# Présentation station météo avec capteurs Grove

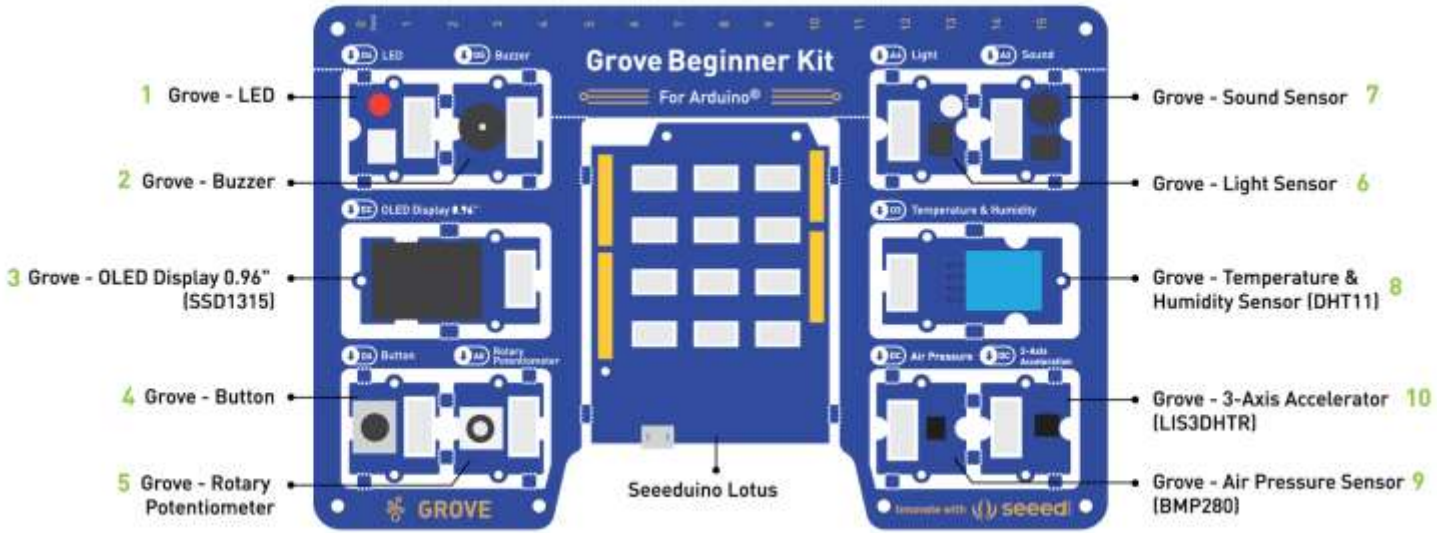
# Table des matières

# Introduction à Grove

Présentation de Grove : Conçu par Seeed Studio, Grove est une gamme de module électroniques capteurs et actionneurs pratiques et faciles à brancher, permettant de rendre les montages plus simples et plus rapides à réaliser. Des cartes d'interface (appelés shields) permettent de raccorder les modules Grove à des cartes programmables, tels que les cartes Arduino, Rapsberry PI, etc… Il existe également des cartes de la gamme Seed qui sont basées sur des cartes existantes. C'est le cas de la Seeeduino Lotus, que nous allons utiliser pour notre station.

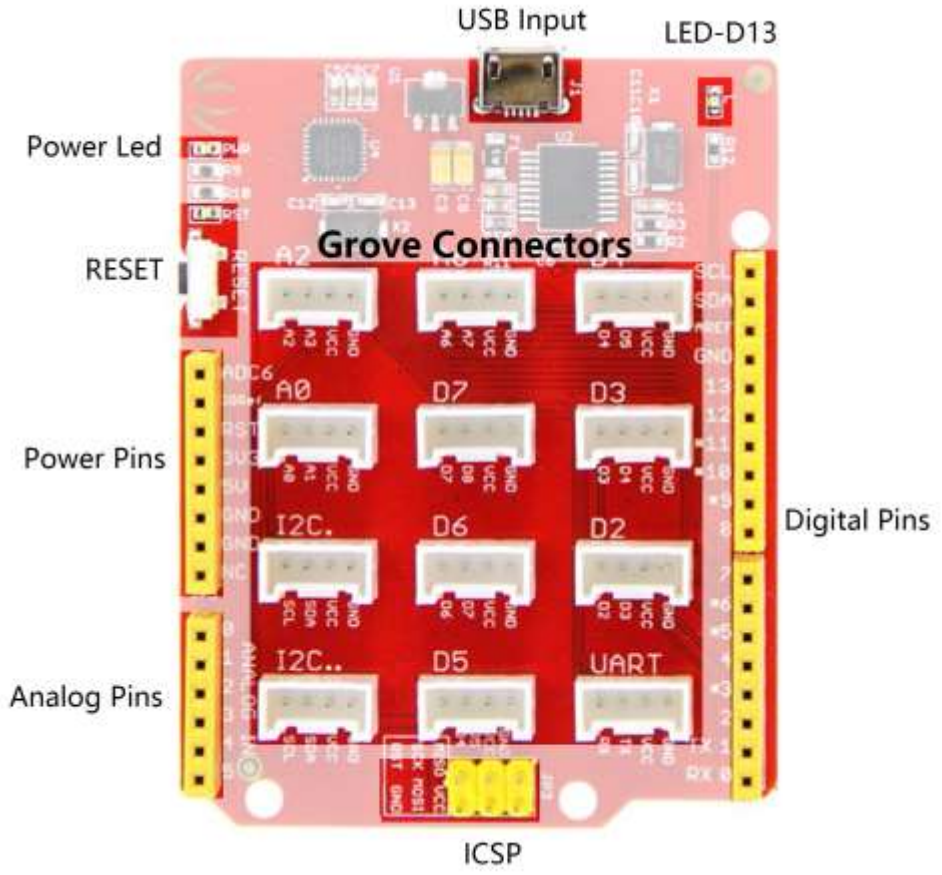
Pour notre station météo, nous avons opté pour un starter kit, disponible sur le site lextronic.

Voici un schéma ci-contre présentant les différents modules du kit :

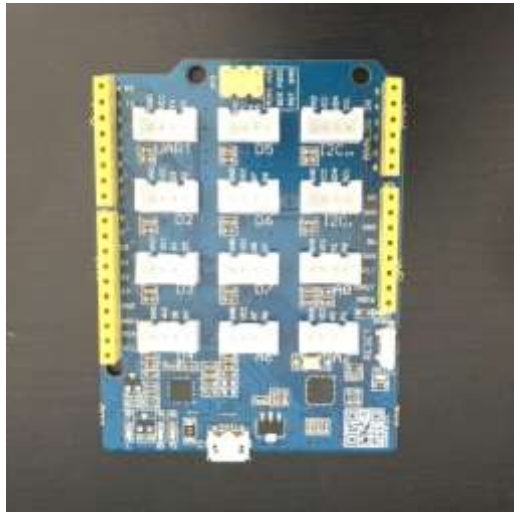


# Présentation de la carte Seeeduino Lotus

La carte Seeeduino Lotus est basée sur la carte Arduino UNO, ce qui la rend compatible avec les shields, les programmes et l'IDE Arduino. Elle peut être programmée en utilisant les ports sur le côté, comme une carte « classique » (ports analogiques, digitales et d'alimentation), ou en utilisant les connecteurs Grove qui se trouvent au centre de la carte. Certains de ces connecteurs sont dédiés à une entrée / sortie numérique (D2 à D7), d'autres à une entrée / sortie analogique (A0, A2, A6). Les ports I2C sont réservés aux modules OLED, et le port UART est dédié au Wi-fi.



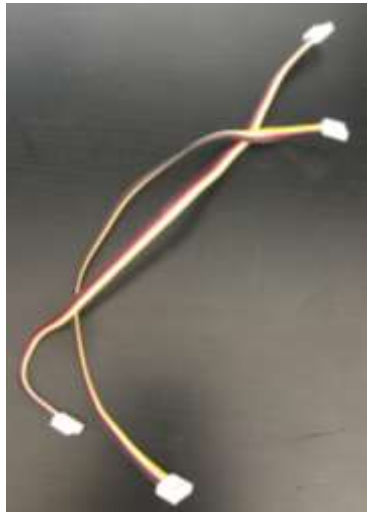
# Présentation du matériel utilisé (1)



Carte Seeeduino Lotus

**Il faut souder les résistances et les ports à la plaque**

Platine d'interface : d'un côté nous avons les 22 connecteurs RJ11 (noirs) qui nous permettent de relier les capteurs météo, et de l'autre nous avons les connecteurs Grove (blancs) qui relient les capteurs à la carte



Câble grove, qui sont reliés aux connecteurs de la carte

# Présentation du matériel utilisé (2) : les capteurs météo

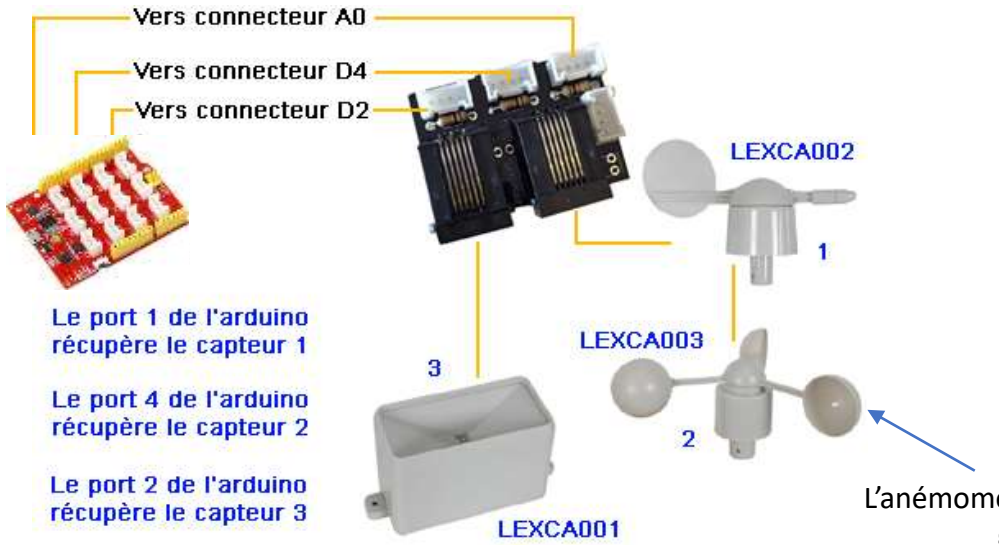
Nous avons choisi un kit de 3 capteurs météo Grove, à savoir : un anémomètre, une girouette, et un pluviomètre. Voici le lien du kit, disponible sur le site lextronic.

Pluviomètre :
Calcule la quantité d'eau qui s'écoule

Girouette :
Montre la direction du vent

Anémomètre :
Mesure la force du vent

# Présentation du montage

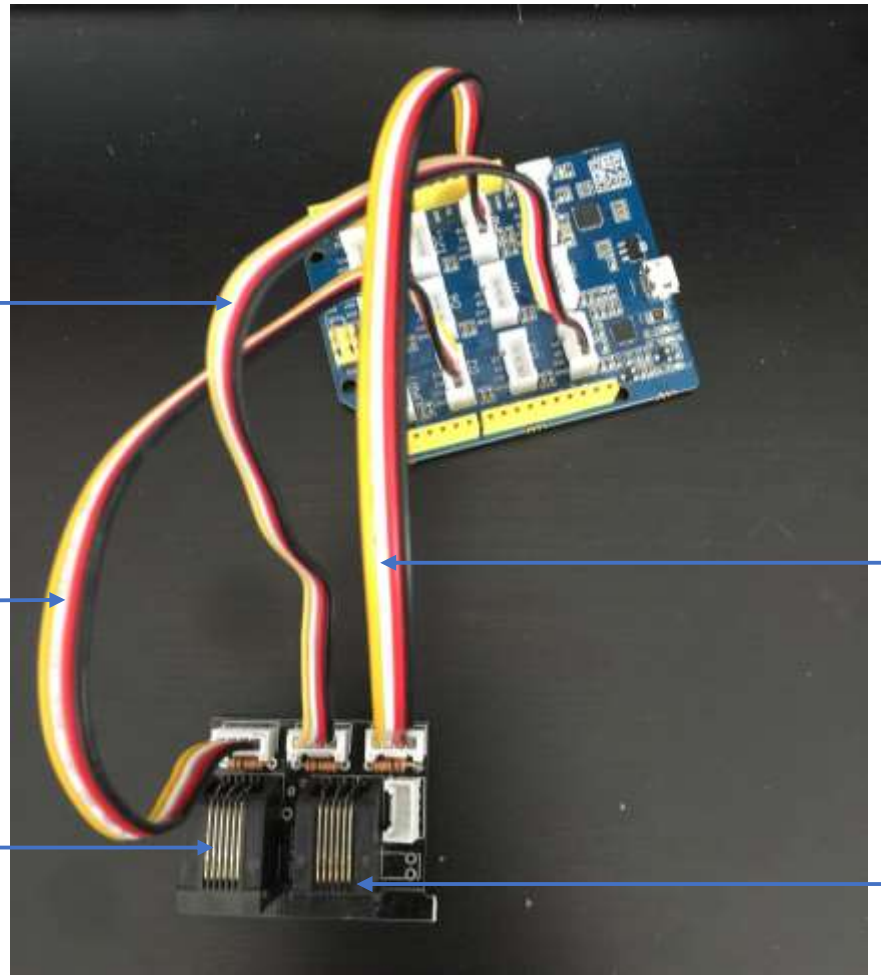



L'anémomètre est associé à la girouette

*Remarque :*

*L'anémomètre est relié à la girouette, c'est-à-dire que son branchement se fait sur ce capteur. On relie ensuite les connecteurs RJ11 du pluviomètre et de la girouette sur la platine d'interface. Enfin, on relie la platine avec la carte Seeeduino à l'aide de capteurs Grove, sur les connecteurs adéquats (dans notre cas, nous avons branché l'anémomètre au port 4, le pluviomètre au port 2, et l'anémomètre au port A0).*

# Présentation du matériel utilisé (1) : suite



Câble reliant l'anémomètre (D4)

Câble reliant le pluviomètre (D2)

Port RJ11 dédié au pluviomètre

Câble dédié à la girouette (D4)

Port RJ11 dédié à l'anémomètre + girouette

Connecteurs RJ11

Connecteurs Grove

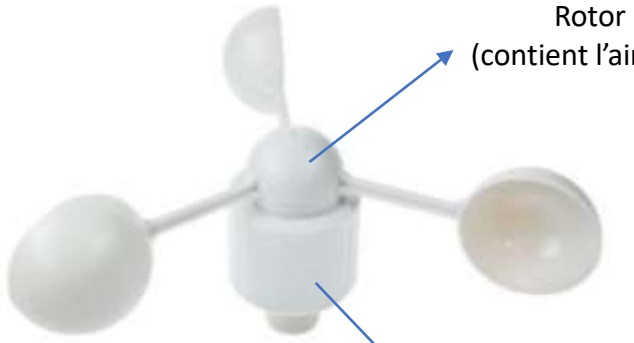
# Fonctionnement de l’anémomètre

L’anémomètre est un appareil capable de mesurer la force du vent. Notre modèle est à coupelles (il en possède 3). Pour détecter la rotation des coupelles, le capteur est équipé d’un ILS (=interrupteur à lames souples) au niveau du stator. Lorsqu’on approche un aimant de l’ILS, les 2 lames se touchent (=circuit fermé), et elles se détachent lorsque l’aimant s ’éloigne.

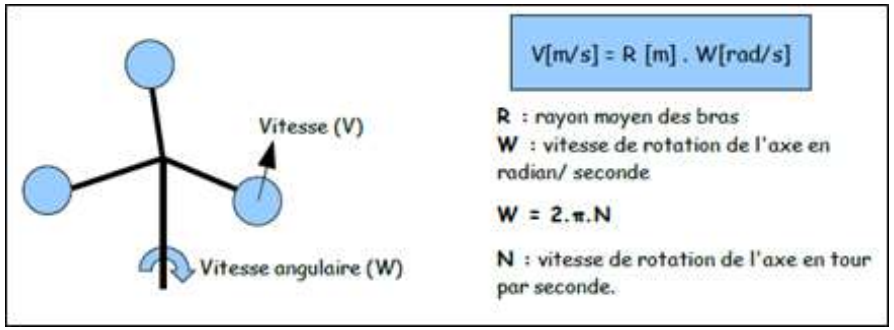
Interrupteur à lames souples
(ILS)

Capteur ILS : Ampoule de verre contenant une atmosphère inerte et deux lames métalliques, magnétisables et électriques, très faiblement espacées.

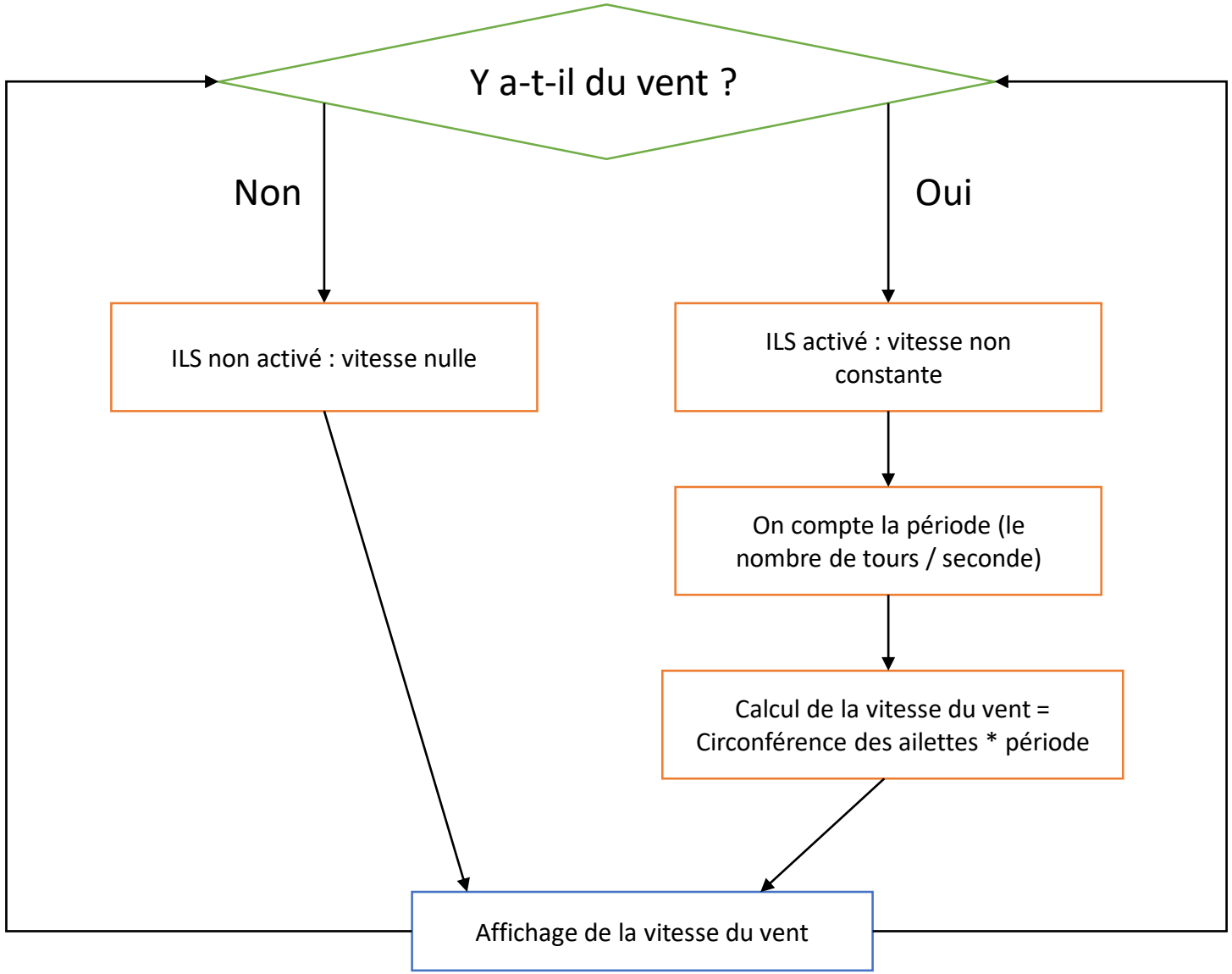
Le rotor contient quant à lui un aimant. Lorsque le rotor fait un tour autour du stator, l’ILS va se déclencher 2 fois. On considère qu’un vent de 2,4 km/h provoque une fermeture du contact/seconde.



# Algorithme fonctionnel de l'anémomètre




Délai : 35 ms



# Code de l’anémomètre

```
#define pinILS 4
#define pi 3.14159265359
#define RayonDesBras    0.07 // en mètre de l'anénomètre

void setup()
{
  pinMode(pinILS, INPUT);
  Serial.begin(9600);
}

unsigned long millis_old(0);
float deltaTime(0);
float vitesseVent(0);
float tourSec(0);

void loop()
{
    UpdateILS();

    nombreTourSec = 1/deltaTime;

    //vitesse du vent
    vitesseVent = (2*pi*RayonDesBras*nombreTourSec);

    //affichage de la vitesse
    Serial.print("la vitesse du vent est de ");
    Serial.print(vitesseVent);
    Serial.println(" m/s.");

}
```

```
void UpdateILS()
{
  static byte etatPrecedent=LOW;
  //lecture du capteur
  byte etat = digitalRead(pinILS);


   if (etat && etat!=etatPrecedent)
 {
   deltaTime = (millis() - millis_old) / 1000.0 ;
   millis_old = millis();
 }
 etatPrecedent=etat;
}
```

Le but de ce programme est de récupérer la vitesse (en m/s) à laquelle tourne l’anémomètre

# Code de l'anémomètre : Explication du code (1)

Variables utilisées :

Millis_old / deltaTime : variables qui vont nous permettre de déterminer la vitesse de rotation, en tour par seconde.

VitesseVent : vitesse du vent, calculée un peu plus loin dans le programme

Remarque : au début du programme, on définit le pin que l'on utilise (ici le pin 4), ainsi que la constante π et le rayon des ailettes du capteur, qui vont nous être utiles pour le calcul de la vitesse du vent.

```
#define pinILS 4
#define pi 3.14159265359
#define RayonDesBras    0.1 // en mètre de l'anénomètre

void setup()
{
  pinMode(pinILS, INPUT);
  Serial.begin(9600);
}

unsigned long millis_old(0);
float deltaTime(0);
float vitesseVent(0);
float nombreTourSec(0);
```

# Code de l’anémomètre : Explication du code (2)

On appelle la fonction UpdateILS(), dont le code sera expliqué dans la diapositive suivante.

Cette fonction récupère la valeur deltaTime, qui nous permettra de déterminer en combien de temps le stator effectue un tour complet autour de l’anémomètre.

Ensuite, on calcule la vitesse du vent. Pour cela, on calcule d’abord la circonférence du cercle décrit par les ailettes de l’anémomètre en mètres (2 x Pi x rayon) et on la multiplie par NombreTourSec. Attention ! Il faut ensuite diviser le résultat par 2, car à chaque tour, l’ILS est activé DEUX fois.

```
void loop()
{
    UpdateILS();

    nombreTourSec = 1/deltaTime;

    //vitesse du vent
    vitesseVent = (2*pi*RayonDesBras*nombreTourSec);

    //affichage de la vitesse
    Serial.print("la vitesse du vent est de ");
    Serial.print(vitesseVent);
    Serial.println(" m/s.");

}
```

# Code de l'anémomètre : Explication du code (3)

La fonction UpdateILS va nous permettre de calculer le temps

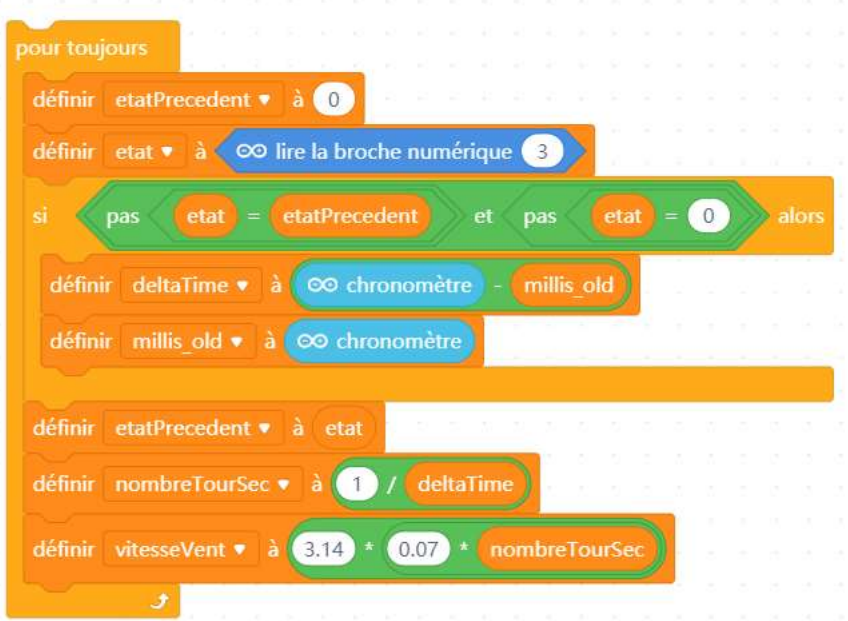
Pour cela, a chaque fois qu'on appelle la fonction, on va capturer la différence de temps entre l'état actuel du capteur et son état précédent. Cette différence de temps est stockée dans la variable *deltaTime*. Attention! Il faut diviser par 1000 afin d'avoir le résultat en seconde.

La variable *millis_old* prend alors la valeur de *millis()* , afin de pouvoir capturer la différence de temps au prochain tour.

*millis()* : chronomètre qui début lors du lancement du programme (en millisecondes). On ne peut pas le réinitialiser pendant.

```
void UpdateILS()
{
  static byte etatPrecedent=LOW;
  //lecture du capteur
  byte etat = digitalRead(pinILS);


   if (etat && etat!=etatPrecedent)
 {
   deltaTime = (millis() - millis_old) / 1000.0 ;
   millis_old = millis();
 }
 etatPrecedent=etat;
}
```

# Code de l’anémomètre (mBlock)



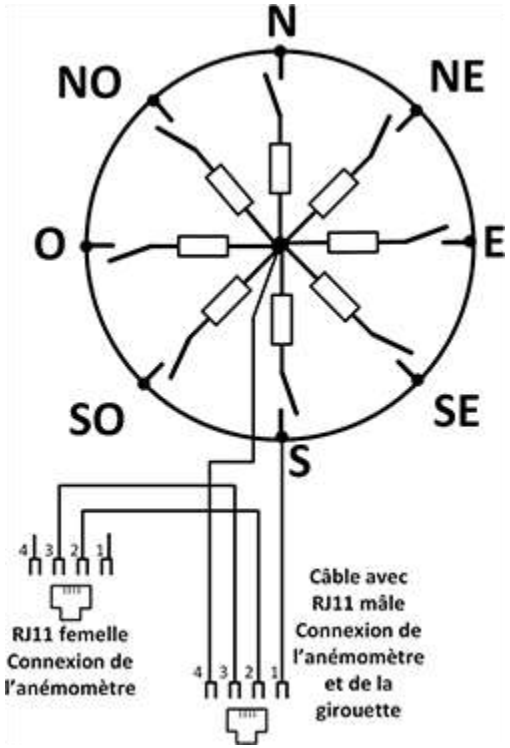

Code de l’Arduino


Code de l’objet

# Fonctionnement de la girouette

La girouette indique la direction d'où provient le vent. Elle est constituée d'un système rotatif comportant une pointe et une plaque qui s'aligne dans la direction du vent. La pointe indique alors la provenance du vent.

Le stator de la girouette comporte 8 ILS, qui correspondent aux 8 points cardinaux : N, NE, E, SE, S, SO, O, NO (voir schéma ci-contre). L'axe de la girouette comporte un aimant qui va tourner autour des ILS. Chaque ILS est connecté à une résistance avec une valeur propre à chaque direction. Lorsque la direction coïncide avec la position d'un ILS, l'aimant établit un contact et connecte une résistance. Lorsqu'il se trouve entre deux ILS, les deux ILS sont activés et connectent les deux résistances en parallèle.



# Code de base de la girouette (1)

La girouette doit être branchée sur un port analogique. Dans le programme ci-contre, on branche la girouette au port analogique A0.

Explication du code : Ici, on va lire la valeur analogique envoyée par la girouette. Cette valeur, comprise entre 0 et 1023, va varier en fonction des ILS activés, et donc de la direction de la girouette.

Remarque : Les 4 points cardinaux principaux (N,S,E,O) sont indiqués sur la girouette.

```
void setup() {
Serial.begin(9600);
}
void loop() {
int sensorValue = analogRead(A1);
Serial.println(sensorValue);
delay(100);
}
```

Code Arduino

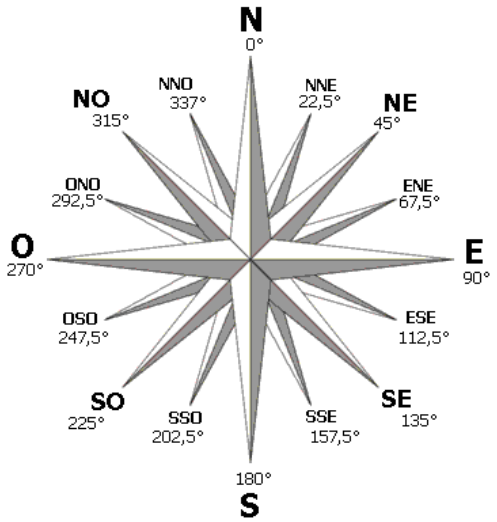
# Code de base de la girouette (2)

En fonction de la valeur relevée, on va pouvoir en déduire la tension, en sachant qu'une valeur analogique de 1023 correspond à la tension maximale, à savoir 5V.

Il faut ensuite faire correspondre la tension à une valeur d'angle. Le tableau ci-contre nous donne cette correspondance.

Enfin, on se reporte à la rose des vents afin de savoir dans quelle direction pointe la girouette.

Rose des vents




| Direction (Degrees) | Resistance (Ohms) | Voltage (V=5v, R=10k) |
|---|---|---|
| 0 | 33k | 3.84v |
| 22.5 | 6.57k | 1.98v |
| 45 | 8.2k | 2.25v |
| 67.5 | 891 | 0.41v |
| 90 | 1k | 0.45v |
| 112.5 | 688 | 0.32v |
| 135 | 2.2k | 0.90v |
| 157.5 | 1.41k | 0.62v |
| 180 | 3.9k | 1.40v |
| 202.5 | 3.14k | 1.19v |
| 225 | 16k | 3.08v |
| 247.5 | 14.12k | 2.93v |
| 270 | 120k | 4.62v |
| 292.5 | 42.12k | 4.04v |
| 315 | 64.9k | 4.78v |
| 337.5 | 21.88k | 3.43v |

Tableau de correspondance entre tension et angle

# Explication circuit

Alimentation : La différence de potentiel entre les bornes + et – de l'alimentation va générer une tension électrique, qui entraîne le déplacement d'électrons (on appelle ce déplacement le courant électrique) et créer un circuit électrique.

Résistance : Lorsque le courant rencontre une résistance, cette dernière va s'opposer à une partie du courant. Plus la résistance est élevée, plus le courant va être faible.

ILS : Lorsque l'ILS est activé, ce dernier va laisser passer le courant, on dit que le circuit électrique est fermé. Dans le cas contraire, on dit que le circuit est ouvert.

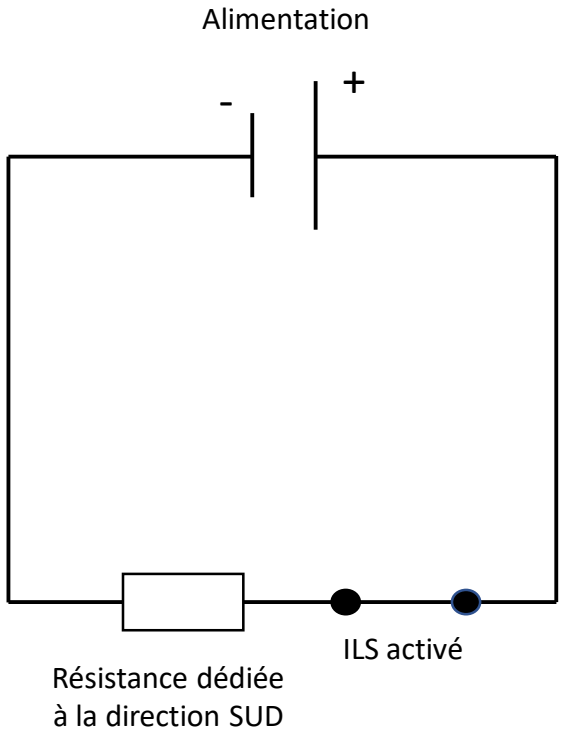



Schéma simplifié du circuit

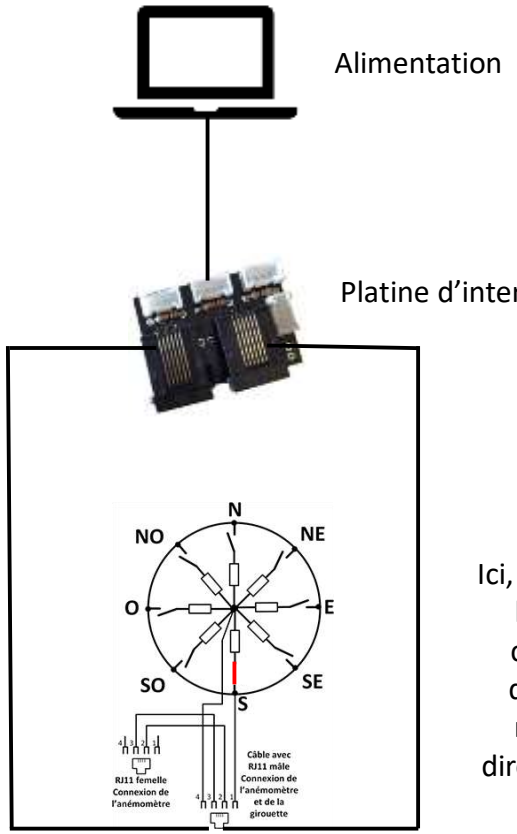



Schéma du circuit

Ici, la girouette pointe dans la direction SUD. L'ILS correspondant à cette direction s'active, et la résistance dédiée à la direction SUD est reliée au circuit.

# Loi d’Ohm (tension, courant, résistance)

La loi d’Ohm est une loi permettant de mettre en relation la valeur d’une résistance (en ohms), le courant qui la traverse (en Ampère) et la tension entre les bornes de la résistance (en volt).

Conventionnellement, ces 3 éléments sont représentés de la manière suivante :

- U : tension, exprimée en volts (symbole : V)
- R : valeur de la résistance, exprimée en ohms (symbole : Ω)
- I : le courant, exprimée en Ampère (symbole : A)

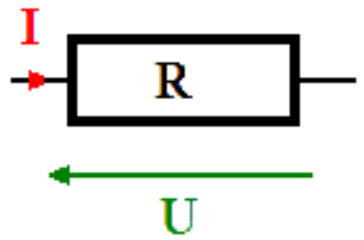



Schéma mettant en lien la résistance, le courant et la tension

**Calcul de la tension :**

$U = R * I$

**Calcul de la résistance :**

$R = \frac{U}{I}$

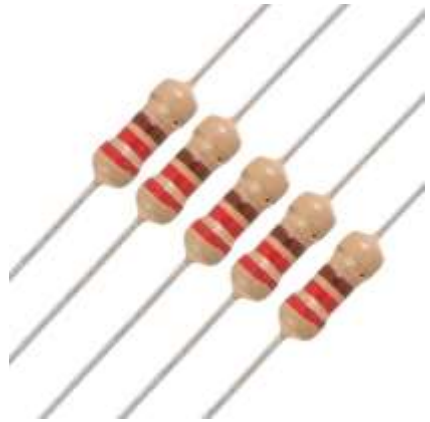
**Calcul du courant :**

$I = \frac{U}{I}$

# Calculer physiquement U, R et I


Un ampèremètre est un appareil de mesure de l'intensité d'un courant électrique dans un circuit


Le voltmètre est un appareil qui permet de mesurer la tension entre deux points



Chaque résistance possède plusieurs anneaux de couleur, qui varient en fonction de la valeur de la résistance. Il est donc possible de connaître la valeur de la résistance en regardant ses anneaux de couleurs

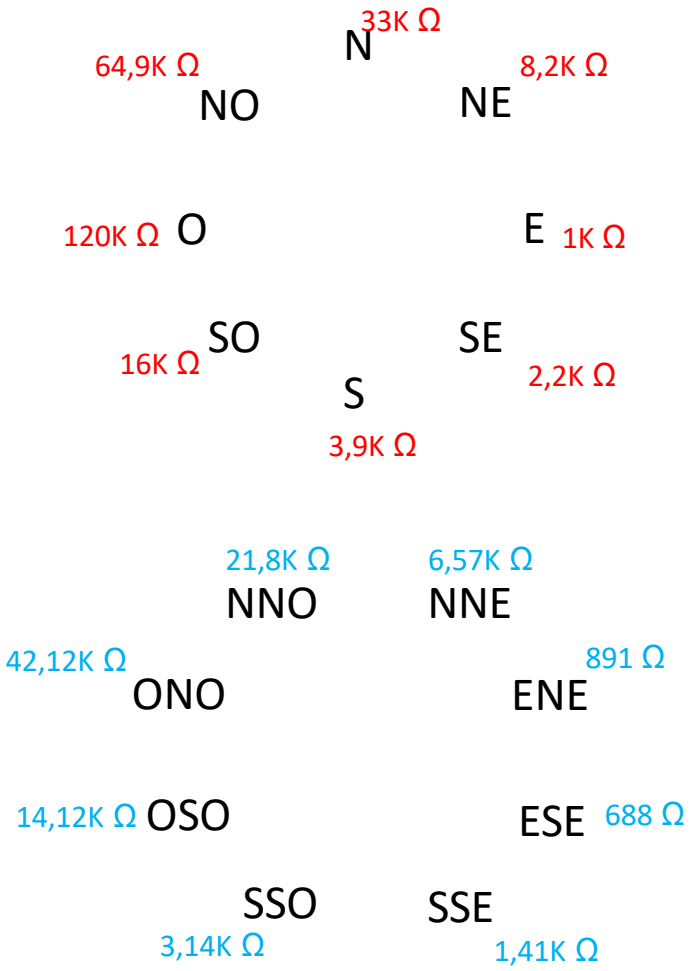
# Trouver les résistances des directions intermédiaires (NNO, NNE, etc…)

La girouette possède 8 ILS reliés à 8 résistances dont la valeur varie selon la direction. Cependant, il est possible d'affiner la direction vers laquelle pointe la girouette (NNO,NNE,ect…). En effet, lorsque la girouette entre deux directions (Nord et Nord-Est par exemple), l'aimant va activer les 2 ILS en même temps. On dit que les résistances sont reliées en parallèle. On utilise alors la formule

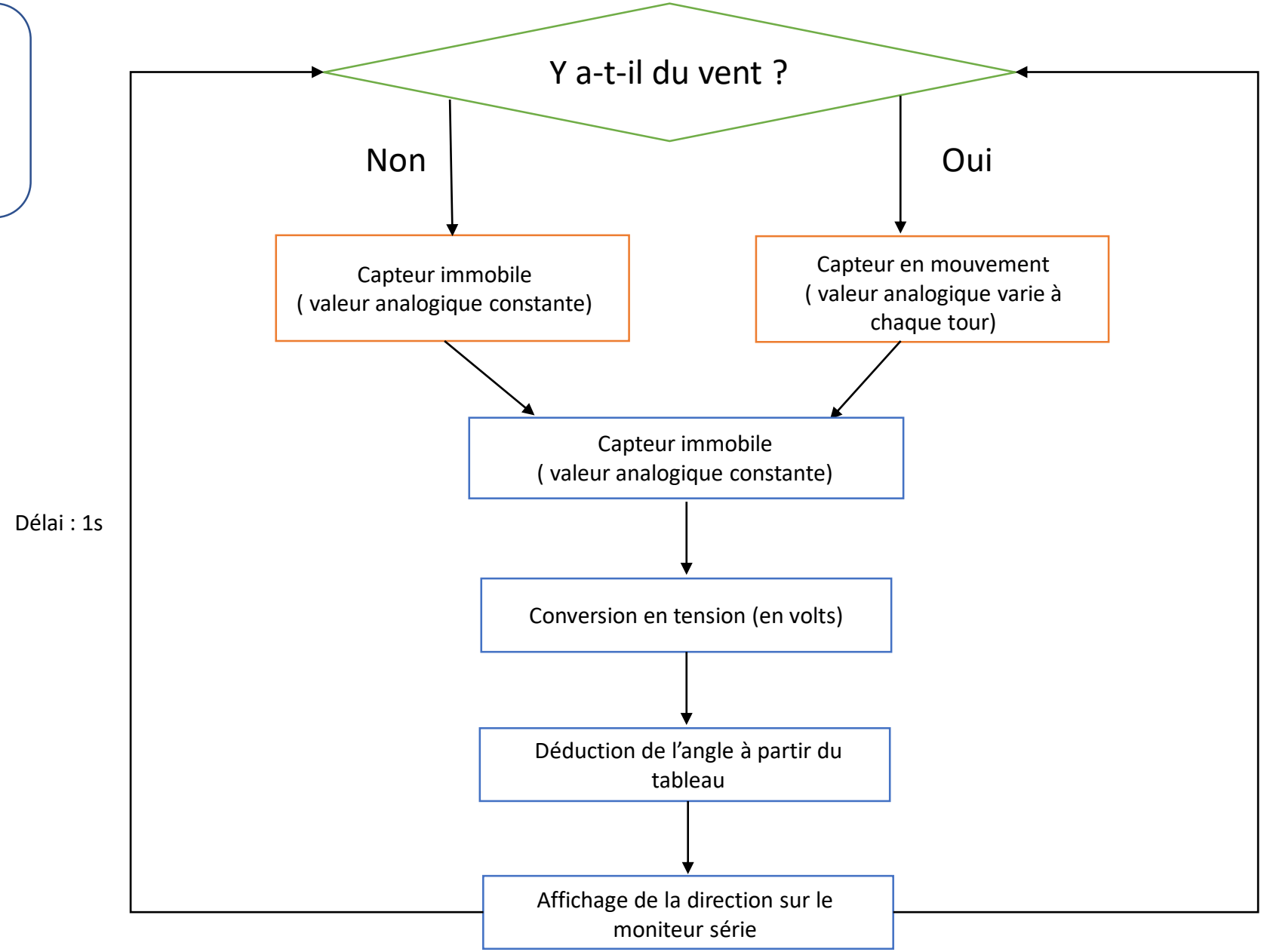
$R = \frac{R1 * R2}{R1 + R2}$ , avec R1 et R2 deux résistances différentes.

Exemple : Si la girouette pointe entre le Nord et le Nord-Est, on a R1 qui vaut 33K (= $33*10^3$) pour le Nord et R2 qui vaut 8,2K. On applique la formule :

$R = \frac{(33 * 10^3) * (8{,}2 * 10^3)}{(33 * 10^3) + (8{,}2 * 10^3)} = 6{,}57 * 10^3$ Ohms



# Algorithme fonctionnel de la girouette



# Code finale de la girouette (Arduino)

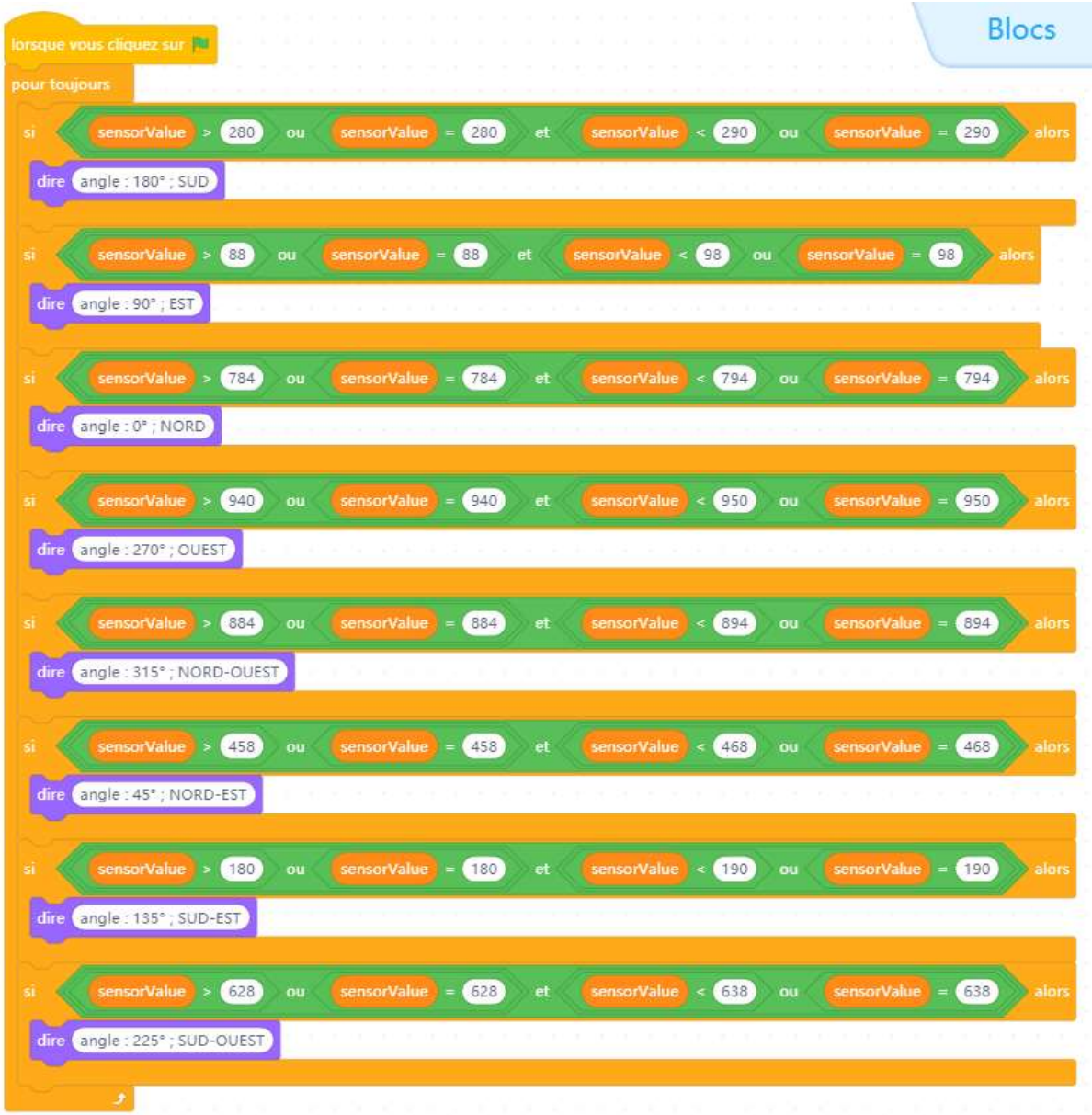
```
void setup() {
Serial.begin(9600);
}
void loop() {
int sensorValue = analogRead(A1);

//angle 180, SUD
if (sensorValue >=280 && sensorValue <= 290)
{
  Serial.print("angle : 180°; ");
  Serial.println("direction : SUD");
}

if (sensorValue >=88 && sensorValue <=98)
{
  Serial.print("angle : 90°; ");
  Serial.println("direction : EST");
}

if (sensorValue >=784 && sensorValue <=794 )
{
  Serial.print("angle : 0°; ");
  Serial.println("direction : NORD");
}
if (sensorValue >=940 && sensorValue <=950 )
{
  Serial.print("angle : 270°; ");
  Serial.println("direction : OUEST");
}
if (sensorValue >=884 && sensorValue <=894 )
{
  Serial.print("angle : 315°; ");
  Serial.println("direction : NORD-OUEST");
}
if (sensorValue >=458 && sensorValue <=468 )
{
  Serial.print("angle : 45°; ");
  Serial.println("direction : NORD-EST");
}
if (sensorValue >=180 && sensorValue <=190 ) //
{
  Serial.print("angle : 135°; ");
  Serial.println("direction : SUD-EST");
}
if (sensorValue >=628 && sensorValue <=638 ) //
{
  Serial.print("angle : 225°; ");
  Serial.println("direction : SUD-OUEST");
}

delay(1000);
}
```

*Explication du code :*

Ici, on relève toutes les secondes la valeur analogique (*sensorValue*) envoyée par la girouette. Comme ces valeurs varient légèrement, nous avons pris des tranches larges (de 10), afin de couvrir toutes les valeurs possibles pour une même direction.
En fonction de la valeur analogique, on va afficher l'angle et la direction de la girouette.

# Code finale de la girouette (mBlock)


Code de l'arduino




Code de l'objet

# Fonctionnement du pluviomètre

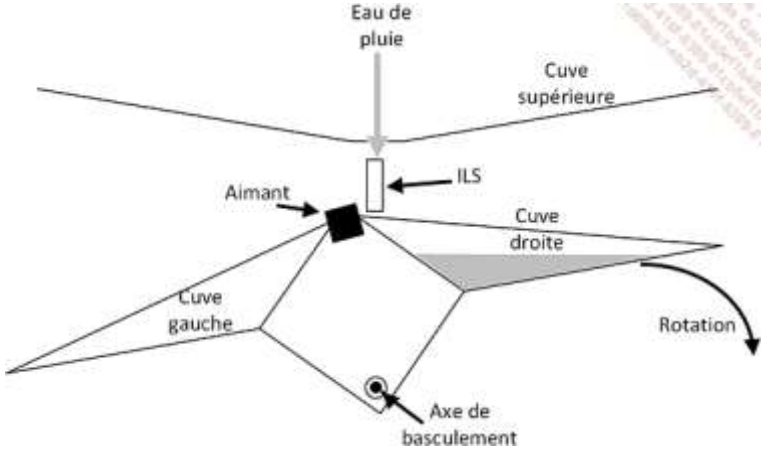
Lorsque de l'eau entre dans le capteur, un des côtés de la cuve inférieure du pluviomètre va se remplir (soit celle de droite, ou celle de gauche). Sur la photo ci-contre, c'est la cuve de gauche qui va se remplir.

Lorsque la cuve se remplit, le poids de l'eau augmente jusqu'à faire basculer la partie centrale. C'est alors la cuve droite qui va se retrouver sous le trou de la cuve supérieure et se remplir.

Le pluviomètre comporte un ILS, ainsi qu'un aimant sur la partie centrale de la cuve inférieure. A chaque basculement de la cuve, l'aimant va déclencher l'ILS (on dit que le pluviomètre passe à l'état HAUT).

On estime que l'ILS se déclenche lorsqu'il tombe 0,2794 mm de pluie dans le capteur.

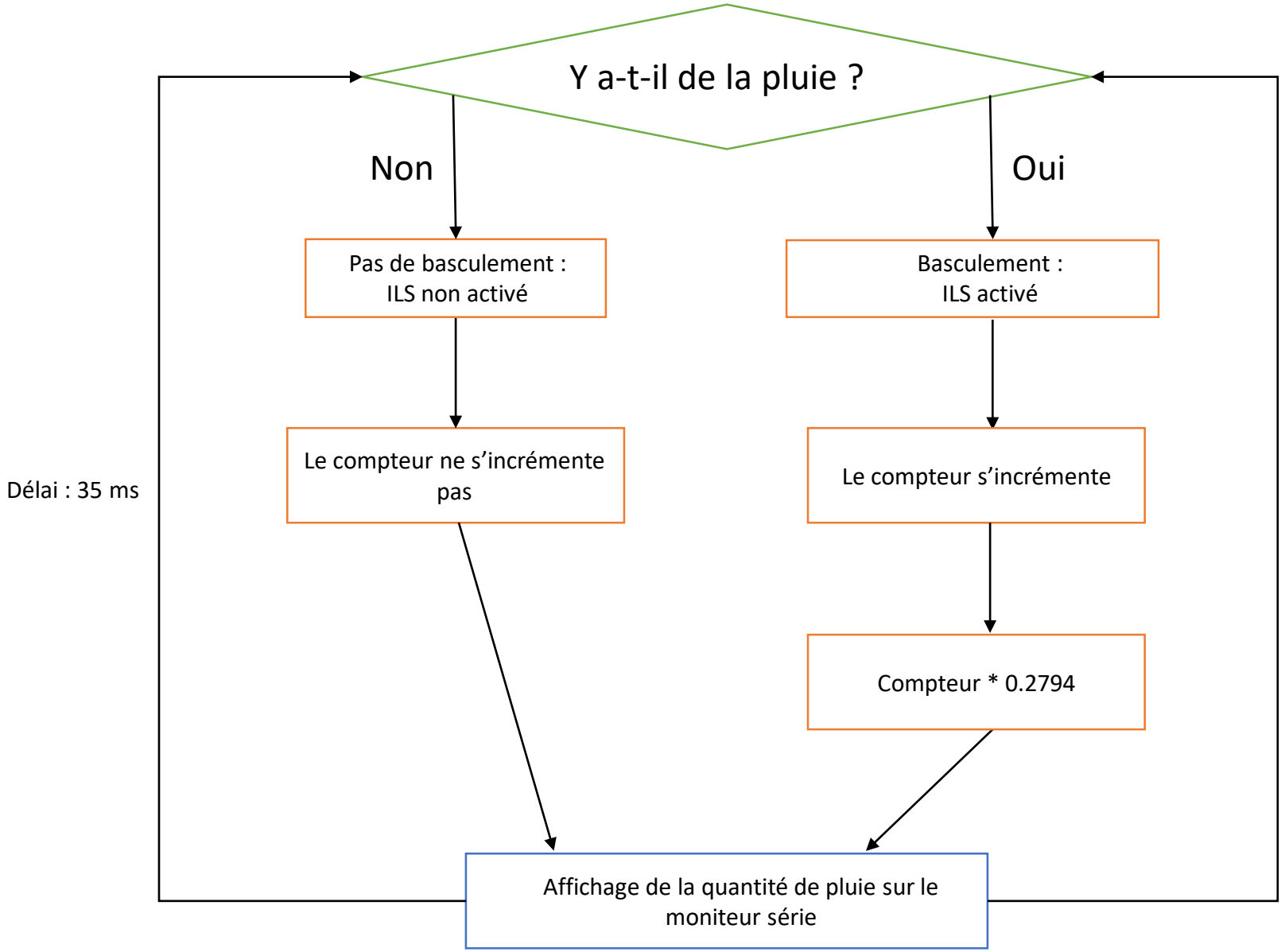
Pour calculer la quantité de pluie tombé, il suffit donc de calculer le nombre d'impulsions (= nombre de fois où l'ILS se déclenche) dans un temps donné.



# Algorithme fonctionnel du pluviomètre

Rappel :
On estime que l'ILS se déclenche lorsqu'il tombe 0,2794 mm de pluie dans le capteur.

Donc la quantité de pluie = nombre de basculements (compteur) * 0.2794



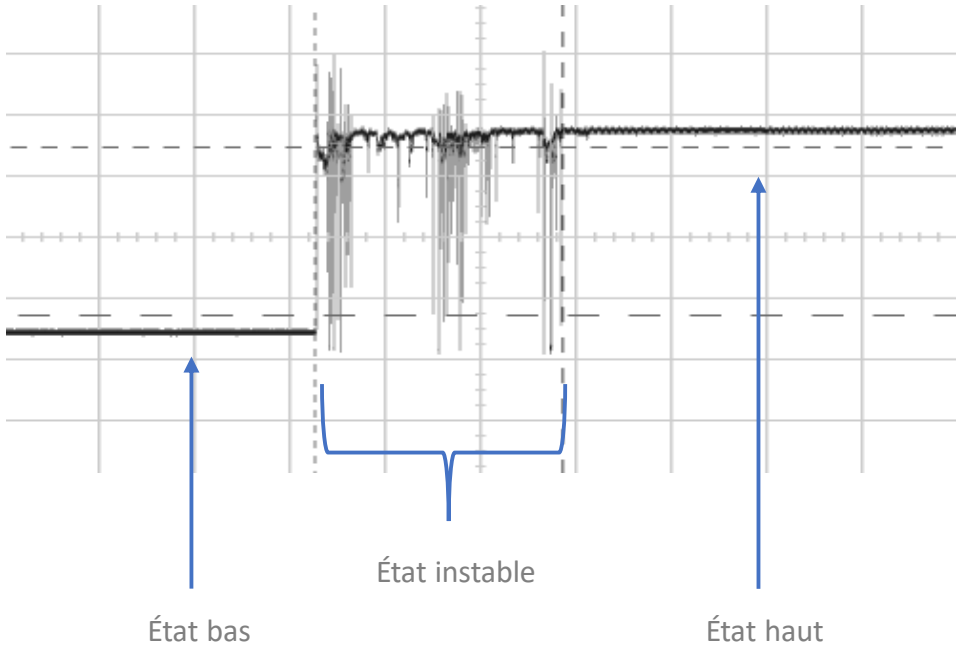
# Pluviomètre : micro-rebondissements

Au moment où l'ILS va s'activer, le pluviomètre va passer par un « état instable » (voir image ci-contre).

La raison est que l'exécution par Arduino est si rapide que ça va détecter des micro-rebondissements. Ainsi, le capteur va passer par plusieurs changements d'état HAUT – BAS avant de se stabiliser à l'état HAUT.

Ces micro-rebondissements vont nous poser problèmes pour le programme avec le pluviomètre : en effet, pour calculer la quantité de pluie, il va falloir compter le calculer le nombre de fois où l'ILS se déclenche, c'est-à-dire le nombre de fois où le capteur passe à l'état HAUT. Cependant, avec ces micro-rebondissements, le compteur va s'incrémenter plusieurs fois par basculements.

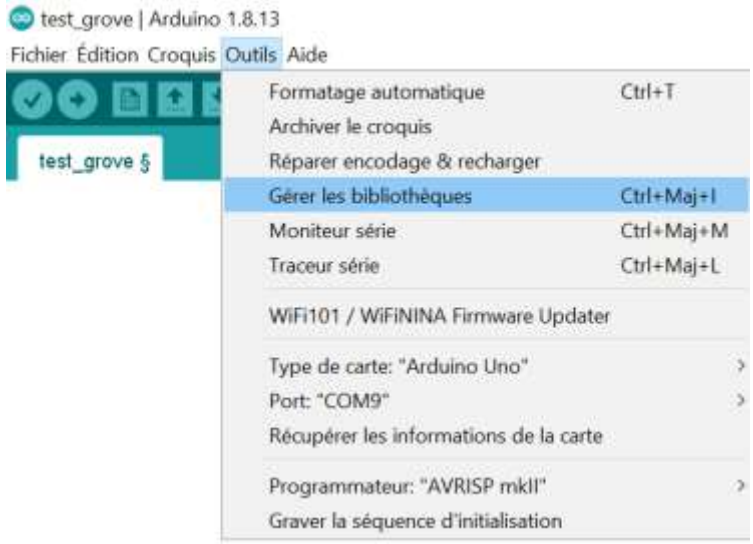
Nous avons 2 solutions : modifier le montage afin de lisser la courbe et enlever les micro-rebondissements (avec une résistance pullup), ou modifier notre programme afin de ne pas prendre en compte l'état instable du capteur.



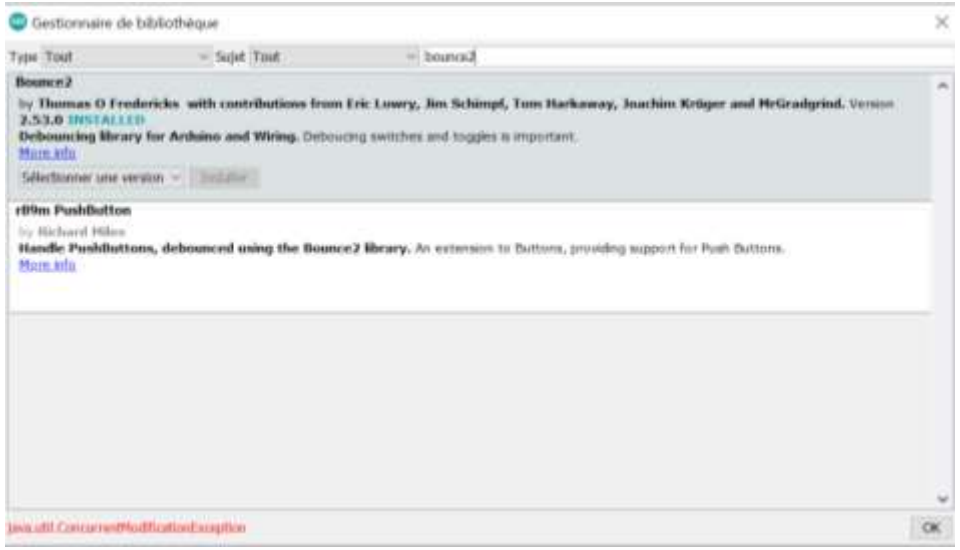
# Pluviomètre : la fonction Bounce

Dans notre cas, nous allons utiliser la fonction **Bounce2**, qui est une logithèque conçue pour éliminer les rebondissements et pour faciliter l'utilisation des boutons. Il faut donc installer cette bibliothèque sur votre IDE Arduino si vous ne l'avez pas déjà (voir photo ci-dessous).

La fonction Bounce permet de mettre un temps de latence lors de la mesure d'un état.




Allez dans Outils, et cliquez sur Gérer les bibliothèques




Tapez Bounce2 dans la barre de recherche, et installez la bibliothèque

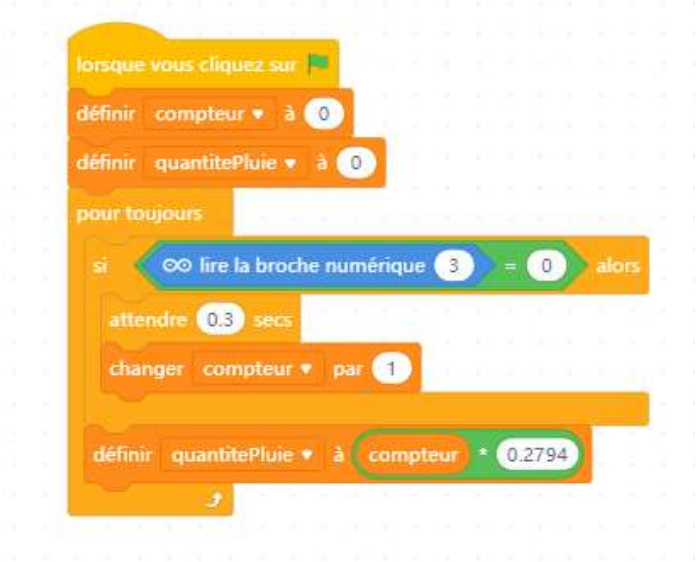
# Code du pluviomètre (Arduino)

```
#include <Bounce2.h>

#define INTER 3

// Créer (instancier) un objet Bounce
Bounce inter;
int compteur=0;

void setup()
{
  Serial.begin(9600);
  // Configure la broche avec la résistance de tirage
  pinMode(INTER, INPUT_PULLUP);
  // Attache l'objet 'inter' à la même broche
  inter.attach(INTER);
  inter.interval(5); // temps de latence entre deux mesures d'état en ms
}

// Variable qui va contenir l'état courant du bouton
// On commence avec -1 qui est un état inconnu...
int courant = -1;
```

```
void loop()
{

  // Mise à jour de l'état de inter : bouton pressé ou non ?
  inter.update();

    Serial.print("La quantité d'eau est de ");
    Serial.print(compteur*0.2794);
    Serial.println(" mm");

  // Récupération de l'état du bouton
  int valeur = inter.read();

  // Si la valeur n'a pas changé depuis la dernière fois, ne rien faire.
  if (valeur == courant)
    return;

  // Agir en fonction de l'état du bouton
  if ( valeur == LOW)
    {
      compteur++;
    }
  courant = valeur;

}
```

# Code du pluviomètre (mBlock)

Dans le code mBlock, nous plaçons un délai de 0.3 secondes au moment de la détection de l'activation de l'interrupteur. Cela permet de ne pas prendre en compte le début de la mesure, et donc d'éliminer les micro-rebondissements éventuels, comme pour la fonction Bounce.




Code de l'Arduino


Code de l'objet