捣鼓安信可A9G开发板(GSM/GPRS+GPRS/GPS)

以前在关注乐鑫的ESP32的时候注意到了安信可的模块，也注意到了A9G。安信可的模块都挺不错的，一直印象不错，于是就买了一块A9G的开发板来玩。
本文所折腾的只是模块本身，除了前半篇是默认的AT固件，后文都是不额外加单片机操作AT指令的。有一些文档里面写了的，我没有完全搬运过来，如果这样那就要写的东西太多了。
上一次有一篇水文，在这里额外补充一些内容吧，两个MQTT开源平台ActiveMQ Artemis和EMQ X Broker图赏。
说一下本文的折腾顺序，是按从简到繁。

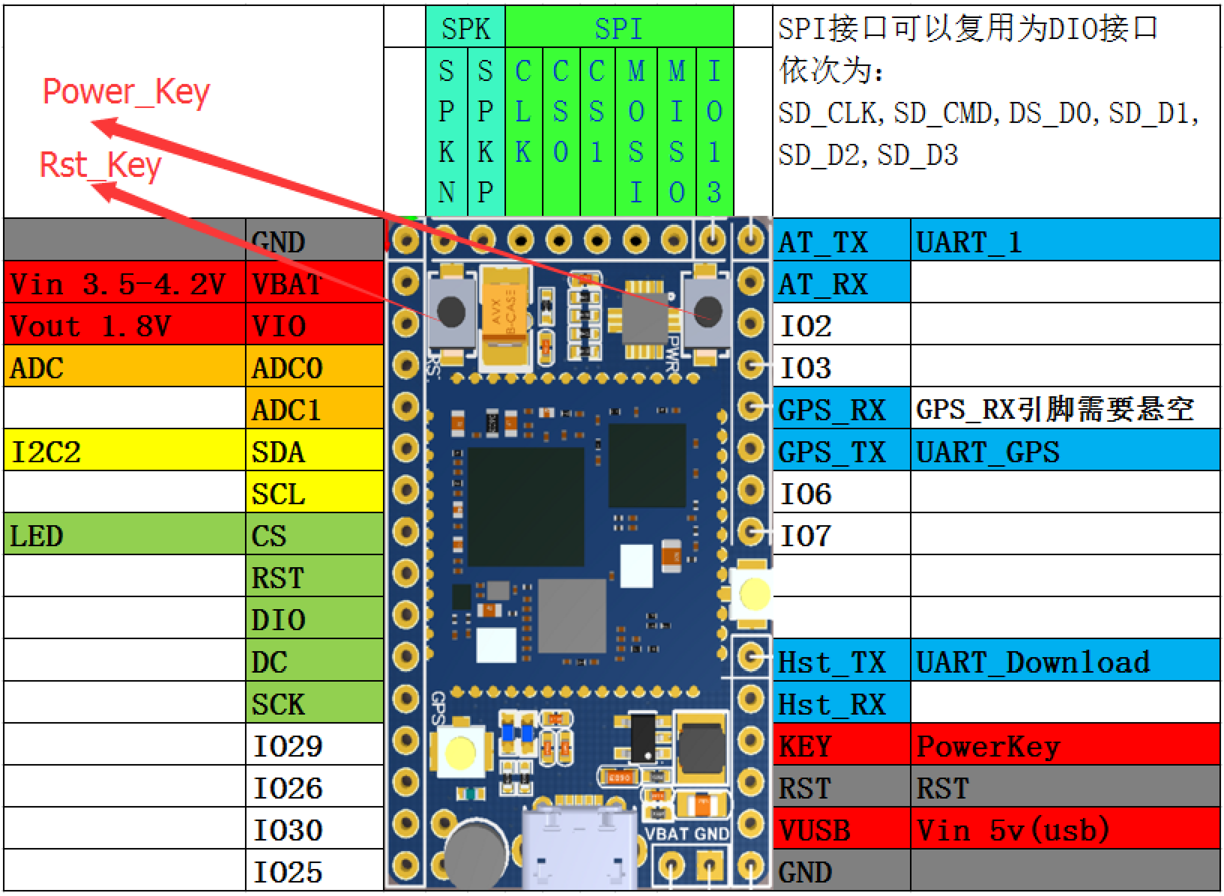
A9G的开发板将一些极其有用的外设整合到了一起，这相当好。

- 1个A9G模块（A9和A9G采用相同封装，引脚相同,所以开发板通用）
- 引出模块29个GPIO（包括2个下载调试引脚（ `HST_TX` , `HST_RX` ）
- 1个SIM卡（Micro卡）卡槽(Nano卡<Micro卡<标准卡)
- 1个TF卡卡槽
- 1个GPRS IPEX1代座子
- 1个GPS IPEX1代座子
- 1个USB接口
- 5v-4.2V DC-DC，故可以5v供电或者3.8~4.2V供电
- 1个加速度计LIS3DHx芯片
- 1个开机按键，1个复位按键
- 2个连接到GPIO的LED灯
- 1个麦克风

1. AT固件

这类模块大多都是原厂烧写默认的AT固件，就像是ESP哪一类的，这样可以降低开发成本，我这就自己折腾，就不玩AT了。
下面列举了官方的文档，已经写得很详细了：
at指令集20180825

A9/A9G AT指令操作示例大全



我从文档上面搬了几个简单例子MQTT和HTTP：
MQTT
示例：

```
AT+CGATT=1          //附着网络
OK

AT+CGDCONT=1,"IP","CMNET" //设置PDP参数
OK

AT+CGACT=1,1 //激活PDP，正确激活以后就可以上网了
OK

AT+MQTTCONN="www.anthinkerwx.com",1883,"12345",120,0,"Ai-thinker","123456" //客户端等待和连接服务器，同时发送CONNECT
OK

AT+MQTTPUB="test","124563",0,0,0                    //客户端向服务端传输一个应用消息
+MQTTPUBLISH: 1, test, 6, 124563
OK

AT+MQTTSUB="test",1,0                       //客户端向服务端发送SUB报文用于创建订阅
OK

AT+MQTTDISCONN                             //客户端发给服务端的DISCONNECT控制报文,表示客户端正常断开连接
```

HTTP

相关指令：

```
AT+HTTPGET=<url>    //统一资源标志符，可以是域名或者是IP地址
AT+HTTPPOST=<url>,<content\_type>,<body\_content>   //<content_type> : 网络文件的类型和网页的编码的内容类型
                            //<body_content> : body的文本

AT+CGATT=1  //附着网络，如果需要上网，这条指令是必选的
+CGATT:1
OK

AT+CGDCONT=1,"IP","CMNET" //设置PDP参数
OK

AT+CGACT=1,1 //激活PDP，正确激活以后就可以上网了
OK

AT+HTTPGET="http://wiki.ai-thinker.com/gprs_download"    //连接网站，请求网站资源
OK
紧接着接受的是服务器响应的信息
```

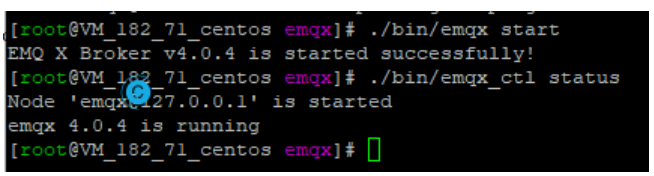
然后下面都是要重新烧录固件的二次开发了。如果你只是想了解一下，或者AT固件已经足够你使用了，可以不看下面的内容。

2. 二次开发环境搭建

2.1. EMQ X Broker(MQTT)

在文章的开头就提到了上一次的那篇水文两个MQTT开源平台ActiveMQ Artemis和EMQ X Broker图赏，只写了个图赏，里面的正文并没有内容，主要是我一开始打算讲ESP8266和ESP32连接到这两个MQTT平台，图都截好了，但是并没有很多要写的，水文也不能水过头了嘛。恰好，本篇文章写A9G连接MQTT，也就顺理成章的在这里补充没有加到那篇文章中的内容了。
安装的话直接看官方文档就好了，用户指南 (User Guide) EMQ X - 百万级开源 MQTT 消息服务器 3.2.0 文档
本篇文章的几样折腾虽然类目多，但是文档都非常的齐备，都是手把手级别的文档了，非常好。

下载，解压，启动就好了。注意放开端口组。



使用 `./bin/emqx_ctl status` 查看当前工作状态。使用 `你的IP:18083` 进入图形化面板。

这样就已经启动上了，界面的图片从上一篇水文中看就好了，这里我就不额外贴出来了。

EMQ X Broker中的功能都是通过插件来实现拓展的，我们就需要其中的一些功能，可以通过图形化面板的操作来控制开启关闭，也可以命令行，但是有的需要配置配置文件才能正常工作。
这里我开启了两个插件(在系统原有的基础上，额外打开的)。

- emqx_auth_username(EMQ X Authentication with Username and Password)
- emqx_auth_mysql(EMQ X Authentication/ACL with MySQL)(MySQL 认证)
- emqx_auth_clientid(EMQ X Authentication with ClientId/Password)
- emqx_web_hook(WebHook)

当然，一开始是可以通过默认用户名和密码登录到图形化面板的，可以在上面修改登录密码。但是这个时候的设备连接是不受限的，不要用户名和密码就可以连接，我们为了科学一点，去掉这个匿名登录。参考文档认证（认证鉴权）

进入 `/etc/emqx.conf` ，找到：

```
##--------------------------------------------------------------------
## 2.2. Authentication/Access Control
##--------------------------------------------------------------------

## 2.3. Allow anonymous authentication by default if no auth plugins loaded.
## 2.4. Notice: Disable the option in production deployment!
##
## 2.5. Value: true | false
allow_anonymous = false
```

将这一条的 `allow_anonymous` 改为 `false` 。

然后打开 `emqx_auth_username` 插件。使用 `.bin/emqx_ctl plugins load emqx_auth_username` 。

打开了以后还需要添加登录的用户名和密码（设备用的）。

```
./bin/emqx_ctl users add cyqsd2 cyqsdpwd
```

日志是这样：

```
[root@VM_182_71_centos emqx]# ./bin/emqx_ctl plugins load emqx_auth_username
[root@VM_182_71_centos emqx]# ./bin/emqx_ctl users add cyqsd2 cyqsdpwd
ok
[root@VM_182_71_centos emqx]# ./bin/emqx_ctl users list
cyqsd2
```

还可以打开 `emqx_auth_mysql` ，连接到Mysql数据库，emqx_auth_mysql 支持访问 MySQL 实现 连接认证、访问控制功能。。MySQL 认证/访问控制插件
下面的内容**照搬的文档**：
首先得部署好Mysql，执行下面的SQL。
MQTT 用户表

```
CREATE TABLE `mqtt_user` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(100) DEFAULT NULL,
  `password` varchar(100) DEFAULT NULL,
  `salt` varchar(35) DEFAULT NULL,
  `is_superuser` tinyint(1) DEFAULT 0,
  `created` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `mqtt_username` (`username`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

注解：插件同样支持使用自定义结构的表，通过 auth_query 配置查询语句即可。

MQTT 访问控制表

```
CREATE TABLE `mqtt_acl` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `allow` int(1) DEFAULT NULL COMMENT '0: deny, 1: allow',
  `ipaddr` varchar(60) DEFAULT NULL COMMENT 'IpAddress',
  `username` varchar(100) DEFAULT NULL COMMENT 'Username',
  `clientid` varchar(100) DEFAULT NULL COMMENT 'ClientId',
  `access` int(2) NOT NULL COMMENT '1: subscribe, 2: publish, 3: pubsub',
  `topic` varchar(100) NOT NULL DEFAULT '' COMMENT 'Topic Filter',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `mqtt_acl` (`id`, `allow`, `ipaddr`, `username`, `clientid`, `access`, `topic`)
VALUES
    (1,1,NULL,'$all',NULL,2,'#'),
    (2,0,NULL,'$all',NULL,1,'$SYS/#'),
    (3,0,NULL,'$all',NULL,1,'eq #'),
    (5,1,'127.0.0.1',NULL,NULL,2,'$SYS/#'),
    (6,1,'127.0.0.1',NULL,NULL,2,'#'),
    (7,1,NULL,'dashboard',NULL,1,'$SYS/#');
```

配置配置文件，在 `etc/plugins` 中， `emqx_auth_mysql.conf` 。这里只是我列出来的几条。

```
## Mysql 服务器地址
auth.mysql.server = 127.0.0.1:3306

## Mysql 连接用户名
## auth.mysql.username =

## Mysql 连接密码
## auth.mysql.password =

## Mysql 认证用户表名
auth.mysql.database = mqtt
```

还有密码加盐操作等其他操作就看文档了。

然后启动就好了。

还有就是 `emqx_web_hook` 应该启动，这个插件的作用就是将EMQ X Broker的消息转发到HTTP服务器上，这就又少了很多很多事。
具体的使用吗，又可以专门看它的文档：
emqx / emqx-web-hook

```
client.connected
{
    "action":"client_connected",
    "client_id":"C_1492410235117",
    "username":"C_1492410235117",
    "keepalive": 60,
    "ipaddress": "127.0.0.1",
    "proto_ver": 4,
    "connected_at": 1556176748,
    "conn_ack":0
}
client.disconnected
{
    "action":"client_disconnected",
    "client_id":"C_1492410235117",
    "username":"C_1492410235117",
    "reason":"normal"
}
client.subscribe
{
    "action":"client_subscribe",
```

当然，直接用Websocket连接也挺好的。

2.2. CSDTK 4

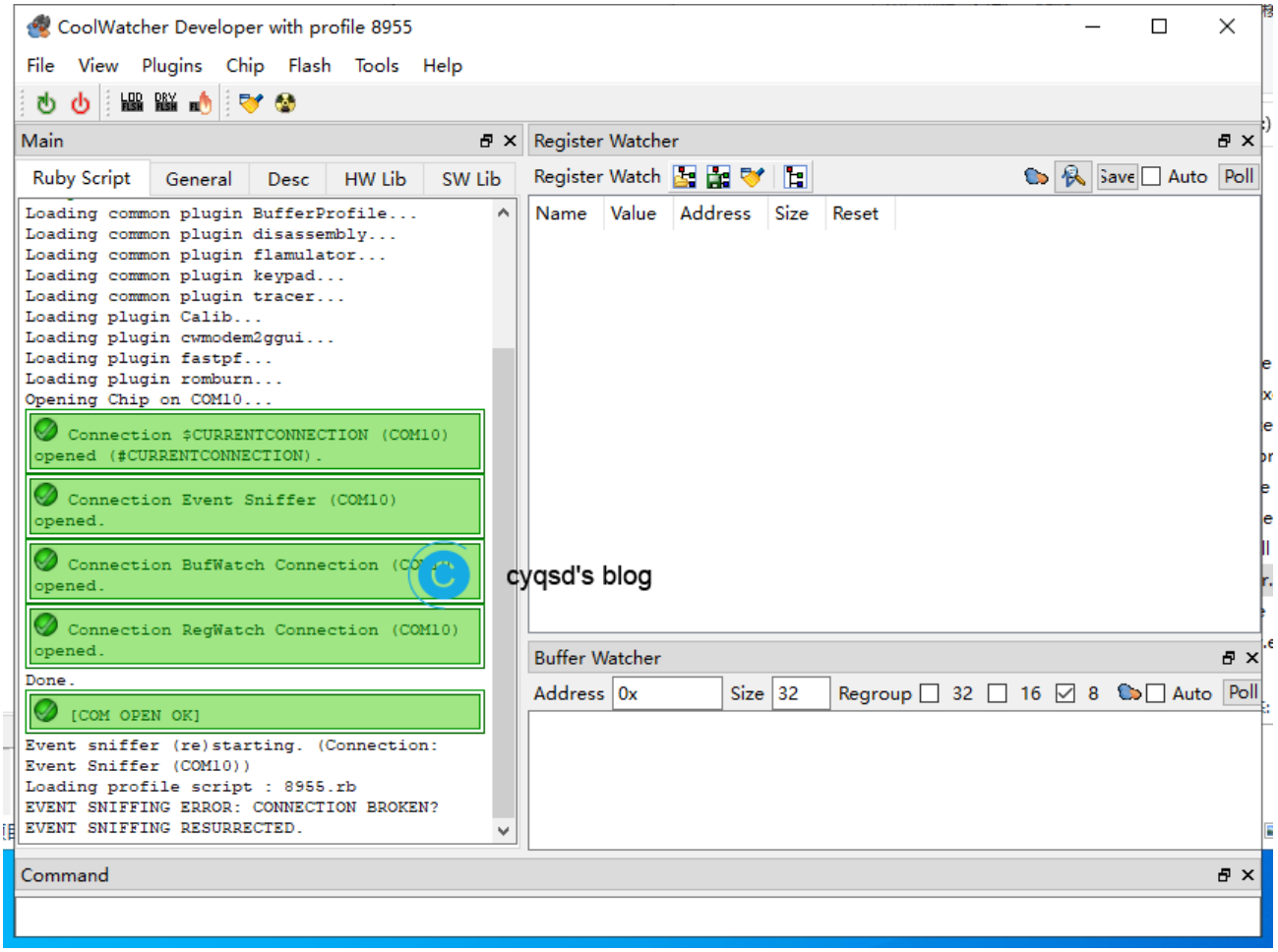
上面是MQTT服务器的部署，如果不需要的话，也可以跳过，但是这一步不能跳过，这一小节是编译环境，官方提供了一个视频教程：A9 A9G GPRS-C-SDK环境建立及SDK使用简介。

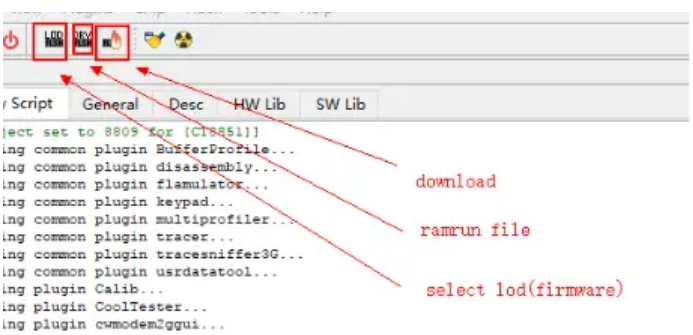
> CSDTK 4是基于纯Windows应用的编译环境，不再基于cygwin。因此对不同的Windows版本会有更好的兼容性，而且可以很方便地集成到其他各种开发环境中。
>
> CSDTK 4提供的是一个压缩包，可以解压到任何地方使用（以下以C:\CSDTK4为例）。
>
> CSDTK 4只包含编译需要的工具，不包含 `svn, git` 等版本管理软件。
>
> CSDTK 4以及工程所在的路径不要包含空格，中文字符等特殊字符。
>
> 以前的代码可能不能在CSDTK 4下编译，需要合并新代码中的改动，只要是 `compilerules.mk` 以及 `usrgen, resgen` 相关的改动。

我就偷懒了，直接设置了环境变量指向到根目录。

```
GPRS_CSDTK42_PATH        D:\BaiduNetdiskDownload\CSDTK42
```

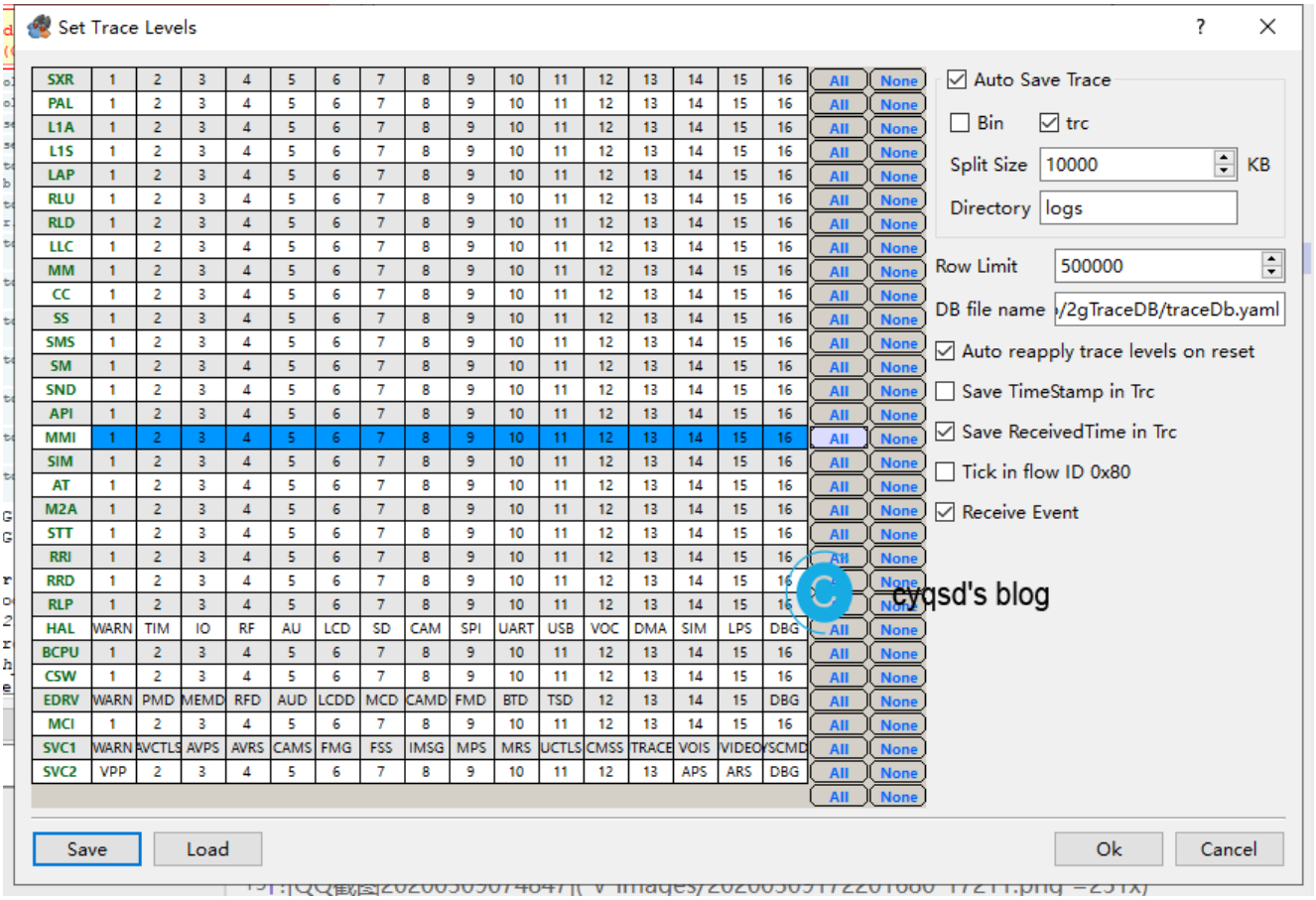
二次开发烧写的工具是coolwatcher(关于里面写着，Devlopped by RDATol Team ； RDA Microelectronic)，用着还凑合，真是凑合，小巧，但是功能齐全。
在 `CSDTK42\cooltools` 目录中的 `coolwatcher.exe` 。coolwatcher中需要指定 `.lod` 文件： `host_8955_flsh_spi32m_ramrun.lod`

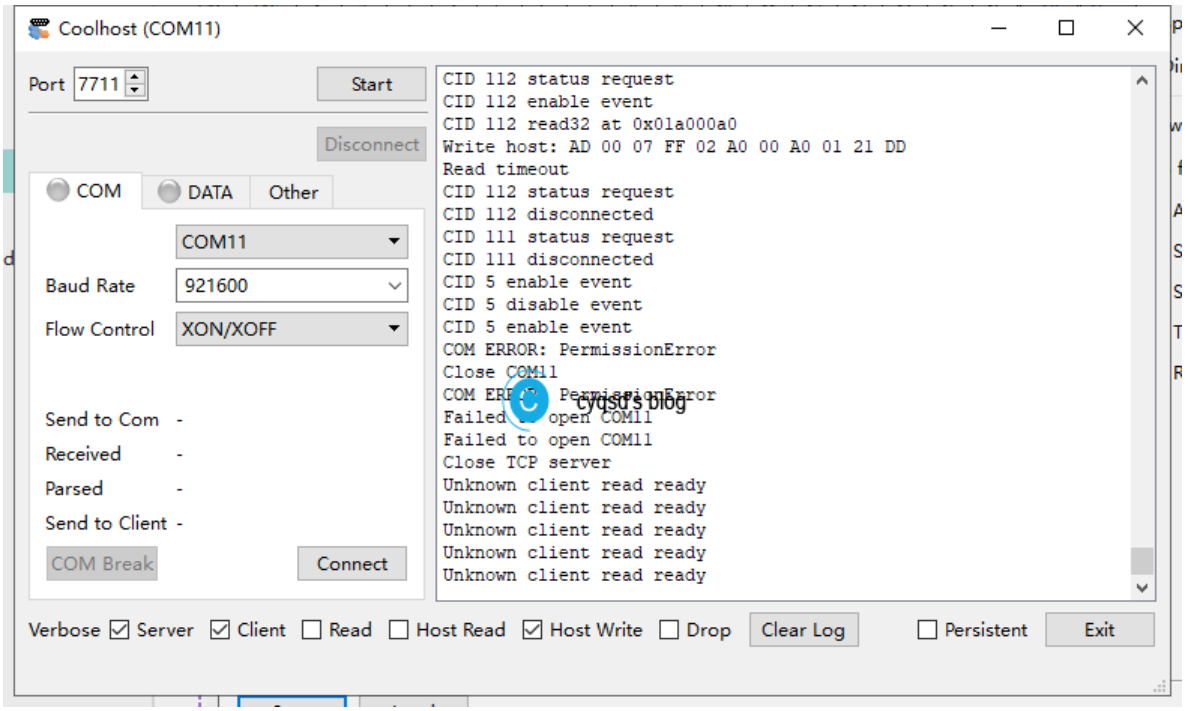


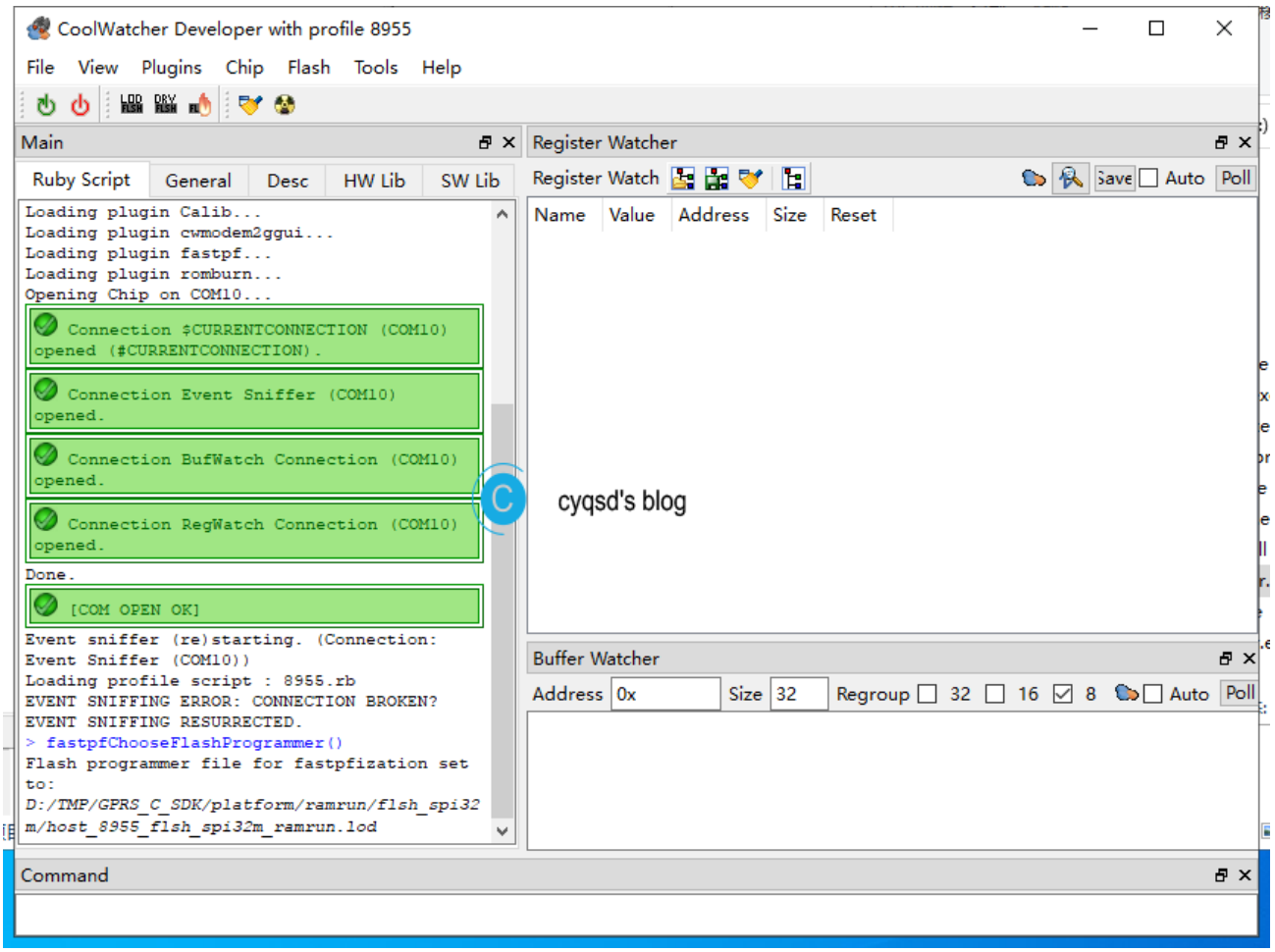


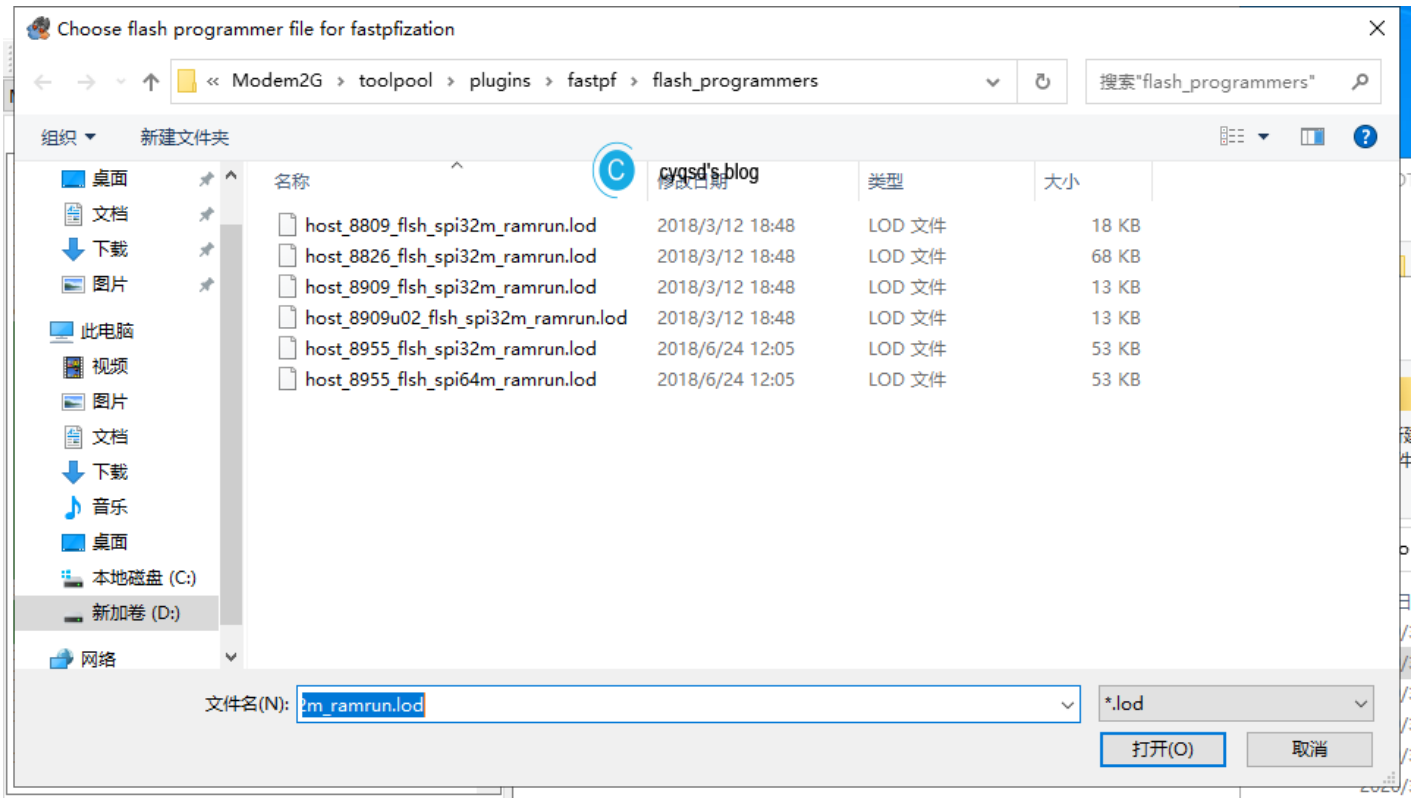
点击 DRY 图标，选择名字包含 8955 ramrun spi32m .lod 的文件（在工程 platform/ramrun 下或者coolwatcher cooltools\chipgen\Modem2G\toolpool\plugins\fastpf\flash_programmers 目录下）。

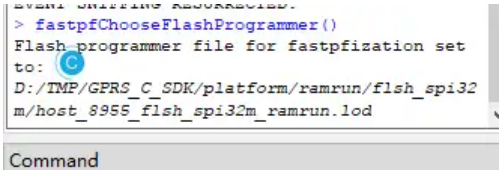
调整Trace等级，可以减少抓取一些作用不是特别大的调试信息。

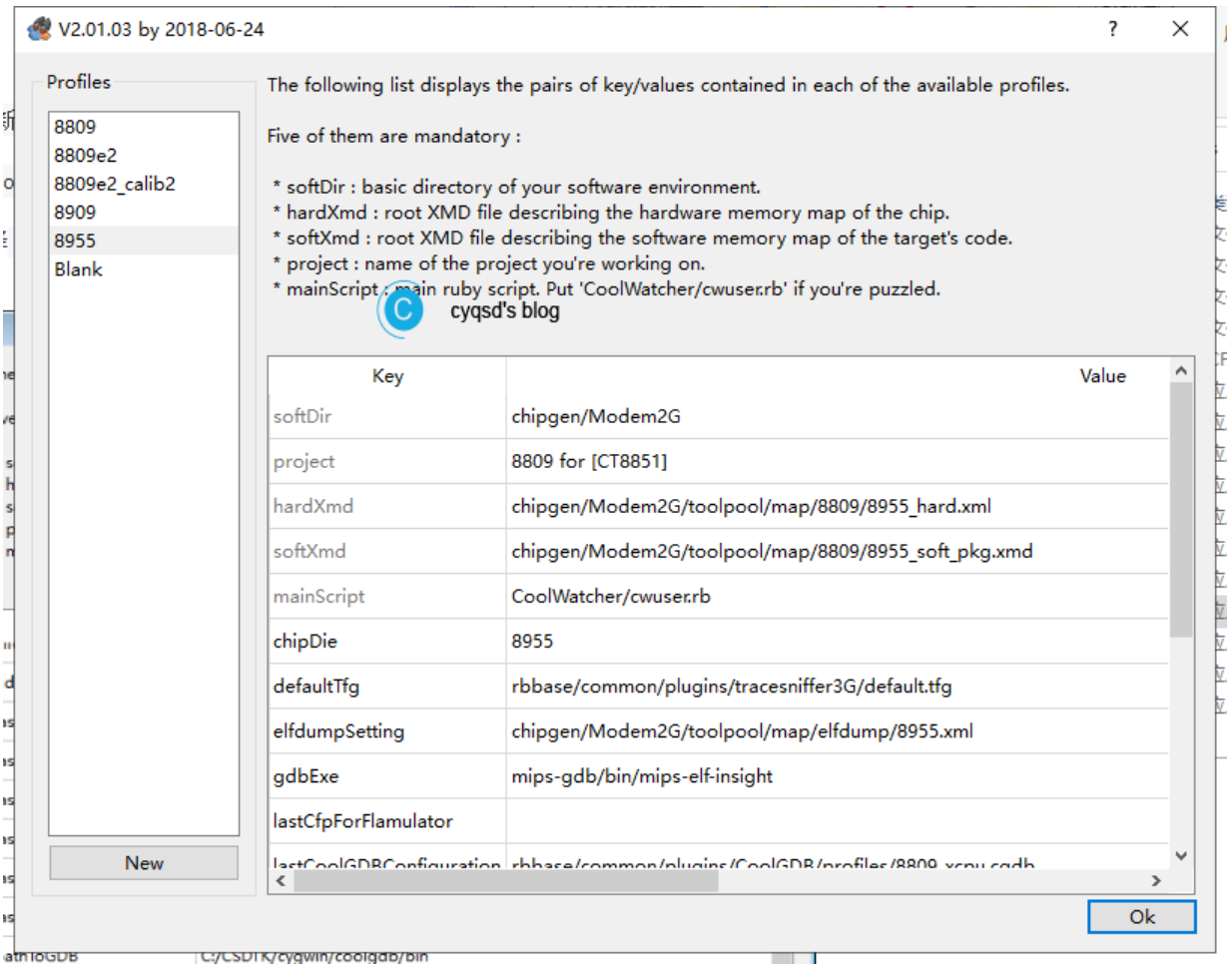





CoolWatcher Developer with profile 8955
File View Plugins Chip Flash Tools Help
Main
Ruby Script General Desc HW Lib SW Lib
Loading plugin Calib...
Loading plugin cwmodem2ggui...
Loading plugin fastpf...
Loading plugin romburn...
Opening Chip on COM10...
Connection $CURRENTCONNECTION (COM10) opened (#CURRENTCONNECTION).
Connection Event Sniffer (COM10) opened.
Connection BufWatch Connection (COM10) opened.
Connection RegWatch Connection (COM10) opened.
Done.
[COM OPEN OK]
Event sniffer (re)starting. (Connection: Event Sniffer (COM10))
Loading profile script : 8955.rb
EVENT SNIFFING ERROR: CONNECTION BROKEN?
EVENT SNIFFING RESURRECTED.
> fastpfChooseFlashProgrammer()
Flash programmer file for fastpfization set to:
D:/TMP/GPRS_C_SDK/platform/ramrun/flsh_spi32m/host_8955_flsh_spi32m_ramrun.lod
Register Watcher
Register Watch
Save Auto Poll
Name Value Address Size Reset
cyqsd's blog
Buffer Watcher
Address 0x Size 32 Regroup 32 16 8 Auto Poll
Command


Choose flash programmer file for fastpfization
« Modem2G › toolpool › plugins › fastpf › flash_programmers
搜索"flash_programmers"
组织 新建文件夹
桌面
文档
下载
图片
此电脑
视频
图片
文档
下载
音乐
桌面
本地磁盘 (C:)
新加卷 (D:)
网络
名称 修改日期 类型 大小
cyqsd's blog
host_8809_flsh_spi32m_ramrun.lod 2018/3/12 18:48 LOD 文件 18 KB
host_8826_flsh_spi32m_ramrun.lod 2018/3/12 18:48 LOD 文件 68 KB
host_8909_flsh_spi32m_ramrun.lod 2018/3/12 18:48 LOD 文件 13 KB
host_8909u02_flsh_spi32m_ramrun.lod 2018/3/12 18:48 LOD 文件 13 KB
host_8955_flsh_spi32m_ramrun.lod 2018/6/24 12:05 LOD 文件 53 KB
host_8955_flsh_spi64m_ramrun.lod 2018/6/24 12:05 LOD 文件 53 KB
文件名(N): m_ramrun.lod
*.lod
打开(O)
取消


> fastpfChooseFlashProgrammer()
Flash programmer file for fastpfization set to:
D:/TMP/GPRS_C_SDK/platform/ramrun/flsh_spi32m/host_8955_flsh_spi32m_ramrun.lod
Command



使用命令 `build.bat demo aprs` 进行编译操作，具体你是那个项目就是那个，下面的说明是引用的官方文档的GPRS C SDK 开发环境搭建。

保证环境可以使用后，就可以在CMD或powershell窗口中使用 `build.bat` 脚本来编译工程，有以下参数：

- 使用 `./build.bat $PROJ` 来编译你的应用模块，如 `./build.batsh app` 则是编译app目录下的源码
- 使用 `./build.bat demo $PROJ` 来编译demo目录下的特定例程，比如 `./build.bat demo gpio`
- 使用 `./build.bat clean $PROJ` 清除 `$PROJ` 目录的中间文件
- 使用 `./build.bat clean all` 清除所有中间文件
- 使用 `./build.bat demo $PROJ release` 来生成release版本，比如 `./build.bat demo gpio release` ， 如果最后一个参数不是 `release` ，则默认是 `debug` 版本， `debug` 版本在死机后会停止运行并可以使用GDB调试，而**release版本加入了看门狗功能，在死机时会自动重启系统**，所以**实际投入使用时请使用release版本以防止出现bug时死机**，测试时使用debug版本

比如：

```
./build.sh demo gpio
```

进行编译，编译会生成一个 `build` 目录，编译完成会在 `hex` 目录下生成两个格式为 `lod` 的文件，这就是我们用来下载到开发板的目标文件

```
D:\TMP\cyqsd-blog\Demo\GPRS_C_SDK>build.bat demo aprs
build folder exist
number of processors: 2
Build host is WINDOWS

MAKE            init

MAKE            libs

MAKE            libs/gps

MAKE            libs/gps/minmea

PREPARING       libgps_debug.a

MAKE            libs/utils
```

```
PREPARING         liblibs_debug.a

MAKE              demo/aprs
CC                demo_aprs.c
```

还有很多细节前文没有提到，详细的，去看官方文档更好，额外搬运太多，并没有实际意义。

3. GPS Tracker

gps tracker就是我们平常说的GPS定位器，每隔一段时间上报一下当前数据，然后可以设置电子围栏之类的。我相信，就是只是用gps tracker这一个Demo，对服务器发起GET，POST请求就可以解决绝大多数问题。我这里的折腾也就是围绕着这个Demo了，不额外修改了，其实要修改也是很简单的事，自己开发一个服务器端，接收

在这个上面花这么大功夫没有太大作用，还不如用MQTT来实现，这样能保持连接，少了很多事，这在下一小节就是

官方提供的Demo里面就有写好了的，我提取了一小段代码上来，就不额外写了。

例程中的 `Http_Post` 函数就是请求的http post。

```
//http post with no header
int Http_Post(const char* domain, int port,const char* path,uint8_t* body, uint16_t bodyLen, char* retBuffer, int bufferLen)


snprintf(temp,2048,"POST %s HTTP/1.1\r\nContent-Type: application/x-www-form-urlencoded\r\nConnection: Keep-Alive\r\nHost: %s\r\nContent-Le
                        path,domain,bodyLen);
char* pData = temp;
int fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
```

上面是网络请求的，下面是GPS请求的：

```
//open GPS hardware(UART2 open either)
GPS_Init();
GPS_SaveLog(true,GPS_NMEA_LOG_FILE_PATH);
```

`GPS_SaveLog` 就可以实现保存数据到内存卡了。 `GPS_NMEA_LOG_FILE_PATH` 指的是保存的文件名称和目录，预先宏定义在文件头了。

```
#define GPS_NMEA_LOG_FILE_PATH "/t/gps_nmea.log"
```

`/t/` 要加，文件系统挂载在这上面。
如果不使用 `GPS_SaveLog` 这个函数，还可以使用 `demo/fs/demo_fs_new.c` 中的例程, `SaveData` ， fs就是指的是filesystem。我一开始不知道能用 `GPS_SaveLog` 。要注意有没有 `GPS_NMEA_LOG_FILE_PATH` 。

```
bool SaveToTFCard2(char* str)
{
    int32_t fd;
    int32_t ret;
    uint8_t *path = (uint8_t*)GPS_NMEA_LOG_FILE_PATH;

    fd = API_FS_Open(path, FS_O_RDWR|FS_O_APPEND | FS_O_CREAT, 0);
    if ( fd < 0)
    {
        Trace(1,"Open file failed:%d",fd);
        return false;
    }
    ret = API_FS_Write(fd, (uint8_t*)str, strlen(str));
    API_FS_Close(fd);
    if(ret <= 0)
        return false;
    return true;
}
```

上面初始化了GPS，拿到数据并格式化输出到Trace。

```
snprintf(buffer,sizeof(buffer),"GPS fix mode:%d, BDS fix mode:%d, fix quality:%d, satellites tracked:%d, gps sates total:%d, is fixed:%s,
                                gpsInfo->gga.fix_quality,gpsInfo->gga.satellites_tracked, gpsInfo->gsv[0].t
```

还有一些电压和IMEI。

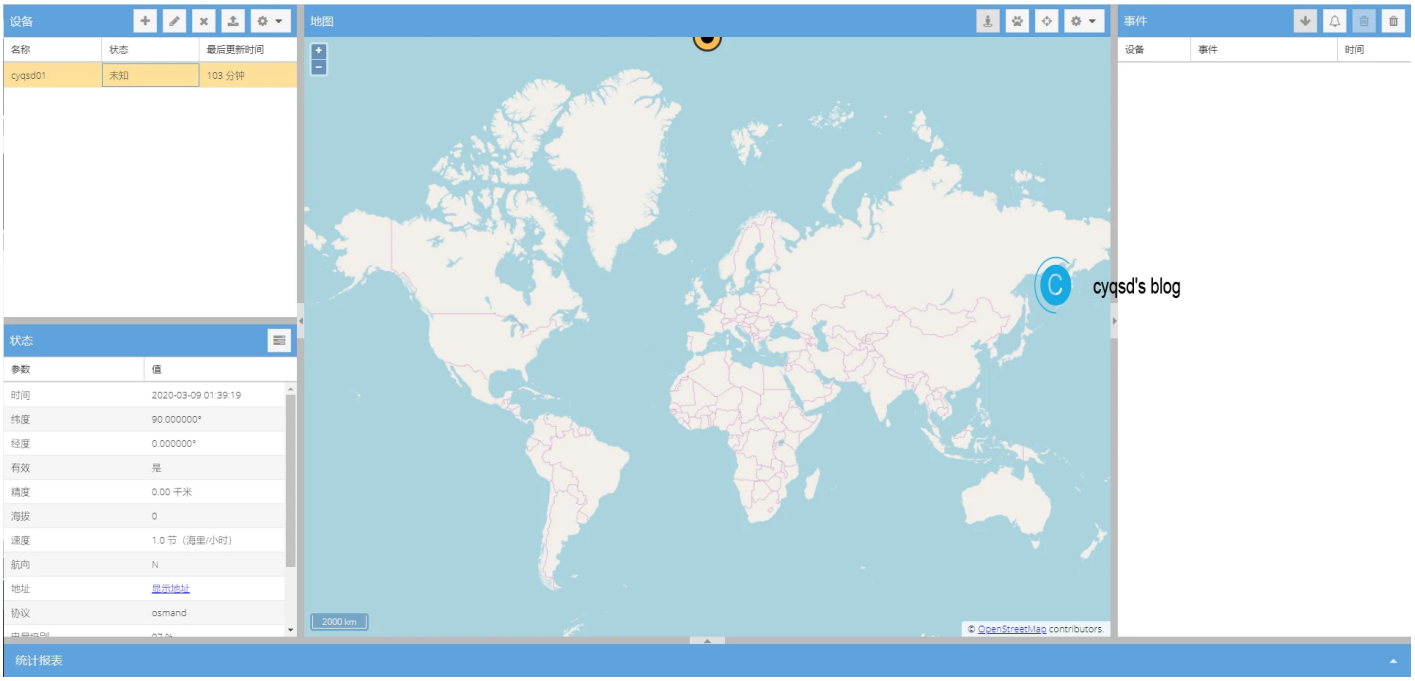
```
uint8_t percent;
uint16_t v = PM_Voltage(&percent);
Trace(1,"power:%d %d",v,percent);
memset(buffer,0,sizeof(buffer));
if(!INFO_GetIMEI(buffer))
    Assert(false,"NO IMEI");
Trace(1,"device name:%s",buffer);
snprintf(requestPath,sizeof(buffer2),"/?id=%s&timestamp=%d&lat=%f&lon=%f&speed=%f&bearing=%.1f&altitude=%f&accuracy=%.1f&batt=%.1f",
                                    buffer,time(NULL),latitude,longitude,isFixed*1.0,0.0,gpsInfo->gga.altitude,0.0,percent*1.0);
if(Http_Post(SERVER_IP,SERVER_PORT,requestPath,NULL,0,buffer,sizeof(buffer)) <0 )
    Trace(1,"send location to server fail");
else
{
    Trace(1,"send location to server success");
    Trace(1,"response:%s",buffer);
}
```

上面的就是单片机的代码。如果没有其他需求，我们直接修改配置，烧录进去就好。官方的例程是为了一个叫Traccar的开源tracker服务器适配的，Traccar提供公用的服务器，但是因为完全开源，是使用Java开发的，我们可以自行搭建一个私有服务器。Traccar
我不建议你在局域网上面部署，因为A9G直接就走的GSM，上的外网，你搭建局域网想连接上来反倒是很麻烦。
我依旧使用的上面MQTT的CentOS7，具体上官网的下载页面可以看到：https://www.traccar.org/download/
因为就是Java开发的，跑起来并不费力。https://www.traccar.org/documentation/

```
[root@VM_182_71_centos ~]# ./traccar.run
Creating directory out
Verifying archive integrity... All good.
Uncompressing traccar  100%
```

通过你的 `IP+:8082` 进入图形化界面，就像是下面的这样。

下面是登陆后的主界面:



当然，一开始是没有设备的，设备要手动添加。IMEI就是A9G上的二维码。再图中，我的经纬度是因为没有定位显示的，但是连接是没有问题的。
如果并没有连接上，就检查设备输出的Trace看错误出在什么地方。如果并没有修改过Demo依旧有问题的话，那可能是模块没上网，我偶尔也遇到些玄学问题。

4. MQTT和GPS

上面折腾了GPS Tracker这个demo，这里我们折腾MQTT，好连接上EMQ X Broker。修改官方自带的例程mqtt和mqtt_ssl足够了。
这一小节被我额外分出去成一篇独立的内容了。可以移步：安信可A9G使用官方例程实现MQTT和GPS

5. A9G使用MicroPython进行开发

这一小节也被我额外分出去了。
可以移步：安信可A9G使用MicroPython进行开发

6. Arduino和STM32开发

Arduino和STM32都是使用的AT固件，STM32官方文档提供了简单的例程，Arduino可以参考下面类似的：

GSM Testing
Topic: IDE support for Ai-Thinker A9G GPRS + GPS board

7. 总结

A9G的这个模组，特别是这个现成的开发板价格很便宜，经常用到的功能都整合起来了，无需再去买周边了，少了很多事，整体的折腾来说都是很顺利的，文档资料很齐全，对二次开发很友好，难免会遇到一些小问题。我用的CH340C做的USB转TTL，不是官方推荐的CP2102。有经常烧写不正常的问题，我不觉得完全是这个转换芯片的问题。
你还可以使用微信小程序等支持Websocket的东西，连接到服务器，就可以订阅这些内容进行操作了。