

Raspberry Pi

??????Linux??

- [Learning](#)
- [Picroft](#)
- [Raspberry Pi OS](#)
- [PiShrink](#)
- [Xiaomi Mijia Temperature and Humidity Sensor](#)
- [Monitoring](#)
- [Optimization](#)
- [USB to TTL](#)
- [DietPi](#)

Learning

Tutorials

- [Create your own custom Raspberry Pi image](#)
- [PiShrink – Make Raspberry Pi Images Smaller](#)
- [Getting Started with Raspberry Pi | Random Nerd Tutorials](#)
- [RaspberryTips – Raspberry Pi projects and tutorials](#)

Pi Kiosk

- <https://github.com/geerlingguy/pi-kiosk>

Hardware

- [CrowPi](#) - Raspberry Pi Laptop
 - [Video] [This New Raspberry Pi Laptop Is Actually Pretty Good, CrowPi L Hands On](#)
- Case
 - [???Raspberry Pi4B???????](#)
 - [???4B????????\(?????\)](#)
 - [Is This The Best Raspberry Pi 4 Case? The New Argon One M.2 - Review](#)
- DHT22 Sensor
 - [Monitor temperature and humidity with Grafana and Raspberry Pi](#)
- NVMe SSD
 - [Video] [FINALLY! NVMe SSDs on the Raspberry Pi](#)
- SunFounder
 - [Raspberry Pi Store \(sunfounder.com\)](#)

Picroft

Introduction

Picroft is a ready-made way to run Mycroft on a Raspberry Pi 3, 3B+ or 4 and is provided as a disk image that you can burn to a Micro SD card.

Links

- [Official Home](#)
- Youtube Tutorials
 - <https://www.youtube.com/watch?v=M398I6YIleM>
 - <https://www.youtube.com/watch?v=4luTnRpfgbs>

Raspberry Pi OS

Installation

- [Raspberry Pi Imager](#)

Ubuntu 22.04

```
sudo apt install rpi-imager
```

Create User

NOTE: ?? Raspberry Pi OS ???????? pi????????????????

Headless Setup

SD Card > Boot partition > File: `userconf`

userconf:

```
alang:<encrypted-password>
```

Generate encrypted password

```
echo 'mypassword' | openssl passwd -6 -stdin
```

CLI

```
# Add user
sudo adduser <username>
sudo usermod -a -G adm,dialout,cdrom,sudo,audio,video,plugdev,games,users,input,netdev,gpio,i2c,spi
<username>

# Delete user
```

```
sudo deluser -remove-home <username>
```

Enable SSH

🔊 Default credential is pi / raspberry

Headless setup

SD Card > Boot partition > File: `ssh` (an empty file)

SSH can be enabled by placing a file named `ssh`, without any extension, onto the boot partition of the SD Card.

Desktop

1. Launch *Raspberry Pi Configuration* from the *Preferences* menu
2. Navigate to the *Interfaces* tab
3. Select *Enabled* next to *SSH*
4. Click *OK*

Option #3: Using the `raspi-config`

1. Enter `sudo raspi-config` in a terminal window
2. Select *Interfacing Options*
3. Navigate to and select *SSH*
4. Choose *Yes*
5. Select *Ok*
6. Choose *Finish*

Wireless LAN

Headless setup

SD Card > Boot partition > File: `wpa_supplicant.conf`

wpa_supplicant.conf :

```
country=TW # Your 2-digit country code
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
network={
    ssid="YOUR_NETWORK_NAME"
    psk="YOUR_PASSWORD"
```

```
key_mgmt=WPA-PSK
}
```

CLI

raspi-config CLI

```
# Usage: sudo raspi-config nonint do_wifi_ssid_passphrase <ssid> <passphrase> [hidden] [plain]
sudo raspi-config nonint do_wifi_ssid_passphrase myssid 'mypassphrase' 0 0 # Visible SSID, passphrase quoted
```

nmcli

```
nmcli dev wifi list
sudo nmcli dev wifi connect <example_ssid>
sudo nmcli --ask dev wifi connect <example_ssid> hidden yes
```

Python

Install pip

```
sudo apt install python3-pip
pip --version
sudo pip install --upgrade pip
```

?? pip ???????????????? 3rd-party ????????????????

“ error: externally-managed-environment

× This environment is externally managed

??> To install Python packages system-wide, try apt install python3-xyz, where xyz is the package you are trying to install.

If you wish to install a non-Debian-packaged Python package, create a virtual environment using python3 -m venv path/to/venv. Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make sure you have python3-full installed.

For more information visit <http://rptl.io/venv>

?? Raspberry Pi OS ?????????? 3rd-party ?????????? python ??????????????

?? python ????

```
# [root] user [root] pi
mkdir myproject
cd myproject
python -m venv env
source env/bin/activate
which python
# [root] pip [root]
pip install --upgrade pip
# [root]
pip install paho-mqtt
# [root]
pip list
```

??? python ?????????????????? python????????????????? python venv?

```
#!<path-to-venv>/bin/python
```

PiShrink

Dumping existing image

Install cockpit on Raspberry Pi

```
sudo apt install cockpit
```

Shutdown Raspberry PI, take out the SD card, and connect it to your PC.

The `boot` - and `rootfs` -partitions were automatically mounted at the mentioned mount points. Before you proceed, unmount them with :

```
umount /dev/mmcblk0p1  
umount /dev/mmcblk0p2
```

Now we copy the contents of the SD card to our file system.

```
sudo dd if=/dev/mmcblk0 of=/home/along/worktmp/PiShrink/my_rasp.v2.img bs=32M
```

Download the PiShrink

<https://github.com/Drewsif/PiShrink>

Now invoke the script by typing:

```
sudo ./pishrink.sh my_rasp.v2.img
```

Flash the image to the SD card

- [BalenaEtcher](#)
- [SD Cards and Writing Images](#)

Xiaomi Mijia Temperature and Humidity Sensor

????

- ????????????
- ???LYWSD03MMC

Raspberry Pi OS

??

“ ???????????????????????????

```
bluetoothctl
> scan on
> devices
> scan off
> exit
```

?? Python module

“ NOTE: ?? Pi OS ???? Python venv ???? pip ??????????

```
# [user] user [user] (non-root)
mkdir mitemp
cd mitemp
python -m venv env
source env/bin/activate
which python
# [user] pip [user]
pip install --upgrade pip
# [user]
sudo apt install libglib2.0-dev
```

```
pip install bluepy
pip install lywsd03mmc
pip install paho-mqtt
```

“ ???ERROR: Could not build wheels for bluepy, which is required to install pyproject.toml-based projects

???

```
sudo apt install libglib2.0-dev
```

Telegraf

Option 1: Python script

?????

```
mv /etc/telegraf/telegraf.conf /etc/telegraf/telegraf.conf.orig
telegraf --input-filter exec --output-filter influxdb_v2 config > /etc/telegraf/telegraf.conf
```

telegraf.conf

```
[[outputs.influxdb_v2]]
  urls = ["http://127.0.0.1:8086"]

  ## Token for authentication.
  token = "YOUR-TOKEN"

  ## Organization is the name of the organization you wish to write to.
  organization = "YOUR-ORAG"

  ## Destination bucket to write into.
  bucket = "YOUR-BUCKET"

  ## Timeout for HTTP messages.
  timeout = "5s"

[[inputs.exec]]
  interval = "180s"
  commands = ["sudo -u pi /home/pi/mitemp/env/bin/python /home/pi/mitemp/go-mitemp.py"]
```

```
## Timeout for each command to complete.
```

```
timeout = "30s"
```

```
## Data format to consume.
```

```
## Each data format has its own unique set of configuration options, read
```

```
## more about them here:
```

```
## https://github.com/influxdata/telegraf/blob/master/docs/DATA_FORMATS_INPUT.md
```

```
data_format = "influx"
```

Test the configuration

```
telegraf -config /etc/telegraf/telegraf -test
```

Set the sudo for telegraf

“ NOTE: Pi OS ? telegraf ??? non-root ?????????????????? script????? sudo?

Edit: `/etc/sudoers.d/011_telegraf-nopasswd`

```
telegraf ALL=(ALL) NOPASSWD: /home/pi/mitemp/env/bin/python /home/pi/mitemp/go-mitemp.py
```

Option 2: MQTT Consumer

- Plugin: [telegraf/plugins/inputs/mqtt_consumer/README.md](https://github.com/influxdata/telegraf/blob/master/docs/telegraf/plugins/inputs/mqtt_consumer/README.md) at release-1.29 .
[influxdata/telegraf \(github.com\)](https://github.com/influxdata/telegraf)
- [MQTT Broker ??? Mosquitto ?????????????????? - Office ??](#)

Tutorials

- [pip: LYWSD03MMC](#)
- [?? IoT 30Day? Day 18 ????????](#)
- [Xiaomi Mijia Hygrothermo v2 sensor data on Raspberry Pi](#)
- [Custom firmware for the Xiaomi Thermometer LYWSD03MMC](#)
- [Home Assistant Xiaomi Mijia LYWSD03MMC Temperature and Humidity Sensor Tutorial September 2022](#)

- [Xiaomi Mijia Temperature, and Humidity Dashboard](#)
- [How to Use Bluetooth on Raspberry Pi: GUI & Command Guide – RaspberryTips](#)

Monitoring

Tutorials

Telegraf + InfluxDB + Grafana

- [TheMickeyMike/raspberrypi-temperature-telegraf: Collect RaspberryPi CPU and GPU temperature with telegraf \(github.com\)](#)
- [Raspberry Pi Monitoring | Grafana Labs](#)
- [System monitoring | Nelis Oostens](#)
- [Simple Pi Stats | Grafana Labs](#)
- [Telegraf / Raspberry Metrics InfluxDB 2.0 \(Flux\) | Grafana Labs](#)
- [Telegraf: system PI dashboard | Grafana Labs](#)
- [?? vcgencmd ???? Raspberry Pi ? CPU ?????????????? - G. T. Wang](#)

Telegraf + InfluxDB

- [Monitoring Your Raspberry Pi System using InfluxDB Telegraf | Random Nerd Tutorials](#)

Dashboard ID 17191

- Link: [Telegraf / Raspberry Metrics InfluxDB 2.0 \(Flux\) | Grafana Labs](#)
- Dependencies: InfluxDB v2, Flux Language, Telegraf

telegraf.conf:

```
[[outputs.influxdb_v2]]
  ## The URLs of the InfluxDB cluster nodes.
  ##
  ## Multiple URLs can be specified for a single cluster, only ONE of the
  ## urls will be written to each interval.
  ## ex: urls = ["https://us-west-2-1.aws.cloud2.influxdata.com"]
  urls = ["http://your.influxdb.server:8086"]
  ## Token for authentication.
```

```
token = "THIS-IS-TOKEN"
```

```
## Organization is the name of the organization you wish to write to.
```

```
organization = "YOUR-ORG"
```

```
## Destination bucket to write into.
```

```
bucket = "YOUR-BUCKET"
```

```
[[inputs.net]]
```

```
[[inputs.netstat]]
```

```
[[inputs.file]]
```

```
files = ["/sys/class/thermal/thermal_zone0/temp"]
```

```
name_override = "cpu_temperature"
```

```
data_format = "value"
```

```
data_type = "integer"
```

```
[[inputs.exec]]
```

```
#Für ein 32bit System (welches überwacht werden soll)
```

```
commands = ["/usr/bin/vcgencmd measure_temp"]
```

```
#oder für ein 64bit System (welches überwacht werden soll), dann die Raute(#) entfernen und bei 32bit die  
Raute hinzufügen
```

```
# commands = ["/usr/bin/vcgencmd measure_temp"]
```

```
name_override = "gpu_temperature"
```

```
data_format = "grok"
```

```
grok_patterns = ["%{NUMBER:value:float}"]
```

```
# Read metrics about cpu usage
```

```
[[inputs.cpu]]
```

```
## Whether to report per-cpu stats or not
```

```
percpu = true
```

```
## Whether to report total system cpu stats or not
```

```
totalcpu = true
```

```
## If true, collect raw CPU time metrics
```

```
collect_cpu_time = false
```

```
## If true, compute and report the sum of all non-idle CPU states
```

```
report_active = false
```

```
## If true and the info is available then add core_id and physical_id tags
```

```
core_tags = false
```

```
# Read metrics about disk usage by mount point
```

```
[[inputs.disk]]
```

```
## By default stats will be gathered for all mount points.
```

```
## Set mount_points will restrict the stats to only the specified mount points.
```

```

# mount_points = ["/"]
## Ignore mount points by filesystem type.
ignore_fs = ["tmpfs", "devtmpfs", "devfs", "iso9660", "overlay", "aufs", "squashfs"]
## Ignore mount points by mount options.
## The 'mount' command reports options of all mounts in parathesis.
## Bind mounts can be ignored with the special 'bind' option.
# ignore_mount_opts = []
# Read metrics about disk IO by device
[[inputs.diskio]]
## By default, telegraf will gather stats for all devices including
## disk partitions.
## Setting devices will restrict the stats to the specified devices.
# devices = ["sda", "sdb", "vd*"]
## Uncomment the following line if you need disk serial numbers.
# skip_serial_number = false
#
## On systems which support it, device metadata can be added in the form of
## tags.
## Currently only Linux is supported via udev properties. You can view
## available properties for a device by running:
## 'udevadm info -q property -n /dev/sda'
## Note: Most, but not all, udev properties can be accessed this way. Properties
## that are currently inaccessible include DEVTYPE, DEVNAME, and DEVPATH.
# device_tags = ["ID_FS_TYPE", "ID_FS_USAGE"]
#
## Using the same metadata source as device_tags, you can also customize the
## name of the device via templates.
## The 'name_templates' parameter is a list of templates to try and apply to
## the device. The template may contain variables in the form of '$PROPERTY' or
## '${PROPERTY}'. The first template which does not contain any variables not
## present for the device is used as the device name tag.
## The typical use case is for LVM volumes, to get the VG/LV name instead of
## the near-meaningless DM-0 name.
# name_templates = ["$ID_FS_LABEL", "$DM_VG_NAME/$DM_LV_NAME"]
# Get kernel statistics from /proc/stat
[[inputs.kernel]]
# no configuration
# Read metrics about memory usage
[[inputs.mem]]
# no configuration

```



```
# Get the number of processes and group them by status
```

```
[[inputs.processes]]
```

```
  # no configuration
```

```
# Read metrics about swap memory usage
```

```
[[inputs.swap]]
```

```
# Read metrics about system load & uptime
```

```
[[inputs.system]]
```

Set the sudo

```
sudo usermod -aG video telegraf
```

Optimization

Disable Desktop Booting

`raspi-config` > 1 System Options > S5 Boot / Auto Login

Limit GPU Memory to 16B

`raspi-config` > 4 Performance Options > P2 GPU Memory

Disable HDMI

Edit `/etc/rc.local`

```
/usr/bin/tvservice -o  
exit 0
```

Disable Bluetooth

```
sudo systemctl disable hciuart  
systemctl disable bluetooth.service
```

Edit `/boot/config.txt`

```
dtoverlay=disable-bt
```

Disable Onboard LED

Edit `/boot/config.txt`

```
dtparam=pwr_led_trigger=none  
dtparam=pwr_led_activelow=off  
  
dtparam=act_led_trigger=none  
dtparam=act_led_activelow=off
```


USB to TTL

Tutorials

- [?? PL2303 HXD USB ? TTL ??????????? Raspberry Pi B+ - G. T. Wang \(gtwang.org\)](#)
- [? Raspberry Pi 4B ??? UART ?? - HackMD](#)
- [Using a USB Serial Adapter \(UART\) to Help Debug Your Raspberry Pi](#)

Hardware

- [???/???PL2303HXD USB ? TTL ?????? - ?????? \(piepie.com.tw\)](#)
- [PL2303HX USB ? TTL ?????? – ??????? TaiwanIOT Studio](#)

DietPi

DietPi is a **highly optimised & minimal Debian-based Linux distribution**. DietPi is extremely lightweight at its core, and also extremely easy to install and use.

Setting up a single board computer (SBC) or even a computer, for both regular or server use, takes time and skill. DietPi provides an **easy way to install and run favourite software you choose**.

- <https://dietpi.com/>