

PyCon China 2024

For Good . For fun.
2024/11/23 中国 上海

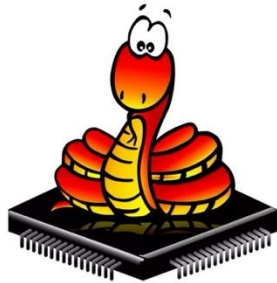
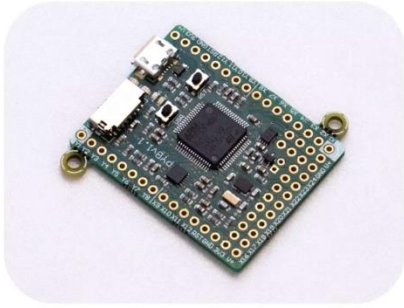


人生苦短， 嵌入式AI用
“Python”

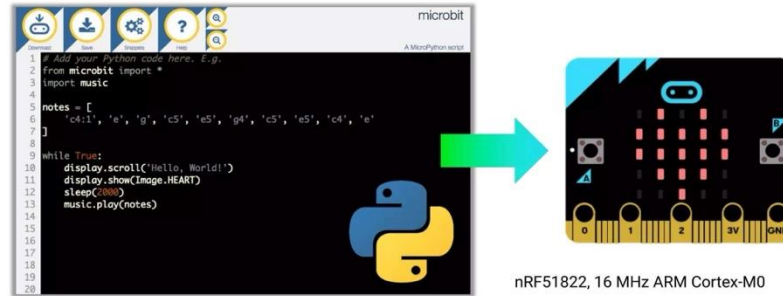
Rb

MicroPython

Kickstarter 2013 : PyBoard + MicroPython



BBC micro:bit : 2016

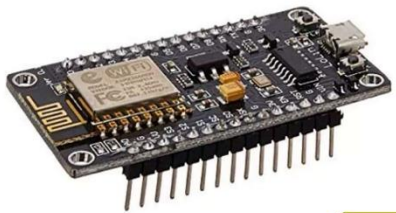


MicroPython on servers and desktop



<https://github.com/micropython/micropython/wiki/Getting-Started>

ESP8266



- Wifi 802.11 b/g/n
- 4MB Flash
- 32 KiB instruction RAM
- 32 KiB instruction cache RAM
- 80 KiB user-data RAM

US\$ 5

ESP32



US\$ 8

- Wifi 802.11 b/g/n
- Bluetooth v4.2 BR/EDR^(*) and BLE^(*)
- Dual Core
- 4MB Flash
- 520 KiB SRAM

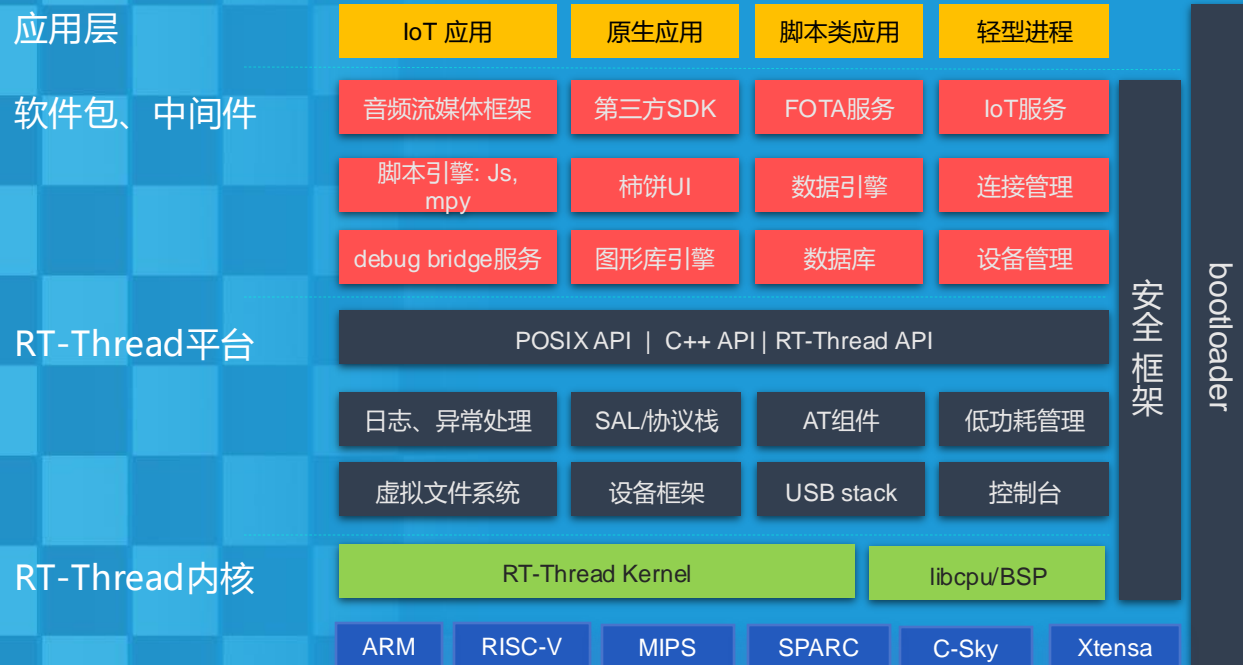
^(*) BR = Basic Rate, EDR = Enhanced Data Rate, BLE = Bluetooth Low Energy

MicroPython on bare metal Raspberry Pi Zero



<https://github.com/kochoch/micropython-raspberrypi>

Introduction to the rtthread rtos



01. 高度可定制化

- 结构化清晰，组件间耦合性低.
- 方便的定制化工具

02. 完整的生态

- 初具完整的生态：芯片，组件/软件包，开发者，网络应用

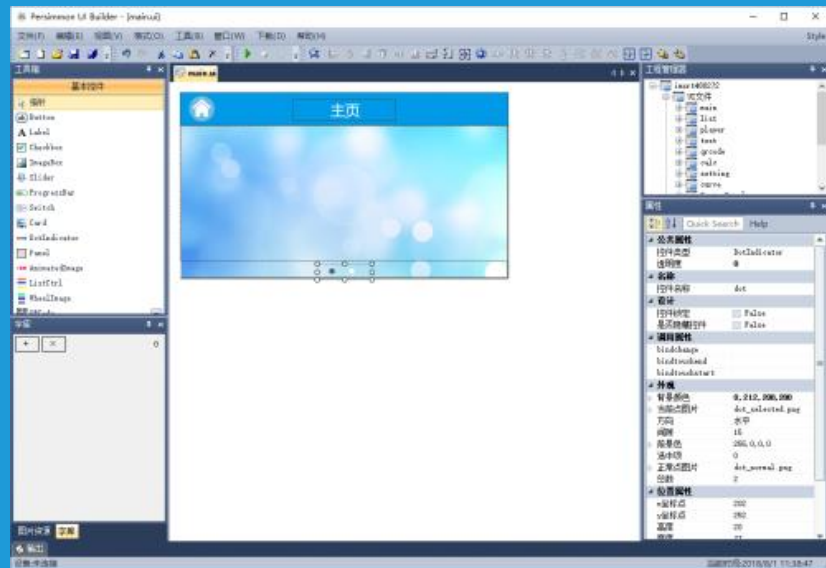
03. 一体化系统

- 应用与内核编译在一起，做为一个整体工程进行维护
- 资源占用精炼，但功能复杂后，维护代价大，容易带来安全性问题

RT-Thread Applets Support

追求极致简单，最大化提升生成效率

- ◆ 突破原有固件弊端：
 - ◆ 维护多套版本
 - ◆ 程序复杂后，难以维护
- ◆ 以JavaScript、MicroPython作为应用开发语言
- ◆ 通过udb打通PC、设备的交互障碍：
 - ◆ udb over USB
 - ◆ udb over 网络
 - ◆ 叠加vscode



通过USB或网络下载



MicroPython for RT-Thread

RT-Thread文档中心

RT-Thread 标准版本

> 简介

> 快速上手

> 内核

> 设备和驱动

> Device IPC

> 组件

√ 软件包

> 物联网

> 工具

> LVGL 用户手册

√ MicroPython 用户手册

介绍

主要特性

MicroPython 的优势

MicroPython 的应用领域

潘多拉 MicroPython 开发板

W601 MicroPython 开发板

麻雀一号音视频开发板

MicroPython IDE

MicroPython 库介绍

MicroPython 固件开发指南

MicroPython C 模块扩展

> PikaScript 用户手册

> RT-Thread 用户手册

修改此文档

PR提交

cmd - menuconfig

MicroPython 入门

本文档将初步介绍 MicroPython

主要特性

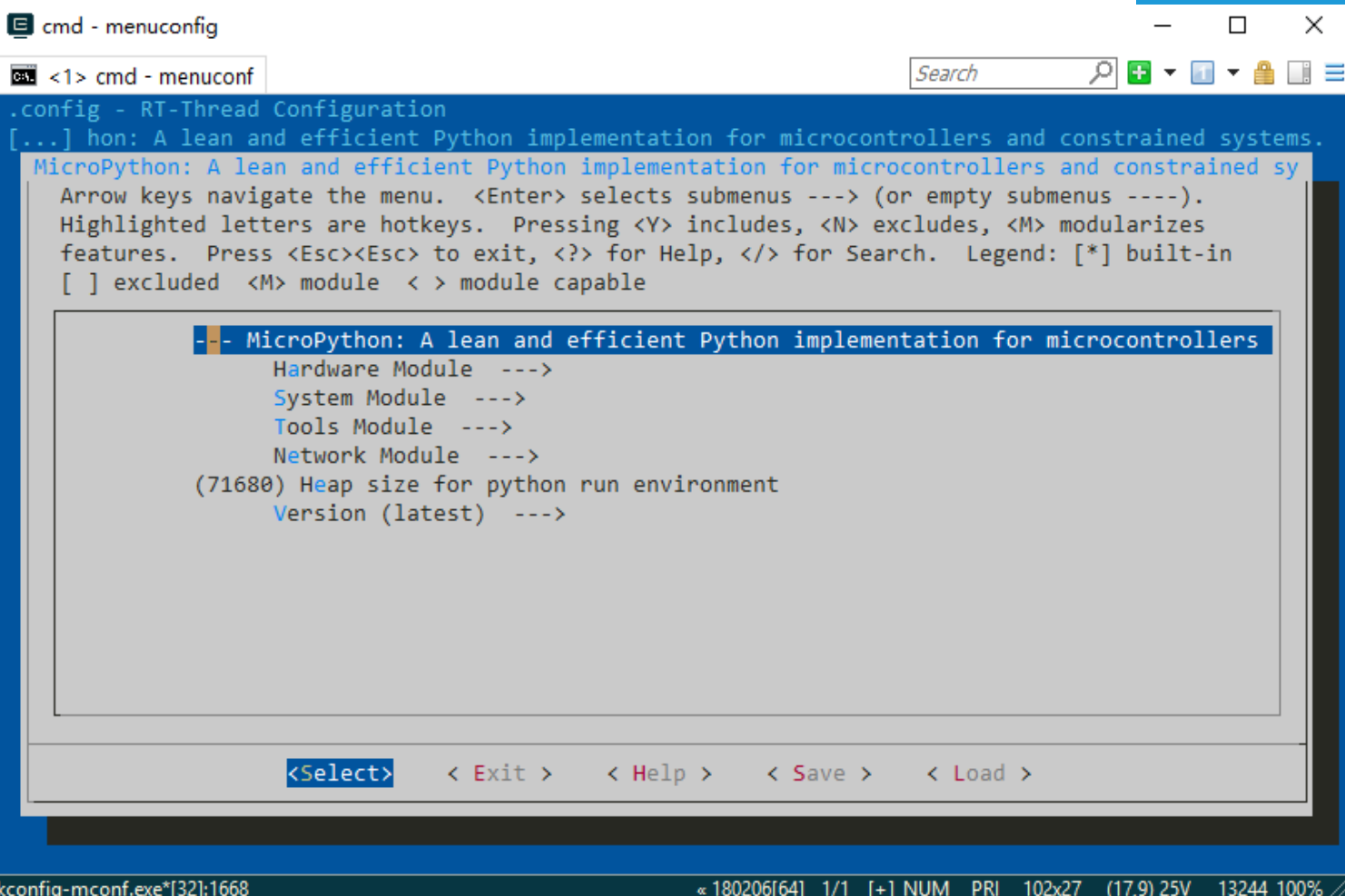
- MicroPython 是 Python 3 编译器
- RT-Thread MicroPython 可以运行在 RT-Thread 内核上
- MicroPython 可以运行在有一块微控制器的开发板上
- MicroPython 富有各种高级特性
- MicroPython 的目标是尽可能接近 Python 3 的语法和语义，考虑底层驱动，所以程序移植方便

MicroPython 的优势

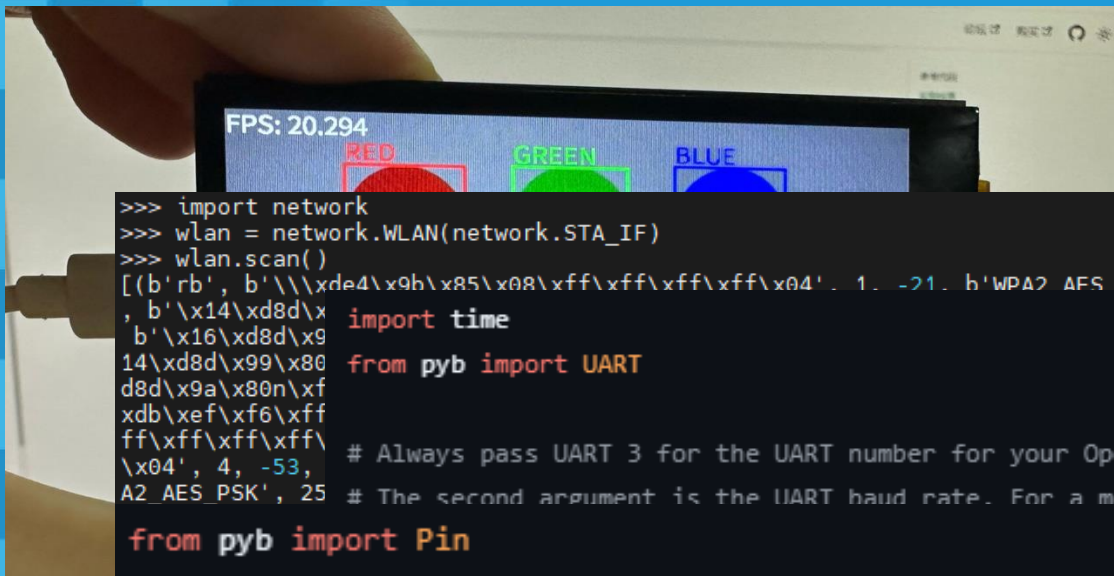
- Python 是一款容易上手的脚本语言，学习成本低，开发效率高，开发乐趣。
- 通过 MicroPython 实现硬件驱动，可以方便地移植到不同的开发板上。
- 外设与常用功能都有相应的库支持，开发效率高。

MicroPython 的应用

- MicroPython 在嵌入式系统上完整实现了 Python3 的核心功能，可以在产品开发的各个阶段给开发者带来便利。



MicroPython Peripherals



```
>>> import network
>>> wlan = network.WLAN(network.STA_IF)
>>> wlan.scan()
[('b'rb', b'\\x4e4\\x9b\\x85\\x08\\xff\\xff\\xff\\xff\\x04', 1, -21, b'WPA2 AES PSK', 255), (b'realthread',
b'\\x14\\xd8d\\x',
b'\\x16\\xd8d\\x9',
14\\xd8d\\x99\\x80',
d8d\\x9a\\x80n\\xf',
xdb\\xef\\xf6\\xff',
ff\\xff\\xff\\xff',
\\x04', 4, -53,
A2_AES_PSK', 255)]

import time
from pyb import UART

# Always pass UART 3 for the UART number for your OpenMV Cam.
# The second argument is the UART baud rate. For a more advanced UART control

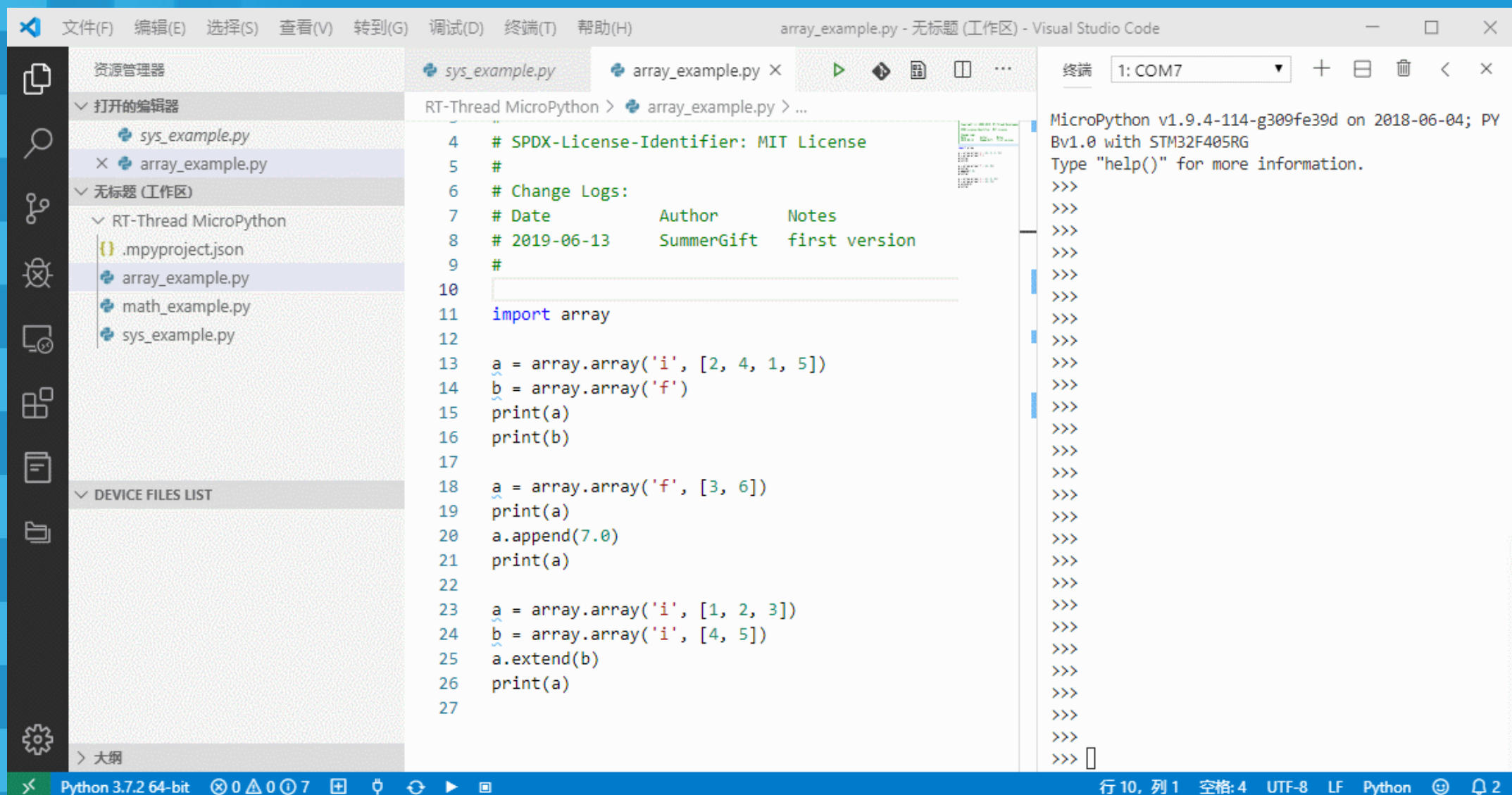
from pyb import Pin

# Connect a switch to pin 0 that will pull it low when the switch is closed.
# Pin 1 will then light up.
pin0 = Pin("P0", Pin.IN, Pin.PULL_UP)
pin1 = Pin("P1", Pin.OUT_PP, Pin.PULL_NONE)

while True:
    pin1.value(not pin0.value())
```

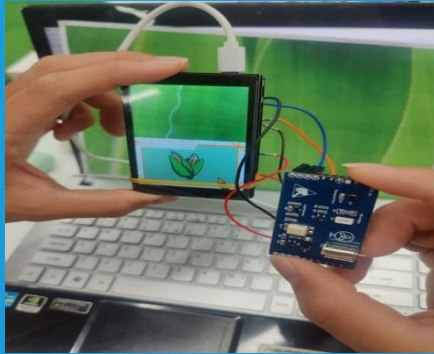
- Thread: RT-Thread 系统功能模块
- Network: 网络功能配置模块
- Machine: 硬件控制模块
 - PIN
 - I2C
 - SPI
 - UART
 - LCD
 - RTC
 - PWM
 - ADC
 - TIMER
 -

MicroPython for VSCode



MicroPython Application

玉米叶病虫害检测



飞机图像分割



人脸特征点检测



火灾检测



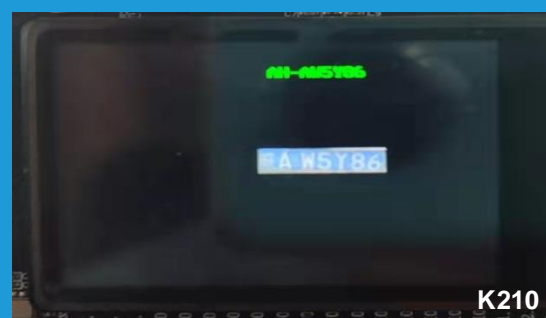
猜拳游戏



人脸表情分类



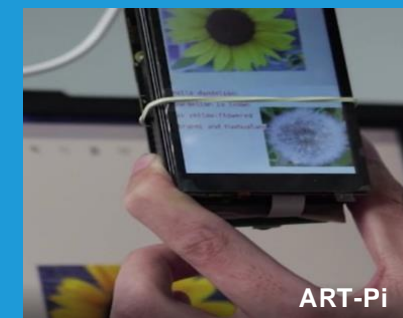
车牌识别



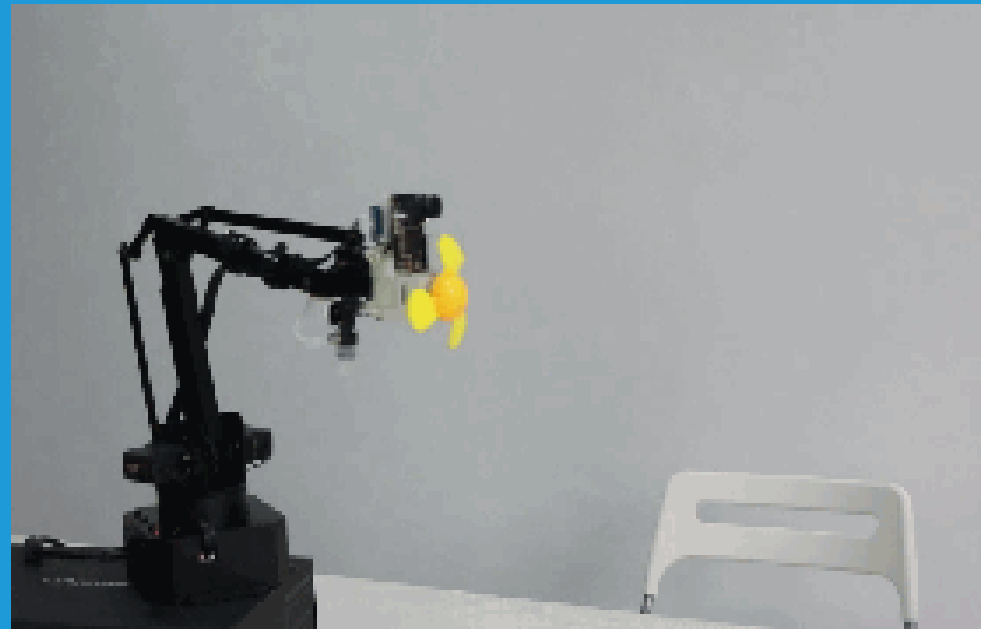
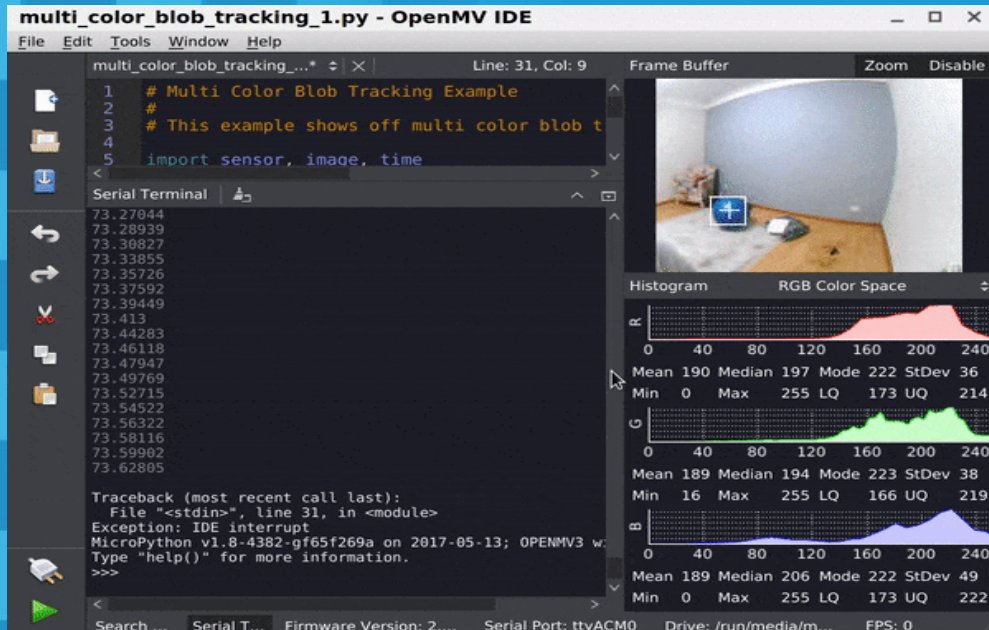
手势识别



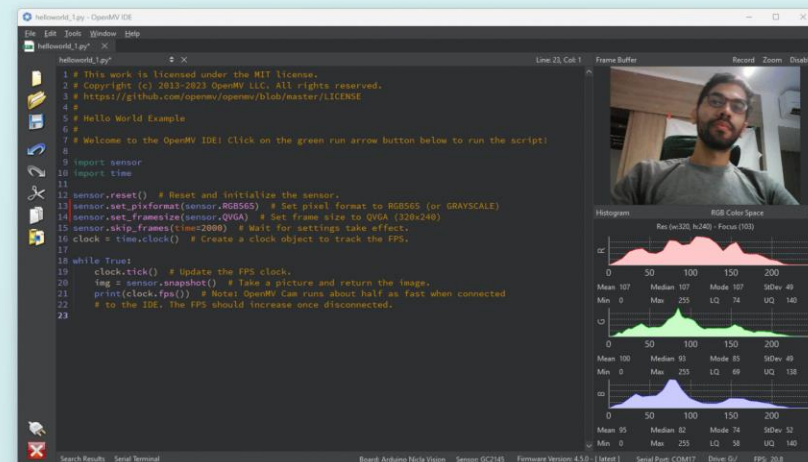
鲜花分类



OpenMV Project



- 颜色追踪
- AptilTag
- 二维码
- 条形码
- 人脸识别
- 人眼追踪
- 圆形识别
- 矩形识别
- 数字识别
- 线性回归-巡线
- 模板匹配
- 特征点追踪
- 光流
- 边缘检测
-



OpenMV

helloworld 1.py - OpenMV IDE

File Edit Tools Window Help

helloworld 1.py* Line: 18, Col: 34

```
1 import sensor, image, time
2
3 # Setup Camera
4 sensor.reset()
5 sensor.set_pixformat(sensor.RGB565)
6 sensor.set_framesize(sensor.QQVGA)
7 sensor.skip_frames(10)
8 threshold = (10, 90, -80, -30, 20, 50)
9 clock = time.clock()
10
11 # Find blobs
12 while(True):
13     clock.tick()
14     img = sensor.snapshot()
15     for b in img.find_blobs([threshold]):
16         img.draw_rectangle(b[0:4])
17         print("====\nBlob %s" % str(b))
18         print("FPS %d" % clock.fps())
19
```

Serial Terminal

```
====
Blob (88, 33, 46, 48, 839, 116, 50, 0.9424605, 1, 1)
FPS 15
====
Blob (24, 27, 34, 28, 411, 37, 37, 2.640746, 1, 1)
====
Blob (88, 33, 46, 47, 833, 115, 50, 0.898667, 1, 1)
FPS 15
====
Blob (24, 27, 34, 28, 413, 37, 37, 2.638215, 1, 1)
====
Blob (88, 33, 47, 48, 842, 116, 50, 0.9409514, 1, 1)
FPS 15
====
Blob (24, 27, 34, 28, 408, 37, 37, 2.625929, 1, 1)
====
Blob (88, 33, 47, 48, 839, 115, 50, 0.9321314, 1, 1)
FPS 15
```

Frame Buffer

Zoom Disable

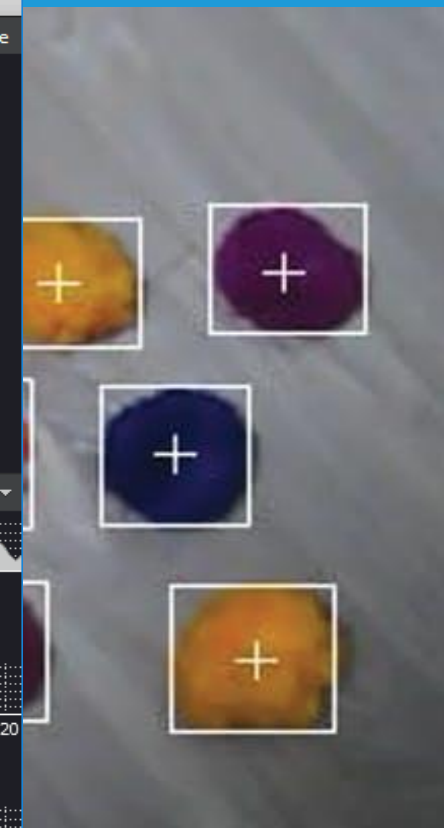
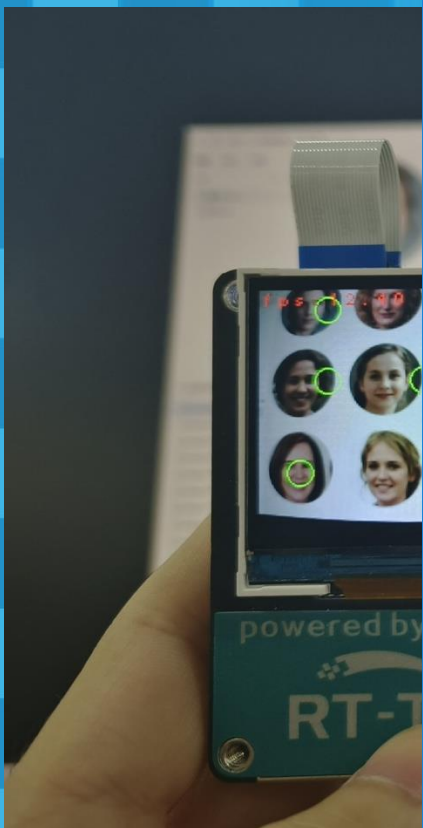
Histogram LAB Color Space

Mean	Median	Mode	StDev
76	86	88	20
Min 11	Max 100	LQ 56	UQ 92

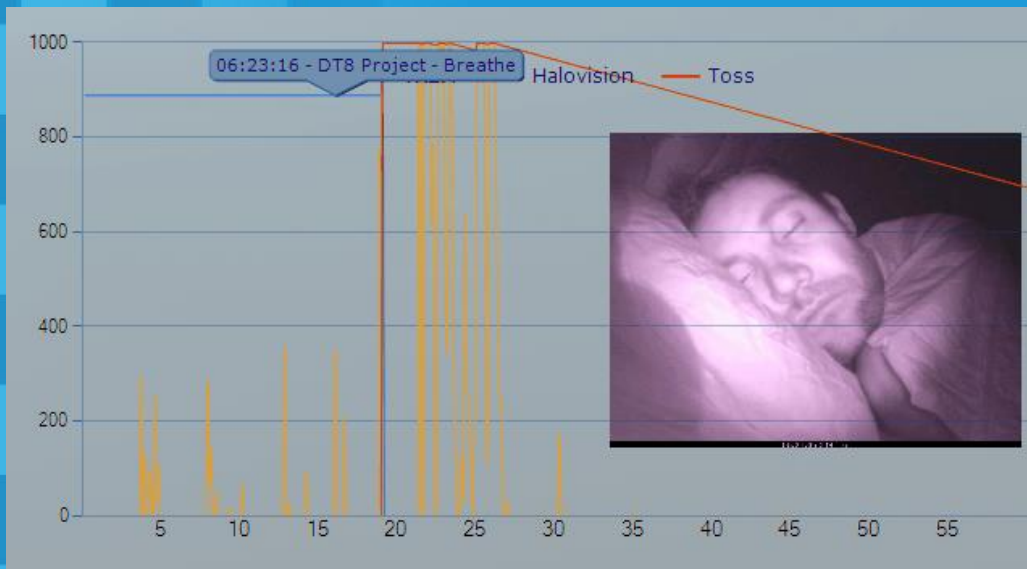
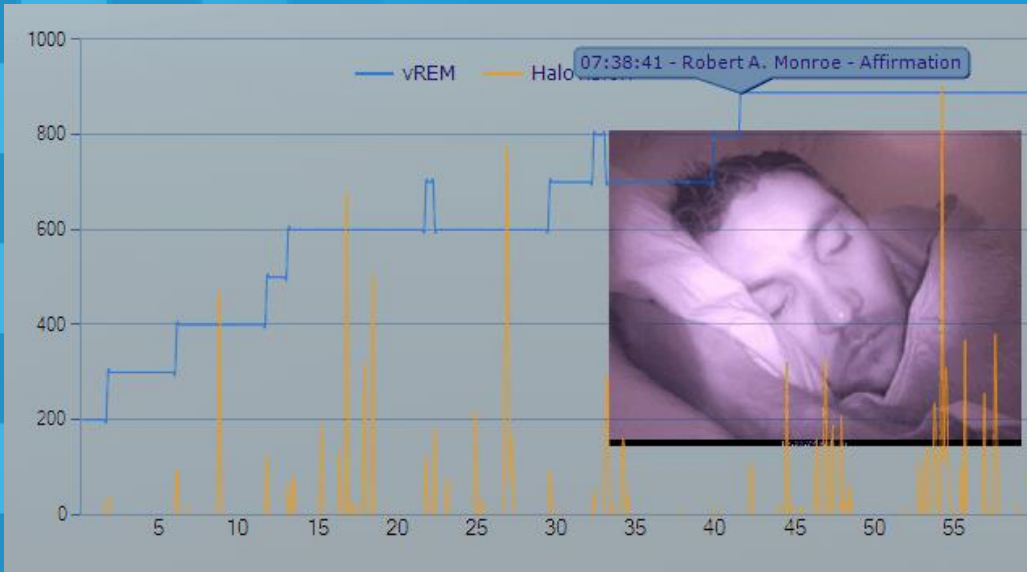
Mean	Median	Mode	StDev
2	-17	-18	40
Min -75	Max 81	LQ -21	UQ 32

Mean	Median	Mode	StDev
7	-5	-7	31
Min -103	Max 70	LQ -9	UQ 29

Firmware Version: 2.0.0 - [latest] Serial Port: COM5 Drive: I:/ FPS: 15.2



OpenMV AI



← Lucid Scribe

1 INSPEC Halovision

06:58:05.226

12 12

12 12

Home Stats Settings

← INSPEC Settings

Media Volume 100

Audio Track My Sound

Reset Audio Track

LEDs

Red ☐

Green ☐

Blue ☐

Interval 500

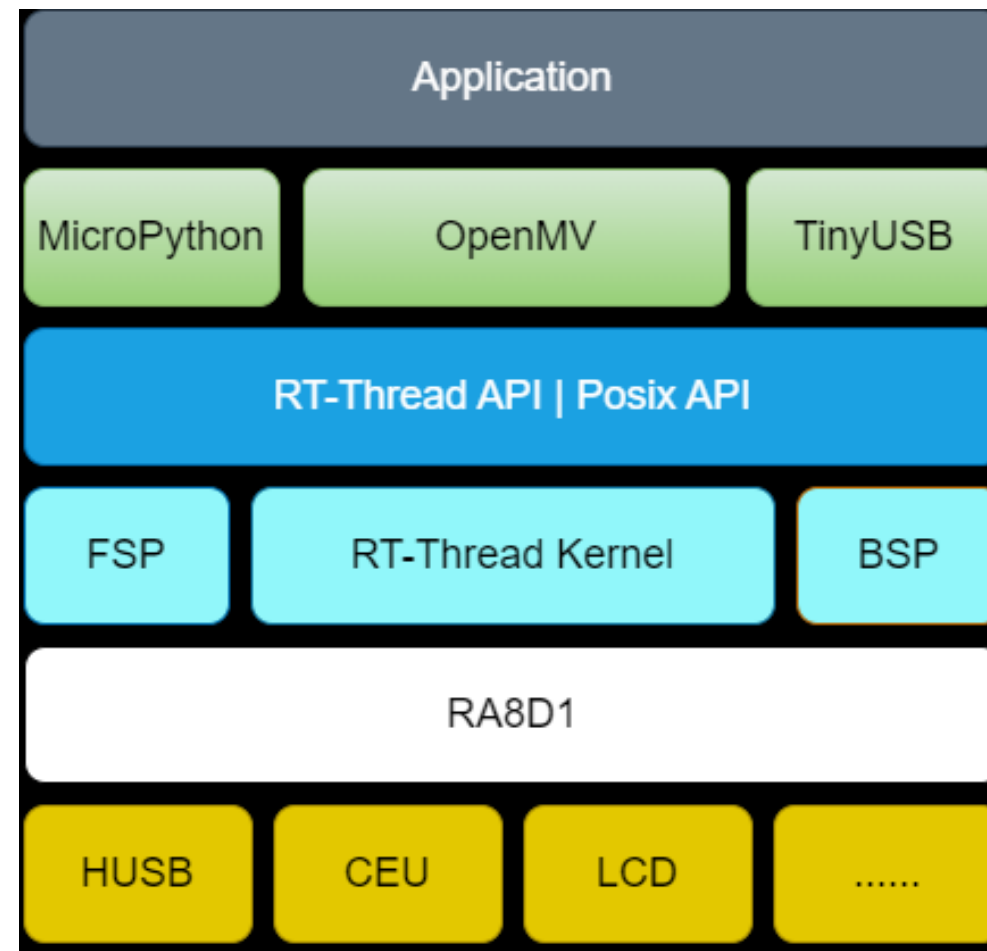
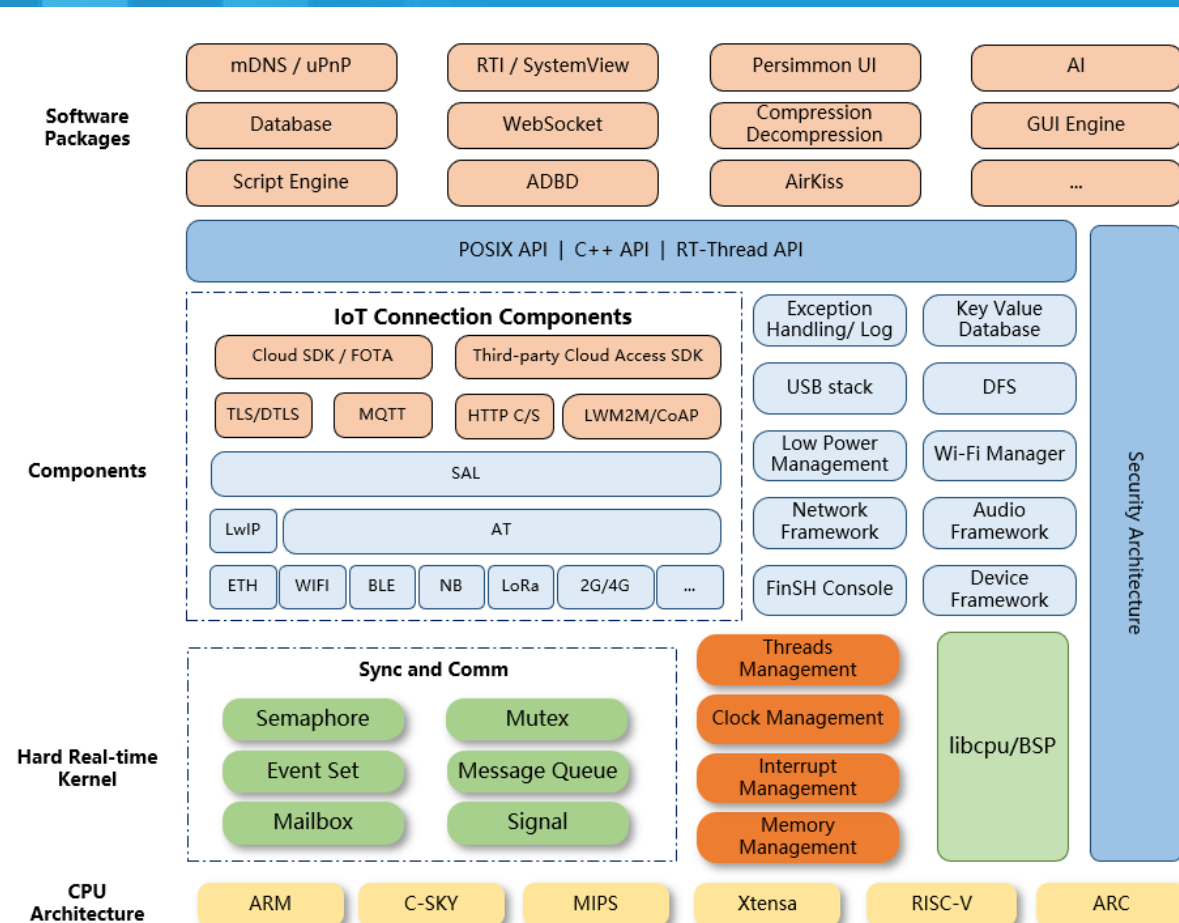
Flashes 5

Algorithm REM Detection

Home Stats Settings

from <https://openmv.io/blogs/news>

OpenMV Hardware



Video Streaming



```

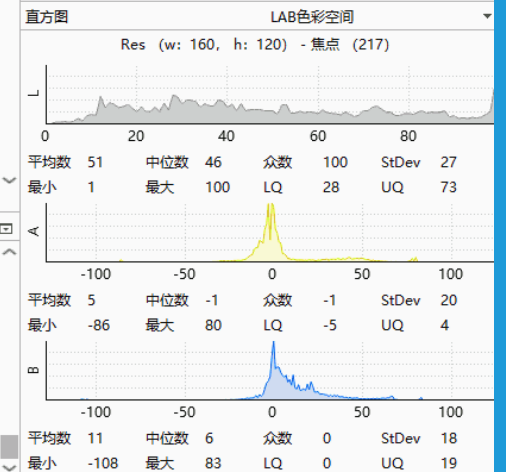
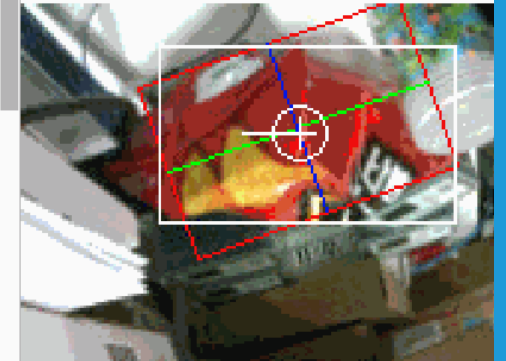
10
11 import sensor
12 # This work is licensed under the MIT license
13 # Copyright (c) 2013-2023 OpenMV LLC. All rights reserved.
14 # https://github.com/openmv/openmv/blob/master/LICENSE
15 #
16 # MJPEG Streaming
17 #
18 # This example shows off how to do MJPEG streaming
19 # Chrome, Firefox and MJpegViewer App on Android
20 # Connect to the IP address/port printed out
21
22 import sensor
23 import network
24 import socket
25 import image, time, math
26
27 SSID = "rb" # Network SSID
28 KEY = "12345678" # Network key
29 HOST = "" # Use first available interface
30 PORT = 8080 # Arbitrary non-privileged port
31
32 threshold_index = 0 # 0 for red, 1 for green
33
34 # Color Tracking Thresholds (L Min, L Max, A Min, A Max)
35 # The below thresholds track in general red, yellow, and orange
36 thresholds = [(30, 100, 15, 127, 15, 127), # Red
37               (30, 100, -64, -8, -32, 32), # Yellow
38               (0, 30, 0, 64, -128, 0)] # Green
39
40 # Init sensor
41 sensor.reset()
42 sensor.set_framesize(sensor.QQVGA)
43 sensor.set_pixformat(sensor.RGB565)
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

21.17804908752441
21.18089866638184
21.18374061584473
21.18657493591309
21.1752758026123
21.1781120300293
21.18094062805176

```



搜索结果 串行终端 板子: M4 传感器: OV5640 固件版本: 4.5.0 - [过时 - 点击此处升级] 串行端口: COM10 驱动: FPS: 20.8

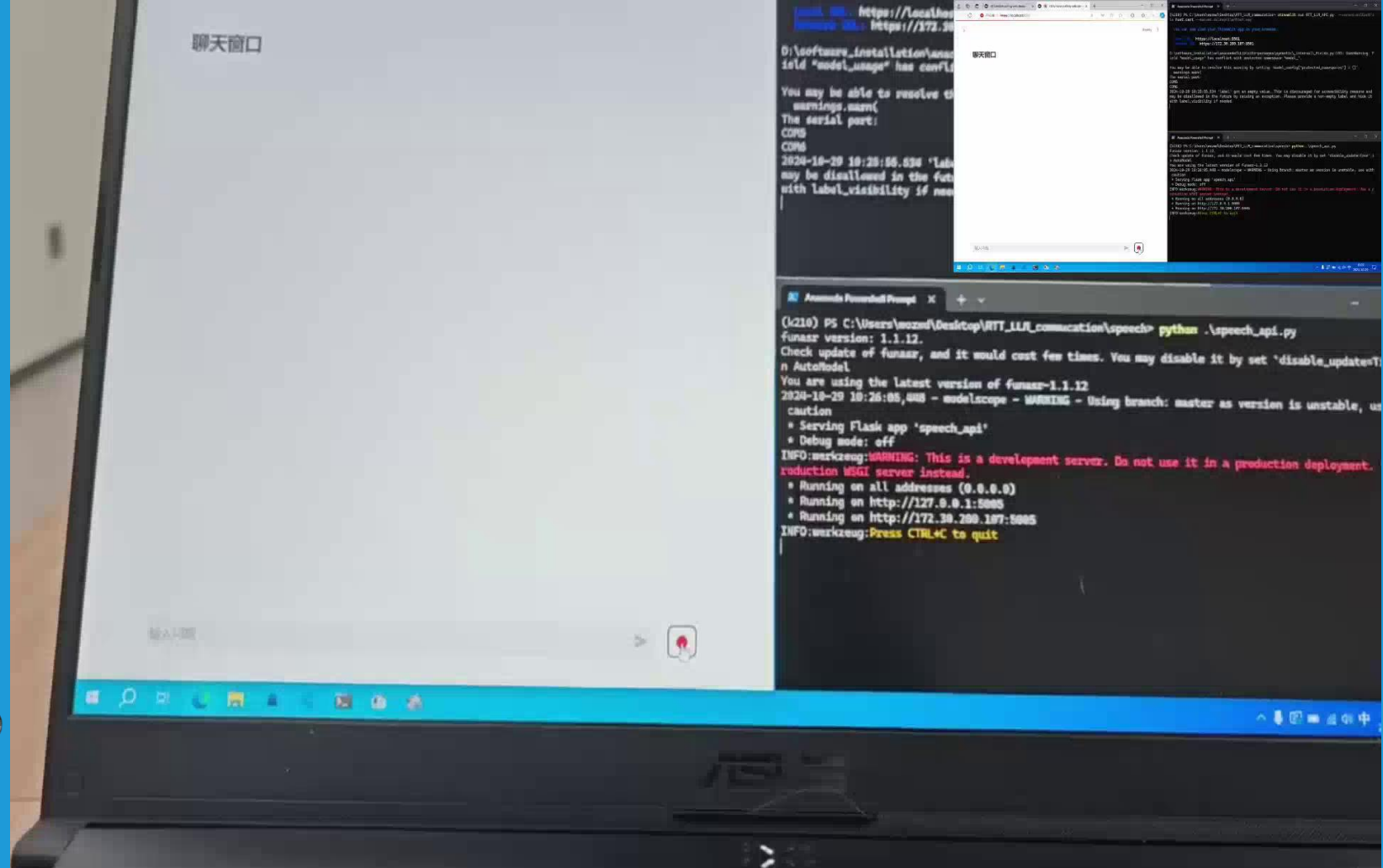
搜索结果 串行终端 板子: M4 传感器: OV5640 固件版本: 4.5.0 - [过时 - 点击此处升级] 串行端口: COM10 驱动: FPS: 0

AI-CAR

颜色追踪

颜色巡线

MicroPython AI-CAR



APP流程

```
import sensor, time, image
```

```
# Reset sensor
sensor.reset()
```

```
# Sensor settings
sensor.set_contrast(3)
sensor.set_gainceiling(16)
# HQVGA and GRAYSCALE are the best for f
sensor.set_framesize(sensor.HQVGA)
sensor.set_pixformat(sensor.GRAYSCALE)
```

```
# Load Haar Cascade
# By default this will use all stages, 1
face_cascade = image.HaarCascade("fronta
print(face_cascade)
```

```
# FPS clock
clock = time.clock()
```

```
while (True):
    clock.tick()
```

```
    # Capture snapshot
    img = sensor.snapshot()
```

```
    # Find objects.
    # Note: Lower scale factor scales-do
    # Higher threshold results in a high
    objects = img.find_features(face_cas
```

```
    # Draw objects
    for r in objects:
        img.draw_rectangle(r)
```

```
    # Print FPS.
    # Note: Actual FPS is higher, stream
    print(clock.fps())
```

```
sensor.reset() # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA) # Set frame size to QVGA (320x240)
sensor.set_windowing((240, 240)) # Set 240x240 window.
sensor.skip_frames(time=2000) # Let the camera adjust.
```

```
net = tf.load('<object_detection_network>', load_to_fb=True)
labels = []
```

```
try: # Load labels if they exist
    labels = [line.rstrip('\n') for line in open("labels.txt")]
except:
    pass
```

```
colors = [ # Add more colors if you are detecting more than 7 types of classes at once.
    (255, 0, 0),
    (0, 255, 0),
    (255, 255, 0),
    (0, 0, 255),
    (255, 0, 255),
    (0, 255, 255),
    (255, 255, 255),
]
```

```
clock = time.clock()
while(True):
    clock.tick()
```

```
    img = sensor.snapshot()
```

```
    # detect() segments an object using the provided segmentation model. This produces mutiple
    # grayscale images per object class that we are trying to detect. detect() then runs
    # find_blobs() internally on the segmented images to find all blob locations and then returns
    # the bound boxes of all blobs found per object class. So, detect() returns a list of lists of
    # classification objects and the respective confidence level.
```

```
for i, detection_list in enumerate(net.detect(img, thresholds=[(128, 255)])):
    if (i < len(labels)):
        print("***** %s *****" % labels[i])
    for d in detection_list:
        print(d)
        img.draw_rectangle(d.rect(), color=colors[i])
```

le is faster

get their rotation angle in degrees.
on angle.

and `y2()` methods to get their end-points
as one 4 value tuple for `draw_line()`.

for 2.8mm lens...

the image are found. Only lines with
than `threshold` are detected...

the image contributes a magnitude value
is the magintude for that line. Then
are added togheter. Note that `threshold`
before merging. To see the magnitude of
`rho_margin` to 0...

merging similar lines. If two lines
s than the margins then they are merged.

```
theta_margin = 25, rho_margin = 25):
    heta() <= max_degree):
    255, 0, 0))
```

Thanks for watching.

