联发科技 LinkIt ONE 开发指南

版本：1.3

发布日期：2015 年 6 月 1 日

本指南内容及产品规范如有变更，恕不另行通知。

文档修改历史

版本	日期	说明
1.0	2014 年 9 月 8 日	初版
1.1	2014 年 12 月 4 日	修改: • Arduino 版本支持 • 移动网络支持说明（2G）
1.2	2015 年 2 月 13 日	更新: • 蓝牙（Bluetooth）GATT profile 和 API 支持 • SDK v1.1 发布
1.3	2015 年 6 月 1 日	SDK1.1.08（beta）版提供： • Mac OSX SDK 安装步骤 • 端口认别说明更新 • MacOSX 故障排除

目录

图表列表

1. 介绍

1.1. 什么是 LinkIt?

MediaTek LinkIt™是为穿戴式与物联网 (IoT)所量身打造的开发平台收集，该开发平台收集分别于两个不同的家族系列：

- LinkIt Assist, 针对简单的应用适合可穿戴与物联网设备，例如智能腕带，智能手表和智能安全和跟踪设备。这些设备提供用户设备上的反馈和控制选项，还可以使用 GSM 短信，GPRS，Wi-Fi 或蓝牙连接来与用户，其它智能设备和云应用来交换数据和控制消息。
- LinkIt Connect，针对单一应用适合物联网设备，例如智能灯泡以及智能家电并通过 Wi-Fi 或蓝牙连接， 使用云服务和智能手机来做远端控制。

而每个开发平台又将提供一个或更多芯片组和个别的 API，以满足客制式开发和设备的要求。因此，为了启动您创建原型器的创意， 我们提供个别不同的开发平台工具包含：

- 一个或更多 H D K 让您创建原型器。
- 一个 S D K 让您为装置编程固件或软件。
- 一个或更多硬件参考设计作为最终产品的电路板布局的基础。
- 全面的文件例如 API 参考，开发指南，芯片组说明和引脚图。
- 支持论坛。

1.2. 联发科技 LinkIt ONE 开发平台

联发科技的 LinkIt ONE 开发平台（如图 1）它替穿戴式与物联网装置提供既稳固又富弹性开发平台。此平台包含：

- 联发科技 Aster (MT2502) 世界上最小的系统级芯片 (SoC) 和搭配低功耗的 Wi-Fi 和 GPS 芯片组。
- LinkIt ONE API。
- LinkIt ONE 硬件开发工具包(HDK)。
- LinkIt ONE 软件开发工具包(SDK)。

图 1 LinkIt ONE 开发平台组件

LinkIt ONE HDK 包含了 LinkIt ONE 开发板，它与市面上功能最丰富的 Arudino UNO 的引脚相似。这块充满特色的开发板是联发科技与 Seeed Studio（知名的创新平台先驱厂商）同设计的技术结晶。与 UNO 不同的是：开发板内置了所有 LinkIt ONE API, GPS 与 Wi-Fi 功能，并提供多样的接口来连接传感器与其他周边设备。

LinkIt ONE SDK 以 Arduino IDE 插件呈现，是创客们软件开发工具的首选。安装此插件后，您能够容易的移植既有的 Arduino 代码到 LinkIt ONE，并使用各种 LinkIt ONE 的通讯 API：2G 移动式网络（GSM 和 GPRS）、蓝牙与 Wi-Fi。

有了 LinkIt ONE SDK 与 LinkIt ONE 开发板，等同于具备了开发创新原型产品的一切工具。

如果您不熟悉 Arduino，请参阅 Arduino Playground wiki ，或访问 Arduino.cc 以取得更多资讯。

1.3. LinkIt ONE 软件开发工具包 (SDK)

LinkIt ONE 软件开发工具包是以 Arudino IDE 插件方式发布。在 SDK 的 API 除了具备 Arduino 的核心功能，并兼具 LinkIt ONE 开发平台独特的功能包括控制数字引脚与解析模拟传感器的输入，让您快速的打造穿戴式与物联网原型产品。

如图 2 所示，用 LinkIt ONE SDK 可以写出利用 LinkIt ONE API 的 Arduino 应用程序(Sketch)。这些 API 运行在 Run-time 操作系统之上，来让你使用 LinkIt ONE 开发板上的功能。

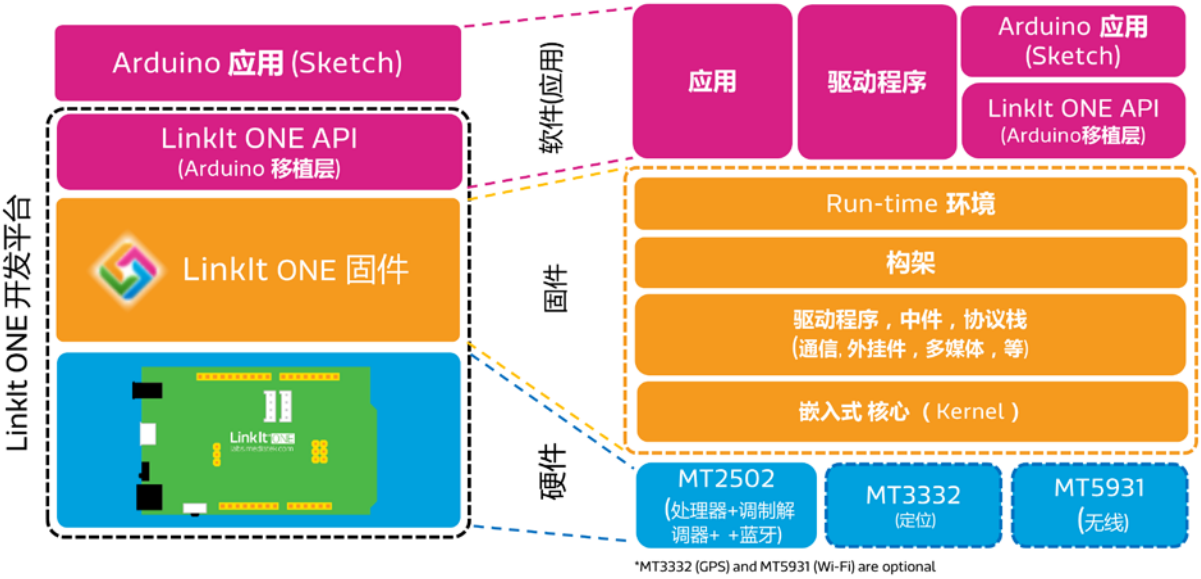

图 2 LinkIt ONE 开发平台的架构

如果您开发过 Arduino，请点击联发科技 LinkIt ONE API reference，可以发现大部分的模块是对应 Arduino API 风格所设计的，便于使用。

1.3.1. 描绘你的构想

使用 LinkIt ONE SDK 编程就跟写 Arduino Sketch 一样简单。一个 Arduino Sketch 是一个源代码文件，代表了 LinkIt ONE 的核心控制逻辑。它包含了：

- `setup()` 负责初始化资源，如 Wi-Fi 模块。
- `loop()`连续监听并处理由硬件传感器或软件模块（如蓝牙）产生的事件。loop()会不断的运行，直到开发板关机才会停止。

1.3.2. 运行 Arduino Sketch

如图 3 所示，LinkIt ONE SDK 首先将 Sketch 编译成 LinkIt ONE 执行文件（VXP 文件）。IDE 插件接下来会将 VXP 文件读入 LinkIt ONE 开发板的文件系统中。开机后，LinkIt ONE 会自动执行加载的 VXP 文件。VXP 可执行文件是由 run-time 运行时环境加载的。

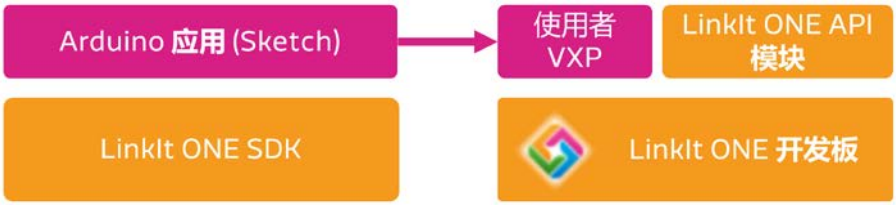

图 3 开发与测试过程

虽然 Sketch 是个单线程循环，Run-time 的运行环境则是多线程的软实时环境。VXP 是以一个独立的线程在运行。 Arduino 插件的包装层则是负责传送请求到，在其他线程上运行的，服务模块。

1.3.3. LinkIt ONE API

LinkIt ONE SDK 包含所有 LinkIt ONE 板可用的 LinkIt ONE API，如表 1：

表 1 LinkIt ONE SDK 的可用模块表

Arduino 功能	数字 I/O	由单个引脚传送与接收数字信号
	高级 I/O	由单个引脚传送与接收以 pulse 或 byte 为形式的数据。
	模拟 I/O	由单个别引脚经 ADC 传送与接收模拟信号
	Serial	经串行连接到其他装置以交换数据
	Time	获得当前时间与设定延迟
	Interrupts	由中断服务程序(ISR)处理外部中断
	Math	各种基本的数学函数，如比特与字节、随机数与三角函数。
	Servo	用于控制伺服马达
	SPI	用串行外设接口(SPI)协议与外围设备沟通
	Wire (I^2C)	用内部集成电路(I2C)协议与外围设备沟通
	Stepper	用于控制步进马达

LinkIt ONE 连接、存储、与多媒体	GSM	对短信服务(SMS)进行编辑、发送、接收、读取
	GPRS	由 GPRS（2G）连接网路与传输数据
	Storage	在 SD 卡与内部闪存操作文件系统
	GPS	由 GPS 硬件取得位置信息
	WiFi	由 Wi-Fi 连接网路与传输数据
	Bluetooth	由 SPP 蓝牙规范连接并与其他蓝牙设备之间进行数据传输
	Audio	播放 MP3, AAC 与 AMR 文件
	Battery	检测电池电量与充电状态
	DateTime	获取与设定当前日期与时间
	EEPROM	从 LinkIt 的 EEPROM 写入与读取，使数据在断电后仍会保存。

更多 API 的细节请参阅第 4 章：LinkIt ONE API 指南；完整的 API 文档 MediaTek LinkIt ONE API reference 则在 MediaTek 创意实验室网站上。

1.3.4. 延伸你的 Sketch

Sketch 文件都是用 C++编写，而 LinkItAssist 2502 执行环境的底层接口为 C，编译工具链为 GNU gcc。这可能开创了使用 C 与 C++开源库做为 LinkIt ONE 的可能性。然而 SDK 仅支持单线程的编程模式—以提供简单易用的硬件模型开发环境 —所以将依赖于多线程的程序库移植到 LinkIt ONE 不是一件容易的事。另一个加入外部程序库的考虑是：您的 Sketch 占用的 RAM 的大小，在 LinkIt ONE 开发板的 4MB 中仅有 2MB 可用。

1.4. 硬件

LinkIt ONE 是开源且高效能的开发板。核心为世界领先的穿戴式联发科 Aster (MT2502) 系统芯片 (SoC)，合并了高效能 Wi-Fi (MT5931)与 GPS (MT3332) 芯片组，提供了功能丰富的开发板。也与 Arduino UNO 有着相似的连线引脚，以连接各种传感器与周边设备。

LinkIt ONE 开发板是 Seeed Studio 与联发科技共同设计的产品。结合了双方对开源硬体与领先同行业的参考设计知识。

1.4.1. LinkIt ONE 接脚图

LinkIt ONE 开发板提供与 Arduino UNO 相似的引脚设置，如图 4 所示：

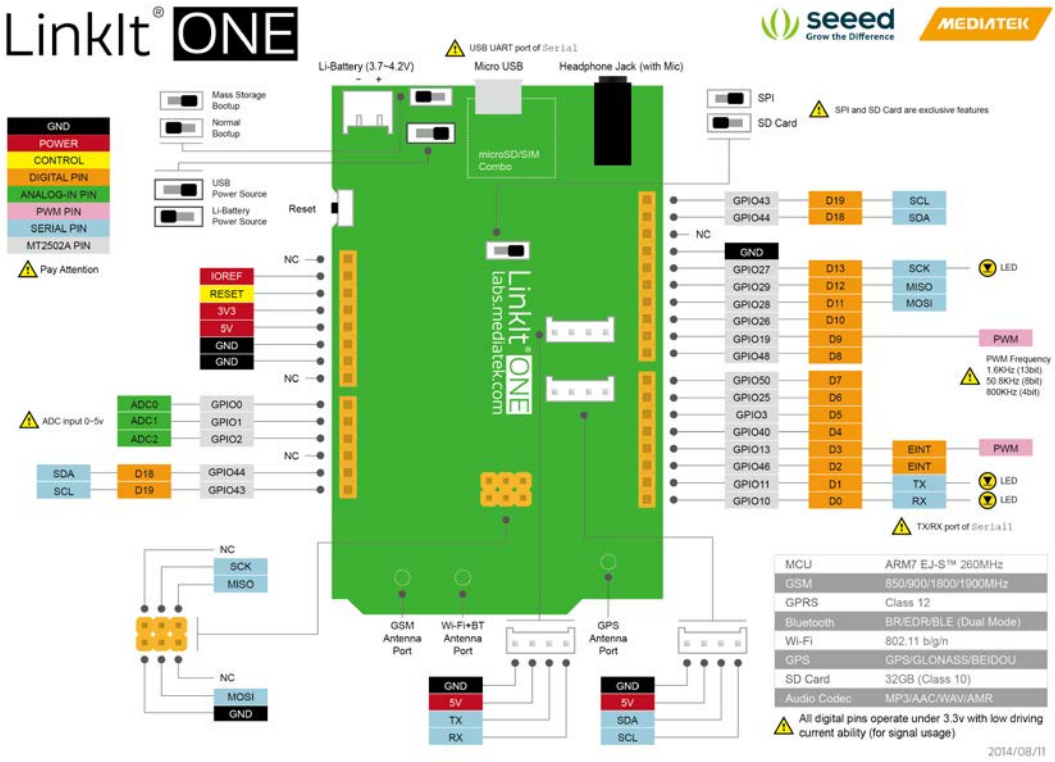

图 4 LinkIt ONE 开发板的引出脚配置

1.4.2. LinkIt ONE 开发板规格

表 2 为 LinkIt ONE 开发板的详细规格

表 2 LinkIt ONE 开发板规格

分类	功能	规格
微控制器	芯片组	MT2502A (Aster)
	核心	ARM7 EJ-S™
	主频	260MHz
PCB 板大小	尺寸	3.3 x 2.1 inches
内存	闪存	16MB
	RAM	4MB
电源	电池插孔	3.7~4.2V 锂离子电池
	单个 I/O 引脚的直流电流	0.3~3mA

分类	功能	规格
数字 IO 脚	引脚数目	16 D0~D13, SDA, SCL
	电压	3.3v
模拟输入脚	引脚数目	3 A0, A1, A2
	电压	0~5V
PWM 输出脚	引脚数目	2 D3 and D9
	电压	3.3v
	最大分辨率	13bit (自定义)
	最高频率@分辨率	1.6kHz@13bit 50.8kHz@8bit 800kHz@4bit (自定义)
外部中断	引脚数目	2 D2 and D3
I2C (master only)	组数目	1 SDA, SCL
	速度	100Kbps, 400Kbps, 3.4Mbps
SPI (master only)	组数目	1 D11(MOSI), D12(MISO), D13(SCK)
	速度	104Kbps~26Mbps
UART (Serial1)	组数目	1 D0(RX), D1(TX)
	电压	3.3v
UART on USB (Serial)	组数目	1
通讯	GSM/GPRS	850/900/1800/1900 MHz
	GPRS	Class 12
	蓝牙	BR/EDR/BLE (Dual 模式)
	Wi-Fi	802.11 b/g/n
定位	GPS	GPS/GLONASS/BEIDOU
用户存储	闪存	10MB
	SD 卡	最高 32GB (Class 10)
可执行程序大小 (编译 Sketch 文件)	RAM (Code+RO+RW+ZI+Heap)	2MB

1.5. 加入我们的生态系统

下一波电子产品革新的浪潮已到来- 可穿戴与物联网设备。在此领域一直扮演关键角色的联发科技将两个领域的精华合并—联发科技的手机制造商、电子装置制造商与电信商的既有生态环境，同开放式、充满活力的 Arduino 创客社区相结合。不论您是个创客、设备制造商、学生、DIY 爱好者，或是程序设计师，都可以用这个强大且易用的平台来做出创新的设计。欢迎前往 labs.mediatek.com 并加入联发科技创意实验室 ，我们期待您的加入并一同做出好产品。

2. 开始使用

这章节提供开始 LinkIt ONE 开发平台的准备，包含了：

- 支持的开发环境。
- 安装 Arduino IDE。
- 安装与设置 LinkIt ONE SDK。
- 创建第一个项目。

2.1. 环境

目前支持的环境为微软 Windows XP、Vista， 7， 8 与 MacOS X10.09 或 10.10。

2.2. 安装联发科技 LinkIt ONE SDK（针对 Arduino）Windows 版本

本章节说明如何安装联发科技 LinkIt ONE SDK（针对 Arduino）在使用微软 Windows 操作系统的电脑。

2.2.1. 安装 Arduino IDE

LinkIt ONE SDK 以 Arduino IDE 1.5.6-r2 BETA 和 1.5.7 BETA 插件的形式发布。如果您已经安装一个支持的 Arduino ，您可以忽略这个步骤。如果您还没有安装，请依照：

1) 从 Arduino 网站下载 Arduino 软件
2) 安装 Arduino 。

3) 安装结束后，请记下您安装 Aruino 的文件夹，这也是 arduino.exe（如图 5）所在的文件夹。您将安装 LinkIt ONE SDK 到此文件夹。

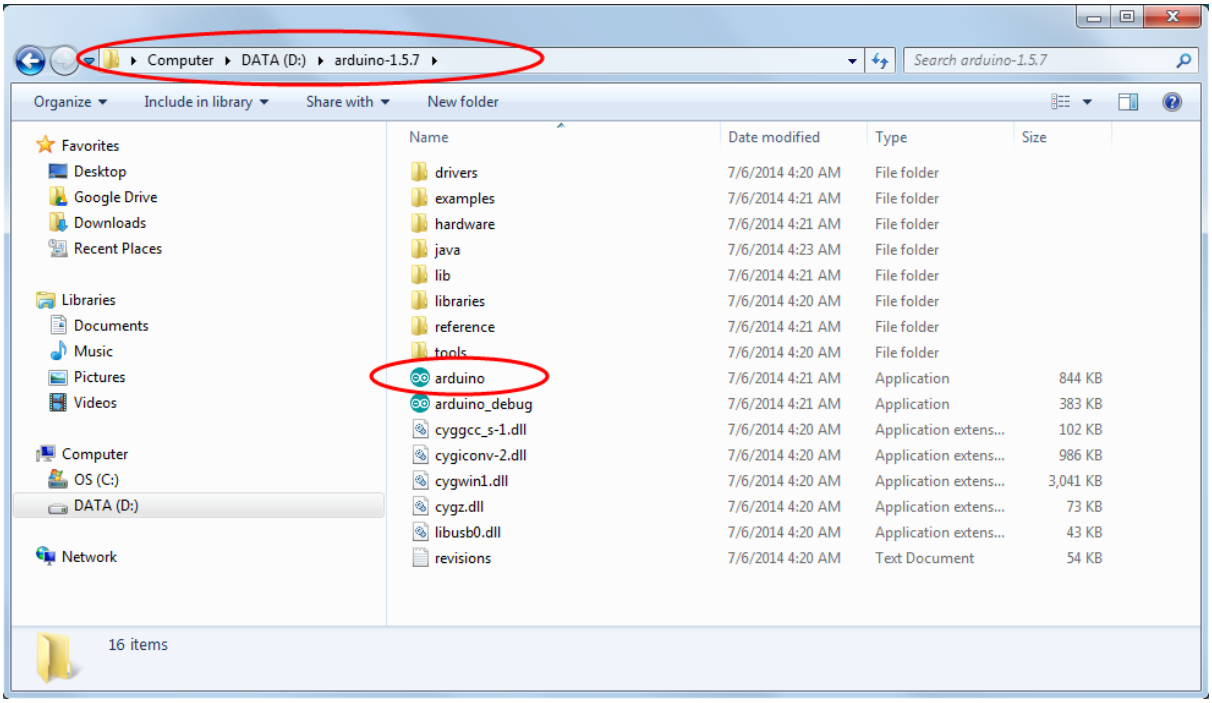

图 5 Arduino 安装文件夹包含了 Arduino.exe

2.2.2. 安装 LinkIt ONE SDK

如何安装 LinkIt ONE SDK：

1) 由此下载 Windows 版本的 LinkIt ONE SDK（针对）ZIP 文件。
2) 解压 LinkIt ONE SDK ZIP 文件的内容。
3) 确认 Arduino IDE 没有在运行。
4) 执行 LinkIt ONE SDK 安装程序。
5) 在 **Select Destination Location** 按下 **Browse** 并指定您安装 Arduino IDE 的文件夹（之前所提到的）并点击 **Next**。

6) SDK 安装完成后的对话框如图 6 所示。如果您是首次安装 SDK，在按下 **Finish** 之前请确保框选 **Install the MediaTek USB Driver。**

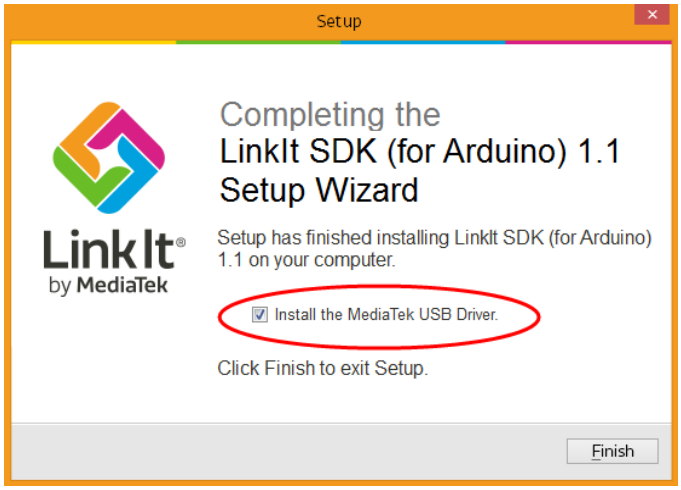

图 6 在结束前请框选 install the MediaTek USB driver

下一次更新 SDK 时就不用再安装联发科技 USB 驱动程序。

2.2.3. 设置 Arduino IDE 以使用 LinkIt ONE 开发板

现在您已经安装好 LinkIt ONE SDK 了，再设置好 Arduino IDE 即可使用您的 LinkIt ONE 开发板：

1) 执行 Arudino IDE。

2) 在 Arduino IDE 的 **Tools** 选单指向 **Board** 然后点击 **LinkIt ONE**，如图 7 所示。

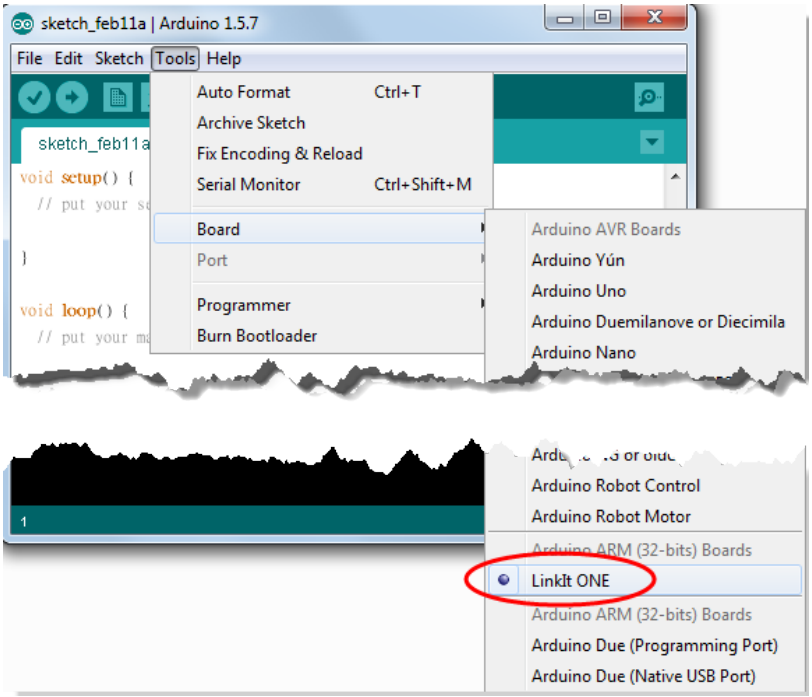

图 7 设置 LinkIt ONE 开发板选项

3) 打开微软控制面板在点击**系统**：

- 在 Windows 7 and 8，点击**装置管理。**
- 在 Windows XP，点击**硬件**再选择**装置管理。**

4) 在**装置管理**内，浏览 **Ports (COM & LPT)**，如图 9。

5) 用 Mirco-USB 电缆线将 LinkIt ONE 开发板与电脑连接，如图 8。

图 8 Micro USB 连接器在 LinkIt ONE 开发板上

6) 一个新的 COM 装置会在装置管理的 **Ports (COM & LPT)**底下出现，如图 9。请记下 **MTK USB Debug Port**,的 COMx 串口号码，会在下一步用到。

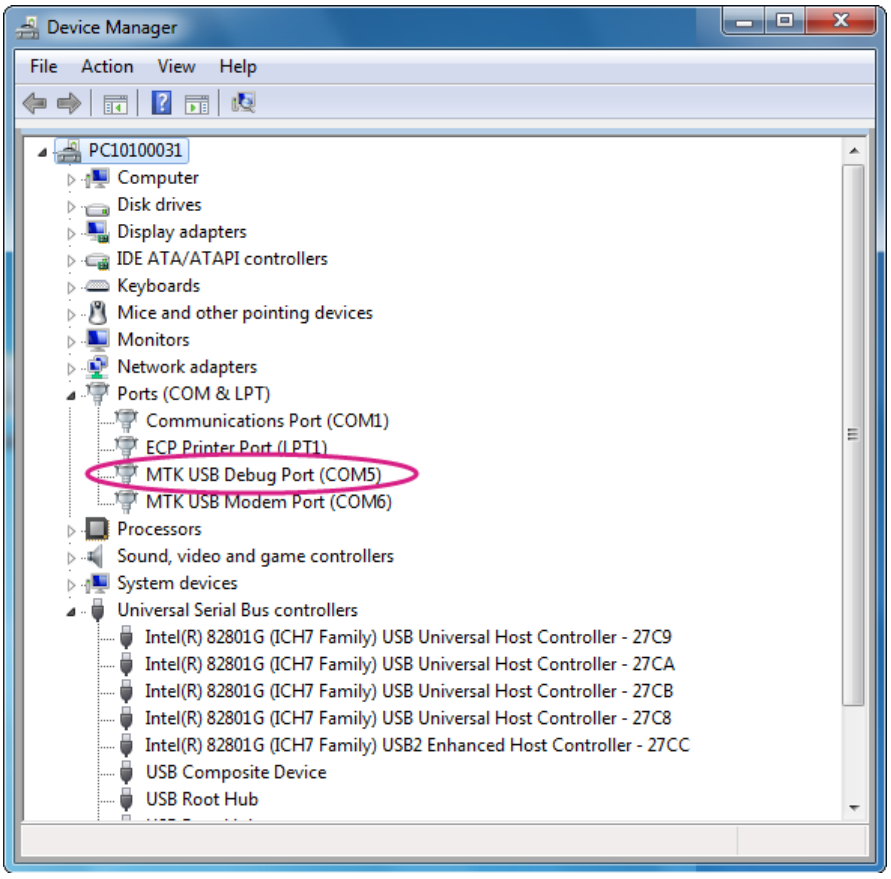

图 9 列在装置管理的 LinkIt ONE 开发板

7) 在 Arduino IDE 的 **Tools** 选单上，指向 **Port** 再选择刚刚所记下的 LinkIt ONE 开发板串口号，如图 10 所示。

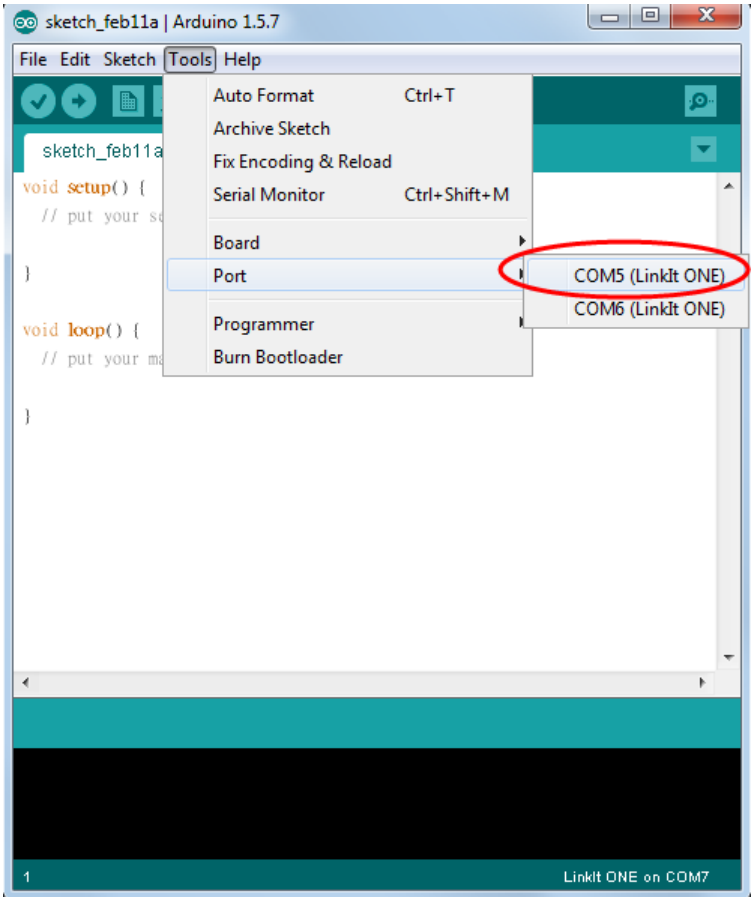

图 10 在 Arduino IDE 指定到 LinkIt ONE 开发板串口号码

您的 Arduino IDE 现在已经设置好 LinkIt ONE SDK 与 LinkIt ONE 开发板，您可以开始第一个项目了！

2.3. 安装联发科技 LinkIt ONE SDK（针对 Arduino）苹果 Mac 版本

苹果 Mac OS X 只有在联发科技 LinkIt ONE SDK (针对 Arduino) 1.1.08 (beta)版本支持。

本章节解释如何安装 Mac 版本的联发科技 LinkIt ONE SDK（Arduino 适用）在 Mac OSX 10.09 或 10.10 的电脑上。

2.3.1. 安装 Arduino 软件

LinkIt ONE SDK 是个 Arduino1.5.6 或 1.5.7BETA 的插件。如果您已经安装了该 Arduino 软件，请直接看下一个章节，否则请按照以下步骤安装 Arduino：

1) 下载 Java SE Runtime 环境 7 并安装在电脑上。

2) 下载 Arduino 软件于 Arduino 网站。

3）安装 Arduino 软件至/Applications 文件夹，并确认 Arduino 安装路径是 /Applications/Arduino.app.

2.3.2. 安装 LinkIt ONE SDK

请依照以下步骤安装 LinkIt ONE SDK：

1) 由此下载苹果 Mac 版本的 LinkIt ONE SDK ZIP 文件。

2) 解压 LinkIt ONE SDK ZIP 文件的内容至您的 Applications 文件夹。

3) 确认 Arduino IDE 没有在运行。

4) 在 Applications 文件夹里打开 Utilities，打开 Terminal 然后：

   a) 切换文件夹（用 cd 命令）至 LinkIt ONE SDK ZIP 文件, 例如 mediatek_linkit_sdk(for arduino)macosx-xxxx 文件夹。

   b) 在 LinkIt ONE SDK 文件夹里之后请执行以下命令：

```
# ./install_linkIt_sdk -i 您的_arduino_app 路径
```

   您的 arduino 路径是指您的 arduino app 文件夹，它通常是 /applications/arduino.app.

安装好之后，arduino.app 文件夹内构造将与下图 11 相似：

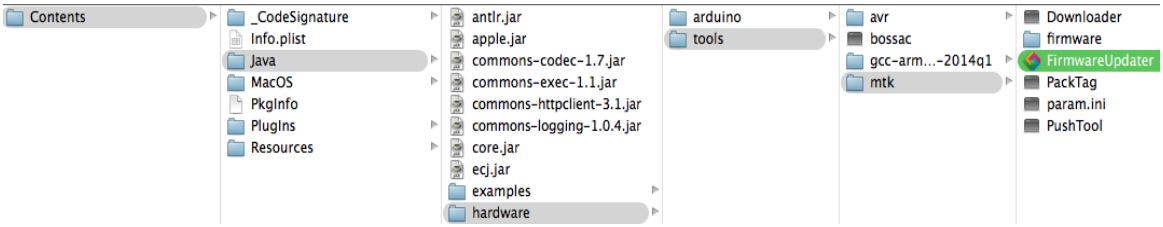

图 11 Arduino.app 文件夹内容显示 LinkIt ONE SDK 插件

2.3.3. 配置 Arduino 软件准备使用 LinkIt ONE 开发板

注：第一次使用 LinkIt ONE 开发板时，建议您更新固件，请参考 3.1 “固件更新请求 “。

LinkIt ONE SDK 安装好之后您需要配置 Arduino IDE 来使用 LinkIt ONE 开发板。请依以下步骤配置动作：

1) 请断开您的 LinkIt ONE 开发板于您的 Mac。

2) 将 LinkIt ONE 开发板切换至 UART (Normal bootup) 模式（图 12）。请参见 LinkIt ONE 引脚图（图 4）了解更多细节。

图 12 Normal bootup (UART)切换模式

3) 重新开启您的 Mac。

4) 使用一个 micro-USB 线将 LinkIt ONE 开发板和 Mac 连接起来，如图 8。

5) 在 Arduino 软件 Tools 菜单指向 Board 并电击 LinkIt ONE,如图 13。

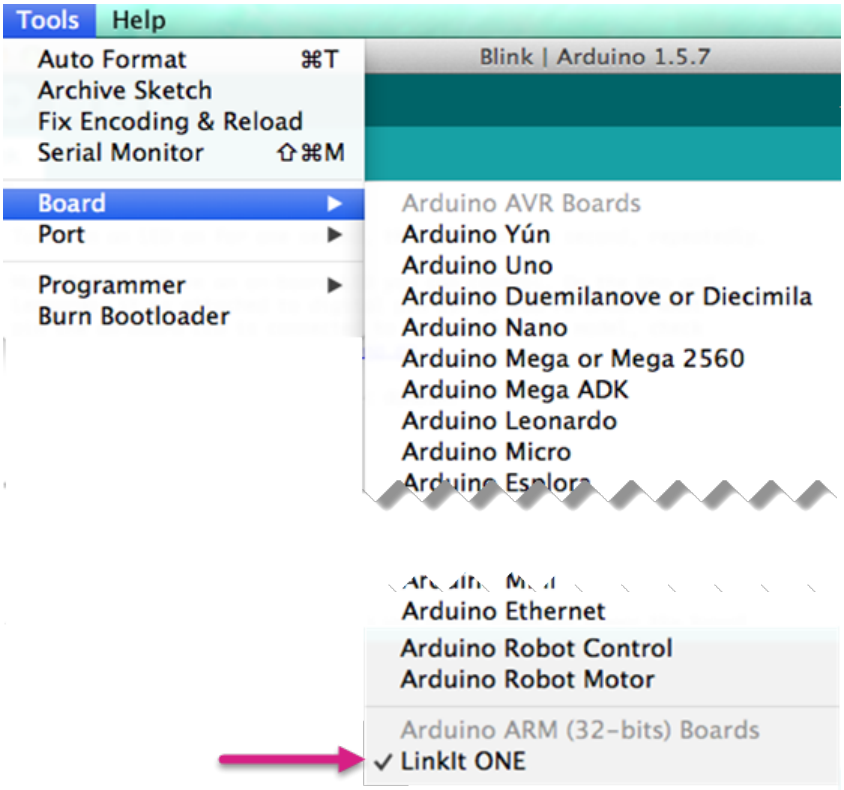

图 13 Mac OSX 的 LinkIt ONE 开发板配置菜单选择

注：如果您在此列表中未看见您的开发板，请确认您的板正确的连接上 Mac 并有电源，如 2.3.3“配置 Arduino 软件准备使用 LinkIt ONE 开发板“

在 Arduino 软件 Tools 菜单指向 Port,您将看见两个设备给 LinkIt ONE 开发板 `/dev/cu.usbmodem(xxxx)`。电击那个没有 LinkIt ONE 字尾和数字较大的选择，如图 14。

注：因不同电脑而会有不同的字尾，例如您可能看见类似 1413 至 fa132 的字尾。

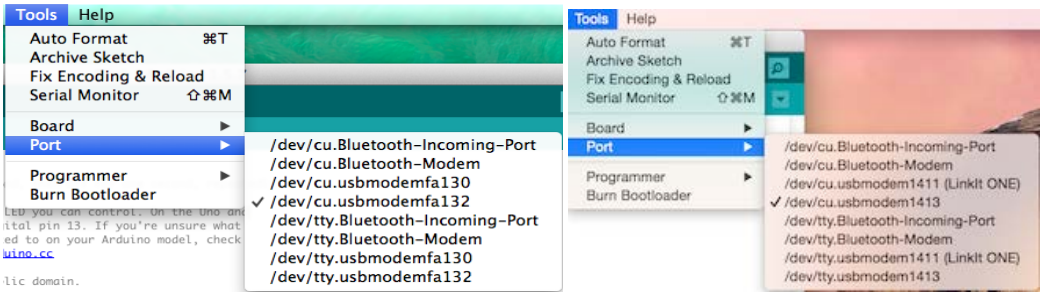

图 14 在 Arduino 与不同 Mac OS X 版本中配置 LinkIt ONE 开发板的接口

确认接口选择：

- 电击苹果菜单图标，然后关于这个 Mac。
- 窗口开启之后，在 **Overview** 电击 **System report**。
- 在 System Information 左边电击 USB，USB device tree 显示接口和开发板信息列表。每一个板或设备将列在与它们连接的接口下面。

2.4. 设计第一个项目

按步骤创建第一个 LinkIt ONE 项目， Windows 和 Mac OSX 的步骤是同样的：

1) 在 Arduino IDE 的 **File** 选单上指向 **Examples**，再指向 **01. Basics** 点击 **Blink** 以开启 Blink 范例。
2) 查看图 12 的开关以确保您的开发板为 SPI 模式，开关应在靠近 LED 的位置。（到 4.9“SPI”串行外围接口(SPI)查看更多细节）。

3) 在 **File** 选单点击 **Upload**，或直接点击图 15 的上传按钮。

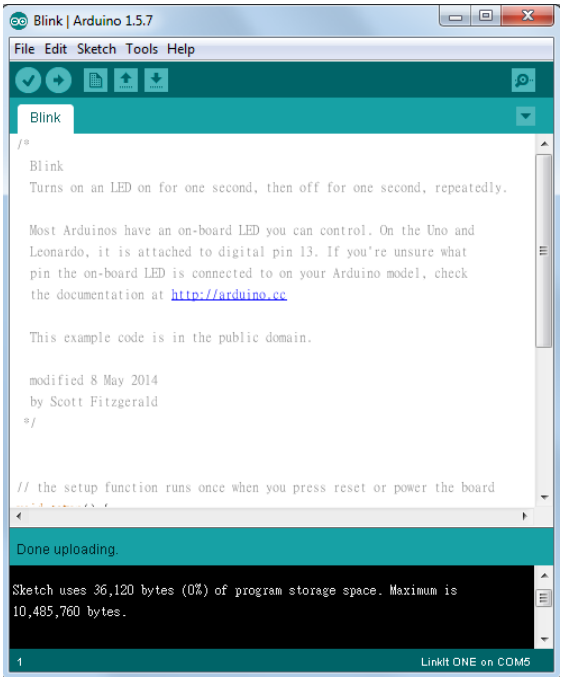

图 15 在 Arduino IDE 工具栏的上传按钮

4) 稍待一会。如果一切运作正常，一个 **Upload Done** 的信息就会显示在 Arduino 软件，而且在 LinkIt ONE 开发板上的 LED 会开始每秒闪动，如图 16 所示。

图 16 LinkItONE 板 LED

恭喜您，您已经安装好 LinkIt ONE SDK 并执行了第一个 LinkIt ONE 项目。您已经准备好开始用 LinkIt ONE 与 LinkIt ONE SDK 创建新一代的穿戴式与物联网装置！

2.5. 在 Arduino 软件连接串口监视

Serial Class 让软件能够输出字符至 LinkIt ONE 开发板的 USB COM 口，您可以使用 Arduino 内置的串口监视或另一个 COM 口来输出字符。用于 Serial class 输出的 COM 口与上传您的 Arduino sketch 的 COM 口是不同个的。

2.5.1. 在 Windows 选择串口监视 COM 接口

LinkIt ONE 开发板启动后会有两个 COM 口，一个是 Debug 口，另一个是 Modem 口，请选择 Modem 口来经由 Arduino 的 Serial classs 串口监视 来显示字符输出。请依照以下步骤来选择 Modem 口：

1) 如何确定 Modem 口的 COM 口号码：
   a) 打开 Windows 控制面板并电击系统。
      i) Windows 7 和 8 电击装置管理。
      ii) Windows XP 电击硬件然后电击装置管理。
   b) 在装置管理，到 Ports（COM&LPT）并找到 MTKUSB Modem 口（COMxx），如图 17 请注意它的口号码（xx）。

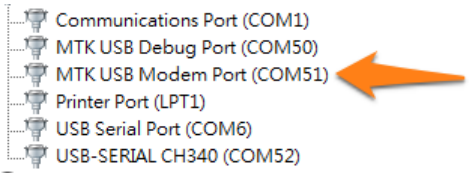

图 17 装置管理内的 MTKUSB Modem 口（COMxx）

2) 在 Arduino Tools 菜单指向 Port 并电击 COMxx (LinkIt ONE)选项，如之前 Device Manager 里相同的 COM 号码，如图 18。

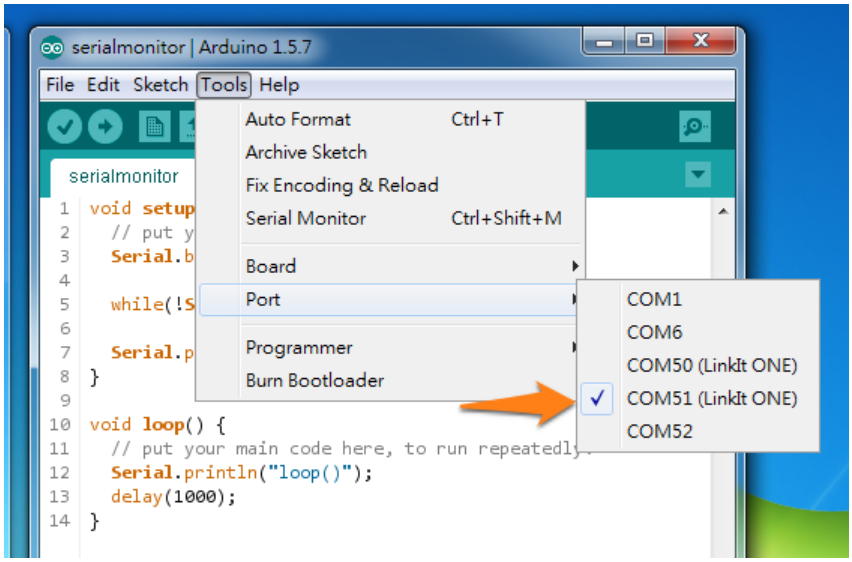

图 18 设置串口监视的接口

3) 在 Tools 菜单电击串口监视。

2.5.2. 在 Mac OS X 选择串口监视 COM 接口

Mac OS X 里每一个 COM 口会产生一个 CU device 和一个 TTY device,因此 LinkIt ONE 开发板总共会有四个 device 。

要经由 Arduino 的 Serial classs 串口监视 来显示字符输出，请在 Arduino 选 TTY USB Modem 口，例如/dev/tty.usbmodemxxxx (LinkIt ONE), 如图 19。该 USBModem 字尾（下图范例 1411）将因不同 Mac 而不一样。

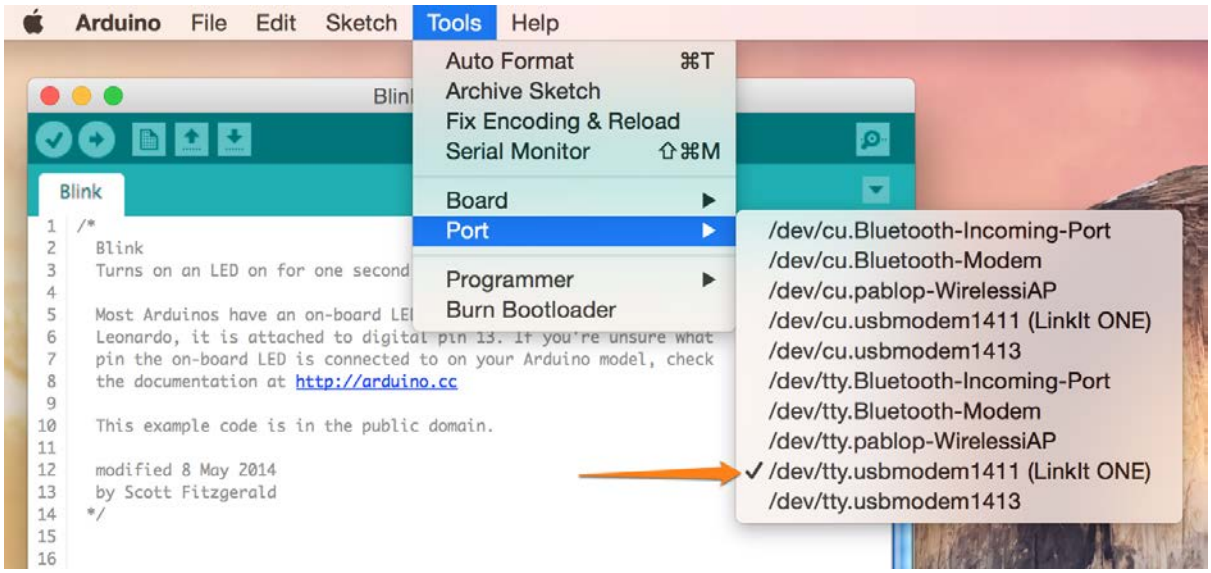

图 19 在 Mac OSX 选串口监视 COM 接口

3. 故障排除

3.1. 固件更新请求

您上传 Sketch 时，如果 Arduino 软件回复如”Please upgrade your firmware”的错误讯息，请按照接下来的步骤更新 LinkIt ONE 的固件：

1) 请切换至大容量存储模式，如图 20。

图 20 LinkIt ONE 大容量存储模式

2) 开启 LinkIt ONE Firmware Updater

a) 在 Windows,它位于 Arduino 软件文件夹内的 hardware/tools/mtk，如图 21。

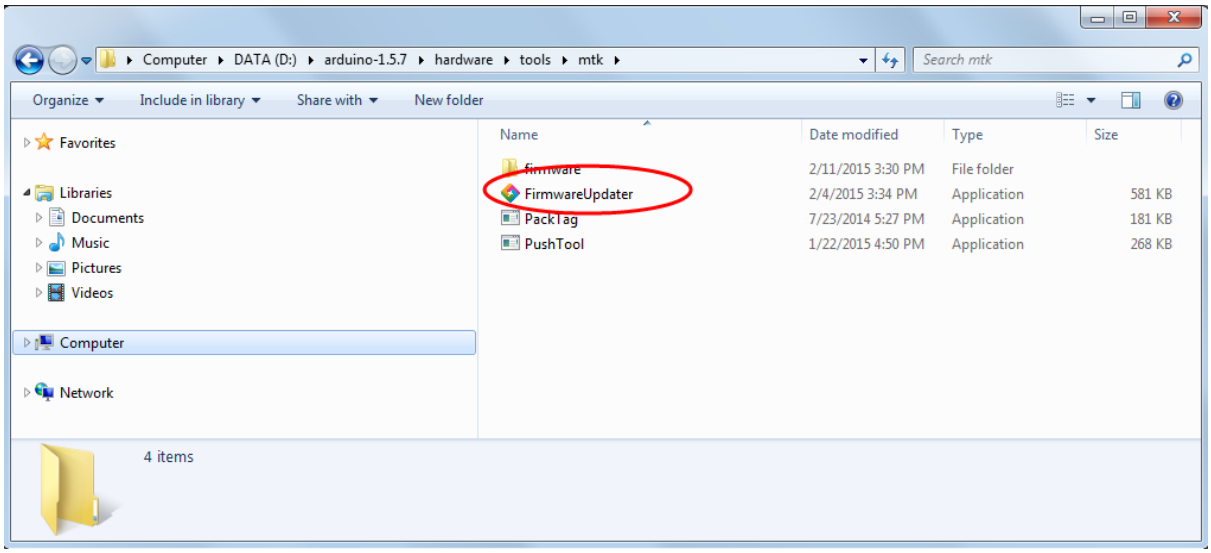

图 21 在 Arduino IDE 文件夹内 LinkIt Firmware Updater 的位置

b) 在 Mac OS X, 先安装 COM 口驱动程序，这是要给固件更新使用，如章节描述。驱动程序安装好了之后，打开 Arduino Package Content 如图 22。

图 22 Arduino Package Content

启动固件更新软件：

i) Arduino 1.5.6：

```
Arduino.app/Contents/Resources/Java/hardware/tools/mtk/FirmwareUpdater
```

ii) Arduino 1.5.7：

```
Arduino.app/Contents/Java/hardware/tools/mtk/FirmwareUpdater
```

您不会在 Mac OS X 上看见任何有关固件更新的提示或错误信息显示，但它是一个必要的步骤，以便初始化指定通信端口的顺序。

3) 按下 LinkIt Firmware Updater 内的下载按钮，如图 23。

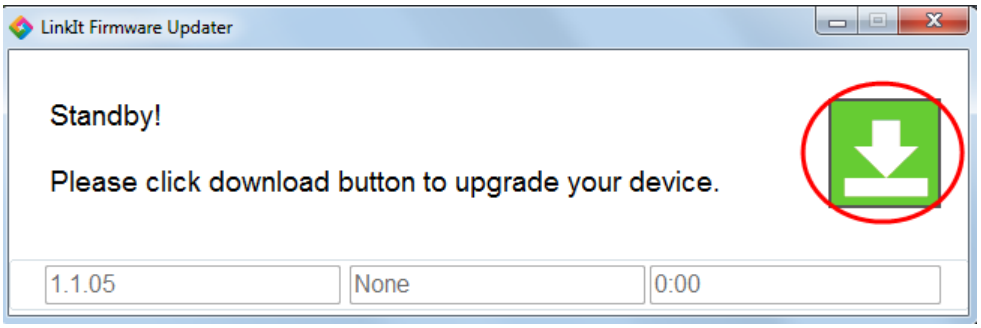

图 23 按下下载按钮来开始固件更新程序

4) 依照图 24 说明，断开 LinkIt ONE 与您电脑的连线，再重新接上。

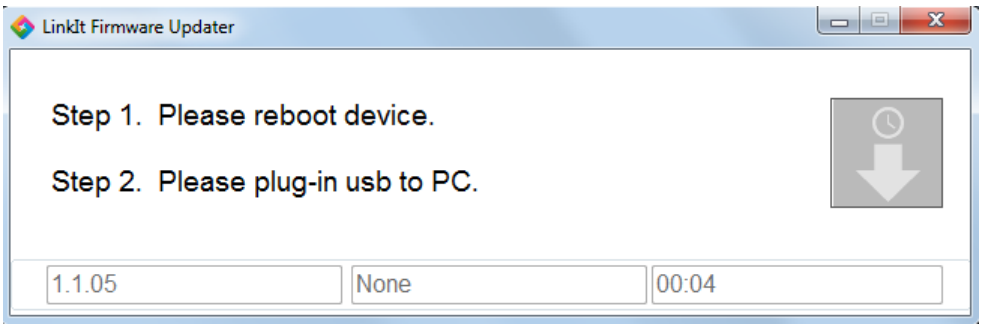

图 24 更新开发板固件的说明

5) 当 LinkIt ONE 开发板重新接上，固件即自动开始更新。LinkIt Firmware Updater 会回复正在下载固件，如图 25。

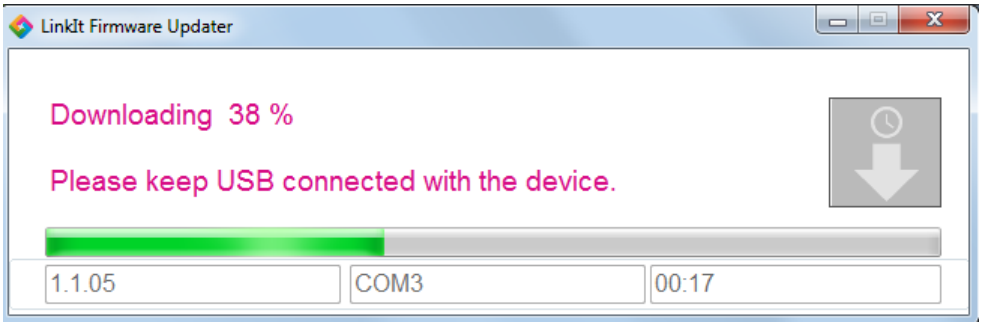

图 25 正在下载开发板的固件

6) 稍后，LinkIt Firmware Updater 会确认固件更新已经完成，如图 26。

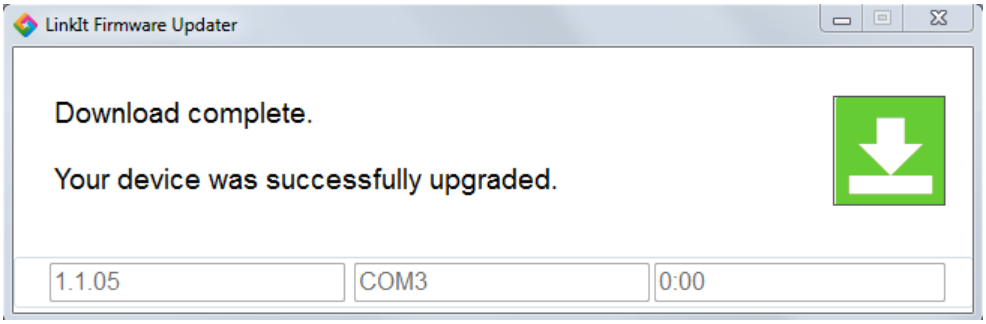

图 26 固件更新已经完成

7) 将开发板切回一般存储模式，并断开 USB 并重新接回以重新开机。

您应现在应该能够上传 Sketch 到 LinkIt ONE 了。

3.2. Sketch 不能上传到 LinkIt ONE

您的 Sketch 不能上传到 LinkIt ONE，请确认：

1) 为了确认 LinkIt ONE 有连线到您的 Windows 电脑，请依照以下步骤：

   a) 开启微软控制面板，点击**系统**并：

      i) 在 Windows 7 和 8, 点击 **装置管理。**

      ii) 在 Windows XP, 点击**硬件**再进入**装置管理。**

b) 在**装置管理**，浏览 **Ports (COM & LPT)** 并确认 **MediaTek USB Debug Port** 有在列表中，如图 9。

Mac OS X，请依照以下步骤：

i) 电击苹果菜单图标，再电击关于这台苹果。

ii) 新窗口的 Overview 里电击 System report.

iii) 在 System Information 里左边电击 USB， USB Device Tree 将列出设备和接口连接表，您将清楚的看见它们互连的信息。

2) 为确认 Arduino IDE 已设定使用 LinkIt ONE，请在 **Tools** 选单指向 **Board** 并确认 **LinkIt ONE** 已经勾选。

3) 为确认 Arduino IDE 内的 LinkIt ONE 已指向正确的接口。请在 **Tools** 选单指向 Port 并确认有无勾选正确的接口。

4) 如果 Sketch 仍然无法更新，请依照 3.1 “固件更新请求”以更新开发板的固件。

如果这些步骤仍无法解决问题，您可以在 Arduino Troubleshooting Guide 找到有用的信息或搜寻 LinkIt ONE HDK 论坛（若这个议题没有讨论过，请新增这个议题）。

3.3. 开发板没有回应

如果您的开发板经过检查设定以及更新固件仍无法运作，请查看 Arduino 网站，遵循论坛的建议，或请联络 Seeed Studio 的技术支持。

3.4. Mac OS X 10.10 COM 端口认别

LinkIt ONE 开发板总共有三个 COM 端口：bootread-only-memory (BROM) COM 接口，Debug 端口和 Modem 端口。当它们与电脑连接上之后，这些端口的自动认别（LinkIt ONE 开发板）功能是由固件和驱动程序所支持与提供的。以下说明提供关于 COM 端口连接和使用 USB 接口访问 Mac OS X10.10．

1) BROM COM 端口：

BROM COM 端口在设备启动不久后出现，并在大约两秒钟后消失。COM 端口是硬线化的，因此无法进行修改。LinkIt 开发板的固件更新就是由 BROM USB 端口完成， 早期版本的 OS X 兼容了通用通信设备类（CDC）的抽象控制模型（ACM）的 USB 驱动程序支持。但是，在 OS X10.10，内置的驱动器具有受限顺序由 USB 设备提供的服务。要解决这个问题，您必须在 OS X10.10 安装一个特殊的驱动程序。该驱动程序包含在苹果 Mac 版 LinkIt ONE 的 SDK（针对 Arduino）让您执行固件更新。

2) Debug COM 端口和 Modem COM 端口：

该两个端口在嵌入操作系统启动后出现，当开发板处于正常启动模式时（如）您可以：

a) 用 XMODEM 协议上传 sketch 到开发板.

b) 提供串口通信至 Arduino 软件:

这些 COM 端口由微控制器（MCU）和嵌入式操作系统驱动程序仿真。一旦固件更新完成，debug COM 端口和 Modem COM 端口将通过内置的 OS X 通用 CDC ACM 的 USB 驱动程序来支持。

3.4.1. Mac OS X 10.10 访问 USB COM 解决方案

如果您在 Mac OS X 10.10 上使用 LinkIt ONE 开发板，那么您将需要安装 USB 驱动程序才能在 LinkIt ONE 开发板上启动连接固件更新。 OS X 10.10 内置通用 CDC ACM 的 USB 驱动程序与 LinkIt ONE 开发板的三个端口不能兼容。因此您需要安装客制的驱动程序来启用固件更新，BROM 端口的驱动程序安装步骤如下：

1) 解压 LinkIt ONE SDK 驱动程序文件夹（如 2.3.2 “安装 LinkIt ONE SDK” ）内的 `MacOS_USB_COM_Driver_BM_xxx.xx.x.zip`。
2) 执行安装包（BMCDCACM_Driver_v115.05.0.mpkg）和电击下一步。
3) 驱动程序安装完毕之后从新开启您的 Mac 电脑。

BROM USB 驱动程序已安装，您现在可以访问联发科技 USB 串口端口。

当执行 BROM USB 驱动程序安装时，您可能遇见困难如图 27。

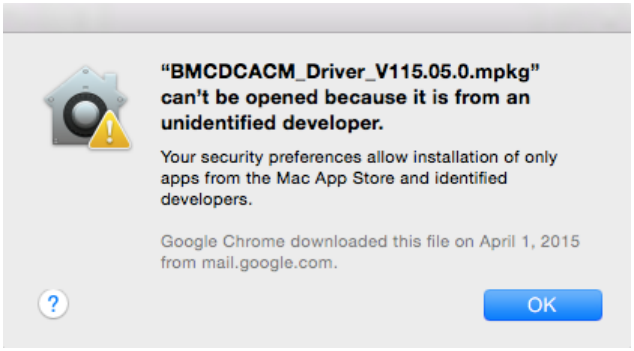

图 27 安装 BROM USB 驱动程序失败信息

解决方法是在安全设定内修改安装许可如下：

1) 在苹果菜单里启动系统偏好设置并电击安全和隐私。

2) 在一般标签里电击 Open Anyway (如图 28)。

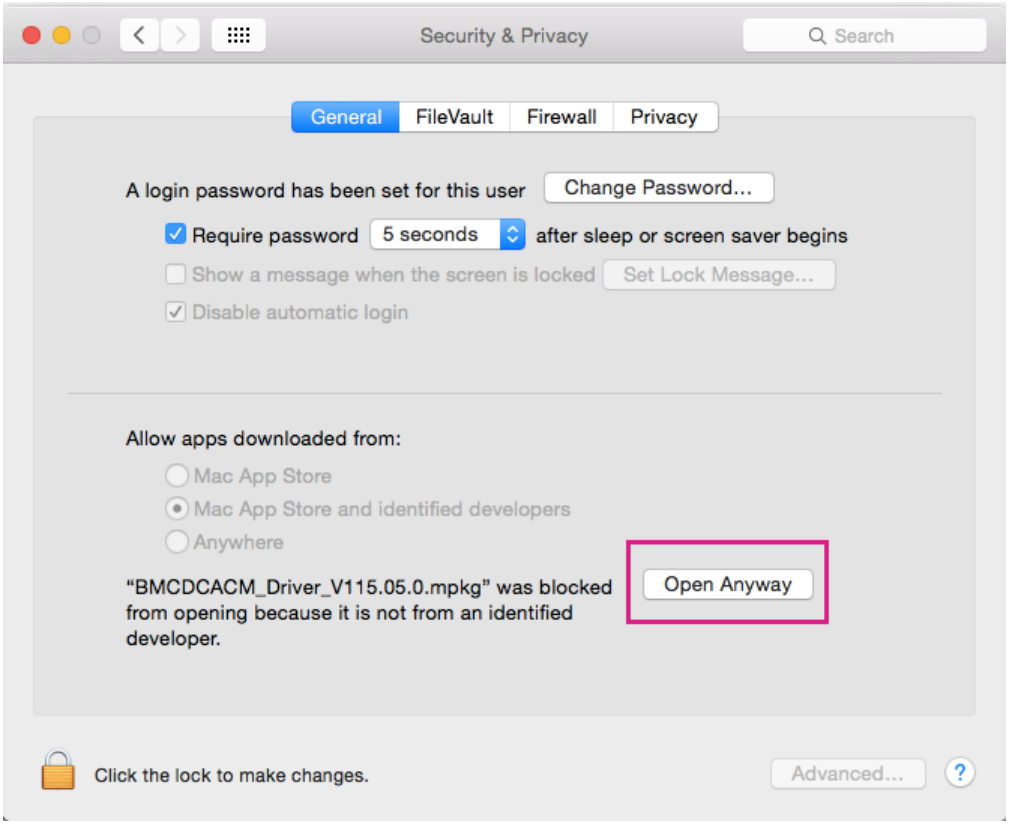

图 28 BROM USB 驱动程序安装许可配置

3) 电击上锁图标（如）并输入您的管理员密码并确认修改。

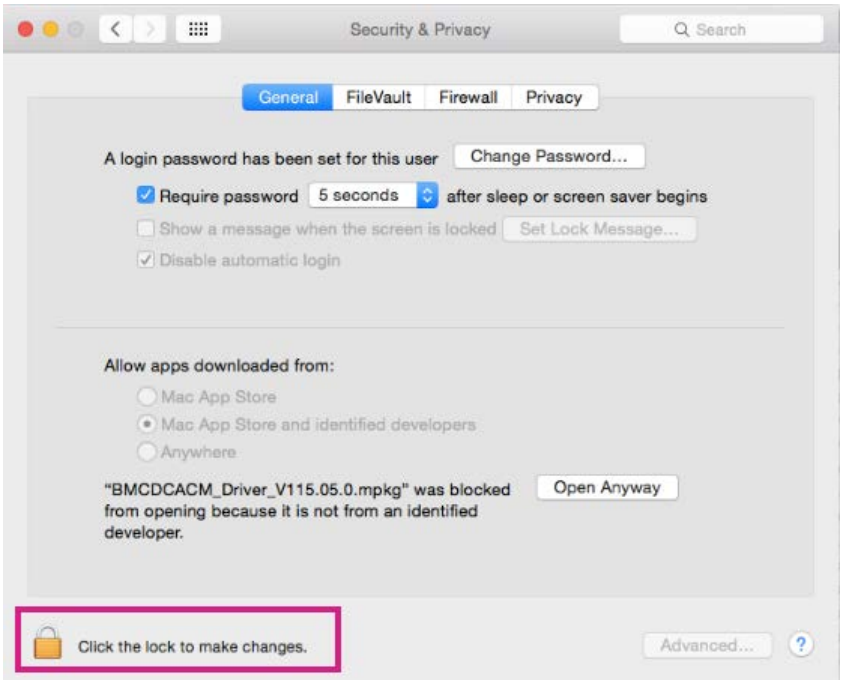

图 29 安全与隐私设定和修改确认

3.5. 既有的问题与限制

3.5.1. 模拟 I/O

LinkIt ONE 开发板有三个模拟输入（D0、D1、D2），而 Arduino UNO 有 6 个。这三个模拟输入无法用于数字 I/O，然而 Arudino UNO 可以。模拟输入的参考电压固定为 5V，然而 UNO 有数个选项。

3.5.2. 数字 I/O

LinkIt ONE 的驱动电流较小，仅适合用于信号处理。用于大电流装置（如 LED、马达、继电器、等等）也许会有问题。需要转换电路来控制这些装置。

3.5.3. PWM

LinkIt ONE 有两组 PWM，Arduino UNO 有六组。LinkIt ONE 的 PWM 分辨率高达 13-bit，而 Arduino 的 PWM 为 8-bit 分辨率。

3.5.4. SPI

LinkIt ONE 仅支持 SPI master 模式，而 Arudino UNO 支持 master 与 slave 两种模式。

3.5.5. I2C

LinkIt ONE 仅支持 I2C master 模式，而 Arudino UNO 支持 master 与 slave 两种模式。

3.5.6. SD/SPI

LinkIt ONE 有 Arduino 没有的内置 SD 支持。为了启用 SD 功能，您需要将开关（jumper 已经被改成开关）切换至 SD 位置。而 SD 模式下，SPI 与相对应的 LinkIt ONE 引脚（D13、D12、D11）将无法运作。

3.5.7. GSM/GPRS

LinkIt ONE 并不支持密码(PIN)锁定的 SIM 卡。如果 GSM/GPRS 无法运作，请确认该 SIM 卡没有被密码锁定。

3.5.8. 中断

在 LinkIt ONE 的中断引脚（D2、D3）仅支持 RISING、FALLING 和 CHANGE 三种模式。并不像 Arduino UNO 的中断引脚支持包含 HIGH 与 LOW 共五种的中断模式。

4. LinkIt ONE API 指南

这章节介绍 LinkIt ONE 的 API，若您需要 API 的详细文件请前往在联发科技创意实验室网站的联发科技 LinkIt ONE API reference 。

4.1. 数字 I/O

数字 I/O API 是 LinkIt ONE API 的基础，其他的 API 与程序库都以此为基础（如高级 I/O、SPI、I2C 与 UART）。它用于传输数字信号，与传输模拟信号的模拟 I/O API 相辅相成。

数字信号有两种状态，高电位（3.3v）与低电位（0v），由静态变量 HIGH 与 LOW 来表示示。

LinkIt ONE 提供十六个数字 I/O 引脚，其中有一些引脚与其他 API 共享（见表 3）。大部分情况下，您合并的使用几个数字 I/O 引脚来建立与周边设备连线的接口。

开发板	引脚
LinkIt ONE	D0 ~ D13 SDA/SCL or D18/D19 (shared with Wire/I^2C)

表 1 数字 I/O 引脚

使用数字 I/O 引脚之前, 您需要设置引脚模式。如果不设置正确模式，可能会导致意想不到的行为。默认的引脚模式是输入模式。有关详细信息，请参考联发科技 LinkIt ONE API reference 的 pinMode() 。

4.2. 高级 I/O

高级 I/O API 综合了数字 I/O 的简单逻辑运算，以取得指定引脚产生的数字信号，或读取指定引脚的电压数值。由于逻辑处理能通过软件进行，因此能使用的引脚非常弹性，所有的数字 I/O 引脚（如表 3）都可以用于高级 I/O。

如果您使用高级 I/O 引脚，请确认该引脚模式是否设置正确。

4.3. 模拟 I/O

模拟 I/O API 利用 LinkIt 模拟数字转换器（ADC）将连续的模拟信号转换成离散的数字信号（见表 4），并由指定点模拟引脚读取，例如读取传感器的电压信号。反过来，将数字信号写入指定引脚，以此来实现将离散的数字信号转换为连续的模拟信号。

开发板	模拟输入引脚	模拟输出引脚
LinkIt ONE	ADC0, ADC1, ADC3	D3, D9

表 2 模拟 I/O 引脚

请注意：

- `analogWrite()` 用 PWM 驱动硬件模拟输出以实现模拟信号输出。
- `analogWriteAdvance()` 用可微的调参数以达到更精确的 PWM 控制。

LinkIt ONE 开发板具备三个模拟输入（A0 到 A2），不同于 Ardiuno UNO 的六个输入（A0 到 A5）。虽然 LinkIt ONE 以 3.3V 运作，但模拟输入仍可接受 0 至 5V 的输入并转换成 0 至 1023 的数字。

LinkIt 提供两组 PWM 输出（D3、D9），不同于 Arduino UNO 的六组输出（D3、D5、D6、D9、D10、D11）。然而 LinkIt 的 D3 与 D9 输出支持高至 13-bit 的分辨率。分辨率影响输出频率，如 1.6kHz 支持 13-bit 分辨率、50.8kHz 支持 8-bit 分辨率、800kHz 支持 4-bit 分辨率。

4.4. 串行

串行 API 用来实现与电脑或其他装置的交换数据。用 `begin()`与远端装置设定好一样的波特率，并用 `read()`与 `write()`交换数据。

LinkIt ONE 提供两组 UART，实体 UART(引脚 0 和 1)与 USB UART。当一个 LinkIt ONE 开发板经 USB 连接至电脑，在 IDE 看到的 COM port 即是 USB UART，而不是实体 UART(引脚 0 和 1)。该串口会在计算机的装置管理中被显示成 **MTK USB modem Port**。

Serial Object 对应于 USB UART。用 Serial1 Object 以读取与写入实体 UART(引脚 0 和 1)，如表 5。

UART	对应的串行 Object
USB UART	Serial
UART 0/1	Serial1

表 3 UART 与串行 Object 的对应表

4.5. 时间

时间 API 是用来获取相对时间信息并提供暂停功能。`millis()` 从 LinkIt 开发板获取从开机到现在的时间（毫秒），`micros()` 取得同样的相对时间，但为微秒。

`delay()` 根据输入的毫秒暂停程序，`delayMicroseconds()` 有一样的功能但为微秒。

4.6. 中断

外部中断 API 设置了中断服务程序(ISR)，当中断信号到达中断引脚（见表 6）时，中断服务程序会被自动调用。中断服务不能抢占执行。当下一个中断来临，它会等待上一个中断结束后才会执行。

`interrupts()` 与 `noInterrupts()` 启用与禁止中断，如果中断被禁止，请尽可能在最短时间内解除禁止。

开发板	int.0	int.1
LinkIt ONE	D2	D3

表 4 中断引脚

您不能在 ISR 执行任何 LinkIt ONE API，因为 ISR 是在比 LinkIt ONE Sketch 代码更高优先权的另外线程上运行。在执行 ISR 线程时，执行 Sketch 的线程会被挂起，等待执行 ISR 的线程结束。如果中断的结果需要执行 LinkIt ONE API，这可以通过设置一个全局变量来实现。Sketch 代码依据这一全球变量来执行所需要的 API。

`noInterrupts()`不能禁止所有的硬体中断，只能禁止来自引脚的中断。禁止中断后会让其他 API 较稳定地执行，但这并不代示这个程序不会被其他事件中断。

4.7. 数学

Math API 提供四组数学功能：

- 基础数学：提供基础数学计算与操作。
- 比特与字节：提供各式比特与字节的 `get()`与 `set()`函数。
- 随机数：伪随机数函数库能使用 `randomSeed()`以指定种子，和 `random()`以取得随机数。
- 三角函数：基于弧度的三角函数 `sin()`、 `cos()` 与 `tan()`。

4.8. 伺服

伺服 API 让 LinkIt ONE 来控制由无线电控制(RC)的伺服：一个 RC 伺服具备齿轮与转轴以准确控制其转轴至特定角度，通常为 0 至 180 度。

只有具备 PWM 能力的引脚(D3 与 D9)能使用伺服。

4.9. SPI

SPI（串行外设接口）API 提供处理器与各种周边装置在同步外部串行端口进行数据通讯。在 LinkIt ONE 上，SPI 需要以下 3 个引脚以完成通讯（见表 7）：

- **MISO**（主机入从机出）: 由从机至主机传输数据。
- **MOSI**（主机出从机入）: 由主机至从机传输数据。
- **SCK**（串行时钟）：串行时钟由主输出，用于同步主仆讯号的时序。

开发板	MOSI	MISO	SCK
LinkIt ONE	D11	D12	D13

表 5 SPI 引脚

许多从机可能同时连接至一个主机，SS(Slave Select)引脚让主机选择要沟通的从机。当一个从机的 SS 讯号为低电位，主机会开始与其沟通。

由于 SPI 通讯协议非常灵活，因此在使用 SPI 进行数据交换前请再三确认从机的设置表。只有当相关参数设置完全符合从机的要求，才能进行正常的主从数据传输。在使用新的从机之前，请先回答以下问题：

- 装置的程序是否会送出与接收"MSB first"或"LSB first"？（您能用 `setBitOrder()`设置）。
- 装置的时钟闲置时为高电位或低电位？数据传输发生在时钟周期的上升边缘或下降边缘？（您能用 `setDataMode()`设置）。
- 从机要求的传输速度为多少？（您能用 `setClockDivider()`设置）。

回答以上问题并适当的设置好参数后，您能够使用 LinkIt ONE 以控制装置了。

4.9.1. 硬件設置

SPI 与内置 SD 卡功能是互斥的。在使用 SPI 的 API 之前请确认开关在正確的位置（图 19 图 19）。请注意一旦开发板设置为 SPI，调用 SD 的 API 可能会导致 SPI 异常。

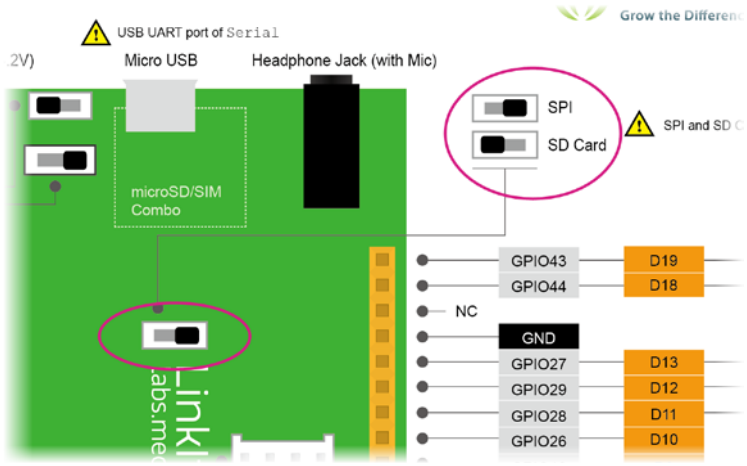

图 30 SPI/SD 开关

如果您想要同时使用 SPI 与 SD 功能，请使用外部 SD 扩充板。

在 LinkIt ONE 的处理器只能为主装置，因此其他周边装置仅可以为从装置。

4.10. 总线 Wire (I2C)

Wire 的 API 能让 LinkIt ONE 开发板与其他装置通过 I2C/TWI 总线连接，如表 8。

开发板	[I2C/TWI] 引脚
LinkIt ONE	D18 Serial Data Wire (SDA) D19 Serial Clock Wire (SCL)

表 6 I2C 引脚

LinkIt ONE 能仅以 master 身分加入 I2C 总线，不能作为 slave。

4.11. 步进

步进 API 能控制单极与多极式步进马达，用于如绘图或打印机等等需要精准定位的应用。这些马达接收脉冲信号，脉冲讯号的频率与转速成比例，在没有脉冲信号时停止。快速致动与迅速停止是其特色。您可以更改脉冲顺序以改变步进马达的旋转方向。

4.12. GSM/GPRS

GSM/GPRS API 支持：

- 发送与接收短信(SMS)。
- 通过 GPRS（2G 移动式网络）传输数据。

使用 GSM/GPRS 的应用以通过 TCP，作为客户端或服务器，连接至远端网路服务。调用 `LGPRS.attachGPRS()`以进行连网的第一步，然后：

- 创建 `LGPRSClient` 对象并调用 `LGPRSClient.connect()`以作为客户端连接至远端 TCP/IP 服务器。
- 调用 `LGPRSServer.available()`作为服务器来以侦测传入的连接申请。此方法回传一个 `LGPRSClient` 对象来代表连接。调用 `read()`与 `write()`与远端客户端交换数据。

LinkIt 能够从电信营运商自动侦测 APN 设置，也能够在程序上设置。

不支持 PIN 锁定的 SIM 卡。若您使用 PIN 锁定的 SIM 卡，请在插入 LinkIt ONE 开发板前解除 PIN 锁定。

使用 GPRS API 的范例请见在第 61 页的 5.7 “使用 GPRS 连网” 。

使用 GSM API 的范例请见在第 4.12 “GSM/GPRS” 5.1 “传送短信（SMS）”与 5.2, “接收短信（SMS）” 。

4.13. 存储器(SD/Flash)

LinkIt ONE 提供 SD 卡插槽与 10 MB 的内部闪存。您能够使用 `LSD` 与 `Lflash` 的存储 `API` 操作这两个存储区。

LSD 类别是为 LinkIt SD 卡插槽所设计。如果您想要使用 Arduino SD 扩充，请使用 Arduino 提供的 SD 函数库。这样您可以装上并使用 Arduino SD 扩充板同时也使用 LinkIt SD 卡插槽。

操作文件或文件夹，请使用 `LSD.open()` 或 `LFlash.open()`以取得 `Lfile` 对象，然后：

- 如果对象为文件（`isDirectory()`回传 `false`），使用该对象以读写文件。
- 如果对象为文件夹（`isDirectory()` 回传 `true`），使用该对象以列出文件夹中的子文件。

4.13.1. 硬件设置

板上的 SD 卡与 SPI 功能互斥，所以在使用 SD API 之前，请确认开关是否设置正确（见第 31 页的图 19）。请注意开发板设置为 SD 后，调用 SPI API 可能导致 SD 异常。如果您想要同时使用 SD 与 SPI，请使用 SD 扩充板。

4.14. 蓝牙

本软件开发包支持两种蓝牙协议：

- 蓝牙串行协议 (SPP) 经由蓝牙 2.1.
- 通用属性协议(GATT) 经由蓝牙 4.0.

4.14.1. 串行协议

LinkIt ONE 提供内置蓝牙串口协议(SPP)，提供一对一连接。请参见蓝牙开发者网页 SPP 页面了解更多信息。

蓝牙 API 使用此功能连接两个蓝牙装置并互相交换数据。该 API 提供类让

LinkIt ONE 作为客户端（`LBTClientClass`）或伺服端（`LBTServerClass`）在 SPP 连接模式中。

当 LinkIt ONE 作为客户端时，它具备以下功能：

- 扫描蓝牙装置，并连接至指定服务器装置。
- 传送与接收数据至连上的客户端装置。

当作为伺服端时，LinkIt ONE 会：

- 等待蓝牙 SPP 客户端连接才传收数据。
- 传送与接收数据至连上的伺服端装置。

使用蓝牙 API 的范例请见第 47 页的 第 5.4 章节 "安卓手机蓝牙连接至 LinkIt ONE”。

4.14.2. 通用属性协议

LinkIt ONE 也提供内置蓝牙通用属性协议(GATT)。请参见蓝牙开发者网页 GATT 页面了解更多信息。

蓝牙 API 使用此功能连接两个蓝牙装置并互相交换数据。该 API 提供类让

LinkIt ONE 作为客户端（`LGATTClient`）或伺服端（`LGATTServer`）在 GATT 连接模式中。另外，为了让您访问这类的数据交换，`LGATTService` 类提供服务细节数据和访问。

当 LinkIt ONE 作为客户端时（GAP 中心设备），它具备以下功能：

- 创建一个 GATT 客户端。
- 扫描 GATT 蓝牙协议装置，并连接至指定服务器装置。
- 传送与接收数据至连上的伺服端装置。

当作为伺服端时（GAP 外部设备），LinkIt ONE 会：

- 创建一个 GATT 伺服端并定义所提供的服务（GATT 协议兼容服务或客制服务）。
- 等待蓝牙 GATT 客户端连接才传收数据。
- 传送与接收数据至连上的客户端装置。

使用蓝牙 API 的范例请见 `LGATTSUart`， 该范例教您如何用 TX 和 RX 定义伺服端的服务类似串口协议。

4.15. GPS

LinkIt ONE 具备内置 GPS 装置，您能够通过 GPS API 取得装置的 GPS 数据。

控制 GPS 基础流程如下：

- `powerOn()`: 供电 GPS。
- `setMode()`: 设置工作模式 (可选)。
- `getData()`:查询并处理 GPS 数据。
- `powerOff()`: 断电 GPS。

由 `getData()`回传的数据可能为 GPGGA、GPGSA、GPRMC、GPVTG、GPGSV、GLGSV、GLGSA、BDGSV 与 BDGSA。皆为标准 NMEA 信息种类。解析后可得 GPS 位置、时间与其他细节。

NMEA 详情请参阅 http://www.gpsinformation.org/dale/nmea.htm

使用 GPS API 的范例请见第 61 页的 5.6 章节“使用 GPRS 连网”，第 57 页的 5.5 章节 “使用 GPS” 。

4.16. Wi-Fi

Wi-Fi API 使用 LinkIt ONE 内的内置 Wi-Fi 模块扫描并连接 Wi-Fi AP。具备了此 Wi-Fi 连接能力，LinkIt ONE 能够获得各种网路资源，包含一系列的 Web 服务。

Wi-Fi 功能分为以下三种类别：

- LWiFi 能扫描与连接 Wi-Fi AP，包含：
- begin()以启用 Wi-Fi 模块：
  - connect()以连接至无加密的 Wi-Fi AP 。
- connectWEP()与 connectWPA()连接至加密的 AP。
- LWiFiClient 能通过 TCP 协议连接至网际网路服务，包含：
- connect() 开启 TCP/IP 连接至 Web 服务器并读取内容。
- 您的应用能包含多达 7 个客户端。
- LWiFiServer 能设置 TCP 端口并聆听远端 TCP 客户端，包含：
  - available()检查传入的连接，此 API 回传一个 LWiFiClient()对象代表连接。
- client.read() 与 client.write() 与远端客户端交换数据。

对于 LwiFiClient 与 LWiFiServer，print() 与 write()方法可用于实现流接口。虽然传送 ASCII 字串或使用 print()将数值转换成字串较为简单，您仍能使用 write()以传送低阶原始缓冲区内容。print()是由 write()来实现的。

Wi-Fi API 目前有以下既有限制：

- 您连接至 WEP 加密的 AP 时，不能指定组 ID 的密码。
- 不支持询问 AP 加密协议。
- 不支援静态网路设置；也就是 static IP、DNS 服务器或子网掩码。而是用 DHCP 来提供网路设置。

使用 Wi-Fi API 的范例请见第 44 页的 5.3, “使用 Wi-Fi 连网”。

4.17. 音频

LinkIt ONE 支持存储在 SD 或内置闪存中 AMR、MP3 与 AAC 音频文件的播放。通过 LinkIt ONE 开发板的耳机插孔输出，见图 31。

图 31 LinkIt 的耳机插孔

音频 API 具备播放、暂停与停止播放、与调整音量的功能。

播放（解码）在 LinkIt ONE 内部处理，因此所有功能都是非阻塞的。当你调用 `playFile()`后会播放音频并马上返回至您的程序，您能够使用 `getStatus()`检查播放状态。

如果您尝试在上一个完成前播放其他音频文件，上一个会停止并播放新的音频文件。

4.18. 电池

电池 API 提供：

- 电量 API 回传当下电量百分比(0-100)。
- 充电状态 API 回传电池是否在充电中。

4.19. 日期时间

RTC (Real Time Clock) API 具备设置与取得日期与时间功能。提供了别于 `millis()`的长时间（月与年）量测，因为 `millis()`会在 50 天后溢出并重置。

只要供电 `LinkIt ONE`（由 USB 连线或电池）时钟就会持续运行。当断电或开发板重启，当前时间可以用 `GPS.getData()`解析出的 GPGGA 或由 Wi-Fi 连接的网路时间协议(NTP)来设置。

4.20. EEPROM

LinkIt ONE 提供内置电可擦除可编程只读存储器(EEPROM)。存储在 EEPROM 的数据即使开发板断电仍会保存。EEPROM 的 API 提供 EEPROM 读写功能。在 LinkIt ONE 的 EEPROM 能存储达 1,024 字节。

4.21. 数据类型的大小

表 9 显示 LinkIt ONE 与 Arudino API 中可用的各个数据类型大小。

数据类型	LinkIt ONE SDK	Arduino SDK
boolean	1 byte	1 byte
byte	1 byte	1 byte
char	1 byte	1 byte
short	2 bytes	2 bytes
word	4 bytes	2 bytes
int	4 bytes	2 bytes
long	4 bytes	4 bytes
float	4 bytes	4 bytes
double	8 bytes	4 bytes

表 7 LinkIt 与 Arduino SDK 中变量的大小

5. 使用 LinkIt ONE 的 API

这个章节提供使用各种 LinkIt ONE API 的导引，以执行 LinkIt ONE 开发板的常见任务。

5.1. 传送短信（SMS）

以下描述如何调整您的 LinkIt ONE 开发板与程序以传送短信（SMS）。

5.1.1. 硬件设置

请依照以下步骤准备好您的 LinkIt ONE 开发板（如图 32）：

1) 插入标准大小的 SIM 卡到开发板背面的 SIM 插槽。如果您使用 mirco 或 nano 的 SIM 卡，则需要一个适配器。由于 LinkIt ONE SDK 不支持 PIN 锁定的 SIM 卡，所以也请确认 SIM 卡没有 PIN 锁定。
2) 将 GSM/GPRS 天线接至天线接口。

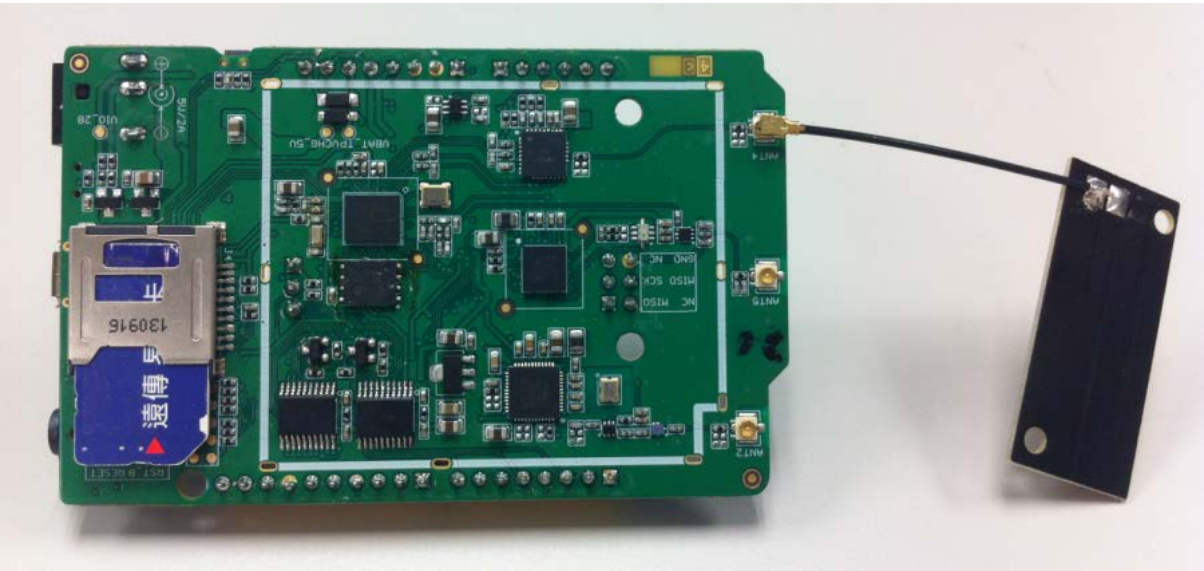

图 32 插入 SIM 卡并连接 GSM 天线的 LinkIt ONE 开发板

5.1.2. 软件设置

此章节介绍创建接收 SMS 代码的必须步骤：

5.1.2.1. 包含 GSM 函数库

您应该在您的程序中包含 GSM 函数库。请在您程序的 Arduino IDE 操作：点击 **Sketch** 选单并指向 **Import Library** 并点击 **LGSM。**您就会看到您的 Sketch 包含了 GSM 头文件。

```
#include <LGSM.h>
```

您现在可以在 Sketch 使用 LSMS 物件来执行 SMS 相关任务。LSMS 的函数细节请参阅 LinkIt API 参考手册。

5.1.2.2. 等待 SMS 的初始化

SIM 卡是相对缓慢的装置，使用前需要几秒的初始时间。您可以使用 LSMS 来确认 SMS 就绪与否，如果还没就绪就再等一秒确认。

通常 SIM 需要 5～10 秒的起始时间，如果需要超过 30 秒，请确认 SIM 的安装状况。

```
while(!LSMS.ready())
{
  delay(1000);
}
```

5.1.2.3. 传送 SMS！

传送 SMS 短信需要收信号码与短信内容。您需要三个 API：`beginSMS()`、`print()`与 `endSMS()`。

1) 使用 `beginSMS()` 以指定一个或更多收信号码。
2) 使用 `print()`以编写 SMS 短信内容。
3) 使用 `endSMS()`以传送短信。

以下片段代码假设您已经有收信号码与短信内容 - 通常您会在 Sketch 程序的其他地方设立号码与内容。

```
LSMS.beginSMS("0123456789");
LSMS.print("Hello from LinkIt!");
LSMS.endSMS();
```

您可以测试 `endSMS()`的返回值以确定 SMS 有无成功送出，失败会回传 0(见 5.1.2.4, “完整代码” 以了解如何做到这点)，反复确认 SIM 卡与 GSM 天线有无安装正确。如果仍然失败，请确认 SIM 卡没有锁定并确认插入其他行动装置时也能送出 SMS。

5.1.2.4. 完整代码

以下为完整 Sketch 代码：

```
#include <LGSM.h>

void setup() {
  Serial.begin(9600);

  while(!LSMS.ready())
    delay(1000);

  Serial.println("SIM ready for work!");

  LSMS.beginSMS("0123456789");
  LSMS.print("Hello from LinkIt");
  if(LSMS.endSMS())
  {
    Serial.println("SMS is sent");
  }
  else
  {
    Serial.println("SMS is not sent");
  }
}

void loop()
{
  // do nothing
}
```

5.2. 接收短信（SMS）

本章节介绍如何设置您的 LinkIt ONE 开发板以及必须的程序以接收短信。

5.2.1. 硬件安装

1) 插入标准大小的 SIM 卡到开发板背面的 SIM 插槽。如果您使用 mirco 或 nano 的 SIM 卡，则需要一个适配器。由于 LinkIt ONE SDK 不支持 PIN 锁定的 SIM 卡，所以也请确认 SIM 卡没有 PIN 锁定。

2) 连接 GSM/GPRS 天线到天线接口。

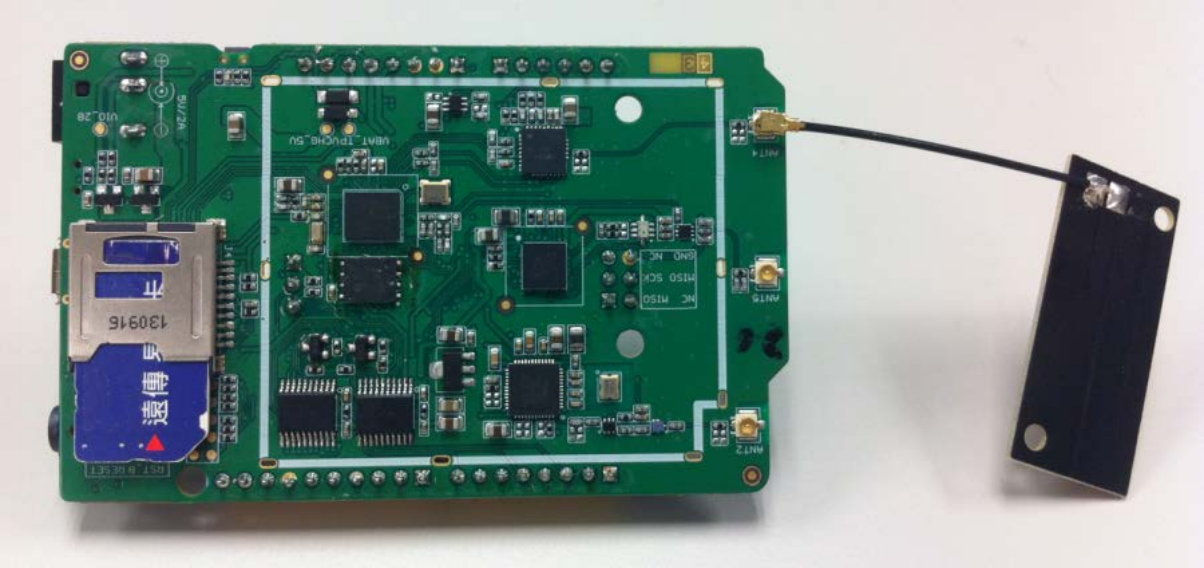

图 33 插入 SIM 卡并连接 GSM 天线的 LinkIt ONE 开发板 Software setup

5.2.2. 软件设置

本章节介绍建立软件来接收 SMS 的必要步骤。

5.2.2.1. 包含 GSM 函数库

您应该在您的程序中包含 GSM 函数库。请在您程序的 Arduino IDE 中：点击 **Sketch** 选单并指向 **Import Library** 并点击 **LGSM。**您就会看到您的 Sketch 包含了 GSM 头文件。

```
#include <LGSM.h>
```

您现在可以在 Sketch 使用 LSMS 对象来执行 SMS 相关任务。LSMS 的特色请参阅 LinkIt ONE API 参考手册。

5.2.2.2. 等待 SIM 的初始化

SIM 卡是相对缓慢的装置，使用前需要几秒的初始时间。您可以使用 LSMS 来确认 SMS 就绪与否，如果还没就绪请等一秒再确认一次。

通常 SIM 需要 5～10 秒的起始时间，如果需要超过 30 秒，请确认 SIM 的安装状况。

```
while(!LSMS.ready())
{
  delay(1000);
}
```

5.2.2.3. 查询 SMS 是存在

LSMS 的 avaiable() API 能确认板上是否有 SMS。您应该会将以下代码放入 loop()以定期查询板上是否有 SMS。

```
void loop()
{
  if(LSMS.available())
  {
    // continue to display
  }
  delay(1000);
}
```

5.2.2.4. 读取 SMS 信息的内容

首先提供输出缓存与缓存大小参数到 remoteNumber()以找寻寄信号码（寄 SMS 的号码）。您需要的号码与缓存长度由您所在国家的号码长度决定，20 字符应该包含绝大部分地区了。

第二步是使用 read()查询短信内容。信息内容的长度是不定的。因此 read()从串流中每次一个字节地读取内容，直到读到一个负值。

```
char buf[20];
LSMS.remoteNumber(buf, 20); // number is stored into buf

int c;
while(true)
{
  c = LSMS.read();          // message content (one byte at a time)
  if(c < 0)
    break;                  // enf of message content
}
```

5.2.2.5. 删除 SMS

在号码跟内容都读取后，应该删除信息才能收取下一笔短信。使用 `flush()`，就可以删除先前由 `available()`查询过的信息。

```
LSMS.flush()
```

5.2.2.6. 完整代码

以下是完整的 Sketch 代码：

```
#include <LGSM.h>

void setup() {
  Serial.begin(9600);

  while(!LSMS.ready())
    delay(1000);

  Serial.println("SIM ready for work!");

}

void loop()
{
  char buf[20];
  int v;
  if(LSMS.available())              // Check if there is new SMS
  {
    Serial.println("There is new message.");

    LSMS.remoteNumber(buf, 20);     // display Number part
    Serial.print("Number:");
    Serial.println(buf);

    Serial.print("Content:");       // display Content part
    while(true)
    {
      v = LSMS.read();
      if(v < 0)
        break;
      Serial.print((char)v);
    }
    Serial.println();

    LSMS.flush();                   // delete message
  }

  delay(1000);
}
```

5.3. 使用 Wi-Fi 连网

以下描述如何设置您的 LinkIt ONE 开发板与所需的代码连接至 Wi-Fi AP（Access Point）并检索网页内容。

5.3.1. 硬件设置

将 Wi-Fi 天线接至天线接口（如图 34）来准备好您的 LinkIt ONE 开发板。

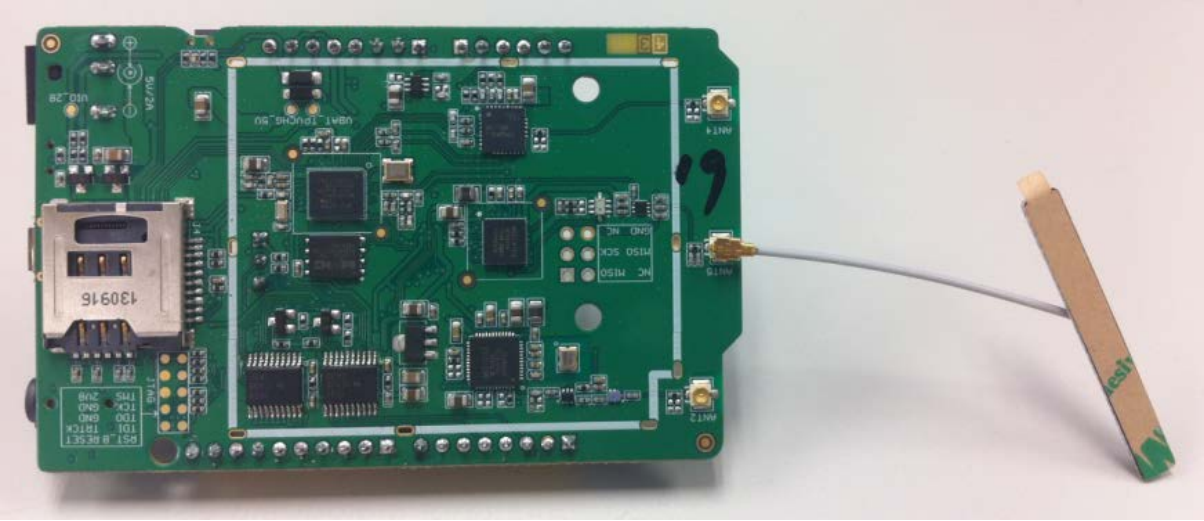

图 34 接上 Wi-Fi 天线的 LinkIt ONE 开发板

5.3.2. 软件设置

本章节介绍创建代码的必须步骤，以达到安装 Wi-Fi 连接和检索网页内容的目的。

5.3.2.1. 包含 Wi-Fi 函数库

您应该在您的程序中包含 Wi-Fi 函数库。请在您程序的 Arduino IDE 操作：点击 **Sketch** 选单并指向 **Import Library** 并点击 **LWiFi**。您就会看到您的 **Sketch** 包含了 Wi-Fi 头文件。LinkIt 在这个手册为 Wi-Fi 客户端，所以保留前两个头文件并删去其余的。

```
#include <LWiFi.h>
#include <LWiFiClient.h>
```

您现在可以在您的 Sketch 使用 `LWiFi` 与 `LwiFiClient` 对象来执行 Wi-Fi 相关任务。`LWifi` 与 `LwiFiClient` 的详细功能请参阅 LinkIt API 参考手册。

5.3.2.2. 连接至 Wi-Fi AP

首先使用 LWiFi.begin()以启动 Wi-Fi API。连接 Wi-Fi AP 的方式有许多种，您的 AP 如果没有加密可以使用 LWiFi.connect()，否则请确认使用哪种加密。LinkIt 支持 WEP 与 WPA 加密。对应 WEP 加密，请用 LwiFi.connectWEP()；对应 WPA 加密，请用 LwiFi.connectWPA()。

```
#define WIFI_AP "Name_of_your_AP"
#define WIFI_PWD "Password_of_your_AP"

LWiFi.begin();
LWiFi.connect(WIFI_AP);                // if the AP is not encrypted
LWiFi.connectWEP(WIFI_AP, WIFI_PWD);  // if the AP uses WEP encryption
LWiFi.connectWPA(WIFI_AP, WIFI_PWD);  // if the AP uses WPA encryption
```

如果连接失败 connect()会回传负值。如果此情况发生，请确定是否正确连接天线并且 LinkIt ONE 在目标 AP 的范围内。

5.3.2.3. 连接到网站

第二步是连接到网站，由 LwiFiClient 对象达成。一旦连接到 Wi-Fi AP，能同时建立七个连接。在此范例仅需一个连接。 connect()的第一个参数为您想要连接的 URL，第二个参数为接口；HTTP 使用接口 80。

```
#define SITE_URL "www.mediatek.com"

LWiFiClient c;
c.connect(SITE_URL, 80);
```

5.3.2.4. 传送 HTTP 请求

建立连接后，就像串流一样，您可读取与写入。为检索网页内容，您需要传送 HTTP GET 请求，如下面的片段代码所示：

```
c.println("GET / HTTP/1.1");
c.println("Host: " SITE_URL);
c.println("Connection: close");
c.println();
```

5.3.2.5. 取得网页内容

如果一切正常，远端网路服务器会回应您的 GET 请求并开始回传内容。您接下来可以从连接对象读取数据：

```
int v;
while(c.available())
{
  v = c.read();      // return one byte at a time
  if(v < 0)
    break;           // no more data
}
```

5.3.2.6. 完整代码

这是完整的 Sketch 代码：

此代码将检索的数据打印在波特率 9600 的串行接口。

```
#include <LWiFi.h>
#include <LWiFiClient.h>

#define SITE_URL "www.mediatek.com"
#define WIFI_AP "Name_of_your_AP"          // replace with your setting
#define WIFI_PWD "Password_of_your_AP"     // replace with your setting

LWiFiClient c;

void setup() {

  Serial.begin(9600);
  LWiFi.begin();

  Serial.println();
  Serial.print("Connecting to AP...");
  if(LWiFi.connectWEP(WIFI_AP, WIFI_PASSWORD) < 0)
  {
    Serial.println("FAIL!");
    return;
  }
  Serial.println("ok");

  Serial.print("Connecting to site...");
  if(!c.connect(SITE_URL, 80))
  {
    Serial.println("FAIL!");
    return;
  }
  Serial.println("ok");

  Serial.println("send HTTP GET request");
```

```
    c.println("GET / HTTP/1.1");
    c.println("Host: " SITE_URL);
    c.println("Connection: close");
    c.println();
  }

  void loop() {
    int v;
    while(c.available())
    {
      v = c.read();
      if(v < 0)
        break;
      Serial.print((char)v);
    }
    delay(100);
  }
```

5.4. 安卓手机蓝牙连接至 LinkIt ONE

以下描述如何设置 LinkIt ONE 开发板与需要的代码，同安卓手机以蓝牙 SPP 规范交换数据。在这个指导 安卓手机为主；而 LinkIt ONE 为仆。

5.4.1. 硬件设置

请连接 Wi-Fi 天线（Wi-Fi 与蓝牙共用同一天线）至天线接口以准备好您的 LinkIt ONE 电路板（如图 35）。

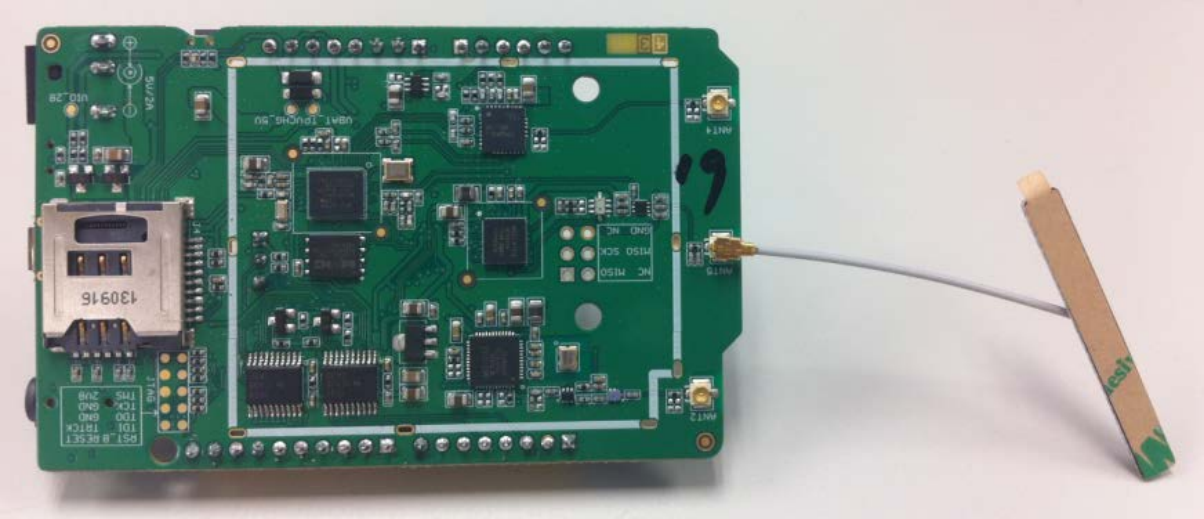

图 35 接上天线的 Wi-Fi/蓝牙天线的 LinkIt ONE 开发板

5.4.2. 软件设置（LinkIt ONE 端）

以下描述创建代码的必须步骤，以设置蓝牙服务器并允许安卓手机连接至 LinkIt ONE。

5.4.2.1. 包含蓝牙函数库

您应该在您的代码内包括蓝牙(BT)函数库。请在您程序的 Arduino IDE 操作：点击 **Sketch** 选单并指向 **Import Library** 并点击 **LBT**。您就会看到您的 Sketch 包含了 BT 头文件。在这个手册中 LinkIt 作为 BT 服务器，所以仅需要 LBT 与 LBTServer 头文件，其他可以移除。

```
#include <LBT.h>
#include <LBTServer.h>
```

您现在可以用 LBTServer 对象来执行 Sketch 中与 BT 相关任务。LBTServer 相关的详细信息，请参阅 LinkIt ONE API 参考手册。

5.4.2.2. 开启蓝牙服务器

您需要一个特定名称，蓝牙客户端才能辨识您的可用装置并且连接。

```
LBTServer.begin((uint8_t*)"My_BTServer")
```

5.4.2.3. 等待客户连接

您需要使用 accept()让客户能够连接，它的参数是以秒为单位的超时值。在此指导里，LinkIt ONE 会持续等待至连接为止。而连接完成后，LinkIt ONE 会切换到数据交换模式。

```
void loop() {
  if(LBTServer.connected())
  {
    // There is active connection
  }
  else
  {
    // Wait 5 secs and retry forever
    LBTServer.accept(5);
  }
}
```

5.4.2.4. 与连上的客户交换数据

连接完成之后，使用 read()从客户读取数据并用 write()将数据写入客户。在此指导里，LinkIt ONE 单纯作为一个“回响”机：会将所有从客户读到的数据再写回去。

```
uint8_t buf[64];
int bytesRead;

while(true)
{
  bytesRead = LBTServer.read(buf, 64); // read from client
  if(!bytesRead)
    break;
  LBTServer.write(buf, bytesRead);     // write the same data back to
client
}
```

5.4.2.5. 完整代码

此为完整 Sketch 代码：

```
#include <LBT.h>
#include <LBTServer.h>

void setup() {
  Serial.begin(9600);

  if(!LBTServer.begin((uint8_t*)"My_BTServer"))
  {
    Serial.println("Fail to start BT.");
    return;
  }

  Serial.println("BT server is started.");
}

void loop() {
  uint8_t buf[64];
  int bytesRead;

  if(LBTServer.connected())
  {
    // echo back all received data
    while(true)
    {
      bytesRead = LBTServer.readBytes(buf, 64);
      if(!bytesRead)
        break;
      Serial.write(buf, bytesRead);
      LBTServer.write(buf, bytesRead);
    }
    delay(100);
```

```
    }
    else
    {
      LBTServer.accept(5);
    }
  }
```

5.4.3. 软体设置（安卓端）

以下描述创建连接并与 LinkIt ONE 沟通的安卓应用程序的步骤：

5.4.3.1. 取得蓝牙聊天室源代码

安卓 SDK 包含名为“BluetoothChat”的范例应用程序，允许两个安卓装置通过蓝牙连接互相沟通，仅需一些更改即可与 LinkIt ONE 装置沟通。

源代码位于安卓 SDK 范例代码文件夹。如果您标准安卓 SDK，安装范例可以在此找到：`C:\Program files\Android\android-sdk\samples\android-16\BluetoothChat`.

5.4.3.2. 修改范例以与 LinkIt ONE 沟通

使用我们的 LinkIt ONE 代码让应用程序与 LinkIt ONE 开发板沟通，您需要做两个小修改：

1) 在 `AndroidManifest.xml` 修改
   - `<uses-sdk … />`
     - 变成：
   - `<uses-sdk android:maxSdkVersion="17" android:targetSdkVersion="11" android:minSdkVersion="11" />`
2) 在 `src/com/example/android/BluetoothChat/BluetoothChatService.java` 更改 SPP profile 的 UUID 设置至如下：

```
// Unique UUID for this application
private static final UUID MY_UUID_SECURE =
    UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
private static final UUID MY_UUID_INSECURE =
    UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
```

5.4.3.3. 构建此应用程序

现在即可构建这个应用程序，请参阅 Android Developer website 以得到如何创建安卓范例程序信息。

5.4.3.4. 测试蓝牙通讯

在 LinkIt ONE 上启动 Sketch，并安装且执行范例程序至您的安卓手机以开始测试蓝牙通讯。然后：

1) 点击范例程序中左边的“搜索”按钮（如图 36），这会列出所有发现过的蓝牙装置，并应该列出在您在 LinkIt ONE 设置名为 **My_BTServer** 的蓝牙服务器。

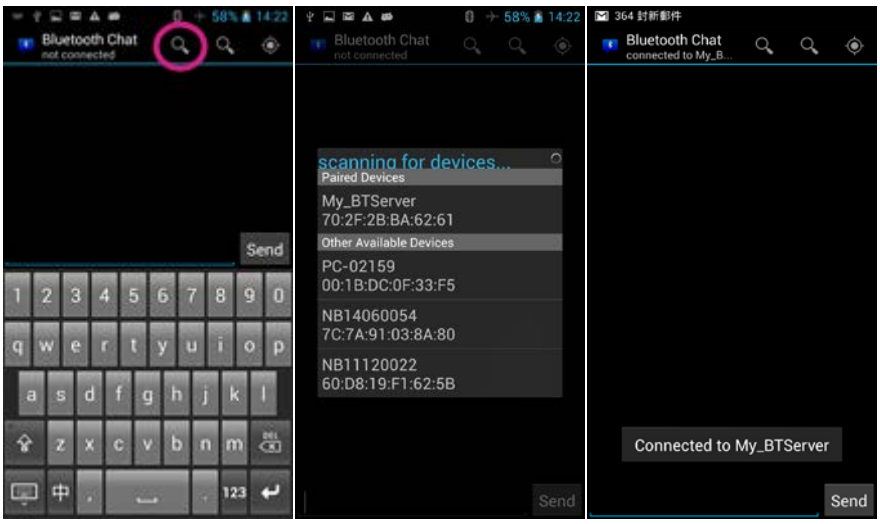

图 36 连接至 LinkIt ONE 蓝牙服务器

2) 选取 **My_BTServer** 后会显示如图 37 的 **Connected to My_BTServer** 的信息。

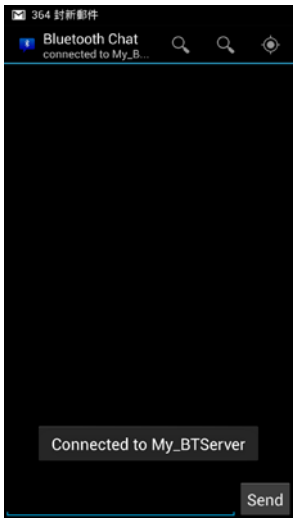

图 37 成功连接至 LinkIt ONE 蓝牙服务器

3) 现在能输入送至 LinkIt ONE 的内容，您能看到由 LinkIt ONE 回传值，如图 38。

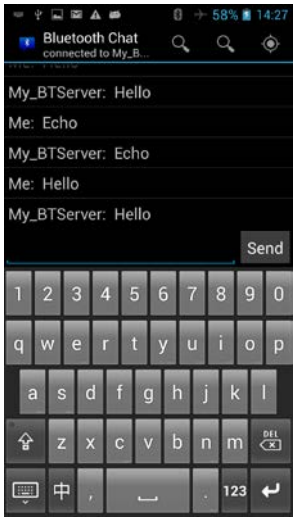

图 38 数入的文字会由 LinkIt ONE 回显

5.5. 使用蓝牙 GATT 协议

根据您的需求，您可以设定 LinkIt ONE 作为连接其他蓝牙 4.0 设备的核心点，并访问它们的个人资料和服务，或设定 LinkIt ONE 作为外部设备，提供资料和服务给其他设备使用。

5.5.1. 硬件设置

5.5.2. 将一个 Wi-Fi 天线接上天线连接器，如图 34。准备好您的 LinkIt ONE 开发板（Wi-Fi 和蓝牙共享同一个天线）。执行一个通用属性协议 (GATT) 伺服器

将 LinkIt ONE 开发板定义成一个 GATT 伺服器，您需要定义一个 GATT 伺服协议。调用 LGATTServer 的 begin()method。

GATT 协议由一个或超过一个服务，请参见蓝牙开发者网站的 GATT 网页了解更多信息。以下范例创建两个服务的协议， begin() method 会要求服务的数量，和子类 LGATTService 的 instance, 它定义所有服务的实际行为，如下面的代码实例。

```
#include <LGATTServer.h>

void setup() {
  LGATTServer.begin(2, &myServiceA, &myServiceB);
}
```

GATT 服务应该处理来自 GATT 客户的要求，其包含服务的读取和写入请求到特性要求。然后这些请求被传递到服务实例作为事件回调，如 LGATTService.onRead（ ）。因此，GATT 伺服器应定期检查传入的请求，并分派事件服务对象。为了处理传入的事件，调用 LGATTServer.handleEvents（ ）在你的 Arduino 草图的 loop（ ）函数。

```
void loop() {
  LGATTServer.handleEvents();
}
```

要定义服务的话就要继承 LGATTService 类，并实施处理传入事件的方法。你不必直接调用这些方法，因为这些方法都是由 GATT 库调用当 LGATTServer.handleEvents（ ）被调用。下面的代码显示如何通过定义一个类继承 LGATTService 类，并覆盖了 onLoadService 方法：

```
class LGATTSample : public LGATTService {
public:
    virtual LGATTServiceInfo *onLoadService(int32_t index);
};
static LGATTServiceInfo MY_SERVICE_DEFINITION [] = {
    {TYPE_SERVICE, "6e400001-b5a3-f393-e0a9-e50e24dcca9e", TRUE, 0, 0, 0},
    {TYPE_CHARACTERISTIC, "6e400002-b5a3-f393-e0a9-e50e24dcca9e", FALSE,
     VM_GATT_CHAR_PROP_WRITE, VM_GATT_PERM_WRITE, 0},
    {TYPE_CHARACTERISTIC, "6e400003-b5a3-f393-e0a9-e50e24dcca9e", FALSE,
     VM_GATT_CHAR_PROP_NOTIFY | VM_GATT_CHAR_PROP_INDICATE,
VM_GATT_PERM_READ, 0},
    {TYPE_DESCRIPTOR, "00002902-0000-1000-8000-00805f9b34fb", FALSE,
     VM_GATT_CHAR_PROP_NOTIFY, VM_GATT_PERM_READ | VM_GATT_PERM_WRITE, 0},
    {TYPE_END, 0, 0, 0, 0, 0}
};
LGATTServiceInfo* LGATTSample::onLoadService(int32_t index){
    return MY_SERVICE_DEFINITION;
}
```

onLoadService（ ）类会比任何其他方法之前被调用。正如从以上代码中显示，这个方法应该返回 ard_gatts_service_decl_struct 数组，介绍有关服务和所有的特征和描述符的信息。但请注意，此结构必须保持不变，直到 LGATTServer.end（ ）被调用。因此，你应该避免在局部范围的数据声明结构数组。无论是把它定义为一个全局变量，或者分配在堆上。返回的数组必须先从 TYPE_SERVICE 的元素，并有自己的 UUID。请参阅 API 文档中的元素等领域。

```
{TYPE_SERVICE, "6e400001-b5a3-f393-e0a9-e50e24dcca9e", TRUE, 0, 0, 0},
```

所述阵列由一个或多个特性的条目延续，跟在后面的是零或多个描述条目。再次提醒，每个条目都有自己的 UUID 和不同的读/写权限和属性。

```
{TYPE_CHARACTERISTIC, "6e400002-b5a3-f393-e0a9-e50e24dcca9e", FALSE,
VM_GATT_CHAR_PROP_WRITE, VM_GATT_PERM_WRITE, 0},
    {TYPE_CHARACTERISTIC, "6e400003-b5a3-f393-e0a9-e50e24dcca9e", FALSE,
VM_GATT_CHAR_PROP_NOTIFY | VM_GATT_CHAR_PROP_INDICATE, VM_GATT_PERM_READ,
0},
    {TYPE_DESCRIPTOR, "00002902-0000-1000-8000-00805f9b34fb", FALSE,
VM_GATT_CHAR_PROP_NOTIFY, VM_GATT_PERM_READ | VM_GATT_PERM_WRITE, 0},
```

要标明数组末尾，请放置 TYPE_END 元素如下：

```
{TYPE_END, 0, 0, 0, 0, 0}
```

该 LGATTServer 类将解析这个数组并生成相应的内部资源。然后，它调用下面的方法来通知这些内部资源的柄。您应该保存这些柄，因为在连接中央设备时会需要被读取或写入属性。

- onCharacteristicAdded（）：该方法在 onLoadService（）之后调用并传柄给服务结构中定义的每个特征。这些柄应该在调用 send（），onRead（）或 onWrite（）方法时使用。
- onDescriptorAdded(): 类似 onCharacteristicAdded(), 但是调用给每个数组的说明元素当 onLoadService() 返回时。

当 LGATTServer.begin（）返回后，onLoadService（），onCharacteristicAdded（）和 onDescriptorAdded（）已全部调用，LinkIt ONE 就会开始侦听符合您定义的中央设备。如果发现传入连接，onConection（）将被调用而 data.connected 字段将设置为 true。如果与中央装置连接断开，onConection（）奖被调用与 data.connected 字段设置为 false。

如果中央设备是连接的并且尝试读取某一特性或描述符，onRead（）将被每个连接中央装置的读取请求被调用。你应该调用 ackOK（）方法 onRead（）里面执行所请求的属性值传送回中央设备。确认（ACK）将首先被发送，然后由特性或描述符的值，如下所示：

```
boolean LGATTSample::onRead(LGATTReadRequest &request){
        LGATTAttributeValue value = {0};
        const char *str = "value string";
        memcpy(value.value, str, strlen(str));
        value.len = strlen(str);
        request.ackOK(value);
}
```

您应该为每个请求最多只需要调用 request.ackOk()一次，

onWrite（ ）的用法类似，但发送 ACK 返回到中央装置时不是发送数据，则应该基于中央装置发送的值更新设备状态。

```
boolean LGATTSample::onWrite(LGATTWriteRequest &data){
      // if need to rsp to central.
      if (data.need_rsp){
            data.ackOK(); // Send ACK
      }
      // Update device internal status by the written data according.
      // Here we simply print the length of received data.
      Serial.print("Received data length:");
      Serial.println(data.value.len);
      return true;
}
```

有一点要注意的是，在 LinkIt ONE 的每笔交易您只能读取和写入 20 个字节数据的限制。如果服务定义值的长度超过 20 个字节的属性，客户端必须多次读取或写入 offset，来表明该属性值的那一个部分以进行读或写。

5.5.3. 执行一个通用属性协议 (GATT) 客户端

实现一个客户端 GATT 协议（中央设备）涉及发现附近的外设充当 GATT 协议伺服器，并列举他们的个人资料。因此，访问 GATT 伺服器协议包括以下步骤：

- 扫描外部 GATT 蓝牙装置并取得它们的资料。
- 连接至指定的外部装置。
- 读或写装置提供的服务特征值。

以下范例显示如何扫描附近的 GATT 装置。首先开启 GATT 客户，用 LGATTClient.begin()来实力化一个对象，然后传递辨别您应用的 UUID。该 UUID 可由合时的生成器产生。注册可能须要一阵子来完成。调用 scan()来扫描附近的 GATT 装置和取回信息包含 RSSI 值，用来表示两个装置的距离。

```
#include <LGATTClient.h>
LGATTUUID appUUID("AFA5B1C5-2B7B-471B-BD4C-7A92DE9D6DBD");
LGATTClient client;
void setup() {
    client.begin(appUUID);
    int numberOfDevices = client.scan(3);
    LGATTDeviceInfo info = {0};
    for(int i = 0; i < numberOfDevices; ++i){
        client.getScanResult(i, info);
    }
}
```

描后到 GATT 装置后，您就能与它们连接和发现它们提供的服务。例如，如果您想要和心率配置文件连接（HRP），您需要两个服务， 一个是心率服务（HRP），另一个是装置信息服务（DIS），因此要找

到一个支持 HRP 设备，调用 LGATTClient.getServiceCount() ，然后 getServiceInfo()并依指数检查它们的 HRS 和 DIS UUID 是否有出现在列表中。如果两个都有，那表示装置支持 HRP，您已可以开始查询对应的特性。

以下范例显示如何获得一个装置所提供的所有服务。

```
client.connect(info.bd_addr);
int numberofServices = client.getServiceCount();
// all services uuid supported by this device
for(int i = 0; i < numberofServices; ++i){
    LGATTUUID serviceUUID;
    boolean isPrimary = false;
    client.getServiceInfo(i, serviceUUID, isPrimary);
    Serial.println(serviceUUID);
}
client.disconnect(info.bd_addr);
```

当您已确认连接上的装置支持您想要的配置文件之后，您就能用 LGATTClient.readCharacteristic()来读属性。

```
// read characteristic
LGATTUUID characteristicUUID = 0x2A37;
LGATTAttributeValue attrValue;
boolean isPrimary;
if(client.readCharacteristic(serviceUUID, isPrimary, characteristicUUID,
attrValue)){
    Serial.print("characteristic value:");
    Serial.print((char*)attrValue.value);
    Serial.println();
}
```

和用 LGATTClient.writeCharacteristic()来写属性。

```
// write characteristic
LGATTAttributeValue attrValue;
char szbuf[] = "value to write";
memset(&attrValue, 0, sizeof(attrValue));
memcpy(attrValue.value, szbuf, strlen(szbuf));
LGATTUUID writeUUID = 0x2A39;
attrValue.len = strlen(szbuf);
if (client.writeCharacteristic(serviceUUID, isPrimary, writeUUID,
attrValue)){
    Serial.println("Attribute written");
}
```

5.6. 使用 GPS

这个章节介绍如何调整您的 LinkIt ONE 开发板与所需的代码以使用内置 GPS 功能。此指导说明如何打开 GPS 并等待至得到固定位置。在位置固定后，检索经度/纬度信息与可见的卫星数目。

5.6.1. 硬件设置

将 GPS 天线接至天线接口以准备好 LinkIt ONE 硬体设置，如图 39 所示。

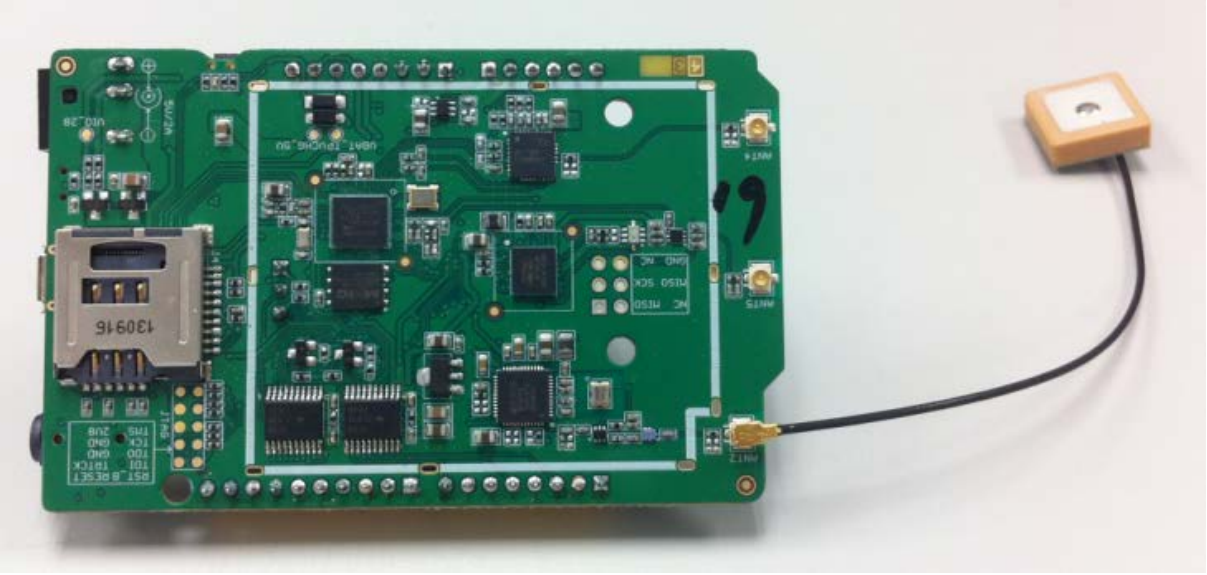

图 39 接上 GPS 天线的 LinkIt ONE 开发板

5.6.2. Software setup 软件设置

这个章节描述创建设置与使用 GPS 模块代码的必要步骤。

5.6.2.1. 包含 GPS 函数库

您应该在您的代码包含 GPS 函数库，请在您程序的 Arduino IDE 中：点击 **Sketch** 选单并指向 **Import Library** 并点击 **GPS**。您就会看到您的 Sketch 包含了 GPS 头文件。

```
#include <LGPS.h>
```

您现在可以在您的 Sketch 使用 LGPS 对象来执行 GPS 相关任务。请参阅 LinkIt ONE API 参考手册以详细了解 LGPS 功能。

5.6.2.2. 启动 GPS 模块

依照以下启动 GPS

```
LGPS.powerOn();
```

5.6.2.3. 获取 GPS 数据

GPS 数据能使用 getData()获取。参数为包含所有开发板位置的 GPS 信息的结构。在此结构的 GPGGA 部分具备了您 Sketch 所需要的讯息：位置是否锁定、多少可见的卫星与经度纬度。

GPGGA 文件格式的信息能从 NMEA 网站找到，但此指南用逗号来分隔出需要的细节。

使用 next token() 来解析 GPGGA 文件里的数据，解析结果将令牌（token）存入缓冲存储。解析后的数据第一令牌为$GPGGA，第二为 UTC 时间，第三与第六为纬度与经度；第七为 GPS 是否达到定位的显示；第八为可见的卫星数目。

```
gpsSentenceInfoStruct info;
LGPS.getData(&info);
printGPGGA((char*)info.GPGGA);

void printGPGGA(const char* str)
{
  char latitude[20];
  char longitude[20];
  char buf[20];
  const char* p = str;

  p = nextToken(p, 0);       // GGA
  p = nextToken(p, 0);       // Time
  p = nextToken(p, latitude);  // Latitude
  p = nextToken(p, 0);       // N
  p = nextToken(p, longitude); // Longitude
  p = nextToken(p, 0);       // E
  p = nextToken(p, buf);       // fix quality

  if(buf[0] == '1')
  {
    // GPS fix
    p = nextToken(p, buf);     // number of satellites
    Serial.print("GPS is fixed:");
    Serial.print(atoi(buf));
    Serial.println(" satellite(s) found!");
    Serial.print("Latitude:");
    Serial.println(latitude);
    Serial.print("Longitude:");
    Serial.println(longitude);
  }
  else
  {
    Serial.println("GPS is not fixed yet.");
  }
}

const char *nextToken(const char* src, char* buf)
{
  int i = 0;
  while(src[i] != 0 && src[i] != ',')
    i++;
```

```
  if(buf)
  {
    strncpy(buf, src, i);
    buf[i] = 0;
  }

  if(src[i])
    i++;
  return src+i;
}
```

5.6.2.4. 完整代码

以下为完整 Sketch 代码：

```
#include <LGPS.h>

gpsSentenceInfoStruct info;

const char *nextToken(const char* src, char* buf)
{
  int i = 0;
  while(src[i] != 0 && src[i] != ',')
    i++;

  if(buf)
  {
    strncpy(buf, src, i);
    buf[i] = 0;
  }

  if(src[i])
    i++;
  return src+i;
}

void printGPGGA(const char* str)
{
  char latitude[20];
  char longitude[20];
  char buf[20];
  const char* p = str;

  p = nextToken(p, 0);       // GGA
  p = nextToken(p, 0);       // Time
  p = nextToken(p, latitude);  // Latitude
  p = nextToken(p, 0);       // N
  p = nextToken(p, longitude); // Longitude
  p = nextToken(p, 0);       // E
  p = nextToken(p, buf);       // fix quality

  if(buf[0] == '1')
  {
```

```
    // GPS fix
    p = nextToken(p, buf);     // number of satellites
    Serial.print("GPS is fixed:");
    Serial.print(atoi(buf));
    Serial.println(" satellite(s) found!");
    Serial.print("Latitude:");
    Serial.println(latitude);
    Serial.print("Longitude:");
    Serial.println(longitude);
  }
  else
  {
    Serial.println("GPS is not fixed yet.");
  }
}

void setup() {
  Serial.begin(9600);
  LGPS.powerOn();
}

void loop() {
  LGPS.getData(&info);
  printGPGGA((char*)info.GPGGA);
  delay(1000);
}
```

5.7. 使用 GPRS 连网

这个章节介绍如何设置您的 LinkIt ONE 开发板与所需的代码以通过 GPRS 连接来获取网页内容。

5.7.1. 硬件设置

依照以下步骤准备好 LinkIt ONE 开发板，如图 40：

1) 插入标准大小的 SIM 卡到开发板背面的 SIM 插槽。如果您使用 mirco 或 nano 的 SIM 卡，则需要一个适配器。也请确认 SIM 卡没有 PIN 锁定，由于 LinkIt ONE SDK 不支持 PIN 锁定的 SIM 卡。
2) 将 GSM/GPRS 天线接至天线接口。

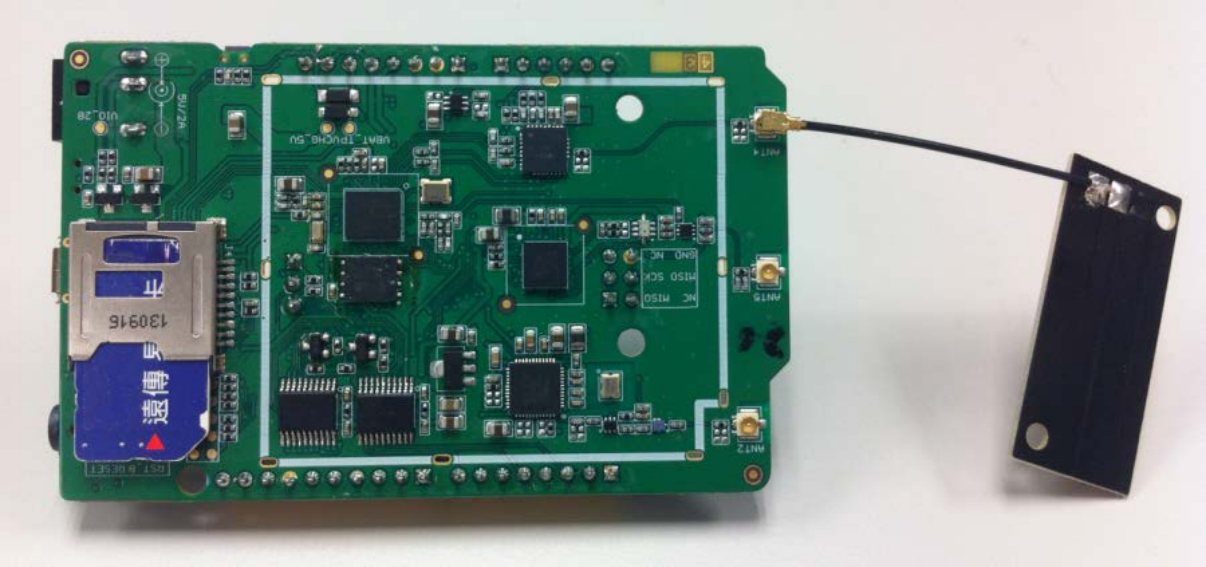

图 40 插入 SIM 卡并接上 GSM 天线的 LinkIt ONE 开发板

5.7.2. 软件设置

这个章节介绍的必要步骤能创建检索网页内容的代码通过 GPRS 连接。

5.7.2.1. 包含 GPRS 函数库

您应该在您的代码包含 GPRS 函数库，请在您程序的 Arduino IDE 中：点击 **Sketch** 选单并指向 **Import Library** 并点击 **LGPRS。**您就会看到您的 **Sketch** 包含了 GPRS 头文件。在本手册 LinkIt ONE 示范为 GPRS 客户，所以仅需保留前两个头文件并删去其余。

```
#include <LGPRS.h>
#include <LGPRSClient.h>
```

您现在能在 Sketch 使用 `LGPRS` 与 `LGPRSClient` 对象来执行 GPRS 相关任务，请参阅 LinkIt ONE API 参考手册以取得 `LGPRS` 与 `LGPRSClient` 的详细功能。

5.7.2.2. 等待 GPRS 初始化

等待 GPRS 模块初始化。attachGPRS()在 GPRS 就绪后会回传非 0 值。

```
while(!LGPRS.attachGPRS())
{
  delay(1000);
}
```

5.7.2.3. 连接到网站

第二步是连接到网站，由 LGPRSClient 对象达成。connect()的第一个参数为 URL，第二个参数为串口；HTTP 使用接口 80。

```
#define SITE_URL "www.mediatek.com"

LGPRSClient client;
client.connect(SITE_URL, 80);
```

5.7.2.4. 传送 HTTP 请求

建立连接后即可像串流一样，供您读取与写入。为检索网页内容，您需要传送 HTTP GET 请求，如下面的片段代码所示：

```
client.println("GET / HTTP/1.1");
client.println("Host: " SITE_URL ":80");
client.println();
```

5.7.2.5. 取得网页内容

如果一切正常，远端网路服务器会回应您的 GET 请求并开始回传内容。您接下来可以从连接对象读取数据：

```
int v;
while(c.available())
{
  v = c.read();      // return one byte at a time
  if(v < 0)
    break;           // no more data
}
```

5.7.2.6. 完整代码

以下是完整代码：

此代码获取的数据打印在波特率 9600 的串行端口。

```
#include <LGPRS.h>
#include <LGPRSClient.h>

#define SITE_URL "www.mediatek.com"

LGPRSClient client;

void setup()
{
  Serial.begin(9600);

  while(!LGPRS.attachGPRS())
  {
    Serial.println("wait for SIM card ready");
    delay(1000);
  }

  Serial.print("Connecting to : " SITE_URL "...");
  if(!client.connect(SITE_URL, 80))
  {
    Serial.println("FAIL!");
    return;
  }
  Serial.println("done");

  Serial.print("Sending GET request...");
  client.println("GET / HTTP/1.1");
  client.println("Host: " SITE_URL ":80");
  client.println();
  Serial.println("done");
}

void loop()
{
  int v;
  while(client.available())
  {
    v = client.read();
    if (v < 0)
      break;

    Serial.write(v);
  }
  delay(500);
}
```