# Table of Contents

## Overview
- Prerequisite guides
- Parts
- Other parts
- Other materials

## Connect Electronics
- Connect without soldering
- Connect by soldering:

## Install CircuitPython
- Set up CircuitPython Quick Start!
- PyPortal Default Files

## PyPortal CircuitPython Setup
- Adafruit CircuitPython Bundle

## Internet Connect!
- What's a secrets file?
- Connect to WiFi
- Requests
- HTTP GET with Requests
- HTTP POST with Requests
- Advanced Requests Usage
- WiFi Manager

## Code PyPortal With CircuitPython
- CircuitPython Code
- Downloading the libraries

## Using the Wake-Up Light
- Here's an example of the wake-up light in use (for a late riser...)
- Before wake-up time
- During wake-up time
- After wake-up time
- How it works
- Displaying text
- Text position
- Font
- Text color
- Initializing wake-up times
- Main loop

## Affixing the Strip
- Attaching strip to bed frame or other furniture
- Adding features
Overview

Do you suffer from a lack of natural sunlight in your room? Does the lack of sunlight make it almost impossible to get out of bed when you hear a dreadful alarm?

Yes, windows would be nice, but we live in New York City and who needs those when you got a PyPortal and a NeoPixel strip?

In this project, you can wake up to a soft white light fading into your consciousness! You could get a consumer grade sunrise alarm clock that may solve your problem but you can also do it yourself!

Using a PyPortal, and a NeoPixel strip, make a sunrise alarm clock that wakes you up with light! Program the PyPortal with CircuitPython to set your wake-up times and the strip will begin to brighten 30 minutes before your wake-up time.

Prerequisite guides

This project may require some soldering depending on the NeoPixel strip you get. Here's a great guide on soldering if it's your first time or you need some refreshing!

- Adafruit Guide To Excellent Soldering ()
Parts

You can pick up an Adafruit PyPortal and a USB cable (if needed). If you like, you can mount the PyPortal in the Adafruit laser-cut acrylic stand. All these parts are bundled in AdaBox 011 if you’d like to buy them together.

AdaBox011 - PyPortal
Reach out beyond your desk - to the stars and beyond - with PyPortal! This ADABOX features a new, easy-to-use IoT device that allows you to customize and create your...
https://www.adafruit.com/product/4061

Instead of AdaBox 011, you can buy parts separately:

Adafruit PyPortal - CircuitPython Powered Internet Display
PyPortal, our easy-to-use IoT device that allows you to create all the things for the “Internet of Things” in minutes. Make custom touch screen interface...
https://www.adafruit.com/product/4116

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592
Adafruit PyPortal Desktop Stand Enclosure Kit
PyPortal is our easy-to-use IoT device that allows you to create all the things for the “Internet of Things” in minutes. Create little pocket...
https://www.adafruit.com/product/4146

Other parts
You only need 1m of neopixel strip ():

Adafruit NeoPixel Digital RGBW LED Strip - White PCB 30 LED/m
What is better than smart RGB LEDs? Smart RGB+White LEDs! These NeoPixels now have 4 LEDs in them (red, green, blue and white) for excellent lighting effects. These LED...
https://www.adafruit.com/product/2832

STEMMA JST PH 2mm 3-Pin to Male Header Cable - 200mm
This cable will let you turn a JST PH 3-pin cable port into 3 individual wires with high-quality 0.1" male header plugs on the end. We're carrying these to match up with our...
https://www.adafruit.com/product/3893
Breadboarding wire bundle
75 flexible stranded core wires with stiff ends molded on in red, orange, yellow, green, blue, brown, black and white. These are a major improvement over the "box of bent...
https://www.adafruit.com/product/153

Small Alligator Clip Test Lead (set of 12)
Connect this to that without soldering using these handy mini alligator clip test leads. 15" cables with alligator clip on each end, color coded. You get 12 pieces in 6 colors....
https://www.adafruit.com/product/1008

Hook-up Wire Spool Set - 22AWG Stranded-Core - 6 x 25ft
This is a box of six 25ft spools of stranded-core wire. Stranded-core wire is best used for wiring jigs where there's...
https://www.adafruit.com/product/3111

Premium Female/Female Jumper Wires - 40 x 6"
Handy for making wire harnesses or jumpering between headers on PCB's. These premium jumper wires approximately 6" (150mm) long and come in a 'strip' of 40 (4 pieces...
https://www.adafruit.com/product/266
5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable

Our all-in-one 5V 2.5 Amp + MicroUSB cable power adapter is the perfect choice for powering single-board computers like Raspberry Pi, BeagleBone, or anything else that's...


Other materials

- single-sided tape
- foam tape
- hot glue gun and sticks
- a bed frame or dresser to attach NeoPixel strip to

Connect Electronics

First we'll connect the electronics before we add the code and attach the strip to a bed frame (or another piece of furniture).

Because these specific NeoPixel strips come in 5 meter reels, if you order 1m of reel you may get a piece of reel in the middle and thus may have to solder to the pads of
the strip. Here are the two possibilities of strip end you may get and how to connect to the PyPortal with each.

Connect without soldering

If the data in end of your neopixel strip looks like this, connect wires the following way.

The data in end of a NeoPixel strip is the end where the arrows on the strip are pointing away from.
Attach the JST connector to D3 on the back of the PyPortal.
Plug the white and black cables into the corresponding inputs on the NeoPixel strip.
Connect the alligator clips from the red JST cable to the red NeoPixel cable.
Connect by soldering:

Here's a great guide to soldering () for your reference.

If the data in end of your NeoPixel strip looks like this, connect wires the following way.
Prep 3 wires, preferable a red, black and white. Solder red to 5V, white to Din and black to GND on the neopixel strip. Attach the JST connector to D3 on the back of the PyPortal. Connect the female breadboard cables from the PyPortal to the NeoPixel strip.
Install CircuitPython

**CircuitPython** is a derivative of **MicroPython** 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 the PyPortal via CircuitPython.org

Download the latest version of CircuitPython for the PyPortal Pynt via CircuitPython.org

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

Download and save it to your desktop (or wherever is handy).
Plug your PyPortal 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 in the middle (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 PORTALBOOT.

Drag the adafruit-circuitpython-pyportal-<whatever>.uf2 file to PORTALBOOT.
The LED will flash. Then, the PORTALBOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

If you haven't added any code to your board, the only file that will be present is boot_out.txt. This is absolutely normal! It's time for you to add your code.py and get started!

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

**PyPortal Default Files**

Click below to download a zip of the files that shipped on the PyPortal or PyPortal Pynt.

- [PyPortal Default Files](#)
- [PyPortal Pynt Default Files](#)

**PyPortal CircuitPython Setup**

To use all the amazing features of your PyPortal with CircuitPython, you must first install a number of libraries. This page covers that process.

**Adafruit CircuitPython Bundle**

Download the Adafruit CircuitPython Library Bundle. You can find the latest release here:

- [Latest Adafruit CircuitPython Library Bundle](#)
Download the adafruit-circuitpython-bundle-*.x-mpy-*.zip bundle zip file where *.x MATCHES THE VERSION OF CIRCUITPYTHON YOU INSTALLED, and unzip a folder of the same name. Inside you'll find a lib folder. You have two options:

- You can add the lib folder to your CIRCUITPY drive. This will ensure you have all the drivers. But it will take a bunch of space on the 8 MB disk
- Add each library as you need it, this will reduce the space usage but you'll need to put in a little more effort.

At a minimum we recommend the following libraries, in fact we more than recommend. They're basically required. So grab them and install them into CIRCUITPY/lib now!

- adafruit_esp32spi - This is the library that gives you internet access via the ESP32 using (you guessed it!) SPI transport. You need this for anything Internet
- adafruit_requests - This library allows us to perform HTTP requests and get responses back from servers. GET/POST/PUT/PATCH - they're all in here!
- adafruit_pyportal - This is our friendly wrapper library that does a lot of our projects, displays graphics and text, fetches data from the internet. Nearly all of our projects depend on it!
- adafruit_portalbase - This library is the base library that adafruit_pyportal library is built on top of.
- adafruit_touchscreen - a library for reading touches from the resistive touchscreen. Handles all the analog noodling, rotation and calibration for you.
- adafruit_io - this library helps connect the PyPortal to our free datalogging and viewing service
- adafruit_imageload - an image display helper, required for any graphics!
- adafruit_display_text - not surprisingly, it displays text on the screen
- adafruit_bitmap_font - we have fancy font support, and its easy to make new fonts. This library reads and parses font files.
- adafruit_slideshow - for making image slideshows - handy for quick display of graphics and sound
- neopixel - for controlling the onboard neopixel
- adafruit_adt7410 - library to read the temperature from the on-board Analog Devices ADT7410 precision temperature sensor (not necessary for Titano or Pynt)
- adafruit_bus_device - low level support for I2C/SPI
- adafruit_fakerequests - This library allows you to create fake HTTP requests by using local files.
Internet Connect!

Once you have CircuitPython setup and libraries installed we can get your board connected to the Internet. Note that access to enterprise level secured WiFi networks is not currently supported, only WiFi networks that require SSID and password.

To get connected, you will need to start by creating a secrets file.

What's a secrets file?

We expect people to share tons of projects as they build CircuitPython WiFi widgets. What we want to avoid is people accidentally sharing their passwords or secret tokens and API keys. So, we designed all our examples to use a secrets.py file, that is in your CIRCUITPY drive, to hold secret/private/custom data. That way you can share your main project without worrying about accidentally sharing private stuff.

Your secrets.py file should look like this:

```python
# This file is where you keep secret settings, passwords, and tokens!
# If you put them in the code you risk committing that info or sharing it

secrets = {
    'ssid' : 'home ssid',
    'password' : 'my password',
    'timezone' : "America/New_York", # http://worldtimeapi.org/timezones
    'github_token' : 'fawfj23rakjnfawiefa',
    'hackaday_token' : 'h4xx0rs3kret',
}
```

Inside is a python dictionary named secrets with a line for each entry. Each entry has an entry name (say 'ssid') and then a colon to separate it from the entry key and finally a comma.

At a minimum you'll need the `ssid` and `password` for your local WiFi setup. As you make projects you may need more tokens and keys, just add them one line at a time. See for example other tokens such as one for accessing github or the hackaday API. Other non-secret data like your timezone can also go here, just cause it's called secrets doesn't mean you can't have general customization data in there!

For the correct time zone string, look at [http://worldtimeapi.org/timezones()](http://worldtimeapi.org/timezones) and remember that if your city is not listed, look for a city in the same time zone, for example Boston, New York, Philadelphia, Washington DC, and Miami are all on the same time as New York.
Of course, don’t share your secrets.py - keep that out of GitHub, Discord or other project-sharing sites.

Connect to WiFi

OK now you have your secrets setup - you can connect to the Internet.

To do this, 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 examples/ 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:

```python
import board
import busio
from digitalio import DigitalInOut
import adafruit_requests as requests
import adafruit_esp32spi.adafruit_esp32spi_socket as socket
from adafruit_esp32spi import adafruit_esp32spi
	ry:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise
```

©Adafruit Industries
print("ESP32 SPI webclient test")

TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
JSON_URL = "http://api.coindesk.com/v1/bpi/currentprice/USD.json"

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an AirLift Shield:
# esp32_cs = DigitalInOut(board.D10)
# esp32_ready = DigitalInOut(board.D7)
# esp32_reset = DigitalInOut(board.D5)

# If you have an AirLift Featherwing or ItsyBitsy Airlift:
# esp32_cs = DigitalInOut(board.D13)
# esp32_ready = DigitalInOut(board.D11)
# esp32_reset = DigitalInOut(board.D12)

# If you have an externally connected ESP32:
# NOTE: You may need to change the pins to reflect your wiring
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)
requests.set_socket(socket, esp)

if esp.status == adafruit_esp32spi.WL_IDLE_STATUS:
    print("ESP32 found and in idle mode")
    print("Firmware vers.", esp.firmware_version)
    print("MAC addr:", [hex(i) for i in esp.MAC_address])

for ap in esp.scan_networks():
    print("%s		RSSI: %d" % (str(ap["ssid"], "utf-8"), ap["rssi"]))

print("Connecting to AP...")
while not esp.is_connected:
    try:
        esp.connect_AP(secrets["ssid"], secrets["password"])
    except OSError as e:
        print("could not connect to AP, retrying: ", e)
        continue
    print("Connected to", str(esp.ssid, "utf-8"), "\tRSSI:", esp.rssi)
    print("My IP address is", esp.pretty_ip(esp.ip_address))
    print("IP lookup adafruit.com: %s" %
        esp.pretty_ip(esp.get_host_by_name(" adafruit.com")))
    print("Ping google.com: %d ms" % esp.ping(" google.com"))

# esp._debug = True
print("Fetching text from", TEXT_URL)
r = requests.get(TEXT_URL)
print("-" * 40)
print(r.text)
print("-" * 40)
r.close()

print()
print("Fetching json from", JSON_URL)
r = requests.get(JSON_URL)
print("-" * 40)
print(r.json())
print("-" * 40)
r.close()
print("Done!")

And save it to your board, with the name code.py.

Don't forget you'll also need to create the secrets.py file as seen above, with your WiFi ssid and password.

In a serial console, you should see something like the following. For more information about connecting with a serial console, view the guide Connecting to the Serial Console.

In order, the example code...

Initializes the ESP32 over SPI using the SPI port and 3 control pins:

```python
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)
```

Tells our requests library the type of socket we're using (socket type varies by connectivity type - we'll be using the adafruit_esp32spi_socket for this example).
We'll also set the interface to an \texttt{esp} object. This is a little bit of a hack, but it lets us use \texttt{requests} like CPython does.

\begin{verbatim}
requests.set_socket(socket, esp)
\end{verbatim}

Verifies an ESP32 is found, checks the firmware and MAC address

\begin{verbatim}
if esp.status == adafruit_esp32spi.WL_IDLE_STATUS:
    print("ESP32 found and in idle mode")
    print("Firmware vers.", esp.firmware_version)
    print("MAC addr:", [hex(i) for i in esp.MAC_address])
\end{verbatim}

Performs a scan of all access points it can see and prints out the name and signal strength:

\begin{verbatim}
for ap in esp.scan_networks():
    print("\t%s\t	RSSI: %d" % (str(ap['ssid'], 'utf-8'), ap['rssi']))
\end{verbatim}

Connects to the AP we've defined here, then prints out the local IP address, attempts to do a domain name lookup and ping google.com to check network connectivity (note sometimes the ping fails or takes a while, this isn't a big deal)

\begin{verbatim}
print("Connecting to AP...")
while not esp.is_connected:
    try:
        esp.connect_AP(secrets["ssid"], secrets["password"])
    except RuntimeError as e:
        print("could not connect to AP, retrying: ", e)
        continue
print("My IP address is", esp.pretty_ip(esp.ip_address))
print("IP lookup adafruit.com: %s " % esp.pretty_ip(esp.get_host_by_name("adafruit.com")))
\end{verbatim}

OK now we're getting to the really interesting part. With a SAMD51 or other large-RAM (well, over 32 KB) device, we can do a lot of neat tricks. Like for example we can implement an interface a lot like \texttt{requests} - which makes getting data really really easy

To read in all the text from a web URL call \texttt{requests.get} - you can pass in \texttt{https} URLs for SSL connectivity

\begin{verbatim}
TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
print("Fetching text from", TEXT_URL)
r = requests.get(TEXT_URL)
print("---")
print(r.text)
print("---")
r.close()
\end{verbatim}
Or, if the data is in structured JSON, you can get the json pre-parsed into a Python dictionary that can be easily queried or traversed. (Again, only for nRF52840, M4 and other high-RAM boards)

```python
JSON_URL = "http://api.coindesk.com/v1/bpi/currentprice/USD.json"
print("Fetching json from", JSON_URL)
r = requests.get(JSON_URL)
print('.'*40)
print(r.json())
print('.'*40)
r.close()
```

Requests

We've written a requests-like () library for web interfacing named Adafruit_CircuitPython_on_Requests (). This library allows you to send HTTP/1.1 requests without "crafting" them and provides helpful methods for parsing the response from the server.

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 examples/ 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: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

# adafruit_requests usage with an esp32spi_socket
import board
import busio
from digitalio import DigitalInOut
import adafruit_esp32spi.adafruit_esp32spi_socket as socket
import adafruit_requests
```
from adafruit_esp32spi import adafruit_esp32spi
import adafruit_requests as requests

# Add a secrets.py to your filesystem that has a dictionary called secrets with
# "ssid" and
# "password" keys with your WiFi credentials. DO NOT share that file or commit it
# into Git or other
# source control.
# pylint: disable=no-name-in-module,wrong-import-order
try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

# If you have an AirLift Featherwing or ItsyBitsy Airlift:
# esp32_cs = DigitalInOut(board.D13)
# esp32_ready = DigitalInOut(board.D11)
# esp32_reset = DigitalInOut(board.D12)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

print("Connecting to AP...")
while not esp.is_connected:
    try:
        esp.connect_AP(secrets["ssid"], secrets["password"])  
        except RuntimeError as e:
            print("could not connect to AP, retrying: ", e)
            continue
        print("Connected to", str(esp.ssid, "utf-8"), ",\nRSSI:", esp.rssi)

    # Initialize a requests object with a socket and esp32spi interface
    socket.set_interface(esp)
    requests.set_socket(socket, esp)

    TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
    JSON_GET_URL = "https://httpbin.org/get"
    JSON_POST_URL = "https://httpbin.org/post"

    print("Fetching text from %s" % TEXT_URL)
    response = requests.get(TEXT_URL)
    print("\n" * 40)
    print("Text Response: ", response.text)
    print("\n" * 40)
    response.close()

    print("Fetching JSON data from %s" % JSON_GET_URL)
    response = requests.get(JSON_GET_URL)
    print("\n" * 40)
    print("JSON Response: ", response.json())
    print("\n" * 40)
    response.close()

data = "31F"
print("POSTing data to \{0\}: \{1\}.format(JSON_POST_URL, data))
response = requests.post(JSON_POST_URL, data=data)
import board
import busio
from digitalio import DigitalInOut
import adafruit_esp32spi.adafruit_esp32spi_socket as socket
from adafruit_esp32spi import adafruit_esp32spi
import adafruit_requests as requests

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

print("Connecting to AP...")
while not esp.is_connected:
    try:
        esp.connect_AP(b'MY_SSID_NAME', b'MY_SSID_PASSWORD')
    except RuntimeError as e:
        print("could not connect to AP, retrying: ",e)
        continue
    print("Connected to", str(esp.ssid, 'utf-8'), "\tRSSI:" , esp.rssi)

# Initialize a requests object with a socket and esp32spi interface
requests.set_socket(socket, esp)

**HTTP GET with Requests**

The code makes a HTTP GET request to Adafruit's WiFi testing website - [http://wifitest.adafruit.com/testwifi/index.html](http://wifitest.adafruit.com/testwifi/index.html).
To do this, we'll pass the URL into `requests.get()`. We're also going to save the response from the server into a variable named `response`.

Having requested data from the server, we'd now like to see what the server responded with. Since we already saved the server's `response`, we can read it back. Luckily for us, requests automatically decodes the server's response into human-readable text, you can read it back by calling `response.text`.

Lastly, we'll perform a bit of cleanup by calling `response.close()`. This closes, deletes, and collects the response's data.

```
print("Fetching text from %s"%TEXT_URL)
response = requests.get(TEXT_URL)
print('.'*40)

print("Text Response: ", response.text)
print('.'*40)
response.close()
```

While some servers respond with text, some respond with json-formatted data consisting of attribute–value pairs.

CircuitPython_Requests can convert a JSON-formatted response from a server into a CPython `dict` object.

We can also fetch and parse json data. We'll send a HTTP get to a url we know returns a json-formatted response (instead of text data).

Then, the code calls `response.json()` to convert the response to a CPython `dict`.

```
print("Fetching JSON data from %s"%JSON_GET_URL)
response = requests.get(JSON_GET_URL)
print('.'*40)

print("JSON Response: ", response.json())
print('.'*40)
response.close()
```

**HTTP POST with Requests**

Requests can also POST data to a server by calling the `requests.post` method, passing it a `data` value.

```
data = '31F'
print("POSTing data to {0}: {1}".format(JSON_POST_URL, data))
response = requests.post(JSON_POST_URL, data=data)
```
```python
print('-'*40)
json_resp = response.json()  # Parse out the 'data' key from json_resp dict.
print("Data received from server:", json_resp['data'])
print('-'*40)
response.close()

You can also post json-formatted data to a server by passing json_data into the requests.post method.

```
```python

json_data = {"Date" : "July 25, 2019"}
print("POSTing data to {0}: {1}".format(JSON_POST_URL, json_data))
response = requests.post(JSON_POST_URL, json=json_data)
print('-'*40)

json_resp = response.json()  # Parse out the 'json' key from json_resp dict.
print("JSON Data received from server:", json_resp['json'])
print('-'*40)
response.close()
```

### Advanced Requests Usage

Want to send custom HTTP headers, parse the response as raw bytes, or handle a response's http status code in your CircuitPython code?

We've written an example to show advanced usage of the requests module below.

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 examples/ 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:
import board
import busio
from digitalio import DigitalInOut
import adafruit_esp32spi.adafruit_esp32spi_socket as socket
from adafruit_esp32spi import adafruit_esp32spi
import adafruit_requests as requests

# Add a secrets.py to your filesystem that has a dictionary called secrets with
# "ssid" and
# "password" keys with your WiFi credentials. DO NOT share that file or commit it
# into Git or other
# source control.
# pylint: disable=no-name-in-module,wrong-import-order
try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

print("Connecting to AP...")
while not esp.is_connected:
    try:
        esp.connect_AP(secrets["ssid"], secrets["password"])
    except RuntimeError as e:
        print("could not connect to AP, retrying: ", e)
        continue
print("Connected to", str(esp.ssid, "utf-8"), "\tRSSI: ", esp.rssi)

# Initialize a requests object with a socket and esp32spi interface
socket.set_interface(esp)
requests.set_socket(socket, esp)

JSON_GET_URL = "http://httpbin.org/get"

# Define a custom header as a dict.
headers = {"user-agent": "blinky/1.0.0"}
WiFi Manager

That simplest example works but it's a little finicky - you need to constantly check WiFi status and have many loops to manage connections and disconnections. For more advanced uses, we recommend using the WiFiManager object. It will wrap the connection/status/requests loop for you - reconnecting if WiFi drops, resetting the ESP32 if it gets into a bad state, etc.

Here's a more advanced example that shows the WiFi manager and also how to POST data with some extra headers:

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 examples/ 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:
import time
import board
import busio
from digitalio import DigitalInOut
import neopixel
from adafruit_esp32spi import adafruit_esp32spi
from adafruit_esp32spi import adafruit_esp32spi_wifimanager

print("ESP32 SPI webclient test")

# Get wifi details and more from a secrets.py file
try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

status_light = neopixel.NeoPixel(
    board.NEOPIXEL, 1, brightness=0.2
)  # Uncomment for Most Boards

# Uncomment below for ItsyBitsy M4
# status_light = dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1,
#                                brightness=0.2)

# Uncomment below for an externally defined RGB LED
# import adafruit_rgbled
# from adafruit_esp32spi import PWMOut
# RED_LED = PWMOut.PWMOut(esp, 26)
# GREEN_LED = PWMOut.PWMOut(esp, 27)
# BLUE_LED = PWMOut.PWMOut(esp, 25)
# status_light = adafruit_rgbled.RGBLED(RED_LED, BLUE_LED, GREEN_LED)

wifi = adafruit_esp32spi_wifimanager.ESPSPI_WiFiManager(esp, secrets, status_light)

counter = 0

while True:
    try:
        print("Posting data...", end="")
        data = counter
        feed = "test"
        payload = {"value": data}
        response = wifi.post("https://io.adafruit.com/api/v2/
                        + secrets["aio_username"]
                        + "feeds/
                        + feed
                        + "/data",
                        json=payload,
                        headers={"X-AIO-KEY": secrets["aio_key"]},
                        )
        print(response.json())
        counter = counter + 1
        print("OK")
    except OSError as e:
        print("Failed to get data, retrying\n", e)
        wifi.reset()
        continue
        response = None
        time.sleep(15)

You'll note here we use a secrets.py file to manage our SSID info. The wifimanager is given the ESP32 object, secrets and a neopixel for status indication.

Note, you'll need to add a some additional information to your secrets file so that the code can query the Adafruit IO API:

- **aio_username**
- **aio_key**

You can go to your adafruit.io View AIO Key link to get those two values and add them to the secrets file, which will now look something like this:

```python
# This file is where you keep secret settings, passwords, and tokens!
# If you put them in the code you risk committing that info or sharing it

secrets = {
    'ssid' : '_your_ssid_',
    'password' : '_your_wifi_password_',
    'timezone' : "America/Los_Angeles", # http://worldtimeapi.org/timezones
    'aio_username' : '_your_aio_username_',
    'aio_key' : '_your_aio_key_',
}
```
Next, set up an Adafruit IO feed named **test**

- If you do not know how to set up a feed, [follow this page and come back when you've set up a feed named test.](#)

We can then have a simple loop for posting data to Adafruit IO without having to deal with connecting or initializing the hardware!

Take a look at your test feed on Adafruit.io and you'll see the value increase each time the CircuitPython board posts data to it!

![Graph showing data increase](image)

## Code PyPortal With CircuitPython

### CircuitPython Code

In the embedded code element below, click on the Download: Project Zip link, and save the .zip archive file to your computer.

Then, uncompress the .zip file, it will unpack to a folder named PyPortal_Wakeup_Light.

Copy the contents of the PyPortal_Wakeup_Light directory to your PyPortal's CIRCUIT PY drive.

```python
# SPDX-FileCopyrightText: 2019 Isaac Wellish for Adafruit Industries
```
This example uses a PyPortal and rgbw leds for a simple "wake up" light. The strip starts to brighten 30 minutes before set wake up time. This program assumes a neopixel strip is attached to D3 on the Adafruit PyPortal.

```python
import time
import board
import neopixel
from adafruit_pyportal import PyPortal
from adafruit_bitmap_font import bitmap_font
from adafruit_display_text.Label import Label

# type in time to get up each day of the week
default_wake_up = "6:30A"
up_time_monday = default_wake_up
up_time_tuesday = default_wake_up
up_time_wednesday = default_wake_up
up_time_thursday = default_wake_up
up_time_friday = default_wake_up
up_time_saturday = "10:00A"
up_time_sunday = "10:00A"
wake_up_times = (up_time_monday,
                 up_time_tuesday,
                 up_time_wednesday,
                 up_time_thursday,
                 up_time_friday,
                 up_time_saturday,
                 up_time_sunday,
                 default_wake_up)

days_str = ("Mon.", "Tues.", "Wed.", "Thurs.", "Fri.", "Sat.", "Sun.")

# set neopixel min and max brightness
BRIGHTNESS = 0
MIN_BRIGHTNESS = 0
MAX_BRIGHTNESS = 0.85
# initialize neopixel strip
num_pixels = 30
ORDER = neopixel.RGBW
strip = neopixel.NeoPixel(board.D3, num_pixels, brightness=BRIGHTNESS,
                           pixel_order=ORDER)
strip.fill(0) # start it set to off
# color of strip
WHITE = (0, 0, 0, 255)
# number of minutes it takes for strip to fade from min to max
light_minutes = 30

determine the current working directory
# needed so we know where to find files
cwd = ("/"+_file_).rsplit('/', 1)[0]

# initialize the pyportal object and let us know what data to fetch and where
# to display it
pyportal = PyPortal(status_neopixel=board.NEOPIXEL,
                   default_bg=0x000000)

# set backlight default to off
backlight_off = 0
backlight_on = 0.8
pyportal.set_backlight(backlight_off)

# assign fonts
big_font = bitmap_font.load_font(cwd+'/fonts/Nunito-Light-75.bdf')
big_font.load_glyphs(b'0123456789:AP') # pre-load glyphs for fast printing
print('loading fonts...')
info_font = bitmap_font.load_font(cwd+'/fonts/Nunito-Black-17.bdf')
info_font.load_glyphs(b'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ-,.:/')
```

© Adafruit Industries
```python
time_color = 0xFFFFFF
time_position = (75,130)
time_textarea = Label(big_font, color=time_color,
x=time_position[0], y=time_position[1])

wakeup_time_color = 0xFFFFFF
wakeup_time_position = (15,200)
wakeup_time_textarea = Label(info_font, color=wakeup_time_color,
x=wakeup_time_position[0], y=wakeup_time_position[1])

light_on_time_color = 0xFFFFFF
light_on_time_position = (15,220)
light_on_time_textarea = Label(info_font, color=light_on_time_color,
x=light_on_time_position[0], y=light_on_time_position[1])

pyportal.splash.append(time_textarea)
pyportal.splash.append(wakeup_time_textarea)
pyportal.splash.append(light_on_time_textarea)

while True:
    try:
        print("Getting time from internet!")
        pyportal.get_local_time()
    except RuntimeError as e:
        print("Some error occurred, retrying! -", e)
        continue
    break

# parse given time string into hour minute and AM_PM elements

def parseTime(time_before):
    hours_before, minutes_before = time_before.split(":")
    minutes_before = int(minutes_before[:-1])
    AM_PM_str = minutes_before[-1:]
    if (hours_before != '12') and AM_PM_str == 'P':
        hours_before = int(hours_before) + 12
    elif ((hours_before == '12') and (AM_PM_str == 'P')):
        hours_before = int(hours_before)
    elif ((hours_before == '12') and (AM_PM_str == 'A')):
        hours_before = 0
    else:
        hours_before = int(hours_before)
    parsed_time = [hours_before, minutes_before]
    return parsed_time

# get time objects for wake up times

val_times = []
parsed_times = []
for i in range(len(wake_up_times)):
    parsed_time_day = parseTime(wake_up_times[i])
    hours, minutes = parsed_time_day[0:2]
    now_day = time.localtime()
    now_obj_mk = time.mktime((now_day[0], now_day[1], now_day[2], hours,
                               minutes, now_day[5], i, now_day[7], now_day[8]))
    time_obj = time.localtime(now_obj_mk)
    val_times.append(now_obj_mk)
parsed_times.append(time_obj)

# determine which day it is and print which time waking up on screen

def whichDay():
    now = time.localtime()
    current_day = now[6]
    now_mk = time.mktime((now[0], now[1], now[2], now[3], now[4], now[5], now[6],
                          now[7], now[8]))
    if now_mk < val_times[current_day]:
        # if it's after midnight and before todays wakeup time, display the wake up time of today
        for day in range(len(wake_up_times)):
            if now_mk < val_times[day]:
                # print the wake up time for that day
```
if current_day == day:
    input_wake_up_time = wake_up_times[day]
    use_day = day
    # set wake up time to the next day's wake up time after current day's wake up time
else:
    if current_day == 6:
        input_wake_up_time = wake_up_times[0]
        use_day = 0
    else:
        if current_day == day:
            input_wake_up_time = wake_up_times[day+1]
            use_day = day + 1
        input_wake_up_time_text = "Wake up " + days_str[use_day] + " at " + input_wake_up_time
    wakeup_time_textarea.text = input_wake_up_time_text
    return use_day

def displayTime():
    now = time.localtime()
    hour, minute = now[3:5]
    print(now)
    print("Current time: %02d:%02d" % (hour, minute))
    formatTime(hour, minute)
    time_textarea.text = formatTime(hour, minute)
    return formatTime(hour, minute)

def formatTime(raw_hours, raw_minutes):
    # display the time in a nice big font
    format_str = "%d:%02d"
    if raw_hours >= 12:
        raw_hours -= 12
        format_str = format_str + "P"
    else:
        format_str = format_str + "A"
    if raw_hours == 0:
        raw_hours = 12
    time_str = format_str % (raw_hours, raw_minutes)
    return time_str

def backLight():
    now = time.localtime()
    now_val = time.mktime((now[0], now[1], now[2], now[3], now[4], now[5], now[6], now[7], now[8]))
    wake_up_day_val = val_times[now[6]]
    # if time is more than 9 hours after current day's wake up time,
    # or time is before light start time, backlight off, tap to turn on
    if (now_val - wake_up_day_val) > 32400 or (now_val - wake_up_day_val) < -1800:
        pyportal.set_backlight(backlight_off)
        if pyportal.touchscreen.touch_point:
            pyportal.set_backlight(backlight_on)
            time.sleep(5)
            pyportal.set_backlight(backlight_off)
    else:
        pyportal.set_backlight(backlight_on)

def subtract30min(day): # subtract 30 min
    # get the time object from the corresponding day
    raw_wake_up_time = parsed_times[day]
    now = time.localtime()  
    # new time subtracting 30 min from wake up time
    minus30 = time.mktime((now[0], now[1], now[2], raw_wake_up_time[3], raw_wake_up_time[4] - 30, now[5], now[6], now[7], now[8]))
    time_minus30 = time.localtime(minus30)
    hour_minus30 = time_minus30[3]
    minutes_minus30 = time_minus30[4]
    light_on_time_textarea.text = "Light starting at: " + formatTime(hour_minus30, minutes_minus30)
return formatTime(hour_minus30, minutes_minus30)

refresh_time = None

while True:
    time_now = time.localtime()
    # only query the online time once per hour (and on first run)
    if (not refresh_time) or (time.monotonic() - refresh_time) > 3600:
        try:
            print("Getting time from internet!")
            pyportal.get_local_time()
            refresh_time = time.monotonic()
        except RuntimeError as e:
            print("Some error occurred, retrying! -", e)
            continue
    time_str_text = displayTime()
    print(time_str_text)
    # determine which wake up time to choose based on the day
    wake_up_day = whichDay()
    # if time is more than 9 hours after previous day's wake up time,
    # backlight off and can tap to turn on
    backLight()
    # start the light 30 min before wake up time
    start_light_time = subtract30min(wake_up_day)
    # If current day is same as wake up day and
    # wake up time - 30 minutes equals current time, start the light
    if wake_up_day == time_now[6] and time_str_text == start_light_time:
        print("Starting wake up light")
        # turn on backlight
        pyportal.set_backlight(backlight_on)
        for i in range(light_minutes - 1):
            BRIGHTNESS = BRIGHTNESS + (MAX_BRIGHTNESS/light_minutes) # max 0.25,
            min 0.0
            strip.fill(WHITE)
            strip.brightness = BRIGHTNESS
            displayTime()
            time.sleep(60) # 60 for once per min
        while not pyportal.touchscreen.touch_point: # turn strip off
            displayTime()
            time.sleep(1)
            continue
        strip.brightness = MIN_BRIGHTNESS
    # update every second so that screen can be tapped to view time
    time.sleep(1)

Downloading the libraries

Make sure to add the necessary libraries to the lib folder, info on how to do this
 can be found in the "PyPortal CircuitPython Setup" section
This project uses the following CircuitPython libraries:

- adafruit_bitmap_font (directory)
- adafruit_bus_device (directory)
- adafruit_display_shapes (directory)
- adafruit_display_text (directory)
- adafruit_esp32spi (directory)
- adafruit_imageload (directory)
- adafruit_io (directory)
- adafruit_pyportal.mpy (file)
- adafruit_requests (file)
- adafruit_touchscreen.mpy (file)
- neopixel.mpy (file)

This is what the final contents of the CIRCUITPY drive will look like:
Using the Wake-Up Light

Here's an example of the wake-up light in use (for a late riser...)

First enter in the time you want to wake up each day of the week and save the code.py file.

```
# type in time to get up each day of the week
default_wake_up = "12:10P"
up_time_monday = default_wake_up
up_time_tuesday = default_wake_up
up_time_wednesday = default_wake_up
up_time_thursday = default_wake_up
up_time_friday = default_wake_up
up_time_saturday = "1:00P"
up_time_sunday = "1:00P"
```

That's it! The program does the rest of the work for you!

Before wake-up time

Current Time: 11:39AM

Wake-up time: 12:10PM

Light starting at 11:40AM
If the current time is before the "light on" time, the display's backlight should be off unless activated by touch, in which case the backlight comes on for 5 seconds then turns back off.

During wake-up time

Current Time: 11:40AM

Wake-up time: 12:10PM

Light starting at 11:40AM

If the current time = the "light on" time, the strip will turn on and get brighter every minute for 30 minutes until the wake-up time is reached.
After wake-up time

Current Time: 12:11PM

Wake-up time: 12:10PM

Light starting at 11:40AM

After the wake-up time is reached, tap the screen to turn off the strip and the next day's wake-up time will be displayed on the screen.

How it works

The wake-up light does quite a few tricks to deliver light at your wake-up time.
Displaying text

When the backlight is on, we see the current time, wake-up time and light-starting time displayed on the screen.

The fonts, color and locations of these text elements are initialized at the beginning of the program.

```python
# assign fonts
big_font = bitmap_font.load_font(cwd+"/fonts/Nunito-Light-75.bdf")
big_font.load_glyphs(b'0123456789:AP') # pre-load glyphs for fast printing
print('loading fonts...')
info_font = bitmap_font.load_font(cwd+"/fonts/Nunito-Black-17.bdf")
info_font.load_glyphs(b'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ-,.:/'"

time_color = 0xFFFFFF
time_position = (75,130)
time_textarea = Label(big_font, color=time_color,
                      x=time_position[0], y=time_position[1])

wakeup_time_color = 0xFFFFFF
wakeup_time_position = (15,200)
wakeup_time_textarea = Label(info_font, color=wakeup_time_color,
                             x=wakeup_time_position[0], y=wakeup_time_position[1])

light_on_time_color = 0xFFFFFF
light_on_time_position = (15,220)
light_on_time_textarea = Label(info_font, color=light_on_time_color,
                               x=light_on_time_position[0],
y=light_on_time_position[1])

pyportal.splash.append(time_textarea)
pyportal.splash.append(wakeup_time_textarea)
pyportal.splash.append(light_on_time_textarea)
```
Text position

The PyPortal's display is 320 pixels wide and 240 pixels high. In order to refer to those positions on the screen, we use an x/y coordinate system, where x is horizontal and y is vertical.

The origin of this coordinate system is the upper left corner. This means that a pixel placed at the upper left corner would be (0,0) and the lower right corner would be (320, 240).

![Coordinate System](image)

Font

The fonts used here are bitmap fonts made from the Nunito typeface. You can learn more about [converting type in this guide](https://).  

Text color

Customize your wake-up light by changing the text colors! You will need to use the hexadecimal value for any color you want to display.

You can use something like [https://htmlcolorcodes.com/](https://htmlcolorcodes.com/) to pick your color and then copy the hex value.
Initializing wake-up times

This part of the program converts all the user-entered wake up times into time "objects" that can be easily manipulated later to add / subtract minutes, hours and days to the times!

```python
# parse given time string into hour minute and AM_PM elements
def parseTime(time_before):
    hours_before, minutes_before = time_before.split(":")
    AM_PM_str = minutes_before[-1:]
    minutes_before = int(minutes_before[:-1])
    if (hours_before != '12') and AM_PM_str == 'P':
        hours_before = int(hours_before) + 12
    elif ((hours_before == '12') and (AM_PM_str == 'P')):
        hours_before = int(hours_before)
    elif ((hours_before == '12') and (AM_PM_str == 'A')):
        hours_before = 0
    else:
        hours_before = int(hours_before)
    parsed_time = [hours_before, minutes_before]
    return parsed_time

# get time objects for wake up times
val_times = []
parsed_times = []
for i in range(len(wake_up_times)):
    parsed_time_day = parseTime(wake_up_times[i])
    hours, minutes = parsed_time_day[0:2]
    now_day = time.localtime()
    time_obj_mk = time.mktime((now_day[0], now_day[1], now_day[2], hours, minutes, now_day[5], i, now_day[7], now_day[8]))
    time_obj = time.localtime(time_obj_mk)
    val_times.append(time_obj_mk)
    parsed_times.append(time_obj)
```

Main loop

```python
while True:

In the repeating main loop, the following actions are performed:

1. Declare a variable to reference local time.
2. Query local time on first run and once an hour after.
3. Display current time to screen.
4. Determine which wake-up time to choose based on the current day.
5. Turn on/off backlight depending on time.
6. Get the time that is 30 min before the chosen wake-up time.
7. If current day is same as wake-up day and wake-up time - 30 minutes equals current time, start the strip light.
8. Increase brightness of light once a min for 30 min until light has reached maximum brightness. Also update the time.
9. After wake-up time is reached, if user taps screen, turn off strip light.
```
while True:
    time_now = time.localtime()
    # only query the online time once per hour (and on first run)
    if (not refresh_time) or (time.monotonic() - refresh_time) > 3600:
        try:
            print("Getting time from internet!")
            pyportal.get_local_time()
            refresh_time = time.monotonic()
        except RuntimeError as e:
            print("Some error occurred, retrying! -", e)
            continue
    time_str_text = displayTime()
    print(time_str_text)
    # determine which wake up time to choose based on the day
    wake_up_day = whichDay()
    # if time is more than 9 hours after previous day's wake up time,
    # backlight off and can tap to turn on
    backLight()
    # start the light 30 min before wake up time
    start_light_time = subtract30min(wake_up_day)
    # If current day is same as wake up day and
    # wake up time - 30 minutes equals current time, start the light
    if wake_up_day == time_now[6] and time_str_text == start_light_time:
        print("Starting wake up light")
        # turn on backlight
        pyportal.set_backlight(backlight_on)
        for i in range(light_minutes - 1):
            BRIGHTNESS = BRIGHTNESS + (MAX_BRIGHTNESS/light_minutes) # max 0.25,
            min 0.0
            strip.fill(WHITE)
            strip.brightness = BRIGHTNESS
            displayTime()
            time.sleep(60) # 60 for once per min
        while not pyportal.touchscreen.touch_point: # turn strip off
            displayTime()
            time.sleep(1)
            continue
    strip.brightness = MIN_BRIGHTNESS
    # update every second so that screen can be tapped to view time
    time.sleep(1)
Affixing the Strip

Attaching strip to bed frame or other furniture

Cut 5 or more pieces of double sided foam tape.
Take off one side of a piece, add some hot glue to it and stick to the NeoPixel strip on one end.
Repeat with the rest of the pieces trying to make each piece equidistant to the next.
Reinforce pieces with extra hot glue if desired.
Now take the other sides of the foam pieces off exposing the sticky sides.
Carefully place behind bed frame or another convenient bedroom object.
Attach alligator clips to the ends of the strip to lengthen reach to PyPortal on a night stand.
Use a piece of tape to hold alligator clips in place.
Connect clips to JST wires on PyPortal.
All done, now go to bed! And get ready to be waken up by light...

Adding features

Want to take this light to the next level? What features can you add to make it even better?
Some initial ideas:

- Setting the wake-up time using the touch screen.
- Ability to change the color of light you wake up to.
- Ability to change between 12 hour and 24 hour time.
- Add some fun sounds to wake up to once the light has reached full brightness!