Systemy Mechatroniczne - Laboratorium

Lab. 4 Programowanie RPi3 w Pythonie

Autorzy:
Ambroziak Arkadiusz
Sienicki Adrian

1. PWM w Pythonie

```
import RPi.GPIO as GPIO
import time
import random
LED = 18
def setup():
	global pwm
	GPIO.setmode(GPIO.BCM)
	GPIO.setup(LED, GPIO.OUT)
	pwm = GPIO.PWM(LED, 200)
	pwm.start(100)
def ustaw_jasnnosc(nowa_jasnosc):
	pwm.ChangeDutyCycle(nowa_jasnosc)
def flicker():
	ustaw_jasnosc(random.randrange(0, 100))
	time.sleep(random.randrange(1, 10) * 0.01)
def loop():
	try:
		while True:
			flicker()
	except KeyboardInterrupt:
		pass
	finally:
		GPIO.cleanup()
setup()
loop()
```

2. Odczytywanie wartości z przetwornika temperatury DS18B20

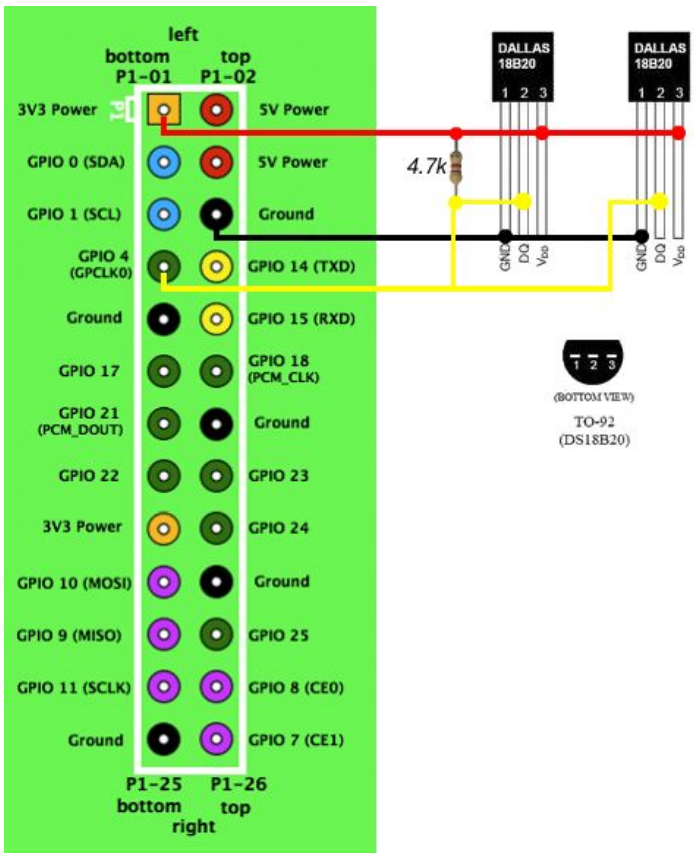
1-Wire jest interfejsem opracowanym przez firmę Dallas Semiconductor. Interfejs ten swoją popularność zawdzięcza głównie dużej liczbie urządzeń peryferyjnych obsługujących wymieniony standard (czujniki temperatury, elektroniczne moduły dostępu *iButton* , sterowniki ładowania akumulatorów, itd.), jak i również łatwej implementacji programowej (urządzenie nadzorujące transmisję nie musi posiadać sprzętowego kontrolera magistrali – programista sam może napisać proste procedury realizujące wymianę danych).

Magistrala 1-Wire jest przedstawicielem asynchronicznego interfejsu szeregowego. Komunikacja na magistrali odbywa się za pomocą jednej linii danych oraz linii masy, która stanowi poziom odniesienia dla sygnału. W tzw. trybie pasożytniczym (ang. *parasite power*) urządzenia peryferyjne nie potrzebują podłączenie osobnej linii zasilania – moc dostarczana jest przez wbudowany w układy peryferyjne kondensator gromadzący energię z linii danych.

Takie rozwiązanie redukuję liczbę przewodów magistrali, jednak wpływa negatywnie na zasięg transmisji, zwiększając obciążenie prądowe linii danych.

Komunikacja na magistrali odbywa się dwukierunkowo w trybie *master* – *slave* W typowej konfiguracji rolę *mastera* (urządzenia przeprowadzającego i nadzorującego transmisję) pełni układ mikroprocesora/mikrokontrolera (np. Raspberry Pi). Urządzeniami *slave* są wszystkie układy peryferyjne dołączane do magistrali. Ponieważ urządzenia *slave* nie zapewniają możliwości konfiguracji adresu (jak ma to miejsce np. w przypadku układów z magistralą I2C), każdy z układów musi zawierać fabrycznie zdefiniowany unikatowy numer seryjny. Standard 1-Wire definiuje dla każdego z urządzeń peryferyjnych 64-bitowy kod identyfikacyjny zapisany w nieulotnej pamięci ROM.[1]

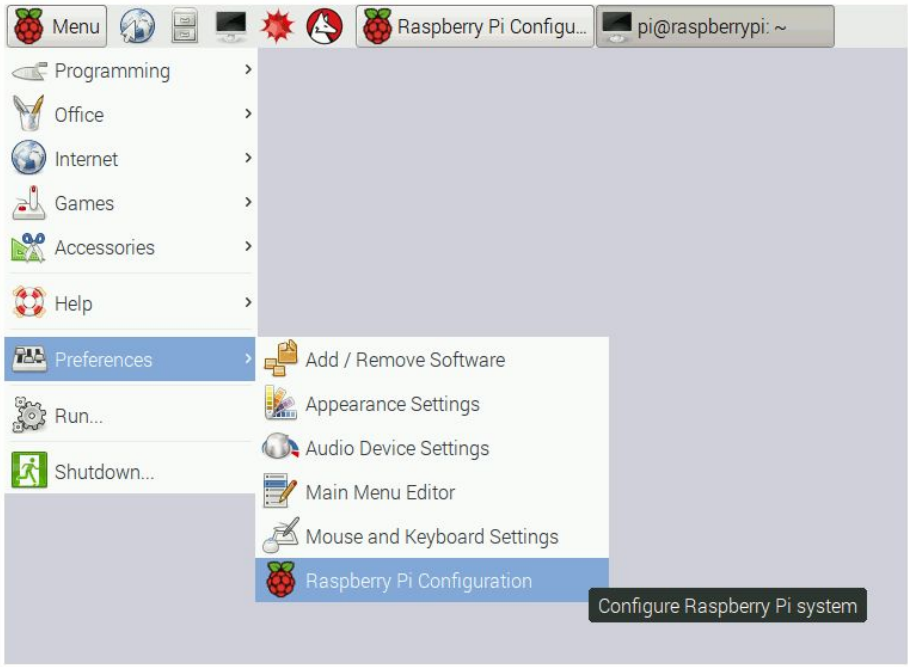
Raspberry Pi posiada wbudowaną obsługę 1-Wire. Pin obsługujący magistralę ma numer 4. Sposób podłączenia czujników przedstawia Rys. 2.1.



Rys 2.1. Podłączenie czujników DS18B20 do Raspberry Pi

Obsługę magistrali można włączyć za pomocą wbudowanego w system Raspbian narzędzia “Raspiconfig”. Wersję graficzną konfiguratora można znaleźć w menu systemu (Rys. 2.2). W tym celu należy uruchomić konfigurator i w zakładce “Interfaces” zaznaczyć opcję 1-Wire. Następnie zatwierdzić zmiany.

[1] http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf



Rys. 2.2. Konfigurator Raspberry Pi

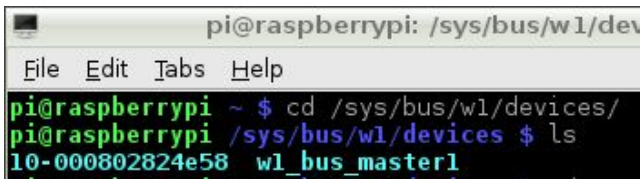
Odczytanie zmierzonej wartości temperatury za pomocą konsoli

Wartości odczytane z magistrali 1-Wire Raspberry Pi przechowuje w pliku tekstowym znajdującym się w folderze *"/sys/bus/w1/devices/adres_czujnika/"*. Odczyt danych odbywa się poprzez przeniesienie się do tego folderu i odczyt pliku tekstowego przypisanemu danemu czujnikowi. Po otwarciu okna terminala należy wykonać następujące komendy (każdą należy zatwierdzić klawiszem Enter):

cd /sys/bus/w-1/devices/

ls

W konsoli zostaną wyświetlone foldery oraz pliki znajdujące się w folderze */sys/bus/w1/devices/.* Folder zawierający plik ze zmierzoną temperatura znajduje się w folderze, którego nazwą jest nazwa czujnika (Rys. 2.3). Nazwy czujników DS18B20 w Raspberry Pi zaczynają się od zestawu znaków *28-*.



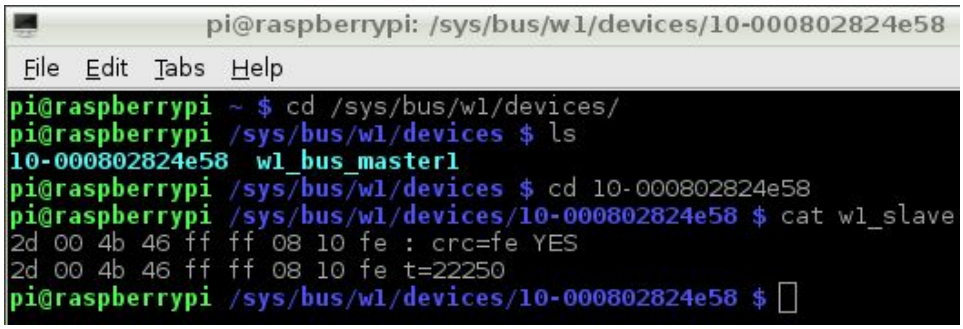
Rys. 2.3. Foldery magistrali 1-Wire

Następnie należy wejść do folderu czujnika i wyświetlić zawartość pliku z odczytaną temperaturą za pomocą komend:

cd adres_czujnika

cat w1_slave

Końcowy wynik zaprezentowano na rys. 2.4.



Rys. 2.4. Odczytana wartość z czujnika

Skrypt w języku Python do odczytu temperatury

```
# Otwarcie pliku, w którym znajdują się dane na temat temperatury
plik = open("/sys/bus/w1/devices/adres_czujnika/w1_slave")
# Odczytanie tekstu z pliku
tekst = plik.read()
# Zamknięcie pliku
plik.close()
# Oddzielenie tekstu na podstawie znaku nowej linii (\n) i wybranie drugiego wiersza.
drugalinia= text.split("\n")[1]
#Podzielenie linii na słowa, na podstawie znaku spacji oraz zapisanie 10 słowa do zmiennej (licząc od zera - tablica)
temperaturadate = drugalinia.split(" ")[9]
# Dwa pierwsze znaki to “t=”, więc je pomijamy i pozostałą wartość konwertujemy na typ liczbowy zmiennoprzecinkowy
temperatura = float(temperaturadata[2:])
#Temperatura jest podawana w formie liczby całkowitej, więc trzeba podzielić odczytaną wartość przez 1000
temperatura = temperatura / 1000
print temperatura
```

Skrypt w języku Python do odczytu temperatury (w pętli)[2]

Początek programu trzeba napisać analogicznie jak zostało to zrobione z wiersza poleceń w Raspbianie, czyli załadować moduły do obsługi magistrali 1-Wire oraz załadować niezbędne biblioteki.

```
import os
import glob
import time


os.system ('modprobe w1-gpio')      #załadowanie modułu 1-wire GPIO
os.system ('modprobe w1-therm')     #załadowanie modułu 1-wire THERM
```

[2] P. Kubicki, “Ethernetowy wielokanałowy rejestrator temperatury oparty o platformę komputerową Raspberry Pi”, Praca dyplomowa magisterska, 2016.

Następnie definiujemy zmienną przechowującą folder bazowy, w którym znajdują się foldery czujników. Utworzenie folderu takiej zmiennej ułatwia późniejsze odwoływanie się do kolejnych czujników, ponieważ nie ma potrzeby przepisywania dosyć długiej ścieżki dostępu.

```
folder_czujnikow = '/sys/bus/w1/devices/'
```

Następnym krokiem jest utworzenie zmiennej przechowującej ścieżki dostępu do folderu konkretnego czujnika. „Folder_czujnika_t1" odpowiada czujnikowi o numerze 28-000006d383a6.

```
folder_czujnika_t1 = glob.glob(folder_czujnikow + '28-000006d383a6')[0]
```

Gdy zdefiniowany jest folder czujnika to kolejnym etapem jest określenie polecenia do odczytu wartości temperatury, czyli „w1_slave".

```
plik_czujnika_t1 = folder_czujnika_t1 + '/w1_slave'
```

W ten sposób została zdefiniowana ścieżka dostępu do czujnika 28-000006d383a6. Poniżej fragment skryptu z Pythona.

```
folder_czujnikow = '/sys/bus/w1/devices/'
#Temperatura 1 deklaracja lokalizacji
folder_czujnika_t1 = glob.glob(folder_czujnikow + '28-000006d383a6')[0]
plik_czujnika_t1 = folder_czujnika_t1 + '/w1_slaves'
```

„Plik_czujnika_t1" jest ścieżką dostępu pliku, do którego trzeba będzie się odwoływać w dalszej części programu.

W celu odczytania wartości temperatury konieczne jest otworzenie pliku czujnika, którego ścieżka dostępu została już zdefiniowana powyżej. W tym celu definiujemy funkcję o nazwie „odczyt_pliku_t1", w której otwierany jest plik (za pomocą polecenia „open"). Wiersze pliku są odczytywane przy wykorzystaniu polecenia „readlines". Odczyt kończony jest poleceniem „close".

```
def odczyt_pliku_t1():
    f=open(plik_czujnika_t1,'r')
    lines=f.readlines()
    f.close()
    return lines
```

Funkcja „odczyt_pliku_t1" zwraca zawartość wszystkich danych z czujnika temperatury. W pliku tym jak było widać w wierszu poleceń (Rys. 2.4) oprócz informacji o zawartości temperatury były inne informacje nie przydatne na nasze potrzeby. Kolejnym zadaniem było uzyskanie z odczytanego pliku informacji o wartości temperatury. Do tego celu została zdefiniowana kolejna funkcja „odczyt_pliku_t1".

```
def odczyt_pliku_t1():
        lines=odczyt_pliku_t1()
        while lines[0].strip()[-3:]!='YES':
                time.sleep(0.2)
                lines=odczyt_pliku_t1()
        equals_pos=lines[1].find('t=')
        if equals_pos != -1:
                temp_string = lines [1][equals_pos+2:]
                temp_c_1 = float(temper_string)/1000.0
        return temp_c_1
```

W funkcji „odczyt_pliku_t1” zostaje utworzona zmienna lines odwołująca się do funkcji „odczyt_pliku_t1”. Następnie trzeci znak od końca zerowego wiersza jest sprawdzany pod względem zezwolenia odczytu. Jest to wewnętrzny test czujnika. „Equals_pos” została przypisana wartość znajdująca się obok po litery ciągu znaków „t=”, czyli wartości temperatury danego czujnika. Ostatnim zadaniem było podzielenie wartości przez tysiąc, ponieważ wartość z czujnika nie obsługuje części dziesiętnych. Wyświetlenie temperatury zrealizowane zostało za pomocą polecenia „print”.

```
while True:
        print('Temperatura')
        t1=odczyt_pliku_t8()
        print(t1)
```

3. Literatura

http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf