CTCSS encoder with Arduino

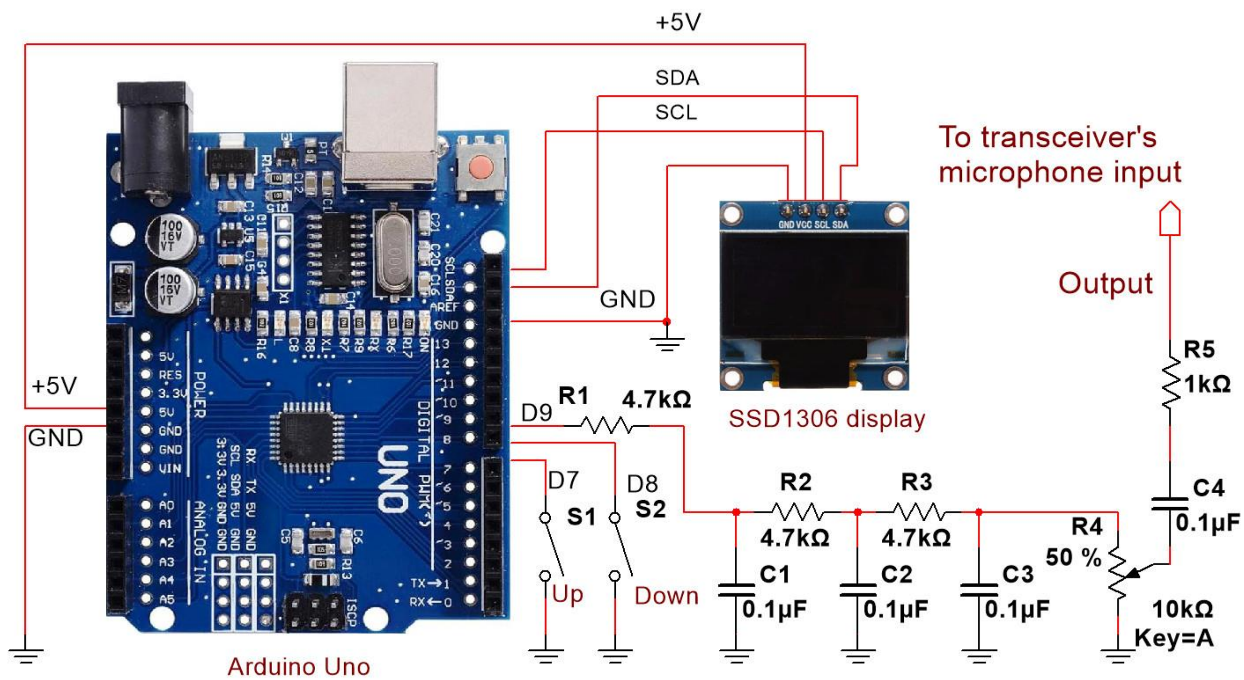
by Daniel Romila VE7LCG

I wanted to make a fast project by copying the work of somebody else, an Arduino CTCSS generator for older transceiver which do not have tones. I searched on youtube.com and I was surprised to see just several projects, which obviously were exactly what I wanted, but which did not offer everything (code, schematic, the used libraries for the Arduino program). They showed the product I was interested in, but with a code generating only one frequency, or using a computer for displaying the frequency, no schematic and so on.

I felt that I was left to my own devices, but this did not discourage me. I tried code that was already made by others, even if the LCD display was not there, the UP and DOWN buttons were not there. I gave credit especially to Geert Vanhulle - ON7GF, to the initial coder(s) on the program I used, which is available from my github:

[danielromila/CTCSS-encoder: test (github.com)]

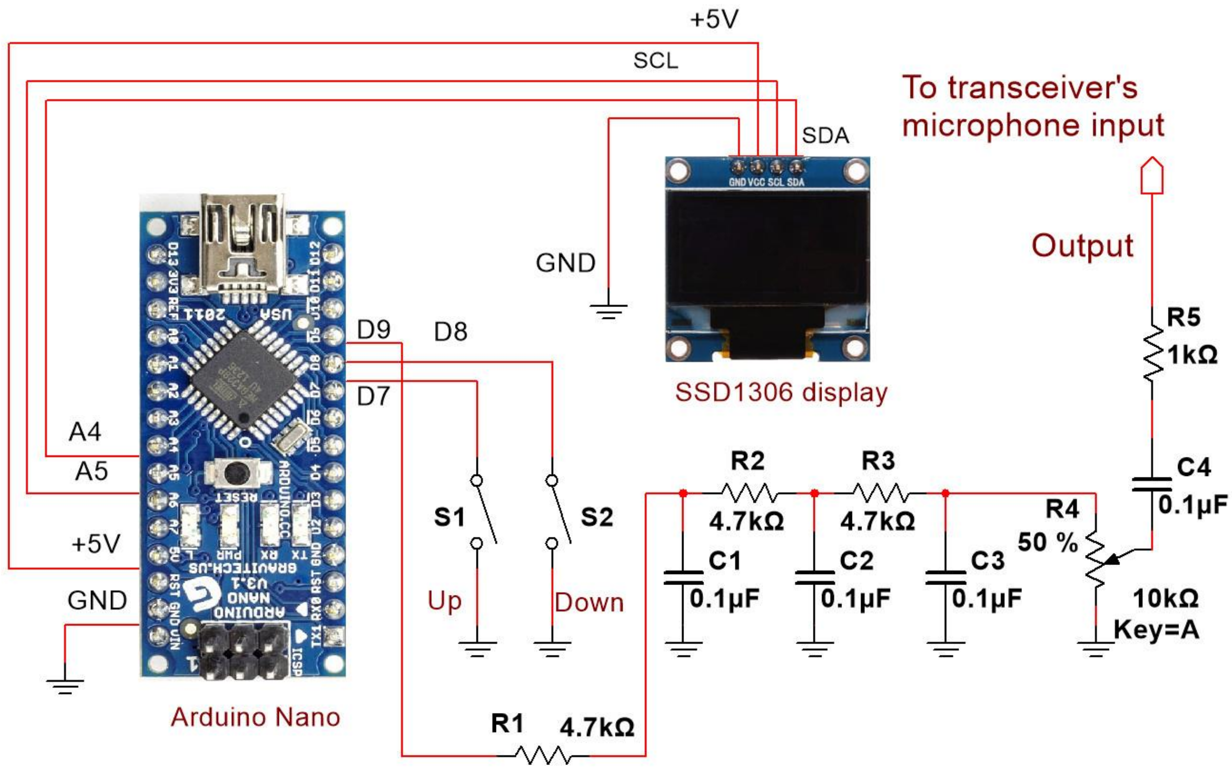
I wanted something simple, and I used Arduino Uno for test, but Arduino Nano also works (and other Arduino boards). I used a 128 X 64 SSD1306 LCD display, which requires I2C connection to the Arduino board. For UP and DOWN (to select the CTCSS tone/frequency, from predefined 50 tones/frequencies) I used buttons, not a rotary encoder, for simplicity and price. The S1 and S2 are connected to the D7 and D8 digital pins of Arduino Uno (Nano).



The power supply with 5V is done through the USB port of the Arduino UNO board.

The output is collected from D9 through R1. Arduino generates a square signal, which would make a harsh voice in transmission. The R1, R2, R3 and the capacitors C1, C2 and C3 round the signal and make it look more like a sinus wave.

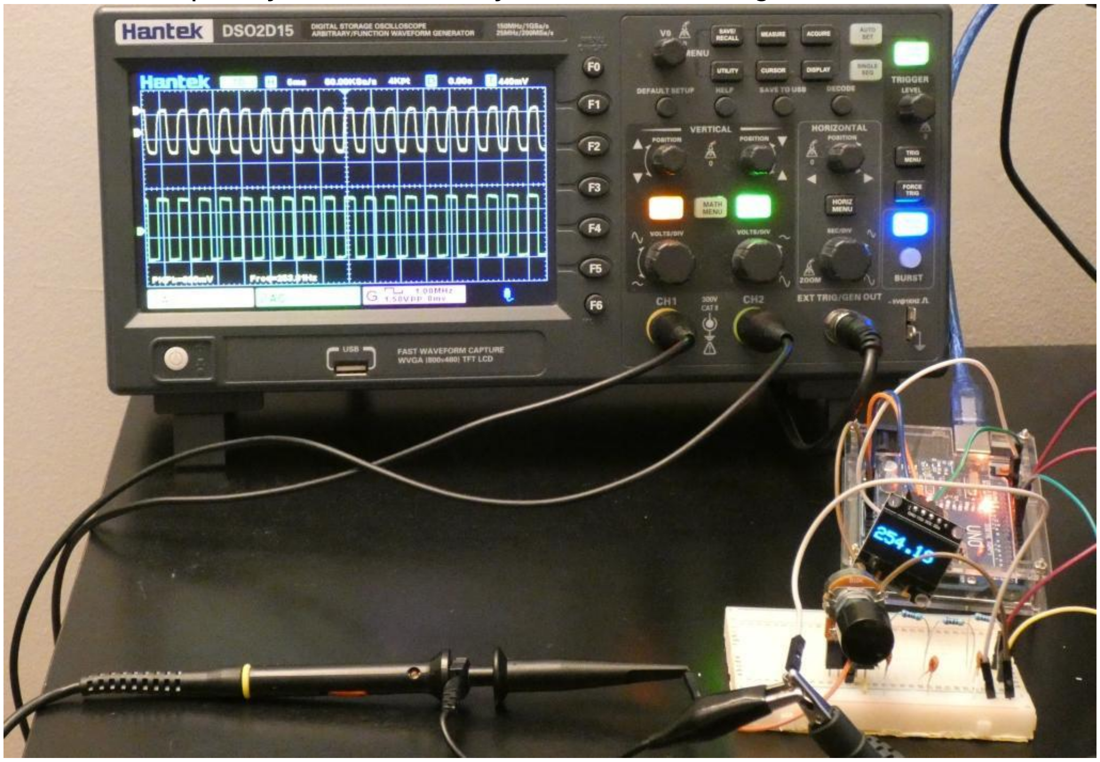
The schematic with Arduino Nano:



The power supply with 5V is done through the USB port of the Arduino NANO board.

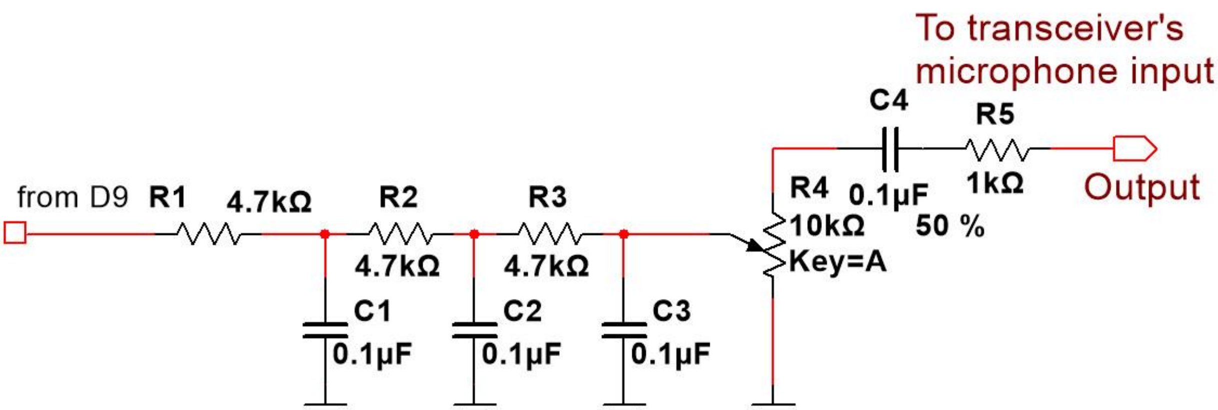
At 67 Hz the maximum output level which can be obtained from the CTCSS generator is 3V peak to peak (1.25V RMS). At 254.1 Hz the output is 2.52V peak to peak, that meaning 1.08V RMS. It is more than enough for inputting this signal in parallel with the microphone of the transceiver. The before mentioned values were measured on the oscilloscope which has high impedance output, and the values will be lower when connecting the microphone in parallel. Some 50mV to 350mV RMS should be more than enough, and this lower value can be obtained by adjusting the potentiometer from the output of the generator.

S1 and S2 are active in the LOW state. I used the PULLUP option of Arduino, so usually D7 and D8 inputs (where S1 and S2 are connected) stay HIGH. No need for resistors, no need for debouncing. Pressing the either S1 or S2 once makes the Arduino to pass to the next standard CTCSS frequency. When getting at the maximum or at the minimum frequency it knows to stay there and not to go further.



Keeping pressed either S1 or S2 will make the going through the standard CTCSS frequencies faster, UP or DOWN, respectively.

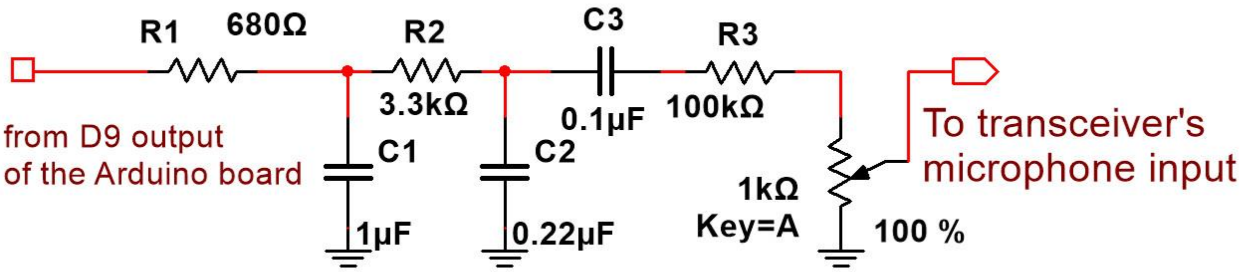
A version for the audio CTCSS tone generator that affects less the transceiver's microphone input, by placing the potentiometer in a different way:



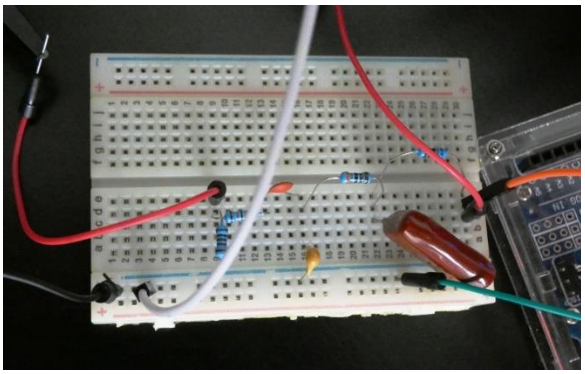
I need to mention here an article written by PA3GUO. He designed an Arduino CTCSS tone generator for 4 tones:

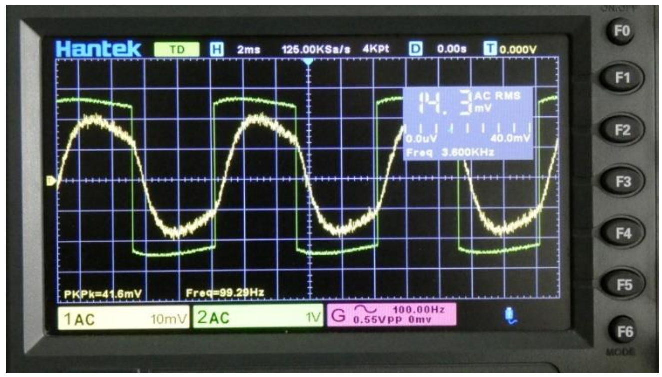
http://www.pa3guo.com/PA3GUO_Arduino_CTCSS_v1.2.pdf

In that article the author cited Joris PE1GLX who designed an aggressive low pass RC filter for transforming the square wave generated by the Arduino board into a more sinus like wave:



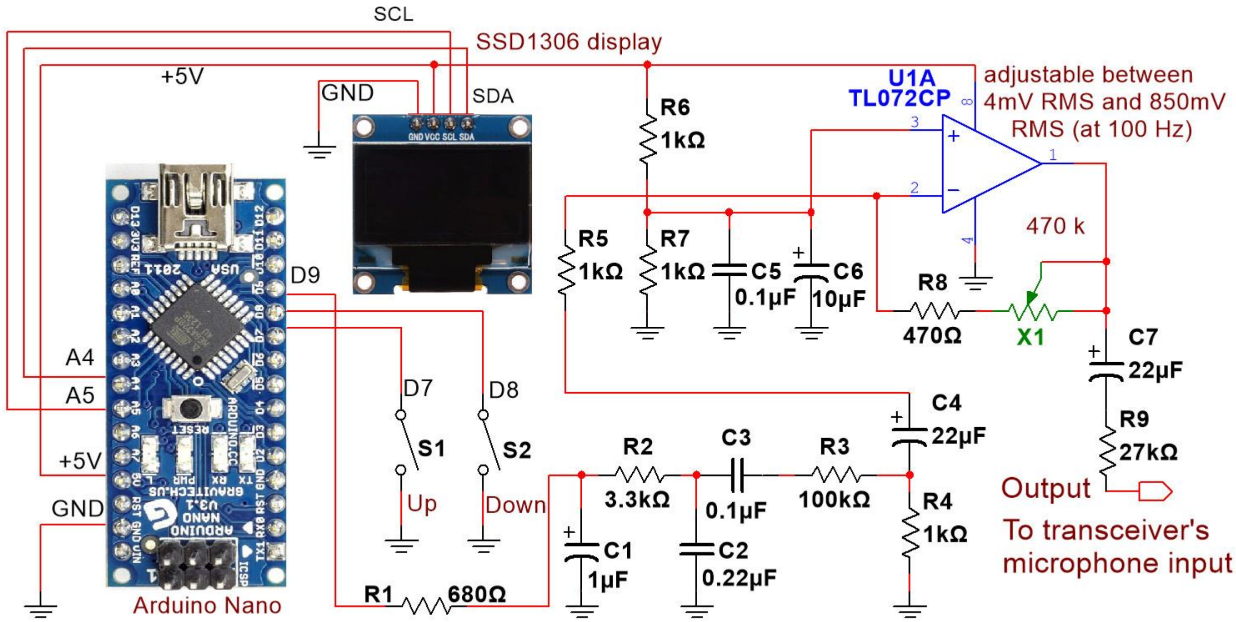
I built on a breadboard this filter. Instead of a 1K potentiometer I put a fix 1 KOhm resistor, because – anyhow – I was interested only in the maximum obtainable level of output signal, not on an adjustable one.



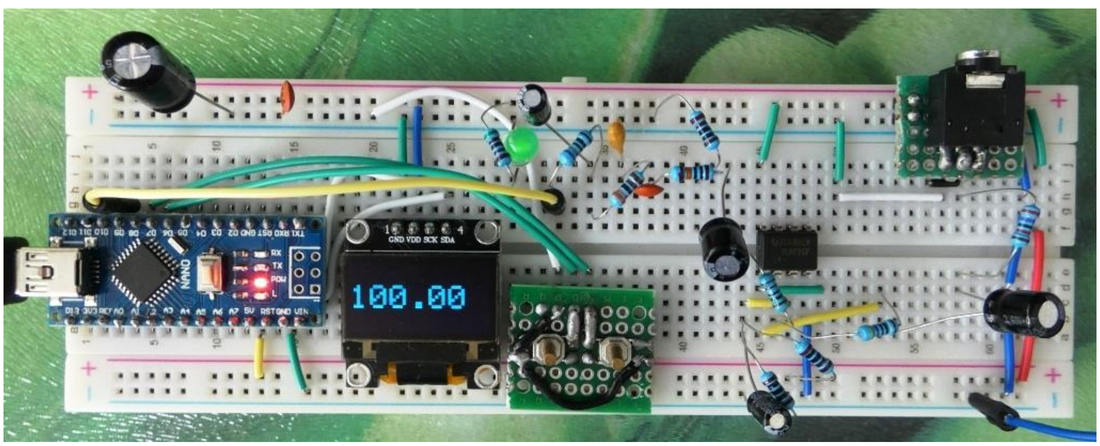


At a CTCSS tone of 100 Hz the maximum output level is 42 mV peak to peak, which means 14.3 mV RMS. My oscilloscope had problems to show the frequency of this signal on the digital voltmeter window, while showing it correctly at the bottom of the screen. That 14.3 mV RMS is the maximum, on the oscilloscope. When connecting at the output the dynamic microphone of a transceiver count on that value to drop further, even some 10 times. It is really necessary to put an amplifier, because 1 – 10 mV CTCSS tone probably will not activate the repeater. Such filter and amplifier put on the board would bring not much improvement in comparison with my version, and it would start amplifying all kind of noises made by power supply and the Arduino board itself. I sticked with my schematic.

But, just for the sake of the experiment I drew a schematic using such filter and I also built it on a breadboard and measured the output:



I used a half of an integrated circuit TL072 containing two operational amplifiers inside. At the output of the OpAmp the signal is between 4mV RMS AC and 850mV RMS AC, measurements done at 100 Hz. I want again to notice such small signals and big amplification obtained with operational amplifier make the whole project sensitive to noise. The power supply must be very well filtered and a metallic case would help, otherwise unwanted signals will be given to the microphone of the transceiver together with the wanted CTCSS tone.



I tried to give others what I was looking for, a tested project which can be replicated immediately, by copying the schematic, the code and adding the same libraries that worked for me.

Note: Please use my code and my libraries, as I used them. This works. Other libraries, with updates and other versions might not work. Everything is available at:

[danielromila/CTCSS-encoder: test (github.com)]

My thanks to VE7RST who helped me do the real radio tests through the VE7RMR repeater. The repeater is also connected to EchoLink and on EchoLink I could hear some “huum” on transmissions (coming from the CTCSS tones) but they cannot be heard at all on real radios.