adafruit learning system

No-Touch Hand Wash Timer for Circuit Playground Express and CLUE

Created by John Park

Last updated on 2020-04-05 06:57:35 PM EDT

Overview

This guide will show two ways to build a hand washing timer. Bored of singing "Happy Birthday"? This countdown timer will help you scrub your hands with soap for a full 20 seconds with confidence, as outlined by the US Centers For Disease Control and Prevention handwashing guidelines (https://adafru.it/JRB).

Plus, there's no need to touch the timer, in order keep it clean.

Two different version of the project here -- one in MakeCode for Circuit Playground Express, the other in CircuitPython for the CLUE. Use a loud sound, such as a clap or snap to activate the CPX timer, or a wave of a hand to start the CLUE.

This is a great beginner project to show how to make a real product with a few 'blocks' of code!

Parts

Your browser does not support the video tag.

Circuit Playground Express

$24.95
IN STOCK
Add To Cart

Your browser does not support the video tag.

Adafruit Circuit Playground Express or Bluefruit Enclosure

$4.95
IN STOCK
Add To Cart

Your browser does not support the video tag.

Adafruit CLUE - nRF52840 Express with Bluetooth LE

OUT OF STOCK

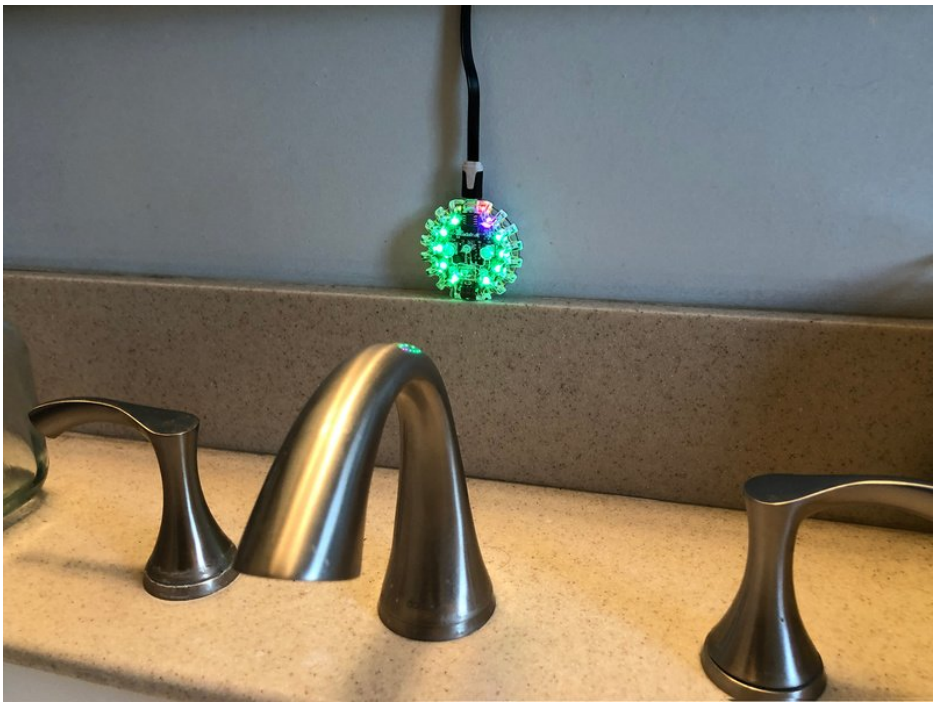
Out Of Stock

USB cable - USB A to Micro-B

$2.95
IN STOCK

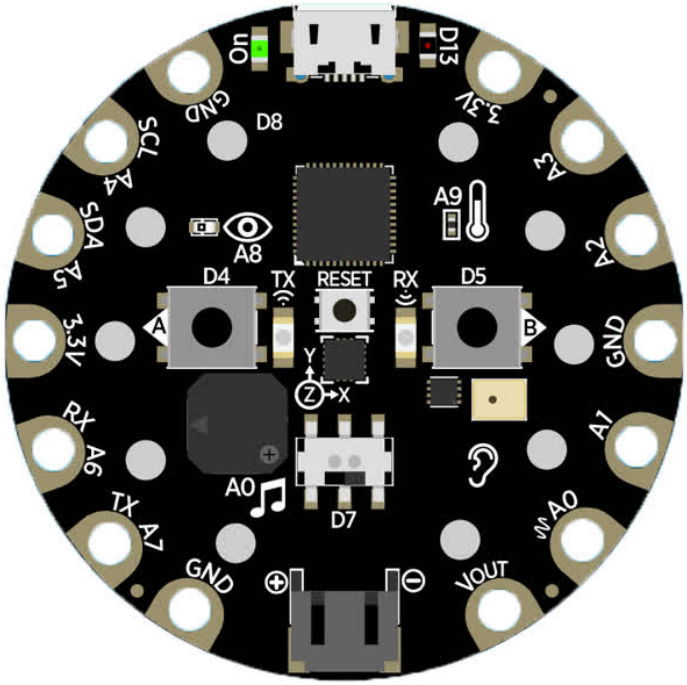
Add To Cart

Circuit Playground Express Timer



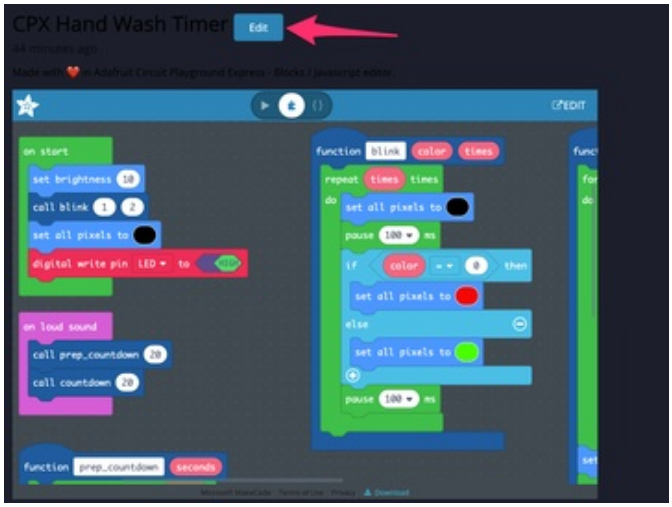
The Circuit Playground Express (CPX) hand wash timer will be created with MakeCode. If you're new to MakeCode, check out this guide (https://adafru.it/wWd) on getting started.

We'll use MakeCode's graphical, block-based interface to create the code that runs on the CPX.



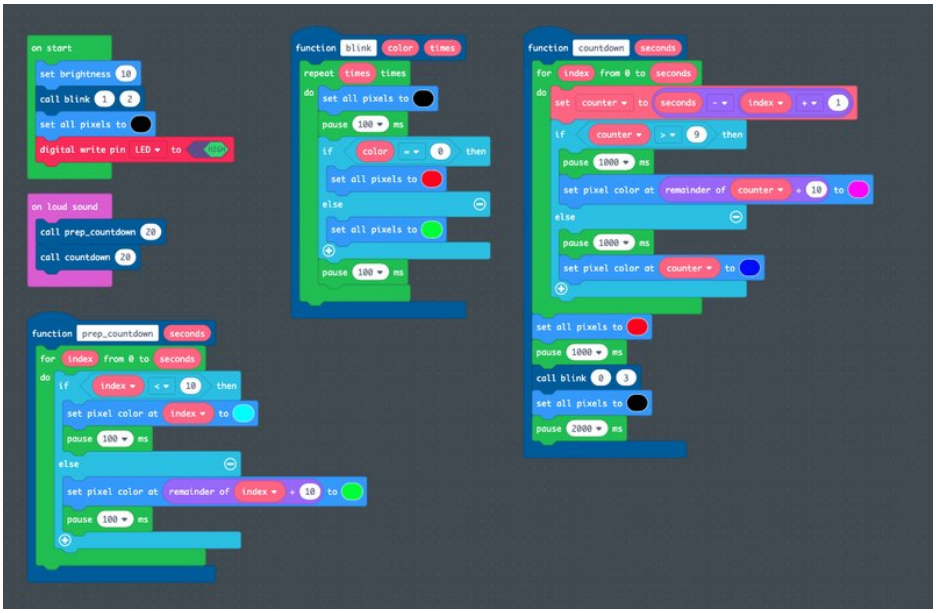
https://adafru.it/JRC

https://adafru.it/JRC



Click the link above to open the MakeCode file -- be sure to use an approved browser, such as Chrome or Edge.

Once you have the code open, click on the Edit button.



The key features of the code are the **on start** block, the **on loud sound** block, and the three functions:

- **blink**
- **prep_countdown**
- **countdown**

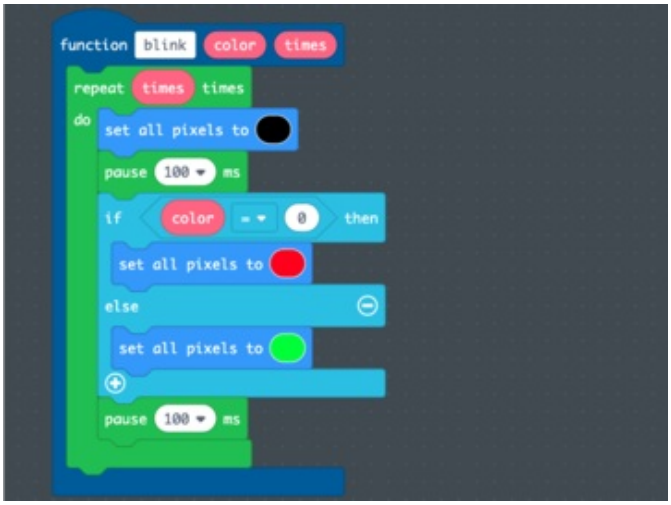
On Start

When the CPX is first powered on or reset, the **on start** block will run. In it we set the NeoPixel **brightness** to **10**, call the **blink** function with arguments of **1** for **color** and **2** for **times** (more on **blink** below).

Then the pixels are turned of and the on-board red LED is turned on by setting the pin **high.**

On Loud Sound

When a lound sound is detected, the code in the **on loud sound** block runs. This calls the **prep_countdown** function and the **countdown** function. More on those below.



Blink

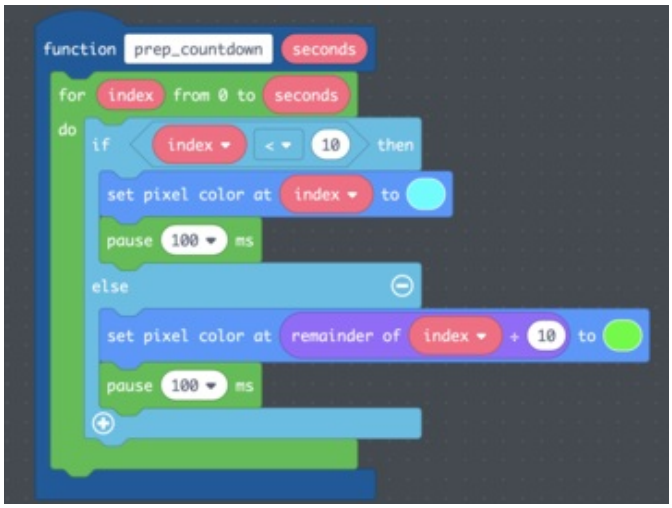
The **blink** function is called, along with two arguments, **color** and **time.**

The **repeat** loop will iterate however many times it is called, for example the startup loop blink calls it twice.

Inside the repeat, the pixels are first turned off, then after a short pause the color argument is checked -- if it is a **0** then the pixels will be set to **red**, if not, **green**.

This could be expanded to include more colors.

Then, there is a short **pause** and the process repeats, thus creating the blinking pattern.



Prep Countdown

The **prep_countdown** function is used to put a certain number of seconds "on the clock" by lighting up the NeoPixels one at a time in a counterclockwise direction. It is called along with an argument of a number of seconds.

The **for index** loop iterates whatever code is contained within it the number of times specified by the **seconds** value. Additionally, the **index** variable's value will start at **0** and increase by one for each iteration through the loop.

On the first ten iterations (0-9) the corresponding NeoPixel will be lit up cyan, thanks to the **if index < 10** check.

Once **index** is greater than 9, the pixels will loop around the circle a second time, this time being lit up green.

Since the value of **index** will be 10-20 and there are no NeoPixels on the CPX with those values, we need to do a bit of modulo math to loop back around to 0-9 again. That is how the **remainder of index / 10** block is used.

Countdown

The **countdown** function works similarly to the **prep_countdown** function, running back from **seconds** to **0,** and with a one second pause between each NeoPixel call.

To run backwards, a variable called **counter** is created, and is set to equal the **seconds** variable minus the **index + 1.** So, for example, **seconds is 20, index is 0** on the first loop through, so **counter = 20 - (0 + 1)** or 19.

For the first ten seconds, **counter > 10,** so after a one second pause the appropriate pixel is set to magenta for each iteration. Again, a modulo function is used to light the proper NeoPixel.

On the second ten seconds through, the **counter** is less than or equal to ten, so the pixels are set to blue when its their turn.

After the **20** seconds elapse, the whole ring is set to red and blinks red three times to tell you its OK to stop scrubbing!

CLUE Hand Wash Timer

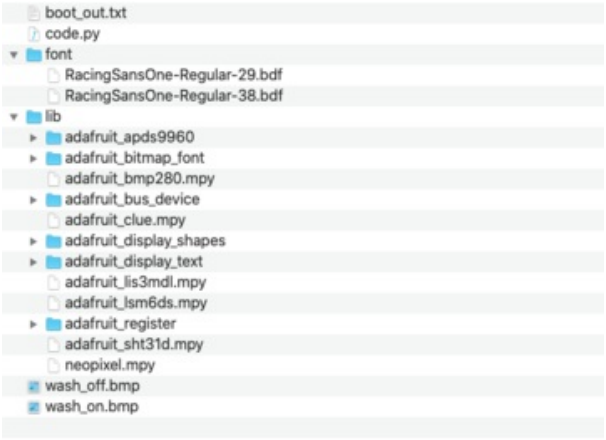
The CLUE version of this project is a bit more sophisticated than the CPX one, since we have a TFT display to work with and a proximity sensor.

We'll set up the board with CircuitPython and the necessary libraries first, and then code it.

Setup

First, follow this guide (https://adafru.it/Jab) on getting CircuitPython and the on your CLUE board. Then, use these instructions (https://adafru.it/Jb9) to add the libraries you'll need.

Once you've gotten the board set up, click Download: Project Zip below in the code guide. Expand the .zip file and then drag the two **.bmp** images and the **font** folder to your CLUE's **CIRCUITPY** drive via USB.



Your CLUE's **CIRCUITPY** drive should look like this.

Code

Copy the code from the code-block below and paste it into the Mu editor and save it to your CLUE as **code.py** (or copy **code.py** from the zip file and place on the **CIRCUITPY** drive).

```
"""
Start a 20 second hand washing timer via proximity sensor.
Countdown the seconds with text and beeps.
Display a bitmaps for waiting and washing modes.
"""

import time
import board
from adafruit_clue import clue
from adafruit_display_text import label
from adafruit_bitmap_font import bitmap_font
import displayio
import pulseio

clue.display.brightness = 0.8
clue_display = displayio.Group(max_size=4)

# draw the background image
wash_on_file = open("wash_on.bmp", "rb")
```

```
wash_on_bmp = displayio.OnDiskBitmap(wash_on_file)
wash_on_sprite = displayio.TileGrid(wash_on_bmp, pixel_shader=displayio.ColorConverter())
clue_display.append(wash_on_sprite)

# draw the foreground image
wash_off_file = open("wash_off.bmp", "rb")
wash_off_bmp = displayio.OnDiskBitmap(wash_off_file)
wash_off_sprite = displayio.TileGrid(wash_off_bmp, pixel_shader=displayio.ColorConverter())
clue_display.append(wash_off_sprite)


# Create text
# first create the group
text_group = displayio.Group(max_size=5, scale=1)
# Make a label
title_font = bitmap_font.load_font("/font/RacingSansOne-Regular-38.bdf")
title_font.load_glyphs("HandWashTimer".encode('utf-8'))
title_label = label.Label(title_font, text="Hand Wash", color=0x001122)
# Position the label
title_label.x = 10
title_label.y = 16
# Append label to group
text_group.append(title_label)

title2_label = label.Label(title_font, text="Timer", color=0x001122)
# Position the label
title2_label.x = 6
title2_label.y = 52
# Append label to group
text_group.append(title2_label)

timer_font = bitmap_font.load_font("/font/RacingSansOne-Regular-29.bdf")
timer_font.load_glyphs("0123456789ADSWabcdefghijklmnopqrstuvwxyz:!".encode('utf-8'))
timer_label = label.Label(timer_font, text="Wave to start", color=0x4f3ab1, max_glyphs=15)
timer_label.x = 24
timer_label.y = 100
text_group.append(timer_label)

clue_display.append(text_group)
clue.display.show(clue_display)

def countdown(seconds):
    for i in range(seconds):
        buzzer.duty_cycle = 2**15
        timer_label.text = ("Scrub time:  {}".format(seconds-i))
        buzzer.duty_cycle = 0
        time.sleep(1)
    timer_label.text = ("Done!")
    wash_off_sprite.x = 0
    buzzer.duty_cycle = 2**15
    time.sleep(0.3)
    buzzer.duty_cycle = 0
    timer_label.x = 24
    timer_label.y = 100
    timer_label.text = ("Wave to start")

# setup buzzer
buzzer = pulseio.PWMOut(board.SPEAKER, variable_frequency=True)
buzzer.frequency = 1000
```

```
while True:
    # print("Distance: {}".format(clue.proximity))  # use to test the sensor
    if clue.proximity > 1:
        timer_label.x = 12
        timer_label.y = 226
        timer_label.text = "Scrub Away!"
        wash_off_sprite.x = 300
        time.sleep(2)
        countdown(20)
```

What the code is doing

The code does a few fairly simple things. First, we import the necessary libraries.

```
import time
import board
from adafruit_clue import clue
from adafruit_display_text import label
from adafruit_bitmap_font import bitmap_font
import displayio
import pulseio
```

Display and Images

Next, we set the display brightness and then use the **clue_display** commands to set up the **displayio** group that will hold the bitmap images and text.

We then draw both images onto the screen, one on top of the other. (The reason we do this is for speed. By loading both images into memory at startup, it will be fast to later move the top image out of the way to reveal the bottom image.)

```
# draw the background image
wash_on_file = open("wash_on.bmp", "rb")
wash_on_bmp = displayio.OnDiskBitmap(wash_on_file)
wash_on_sprite = displayio.TileGrid(wash_on_bmp, pixel_shader=displayio.ColorConverter())
clue_display.append(wash_on_sprite)

# draw the foreground image
wash_off_file = open("wash_off.bmp", "rb")
wash_off_bmp = displayio.OnDiskBitmap(wash_off_file)
wash_off_sprite = displayio.TileGrid(wash_off_bmp, pixel_shader=displayio.ColorConverter())
clue_display.append(wash_off_sprite)
```

Text Labels

Then we add groups for each line of text we'll be displaying. This involves specifying bitmap fonts (learn how to make your own in this guide (https://adafru.it/E7E)!)

Again, to speed things up later, we'll preload the glyphs (a.k.a., characters) now. Notice that we're loading only the glyphs we intend to use to save time.

We also specify the text to be displayed, the colors, and positions in x/y screen space, and then append each text label

to the **text_group**.

The `clue.display.show(clue_display)` command finally tells the screen to display the bitmaps and text lines.

```
# Create text
# first create the group
text_group = displayio.Group(max_size=5, scale=1)
# Make a label
title_font = bitmap_font.load_font("/font/RacingSansOne-Regular-38.bdf")
title_font.load_glyphs("HandWashTimer".encode('utf-8'))
title_label = label.Label(title_font, text="Hand Wash", color=0x001122)
# Position the label
title_label.x = 10
title_label.y = 16
# Append label to group
text_group.append(title_label)

title2_label = label.Label(title_font, text="Timer", color=0x001122)
# Position the label
title2_label.x = 6
title2_label.y = 52
# Append label to group
text_group.append(title2_label)

timer_font = bitmap_font.load_font("/font/RacingSansOne-Regular-29.bdf")
timer_font.load_glyphs("0123456789ADSWabcdefghijklmnopqrstuvwxyz:!".encode('utf-8'))
timer_label = label.Label(timer_font, text="Wave to start", color=0x4f3ab1, max_glyphs=15)
timer_label.x = 24
timer_label.y = 100
text_group.append(timer_label)

clue_display.append(text_group)
clue.display.show(clue_display)
```

Countdown Function

We'll create a separate function called `countdown()` to handle the hand washing countdown process after the CLUE has been triggered.

It will accept an argument for the number of `seconds` you intend, which makes it easier to adjust if you like versus a hard coded value.

The function iterates through a loop for the number of `seconds`, incrementing the value of `i` each time. During each run through this loop it will buzz the buzzer, displaying the scrub time remaining in seconds, pause a second, and repeat.

When the loop finishes the last iteration (20 seconds, say), the **wash_off.bmp** sprite is moved into place, and the timer label resets to say "**Wave to start**".

```
def countdown(seconds):
    for i in range(seconds):
        buzzer.duty_cycle = 2**15
        timer_label.text = ("Scrub time:  {}".format(seconds-i))
        buzzer.duty_cycle = 0
        time.sleep(1)
    timer_label.text = ("Done!")
    wash_off_sprite.x = 0
    buzzer.duty_cycle = 2**15
    time.sleep(0.3)
    buzzer.duty_cycle = 0
    timer_label.x = 24
    timer_label.y = 100
    timer_label.text = ("Wave to start")
```

Buzzer Setup

The buzzer is setup to use the `pulsio()` command and buzz the onboard speaker using a pulse wave modulation signal (PWM) set to a frequency of 1000.

When the `buzzer.duty_cycle = 2**15` we'll hear the buzzing, and when the `duty_cycle = 0` it will stop. These commands are found inside the `countdown()` function.

```
buzzer = pulseio.PWMOut(board.SPEAKER, variable_frequency=True)
buzzer.frequency = 1000
```

Main Loop

Finally, we have finished setting things up and can run the main loop.

Here, we'll use `clue.proximity` to check the value of the CLUE's on-board proximity sensor. The value will read 0 when nothing is in front of the sensor. When an object -- such as your hand -- is about 8cm or closer, the sensor will report a value greater than 1 and trip the rest of the conditional statement to run.

In this case, we'll move the timer label to the bottom, replacing the text with "**Scrub Away!**", move the "off" bitmap out of the way to reveal the hand washing "on" bitmap, wait two seconds and then call the `countdown` function for 20 seconds.

```
while True:
    # print("Distance: {}".format(clue.proximity))  # use to test the sensor
    if clue.proximity > 1:
        timer_label.x = 12
        timer_label.y = 226
        timer_label.text = "Scrub Away!"
        wash_off_sprite.x = 300
        time.sleep(2)
        countdown(20)
```

Now, you can mount the CLUE over your sink, run a USB power cable to it from an outlet, and wash your hands to a stylish timer!