ESP32-S3 Reverse TFT Digital Clock Display featuring Blanka-chan!
Created by Trevor Beaton


Last updated on 2023-08-29 04:47:02 PM EDT
# Table of Contents

## Overview
- Use PyLeap
- Parts

## WorldTimeAPI

## CircuitPython
- CircuitPython Quickstart

## Using PyLeap

## Setting up your Credentials

## Digital Clock Display Code
- Connecting to WiFi
- Creating a Socket Pool
- Setting the background image
- Setting up a label for displaying the current time
- Creating a function to set the background image
- Creating a function to parse the JSON time data
- The Main Loop
- Having Problems Getting Data?
Overview

Here, we'll build a digital world clock using the ESP32-S2 Reverse TFT or ESP32-S3 and fetch time data from the WorldTimeAPI(). But, to add some fun to our project, we'll also bring in some gaming nostalgia. We'll be integrating two different background images of the Street Fighter 6 character, Blanka-chan, to create an animated display of Blanka-chan using an electrical attack.

This is a perfect beginner project if you love IoT, video games or want to learn something new and fun!

Use PyLeap

You can quickly transfer this project to your ESP32-S3 TFT via a WiFi connection using the PyLeap app.

PyLeap() is a free iOS, iPad, and Android app. It can be downloaded from the Apple App Store() or Google Play Store().
ESP32-S2 Rev TFT Digital Clock Display featuring Blanka-chan!

Display the current time using the ESP32-S2 Reverse TFT

Compatible with:

✅ ESP32-S2

Learn Guide

Run
Parts

**Adafruit ESP32-S3 Reverse TFT Feather**
Like Missy Elliot, we like to "put our [Feather] down, flip it and reverse it" and that's exactly what...
https://www.adafruit.com/product/5691

**Adafruit ESP32-S2 Reverse TFT Feather**
Like Missy Elliot, we like to "put our [Feather] down, flip it and reverse it" and that's exactly what...
https://www.adafruit.com/product/5345

**Lithium Ion Battery - 3.7V 2000mAh**
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
USB Type A to Type C Cable - approx 1 meter / 3 ft long
As technology changes and adapts, so does Adafruit. This USB Type A to Type C cable will help you with the transition to USB C, even if you're still...
https://www.adafruit.com/product/4474

WorldTimeAPI

WorldTimeAPI is a simple web service which returns the current local time for a given timezone as either plain-text or JSON.

We'll use the WorldTimeAPI() to retrieve the current time through its API. There's no need to sign up or register for an account. We use the given curl to update it with our current location.

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 drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

Download the latest version of CircuitPython for this board via circuitpython.org
Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.

Plug your board into your computer, using a known-good data-sync cable, directly, or via an adapter if needed.

Double-click the reset button (highlighted in red above), and you will see the RGB status LED(s) turn green (highlighted in green above). If you see red, try another port, or if you’re using an adapter or hub, try without the hub, or different adapter or hub.

For this board, tap reset and wait for the LED to turn purple, and as soon as it turns purple, tap reset again. The second tap needs to happen while the LED is still purple.

If double-clicking doesn’t work the first time, try again. Sometimes it can take a few tries to get the rhythm right!

A lot of people end up using charge-only USB cables and it is very frustrating! Make sure you have a USB cable you know is good for data sync.
You will see a new disk drive appear called FTHRS2BOOT.

Drag the adafruit_circuitpython_etc.uf2 file to FTHRS2BOOT.

The BOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it!
Using PyLeap

You can quickly transfer this project to your device using the PyLeap app.

PyLeap is a free app available for iOS, iPad, and Android devices. It can be downloaded from the Apple App Store or Google Play Store. It allows users to easily download code files and assets and transfer them to their Adafruit devices using Bluetooth Low Energy (BLE) or WiFi.

To upload a project to your PyLeap-enabled device, select the project in the project list.
# PyLeap Projects

Browse available WiFi

<table>
<thead>
<tr>
<th>Project</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boxing Glove Tracker</td>
<td></td>
</tr>
<tr>
<td>ESP32-S2 NeoPixel LED</td>
<td></td>
</tr>
<tr>
<td>I2C: On-Board Battery Monitor</td>
<td></td>
</tr>
</tbody>
</table>
Then, once the project cell has collapsed, press the "Run it" button to download and transfer the project over to your PyLeap-enabled device.

Setting up your Credentials

The digital clock needs your WiFi and WorldTimeAPI token to fetch JSON current time data.

Plug your Feather board into your computer via a known good data + power USB cable. Your board should appear as a thumb drive in your File Explorer / Finder (depending on your operating system) named CIRCUITPY.

Create a file with the name settings.toml in the root directory of the CIRCUITPY drive.

Add the following below:

The file should contain the keys `CIRCUITPY_WIFI_SSID`, `CIRCUITPY_WIFI_PASSWORD`, `CIRCUITPY_WEB_API_PASSWORD`, `CIRCUITPY_WEB_API_PORT` and `TIMEZONE`.

Change `CIRCUITPY_WIFI_SSID` and `CIRCUITPY_WIFI_PASSWORD` to match your network credentials.
Valid time zones can be found in the WorldTimeAPI documentation here. Pick the closest place that has your same time (for example America/New_York also works for Boston, Washington DC, and Miami).

The web server is on port 80 unless overridden by CIRCUITPY_WEB_API_PORT. It also enables MDNS.

Once these are defined, CircuitPython will automatically connect to the network and start the webserver used for the workflow.

Here is an example settings.toml:

```toml
# To auto-connect to WiFi
CIRCUITPY_WIFI_SSID="YOUR-WIFI-NETWORK-NAME"
CIRCUITPY_WIFI_PASSWORD="YOUR-WIFI-NETWORK-PASSWORD"

# Current location
TIMEZONE="YOUR-COUNTRY/YOUR-STATE"
# example: "America/New_York"

# To enable modifying files from the web. Change this too!
# Leave the User field blank in the browser.
CIRCUITPY_WEB_API_PASSWORD="passw0rd"

CIRCUITPY_WEB_API_PORT=80
```
Digital Clock Display Code

```python
import os
import ssl
import time
import wifi
import board
import displayio
import terminalio
import socketpool
import adafruit_requests

# Initialize Wi-Fi connection
try:
    wifi.radio.connect(os.getenv("CIRCUITPY_WIFI_SSID"), os.getenv("CIRCUITPY_WIFI_PASSWORD"))
    print("Connected to %s!" % os.getenv("CIRCUITPY_WIFI_SSID"))
except Exception as e:
    print("Failed to connect to WiFi. Error: ", e, "\nBoard will hard reset in 30 seconds." )

pool = socketpool.SocketPool(wifi.radio)
requests = adafruit_requests.Session(pool, ssl.create_default_context())

DATA_SOURCE = "http://worldtimeapi.org/api/timezone/" + os.getenv("TIMEZONE")
```

©Adafruit Industries
# Set up display a default image
display = board.DISPLAY
default_bitmap = displayio.OnDiskBitmap("/images/blanka-chan.bmp")
default_tile_grid = displayio.TileGrid(default_bitmap,
pixel_shader=default_bitmap.pixel_shader)

# Create label for displaying time


# Create main group to hold all display groups
main_group = displayio.Group()
main_group.append(group)
main_group.append(time_label)
# Show the main group on the display
display.show(main_group)

# Set up display a default image

def set_background_image(filename):
    global current_background_image  # pylint: disable=global-statement
tile_bitmap = displayio.OnDiskBitmap(filename)
new_tile_grid = displayio.TileGrid(tile_bitmap,
pixel_shader=tile_bitmap.pixel_shader)
group[0] = new_tile_grid
current_background_image = filename

def parse_time(datetime_str):
    # Extract the time part from the datetime string
    time_str = datetime_str.split("T")[1].split("."")[0]
    hour, minute, _ = map(int, time_str.split(":"))
    # Convert 24-hour format to 12-hour format and determine AM/PM
    period = "AM"
    if hour >= 12:
        period = "PM"
    if hour > 12:
        hour -= 12
    elif hour == 0:
        hour = 12
    return hour, minute, period

while True:
    # Fetch time data from WorldTimeAPI
    response = requests.get(DATA_SOURCE)
data = response.json()
    # Parse the time from the datetime string
    current_hour, current_minute, current_period = parse_time(data["datetime"])
    # Display the time
time_label.text = "{:2}{}{:<02}".format(current_hour, current_period, current_minute)
    # Switch between two images
    if current_background_image == "/images/blanka-chan.bmp":
        set_background_image("/images/blanka-chan-charged.bmp")
    else:
        set_background_image("/images/blanka-chan.bmp")

time.sleep(5)
This code is doing several things:

- Connects to the internet via WiFi using your credentials
- Creates a label for displaying the current time
- Sets a background image and toggles between another background image
- Fetches and displays the current time from WorldTimeAPI

Connecting to WiFi

This code block connects to a WiFi network using the `wifi.radio.connect` function and passes in the network’s SSID and password as arguments from your credentials. The values of the SSID and password are read from environment variables `CIRCUITPY_WIFI_SSID` and `CIRCUITPY_WIFI_PASSWORD`.

```python
try:
    wifi.radio.connect(
        os.getenv("CIRCUITPY_WIFI_SSID"), os.getenv("CIRCUITPY_WIFI_PASSWORD")
    )
    print("Connected to %s!" % os.getenv("CIRCUITPY_WIFI_SSID"))
except Exception as e:  # pylint: disable=broad-except
    print(  
        "Failed to connect to WiFi. Error:", e, "\nBoard will hard reset in 30 seconds."
    )
```

Return to the "Setting up your Credentials" page if you haven't set up your credentials.

Creating a Socket Pool

The `socketpool.SocketPool` function creates a pool of sockets for managing network connections. It takes the `wifi.radio` object as an argument to allow for network communication over a WiFi connection.

You will also need to create session object for making HTTP requests using `adafruit_requests.Session`.

```python
pool = socketpool.SocketPool(wifi.radio)
requests = adafruit_requests.Session(pool, ssl.create_default_context())
```

Next, set up the URL for fetching time data.

```
DATA_SOURCE = "http://worldtimeapi.org/api/timezone/" + os.getenv("TIMEZONE")
```
Setting the background image

This chunk of code sets up the TFT’s display to show a default image. First, access the TFT’s display, then loads an image from the disk into a bitmap. This bitmap is then placed into a TileGrid for display purposes.

A new group is created, and the TileGrid is added to this group. This group can now be shown on the display, allowing the image to be viewed.

```python
display = board.DISPLAY
default_bitmap = displayio.OnDiskBitmap("/images/blanka-chan.bmp")
default_tile_grid = displayio.TileGrid(default_bitmap, pixel_shader=default_bitmap.pixel_shader)
group = displayio.Group()
group.append(default_tile_grid)
```

Setting up a label for displaying the current time

This part of the code creates a label to display the time and positions it in the center of the TFT’s display. The label is created using a bitmap font, and its size is scaled by a factor of 5.

It then makes a main group to hold all display components, which includes the previously created image group and the time label. Lastly, it shows the main group on the TFT, displaying the image and the time label on the screen.

```python
time_label = bitmap_label.Label(terminalio.FONT, scale=5)
time_label.anchor_point = (0.2, 0.5)
time_label.anchored_position = (display.width // 2, display.height // 2)

main_group = displayio.Group()
main_group.append(group)
main_group.append(time_label)
display.show(main_group)
```

Creating a function to set the background image

The `set_background_image(filename)` function is used to change the background image of the display. Here’s how it works:

1. First, it takes in an input `filename` that will be used as the new background.
Then uses the `global` keyword to declare `current_background_image` as a global variable, which means this function can access and modify the value of `current_background_image` defined outside the function.

It then creates a bitmap from the new image file using `displayio.OnDiskBitmap(filename)`.

A new `TileGrid` is created from this bitmap. The `TileGrid` is a collection of graphical tiles that can be positioned in a grid and drawn on the display.

The group's first element (the existing tile grid that displays the current background image) is replaced by this new tile grid. This effectively changes the background image on the display.

Finally, the `current_background_image` is updated with the new filename. This helps keep track of which image is currently being displayed.

```python
def set_background_image(filename):
    global current_background_image  # pylint: disable=global-statement
    tile_bitmap = displayio.OnDiskBitmap(filename)
    new_tile_grid = displayio.TileGrid(tile_bitmap, pixel_shader=tile_bitmap.pixel_shader)
    group[0] = new_tile_grid
    current_background_image = filename
```

Creating a function to parse the JSON time data

The `parse_time(datetime_str)` function extracts and formats the time from the datetime string, typically in ISO 8601 format (YYYY-MM-DDTHH:MM:SSZ).

1. The function takes as input `datetime_str`, a string representing date and time.

2. It first extracts the time part from the datetime string. In the ISO 8601 format, the date and time are separated by "T," and the seconds are followed by a ".". So, the function splits the string at "T" and "." and picks the second element from each resulting list, which gives the time string in HH:MM:SS format.

3. This time string is further split at ":", and the hour, minute, and second are converted to integers using the `map` function.

4. It then converts the hour from a 24-hour format to a 12-hour format. If the hour is greater than or equal to 12, it's considered PM, and the hour is reduced by 12 for hours greater than 12. If the hour is 0, it's converted to 12.
The function returns the hour, minute, and period ("AM" or "PM"). The seconds are not used in this function, so they are discarded.

def parse_time(datetime_str):
    # Extract the time part from the datetime string
    time_str = datetime_str.split("T")[1].split(".")[0]
    hour, minute, _ = map(int, time_str.split(":"))

    # Convert 24-hour format to 12-hour format and determine AM/PM
    period = "AM"
    if hour >= 12:
        period = "PM"
        if hour > 12:
            hour -= 12
    elif hour == 0:
        hour = 12

    return hour, minute, period

The Main Loop

This part of the code runs in an infinite loop, continuously updating the time display and switching between two images on the TFT every five seconds. Here's a breakdown of this loop:

1. Fetch Time Data: First, make a GET request to the WorldTimeAPI using the specified URL assigned to the `DATA_SOURCE` variable earlier in the code. The API responds with the current date and time data in JSON format, then converted to a Python dictionary using the `json()` method.

2. Parse Time: The function `parse_time` is called with the datetime string received from the API. This function returns the hour, minute, and period (AM/PM) after processing the input string.

3. Display Time: The current time is formatted into a string and assigned to the `text` attribute of the `time_label` object, which updates the time display on the board.

4. Switch Images: The code then checks the current background image file path. If it's set to the path of blanka-chan.bmp, it calls the `set_background_image` function to change the background image to blanka-chan-charged.bmp. If the current image is not blanka-chan.bmp, it switches the image back to blanka-chan.bmp.

5. Sleep: The program pauses for 5 seconds using `time.sleep(5)`. After this pause, the loop starts over, getting the updated time from the API, parsing it, updating the display, and switching the image again.
while True:
    # Fetch time data from WorldTimeAPI
    response = requests.get(DATA_SOURCE)
    data = response.json()

    # Parse the time from the datetime string
    current_hour, current_minute, current_period = parse_time(data["datetime”])

    # Display the time
    time_label.text = " {:2}{\n  :{:02}".format(current_hour, current_period, current_minute)

    # Switch between two images
    if current_background_image == " /images/blanka-chan.bmp":
        set_background_image("/images/blanka-chan-charged.bmp")
    else:
        set_background_image("/images/blanka-chan.bmp")

time.sleep(5)

Having Problems Getting Data?

Check your settings if you have problems getting the data to display correctly. The set
tings.toml file has the information noted in the "Setting up your Credentials” page.