iovBOX 开发手册

易通星云（北京）科技发展有限公司

修订记录

版本	时间	作者	描述
0.1	2018.3.22	陈莞	文档创建。
0.2	2018.3.24	周小波	补充设备介绍，环境搭建，开发示例
0.3	2018.3.26	陈莞	补充调试技巧，附录 2，附录 4。
0.4	2018.3.27	陈莞	补充 CAN 示例,插件使用方法
0.5	2018.3.28	陈莞	补充附录 3，修改部分描述，修改附录 1
0.7	2018.3.28	吕昱	补充示意图，修改部分描述
0.8	2018.3.29	陈莞	修改 CAN 用例,修改设备介绍以及部分描述
0.9	2018.3.29	陈莞	补充 GPS 用例，修改用例格式。
1.0	2018.3.30	陈莞	调整格式，增加图片
1.1	2018.3.30	陈莞	修改代码，以及部分错误
1.2	2018.4.4	陈莞	更改图片，修改用例
1.3	2018.4.11	陈莞	增加 CAN 高级用例
1.4	2018.4.17	陈莞	修改 GPS 用例

目录

1 产品介绍

1.1 产品概述

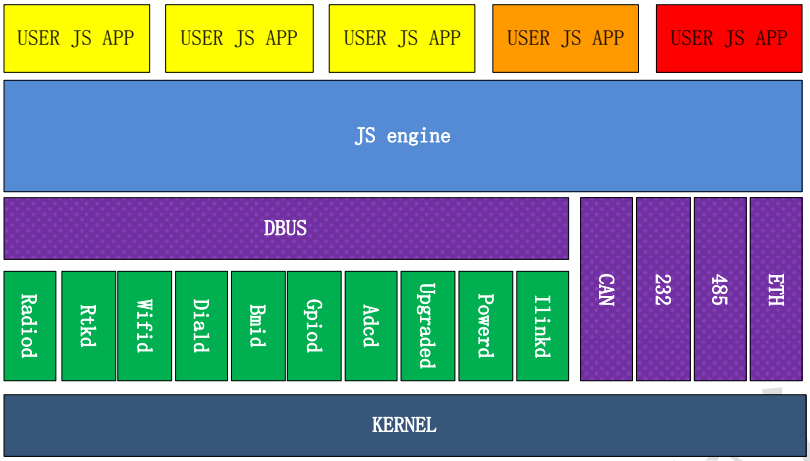
iovBOX 设备（以下简称 iovBOX）按工业级产品严酷应用环境而设计，适合专用车联网系统的车端嵌入式应用二次开发，iovBOX 也能用于工业自动化领域的特色应用定制开发（技术指标见附录 1）。

iovBOX 集成了两路以太网口、全网通 4G 模块、WiFi 模块和蓝牙模块，具有强大稳定的网络通信功能。iovBOX 采用高性能的 ARM 处理器，内置高可靠的 GPS 模块、SD 卡、MEMS 惯性导航器件、温度传感器，并提供 RS485、CAN 工业现场总线接口和丰富的 GPIO 功能（详见附录 2）。iovBOX 的电源系统符合严格的车规要求，并内置锂电池能在外部供电断开的情况下持续工作，并回传信息。iovBOX 还具备开盖监测报警的功能，适合应用在安全性要求高的应用场景。

iovBOX 内置了 JS 引擎库的支持，可以使用 JavaScript 开发应用程序，对 iovBOX 进行操作。

1.2 系统划分

iovBOX 软件构架如下图，iovBOX 支持 JavaScript 应用程序基于 DBUS 服务访问小数据量的硬件接口，也支持通过 JavaScript 的标准接口访问大数据量的网络、RS485 和 CAN 总线接口。



图表 1 系统划分图

2 开发环境

2.1 设备及附件

设备清单如下

设备及清单

设备及附件	数量
iovBOX	1 个
以太网线	1 根
GPS 天线	1 根
Wifi 天线	1 根
26Pin 线	1 根
电源适配器	1 个
CANalyst-II 设备	1 个
MicroUSB 线	1 根
iovBOX 开发手册	1 份
API 手册	1 份

2.2 硬件连接

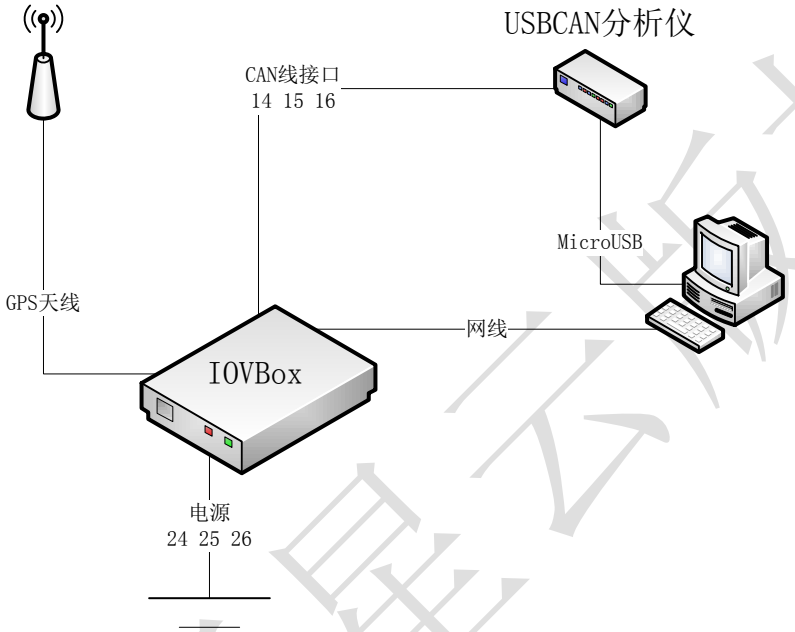
将 iovBOX 的 26pin 安普标准引脚座中的 14，15，16 引脚（26PIN 线序编号见附录 2）分别连接到 CANalyst-II 分析仪的 CAN1 口的 S，L，H 上，将 CANalyst-II 分析仪通过 MicroUSB 线连接到 PC 上。

将 iovBOX 的 GPS 天线接上，并放置在足够空旷的位置。

由于 iovBOX 以太网地址为 192.168.84.88，将 iovBOX 任一以太网口与电脑连接，并将电脑 IP 地址改为与 iovBOX 同一网段。

最后通过将电源适配器和电源端口连上，启动 iovBOX。

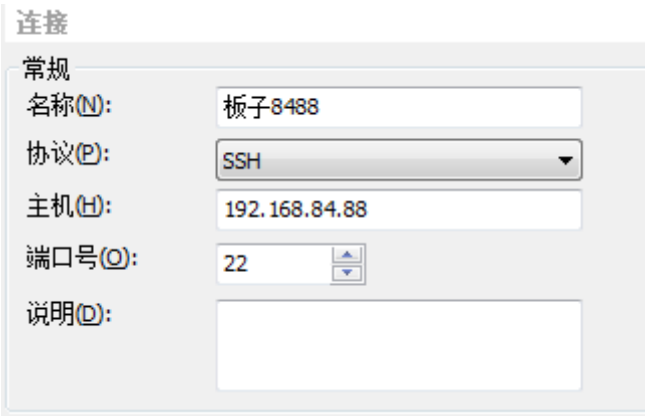
iovBOX 具体连接如下图



图表 2 iovBOX 连接示意图

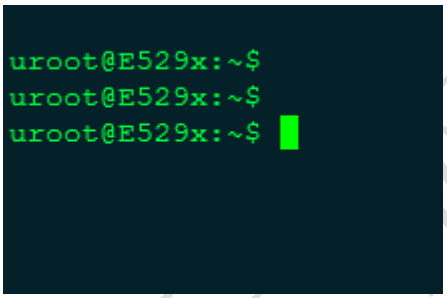
2.3 用户登录

iovBOX 上电后，通过 SSH 可以登录 iovBOX，SSH 登录配置如下图，注意主机为 192.168.84.88，为 iovBOX 的以太网地址。SSH 登录用户名和密码为 uroot / 123456。



图表 3 登录配置界面

SSH 登录成功后，可以对盒子进行操作，如下图



图表 4 登录成功界面

2.4 环境配置

2.4.1 安装工具

开发工具使用 Visual Studio Code，有完整的插件支持，同时也支持远程调试 iovBOX 应用程序的 JavaScript 代码。

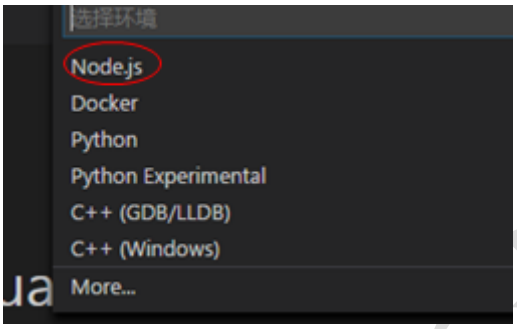
首先在您的 Windows PC 上安装 Visual Studio Code，要求版本号高于 1.20.0.安装完成后打开 Visual Studio Code，我们开始配置一个远程开发环境。

2.4.2 配置工具

步骤 1.打开 Visual Studio Code，“菜单”->“文件”->“打开工

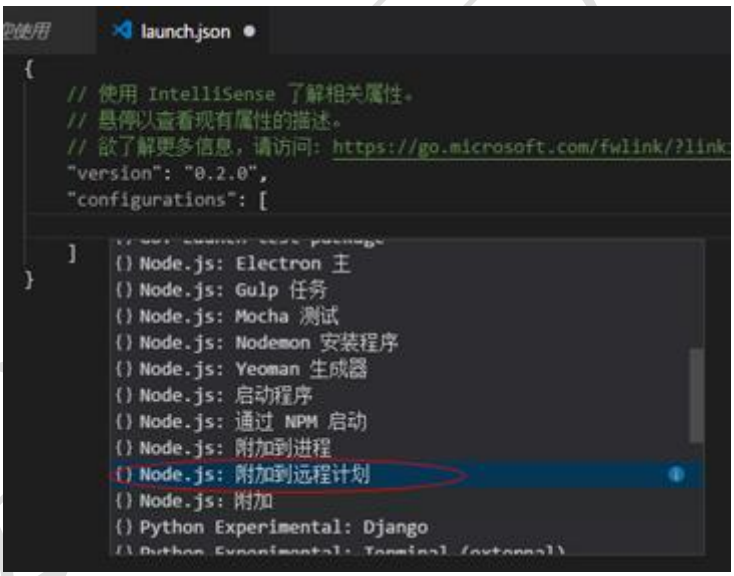
作区...”，任意选定一个文件夹作为工作区。

步骤 2.选择“菜单”->“调试”->“添加配置...”，出现如下提示框，选择 Node.js：



图表 5 VSC 配置提示框

步骤 3.自动弹出 launch.json，并出现如下图提示，滚动选择“() Node.js: 附加到远程计划”，自动填充部分配置选项：



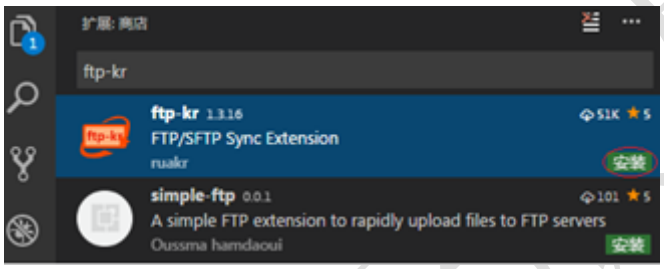
图表 6 VSC 配置选项提示框

步骤 4.按照如下图所示修改配置选项。

```
"configurations": [
    {
        "type": "node",
        "request": "attach",
        "name": "Attach to Remote",
        "address": "192.168.84.88",
        "port": 8580,
        //"localRoot": "${workspaceFolder}",
        //"remoteRoot": "your code running path"
    }
]
```

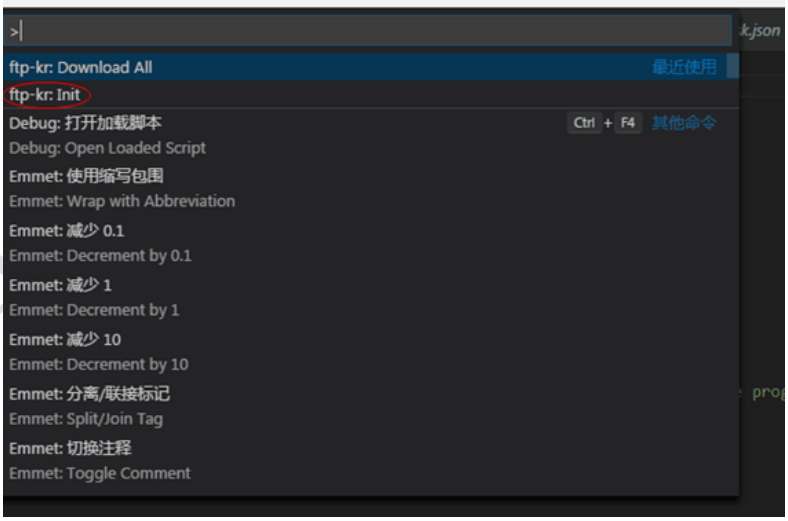
图表 7 VSC 配置参数

步骤 5. 安装 ftp-kr 插件，安装后，点击重新刷新。



图表 8 VSC 安装插件

步骤 6. 配置 ftp-kr 插件，输入 Ctrl+p，输入>，输入 ftp-kr：Init，如下图所示：

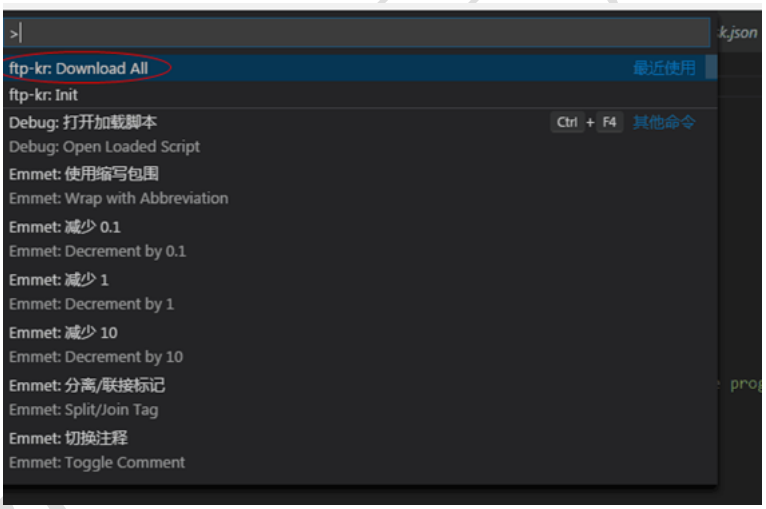


图表 9 ftp-kr 初始化选项

步骤 7. 按照如下图所示配置 ftp-kr：

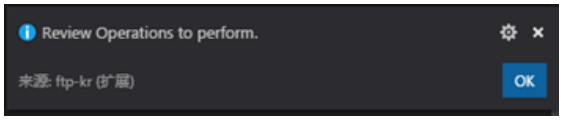
图表 10 ftp-kr 配置

步骤 8. 输入 Ctrl+p，输入>，输入 ftp-kr：Download All，下载 iovBOX 上的 JavaScript 演示代码



图表 11 ftp-kr 下载代码

步骤 9.在弹出的窗口点击 OK，完成代码下载，如下。



图表 12 ftp-kr 下载完成

步骤 10. 修改 launch.json，链接本地代码与 iovBOX 上的远程代码，如下。

```
"configurations": [
    {
        "type": "node",
        "request": "attach",
        "name": "Attach to Remote",
        "address": "192.168.84.88",
        "port": 8580,
        "localRoot": "${workspaceFolder}",
        "remoteRoot": "/home/uroot/"
    }
]
```

图表 13 链接远程代码

步骤 11. 执行 node –inspect=8580 –debug-brk demo.js，可以看到如下提示。

```
[dare@localhost NodeServ]$ node --inspect=8580 --debug-brk KY_websock.js
Debugger listening on port 8580.
Warning: This is an experimental feature and could change at any time.
To start debugging, open the following URL in Chrome:
```

图表 14 执行调试代码

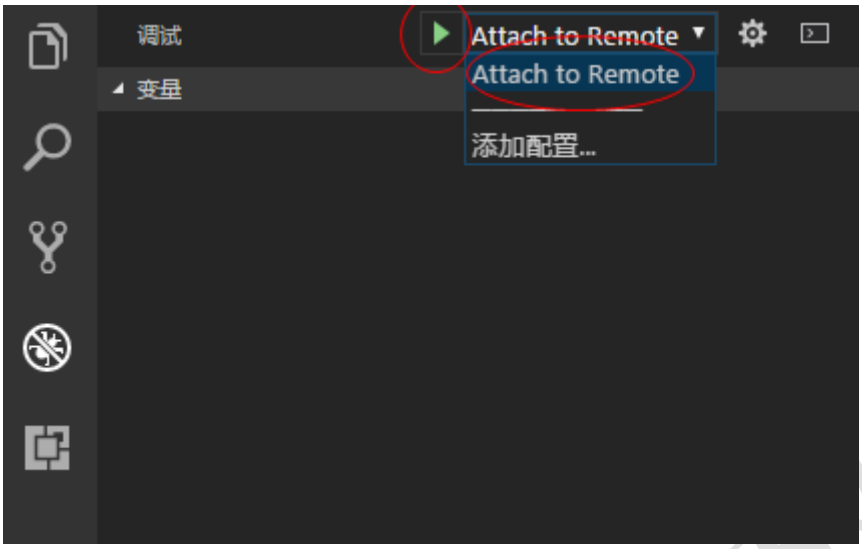
这里需要注意三点:

① 请无视最后的从 chrome 打开 URL 的提示,如果需要使用 chrome 调试，请参考附录 3.

② 选项--inspect=8580，其中 8580 为端口号，默认端口号为 9229。

③ 选项--debug-brk，在没有断点的时候，必须添加此选项以保证可以看到完整主线代码。

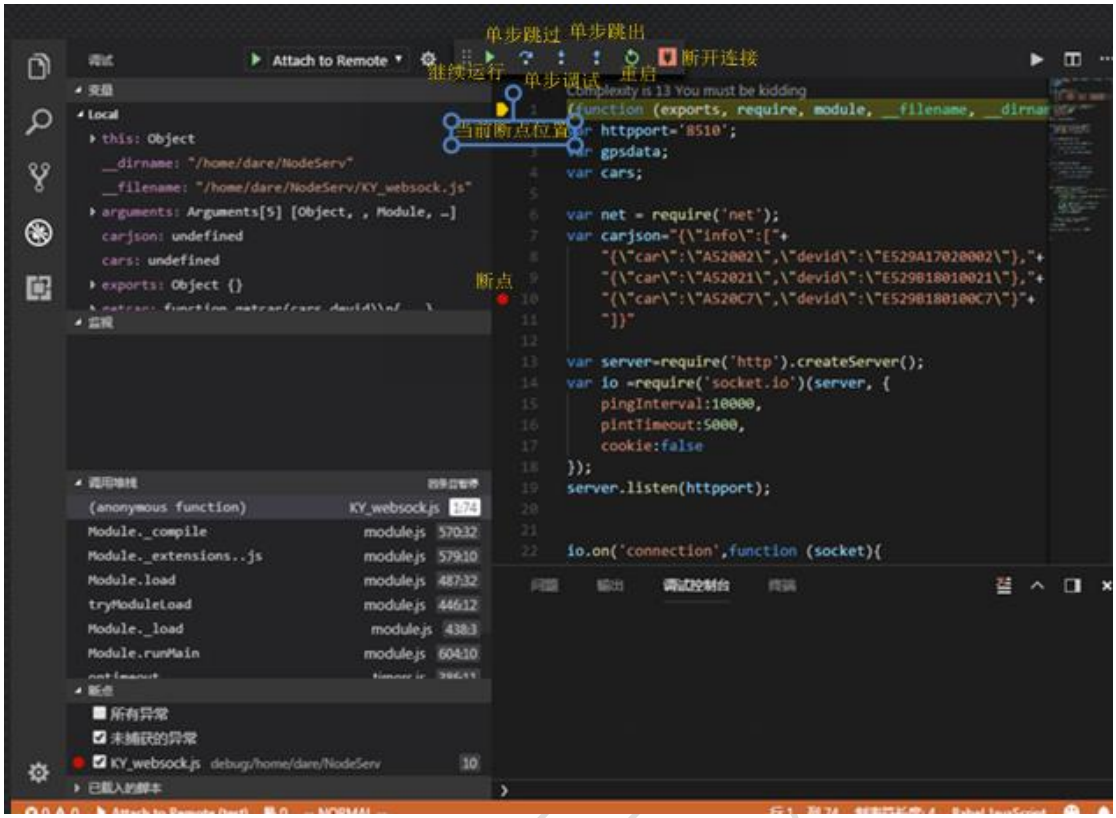
步骤 12. 使用“菜单”->“调试”（或者快捷键 Ctrl+Shift+D），按照如图所示选择 Attach to Remote，并点击“调试”按钮，开始远程调试。



图表 15 调试选项

步骤 13. 进入如下调试界面，按钮功能如图上注释，同时请注意以下两点：

1. 重启后，初始化行为中的断点会被自动略过，所以如果要调试初始化行为，请在 iovBOX 上重新运行 node –inspect –debug-brk 命令.
2. 由于远程代码跟本地代码有链接，实际修改的是本地代码，然后通过右键命令（Upload this）或者是打开自动上传后的保存命令自动上传代码，完成代码同步，实现远程调试的效果。代码同步以后，需要重新启动 Node --inspect --debug-brk 命令来刷新代码。



图表 16 VSC 调试界面

3 开发示例

本手册提供基于 DBUS 服务访问和直接访问 CAN 接口两个典型示例，其中一个是使用 DBUS 接口访问 GPS 服务获取经纬度等信息，一个是直接访问 CAN 接口并收发数据。

3.1 GPS 示例

3.1.1 示例说明

这里借 GPS 的使用示例来说明 DBUS 接口的开发流程。

在 iovBOX 中运行了一个 GPSD 的进程，如下图所示，该进程为一个 DBUS 服务进程。

```
uroot@E529x:~$
uroot@E529x:~$ ps -af | grep "gpsd" | grep -v "grep"
 1200 root     /usr/sbin/gpsd -g -L 6
uroot@E529x:~$
```

图表 17 GPSD 进程

GPSD 进程主要实现了以下功能：

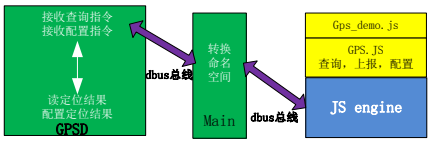
（1）为其他进程与之通信提供服务；

（2）按照一定格式解析来自其他进程的 GPS 配置指令和查询指令；

（3）GPS 实际定位点状态查询；

（4）向其他进程返回 GPS 定位点结果。

GPSD 进程解耦了 JS 引擎与底层硬件，我们在 JS 引擎上二次开发 GPS 定位点相关的 JavaScript 应用程序时， JS 引擎不能直接对 GPS 的硬件进行控制，它需要通过 DBUS 总线，发送一定格式的 GPS 配置指令和查询指令到 GPSD，GPSD 实际控制输出或配置输入，然后将定位点结果或查询结果通过 JS 引擎送给 JS 应用程序。JS 引擎通过 GPSD 对 GPS 进行操作框图如下：

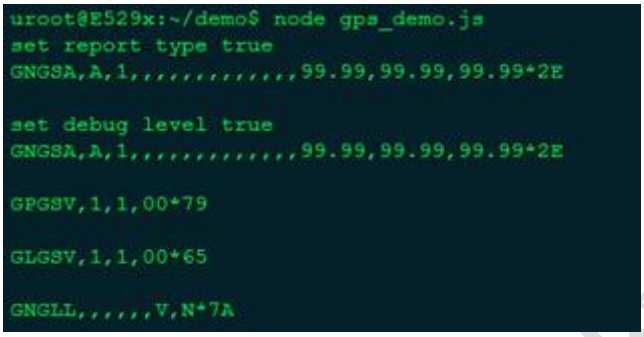


图表 18 GPSD 交互

3.1.2 代码运行步骤以及结果

步骤 1.登录 iovBOX，在运行目录下执行 Node gps_demo.js

步骤 2.在天线放置在足够空旷的地方下，会出现如下打印，表示运行成功。



3.1.3 代码实现以及说明

```
var dbus_app = require('dbus-js/dbus_app.js');
var dbus_obj = new dbus_app();
```

生成 DBUS 服务总线。

```
var gpslib = require('dbus-js/gps.js');
var gps = new gpslib(dbus_obj);
```

在 DBUS 服务总线上申请注册 GPS 服务。

```
dbus_obj.register_app_name();
```

DBUS 服务初始化，会将已经申请的服务统一初始化。

```
gps.onReportNMEA(function(res) {
    console.log('report nmea :'+ (res));
});
```

注册打印函数为 GPS 定位点 NMEA 上报服务。

```
gps.onReportData(function(res) {
    console.log('report nmea :'+ JSON.stringify(res));
});
```

注册打印函数为 GPS 定位点 JSON 上报服务。

```
gps.setReportEnable(1, function(res) {
    console.log('set report enable :'+ JSON.stringify(res));
})
```

设置 GPS 定位点上报使能（1）。

```
gps.setReportType(2,(res)=>{
    console.log('set report :'+ JSON.stringify(res));
})
```

设置 GPS 定位点上报格式为 JSON 字符串（2）

```
gps.setDebugLevel(7,(res)=>{
    console.log('set debug level :'+ JSON.stringify(res));
})
```

设置 GPS 模块打印级别为 7。（具体接口见附录 4）

3.2 CAN 示例

3.2.1 示例说明

CAN 使用了 iovBOX 的 SocketCan 实现，将 CAN 接口虚拟成 Socket 接口，同时与 JS 引擎的 SocketCan 插件包对接。在配置好 iovBOX 的 SocketCan 以后，能够在 JS 引擎中以类似实现 Socket 的方式实现 CAN 的数据传输，同时在 JS 引擎的 SocketCan 插件包中还集成了对于 CAN 数据解析的 XML 支持（KCD 文件）。

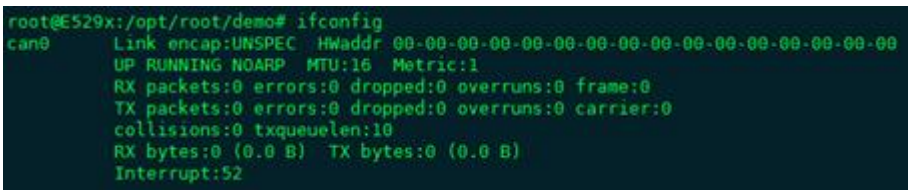
3.2.2 代码运行步骤以及结果

步骤 1.　SSH 登录 iovBOX，并在 iovBOX 上使用如下命令配置 SocketCan 链接：

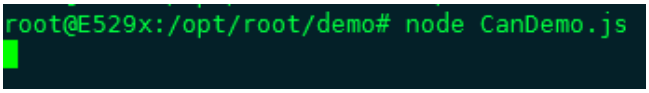
sudo ip link set can0 type can bitrate 1000000 > /dev/null 2>&1

sudo ip link set can0 up > /dev/null 2>&1

步骤 2. 运行 ifconfig，如下图表示成功。

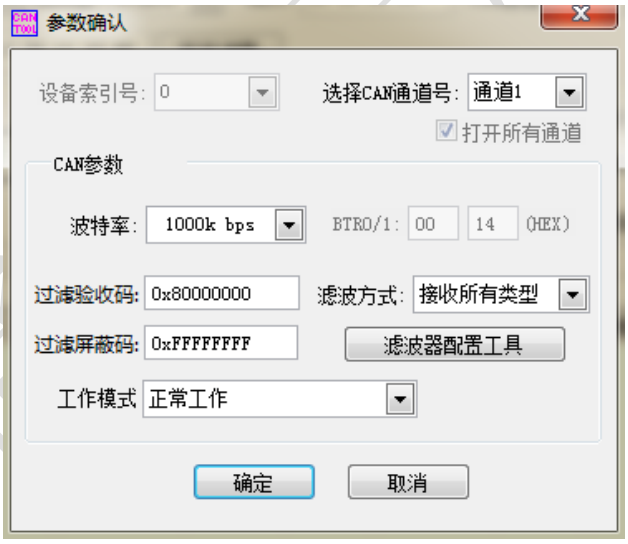


步骤 3.　使用 Node CanDemo.js 命令运行 CanDemo



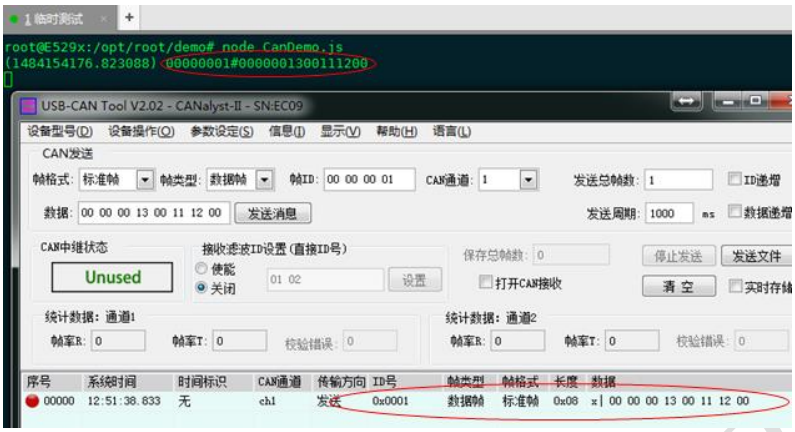
步骤 4. 打开 USB_CAN_TOOL，菜单->设备操作->启动设备

步骤 5. 设置设备参数如下:



步骤 6. 点击发送。

步骤 7. 在 iovBOX 界面可以看到 CAN 总线上收到的信息。



3.2.3 代码实现以及说明

```
var vcan=require('socketcan');
```

申请 socketcan 实例。

```
var channel = vcan.createRawChannel("can0", true /* ask for timestamps */);
```

创建与 can0 链接的通道 channel（can0 为 socketcan 配置的名称）

```
function toHex(number) {
  return ("00000000" + number.toString(16)).slice(-8);
}
function dumpPacket(msg) {
  console.log('(' + (msg.ts_sec + msg.ts_usec / 1000000).toFixed(6) + ') '
+
    toHex(msg.id).toUpperCase() + '#' +
msg.data.toString('hex').toUpperCase());
}
channel.addListener("onMessage", dumpPacket);
```

实现打印函数 dumpPacket 并挂载到通道 channel 的消息处理事件上。

```
channel.start();
```

监听 can0 上接受的消息。

3.3 CAN 高级应用示例

3.3.1 示例说明

CAN 高级示例中除了使用了 iovBOX 的 SocketCan 实现，同时使用了对于 CAN 数据解析的 XML 支持（KCD 文件）。通过使用 KCD 文件完成了针对 DBC 文件的 CAN 数据解析。其中 DBC 向 KCD 文件的转化，需要使用另外的工具 CAN_BABEL，详见附录 5.

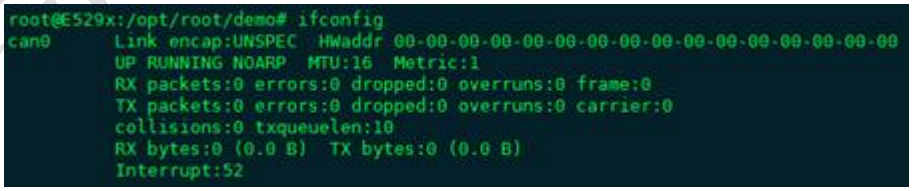
3.3.2 代码运行步骤以及结果

步骤 1. SSH 登录 iovBOX，并在 iovBOX 上使用如下命令配置 SocketCan 链接：

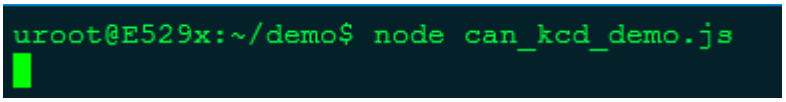
sudo ip link set can0 type can bitrate 1000000 > /dev/null 2>&1

sudo ip link set can0 up > /dev/null 2>&1

步骤 2. 运行 ifconfig，如下图表示成功。

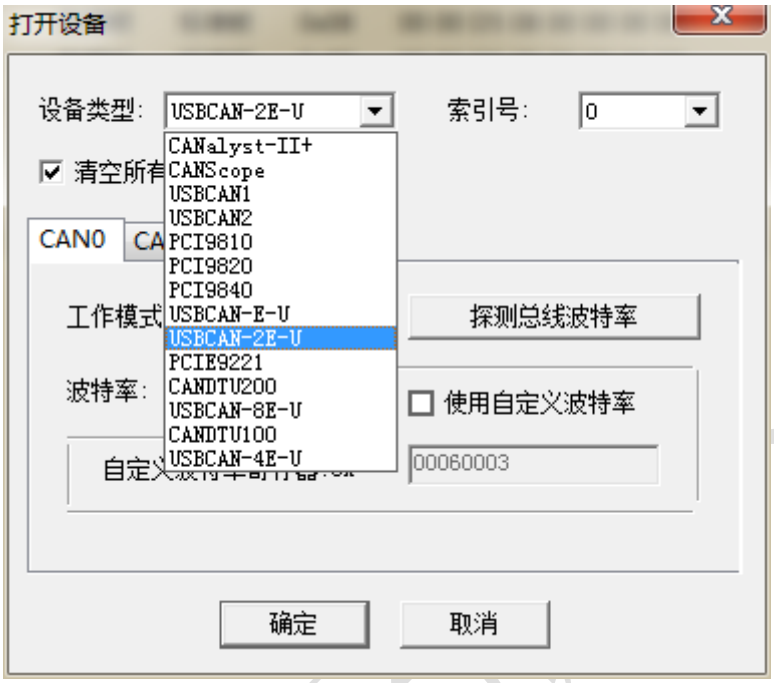


步骤 3. 使用 Node can_kcd_demo.js 命令运行 can_kcd_demo

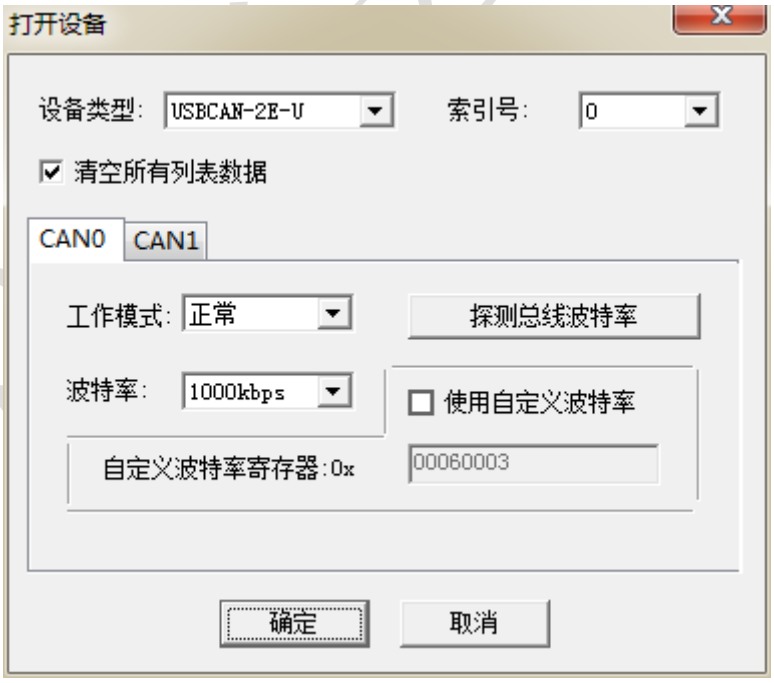


步骤 4. 打开 CANPro，具体安装方式详见附录 6，菜单->操作->

启动系统，选择如下系统。

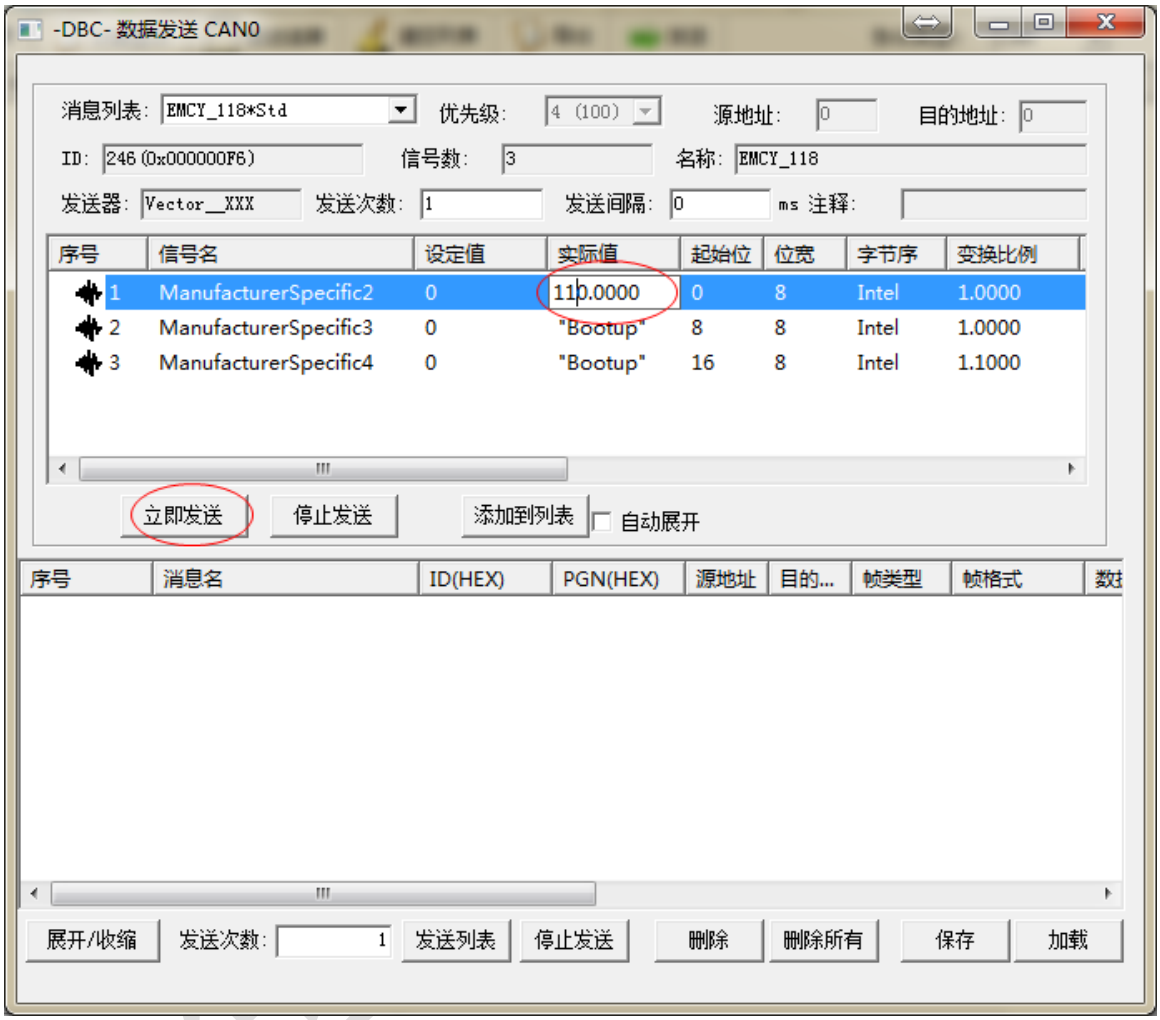


步骤 5. 设置设备参数如下:



步骤 6. 菜单->操作->DBC 解析，在新窗口中，选择发送按钮。

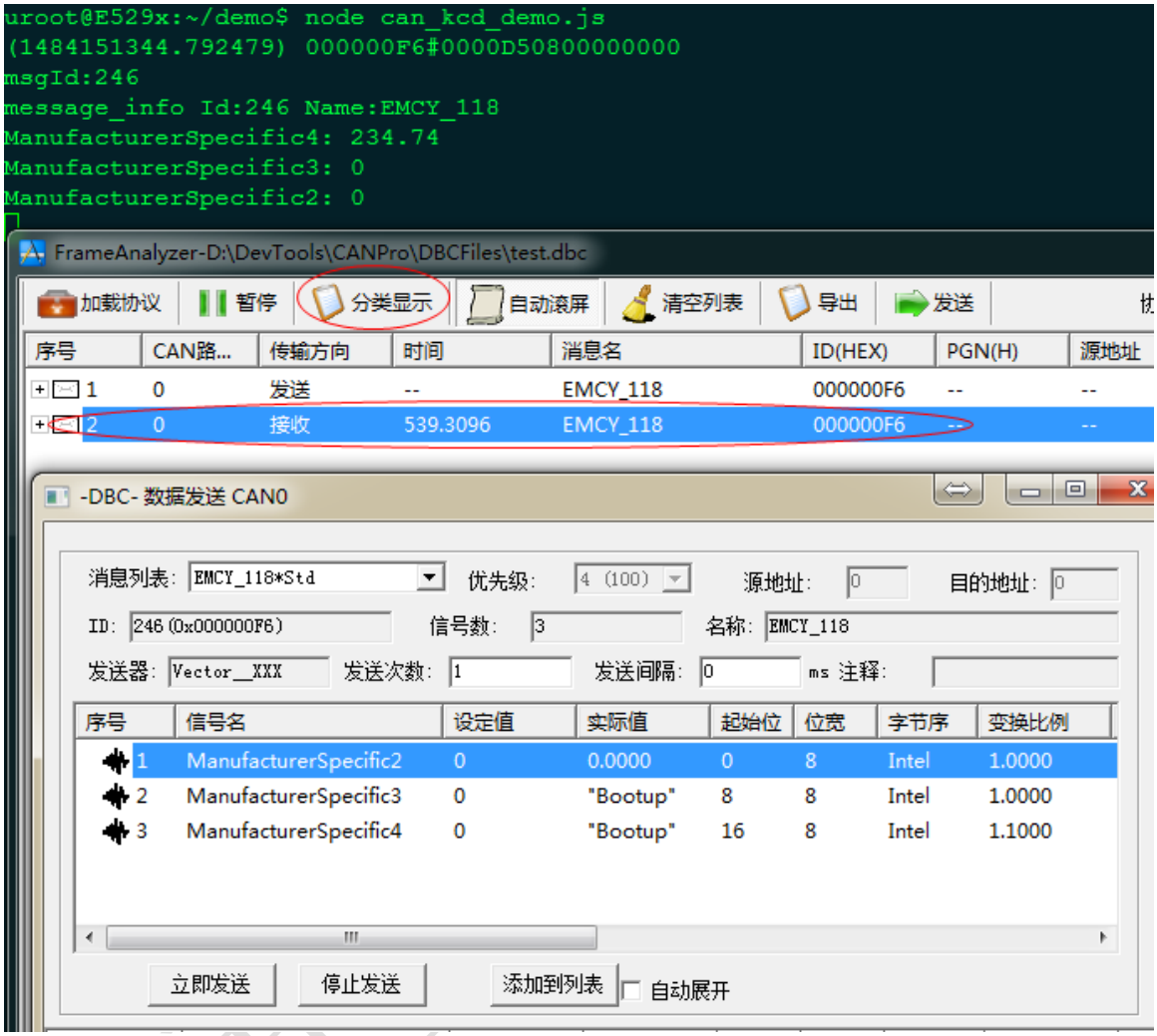
步骤 7. 在弹出界面可以看到 DBC 的格式，并且可以设定发送的信息。



步骤 8. 点击发送后，可以在 IOVBox 上看到被解析的 CAN 数据信息，并且回发一条信息给 CANPro。

```
uroot@E529x:~/demo$ node can_kcd_demo.js
(1484151344.792479) 000000F6#0000D50800000000
msgId:246
message_info Id:246 Name:EMCY_118
ManufacturerSpecific4: 234.74
ManufacturerSpecific3: 0
ManufacturerSpecific2: 0
```

步骤 8. 可以在 CANPro 的 FrameAnalyzer 窗口，点击“分类显示”，可以看到接收到的 CAN 数据，跟代码中设定的数据值一致。



3.3.3 代码实现以及说明

```
var vcan=require('socketcan');
```

申请 socketcan 实例。

```
var channel = vcan.createRawChannel("can0", true /* ask for timestamps */);
```

创建与 can0 链接的通道 channel（can0 为 socketcan 配置的名称）

```
var network = vcan.parseNetworkDescription("test.kcd");
var db      = new vcan.DatabaseService(channel, network.buses["Private"]);
```

配置 test.kcd 的分析数据库 db。

```
function toHex(number) {
  return ("00000000" + number.toString(16)).slice(-8);
```

```
}
   //打印数据内容并回发指定数据
function dumpPacket(msg) {
  console.log('(' + (msg.ts_sec + msg.ts_usec / 1000000).toFixed(6) + ') '
+ toHex(msg.id).toUpperCase() + '#' +
msg.data.toString('hex').toUpperCase());
    // 显示消息
    let message_info = db.messages[msg.id];
    console.log('msgId:' + msg.id);
    console.log('message_info Id:' + message_info.id + ' Name:' +
message_info.name);
    //显示数据中各个数据值的内容
    let db_messages = db.messages[message_info.name];
    for (let signal_name in message_info.signals)
    {
        let db_signal = db_messages.signals[signal_name];
        console.log(db_signal.name + ": " + db_signal.value);
    }

    //回发数据
    var msg_name     = message_info.name;
    var msg_EMCY_118 = db.messages[msg_name];
    //设定回发数据的数据值内容
    msg_EMCY_118.signals["ManufacturerSpecific2"].update(0);
    msg_EMCY_118.signals["ManufacturerSpecific3"].update(0);
    msg_EMCY_118.signals["ManufacturerSpecific4"].update(500.234);
    //回发数据
    db.send(msg_name);
}

channel.addListener("onMessage", dumpPacket);
```

实现打印并回发消息的函数 dumpPacket 并挂载到通道 channel 的消息处理事件上。

```
channel.start();
```

监听 can0 上接受的消息。

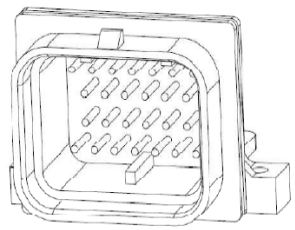
附录 1 技术指标

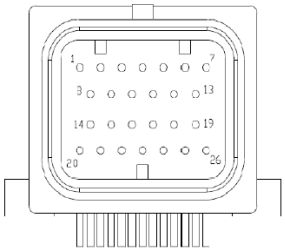
定位芯片	GPS L1、BeiDou B1,167db，定位精度≤2.5m，天线馈电 3.2V/100mA
4G	标准尺寸 SIM 卡，支持全网通，上行 50Mbps，下行 150Mbps
WiFi	802.11 b/g/n 标准，2.412GHz-2.484GHz 频率范围
蓝牙	蓝牙 4.0，2.412GHz-2.484GHz 频率范围
以太网口	两个 10/100Mbps 自适应，DF50A 接口
CAN	两路，支持 CAN2.0A 和 2.0B，波特率 50 Kbits/s...1 Mbit/s
RS485	一路，波特率 1200~115200bps（默认 9600bps）
IO 口	4 路数字 IO 输入和 2 路数字 IO 输出
温度传感器	一个温度传感器，测量范围-40℃-85℃
陀螺仪	一个 16bit 陀螺仪
锂电池充电座	一个充电插座，XH2.54-2A 接口
TF 卡座	一个 TF 卡插槽，最大支持 128G(exFAT)
ADC	两路模拟输入，采集 0-27V 电压
音频输出	一个音频输出，可驱动3W/4R或者3W/8R的扬声器
MIC输入	一个MIC输入口
电源输入	DC：+9V ～ +36V
工作温度	-30℃～+70℃
存储湿度	-40℃～+85℃
功耗(正常模式)	≤6W（不接负载，不含充电功耗，充电功耗视电池电量而定）
功耗(主电模式)	≤1W（不接负载，不含充电功耗，充电功耗视电池电量而定）
功耗(备电模式)	≤8mW
尺寸（mm）	153 x 96mm
重量	0.15kg

附录 2 26pin 引脚座线序编号图

安普标准引脚定义

位置	名称	功能定义		位置	名称	功能定义
1	输入3(IO3)	1.输入		14	CAN1_GND	CAN1总线地
2	输入2(IO2)	1.输入		15	CAN1_L	CAN1总线
3	输入1(IO1)	1.输入		16	CAN1_H	CAN1总线
4	电源地（GND）	电源地		17	RS485_GND	RS485总线地
5	输出2	继电器驱动输出2，默认高阻		18	RS485_B	RS485总线
6	输出1	继电器驱动输出1，默认高阻		19	RS485_A	RS485总线
7	模拟信号采样地	模拟信号采样地,与数字地隔离		20	CAN2_GND	CAN2总线地
8	音频功放输出（AUDIO_OUT）	语音播报音频输出，驱动3W喇叭		21	CAN2_L	CAN2总线
9	音频功放输出地	模拟信号地，与数字地隔离		22	CAN2_H	CAN2总线
10	MIC信号地（MIC_GND）	MIC信号地，与数字地隔离		23	输入（IO4）	1.输入
11	MIC输入（MIC_IN）	录音和通话功能		24	ACC电源（ACC）	ACC电源，车钥匙控制
12	模拟信号采集输入2（ADC2）	直流电压或电流信号采样		25	电源地（GND）	车载蓄电池负极。
13	模拟信号采集输入1（ADC1）	直流电压或电流信号采样		26	电源正极	车载蓄电池电源正极

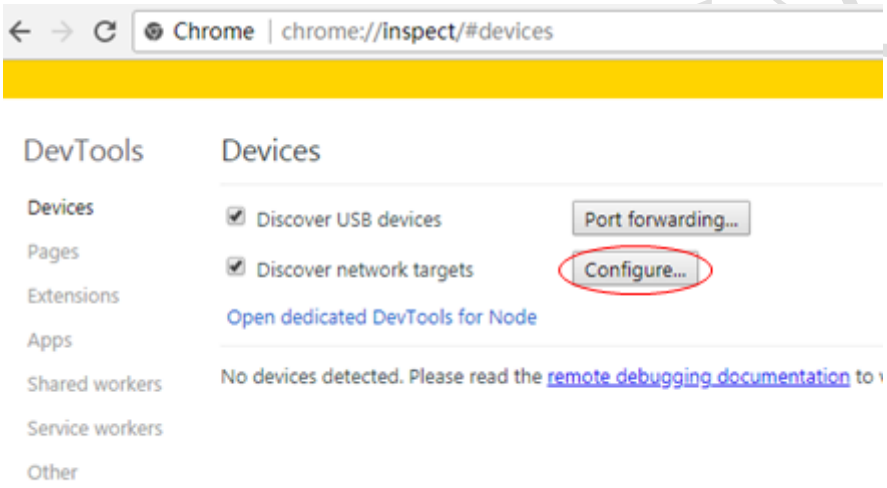




iovBOX 采用 26PIN 安普标准引脚座供电，26PIN 引脚座 24、25、26 号引脚为电源输入脚（24 和 26 号引脚可接在一起），其中供电电压为 9~36V 宽电压供电范围，典型值为 12V 和 24V。

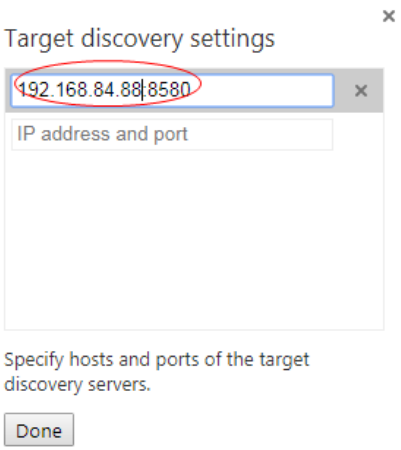
附录 3 Chrome 远程调试设置方法

- 打开 Chrome，确认版本号（高于 64.0.xxxx.xxx）
- 地址栏打开 Chrome://inspect/#devices,显示如下界面，点击 Configure...



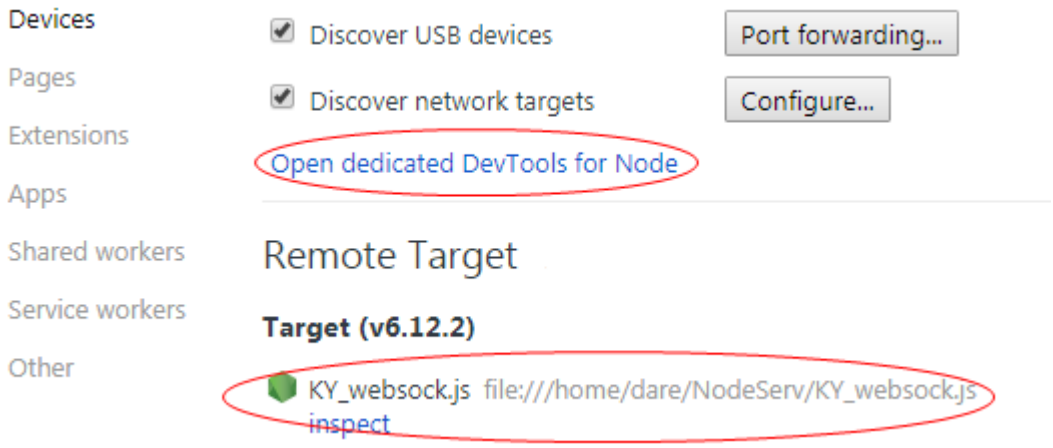
图表 19 Chrome 配置

- 在弹出的配置框中，填写 iovBOX 的 ip，以及 inspect 输出的端口地址，点击 Done。



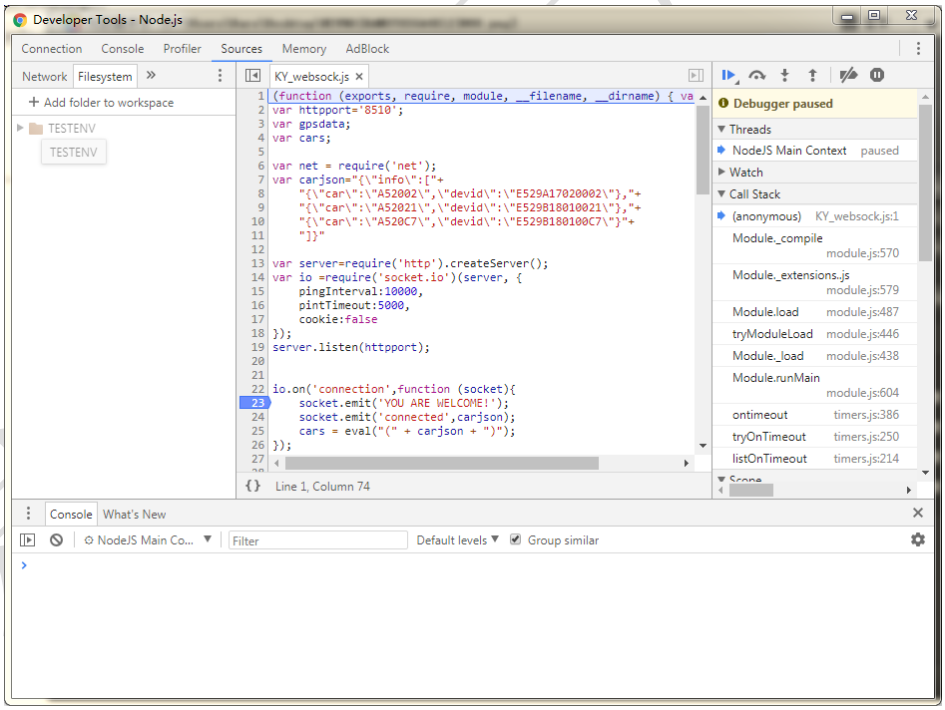
图表 20 Chrome 端口配置

- 完成后等待几秒钟链接，会显示如下提示(*)，然后点击 Open dedicated DevTools for Node.(**)



图表 21 打开 Chrome 调试工具

- 进入到如下页面，然后就可以开始单步调试了。



图表 22 Chrome 调试界面

- 注意事项：

(*):如果没有链接成功，请检查 PC 或者 iovBOX 的防火墙设置，是否关闭了相应端口的使用。

(**):针对 nodejs，不使用 inspect 按钮。

附录 4 iovBOX 服务开发接口（略）

附录 5 CAN_BABEL 使用方式（略）

附录 6 CANPro 安装方法（略）

重要申明

易通星云（北京）科技发展有限公司为客户提供全方位的技术支持，用户可与就近的易通办事处、分公司、代理商联系，也可直接与公司总部联系。

易通星云（北京）科技发展有限公司
北京市朝阳区霄云路 3 号中科霄云大厦 309 室
公司总机：　　010-64612620
网　　址：　　www.beidouapp.com
客户服务邮箱：support@beidouapp.com
官方支持 QQ 群：445880047 （开发快-智能硬件开发）

注意

本手册描述的产品及其附件的某些特性和功能，取决于当地网络的设计和性能，以及您安装的软件。某些特性和功能可能由于当地网络运营商或网络服务供应商不支持，或者由于当地网络的设置，或者您安装的软件不支持而无法实现。因此，本手册中的描述可能与您购买的产品或其附件并非完全一一对应。
易通星云（北京）科技发展有限公司保留随时修改本手册中任何信息的权利，无需进行任何提前通知且不承担任何责任。

无担保声明

本手册中的内容均“如是”提供，除非适用法律要求，易通星云（北京）科技发展有限公司对本手册中的所有内容不提供任何明示或暗示的保证，包括但不限于适销性或者适用于某一特定目的的保证。
在法律允许的范围内，易通星云（北京）科技发展有限公司在任何情况下，都不对因使用本手册相关内容而产生的任何特殊的、附带的、间接的、继发性的损害进行赔偿，也不对任何利润、数据、商誉或预期节约的损失进行赔偿。