How to build a Simplified Arduino CW Trainer by Tom N4TL

See page 17 for V2.1 information.

See pages 18—19 for information about a version that does not need a keyboard.

The September 2016 issue of QST has my original article about the Arduino CW trainer. That article does not have too much information on how to build the trainer. More information is in this paper. Figure one shows the original CW trainer, figure two shows a simplified version. The simplified version is easier to build because it does not have a relay output or a connection for a nine volt battery through a switch. Both are not need-

Continued on page 3

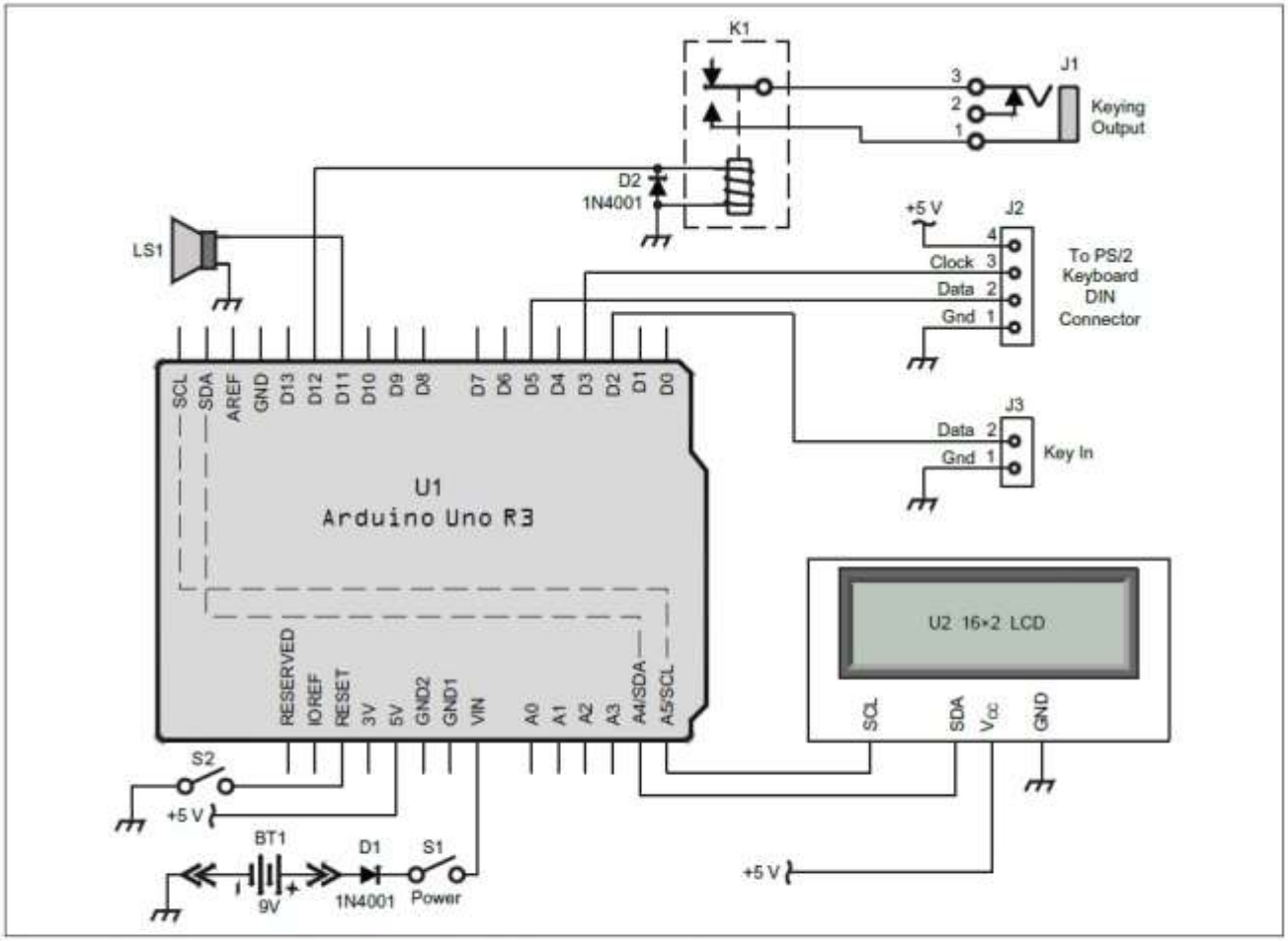


Figure 1 — Schematic of the CW Trainer. (PID numbered parts from **www.adafruit.com.**)

BT1 — 9 V battery, or ac adapter PID 63
D1, D2 — 1N4001 Diode, PID 755, 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
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
S2 — Momentary switch

Schematic of the original Arduino CW Trainer.

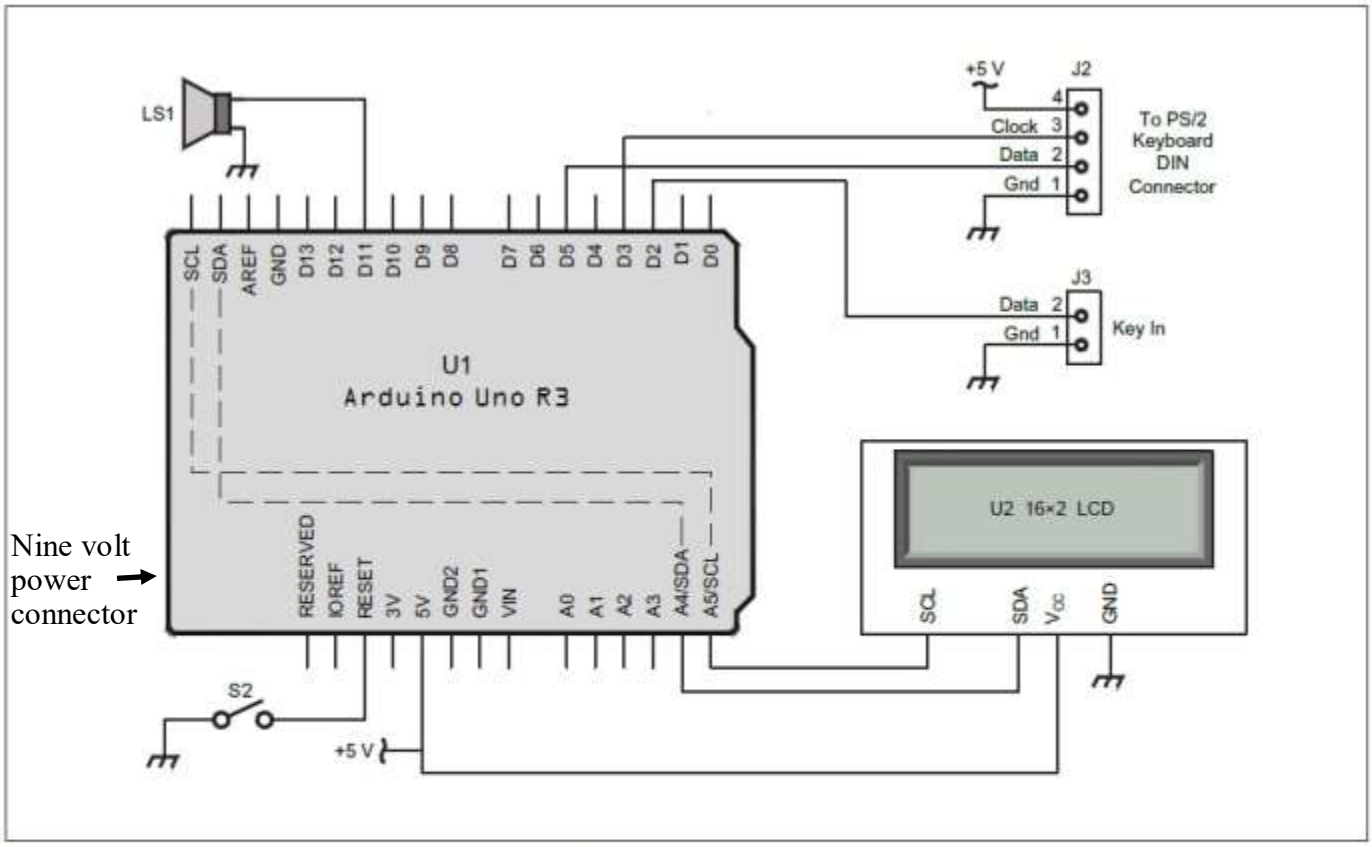


Figure 2 — Schematic of the CW Trainer. (PID numbered parts from **www.adafruit.com.**)

BT1 — 9 V battery, or ac adapter PID 63

J1 -- Not used

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

LS1 — Mini speaker, PID 1890

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

S1 -- Not used

S2 -- Momentary switch

Schematic of the simplified Arduino CW Trainer.

Figure 3, Nine Volt Power Adapter, PID 63

Figure 4, Nine Volt Battery Case, PID 67

ed for the trainer. The speaker will provide the audio output. The power is supplied by an AC adapter or an external nine volt battery. Adafruit has a nine volt battery case with a plug that will plug in to the Arduino power connector. They also sell an AC adapter that can be used. See Figures 3 and 4 for pictures of them. When searching for parts at the adafruit website, enter the product number in their search box without a pid in front of it. I have no connection to adafruit. You can buy parts anywhere but my LCD code works with the adafruit LCD display and may not work with other displays. I have Hyperlinks for many of the parts in this document. To open a Hyperlink just click on the blue underlined text.

Figure 5, New Arduino pid 50 powered by a nine volt battery in a case

Figure five shows a nine volt battery in a case powering a brand new Arduino. One green LED is on and one yellow one is blinking. The brand new Arduinos are programed to blink the yellow LED.

Here is an easier to read parts list for the Simplified CW Trainer.

BT1 Power can be supplied by either of these.
Nine Volt Power Adapter, adafruit PID 63.
Nine Volt Battery Case, adafruit PID 67.
J2 PS2 Keyboard connector adafruit pid 804.
J3 RCA connector, for keying input.
LS1 Speaker, adafruit pid 1890.
S2 Momentary push button switch for reset.
U1 Arduino adafruit pid 50.
U2 RGB LCD Shield with a 16x2 character display, adafruit PID 714.
U3 Proto Shield kit, adafruit, PID 2077.

- PS2 keyboard.
- Keyer and paddle. A straight key can be used but it is better to learn how to use a paddle and Keyer. The Keyer needs to have a side tone.

I used a clear plastic Lustroware 13 oz. food container to hold the completed CW Trainer. The food container was the right size for the project and it was easy to drill holes in it. I bought it at The Container Store here in Raleigh.

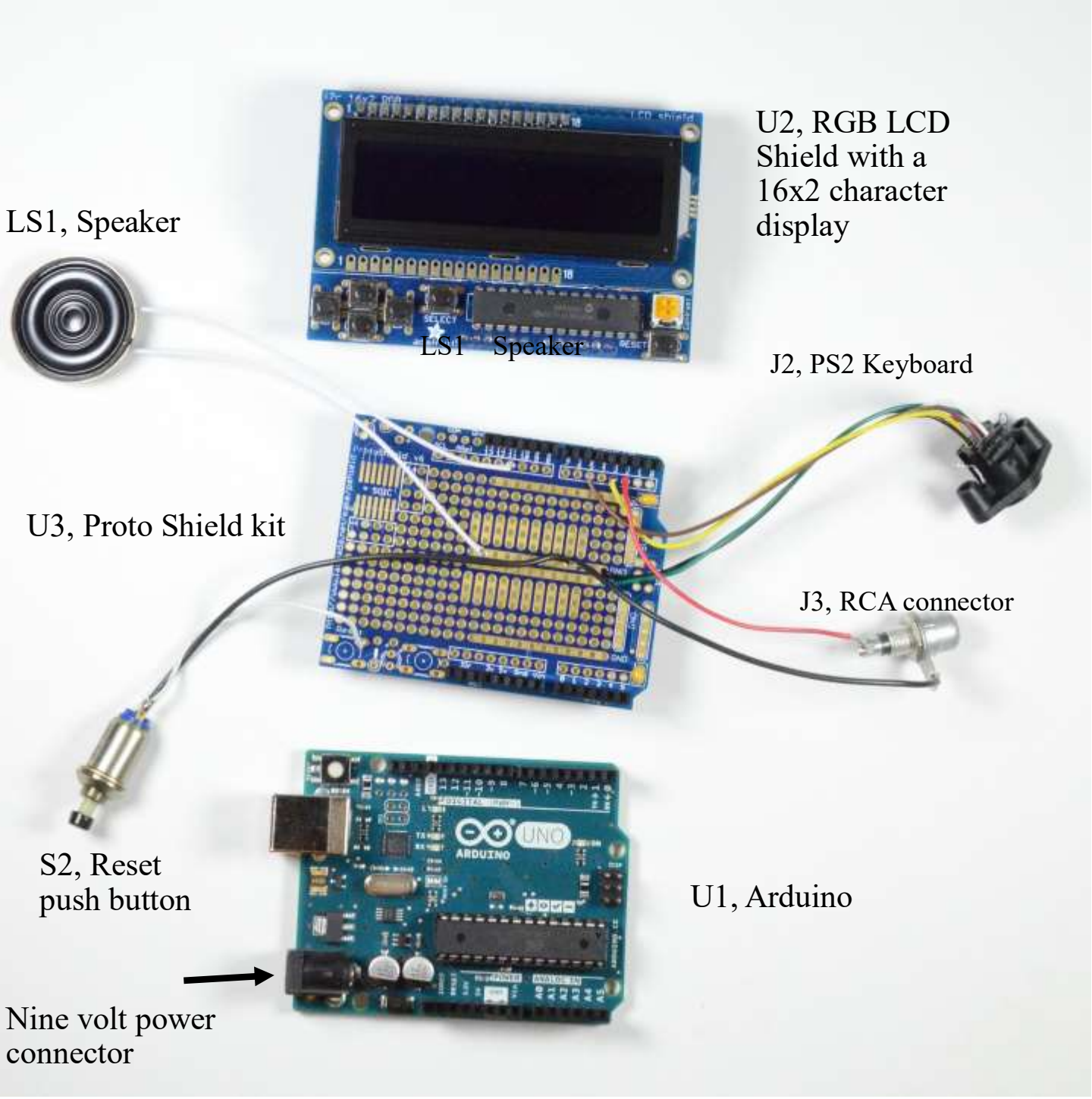


Figure 6, The three circuit boards that make the Arduino CW Trainer.

Figure 6 shows the three circuit boards that make the Arduino CW Trainer. J1 and S1 are not used.

Figure 7, The three circuit boards connected to each other and turned on.

Figure 8, Side view of the Arduino CW Trainer

The two white wires from the trainer to the speaker don't show up very well. More pictures are on the following pages.

Here is more information on the Proto Shield kit. It provides places to connect the connectors, switch and speaker.

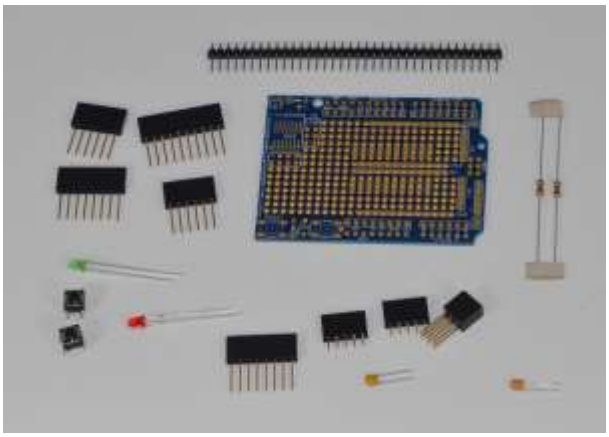


Figure 9, adafruit Proto Shield kit, PID 2077.

Figure nine show the parts that come with the Proto Shield kit, adafruit PID 2077. All the parts are not needed for this project. For information on how to assemble it, go to the adafruit website. On the 2077 product page, slide down to where it says, "**Comes as a kit of parts** so you can choose how you want to configure your shield. However its very easy to assemble even for beginning solderers. More information & instructions are available here. ". Click on More information and down load the PDF file. Read the document and install the stacking headers. Do not use the plain headers shown on page 22 of the document. Install the stacking headers and the two small capacitors. After that follow the pictures on the next few pages.

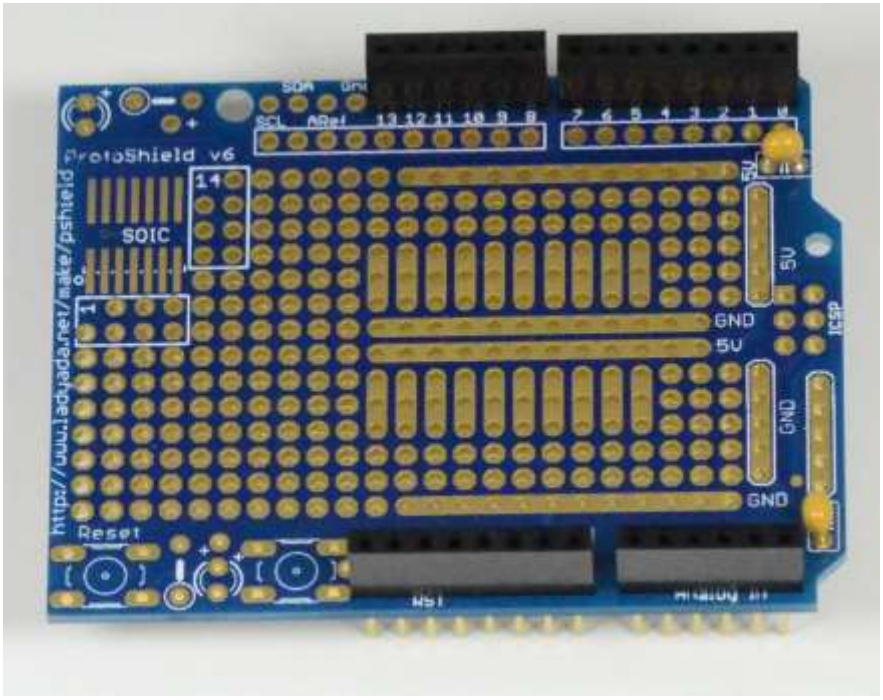


Figure 10, adafruit Proto Shield kit with the stacking headers and two capacitors installed

Now add the speaker and PS2 Keyboard connector.

Figure 11, adafruit keyboard connector pid 804 and speaker pid 1890

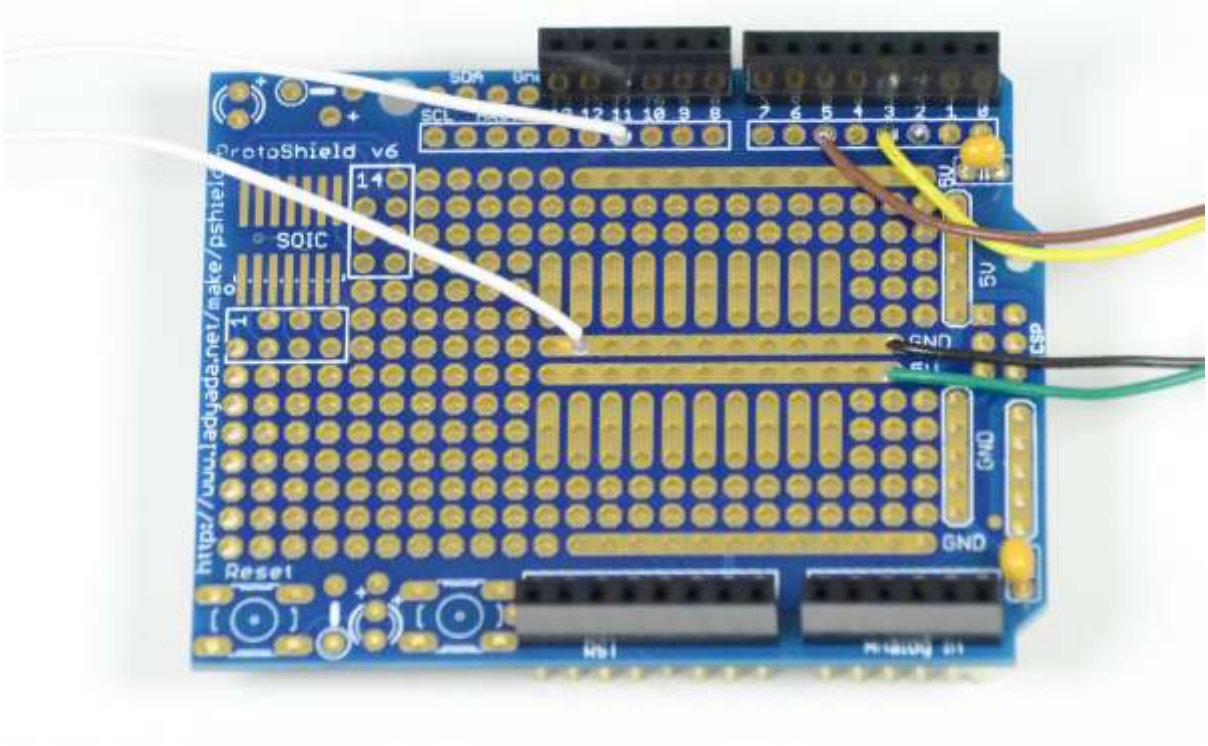


Figure 12, Close up of the keyboard connector wiring and speaker wiring

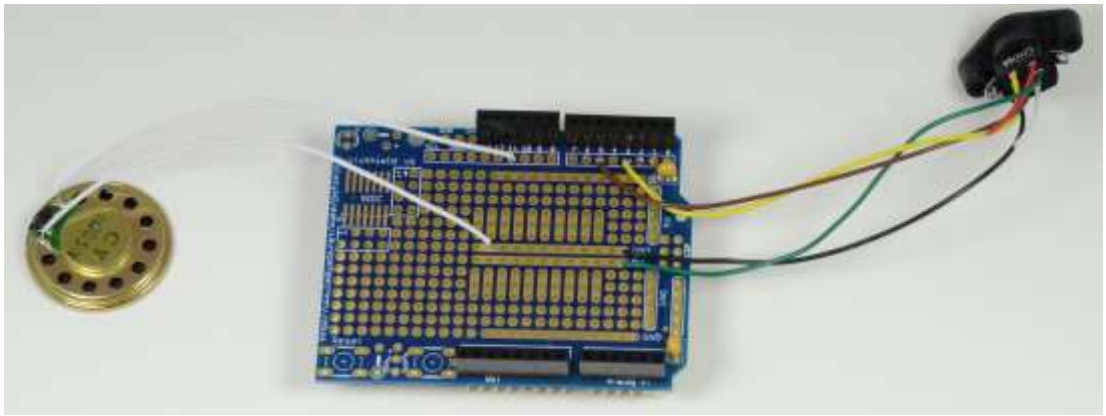


Figure 13, keyboard connector and speaker connected.
The red and white wires on the PS2 connector are not used.

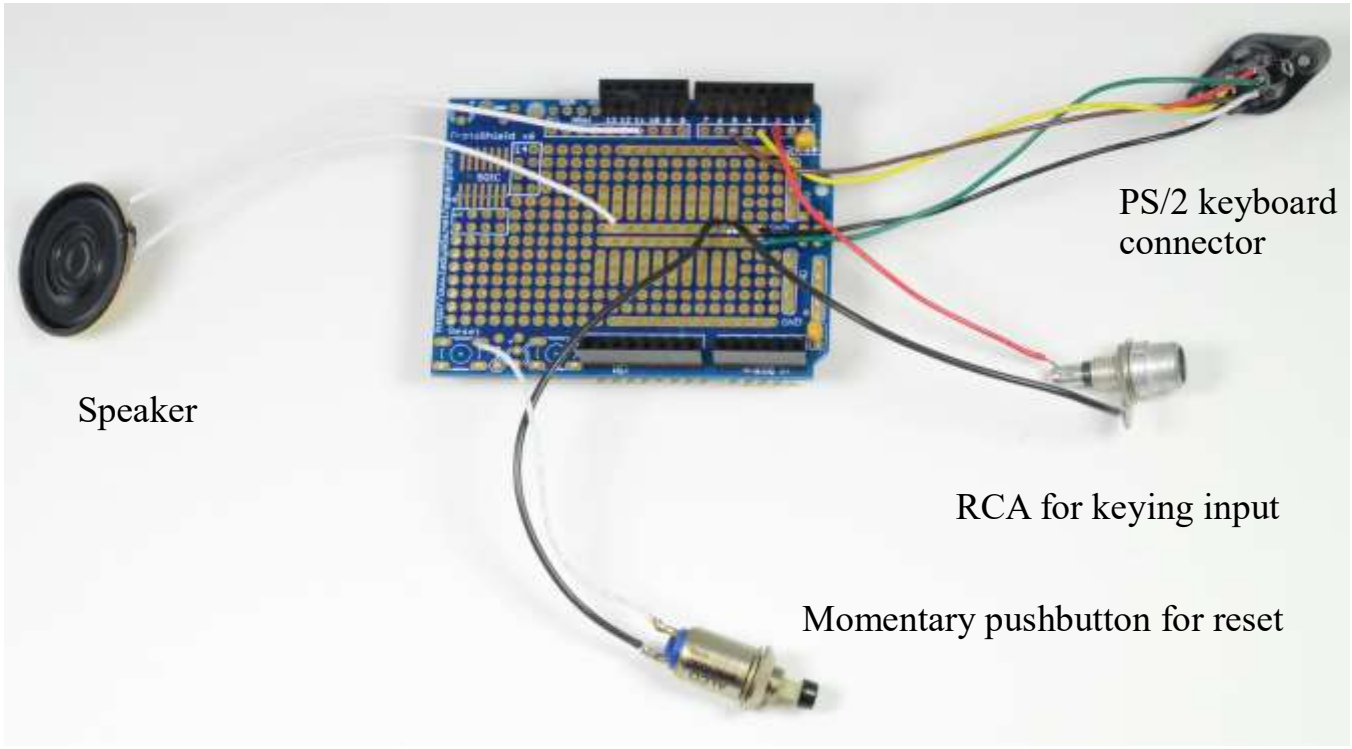


Figure 14, This picture shows the reset pushbutton and the keying input added

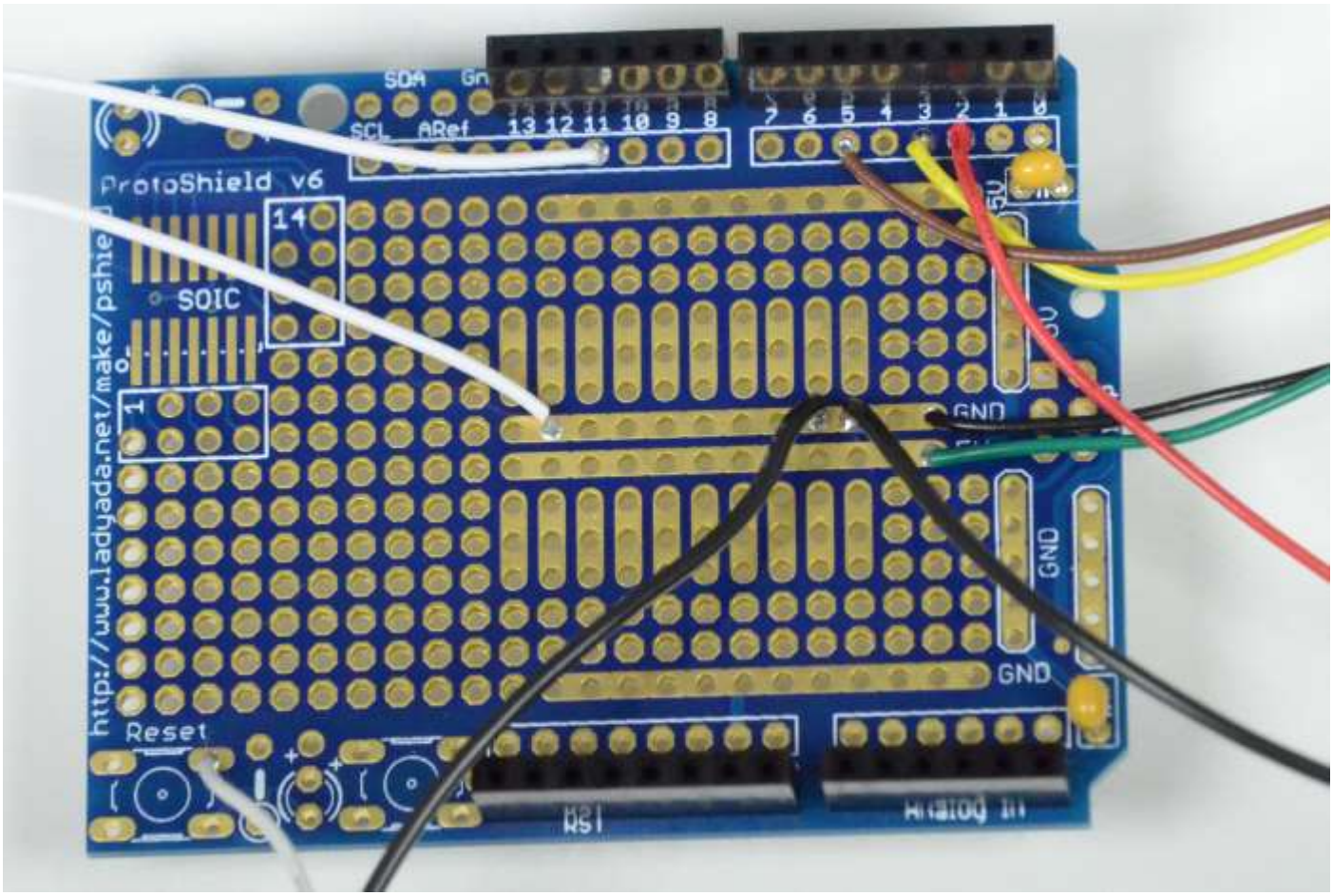


Figure 15, close up of the shield in figure 14

Figure 15 shows the completed Proto shield for the simplified version of the CW Trainer.

Figure 16 shows an assembled RGB LCD Shield with a 16x2 character display. The adafruit website shows the display, PID 714 as a *RGB LCD Shield Kit w/ 16x2 Character Display - Only 2 pins used!* -. It really uses four pins, two for signals, one for power and one for ground. Assembly instructions for the display are at the adafruit website. After it is assembled and turned on don't forget to adjust the contrast. On mine I did not see anything until the contrast was adjusted.

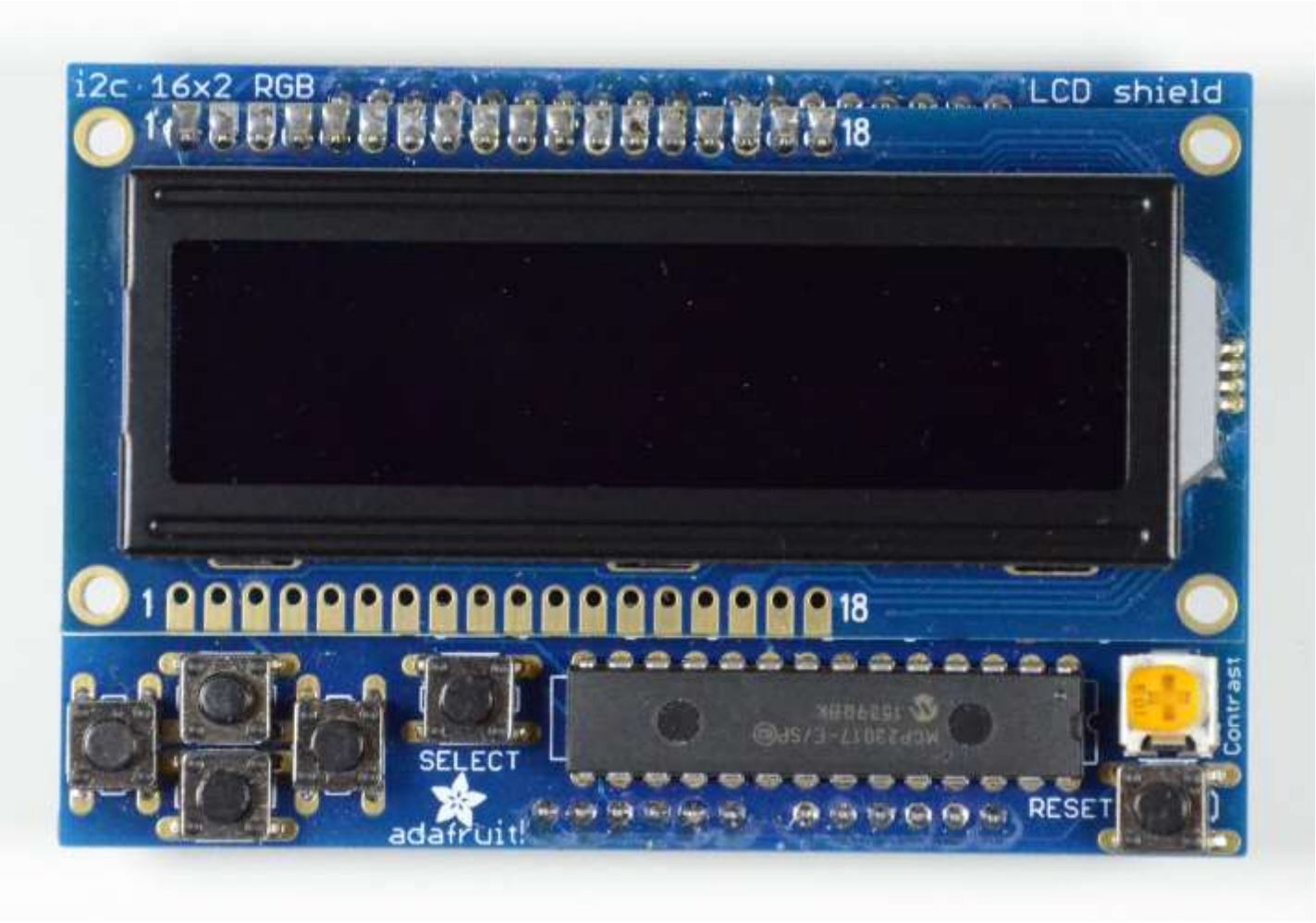


Figure 16, RGB LCD Shield with a 16x2 character display, adafruit PID 714.

All the parts that make this RGB LCD shield are shown on the next page.

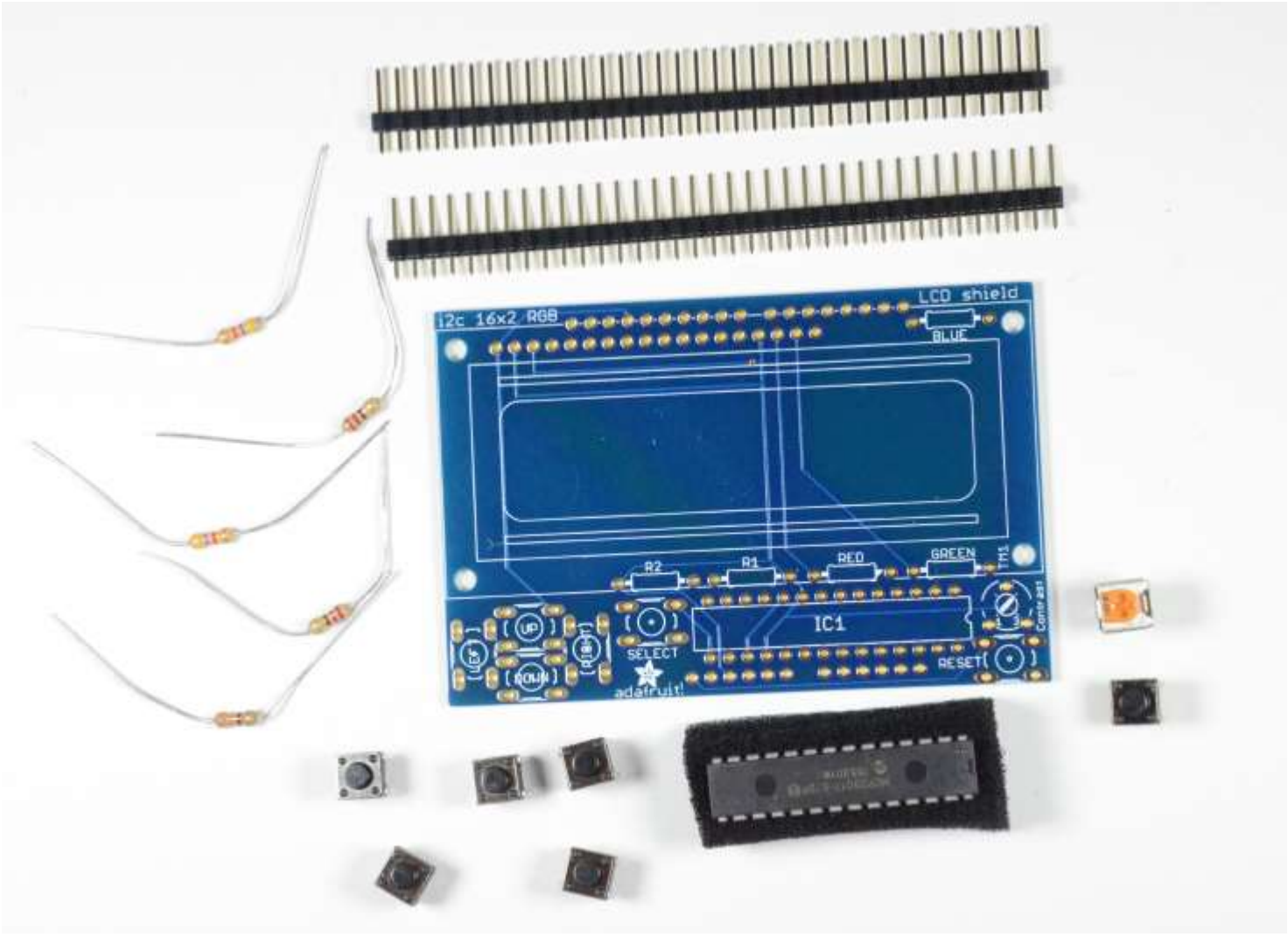


Figure 17, I2C controller and keypad shield kit for 16x2 LCD (PID 715).

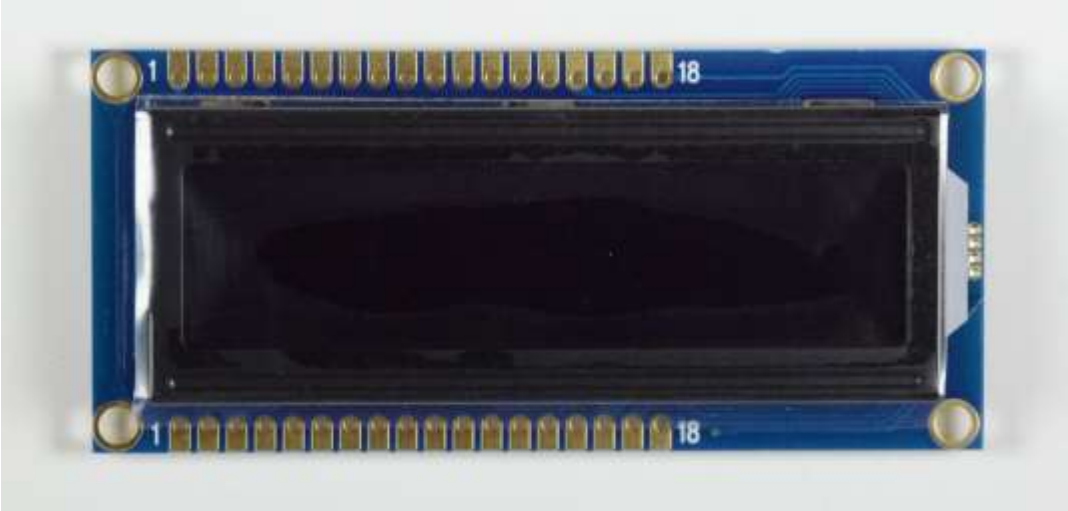


Figure 18, RGB LCD 16X2 (PID 399)

My Parts Costs

Parts cost at adafruit. July 2016

Products
---
1 x 9 VDC 1000mA regulated switching power adapter - UL listed[ID:63] = $6.95
1 x PS/2 Wired Connector - Panel Mount MiniDIN-6[ID:804] = $3.95
1 x Arduino Uno R3 (Atmega328 - assembled)[ID:50] = $24.95
1 x RGB LCD Shield Kit w/ 16x2 Character Display - Only 2 pins used! (NEGATIVE DISPLAY) [ID:714] = $24.95
1 x Adafruit Proto Shield for Arduino Kit - Stackable Version R3[ID:2077] = $9.95

Sub-Total: $70.75
United Parcel Service (1 pkg x 0.98 lbs total) (UPS GROUND): $8.55
Sales Tax: $0.00
Total: $79.30

I already had a Speaker, adafruit pid 1890 ($1.50) and did not buy another one. I also had a momentary push button for the reset and a RCA jack for the keying input.

Other builders Parts Costs

One person that I have been exchanging emails with has built the CW Trainer for less then $10.00. Here are two links he sent me. He had to change the LCD code so it would work with this display

http://www.banggood.com/IIC-I2C-1602-Blue-Backlight-LCD-Display-Module-For-Arduino-p-950726.html

http://www.banggood.com/UNO-R3-ATmega328P-Development-Board-For-Arduino-No-Cable-p-964163.html

Another person told me about this place.

http://www.dx.com/c/electrical-tools-499/arduino-scm-supplies-436

Another person's input.

Digikey stocks the PS/2 connector for $2.47. They will ship via US Mail for a lot less than Adafruit. Digi-Key Part Number CP-2960-ND.

I am sure there are many more places to buy parts.

Programming the Arduino

You will need to program the Arduino with the trainer sketch (program). First you will need the Arduino Development Environment (ADE) on your home computer. Chapter five of the book ARRL book Arduino for Ham Radio has information about it. This web-site also has information, https://www.arduino.cc/en/Guide/Environment.

Here is the sketch and the libraries you will need.

The source code for the Trainer is on the ARRL website. Go to http://www.arrl.org/qst-in-depth and look for the September issue and Trainer Sketch.

The LCD library from the adafruit website https://learn.adafruit.com/rgb-lcd-shield/downloads.

The MorseEnDecoder library found at http://www.w5obm.us/Arduino/. Go to chapter 22 and MorseEnDecoder. Copy the files MorseEnDecoder.cpp, MorseEnDecoder.h and key-words.txt to your Arduino libraries.
C:\Program Files (x86)\Arduino\libraries\MorseEnDecoder\

You will get an error when you compile the program that reads like this.

...\Arduino\libraries\MorseEnDecoder\MorseEnDecoder.cpp:69:19: error: variable 'morseTable' must be const
in order to be put into read-only section by means of '__attribute__((progmem))'
 char morseTable[] PROGMEM = "*5*H*4*S***V*3*I***F***U?
*_**2*E*&*L\"**R*+.****A***P@**W***J'1* *6-B*=*D*/"
                ^
exit status 1
Error compiling for board Arduino/Genuino Uno.

To fix this error, edit the MorseEnDecoder.cpp file and add const to line that starts with char morseTable[].
Here is the corrected line.
const char morseTable[] PROGMEM = "*5*H*4*S***V*3*I***F***U?
*_**2*E*&*L\"**R*+.****A***P@**W***J'1* *6-B*=*D*/"

The Morse Library is at http://www.w5obm.us/Arduino/ go to chapter 7 and then Morse. I copied all the files keyworks.txt, Morse.cpp, Morse.h and a directory of Examples to my system. The directory is C:\Program Files (x86)\Arduino\libraries\Morse.

The PS2 keyboard library can be found at http://www.w5obm.us/Arduino/ go to chapter 19. Then PS2Keyboard and then copy the PS2Keyboard.cpp, PS2Keyboard.h and keywords.txt. Copy them to your Arduino libraries,
C:\Program Files (x86)\Arduino\libraries\PS2Keyboard.

```
This PS2Keyboard library was modified to add F1-F12 keys and other minor
corrections by Glen Popiel - KW5GP
```

Once you have all the items on your computer you can use the ADE to program the Arduino via the computer's and Arduino USB ports.

Adafruit PS/2 keyboard connector information.

- Green wire is +5V
- Black wire is Ground
- Brown wire is Data, goes to pin D5 on Arduino.
- Yellow wire is Clock, goes to pin D3 on Arduino.
- White wire is sometimes Mouse Data for 2-in-1 splitter cables, not used.
- Red wire is sometimes Mouse Clock for 2-in-1 splitter cables, not used.

I tried a USB keyboard on this project by buying a USB to PS2 adapter at Office Depot. It will convert a USB keyboard to a PS2 connector. It did not work.

Code Practice Oscillator

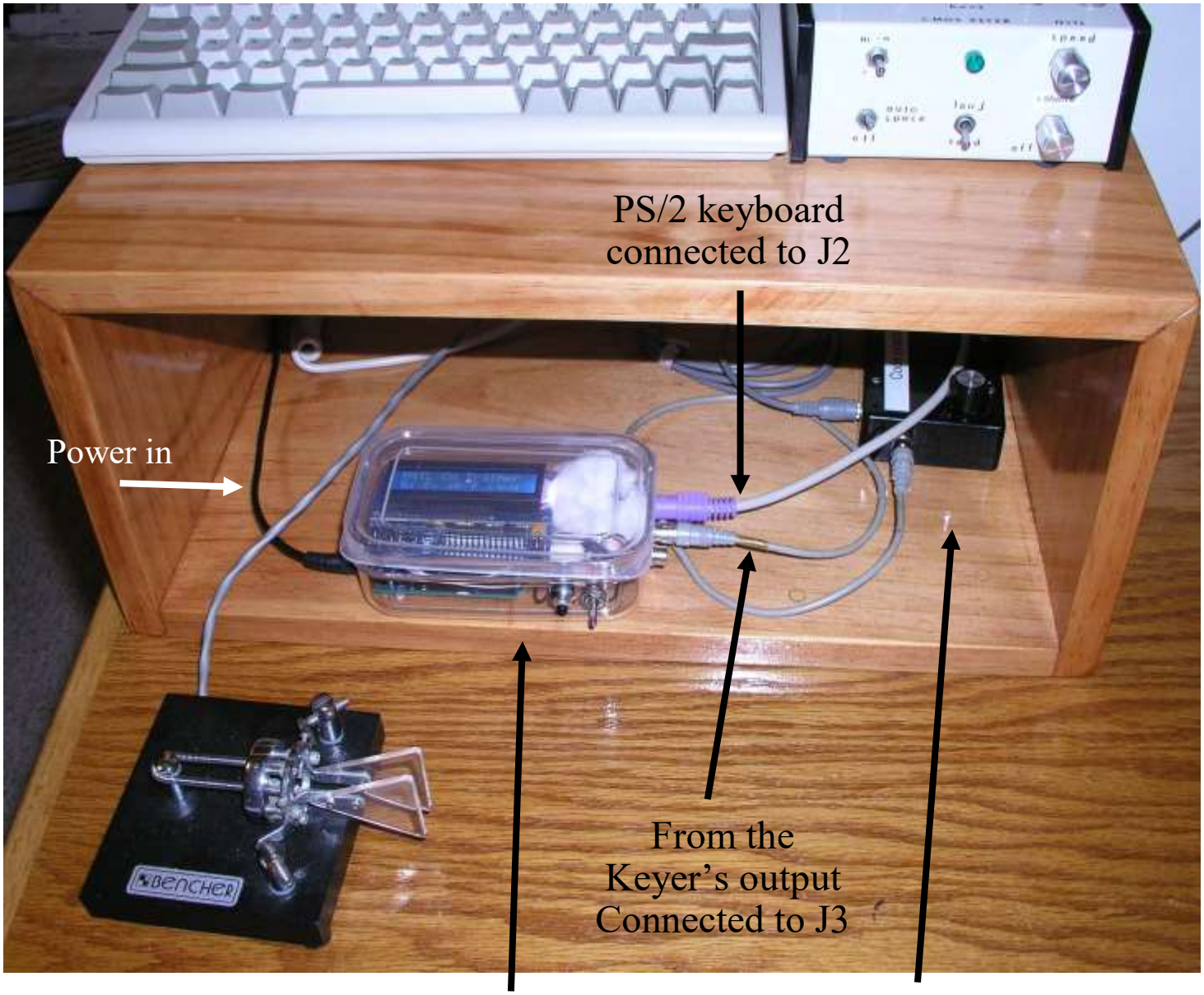
Some people who have built the code trainer have missed the fact that it will not generate any sound when the student is sending. If you are using a straight key, you have to connect a code practice oscillator (CPO) to the key. If you are using a Keyer, connect the CPO to the Keyer's output. Some Keyers have a built in speaker so you can hear your sending and a CPO is not needed.

This website has information on a simple one.

Code Practice Oscillator Using 555 Timer

CPO information added June 18, 2018.

PS/2 Keyboard Keyer



Bencher Key

Arduino CW Trainer

Code practice oscillator

This picture shows the whole setup. The keyboard at the top is connected to the trainer with the purple connector. A USB keyboard will not work. The Bencher key is connected to the Keyer on the top right of the shelf. The Keyer does not have a speaker for a side tone, so it is connected to a code practice oscillator just below it in the black box. The output of the Keyer is also connected to the Arduino CW Trainer. If my Keyer had a side tone oscillator built in the Code Practice Oscillator would not be needed.

The relay output J1 is not used. It is not needed. I had it in my original design because I was also experimenting with a Arduino Keyer.

I connected my straight key to the input of the Arduino CW Trainer and the Trainer was able to decode my sending. I did not even have to change the speed in the trainer for the slower sending

This is a list of keys on the keyboard that are used by the Arduino CW Trainer Sketch.

- The Up arrow increases the sending speed.
- The down arrow decreases sending speed.
- The Right arrow increases the number of characters sent before the Arduino checks for incoming characters sent by the student.
- The Left arrow decrease the number of characters sent before the Arduino checks for incoming characters sent by the student.
- F1, Sets the character set to the twenty six letters of the alphabet.
- F2, Sets the character set to the ten numbers.
- F3, Sets the character set to these four characters, period, comma, slash and question mark.
- F4, Sets the character set to all forty characters listed above.
- F5, Kotch method. Enter a one or two digit number followed by the enter key.
- F6, Kotch method with different starting point. Enter a one or two digit number followed by the enter key. Use F5 before using F6.
- F7, Set a delay from 0 to 30 to slow down sending. Version 2 of the sketch.
- F9, is used to toggle between the internal speaker or relay output. Do not use if your trainer does not have the relay output.
- F10, saves the parameters in to the Arduino's EEPROM. F10 has to be pushed before the G is entered to start the code generation. The next time the trainer is used, the parameters will be read from the EEPROM. The user only has to push G to continue.
- G, short for go. It starts the Arduino sending code via the speaker or relay.
- D, for decoder. It runs the CW decoder only. The speaker is not used.
- P, for checking the sending speed, PARIS is sent until reset is pushed. Version 2 of the sketch.

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

Using the Arduino CW Trainer

Learn to send.

The first thing to do is learn how to send so that the trainer can copy your code. Turn on the trainer and push D. That will run the decoder only. Now send to the trainer. Make sure your sending can be copied by the trainer. I found that I need to leave a little more space between each character.

Learn the sounds.

Now use the Kotch method to learn the sound of each character. Push F5 followed by a 1. Push the right and left arrows so that you have 2 characters selected. Then push F10 to save the values. Now push G for go. The trainer will send you K and K. You send K and K back. After you know the sound of K push reset. Now push F5 followed by a 2. Now push F10 and G. The trainer will send you K and M. Now send K and M back with out looking at the display so that you learn the sounds. Keep doing this with more and more characters until you know all of them. You can use F6 to set a different starting point in the Kotch method.

Work on remembering what was sent.

Now change the number of characters sent from 2 to higher numbers, and practice remembering what was sent, again with out looking at the display and not writing anything down.

If you stop one day and come back the next day the trainer will remember where you left off because you pushed F10. If you want to continue where you were the day before all you have to do is push G.

As I said in the article I am not an expert on learning the code. Jack W0UCE (SK) was. Do a google search on “W0UCE's method for teaching CW” and read what he wrote. Also look at www.kb6nu.com/another-method-for-teaching-and-learning-morse-code.

73 Tom, N4TL

September 16, 2016, Version 2.1

I updated the sketch after receiving feedback from some users.

Added F7, This is used slow down sending by putting a delay between characters. Zero to 30 can be entered. Each count is 0.01 seconds. I found a number of 10 to 20 will slow the sending down okay. This delay allows the characters to be sent a full speed with delay between each character. This is called the Farnsworth method. So now you can use the Koch and Farnsworth methods at the same time.

Added P to the keyboard control. When P is pushed the trainer will send PARIS over and over again until reset is pushed The length of time it takes to send PARIS divided in to 60 gives the speed in WPM. I changed the Key_speed_adj = -2 and measured these speeds. I don't know if different processor speeds will change these speeds.

Trainer speed	Measured speed F7 delay = 0	Measured speed F7 delay = 10	Measured speed F7 delay = 20
20	19.9	17.7	15.7
25	25.3	21.6	18.9
30	30.8	25.6	21.9

Modified D, when characters are written to a new line, it is blanked first instead of being overwritten. The characters are also written to the serial monitor.

One bug was fixed. Up to 10 characters are stored and compared to the student's sending. The right arrow routine allowed up to 15. This was changed to 10. I did some code clean-up and added a few constants. When the trainer first starts, I added a period after N4TL. That allows me to know I am running the second version.

I may write another sketch that will send short words. If I do I will put information about it on QRZ under my call's information, and at QSL net, www.qsl.net/n4tl.

If you build a Arduino CW Trainer please let me know by sending me your name and call via email, n4tl2 at yahoo.com, thanks.

73 Tom N4TL

November 4, 2016

A New Version that does not need a keyboard.

Michael Hughes, KC1DMR, sent me an email and told me about his updated sketch. He rewrote the sketch so that it uses the buttons on the display and the keyboard is not needed. I tested it in my hardware and it works fine. Here is a copy of Mike's email to me. His code is at https://github.com/ mfhughes128/cw-trainer.

Subject: Update to Code trainer article

The LCD shield has five buttons which are referred to as 'up', 'down', 'left', 'right' and 'select'. On power up the first menu displayed is -

N4TL CW Trainer
>Start Trainer
>Start Decoder
>Set Preferences
>Send PARIS test

Of course only one line of the menu can be shown at a time. The 'up' and 'down' buttons scroll through the options. The 'left' and 'right' buttons do nothing. The 'select' button activates the option showing.

The options are -
- The 'Start Trainer' option begins sending characters and listening for the trainee's response as in the original sketch.
- The 'Start Decoder' option begins listening for Morse characters and displays them on the lower line of the display. The upper line is reserved for now.
- The 'Set Preferences' option opens a second menu to set the trainer parameters.

On the second menu the name of a parameter will be displayed on the top line and the parameter's current value will be displayed on the bottom line. The 'up' and 'down' buttons scroll through the parameters and the 'left and 'right' buttons decrement or increment the value. The 'select' button saves the parameters to EEprom and returns to the first menu. The parameters are -

"Code Group Size:" - The number of characters to send [1 to 15]
"Character delay:" - " - The delay between characters in a group
in 0.01 second increments [0 - 30]
"Code Speed:" - Speed in wpm [20 - 30]
"Character Set:" -
[1=alpha,
2=numbers,
3=special chars.,
4=all alphabetic,
5=Koch order,
6=same as 5 for now]
"Koch Number:" - The depth of the Koch character table
"Skip Characters" - Characters to skip at the beginning of the Koch table.
"Out:0=key,1=spk"- Selects the output pin and analog or digital output mode.

November 4, 2016

A New Version that does not need a keyboard.

The modified sketch is attached and is also available in a Git repository at

https://github.com/ mfhughes128/cw-trainer

This should work with the hardware as described in the article. Please send me any bugs you find, or you can always fork from git hub and send a pull request.

I hope you all find this work useful, or at least interesting.

Thanks

Mike Hughes
KC1DMR