CTCSS encoder with Arduino

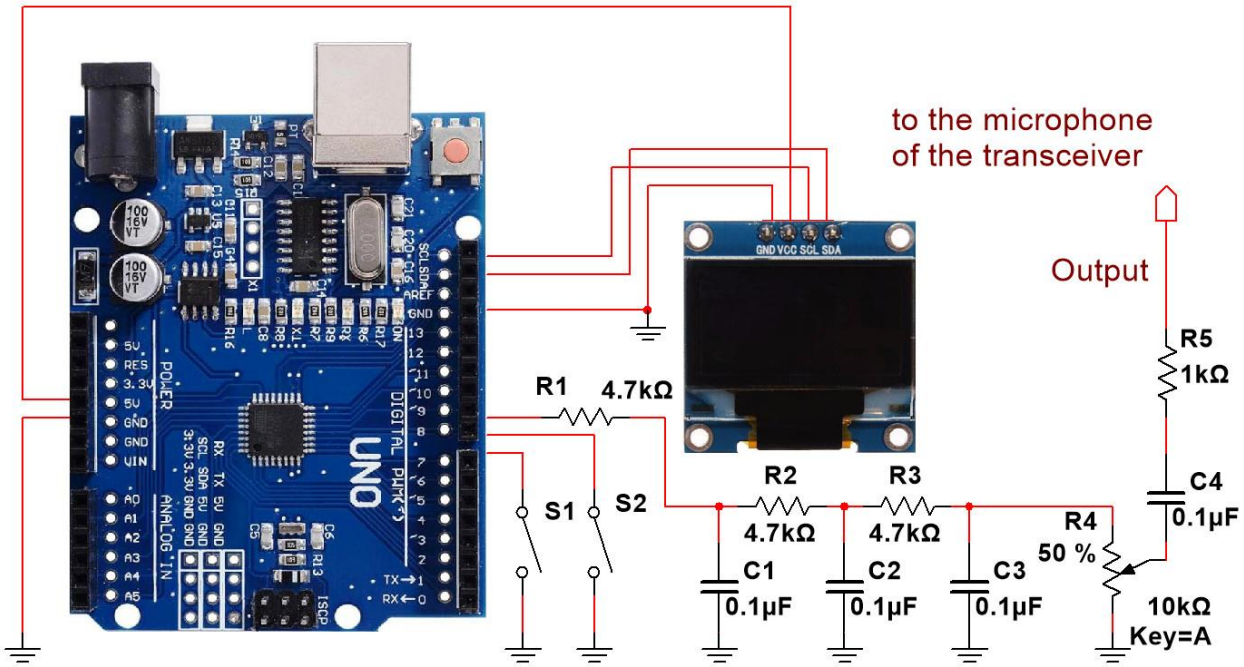
by Daniel Romila VE7LCG

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 to the initial coder(s) on the program I used, which is available from my github:

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

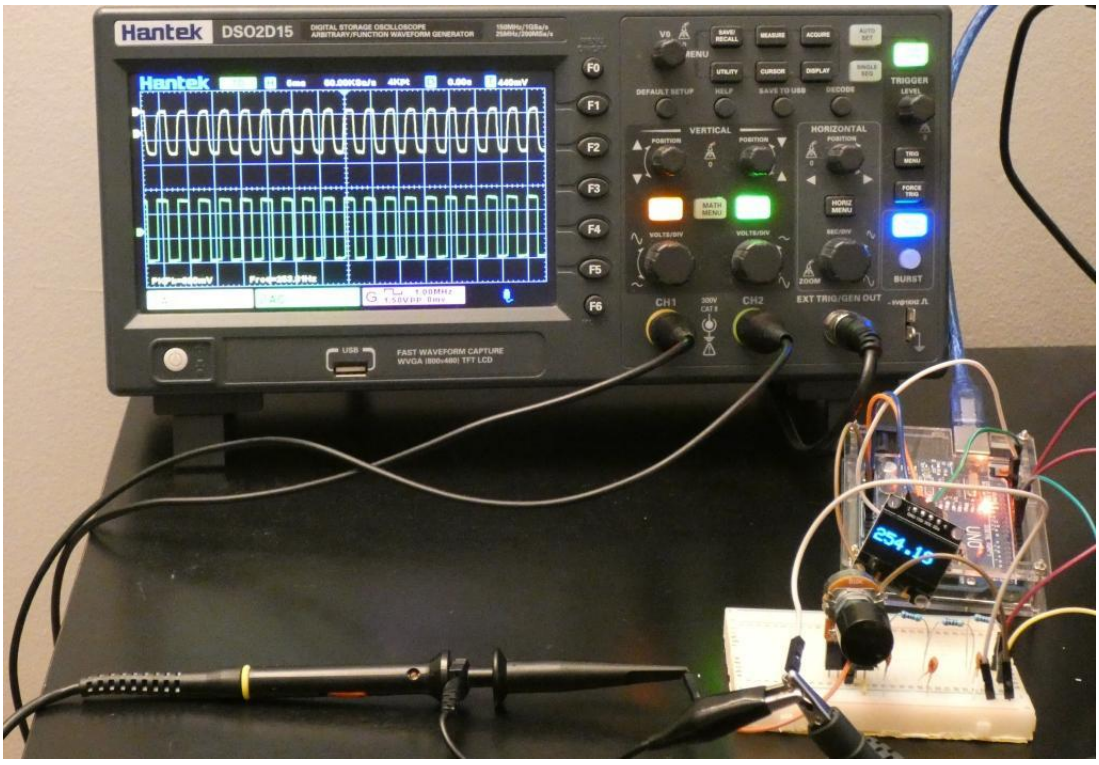
I wanted something simple, and I used Arduino Uno for test, but Arduino Nano also works (and other Arduino boards). I used a 128 X 64 SSD1306 LCD display, which requires I2C connection to the Arduino board. For UP and DOWN (to select the CTCSS frequency, from predefined 50) I used buttons, not a rotary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



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.

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 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 try to move further. Keeping pressed either S1 or S2 will make the going through the standard CTCSS frequencies faster.



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.