

Arduino CW Trainer

Hone your Morse code skills with this entry-level project.

Tom Lewis, N4TL

Many years ago, I wanted to write a CW training program that would send a few characters of Morse code and then listen for me to send the same characters back with my Morse key. If I replied correctly, the program would go on to new characters. If I got it wrong, it would tell me I made a mistake and send the same ones again. A person using the trainer should not write anything down, but rather listen to the code, remember what had been sent, then send it back to the trainer. You learn to receive

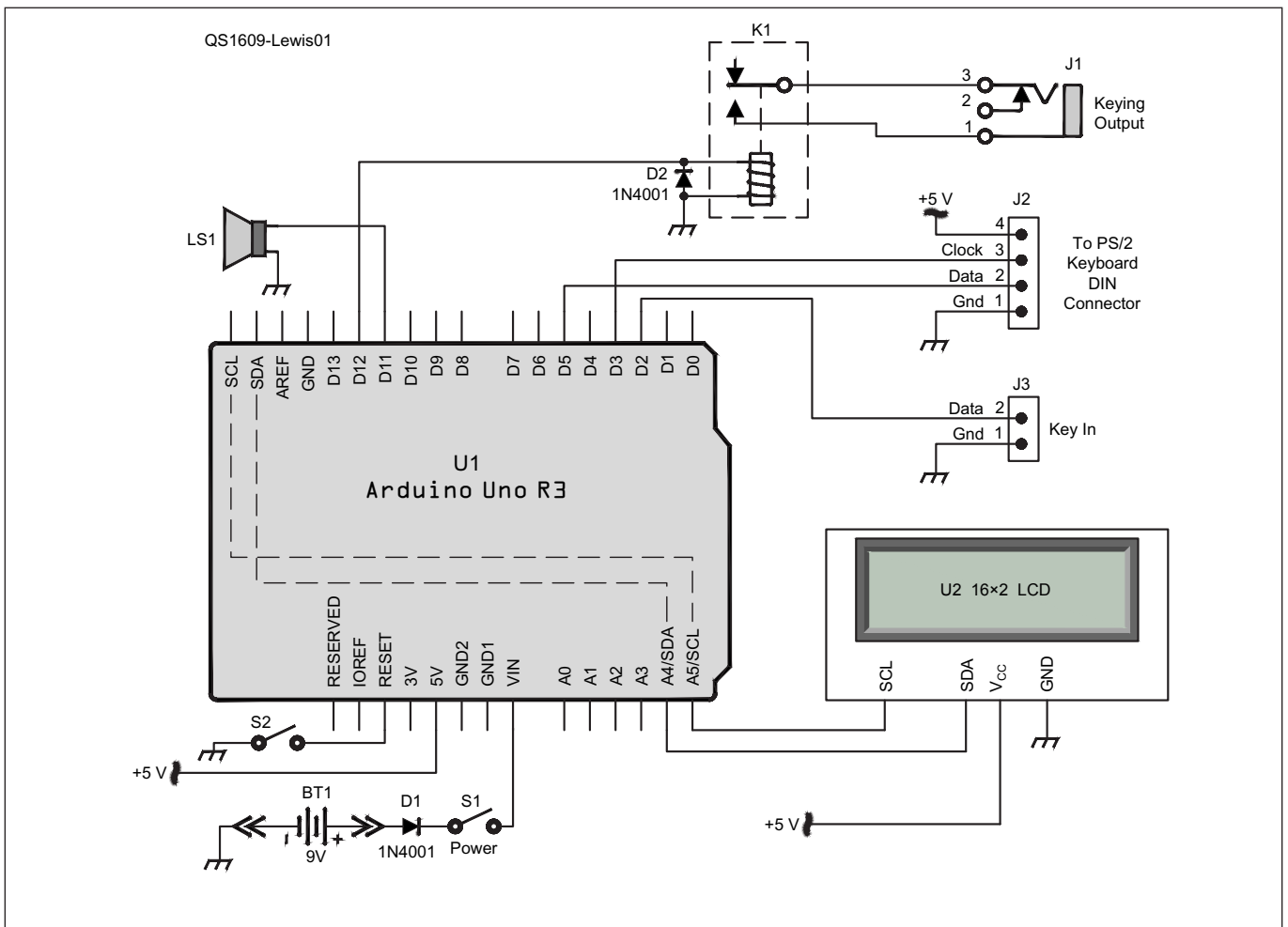
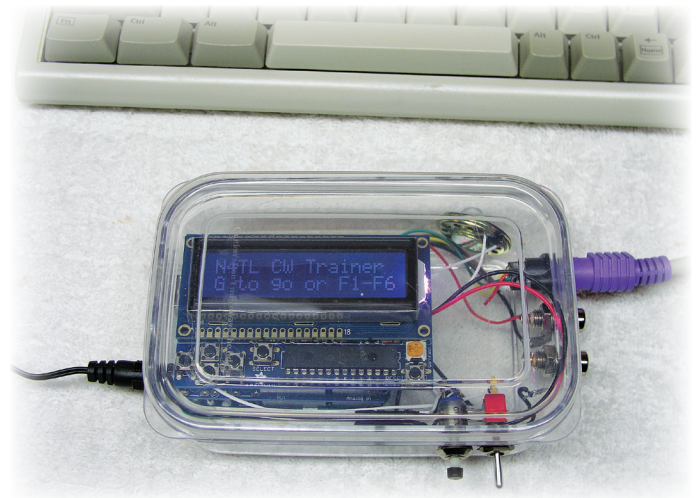


Figure 1 — Schematic of the CW Trainer. (PID numbered parts from www.adafruit.com.)

BT1 — 9 V battery, or ac adapter PID 63
 D1, D2 — 1N4001 Diode, PID 755. D1 is needed only if a battery is used
 J1 — Mini output jack
 J2 — PS/2 wired connector, PID 804; green is power, black is ground, yellow is clock, and

brown is data; the red and white wires are not used
 J3 — RCA jack, keying input
 K1 — RadioShack 5 V reed relay
 LS1 — Mini speaker, PID 1890
 S1 — SPST switch
 S2 — Momentary contact pushbutton

U1 — Arduino Uno R3, PID 50
 U2 — RGB LCD Shield Kit with 16 x 2 display PID 714 — requires soldering; pushbuttons are not used in this project
 U3 — Adafruit Proto Shield kit R3, PID 2077; not shown; goes between the Arduino and the LCD display shield

and send the code entirely by sound.

Recently, I examined *Arduino for Ham Radio* by Glen Popiel, KW5GP, and realized I could write the CW training program for the Arduino.¹ Arduino provides a free Arduino Development Environment (ADE) to develop sketches (programming code).² The ADE compiles and uploads the code to the Arduino through a USB cable. The Arduino uses open-source code, meaning that the code is freely available for you to use or modify.

Figure 1 shows the schematic of the CW Trainer based on U1, the Arduino Uno R3 and U2, the 16-character by two-line LCD display. This kit has an IC that converts the many wires from the LCD to just four — two for power and ground and two for signaling. The IO shield, U3 (not shown), goes between the Arduino and the LCD display shield. It has a reed relay and places to connect all the wires to the switches and jacks.

The library for the Adafruit LCD shield kit can be downloaded from the Adafruit website.³ The LCD display also has a library of code that supports its operation. Because my LCD is different than the one Glen uses in his book, I needed to install the new library in the ADE and make changes in the sketch to use the correct LCD library.

Figure 2 shows my original breadboard with the LCD display connected by jumper wires to the Arduino board. The lead photo shows the final assembly housed in the food container. I left room for a 9 V battery, but I found that the battery did not last very long. I use an external power supply that plugs into the Arduino. I used some cotton to hold the speaker in place.

The Sketch

My sketch, which uses parts of the sketches in Glen’s book, is available on the *QST* in Depth web page.⁴ One part is from Chapter 19, “PS/2 CW Keyboard” and the other is from Chapter 22, “CW Decoder.” I wanted to generate and send random characters, so I used the random number generator built into the Arduino language. I wanted to associate a random number from 0 to 39 with 40 Morse characters. In my sketch, I first associate the 10 numbers with 0–9, the 26 alphabet characters with 10–35, and the comma (,), period (.), slash (/), and question mark (?) with 36–39.

The training sketch first runs some setup code, then runs a keyboard loop. The keyboard is used to set parameters in the program according to the keyboard commands in Table 1. The selected function is

displayed on the LCD. I limited the code speed to between 20 and 30 WPM. I think 25 WPM is a good speed for learning the sounds of the characters.

**Table 1
Keyboard Keys and Their Function**

Key	Function
Up arrow	Increases the sending speed
Down arrow	Decreases sending speed
Right arrow	Increases the number of characters sent before the Arduino checks for incoming characters sent by the learner
Left arrow	Decreases the number of characters sent before the Arduino checks for incoming characters sent by the learner
F1	Sets the character set to the 26 letters of the alphabet
F2	Sets the character set to the 10 numbers
F3	Sets the character set to “period,” “comma,” “slash,” and “question mark”
F4	Sets the character set to all 40 characters listed in Table 2
F5	Number of Koch method characters — enter 1 or 2 digits followed by the ENTER key
F6	Number of characters to skip over in Koch method of Table 2 — enter 1 or 2 digits followed by the enter key
F9	Toggle between the internal speaker or relay output
F10	Saves the parameters in EEPROM. F10 must be pushed before pushing G
G	“Go” — starts sending code characters via the speaker or relay
D	“Decode” runs the CW decoder only

**Table 2
The Numbers 1 to 40 Associate with the Character Below in the Koch Method**

1	2	3	4	5	6	7	8	9	10
K	M	R	S	U	A	P	T	L	O
11	12	13	14	15	16	17	18	19	20
W	I	.	N	J	E	F	0	Y	V
21	22	23	24	25	26	27	28	29	30
.	G	5	/	Q	9	Z	H	3	8
31	32	33	34	35	36	37	38	39	40
B	?	4	2	7	C	1	D	6	X

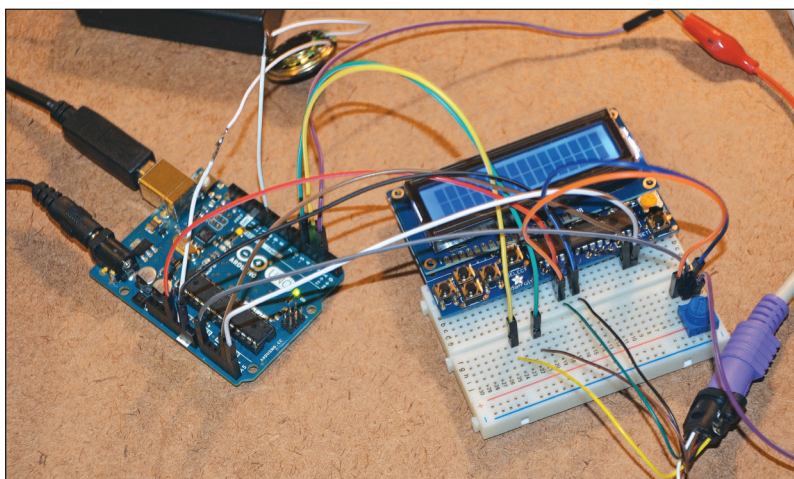


Figure 2 — My original breadboard shows the LCD display connected by jumper wires to the Arduino board.

Operating the CW Trainer

Turn the trainer on, then push one of the function keys to select the character set. Set the speed and number of characters to be sent, and push the G key. The trainer will send a few characters and wait for you to send them back using your Morse key. The trainer checks each character as it is received. When an incorrect character is received, the background color of the LCD turns red and the trainer sends the same characters again. If all the characters are correct, the LCD's background will stay white and send new characters. The characters are displayed on the LCD. This operation will continue until the trainer is turned off or the reset button is pushed. When the trainer is turned on again, it will load previously saved parameters from EEPROM.

I implemented the Koch method in my sketch — see the G4FON web page — that teaches by sending a single character at full speed.⁵ Then a second character is added, and as each character is learned, another one is added. To do this with my sketch, I associated a second number to the character table (see Table 2) in the same order as in the G4FON program. Table 2 starts with *K* and ends with *X*. When F5 is pushed the CW Trainer asks on the LCD display for a one- or two-digit number followed by the enter key to set the number of characters to be used. If “2” is entered, only *K* and *M* are sent. If “4” is entered only *K*, *M*, *R*, and *S* are sent, and so on.

By the time you are up to 30 characters or so, you will know the first characters very well and they need not be sent very often. F6 is used to skip over the characters at the

By the time you are up to 30 characters or so, you will know the first characters very well and they need not be sent very often.

beginning of the table. When F6 is pushed, the CW Trainer will ask for a one- or two-digit number followed by the ENTER key. This number is the starting index of Table 2. If F5 is pushed and “3 6” is entered, then F6 is pushed and “3 1” is entered, only *B*, *?*, *4*, *2*, *7*, and *C* will be sent. This corresponds to entries 31 to 36 in Table 2.

I added a D option to the keyboard control so you can practice sending to the CW Trainer. When D is typed after power-on or a reset, the Morse code decoder will be run by itself. You can send CW and look at the LCD to discover how well it decodes your sending.

I tested the decoder by sending the alphabet to it from a *WinKey USB Keyer* at various speeds, and found that if the LCD was scrolled, the Arduino CW Trainer would make mistakes at higher speeds. I removed the scrolling and found that the decoder will work up to about 22 WPM on my 16 MHz CPU. Higher speed CPUs should allow faster decoding of CW. Leaving a little space between characters will improve decoding at higher speeds.

When compiling the sketch, make sure that the PS2 keyboard library has the function keys defined. The text below is copied from the first four lines of the PS2Keyboard.cpp file that I used.

PS2Keyboard.cpp — PS2Keyboard library
Copyright (c) 2007 Free Software Foundation. All right reserved. Written by Christian Weichel <info@32leaves.net> Modified to add F1-F12 keys and other minor corrections by Glen Popiel — KW5GP.

Final Words

I am not an expert on learning Morse code, and built this CW Trainer mainly to learn about the Arduino. For information on learning Morse code, see ARRL's “Learning Morse Code” page.⁶

I would like to thank my wife Jan, as well as Ted Webb, W4NE, for reading this article and helping me improve it.

Notes

- ¹Glen Popiel, KW5GP, *Arduino for Ham Radio*, ARRL Item no. 0161, available from your ARRL dealer, or from the ARRL Store. Telephone toll-free in the US 888-277-5289, or 860-594-0355, fax 860-594-0303; www.arrrl.org/shop/; pubsales@arrrl.org.
- ²www.arduino.cc/en/main/software
- ³<https://www.adafruit.com>
- ⁴www.arrrl.org/qst-in-depth
- ⁵www.g4fon.net
- ⁶www.arrrl.org/learning-morse-code

Amateur Extra class license holder and ARRL Life Member Tom Lewis, N4TL, has been licensed since 1967. His early Amateur Radio interests led to BSEE and MSEE degrees from the State University of New York at Buffalo. After college, he worked for Stromberg Carlson in Rochester, New York, and then OKI Electronics in Fort Lauderdale, Florida. Tom joined IBM in Boca Raton, Florida in 1984, where he was one of the engineers who designed the Video Graphics Array (VGA). He has 14 patents. Tom retired in 2014 and now spends time with his family and local ham radio clubs, fixes vintage radios, and works the NPOTA stations. He has chased DX for many years and has confirmed all the DXCC entities. You can reach Tom at n4tl@arrrl.net.

For updates to this article, see the *QST* Feedback page at www.arrrl.org/feedback.

VOTE
Did you enjoy this article?
Cast your vote at
www.arrrl.org/cover-plaque-poll

Ham Radio is for Everyone – Including You!

This article highlights only one aspect of Amateur Radio. Whether you're interested in electronics or technical experimentation, being of service to your community, connecting with people worldwide, or you just want reliable communication available to you at any time, Amateur Radio offers all this and much more.



Learn more! <http://www.arrrl.org/what-is-ham-radio>

ARRL The national association for
AMATEUR RADIO®
www.arrrl.org

