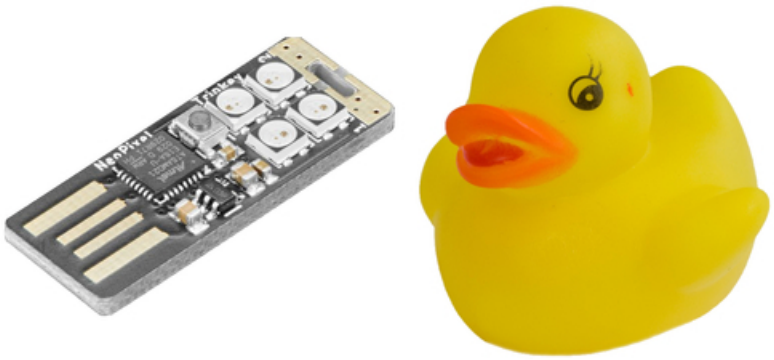
adafruit learning system

Neo Trinkey CircuitPython Rubber Ducky

Created by Dylan Herrada



Last updated on 2021-10-10 12:30:14 PM EDT

Guide Contents

Guide Contents 2
Overview 3
Parts 4
Project Code 5
Usage 6
Non-US Keyboard Layouts 7
Code Run-Through 7
Using the Rubber Ducky 8

Overview

In this project, you'll learn how to use your Neo Trinkey and the new Adafruit CircuitPython Ducky (https://adafru.it/Ue8) library to run a DuckyScript file. DuckyScript (https://adafru.it/Ue9) is a language that can be used on USB rubber duckies.

Rubber duckies are devices that you can plug into a computer to send a predefined set of keystrokes. They are usually considered hacking tools, and there's no question that if you have physical access to a computer, rubber duckies are an extremely fast and efficient way to do something, but they can also be very useful for running macros to save you time.

This project will turn your Trinkey into one of those rubber duckies. This project can be done with any of the Trinkeys, but I'd personally recommend the Neo Trinkey (https://adafru.it/Uea) or the NeoKey Trinkey (https://adafru.it/Ueb). This guide will use the Neo Trinkey, but assuming you already have CircuitPython installed, the instructions and code should be exactly the same.

Disclaimer: This project is for fun. Don't hack people unless they say you can.

Parts

Adafruit Neo Trinkey - SAMD21 USB Key with 4 NeoPixels

It's half USB Key, half Adafruit Trinket...it's Neo Trinkey, the circuit board with a Trinket M0 heart and four RGB NeoPixels for customizable...

$6.95

In Stock

Add to Cart

Your browser does not support the video tag.

Adafruit NeoKey Trinkey - USB NeoPixel Mechanical Key Switch

It's half USB Key, half Adafruit Trinket, half mechanical keeb...it's NeoKey Trinkey, the circuit board with a Trinket M0 heart, a NeoPixel glow, and...

$6.95

In Stock

Add to Cart

Project Code

This guide assumes you already have CircuitPython installed on your Trinkey. The Trinkey ships with CircuitPython pre-installed, so if you have put Arduino on it or need to re-install CircuitPython, check out this guide (https://adafru.it/Ues).

This project only uses built-in libraries, so all you have to do is plug your Trinkey into your computer, click **Download Project Bundle** below and unzip it.

Now, copy **code.py** and **duckyscript.txt** over to your Trinkey and you should be all set.

You can use the script included in the project bundle which can be used to enroll a Chromebook, or you can make your own. You can find the syntax and available commands (https://adafru.it/Ue9) and a bunch of example scripts (https://adafru.it/Uet) in the wiki.

```
# SPDX-FileCopyrightText: Copyright (c) 2021 Dylan Herrada for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense
import time

import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
import adafruit_ducky

import touchio # pylint: disable=unused-import
import board
import neopixel
from digitalio import DigitalInOut, Pull # pylint: disable=unused-import

# Uncomment for Neo Trinkey
touch1 = touchio.TouchIn(board.TOUCH1)
touch2 = touchio.TouchIn(board.TOUCH2)

# Uncomment for NeoKey Trinkey
#button = DigitalInOut(board.SWITCH)
#button.switch_to_input(pull=Pull.DOWN)
#button_state = False

pixels = neopixel.NeoPixel(board.NEOPIXEL, 4)

pixels.fill((0xFFFFFF))

time.sleep(1)  # Sleep for a bit to avoid a race condition on some systems
keyboard = Keyboard(usb_hid.devices)
keyboard_layout = KeyboardLayoutUS(keyboard)  # We're in the US :)

duck = adafruit_ducky.Ducky("duckyscript.txt", keyboard, keyboard_layout)

result = True
running = False
while result is not False:
    #if button.value: # Uncomment for NeoKey Trinkey
    if any([touch1.value, touch2.value]): # Uncomment for Neo Trinkey
        running = not running
        if running:
            pixels.fill((0x00FF00))
        else:
            pixels.fill((0xFF0000))
        time.sleep(0.2)
    if running:
        result = duck.loop()
```

Usage

To use this device, simply plug it into the target computer and press the key switch. You can find a number of other scripts and attacks here:

https://adafru.it/Uet

https://adafru.it/Uet

Non-US Keyboard Layouts

A library of non-US keyboard layouts is available (https://adafru.it/UYD) if you want to adapt this for other keyboards.

Code Run-Through

First, the code imports the required libraries.

```
import time

import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
import adafruit_ducky

import touchio # pylint: disable=unused-import
import board
import neopixel
from digitalio import DigitalInOut, Pull # pylint: disable=unused-import
```

The code next sets up the buttons used to control the script and the NeoPixels used to indicate the status of the script.

```
# Uncomment for Neo Trinkey
touch1 = touchio.TouchIn(board.TOUCH1)
touch2 = touchio.TouchIn(board.TOUCH2)

# Uncomment for NeoKey Trinkey
#button = DigitalInOut(board.SWITCH)
#button.switch_to_input(pull=Pull.DOWN)
#button_state = False

pixels = neopixel.NeoPixel(board.NEOPIXEL, 4)

pixels.fill((0xFFFFFF))
```

Then, the code sets up the objects required to emulate a keyboard.

```
time.sleep(1)  # Sleep for a bit to avoid a race condition on some systems
keyboard = Keyboard(usb_hid.devices)
keyboard_layout = KeyboardLayoutUS(keyboard)  # We're in the US :)
```

Before the ducky script is run, the code needs to initialize the object it will use to send the keystrokes in the script.

```
duck = adafruit_ducky.Ducky("duckyscript.txt", keyboard, keyboard_layout)
```

Finally, the code runs the script. It checks if the control button has been pressed and if it has, it runs the loop. If the button is pressed again, the script is paused.

```
while result is not False:
    #if button.value: # Uncomment for NeoKey Trinkey
    if any([touch1.value, touch2.value]): # Uncomment for Neo Trinkey
        running = not running
        if running:
            pixels.fill((0x00FF00))
        else:
            pixels.fill((0xFF0000))
        time.sleep(0.2)
    if running:
        result = duck.loop()
```

Using the Rubber Ducky

To use it, simply plug it into a computer and press the control button to run it. You'll know it's running when the NeoPixels turn green. When the script is done running, the NeoPixels will return to being white.