Snow Globe with Circuit Playground Bluefruit
Created by John Park

Last updated on 2021-02-08 06:46:16 PM EST
Overview

You can make your own fun, custom snow globe for the holidays, or any time of year! Using a Circuit Playground Bluetooth running CircuitPython, the built in accelerometer will detect shaking and launch a little local light show.

Multiple animated sequences can be selected from within the Adafruit Bluefruit App for iOS and Android, allowing you to adjust the show from your mobile device.

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...
$24.95
In Stock
Add to Cart

DIY Snow Globe Kit
Not merely a snow globe, but a show globe! This custom container is durable, clear and round, with a large flat screw-on top and a press-fit rubber stopper. The stopper makes...
$4.95
In Stock
Add to Cart
Lithium Ion Polymer Battery with Short Cable - 3.7V 350mAh
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 battery...
$5.95
In Stock
Add to Cart
Adafruit Micro Lipo - USB LiIon/LiPoly charger
Oh so adorable, this is the tiniest little lipo charger, so handy you can keep it any project box! Its also easy to use. Simply plug in the gold plated contacts into any USB port and a...
$5.95
In Stock
Add to Cart

Pink and Purple Braided USB A to Micro B Cable - 2 meter long
This cable is super-fashionable with a woven pink and purple Blinka-like pattern! First let's talk about the cover and over-molding. We got these in custom colors,...
Out of Stock

Tools and Materials
In addition to the parts above, you'll also need:

- Water
- Glycerine
- Glitter
- Sequins
- Figurines/scenery
- E6000 glue
- Double stick foam
Install or Update CircuitPython

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

1. Click the link above and download the latest UF2 file
   Download and save it to your Desktop (or wherever is handy)

2. 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! :)

© Adafruit Industries

https://learn.adafruit.com/snow-globe-bluefruit-cpb
Code the Bluetooth Snow Globe

Libraries

Now we'll install the libraries that we need to run the code on the Circuit Playground Bluefruit.

Click this link (https://adafru.it/ENC) to go to the circuitpython.org Libraries page. Download the latest version of the Bundle library .zip file that matches the version of CircuitPython you're using on the board.

Uncompress the .zip file and then copy the following directories and .mpy files to the lib directory of the CIRCUITPY drive:

- adafruit_ble
- adafruit_bluefruit_connect
- adafruit_bus_device
- adafruit_lis3dh.mpy
- neopixel.mpy

Your lib directory on the CIRCUITPY drive should be similar to the picture here.

The Mu Editor

Adafruit recommends using the free program Mu to edit your CircuitPython programs and save them on your Circuit Playground Bluefruit. You can use any text editor, but Mu has some handy features.

See this page on the Circuit Playground Bluefruit guide (https://adafru.it/GDt) on the steps used to install Mu.

Snow Globe Code

Here is the code that we'll run on the Circuit Playground Bluefruit.

Copy this code and then paste it into a new document in Mu, then save it to your CIRCUITPY drive as code.py

```python
import time
import random
import board
import busio
import neopixel
import adafruit_lis3dh
from adafruit_bluefruit_connect.packet import Packet
from adafruit_bluefruit_connect.color_packet import ColorPacket
from adafruit_bluefruit_connect.button_packet import ButtonPacket

from adafruit_ble import BLERadio
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble.services.nordic import UARTService
```

© Adafruit Industries
SNOWGLOBE_NAME = "SNOWGLOBE"  # name that will show up on smart device
DEFAULT_ANIMATION = 0   # 0-3, index in ANIMATIONS list
DEFAULT_DURATION = 5    # total seconds to play animation
DEFAULT_SPEED = 0.1     # delay in seconds between updates
DEFAULT_COLOR = 0xFF0000 # hex color value
DEFAULT_SHAKE = 20      # lower number is more sensitive

# you can define more animation functions below
# here, specify the four to be used
ANIMATIONS = ('spin', 'pulse', 'strobe', 'sparkle')

# Configuration settings
snow_config = {
    'animation': DEFAULT_ANIMATION,
    'duration': DEFAULT_DURATION,
    'speed': DEFAULT_SPEED,
    'color': DEFAULT_COLOR,
    'shake': DEFAULT_SHAKE,
}

# Setup NeoPixels
pixels = neopixel.NeoPixel(board.NEOPixel, 10)

# Setup accelo
accelo_i2c = busio.I2C(board.ACCELEROMETER_SCL, board.ACCELEROMETER_SDA)
accelo = adafruit_lis3dh.LIS3DH_I2C(accelo_i2c, address=0x19)

# Setup BLE
ble = BLERadio()
uart = UARTService()
advertisement = ProvideServicesAdvertisement(uart)
ble._adapter.name = SNOWGLOBE_NAME #pylint: disable=protected-access

--- ANIMATIONS ---------------------------------

def spin(config):
    start_time = time.monotonic()
    last_update = start_time
    p = -1
    while time.monotonic() - start_time < config['duration']:
        if time.monotonic() - last_update > config['speed']:
            pixels.fill(0)
            pixels[p % 10] = config['color']
            p -= 1
            last_update = time.monotonic()

def pulse(config):
    start_time = time.monotonic()
    last_update = start_time
    brightness = 0
    delta = 0.05
    pixels.brightness = 0
    pixels.fill(config['color'])
    while time.monotonic() - start_time < config['duration']:
        if time.monotonic() - last_update > config['speed']:
            brightness += delta
            if brightness > 1:
                brightness = 1
            delta *= -1
        if brightness < 0:
brightness = 0
delta *= -1
pixels.brightness = brightness
last_update = time.monotonic()

def strobe(config):
    start_time = time.monotonic()
    last_update = start_time
    turn_on = True
    while time.monotonic() - start_time < config['duration']:
        if time.monotonic() - last_update > config['speed']:
            if turn_on:
                pixels.fill(config['color'])
            else:
                pixels.fill(0)
            turn_on = not turn_on
        last_update = time.monotonic()

def sparkle(config):
    start_time = time.monotonic()
    last_update = start_time
    while time.monotonic() - start_time < config['duration']:
        if time.monotonic() - last_update > config['speed']:
            pixels.fill(0)
            pixels[random.randint(0, 9)] = config['color']
        last_update = time.monotonic()

#---| ANIMATIONS |----------------------------------------------------

def play_animation(config):
    #pylint: disable=eval-used
    eval(ANIMATIONS[config['animation']])(config)
    pixels.fill(0)

def indicate(event=None):
    if not isinstance(event, str):
        return
    event = event.strip().upper()
    if event == 'START':
        for _ in range(2):
            for i in range(10):
                pixels[i] = DEFAULT_COLOR
                time.sleep(0.05)
                pixels.fill(0)
    if event == 'CONNECTED':
        for _ in range(5):
            pixels.fill(0x0000FF)
            time.sleep(0.1)
            pixels.fill(0)
            time.sleep(0.1)
    if event == 'DISCONNECTED':
        for _ in range(5):
            pixels.fill(0x00FF00)
            time.sleep(0.1)
            pixels.fill(0)
            time.sleep(0.1)
    indicate('START')

# Are we already advertising?
advertising = False
while True:
    # While BLE is *not* connected
    while not ble.connected:
        if accelo.shake(snow_config['shake'], 5, 0):
            play_animation(snow_config)
        if not advertising:
            ble.start_advertising(advertisement)
        advertising = True

    # connected
    indicate('CONNECTED')

while ble.connected:
    # Once we're connected, we're not advertising any more.
    advertising = False

    if accelo.shake(snow_config['shake'], 5, 0):
        play_animation(snow_config)

    if uart.in_waiting:
        try:
            packet = Packet.from_stream(uart)
        except ValueError:
            continue

        if isinstance(packet, ColorPacket):
            # COLOR
            snow_config['color'] = packet.color
            pixels.fill(snow_config['color'])
            time.sleep(0.5)
            pixels.fill(0)

        if isinstance(packet, ButtonPacket) and packet.pressed:
            # SPEED
            if packet.button == ButtonPacket.UP:
                speed = snow_config['speed'] - 0.05
                speed = 0.05 if speed < 0.05 else speed
                snow_config['speed'] = speed
                play_animation(snow_config)
            if packet.button == ButtonPacket.DOWN:
                speed = snow_config['speed'] + 0.05
                snow_config['speed'] = speed
                play_animation(snow_config)

            # DURATION
            if packet.button == ButtonPacket.LEFT:
                duration = snow_config['duration'] - 1
                duration = 1 if duration < 1 else duration
                snow_config['duration'] = duration
                play_animation(snow_config)
            if packet.button == ButtonPacket.RIGHT:
                duration = snow_config['duration'] + 1
                snow_config['duration'] = duration
                play_animation(snow_config)
Once you've saved the code to the board, the Circuit Playground Bluefruit will swirl the NeoPixel ring once in the default color. It is now ready to be shaken, try it out!

The default animation will play each time you shake the CPB. You can now connect to it with the Bluefruit app on iOS or Android to adjust the color, animation pattern, speed, and duration.

We'll take a closer look at those settings on the next page after we build the snow globe!
Snow Globe DIY

Here's the crafty part of this project! You now get to choose decorations to use inside your snow globe, fill it up, seal it, and then embed the Circuit Playground Bluefruit into the base!

Scene Building

You can get creative with your snow globe filling.

- There are lots of figures, scenic elements, and models you can add to a globe as well.

  NOTE: be sure to test objects to make sure they'll work well in water -- some paints and dyes are not water fast.

  Use E6000 glue to secure them to the plug, and even add some to just float around!
A little glue under the base of this owl, deer, and tree is all it takes. Give the glue time to cure before adding it to the water filled globe.
Globe Filling

The key ingredients for a beautiful snow globe are water, glycerine, and glitter.

Fill the globe with filtered or distilled water for best clarity, about 2-1/2 cups. Don't go all the way to the top, you'll need to leave some room for the plug.

The glycerine makes the liquid a bit more viscous, encouraging slower, lazier falling motion of the glitter. Just a small amount will do.
Next, add your glitter, snowflakes, sequins, beads, and any other things you'd like to see in your snow globe.

Here, I went with a mostly white array of decoration, which look really nice with the colored LED lighting.

You can then top off the water level if necessary to the line in the neck of the globe right before the threading begins.
While the plug is meant to keep the globe watertight from the pressure of the screw cap, we'll take the extra measure of sealing it with E6000 glue. This is because our CPB and battery will be right there and we don't want them to get wet should any small leak occur!

Wipe off the inside of the globe neck, then apply a moderate amount of glue around the perimeter of the plug as shown.

Press the plug into place and then remove any excess glue.

Allow the glue to dry, a full cure can take a full day or more
depending on ambient temperature and humidity -- refer to the tube instructions for more details.
CPB Prep

Use some double stick foam tape to secure the battery to the back of the Circuit Playground Bluefruit in such a way that you can plug the cable into the JST connector on the board.
Battery Charging

The CPB does not contain a battery charging circuit, so you'll need to unplug the battery from the board and use a dedicated LiPo charger such as the one shown here (https://adafruit.it/doR) that plugs directly into a USB port.
Globe Lights

With the CPB programmed, the battery charged and plugged into the board, and the water plug glue seal cured, we'll put it all together.

Place the CPB with the NeoPixel side of the board facing into the globe, then screw on the cap.
Control the Snow Globe with the Bluefruit LE Connect App

Download the App

To control the NeoPixel colors, speed, duration, and animation presets from a mobile device (phone or tablet), you'll use the free Adafruit Bluefruit LE Connect App (https://adafruit.it/iCi). Install it from the Apple App Store (https://adafruit.it/BYj) or Google Play App Store (https://adafruit.it/f4G).
Custom Lighting Color

Now, give the globe a shake and you'll see the lights animate! Launch the Bluefruit app on your iOS or Android device at this point to learn how to adjust the colors and animations.

Making sure Bluetooth is enabled on your device, launch the app and then connect to the snow globe CPB as shown. The LEDs will flash blue when you connect.

Then, go to the Controller module, followed by the Color Picker module.

Here, you can choose a hue and saturation with the color wheel, and brightness with the slider. Press the Send selected color button to send the color to the CPB, which will flash that color briefly, helping you judge the look of it on the actual snow globe.

This color will be used when you shake the globe. Once you like your color choice, press the Controller back arrow to return to the controller module.
Light Show Tuning

You can shake the globe again to see the animation with your newly selected color. What if you want to speed it up/down, make it run for shorter or longer duration, or pick a different animation? That's where the Control Pad module comes in!

Click Control Pad and you'll be presented with the arrow key d-pad and 1-4 buttons.

Each of these controls with adjust a parameter of the lighting animation on the CPB when pressed:

- up/down arrows make the animation fun faster or slower
- left/right arrow decrease or increase the duration of the animation
- 1-4 are four different animation patterns as defined in the CircuitPython code, such as spin, strobe, sparkle, and fade

After pressing each button the globe will run the animation to show the effect of the new settings.