Utilisation de la carte d'acquisition/commande ARDUINO
Utilisation des CNA /CAN en entrée/sortie
Caractérisation du caractère gaussien ou pas d'un capteur

1. Cartes d'acquisition à microcontrôleur

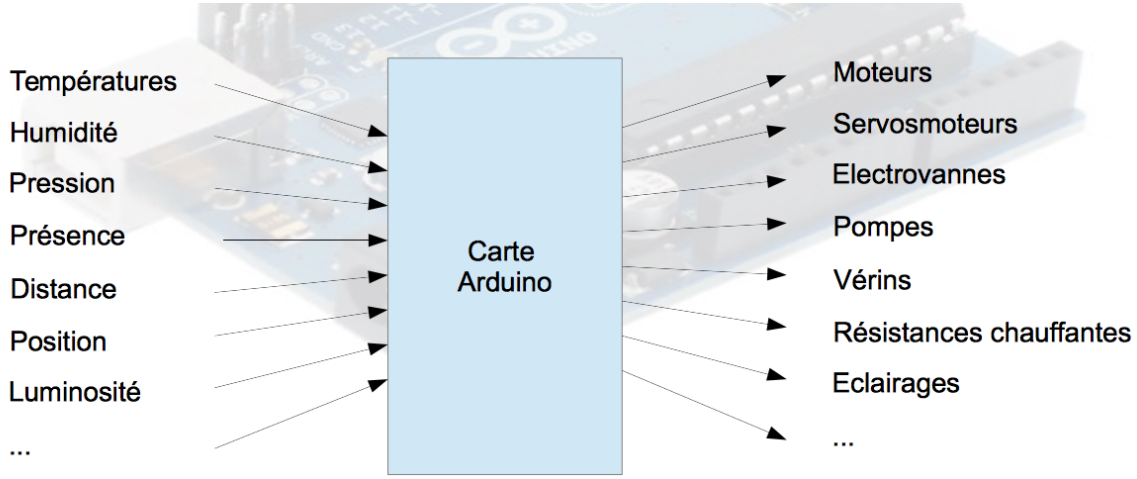
Nous verrons ensuite, en plus de l'Arduino, les BBCMicrobit, les ESP32, les Pi Pico ou Pi W etc.
Voir la page `http://psiamiens.site/tipe.html`

2. Carte ARDUINO à microcontrôleur Atmega328p

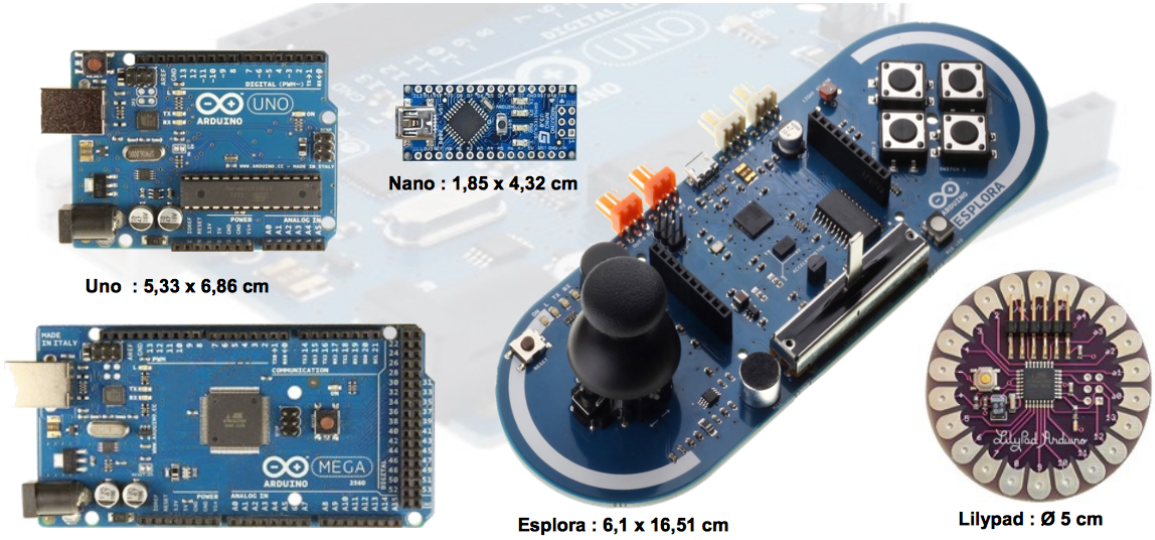
Rappels : cette carte est une carte micro-contrôleur (noté μC) qui permet de gérer des entrées analogiques ou numériques et des sorties numériques en tension, ainsi qu'un "semblant" de tension analogique en sortie (PWM).
Données : `https://fr.wikipedia.org/wiki/Arduino`
Entrées : ANALOG IN, 6 entrées numérotées A0 à A5
Entrées/Sorties : DIGITAL, 14 bornes notées 0 à 13, masse notée GND, les bornes 0 et 1 étant réservées pour la communication UART série avec les bornes TX et RX.
Sorties PWM : bornes 3, 5, 6, et 9, 10, 11
PWM = Pulse Width Modulation et en français cela donne Modulation à Largeur d'Impulsion (MLI). La PWM est en fait un signal numérique qui, à une fréquence donnée, a un rapport cyclique α qui change et en résumé : on peut simuler sur ces bornes une valeur analogique réglable entre 0 et 5V grâce à α

L'alimentation et la connexion au PC se fait à l'aide de la borne USB-B.
On peut également alimenter avec une source externe comme une pile.
Une fois un programme chargé dans la (petite) mémoire de la carte, elle devient autonome.

3. Pourquoi un micro-contrôleur ?



L'imagination est la seule limite.



Des capteurs


Smoke Sensor


Line Finder

Photorésistance (CdS)


Capteur Infrarouge émetteur récepteur


Capteur Piezoelectrique


Capteur de Poussieres - PPD42NS


Capteur de qualité d'air - Grove


Détecteur d'alcool MQ303A


Détecteur de gaz (MQ2)


Capteur de temperature - Grove


Capteur de temperature et d'humidité


Thermographe infrarouge - Grove


TMP36 - Capteur de Temperature


Capteur de mesure à ultrasons URM37 V3.2


Capteur de mouvement à infrarouge


Accéléromètre 3 axes +/-8g ADXL362


Gyroscope 3 axes (Grove)


Capteur de débit d'eau


Capteur de débit d'eau - G3/4


Capteur Tactile - Grove


Capteur Touch


Module RFID 125Khz - UART


Module RFID 13.56MHz ISO/IEC 14443 type a


Capteur de poids - 50kg


Capteur de son


Carte magnétique LoCo


Compteur Geiger


Lecteur - Encodeur de cartes magnétiques


Scanner code barre USB Série


GPS Bee kit (with Mini Embedded Antenna)


Grove - Barometer Sensor


Grove - Moisture Sensor


Photo interrupter (OS25B10)


Grove - RTC


Grove - Serial Camera


Omnidirectional Sensor


Grove - Electricity Sensor


Grove - Hall Sensor


Non-invasive AC current sensor (100A max)

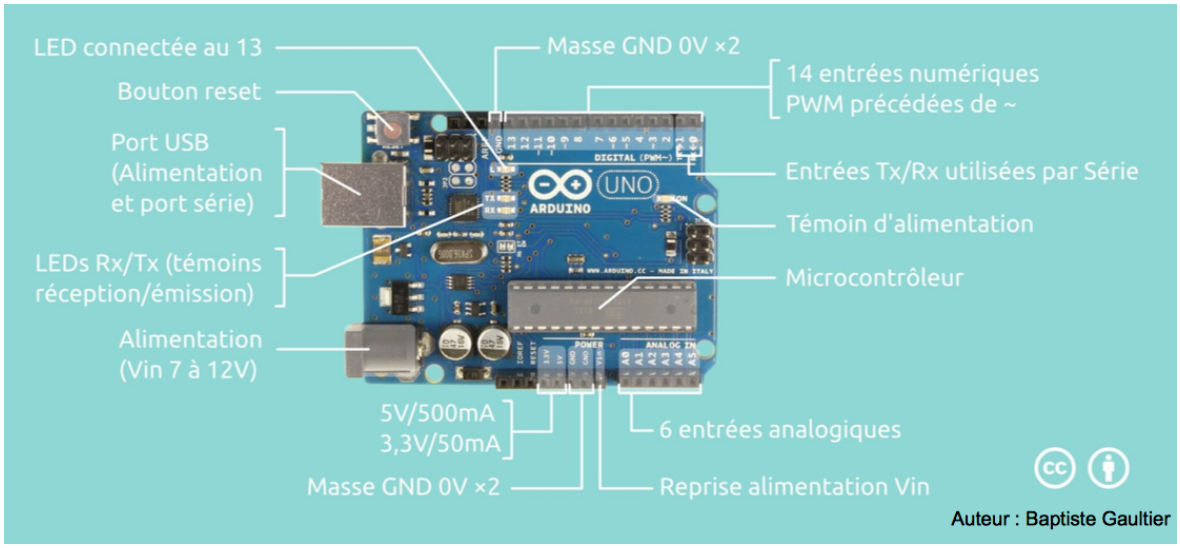
Non-invasive AC current sensor (30A max)

Waterproof Metal Pushbutton with Blue LED Ring

These chrome-plated metal buttons are rugged and waterproof and look real good while doing it! Simply drill a 16mm hole into any material up to 1/2"...
ID : 481

Rotary Encoder + Extras

This rotary encoder is the best of the best. its a high quality 24-pulse encoder, with detents and a nice feel. It is panel mountable for placement...
ID : 377



Notre Arduino : le Uno

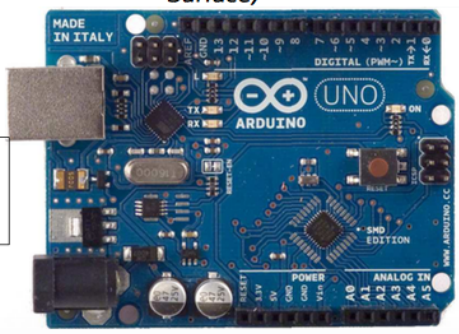
- Micro contrôleur : ATmega328
- Tension d'alimentation interne = 5V
- tension d'alimentation (recommandée)= 7 à 12V, limites =6 à 20 V
- Entrées/sorties numériques : 14 dont 6 sorties PWM
- Entrées analogiques = 6 (avec une résolution de 10 bits => 1024 valeurs différentes)
- Courant max par broches E/S = 40 mA
- Courant max sur sortie 3,3V = 50mA
- Mémoire Flash 32 KB dont 0.5 KB utilisée par le bootloader*
- Mémoire SRAM 2 KB
- mémoire EEPROM 1 KB
- Fréquence horloge = 16 MHz
- Dimensions = 68.6mm x 53.3mm

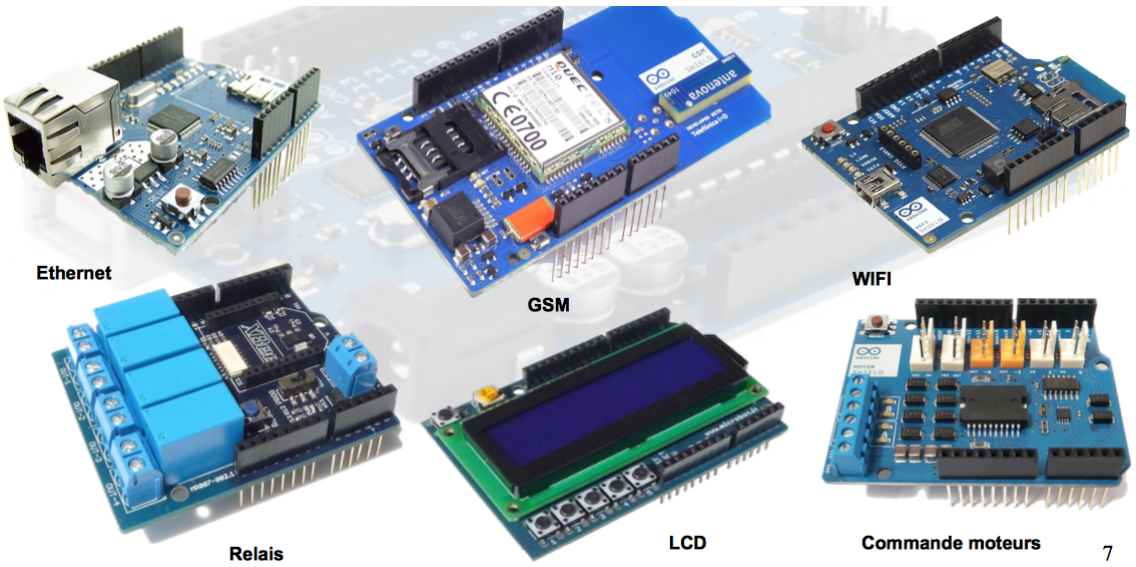
La carte s'interface au PC par l'intermédiaire de sa prise USB.

La carte s'alimente par le jack d'alimentation (utilisation autonome) mais peut être alimentée par l'USB (en phase de développement par exemple).

*Bootloader : un petit programme chargé sur le microcontrôleur. Il permet de charger le code sans programmateur. Il est activé quelques secondes lorsque la carte est « *resetée* ». Ensuite, il démarre le sketch (programme) qui a été chargé sur le microcontrôleur.

Arduino Uno SMD (ou CMS en français pour Composant Monté en Surface)





4. Introduction aux mesures grâce à un micro-contrôleur

Nous avons à notre disposition :

4. 1 des entrées "analogiques" avec un CAN de 5V max et un faible ampérage (danger de destruction)
4. 2 des sorties numériques dites "tout ou rien" 0 ou 5V
4. 3 des sorties "analogiques" avec simulation de tension analogique entre 0 et 5V

5. Entrée : lecture de donnée analogique

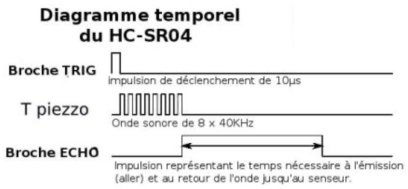
La tension analogique extérieure est convertie sur les 6 entrées `A0` à `A5` par un CAN.
La précision résulte d'un codage sur **10 bits** pour l'Arduino
La récupération et le traitement des données analogiques converties peut se faire de différentes façons :

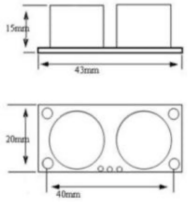
- envoi sur la fenêtre de communication de l'IDE de l'Arduino de l'ordinateur, et récupération des données par copier-coller dans un tableur de type `LibreOffice`
- écriture dans un fichier grâce à une carte supplémentaire comportant un lecteur de carte SD, et lecture du fichier `.txt` avec un programme `Python`
- capture par un programme Python des données envoyées par le port Série grâce à la bibliothèque `pySerial`
- Commande directe de l'Arduino par un programme Python grâce à la bibliothèque `pyFirmata`

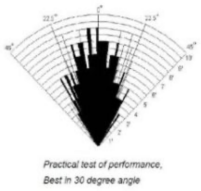
6. Cas d'un capteur ultrasonore

On utilise le composant "Capteur de distance à ultrason" HC-SR04.
1. Le module émet une onde sonar composée d'une série de 8 impulsions à 40 kHz . Un son à 40 KHz est inaudible pour l'oreille humaine capable de percevoir des sons entre 20 Hz et 20 KHz
2. En utilisant la broche Trig, vous envoyez un signal pour activer le déclenchement de l'impulsion sonar.
3. La sortie ECHO passe au niveau haut durant toute la période où l'onde voyage vers l'objet et revient après avoir été réfléchit par se dernier. La sortie est la mesure du temps durant lequel le signal de sortie est au niveau haut, donc de l'aller-retour. En multipliant par la vitesse du son, on obtient le double de la distance à l'objet.

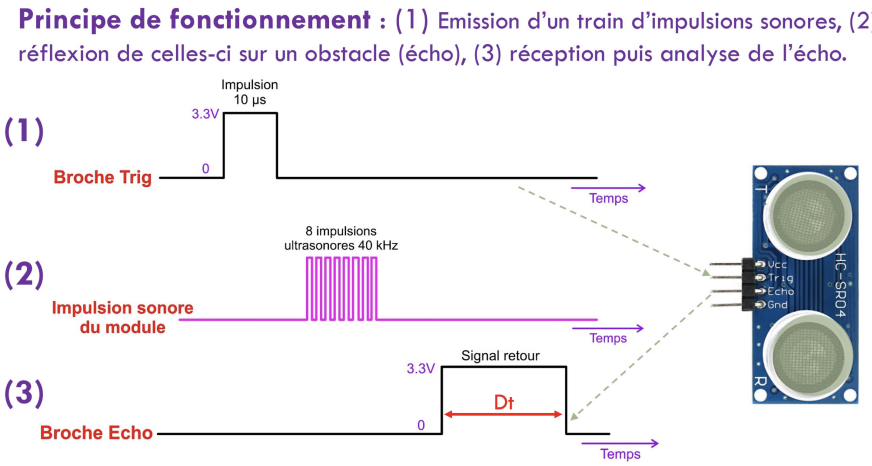
Branchement
- VCC - alimentation 5 V
- Trig - Entrée permettant de déclencher l'impulsion
- Echo - Sortie, permet de déterminer la réception de l'écho
- GND - La masse







Caractéristiques techniques
- Tension de fonctionnement : 5 V continu
- Consommation (en fonctionnement) : 15 mA
- Fréquence de fonctionnement : 40 Hz
- Distance Minimale : 2 cm
- Distance maximale : 400 cm(4 m) (dans de bonnes conditions)
- Angle de mesure : 15 degrés
- Signal d'entrée - trig : déclenchement de la mesure : impulsion $10\mu s$ TTL
- Signal de sortie - echo : impulsion lorsque l'écho est reçu : signal TTL (dépend de la distance mesurée)
- Dimensions : 45×20 x 15 mm



Ce capteur est une "boîte noire" donc on va chercher à le caractériser mieux que simplement utiliser les programmes que l'on voit sur l'Internet. Il nous faut un oscilloscope et une plaquette de connexions.

- Utilisez le programme TP_arduino_ultrasons_1.ino Faire les mesures et vérifier la distance "moyenne" affichée pour votre cible placée devant le HC-SR04.

```
/* Capteur de distance Ultrason HC-SR04:
VCC sur Arduino 5v
GND sur Arduino GND
Echo sur Arduino broche 7
Trig sur Arduino broche 8
*/

#define echoPin 7 // broche Echo (mesure)
#define trigPin 8 // broche Trigger (declenchement de la mesure)
#define LEDPin 13 // LED de la carte Arduino (branché sur la broche 13)
long duree; // Durée utilisée pour calculer la distance
float distance ;
float vitesse = 0.340 ; // 340 m/s donc 340*1000/1000000 mm/microseconde

//-----------------------------------------------------------------------
void setup() {
```

```
        // Activer la communication série à 9600 bauds
        Serial.begin (9600);
        // Activer les broches, définies en sortie pour l'une, en entrée pour l'autre
        pinMode(trigPin, OUTPUT);
        pinMode(echoPin, INPUT);
        pinMode(LEDPin, OUTPUT); // activer la LED sur la carte (si nécessaire)
}
//-----------------------------------------------------------------------
// ENSUITE : Partie du code continuellement exécuté (boucle "loop")
// Son but est d'effectuer un cycle de détection pour déterminer
// la distance de l'objet le plus proche (par réflexion de
// l'onde sur ce dernier)
//
void loop() {
        // Envoi une impulsion de 10 micro seconde sur la broche "trigger" :
        //    0 puis 1 puis 0
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);

        // Attend que la broche Echo passe au niveau HAUT
        // retourne la durée en microsecondes dans la variable durée
        duree = pulseIn(echoPin, HIGH);

        Serial.println("Début des mesures :");

        //Calculer la distance (en mm, basé sur la vitesse du son).
        distance = duree*vitesse/2.0;

        // affichage sur le port série, donc dans la fenêtre Com sur l'ordinateur
        Serial.println(distance,1);  // un chiffre après la virgule

        //Attendre 50ms avant d'effectuer la lecture suivante.
        delay(50);
}
```

- Exploitation statistique des enregistrements

  1— exploitation avec un tableur :

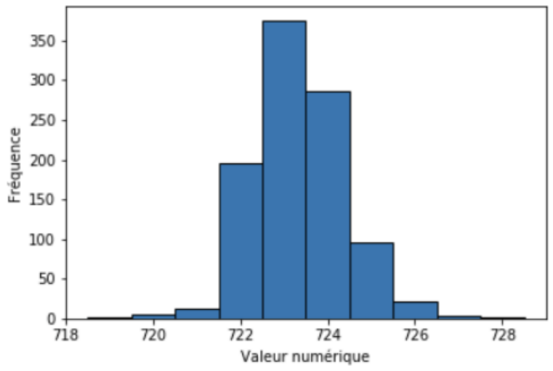
  Copier les données de mesure dans la fenêtre du "Moniteur série" d'Arduino et les coller dans un fichier .csv puis ouvrir ce fichier avec le tableur.

  On peut ensuite faire le traitement.

  voir `https://www.youtube.com/watch?v=XzdKfRTLdMc&t=228s`

  2— Exploitation avec PYTHON :

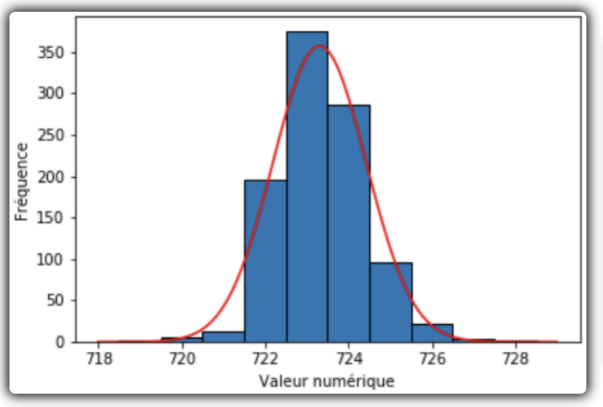
```
# l'intervalle des mesures doit être adapté avec vos valeurs
plt.hist(mesures, range= [718.5, 728.5], bins=10, edgecolor = 'black')
plt.xlabel("Valeur numérique")
plt.ylabel("Fréquence")
plt.show()
```



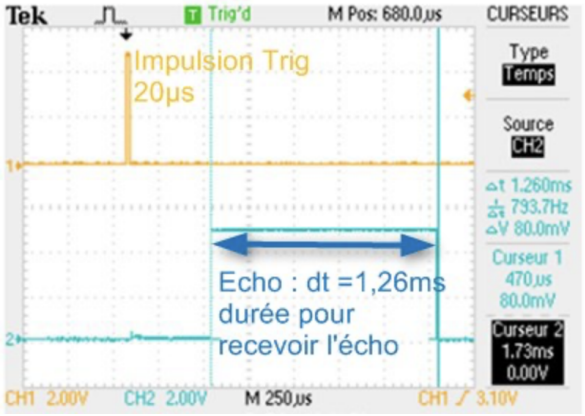
Pour superposer la loi normale donnée par :

$$f : x \mapsto \frac{A}{\sigma_x\sqrt{2\pi}} e^{\frac{-(x-\bar{x})^2}{2\sigma_x^2}}$$

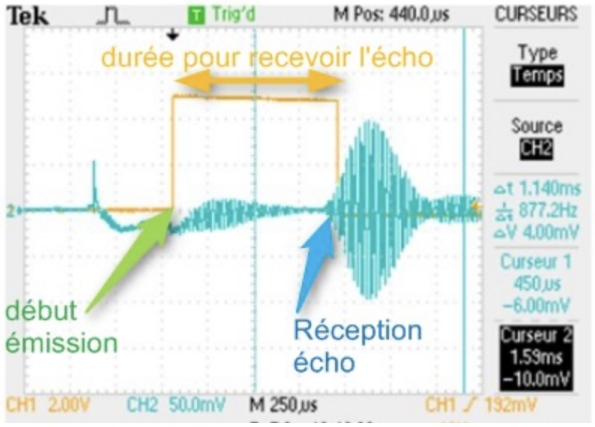
```
import numpy as np

# La fonction de la loi normale
def gaussienne(x, A, moyenne, ecarttype):
return A / (ecarttype *np.sqrt(2*np.pi)) * np.exp(-(x-moyenne) **2 / (2*ecarttype**2))

#Intervalle de définition
x = np.linspace(min(mesures), max(mesures), 100)
y = Gaussienne(x, 1000, moyenne, std_m)

#Le graphique pour un intervalle choisi par rapport aux mesures :
plt.hist (mesures, range= [718.5, 728.5], bins=10, edgecolor = 'black')
plt.xlabel("Valeur numérique")
plt.ylabel("Fréquence")
plt.plot(x, y, 'red')
plt.show()
```



- Visualiser sur l'oscilloscope les signaux des pattes Trig et Echo. Placer les curseurs de mesure de l'écran de l'oscilloscope afin de mesurer ΔT la durée de l'impulsion Echo d'une manière assez précise. Mesurer la durée de Trig. Pensez à prendre une capture d'écran de votre oscilloscope. Modifier le programme pour visualiser la durée de l'écho en microsecondes et comparez à la mesure sur l'oscilloscope.

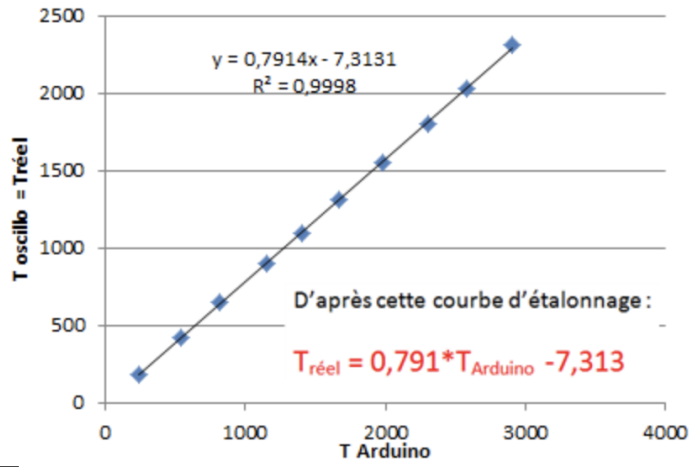


- Visualiser la tension reçue par le récepteur en synchronisme avec le signal Echo.



- Pour plusieurs distances, mesurer avec un réglet en même temps qu'avec le capteur. Faire un tracé comparatif de ces deux mesures.
  On peut également, pour des distances fixes, mesurer Δt ECHO sur l'oscilloscope ou celui rendu par la fonction `PulseIn` :

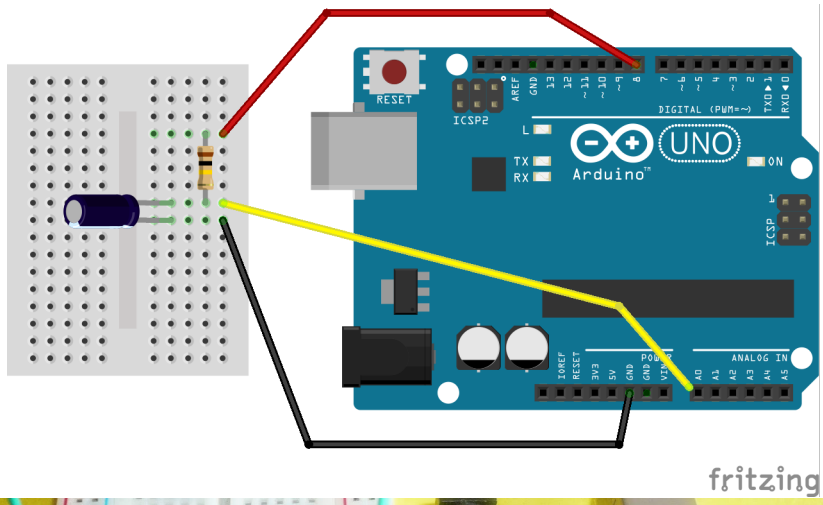
Référence Règle en cm	5	10	15	20	25	30	35	40	45	50
dt en μs (Arduino)										
dt en μs (Oscilloscope)										

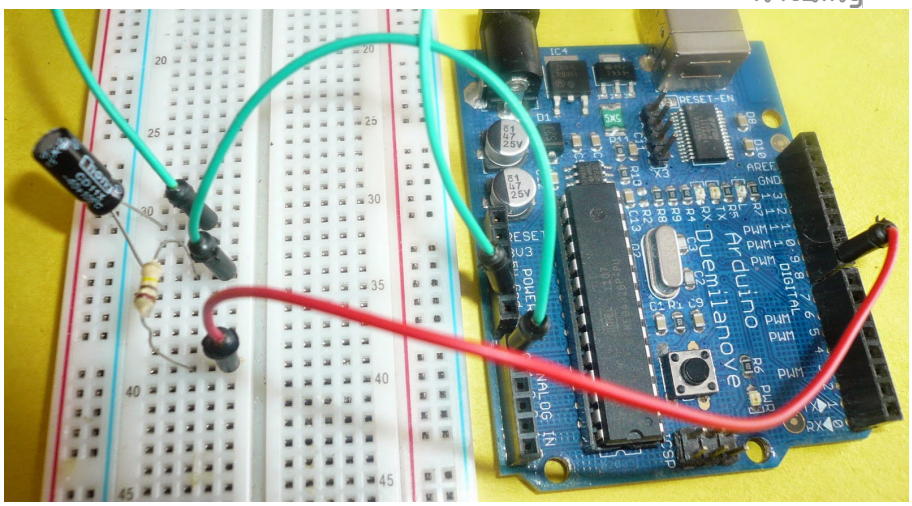


Etude de l'erreur de mesure due à la fonction `pulseln()` : les deux mesures devraient être proportionnelles. Est-ce le cas ? A-t-on une erreur systématique ou aléatoire ? Peut-on corriger cette erreur ?

7. Cas de la charge d'un condensateur.

Par commodité, notre choix sera **une fenêtre de communication par l'IDE**.
La tension analogique extérieure est convertie sur les 6 entrées `A0` à `A5` par un CAN.
La précision résulte d'un codage sur **10 bits** pour l'Arduino et la vitesse d'acquisition correspond à une durée de conversion (de conversion) de 100 μs=0,1ms



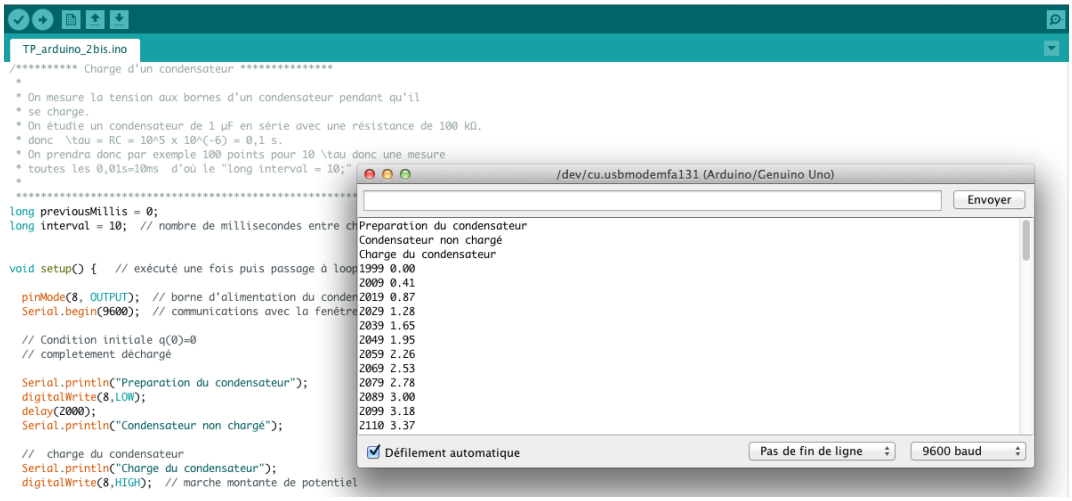


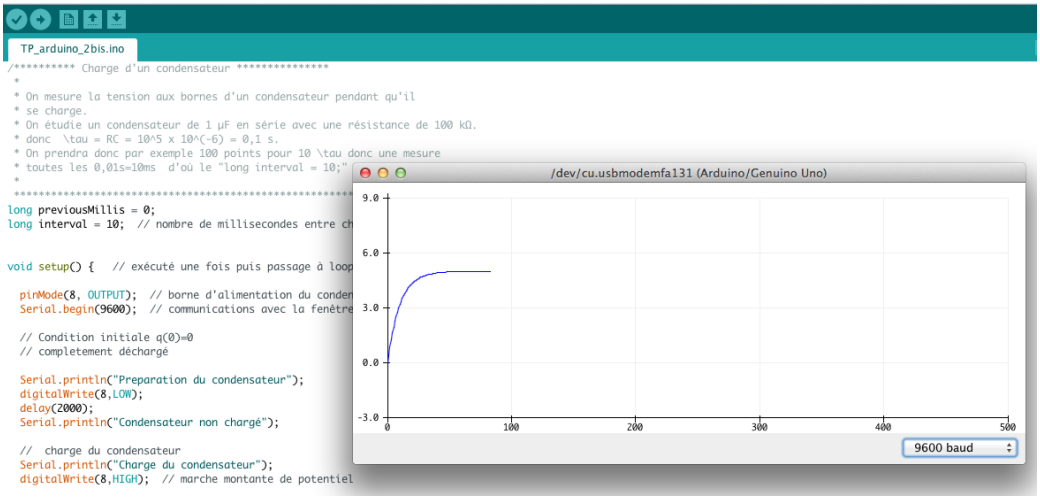
▶ Câbler le montage. On étudie un condensateur de 1 μF en série avec une résistance de 100 $k\Omega$
▶ Utiliser le code suivant. Des questions sont incluses.

```
/********** Charge d'un condensateur ***************
* On mesure la tension aux bornes d'un condensateur pendant qu'il
* se charge. On étudie un condensateur de 1 $\mu F$ en série avec une
* résistance de 100 $k\Omega$ donc  $\tau$ = RC = 10^5 x 10^(-6) = 0,1 s.
* On prendra donc par exemple 100 points pour 10 \tau donc une mesure
* toutes les 0,01s=10ms  d'où le "long interval = 10;"
*************************************************************/
long previousMillis = 0;
long interval = 10;  // nombre de millisecondes entre chaque mesure

void setup() {   // exécuté une fois puis passage à loop()
pinMode(8, OUTPUT);  // borne d'alimentation du condensateur
Serial.begin(9600);  // communications avec la fenêtre "série"
// Condition initiale q(0)=0
// completement déchargé
```

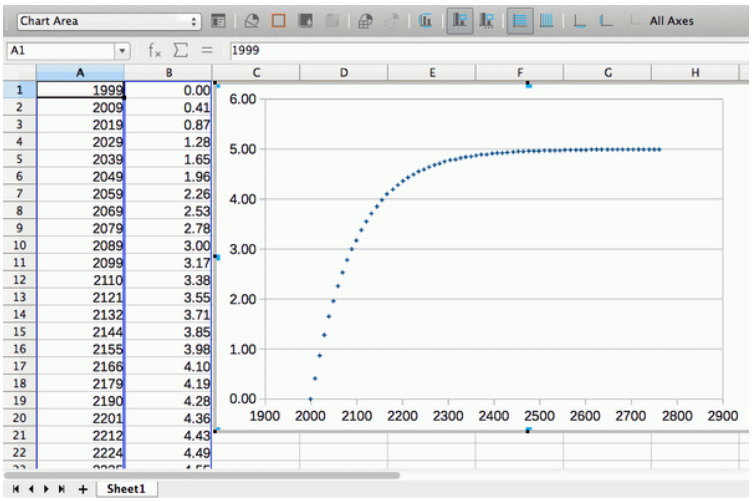
```
16 Serial.println("Preparation du condensateur");
17 digitalWrite(8,LOW);
18 delay(2000);          // QUESTION : pourquoi cette ligne ?
19 Serial.println("Condensateur non chargé");
20 //  charge du condensateur
21 Serial.println("Charge du condensateur");
22 digitalWrite(8,HIGH);  // marche montante de potentiel
23 }
24
25 void loop() {  // exécuté en boucle
26 unsigned long currentMillis = millis(); // QUESTION : fonctionnement?
27 int tension_int;
28 if(currentMillis - previousMillis >= interval) {
29 // QUESTION : à quoi sert ce test ?
30 previousMillis = currentMillis;
31 tension_int = analogRead(A0);
32 if (tension_int < 1022){  // QUESTION : pourquoi 1022 et pas 1024?
33 // méthode 1 : envoi des données sur le port série
34 Serial.print(currentMillis);
35 Serial.print(" ");
36 Serial.println(tension_int*5.0/1023.0);// QUESTION : expliquez le calcul
37 }
38 else{  // completement chargé
39 Serial.println("Chargé");
40 digitalWrite(8,LOW);  // désactivation de l'alimentation
41 // boucle infinie de fin :
42 while(1)   // QUESTION : à quoi sert cette boucle ?
43 {}
44 }  }  }
```

Dans le "moniteur série" ou le "traceur série" (modifier le code), on obtient les valeurs mesurées :





► Récupérer par copier-coller les données affichées dans le ”moniteur série”.
► Ouvrir un tableur comme LibreOffice
► Tracer la courbe de charge du condensateur. Comment en déduire le temps de charge τ ?
► Comment pourrait-on tracer une fonction linéaire permettant d'en déduire τ facilement ? Et quel est l'incertitude sur la mesure ?

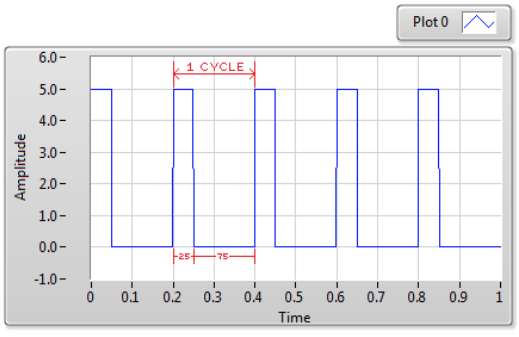


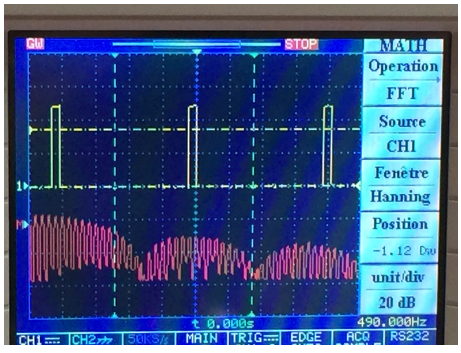
8. Sortie par la fonction PWM : obtention d'une tension analogique

Les sorties numériques de l'ARDUINO peuvent, pour certaines, délivrer un signal créneau dissymétrique PWM (Pulse Width Modulation) ou MLI (Modulation de la Largeur d'Impulsion).

```
analogWrite(brochePWM, sensorValue);
```

Le signal est :





Rappel :
les instructions `var_lu=analogRead(A0)` pour lire une donnée extérieure analogique (utilisation d'un CAN) puis en sortie `analogWrite(brochePWM,sensorValue)` utilisent des données entières comme `var_lu` qui varie de 0 à 1023 (10 bits en lecture) alors que `analogWrite` écrit de 0 à 255 (8 bits)

Comme on le voit sur le tracé, on a ici la même forme de signal que ce qui est obtenu par hachage d'une tension continue. (Voir poly de cours sur les hacheurs série et parallèle)

Comment donc récupérer un "vrai" signal analogique constant ?

- visualiser le spectre de Fourier et en déduire les **caractéristiques du filtre** à utiliser afin de transformer la sortie PWM en signal continu.
- Question : le signal obtenu à la sortie du filtre peut-il prendre toutes les valeurs de tension entre 0 et 5V ?