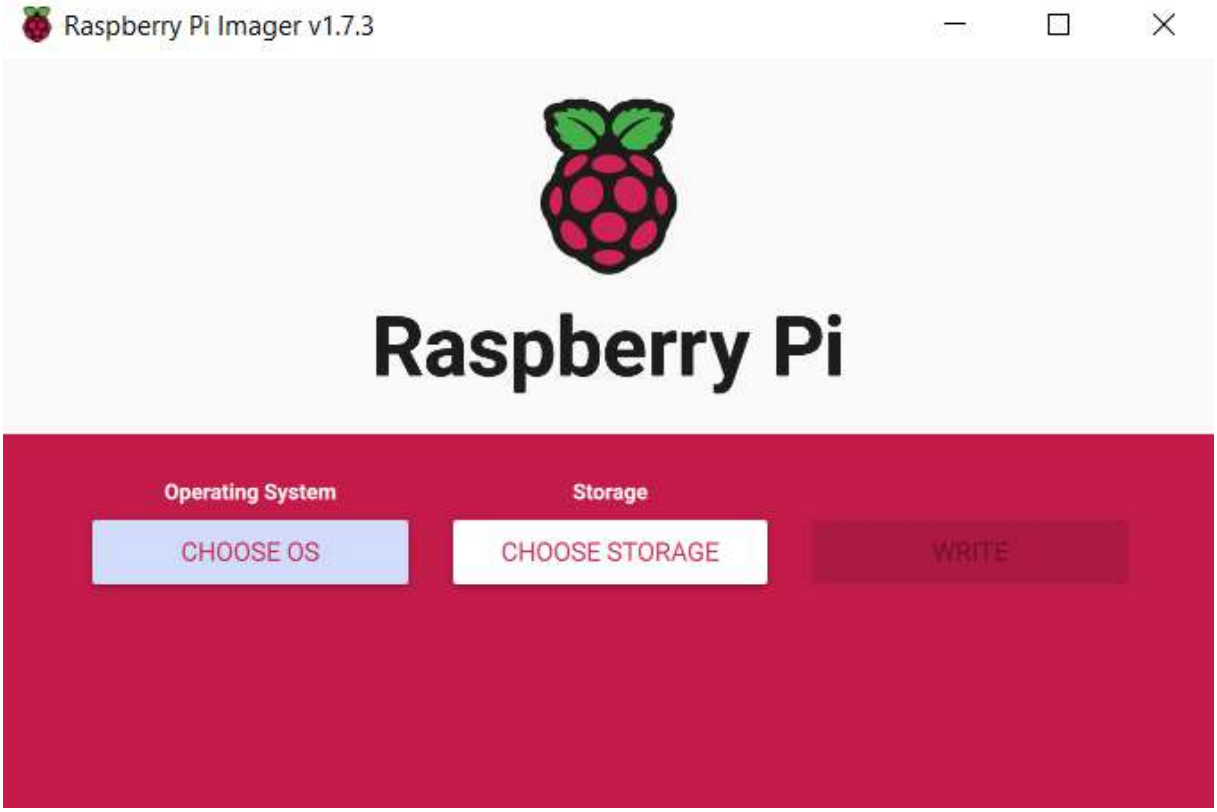
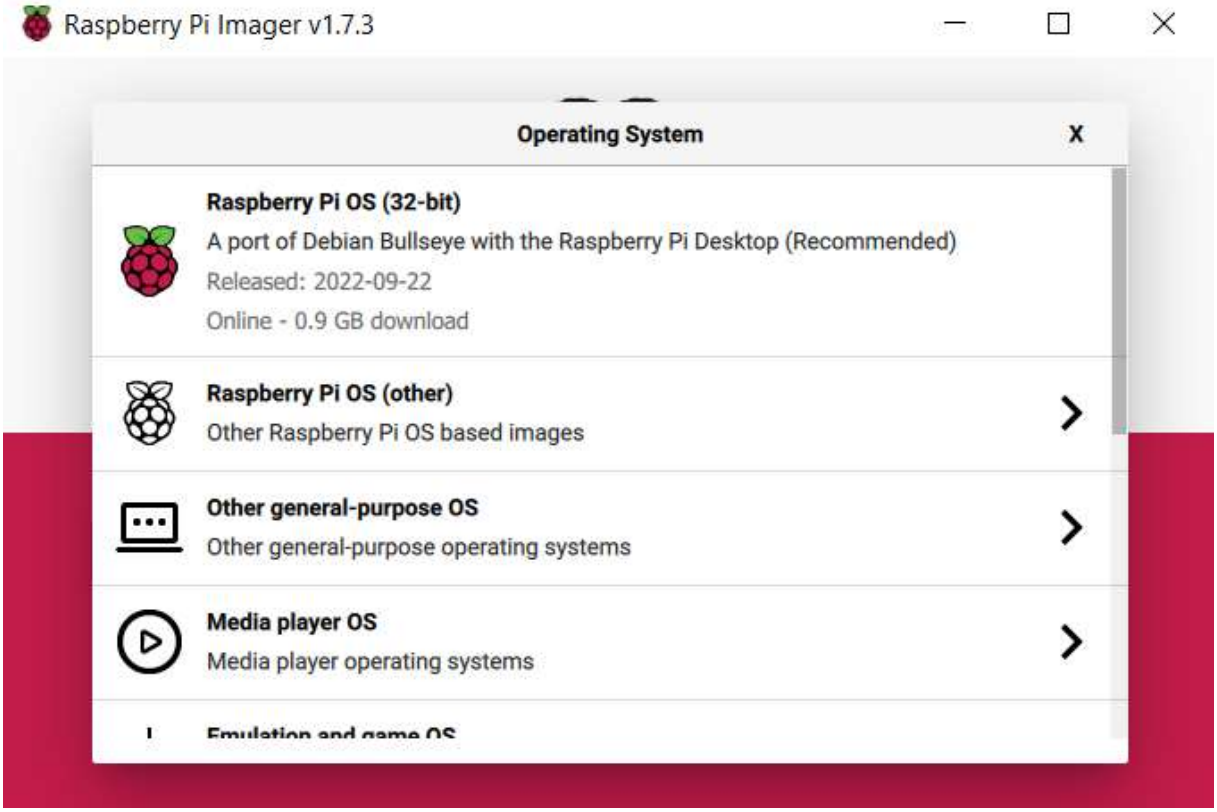
Przygotowanie systemu (pamięci SD) do płytki RaspberryPI (AlphaBot)

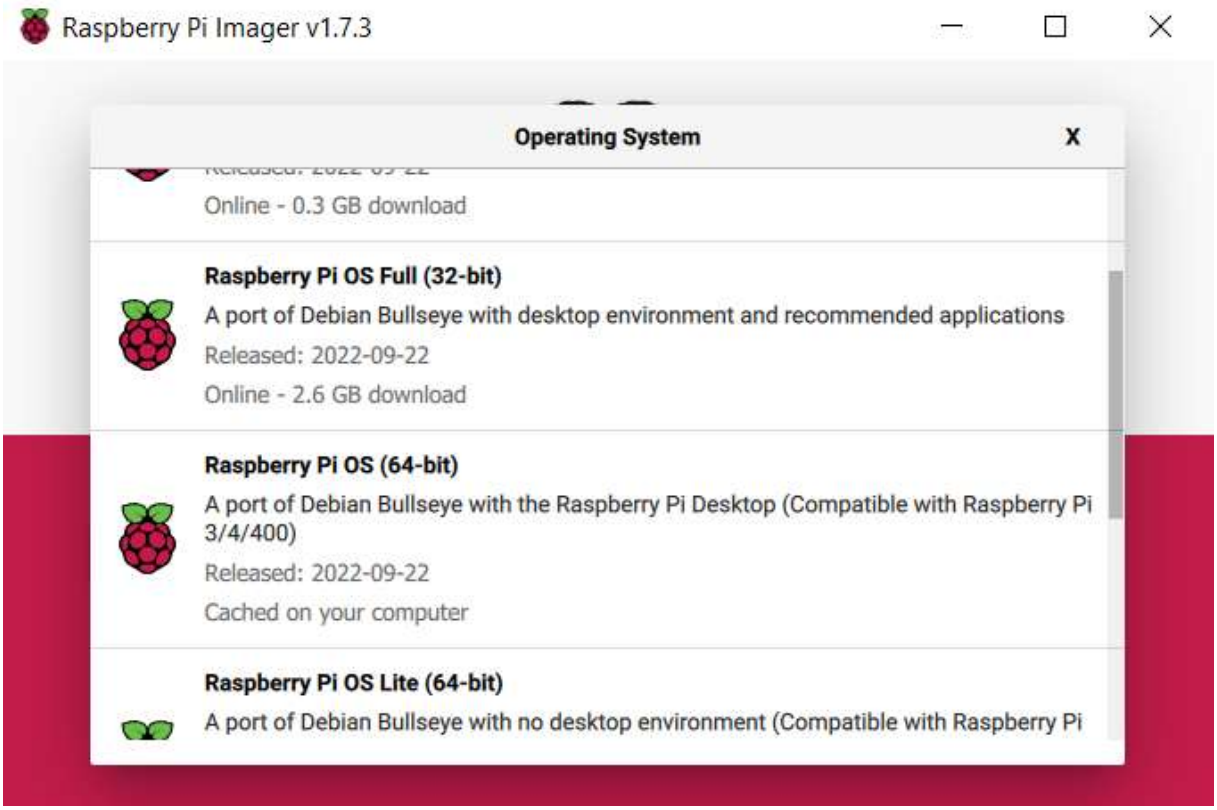
1. Znajdź program Imager, zainstaluj, uruchom, wybierz Operating System



2. wybierz opcję RaspberryPI OS (other)

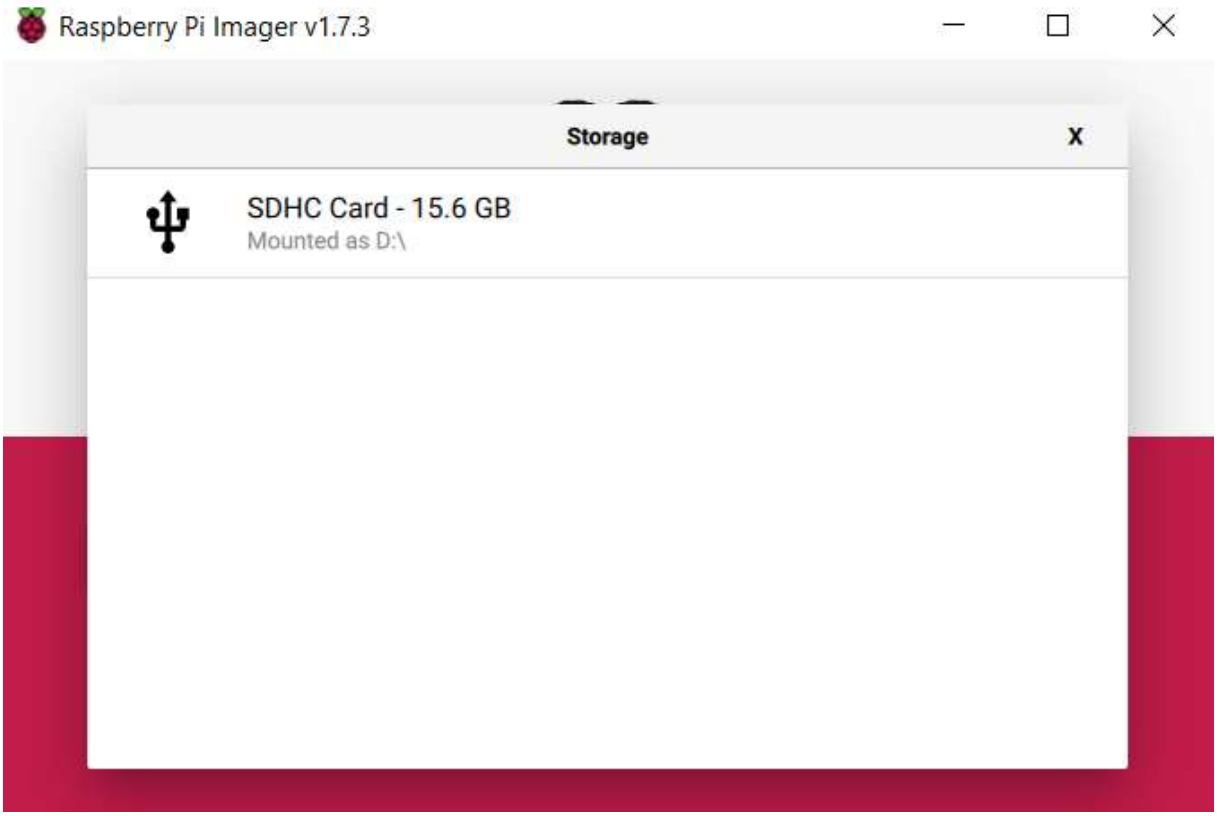


3. Wybierz opcję RspberryPI OS (64-bit)

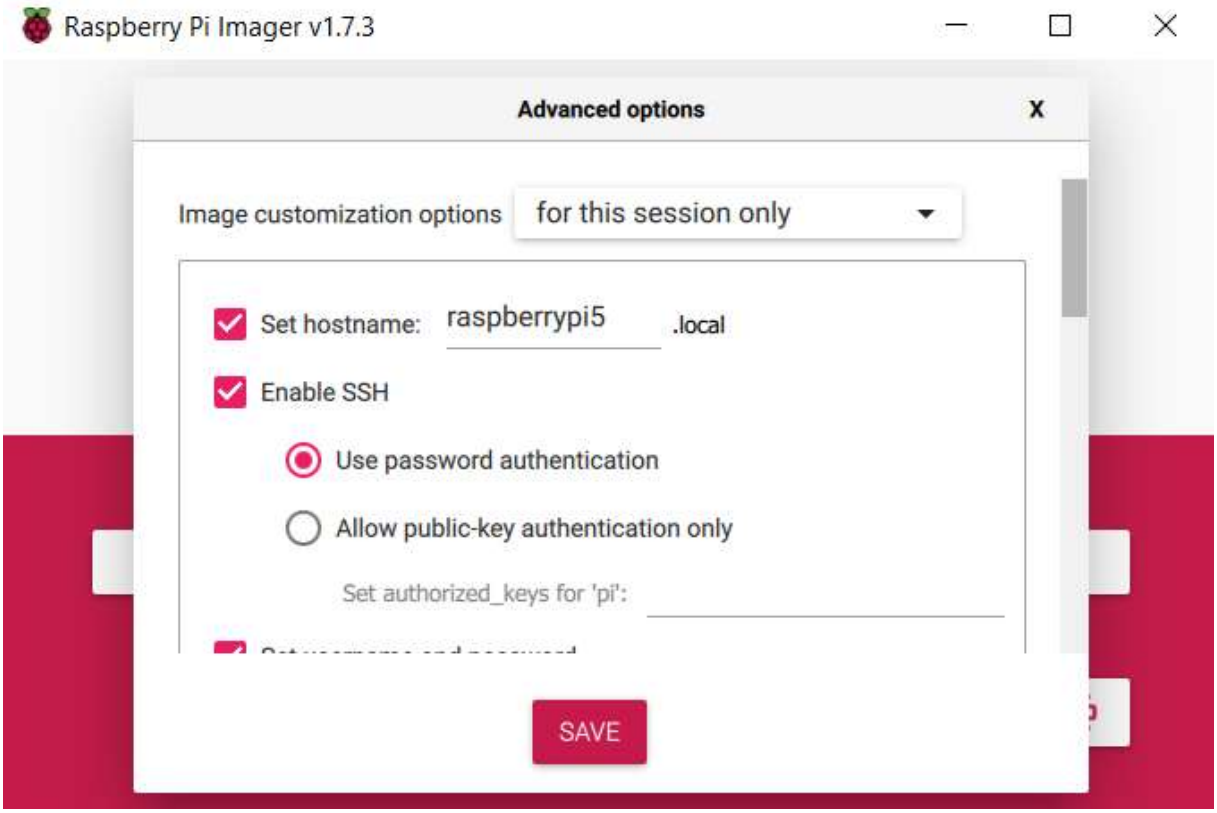


4. Wybierz opcję Storage

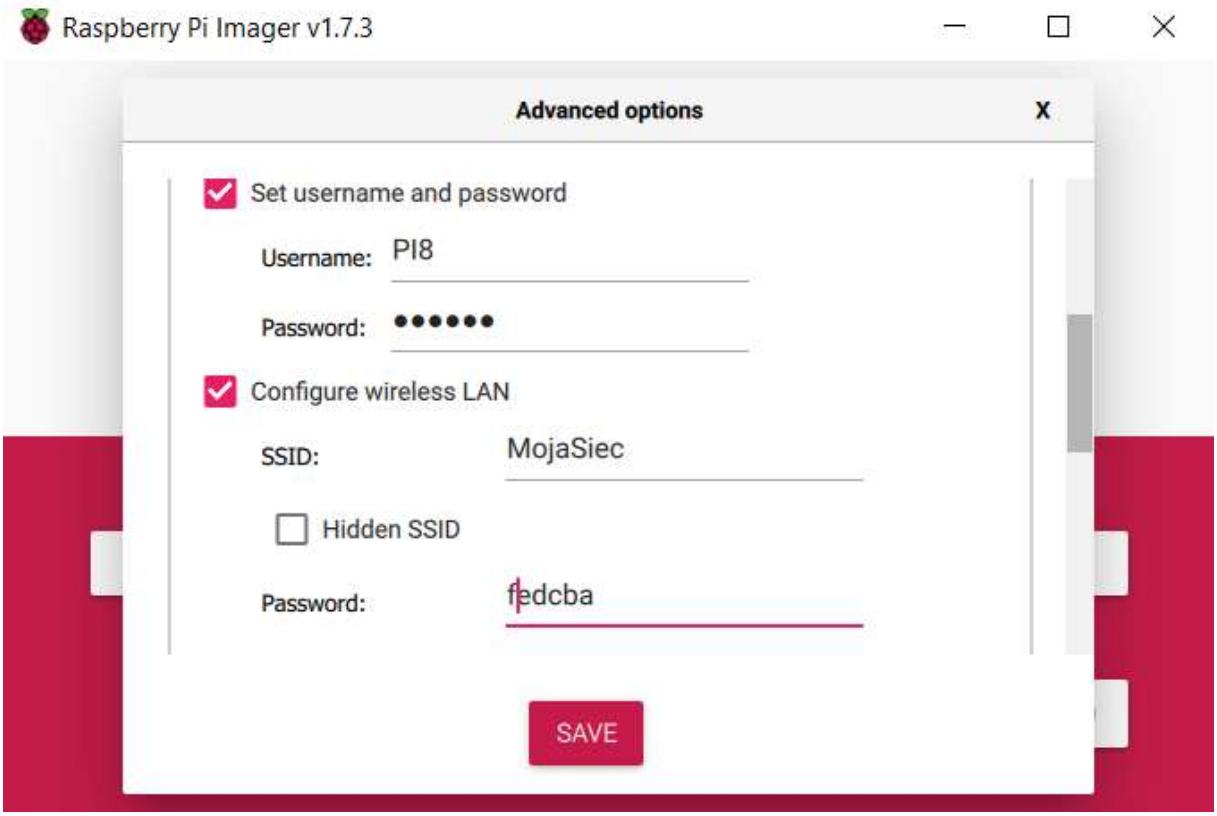
5. Wybierz dysk z pamięcią, który zostanie skasowany i zformatowany, a następnie nagrany LINUX



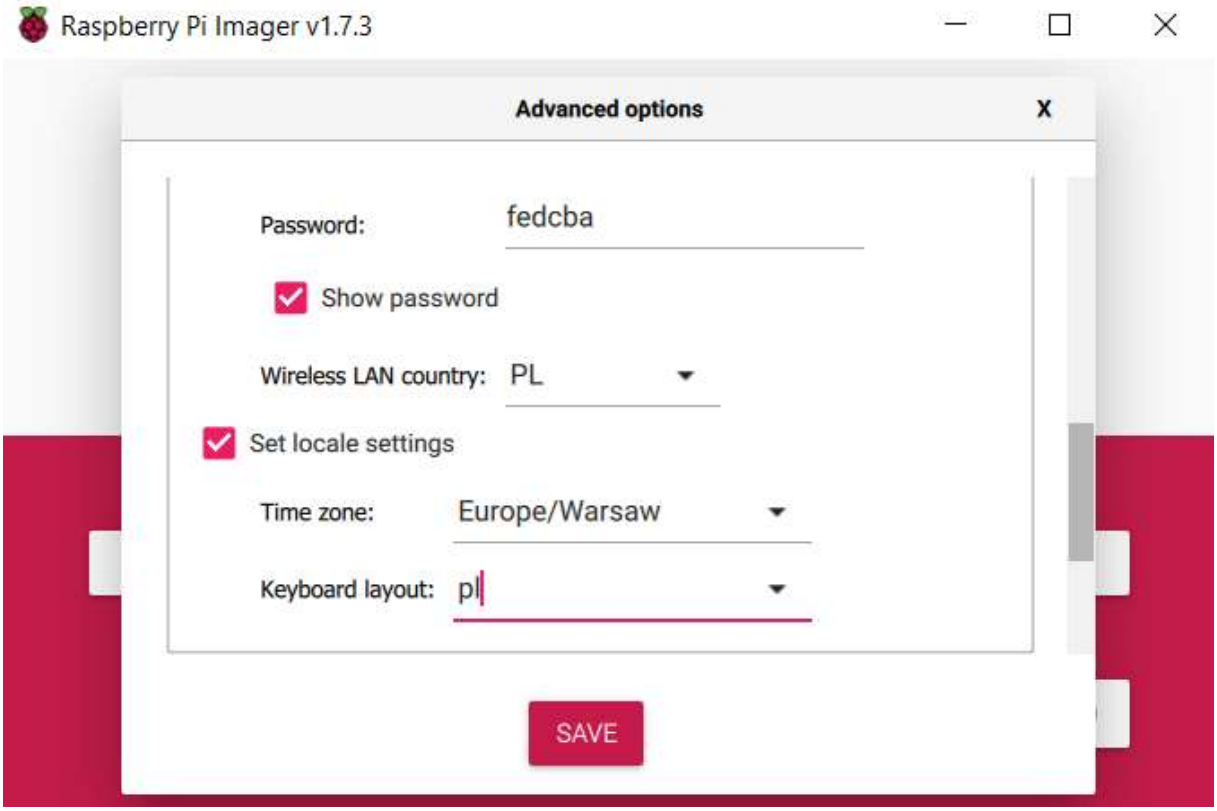
6. Wybierz ustawienia:
   Set hostname: na przykład ‘raspberrypi5’; - Set Enable SSH



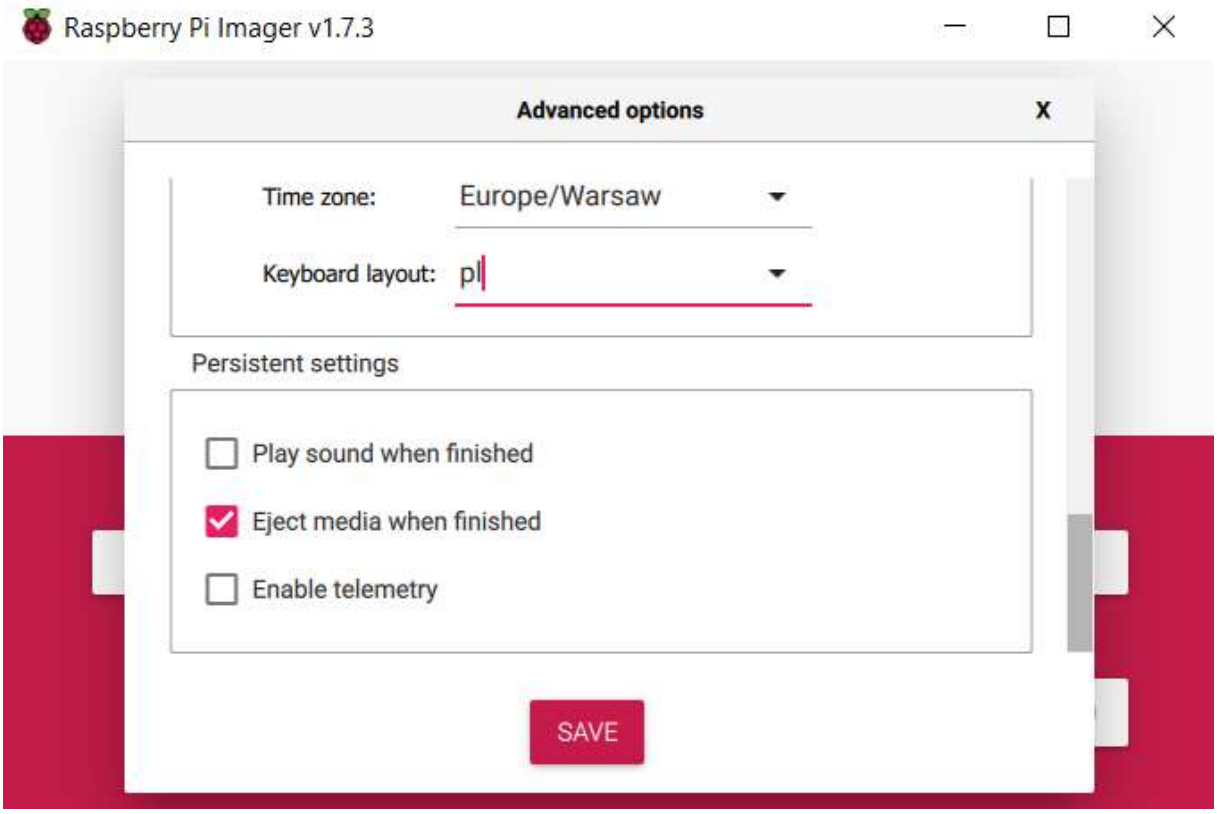
7. Wybierz ustawienia:
   - Set username and password; - Set configure wireless LAN and password



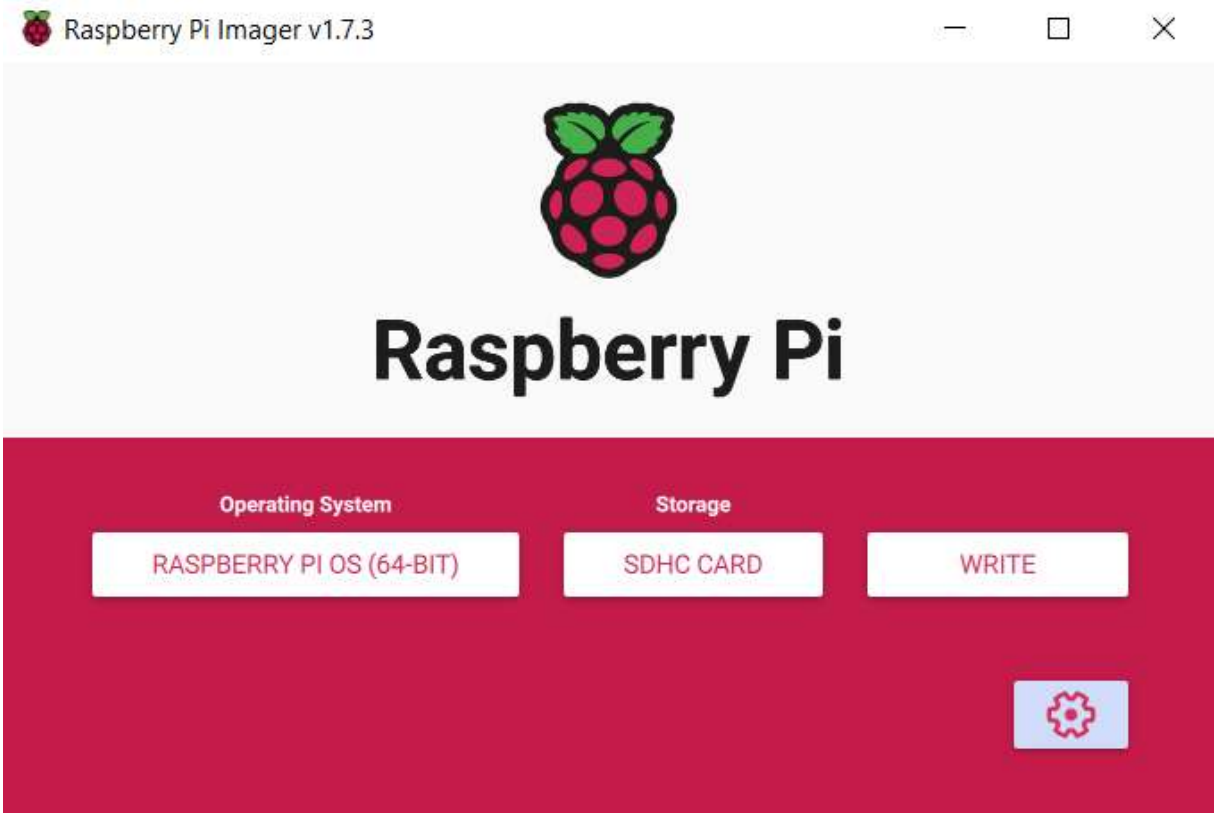
8. Wybierz ustawienia



9. Pozostałe ustawienia i zapisz



10. Pozostało już tylko sformatować – wciśnij WRITE

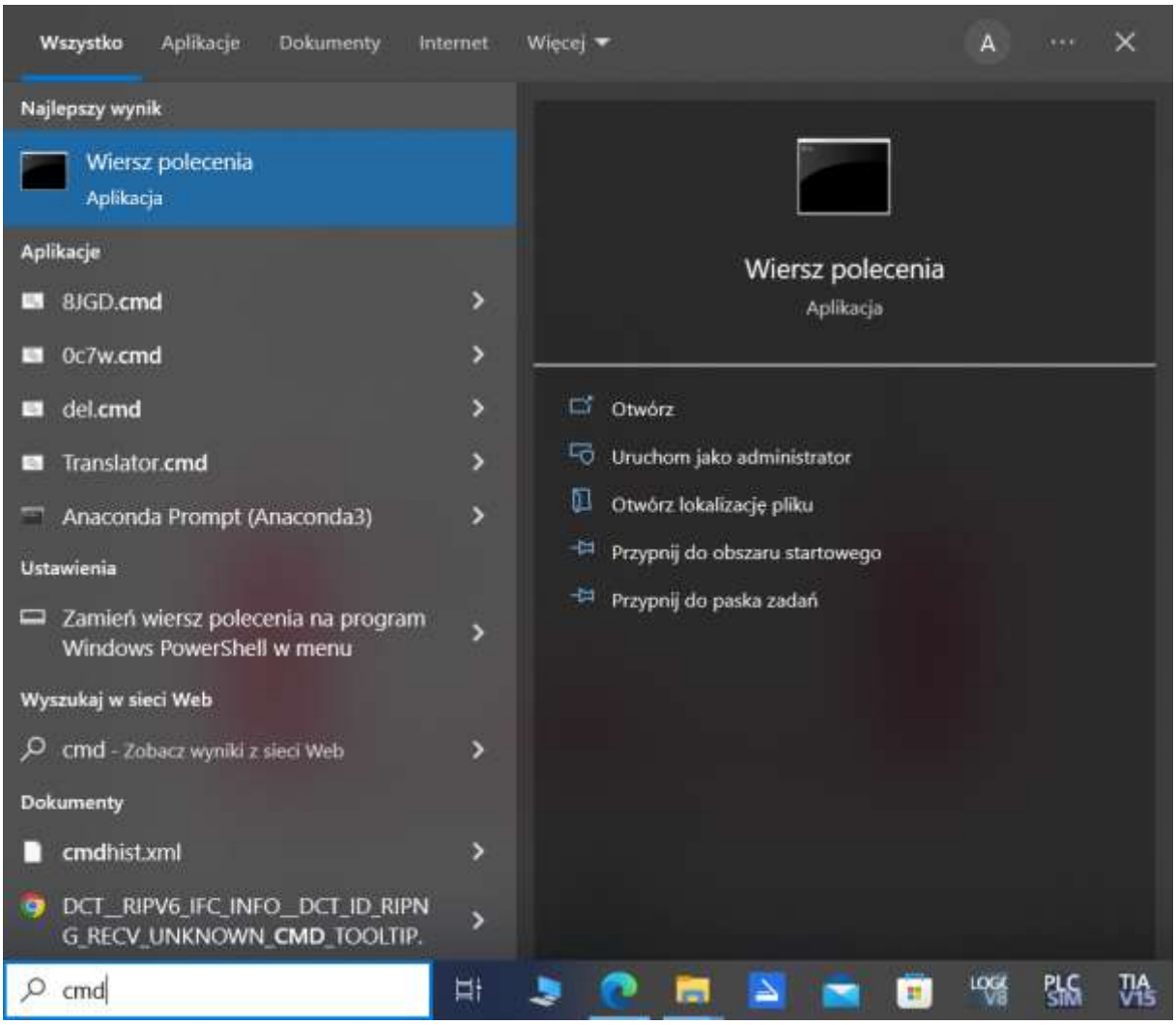


W ten sposób przygotowaną kartę SD wkładamy do płytki RaspberryPI, która to łączy się z podaną siecią bezprzewodową. Musi być załączone zasilanie. Po podłączeniu monitora, myszy i klawiatury możemy pracować na komputerze RaspberryPI .

Inną opcją pracy jest wirtualny ekran na laptopie pod Windowsem. Laptop musi być podłączony do tej samej sieci wi-fi. Może to być sieć z routerem, hotspot mobilny z laptopa. Sieć wi-fi hotspot z komórki też działa. Laptop i RaspberryPI musi być w jednej sieci, która została zdefiniowana na karcie SD.

Przygotowanie wirtualnego ekranu:

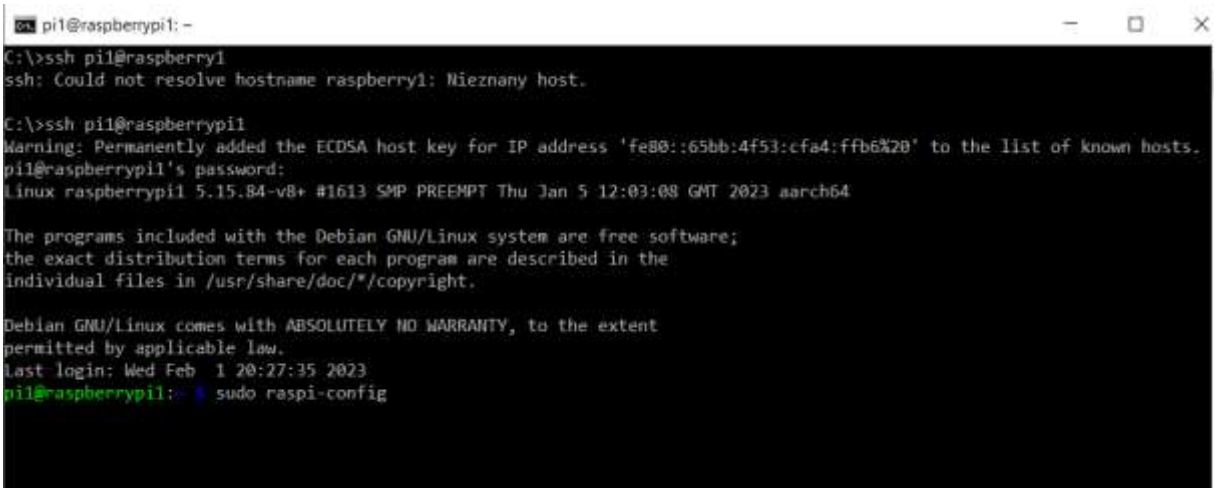
1. W wierszu poleceń wpisujemy **cmd** i uruchamiamy



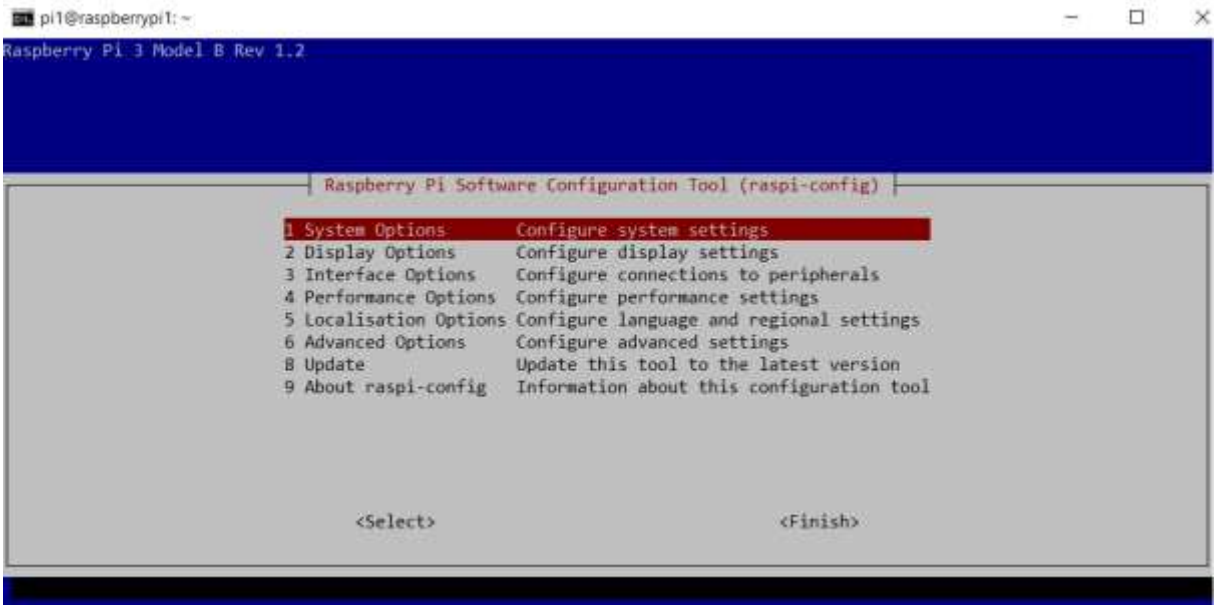
2. Łączymy się z RaspberryPI poprzez nazwę Hostname i Username zadeklarowane na karcie SD i polecenie **ssh**

   # ssh pi8@raspberrypi5

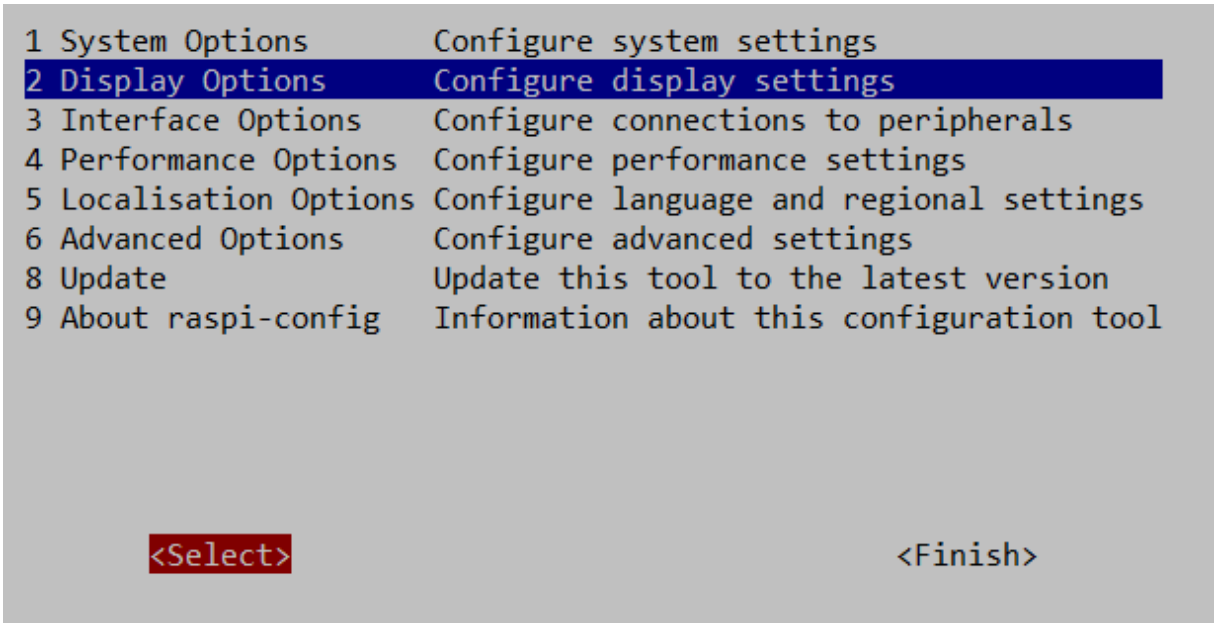
   system pyta się o password użytkownika zadeklarowany na karcie SD, wpisujemy go poprawnie (system nie pokazuje żadnych znaków i kursor nie przesuwa się) i wciskamy Enter
   System pokazuje poprawne zalogowanie się do RaspberryPI.
   Następnie uruchamiamy plik konfiguracyjny do Linuxa:
   sudo raspi-config

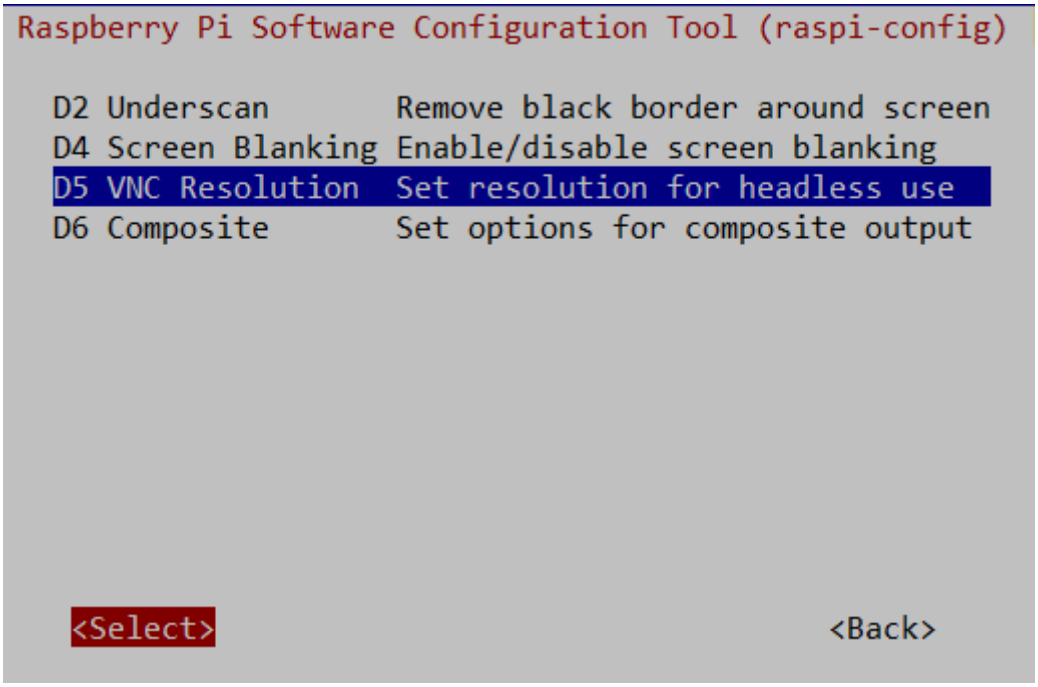


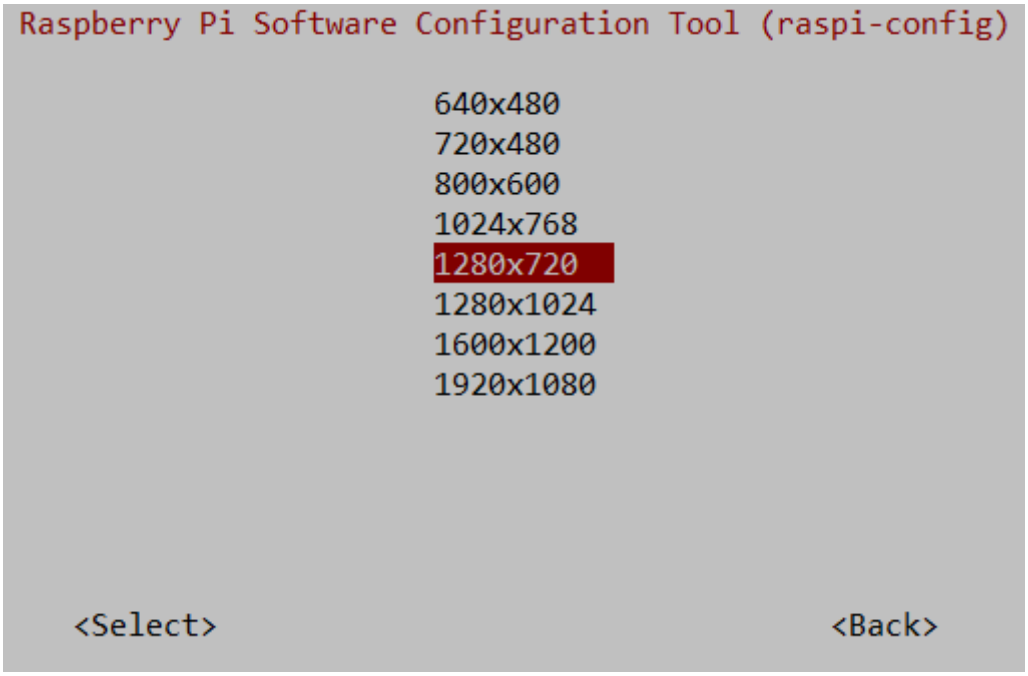
3. Otrzymujemy zgłoszenie do ustawień konfiguracyjnych



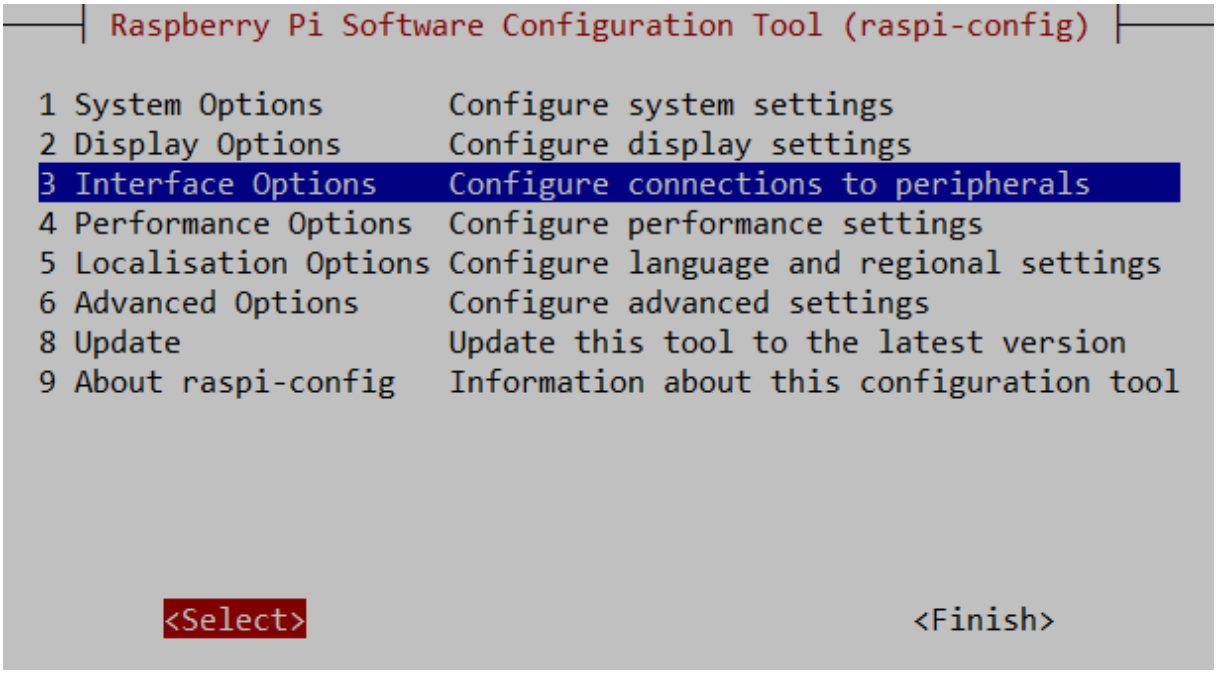
4. Warto ustawić rozdzielczość ekranu **Display Option** – 1280-720

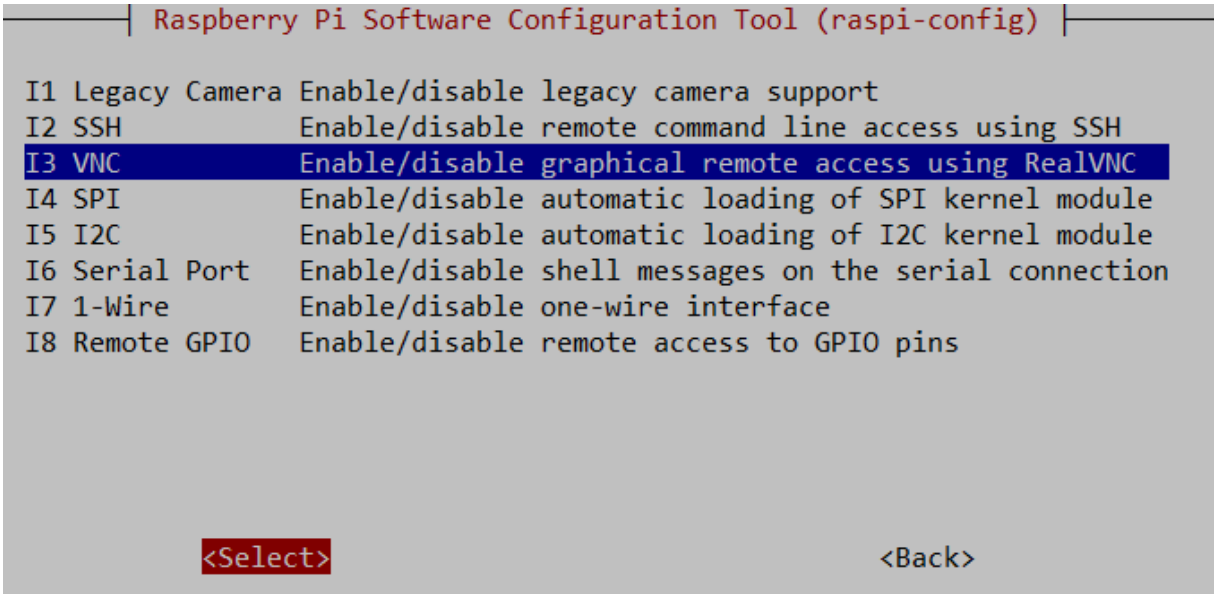


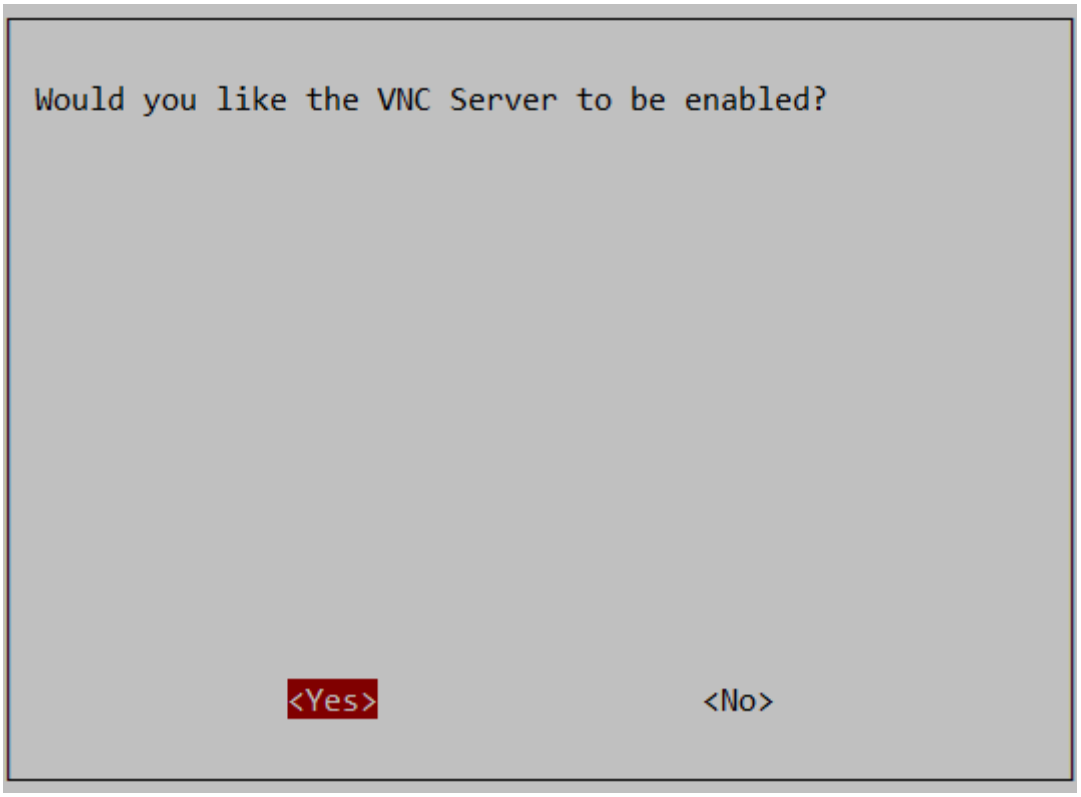




5. Konieczne jest ustawienie wirtualnego ekranu **Interface Option** i wybór **VNC** z potwierdzeniem **YES** i **OK**

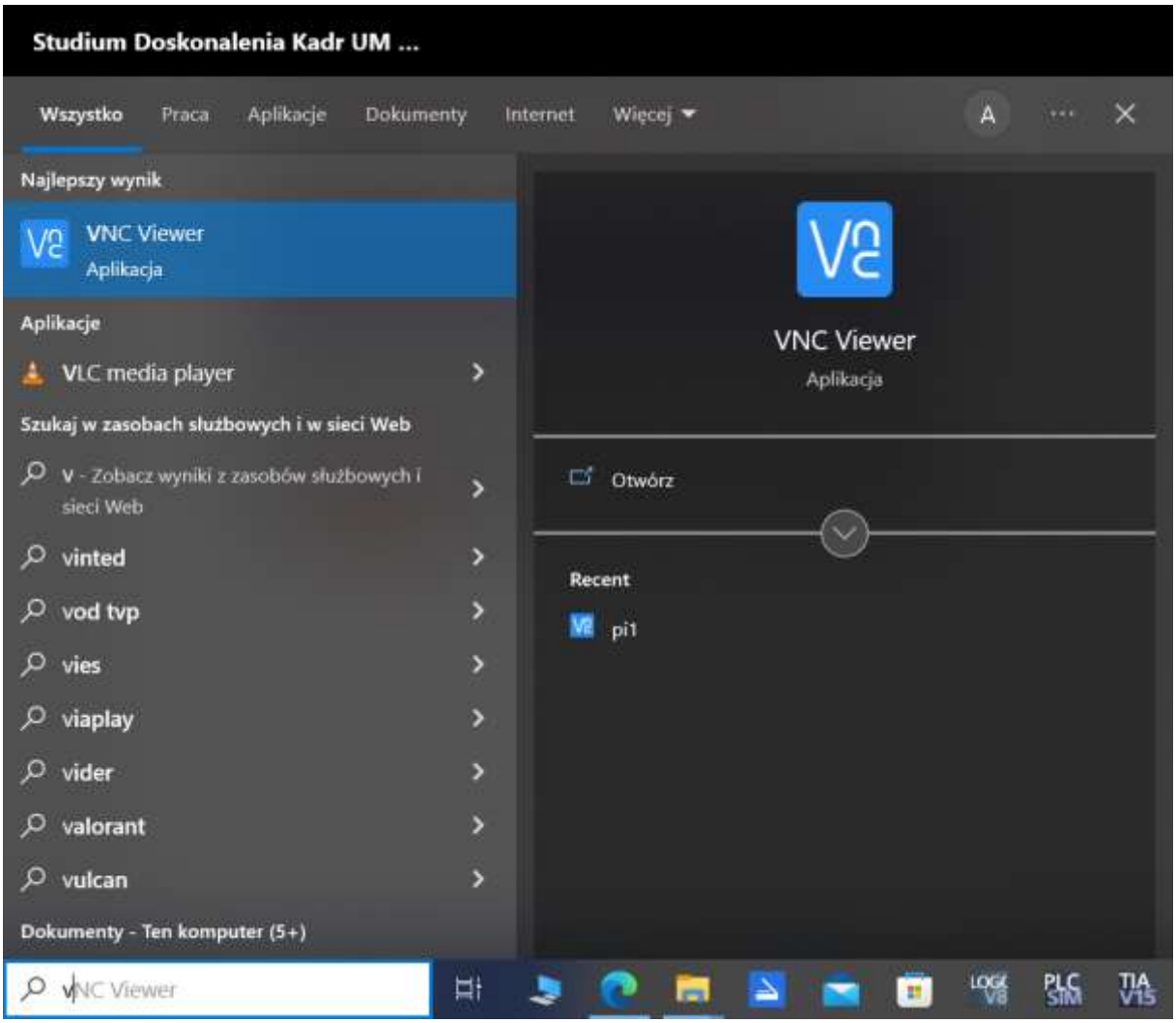




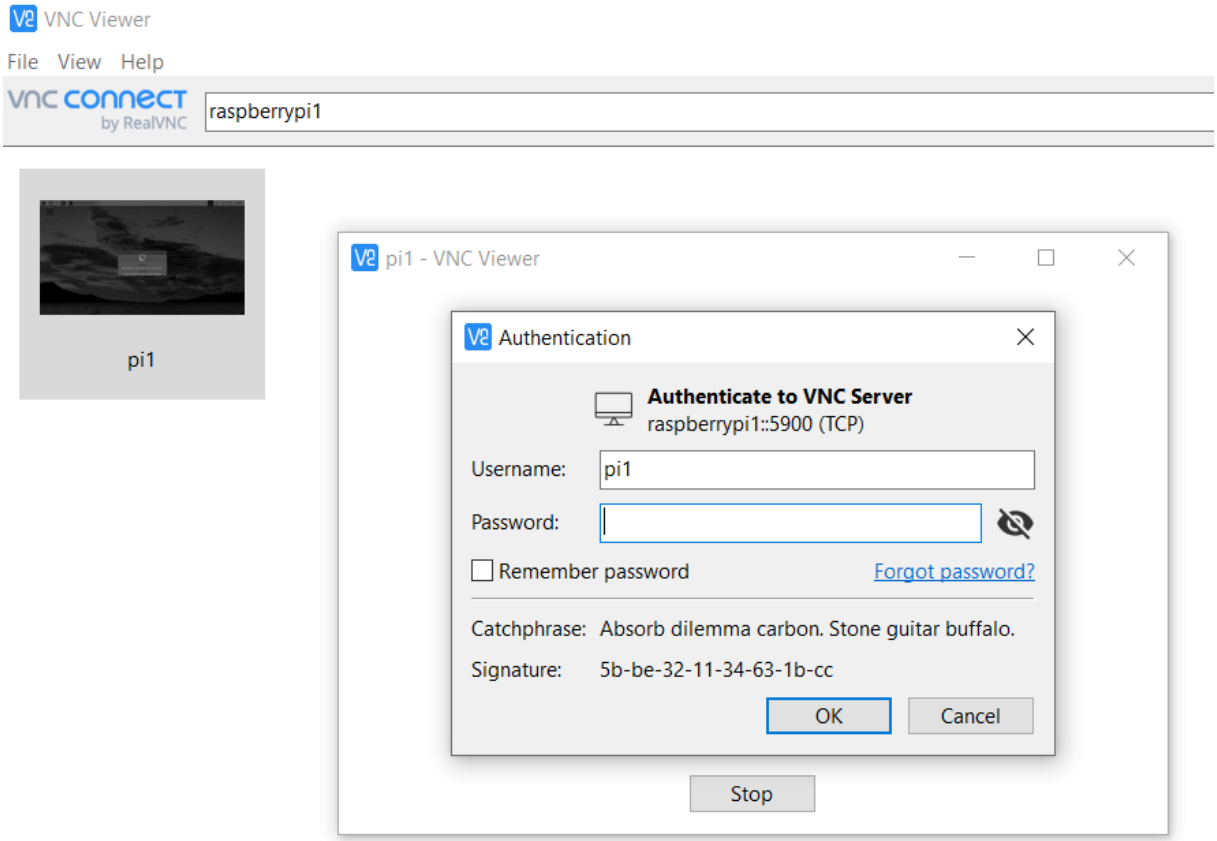


6. Ostatecznie wybieramy **Finish** i wychodzimy z okna programu **cmd**. Pozostałe opcje warto zmieniać już po uzyskaniu ekranu wirtualnego.

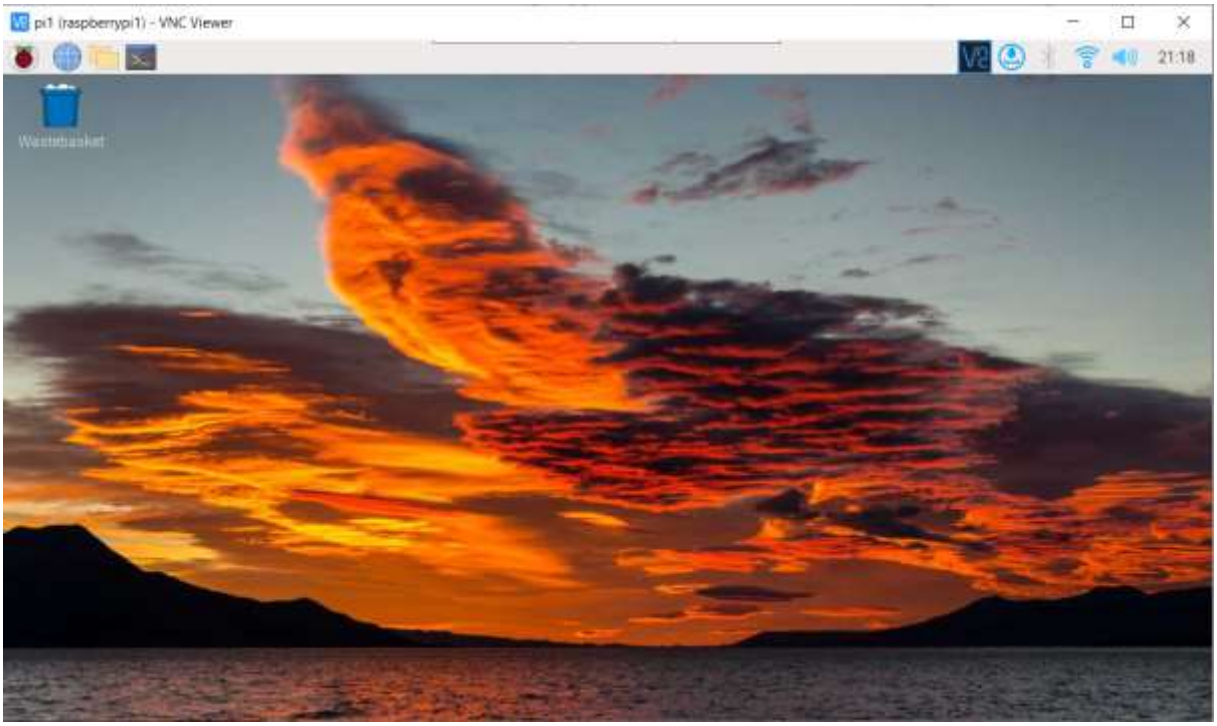
7. Instalujemy na laptopie program **VNC Viewer** i uruchamiamy



8. Wpisujemy nazwę Hostname i czekamy na zgłoszenie się systemu, kolejno wpisujemy username i password

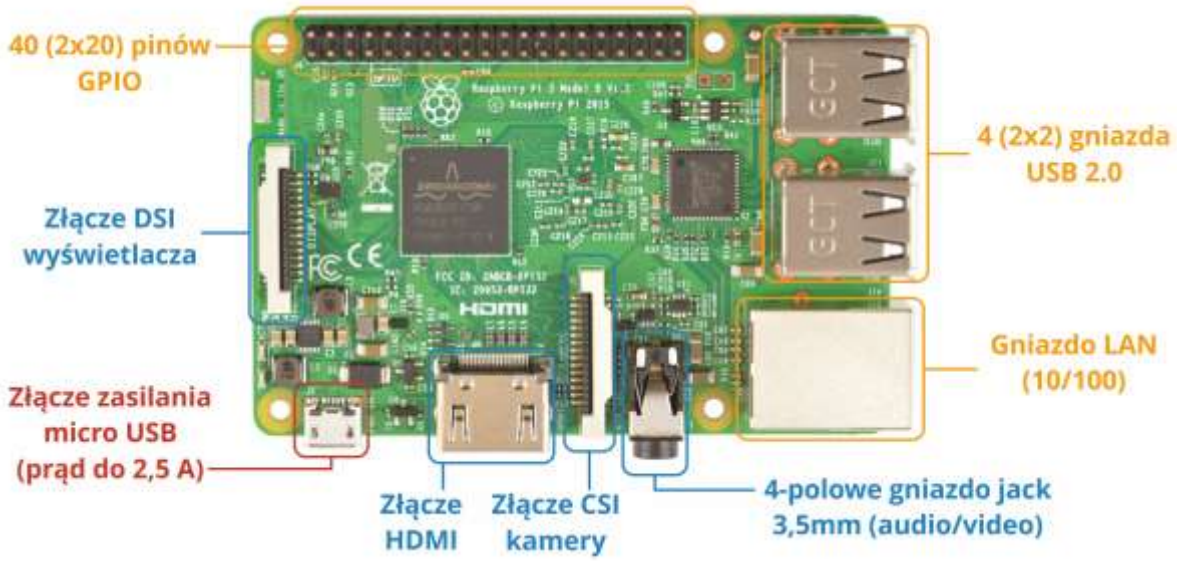


9. Nasza malinka gotowa do pracy

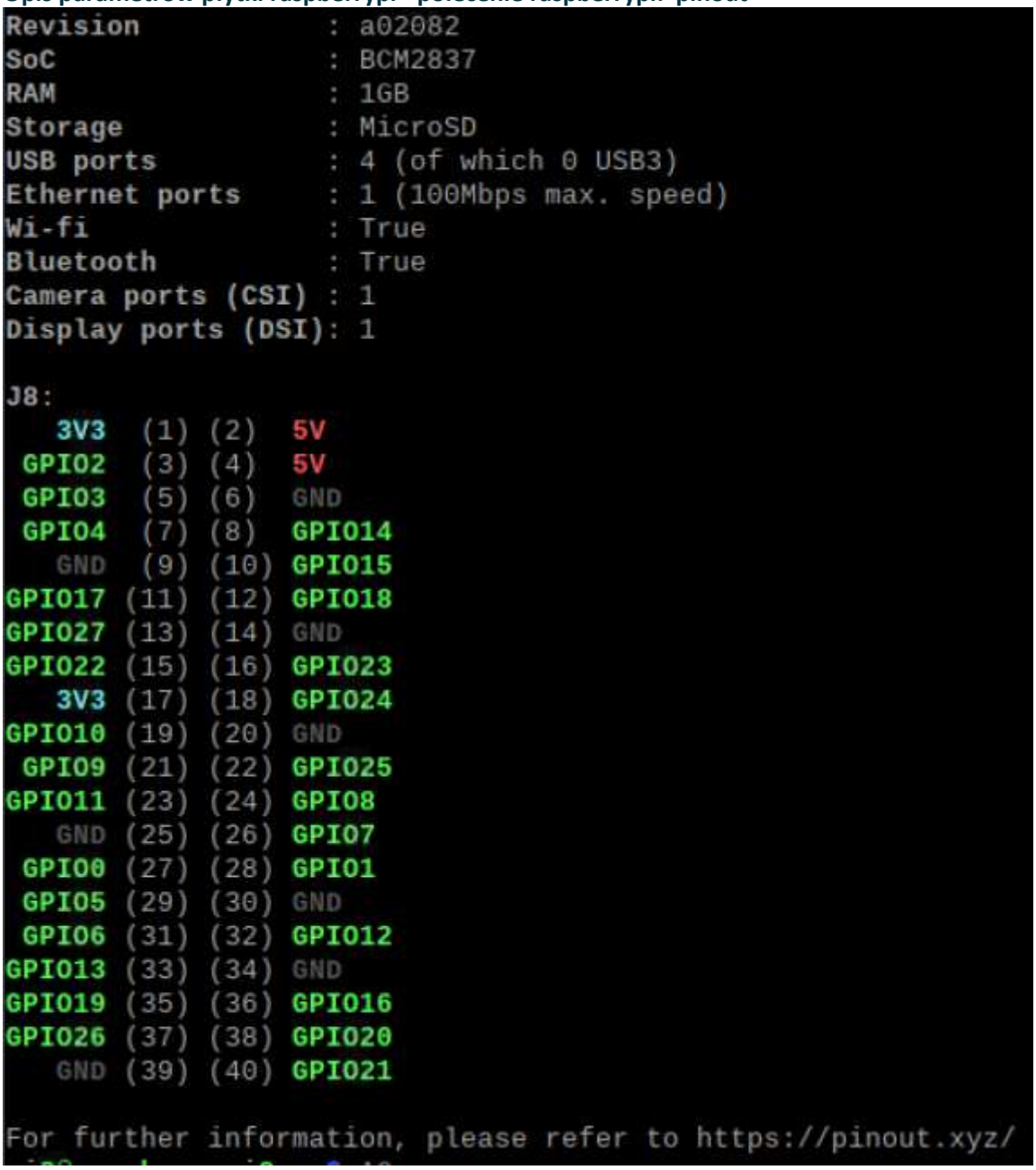


Specyfikacja techniczna Raspberry Pi 3:

- **Procesor:** Broadcom BCM2837, 64-bity, architektura ARMv8-A, **4 rdzenie 1,2 GHz**
- **Pamięć:** RAM: 1 GB LPDDR2 900 MHz, Flash: złącze karty microSD
- **USB:** 4x USB 2.0 typ A
- **Wideo:** HDMI HD 1080 px / 30 fps, złącze kompozytowe (PAL, NTSC)
- **Audio:** 3,5 mm jack, HDMI
- **Połączenie z Internetem:** Ethernet: 10/100 Mbps, 802.11 b/g/n 150 Mbps
- **Bluetooth:** BLE 4.1
- **Zasilanie:** 5V, 2.5A (przez gniazdo micro-usb)
- **Złącze rozszerzeń (GPIO):** 40 pinów
- **Interfejs wyświetlacza:** DSI
- **Interfejs kamery:** CSI



Opis parametrów płytki raspberrypi - polecenie raspberrypi: pinout

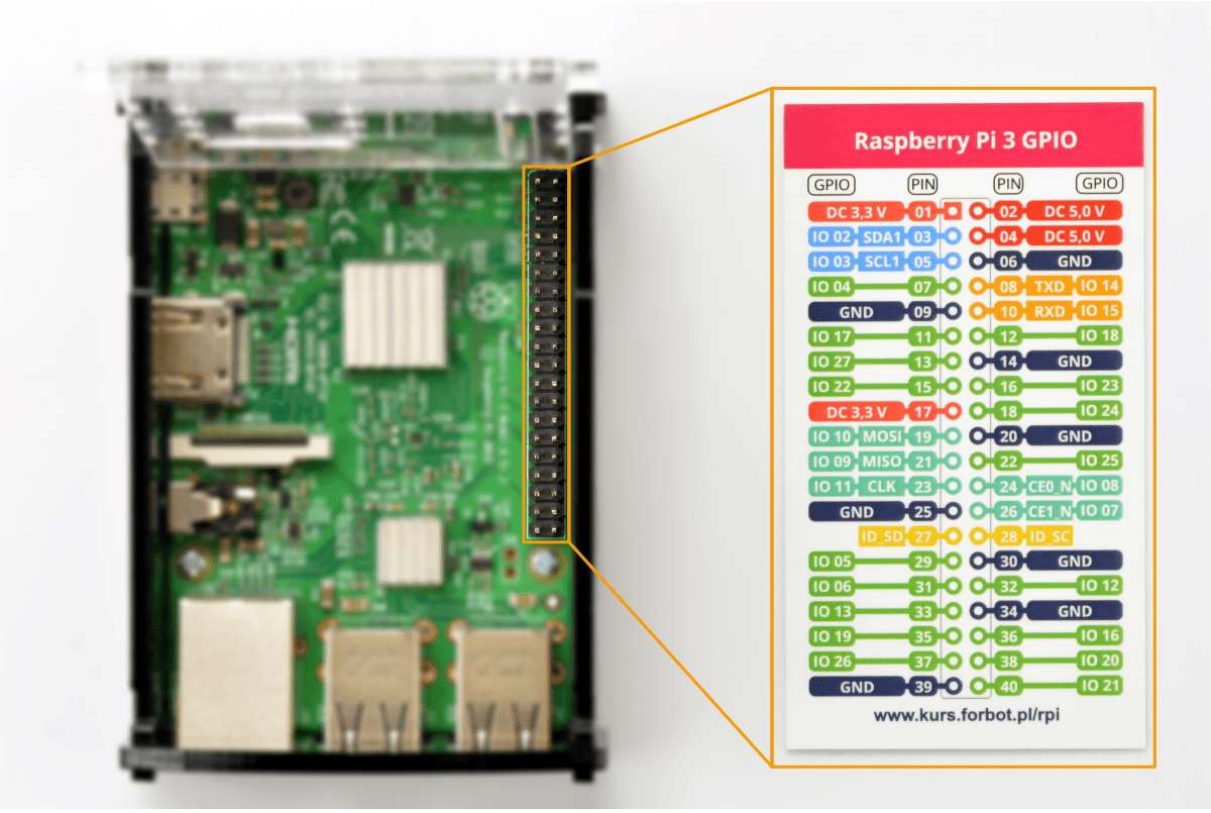


```
Revision           : a02082
SoC                : BCM2837
RAM                : 1GB
Storage            : MicroSD
USB ports          : 4 (of which 0 USB3)
Ethernet ports     : 1 (100Mbps max. speed)
Wi-fi              : True
Bluetooth          : True
Camera ports (CSI) : 1
Display ports (DSI): 1

J8:
   3V3  (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
   3V3 (17) (18) GPIO24
GPIO10 (19) (20) GND
 GPIO9 (21) (22) GPIO25
GPIO11 (23) (24) GPIO8
   GND (25) (26) GPIO7
 GPIO0 (27) (28) GPIO1
 GPIO5 (29) (30) GND
 GPIO6 (31) (32) GPIO12
GPIO13 (33) (34) GND
GPIO19 (35) (36) GPIO16
GPIO26 (37) (38) GPIO20
   GND (39) (40) GPIO21

For further information, please refer to https://pinout.xyz/
```

Strona pinout.xyz

- GPIO (General Purpose IO)
- SPI (Serial Peripheral Interface)
- I2C (Inter-integrated Circuit)
- UART (Universal Asyncronous Receiver/Transmitter)
- PCM (Pulse Code Modulation)
- Ground
- 5v (Power)
- 3.3v (Power)

Numeracja pinów (wejść/wyjść) na płytce raspberrypi:



Numeracja pinów – polecenie `gpio readall`

```
pi@raspberrypi: ~
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  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
pi@raspberrypi:~ $
```

Tryb w jakim będziemy odwoływali się do pinów może być:

- GPIO.BOARD – odwołanie bezpośrednio do numeru pinu w złączu, zalecam ten tryb ze względu na to, że jest uniwersalny dla wszystkich malinek.
- GPIO.BCM – odwołanie do numeru pinu na procesorze – może być różne w różnych wersjach Raspberry.

Tryb ustawiamy za pomocą `GPIO.setmode(GPIO.BOARD) lub GPIO.setmode(GPIO.BCM)`

import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False) - kasowanie alarmów
GPIO.setup(21, GPIO.OUT) – ustawienie kanału jako wyjście
GPIO.setup(21, GPIO.IN) – ustawienie kanału jako wejście
GPIO.output(21, GPIO.HIGH) – ustawienie stanu wysokiego
GPIO.output(21, GPIO.LOW) – ustawienie stanu niskiego
GPIO.cleanup() - zakończenie
python3 motor.py - uruchomienie programu

Sygnał PWM - programy

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(12, GPIO.OUT)

dioda = GPIO.PWM(12, 50) #Nowa instancja PWM
wypelnienie = 0 #Wypełnienie sygnału PWM
dioda.start(wypelnienie) #Uruchomienie sygnału PWM

try:
        while True:
          wypelnienie += 5
          if wypelnienie > 100:
              wypelnienie = 0
          dioda.ChangeDutyCycle(wypelnienie) #Ustaw nową wartość wypełnienia
          time.sleep(0.05)
except KeyboardInterrupt:
        print('Koniec')

dioda.stop()
GPIO.cleanup()
```

```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(26, GPIO.IN, pull_up_down=GPIO.PUD_UP)

try:
        while True:
                if GPIO.input(26) == 0:
                  GPIO.output(12, GPIO.HIGH)
                else:
                  GPIO.output(12, GPIO.LOW)
except KeyboardInterrupt:
        print('Koniec')

GPIO.cleanup()
```

Podstawy Pythona

Aby uruchomić interpreter Pythona, który będzie na żywo przetwarzał nasze polecenia wystarczy wpisać w konsoli Raspberry polecenie `python`. Aby opuścić program możemy wywołać funkcję `exit()`

Komentarze

Komentarz zaczyna się znakiem # (pod warunkiem, że nie jest on pomiędzy cudzysłowami – wtedy jest to normalnie część stringa)

```
# oto przykładowy komentarz :)
x = 5 # to też jest komentarz
y = "# to nie jest komentarz :("
```

Zmienne

Python jest językiem typowanym dynamicznie, dlatego przy tworzeniu zmiennej nie musimy podawać jej typu.

```
calkowita = 5
zmiennoprzecinkowa 9.6
ciag_znakow = "To jest ciag znakow"
ciag_znakow2 = 'To tez jest ciag znakow'
```

Należy zauważyć, że stringi w Pythonie możemy wpisywać używając cudzysłowu (”) lub apostrofu (’).

Wcięcia, wcięcia jeszcze raz wcięcia

Przed omówieniem pętli i funkcji warto zwrócić uwagę na to, że w Pythonie nie używamy średników ani klamerek otwierających i zamykających bloki kodu { }. Zamiast tego używamy odpowiedniej indentacji. W ten sposób mając np. dwie zagnieżdżone pętle zawartość pierwszej będzie wcięta jednym tabem, natomiast zawartość drugiej dwoma tabami. Za chwilę zobaczycie o co chodzi na realnym przykładzie :)

Pętla for

Pętla for w Pythonie jest w zasadzie pętlą foreach znaną z innych języków. Iteruje ona po danym zbiorze, a nie według warunku pętli.

Dla zademonstrowania o co chodzi użyjemy funkcji range(x), która zwraca nam zbiór liczb całkowitych od 0 do x-1.

```
>>> range(5)
[0, 1, 2, 3, 4]
```

Tworząc pętlę for iterującą zmienną x od 0 do 4 w języku Python wpiszemy następującą formułkę:

```
for x in range(5):
  print x
```

Jak widać linijka `print x` jest wcięta jedną tabulacją.

Gdybyśmy chcieli zagnieździć w sobie dwie pętle for ciało drugiej pętli musiałoby być wcięte o kolejny tabulator:

```
output = ""
for x in range(3):
  for y in range(4):
    output += str(x*y) + " "
  print output
  output = ""
```

W ten sposób na wyjściu uzyskamy

```
0 0 0 0
0 1 2 3
0 2 4 6
```

Funkcja range może przyjmować także więcej parametrów. Jeżeli przekażemy jej dwa argumenty x oraz y to zwróci zbiór liczb całkowitych od x do y-1.

```
>>> range(1,5)
[1, 2, 3, 4]
```

W przypadku wywołania range(x, y, z) otrzymamy zbiór od x do y-1 z krokiem do z.

```
>>> range(1, 10, 2)
```

```
[1, 3, 5, 7, 9]
```
Z racji tego, że for jest foreachem i może przyjmować dowolny zbiór spróbujmy takiej sztuczki:
```
zbior = [1,1,2,3,5,8,13]
output = ""
for x in zbior:
  output += str(x) + " "
print output
```
Jak się zapewne domyślacie na wyjściu otrzymamy
```
1 1 2 3 5 8 13
```
Tak jak w przypadku innych języków z pętli możemy wyjść przed jej planowanym końcem słówkiem kluczowym `break`, oraz przejść do kolejnej iteracji słowem `continue`.
Ponadto możemy użyć klauzuli `else` na końcu pętli. Wykona się ona w przypadku kiedy zbiór po którym pętla iteruje się wyczerpie, ale nie kiedy pętla została przerwana przez `break`.
Przykładowo wykonując kod
```
for x in range(5):
  print x
  break
else:
  print "Koniec zbioru"
```
Program wypisze nam na wyjście tylko 0. Jeżeli jednak zakomentujemy linijkę ze słowem break to na wyjściu otrzymamy
```
0
1
2
3
4
Koniec zbioru
```
Instrukcja warunkowa
Chcąc sprawdzić jakiś warunek wystarczy, że napiszemy
```
x = 10
if x > 5:
  print "x jest wiekszy od 5"
```
i już :) Jeżeli jednak chcemy nieco rozbudować naszą instrukcję warunkową możemy użyć klauzul `elif` oraz `else`.
```
x = 10
if x > 5:
  print "x jest wiekszy od 5"
elif x < 5:
  print "x jest mniejszy od 5"
else:
  print "x jest rowny 5"
```
Pętla while
Pętla while także może zawierać klauzulę `else`. Wykona się ona w przypadku kiedy warunek pętli będzie fałszywy, nie wykona się natomiast kiedy sami przerwiemy pętlę.
```
x = 5
while x < 10:
  print x
  x += 1
  break
else:
  print "Warunek petli falszywy"
```
Wynik: 5
Kiedy zakomentujemy break na wyjściu otrzymamy
```
5
6
7
8
9
Warunek petli falszywy
```

Funkcje

Funkcję możemy utworzyć według następującej składni:

```
def nazwa_funkcji(argument1, argument2):
```

Przykładowo funkcja wypisująca na wyjście wynik mnożenia liczb a i b, zwraca ten wynik w postaci liczby oraz jej wywołanie:

```
def calculate(a, b):
  result = a * b
  print str(a) + " x " + str(b) + " = " + str(wynik)
  return wynik

calculate(2,6)
result = calculate(2,6)
print result
```

Wynik tych operacji to:

```
2 x 6 = 12
```

Plik wykonywalny pythona

Aby używac Pythona nie musimy za każdym razem uruchamiać interpretera i wpisywać w nim ręcznie wszystkich poleceń. Możemy utworzyć plik z rozszerzeniem .py oraz nadać mu prawa do wykonywania, a następnie wykonać za pomocą komendy

```
python nazwa_skryptu
```

Przykład:

```
pi8@raspberrypi8:~ $ echo "print 'Hello World'" > hello_world.py
pi8@raspberrypi8:~ $ chmod +x hello_world.py
pi8@raspberrypi8:~ $ python hello_world.py
Hello World
```

Sterowanie GPIO w RaspberryPi

Stosując podstawy pythona możemy wysterować porty wejścia wyjścia w malince.
Pierwsze co musimy zrobić w naszym pliku ze skryptem to zaimportowanie biblioteki do sterowania portami GPIO. W tym celu na początku piszemy

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

Dzięki temu będziemy mogli się dostać do metod klasy RPi.GPIO odwołując się do nich przez GPIO.nazwa_metody.
Następnie musimy ustalić tryb w jakim będziemy odwoływali się do pinów. Może to być:

- GPIO.BOARD – odwołanie bezpośrednio do numeru pinu w złączu, zalecam ten tryb ze względu na to, że jest uniwersalny dla wszystkich malinek.
- GPIO.BCM – odwołanie do numeru pinu na procesorze – może być różne w różnych wersjach Raspberry.

tryb ustawiamy za pomocą `GPIO.setmode(GPIO.BOARD)`
Teraz możemy wybrać któryś z pinów i ustalić czy będzie wejściem, czy wyjściem (odpowiednik funkcji pinMode() z Arduino).

```
przycisk = 36
dioda = 40

GPIO.setup(przycisk, GPIO.IN) #pin 36 bedzie wejsciem
GPIO.setup(dioda, GPIO.OUT) #pin 40 bedzie wyjsciem
```

Możemy także ustawić pin jako wejście z rezystorem podciągającym lub ściągającym. WYstarczy wywołać

```
GPIO.setup(numer_pinu, GPIO.IN, pull_up_down=GPIO.PUD_UP) # Rezystor
podciagajacy do +3V3
  # lub
GPIO.setup(numer_pinu, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Rezystor
sciagajacy do masy
```

Stan pinu wyjściowego możemy ustalić przy pomocy

```
GPIO.output(numer_pinu, GPIO.HIGH) # stan wysoki
    # lub
GPIO.output(numer_pinu, GPIO.LOW) # stan niski
```

Natomiast odczytujemy w następujący sposób:
`GPIO.input(numer_pinu)`
Wynik tej metody możemy porównać do GPIO.HIGH lub GPIO.LOW.
Po wszystkim możemy po sobie posprzątać resetując ustawienia GPIO do domyślnych za pomocą
`GPIO.cleanup()`
opcjonalnie podając numer pinu, który chcemy przywrócić do stanu domyślnego. Jeżeli tego nie zrobimy wyzerowane zostaną wszystkie piny.

Przykładowy program

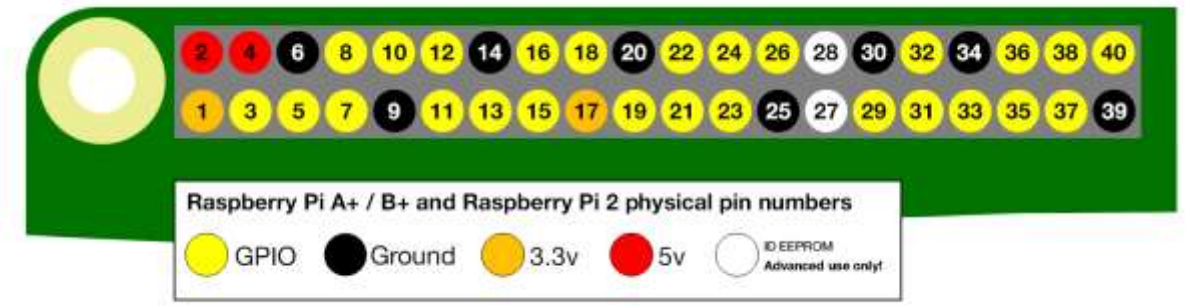
Dla zademonstrowania działania portów GPIO napisano program, który sprawdzi czy przycisk jest wciśnięty. Jeżeli tak to zaświeci diodę i będzie czekał na puszczenie przycisku. Wtedy zgasi diodę i zakończy działanie. Jeżeli przycisk nie będzie wciśnięty to tylko o tym poinformuje i natychmiast zakończy działanie.

```
import RPi.GPIO as GPIO
import time

ledPin = 40
btnPin = 5

GPIO.setmode(GPIO.BOARD)

GPIO.setup(ledPin, GPIO.OUT)
GPIO.setup(btnPin, GPIO.IN)

btnState = GPIO.input(btnPin)

if btnState:
        print 'Przycisk nie jest wcisniety'
else:
        print 'Przycisk jest wcisniety'

GPIO.output(ledPin, not btnState)

while GPIO.input(btnPin) == GPIO.LOW:
        pass

GPIO.cleanup()
print 'Koniec programu'
```

Przycisk podłączyłem do pinu 5 podciągając go rezystorem do pinu 3.3V. Dioda LED podłączona jest do pinu numer 40.

Należy pamiętać, aby pod żadnym pozorem nie podłączyć do żadnego pinu Raspberry napięcia 5V. Natychmiast zabije to dany pin.

PROTIP: Piny 5V sąsiadują ze zobą, więc żeby przypadkiem nie podłączyć tam nic tworząc połączenia można założyć na nie zworkę i zdejmować ją tylko wtedy, gdy świadomie potrzebujemy użyć napięcia 5V. Na poniższej grafice przedstawiono numery pinów zgodnie z trybem GPIO.BOARD:



Wybrane polecenia LINUX:

sudo	realizacja poleceń jako administrator
sudo raspi-config	wywołanie zbioru konfiguracyjnego
ssh użytkownik@host	zdalny dostep do linux przez sieć np. z Windows (wiersz poleceń cmd)
ssh pi1@raspberry pi1	przykład lub po numerze IP: np. 192.168.1.34
cmd /k ipconfig	lista adresów IP w danej sieci lokalnej z Windows (wiersz poleceń cmd) lub
arp –a	lista adresów IP w danej sieci lokalnej z Windows (wiersz poleceń cmd)

adresy MAC wszystkich Raspberry Pi rozpoczynają się od ciągu **b8-27-eb**

passwd	zmiana hasła dostępu
apt	menager programów
sudo apt update	informacja o najnowszych wersjach programów
sudo apt upgrade	instalacja nowej wersji
man	dokumentacja danego polecenia
man apt	dokumentacja systemowa menagera programów
top	menager programów
htop	nowszy menager rogramów z podglądem rdzeni procesora
sudo apt install htop	instalacja
vcgencmd measure_temp	sprawdzenie temperatury procesora
nano	edytor tekstu
cp	kopiowanie pliku
mv	przesuniecie pliku
rm	usuwanie pliku
cd	zmiana katalogu
ls	lista plików
mkdir	nowy katalog
rmdir	usuwanie katalogu

Instrukcja do instalacji Jupiter

```
ssh pi@raspberrypi
#skopiuj katalog jupyter do Home Folder czyli home/pi8

sudo raspi-config
#enable VNC, I2C, serial - nie logowanie przez serial

sudo apt update
sudo apt upgrade

pip install opencv-python
sudo systemctl enable pigpiod    #To                pip install pigpio


### Jupyter
sudo apt-get install python3-matplotlib
sudo apt-get install python3-scipy
pip3 install --upgrade pip
pip install jsonschema'[format-nongpl]'
pip3 install matplotlib
curl https://sh.rustup.rs -sSf | sh
sudo reboot
sudo pip3 install jupyter
pip3 install jupyterlab

sudo mv /home/pi8/jupyter/jupyter.service /etc/systemd/system
sudo systemctl enable jupyter

### AI
sudo nano /etc/dphys-swapfile #set 1024 swap
sudo apt install python3-dev python3-pip
sudo apt install libatlas-base-dev
pip install tflite-support
pip install  tensorflow_io
pip install uuid
pip install --upgrade tensorflow  --no-cache-dir
```

Kody przycisków na pilocie do RaspberryPi

Przycisk	Decimal	Hexadecimal
1	12	0x0c
2	24	0x18
3	94	0x5e
4	8	0x08
5	28	0x1c
6	90	0x5a
7	66	0x42
8	82	0x52
9	74	0x4a
0	22	0x16
-	7	0x07
+	21	0x15

100+	25	0x19
200+	13	0x0d
CH-	69	0x45
CH	70	0x46
CH+	71	0x47
wstecz	68	0x44
wprzód	64	0x40
play	67	0x43
EQ	9	0x09