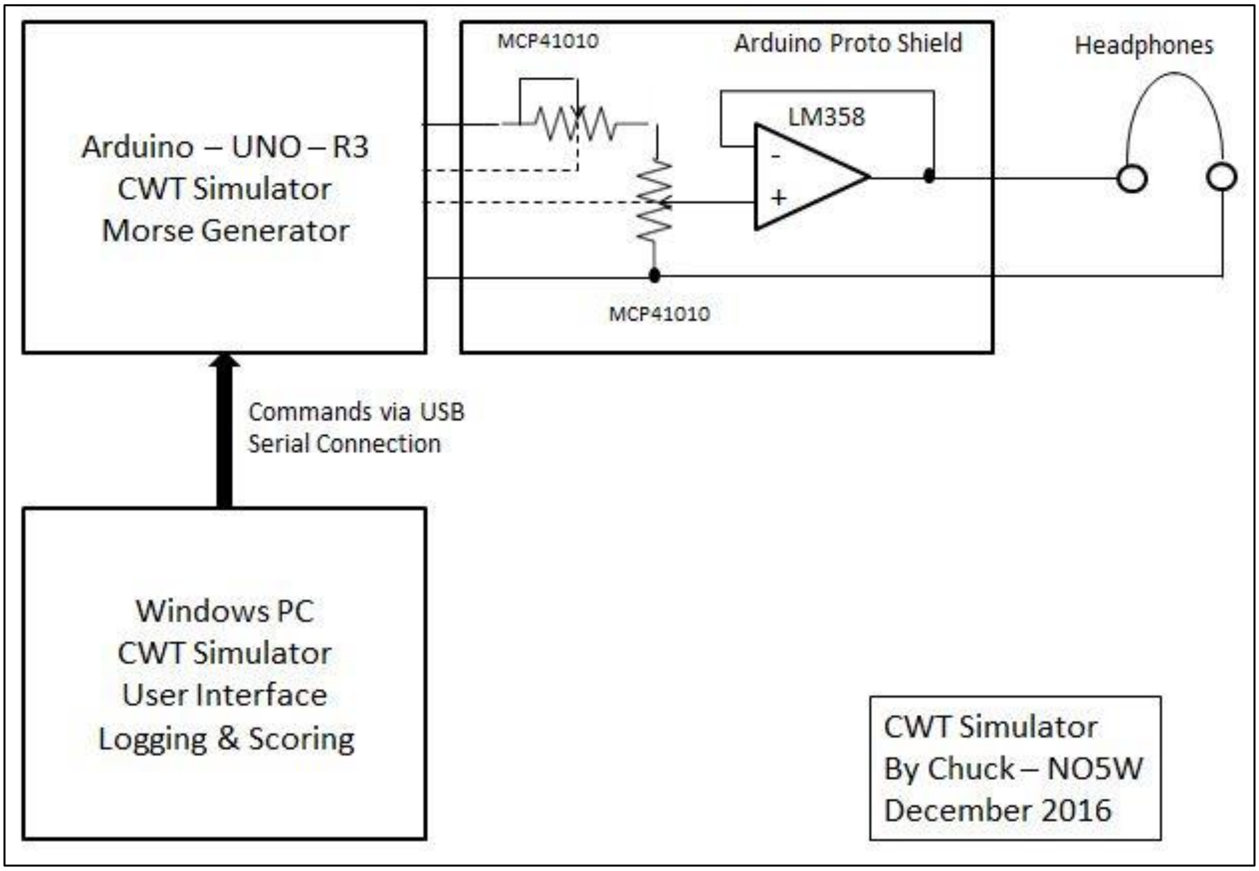
Construction Notes for the NO5W CWT Simulator

Overview of the Simulator – As shown in the following diagram the Simulator consists of three main parts:

- an application running on the PC (or Mac) which provides the logging interface for the user;
- an application running on the Arduino that generates Morse in response to commands from the PC application;
- a signal conditioning subsystem consisting of two digital potentiometers and a buffer amplifier that provides the final audio to the user headphones. The signal conditioning is necessary in order to generate Morse of varying signal strength under control of the Arduino since audio tones from the Arduino itself have fixed amplitude.

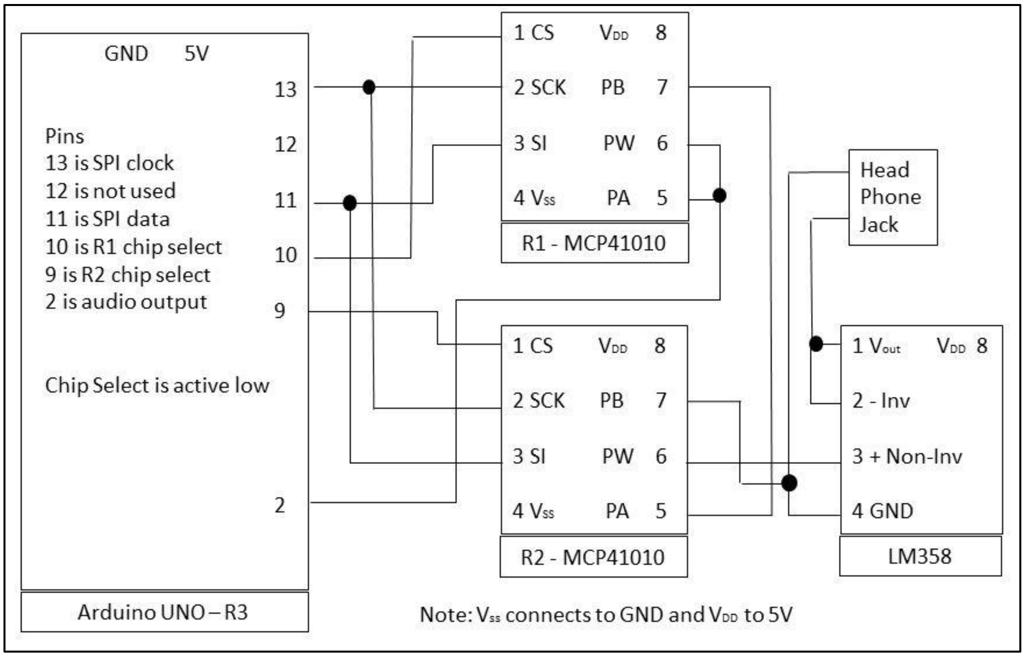
The Arduino is essentially a microcontroller (ATmega 328) with a few peripherals including a USB interface, a Serial Peripheral Interface (SPI) and a relatively large amount of digital I/O and analog inputs. An integrated development environment (IDE) is freely available for download and is used on a PC to develop code using C/C++ for compiling and uploading the resulting hex code to the device over a USB cable.



Constructing the Simulator – Building your own version of the simulator requires only a small amount of construction, primarily in the wiring of the digital pots, and installation of some software. Here is an overview of what you will need and the steps involved. Following this overview there is a section that provides more detailed steps. You will need the following software and hardware.

- **PC Application Code** – The executable for the PC that provides the user interface as well as the command interface to the Arduino is available for download from the CWT-Simulator page of the NO5W website. This code was developed for the Windows operating systems (XP, 7, 8, 10) but tests have confirmed that it will run fine on a Mac under the Parallels desktop software. The executable and other files needed by the application will be installed when you run the setup exe contained in the download package.
- **Arduino Code** – The download package obtained from the author's website also contains source code to build the application that runs the Arduino Morse generator. Following installation of the PC application the Arduino code, which consists of a single file (CWT-Simulator.ino), may be found in the Arduino subdirectory of the PC application. To use this code you will need to setup an Arduino development capability. This is easy to do since the Arduino IDE is available for free from this site. Using this IDE and the source code, compiling and uploading the resulting hex code to the Arduino is straightforward and is handled completely by the Arduino IDE over the USB connection.
- **Arduino Hardware** – There are several versions of the pre-assembled Arduino. The one used in constructing the Simulator is Rev 3 of the Arduino UNO (UNO-R3). Since this is open-source hardware it is available from a number of manufacturers with prices ranging between $5.00 and $25.00. The version used in the development of the prototype was purchased at a ham fest from MCM Electronics and cost about $22.00. I have since also found it available from Adafruit, Sparkfun Electronics, and several other vendors. During development a version of the Simulator was also built using the Arduino Mega 2560 which is a more capable device with more memory and more I/O but of course it is also more expensive (~$50.00). For the CWT Simulator in its current form use of the Mega 2560 would be overkill. It may also be possible to use other Arduino devices lower in the food chain than the UNO-R3 including for example the Pro-Mini but in order to do that you would need to modify the pin definitions in the Arduino source code as well as changing the pins used in connecting to the digital pots on the proto-shield. Also the lower level devices tend to be more difficult to work with. So my recommendation would be to stick with the UNO-R3 and preferably one from one of the vendors mentioned above.
- **Miscellaneous Parts** – Enclosures are available with cutouts and mounting tabs pre-made for the USB and power connectors on the Arduino. One such enclosure is available from Sparkfun as part PRT-10088. You will also need a USB –AB cable for connecting between the PC and the Arduino. A headphone jack will also be needed. I added a 10K manual pot between the output of the op-amp and the headphone jack for convenience as a final master volume control. The USB-AB cable and the headphone jack are available from many vendors and may already be in your junk box. Depending on how you decide to install the three 8-pin ICs on the proto-shield you might need sockets.
- **Proto-shield Construction** – Daughter boards for the Arduino are called shields and provide a way of adding functionality to that of the Arduino and a reliable way to make connections to the Arduino board and easy removal if you find you have other applications for the Arduino. The additional functionality needed in the Simulator consists of two MCP 41010 SPI- controlled 256-level 10K digital potentiometers and a single LM358 operational amplifier. These parts as well as

the proto-shield (Sparkfun DEV-07914) were obtained from Mouser Electronics. The digital pots and the op-amp were in the $1.50 price range in single quantities and the proto-shield was about $10.00. The digital pots are also available in a dual 14-pin package as MCP 42010. Some soldering is required in assembling the proto-shield and in connecting the parts. Since I did not have a ready source of sockets I chose to use a breadboard during development and then to solder the ICs directly to a proto-board (Radio Shack 276-150) in the final assembly. An option to the Radio Shack board would be a Quarter-size breadboard PCB (Product 589) available from Adafruit. The source code for the Arduino assumes certain pins are used for the audio output and SPI interface lines to the digital pots on the proto-shield. Here is the wiring diagram for the output section.



Note: V_{ss} connects to GND and V_{DD} to 5V

The audio tone output from the Arduino is obtained from pin 2. During key-down it has a 5-volt amplitude and the eventual volume/signal strength produced for the user is controlled by digital pots R1 and R2 using the SPI interface provided by the Arduino. The SPI interface uses four pins on the Arduino: the clock from pin 13 connects to both of the digital pot clock inputs (SCK) to synchronously load data from pin 11 which connects to both of the digital pot data inputs (SI). The digital pot receiving a given data setting is controlled by the active-low chip select inputs from Arduino pins 9 and 10. R1 is the digital pot wired in rheostat mode and is used to set the maximum volume. R1 is fixed during a session whereas R2 is wired as a voltage divider and is used to dynamically vary the “signal strength” of the callers. R2 provides output to the user via the op-amp which is simply a buffer amplifier to isolate R1-R2 from loading by any headphone(s). I have also found it useful to add a manual pot between the output of the LM358 and the headphones jack to serve as a master volume control. A 10K pot should work fine for that purpose.

Detailed Construction Steps – Here is a somewhat detailed description of the steps involved and the recommended sequence. It assumes you have all of the necessary components described above.

Install the Application Files –

1. Browse to the CWT-Simulator page of the NO5W website and download the installation zip file to a directory on your PC.
2. Extract the files from the package and run the setup file (Setup-CWT-Simulator-110.exe) to install the PC application

Setup and Test the Arduino Integrated Development Environment (IDE) -

3. Browse to the Arduino site and follow the instructions to download and install the Arduino IDE on your system. A good on-line resource of detailed instructions for this is at this site.
4. Use a USB cable to connect your Arduino UNO to your PC. You should see an LED light up on the UNO and remain steady indicating that the Arduino is receiving power via the USB connection.
5. Compile and upload the simple blink application that comes with the Arduino IDE to check that your UNO is working. When you do the compile be sure to select the correct Arduino. If all goes well you should see a slowly blinking LED.

Compile and Upload the CWT Simulator Code to the Arduino

6. The Arduino IDE will keep track of where your source code (these are called sketches in Arduino terminology) is located if you follow its scheme of placing your sketch in a separate directory having the same name as the sketch and where the directory is a sub-directory of the Arduino documents directory which is normally located in the MyDocuments folder. Browse to your MyDocuments folder and you should find a sub-directory called Arduino. Make a subdirectory under the Arduino directory called CWT-Simulator.
7. Move or copy the Arduino source code (CWT-Simulator.ino) found in the Arduino sub-directory of the CWT-Simulator that you installed in step 2 to the …MyDocuments\Arduino\CWT-Simulator\ directory.
8. Start the Arduino IDE and notice that if you use menu item File | Sketchbook one of the sub-menus of Sketchbook will be called CWT-Simulator. Click on that menu item and the IDE will load the CWT-Simulator source code. At the top of the source code file you will notice two #include statements which tell the compiler to go out, find the two named files and include them in the code:
    - NewTone which is an improved version of the tone library that is responsible for creating the tones used in the Morse generator
    - The SPI library is the one that allows the Arduino to talk to external devices (in our case the digital pots) using the SPI communication protocol.

    The two files are called libraries and contain code that was written by others saving the user a lot of work. Of course it is very important that the compiler be able to locate and use the two libraries. Some libraries are included as part of the Arduino IDE installation performed in Step 3 above. You can find information on libraries including how to use them here . If you go there you will note that the SPI library is part of the standard but the NewTone library is not. So we need to make sure that the NewTone library is accessible to the IDE. There is capability within the IDE to manage libraries and this will enable the IDE to install contributed libraries that you find on the internet. The steps required to install a contributed library may be found here.

9. Use the menu item Tools | Board and verify that the selected board agrees with the board you are using (Arduino/Genuino UNO). If it does not agree use that same menu to select the correct board.
10. Use menu item Sketch | Verify/Compile to verify that the sketch will compile for the selected board.
11. Use menu item Sketch | Upload to upload the compiled code to the Arduino.

Check the Operation of the PC and Arduino

12. Now that you have the PC application installed and the Arduino Morse generator installed you can check out how they operate together. At this point you will not have control over the signal strength of the simulated signals but it is a good idea to test prior to adding that capability. It is also a good idea to have some means of controlling the volume to the headphones. This can be done by connecting a 10K manual pot between the output pin (2) of the Arduino and GND on the Arduino and connecting the headphone jack between the wiper terminal of the pot and GND on the Arduino.
13. Now perform the following steps on the PC application CWT-Simulator
    1. Make sure that the Arduino IDE is off
    2. Double-click on the CWT icon on the desktop
    3. Say NO to each of the startup message boxes asking do you want to continue with…
    4. If necessary use Device Manager to determine the com port that the Arduino is connected to and use the Port menu to connect to the Arduino, leaving all items but the port number unchanged.
    5. Use menu item File | Build New Set of Callers filling in all of the information requested on the resulting dialog. For the source in the root directory of the CWT-Simulator you should find a file having a name like _CWT_CH_VE2FK_1744.txt. Use that as the source and for the destination choose a name like CallingStations.txt in the root directory of the application.
    6. Use menu item Setup | Operator Info to replace the info currently there with your info
    7. Use menu item Setup | Configure Session to define a session, probably best to use a fixed duration session of long enough time (say 5 minutes) while you adjust the manual pot for a comfortable volume level.
    8. Click on the Start button. Because this may be the first startup you may be asked to do 4-7 again but all of the info you have already entered should still be there. And the next restart of the program you can answer either YES or NO to the initial “Do you want to continue” messages.
    9. Play the simulator a few times to get the feel and to determine whether you want to proceed with the building of the proto-shield part.

10. Remove the manual pot in preparation for use in the final assembly where it will be connected to the output of the op-amp.

Install the Digital Pots and the Op Amp on a PCB

14. Finally it's time to do some hardware construction by mounting the three ICs to a PCB and making the necessary interconnections shown in the wiring diagram. Several PCB breadboards are available including Radio Shack 276-150 and the Quarter-size breadboard PCB (Product 589) available from Adafruit. Use of sockets for the ICs is optional. Careful soldering is required to avoid solder bridges on the closely spaced pads. Be sure to connect the manual pot and the headphone jack to the PCB.
15. Once you have all of the solder connections made you can either connect the resulting PCB directly to the Arduino headers or you can connect it to a proto-shield such as the one from Sparkfun (Part 10088). The main advantage of using the proto-shield is that you can leave the PCB in place on the proto-shield while removing the shield for some other use of the Arduino.
16. After making connection to the Arduino it's time to test by doing the following:
    1. Make sure that the Arduino IDE is off
    2. Initially leave the manual pot in its current setting
    3. Double-click on the CWT icon on the desktop to start the PC application
    4. Say YES or NO to each of the startup message boxes
    5. Connect to the Arduino
    6. Use the Setup | Test Arduino menu to set up the desired max volume by clicking on the Start button on the resulting dialog and listen to the volume increases until it is at the desired level and click the Stop button. Click the Start button associated with the Min volume and when that volume reaches a desired level click on its Stop button. You may need to experiment some by repeating this process for different settings of the manual pot. Once you arrive at desired settings for the Max/Min volumes you should note them and in the next step use them in setting the volume range in the creation of the pool of callers.
    7. Close the Test Arduino dialog and use menu File | Build New Set of Callers to call up the dialog. Your previous entries should still be available and you may only need to change the volume range.
17. That should complete the construction phase of the project.

If you have questions, encounter problems, or have suggestions for additional features and/or areas of improvement feel free to contact the author using the CQ/X support email found on the NO5W website.

Enjoy,

73/Chuck/NO5W

References – The following were found useful in the development of the CWT Simulator and may be of interest to those wishing to dig deeper into certain areas.

1. MicroChip, Datasheet DS11195C, MCP41XXX/42XXX Single/Dual Digital Potentiometer with SPI Interface, 2003.
2. National Semi-conductor, Datasheet DS007787, Low Power Dual Operational Amplifiers, LM358, October 2005.