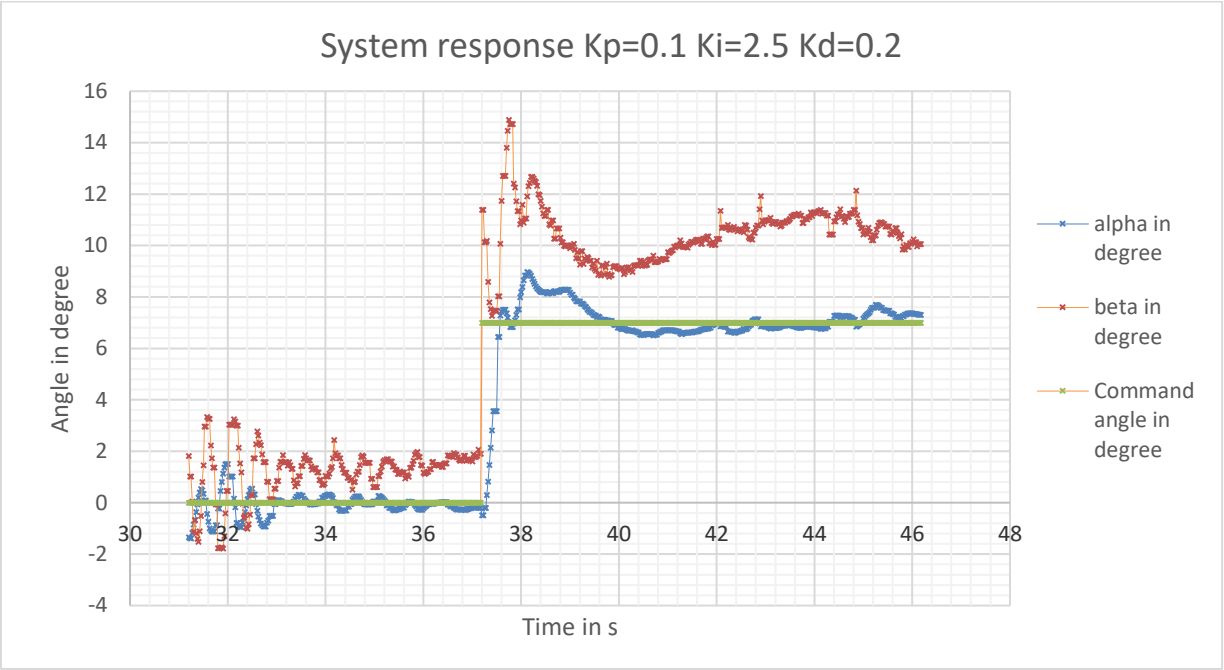


Figure 10-23 System response with a PID controller tuned using Matlab®

The real system response shows a small overshoot as well as a good rejection of the perturbation. The steady-state error is null as expected. There is almost no oscillation during the transient response.

Performances		
Dimension	Value	unity
os	18.86	%
ess	0.00	deg
Stability	Yes	
tp	1.05	s
ts	3.75	s
tr	0.22	s

A PID control is worth to be implemented on the demonstrator. In fact, the system response is significantly improved in terms of steady-state error, perturbation rejection, overshoot and settling time. The Matlab® model was very useful and saved a lot of time regarding the tuning and testing process.

This PID settings is only optimal regarding:

- The overshoot
- The oscillations
- The steady-state

10.5.6 Conclusion

This section aimed to find PID settings for the demonstrator. Several methods are used including the Ziegler-Nichols tuning method, the Bode diagram from the empirical open-loop model and the PID tuning tool from Matlab®. The PID settings focus on reference tracking keeping in mind the system shows a sinusoidal perturbation. Investigating settings for disturbance rejection may be an interesting study but the disturbance needs to be characterised before.

The PID settings are very effective regarding the system response and it is easy to see the differences between the settings. A proportional control is not sufficient to get an acceptable response and this kind of controller can make the system unstable. However, a PI controller shows good results regarding the steady-state error as well as a PD controller regarding the overshoot. A PID setting takes the advantages of both previous controllers making the system more responsive and stable.

Regarding the optimal PID setting, it is noticeable the proportional coefficient is low comparing to the other coefficient because it can make the system unstable. Contrary to the integral coefficient which is high to reach the reference quickly. A high integral coefficient makes the system slow in case of disturbance and then unstable due to the high delay induced. That is why the derivative term improves a lot the system behaviour.

Finally, the demonstrator manages to show the effectiveness of a PID control and many settings can be tested.

11 Conclusion

This document presents the design of a flight desk control demonstrator to explain the concept of feedback control and the choice of the three gains (Proportional, Integrator, Derivative) for a simple PID controller. The system consists of a demonstrator for the control of the pitch of a simple aerofoil by means of a regulated flap. The aerofoil is enclosed into a wind tunnel. The flap is controlled through a servomotor connected to a microcontroller. The user can send the command angle and the PID settings thanks to a GUI. The GUI allows the user to get the real time position of the aerofoil and the flap as well as exporting the data into Excel files.

The first step was to design the wind tunnel using engineering data. The losses are quantified, and the design is checked using CFD. The aim of the wind tunnel is to get an ideal airflow inside the structure to focus only on the system control and not on the aerodynamic phenomena. The structure is made of polycarbonate sheets and leans on a wood structure. The turbine is composed of a brushless motor and a propeller.

The second step was to create an aerodynamic model of the aerofoil. The model is computed using engineering data and checked using CFD. The aim of the model is to determine the static equilibrium between the aerofoil AoA and the flap deflection and, the dynamic behaviour of the system. The aerofoil is then 3D printed and implemented into the wind tunnel. The pivot liaison between the aerofoil axis and the wind tunnel is created to minimise the friction.

The third step was to program in C++ the microcontroller to control the flap. A PID control is implemented as well as the other necessary features such as the communication, the turbine power control, buzzer and LED indications. The microcontroller is also in charge to read the different values from the sensors. The AoA of the aerofoil is read thanks to a potentiometer and the airflow velocity is obtained with a Pitot tube. All the components are soldered on a PDB.

The next step was to program the GUI in JAVA. The GUI is user-friendly and offers useful features such as selectable pre-set PID settings, real-time aerofoil and flap position and the possibility to export the data. The connexion between the microcontroller and the computer can be established using Bluetooth or an USB cable. The data from the microcontroller are filtered to give the best experience to the user.

The final step was to assess the benchtop by comparing the theoretical model and the real system. Several differences are found. Many assumptions have been made for the sake of the project and to make the calculation feasible. In fact, data were missing and sometimes not available. The aerodynamic characteristics of the wind tunnel are not perfect especially the distance between the aerofoil and the turbine. This distance should be greater, and it should be worth to implement a screen or a honeycomb at the inlet of the wind tunnel to straighten the airflow. A duct around the propeller should be worth to be implemented to reduce perturbations inside the wind tunnel. In fact, the real system shows a high steady-state error, it is quicker than the model and a sinusoidal perturbation is present. Despite these imperfections, the demonstrator is fully functional and the PID's terms influence is clearly visible.

This document presents the characteristics of the system working in open-loop. Several PID settings are investigated and presented in this document. The attempts focus on reference tracking and tried to minimise the sinusoidal perturbation. The advantages and the limit of different PID settings for the demonstrator are discussed. A proportional controller is not satisfying but a PI controller shows acceptable results. The derivative term of a full PID setting is worth to be implemented and improves significantly the system response.

To go further in the study some points must be explored. The theoretical model and the wind tunnel characteristics can be improved. PID settings can be investigated to improve disturbances rejection. The Pitot tube location inside the wind tunnel can be improved using CFD. A more advanced study of the PID tuning process can be achieved to better control the system. The code inside the microcontroller and the GUI could offer more features such as the possibility of tuning the derivative filter coefficient or modifying the sampling time.

Moreover, with the recent growth of the drone industry the demonstrator can be used to test other type of correction such as RST corrections or PFC (Predictive Functional Control). The PID control can also be enriched by feedforward or setpoint weighting.

To conclude, the project required knowledge from many fields such as mechanical engineering, electronics, programming and aerodynamics. It was a great and valuable experience.

REFERENCES

[1] Antonio Visioli (2006) *Practical PID Control*, *Springer*. doi: 10.1007/978-1-4471-4399-4.

[2] Raut, K. H. and Vaishnav, S. R. (2011) 'A Study on Performance of Different PID Tuning Techniques', *IEEE conference*, pp. 3–6.

[3] Elliott, R. (2002) *Beginners' Guide to Potentiometers*. Available at: http://sound.whsites.net/pots.htm#markings (Accessed: 19 July 2018).

[4] Store, A. (2014) *Arduino UNO Rev3*, *online*. Available at: https://store.arduino.cc/usa/arduino-uno-rev3%0Ahttps://store.arduino.cc/usa/arduino-uno-rev3%0Ahttps://store.arduino.cc/usa/arduino-uno-rev3%0Ahttps://store.arduino.cc/arduino-uno-rev3%5Cnhttp://store.arduino.cc/product/A000066.

[5] Spectrol, V. (2013) 'Model 157 Vishay Spectrol (22 . 2 mm) Precision Industrial Potentiometer Bushing and Servo Mount Versions , Conductive Plastic ELECTRICAL SPECIFICATIONS Model 157 Vishay Spectrol IMPERFECT THREAD AREA SCHEMATIC', pp. 16–18.

[6] Honeywell (2011) 'Hall Effect Sensing and Application', *Sensing and Control*, p. 126. doi: 005715-2-EN GLO 1198.

[7] Series, S. S. and Miniature, M. R. L. (no date g) 'Solid State Sensors Miniature Ratiometric Linear SS490 Series Solid State Sensors Miniature Ratiometric Linear', *Control*, pp. 20–23.

[8] Further, G. (2015) 'Accelerometer Basics', *SparkFun Electronics*, p. 5377. Available at: https://learn.sparkfun.com/tutorials/accelerometer-basics.

[9] Kmiecik, M. and Sibilski, K. (2013) '3D Attitude Estimation in Indoor Environments with a Complementary Filter for the Special Orthogonal Group SO (3)', *Solid State Phenomena*, (January), pp. 1–11. doi: 10.4028/www.scientific.net/SSP.198.153.

[10] C.Ifantithis (1993) *Design of demonstrator equipment for the control of an inverted pendulum*. Cranfield university.

[11] Herrmann, G. (2012) 'System Specifications 2 Dof Helicopter & 3 Dof Helicopter', pp. 1–2.

[12] Pereira, J. D. (2011) *Wind Tunnels - Aerodynamics, Models and Experiments*, *Wind Tunnels: Aerodybamics, Models and Experiments*.

[13] *Singular losses K calculation* (2014). Available at: http://www.mecaflux.com/turbines.htm

(Accessed: 11 December 2014).

[14] Ansys (2009) *Ansys fluent 12.0 Theory Guide*, *ANSYS Fluent*. doi: 10.1016/0140-3664(87)90311-2.

[15] Yousefi, K. and Razeghi, A. (2018) 'Determination of the Critical Reynolds Number for Flow over Symmetric NACA Airfoils', in *2018 AIAA Aerospace Sciences Meeting*. doi: 10.2514/6.2018-0818.

[16] Anon (1972) 'AERODYNAMIC CHARACTERISTICS OF AEROFOILS IN COMPRESSIBLE INVISCID AIRFLOW AT SUBCRITICAL MACH NUMBERS.', *Engineering Sciences Data Unit, Data Items*.

[17] The Royal Aeronautical Society (1997) 'ESDU 97020 - Slope of aerofoil lift curve for subsonic two-dimensional flow', (September).

[18] NASA (2018) *NACA 0012 AIRFOILS*.

[19] Esdu (2003) 'Rate of change of lift coefficient with control deflection in incompressible two-dimensional flow', *Aero C.01.01.03*, 1(May 1956), pp. 1–5. Available at: www.esdu.com.

[20] Flow, T. (no date t) 'ESDU Controls 08.01.01', pp. 1–3.

[21] Data, S. T. (2001) 'Integrated Silicon Pressure Sensor On-Chip Signal Conditioned , Temperature Compensated and Calibrated SERIES', *Time*, (2), pp. 1–9.

[22] Range, W., Voltages, S., Supply, S., Supplies, D., Drain, L. S. and Voltage, S. (1998) 'Lm158, lm158a, lm258, lm258a lm358, lm358a, lm358y, lm2904, lm2904q dual operational amplifiers', *Office*, (July), pp. 1–10.

[23] J.F Whidborne (2017) *Control Systems*. Cranfield.

APPENDICES

Appendix A T-MOTOR® MT2814 Datasheet

Technical Datas	
KV	770
Configu-ration	12N14P
Stator Diameter	28mm
STator Length	14mm
Shaft Diameter	4mm
Motor Dimension(Dia. * Len)	Φ35×36mm
Weight(g)	125G
Idle current(10)@10v(A)	1.0
No.of Cells(Lipo)	3-4S
Max Continuous current(A)180S	30A
Max Continuous Power(W)180S	430W
Max. efficiency current	6-16A>76%
internal resistance	100mΩ

Volts (V)	Prop	Throttle	Amps (A)	Watts (W)	Thrust (G)	Thrust (OZ)	RPM	Efficiency (G/W)	Efficiency (OZ/W)	Remark
11.1	APC10*3.8	50%	4.1	45.51	350	12.35	4337	7.69	0.27	
		65%	5.6	62.16	500	17.64	5190	8.04	0.28	
		75%	8.6	95.46	690	24.34	6120	7.23	0.25	
		85%	11	122.1	820	28.92	6700	6.72	0.24	
		100%	12.5	138.75	920	32.45	6970	6.63	0.23	
	APC11*3.8	50%	3.6	39.96	340	11.99	4037	8.51	0.30	
		65%	4.9	54.39	430	15.17	4825	7.91	0.28	
		75%	6.2	68.82	550	19.40	5520	7.99	0.28	
		85%	8.8	97.68	710	25.04	5830	7.27	0.26	
		100%	13.2	146.52	940	33.16	6265	6.42	0.23	
	APC12*3.8	50%	5.2	57.72	470	16.58	4250	8.14	0.29	
		65%	9.4	104.34	750	26.46	5091	7.19	0.25	
		75%	13.5	149.85	960	33.86	6050	6.41	0.23	
		85%	16.6	184.26	1140	40.21	6560	6.19	0.22	
		100%	19	210.9	1200	42.33	7215	5.69	0.20	
	APC13*4	50%	3.9	43.29	380	13.40	3610	8.78	0.31	
		65%	5.6	62.16	570	20.11	4100	9.17	0.32	
		75%	10.2	113.22	860	30.34	4705	7.60	0.27	
		85%	12.6	139.86	1000	35.27	5000	7.15	0.25	
		100%	14.1	156.51	1100	38.80	5314	7.03	0.25	
14.8	APC10*3.8	50%	4.7	69.56	550	19.40	5375	7.91	0.28	
		65%	7.7	113.96	770	27.16	6300	6.76	0.24	
		75%	14.9	220.52	1210	42.68	7130	5.49	0.19	
		85%	17.8	263.44	1370	48.32	7800	5.20	0.18	
		100%	20	296	1420	50.09	8520	4.80	0.17	
	APC11*3.8	50%	5.5	81.4	580	20.46	5065	7.13	0.25	
		65%	7.7	113.96	750	26.46	5800	6.58	0.23	
		75%	10.4	153.92	940	33.16	6490	6.11	0.22	
		85%	14.5	214.6	1180	41.62	7070	5.50	0.19	
		100%	20.7	306.36	1480	52.20	7360	4.83	0.17	
	APC12*3.8	50%	8.2	121.36	810	28.57	5530	6.67	0.24	
		65%	11.8	174.64	1060	37.39	6610	6.07	0.21	
		75%	17.4	257.52	1380	48.68	7560	5.36	0.19	
		85%	23.6	349.28	1620	57.14	8240	4.64	0.16	
		100%	29.2	432.16	1870	65.96	8930	4.33	0.15	
	APC13*4	50%	6.3	93.24	680	23.99	4380	7.29	0.26	
		65%	9.7	143.56	980	34.57	4850	6.83	0.24	
		75%	14.5	214.6	1330	46.91	5610	6.20	0.22	
		85%	18.5	273.8	1610	56.79	5611	5.88	0.21	
		100%	21.3	315.24	1780	62.79	6360	5.65	0.20	

Appendix B Velocity iterative calculation

```
Public Sub UpdateLosses()
    Application.ScreenUpdating = False
    Dim lossesComputed As Double
    Dim lossesForCalculation As Double
    Dim i As Integer
    i = 0
    With Sheets("Fan")
        lossesComputed = .Range(CelluleTarget("losses computed", "Fan")).Offset(0, 1).Value()
        lossesForCalculation = .Range(CelluleTarget("losses for calculation", "Fan")).Offset(0, 1).Value()
    End With
    While Abs(lossesForCalculation - lossesComputed) >= 0.1
        If lossesComputed < lossesForCalculation Then
            lossesForCalculation = lossesForCalculation - 0.05
            Else
            lossesForCalculation = lossesForCalculation + 0.05
        End If
    Sheets("Fan").Range(CelluleTarget("losses for calculation", "Fan")).Offset(0, 1).FormulaR1C1 = lossesForCalculation
    Sheets("Fan").Calculate
    Sheets("Losses").Calculate
    Sheets("Fan").Calculate
    Sheets("Fan").lossesComputed = .Range(CelluleTarget("losses computed", "Fan")).Offset(0, 1).Value()
    i = i + 1
    If i >= 5000 Then
        MsgBox "Too many iteration", vbOKOnly + vbInformation, "Erreur"
        Exit Sub 'security
    End If
    Wend
    Application.ScreenUpdating = True
End Sub
```

Appendix C CFD Results

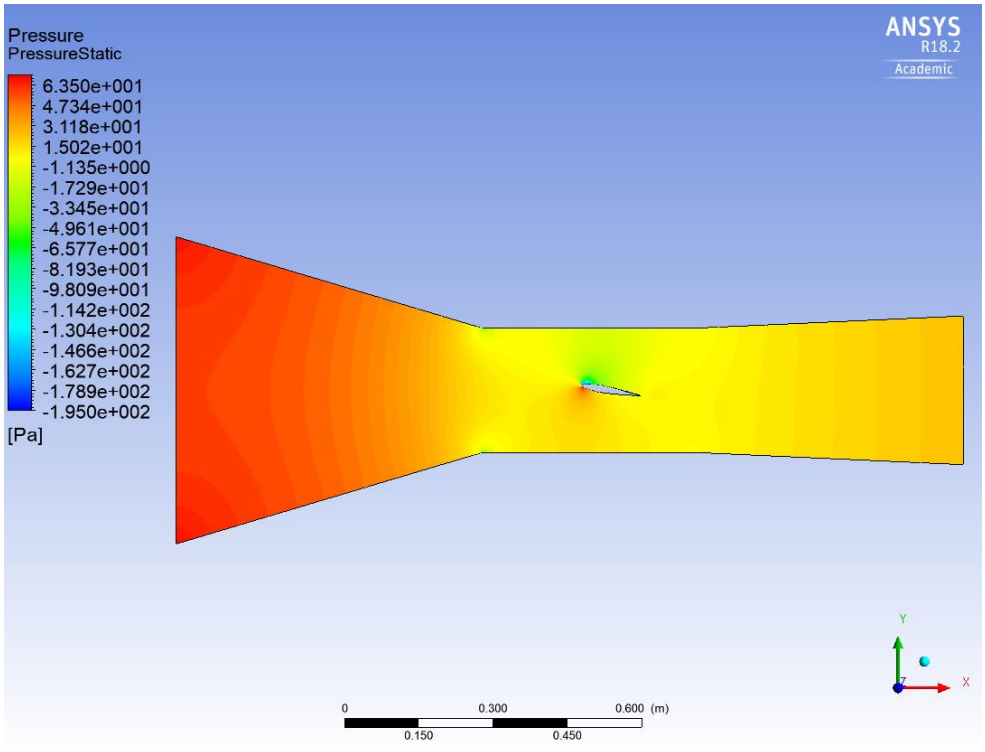


Figure C-1 Wind tunnel pressure distribution

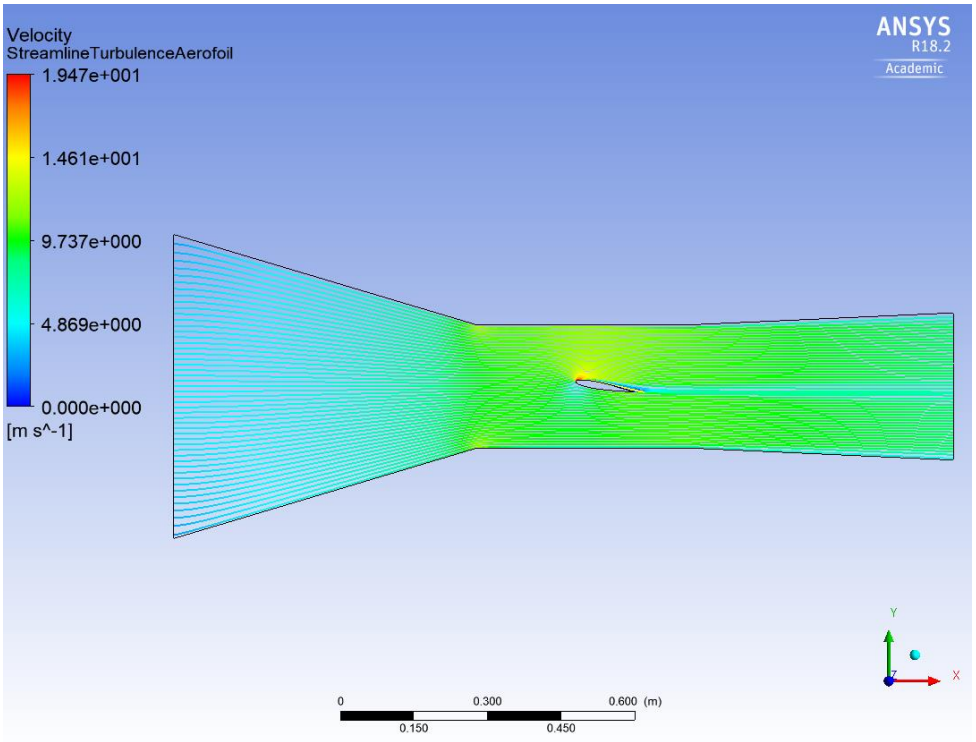


Figure C-2 Wind tunnel streamline distribution

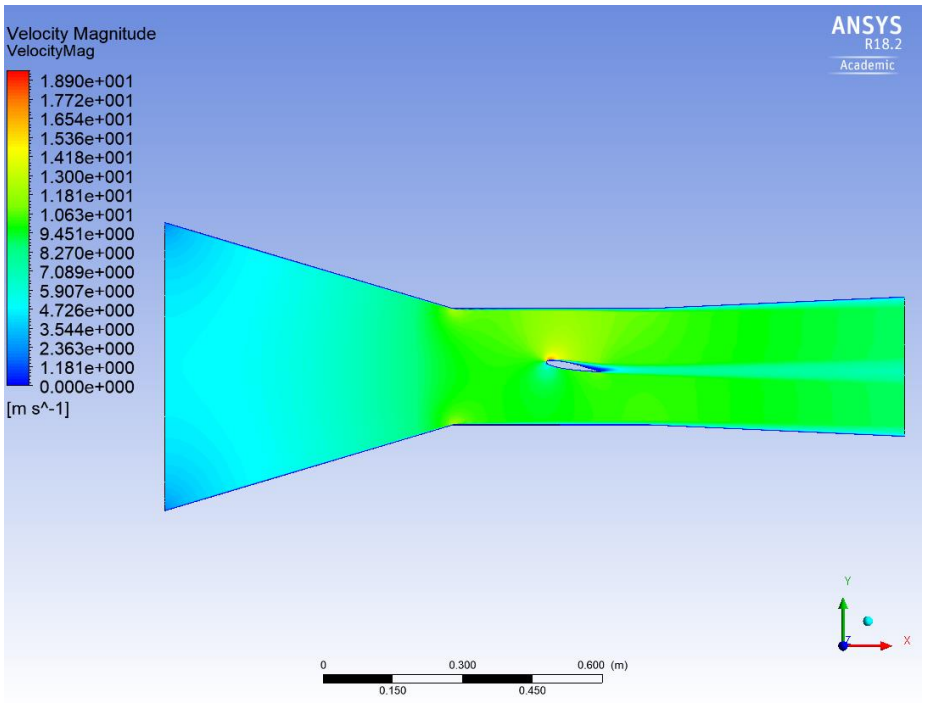


Figure C-3 Wind tunnel velocity magnitude distribution

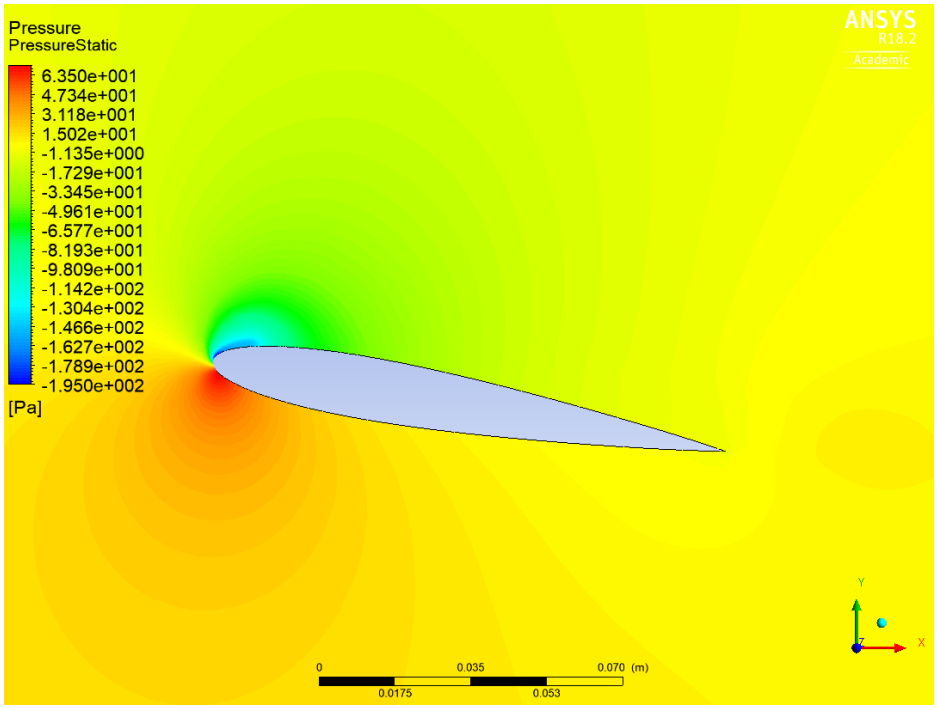


Figure C-4 Pressure distribution around the aerofoil

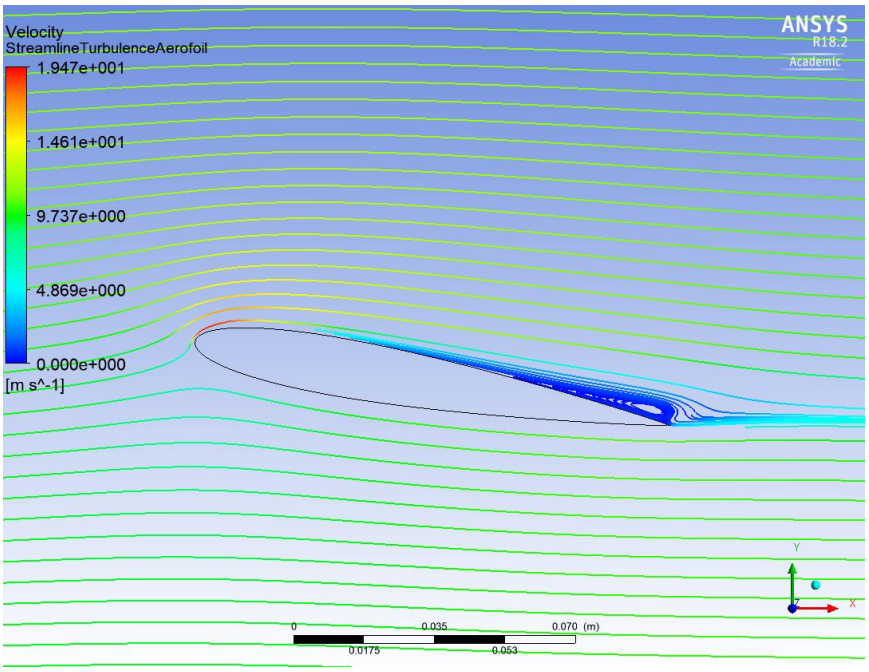


Figure C-5 Streamline distribution around the aerofoil

Appendix D Aerodynamic Model

D.1 Theoretical results

Table D-1 Theoretical results from the aerodynamic analysis

Static test Model				
AoA α deg	β rad	β deg	Cm	Cl
-10	-0.18	-10.12	-0.093	-0.535
-9	-0.16	-9.11	-0.084	-0.481
-8	-0.14	-8.10	-0.074	-0.428
-7	-0.12	-7.09	-0.065	-0.374
-6	-0.11	-6.07	-0.056	-0.321
-5	-0.09	-5.06	-0.046	-0.267
-4	-0.07	-4.05	-0.037	-0.214
-3	-0.05	-3.04	-0.028	-0.160
-2	-0.04	-2.02	-0.019	-0.107
-1	-0.02	-1.01	-0.009	-0.053
0	0.00	0.00	0.000	0.000
1	0.02	1.01	0.009	0.053
2	0.04	2.02	0.019	0.107
3	0.05	3.04	0.028	0.160
4	0.07	4.05	0.037	0.214
5	0.09	5.06	0.046	0.267
6	0.11	6.07	0.056	0.321
7	0.12	7.09	0.065	0.374
8	0.14	8.10	0.074	0.428
9	0.16	9.11	0.084	0.481
10	0.18	10.12	0.093	0.535

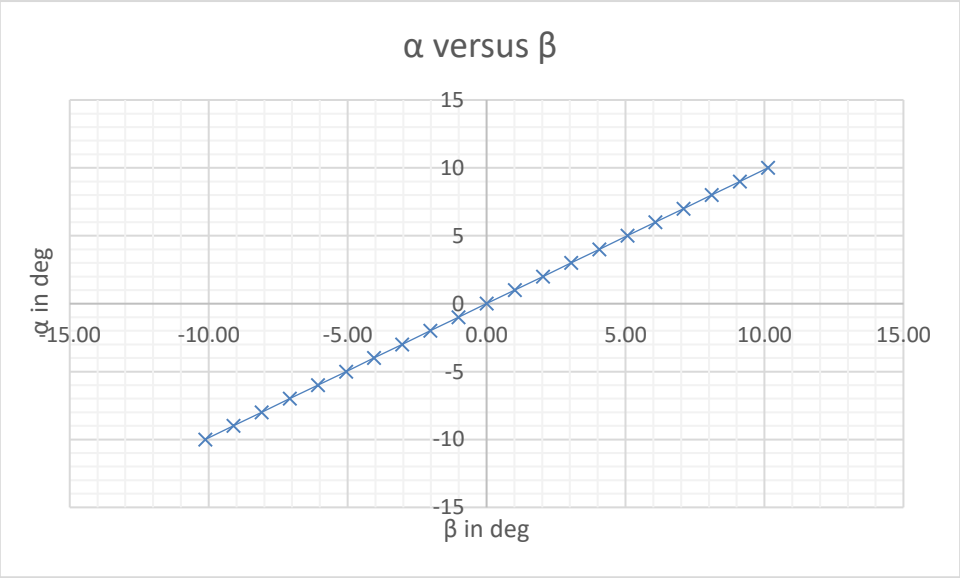


Figure D-1 Evolution of the AoA α versus the flap angle β

D.2 Results from JAVAFOIL®

Table D-2 2D aerofoil model validation

Javafoil			
Cmj	error %	Clj	error %
-0.085	-9.3	-0.470	-13.8
-0.080	-4.6	-0.439	-9.7
-0.072	-3.3	-0.395	-8.3
-0.064	-1.7	-0.349	-7.3
-0.057	-2.2	-0.313	-2.5
-0.048	-3.2	-0.262	-2.1
-0.038	-2.2	-0.210	-1.9
-0.029	-3.9	-0.158	-1.6
-0.019	-2.2	-0.105	-1.9
-0.010	-7.1	-0.053	-0.9
0.000	0.0	0.000	0.0
0.010	7.1	0.053	0.9
0.019	2.2	0.105	1.9
0.029	3.9	0.158	1.6
0.038	2.2	0.210	1.9
0.047	1.1	0.262	2.1
0.056	0.4	0.313	2.5
0.062	4.9	0.349	7.3
0.070	6.2	0.395	8.3
0.078	7.2	0.439	9.7
0.083	12.0	0.470	13.8

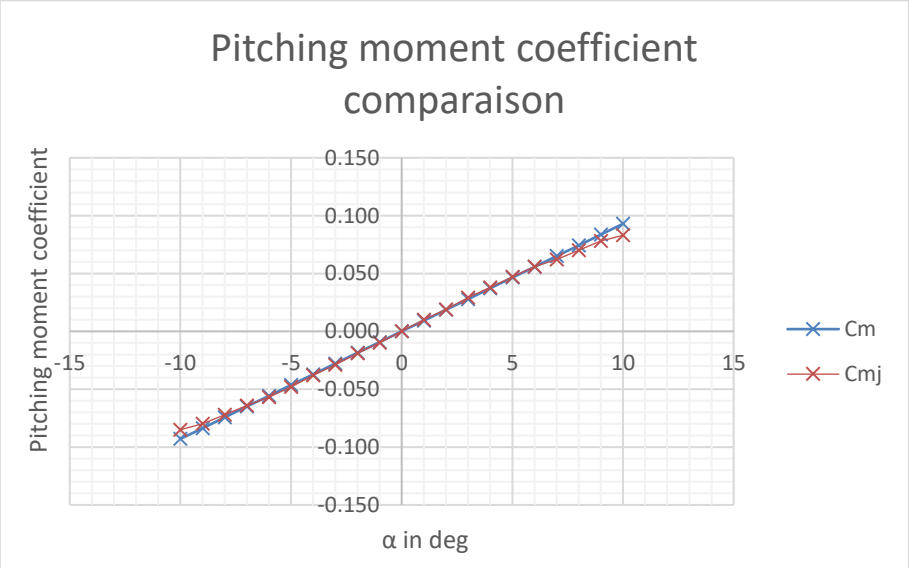


Figure D-2 Pitching moment coefficient model validation

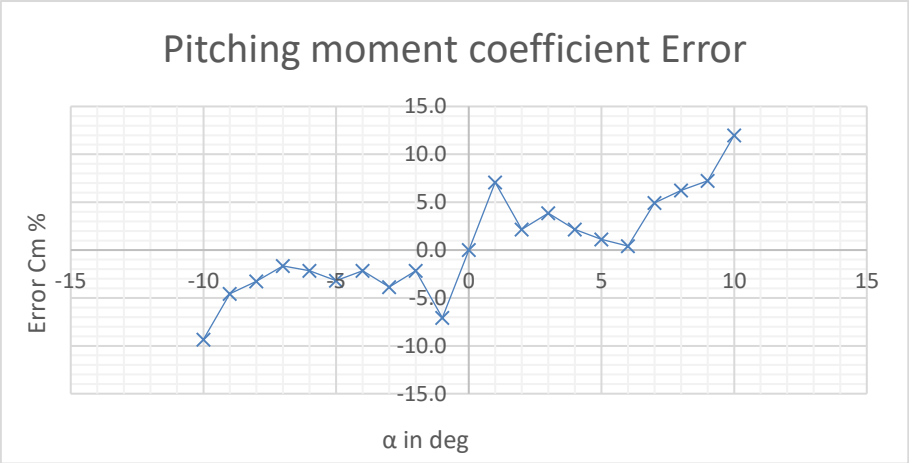


Figure D-3 Pitching moment coefficient model error

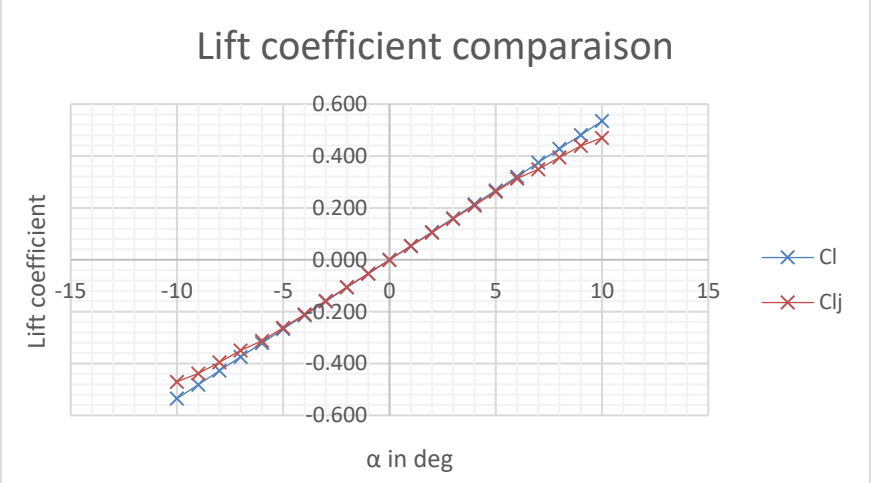


Figure D-4 Lift coefficient model validation

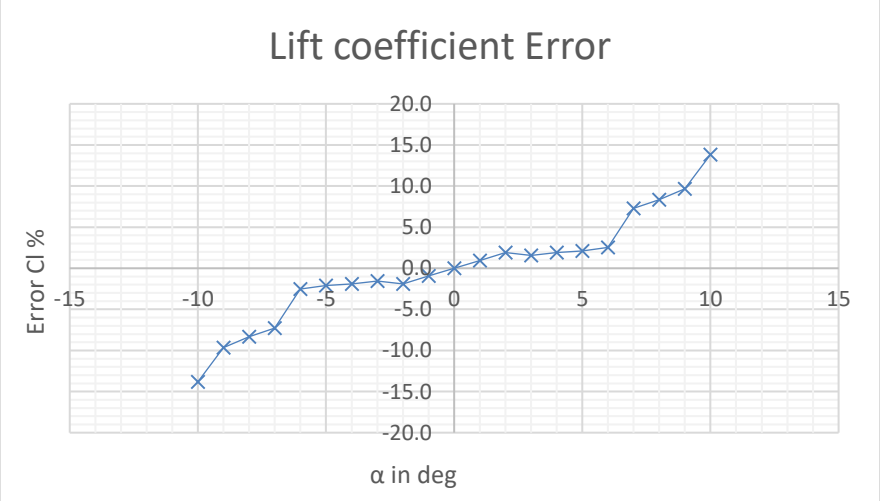
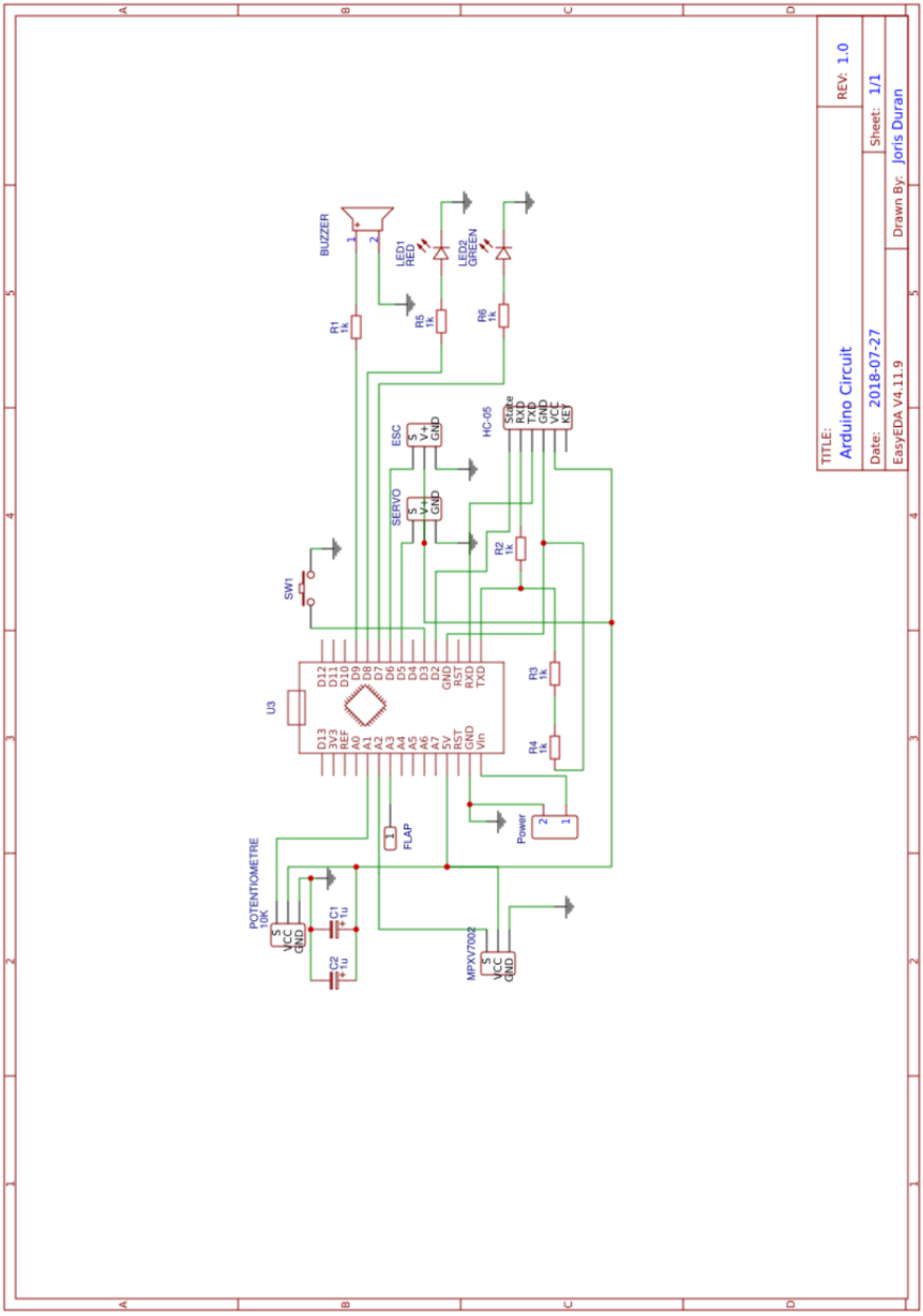


Figure D-5 Lift coefficient model error

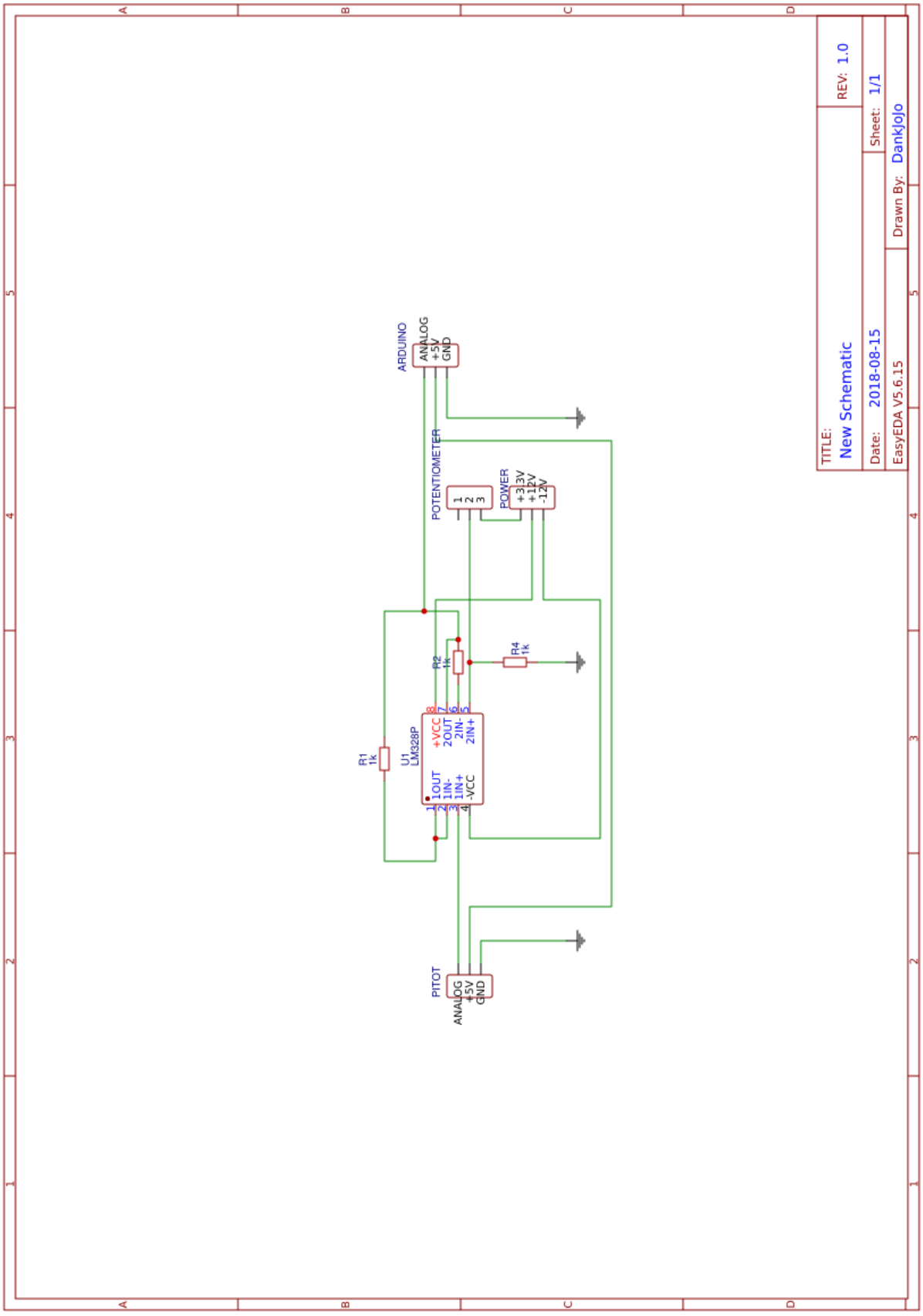
Appendix E JAVAFOIL® scrip

```
// Enregistré le 24/04/18 à 14:43 par PC-JORIS

Options.Country(1);

Modify.Select(1);
Options.MachNumber(0.0295);
Options.StallModel(0);
Options.TransitionModel(1);
Options.GroundEffect(0);
Options.HeightOverSpan(0.5);
Options.AspectRatio(0);
Options.SweepAngle(0.0);
Modify.SetPivot(0,0);

var g = 1.01238;

var a = -10;
Geometry.CreateAirfoil(0,59,12,30,0,40,0,0,1);
Modify.Flap(20,-a*g);
Polar.Analyze(80000,80000,0,a,a,0,30,100,2,0);
Polar.Save("C:\\Users\\PC-JORIS\\Desktop\\Test_beta_" + a);

a = a+1;
Geometry.CreateAirfoil(0,59,12,30,0,40,0,0,1)
Modify.Flap(20,-a*g);
Polar.Analyze(80000,80000,0,a,a,0,30,100,2,0);
Polar.Save("C:\\Users\\PC-JORIS\\Desktop\\Test_beta_" + a);
                                                    .
                                                    .
                                                    .
```

Appendix F Electronic circuit main board



Appendix G Pitot tube voltage adaptor circuit



Appendix H Matlab® variables

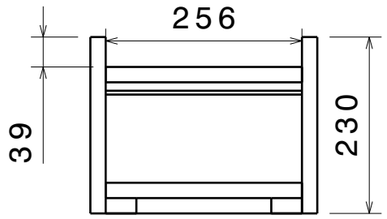
```
%      Flight Desk Demonstrator PID      %
%                  V1                    %
% ---------------------------------------%
%              Joris Duran               %
% ---------------------------------------%

% Variable Declaration %
% Alpha command instead of knob
a=12; % in deg
% PID coefficients
Kp=0.0;
Ki=0.0;
Kd=0.0;
Kf=52.68;
% Servo time constant
T= 8*0.04/60; % in s for 8deg as command angle
% System Inertia and damping
q=68.4; % in Kg/m.s-2
S=28571 % in mm2
I=0.790; % in Kg.mm2/q/S I=0.001544 % in Kg.m2
v=0.5; % in N.mm/deg/s/q/S
C=0; % in N.mm/deg/q/S %
% System properties
l= 20.68; % in mm
a1= 0.114; % in deg-1 aerofoil's curve slope
a2= 3.44*pi/180; % in deg-1 rate of change of Cl due to the trailing edge device deflection
m0= 0.526*pi/180; % in deg-1 rate of change of Cm due to the trailing edge device deflection
c=119.05; % in mm aerofoil's chord length
% Gain Calculation %
A=l*a1;
B=(m0*c+l*a2);
```

Appendix I Engineering Drawings

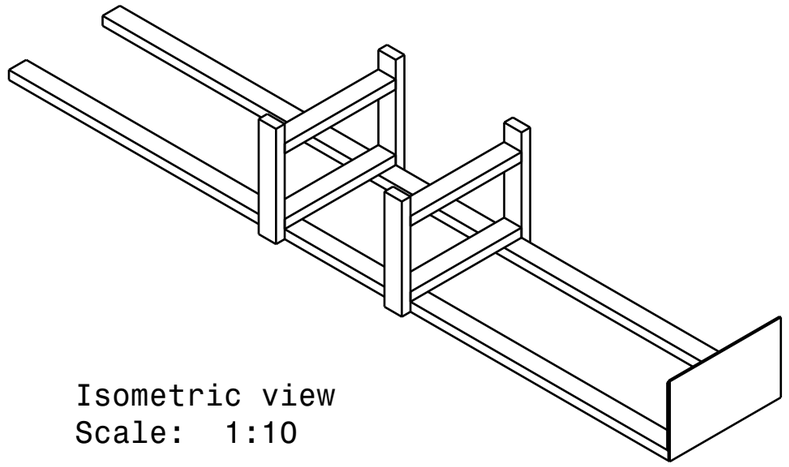
List of Drawings	Name
D1	Wind Tunnel Support
D2	Aerofoil Assembly
D3	Wind Tunnel Assembly
D4	Flight Desk Control Demonstrator
D5	Aerofoil_Part_1
D6	Flap_Part_1
D7	Top
D8	Nozzle Attachment
D9	Chamber Panel
D10	Nozzle Panel

Drg No.1 | Sht No.1 | Rev. 1

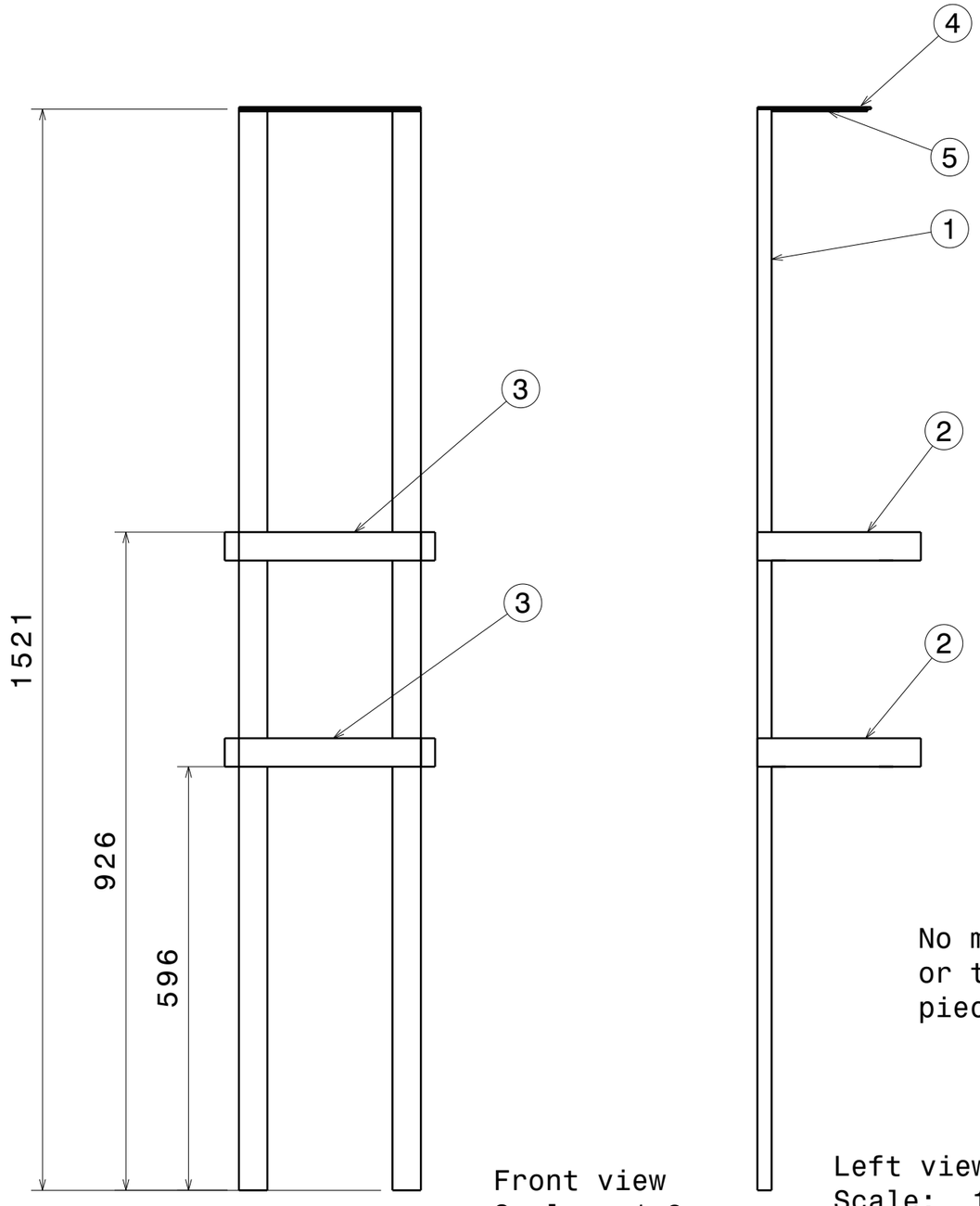
Zone	Rev.	Descripton of change



Bottom view
Scale: 1:8



Isometric view
Scale: 1:10



Front view
Scale: 1:8

Left view
Scale: 1:8

No matter the thickness
or the width of the
pieces of wood

Recapitulation of: Support
Different parts: 5
Total parts: 12

Quantity	Number	Part Number
1	1	Long_Beam_1
1	1	Long_Beam_2
1	2	Small_Beam_1
1	2	Small_Beam_2
1	2	Small_Beam_3
1	3	Beam_Support_1
1	2	Small_Beam_4
1	3	Beam_Support_2
1	3	Beam_Support_3
1	3	Beam_Support_4
1	4	Wood_Plate_1
1	5	Wood_Plate_2

Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 1deg Decimals: 1mm

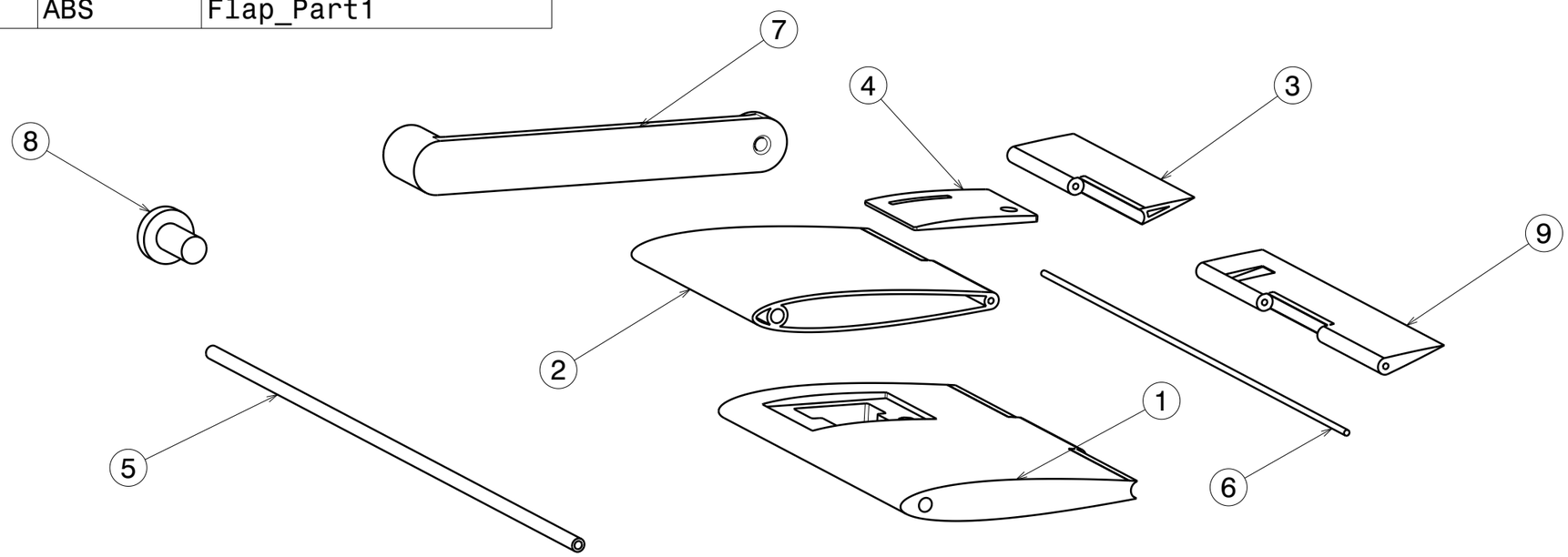
Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	J.DURAN	D1
	Date drawn	
A3	16/06/2018	Drawing Description
Scale	CATIA code	Wind Tunnel Support
1:8		

Drg No.	Sht No.	Rev.

Zone Rev. Descripton of change

Recapitulation of: Aerofoil_Assembly_V1
Different parts: 9
Total parts: 9

Quantity	Number	Mat	Part Number
1	2	ABS	Aerofoil_Part2
1	3	ABS	Flap_Part2
1	4	ABS	Top
1	5	Aluminium	AerofoilShaft
1	6	Aluminium	AileronShaft
1	7	Wood	Mass
1	8	Steel	HeavyScrew
1	9	ABS	Aerofoil_Part1
1	1	ABS	Flap_Part1



Isometric view
Scale: 1:2

Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 0.1deg Decimals: 0.1mm

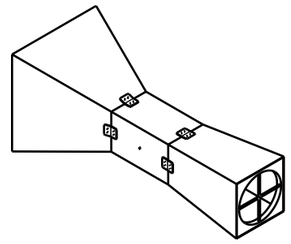
Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	JORIS RENE DURAN	D2
	Date drawn	
A3	18/08/2018	Drawing Description
Scale	CATIA code	Aerofoil Assembly
1:2		

Drg No.	Sht No.	Rev.

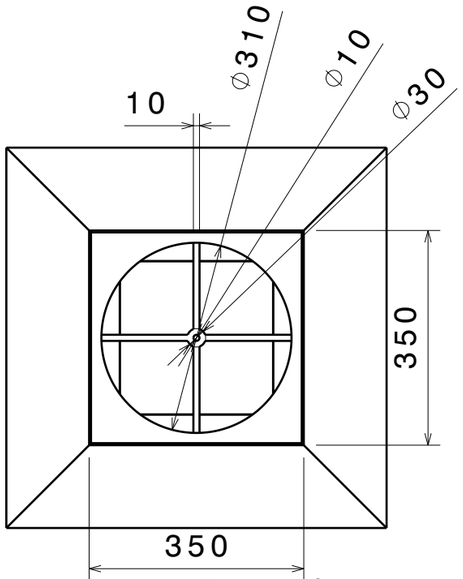
Recapitulation of:
Wind_Tunnel_Assembly_V1
Different parts: 5
Total parts: 9

Quantity	Number	Mat	Part Number
1	1	PMMA	Wind Tunnel
2	2	ABS	NozzleAttacment
2	3	ABS	Symmetry of NozzleAttacment
2	4	ABS	DiffuserAttacment
2	5	ABS	Symmetry of DiffuserAttacment

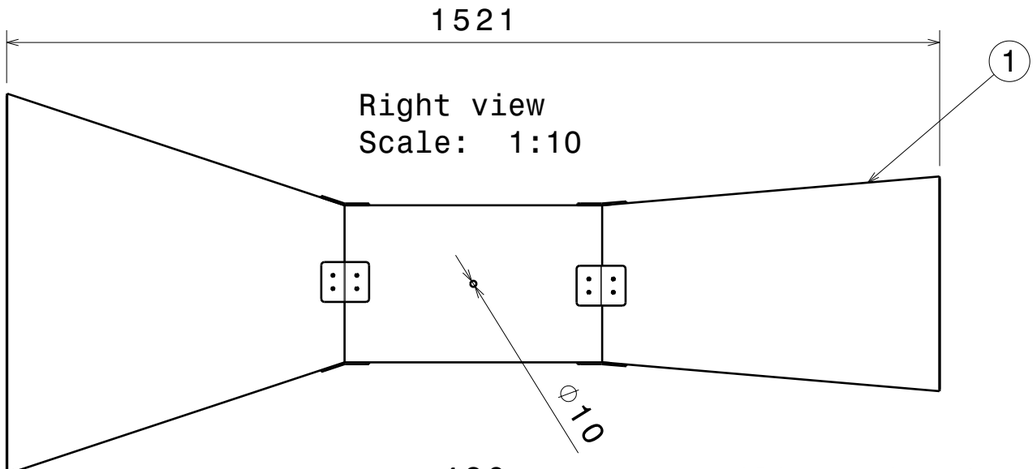
Zone	Rev.	Descripton of change



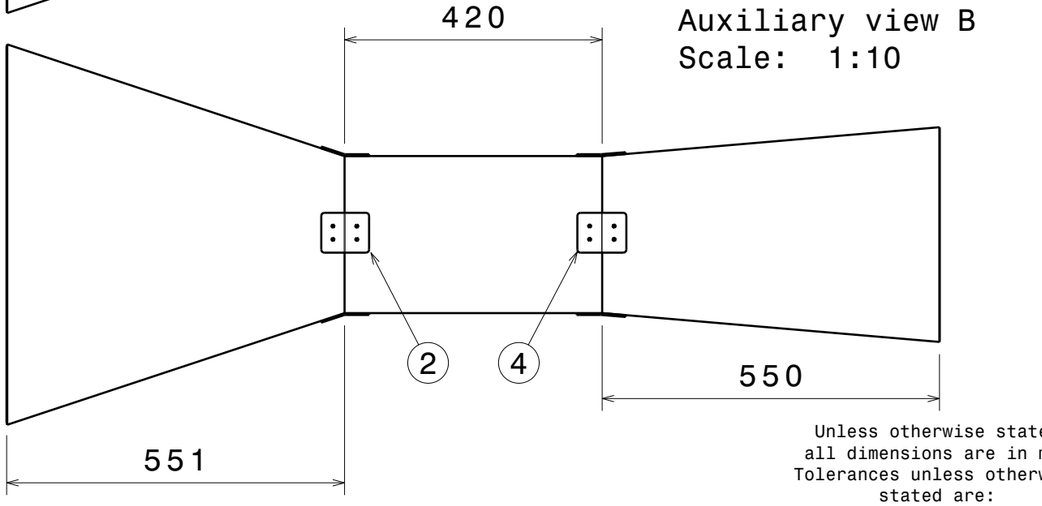
Isometric view
Scale: 1:32



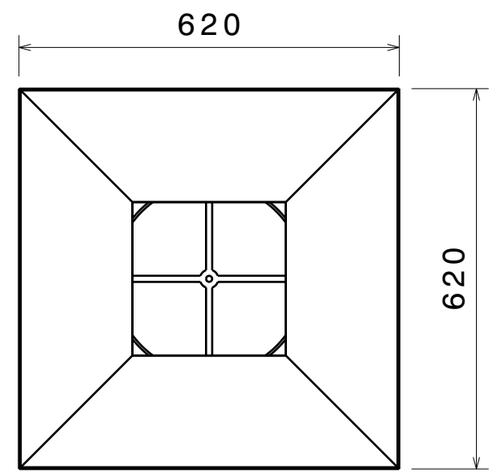
Rear view
Scale: 1:10



Right view
Scale: 1:10



Auxiliary view B
Scale: 1:10



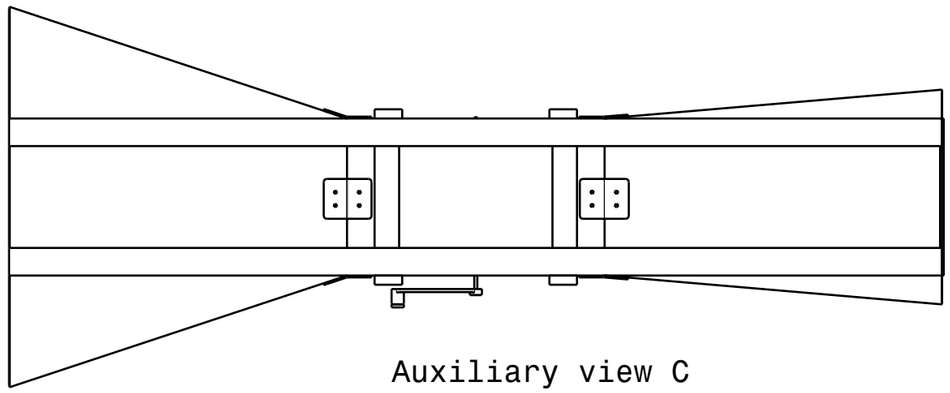
Auxiliary view A
Scale: 1:10

Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 0.1deg Decimals: 0.1mm

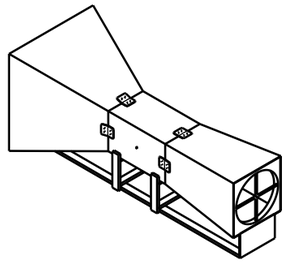
Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	JORIS RENE DURAN	D3
	Date drawn	
A3	18/08/2018	Drawing Description
Scale	CATIA code	Wind Tunnel Assembly
1:8		

Drg No.1	Sht No.1	Rev. 1

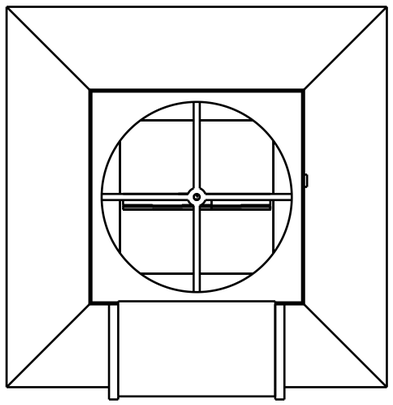
Zone	Rev.	Descripton of change



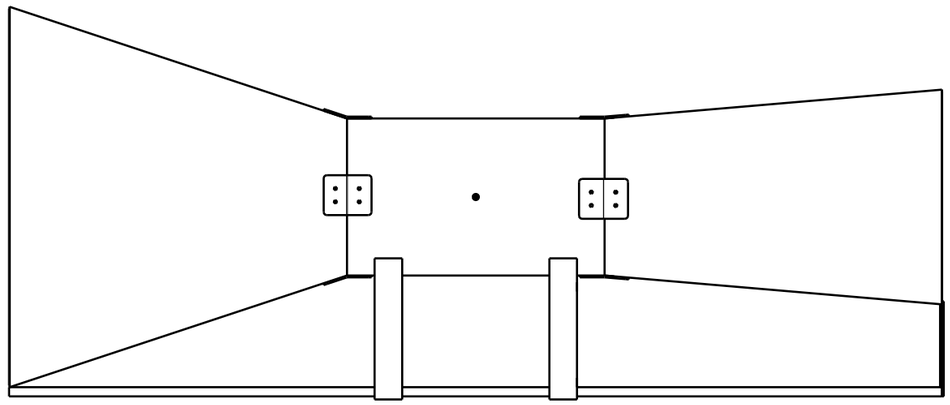
Auxiliary view C
Scale: 1:10



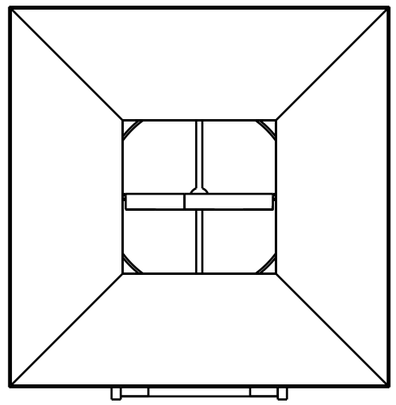
Isometric view
Scale: 1:32



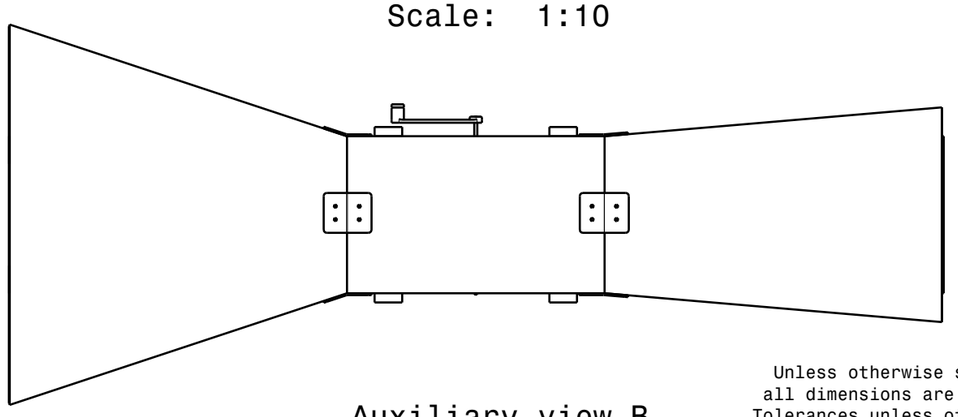
Rear view
Scale: 1:10



Right view
Scale: 1:10



Auxiliary view A
Scale: 1:10



Auxiliary view B
Scale: 1:10

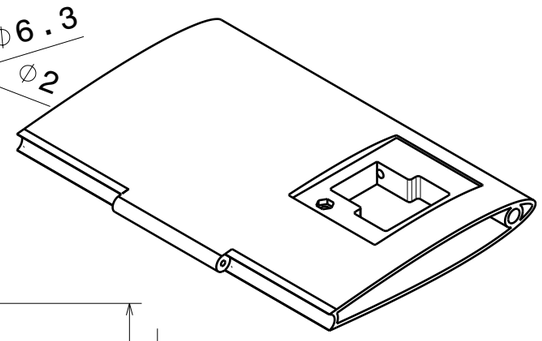
Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 1deg Decimals: 1mm

Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	J.DURAN	D4
	Date drawn	
A3	18/08/2018	Drawing Description
Scale	CATIA code	Flight Desk Control Demonstrator
1:10		

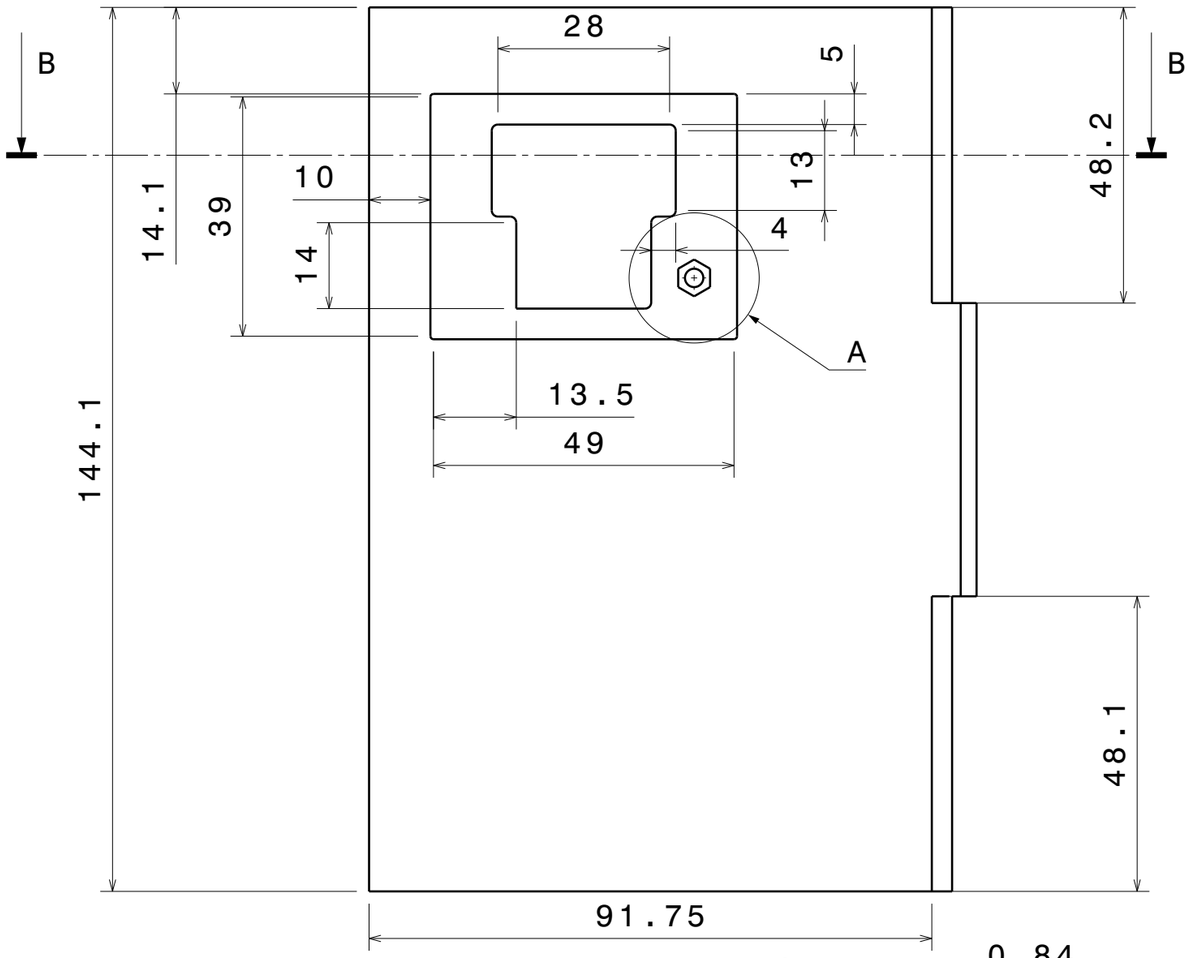
Drg No.1	Sht No.1	Rev. 1

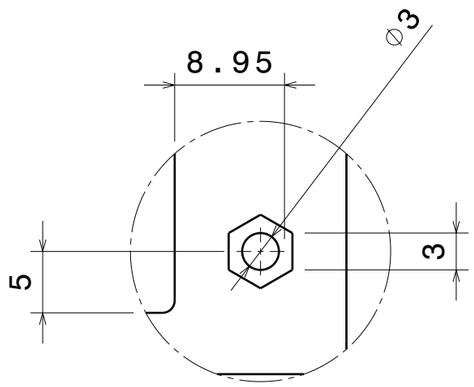
Zone	Rev.	Descripton of change

Bottom view
Scale: 1:1

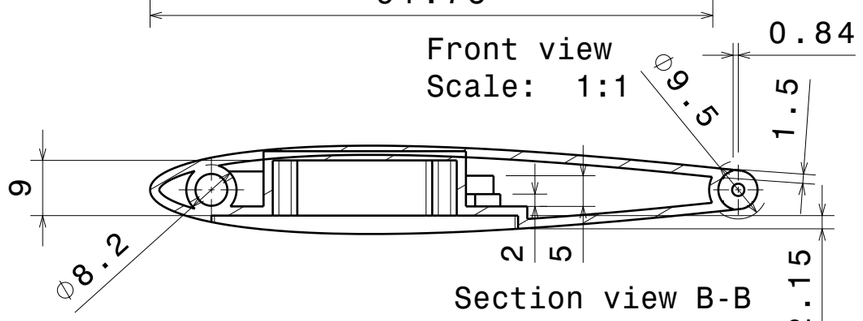


Isometric view
Scale: 1:2





Detail A
Scale: 2:1



Front view
Scale: 1:1

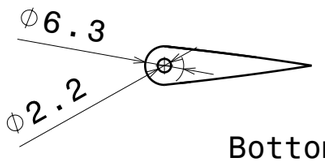
Section view B-B
Scale: 1:1

Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 1deg Decimals: 1mm

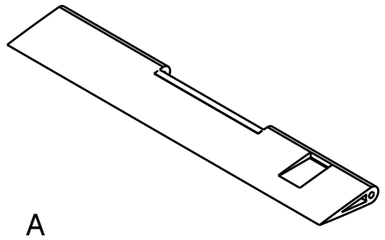
Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	J.DURAN	D5
	Date drawn	
A3	18/08/2018	Drawing Description
Scale	CATIA code	Aerofoil_Part_1
1:1		

Drg No.1	Sht No.1	Rev. 1

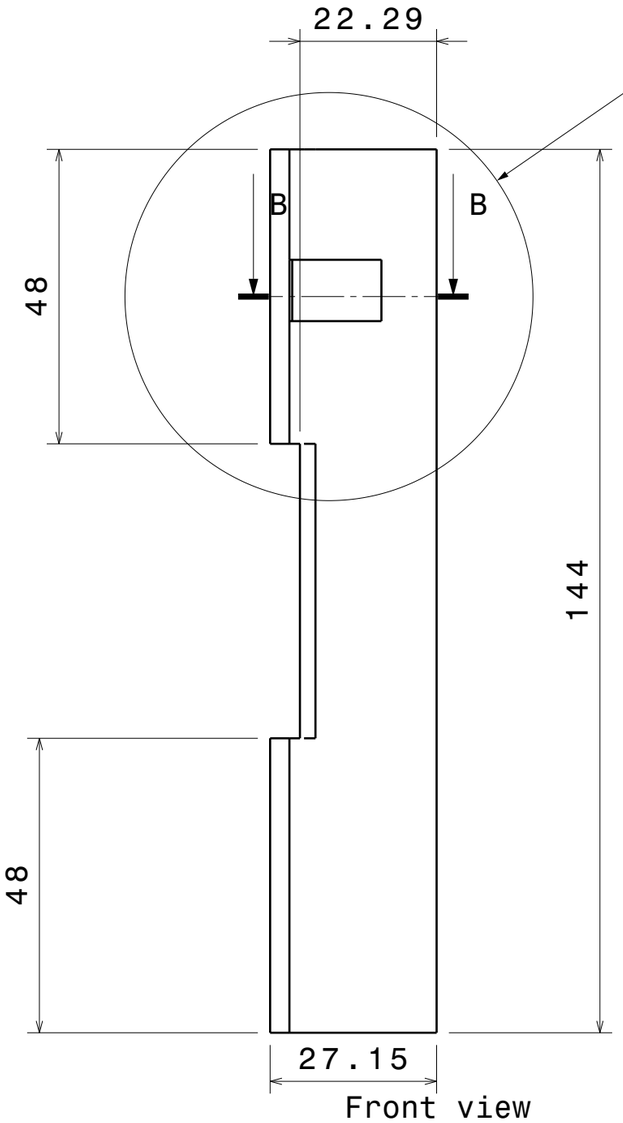
Zone	Rev.	Descripton of change



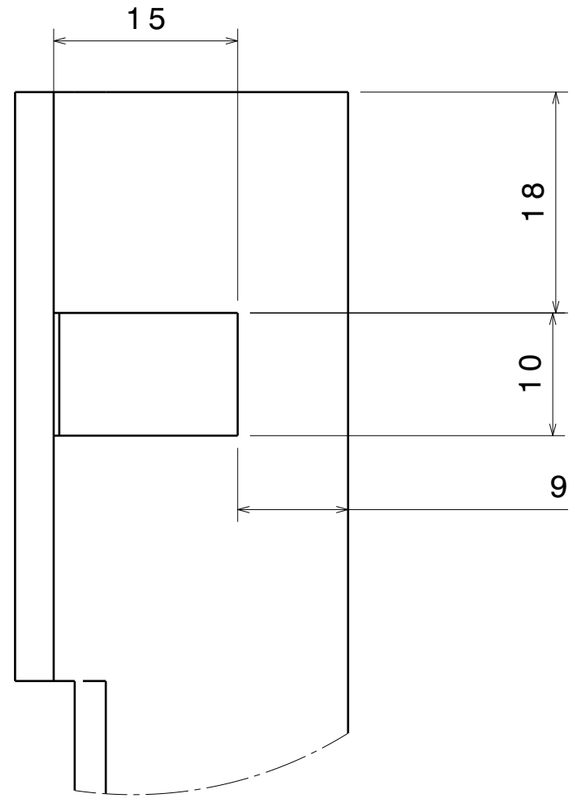
Bottom view
Scale: 1:1



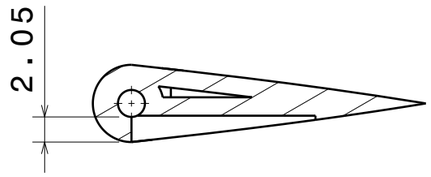
Isometric view
Scale: 1:2



Front view
Scale: 1:1



Detail A
Scale: 2:1



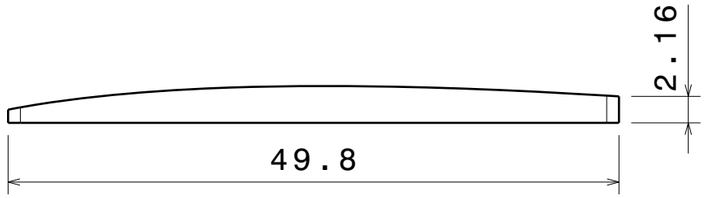
Section view B-B
Scale: 2:1

Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 1deg Decimals: 1mm

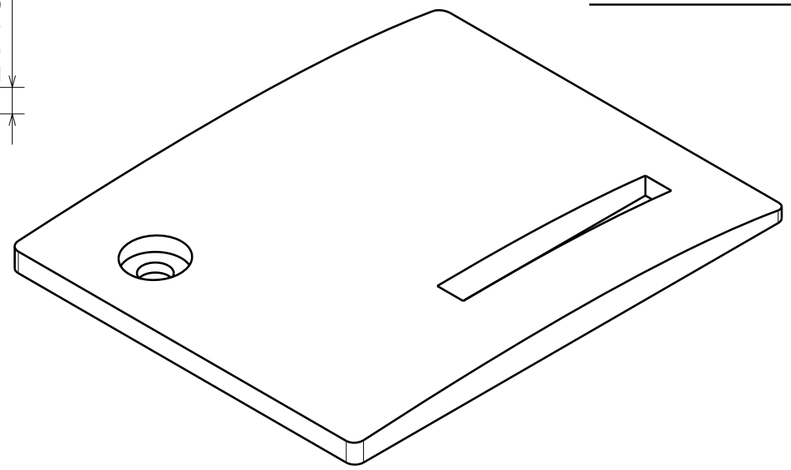
Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	J.DURAN	D6
	Date drawn	
A3	18/08/2018	Drawing Description
Scale	CATIA code	Flap_Part_1
1:1		

Drg No.1	Sht No.1	Rev. 1

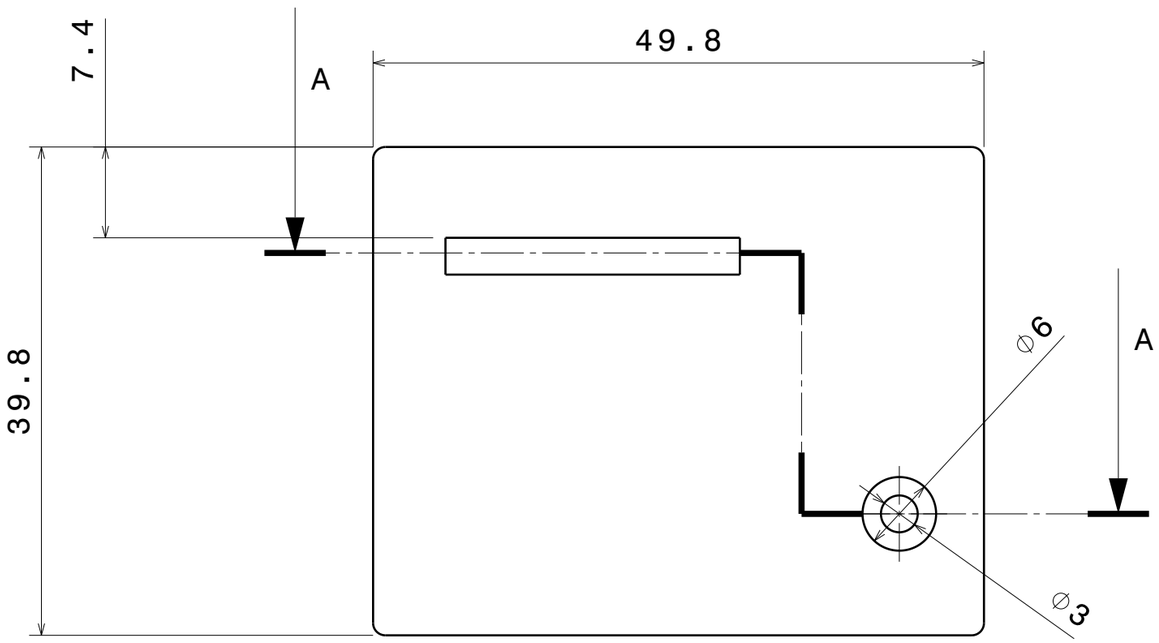
Zone	Rev.	Descripton of change



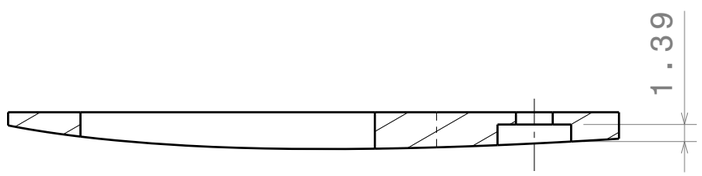
Bottom view
Scale: 2:1



Isometric view
Scale: 2:1



Front view
Scale: 2:1



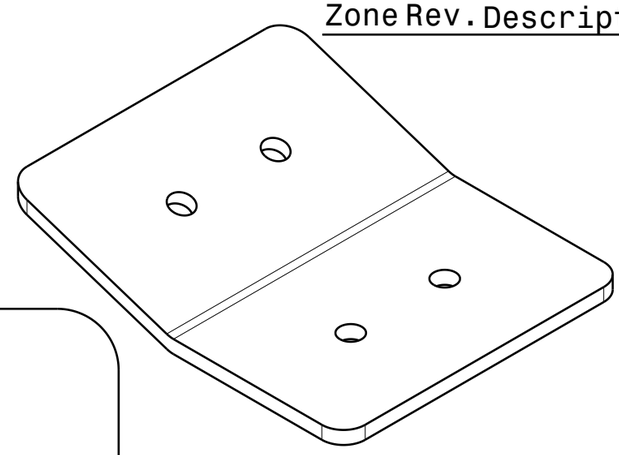
Section view A-A
Scale: 2:1

Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 1deg Decimals: 1mm

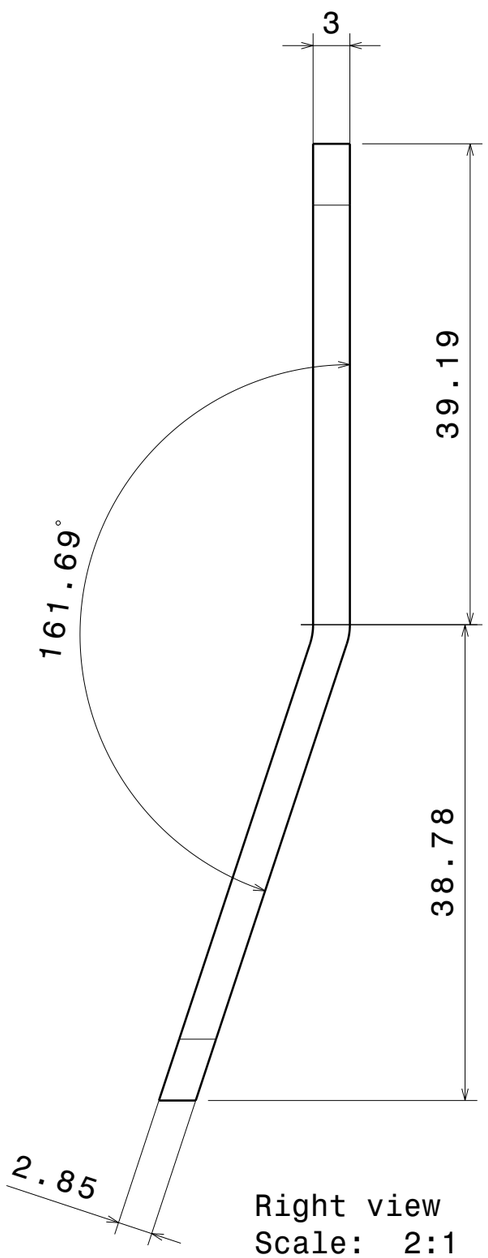
Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	J.DURAN	D7
	Date drawn	
A3	18/08/2018	Drawing Description
Scale	CATIA code	Top
1:2		

Drg No.1	Sht No.1	Rev. 1

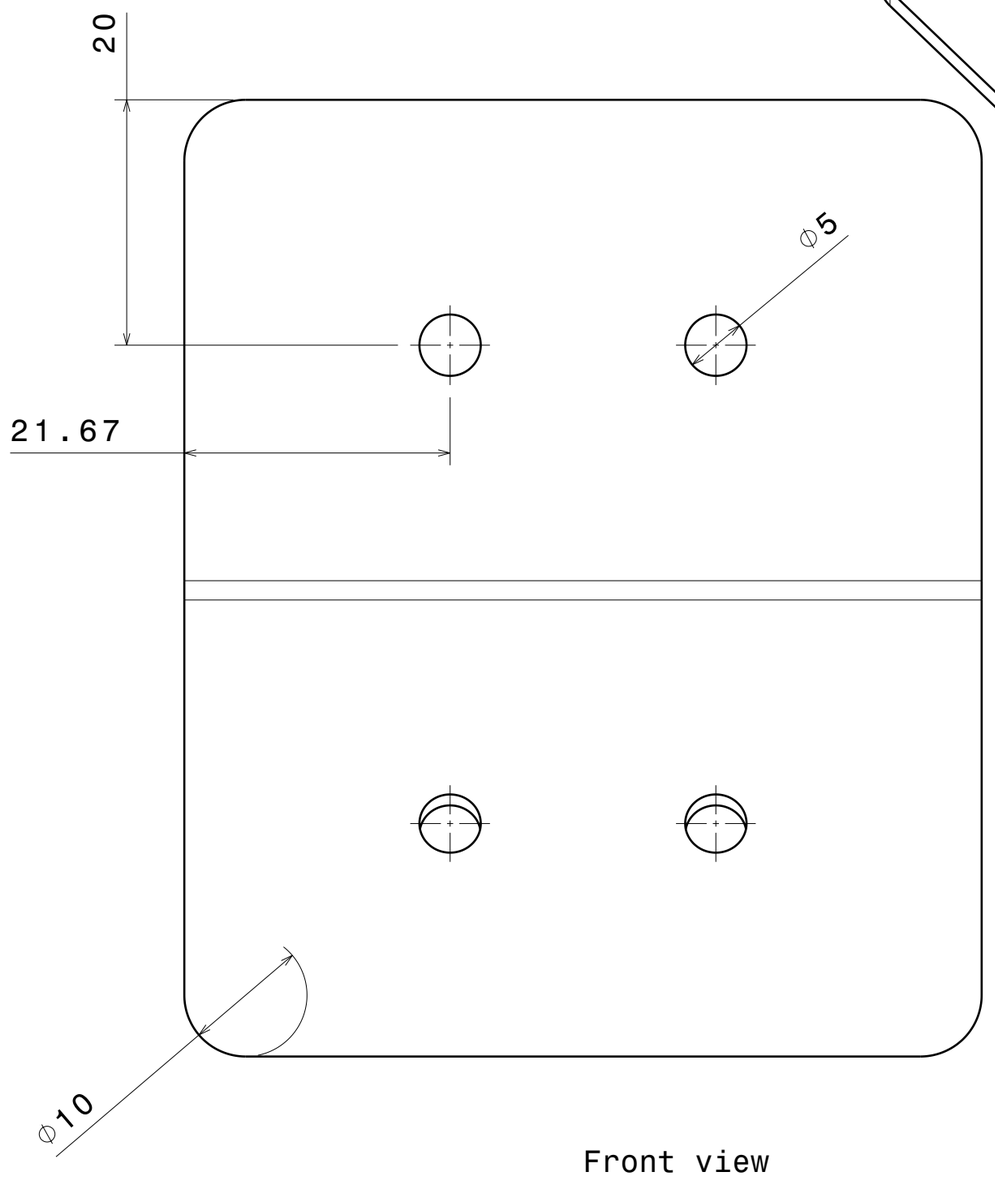
Zone	Rev.	Descripton of change



Isometric view
Scale: 1:1



Right view
Scale: 2:1



Front view
Scale: 2:1

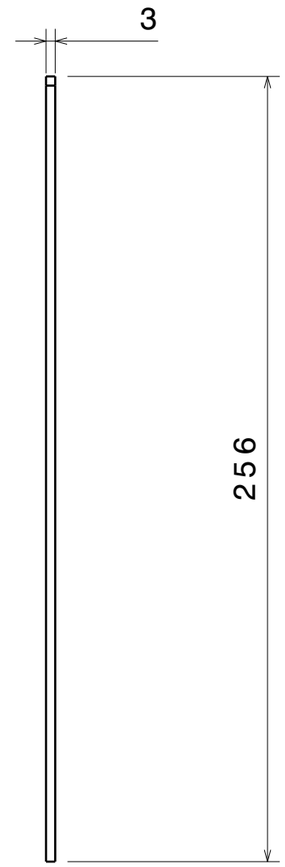
Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 1deg Decimals: 1mm

Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	J.DURAN	D8
	Date drawn	
A3	18/08/2018	Drawing Description
Scale	CATIA code	Nozzle Attachment
2:1		

Drg No.1	Sht No.1	Rev. 1

Zone	Rev.	Descripton of change

Right view
Scale: 1:2



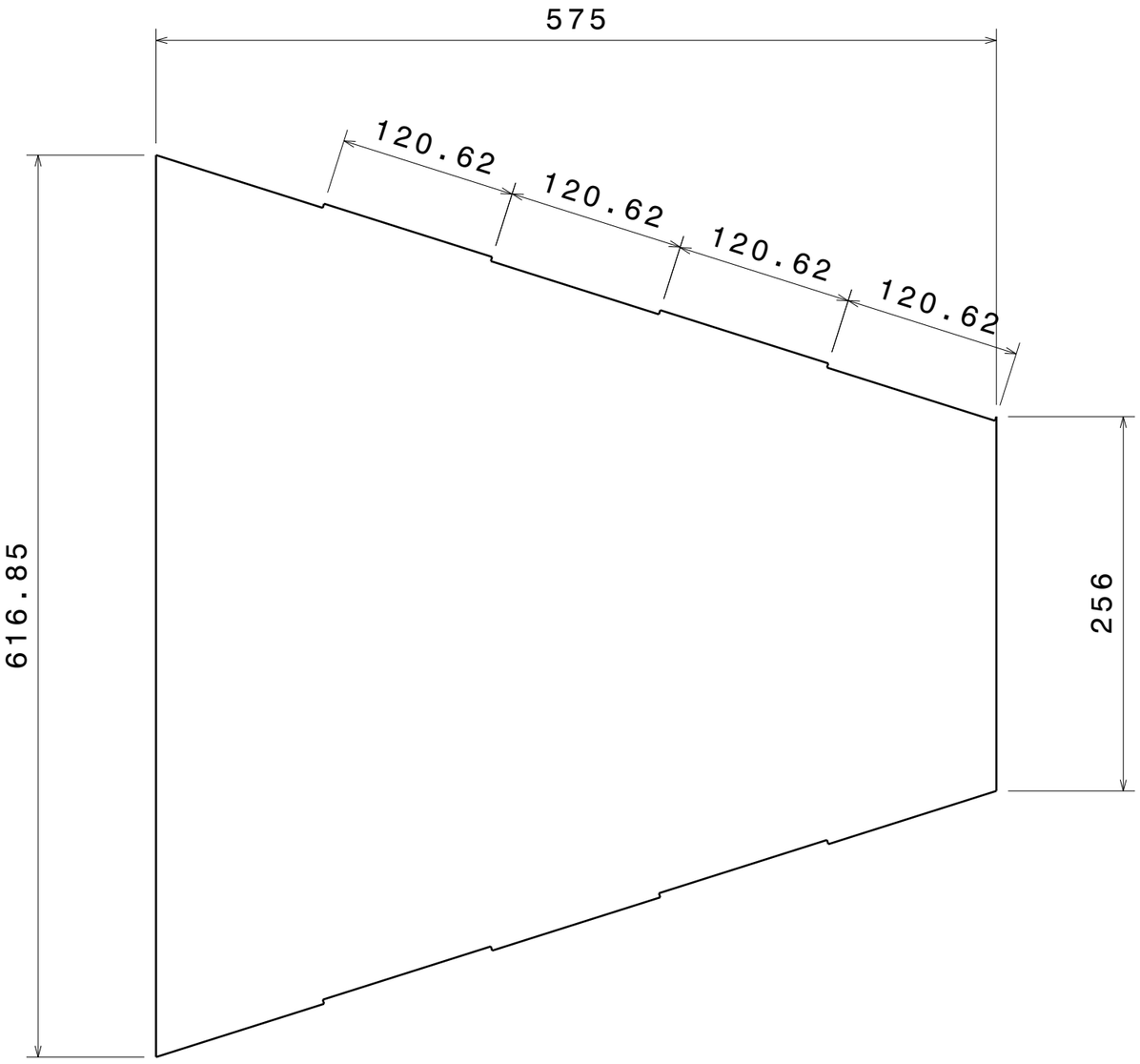
Front view
Scale: 1:2

Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 1deg Decimals: 1mm

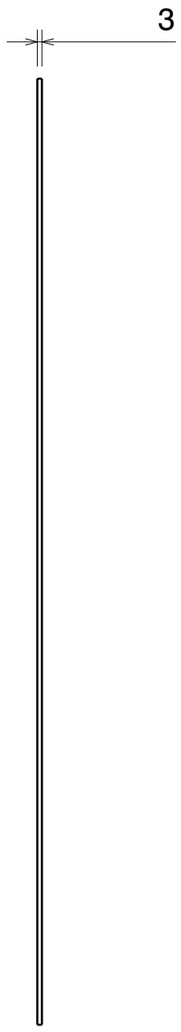
Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	J.DURAN	D9
	Date drawn	
A3	18/08/2018	Drawing Description
Scale	CATIA code	Chamber Panel
1:2		

Drg No.1	Sht No.1	Rev. 1

Zone	Rev.	Descripton of change



Right view
Scale: 1:4



Front view
Scale: 1:4

Unless otherwise stated
all dimensions are in mm.
Tolerances unless otherwise
stated are:
Angles: 1deg Decimals: 1mm

Group	Cranfield University Cranfield	
	Drawn by	Drawing Number
Sheet Size	J.DURAN	D10
	Date drawn	
A3	18/08/2018	Drawing Description
Scale	CATIA code	Nozzle Panel
1:4		