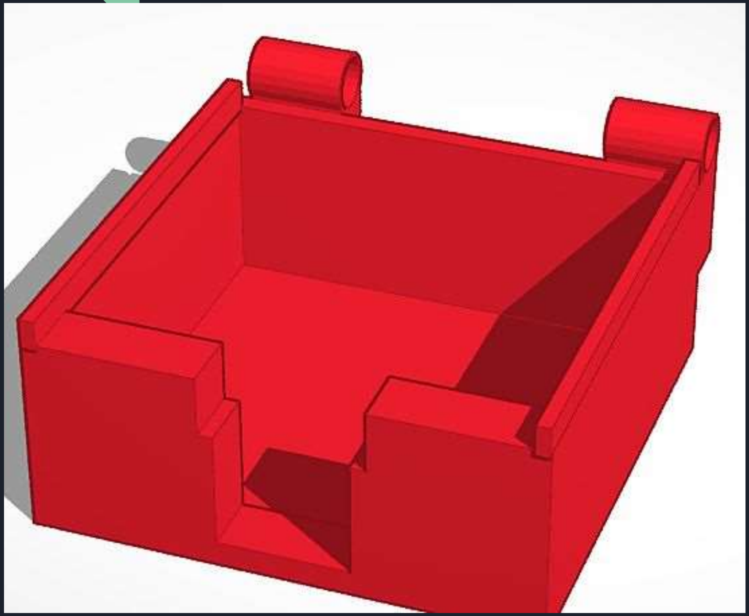
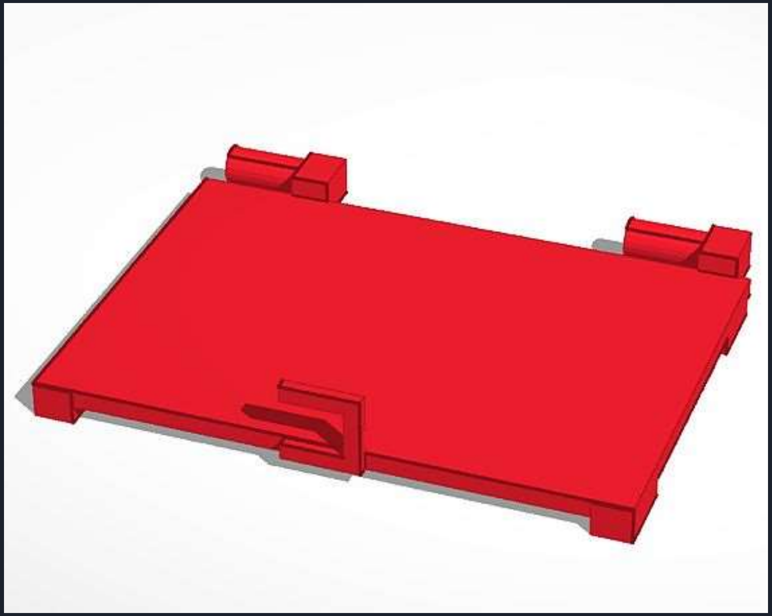
STEP 1: 3D printing the box

First you'll need to download the files of the box as well as the cover (the type of color filament doesn't matter). When downloading the 3D design make sure to download it as an stl file



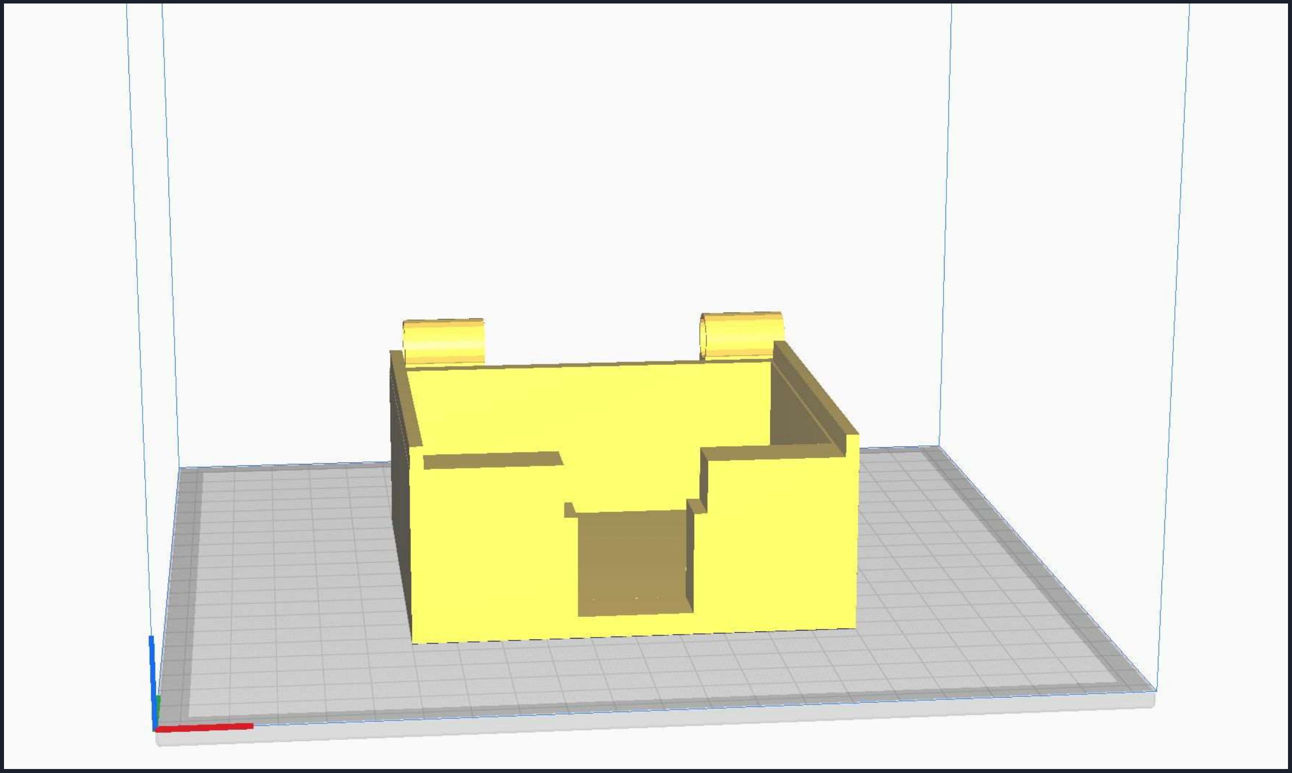


https://www.tinkercad.com/things/jjDmHgweiZ8

https://www.tinkercad.com/things/62ISlmLdmn5

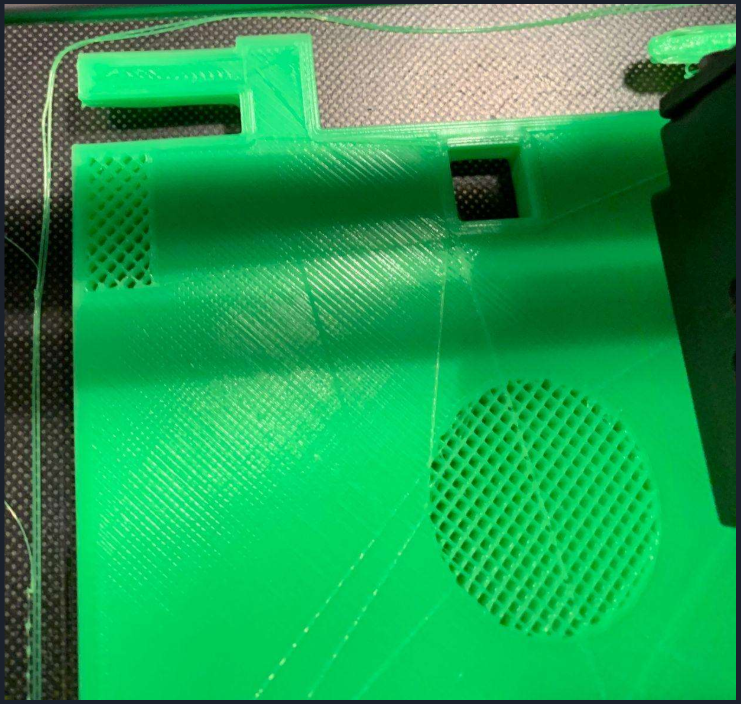
STEP 1: 3D printing the box

After downloading the files, place them in whatever 3D printer software that you use to turn them into 3D files. Do not adjust them as it most likely will screw up the sizes of the axles and their sockets. Figure out whether or not to use supports on the design. For this design there are a couple overhanging parts so it is smart to add supports just in case.



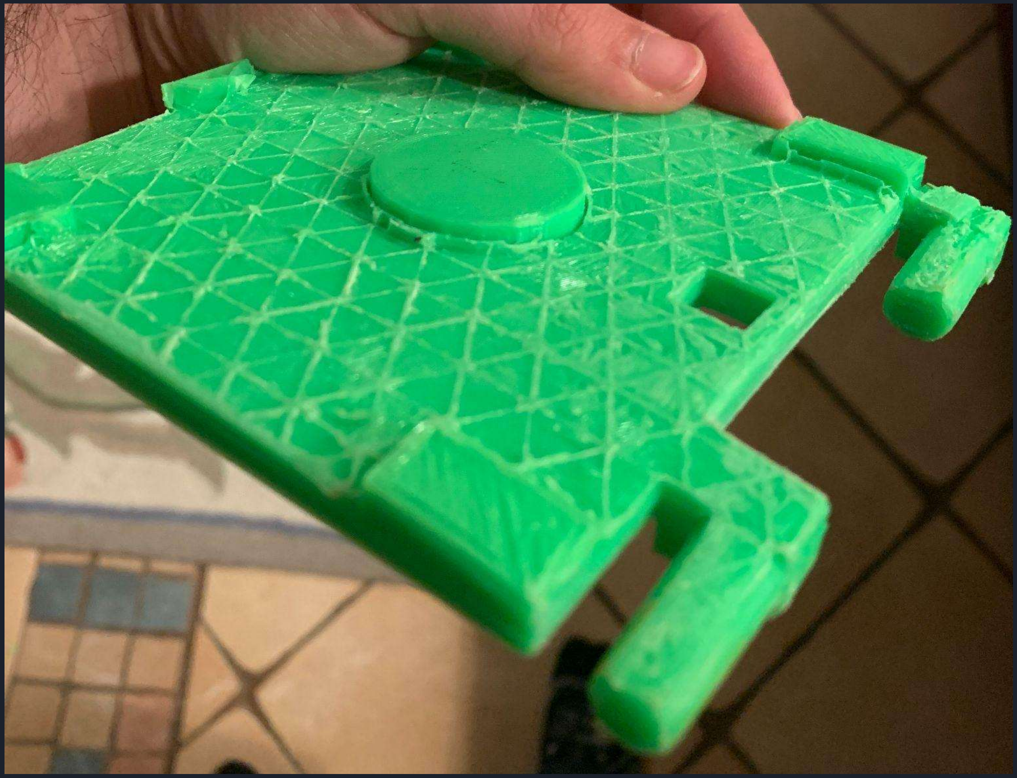
STEP 1: 3D printing the box

Third is that you'll wanna print both of them separately as they are both rather large objects that will take up the entire 3D printer bed (or at least mine did). When printing the two parts separately is especially important to make sure all of the printer settings are the same. If something is different it make cause the two parts to not work together.



STEP 2: cleaning the box and its cover

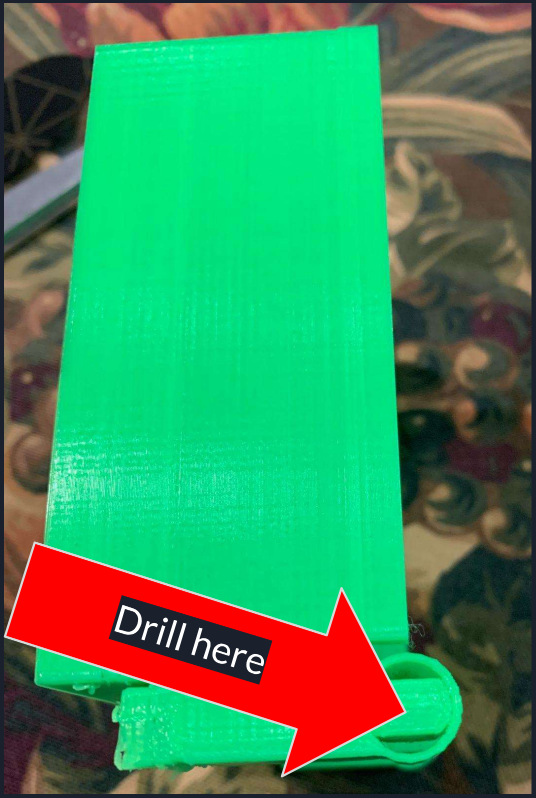
For the cover, you wanna remove filament from from on top and on the joints. The joints will need some extra care as to make sure that the axles can still move freely inside the box (it be preferable to use some sandpaper).

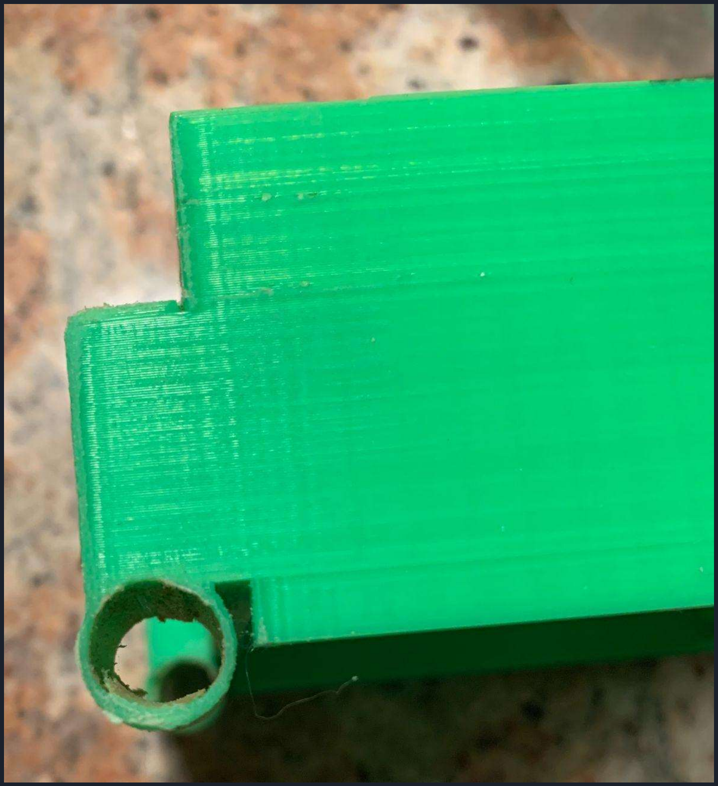


STEP 2: cleaning the box and its cover

You’ll will wanna use the drill bit to clean out the excess filament inside the joints and battery box

Note: go slowly with the drill as the pivot point on the box is rather thin and could break if handled improperly





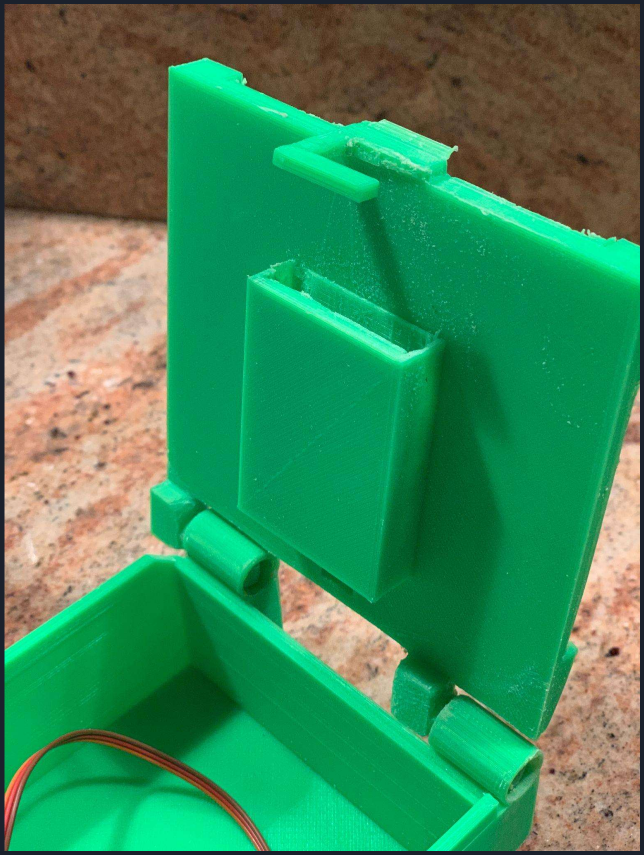
STEP 2: cleaning the box and its cover

Use sandpaper or a file to remove any excess plastic, or to smooth the surfaces and the joints (the more durable the better). Nobody wants to get cut while handling the product.

Something like this worked for me.

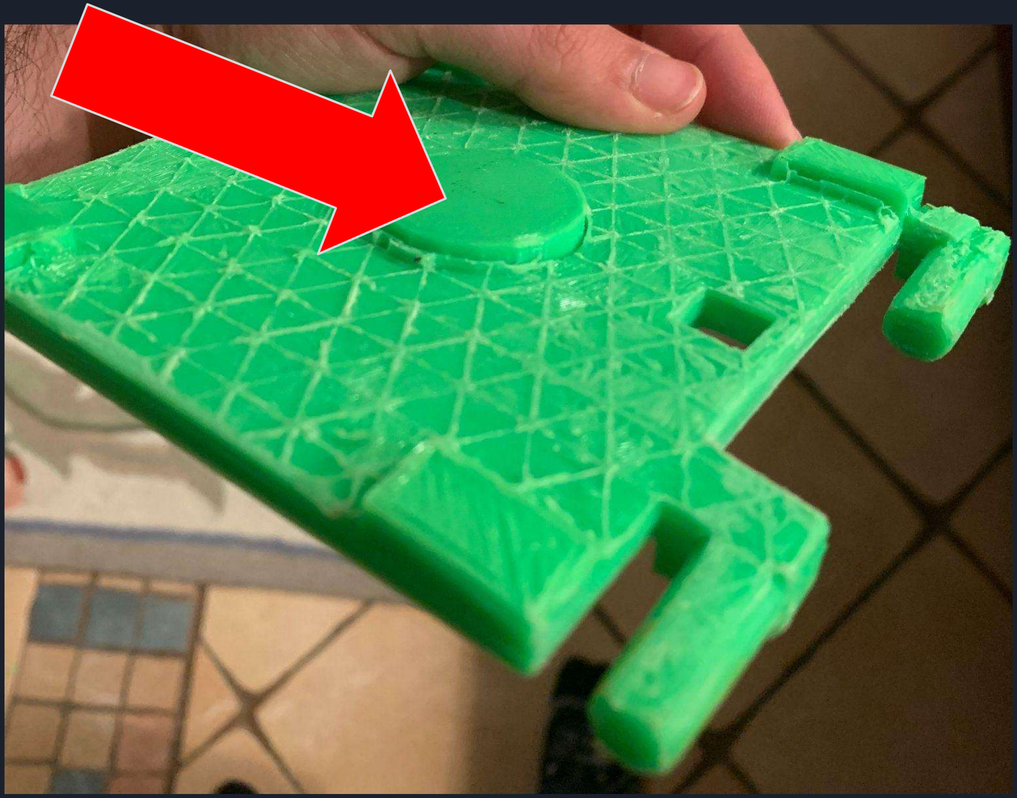
STEP 3: setting up the locked box

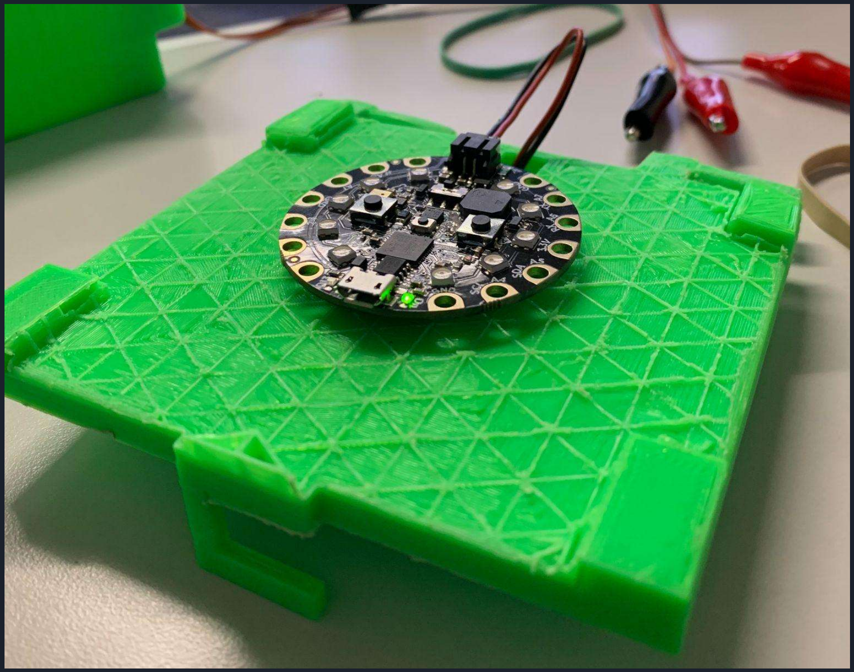
Place the box cover’s axles inside the holes on the back of the box and test to make sure the box can move without issue



STEP 3: setting up the locked box

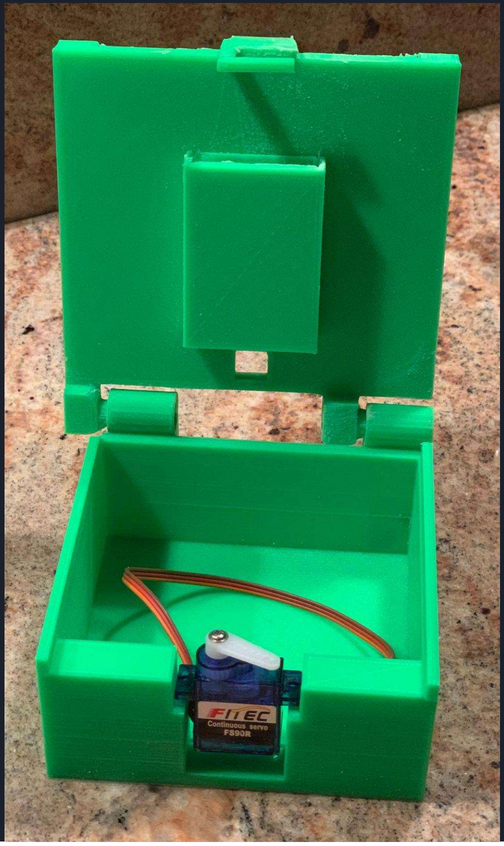
Place some tape or glue on top of the circle on the cover and place the CPX on top. Make sure the CPX battery connecter faces the hole in the back.

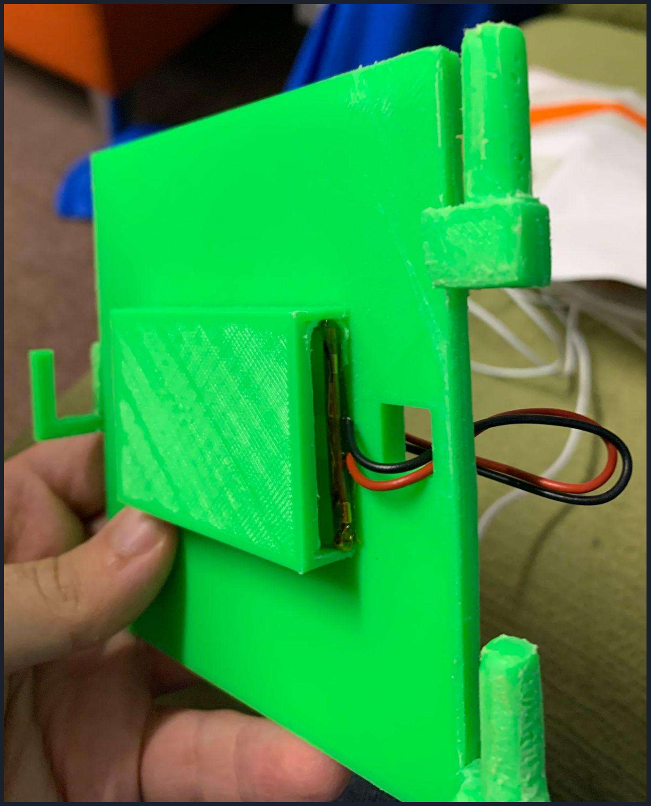




STEP 3: setting up the locked box

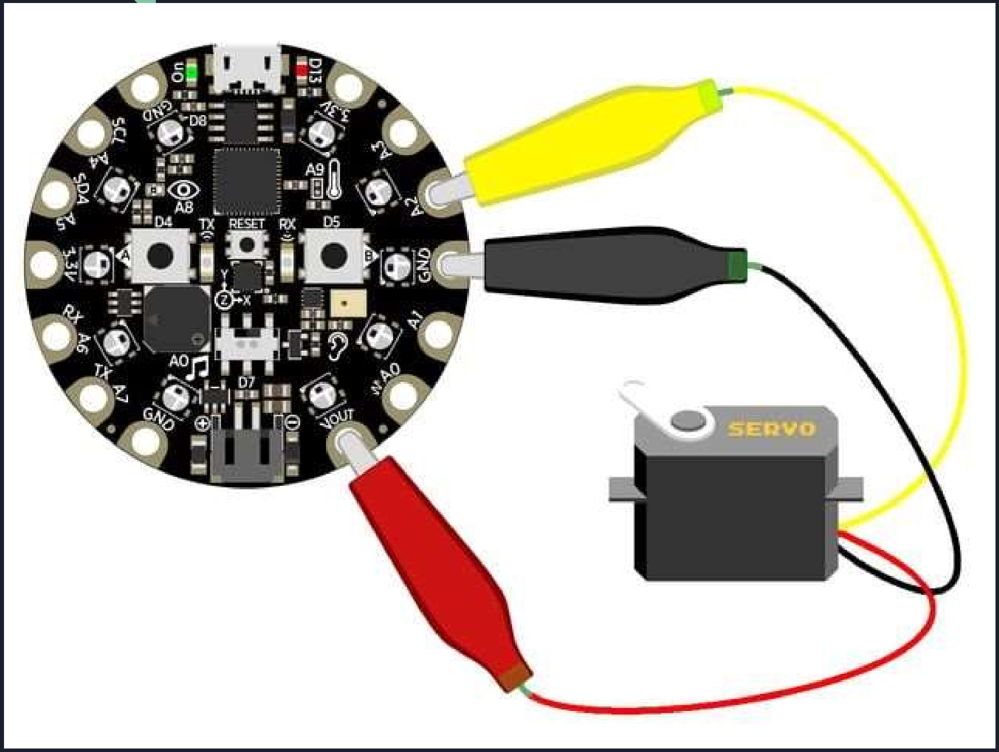
Place the battery inside the slit on the underside of the cover and feed the cable through the hole in the cover and connect it to the CPX. Then place the servo in the hole in the front of the box (you can use tape, glue or screws to hold it in place if you feel like it)



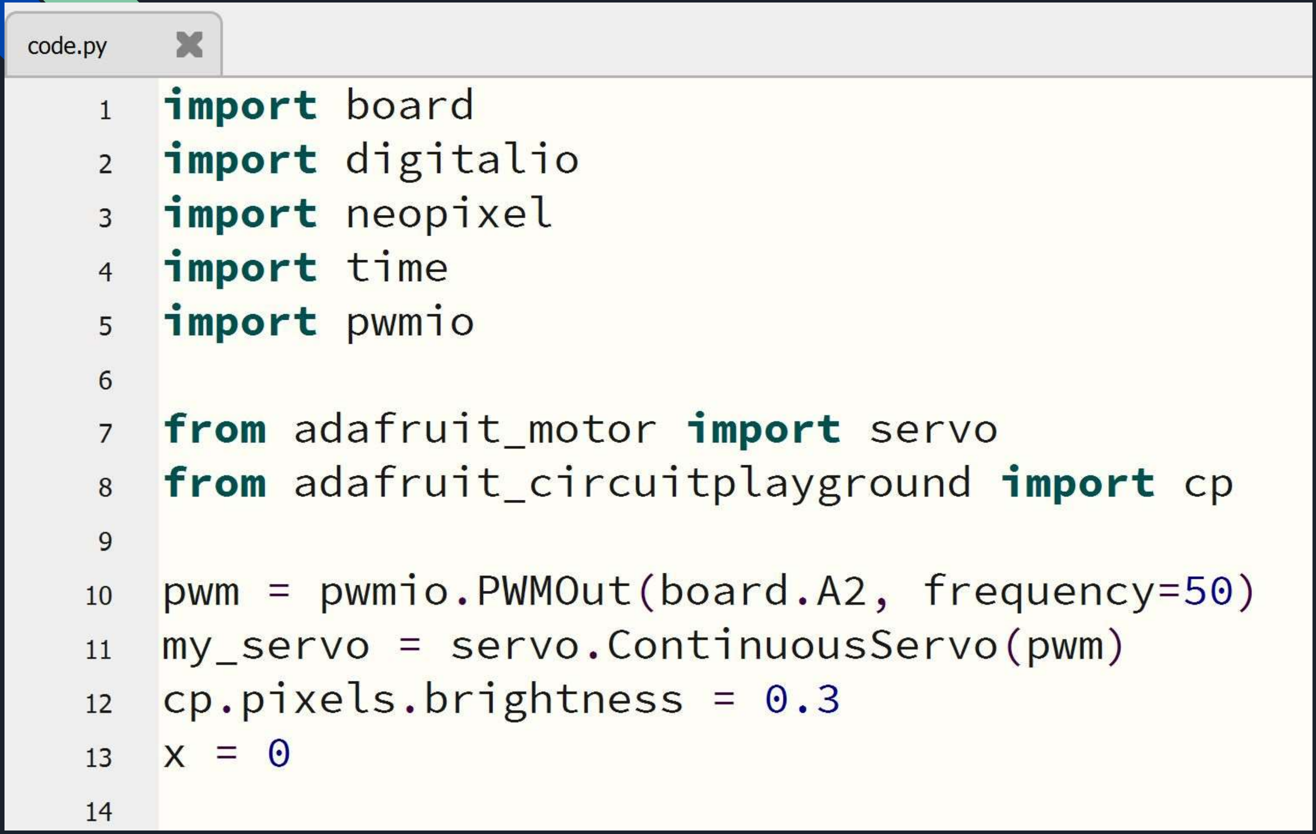


Step 4: How to connect the servo

Run the cords through the space between the cover and the box in the back. Follow the picture below in order to connect alligator cables that connect to the servo to the CPX (black is brown, red is red and yellow is orange for reference).



Step 5: How to write the code



```
1   import board
2   import digitalio
3   import neopixel
4   import time
5   import pwmio
6
7   from adafruit_motor import servo
8   from adafruit_circuitplayground import cp
9
10  pwm = pwmio.PWMOut(board.A2, frequency=50)
11  my_servo = servo.ContinuousServo(pwm)
12  cp.pixels.brightness = 0.3
13  x = 0
14
```

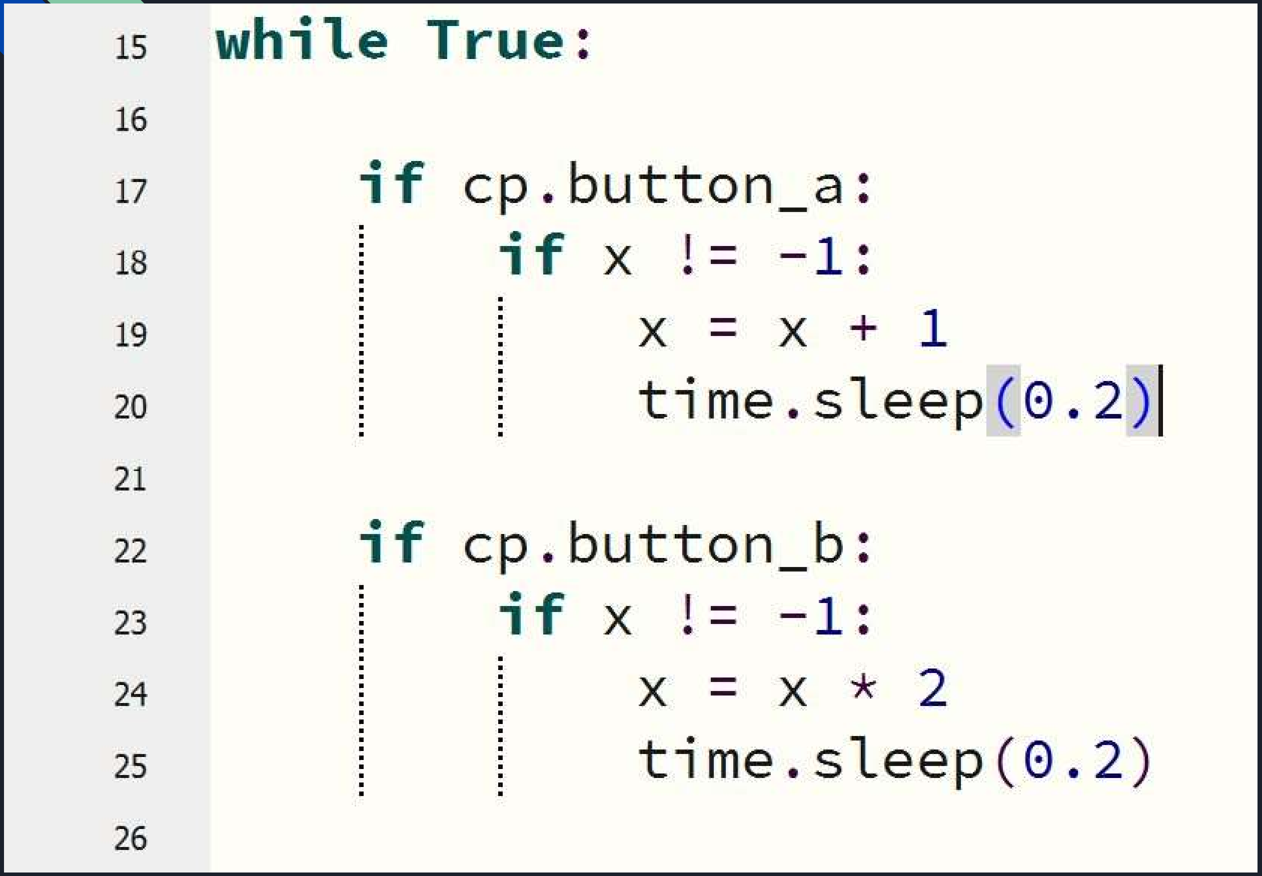
These are the imports and beginning variables that one will need before they begin actually writing the code

- Lines 1 through 5 are importing the key words for the:
  - CPX board
  - The buttons and the switch
  - The lights
  - Time
  - The servo

Lines 7 and 8 are importing the whole servo and the whole CPX so they are able to be talked to

Lines 10 through 13 are the variables that need to be set before the while loop. This is because these variables will be changed and can't be constantly reset.

Step 5: How to write the code



```
while True:

    if cp.button_a:
        if x != -1:
            x = x + 1
            time.sleep(0.2)

    if cp.button_b:
        if x != -1:
            x = x * 2
            time.sleep(0.2)

```

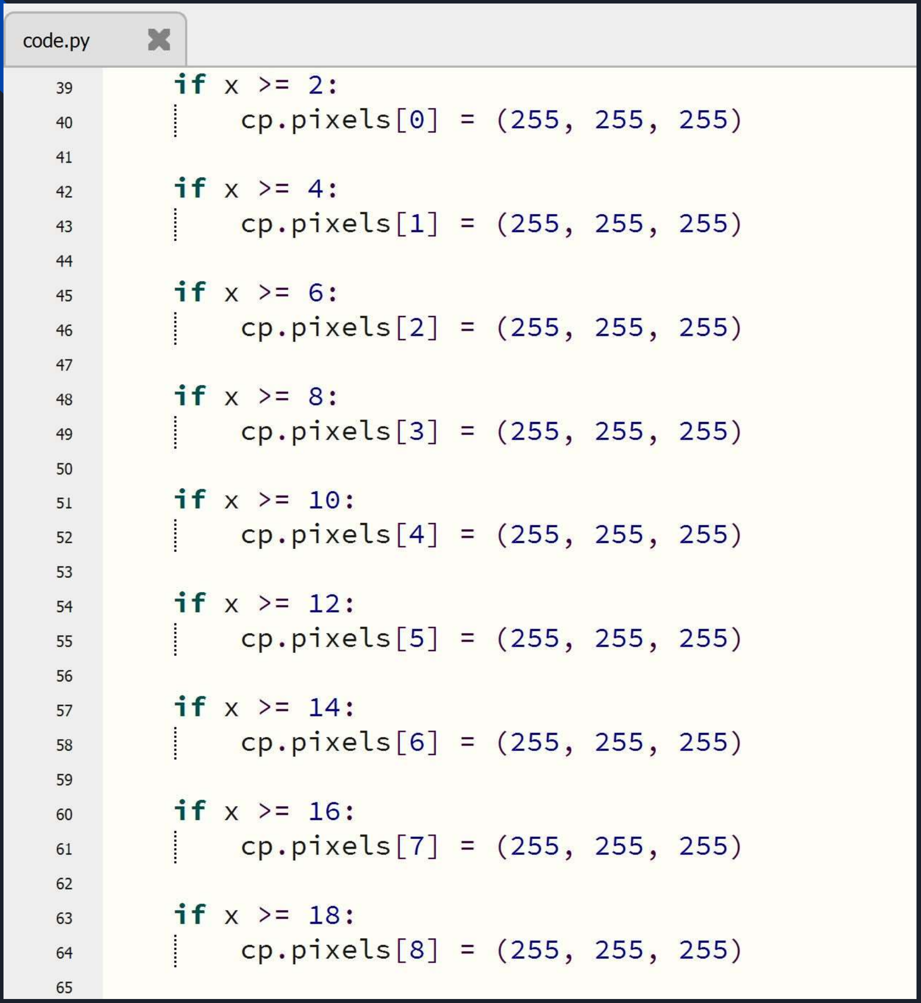
This is code for when button a is clicked or when button b is clicked

When button a is clicked the variable x will be added by one

When button b is clicked variable x will be multiplied by two

Time sleep is to give time to the user to take their finger off the button

Step 5: How to write the code



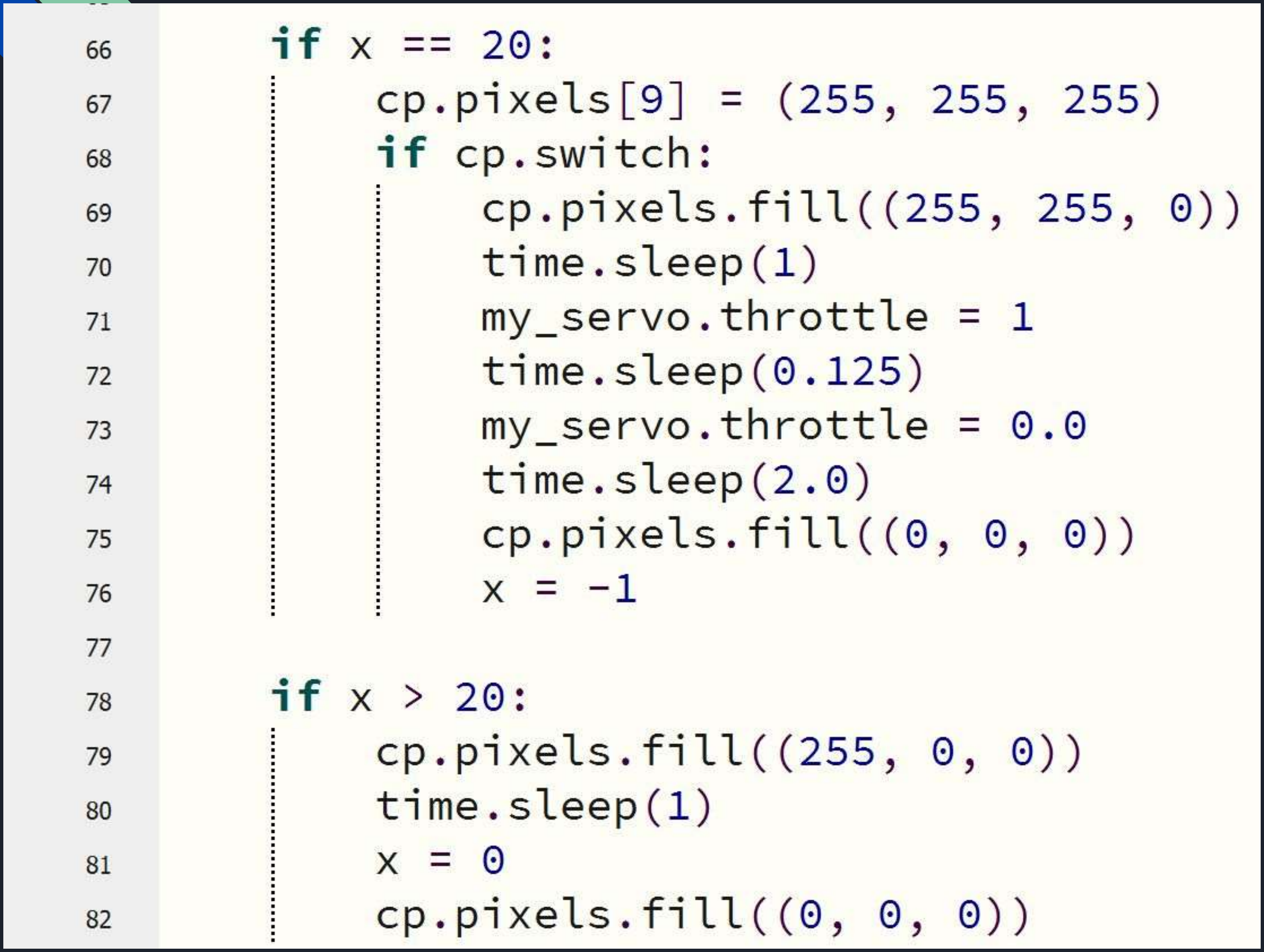
code.py

```
    if x >= 2:
        cp.pixels[0] = (255, 255, 255)

    if x >= 4:
        cp.pixels[1] = (255, 255, 255)

    if x >= 6:
        cp.pixels[2] = (255, 255, 255)

    if x >= 8:
        cp.pixels[3] = (255, 255, 255)

    if x >= 10:
        cp.pixels[4] = (255, 255, 255)

    if x >= 12:
        cp.pixels[5] = (255, 255, 255)

    if x >= 14:
        cp.pixels[6] = (255, 255, 255)

    if x >= 16:
        cp.pixels[7] = (255, 255, 255)

    if x >= 18:
        cp.pixels[8] = (255, 255, 255)
```

This code is for the light-up progress bar on the CPX

The closer the user gets to the correct code a progress bar will appear using the lights on the CPX

Step 5: How to write the code



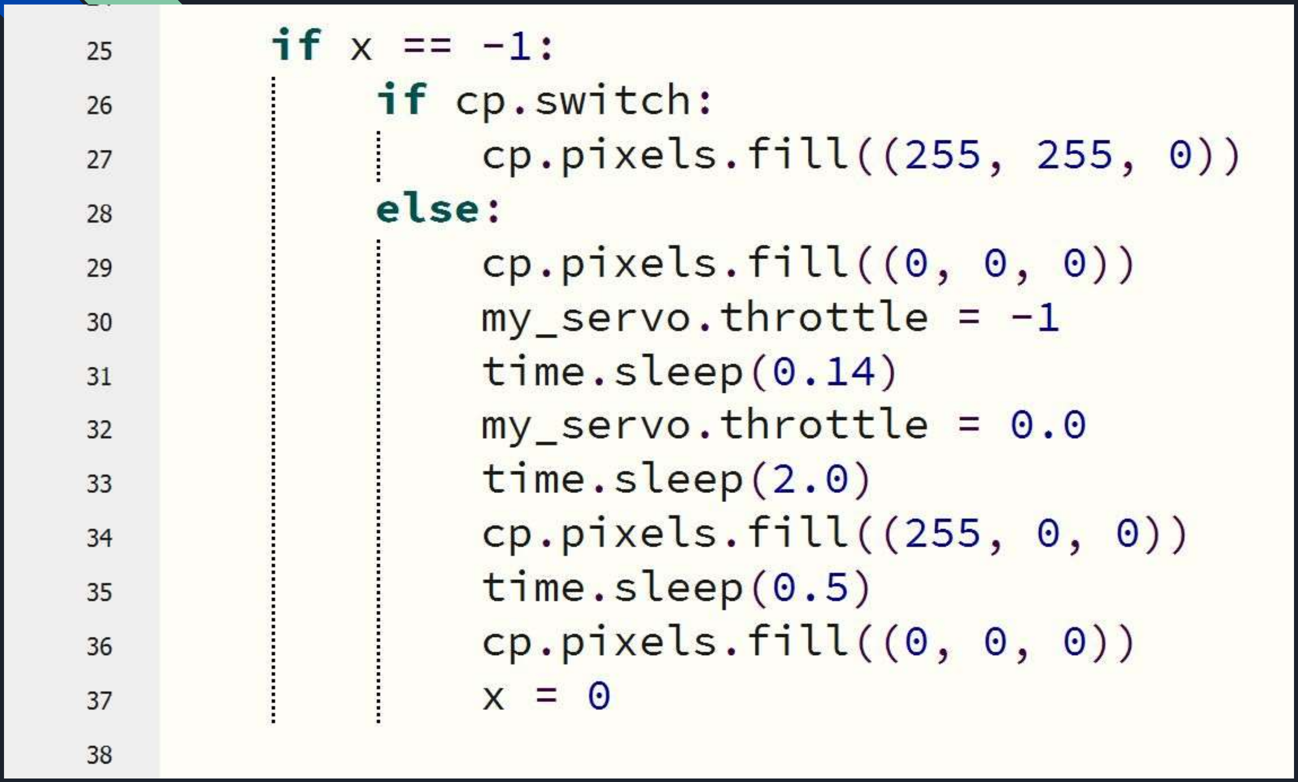
```
    if x == 20:
        cp.pixels[9] = (255, 255, 255)
        if cp.switch:
            cp.pixels.fill((255, 255, 0))
            time.sleep(1)
            my_servo.throttle = 1
            time.sleep(0.125)
            my_servo.throttle = 0.0
            time.sleep(2.0)
            cp.pixels.fill((0, 0, 0))
            x = -1

    if x > 20:
        cp.pixels.fill((255, 0, 0))
        time.sleep(1)
        x = 0
        cp.pixels.fill((0, 0, 0))
```

If the user gets the correct answer (20) all of the lights will be white, signaling to the user that all they have to do is flick the switch and the box will unlock

Once the switch is moved to the left the lights will become yellow the servo will move and the box will unlock and x will be set to -1

If the user goes past the correct answer the lights will flash red and x will be reset back to 0

Step 5: How to write the code



```
    if x == -1:
        if cp.switch:
            cp.pixels.fill((255, 255, 0))
        else:
            cp.pixels.fill((0, 0, 0))
            my_servo.throttle = -1
            time.sleep(0.14)
            my_servo.throttle = 0.0
            time.sleep(2.0)
            cp.pixels.fill((255, 0, 0))
            time.sleep(0.5)
            cp.pixels.fill((0, 0, 0))
            x = 0
```

This is code to lock the box again once the user has closed it

- The lights will stay yellow until the switch is to the left
- The buttons will not be able to do anything while the switch is to the left
- Once the user is ready to lock the box again they will flick the switch to the right.
- The lights will flash red indicating that the box has been locked and everything will be able to be used again

STEP 6: test

After building your locked box all you need to do is test to make sure that the code works. Once that is complete CONGRATULATIONS you completed the project