CW Decoder

Build this device to decode 5-35 WPM.

There are times when even veteran operators have trouble decoding CW when speeds reach 30-35 wpm. This novel Arduino based decoder can aid both the novice and veteran while providing a chance to get familiar with the Arduino microcontroller and programming language.

Circuit Description

The decoder is based on the work of OZ1JHM[1], with later refinements by K2JJI[2]. The Arduino microcontroller is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. The Nano model also contains an integrated USB port for programming and serial monitoring.

Controls and I/O

Vector board mounted controls include display Contrast (R2), display Back Light (R3), and Bandwidth/Target Frequency dip switch DS1. USB programming is accomplished via the Nano USB port. I also tested a much lower priced Arduino Nano, (Deegoo Mini Nano V3.0 ATmega328P Microcontroller Board w/USB Cable) from Amazon. This chip worked as well as the one in the Bill of Materials (BM).

Microcontroller (U1, Arduino Nano V3.0)

Microcontroller	Atmel ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (of which 2KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Dimensions	0.70” x 1.70”

Microphone Module (M1)

This small breakout board couples an electret microphone (100Hz-10kHz) with a 60x mic preamplifier to amplify any sounds loud enough to be picked up by a microcontroller’s analog-to-digital converter. The module comes fully assembled and works from 2.7V up to 5.5V.

Power (U2, LM7805)
12VDC power is supplied via J1 and is regulated to 5V by this voltage regulator. Diode D2 has been added for reverse battery protection. The diode is reversed biased by normal polarity when placed in parallel with the circuit. When the voltage is reversed, the diode conducts and clamps the reverse voltage to no more than one diode drop (0.7V). A small TO-220 heatsink was added to aid in heat dissipation. A red LED (D1) indicates power.

Enclosure
I chose a clear plastic enclosure to house the vector board and display. The display is viewable through the plastic and the only holes required are for the vector board standoffs, M1, the Arduino Nano USB port, and the 12VDC power jack J1.

Assembly
The circuit was built using perforated vector board and point to point wiring. Using the supplied template, cut the vector board to fit. Layout is not critical except for the location of the 16 pin connectors used to connect the display to the main board. These connectors need to be located so that they don't interfere with each other when the display board is attached to the vector board.

Setup and Operation
Download the Arduino IDE software[3]. After assembly, download the CW Decoder code[4] and upload it to the Nano via the Nano USB port. Ensure that the USB cable is a true USB to mini-B USB cable. Using the Arduino IDE software, I had to list the processor as ATMega328P (Old Bootloader) to get the Nano to load. (Go to Tools, Processor to select). In order to get the Bandwidth/Target Frequency values as listed in the K2JJI chart, I modified six lines of code and bumped the software version to 1.3:

```
if (digitalRead(A0) && digitalRead(A1))       // Mod by KG4JJH //
  n=48;                                       // Mod by KG4JJH //
else if (!digitalRead(A0) && digitalRead(A1))
  n=64;                                       // Mod by KG4JJH //
else if (digitalRead(A0) && !digitalRead(A1))
  n=96;                                       // Mod by KG4JJH //
else
  n=128;                                      // Mod by KG4JJH //

if (digitalRead(A2) && digitalRead(A3))       // Mod by KG4JJH //
  target_freq=496.0;
```

DIP Switch DS1 Settings										
A0 (DS1-1)	OFF	ON	OFF	ON		A2 (DS1-3)	OFF	ON	OFF	ON
A1 (DS1-2)	OFF	OFF	ON	ON		A3 (DS1-4)	OFF	OFF	ON	ON
Bandwidth	186 Hz	139 Hz	93 Hz	70 Hz		Target Frequency	496 Hz	558 Hz	744 Hz	992 Hz

Values are chosen by switching each dip switch (DS1) on or off. This corresponds to setting the A0-A3 pins to ground and +5V, respectively. These values are read only during powerup, so you will have to remove power to the decoder and then power up to see DS1 switch changes. I used the W1AW Code Practice MP3 files for testing the decoder.[5]

Selection of TF and BW depends on the CW sidetone frequency. Lower frequency=TF-BW/2, and the upper frequency=TF+BW/2. Use the following chart to choose dip switch settings (all values are in Hz):

BW	TF	TF Low	TF High	BW	TF	TF Low	TF High	BW	TF	TF Low	TF High	BW	TF	TF Low	TF High
70	496	461	531	70	558	523	593	70	744	709	779	70	992	957	1027
93	496	450	543	93	558	512	605	93	744	698	791	93	992	946	1039
139	496	427	566	139	558	489	628	139	744	675	814	139	992	923	1062
186	496	403	589	186	558	465	651	186	744	651	837	186	992	899	1085

As an example, the W1AW practice MP3 files are at approximately 750 Hz. Choose TF=744, and BW=186.

Results

Thanks to OZ1JHM for the original design and KJ2II for further enhancements. I was very pleased to get the decoder to work for 5-35 wpm signals. I have not been able to get 40 wpm but will continue to experiment.

Happy building and I hope to hear you on the air!

Allen Baker, KG4JJH
211 Brochardt Blvd.
Knoxville, TN 37934
http://www.kg4jjh.com/
kg4jjh@arrl.net

Attachments

Attachment 1: Assembly and Vector Board Details Drawing
Attachment 2: Schematic Drawing
Attachment 3: Materials List
CW Decoder V1.3 code

Acknowledgements

[1] *Easy Build CW Decoder Based on DSP Goertzel Code,*
http://www.skovholm.com/content/very-simpel-cw-decoder-easy-build
[2] *Arduino Based CW Decoder,*
http://k2jji.org/2014/09/18/arduino-base-cw-decoder/
[3] *Arduino IDE program,*
https://www.arduino.cc/en/main/software
[4] *CW_Decoder_V1.3*,

CW_Decoder_V1.3.ino

[5] *W1AW Code Practice MP3 Files,*
http://www.arrl.org/code-practice-files