Kieszonkowy Linux

Raspberry Pi Zero może pełnić funkcję małego komputera edukacyjnego. Ze względu na niewielkie rozmiary można zabrać go ze sobą wszędzie i z pomocą laptopa wgłębiać się w tajniki Linuxa. Dodając do niego kilka typowych układów współpracujących po magistrali I²C, konwerter USB/UART oraz nadając całości formę wygodnego modułu, korzystającego z portu USB, otrzymujemy mobilne laboratorium edukacyjne za przystępną kwotę.

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT-5802

Podstawowe parametry:
- wbudowany konwerter USB-UART do komunikacji poprzez terminal SSH,
- wbudowany 2-kanałowy, 16-bitowy przetwornik ADC,
- wbudowany 4-bitowy, dwukierunkowy ekspander wejść/wyjść cyfrowych oraz ekspander z wyjściami PWM,
- wbudowana pamięć EEPROM oraz czujnik temperatury,
- praktyczna i wygodna konstrukcja w formie USB Stick.

Projekty pokrewne na www.media.avt.pl:

AVT-5776	Miniaturowa czujka ruchu dla Raspberry Pi, Arduino i nie tylko (EP 6/2020)
AVT-5770	Arduino i nie tylko (EP 5/2020)
AVT-5761	Czterokanałowy moduł przekaźnikowy sterowany I2C (EP 4/2020)

* **Uwaga!** Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!

Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [**UK**] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [**C**] – zmontowany, uruchomiony i przetestowany zestaw [**B**] (elementy wlutowane w płytkę PCB)
- wersja [**A**] – płytka drukowana bez elementów i dokumentacji

Kity w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:
- wersja [**A+**] – płytka drukowana [**A**] + zaprogramowany układ [**UK**] i dokumentacja
- wersja [**UK**] – zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! *http://sklep.avt.pl*. W przypadku braku dostępności na *http://sklep.avt.pl*, osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: kity@avt.pl.

Najważniejszym elementem modułu jest konwerter USB/UART, w którym zastosowano układ FT234X (U5) który służy do obsługi komunikacji terminalu SSH pomiędzy Pi Zero a podłączonym PC, przy wykorzystaniu portu szeregowego, dostępnego na wyprowadzeniach GPIO Raspberry Pi. Dzięki temu mamy komunikację i możliwość zarządzania RPi przez terminal, a jedyne złącze USB pozostaje dostępne dla użytkownika. Nie ma potrzeby użycia droższego RPi ZeroW z wbudowanym Wi-Fi. Moduł pobiera zasilanie z komputera PC, bezpośrednio poprzez wtyk USB. Zasilanie 5 V wraz z sygnałami USB D+/D– z punktów lutowniczych, dostępnych na płytce RPi Zero, doprowadzone są do gniazda USB, umożliwiając podłączenie np. pendrive'a, karty sieciowej, itp.

Budowa i działanie

Schemat ideowy modułu został pokazany na **rysunku 1**. Na płytce znalazło się miejsce na układy z interfejsem I²C, obsługujące typowe GPIO. Układ U1 typu MCP3426 pełni funkcję przetwornika AD z dwoma wejściami, o zakresie mierzonego napięcia 0…2,048 V i maksymalnej rozdzielczości 16 bitów. Sygnał analogowy ze złącza A01, po filtracji realizowanej za pomocą elementów R1, R2, C1, C2, doprowadzony jest bezpośrednio do układu U1, typu MCP3426. Ma on ustalony adres I²C o wartości 0x68 (dla MCP3426A0). Zwracam uwagę na konflikt adresów z często używanym układem RTC typu DS1338. Jeżeli planujemy zastosować na magistrali I²C także zewnętrzny RTC, należy wykorzystać w miejsce U1 układ o zaprogramowanym fabrycznie adresie, różnym od 0x68, np. MCP3426A1 do MCP3246A7.

Układ U2 typu PCA9536D jest 4-bitowym, dwukierunkowym ekspanderem wejść/wyjść cyfrowych. Wyprowadzenia U2 IO0…3 dostępne są na złączach D01, D23. Układ ma ustalony adres I²C o wartości 0x41. Układ U4 typu PCA9632 jest ekspanderem umożliwiającym sterowanie bezpośrednie LED małej mocy, a w naszej aplikacji pełni funkcje wyjść PWM. Wyprowadzenia LED0 i LED1 dostępne są na złączu PWM01. Wyprowadzenia LED2 i LED3 sterują wbudowanymi diodami LED opisanymi jako PWM2 i PWM3. Układ U4 ma ustalony wyprowadzeniami A0 i A1 adres równy 0x60.

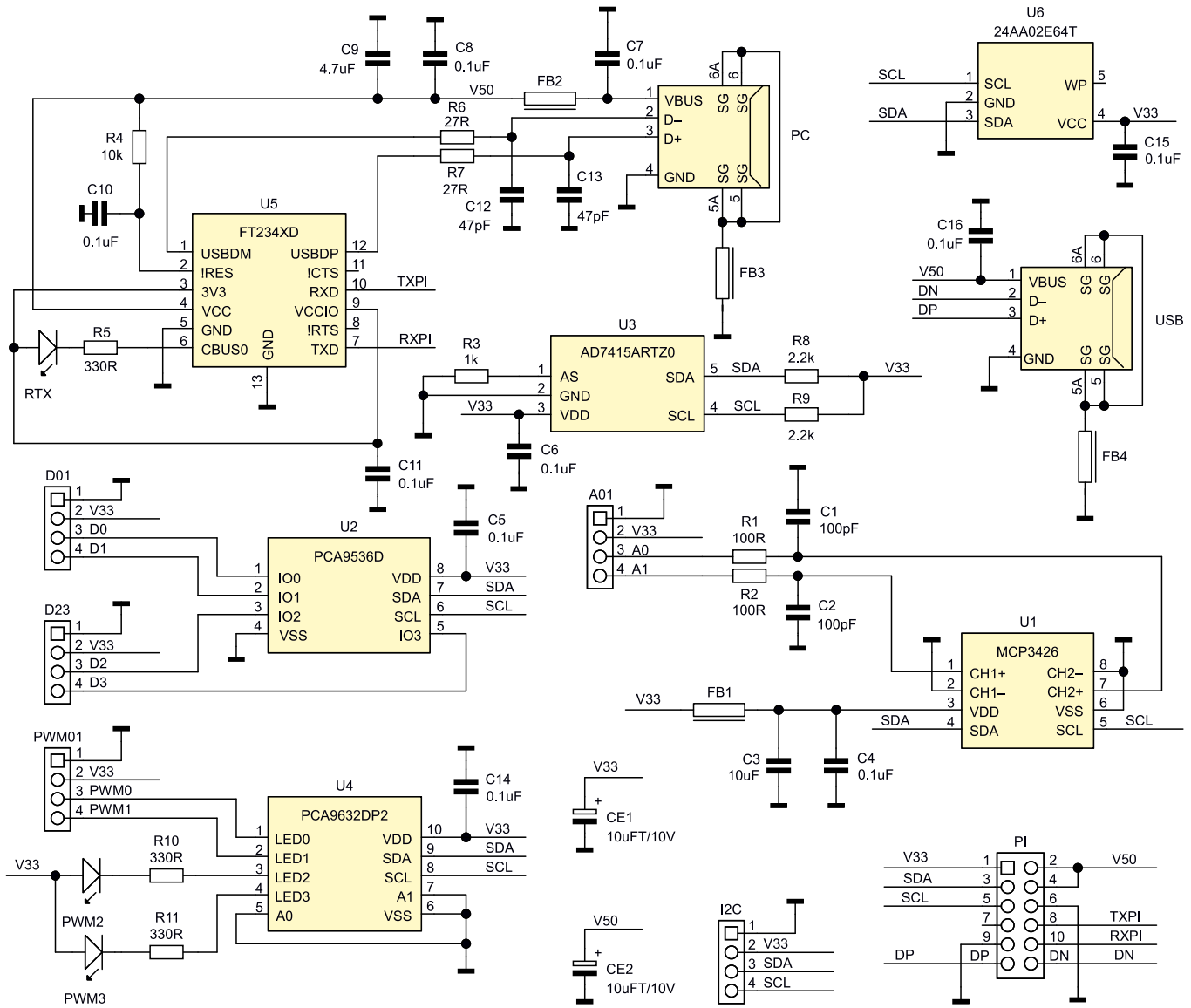
Na płytce znalazło się miejsce także na termometr I²C, U3 typu AD7415, umożliwiający pomiar temperatury płytki, dostępny pod adresem 0x49. U6 typu 24AA02E64 jest pamięcią EEPROM (adres 0x50…0x57), w której oprócz obszaru dostępnego dla użytkownika zapisany jest trwale unikalny 8-bajtowy ciąg wartości, który może służyć jako adres MAC lub może znaleźć zastosowanie jako klucz identyfikacyjny/autoryzacyjny dla RPi Zero.

Wszystkie układy zasilane są napięciem 3,3 V z RPi Zero i z takimi poziomami logicznymi mogą współpracować. Płytkę uzupełniają kondensatory odsprzęgające i rezystory R8 i R9 polaryzujące magistralę I²C. Ze względu na niewielką ilość miejsca jako złącza IO wykorzystane zostały SM04B_SRSS_TB o rozstawie 1,0 mm, stosowane także w modułach I²C zgodnych z QWIIC.

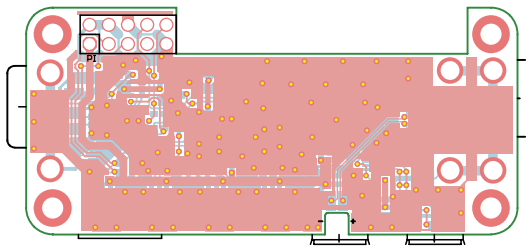
Montaż i uruchomienie

Moduł zmontowany jest na dwustronnej płytce drukowanej, zgodnej mechanicznie z RPi Zero. Schemat płytki, wraz z rozmieszczeniem elementów, pokazują **rysunki 2** i **3**, a zmontowany moduł został pokazany na fotografii tytułowej. Sposób montażu jest klasyczny i nie wymaga opisu.

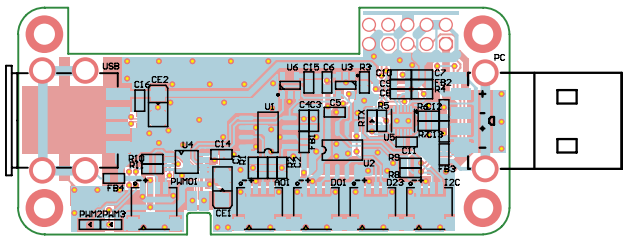
Moduł jest montowany do RPi Zero za pomocą wkrętów M2.5. Pomiędzy Stickiem a RPI Zero koniecznie jest umieszczenie płytki izolacyjnej, wyciętej z pleksi lub



Rysunek 1. Schemat modułu



Rysunek 2. Schemat płytki PCB wraz z rozmieszczeniem elementów – strona TOP



Rysunek 3. Schemat płytki PCB wraz z rozmieszczeniem elementów – strona BOTTOM

innego materiału, o grubości 0,25…0,5 mm, w kształcie RPi Zero. Płytka izolacyjna z odpowiednio wyciętymi otworami i szczelinami w okolicy GPIO zapobiega zwarciom pomiędzy stykającymi się mechanicznie obwodami. Po wlutowaniu należy skrócić wyprowadzenia ustalające złącza USB/PC, aby nie wystawały ponad płytkę i nie uszkodziły izolacji pomiędzy płytkami. W modelu do RPI Zero wlutowane jest także pełne złącze GPIO, ponieważ często korzystam z różnych innych modułów HAT, które mocowane są do dystansów M2.5×10. Po skręceniu płytki z RPi, w miejscach wykorzystanych przez PI GPIO należy kroplą cyny połączyć pola lutownicze obu płytek. Wyprowadzenia złącza wystające ponad płytkę RPi Zero ułatwiają lutowanie, ale należy zwrócić uwagę na ewentualne zwarcia pomiędzy pinami. Połączenie interfejsu USB wymaga użycia krótkich odcinków odizolowanego kynaru i ostrożnego przylutowania ich bezpośrednio do punktów testowych USB na RPI Zero, co pokazuje **fotografia 1**.

Jeżeli nie przewidujemy wykorzystania GPIO w miejscach punktów lutowniczych, należy umieścić kawałki kynaru i zlutować płytki z dwóch stron, zapewniając kontakt.

Wykaz elementów:

Rezystory: (SMD0603 1%)
R1, R2: 100 Ω
R3: 1 kΩ
R4: 10 kΩ
R5, R10, R11: 330 Ω
R6, R7: 27 Ω
R8, R9: 2,2 kΩ

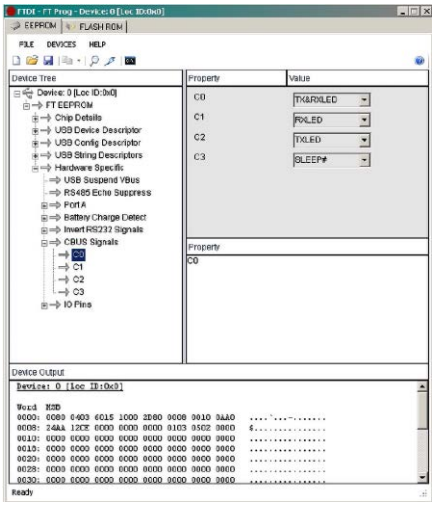
Kondensatory:
C1, C2: 100 pF SMD0603
C3: 10 µF SMD0603
C4…C8, C10, C11, C14…C16: 0,1 µF SMD0603
C9: 4,7 µF SMD0603
C12, C13: 47 pF SMD0603
CE1, CE2: 10 µF/10 V SMD3216 tantalowy

Półprzewodniki:
PWM2, PWM3: dioda led SMD0603 zielona
RTX: dioda led SMD0603 czerwona
U1: MCP3426 (SO8)
U2: PCA9536D (SO8)
U3: AD7415ARTZ0 (SOT-23-5)
U4: PCA9632DP2 (TSSOP10_050)
U5: FT234XD (DFN12_045)
U6: 24AA02E64T (SOT-23-5)

Pozostałe:
A01, D01, D23, I2C, PWM01: złącze JST 4 pin 1 mm kątowe
FB2: ferryt WE-TMSB SMD 1,5 A WE74269262601
FB1, FB3, FB4: ferryt WE-CBF SMD 0,5 A WE742792653
PC: wtyk USB A SMD
USB: gniazdo USB A SMD

Brak złącza GPIO skutkuje niewielką wysokością modułu. Decyzję o jego wlutowaniu pozostawiam użytkownikowi.

Po zmontowaniu modułu konieczne jest skonfigurowanie FT234 za pomocą oprogramowania *FT Prog* ze strony FTDI. Po detekcji układu (Scan & Parse F5) wymagana jest zmiana funkcji (Hardware Specyfic\ CBUS Signals) pinu C0 na TX&RXLED. Wbudowana LED RTX będzie wtedy sygnalizowała pełną komunikację pomiędzy PC a RPi. Po zmianie ustawień (**rysunek 4**) należy układ zaprogramować (CTRL+P) i zrestartować port USB (Cycle Ports).



Rysunek 4. Zmiana konfiguracji FT234

Prawidłowo zmontowany moduł nie wymaga uruchamiania, konieczne jest tylko niewielkie dostosowanie systemu operacyjnego. Najlepszym wyborem dla Sticka jest Raspbian w wersji Lite. Po zainstalowaniu systemu na karcie SD należy edytować plik *config.txt*:

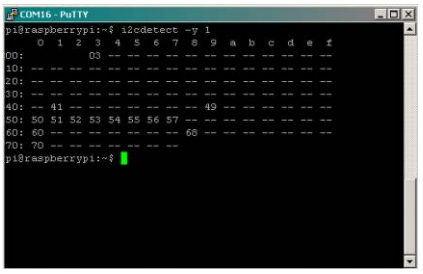
- załączyć obsługę I²C: `dtparam=i2c_arm=on`,
- załączyć obsługę UART: `enable_uart=1`.

W pliku *cmdline.txt*:

- ustawić konsolę na port szeregowy: `console=serial0, 115200 console=tty1`.

Dostęp do obu plików jest możliwy bezpośrednio w głównym katalogu, po instalacji systemu na karcie SD.

Na komputerze PC należy zainstalować terminal, np. Putty i skonfigurować połączenie na port szeregowy (pod adresem COM, z którym zgłosi się FT234), z parametrami



Rysunek 5. Detekcja układów z magistrali I²C

115200,8,N,1. Po uruchomieniu sesji powinno być możliwe zalogowanie się do RPi na domyślnych ustawieniach pi/raspberry (co należy bezwzględnie zmienić, korzystając z raspiconfig). Teraz, o ile nie korzystamy z RPi Zero, przyda się karta sieciowa podłączona do portu USB, ze względu na konieczność pobrania kilku narzędzi i aktualizacji systemu:

```
sudo apt-get update
sudo apt-get upgrade
```

Potrzebne będą narzędzia *i2ctools*, interpreter *Pythona* i biblioteki *smbus*, *time*.

```
sudo apt-get update
```

Listing 1. Skrypt testowy mcp3426.py

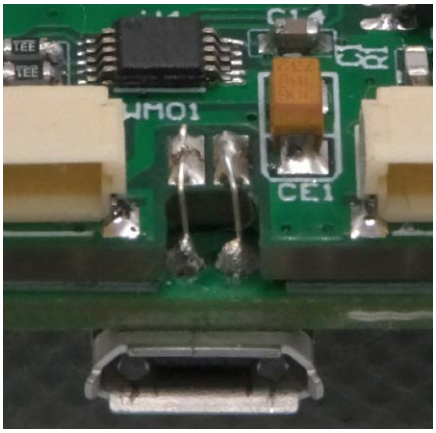
```
import smbus
import time
from time import sleep

print "\nMCP4326 ADC Test\n"

# Get I2C bus
bus = smbus.SMBus(1)

# MCP3426 0x68 CH1
bus.write_byte(0x68, 0x10)
time.sleep(0.1)

# MCP3426 0x68 Read 0x00 3 bytes
data = bus.read_i2c_block_data(0x68, 0x00, 3)
adch = data[0]
print "ADCH 1 : %d" %adch
adcl = data[1]
print "ADCL 1 : %d" %adcl
adc = (data[0] & 0x0F) * 256 + data[1]
print "Analog Input 1: %d" %adc

# MCP3426 0x68 CH2
bus.write_byte(0x68, 0x30)
time.sleep(0.1)

# MCP3426 0x68 Read 0x00 3 bytes
data = bus.read_i2c_block_data(0x68, 0x00, 3)
adch = data[0]
print "ADCH 2 : %d" %adch
adcl = data[1]
print "ADCL 2 : %d" %adcl
adc = (data[0] & 0x0F) * 256 + data[1]
print "Analog Input 2: %d" %adc
bus.close()
print"\nQuit\n"
```

Listing 2. Skrypt testowy pca9536.py

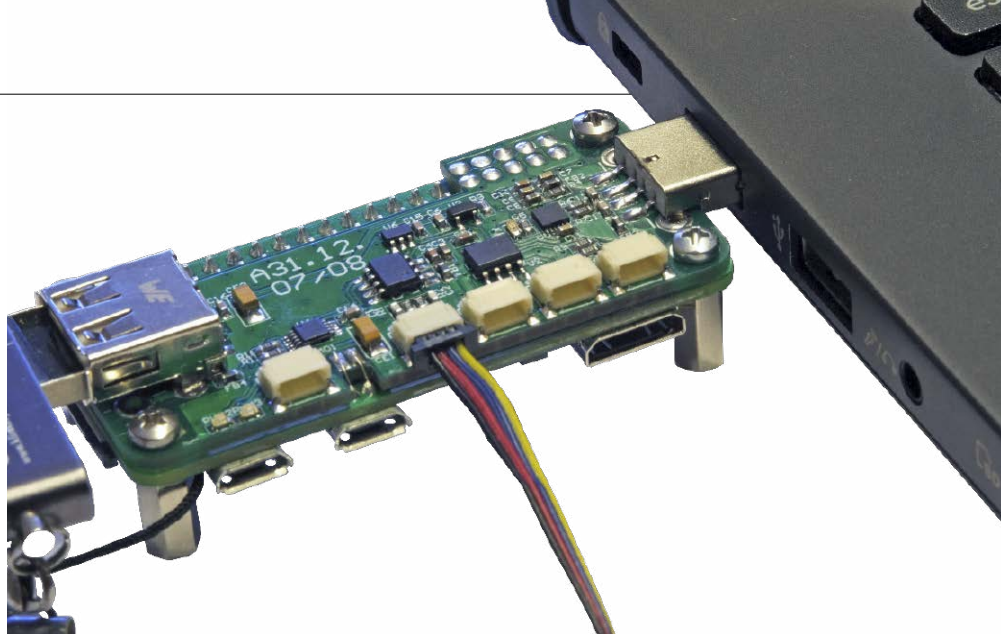
```
import smbus
import time
from time import sleep

print "\nPCA9536 GPIO Test\n"

# Get I2C bus
bus = smbus.SMBus(1)

# PCA9536 0x41
bus.write_byte_data(0x41, 0x03, 0x00)
bus.write_byte_data(0x41, 0x01, 0x01)
time.sleep(0.1)
x=5;

# PCA9536 0x41 Write Output
print " Out 1 set"
bus.write_byte_data(0x41, 0x01, 0x01)
time.sleep(x)
print " Out 2 set"
bus.write_byte_data(0x41, 0x01, 0x02)
time.sleep(x)
print " Out 3 set"
bus.write_byte_data(0x41, 0x01, 0x04)
time.sleep(x)
print " Out 4 set"
bus.write_byte_data(0x41, 0x01, 0x08)
time.sleep(x)
print " Out ALL set"
bus.write_byte_data(0x41, 0x01, 0x0F)
time.sleep(x)
print " Out ALL reset"
bus.write_byte_data(0x41, 0x01, 0x00)
bus.close()
time.sleep(x)

print "\nQuit\n"
```

Listing 3. Skrypt testowy pca9632.py

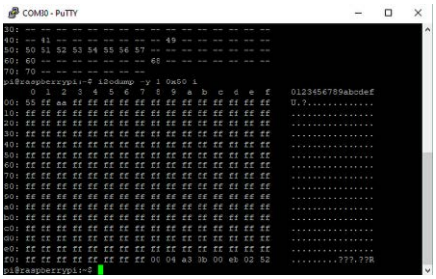
```
import smbus
import time
from time import sleep

print "\nPCA9632 PWM Test\n"

# Get I2C bus
bus = smbus.SMBus(1)

# PCA9632 0x60 config
bus.write_byte_data(0x60, 0x00, 0x00)
bus.write_byte_data(0x60, 0x01, 0x14)
bus.write_byte_data(0x60, 0x02, 0x00)
bus.write_byte_data(0x60, 0x03, 0x00)
bus.write_byte_data(0x60, 0x04, 0x00)
bus.write_byte_data(0x60, 0x05, 0x00)
bus.write_byte_data(0x60, 0x06, 0xFF)
bus.write_byte_data(0x60, 0x07, 0x00)
bus.write_byte_data(0x60, 0x08, 0xAA)
bus.write_byte_data(0x60, 0x09, 0xE2)
bus.write_byte_data(0x60, 0x0A, 0xE4)
bus.write_byte_data(0x60, 0x0B, 0xE8)
bus.write_byte_data(0x60, 0x0C, 0xE0)
time.sleep(0.1)
x=5;

# PCA9632 PWM
print " PWM Out 1 set 25%"
bus.write_byte_data(0x60, 0x02, 0x3F)
print " PWM Out 2 set 50%"
bus.write_byte_data(0x60, 0x03, 0x7F)
print " PWM Out 3 set 75 - LED 25%"
bus.write_byte_data(0x60, 0x04, 0xBD)
print " PWM Out 4 set 100% - LED 0%"
bus.write_byte_data(0x60, 0x05, 0xFF)
time.sleep(x)

print " PWM Global"
bus.write_byte_data(0x60, 0x08, 0xFF)
bus.write_byte_data(0x60, 0x06, 0x3F)
time.sleep(x)
bus.write_byte_data(0x60, 0x06, 0xBD)
time.sleep(x)

print " BLINK Global"
bus.write_byte_data(0x60, 0x08, 0xFF)
bus.write_byte_data(0x60, 0x01, 0x34)
bus.write_byte_data(0x60, 0x07, 0x0F)
time.sleep(x)

print " All 0% - LED ON"
bus.write_byte_data(0x60, 0x01, 0x14)
bus.write_byte_data(0x60, 0x08, 0xAA)
bus.write_byte_data(0x60, 0x02, 0x00)
bus.write_byte_data(0x60, 0x03, 0x00)
bus.write_byte_data(0x60, 0x04, 0x00)
bus.write_byte_data(0x60, 0x05, 0x00)
time.sleep(x)

print " LED OFF"
bus.write_byte_data(0x60, 0x04, 0xFF)
bus.write_byte_data(0x60, 0x05, 0xFF)
bus.close()

print "\nQuit\n"
```



Fotografia 1. Sposób połączenia USB



Fotografia 2. Przykładowa aplikacja modułu



Rysunek 6. Odczyt zawartości EEPROM

```
sudo apt-get install python-smbus python3-smbus python-dev python3-dev i2c-tools
```

Po restarcie systemu, dla sprawdzenia poprawności detekcji układów, wydajemy polecenie (**rysunek 5**):

```
sudo i2cdetect -y 1
```

Dla sprawdzenia poszczególnych układów przygotowałem proste skrypty. Przetwornik ADC MCP3426 można przetestować skryptem *mcp3426.py* z **listingu 1**. Po podłączeniu do wejść A0, A1, np. dzielnika z potencjometrem (maksymalne napięcie na wejściu 2,048 V) i uruchomieniu skryptu powinny zostać odczytane wartości napięć na jego wejściach. Do sprawdzenia ekspandera GPIO PCA9536 użyjemy skryptu *pca9536.py* (**listing 2**), kolejno zmieniającego stan wyjść układu, dostępnych na złączach D01, D23. Ekspander PWM PCA9632 sprawdzimy skryptem *pca9632.py* (**listing 3**). Steruje on wyjściami PWM dostępnymi na złączu PWM01 i wbudowanymi LED. Skrypt odczytujący temperaturę zmierzoną przez AD7415.py pokazuje **listing 4**. Testowanie pamięci 24AA02E64 można wykonać, zarówno korzystając z *i2ctools*, jak i ze skryptu *24AA02E064.py* (**listing 5**). Skrypt odczytuje ostatnie osiem bajtów zawierających adres MAC. Zawartość całej pamięci można odczytać poleceniem:

```
i2cdump -y 1 0x50 i
```

Rezultat pokazuje **rysunek 6.** Do zapisu komórki można użyć polecenia i2cset, przykładowo zapisując pod adres EEPROM 0x50 i pod adres 0x00 wartość 0xA5:

```
i2cset -y 1 0x50 0x00 0xA5
```

Do odczytu z adresu 0x50 i spod adresu 0x00 używamy polecenia:

```
i2cget -y 1 0x50 0x00
```

Rezultat powinien zwrócić wartość zapisaną wcześniej – 0xA5. Jeżeli wszystkie testy przebiegły pomyślnie, można zastosować moduł we własnych aplikacjach (**fotografia 2**). Powodzenia!

Listing 4. Skrypt ad7415.py odczytujący temperaturę AD7415

```
import smbus

print "\nAD7415-0 0x49 Temp\n"

# Get I2C bus
bus = smbus.SMBus(1)

# AD7415 read 8-bytes temp

temp = bus.read_byte_data(0x49, 0x00)
print"temp: (‘C): "
print(temp)
bus.close()
print “\nQuit\n”
```

Listing 5. Skrypt odczytujący MAC zapisany w 24AA02E64

```
import smbus

print "\n24AA02E64 MAC Dump\n"

# Get I2C bus
bus = smbus.SMBus(1)

# 24AA02E64 read last (0xF8 offset) 8-bytes MAC ROM

macblock = bus.read_i2c_block_data(0x50, 0xF8, 0x08)
print"MAC (hex): "
print([‘{:02x}’.format(x) for x in macblock])
print"MAC (dec): "
print(macblock)
bus.close()
print “\nQuit\n”
```

Adam Tatuś
adam.tatus@ep.com.pl