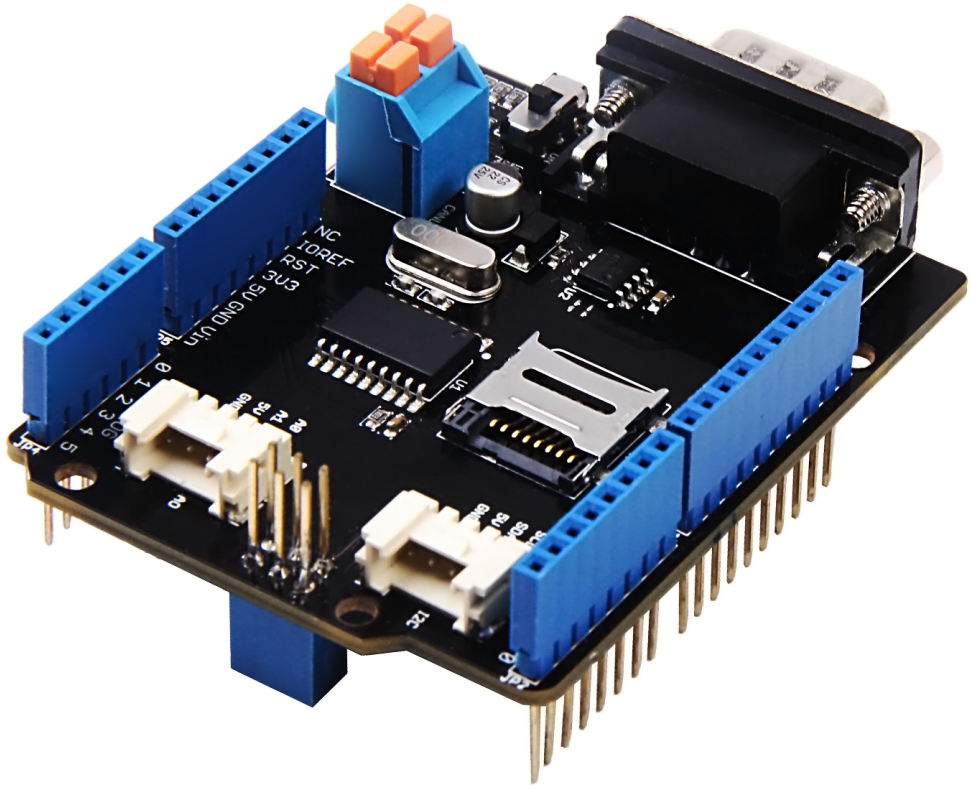
CAN-BUS Shield V2.0SKU: 103030215



CAN-BUS 是一种普通的工业总线，由于其长距离，中等通信速度和高可靠性。它通常在现代机床上应用，例如汽车诊断总线。

该CAN总线扩展板采用MCP2515 CAN总线控制器与SPI接口和MCP2551 CAN收发器，为您提供Arduino / Seeeduino CAN-BUS函数。通过添加OBD-II转换器电缆并导入OBD-II库，就可以构建板载诊断设备或数据记录器。

之前，我们制作了两个版本的CAN-BUS Shield，即V1.0和V1.2。 它们都广受我们用户的好评。 为了做得更好，几个月前，我们进行了关于CAN-BUS Shield V1.2的调研，收到了许多有价值的建议（感谢所有反馈给我们的用户）。我们十分重视这些用户的意见，我们决定进行更新。于是，新一代产品就此孕育而生 —— **CAN-BUS Shield V2**

版本

本文档适用于以下版本的产品：：

版本	发布日期	购买链接	
CAN BUS Shield V1.0	2012 11月 14号	E.O.L	不再销售
CAN BUS Shield V1.1	2013 8月10号	E.O.L	不再销售

版本	发布日期	购买链接
CAN BUS Shield V1.2	2015 2月5号	Get One Now 点击购买
CAN BUS Shield V2.0	2017 8月1号	Get One Now 点击购买

版本对比

特性	**V1.2**	**V2.0**
CAN-BUS 控制芯片	MCP2515	MCP2515
CAN 收发芯片	MCP2551	MCP2551
默认 OBD 引脚接口	标准 OBD-II	标准 OBD-II
标准 CAN 引脚接口	不兼容	兼容 (跳线)
INT Pin	不可改变	D2 或者 D3 (跳线)
TF card 套的CS引脚	没有 TF card 卡套	D4 或者 D5 (跳线)
Grove 串行接口	D0/D1	A0/A1
I2C Grove	A4/A5	SDA/SCL
Grove 接口方向	垂直	水平
P1 焊盘	扩展板正面	扩展板反面

CAN BUS Shield V2.0 的新特性 ###硬件

1. 原本DB9接口是OBD标准，通过增加跳线，兼容CAN OPEN标准
2. INT通过一个跳线选择D2或者D3
3. 增加SD卡座子，CS默认接到D4，可以通过跳线到D5（使用stalker的sd座子）
4. Serial的Grove座子改到A0/A1 （考虑到D0/D1通常需要用来下载代码，没有办法接其他模块，A0/A1既可以当做模拟口，也可以当做数字口，使用起来更加方便）
5. I2C的连线不再接到A4/A5，改到标准引脚的SDA/SCL
6. 两个Grove座子躺着，ICSP两边各一个
7. P1放到板子底下，方便切割和焊接
8. 其他基于电子/成本方面的优化

###软件

1. 增加读取汽车数据的函数及examples
2. 增加SD卡的读取
3. 其他bug修复以及优化
4. 增加把汽车数据存储到SD卡的example

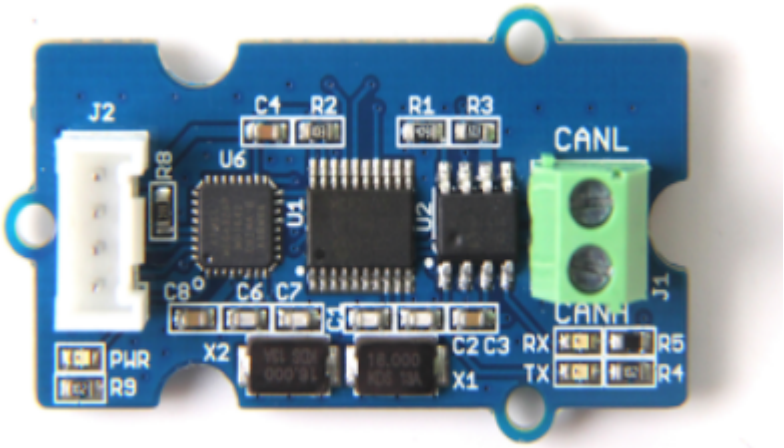
D-Sub CANbus 引脚接口

pin#	信号名称	信号描述

pin#	信号名称	信号描述
1	Reserved	Upgrade Path
2	CAN_L	Dominant Low
3	CAN_GND	地
4	Reserved	Upgrade Path
5	CAN_SHLD	扩展板, 可选
6	GND	地，可选
7	CAN_H	Dominant High
8	Reserved	Upgrade Path
9	CAN_V+	Power, 可选

替代方案

如果您的项目是空间有限的，除了CAN-BUS 之外，不需要其他功能，这里是一个Grove CAN-BUS模块，与Arduino兼容，更紧凑，性价比更高，参考请点击 这里 。



如果我想将这个扩展板连接到我的车，该怎么办？

如果您想要读取数据或控制汽车，则可以使用OBD> DB9电缆，该电缆将轻松完成OBD连接器到DB9连接器的转换。该电缆也可以连接到具有OBD连接器的任何东西。我们在电缆上添加了一个开关，方便您的控制。您可以点击下面的图标购买该电缆。

USB-CAN分析仪

如果您需要一个CAN总线分析仪来检测您的CAN总线，我们推荐下面的 USB-CAN Analyzer 。(同样您可以点击下面的图片购买)

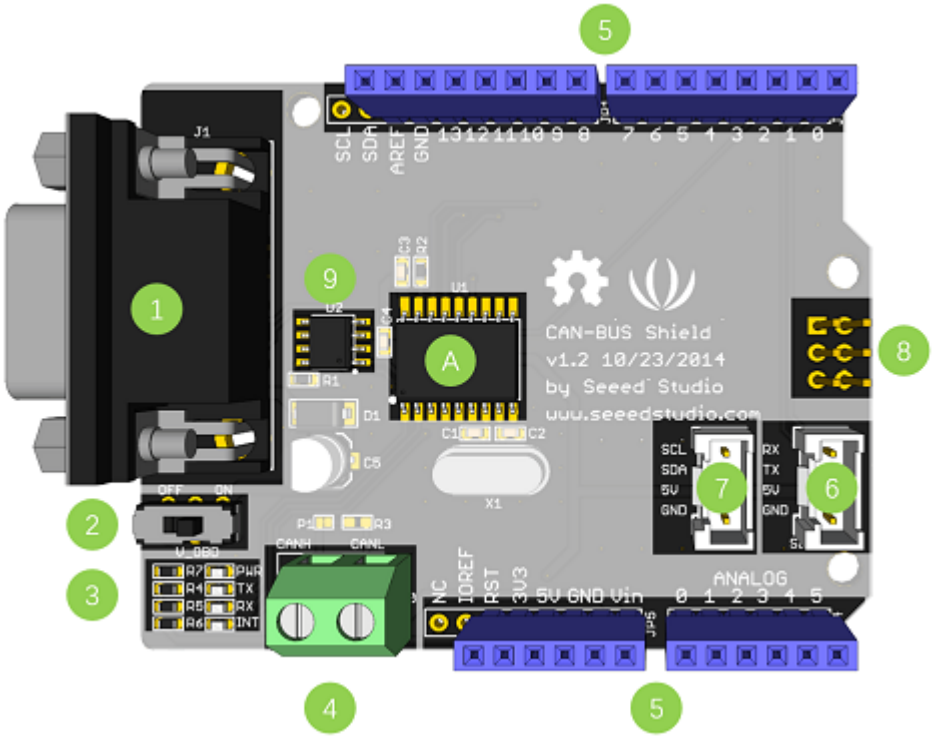
产品特性

- CAN V2.0 速度高达1 Mb / s
- SPI接口速度高达10 MHz
- 支持标准（11位）和扩展（29位）数据以及远程帧
- 两个具有优先消息存储的接收缓冲区
- 工业标准DB-9连接器
- OBD-II和标准CAN引脚兼容可选
- 可选TF card CS引脚
- 可选择INT引脚
- 新型按压接线端子易于连接CAN_H 和 CAN_L
- 两个Grove连接器（I2C和UART）

- LED指示灯

!!!Note CAN BUS 可以良好地兼容Arduino UNO (ATmega328), Arduino Mega (ATmega1280/2560) 、 Arduino Leonardo (ATmega32U4).

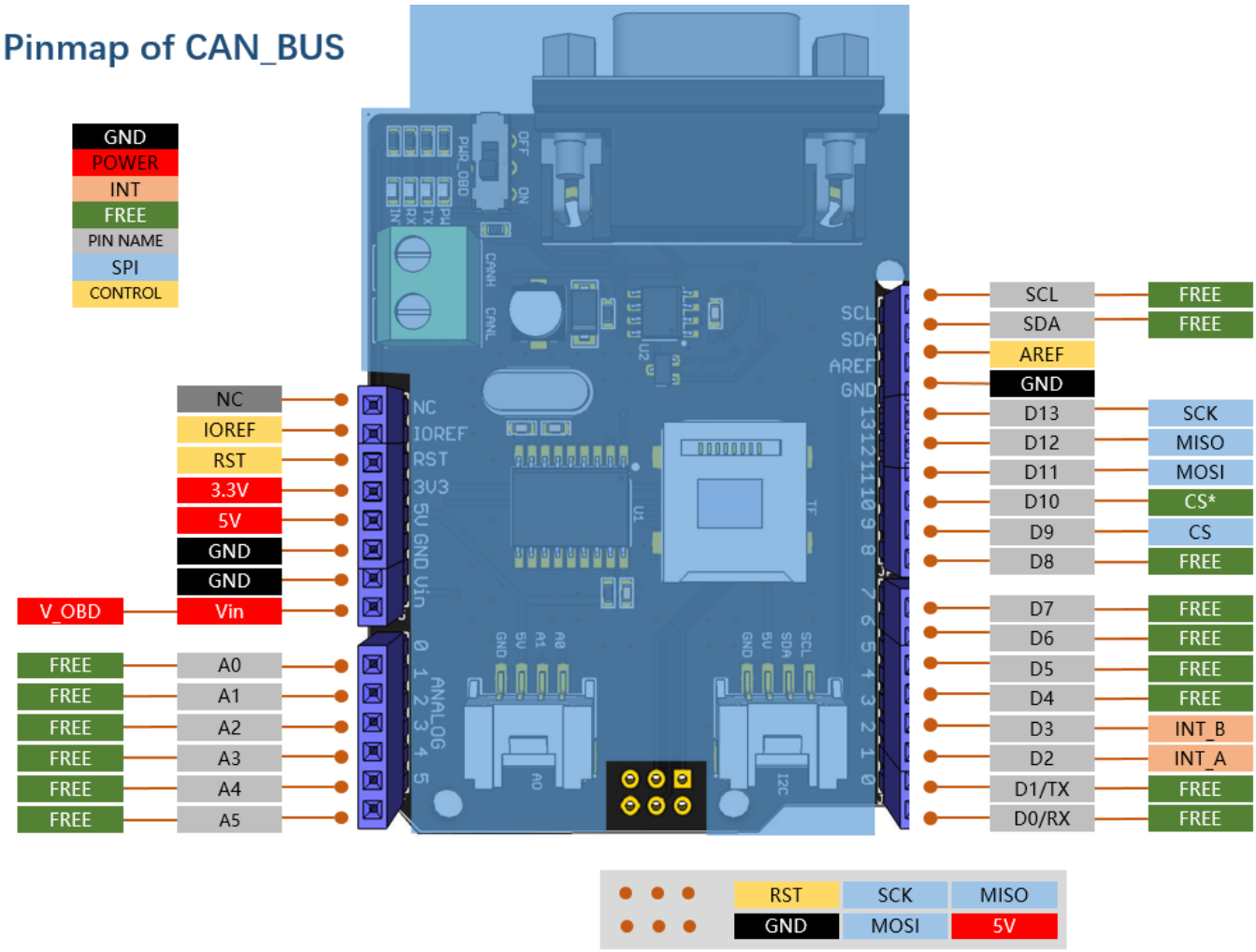
硬件概述



1. **DB9** 接口 - 通过DBG-OBD线缆来连接OBDII 接口。
2. **V_OBD** - 通过OBDII 接口获取电源 (来着 DB9)
3. **Led** 指示器:
    - **PWR**: 电源
    - **TX**: 发送数据时闪烁
    - **RX**: 接收数据时闪烁
    - **INT**: 数据中断
4. 接线端子 - CAN_H 和 CAN_L
5. **Arduino UNO** 引脚接口
6. **Grove** 串行连接器
7. **Grove I2C连接器 **
8. **ICSP** 引脚
9. 芯片 - MCP2551, 高速 CAN 收发器 (datasheet)
10. 芯片 - MCP2515, 带有 SPI 接口的标准 CAN 总线控制器 (datasheet)
11. SD卡卡座

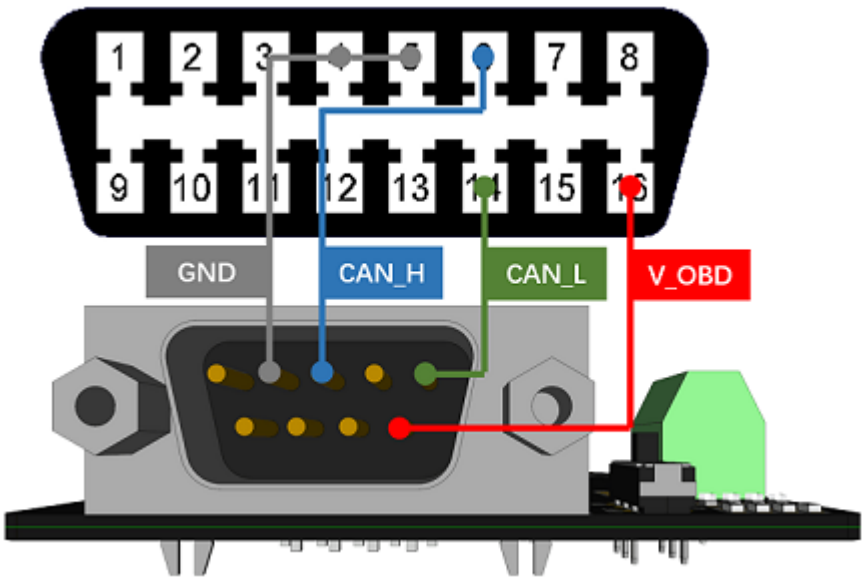
!!!warning - 当您在一个网络中使用两个以上的CAN总线扩展板时，应考虑阻抗。您应该用刀切割PCB中的P1，或者干脆取下PCB上的R3。

管脚图



!!!note - 空闲针可用于其他用途。

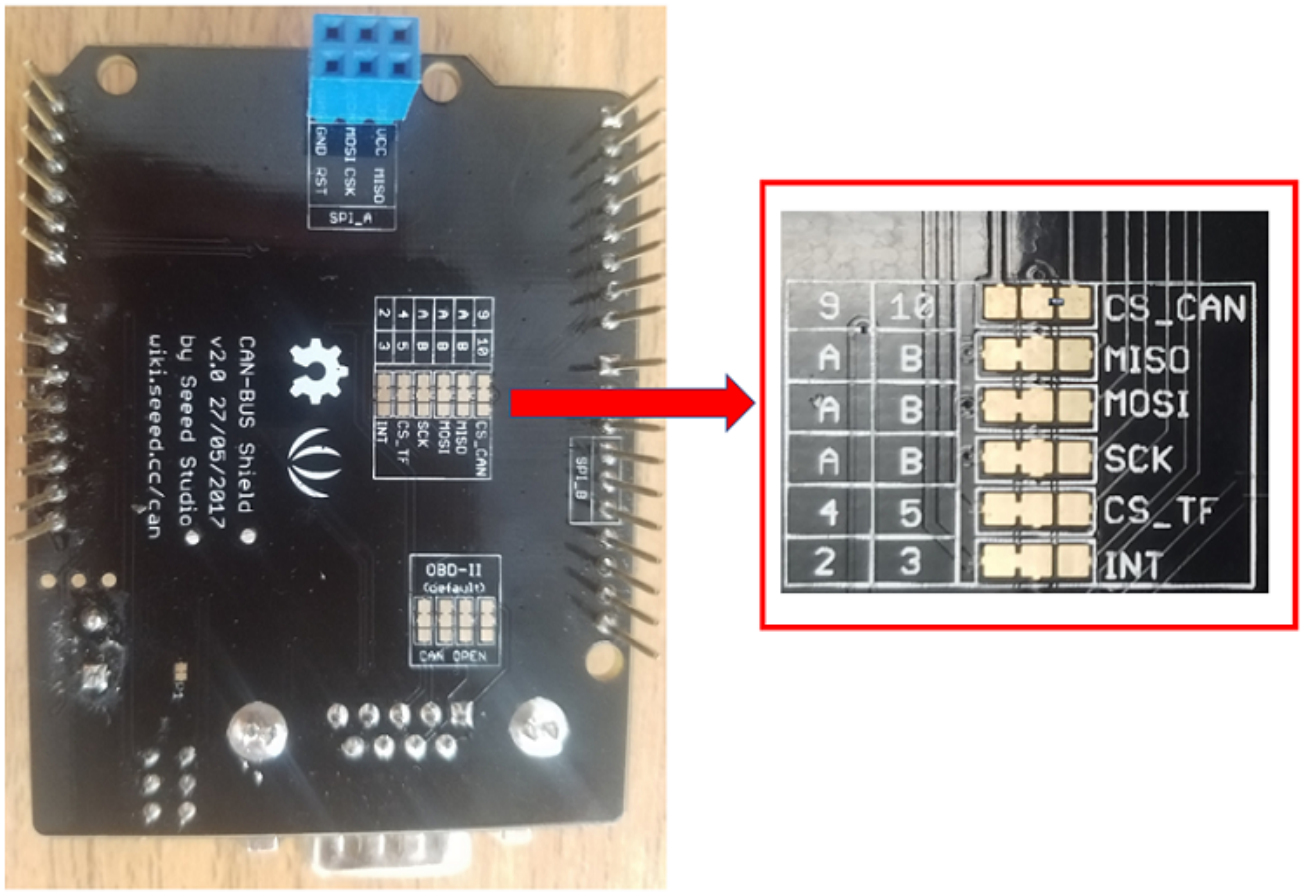
DB9&OBDii 接口



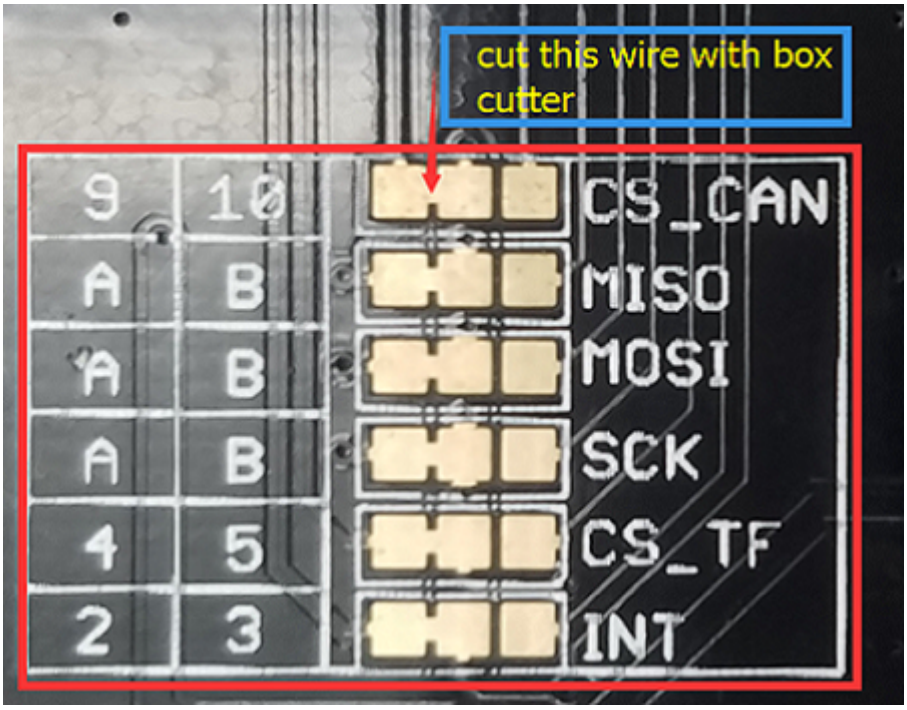
CS_CAN 引脚

默认情况下，V2.0的SPI_CS引脚连接到 **D9**。如果要更改为 **D10** ，请按照以下说明进行操作。

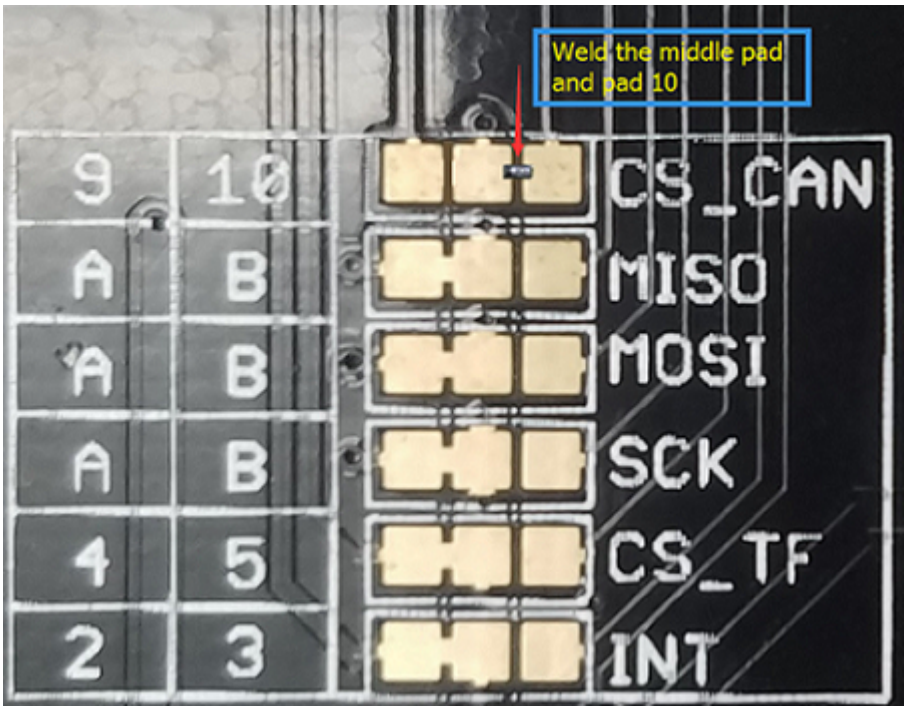
- 步骤1：查看PCB的背面，你会发现一个名为 CS_CAN 的焊盘。



- 步骤2: 割断焊盘9和中间焊盘之间的导线



- 步骤3: 将中间焊盘和焊盘10焊接在一起。

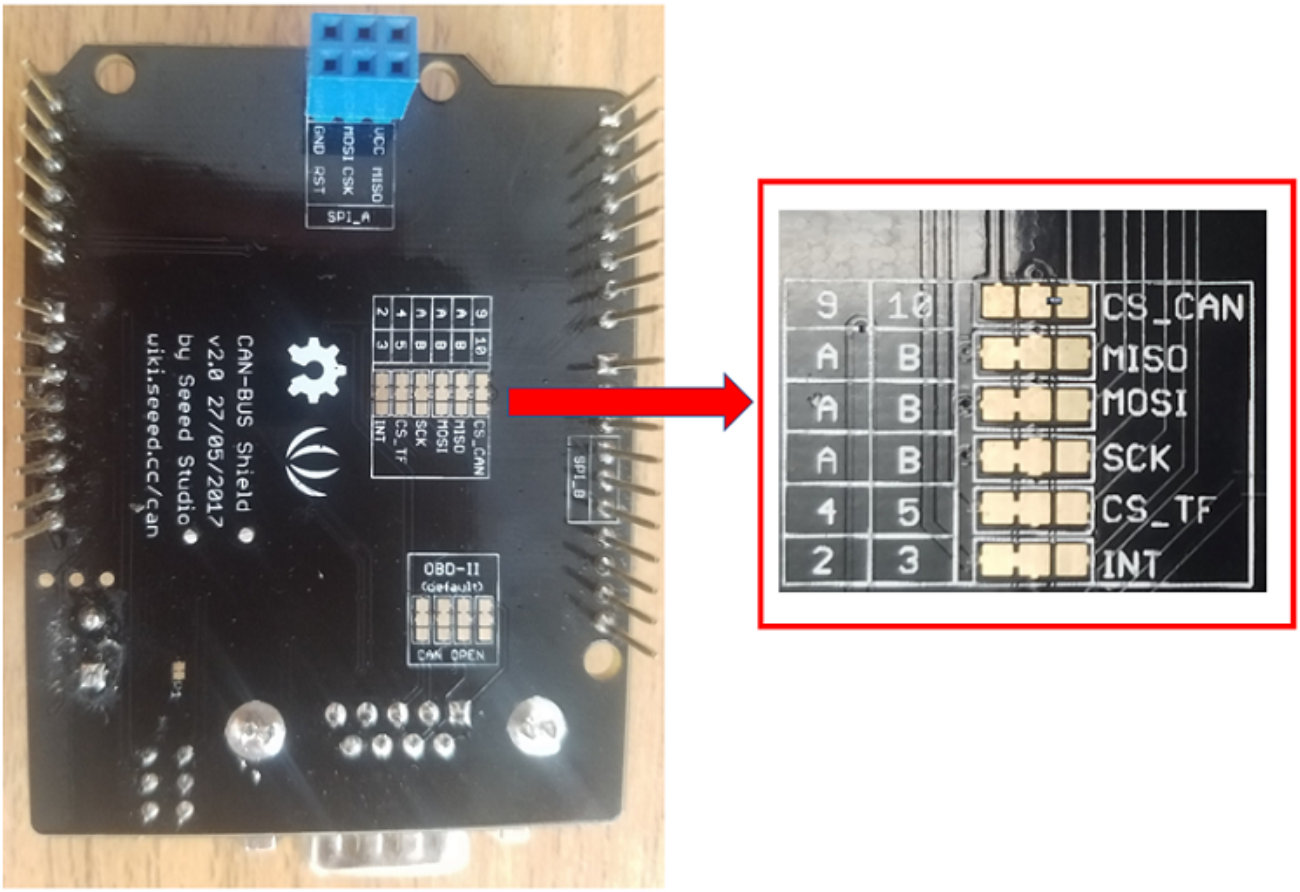


!!!warning - 切割的时候请您小心，不要伤到自己或者割坏PCB板子。

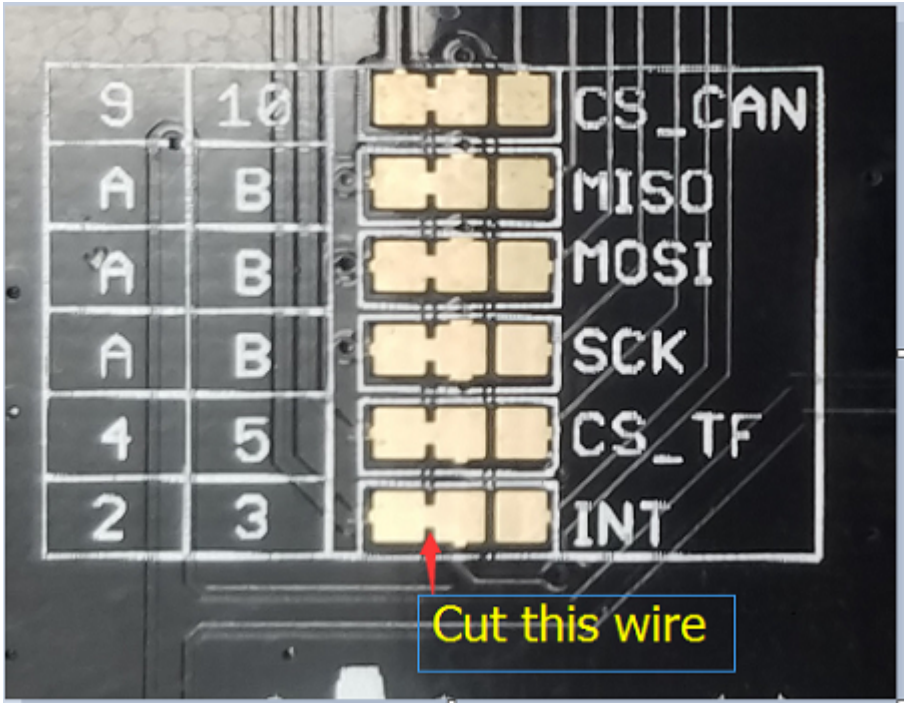
INT pin 引脚

默认情况下，V2.0的INT引脚连接到**D2**。如果要更改为**D3**，请按照以下说明进行操作。

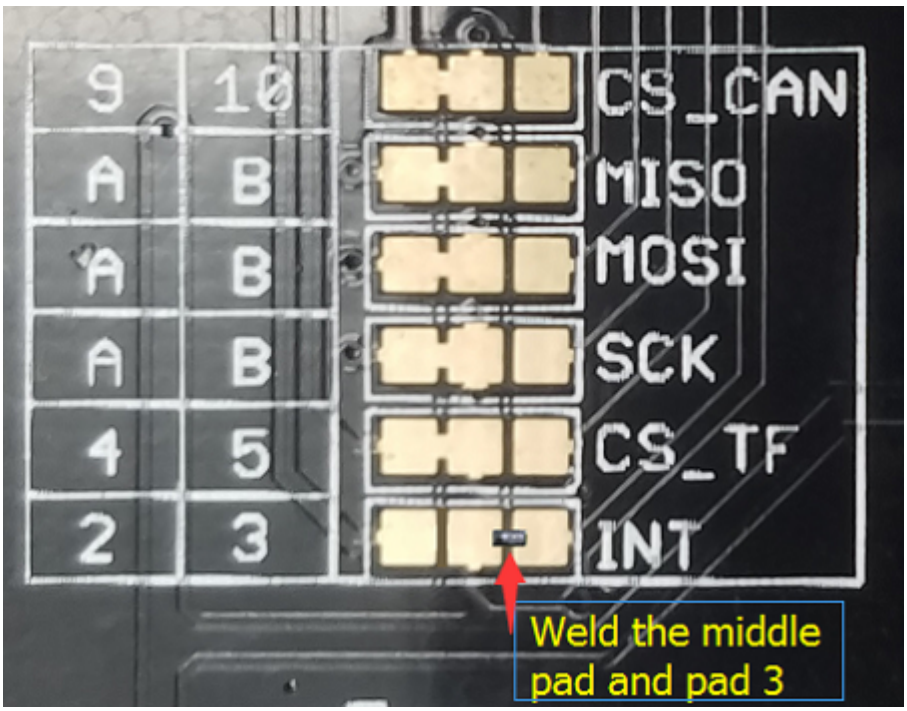
- 步骤1：查看PCB的背面，你会发现一个名为 `INT` 的焊盘。



- 步骤2：割断焊盘2和中间焊盘之间的导线



- 步骤3：将中间焊盘和焊盘3焊接在一起。



SPI 引脚

默认情况下，SPI引脚（SCK，MISO，MOSI）被连接到ICSP引脚。但是对于某些核心板，SPI引脚位于D11～D13。如果发生这种情况，您需要对PCB进行一些更改。看看PCB的背面，有三个焊盘，MOSI，MISO和SCK，它们默认连接到A。如果需要，您可以将它们更改为B。

!!!note - 对于Arduino UNO, Arduino Mega, Arduino Leonardo 以及任何其他基于AVR 的Arduino 核心板, 我们推荐初始设置。

!!!warning - 切割的时候请您小心，不要伤到自己或者割坏PCB板子。

入门指南

以下是一个简单的例子来说明CAN-BUS Shield的工作原理。在这个例子中，我们需要2块CAN-BUS扩展板以及Arduino或Seeeduino。

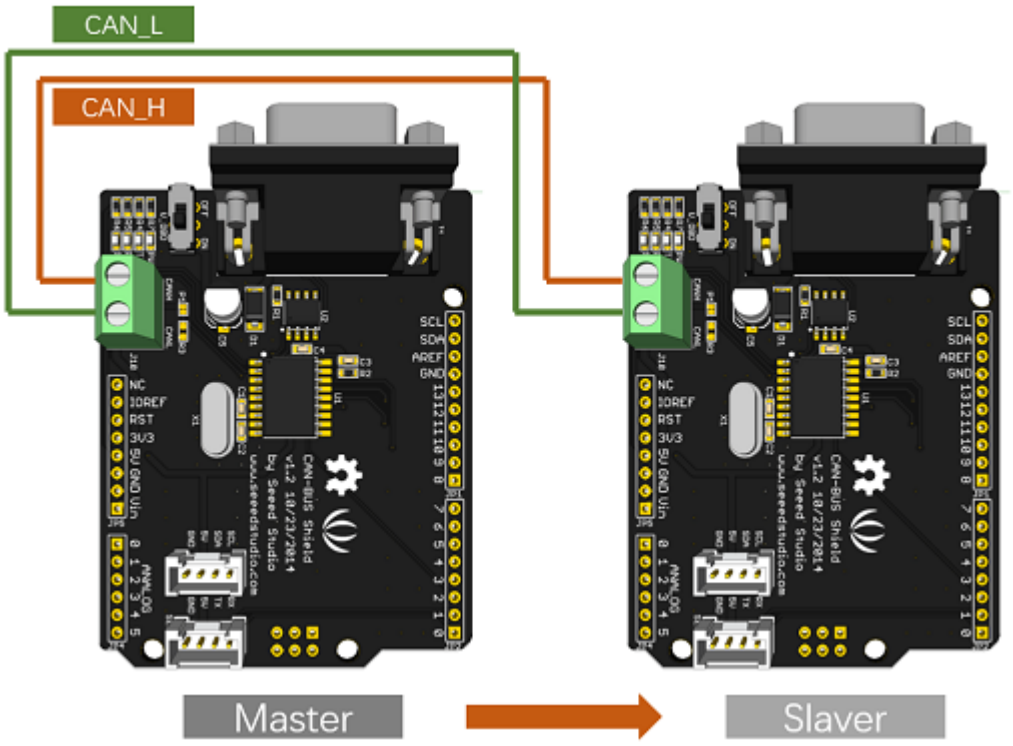
!!!note 此示例基于 Arduino IDE version 1.6.9.

步骤1: 我们需要准备些啥

名称	功能	数量	购买链接
CAN-BUS Shield	CAN 总线通信	2	购买链接
Seeeduino V4.2	控制器	2	购买链接
跳线	连接	2	购买链接

步骤2: 硬件连接

将每块CAN-BUS 扩展板都插在Seeeduino V4.2上，然后如下图所示使用两根跳线将两块CAN-BUS 扩展板连接起来。



!!!note - CAN_H 连接到 CAN_H, CAN_L 连接到 CAN_L

步骤3: 软件

请遵循 如何安装Arduino库 来安装 CAN BUS shield library.

点击下面图标来下载CAN BUS shield library

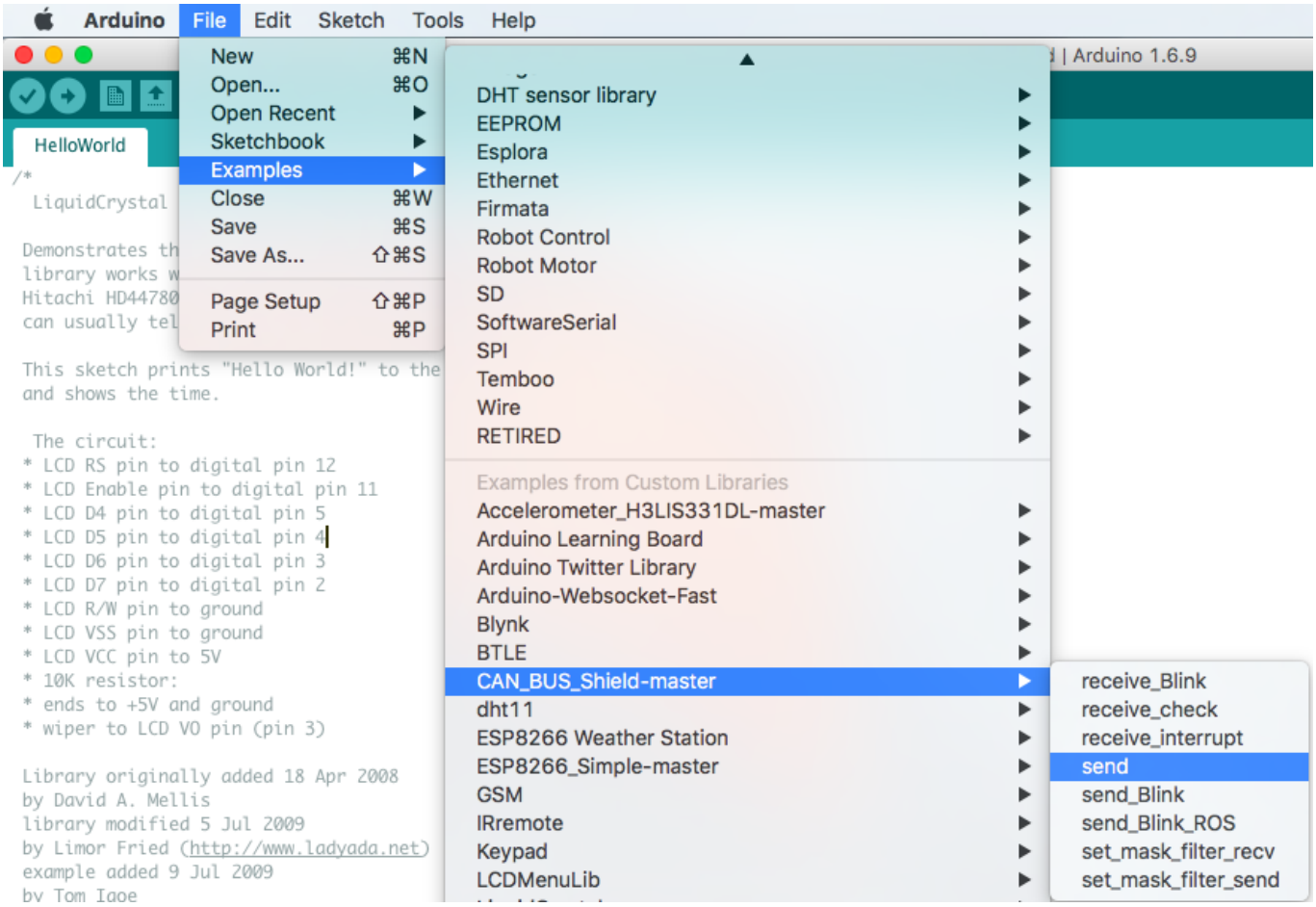
Download CAN BUS Shield Library

下载完成后安装库文件到您的Arduino。

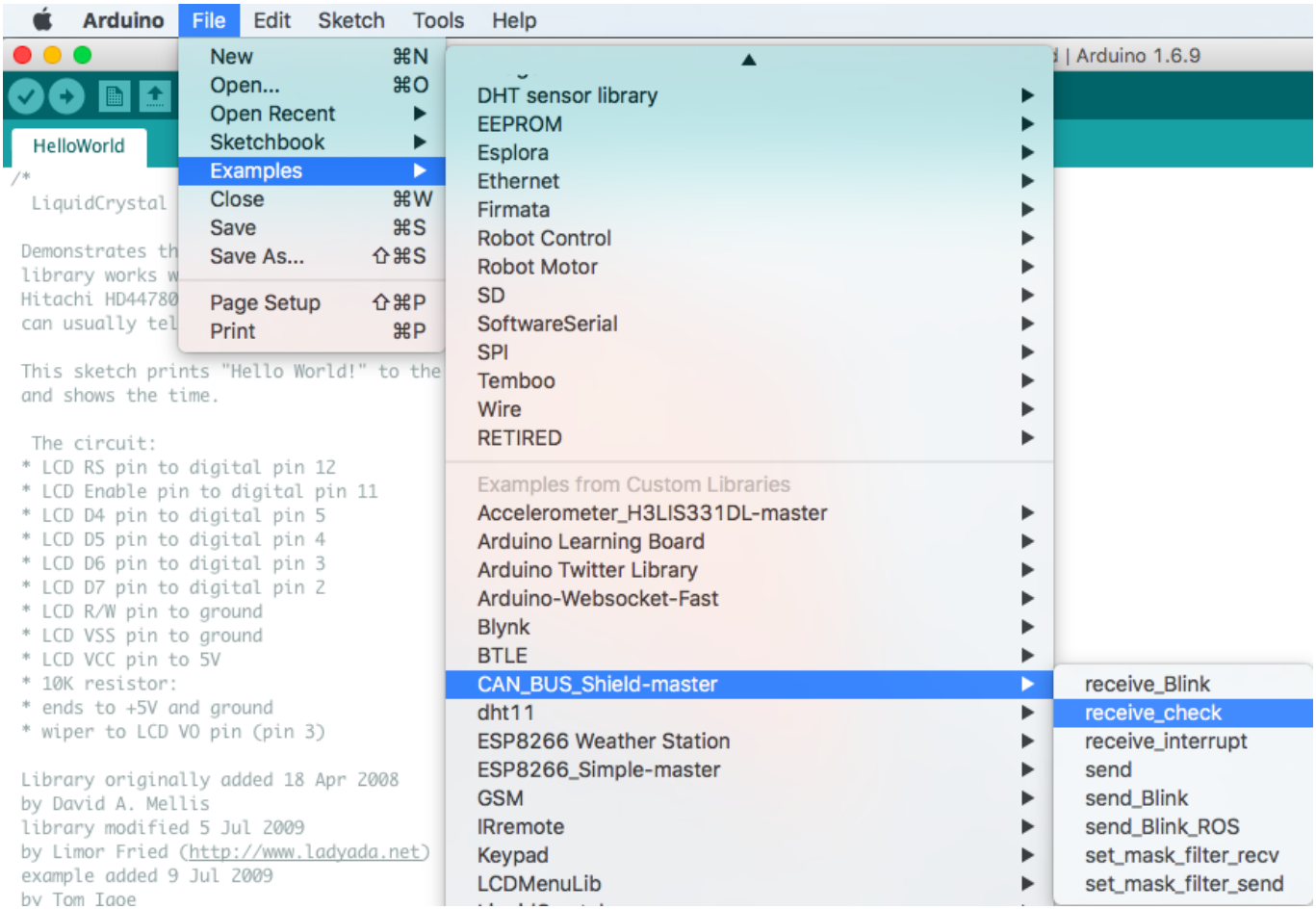
一个节点(此处代表Seeeduino + CAN_BUS 扩展板) 作为主机，另外一个节点作为从机。 主机将持续地向从机发送数据。

!!!note 在上传代码前，每一个节点都可以作为主机。

打开发送示例（文件>示例> CAN_BUS_Shield-master>发送）并上传到主机。

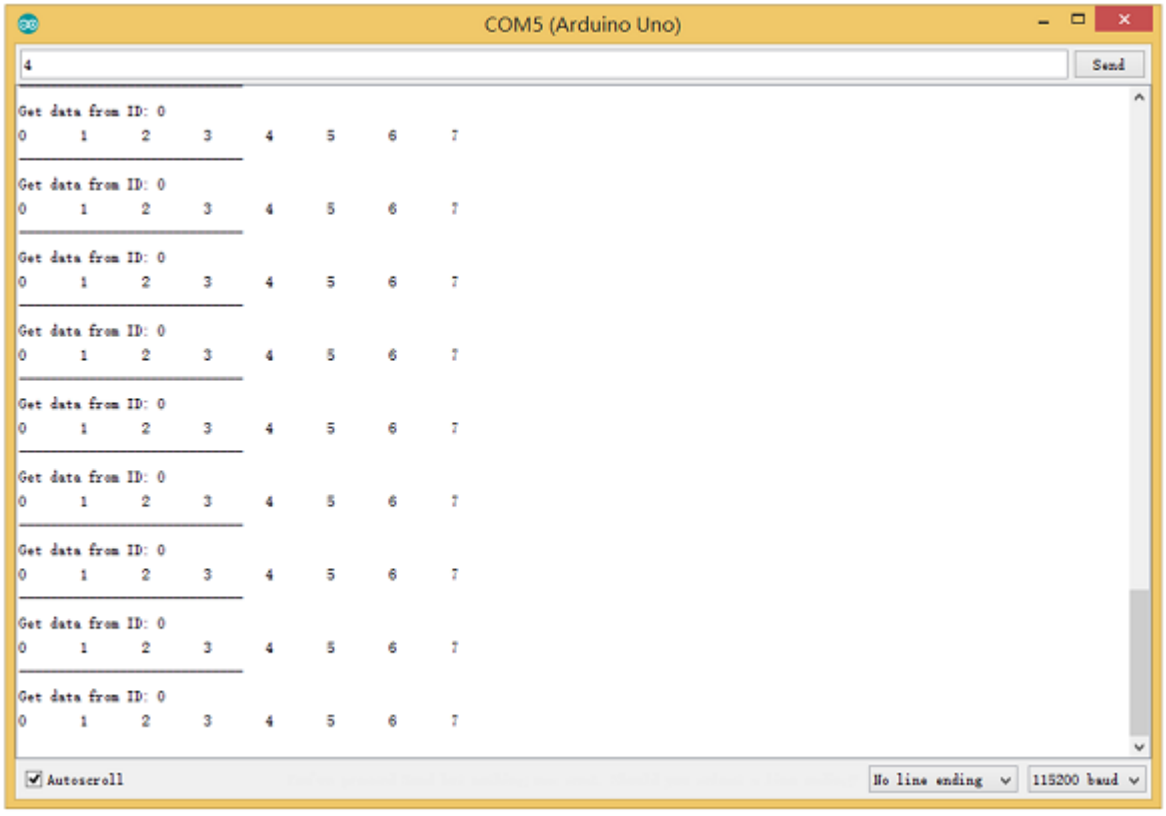


打开receive_check示例（文件>示例> CAN_BUS_Shield-master> receive_check）并上传到从机。



步骤**4:** 查看结果

打开Arduino IDE（从机）的串行监视器，您将获得从主机发送的数据。



APIs

1. 设置波特率

该函数用于初始化CAN总线系统的波特率。

可用的波特率列表如下：

```
#define CAN_5KBPS    1
#define CAN_10KBPS   2
#define CAN_20KBPS   3
#define CAN_25KBPS   4
#define CAN_31K25BPS 5
#define CAN_33KBPS   6
#define CAN_40KBPS   7
#define CAN_50KBPS   8
#define CAN_80KBPS   9
#define CAN_83K3BPS  10
#define CAN_95KBPS   11
#define CAN_100KBPS  12
#define CAN_125KBPS  13
#define CAN_200KBPS  14
#define CAN_250KBPS  15
#define CAN_500KBPS  16
#define CAN_666kbps  17
#define CAN_1000KBPS 18
```

2. 设置接收屏蔽寄存器和过滤寄存器

控制器芯片上有2个接收屏蔽寄存器和5个滤波器寄存器，用于保证从目标设备获取数据。它们特别适用于由许多节点组成的大型网络。

我们提供两种函数来利用这些屏蔽寄存器和滤波器寄存器。他们是：

Mask:

```
init_Mask(unsigned char num, unsigned char ext, unsigned char ulData);
```

Filter:

```
init_Filt(unsigned char num, unsigned char ext, unsigned char ulData);
```

- **num** num表示要使用哪个寄存器。您可以设置0或1选择屏蔽寄存器，0至5选择过滤寄存器。
- **ext** 表示帧的状态。0表示它是标准帧。1表示它是一个扩展帧。
- **ulData** 代表屏蔽帧或过滤帧的内容。

###3. 校验接收 MCP2515可以在轮询模式下工作，其中软件检查接收到的帧，或使用附加引脚来发信号通知帧已被接收或发送完成。

使用以下函数轮询接收到的帧。

```
INT8U MCP_CAN::checkReceive(void);
```

如果帧到达，该函数将返回1，如果没有到达，则返回0。

###4. 获取 CAN ID 当某些数据到达时，您可以使用以下函数获取“发送”节点的CAN ID。

```
INT32U MCP_CAN::getCanId(void)
```

###5. 发送数据

```
CAN.sendMsgBuf(INT8U id, INT8U ext, INT8U len, data_buf);
```

这是将数据发送到总线上的函数。其中：

- **id** 表示数据来源
- **ext** 表示帧的状态。'0'表示标准帧。'1'表示扩展帧。
- **len** 表示此帧的长度。
- **data_buf** 是此消息的内容。

例如，在“发送”示例中，我们有：

```
unsigned char stmp[8] = {0, 1, 2, 3, 4, 5, 6, 7};
CAN.sendMsgBuf(0x00, 0, 8, stmp); //send out the message 'stmp' to the bus and
tell other devices this is a standard frame from 0x00.
```

###6. 接收数据

以下函数用于在“接收”节点上接收数据：

```
CAN.readMsgBuf(unsigned char len, unsigned char buf);
```

在屏蔽器和过滤器都设置好的条件下，这个函数只会接收满足屏蔽器和过滤器筛选条件的数据帧。

- **len** 表示数据长度。
- **buf** 是存储数据的位置。

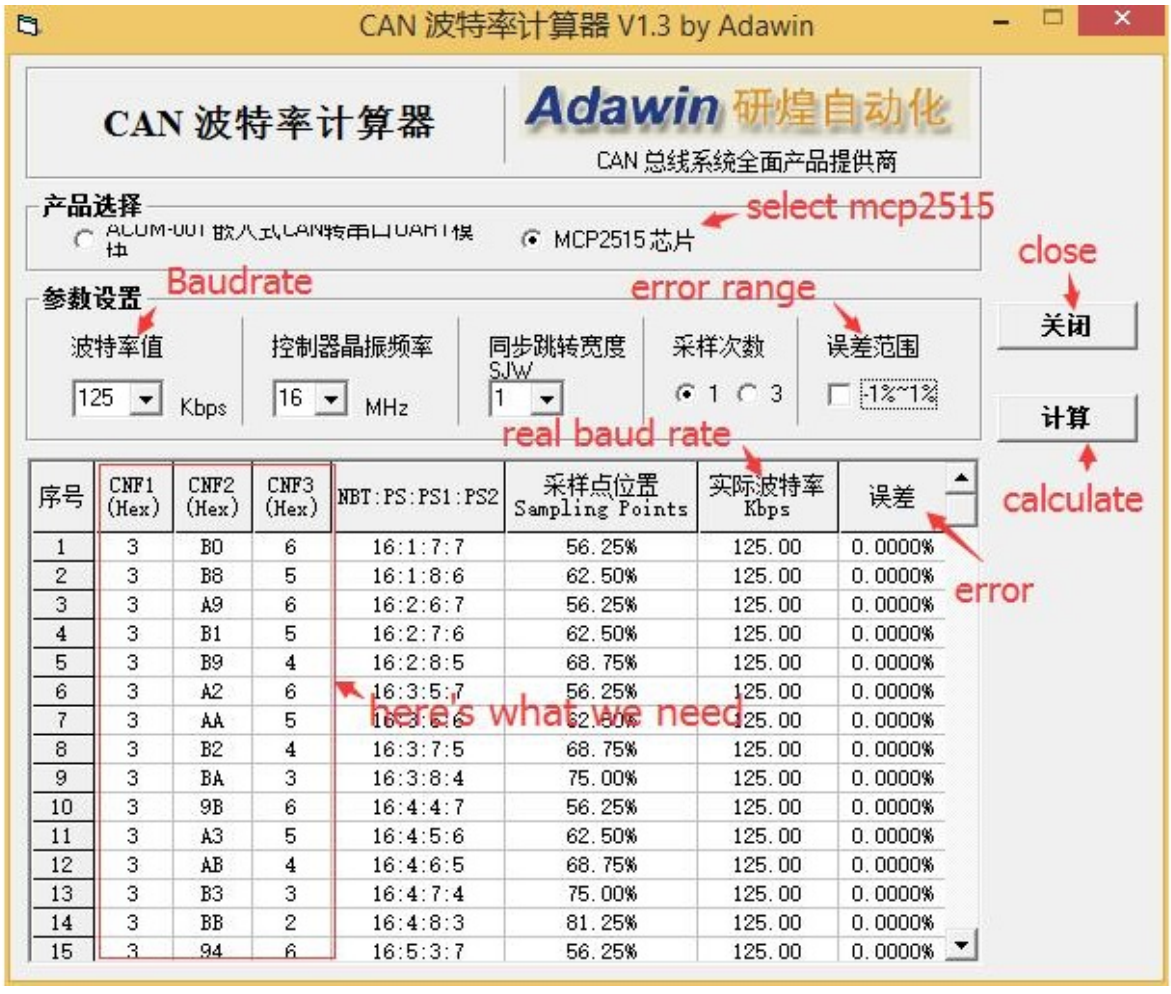
生成一个新的波特率

我们提供了许多频繁使用的波特率，如下所示：

```
#define CAN_5KBPS    1
#define CAN_10KBPS   2
#define CAN_20KBPS   3
#define CAN_25KBPS   4
#define CAN_31KBPS   5
#define CAN_33KBPS   6
#define CAN_40KBPS   7
#define CAN_50KBPS   8
#define CAN_80KBPS   9
#define CAN_83KBPS   10
#define CAN_95KBPS   11
#define CAN_100KBPS  12
#define CAN_125KBPS  13
#define CAN_200KBPS  14
#define CAN_250KBPS  15
#define CAN_500KBPS  16
#define CAN_666KBPS  17
#define CAN_1000KBPS 18
```

然而，您仍然可能找不到所需的波特率。在这里，我们提供一个软件来帮助您计算所需的波特率。

点击 这里 来下载该软件。



!!!note 这个软件仅仅支持Windows系统，如果您无法打开它，欢迎随时通过邮件联系 loovee@seeed.cc 来获取技术支持。

打开软件，你需要做的就是设置你想要的波特率，然后做一些简单的设置，然后点击计算。

然后，您将获得一些数据，cfg1，cfg2和cfg3。

您需要在库中添加一些代码。

打开 **mcp_can_dfs.h**, 你需要将下列代码添加到第272行左右。

```
#define MCP_16MHz_xxxkBPS_CFG1 (cfg1)    // xxx is the baud rate you need
#define MCP_16MHz_xxxkBPS_CFG2 (cfg2)
#define MCP_16MHz_xxxkBPS_CFG3 (cfg2)
```

然后让我们跳转到第390行, 添加如下代码：

```
#define CAN_xxxKBPS NUM       // xxx is the baudrate you need, and NUM is a
number, you need to get a different from the other rates.
```

打开 **mcp_can.cpp**, 跳转到函数**mcp2515_configRate**(大致在第190行), 然后添加如下代码:

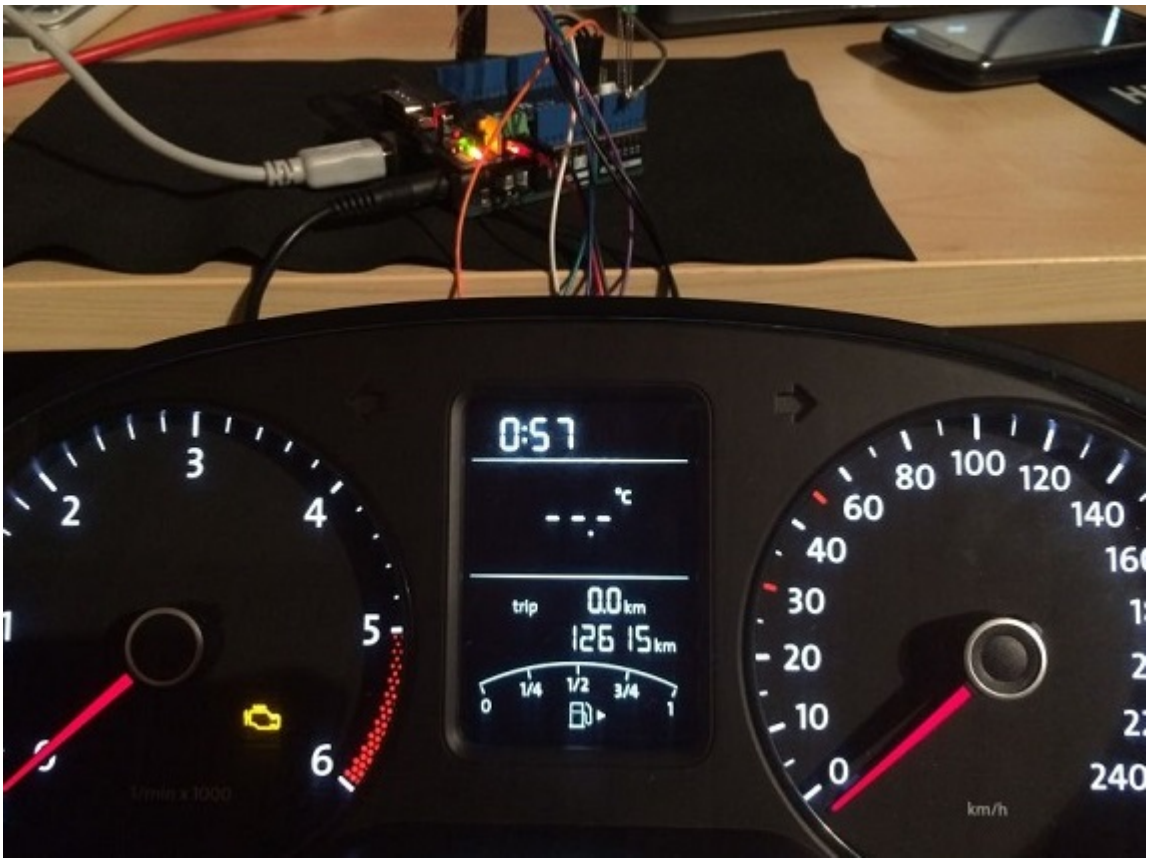
```
case (CAN_xxxKBPS):
    cfg1 = MCP_16MHz_xxxkBPS_CFG1;
    cfg2 = MCP_16MHz_xxxkBPS_CFG2;
    cfg3 = MCP_16MHz_xxxkBPS_CFG3;
    break;
```

然后，您就可以使用您需要的波特率了。当您使用的波特率时，请给我们一个github的请求，这样我可以添加到库来帮助其他人。感谢！

项目工程

如果您想使用CAN-BUS 扩展板做一些酷炫的项目，这里有一些项目可供参考。

大众 CAN BUS 小游戏



Ever wanted to play a car/truck simulator with a real dashboard on your PC? Me too! I'm trying to control a VW Polo 6R dashboard via CAN Bus with an Arduino Uno and a Seeed CAN Bus Shield. Inspired by Silas Parker. Thanks Sepp and Is0-Mick for their great support!

Make it NOW

黑进车载 CAN-BUS

Modern Vehicles all come equipped with a CAN-BUS Controller Area Network, Instead of having a million wires running back and forth from various devices in your car to the battery, its making use of a more clever system.

All electronic functions are connected to the TIPM, (Totally integrated Power Module), such as solenoids/relays to lock the doors or mini motors to wind the windows etc.

From each node (IE Switch pod that controls your windows or electric door locks) it broadcasts a message across the CAN. When the TIPM detects a valid message it will react accordingly like, lock the doors, switch on lights and so on.

Make it NOW

FAQ

Q1: 我无法从其他**CAN**设备获取数据。

- 检查连接是否正确
- 检查波特率设置是否正确

Q2: 串行监视器打印失败。

*检查CS引脚设置是否与代码匹配。对于CAN总线扩展板V1.1 / 1.2 /2.0 CS 引脚默认连接到 D9，其他则为 D10。

Q3. 如果我有其他问题，我在哪里可以找到技术支持。

- 您可以在 Seeed 论坛 Seeed Forum 留言或者or 给下列邮箱发送邮件至 **techsupport@seeed.cc**.

资源下载

- 【**PDF**】CAN-BUS Shield V2.0 Schmatics
- 【**Eagle**原理图】Schematic & PCB of CAN-BUS Shield V2.0
- 【**CAN_BUS_Shield**库】Arduino Library for CAN-BUS Shield
- 【**MCP2515**数据手册】MCP2515 datasheet
- 【**MCP2551**数据手册】MCP2551 datasheet
- 【示例程序】An OBD Demo
- 【**MCP2515**波特率计算工具】MCP2515 Baud Rate Tool
- 【**USB-CAN**分析器】USB-CAN Analyzer
- 【线缆**DB9**转**OBD2**】DB9 to OBD2 Cable