## Interfejsy SPI, I2C, USART Raspberry Pi

Angelika Tefelska Dariusz Tefelski

#### 2025-05-06

Celem ćwiczenia jest zapoznanie z możliwościami programistycznymi obsługi interfejsów SPI, I2C, Serial dostępnymi na Rasbyberry PI.

### Uwagi wstępne:

- W celu minimalizacji zagrożenia uszkodzenia Raspberry Pi, wszelkie podłączenia na płytce stykowej należy wykonywać po wyjęciu złącza kabla 40-pin z gniazda umieszczonego na płytce stykowej albo po wyłączeniu zasilania Raspberry Pi.
- Moduł do obsługi 1-wire może powodować problemy z kontrolą pinu GPIO4 (BCM4), ponieważ domyślnie ten pin jest wykorzystywany do transmisji 1-wire. Aby uniknąć problemów należy wstępnie wyładować moduł jądra poleceniem:

sudo rmmod w1\_gpio

Możliwe jest także przemapowanie pinu używanego przez wpisanie do pliku: /boot/config.txt:

dtoverlay=w1-gpio,gpiopin=x

gdzie x- oznacza numer pinu, który ma służyć jako 1-wire.

## Interfejs szeregowy - Serial

W Raspberry PI 3 pojawiły się interfejsy bezprzewodowe WIFI oraz Bluetooth. Bluetooth wymagał podłączenia poprzez interfejs szeregowy *Serial* i wykorzystano do niego w pełni działający interfejs szeregowy (/dev/ttyAMA0), który był dostępny programistycznie oraz na pinach GPIO14 (8), GPIO15 (10) we wcześniejszych wersjach Raspberry PI. Interfejs szeregowy dostępny na pinach został dodatkowo domyślnie wyłączony.

Możliwe jest włączenie interfejsu szeregowego poprzez raspi-config. Wtedy w pliku konfiguracyjnym /boot/config.txt zostaje dodana linijka enable\_uart=1. Włączony zostaje wtedy tzw. miniuart (/dev/ttyS0) i podłączony do pinów GPIO14 i GPIO15. Niestety jest on wrażliwy na skalowanie prędkości rdzenia core\_freq, dlatego jest także ustawiane na stałe core\_freq=250, które nieco zmniejsza wydajność Raspberry Pi 3. Miniuart ma także kilka innych ograniczeń np. co do rozmiaru znaku, ale generalnie działa prawidłowo w standardowych ustawieniach.

Możliwe są także 2 inne rozwiązania:

1. Wyłączenie interfejsu Bluetooth i przekazanie w pełni sprawnego portu szeregowego na piny GPIO14 i GPIO15. Potrzebna linijka w *config.txt*:

#### dtoverlay=pi3-disable-bt

2. Zamienienie miejscami portów szeregowych - w pełni sprawny na piny GPIO14 i GPIO15, natomiast miniuart do Bluetooth. Ogranicza to zastosowanie Bluetooth do niskich prędkości.

#### dtoverlay=pi3-miniuart-bt

Dodatkowo, aby uniknąć problemów przy zmianach portów i aby utrzymać kompatybilność ze wcześniejszymi Raspberry Pi, wprowadzono aliasy na urządzenia portów szeregowych, które zaleca się stosować pisząc programy je wykorzystujące: /dev/serial0 -> /dev/ttyS0 /dev/serial1 -> /dev/ttyAMA0

#### Zadanie - wykorzystanie modułu GPS poprzez interfejs szeregowy

W ćwiczeniu 1 - po instalacji systemu Raspbian, włączony został interfejs szeregowy - w pliku *config.txt* znajduje się linijka *enable\_uart=1*.

Ustawiona została także domyślnie konsola oraz informacje typu *debug* pochodzące z jądra na port szeregowy. Aby skorzystać z portu szeregowego w inny sposób należy te opcje wyłączyć w następujący sposób:

- 1. Uruchomić sudo raspi-config i w 5 Interfacing Options -> P6 Serial wybrać NO w odpowiedzi na pytanie Would you like to a login shell to be accessible over serial?, natomiast wybrać YES w odpowiedzi na pytanie Would you like the serial port hardware to be enabled?.
- 2. Zrestartować Raspberry Pi

sudo reboot

Ręczna zmiana trybu pracy portu szeregowego - nie potrzebna, jeśli wybrano, jak wyżej, poprzez narzędzie raspi-config:

1. Zachować kopię pliku /boot/cmdline.txt

sudo cp /boot/cmdline.txt /boot/cmdline.txt.bak

- 2. Należy w pliku /boot/cmdline.txt usunąć opcję console=serial0,115200
- 3. Wyłączyć serwis w system<br/>d odpowiedzialny za usługi konsolowe:

```
sudo systemctl stop serial-getty@ttyS0.service
sudo systemctl disable serial-getty@ttyS0.service
```

#### Testowanie odbioru sygnału gps.

1. Podłączyć moduł WaveShare NEO-6M/7M GPS w następujący sposób:

Wyprowadzenie GPS	Wyprowadzenie Raspberry PI $3~(\mathrm{nr}$ pinu złacza)
VCC	3V3 (pin 1)
GND	GND (pin 6)
TX	RxD - GPIO15 (pin 10)
$\mathbf{RX}$	TxD - GPIO14 $(pin 8)$
PPS	nie podłaczone

2. Ustawić prędkość portu szeregowego i przetestować odbiór:

```
sudo stty -F /dev/serial0 9600
cat /dev/serial0
```

3. Przetestować odbiór danych poprzez program do komunikacji poprzez port szeregowy picocom albo minicom.

```
sudo apt install picocom
picocom /dev/serial0 -b 9600
```

Dane z gps powinny być widoczne w terminalu.

Albo minicom:

```
sudo apt install minicom
minicom -s
```

- Ustawić w minicom port szeregowy jako urządzenie /dev/serial0 oraz prędkość i parametry transmisji na 9600 8N1, wyjść z ustawień poprzez klawisz Esc.
- W minicom-ie powinny napływać kolejne komunikaty z gps.
- 4. Zainstalować daemon gps oraz programy typu klient:

sudo apt-get install gpsd gpsd-clients

5. Wyłączyć domyślne ustawienia uruchamiania daemona gps

```
sudo systemctl stop gpsd.service
sudo systemctl stop gpsd.socket
sudo systemctl disable gpsd.service
sudo systemctl disable gpsd.socket
```

6. Uruchomić daemon gps następująco (można jeszcze dodać opcję "-N" - aby nie został uruchomiony w trybie daemon-a, tzn. odłączony od konsoli):

sudo gpsd /dev/serial0 -F /var/run/gpsd.sock

7. Uruchomić testowego klienta gps:

cgps -s

Po chwili powinny pojawić się zdekodowane dane z GPS.

8. Napisać program do odczytu szerokości i długości geograficznej w pythonie.

Wykorzystamy działający - uruchomiony wcześniej daemon gpsd.

Potrzebna jest biblioteka: python-gps.

# UWAGA! Nie nazywać programu: gps.py - ponieważ spowoduje to konflikt z wykorzystywaną biblioteką gps

Listing programu:

```
import gps
import webbrowser

if __name__ == "__main__":
    print("Odczyt danych z gpsd.")
    g = gps.gps()
    g.stream(gps.WATCH_ENABLE | gps.WATCH_NEWSTYLE)

while True:
    r = g.next()
    if r['class'] == "TPV":
        lon = r['lon']
        lat = r['lat']
        print("Szerokość geograficzna: "+str(lat)+" Długość geograficzna: "+str(lon))
        webbrowser.open("http://www.google.com/maps/place/"+str(lat)+","+str(lon))
        break
```

- Zainstalować pakiety firefox-esr albo chromium-browser.
- Program przetestować w środowisku graficznym X.
- W rezultacie wykonania programu powinna pojawić się mapa google w przeglądarce z zaznaczoną obecną pozycją gps. Uwaga: uruchomienie przeglądarki zajmuje trochę czasu.

## Interfejs I2C

#### UWAGA:

Należy zwrócić uwagę, czy właściwy moduł jądra obsługuje interfejs I2C, ponieważ może się zdarzyć, że włączony będzie starszy moduł, który nie współpracuje prawidłowo z magistralą.

Aktualnie załadowane w pamięci moduły obsługujące i2c można wyświetlić poleceniem:

lsmod | grep i2c

Dla Raspberry PI 3, powinny to być: *i2c\_bcm2835*, natomiast blędnie może być załadowany moduł: *i2c\_bcm2708*.

W przypadku obecności niewłaściwego modułu zalecany jest update/upgrade systemu oraz narzędzia raspi-config.

sudo apt-get update
sudo apt-get dist-upgrade
sudo raspi-config

W narzędziu raspi-config wykonać Update.

#### Testowanie obecności urządzeń I2C na magistrali 1 Raspberry PI 3

Adresy urządzeń podłączonych do magistrali I2C, można przeskanować, wykorzystującode pakiet narzędziowy *i2c-tools* wykonując polecenie:

i2cdetect -y 1

#### Zadanie: Odczyt danych z cyfrowego precyzyjnego barometru MPL3115A2

Podłączenie:

Wyprowadzenia modułu MPL3115A2	Wyprowadzenia Raspberry PI (pin na złączu)
INT2	Nie podłączone
INT1	Nie podłączone
SDA	SDA - GPIO2(3)
$\operatorname{SCL}$	SCL - GPIO3(5)
VCC	3V3(1)
GND	GND (6)

Obsługa interfejsu I2C jest możliwa np. z wykorzystaniem biblioteki smbus. Listing programu:

#### #!/usr/bin/env python

```
from smbus import SMBus
import time
# Special Chars
deg = u'\N{DEGREE SIGN}'
# I2C Constants
ADDR = 0x60
CTRL_REG1 = 0x26
PT_DATA_CFG = 0x13
bus = SMBus(1)
who_am_i = bus.read_byte_data(ADDR, 0x0C)
print hex(who_am_i)
if who_am_i != 0xc4:
    print "Device not active."
    exit(1)
# Set oversample rate to 128
setting = bus.read_byte_data(ADDR, CTRL_REG1)
newSetting = setting \mid 0x38
bus.write_byte_data(ADDR, CTRL_REG1, newSetting)
```

```
# Enable event flags
bus.write_byte_data(ADDR, PT_DATA_CFG, 0x07)
```

```
# Toggle One Shot
setting = bus.read_byte_data(ADDR, CTRL_REG1)
if (setting & 0x02) == 0:
    bus.write_byte_data(ADDR, CTRL_REG1, (setting | 0x02))
# Read sensor data
print "Waiting for data..."
status = bus.read_byte_data(ADDR,0x00)
while (status & 0x08) == 0:
    #print bin(status)
    status = bus.read byte data(ADDR,0x00)
    time.sleep(0.5)
print "Reading sensor data..."
p_data = bus.read_i2c_block_data(ADDR,0x01,3)
t_data = bus.read_i2c_block_data(ADDR,0x04,2)
status = bus.read_byte_data(ADDR,0x00)
print "status: "+bin(status)
p_msb = p_data[0]
p_{csb} = p_{data[1]}
p_lsb = p_data[2]
t_msb = t_data[0]
t_{lsb} = t_{data[1]}
pressure = (p_msb << 10) | (p_csb << 2) | (p_lsb >> 6)
p_decimal = ((p_lsb & 0x30) >> 4)/4.0
celsius = t_msb + (t_lsb >> 4)/16.0
print "Pressure and Temperature at "+time.strftime('%m/%d/%Y %H:%M:%S%z')
print str(pressure+p_decimal)+" Pa"
print str(celsius)+deg+"C"
```

## Interfejs SPI

#### Zadanie: Odczyt danych z przetwornika analogowo-cyfrowego MCP3208.

Przetwornik MCP3208 jest przetwornikiem analogowo-cyfrowym, 8-kanałowym, 12 bitowym z sukcesywną aproksymacją i interfejsem SPI.

Przygotować układ mierzący temperaturę ciała człowieka z wykorzystaniem scalonego czujnika LM35 i wyświetlający ją w postaci bar-graph'u stworzonego z 5 diod LED (2 czerwone, 2 żółte, 1 zielona). Każda dioda LED musi posiadać rezystor 330  $\Omega$  ograniczający prąd.

Wykorzystać:

- 2 x dioda LED czerwona
- 2 x dioda LED żółta
- 1 x dioda LED zielona
- 5 x rezystor 330  $\Omega$
- 1 x kondensator 100 nF
- 1 x przetwornik A/C MCP3208
- 1 x scalony czujnik temperatury LM35

#### Realizacja:

1. Zmontuj układ wg schematu



Figure 1: Schemat podłączenia przetwornika MCP3208 oraz diod LED do Raspberry PI 3. Oznaczenie BCM jest tożsame z oznaczeniem GPIO.

- 2. Przygotuj program w Python-ie z wykorzystaniem biblioteki gpiozero do odczytu danych z przetwornika, przetworzenia ich na temperaturę w stopniach Celsjusza, wyświetlenia na ekranie oraz wskazania na zbudowanym bar-graph'ie.
- W bibliotece gpiozero istnieją klasy obsługujące przetworniki A/C MCP, w tym także MCP3208 (Zwróć uwagę, że są także inne przetworniki np *MCP3008*, a my wykorzystujemy *MCP3208*. Zwróć także uwagę, że czujnik temperatury podłączony jest do kanału (Channel) 7, przetwornika A/C). Przeszukaj i wykorzystaj dokumentację API biblioteki gpiozero.
- Uwaga! Metoda *value* zwraca wartość po prztworzeniu, unormowaną do zakresu <0,1>. Jeśli potrzebna jest wartość bez normowania, to należy skorzystać z metody *raw\_value*.
- Uwaga!! Napięcie referencyjne przetwornika to **3.3V**.
- Znajdź w nocie katalogowej układu LM35 przelicznik temperatura napięcie i wykorzystaj w programie.
- Do obsługi bar-graph'u wykorzystać w bibliotece gpiozero klas<br/>ęLedBarGraph
- 3. Zademonstruj działanie bar-graph'u, który powinien wskazywać, czy temperatura jest normalna, występuje stan podgorączkowy czy też gorączka.

## Stanowisko laboratoryjne

#### Podzespoły potrzebne do realizacji ćwiczenia:

- Raspberry Pi
- Zasilacz do Raspberry Pi
- Płytka prototypowa
- Przewody do połączeń wewnętrznych na płytce prototypowej
- moduł GPS
- moduł przetwornika A/C MCP3208
- moduł cyfrowego precyzyjnego barometru MPL3115A2
- Rezystory  $330\Omega$
- Kondensatory 100nF
- scalony czujnik temperatury LM35
- diody LED czerwone

- diody LED żółte
- diody LED zielone

Pin#	NAME		NAME	Pint
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	00	DC Power 5v	04
05	GP1003 (SCL1 , 12C)	00	Ground	06
07	GPIO04 (GPIO_GCLK)	00	(TXD0) GPI014	08
09	Ground	0 🔾	(RXD0) GPI015	10
11	GPIO17 (GPIO_GEN0)	00	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	00	Ground	- 14
15	GPIO22 (GPIO_GEN3)	00	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	00	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	$\odot \circ$	Ground	20
21	GPIO09 (SPI_MISO)	00	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	00	(SPI_CE0_N) GPIO08	24
25	Ground	00	(SPI_CE1_N) GPIO07	26
27	ID_SD (IPC ID EEPROM)	$\odot$	(I <sup>2</sup> C ID EEPROM) ID_SC	28
29	GP1005	00	Ground	30
31	GPIO06	00	GPI012	32
33	GPI013	00	Ground	- 34
35	GPI019	00	GPIO16	36
37	GP1026	00	GP1020	38
39	Ground	00	GPIO21	40

Figure 2: Złącze 40-pin Raspberry Pi.