

---

# BPI-Scratch 用户手册

V0.1.0

2015-03-24

---

## Revision History

Revision	Data	Author	Description
0.1.0	2015-03-24	Alessia	Initial version

---

## Table of Contents

### 目录

目录.....	2
1. 安装说明.....	3
1.1 预装软件包.....	3
1.2 运行.....	4
2. Scratch 指令说明.....	4
2.1 GPIOs 通用 IO 接口 .....	4
2.1.1 简介.....	4
2.1.2 示例 1 – GPIO 板 .....	5
2.2 I2C.....	6
2.2.1 简介.....	6
2.2.2 示例 1 – USB Hub .....	7
2.3 SPI.....	8
2.3.1 简介.....	8
2.3.2 示例 1 – LNdigital .....	8
2.4 LN Digital .....	9
2.4.1 简介.....	9
2.4.2 示例 1 - LNIO.....	9

---

# 1. 安装说明

## 1.1 预装软件包

BPI-Scratch是香蕉派的能够与教育软件 Scratch v1.4 版本通信并用来控制硬件 IO 口和扩展外围硬件的处理程序，由 python 编写，提供了相关函数接口控制其他的扩展板读写等操作。

Links@GitHub <https://github.com/BPI-SINOVOIP/BPI-Scratch>

安装

=====

用如下命令安装软件包 python-dev or python3-dev::

```
$ sudo apt-get install python-dev
$ sudo apt-get install python3-dev
```

以及 python-setuptools, python-smbus, i2c-tools 软件包::

```
$ sudo apt-get install python-setuptools python-smbus i2c-tools
```

由于香蕉派内置并默认加载 i2c 驱动, 则还需要 spi 驱动相关软件包。spidev 软件包下载解压到文件夹之后, 安装步骤如下:

Links@GitHub <https://github.com/doceme/py-spidev>:

```
$ cd py-spidev
$ sudo python setup.py install
$ sudo python3 setup.py install
```

安装最新版本 RPi.GPIO 软件包。对于 Banana Pro 和 Banana Pi 分别是 (2 选 1):

```
$ git clone https://github.com/BPI-SINOVOIP/RPi.GPIO_BP -b bananapro
$ git clone https://github.com/BPI-SINOVOIP/RPi.GPIO_BP -b bananapi
```

然后进入相应的目录开始安装, 步骤如下:

```
$ sudo apt-get update
$ cd /RPi.GPIO_BP
$ python setup.py install
$ sudo python setup.py install
$ python3 setup.py install
$ sudo python3 setup.py install
```

## 加载 SPI 驱动模块

最新版本的香蕉派镜像都是默认加载 I2C 模块。但是 SPI 需要手动加载如下::

```
$ sudo modprobe spi-sun7i
```

以上方式每次重启后失效，对于需要使用 SPI 模块通信的硬件扩展很不方便，你可以通过下面两条命令永久修改此文件保证启动时默认加载 SPI 模块：

打开文件 ``/etc/modprobe.d/bpi-blacklist.conf``

注释掉 ``blacklist spi-sun7i`` 此行

```
$ sudo nano /etc/modprobe.d/bpi-blacklist.conf
```

在``/etc/modules``添加两行``spi-sun7i`` 和 ``spidev``

```
$ sudo nano /etc/modules
```

## 配置 Scratch MESH 模式

请参考官方链接的详细步骤 <http://wiki.scratch.mit.edu/wiki/Mesh> => 1.3 Mesh by Modification of Scratch. 注意，修改 Scratch image 需要 sudo 权限，从终端命令行通过如下命令启动 Scratch 才能保存修改，否则无法保存永久修改。

```
$ sudo scratch
```

完成所有上述步骤后重启香蕉派。

## 1.2 运行

LeScratch.py 运行非常简单，python2 和 python3 都支持：首先打开 scratch (保存的 Mesh 模式默认开启)，然后运行 python 脚本，与 scratch 自动建立连接。

```
$ sudo python LeScratch.py
```

或

```
$ sudo python3 LeScratch.py
```

# 2. Scratch 指令说明

## 2.1 GPIOs 通用 IO 接口

### 2.1.1 简介

在 scratch 程序最开始声明哪些 GPIO 口需要被使用：

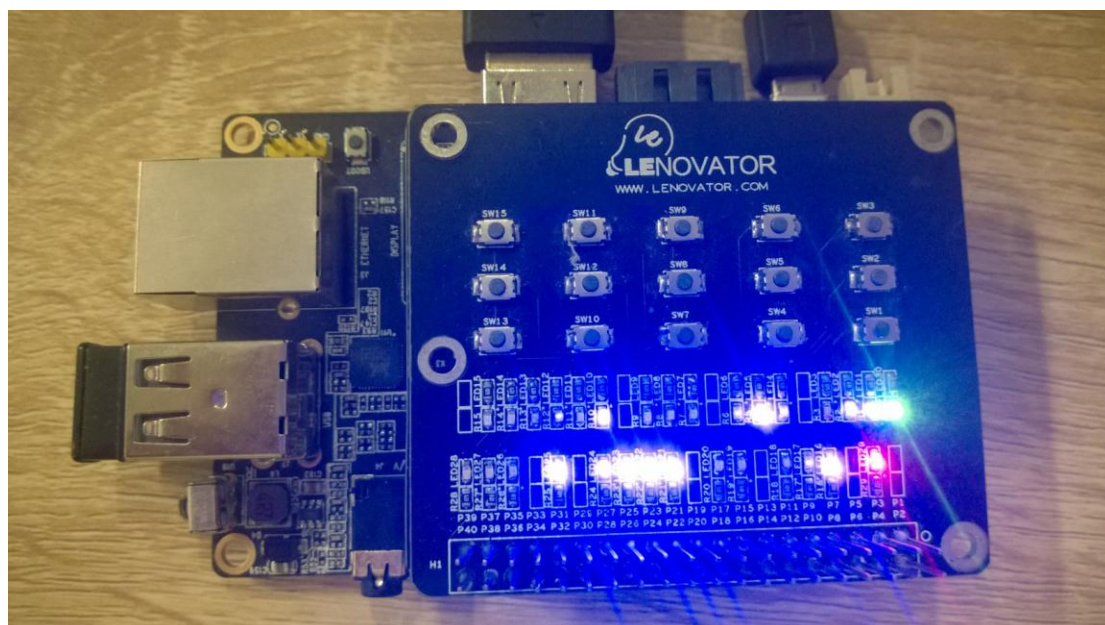
broadcast g[num]in, g[num]out, 其中 “in” 表示作为输入 “out” 表示作为输出  
 输出口通过如下指令设为 1 或 0: 拉高: g[num]on; 拉低: g[num]off

通用 IO 口输入输出控制:

指令格式:	<i>g[num]in</i>	<i>g[num]out</i>	<i>g[num]on</i>	<i>g[num]off</i>
	输入	输出	拉高	拉低

可用的 GPIO 数字列表 [4, 5, 6, 12, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]

## 2.1.2 示例 1 – GPIO 板



购买链接: <http://www.lenovator.com/>

GPIO 板的 40 个 LED 指示灯可以指示香蕉派 40 个排针的高低电平状态，开关对应的 GPIO 口可以用作指定排针输入检测。

当键盘的空格键按下时:  
 声明 GPIO 17 为输入  
 声明 GPIO 4 为输出

While 循环:  
 声明更新, 检测扫描  
 如果输入口 GPIO 17 等于 0  
 将 GPIO 4 拉高  
 等待一秒  
 否则(即输入口 GPIO 17 等于 1)  
 将 GPIO 4 拉低  
 等待一秒

单击绿旗 开始运行:  
声明 GPIO 17 作为输出

While 循环:  
将 GPIO 17 拉高  
等待一秒

将 GPIO 17 拉低  
等待一秒

单击绿旗 开始运行:  
声明 GPIO 12, 13, 19, 16, 6 作为输出

While 循环:  
将 GPIO 12, 13, 19, 16, 6 拉高  
等待一秒

将 GPIO 12, 13, 19, 16, 6 拉低  
等待一秒

## 2.2 I2C

### 2.2.1 简介

I2C 通过选址确定扩展板，在命令行终端输入如下命令即可使用 i2c-tools 检测扩展板地址:

```
$ sudo i2cdetect -y 2
```

Scratch 指令相应的也需要指定 i2c 地址，格式如下:

指令*"i2" + "地址 0x(20-27)" + "a" + "bit (0 to 7)"* : mcp23017 端口 A 寄存器

指令*"i2" + "地址 0x(20-27)" + "b" + "bit (0 to 7)"* : mcp23017 端口 B 寄存器

指令*"bit" + "地址 0x(20-27)" + "a" + "bit (7 to 0)"* mcp23017 端口 A 寄存器

指令*"bit" + "地址 0x(20-27)" + "b" + "bit (7 to 0)"* mcp23017 端口 B 寄存器

示例:

i221a1 => i2c 地址 21 端口 A 寄存器位 1(bit 1) ON

i222b4 => i2c 地址 22 端口 B 寄存器位 4(bit 4) ON

bit22b01010101 => 地址 22 端口 B 寄存器全八位, 输出 => 0b01010101

bit21a01010101 => 地址 21 端口 A 寄存器全八位, 输出 => 0b 01010101

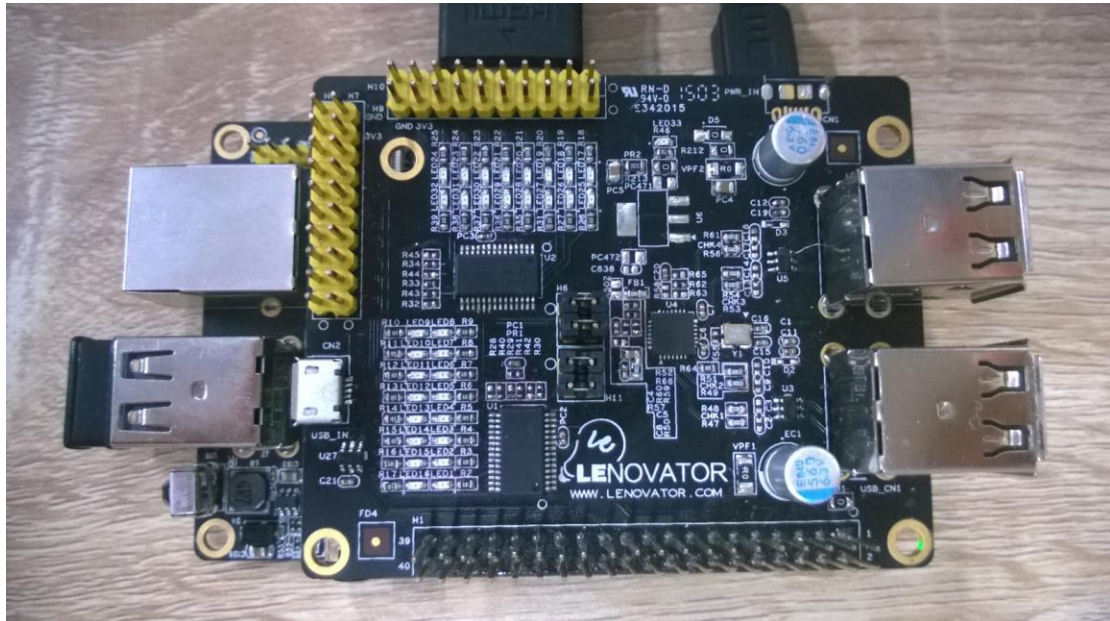
bit21aon => 地址 21 端口 A 寄存器 全八位 ON, 0b11111111

bit21boff => 地址 21 端口 B 寄存器全八位 OFF/clear, 0b11111111

bit22aoff => 地址 22 端口 A 寄存器全八位 OFF/clear

## 2.2.2 示例 1 – USB Hub

USB Hub 扩展板配有 4 个 USB 接口，两块 mcp23017 共 32 GPIOs，用 I2C 接口与香蕉派运行的 Scratch 通信从而扩展了硬件功能。



购买链接: <http://www.lenovator.com/>



当键盘的字母键 a 按下：  
(地址 0x24, 0x26)  
清除 端口 A, B

While 循环


设置地址 0x24 端口 A 寄存器 0b10010111  
设置地址 0x24 端口 B 寄存器 0b01101000

当键盘的字母 b 按下：  
(地址 0x24, 0x26)  
清除 端口 A, B

While 循环

设置地址 0x24 端口 A 寄存器第四位 bit4  
和第七位 bit7 为 1 (其他清零), 等待一秒  
设置地址 0x24 端口 B 寄存器第一位 bit1  
和第六位 bit6 为 1 (其他清零)





当键盘的上箭头按下:  
(地址 0x24, 0x26)  
清除 端口 A, B

当键盘的下箭头按下:  
(地址 0x24, 0x26)  
清除端口 A, B

## 2.3 SPI

### 2.3.1 简介

LN Digital 的 16 位 mcp23s17 通过 SPI 通信与香蕉派扩展, mcp23s17 拥有 8 位地址选择, 可以同时扩展 8 块 LN Digital。

指令"sp"+"地址(0-7)"+"a"+"bit(0 to 7)": mcp23s17 端口 A 寄存器  
 指令"sp"+"地址(0-7)"+"b"+"bit(0 to 7)": mcp23s17 端口 B 寄存器  
 指令"bits"+"地址(0-7)"+"a"+"bit(7 to 0)": mcp23s17 端口 A 寄存器  
 指令"bits"+"地址(0-7)"+"b"+"bit(7 to 0)": mcp23s17 端口 B 寄存器

示例: (地址 0-7 = 0x40-4E)

sp0a1 => spi 地址 0x40 端口 A 寄存器位 1(bit 1) ON

sp1b4 => spi 地址 0x42 端口 B 寄存器位 4(bit 4) ON

bits2b01010101 => 地址 0x44 端口 B 寄存器全八位, output => 01010101

bits3a01010101 => 地址 0x46 端口 A 寄存器全八位, output => 01010101


bits4aon => 地址 0x48 端口 A 寄存器全八位 ON, 0b11111111

bits5boff => 地址 0x4A 端口 B 寄存器全八位 OFF/clear, 0b00000000

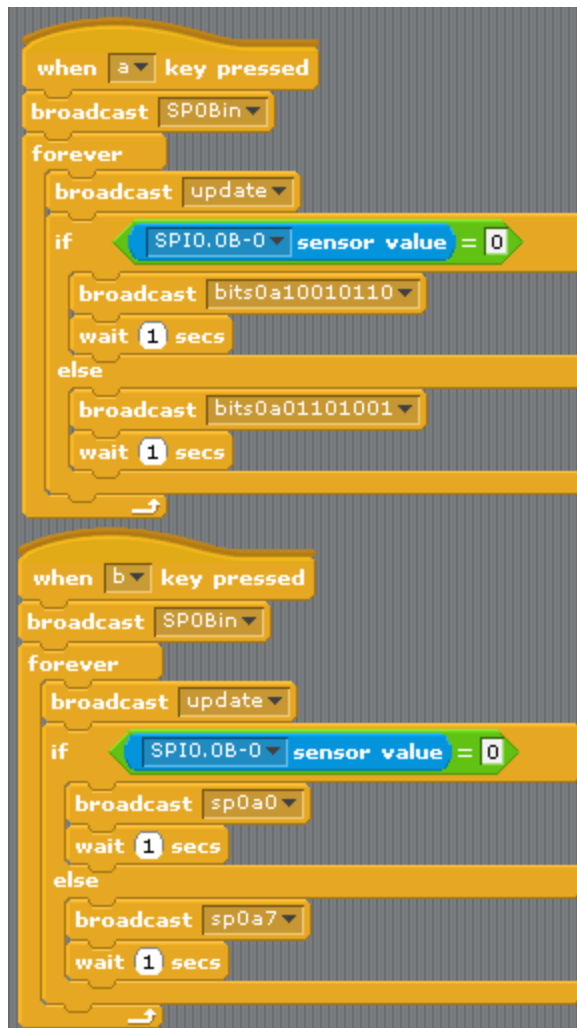
bits6aoff => 地址 0x4B 端口 A 寄存器全八位 OFF/clear

### 2.3.2 示例 1 – LNdigital

LNdigital 能用通用的 SPI 指令格式或下一节内容的 LONIO 专用指令进行控制。



单击绿旗 开始运行  
While 循环  
设置地址 0x40 端口 A 寄存器全八位为 1, 等待 1 秒  
设置地址 0x40 端口 A 寄存器全八位为 0, 等待 1 秒



当键盘的字母键 a 按下：  
 声明地址 0x40 端口 B 寄存器输入  
 While 循环  
 声明更新，检测扫描  
 如果 SPI 地址 0x40 端口 B 寄存器位 0 等于 0  
 设置 SPI 地址 0x40 端口 A 寄存器为 0b10010110，等待一秒  
 否则  
 设置 SPI 地址 0x40 端口 A 寄存器为 0b01101001，等待一秒

当键盘的字母键 b 按下：  
 声明地址 0x40 端口 B 寄存器输入  
 While 循环  
 声明更新，检测扫描  
 如果 SPI 地址 0x40 端口 B 寄存器位 0 等于 0  
 设置地址 0x40 端口 A 寄存器位 0 为 1，等待一秒  
 否则  
 设置地址 0x40 端口 A 寄存器位 7 为 1，等待一秒

## 2.4 LN Digital

### 2.4.1 简介

LN Digital 有一个 16 位 mcp23s17 配置成 8 位端口 A 寄存器和端口 B 寄存器或 16 位。每个端口(A/B) 可以配置为输入或者输出。默认情况为配置端口 A 8 位输出，端口 B 8 位输入，中断事件监听 4 个按键输入。

LNDI[num]in LNDI[num]out LNDI[num]on LNDI[num]off      Number (1 to 8)

输出拉高: LNDI[num]on

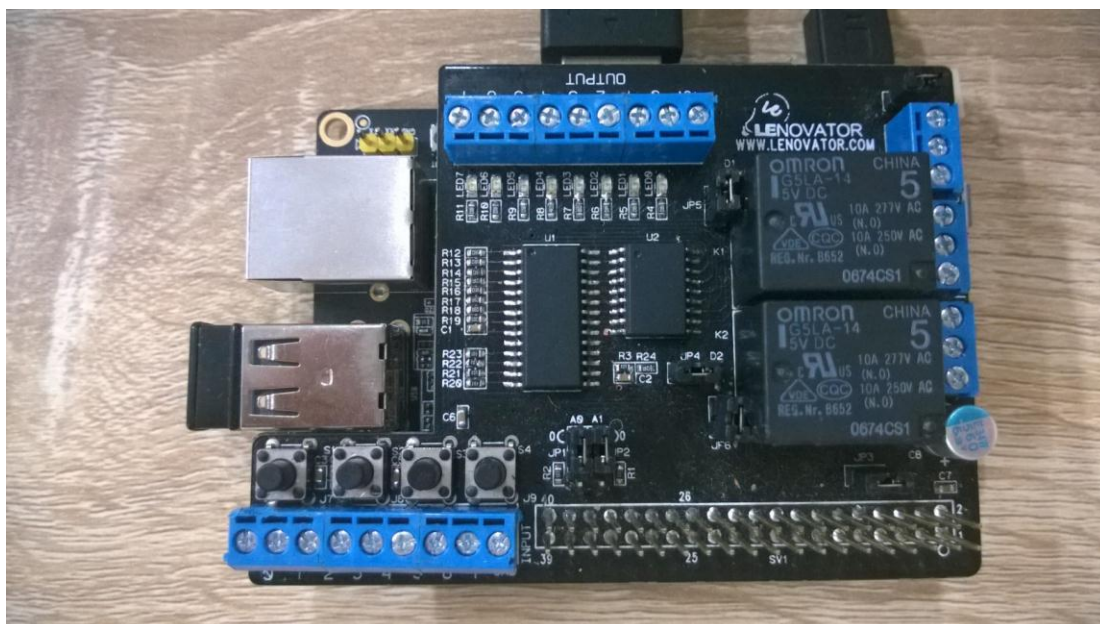
输出拉低: LNDI[num]off

### 2.4.2 示例 1 - LNIO

LN Digital 扩展板与香蕉派使用 SPI 通信，Scratch 功能与之扩展：

8 个输出，8 个 LED 指示灯，8 个逻辑输入，4 个按键

2 个继电器 (端口 A 输出第 1 位- 继电器 1, 端口 A 输出第 2 位-继电器 2.)



购买链接: <http://www.lenovator.com/LN-Digital%28PCBA%29>

<pre> when space key pressed broadcast LNDI0in broadcast LNDI2out LNDI3out forever   broadcast update   if LNDI-in-0 sensor value = 0     broadcast LNDI2on LNDI3off   else     broadcast LNDI2off LNDI3on </pre>	<p>当键盘的空格键按下时：          声明 LNDI1, LNDI2, LNDI3 输出          声明 LNDI1 输入，          While 循环          声明更新，中断监听          如果输入口 LNDI1 按键按下          LNDI1, 3 拉高，LNDI2 拉低          否则 (LNDI1 输入无按键)          LNDI1, 3 拉低，LNDI2 拉高</p>
<pre> when clicked broadcast LNDI1out LNDI2out forever   broadcast LNDI1on LNDI2off   wait 1 secs   broadcast LNDI1off LNDI2on   wait 1 secs </pre>	<p>单击绿旗 开始运行：          声明 LNDI1, LNDI2 输出          While 循环          LNDI1 拉高，LNDI2 拉低          等待一秒          LNDI1 拉低，LNDI2 拉高          等待一秒</p>