Urządzenia Peryferyjne

Sprawozdanie

RaspberryPi

Damian Koper, 241292

Wiktor Pieklik, 241282

14 stycznia 2020

Spis treści

Spis rysunków

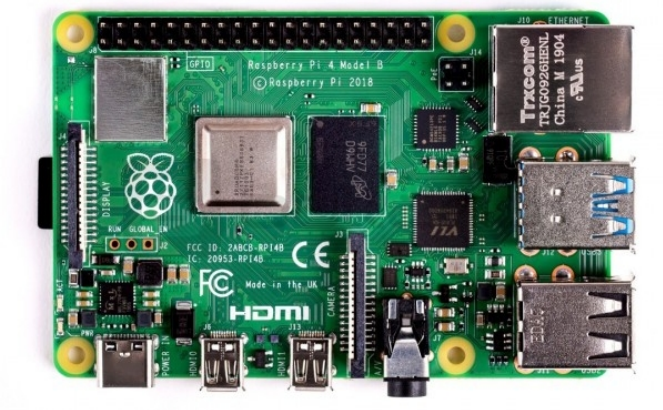
Spis listingów

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.



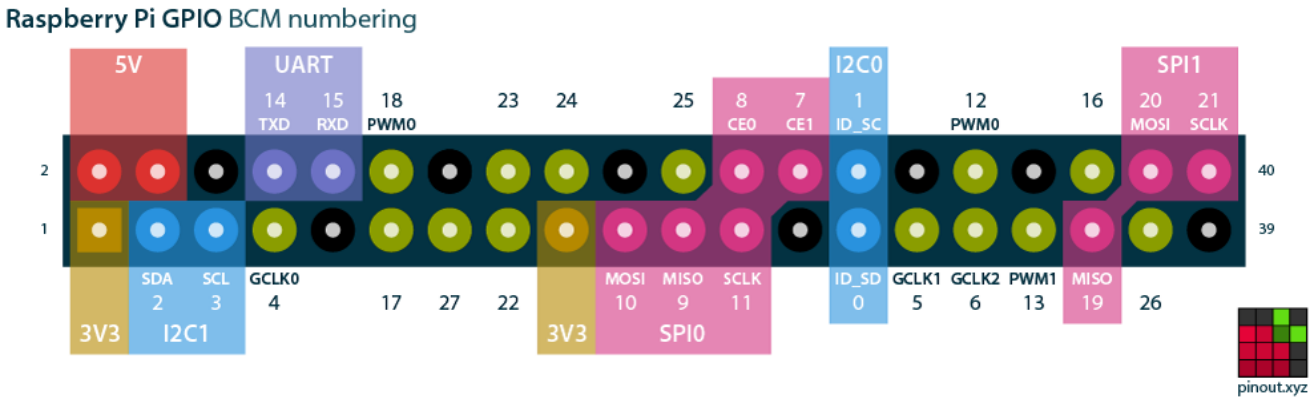
Rysunek 1: RaspberryPi w wersji 4B[3].

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.



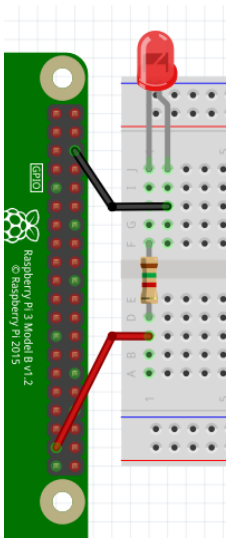
Rysunek 2: Piny RaspberryPi w wersji 4B[4].

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 3. 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
```

```
while true; do
  echo 0 > /sys/class/gpio/gpio${PIN}/value
  echo "OFF"
  sleep 1s
  echo 1 > /sys/class/gpio/gpio${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.

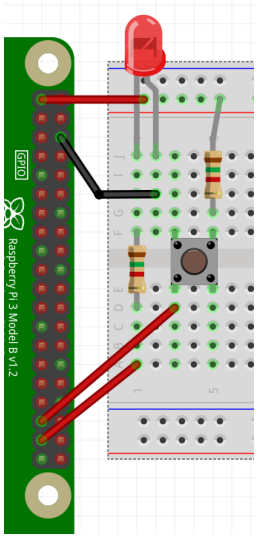
```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM);
ledPin = 26;
GPIO.setup(ledPin, GPIO.OUT, initial=GPIO.HIGH);
while True:
  print ("OFF");
  GPIO.output(ledPin, GPIO.LOW);
  time.sleep(1);
  print ("ON");
  GPIO.output(ledPin, GPIO.HIGH);
  time.sleep(1);
```

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

Sterowanie diodą za pomocą przycisku wymagało dodania układu przycisku wraz z rezystorem, który ogranicza przepływ prądu. Jeden koniec podłączono pod pin zasilający 3.3V, a drugi do pinu GPIO, za pomocą którego odczytywany był stan naciśnięcia przycisku. Schemat podłączenia pokazany został na rysunku 4.



Rysunek 4: Podłączenie diody LED i przycisku.

Zdarzenie naciśnięcia przycisku obsługiwane było za pomocą przerwania. Było ono wywoływane przy rosnącym zboczu sygnału. Za konfigurację tego zachowania odpowiada metoda `GPIO.add_event_detect` wywołana w linii 17 listingu 3, która w trzecim argumencie przyjmuje funkcję obsługi zdarzenia. Stan diody przechowywany był w zmiennej typu `boolean` i zmieniany był na przeciwny przy każdym naciśnięciu przycisku.

```
1  GPIO.setup(ledPin, GPIO.OUT, initial=GPIO.HIGH)
2  GPIO.setup(btnPin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
3  ledOn = True
4  def button_callback(channel):
5    global ledOn
6    time.sleep(0.015)
7    btnInput = GPIO.input(btnPin)
8    if btnInput is GPIO.HIGH:
9      if ledOn is True:
10       ledOn = False
11       print ("OFF")
```

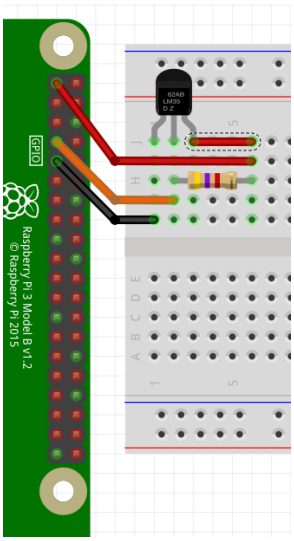
```
12        GPIO.output(ledPin, GPIO.LOW)
13      else:
14        ledOn = True
15        print ("ON")
16        GPIO.output(ledPin, GPIO.HIGH)
17  GPIO.add_event_detect(btnPin,GPIO.RISING,callback=button_callback)
```

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

Naciśnięcie przycisku nie powoduje od razu idealnego zwarcia. Występują wtedy zakłócenia powstawania zbocza rosnącego, które mogą powodować wielokrotne wykrycie zdarzenia tego samego naciśnięcia przycisku. W celu wyeliminowania tego zjawiska w linii 6 listingu 3 następuje oczekiwania na ustabilizowanie się sygnału przed jego odczytem wynoszące 15 milisekund.

4 Pomiar temperatury

Pomiar temperatury odbywał się z wykorzystaniem termometru DS18B20[8] komunikującego się z RaspberryPi poprzez protokół 1-Wire. Jest to protokół pozwalający na komunikację tylko z wykorzystaniem, oprócz złącza masy, jednego przewodu wykorzystując zasilanie pasożytnicze. W wypadku podłączonego układu napięcie zasilające dostarczane jest osobnym złączem V_{dd}.



Rysunek 5: Podłączenie termometru.

Obsługę protokołu 1-Wire należy włączyć w opcjach konfiguracyjnych systemu Raspbian, a przewód transmisji danych musi być podłączony do pinu GPIO nr 4. Każde urządzenie identyfikowane jest 64-bitowym numerem, dzięki czemu istnieje możliwość podłączenia pod jedną magistralę wielu urządzeń. Aby odczytać temperaturę należy wyświetlić zawartość pliku `/sys/bus/w1/devices` ↪ `/XXX/w1_slave`, gdzie `XXX` to numer urządzenia. Plik ten jest automatycznie aktualizowany przez system.

Aby odczytać temperaturę w języku Python wykorzystano bibliotekę `w1thermsensor`, która pozwala stworzyć obiekt typu `W1ThermSensor`, który metodą `get_temperature` pozwala odczytać temperaturę w stopniach Celsjusza. Odczyt przeprowadzany jest co sekundę. Cały kod za to odpowiedzialny widoczny jest na listingu 4

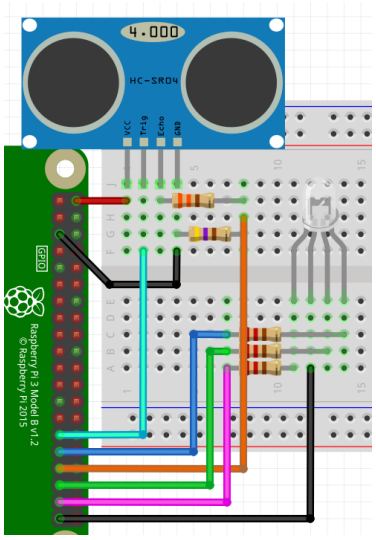
```
#!/usr/bin/python3
import time
from w1thermsensor import W1ThermSensor
sensor = W1ThermSensor()

while True:
    temperature = sensor.get_temperature()
    print("The temperature is %s celsius" % temperature)
    time.sleep(1)
```

Listing 4: Odczytywanie temperatury.

5 Pomiar odległości i dioda RGB

Na rysunku 6 pokazano połączenie sensora HC-SR04[9]. Zasilany jest on napięciem 5V i oprócz pinów V_{CC} i GND ma on jeszcze wyprowadzenia Trigger i Echo. Pierwsze z nich odpowiada za rozpoczęcie pomiaru, a drugie informuje o jego zakończeniu. Pomiar zakończony jest wtedy, kiedy wysłana i odbita fala dźwiękowa wyemitowana przez sensor do niego wróci. Znając prędkość dźwięku w powietrzu i moment rozpoczęcia pomiaru możemy na tej podstawie wyliczyć odległość.



Rysunek 6: Podłączenie sensora HC-SR04 i diody RGB.

Na rysunku 6 widać również podłączenie diody RGB LED, gdzie każdym kolorem steruje osobny pin GPIO.

```
def distance():
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    TimeElapsed = StopTime - StartTime
    distance = (TimeElapsed * 34300) / 2
    return distance
```

Listing 5: Odczytywanie odległości.

Pokazana na listingu 5 metoda `distange` włącza pomiar wysyłając impuls o długości $10\mu s$. Następnie czeka na stan wysoki na pinie Echo, który informuje o dotarciu odbitej fali. Na podstawie czasów tych wydarzeń obliczana jest odległość wg wzoru 1.

$$d = ((t_k - t_s) \cdot 34300)/2 \tag{1}$$

Gdzie:

d : szukana odległość

t_k : czas końca pomiaru

t_s : czas początku pomiaru

$34300[\frac{cm}{s}]$: prędkość dźwięku w powietrzu wyrażona w centymetrach

Wynik dzielony jest na dwa, ponieważ otrzymany czas zawiera drogę fali do obiektu i od niego po odbiciu.

Na podstawie odczytanej odległości odbywało się sterowanie kolorem świecenia diody RGB. Co sekundę wykonywany był pomiar i w zależności od odległości dioda świeciła na czerwono lub zielono. Kod odpowiedzialny za to zachowanie znajduje się w listingu 6.

```
while True:
    dist = distance()
    if dist < 20:
        GPIO.output(rpin, GPIO.HIGH)
        GPIO.output(gpin, GPIO.LOW)
    else:
        GPIO.output(rpin, GPIO.LOW)
        GPIO.output(gpin, GPIO.HIGH)
    print ("Measured Distance = %.1f cm" % dist)
    time.sleep(1)
```

Listing 6: Odczytywanie odległości.

6 Podsumowanie

Komputer RaspberryPi ze względu na swoją cenę, mały rozmiar i rozbudowaną społeczność użytkowników stanowi doskonałe źródło edukacyjne, pozwalające poznać niskopoziomowe mechanizmy,

protokoły i interfejsy. Pracując z w pełni funkcjonalnym systemem operacyjnym RaspberryPi nie ogranicza możliwości użycia jej również w wysokopoziomowych projektach.

Model RaspberryPi w wersji 4B ma jednak wadę, którą jest nieodpowiedni odpływ ciepła z procesora. Skutkowało to przegrzewaniem się układu, który w celu ochłodzenia obniżał swoje taktowanie. Działo się to przy bardziej wymagających obliczeniowo zadaniach, którymi okazały się otwarcie kilku stron internetowych.

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

[8] DS18B20: https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf

[9] HC-SR04: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf