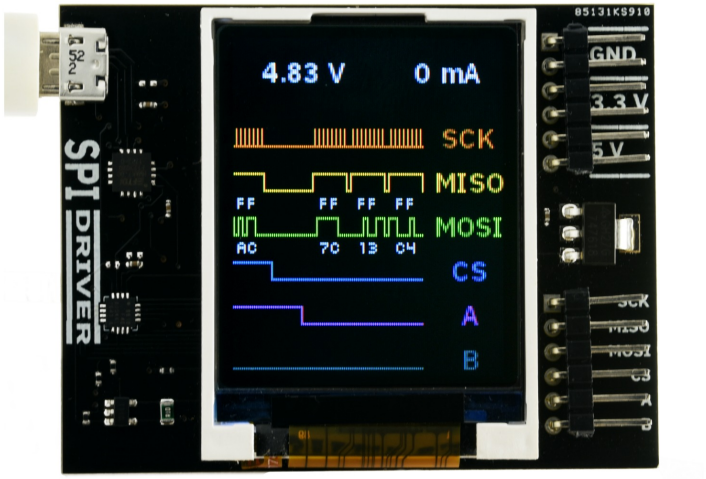


Last updated on 2019 年 8 月 13 日

Contents

1 概论

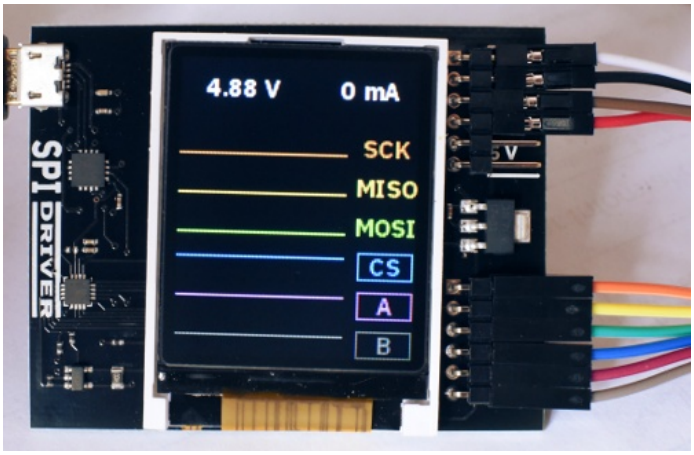
是一个易于使用，开源的工具，通過USB控制SPI设备. 它可以运行于Windows, Mac, and Linux 平台上。可以实时显示所有的SPI活動信息

1.1 特性

- 实时显示: 精确显示
- 高達500 K bps 的传送速率
- USB 电压监控: 侦测USB电压供应问题, 低至0.01v
- 目标设备电源监控: 测量目标设备高侧电流, 精确至5 mA
- 兩個輔助輸出信號：A 和B
- 專門的電源輸出：3.3v和5v 兩組電源和地綫
- 信号线均与调线的颜色相同
- 所有信号都是3.3 伏特，兼容5 伏
- 使用FTDI USB转串口芯片, 和Silicon Labs 汽车级EFM8 控制器
- 报告运行时间，温度和所有流量的CRC
- 所有的传感器和信号都由一套简单的信号协议控制
- 提供了在Windows, Mac, 和Linux 上运行的GUI, 命令行, C/C++, and Python 2/3 软件支持工具

2 开始

当你第一次连接到USB端口，屏幕会短暂闪白色后显示如下：



按照下图所示的顺序连接六个颜色的信号线。

SCK	橙色
MISO	黄色
MOSI	绿色
CS	蓝色
A	紫色
B	灰色

最上面为六条信号线。剩下六条线为地线，3.3V 和5V 各两条。SPIDriver 持续测量USB输入电压和输出电流，并显示在屏幕上方。

3 软件安装

所有的软件源代码在：repository. 包含:

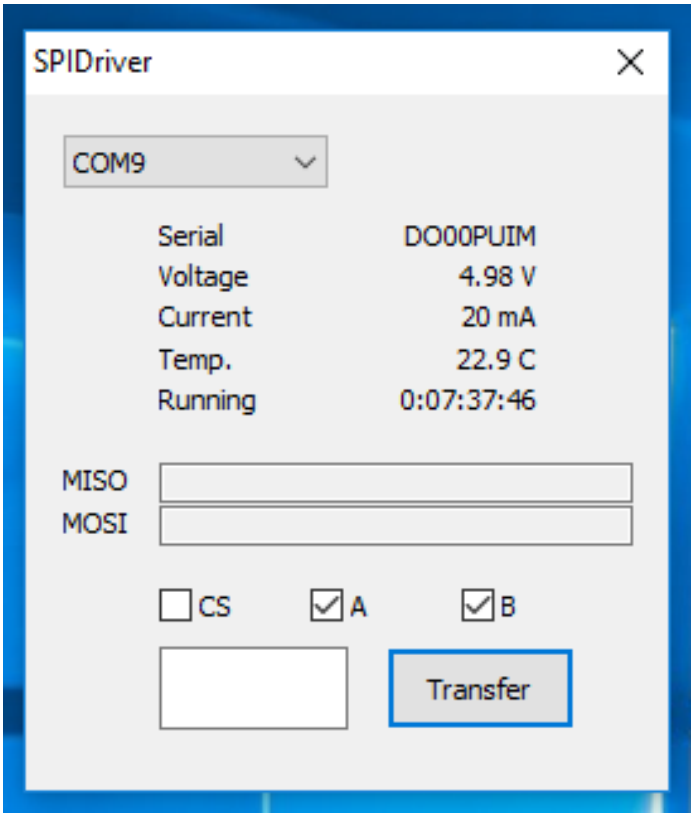
- Windows/Mac/Linux 图形界面工具
- Windows/Mac/Linux 命令行工具
- Python 2/3 绑定
- Windows/Mac/Linux 之上的C/C++ 绑定

图形和命令行工具根据平台不同，安装也不尽相同。

3.1 Windows

这个安装包 包含了图形界面和命令行工具. 以下的图标会出现在桌面上:

启动它就出现下述控制窗口



如果你只有一台串口设备，SPIDriver 会自动被选中。

如果你有多台设备，需要在上面的下拉菜单中选择COM 端口。一旦选中，你就可以控制所有的信号线，并传输十六进制的数据

命令行工具`spicl` 也已经安装. 例如，可以展示如下信息:

```
C:\>"c:\Program Files\Excamera Labs\SPIDriver\spicl.exe" COM3 i
uptime 1625 4.810 V 45 mA 23.3 C
```

关于命令行的语法，请参考下述信息

3.2 Linux

Linux上运行的图形化工具可以在此spigui-linux64 下载. 或者你也可以用如下所示的方法运行Python GUI 要构建命令行工具，克隆repository, 然后:

```
$ cd spidriver/c
$ make -f linux/Makefile
$ ./build/spicl /dev/ttyUSB0 i
```

你会看到:

```
uptime 2285 4.812 V 45 mA 23.6 C
```

3.3 MacOS

MacOS 的图形化工具可以在此spigui-macos 下载. 这是一个Mac的可执行文档，下载后需要做:

```
$ cd Downloads
$ chmod a+x spigui-macos
$ ./spigui-macos
```

或者你也可以用如下所示的方法运行Python GUI.

要构建命令行工具，克隆repository, 然后:

```
cd spidriver/c
make -f linux/Makefile
./build/spicl /dev/cu.usbserial-DO00QS8D i
```

(substituting your actual SPIDriver's ID for `DO00QS8D`) and you should see something like:

```
uptime 2285 4.812 V 45 mA 23.6 C
```

请注意使用的端口在`/dev/cu.usbserial-XXXXXXXX`, 这里here 有详细解释.

4 APIs

4.1 Python 2 和3

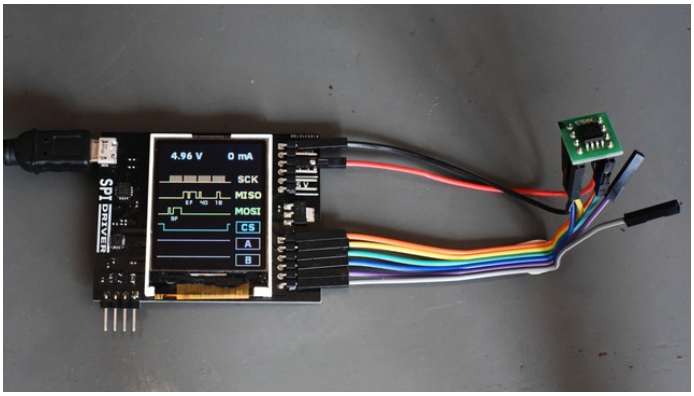
SPIDriver 绑定可以通过pip 安装:

```
pip install spidriver
```

然后从Python环境做:

```
>>> from spidriver import SPIDriver
>>> s = SPIDriver("/dev/ttyUSB0")
>>> s.sel()
>>> s.write([0x9f])
>>> list(s.read(3))
[239, 64, 24]
>>> s.unsel()
>>>
```

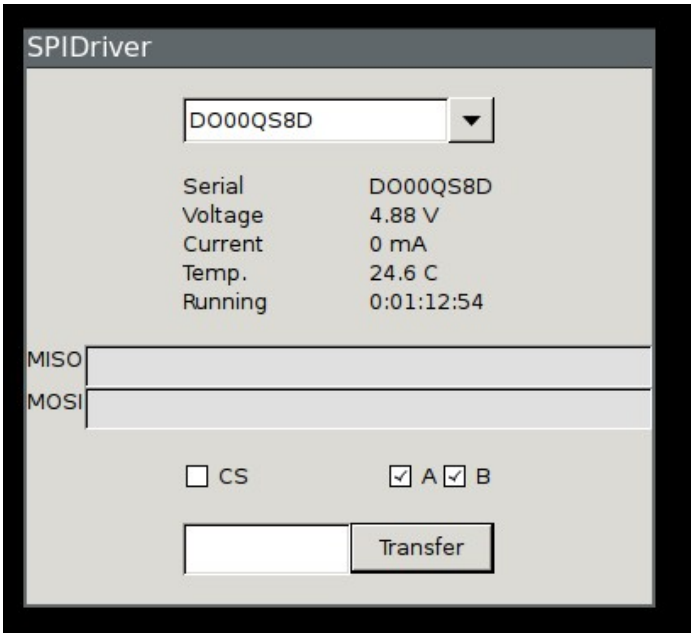
然后你会看到：



使用wxPython 的图形界面工具可以如此运行:

```
python spigui.py
```

根据你所得到的安装包不同，你会得到：



更多例子在此：python/samples .

4.2 C/C++

SPIDriver 包含于一个单一头文件的唯一源文件中。他们在此c/common . 使用下列的Python API 是非常直观的.

5 使用SPI Drive

5.1 The command-line tool `spicl`

`spicl` 在所有平台上都一样.

The first parameter to the command is the serial port, which depends on your operating system. All following parameters are control commands. These are:

第一个参数是串口，这取决于你的操作系统。所有随后的参数都是控制命令，它们是：

`i`	打印状态信息(运行时间, 电压, 电流, 温度)
`s`	SPI 开始
`u`	SPI 结束
`w` *byte*,...	写入字节到SPI
`r N`	从SPI读字节
`a 0/1`	设置A
`b 0/1`	设置B

比如命令:

```
spicl /dev/ttyUSB0 s w 0x9f r 3 u
```

做了以下事情:

`s`	SPI 开始
`w 0x9f`	写字节0x9f
`r 3`	读出三个自己
`u`	SPI 结束

字节默认是十进制，十六进制会以0x开头。

若要发送多个自己，中间请用逗号隔开。

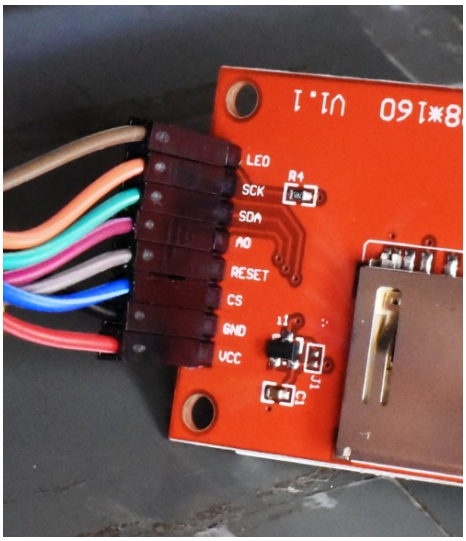
6 例子

6.1 ST7735R 1.8” LCD

ST7735R 是一个小型彩色LCD控制器，其分辨率为160x128. 它有个SPI-Like的控制接口。它并不适用MISO，而且需要额外一根控制线来区别命令字节和数据字节。

它还需外部的重置信号获得稳定的启动。

连接LCD 如下方式:



SPIDriver		LCD
3.3V	棕色	LED
SCK	橙色	SCK
MOSI	绿色	SDA
A	紫色	A0
B	灰色	RESET
CS	蓝色	CS
GND	黑色	GND
3.3V	红色	VCC

这里有一个Python的例子程序，展示在屏幕上显示一张图片。它需要pillow 来做图片的装载，所以你需要安装：

```
pip install pillow
```

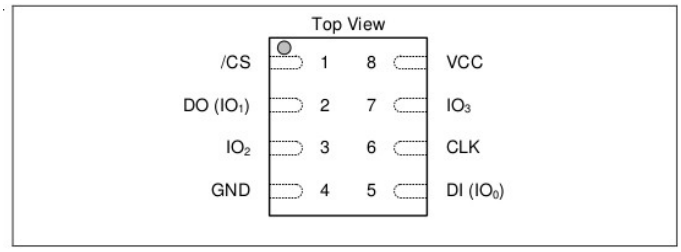
然后在SPIDriver的库中运行:

```
cd python/samples
python st7735s.py -h /dev/ttyUSB0 grace.png
```

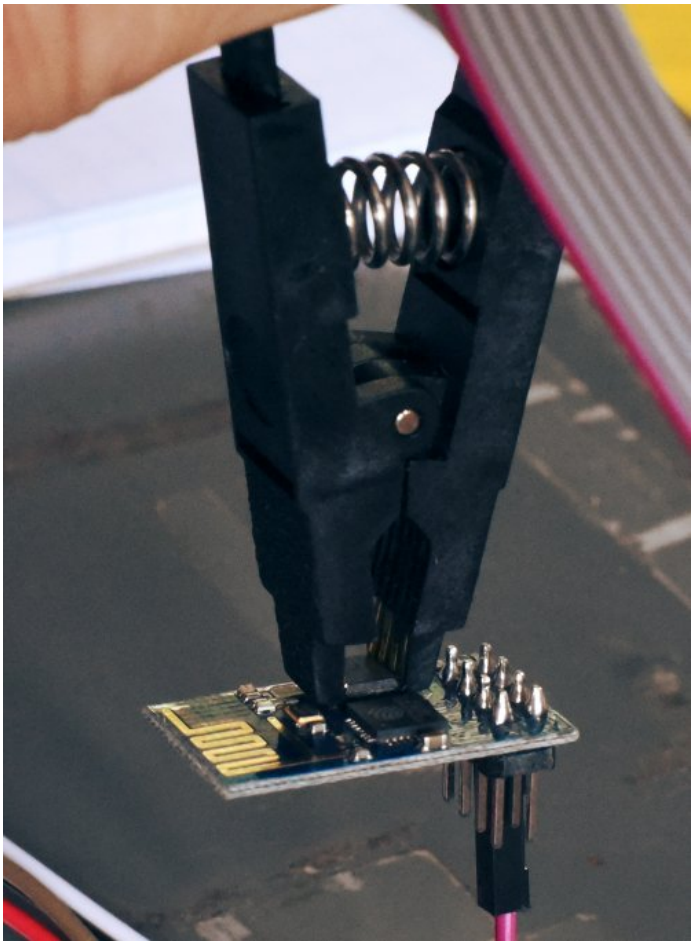
你可以把最后一个参数设置为任何的图形文件，脚本会缩放后显示它。

6.2 SPI Flash

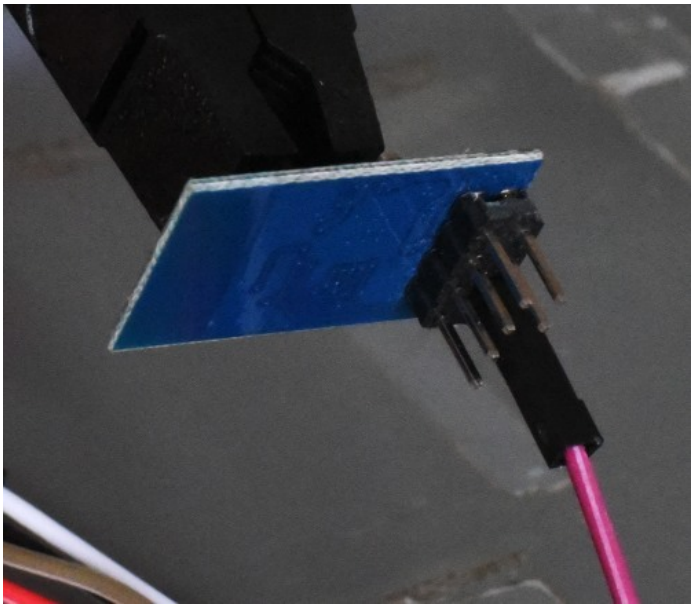
要先在其手册中确定Flash的管脚输出，现代的SPI Flash通常有下列类似的输出：

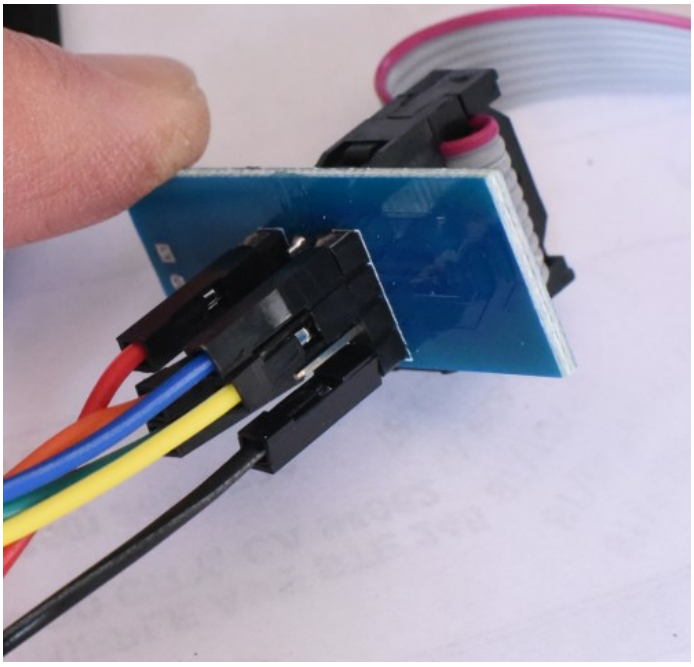


连接夹子到Flash芯片，红色线在管脚1. 下图为夹子连接到ESP8266板子上的Flash 芯片：



对于ESP8266, CPU 需要保持在重置状态，以便保证在SPIDriver 驱动Flash芯片时不会产生竞争。连接SPIDrive 的A信号（紫色）到ESP8266的重置信号：





连接夹子与SPIDriver 如下：

SPIDriver		flash
CS	蓝色	1
MISO	黄色	2
		3
GND	黑色	4
MOSI	绿色	5
SCK	橙色	6
		7
3.3V	红色	8

你可以用命令行工具确认3个字节的JEDEC ID:

```
$ spicl /dev/ttyUSB0 a 0 u s w 0x9f r 3 u
0xc8,0x40,0x13
```

具体的ID 编码因制造商不同而不同。它们通常列在Flash的手册上。第三个字节为Flash的大小（以位为单位），所以上述的Flash是$2^{19} = 524288$ 字节, or 512K 字节. 如果你没有看到有效的ID，你应该重新检查管脚，确保夹子和所有的Flash管脚已经完全连接。这是一个Python的例子flash.py 它用标准的SPI Flash 命令读写Flash的内容. 若是没有其他选项，它会打印出来JEDEC ID作为确认：

```
$ python flash.py -h /dev/ttyUSB0
Got JEDEC ID: c8 40 13
Flash size is 524288 bytes
```

你可以把所有的Flash内容读入一个文件中，使用-r 的选项：

```
$ python flash.py -h /dev/ttyUSB0 -r flashfile
Got JEDEC ID: c8 40 13
Flash size is 524288 bytes
0/512 KBytes
8/512 KBytes
...
504/512 KBytes
```

相似的, -w 选项擦除Flash并写入一个文件：

```
$ python flash.py -h /dev/ttyUSB0 -w flashfile
Got JEDEC ID: c8 40 13
Flash size is 524288 bytes
0/512 KBytes
...
504/512 KBytes
```

许多Flash设备，如同上面的那个，在ID的第三个字节报告它们的大小。然而，有一些设备并不遵守上述规范。为了支持这些设备，脚本的-s 选项可以指定设备的大小，而覆盖默认的计算结果。

7 注意事项

7.1 Port names

SPIdriver 在不同的操作系统会有不同的串口名称。在Windows上，它成为COM1, COM2, COM3 等等. 你可以使用设备管理器或者MODE 命令来显示所有已知的端口这篇文章 描述了如何把一个设备设置为固定的端口.

在Linux平台，它显示为/dev/ttyUSB0, 1, 2 等. 实际的数字取决于设备加入的顺序。然而，它也有可能显示为如下：

```
/dev/serial/by-id/usb-FTDI_FT230X_Basic_UART_DO00QS8D-if00-port0
```

DO00QS8D 是SPIDriver的串行代码(它通常打印在SPIDriver的地步). 它长了一些，但是每次都是同一个名称。

与此相似的是**Mac OS**, SPIDriver 出现为/dev/cu.usbserial-DO00QS8D.

7.2 减少USB的延迟时间

SPIDriver的性能可以通过设置USB的先吃时序到最小值1毫秒得以提高。这可以提高速度最高可达10倍。

在**Linux** 平台设置:

```
setserial /dev/ttyUSB0 low_latency
```

在**Windows** 或者**Mac OS** 平台上，参考文章.

7.3 温度传感器

温度传感器位于EFM8 控制器内。它在出厂时已经校正到精确至2摄氏度。突然的温度上升表明其中的一个管脚(MOSI, SCK, CS, A, or B)与VCC 或地线短路

7.4 原始协议

SPIDriver使用一个串口协议来发送和接受SPI命令。以此参数460800 baud, 8 bits, no parity, 1 stop bit (460800 8N1)连接到SPIDriver

许多SPIDriver commands 是ASCII，你可以从任何支持460800波特率的终端程序控制它。比如输入u 和s 以切换CS 线，键入？显示状态信息。命令是：

?	状态信息(参考下面)
e *byte*	回显*byte*
s	选中
u	取消选中
a *byte*	设置A到0或1
b *byte*	设置B到0或1
x	从SPI Bus断开连接
0x80-bf	读取1-64 字节
0xc0-ff	写入1-64 字节

别入选中和传输2个字节0x12,0x34,取消选中, 主机发送5 个字节. 命令0x81 是两字节读取，两个字节回到PC.

```
s
0x81
0x12
0x34
u
```

状态信息总是80个字符，以空白符填充。例如：

```
[spidriver1 D000QS8D 000007219 4.807 045 25.4 1 1 1 49c1                    ]
```

信息中的位域由空白分割:

spidriver1	固定字符
serial	设备序列号
uptime	运行时间0-999999999, 以秒为单位
voltage	USB总线电压，以伏特为单位
current	设备总线电流, 以mA 毫安为单位
temperature	设备温度，以摄氏度为单位
CS	CS 线状态
A	A 线状态
B	B 线状态
crc	所有输入和输出字节的16-bit CRC (CRC-16-CCITT)

7.5 规范

DC 特性

	最小	典型	最大	单位
电压精度		0.01		V
电流精度		5		mA
温度精度		± 2		°C
MISO				
低压			0.6	V
高压	2.7		5.8	V
输出信号电流(SCK, MOSI, CS, A, B)			8	mA
输出电流			470	mA
电流消耗		25		mA

AC 特性

	最小	典型	最大	单位
SPI 速度	495	500	505	Kbps
运行时间精度		150		ppm
运行时间回滚		31.7		years
启动时间			200	ms

8 支援信息

技术和产品支援可以在此support@spidriver.com找到

SPIDriver 由Excamera Labs 创建和维护.