Urządzenia Peryferyjne

Sprawozdanie

RaspberryPi

Damian Koper, 241292

Wiktor Pieklik, 241282

13 stycznia 2020

Spis treści

1 Cel ćwiczenia **3**

2 RaspberryPi **3**

2.1 Piny 4

2.2 System operacyjny 4

3 Dioda LED **4**

3.1 Sterowanie z poziomu powłoki systemowej 5

3.2 Sterowanie z poziomu języka Python 6

3.3 Sterowanie przyciskiem 7

4 Pomiar temperatury **8**

5 Pomiar odległości **8**

6 Podsumowanie **8**

Spis rysunków

1 RaspberryPi w wersji 4B[3]. 3

2 Piny RaspberryPi w wersji 4B[4]. 4

3 Podłączenie diody LED. 5

4 Podłączenie diody LED i przycisku. 7

Spis listingów

1 Obsługa diody LED z poziomu powłoki systemowej. 5

2 Obsługa diody LED z poziomu jezyka Python. 6

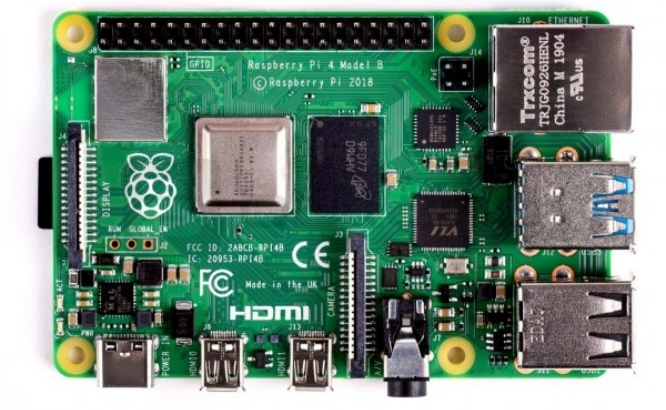
3 Sterowanie diodą LED za pomocą przycisku. Najważniejsze fragmenty. 7

1 Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z urządzeniem RaspberryPi w wersji 4B[1]. Należało obsłużyć miganie diodą z poziomu powłoki systemowej, języka Python oraz wykonać pomiar temperatury i odległości z wykorzystaniem odpowiednich sensorów.

2 RaspberryPi

RaspberryPi to komputer umieszczony na płytce o wymiarach 85,60 x 56,50 mm. Posiada on 64-bitowy czterordzeniowy procesor Cortex-A72 wykorzystujący architekturę ARM[2]. W zależności od wersji posiada od 1GB do 4GB pamięci RAM. RaspberryPi wyposażone jest również w port Ethernet, 3.5mm jack oraz 2x micro HDMI, interfejs CSI oraz DSI. Posiada również wyprowadzenia na 40 pinów, które realizują wiele popularnych interfejsów oraz mogą być wykorzystywane w dowolny sposób.



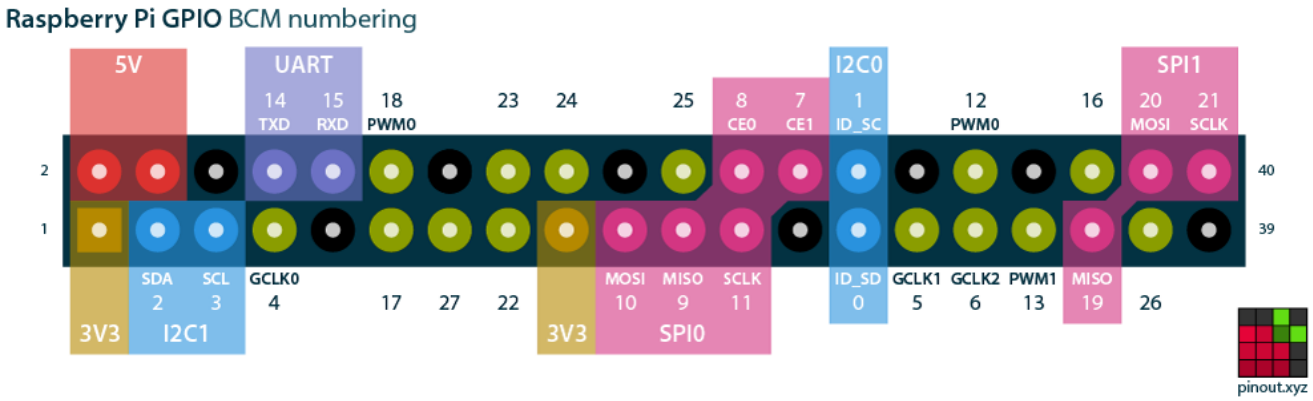
Rysunek 1: RaspberryPi w wersji 4B[3].

2.1 Piny

W RaspberryPi wyprowadzone piny odpowiedzialne są za:

- wyprowadzenie masy,
- zasilanie 3.3V i 5V,
- interfejs UART,
- interfejs I^2C,
- interfejs SPI.

Rysunek 2 prezentuje jak rozmieszczone są piny odpowiedzialne za poszczególne funkcje. Piny GPIO (General Purpose IO) mogą być używane jako piny wejściowe lub wyjściowe, mogą również pełnić funkcje zegara lub stanąć się wyjściem działającym w trybie PWM.



Rysunek 2: Piny RaspberryPi w wersji 4B[4].

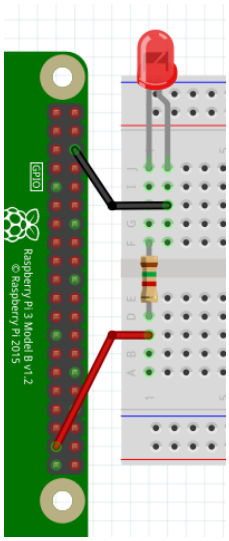
2.2 System operacyjny

Na RaspberryPi uruchomione mogą zostać systemy skompilowane na architekturę ARM. Najpopularniejsze z nich oparte są na Linuxie, jednym z nich jest system Raspbian[5]. Na RaspberryPi może uruchomiony również zostać system Windows 10 w wersji *Internet of Things*[6].

3 Dioda LED

Podstawowa obsługa diody LED polegała na włączaniu i wyłączaniu jej co sekundę. Układ został podłączony zgodnie ze schematem z rysunku 4. Do zaświecenia diodą wystarczy prąd płynący

przy ustawieniu stanu pinu GPIO na wysoki. Pin GPIO na wyjściu dostarcza napięcie 3.3V i może obsłużyć natężenie maksymalnie 50mA[7].



Rysunek 3: Podłączenie diody LED.

3.1 Sterowanie z poziomu powłoki systemowej

Jądro Linuxa domyślnie wyposażone jest w sterowniki obsługujące interface GPIO. Jego cechą, tak jak wszystkich systemach operacyjnych opartych ba Unixie, jest obsługa urządzeń poprzed interface udostępniany użytkownikowi w postaci pliku. Sprowadza to obsługę urządzenia do wykonywania operacji odczytu i zapisu do określonych plików.

W przypadku obsługi diody LED w pierwszej kolejności należało aktywować odpowiedni pin w trybie wyjścia - linie 3 i 4 w listingu 1. Następnie w nieskończonej pętli skrypt zapisywał do pliku przypisanego do aktywowanego pinu wartości `0` i `1`. Skutkowało to zmianą napięcia na wyjściu, a w konsekwencji zapalenie i gaśnięcie diody.

```
1 #!/bin/bash
2 PIN=26
3 echo $PIN > /sys/class/gpio/export
4 echo out > /sys/class/gpio/gpio${PIN}/direction
5 while true; do
6   echo 0 > /sys/class/gpio/gpio${PIN}/value
7   echo "OFF"
8   sleep 1s
9   echo 1 > /sys/class/gpio/gpio${PIN}/value
```

```
  echo "ON"
  sleep 1s
done
```

Listing 1: Obsługa diody LED z poziomu powłoki systemowej.

3.2 Sterowanie z poziomu języka Python

Sterowanie diodą z poziomu języka Python odbywa się na podobnej zasadzie co wcześniej. Użyta biblioteka RPi.GPIO posiada API, które wewnętrznie również korzysta ze sterowników jądra systemu operacyjnego.

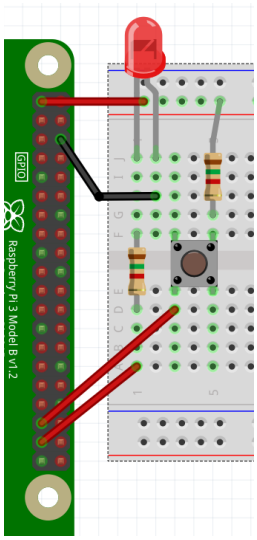
```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM);
ledPin = 26;
GPIO.setup(ledPin, GPIO.OUT, initial=GPIO.HIGH);
while True:
  print ("OFF");
  GPIO.output(ledPin, GPIO.LOW);
  time.sleep(1);
  print ("ON");
  GPIO.output(ledPin, GPIO.HIGH);
  time.sleep(1);
```

Listing 2: Obsługa diody LED z poziomu jezyka Python.

Wywołanie funkcji `GPIO.setmode` odpowiada za ustawienie mapowania numeru pinu na jego fizyczny odpowiednik. Funkcja `GPIO.setup` ustawia kierunek i początkową wartość dla pinu, a funkcja `GPIO.output` zmienia wyjście pinu na wartość z argumentu.

3.3 Sterowanie przyciskiem

Cośtam blabla



Rysunek 4: Podłączenie diody LED i przycisku.

```
GPIO.setup(ledPin, GPIO.OUT, initial=GPIO.HIGH)
GPIO.setup(btnPin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
ledOn = True
def button_callback(channel):
  global ledOn
  time.sleep(0.015)
  btnInput = GPIO.input(btnPin)
  if btnInput is GPIO.HIGH:
    if ledOn is True:
      ledOn = False
      print ("OFF")
      GPIO.output(ledPin, GPIO.LOW)
    else:
      ledOn = True
      print ("ON")
      GPIO.output(ledPin, GPIO.HIGH)
GPIO.add_event_detect(btnPin,GPIO.RISING,callback=button_callback)
```

Listing 3: Sterowanie diodą LED za pomocą przycisku. Najważniejsze fragmenty.

4 Pomiar temperatury

5 Pomiar odległości

6 Podsumowanie

Literatura

[1] RaspberryPi: https://pl.wikipedia.org/wiki/Raspberry_Pi

[2] ARM: https://pl.wikipedia.org/wiki/Architektura_ARM

[3] RaspberryPi - wygląd: https://botland.com.pl/59269-thickbox_default/raspberry-pi-4-model-b-wifi-dualband-bluetooth-4gb-ram-15ghz.jpg

[4] Pinout: http://pinout.xyz

[5] Raspbian: https://www.raspbian.org/

[6] Windows 10 IoT: https://docs.microsoft.com/pl-pl/windows/iot-core/

[7] Power pins: https://elinux.org/RPi_Low-level_peripherals#Power_pins