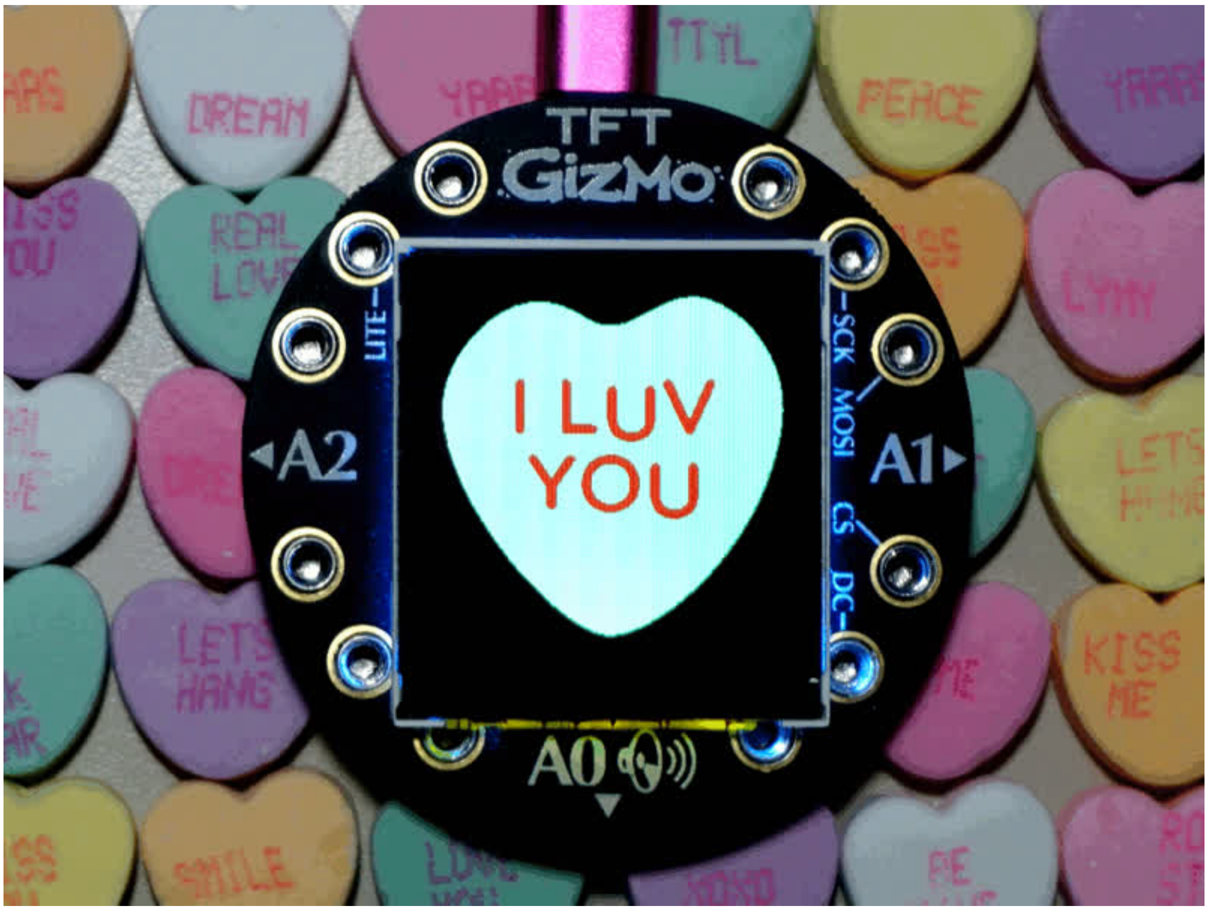
CircuitPython TFT Candy Hearts

Created by Carter Nelson



https://learn.adafruit.com/circuit-python-tft-gizmo-candy-hearts

Last updated on 2024-06-03 03:02:53 PM EDT

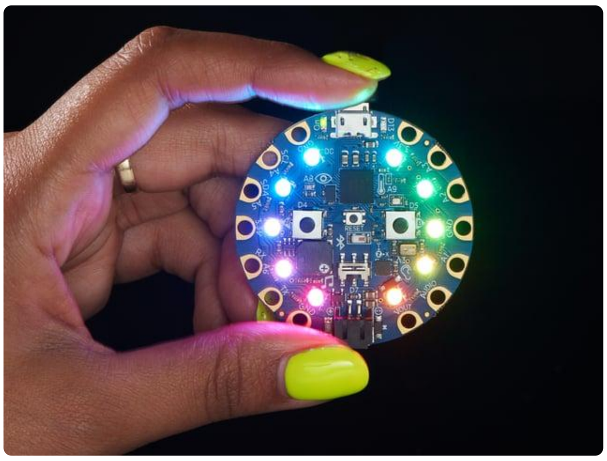
Table of Contents

Overview

This guide turns a TFT Gizmo (http://adafru.it/4367) into a conversation heart, just like those chalky candies that show up around Valentines. The code is written in CircuitPython (https://adafru.it/cpy-welcome) and can run on the Circuit Playground Bluefruit (http://adafru.it/4333). If you bought an AdaBox014 (https://adafru.it/IUd), then you have everything you need. Or you can get the items from the links below.

Two versions of the code are provided. The first is a simple one that will let you set predefined messages and colors and then cycle through them by pressing the buttons on the Circuit Playground. The second one uses the Bluetooth Low Energy (BLE) feature of the Circuit Playground along with the Adafruit Bluefruit LE Connect app (https://adafru.it/DNc) to allow you to set message and colors from your BLE enabled smart phone or tablet.

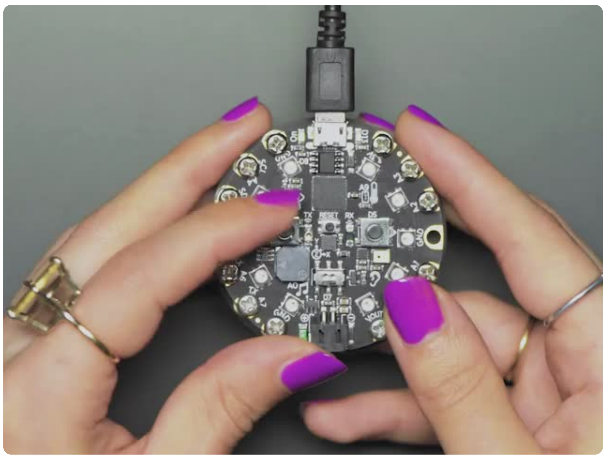
Parts



Circuit Playground Bluefruit - Bluetooth Low Energy

Circuit Playground Bluefruit is our third board in the Circuit Playground series, another step towards a perfect introduction to electronics and programming. We've...

https://www.adafruit.com/product/4333



Circuit Playground TFT Gizmo - Bolt-on Display + Audio Amplifier
Extend and expand your Circuit Playground projects with a bolt on TFT Gizmo that lets you add a lovely color display in a sturdy and reliable fashion. This PCB looks just like a round...
https://www.adafruit.com/product/4367

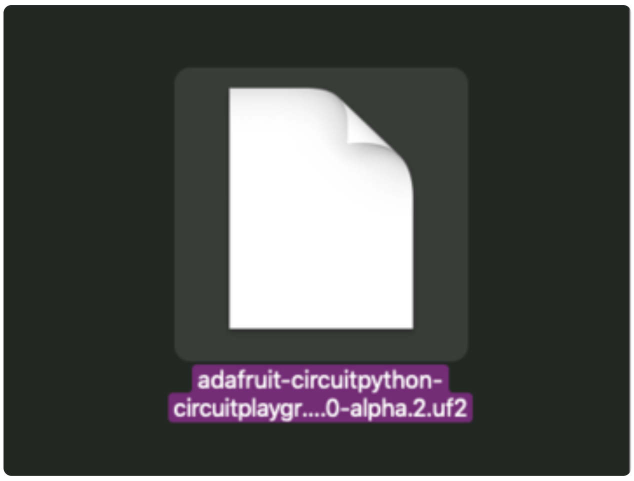
Adafruit Circuit Playground Bluefruit Express Starter Kit
If you missed out on ADABOX 014, its not too late for you to pick up the parts necessary to build many of the projects! This kit pack doesn't come with tissue paper or the nifty...
https://www.adafruit.com/product/4504

CircuitPython on Circuit Playground Bluefruit

Install or Update CircuitPython

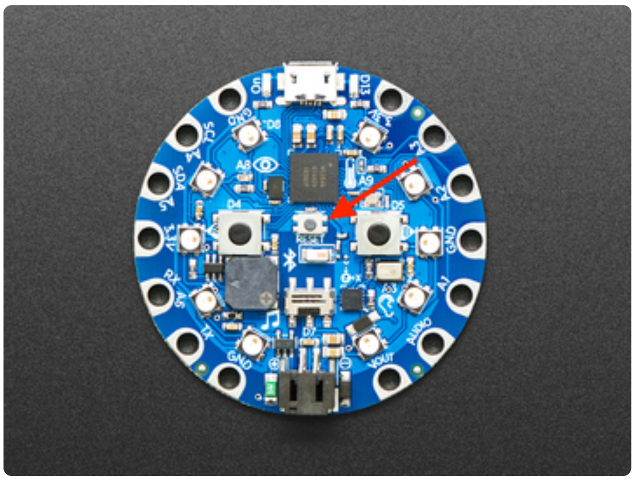
Follow this quick step-by-step to install or update CircuitPython on your Circuit Playground Bluefruit.

Download the latest version of CircuitPython for this board via circuitpython.org
https://adafru.it/FNK



Click the link above and download the latest UF2 file

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



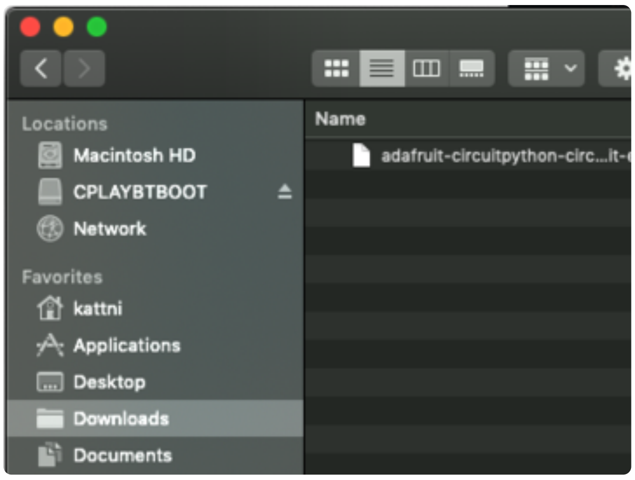
Plug your Circuit Playground Bluefruit into your computer using a known-good data-capable USB cable.

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.

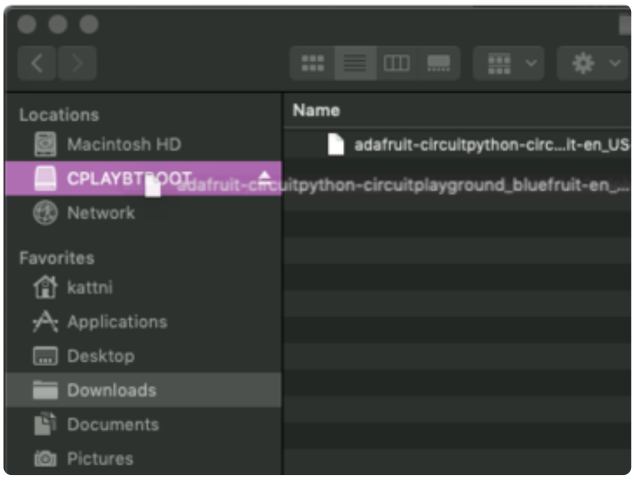
Double-click the small **Reset** button in the middle of the CPB (indicated by the red arrow in the image). The ten NeoPixel LEDs will all turn red, and then will all turn green. If they turn all red and stay red, check the USB cable, try another USB port, etc. The little red LED next to the USB connector will pulse red - this is ok!

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

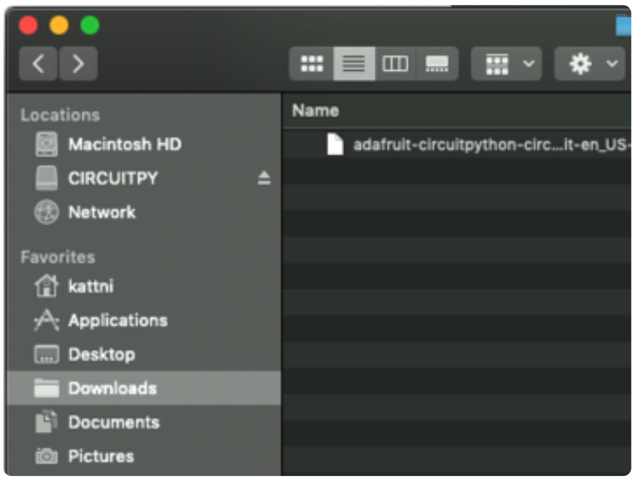
(If double-clicking doesn't do it, try a single-click!)



You will see a new disk drive appear called **CPLAYBTBOOT**.



Drag the **adafruit_circuitpython_etc.uf2** file to **CPLAYBTBOOT.**



The LEDs will turn red. Then, the **CPLAYBTBOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

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

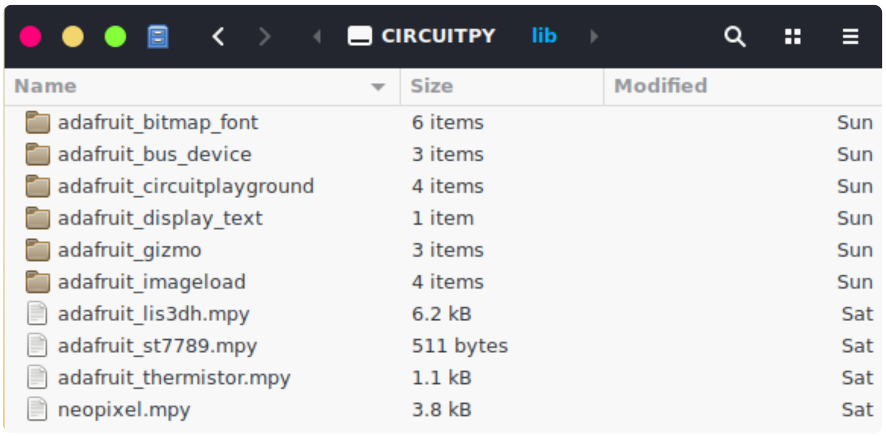
Basic Code

This is the basic version of the code. You set predefined message text and heart colors. Then, you can cycle through them using either of the buttons on the Circuit Playground Bluefruit.

Connect your Circuit Playground Bluefruit to your computer via a known good data USB cable. The board should show up in your operating system file explorer/finder as a flash drive named **CIRCUITPY**. Time to copy over some files for this project.

Libraries

First, make sure you have these libraries copied over to the board following this guide page (https://adafru.it/I3c):



Name	Size	Modified
adafruit_bitmap_font	6 items	Sun
adafruit_bus_device	3 items	Sun
adafruit_circuitplayground	4 items	Sun
adafruit_display_text	1 item	Sun
adafruit_gizmo	3 items	Sun
adafruit_imageload	4 items	Sun
adafruit_lis3dh.mpy	6.2 kB	Sat
adafruit_st7789.mpy	511 bytes	Sat
adafruit_thermistor.mpy	1.1 kB	Sat
neopixel.mpy	3.8 kB	Sat

Make sure you have at least version 2.2.0 of the Adafruit Display Text library.

Code

Here's the code for the basic version. We'll go in to more detail about how it works in the following pages.

This code uses a custom font file and a BMP file for the heart image. To get those as well as the code, click the **Project Zip** link in the embedded code listing below. This will download all the files you need at once.

In addition to the code, be sure to copy the **Multicolore_36.bdf** and **heart_bw.bmp** file to you **CIRCUITPY** folder.

```
# SPDX-FileCopyrightText: 2020 Carter Nelson for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
from random import choice
import displayio
import adafruit_imageload
from adafruit_bitmap_font import bitmap_font
from adafruit_display_text import label
from adafruit_gizmo import tft_gizmo
from adafruit_circuitplayground import cp
```

```
#---| User Config |--------------------------------------------------
HEART_MESSAGES = (
    ("I LUV", "YOU"),
    ("SAY", "YES"),
    ("HUG", "ME"),
    ("BE", "MINE"),
    ("TEXT","ME"),
    ("OMG","LOL"),
    ("PEACE",""),
)
HEART_COLORS = (
    0xEAFF50, # yellow
    0xFFAD50, # orange
    0x9D50FF, # purple
    0x13B0FE, # blue
    0xABFF96, # green
    0xFF96FF, # pink
)
MESSAGE_COLORS = (
    0xFF0000, # red
)
#---| User Config |--------------------------------------------------

# Create the TFT Gizmo display
display = tft_gizmo.TFT_Gizmo()

# Load the candy heart BMP
bitmap, palette = adafruit_imageload.load("/images/heart_bw.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)

heart = displayio.TileGrid(bitmap, pixel_shader=palette)

# Set up message text
font = bitmap_font.load_font("/fonts/Multicolore_36.bdf")
line1 = label.Label(font, text="?"*9)
line2 = label.Label(font, text="?"*5)
line1.anchor_point = (0.5, 0)    # middle top
line2.anchor_point = (0.5, 1.0)  # middle bottom

# Set up group and add to display
group = displayio.Group()
group.append(heart)
group.append(line1)
group.append(line2)
display.root_group = group

while True:
    # turn off auto refresh while we change some things
    display.auto_refresh = False
    # pick a random message
    line1.text, line2.text = choice(HEART_MESSAGES)
    # update location for new text bounds
    line1.anchored_position = (120, 85)
    line2.anchored_position = (120, 175)
    # pick a random text color
    line1.color = line2.color = choice(MESSAGE_COLORS)
    # pick a random heart color
    palette[1] = choice(HEART_COLORS)
    # OK, now turn auto refresh back on to display
    display.auto_refresh = True
    # wait for button press
    while not cp.button_a and not cp.button_b:
        pass
    # just a little debounce
    time.sleep(0.25)
```

Be sure you click the "Download Project Zip" link to get all the files and not just the code. You'll get the heart bitmap and the font file too.

Customizing

You can customize the messages and the colors if you want. All the lines of code to do this are at the top.

Changing Messages

The messages are stored as a tuple of tuples. Yah, that sounds weird, but tuples can contain anything, including other tuples. It's these lines of code at the top:

```
HEART_MESSAGES = (
    ("I LUV", "YOU"),
    ("SAY", "YES"),
    ("HUG", "ME"),
    ("BE", "MINE"),
    ("TEXT","ME"),
    ("OMG","LOL"),
    ("PEACE",""),
)
```

Each message is a tuple with two text strings - one for each line. You need to specify both, even if you only want one line. See the **PEACE** message as an example of a single line message. Its second line is blank. You can remove or add more messages as you wish or edit the ones that are there.

NOTE: Line 1 can have up to 9 characters. Line 2 can have up to 5 characters.

Be careful not to forget the trailing comma at the end of each line in the tuple. It is OK if the last line has one as well.

Changing Colors

You can also change the heart and message text colors if you want. These are also defined as tuples, but this time just a straight simple tuple of color values. It's these lines of code at the top:

```
HEART_COLORS = (
    0xEAFF50, # yellow
    0xFFAD50, # orange
    0x9D50FF, # purple
    0x13B0FE, # blue
    0xABFF96, # green
    0xFF96FF, # pink
)
MESSAGE_COLORS = (
    0xFF0000, # red
)
```

There's a separate tuple for the heart colors, `HEART_COLORS`, and one for the message colors, `MESSAGE_COLORS`. Just add/remove entries as you wish. We kept `MESSAGE_COLORS` to a single value in the default code just because it seemed to look the best. But you can add more if you want.

Changing Interaction

The default code changes messages whenever either of the buttons on the Circuit Playground are pressed. That's done with these lines of code:

```
# wait for button press
while not cp.button_a and not cp.button_b:
    pass
# just a little debounce
time.sleep(0.25)
```

You could change that to something else if you want. You would replace those lines with something else. Some ideas are **tap**:

```
# wait for tap
while not cp.tapped:
    pass
```

shake:

```
# wait for shake
while not cp.shake():
    pass
```

or a simple **timer**:

```
# change every 5 seconds
time.sleep(5)
```

But you could also maybe do sound? Or temperature?

How It Works

This code isn't very complex. However, there a few things it does that are worth discussing so you can use the same ideas in your code and projects.

Custom Font

To get the TFT candy heart to look like an actual candy heart, a custom font is used. There's a guide about this here:

Custom Fonts for CircuitPython Displays

https://adafru.it/EFI

The font used for the TFT Candy Hearts is this one:

Multicolore font from dafont

https://adafru.it/IZA

which we've already converted for use with CircuitPython and this project. The font file will be included along with the code when you download the Project Zip. See the Code pages for details about downloading it and copying it to your board.

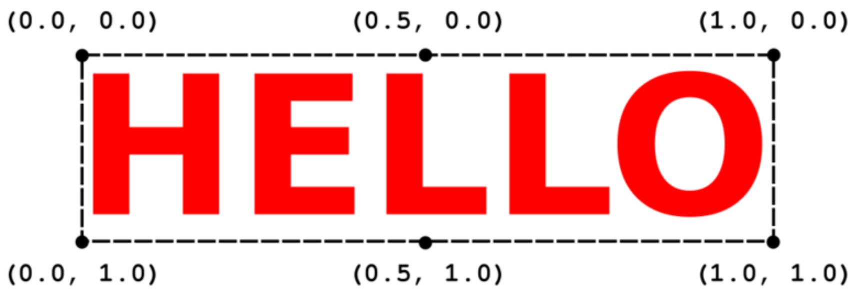
Text Positioning

Thanks to a recent update (https://adafru.it/IUe) to the CircuitPython Display Text (https://adafru.it/FiA) library, you can now change the anchor point used to locate the text label. You do so by using the `anchor_point` property of the `label` and giving it an `(x, y)` tuple, like this:

```
label.anchor_point = (0.1, 0.8)
```

The values range from 0 to 1 with `x` being the horizontal and `y` being the vertical. The origin is in the upper left corner. A value of 0 is at the origin. A value of 1 is all the way to the right/down.

Here are some example locations:

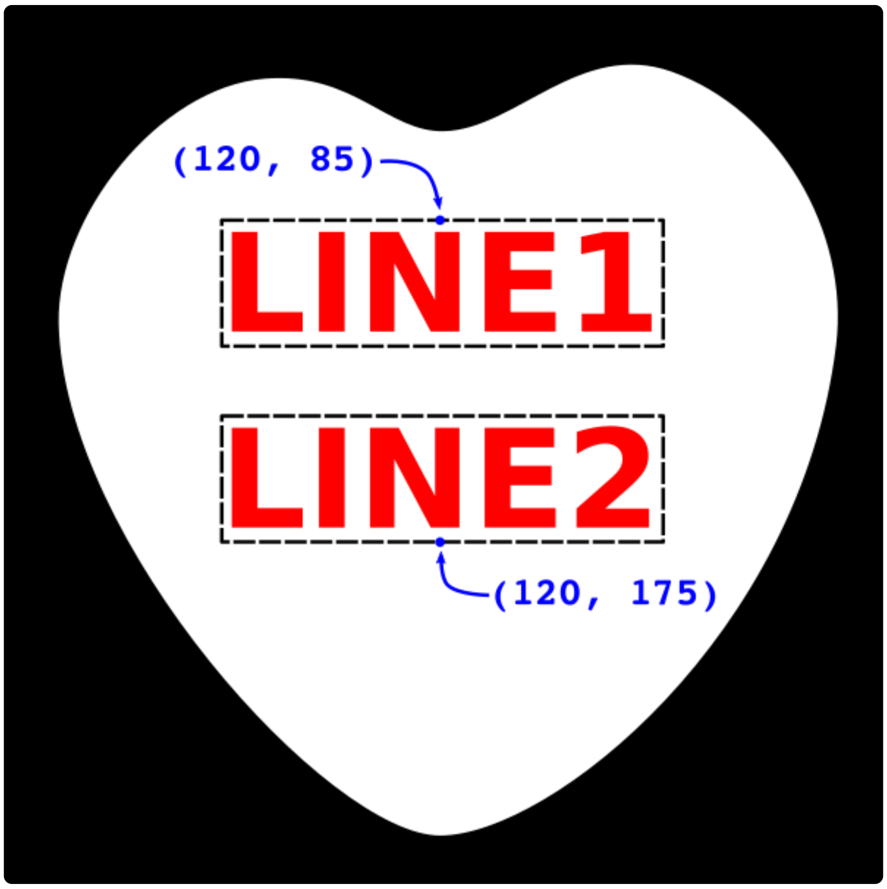


You can then set the position of the label on the display using the `anchored_position` property and also specifying an `(x, y)` tuple. But this time, the values are actual screen coordinates in pixels, like this:

```
label.anchored_position = (120, 85)
```

The candy hearts code uses two labels, one for each line of text. The `x` position is simply half the screen width, 240 / 2 = 120, so that it is centered. The `y` positions were determined by trial and error. Just came up with values that looked about right.

The end result is something like this:



If you want to tweak the position of the messages, you can change these lines of code:

```
# update location for new text bounds
line1.anchored_position = (120, 85)
line2.anchored_position = (120, 175)
```

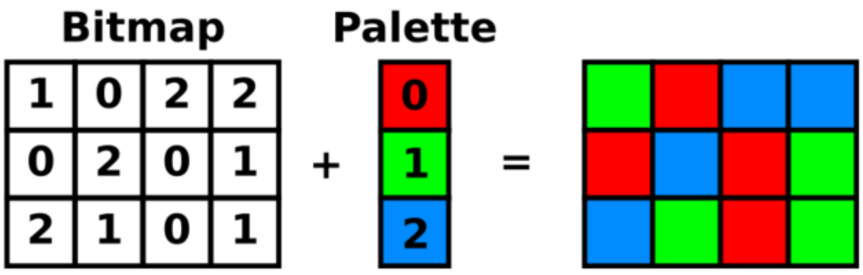
Changing Heart Color

The way the heart color is changed is kind of interesting. Instead of having separate bitmap files for each color, the palette is altered directly. For a review of how `bitmaps` and `palettes` work in CircuitPython displayio, see here:

Bitmap and Palette in displayio

https://adafru.it/GC2

But for quick reference, we copy the important illustration here:



It is the `palette` that defines the colors that are used to render the `bitmap` on the screen. And you can change any of the palette entries anytime you want by doing something like:

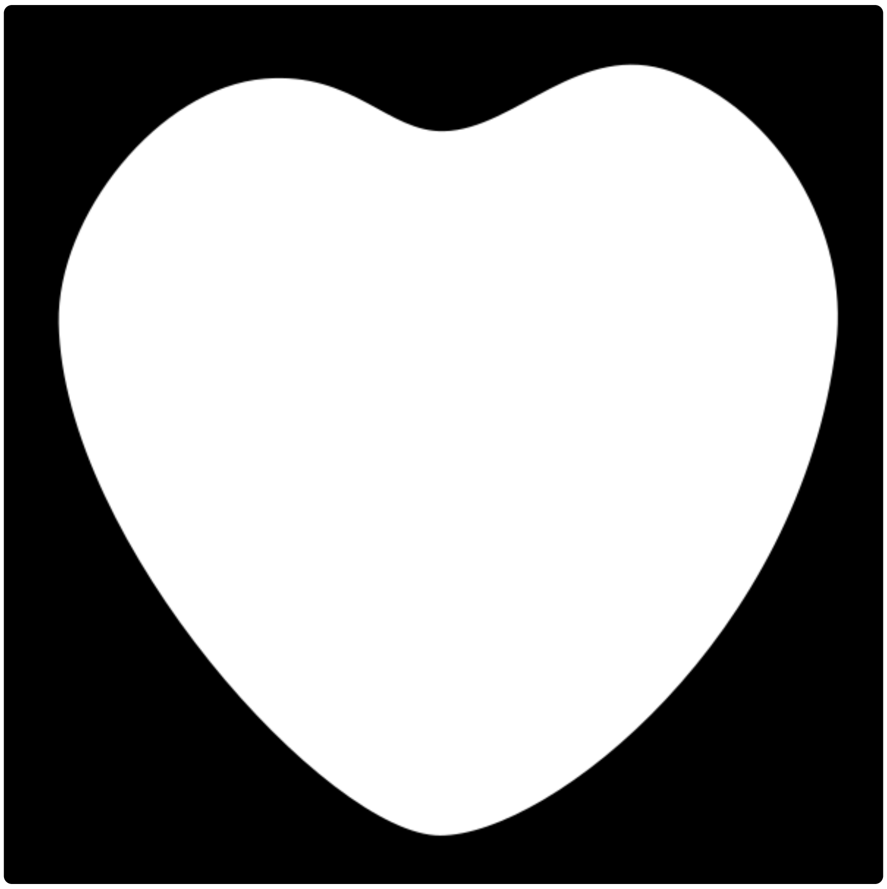
```
palette[1] = 0x0000FF # blue
```

This would set the second entry (index 1) of the palette to the color blue. Any pixel of the bitmap using that palette entry would then get rendered as blue. But then you can do:

```
palette[1] = 0x00FF00 # green
```

and those same pixels would now become green.

This trick is used for the heart. The actual bitmap is only two colors - black and white:



The black pixels are left alone so the background is always black. But you could change that if you want. The candy heart code only changes the white entry of the bitmap. The actual colors are defined by you at the top of the code. The `choice` function from the `random` library is used to select one of these at random. And then the palette entry for white is set to it:

```
palette[1] = choice(HEART_COLORS)
```

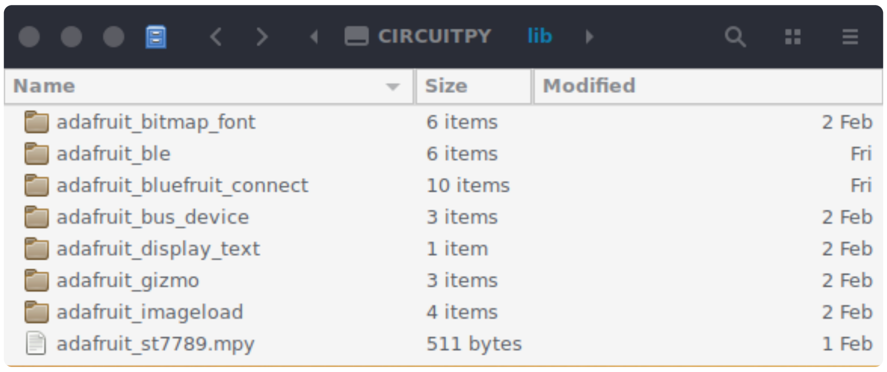
Then whatever used to be white becomes the new color. And when it happens again, the color is replaced by a new random color. And so on.

BLE Code

How about a version that works over Bluetooth? You know, so you could like control it from your phone? Sure, we can do that. Here's a version of the code that let's you use the Adafruit Bluefruit LE Connect App (https://adafru.it/DNc) to send text and color. Now, instead of pre-defined messages and colors, you can use the BLE Connect App (https://adafru.it/DNc) to send them.

Libraries

First, make sure you have these libraries copied over to the **/lib** directory on the board's **CIRCUITPY** drive following this guide page (https://adafru.it/I3c):



Name	Size	Modified
adafruit_bitmap_font	6 items	2 Feb
adafruit_ble	6 items	Fri
adafruit_bluefruit_connect	10 items	Fri
adafruit_bus_device	3 items	2 Feb
adafruit_display_text	1 item	2 Feb
adafruit_gizmo	3 items	2 Feb
adafruit_imageload	4 items	2 Feb
adafruit_st7789.mpy	511 bytes	1 Feb

Make sure you have at least version 2.2.0 of the Adafruit Display Text library.

Code

Here's the BLE version of the code. We'll talk about how to interact with it next.

This code uses a custom font file and a BMP file for the heart image. To get those as well as the code, click the **Project Zip** link in the embedded code listing below. This will download all the files you need at once.

In addition to the code, be sure to copy the **Multicolore_36.bdf** and **heart_bw.bmp** file to you **CIRCUITPY** folder.

```
# SPDX-FileCopyrightText: 2020 Carter Nelson for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Display stuff
import displayio
import adafruit_imageload
from adafruit_bitmap_font import bitmap_font
from adafruit_display_text import label
from adafruit_gizmo import tft_gizmo
# BLE stuff
from adafruit_ble import BLERadio
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.nordic import UARTService
from adafruit_bluefruit_connect.packet import Packet
from adafruit_bluefruit_connect.color_packet import ColorPacket

#---| User Config |--------------------------------------------------
BLE_NAME = "Candy Heart"
MESSAGE_DELIMITER = ","
MESSAGE_COLOR = 0xFF0000
#---| User Config |--------------------------------------------------
```

```
# Setup BLE radio and service
ble = BLERadio()
uart = UARTService()
advertisement = ProvideServicesAdvertisement(uart)
ble._adapter.name = BLE_NAME #pylint: disable=protected-access

# Create the TFT Gizmo display
display = tft_gizmo.TFT_Gizmo()

# Load the candy heart BMP
bitmap, palette = adafruit_imageload.load("/images/heart_bw.bmp",
                                          bitmap=displayio.Bitmap,
                                          palette=displayio.Palette)

heart = displayio.TileGrid(bitmap, pixel_shader=palette)

# Set up message text
LINE1_MAX = 9
LINE2_MAX = 5
font = bitmap_font.load_font("/fonts/Multicolore_36.bdf")
line1 = label.Label(font, text="?"*LINE1_MAX)
line2 = label.Label(font, text="?"*LINE2_MAX)
line1.anchor_point = (0.5, 0)    # middle top
line2.anchor_point = (0.5, 1.0)  # middle bottom
line1.anchored_position = (120, 85)
line2.anchored_position = (120, 175)
line1.color = line2.color = MESSAGE_COLOR

# Set up group and add to display
group = displayio.Group()
group.append(heart)
group.append(line1)
group.append(line2)
display.root_group = group

def update_heart(message, heart_color):
    # turn off auto refresh while we change some things
    display.auto_refresh = False
    # set message text
    text1, _, text2 = message.partition(MESSAGE_DELIMITER)
    line1.text = text1[:LINE1_MAX] if len(text1) > LINE1_MAX else text1
    line2.text = text1[:LINE2_MAX] if len(text2) > LINE2_MAX else text2
    # update location for new text bounds
    line1.anchored_position = (120, 85)
    line2.anchored_position = (120, 175)
    # set heart color
    palette[1] = heart_color
    # OK, now turn auto refresh back on to display
    display.auto_refresh = True

# Initial update
text = "TEXT,ME"
color = 0x00FFFF
update_heart(text, color)

# Loop forever
while True:
    # advertise and wait for connection
    print("WAITING...")
    ble.start_advertising(advertisement)
    while not ble.connected:
        pass

    # connected
    print("CONNECTED")
    ble.stop_advertising()

    # receive and handle BLE traffic
    while ble.connected:
        if uart.in_waiting:
            raw_bytes = uart.read(uart.in_waiting)
            if raw_bytes[0] == ord('!'):
            # BLE Connect Control Packet
                packet = Packet.from_bytes(raw_bytes)
                if isinstance(packet, ColorPacket):
                    print("color = ", color)
                    color = packet.color
            else:
            # Just plain text
                text = raw_bytes.decode("utf-8").strip()
                print("text = ", text)
            update_heart(text, color)

    # disconnected
    print("DISCONNECTED")
```

Be sure you click the "Download Project Zip" link to get all the files and not just the code. You'll get the heart bitmap and the font file too.

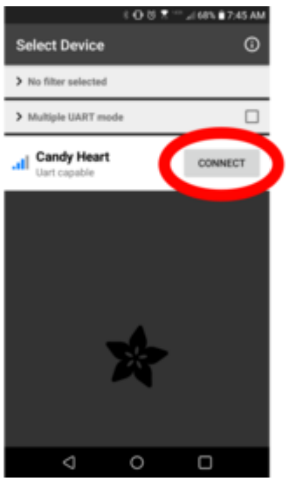
Bonus CLUE Versions

The Adafruit CLUE (http://adafru.it/4500) uses the same TFT display as the Gizmo. This makes it pretty easy to adapt the code for use on the CLUE. You'll find CLUE versions of the code in the project zip. Looks for the two files that names start with **clue**.

Using BLE Connect App

Connect

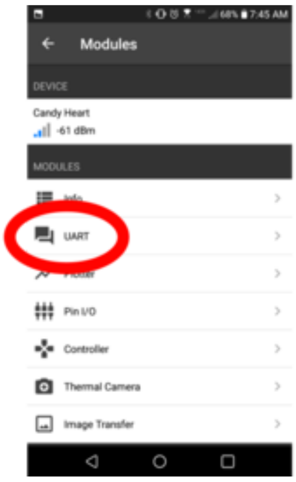
To make the initial connection, launch the BLE Connect App (https://adafru.it/DNc), and then:



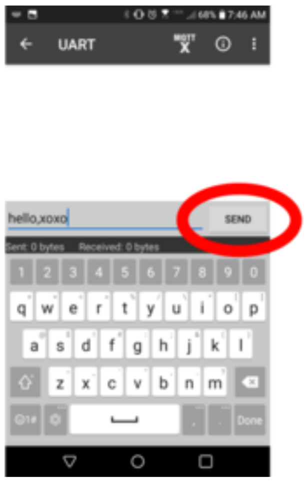
Look for the device named **Candy Heart** and click **CONNECT**.

Change Text

To change the heart text message, follow these steps.

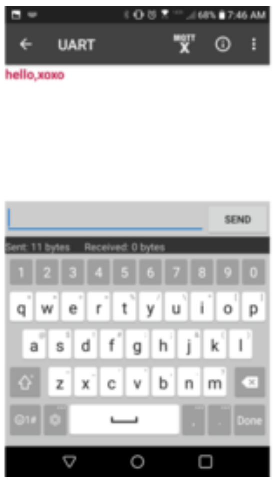


Select the **UART** option in the modules list.



Type in the text you want for the message and click **SEND**.

NOTE: Use a comma to separate the two lines of text, ex: `line1,line2`

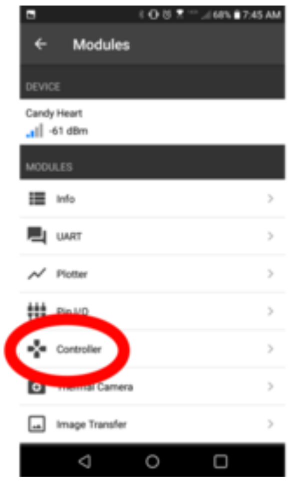


The text will be sent.

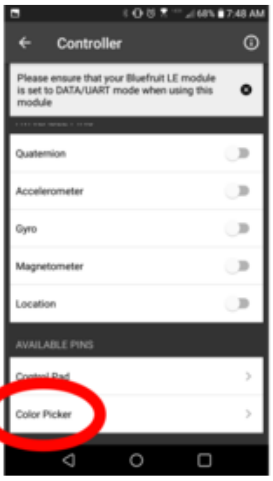
And the heart message will be updated.

Change Color

To change the heart color, follow these steps.

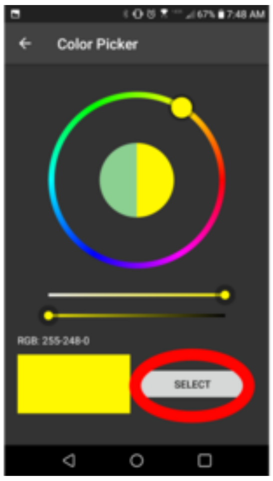


Select the **Controller** option in the modules list.

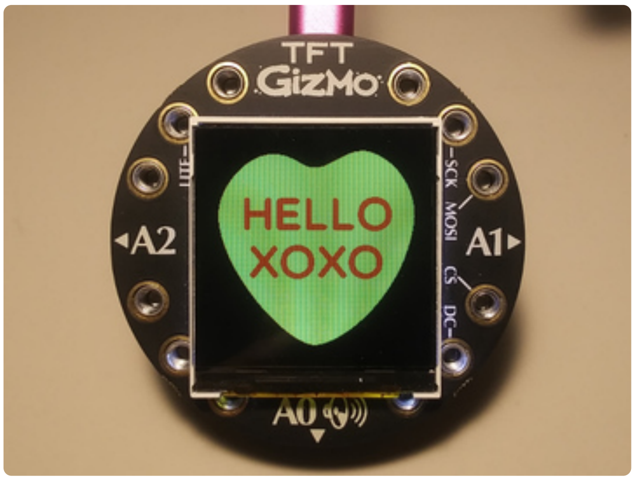


Please ensure that your Bluefruit LE module is set to DATA/UART mode when using this module

Click the **Color Picker** option.



Select the color you want and hit **SELECT**.



And the heart color will be updated.