Podstawy Internetu Rzeczy
Instrukcja do zajęć laboratoryjnych

Laboratorium 10

Urządzenia peryferyjne i komunikacja z nimi
Część 2

Opracował opiekun przedmiotu dr inż. Krzysztof Chudzik

Wydział Informatyki i Telekomunikacji
Politechnika Wrocławska

Wrocław, 2024.09.28 14:57:29

Spis treści

Lista zadań

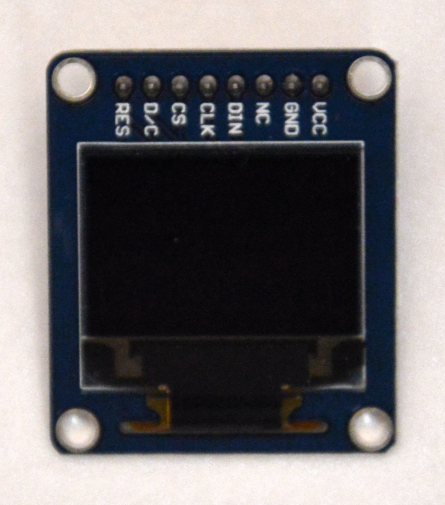
Zanim przystąpimy do konfigurowania i programowania, pamiętajmy, że z tych samych zestawów korzysta wielu Studentów. Kolejne instrukcje laboratoryjne przygotowane są z założeniem, że zestawy posiadają oryginalną konfigurację laboratoryjną. **Zabrania się wprowadzania trwałych zmian konfiguracyjnych systemu operacyjnego oraz jakiegokolwiek oprogramowania w sposób zmieniający jego działanie, w tym działanie interfejsów graficznych. Zmiany takie mogą uniemożliwić prawidłowe przeprowadzenie kolejnych zajęć.** Proszę umożliwić innym Studentom odbycie zajęć, tak jak zostały one zaplanowane. Kto nie zastosuje się do tego i będzie dezorganizował zajęcia, będzie traktowany jako uszkadzający zestawy i zostanie odsunięty od zajęć.

1 Wyświetlacz OLED

Podstawą budowy wyświetlaczy tego typu są organiczne diody elektroluminescencyjne OLED (ang. Organic Light-Emitting Diode). Diody te po pobudzeniu elektrycznym emitują światło. W przeciwieństwie do, na przykład, wyświetlaczy LCD i wielu o podobnej zasadzie działania, wyświetlacze OLED nie wymagają dodatkowego podświetlenia. Dzięki temu można konstruować bardzo cienkie wyświetlacze OLED, które montowane są w niewielkich urządzeniach, na przykład, zegarkach, bransoletach, w urządzeniach mobilnych, itp. W urządzeniach większych, cenione są za bardzo głęboką czerń i wykorzystywane są w telewizorach, monitorach, itp.

W zestawie laboratoryjnym zamontowany został niewielki wyświetlacz OLED o rozmiarze 0.95 cala (31.7mm x 37mm) i rozdzielczości 96 x 64 piksele. Wyświetlacz potrafi wyświetlić 65 tysięcy kolorów. Układem sterującym jest układ SSD1331. Wyświetlacz komunikuje się z Raspberry Pi za pomocą łącza szeregowego SPI. Dokładny opis modułu wyświetlacza znajduje się na stronie udostępnionej przez producenta: 0.95inch RGB OLED (B) (link). Na stronie tej można znaleźć instrukcję

użytkownika, schemat modułu, przykładowe kody (programy) oraz dokumenty ze specyfikacją. Wyświetlacz przedstawiony jest na fotografii 1.



Fotografia 1: Wyświetlacz OLED z zestawu laboratoryjnego.

Z zestawu kodów dostępnych na stronie producenta, wydobyte zostały przykładowe kody ze sterownikami dla Raspberry Pi i języka Python. Znaleźć można je w programie do testowania zestawu, w podkatalogu `lib/oled`. Jako przykład użycia można przedstawić skrypt `oled.py` zamieszczony jako kod 1, który korzysta ze skryptów w katalogu `lib/oled`. Proszę korzystać, właśnie z biblioteki zawartej w katalogu `lib/oled` programu testującego zestaw, ponieważ w kodzie samej biblioteki zostały wprowadzone zmiany dopasowujące ją do konfiguracji sprzętowej zestawu laboratoryjnego, ściślej, podłączenia wyświetlacza OLED do konkretnych pinów płytki Raspberry Pi. Oryginalna biblioteka, niezmodyfikowane, nie będzie działać poprawnie.

Kod 1: Plik `oled.py`.

```
#!/usr/bin/env python3

import time
from PIL import Image, ImageDraw, ImageFont
import lib.oled.SSD1331 as SSD1331


def oledtest():
    disp = SSD1331.SSD1331()

    # Initialize library.
    disp.Init()
    # Clear display.
    disp.clear()

    # Create blank image for drawing.
    image1 = Image.new("RGB", (disp.width, disp.height), "WHITE")
    draw = ImageDraw.Draw(image1)
    fontLarge = ImageFont.truetype('./lib/oled/Font.ttf', 20)
    fontSmall = ImageFont.truetype('./lib/oled/Font.ttf', 13)

    print("- draw line")
    draw.line([(0, 0), (0, 63)], fill="BLUE", width=5)
    draw.line([(0, 0), (95, 0)], fill="BLUE", width=5)
    draw.line([(0, 63), (95, 63)], fill="BLUE", width=5)
    draw.line([(95, 0), (95, 63)], fill="BLUE", width=5)

    print("- draw rectangle")
    draw.rectangle([(5, 5), (90, 30)], fill="BLUE")

    print("- draw text")
    draw.text((8, 0), u'Hello', font=fontLarge, fill="WHITE")
    draw.text((12, 40), 'World !!!', font=fontSmall, fill="BLUE")

    # image1 = image1.rotate(45)
    disp.ShowImage(image1, 0, 0)
    time.sleep(2)

    print("- draw image")
    image = Image.open('./lib/oled/pic.jpg')
    disp.ShowImage(image, 0, 0)
    time.sleep(2)
```

```
43
44      disp.clear()
45      disp.reset()
46
47  def test():
48      print('\nThe OLED screen test.')
49      oledtest()
50
51
52  if __name__ == "__main__":
53      test()
```

Rysowanie obrazów na wyświetlaczu odbywa się przez wyświetlanie przygotowanego wcześniej obrazu poprzez obiekt `disp` klasy `SSD1331` za pomocą metody `ShowImage()`. Nazwa klasy odpowiada układowi sterującemu w wyświetlaczu. Klasa i metoda są zdefiniowane w pliku `lib/oled/SSD1331.py`.

Taki sposób obsługi wyświetlacza jest bardzo nieefektywny. Wymiana całego obrazu na ekranie jest czasochłonna i nie robi dobrego wrażenia wizualnego. Z całą pewnością nie nadaje się do wyświetlania treści szybko zmieniających się. Lepiej byłoby pisać, czy rysować, od razu w pamięci wyświetlacza, ale producent nie udostępnił takiego rozwiązania dla języka Python.

Jednak, z drugiej strony, stosowana metoda jest dosyć uniwersalna, ponieważ jako obraz może być przyjęty obiekt skonstruowany za pomocą biblioteki Pillow (link na pypi.org), której wymagane w kodzie elementy importowane są komendą:

```
from PIL import Image, ImageDraw, ImageFont
```

Strona domowa biblioteki (link) zawiera obszerną dokumentację (link). Proszę zwrócić uwagę na wersję posiadanej biblioteki, ponieważ niekoniecznie będzie najnowsza. Wersję biblioteki zainstalowanej w systemie można sprawdzić komendą:

```
pip3 show Pillow
```

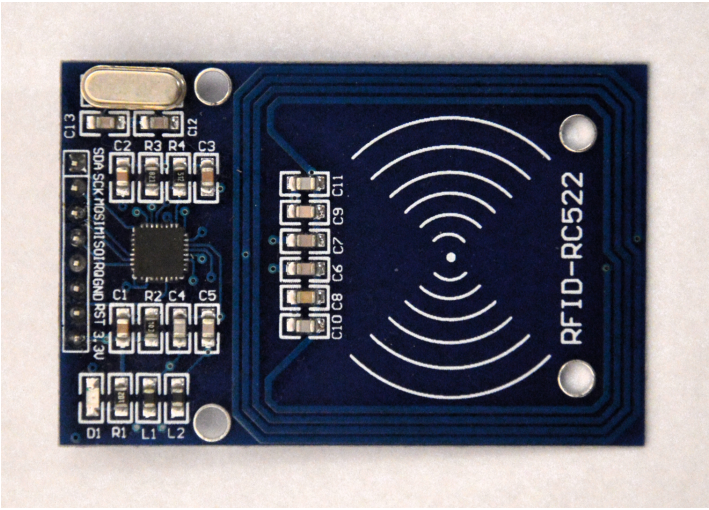
Przed przystąpieniem do programowania wyświetlacza, proszę zapoznać się z skryptem testującym zestaw `oled.py`, skryptami zawartymi w kodzie udostępnionym przez producenta modułu wyświetlacza i potem, w miarę potrzeb podczas programowania, z dokumentacją biblioteki `Pillow`.

Zadanie 1: Wyświetlanie na wyświetlaczu OLED wartości parametrów środowiskowych

Przygotuj program, który będzie odczytywał z czujnika BME280 wartości parametrów środowiskowych, które ten czujnik mierzy, i będzie wyświetlał je na ekranie OLED. Zilustruj wartości parametrów nie tylko jako wartości liczbowe, ale i za pomocą niewielkich symboli graficznych, na przykład, piktogramów.
Jest to przykładowe zadanie, które może być zmienione przez Prowadzącego zajęcia i połączone z innym.

2 Czytnik kart zbliżeniowych RFID

Zastosowany w zestawie laboratoryjnym moduł czytnika kart zbliżeniowych przedstawiony jest na fotografii 2.



Fotografia 2: Czytnik kart zbliżeniowych RFID z zestawu laboratoryjnego.

Zbudowany jest na bazie układu scalonego MFRC522. MFRC522 jest wysoko zintegrowanym czytnikiem dla bezkontaktowej komunikacji na częstotliwości 13.56 MHz. Czytnik MFRC522 wspiera standardy ISO/IEC 14443 A/MIFARE oraz NTAG. Więcej informacji dostepnych jest w karcie katalogowej układu MFRC522 (link).

W zestawie laboratoryjnym, moduł podłączony jest za pomocą łącza szeregowego SPI. Do obsługi programowej modułu czytnika kart zbliżeniowych wykorzystamy bibliotekę mfrc522 (link). W celu instalacji biblioteki, tak jak opisano to w instrukcji do laboratorium 7, wydajemy komendę:

```
sudo pip3 install mfrc522
```

Najprostszą ilustrację wykorzystania czytnika i jego obsługi programowej stanowi skrypt testujący moduł `rfid.py` z programu testującego cały zestaw. Zawartość pliku `rfid.py` jest przedstawiona jako kod 2. Program dokonuje trzykrotnego odczytu identyfikatora karty zbliżonej do czytnika i ze względu na swoja prostotę nie wymaga raczej dodatkowego komentarza.

Kod 2: Plik `rfid.py`.

```python
#!/usr/bin/env python3

# pylint: disable=no-member

import time
import RPi.GPIO as GPIO
from config import * # pylint: disable=unused-wildcard-import
from mfrc522 import MFRC522

def rfidRead():
    MIFAREReader = MFRC522()
    counter = 0
    while counter < 3:
        (status, TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)
        if status == MIFAREReader.MI_OK:
            (status, uid) = MIFAREReader.MFRC522_Anticoll()
            if status == MIFAREReader.MI_OK:
                num = 0
                for i in range(0, len(uid)):
                    num += uid[i] << (i*8)
                print(f"Card read UID: {uid} > {num}")
                time.sleep(0.5)
                counter += 1

def test():
    print('\nThe RFID reader test.')
    print('Place the card close to the reader (on the right side of the set).')
    rfidRead()
    print("The RFID reader tested successfully.")


if __name__ == "__main__":
    test()
    GPIO.cleanup() # pylint: disable=no-member
```

Zadanie 2: Rejestracja użycia kart RFID

Przygotuj program, który będzie reagował na przyłożenie karty do czytnika RFID, identyfikował tą kartę i rejestrował dokładny czas jej przyłożenia do czytnika. Zadbaj, aby karta przyłożona jednokrotnie, była zarejestrowana jeden raz, niezależnie jak długo pozostawała przyłożona do czytnika. O fakcie zarejestrowania przyłożenia karty poinformuj sygnałem dźwiękowym z buzzera i wizualnym, na przykład, diodami programowalnymi LED WS2812.
Jest to przykładowe zadanie, które może być zmienione przez Prowadzącego zajęcia i połączone z innym.