Wykorzystanie linii GPIO Raspberry Pi 3

Angelika Tefelska Dariusz Tefelski

2022-01-04

Celem ćwiczenia jest zapoznanie z możliwościami programistycznymi obsługi wejścia-wyjścia ogólnego przeznaczenia (GPIO) Rasbpberry PI.

Uwagi wstępne:

- W celu minimalizacji zagrożenia uszkodzenia Raspberry Pi, wszelkie podłączenia należy wykonywać po wyłączeniu zasilania.
- Potrzebne biblioteki należy pobrać korzystając z narzędzia: *apt* na przykład:

```
sudo apt install wiringpi
```

- Moduł do obsługi 1-wire może powodować problemy z kontrolą pinu *GPIO4* (BCM4), ponieważ domyślnie ten pin jest wykorzystywany do transmisji 1-wire. Aby uniknąć problemów należy wstępnie wyładować moduł jądra poleceniem:

```
sudo rmmod w1_gpio
```

Możliwe jest także przemapowanie pinu używanego przez wpisanie do pliku: */boot/config.txt*:

```
dtoverlay=w1-gpio,gpiopin=x
```

gdzie x - oznacza numer pinu, który ma służyć jako 1-wire.

WiringPi

Biblioteka, która ułatwia programowanie portów GPIO Raspberry PI.

Polecenie gpio:

Opis parametrów dostępny na stronie http://wiringpi.com/the-gpio-utility/ Można zapoznać się także wydając polecenie *man gpio*

```
gpio readall
gpio -g mode 17 out
```

Opcja **-g** oznacza, że posługujemy się numeracją pinów BCM_GPIO, tzn. odpowiadające wyprowadzeniom układu SoC.

Zadanie dotyczące biblioteki wiringpi

Na wykładzie przedstawiono podstawy stosowania biblioteki *wiringPi*. Wykorzystując przykład skryptu bash z wykładu dotyczący obsługi diody LED oraz przycisku, przygotować skrypt, który będzie cyklicznie sterował światłami ulicznymi.

Światła uliczne mają następujące stany:

1. Światło czerwone - zabrania wjazdu za sygnalizator (czas trwania 3s)
2. Światło czerwone + żółte - zabrania przejazdu za sygnalizator, informuje o zmianie na zielone, (czas trwania 1s)

3. Światło zielone - zezwala na wjazd za sygnalizator (czas trwania 3s)
4. Światło żółte - zabrania wjazdu za sygnalizator, informuje o zmianie na czerwone (czas trwania 1s)

Wykorzystać:

- piny: GPIO16 (36), GPIO20 (38), GPIO21 (40)
- 3 x rezystory 330 Ω
- Diody LED: czerwona, żółta, zielona

Wykorzystanie modulacji PWM

Do sprawdzenia działania modulacji PWM możemy wykorzystać serwo modelarskie. Serwo wymaga aby częstotliwość fali wynosiła 50 Hz (okres 20 ms). Natomiast czas występowania stanu wysokiego mieści się najczęściej w zakresie od 600 μs do 2400 μs. Serwo należy podłączyć do Raspberry PI następująco:

Wyprowadzenie Serwomechanizmu	Pin GPIO Raspberry PI (nr pinu na złączu)
Przewód czarny (GND)	GND (pin 14)
Przewód czerwony (+5V)	+5V (pin 2)
Przewód biały (PWM)	GPIO18 (pin nr 12)

Przetestować z wykorzystaniem gpio:

```
gpio -g mode 18 pwm
gpio -g pwmr 2000
gpio -g pwmc 192
gpio -g pwm-ms

gpio -g pwm 18 150
gpio -g pwm 18 220
gpio -g pwm 18 50
```

Hardware'owy PWM dostępny jest na 2 kanałach PWM0 i PWM1. PWM0 to piny BCM18 lub BCM12, a PWM1 to piny BCM13 lub BCM19. Dostępne informacje na stronie: https://pinout.xyz

Częstotliwość podstawowa zegara obsługującego pwm w Raspberry Pi wynosi 19.2 MHz. Parametr pwmc (pwmClock, w C - funkcja pwmSetClock) określa dzielnik tej częstotliwości. Podając 192, otrzymujemy czas pojedynczego zliczenia wynoszący 0.01 ms. Parametr pwmc może przyjmować wartości od 2 do 4095. Dobierając parametr pwmr (range - zakres, w C - funkcja pwmSetRange) określamy okres przebiegu pwm (długość licznika a zarazem rozdzielczość licznika). Maksymalna wartość parametru pwmr to 4096. Wyznaczając parametr pwmr 2000, okres zliczeń licznika będzie wynosił 20 ms, czyli częstotliwość 50 Hz, co jest właściwą wartością dla serwomechanizmów. Położenie środkowe serwomechanizmu odpowiada szerokości impulsu 1,5 ms. Dlatego wysyłana jest wartość 150.

Program w C wykorzystujący **SOFT-PWM** *servo.c*:

```c
#include <wiringPi.h>
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <softPwm.h>

int main(int argc, char* argv[]){
  int pos;
  if(argc != 2){
    printf("Usage: servo <value 5-20>\n");
    exit(1);
  }
  pos = atoi(argv[1]);
    if ( wiringPiSetupGpio() == -1 ){
```

```
        printf("Cannot get GPIO!\n");
        exit(1);
    }
    pinMode(18, OUTPUT);
    digitalWrite(18, LOW);
    softPwmCreate(18,0,200);
    softPwmWrite(18,pos);
    delay(1000);
    return 0;
}
```

Do kompilacji warto przygotować plik *Makefile* o zawartości:

```
all:
      gcc -o servo servo.c -lwiringPi
```

Wykonać budowanie:

```
make
```

i uruchomić badając wartości w zakresie <5 - 20>.

```
./servo 10
```

Program w C wykorzystujący hardware'owy PWM.

```
#include <stdio.h>
#include <stdint.h>
#include <wiringPi.h>
#include <unistd.h>

int main(int argc, char* argv[]){
    int pos;
    if(argc !=2){
        printf("Usage: servo <value 50-230>\n");
        exit(1);
        }
    pos = atoi(argv[1]);

    if( wiringPiSetupGpio() == -1 ){
        printf("Cannot get GPIO!\n");
        exit(1);
    }

    if( geteuid() != 0 ){
        printf("You must run as root (or sudo)\n);
        exit(1);
    }

    pinMode(18, PWM_OUTPUT);
    pwmSetMode(PWM_MODE_MS);
    pwmSetRange(2000);
    pwmSetClock(192);
    pwmWrite(18, pos);
    return 0;
}
```

UWAGA! Uruchomić skompilowany program poprzez **sudo**, inaczej całe Raspberry PI ulegnie zawieszeniu, po którym tylko wyłączenie i ponowne włączenie zasilania pomoże. Związane jest to z dostępem do rejestrów ustawiających częstotliwość zegara PWM.

Biblioteka pigpio

Zainstalować przez:

```
sudo apt install pigpiod pigpio-tools
```

Oprócz daemona, jako zależność zainstaluje się właściwa biblioteka libpigpio1.

Biblioteka wymaga uruchomionego daemona - pigpiod, ale jest lepiej przystosowana np. do obsługi software-owego PWM.

Przed wykonaniem programu, musi być uruchomiony daemon przez polecenie:

```
sudo pigpiod
```

Ewentualnie można umieścić to wywołanie w skrypcie startowym np. /etc/rc.local aby uruchamiał się automatycznie w trakcie uruchamiania systemu. Ale na potrzeby naszych testowych zajęć nie jest to potrzebne.

Komenda pigs

W bibliotece pigpio istnieje także polecenie, którym można kontrolować GPIO Raspberry Pi z linii poleceń - *pigs*.

Sprawdzenie działania software-owego PWM można wykonać poleceniem:

```
pigs s 18 1300
```

Gdzie wartości dla serwomechanizmu zamiast 1300 (μs), można wpisać w zakresie od około 500 do 2200 μs.

Wyłączenie daemona pigpiod

```
sudo pkill pigpiod
```

Zadanie - Gra Reakcyjna (Python i biblioteka gpiozero)

Projekt umożliwia zbudowanie prostej reakcyjnej gry dla 2 graczy. Gra zawiera pułapki, podlicza wynik i wskazuje zwycięzcę.

Potrzebne:

- Raspberry Pi
- Płytka prototypowa
- 2 x dioda LED czerwona
- 2 x przycisk
- 1 x dioda LED RGB
- 5 x 330Ω rezystory
- 12 x przewód do połączeń wewnętrznych na płytce prototypowej

Instrukcja wykonania

1. Upewnij się, że Raspberry PI jest wyłączone z zasilania, wykonaj obwód na płytce prototypowej i podłącz do Raspberry PI.

UWAGA 1 !!! Proponuję zamienić pin BCM14 - dioda czerwona, na pin BCM23 - odpowiednio trzeba będzie zmienić w pobranym programie!!!

UWAGA 2 !!! Jeśli dysponujesz diodą RGB ze wspólną anodą, to tą wspólną anodę (najdłuższa nóżka) podłączamy do +3.3 V a nie do masy jak na schemacie. W programie przy tworzeniu obiektu rgb, należy w argumentach dodać: **active_high=False**.

Uwzględniając powyższe uwagi, inicjalizacja obiektu rgb powinna wyglądać następująco:

```
rgb = RGBLED(red=23, green=17, blue=27, active_high=False)
```

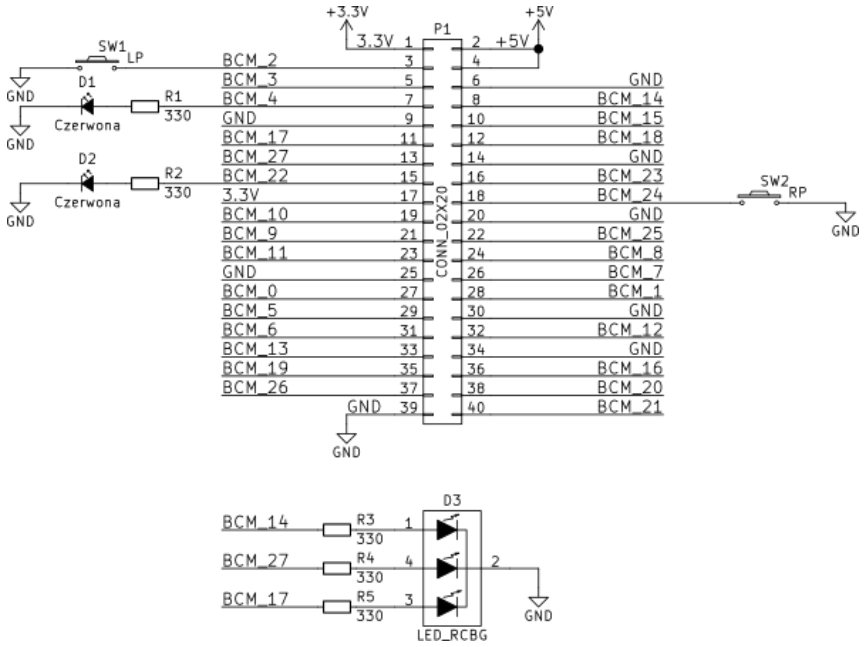

Figure 1: Schemat połączeń gry reakcyjnej

2. Włącz Raspberry Pi
3. Wyładuj kolidujący moduł jądra - patrz *uwagi* na początku instrukcji i uruchom ponownie Raspberry Pi
4. Włącz Raspberry Pi i pobierz przykładowy kod programu

```
git clone https://github.com/henrybudden/rpesk-advanced
sudo apt-get update
sudo apt-get install python-gpiozero python-colorzero python-pkg-resources
cd rpesk-advanced
sudo python 02_reaction_game.py
```

5. Gra przeznaczona jest dla 2 graczy. Polega na jak najszybszym wciśnięciu przycisku, jeśli dioda LED zaświeci się innym kolorem niż czerwony. Wciśnięcie przycisku gdy pojawi się kolor czerwony powoduje odjęcie jednego punktu od liczby punktów gracza. Diody na płytce wskazują który gracz wygrywa. Jeśli świecą się 2 jest remis.

Gra typu papier, nożyczki, kamień

Zaprojektuj podobną grę - ale dla jednego gracza i wykonaj z wykorzystaniem języka Python oraz biblioteki gpiozero oraz dostępnych elementów.

Gra ma być oparta na zasadzie “Papier - Nożyczki - Kamień” i ma rozgrywać się pomiędzy graczem a Raspberry PI.

Wykorzystać co najmniej:

- 3 przyciski
- 2 diody LED + 2 rezystory 330Ω

Generalna zasada polega na tym, że użytkownik wybiera - za pomocą przycisku: papier, nożyczki albo kamień.

Raspberry Pi losuje swój wybór.

Mamy następujące zależności:

1. Jeśli użytkownik wybrał papier:
   - Jeśli Raspberry PI wybrało papier - to mamy remis.
   - Jeśli Raspberry PI wybrało kamień - to wygrywa gracz.

    - Jeśli Raspberry PI wybrało nożyczki - to gracz przegrywa.
2. Jeśli użytkownik wybrał nożyczki:
    - Jeśli Raspberry PI wybrało papier - to wygrywa gracz.
    - Jeśli Raspberry PI wybrało kamień - to gracz przegrywa.
    - Jeśli Raspberry PI wybrało nożyczki - to mamy remis.
3. Jeśli użytkownik wybrał kamień:
    - Jeśli Raspberry PI wybrało papier - to gracz przegrywa.
    - Jeśli Raspberry PI wybrało kamień - to mamy remis.
    - Jeśli Raspberry PI wybrało nożyczki - to wygrywa gracz.

Stanowisko laboratoryjne

Podzespoły potrzebne do realizacji ćwiczenia:

- Raspberry Pi
- Serwomechanizm (mikroserwo)
- Płytka prototypowa
- diody LED czerwone
- diody LED żółte
- diody LED zielone
- przyciski
- 1 x dioda LED RGB
- Rezystory 330Ω
- przewody do połączeń wewnętrznych na płytce prototypowej

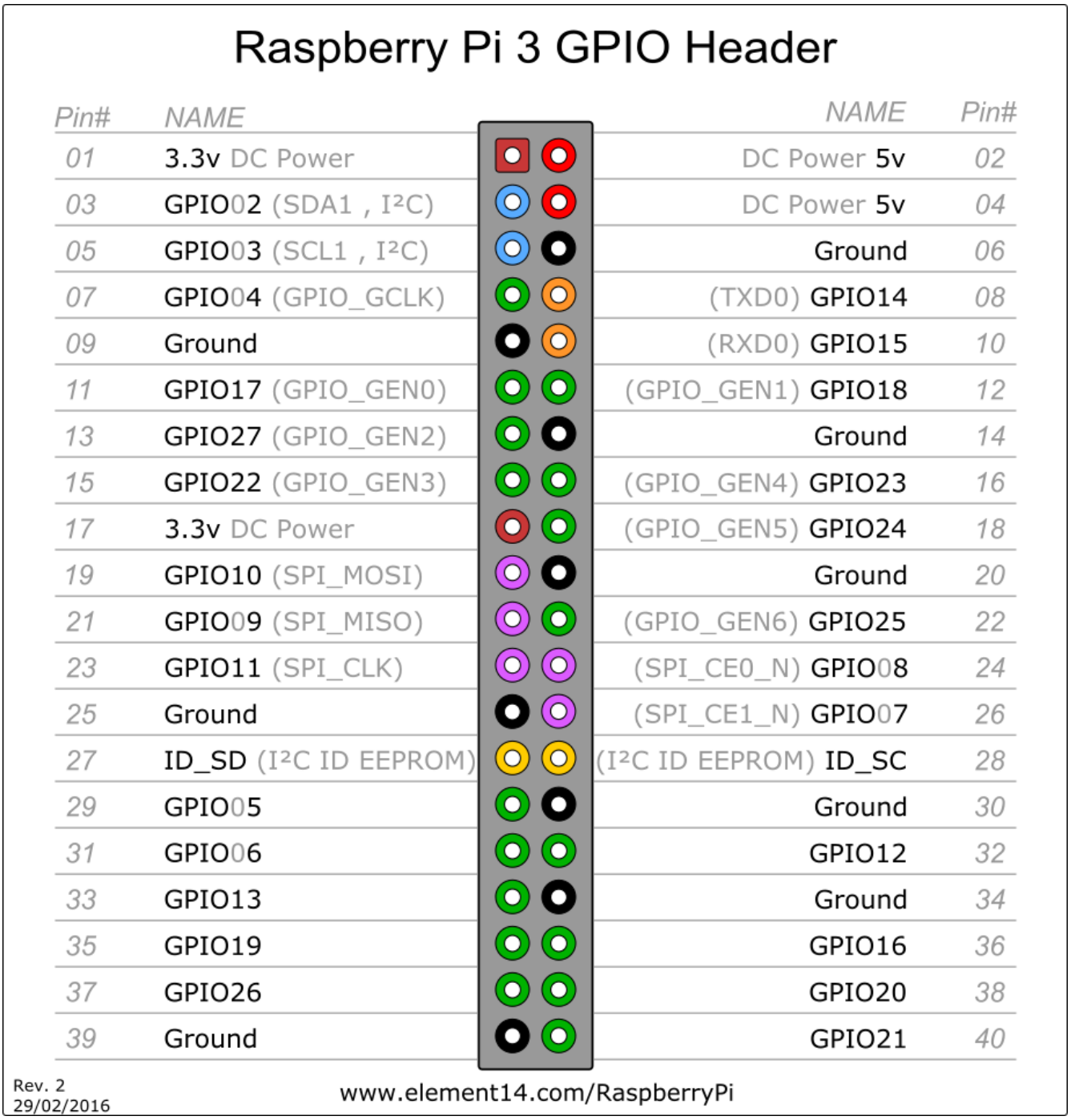

Figure 2: Złącze 40-pin Raspberry Pi.