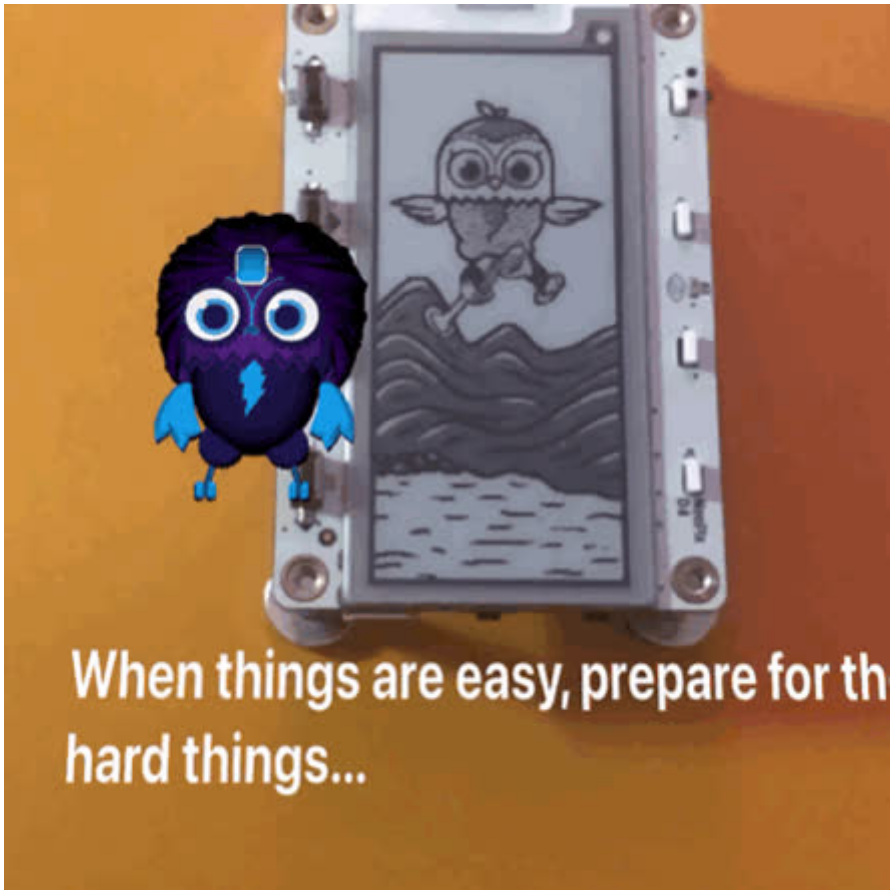
MagTag AR Tarot Card Reading

Created by Trevor Beaton



https://learn.adafruit.com/magtag-tarot-cards

Last updated on 2024-03-08 03:52:57 PM EST

Table of Contents

Overview

This guide will show you how to display our Adafruit Tarot Cards as a looping slideshow after you've setup your MagTag.

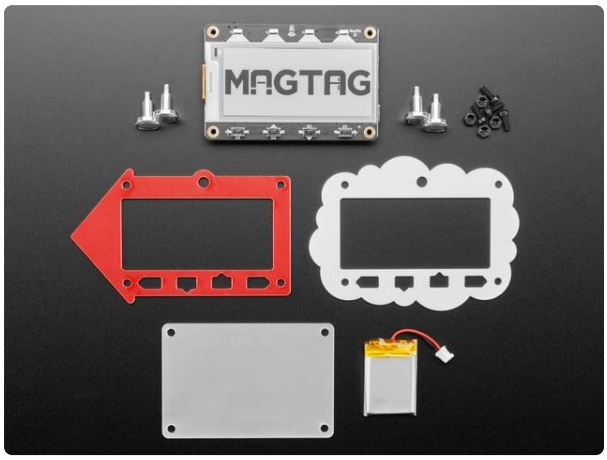
Want a cherry on top? With our new Fortune Teller feature on the Adafruit AR app, you can scan your Adafruit Tarot Cards on the MagTag, and Minerva will read your card in Augmented reality!

What you'll need:

- Adafruit MagTag - 2.9" Grayscale E-Ink WiFi Display (http://adafru.it/4800)
- An **iPhone or iPad** running **iOS 13** or greater.
- Adafruit AR (https://adafru.it/Ptb)
- Mu-Editor (https://adafru.it/19AC)

Please note - the Adafruit AR app is currently only available for iOS.

Parts



Adafruit MagTag Starter Kit - ADABOX017 Essentials

The Adafruit MagTag combines the new ESP32-S2 wireless module and a 2.9" grayscale E-Ink display to make a low-power IoT display that can show data on its screen...

https://www.adafruit.com/product/4819

Adafruit MagTag - 2.9" Grayscale E-Ink WiFi Display

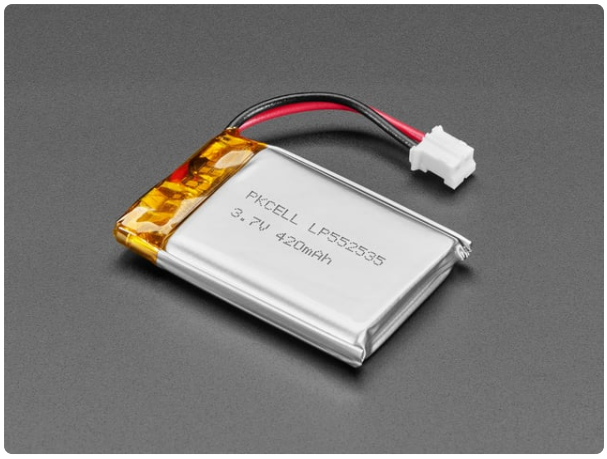
The Adafruit MagTag combines the new ESP32-S2 wireless module and a 2.9" grayscale E-Ink display to make a low-power IoT display that can show data on its screen even when power...

https://www.adafruit.com/product/4800

Mini Magnet Feet for RGB LED Matrices (Pack of 4)

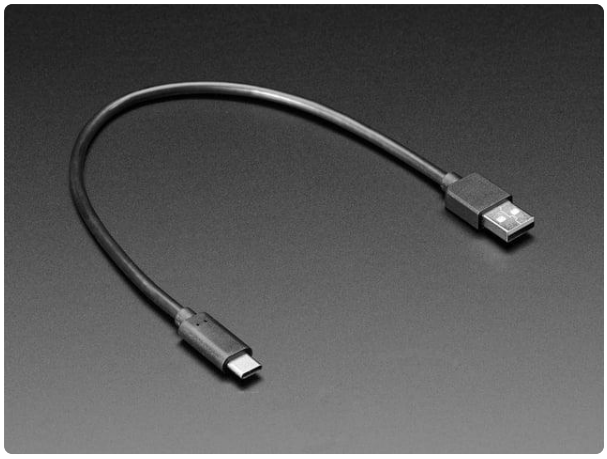
Got a glorious RGB Matrix project you want to mount and display in your workspace or home? If you have one of the matrix panels listed below, you'll need a pack of these...

https://www.adafruit.com/product/4631



Lithium Ion Polymer Battery with Short Cable - 3.7V 420mAh

Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/4236



USB Type A to Type C Cable - 1ft - 0.3 meter

As technology changes and adapts, so does Adafruit. This USB Type A to Type C cable will help you with the transition to USB C, even if you're still...
https://www.adafruit.com/product/4473

This project builds off of John Park's Magtag Slideshow guide (https://adafru.it/P9F). The steps on the following pages will show you how to set it up, then add the tarot cards.

Install CircuitPython

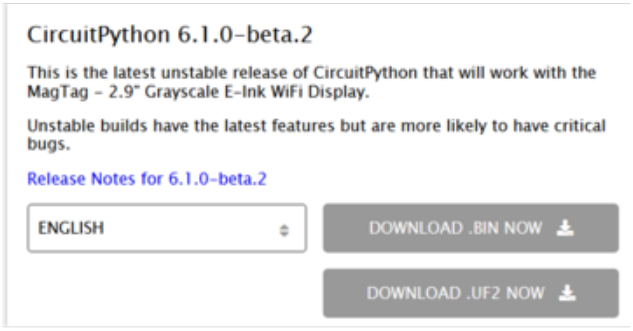
CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files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

Set Up CircuitPython

Follow the steps to get CircuitPython installed on your MagTag.

Download the latest CircuitPython for your board from circuitpython.org

https://adafru.it/OBd



Click the link above and download the latest .BIN and .UF2 file

(depending on how you program the ESP32S2 board you may need one or the other, might as well get both)

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

Plug your MagTag 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.

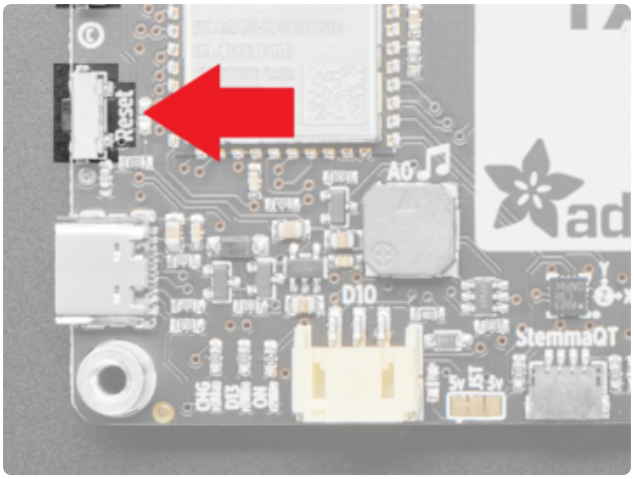
Option 1 - Load with UF2 Bootloader

This is by far the easiest way to load CircuitPython. **However it requires your board has the UF2 bootloader installed. Some early boards do not (we hadn't written UF2 yet!) - in which case you can load using the built in ROM bootloader.**

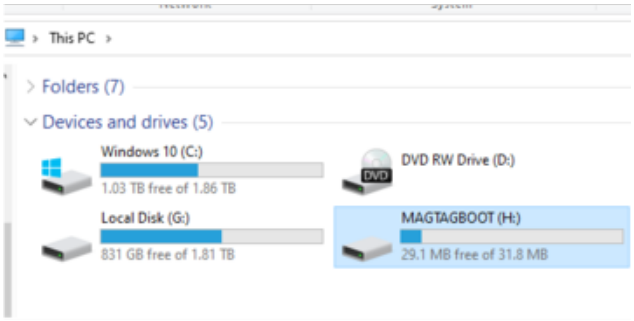
Still, try this first!

Try Launching UF2 Bootloader

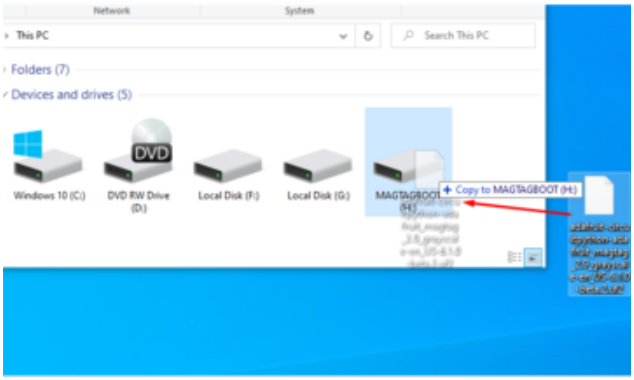
Loading CircuitPython by drag-n-drop UF2 bootloader is the easier way and we recommend it. If you have a MagTag where the front of the board is black, your MagTag came with UF2 already on it.



Launch UF2 by **double-clicking** the Reset button (the one next to the USB C port). You may have to try a few times to get the timing right.

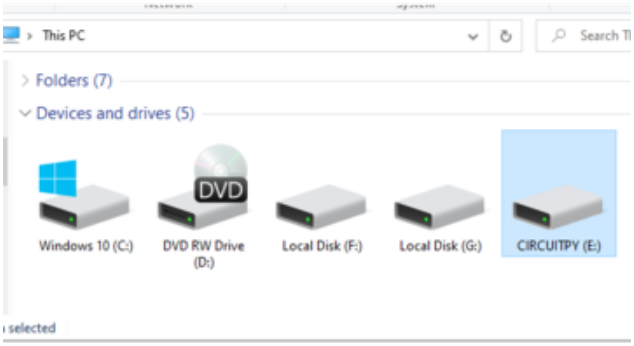


If the UF2 bootloader is installed, you will see a new disk drive appear called **MAGTAGBOOT**



Copy the **UF2** file you downloaded at the first step of this tutorial onto the **MAGTAGBOOT** drive

If you're using Windows and you get an error at the end of the file copy that says **Error from the file copy, Error 0x800701B1: A device which does not exist was specified.** You can ignore this error, the bootloader sometimes disconnects without telling Windows, the install completed just fine and you can continue. If its really annoying, you can also upgrade the bootloader (the latest version of the UF2 bootloader fixes this warning) (https://adafru.it/Pfk)

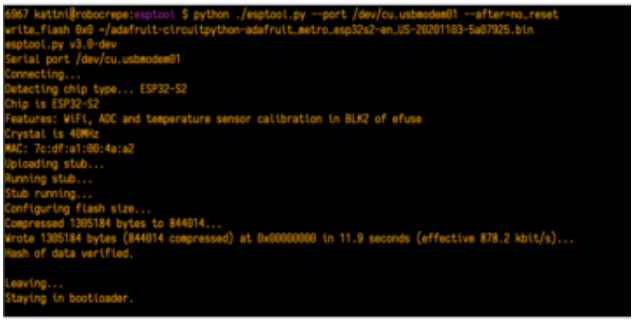


Your board should auto-reset into CircuitPython, or you may need to press reset. A **CIRCUITPY** drive will appear. You're done! Go to the next pages.

Option 2 - Use esptool to load BIN file

If you have an original MagTag with while soldermask on the front, we didn't have UF2 written for the ESP32S2 yet so it will not come with the UF2 bootloader.

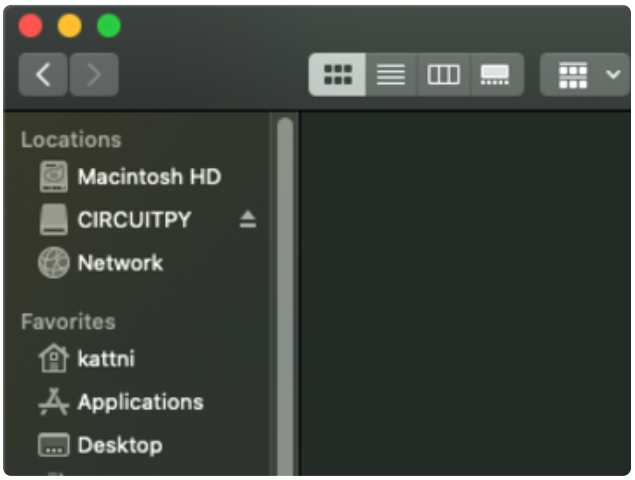
You can upload with **esptool** to the ROM (hardware) bootloader instead!



Follow the initial steps found in the Run esptool and check connection section of the ROM Bootloader page (https://adafru.it/OBc) to verify your environment is set up, your board is successfully connected, and which port it's using.

In the final command to write a binary file to the board, replace the port with your port, and replace "firmware.bin" with the the file you downloaded above.

The output should look something like the output in the image.



Press reset to exit the bootloader.

Your **CIRCUITPY** drive should appear!

You're all set! Go to the next pages.

Option 3 - Use Chrome Browser To Upload BIN file

If for some reason you cannot get esptool to run, you can always try using the Chrome-browser version of esptool we have written. This is handy if you don't have Python on your computer, or something is really weird with your setup that makes esptool not run (which happens sometimes and isn't worth debugging!) You can follow along on the Web Serial ESPTool (https://adafru.it/Pdq) page and either load the UF2 bootloader and then come back to Option 1 on this page, or you can download the CircuitPython BIN file directly using the tool in the same manner as the bootloader.

Code the MagTag Slideshow Viewer

First, make sure you've installed the fundamental MagTag libraries as shown on the previous page.

Text Editor

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in this guide (https://adafru.it/ANO).

Alternatively, you can use any text editor that saves simple text files.

Code

To use with CircuitPython, you need to first install a few libraries, into the lib folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, open the directory **Magtag_Slideshow/** and then click on the directory that matches the version of CircuitPython you're using and copy the contents of that directory to your **CIRCUITPY** drive.

Your **CIRCUITPY** drive should now look similar to the following image:

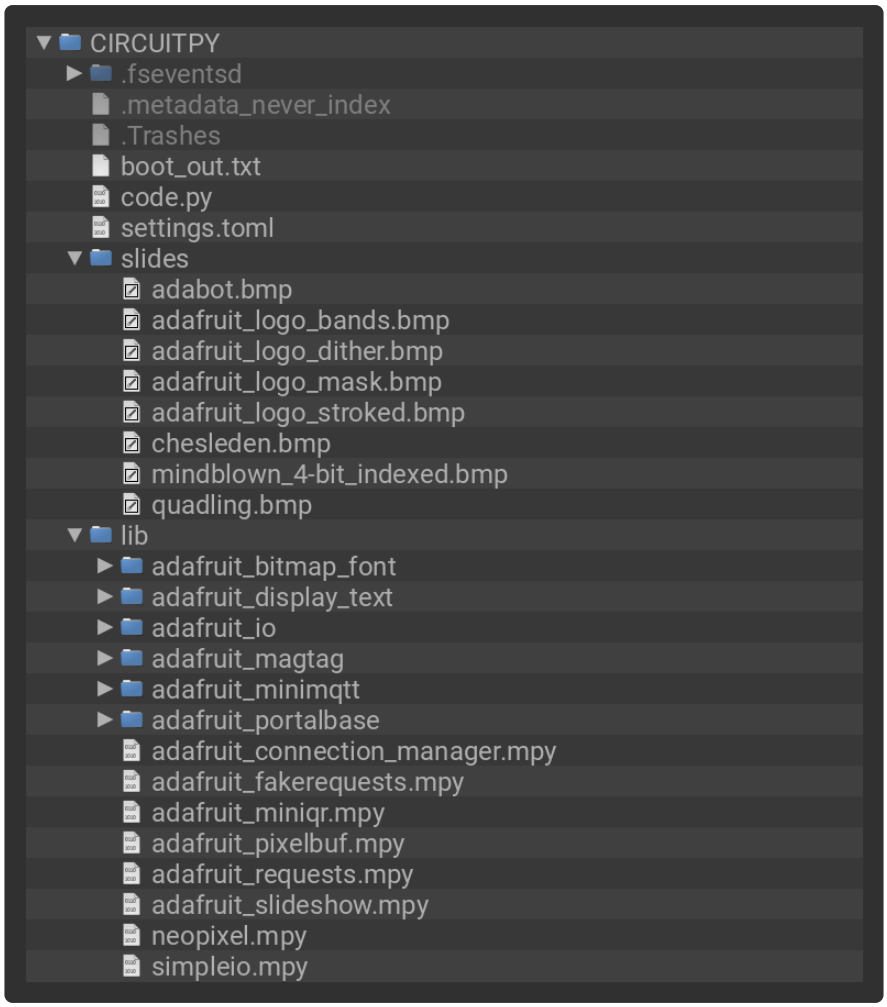


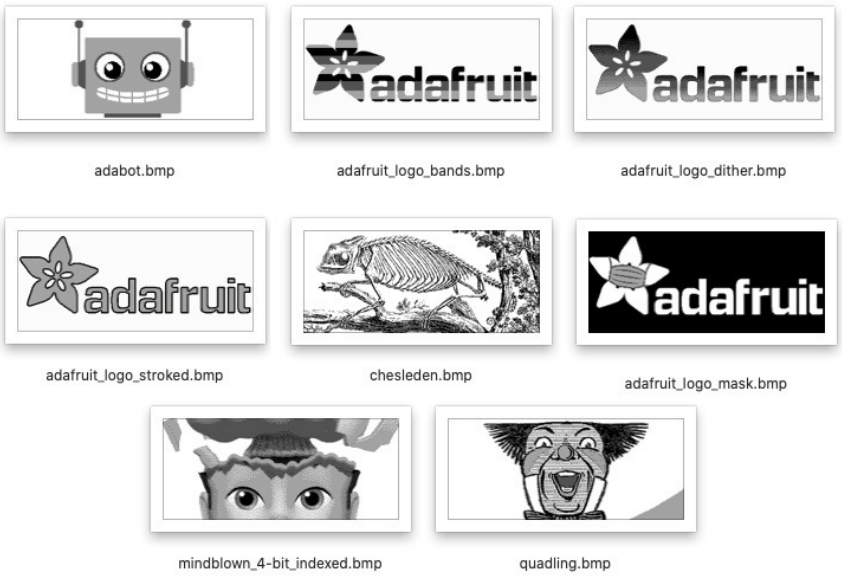
Image Prep

You can use the images included with the .zip download as a guide for making new images -- and we've got a whole guide on preparing graphics for e-ink displays called, wait for it... Preparing Graphics for E-Ink Displays (https://adafru.it/OFT)!

The recommended specifications for the MagTag Slideshow* images are as follows:

- **.bmp** file format
- 4-bit indexed (these are nice and small, and it's a four-shades-of-grey display, so these look just as good as any higher bit depth)
- 296 x 128 pixels

*note, there are different methods in CircuitPython for displaying images on e-ink displays, so for other projects you may need to use different bit-depths.




```
# SPDX-FileCopyrightText: 2020 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Magtag Slideshow
# auto plays .bmp images in /slides folder
# press left and right buttons to go back or forward one slide
# press down button to toggle autoplay mode
# press up button to toggle sound

import time
import terminalio
from adafruit_magtag.magtag import MagTag
from adafruit_slideshow import PlayBackOrder, SlideShow, PlayBackDirection

magtag = MagTag()


def blink(color, duration):
    magtag.peripherals.neopixel_disable = False
    magtag.peripherals.neopixels.fill(color)
    time.sleep(duration)
    magtag.peripherals.neopixel_disable = True


RED = 0x880000
GREEN = 0x008800
BLUE = 0x000088
YELLOW = 0x884400
CYAN = 0x0088BB
MAGENTA = 0x9900BB
WHITE = 0x888888

blink(WHITE, 0.3)

# pylint: disable=no-member
magtag.add_text(
    text_font=terminalio.FONT,
    text_position=(
        5,
        (magtag.graphics.display.height // 2) - 1,
    ),
    text_scale=3,
```

```
)
magtag.set_text("MagTag Slideshow")
time.sleep(5)
magtag.add_text(
    text_font=terminalio.FONT,
    text_position=(3, 120),
    text_scale=1,
)
magtag.set_text("  back        mute      pause/play     fwd", 1)
time.sleep(8)

timestamp = time.monotonic()
sound_toggle = True  # state of sound feedback
autoplay_toggle = True  # state of autoplay
auto_pause = 60  # time between slides in auto mode

# Create the slideshow object that plays through alphabetically.
slideshow = SlideShow(
    magtag.graphics.display,
    None,
    auto_advance=autoplay_toggle,
    folder="/slides",
    loop=True,
    order=PlayBackOrder.ALPHABETICAL,
    dwell=auto_pause,
)

while True:
    slideshow.update()
    if magtag.peripherals.button_a_pressed:
        if sound_toggle:
            magtag.peripherals.play_tone(220, 0.15)
        blink(YELLOW, 0.4)
        slideshow.direction = PlayBackDirection.BACKWARD
        time.sleep(5)
        slideshow.advance()

    if magtag.peripherals.button_b_pressed:
        if not sound_toggle:
            magtag.peripherals.play_tone(660, 0.15)
            blink(CYAN, 0.4)
        else:
            blink(MAGENTA, 0.4)
        sound_toggle = not sound_toggle

    if magtag.peripherals.button_c_pressed:
        if not autoplay_toggle:
            if sound_toggle:
                magtag.peripherals.play_tone(440, 0.15)
            blink(GREEN, 0.4)
            autoplay_toggle = True
            slideshow.direction = PlayBackDirection.FORWARD
            slideshow.auto_advance = True
        else:
            if sound_toggle:
                magtag.peripherals.play_tone(110, 0.15)
            blink(RED, 0.4)
            autoplay_toggle = False
            slideshow.auto_advance = False

    if magtag.peripherals.button_d_pressed:
        if sound_toggle:
            magtag.peripherals.play_tone(880, 0.15)
        blink(BLUE, 0.4)
        slideshow.direction = PlayBackDirection.FORWARD
        time.sleep(5)
        slideshow.advance()

    time.sleep(0.01)
```

How It Works

Libraries

We import libraries to take care of the heavy lifting, in this case:

```
import time
import terminalio
from adafruit_magtag.magtag import MagTag
from adafruit_slideshow import PlayBackOrder, SlideShow, PlayBackDirection
```

The `time` library allows us to do some pausing between steps.

In order to easily display a title and some labels, the `terminalio` library gives us a typeface to use.

The `adafruit_magtag` library makes it very simple to set up the MagTag's display, use the buttons, NeoPixels, and speaker.

The `adafruit_slideshow` library similarly is a convenience library that makes it a snap to run a slideshow on the device!

Setup

There are a number of setup steps we'll take next. First, we create the `MagTag()` object named `magtag` (all lower-case is easier to type anyway!).

Next, we'll create a function called `blink()` in order to solve the Riemann Hypothesis (https://adafru.it/OFP). OK, OK, just kidding; it'll be used to blink the NeoPixels.

```
def blink(color, duration):
    magtag.peripherals.neopixel_disable = False
    magtag.peripherals.neopixels.fill(color)
    time.sleep(duration)
    magtag.peripherals.neopixel_disable = True
```

Notice how there's no need for a separate NeoPixel setup, we can simply call them with the `magtag.peripherals.neopixels` commands.

The function has two arguments -- `color` and `duration` -- which make it simple to blink a particular color for a particular time period.

Next, we'll define some color variable names.

```
RED = 0x880000
GREEN = 0x008800
BLUE = 0x000088
YELLOW = 0x884400
CYAN = 0x0088BB
MAGENTA = 0x9900BB
WHITE = 0x888888
```

And then we will flash the lights white upon startup:

`blink(WHITE, 0.3)`

Text

Next up, we'll use the magtag library's text commands to put a title and some instructional labels on screen.

```
magtag.add_text(
    text_font=terminalio.FONT,
    text_position=(
        5,
        (magtag.graphics.display.height // 2) - 1,
    ),
    text_scale=3,
)
magtag.set_text("MagTag Slideshow")
time.sleep(5)
magtag.add_text(
    text_font=terminalio.FONT,
    text_position=(3, 120),
    text_scale=1,
)
magtag.set_text("  back        mute      pause/play     fwd", 1)
```

Note that each instance of text can be referred to by index number, so the first one is index 0, and the second is index 1, which appears as the second argument in this line:

`magtag.set_text(" back mute pause/play fwd", 1)`

Variables

We set a few variables that will used to track states and timing.

```
sound_toggle = True  # state of sound feedback
autoplay_toggle = True  # state of autoplay
auto_pause = 60  # time between slides in auto mode
```

Slideshow Setup

The final bit of setup is to create the `slideshow` object:

```
slideshow = SlideShow(
    magtag.graphics.display,
    backlight_pwm=None,
    auto_advance=autoplay_toggle,
    folder="/slides",
    loop=True,
    order=PlayBackOrder.ALPHABETICAL,
    dwell=auto_pause,
)
```

This sets up the slideshow to:

- use the MagTag display
- there is no backlight! so don't try
- sets the auto advance on (initially)
- specifies the folder for bitmaps
- sets looping on, the playback order is alphabetical (`RANDOM` is the other option)
- the dwell time between slides is set to 60 seconds

Main Loop

In the main loop of the program, the `slideshow.update()` line is all that's needed to run the slideshow with the initial settings! Everything else in the code here is for using the buttons.

```
if magtag.peripherals.button_a_pressed:
        if sound_toggle:
            magtag.peripherals.play_tone(220, 0.15)
        blink(YELLOW, 0.4)
        slideshow.direction = PlayBackDirection.BACKWARD
        time.sleep(5)
        slideshow.advance()

    if magtag.peripherals.button_b_pressed:
        if not sound_toggle:
            magtag.peripherals.play_tone(660, 0.15)
            blink(CYAN, 0.4)
        else:
            blink(MAGENTA, 0.4)
        sound_toggle = not sound_toggle

    if magtag.peripherals.button_c_pressed:
        if not autoplay_toggle:
            if sound_toggle:
                magtag.peripherals.play_tone(440, 0.15)
            blink(GREEN, 0.4)
            autoplay_toggle = True
            slideshow.direction = PlayBackDirection.FORWARD
            slideshow.auto_advance = True
        else:
            if sound_toggle:
                magtag.peripherals.play_tone(110, 0.15)
            blink(RED, 0.4)
            autoplay_toggle = False
            slideshow.auto_advance = False
```

```
    if magtag.peripherals.button_d_pressed:
        if sound_toggle:
            magtag.peripherals.play_tone(880, 0.15)
        blink(BLUE, 0.4)
        slideshow.direction = PlayBackDirection.FORWARD
        time.sleep(5)
        slideshow.advance()
```

The A button and D button are used for reversing and advancing the slideshow by one slide, respectively. You can see the we play a quick beep sound if sound is enabled. The NeoPixels are also blinked a unique color depending on the direction.

The B button toggles the sound on or off each time it is pressed.

And, finally, the C button will toggle the auto advance state on and off.

Load up some images and give it all a try! When the player starts it will automatically advance and loop the slides.

Press the C button to pause playback, press it again to resume. You'll see the NeoPixels blink red for "pause" and "green" for resume.

Want to advance to the next slide, just press the D button. You can go back one slide by pressing the A button.

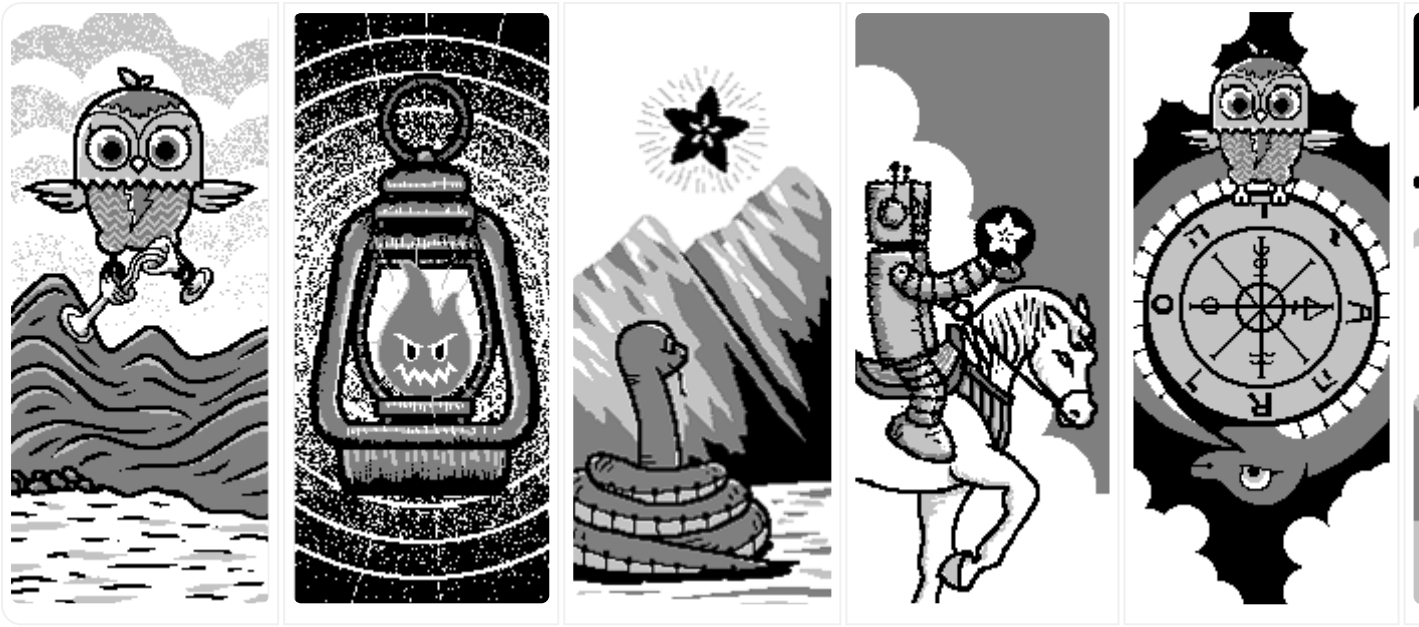
Oh, and lastly, if you don't want to hear beeps when you press the buttons, you can toggle sound by pressing the B button.

Using Adafruit AR

Once you've got your slideshow working, click the button below to download the tarot card image files by clicking the button below. Once downloaded, unzip the files and copy them to your **CIRCUITPY/Slideshow** folder.

Tarot Cards

https://adafru.it/Ptc



When adding graphics to the MagTag, remember that the MagTag can only display .bmp files.

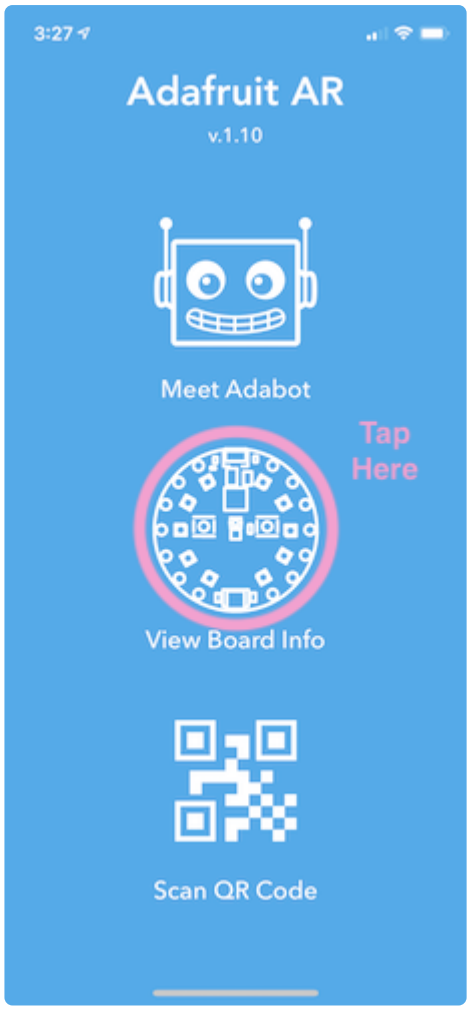
If you need simpler code to get your slideshow going you can copy this block of code to mu-editor. Save the code to the MagTag **CIRCUITPY** drive as **code.py**:

```
from adafruit_magtag.magtag import MagTag
from adafruit_slideshow import SlideShow

magtag = MagTag()

slideshow = SlideShow(
    magtag.graphics.display,
    None,
    auto_advance = True,
    folder="/SlideShow",
    dwell= 40
)

while slideshow.update():
    pass
```

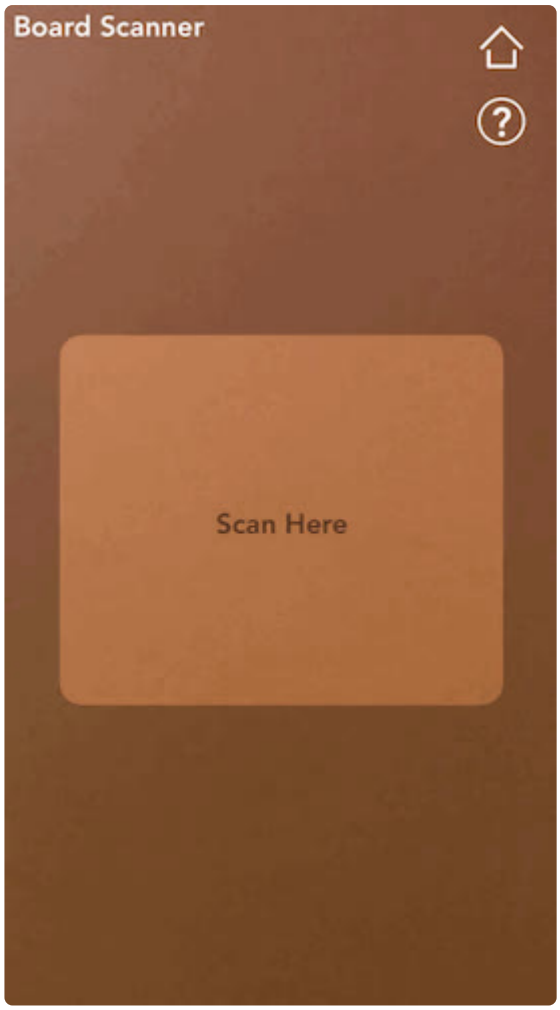
This code allows your MagTag to display a new image from the slideshow folder every 40 seconds (depending on your duration choice).

After you've tested your MagTag slideshow, let's download Adafruit AR.

If you haven't already, download Adafruit AR on the Apple App Store (https://adafru.it/Ptb). Once you've passed the onboarding pages, select the View Board Info.



Once you're there, scan your tarot cards. Minerva will pop up and read your tarot card.


Board Scanner
Scan Here