LED Matrix Scoreboard

Created by Ruiz Brothers

https://learn.adafruit.com/led-matrix-scoreboard

Last updated on 2021-12-08 04:58:06 PM EST
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
<td>5</td>
</tr>
<tr>
<td>• LED Pixel Hoops</td>
<td>5</td>
</tr>
<tr>
<td>• IR Sensing Hoop</td>
<td>5</td>
</tr>
<tr>
<td>• Prerequisite Guides</td>
<td>5</td>
</tr>
<tr>
<td>• Adafruit Parts</td>
<td>6</td>
</tr>
<tr>
<td>• Black LED Acrylic</td>
<td>8</td>
</tr>
<tr>
<td>• Hardware Screws</td>
<td>8</td>
</tr>
<tr>
<td><strong>Circuit Diagram</strong></td>
<td>9</td>
</tr>
<tr>
<td>• Adafruit Library for Fritzing</td>
<td>9</td>
</tr>
<tr>
<td>• Wired Connections</td>
<td>9</td>
</tr>
<tr>
<td>• Powering</td>
<td>10</td>
</tr>
<tr>
<td><strong>3D Printing</strong></td>
<td>10</td>
</tr>
<tr>
<td>• Parts List</td>
<td>10</td>
</tr>
<tr>
<td>• Build Plate</td>
<td>11</td>
</tr>
<tr>
<td>• CAD Assembly</td>
<td>11</td>
</tr>
<tr>
<td>• Slicing Parts</td>
<td>12</td>
</tr>
<tr>
<td><strong>Using Feather M4 Express</strong></td>
<td>12</td>
</tr>
<tr>
<td><strong>CircuitPython on Feather M4 Express</strong></td>
<td>13</td>
</tr>
<tr>
<td>• Set up CircuitPython Quick Start!</td>
<td>13</td>
</tr>
<tr>
<td>• Further Information</td>
<td>15</td>
</tr>
<tr>
<td><strong>Using MatrixPortal</strong></td>
<td>15</td>
</tr>
<tr>
<td><strong>Install CircuitPython</strong></td>
<td>16</td>
</tr>
<tr>
<td>• Set up CircuitPython Quick Start!</td>
<td>16</td>
</tr>
<tr>
<td>• Further Information</td>
<td>17</td>
</tr>
<tr>
<td><strong>Prep the MatrixPortal</strong></td>
<td>19</td>
</tr>
<tr>
<td>• Power Prep</td>
<td>19</td>
</tr>
<tr>
<td>• Power Terminals</td>
<td>20</td>
</tr>
<tr>
<td>• Panel Power</td>
<td>20</td>
</tr>
<tr>
<td>• Board Connection</td>
<td>21</td>
</tr>
<tr>
<td><strong>CircuitPython Libraries</strong></td>
<td>22</td>
</tr>
<tr>
<td>• The Adafruit CircuitPython Library Bundle</td>
<td>23</td>
</tr>
<tr>
<td>• Downloading the Adafruit CircuitPython Library Bundle</td>
<td>24</td>
</tr>
<tr>
<td>• The CircuitPython Community Library Bundle</td>
<td>24</td>
</tr>
<tr>
<td>• Downloading the CircuitPython Community Library Bundle</td>
<td>25</td>
</tr>
<tr>
<td>• Understanding the Bundle</td>
<td>25</td>
</tr>
<tr>
<td>• Example Files</td>
<td>26</td>
</tr>
<tr>
<td>• Copying Libraries to Your Board</td>
<td>26</td>
</tr>
<tr>
<td>• Understanding Which Libraries to Install</td>
<td>27</td>
</tr>
<tr>
<td>• Example: ImportError Due to Missing Library</td>
<td>29</td>
</tr>
<tr>
<td>• Library Install on Non-Express Boards</td>
<td>30</td>
</tr>
<tr>
<td>• Updating CircuitPython Libraries and Examples</td>
<td>31</td>
</tr>
<tr>
<td><strong>Coding the LED Matrix Scoreboard</strong></td>
<td>31</td>
</tr>
</tbody>
</table>
LED Matrix Scoreboard Code Walkthrough

**Hoop Assembly**
- Wiring IR breakbeam receiver
- Wiring IR breakbeam transmitter
- IR breakbeam Hoop Wiring
- Secure IR breakbeam receiver
- Thread Wiring
- Hole Relief for Wiring
- Exit Hole
- Installed IR receiver
- Solder JST cable
- Soldered JST cable
- Secure IR Transmitter

**Wiring Feather**
- Matrix Power Cable
- Feather & FeatherWing Setup
- JST Cables
- Feather & FeatherWing Connections
- Solder Cable – IR breakbeam receiver
- Solder Cable – IR breakbeam TX
- FeatherWing Connections
- Button Wiring
- Solder Signal – IR transmitter
- Wiring STEMMA Speaker
- Wired Feather & FeatherWing
- Wire Check

**Matrix Assembly**
- Install Corner Brackets
- Insert Hex Nuts
- Fitting Corner Brackets
- Installing Feet
- Secure Feet
- Installed Feet
- Button Wiring
- Install Button Mounting Plate
- Secure Button Mounting Plate
- Panel Mount Arcade Button
- Secure Arcade Button
- Install Hoop Brackets to Feet
- Secure Hoop Brackets
- Secured Brackets
- Second Hoop Bracket
- Install Hoop
- Secure Hoop
- Secured Hoop
- Installing Feather & FeatherWing
- Installed Feather & FeatherWing
- STEMMA Speaker
- Mount Stemma Speaker
- Secure Speaker Mount
- Plug In Connections
- Final Build
Overview

LED Pixel Hoops
Build an LED scoreboard with mini IR sensing hoop using an RGB Matrix and CircuitPython! This indoor game lets you shoot hoops and display your score on a 64x32 LED Matrix.

IR Sensing Hoop
The hoop houses an IR breakbeam sensor that detects when a ball has gone through it. The 3D printed parts hold everything together and are designed to be easy to build. There’s an arcade button on the side that’s used for starting a new game and resetting the score. This build is powered by an Adafruit Feather M4 Express and the RGB matrix FeatherWing.

Prerequisite Guides

Take a moment to walk through the following guides to get better understanding of the parts used in this project.

- RGB LED Matrix Guide (https://adafru.it/Me5)
- Adafruit Feather M4 Guide (https://adafru.it/CJN)
- RGB Matrices with CircuitPython (https://adafru.it/L7b)

Black LED acrylic is used to diffuse the LEDs that are otherwise harsh and over exposed. This material features a matte finish and has the ability to transmit any color of light. That makes it perfect for bringing out the vibrance in the colors of the LEDs.

The 3D printed frame is designed with a slot on the side so it’s easy to insert a sheet of Black LED acrylic.
Adafruit Parts

Adafruit Feather M4 Express - Featuring ATSAMD51
It's what you've been waiting for, the Feather M4 Express featuring ATSAMD51. This Feather is fast like a swift, smart like an owl, strong like a ox-bird (it's half ox,...
https://www.adafruit.com/product/3857

Adafruit RGB Matrix Featherwing Kit
Ahoy! It's time to create a dazzling light up project with our new RGB Matrix FeatherWing. Now you can quickly and easily create...
https://www.adafruit.com/product/3036

64x32 RGB LED Matrix - 4mm pitch
Bring a little bit of Times Square into your home with this sweet 64 x 32 square RGB LED matrix panel. These panels are normally used to make video walls, here in New York we see them...
https://www.adafruit.com/product/2278
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 x IR Break Beam Sensor</td>
<td>5mm LEDs</td>
<td><a href="https://www.adafruit.com/product/2168">https://www.adafruit.com/product/2168</a></td>
</tr>
<tr>
<td>1 x IR Break Beam Sensor</td>
<td>3mm LEDs</td>
<td><a href="https://www.adafruit.com/product/2167">https://www.adafruit.com/product/2167</a></td>
</tr>
<tr>
<td>1 x 2-pin JST Cable</td>
<td>Female Connector 100mm</td>
<td><a href="https://www.adafruit.com/product/261">https://www.adafruit.com/product/261</a></td>
</tr>
<tr>
<td>1 x 2-pin JST Cable</td>
<td>Male Connector 200mm</td>
<td><a href="https://www.adafruit.com/product/3814">https://www.adafruit.com/product/3814</a></td>
</tr>
<tr>
<td>1 x 3-pin JST Cable</td>
<td>Male Connector 200mm</td>
<td><a href="https://www.adafruit.com/product/4046">https://www.adafruit.com/product/4046</a></td>
</tr>
<tr>
<td>1 x 3-pin JST Cable</td>
<td>female socket cable</td>
<td><a href="https://www.adafruit.com/product/3894">https://www.adafruit.com/product/3894</a></td>
</tr>
<tr>
<td>1 x 10-wire Ribbon Cable</td>
<td>Silicone Cover Stranded-Core 1m long</td>
<td><a href="https://www.adafruit.com/product/3890">https://www.adafruit.com/product/3890</a></td>
</tr>
</tbody>
</table>

**Adafruit STEMMA Speaker - Plug and Play Audio Amplifier**

Hey, have you heard the good news? With Adafruit STEMMA boards you can easily and safely plug sensors and devices together, like this Adafruit STEMMA Speaker - Plug and Play...

https://www.adafruit.com/product/3885

**Mini LED Arcade Button - 24mm Translucent Clear**

A button is a button, and a switch is a switch, but these translucent arcade buttons are in a class of their own. Particularly because they have LEDs built right in!...

https://www.adafruit.com/product/3429
1 x **Header kit for Feather**  
Female Header Set  
[https://www.adafruit.com/product/2886](https://www.adafruit.com/product/2886)

1 x **M2.5 Screw Kit**  
Black Nylon M2.5  
[https://www.adafruit.com/product/3299](https://www.adafruit.com/product/3299)

1 x **5V 4A Wall Adapter**  
switching power supply  
[https://www.adafruit.com/product/1466](https://www.adafruit.com/product/1466)

---

**Black LED Acrylic**

A sheet of black LED acrylic used to diffuse the RGB LED Matrix. From their website:

> It looks like a matte black acrylic when it is not lit, but add a light behind it and watch it transform. This fantastic material has the ability to transmit light of any color.

Recommended Cut To Size Sheet: 257mm x 128mm (10.12in x 5.04in)

1 x **Black LED Acrylic**  
Black LED Acrylic from TAP Plastics  

---

**Hardware Screws**

Fasteners used to secure 3d printed parts together.

- 8x M3 x 10mm pan head screws  
- 4x M3 x 6mm pan head screws  
- 6x M2.5 x 10mm pan head screws  
- 4x M2.5 hex nuts
Circuit Diagram

The diagram below provides a visual reference for wiring of the components. This diagram was created using the software package Fritzing (https://adafruit.it/oEP).

Adafruit Library for Fritzing

Use Adafruit's Fritzing parts library to create circuit diagrams for your projects. Download the library or just grab individual parts. Get the library and parts from GitHub - Adafruit Fritzing Parts (https://adafruit.it/AYZ).

Wired Connections

- VCC from Speaker to 3V on Feather
- GND from Speaker to GND on Feather
- Signal from Speaker to A0 on FeatherWing
- Button to GND on Feather
- Button to D4 on FeatherWing
- IR break beam (2-wire) to GND on Feather
- IR break beam (2-wire) to 3V on Feather
- IR break beam (3-wire) to GND on Feather
- IR break beam (3-wire) to 3V on Feather
- IR break beam (3-wire) to A1 on FeatherWing
Powering

The RGB Matrix will need to be powered by a [5V 4A wall adapter](https://adafruit.it/e50). 5V USB cable is not enough to fully power the display – Glitches and artifacts are displayed when RGB matrix is only powered by USB cable.

5V 4A (4000mA) switching power supply - UL Listed

Need a lot of 5V power? This switching supply gives a clean regulated 5V output at up to 4 Amps (4000mA). 110 or 240 input, so it works in any country. The plugs are "US...

[https://www.adafruit.com/product/1466](https://www.adafruit.com/product/1466)

3D Printing

Parts List

STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below.

- hoop-190.stl
- hoop-204.stl
- foot a.stl
- foot b.stl
- hoop bracket a.stl
- hoop bracket b.stl
- 2x corner-a.stl
- 2x corner-b.stl
- speaker mount.stl
- button mount.stl

The corner mounts are printed twice. Make two copies of each corner.
Build Plate
The hoop requires a minimum build area of 250mm x 210mm x 200mm. Two sizes are provided. hoop-190.stl has an outer diameter of 190mm. hoop-204.stl has an outer diameter of 204mm. Choose the one that best fits on your printer's build plate.

CAD Assembly
The corner mounts are fitted over the RGB matrix display and feature a spot for additional black LED acrylic. The two foot mounts are placed over the corners and secured with screws. A button mounting plate is secured to the side of the display. The hoop is attached to a set of brackets that are secured to the two mounting feet. The Stemma speaker is attached to a mounting plate and secured to the back of the display.
Slicing Parts

No supports are required. Slice with settings for PLA material.

The parts were sliced using CURA using the slice settings below.

- PLA filament 220c extruder
- 0.2 layer height
- 10% gyroid infill
- 60mm/s print speed
- 60c heated bed

We suggest using PETG filament for stronger more durable parts.

Using Feather M4 Express

You can build this project using the Feather M4 Express. For a detailed list of all parts needed for this project, check out the Overview.

64x32 RGB LED Matrix - 6mm pitch

Bring a little bit of Times Square into your home with this sweet 64x32 square RGB LED matrix panel. These panels are normally used to make video walls, here in New York we see them on...

https://www.adafruit.com/product/2276

Adafruit RGB Matrix Featherwing Kit

Ahoy! It's time to create a dazzling light up project with our new RGB Matrix FeatherWing. Now you can quickly and easily create...

https://www.adafruit.com/product/3036
CircuitPython on Feather M4 Express

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CIRCUITPY drive to iterate.

The following instructions will show you how to install CircuitPython. If you've already installed CircuitPython but are looking to update it or reinstall it, the same steps work for that as well!

Set up CircuitPython Quick Start!

Follow this quick step-by-step for super-fast Python power :)

Download the latest version of CircuitPython for this board via CircuitPython.org

https://adafru.it/Emh

Click the link above and download the latest UF2 file.

Download and save it to your desktop (or wherever is handy).
Plug your Feather M4 into your computer using a known-good USB cable.

A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

Double-click the Reset button next to the USB connector on your board, and you will see the NeoPixel RGB LED turn green. If it turns red, check the USB cable, try another USB port, etc. Note: The little red LED next to the USB connector will pulse red. That's ok!

If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!

You will see a new disk drive appear called FEATHERBOOT.

Drag the adafruit_circuitpython_etc.uf2 file to FEATHERBOOT.
The LED will flash. Then, the FEATHERBOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it, you're done! :)

Further Information

For more detailed info on installing CircuitPython, check out Installing CircuitPython (https://adafru.it/Amd).

Using MatrixPortal

You can build this project with an all-in-one Matrix Portal board, its definitely the easiest and least-expensive way to go about it.

You will need a matrix portal, matrix, and USB C power/data cable

Adafruit Matrix Portal - CircuitPython Powered Internet Display
Folks love our wide selection of RGB matrices and accessories, for making custom colorful LED displays... and our RGB Matrix Shields...
https://www.adafruit.com/product/4745
Install CircuitPython

CircuitPython ([https://adafru.it/tB7](https://adafru.it/tB7)) is a derivative of MicroPython ([https://adafru.it/BeZ](https://adafru.it/BeZ)) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CIRCUITPY drive to iterate.

Set up CircuitPython Quick Start!

Follow this quick step-by-step for super-fast Python power :

[Download the latest version of CircuitPython for this board via circuitpython.org](https://adafru.it/Nte)
Further Information

For more detailed info on installing CircuitPython, check out [Installing CircuitPython (https://adafruit.it/Amd)](https://adafruit.it/Amd).

Click the link above and download the latest UF2 file.

Download and save it to your desktop (or wherever is handy).

Plug your MatrixPortal M4 into your computer using a known-good USB cable.

A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

Double-click the Reset button (indicated by the green arrow) on your board, and you will see the NeoPixel RGB LED (indicated by the magenta arrow) turn green. If it turns red, check the USB cable, try another USB port, etc.

If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!
You will see a new disk drive appear called MATRIXBOOT.

Drag the adafruit_circuitpython_etc.uf2 file to MATRIXBOOT.

The LED will flash. Then, the MATRIXBOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it, you're done! :)

©Adafruit Industries
Prep the MatrixPortal

Power Prep
The MatrixPortal supplies power to the matrix display panel via two standoffs. These come with protective tape applied (part of our manufacturing process) which MUST BE REMOVED!

Use some tweezers or a fingernail to remove the two amber circles.
Power Terminals
Next, screw in the spade connectors to the corresponding standoff.

- red wire goes to +5V
- black wire goes to GND

Panel Power
Plug either one of the four-conductor power plugs into the power connector pins on the panel. The plug can only go in one way, and that way is marked on the board's silkscreen.
Board Connection

Now, plug the board into the left side shrouded 8x2 connector as shown. The orientation matters, so take a moment to confirm that the white indicator arrow on the matrix panel is oriented pointing up and right as seen here and the MatrixPortal overhangs the edge of the panel when connected. This allows you to use the edge buttons from the front side.

Check nothing is impeding the board from plugging in firmly. If there's a plastic nub on the matrix that's keeping the Portal from sitting flat, cut it off with diagonal cutters.
CircuitPython Libraries

As CircuitPython development continues and there are new releases, Adafruit will stop supporting older releases. Visit https://circuitpython.org/downloads to download the latest version of CircuitPython for your board. You must download the CircuitPython Library Bundle that matches your version of CircuitPython. Please update CircuitPython and then visit https://circuitpython.org/libraries to download the latest Library Bundle.

Each CircuitPython program you run needs to have a lot of information to work. The reason CircuitPython is so simple to use is that most of that information is stored in other files and works in the background. These files are called libraries. Some of them are built into CircuitPython. Others are stored on your CIRCUITPY drive in a folder called lib. Part of what makes CircuitPython so great is its ability to store code separately from the firmware itself. Storing code separately from the firmware makes it easier to update both the code you write and the libraries you depend.

Your board may ship with a lib folder already, it's in the base directory of the drive. If not, simply create the folder yourself. When you first install CircuitPython, an empty lib directory will be created for you.
CircuitPython libraries work in the same way as regular Python modules so the [Python docs](https://adafruit.it/rar) are an excellent reference for how it all should work. In Python terms, you can place our library files in the lib directory because it's part of the Python path by default.

One downside of this approach of separate libraries is that they are not built in. To use them, one needs to copy them to the CIRCUITPY drive before they can be used. Fortunately, there is a library bundle.

The bundle and the library releases on GitHub also feature optimized versions of the libraries with the .mpy file extension. These files take less space on the drive and have a smaller memory footprint as they are loaded.

Due to the regular updates and space constraints, Adafruit does not ship boards with the entire bundle. Therefore, you will need to load the libraries you need when you begin working with your board. You can find example code in the guides for your board that depends on external libraries.

Either way, as you start to explore CircuitPython, you'll want to know how to get libraries on board.

**The Adafruit CircuitPython Library Bundle**

Adafruit provides CircuitPython libraries for much of the hardware they provide, including sensors, breakouts and more. To eliminate the need for searching for each library individually, the libraries are available together in the Adafruit CircuitPython Library Bundle. The bundle contains all the files needed to use each library.
Downloading the Adafruit CircuitPython Library Bundle

You can download the latest Adafruit CircuitPython Library Bundle release by clicking the button below. The libraries are being constantly updated and improved, so you'll always want to download the latest bundle.

Match up the bundle version with the version of CircuitPython you are running. For example, you would download the 6.x library bundle if you're running any version of CircuitPython 6, or the 7.x library bundle if you're running any version of CircuitPython 7, etc. If you mix libraries with major CircuitPython versions, you will get incompatible mpy errors due to changes in library interfaces possible during major version changes.

Download the bundle version that matches your CircuitPython firmware version. If you don't know the version, check the version info in boot.out.txt file on the CIRCUITPY drive, or the initial prompt in the CircuitPython REPL. For example, if you're running v7.0.0, download the 7.x library bundle.

There's also a py bundle which contains the uncompressed python files, you probably don't want that unless you are doing advanced work on libraries.

The CircuitPython Community Library Bundle

The CircuitPython Community Library Bundle is made up of libraries written and provided by members of the CircuitPython community. These libraries are often written when community members encountered hardware not supported in the Adafruit Bundle, or to support a personal project. The authors all chose to submit these libraries to the Community Bundle make them available to the community.

These libraries are maintained by their authors and are not supported by Adafruit. As you would with any library, if you run into problems, feel free to file an issue on the GitHub repo for the library. Bear in mind, though, that most of these libraries are supported by a single person and you should be patient about receiving a response. Remember, these folks are not paid by Adafruit, and are volunteering their personal time when possible to provide support.
**Downloading the CircuitPython Community Library Bundle**

You can download the latest CircuitPython Community Library Bundle release by clicking the button below. The libraries are being constantly updated and improved, so you'll always want to download the latest bundle.

[Click for the latest CircuitPython Community Library Bundle release](https://adafru.it/VCn)

The link takes you to the latest release of the CircuitPython Community Library Bundle on GitHub. There are multiple versions of the bundle available. Download the bundle version that matches your CircuitPython firmware version. If you don't know the version, check the version info in `boot_out.txt` file on the CIRCUITPY drive, or the initial prompt in the CircuitPython REPL. For example, if you're running v7.0.0, download the 7.x library bundle.

**Understanding the Bundle**

After downloading the zip, extract its contents. This is usually done by double clicking on the zip. On Mac OSX, it places the file in the same directory as the zip.

Open the bundle folder. Inside you'll find two information files, and two folders. One folder is the `lib` bundle, and the other folder is the examples bundle.

Now open the `lib` folder. When you open the folder, you'll see a large number of `.mpy` files, and folders.
Example Files

All example files from each library are now included in the bundles in an examples directory (as seen above), as well as an examples-only bundle. These are included for two main reasons:

- Allow for quick testing of devices.
- Provide an example base of code, that is easily built upon for individualized purposes.

Copying Libraries to Your Board

First open the lib folder on your CIRCUITPY drive. Then, open the lib folder you extracted from the downloaded zip. Inside you'll find a number of folders and .mpy files. Find the library you'd like to use, and copy it to the lib folder on CIRCUITPY.

If the library is a directory with multiple .mpy files in it, be sure to copy the entire folder to CIRCUITPY/lib.

This also applies to example files. Open the examples folder you extracted from the downloaded zip, and copy the applicable file to your CIRCUITPY drive. Then, rename it to code.py to run it.

If a library has multiple .mpy files contained in a folder, be sure to copy the entire folder to CIRCUITPY/lib.
Understanding Which Libraries to Install

You now know how to load libraries on to your CircuitPython-compatible microcontroller board. You may now be wondering, how do you know which libraries you need to install? Unfortunately, it's not always straightforward. Fortunately, there is an obvious place to start, and a relatively simple way to figure out the rest. First up: the best place to start.

When you look at most CircuitPython examples, you'll see they begin with one or more `import` statements. These typically look like the following:

```
    import library_or_module
```

However, `import` statements can also sometimes look like the following:

```
    from library_or_module import name
    from library_or_module.subpackage import name
    from library_or_module import name as local_name
```

They can also have more complicated formats, such as including a `try / except` block, etc.

The important thing to know is that an `import` statement will always include the name of the module or library that you're importing.

Therefore, the best place to start is by reading through the `import` statements.

Here is an example import list for you to work with in this section. There is no setup or other code shown here, as the purpose of this section involves only the import list.

```
import time
import board
import neopixel
import adafruit_lis3dh
import usb_hid
from adafruit_hid.consumer_control import ConsumerControl
from adafruit_hid.consumer_control_code import ConsumerControlCode
```

Keep in mind, not all imported items are libraries. Some of them are almost always built-in CircuitPython modules. How do you know the difference? Time to visit the REPL.
In the Interacting with the REPL section on The REPL page in this guide, the `help("modules")` command is discussed. This command provides a list of all of the built-in modules available in CircuitPython for your board. So, if you connect to the serial console on your board, and enter the REPL, you can run `help("modules")` to see what modules are available for your board. Then, as you read through the `import` statements, you can, for the purposes of figuring out which libraries to load, ignore the statement that import modules.

The following is the list of modules built into CircuitPython for the Feather RP2040. Your list may look similar or be anything down to a significant subset of this list for smaller boards.

```python
>>> help("modules")
<_main_>            board
❶ bleio              builtins
❶ adafruit_bus_device  collections
❶ adafruit_pixelbuf  countio
❶ analogio          displayio
array               errc
ser2ttx             fontio
audiobusio          framebufferto
audiocore           gc
audioencoder        getpass
audioop3            imagecapture
audiopwmio          io
binascii            json
bitbangio           keypad
bitmaptools         math
bitmaps             microcontroller
https                sharphandle
micropython          storage
msgpack             struct
busio               neopixel
neopixel           supervisor
onewireio          os
os                  sys
paralleldisplay     terminal
pulsio             time
pwmio              touchio
qrc                   traceback
random               ublf
re                  usb.cdc
re                  usb.vid
rtificial          usb.product
rtificial          usb.product
rtificial          usb.vendor
rtificial          usb Product
rtc                  watchdog
```

Now that you know what you're looking for, it's time to read through the import statements. The first two, `time` and `board`, are on the modules list above, so they're built-in.

The next one, `neopixel`, is not on the module list. That means it's your first library! So, you would head over to the bundle zip you downloaded, and search for neopixel. There is a neopixel.mpy file in the bundle zip. Copy it over to the lib folder on your CI RCUITPY drive. The following one, `adafruit_lis3dh`, is also not on the module list. Follow the same process for adafruit_lis3dh, where you'll find adafruit_lis3dh.mpy, and copy that over.

The fifth one is `usb_hid`, and it is in the modules list, so it is built in. Often all of the built-in modules come first in the import list, but sometimes they don't! Don't assume that everything after the first library is also a library, and verify each import with the modules list to be sure. Otherwise, you'll search the bundle and come up empty!

The final two imports are not as clear. Remember, when `import` statements are formatted like this, the first thing after the `from` is the library name. In this case, the library name is `adafruit_hid`. A search of the bundle will find an adafruit_hid folder.
When a library is a folder, you must copy the entire folder and its contents as it is in the bundle to the lib folder on your CIRCUITPY drive. In this case, you would copy the entire adafruit_hid folder to your CIRCUITPY/lib folder.

Notice that there are two imports that begin with `adafruit_hid`. Sometimes you will need to import more than one thing from the same library. Regardless of how many times you import the same library, you only need to load the library by copying over the adafruit_hid folder once.

That is how you can use your example code to figure out what libraries to load on your CircuitPython-compatible board!

There are cases, however, where libraries require other libraries internally. The internally required library is called a dependency. In the event of library dependencies, the easiest way to figure out what other libraries are required is to connect to the serial console and follow along with the `ImportError` printed there. The following is a very simple example of an `ImportError`, but the concept is the same for any missing library.

**Example: `ImportError` Due to Missing Library**

If you choose to load libraries as you need them, or you're starting fresh with an existing example, you may end up with code that tries to use a library you haven't yet loaded. This section will demonstrate what happens when you try to utilise a library that you don't have loaded on your board, and cover the steps required to resolve the issue.

This demonstration will only return an error if you do not have the required library loaded into the lib folder on your CIRCUITPY drive.

Let's use a modified version of the Blink example.

```python
import board
import time
import simpleio

led = simpleio.DigitalOut(board.LED)

while True:
    led.value = True
    time.sleep(0.5)
    led.value = False
    time.sleep(0.5)
```
Save this file. Nothing happens to your board. Let's check the serial console to see what's going on.

![Serial Console Output]

You have an **ImportError**. It says there is *no module named 'simpleio'* that's the one you just included in your code!

Click the link above to download the correct bundle. Extract the lib folder from the downloaded bundle file. Scroll down to find simpleio.mpy. This is the library file you're looking for! Follow the steps above to load an individual library file.

The LED starts blinking again! Let's check the serial console.

![Serial Console Output]

No errors! Excellent. You've successfully resolved an **ImportError**!

If you run into this error in the future, follow along with the steps above and choose the library that matches the one you're missing.

**Library Install on Non-Express Boards**

If you have an M0 non-Express board such as Trinket M0, Gemma M0, QT Py M0, or one of the M0 Trinkeys, you'll want to follow the same steps in the example above to install libraries as you need them. Remember, you don't need to wait for an **ImportError** if you know what library you added to your code. Open the library bundle you downloaded, find the library you need, and drag it to the lib folder on your CIRCUITPY drive.

You can still end up running out of space on your M0 non-Express board even if you only load libraries as you need them. There are a number of steps you can use to try
to resolve this issue. You'll find suggestions on the [Troubleshooting page](https://adafruit.io/Den).

### Updating CircuitPython Libraries and Examples

Libraries and examples are updated from time to time, and it's important to update the files you have on your CIRCUITPY drive.

To update a single library or example, follow the same steps above. When you drag the library file to your lib folder, it will ask if you want to replace it. Say yes. That's it!

A new library bundle is released every time there's an update to a library. Updates include things like bug fixes and new features. It's important to check in every so often to see if the libraries you're using have been updated.

### Coding the LED Matrix Scoreboard

Once you've finished setting up your Feather M4 Express with CircuitPython, you can add these libraries to the lib folder:

- adafruit_bitmap_font
- adafruit_bus_device
- adafruit_display_shapes
- adafruit_display_text
- adafruit_imageload

Then, you can click on the Download: Project Zip link below at the top of the code to download the code, bitmap, font and .mp3 files.

```python
import time
import board
import audioio
import audiomp3
import framebufferio
import rgbmatrix
import displayio
import adafruit_imageload
import digitalio
from adafruit_display_shapes.rect import Rect
from adafruit_bitmap_font import bitmap_font
from adafruit_display_text import label

# matrix setup
displayio.release_displays()
matrix = rgbmatrix.RGBMatrix(
    width=64,
    height=32,
    mode=rgbmatrix.MODE_RGB_RGB...
```
bit_depth=4,
addr_pins=[board.A5, board.A4, board.A3, board.A2],
clock_pin=board.D13,
latch_pin=board.D0,
output_enable_pin=board.D1,
)
display = framebufferio.FramebufferDisplay(matrix)

# display groups
start_group = displayio.Group()
score_group = displayio.Group()

# text & bg color setup for scoreboard
score_text = "      
font = bitmap_font.load_font("/Fixedsys-32.bdf")
yellow = (255, 127, 0)
pink = 0xFF00FF

score_text = label.Label(font, text=score_text, color=0x0)
score_text.x = 23
score_text.y = 15

score_bg = Rect(0, 0, 64, 32, fill=yellow, outline=pink, stroke=3)

# start splash screen graphic
start, start_pal = adafruit_imageload.load("/pixelHoops.bmp",
    bitmap=displayio.Bitmap,
    palette=displayio.Palette)

start_grid = displayio.TileGrid(start, pixel_shader=start_pal,
    width=64, height=32)

# adding graphics to display groups
start_group.append(start_grid)
score_group.append(score_bg)
score_group.append(score_text)

# start by showing start splash
display.show(start_group)

# setup for break beam LED pin
break_beam = digitalio.DigitalInOut(board.A1)
break_beam.direction = digitalio.Direction.INPUT
break_beam.pull = digitalio.Pull.UP

# setup for button pin
button = digitalio.DigitalInOut(board.D4)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP

# setup for speaker pin
speaker = audioio.AudioOut(board.A0)

# mp3 decoder setup
file = "/hoopBloop0.mp3"
mp3stream = audiomp3.MP3Decoder(open(file, "rb"))

# state machines used in the loop
score = 0
hoops = False
button_state = False
beam_state = False
sample = 0

while True:
    # button debouncing
    if not button.value and not button_state:
        button_state = True
    # debouncing for break beam LED
if not break_beam.value and not beam_state:
    beam_state = True
# if a game hasn’t started and you press the button:
if not button.value and not hoops:
    # game starts
    hoops = True
    button_state = False
    # display shows scoreboard
display.show(score_group)
print("start game!")
time.sleep(0.5)
if hoops:
    # if the break beam LED detects a hoop:
    if not break_beam.value and beam_state:
        # score increase by 2 points
        score += 2
        # an mp3 plays
        file = "/hoopBloop{}\.mp3".format(sample)
        mp3stream.file = open(file, "rb")
        speaker.play(mp3stream)
        print("score!")
        # resets break beam
        beam_state = False
        # increases mp3 file count
        sample = (sample + 1) % 3
        # score text x pos if 4 digit score
        if score >= 1000:
            score_text.x = -1
        # score text x pos if 3 digit score
        elif score >= 100:
            score_text.x = 7
        # score text x pos if 2 digit score
        elif score >= 10:
            score_text.x = 16
        elif score >= 0:
            score_text.x = 23
        # updates score text to show current score
        score_text.text = score
        time.sleep(0.01)
# if a game is in progress and you press the button:
if not button.value and hoops:
    # game stops
    hoops = False
    button_state = False
    # score is reset to 0
    score = 0
    score_text.text = score
    # display shows the start splash graphic
display.show(start_group)
print("end game!")
time.sleep(0.5)

Your Feather M4 Express CIRCUITPY drive should look like this after you load the libraries, bitmap, font, .mp3 and code.py files:
LED Matrix Scoreboard Code Walkthrough

The code begins by importing the CircuitPython libraries.

```python
import time
import board
import audioio
import audiomp3
import framebufferio
import rgbmatrix
import displayio
import adafruit_imageload
import digitalio
from adafruit_display_shapes.rect import Rect
from adafruit_bitmap_font import bitmap_font
from adafruit_display_text import label
```

The RGB matrix is setup using the rgbmatrix library.

```python
# matrix setup
displayio.release_displays()
matrix = rgbmatrix.RGBMatrix(
    width=64,
    height=32,
    bit_depth=4,
    addr_pins=[board.A5, board.A4, board.A3, board.A2],
    clock_pin=board.D13,
    latch_pin=board.D0,
    output_enable_pin=board.D1,
)
display = framebufferio.FramebufferDisplay(matrix)
```

Then, the two display groups are created. There will be a group to show the start-up graphic and a second one to display the score when you're playing the game.
The text (score_text) and colorful background (score_bg) are setup next for the scoreboard. The text is using a bitmap font.

```
# text & bg color setup for scoreboard
score_text = "     ", font = bitmap_font.load_font("/FixedSys-32.bdf")
yellow = (255, 127, 0)
pink = 0xFF00FF
score_text = label.Label(font, text=score_text, color=0x0)
score_text.x = 23
score_text.y = 15
score_bg = Rect(0, 0, 64, 32, fill=yellow, outline=pink, stroke=3)
```

Following the score text setup, the start-up graphic bitmap is setup using the adafruit _imageload library.

```
# start splash screen graphic
start, start_pal = adafruit_imageload.load("/pixelHoops.bmp",
            bitmap=displayio.Bitmap,
            palette=displayio.Palette)
start_grid = displayio.TileGrid(start, pixel_shader=start_pal,
                                width=64, height=32)
```

The start-up bitmap and score text graphics are added to their respective display groups. The start-up bitmap is shown first using `display.show(start_group)`.

```
# adding graphics to display groups
start_group.append(start_grid)
score_group.append(score_bg)
score_group.append(score_text)

# start by showing start splash
display.show(start_group)
```

After the display is setup, the I/O pins are setup for the break beam LED, push button and speaker. The LED and button are setup as inputs and the speaker is setup as an AudioOut using the audioio library.

```
# setup for break beam LED pin
break_beam = digitalio.DigitalInOut(board.A1)
break_beam.direction = digitalio.Direction.INPUT
break_beam.pull = digitalio.Pull.UP

# setup for button pin
button = digitalio.DigitalInOut(board.D4)
button.direction = digitalio.Direction.INPUT
```
button.pull = digitalio.Pull.UP

# setup for speaker pin
speaker = audioio.AudioOut(board.A0)

The **MP3Decoder** is setup before the loop, similar to how display objects are setup early in the code. This helps to save memory when playing multiple .mp3 files. Notice how you aren't playing an .mp3 in this section, it's purely for setup.

```
# mp3 decoder setup
file = "/hoopBloop0.mp3"
mp3stream = audiomp3.MP3Decoder(open(file, "rb"))
```

The final portion of the setup creates that state machines that will be used in the loop. Their functions are commented below.

```
# state machines used in the loop
score = 0  # holds your score for the game
hoops = False  # tracks if a game is active or not
button_state = False  # button debouncing
beam_state = False  # break beam LED debouncing
sample = 0  # file count for the mp3 files
```

The loop begins with debouncing setup for the button and break beam LED. This eliminates any glitches in receiving data from their inputs.

```
while True:
    # button debouncing
    if not button.value and not button_state:
        button_state = True
    # debouncing for break beam LED
    if not break_beam.value and not beam_state:
        beam_state = True
```

The button has a different function depending on whether or not a game is in progress. If a game has not started, then pressing the button begins a new game and also changes the display to show the scoreboard graphic.

```
# if a game hasn't started and you press the button:
if not button.value and not hoops:
    # game starts
    hoops = True
    button_state = False
    # display shows scoreboard
display.show(score_group)
    print("start game!")
time.sleep(0.5)
```

A game runs if the **hoops** state is **True**. Every time the break beam LED detects a break, your score increases by two points and an .mp3 file plays.
There are three different .mp3 files and they play in order as the game goes on. The code is able to iterate through the files easily because they are all named hoopBloop*. By naming them like this, you can use "/ hoopBloop{}.mp3.format(sample)", to iterate through them with sample holding the number.

The code is referencing the earlier mp3stream that was setup before the loop to decode and play the .mp3 files.

Additionally, there is some text formatting that takes place depending on how many digits your score is. This allows for the score to remain centered during game play.

```python
if hoops:
    # if the break beam LED detects a hoop:
    if not break_beam.value and beam_state:
        # score increase by 2 points
        score += 2
        # an mp3 plays
        file = "/hoopBloop{}.mp3.format(sample)
        mp3stream.file = open(file, "rb")
        speaker.play(mp3stream)
        print("score!")
        # resets break beam
        beam_state = False
        # increases mp3 file count
        # plays the 3 files in order
        sample = (sample + 1) % 3
    # score text x pos if 4 digit score
    if score &gt;= 1000:
        score_text.x = -1
    # score text x pos if 3 digit score
    elif score &gt;= 100:
        score_text.x = 7
    # score text x pos if 2 digit score
    elif score &gt;= 10:
        score_text.x = 16
    # score text x pos if 1 digit score
    elif score &gt;= 0:
        score_text.x = 23
    # updates score text to show current score
    score_text.text = score
    time.sleep(0.1)
```

If a game is in progress and you press the button, it changes the hoops state to False and ends the game. The score is also reset to 0 and the display shows the start-up bitmap.

```python
# if a game is in progress and you press the button:
if not button.value and hoops:
    # game stops
    hoops = False
    button_state = False
    # score is reset to 0
    score = 0
    # display shows the start splash graphic
    display.show(start_group)
```
print("end game!")
time.sleep(0.5)

Hoop Assembly

Wiring IR breakbeam receiver
Use a 2-wire ribbon cable and measure to 40cm in length. This wire is long in order to wrap around the 3D printed hoop. The ground and voltage wires from the IR sensor are extended by slicing them with the 2-wire ribbon cable. Use heat shrink tubing to insulate exposed wire.

Wiring IR breakbeam transmitter
A 3-wire JST cable is soldered to the three wires on the IR transmitter. A JST connector allows the sensor to be easily connected and disconnected. Use heat shrink tubing to insulate exposed wire.

IR breakbeam Hoop Wiring
Place the IR breakbeam receiver inside the hoop and gauge the length of wire. It should be long enough to wrap around half of the hoop with extra slack at the end for connecting to a 2-wire JST connector.
Secure IR breakbeam receiver
Use an M2.5 x 10mm long screw to secure the IR receiver to the hoop. Press the IR sensor into the recess on the end of the hoop. Hold in place while fastening M2.5 screw. The LED should be orientated with the mounting tabs facing up.

Thread Wiring
The 3D printed hoop features a channel for housing the wiring. This keeps the ribbon cable from coming out during gameplay. Insert and thread the wiring through one of the channels. Doesn't matter which side as they're both symmetrical.

Hole Relief for Wiring
The channels have an opening nearly half way through the hoop to allow access to the wiring. This can help provide a hold on the wiring to further thread the wiring half-way around the hoop.
Exit Hole
Continue to thread the wiring from the IR receiver until it's poking through the other end. Carefully insert the wiring through the hole closest to the channel exit.

Installed IR receiver
Double check the wiring has been pulled all the way through. The wiring should wrap around half-way through the hoop. The IR LED should be orientated with the mounting tabs pointing up.

Solder JST cable
A 2-wire JST female cable is soldered to the wiring on the IR receiver. Double check the polarity is matching with the ground connecting to the black wire and voltage to the red wire. Use heat shrink tubing to insulate the exposed wire.
Soldered JST cable
The hoop now has the IR receiver installed and wired up.

Secure IR Transmitter
The IR transmitter is installed to the opposite end of the hoop near the mounting tabs. Place and insert the IR transmitter into the recess in the middle. Hold in place while fastening M2.5 x 10mm long screw.

Wiring Feather

Matrix Power Cable
The 64x32 RGB matrix display features a power cable. The wiring has been shortened and simplified to make the wiring more manageable during assembly.
Feather & FeatherWing Setup
The Feather M4 Express has a 12-pin and 16-pin strip of male header pins soldered to the bottom of the PCB. The RGB Matrix FeatherWing has 12-pin and 16-pin female headers soldered to the top of the PCB. The 2x10 female socket soldered is to the bottom of the FeatherWing.

JST Cables
Four JST cables are used to connect the various components to the Feather and FeatherWing. Here's a list of the cables and where they're connected to.

- 2-wire male JST: Arcade Button
- 3-wire male JST: STEMMA Speaker
- 3-wire male JST: IR transmitter
- 2-wire male JST: IR receiver

Feather & FeatherWing Connections
Double check the wires and cables are ready for soldering to the PCBs.
Solder Cable – IR breakbeam receiver
Begin by soldering the 2-wire JST cable to the ground and 3V pins on the Feather. This cable will connect to the IR receiver. The wiring is soldered to the bottom of the Feather.

Solder Cable – IR breakbeam TX
Solder the 3-wire JST cable to available ground and 3V pins on the Feather. This cable will connect to the IR transmitter. The white wire will be connected to the A1 pin on the RGB Matrix FeatherWing.

FeatherWing Connections
The remaining JST cables will be soldered to the RGB Matrix FeatherWing.
**Button Wiring**
Soldered the second 2-wire JST cable to the ground and D4 pins on the bottom of the FeatherWing. Reference the pins on the Feather to ensure the pins are correct.

**Solder Signal – IR transmitter**
Solder the white wire from the 3-wire JST cable to the A1 pin on the bottom of the FeatherWing. This cable will be used to connect the IR transmitter.

**Wiring STEMMA Speaker**
Solder the wires from the second 3-wire JST cable to the following pins on the FeatherWing. White wire to A0, black wire to ground and red wire to 3V.
**Wired Feather & FeatherWing**
The wiring is complete and ready for plugging in the various components.

**Wire Check**
Double check the wiring is correct.

---

**Matrix Assembly**

**Install Corner Brackets**
Corner brackets are secured to the four corners on the RGB matrix display. These brackets are secured to the mounting holes. They also fit a sheet of black LED acrylic.
Insert Hex Nuts
Grab two of the four corner brackets and insert M3 hex nuts into the recesses. The two corner brackets are used to secure the button mount to the side of the display. Choose which side you'd like to have the button mounted.

Fitting Corner Brackets
Place the brackets over the corners and orient them so the mounting holes are properly lined up. The side of display should have the brackets with visible mounting holes. These holes are used for securing the button mounting plate.

Optional: Insert the sheet of Black LED Acrylic by sliding into the corner brackets.

Installing Feet
Place the foot over the corner brackets and line up the mounting holes. Reference the photo for correct placement.
Secure Feet
Use two M3 x 10mm long screw to secure the feet to the corner brackets. The screws will also fasten into the threaded inserts that are on the display.

Installed Feet
Repeat this process for the second foot.

Button Wiring
A short 2-wire JST cable is soldered to the arcade button. This is connected to the JST cable that is soldered to the D4 pin on the FeatherWing.
Install Button Mounting Plate
Place the button mounting plate over the corner brackets with the hex nuts you installed earlier. Line up the mounting holes.

Secure Button Mounting Plate
Insert and fasten two M3 x 6mm long machine screws through the mounting holes. Reference the photo for best placement.

Panel Mount Arcade Button
Insert the arcade button to the button mounting plate. Make sure the button is facing in the correct orientation.
Secure Arcade Button
Use the screw-on sleeve to secure the arcade button to the button mounting plate.

Install Hoop Brackets to Feet
Place the two hoop brackets onto the feet. Line up the mounting holes. Reference the photo for correct placement and orientation.

Secure Hoop Brackets
Insert and fasten an M3 x 10mm long screw through the mounting holes on the hoop brackets.
Secured Brackets
Use an M3 hex nut to secure the screws. Insert the M3 hex nut into the recess on the back of the foot.

Second Hoop Bracket
Repeat this process for the other foot.

Install Hoop
Place the mounting tabs from the hoop into the two brackets on the feet. The brackets feature walls for press fitting the mounting tabs. The mounting holes should be lined up properly.
Secure Hoop
Insert and fasten M3 x 10mm long screws.

Secured Hoop
The mounting tabs on the hoop feature recesses for fitting M3 hex nuts. Insert the hex nuts to secure the M3 screws.

Installing Feather & FeatherWing
Get the PCBs ready to fit onto the back of the RGB matrix display.
Installed Feather & FeatherWing
The Feather M4 express is fitted on top of the RGB matrix FeatherWing. The FeatherWing is press fitted on top of the 2x8 IDC header on-board the display. Make sure it is placed on the correct header. Use the arrows to reference the input socket.

STEMMA Speaker
The Stemma speaker is secured to the speaker mount using 4x M2.5 x 10mm long machine screws.

Mount Stemma Speaker
Place the Stemma speaker over the built-in stand-offs on the speaker mount. Insert M2.5 screws and use hex nuts to secure the PCB to the mount.
Secure Speaker Mount
Place the speaker mount over the mounting holes in the middle on the back of the display. Line up the mounting holes and insert two M3 x 6mm long screws to secure the speaker mount to the display. Reference the photo for best orientation.

Plug In Connections
Carefully make connections by plugging in JST cables to their respective counterparts. Take your take and make sure the connections are correct.

Final Build
And there we have it! The build is ready for testing. We suggest using mounting tack to temporarily secure the feet to a table top or work surface. The feet feature various mounting holes for securing to a frame or table top with hardware screws.

Take caution and be careful during gameplay. You are throwing a ball at a display!