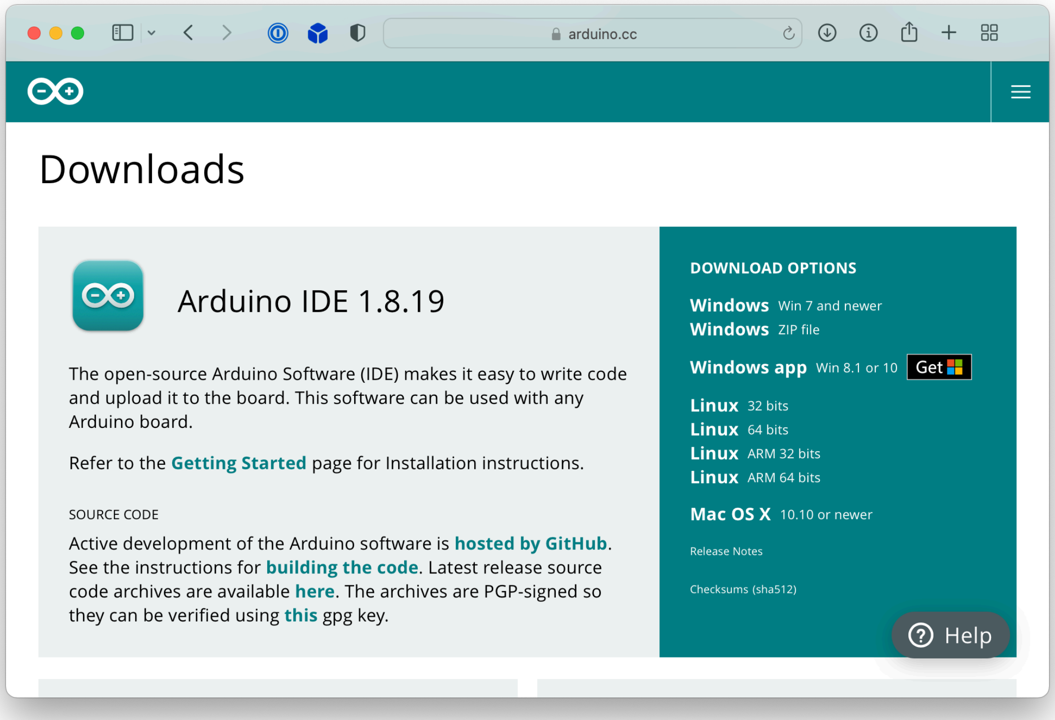
1 Setting Up

1.1 Installation

First, install the **Arduino IDE**. Navigate to `https://www.arduino.cc` in a web browser. Click the '**Software**' tab. Download and install the version of the Arduino IDE for your operating system.



Downloads

Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Getting Started** page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file

Windows app Win 8.1 or 10 Get

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Mac OS X 10.10 or newer

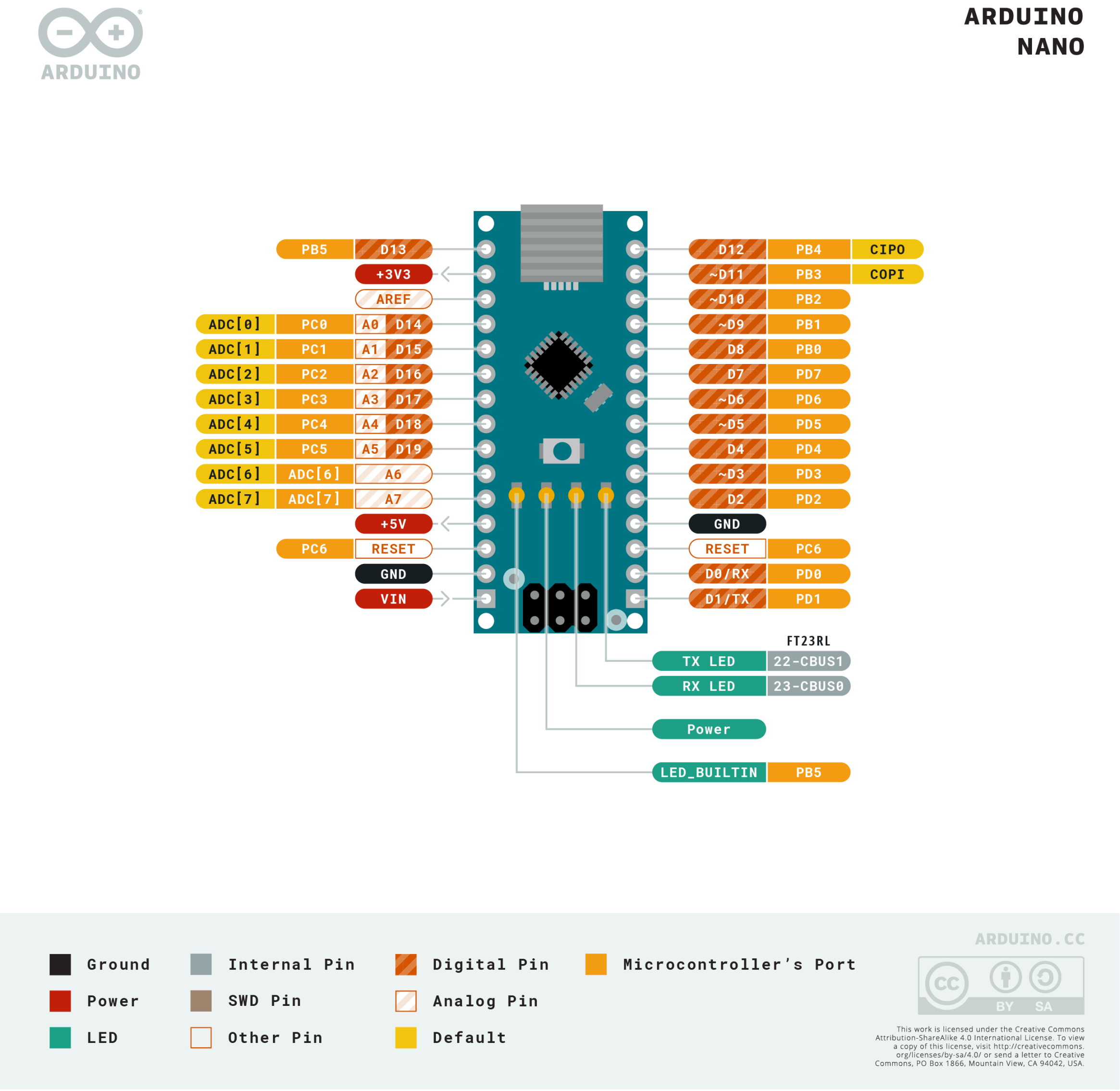
Release Notes

Checksums (sha512)

1.2 Drivers

Some of our Arduino Nanos use USB-Serial convertor chips that require additional drivers (particularly on Macs). If your Arduino does not show up in Tools → Port, try downloading additional drivers by following this guide: `https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all`

2 Arduino Hardware



Digital Pins The digital pins (D0–D19) can either read or output on and off values. For example, you could use a digital pin to turn on an LED, or read a button press.

Analogue Pins The analogue pins (A0–A7) can read or write analogue values, ranging from 0 to 1. For example, you could use an analogue pin to read a temperature sensor, a distance sensor, or a potentiometer.

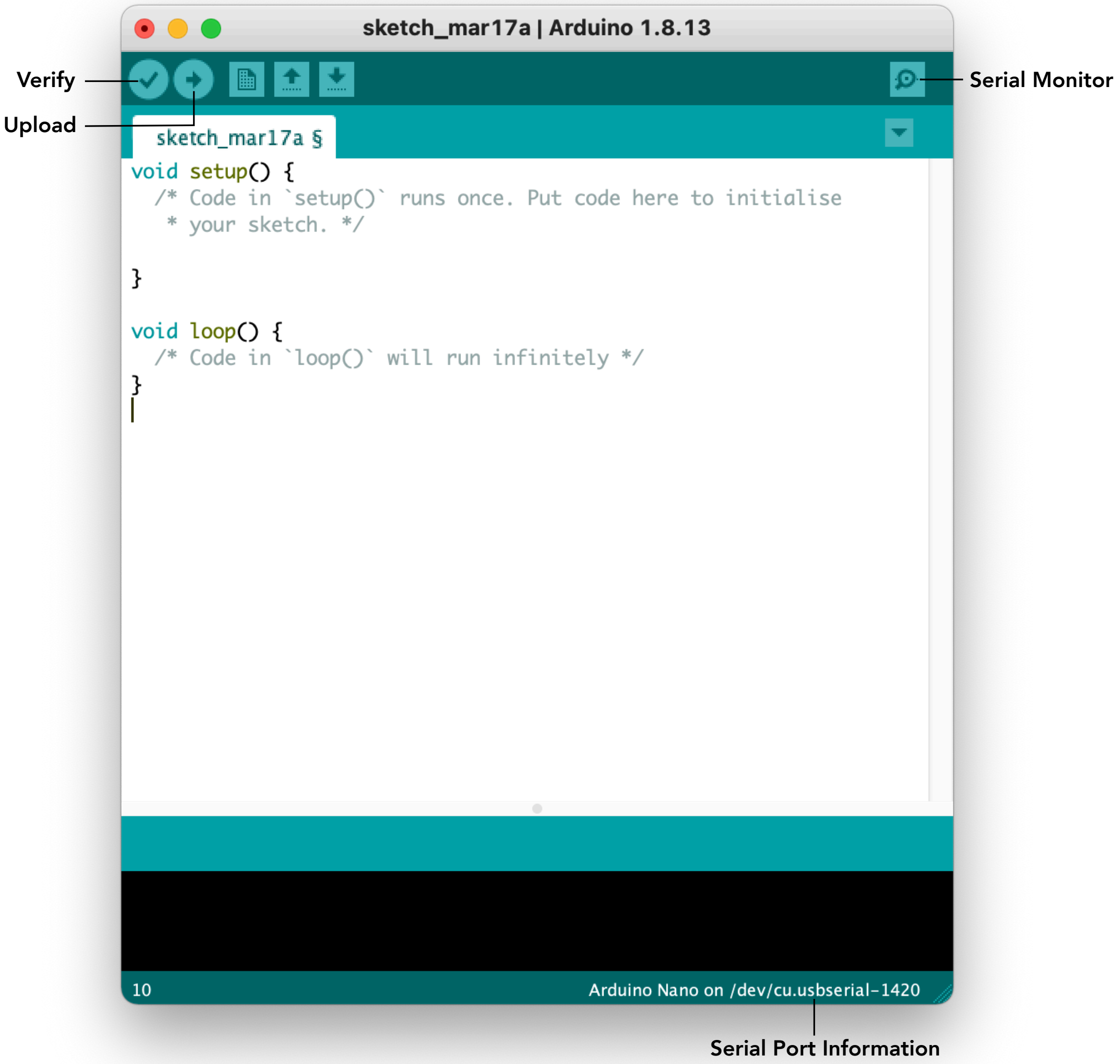
PWM Pins Pins D3, D5, D6, D9, D10 and D11, marked with a tilde in the diagram, are capable of pulse-width modulation—a technique for emulating analogue output with a digital pin, by cycling power to the pin at a fast rate.

Power Pins The Arduino can provide power at 5 volts (from the +5V pin) and 3.3 volts (from the +3V3 pin). The Nano has two GND (negative) pins.

3 Programming the Arduino

3.1 Using the IDE

When you first launch the Arduino IDE, you will see an interface like this:


sketch_mar17a | Arduino 1.8.13
Verify
Upload
Serial Monitor
sketch_mar17a §
10
Arduino Nano on /dev/cu.usbserial-1420
Serial Port Information

```
void setup() {
  /* Code in `setup()` runs once. Put code here to initialise
   * your sketch. */

}

void loop() {
  /* Code in `loop()` will run infinitely */
}
```

Verify Checks that your code is valid, by compiling it to machine code.

Upload Compiles your code and uploads it to the Arduino board.

Serial Monitor Opens the serial monitor window, which shows messages from the Arduino and allows you to send messages to the Arduino.

3.2 Blink an LED

We'll start with the classic Arduino example: blinking an LED. The code below will continually blink an LED on and off:

```
// The setup function runs once when you press reset or power the board
void setup() {
  // Initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// The loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // Turn the LED on (HIGH is the voltage level)
  delay(1000);                       // Wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // Turn the LED off by making the voltage LOW
  delay(1000);                       // Wait for a second
}
```

There's a few functions being introduced at once here. To explain:

`setup()` This function runs once, when the program first starts. Any code to initialise your sketch goes here.

`pinMode(pin, mode)` Initialises a pin as either an input. The pin is a number (in this code, `LED_BUILTIN` is automatically replaced with the pin number of the Arduino's built-in LED), and mode can either be `INPUT`, `OUTPUT`, or `INPUT_PULLUP`.

> ⚠ Do not output to a pin that has not been initialised as an output with the `pinMode()` function.

`loop()` This function is run over and over again, until your Arduino is powered off. Most of your code will go here.

`delay(millis)` Pauses your sketch for the given number of milliseconds. (There are 1000 milliseconds in a second.)

`digitalWrite(pin, value)` Turns a pin on or off. Writing a `HIGH` value turns on a pin; writing a `LOW` value turns it off.

3.2.1 Uploading the Sketch

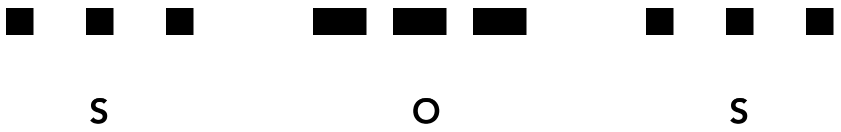
> If your board does not appear in the Tools → Port menu, check that the board is properly connected to your computer, and restart the Arduino IDE. If the board still does not appear, follow the driver installation guide in section 1.2.

First, make sure the Arduino IDE is configured to program Arduino Nanos. From the main menu, select **Tools** → **Board** → **Arduino AVR Boards** → **Arduino Nano**.

Then, in the main menu, select your board's serial port from **Tools** → **Port**.

- On Windows, this will start with `COM*`
- On Mac, this will start with `/dev/cu.usbserial*` or `/dev/cu.usbmodem*`
- On Linux, this will start with `/dev/tty*`

3.3 Exercise: SOS Blink



The message 'SOS' in Morse code is 3 long flashes, followed by 3 short flashes, followed by 3 long flashes. Modify your blink code to blink an SOS message. Use a length of 150ms for the short flashes, and a length of 450ms for the long flashes.

3.4 Button Toggle

Write a sketch to use a button (connected to pin 2) to toggle the LED each time it is pressed.

Use the `digitalRead(pin)` function to read the value from a pin (remember to set the pin as an input).

```
// Constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // The number of the pushbutton pin
const int ledPin =  13;      // The number of the LED pin

// Variables will change:
int buttonState = 0;         // Variable for reading the pushbutton status

void setup() {
  // Initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // Initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // Read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // Check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // Turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // Turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```