**Pebbel固件开发说明**

目录

[1. Windows下建立开发环境 1](#_Toc51083066)

[1.1 下载安装EmbeddedStudio IDE工具 1](#_Toc51083067)

[1.2 下载安装 nRF Connect for Desktop 1](#_Toc51083068)

[1.3 通过Assistant建立开发环境 1](#_Toc51083069)

[1.4 ToolChain Manager 一键建立开发环境 1](#_Toc51083070)

[1.5 EmbeddedStudio IDE 配置 1](#_Toc51083071)

[1.5.1 编译工具链配置 1](#_Toc51083072)

[1.5.2 pebble app project配置 1](#_Toc51083073)

[1.5.3 pebble board 配置 1](#_Toc51083074)

[1.5.4 menuconfig 系统配置 1](#_Toc51083075)

[2. Linux下建立开发环境 1](#_Toc51083076)

[2.1 安装依赖 1](#_Toc51083077)

[2.2 安装编译工具 1](#_Toc51083078)

[2.3 命令行配置 1](#_Toc51083079)

[2.4 命令行编译 1](#_Toc51083080)

[2.5 烧录 1](#_Toc51083081)

[3. Pebble App源码说明 1](#_Toc51083082)

[3.1 Pebble源码下载 1](#_Toc51083083)

[3.2 源码目录 1](#_Toc51083084)

[3.3 pebble app 源码详细说明 1](#_Toc51083085)

[3.3.1 程序流程图 1](#_Toc51083086)

[3.3.2 部分API说明 1](#_Toc51083087)

[3.3.3 mqtt upload 数据包格式 1](#_Toc51083088)

[3.4 配置说明 1](#_Toc51083089)

[3.4.1 pebble工作模式配置 1](#_Toc51083090)

[3.4.2 GPS配置 1](#_Toc51083091)

[3.4.3 modem模式选择 1](#_Toc51083092)

[3.4.4 MQTT 证书内置 1](#_Toc51083093)

[3.4.5 ECDSA 数字签名证书内置 1](#_Toc51083094)

[4. 通过串口升级程序 1](#_Toc51083095)

[4.1 mcuboot recover 模式升级 1](#_Toc51083096)

# Windows下建立开发环境

## 下载安装EmbeddedStudio IDE工具

通过下面这个连接

“https://www.segger.com/downloads/embedded-studio/EmbeddedStudio\_ARM\_Nordic\_v452\_win\_x64.zip“

下载EmbeddedStudio IDE V4.5.2版本，下载后解压后

“EmbeddedStudio\_ARM\_Nordic\_v452\_win\_x64\arm\_segger\_embedded\_studio\_v452\_win\_x64\_nordic\bin\emStudio.exe“

即为pebble开发使用的IDE。其他版本下载连接如下：

“https://www.segger.com/downloads/embedded-studio/EmbeddedStudio\_ARM\_Nordic\_v452\_win\_x64.zip“

“https://www.segger.com/downloads/embedded-studio/EmbeddedStudio\_ARM\_Nordic\_v452\_win\_x86.zip“

“https://www.segger.com/downloads/embedded-studio/EmbeddedStudio\_ARM\_Nordic\_v452\_linux\_x64.tar.gz“

“https://www.segger.com/downloads/embedded-studio/EmbeddedStudio\_ARM\_Nordic\_v452\_linux\_x86.tar.gz“

“https://www.segger.com/downloads/embedded-studio/EmbeddedStudio\_ARM\_Nordic\_v452\_macos\_x64.dmg“

## 下载安装 nRF Connect for Desktop

在这个连接下载 nRF Connect

“https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Connect-for-desktop/Download#infotabs”

这里以 V3.3.1 为例:

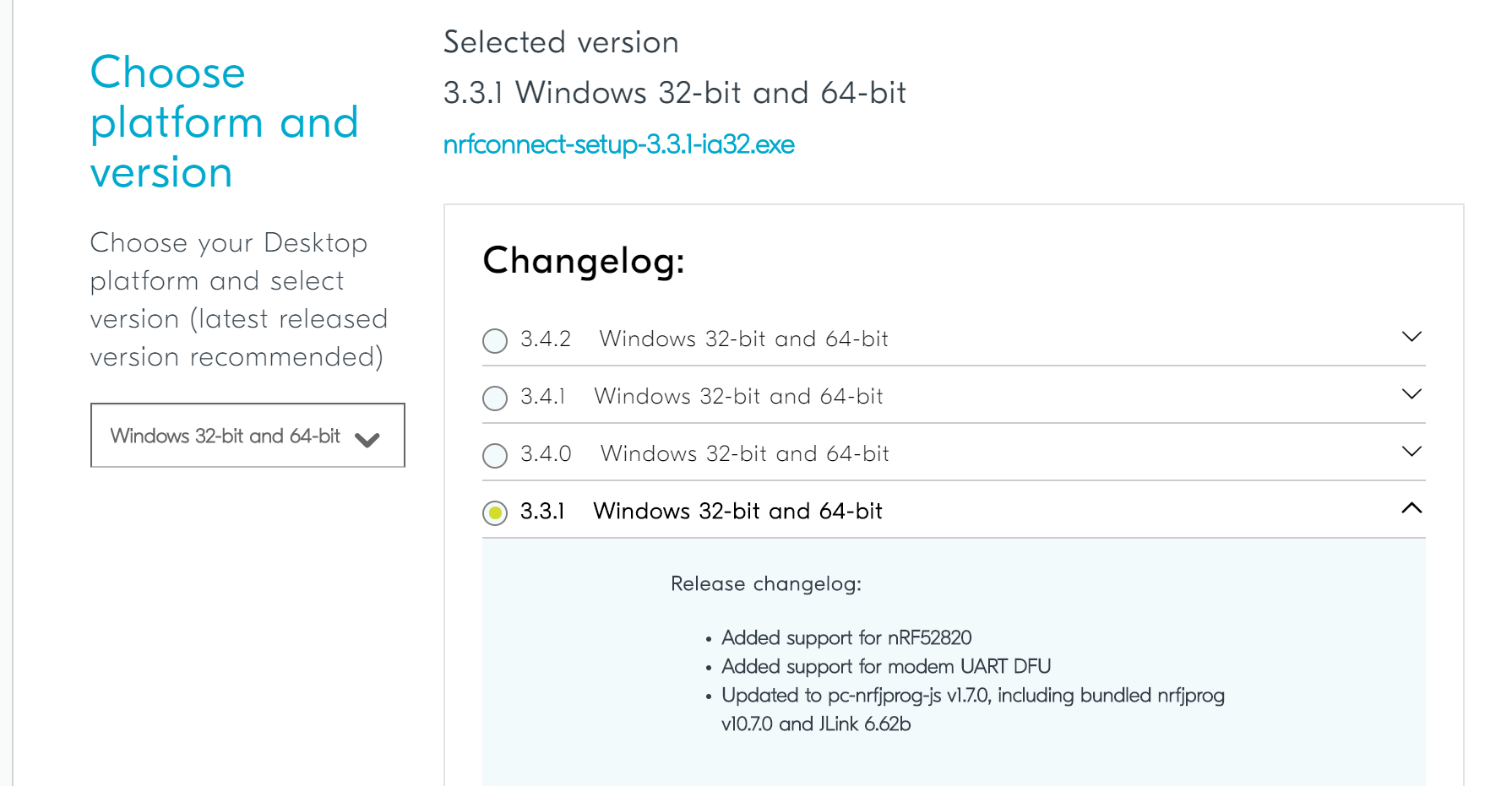


图1 下载nRF Connect for Desktop

下载后直接运行 nrfconnectsetup331ia32.exe 完成安装。

## 通过Assistant建立开发环境

打开 nRF Connect.exe 选择打开Getting Started Assistant，如下图：

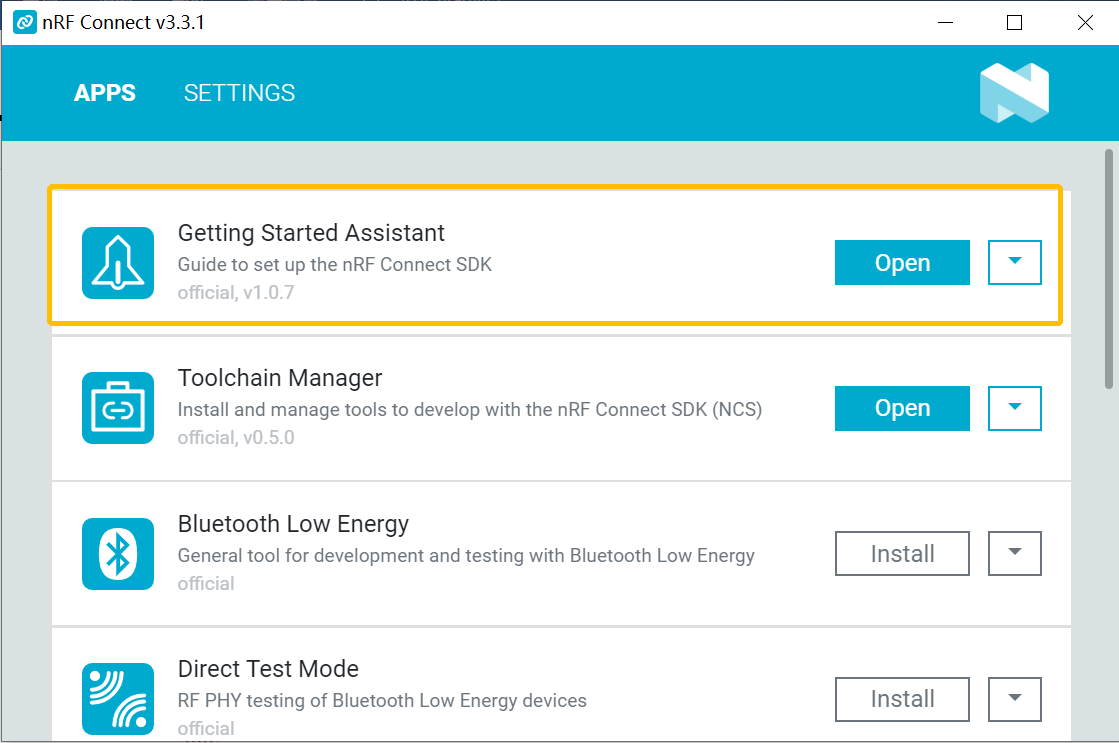


图2 nRF Connect.exe 打开助手

按照Assistant中的步骤逐个安装windows下的工具， 如下图:

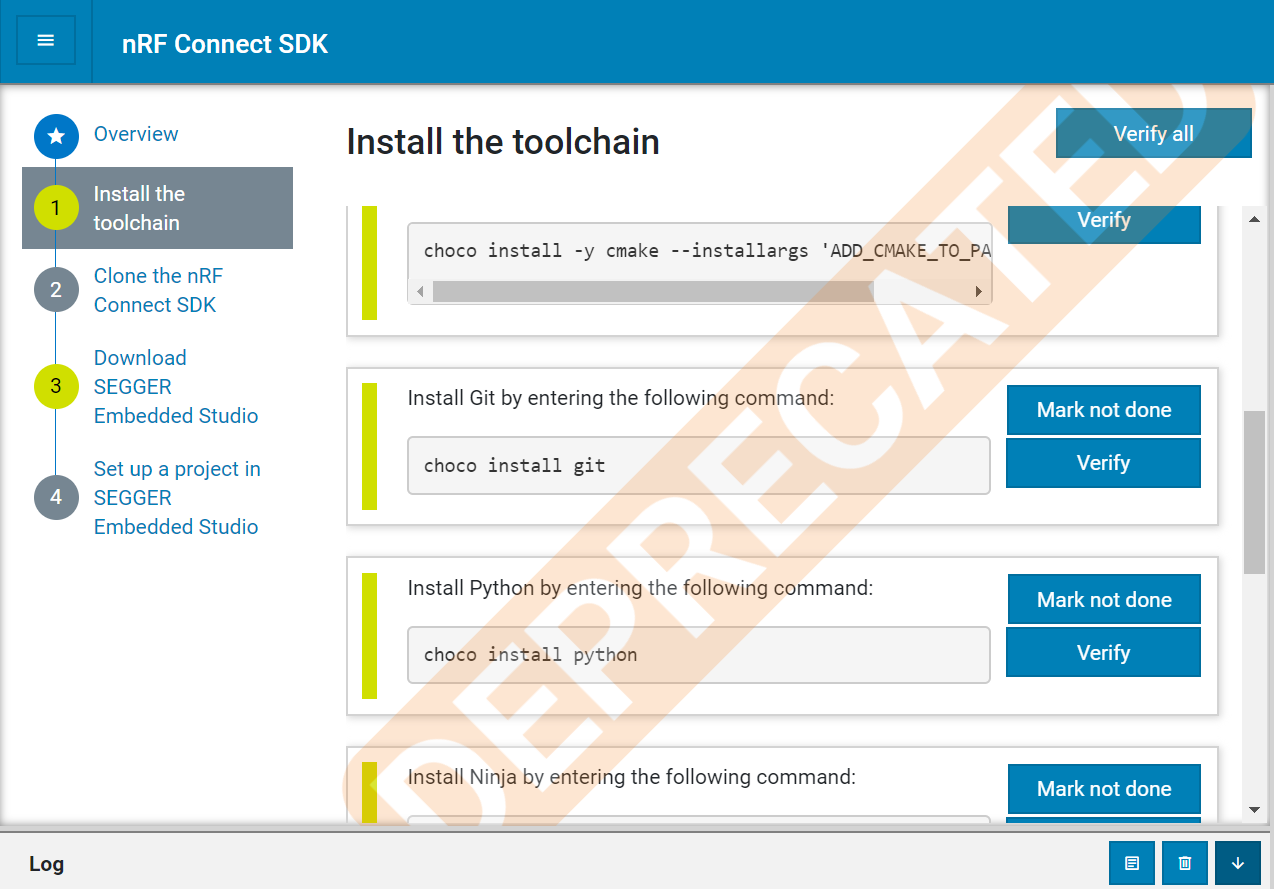


图3 通过Assistant安装开发环境

## ToolChain Manager 一键建立开发环境

打开 nRF Connect.exe 选择打开Getting Started Assistant, 如上图2所示。选择 ToolChain Manager install 后Open，在弹出

的窗口中可选择不同的 SDK 版本，如下图:

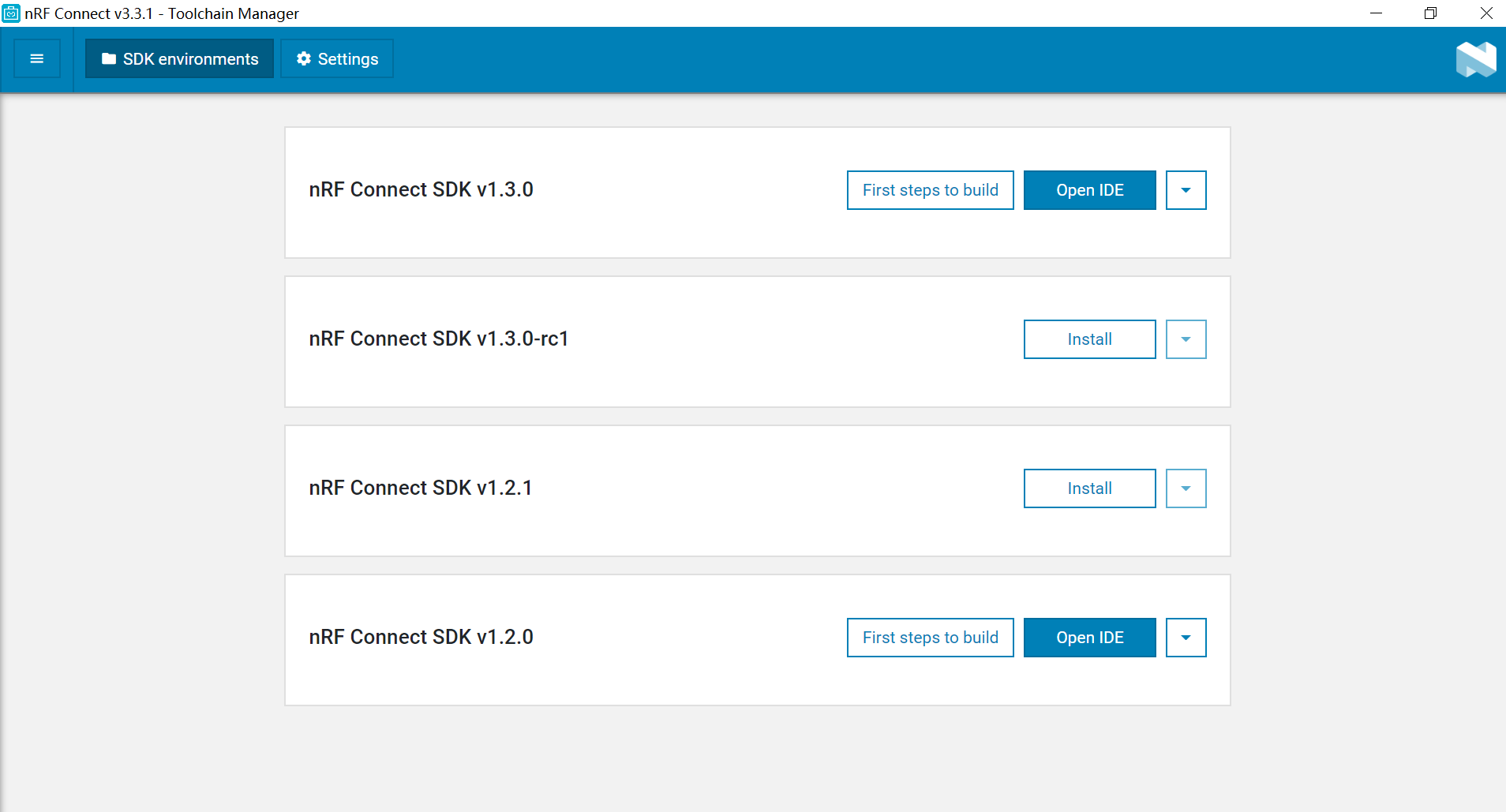


图4 通过ToolChain Manager安装开发环境

## 1.5 EmbeddedStudio IDE 配置

### 1.5.1 编译工具链配置

下载pebble 固件源码， 在nrf\applications\ 目录下asset\_tracker 即为 pebble app 项目源码。打开上述安装好的

emStudio.exe IDE工具，在 Tools菜单中选择Options 选项，在 Options窗口中选择 “nRF Connect” 配置项，点击 Directories 标签，分别配置 “GNU ARM Embedded Toolchain Directory” 和 “Zephyr Base” 变量，如下图：

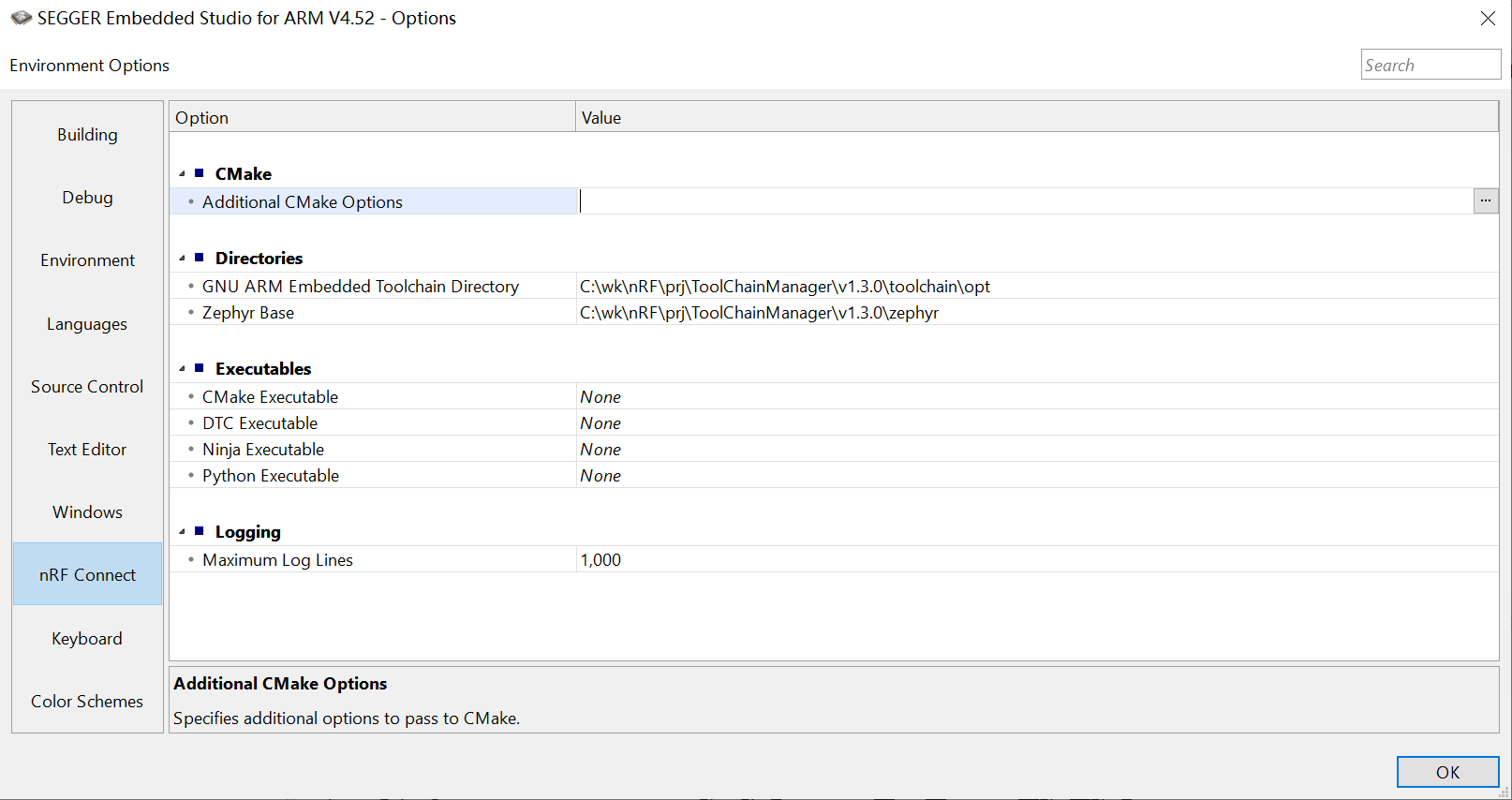


图 4 编译工具链配置

### 1.5.2 pebble app project配置

打开emStudio.exe IDE工具， 在File 菜单中选择 Open nRF Connect SDK Project… 选项。在弹出的窗口中分别配置 CMakeLists.txt、Board Directory 路径， Board Name 选择 “thingy91\_nrf9160ns” , 如下图所示：

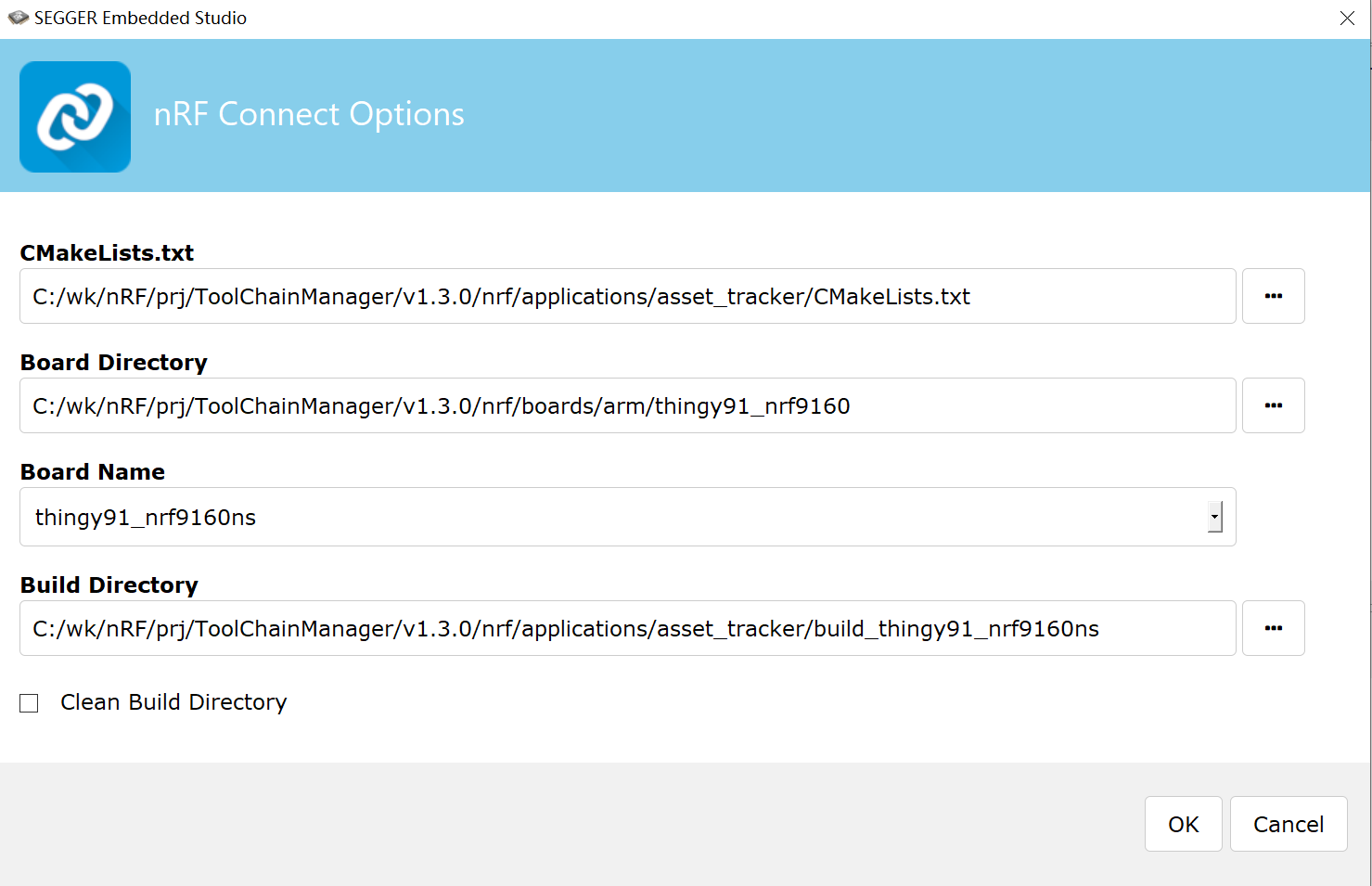


图 5 pebble app配置

### 1.5.3 pebble board 配置

Pebble 硬件配置目录为 pebble\nrf\boards\arm\thingy91\_nrf9160， 即上图中 Board Directory 的配置。硬件配置中主要包括

nRF9160 系统分区配置、外围设备驱动配置、以及在secure或 no secure模式下硬件的初始化。

### 1.5.4 menuconfig 系统配置

打开emStudio.exe IDE工具，选择 Project 菜单，选择 Configure nRF Connect SDK Project… 子菜单。在弹出的串口中选择

menuconfig后可以看到系统的各配置项，修改配置后点击右下角的 Configure 按钮保存。选择 menuconfig 后可以通过搜索功能查找需要配置的项，在右上角的搜索框中输入需要配置的关键字。例如修改 mqtt 配置，在搜索框中输入 mqtt , 如下图：

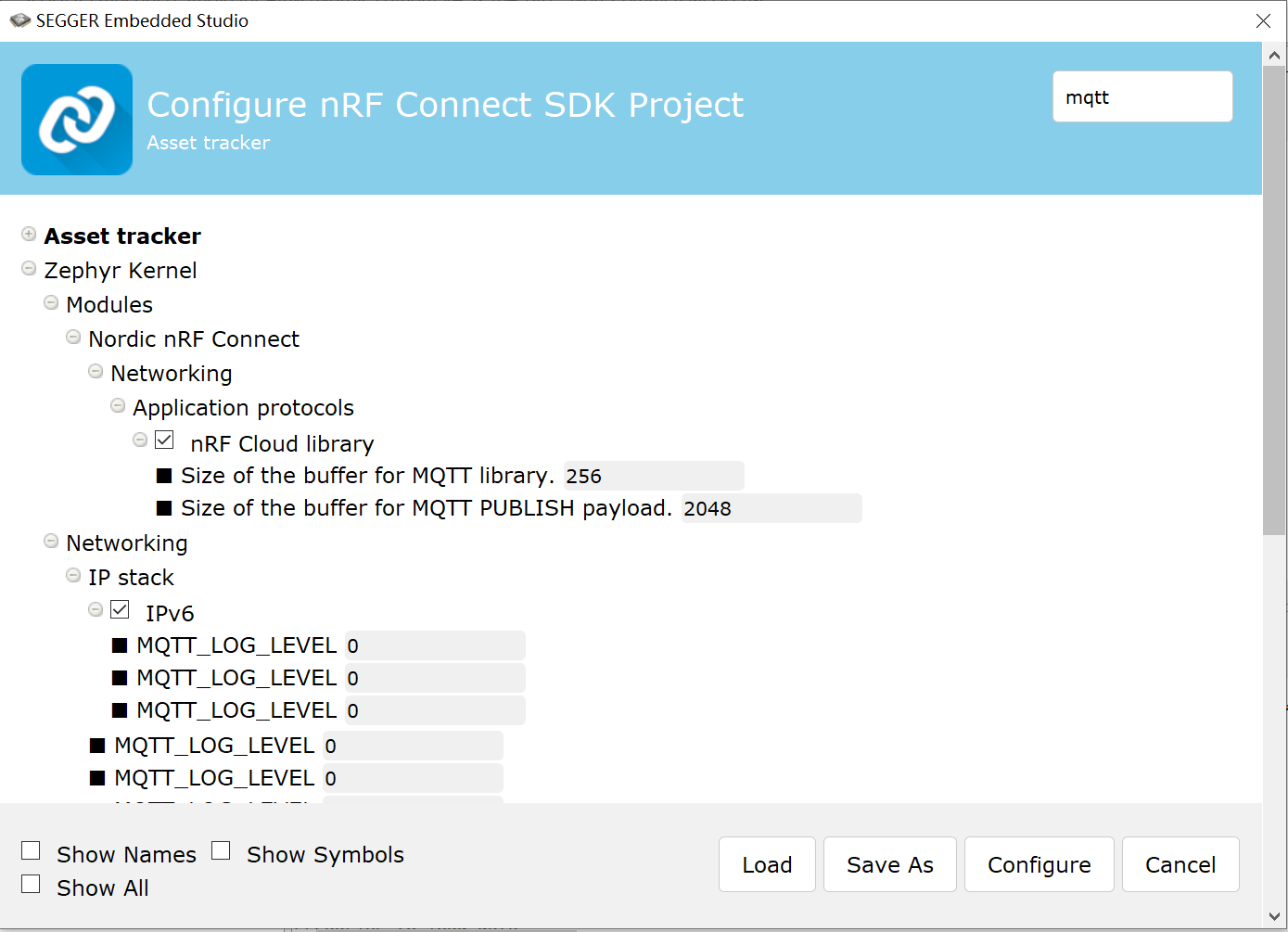


图 6 示例 mqtt 配置

# Linux下建立开发环境

命令行开发环境搭建，可参考 ：

https://developer.nordicsemi.com/nRF\_Connect\_SDK/doc/latest/nrf/gs\_installing.html#installing-the-required-tools

## 安装依赖

1. 更新系统，安装必要的工具

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install --no-install-recommends git cmake ninja-build gperf \

ccache dfu-util device-tree-compiler wget \

python3-dev python3-pip python3-setuptools python3-tk python3-wheel xz-utils file \

make gcc gcc-multilib g++-multilib libsdl2-dev

2. 安装cmake

cmake --version 查看版本，确保cmake 版本为3.13.1 或者更高

cd ~

wget https://github.com/Kitware/CMake/releases/download/v3.15.3/cmake-3.15.3-Linux-x86\_64.sh

chmod +x cmake-3.15.3-Linux-x86\_64.sh

sudo ./cmake-3.15.3-Linux-x86\_64.sh --skip-license --prefix=/usr/local

hash -r

3. 安装DTC (Device Tree Compiler)

dtc --version 查看版本，确保dtc版本为 1.4.6 或者更高

下载 DTC

wget <https://launchpadlibrarian.net/384086505/device-tree-compiler_1.4.7-1_amd64.deb>

安装

sudo dpkg -i device-tree-compiler\_1.4.7-1\_amd64.deb

4. 安装Python

确保python版本为3.6 或者 更高

sudo apt-get install python3.6

## 安装编译工具

1. 下载编译工具

wget <https://armkeil.blob.core.windows.net/developer/Files/downloads/gnu-rm/9-2020q2/gcc-arm-none-eabi-9-2020-q2-update-x86_64-linux.tar.bz2>

解压并添加环境变量

cd ~

mkdir gnuarmemb

tar -jxvf gcc-arm-none-eabi-9-2020-q2-update-x86\_64-linux.tar.bz2 --strip-components 1 ~/ gnuarmemb

echo 'export ZEPHYR\_TOOLCHAIN\_VARIANT=gnuarmemb

export GNUARMEMB\_TOOLCHAIN\_PATH="~/gnuarmemb"' > ~/.zephyrrc

1. 下载pebble 源码

源码下载详见3.1 pebble 源码下载

3. 安装python依赖

cd ncs

pip install -r zephyr/scripts/requirements.txt

pip install -r nrf/scripts/requirements.txt

pip install -r bootloader/mcuboot/scripts/requirements.txt

## 命令行配置

cd ncs

west build -t menuconfig

## 命令行编译

cd ncs/

source zephyr/zephyr-env.sh

cd ncs/nrf/applications/asset\_tracker

west build -b thingy91\_nrf9160ns

编译完成，ncs/nrf/applications/asset\_tracker/build/zephyr/merged.hex 文件即pebble 固件

## 烧录

1. 烧录固件有2 种模式，本部分介绍通过 JTAG 烧录程序到pebble, 下面 4.1 章节中介绍通过串口直接烧录固件。
2. 连接JTAG 并在下面页面下载、安装驱动:

<https://www.segger.com/downloads/jlink/#SystemView>

Jtag 连接图如下：



图 8 JTAG 连接

1. 在此页面下载Flash 烧录工具

<https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Command-Line-Tools/Download#infotabs>

这里选择nRFCommandLineTools1090Linuxamd64.tar.gz，下载解压后安装 Jlink 与 nrfjprog

sudo dpkg -i JLink\_Linux\_V680a\_x86\_64.deb

sudo dpkg -i nRF-Command-Line-Tools\_10\_9\_0\_Linux-amd64.deb

1. 命令行烧录 pebble 固件

pebble 上电，J-link 连接pebble 与 PC, 执行下面的命令烧录固件

cd ncs/nrf/applications/asset\_tracker

west flash

# Pebble App源码说明

## Pebble源码下载

从github上下载源码分为以下4步：

1. git clone https://github.com/iotexproject/pebble-firmware.git
2. git clone -b v2.3.0-rc1-ncs1 https://github.com/nrfconnect/sdk-zephyr zephyr
3. cd zephyr
4. git branch manifest-rev

## 源码目录

Pebble app 源码目录为” pebble-firmware\nrf\applications\asset\_tracker”, 各主要文件、目录介绍如下:

configuration : 分区管理配置文件，详细说明可参考:

“https://developer.nordicsemi.com/nRF\_Connect\_SDK/doc/latest/nrf/scripts/partition\_manager/partition\_manager.html”

src : pebble 源码目录

CMakeLists.txt ： cmake 构建文件

Kconfig ： menuconfig 配置文件

prj\_thingy91\_nrf9160ns.conf ：pebble 默认配置

spm.conf ： 安全分区管理配置

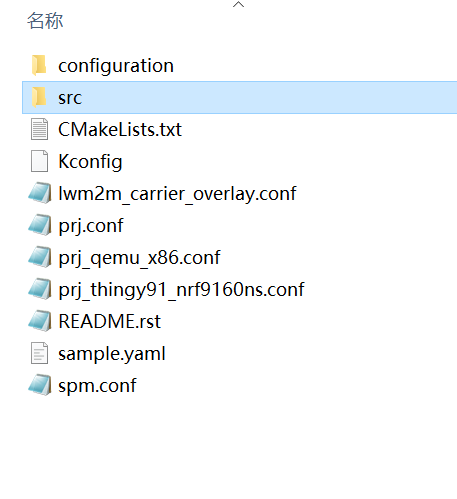


图 9 pebble app 源码目录

## pebble app 源码详细说明

### 程序流程图

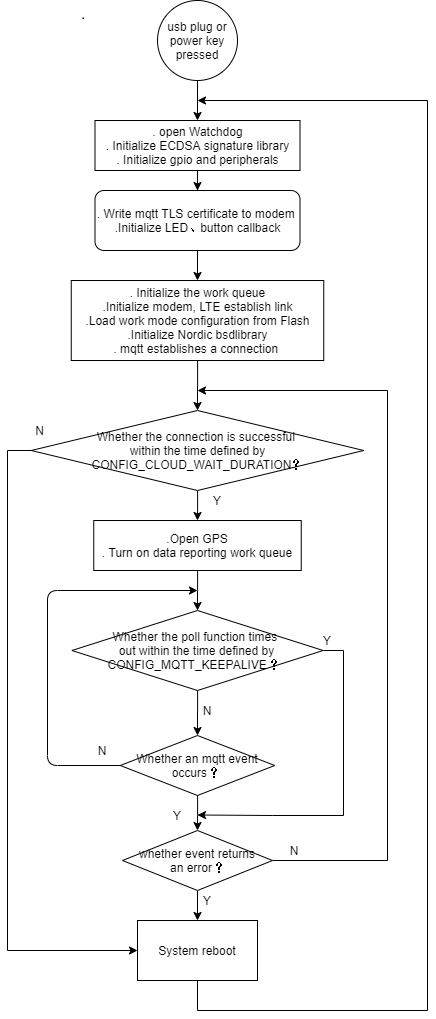


图 10 pebble app flow chart

### 部分API说明

|  |  |  |  |
| --- | --- | --- | --- |
| **函数名** | **参数与返回值** | | **功能说明** |
| modem\_configure | | 无 | 建立LTE 连接并设置LED指示灯 |
| iotex\_mqtt\_client\_init | | client : struct mqtt\_client 结构  fds : poll 函数使用的文件描述符  返回值: 连接正常返回 0 , 否则返回错误值 | 与Borker 建立连接，连接成功后订阅“topic/config/nrf-device\_id”, 关闭mqtt 连接超时后重启的work queue. |
| mqtt\_live | | client : iotex\_mqtt\_client\_init 函数中生成的mqtt\_client 结构  返回值：正常返回 0 或 -EAGAIN， 否则返回错误值 | 定时给Broker发布mqtt ping 数据包，定时时间由 CONFIG\_MQTT\_KEEPALIVE 定义。CONFIG\_MQTT\_KEEPALIVE 定义为0时不发送 mqtt ping 包。为节省流量，当发布数据的周期小于或者等于  CONFIG\_MQTT\_KEEPALIVE 定义的时间时， mqtt ping 将不会发送。 |
| iotex\_mqtt\_get\_selected\_payload | | channel: 用于指示哪些sensor 数据需要定时发布。  output: 指向json数据包  返回值： 返回0 表示正常， -ENOMEM 表示出错. | 将channel中指定sensor的数据读取出来，打包成jsong格式后通过 output 返回发给调用者。  注意： output->buf 指针在本函数中自动分配内存，需要在外部主动释放。 |
| doESDA\_sep256r\_Sign | | Inbuf: 指向待签名签名的字符串的指针  len: 待签名字符串长度  buf：签名后输出的r,s 数值，格式为16进制，buf[0] ~ buf[31] 为 r 数值， buf[32] ~ buf[63]为 s 数值。  sinlen: 为buf数据长度，通常为 64。  返回值： 0 正常，否则返回错误码。 | 使用用ARM TrustZone CryptoCell 310 对指定字符串实现 ECDSA spec256r 数字签名。 |
| iotex\_mqtt\_sampling\_data\_and\_store | | channel: 用于指示哪些sensor 数据需要存储。  返回值： 0 正常，否则返回错误码。 | 将channel中指定sensor的数据读取出来，存储到Flash中。 |
| iotex\_mqtt\_bin\_to\_json | | Buffer: 从Flash中读取出来，需要打包成json格式的数据  channel: 用于指示哪些sensor 数据需要打包。  Output： 打包的json数据  返回值： 0 正常，否则返回错误码。 | 将buffer 中数据打包成json格式，通常buffer中的数据是从 Flash 中读取出来，即  iotex\_mqtt\_sampling\_data\_and\_store 函数存入Flash 的数据。  注意： output->buf 指针在本函数中自动分配内存，需要在外部主动释放。 |
| iotex\_mqtt\_get\_topic | | 返回值： mqtt topic | 返回mqtt 数据上报的 topic |
| iotex\_mqtt\_get\_config\_topic | | 返回值： mqtt topic | 返回mqtt 订阅配置数据的 topic |

### 3.3.3 mqtt upload 数据包格式

Mqtt pubilish 数据采用 json 格式，如下图所示：



图 11 mqtt upload 数据包

各字段说明如下：

1. Sensor 数据，保存在 message 子对象中

● SNR： Number类型数据，指示NBIOT的信噪比

●VBAT： Number 类型浮点数据, 指示电池电压

●gas\_resistance： Number 类型数据，指示空气质量

●temperature： Number 类型数据，指示环境温度

●pressure： Number类型数据，指示大气压

●humidity： Number类型数据，指示环境温度

●light： Number类型数据，环境光数据

●gyroscope： Array类型数据，3 个数据分别表示陀螺仪的X，Y和Z轴角速率

●accelerometer: Array 类型数据，3 个数据分别表示陀螺仪的X，Y和Z轴加速度

●timestamp：String 类型数据，记录对sensot采样的时间

1. 签名数据

●signature\_r： ECDSA secp256r1 算法对 “message” 字段做的数字签名 r 值

●signature\_s： ECDSA secp256r1 算法对 “message” 字段做的数字签名 s 值

## 配置说明

### pebble工作模式配置

pebble 提供2中可配置的工作模式： a. 周期性数据上报模式； b. 缓存数据到 Flash中，缓冲区满后批量上传数据。通过订阅 “topic/config/nrf-device\_id”接收server端的配置，配置的json数据包格式如下：

{

  "bulk\_upload": 1,

  "data\_channel": 38,

  "upload\_period": 10,

  "bulk\_upload\_sampling\_cnt": 60,

  "bulk\_upload\_sampling\_freq": 10

}

对应的 C 数据结构为： config.h/iotex\_mqtt\_config

字段的含义分别是：是否打开批量上传、数据通道 Mask、 周期上传时上传周期(单位秒)、批量上传时采

样点数（采样点数达到后开始批量上传）、批量上传采样周期（批量上传时按照此周期进行数据的采样存

储，需要考虑 NVS 的容量）

data\_channel 是一下几种组合，例如： 38 ==> 0x26 ==> DATA\_CHANNEL\_TEMP |

DATA\_CHANNEL\_VBAT | DATA\_CHANNEL\_SNR

typedef enum {

  DATA\_CHANNEL\_GPS = 0x1,

  DATA\_CHANNEL\_SNR = 0x2,

  DATA\_CHANNEL\_VBAT = 0x4,

  DATA\_CHANNEL\_DEVINFO = 0x6,

  DATA\_CHANNEL\_GAS = 0X10,

  DATA\_CHANNEL\_TEMP = 0x20,

  DATA\_CHANNEL\_PRESSURE = 0x40,

  DATA\_CHANNEL\_HUMIDITY = 0X80,

  DATA\_CHANNEL\_ENV\_SENSOR = 0xf0,

  DATA\_CHANNEL\_TEMP2 = 0x100,

  DATA\_CHANNEL\_GYROSCOPE = 0x200,

DATA\_CHANNEL\_ACCELEROMETER = 0x400,

  DATA\_CHANNEL\_CUSTOM\_MOTION = 0x800,

  DATA\_CHANNEL\_ACTION\_SENSOR = 0xf00,

DATA\_CHANNEL\_LIGHT\_SENSOR = 0x1000,

} iotex\_data\_channel;

### GPS配置

Pebble 有片内GPS和片外GPS模块，可通过配置选择使用片内GPS或者外部GPS模块。选择片内GPS则在asset\_tracker\prj\_thingy91\_nrf9160ns.conf文件中增加定义:

CONFIG\_NRF9160\_GPS=y

CONFIG\_NRF9160\_GPS\_LOG\_LEVEL\_DBG=y

CONFIG\_GPS\_USE\_EXTERNAL=y

CONFIG\_GPS\_DEV\_NAME="NRF9160\_GPS"

CONFIG\_GPS\_CONTROL\_FIRST\_FIX\_CHECK\_DELAY=10

CONFIG\_GPS\_CONTROL\_FIX\_CHECK\_INTERVAL=30

CONFIG\_GPS\_CONTROL\_FIX\_TRY\_TIME=360

CONFIG\_GPS\_CONTROL\_PSM\_DISABLE\_ON\_STOP=n

去掉这部分定义则选择使用片外 GPS。 注意：使用片内 GPS 时，GPS与LTE不能同时开启。通过按住 button 3s 开启GPS，GPS开启后 LTE 连接将关闭。片外GPS 不受此限制，可同时使用 LTE 网络与 GPS。

### modem模式选择

modem可选择LTE-M与 NBIO 两种模式，每种模式可选择是否开启内部 GPS。

LTE-M 模式配置： CONFIG\_LTE\_NETWORK\_MODE\_LTE\_M=y

需要内部GPS时的配置为： CONFIG\_LTE\_NETWORK\_MODE\_LTE\_M\_GPS=y

NBIOT模式配置： CONFIG\_LTE\_NETWORK\_MODE\_NBIOT=y

NBIOT GPS模式：CONFIG\_LTE\_NETWORK\_MODE\_NBIOT\_GPS=y

### MQTT 证书内置

在 pebble\nrf\applications\asset\_tracker\src\certificates.h 文件中分别定义了

NRF\_CLOUD\_CLIENT\_PRIVATE\_KEY, NRF\_CLOUD\_CLIENT\_PUBLIC\_CERTIFICATE,

NRF\_CLOUD\_CA\_CERTIFICATE 用于存储MQTT Borker颁发的 TLS私有证书、共有证书、和CA 根证书。

在v1.3.0\nrf\applications\asset\_tracker\prj\_thingy91\_nrf9160ns.conf 文件中增加配置：

CONFIG\_USE\_PROVISIONED\_CERTIFICATES=n

CONFIG\_MODEM\_KEY\_MGMT=y

通过程序写入证书到modem 中， 默认情况写入证书功能是关闭的。

### ECDSA 数字签名证书内置

在pebble\nrf\applications\asset\_tracker\src\sign\ecdsa.c 中定义了

test\_vector\_ecdsa\_sign\_t test\_case\_ecdsa\_data 结构体，用于存储 ECDSA 数字签名证书。

其中

p\_qx : 存储公有证书的 x 分量

p\_qy：存储共有证书的 y 分量

p\_x : 存储私有证书

# 通过串口升级程序

## mcuboot recover 模式升级

1. 按住pebble 开机键，等待5s蓝色LED灯熄灭，pebble 进入recover 模式
2. usb 线连接pebble 与 PC, 打开 pc-programmer 工具
3. 配置 pc-programmer 后按如下步骤烧入程序，如下图：

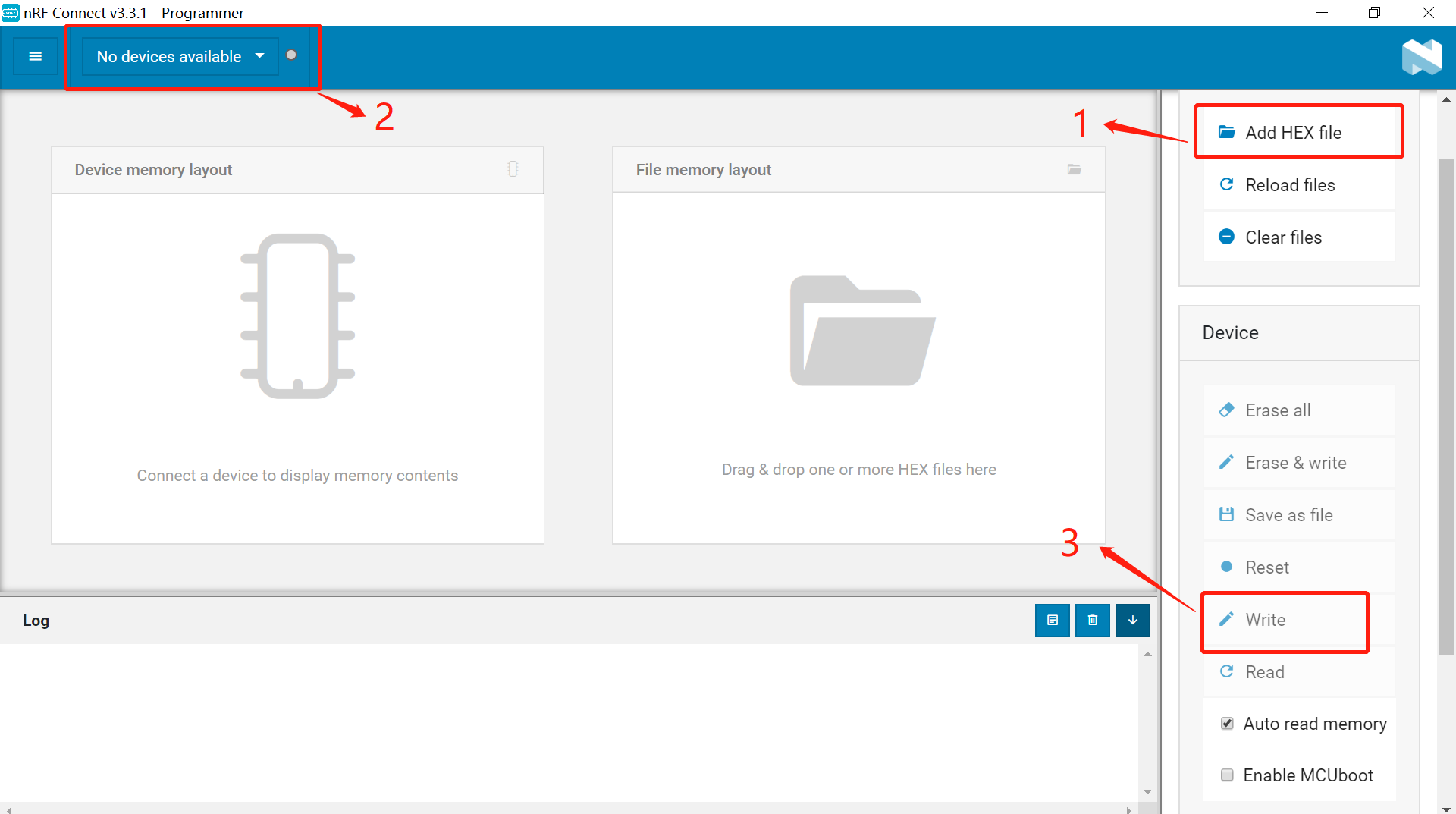


图 12 pebble mcuboot 升级

注意： Add HEX file 中选择“nrf\applications\asset\_tracker\build\_thingy91\_nrf9160ns\zephyr\app\_test\_update.hex” 文件

1. 升级过程中，红色LED 闪烁， 升级完成后设备自动启动
2. 如按键超过5s进入 recover 模式后不进行升级，pebble 等待 20s 后自动推出 recover 模式进入 app