

# PR-2101: Arduino

2018- 2019

— Émetteur VLF Morse

## SOMMAIRE

<b>I - Introduction.....</b>	<b>2</b>
1) Vue d'ensemble.....	2
2) Principe de fonctionnement.....	2
 <b>II - Déroulement.....</b>	 <b>3</b>
1) Programme	
2) Mesures/tests	
 <b>IV - Conclusion.....</b>	 <b>7</b>

Auteur:

BENKEMOUN Jules F4IEY

ESIEE Paris

Noisy-le-Grand

21 Janvier 2020

## I - Introduction

Pour ce samedi technique confiné, j'ai choisi de réaliser une *courte présentation sur la VLF* (très basse fréquence). La présentation est composée de manière suivante:

- SAQ c'est quoi
- Communiquer des données en VLF avec Arduino
- Tests

## **Vue d'ensemble**



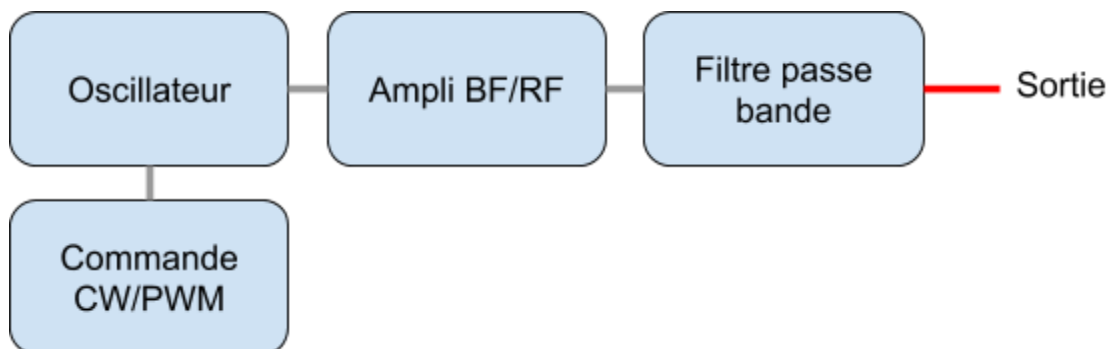
SAQ est l'indicatif d'appel d'une station suédoise ancestrale. Elle émet uniquement en CW sur 17.2KHz deux fois dans l'année (noël et le jour de Alexanderson). La porteuse est générée par un alternateur électrique lancé à pleine puissance (500HP/370kW à 2.2KV) et à plein régime (plus de 700 tr/min).

Par la suite, nous allons voir comment recevoir ces signaux BF simplement et aussi comment transmettre facilement!

## **Principe de fonctionnement**

L'émetteur envoie une porteuse CW sur 17.5KHz générée par une sortie PWM du microcontrôleur Arduino. Cette sortie est injectée dans un oscillateur pour être ensuite amplifiée, filtrée et envoyée soit dans un câble ou une antenne.

Schéma synoptique:



## II -Déroulement:

### Programme:

Le code morse est envoyé automatiquement.

Le code Arduino est composé de:

- la fonction **tone(pin, fréquence)** qui génère la sortie sous forme d'un signal carré
- les fonctions **dot()** et **dash()** pour définir ce que est un point ou un trait.
- Un **switch()...case** qui rassemble les deux fonctions pour obtenir l'alphabet morse
- la fonction **wpmToMillis()** pour convertir le standard de vitesse morse (mot par minute) en millisecondes.

3 fonctions intéressantes:

```
unsigned long wpmToMillis(int wpm) {
    //le mot de référence est PARIS: 50 éléments
    unsigned long ptMs = 60000/(wpm*50);
    return ptMs;
}
unsigned long tdot = wpmToMillis(WPM);
unsigned long tdash = 3*tdot;
```

```
void dot() {
    //on écrit un point
    tone(outPin, vfo);
    delay(tdot);
    noTone(outPin);
    delay(tdot); //espace inter-caractère
}
void dash() {
    //on écrit un trait
    tone(outPin, vfo);
    delay(tdash);
    noTone(outPin);
    delay(tdot); //espace inter-caractère
}
```

On a WPM un nombre de référence (par exemple 17).

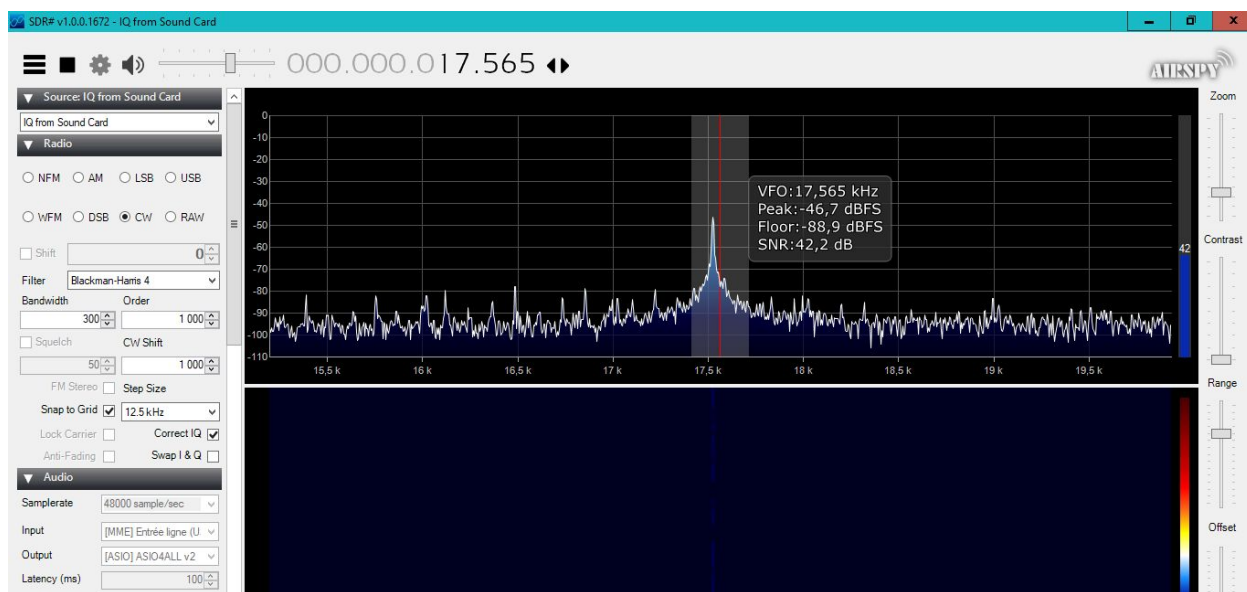
Pour mieux comprendre les durées: <https://www.f8eho.net/content/calcul-de-la-vitesse>

Tout le code peut être retrouvé sur [ma page GitLab](#)



### III - Essais et mesures

Une fois assemblé, nous mesurons le SNR (rapport signal/bruit) à travers la masse du réseau électrique et plusieurs types d'antennes (bobine avec ferrite, boucle magnétique, verticale, patch...). Pour réceptionner un signal avec une fréquence aussi basse, nous ne pouvons pas utiliser un récepteur grandes ondes classique; nous avons besoin d'une carte son externe branchée à la masse du réseau ou une autre antenne. Pour visualiser, nous avons utilisé le logiciel d'analyse spectrale *SDRSharp*. (capture d'écran ci-dessous)



Mesures le long d'un fil (masse réseau électrique)

distance(m)	SNR(dBm)
1	-61.9
4	-63.6
10	-74.4
Perte estimée(dB/m)	1.4

## Mesures sans fil (RF)

distance(m)	SNR moyen avec antenne loop et verticale (dBm)
1	-55.6
2	-61.3
4	-71.2
4.5	-73.0

On peut en déduire une estimation de la portée maximale en faisant l'hypothèse que l'onde qui se propage est une OPPM et que sa polarisation est linéaire.

On approxime le champ  $E_r = A \cos(\omega t - kx)$  Avec  $k$  la direction de la propagation.

à  $t = 0$ , on a  $E_r = A \cos(-kx)$

on sait que la longueur d'onde  $\lambda = \frac{2\pi}{k} \approx 17 \text{ km}$

On a alors  $E_r = A \cos(-\frac{2\pi}{17.10^3}x)$

Si on suppose le comportement du champ linéaire, de 0 à 4.5m, on a  $E_r = A \cos(-\frac{2\pi}{17.10^3} \times 4.5) \approx A$

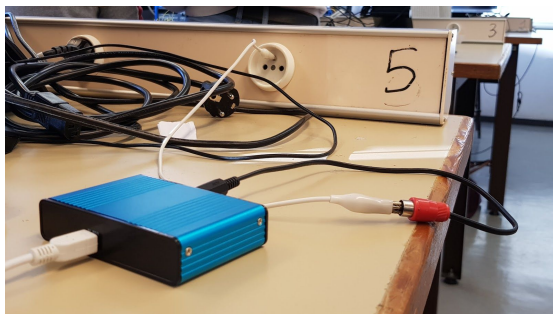
Or, à cette distance, l'amplitude est très faible voire négligeable. Donc si l'on s'éloigne encore avec la même puissance P.A.R (500mW), l'amplitude sera fortement réduite. Il est alors cohérent que la portée soit faible.

## IV - Conclusion

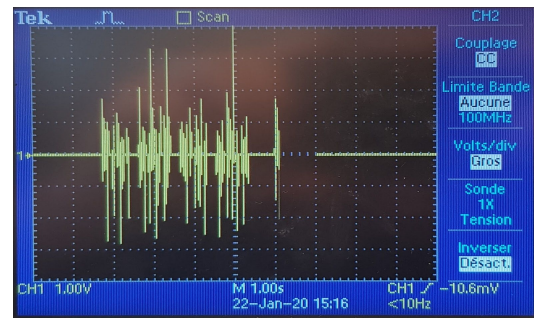
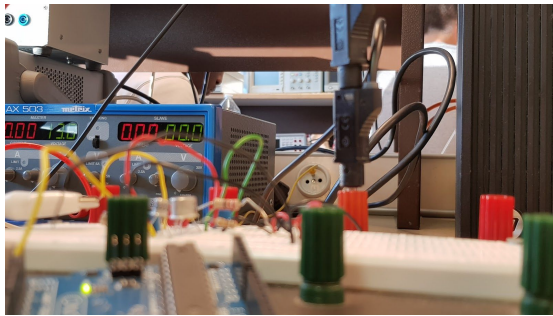
Et voilà, j'espère qu'après la présentation (no spoil here ;) ) vous serez capable de transformer votre Arduino en TX VLF et de recevoir ces signaux

73 et à samedi, DE F4IEY Jules

La carte son: ATTENTION! on se branche bien sur **la masse du réseau** (qui est sans danger)!



Émetteur sur réseau (masse):



RF:

