Simple Weather App

Sep 2026

Back to Micro Python

Introduction

Please be aware, that this project uses the same hardware, wiring and 3D Printed enclosure as this project Pico Wifi Analyzer with LCD so please have a look at this project for much more information, and to get the 3D enclosure if you want it.

This project is a bit of an accident. We want to make a new clock, we made one around 10 years ago from a Raspberry Pi Zero and some 7 segment LEDs. That project can be found here - Raspberry Pi LED Clock and NTP Server.

Now, it has worked flawlessly for 10 years. But it is still a small computer with an OS, and you have to do some maintenance on the OS (upgrades etc) periodically. So the thinking was, a new clock using a Raspberry Pi Pico would be cool, newer hardware, no OS worries, better display (LCD rather than 7 segment) and include other things than the time like the weather, or a news feed etc.

The first thing we explored was how to get the weather from the internet, turns out this is pretty simple. We used something called Open Meto, which is:

- Open-Meteo is a free, open-source weather API that delivers localized weather forecasts and historical climate data without requiring an API key.

So basically a free weather service to pull information from using an API.

If you wish to make one of these, then great. Please check out the Project called 'Network Analyzer' and look at Phase 2 (which is the LCD we are using).

You will need:

- The same LCD.
- The Same Driver.
- The Test Code for the Driver

Check out this link: Pico with 2.4" Tenstar SPI LCD. This will ensure your hardware is good, before you start to worry about the software.

A Simple Test

If you have completed the hardware setup, and see some test text on your LCD, then we are good to start. The first test is to get some weather data from Open-Meteo. Now, this means connecting to the internet from your Pico, so please ensure you have done the following:

- Connect Pico to Wifi
- Find Pico MAC Address

This will ensure you have tested your Pico Wifi and you have your MAC address in case your router requires this.

Test Open-Meteo

From Thonny, save this code to your Pico as Meteo-test and run it.

In the code, don't forget to change the following:

Set your Wifi Details.

| download

```
# --- CONFIGURATION ---
WIFI_SSID = "YOUR_WIFI_SSID"
WIFI_PASS = "YOUR_WIFI_PASSWORD"
```

Set your Latitude and Longitude (you can leave it at London for a simple test if you want).

| download

```
# Coordinates (e.g., London: lat=51.5074, lon=-0.1278)
LATITUDE = "51.5074"
LONGITUDE = "-0.1278"
```

Save this code as 'Meteo-test.py' and run it.

| download

```
import network
import urequests
import time

# --- CONFIGURATION ---
WIFI_SSID = "YOUR_WIFI_SSID"
WIFI_PASS = "YOUR_WIFI_PASSWORD"

# Coordinates (e.g., London: lat=51.5074, lon=-0.1278)
LATITUDE = "51.5074"
LONGITUDE = "-0.1278"

# Open-Meteo API URL for temperature, humidity, and weather code
URL =
f"http://api.open-meteo.com/v1/forecast?latitude={LATITUDE}&longitude={LONGITUDE}&current_weather=
true"

def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print(f"Connecting to Wi-Fi network: {WIFI_SSID}...")
```

```
        wlan.connect(WIFI_SSID, WIFI_PASS)

        timeout = 10
        while not wlan.isconnected() and timeout > 0:
            print("Waiting for connection...")
            time.sleep(1)
            timeout -= 1

    if wlan.isconnected():
        print("Connected successfully!")
        print("Network status/IP:", wlan.ifconfig())
        return True
    else:
        print("Failed to connect to Wi-Fi.")
        return False

def fetch_weather():
    print("Fetching weather data...")
    try:
        response = urequests.get(URL)
        if response.status_code == 200:
            data = response.json()
            current = data.get("current_weather", {})

            temp = current.get("temperature")
            windspeed = current.get("windspeed")
            weathercode = current.get("weathercode")

            print("\n--- WEATHER DATA RECEIVED ---")
            print(f"Temperature: {temp} °C")
            print(f"Wind Speed:  {windspeed} km/h")
            print(f"Weather Code: {weathercode}")
            print("------------------------------\n")
        else:
            print(f"Error: Server responded with status code {response.status_code}")

        response.close()
    except Exception as e:
        print("Error fetching data:", e)

# --- RUN TEST ---
if connect_wifi():
    fetch_weather()
```

At the bottom of your Thonny, you should see something like the following:

download

```
Connected successfully!
Network status/IP: ('10.194.1.168', '255.255.255.0', '10.194.1.1', '10.194.1.1')
Fetching weather data...

--- WEATHER DATA RECEIVED ---

Temperature: 20.8 °C
Wind Speed:  15.1 km/h
Weather Code: 2
------------------------------
```

If this has worked, then fantastic, you are ready to use the full weather app code. Some information.

- Network Status is your Internal IP Range
- Temp and Speed is that in London (as denoted by the co-ordinates
- Weather Code 2 is a value for your weather, see below

So the weather code is Meteos 'code' for “Partly Cloudy” here are the Meteo weather codes:

Code Range	Weather Condition	Suggested Icon
0	Clear sky	Sun
1, 2, 3	Mainly clear, partly cloudy, overcast	Sun/Cloud or Cloud
45, 48	Fog and depositing rime fog	Fog / Low mist
51–57	Drizzle (light to dense, freezing)	Light Rain / Drops
61–67	Rain (slight to heavy, freezing)	Rain / Cloud with rain
71–77	Snow fall & snow grains	Snowflake
80–82	Rain showers (slight, moderate, violent)	Heavy Rain / Cloud with shower
95, 96, 99	Thunderstorm (with/without hail)	Lightning / Storm

Now we can move on to the full code.

The Full Weather App Code

Don't forget in this new code to do the following:

- Ensure your WIFI details are correct.
- Ensure that your Latitude and Longitude details are correct.

Download the following code and save it as 'Weather.py' and run it from Thonny. If it is working fine, you can save it as main.py so it runs as soon as you turn on your Pico.

| download

```
import network
import time
import urequests
import ujson
import framebuf
from machine import RTC, SPI, Pin
from st7789nf import (
    ST7789, COLOR_BLACK, COLOR_WHITE, COLOR_CYAN, COLOR_AMBER,
    COLOR_YELLOW, COLOR_NAVY, COLOR_GRAY, COLOR_DARKGRAY
)

# ==========================================
# CONFIGURATION
# ==========================================
WIFI_SSID = "YOUR_WIFI_SSID"
WIFI_PASS = "YOUR_WIFI_PASSWORD"
```

```
# London Coordinates
LATITUDE = "51.5074"
LONGITUDE = "-0.1278"

# BST / UK Daylight Saving Time offset (1 for BST, 0 for GMT)
BST_OFFSET = 1

# Open-Meteo API Endpoint (configured for mph)
WEATHER_URL = (
    "https://api.open-meteo.com/v1/forecast?"
    f"latitude={LATITUDE}&longitude={LONGITUDE}"
"&current=temperature_2m,relative_humidity_2m,apparent_temperature,weather_code,wind_speed_10m"
    "&hourly=temperature_2m,weather_code"
    "&wind_speed_unit=mph"
    "&timezone=auto"
)

# ==========================================
# HARDWARE PINS (SPI1)
# ==========================================
LCD_WIDTH  = 320
LCD_HEIGHT = 240

PIN_SCK  = 10  # GP10
PIN_MOSI = 11  # GP11
PIN_RST  = 12  # GP12
PIN_DC   = 13  # GP13
PIN_CS   = 14  # GP14

# ==========================================
# BITMAP ICONS (16x16)
# ==========================================
ICON_SUN_16 = bytes([
    0x01, 0x80, 0x41, 0x82, 0x20, 0x04, 0x10, 0x08,
    0x03, 0xC0, 0x07, 0xE0, 0x0F, 0xF0, 0x0F, 0xF0,
    0x0F, 0xF0, 0x0F, 0xF0, 0x07, 0xE0, 0x03, 0xC0,
    0x10, 0x08, 0x20, 0x04, 0x41, 0x82, 0x01, 0x80
])

ICON_CLOUD_16 = bytes([
    0x00, 0x00, 0x00, 0x00, 0x01, 0xC0, 0x03, 0xE0,
    0x07, 0xF0, 0x0F, 0xF8, 0x1F, 0xFC, 0x3F, 0xFE,
    0x7F, 0xFF, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0x7F, 0xFE, 0x3F, 0xFC, 0x00, 0x00, 0x00, 0x00
])

ICON_RAIN_16 = bytes([
    0x01, 0xC0, 0x03, 0xE0, 0x07, 0xF0, 0x0F, 0xF8,
    0x1F, 0xFC, 0x3F, 0xFE, 0x7F, 0xFF, 0xFF, 0xFF,
    0x7F, 0xFE, 0x00, 0x00, 0x24, 0x48, 0x48, 0x90,
    0x12, 0x24, 0x24, 0x48, 0x48, 0x90, 0x00, 0x00
])

ICON_THUNDER_16 = bytes([
    0x01, 0xC0, 0x03, 0xE0, 0x07, 0xF0, 0x0F, 0xF8,
    0x1F, 0xFC, 0x3F, 0xFE, 0x7F, 0xFF, 0xFF, 0xFF,
    0x03, 0x80, 0x07, 0x00, 0x0E, 0x00, 0x1F, 0xC0,
    0x03, 0x80, 0x07, 0x00, 0x06, 0x00, 0x04, 0x00
])

# ==========================================
# NETWORK & API FUNCTIONS
# ==========================================
def connect_wifi(ssid, password, timeout=15):
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print(f"Connecting to Wi-Fi: {ssid}...")
        wlan.connect(ssid, password)
        start_time = time.time()
```

```
        while not wlan.isconnected():
            if time.time() - start_time > timeout:
                print("Wi-Fi connection failed.")
                return False
            time.sleep(0.5)
    print(f"Wi-Fi Connected! IP: {wlan.ifconfig()[0]}")
    return True

def sync_ntp():
    import ntptime
    try:
        ntptime.settime()
        print("NTP Sync Success!")
        return True
    except Exception as e:
        print(f"NTP Sync Failed: {e}")
        return False

def fetch_weather():
    print("Fetching updated weather data...")
    try:
        response = urequests.get(WEATHER_URL)
        if response.status_code == 200:
            data = ujson.loads(response.text)
            response.close()

            current = data.get("current", {})
            current_weather = {
                "temp": current.get("temperature_2m"),
                "feels_like": current.get("apparent_temperature"),
                "humidity": current.get("relative_humidity_2m"),
                "wind_speed": current.get("wind_speed_10m"),
                "code": current.get("weather_code")
            }

            hourly = data.get("hourly", {})
            hourly_times = hourly.get("time", [])
            hourly_temps = hourly.get("temperature_2m", [])
            hourly_codes = hourly.get("weather_code", [])

            rtc = RTC()
            now_hour = rtc.datetime()[4]

            forecast = []
            for i in range(len(hourly_times)):
                slot_hour = int(hourly_times[i].split("T")[1].split(":")[0])
                if slot_hour >= now_hour and len(forecast) < 5:
                    forecast.append({
                        "hour": f"{slot_hour:02d}:00",
                        "temp": hourly_temps[i],
                        "code": hourly_codes[i]
                    })
            return {"current": current_weather, "forecast": forecast}
        else:
            response.close()
            return None
    except Exception as e:
        print(f"Weather fetch failed: {e}")
        return None

# ==========================================
# GRAPHICS HELPER FUNCTIONS
# ==========================================
def draw_text_scaled(display, text, x, y, color, scale=2):
    """Renders scaled text using character framebuffers."""
    char_buf = bytearray(8 * 8 * 2)
    char_fb = framebuf.FrameBuffer(char_buf, 8, 8, framebuf.RGB565)

    for idx, char in enumerate(text):
        char_fb.fill(COLOR_BLACK)
```

```
        char_fb.text(char, 0, 0, color)

        char_x = x + int(idx * 8 * scale)
        for row in range(8):
            for col in range(8):
                pixel = char_fb.pixel(col, row)
                if pixel != COLOR_BLACK:
                    display.fill_rect(char_x + int(col * scale), y + int(row * scale), int(scale),
int(scale), color)

def draw_bitmap_scaled(display, bitmap_bytes, x, y, width, height, fg_color, scale=1):
    """Draws 1-bit monochrome bitmaps to screen."""
    for row in range(height):
        for col in range(width):
            byte_idx = (row * width + col) // 8
            bit_idx = 7 - (col % 8)
            pixel = (bitmap_bytes[byte_idx] >> bit_idx) & 1
            if pixel:
                if scale == 1:
                    display.pixel(x + col, y + row, fg_color)
                else:
                    display.fill_rect(x + col * scale, y + row * scale, scale, scale, fg_color)

def get_wmo_icon_and_color(wmo_code):
    """Maps Open-Meteo WMO weather codes to icons and colors."""
    if wmo_code in (0, 1):
        return ICON_SUN_16, COLOR_YELLOW
    elif wmo_code in (2, 3, 45, 48):
        return ICON_CLOUD_16, COLOR_WHITE
    elif 51 <= wmo_code <= 67 or 80 <= wmo_code <= 82:
        return ICON_RAIN_16, COLOR_CYAN
    elif wmo_code >= 95:
        return ICON_THUNDER_16, COLOR_AMBER
    else:
        return ICON_CLOUD_16, COLOR_GRAY

def render_ui(display, weather_data):
    """Draws dark theme UI layout."""
    display.fill(COLOR_BLACK)

    # --- TOP BAR ---
    display.fill_rect(0, 0, 320, 32, COLOR_NAVY)
    display.hline(0, 32, 320, COLOR_CYAN)

    rtc = RTC()
    dt = rtc.datetime()
    hour = (dt[4] + BST_OFFSET) % 24
    time_str = f"{hour:02d}:{dt[5]:02d}:{dt[6]:02d}"

    # Header Title (Larger Purbrook text) & Live Clock
    draw_text_scaled(display, "Purbrook", 10, 8, COLOR_WHITE, scale=1.9)
    draw_text_scaled(display, time_str, 185, 8, COLOR_AMBER, scale=2)

    # --- CENTER PANEL ---
    if weather_data and "current" in weather_data:
        curr = weather_data["current"]

        # Main Weather Icon (Scale=3 -> 48x48)
        icon_data, icon_color = get_wmo_icon_and_color(curr["code"])
        draw_bitmap_scaled(display, icon_data, x=15, y=42, width=16, height=16,
fg_color=icon_color, scale=3)

        # Temperature & Feels Like
        draw_text_scaled(display, f"{curr['temp']:.1f}C", 75, 40, COLOR_WHITE, scale=3)
        draw_text_scaled(display, f"Feels: {curr['feels_like']:.1f}C", 75, 70, COLOR_GRAY,
scale=2)

        # Environmental Details (In mph)
        display.hline(15, 98, 290, COLOR_DARKGRAY)
        draw_text_scaled(display, f"Humidity: {curr['humidity']}%", 15, 106, COLOR_CYAN, scale=2)
```

```
        draw_text_scaled(display, f"Wind: {curr['wind_speed']} mph", 15, 128, COLOR_AMBER,
scale=2)

    # --- BOTTOM STRIP (5-Hour Forecast) ---
    display.hline(0, 152, 320, COLOR_CYAN)

    if weather_data and "forecast" in weather_data:
        slot_width = 320 // 5
        for idx, slot in enumerate(weather_data["forecast"][:5]):
            x = idx * slot_width

            if idx > 0:
                display.vline(x, 152, 88, COLOR_DARKGRAY)

            # Forecast Hour Header
            draw_text_scaled(display, slot["hour"][:2], x + 18, 158, COLOR_AMBER, scale=1.5)

            # Forecast Icon
            f_icon, f_color = get_wmo_icon_and_color(slot["code"])
            draw_bitmap_scaled(display, f_icon, x=x + 16, y=175, width=16, height=16,
fg_color=f_color, scale=2)

            # Forecast Temperature
            draw_text_scaled(display, f"{int(slot['temp'])}C", x + 12, 212, COLOR_WHITE, scale=2)

    # Push to display
    display.show()

# ==========================================
# MAIN ROUTINE
# ==========================================
spi1 = SPI(1, baudrate=20_000_000, sck=Pin(PIN_SCK), mosi=Pin(PIN_MOSI))

tft = ST7789(
    spi=spi1,
    width=LCD_WIDTH,
    height=LCD_HEIGHT,
    dc=Pin(PIN_DC),
    reset=Pin(PIN_RST),
    cs=Pin(PIN_CS)
)

if connect_wifi(WIFI_SSID, WIFI_PASS):
    sync_ntp()

current_weather_data = fetch_weather()

last_weather_fetch = time.time()
WEATHER_REFRESH_INTERVAL = 900  # 15 Minutes

print("Starting main dashboard execution loop...")

while True:
    now = time.time()

    if now - last_weather_fetch >= WEATHER_REFRESH_INTERVAL:
        updated_data = fetch_weather()
        if updated_data:
            current_weather_data = updated_data
        last_weather_fetch = now

    render_ui(tft, current_weather_data)
    time.sleep(1)
```

From:
https://wiki.alanwalker.uk/ - **WalkerWiki - wiki.alanwalker.uk**

Permanent link:
https://wiki.alanwalker.uk/doku.php?id=simple_weather_app&rev=1789052555

Last update: **2026/09/10 15:02**