Travaux pratiques

Traitement du signal

Mise en œuvre d'une détermination automatique de période

Mise en garde :
Les cartes microcontrôleurs nécessitent une attention modeste pour être utilisées correctement. Je considérerai que ce minimum d'attention n'a pas été mobilisé si l'un d'entre vous en vient à confondre une entrée avec une sortie et à me griller la carte.
Merci de suivre attentivement les plans de connexion.

A. Objectifs de la séquence

Mettre en place une démarche algorithmique élémentaire pour mesurer la période d'un signal périodique quelconque.

Si ce premier objectif est atteint, nous pouvons chercher à identifier les grandeurs statistiques du signal. La moyenne et la valeur efficace sont visées.

Compétences :

- Mise en place d'une chaine d'acquisition à partir d'un capteur associé à un microcontrôleur
- Traitement du signal : acquisition d'un signal sous forme numérique, mise en forme, détection et synchronisation
- Mise en œuvre d'un algorithme

Outils et matériels :

- Microcontrôleur Arduino™
- Capteur de type photorésistance
- Amplificateur audio + micro (capteur sonore)
- Oscilloscope
- Fichiers CSV
- Ordinateur équipé d'une distribution python (idéalement Anaconda & Jupyter)

B. Détermination de la période stroboscopique d'une source

1. De l'usage d'un microcontrôleur et d'un capteur

a) Principe de la manipulation

Nous allons utiliser un capteur photométrique élémentaire : une photorésistance.

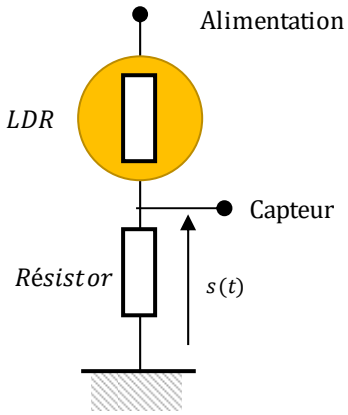
Souvent dénommée par leur acronyme anglais LDR, ces composants ont une très forte résistance dans l'obscurité, et une résistance beaucoup plus faible sous une lumière importante.
Celles que vous utiliserez ont des résistances pleines lumières de l'ordre de la dizaine de $k\Omega$.

L'idée simple est de mesurer le potentiel du point central d'un diviseur potentiométrique comportant une LDR et un résistor adapté.

En situation d'obscurité, la LDR domine la résistance d'ensemble et le potentiel est voisin du zéro. A l'inverse quand la lumière est vive, le résistor ne peut plus être négligé et le potentiel s'approche de sa valeur la plus élevée.

b) Schémas électriques

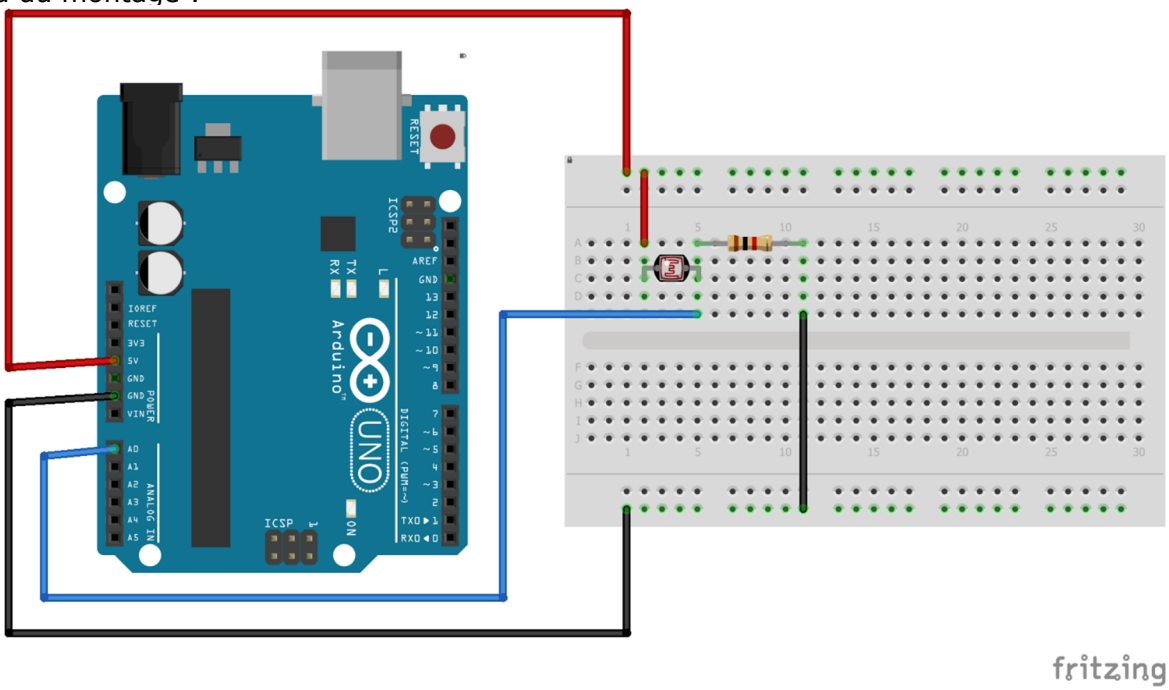
Le principe du pont diviseur est rappelé ici :



Question : Selon vous, comment faut-il choisir la résistance du résistor associé au pont ?

Une fois le pont réalisé sur votre breadboard, vous le connecterez à l'Arduino™ en respectant le schéma de connexion présenté ci-après

Schéma du montage :



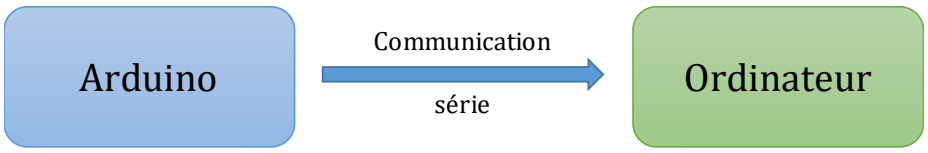
Considération sur l'entrée analogique :
Le signal, reçu sur l'entrée A0, compris entre la masse et la tension $V_{ref} = 5\,V$ va être converti en entier non signé sur 10 bits.
La valeur transmise sera l'image de la tension projetée sur cette plage d'acquisition entière.

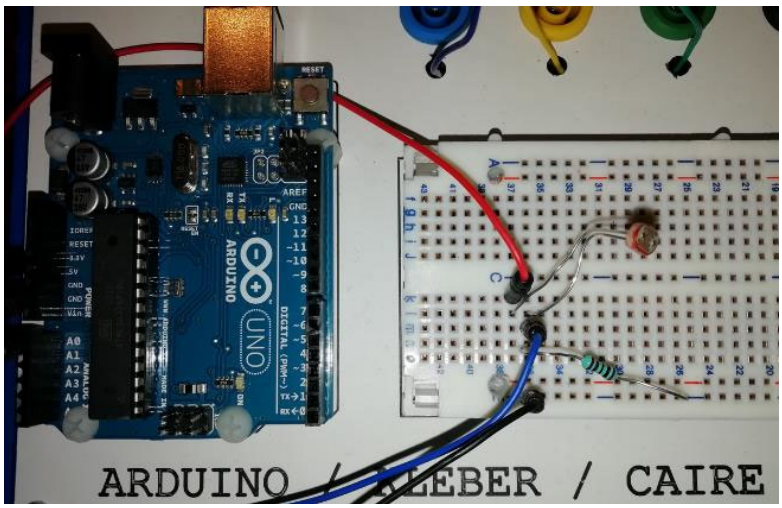
Question : Quelle est la plage de valeurs que peut fournir une entrée analogique de l'Arduino™ ?

c) Code de l'Arduino™

Le code fonctionnel de l'Arduino™ permet de générer une trame de communication série avec le PC.
Il est présenté en annexe et est téléchargeable sur le site usuel de la maison.

Implanter ce code et téléverser le sur votre platine dès que le montage est effectué.
Vous aurez alors dès que le port série du PC sera ouvert une communication du microcontrôleur vers l'ordinateur.





2. Un stroboscope téléphoné

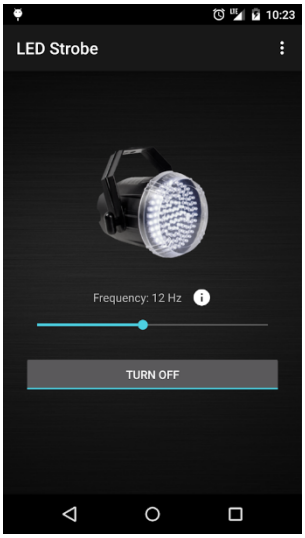
Pour solliciter ce montage nous allons utiliser un stroboscope de haute précision et de grande qualité, réalisé et construit dans ce seul but : votre **téléphone** !

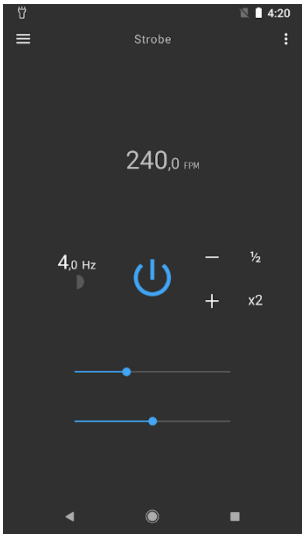
Que vous ayez un machin à pomme ou un truc à petit robot, connectez vous sur votre serveur d'application et téléchargez un stroboscope.

Pour la version android, choisissez par exemple :

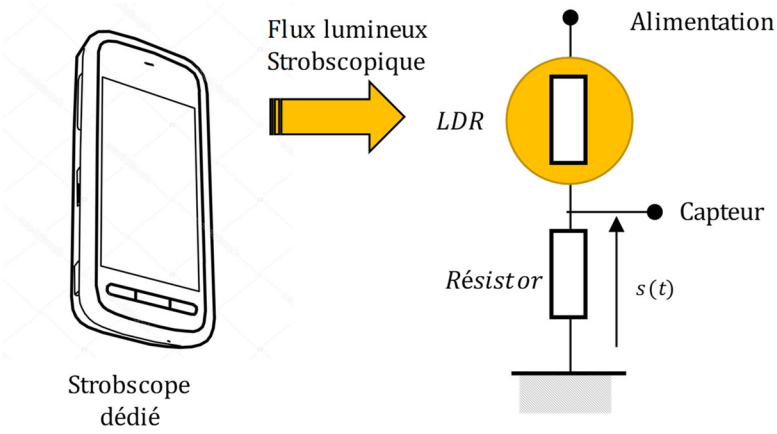
- Strobe de Zidsoft

- Stroboscopique de Easylabs





Ce dispositif doit pouvoir servir de source pour éclairer votre photodiode.



Votre carte Arduino™ lira la valeur du potentiel du pont. La fluctuation de cette donnée doit vous permettre de déterminer la période d'émission des flashs en analysant le signal sur une trame temporelle adaptée.

3. Un ordinateur au chevet du microcontrôleur

Le microcontrôleur Arduino™ a la capacité de traiter en direct le signal acquis, mais il faut alors s'assurer d'une maitrise minimale de son langage de programmation, le C. Ce langage est, certes l'un des plus fréquents langages exploités en systèmes embarqués, mais il est majoritairement ignoré en CPGE.

Nous contournons cette difficulté en communiquant en python avec la carte Arduino™ en mode maitre-esclave. Elle est alors simplement chargée de transmettre les séries temporelles lues sur les capteurs.

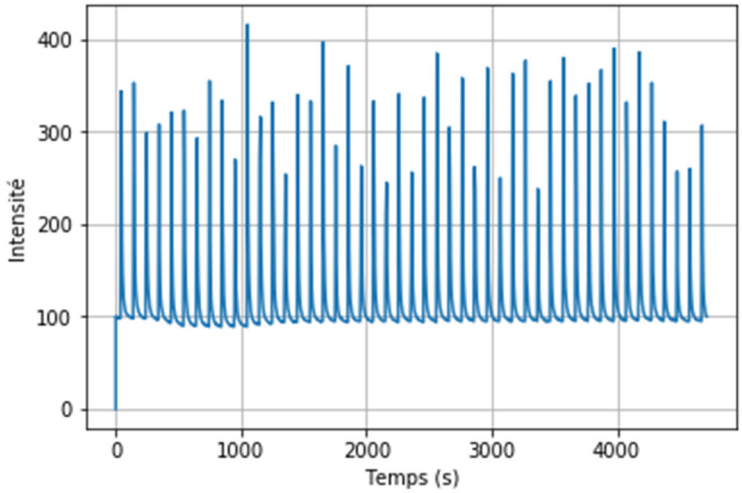
Le principe de la communication consiste à se servir du port USB comme d'un port virtuel série.
La réalisation est décrite dans le document pdf et le fichier jupyter intitulé **Arduino light strobe**.

Nous vous invitons à en prendre connaissance.

4. Un algorithme de recherche de période ?

Le code python fourni doit vous permettre d'acquérir des séries temporelles figurant l'évolution du potentiel de votre capteur au cours du temps.

Vous pouvez obtenir des figures de ce type :

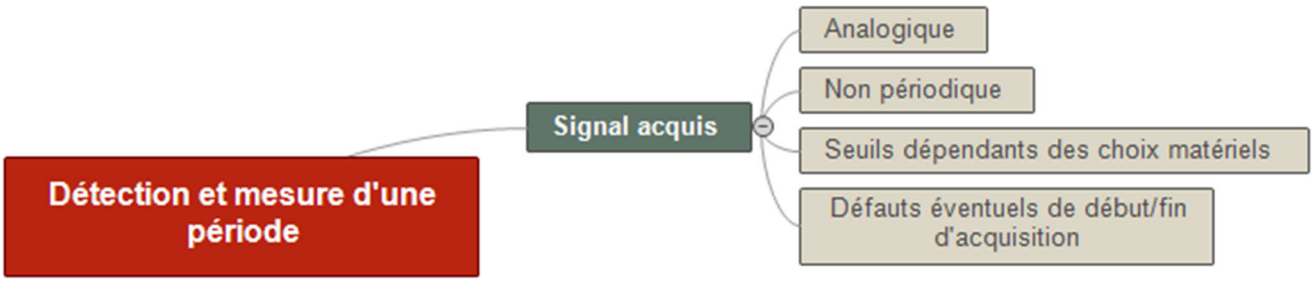


Vous remarquez que le signal n'est pas vraiment périodique.
Des fluctuations d'amplitudes sont aisément identifiées. Elles sont dues aux irrégularités de charge de vos flashs et à la tenue parfois maladroite des « stroboscopes ».

Nous supposerons néanmoins que la période d'émission de chaque impulsion est constante.

Nous attendons maintenant vos suggestions pour élaborer, mettre au point un algorithme de recherche de cette période.
Selon vous, que faut-il faire pour aboutir ?
Quels sont les moyens à mettre en œuvre pour éventuellement vous faciliter la tâche ?

Mise en œuvre d'une carte mentale



C. Détermination de la période d'un signal sonore élémentaire

1. De l'usage d'un microcontrôleur et d'un capteur

a) Principe de la manipulation

L'objectif est maintenant d'identifier la période d'un signal sonore monochromatique à l'aide de la chaine de traitement précédente.

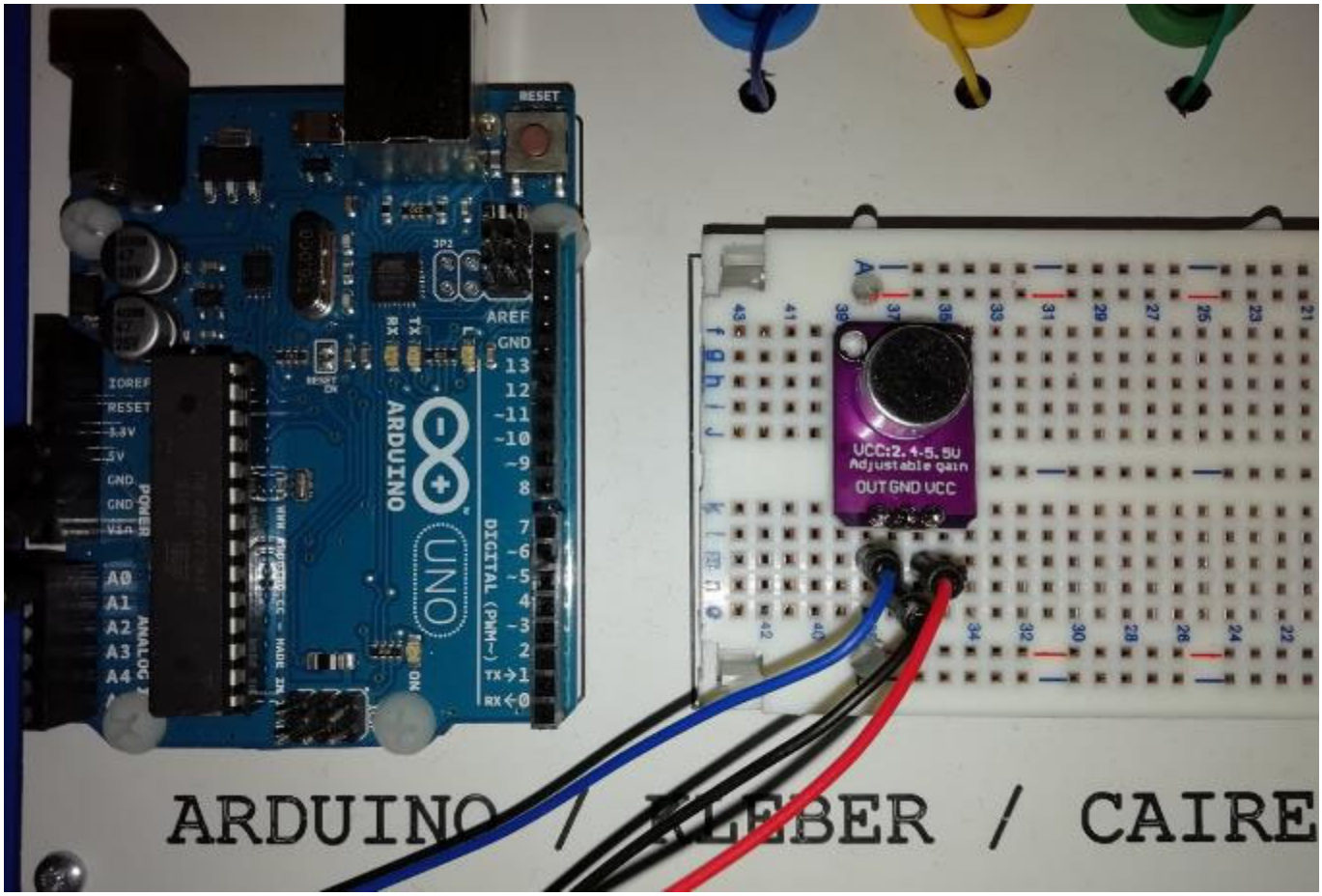
Nous reprenons la même démarche mais nous allons modifier le capteur d'acquisition qui sera un micro préamplifié adapté aux microcontrôleurs.
Nous verrons si d'autres modifications sont à effectuer pour s'assurer de nos mesures.

Avantages :

- Capteur distinct
- Support signal acoustique
- Dynamique différente
- Problème des limites

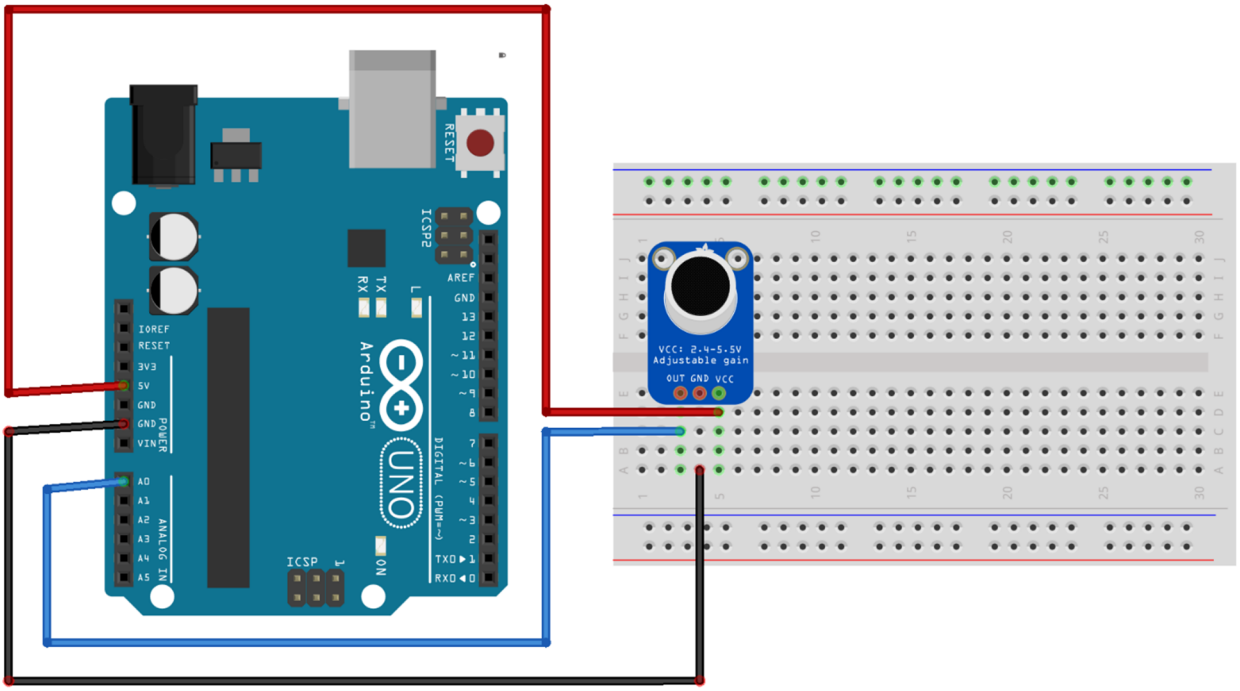
b) Schéma électrique

Le montage à réaliser est le suivant, il utilise un microphone MAX4466 de Maxim :



La carte entièrement montée est équipée d'un microphone à électret de 20 Hz à 20 kHz. L'amplificateur d'opération possède une suppression du bruit du bloc d'alimentation. Il convient donc pour des distorsions vocales, des enregistrements audios, des enregistrements d'échantillons et des projets audio utilisant la transformation de Fourier (FFT). Un petit potentiomètre à l'arrière fait varier l'amplification de 25x à 125x (entre 200 mV (crête à crête, pour un volume normal à une distance d'environ 15 cm) et 1 V (crête à crête)). Il peut se raccorder soit sur un amplificateur audio ou sur une entrée analogique d'un microcontrôleur. La sortie se fait rail-à-rail et permet une amplitude de crête jusqu'à 5 V.

Le schéma est simple à réaliser :



Éviter sous peine de destruction toute confusion entre la masse et V_{cc}. Une erreur de polarisation ne pardonne pas !

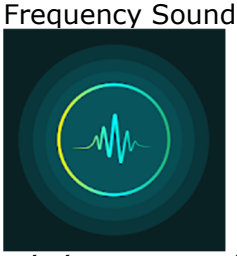
2. Un générateur harmonique sonore

Pour solliciter ce montage nous allons utiliser un appareil de synthèse sonore de haute précision et de grande qualité, réalisé et construit dans ce seul but : votre **téléphone** !

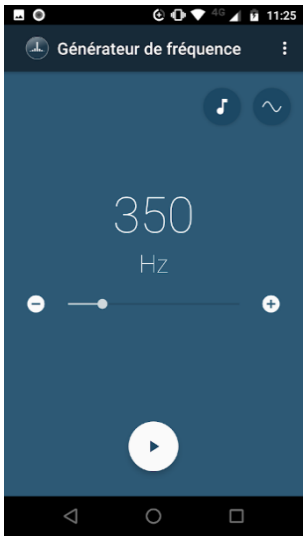
Que vous ayez un machin à pomme ou un truc à petit robot, connectez-vous sur votre serveur d'application et téléchargez un générateur sonore monochromatique.

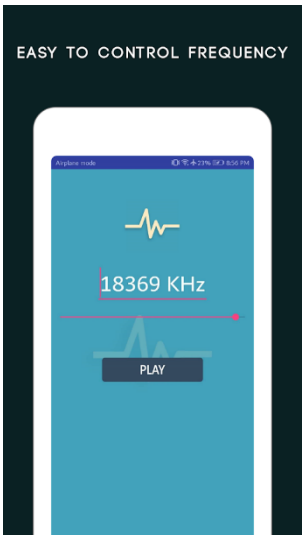
Pour la version android, choisissez par exemple :

- Frequency Sound Generator



- Générateur de fréquence de Boedec





Ce dispositif doit pouvoir servir de source sonore pour alimenter en information votre micro.

3. Un ordinateur au chevet du microcontrôleur

Nous exploitons la même recette que précédemment.
Commencez vos mesures avec les plus basses fréquences audibles.
Si elles ne sont pas satisfaisantes, pensez à augmenter vos débits (baud rates).

4. Un algorithme de recherche de période ?

Le code python fourni dans le cadre précédent doit vous permettre d'acquérir des séries temporelles figurant l'évolution du potentiel de votre capteur sonore.

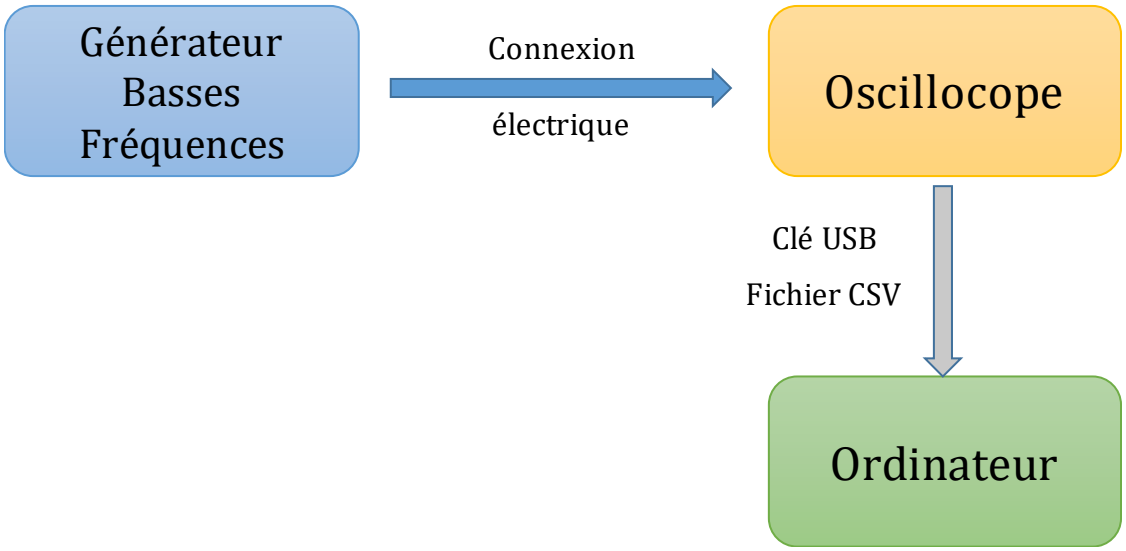
1. Exploitez le pour vous assurer d'une acquisition.
2. Représentez votre fluctuation temporelle.
3. Essayez à partir de là de définir un algorithme de traitement adéquat.

Avez-vous eu besoin de modifier la version employée au cours de la manipulation précédente ?

D. Détermination de la période d'un signal

Nous allons de nouveau essayer de déterminer la fréquence d'un signal périodique mais nous allons le faire dans un environnement de mesure contrôlé et plus classique que les manipulations précédentes.

1. Votre signal sera issu d'un GBF et enregistré sur un oscilloscope.
2. Les données de l'oscilloscope seront exportées à partir d'une clé USB sur votre ordinateur.
3. Utilisez le complément pdf et ipynb intitulé **Ouverture Fichier oscilloscope Keysight Student** pour lire vos fichiers de données.



Testez l'algorithme précédent et confirmez ou infirmez ses résultats.

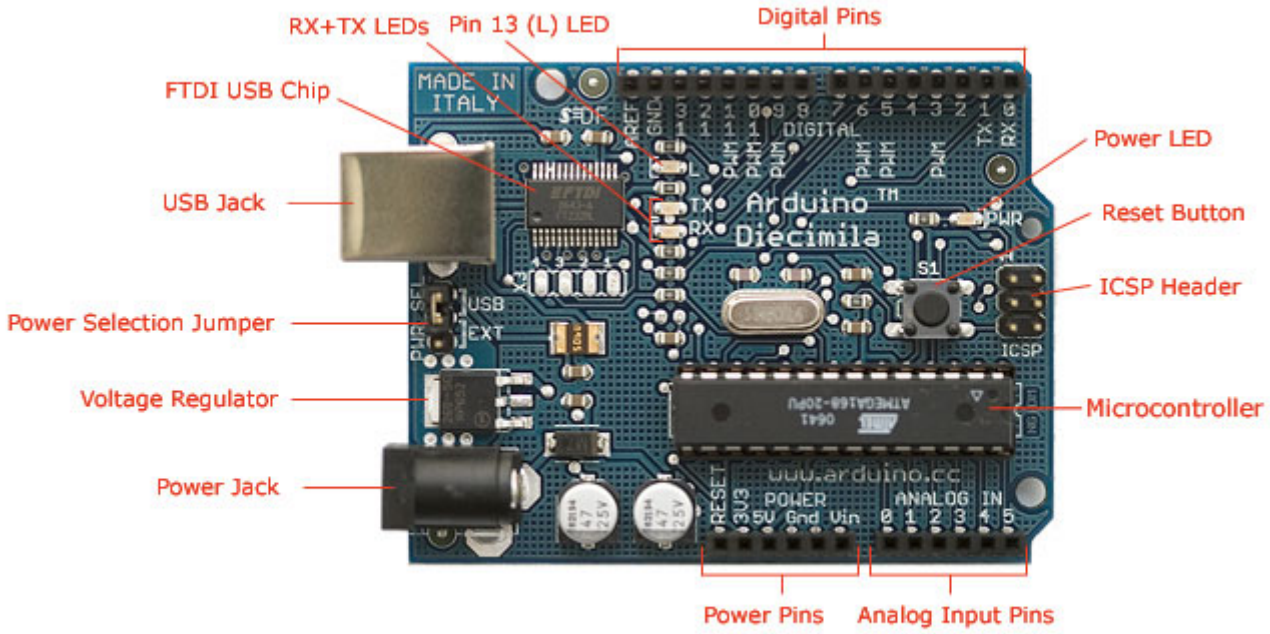
Pour information des signaux exemples d'un même signal sinusoïdal sont fournis en ligne. Vérifiez le fonctionnement de votre code sur ces exemples.

Pourquoi certains d'entre eux donnent-ils des résultats aberrants ?

E. Annexe : Description de l'Arduino™

1. La carte Arduino™

a) Des entrées & sorties



Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.

Vous retrouvez ci-dessus la structure familière de nos cartes microcontrôleurs Arduino™, elles possèdent dans leur version de base :

- 6 entrées analogiques (A0-A5 en bas à droite de la figure) codées sur 10 bits
- 14 entrées/sorties numériques (0-13 en haut, de droite à gauche)

b) Des sorties analogiques

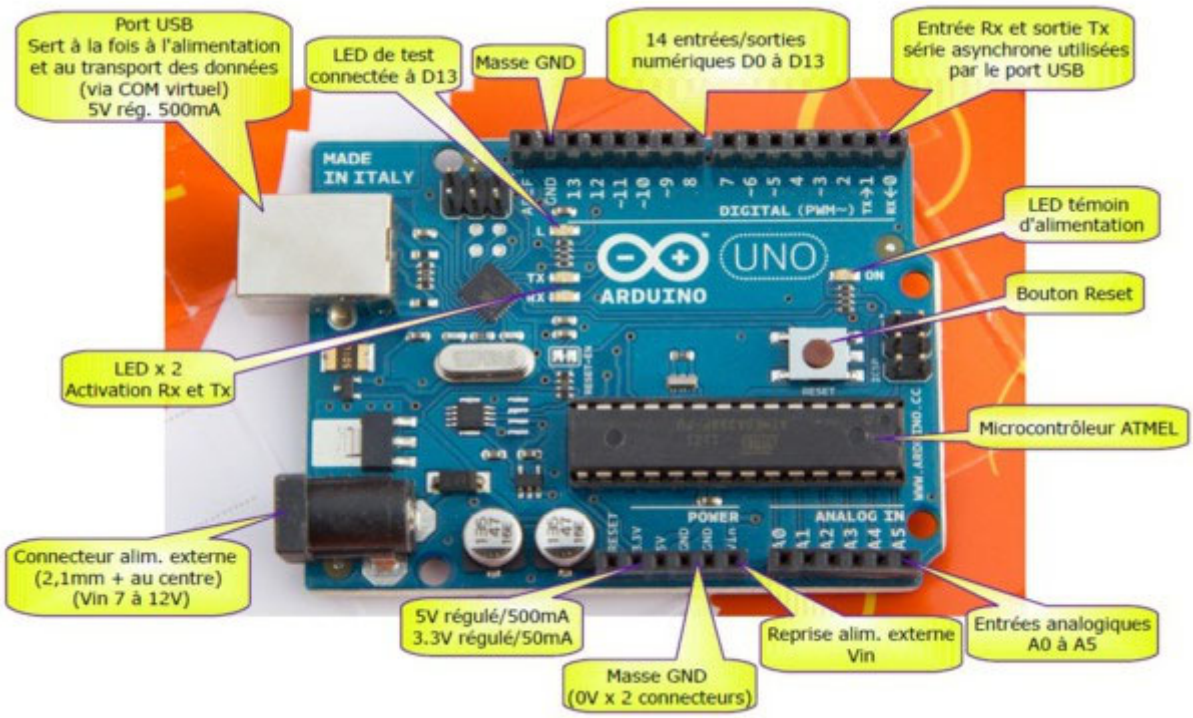
Et les sorties analogiques me direz-vous ?
La réponse est simple, l'Arduino™ n'en possède pas !

Et pourtant vous trouverez la fonction **analogwrite** dans la libraire de base de l'Arduino™.
Qu'en est-il ? Lisons la description de la fonction dans la bonne vieille langue de Shakespeare :

> *[Analog I/O] Description*
> *Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin. The frequency of the PWM signal on most pins is approximately 490 Hz. On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz.*
>
> *On most Arduino™ boards (those with the ATmega168 or ATmega328P), this function works on pins 3, 5, 6, 9, 10, and 11. On the Arduino™ Mega, it works on pins 2 - 13 and 44 - 46. Older Arduino™ boards with an ATmega8 only support analogWrite() on pins 9, 10, and 11. The Arduino™ DUE supports analogWrite() on pins 2 through 13, plus pins DAC0 and DAC1. Unlike the PWM pins, DAC0 and DAC1 are Digital to Analog converters, and act as true analog outputs.*
> *You do not need to call pinMode() to set the pin as an output before calling analogWrite().*
> *The analogWrite function has nothing to do with the analog pins or the analogRead function.*

Résumons le propos, il n'existe effectivement pas de sorties analogiques sur nos cartes mais il est possible d'en simuler grâce à l'usage d'une modulation de largeur d'amplitude. Une technique classique d'émulation désignée en anglais sous le sigle **PWM**.

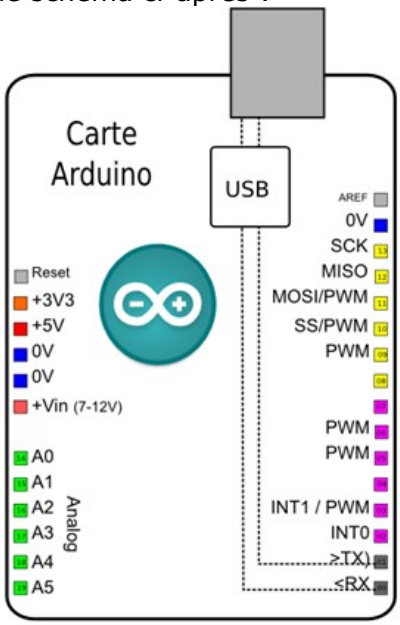
c) Les spécificités techniques



Une carte Arduino™ Uno comporte les ports (pins en anglais) suivants :

- 14 (+6) Entrées – Sorties logiques (port ou « pin Digital » de 0 à 13)
- Série asynchrone (avec 0 sur Rx et 1 sur Tx) : les pins 0 et 1 ne seront donc pas utilisables
- 2 Interruptions externes sur 2 et 3
- Sortie 13 couplée à une LED sur la carte
- 6 Entrées analogiques (A0 à A5)
- La tension d'entrée doit nécessairement être inférieure à la tension de référence (5 V ou 1,1V ou AREF : référence externe)
- 6 CAN 10 bits (plage de 1024) à 10kHz maximum, ces entrées peuvent aussi fonctionner comme des E/S numériques
- 6 Sorties « Analogiques » : 6 PWM sur les ports 3,5, 6, 9, 10 et 11, construites sur les pins d'entrées- sorties logiques

Vous retrouverez ces fonctions sur le schéma ci-après :



F. Annexe : Programmation de l'Arduino™

Le langage de programmation Arduino™ est basé sur les langages C et C++. Si vous connaissez déjà ceux-ci, vous n'aurez aucun mal à vous familiariser avec l'environnement Arduino™.

Bien entendu, en classe préparatoire, l'apprentissage du C est remis aux calendes grecques !
Le choix s'est porté avec raison sur python, et avec sans doute moins de raison sur Caml. Le C vous est donc inconnu à moins d'heureuses initiatives de votre part.

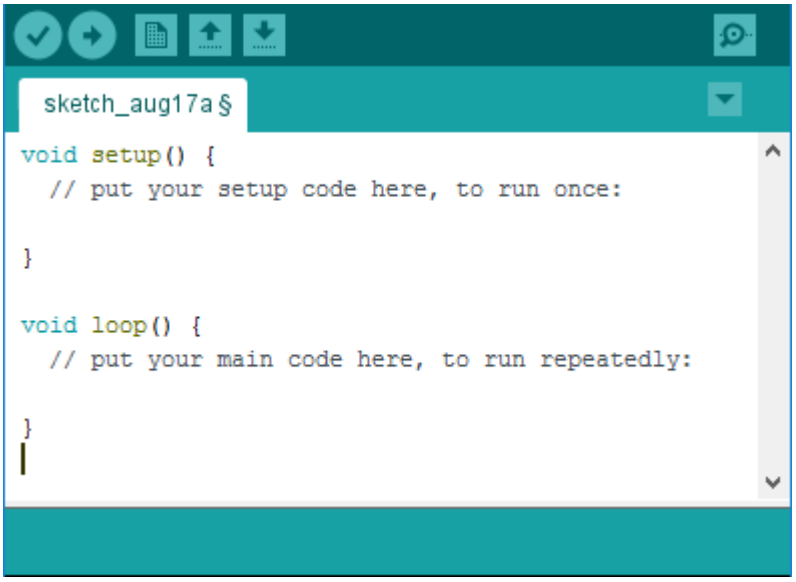
Pas d'inquiétude, le principe de ce TP repose sur l'existence d'éléments de code en C qui vous seront fournis.

1. IDE officiel de l'Arduino™

Il est téléchargeable gratuitement sur internet à cette adresse :
https://www.Arduino.cc/en/Main/Software

Il y est abondamment documenté. N'hésitez pas à profiter de l'occasion pour vous documenter si vous disposez de temps libre.

L'IDE est normalement installée sur vos machines, il suffit de la solliciter. Rechercher Arduino !
Une fois lancé, vous verrez la fenêtre type suivante s'ouvrir :



C'est la fenêtre de script qu'il vous faudra compléter en fonction de vos besoins.

Elle contient deux fonctions qui ne renvoient rien (déclaration void), ce sont donc pour les puristes des procédures.

La première **setup** est exécutée à l'initialisation du microcontrôleur. Elle contient généralement la déclaration des constantes, des variables et des ports d'entrée-sortie que vous allez utiliser.

La seconde **loop** est exécutée en continu après la première. C'est par défaut une boucle infinie dont le microcontrôleur ne sortira jamais.

2. Votre code

La programmation en C n'étant pas votre tasse de thé, nous vous invitons à charger le code ci-après dans l'interface et à le téléverser sur le microcontrôleur.

Code à installer :

```
// Constantes et variables
int tempsPause=1000; // eventuelle temporisation
int valeur =0; // variable de stockage entière
int analogPin=0; // Choix de la voie d'acquisition (à adapter)

// Variables à déclarer -- protocole d'initialisation
void setup ( ) {
pinMode(analogPin,INPUT); // Définition de la voix d'entrée
Serial.begin (19200) ; // choix du mode de communication
while(!Serial); // Attend la connexion du port
Serial.println("Début");
}

// Boucle principale - flux continu
void loop ( ) {
valeur=analogRead(analogPin); // lecture de la valeur analogique d'entrée
Serial.print( valeur ) ;
// Envoi la mesure au PC par la liaison série (port USB)
Serial.print("\t" ) ; // Ajout d'une tabulation
Serial.println( millis( )) ;
// Envoi de la valeur temps puis retour à la ligne
// Une éventuelle temporisation si le débit est trop rapide (peu probable)
}
```