PyPortal Voice Controlled Smart Switch and Time Display

Created by Dan Cogliano

https://learn.adafruit.com/pyportal-voice-controlled-smart-switch-and-time-display

Last updated on 2024-06-03 02:57:23 PM EDT

Table of Contents

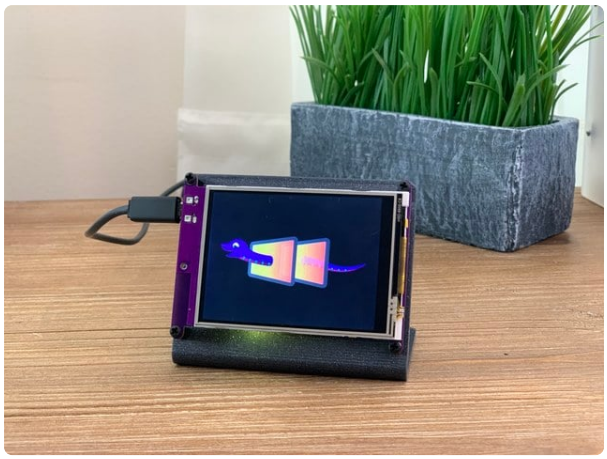
Overview

How would you like a PyPortal that nicely displays the current date and time? How about using a PyPortal's touch screen to turn on and off lights or appliances? Or, how about a PyPortal that is internet connected so you can turn on and off lights or appliances using your smartphone? Even still, how about a PyPortal that can be controlled using your voice with Alexa or Google Assistant to turn on and off lights or appliances? What would you think of a PyPortal that does all of this in a single device (the the ghost of Steve Jobs is speaking)? Yes, with this project you can do all of this with a PyPortal!

Best of all, this project has no soldering - you only need to plug in a cable from the PyPortal to the IoT relay!

There are several secrets to this magic. The IoT relay module allows us to control appliances using a PyPortal quite easily without worrying about high voltage wiring. Another secret is the use of the PyPortal ESP32 WiFi coprocessor, which we use to connect to the internet and Adafruit IO. We then use the site IFTTT.com (you can remember it as "**IF** **T**his **T**hen **T**hat") to integrate Adafruit IO with Amazon's Alexa and Google Assistant. The PyPortal touch screen allows us to turn the light on or off simply by touching the screen. As a bonus, the display dims itself in a dark room to be less obtrusive.

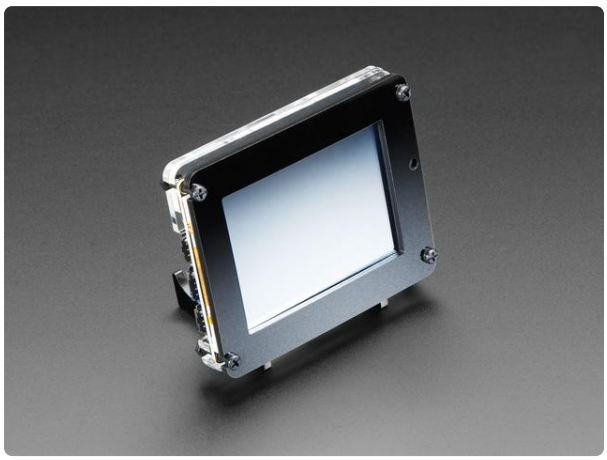
Parts



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



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...

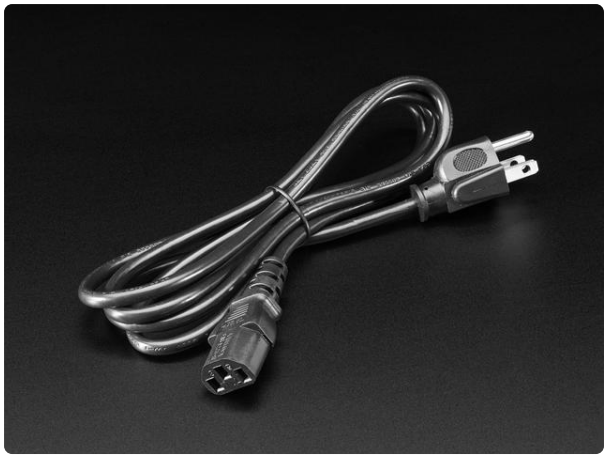
https://www.adafruit.com/product/4146

Controllable Four Outlet Power Relay Module version 2

Say goodbye to hazardous high voltage wiring and create the Internet of Things with safe, reliable power control....

https://www.adafruit.com/product/2935

The power cord that comes with the power relay module is short. If you don't want to use an extension cord, you can purchase this longer power cord in the Adafruit shop.



[Power Cord Cable w/ 3 Conductor PC Power Connector Socket]
The standard 3 prong wall plug (NEMA 5-15p) to PC connector (IEC C13) cable was made commonplace by PC power supplies. Today, they can be found providing power connections for many...
https://www.adafruit.com/product/3311

[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

Connecting the PyPortal

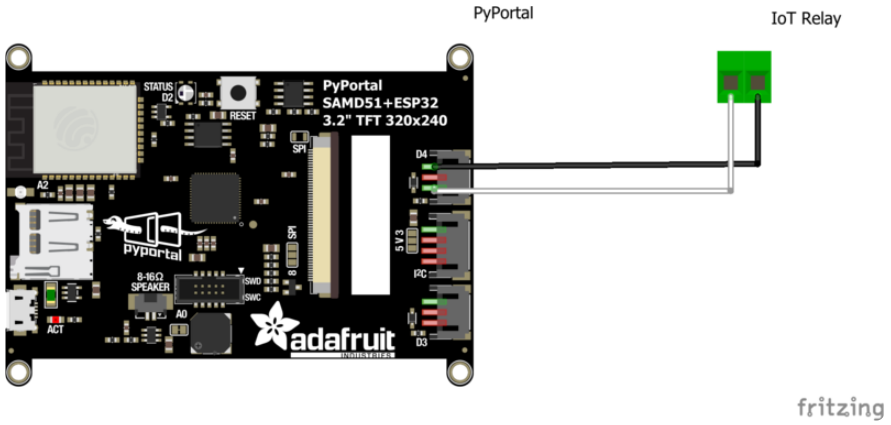
The PyPortal needs to connect to an IoT Relay in order to control power to the outlets. The IoT relay has 4 outlets:

- 1 outlet normally on
- 2 outlets normally off
- 1 outlet always on

Connect your light or appliance to the outlet marked "Normally Off" and connect the PyPortal USB adapter to the outlet marked "Always On". This allows the PyPortal to be plugged into the same IoT relay as the device it is controlling without turning itself off.

You will need to connect the PyPortal to the IoT Relay using the D4 connector. See the diagram below for the wiring. This connector uses 3 wires (white, red and black) but we only use two of them (white and black). Make sure the black wire is the topmost wire on the PyPortal and the rightmost wire connected to he IoT Relay. If you want to extend the wires so the IoT Relay can be placed further away, then read this learning guide (https://adafru.it/HA1) for some tips about how to solder wires together for a

battery switch, where the techniques explained there will work for extending the wires.



The PyPortal needs to connect to the internet to control the switch remotely. Read the next section on configuring the PyPortal for the internet using the **secrets.py** file and installing the CircuitPython files.

CircuitPython Code

Update CircuitPython

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Egk) for your board.

See the guide page Install CircuitPython (https://adafru.it/EnM) in the Adafruit Learning System guide Adafruit PyPortal (https://adafru.it/HA2).

Grab the Project Files

Once you have your CircuitPython libraries installed, you can grab the rest of the project files. Use the **Download Project Bundle** button in the code window below to download the code, fonts, background bitmap and a starter **secrets.py** file.

Copy the contents of the zip file to your PyPortal **CIRCUITPY** drive.

Code

```
# SPDX-FileCopyrightText: 2019 Dan Cogliano for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
```

```
import gc
import board
import busio
import adafruit_connection_manager
from adafruit_esp32spi import adafruit_esp32spi, adafruit_esp32spi_wifimanager
import adafruit_requests
import digitalio
import analogio
from adafruit_pyportal import PyPortal
from adafruit_display_shapes.circle import Circle
from adafruit_display_shapes.roundrect import RoundRect
from adafruit_bitmap_font import bitmap_font
from adafruit_display_text import label
import adafruit_touchscreen

from adafruit_minimqtt import MQTT

DISPLAY_COLOR = 0x006600
SWITCH_COLOR = 0x008800
SWITCH_FILL_COLOR = 0xffffff

# Switch location
SWITCHX = 260
SWITCHY = 4

FEED_NAME = "pyportal-switch"

months = ["JAN", "FEB", "MAR", "APR", "MAY", "JUN",
          "JUL", "AUG", "SEP", "OCT", "NOV", "DEC"]


def get_local_timestamp(location=None):
    # pylint: disable=line-too-long
    """Fetch and "set" the local time of this microcontroller to the local time at
the location, using an internet time API.
    :param str location: Your city and country, e.g. ``"New York, US"``.
    """
    # pylint: enable=line-too-long
    api_url = None
    try:
        aio_username = secrets['aio_username']
        aio_key = secrets['aio_key']
    except KeyError:
        raise KeyError("\n\nOur time service requires a login/password to rate-
limit. Please register for a free adafruit.io account and place the user/key in
your secrets file under 'aio_username' and 'aio_key'")# pylint: disable=line-too-
long

    location = secrets.get('timezone', location)
    if location:
        print("Getting time for timezone", location)
        api_url = (TIME_SERVICE + "&tz=%s") % (aio_username, aio_key, location)
    else: # we'll try to figure it out from the IP address
        print("Getting time from IP address")
        api_url = TIME_SERVICE % (aio_username, aio_key)
    api_url += TIME_SERVICE_TIMESTAMP
    try:
        print("api_url:",api_url)
        response = requests.get(api_url)
        times = response.text.split(' ')
        seconds = int(times[0])
        tzoffset = times[1]
        tzhours = int(tzoffset[0:3])
        tzminutes = int(tzoffset[3:5])
        tzseconds = tzhours * 60 * 60
        if tzseconds < 0:
            tzseconds -= tzminutes * 60
        else:
            tzseconds += tzminutes * 60
```

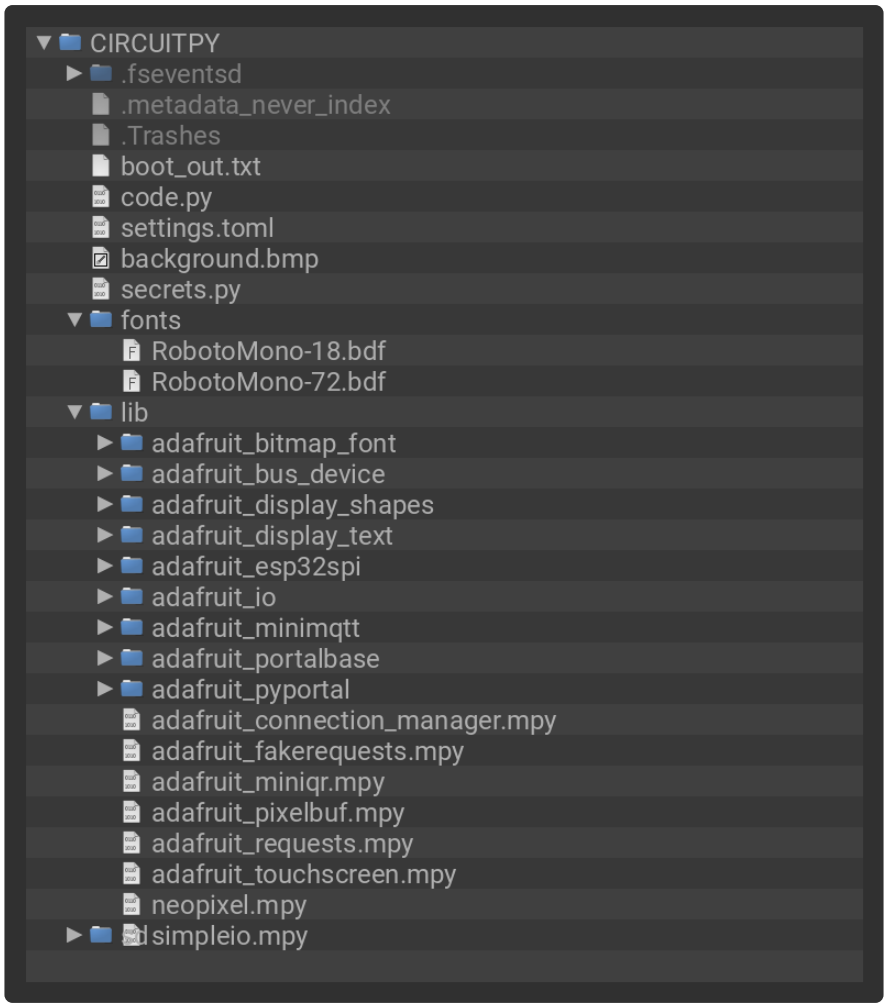
```
        print(seconds + tzseconds, tzoffset, tzhours, tzminutes)
    except KeyError:
        raise KeyError("Was unable to lookup the time, try setting
secrets['timezone'] according to http://worldtimeapi.org/timezones")  # pylint:
disable=line-too-long

    # now clean up
    response.close()
    response = None
    gc.collect()
    return int(seconds + tzseconds)

def create_text_areas(configs):
    """Given a list of area specifications, create and return text areas."""
    text_areas = []
    for cfg in configs:
        textarea = label.Label(cfg['font'], text=' '*cfg['size'])
        textarea.x = cfg['x']
        textarea.y = cfg['y']
        textarea.color = cfg['color']
        text_areas.append(textarea)
    return text_areas

class Switch(object):
    def __init__(self, pin, my_pyportal):
        self.switch = digitalio.DigitalInOut(pin)
        self.switch.direction = digitalio.Direction.OUTPUT
        rect = RoundRect(SWITCHX, SWITCHY, 31, 60, 16, outline=SWITCH_COLOR,
                         fill=SWITCH_FILL_COLOR, stroke=3)
        my_pyportal.splash.append(rect)
        self.circle_on = Circle(SWITCHX + 15, SWITCHY + 16, 10,
fill=SWITCH_FILL_COLOR)
        my_pyportal.splash.append(self.circle_on)
        self.circle_off = Circle(SWITCHX + 15, SWITCHY + 42, 10, fill=DISPLAY_COLOR)
        my_pyportal.splash.append(self.circle_off)

    # turn switch on or off
    def enable(self, enable):
        print("turning switch to ", enable)
        self.switch.value = enable
        if enable:
            self.circle_off.fill = SWITCH_FILL_COLOR
            self.circle_on.fill = DISPLAY_COLOR
        else:
            self.circle_on.fill = SWITCH_FILL_COLOR
            self.circle_off.fill = DISPLAY_COLOR

    def toggle(self):
        if self.switch.value:
            self.enable(False)
        else:
            self.enable(True)

    def status(self):
        return self.switch.value

# you'll need to pass in an io username and key
TIME_SERVICE = "http://io.adafruit.com/api/v2/%s/integrations/time/strftime?x-aio-
key=%s"
# See https://apidock.com/ruby/DateTime/strftime for full options
TIME_SERVICE_TIMESTAMP = '&fmt=%25s+%25z'

class Clock(object):
    def __init__(self, my_pyportal):
        self.low_light = False
        self.update_time = None
        self.snapshot_time = None
        self.pyportal = my_pyportal
        self.current_time = 0
```

```
        self.light = analogio.AnalogIn(board.LIGHT)
        text_area_configs = [dict(x=0, y=105, size=10, color=DISPLAY_COLOR,
font=time_font),
                             dict(x=260, y=153, size=3, color=DISPLAY_COLOR,
font=ampm_font),
                             dict(x=110, y=40, size=20, color=DISPLAY_COLOR,
font=date_font)]
        self.text_areas = create_text_areas(text_area_configs)
        self.text_areas[2].text = "starting..."
        for ta in self.text_areas:
            self.pyportal.splash.append(ta)

    def adjust_backlight(self, force=False):
        """Check light level. Adjust the backlight and background image if it's
dark."""
        if force or (self.light.value >= 1500 and self.low_light):
            self.pyportal.set_backlight(1.00)
            self.low_light = False
        elif self.light.value <= 1000 and not self.low_light:
            self.pyportal.set_backlight(0.1)
            self.low_light = True

    def tick(self, now):
        self.adjust_backlight()
        if (not self.update_time) or ((now - self.update_time) >= 300):
            # Update the time
            print("update the time")
            self.update_time = int(now)
            self.snapshot_time = get_local_timestamp(secrets['timezone'])
            self.current_time = time.localtime(self.snapshot_time)
        else:
            self.current_time = time.localtime(int(now) - self.update_time +
self.snapshot_time)
        hour = self.current_time.tm_hour
        if hour > 12:
            hour = hour % 12
        if hour == 0:
            hour = 12
        time_string = '%2d:%02d' % (hour,self.current_time.tm_min)
        self.text_areas[0].text = time_string
        ampm_string = "AM"
        if self.current_time.tm_hour >= 12:
            ampm_string = "PM"
        self.text_areas[1].text = ampm_string
        self.text_areas[2].text = (months[int(self.current_time.tm_mon - 1)] +
                                   " " + str(self.current_time.tm_mday))
        try:
            board.DISPLAY.refresh(target_frames_per_second=60)
        except AttributeError:
            board.DISPLAY.refresh_soon()
            board.DISPLAY.wait_for_frame()

# Define callback methods which are called when events occur
# pylint: disable=unused-argument, redefined-outer-name
def connected(client, userdata, flags, rc):
    # This function will be called when the client is connected
    # successfully to the broker.
    onoff_feed = secrets['aio_username'] + '/feeds/' + FEED_NAME
    print('Connected to Adafruit IO! Listening for topic changes on %s' %
onoff_feed)
    # Subscribe to all changes on the onoff_feed.
    client.subscribe(onoff_feed)

def disconnected(client, userdata, rc):
    # This method is called when the client is disconnected
    print('Disconnected from Adafruit IO!')

def message(client, topic, message):
    # This method is called when a topic the client is subscribed to
```

```
    # has a new message.
    print('New message on topic {0}: {1}'.format(topic, message))
    if message in ("ON","TRUE","1"):
        switch.enable(True)
    else:
        switch.enable(False)

##########################################

try:
    from secrets import secrets
except ImportError:
    print("""WiFi settings are kept in secrets.py, please add them there!
the secrets dictionary must contain 'ssid' and 'password' at a minimum""")
    raise

esp32_cs = digitalio.DigitalInOut(board.ESP_CS)
esp32_ready = digitalio.DigitalInOut(board.ESP_BUSY)
esp32_reset = digitalio.DigitalInOut(board.ESP_RESET)

WIDTH = board.DISPLAY.width
HEIGHT = board.DISPLAY.height

ts = adafruit_touchscreen.Touchscreen(board.TOUCH_XL, board.TOUCH_XR,
                                      board.TOUCH_YD, board.TOUCH_YU,
                                      calibration=(
                                          (5200, 59000),
                                          (5800, 57000)
                                          ),
                                      size=(WIDTH, HEIGHT))

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset,
debug=False)
pool = adafruit_connection_manager.get_radio_socketpool(esp)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(esp)
requests = adafruit_requests.Session(pool, ssl_context)

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])

pyportal = PyPortal(esp=esp,
                    external_spi=spi,
                    default_bg="/background.bmp")

ampm_font = bitmap_font.load_font("/fonts/RobotoMono-18.bdf")
ampm_font.load_glyphs(b'ampAMP')
date_font = bitmap_font.load_font("/fonts/RobotoMono-18.bdf")
date_font.load_glyphs(b'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789')
time_font = bitmap_font.load_font("/fonts/RobotoMono-72.bdf")
time_font.load_glyphs(b'0123456789:')

clock = Clock(pyportal)

for ap in esp.scan_networks():
    print("\t%s\t\tRSSI: %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 RuntimeError 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))
```

```
wifi = adafruit_esp32spi_wifimanager.ESPSPI_WiFiManager(
    esp, secrets, debug = True)


# Set up a MiniMQTT Client
mqtt_client = MQTT(broker='io.adafruit.com',
                   username=secrets['aio_username'],
                   password=secrets['aio_key'],
                   network_manager=wifi,
                   socket_pool=pool,
                   ssl_context=ssl_context)

mqtt_client.on_connect = connected
mqtt_client.on_disconnect = disconnected
mqtt_client.on_message = message

mqtt_client.connect()

switch = Switch(board.D4, pyportal)

second_timer = time.monotonic()
while True:
    #time.sleep(1)
    p = ts.touch_point
    if p:
        #if p[0] >= 140 and p[0] <= 170 and p[1] >= 160 and p[1] <= 220:
        # touch anywhere on the screen
        print("touch!")
        clock.adjust_backlight(True)
        switch.toggle()
        time.sleep(1)

    # poll once per second
    if time.monotonic() - second_timer >= 1.0:
        second_timer = time.monotonic()
        # Poll the message queue
        try:
            mqtt_client.loop()
        except RuntimeError:
            print("reconnecting wifi")
            mqtt_client.reconnect_wifi()

        # Update the PyPortal display
        clock.tick(time.monotonic())
```

When all the files are loaded onto the PyPortal **CIRCUITPY** drive, the following files should be on there:



Library Files

Before continuing, please make sure these files and folders are in your board's **lib** folder.

- **adafruit_bitmap_font**
- **adafruit_bus_device**
- **adafruit_display_shapes**
- **adafruit_display_text**
- **adafruit_esp32spi**
- **adafruit_io**
- **adafruit_logging.mpy**
- **adafruit_minimqtt.mpy**
- **adafruit_pyportal.mpy**
- **adafruit_requests.mpy**
- **adafruit_touchscreen.mpy**
- **neopixel.mpy**

Secrets File Setup

Next is to get the PyPortal connected to the internet. To do this, we need to create a secrets file. If you have not yet set up a **secrets.py** file in your **CIRCUITPY** drive and connected to the internet using it, follow this guide and come back when you've successfully connected to the internet (https://adafru.it/Eao).

Using Mu (https://adafru.it/HA8) or any text editor, you should add your Adafruit IO Username and Adafruit IO Key to the **secrets.py** file. This project connects to Adafruit IO to retrieve the current date and time as well as integrating voice commands using Amazon Alexa and Google Assistant. Adafruit IO is free to use, but you'll need to log in with your Adafruit account to use it. If you don't already have an Adafruit login, create one here (https://adafru.it/dAQ).

If you haven't used Adafruit IO before, check out this guide for more info (https://adafru.it/Ef8).

Once you have logged into your account, there are two pieces of information you'll need to place in your **secrets.py** file: Adafruit IO username, and Adafruit IO key. Head to io.adafruit.com (https://adafru.it/fsU) and simply click the **View AIO Key** link on the left hand side of the Adafruit IO page to get this information.

Then, update the `aoi_username` and `aio_key` values in the **secrets.py** file.
Your **secrets.py** file should look like this:

```
Temporarily unable to load content:
```

Fonts

Two fonts are used to display the date and time on the PyPortal. You will need to create a **fonts** folder on your CircuitPython **CIRCUITPY** drive and Ensure above you downloaded the font files from the project's GitHub repository

The Background Image

You will also need a background image for the display. You can use the one included in the project. This background is in the public domain and can be found at here (https://adafru.it/HA6), as well as other similar backgrounds.

The **background.bmp** file is placed in the main (root) directory of the **CIRCUITPY** drive.

In the next pages, we will connect to Adafruit IO for remote controlling the appliance.

Adafruit IO Configuration

The project as is works great - it has a nice date and time display with a touch screen switch. However, when combined with Adafruit IO, it can control a connected light or appliance through the internet or voice command. This is done using an Adafruit IO feed. This feed is then used to create a virtual switch. This web based button allows you to turn on and off your smart switch remotely. Later this will be extended to voice commands using Alexa or a Google Assistant. But first is to get the switch working in Adafruit.io.

Read the Welcome to Adafruit IO learning guide (https://adafru.it/BRB) to get learn about the features of Adafruit IO.

Integrating Adafruit IO with CircuitPython

Earlier you created an Adafruit.io account and entered your Adafruit.io credentials `aio_username` and `aio_key` into the **secrets.py** file. Now to create a virtual switch using Adafruit.io. This requires creating a feed at Adafruit.io . The feed name is hard-coded into **code.py** as `pyportal-switch`. You can change this if needed, particularly if you creating more than one switch when you will need different names.

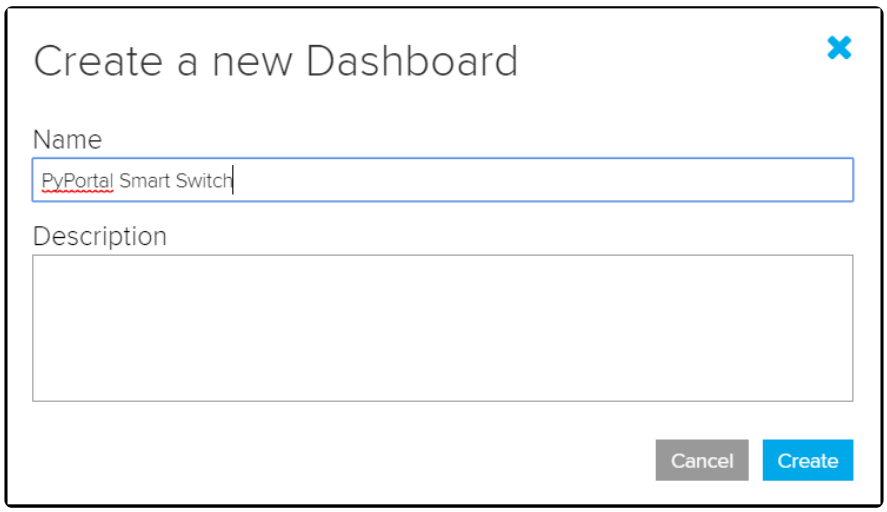
Creating a New Feed

To create a new feed, navigate to the feed list using "Feeds | View All". Create a new feed from the "Actions" menu and name it "PyPortal Switch". Verify the feed key is `pyportal-switch` since this is the name the PyPortal uses, not the name.

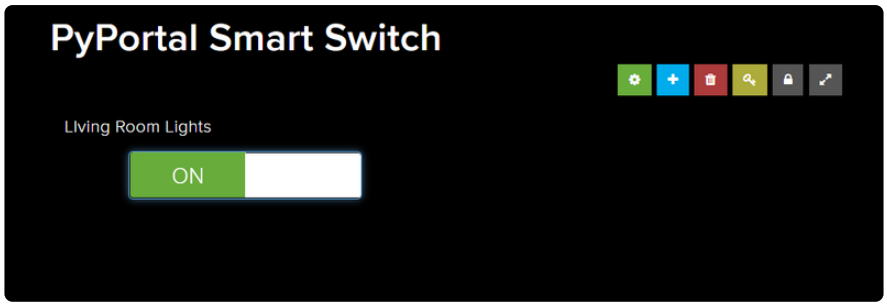
Once the feed is created, next is to create a virtual switch using an Adafruit.io dashboard. This is not necessary for use with Alexa or Google Assistant, but it is a easy method to test the feed using a web browser.

Creating a New Dashboard

Navigate to "Dashboards | View All", create a new dashboard. You can name it whatever you would like. For our guide we named it "PyPortal Smart Switch".



Now to create the switch by clicking the blue plus sign icon on the right side of the dashboard. Create a toggle block from the list of block types. Then, select the "PyPortal Switch" feed from the "Choose Feed" dialog and proceed to the next step by clicking the "next step" button. Finally, give this block a descriptive name and keep the default button text values. Click the "create block" button to save the button.



To verify the switch and the underlying feed is working correctly, you can click on the "ON" or "OFF" button that appears on the dashboard. If your PyPortal is correctly running the **code.py** file and the proper Adafruit.io credentials were entered, You should see your PyPortal's switch icon change as the value is changed on Adafruit.io. If you have a light or appliance plugged in to the relay switch, it should also turn on or off when the virtual button is pressed.

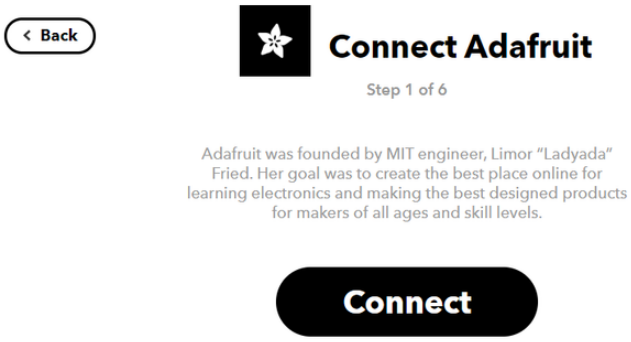
You now have a web controlled smart switch. You can turn your appliance on or off from a web browser on your PC or smartphone. Continue to the next section to configure you smart switch for voice commands using either Amazon Alexa or Google Assistant.

IFTTT Configuration

OK, so we have a functioning time display using Adafruit.io and the ability to turn your device on or off using the PyPortal touch screen or from a web page. We could be done here, this is a functional product as it stands. However, we just need one more step to enable the PyPortal to act on voice commands from an Amazon Alexa or Google Assistant. The magic to do this is done using the web site **ifttt.com**. This site, short for "If This Then That", connects the voice device with Adafruit IO. Configure one or both if you happen to have both types of voice assistants.

To begin, click the "Explore" button in the upper right of the page on the IFTTT web site. Next, click the "+" button next to "Make your own applets from scratch".

The first time you connect to a service like Amazon Alexa, Google Assistant, or Adafruit IO, you will be prompted to connect to that service with your login name and password.



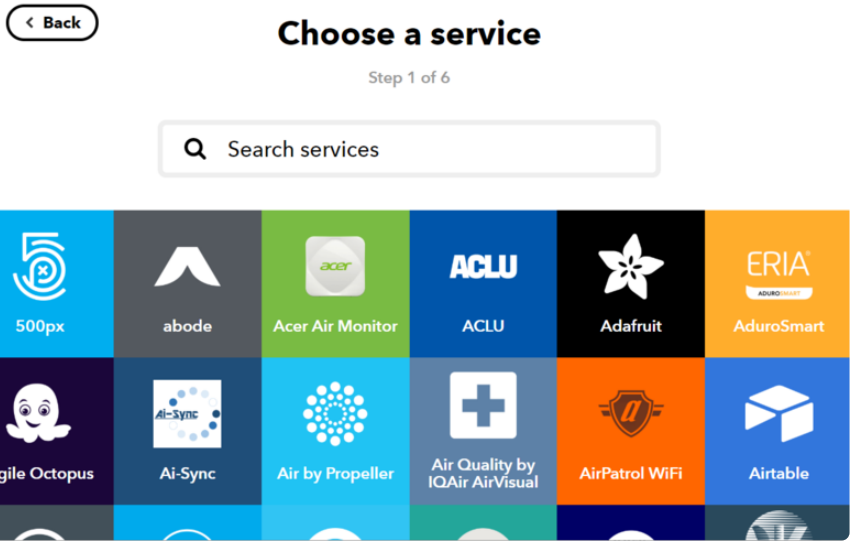
Depending on the device you are using, continue with either the "Alexa Setup" or "Google Assistant Setup" page.

Alexa Setup

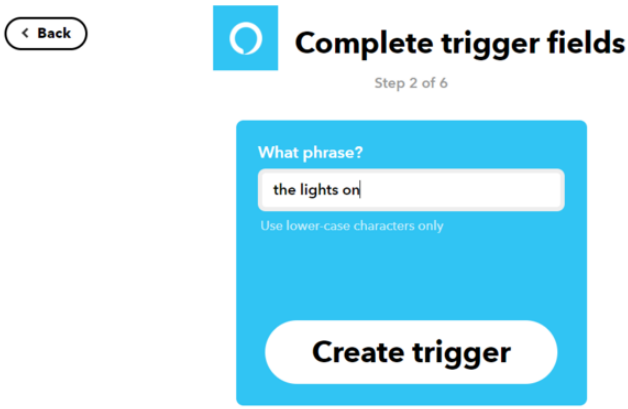
Here are the steps to configure Amazon Alexa with IFTTT. You will need to create two applets, one to turn on the lights and the other to turn off the lights.

Click "This" to create the first half of the applet.

Search for and select "Amazon Alexa". Don't be tempted to select Adafruit yet! We will be selecting that in the second half of the applet.

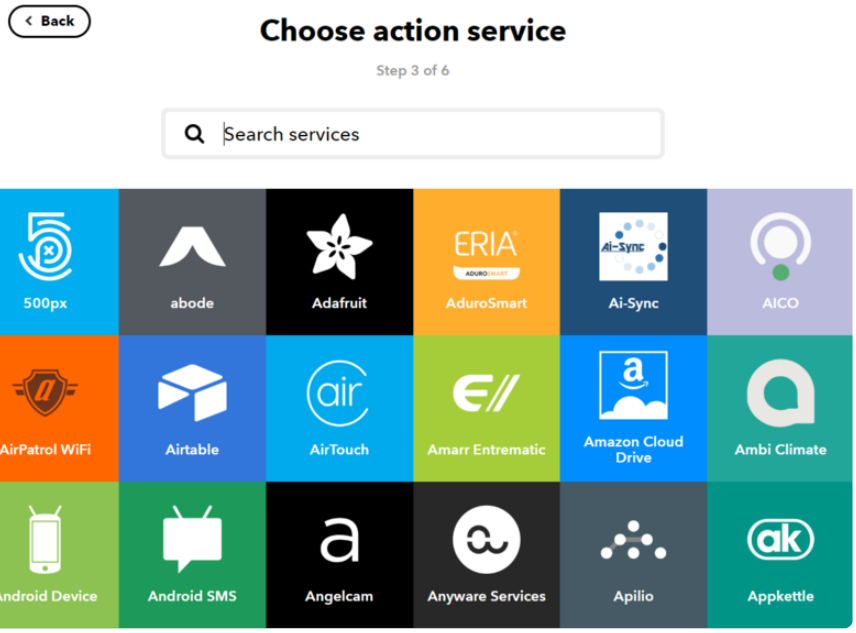


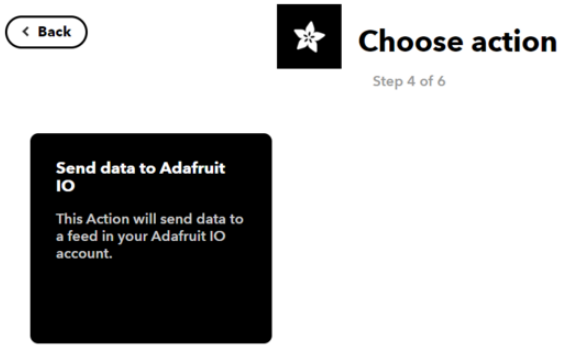
Click on the "Say a specific phrase" trigger. Then enter the phrase "the lights on" in the trigger fields. IFTT requires phrase to start with "Alexa trigger", so adding this text will complete the phrase with "Alexa trigger the lights on".



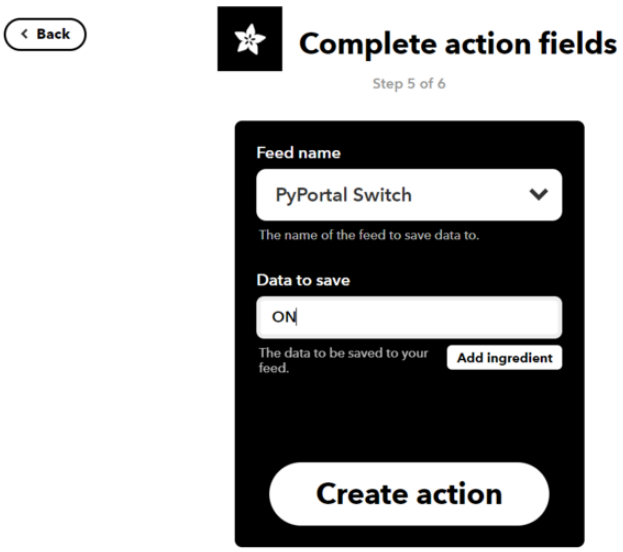
Click "That" to create the second half of the applet. Notice the Amazon Alexa icon in the "If" part of the applet.

Here is where we connect our applet to Adafruit IO. Select the "Adafruit" icon from the list of action services, and then select the action, "Send data to Adafruit IO".

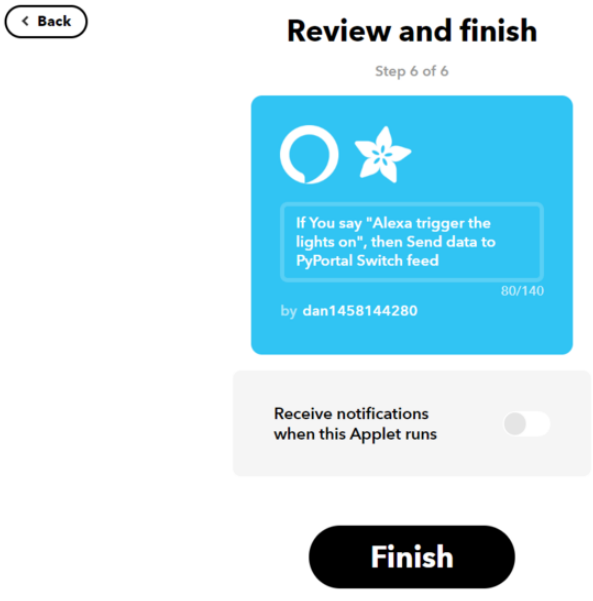




Pick the feed we created earlier on Adafruit IO, "PyPortal Switch". For the data to save, enter "ON" if this applet turns on the appliance, or "OFF" if this applet turns off the appliance.



Review the applet to confirm and click the "Finish" button to complete. You may want to disable notifications when the applet runs to avoid getting notified every time you turn the appliance on or off when using Alexa.



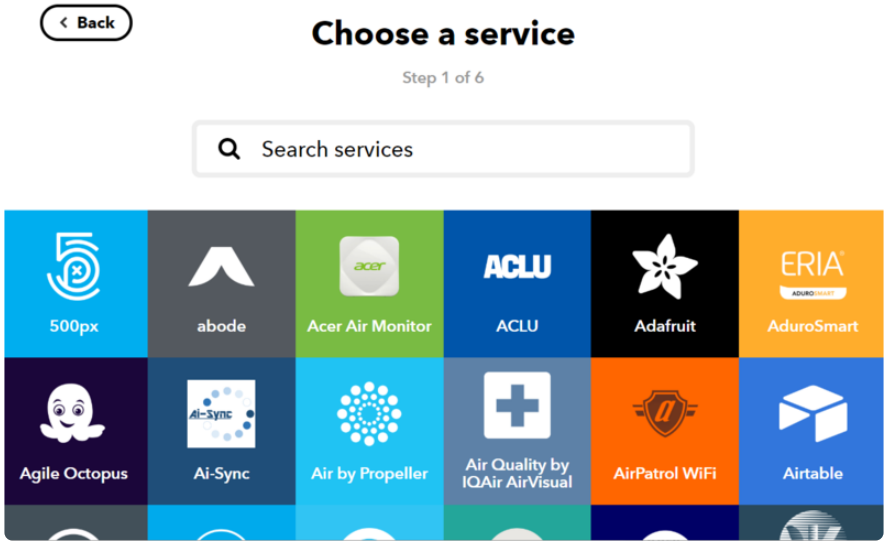
You can now test your applet by speaking the phrase from the previous step. Don't forget you need to create two applets in order to turn the appliance on and off.

Google Assistant Setup

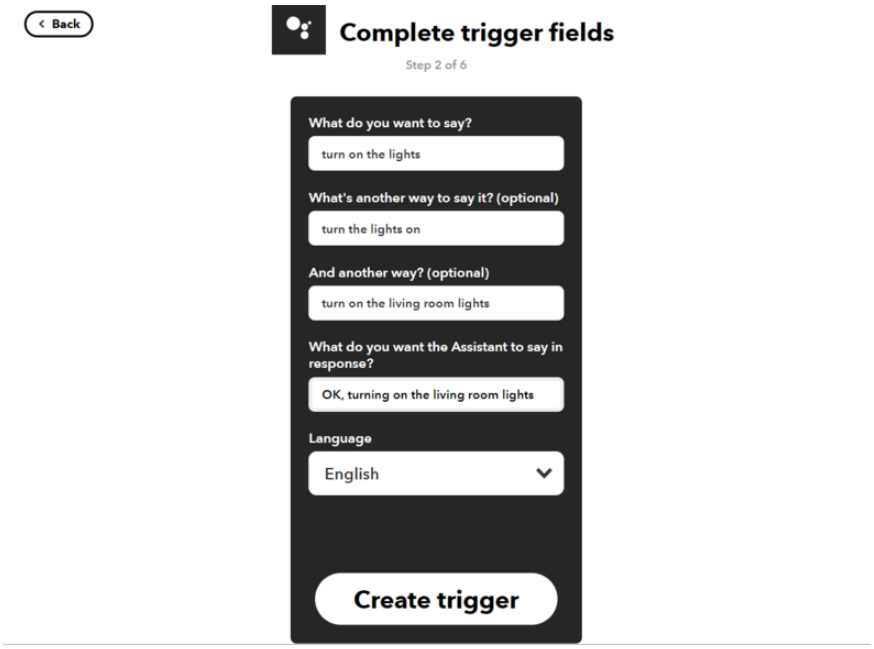
Here are the steps to configure Google Assistant with IFTTT. You will need to create two applets, one to turn on the lights and the other to turn off the lights.

Click "This" to create the first half of the applet.

Search for and select "Google Assistant". Don't be tempted to select Adafruit yet! We will be selecting that in the second half of the applet.

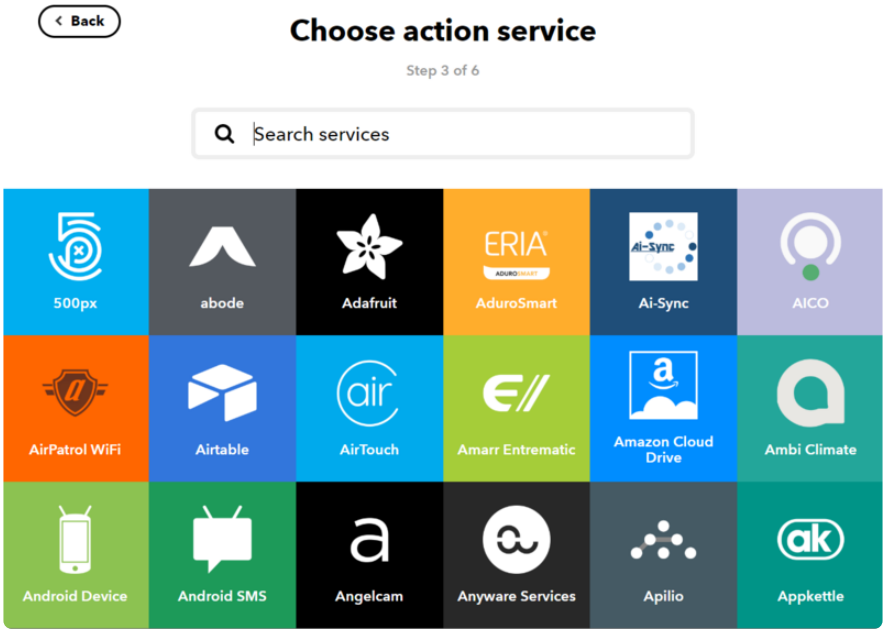


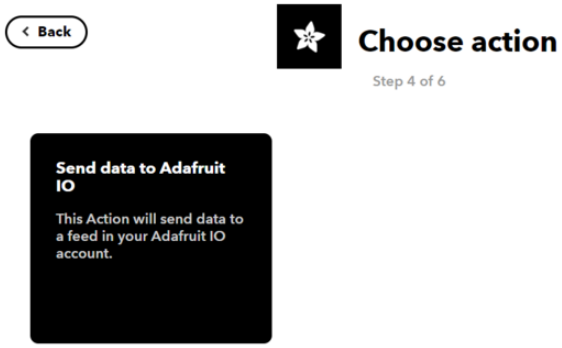
Click on the "Say a simple phrase" trigger. The Google Assistant trigger here is more customizable than the equivalent Amazon trigger. Here you can enter up to three phrases to run the action, and you can also tell Google Assistant what to say in response. In this example, three phrases are entered: "turn on the lights", "turn the lights on" and "turn on the living room lights". For a response, we used "OK, turning on the living room lights"



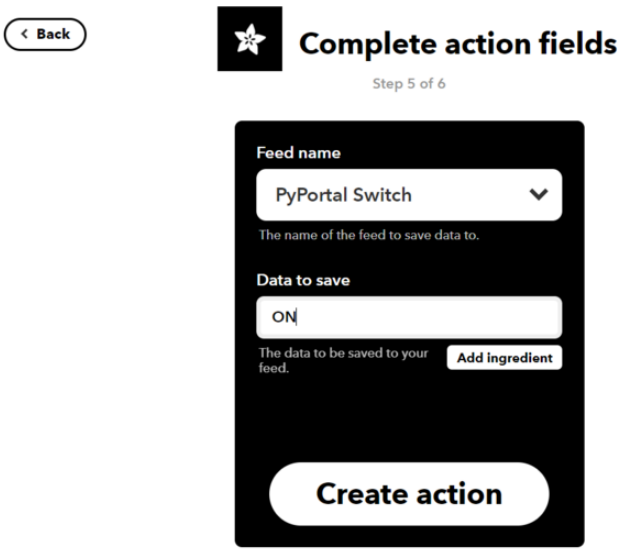
Click "That" to create the second half of the applet. Notice the Google Assistant icon in the "If" part of the applet.

Here is where we connect our applet to Adafruit IO. Select the "Adafruit" icon from the list of action services, and then select the action, "Send data to Adafruit IO".

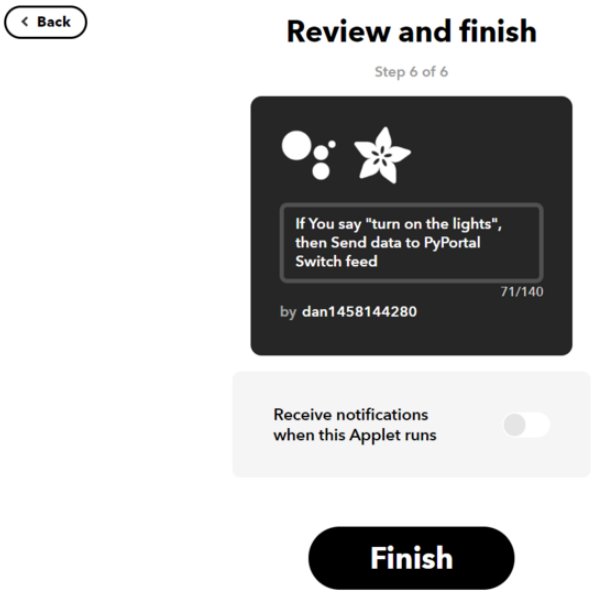




Pick the feed we created earlier on Adafruit IO, "PyPortal Switch". For the data to save, enter "ON" if this applet turns on the appliance, or "OFF" if this applet turns off the appliance.



Review the applet to confirm and click the "Finish" button to complete. You may want to disable notifications when the applet runs to avoid getting notified every time you turn the appliance on or off when using Google Assistant.



You can now test your applet by speaking the phrase from the previous step. Don't forget you need to create two applets in order to turn the appliance on and off.