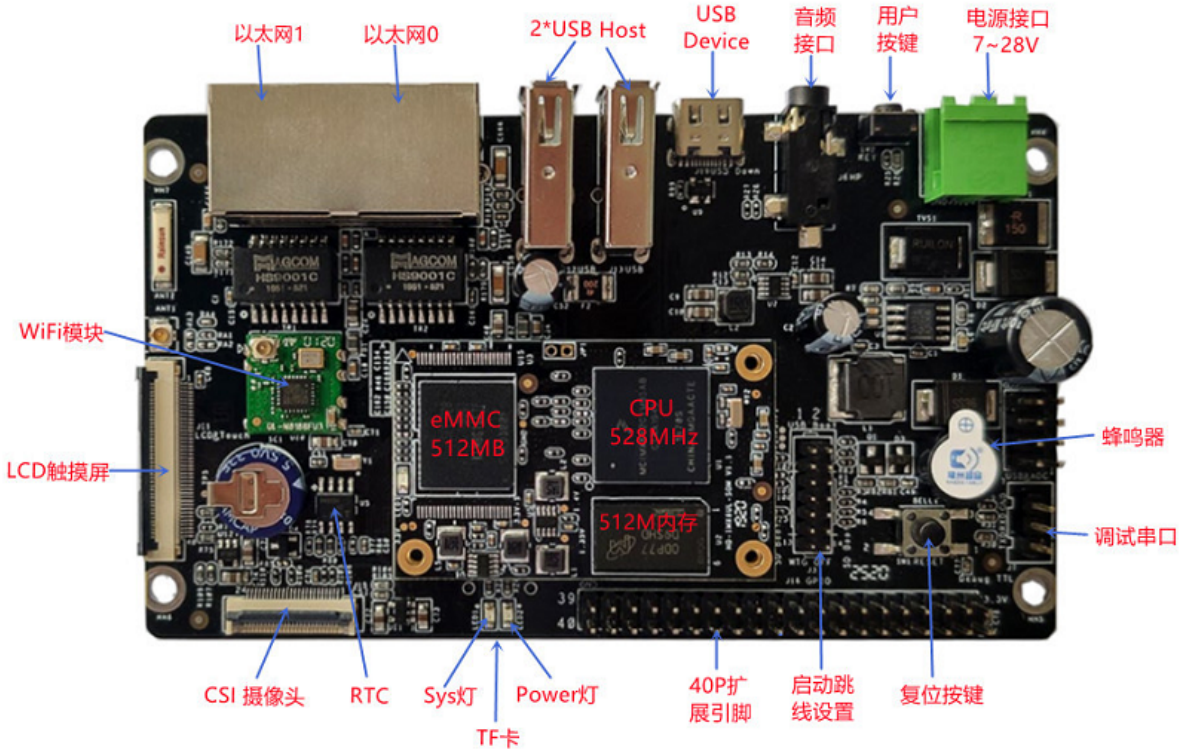
版权声明

- **Copyright (C) 2021 凌云物网智科实验室·郭工**
- **Author:　GuoWenxue <guowenxue@gmail.com>　QQ: 281143292**

1. 开发板硬件介绍

IGKBoard *(IoT Gateway Kit Board)* 开发板 是 **凌云物网智科实验室** 推出的一款ARM Linux物联网网关开发板。此开发板基于 ***NXP i.MX6ULL*** 系列 Cortex-A7 高性能处理器设计，适用于快速开发一系列具有创新性的产品如物联网网关、人机界面工业 4.0 扫描仪、车载终端以及便携式医疗设备。



1.1 硬件资源介绍

说明	芯片	参数	备注
电源供电		9V/1A	7~28V
处理器	MCIMX6Y2CVM05AB	工业级 528MHz	i.MX6ULL Cortex-A7
内存	MT41K256M16TW-107	512MB	DDR3L SDRAM
Flash	KLM8G1GETF-B041	8GB	Samsung eMMC 5.1
WiFi	板载 RealTek 8188FTV	USB2.0	2.4GHz 802.11b/g/n
以太网	2*LAN8720Ai	10/100Mbps	2 路以太网
音频	NAU8822L		1 路输出
RTC	ISL1208	I2C2	1*RTC 实时钟
LCD/TS	RGB565 接口	最高 1366*768	支持电容触摸屏
摄像头	1*CSI 接口		
USB 接口			2*USB Host, 1*USB Device
TF 接口			1*TF 卡
串口		UART1	调试输出串口
蜂鸣器			1*
Led 灯	1*Power 灯+1*SysRun 灯	GPIO4_IO16(SysRun)	高电平亮，低电平灭
按键	SW1 复位 SW2 用户	SNVS_TAMPER9	默认高电平，按下为低电平
尺寸		117mm*67mm	

1.2 扩展接口说明

对于CPU未使用完的多余的管脚，通过40P引脚扩展接引出。需要注意的是，扩展 IO 第 21 脚 LCD_DATA23 不可外接上拉电阻，否则会影响系统启动。这40P引脚定义如下：

sys/class 编 码	GPIO	功能名	物理引脚 BOARD编码		功能名	GPIO	sys/class 编 码
	3.3V	3.3V	1	2	5V	5V	
3	GPIO1_IO03	I2C1_SDA	3	4	5V	5V	
2	GPIO1_IO02	I2C1_SCL	5	6	GND	GND	
18	GPIO1_IO18	GPIO1_IO18	7	8	UART2_TX	GPIO1_IO20	20
	GND	GND	9	10	UART2_RX	GPIO1_IO21	21
24	GPIO1_IO24	UART3_TX	11	12	UART3_RX	GPIO1_IO25	25
28	GPIO1_IO28	UART4_TX	13	14	GND	GND	
29	GPIO1_IO29	UART4_RX	15	16	UART7_TX	GPIO3_IO21	85
	3.3V	3.3V	17	18	UART7_RX	GPIO3_IO22	86
91	GPIO3_IO27	ECSPI1_MOSI	19	20	GND	GND	
92	GPIO3_IO28	ECSPI1_MISO	21	22	GPIO3_IO23	GPIO3_IO23	87
89	GPIO3_IO25	ECSPI1_SCLK	23	24	ECSPI1_SS0	GPIO3_IO26	90
	GND	GND	25	26	GPIO3_IO24	GPIO3_IO24	88
26	GPIO1_IO26	CAN1_TX	27	28	PWM8	GPIO1_IO15	15
27	GPIO1_IO27	CAN1_RX	29	30	GND	GND	
22	GPIO1_IO22	CAN2_TX	31	32	PWM7	GPIO1_IO14	14
23	GPIO1_IO23	CAN2_RX	33	34	GND	GND	
129	GPIO5_IO01	GPIO5_IO01	35	36	GPIO1_IO11	GPIO1_IO11	11
136	GPIO5_IO08	GPIO5_IO08	37	38	GPIO1_IO10	GPIO1_IO10	10
	GND	GND	39	40	GPIO5_IO09	GPIO5_IO09	137

1.3 跳线帽子说明

J3跳线帽子用来选择启动方式，下图描述了开发板具体启动模式。注意由于核心板上默认关闭了 WDG 功能，所以 WDG 功能无法使用跳线帽开启。

序号 \ 模式	eMMC 启动 NAND 启动	SD 卡启动	升级 eMMC	升级 SD 卡
1 2				
3 4				
5 6				
7 8				
9 10				
11 12				

序号 \ WDG	使能 WDG	关闭 WDG
13 14		

 ： 跳线帽连接

 ： 跳线帽断开

2. 项目源码介绍

凌云实验室针对该开发板从制作交叉编译器开始，从零构建了Buildroot、Yocto、Debian等Linux系统，所有系统相关源码维护在凌云实验室的官方git服务器上，接下来我们将讲解如何使用git仓库上的源码搭建交叉编译环境、编译制作系统镜像文件。

2.1 项目源码介绍

该项目测试开发环境为Ubuntu-20.04， 首先在Linux服务器上使用 **git** 命令克隆下载该项目源码，如果没有 ***git*** 命令，则首先安装 **git** 命令并配置 **git** 的账号信息。

```
guowenxue@ubuntu20:~$ sudo apt install git
guowenxue@ubuntu20:~$ git config --global user.name guowenxue
guowenxue@ubuntu20:~$ git config --global user.email "guowenxue@gmail.com"
```

使用 **git** 命令克隆下载该项目源码：

```
guowenxue@ubuntu20:~$ mkdir -p ~/workspace
guowenxue@ubuntu20:~$ cd ~/workspace
guowenxue@ubuntu20:~/workspace$ git clone git://weike-iot.com/imx6ull.git
Cloning into 'imx6ull'...
remote: Counting objects: 330, done
remote: Finding sources: 100% (330/330)
remote: Getting sizes: 100% (179/179)
remote: Total 330 (delta 98), reused 330 (delta 98)
Receiving objects: 100% (330/330), 110.55 KiB | 884.00 KiB/s, done.
Resolving deltas: 100% (98/98), done.
```

下面是该项目当前源码目录结构的介绍：

```
guowenxue@ubuntu20:~$ cd ~/workspace/imx6ull/
guowenxue@ubuntu20:~/workspace/imx6ull$ tree -L 2
.
├── README.md              本项目介绍文档
├── 3rdparty               第三方开源库的交叉编译脚本
│   ├── evtest             输入设备驱动测试工具程序
│   └── QT5                知名的图形化界面开发库
├── bsp                    Linux板级支持包源码目录
│   ├── build.sh           BSP目录总的编译脚本，依次进入到子目录下编译
│   ├── bootloader         U-boot编译目录及编译脚本
│   ├── images             制作系统镜像的脚本
│   ├── kernel             Linux内核编译目录及编译脚本
│   ├── rootfs             根文件系统制作脚本
│   ├── tarball            Linux内核等源码下载目录及脚本
│   └── uapi               Linux内核驱动测试脚本
├── documents              文档目录
│   ├── datasheet          芯片datasheet
│   ├── documents          使用手册说明文档
│   ├── NXP                NXP官方文档
│   └── schematic          开发板原理图
├── program                自己编写的一些应用程序
│   └── lvgl               另外一个跨平台、轻量级图形库
└── yocto                  Yocto项目支持目录
│   └── meta-igkboard      Yocto移植的开发板Meta-layer
└── tools                  本项目开发工具
    ├── buildroot          buildroot交叉编译器一键制作脚本
    └── setup_tools.sh     开发环境安装脚本

16 directories, 3 files
```

2.2 编译环境安装

该项目源码路径 tools下有该项目的开发环境一键安装Shell脚本，以root权限执行该脚本将会自动下载并安装接下来开发过程中所依赖的系统命令工具，并解压缩安装交叉编译器到 ***/opt/buildroot/cortexA7*** 路径下。

```
guowenxue@ubuntu20:~$ cd ~/workspace/imx6ull/tools/
```

```
guowenxue@ubuntu20:~/workspace/imx6ull/tools$ ls
buildroot  setup_tools.sh

guowenxue@ubuntu20:~/workspace/imx6ull/tools$ sudo ./setup_tools.sh

+------------------------------------------------------------------------
|  /apps and /opt not writable, use chmod to give writable permissions
+------------------------------------------------------------------------


+------------------------------------------------------------------------
|  start apt install system tools(commands)
+------------------------------------------------------------------------

... ...

+------------------------------------------------------------------------
|  start apt install devlopment tools(commands)
+------------------------------------------------------------------------

... ...

+------------------------------------------------------------------------
|  download buildroot-2021.02.7-cortexA7 now...
+------------------------------------------------------------------------

... ...
+------------------------------------------------------------------------
|  install buildroot-2021.02.7-cortexA7 to /opt/buildroot/cortexA7 now...
+------------------------------------------------------------------------

Using built-in specs.
COLLECT_GCC=/opt/buildroot/cortexA7/bin/arm-linux-gcc.br_real
COLLECT_LTO_WRAPPER=/opt/buildroot/cortexA7/libexec/gcc/arm-buildroot-linux-
gnueabihf/9.4.0/lto-wrapper
Target: arm-buildroot-linux-gnueabihf
Configured with: ./configure --prefix=/opt/buildroot/cortexA7 --
sysconfdir=/opt/buildroot/cortexA7/etc --enable-static --target=arm-buildroot-
linux-gnueabihf --with-sysroot=/opt/buildroot/cortexA7/arm-buildroot-linux-
gnueabihf/sysroot --enable-__cxa_atexit --with-gnu-ld --disable-libssp --disable-
multilib --disable-decimal-float --with-gmp=/opt/buildroot/cortexA7 --with-
mpc=/opt/buildroot/cortexA7 --with-mpfr=/opt/buildroot/cortexA7 --with-
pkgversion='Buildroot 2021.02.7' --with-bugurl=http://bugs.buildroot.net/ --
without-zstd --disable-libquadmath --disable-libquadmath-support --enable-tls --
enable-threads --without-isl --without-cloog --with-abi=aapcs-linux --with-
cpu=cortex-a7 --with-fpu=neon-vfpv4 --with-float=hard --with-mode=arm --enable-
languages=c,c++ --with-build-time-tools=/opt/buildroot/cortexA7/arm-buildroot-
linux-gnueabihf/bin --enable-shared --disable-libgomp
Thread model: posix
gcc version 9.4.0 (Buildroot 2021.02.7)

+------------------------------------------------------------------------
|  Cross compiler: /opt/buildroot/cortexA7/bin/arm-linux-
+------------------------------------------------------------------------
```

2.3 自构建系统编译

2.3.1 自构建系统介绍

该项目源码路径 bsp/ 文件夹下存放有U-boot、Linux内核、根文件系统树等编译制作脚本，同时顶层还有一个build.sh 的Shell脚本。该脚本用来一键编译所选择的目标源码，并将编译产生的烧录文件放到新生成的 images路径下。

```
guowenxue@ubuntu20:~$ cd ~/workspace/imx6ull/bsp/
guowenxue@ubuntu20:~/workspace/imx6ull/bsp$ tree -L 2
.
├── build.sh                    总的编译脚本
├── bootloader
│   ├── build.sh                u-boot编译脚本
│   └── patch                   u-boot修改补丁文件
├── kernel
│   ├── build.sh                Linux内核编译脚本
│   └── patch                   Linux内核修改补丁文件
├── drivers
│   ├── build.sh                Linux内核驱动编译脚本，WiFi模块驱动将会自动下载并编译
│   └── users                   用户编写的驱动文件将会放到这里
├── images
│   ├── build.sh                烧录的系统镜像制作脚本
│   └── wintools                Windows系统下系统镜像烧录工具和脚本
├── rootfs
│   ├── build.sh                根文件系统树制作脚本
│   └── extra_apps.json         debian文件系统额外安装的应用程序配置文件
├── tarball
│   └── build.sh                Linux内核、u-boot、根文件系统树等源码包自动下载脚本
└── uapi
    └── test                    驱动测试程序
```

下面是总的编译脚本的使用帮助信息：

```
guowenxue@ubuntu20:~/workspace/imx6ull/bsp$ ./build.sh

Usage:
 ./build.sh [-b] [-c] [-h] bsp/image
     -b bsp/image: build BSP or system image
     -c bsp/image: clean BSP or system image
     -h          : show this help message

 WARNNING: build/clean image need run as sudo

  Example: ./build.sh -b bsp && sudo ./build.sh -b image
```

2.3.2 系统类型介绍

在总的编译脚本里，变量 **ROOTFS** 用来指定要编译的Linux系统类型，当前支持 ***buildroot***、***yocto***、***bullseye (Debian11)*** 系统。

```
guowenxue@ubuntu20:~/workspace/imx6ull/bsp$ vim build.sh
... ...
# rootfs should be buildroot/yocto or bullseye for debian system
ROOTFS=buildroot
... ...
```

- 如果想使用 **buildroot** 制作的文件系统，则将其设置为 ***buildroot*** ，默认为该值；
- 如果想使用 **yocto** 制作的文件系统，则将其值更改为 **yocto** ；
- 如果想使用 ***debian*** 文件系统，则将其值设置为 **bullseye** ，即 **Debian 11** ；

2.3.3 自构建系统编译

使用下面命令，一键下载、编译并制作系统烧录镜像文件，默认将编译 ***buildroot*** 系统镜像：

```
guowenxue@ubuntu20:~/workspace/imx6ull/bsp$ ./build.sh -b bsp && sudo ./build.sh
-b image
... ...
+----------------------------------------------+
 Stage 5:  <4> Install u-boot image
+----------------------------------------------+

1006+0 records in
1006+0 records out
515072 bytes (515 kB, 503 KiB) copied, 0.00320756 s, 161 MB/s

+----------------------------------------------+
 Stage 6:  <5> Install linux kernel image
+----------------------------------------------+

+ cp /home/guowenxue/workspace/imx6ull/bsp/images/boot/zImage ./mnt
+ cp /home/guowenxue/workspace/imx6ull/bsp/images/boot/igkboard-emmc.dtb ./mnt
+ sync
+ set +x

+----------------------------------------------+
 Stage 7:  <6> Install root filesystem
+----------------------------------------------+


+----------------------------------------------+
 Stage 8:  bzip2 compress system image
+----------------------------------------------+


 -- generate system image done --

Shell script exit now, do some clean work

guowenxue@ubuntu20:~/workspace/imx6ull/bsp$ cd images/
```

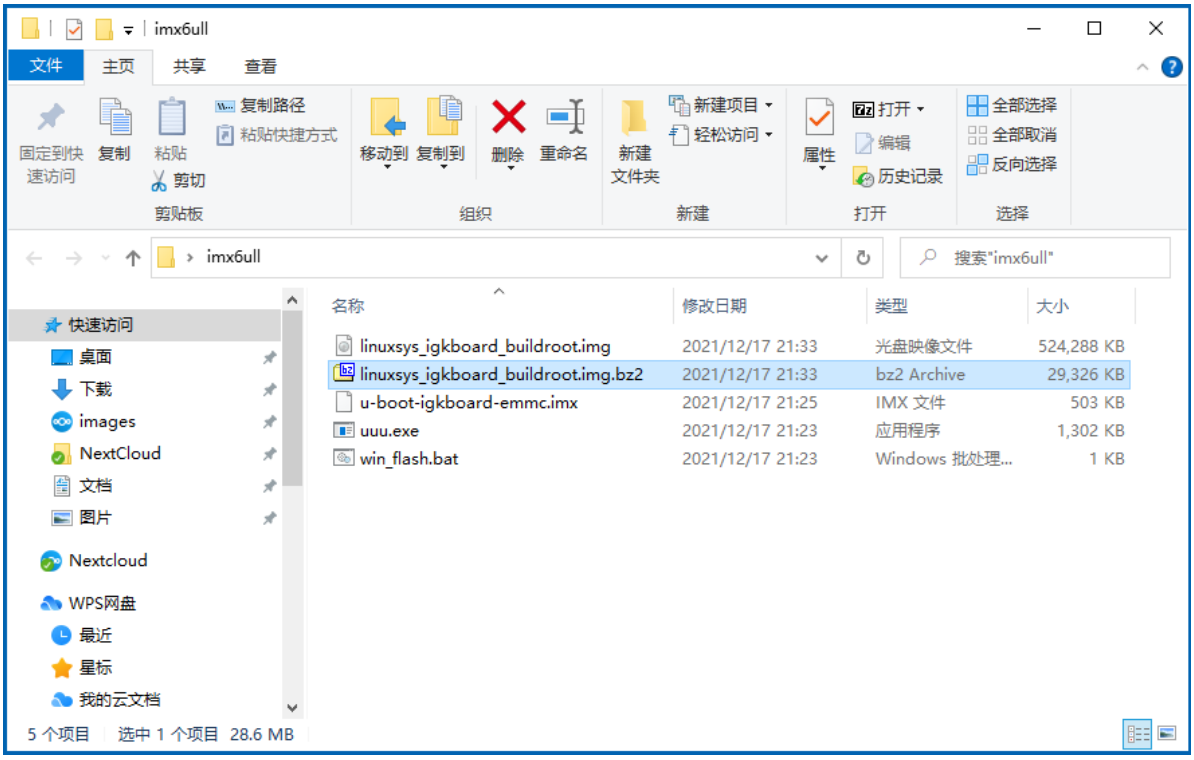
```
guowenxue@ubuntu20:~/workspace/imx6ull/bsp/images$ tree
.
├── boot
│   ├── igkboard-emmc.dtb                    Linux内核DTB文件
│   └── zImage                               Linux内核Image
├── build.sh
├── linuxsys_igkboard_buildroot.img.bz2      烧录的系统镜像bz2压缩文件，使用的文件系统不一样，系统镜像名也不一样。
├── u-boot-igkboard-emmc.imx                 U-boot EMMC启动镜像文件
└── wintools
    ├── uuu.exe                              Windows下UUU烧录工具
    └── win_flash.bat                        Windows下的烧录批处理脚本

2 directories, 7 files
```

2.3.4 编译输出文件

这里，我们需要将如下几个文件下载/拷贝到Windows主机下，**并将系统镜像文件解压缩出来**，接下来烧录系统镜像时会用到。

- linuxsys_igkboard_buildroot.img.bz2　　烧录的系统镜像bz2压缩文件
- u-boot-igkboard-emmc.imx　　U-boot EMMC启动镜像文件
- uuu.exe　　Windows下UUU烧录工具
- win_flash.bat　　Windows下的烧录批处理脚本



2.4 Yocto系统编译

凌云实验室针对IGKBoard开发板移植了Yocto Hardknott (Yocto 3.3)系统，该源码也托管在上面的 git 服务器上，接下来将讲解如何编译该系统。关于Yocto系统的详细使用，大家可以参考NXP官方的用户手册文档，点此链接下载 。

如果想源码编译Yocto系统的话，则系统需要满足：

- 推荐使用 Ubuntu 20.04 系统；
- 推荐使用 4核以上CPU；
- 硬盘空间至少要求 200GB+；

2.4.1 下载Yocto项目源码

使用 **git** 命令克隆凌云实验室 ***IGKBoard*** 项目源码，如果之前已下载则可以跳过该步骤。

```
guowenxue@ubuntu20:~$ mkdir -p ~/workspace
guowenxue@ubuntu20:~$ cd ~/workspace
guowenxue@ubuntu20:~/workspace$ git clone git://weike-iot.com/imx6ull.git
Cloning into 'imx6ull'...
remote: Counting objects: 330, done
remote: Finding sources: 100% (330/330)
remote: Getting sizes: 100% (179/179)
remote: Total 330 (delta 98), reused 330 (delta 98)
Receiving objects: 100% (330/330), 110.55 KiB | 884.00 KiB/s, done.
Resolving deltas: 100% (98/98), done.
```

下载并安装 google 的 ***repo*** 命令:

```
guowenxue@ubuntu20:~$ curl https://storage.googleapis.com/git-repo-downloads/repo
-o repo
guowenxue@ubuntu20:~$ chmod a+x repo
guowenxue@ubuntu20:~$ sudo mv repo /usr/bin/
```

接下来就可以使用 ***repo*** 命令下载 **Yocto Hardknott** 源码：

```
guowenxue@ubuntu20:~$ mkdir -p ~/workspace/imx-yocto-bsp/
guowenxue@ubuntu20:~$ cd ~/workspace/imx-yocto-bsp/
guowenxue@ubuntu20:~/workspace/imx-yocto-bsp$ repo init -u
https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-hardknott -m
imx-5.10.52-2.1.0.xml
guowenxue@ubuntu20:~/workspace/imx-yocto-bsp$ repo sync
guowenxue@ubuntu20:~/workspace/imx-yocto-bsp$ ls
imx-setup-release.sh  README  README-IMXBSP  setup-environment  sources
```

2.4.2 编译Yocto系统

在前面的开发板项目源码中，有提供凌云实验室 IGKBoard 开发板的Yocto支持源码，这里首先将该源码拷贝到 Yocto 系统中。

```
guowenxue@ubuntu20:~/workspace/imx-yocto-bsp$ cp -rf
~/workspace/imx6ull/yocto/meta-igkboard/ sources/
```

接下来执行下面命令，初始化 IGKBoard 开发板的 Yocto 编译环境：

```
guowenxue@ubuntu20:~/workspace/imx-yocto-bsp$ MACHINE=igkboard source
sources/meta-igkboard/tools/igkboard-setup.sh -b igkboard

 Build directory is  igkboard
/home/guowenxue/workspace/imx-yocto-bsp
    Welcome LingYun IoT Gateway Kit Board Yocto BSP

    The Yocto Project has extensive documentation about OE including a
    reference manual which can be found at:
        http://yoctoproject.org/documentation

    You can now run 'bitbake <target>'

    Common targets are:
        linuxsys-image
        core-image-minimal
        imx-image-full
```

接下来使用下面命令，开始Yocto系统的源码编译，这个系统编译时间较长，如果编译的过程中出现错误，再重新开始编译即可。也可以使用 ***-k*** 选项先跳过编译出错的软件包。

```
guowenxue@ubuntu20:/workspace/imx-yocto-bsp/igkboard$ bitbake linuxsys-image
```

2.4.3 Yocto编译输出

编译过程中下载的软件包存放在 ***~/workspace/hardknott_packets/*** 路径下:

```
guowenxue@ubuntu20:/workspace/imx-yocto-bsp/igkboard$ ls
~/workspace/hardknott_packets/
```

编译完成输出的系统镜像及文件系统存放在下面路径下：

```
guowenxue@ubuntu20:~/workspace/imx-yocto-bsp/igkboard$ ls ls
tmp/deploy/images/igkboard/
linuxsys-image-igkboard-*.rootfs.tar.bz2   --Yocto根文件系统压缩包
linuxsys-image-igkboard-*.rootfs.wic.bz2   --Yocto烧录系统镜像文件
```

3. 系统镜像烧录

3.1 跳线设置

iMX6ULL支持eMMC、Nandflash、TF卡(SD卡)等多种启动方式，**IGKBoard开发板** 上板载了一颗Samsung公司的8GB eMMC芯片，此外它还带有一个TF卡槽，这样该开发板支持eMMC和SD卡两种启动方式。由下图可知，系统具体采用哪种方式启动由J3跳线帽子决定。

模式 / 序号	eMMC 启动 NAND 启动	SD 卡启动	升级 eMMC	升级 SD 卡
1 2				
3 4				
5 6				
7 8				
9 10				
11 12				

WDG / 序号	使能 WDG	关闭 WDG
13 14		

：跳线帽连接

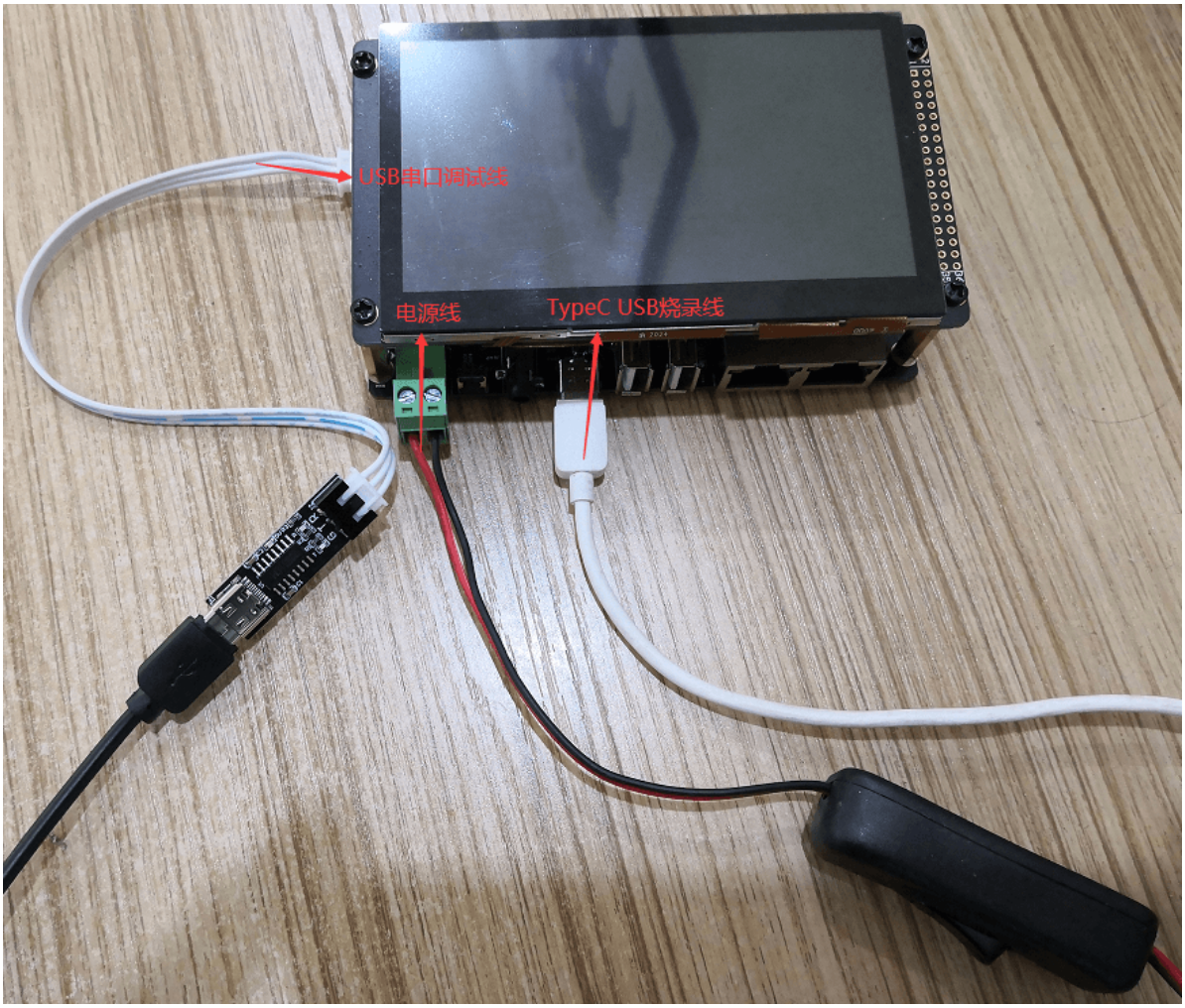
：跳线帽断开

- 如果想设置系统从 EMMC启动(mmc1)，则将除看门狗跳线(13、14)短接以外，所有其他跳线帽子全部断开；
- 如果想设置系统从TF卡启动(mmc0)，则将 5/6、7/8、9/10、11/12 、13/14(看门狗跳线) 5个跳线全部短接；

3.2 硬件连接

如下图所示，连接相关的硬件接口设置：

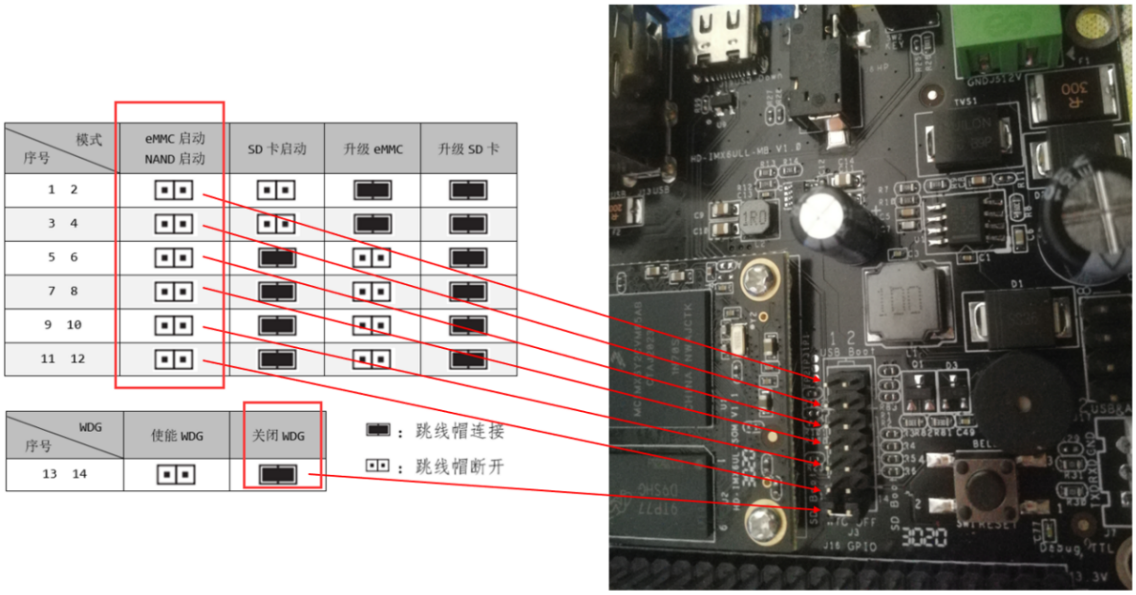
- 使用 ***9V/1A*电源**（7~28V电压范围）供电；
- 使用 ***TypeC USB*线** 连接开发板和PC，Win10系统自带有其驱动，该接口专门用来烧录 ***u-boot*** 或 **系统镜像** ；
- 使用 ***USB*串口调试线** 连接开发板和PC，该模块使用CH340 USB转串口芯片，需要安装其驱动。



3.3 EMMC系统烧录

3.3.1 跳线设置

如下图所示，将J3的除WDG以外的跳线帽子全部断开，则进入eMMC启动模式，该模式下开发板上电后将从eMMC读取运行u-boot，进而加载并启动Linux系统。

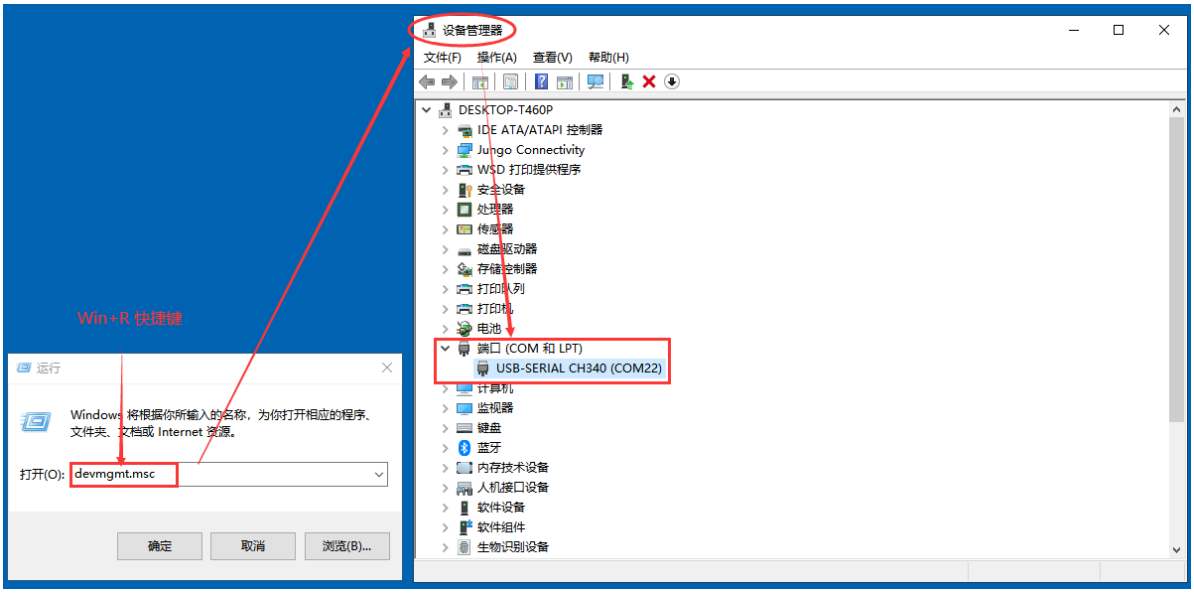


3.3.2 软件准备

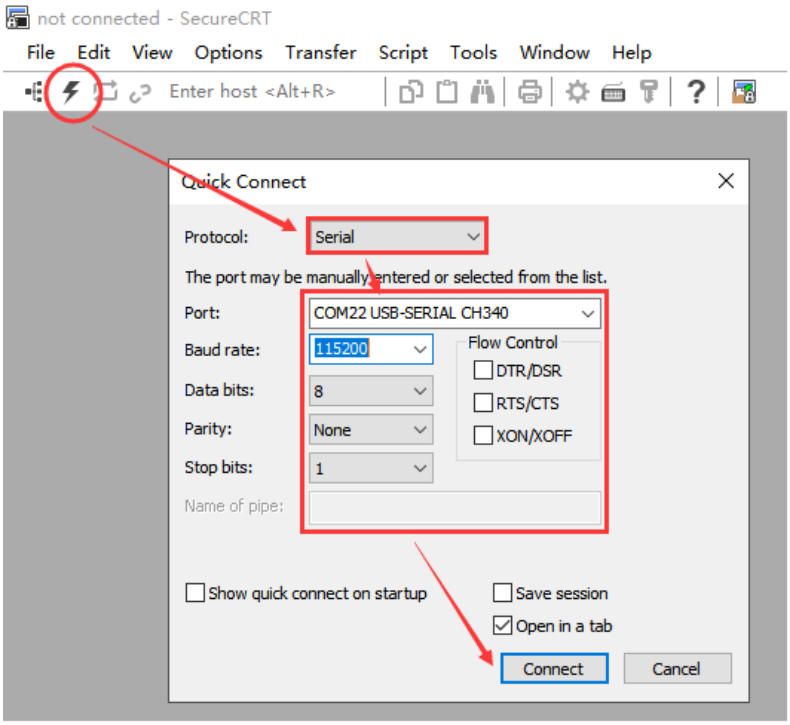
在硬件设备准备好之后，我们还需要安装或准备好如下软件：

- **点击此链接下载** 并安装好 ***USB转串口调试器*** 驱动；
- **点击此链接下载** NXP官方烧录程序 ***uuu.exe***， 也可以从前面的项目源码路径 ***imx6ull/bsp/images/wintools/*** 中获取；
- **点击此链接下载** 凌云实验室的 i.MX6ULL开发板烧录批处理脚本 ***win_flash.bat***，也可以从前面的项目源码路径 ***imx6ull/bsp/images/wintools/*** 中获取 。

串口驱动下载安装好后，将 ***USB转串口调试器*** 接入 使用 ***Win+R*** 快捷键打开 **运行** ， 然后输入 **devmgmt.msc** 命令打开 **设备管理器**， 接下来我们应该可以看到相应的串口设备文件。如下图所示：

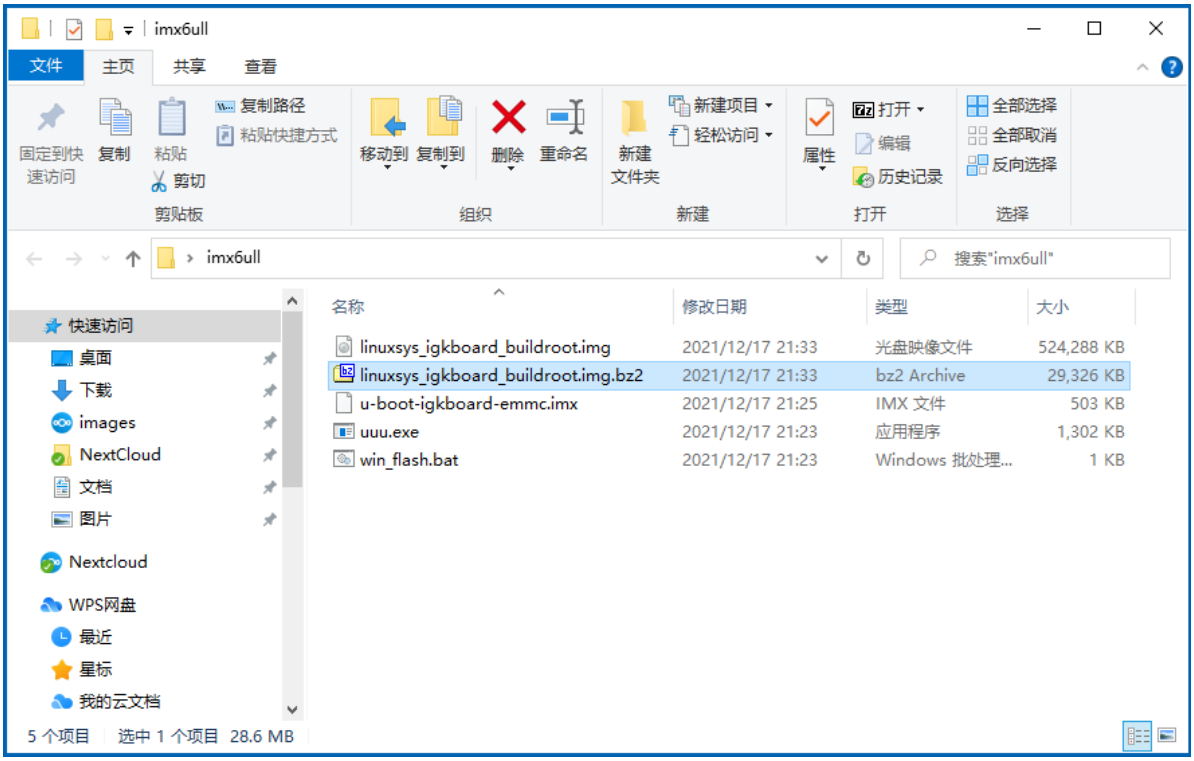


接下来使用 **SecureCRT** 或其他串口调试软件，打开相应的USB转串口设备，并监听串口：



3.3.3 烧录文件

如前面 **自构建系统编译** 相关章节所述，将前面源码编译输出的相关文件 拷贝/下载 到Windows系统下，并将系统镜像文件 解压缩出来。这里以 ***buildroot*** 系统镜像 ***linuxsys_igkboard_buildroot.img*** 文件为例，详细讲解 ***emmc* 系统** 的烧录过程，其它的系统镜像同样也可以使用同样的方式来烧录。



IGKBoard-iMX6ULL 开发板 当前支持 ***Buildroot***、***Yocto***、***Debian Bullseye*** 等不同的文件系统，编译的时候使用的文件系统不一样，生成的系统镜像也不一样。如果系统镜像不一样，可以修改 ***win_flash.bat*** 批处理脚本中的 ***ROOTFS*** 变量值:

```
REM This bat script used to flash u-boot or linux system image to EMMC

REM uuu.exe official address:  https://github.com/NXPmicro/mfgtools
REM LingYun download address:  http://weike-iot.com:2211/imx6ull/tools

REM Goes into fastboot mode command:  fastboot 0
REM Erase u-boot command in u-boot :  mmc dev 1 1 && mmc erase 0 40000

@echo off

set BOARD=igkboard
set ROOTFS=buildroot

set IMAGE_UBOOT=u-boot-%BOARD%-emmc.imx
set IMAGE_SYS=linuxsys_%BOARD%_%ROOTFS%.img

echo "Image Download Choices:"

echo 1: Download bootloader   [ %IMAGE_UBOOT% ]
echo 2: Download System Image [ %IMAGE_SYS% ]

set /p choice= Please Input Your Choice:

if %choice% == 1  uuu -b emmc %IMAGE_UBOOT%
if %choice% == 2  uuu -b emmc_all %IMAGE_UBOOT% %IMAGE_SYS%
```

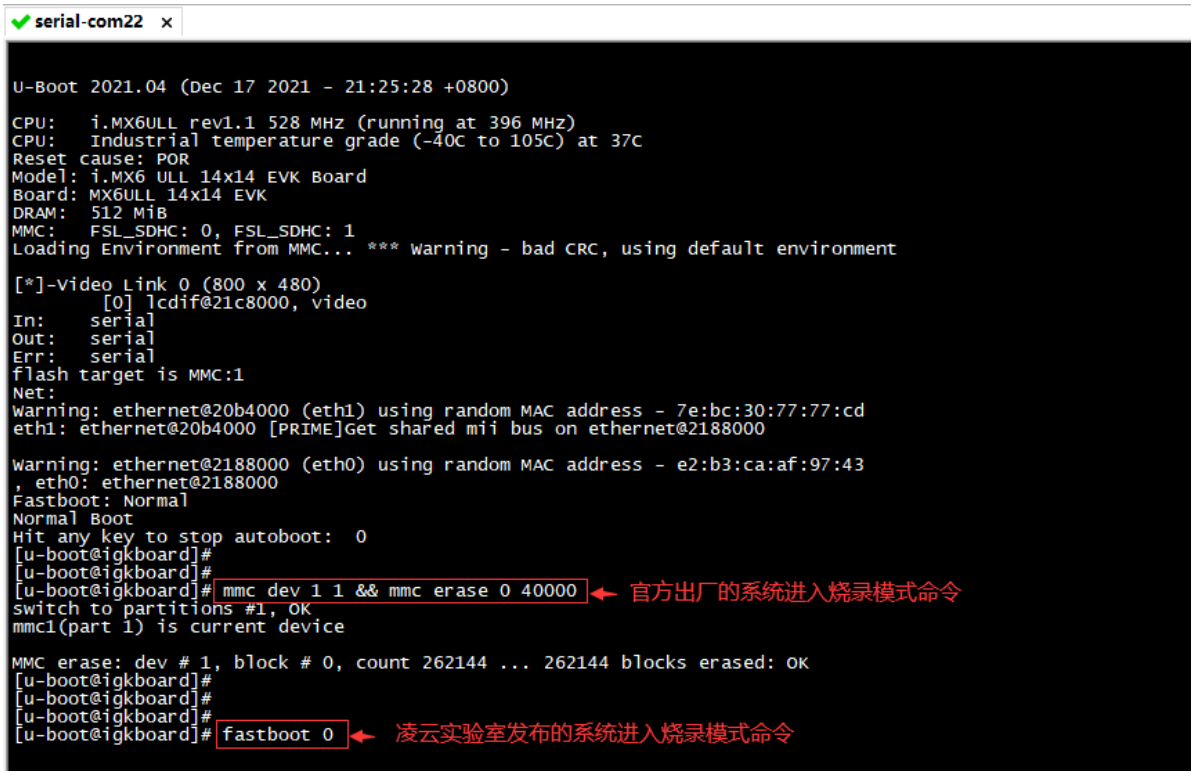
3.3.4 烧录模式

要想重新烧录开发板，则需要让开发板首先进入到烧录模式，然后通过 USB TypeC接口烧录。不同情况下进入烧录模式不一样：

- 硬件开发板刚生产出来时，并没有系统软件，这时将会默认进入到烧录模式；
- 如果已经烧录了u-boot程序，系统上电后出现 ***"Hit any key to stop autoboot: 3"*** 时按任意键进入到u-boot的 **调试模式**.

在u-boot调试模式下，输入下面命令将进入到开发板的烧录模式：

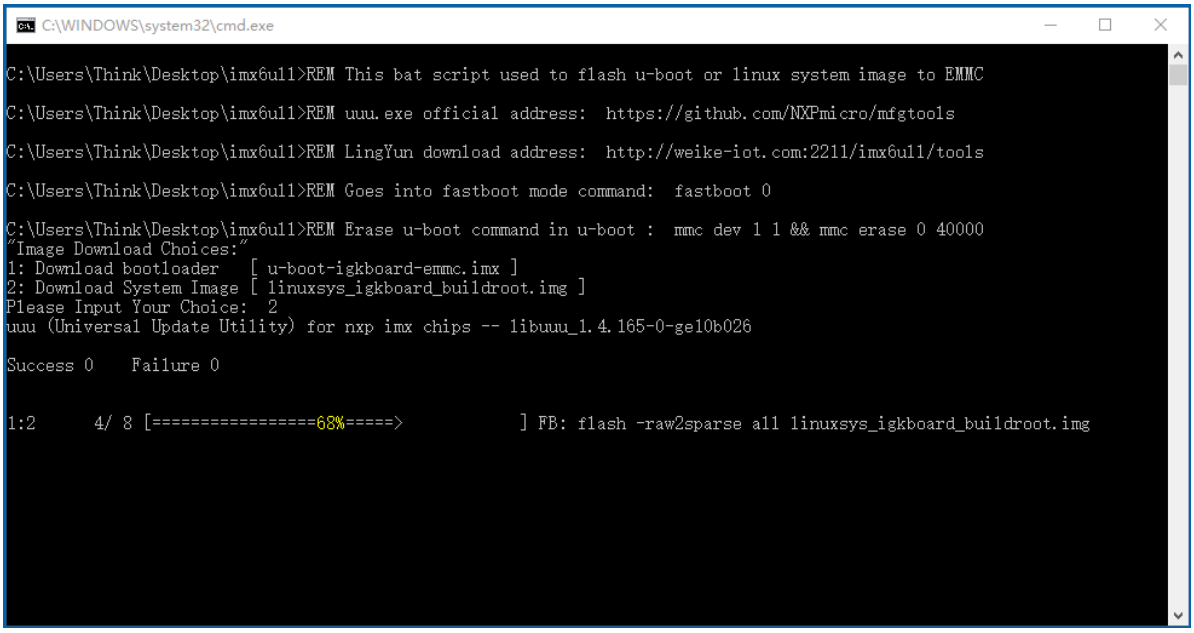
- 如果开发板已经烧录了万象奥科的出厂系统，则需要使用 ***mmc dev 1 1 && mmc erase 0 40000*** 命令先擦除 u-boot 后再重启进入到烧录模式；
- 如果开发板已经烧录了凌云实验室的系统，则可以使用 ***fastboot 0*** 命令进入到烧录模式；



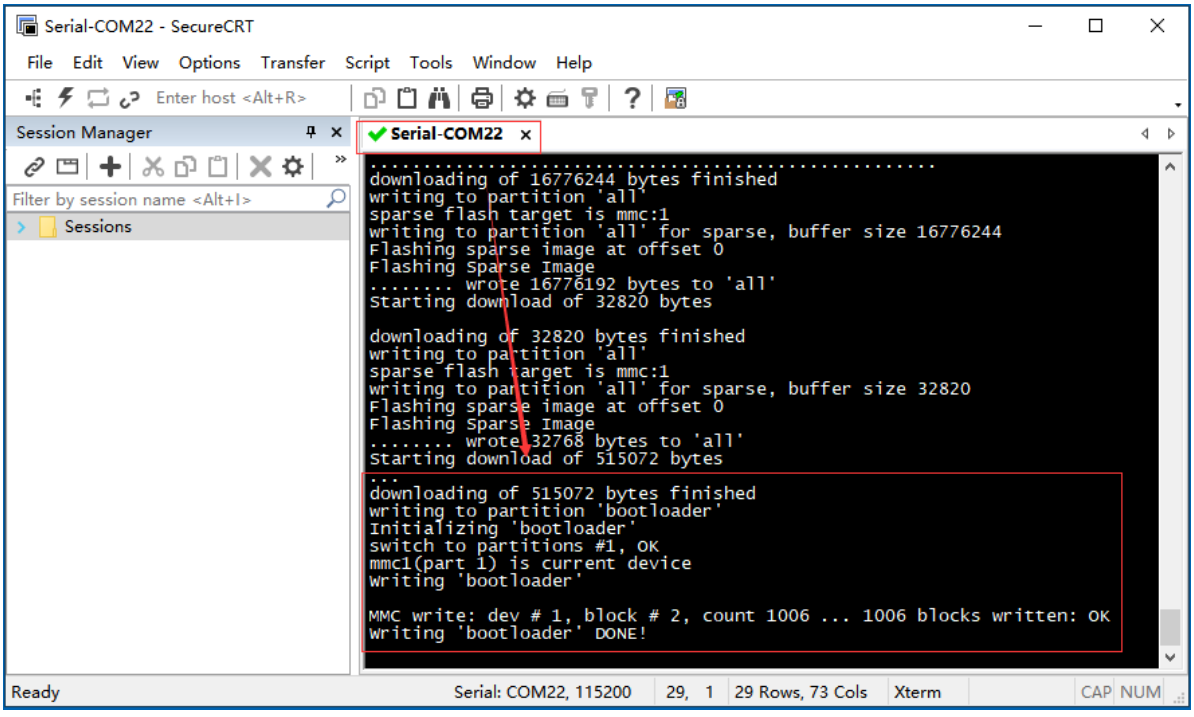
3.3.5 系统烧录

上述准备工作都准备好之后，直接双击运行***win_flash.bat*** 批处理脚本将会进入系统烧录。接下来会出现一个选项：

- 如果想只烧录/更新 **u-boot** 程序，则选择输入 **1**；
- 如果想烧录/更新 **整个系统镜像**， 则选择输入 **2**；

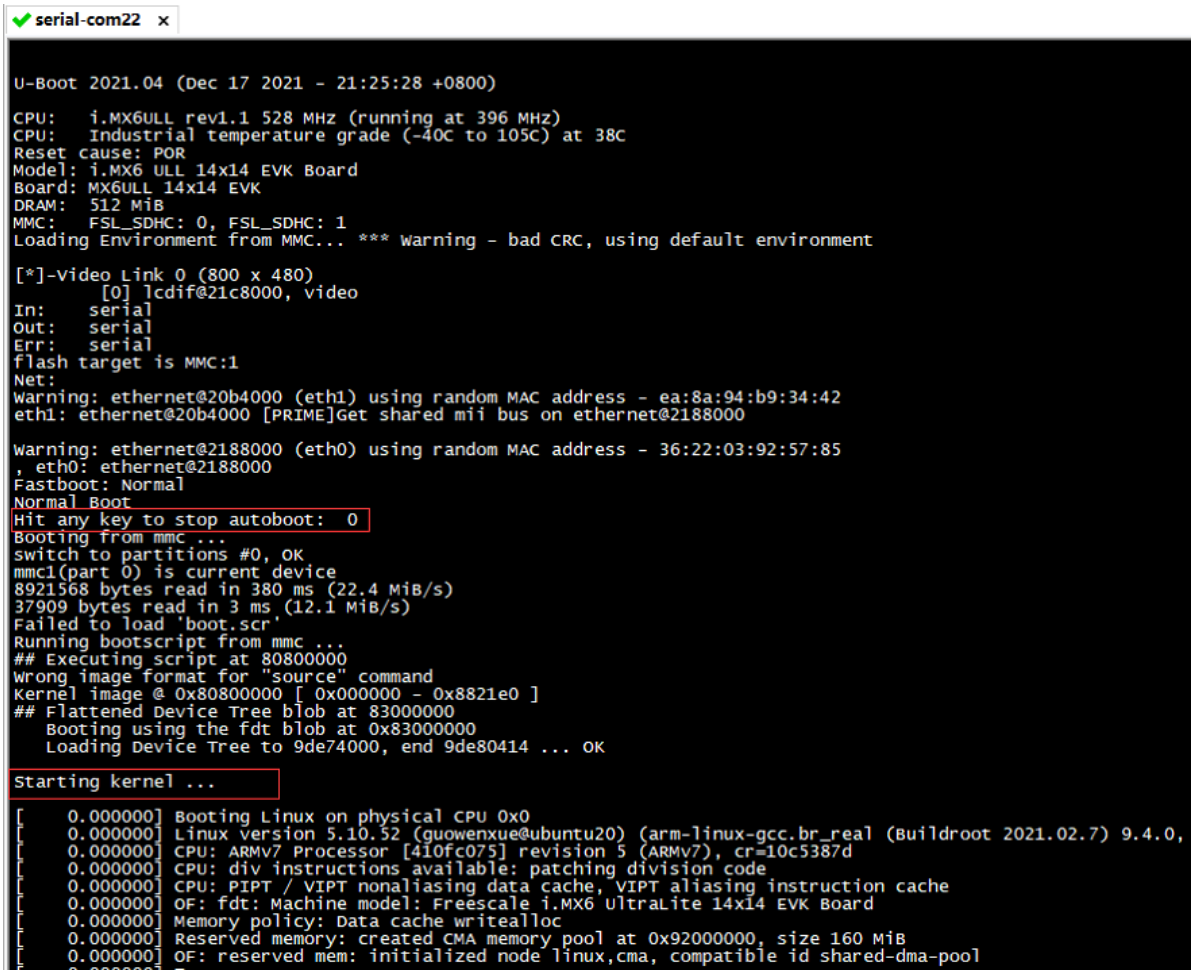


烧录成功之后，批处理将会自动退出，调试串口上也将会提示相关信息。此后可以给开发板重新上电，设备将正常启动。

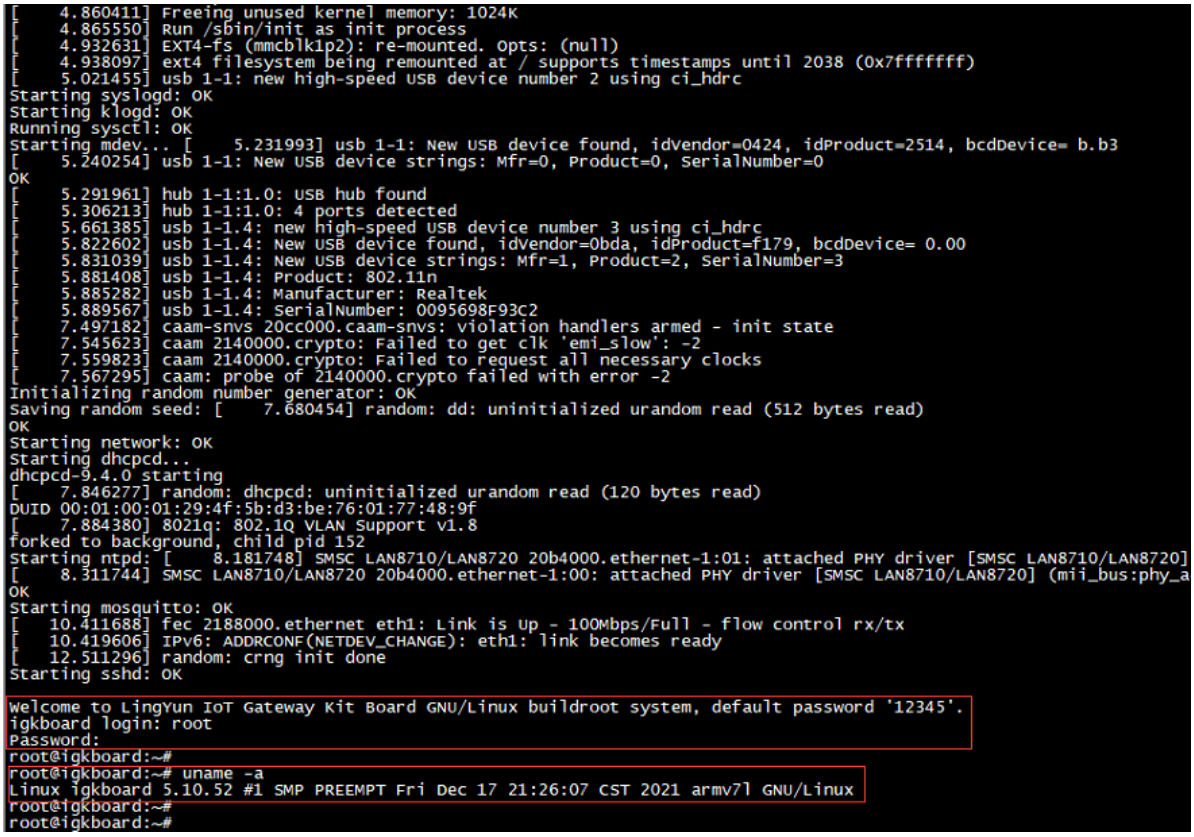


3.3.6 系统启动

系统重新上电后，串口调试终端上将会输出U-boot的启动信息，3秒内按任意键将会进入到 **U-boot** 调试模式，否则将会自动启动系统。



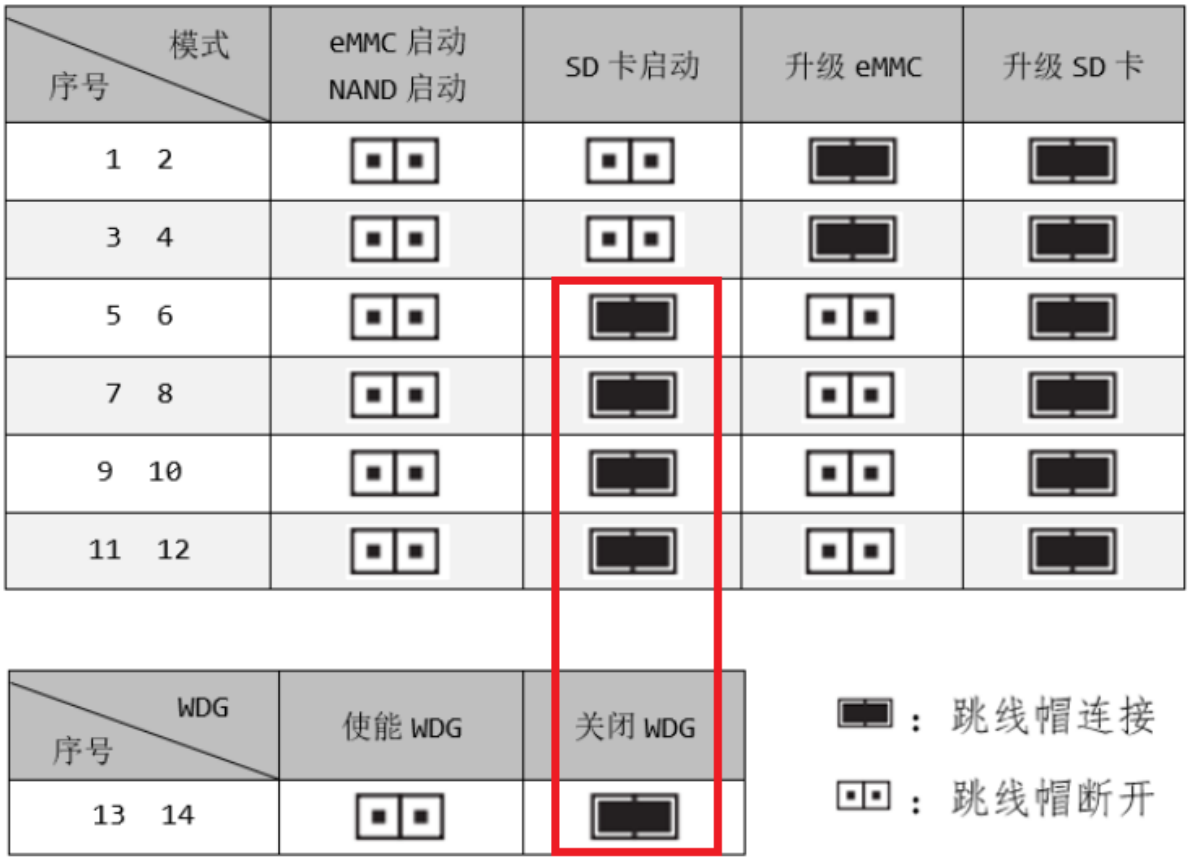
Linux系统启动后，我们可以使用用户名 **root** ，默认密码 **12345** 登录系统。



3.4 外置TF卡系统烧录

3.3.1 跳线设置

如下图所示，将 5/6、7/8、9/10、11/12 、13/14(看门狗跳线) 5个跳线全部短接，则进入外置TF卡启动模式，该模式下开发板上电后将从TF卡读取运行u-boot，进而加载并启动Linux系统。



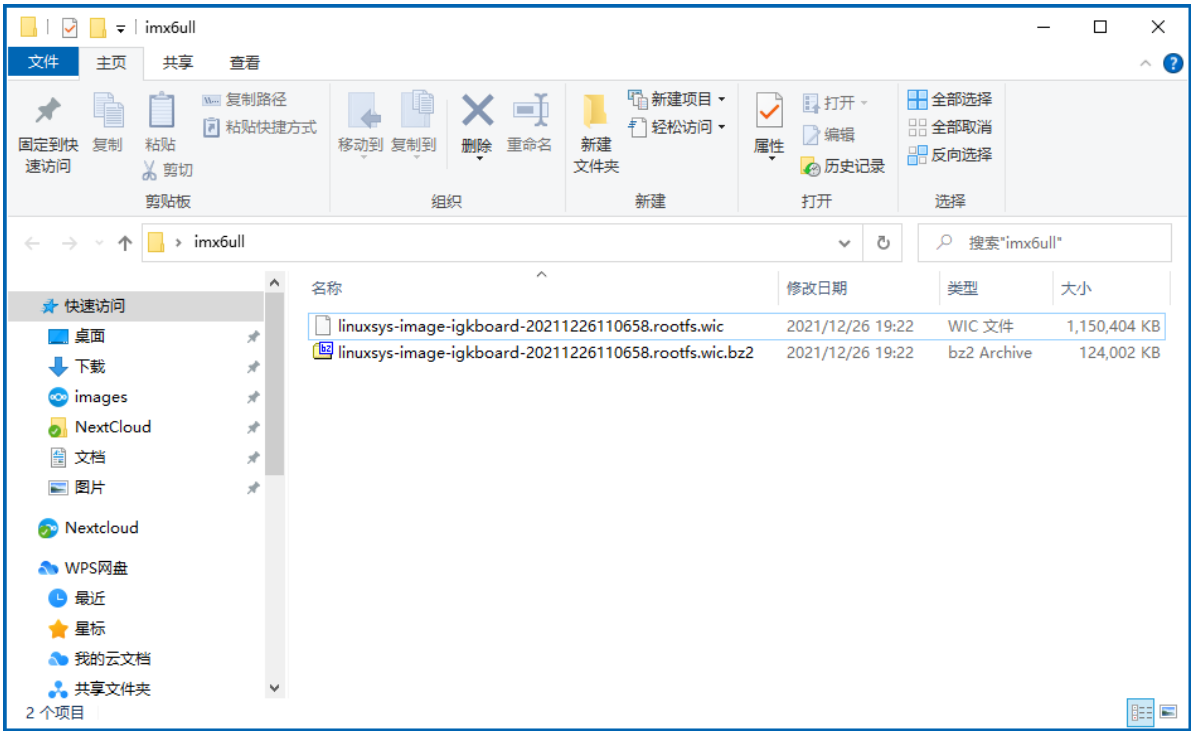
3.3.2 软件准备

在硬件设备准备好之后，我们还需要安装或准备好如下软件：

- 点击此链接下载 并安装好 ***USB*转串口调试器** 驱动；
- 点击此链接下载 并安装好 ***TF*卡系统镜像烧录工具软件 *win32diskimager*** ；

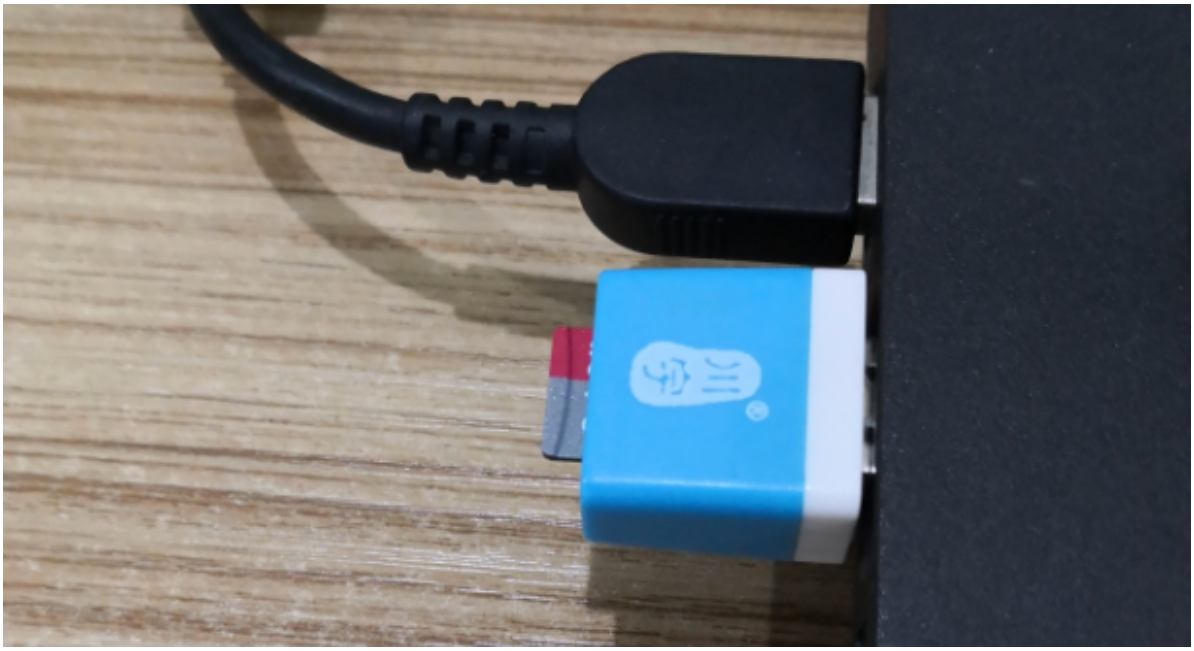
3.3.3 烧录文件

如前面 **Yocto系统编译** 相关章节所述，将前面源码编译输出的相关文件 拷贝/下载 到Windows系统下，并将系统镜像文件 解压缩出来。这里我们以 ***Yocto* 系统编译出来的镜像** 为例，详细讲解 ***TF*卡的系统烧录过程**，其它的镜像也都可以使用同样的方式来烧录。

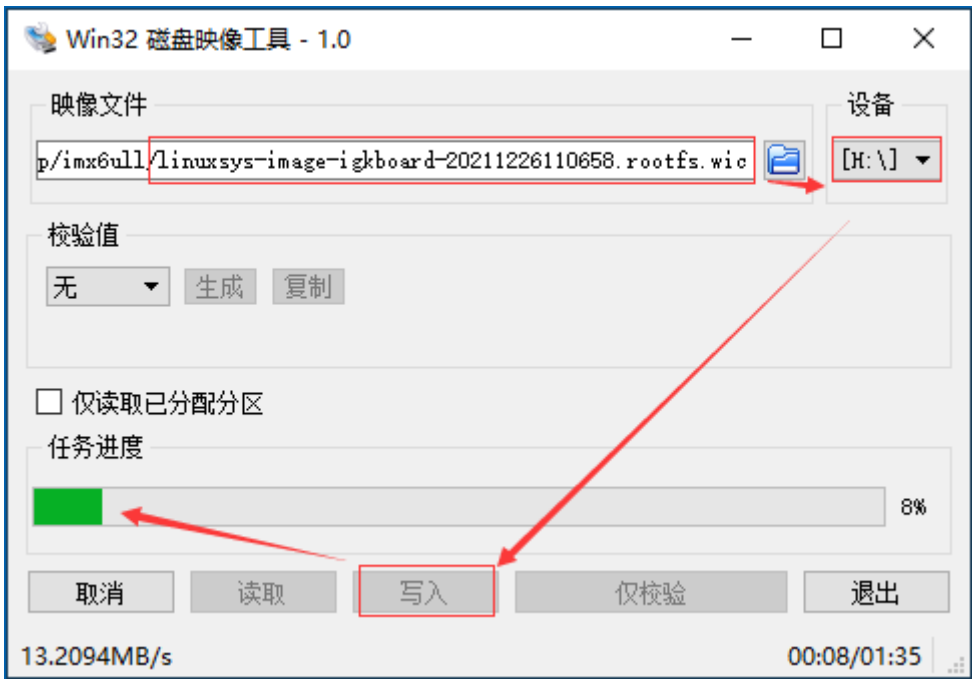


3.3.4 镜像烧录

在Windows系统下，我们可以使用 ***USB接口的TF读卡器*** 配合 ***win32diskimager软件*** 来烧录前面编译制作好的系统镜像。



运行 ***Win32DiskImager*** 系统镜像烧录程序，选中相应的系统镜像文件和TF卡对应的分区，然后点击 **写入** 开始烧写系统镜像。



3.3.5 系统启动

TF卡上系统烧录成功之后，将TF卡插入到 IGKBoard 开发板上的TF卡槽上，设置好开发板的跳线帽子，设置从TF卡启动。开发板上电后，串口调试终端上将会输出U-boot的启动信息，3秒内按任意键将会进入到 **U-boot** 调试模式，否则将会自动启动Linux系统。Linux系统启动后，我们可以使用用户名 **root**，默认密码 **12345** 登录系统。

```
[  OK  ] Reached target Hardware activated USB gadget.
[  OK  ] Started Network Time Synchronization.
[  OK  ] Reached target System Initialization.
[  OK  ] Started Daily Cleanup of Temporary Directories.
[  OK  ] Reached target System Time Set.
[  OK  ] Reached target System Time Synchronized.
[  OK  ] Started Daily rotation of log files.
[  OK  ] Reached target Timers.
[  OK  ] Listening on Avahi mDNS/DNS-SD Stack Activation Socket.
[  OK  ] Listening on D-Bus System Message Bus Socket.
         Starting sshd.socket.
         Starting Weston socket.
[  OK  ] Listening on sshd.socket.
[  OK  ] Listening on Weston socket.
[  OK  ] Created slice system-systemd\x2dfsck.slice.
[  OK  ] Reached target Sockets.
[  OK  ] Reached target Basic System.
[  OK  ] Started Job spooling tools.
[  OK  ] Started Periodic Command Scheduler.
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started Linux Firmware Loader Daemon.
         Starting IPv6 Packet Filtering Framework...
[   23.355762] imx-sdma 20ec000.sdma: firmware found.
         Starting IPv4 Packet Filtering Framework...
[   23.410514] imx-sdma 20ec000.sdma: loaded firmware 3.6
         Starting Telephony service...
[  OK  ] Started System Logging Service.
         Starting File System Check on /dev/mmcblk0p1...
         Starting User Login Management...
         Starting OpenSSH Key Generation...
[  OK  ] Finished IPv6 Packet Filtering Framework.
[  OK  ] Finished IPv4 Packet Filtering Framework.
[  OK  ] Reached target Network (Pre).
         Starting Network Service...
[  OK  ] Started Telephony service.
[  OK  ] Finished File System Check on /dev/mmcblk0p1.
         Mounting /run/media/mmcblk0p1...
[  OK  ] Mounted /run/media/mmcblk0p1.
[  OK  ] Started User Login Management.
[  OK  ] Started Network Service.
         Starting Network Name Resolution...
[  OK  ] Started Network Name Resolution.
[  OK  ] Reached target Network.
[  OK  ] Reached target Host and Network Name Lookups.
         Starting Avahi mDNS/DNS-SD Stack...
[  OK  ] Started NFS status monitor for NFSv2/3 locking..
[  OK  ] Started Respond to IPv6 Node Information Queries.
[  OK  ] Started Network Router Discovery Daemon.
         Starting Permit User Sessions...
[  OK  ] Started Avahi mDNS/DNS-SD Stack.
[  OK  ] Finished Permit User Sessions.
[  OK  ] Started Getty on tty1.
[  OK  ] Started Serial Getty on ttymxc0.
[  OK  ] Reached target Login Prompts.
[  OK  ] Reached target Multi-User System.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Finished Update UTMP about System Runlevel Changes.

Welcome to LingYun IoT Gateway Kit Board GNU/Linux Yocto System, Default Password '12345'.
igkboard login: root
Password:
root@igkboard:~#
root@igkboard:~# uname -a
Linux igkboard 5.10.52-lts-5.10.y+gc42346de2df4 #1 SMP PREEMPT Wed Sep 8 10:41:11 UTC 2021 armv7l armv7l armv7l GNU/Linux
root@igkboard:~#
```