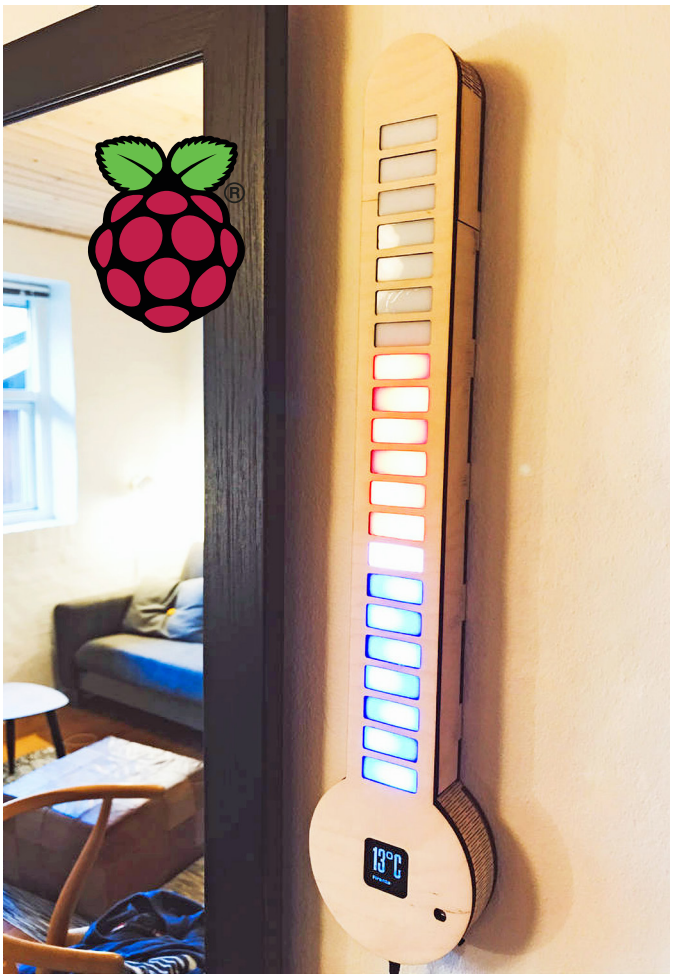
Cyfrowy termometr z podświetleniem LED

Termometr cyfrowy jest jednym z prostszych projektów, jakie można zrealizować. Co jednak, jeśli chcemy poznać temperaturę w dalekim od nas mieście? Opisane w poniższym artykule urządzenie wykorzystuje API serwisu pogodowego do pobierania danych na temat pogody z Internetu. Informacje są następnie wyświetlane z pomocą kolorowych diod LED.



Klasyczny termometr – cyfrowy bądź analogowy – składa się z sensora temperatury i wyświetlacza. Sensorem może być element taki jak termistor czy dioda półprzewodnikowa lub też scalony sensor temperatury (często wyposażony w cyfrowy interfejs do komunikacji – na rynku dostępne są układy do pomiaru temperatury z interfejsami I²C czy 1-Wire). Wyświetlacz powinien podawać informację w sposób czytelny dla użytkownika. Może to być wyświetlacz siedmiosegmentowy LED czy ekran LCD.

Opisywane urządzenie prezentuj trochę inne podejście. Układ zamiast sensora temperatury korzysta z Internetu, aby połączyć się z portalem udostępniającym dane pogodowe dla wielu miejsc na świecie. Pobiera informacje o pogodzie, a następnie wyświetla na kolorowych diodach LED, które emulują słupek rtęci. Dodatkowo, na niewielkim ekranie OLED wyświetlana jest nazwa miasta i wartość temperatury. Układ może obsługiwać całą listę miast, przełączając się pomiędzy nimi po kolei. Można także ręcznie wybrać miasto do wyświetlenia za pomocą przycisków z boku. Diody LED i ekran OLED są wyłączane, gdy układ jest nieaktywny przez kilka minut i włączy się ponownie, jeśli wykryje znaczną zmianę w poziomie oświetlenia w otoczeniu.

Potrzebne elementy

Do budowy opisywanego urządzenia potrzebne będą:

- Moduł Raspberry Pi Zero W z kartą SD wraz z zainstalowanym systemem operacyjnym Raspbian,
- 1,5-calowy moduł OLED sterowany poprzez interfejs I²C,
- Zasilacz 5 V, 2,4 A dla Raspberry Pi,
- Taśma LED RGB z diodami WS2812B (30 diod LED/m) o długości 5 metrów,
- Sklejka o grubości 4 i 6 mm, klej do drewna i pleksi (do wykonania obudowy),
- Opcjonalnie: płytka drukowana, przyciskami, gniazdem fotorezystora itp. jednakże układ zmontować można również np. na płytce uniwersalnej bądź podłączając poszczególne elementy kablami.

Na **fotografii 1** pokazano częściowo zmontowaną konstrukcję z widocznymi elementami obudowy.

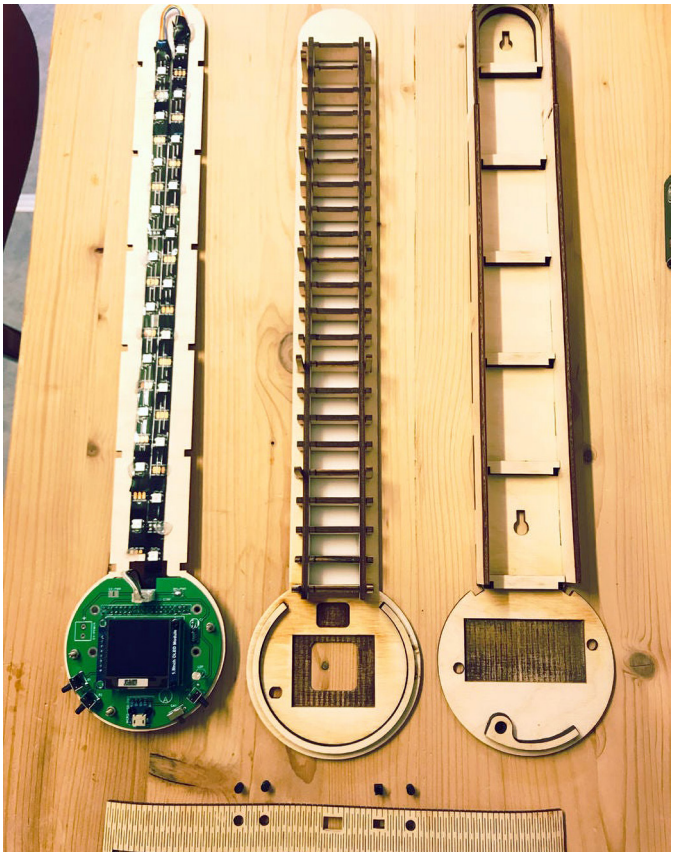
Połączenie wszystkich elementów i sterowanie diodami LED

Wszystkie elementy sterowane są poprzez poszczególne porty komputera jednopłytkowego Raspberry Pi Zero. Do minikomputera podpięto trzy przyciski, czujnik natężenia oświetlenia (fotorezystor), adresowalne diody RGB oraz ekran OLED. Dodatkowo, do komputera podłączyć musimy zasilanie 5 V poprzez port microUSB na płytce drukowanej Raspberry Pi. Zasilanie pozostałych elementów pobierane jest z odpowiednich pinów samego komputera jednopłytkowego.

Wyświetlacz OLED dołączony jest do interfejsu I²C Raspberry Pi. Interfejs ten posiada dwie linie – danych (SCL) i zegarową (SDA). Dołączone są one do, odpowiednio, pinów GPIO3 oraz GPIO2, gdzie wystawiony jest interfejs I2C1 tego komputera jednopłytkowego. Dodatkowo oczywiście, do modułu z ekranem OLED dołączyć należy zasilanie i masę. 1,5-calowy moduł OLED, jaki wykorzystał autor konstrukcji, zasilany może być napięciem 5 V lub 3,3 V, które pobrać można z Raspberry Pi z 2 i 4 pinu lub z 1 pinu złącza GPIO.

Diody LEF RGB wykorzystane w tym projekcie to moduły WS2812B. Każda dioda posiada wejście i wyjście cyfrowe, zachowujące się jak rejestr przesuwny dzięki temu dane z wejścia są przenoszone na wyjście cyfrowe diody, przesunięte o 24 bity potrzebne do zaprogramowania koloru na danym elemencie. W taśmie LED-owej diody połączone są szeregowo, dzięki czemu komputer musi komunikować się jedynie z jedną diodą, co wymaga zaangażowania tylko jednego pinu Raspberry Pi. Tym pinem jest GPIO18 (pin 12 na 40-pinowym złączu Raspberry Pi).

Do modułu podłączone są trzy przyciski – następne (next), poprzednie (previous) i wyłącz (shutdown). Są one podłączone z jednej strony do pinów cyfrowych Raspberry Pi (odpowiedmoop GPIO23, GPIO24 oraz GPIO25), a z drugiej strony do masy. Piny wejściowe są wewnętrznie podciągane do zasilania, więc zwarcie ich z masą



Fotografia 1. Częściowo zmontowana konstrukcja z widocznymi elementami obudowy

powoduje pojawienie się na wejściu stanu niskiego, który może być wykrywany przez skrypt.

Fotorezystor podłączony jest do wejścia cyfrowego (pin GPIO4 – siódmy pin złącza GPIO). Z uwagi na fakt, że Raspberry Pi nie posiada przetwornika analogowo-cyfrowego, konieczne było zastosowanie innej metody pomiaru wartości rezystancji. W tym celuć zestawiono prosty układ RC na wejściu GPIO4. Kondensator podpięty jest bezpośrednio pomiędzy wejście a masę, a fotoopornik podłączony jest pomiędzy wejście cyfrowe a linię zasilania. W kodzie skryptu zaszyto prosty algorytm pomiaru rezystancji z pomocą tego układu. Najpierw linia GPIO konfigurowana jest jako wyjście i ustawiana w stan niski. Będąc na potencjale masy rozładowuje ona dołączony kondensator. Następnie układ ustawia pin cyfrowy jako wejście, a fotorezystor zaczyna ładowanie kondensatora. Skrypt mierzy czas, potrzebny do naładowania kondensatora do poziomu stanu wysokiego, a czas ten jest proporcjonalny do prądu ładowania, który jest odwrotnie proporcjonalny do rezystancji fotoopornika. Wartość rezystancji bazowej fotoopornika i pojemności w układzie RC należy dobrać tak, aby w typowych warunkach pracy układu czas ładowania kondensatora nie przekraczał kilkuset cykli pracy skryptu.

Instalacja potrzebnych bibliotek

Zanim rozpoczniemy uruchamianie oprogramowania sterującego systemem na Raspberry Pi musimy zainstalować wszystkie potrzebne biblioteki, które pozwolą obsłużyć podłączane peryferia. W pierwszej kolejności, po podłączeniu wyświetlacza OLED, musimy zainstalować bibliotekę, pozwalającą na sterowanie tym ekranem. Upewnijmy się, że Raspberry Pi podłączane jest do Internetu, gdyż biblioteki pobierać będziemy z sieci. Wykonujmy, krok po kroku, wszystkie opisane niżej polecenia.

1. Włączamy interfejsy I²C oraz SPI w raspi-config. Po uruchomieniu tych interfejsów musimy zapisać ustawienia i uruchomić ponownie komputer.
2. Zaktualizujemy system:

```
sudo apt-get update
```

3. Zainstalujemy biblioteki, potrzebne do działania skryptu, kontrolującego układ:

```
sudo apt-get install python-dev
sudo apt-get install python-smbus
sudo apt-get install python-serial
sudo apt-get install python-imaging
```

4. Otwieramy plik konfiguracyjny z modułami jądra, jakie system ładować ma przy uruchomieniu z pomocą edytora tekstowego nano:

```
sudo nano /etc/modules
```

Dodajemy do pliku konfiguracyjnego dwa nowe moduły do obsługi I²C i restartujemy system:

```
i2c-bcm2708
i2c-dev
```

5. Teraz możemy pobrać (sklonować) repozytorium z GitHuba, gdzie znajdziemy pozostałe biblioteki do zainstalowania i skrypt potrzebny do uruchomienia systemu:

```
git clone https:/ /github.com/Anders644PI/1.5inch-
OLED-with-RPi .git
```

6. Znajdujemy utworzony folder i rozpakakowujemy plik *RPi_GPIO-0_6_5.zip*:

```
unzip RPi_GPIO-0_6_5.zip
```

Instalujemy bibliotekę z rozpakowanego pliku:

```
cd RPi_GPIO-0_6_5
sudo python setup.py install
```

7. Wracamy do głównego folderu:

```
cd /home/pi/1.5inch-OLED-with-RPi/
```

Powtarzamy to samo dla biblioteki spidev spakowanej w *spidev-3_2.zip*:

```
unzip spidev-3_2.zip
cd spidev-3_2.zip
sudo python setup.py install
```

8. Rozpakowujemy plik *wiringPi.zip*:

```
unzip wiringPi.zip
cd wiringPi
```

Wpisujemy następujące komendy, by zainstalować tą bibliotekę:

```
chmod 777 build
./build
```

Sprawdzamy czy instalacja się powiodła wpisując komendę do sprawdzenia wersji. Jeśli nie zwróci ona błędu to wiringPI zainstalowano poprawnie:

```
gpio –v
```

9. Ponownie wracamy do głównego folderu, rozpakowujemy plik *bcm2835-1_45.zip* i otwieramy powstały folder:

```
unzip bcm2835-1_45.zip
cd bcm2835-1_45
```

10. Następnie instalujemy znajdującą się tam bibliotekę:

```
./configure
make
sudo make check
sudo make install
```

11. Ponownie wracamy do nadrzędnego folderu */home/pi/1.5inch-OLED-with-RPi/* i uruchamiamy test wyświetlacza OLED z jednego z podfolderów:

```
cd /Demo_Code/Python/
sudo python main.py
```

Test powinien wyświetlić na ekranie linie, prostokąty, tekst i obrazek z kwiatem.

Firmware termometru

Skrypt sterujący termometrem korzysta z danych zdalnego serwisu pogodowego z danymi meteorologicznymi. Aby móc korzystać z jego API należy najpierw wykonać kilka prostych kroków.

API Pogodowe

Duża część kodu, jaka została użyta w poniższym skrypcie, została napisana przez StuffWithKirby. Dotyczy głownie kodu

Listing 1. Fragment pliku z kodami identyfikacyjnymi miast

```
[
    {
        „id”: 798544,
        „name”: „Republic of
Poland”,
        „country”: „PL”,
        „coord”: {
        „lon”: 20,
        „lat”: 52
        }
    },
    {
        „id”: 3094325,
        „name”: „Kuchary”,
        „country”: „PL”,
        „coord”: {
        „lon”: 19.383329,
        „lat”: 52.150002
        }
    },
    {
        „id”: 3082707,
        „name”: „Walbrzych”,
        „country”: „PL”,
        „coord”: {
        „lon”: 16.284321,
        „lat”: 50.771412
        }
    },
    {
        „id”: 3091150,
        „name”: „Naklo nad
Notecia”,
        „country”: „PL”,
        „coord”: {
        „lon”: 17.60181,
        „lat”: 53.142139
        }
    },
    {
        „id”: 756135,
        „name”: „Warsaw”,
        „country”: „PL”,
        „coord”: {
        „lon”: 21.01178,
        „lat”: 52.229771
        }
    },
    {
        „id”: 3094802,
        „name”: „Krakow”,
        „country”: „PL”,
        „coord”: {
        „lon”: 19.91667,
        „lat”: 50.083328
        }
    },
    {
        „id”: 3093524,
        „name”: „Leszno”,
        „country”: „PL”,
        „coord”: {
        „lon”: 16.57494,
        „lat”: 51.84034
        }
    },
    {
        „id”: 3083271,
        „name”: „Torun”,
        „country”: „PL”,
        „coord”: {
        „lon”: 18.598141,
        „lat”: 53.013748
        }
    },
    {
        „id”: 3083267,
        „name”: „Toruń Północny”,
        „country”: „PL”,
        „coord”: {
        „lon”: 18.6,
        „lat”: 53.033329
        }
    }
]
```

odpowiedzialnego za odczyt danych w formacie JSON z API serwisu pogodowego OpenWeatherMap.org.

Aby korzystać z danych gromadzonych przez OpenWeatherMap.org konieczne jest stworzenie darmowego konta na tym portalu. Po jego założeniu uzyskamy tzw. klucz API, który pozwoli nam zalogować się do API systemu (należy go zapisać). Oprócz konta musimy także znać numer identyfikacyjny miasta, jakie nas interesuje. W tym celu musimy pobrać ze strony projektu (link na końcu artykułu) plik *city.list.json.gz*. Należy go rozpakować, aby wyszukać interesujące nas dane. Na liście znajduje się wiele miast Polski. Przykładowe wpisy pokazano na **listingu 1**. Wybieramy spośród wszystkich dostępnych miast te, które nas interesują i zapisujemy ich identyfikatory (ciąg cyfr zawarty w polu id). Będziemy musieli je wpisać w naszym skrypcie sterującym, aby pobrać dane pogodowe dla tych lokalizacji.

Skrypt sterujący

Skrypt sterujący pobraliśmy z repozytorium na GitHubie już wcześniej, wraz z bibliotekami. Aby go uruchomić, w pierwszej kolejności musimy otworzyć folder LED_Thermometer_Code_and_OLED_driver:

```
cd LED_Thermometer_Code_and_OLED_driver
```

Następnie otwieramy edytorem tekstowym nano skrypt *Official_Digital_LED_Thermometer_v1-0.py*, aby uzupełnić klucz API oraz ID miast, jakie chcemy by były wyświetlane:

```
nano Official_Digital_LED_Thermometer_v1-0.py
```

Fragment skryptu zaprezentowano na **listingu 2**, miejsca wpisania potrzebnych danych opisano w komentarzach. Po zmodyfikowaniu i zapisaniu skryptu możemy wyjść z edytora i uruchomić skrypt wpisując:

```
sudo python Official_Digital_LED_Thermometer_v1-0.py
```

Listing 2. Skrypt do zmodyfikowania

```
import DEV_Config, OLED_Driver # pliki konfiguracyjne systemu
import Image, ImageDraw, ImageFont, ImageColor

import json, requests, os, time
import RPi.GPIO as GPIO
from neopixel import *

# Czcionki:
temp_fontfamily = 'steelfish_bd.ttf'
city_fontfamily = 'Cocogoose-Pro-Semilight-trial.ttf'
weather_fontfamily = 'BebasNeue-Regular.ttf'

# Przyciski
button_next = 23 # Pin 23
button_prev = 24 # Pin 24
button_shutdown = 25 # Pin 25
GPIO.setmode(GPIO.BCM)
GPIO.setup(button_next,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(button_prev,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(button_shutdown,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setwarnings(False)

# Fotorezystor
ldr_pin = 4             # Pin GPIO do którego podpięto fotorezystor

# Konfiguacja paska diodowego LED RGB
LED_COUNT      = 21     # Liczba diod LED RGB na pasku
LED_PIN        = 18     # Pin GPIO podłączony do wejścia danych diod LED (pin 18 wykorzystuje PWM).
LED_FREQ_HZ    = 800000 # Częstotliwość sygnału sterującego LEDami (typowo 800khz)
LED_DMA        = 10     # Kanał DMA wykorzystany do generowania sygnału (wstępnie 10)
LED_BRIGHTNESS = 155    # Jasność - 0 najniższa, 255 najwyższa
LED_INVERT     = False  # True aby odwrócić sygnał, gdy korzystamy z przesuwacza poziomu na tranzystorze NPN
LED_CHANNEL    = 0      # Ustaw na '1' dla pinów GPIO 13, 19, 41, 45 or 53

strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT, LED_BRIGHTNESS, LED_CHANNEL)
strip.begin()

# Konfiguracja wyświetlania na diodach LED
spacing = 2             # Krok temperatury pomiędzy LEDami
cold_color_spacing = 2  # Ilość diod zapalnych na niebiesko
hot_color_spacing = 4   # Ilość diod zapalnych na pomarańczowo
min_temp = -14          # Minimalna wyświetlana temperatura (musi być parzysta!)
max_temp = min_temp + LED_COUNT * spacing   # Wyliczona, maksymalna temperatura
led_before_show = False

##### KONFIGURACJA API POGODOWEGO
key = 'API_KEY'         # Zmień tą wartość na swój klucz API
units = 'metric'        # alternatywnie units = 'imperial' dla jednostek imperialnych, a nie metrycznych
city_ids = ['6455259', '2643743', '5128638', '6542285', '292223'] # identyfikatory miast
city_names = ['Paris', 'London', 'New York', 'Firenze', 'Dubai']  # nazwy miast
city_temp = []

# Pozostałe dane
update_displays = True
count_max = 30
count = count_max
weather_check = 180     # Czas pomiędzy sprawdzaniem temperatury
check = weather_check
c = 0
t = 0
ldr_list = [0,50,90,160,600]
LED_BRIGHTNESS_list = [150,130,120,90,60]

# Wykrywanie ruchu
ldr = 0
motion = False
```

Podsumowanie

Opisany system korzysta z API serwisu pogodowego OpenWeatherMap.org do zdobywania informacji o temperaturze w dowolnej lokalizacji. Temperatura i nazwa danego miasta wyświetlana jest na ekranie OLED, zamontowanym w urządzeniu oraz na dołączonych do układu diodach LED RGB. Diody RGB ułożone są pionowo, dzięki czemu ten wyświetlacz termometru przypomina klasyczny termometr rtęciowy. Z pewnością urządzenie będzie praktycznym i estetycznym dodatkiem do wielu wnętrz.

Nikodem Czechowski, EP

Źródło: ***http://bit.ly/30G4fFX***