Tianjin University
Sensor Course

Denche Thibaud
Dolphens Nicolas

# Angle manager

# Summary

1 – Concept, material and assembly

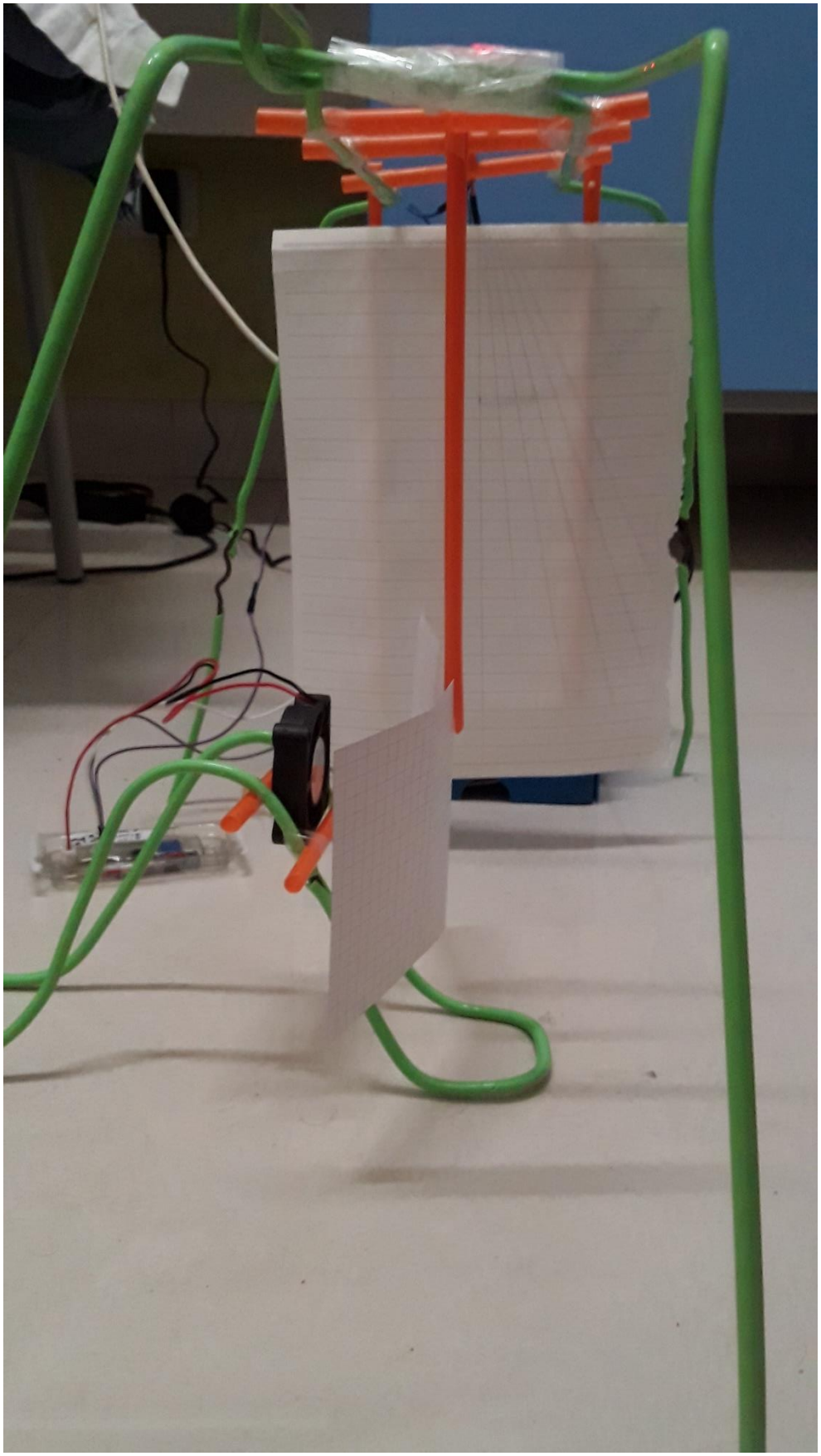
The goal is to push a rod thanks to wind strength to reach an angle which is asked by the user. This angle will be printed on a GUI and readable on a paper thanks to the shadow of the rod.

For this project, we have :

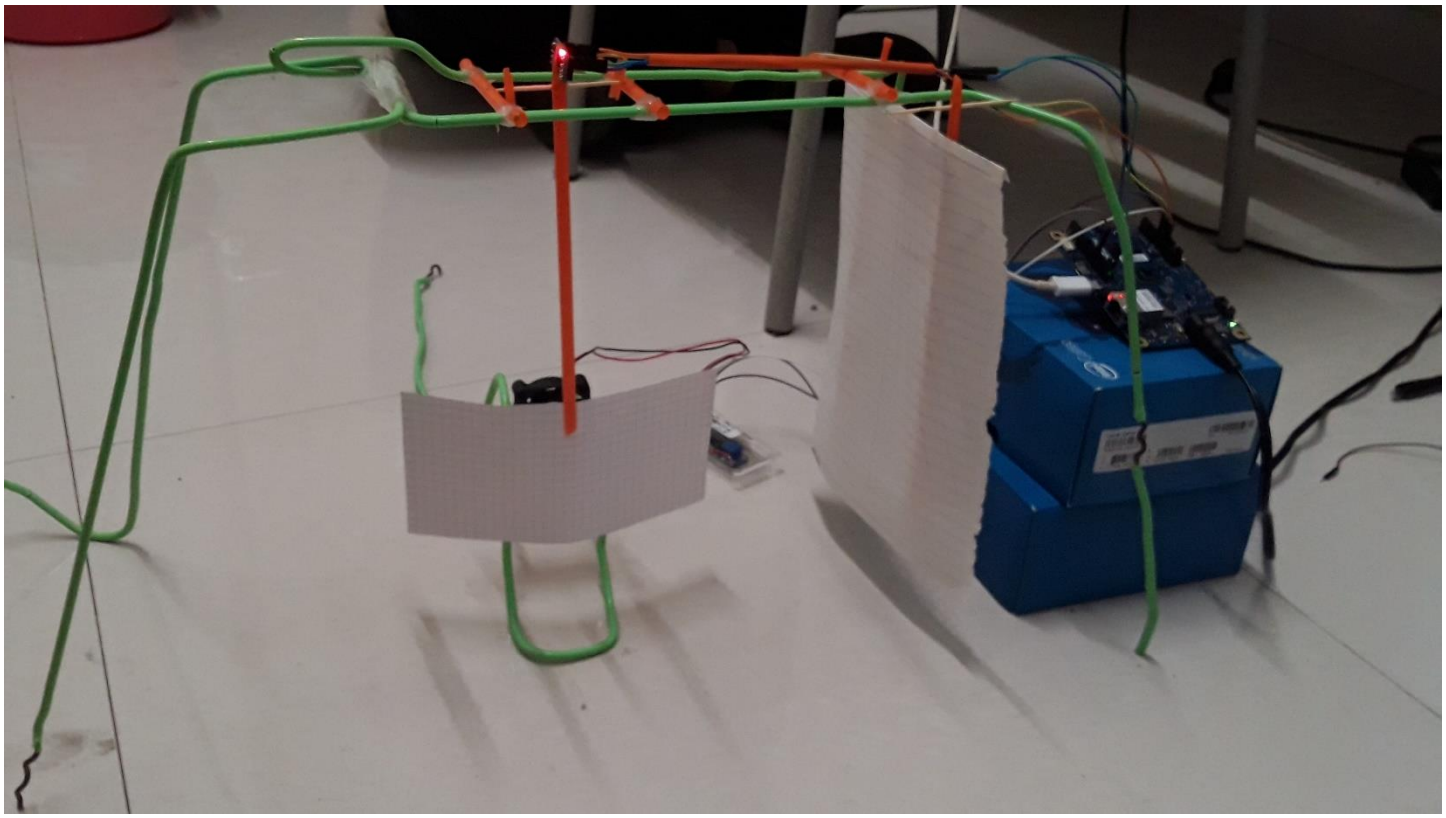
- An Arduino
- Two sensors :
    - An accelerometer
    - A fan
- Electricals wires
- A structure made by ourselves which is made from :
    - Hangers
    - Drinking straws
    - Toothpicks
    - Adhesive tape
    - Paper
    - A light (in this case, from a cellphone)

With all of these things, we have the structure below :

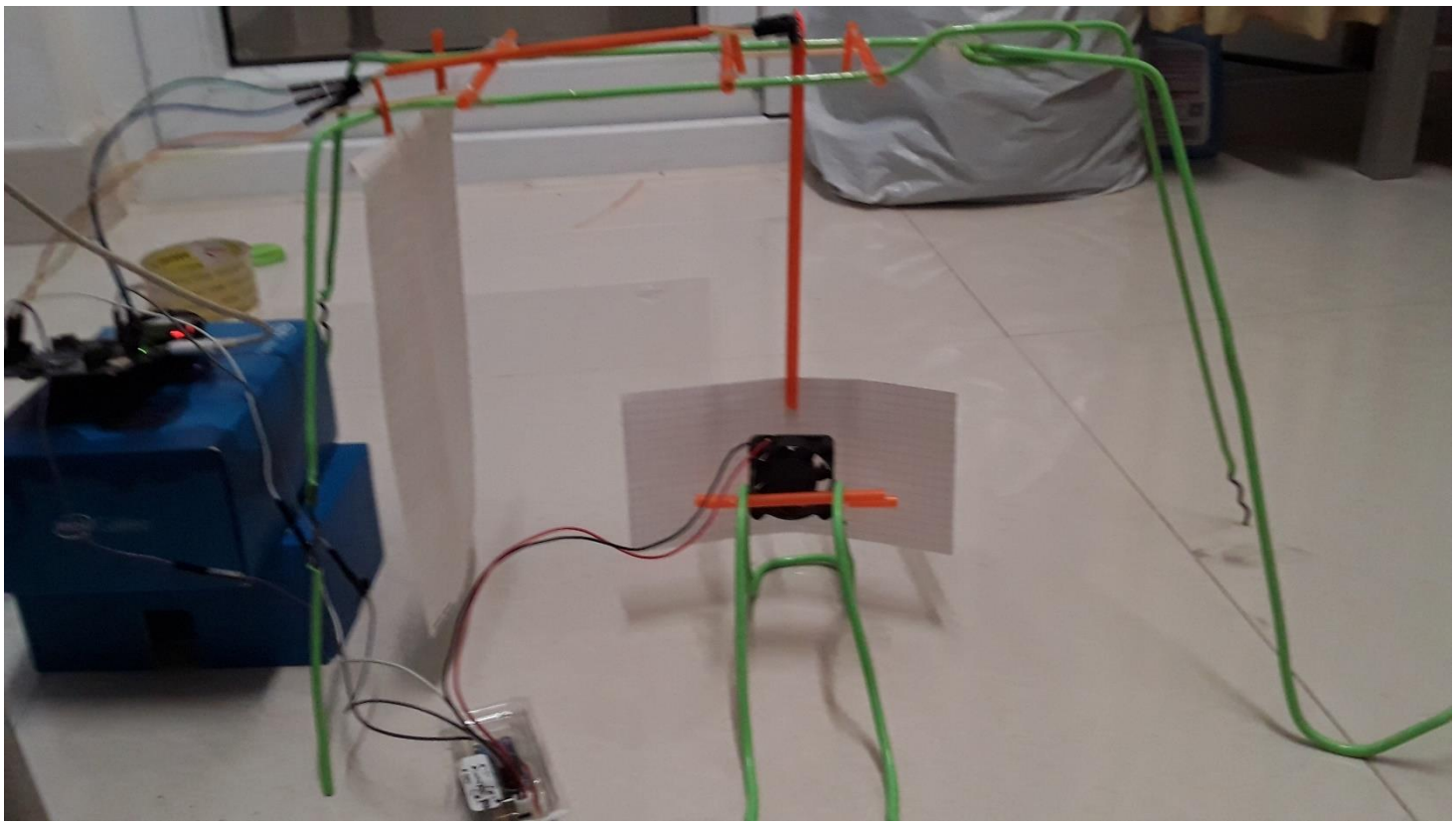
At the position of the light (south side) :



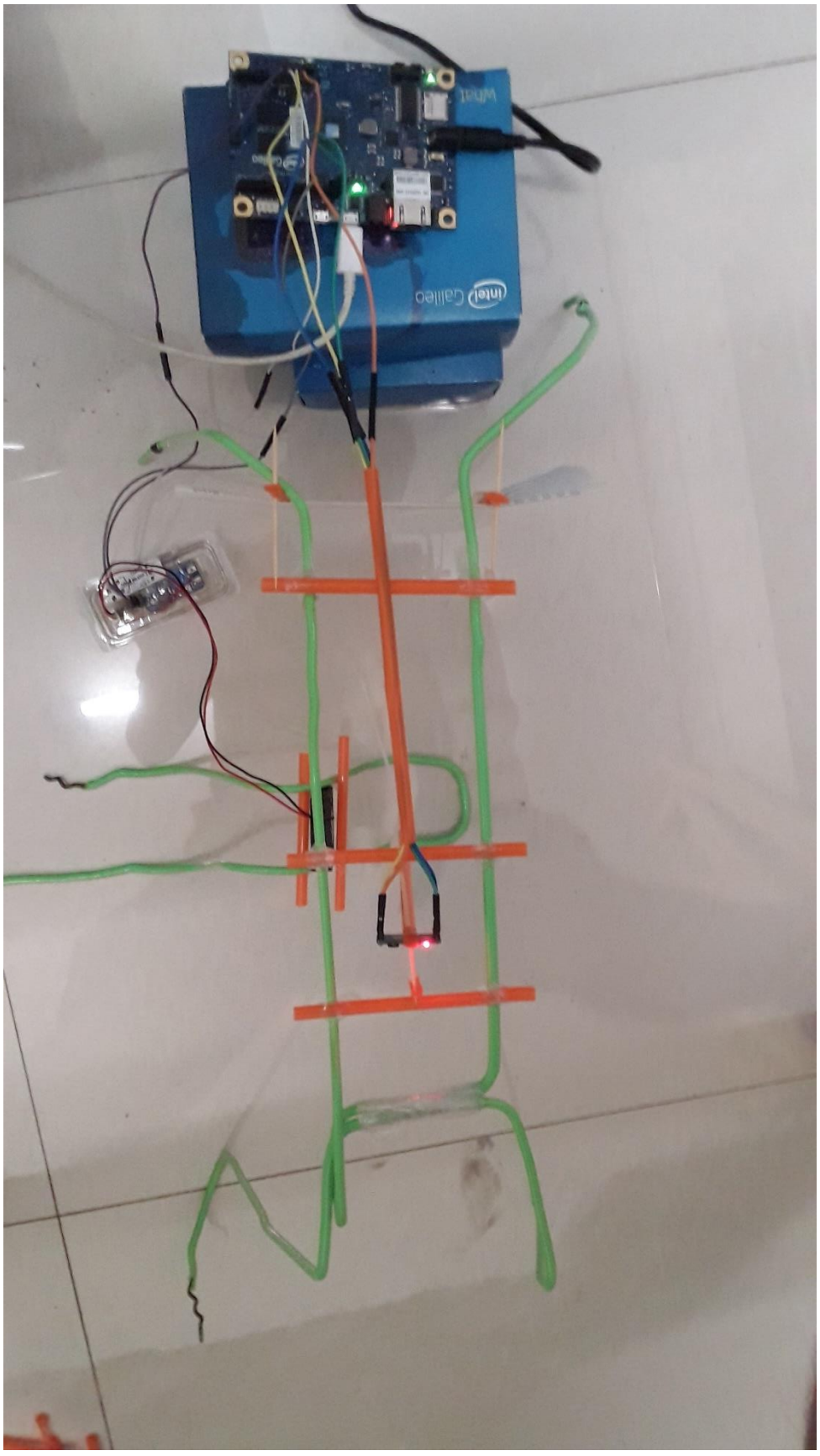
East side :



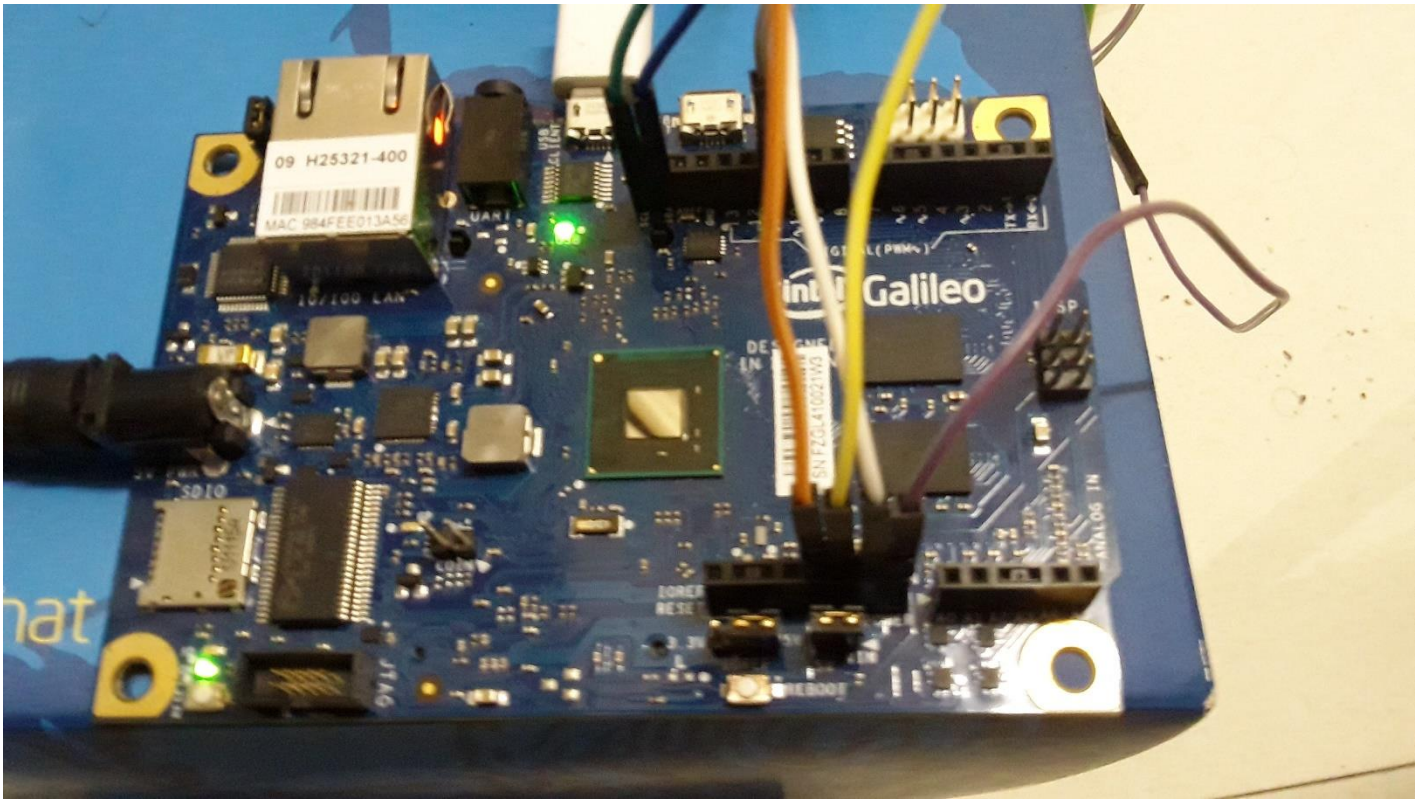
West side :



Top side :



Wires connexions on Galileo card :




On this picture, there are the connexion of the fan and and accelerometer to the Galileo card.

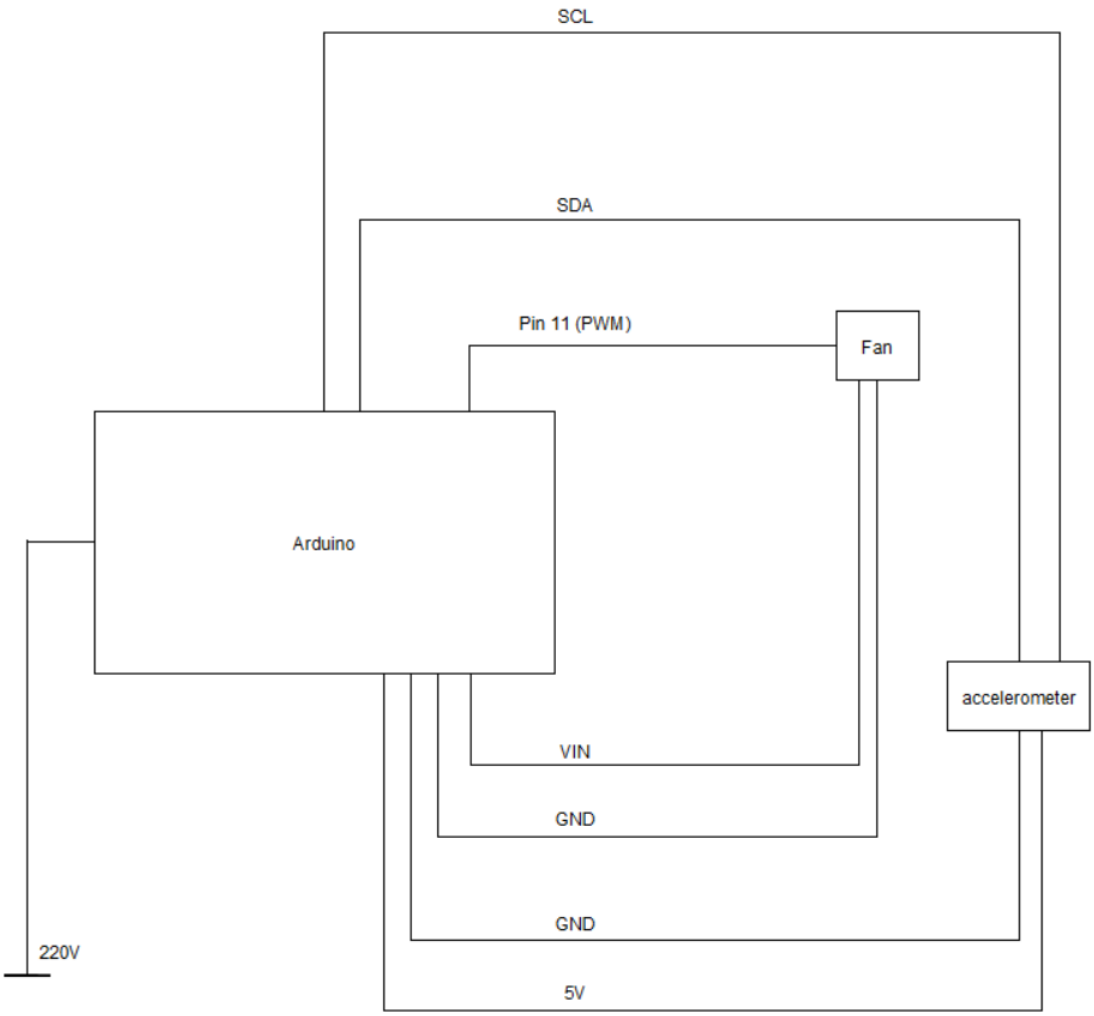
4 wires accelerometers to :

- 5V
- GND
- SCL
- SDA

3 wires fan to :

- GND
- VIN
- pin 11 (pin wich support PWM mod)).

Furthermore, the Galileo card is connected with it serial port to a computer (USB port) and the Galileo card is fed from a wall outlet (220V) wich deliver a 5 volt power to the card.




Lastly, the structure have been thought for a minimal resistance/strain at rode level. In fact, the principal problem was to build a structure where the « pendulum » was as much free as possible in its movements without frictions. We solved this problem thanks to a toothpick between two drink straws. Moreover, we had to adapt the height of the structure because of the aerodynamism of the « pendulum » and for a strain problem because of the size of the wires.

# 2- Softwares and code architecture

## a) Softwares

About the softwares, we actually use two. The first one is « arduino », which interact with the Galileo card. The second one is « Processing 3 » coupled to « G4P » to build a user interface (GUI).

« Processing 3 » is the software used in this case to communicate (read and write) with the Galileo card on the serial port.

G4P have been chosen instead of another one, like « control IP5 » because of it simplicity.

## b) Code architecture

First, the user need to enter an angle on the GUI wich will be communicated to the arduino as soon as the user have pushed the « ok » button. If the angle is impossible to reach, there will be message in the text box expected for the messages.

After that, the computer will communicate the value thanks to the serial port to the arduino. For a better communication, the read and write acces will be managed by two circular

buffer (loop buffer). One on the computer, and one on the arduino which allows a clean communication without interuption.

In the third, the arduino will calculate the value required by the accelerometer to reach the angle and communicate it to the computer, which will display it on the GUI.

The arduino will start to reach the angle now. The value given to the fonction which will manage the PWM method (further explanations in the part 3) is 10 * the angle asked by the user. This allows the fan to start fastly, instead of increase it duty cycle every 300ms from 0%. At the same time, the arduino communicate the value given to the function which manage the PWM method to the GUI.

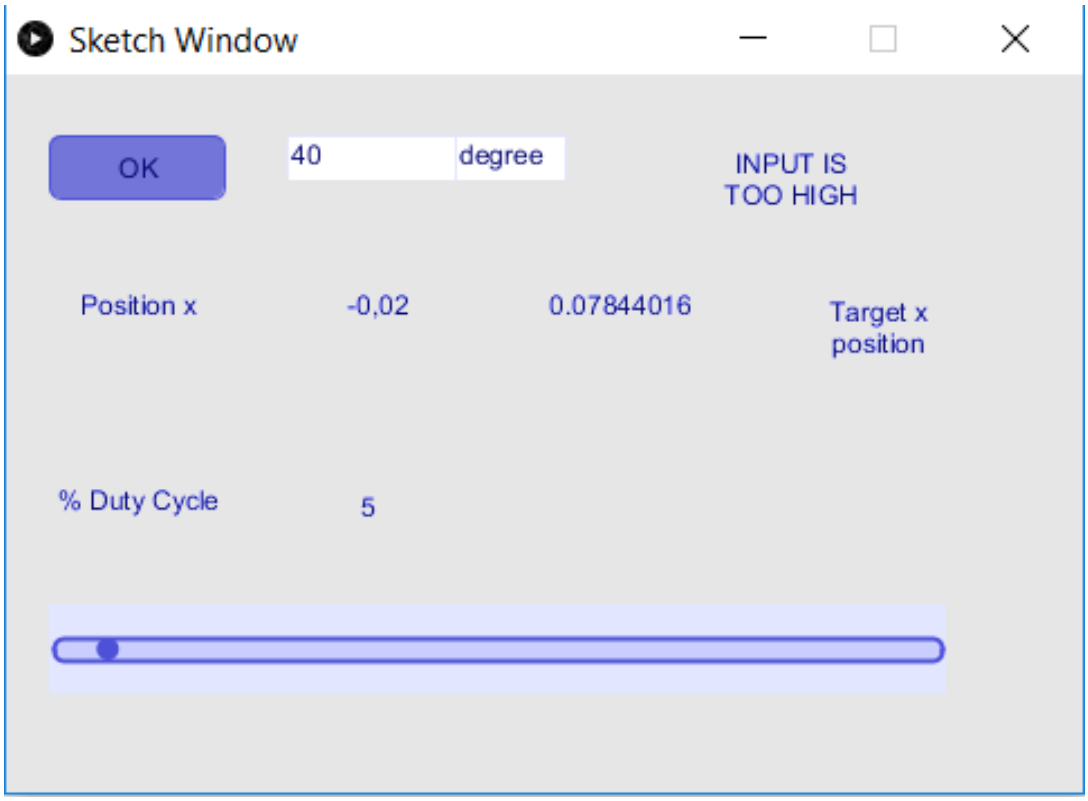
Until the angle asked by the user have been reached, the arduino calculate the mobile average of the X position of the accelerometer (on the last 10 values) and send it to the GUI. As soon as the angle is reached, the arduino keep the same value for the duty cycle of the fan but keep sending the mobile average of the X position of the accelerometer. The text box of the GUI will confirm that the angle have been reached correctly. Moreover, you can read the duty cycle of the fan in percentage on the slidebar.

Finally, the program will keep running like this and stabilize the angle as much as possible until the user enter a new angle in the GUI and press the « ok » button.

# 3- functioning of the code

For a greater clarity, let's explain how the code works.

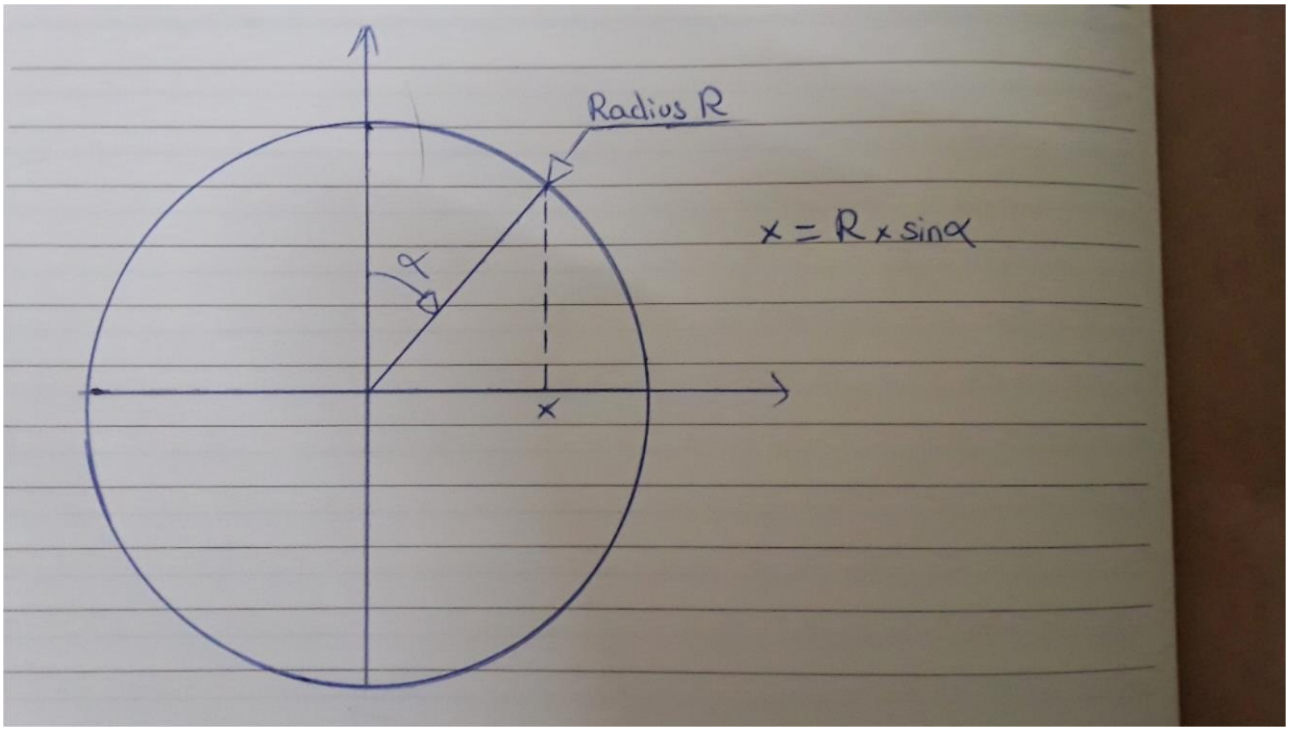
First, an angle will have to be entered on the GUI by the user.




At this step, we know by calibration that the radius of our pendulum is 0.90 (value returned by the accelerometer from an 90° angle (0.95) less the value at the initial position (0.05)). In this case we just need to apply trigonometry formulas. There is a easy way to remember those formulas : SOHCAHTOA.

In fact, to calculate sinus of an angle, we have to divide the opposite side by the adjacent one. For cosinus, it's the

adjacent divided by the hypotenuse. And for the tangent, divide the opposite by the adjacent. Since we know that, we have calcul the value of the accelerometer at the angle the user entered. So we use the diagram below :




Hence, we apply the formula **X = radius x sin(angle * (π / 180))** and we have the value which have to be reached by the accelerometer to obtain the angle asked by the user.

For example :

The user want an angle of 34°. So the formula will be

X = 0.90 x sin(34 * ( π / 180)).

In this case, X = 0.53 (rounded value).

So when the value of 0.53 will be reached by the accelerometer, the angle will be reached by the « pendulum ».

For more accuracy and because the accelerometer pushed by the fan will always have variations, we calculate a mobile average on the ten last values on the position of the accelerometer. So the exact sentence should be when the mobile average calculate by the arduino will be reached, the angle asked by the user will be reached.

And now ? How to stabilize the fan to keep this angle ?

Thanks to the pin 11 we chosed on the Galileo card, we are able to use PWN method.

So What is the PWM (Pulse with modulation) method ?

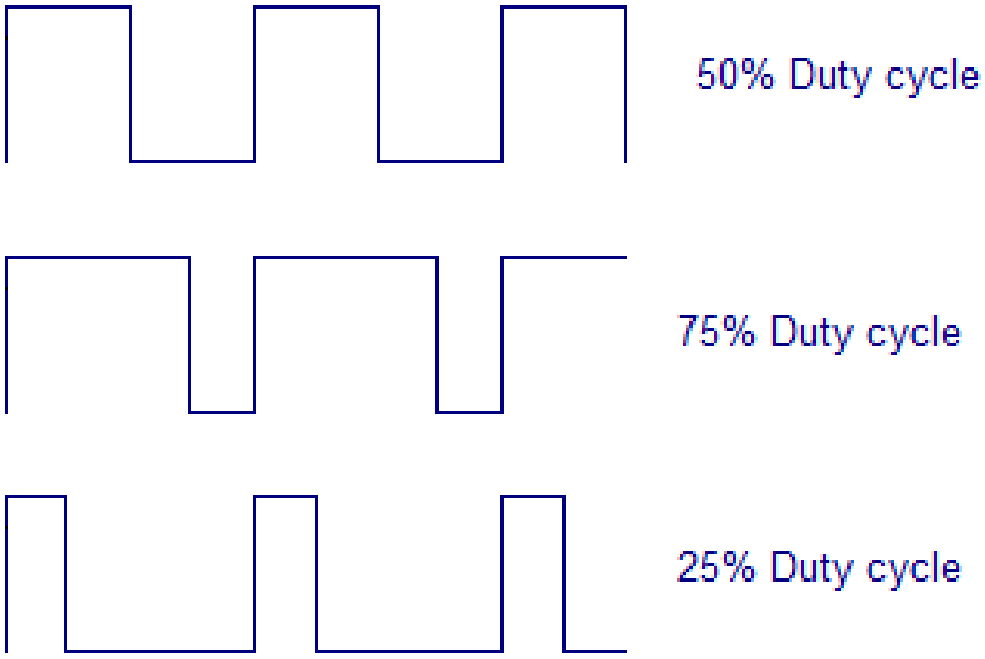
The PWM signal is essentially a high frequency square wave and the duty cycle of this square wave is varied in order to vary the power supplied to the load. Stated in percentage, it can be expressed using the equation :

**% Duty cycle = ($T_{on}$/($T_{on}$ + $T_{off}$)) * 100**

Where $\mathbf{T_{on}}$ is the time for wich the square wave is high and $\mathbf{T_{off}}$ is the time for wich the square wave is slow.

When duty cycle is increased, the power dropped across the load increases. However, when duty cycle is reduced, power across the load decreases.

Below some examples of duty cycles :




With this method, we are able to give a duty cycle to our fan to keep the angle asked by the user. For that, we use a fonction or the arduino software wich is :

**analogWrite(digitalPin, value)**

With digitalPin the pin where the sensor is connected (in this case, 11) and value the return of the fonction **analogRead(analogPin)** which return the voltage of the analog pin into integer value between 0 and 1023.

The value passed as second parameter in the fonction analogWrite have to be divided by 4 because the fonction need a value from 0 to 255. 0 for a duty cycle of 0% (fan OFF), and 255 for a duty cycle of 100% (fan ON full powered).

Go back to our code. Using this method, we just have to check the value of the accelerometer and increase the value of the second parameter of the analogWrite function. This will have for effect to increase the duty cycle of the fan and it has resulted in a bigger angle.

When the duty cycle don’t have to be increased anymore, the angle will be stable. It’s readable on the text box, but also on the slidebar of the GUI. If the value can’t be reached because of the power of the fan is too weak, the program will display an error message in the text box. Otherwise, it’ll display a confirmation message in the same text box.