Published December 2021

Iffy Books

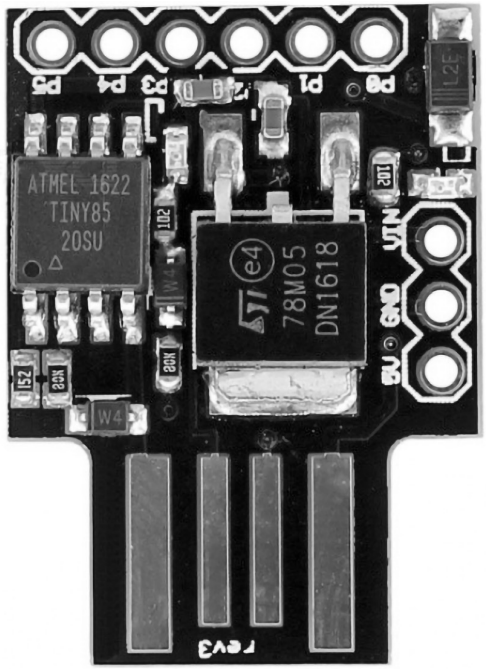
Philadelphia, Pennsylvania

Version 0.7

Download this zine as a PDF:
https://iffybooks.net/mousejiggler

No rights reserved.

Program a Digispark mouse jiggler



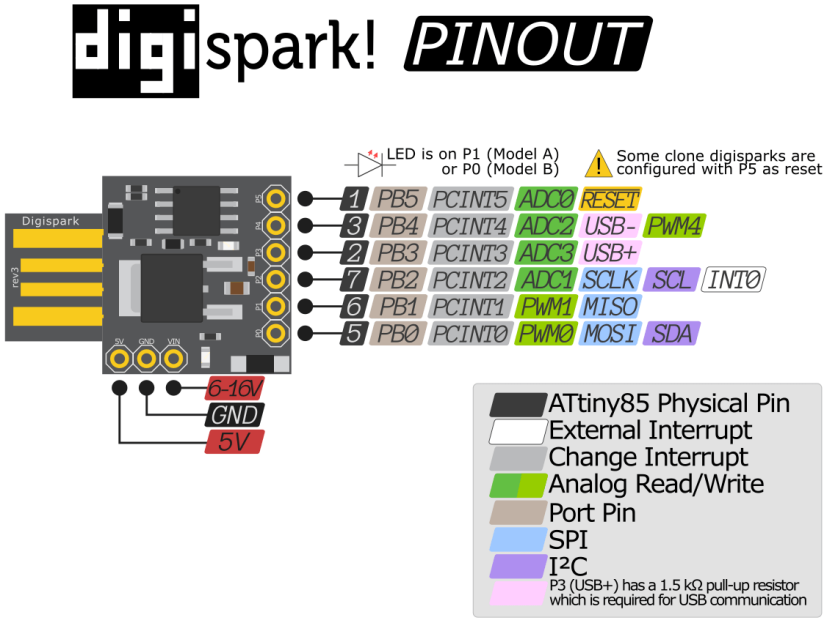
a how-to guide from Iffy Books

319 N. 11th St. #3D
Philadelphia, PA 19107

Join our email list
at iffybooks.net

Follow @iffybooks
on social media

Send corrections to
iffybooks@protonmail.com

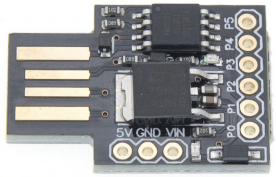


A Digispark is a tiny microcontroller board (similar to an Arduino) that can connect to a computer's USB port and pretend to be a mouse or keyboard.

This zine will teach you to program a Digispark mouse jiggler using the Arduino programming language. When you plug your Digispark clone into a computer's USB port, it will quietly jiggle the mouse cursor to prevent the computer from going to sleep. You'll also learn to program a keyboard simulator, which can type any sequence of keys you want.

The Digispark is built around the ATtiny85, a microcontroller chip developed by the company Atmel (based in San Jose, CA and acquired by Microchip Technology in 2016). The ATtiny85 has 8 KB of flash memory and a clock speed of 20 MHz.

To make the ATtiny85 chip easier to use for DIY projects, the company Digistump (based in Portland, OR) created the Digispark. They don't sell Digispark boards anymore, but it's easy to find cheap clones based on their designs. This tutorial uses a Digispark Rev.3 clone that looks like this:



The Digispark has a built-in USB interface, a 500mA 5V regulator, 6 I/O Pins, and 8KB of flash memory (~6KB available after writing the USB bootloader). It has two LEDs: a green one for power, and a red one you can control using code. You'll program your Digispark clone using the **Arduino IDE** (integrated development environment).

This tutorial includes instructions for **macOS**, **Windows** and **Ubuntu**. (With macOS, your computer might not recognize the Digispark as a USB mouse every time you plug it in. Unplug and plug it back in a few times, and it should work.)

> *Note:* If you're using a Digispark clone that doesn't have a USB bootloader installed, read the appendix first to learn how to burn Micronucleus to your device.

➡️ Install the Arduino IDE (Ubuntu and similar)

The **Arduino IDE** (integrated development environment) is the application you'll use to program your Digispark clone.

Installation method 1: Snap package manager

Open a terminal window and run the following command to install the Arduino IDE using the Snap package manager. You'll need to enter your password, and it will take a few minutes to install.

```
sudo snap install arduino
```

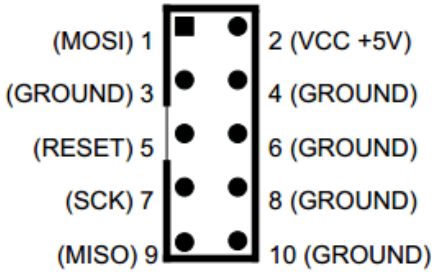
Now run the following command to add your username to the dialout group. This will let you program the board via USB.

```
sudo usermod -a -G dialout $USER
```

Restart your computer.

Open a terminal window and run the command **`arduino`** to start the Arduino IDE.

Note: The pinout diagram below is for the pins on the USBASP device, without the ribbon wire attached.



Once the wires are connected, plug the USBASP into your computer's USB port. A green light on your Digispark clone will turn on.

Run the following command to flash the Micronucleus firmware to your Digispark. (Replace line breaks with spaces.)

```
avrdude -c USBasp -p attiny85  -U lfuse:w:0xe1:m
   -U hfuse:w:0xdd:m -U efuse:w:0xfe:m -B 20
        /<add path here>/t85_default.hex
```

If you're successful, you'll see a confirmation in the terminal. Disconnect the USBASP from your USB port, then disconnect the wires from the Digispark. Now you can program your Digispark clone from the Arduino IDE!

Installation method 2: APT package manager

Depending on the version of Ubuntu you're using, you may be able to install Arduino using the APT package manager. Simply type the following command in a terminal window and press enter. You'll need to enter your password, and it will take a few minutes to install.

```
sudo apt-get install arduino
```

Now run the following command to add your username to the dialout group. This will let you program the board via USB.

```
sudo usermod -a -G dialout $USER
```

Run the command **`arduino`** to start the Arduino IDE.

Installation method 3: Installer Script

Go to the following URL and follow the steps to install the Arduino IDE using the installer script from **https://arduino.cc**:

https://linoxide.com/how-to-install-arduino-ide-on-ubuntu-20-04

➡️ Install the Arduino IDE (macOS)

If you're using macOS, you'll need **version 2.0 of the Arduino IDE**, which is currently in beta. To download it, go to the following URL and click **MacOS**.

https://www.arduino.cc/en/software#experimental-software

Open the .dmg file you just downloaded and drag the **Arduino IDE** application file to your Applications folder.

➡️ Install the Arduino IDE (Windows)

Go to the following URL and download the Arduino IDE for WIndows. Follow the instructions to install it on your machine.

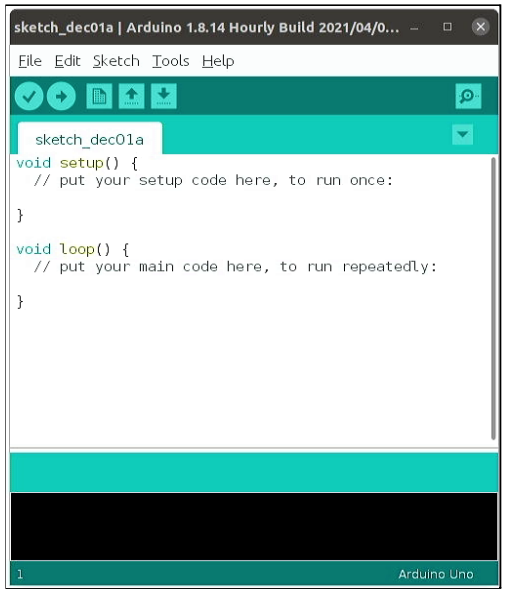
https://www.arduino.cc/en/software

Next, go to the following URL (all one line) to download the Digistump driver. Follow the provided installation instructions.

https://github.com/digistump/DigistumpArduino/releases/download/1.6.7/Digistump.Drivers.zip

➡️ Install Digistump software package

When you open the Arduino IDE, you'll see a window that looks like the one below. Before you write any code, you'll need to install a package with the software required to program your Digispark clone.



If you want to install a bootloader on your Digispark clone, this how-to guide is a good place to start:

https://asdasd.page/2021/Program-Bootloader-of-Digispark-ATTiny85-with-A-ISP-Programmer/

Here's a brief summary of what you'll need to do:

First, run the following command to download the Micronucleus bootloader firmware for the ATtiny85:

```
git clone https://github.com/micronucleus/micronucleus.git
```

Next you'll install the command-line program **`avrdude`**. On Ubuntu (or similar), you can use the command **`sudo apt-get install avrdude`**.

On macOS you can use **`brew install avrdude`**. (You'll need to install the Homebrew package manager first: https://brew.sh.)

If you're using Windows, go to the following URL and follow the instructions to install WinAVR:

http://www.ladyada.net/learn/avr/setup-win.html

Now you'll use wires to connect the pins on your USBASP to the pins on your Digispark clone. Here are the connections you'll need to make:

- MOSI (Pin1) from the programmer to MOSI (PB0) on the board
- MISO (Pin9) from the programmer to MISO (PB1) on the board
- RES (Pin5) from the programmer to RESET (PB5) on the board
- SCK (Pin7) from the programmer to SCK (PB2) on the board
- VCC (VTG Pin2) from the programmer to VCC on the board
- GND (Pin4,6,8,10) from the programmer to GND on the board

➡️ More Digispark resources

- Official Digispark wiki
  https://digistump.com/wiki/digispark

- "USB Rubber Ducky: A Cheeky Prank Device" by Johann Wyss
  https://diyodemag.com/projects/usb_rubber_ducky

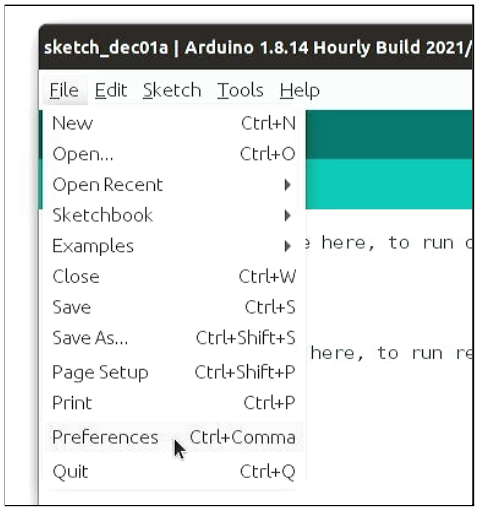
- DigiSpark Scripts by CedArctic (including pranks and malicious code)
  https://github.com/CedArctic/DigiSpark-Scripts

- This article by Eric Draken will show you how to emulate specific mouse and keyboard devices:
  https://ericdraken.com/usb-mouse-jiggler

➡️ Appendix: Burn Micronucleus bootloader to a Digispark using USBASP programmer

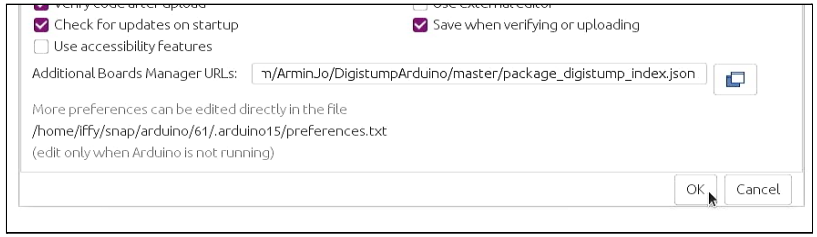
If you're using a plain Digispark clone that doesn't have a bootloader, you'll need to flash one to the device before you program it with the Arduino IDE. Since a bootloader is required to start a USB connection, you won't be able to flash the bootloader over USB. Instead, you'll use a USB-ISP (in-system programmer) device such as the USBASP (below). You can find one online for around $10.

In the menu bar, select **File > Preferences**. (Or **Arduino > Preferences** on macOS.)



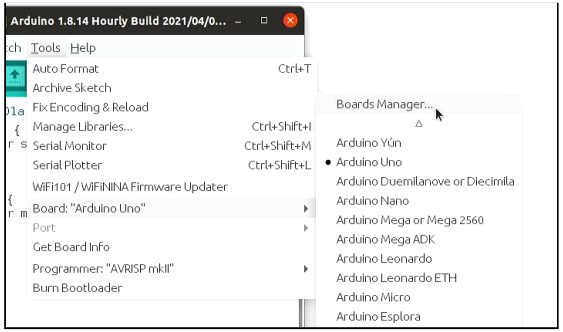
Add the following URL (without the line break) to the list of **Additional Boards Manager URLs**, then click **OK**.

https://raw.githubusercontent.com/ArminJo/DigistumpArduino/master/package_digistump_index.json

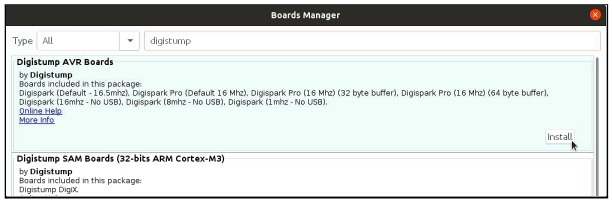


Close and relaunch the Arduino IDE.

In the menu bar, go to **Tools > Board > Boards Manager...**.



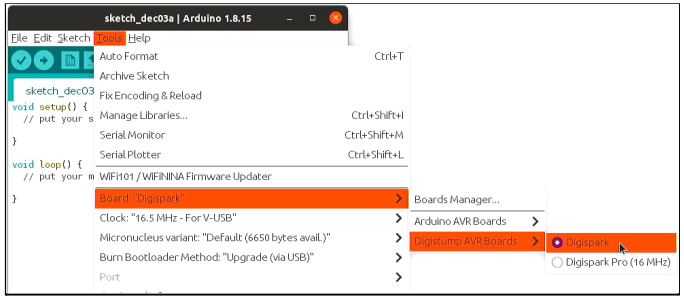
In the search box at the top of the Boards Manager window, type "digistump." Look for the package called **Digistump AVR Boards** and click **Install**. Installation will take a minute or so.



You may need to click **Install** again to get the latest version of the package (version 1.7.5 as of December 2021).

Click **Close** to exit the Boards Manager window.

In the menu bar, go to **Tools > Board > Digistump AVR Boards** and select **Digispark**.



➡️ Completely reset the Arduino IDE

If something goes wrong with the Arduino IDE, you may need to reset its packages and settings. First, quit the application.

On Linux, run the following command to delete the directory that contains the Arduino IDE's packages and settings:

```
rm -rf ~/.arduino15/
```

If you're using macOS, open Terminal and run the following command instead:

```
rm -rf ~/Library/Arduino15/
```

On Windows, open PowerShell and run the following command:

```
rd /s /q C:\Users\{username}\AppData\Local\Arduino15
```

➡️ View connected USB devices

If you're using Ubuntu (or similar), open a terminal window and run the following command to see a list of connected USB devices:

```
lsusb
```

On macOS, open Terminal and use this command:

```
system_profiler SPUSBDataType
```

If you're using Windows 10 or higher, open PowerShell and run the following command to view connected USB devices.

```
pnputil /enum-devices | findstr USB
```

➡️ Random string generator

This program types a pseudorandom series of letters and numbers. The function **`randomSeed(analogRead(5))`** uses random noise from pin 5 to seed the random number generator.

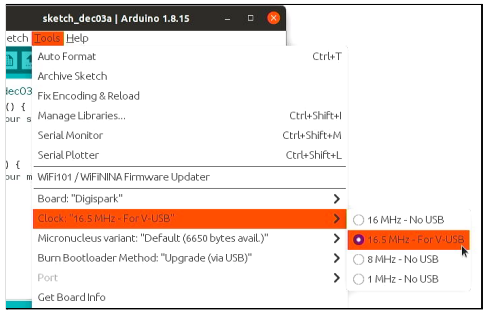
```
#include <DigiKeyboard.h>

void setup(){
  // Seed pseudorandom generator using noise from pin 5
  randomSeed(analogRead(5));
}

void loop() {
  //Choose a random key code
  unsigned int char_number = random(4, 40);

  //Type key
  DigiKeyboard.sendKeyStroke(char_number);

  // Pause for 0.1 second
  DigiKeyboard.delay(100);
}
```

➡️ Save and close everything

The code below presses Ctrl+S and Ctrl+W repeatedly.

```
#include <DigiKeyboard.h>

void setup(){}

void loop() {
  // Pause for 1.5 seconds
  DigiKeyboard.delay(1500);

  // Press Ctrl+S (or command+S for macOS)
  DigiKeyboard.sendKeyStroke(KEY_S, MOD_CONTROL_LEFT);
  //DigiKeyboard.sendKeyStroke(KEY_S, MOD_GUI_LEFT);

  // Pause for 1.5 seconds
  DigiKeyboard.delay(1500);

  // Press Ctrl+W (or command+W for macOS)
  DigiKeyboard.sendKeyStroke(KEY_W, MOD_CONTROL_LEFT);
  //DigiKeyboard.sendKeyStroke(KEY_W, MOD_GUI_LEFT);
}
```

Under **Tools > Clock** in the menu bar, select **16.5 MHz - for V-USB**.



Finally, go to **Tools > Programmer** and select **micronucleus.2.5**.

You're now ready to program your Digispark clone. Nice work!

➡️ Make the LED blink

As a test, let's make the red LED on your Digispark blink on and off. Type out the example code in the Arduino IDE, or download it from the following URL:

https://iffybooks.net/mousejiggler

```
#include <DigiMouse.h>

void setup(){
  pinMode(1, OUTPUT);
  DigiMouse.begin();
}

void loop() {
  digitalWrite(1, HIGH);
  DigiMouse.delay(1000);
  digitalWrite(1, LOW);
  DigiMouse.delay(1000);
}
```

The first line of code, **#include <DigiMouse.h>**, imports the mouse software you'll use.

When you plug your Digispark clone into a computer's USB port, it will automatically run a function called **`setup()`**, then a function called **`loop()`**. In the example code, **`setup()`** has two lines of code:

1. The function **`pinMode(1, OUTPUT)`** sets pin 1 to accept output. Pin 1 controls the red LED.
2. The **`DigiMouse.begin()`** function initializes the mouse software.

The **`loop()`** function runs next, executing the same four lines of code infinitely:

1. The function **`digitalWrite(1, HIGH)`** turns the red LED on.
2. The function **`DigiMouse.delay(1000)`** pauses for 1 second.
3. The function **`digitalWrite(1, LOW)`** turns the red LED off.
4. The function **`DigiMouse.delay(1000)`** pauses for 1 second.

If you've written code in C, C++, or Java, the syntax of the Arduino programming language will look familiar. Curly brackets (**`{}`**) are used to enclose code inside loops and function definitions. After each code statement, you'll need to use a semicolon (**`;`**). Whitespace (spaces, tabs, line breaks) doesn't affect your code.

Functions, such as **`DigiMouse.delay(1000)`** or **`digitalWrite(1, HIGH)`**, can accept one or more arguments (i.e., input values) in the parentheses following the function name.

➡️ Some useful DigiKeyboard functions

`DigiKeyboard.print("Hello!")`	Types the characters "Hello!" (without quotation marks)
`DigiKeyboard.sendKeyStroke(KEY_S, MOD_CONTROL_LEFT);`	Presses Ctrl + S (Windows/Linux shortcut)
`DigiKeyboard.sendKeyStroke(KEY_S, MOD_GUI_LEFT);`	Presses Command + S (macOS shortcut)
`DigiKeyboard.sendKeyStroke(KEY_ENTER);`	Presses enter
`DigiKeyboard.sendKeyStroke(KEY_SPACE);`	Presses the space bar
`DigiKeyboard.sendKeyStroke(random(4, 40));`	Presses a random letter or number

You can reference the complete list of key codes and names in the file **DigiKeyboard.h**, which you can find at the following URL:

https://github.com/digistump/DigisparkArduinoIntegration/blob/master/libraries/DigisparkKeyboard/DigiKeyboard.h

➡️ Program a keyboard simulator

In addition to simulating a mouse, your Digispark clone can pretend to be a keyboard. The code below types the sentence "This is a test!", then presses enter and pauses for 500 milliseconds, or 0.5 seconds.

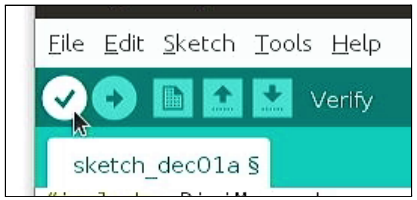
```
#include <DigiKeyboard.h>

void setup() {}

// Infinite loop
void loop() {
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.delay(500);
  DigiKeyboard.print("This is a test!");
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
}
```

Including the function **`DigiKeyboard.sendKeyStroke(0)`** at the beginning of the loop isn't required, but it helps ensure the first real keystroke doesn't get cut off.

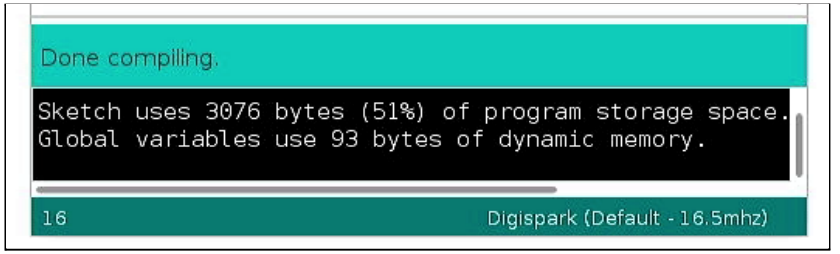
The example code below types "https://iffybooks.net", then presses enter. Then a **for loop** at the end runs infinitely, which effectively disconnects the device.

```
#include <DigiKeyboard.h>

void setup() {
  pinMode(1, OUTPUT); //LED on Model A
}

// Continued on the next page …
void loop() {
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.delay(500);
  DigiKeyboard.print("https://iffybooks.net");
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  for(;;){/*Infinite loop to disconnect device*/}
}
```

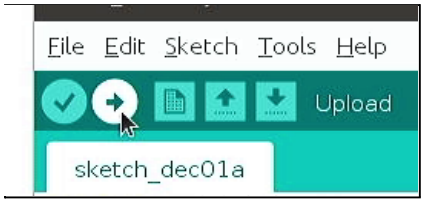
Click the **Verify** icon to compile your code. You'll be prompted to save your project first.



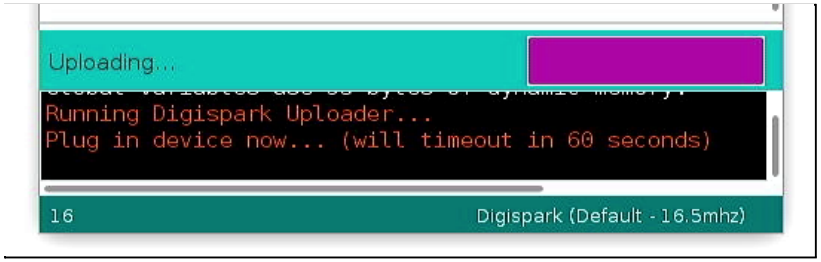
If your code compiled successfully, you'll see a message like this:



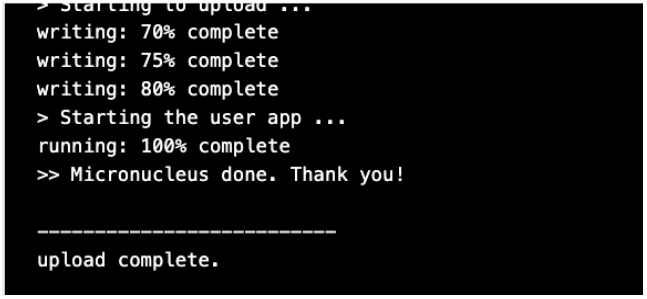
Make sure your Digispark is disconnected from your computer. Then click the **Upload** icon at the top of the Arduino IDE.



At the bottom of the Arduino IDE window, you'll see a message that says "Plug in device now... (will timeout in 60 seconds)."



Plug your Digispark into a USB port on your computer and wait a few moments for your code to upload. If it's a success, you'll see "upload complete" in the console.



When the upload is done, the red light on the board should start blinking on and off.

Disconnect the Digispark from your computer and plug it back in. It should start blinking again. Pretty neat!

➡️ Program a basic mouse jiggler

The example code below will move the mouse cursor in a 20-pixel square every 2 seconds.

First, the **`setup()`** function runs two lines of code: **`pinMode(1, OUTPUT)`** sets up pin 1 for output, which lets you control the red LED. Next, **`DigiMouse.begin()`** initializes the DigiMouse software.

When the **`loop()`** function runs, the code within will repeat infinitely. First, **`DigiMouse.delay(2000)`** pauses for 2000 milliseconds, or 2 seconds. Next, **`DigiMouse.moveX(20)`** moves the mouse to the right 20 pixels, followed by a 0.1-second pause. Then the **`DigiMouse.moveY(20)`** moves the mouse 20 pixels down, and so on.

Finally, the functions **`digitalWrite(1, HIGH)`** and **`digitalWrite(1, LOW)`** turn the LED on for 0.25 seconds.

➡️ Program a random clicker

The program below moves the mouse cursor -250 to 250 pixels in each direction, with the distance selected at random. This one works well for clicking around on Wikipedia.

```
#include <DigiMouse.h>

void setup(){
  pinMode(1, OUTPUT);
  DigiMouse.begin();

  // Turn on LED for 0.25 seconds
  digitalWrite(1, HIGH);
  DigiMouse.delay(250);
  digitalWrite(1, LOW);
}

void loop() {
  // Choose x and y values from -250 to 250
  unsigned int random_x = 250 - random(501);
  unsigned int random_y = 250 - random(501);

  DigiMouse.move(random_x, random_y, 0);
  DigiMouse.delay(500);
  DigiMouse.leftClick();
  DigiMouse.delay(10);
  DigiMouse.setButtons(0);
  DigiMouse.delay(random(5000));
}
```

```
void loop() {
  // Choose x and y values from -3 to 3
  unsigned int random_x = 3 - random(7);
  unsigned int random_y = 3 - random(7);

  // Move the mouse cursor
  DigiMouse.moveX(random_x);
  DigiMouse.moveY(random_y);

  // Turn on the LED for 0.25 seconds
  digitalWrite(1, HIGH);
  DigiMouse.delay(250);
  digitalWrite(1, LOW);

  // Pause for 5 to 10 seconds
  DigiMouse.delay(5000 + random(5000));
}
```

➡️ Program an infinite scroller

The code below uses **`DigiMouse.scroll(5)`** to scroll down endlessly (or scroll up, depending on your mouse settings). To change the scroll direction, use the argument **`-5`** instead.

```
#include <DigiMouse.h>

void setup(){
  pinMode(1, OUTPUT);
  DigiMouse.begin();

  // Turn on LED for 0.25 seconds
  digitalWrite(1, HIGH);
  DigiMouse.delay(250);
  digitalWrite(1, LOW);
}

void loop() {
  DigiMouse.scroll(5);
  //DigiMouse.scroll(-5);
  DigiMouse.delay(100);
}
```

```
#include <DigiMouse.h>

void setup(){
  pinMode(1, OUTPUT);
  DigiMouse.begin();
}

void loop() {
  DigiMouse.delay(2000);

  DigiMouse.moveX(20);
  DigiMouse.delay(100);

  DigiMouse.moveY(20);
  DigiMouse.delay(100);

  DigiMouse.moveX(-20);
  DigiMouse.delay(100);

  DigiMouse.moveY(-20);
  DigiMouse.delay(100);

  // Turn on LED for 0.25 seconds
  digitalWrite(1, HIGH);
  DigiMouse.delay(250);
  digitalWrite(1, LOW);
}
```

➡️ Some useful DigiMouse functions

Function	What it does
`DigiMouse.begin();`	Initializes the software you'll use to control your mouse. Typically included in the **setup()** function.
`DigiMouse.delay(1000);`	Pauses for 1000 milliseconds (1 second)
`DigiMouse.moveX(20);`	Moves the cursor 20 pixels to the right

`DigiMouse.moveX(-20);`	Moves the cursor 20 pixels to the left
`DigiMouse.moveY(20);`	Moves the cursor down 20 pixels
`DigiMouse.moveY(-20);`	Moves the cursor up 20 pixels
`DigiMouse.move(15,-33,0);`	Moves the cursor 15 pixels right and 20 pixels up, and scrolls zero pixels
`DigiMouse.scroll(10);`	Scrolls down 10 units (or up, depending on your mouse preferences)
`DigiMouse.scroll(-10);`	Scrolls up 10 units (or down, depending on your mouse preferences)
`DigiMouse.leftClick();`	Presses down the left click button
`DigiMouse.setButtons(1<<0);`	Hold down the left click button
`DigiMouse.setButtons(1<<1);`	Holds down the right click button
`DigiMouse.setButtons(0);`	Releases all mouse buttons
`DigiMouse.update();`	Preserves USB connection if your mouse jiggler pauses for more than a couple seconds

➡️ Some useful Arduino functions

Example function	What it does
`pinMode(1, OUTPUT);`	Configures pin 1 (the red LED) to behave as an output. You should include this in your `setup()` function.
`digitalWrite(1, HIGH);`	Sets pin 1 to "high" voltage, turning on the red LED light
`digitalWrite(1, LOW);`	Sets pin 1 to "low" voltage, turning off the red LED light
`random(100);`	Returns a pseudorandom number from 0 to 99
`random(100, 2500);`	Returns a pseudorandom number from 100 to 2499

➡️ Program a low-key mouse jiggler

If you want your mouse jiggler's movements to be subtle, you can write a program like the one below. It chooses random X and Y values from -3 to 3, then moves the cursor that many pixels. It then blinks the LED and pauses for a random duration from 5 to 10 seconds.

```
#include <DigiMouse.h>

void setup(){
  DigiMouse.begin();
  pinMode(1, OUTPUT);
}

//Continued on the next page ...
```