

Raspberry Pi



-electronic work for controlling a robot-

Part #1

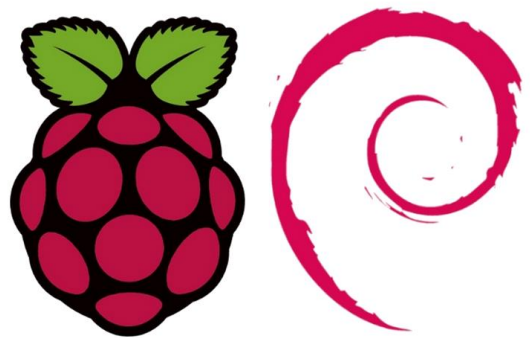
2019. 08. 01. (Thur.)

Dai Owaki, Ph. D,

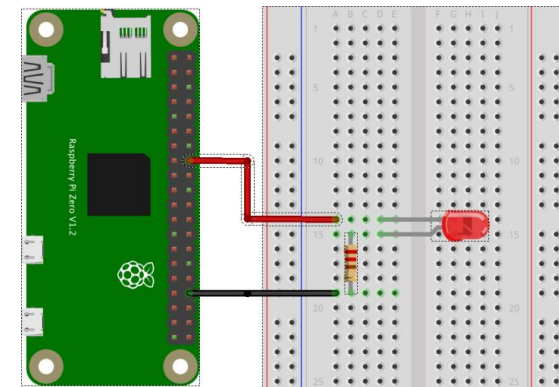
Asocci. Prof. Neuro-robotics Lab. (Hayashibe Lab.),
Dept. of Robotics, Graduate School of Engineering,
Tohoku University

Presenter:

Hannan Ahmed, Master's 1st Year

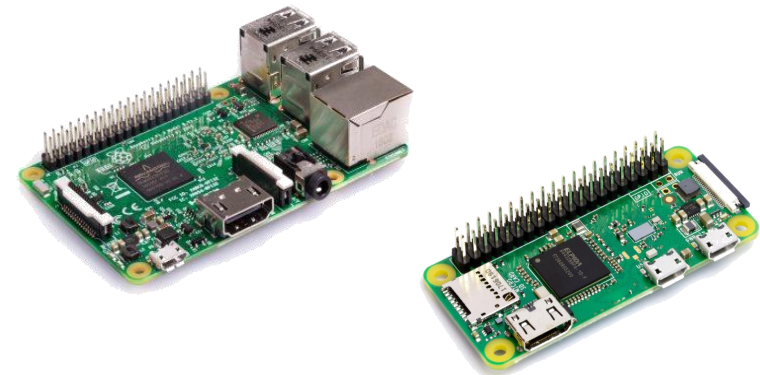


Raspbian



Let's Enjoy Electronic Work with *Raspberry Pi*

- ✓ *Electronic work* is **good education topics** for both making things (e.g. robot) and coding software (e.g. using C++, python)
- ✓ *Raspberry Pi* is a good tool for this, because it's **cheap** (cheapest 5USD), **high spec** (lowest CPU around 1GHz), installed a **Linux OS** (Raspbian), **control electronic devices** (GPIO pins), **wireless** (wifi & Bluetooth), easy to use **cameras** (take pics & movies), and **much information on the web communities!!!**
- ✓ The point is “***enjoy the electronic work***”



Spec of Raspberry Pi

	Raspberry Pi 2 Model B	Raspberry Pi 3 Model B	Raspberry Pi Zero	Raspberry Pi Zero W
CPU	ARM Cortex-A7 (900MHz) クアッドコア	ARM Cortex-A53 (1.2GHz) クアッドコア	ARM1176 ZF-S (1GHz) シングルコア	
SoC	Broadcom BCM2836	Broadcom BCM2837	Broadcom BCM2835	
グラフィック	Broadcom VideoCore IV (250MHz)			
メモリー	1Gバイト		512Mバイト	
USB インタフェース	USB 2.0×4ポート		USB 2.0 (microUSB) ×1ポート	
ビデオ出力	HDMI、コンポジット (NTSC, PAL)、DSI		Mini-HDMI、コンポジット (GPIO経由)	
ビデオ入力	CSI		CSI (小型タイプ)	
オーディオ出力	φ3.5mmジャック、HDMI、I2S		HDMI、GPIOから出力可能	
ストレージ用スロット	microSDカードスロット			
ネットワーク	10/100Mbps Ethernet	10/100Mbps Ethernet、 IEEE802.11b/g/n	無し	IEEE802.11b/g/n
Bluetooth	無し	Bluetooth 4.1、 Bluetooth Low Energy	無し	Bluetooth 4.1、 Bluetooth Low Energy
その他インタフェース	GPIO、UART、I ² C、SPI			
電源出力端子	3.3V、5V			
電源電圧／電力定格	5V / 3W (600mA)	5V / 4.5W (800mA)	5V / 0.8W (160mA)	
サイズ	85.6mm×56.5mm		65mm×30mm	
重さ	45g		9g	

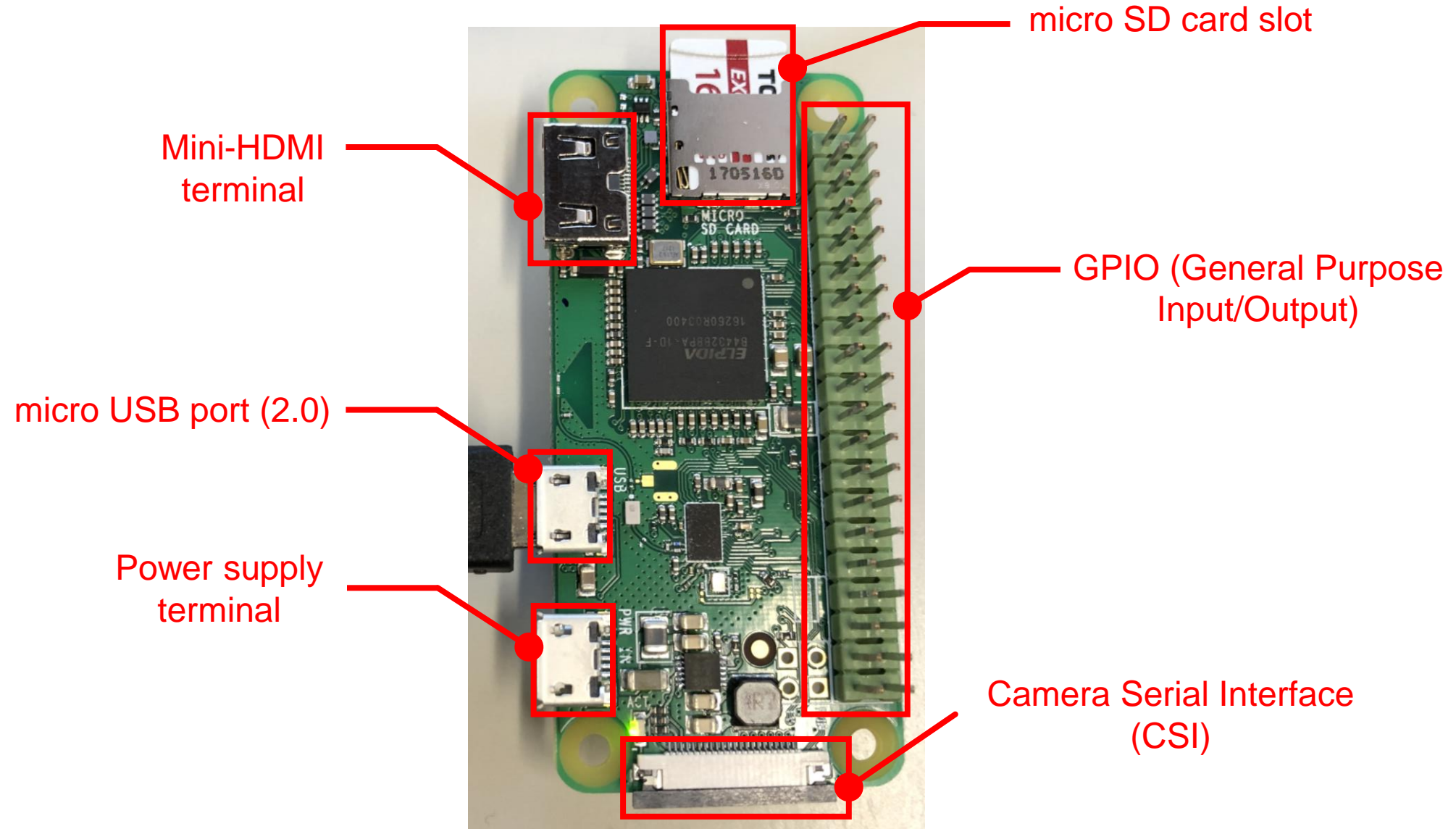
35USD

35USD

5USD

10USD

Hardware Configurations on Raspberry Pi Zero W



What do you need to begin *Raspberry Pi*?

Required

- Raspberry Pi (3 or Zero W are recommended)
- Micro SD card (more than 4GB (16GB&class 10 is better))
- Micro-USB cable (for Zero W) or LAN-USB cable for (3):
Data communication cable (USB OTG: On-the-Go cable)
not only for power supply



Optional

- Power 5V
- Display (HDMI input)
- Keyboard (USB or Bluetooth)
- Mouse (USB or Bluetooth)



One of purposes of today's lecture is to use raspberry pi without these annoying things!

Step 1: Connecting to RasPi via SSH

```
$ssh pi@raspberrypi.local
```

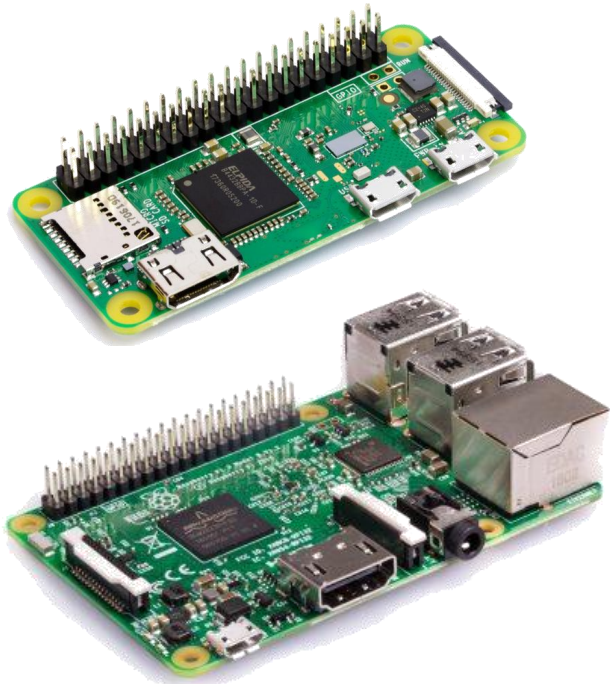
pi = username

raspberrypi = local hostname



SSH = Secure Shell

-> Protocol for connection to remote computer



Install Software in Your PC

Tera Term

OSDN > ソフトウェアを探す > 端末 > シリアル > Tera Term > 概要

Tera Term

概要 ▾ [ダウンロード](#) [ソースコード](#) ▾ [チケット](#) ▾ [文書](#) ▾ [コミュニケーション](#) ▾ [ニュース](#)

[ス] 天然ダイヤモンドとの区別が困難な人工ダイヤモンドが増えている

プロジェクトの説明

[レビューする](#)

[Webページ](#)

[開発情報](#)



Tera Term は、オリジナルの [Tera Term Pro 2.3](#) の原作者公認の後継版です。オープンソースで開発されており、UTF-8 表示に対応しています。また、SSH1 対応モジュール TTSSH を拡張し、SSH2 プロトコルをサポートしています。

[バグを報告する](#)

[文書を見る](#)

[フォーラムで情報交換](#)

[RSSを取得](#)

[画像一覧](#)

インストール

ダウンロードが完了したら、パッケージをクリック（もしくはダブルクリック）して実行する。するとインストールウィザードが起動するので、ウィザードの指示に従ってインストールする。なお、途中でインストール。

[more](#)

ダウンロード

VNC Viewer

VNC® Connect consists of VNC® Viewer and VNC® Server

Download VNC® Viewer to the device you want to control from, below. Make sure you've [installed VNC® Server](#) on the computer you want to control.



Windows



macOS



Linux



Raspberry Pi



iOS



Android



Chrome



Solaris



HP-UX



AIX

[Download VNC Viewer](#)

SHA-256: 6cb4fdeeabb779f6cefacc867618299bec152673a90fe52ac8211c48d7074bb03

EXE x86/x64

[Looking for VNC® Server?](#)

Raspbain OS has been installed. Let's Boot!!

Overview of Raspberry Pi Zero W



Just connecting to USB port!
Power is supplied via the port



SSH connection to Raspi via Tera Term

hostname = raspberrypi.local (initial)

username = pi
pass = raspberry (initial)

Success connection (PC to raspi) !!!

Checking IP Address of Raspberry Pi

```
raspberrypi.local - pi@raspberrypi: ~ VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
pi@raspberrypi:~$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.137.177 netmask 255.255.255.0 broadcast 192.168.137.255
    inet6 fe80::726c:9cc9:fc25:ba62 prefixlen 64 scopeid 0x20<link>
    ether 02:22:82:ff:ff:11 txqueuelen 1000 (Ethernet)
    RX packets 510 bytes 104905 (102.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 187 bytes 34895 (34.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b8:27:eb:4e:a4:59 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@raspberrypi:~$
```

\$ ifconfig

IP Address for the Internet connection via USB.

192.168.137.*** -> Success!!!

IP Address for the Internet connection via Wifi module on Raspberry pi.

Now, not connected via Wifi

Checking for the Internet Connection

\$ ping 8.8.8.8

```
pi@raspberrypi:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=53 time=8.17 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=53 time=8.97 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=53 time=8.87 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=53 time=8.47 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=53 time=8.84 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=53 time=7.79 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=53 time=14.4 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=53 time=8.68 ms
```

Successfully connecting the internet via PC

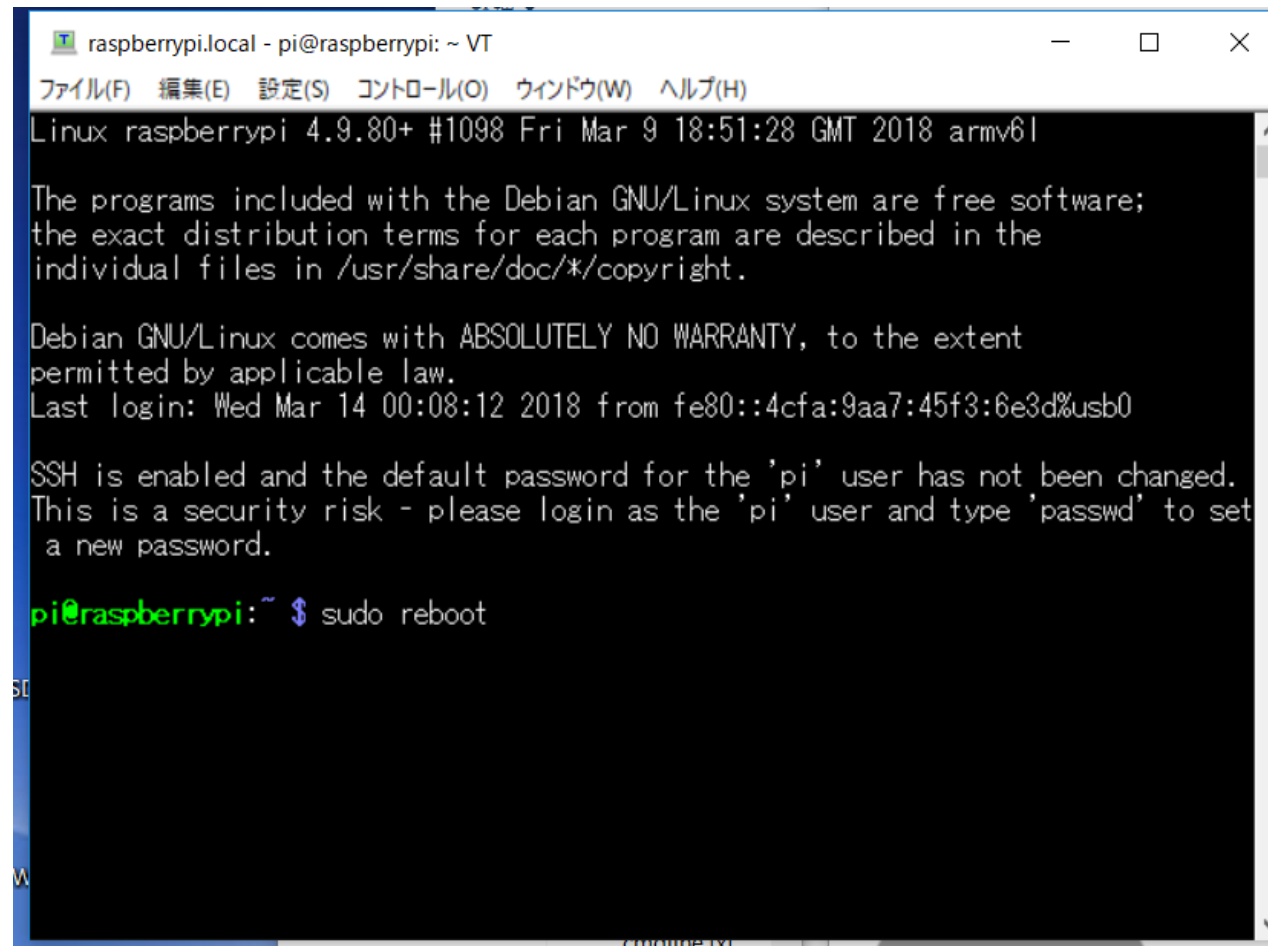


Reboot of Raspberry Pi via SSH

`$ sudo reboot`

-> disconnect Teratern and LED on Raspi will be blinking (rebooting)

"sudo" means execute a command with "administrative right"



```
raspberrypi.local - pi@raspberrypi: ~ VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
Linux raspberrypi 4.9.80+ #1098 Fri Mar 9 18:51:28 GMT 2018 armv6l

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: Wed Mar 14 00:08:12 2018 from fe80::4cfa:9aa7:45f3:6e3d%usb0

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:~ $ sudo reboot
```


Step 2: Connecting via VNC

VNC (Virtual Network Computing) = Remote control of raspberry pi (or some PC) via network

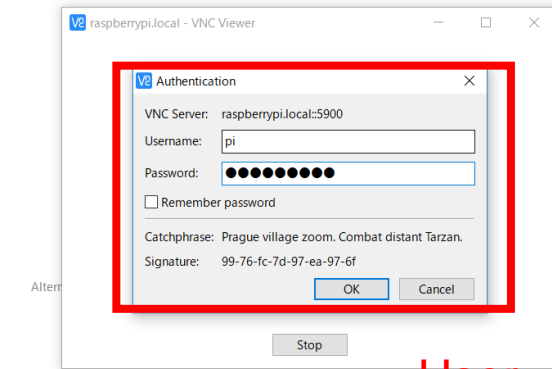
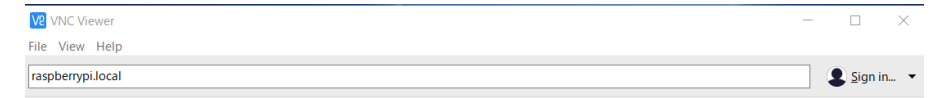
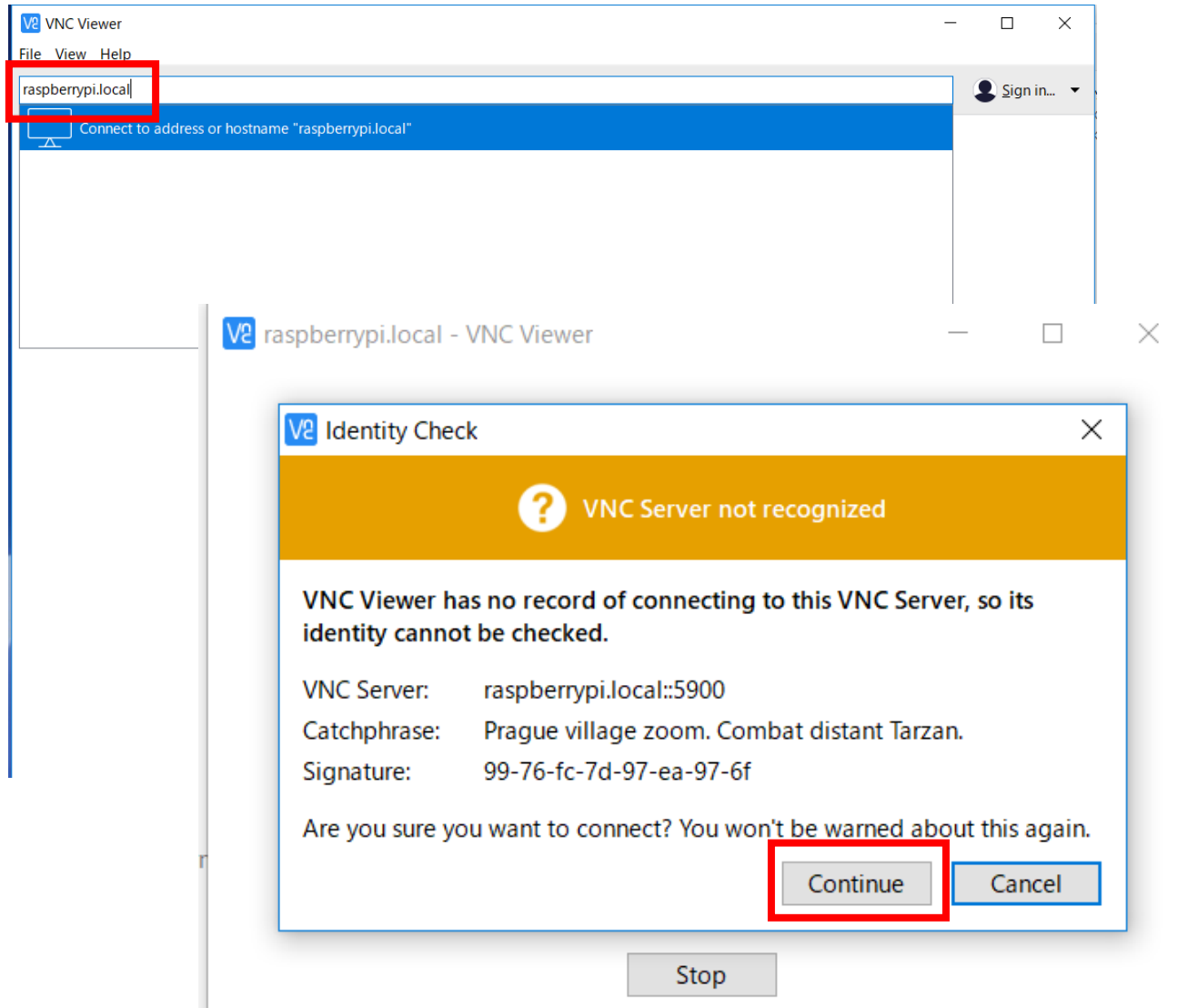
Server software on Raspi (host, already installed) and client software (viewer) on PC

-> you don't need display, keyboard, and mouse for Raspi, PC will become them via VNC

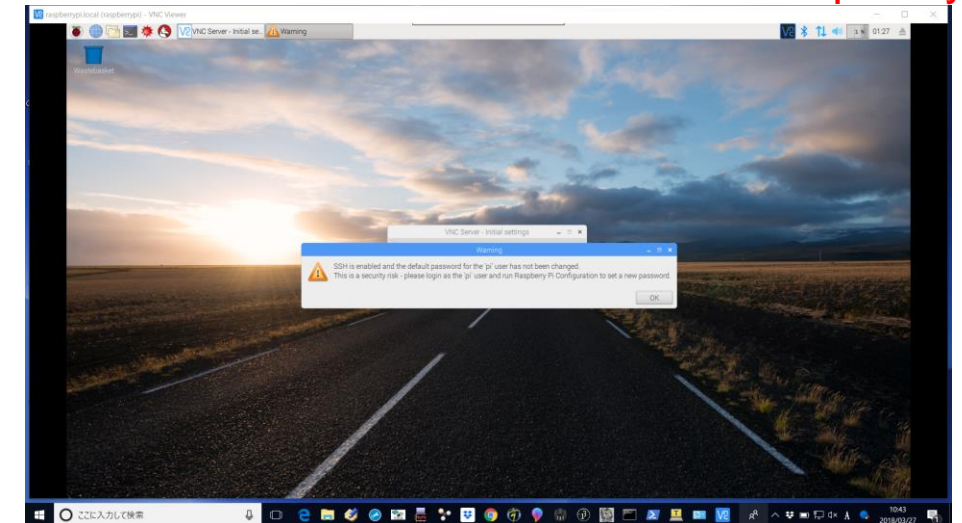


VNC Viewer on PC

raspberrypi.local (=hostname)



User = pi
Pass = raspberry



Success!!!, You can manage Raspi via PC

Connecting to Raspi's disk via Samba

The image consists of three screenshots illustrating the process of connecting to a Raspberry Pi's disk via Samba on a Windows machine.

Left Screenshot: A Windows File Explorer window showing the network location `oscillex` in the address bar. The address bar is highlighted with a red box, and the text `¥¥hostname` is written next to it. The left sidebar shows the network location `OSCILLEX` selected.

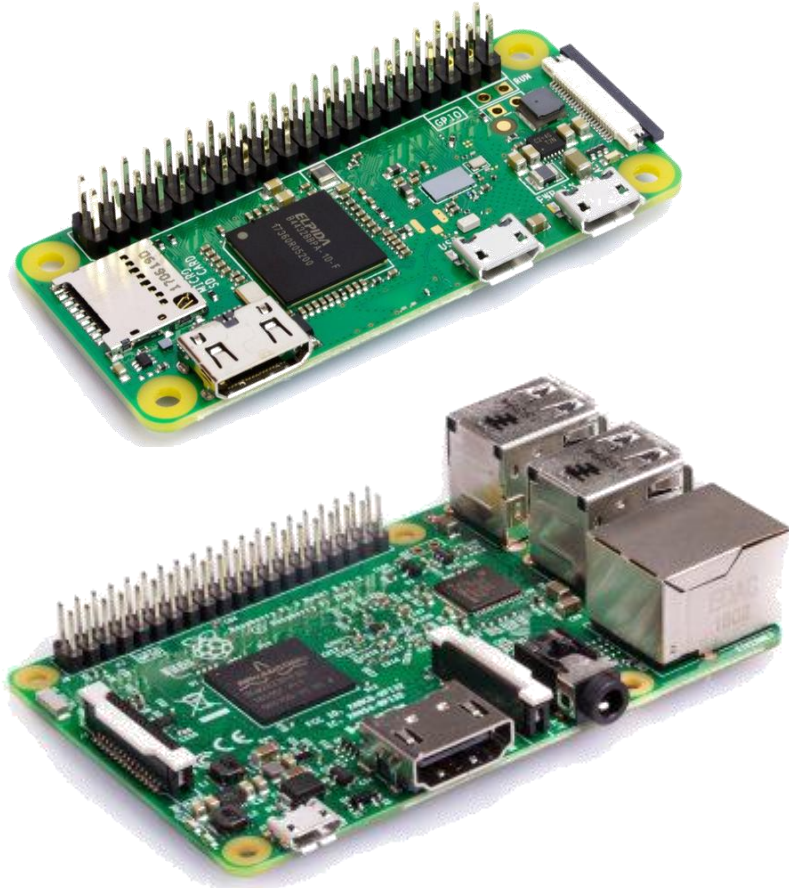
Middle Screenshot: A Windows File Explorer window showing the network location `oscillex` in the address bar. The address bar is highlighted with a red box, and the text `¥¥hostname` is written next to it. The left sidebar shows the network location `OSCILLEX` selected. A red box highlights the `pi` folder, and the text `"Double click"` is written next to it.

Right Screenshot: A Windows Security dialog box titled "ネットワーク資格情報の入力" (Enter network credentials). The dialog prompts for credentials to connect to `oscillex`. The username field contains `pi` and the password field contains a series of black dots. The text `user=pi` and `pass` are written next to the respective fields. The dialog also includes a checkbox for "資格情報を記憶する" (Remember credentials) and buttons for "OK" and "キャンセル" (Cancel).

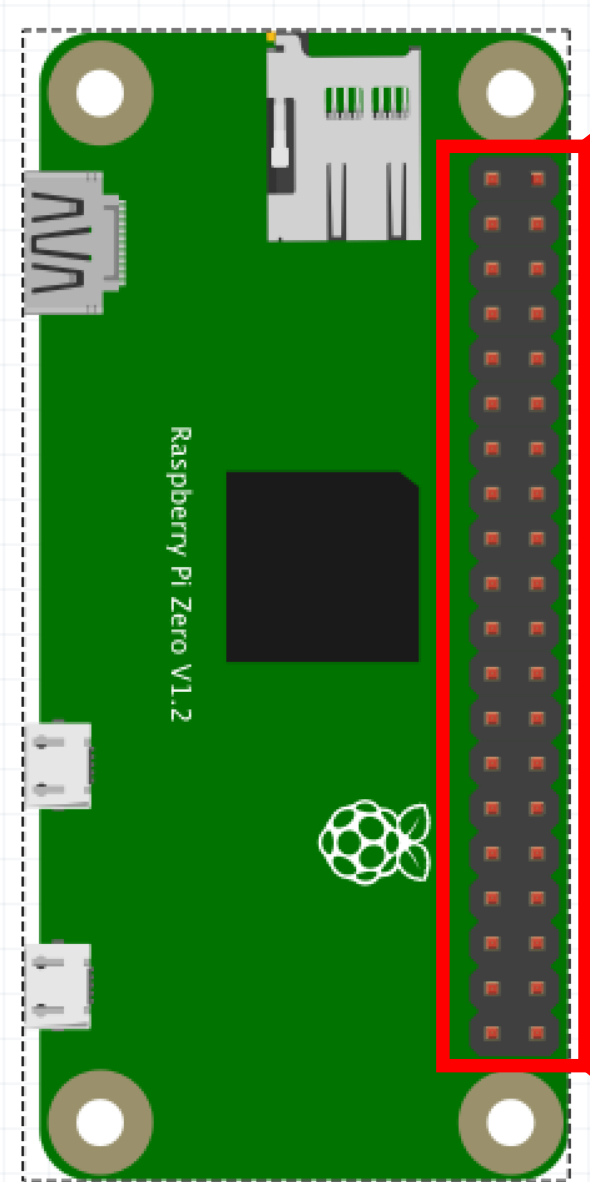
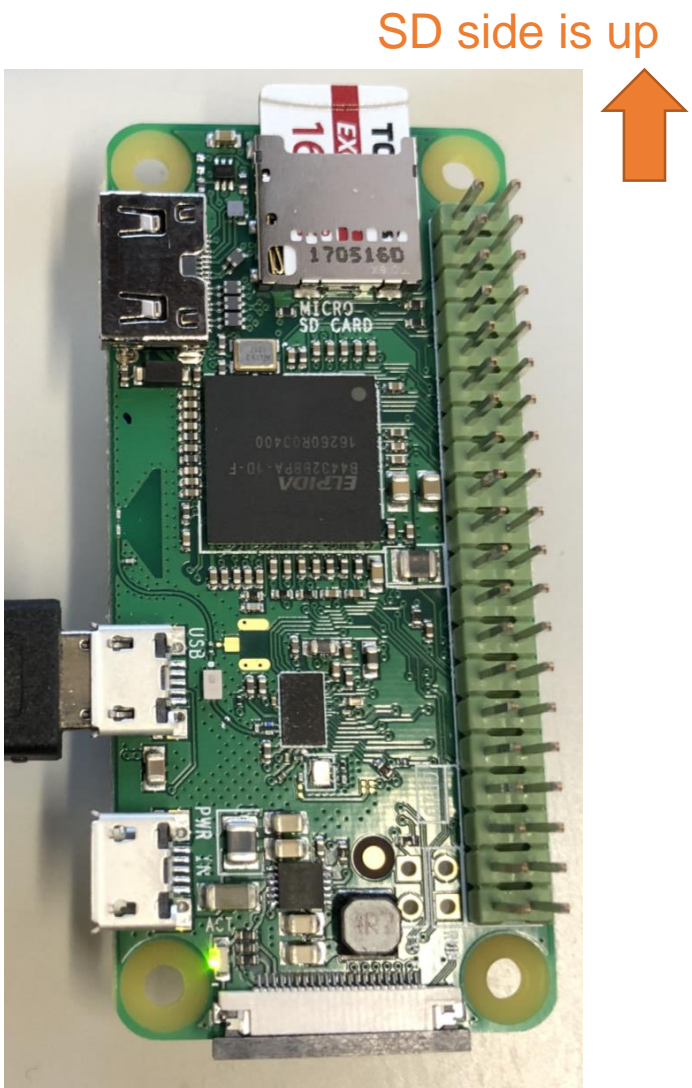
Bottom Right: The text `Success!!!` is written in red.

Power off Raspberry Pi

\$ sudo poweroff



I/O Interfaces on Raspberry PI



1	+3.3V		+5V	2
3	SDA	2	+5V	4
5	SCL	3	GND	6
7		4	14 TxD	8
9	GND		15 RxD	10
11		17	18	12
13		27	GND	14
15		22	23	16
17	+3.3V		24	18
19	MOSI	10	GND	20
21	MISO	9	25	22
23	SCLK	11	8 CE0	24
25	GND		7 CE1	26
27	ID_SD		ID_SC	28
29		5	GND	30
31		6	12	32
33		13	GND	34
35		19	16	36
37		26	20	38
39	GND		21	40

GPIO (General Purpose Input/Output)

+3.3V	1	2	+5V
SDA	2	3	+5V
SCL	3	5	GND
	4	7	14 TxD
GND		9	15 RxD
	17	11	18
	27	13	GND
	22	15	23
+3.3V		17	24
MOSI	10	19	GND
MISO	9	21	25
SCLK	11	23	8 CE0
GND		25	7 CE1
ID_SD		27	ID_SC
	5	29	GND
	6	31	12
	13	33	GND
	19	35	16
	26	37	20
GND		39	21

Power: +3.3V (1&17) and +5V(2 & 4) can use for power supply to electronic devices or input for circuit

GND (6,9,11,20,25,30,34,39): 0V output pins

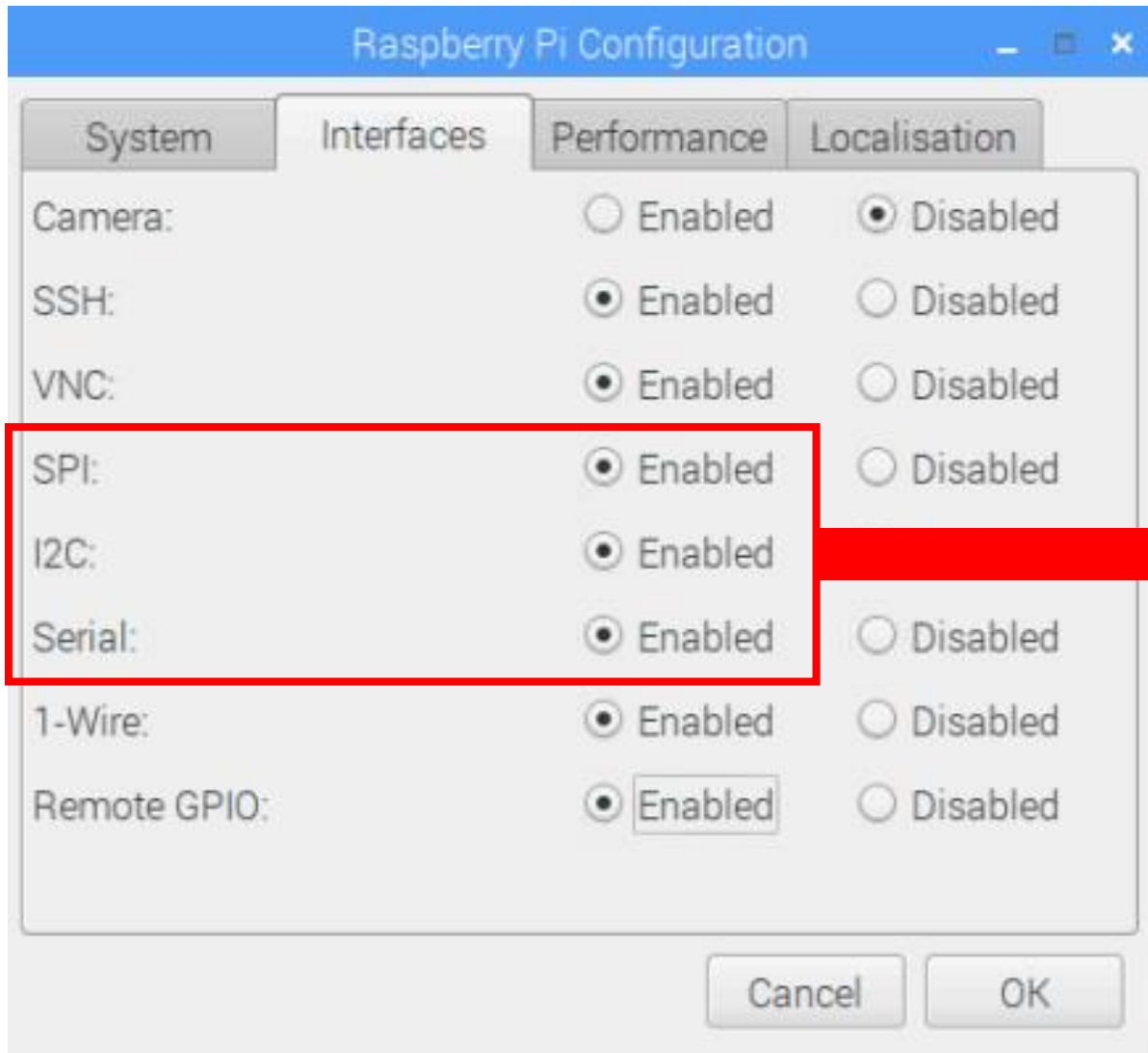
GPIO(white): General Purpose input and output pins (**3.3V** or **0 V**)

UART (Universal Asynchronous Receiver Transmitter):
2-wired (**TxD: Transmit**, **RxD: Receive**) Communication to PC or electronics devices

I2C (Inter-Integrated Circuit): Communication standard to electronics devices (motors and sensors). **SDA (3)** is for data transmission and reception. **SCL (5)** is for clock synchronization between devices.

SPI (Serial Peripheral Interface): Communication standard to electronics devices. **MOSI (19)**=data transmission, **MISO (21)**=data reception, **SCLK (23)**=synchronization between devices, **CE0(24)**, **CE1(26)** = port for selecting the target device

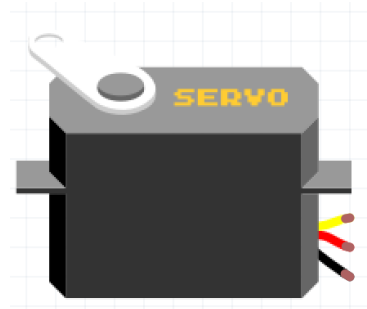
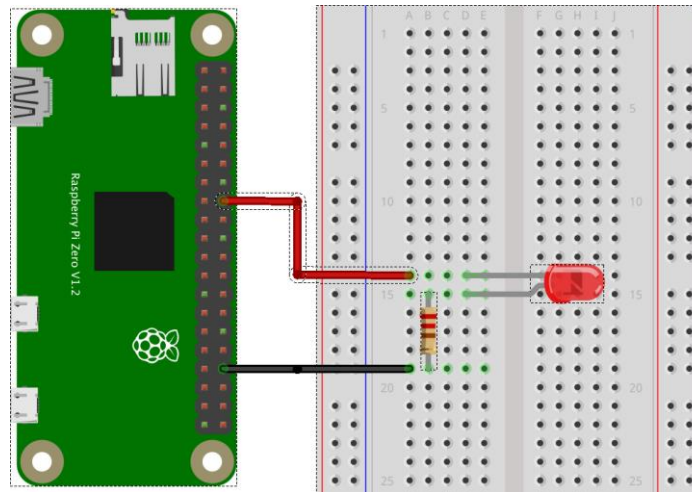
Pin Configuration



+3.3V		+5V	
SDA	2		+5V
SCL	3		GND
	4		14 TxD
GND			15 RxD
	17		18
	27		GND
	22		23
+3.3V			24
MOSI	10		GND
MISO	11		25
SCLK	11		8 CE0
GND			7 CE1
ID_SD			ID_SC
	5		GND
	6		12
	13		GND
	19		16
	26		20
GND			21

If you activate these interfaces, you cannot use these pins as GPIO (Digital I/O or PWM)

How to control LEDs and Motors



Libraries for Using GPIO with Python

- `raspberry-gpio-python`: **PRI.GPIO**, standard libraries for python
- **WiringPI**: High-performance lib., support for the use of I2C and SPI

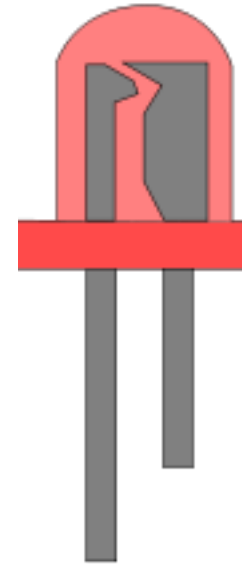
Installing WiringPI

```
$ sudo pip install wiringpi
```

Importing WiringPI module for python

```
import wiringpi (as pi)
```

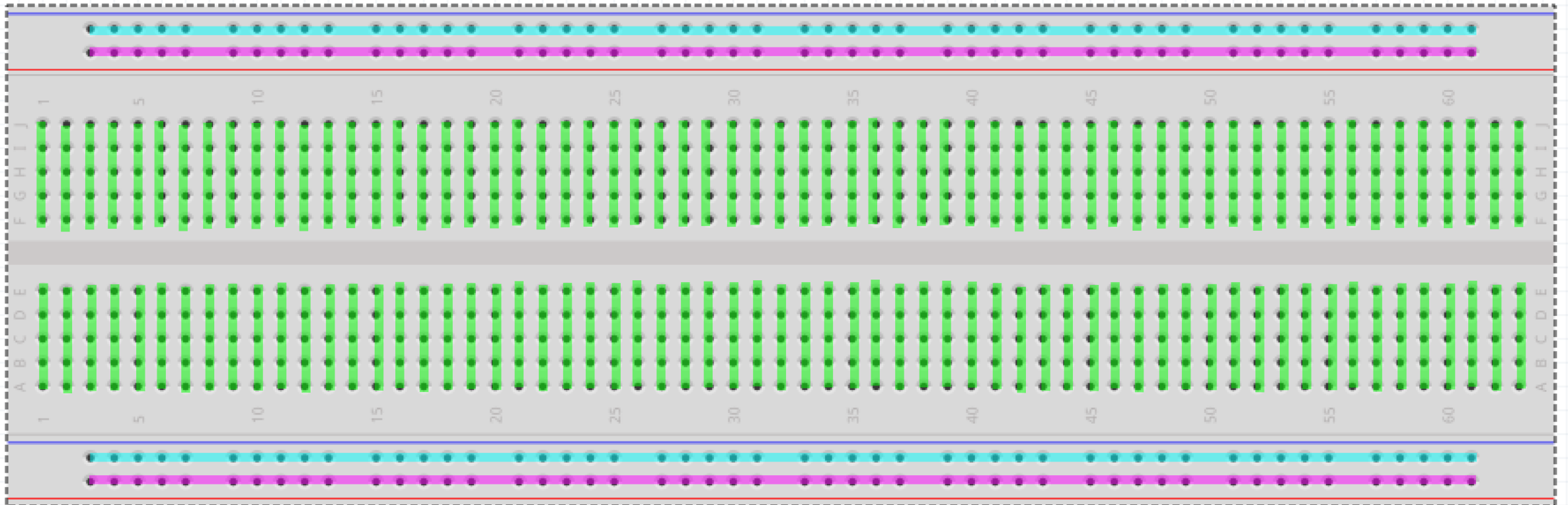
Which is Anode(+) or Cathode(-) on a LED?



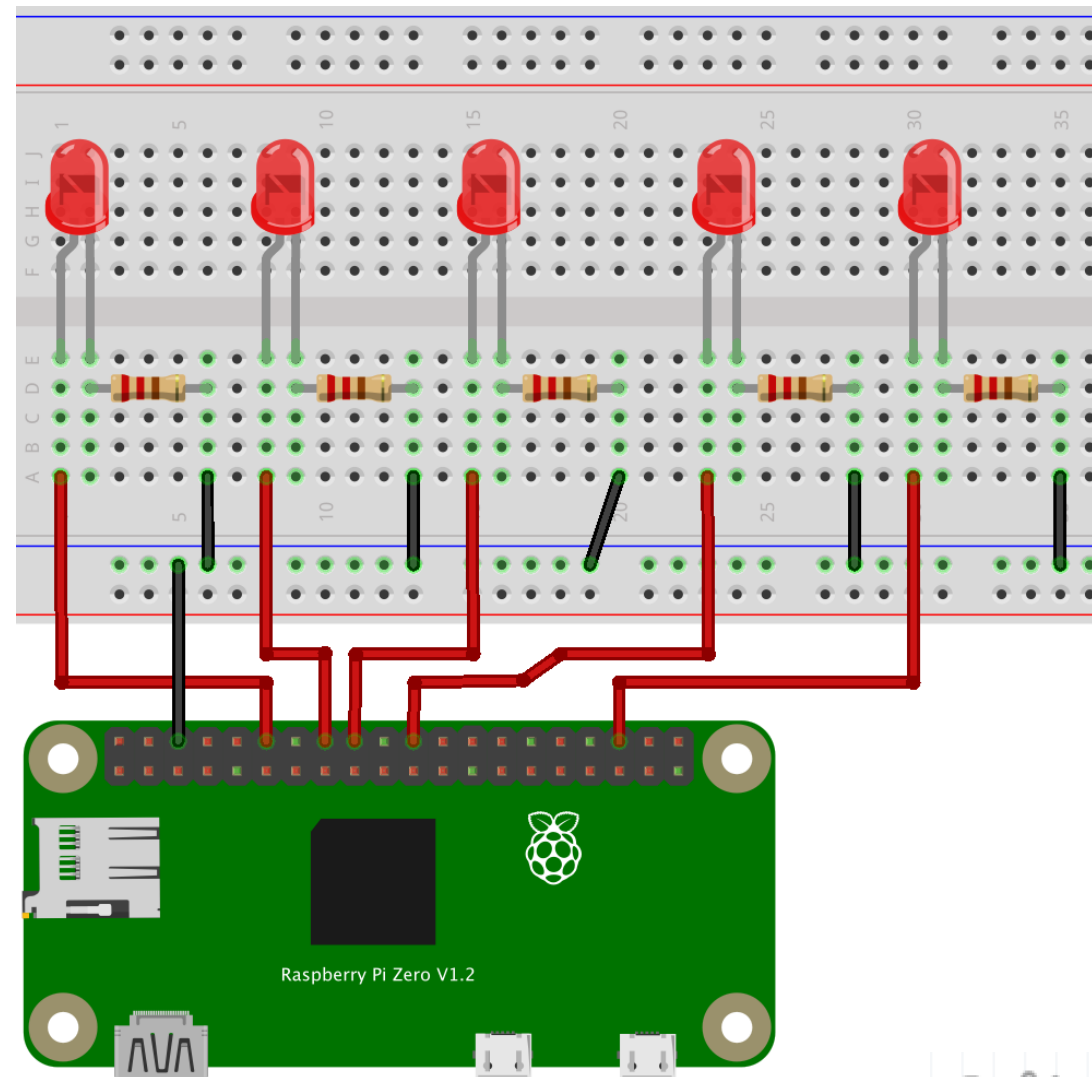
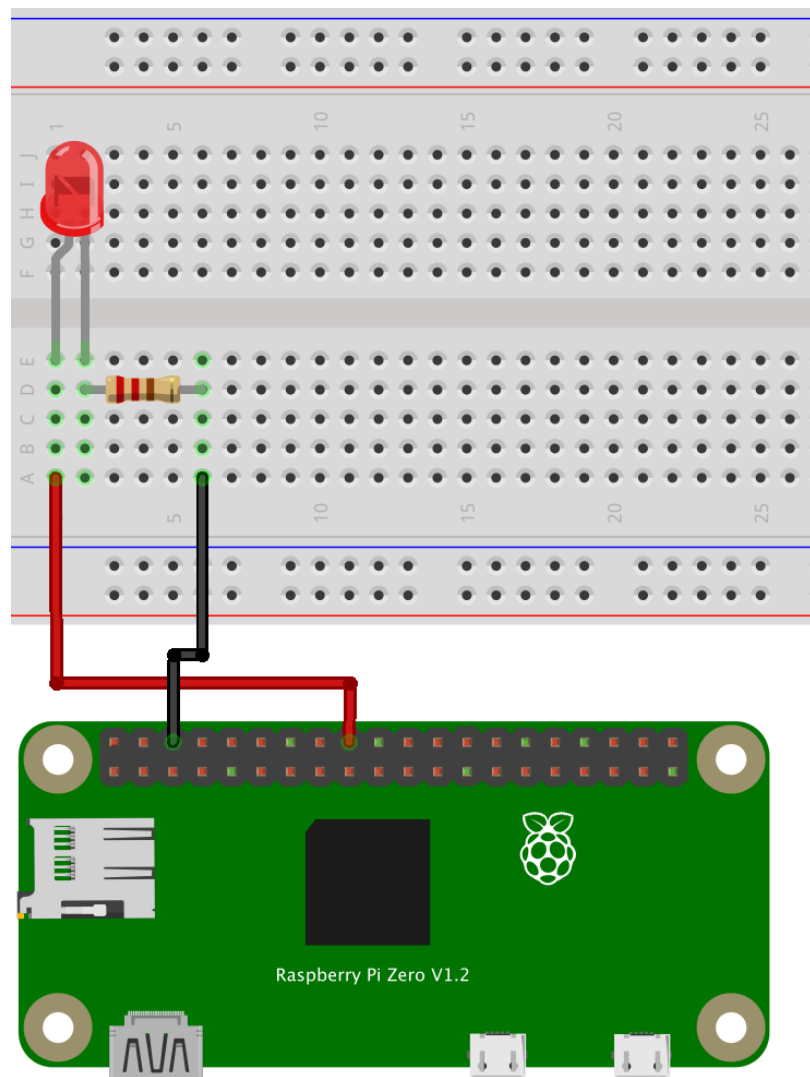
1. Can you find anode/cathode if you are given an LED without any information?
2. What if the wires were cut?

Tips: How to Use Breadboard

- ✓ Prototyping board to make a test electronic circuit
- ✓ For examples, **green lined holes** are electrically connected on the background
- ✓ **All red lined holes (e.g. for +V)** and **blue lined holes (e.g. for GND)** are connected



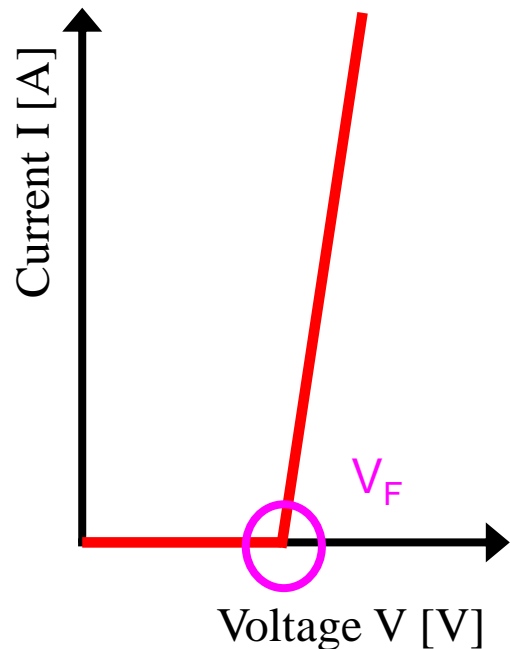
Blinking LEDs



How to Choose An Appropriate Resistor?

Data sheet from the LED used: $V_F = 2.2\text{ V}$, $I_F = 20\text{ mA}$

Symbol	Parameter	Device	Typ.	Max.	Units	Test Conditions
λ_{peak}	Peak Wavelength	Green	565		nm	$I_F=20\text{mA}$
λ_D	Dominate Wavelength	Green	568		nm	$I_F=20\text{mA}$
$\Delta\lambda_{1/2}$	Spectral Line Half-width	Green	30		nm	$I_F=20\text{mA}$
C	Capacitance	Green	15		pF	$V_F=0\text{V}; f=1\text{MHz}$
V_F	Forward Voltage	Green	2.2	2.5	V	$I_F=20\text{mA}$
I_R	Reverse Current	Green		10	μA	$V_R = 5\text{V}$

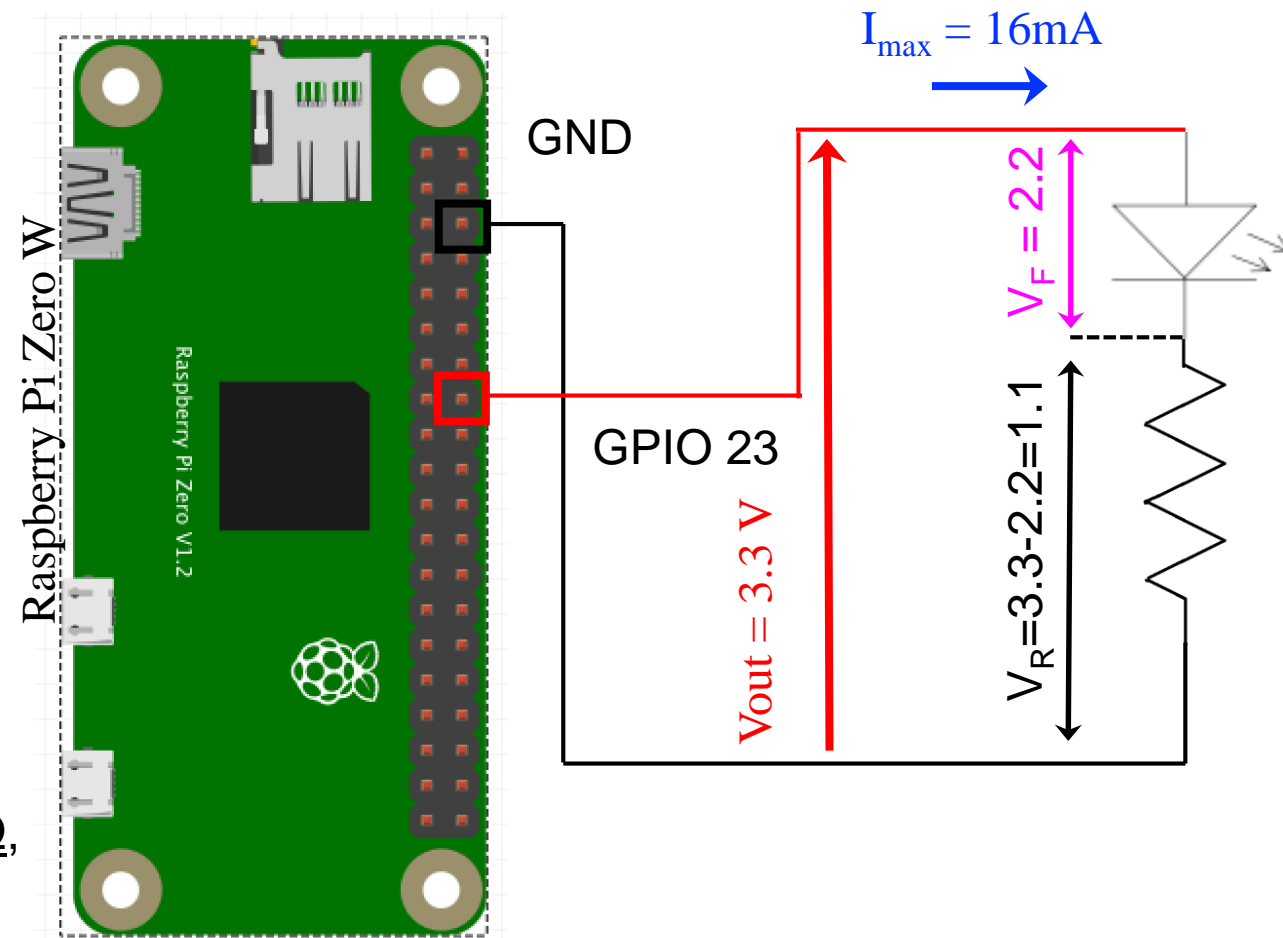


1. $V_R = 3.3 - 2.2 = 1.1\text{ V}$

2. GPIO output current for each pin is limited within 16mA !

3. $R > (1.1\text{V}) / (16\text{mA}) = \underline{68.75\ \Omega}$

For example, If you choose $100\ \Omega$, current will be $\underline{11\text{mA}}$. ($=1.1/100$).



Python Scripts for LED Blinking

```
blink_led.py x
1 import wiringpi as pi
2 import time
3
4 LED_PIN = 23
5
6 pi.wiringPiSetupGpio()
7 pi.pinMode( LED_PIN, pi.OUTPUT )
8
9 while True:
10     pi.digitalWrite( LED_PIN, pi.LOW )
11     time.sleep( 1 )
12
13     pi.digitalWrite( LED_PIN, pi.HIGH )
14     time.sleep( 1 )
15
```

#1: import wiringpi as pi (for GPIO)

#2: import time module (including stop method in seconds (s) that are defined)

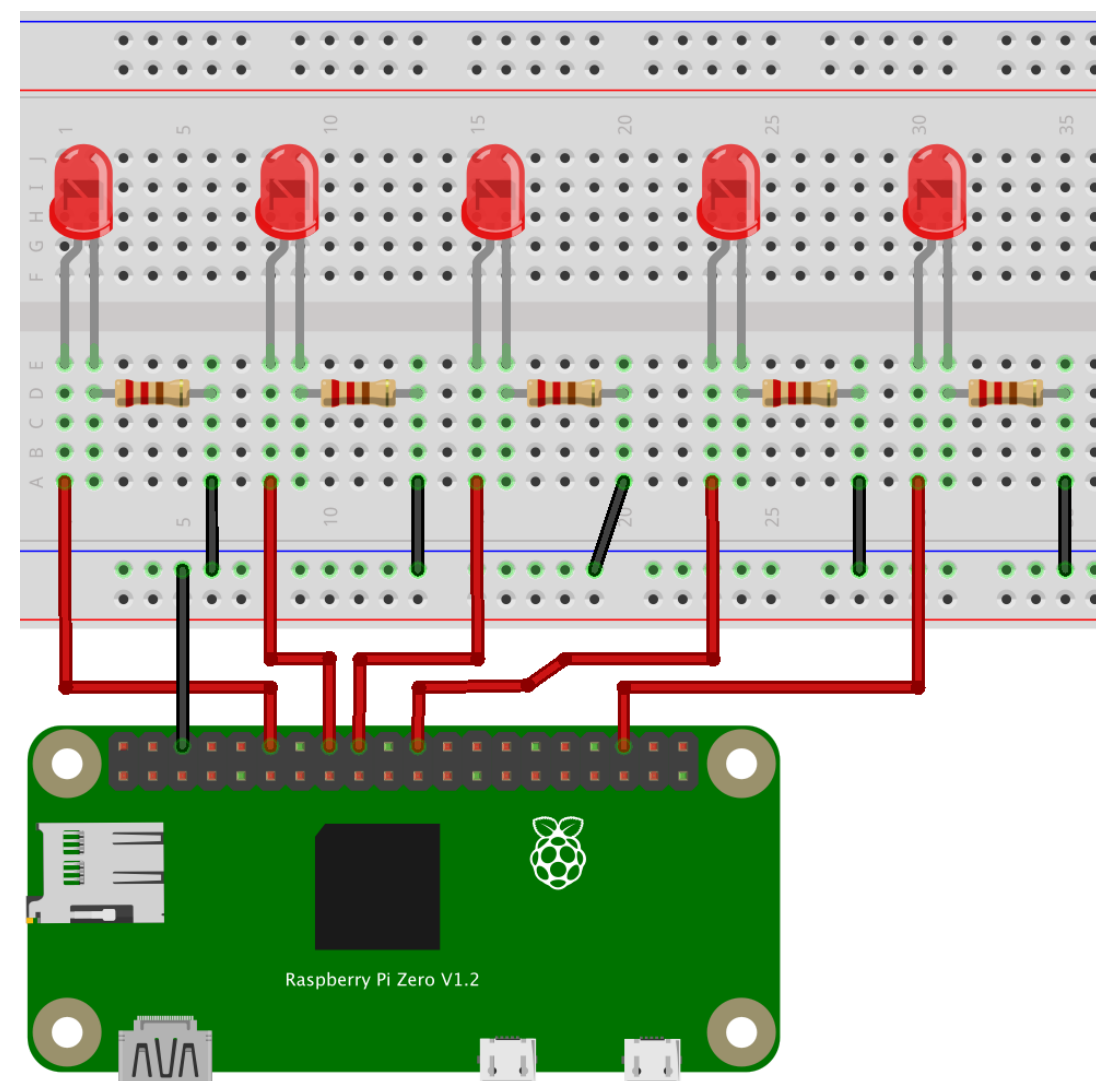
#4: Setting the led pin

#6&7: Setting the GPIO mode as "output"=pi.OUTPUT

#10 Output from LED_PIN, here, pi.LOW means 0V (=no output)

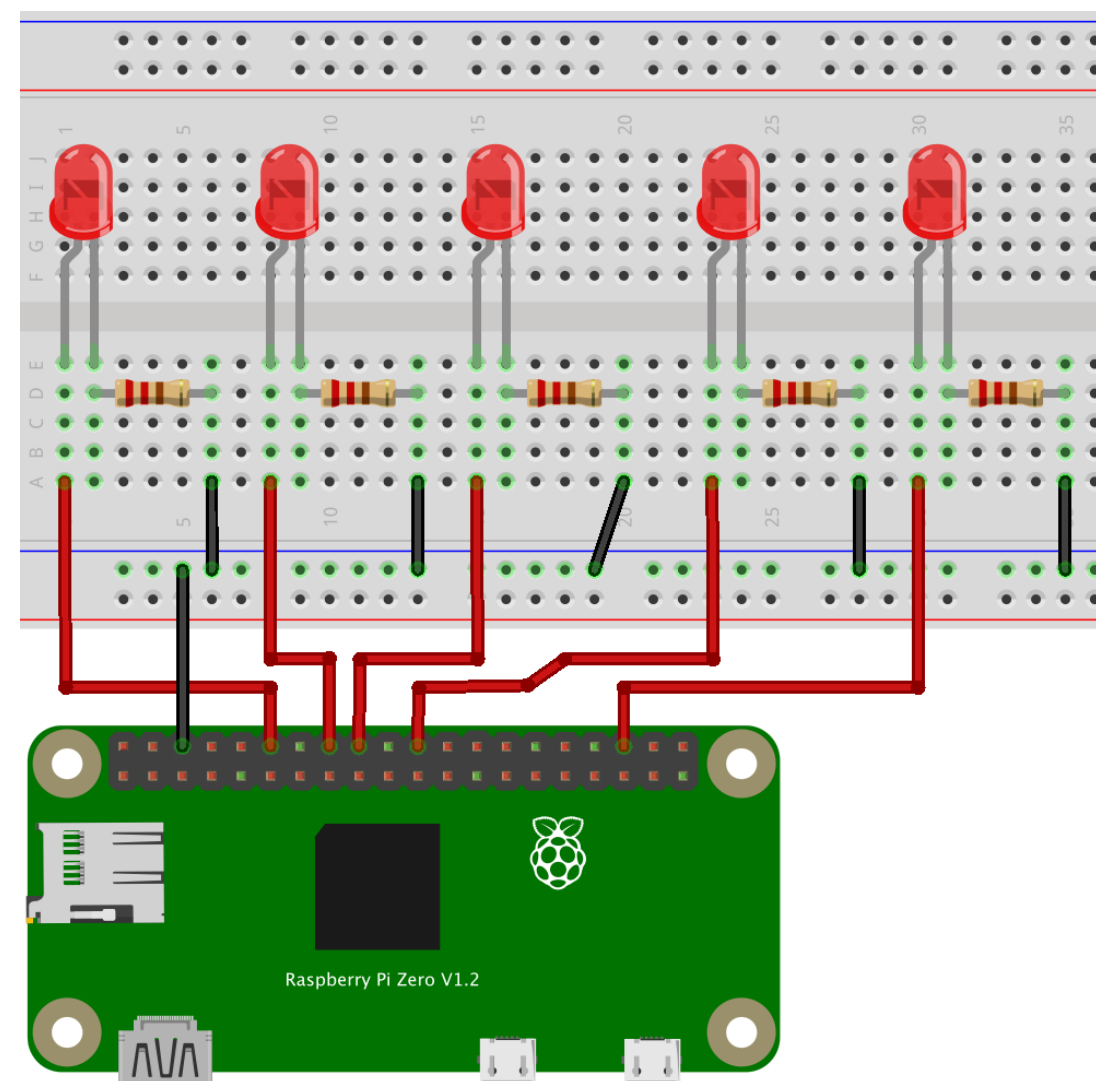
#11 Output from LED_PIN, here, pi.HIGH means 3.3V (=default Digital output [V])

Question 1: How to code the script to blink 5 LEDs?



```
blink_led.py
1 import wiringpi as pi
2 import time
3
4
5
6
7
8
9
10
11
12
13
14
15
```

Question 2: How to Change Brightness of LEDs?

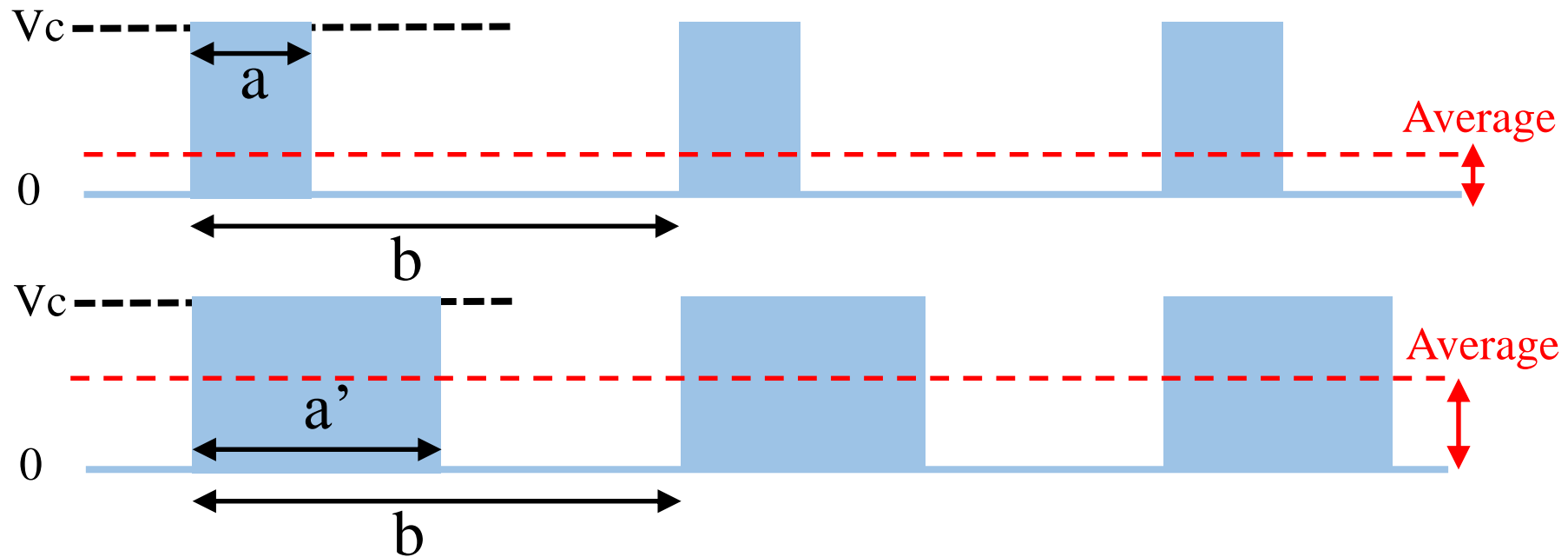


```
blink_led.py
1 import wiringpi as pi
2 import time
3
4
5
6
7
8
9
10
11
12
13
14
15
```

?

Pulse Width Modulation (PWM)

Output modulation like “analog” signal by using only digital output signal!



PWM signals: change in average value **by only modulating pulse width (a or a')** with the fixed pulse period (b)

Python script for “software” PWM in Raspi

```
pwm_led.py x servo2.py x
1 import wiringpi as pi
2 import time
3
4 LED_PIN = 23
5
6 pi.wiringPiSetupGpio()
7 pi.pinMode( LED_PIN, pi.OUTPUT )
8
9 pi.softPwmCreate( LED_PIN, 0, 100 )
10
11 while True:
12     pi.softPwmWrite( LED_PIN, 0 )
13     print("Duty Factor:0%")
14     time.sleep(0.5)
15
16     pi.softPwmWrite( LED_PIN, 50 )
17     print("Duty Factor:50%")
18     time.sleep(0.5)
19
20     pi.softPwmWrite( LED_PIN, 100 )
21     print("Duty Factor:100%")
22     time.sleep(0.5)
23
24     pi.softPwmWrite( LED_PIN, 50 )
25     print("Duty Factor:50%")
26     time.sleep(0.5)
27
```

#1: import wiringpi as pi (for GPIO)

#2: import time module

#4: Setting the led pin

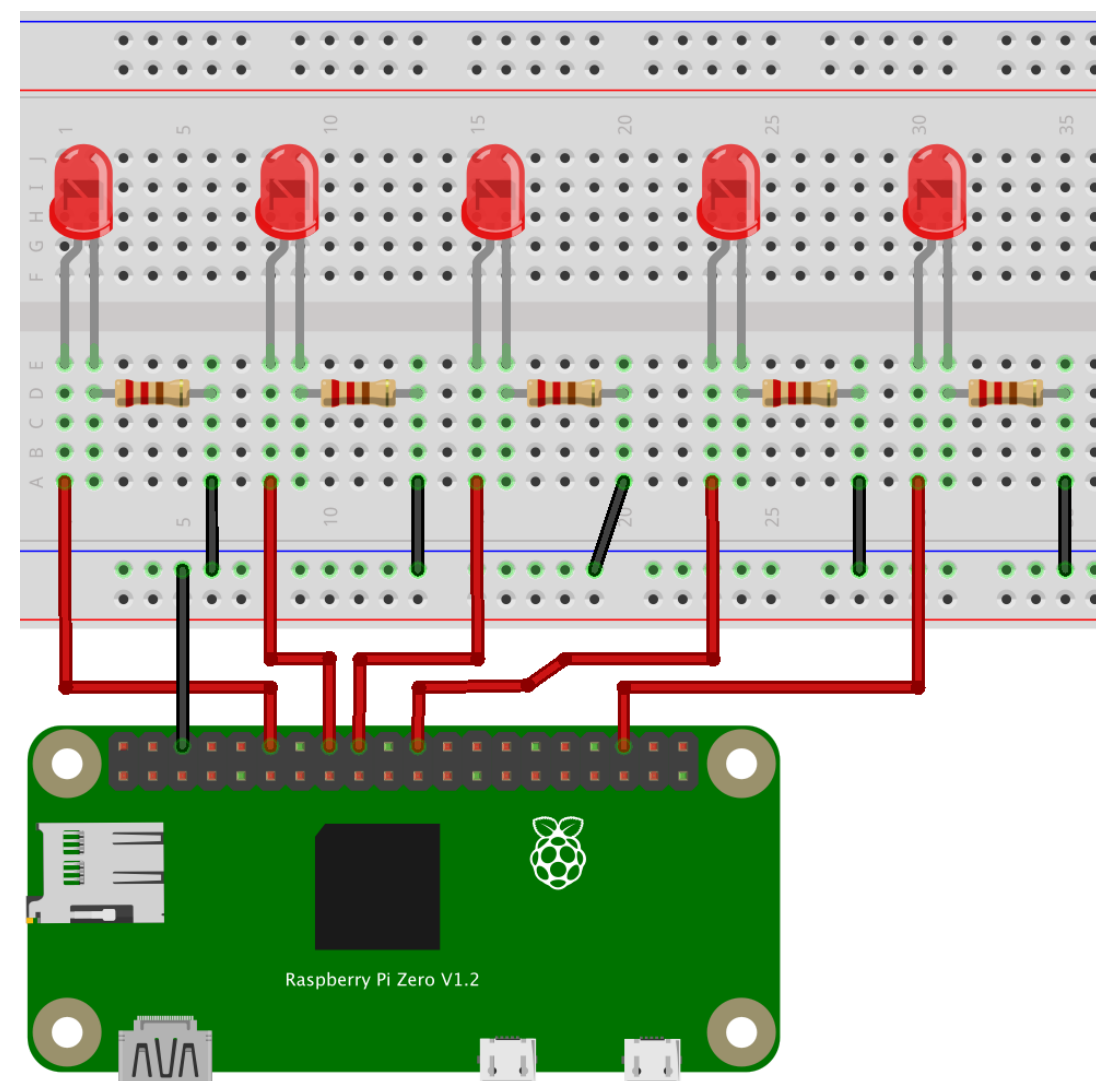
#6&7: Setting the GPIO mode
as “output”=pi.OUTPUT

#9 Setting PWM range (0 to 100) of LED_PIN

#12 Setting PWM output (0) of LED_PIN

Please check how the brightness are changed according
to the parameter (duty factor)

Question 3: Change the Brightness of 5 LEDs, respectively



```
blink_led.py
1 import wiringpi as pi
2 import time
3
4
5
6
7
8
9
10
11
12
13
14
15
```

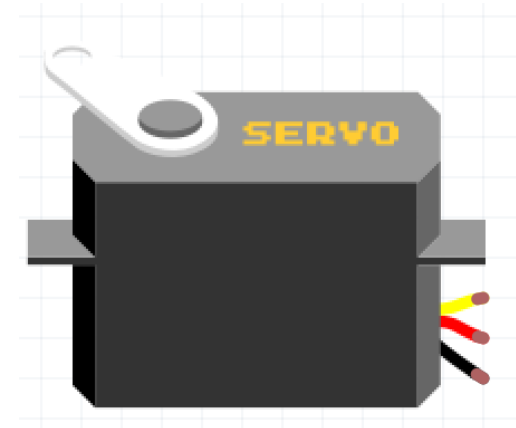
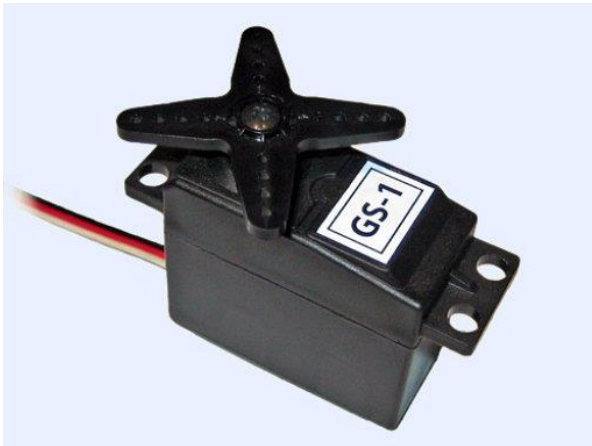
What is a *Motor*?

- ✓ Driving **force (torque)** source (Actuator) for Robots
- ✓ AC(Alternate current) motors, DC(Direct current) motors, Stepping motor, **Servo motors**, etc.

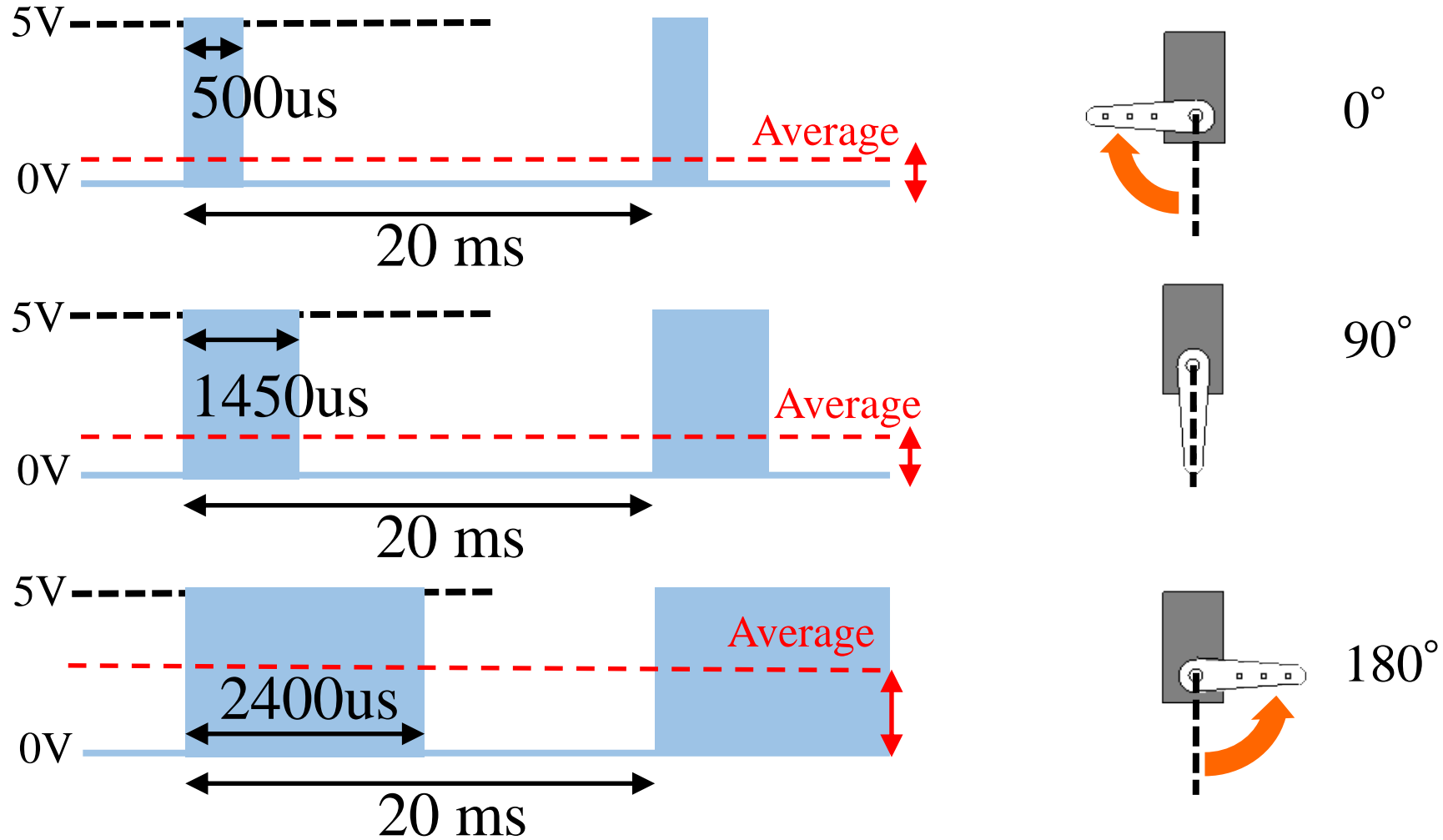


What is a *Servo Motor*?

- ✓ For **precise** control of angular or linear **position, velocity, and acceleration**
- ✓ A servomotor is **a closed-loop servomechanism** that uses position feedback to control its motion and final position.
- ✓ Servo motor includes **an encoder** (detect position, velocity, etc.), **a control circuit** for precise control (e.g. PD control), **a DC motor**, and some gears.



Position (Angle) Control via PWM , e.g., SG90



Note: Parameters for PD control on servo motors **depends on the individual motors**, thus please check the **datasheet** of the corresponding motors.

“Software” PWM and “Hardware” PWM

To control a servo motor, **signal wave form should be precise**, because unstable signal make motor behavior unstable.

“Software” PWM (we use it to control LED brightness) is not sufficient to control a servo motor. Control of a servo motor requires precise PWM wave from (the required resolution is about 0.1ms).

In Wiringpi, we can use **“hardware” PWM** that generated by CPU (SoC) clock timer on **GPIO18 (12pin) & GPIO13 (33pin)**.

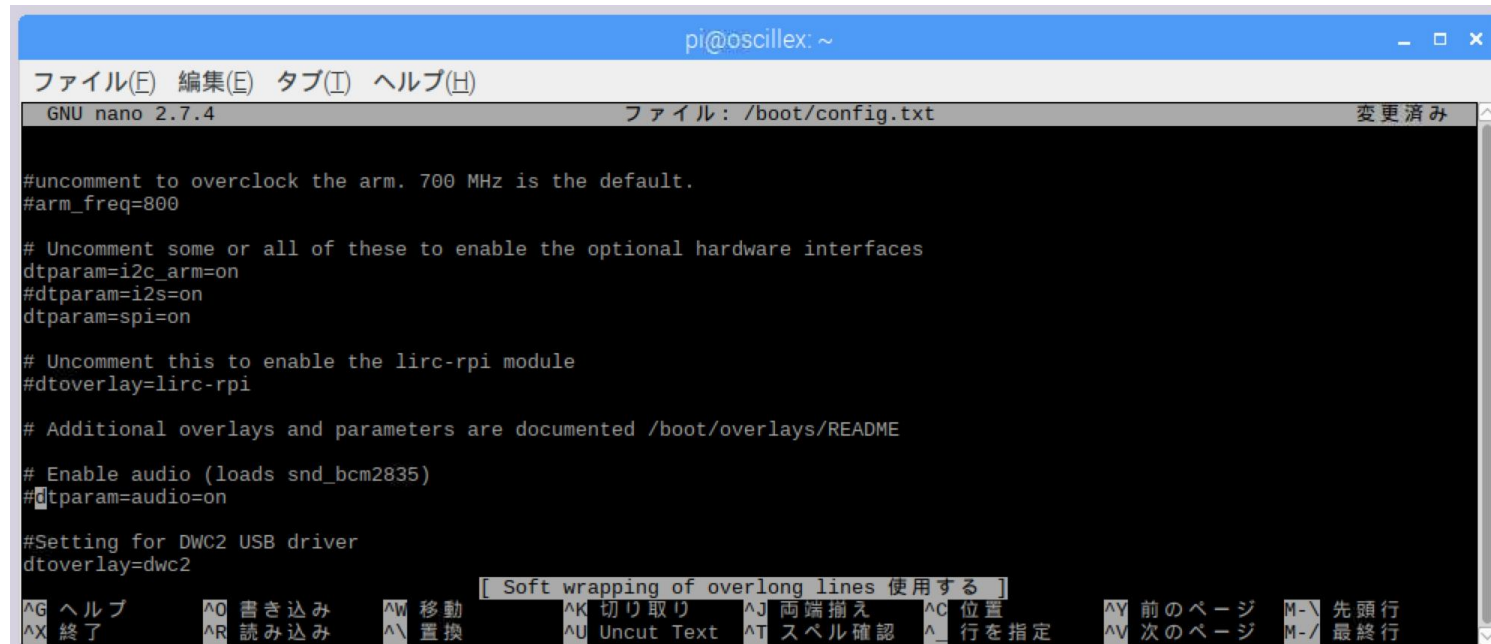
Deactivate Audio Output

Audio output of Raspberry Pi would effect the (hardware) PWM signals, because audio output also use hardware PWM to generate sounds.

`$ sudo nano /boot/config.txt`

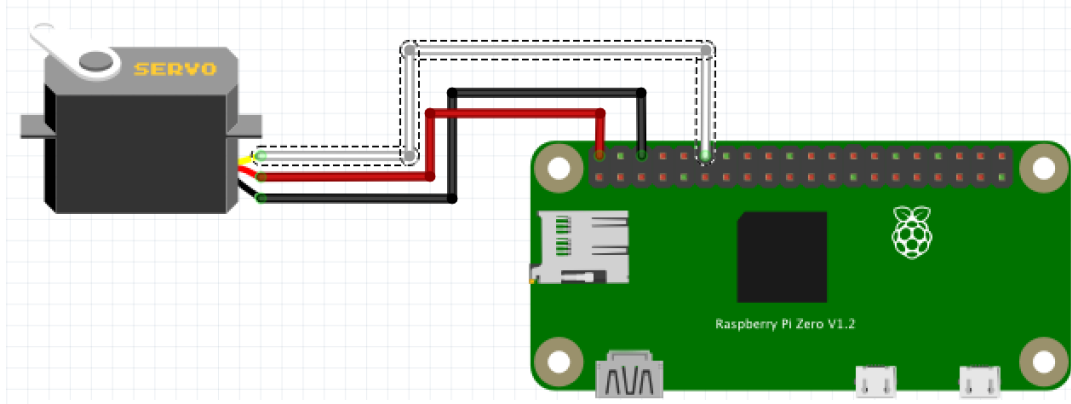
comment out the following sentence:

`#dtparam=audio=on`



```
pi@oscillex: ~  
ファイル(E) 編集(E) タブ(I) ヘルプ(H)  
GNU nano 2.7.4 ファイル: /boot/config.txt 変更済み  
  
#uncomment to overclock the arm. 700 MHz is the default.  
#arm_freq=800  
  
# Uncomment some or all of these to enable the optional hardware interfaces  
dtparam=i2c_arm=on  
#dtparam=i2s=on  
dtparam=spi=on  
  
# Uncomment this to enable the lirc-rpi module  
#dtoverlay=lirc-rpi  
  
# Additional overlays and parameters are documented /boot/overlays/README  
  
# Enable audio (loads snd_bcm2835)  
#dtparam=audio=on  
  
#Setting for DWC2 USB driver  
dtoverlay=dwc2  
  
[ Soft wrapping of overlong lines 使用する ]  
^G ヘルプ ^O 書き込み ^W 移動 ^K 切り取り ^J 両端揃え ^C 位置  
^X 終了 ^R 読み込み ^\ 置換 ^U Uncut Text ^T スペル確認 ^_ 行を指定 ^V 前のページ M- 先頭行  
^V 次のページ M- 最終行
```

Control of a Servo Motor with “Hardware” PWM



\$ sudo python servo2.py

For “hardware” PWM, you should use **sudo** to access CPU (SoC)

```
pwm_led.py x servo2.py x
1 import wiringpi as pi
2 import time
3
4 servo_pin = 18 # GPIO pin for servo motor
5
6 CYCLE = 20 # pwm period of motots [20 ms]
7 MIN_PULSE = 0.5 # minimun pulse width for pwm [ms]
8 MAX_PULSE = 2.4 # maximun pulse width for pwm [ms]
9 MIN_DEG = 0 # minimun angle [deg]
10 MAX_DEG = 180 # maximun angle [deg]
11 RANGE = 2000 # resolution for PWM signal
12
13 clock = int( 19.2 / float(RANGE) * CYCLE * 1000 ) # calculate pwm clock
14 min_val = RANGE * MIN_PULSE / CYCLE # minimum pulse width
15 max_val = RANGE * MAX_PULSE / CYCLE # maximun pulse width
16
17 pi.wiringPiSetupGpio() # setting GPIO
18 pi.pinMode( servo_pin, pi.GPIO.PWM_OUTPUT ) # output as "hardware" PWM
19 pi.pwmSetMode( pi.GPIO.PWM_MODE_MS ) #
20 pi.pwmSetRange( RANGE ) # set PWM Range (Resolution)
21 pi.pwmSetClock( clock ) # set PWM Clock
22
23 # definition of function Angle to Pulse width for PWM
24 def Angle_to_PWM(angle):
25     return int( ((max_val-min_val)/MAX_DEG) * angle + min_val )
26
27
28 while True:
29     pi.pwmWrite( servo_pin, Angle_to_PWM(90) )
30     print("middle 90 deg")
31     time.sleep(1)
32
33     pi.pwmWrite( servo_pin, Angle_to_PWM(0) )
34     print("left 0 deg")
35     time.sleep(1)
36
37     pi.pwmWrite( servo_pin, Angle_to_PWM(90) )
38     print("middle 90 deg")
39     time.sleep(1)
40
41     pi.pwmWrite( servo_pin, Angle_to_PWM(180) )
42     print("right 180 deg")
43     time.sleep(1)
44
```

$$\text{PWM period (20 [ms])} = \frac{1}{19.2 * 10^6 [\text{Hz}] / \text{clock}} * \text{range}$$

$19.2 * 10^6 [\text{Hz}]$: base clock frequency for PWM



$(19.2 * 10^6) / \text{clock} [\text{Hz}]$: PWM counter frequency

range : PWM resolution

#18: Setting for **Hardware PWM**

#19: Set **Mark Space** mode
(Default=Balanced mode)

$$\text{clock} = 20 * 10^{-3} * 19.2 * 10^6 / \text{range}$$

**Thanks for
Listening!!!**

**Looking forward to
working with you all**

^^

