Design of a Three-axis Wind Tunnel Force Balance

Bueno Tintoré, Itziar

University of Zagreb
FACULTY OF TRANSPORT AND TRAFFIC SCIENCES

Itziar Bueno Tintoré

DESIGN OF A THREE-AXIS WIND TUNNEL FORCE BALANCE

BACHELOR THESIS

Zagreb, July 2018

University of Zagreb
FACULTY OF TRANSPORT
AND TRAFFIC SCIENCES
Vukelićeva 4, HR-10000 Zagreb
UNDERGRADUATE STUDY

Undergraduate study: Aeronautics
Chair: Aeronautical Engineering
Course: Theory of Flight I

UNDERGRADUATE THESIS ASSIGNMENT

Applicant: Itziar Bueno Tintoré
Matriculation number: 0135254109
Study programme: Aeronautics

Assignment title:

Design of a Three-axis Wind Tunnel Force Balance

Assignment title in Croatian:

Projektiranje trokomponentne aerodinamičke vage

Assignment description:

A force balance is the most often used tool to measure aerodynamic forces and moments in wind tunnels. The objective of this thesis is to design and build a force balance for a small low speed wind tunnel AT-1. The force balance should be able to measure lift, drag and pitching moment in small models. The specific thesis assignments are:
- design a stable enough model support structure
- make preliminary design of the force balance configuration and electronics
- design the force balance structure
- design the data acquisition and processing with Arduino
- design a force balance calibration system (including an electronic calibration Arduino program)
- purchasing of materials, mounting and initial testing of the system.

Supervising teacher:

Chairperson of undergraduate thesis committee:

Administrator

University of Zagreb
FACULTY OF TRANSPORT AND TRAFFIC SCIENCES

Itziar Bueno Tintoré

DESIGN OF A THREE-AXIS WIND TUNNEL FORCE BALANCE

BACHELOR THESIS

Supervisor: Karolina Krajček Nikolić, PhD

Zagreb, July 2018

Summary: In this thesis, the design of a force balance for a small low-speed wind tunnel capable of measuring lift, drag and pitching moment of small models is presented. The aim is to provide an accessible tool to do measurements in the faculty's wind tunnel, now used mostly for demonstration purposes. The design is intended to be inexpensive and understandable for students and it includes the base, the structure of the balance, and the means to transmit the forces to the sensors, along with the data acquisition methodology to process the results. The latter is made possible by using three load cells: two to measure lift and pitching moment, and another one to measure drag. Data processing is performed by an Arduino, which also enables a quick and easy calibration and tare process. The electronics and the accompanying code have been tested, while the structure design is left ready to be built. This work also includes a manual of use of the force balance.

Key words: Wind tunnel, force balance, Arduino, HX711, Load cell, three-axis, lift, drag, pitching moment.

Contents

1 Introduction

A wind tunnel is a resource used in aerodynamic research to study the behavior of the flow and its effect on bodies. It is one of the four main methods used for this purpose, which also include computer simulations (i.e., CFD), analytical problem solving, and making on-field measurements. Wind tunnel testing involves building a scale model of the object of the study and then measuring its response under specific conditions, which are usually adapted to be dynamically similar to the real. There are many kinds of wind tunnel applications, and specific tunnels exist for each of them. The wind tunnel of this thesis is classified as closed air circuit or Gottingen, meaning that the air inside the tunnel re-circulates. The test section of the tunnel is small and is used for aerodynamic experiments.

1.1 Objective of the study

The objective of this thesis is to design and build, if possible, a force balance for a small low-speed wind tunnel able to measure lift, drag and pitching moment in small models. The design includes its base, the structure of the balance and its means to transmit the forces to the sensors and the obtaining and processing of the results. It also includes calibration of the balance and a user manual.

1.2 Justification of the project and state of the art

A force balance is one of the four main methods to make available the forces and moments in wind tunnels, and the most frequently used one [1]. The second most frequently used method is the one implemented at the wind tunnel of the study, which consists in measuring the stress distribution over the model by means of orifices connected to pressure-measuring devices. This is not only possible with orifices but also with pressure or shear sensitive coatings. Other methods such as measuring the effect that the model has on the airstream by wake surveys and tunnel wall pressures and measuring the motion of the model under the action of aerodynamic forces and computing the forces from equations of motion are also widely known [1].

The implementation of a force balance will provide an accessible tool to do measurements in the wind tunnel. This will improve upon the existent method, which requires the model to be tested to follow a specific design, making it difficult to obtain data. This thesis is an attempt to extend the use of the wind tunnel of the faculty. Even though the wind tunnel of the study has been used to run some experiments, it is generally only used for demonstration purposes. This project is a good way to exploit the resources of the faculty as the tunnel will be able to be used not only for demonstrational purposes but also for testing.

This load measurements to make available the forces and moments will be done by measuring directly them with a balance. A balance is expected to separate force and moment components in a reference frame, and there are two fundamental types for measuring total model forces and moments in general use. External balances, which carry the loads outside the tunnel before they

are measured and internal balances, which fit into the models and send data out through electrical signals [1].

For wind tunnel applications, the reference frame used is wind axes. Wind axes have x_w pointing to the wind, z_w pointing down and y_w pointing to the right looking into the wind [1]. In this reference frame lift is in the negative z_w direction, drag in the negative x_w direction and side force in the positive y_w direction. On the *x*, *y*, *z*, axes moment components are rolling moment, pitching moment and yawing moment respectively [1]. All these components are generally measured with a force balance, but the scope of this project is to get to measure only three of them for the following reasons.

The force balance is intended to be built as simple and accessible as possible. That is the reason why building a 6-component balance was rejected in favor of a 3-component, as calibration, mounting and data obtainment for a 6-component balance is complex and expensive. Also, as the testing section of the wind tunnel is small, the balance is intended to be used to measure loads in small wing models, which are normally symmetrical, so that being able to measure side force, rolling and yawing moment is unnecessary. This simplifies the problem of building the balance, often defined as among the most challenging problems in the field [1]. However, it is interesting to be able to measure the lift a wing can give and its drag force, as well as its pitching moment, which is useful to find the center of pressure of the wing.

Making it simple and accessible is also the reason why data acquisition with Arduino is chosen, as it is a simple and widely extended way of processing sensor data among young engineers, in a way that it makes testing for students more accessible. Arduino is also a modern, simple and inexpensive way of obtaining data and offers a lot of modification and extension possibilities that can be implemented afterwards. Also, the sensors used for measuring the force are simple and easy to understand for students.

The design presented in this work is preferable than the one currently in use in Barcelona [2] or other wind tunnels [3], as the main goal was to have a simple setup, both for its users and for its construction, and the aforementioned experimental setups require the use of expensive data acquisition software and hardware (i.e., LabVIEW).

1.3 Methodology and scope

This thesis is structured following the design process of the balance. Firstly, the preliminary design of the balance —necessary to define its shape and basic configuration— is presented, considering multiple options. The proposals will include basic features, like amount and type of sensors, the electronics to be used and the general configuration of the load transmission system. Then, one of them is chosen among the two final preliminary designs to delve into it, specifying all its components and measures. Its structure will be defined; including the base, load transmission and angle of attack adjustment. Also, once the preliminary design is chosen, the data acquisition

and processing system can be designed and built, defining the sensors and the necessary electronics, as well as the code needed to read and process data. Finally, construction and calibration of the whole structure can be done.

2 Preliminary design

The balance design must follow a series of specifications to fulfill its objective, being able to measure the two forces and the momentum mentioned before in an enough accurate way. To do so, it is necessary that there is no interaction between measurements and that friction does not cause the system to be non-linear. It also must be intuitive and easy to use as its intention is to be used by other students of the faculty. With data of the tunnel (mentioned before) and its objective, the two following preliminary designs are proposed. Both designs use Arduino for data acquisition and processing.

2.1 First proposal

First proposal consists of a base holding a beam that is introduced in the wing cross section. Measures of lift, drag and momentum are done with the movement of this beam. A momentum rotor sensor, as used in other experiments in the field [4], (or two, to make it symmetrical and avoid a ball joint with friction) and four load cells are needed for the measures.

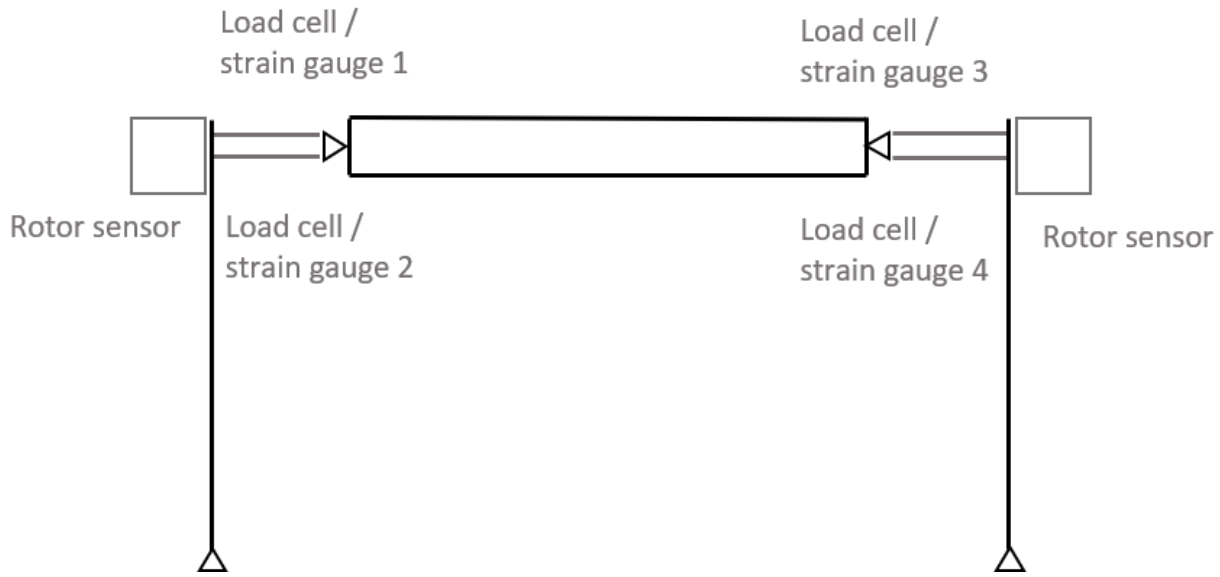


Figure 1- Front view of the first proposal preliminary design.

Table 1 - List of advantages and disadvantages of design 1.

Advantages	**Disadvantages**
No friction or interaction with the different measurements.	More complex to design in order to avoid this interaction. No literature.
Easier to use and intuitive. Possibility to find the center of pressure visually if the wing is designed to do so.	A momentum rotor sensor is needed. No literature about how to measure with these sensors with Arduino.

Easy to adjust angle of attack.	Need to design a system to adapt the balance to different wing widths.
Easier to build with existing base.	Difficult to calibrate.
Literature about a similar design.	Higher number of sensors needed, to make it symmetrical more mount of cells are needed.
	Difficulties to design a system to transmit loads efficiently.
	More complex data acquisition program.
	More quantity of pieces to design and build specifically for the project.
	Need of different materials and study of their properties.
	Electronics and kind of load cells cannot be easily defined.

2.2 Second proposal

Second proposal consists of two vertical bars holding the model to measure lift and momentum and one horizontal surface to under them to measure drag. Three beam load cells will be used as measuring sensors and HX711 amplifiers will be used to transfer the signal to Arduino.

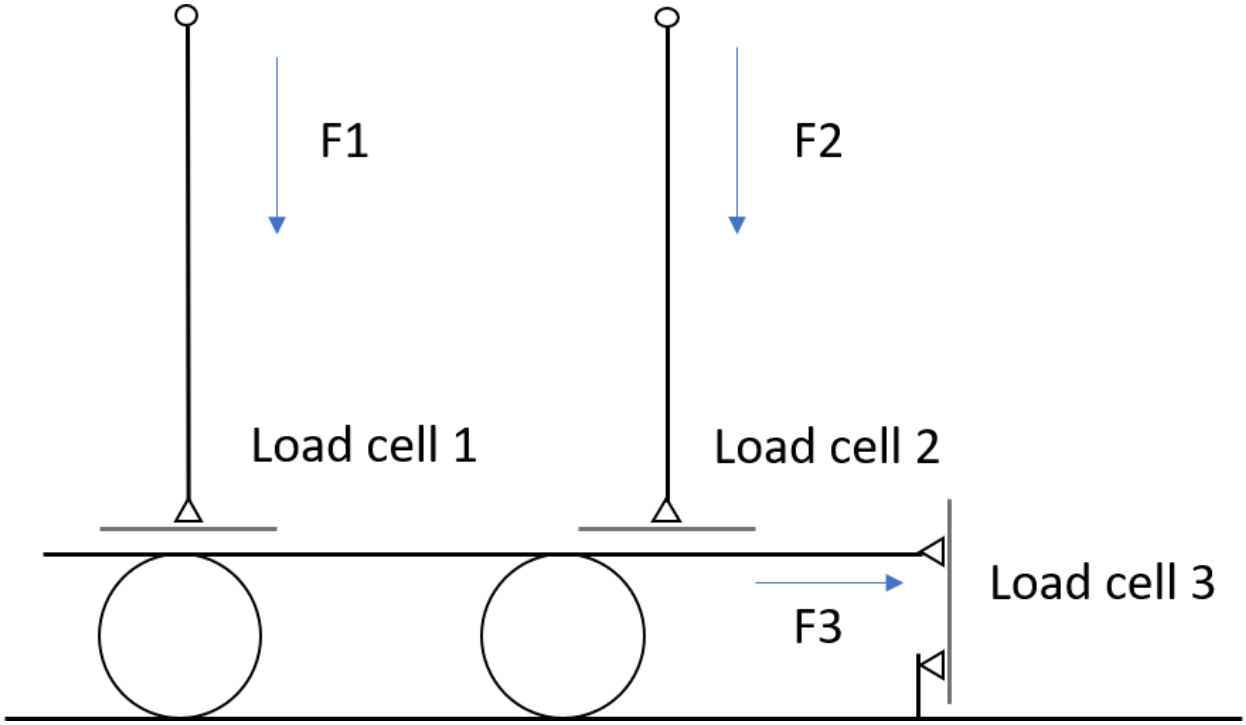


Figure 2- Side view of the second proposal preliminary design.

This design needs to have the following joints to transmit the loads but be able to regulate angle of attack. The vertical bars need joints at the base, as it can be seen in Figure 2, to transmit all the drag force to the platform and not bend, but the free edges at the top need ball joints, as when calibrating the angle of attack the model must be able to turn freely around the *y* axis. The load cell 3 also needs joints with the sliding platform and the base.

Table 2 - List of advantages and disadvantages of design 2.

Advantages	**Disadvantages**
Only one kind of load cell. Literature about how to do it with Arduino.	Having all the structure on a moving apart may reduce drag measurement precision
Easy to mount load cells.	Need of a more complex system to regulate angle of attack.
Intuitive separation of the loads.	Angle of attack regulation is not so user-friendly.
No need to do a deep study to choose materials.	More complex force balance base design.
Most of the pieces can be easily found and tested at shops.	No literature about the load transmission design.
Less amount of load cells needed.	
Even though the base design is more complex it can be easily design and attached to the existing structure.	
Simpler data acquisition program.	
Literature about similar angle of attack regulation system.	

As the objective of the thesis is to design a cheap and easy system to do measurements in the wind tunnel, the second option is chosen, as its design and construction contains less complex pieces and lower number of sensors, which implies a reduction in price. As already existing pieces are used, they can be tested at shops to ensure that they work in the desired way. Also, time is limited, and the time saved in specific piece design can be used to improve data obtainment, focus on the points that can be improved to make its use more understandable and design the electronics to be more adaptable to new applications in the future as well as write a more detailed manual of use. This also makes that the thesis is more focused on the real use of the balance and less on material properties and load transmission, which adjusts more to an aeronautical engineering project.

3 Force balance structure

3.1 Structure base

The already existing structure of the pressure distribution measuring equipment will be used as base for the force balance to take advantage of the already existing resources. It fulfils the height and stability requirements to not interfere with results, as being metallic makes it rigid enough and has a good subjection to the floor. It is also easy to attach to other structures due to its shape and holes.

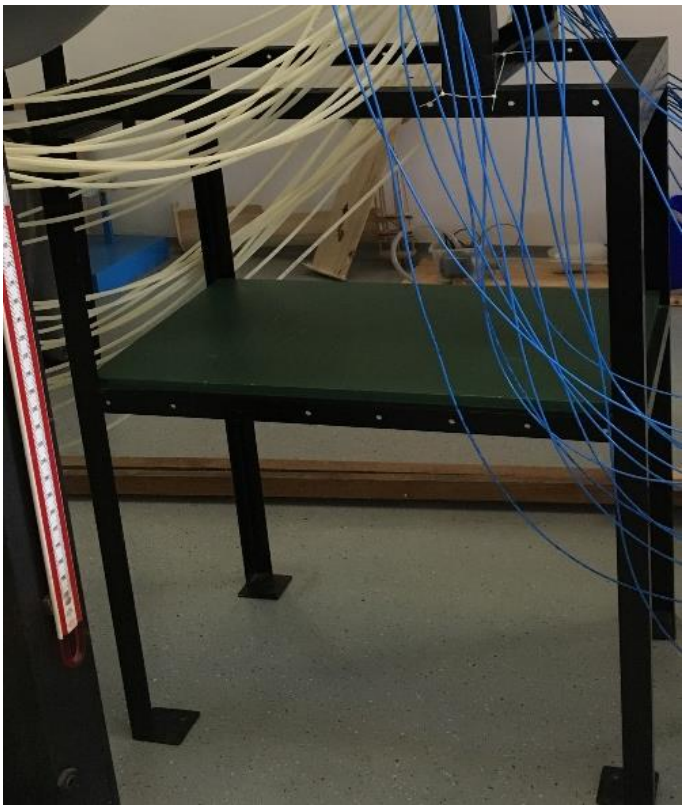


Figure 3- Structure for the pressure distribution measuring equipment.

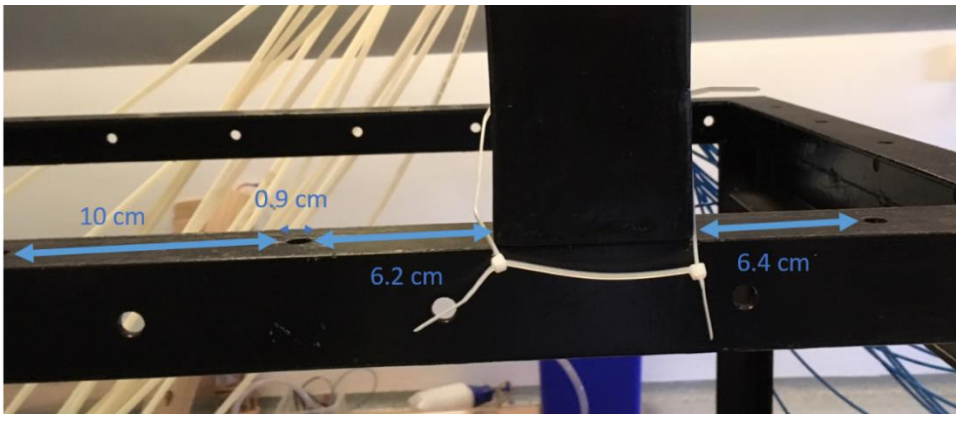


Figure 4- Hole measures in the structure.

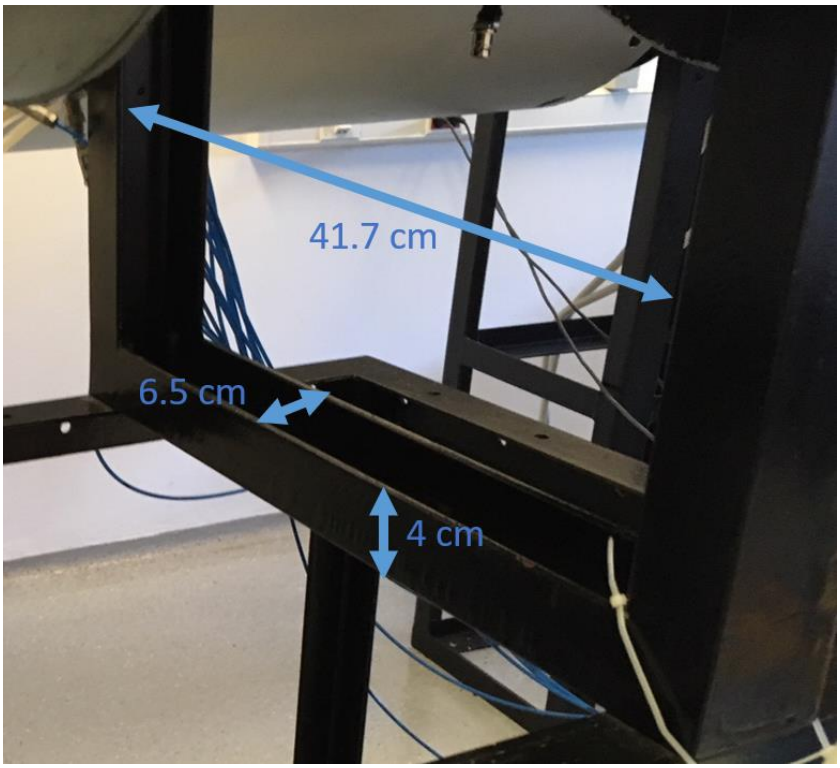


Figure 5- Measures of the vertical attached structure of the base.

The structure of the balance consists of a base that is used to attach all the elements and a moving platform in which the load cells will be placed. Two of them (lift) on it and the third one (drag) at the side and attached also to the structure base. The base holds a sliding wheel on which the sliding platform is placed. The base is designed to hold a sliding wheel of 32 mm outside diameter, 12 mm of inside diameter and 10 mm width.

The base piece is designed to be attached to the holes in the structure with bolts. As the shape of the existing base is not regular, the edges of the structure will not be directly jointed with the base and the distance will be covered with bolts which are suggested to be placed in an adjustable way. The base of the balance is restricted by the shape of the existing structure and elements already existing on the wind tunnel. The existence of the element of Figure 6, forces the position of the model to be at the left, as the bars of the balance cross the wind tunnel through a hole.

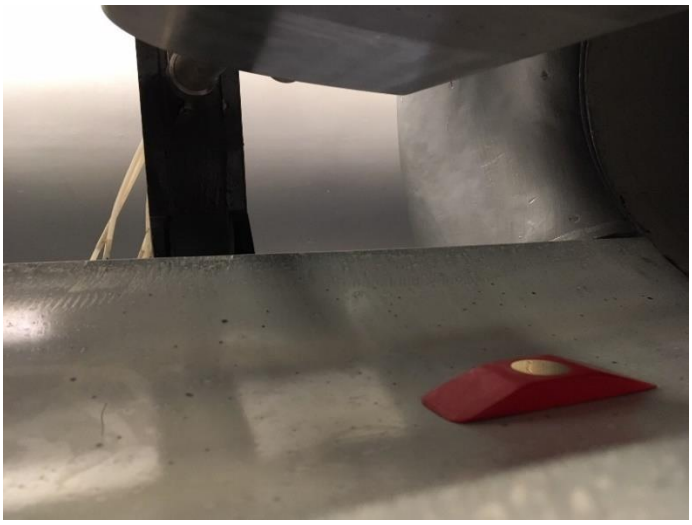


Figure 6- Existing element at the tunnel.

The length of the bars to transmit the loads (Figure 10) are restricted by the height of the testing section (see Figure 5), the base and load cell height and placement (2 cm), the position of the vertical load cell and the height regulation system (8 cm).

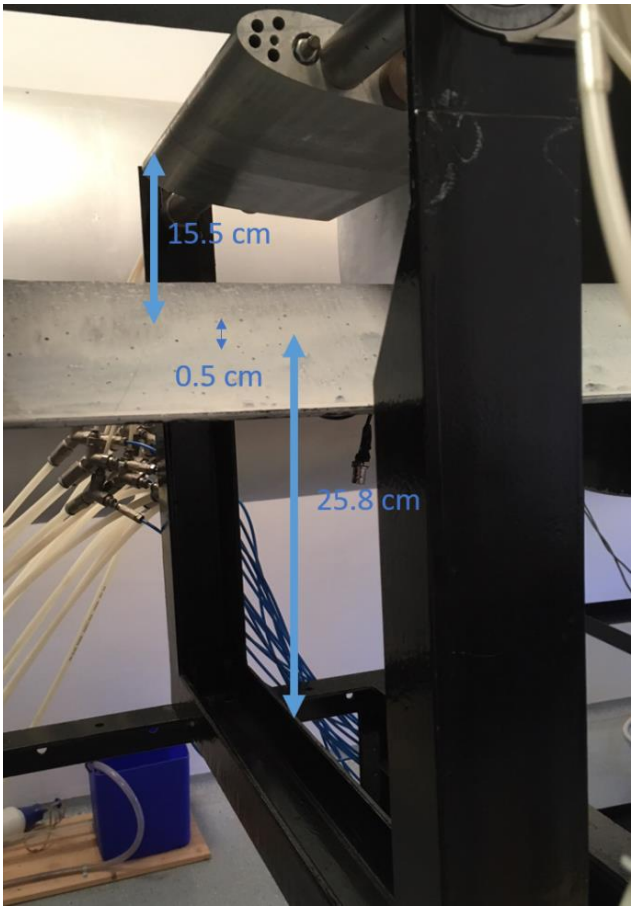


Figure 7 - Vertical measures of the structure and the testing section.

3.2 Load transmission

As the wind tunnel test section is horizontal and the wind direction is perpendicular to the direction of Earth's gravity, the wind axes are parallel to ground axes. This means that the lift is parallel direction to Earth's gravity and drag is perpendicular to it. This is the reason why the lift measurement will be done vertically and drag measurement will be done horizontally.

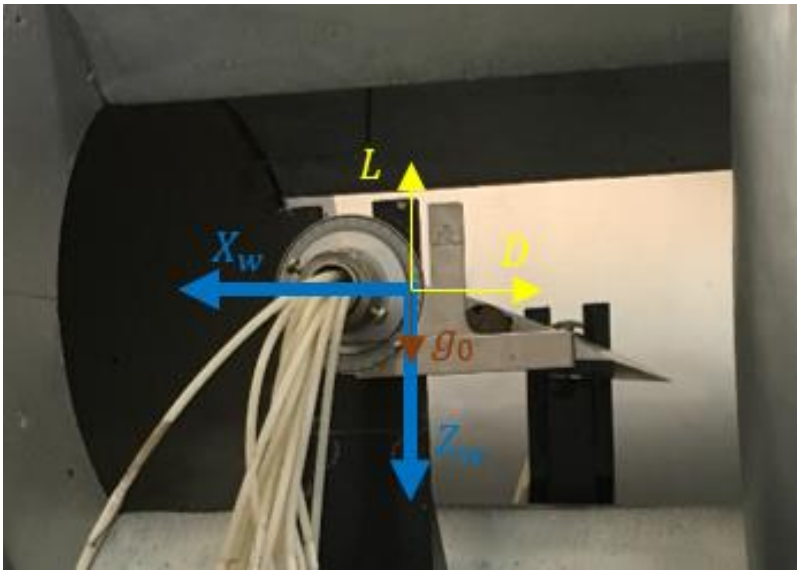


Figure 8 - Wind tunnel test section with wind axis and forces.

Before doing the measurements the horizontal and vertical load must be separated and transmitted to the sensors minimizing interferences between them in order to read the most near to the real value. As the objective of the project is to be able to read 3 different values, at least 3 sensors must be used.

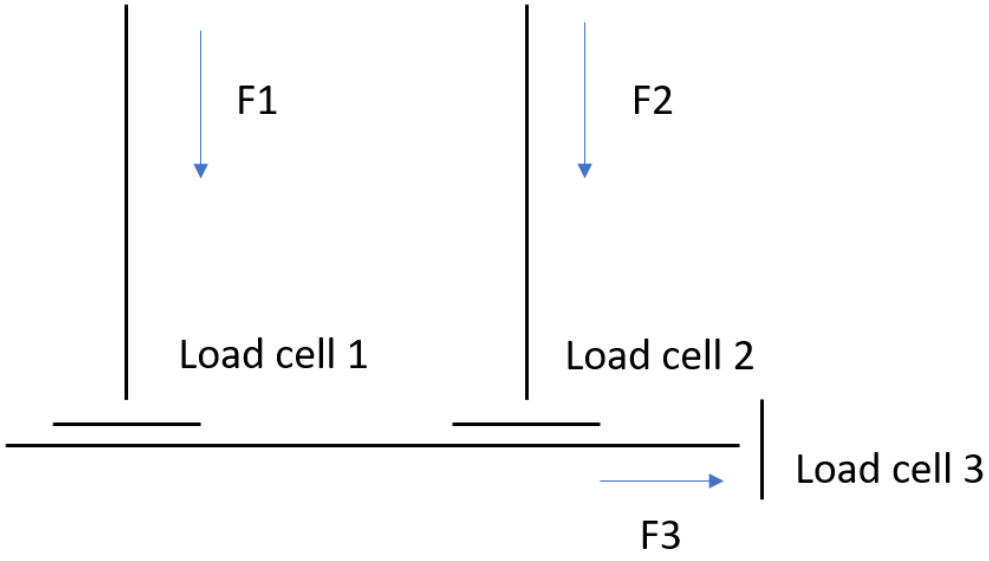


Figure 9 - Design for the load transmission of the balance.

For the load transmission, the second proposal configuration is chosen to separate and transmit the loads, where two sensors reading loads in the vertical axis are used to measure lift and pitching moment and one sensor reading loads in the horizontal axis is used to measure drag. Load cell 1, 2 and 3 (Figure 9) measure F1, F2, and F3 or drag respectively.

3.2.1 Lift and pitching moment transmission

Two vertical bars (main load carrying members), which are rigid enough to not absorb any of the horizontal force by bending, transmit the load from the two joints with the model to their respective load cell located in a platform. The shape of the bars is a prism with enough height to reach the testing section from the base of the balance and a prism base that is long in the wind direction, to avoid bending, and thin in the perpendicular direction of the wind to create the minimal interferences with measurements. The position of the load cell only allows measuring of vertical loads and the platform is placed in a way that all horizontal loads are not absorbed there.

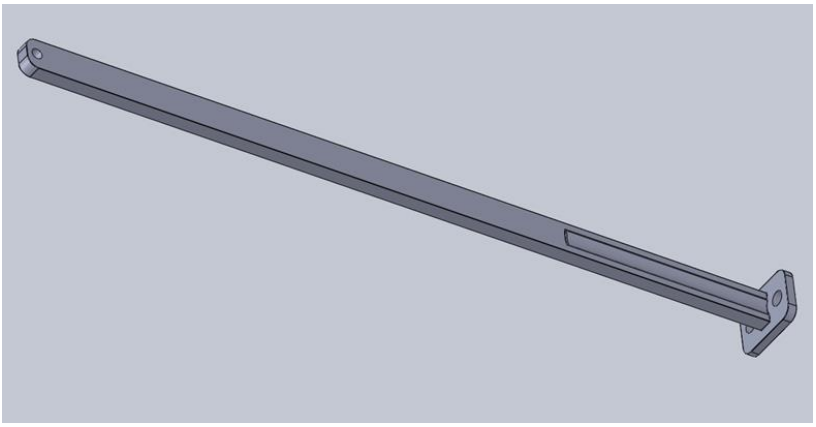


Figure 10- Vertical bar to transmit the load to the load cells

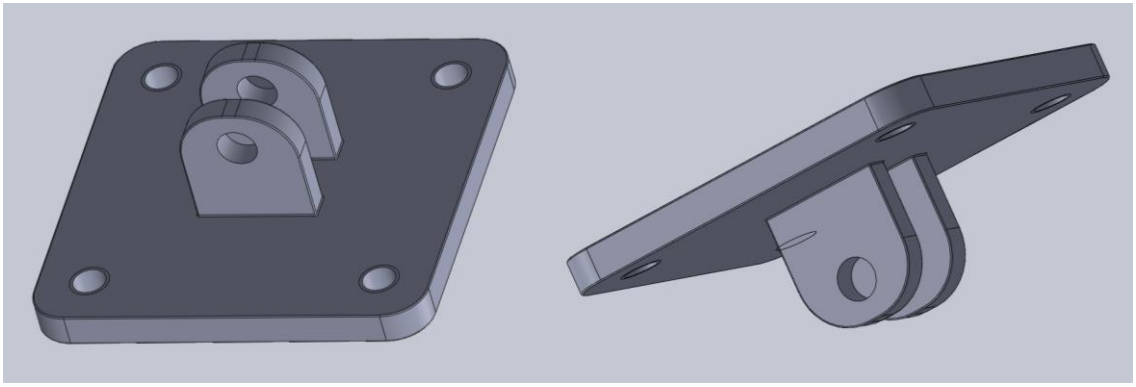


Figure 11- Top piece to attach the model.

The sum of the difference in vertical loads measured when the wind tunnel is on and off will be the lift force of the model. The pitching moment could be calculated measuring forces in two points in the same axis, but it is decided to be calculated in lift axis as it is the easiest due to the

balance configuration and as the lift force is the biggest force, the measurement in this axis is the most accurate.

3.2.2 Drag transmission

The main load carrying member is the base platform of the other load cells, which could move freely in the wind axis if the load was not transmitted to a load cell that can measure force only in the horizontal position. All the load in the horizontal axis is transmitted to this load cell, except for the absorbed by the friction of the sliding platform, as union between bars and load cells are strong enough not to absorb drag.

3.3 Angle of attack adjustment

The system to adjust the angle of attack is based on an already existing system used on two axis force balances [5] and consists of a protractor that can be placed at the back of the testing section to measure inclination and a bolt system that is located under the testing section and attached to the lift and pitching moment load cells. Two 8 cm long bolts with their respective flat nut that can move up and down are attached to each load cell (see Figure 12, left side of the load cell), and each of them is then tied to a vertical bar (Figure 10) that is placed on the flat nuts. When the desired inclination is selected, the pieces can be tighten to maintain their position.

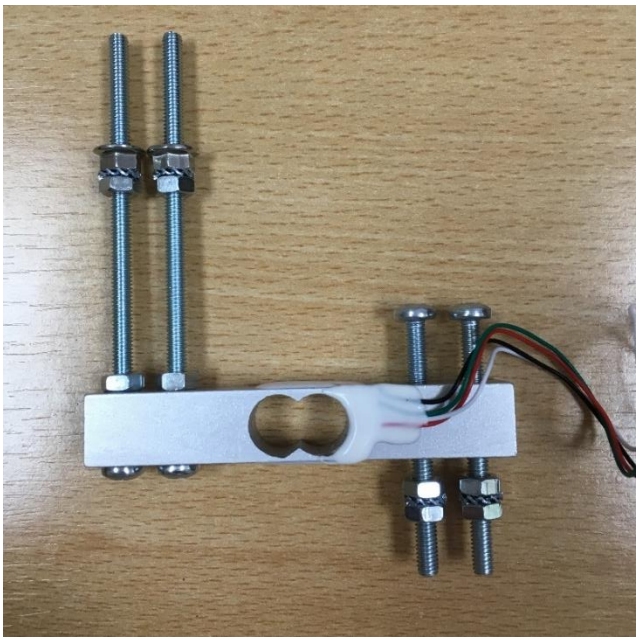


Figure 12- Lift load cell attached to the height regulation system.

The maximum angle of attack designed to be reached with the system is ± 45 degrees, which defines the distance between load cells, the maximum distance possible between them to be able to reach this inclination, as to the 8 cm length of the bolts, the height of the load cell and the occupied by the nuts and the bar base reduce the moving distance to 4.1 cm. It is very important

that the system is very tightly adjusted to eliminate any possible vibration that could cause noise in the measurements.

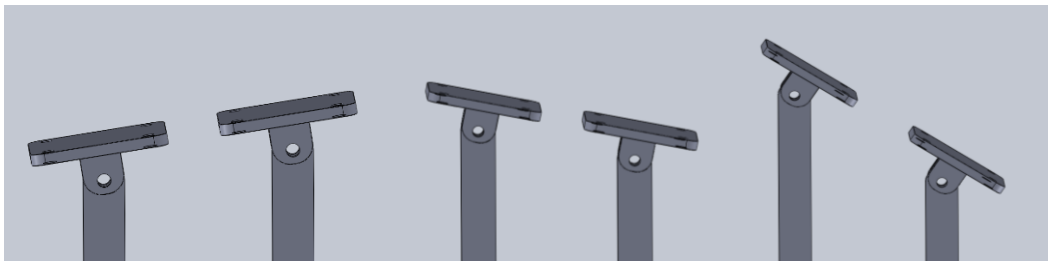


Figure 13- Movement of the top piece of the balance with bar height change.

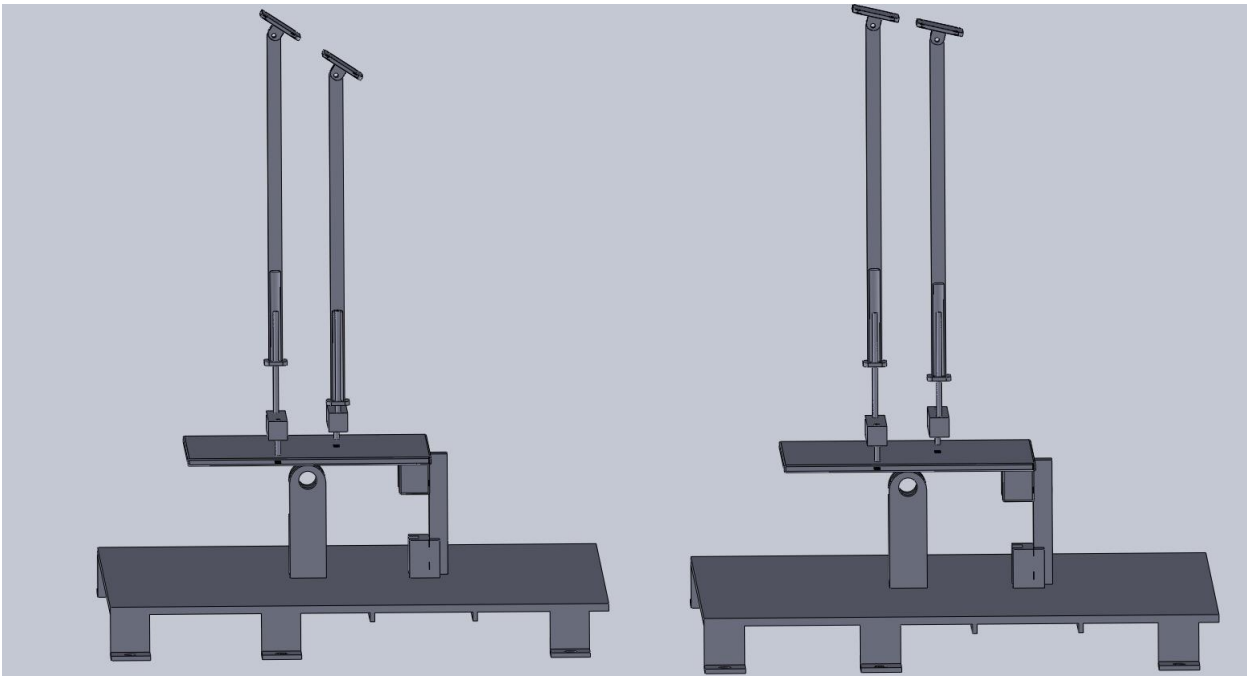


Figure 14- Configuration of the balance at different angles of attack

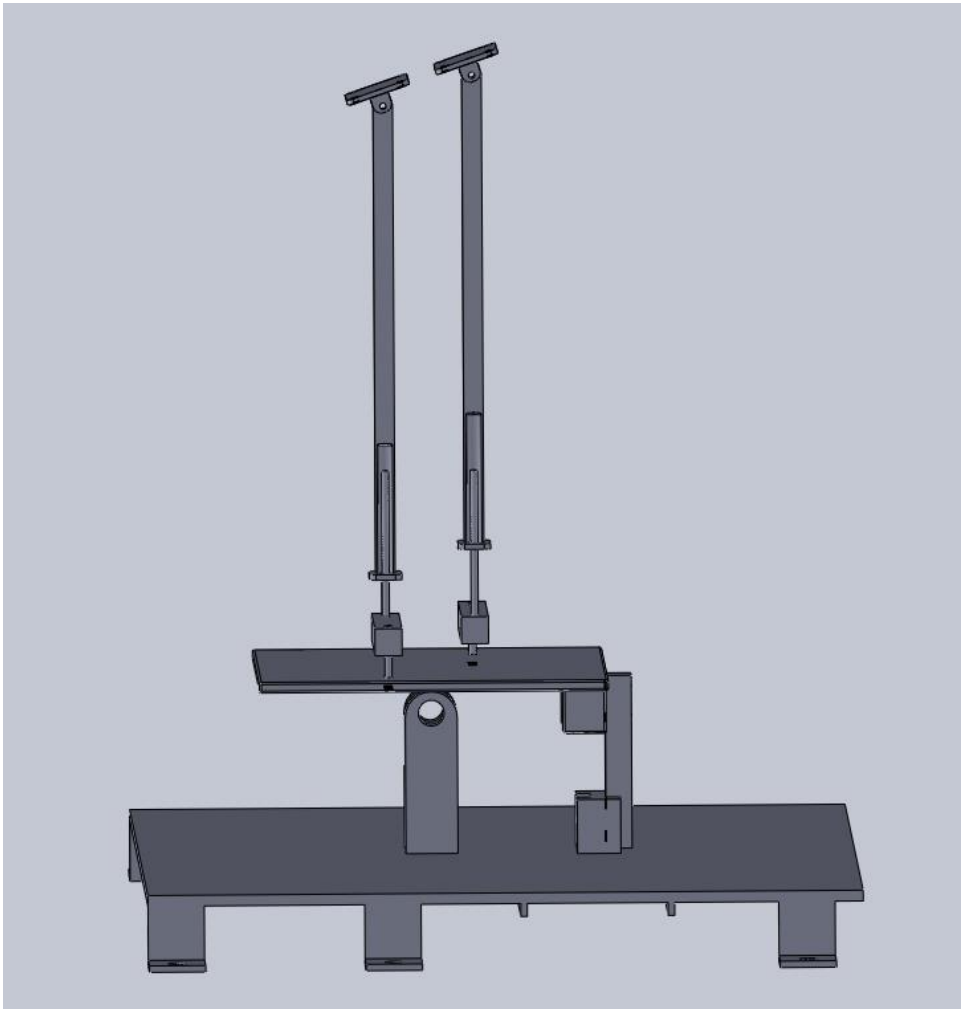


Figure 15- Configuration of the balance with a negative angle of attack.

4 Data acquisition and processing

The system to obtain data is composed of three load cells with their respective HX711 amplifier connected to an Arduino that is connected to a laptop to read the data. To more detailed information about how to read the data from the laptop the user's manual is included at the annex (page 48).

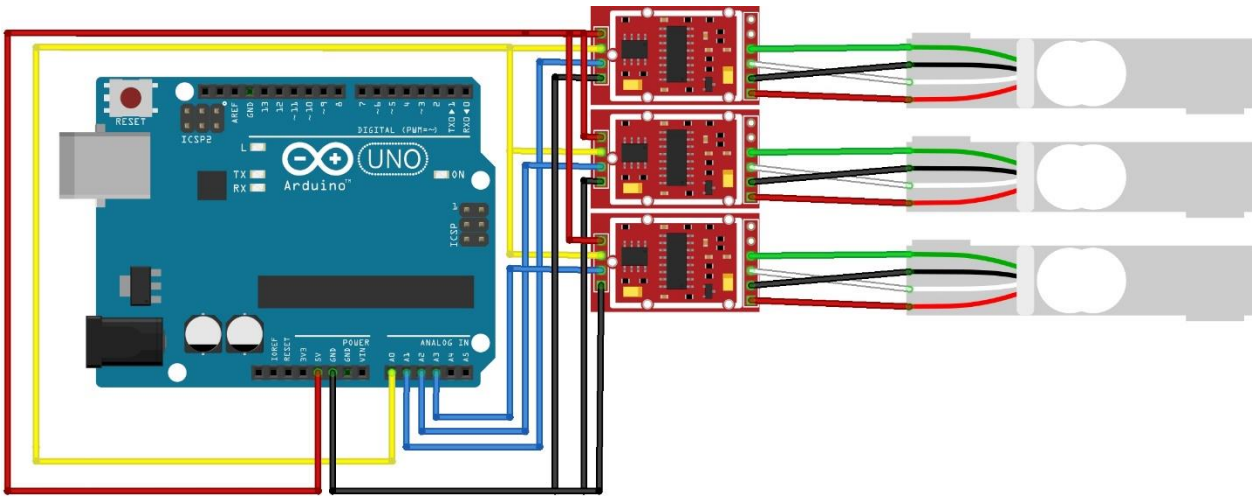


Figure 16 - Scheme of the connections of data acquisition system.

4.1 Load cells

The measurement of the forces is done by three load cells, two of them are able to measure up to 5 kg and are used for lift and pitching moment and another one able to measure up to 1 kg is used for drag. All of them can measure positive and negative forces.

The maximum axial force, force in the direction of drag, measured in the wind tunnel of the study for a scale 1:100 DHC 400 model with S_{ref} = 0.00634 m^2, is 0.318 N, while the maximum normal force, corresponding to the one in the direction of lift, measured is 29.8 N [6]. This model is representative of the models that can be tested at the wind tunnel and the data obtained is used to calculate the approximate maximum drag and lift load in kg that can be reached with the wind tunnel, which are 0.033 kg and 3.04 kg respectively. The size of the load cells must be enough to measure this maximum force and, in the case of the lift load cells, to support the weight of the model. They should also have a size that is adequate to the scale of the measurements they are doing in order to make precise measurements. Two 5 kg load cells in the lift axis allow a maximum load of 10 kg, which gives a wide enough margin, and the 1 kg load cell is also enough for measuring drag force.

4.2 HX711 Amplifier

This amplifier is used because it is designed for weight scales and industrial control applications to interface directly with a bridge sensor and its communication protocol allows the connection

of the same clock (yellow wire Figure 16) for all of them, so that only 4 pins for data are needed. This type of connection eases the addition of new functions or sensors to the Arduino that could be done in the future. HX711 amplifier is also the simplest to use with Arduino and a lot of literature and libraries are available for it.

This amplifier is a precision 24-bit analog-to-digital converter (ADC), what makes the acquisition of the data more precise than receiving an analogic signal directly with Arduino, as it has only a 10-bit analog-to-digital converter. To not interfere with measurement precision (load cells have 0.05 % error, which means a ratio of 2000 : 1 the minimum number of bits would be 11, or 12 taking into account that values can be positive or negative, ($2^{10} = 1{,}024$ and $2^{11} = 2{,}048$) which makes 24 bits more than enough to ensure that electronics will not interfere in precision ($2^{24} = 16{,}777{,}216$).

The communication protocol of the HX711 amplifier is not normalized and uses the following process. The amplifier has three different gains that can be chosen, although for the project only gain 128 will be used (channel A Figure 17). Pin PD_SCK (yellow wire Figure 16) and DOUT (blue wire Figure 16) are used for this gain selection and also for data retrieval, input selection and power down controls. When output data is not ready for retrieval, digital output pin DOUT is high. Serial clock input PD_SCK should be low. When DOUT goes to low, it indicates data is ready for retrieval. By applying 25, 26 or 27 positive clock pulses at the PD_SCK pin, depending on the gain chosen (see Figure 17) as input and gain selection is controlled by the number of the input PD_SCK pulses, data is shifted out from the DOUT output pin (Figure 17). Each PD_SCK pulse shifts out one bit, starting with the most significant bit first, until all 24 bits are shifted out. The 25th pulse at PD_SCK input will pull DOUT pin back to high [7].

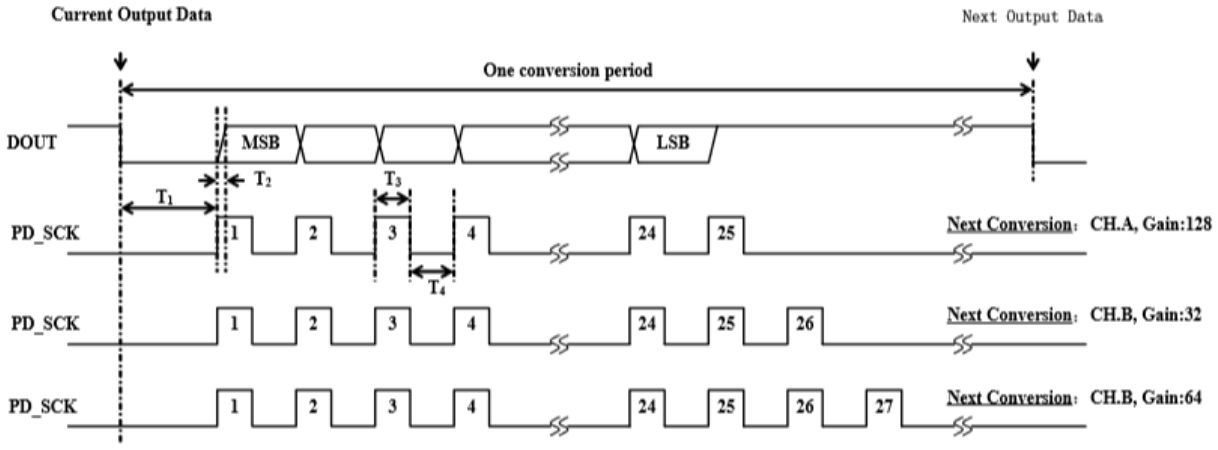


Figure 17 - Communication protocol for hx711, data output, input, and gain selection timing and control. [7]

To control more than one amplifier with the same clock the Arduino must wait until all of them are ready to send data, and then start the process reading data from each one of the amplifiers, with their DOUT pin connected to a different pin from the Arduino for each. This data is stored in an array that contains the read value for each amplifier, that's why the library has also to include functions that can process an array of values instead of just single data. Although the pin limitation of amplifiers that can be connected to the Arduino is not so strict, this way of connecting the amplifiers could also be limited by another factor. For every clock pulse, values for each load cell have to be read, and Arduino cannot do it simultaneously. But, as every clock pulse lasts 25 ms and the microprocessor can make the reading in approximately $5\ \mu s$, the limitation for this type of connection is still the number of pins available.

4.3 Arduino

To interface all amplifiers with the same clock a specific library for this purpose is used called HX711-multi. Not all the necessary functions are implemented in this library so programming some of them is necessary. Although these functions could be directly included at the code, for other users to be able to use them they are implemented at the library and a pull request is done at GitHub to update the library with these new functions.[1]

The program includes the following functions to extract data from the values obtained from the sensors. These functions process the data available of loads in grams from the array 'weights', which contains the output in grams for each one of the load cells.

4.3.1 Lift

As there are two load cells measuring in the *z* direction lift force is the sum of both readings from load cell 1 and load cell 2 (see Figure 9).

Arduino code to calculate lift:

```
F1 = weights[0]*go/1000; //[N]
F2 = weights[1]*go/1000; //[N]
lift = F1 + F2;
```

4.3.2 Drag

As there is only one load cell measuring in the parallel direction of the wind the drag measure will be the resultant force measured by this load cell. Arduino code to calculate drag:

```
drag = (weights[2])*go/1000; //[N]
```

[1] For library code see annex page 24

4.3.4 Moment at the leading edge

Pitching moment at the leading edge can be calculated with the values obtained from 1st and 2nd load cell and known data from the force balance and the wing model.

$x_{01} =$ Distance between leading edge and 1st load cell (depends on the wing model that is being tested)

$x_{12} = 0.041 =$ Distance between 1st load cell and 2nd load cell

$x_{02} = x_{01} + x_{12}$

$F_1 =$ force measured at 1st load cell

$F_2 =$ force measured at 2nd load cell

$$M_{le} = x_{01} \times F_1 + x_{02} \times F_2$$

Equation 1- Calculation of the momentum at the leading edge with balance readings.

Arduino code to calculate momentum at the leading edge:

```
Mle = x01 * F1 + x02 *F2; // [Nm]
```

4.3.5 Centre of pressure

Also with the readings from 1st and 2nd load cell and distance measures center of pressure is calculated.

$$M_{cp} = 0$$

$x_{cp} =$ Distance between leading edge and center of pressure

$x_{cp1} = x_{01} - x_{cp}$

$x_{cp2} = x_{02} - x_{cp}$

$$M_{cp} = x_{cp1} \times F_1 + x_{cp2} \times F_2 = 0$$

$$(x_{01} - x_{cp}) \times F_1 + (x_{02} - x_{cp}) \times F_2 = 0$$

$$x_{02} \times F_2 + x_{01} \times F_1 = x_{cp} \times (F_1 + F_2)$$

$$x_{cp} = \frac{x_{02} \times F_2 + x_{01} \times F_1}{F_1 + F_2}$$

Equation 2- Calculation of the center of pressure with balance readings.

Arduino code to calculate the center of pressure

```
xcp = (x02 * F2 + x01 * F2) / (F1 + F2)*1000;
```

4.3.6 Aerodynamic coefficients

When introducing data of the model and the conditions of the experiment in the program, aerodynamic coefficients can also be calculated. The following lines of code are used for this purpose. Arduino code to calculate aerodynamic coefficients are:

```
Cl = (lift*2)/(rho*v*v*S);
Cd = (drag*2)/(rho*v*v*S);
Cm = (Mle*2)/(rho*v*v*S*C);
```

5 Construction

This thesis includes the mounting and preparation of all the electronics and sensors and the design of the pieces missing to build the balance. Electronics is ready to be connected to a computer with the program and the wires are ready to be all connected to the Arduino following the scheme on Figure 16. Instructions on how to run the program are also included in User's manual, in page 48. Mounting of the system not only includes the construction of the pieces designed in SolidWorks, but also making a hole in the wind tunnel shell. The hole should be made when all the pieces are prepared and mounted to ensure that the whole system is working before making changes on the wind tunnel.

5.1 Electronics

The correct running of the electronics is tested and verified and two successful calibration tests are done, first one consisting on testing if the response of the load cell of 1 Kg is linear, and second one consisting on seeing if three load cells connected to the same clock at the same time can be calibrated successfully (to see how the balance is calibrated see chapter Calibration, page 30). It has to be noted that these tests are done holding the load cell with the hands and not with bolts, so an error up to 2-3% due to the possible movement can be accepted.

In test 1, first, the 1 kg load cell is calibrated with a weight of 21.2g. Then, when the weight is placed again at the load cell it shows 21.3g. A weight of 132.6 kg is placed and the load cell shows 133.5 kg. Using library for only one HX711. An error of 0, 67%, mostly due to inaccuracies created by the fact that the load cell was not tight to a surface shows that the load cell response is as expected. In test 2, load cells with their amplifiers are connected to Arduino following the scheme on Figure 16, and code with the updated library is uploaded. Calibration is done with known weights placed at each one of the load cells and the system is tested. Each of the load cells is calibrated successfully and the values can be read in grams.

5.2 Structure of the balance

For the construction of the balance pieces shown in Figure 10, Figure 11, Figure 18 and Figure 19 designed for the balance should be built in metal, preferably an aluminum alloy. Sketches of the pieces can be seen in Annex, page 48. The load cells already have the bolt system built, Figure 24, with the only need to be tighten and placed in the structure. Also, it is recommended that for making angle of attack adjustment easier, a protractor is placed at the back part of the testing section.

The base should be put on the existing structure and adjust it to the holes, as mentioned in Structure base (page 11). As the structure is not smooth, it is recommended to use bolts of a smaller diameter, to be able to adjust the piece at least two millimeters when building the balance, but without making the unions less strong, as they must not allow vibration. A rail under the base helps placing the piece in its position as it can be seen in Figure 20.

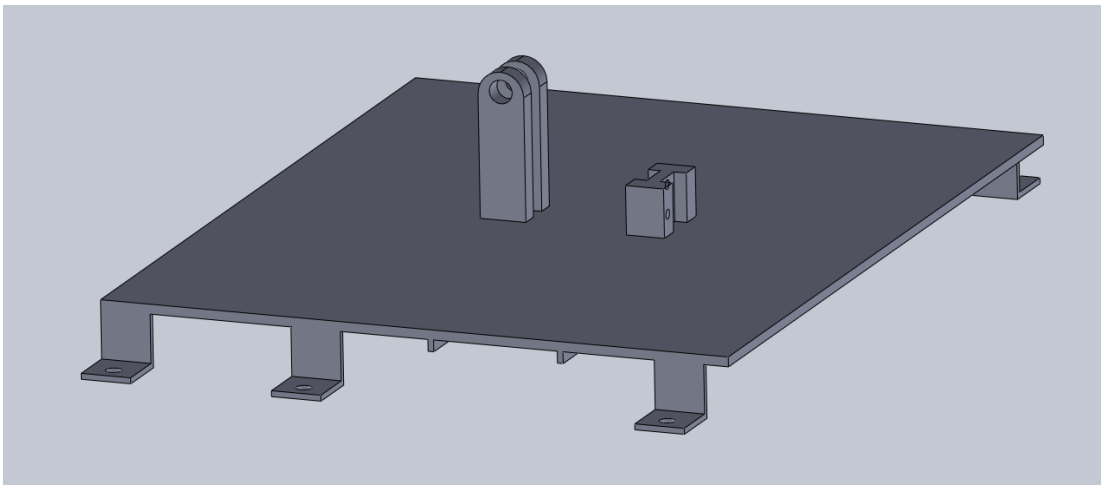


Figure 18- Base of the balance.

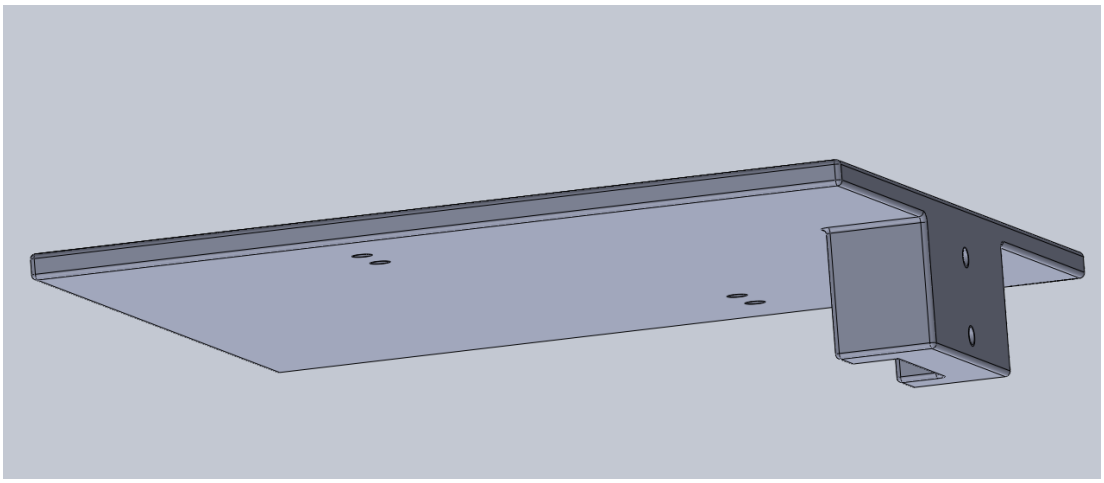


Figure 19- Sliding platform of the balance viewed from underneath.

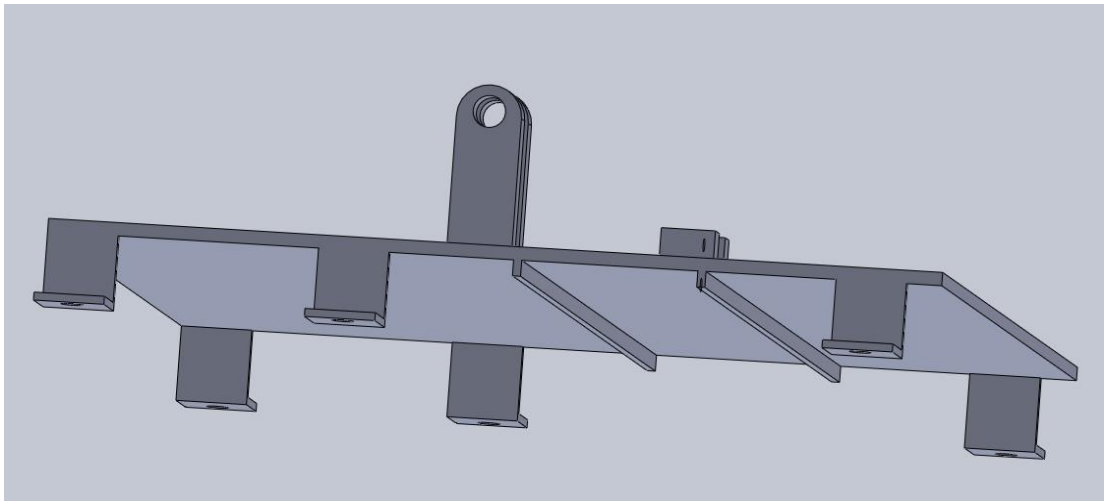


Figure 20- Base of the balance viewed from underneath.

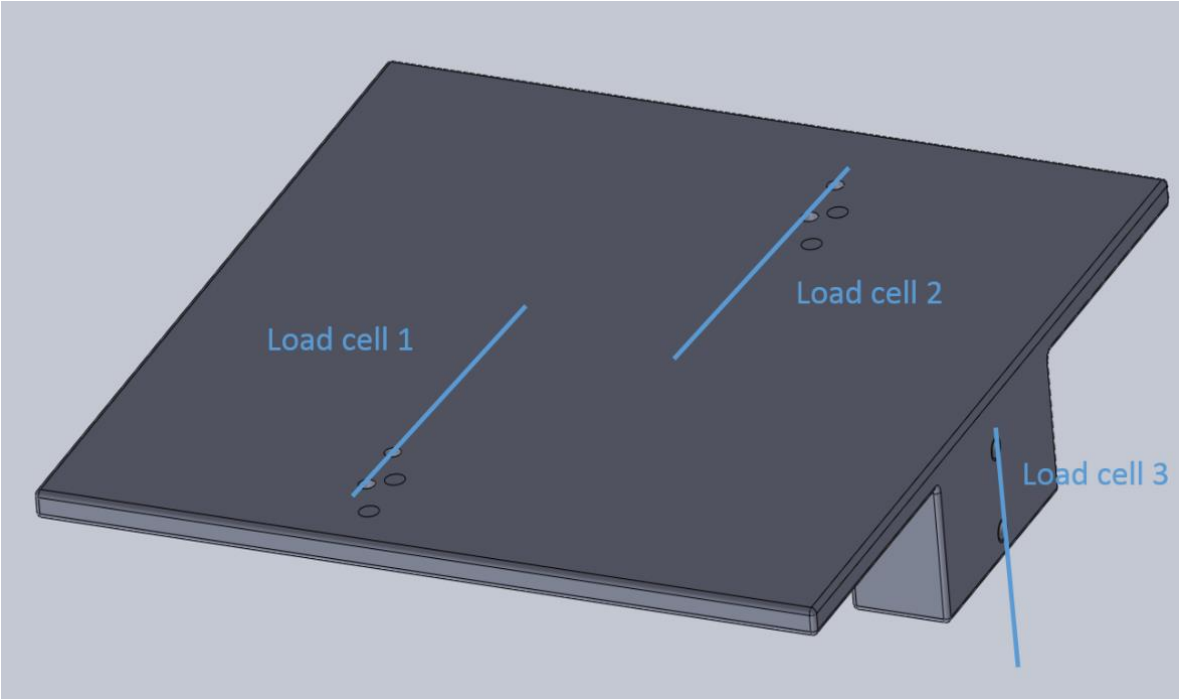


Figure 21- Upper side of the sliding platform with the position of the load cells.

When attaching the lift load cells (load cell 1 and load cell 2) to the moving platform, it is important to leave a certain distance between the base and the load cell. Bolt system shown in Figure 22 is recommended, in this figure only the system to make it stay in a height can be seen. Nuts should be tight to each other in order to create a strong union that remains in the same place. It is important that the nuts are very tight to avoid any kind of vibration. Another two nuts should be put, one under the base and the other one tied to the load cell (see Figure 24).

Figure 22- Bolt system to leave a distance between the load cell and the base.

The drag load cell should be placed vertically in the side holes (as shown in Figure 21) between the moving platform and the base. Bolts used to tighten this load cell are different size (DIN M4 for the upper part and DIN M5 for the lower part), so it can easily be seen where they have to be attached.

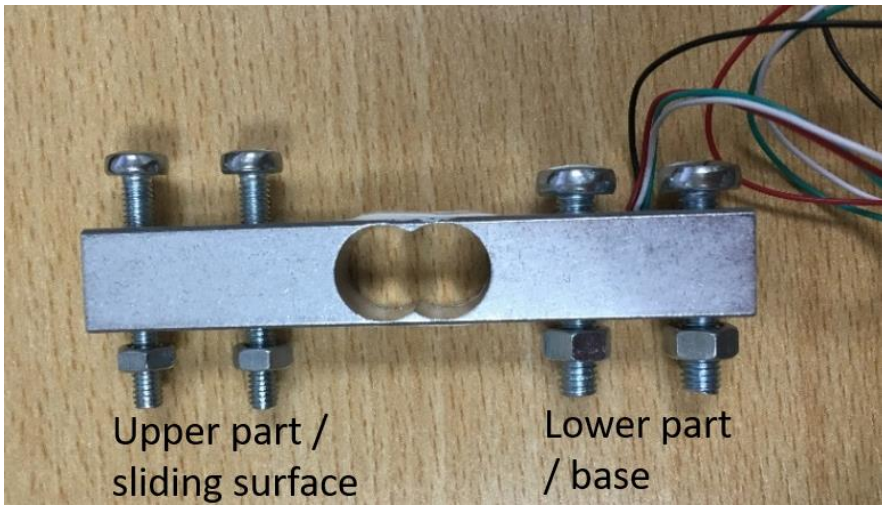


Figure 23- Drag load cell with its bolts.

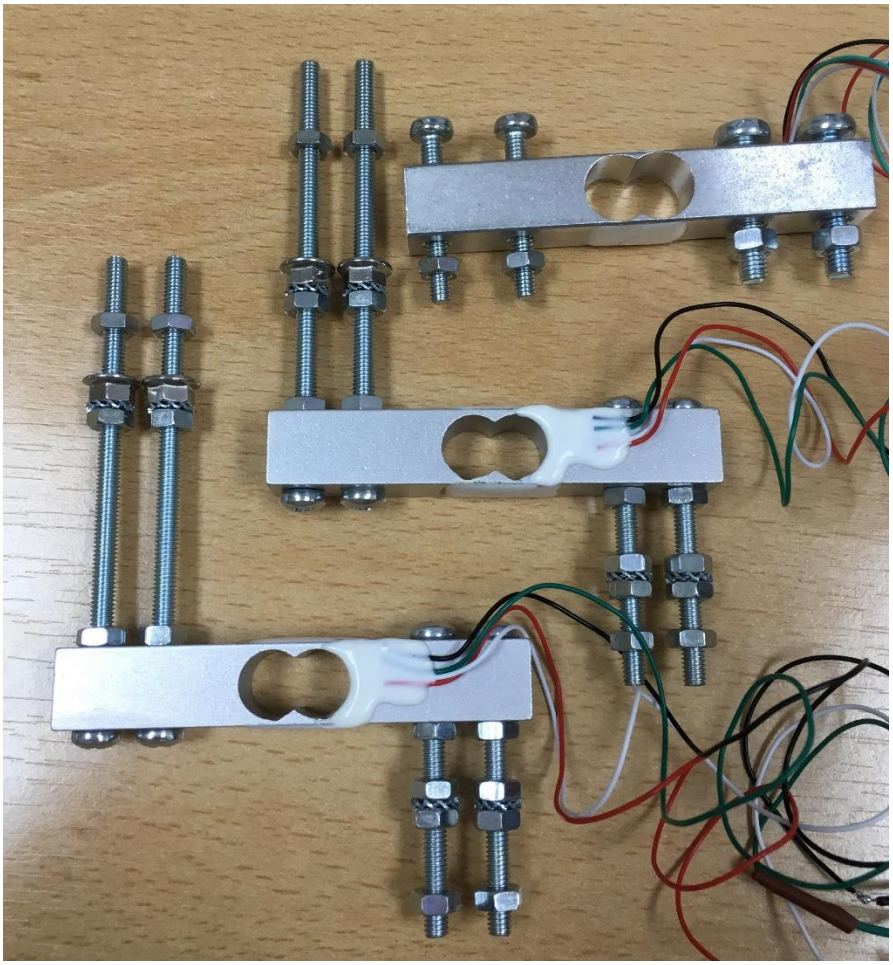


Figure 24- The three load cells of the balance with all the bolts and nuts that are needed for construction. From left to right, load cells 1 and 2 and load cell 3.

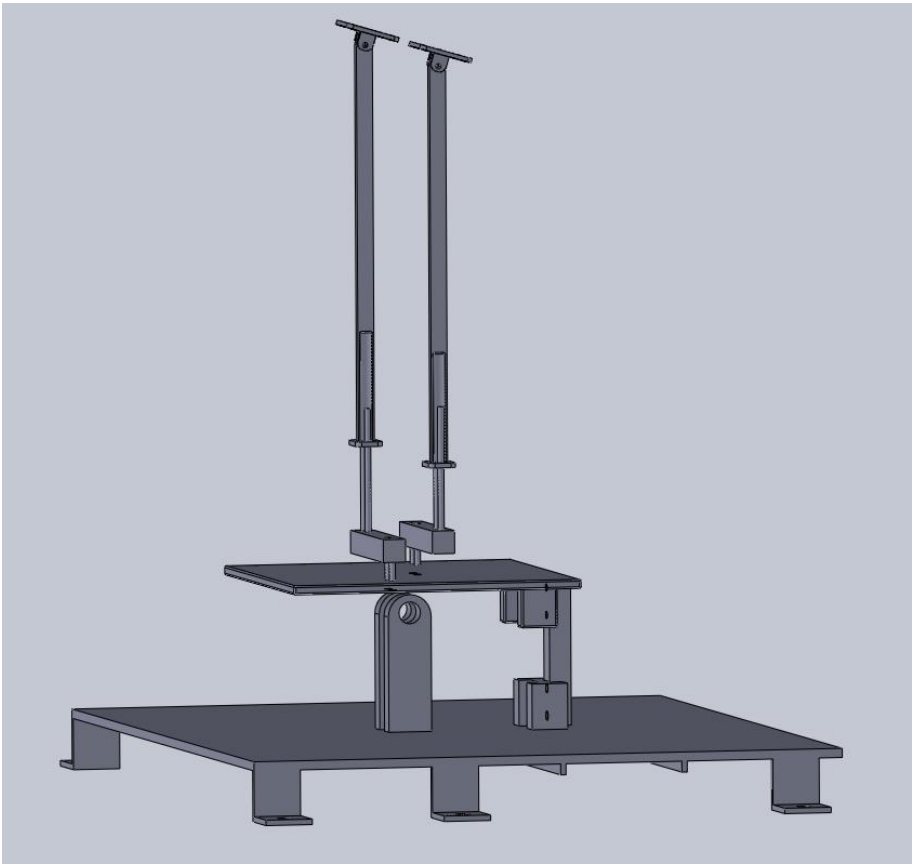


Figure 25- Side view of the force balance structure.

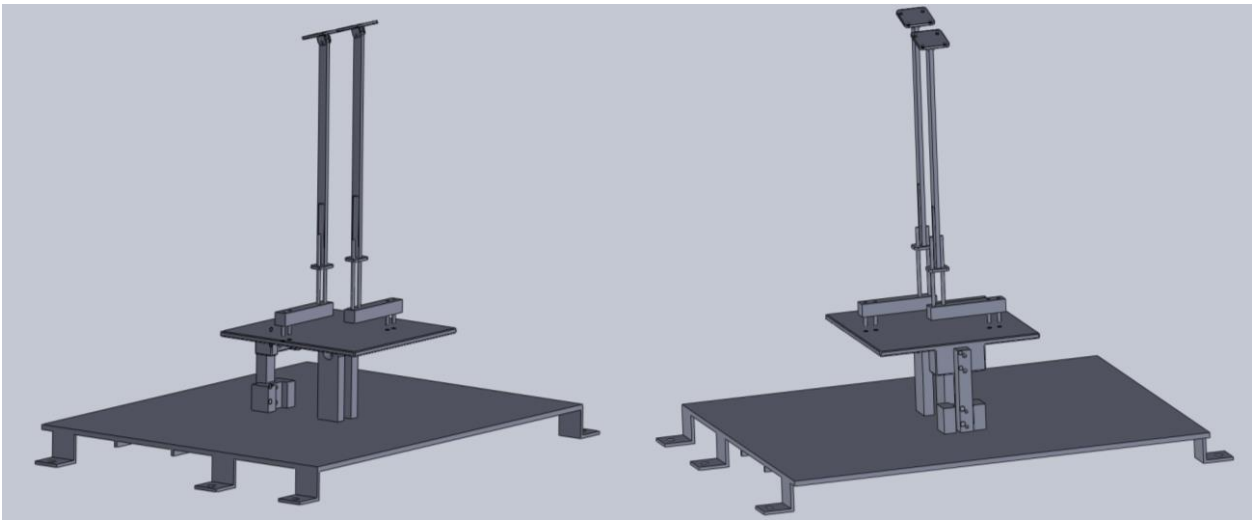


Figure 26- Structure of the balance from the back side (left) and the rear (right).

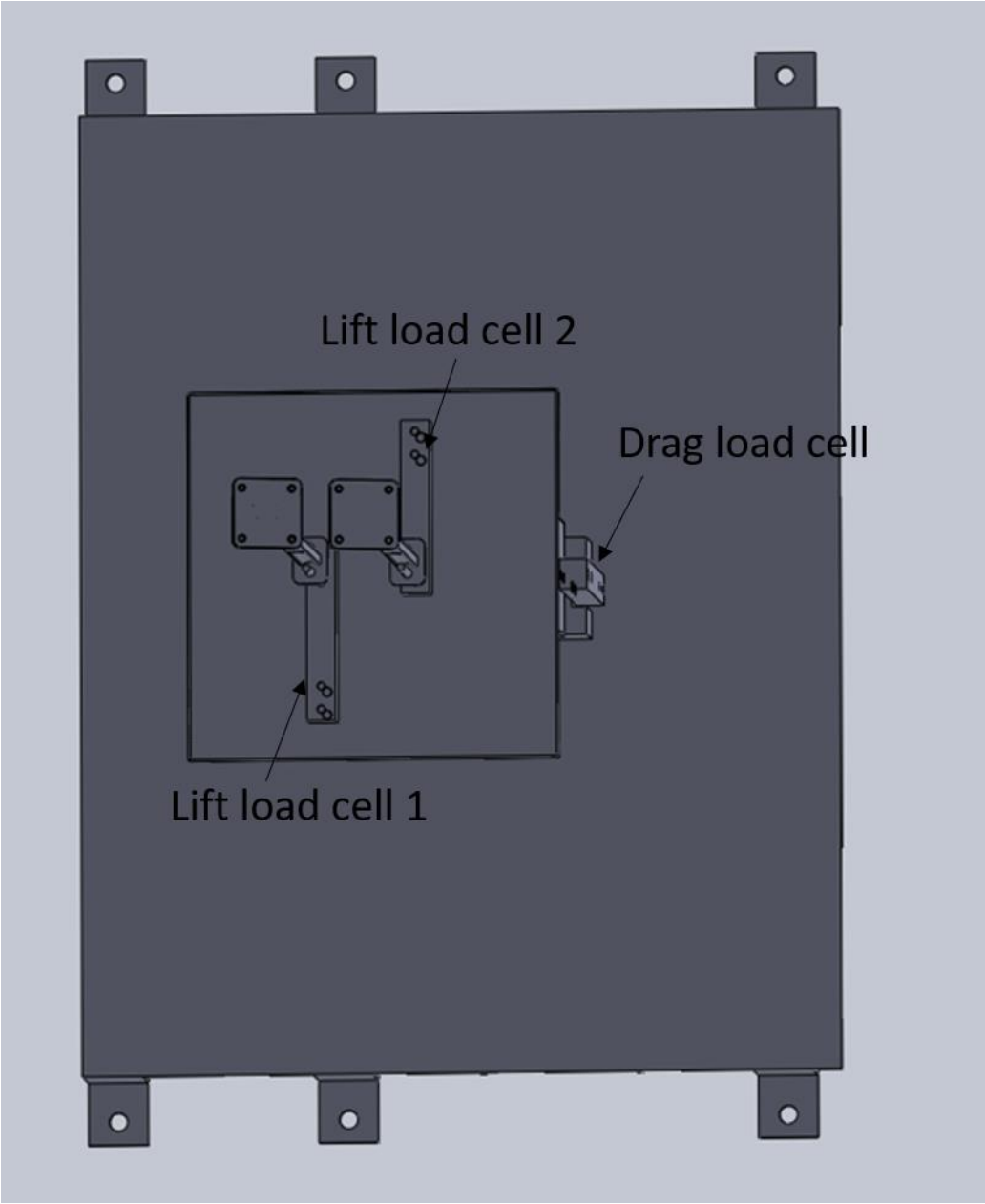


Figure 27- Top view of the structure.

6 Calibration

The balance is designed to be used to test different wing models and be able to adapt to the conditions of the experiment. As the use of the balance throughout the time can affect the output of the sensors and, when measuring, the model is placed on them, it has to be ensured that none of these facts make the measurements of the balance unreliable. The procedure for adjusting the balance for each measurement is designed to be as easy as possible for users. In this procedure balance tare and balance calibration should be distinguished, as tare is the procedure to adapt the balance to the different weight configuration of the model and should be done every time a measurement is done, and calibration is done to adjust the balance to possible new conditions, like state of the load cells, load cell replacement, changes in electronics or environmental conditions like temperature. Also, it is important for the maintenance of the balance not to leave models or heavy parts on the balance for a long time to avoid creep in the load cells.

Tare of the balance is a function included in the code that automatically sets the balance to 0 adjusting the values of the output. This function is included in the library and is programed specifically for a balance with more than one sensor connected to the same clock. The functioning of tare is similar to the one of a kitchen balance, when a weight is placed on the balance the program saves the value of the weight when tare is requested and subtracts the value from the output. This is programed to be done for each of the load cells when any character is sent to the Arduino through the serial monitor and each time the program is uploaded, or the Arduino is reset. In this way, models with different weight distribution can be tested without doing changes in the configuration of the balance or the program. Arduino code calling the tare function:

```
// On serial data (any data), re-tare
if (Serial.available() > 0) {
  while (Serial.available()) {
    Serial.read();
  }
  tare();
}
```

As it can be seen in the code, when a character is sent to the serial monitor, tare function is called (see annex page REF), with the parameters 20 measures and a tolerance in measurements of 10,000 units. This tolerance means that if the difference between two read values of this measures is higher than it, tare is considered unsuccessful and it is restarted until it is. The tare timeout is of 4 seconds.

Calibration of the balance has to be done manually. It is necessary to transform the output values of the amplifier into weight values measured by the load cells. It has to be done individually at each cell by placing a known weight on it and measuring the output value to calculate a conversion factor (see Equation 3) between output values and weight measurement. This can be done because the output of the sensor is linear. Then, once the conversion factor is calculated, it is introduced in an array of the program. This array is sent to the method ‘set scales’ of the library

(see page number 42) to be used as conversion factor. To account for the friction in the system, calibration will be done with the balance mounted. In the lift measuring load cells, a weight will be placed at the top of each bar and in the drag measuring load cell a weight with a pulley placed to make the weight pull horizontally should be used. Example of the array with the calibration values:

```
float measurements[N] = {(-10000/21.2), -8600/21.2, 40750/21.2}; // measured SCALES parameters
```

$$Conversion\ factor = \frac{reading}{known\ weight\ [g]}$$

Equation 3- Conversion factor to calibrate the balance.

7 Conclusions

A force balance able to measure lift, drag and pitching moment in a small low-speed wind tunnel has been designed, fulfilling the main objective of the thesis. The design presents all the necessary parts to build and mount the force balance, along with extensive tests of the electronics. Other goals of the thesis were to make the design as simple, inexpensive and understandable as possible. In this work, the decisions made to accomplish this objective are explained. The final configuration of the balance is an accessible design with a simple data acquisition method, and a manual of use is included to make it accessible for students.

This design consists on a three-sensor based force balance for measuring lift drag and pitching moment, two of them measuring lift and pitching moment and one to measure drag. It is based on a typical configuration used for two-axis force balances which is extended to measure the force in one more axis. The balance is designed to be used from -45 to 45 degrees of angle of attack (*see chapter* 3.3) and, as explained in point 4.1, to be able to be used in all the speed range of the wind tunnel.

Data acquisition with Arduino instead of other setups makes the design more adaptable to the needs of each experiment, while letting students see the code behind the results in the screen. It also avoids expensive laboratory hardware (there is no need for external data acquisition cards) and software (since Arduino is an open-source platform). The code can also be easily adapted if further changes are done to the balance. Evidently, obtaining data with this microcontroller has its drawbacks, as it is not as precise as other methods that might be more common in laboratories.

As the balance has not been built, testing of the whole system could not be performed. Partial testing has been done though, proving that the electronics, the sensors and the program work. Bolt adjustment has also been checked, in order to avoid imprecisions that could cause vibrations that could negatively affect the accuracy of the results. Even though the manufacturing of the balance is not complete, recommendations on how to maintain and adjust the balance have been done. Once the balance is finished, other recommendations can be added to improve its maintenance. Calibration and tare of the balance are also described, which can also amend some inaccuracy problems like the creep of the load cells with time. The balance should be tared every time it is used, and it has been programmed to automatically do so every time it is turned on. It can also be manually done very quickly and easily. Calibration should also be performed before new measurements are taken, but since it is only a 3-axis balance, it can be completed in a rather short time.

One of the main problems the balance can have is the measurement of drag, as there is no literature regarding fore balances with sliding surfaces, and the complete design has yet to be tested. Drag is also the smallest force measured at the balance and then the most sensible to inaccuracies. In comparison, before testing the balance, as long as there are no issues during its

construction, there are no reasons to suspect that there will be any problems when measuring lift or pitching moment, and this setup should provide correct and accurate results. This is expected, as it was already planned from the beginning that the balance would be more accurate when measuring lift than drag. For this very reason, pitching moment computation is done with lift results. It is also assumed that even though there is friction in the sliding wheel, the measured response will still be linear.

To be able to use the balance, work done in this thesis should be continued. The pieces designed should be built and the system should be mounted, including electronics, as specified in this paper. When the balance is built, it should be calibrated and tested. Changes in the program can be done to include further calculations. After building the balance, the work can be also continued including other features, (i.e. a user interface, a screen to read the measurements connected to the microprocessor) or complementing the readings of the balance with other sensors that can be connected to the Arduino.

8 Budget and list of materials

The budget contains the list and price of the materials purchased at the moment of the thesis defense.

8.1 Electronics

Table 3 - List of prices of the electronics used for the project.

Name	Amount	Price (HRK)
HX711 Module	3	87,00
1 kg load cell	1	66,00
Arduino Uno	1	245,00
Arduino wire set	1	47,00
Pin /headers	1	2,30
USB cable type A to type B	1	29,00
5 kg load cell	2	120,00
Total		596,30

8.2 Structure materials

Table 4 - List of prices of the structure materials purchased at the moment of the defense.

Name	Amount	Price (HRK)
DIN 7985 M 4x80 Zn	4	6,08
DIN 6923 M 4 A2	4	5,48
DIN 934 M 4 Zn	27	2,16
DIN 6798 M 4 Zn	8	1,60
DIN 7985 M 4X45 Zn	4	2,24
DIN 7985 M 5x25 Zn	2	1,10
DIN 7985 M 4x25 Zn	2	0,76
DIN 934 M 5 Zn	2	0,22
Total		19,64

9 Bibliography

[1] J. B. ,. R. W. &. P. A. Barlow, Low-speed wind tunnel testing., New York: John Wiley & Sons, 1999.

[2] S. Illera, "Diseño de una balanza de viento para túnel aerodinámico.," Universitat Politècnica de Catalunya, Terrassa, 2007.

[3] G. Barden, "Development of a wind tunnel force balance and related practical exercise manual.," Charles Darwin University, Darwin, 2014.

[4] A. Buljac, "Master's thesis," University of Zagreb, Zagreb, 2014.

[5] S. &. M. M. Post, "Force balance design for educational wind tunnels.," AC 2010-393., 2010.

[6] S. Janković, "Aerodynamic Coefficients Estimation for Dash 8 Q400 in Flight," Faculty of Transport and Traffic Sciences, Zagreb, 2017.

[7] *24-Bit Analog-To-Digital Converter (ADC) for Weigh Scales HX711.,* Avia Semiconductor.

10 List of tables

11 List of figures

ANNEX

Component specifications

Load cells

Table 5- Load cell specifications.

Application	Kitchen scale, pricing scale, force test equipment
Model	SC133
Capacity	1Kg, 2kg, 3kg, 5kg, 10kg, 20kg
Rated output	1.0±0.15mV/V
Nonlinearity	0.05
Repeatability	0.03
Hyteresis	0.03
Accuracy	0.05 %
Creep (5 min)	0.1
Temp. effect on sensitivity	0.003 %RO/℃
Temp.effect on zero	0.02%RO/℃
Zero balance	±1.0 %RO
Input resistance	1066±20 ohm
Output resistance	1000±20 ohm
Insulation resistance	2000 MΩ(50V)
Recommended excitation voltage	5V
Compensated temperature range	-10-+50℃
Operating temperature range	-20+65 ℃
Safe overload	120 %RO
Ultimate overload	150 %RO
Load cell material	Aluminum alloy
Connecting cable	Φ0.8x180m
Method of connecting wire	red/black/green/white

Ball bearing

Table 6- Ball bearing measures for the proposed base of the balance.

	Measure (mm)
D	32
d	16
B	10

HX711 Amplifier

Table 7- HX711 Amplifier Specifications.

Parameter	Notes	MIN	TYP	MAX	UNIT
Full scale differential input range	V(inp)-V(inn)		±0.5(AVDD/GAIN)		V
Common mode input		AGND+1.2		AVDD-1.3	V
Output data rate	Internal Oscillator, RATE = 0		10		Hz
	Internal Oscillator, RATE = DVDD		80		
	Crystal or external clock, RATE = 0		f_{clk}/1,105,920		
	Crystal or external clock, RATE = DVDD		f_{clk}/138,240		
Output data coding	2's complement	800000		7FFFFF	HEX
Output settling time (1)	RATE = 0		400		ms
	RATE = DVDD		50		
Input offset drift	Gain = 128		0.2		mV
	Gain = 64		0.4		
Input noise	Gain = 128， RATE = 0		50		nV(rms)
	Gain = 128， RATE = DVDD		90		
Temperature drift	Input offset（Gain = 128）		±6		nV/℃
	Gain（Gain = 128）		±5		ppm/℃
Input common mode rejection	Gain = 128， RATE = 0		100		dB
Power supply rejection	Gain = 128， RATE = 0		100		dB
Reference bypass （V_{BG}）			1.25		V
Crystal or external clock frequency		1	11.0592	20	MHz
Power supply voltage	DVDD	2.6		5.5	V
	AVDD，VSUP	2.6		5.5	
Analog supply current (including regulator)	Normal		1400		μA
	Power down		0.3		
Digital supply current	Normal		100		μA
	Power down		0.2		

Code

The library is already implemented and necessary for running the program while some modifications can be done in the Arduino program while using it.

Arduino library

This section contains the two main files of the library (.h and .cpp). The contributions made to the library for this thesis are highlighted in green, while the comments explaining the rationale behind these changes are highlighted in yellow.

HX711-multi.h

```
#ifndef HX711_MULTI_h

#define HX711_MULTI_h
#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
class HX711MULTI {
private:
byte PD_SCK;   // Power Down and Serial Clock Input Pin
byte COUNT;           // The number of channels to read
byte *DOUT;           // Serial Data Output Pin
byte GAIN;            // Amplification Factor
bool debugEnabled; // print debug messages?
long *OFFSETS;        // used for tare weight

// Array of values used to convert the integer digital data to weight (in grams, ounces,
etc.)
float *SCALES;
public:
// define clock and data pin, channel, and gain factor
// channel selection is made by passing the appropriate gain: 128 or 64 for channel A, 32
for channel B
// count: the number of channels
// dout: an array of pin numbers, of length 'count', one entry per channel
HX711MULTI(int count, byte *dout, byte pd_sck, byte gain = 128);
virtual ~HX711MULTI();
// returns the number of channels
byte get_count();
// check if HX711 is ready
// from the datasheet: When output data is not ready for retrieval, digital output pin
DOUT is high. Serial clock
// input PD_SCK should be low. When DOUT goes to low, it indicates data is ready for
retrieval.
bool is_ready();
// set the gain factor; takes effect only after a call to read()
// channel A can be set for a 128 or 64 gain; channel B has a fixed 32 gain
// depending on the parameter, the channel is also set to either A or B
void set_gain(byte gain = 128);
// waits for the chip to be ready and returns a reading
void read(long *result = NULL);
// same as read, but does not offset the values according to the tare
void readRaw(long *result = NULL);
// set the OFFSET value for tare weight
// times: how many times to read the tare value
```

```
    // returns true iff the offsets have been reset for the scale during this call.
    // tolerance: the maximum deviation of samples, above which to reject the attempt to tare.
    (if set to 0, ignored)
    bool tare(byte times = 10, uint16_t tolerance = 0);
    // puts the chip into power down mode
    void power_down();
    // wakes up the chip after power down mode
    void power_up();
    void setDebugEnable(bool debugEnable = true);
    /* NEW FEATURES BELOW */

    // Method to return the array of SCALES values
    float* get_scales();

    // Method to set the array of SCALES values
    void set_scales(float *_scales);

    // Method to actually get the weight data (does not return it, but changes the parameter
    passed by reference)
    void get_units(double *result = NULL);

    /* END OF NEW FEATURES */
    };

    #endif /* end HX711_MULTI_h */
```

HX711-multi.cpp

```
    #include <Arduino.h>
    #include <HX711-multi.h>
    HX711MULTI::HX711MULTI(int count, byte *dout, byte pd_sck, byte gain) {
            PD_SCK  = pd_sck;
            DOUT    = dout;
            COUNT   = count;
            debugEnabled = false;
            pinMode(PD_SCK, OUTPUT);
            for (int i=0; i<count; i++) {
                    pinMode(DOUT[i], INPUT);
            }
            set_gain(gain);
            OFFSETS = (long *) malloc(COUNT*sizeof(long));

            // Allocate memory space for the SCALES array when creating a new HX711MULTI object
            SCALES = (float *) malloc(COUNT*sizeof(float));
            for (int i=0; i<COUNT; ++i) {
                    OFFSETS[i] = 0;

                    // Initialize the SCALES array with 1's to avoid dividing by 0
                    SCALES[i] = 1.0;
            }
    }
    HX711MULTI::~HX711MULTI() {
            free(OFFSETS);

            // Recover the memory space allocated earlier when deleting the object
            free(SCALES);
    }
    bool HX711MULTI::is_ready() {
            bool result = true;
```

```
	for (int i = 0; i<COUNT; ++i) {
		if (digitalRead(DOUT[i]) == HIGH) {
			result = false;
		}
	}
	return result;
}
void HX711MULTI::set_gain(byte gain) {
	switch (gain) {
		case 128:		// channel A, gain factor 128
			GAIN = 1;
			break;
		case 64:		// channel A, gain factor 64
			GAIN = 3;
			break;
		case 32:		// channel B, gain factor 32
			GAIN = 2;
			break;
	}
	digitalWrite(PD_SCK, LOW);
	read(); // a read is needed to get gain setting to come into effect.
}

byte HX711MULTI::get_count() {
	return COUNT;
}
bool HX711MULTI::tare(byte times, uint16_t tolerance) {
	int i,j;
	long values[COUNT];
	long minValues[COUNT];
	long maxValues[COUNT];
	for (i=0; i<COUNT; ++i) {
		minValues[i]=0x7FFFFFFF;
		maxValues[i]=0x80000000;
	}

	for (i=0; i<times; ++i) {
		readRaw(values);
		for (j=0; j<COUNT; ++j) {
			if (values[j]<minValues[j]) {
				minValues[j]=values[j];
			}
			if (values[j]>maxValues[j]) {
				maxValues[j]=values[j];
			}
		}
	}
	if (tolerance!=0 && times>1) {
		for (i=0; i<COUNT; ++i) {
			if (abs(maxValues[i]-minValues[i])>tolerance) {
				// one of the cells fluctuated more than the allowed
				tolerance, reject tare attempt;
				if (debugEnabled) {
					Serial.print("Rejecting tare: (");
					Serial.print(i);
					Serial.print(") ");
					Serial.println(abs(maxValues[i]-
					minValues[i]));
				}
				return false;
			}
```

```
                }
        }
        // set the offsets
        for (i=0; i<COUNT; ++i) {
                OFFSETS[i] = values[i];
        }
        return true;
}
// reads from all cahnnels and sets the values into the passed long array pointer (which
must have at least 'count' cells allocated)
// if you are only reading to toggle the line, and not to get values (such as in the case
of setting gains) you can pass NULL.
void HX711MULTI::read(long *result) {
        readRaw(result);

        // Datasheet indicates the value is returned as a two's complement value, so
        'stretch' the 24th bit to fit into 32 bits.
        if (NULL!=result) {
                for (int j = 0; j < COUNT; ++j) {
                    result[j] -= OFFSETS[j];
                }
        }
}
void HX711MULTI::readRaw(long *result) {
        int i,j;

        // wait for all the chips to become ready
        while (!is_ready());
        // pulse the clock pin 24 times to read the data
        for (i = 0; i < 24; ++i) {
                digitalWrite(PD_SCK, HIGH);
                if (NULL!=result) {
                        for (j = 0; j < COUNT; ++j) {
                                bitWrite(result[j], 23-i, digitalRead(DOUT[j]));
                        }
                }
                digitalWrite(PD_SCK, LOW);
        }

        // set the channel and the gain factor for the next reading using the clock pin
        for (i = 0; i < GAIN; ++i) {
                digitalWrite(PD_SCK, HIGH);
                digitalWrite(PD_SCK, LOW);
        }


        // Datasheet indicates the value is returned as a two's complement value, so
        'stretch' the 24th bit to fit into 32 bits.
        if (NULL!=result) {
            for (j = 0; j < COUNT; ++j) {
                        if ( ( result[j] & 0x00800000 ) ) {
                                result[j] |= 0xFF000000;
                        } else {
                        result[j] &= 0x00FFFFFF; // required in lieu of re-setting the value
                        to zero before shifting bits in.
                        }
            }
         }
}
void HX711MULTI::setDebugEnable(bool debugEnable) {
        debugEnabled = debugEnable;
```

```
}
void HX711MULTI::power_down() {
	digitalWrite(PD_SCK, LOW);
	digitalWrite(PD_SCK, HIGH);
}
void HX711MULTI::power_up() {
	digitalWrite(PD_SCK, LOW);
}

float* HX711MULTI::get_scales() {
	// Just return the SCALES array (property of the object)
	return SCALES;
}


void HX711MULTI::set_scales(float *_scales) {
	// Just set the SCALES object property with the array parameter
	SCALES = _scales;
}


void HX711MULTI::get_units(double *result) {
	// Use the library's built-in read() method to obtain the weight as an integer
	value and save it in a temporary array
	long int rawResult[COUNT];
	read(rawResult);

	// Convert each value of the temporary array to weight data using the SCALES
	property and save it to the result array passed by reference
	for (int i = 0; i < COUNT; ++i) {
		result[i] = ((double) rawResult[i]) / SCALES[i];
	}
}
```

Force balance program code

This section contains the code used to extract aerodynamic information of the sensors at the designed configuration.

arduino_get_data_def.ino

```
#include "HX711-multi.h"

#define CLK A0 // clock pin to the load cell

#define N 3 // # of channels
#define DOUT1 A1 // data pin to the 1st load cell lift 1
#define DOUT2 A2 // data pin to the 2nd load cell lift 2
#define DOUT3 A3 // data pin to the 3rd load cell drag

#define TARE_TIMEOUT_SECONDS 4


// INSERT THE VALUES CORRESPONDING TO YOUR MODEL
```

```
// Inputs
double x01 = 0.02; // distance between the leading edge and the first load cell
[m]
double v = 10; // speed of the wind [m/s]
double rho = 1.225; // density [kg/m^3]
double S =  0.04; // wing surface [m^2]
double C = 0.2; // chord length [m]



//CALLIBRATION OF THE BALANCE

// To re-calibrate the balance (if a load cell is replaced or one of the existing
changes its behavior) write a 1.0 in the corresponding cell of measurements[]
// vector and tare the balance (this can be done sending any character to the
Arduino through Serial Monitor).
// Then read the value of 'WEIGHT DATA BELOW' corresponding to the load cell at
the program when a known weight is placed at the load cell.
// Substitute the 1.0 by  (value obtained)/(known weight [g]).

float measurements[N] = {(-10000/21.2), -8600/21.2, 40750/21.2}; // measured
SCALES parameters




// MAIN CODE, DO NOT SAVE THE CHANGES MADE HERE

// Init
byte DOUTS[N] = {DOUT1, DOUT2, DOUT3};
long int results[N];
double weights[N];
double go = 9.80662;
double lift;
double drag;
double Mle;
double F1;
double F2;
double x02 = 0.041;
double x12 = x01 + x02;
double xcp;
double Cl;
double Cd;
double Cm;

HX711MULTI scales(N, DOUTS, CLK);

void setup() {
    Serial.begin(115200);
    Serial.flush();

    tare();
    scales.set_scales(measurements);
}

void tare() {
```

```
    bool tareSuccessful = false;

    unsigned long tareStartTime = millis();
    while (!tareSuccessful && millis() < (tareStartTime + TARE_TIMEOUT_SECONDS
* 1000)) {
        tareSuccessful = scales.tare(20, 10000); //reject 'tare' if still
ringing
    }
}

// void sendRawData() {
//    scales.read(results);
//    for (int i = 0; i < scales.get_count(); ++i) {
//        Serial.print(-results[i]);
//        Serial.print((i != scales.get_count() - 1) ? "\t" : "\n");
//    }
//    delay(1000);
//}

void sendWeightData() {
    scales.get_units(weights);
    for (int i = 0; i < scales.get_count(); ++i) {
        Serial.print(-weights[i]);
        Serial.print((i != scales.get_count() - 1) ? "\t" : "\n");
    }
    delay(1000);
}

void loop() {
//    Serial.println("RAW DATA BELOW");
//    sendRawData(); // this is for sending raw data, for where everything else
is done in processing
//    Serial.println();

    Serial.println("WEIGHT DATA BELOW");
    sendWeightData();
    Serial.println();


    //calculate forces in N

F1 = weights[0]*go/1000; //[N]
F2 = weights[1]*go/1000; //[N]
lift = F1 + F2;
drag = (weights[2])*go/1000; //[N]
Mle = x01 * F1 + x02 *F2; // [Nm]
xcp = (x02 * F2 + x01 * F2) / (F1 + F2)*1000;

   //aerodynamic coefficients

Cl = (lift*2)/(rho*v*v*S);
Cd = (drag*2)/(rho*v*v*S);
Cm = (Mle*2)/(rho*v*v*S*C);

Serial.print("lift = ");
Serial.print(lift);
Serial.println(" [N]");
```

```
Serial.print("drag = ");
Serial.print(drag);
Serial.println(" [N]");  // REPRESENT IT IN MILINEWTONS FOR MORE PRECISSION
Serial.print("momentum at the leading edge = ");
Serial.print(Mle);
Serial.println(" [Nm]");
Serial.print("position of the centre of pressure: ");
Serial.print(xcp);
Serial.println(" [mm]");
Serial.println();
Serial.println("Aerodynamic coefficients");
Serial.print("Cl = ");
Serial.println(Cl);
Serial.print("Cd = ");
Serial.println(Cd);
Serial.print("Cm = ");
Serial.println(Cm);
Serial.println();


  // On serial data (any data), re-tare
  if (Serial.available() > 0) {
    while (Serial.available()) {
      Serial.read();
    }
    tare();
  }

}
```

User’s manual

Contents

How to use the balance

Before using the balance there are some basic recommendations to follow to ensure the maintenance of the system.

- To avoid creep it is important NOT to leave the model on the balance after testing, as it would decrease the precision of the measurements with time.
- The weight of the model should NOT be higher than 5 kg.
- Make sure the balance remains horizontal and no loads are continuously applied to the drag load cell.
- Be careful with cable connections and soldering, and if they disconnect follow the connection diagram at “useful information”.

And one last tip for a very common issue.

- If the serial monitor is not showing values after waiting more than one minute or is showing strange characters, check the wiring, that the program is uploaded and the Arduino is connected to the correct USB port and that the ‘baud’ selection at the monitor is 115200 or equal to the number inside Serial.begin(115200) at void setup().

Model

The model should be as light as possible to avoid non-linearities that may occur mostly in the near to 100% load zone. It should also be able to tie to the balance correctly and tightly to avoid vibrations.

Model building

- Max 5 kg.
- The measures of the piece to attach are at the end of the document, ‘Useful information’.

To test the model at the balance, it has to be attachable to the piece at the top.

Specify distances at where the holes should be done at the model size how many etc

Testing

1. Attach the top piece tightly to the model and place he model between the two top parts.
2. Introduce the bar in the holes and leave it loose but inside.
3. Adjust the angle of attack with the height regulation system.
   3.1. Untie the nut with flat surface and move it up or down to the desired position. DO NOT TOUCH THE NUT RIGHT NEXT TO THE LOAD CELL.
   3.2. Make sure both flat nuts are at the same height for each lad cell, otherwise the bar will be twisted.
   3.3. Tie the nut with the rough piece to the flat nut tightly so it does not move and then the nut on the top of the large bar base.
   3.4. Make sure the unions are strong so that the bars do not move.
4. Tie the bar at the top and make sure it is not susceptible to vibrations.

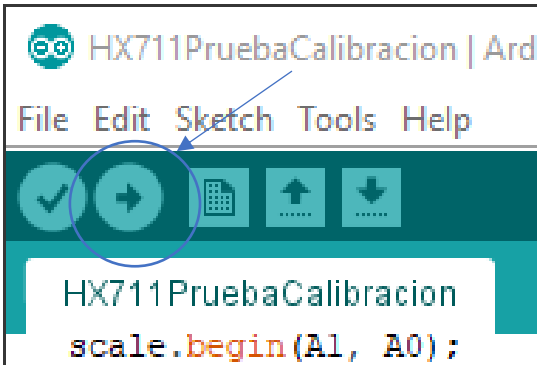
Be careful not to put too much pressure on the load cells while adjusting the angle of attack.

Recommendations

To make the attachment easier, glue a M3 nut to the model in each of the holes. Modifications can be made in the SolidWorks model to change the M3 nut for something else.

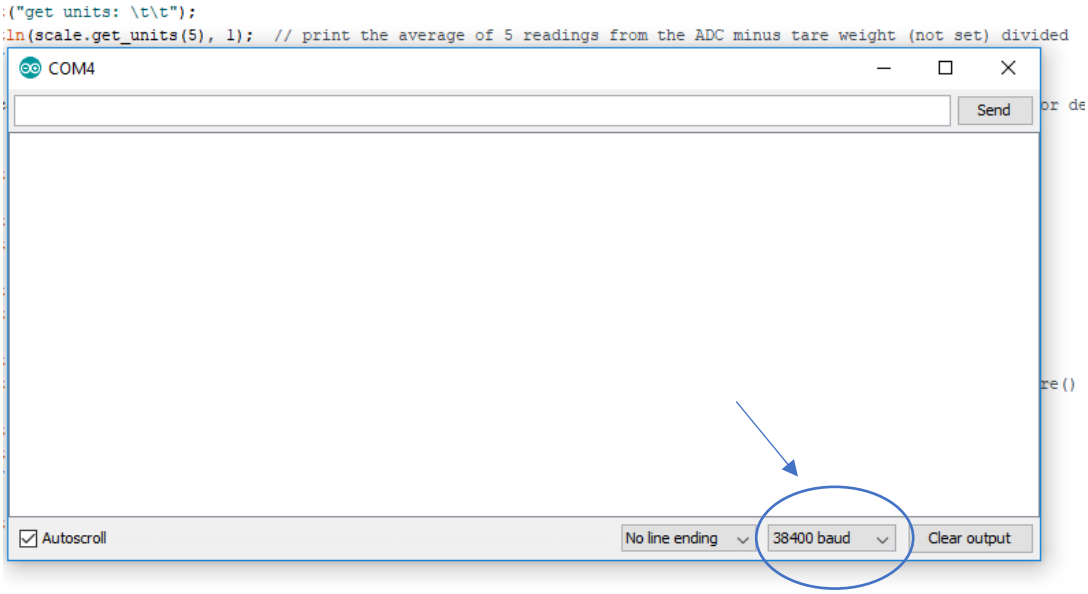
Arduino

1. Open the program *arduino_get_data_def.ino*
2. Connect the Arduino to the computer and open “Tools” and select Board- Arduino/Genuino Uno and Port- COM“X” being X the number of the port where the Arduino is connected.
3. Upload the program to the Arduino



Button to upload the program to the Arduino.

4. Set up the monitor to read the values at (control + shift + M), at 115200 baud.



Baud options.

5. Press the reset button at the Arduino board.
6. The monitor will show the forces and aerodynamic coefficients and their units.
7. If TARE is needed because weight is added or any other reason send any character trough the serial monitor.

Maintenance

It is impossible to maintain the balance in the same conditions through time, so the system should be properly calibrated before use.

Calibration of the system

- While TARE sets the balance to (0,0,0), calibration changes the slope of the linear response that the load cells give when weights are placed.

Save the values at the program before starting calibration process to restore it if something is not working after re-calibration.

1. Remove all models and things that can interfere with calibration process and that can make load cells interfere with each other.
2. Turn the Arduino on and upload the program with the vector 'float measurements[N] = {1, 1, 1}; // measured SCALES parameters' set to all 1s.
3. Tare the program to set the balance to 0,0,0. Don't worry if numbers are not exactly close to 0, they should be considerably lower than the ones when a force is applied to the load cell.
4. Look at the values from 'Weight values' from the serial monitor.
5. When placing a known weight (grams) at a load cell the corresponding value should change. First value is for 1st load cell (lift1) 2nd for 2nd (lift2) and 3rd for 3rd (drag).
6. To include the friction of the system in the calibration a pulley to simulate drag can be used.
7. Annotate the value when the known weight is placed and substitute the 1 at the corresponding cell at the vector 'float measurements' for the following conversion factor. 'Reading' is the value that appears at the corresponding cell of 'weight values'

   Conversion factor = reading/ known weight

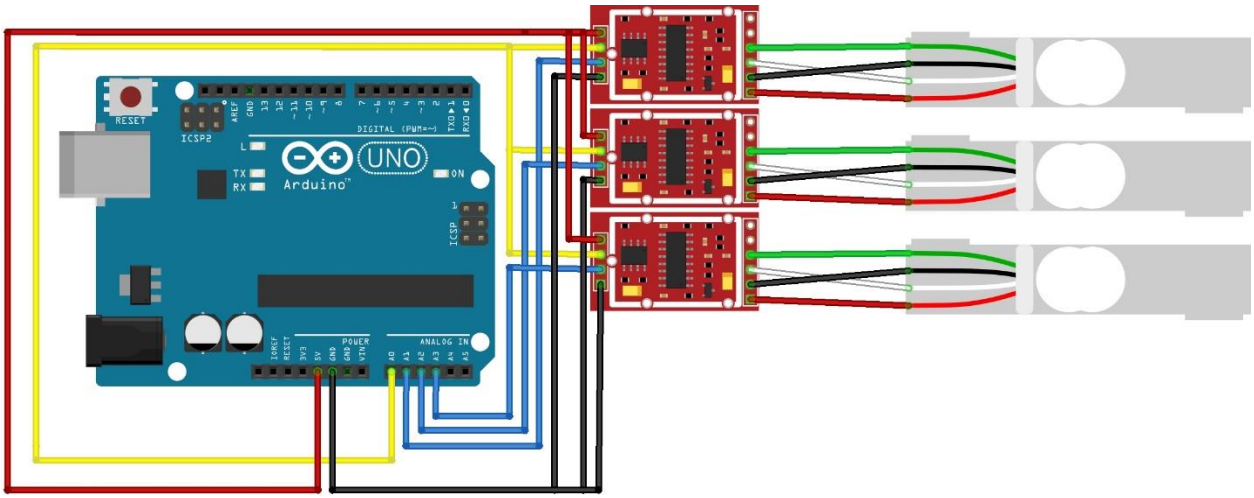
8. If negative numbers appear, don't worry, just put a '-' in the corresponding place of the vector, it means that the load cell is placed upside down.

Calculation of precision

Due to creep and diverse external factors, it is highly probable that load cell measurement precision varies with time until it stabilizes. If the precision of the balance is needed after re-calibration of the system, different known weights can be placed in the load cells and calculate how much the output varies from the real weight. Also calibration with a similar weight to the expected measuring is recommended to get more accurate results.

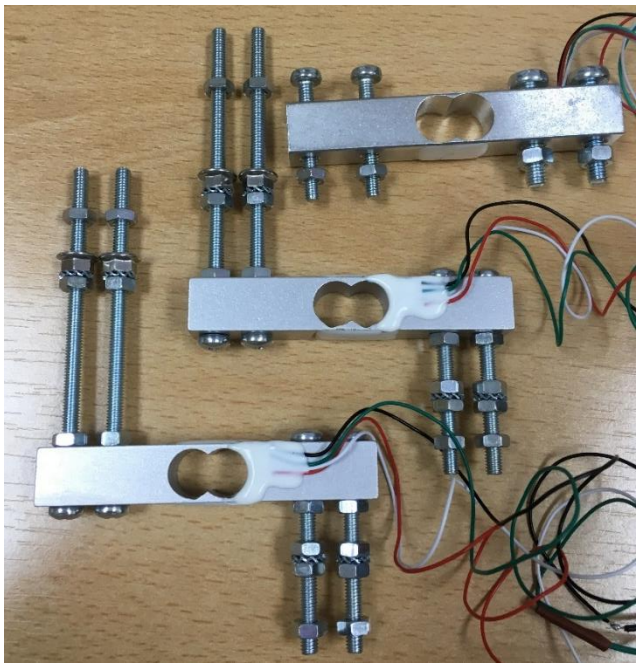
Useful information

- Bolts used in the project are all size M4, except for the ones to tie the model to the balance which are M3, and two of the drag load cell, which are size M5.

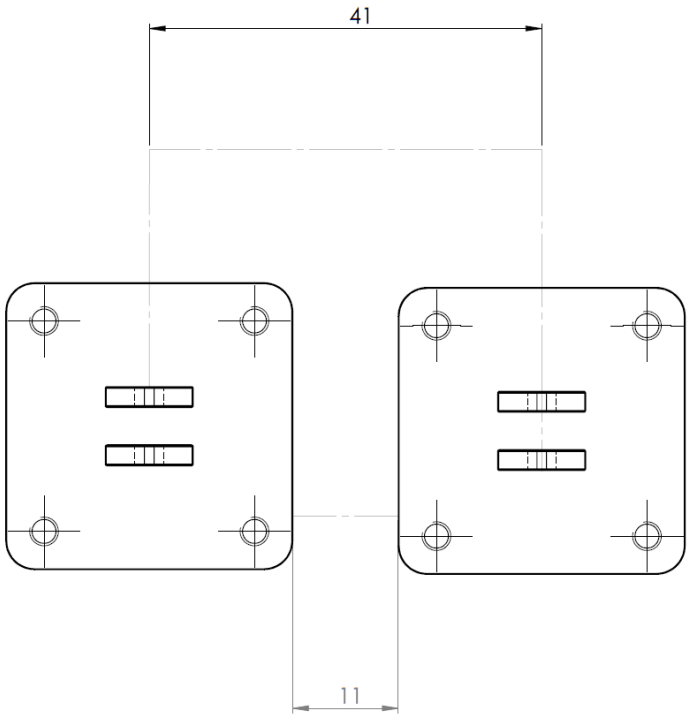


Connection diagram of the load cells and amplifiers to Arduino. Pin A1 is data of load cell 1 (lift), A2 is data of load cell 2 (lift) and pin A3 data of load cell 3 (drag). Note that the Arduino has 3 GND and each black cable is connected to one of them.

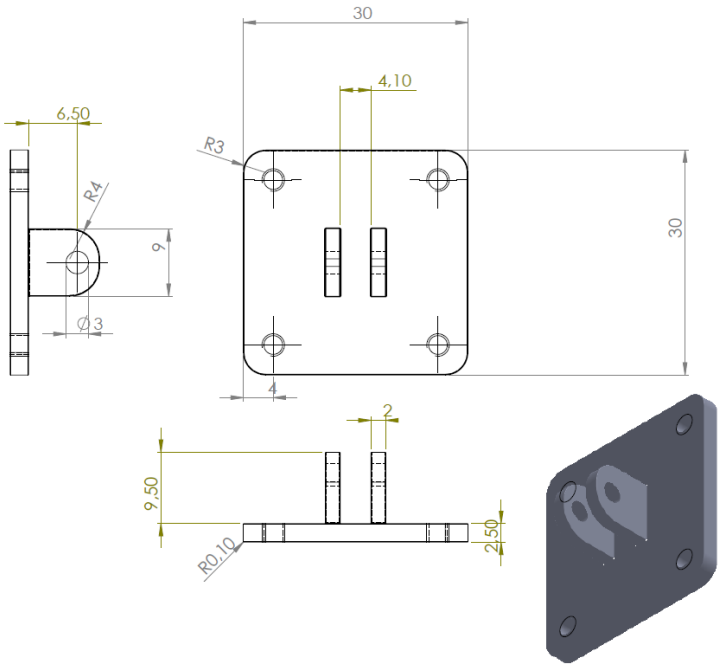
- You can vary the speed at which the numbers appear by changing the delay at the function void()
- You can vary the amount of numbers that are used to obtain a mean value at…
- Data wires are connected to analog pins but they are being used as digital pins, data transmission is not analogic it is digital.
- Safe overload for the load cells is 120%, while ultimate overload is 150%.



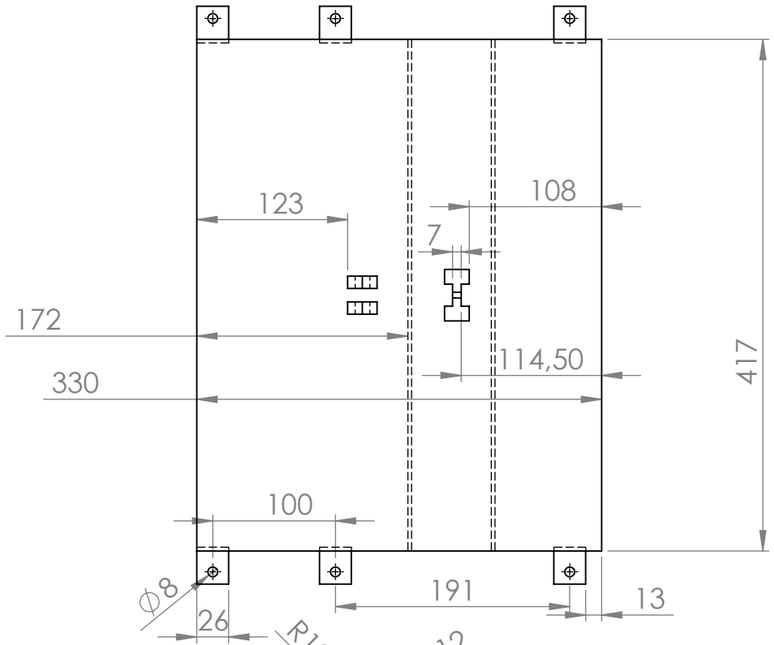
Load cell bolts and nuts.

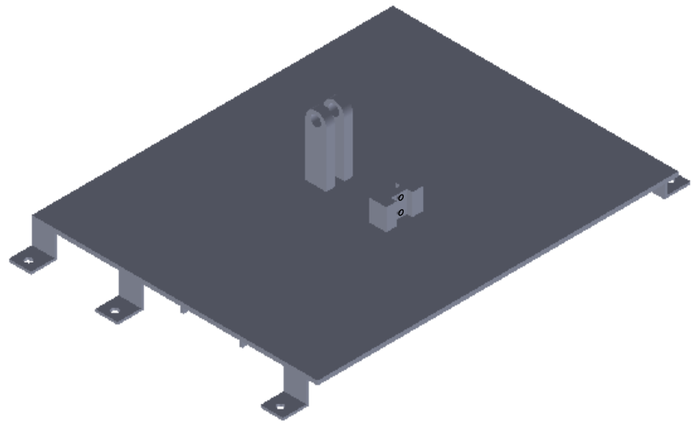


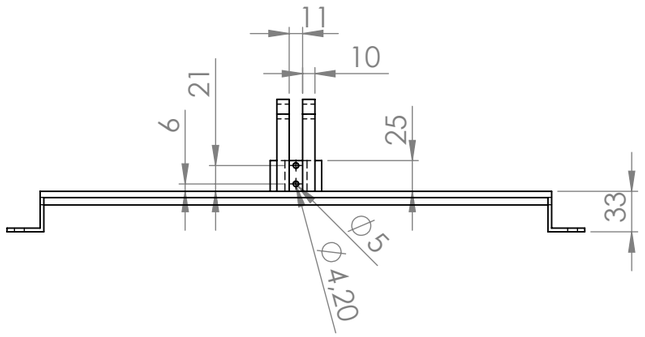
Position of the two pieces to attach the model.

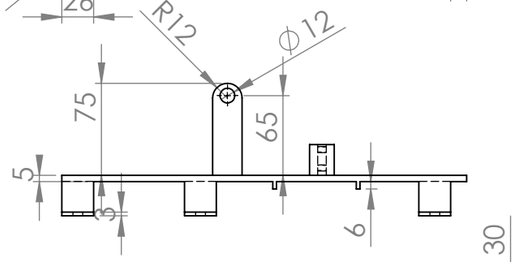


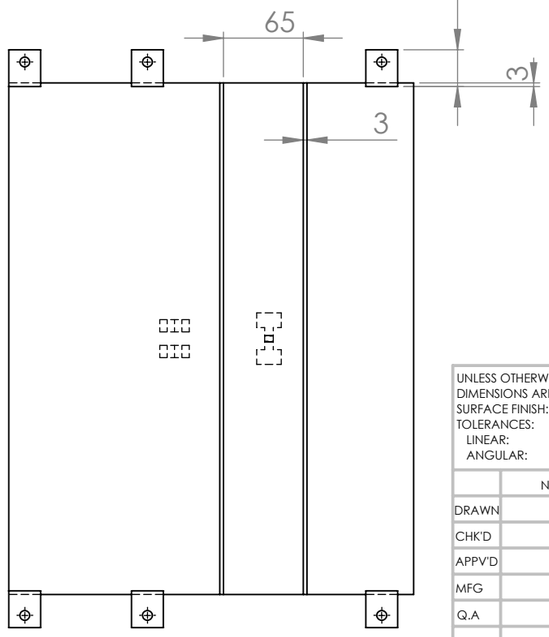
Measures of the piece that should be attached to the model.



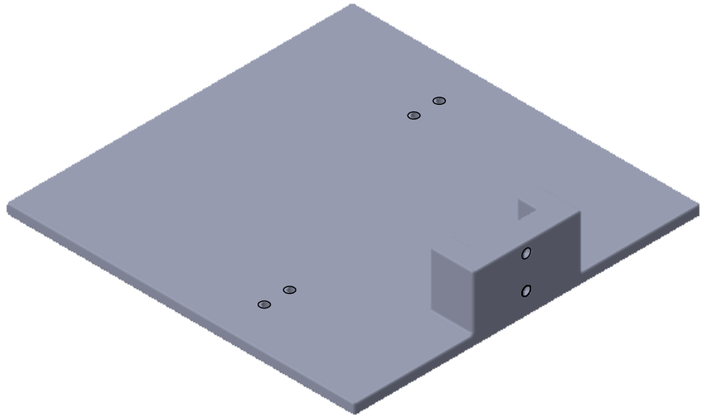


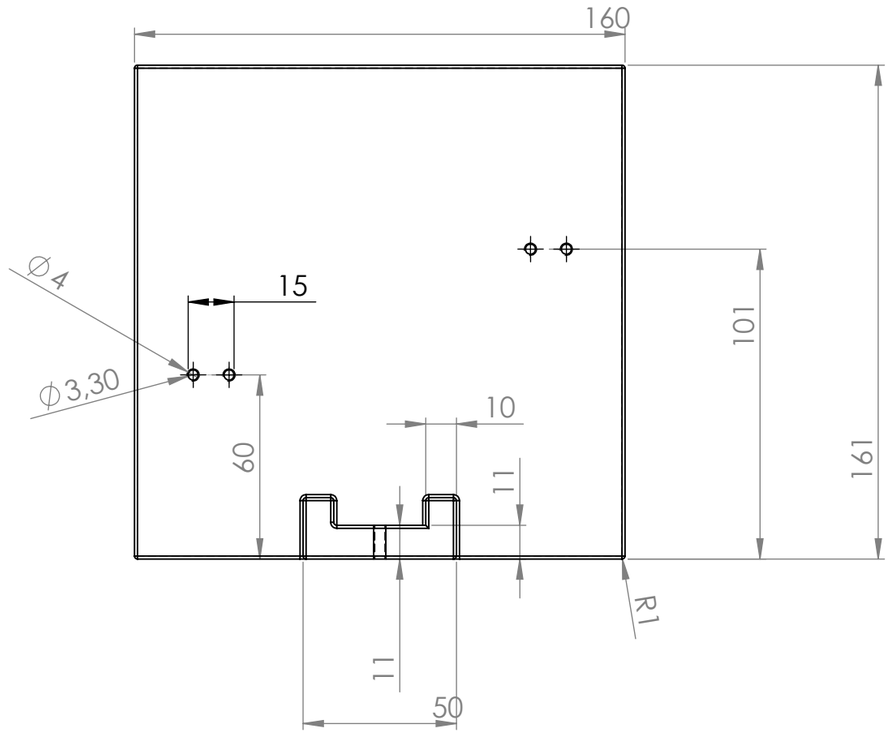


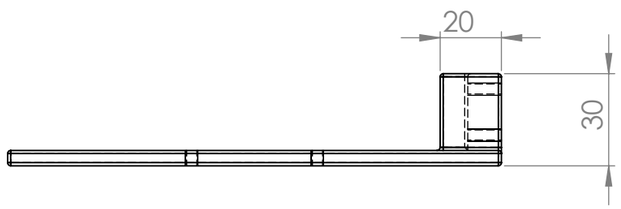


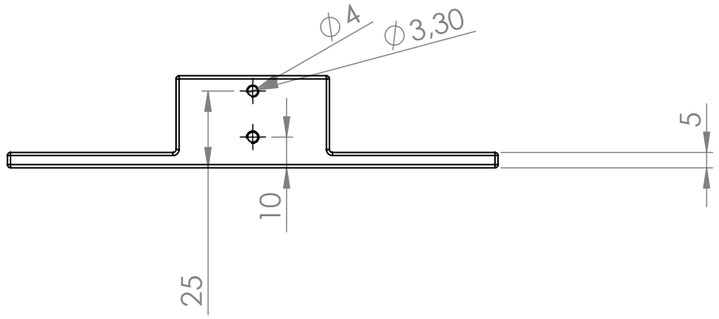


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO. base	A3
				WEIGHT:		SCALE:1:5	SHEET 1 OF 1

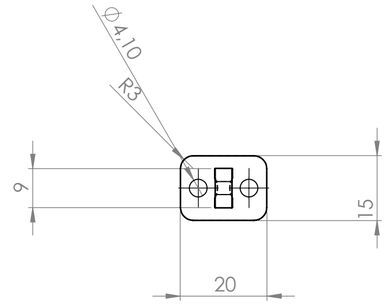


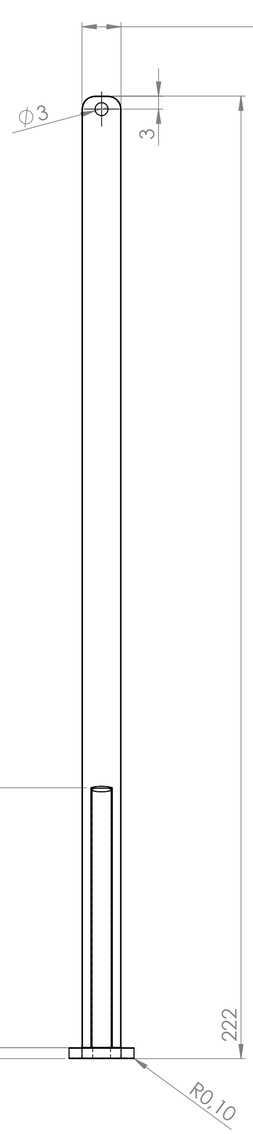


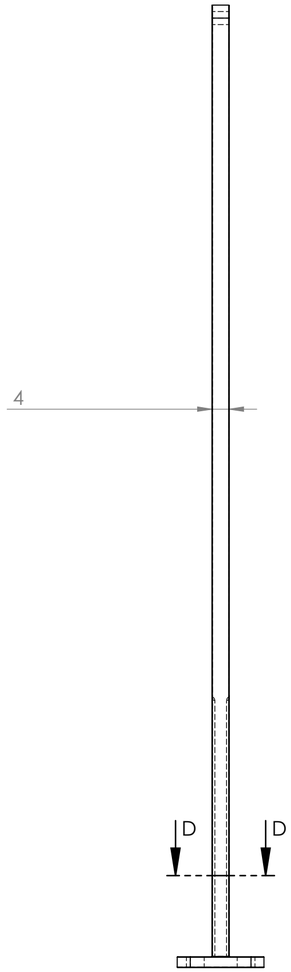


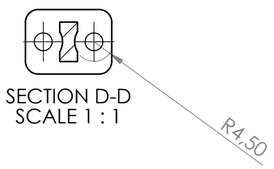


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:				DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING / REVISION
	NAME	SIGNATURE	DATE				TITLE:
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:			DWG NO. base-loadcells / A3
				WEIGHT:			SCALE:1:2 / SHEET 1 OF 1

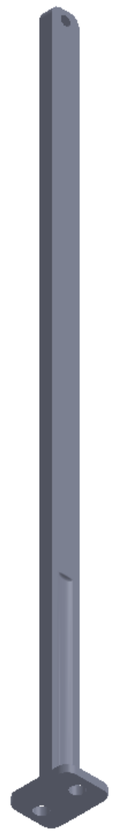




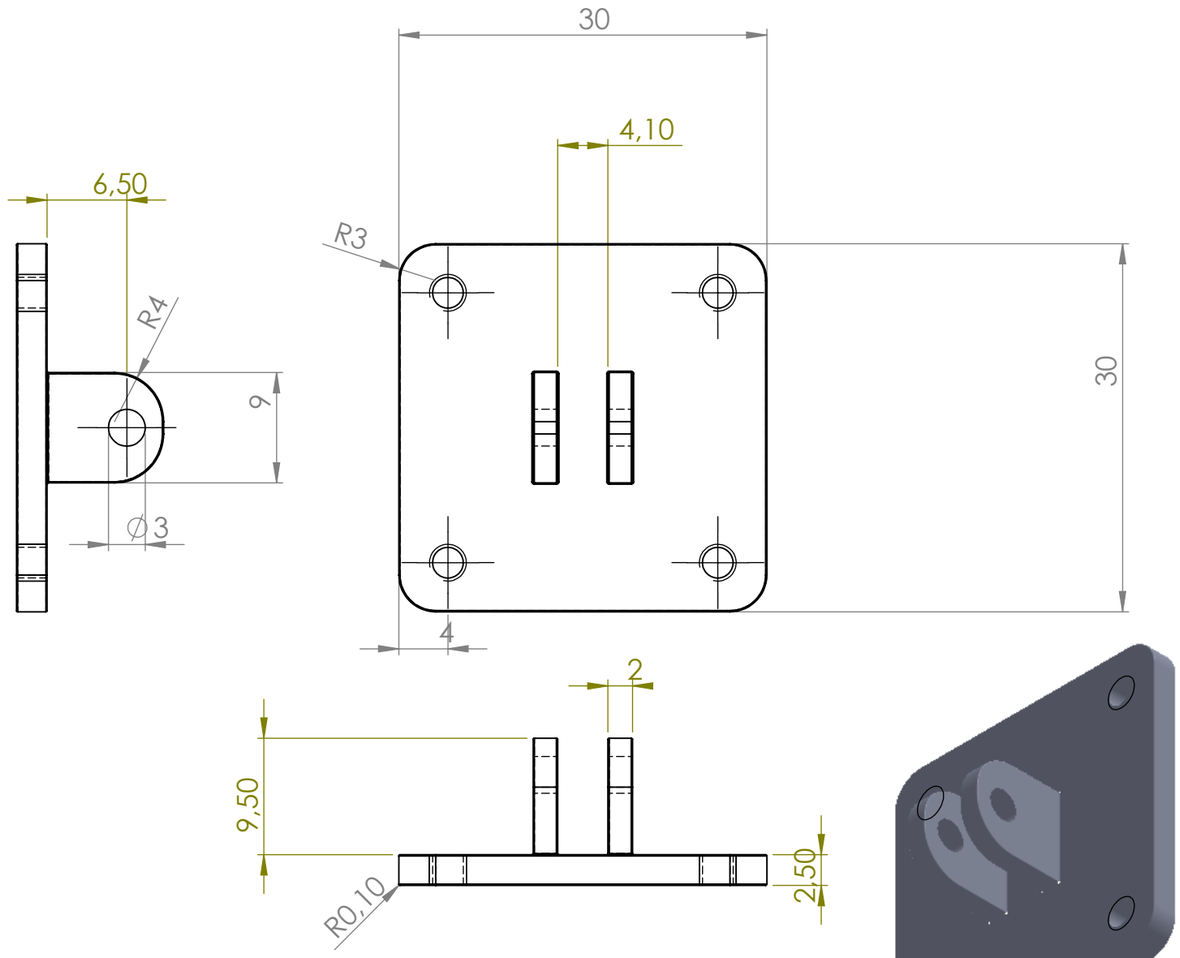




SECTION D-D
SCALE 1 : 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO. long-bar	A2
				WEIGHT:		SCALE:1:2	SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO. top-piece	A3
				WEIGHT:		SCALE:2:1	SHEET 1 OF 1

University of Zagreb
Faculty of Transport and Traffic Sciences
10000 Zagreb
Vukelićeva 4

DECLARATION OF ACADEMIC INTEGRITY AND CONSENT

I declare and confirm by my signature that this undergraduate thesis is an exclusive result of my own work based on my research and relies on published literature, as can be seen by my notes and references.

I declare that no part of the thesis is written in an illegal manner, nor is copied from unreferenced work, and does not infringe upon anyone's copyright.

I also declare that no part of the thesis was used for any other work in any other higher education, scientific or educational institution.

I hereby confirm and give my consent for the publication of my undergraduate thesis titled **Design of a three-axis wind tunnel force balance** on the website and the repository of the Faculty of Transport and Traffic Sciences and the Digital Academic Repository (DAR) at the National and University Library in Zagreb.

In Zagreb, 18 July 2018

Student:

(signature)