Cyber City Circuits Dreamweaver 4N Instructions

Draft 20200620

Contents

1. Overview

The Dreamweaver 4N is an Arduino compatible music arpeggiator/sequencer a *luxurious* **ATMEGA328PB micro-controller**, a *beautiful* **CH-340G** USB interface, 4 *luminous* **WS2812E** RGB LEDs, a *standard* **on-board speaker**, and 4 *morphenomenal* **3.5mm jack adapter** so that you can connect it to your **existing sound systems/mixers**. The board is fully compatible with the Arduino IDE if you select 'Arduino Nano' in the 'Board Manager'.

The WS2812E LEDs are the same ones that Adafruit uses for their NeoPixel lights and it uses the Adafruit Neopixel library.

The four 3.5mm IO jacks are connected to pins on the ATMEGA328PB. IO1 is connected with the speaker and is the default audio output. IO2 is a digital IO connection, while IO3 and IO4 are capable of reading analog inputs.

2. Learning More About Your Dreamweaver 4N

2.1. Arduino Pin Connections

Arduino Pin	Name	Description
0		Used for Serial Data
1		Used for Serial Data
2	sw_9	Button 9
3	io2	IO 2
4	io1	IO 1/On Board Speaker
5	sw_4	Button 4
6	sw_5	Button 5
7	sw_6	Button 6
8	sw_7	Button 7
9		RGB Data Pin
10	sw_3	Button 3
11	sw_2	Button 2
12	sw_1	Button 1
13	sw_8	Button 8
14	io3	IO 3
15	io4	IO 4
16	knob_1	Value 1 Knob (Knob 1)
17	knob_2	Value 2 Knob (Knob 2)
18	knob_3	Value 3 Knob (Knob 3)
19	knob_4	Value 4 Knob (Knob 4)
20	knob_6	Spread Knob (Knob 6)
21	knob_5	Length Knob (Knob 5)

2.2. The Tone Function

The Arduino IDE has a function built-in called ‘tone’. The ‘tone’ function generates a square wave of the specified frequency (and 50% duty cycle) on specific pin. The pin can be connected to a piezo buzzer or other speaker to play

tones and only one tone can be generated at a time. We use this tone function to play music on the Mini-Tone Keyboard. The 'tone' function takes three arguments. Arguments are used to pass different settings and information to the function. The arguments for the 'tone' function are (1) speaker pin, (2) note to play, (3) duration of the note.

2.3. The RGB Lights

The Dreamweaver 4N has eight (8) WS-2812E LEDs on it. These LEDs are very popular with people because they are very easy to use. While programming your Mini-Tone Keyboard, we will be using a library from a company named Adafruit. They write all kinds of Arduino software and we will be using the Adafruit Neopixel library to interact with these lights.

2.4. Input/Output Jacks

Your Dreamweaver 4N has four (4) IO jacks. These are made to connect to 3.5mm audio cables, like the ones that are used with Eurorack systems.

IO	Arduino Pin	Pin Type	Description
IO1	4	Digital Pin	Default Audio Out, Connects to Speaker.
IO2	3	Digital Pin	Standard Digital I/O
IO3	14	Digital/Analog Pin	Capable of Reading Analog Input
IO4	15	Digital/Analog Pin	Capable of Reading Analog Input

2.5. Analog Knobs

Name	Arduino Pin	Default Use Description
Value 1	16	Sets the value for the first note.
Value 2	17	Sets the value for the second note.
Value 3	18	Sets the value for the third note.
Value 4	19	Sets the value for the fourth note.
Length	21	Sets the length of time that a note plays.
Spread	20	Sets the space between notes.

2.6. Buttons

Name	Arduino Pin	Default Use Description
SW1	12	Toggles Note 1
SW2	11	Toggles Note 1
SW3	10	Toggles Note 2
SW4	5	Toggles Note 2
SW5	6	Toggles Note 3
SW6	7	Toggles Note 3
SW7	8	Toggles Note 4
SW8	13	Toggles Note 4

SW9	2	Changes Between Modes
RESET	RESET	Resets the controller in case it locks up or has an error.

3. Using the Arduino IDE

3.1. Setting up the Arduino IDE

You may need to download the Arduino IDE from the Arduino website. (https://www.arduino.cc/)

The version of the firmware on the Micro-Controller is called the 'Arduino Nano'. It comes with the Arduino Board Manager. To set your Arduino IDE to work with this board, change these settings.

- Tools -> Board -> Arduino Nano
- Tools -> Port -> What Com Port your computer is using for the connection.
    - You may need to check your port number using Windows' Device Manager.

3.2. Libraries

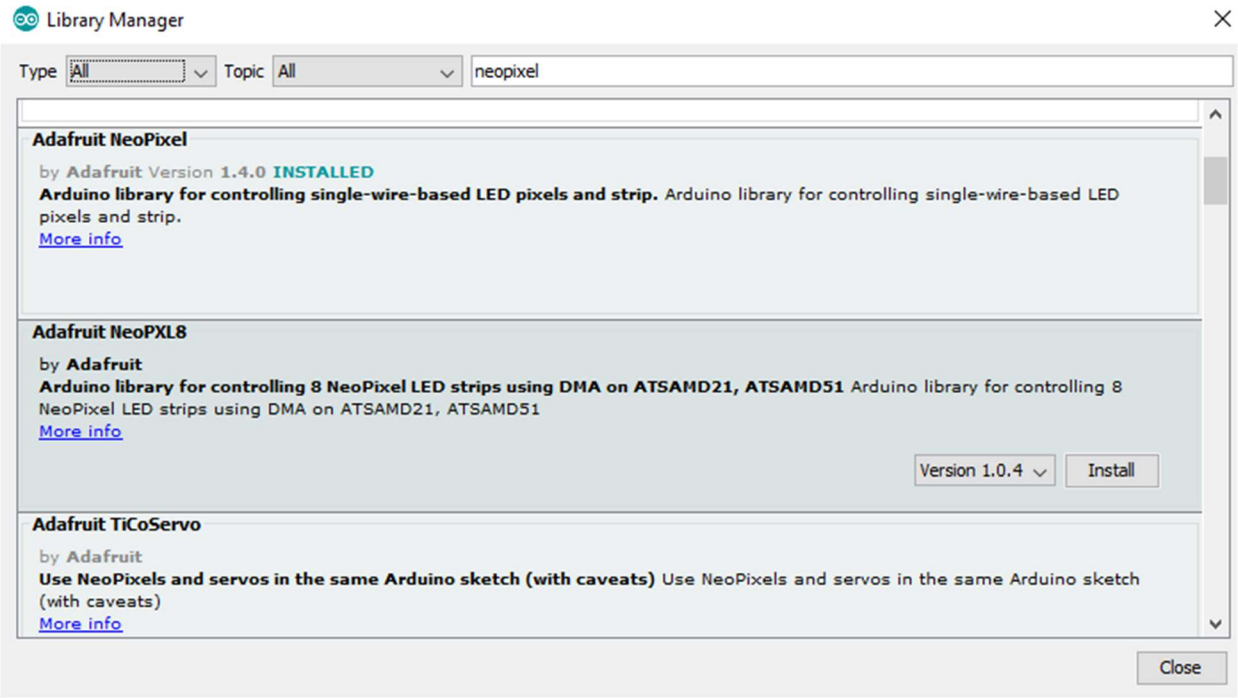
A library is basically a small file or group of files that make it easier to share functionality among different programs. For example, we will need to use a very complex process to use the button matrix. Instead of writing a new process every time we want to use a button matrix we could just use a library. That way we only need to get it working once and then we can easily take that functionality to other projects.

3.2.1. Using the Library Manager

To open the Arduino Library Manager, within the Arduino IDE click the 'Tools' button in the top menu. From there click 'Manage Libraries…'. This is where we will go to install new libraries to your Arduino IDE. We need two libraries.

3.2.2. *Installing the Adafruit Neopixel Library*

The library we will need is the ‘Adafruit Neopixel’ library. In the search bar of the Library Manager, type in ‘neopixel’. Scroll down to find the ‘Adafruit Neopixel’ library by Adafruit and click install.

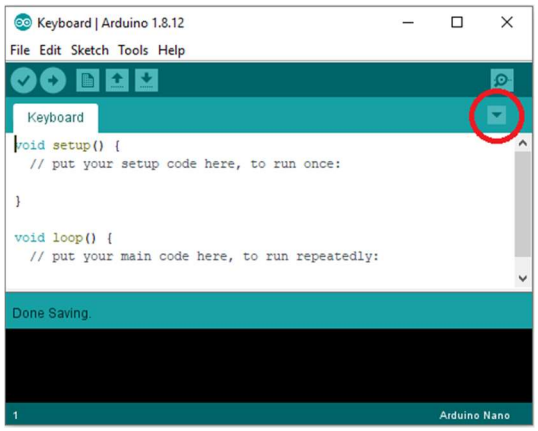


3.3. Header Files

When programming with the Dreamweaver 4N, we will be using a header file named ‘pitches.h’. This file will be used to store the note values for the Dreamweaver 4N. Essentially, all it will be is a lookup table for the software to reference when trying to play notes. Writing things like this in a header file will help you keep your code better organized and will make it easier to use the same variables in other projects. The header file ‘pitches.h’ will need to be in the same folder as your main software program.

4. Let’s Start Writing Code!

Open the Arduino IDE and click the ‘File’ menu in the top bar and then click ‘New’. Next, go ahead and click the ‘File’ menu again and click ‘Save’. A new window will open asking you what you would like to name your program. Name your program ‘Dreamweaver4N’. Now you have your program saved and we need to add our header file. On the right of your Arduino IDE screen, you will see a little button with a down button arrow. Click on that button and then click on ‘New Tab’. Then on the bottom of the



Arduino IDE it will ask you for a name for your new file. We need to name that file 'pitches.h', all lowercase.

4.1. Setting Up Your Header File

As you read before, the 'pitches.h' header file is a way to store the note values. Near the top of your Arduino IDE you will see two tabs. One is named 'Dreamweaver4N' and one is named 'pitches.h'. If you need to write your own, there is a reference copy in the appendix.

Great! Now you can reference these variables easily from within your main program.

4.2. Playing Mary Had A Little Lamb

Now we are going to create a simple and basic program to play a song on your Mini-Tone Keyboard. Mary Had A Little Lamb is a super simple song and is a great way to learn to use the tone function. Let's set it up step-by-step. You may remember that earlier we mentioned two essential functions. These essential functions must be in every Arduino program for it to operate correctly. They are named 'setup()' and 'loop()'.

In the space before the 'setup()' function we will need to add a few lines of code.

The first line will tell the program to use the 'pitches.h' header file that we just made.

```
#include "pitches.h"
```

After that we need to tell the Arduino IDE what pin our speaker is on. To do this we will create a variable named 'speaker' and make that equal to the pin that the speaker is connected to on the micro-controller.

```
int io1 =  4;
```

The notes we use for Mary Had A Little Lamb are:
'B5, A5, G5, A5, B5, B5, B5, A5, A5, A5, B5, B5, B5'

To play a note we will use the 'tone' function followed by the 'delay' function. We have to use the delay function so that the notes that we play with the 'tone' function don't blend together. Go ahead and type in the following 'tone' and 'delay' functions into your 'loop()' function.

```
void loop(){
   tone(speaker, B5, 250);
   delay(500);
   tone(speaker, A5, 250);
   delay(500);
   tone(speaker, G5, 250);
   delay(500);
   tone(speaker, A5, 250);
   delay(500);
   tone(speaker, B5, 250);
   delay(500);
   tone(speaker, B5, 250);
```

```
  delay(500);
  tone(speaker, B5, 250);
  delay(500);
  tone(speaker, A5, 250);
  delay(500);
  tone(speaker, A5, 250);
  delay(500);
  tone(speaker, A5, 250);
  delay(500);
  tone(speaker, B5, 250);
  delay(500);
  tone(speaker, B5, 250);
  delay(500);
  tone(speaker, B5, 250);
  delay(500);
}
```

and then upload your sketch. If everything worked, you should hear that magical square-tone lullaby of the Mary Had A Little Lamb on your Dreamweaver 4N.

Note: You will probably want to disconnect the board from the USB so the music will stop. You can plug it back in later when you are ready to push your new code.

4.3. Introducing Variables into the Code

That was very tedious wasn't it? What if you wanted to make the delays last a little longer or make them shorter? You would have to change every single instance of delay in your code and that can be exhausting if you do it enough. This is where variables can come in real handy. In the area before the 'setup()' function, add a variable named 'wait' and make it equal to 300.

```
int wait = 300;
```

Now we want to change every 'delay' function to use this new 'wait' variable. To do this, we are going to use the find and replace tool in the Arduino IDE. Click on the 'Edit' menu in the menu bar and then click on 'Find…'. A new window will pop up with two fields in it. Put 'delay(500);' into the 'Find:' field and then put 'delay(wait);' into the 'Replace with:' field. Then click the 'Replace All' button. Then, like magic, all the 'delay' functions have changed.

Go ahead and upload the sketch to your Dreamweaver 4N. You should hear the same notes being played as before, but the space between the notes is a lot shorter and the music should sound faster. Good job!

4.4. Creating a Function to Make Your Code Cleaner

Okay, the program is starting to get a little easier to work with, but it is still very tedious to use. This is where creating your own functions can make it a lot easier to work with your code.

In this exercise we are going to create a function called ‘playNote’. We want to be able to play a note and turn the light ‘D1’ on when a note is playing. Before we can use the light ‘D1’ we have to ass a couple of lines to our code. First, we have to assign the light to a pin number. In the area before the ‘setup()’ function, add the following line:

```
int led = 2;
```

Then, we need to tell our Arduino IDE that we need to make this light an OUTPUT. Inside the ‘setup()’ function add the following line:

```
pinMode(led, OUTPUT);
```

Now, we can starting using the light ‘D1’ with our code!

In the last session we made the wait variable to change how long the controller waited before playing the next note. Let’s add another variable for how long the note plays. In the area where you define the wait variable, make a play variable and in the code where it says 250 for the play length, we will use the play variable in the future.

```
int play = 250;
```

Now to add the function to our code. So far all of the code has been within the ‘setup()’ and ‘loop()’ functions. To create a new function, you need to add them AFTER the ‘loop()’ function. We want our ‘playNote’ function to do a few things. We want it to (1) play a note, we want to be able to (2) tell the code how long the note will be, and then (3) tell the code how long to rest before playing the next note. So, we will need to add those three arguments into the function. Then once the function is called, we want (4) the light ‘D1’ to turn on, (5) play the note, when the note is done playing we want (6) the light ‘D1’ to turn off, then we want it the function to (7) wait to play the next note. Add the following lines of code in the area AFTER the ‘loop()’ function.

```
void playNote(int note, int duration, int rest) {
  digitalWrite(led, HIGH);
  tone(speaker, note, duration);
  digitalWrite(led, LOW);
  delay(rest);
}
```

Now, instead of playing the note and having it wait in two lines over and over, we can just call this function. Below is an example of how we replaced four lines that we would need to write over and over again, with just a single line by using the function we just wrote.

Before	After

digitalWrite(led, HIGH);
tone(speaker, A5, play);
digitalWrite(led, LOW);
delay(wait);
	playNote(A5, play, wait);

The next step is going to be to change out all the lines in the ‘loop()’ function to use the new ‘playNote()’ function. Here is a reminder that every statement needs to end with a semicolon (‘;’) and the ‘loop()’ function has to have a closing curly bracket at the end (‘}’).

If this was done correctly you should not notice any difference at all from how it sounds.

5. Appendix – Code References

5.1. ‘pitches.h’

```
#define B0  31
#define C1  33
#define CS1 35
#define D1  37
#define DS1 39
#define E1  41
#define F1  44
#define FS1 46
#define G1  49
#define GS1 52
#define A1  55
#define AS1 58
#define B1  62
#define C2  65
#define CS2 69
#define D2  73
#define DS2 78
#define E2  82
#define F2  87
#define FS2 93
#define G2  98
#define GS2 104
#define A2  110
#define AS2 117
#define B2  123
#define C3  131
#define CS3 139
#define D3  147
#define DS3 156
#define E3  165
#define F3  175
#define FS3 185
#define G3  196
#define GS3 208
#define A3  220
#define AS3 233
#define B3  247
#define C4  262
#define CS4 277
#define D4  294
#define DS4 311
#define E4  330
#define F4  349
```

```
#define FS4 370
#define G4  392
#define GS4 415
#define A4  440
#define AS4 466
#define B4  494
#define C5  523
#define CS5 554
#define D5  587
#define DS5 622
#define E5  659
#define F5  698
#define FS5 740
#define G5  784
#define GS5 831
#define A5  880
#define AS5 932
#define B5  988
#define C6  1047
#define CS6 1109
#define D6  1175
#define DS6 1245
#define E6  1319
#define F6  1397
#define FS6 1480
#define G6  1568
#define GS6 1661
#define A6  1760
#define AS6 1865
#define B6  1976
#define C7  2093
#define CS7 2217
#define D7  2349
#define DS7 2489
#define E7  2637
#define F7  2794
#define FS7 2960
#define G7  3136
#define GS7 3322
#define A7  3520
#define AS7 3729
#define B7  3951
#define C8  4186
#define CS8 4435
#define D8  4699
```

```
#define DS8 4978
```