

Chapter 2 – Getting Started with Arduinos for Model Railroading

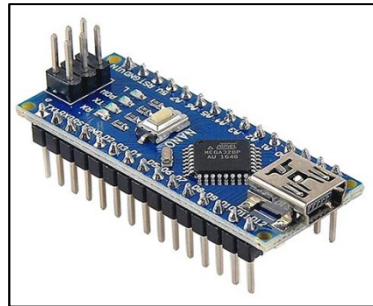
June, 2020

David Ackmann – Gateway Division NMRA

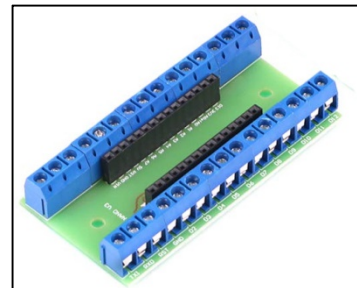
In this chapter, we are going to keep things pretty simple. We are going to learn how to:

- 1) Download the Interactive Development Environment (IDE)
- 2) Download a program from my “Amazing Arduino Animations” clinic
- 3) Watch as the Arduino blinks a LED, as if it was a train horn at a grade crossing

But first you need to buy an Arduino “Nano”, and I get my Arduino parts from AMAZON. I suggest you buy the one with the “Amazon Standard Identification Number, or ASIN, of B07G99NNXL; just put this number in AMAZON’s search bar, click on the orange and black magnifying glass, and the unit I like will pop right up. You will get three microprocessors for \$14, along with three USB cables to connect the Nanos to your Personal Computer. Yes, it would be nice if they sold them one-at-a-time for less, but they don’t. Yes, it would be nice if the package contained only a single USB cable, but it doesn’t. Yes, you could buy just one board for less money from a different vendor, but then you would have to do some soldering, and the purpose of this Chapter is to keep things simple, so just buy the 3-pack; you can thank me later.



I said you would only need to buy an Arduino Nano, and that is true, for this Chapter. But soon, and very soon, you will want to build more sophisticated projects, and this will require you to attach other components to the Arduino. If you also buy an “expansion board” (AMAZON ASIN: B07MGVC18K), you will find this future wiring much easier. Many other Arduino developers will tell you to use an Arduino Uno, jumper wires and breadboards to assemble projects, but I found that method to be unreliable, because the joints kept failing; they were not as tight as screw terminals. Every project I have created uses a combination of an Arduino Nano and an expansion board, so perhaps to save on shipping, you might want to buy a 5-pack for \$12 from the “git-go”.

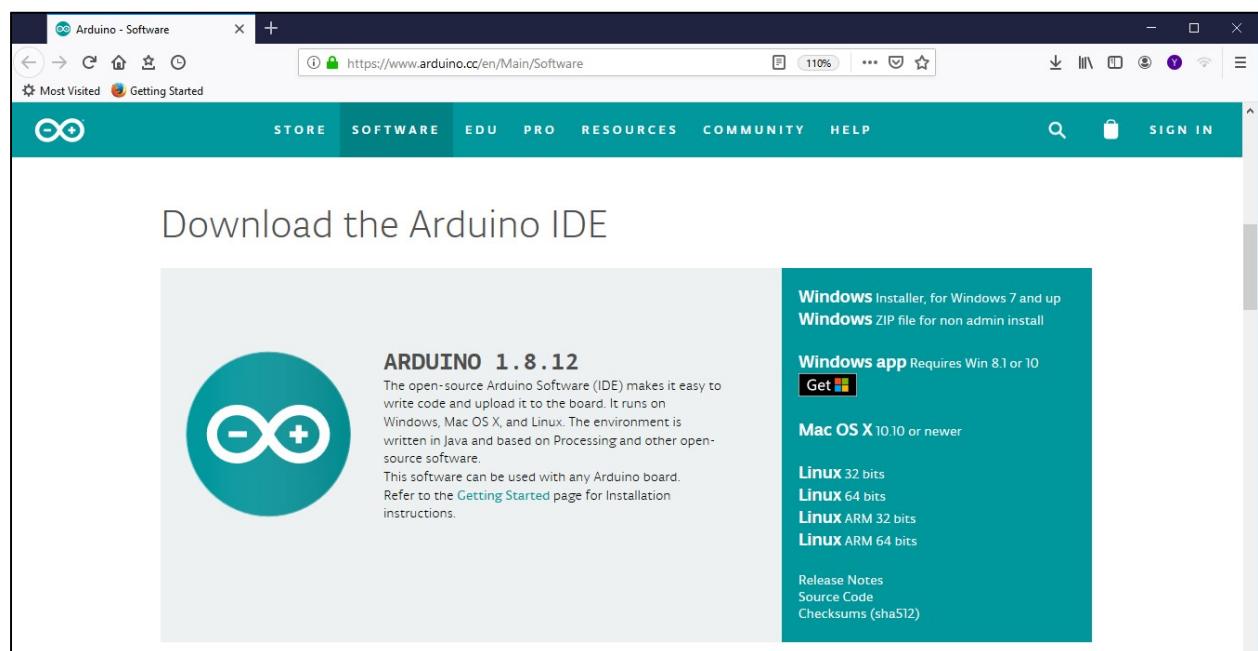


While we are waiting for our shipment to arrive, let's start with downloading the Interactive Development Environment, aka: the IDE. I use a Windows-based Personal Computer, so if you use something else, my examples will look different, but you're probably used to that.

Open an internet browser, I use Firefox but others will work as well, and then visit:

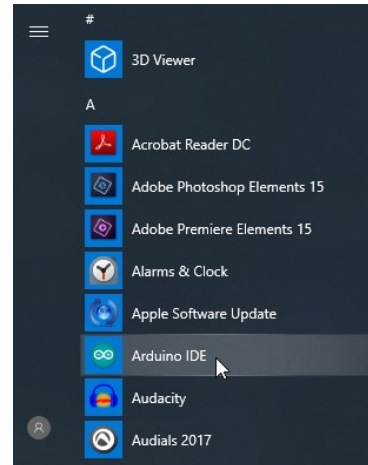
<https://www.arduino.cc/en/Main/Software>

Scroll down about a page's worth, until you see the "Download the Arduino IDE" section and the Arduino logo (a teal-colored circle with an "infinity" sign, with a minus and plus inside the loops).



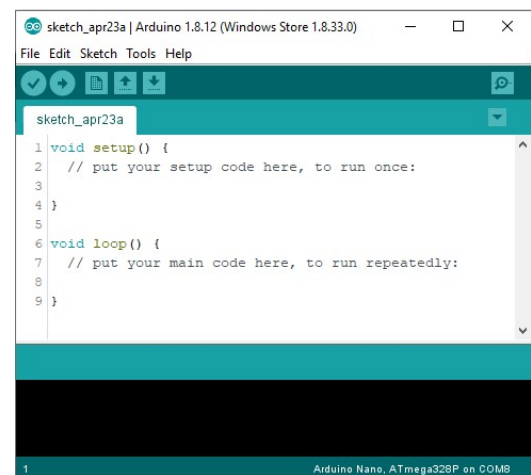
On the right side you will see the words "Windows app Requires Win 8.1 or 10", so click on these words. A new page will open where you can contribute to the development of the IDE software, but for now, click on the "Just Download" button. The IDE is distributed through the Microsoft Store, so don't be surprised if it asks you to login. If you do not have credentials for the Microsoft Store, you will need to register. Once you are inside the store, click on the "Get" and then the "Install" buttons. The download is over 200MB, so it can take a while. Once the installation is completed, you can close the browser windows. If learning from all this text is not your style and you prefer something to watch, the web site at "<https://daackm.github.io>" under the "Chapter 2" heading has a link to a short video showing all the steps.

To launch the IDE, just click on the Windows button in the lower left of the task bar. A list of applications will appear, so just scroll down a bit to the “Arduino IDE” button and click on it. The IDE will open. That’s all there is to it!

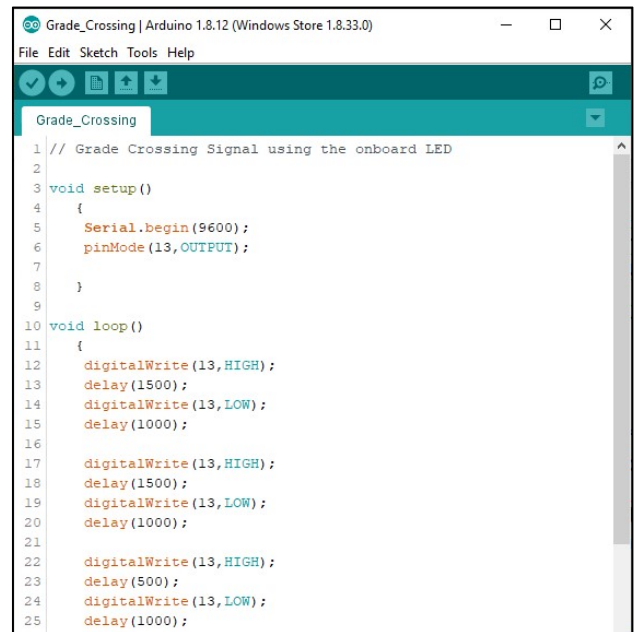


Did I hear you say that your Arduino Nano has arrived? GREAT! Let’s start with the hardware installation. This is pretty easy. Just plug one end of the USB cable into the Nano and the other into a USB port on your PC. The Nano has several LEDs, and one will blink wildly for about 5 seconds, but then settle down to a “heartbeat” indicating that it is functioning properly.

Now it’s time to put the IDE to work. Open the IDE as specified above (the installation process may have placed an icon on your desktop, and you may launch the IDE that way too, by double clicking on the icon). Click on the “File” tab, then on “New” entry; a default, simplistic program will appear, but we don’t want even that much. Click on the “Edit” tab, then the “Select all” entry, then the “Delete” button from the keyboard (or for you keyboard shortcut folks, hit “Control/A” then the delete key). We now have a totally blank sketch.



I have created a test program for you, or “sketch” as they are called in “Arduino-speak”, to help you get started. Visit <https://daackm.github.io> and scroll down to “Chapter 2”. Just below the heading is a line directing you to view the “Grade_Crossing” sketch. Click on that entry. A page will open containing the sketch. Click and hold your mouse button just in front of the first line, drag it to the end of the last line and then release, essentially highlighting the contents of the entire file (or for you keyboard junkies, Control/A and Control/C). Now go back to the IDE and paste the code into the empty sketch. Do a “File” “Save” and give the sketch a name, something like “Grade Crossing”. A portion of the sketch is shown here.

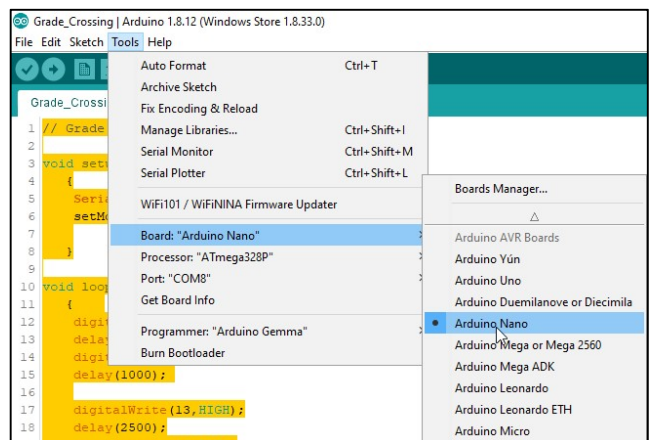


```

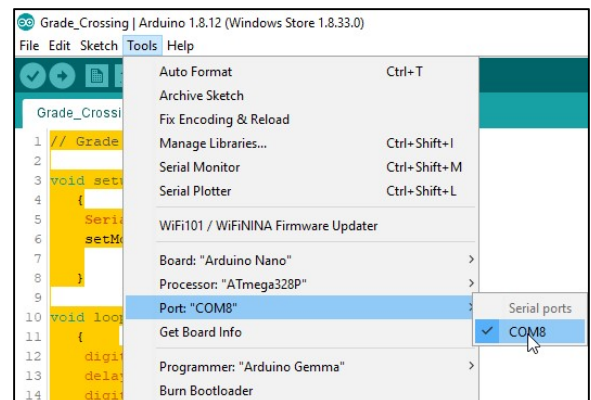
1 // Grade Crossing Signal using the onboard LED
2
3 void setup()
4 {
5   Serial.begin(9600);
6   pinMode(13,OUTPUT);
7
8 }
9
10 void loop()
11 {
12   digitalWrite(13,HIGH);
13   delay(1500);
14   digitalWrite(13,LOW);
15   delay(1000);
16
17   digitalWrite(13,HIGH);
18   delay(1500);
19   digitalWrite(13,LOW);
20   delay(1000);
21
22   digitalWrite(13,HIGH);
23   delay(500);
24   digitalWrite(13,LOW);
25   delay(1000);

```

We are almost ready to run the program, but there are two more steps required to configure the IDE to use our Arduino. There are many types of Arduino microprocessors, the Nano being one of them. And the Arduinos communicate with the IDE over “COM” ports, and there are many of these as well. We need to configure the IDE to talk to our particular Arduino, the Nano. To do so, click on the “Tools” tab, then on the “Board” entry; a window will open showing many different types of Arduino boards, and we want to make sure that “Nano” is the one selected.



Now again click on the “Tools” tab, and then on the “Ports” entry; chances are that the proper port number is already selected. But it is possible that a port number is displayed, but does not have a check mark in front of the number (this can happen when you change Nanos, and the one you are using now uses a different port than the one you used before). If this is the case, just click on the port number suggested by the IDE, and the port parameter will be set properly.



Now to run the sketch. Just under the “Edit” tab is teal colored circle with a right pointing arrow. This icon will analyze our sketch for errors (this is called “Compiling”), and if error-free, will upload the sketch to the Nano and run it. Click on this icon. It may take 15 seconds or more to compile the sketch, and down in the lower left the IDE will tell you whether it is compiling or uploading. When it says “Done Uploading”, take a look at the blinking LED on the Nano. The LED in the center is indicating that power is on, but one on the side is blinking in a “Long Long Short Long” pattern, like a locomotive approaching a crossing. After blinking the pattern, it goes quiet for 5 seconds and then repeats the pattern in an infinite loop. SUCCESS!!!

Now we are going to get a bit more sophisticated, and run a different sketch that has an error in it. Click on the “File” tab, then the “New” entry to again open a minimalistic program. Highlight the entire sketch and hit the “Delete” key from the keyboard to empty its contents. Again return to <https://daackm.github.io>, scroll down to “Chapter 2” and click on the entry labeled “Grade_Crossing_With_Error”. As before, highlight the code, copy it to the clipboard, return to the IDE and paste it in. The code is shown on the next page. Now try to upload it.

```

// Grade Crossing Signal using the onboard LED
int kounter=0

void setup()
{
  Serial.begin(9600);
  pinMode(13,OUTPUT);
}

void loop()
{
  kounter=kounter+1;
  Serial.print("kounter = ");Serial.println(kounter);
  digitalWrite(13,HIGH);
  delay(1500);
  digitalWrite(13,LOW);
  delay(1000);

  digitalWrite(13,HIGH);
  delay(1500);
  digitalWrite(13,LOW);
  delay(1000);

  digitalWrite(13,HIGH);
  delay(500);
  digitalWrite(13,LOW);
  delay(1000);

  digitalWrite(13,HIGH);
  delay(1500);
  digitalWrite(13,LOW);
  delay(1000);

  delay(5000);
}

```

IT DIDN'T WORK! The IDE is no longer a happy camper because there is an error in the second line of the code, the "int kounter=0" line. Almost every line in an Arduino sketch must end in a semi-colon ("voids" and some lines with Curly Braces, like "{" and "}", are exceptions), and line two is missing a semi-colon. The IDE informs you of the error, as shown below. It gives general information about the error and the approximate

place where the error occurred. Here it says that the error is near line 4, character 1, and that a semi-colon is missing. Actually the error was on line 2, but the message gives enough information to find the error.

```
expected ',' or ';' before 'void'
Grade_Crossing_with_Error:4:1: error: expected ',' or ';' before 'void'

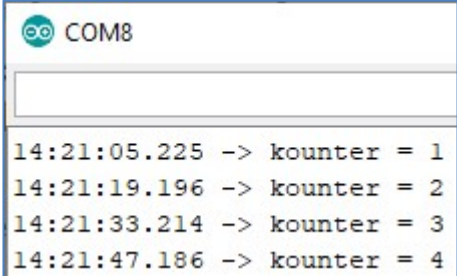
void setup()
^~~~~

exit status 1
expected ',' or ';' before 'void'
```

Let's correct the error. Move your mouse to the end of line 2, enter a semi-colon and re-run the sketch. SUCCESS!

Three more things and this chapter will be completed. First, notice that the first line of the sketch begins with two slashes. Any characters which follow two successive slashes will be interpreted as a comment and ignored; the two slashes can be at the beginning of a line or in the middle, it doesn't matter where; they will be ignored. There is another way to enter comments, a block comment, but we will leave that for another day.

The second thing has to do with debugging logic errors in sketches. Again, upload the sketch in which you corrected the error. Once it is running, click on the "Tools" tab, and then the "Serial Monitor" entry. The sketch will restart, and every time the LEDs begin the sequence, the monitor will display the time of day and the cycle number which is stored in the variable "kounter". How did we do that? By using the "Serial.print" statements to display the value of a variable. This statement is an extremely powerful way to debug code. One thing to note: while the Arduino is connected to the PC the time displayed will correspond to the clock on the PC; when the Arduino is disconnected from the PC, like if it was being employed on your layout and powered by a "wall wart", the time is the time elapsed since the sketch was started.



Time	kounter
14:21:05.225	1
14:21:19.196	2
14:21:33.214	3
14:21:47.186	4

Finally, there are many other good places to learn how to use Arduinos. I particularly like a 30 minute YouTube video from Ron's Trains and Things, where Ron covers some of the same things we have covered here

(<https://www.youtube.com/watch?v=wBn7pHEldWl&t=899s>).

Additionally, I like the series of short tutorials from Ron McWhorter:

(https://www.youtube.com/channel/UCfYfK0tzHZTpNFrc_NDKfTA)

Another good YouTube source is Tom Kvichak of “Toms Trains and Things”

(<https://www.youtube.com/channel/UCE1TsLOIG2wxBZ7fe8512qw>)

Finally, Geoff Bunza is the one whom I consider to be the grandfather of those of us using Arduinos for model railroading. He also covers DCC and writes regularly for Model Train Hobbyist; you can find his blog at

<https://model-railroad-hobbyist.com/blog/geoff-bunza>

And remember that you can always search for “Arduino Tutorial” materials using Google or Yahoo.

Admittedly, these sketches won't add much animation to your model railroad. For heaven's sake, all we did was to “watch” a horn, and heard nothing at all. But knowing how to run a simple program is a building block for knowing how to run a complex program. The following chapters and projects will show you how to do much more, and allow you to spend even more money with great joy.

Thanks for reading.

.