



# Learn Arduboy

INTRODUCTION TO PROGRAMMING BY CREATING SIMPLE GAMES

Noke Codes | Learn To Program, Roanoke! | Part One

## What is Arduboy?

The Arduboy<sup>1</sup> is a handheld game console programmed with open source software, based on the Arduino hardware platform. The original version of the Arduboy was 1.6mm thick, with the height and width of a credit card, and was initially designed by founder Kevin Bates as an electronic business card. Later consumer versions replaced the first version's touch-sensitive panels by physical buttons, and included a protective plastic case, raising the thickness to 5mm. Development was funded through a Kickstarter campaign in 2015.

## What is Noke Codes?

Noke Codes is a civic-minded organization aimed at building a stronger community here in the Roanoke Valley through technology. “Learn To Program, Roanoke!” is a Noke Codes community project to provide fun and educational programming workshops for all ages.

## About the Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs, such as light on a sensor or a finger on a button, and turn it into an output, such as activating a motor or turning on an LED.

You can write code for the Arduino that sends a set of instructions to the microcontroller on the board. This is done with the Arduino programming language (based on Wiring)<sup>2</sup>, and the Arduino Integrated Development Environment (IDE), based on Processing<sup>3</sup>.

## Credits

The Arduboy is supported by an active group of volunteers in the Community Forums.<sup>4</sup> The original Arduboy library was developed as a collaborative effort.<sup>5</sup> An improved Arduboy2 library was developed by MLXXXp (Scott).<sup>6</sup>

---

<sup>1</sup> <https://en.wikipedia.org/wiki/Arduboy>

<sup>2</sup> [https://en.wikipedia.org/wiki/Wiring\\_\(development\\_platform\)](https://en.wikipedia.org/wiki/Wiring_(development_platform))

<sup>3</sup> [https://en.wikipedia.org/wiki/Processing\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Processing_(programming_language))

<sup>4</sup> <https://community.arduboy.com/>

<sup>5</sup> <https://github.com/Arduboy/Arduboy/graphs/contributors>

<sup>6</sup> <https://github.com/MLXXXp/Arduboy2>

## Introduction

What is programming? Why would someone want to learn how to program? Computer programming is a wide and diverse field and offers many challenges. It is also very rewarding as one learns to solve problems and develop new skills. Does everyone that learns programming become a software developer or computer engineer? No. Does learning to program help someone develop problem solving skills that can apply to many different aspects of life? Very definitely, yes!

This workshop is designed for all ages. No matter if someone is an absolute beginner or has some previous experience with programming, there is always something new one can learn. **Don't worry if your programs don't always work the first time.** One important tip is that failure is a part of the normal learning process. Identify what went wrong, learn from it and try again!

*"I have not failed. I've just found 10,000 ways that won't work."*

*"Negative results are just what I want. They're just as valuable to me as positive results. I can never find the thing that does the job best until I find the ones that don't."*

- Thomas A. Edison

The Arduino hardware platform is very open and flexible. That flexibility can also make learning to program in the beginning a bit scary and overly complex. That's where the Arduboy comes in – the built in OLED screen, buttons and compact protective case provides an excellent way to focus more on learning the code and provides immediate feedback to the learner.

This workshop will contain three segments:

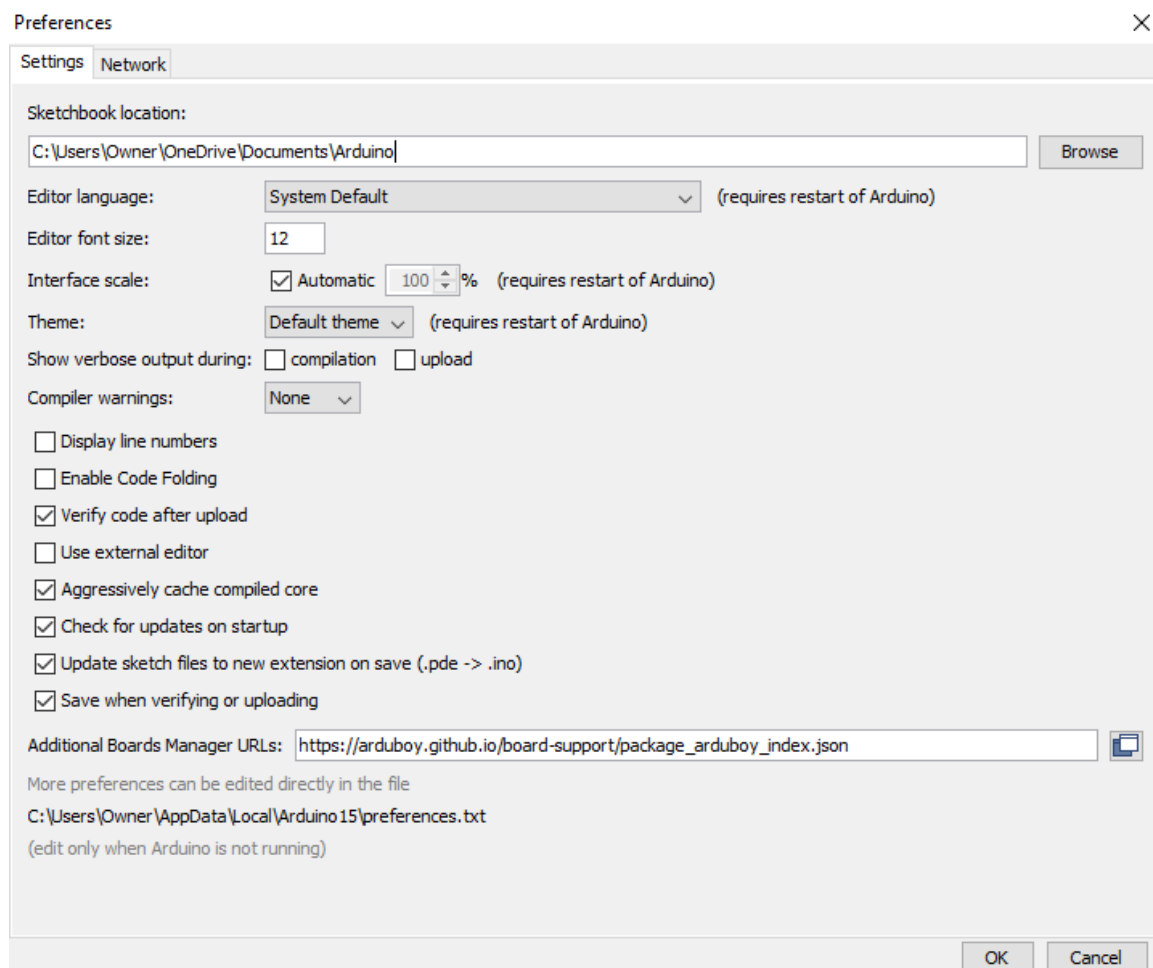
- **Hello World** - an introduction to the Arduino Integrated Development Environment (IDE) and connecting the computer to the Arduboy with an USB cable. To begin the journey, this lesson will include the ever-popular "Hello World" as the very first program and then advance to other variations of displaying text on the screen.
- **Eight Ball Magic** - as the next step, this segment will include a text game based on the classic Magic Eight Ball. This simple program demonstrates using an array to store the responses to display, capture button presses, code different functions and incorporate bitmaps that are displayed on the screen.
- **Ardubreakout** – this exercise will load a popular "breakout" style game to the Arduboy. Students can then step through how the game works and how to make changes to customize the game.

## Getting Started

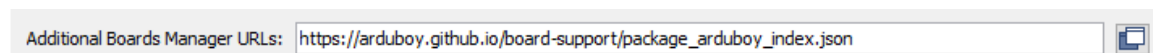
The microprocessor in the Arduboy is an Atmel ATmega32u4, which is the same chip in the Arduino Leonardo. Because of this, programming can be done with the Arduino IDE software. There is a complete “Quick Start Guide” on the Arduboy web site<sup>7</sup> which we will be following in the workshop.

### Adding Board Support

Open the Arduino IDE software and navigate to File > Preferences



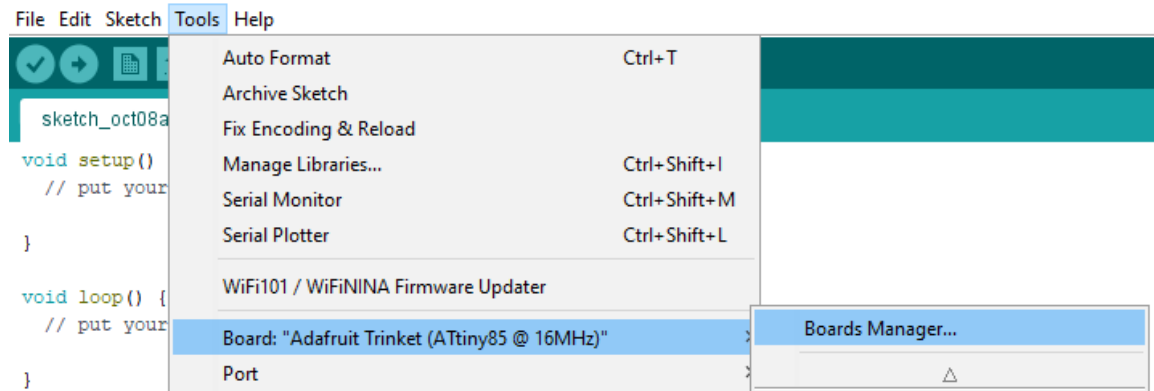
Enter this URL [https://arduboy.github.io/board-support/package\\_arduboy\\_index.json](https://arduboy.github.io/board-support/package_arduboy_index.json)



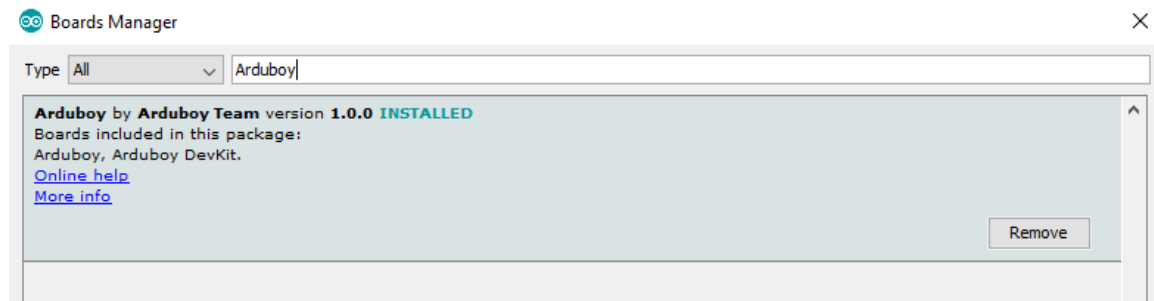
And then click “Ok” to accept.

<sup>7</sup> <https://community.arduboy.com/t/quick-start-guide/2790>

Next, add the Arduboy inside the Boards Manager, found under the Tools menu.



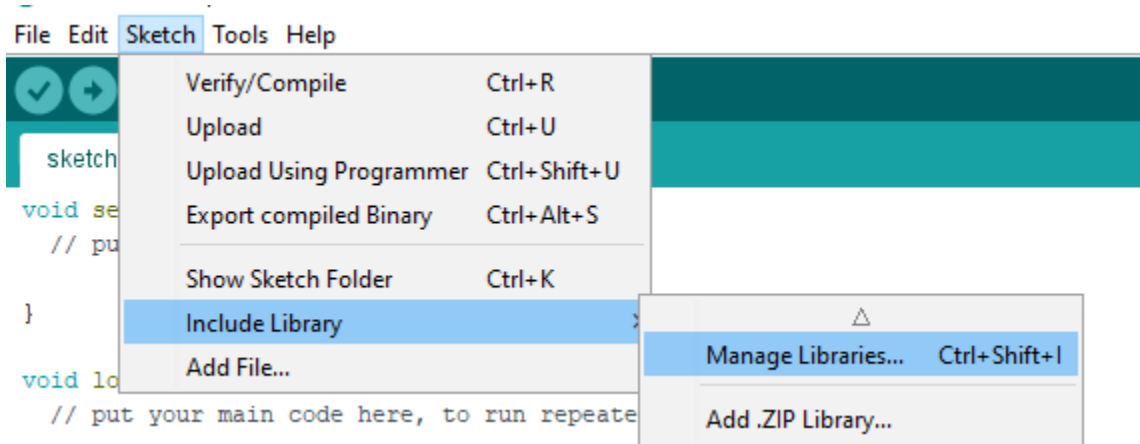
In the search box, type in Arduboy and then click the Install button. After it has been installed, it should look like this:



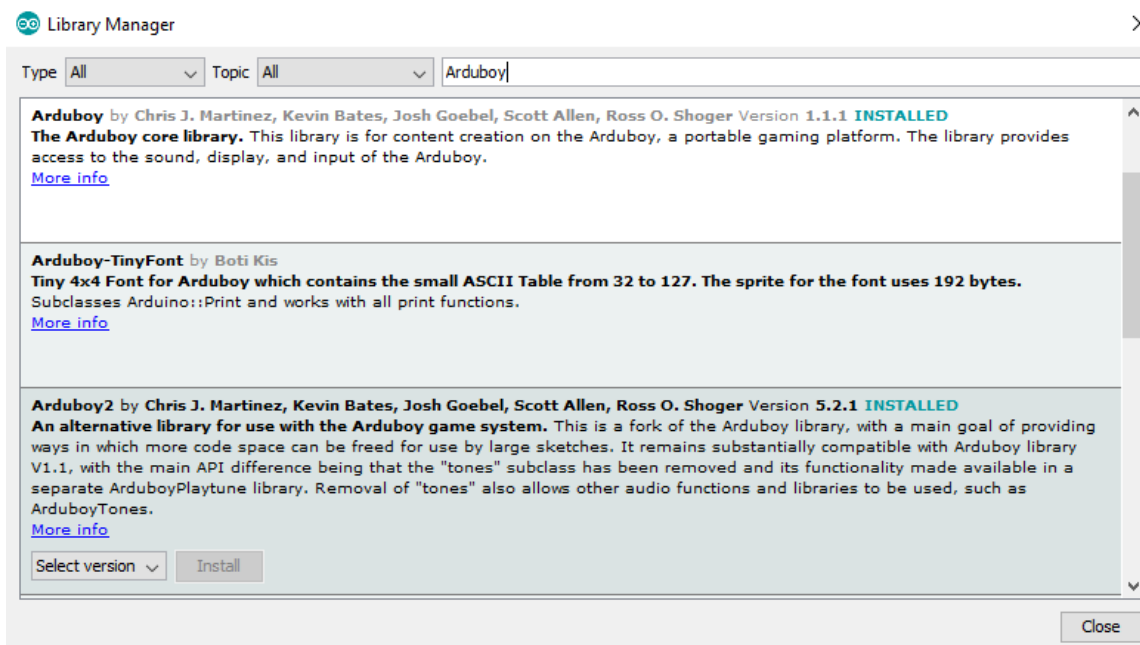
## Installing the Libraries

The libraries make it easier to access the Arduboy hardware (buttons, speaker and OLED screen) by providing an “Application Programming Interface” (API). Both the original Arduboy and the improved Arduboy2 libraries will be used for this workshop. **It is recommended at all of the Arduboy libraries be installed** for any other games that might need them (otherwise, when compiling the code, some strange errors may occur).

Go to Sketch > Include Library > Manage Libraries



Type in Arduboy in the search box and all of the related libraries should be displayed. Once selected, the Arduboy and Arduboy2 libraries should show as “Installed.”



It is recommended that all of the related Arduboy libraries be installed for any other games that might rely on them being present.

We are now ready to begin programming the Arduboy!

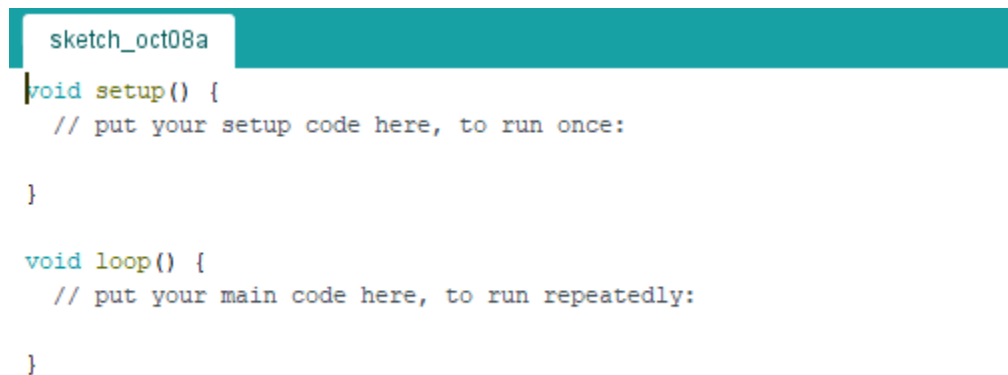
# Hello World

## Concepts

- `setup()`
- `loop()`
- Comments
- Class
- Instance
- Functions

The classical “Hello World” example is a common way to begin the programming experience. Once installed in the Arduino IDE, the Arduboy libraries have several built-in examples for exploring the code and the capabilities of the Arduboy.

Upon opening the Arduino IDE, there is a blank “sketch” (the Arduino name for a program) with some code:



```
sketch_oct08a
void setup() {
  // put your setup code here, to run once:

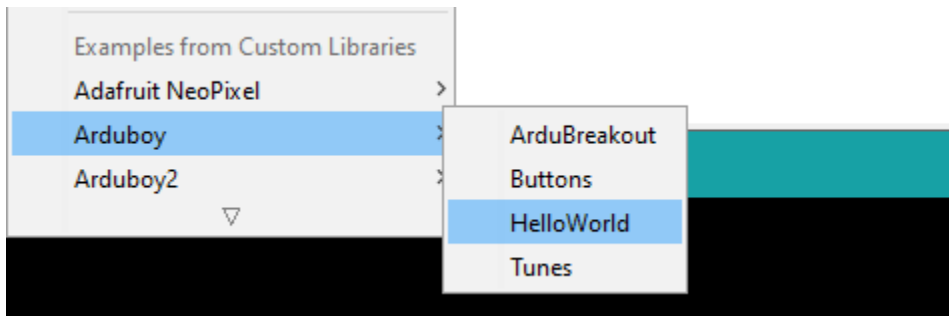
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Every sketch needs to have these two functions. The word “void” means the function is not going to return any values. Each function has a pair of curly brackets {} (also called braces) that contains the code to execute when the function is run. As noted in the comment (a single-line comment begins with a double forward-slash //), `setup()` is run once when the device is powered up. Configuration items for the Arduboy can be placed here. The `loop()` function contains the main logic for the program, which repeats endlessly as long as power to the unit is on.

Open the built-in Hello World sketch from the File > Examples menu, scroll down to the Arduboy section:



Let's examine this sketch one section at a time. First, there is a comment block (created with a `/*` at the beginning and `*/` at the end) with credits to the author of the program and other information. Before the required `setup()` function, there is some code. To use what is contained inside the Arduboy library, the program needs to be told what to include in the code when the program is compiled.

```
#include "Arduboy.h"

// make an instance of arduboy used for many functions
Arduboy arduboy;
```

Notice that certain words are in different colors. These are “reserved” words that have a special meaning inside the IDE. When the `Arduboy.h` file is included, the program now has access to a **Class** (a collection of code that creates an Object – and it's associated functions). The creators of the Arduboy library has already done this coding so that every programmer doesn't have to “re-invent the wheel” each time. This is an example of **Object Oriented Programming (OOP)** and the concept of code reuse. The next line of code creates an **instance** of the Arduboy object. What this means to the programmer (you!), is all the detailed code does not have to be re-typed over and over each time in a program – simply create an instance of that object.



Next, there are two items inside the *setup()* function.

```
// This function runs once in your game.
// use it for anything that needs to be set only once in your game.
void setup() {
  // initiate arduboy instance
  arduboy.begin();

  // here we set the framerate to 15, we do not need to run at
  // default 60 and it saves us battery life
  arduboy.setFrameRate(15);
}
```

To initiate the named instance, the line of code basically means “let the code begin” – the format is “name of instance” dot “the function to call” – *arduboy.begin()* in this example. The other line of code sets the frame refresh rate on the Arduboy’s OLED screen. In the example given, it uses a power-saving 15 frames per second, instead of the default 60 frames.

**Tip:** the named instance can be whatever the programmer chooses. It is helpful to make the name meaningful and easy to remember. Try making the name of the instance something shorter:

```
#include "Arduboy.h"

// make an instance of arduboy used for many functions
Arduboy ab;

// This function runs once in your game.
// use it for anything that needs to be set only once in your game.
void setup() {
  // initiate arduboy instance
  ab.begin();

  // here we set the framerate to 15, we do not need to run at
  // default 60 and it saves us battery life
  ab.setFrameRate(15);
}
```

**Questions:** Is that better or worse? Would someone else reading your code immediately understand what is being created?

Moving on to the *loop()* function:

```
// our main game loop, this runs once every cycle/frame.
// this is where our game logic goes.
void loop() {
  // pause render until it's time for the next frame
  if (!(arduboy.nextFrame()))
    return;

  // first we clear our screen to black
  arduboy.clear();

  // we set our cursor 5 pixels to the right and 10 down from the top
  // (positions start at 0, 0)
  arduboy.setCursor(4, 9);

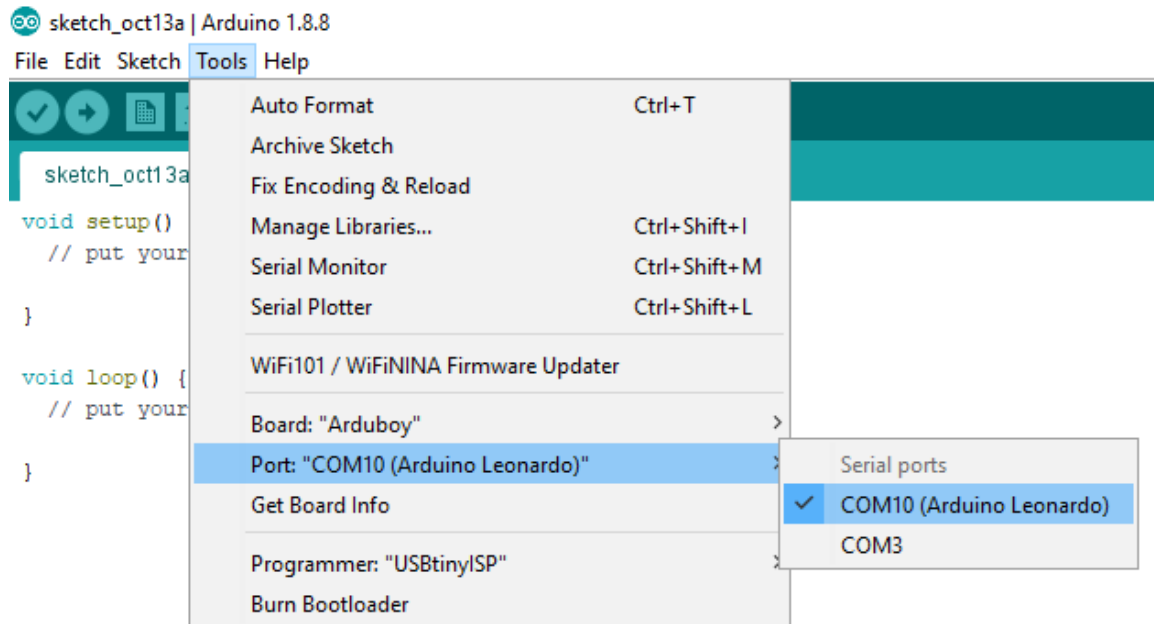
  // then we print to screen what is in the Quotation marks ""
  arduboy.print(F("Hello, world!"));

  // then we finally we tell the arduboy to display what we just wrote to the display
  arduboy.display();
}
```

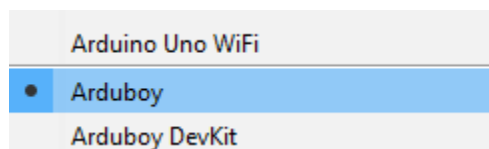
## Uploading the Sketch

Now that we've taken a closer look at the Hello World sketch, it's time to compile the code and upload the program to the Arduboy. The board support and libraries have already been installed, but to upload successfully, there are a couple more items to check.

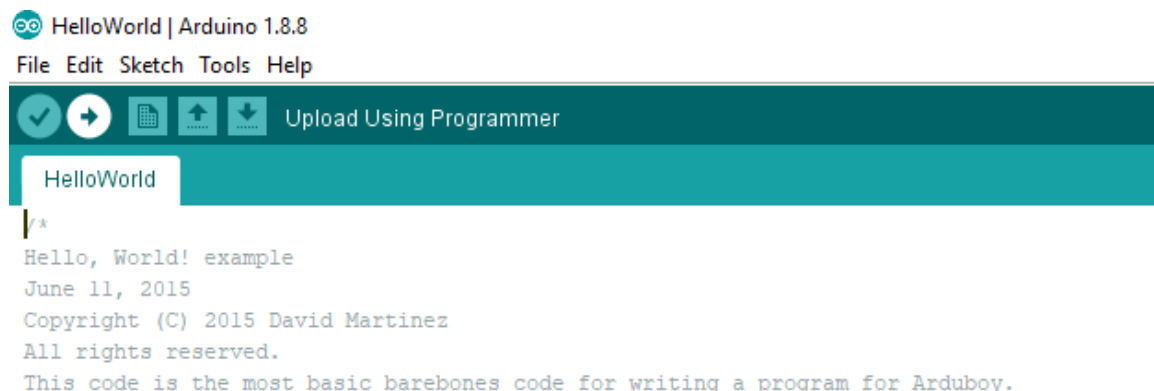
Once the Arduboy is plugged into one of the computer's USB ports, slide the unit's power switch to the right to the "On" position. Next, open the Arduino IDE (if not already) and look at Tools > Board: Arduboy and Port: COM5 (Arduino Leonardo). **Note:** the COM number may vary depending on each PC.



If the Board listed is something other than Arduboy, you may need to click on **the > symbol** and scroll down the list of supported boards to select it.



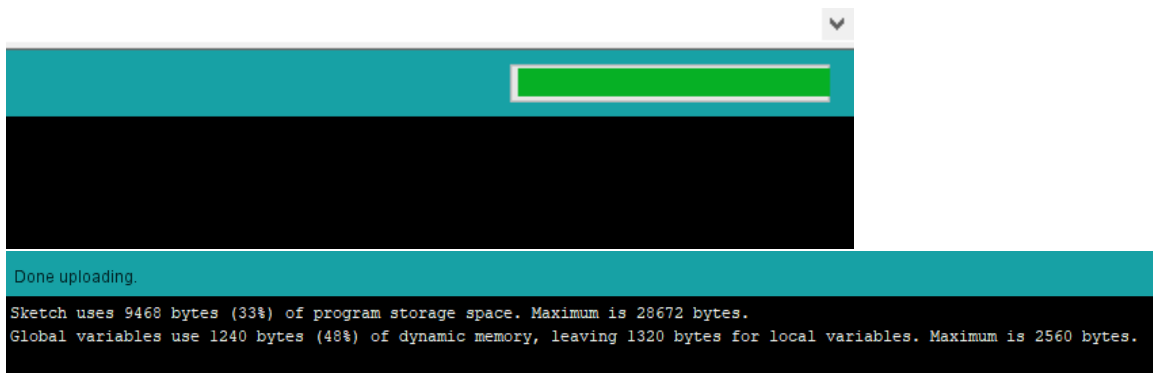
Now that communication between the Arduino IDE and the Arduboy has been established, the code can be compiled and uploaded. There are two special buttons on the top menu bar just for that purpose.



Just below the “File” option, there is a button with a check mark. This button compiles the code and checks for errors. Any errors will be listed in the window across the bottom of the IDE. If successful, there will be a message similar to this:

```
Done compiling.  
  
Sketch uses 9468 bytes (33%) of program storage space. Maximum is 28672 bytes.  
Global variables use 1240 bytes (48%) of dynamic memory, leaving 1320 bytes for local variables. Maximum is 2560 bytes.
```

No errors and ready to upload? That is the next button to the right with an arrow. Clicking this will do one more verification and then upload the compiled code to the Arduboy. There is a progress bar that shows you the status and when it is completed:



Again, if there are any errors in the process, they will be displayed in this window of the IDE.

Did the program run and display “Hello, world!” on the Arduboy screen?

**Congratulations**, you’ve uploaded your very first Arduboy program!

# Hello World Plus Plus

## Concepts

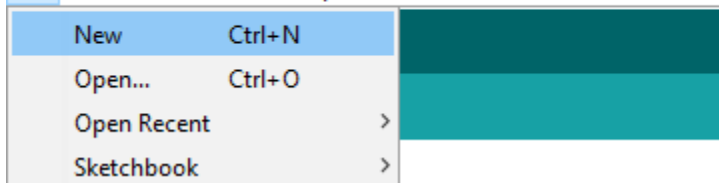
- Arduboy2 Library
- Variables
- Integers
- Calling a Function
- Loops
- Increment operators
- Watching for a Button press

Writing a message to the screen was a good start, but not very exciting. In this exercise, we'll look at moving the text across the screen to make it more interesting to look at. We'll also use the newer Arduboy2 library. While the original Arduboy library is still available for older games that may need it, no new development is being done with this library. That is why it is better to create any **new** programs with the current version of the Arduboy2 library. This exercise will also look at using the buttons on the Arduboy to make some type of action happen.

To get started, open a new Sketch and include the Arduboy2 library:

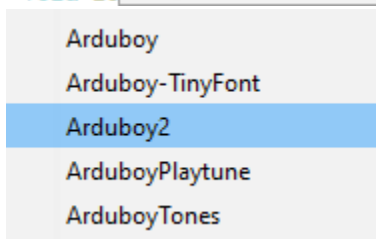
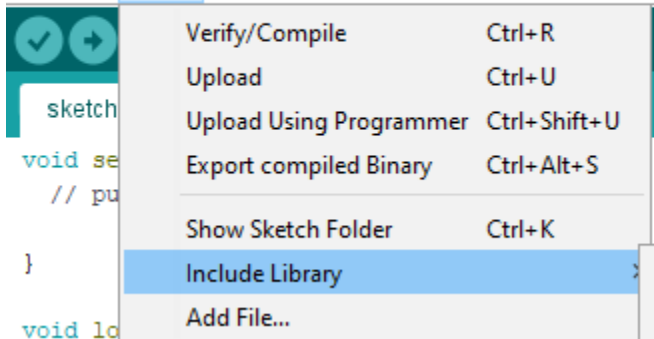
sketch\_oct13b | Arduino 1.8.8

File Edit Sketch Tools Help

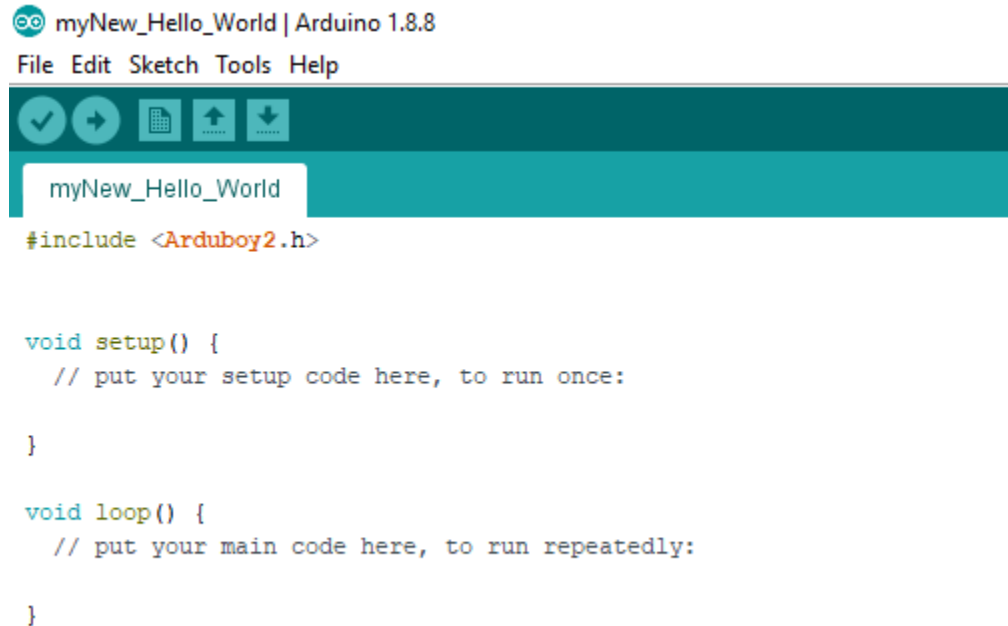


sketch\_oct13b | Arduino 1.8.8

File Edit Sketch Tools Help



Save your new sketch and give it a name that describes what it is. It should look like this:



To use what is in the Arduboy2 library and access the functionality, we need to create an instance of the **Arduboy2** Class. We also need a couple of variables that are declared as an integer.

```
#include <Arduboy2.h>

// make an instance of arduboy
Arduboy2 myArduboy;

// Create variables for starting coordinates
int x;
int y;
```

The C++ language is a “strongly typed” language, meaning that variables must have the data type declared before use. There are a few primitive data types available that are built into the language and available for use in a variable: **integers** (int), **booleans** (bool), **characters** (char), **floating point** (float) and others. In this sketch, we will be using these variables to hold the X, Y coordinates of where the text will be written on the screen. Examples: `arduboy.setCursor(4, 9)` can use variables `arduboy.setCursor(x, y)`.

## Create a Function

Now we begin to program in the good stuff – code that will print the “Hello World” on the screen and give it an appearance of motion. We will do this with a loop that will run the code multiple times and change the value of the X, Y coordinates. This function is created with this code. The first part, *void*, means the function does not return a value. The empty parenthesis means that no parameters are being passed to the function.

```
void intro() {  
    x = 4;  
    y = 9;  
    // Use a loop to let the phrase scroll down the screen  
    for(int i = 0; i < 20; i++) {  
        // first we clear our screen to black  
        myArduboy.clear();  
  
        // we set our cursor 5 pixels to the right and 10 down from the top  
        // (positions start at 0, 0)  
        myArduboy.setCursor(x, y);  
  
        // then we print to screen what is in the Quotation marks ""  
        myArduboy.print(F("Hello, world!"));  
  
        // then we finally we tell the arduboy to display what we just wrote to the display  
        myArduboy.display();  
        x++;  
        y++;  
        // needs a short pause  
        delay(75);  
    }  
}
```

Let's take a closer look at the loop. Using the *for* statement, the loop will run the code inside the curly brackets each time the conditions are True. **Translation:** With a variable *int i* set to equal zero, as long as *i* is than 20, run the code and increment *i* plus 1. Inside the *intro()* function, the *x* and *y* variables are initially set to 4 and 9 (look familiar?). Inside the loop, the screen is cleared, *setCursor(x, y)* uses the current values of the variables, the *print()* statement contains the words to print inside the quotes, and the *display()* function writes the text to the screen. Finally, the function increments the values of the *x* and *y* variables. There is a short delay to slow the process down so it doesn't happen instantly.



This new function will then be **called** inside the built-in *setup()* function:

```
// This function runs once in your game.
// use it for anything that needs to be set only once in your game.
void setup() {
  // initiate arduboy instance
  myArduboy.boot();

  // here we set the framerate to 15, we do not need to run at
  // default 60 and it saves us battery life
  myArduboy.setFrameRate(15);
  // Call the function for scroll down intro
  intro();
  // Pause 1 seconds before going on
  delay(1000);
}
```

At this point, we don't have anything inside the built-in *loop()* function.

```
void loop() {
  // put your main code here, to run repeatedly:

}
```

Go ahead to verify the code to look for any typo errors and upload to the Arduboy. Did "Hello, world!" move down the screen? Awesome!

## Button Presses

The next goal is to print something different on the screen when a button is pressed. We'll follow the pattern already established with the *intro()* function and call the new function *pressed\_A()*:

```

void pressed_A() {
    x = 4;
    y = 9;
    // Use a loop to let the phrase scroll down the screen
    for(int i = 0; i < 20; i++) {
        // first we clear our screen to black
        myArduboy.clear();

        // we set our cursor 5 pixels to the right and 10 down from the top
        // (positions start at 0, 0)
        myArduboy.setCursor(x, y);

        // then we print to screen what is in the Quotation marks ""
        myArduboy.print(F("You pressed A"));

        // then we finally we tell the arduboy to display what we just wrote to the display
        myArduboy.display();
        x++;
        y++;
        // needs a short pause
        delay(75);
    }
}

```

Follow this same code pattern to create another function called *pressed\_B()*.

Now, it's time for some code placed inside the built-in *loop()* function.

```
void loop() {  
  // pause render until it's time for the next frame  
  if (!myArduboy.nextFrame())  
    return;  
  
  myArduboy.pollButtons();  
  
  myArduboy.clear();  
  
  myArduboy.setCursor(20,29);  
  myArduboy.print(F("Press a button"));  
  myArduboy.display();  
  
  if(myArduboy.justPressed(A_BUTTON)) {  
    pressed_A();  
    delay(500);  
  }  
  
  if(myArduboy.justPressed(B_BUTTON)) {  
    pressed_B();  
    delay(500);  
  }  
  
}
```

---

The function *pollButtons()* is included in the Arduboy2 library to make it easy to capture when a button is pressed. The Arduboy has six buttons with known pins connected to the Atmel Leonardo microprocessor. Instead of having to write code to address those pins, all the work has been hidden away in the library.

**Might be a good time to say “Thank You” to all the  
volunteers that have worked on the Arduboy Open  
Source project!**

Next, inside the *loop()* function, the screen is cleared, the *setCursor()* defines a position to write some instructions on the screen, simply prompting “Press a button” for the user.

The two *if()* statements use the *justPressed()* function to then call either the *pressed\_A()* or *pressed\_B()* functions created earlier in the sketch.

Verify the code and upload to the Arduboy. **Good luck!**

This concludes part one of the **Learn Arduboy** series.