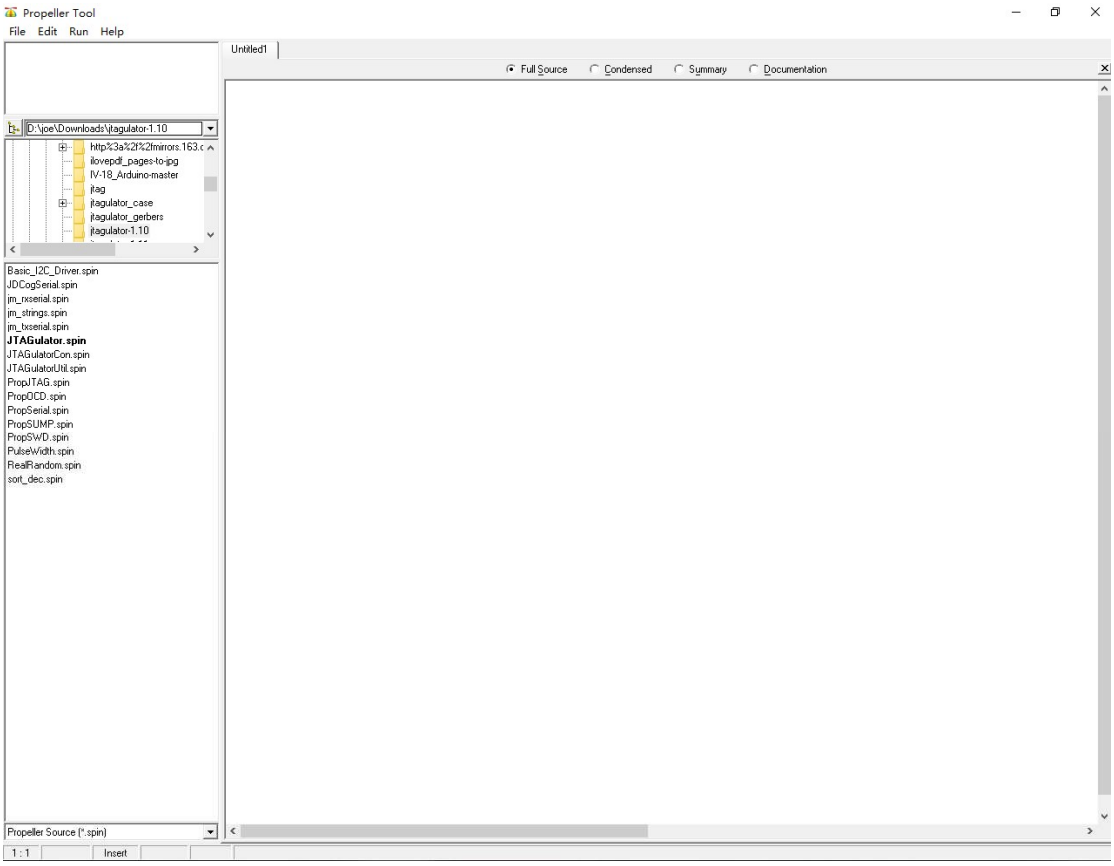
Jtagulator 使用手册

0x00 前言

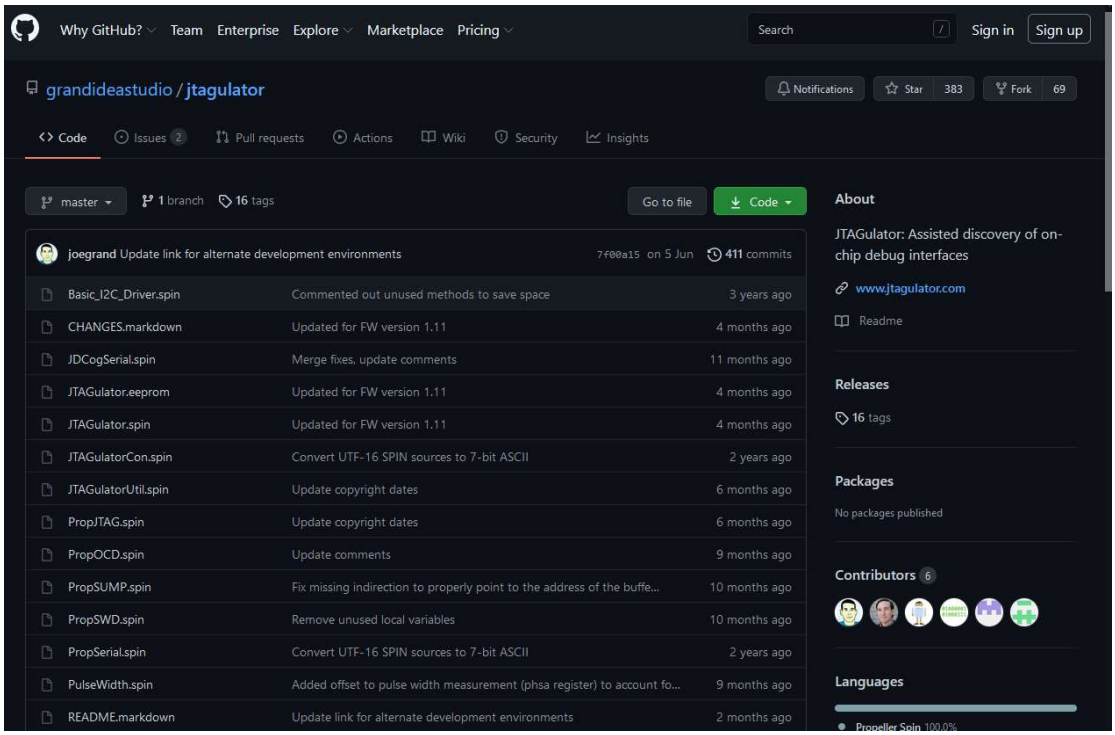
Jtagulator 是由 Joe Grand 开源的硬件调试器，主要有自动识别接口，逻辑分析仪，烧录器三大功能，很适合对各类硬件设备进行逆向，是一款很棒的硬件安全设备。该设备为完全开源的设备，其官网在 http://www.grandideastudio.com/jtagulator/ ，作者在 YouTube 上面也有自己的频道欢迎去关注下：https://www.youtube.com/channel/UCqGONXW1ORgz5Y4qK-0JdkQ 本人也在 bili 上面更新了两期视频，关于硬件原理的解析：https://www.bilibili.com/video/BV1qB4y1T7Ba 使用的教程：https://www.bilibili.com/video/BV1V54y1E7rp ，欢迎各位围观。

0x01 更新固件

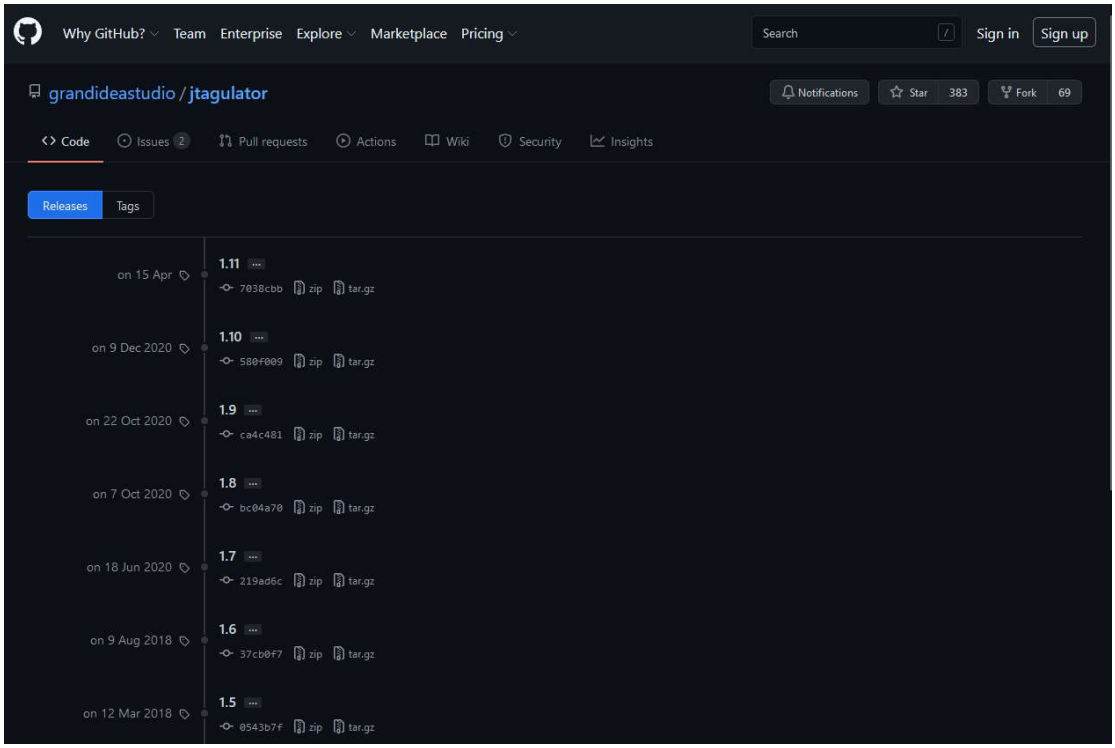
先讲讲这玩意怎么刷固件，www.grandideastudio.com/wp-content/uploads/P8X32A-Setup-Propeller-Tool-v1.3.2.zip 点击这个连接，安装其中的软件，打开之后就是这个样子



在这里 https://github.com/grandideastudio/jtagulator 下载其中的固件

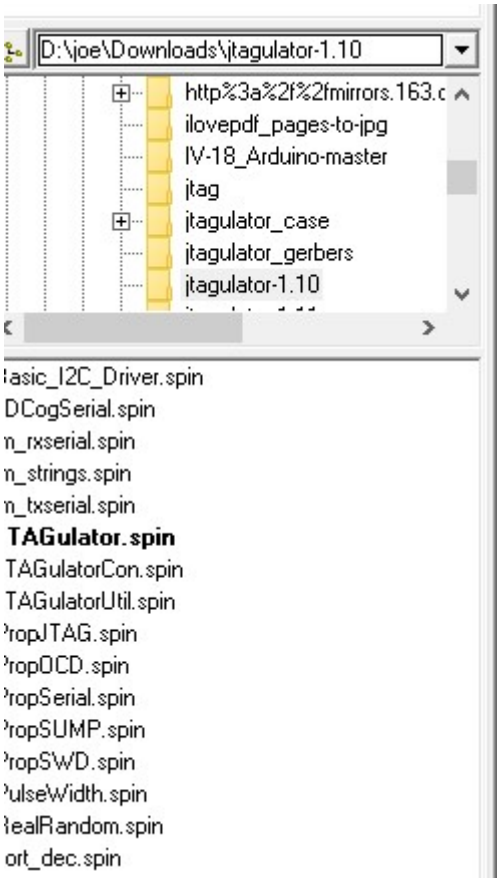


本教程面向完全的小白，所以我说明下，请不要在该界面瞎点，认真观察，右侧栏里面有一个 Release，点击它，就会进入到这个界面



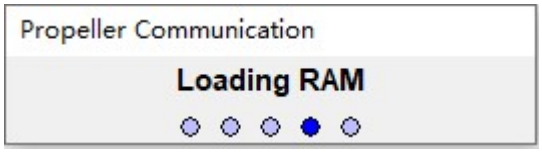
这才是不同版本固件的下载地址，个人推荐下载 1.11 和 1.10 备用，因为 1.11 估计是作者在疫情期间无聊更新出来的，有最新的的功能，也有罗分模式无法连接的 bug，所以，推荐各位在 1.11 和 1.10 之间随意切换。

说下烧录软件的使用，在你把上面的特定版本的 Release 下载解压之后，在烧录软件中点击菜单栏的 File，之后选择 open，之后选择你解压的文件夹中的 JTAGulator.spin 文件，这样你的左侧栏就有如下显示了



这时候还是不能烧录的，需要选择顶层文件，也就是相当于在一堆代码文件中指定一个 main 文件，右键上图中的 JTAGulator.spin 文件，选择 Top Object File，就会发现 JTAGulator.spin 这个文件名加粗了，现在整个项目就可以编译并烧录到你的 Jtagulator 里面了。

这时候按下 F7 键，该软件就会寻找电脑的串口上是否有对应的设备，如果显示找到了有一个设备的话，就可以按下 Ctrl+F11，烧录就会开始。



为啥要在这里讲烧录呢，因为最新版固件有 bug，但是又有新功能，我们后面说。不过我这里也说下另外一个事情，如果你的 Jtagulator 连上上位机，但是完全不反馈任何信息，这时候你的机器并没有任何问题，你重启和重新上电以及烧新的固件都没有用，你需要的是按下 Ctrl+x 组合键，从你上次进入的 openocd 模式或者逻辑分析仪模式退回到 Jtagulator

模式，这样就可以继续操作了。

0x02

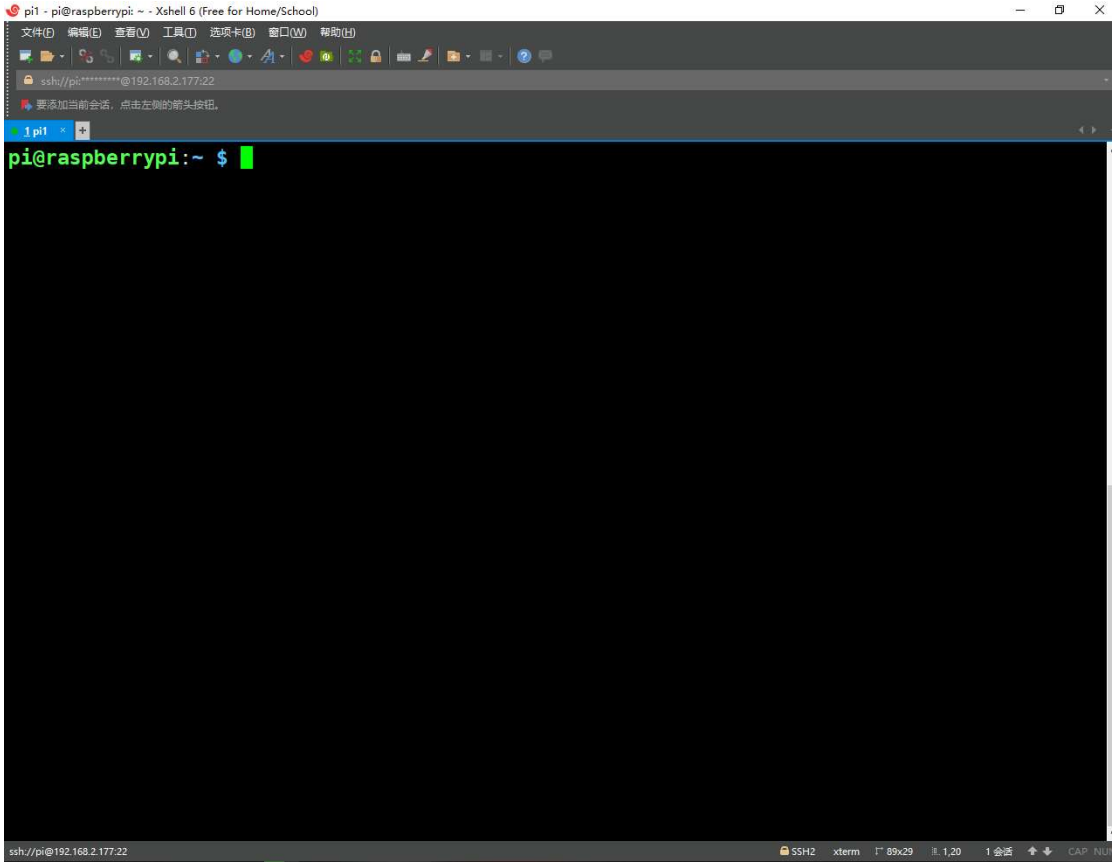
Jtagulator 共有 24 个通道，一个 GND，和一个 VADJ 输出，具体到接口上，就是三组 2.54 排针和一排快速连接器，用户根据自己的使用习惯自行选择连接口即可。

至于说 Jtagulator 和电脑的连接则是通过 USB 数据线，Jtagulator 为 mini USB 接口，连接到电脑之后会自动识别为串口设备，基本无需驱动，如果你的 Windows 电脑并不能识别设备，请在设备管理器中选择该设备，自动更新该设备的驱动，如果你使用的是 Linux 电脑，请运行 lsusb 指令，如果有以下输出识别到了该设备。

```
Bus 001 Device 006: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-S
erial (UART) IC
```

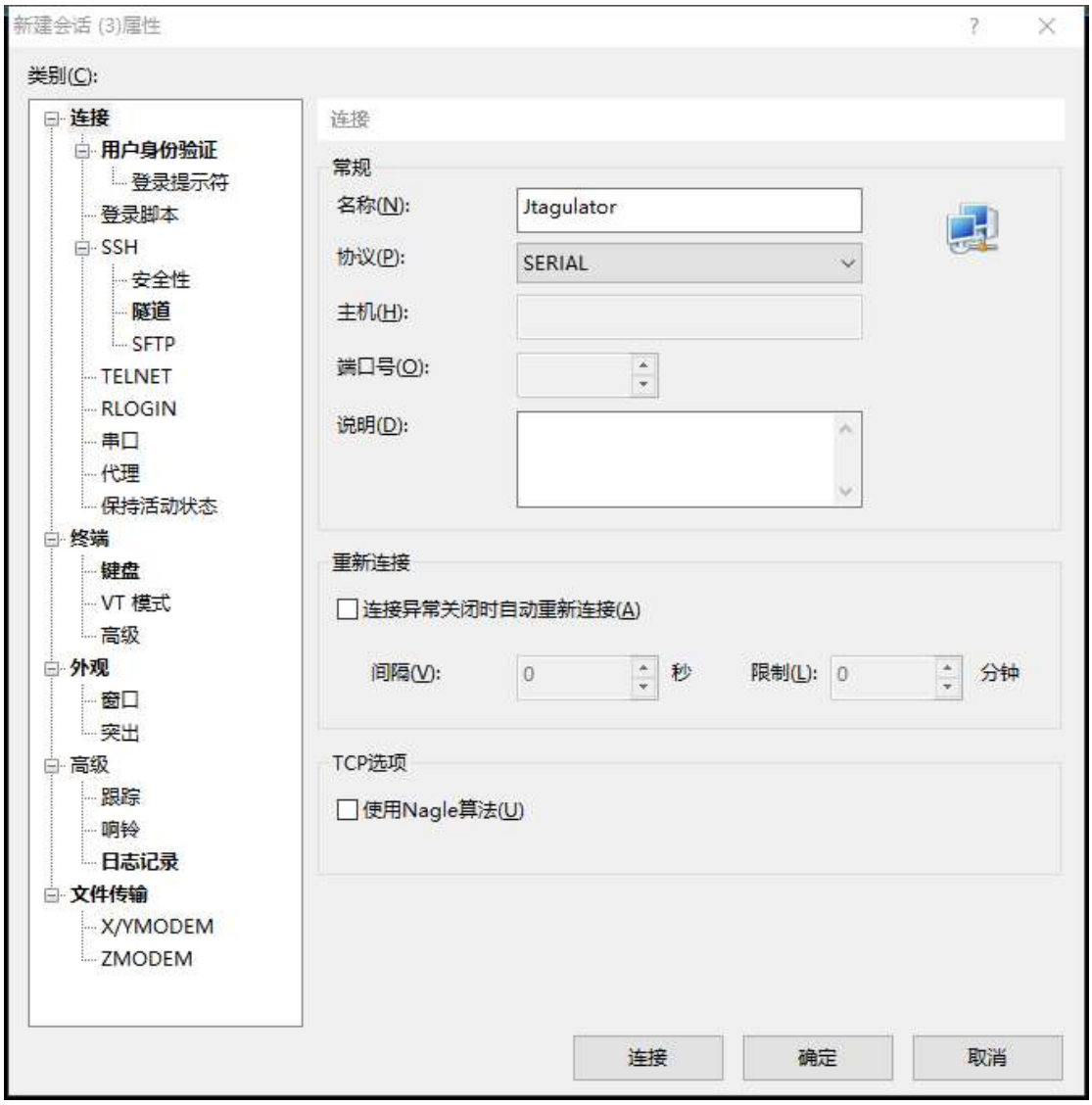
本人仅仅有 Windows 和 Linux 电脑也仅仅推荐这两个系统上的串口软件，至于 Mac 用户，请自行寻找串口软件。

在 Windows 上面，本人推荐的是 Xshell 这款软件，既可以连接 ssh，也可以连接串口，界面友好，下载需要去官网，国内下载站基本带植入。Xshell 属于是免费软件，需要用户注册之后下载，详情请见官网：https://www.netsarang.com/zh/xshell/ 。软件界面见下图

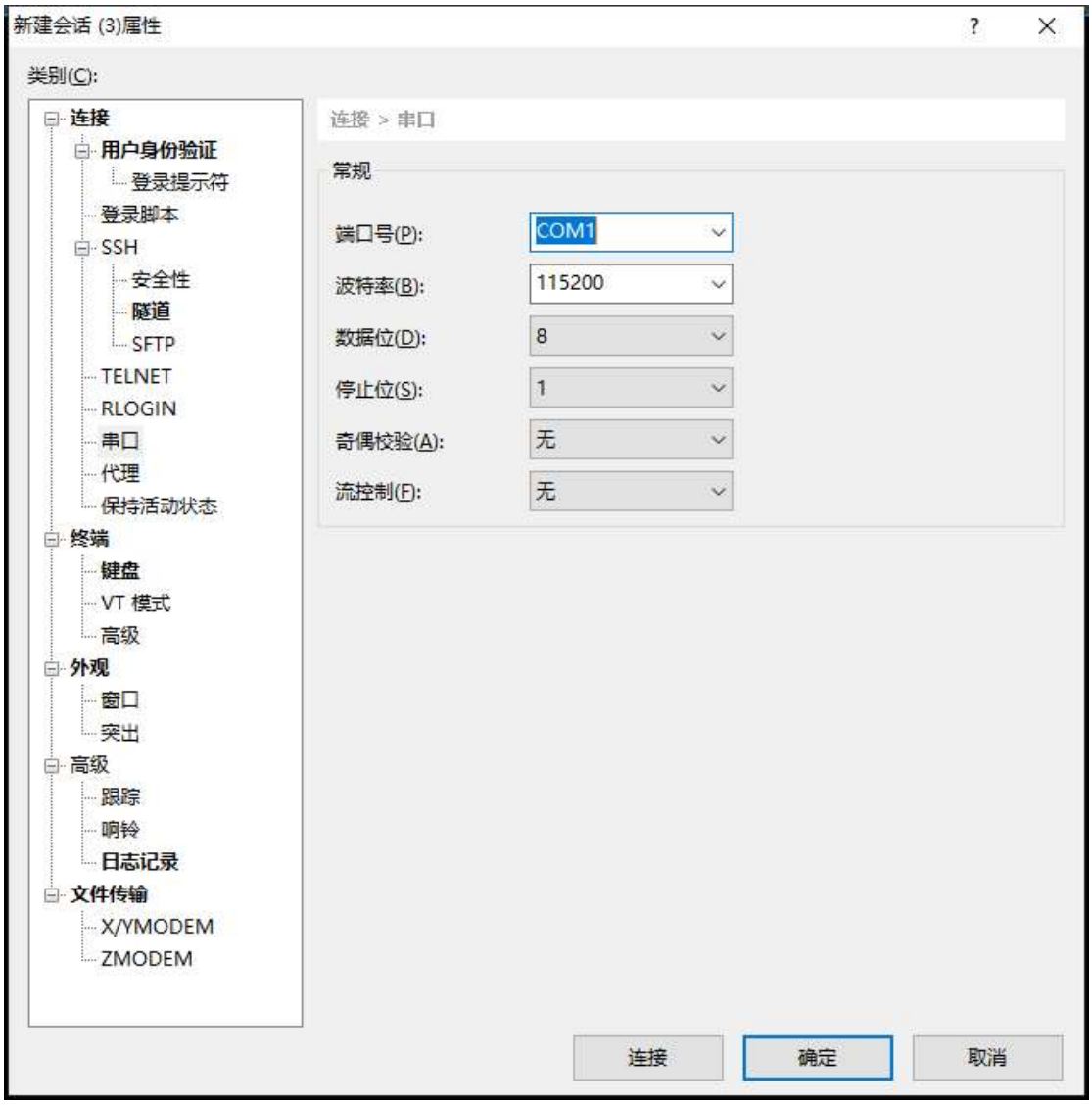


软件打开之后点击菜单栏中的新建选项。

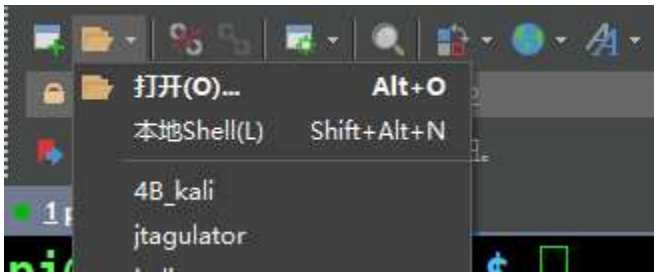
会弹出新建窗口，此处名称改为 Jtagulator，协议改为 Serial。



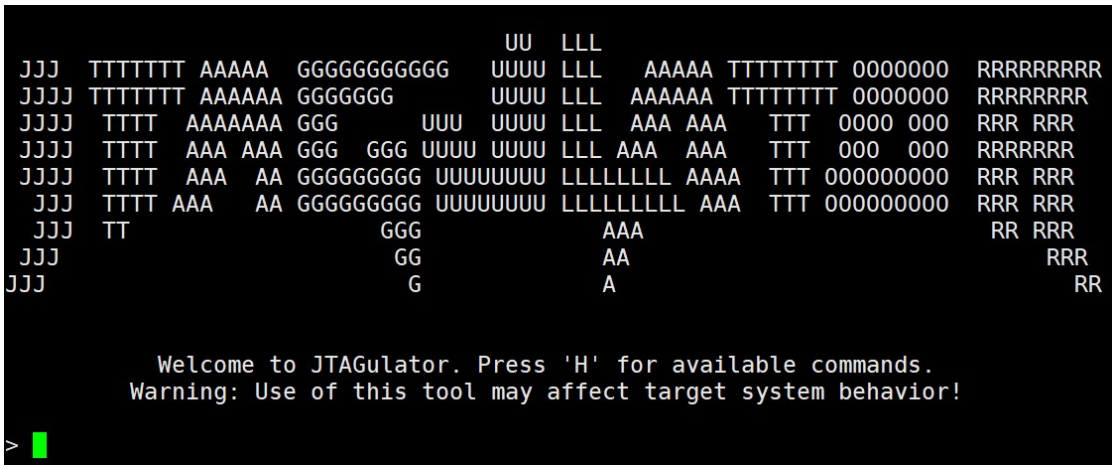
之后点击该窗口的左侧中的串口选项卡，将其中的串口号改为你的 Jtagulator 的串口号。波特率改为 115200，此外的设置不需要修改，点击下方的确定即可，这时候连接的设置就已经保存了。



回到主页，在刚刚新建的图标右侧有一个打开按钮，在其中选择你刚刚新建的 Jtagulator，如果你刚刚串口号之类没有设置错误，就可以正常使用了。



如果你前面没有任何问题，就会有一个字符画的回显，这样在 Windows 下的连接就已经讲完了。



在 Linux 下的串口软件有好多种，比较出名的有 minicom，picocom，这两个软件都很好用，我个人喜欢 picocom，相较于 minicom 更为简单易用，也不会像 minicom 一样出现在嵌入式平台上的输出串行现象。

在 Debian 系的主机上使用

sudo apt install picocom

来安装该软件，如果你使用红帽系的 Linux 主机那指令就是

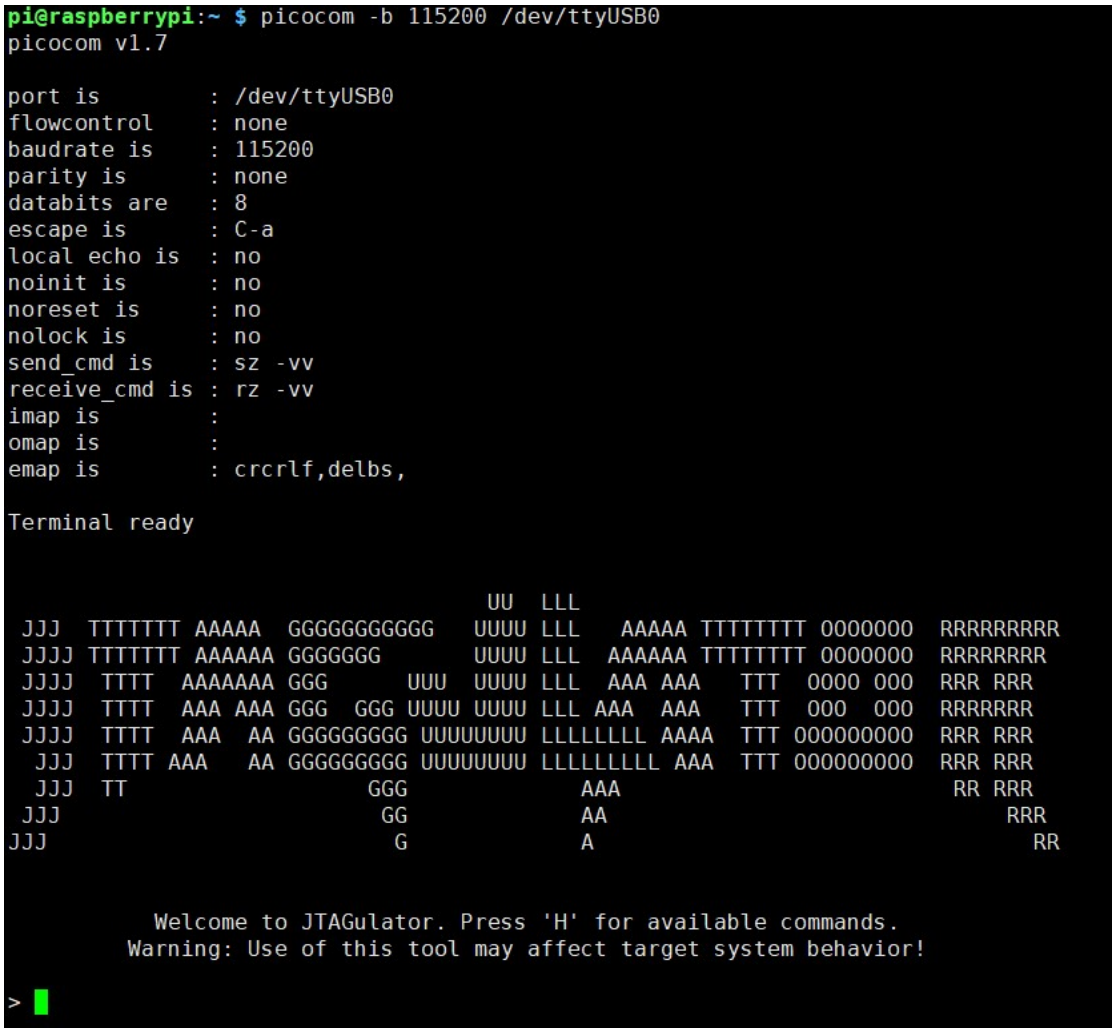
yum install picocom

在这里小声逼逼两句，我个人是用 Windows 上的 xshell 通过 ssh 连接我的树莓派，进而控制插在树莓派上的 Jtagulator，我也推荐各位试试这个方式，Linux 下各种软件的配置太简单了，Linux 永远的神。之后的部分文档是基于 Windows，绝大部分是基于 Linux 来的，而且 Windows 下面能做到的，Linux 都可以做到，还可以极大的简化找软件的流程。

当你的 Linux 主机安装好了 picocom 之后，把 Jtagulator 连接到你的主机上，在命令行中输入

picocom -b 115200 /dev/ttyUSB0

之后就会有以下显示，如果字符画界面没有出现，记得敲下回车。



前半部分是 picocom 对这个连接的描述，从字符画开始，就是 Jtagulator 的回显了。至此连接 Jtagulator 的部分就讲完了，总结下来，就是一个 115200 的串口，只要能连上，无论你用什么软件都可以拿来用。

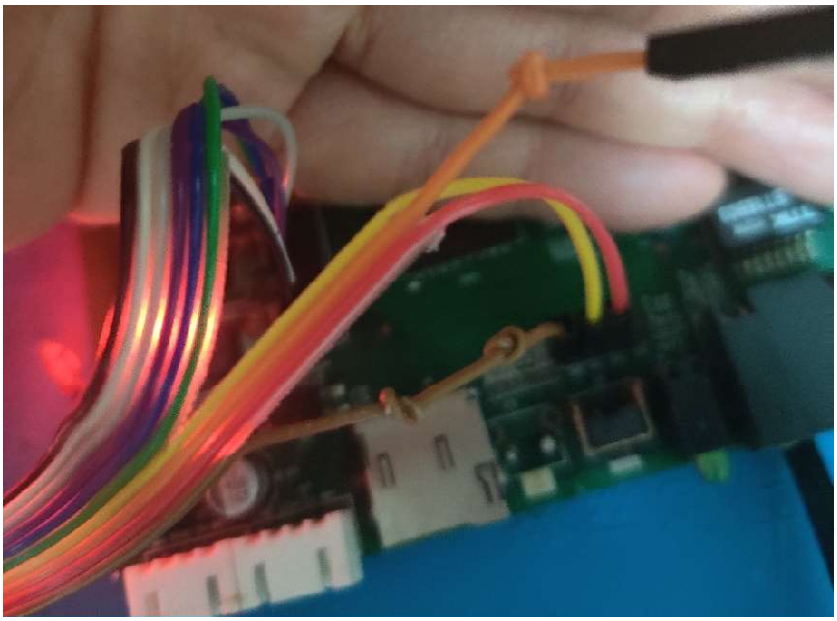
0x03Jtagulator 如何连接目标板

既然要做硬件逆向，我们就要假定我们完全不知道你要测试的板子的针脚定义，这时候你要做的事情就是拿着万用表的通断档或者叫二极管档直接测量你的目标板，哪一个针脚和 GND 是直连的（一般都有 GND 针脚，如果没有，推荐从电源接口飞线，或者从电容负极飞线），之后给目标板上电，这时候我还是不把 Jtagulator 和目标设备连接起来，给 Jtagulator 和目标板上电，我们现在要做的是测量目标板的 GND 和 Jtagulator 的 GND 引脚之间测量电势差，如果在电势差相差 0 到 0.1V 基本就可以连接了。为啥要测量这些，电路调试很重要的一点就是共地，如果你将目标板得 5v 连到了 Jtagulator 得 GND 上，相当于电源短路得同时，通信电平也变成了负值，这样不仅无法嗅探和通信，还会烧板。

当然了，通信电平用万用表是测量不出来的，万用表很多用的是积分式，换言之通信中高低电平占比五五开，那你测到了 2.5V，但是通信电平应该是 5V 才对啊，用示波器才是最好的。

关于通信电平的问题，我建议是示波器看下，如果你前面的 GND 找对了，其实不用太在乎这个问题，fpga 通信电平 1.8V，3.3V 的很多，至于其他板子，5V 也有很多，直接连没啥问题，Jtagulator 用的是隔离电阻加 TVS 的组合，可以有效地防护电涌这类可能的伤害，没问题，大胆的用就好。

另外说下，我做的这一批都带了排线，左侧排针有三组，每一组中有 GND 和 VADJ，我个人直接在排线上面把 gnd 打了两个节，VADJ 打了一个节，这样我就可以最快的找到 GND 和 VADJ 接口。补充说明下，VADJ 并不是电源输出，仅仅是一个指示通信电平的作用，不要连在其他设备上供电，听我一句劝，少烧一块板。



0x04Jtagulator 基本使用

让我们先输入一个 h 看看帮助，有几个通用指令，几个目标交互指令，一个个说。

```
> h
Target Interfaces:
J   JTAG
U   UART
G   GPIO
S   SWD

General Commands:
V   Set target I/O voltage
I   Display version information
H   Display available commands

>
```

通用指令:

V：设置 I/O 通讯电平，一般来说要先测量你要研究的板子的 IO 通信电平，之后设置的。输入 V（大小写无所谓）回车，会出现如下提示

```
> V
Current target I/O voltage: 1.4
Enter new target I/O voltage (1.4 - 3.3, 0 for off):
```

这里输入 1.4 到 3.3 之间任意数字回车都可以设置电压。这个要按照你的目标设备的通信电平而决定。

I：查看设备的软件版本，输一下试试

```
> i
JTAGulator FW 1.10
Designed by Joe Grand, Grand Idea Studio, Inc.
Main: jtagulator.com
Source: github.com/grandideastudio/jtagulator
Support: www.parallax.com/support
```

1.10 版本的固件，嗯，没别的用处了。

H：我们不是刚刚就输过了么？

```
> h
Target Interfaces:
J   JTAG
U   UART
G   GPIO
S   SWD

General Commands:
V   Set target I/O voltage
I   Display version information
H   Display available commands

>
```

好了，剩下几个就是通往几个交互方式的通道了，J 就是 jtag 交互嗅探，U 是串口交互嗅探，G 则是通用输入输出嗅探，S 为 SWD 接口的嗅探。

0x05GPIO 嗅探

我们从最简单的开始，也就是从 G 开始，输入 G 回车，之后输入 H，再回车，就可以看到 G 的帮助文档

```
GPIO> h
GPIO Commands:
R   Read all channels (input, one shot)
C   Read all channels (input, continuous)
W   Write all channels (output)
L   Logic analyzer (OLS/SUMP)

General Commands:
V   Set target I/O voltage
H   Display available commands
M   Return to main menu
```

里面的通用指令还是刚才的样子，专用指令有好几个，我们看看。

R，这个就是读取一次目前各个通道上的电平值，高为 1，低为 0，这里额外说明只要前面的参考电压设置指令 V 设置了，你的每一个通道都会上拉，换言之，只要是没连任何设备，就是高电平，连到目标设备也是把对方的 io 给上拉了，这一点可能导致一些特殊的 bug，你们注意。在运行 R 之前，我将 0 通道连在了 GND，让我们运行下这个指令。

```
GPIO> r
CH23..CH0: 11111111 11111111 11111110  (0xFFFFFE)
```

嗯，不仅仅读取了 IO 的电平还给转化成为了 16 进制数。我前面说的上拉问题也体现出来了。

C，持续读取，一旦有任何变动，都会发生变化，并反馈在屏幕上，按下任意键都会退出该模式

```
GPIO> c
Reading all channels! Press any key to abort...

CH23..CH0: 11111111 11111111 11111110  (0xFFFFFE)
CH23..CH0: 11111111 11111111 11111111  (0xFFFFFF)
CH23..CH0: 11111111 11111111 11111101  (0xFFFFFD)
CH23..CH0: 11111111 11111111 11111111  (0xFFFFFF)
CH23..CH0: 11111111 11111111 11111101  (0xFFFFFD)
CH23..CH0: 11111111 11111111 11111111  (0xFFFFFF)
CH23..CH0: 11111111 11111111 11110111  (0xFFFFF7)
CH23..CH0: 11111111 11111111 11111111  (0xFFFFFF)
CH23..CH0: 11111111 11111111 10111111  (0xFFFFBF)
CH23..CH0: 11111111 11111111 11111111  (0xFFFFFF)
CH23..CH0: 11111111 11111111 10111111  (0xFFFFBF)
```

W，这个指令是输出，输入 w 回车，就有进入输出预备模式，按照提示需要将文件以 16 进制的形式输入进去，回车就会输出出去。

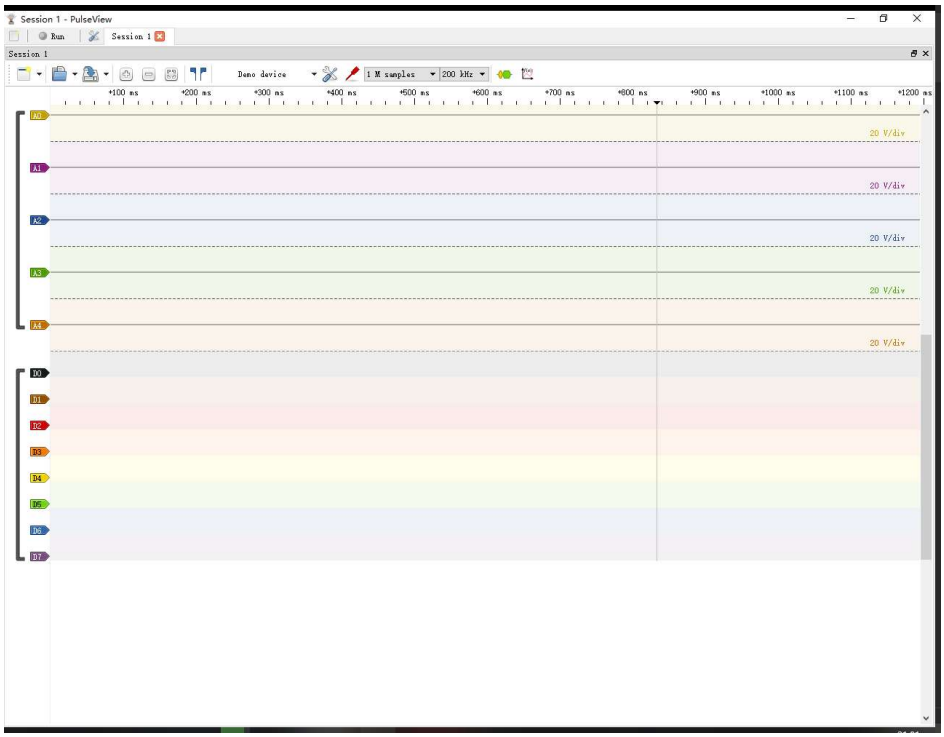
```
GPIO> w
Enter value to output (in hex) [FFFFFF]:
```

我们在这也探讨下这个情况下的 16 进制和 2 进制的转换，比如 11111111 11111111 11111111 就是一个 24 位的二进制数，转换成为 16 进制，就是 FFFFFF。举两个例子 11111111 11111111 11111111，从右到左依次为 24 个通道，转化为 16 进制之后为 FFFFFF，如果你想在 2 通道输出 0，其他通道为 1，转化为二进制也就是 11111111 11111111 11111101，转化为十六进制数也就是 FFFFFD。如果你不熟悉 2 进制和 16 进制，推荐百度下，讲的肯定比我强。嗯把我们计算的结果输入进去，你看后续的输出就显示了输出，CH2 通道为 0，其他通道为 1。再敲下回车，就又回到 gpio 模式下了。

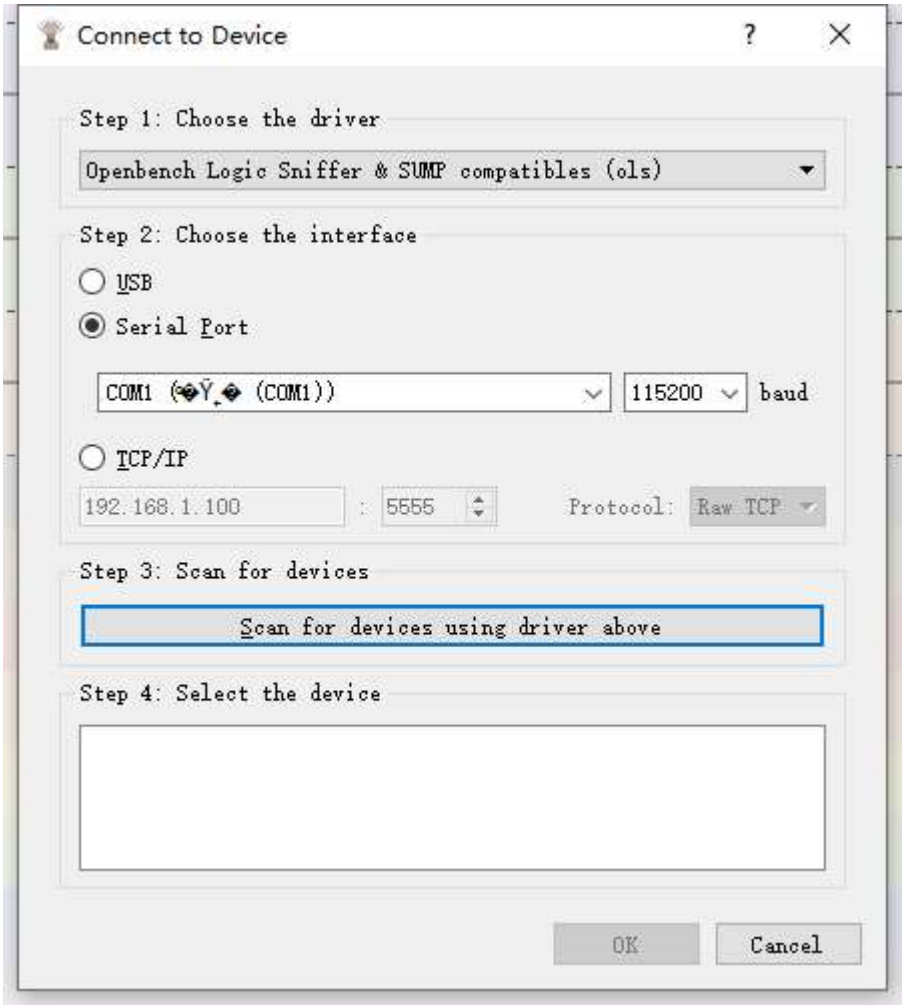
```
GPIO> w
Enter value to output (in hex) [FFFFFF]: FFFFFD
CH23..CH0: 11111111 11111111 11111101  (0xFFFFFD)
Press any key when done...
```

L，这个指令一旦回车就进入逻辑分析仪模式了，本人实测，最新版本的 1.11 固件进入逻辑分析仪模式，并不能被 Windows 上的 PluseView 并不能识别到。有人在 GitHub 上面提了 issue，但是作者老哥回复是，嗯，按照官方 wiki 来，但是我和提出这个 issue 的人一样，绝对是按照官方 wiki 来的。

我们在这里说下官方是如何说的，下图为 pluseview 的主界面，是一个很不错的逻辑分析仪软件，关于这个软件的使用很是简单，和传统逻辑分析仪一样，建议去这其主页上了解更多信息：https://sigrok.org/wiki/PulseView ，可以在官网上下载到其 Windows 的安装包，还有 linux 的安装指南，个人不是很建议 Windows 上面装这个，官方仅仅给出了 Nightly 版本，而 linux 才是该软件的主场。



跑偏了，我们继续聊逻辑分析仪的问题，先点击 Demo Device，之后选择 Openbench Logic Sniffer & SUMP compatible (ols)，之后选择 jtagulator 的串口号，并设置为 115200 的波特率，在下方点击 Scan for devices using the driver above，如果识别到了就会有一个叫 JTAGulator with 24 channels 的设备，之后点 ok 就可以了。



我在 Windows10 下面测试，1.11 版本的固件下，是无法识别的，kali 下面也无法识别到，1.10 在 Windows 下面可以识别到，但也是少数几次。我刷入了之前的固件，之前的版本中并没有逻辑分析仪功能，我认为这个功能仍然不稳定，建议不要耗费太多的精力在里面。想要退出逻辑分析仪功能，需要在串口连接的界面上，输入 Ctrl+X 组合键，即可退出到 GPIO 模式。

官方对于这情况也有准备，贴了张表格，表示了下当前逻辑分析仪的问题，个人认为这个功能刚刚加入并不算是很稳定，我建议各位等待软件迭代一两个版本，就会更为稳定，到时候这个功能会更为好用。

OS	Tool	Issue	Solution
All	PulseView	Upon launching, PulseView will automatically scan for any supported devices. Occasionally, this can cause PulseView to incorrectly choose the generic FTDI FT232R USB UART driver.	Disconnect and reconnect the JTAGulator, then "Connect to Device" as normal. Or, if your OS supports command-line arguments, launch PulseView with the `-D` option to disable automatic scanning.
All	sigrok-cli	The OLS driver does not have support for aborting an acquisition while waiting for a trigger event. If sigrok-cli is stopped, the JTAGulator will remain in its acquisition state.	The next time sigrok-cli is run, the JTAGulator will automatically enter the correct state.
macOS	PulseView	When selecting the JTAGulator within the "Connect to Device" window, the JTAGulator will reset.	Before starting an acquisition, wait for the JTAGulator's LED to turn RED.
macOS	sigrok-cli	When trying to run an acquisition, the JTAGulator will reset and miss the commands needed to start the acquisition. sigrok-cli will wait indefinitely for data that will never come.	Not available.
Windows	PulseView	JTAGulator is not properly detected.	Run sigrok-cli as shown above along with the `--scan` argument before launching PulseView.
Windows	PulseView	Acquisition is inconsistent. Sometimes the correct results are displayed, sometimes no results are displayed (possibly related to Bug 638 and/or Bug 1597).	Not available.
Windows	sigrok-cli	Only certain sample rate and time combinations return reliable acquisition results. Otherwise, incomplete or no results returned.	Not available.

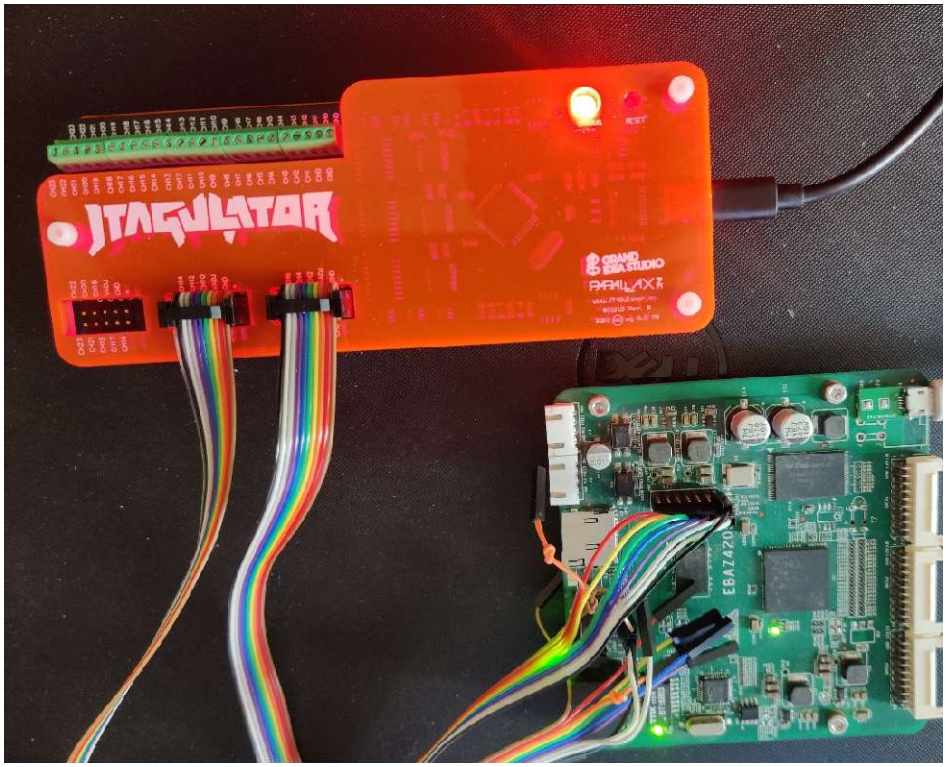
0x06Jtag 扫描功能

先看看总览，敲个 h 试试

```
JTAG> h
JTAG Commands:
J   Identify JTAG pinout
I   Identify JTAG pinout (IDCODE Scan)
B   Identify JTAG pinout (BYPASS Scan)
R   Identify RTCK (adaptive clocking)
D   Get Device ID(s)
T   Test BYPASS (TDI to TDO)
Y   Instruction/Data Register (IR/DR) discovery
P   Pin mapper (EXTEST Scan)
O   OpenOCD interface

General Commands:
V   Set target I/O voltage
H   Display available commands
M   Return to main menu
```

J、I、B 三个指令都是扫描外接的 jtag 接口，个人不认为这三者有特别大的差别，其最后目的都是 JTAG 接口，实际操作中，直接三个都试试，才是正道。
说到这，我们先不如看看什么是 jtag，这玩意是个协议？是板子上的一个接口？是芯片内部的一些设置？浅显点解释下，这是一个给芯片调试和烧录用的协议，芯片上内部首先支持这个协议，之后在器件的引脚上，留一组调试的引脚，工程师在使用这个器件的时候，会将这些引脚引出，配以调试器，用电脑上的软件就可以对芯片进行烧录，调试。因为它能对芯片进行调试的原因，很多逆向工程的第一步就是找到板子上面的 Jtag 接口，或者其他调试口，先和芯片通个信，再说其他的事情。
说的具体点 jtag 是一整套调试的方案，芯片内部有支持，电路板上有接口，调试器也有专用的， jtag 接口在很多电路板上是有引出的，工程师在生产时通过这样的接口为其烧录程序，也许是一组不起眼的排针，你要找到这个接口，之后与板子建立通讯，以达到调试的目的，对于不同的芯片有不同的上位机软件，比如赛灵思家的 fpga 就要用 vivado，如果是 ti 家的 dsp 就要用 ccs，但是他们的调试口都是 jtag 接口。
Jtag 接口有多种形式，一般来说有 14 针，20 针，10 针这几种，但是有定义的针脚仅仅有几个，其余的全部是 NC（无需连接）或者是电源引脚。我们来解释下这些引脚吧。
TCK 脚，为时钟输入引脚，当两个设备通信的时候，得有一个基准的时钟，和我们 24 小时制的时钟不一样，它仅仅是一个周期信号，两端的设备根据这个信号为基准，进行通信；
TDI 脚，信号就是从这个脚单向输入到 jtag 接口；
TDO 脚，信号从这个针脚输出；
TMS 脚，测试模式选择脚。TRST 脚，复位引脚。
当然，板子上面的调试口不一定是一组排针，有各种可能性，最简单的就是 2.54 间距的排针，或者是一组焊盘。一些鸡贼的工程师会把 jtag 接口分开来，留一些焊盘，专门设计一块带弹针的板子为其烧录程序，一些特殊的设备，会在芯片焊在板子上之前，就直接通过烧录座烧录程序，在板子上根本不留任何的烧录口，这种基本很难搞了，也就因为有这么多可能性，我们才需要这个设备来寻找 jtag 接口。
回到 jtag 这个状态下面
下面我将基于一款矿机主板来学习这个 jtag 接口嗅探功能，我这里使用的是 EBAZ4205，是一款矿机的控制板，上面留出来了两排针脚，但是我并不知道其是否为 jtag 接口，如果是 jtag 接口，接口的布局又是如何，该板上有一个排针针脚标注了 GND，所以说，不用万用表寻找 GND 了，直接连接 Jtagulator 和 EBAZ4205 的 GND，并将通道连接到你认为有可能的针脚上，现在我们就可以开始了。



一个个来吧，先试试 j 这种扫描方式，这里先行提示，直接输入 j 有时候并不能成功进入扫描，而是会出现下图中的提示，这里强调下，进入各个模式之前就应该先确认被嗅探设备的 IO 电平高低，之后需要在 jtagulator 中进行设置，进而进入嗅探模式。

```
JTAG> j
Target I/O voltage must be defined!
```

在设置好通信电平之后，我们在这里直接输入 j，紧接着我们要对通道进行设置，jtagulator 的作者确实很有思路，jtagulator 共有 24 个通道，因为通道很多，而且在一次嗅探中也用不到全部的通道，而 24 个通道组成的测试的组合一共有 620448401733239439360000 种，如果每一个都测试的话，那结果必然是测试到天荒地老。于是 jtagulator 的使用思路中有很重要的一点就是希望用户尽可能的把设备集中连到相临的通道上，以减轻设备运算的时间。当你输入 j 并回车，设备会询问你三个问题，开始的通道，结束的通道，是否要在每次单独的嗅探中将 io 的电平设置为低，如下图。

```
JTAG> j
Enter starting channel [0]:
Enter ending channel [0]: 7
Possible permutations: 1680

Bring channels LOW before each permutation? [y/N]:
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort...
---
No target device(s) found!
JTAG scan complete.
```

当你设置好了开始的通道以及结束的通道，jtagulator 还会帮你计算一共有多少种组合，上图中，选择 0 到 7 通道之后，jtagulator 回答了有 1680 种可能的组合。这也引出了 jtag 接口的嗅探方式的本质，用穷举的方式进行测试，在得到合适的回应之后停止，并汇报结果。我在本次设置中选择了 0 到 7 通道，每次设置前不将电平置低，嗯，什么都没有嗅探到。这里我们换一些设置，将每次测试之间的电平置低。如下图中设置即可，时间上会更长，因为每一次单独测试之前，需要停顿一小段时间。嗯，还是什么都没有测试到。

```
JTAG> j
Enter starting channel [0]:
Enter ending channel [7]:
Possible permutations: 1680

Bring channels LOW before each permutation? [y/N]: y
Enter length of time for channels to remain LOW (in ms, 1 - 1000) [100]:
Enter length of time after channels return HIGH before proceeding (in ms, 1 - 1000) [100]:
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort...
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
--
No target device(s) found!
JTAG scan complete.
```

下面我们测试 i 指令，设置和上面一样，但是这次我们得到了正确的结果。由于 TDI 是一个输入引脚，没有嗅探出来，但是 jtag 接口的其他部分全部都嗅探到了结果，也算是成功了。

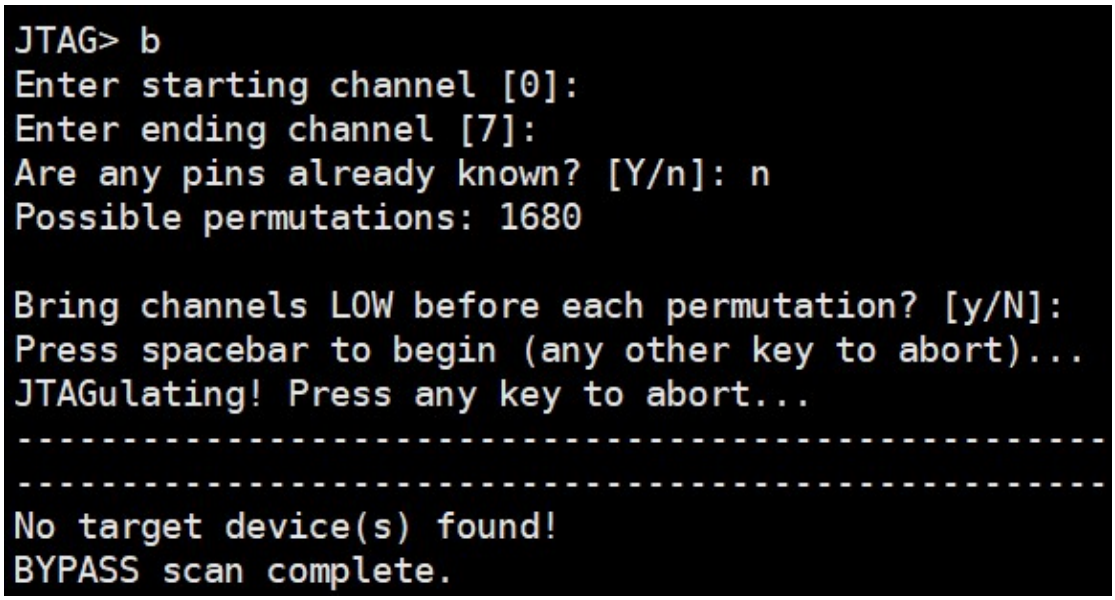
```
JTAG> i
Enter starting channel [0]:
Enter ending channel [7]:
Possible permutations: 336

Bring channels LOW before each permutation? [Y/n]: n
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort...
-
TDI: N/A
TDO: 3
TCK: 4
TMS: 5
Device ID #1: 0011 0110000010110000 00000000001 1 (0x360B0003)

--
IDCODE scan complete.

JTAG>
```

B 指令也是一样的设置，但是却没有找到 jtag 接口，个人还是建议，三种方式都测试下，有一个成功了就是成功了。



除此之外还有几个指令，r 为专门识别 RTCK 引脚的，d 用于得到设备的 id 编号，t 为 bypass 测试，y 为指令/数据寄存器测试，p 为端口映射，o 为 openocd 模式。

我们这里着重讲一下 openocd 模式，我并不打算介绍 Windows 下的安装，仅仅在 Linux 下进行安装和测试，首先我们先把安装流程教一下，依次执行下面的指令。

sudo apt-get install git

sudo apt-get install libtool autoconf texinfo libusb-dev libftdi-dev libsysfs-dev

git clone git://git.code.sf.net/p/openocd/code openocd-git

cd openocd-git

./bootstrap

./configure --enable-maintainer-mode --disable-werror --enable-buspirate

make

sudo make install

这个过程你可以给自己搞杯咖啡喝了，尤其是不建议在树莓派上面编译软件，简直了，速度慢死。如果不出意外的话上面的整个过程走完，openocd 就安装好了。但是意外天天有，我在树莓派的 debian 系统上以及 x86 的 kali 系统上都进行了测试，但是就是无法成功安装上能够使用 jtagulator 的 openocd，这一部分我未来继续进行测试。

0x07 串口扫描测试

此状态下仅仅有三条专属指令，U，T，P，都很简单，U 为寻找串口，T 为只寻找串口中的 TX 针脚，而 P 为在知晓串口针脚之后与其连接并进行直接通信的连接指令。

我们还是一个个测试，我现在将 jtagulator 的 8 到 15 通道的任意两个引脚接入到我们猜测为串口的位置上，对以上三个指令进行测试。

先进行 u 指令的测试，和 jtag 差不多，开始的通道，停止的通道，这里也会询问你是否有已经知晓的 io，剩下的几个需要设置的比较有趣，比如

“Enter text string to output (prefix with \x for hex) [CR]:”是在询问你是否知道串口中有可能传输何种字符，比如[]中的 CR 就是 carriage return，也就是回车的缩写，在与人交互的串口中，CR 是最常见的，如果你要寻找的串口传输的数据是中不包含回车，那你可以自行替换为其他字符。在

“Ignore non-printable characters? [y/N]:”选项中，推荐各位选择默认的 N 选项，但是如果你确定该串口中传递的信息内容为 ASCII 码，可以选择 Y 选项，可以剔除掉一些有可能误导设备的选项。

```
UART> u
Note: UART pin naming is from the target's perspective.
Enter starting channel [0]: 8
Enter ending channel [8]: 15
Are any pins already known? [y/N]: n
Possible permutations: 56
Enter text string to output (prefix with \x for hex) [CR]:
Enter delay before checking for target response (in ms, 0 - 1000) [10]:

Ignore non-printable characters? [y/N]:

Bring channels LOW before each permutation? [y/N]:
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort...
-------------------------------------------------------
```

一敲空格，就开始嗅探了，当它找到串口的时候会给你回传数据的内容，以及它猜测的波特率。需要注意的是，最开始的波特率可能是不对的，截取到的信息会以文本以及 hex 形式显示出来，如下图。

```
TXD: 15
RXD: 14
Baud: 57600
Data: . [ 0D ]

TXD: 15
RXD: 14
Baud: 76800
Data: .. [ F8 FC ]

TXD: 15
RXD: 14
Baud: 115200
Data: ..Password:  [ 0D 0A 50 61 73 73 77 6F 72 64 3A 20 ]
-
UART scan complete.
```

T 指令仅仅会寻找串口中的 TXD 针脚，而且是一直在监听。输入之后在各个通道上一旦有串口通信，就会显示出来，如下图。

```
UART> t
Note: UART pin naming is from the target's perspective.
Enter starting channel [0]:
Enter ending channel [0]: 15

Ignore non-standard baud rates? [y/N]:
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort...
--------------
TXD: 15
Baud (Measured): 114942
Baud (Best Fit): 115200
------------------------------------------------------
UART TXD scan complete.
```

使用指令 P，你可以连接到你之前的找到的串口了，该指令输入后，命令行中会询问你 RX，TX 连接到的通道，以及波特率，如果前面嗅探到了串口，这几个选项下直接回车就好，至于“Enable local echo? [y/N]:”则是指本地回显，即输入也显示出来，个人经验是和人交互的串口不要开回显，单片机之间通信的串口开下回显，如下图，就是一个 Linux 的通信串口，不需要打开串口回显，就可以显示我的输入，这就是本地回显的意义。

```
UART> p
Note: UART pin naming is from the target's perspective.
Enter X to disable either pin, if desired.
Enter TXD pin [15]:
Enter RXD pin [14]:
Enter baud rate [115200]:
Enable local echo? [y/N]:
Entering UART passthrough! Press Ctrl-X to exit...




Login incorrect
```

当然我们想要退出这种模式，输入 Ctrl+X 组合键就可以退回到 jtagulator 的界面。

0x08 SWD 嗅探模式

很遗憾的是我手上没有 SWD 接口的板子，但是理解凯撒不需要成为凯撒，对吧。该模式下一共两条指令，I 和 D，I 指令为嗅探 SWD 接口，基本操作与 jtag 嗅探没有什么大差别。D 指令则是在找到接口之后，找出设备的 ID 号。这个功能是最近加入的，所以并不能从该模式下直接进入 openocd 模式，个人认为这个功能在你有 jlink 烧录器的情况下才能有一定的帮助，我的意见还是等一波更新，坐等其功能更新。

0x09 结语

这个设备留给我们的不仅仅是一个安全工具，其高压隔离的设计以及器件布局十分有看头，我尝试过为其重新布局，之后找到其更棒的布局方案，但是实在是没有更好的方案了，原作者 Joe Grand 的设计确实出彩。但是回到软件上，根据其固件的更新时间，以及现阶段的部分 bug，我觉得有理由相信这位老兄在疫情之中得到了大把时间，多数新功能并不好用，比如逻辑分析仪，但是目前仍然在更新中，让我们等待更新的固件吧。