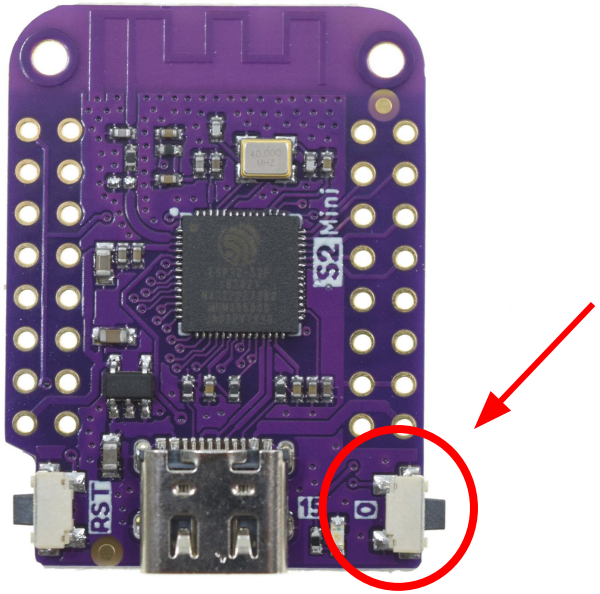
Intro to CircuitPython

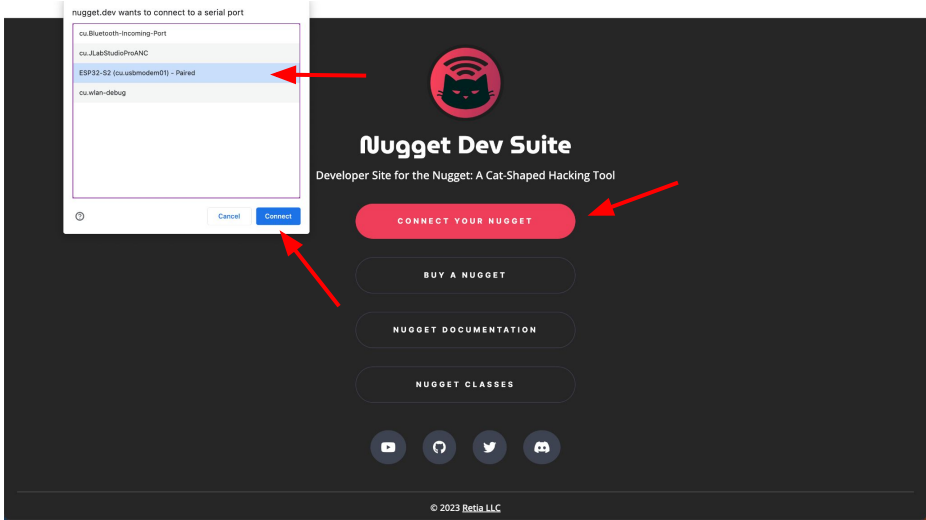
Install CircuitPython to Your Nugget on Nugget.dev

- CAREFULLY take your nugget out of its case
- On the back of your nugget, press and hold the button with the 0
- Continue holding until you plug your nugget in, then release the button



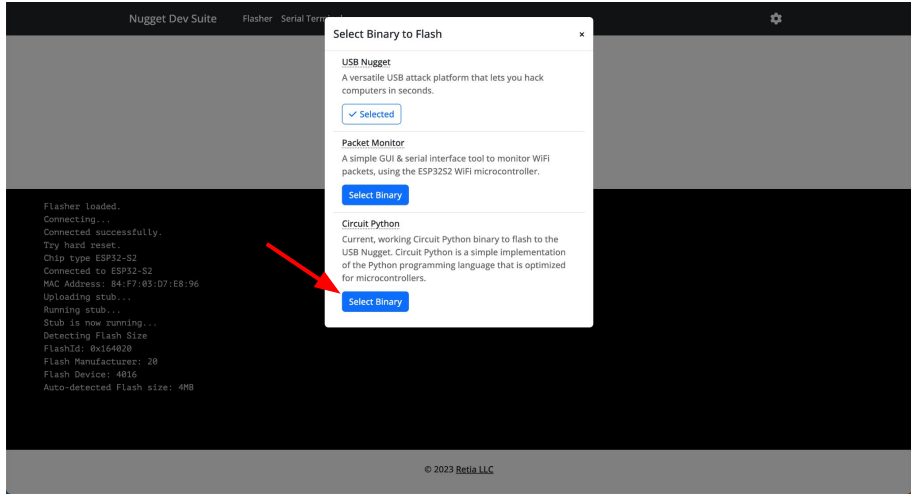
Install CircuitPython to Your Nugget on Nugget.dev

- Now, go to nugget.dev on your Chromium browser
- Click “Connect your Nugget”
- Select the device that says ESP-32-S2 and click Connect



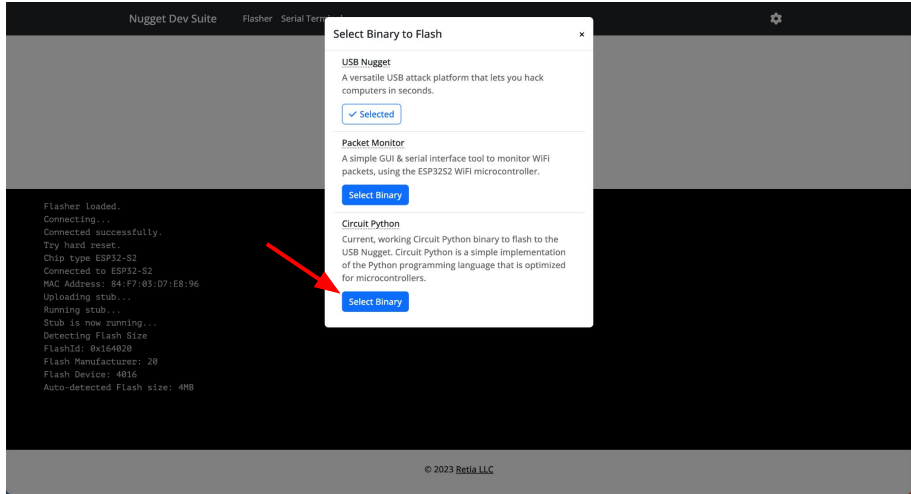
Install CircuitPython to Your Nugget on Nugget.dev

- Click the dropdown that says “USB Nugget”
- You’ll be taken to this screen. Select “Circuit Python”
- Next, hit program



Install CircuitPython to Your Nugget on Nugget.dev

- When it says “To run the new firmware, please reset your device.”, unplug and plug your nugget back in
- Congrats, you now have CircuitPython installed on the USB Nugget!

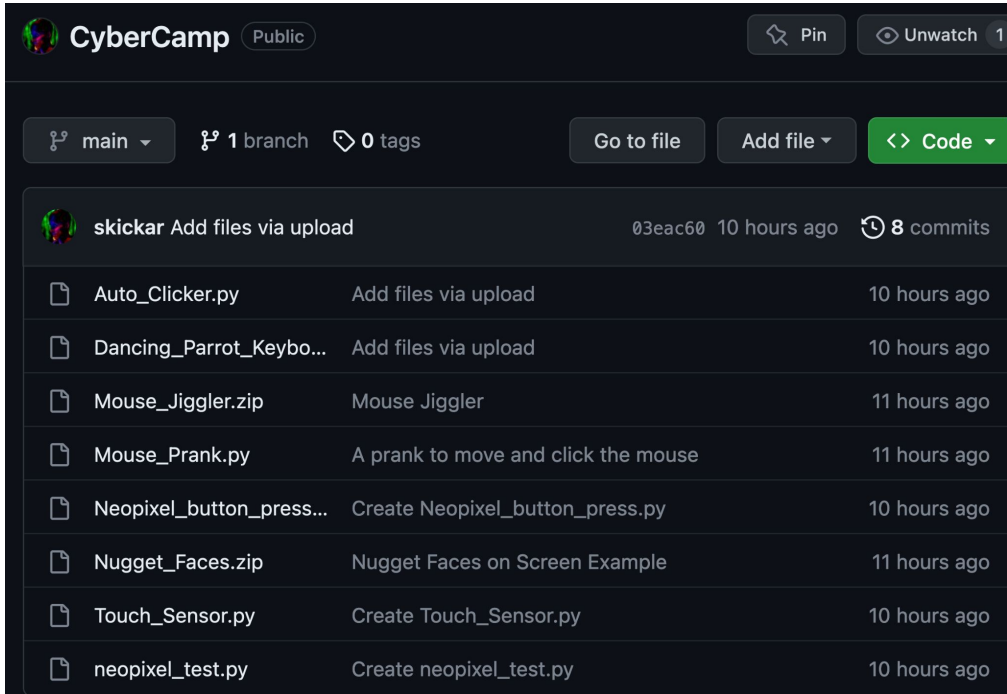


Getting Started with Thonny

Github Code Examples to Follow Along

Visit: kody.fun

Open this link in your browser
You can copy and paste code to follow along!



In Thonny, Open Code.py

Paste in the code from neopixel_test.py

Save it to run automatically!

main

CyberCamp / neopixel_test.py

skickar Create neopixel_test.py

Code Blame 3 lines (3 loc) · 89 Bytes

```
import board, neopixel
pixels = neopixel.NeoPixel(board.IO12, 1)
pixels[0] = (20, 0, 20)
```

First Example: Control a Neopixel

Let’s play with Neopixels!

main

CyberCamp / neopixel_test.py

skickar Create neopixel_test.py

Code Blame 3 lines (3 loc) · 89 Bytes

```
import board, neopixel
pixels = neopixel.NeoPixel(board.IO12, 1)
pixels[0] = (20, 0, 20)
```

Basic Commands

import [library]	Lets you quickly and easily use commands that are in the library
pixels[num] = (0,0,0)	Sets a neopixel color (Red, Green, Blue)
while True:	Makes an infinite loop
if/else	Conditional statements. Ex: If “I have apples” (run code) else (run other code)

Basic Commands

import [library]

This command lets us use code other people already wrote!

Example: If we want to make a delay, or pause our code for a few seconds, we can use the Time library.

This lets us make delays with the time.sleep() command.

In this code, we say “Hello”, wait 5 seconds, and then say “World!”

Example:

```
Import time
print(“Hello”)
time.sleep(5)
print(“world”)
```

Basic Commands

```
pixels [num] = (0,0,0)
```

This command lets you set the color of a specific neopixel! (Yes, you can have multiple neopixels in a single board)

In order to *get* the variable pixels, we need to run a command that tells python where the neopixels are. That can be done with this code: `pixels = neopixel.NeoPixel(board.IO12, 1)`

Basic Commands

```
while True:
```

Runs whatever code that's inside it forever.
It checks if true is true, which is always true.. This makes it an infinite loop!

First CircuitPython Code

- First we import the library for the board and neopixels
- Next, we save the pin that the neopixel is connected to as a variable
- Then, we create a “pixel” to control with the pin, number of pixels
- Now, we can tell the first neopixel (pixel 0) to turn any color
- Set a color by picking a Red, Green, and Blue value from 1-255

```
import board, neopixel
pixels = neopixel.NeoPixel(board.IO12, 1)
pixels[0] = (20, 0, 20)
```

Challenge: Change the code!

- Try changing the color of the neopixel
  - The three values can go anywhere from 0-255

Some possible colors:

- Green (0,255,0)
- Blue (0,0,255)
- Red (255,0,0)
- Purple (200,0,200)

Button-Controlled Neopixel

- You're going to make a bit more complicated of a program now.
- Each time you press one of the buttons, it's going to change through different colors!
- Delete all the code you already wrote and we're ready to start with something new!

Importing the Libraries

- The time library lets you to use time-related commands
- Digitalio stands for digital input/output and lets the software work with the hardware
- The board library is exactly what it sounds- it's the base library for *everything* microcontroller related
- Finally, the neopixel library is used to control the Neopixel LEDs on your Nugget!

```
import board
import digitalio
import neopixel
import time
```

Defining the Variables

```
button = digitalio.DigitalInOut(board.D5)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP
pixels = neopixel.NeoPixel(board.IO12, 1)
```

- First we define a button connected to pin D5
- Next, we tell Python that pin is an input, so we should listen to signals from it
- Now we tell the board to connect our button pin up to power. This lets us tell when the button is pressed.
- Last, we set up our neopixel on pin 12

Checking Button & Changing Neopixel Color

```
while True:  # Make it run FOREVER! (always checking)
  if not button.value:  # If the button is pressed then,
    pixels[0] = (255, 0, 0)  # Turn the LED red
    time.sleep(0.5)  # Wait before turning it off
    pixels[0] = (0, 0, 0)  # Turn the LED off
```

Putting Everything Together

```
import board, digitalio, neopixel, time
button = digitalio.DigitalInOut(board.D5)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP
pixels = neopixel.NeoPixel(board.IO12, 1)
while True:
  if not button.value:
    pixels[0] = (255, 0, 0)
    time.sleep(0.5)
    pixels[0] = (0, 0, 0)
```

Challenge: Change the code!

- Try changing the color the neopixel changes to when you press the button (200, 0, 200)
- Try changing the button we are listening for presses on (D7)
- Try making the time shorter (.5) or longer (10)
- Try making the neopixel turn RED, WHITE, and BLUE

Touch-Sensitive Light

- Are you ready for a challenge?
- Now we're going to program your neopixel to turn on whenever you touch a wire!
- Plug in a breadboard wire to the port marked "A0"
- Clear the code you've written

Open Touch_Sensor.py

- We'll use this code to follow along

main

CyberCamp / Touch_Sensor.py

skickar Create Touch_Sensor.py

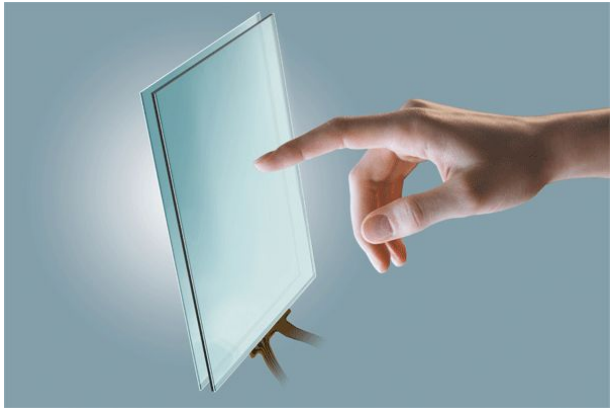
Code Blame 10 lines (10 loc) · 264 Bytes

```
import board, time, touchio, neopixel
touch_pad = board.A0
pixels = neopixel.NeoPixel(board.IO12, 1)
touch = touchio.TouchIn(touch_pad)
while True:
    if touch.value:
        pixels[0] = (255, 255, 255)
    else:
        pixels[0] = (0, 0, 0)
    time.sleep(0.1)
```

Plug a Breadboard Wire into Your Nugget

What is Capacitive Touch?

- Capacitive touch measures small changes in electrical fields
- Humans have electrical fiends around them that we can detect
- This is how smart phones know where your finger is!
- We can use a wire to detect when someone touches a wire
- We can even detect just when someone goes **near** the wire



Importing the Libraries

```
import board
import time
import touchio
import neopixel
```

Define the Variables

```
pixels = neopixel.NeoPixel(board.IO12, 1) # Tell it where the neopixel is
touch = touchio.TouchIn(board.A0) # A variable that tells us if the pin is being touched
```

Check if the wire is touched

It may be a lot of code but don't let it scare you!

```
while True:
  if touch.value: # It will run this first bit of code if it's being touched
    pixels[0] = (255, 255, 255)  # Turn on NeoPixel when touched
  else: # It will run this code, though, if it isn't being touched
    pixels[0] = (0, 0, 0)  # Turn off NeoPixel when not touched
  time.sleep(0.1) # make it wait a bit so it's not overloaded
```

Putting it all together

```
import board, time, touchio, neopixel
pixels = neopixel.NeoPixel(board.IO12, 1)
touch = touchio.TouchIn(board.A0)
while True:
  if touch.value:
    pixels[0] = (255, 255, 255)
  else:
    pixels[0] = (0, 0, 0)
  time.sleep(0.1)
```

Challenge: Change the code!

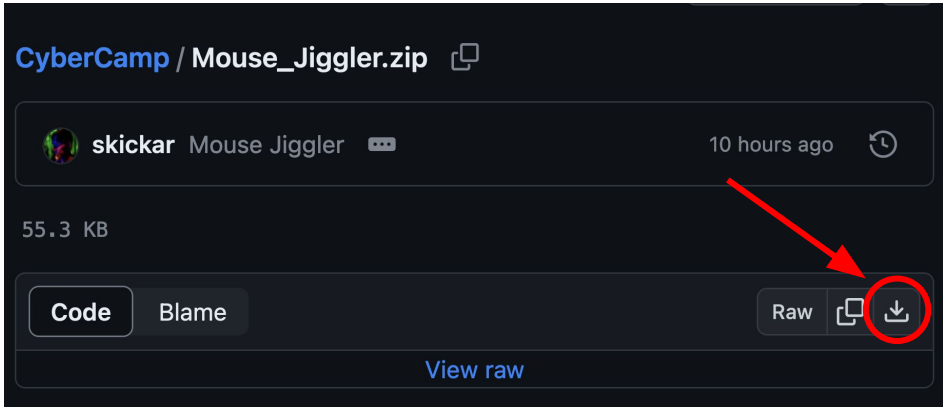
- Try changing the color the neopixel turns when the wire is touched
- Try making a few colors play with a time delay between them when the wire is touched
- Can you make it so the light only turns off when the wire is touched?

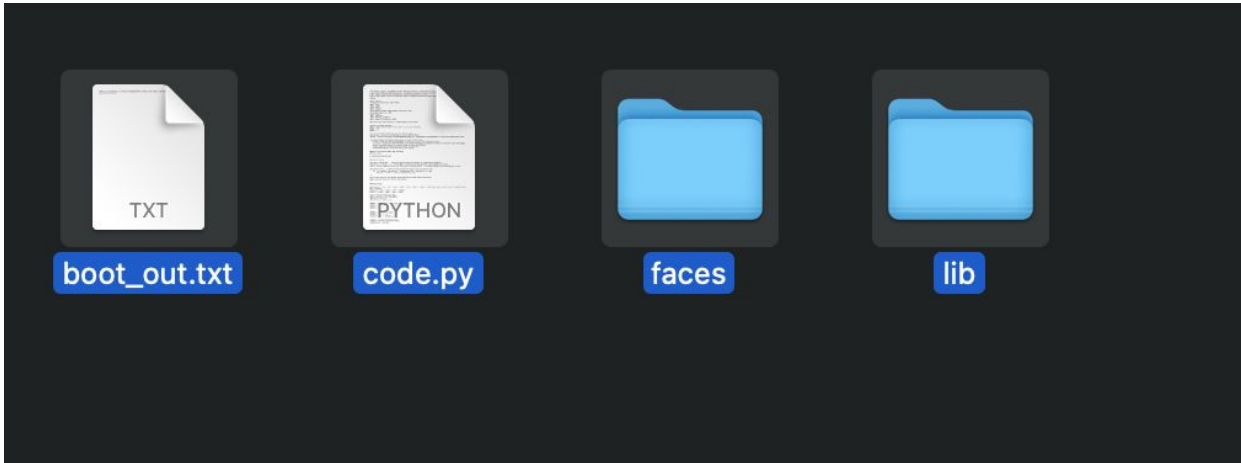
Mouse Jiggler

- Next we'll do a more advanced project
- This is the Nugget Mouse Jiggler
- Let's install it!

Download Mouse_Jiggler

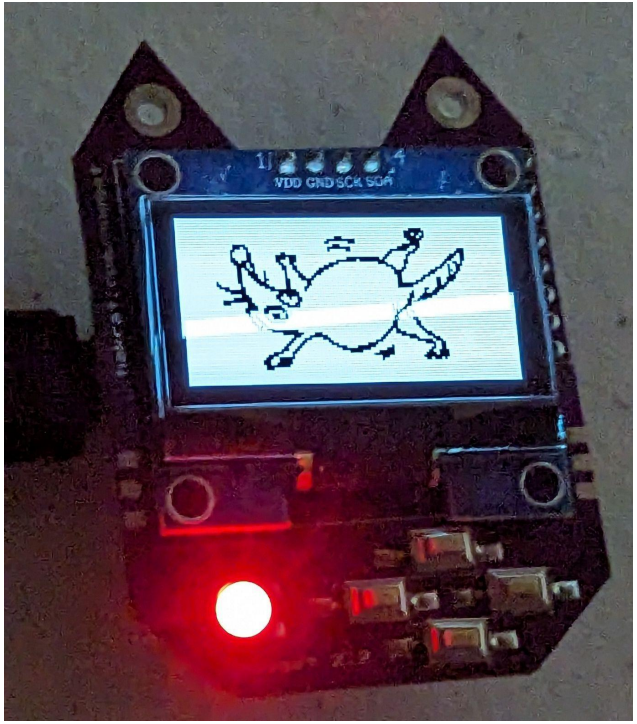
- Open the ZIP file
- Drag and drop these files to your Nugget





Your Jiggler Should Turn On!

- Open the ZIP file
- Drag and drop these files to your Nugget



Challenge: Change the code!

- Make the Nugget jiggle more often: Set JiggleFactor to a low number
- Make the mouse jiggle further: Increase the JitterFactor to a bigger number
- Get Jiggling!

```
WeJiggling = False ##
JiggleFactor = 150 ##
JitterFactor = 20 ## H
JigglingNug = ["/faces
```

Auto-Clicker!

- Now we'll make an auto-clicker!
- We will hold down the "down" button to trigger it
- We can use the right mouse or the left mouse
- We'll use a "While True" loop to make it run forever

Open Auto_Clicker.py

- Copy the code over from the file into code.py
- Hit save, restart the nugget, and watch it click away!

main CyberCamp / Auto_Clicker.py

skickar Add files via upload

Code Blame 16 lines (13 loc) · 347 Bytes

```python
import time
import board
import digitalio
import usb_hid
from adafruit_hid.mouse import Mouse

button = digitalio.DigitalInOut(board.D3)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP

mouse = Mouse(usb_hid.devices)

while True:
    if not button.value:
        mouse.click(Mouse.LEFT_BUTTON)
        time.sleep(0.1)
```

Writing the Clicker

```
import time, board, digitalio, usb_hid
from adafruit_hid.mouse import Mouse

button = digitalio.DigitalInOut(board.D3)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP
mouse = Mouse(usb_hid.devices)

while True:
        if not button.value:
        mouse.click(Mouse.LEFT_BUTTON)
        time.sleep(0.1)
```

Challenge: Change the code!

- Try changing the mouse click to the right mouse button
- Make the delay between clicks longer or shorter
- How fast can you make it click?
- Can you make it click forever?