StarFive
赛昉科技

使用昉·星光点亮LED矩阵

Python语言版本

应用说明

版本：1.0

日期：2022/07/29

Doc ID: VisionFive-ANCH-012-1.0

法律声明

阅读本文件前的重要法律告知。

版权注释

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： 浦东新区盛夏路61弄张润大厦2号楼502，上海市，201203，中国

网站： http://www.starfivetech.com

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

前言

关于本指南和技术支持信息。

关于本手册

本应用说明提供使用昉·星光的GPIO引脚，通过Python示例程序在LED矩阵MAX7219上点亮赛昉科技徽标的步骤。

修订历史

表 0-1 修订历史

版本	发布说明	修订
V1.0	2022/07/29	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某个重要的点。
- **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。

目录

表格清单

插图清单

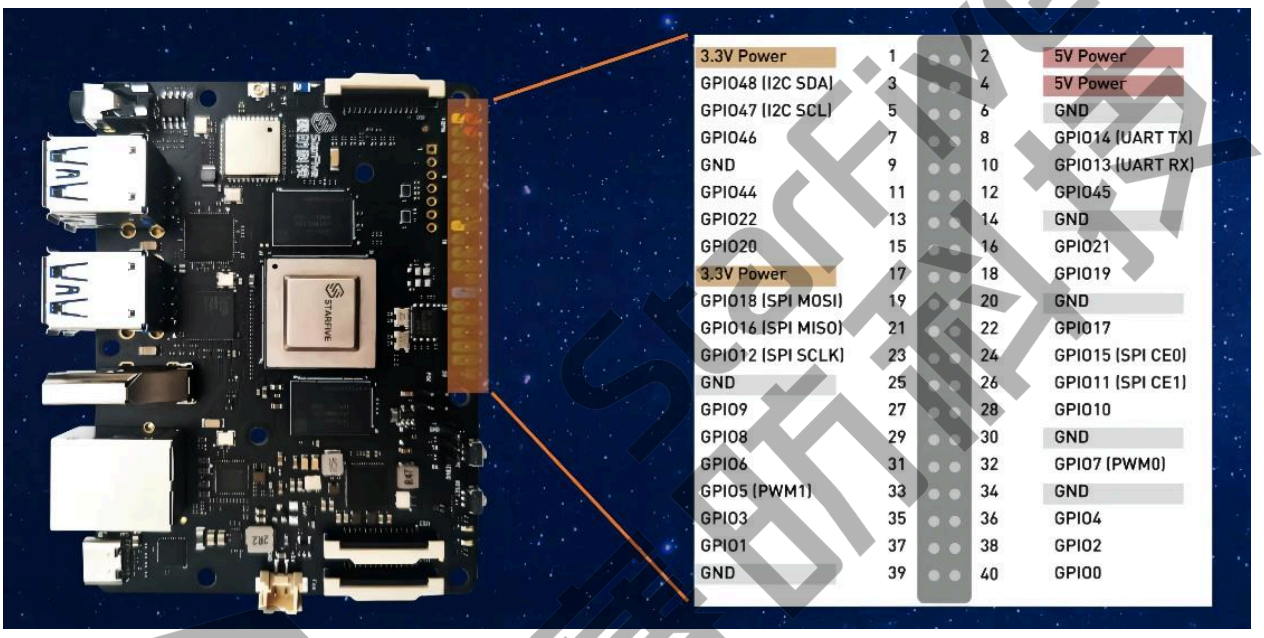
1. 介绍

本应用说明提供使用昉·星光的GPIO引脚，通过Python示例程序在LED矩阵MAX7219上点亮赛昉科技徽标的步骤。

1.1. 40-Pin Header定义

下图以昉·星光开发板为例说明40-Pin Header的位置：

图 1-1 40-Pin定义



2. 准备

在执行演示代码前，请确保您已准备好以下事项：

2.1. 准备硬件

在执行演示程序前，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	赛昉科技 单板计算机	可使用以下单板计算机： • 星光板 • 昉·星光
通用	M	• 16GB（或更大）的Micro SD卡 • Micro SD卡读卡器 • 计算机（PC/Mac/Linux） • USB转串口转换器（3.3 V I/O） • 网线 • 电源适配器 • USB Type-C数据线	上述项目用于将Fedora OS烧录到Micro SD上。
GPIO演示(LED矩阵)	M	MAX7219 串行点阵显示模块（带5路母对母杜邦电缆）	-

注：
*: M：必须。O：可选

2.1.1. 连接硬件

以下图表描述了如何将MAX7219连接到40-Pin Header上：

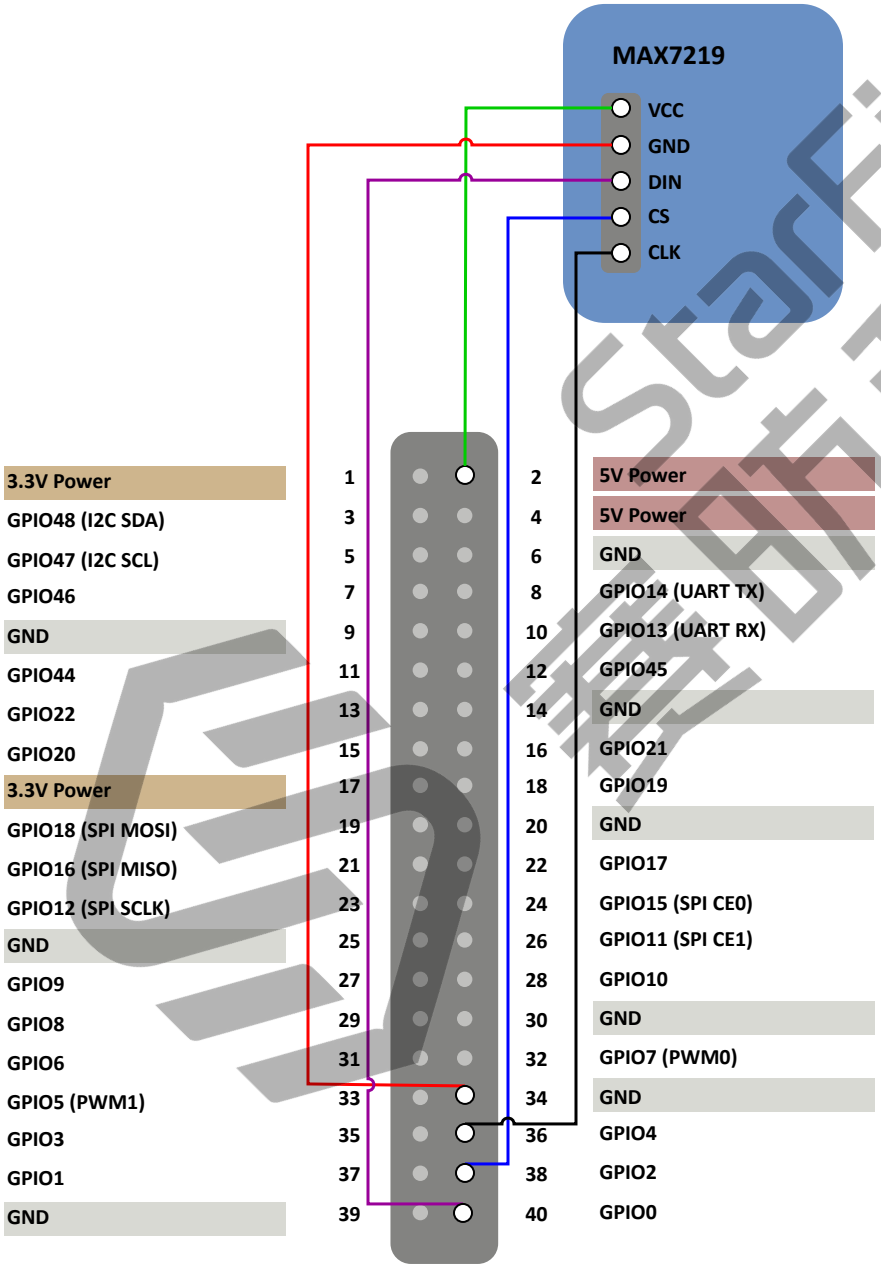
表 2-2 将MAX7219连接到40-Pin Header上

MAX7219	40-Pin GPIO Header	
	引脚序号	引脚名
VCC	2	5V Power
GND	34	GND

表 2-2 将MAX7219连接到40-Pin Header上 (续)

MAX7219	40-Pin GPIO Header	
	引脚序号	引脚名
DIN	40	GPIO0
CS	38	GPIO2
CLK	36	GPIO4

图 2-1 将MAX7219连接到40-Pin Header上



2.2. 准备软件

确认按照以下步骤进行操作:

1. 按照《昉·星光单板计算机快速入门指南》中的“将Fedora烧录到Micro SD上”章节，将Fedora OS烧录到Micro SD卡上。
2. 登录Fedora并确保昉·星光已联网。有关详细说明，请参阅《昉·星光单板计算机快速入门指南》中“通过以太网使用SSH登录”或“使用USB转串口转换器连接并登录”章节。
3. 在昉·星光Fedora上执行`pip`命令，以安装`VisionFive.gpio`包:

   ```
   sudo pip install VisionFive.gpio
   ```

   或者，您也可以执行以下命令:

   ```
   sudo pip3 install VisionFive.gpio
   ```

4. （可选）如果您将源代码复制到昉·星光Fedora的本地目录下，请在源代码目录下执行以下命令:

   提示:
   点击以下链接可下载源代码: VisionFive.gpio。

   ```
   sudo yum install python-devel python3-devel
   sudo python setup.py install
   ```

   或者，您也可以执行以下命令:

   ```
   sudo python3 setup.py install
   ```

3. 执行演示代码

在昉·星光的Fedora上执行以下步骤，运行演示代码：

1. 找到测试代码`LED_Matrix.py`所在的目录：

   a. 执行以下命令以获取`VisionFive.gpio`所在的目录：

   ```
   pip show VisionFive.gpio
   ```

   示例结果：

   ```
   Location: /usr/local/lib64/python3.9/site-packages
   ```

   注：

   实际输出取决于应用的安装方式。

   b. 如前一步输出中所示，执行以下操作进入目录`/usr/local/lib64/python3.9/site-packages`：

   ```
   cd /usr/local/lib64/python3.9/site-packages
   ```

   c. 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：

   ```
   sudo python LED_Matrix.py
   ```

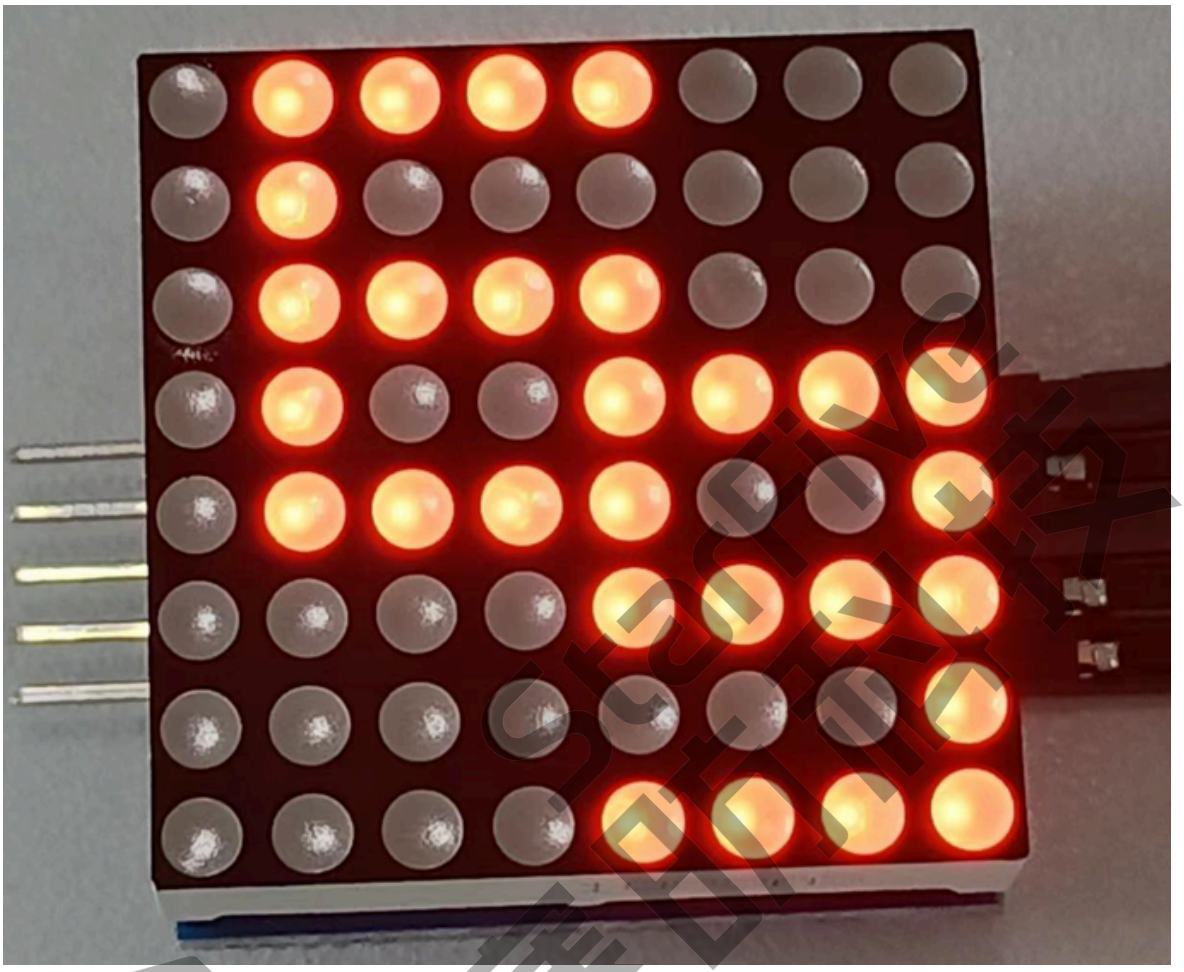
   或者，您也可以执行以下命令：

   ```
   sudo python3 LED_Matrix.py
   ```

   结果：

Led矩阵模块显示出赛昉科技徽标。

图 3-1 示例输出



4. 演示源代码

本演示中的资源代码仅作为参考。

LED_Matrix.py:

```
'''
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             GPIO0
    CS                               38             GPIO2
    CLK                             36              GPIO4
-----------------------------------------
'''

import VisionFive.gpio as GPIO
import sys
import time

DIN = 0
CS  = 2
CLK = 4
#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', '0
00000001', '00001111']

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

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 WriteALLReg():
    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[i], 2))
    time.sleep(5)

    #Display logo.
    for i in range(0, 10):
        for j in range(0, 8):
            #Write data to the register address. Finally turn off the LED
 matrix.
            WriteToReg(i+1, int(buffer_off[i], 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(i+1, int(buffer[i], 2))
        time.sleep(0.1)

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)

def main():
    initData()
    while True:
        WriteALLReg()
```

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