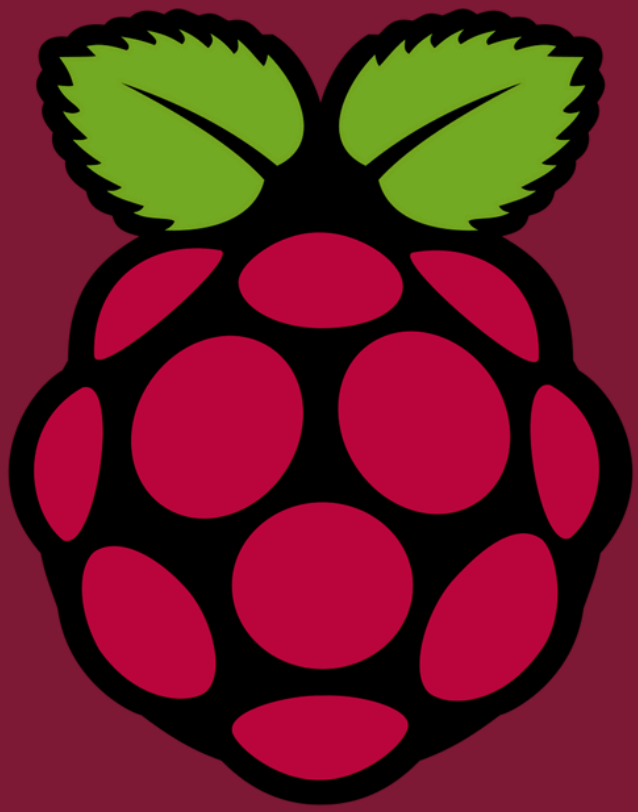
Raspberry Pi Cheat Sheets

Hardware & Software



CPUcademy

Configuration

1. Download the Raspberry Pi **Imager**.
2. Choose **OS** (Legacy **Bullseye**), a Raspberry Pi device that you have, and a storage device (an SD card).
3. Click next and change the settings: **enable SSH**, set username and password, configure wireless LAN, and set locale settings. Wait till the installation on the SD card is finished.
4. Put the **SD card** into the Raspberry Pi and power it on.
5. Find the Raspberry Pi's IP address with **Angry IP Scanner**. Click on the **"IP"** button and try all ranges until you find one that contains your Raspberry Pi (it will be called "raspberrypi.local"). A blue dot means that something is under that address. To search for all blue dots, sort by ping.
6. Type these commands into the CMD:

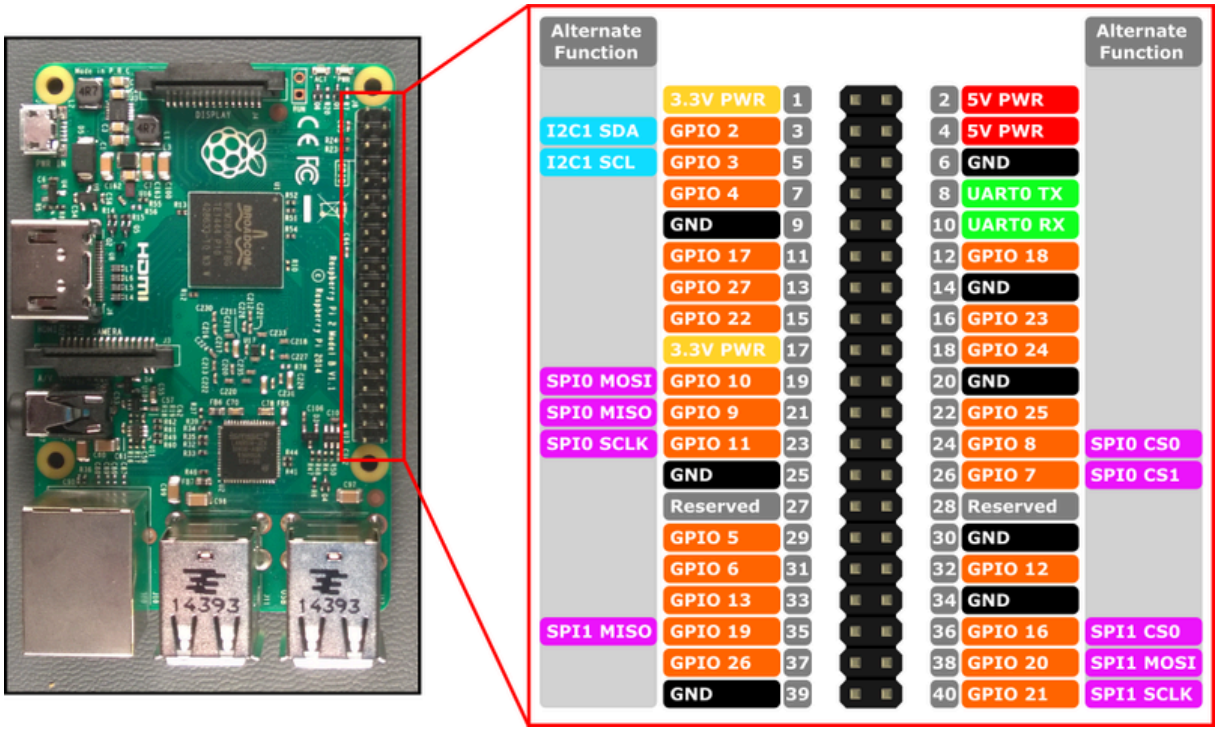
a) ***ssh*** (if it doesn't work - download PuTTy)
b) ***ssh [username]@[ip_of_the_pi]*** (confirm by yes)
c) input the password (it won't be shown)
Now we are in the Raspberry Pi (you can exit by typing ***exit***).
d) ***sudo raspi-config*** (opening configuration on the Raspberry Pi with admin privileges)
Go to "*Interface Options*" and enable VNC (you can move with arrows and the enter key). Click *Finish*.
e) ***sudo reboot***
Log into the menu again (***ssh [username]@[ip_of_the_pi]***, input password, **sudo raspi-config**). Go to "*Systems Options / Auto Login"* and choose "*B4 Autologin Desktop"*. Click *Finish*. Then, log into the menu again. Go to "*Display Options"* and select "*D5 VNC Resolution"* (choose the resolution of your monitor). Again, click *Finish*.

Download RealVNC Viewer for Windows (not for Raspberry Pi because the client is Windows) and sign up. *"Click File / New Connection"*. Input the IP of the Raspberry Pi and enter a name for it. Click on the new icon on the dashboard and log into the Raspberry Pi.
You can't just unplug the Raspberry Pi. You have to **shut it down from the menu**, wait half a minute, and then unplug.
If you want to change the WIFI connected to your Raspberry Pi:
1. Disconnect the SD card and plug it into your computer.
2. Add this (*https://drive.google.com/file/d/1tIP-5_3xNDr08i00fAUSG90vlt-9p0Dp/view?usp=sharing*) file onto it (change the name of the network and its password inside this file).
3. Power it on, check the IP again, and change it in the VNC Raspberry Pi Properties.

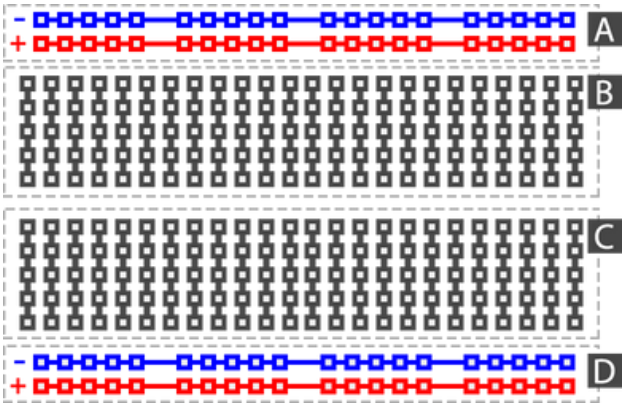
Basics

To code with Python on the Raspberry Pi, we will use the pre-installed Thonny editor. If you want to make a change in the circuit, always power off the board first. When powered on, avoid touching the circuit. Double-check everything before plugging the power cable.

On the right, you can see the schematic of how the GPIOs (General Purpose Input-Output) are numbered and destined on the Raspberry Pi. We connect our sensors to them.

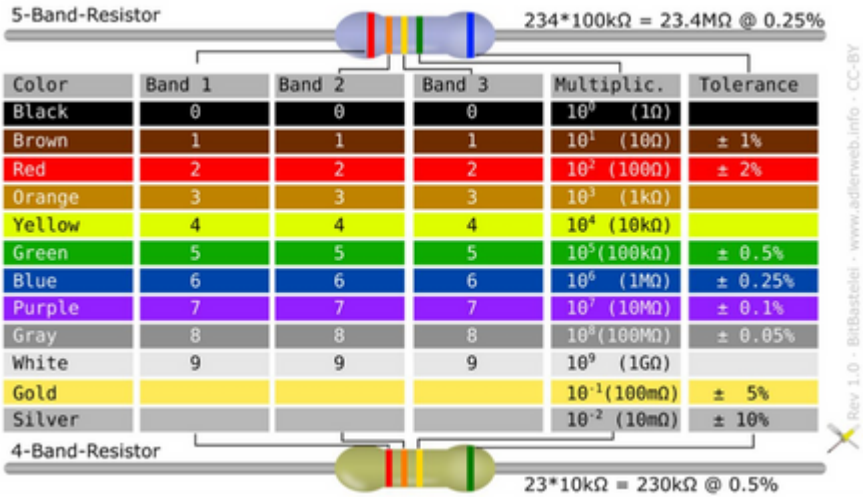


To connect those sensors, we also need a **breadboard** (on the right). It has two special lines on both sides and normal ones in the middle. The plus lines are power lines, and the minus ones are called ground (**GND**). Every five holes in a row are electrically connected below. When we build circuits, we can use male-to-male, female-to-female, and male-to-female cables to connect devices with the Raspberry Pi on the breadboard. We can use one male-to-female cable to connect the power line to the **3.3V** PWR GPIO on the Raspberry Pi. Then, every sensor connected to this line will have the power. We can use the power line from the other side of the board when we want to use a sensor that requires **5V** of power (e.g., a proximity sensor). The Raspberry Pi has two GPIOs with 3.3V of power and two with 5V. Every sensor needs to be somehow connected to the ground (GND). We can connect the GND GPIO to the minus line and then connect all the devices to this line (just like with the power line).



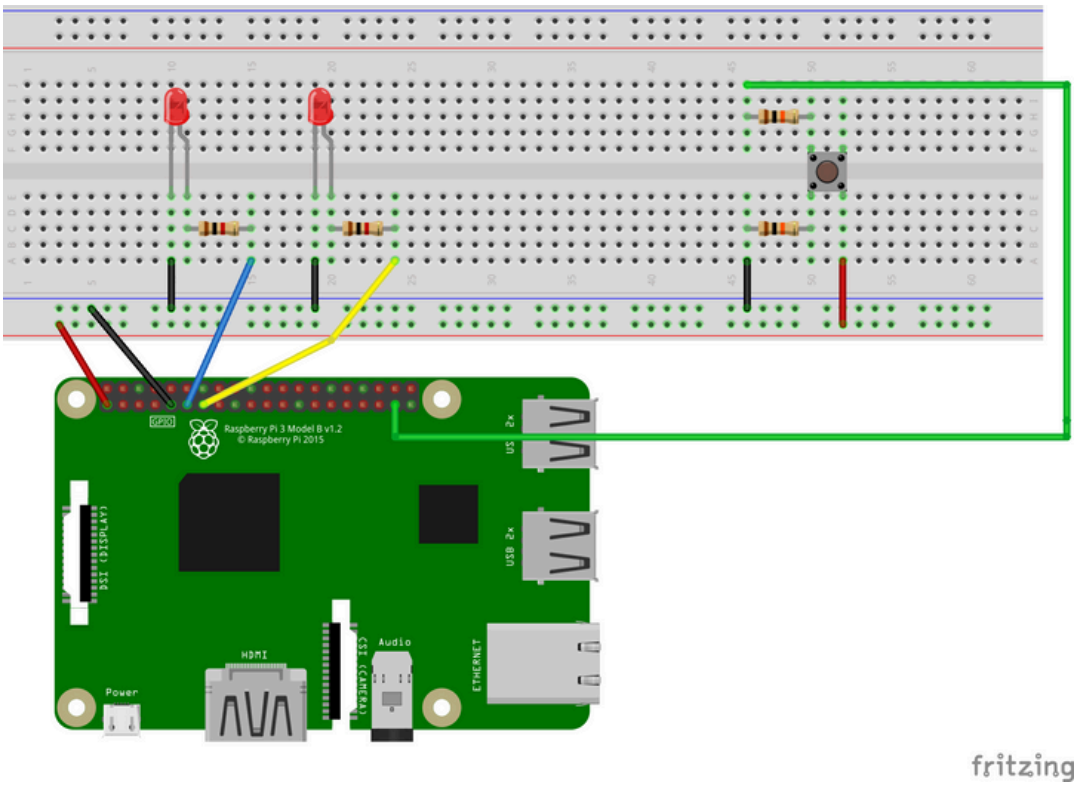
LEDs, to function, need to have their shorter leg plugged into the ground line (or connected to it by a wire) and their longer leg connected to the pin through which we can control it. We cannot connect it to the pin directly, but we have to do it through a **1kΩ** resistor (as shown in the diagram below). The colors of the cables don't matter, but it's a good habit to take red wires for the power line and black for the ground. It doesn't matter to which pin we connect the **LEDs**, as long as it's not a special pin (if it's marked with orange color on the schematic and doesn't have anything more at the side, it is suitable for this purpose).

To create circuits, you will need resistors with different resistance levels. If you don't have them labeled in your kit, use this cheat sheet to distinguish them by colors:



Color	Band 1	Band 2	Band 3	Multiplic.	Tolerance
Black	0	0	0	10^0 (1Ω)	
Brown	1	1	1	10^1 (10Ω)	± 1%
Red	2	2	2	10^2 (100Ω)	± 2%
Orange	3	3	3	10^3 (1kΩ)	
Yellow	4	4	4	10^4 (10kΩ)	
Green	5	5	5	10^5(100kΩ)	± 0.5%
Blue	6	6	6	10^6 (1MΩ)	± 0.25%
Purple	7	7	7	10^7 (10MΩ)	± 0.1%
Gray	8	8	8	10^8(100MΩ)	± 0.05%
White	9	9	9	10^9 (1GΩ)	
Gold				10^{-1}(100mΩ)	± 5%
Silver				10^{-2} (10mΩ)	± 10%

A resistor is the same on both legs, so we can put it however we want. A **push button** has four legs, and we have to place it so that they aren't connected on the breadboard. This means we can place them only in the middle (every leg will have its separate line of five holes). The legs don't have any special order and are all the same. We connect the first leg directly to the power line, the second indirectly to the ground (through a **10kΩ** resistor), and the third through the same type of resistor to a chosen pin (the same rule as with the LEDs). The fourth leg **stays free**. Everything is shown in the diagram below. Remember that if you don't have, e.g., a 1kΩ resistor, you can use a 1.2kΩ resistor, and it won't make any difference.



The Terminal

ls - listing all contents of a directory.
ls -a - showing also the hidden files.
ls -l - showing everything as a list.
ls -la - combination of ***ls -a*** and ***ls -l***.
pwd - printing the directory in which you are now.
cd <dir_name> - navigating to the directory (***cd /*** will show all the options).
clear - clearing the terminal.

nano <file_name> - opening the file in nano editor (also creating the file if it doesn't exist).
touch <file_name> - creating a new file.
mkdir <dir_name> - creating a new directory.
mv <file_path> <new_file_path> - moving a file and / or renaming it.
cp <file_path> <new_file_path> - copying a file to a new file.
rm <file_name> - removing a file.
rm -rf <dir_name> - removing a directory and everything inside, recursively.
cat <file_name> - printing the content of a file.

sudo apt update - updating sources to the latest.
sudo apt install <package_name> - installing a package.
sudo apt remove <package_name> - removing a package.
sudo apt upgrade - upgrading already installed packages to the latest version.

pip3 install <module_name> - installing a Python module.
pip3 uninstall <module_name> - uninstalling a Python module.
pip3 list - printing all installed Python modules.
pip3 list | grep GPIO - searching (like ***CTRL+F***) for the "GPIO" phrase in the list of Python modules.
python3 - opening the Python Shell (use ***exit()*** to exit it).

sudo shutdown now - shutting down the Raspberry Pi.
sudo reboot - rebooting the Raspberry Pi.
hostname -I - printing the IP address of the Raspberry Pi.
df -h - printing currently used and available space on the SD card.
python3 <file_name.py> - executing a Python3 script in the terminal.
man <command_name> - printing a manual for a given command.
which python3 - showing a complete path to a given command (***python3***).

sudo means using a command as an admin. To open an installed program, we just type its name. To copy and paste, we use ***CTRL+SHIFT+C*** and ***CTRL+SHIFT+V***. ***CTRL+X*** exits a Nano file, and ***CTRL+S*** saves it. You can use TAB for auto-completion in the Terminal. To go back to previously executed commands, use the up arrow key on your keyboard. You do not need to add a ***.txt*** extension to create a text file, you can leave it blank (Raspberry Pi OS is a Unix operating system). Press ***CTRL+C*** to stop a running command. If you want to create a Python file using the ***touch*** command, write its extension at the end. Files and folders with a dot at the beginning are hidden.

Raspberry Pi Camera

sudo raspi-config / "*Interface Options*" / "*Legacy Camera*" - enabling the camera in the Raspberry Pi Terminal
Type these commands into the CMD:
- ***ssh [username]@[ip_of_the_pi]*** + enter your password
- ***sudo nano /boot/config.txt*** - uncomment in this file by deleting the # sign:
a) *#hdmi_force_hotplug=1*
b) *#hdmi_group=1*
c) *#hdmi_mode=1* (change it to *hdmi_mode=16*)
Comment this line (by adding the # sign): *dtoverlay=vc4-fkms-v3d*
Save and exit the file.
- ***sudo reboot***

When you are back in the VNC, type ***raspistill*** into the Terminal. If there is no error, it means that you have successfully installed the camera module. It should show you the "help" list of this command.

Using the Camera from the Raspberry Pi Terminal

- ***mkdir camera*** (creating a folder)

Images

raspistill -o camera/img.jpg (o - output) - taking a picture

Raspistill arguments (we add them after the path):

rot 180 - rotating the image by 180 degrees (you may have the camera upside down).

-w 1280 -h 720 - setting a lower quality to save memory.

Videos

raspivid -o camera/video.h264 -rot 180 -w 720 -h 480 (***.h264*** is a video format)

Raspivid, by default, will make a video for 5 seconds. To change that time, we use this argument: ***-t 7000*** (the time is in milliseconds).

Python programs

Powering on a LED

```
import RPi.GPIO as gpio
import time

led_pin = 17 # setting the GPIO number we used on the board

gpio.setmode(gpio.BCM) # setting the right numeration of the GPIO's

gpio.setup(led_pin, gpio.OUT) # gpio.OUT / gpio.IN (setting a certain GPIO to output / input)

for x in range(10):
    gpio.output(led_pin, gpio.HIGH) # HIGH means powering on the LED
    time.sleep(1)
    gpio.output(led_pin, gpio.LOW) # LOW means powering off the LED
    time.sleep(1)

gpio.cleanup() # going to the default settings of the GPIO's to prevent damages
```

Powering on a LED when a button is pressed

```
import RPi.GPIO as gpio
import time

led_pin = 17
button_pin = 26
gpio.setmode(gpio.BCM)
gpio.setup(led_pin, gpio.OUT)
gpio.setup(button_pin, gpio.IN)

while True:
    print(gpio.input(button_pin))
    if gpio.input(button_pin):
        gpio.output(led_pin, gpio.HIGH)
    else:
        gpio.output(led_pin, gpio.LOW)
    time.sleep(0.01) # to reduce the CPU's usage

gpio.cleanup()
```

Powering on one LED after another when a button is pressed (and turning off the previous one) - three LEDs

```
import RPi.GPIO as gpio
import time

gpio.setmode(gpio.BCM)

pins = [17, 27, 22]
for x in pins:
    gpio.setup(x, gpio.OUT)
    gpio.output(x, gpio.LOW)

button_pin = 26
gpio.setup(button_pin, gpio.IN)

previous_state_button = gpio.input(button_pin)
led_index = 0

while True:
    time.sleep(0.1)
    button_state = gpio.input(button_pin)
    if button_state ≠ previous_state_button:
        previous_state_button = button_state
        if button_state:
            for x in pins:
                if pins.index(x) == led_index:
                    gpio.output(x, gpio.HIGH)
                else:
                    gpio.output(x, gpio.LOW)
            led_index += 1
            if led_index ≥ len(pins): led_index = 0

gpio.cleanup()
```

Using the camera

```
from picamera import PiCamera
import time

camera = PiCamera()
camera.resolution = (1280, 720)
camera.rotation = 180
time.sleep(2) # the camera has to have the time to calibrate itself

file_name = "/home/pi/camera/image.jpg" # an absolute path
camera.capture(file_name)
print("The image is done.")

file_name_2 = "/home/pi/camera/vid.h264"
camera.start_recording(file_name_2)
camera.wait_recording(5) # the argument is a number of seconds
camera.stop_recording()
print("The video is done.")
```

Sending emails

```
import yagmail

with open("/home/pi/.local/share/.email_password", "r") as file:
    password = file.read()

yag = yagmail.SMTP("cpucademy.sender@gmail.com", password)

yag.send(to="youremail@gmail.com",
         subject="Raspberyy Pi Email",
         contents="Hello World!",
         attachments="/home/pi/file.txt")

print("Email sent")
```

To use this, create an email address, enable 2-step verification, and search for app passwords. Generate one for the Raspberry Pi and copy it. Create a hidden file with the name and the path like on the screenshot, and paste the password inside it. Now, everything will work.

Creating a local server

```
from flask import Flask
import RPi.GPIO as gpio

button_pin = 26
gpio.setmode(gpio.BCM)
gpio.setup(button_pin, gpio.IN)

# 0.0.0.0:5000 - 0.0.0.0 is an IP address and 5000 is a port
# opening our server in a browser of a device that is connected to the same network:
# [pi_IP_address]:[the port number]

app = Flask(__name__) # initializing

@app.route("/") # creating a route to the homepage
def index():
    return "Hello world!"

@app.route("/push-button") # creating a new route (a new subpage)
def push_button(): # the name of the function doesn't have to be the same
    if gpio.input(button_pin) == gpio.HIGH:
        return "Button is pressed."
    return "Button is not pressed."

app.run(host="0.0.0.0", port=8500) # running the application
# 0.0.0.0 causes that every computer in the network will be able to connect to this server
# "port" will change the port number

gpio.cleanup()
```

Turning the LEDs on and off based on the subpage that we have opened

```
from flask import Flask
import RPi.GPIO as gpio

button_pin = 26
led_pins = [17, 22, 27]
gpio.setmode(gpio.BCM)
for x in led_pins:
    gpio.setup(x, gpio.OUT)
for x in led_pins:
    gpio.output(x, gpio.LOW)

app = Flask(__name__)

@app.route("/")
def index():
    return "LEDs"

@app.route("/led/<int:led_pin>/state/<int:led_state>")
def trigger_led(led_pin, led_state):
    if not led_pin in led_pins:
        return "Wrong GPIO number"
    if led_state == 1:
        gpio.output(led_pin, gpio.HIGH)
    elif led_state == 0:
        gpio.output(led_pin, gpio.LOW)
    else:
        return "State must be 0 or 1"
    return str(led_pin) + " LED is " + str(led_state)

app.run(host="0.0.0.0")
gpio.cleanup()
```

We should always place the code in the ***try*** block and the ***gpio.cleanup()*** in the ***main*** and the ***except*** block. This way, we will prevent not executing this command through killing the program.

```
import time
print("Begin")
try:
    while True:
        print("x")
        time.sleep(0.5)
except KeyboardInterrupt:
    print("Keyboard Interrupt")
print("End")
```

Projects

If we have a program that we would like to run automatically after turning on the Raspberry Pi, we have to follow these steps:

1. Type this command into the Terminal: ***sudo crontab -e***.
2. At the bottom of the file that has opened, write: ***@reboot sudo python3 /home/pi/Desktop/test.py***.
3. If later on you want to stop the program, commend this line. Remember that if you run this program in the background, and then use it from the code editor, some errors may occur.