Internet of Things, zagadnienia sieciowe

Angelika Tefelska Dariusz Tefelski

2023-05-29

Celem ćwiczenia jest zapoznanie z technonlogiami przydatnymi w realizacji projektów typu *IoT*: Internet of Things.

Uwagi wstępne:

- W celu minimalizacji zagrożenia uszkodzenia Raspberry Pi, wszelkie podłączenia należy wykonywać po wyłączeniu zasilania.
- Należy zwrócić uwagę, czy włączona jest obsługa 1-wire w zainstalowanym systemie. Domyślnie do obsługi 1-wire wykorzystywany jest pin *GPIO4* (BCM4).

Interfejs 1-wire

W ramach ćwiczenia wykorzystamy 2 czujniki temperatury *DS18B20*.

W Python-ie do obsługi czujników temperatury z interfejsem 1-wire dostępna jest biblioteka *w1thermsensor*.

Aby ją zainstalować należy:

- zainstalować pakiet python3-pip

```
sudo apt install python3-pip
```

- zainstalować moduł Pythona 3 do obsługi czujników temperatury one-wire:

```
sudo pip3 install w1thermsensor
```

W module w1thermsensor oprócz wersji biblioteki dla Python'a 3, jest także narzędzie do obsługi czujników temperatury z poziomu powłoki: w1thermsensor

Zadanie 1

Zbudować układ mierzący temperaturę z wykorzystaniem 2 czujników DS18B20 oraz wskazujący za pomocą diod LED, który z czujników wskazuje wyższą temperaturę.

Elementy:

- czujnik DS18B20 x 2
- rezystor 4k7 x 1
- rezystory 330 x 3
- diody LED (czerwona, zielona, żółta)

Układ ma wskazywać za pomocą diod LED, który z czujników pokazuje wyższą temperaturę (czerwona albo zielona, natomiast żółta ma się świecić w przypadku gdy temperatura obu czujników nie różni się więcej niż o 0.5 stopnia Celsjusza).

Program wykonać w Pythonie, z wykorzystaniem biblioteki *w1thermsensor* Informacje dotyczące obsługi tej biblioteki dostępne są na stronie: https://pypi.org/project/w1thermsensor albo: https://github.com/timofurrer/w1thermsensor

Wykonanie w linii poleceń polecenia:

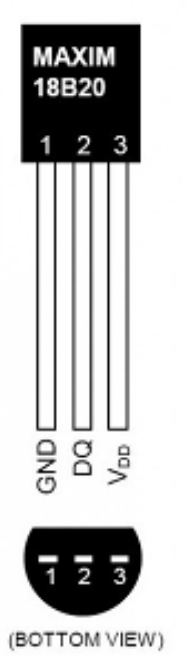

Figure 1: Wyprowadzenia DS18B20

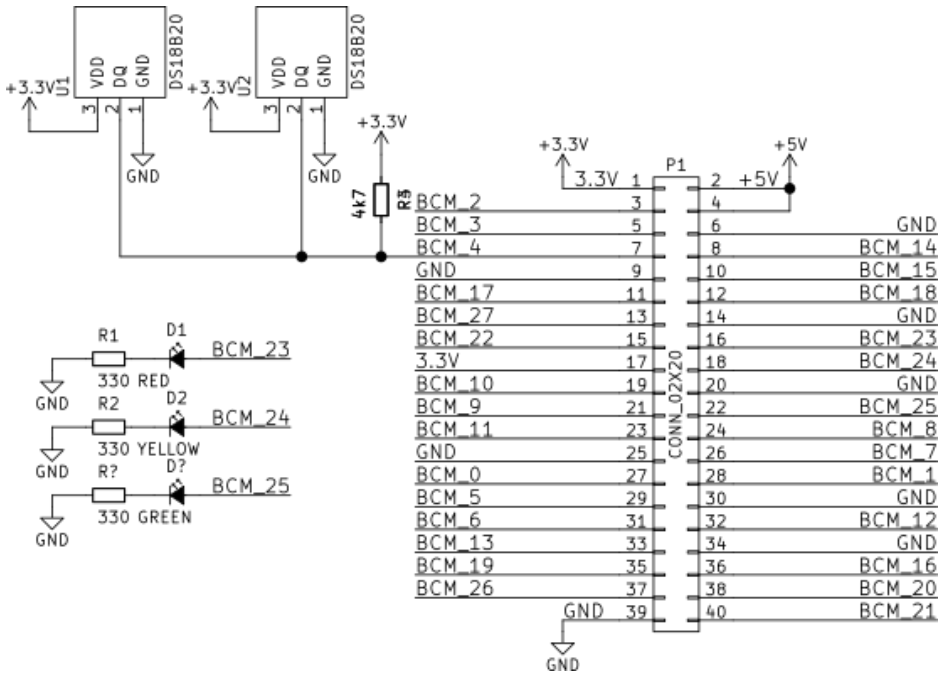

Figure 2: Schemat układu z 2 czujnikami DS18B20

```
w1thermsensor all
```

spowoduje wyświetlenie temperatur z dostępnych czujników.

Odczyt danych z pojedynczego sensora w python:

```
from w1thermsensor import W1ThermSensor, Sensor

sensor = W1ThermSensor(Sensor.DS18B20, "tutaj adres sensora")
temperature_in_celsius = sensor.get_temperature()
```

gdzie w polu "tutaj adres sensora" - podać należy unikalny adres sensora widoczny np. z komendy *w1thermsensor all*.

Do wysterowania diody LED wykorzytać bibliotekę *gpiozero*.

Przykładowy kod w Python'ie:

```
#!/usr/bin/env python3
from w1thermsensor import W1ThermSensor, Sensor
from gpiozero import LED
red = LED(23)
yellow = LED(24)
green = LED(25)
sensor1 = W1ThermSensor(Sensor.DS18B20, "000001c7965f")
sensor2 = W1ThermSensor(Sensor.DS18B20, "000001c796cf")
while(True):
    temp1 = sensor1.get_temperature()
    temp2 = sensor2.get_temperature()
    if abs(temp1 - temp2) < 0.5:
        red.off()
        yellow.on()
        green.off()
    elif temp1 > temp2:
        red.on()
        yellow.off()
        green.off()
    elif temp1 < temp2:
        red.off()
        yellow.off()
        green.on()
    print("T1: %.2f st.C T2: %.2f st.C" % (temp1, temp2))
```

Sterowanie diod LED z poziomu przeglądarki internetowej

W języku Python dostępne są pakiety pełniące funkcje serwera www i ułatwiające tworzenie interaktywnych stron internetowych. Przykładem są np. mikroframework'i takie jak *Bottle* czy *Flask*.

Zadanie 2

Za pomocą git-a pobrać projekt, w którym zademonstrowane jest sterowanie włączaniem i wyłączaniem diod LED za pomocą strony internetowej.

Instalacja (jeśli brakuje) zarządcy pakietów dla języka Python:

```
sudo apt install python3-pip
```

Instalacja mikroframework-a *Bottle*:

```
sudo pip3 install bottle
```

Jeśli nie ma jeszcze zainstalowanego git-a, to należy go zainstalować:

```
sudo apt install git
```

Pobranie projektu:

```
git clone https://github.com/rizansari/RPi-LED-Web.git
cd RPi-LED-Web
```

- Wykonać zmiany w oprogramowaniu, tj. w plikach: *main.py* oraz *main.html*, tak aby można było sterować 3 a nie 2 diodami LED oraz aby kolory diod zgadzały się ze stroną internetową. **UWAGA** Należy pamiętać o ustawieniu właściwych numerów pinów - wg. opisu *BCM - GPIO* oraz o konfiguracji pinów jako wyjściowych.
- Przetestować działanie sterownika, poprzez:

  Uruchomienie oprogramowania:

  ```
  python3 main.py
  ```

- Podłączenie w przeglądarce lokalnej (na Raspberry PI) na adres:

```
http://localhost:8081
```

- Podłączenie zdalne np. za pomocą telefonu podłączonego do tej samej sieci wifi co Raspberry PI, Podając w przeglądarce internetowej na komputerze PC adres Raspberry Pi (można np. odczytać z polecenia

```
ip -c a
```

W przeglądarce podajemy:

```
http://adres_ip_raspberry_pi:8081
```

Docker

Aplikacje, które wykorzystują mechanizmy sieciowe często wymagają różnego typu usług sieciowych, które zwykle trzeba dostosować i skonfigurować, np. serwer www, baza danych itp. Różne dystrybucje Linuks-a często mają odmienne sposoby konfiguracji, inne zestawy bibliotek i inne nazewnictwo pakietów. Stwarza to często problem, gdy potrzeba skonfigurować kilka takich aplikacji na danym systemie operacyjnym.

Aby ułatwić dystrybucję aplikacji łączącej różne technologie zaczęto wykorzystywać technologie wirtualizacyjne. Obecnie nowoczesnym podejściem jest wykorzystanie technologii *Docker*-a, który jest wirtualizacją na poziomie systemu operacyjnego. Pozwala ona na uruchomienie np. innej dystrybucji Linux-a, z własnym zestawem narzędzi, która jest skonfigurowana pod wykonywanie docelowej aplikacji.

Główne zalety dotyczą minimalizacji narzutu związanego z uruchamianiem innej dystrybucji i przekazywaniem danych. Standardowe technologie wirtualizacyjne izolują w większym stopniu uruchamiany system, emulując hardware maszyny wirtualnej i przez to są mniej wydajne. Przechowywanie danych w *Docker*ze jest ponadto zoptymalizowane w ten sposób, że obok obrazu początkowego systemu przechowywane są tylko zmiany, które nastąpiły, bądz wprowadził je użytkownik.

Instalacja *Docker*-a.

2 proponowane rozwiązania:

- Z poziomu pakietów systemu operacyjnego Raspberry PI OS:

  ```
  sudo apt install docker.io
  ```

- Bezpośrednio ze strony docker'a:

```
curl -sSL get.docker.com |sh
```

Dodanie użytkownika pi do grupy docker

```
sudo usermod -a -G docker pi
newgrp docker
```

Zadanie 3

- Wykorzystując *Docker*-a zainstalować prosty testowy serwer www:

```
docker run -d -p 80:80 hypriot/rpi-busybox-httpd
```

- Sprawdzić działanie serwera wpisując w przeglądarce na Raspberry Pi adres: *http://localhost* albo z komputera PC wprowadzając adres IP Raspberry Pi: *http://adres_ip_raspberry_pi

Domoticz

Internet rzeczy (IoT) chociaż obejmuje bardzo szerokie zagadnienia, postrzegany jest często w ramach rozwiązań typu Inteligentny Dom.

Zadaniem jest zainstalowanie systemu *Domoticz* znajdującego zastosowanie jako centralny system zarządzający inteligentnym domem, wykorzystując oprogramowanie *Docker*.

Zadanie 4

Wykorzystać układ 2 czujników temperatury DS18B20 z zadania 1 jako czujników temperatury i skonfigurować je w oprogramowaniu Domoticz, tak aby można było zobaczyć bierzącą temperaturę w zakładce *Dashboard*

Uruchomienie kontenera domoticz-a

Wszystkie potrzebne obrazy zostaną pobrane, a następnie doker utworzy kontener i go uruchomi.

Ważne: Utworzyć na początku pusty katalog np. */home/pi/domoticz_config*:

```
mkdir /home/pi/domoticz_config
```

Aby uruchomić kontener domoticza, wystarczy wydać następującą komendę.

```
docker run -d -p 8080:8080 -p 8443:443 -v <katalog_z_konfiguracja>:/opt/domoticz/userdata \
-e TZ=Europe/Warsaw --name=dom domoticz/domoticz
```

Sprawdzenie działających kontenerów:

```
docker ps
```

Sprawdzenie wszystkich obecnych kontenerów:

```
docker ps -a
```

Sprawdzenie obrazów:

```
docker images
```

Dostępne polecenia w dockerz-e wyświetlą się po wpisaniu:

```
docker
```

Podłączenie się poprzez konsolę do działającego kontenera

```
docker exec -it <nazwa_kontenera> bash
```

Dostęp do domoticz-a:

Dostęp lokalny przez przeglądarkę internetową:

`http://localhost:8080`

Aby dostać się do domoticza można na zewnętrznym komputerze podłączonym do tej samej sieci, co Raspberry Pi wpisać w przeglądarkę:

`http://adres_IP_raspberry_pi:8080`

Ustawianie czujników temperatury w Domoticz-u:

Należy skonfigurować czujniki temperatury poprzez:

- Zakładka *Setup* -> *Settings* -> ustawić język polski
- Zakładka *Konfiguracja* -> *Sprzęt* -> ustawić typ: *1-Wire (System)*, dodać nazwę *DS18B20* oraz usunąć zawartość pola **OWFS PATH:**. Zmienić czas odpytywania sensora na krótszy, np. 9000 ms. Nacisnąć przycisk *Dodaj*
- Zakładka *Konfiguracja* -> *Urządzenia* - powinny się pojawić podłączone czujniki DS18B20, należy je dodać klikając zieloną strzałkę przy każdym z nich
- Sprawdzić zakładkę *Temperatura*. Sprawdzić logi, zaznaczając gwiazdkę, czujniki powinny pojawić się także w zakładce *Pulpit* (Dashboard).

Współpraca Raspberry PI i Arduino

Celem zadania jest praktyka łączenia technologii Arduino z Raspberry PI.

Zadanie 6. Sterowanie silnikiem DC

- Zamontować na płytce prototypowej sterownik silnika DC. Patrz schemat. Podłączyć silnik oraz sterowanie do Arduino. Do wejścia *EN_A* podłączyć pin zapewniający obsługę *PWM*, do wejścia *PH_A* podłączyć dowolny pin, skonfigurowany jako wyjście cyfrowe

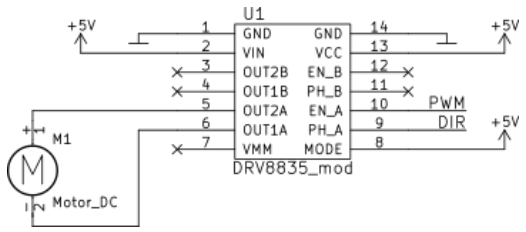

Figure 3: Schemat podłączenia sterownika silnika.

Nr pinu	Nazwa	Funkcja
1	GND	Masa
2	VIN	Zasilanie silnika (2,7V - 10,8V)
3	OUT2B	Wyprowadzenie na silnik B
4	OUT1B	Wyprowadzenie na silnik B
5	OUT2A	Wyprowadzenie na silnik A
6	OUT1A	Wyprowadzenie na silnik A
7	VMM	Wyjście zasilania za zabezpieczeniem przeciw odwrotnemu połączeniu
8	MODE	Wejście logiczne - ustala sposób sterowania
9	PH_A	Kanał A - wybór kierunku obrotów w trybie podstawowym
10	EN_A	Kanał A - włączanie silnika, sterowanie sygnałem PWM
11	PH_B	Kanał B - wybór kierunku obrotów w trybie podstawowym
12	EN_B	Kanał B - włączanie silnika, sterowanie sygnałem PWM
13	VCC	Zasilanie części logicznej dla Arduino +5V, dla Raspberry PI +3,3V
14	GND	Masa

- Zainstalować środowisko Arduino IDE na Raspberry PI.

Można je zainstalować poniższą komendą:

```
sudo apt install arduino
```

Uruchomienie Arduino następuje przez polecenie: arduino.

- Czasami może być lepiej zainstalować najnowsze środowisko, pobrane ze strony arduino.cc i skonfigurować je lokalnie na swoim koncie.
- Arduino można pobrać ze strony **https://www.arduino.cc** - wersja dla **Linux ARM 32-bit** i rozpakować w katalogu domowym, tzn. powinniśmy otrzymać katalog */home/pi/arduino-1.8.19*.

Uruchomienie środowiska *Arduino IDE 1.8.19* polega na wydaniu polecenia: **~/arduino-1.8.19/arduino**

- Do komunikacji przez port szeregowy w Pythonie potrzebujemy zainstalować moduł python3-serial

```
sudo apt install python3-serial
```

- Zainstalować **Arduino Makefile** Do starszej wersji można to zrobić wydając polecenie:

```
sudo apt install arduino-mk
```

Do nowszej wersji, może być konieczne pobranie ze strony https://www.arduino.cc. Należy pobrać nowy makefile z repozytorium, będąc w katalogu domowym (**/home/pi**).

```
git clone https://github.com/sudar/Arduino-Makefile.git
```

Po uruchomieniu środowiska arduino, powinien utworzyć się katalog **/home/pi/Arduino**. Można go także stworzyć samemu + katalog **/home/pi/Arduino/libraries**.

W katalogu **/home/pi/Arduino** powinny się w znajdować szkice użytkownika w podkatalogach o nazwach takich jak pliki programów bez rozszerzenia **.ino**. Dodatkowo znajduje się tam katalog */home/pi/Arduino/libraries*. Do którego należy zainstalować wszystkie potrzebne biblioteki zewnętrzne.

Realizacja programistyczna:

1. Napisać szkic arduino, w którym za pomocą portu szeregowego (Serial) będzie można sterować prędkością obrotową silnika (pwm 0-255) - funkcja *analogWrite()* oraz kierunkiem obrotów silnika - funkcja *digitalWrite()*.

Sterowanie powinno reagować na komendy:

- speed <0...255>
- dir <0...1>
- status

Pamiętać należy o:

- Funkcjach *void setup()* oraz *void loop()*
- Inicjalizacji portu szeregowego: *Serial.begin(115200)*.
- Konfiguracji pinów sterujących - *pinMode(... , OUTPUT)*.
- Do realizacji wygodnie posłużyć się obiektem *String*, który zawiera metody takie jak: *startsWithUntil()*, *substring()*, *toInt()*.
- Wykorzystać Arduino Makefile do budowania i programowania Arduino:
- Plik *Makefile* dla Arduino zainstalowanego z pakietów systemowych powinien wyglądać następująco:

```
BOARD_TAG    = mega
BOARD_SUB    = atmega2560
ARDUINO_PORT = /dev/ttyACM*
ARDUINO_LIBS = OneWire DallasTemperature
ARDUINO_VERSION = 156

#CPPFLAGS     = -DDEBUG

include /usr/share/arduino/Arduino.mk
```

- Plik *Makefile* dla zainstalowanej nowszej wersji (lokalnie na koncie) powinien wyglądać następująco:

```
BOARD_TAG    = mega
BOARD_SUB    = atmega2560
ARDUINO_PORT = /dev/ttyACM*

ARDUINO_DIR  = /home/pi/arduino-1.8.19
ARDMK_DIR = /home/pi/Arduino-Makefile
AVR_TOOLS_DIR = /home/pi/arduino-1.8.19/hardware/tools/avr
ARDUINO_LIBS = OneWire DallasTemperature

#CPPFLAGS     = -DDEBUG

include /home/pi/Arduino-Makefile/Arduino.mk
```

- Wywołanie budowania:

```
make
```

- Wywołanie zaprogramowania Arduino (także wywołuje budowanie, jeśli coś się zmieniło w plikach):

```
make upload
```

2. Przygotować skrypt Python-a, który będzie wykorzystywał moduł **serial** i sterował podłączonym Arduino i docelowo silnikiem, tak aby płynnie zmieniać prędkość obrotową silnika (zmiana prędkości o 1 co 50ms) od 0 do 255 - najpierw wzrost prędkości, następnie zmniejszanie prędkości, a potem zmiana kierunku obrotów i ponownie wzrost i spadek itd.

Należy pamiętać o:

- dołączeniu modułu serial (**import serial**)
- otwarciu portu

```
s = serial.Serial("/dev/ttyACM0", 115200)
```

UWAGA! Po otwarciu portu należy dać czas Arduino na inicjalizację (ok 1 sekundy) zanim zaczniemy transmitować dane przez interfejs szeregowy. Można w tym celu wykonać operację: *sleep(1)*.

- wydanie komendy, np.

```
s.write(b"speed %d\n" % speed)
```

- odczyt zwrotnej informacji:

```
answer =s.read(s.inWaiting())
```

- doprowadzenie ciągu bajtów do postaci tekstowej (string):

```
txt_str = answer.decode("utf-8")
```

- zamiana ciągu znaków na postać ciągu bajtowego:

```
byte_str = text_string.encode("utf-8")
```

Można też od razu stworzyć ciąg bajtowy:

```
byte_str = b"To jest ciag bajtowy numer=%d" % liczba
```

Zadanie 7. Pomiar temperatury z czujników DS18B20 przez Arduino i przekazywanie danych do Raspberry Pi

1. Do Arduino (porzedni układ) dołączyć 2 czujniki temperatury DS18B20 i korzytając z biblioteki *DallasTemperature* odczytywać ich temperatury. Przygotować dodatkowe komendy np. **temp? 1**, **temp? 2**, które zwracać będą temperatury z odpowiedniego czujnika.

- Biblioteki: *OneWire* i *DallasTemperature* można zainstalować w środowisku *Arduino* wybierając Sketch->Include library->Manage Libraries ... i wyszukąć je po nazwie i zainstalować.

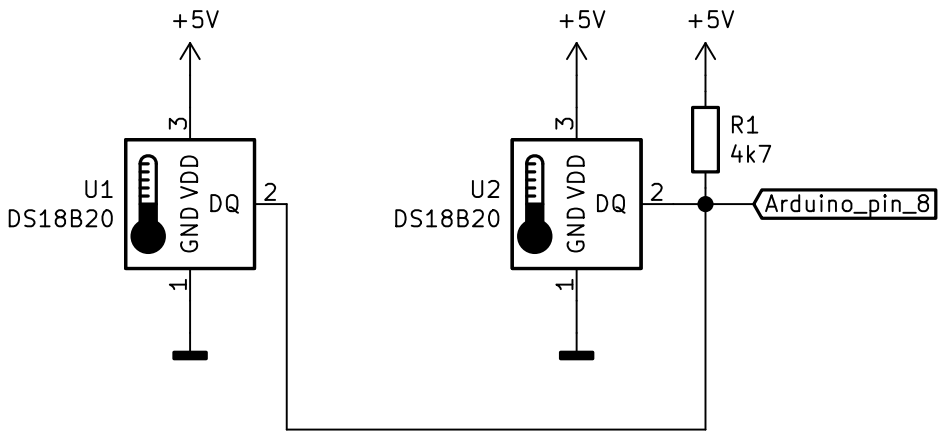

Figure 4: Schemat podłączenia czujników DS18B20 do Arduino.

- Można je także zainstalować ręcznie - pobranie biblioteki do obsługi czujników temperatury 1-wire:

```
cd ~/Arduino/libraries
git clone https://github.com/milesburton/Arduino-Temperature-Control-Library.git
mv Arduino-Temperature-Control-Library DallasTemperature
```

Opis biblioteki - dostępny na stronie autora: https://www.milesburton.com/Dallas_Temperature_Control_Library

- Pobranie biblioteki OneWire, która służy obsłudze protokołu 1-wire:

```
cd ~/Arduino/libraries
git clone https://github.com/PaulStoffregen/OneWire.git
```

2. Przygotować na Raspberry PI program w Python-ie, który będzie odczytywał i wyświetlał temperatury z obu termometrów.

Przykładowy kod na Arduino realizujący sterowanie silnikiem prądu stałego oraz odczytem z dwóch czujników temperatury 1-wire, DS18B20:

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 8
#define LED 13
#define SPEED_PIN 2
#define DIR_PIN 3

String komenda;
int speed;
int dir;

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress termometr1, termometr2;

void printAddress(const DeviceAddress deviceAddress) {
  for (uint8_t i = 0; i < 8; i++){
    if (deviceAddress[i] < 16) Serial.print("0");
    Serial.print(deviceAddress[i], HEX);
  }
}

void printTemperature(const DeviceAddress deviceAddress) {
```

```
  float tempC = sensors.getTempC(deviceAddress);
  if(tempC == DEVICE_DISCONNECTED_C){
    Serial.println("Błąd. Nie mogę odczytać temperatury");
    return;
  }
  Serial.print(tempC);
}

void setup() {
    pinMode(LED, OUTPUT);
    pinMode(SPEED_PIN, OUTPUT);
    pinMode(DIR_PIN, OUTPUT);
    Serial.begin(115200);
    Serial.setTimeout(10000);
    sensors.begin();
    Serial.print("Szukam termometrów...");
    Serial.print("Znalazłem ");
    Serial.print(sensors.getDeviceCount(), DEC);
    Serial.println(".");
    if (!sensors.getAddress(termometr1, 0)) Serial.println("Nie moge znalezc adresu termometru T1");
    if (!sensors.getAddress(termometr2, 1)) Serial.println("Nie moge znalezc adresu termometru T2");
    Serial.print("T1 adres: ");
    printAddress(termometr1);
    Serial.println();
    Serial.print("T2 adres: ");
    printAddress(termometr2);
    Serial.println();
}

void loop() {
    while(Serial.available() > 0) {
        komenda = Serial.readStringUntil('\r');
        if(komenda.startsWith("speed")) {
            speed=komenda.substring(6).toInt();
            if(speed > 255) speed = 255;
            if(speed < 0) speed = 0;
            analogWrite(SPEED_PIN,speed);
        } else if(komenda.startsWith("dir")) {
            dir = komenda.substring(4).toInt();
            if(dir>1) dir=1;
            if(dir<0) dir=0;
            digitalWrite(DIR_PIN,dir);
            digitalWrite(13,dir);
        } else if(komenda.startsWith("status?")) {
            Serial.print("SPEED: ");
            Serial.print(speed);
            Serial.print(" DIR: ");
            Serial.println(dir);
        } else if(komenda.startsWith("temp?")) {
            sensors.requestTemperatures(); //wywołanie globalne do wszystkich termometrów na magistrali
            int tn = komenda.substring(6).toInt();
            if(tn==1) printTemperature(termometr1);
            else if(tn==2) printTemperature(termometr2);
            else {
                printTemperature(termometr1);
                Serial.print("\t");
```

```
                printTemperature(termometr2);
            }
            Serial.println();
        } else {
            Serial.println("Unknown command");
        }
    }
}
```

Uwaga: W pliku Makefile odkomentować linię (usunąć znak hash z początku linii), dotyczącą dołączania biblioteki, jeśli wcześniej ją zakomentowano:

```
#ARDUINO_LIBS = OneWire DallasTemperature
```

Z Arduino można skomunikować się przez komendę:

```
picocom /dev/ttyACM0 -b 115200 -c
```

można wtedy sprawdzić komendy: status? , temp?, speed 100, speed 50, dir 1, dir 0

Przykładowy kod w Python'ie do sterowania silnikiem prądu stałego:

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

import serial
from time import sleep

if __name__ == "__main__":
    s = serial.Serial("/dev/ttyACM0", 115200)
    dir = 0
    speed = 0
    acc = 1
    print("Czekam na Arduino")
    sleep(1)
    while(True):
        s.write(b"speed %d\r" % speed)
        sleep(0.05)
        s.write(b"status?\r")
        odp = s.read(s.inWaiting())
        print(odp.decode("utf-8"))
        if acc==1:
            speed += 1
        else:
            speed -= 1
        if speed > 255:
            speed = 255
            acc = 0
        if speed < 0:
            speed = 0
            acc = 1
            dir ^= 1
            s.write(b"dir %d\r" % dir)
```

Przykładowy kod w Python'ie do odczytu czujników temperatury:

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

import serial
```

```
from time import sleep

if __name__ == "__main__":
    s = serial.Serial("/dev/ttyACM0", 115200)
    print("Czekam na Arduino")
    sleep(1)
    while(True):
        s.write(b"temp?\r")
        sleep(1)
        odp = s.read(s.inWaiting())
        print(odp.strip().decode("utf-8"))
```

Stanowisko laboratoryjne

Podzespoły potrzebne do realizacji ćwiczenia:

- Raspberry Pi
- Zasilacz do Raspberry Pi
- Płytka prototypowa
- Przewody do połączeń wewnętrznych na płytce prototypowej
- układ scalony czujnika temperatury DS18B20 x 2
- rezystory 4k7
- rezystory 330
- diody LED czerwone, żółte, zielone
- Arduino Mega2560
- Kabel USB A-B do Arduino
- Silnik DC
- Sterownik silnika DC

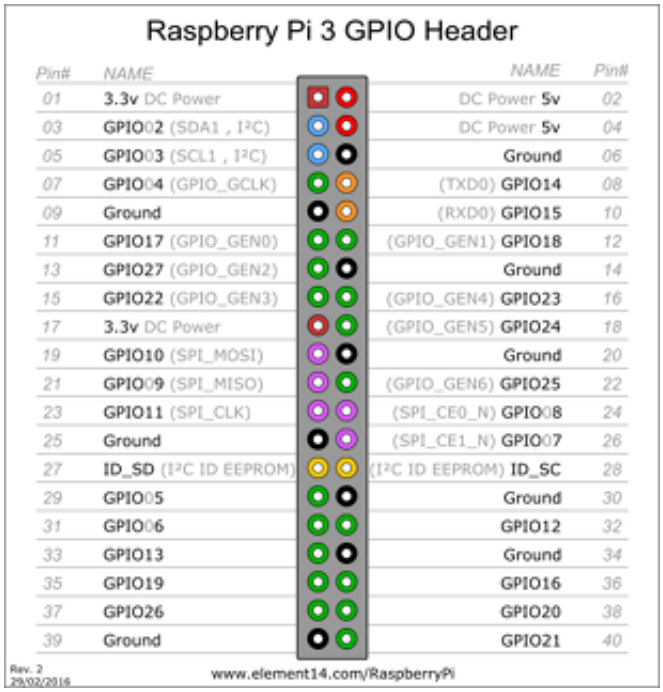

Figure 5: Złącze 40-pin Raspberry Pi.