

## Toelichting bij de Arduino Uno sketch

### Doel

Doel van de sketch is het regelen van het QSK muten van je eigen vertraagde audio CW signaal dat via de WebSDR ontvangen wordt.

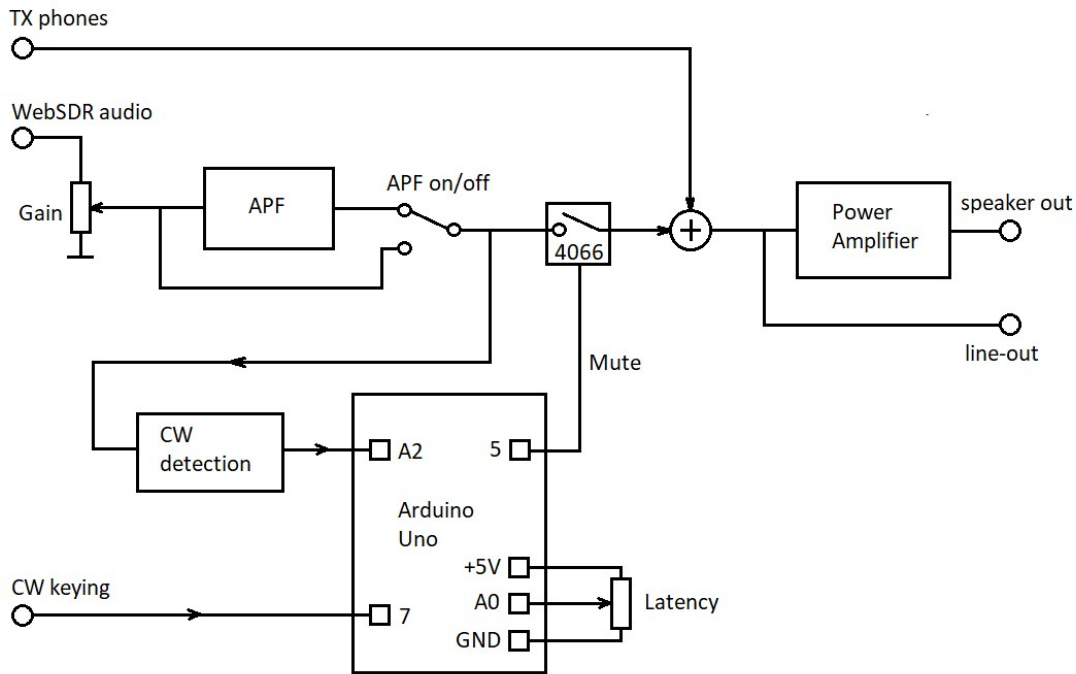


Fig.1. Blokschema van de hardware met de Arduino Uno.

### Automatisch latency meten

De vertraging in het muten kan handmatig met de potentiometer ingesteld worden op de actuele latency van de WebSDR.

De latency heeft echter geen constante of vaste waarde en kan variëren. Het is handiger wanneer de vertraging in het muten automatisch de actuele latency volgt.

Het programma moet hiervoor het tijdsverschil tussen de verzonden en de ontvangen punten en strepen meten. Een geldig gemeten tijdsverschil moet liggen tussen de verwachte minimum (**latency\_min**) en maximum latency (**latency\_max**).

Alléén tijdens het zenden van CW (**transmitting**) kan de latency gemeten worden. Deze periode start bij de eerste punt of streep en eindigt pas de maximaal te verwachten latency tijd na de laatst verzonden punt of streep. Dan weten we zeker dat alle verzonden punten en strepen ontvangen zijn.

## CW detectie

Het CW detectie circuit is een eenvoudige omhullende detector.

De AGC actie in de WebSDR probeert de piekamplitude constant te houden. Dat is een nadeel voor deze manier van CW detectie. Voor sterkere CW signalen is het toch bruikbaar.

Maar het vraagt wel om een complexere verwerking in de Arduino Uno ter compensatie.

De Arduino Uno samplede de amplitude met een frequentie van 500Hz.

Alléén tijdens het zenden wordt de piek amplitude (**peakAMP**) gevolgd met aparte attack en decay snelheden. De drempel voor de detectie van het ontvangen CW signaal (**myCW**) is ingesteld op 75% van de piek amplitude. Dit niveau wordt o.a. bepaald door het gedrag van de AGC van de WebSDR. Vooraf aan de berekening van de latency wordt gefilterd voor mogelijke spikes t.g.v. ruis. Punten en strepen korter dan 10msec (**spikeT**) worden als een spike gezien en overgeslagen.

De piek amplitude niet gevolgd wanneer we niet zenden (**transmitting**). Er wordt wel gedetecteerd, maar dit wordt niet gebruikt, behalve voor het aansturen van de LED.

De analoge omhullende detector en deze processing hebben een vertraging waarvoor in de Arduino Uno sketch gecorrigeerd moet worden (**det\_delay**).

Het analoge amplitude signaal van de omhullende detector heeft een DC offset spanning.

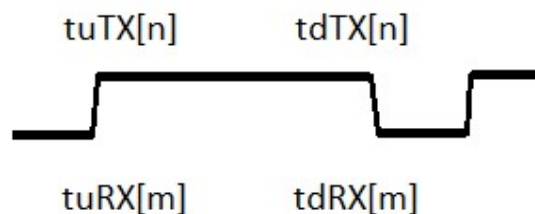
In de sketch moet de correctie (**AMP\_offset**) worden afgesteld op de actuele offset spanning.

## Keuze voor manual of automatic

Door de potmeter voor de handmatige latency op de laagste stand te zetten wordt de automatisch gevonden latency gebruikt.

## Latency berekening

De tijdstippen waarop opvolgende verzonden en ontvangen punten en strepen starten en eindigen worden bijgehouden in circulair buffers met een lengte van 10 (**tuTX/tdTX** en **tuRX/tdRX**).



Op elk moment hebben we van het verzonden CW (TX) en van het ontvangen vertraagde eigen CW signaal (RX) van de laatste 10 punten of strepen de start en stop tijden. Voor elke punt of streep wordt vervolgens de gemiddelde tijd berekend. Dit geeft twee rijen van 10 gemiddelde tijden (**taTX** en **taRX**). Punten en strepen moeten volledig voorbij gekomen zijn voor een geldig gemiddelde.

De routine voor het berekenen van de latency wordt doorlopen direct na een volledig ontvangen punt of streep en alléén tijdens de periode dat er CW gezonden wordt (**transmitting**).

Voor het bepalen van de actuele latency worden de laatste 4 punten of strepen van het ontvangen CW signaal gebruikt. Het tijdspatroon van de vier bijbehorende gemiddelde tijden (**taRx/taRXc**) wordt vergeleken met vier opvolgende tijden uit de 10 gemiddelde tijden van het verzonden CW (**taTX/taTXc**).

Om ze te vergelijken worden ze van elkaar afgetrokken. Wanneer het tijdspatroon van punten en strepen past, zijn de vier verschillen (**diff**) tussen beide gemiddelde tijden gelijk. Past het tijdspatroon niet, zijn de vier verschillen niet gelijk en wijken onderling sterk af. Berekend wordt de som (**sumres**) van de absolute waarden van het verschil van de vier verschillen (**diff**) en het gemiddelde verschil (**mean\_diff**).

In de praktijk zit er altijd ruis op de gemeten tijden. Het tijdspatroon van punten en strepen past wanneer deze som (**sumres**) onder een bepaalde drempel komt (**lat\_thres**).

Alleen wanneer dit eenmalig over de 10 gemiddelde tijden van het verzonden CW (**taTXc**) optreedt (**match\_count**) is de match eenduidig en wordt het gemeten tijdsverschil geaccepteerd als zijnde de actuele latency. Eenduidigheid is er bijvoorbeeld niet wanneer er alleen punten gegeven worden. De latency wordt dus niet altijd na elke ontvangen punt of streep bijgewerkt.

De drempel is een afschatting van hoeveel timing onzekerheid die we verwachten op het ontvangen gedetecteerde CW signaal. Deze drempel zorgt er ook voor dat inbrekende CW signalen met een minimale kans gezien worden als een eigen CW signaal en een verkeerde latency veroorzaken.

## Variaties in de latency

De latency van de WebSDR vertoont variaties die niet altijd op tijd gemeten kunnen worden. De handmatig ingestelde vertraging zal altijd afwijken van de werkelijke latency.

Deze kleine variaties in latency worden opgevangen door de muting van de punten en strepen iets eerder te laten beginnen en iets langer te laten duren. Deze extra tijd wordt ingesteld met **var\_latency**.

## Mute signaal voor de 4066 schakelaar

Het mute (**Mute**) signaal wordt afgeleid van de inhoud van de twee circulaire buffers waarin de start- en eindtijden van de het verzonden CW worden bijgehouden (**tuTX** en **tdTX**). Wanneer de actuele tijd groter is dan een starttijd (**tuTX**) plus de latency én kleiner dan de bijbehorende eindtijd (**tdTX**) plus de latency, wordt het audio gemute.

## CONT en OFF

Met de drie standen schakelaar hebben we de mogelijkheid om het audio volledig te muten (**OFF**) of het muten uit te schakelen zodat we het eigen vertraagde CW kunnen horen (**CONT**).

## Main loop frequentie 500Hz

Om de analoge CW detectie voorspelbaar te houden wordt de main loop tijd met een extra delay op 2msec gehouden.

## Trimmen van de det\_delay

De analoge omhullende detector en de detectie processing in de Arduino Uno introduceren een vertraging waarvoor in de Arduino Uno sketch gecorrigeerd moet worden met **det\_delay**. Het afstellen hiervan kan op het gehoor. Beter is de latency parallel te meten met een oscilloscoop met op het ene kanaal het uitgezonden RF signaal en op het andere kanaal het audio van de WebSDR.

## RF detectie

Het CW keying signaal kan meestal van de transceiver of de externe keyer gehaald worden. Een alternatief is het detecteren van de RF energie op de coaxkabel naar de antenne. Hiermee wordt alleen gemute wanneer er ook echt CW verzonden wordt. En het is bruikbaar voor SSB. Zie: RF\_detection\_circuit.png

## Is een circulair buffer met lengte 10 voldoende?

We nemen alleen de laatste 4 ontvangen punten en strepen voor de latency meting. Deze 4 moeten na de worst case latency nog in het TX buffer zitten. Tijdens deze worst case latency periode kunnen nieuwe punten en strepen verstuurd zijn. Er blijven 6 plaatsen in het TX buffer over voor deze nieuwe punten en strepen. Dat aantal is het grootste bij worst case latency, bij punten en bij hoogste CW snelheid. 12wpm zijn 5 punten/sec  
30wpm zijn 12.5 punten/sec en 6.25 punten per latency\_max van 0.6sec  
=> bij 30wpm is een lengte van 10 net genoeg.  
Dat betekent niet dat bij een hogere CW snelheden het direct fout gaat!  
Er worden praktisch nooit 6 punten direct achter elkaar gegeven en de latency is normaal <0.5sec.

## Ontwikkeling van het algoritme

Het algoritme voor het automatisch volgen van de actuele latency in het audio van de WebSDR is eerst in Matlab zover mogelijk uitgewerkt. Zie: QSK\_CW\_Arduino.m  
In plaats van Matlab kan ook de gratis beschikbare Octave gebruikt worden:  
<https://www.gnu.org/software/octave/>  
Het algoritme is opgezet met in het achterhoofd dat het moet kunnen draaien op een Arduino Uno. De processing mag niet te veel rekentijd kosten of te complex zijn. Tijdens het verder testen is regelmatig teruggegaan naar de Matlab (Octave) implementatie.  
Belangrijk is de betrouwbaarheid. Liever een kleine afwijking dan er volledig naast zitten. De drempel voor de acceptatie van een nieuwe gemeten latency kan beter niet te hoog liggen (lat\_thres laag). Niet iedereen heeft de beschikking over Matlab of Octave. Daarom is een Excel versie gemaakt met daarin het principe van automatisch volgen van de latency (QSK\_CW\_arduino.xlsx).