



W największym skrócie - **Raspberry Pi to miniaturowy komputer.**

Określenie "miniaturowy" jest jak najbardziej na miejscu, ponieważ ma on wymiary zbliżone do karty kredytowej (oczywiście poza grubością).

Natomiast "komputer" oznacza, że nie jest to tylko płytką ewaluacyjną, moduł IoT lub mikrokontroler.

To cały, w pełni funkcjonalny, komputer!

Specyfikacja Raspberry Pi 3 model B+

- **Procesor:** czterordzeniowy Broadcom BCM2837B0 ARMv8 z rdzeniem ARM Cortex-A53 (1,4 GHz)
- **Pamięć RAM** o pojemności 1 GB
- **Pełna kompatybilność z: Raspberry Pi 2 model B oraz Raspberry Pi 3 model B** - układ peryferiów pozostał bez zmian
- **Złącze rozszerzeń (GPIO):** 40-pinowe
- **USB:** 4 x USB 2.0
- **Wideo:** HDMI video/audio
- **Audio:** Jack 3,5 mm
- **Połączenie z Internetem:** port Ethernet (do 300 Mbps),
Moduł Wifi 802.11 b/g/n/ac (2,4 GHz/5 GHz)
- **Bluetooth:** Bluetooth 4.2 oraz Low Energy (BLE)

Specyfikacja Raspberry Pi 3 model B+

- **Złącza na kartę:** MicroSD
- **Interfejs wyświetlacza:** DSI
- **Interfejs kamery:** CSI
- **Zasilanie:** 5V, 2.5A (przez gniazdo micro-usb)
- **Możliwość zasilania** za pomocą PoE (Power over Ethernet), wykorzystując specjalną nakładkę
- **Kompatybilny** ze wszystkimi ostatnimi wersjami ARM GNU/Linux oraz Windows 10
- **Wymiary:** 85 x 56 x 17 mm

40 (2x20) pinów
GPIO

Złącze DSI
wyświetlacza

Złącze zasilania
micro USB
(prąd do 2,5 A)

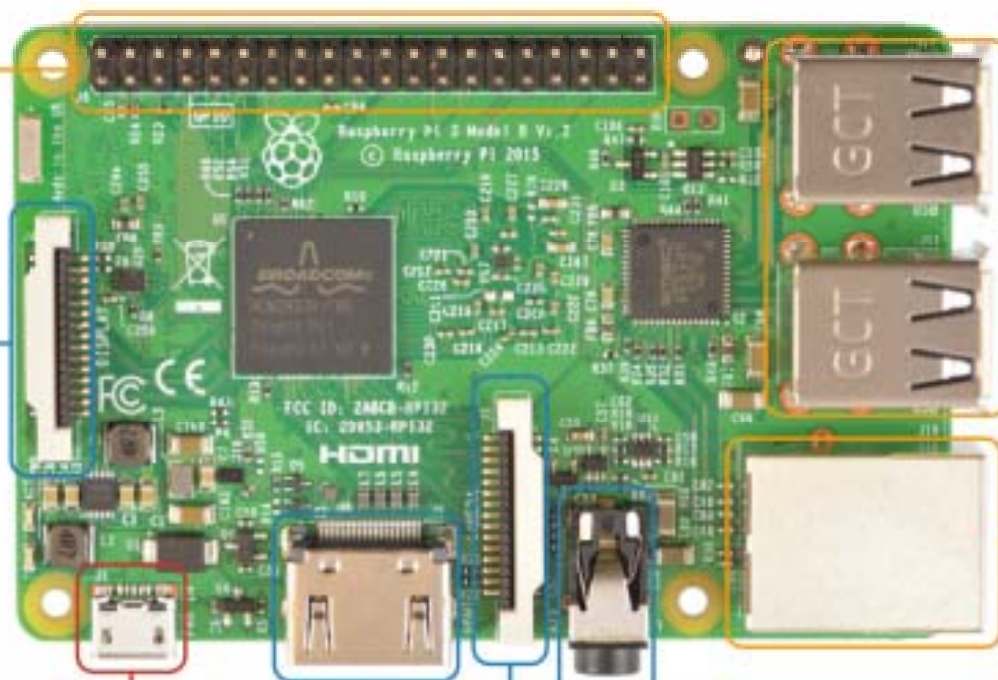
Złącze
HDMI

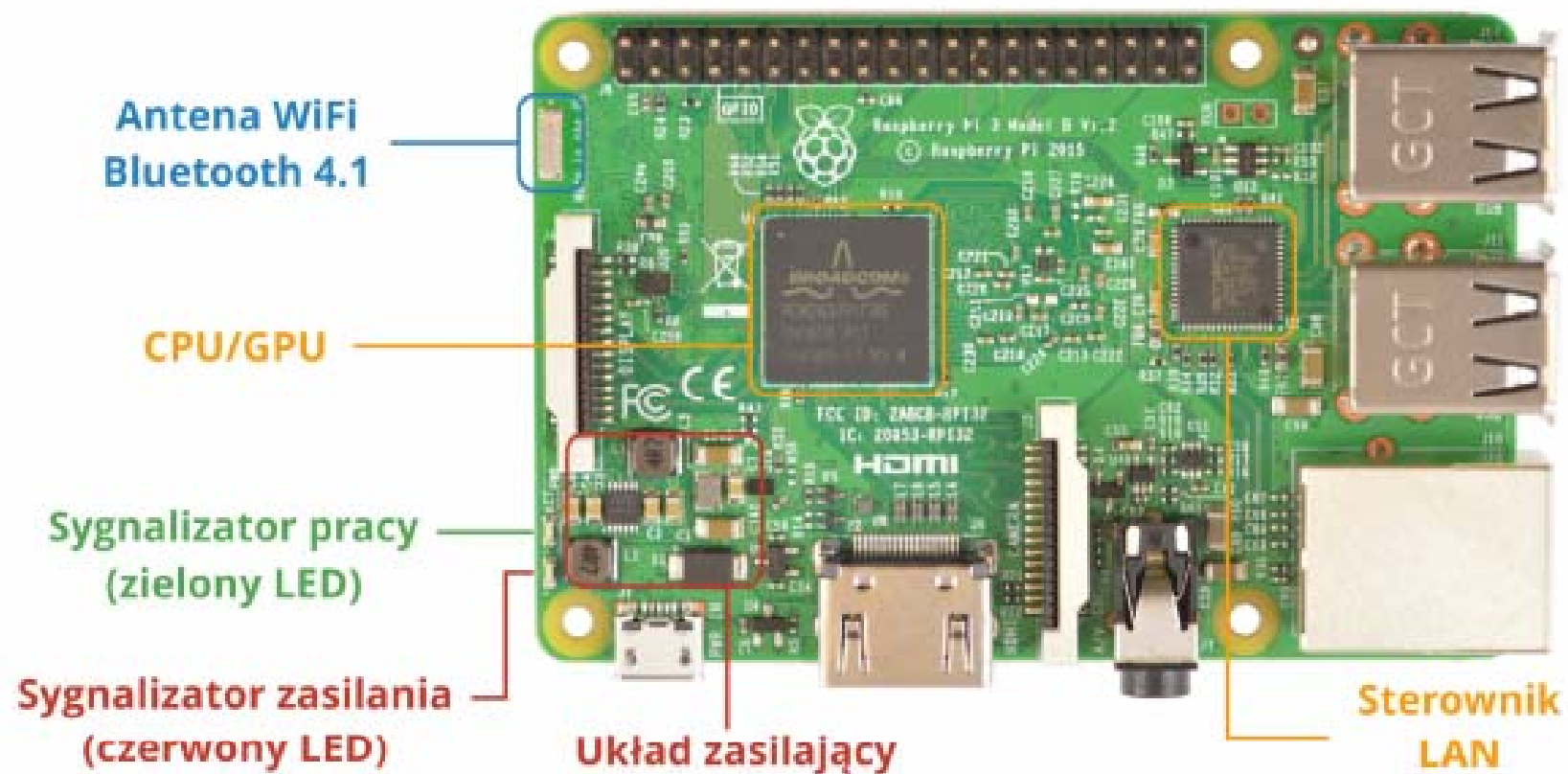
Złącze CSI
kamery

4-polowe gniazdo jack
3,5mm (audio/video)

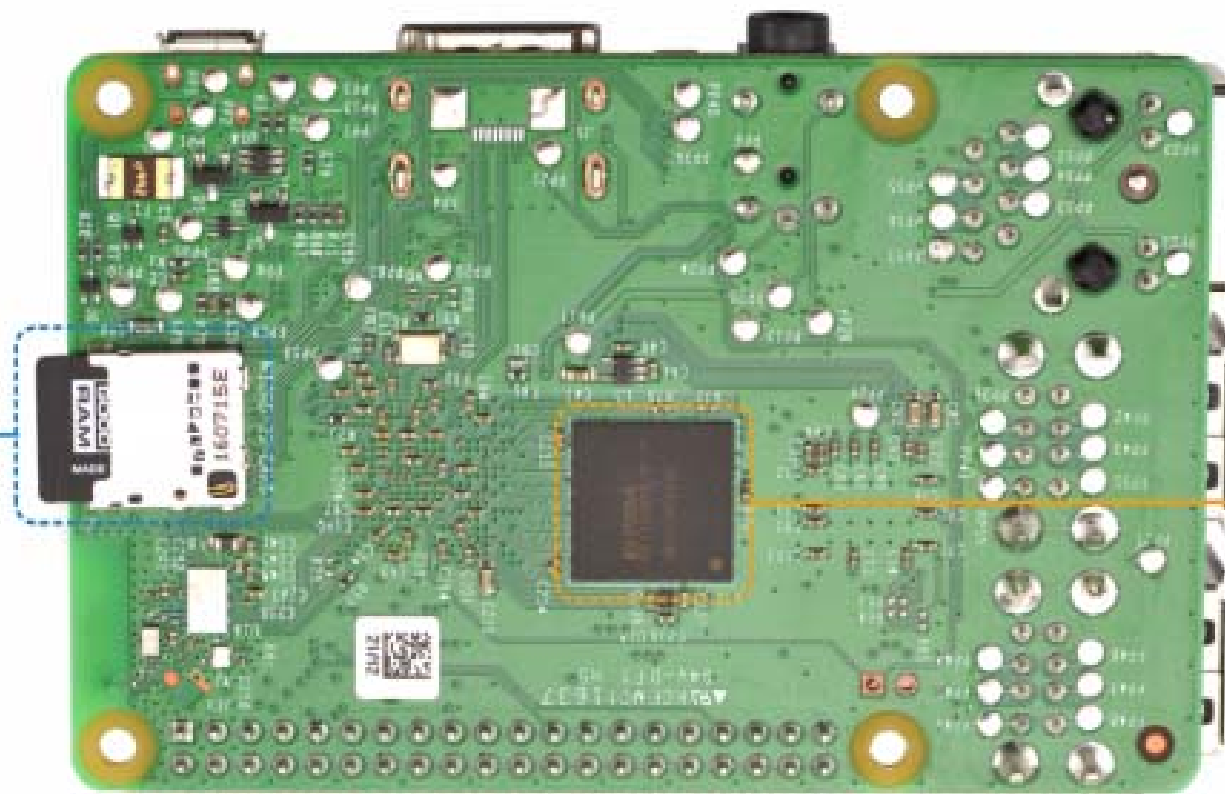
4 (2x2) gniazda
USB 2.0

Gniazdo LAN
(10/100)





Gniazdo kart
micro SD



1GB
RAM

Praca w konsoli, podstawy Linuksa

Pora na przyspieszony kurs podstaw Linuksa. Najważniejsze komendy, funkcje oraz programy, które są konieczne do pracy z Raspberry Pi.

Przed aktualizacją musimy pobrać informacje o najnowszych wersjach. Służy do tego polecenie:

sudo apt update

Kolejny krok to instalacja znalezionych aktualizacji:

sudo apt upgrade

Jak sprawdzić listę zainstalowanych programów:

apt list --installed

Odpowiednik "Menedżera zadań,,

top

Instalacja nowego "Menedżera zadań,,

sudo apt install htop

Usuwanie programów

sudo apt remove htop

sudo apt purge htop

Sprawdzenie aktualnego katalogu

pwd

Sprawdzenie zawartości katalogu

***ls** - wyświetla listę plików w lokalnym katalogu*

***ls -l** - wyświetla tzw. długą listę, z uprawnieniami dostępu do każdego pliku i katalogu*

***ls -a** - wyświetla pliki ukryte (o nazwie zaczynającej się od kropki)*

***ls -1** - lista plików w jednej kolumnie*

***ls <ścieżka>** - np. **ls /** , pokazuje pliki w podanym folderze*

Miejsca które warto "zobaczyć":

***ls /** - katalog główny*

***ls /dev** - zawiera sterowniki urządzeń, przykładowo /dev/ttyAMA0 to port szeregowy do którego możemy podłączyć naszą przejściówkę UART-USB*

***ls /home** - tutaj znajdują się katalogi użytkowników (każdy ma własny podkatalog)*

***ls /root** - katalog administratora, nie mamy do niego dostępu (ale warto spróbować)*

***ls /proc** - wirtualny katalog z informacjami o systemie*

***ls /sys** - nowszy odpowiednik /proc*

***ls /boot** - tutaj znajdziemy pliki niezbędne do startu systemu oraz pliki konfiguracyjne, są one widoczne również dla Windowsa (gdy włożymy kartę SD do PC)*

Uzupełnienie nazw, zapamiętywanie poleceń

Używając konsoli możemy skorzystać z uzupełniania nazw.

Piszemy początek nazwy, **naciskamy tabulator** i konsola uzupełni nazwę (o ile będzie to jednoznaczna i poprawna nazwa).

Możemy napisać **ls /pr** i nacisnąć **tab** - shell sam uzupełni nazwę do **ls /proc/**

Historia wydawanych poleceń jest zapamiętywana.

Do poprzednich można wracać używając kursorów.

Strzałka w górę to poprzednie polecenie (w dół następne jak już się cofniemy).

Dodatkowo strzałki w lewo i w prawo pozwalają na przesuwanie kursora i modyfikację poleceń. Naciskając Enter wykonamy polecenie ponownie.

Listę wszystkich wydanych wcześniej poleceń zobaczymy wpisując polecenie:

history

Aktywacja SSH - zgoda na zdalny dostęp do RPi

Domyślnie zdalna praca jest na Raspberry Pi wyłączona dla bezpieczeństwa.

Początkowa

nazwa użytkownika (pi)

oraz

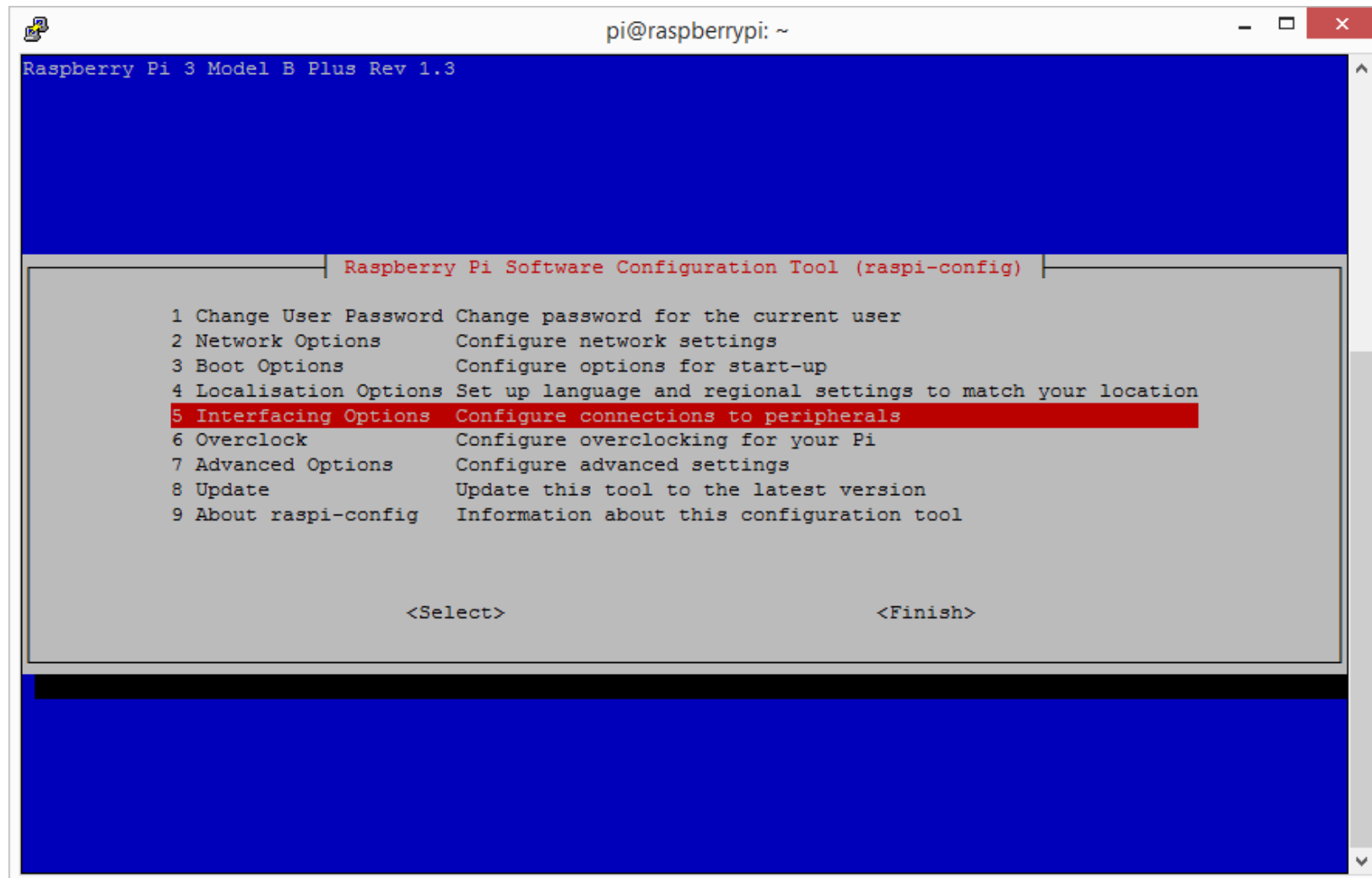
hasło (raspberry)

są ogólnie znane, więc taka konfiguracja byłaby narażona na ataki hakerów.

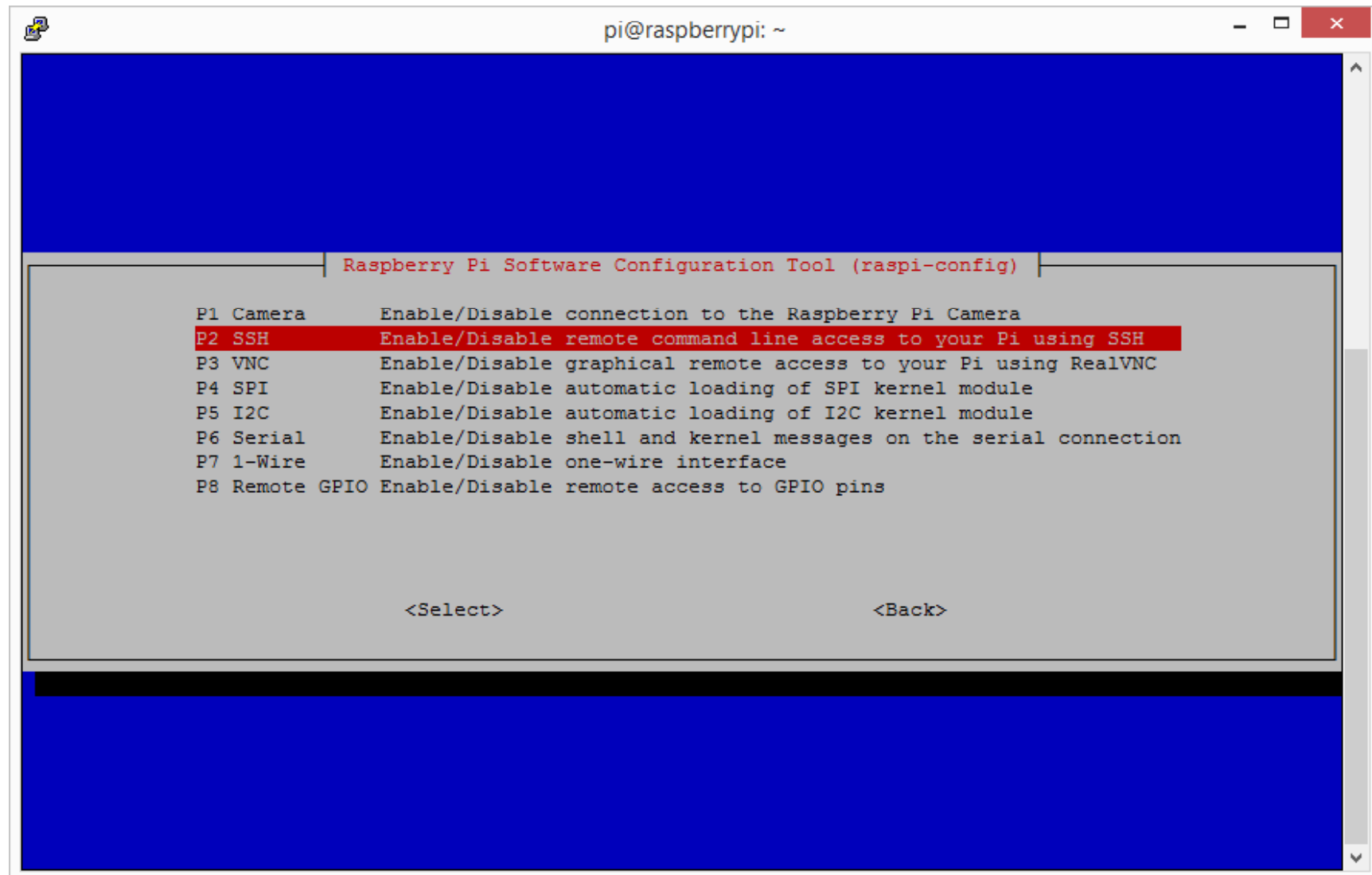
Teraz włączymy zdalny dostęp, połączymy się z malinką i szybko ustawimy nowe hasło. Dzięki temu nasza płytka będzie już bezpieczna.

raspi-config, czyli podstawowa konfiguracja

Spśród wyświetlonych pozycji wybieramy 5- Interfacing Options:

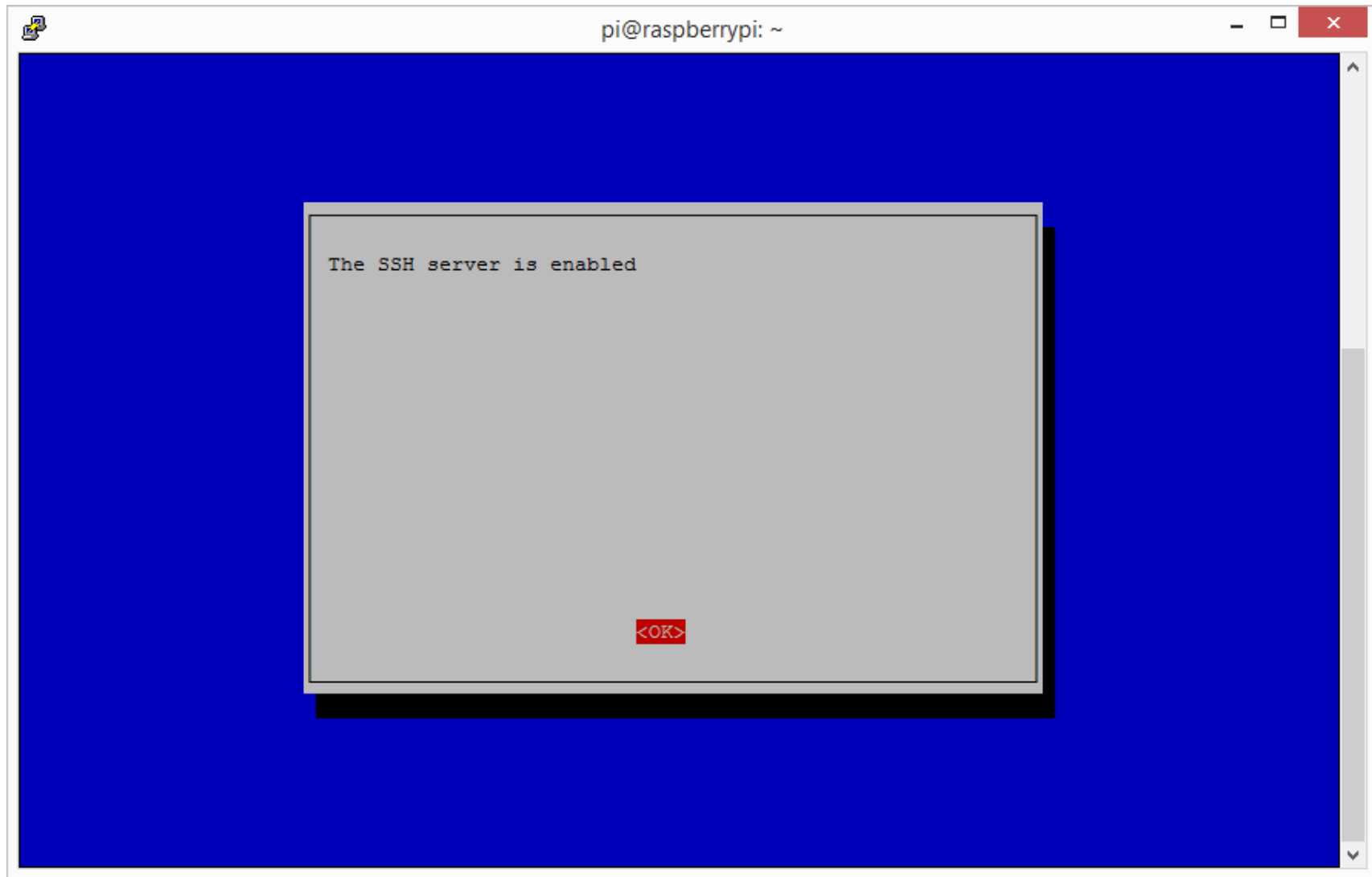


Następnie przechodzimy do P2 SSH - Enable/Disable remote command line access to your Pi using SSH.



Wybieram **TAK** i wciskamy Enter. Po chwili nasz serwer zostanie skonfigurowany.





W przypadku połączenia przewodowego i wykorzystaniu routera z serwerem DHCP, przy domyślnych ustawieniach sieciowych Raspbian'a, już w tym momencie moglibyśmy podłączyć się zdalnie, korzystając z klienta SSH. Oczywiście przydzielany automatycznie adres IP nie jest najlepszym rozwiązaniem.

Jak ustawić statyczne IP

Ustawiając stały adres IP w plikach konfiguracyjnych Raspberry Pi.

W tym celu należy zmienić plik przy użyciu, np. edytora tekstowego ***nano*** wpisując w terminalu komendę:

sudo nano /etc/dhcpd.conf

Domyślnie IP pobierane są z serwera DHCP, aby ustawić statyczne IP dla połączenia poprzez Ethernet oraz WiFi należy dodać następujący kod na końcu pliku::

```
interface eth0  
static ip_address=192.168.66.xxx/24  
static routers=192.168.66.254  
static domain_name_servers=8.8.8.8
```

```
interface wlan0  
static ip_address=192.168.66.xxx/24  
static routers=192.168.66.254  
static domain_name_servers=8.8.8.8
```

xxx → RPI1=110; RPI2=120; RPI3=130; RPI4=140; RPI5=150; RPI6=160

Następnie należy zapisać plik i opuścić edytor tekstowy (Ctrl+X) potwierdzając nadpisanie pliku literą "Y".

Po restarcie będziemy mieli ustawione statyczne IP dla połączenia kablowego.

Połączenie przez sieć bezprzewodową

Za pomocą edytora tekstu otwieramy

plik **/etc/wpa_supplicant/wpa_supplicant.conf**

i wprowadzamy dane dostępu do naszej sieci bezprzewodowej.

Dane do WiFi należy podać w następujący sposób:

country=PL

update_config=1

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev

network={

ssid="Nazwa sieci"

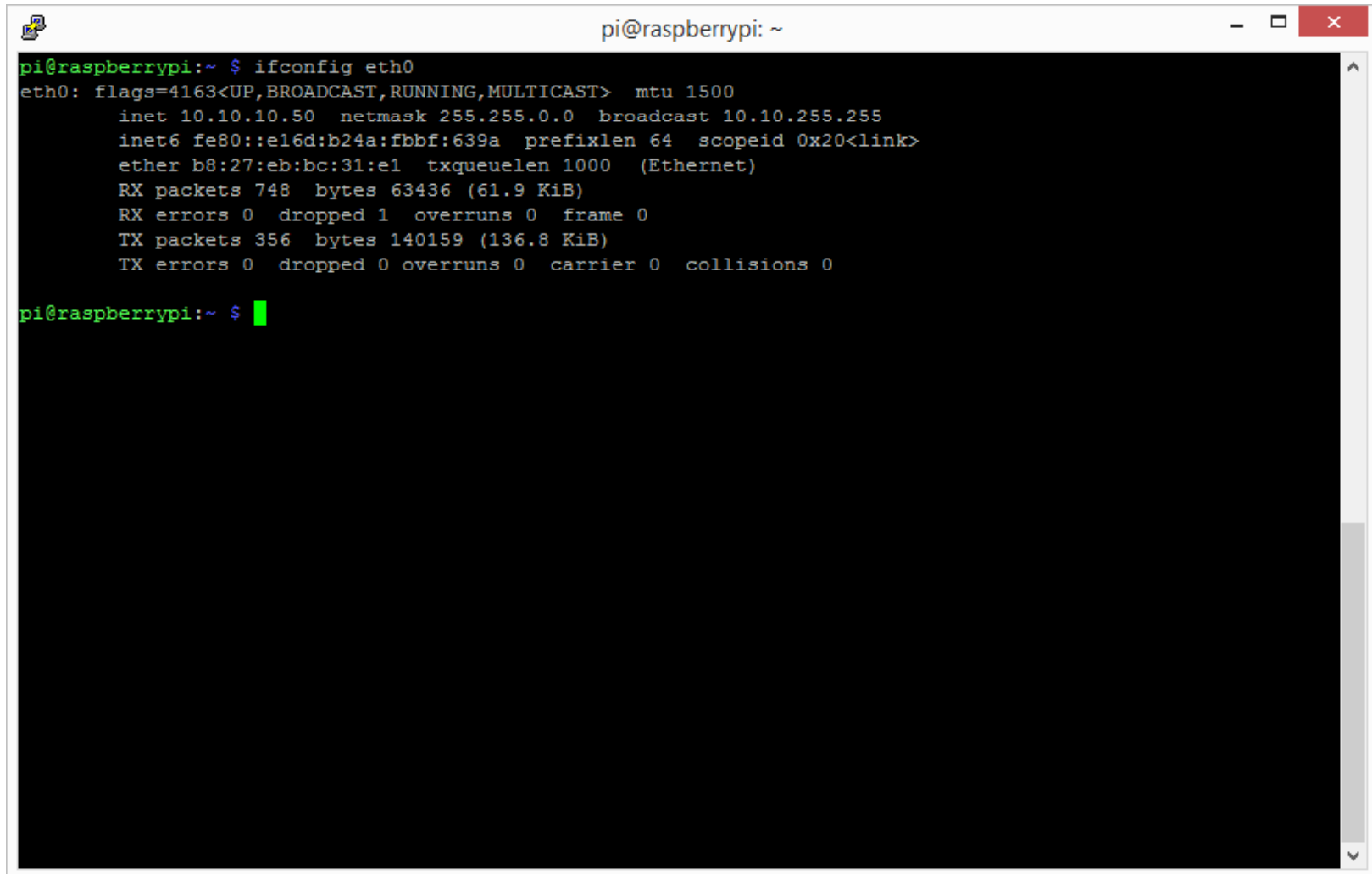
psk="hasło"

key_mgmt=WPA-PSK }

Za pomocą polecenia

ifconfig eth0

sprawdzamy ustawienia karty sieciowej.

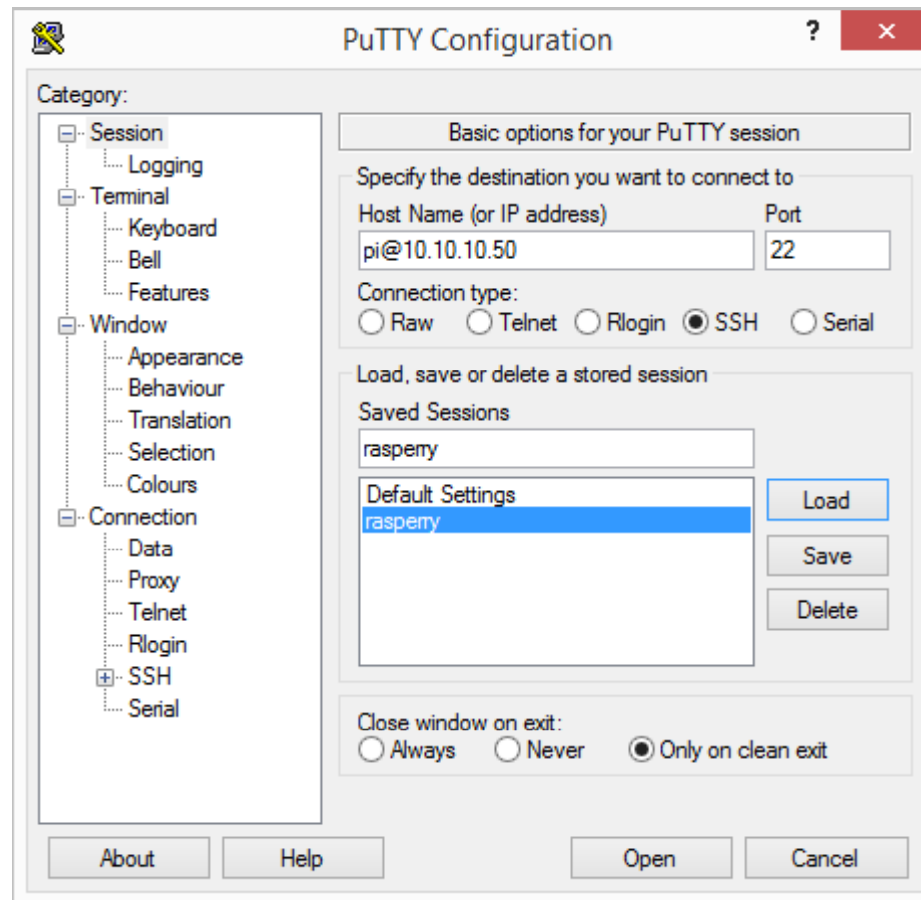
A terminal window titled "pi@raspberrypi: ~" with standard window controls. The terminal shows the command "ifconfig eth0" and its output. The output displays various network parameters for the eth0 interface, including flags, IP address, netmask, broadcast address, MAC address, and statistics for RX and TX packets and errors. The prompt "pi@raspberrypi:~ \$" is visible at the bottom with a green cursor.


```
pi@raspberrypi:~ $ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.10.10.50  netmask 255.255.0.0  broadcast 10.10.255.255
    inet6 fe80::e16d:b24a:fbbf:639a  prefixlen 64  scopeid 0x20<link>
    ether b8:27:eb:bc:31:e1  txqueuelen 1000  (Ethernet)
    RX packets 748  bytes 63436 (61.9 KiB)
    RX errors 0  dropped 1  overruns 0  frame 0
    TX packets 356  bytes 140159 (136.8 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:~ $
```

W tym momencie możemy korzystać z naszego Pi zdalnie.

Najlepszym klientem SSH dla systemu Windows jest PUTTY.





pi@raspberrypi: ~

```
Using username "pi".
pi@10.10.10.50's password:
Access denied
pi@10.10.10.50's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jun  5 20:39:48 2018 from 10.10.10.55

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
```


Aby zalogować się do Pi zdalnie korzystając z protokołu RDP (ang. remote desktop protocol) musimy najpierw zainstalować na nim XRDP. Po raz, być może ostatni korzystając z terminala, logujemy się do naszego serwera (jeśli nie pamiętasz jak to zrobić sprawdź poprzedni wpis) i wykonujemy następujące polecenie

sudo apt-get install xrdp

Aby uruchomić klienta w systemie Windows wystarczy z konsoli wpisać polecenie:

mstsc

Bądź wybrać aplikację 'Remote Desktop Connection' z menu Start

Uniwersalne porty wejścia-wyjścia Raspberry Pi

Podłączając cokolwiek do GPIO należy zawsze pamiętać o tym, że Raspberry Pi przystosowane jest do pracy z napięciem 3,3V. Jeśli do pinów dostarczymy wyższe napięcie to łatwo uszkodzimy malinkę.



Raspberry Pi 3 GPIO

GPIO	PIN	PIN	GPIO
DC 3,3 V	01	02	DC 5,0 V
IO 02 SDA1	03	04	DC 5,0 V
IO 03 SCL1	05	06	GND
IO 04	07	08	TXD IO 14
GND	09	10	RXD IO 15
IO 17	11	12	IO 18
IO 27	13	14	GND
IO 22	15	16	IO 23
DC 3,3 V	17	18	IO 24
IO 10 MOSI	19	20	GND
IO 09 MISO	21	22	IO 25
IO 11 CLK	23	24	CE0_N IO 08
GND	25	26	CE1_N IO 07
IO 05	27	28	ID SC
IO 06	29	30	GND
IO 13	31	32	IO 12
IO 19	33	34	GND
IO 26	35	36	IO 16
GND	37	38	IO 20
	39	40	IO 21

www.kurs.forbot.pl/rpi

GPIO Numbers

Raspberry Pi B
Rev 1 P1 GPIO Header

	Pin No.		
3.3V	1	2	5V
GPIO0	3	4	5V
GPIO1	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO21	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7

Raspberry Pi A/B
Rev 2 P1 GPIO Header

	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7

Raspberry Pi B+
B+ J8 GPIO Header

	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Key

Power +	UART
GND	SPI
I²C	GPIO

Sterujemy GPIO z linii poleceń

WiringPi - instalacja biblioteki

Osoby korzystające z nowego RPi 3B+ powinny zacząć od sprawdzenie wersji zainstalowanego *gpio*.

gpio -v

Jeśli zwrócona wersja będzie niższa od 2.46 (najczęściej 2.44) to należy wykonać poniższe polecenia. Za ich pomocą pobierzemy ze strony autora odpowiednią wersję wiringPi i ręcznie ją zaktualizujemy:

sudo apt-get purge wiringpi

hash -r

cd /tmp

git clone git://git.drogon.net/wiringPi

cd wiringPi

./build

Po instalacji biblioteki możemy zacząć używać narzędzia *gpio*

Warto dodać, że biblioteka korzysta z własnej numeralizacji pinów GPIO

Raspberry Pi GPIO Header

BCM	WiringPi	Name	Physical	Name	WiringPi	BCM
		3.3v	1	2	5v	
2	8	SDA.1	3	4	5V	
3	9	SCL.1	5	6	0v	
4	7	1-Wire	7	8	TxD	15
		0v	9	10	RxD	16
17	0	GPIO. 0	11	12	GPIO. 1	1
27	2	GPIO. 2	13	14	0v	18
22	3	GPIO. 3	15	16	GPIO. 4	4
		3.3v	17	18	GPIO. 5	5
10	12	MOSI	19	20	0v	23
9	13	MISO	21	22	GPIO. 6	6
11	14	SCLK	23	24	CE0	25
		0v	25	26	CE1	10
0	30	SDA.0	27	28	SCL.0	11
5	21	GPIO.21	29	30	0v	7
6	22	GPIO.22	31	32	GPIO.26	31
13	23	GPIO.23	33	34	0v	1
19	24	GPIO.24	35	36	GPIO.27	26
26	25	GPIO.25	37	38	GPIO.28	27
		0v	39	40	GPIO.29	28
						20
						21
BCM	WiringPi	Name	Physical	Name	WiringPi	BCM

Jeśli przyzwyczailiśmy się do standardowej numeralizacji pinów (BCM) możemy użyć flagi `-g` dzięki czemu zamiast podawać numeralizację biblioteki możemy podać numeralizację BCM.

Przykład:

gpio write 4 1

gpio -g write 23 1

Powyższe przykłady robią to samo tyle tylko, że używamy osobnych numeralizacji pinów, pierwszą z biblioteki `wiringpi` drugą zaś ze standardowej BCM i używamy przy tym flagi `-g`

gpio readall

Polecenie to wyświetli nam stany i informacje o wszystkich pinach GPIO

```
pi@raspberrypi:~ $ gpio readall
```

Pi 3												
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM		
		3.3v			1	2		5v				
2	8	SDA.1	IN	1	3	4		5v				
3	9	SCL.1	IN	1	5	6		0v				
4	7	GPIO. 7	IN	1	7	8	0	IN	TxD	15	14	
		0v			9	10	1	IN	RxD	16	15	
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18	
27	2	GPIO. 2	IN	0	13	14		0v				
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4	23	
		3.3v			17	18	0	IN	GPIO. 5	5	24	
10	12	MOSI	IN	0	19	20		0v				
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25	
11	14	SCLK	IN	0	23	24	1	IN	CE0	10	8	
		0v			25	26	1	IN	CE1	11	7	
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1	
5	21	GPIO.21	IN	1	29	30		0v				
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12	
13	23	GPIO.23	IN	0	33	34		0v				
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27	16	
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28	20	
		0v			39	40	0	IN	GPIO.29	29	21	

```
pi@raspberrypi:~ $
```

Na początek ustawiamy pin jako wyjście wydając polecenie:

gpio -g mode 12 out

gpio -g write 12 1

gpio -g write 12 0

Oczywiście dla testu możemy spróbować użyć numeracji biblioteki WiringPi. Wracamy do tabelki z pinami (*gpio readall*) i odnajdujemy nasz pin numer 12. Zgodnie z wewnętrzną numeracją biblioteki będzie miał on numer 26. Możemy więc wydać polecenia:

gpio mode 26 out

gpio write 26 0

gpio write 26 1

Pierwszy skrypt

Na początku musimy utworzyć nowy plik z rozszerzeniem **sh**.

Przykładowo może to być **prog1.sh**, w tym celu skorzystamy z edytora nano:

```
nano prog1sh
```

```
#!/bin/sh
```

```
gpio -g mode 21 out
```

```
gpio -g write 21 1
```

Plik zapisujemy (*CTRL+X*). Następnie musimy naszemu skryptowi nadać odpowiednie uprawnienia, aby był traktowany jak "program". W tym celu korzystamy oczywiście z *chmod*:

```
sudo chmod +x prog1.sh
```

```
./prog1.sh
```

#!/bin/sh

gpio -g mode 21 out

while true

do

gpio -g write 21 1

sleep 1

gpio -g write 21 0

sleep 1

done

Sterowanie liniami GPIO w Pythonie

Do sterowania pinami potrzebujemy odpowiedniej biblioteki. Właściwie zawsze pisząc programy w Pythonie wykorzystujemy jakieś biblioteki. Trzeba przyznać, że w porównaniu z innymi językami Python pozwala na **łatwe i przyjemne używanie gotowych rozwiązań**.

Nie inaczej jest w naszym przypadku - tym co potrzebujemy jest biblioteka [RPi.GPIO](#). Sama biblioteka jest już zainstalowana (pakiet **python3-rpi.gpio**), co możemy sprawdzić poleceniem:

```
apt list --installed | grep rpi.gpio
```

Teraz wystarczy ją zaimportować.

```
import RPi.GPIO as GPIO
```

Jak pamiętamy z wcześniejszych odcinków, obsługa pinów na Raspberry obarczona jest pewnym bałaganem numeracyjnym. W Pythonie ten bałagan również istnieje. Zanim użyjemy pinów **musimy wybrać sposób numeracji**. Chcemy używać pinu GPIO21, więc piszemy:

```
GPIO.setmode(GPIO.BCM)
```

GPIO.setup(21, GPIO.OUT)

Jeśli linia miałaby być wejściem wpisalibyśmy tutaj **GPIO.IN**.

Aby ustawić na linii GPIO21 stan wysoki:

GPIO.output(21, GPIO.HIGH)

Aby ustawić na linii GPIO21 stan niski:

GPIO.output(21, GPIO.LOW)

Na zakończenie zabawy pinami powinniśmy po sobie "posprzątać", czyli przywrócić używane przez nasz program piny do ich domyślnej konfiguracji. Służy do tego polecenie:

GPIO.cleanup()


```
import RPi.GPIO as GPIO
```

```
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setwarnings(False)
```

```
GPIO.setup(21, GPIO.OUT)
```

```
GPIO.output(21, GPIO.HIGH)
```

Na końcu skryptu nie ma wywołania ***GPIO.cleanup()*** ponieważ sprzątanie wyłączyłoby diodę. To w sumie racjonalne rozwiązanie, ale w tym przypadku byłoby niewygodne. Niestety zostawienie kodu bez sprzątania przy kolejnym użyciu pinów generuje ostrzeżenia (ang. warnings). Więc, aby nie widzieć tych ostrzeżeń wyłączamy takie komunikaty wywołaniem ***GPIO.setwarnings(False)***

```
import RPi.GPIO as GPIO  
from time import *
```

```
GPIO.setmode(GPIO.BCM)  
GPIO.setwarnings(False)  
GPIO.setup(21, GPIO.OUT)
```

```
while True:
```

```
    GPIO.output(21, GPIO.HIGH)  
    sleep(1)  
    GPIO.output(21, GPIO.LOW)  
    sleep(1)
```

```
import RPi.GPIO as GPIO  
from time import *
```

```
GPIO.setmode(GPIO.BCM)  
GPIO.setwarnings(False)  
GPIO.setup(16, GPIO.OUT)  
GPIO.setup(20, GPIO.OUT)  
GPIO.setup(21, GPIO.OUT)
```

```
while True:
```

```
    GPIO.output(16, GPIO.HIGH)  
    GPIO.output(20, GPIO.LOW)  
    GPIO.output(21, GPIO.LOW)  
    sleep(1)
```

```
GPIO.output(20, GPIO.HIGH)  
sleep(1)
```

```
GPIO.output(16, GPIO.LOW)  
GPIO.output(20, GPIO.LOW)  
GPIO.output(21, GPIO.HIGH)  
sleep(1)
```

```
GPIO.output(20, GPIO.HIGH)  
GPIO.output(21, GPIO.LOW)  
sleep(1)
```

```
GPIO.output(16, GPIO.HIGH)  
GPIO.output(20, GPIO.LOW)  
sleep(1)
```

Co ciekawe Python nie używa instrukcji typu **begin/end** lub nawiasów klamrowych { } do definiowania bloków kodu. To wcięcia definiują kolejne bloki programu.

Dzięki temu programy pisane w Pythonie muszą być ładnie formatowane - inaczej nie zadziałają.

Cieężko powiedzieć jednoznacznie, czy to zaleta języka, czy brak umiejętności programowania jego twórców. Dawne języki też wymagały starannego dbania o pozycjonowanie instrukcji i jakoś nikt nie traktował wtedy tego jako zalety...

Zdalne sterowanie urządzeń przez Ethernet i Internet

Do wykonania aplikacji niezbędny będzie serwer HTTP, skorzystamy z serwera Apache2, aby go zainstalować wykonujemy polecenie:

apt-get install apache2

Dodatkowo instalujemy PHP5, z wykorzystaniem którego będziemy wywoływać program *gpio* .

apt-get install php5

apt-get install libapache2-mod-php5

Domyślnie pliki stron internetowych serwera Apache2 znajdują się w katalogu */var/www*, po instalacji serwera jest tam plik *index.html* , usuwamy go za pomocą polecenia:

rm /var/www/index.html

Strona internetowa do sterowania kanałami GPIO w Raspberry

ssh pi@192.168.66.26:/home/pi/strona

Aplikacja klient(Android)/serwer(Python) do sterowania GPIO

<https://play.google.com/store/apps/details?id=com.rgc>

<https://github.com/arek125/remote-GPIO-control-server>

Instalujemy potrzebne pakiety poleceniem:

sudo apt-get install python-dev python-crypto

Kopiujemy plik serwera (rgc-server1_1.py) do katalogu np. /home/pi/

Wtedy możemy uruchomić serwer poleceniem:

python /home/pi/rgc-server1_1.py

Dodatkowo do polecenia możemy dopisać parametry aby zmienić port (standardowo 8888) lub ustawić hasło

python /home/pi/rgc-server1_1.py -port 9090 -password haslo123

Robienie zdjęć na Raspberry Pi z *raspistill*

```
raspistill -o test.jpg
```

```
raspistill -n -o test.jpg
```

```
raspistill -n -o test.jpg -t 100
```

```
raspistill -n -o test_obrot.jpg -t 100 -rot 180
```

```
raspistill -n -o test.jpg -t 100 -hf -vf
```

```
raspistill -n -o test_VGA.jpg -t 100 -w 640 -h 480
```

```
raspistill -n -o test_%d.jpg -t 100 -dt
```

```
raspistill -n -o test_%d.jpg -t 100 -ts
```

Transmisja obrazu przez sieć - pakiet Motion

Ciekawym zastosowaniem dla kamery Raspberry Pi jest pakiet Motion, który pozwala między innymi transmitować obraz przez sieć. Co więcej, może on być również wykorzystany do śledzenia ruchu obiektów.

Instalacji programu:

apt install motion

Instalacja sterownika kamery:

modprobe bcm2835-v4l2

Domyślny plik z ustawieniami to:

/etc/motion/motion.conf

W pliku konfiguracyjnym wprowadzamy zmiany.

stream_localhost , domyślnie jest ona włączona (*on*), zmieniamy ją na **off**

stream_localhost off

uruchamiamy pakiet Motion poleceniem:

sudo motion

W pasek adresu wpisujemy adres IP malinki z portem 8081

np. http://192.168.66.26:8081

Do wyszukiwania tekstu w **nano** można
wykorzystać skrót **CTRL+W**.

Transmisja w obecnej formie jest mocno ograniczona, więc zmieniamy opcje w pliku konfiguracyjnym. Zaczynamy od znalezienia miejsca na wpisanie dwóch wartości:

framerate (ustawiamy na xx)

stream_maxrate (ustawiamy na xx)

Zwiększenie rozdzielczości obrazu, zmieniamy opcje

width (ustawiamy na 640),

height (ustawiamy na 480).

Wykrywanie ruchu na Raspberry Pi

Zmieniliśmy Raspberry Pi w kamerkę internetową. Warto pamiętać, że pakiet Motion znacznie **przewyższa możliwości typowych kamerek**. Pozwala na wykrywanie ruchu, generowanie zdarzeń po wykryciu intruza, automatyczne robienie zdjęć lub filmów, powiadamianie o zdarzeniach itd.

Dla testu można uruchomić opcję odpowiadającą za śledzenie ruchu. W tym celu wystarczy w pliku konfiguracyjnym włączyć opcję:

locate_motion_mode on

Od teraz na transmitowanym obrazie ruchome elementy będą obrysowane prostokątem. Aby przedmiot był obrysowany czerwony prostokątem należy zmienić jeszcze `locate_motion_style` na `redbox`.

locate_motion_style redbox

Uruchamianie transmisji w tle

Łatwo możemy sprawić, że *Motion* uruchomi się w tle, czyli jako usługa

Aby było to możliwe możemy:

- zmienić w pliku konfiguracyjnym pcję **daemon** z *off* na *on*
- uruchamiać program z parametrem **-b**

Do sterowania usługami w Linuksie używa się programu systemctl:

Wyświetlenie statusu:

sudo systemctl status motion.service

Zatrzymanie usługi:

sudo systemctl stop motion.service

Uruchomienie usługi:

sudo systemctl start motion.service

Nagrywanie filmów na Raspberry Pi z *raspivid*

raspivid -n -w 1920 -h 1080 -cd MJPEG -o test.mpg

-n, wyłączyło podgląd;

-w oraz **-h**, rozdzielczość obrazu;

-o, plik wynikowy;

-cd, ustalenie metody kompresji obrazu (MJPEG);

-rot, obraz można obrócić;

-t, ustawia długość filmu (w milisekundach) ,domyślna wartość to 5000 ms, czyli 5 sekund.

Aby na PC z Windowsem pliki te były odtwarzane poprawnie należy doinstalować kodeki

paczka odpowiednich kodeków to darmowy [K-Lite Codec Pack Standard](http://www.codecguide.com/download_k-lite_codec_pack_standard.htm).

http://www.codecguide.com/download_k-lite_codec_pack_standard.htm

Jeśli nie ustawimy parametru odpowiadającego za sposób kompresji, to materiały zostaną domyślnie nagrane z użyciem kodeka h.264, który jest znacznie wydajniejszy:

raspivid -n -w 640 -h 480 -o test_2.h264

Filmy w zwolnionym tempie.

Parametr **-fps**, pozwala na zmianę liczby klatek, z którą nagrywane są materiały wideo. Przy rozdzielczości VGA jesteśmy w stanie nagrywać materiały 90 FPS, co pozwala na uzyskanie spowolnionych materiałów.

raspivid -n -w 640 -h 480 -o test.h264 -fps 90

Film poklatkowy

Korzystając z poznanego wcześniej *raspistill* można bardzo łatwo tworzyć **filmy poklatkowe**. Zasada jest bardzo prosta. Robimy zdjęcia co określony czas, a następnie łączymy je w film.

Nagrywanie filmów poklatkowych **wymaga wykonania setek, a często nawet tysięcy zdjęć**.

Przechodzimy do nowego folderu i uruchamiamy w nim *raspistill* z następującymi parametrami:

raspistill -t 300000 -tl 2000 -o image%04d.jpg -w 1280 -h 720 -rot 180

- t** określa jak długo chcemy robić zdjęcia,
300000 ms, czyli 300 s, co daje dokładnie 5 minut.
- tl** określa interwał (co ile czasu chcemy robić nowe zdjęcie),
2000 ms, czyli nowe zdjęcie będzie wykonywane co 2 sekundy.
- o** to oczywiście nazwa pliku z dodaną datą.

Zdjęcia gotowe, pora więc połączyć je w film. Tutaj pomocny będzie program **avconv**, który wchodzi w skład pakietu **libav-tools**. Instalujemy więc konieczne narzędzia:

apt install libav-tools

Po instalacji możemy wywołać polecenie, które wygeneruje dla nas film:

avconv -i image%04d.jpg -r 10 -vcodec libx264 -vf scale=1280:720 timelapse.mp4

-i wybieramy materiał wejściowy (nasze zdjęcia).

-r określamy liczbę klatek na sekundę.

W przykładzie wpisaliśmy 10, co oznacza, że na jedną sekundę naszego filmu przypadnie 10 zrobionych wcześniej zdjęć.

Kolejne parametry określają:

- kodek,
- rozdzielczość
- nazwę utworzonego wideo.

