[知识库]Dthing应用程序开发入门指南

*www.yarlungsoft.com*

V1.0

**版权声明：**

本文档任何内容享有法律保护，版权所有**©2018 YarlungSoft有限公司**

**保密声明：**

文档信息为保密信息，任何使用者未经许可授权严禁将本文档进行二次传播、严禁泄露给任何其他第三方、严禁使用与规定以外的其它目的。

|  |  |  |  |
| --- | --- | --- | --- |
| **历史记录** | | | |
| **修改日期** | **版本号** | **修改内容** | **修改人员** |
| 2018-06-27 | V1.0 | 初始版本 | 谭 星 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**TALBE OF CONTENT**

[正文 6](#_Toc518056982)

[1.1 概述 6](#_Toc518056983)

[1.2 应用开发 7](#_Toc518056984)

[1.2.1 环境搭建 7](#_Toc518056985)

[1.2.2 开发初体验 7](#_Toc518056986)

[1.2.3 实战操作 16](#_Toc518056987)

[1.3 硬件搭载和控制 18](#_Toc518056988)

[1.3.1 前期准备 18](#_Toc518056989)

[1.3.2 硬件搭载 19](#_Toc518056990)

[1.3.3 控制指令 24](#_Toc518056991)

[1.4 数据入云 27](#_Toc518056992)

[1.4.1 入云实战 28](#_Toc518056993)

[1.4.2 消息监控 29](#_Toc518056994)

前言

本文档属于雅鲁知识库的存档文本，用于公司内部技术积累和沉淀，不断促进研发人员知识复用和能力提高，从根本上提高研发生产效率，实现公司和个人共同发展的双赢目标。所有研发人员都将是知识库的贡献者和受益者。

# 正文

## 概述

Dthing，全称DthingOS， 是国内首个基于虚拟机的物联网开源操作系统，也是国内首家提供物联网应用商店的开源项目。目前已经拥有包括DthingOS， Dthing应用API， Dthing集成开发插件，Dthing模拟器，Dthing开发板，Dthing硬件产品和Dthing应用商店在内的完整的生态系统。Dthing提供了对java风格应用的兼容支持，并设计封装了一组针对各种硬件接口和传感器的IOT应用API，屏蔽了不同硬件的实现区别，实现同一个应用跨硬件和跨平台的运行。

DthingOS在应用领域还有更多的优势，例如支持远程应用管理（下载、安装、运行、卸载等），方便使用和调试，不同设备API的控制组合，任由开发者来创造。我们都可以自己从淘宝上购买一款传感器，然后通过应用来驱动它。作为一个开源的物联网操作系统，你甚至可以设计、扩展IOT应用API，然后在中间层实现这些API，开发出一款完全属于自己的物联网应用设备！

本文分为三个部分，应用开发、硬件搭载和数据入云。第一部分以具体示例详细讲解Dthing应用开发的全过程，包括开发环境配置、示例应用开发、打包编译等，作为大家认识Dthing应用开发的初体验。第二部分主要介绍如何将一个开发好的应用搭载到DTU（Dthing生态链中的一个硬件产品）上去以及怎样让程序运行起来。第三部分则分析了应用采集数据后接入云端的相关细节。

## 应用开发

### 环境搭建

Dthing应用是基于java语言开发的，因此需要先在电脑上安装java的JDK（Java Development Kit），并且做好相关环境变量的配置，具体操作步骤可以参考[此处](https://blog.csdn.net/u012934325/article/details/73441617)。（推荐下载jdk7之前的版本）

Dthing应用开发用到的IDE（Integrated Development Environment ）自然也是开发java的经典集成开发环境——Eclipse，对java稍微有过接触的朋友应该是相当的熟悉了，只不过我们又在Eclipse里面集成了基于DthingOS系统开发的API，[点击下载](https://dthingroup.github.io/download.html)，解压即可用（图2.2.1）。



图2.2.1 Dthing IDE解压后的文件包

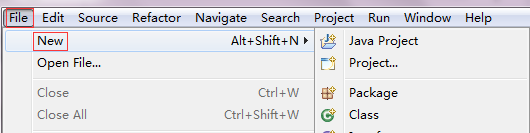
进入此解压文件包，找到启动程序eclipse.exe，双击运行。

### 开发初体验

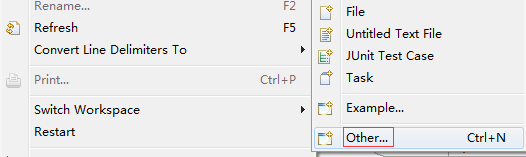
接下来我们将以创建示例工程的步骤操作截图为引，给大家清晰的展示一个非常简单的Dthing Demo应用开发全过程。“Hello，DthingApp，I’m coming…”

1. 新建工程。

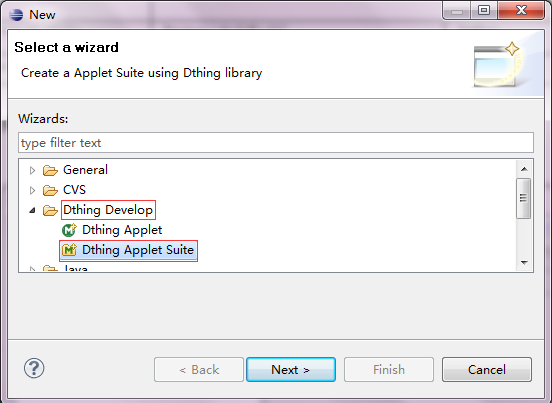
单击’File’（文件），在出现的下拉菜单中选择’New’（新建）。



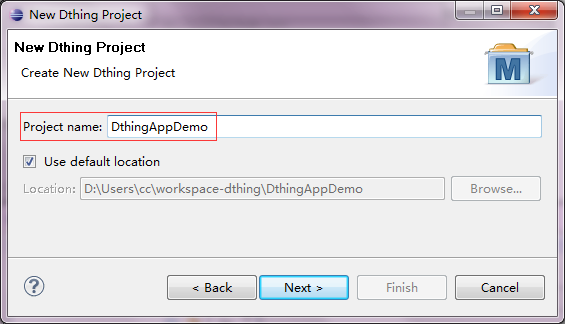
右侧出现新的菜单栏，点击底部的’Other’（其他）项。



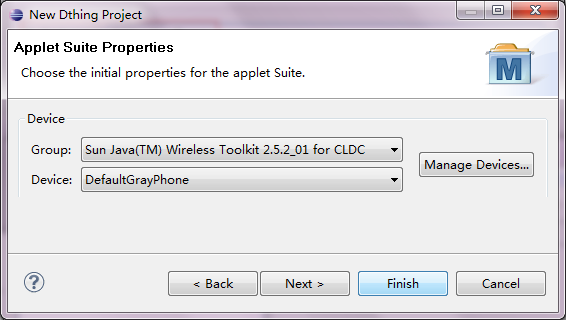
弹出一个新窗口，展开’Dthing Develop’文件，选择’Dthing Applet Suite’，这里就表示要创建的是一个Dthing工程，点击’Next’（下一步）。



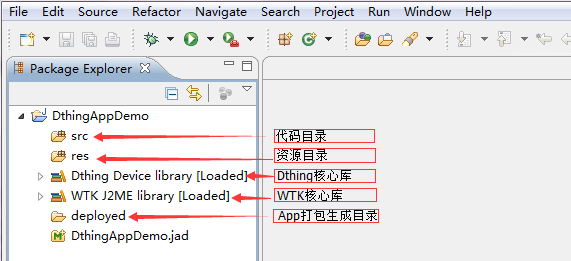
在Project name（项目名称）输入自己新建工程的名称，这里我创建一个名为DthingAppDemo的Dthing工程，工程路径采用默认路径，点击’Next’（下一步）。



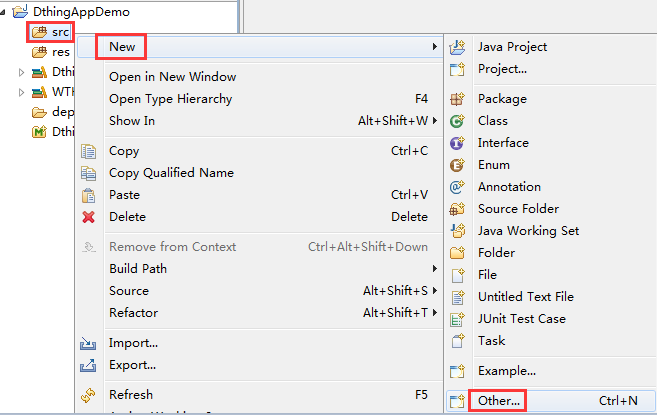
直接在新窗口点击’Finish’ 。（如果Device里面是空白选项，则需要配置WTK路径）



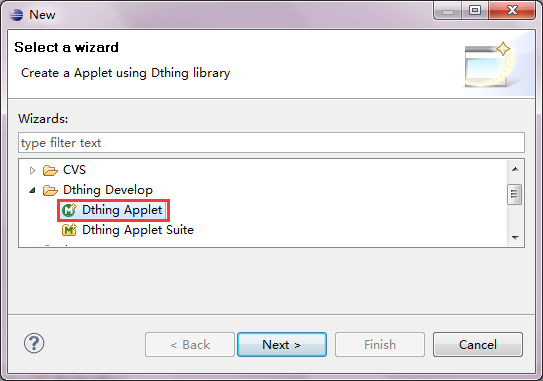
工程创建成功，结构如下，我们就开始在src路径下直接编写代码吧。



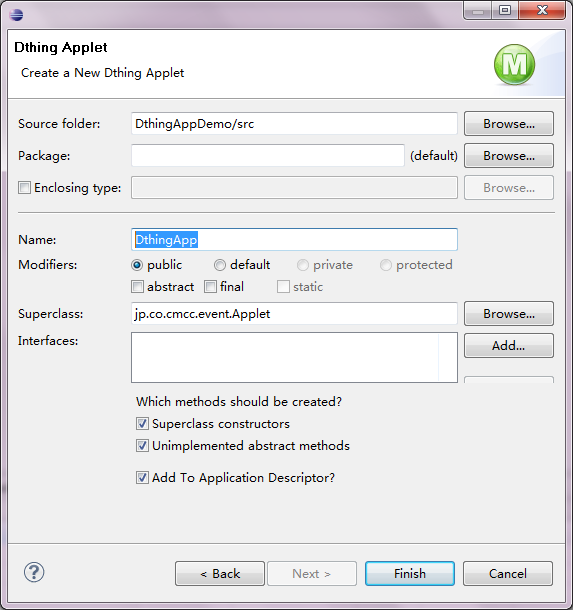
1. 创建主类。右键src 🡪 New 🡪 Other。



在弹出窗口选择’Dthing Applet’，点击’Next’（下一步）。



填入App名称，如本例写为’DthingApp’，其他选项默认即可，点击’Finish’，完成创建。



1. 代码编写

我们创建的DthingApp类会继承Applet类，并且为我们自动生成一个无参构造器和三个需要重写的方法。

* public DthingApp() {}

无参构造器，可以进行一些初始化工作。

* public void cleanup() {}

程序结束运行时调用，该方法内需要调用系统的notifyDestroyed()方法。

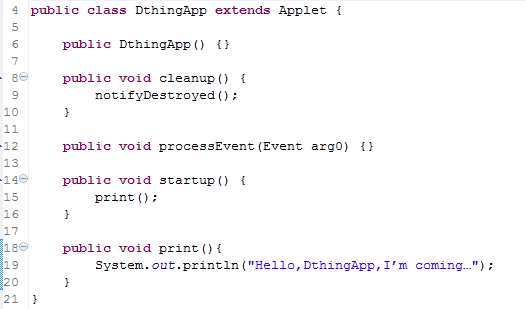
* public void processEvent(Event arg0) {}

处理各类事件。

* public void startup() {}

程序执行入口，处理我们应用的各种逻辑。

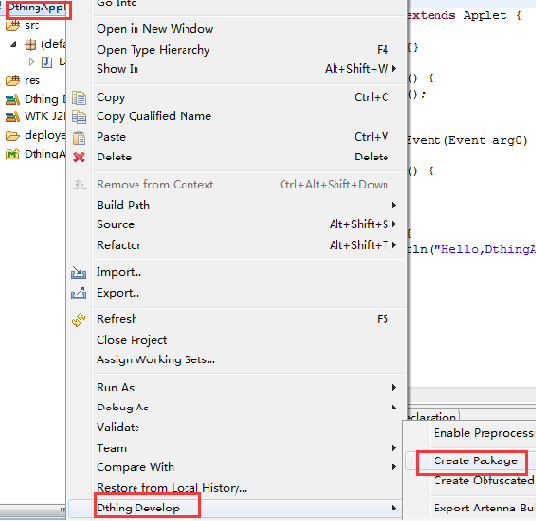
示例代码：



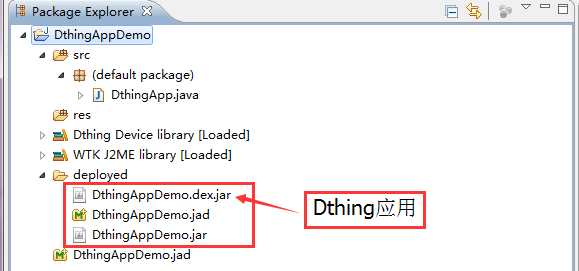
自定义一个print()方法，方法的功能是打印一行字符串，在startup()方法中调用。这样，一个可以打印一条字符串的Dthing应用就算是开发完成了，非常简单和高效。

1. 工程打包

在工程包名处右键，选择下拉菜单下部的’Dthing Develop’，弹出子菜单，点击’Create Package’，系统自动编译，完成对该Dthing应用的打包。



展开deployed文件，便可以看到已经打包好的Dthing应用文件。将已经生成好的x.dex.jar（另一个x.jar文件可以不用管）应用直接下载到搭载了DhingOS的硬件产品上，启动程序，便可以在硬件上执行我们写的逻辑。

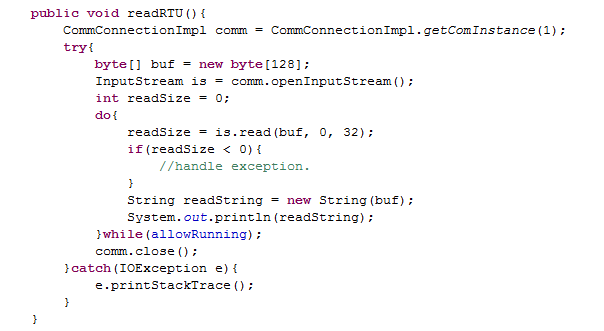


当然，上面创建的Dthing应用只是一个极为简单的演示案例，在实际的物联网中根本不具任何的应用价值。不过，这个案例已经很好的展示了Dthing应用开发的整个流程，一个处理简单逻辑的Dthing应用可以说是分分钟就开发完成了！但是，如果是针对复杂的实际应用开发呢，Dthing是否依然还能够如此得心应手的应对？这里，我就再举两个已经在实际物联网开发项目中用到过的例子，一个是对rtu设备进行数据采集，另一个则是控制LED灯的开关，这两种应用场景自然是要比只单纯的在代码里打印一行字符串要复杂很多，Dthing应用将如何实现这两项功能呢，让我们一起拭目以待。

### 实战操作

1. 采集RTU设备数据

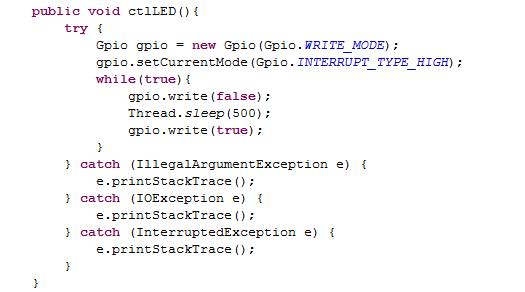
继续用到上面已经开发好的DthingApp工程，工程代码里我们已经添加了一个自定义打印字符串的print()方法，我们需要再定义一个名叫readRTU()的方法，用来读取被采集设备的数据，代码参考下图。



首先获取一个定义名为 comm的串口实例，通过这个串口实例打开一个输入流，获取流对象is，再循环调用is的read()方法从硬件串口不停读取数据，然后将读到的数据readString做一个打印输出。就凭这十来行代码，我们已经基本成功的完成了第一项功能——采集RTU设备数据，要知道，目前市场上大量的硬件设备都是通过RTU/485来实现通信的！我们只要将print()方法用readRTU()方法替换掉，打包成的这个Dthing应用，就可以完成诸如对温湿度、RFID、电表、烟感、压力等等多种硬件设备的数据采集。

1. 控制LED灯

我们再看看实现控制LED灯开关的代码，同样是再创建了一个新方法ctlLED()。



有了上一个应用的开发经验，我们就可以类似的分析本次代码。首先也是拿到一个控制led灯实例gpio，模式设置成高频，推测初始状态是开灯，然后在一个循环里面反复对这个实例进行真假写入，可想而知，便是对led灯开、关灯两种状态的切换控制。抛开异常处理部分，控制led灯的逻辑实现代码仅有几行，也是相当的简单！

通过对上面几个Dthing应用开发过程的分析，不难发现，不管是开发一个演示性的工程还是为真实应用场景做应用开发，实际的开发工作量都是非常小的，这就得归功于我们DthingOs底层的实现了，它已经为我们处理了绝大分复杂的工作，我们唯一要做的就是熟悉它提供的大量API（快来熟悉我）接口！相信Dthing会是你在物联网领域驰骋的一把利器。

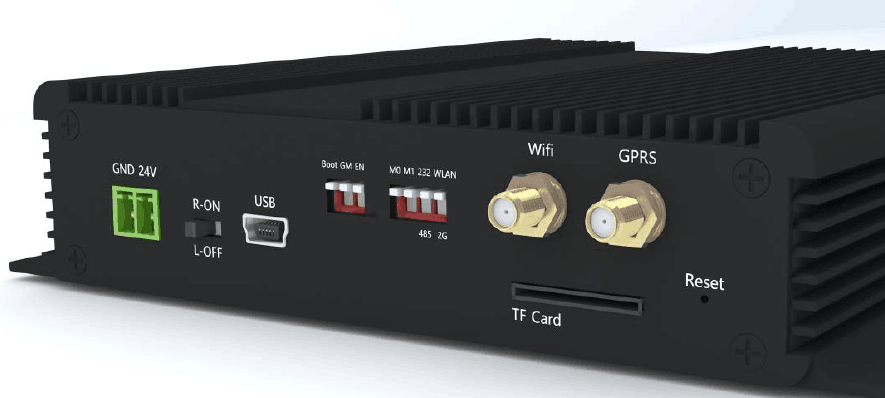
## 硬件搭载和控制

开发一个Dthing应用，现在对我们来说已经不算是什么难事，可是，你知道怎么把一个开发好的应用搭载到硬件上去吗？本章就将介绍与此相关的知识点，同时也对应用在硬件上运行、停止、删除等相关操作做了详细说明。

### 前期准备

1. 硬件部分

* 搭载了DthingOS的硬件设备，这里选择DTU。



* 相关配件，如电源、USB转串口线等。

1. 软件部分

* Dthing应用。
* 串口软件工具，如本例使用——超级终端（下载）。

完成这些准备后，我们就可以开始进行Dthing应用的搭载了。

### 硬件搭载

硬件搭载是指将Dthing应用下载到搭载了DthingOS的硬件产品上，支持本地下载和远程下载。下面是具体的搭载步骤：

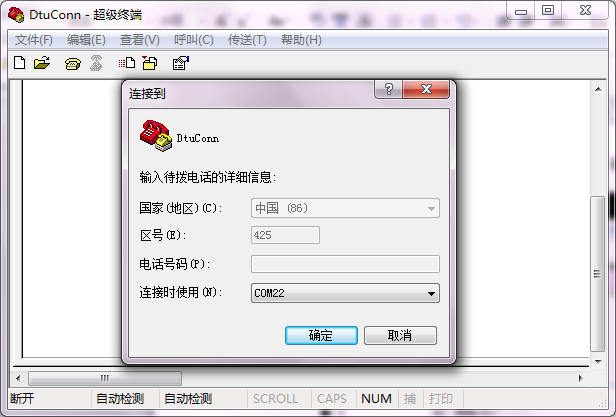
* 用串口线连接DTU和电脑。
* 打开超级终端，查看DTU系统运行状态。

1. 启动超级终端，输入名称，如DtuConn，确定。



1. 选择正确的串口号，如果找不到可以尝试重启软件或

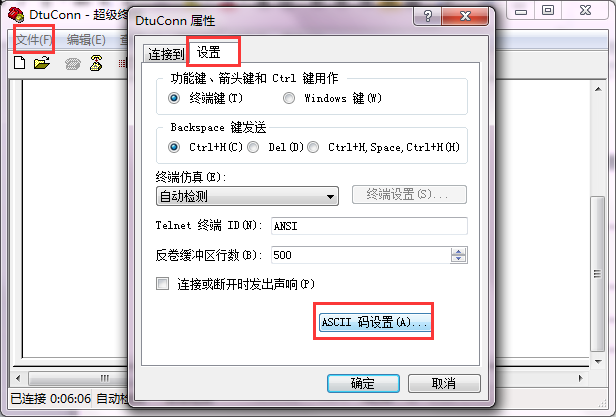
者拔插一下连接DTU的USB转串口线，本机连接DTU的串口号是22，确定。



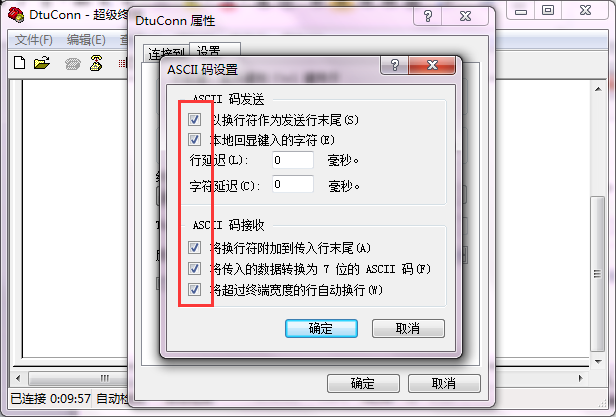
1. 配置串口属性，波特率为115200，流控制选择无，应用并确定。



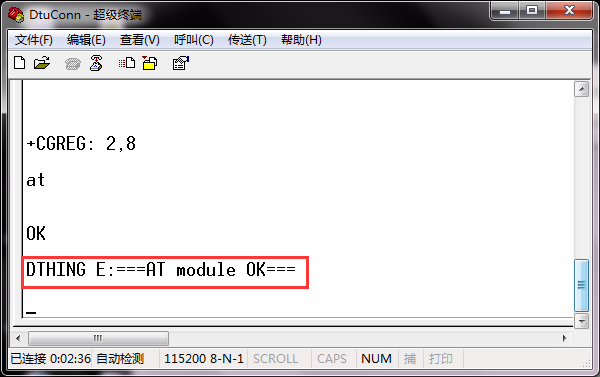
1. 配置软件编码属性。文件下面选择属性，弹出窗口选择设置，点击ASCII设置。



全部勾选，确定。



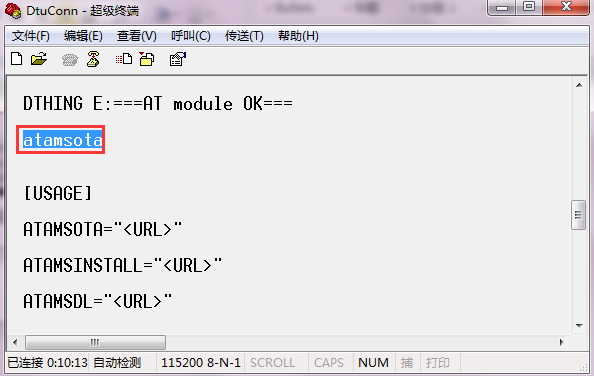
1. 打开DTU电源，通过串口工具界面显示的日志，查看DthingOS系统启动。



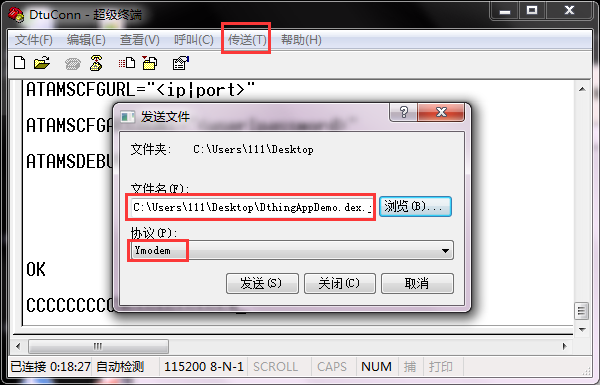
当日志刷出“===AT module OK===”字样，就可以进行应用的下载操作了。

* 本地下载

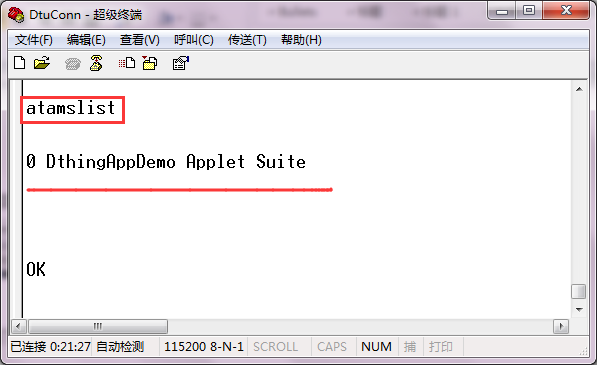
1）界面直接输入atamsota下载指令，回车。



1. 打开‘传送’，选择‘发送文件’，选择应用地址，协议Ymodem，发送。



3）发送成功后，在界面输入atamslist列表指令，查看下载的应用列表。



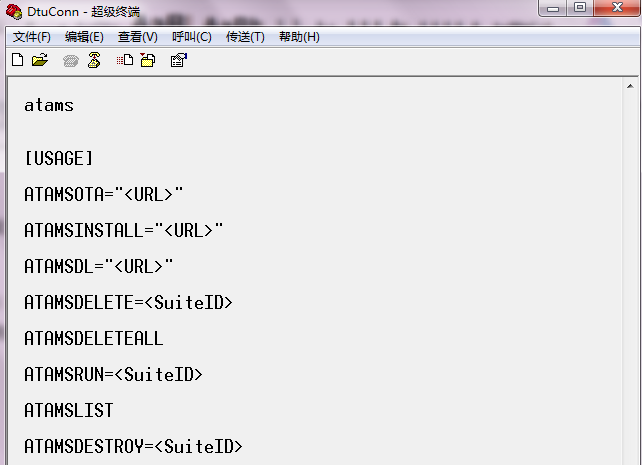
到这里，我们已经把开发的第一个应用搭载到DTU硬件上了，0是系统为应用生成的ID号。

* 远程下载

界面输入atamsota=”应用远程地址”，回车即可。

### 控制指令

控制指令是专门为Dthing应用与DthingOS交互而设计的一系列指令，可以通过在界面输入atams指令查看。



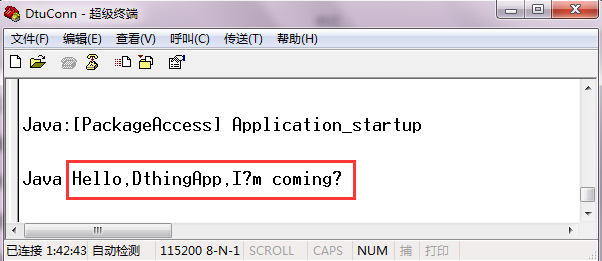
本小节将对常用的指令进行介绍。

* atamslist：应用列表指令，查看当前硬件搭载Dthing应用列表。
* atamsrun：应用运行指令，启动某一个应用，格式atamsrun=ID。
* atamsstatus：应用运行状态指令，查看某个应用的状态，格式atamsstaus=ID。
* atamsdestory：应用结束运行指令，结束一个正在运行的应用，格式atamsdestory=ID。
* atamsdelete：应用删除指令，删除某个应用，格式atamsdelete=ID。
* atamsdeleteall：应用清空指令，删除硬件上所有应用。
* atamscfginit：应用自启动指令，应用伴随系统启动而自行启动，格式atamscfginit=”ID”。
* atamscfgcancel：应用自启动失效指令，取消某个应用的自动启，格式atamscfgcancel=ID。
* atamscfgcancelall：应用自启动清除指令，清除所以应用的自动启。
* atamsota：应用下载指令，本地或者远程下载应用，远程格式atamsota=”url”。
* atamsreset：系统重启指令，重启当前系统。
* 其他指令，如atamsdebug、atamscfgurl等。

注意：无格式说明的指令，指令执行格式就是本身。

示例：

我们在界面输入’atamsrun=0’，回车执行，从日志看到我们写的print()方法已经被系统执行了。



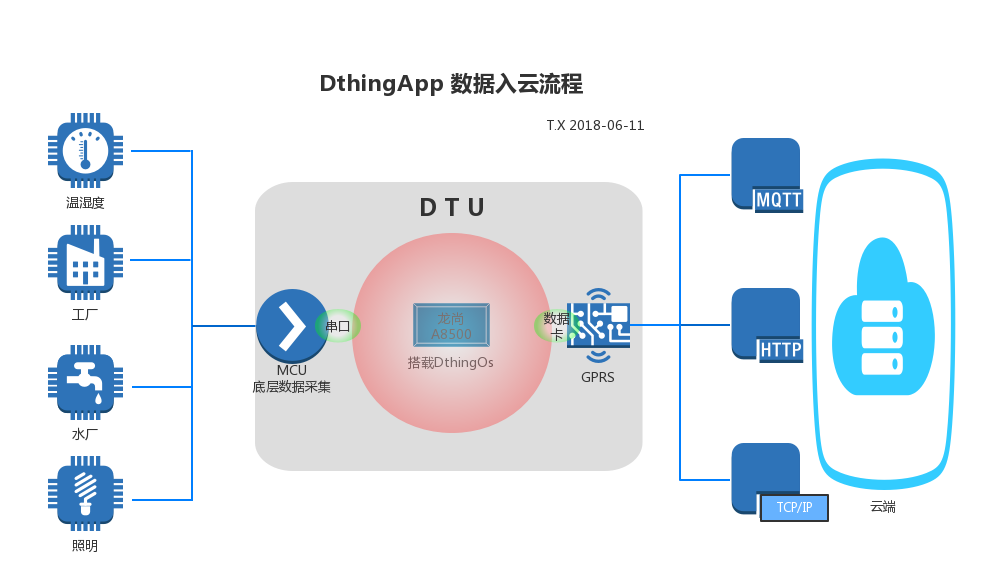
目前，除了通过本地串口操作这些指令达到与应用的交互能力外，我们也已经实现了服务器远程操作能力和Web网页端操作应用能力，指令使用方式一致。

通过对本章的学习，我相信小伙伴们已经可以将自己开发的Dthing应用搭载到硬件上并且能很顺利的跑起来了！在物联网的道路上，Dthing也将不停的奔跑，不断优化自己，提供更多的硬件接口，给各位开发使用的小伙伴带来更丰富、高效、简洁的开发体验！

## 数据入云

Dthing应用的强大，不仅仅体现在对众多设备数据采集的支持，对云端数据接入这块，也是有相当的优势。DthingOS不仅支持各种传统的网络传输协议，还移植了当下许多优秀的物联网接入协议，比如MQTT协议，COAP协议等，势必在物联网这个新兴行业留下浓墨重彩的一笔！

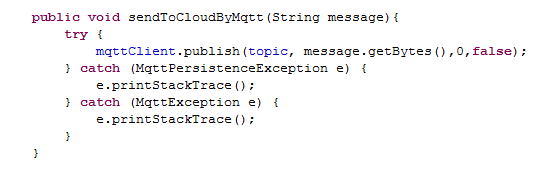
本章将以也将以实际项目中用到的技术为例，介绍Dthing应用是如何将采集数据到的数据接入云端的，下图展示了数据从设备采集到流向云端的全过程。



### 入云实战

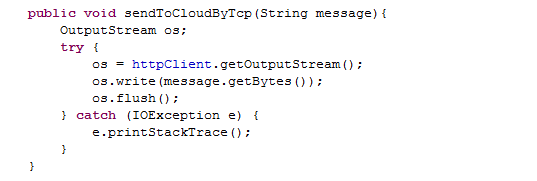
回顾[1.2.3](#_实战操作)章节中第一点“采集RTU设备数据”中给出的示例，当Dthing应用在readRTU()方法中读取到数据后，我们不再做打印处理，而是调用这里将要新建的一个方法，方法的作用便是将这些数据发送到云端。这里，将给出MQTT和TCP两种入云方式的核心代码示例，每种对象的创建过程都已省去，读者可以自行查阅相关协议文档了解。

1. MQTT示例：



mqttClient是API提供的MQTT客户端的一个实例，调用此处的sendTocloudByMqtt(readString)方法，消息便通过topic这个主题发布到云端。

2）TCP示例：



httpClient是API提供的Socket客户端的一个实例，调用此处的sendTocloudByTcp(readString)方法，消息就通过tcp长连接的方式传输到云端。

### 消息监控

