



StarFive
赛昉科技

在昉·星光 2上实现按键触发LED点阵显示

Python语言版本

应用指南

版本：1.2

日期：2025/08/07

Doc ID: VisionFive 2-ANCH-010

法律声明

阅读本文件前的重要法律告知。

版权注释

版权 ©上海赛昉半导体科技有限公司，2025。版权所有。

本文档中的说明均基于“视为正确”提供，可能包含部分错误。内容可能因产品开发而定期更新或修订。上海赛昉半导体科技有限公司（以下简称“赛昉科技”）保留对本协议中的任何内容进行更改的权利，恕不另行通知。

赛昉科技明确否认任何形式的担保、解释和条件，无论是明示的还是默示的，包括但不限于适销性、特定用途适用性和非侵权的担保或条件。

赛昉科技无需承担因应用或使用任何产品或电路而产生的任何责任，并明确表示无需承担任何及所有连带责任，包括但不限于间接、偶然、特殊、惩戒性或由此造成的损害。

本文件中的所有材料受版权保护，为赛昉科技所有。不得以任何方式修改、编辑或断章取义本文件中的说明，本文件或其任何部分仅限用于内部使用或教育培训。使用文件中包含的说明，所产生的风险由您自行承担。赛昉科技授权复制本文件，前提是您保留原始材料中包含的所有版权声明和其他相关声明，并严格遵守此类条款。本版权许可不构成对产品或服务的许可。

联系我们：

地址：中国（上海）自由贸易试验区盛夏路61弄张润大厦2号电梯楼层5层（实际楼层4层）06室

网站：<http://www.starfivetech.com>

邮箱：sales@starfivetech.com（销售） support@starfivetech.com（支持）

目录

表格清单.....	4
插图清单.....	5
法律声明.....	2
前言.....	6
1. 产品简介.....	7
1.1. 40-Pin GPIO Header定义.....	7
2. 准备.....	8
2.1. 运行环境要求.....	8
2.2. 准备硬件.....	8
2.2.1. 连接硬件.....	9
2.3. 准备软件.....	10
3. 执行演示代码.....	13
4. 演示源代码.....	15
5. 资源下载.....	18
6. 立即购买.....	19



Starfive
赛昉科技

表格清单

表 0-1 修订历史.....	6
表 2-1 硬件准备.....	8
表 2-2 将按键连接到40-Pin Header上.....	9
表 2-3 连接MAX7219和40-Pin Header.....	9



插图清单

图 1-1 40-Pin GPIO Header定义.....	7
图 2-1 面包板概述图.....	9
图 2-2 连接按键、MAX7219和40-Pin Header.....	10
图 3-1 倒计时和赛昉科技徽标.....	14



前言

关于本指南和技术支持信息

关于本手册

该应用说明提供使用昉·星光 2的GPIO Pin，实现按键触发LED点阵显示倒计时和赛昉科技徽标。






修订历史

表 0-1 修订历史

版本	发布日期	修订历史
1.2	2025/08/07	更新了以下章节： <ul style="list-style-type: none">• 运行环境要求 (第 8页)• 准备软件 (第 10页)• 执行演示代码 (第 13页)
1.1	2023/06/08	更新了 准备软件 (第 10页) 中安装VisionFive.gpio包的方法。
1.0	2023/05/31	首次发布。

注释和注意事项

本指南中可能会出现以下注释和注意事项：

-  **提示：**
建议如何在某个主题或步骤中应用信息。
-  **注：**
解释某个特例或阐释一个重要的点。
-  **重要：**
指出与某个主题或步骤有关的重要信息。
-  **警告：**
表明某个操作或步骤可能会导致数据丢失、安全问题或性能问题。
-  **警告：**
表明某个操作或步骤可能导致物理伤害或硬件损坏。

1. 产品简介

该应用说明提供使用昉·星光 2的GPIO Pin，实现按键触发LED点阵显示倒计时和赛昉科技徽标。



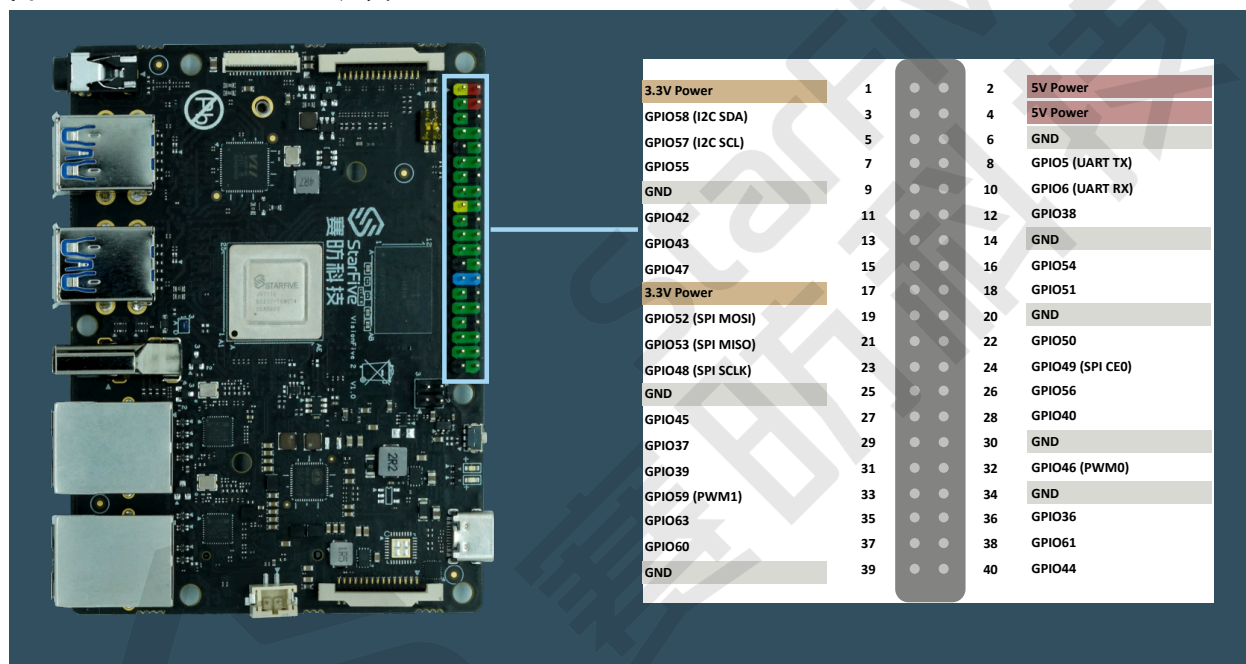
提示：

VisonFive.gpio兼容Rpi.GPIO命令，即Rpi.GPIO python演示可以在昉·星光 2上运行。此外，API `add_event_detect()`的回调函数较Rpi.GPIO做了优化，回调函数增加一个参数`edge_type`。因此Rpi.GPIO涉及到回调函数的python演示需要手动修改，增加一个`edge_type`即可。

1.1. 40-Pin GPIO Header定义

下图显示了40-pin GPIO Header的位置：

图 1-1 40-Pin GPIO Header定义



注：

功能复用pin脚已初始化，不可作为通用GPIO使用。

2. 准备

在执行演示程序之前，务必确认已准备好以下项目：

2.1. 运行环境要求

该演示运行环境要求如下：

- Linux内核版本：Linux 6.6
- 操作系统：Debian 13
- 硬件版本：昉·星光 2
- SoC：昉·惊鸿-7110

2.2. 准备硬件

在执行演示程序之前，请务必准备以下硬件：

表 2-1 硬件准备

类型	M/O*	项目	注释
通用	M	昉·星光 2 单板计算机	-
通用	M	<ul style="list-style-type: none">• 容量不低于32 GB的Micro-SD卡• Micro-SD卡读卡器• 计算机 (Windows/Mac OS/Linux)• USB转串口转换器 (3.3 V I/O, 带线)• 以太网电缆• 电源适配器 (5 V/ 3 A)• USB Type-C数据线	上述项目用于将Debian OS烧录到Micro-SD上。
点阵演示 (LED)	M	<ul style="list-style-type: none">• 一个LED灯• 一个面包板• 7根公母跳线• 一个4-pin按键• MAX7219串行点阵显示模块	<ul style="list-style-type: none">• LED代表发光二极管，当电流通过时发光。长腿（称为“阳极”）始终连接到电路的正极电源。短腿（称为“阴极”）连接到电源的负极，称为“接地”。• 面包板：请参阅下文介绍。



注：

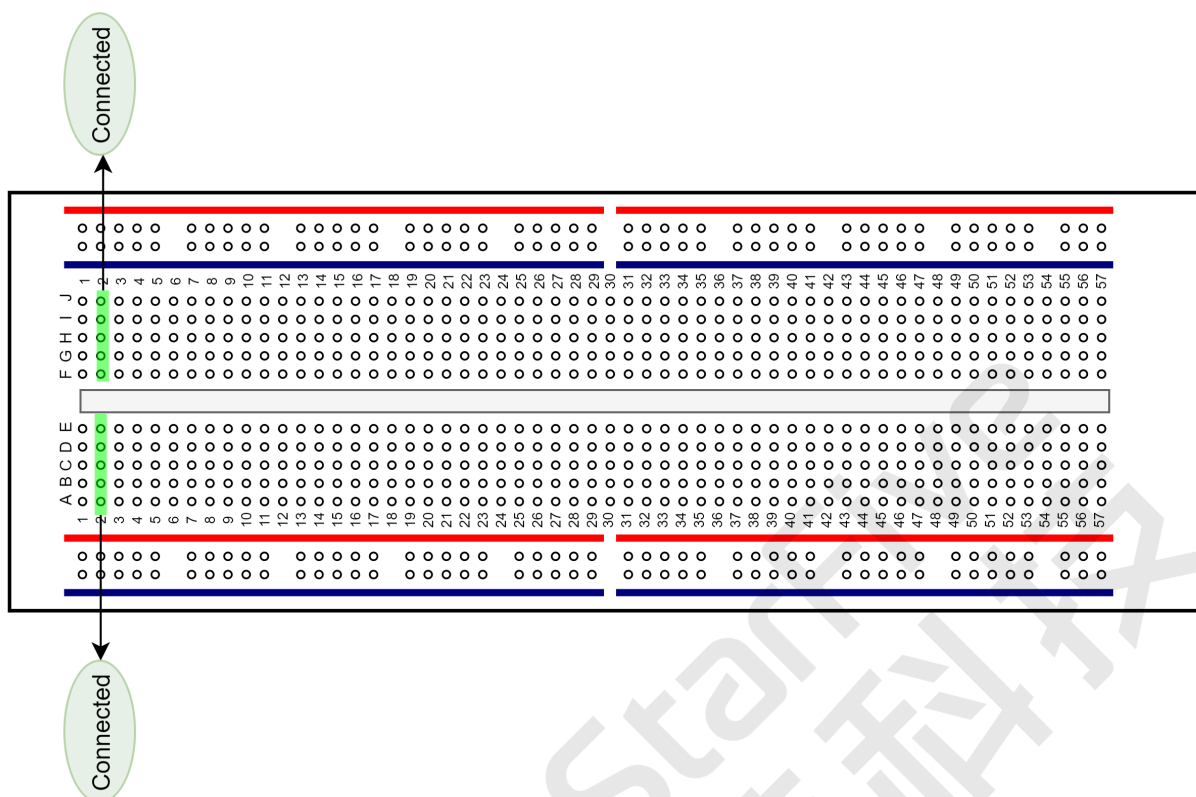
*: M：必须。O：可选

面包板介绍

面包板是一种连接各种电子元件的方法，免去了焊接的工序。面包板通常用于制作PCB前的电路测试。如下图所示，面包板的顶部和底部各有两行线，通常用于主电源连接，蓝线用作 负极，红线用作正极。此外红线和蓝线被分割为两段，每段所覆盖的小孔已经连通。

面包板的每一列（A到E，F到J）的行的的小孔是连接的；每一行（1到57）是没有连接的。

图 2-1 面包板概述图



2.2.1. 连接硬件

将按键的Pin ①和Pin ②分别连接到面包板上第19行和第21行的小孔上，然后将按键的Pin ③连接到昉·星光 2的Pin 39，Pin ④连接到昉·星光 2的Pin 37。以下表格和图片描述了如何将按键连接到40-Pin GPIO Header上：

表 2-2 将按键连接到40-Pin Header上

按键	40-Pin GPIO Header	
	Pin号	Pin名
Pin ④	37	GPIO60
Pin ③	39	GND

表 2-3 连接MAX7219和40-Pin Header

MAX7219	40-Pin GPIO Header	
	Pin号	Pin名
VCC	2	5V Power
GND	34	GND
DIN	40	GPIO44
CS	38	GPIO61
CLK	36	GPIO36


```
sudo apt install python3-venv
python3 -m venv myvenv
```



注：
可自定义 "myvenv" 名称。

5. 在昉·星光 2 Debian上执行pip命令，以安装VisionFive.gpio包：



注：
由于pypi.org官网尚不支持上传RISC-V平台的whl安装包，不能直接执行python3 -m pip install VisionFive.gpio命令在线安装，因此请按照以下步骤安装VisionFive.gpio包。

a. 执行以下命令，在新创建的虚拟环境中安装依赖包：

```
sudo apt install libxml2-dev libxslt-dev
source ./myvenv/bin/activate
python3 -m pip install requests wget bs4
```

b. 执行以下命令，运行安装脚本Install_VisionFive_gpio.py：

```
python3 Install_VisionFive_gpio.py
```

安装脚本代码如下：

```
import requests
import wget
import sys
import os
from bs4 import BeautifulSoup

def parse_data(link_addr, class_type, key_str):
    req = requests.get(url=link_addr)
    req.encoding = "utf-8"
    html = req.text
    soup = BeautifulSoup(req.text, features="html.parser")
    package_version = soup.find(class_type, class_=key_str)
    dd = package_version.text.strip()
    data = dd.split()
    return data

def parse_link(link_addr, class_type, key_str):
    version_list = []
    req = requests.get(url=link_addr)
    req.encoding = "utf-8"
    html = req.text
    soup = BeautifulSoup(req.text, features="html.parser")
    search_data = soup.find_all(class_type, class_=key_str)
    for i in range(0, len(search_data)):
        search_data[i] = search_data[i].find("a").get("href")
        version_list.append(search_data[i].split("cp")[-1].split("-")[0])

    python_version = sys.version
    python_version = python_version.split(".")[0] + python_version.split(".")[1]

    for i in range(0, len(search_data)):
        if python_version == version_list[i]:
            return search_data[i]

    return search_data[0]

def get_dl_addr_page():
    link_address = "https://pypi.org/project/VisionFive.gpio/#history"
    key_str = "release version"
    class_key = "p"
    data_get = parse_data(link_address, class_key, key_str)
    latest_version = data_get[0]
    dl_addr_page
    = "https://pypi.org/project/VisionFive.gpio/{}/#files".format(latest_version)
```



```
        return dl_addr_page

def get_dl_addr_of_latest_version(link_addr):
    key_str = "card file card"
    class_key = "div"
    addr_get = parse_link(link_addr, class_key, key_str)

    return addr_get

def main():
    dl_addr_p = get_dl_addr_page()
    whl_dl_addr = get_dl_addr_of_latest_version(dl_addr_p)

    whl_name = whl_dl_addr.split("/")[-1]
    whl_name_suffix = os.path.splitext(whl_name)[-1]
    whl_name_prefix = os.path.splitext(whl_name)[0]
    whl_name_prefix_no_platform = whl_name_prefix[0: len(whl_name_prefix) - 3]
    new_platform = "linux_riscv64"

    rename_whl_name = "{}{}{}".format(whl_name_prefix_no_platform, new_platform,
whl_name_suffix)

    wget.download(whl_dl_addr, out=rename_whl_name)

    os.system("pip install " + rename_whl_name)
    os.system("rm -rf " + rename_whl_name)

if __name__ == '__main__':
    sys.exit(main())
```

c. (可选) 退出 Python3 虚拟环境。

```
deactivate
```

3. 执行演示代码

执行以下操作，以在昉·星光 2的Debian系统上运行演示代码：

1. 找到测试代码`edge_detection_with_LED_Matrix.py`所在的目录：

- a. 进入 Python3 虚拟环境：

```
source ./myvenv/bin/activate
```

- b. 执行以下命令安装依赖：

```
python3 -m pip install pillow
```

- c. 执行以下命令以获取`VisionFive.gpio`所在的目录：

```
python3 -m pip show VisionFive.gpio
```

示例结果：

```
Location: /home/user/myvenv/lib/python3.11/site-packages
```



注：

实际输出取决于应用的安装方式。

- d. 执行以下命令进入目录，例如上一步所示的 `/home/user/myvenv/lib/python3.11/site-packages`：

```
cd /home/user/myvenv/lib/python3.11/site-packages
```

- e. 执行以下命令进入`sample-code`目录：

```
cd ../VisionFive/sample-code/
```

2. 在`sample-code`目录下，执行以下命令以运行演示代码：

```
ssudo python edge_detection_with_LED_Matrix.py
```

或者，您也可以执行以下命令：

```
sudo python3 edge_detection_with_LED_Matrix.py
```

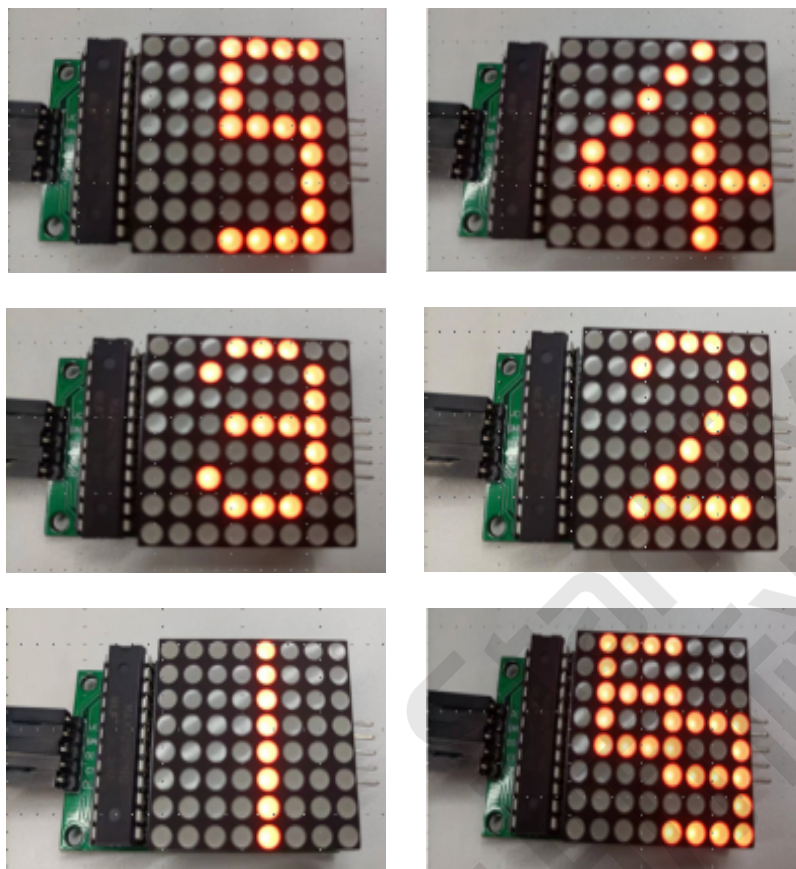
结果：

- 终端显示如下：

```
# python3 edge_detection_with_LED_Matrix.py
*-----*
Please press the key on pin 37 to launch !!!
```

- 根据提示按下按键后，LED点阵将会从5开始倒计时显示5、4、3、2、1，最后闪烁显示赛昉科技的徽标，效果图如下：

图 3-1 倒计时和赛昉科技徽标



3. (可选) 退出 Python3 虚拟环境。

```
deactivate
```

4. 演示源代码

本演示中的资源代码仅作为参考。

edge_detection_with_LED_Matrix.py:

```
'''
Step 1:
Please make sure the LED Dot Matrix is connected to the correct pins.
The following table describes how to connect LED Dot Matrix to the 40-pin header.
-----
MAX7219      Pin Number  Pin Name
-----
VCC          2           5V Power
GND          34          GND
DIN          40          GPIO44
CS           38          GPIO61
CLK          36          GPIO36
-----

Step 2:
Please make sure the button is connected to the correct pins.
The following table describes how to connect the button to the 40-pin header.
-----
button      Pin Number  Pin Name
-----
one end     37          GPIO60
The other end 39          GND
-----
'''

import VisionFive.gpio as GPIO
import sys
import time

DIN = 40
CS = 38
CLK = 36
# Configure the direction of DIN, CS, and CLK as out.
GPIO.setup(DIN, GPIO.OUT)
GPIO.setup(CS, GPIO.OUT)
GPIO.setup(CLK, GPIO.OUT)

# Display logo data.
buffer = ['01111000', '01000000', '01111000', '01001111', '01111001', '00001111', '00000001', '00001111']
# Display arabic numeral 5.
buffer_5 = ["00011110", "00010000", "00010000", "00011110", "00000010", "00000010", "00000010", "00011110"]
# Display arabic numeral 4.
buffer_4 = ["00000100", "00001000", "00010000", "00100100", "01000100", "01111111", "00000100", "00000100"]
# Display arabic numeral 3.
buffer_3 = ["00011100", "00100010", "00000010", "00011110", "00000010", "00100010", "00011100", "00000000"]
# Display arabic numeral 2.
buffer_2 = ["00011100", "00100010", "00000010", "00000100", "00001000", "00010000", "00111110", "00000000"]
# Display arabic numeral 1.
buffer_1 = ["00001000", "00001000", "00001000", "00001000", "00001000", "00001000", "00001000", "00001000"]

# LED turn off data.
buffer_off = ['0', '0', '0', '0', '0', '0', '0', '0']

key_pin = 37

def sendbyte(bytedata):
    for bit in range(0, 8):
        if ((bytedata << bit) & 0x80):
            GPIO.output(DIN, GPIO.HIGH)
        else:
            GPIO.output(DIN, GPIO.LOW)

    # Configure the voltage level of CLK as high.
    GPIO.output(CLK, GPIO.HIGH)
    # Configure the voltage level of CLK as low.
    GPIO.output(CLK, GPIO.LOW)
```

```

def WriteToReg(regaddr, bytedata):
    # Configure the voltage level of cs as high.
    GPIO.output(CS, GPIO.HIGH)
    # Configure the voltage level of led_pin as low.
    GPIO.output(CS, GPIO.LOW)
    GPIO.output(CLK, GPIO.LOW)
    sendbyte(regaddr)
    sendbyte(bytedata)
    GPIO.output(CS, GPIO.HIGH)

def disp_clean():
    time.sleep(0.1)
    for i in range(0, 8):
        # Write data to register address. Finally the LED matrix displays StarFive logo.
        WriteToReg(i+1, int(buffer_off[i], 2))
    time.sleep(1)

def disp_numeral_5_to_1():
    for id in range(5, 0, -1):
        buffer_name = "buffer_{}".format(id)
        list_buffer = eval(buffer_name)
        for j in range(0, 8):
            # Write data to the register address. Finally the LED matrix displays with numeral from 5 to 1.
            WriteToReg(j+1, int(list_buffer[j], 2))
        time.sleep(1)
    for j in range(0, 8):
        # Write data to the register address. Finally turn off the LED matrix.
        WriteToReg(j+1, int(buffer_off[j], 2))
    time.sleep(0.1)

def flash_logo():
    for loop in range(0, 5):
        for j in range(0, 8):
            # Write data to the register address. Finally turn off the LED matrix.
            WriteToReg(j+1, int(buffer_off[j], 2))
        time.sleep(0.1)
    for j in range(0, 8):
        # Write data to the register address. Finally the LED matrix displays with StarFive logo.
        WriteToReg(j+1, int(buffer[j], 2))
    time.sleep(0.1)

def WriteALLReg():
    # clean screen
    disp_clean()

    # display numeral from 5 to 1
    disp_numeral_5_to_1()

    # falsh starfive logo.
    flash_logo()

def initData():
    WriteToReg(0x09, 0x00) #Set the decode mode.
    WriteToReg(0x0a, 0x03) #Set the brightness.
    WriteToReg(0x0b, 0x07) #Set the scan limitation.
    WriteToReg(0x0c, 0x01) #Set the power mode.
    WriteToReg(0x0f, 0x00)

# the callback function for edge detection
def detect(pin, edge_type):
    WriteALLReg()

def main():
    # Configure the direction of key_pin as input.
    GPIO.setup(key_pin, GPIO.IN)
    # Both edge rising and falling can be detected, also set bouncetime(unit: millisecond) to avoid jitter
    GPIO.add_event_detect(key_pin, GPIO.FALLING, callback=detect, bouncetime=2)

    initData()

```

```
print("*-----*")
print("Please press the key on pin {} to launch !!!".format(key_pin))

while True:
    m = 1

if __name__ == "__main__":
    sys.exit(main())
```



5. 资源下载

点击本栏找到所有的代码下载资源。

本页包括所有赛昉科技提供的代码下载资源。

- [RVspace Wiki](#)
- [应用中心](#)
- [文档中心](#)
- [技术论坛](#)
- [昉·星光 2 GitHub代码仓](#)
- [昉·星光 2 Debian操作系统下载](#)
- [代码下载 \(赛昉科技官方GitHub页面\)](#)
- [所有开源技术文档](#)



StarFive
赛昉科技

6. 立即购买

点击本栏获取在线购买链接和配件购买链接。

购买单板计算机

点击以下页面，您可以找到所在地区的经销商，或覆盖全球的销售渠道，以购买昉·星光 2 单板计算机。

- [购买昉·星光 2 开发板](#)

购买配件

点击以下页面，您可以找到所有昉·星光 2 单板计算机已验证适配的配件及其购买链接。

- [购买配件](#)

