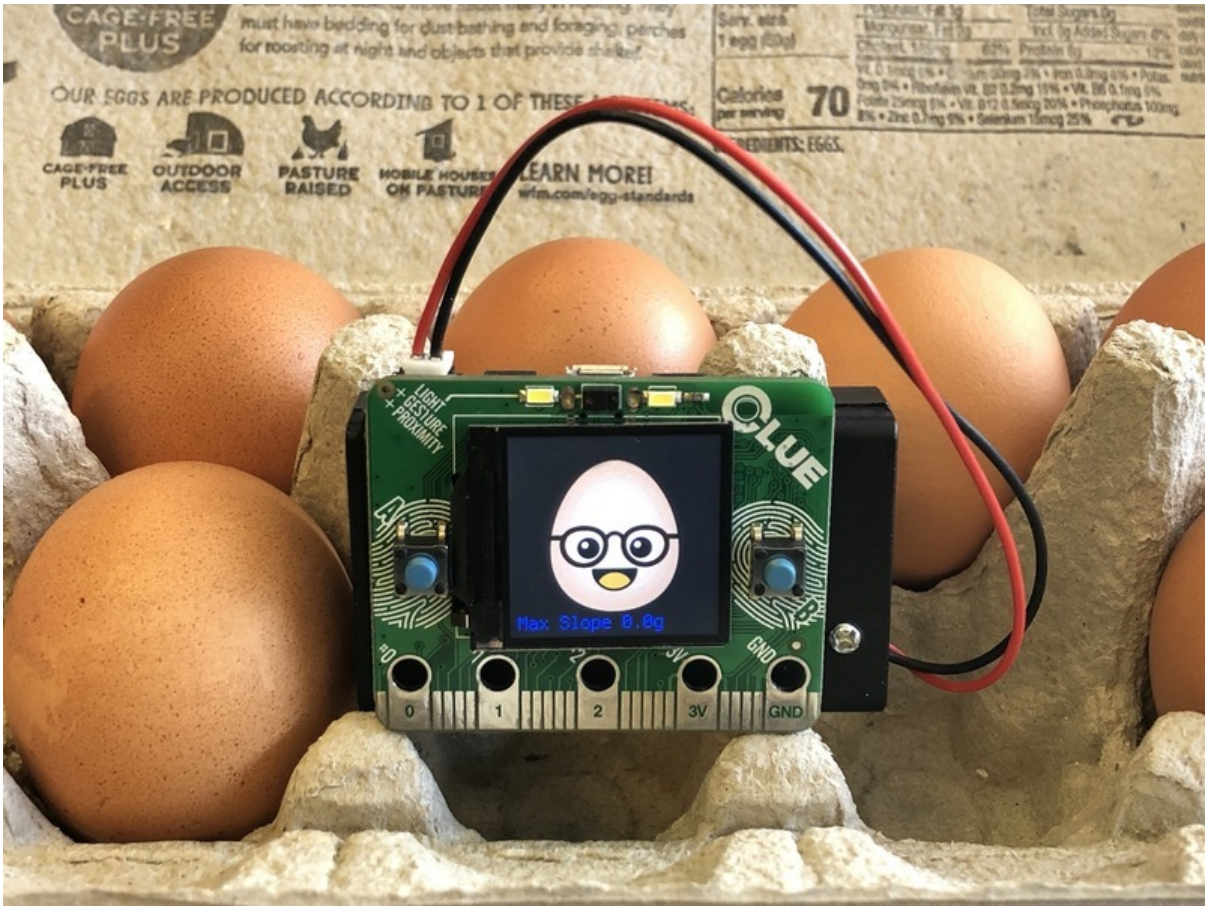
Digital Egg Drop Experiment with CLUE

Created by John Park



https://learn.adafruit.com/egg-drop-clue

Last updated on 2024-06-03 03:04:36 PM EDT

Table of Contents

Overview

This classic physics experiment is a really fun one! You get to create a descent vehicle for an egg dropped from a great height that helps the egg land gently enough to avoid going SPLAT!

Instead of testing design iterations on precious eggs, we can first prove out our vehicles using the CLUE board! It's packed with sensors and has a display and speaker for feedback -- we'll use those to study the impact g-force of our landing. Once we're satisfied with the data, we can swap out for the more fragile ovoid!

Here's an excellent video by Mark Rober explaining the physics behind the egg drop as well as showing five designs you can try out. Instead of testing on eggs each time, we will first determine the g forces sustained on impact with our CLUE board!

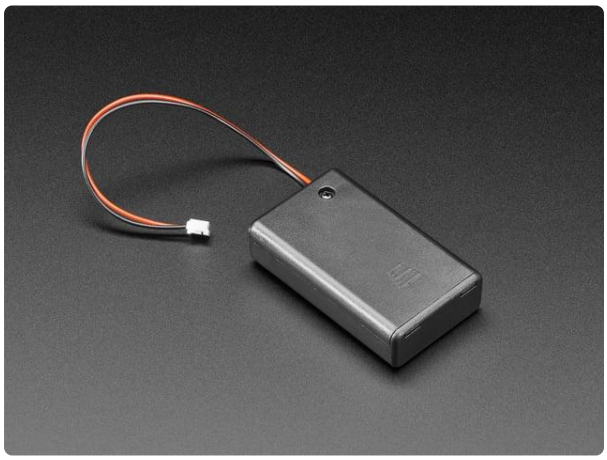
We really don't want you to break your CLUE board -- so be sure to test out your experiments in stages and ideally on softer surfaces before building up your confidence in your design to try it from full height over a harder surface!

Parts

Adafruit CLUE - nRF52840 Express with Bluetooth LE

Do you feel like you just don't have a CLUE? Well, we can help with that - get a CLUE here at Adafruit by picking up this sensor-packed development board. We wanted to build some...

https://www.adafruit.com/product/4500



3 x AAA Battery Holder with On/Off Switch and 2-Pin JST

This battery holder connects 3 AAA batteries together in series for powering all kinds of projects. We spec'd these out because the box is slim, and 3 AAA's add up to about...

https://www.adafruit.com/product/727

Alkaline AAA batteries - 3 pack
Battery power for your portable project! These batteries are good quality at a good price, and work fantastic with any of the kits or projects in the shop that use AAA's. This is a...
https://www.adafruit.com/product/3520

Materials & Tools

You'll want to try out some different materials and descent vehicle strategies.

- Balloons
- Cardboard
- Plastic or paper drinking straws
- Tape
- Zip ties
- Egg cartons
- Rubber bands
- Popcorn
- Plastic or paper bags
- Scissors
- Hobby knife

One more bit of inspiration -- here's a MakeCode egg drop experiment setup running on the micro:bit:

CircuitPython on CLUE

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** flash 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 CLUE from circuitpython.org

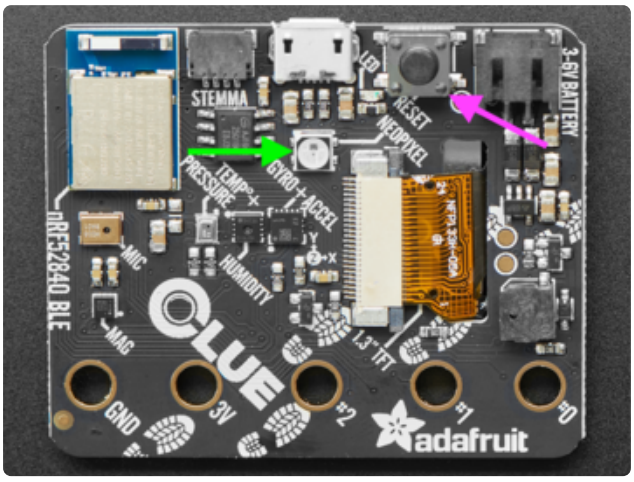
https://adafru.it/IHF

Click the link above to download the latest version of CircuitPython for the CLUE.

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

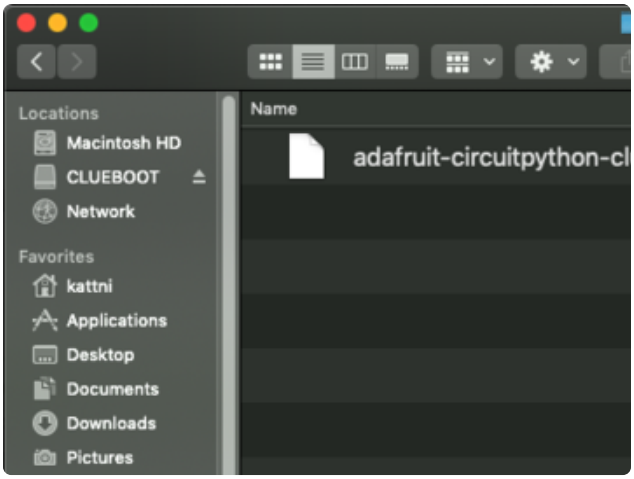
Plug your CLUE into your computer using a known-good USB cable.

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.

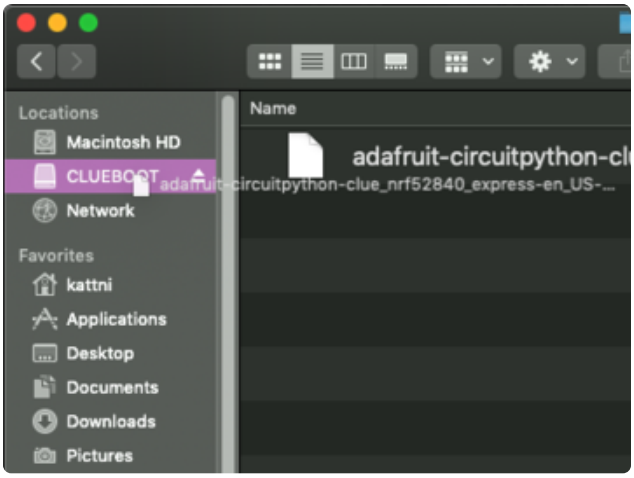


Double-click the **Reset** button on the top (magenta arrow) on your board, and you will see the NeoPixel RGB LED (green arrow) turn green. If it turns red, check the USB cable, try another USB port, etc. **Note:** The little red LED next to the USB connector will pulse red. That's ok!

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

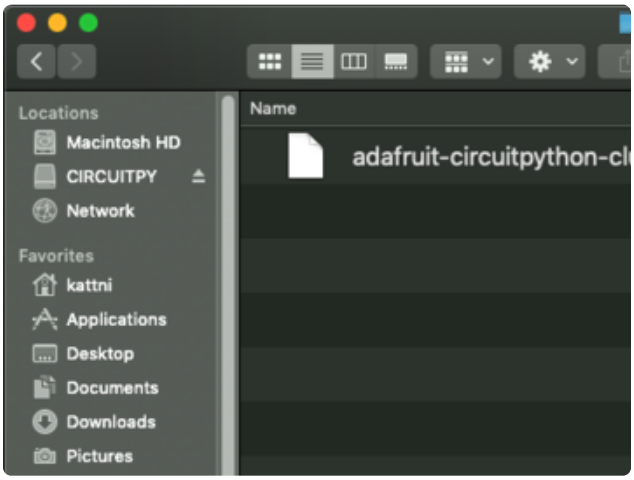


You will see a new disk drive appear called **CLUEBOOT**.



Drag the **adafruit-circuitpython-clue-etc.uf2** file to **CLUEBOOT.**

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



If this is the first time you're installing CircuitPython or you're doing a completely fresh install after erasing the filesystem, you will have two files - **boot_out.txt**, and **code.py**, and one folder - **lib** on your **CIRCUITPY** drive.

If CircuitPython was already installed, the files present before reloading CircuitPython should still be present on your **CIRCUITPY** drive. Loading CircuitPython will not create new files if there was already a CircuitPython filesystem present.

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

Code the Egg Drop CLUE with CircuitPython

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).

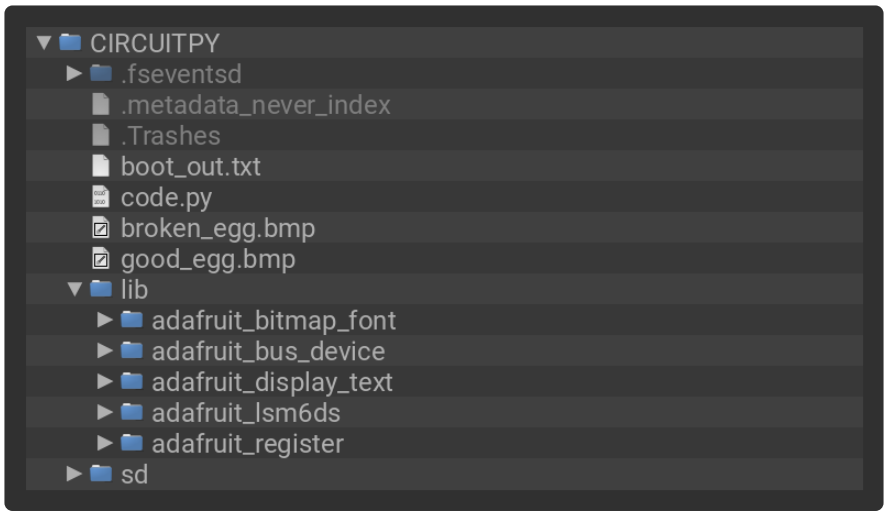
Alternatively, you can use any text editor that saves files.

Installing Project 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 **CLUE_Egg_Drop/** 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:




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

import time
import math
import board
from digitalio import DigitalInOut, Direction, Pull
```

```
import pwmio
from adafruit_lsm6ds.lsm6ds33 import LSM6DS33
from adafruit_lsm6ds import AccelRange, AccelHPF, Rate
from adafruit_display_text import label
import displayio
import terminalio

button_a = DigitalInOut(board.BUTTON_A)
button_a.direction = Direction.INPUT
button_a.pull = Pull.UP

splash = displayio.Group()

# bad egg
BAD_EGG_FILENAME = "broken_egg.bmp"

# CircuitPython 6 & 7 compatible
begg_file = open(BAD_EGG_FILENAME, "rb")
begg_bmp = displayio.OnDiskBitmap(begg_file)
begg_sprite = displayio.TileGrid(
    begg_bmp,
    pixel_shader=getattr(begg_bmp, 'pixel_shader', displayio.ColorConverter())
)

# # CircuitPython 7+ compatible
# begg_bmp = displayio.OnDiskBitmap(BAD_EGG_FILENAME)
# begg_sprite = displayio.TileGrid(begg_bmp, pixel_shader=begg_bmp.pixel_shader)

# good egg
GOOD_EGG_FILENAME = "good_egg.bmp"

# CircuitPython 6 & 7 compatible
gegg_file = open(GOOD_EGG_FILENAME, "rb")
gegg_bmp = displayio.OnDiskBitmap(gegg_file)
gegg_sprite = displayio.TileGrid(
    gegg_bmp,
    pixel_shader=getattr(gegg_bmp, 'pixel_shader', displayio.ColorConverter())
)

# # CircuitPython 7+ compatible
# gegg_bmp = displayio.OnDiskBitmap(GOOD_EGG_FILENAME)
# gegg_sprite = displayio.TileGrid(gegg_bmp, pixel_shader=gegg_bmp.pixel_shader)

# draw the bad egg!
splash.append(begg_sprite)
# draw the good egg on top
splash.append(gegg_sprite)

# Draw a label
text_group = displayio.Group(scale=2, x=10, y=220)
text = "Current & Max Acceleration"
text_area = label.Label(terminalio.FONT, text=text, color=0x0000FF)
text_group.append(text_area) # Subgroup for text scaling
splash.append(text_group)

# display everything so far
board.DISPLAY.root_group = splash

# connect to the accelerometer
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
sensor = LSM6DS33(i2c)
# highest range for impacts!
sensor.accelerometer_range = AccelRange.RANGE_16G
# we'll read at about 1KHz
sensor.accelerometer_rate = Rate.RATE_1_66K_HZ
# Instead of raw accelerometer data, we'll read the *change* in acceleration (shock)
sensor.high_pass_filter = AccelHPF.SLOPE
```

```
sensor.high_pass_filter_enabled = True

# and make a lil buzzer
buzzer = pwmio.PWMOut(board.SPEAKER, variable_frequency=True)
buzzer.frequency = 1000

max_slope = 0
egg_ok = True
while True:
    # This isn't the acceleration but the SLOPE (change!) in acceleration
    x, y, z = sensor.acceleration
    # take the vector length by squaring, adding, taking root
    slope_g = math.sqrt(x*x + y*y + z*z) / 9.8  # take vector, convert to g
    # we'll track the max delta g

    if max_slope < slope_g:
        max_slope = slope_g
        print(slope_g)
        text_area.text = "Max Slope %0.1fg" % max_slope
        if max_slope >= 9 and egg_ok:
            gegg_sprite.x = 300
            time.sleep(0.1)
            egg_ok = False
            buzzer.duty_cycle = 2**15
            time.sleep(2)
            buzzer.duty_cycle = 0
            continue

    if button_a.value is False and egg_ok is False:
        print("Reset")
        time.sleep(0.1)  # debounce
        max_slope = 0
        gegg_sprite.x = 0
        egg_ok = True
```

How It Works

Library Import

First, the code imports libraries: `time`, `math`, `board`, `digitalio` (in this case for using the A button), `pulseio` for sounding the buzzer, the accelerometer `adafruit_lsm6` library, and, so we can display text and graphics on screen, `adafruit_display_text`, `displayio` and `terminalio`.

```
import time
import math
import board
from digitalio import DigitalInOut, Direction, Pull
import pulseio
from adafruit_lsm6ds import LSM6DS33, AccelRange, AccelHPF, Rate
from adafruit_display_text import label
import displayio
import terminalio
```

Button Setup

Next we'll set up the button as an input with pull up resistor.

```
button_a = DigitalInOut(board.BUTTON_A)
button_a.direction = Direction.INPUT
button_a.pull = Pull.UP
```

Display

To prepare the screen for bitmap display, we'll create a `displayio.Group()` called `splash`, and then create tiles for the two egg .bmp image files saved on the CLUE.

```
splash = displayio.Group()

# bad egg
BAD_EGG_FILENAME = "broken_egg.bmp"

# CircuitPython 6 &amp;amp; 7 compatible
begg_file = open(BAD_EGG_FILENAME, "rb")
begg_bmp = displayio.OnDiskBitmap(begg_file)
begg_sprite = displayio.TileGrid(
    begg_bmp,
    pixel_shader=getattr(begg_bmp, 'pixel_shader', displayio.ColorConverter())
)

# # CircuitPython 7+ compatible
# begg_bmp = displayio.OnDiskBitmap(BAD_EGG_FILENAME)
# begg_sprite = displayio.TileGrid(begg_bmp, pixel_shader=begg_bmp.pixel_shader)

# good egg
GOOD_EGG_FILENAME = "good_egg.bmp"

# CircuitPython 6 &amp;amp; 7 compatible
gegg_file = open(GOOD_EGG_FILENAME, "rb")
gegg_bmp = displayio.OnDiskBitmap(gegg_file)
gegg_sprite = displayio.TileGrid(
    gegg_bmp,
    pixel_shader=getattr(gegg_bmp, 'pixel_shader', displayio.ColorConverter())
)

# # CircuitPython 7+ compatible
# gegg_bmp = displayio.OnDiskBitmap(GOOD_EGG_FILENAME)
# gegg_sprite = displayio.TileGrid(gegg_bmp, pixel_shader=gegg_bmp.pixel_shader)

# draw the bad egg!
splash.append(begg_sprite)
# draw the good egg on top
splash.append(gegg_sprite)
```

Label

We'll also prepare the text display by creating a text group with the text "Current & Max Acceleration".

All of these elements will be drawn to the screen with the command `board.DISPLAY.root_group = splash`

```
# Draw a label
text_group = displayio.Group(scale=2, x=10, y=220)
```

```
text = "Current &amp;amp; Max Acceleration"
text_area = label.Label(terminalio.FONT, text=text, color=0x0000FF)
text_group.append(text_area) # Subgroup for text scaling
splash.append(text_group)

# display everything so far
board.DISPLAY.root_group = splash
```

Accelerometer & Buzzer

The accelerometer object will be instantiated as `sensor` connected over the I2C bus to the LSM6DS33 package. The range is set to 16G so it can read large impacts (poor egg!) read at a rate of 1.66KHz.

To filter out noise, the accelerometer's high pass filter is enabled.

We'll also set up the buzzer using pulseio with a frequency of 1000Hz.

```
# connect to the accelerometer
sensor = LSM6DS33(board.I2C())
# highest range for impacts!
sensor.accelerometer_range = AccelRange.RANGE_16G
# we'll read at about 1KHz
sensor.accelerometer_rate = Rate.RATE_1_66K_HZ
# Instead of raw accelerometer data, we'll read the *change* in acceleration (shock)
sensor.high_pass_filter = AccelHPF.SLOPE
sensor.high_pass_filter_enabled = True

# and make a lil buzzer
buzzer = pulseio.PWMOut(board.SPEAKER, variable_frequency=True)
buzzer.frequency = 1000
```

The `max_slope` variable is created with an initial value of 0 -- this is used to measure the slope change in acceleration.

The `egg_ok` variable is set to True and will be used to reset the board after a large impact beyond the maximum slope value of 9.

```
max_slope = 0
egg_ok = True
```

Main Loop

In the main loop of the program the accelerometer is read, and a `slope_g` is determined by squaring, adding and taking the root of the accelerometer value.

When the egg drop CLUE is in freefall, we'll begin checking the `max_slope` value acceleration -- if it is greater than 9, the simulated egg has probably cracked!

In this case, we move the good egg sprite off screen by 300 pixels, revealing the bad egg image. The buzzer will also sound for two seconds.

```
# This isn't the acceleration but the SLOPE (change!) in acceleration
    x, y, z = sensor.acceleration
    # take the vector length by squaring, adding, taking root
    slope_g = math.sqrt(x*x + y*y + z*z) / 9.8  # take vector, convert to g
    # we'll track the max delta g

    if max_slope &lt; slope_g:
        max_slope = slope_g
        print(slope_g)
        text_area.text = "Max Slope %0.1fg" % max_slope
        if max_slope &gt;= 9 and egg_ok:
            gegg_sprite.x = 300
            time.sleep(0.1)
            egg_ok = False
            buzzer.duty_cycle = 2**15
            time.sleep(2)
            buzzer.duty_cycle = 0
            continue
```

Button Reset

Now, we can check the `button_a.value` to see if it changes after the `egg_ok` variable has been set to `False`. Since the button is on a pull up resistor, it is normally `True` until pressed. When pressed, the button value will go to `False` and we can then reset the `max_slope` value, and return the good egg image, and reset the `egg_ok` to `True`!

```
if button_a.value is False and egg_ok is False:
        print("Reset")
        time.sleep(0.1)  # debounce
        max_slope = 0
        gegg_sprite.x = 0
        egg_ok = True
```

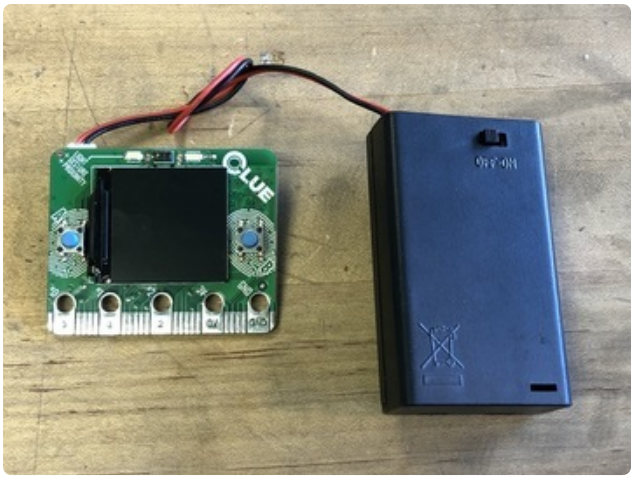
At this point, we're ready for another try!

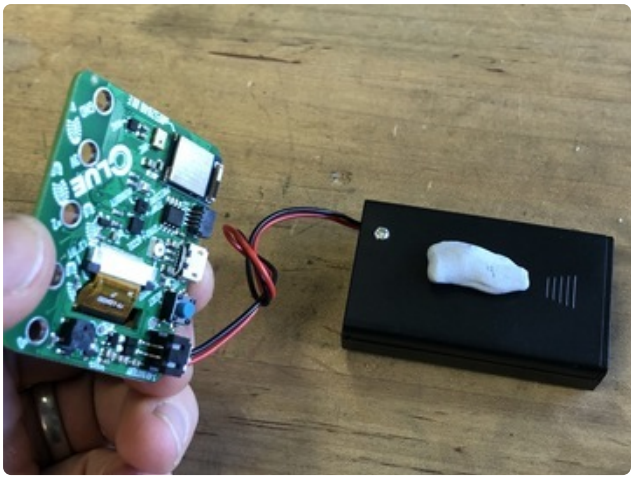
Next, we'll build a landing vehicle and take it for a spin (drop?)!

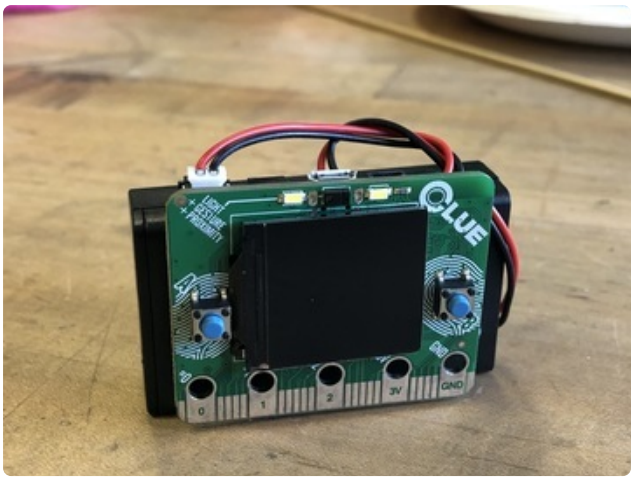
Build an Egg Landing Vehicle for Testing

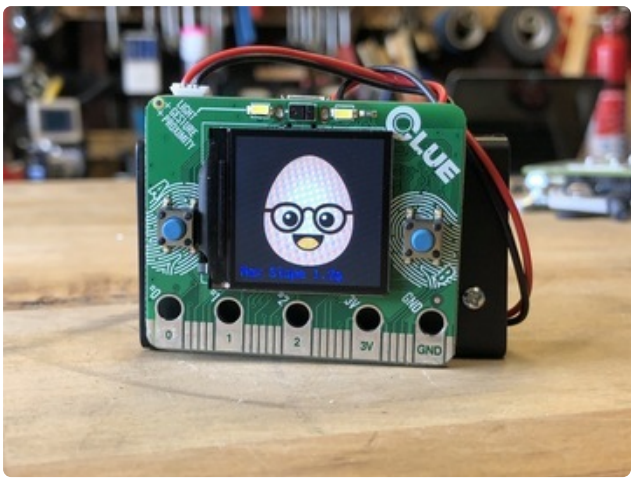
There are a nearly infinite number of different designs and approaches you can take to building an egg landing vehicle -- that's the whole point of this experiment after all!

Here's one approach that works very well -- a partial egg carton pod protected on all sides by impact absorbing balloons.









Power Pack

Place three AAA batteries into the battery holder and then screw the lid shut with the supplied screw.

Plug the JST cable into the CLUE.

Use double stick foam tape or poster putty to secure the CLUE to the battery holder.

Use the battery holder's integrated on/off switch to power the CLUE.

Balloon Lander

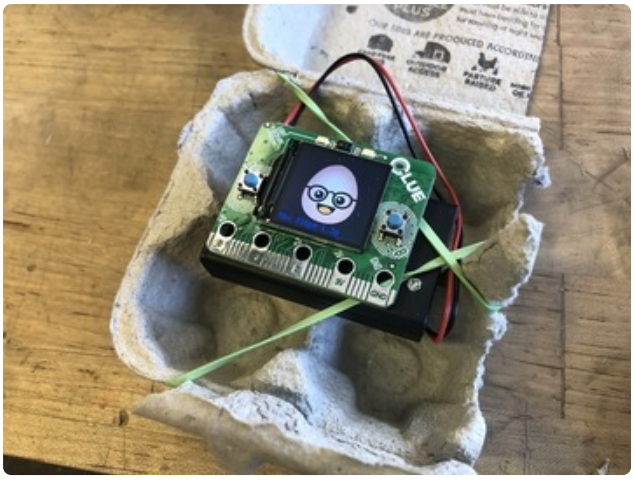
Blow up and knot the ends of six balloons to about 50% full size. We want there to be a nice amount of give so that the balloons deform on impact -- this will increase the amount of time it takes for the CLUE (and later the egg!) to stop moving, thus dissipating much of the kinetic energy.

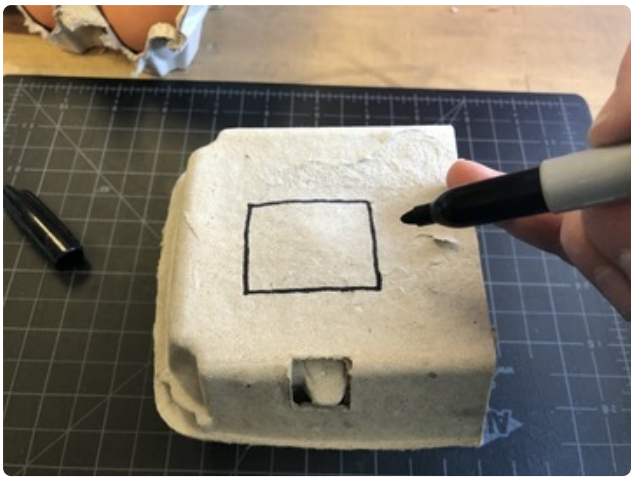
Use zip ties or rubber bands to connect three of the balloons as shown. Repeat this step for the other balloons so you have two separate sets of three.

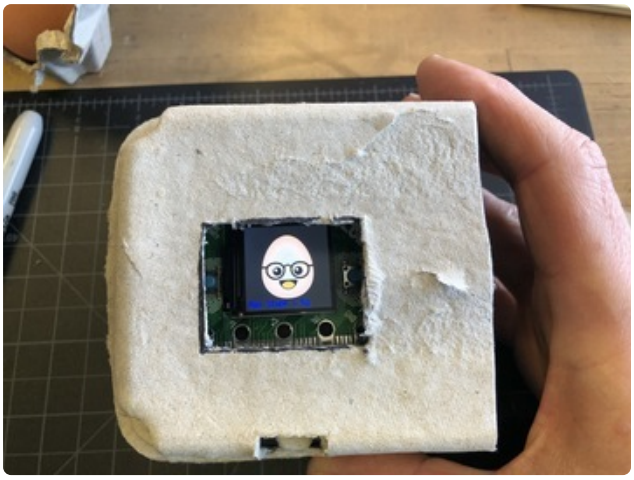
We want to be able to nestle our CLUE or egg in between the two sets of balloons.

Wrap rubber bands around the balloons so they are loosely hanging from the intersection points, then wrap them over the other set to join them with a flexible coupling.

Next, we'll build a pod for the CLUE that will be secured in this space between.







Carton Pod

To begin, cut off a four-egg end of a common egg carton.

An egg can sit perfectly in one of the egg holder spots, but the CLUE and battery pack will need to be strapped down with a rubber band to make sure they stay put.

To make it easier to view the impact data on the CLUE's screen, mark and cut a small window in the lid. This will also allow you to access the A button on the CLUE.

Place the carton pod into the flexible rubber-band coupled space between the two balloon sets as shown here. Use the rubber bands to make sure everything is secure.

Drop Testing

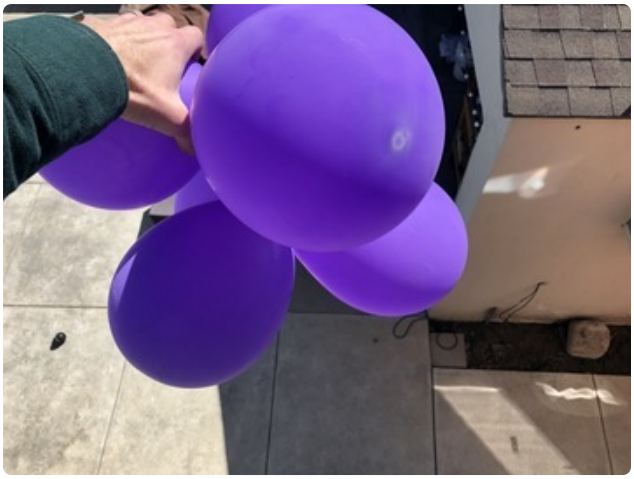
I conducted a low-height drop to see how much impact force would be generated on the CLUE. (Note, this test was done before adding the carton pod.)

From this low height, with the balloons absorbing much of the impact, a 2.5 G force impact was recorded.

By comparison, with no balloons in place the impact from this height exceeds 10.5 Gs!

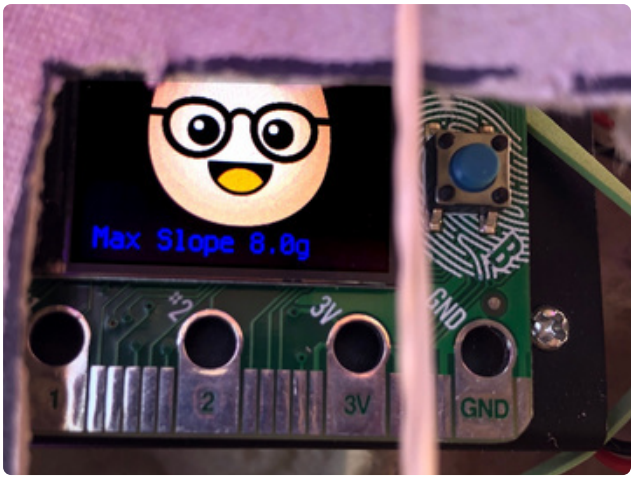

GESTURE
PROXIMITY
Max
#0
#1
0
1

LIGHT
GESTURE
PROXIMITY
Max
#0
#1
0
1



Full Test

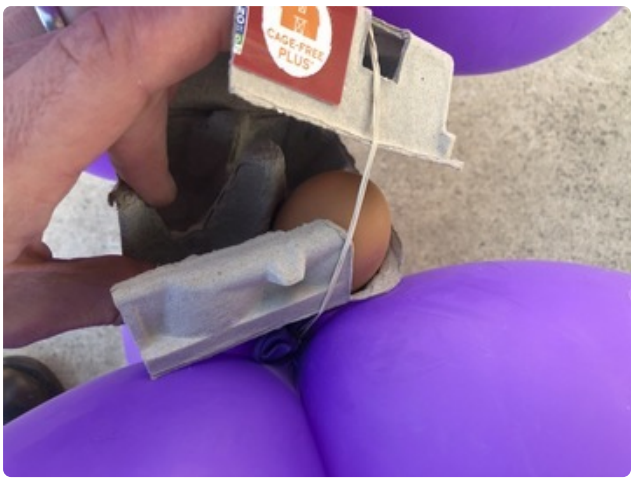
For the full test, I moved to the second story balcony. This time, the balloon landing vehicle containing the CLUE would be dropped from a height of 16 feet onto concrete!



Success!

Our CLUE survived the landing with an 8.0g impact!

Let's put our money where our mouths are and drop a real egg.



New Cargo

Swap in an actual egg for the CLUE, then place the carton pod back into the balloon space.

Repeat the drop...

...it...

...impacts...

...and...

..survives!

What new vehicle types will you construct? Try using a cardboard box, such as a black Adafruit box, and some popcorn. Does that dampen the impact enough? Remember, we're shooting for less than a 9.0g impact.

Cardboard Lander

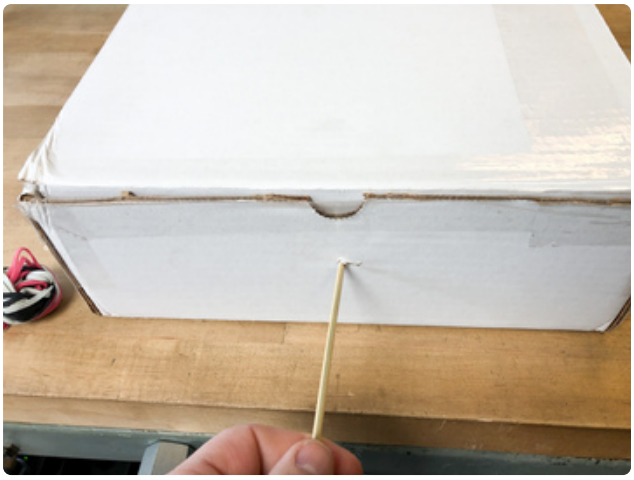
Here's another approach to landing the egg safely -- a cardboard box with shock absorbing rubber bands to secure the payload.

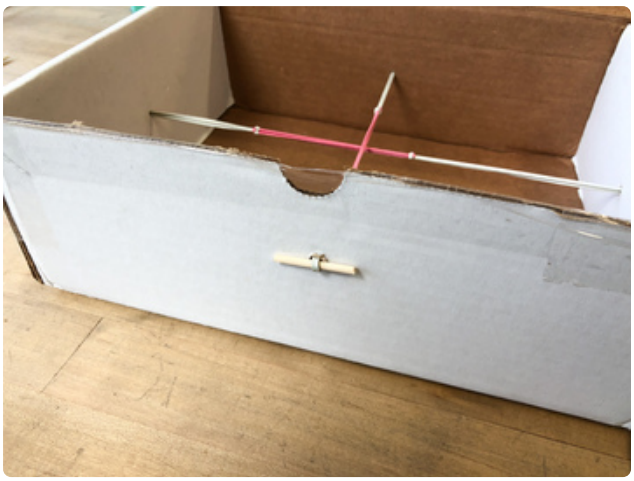
In this vehicle, when the box experiences the jerk force of hitting the ground, the egg/CLUE will slow down more gradually by converting the kinetic energy into heat as the rubber bands stretch. This can act as an effective shock absorber!

Materials

For this vehicle you'll need:

- Medium sized cardboard box
- Rubber bands
- Bamboo skewer, or similar
- Scissors or snips to cut the skewer



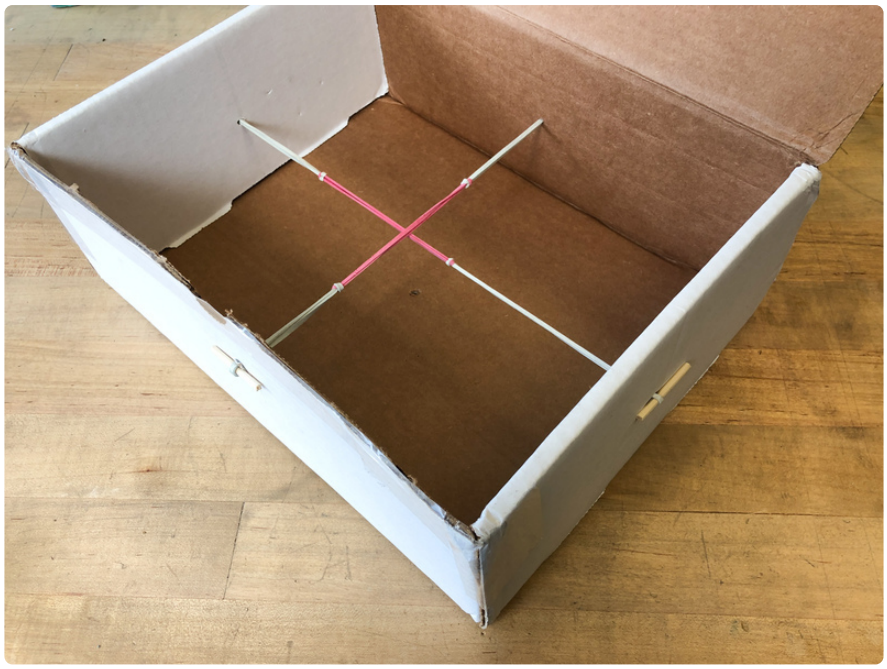


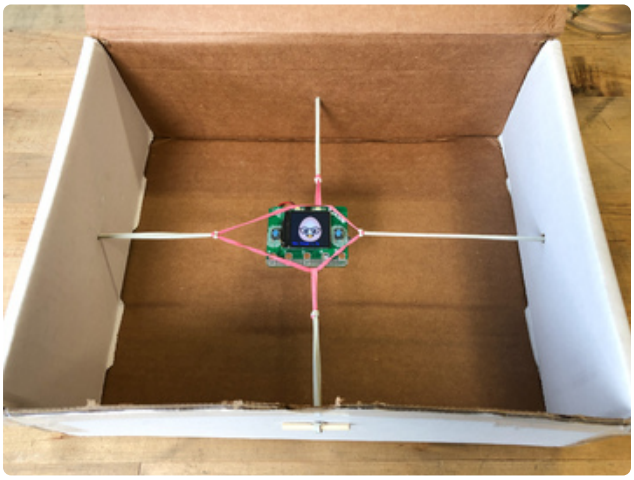
Build the Suspension Rig

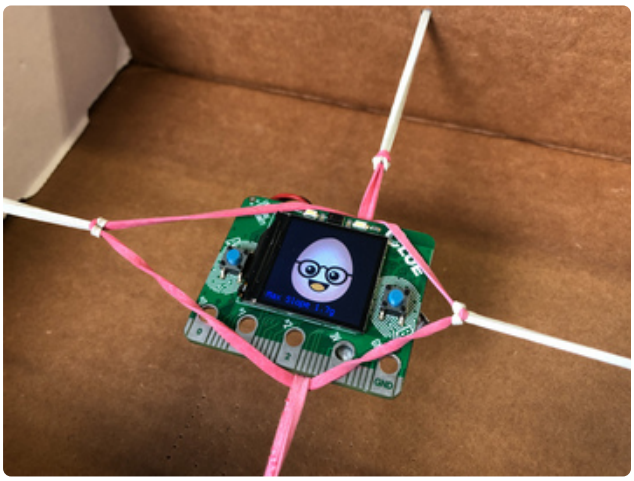
Use the skewer or an awl, nail, or screwdriver to poke a hole at the center of the box walls -- the sides, front, and back will do.

Thread a rubber band (or series of them looped together) through the holes, front to back. Use a small cutoff section of the skewer to hold them in place as shown.

Repeat for the side-to-side connection.





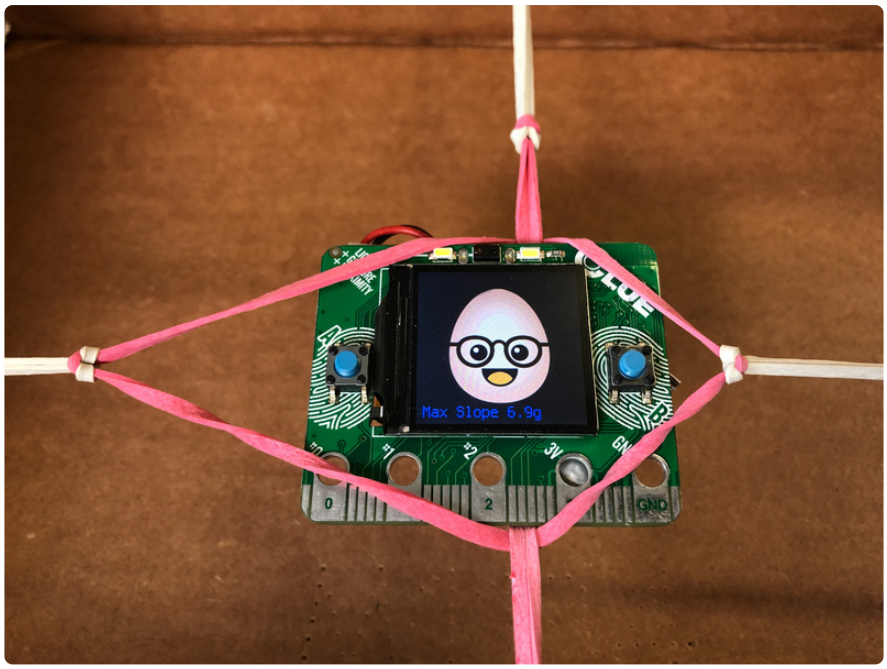


Secure the Payload

Wrap the CLUE in the rubber band intersection as shown.

Testing

Now, test the new vehicle by dropping it -- you should get a very effective reduction in g-force.



Note, a typical AAA battery pack may experience enough jerk force to momentarily jostle the spring-loaded batteries away from their contacts, thus causing a brown-out on the CLUE board. Here I'm using a LiPo battery to prevent this. You may be able to add tape to your battery pack to prevent this.

With safe, sub-9g results, it's time to graduate to a real egg! A couple more carefully applied rubber bands make for a good connection system.