PET-A64-P01 安卓主板/开发板
开发手册

一、编译环境搭建指南

- 安装 Ubuntu 16.04 64 位。
- 安装依赖软件

```
sudo apt clean
sudo apt update
sudo apt -y upgrade
sudo apt -y dist-upgrade
sudo apt -y install openssh-server
sudo apt -y install git flex bison gperf build-essential libncurses5-dev:i386
sudo apt -y install libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-dev g++-multilib
sudo apt -y install tofrodos python-markdown libxml2-utils xsltproc zlib1g-dev:i386
sudo apt -y install dpkg-dev libsdl1.2-dev libesd0-dev
sudo apt -y install git-core gnupg flex bison gperf build-essential
sudo apt -y install zip curl zlib1g-dev gcc-multilib g++-multilib
sudo apt -y install libc6-dev-i386
sudo apt -y install lib32ncurses5-dev x11proto-core-dev libx11-dev
sudo apt -y install lib32z-dev ccache
sudo apt -y install libgl1-mesa-dev libxml2-utils xsltproc unzip m4
sudo apt -y install gawk fakeroot g++-multilib gcc-multilib
sudo apt -y install u-boot-tools make texinfo clang cmake dos2unix unix2dos
sudo apt -y install libssl-dev
```

- 安装 openjdk-8-jdk。

```
sudo apt -y install openjdk-8-jdk
```

- 输入命令 java -version 检查 java 的主版本号是否为 1.8。
- 在开发工具目录下有安装好的虚拟机磁盘镜像文件（VMware 15.5.6 及以上版本），虚拟机内存设置最少需要 16G，磁盘镜像文件所在的 windows 磁盘分区剩余容量大于 200G。
- 编译安卓如果遇到内存不足错误可以加大内存容量或减少编译线程数量。
- 如果使用虚拟机内存容量不足，编译安卓系统，默认参数可能会因内存不足引起错误，可以尝试将虚拟机内存设置为 8G，进行以下修改后再重启虚拟机进行编译。

```
文件 prebuilts\sdk\tools\jack-admin
JACK_SERVER_COMMAND="java -XX:MaxJavaStackTraceDepth=-1 -Djava.io.tmpdir=$TMPDIR
$JACK_SERVER_VM_ARGUMENTS -cp $LAUNCHER_JAR $LAUNCHER_NAME"
修改为
JACK_SERVER_COMMAND="java -Xmx8G -XX:MaxJavaStackTraceDepth=-1 -Djava.io.tmpdir=$TMPDIR
$JACK_SERVER_VM_ARGUMENTS -cp $LAUNCHER_JAR $LAUNCHER_NAME"
重启虚拟机，再运行编译命令。
```

二、解压源代码

将源代码压缩文件全部复制到 Ubuntu 系统下，保证所在磁盘剩余空间要大于 100G，使用以下命令解压源代码：

```
tar xvJf PET_A64_P01_Source.tar.xz
```

三、编译安卓 Android

首次编译请严格按照步骤进行内核、uboot、android 的编译，否则编译可能会出现错误。

1、编译内核

```
cd lichee
./build.sh -p sun50iw1p1_android -k linux-3.10
```

编译完成后正确提示如下：

```
make: Leaving directory `/root/Work/A64_7.1.1_SKD_BASE/lichee/linux-3.10/modules/gpu'
[GPU]: mali400 device driver has been built.
regenerate rootfs cpio
15756 blocks
15756 blocks
build_ramfs
Copy boot.img to output directory ...
Copy modules to target ...

sun50iw1p1 compile Kernel successful


INFO: build kernel OK.
INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.
INFO: ----------------------------------------
INFO: build lichee OK.
INFO: ----------------------------------------
root@SeKeDe:~/Work/A64_7.1.1_SKD_BASE/lichee#
```

2、编译 uboot

首次编译或修改 uboot 代码后需要执行这一步骤。

```
cd lichee/brandy
./build.sh -p sun50iw1p1
编译完成后正确提示如下
```

```
/root/Work/A64_7.1.1_SKD_BASE/lichee/brandy/u-boot-2014.07/../gcc-linaro
opy  -O binary /root/Work/A64_7.1.1_SKD_BASE/lichee/brandy/u-boot-2014.0
14_sunxi_spl/sunxi_spl/sbrom/sboot.axf /root/Work/A64_7.1.1_SKD_BASE/lic
./../bootloader/uboot_2014_sunxi_spl/sunxi_spl/sbrom/sboot.bin
/root/Work/A64_7.1.1_SKD_BASE/lichee/brandy/u-boot-2014.07/../../bootloa
nxi_spl/../../tools/add_hash.sh -f /root/work/A64_7.1.1_SKD_BASE/lichee/
/bootloader/uboot_2014_sunxi_spl/sunxi_spl/sbrom/sboot.bin -m sboot
build_sboot
'sboot_sun50iw1p1.bin' -> '/root/Work/A64_7.1.1_SKD_BASE/lichee/brandy/u-
ack/chips/sun50iw1p1/bin/sboot_sun50iw1p1.bin'
root@SeKeDe:~/Work/A64_7.1.1_SKD_BASE/lichee/brandy#
```

3、编译 android

```
cd android
source build/envsetup.sh
lunch tulip_p3-eng
extract-bsp
```

make -j4

pack

编译完成后正确提示如下

```
config.fex Len: 0xc400
split_xxxx.fex Len: 0x200
sys_partition.fex Len: 0x12c8
sunxi.fex Len: 0x14000
boot0_nand.fex Len: 0x8000
boot0_sdcard.fex Len: 0x8000
u-boot.fex Len: 0xd4000
toc1.fex Len: 0x8
toc0.fex Len: 0x8
fes1.fex Len: 0x4100
boot_package.fex Len: 0x130000
usbtool.fex Len: 0x23600
aultools.fex Len: 0x2847b
aultls32.fex Len: 0x24d23
cardtool.fex Len: 0x41600
cardscript.fex Len: 0x6de
sunxi_mbr.fex Len: 0x10000
dlinfo.fex Len: 0x4000
arisc.fex Len: 0x6
boot-resource.fex Len: 0x4ef400
Vboot-resource.fex Len: 0x4
env.fex Len: 0x20000
Venv.fex Len: 0x4
boot.fex Len: 0xf30800
Vboot.fex Len: 0x4
system.fex Len: 0x2e768144
Vsystem.fex Len: 0x4
recovery.fex Len: 0x110c800
Vrecovery.fex Len: 0x4
diskfs.fex Len: 0x200
Vdiskfs.fex Len: 0x4
BuildImg 0
Dragon execute image.cfg SUCCESS !
----------image is at----------

/root/Work/A64_7.1.1_SKD_BASE/lichee/tools/pack/sun50iw1p1_android_p3_uart0.img

pack finish
```

编译完成后会在 lichee/tools/pack 目录下生成 sun50iw1p1_android_p3_uart0.img 系统烧写镜像文件。

四、GPIO 编程参考

通过 sysfs 方式控制 GPIO，GPIO 的操作接口包括 direction 和 value 等，direction 控制 GPIO 输入和输入模式，而 value 可控制 GPIO 输出或获得 GPIO 输入。

例如控制调试灯 GPIO 操作如下（串口终端命令行方式）：

调试灯 GPIO 设置为输出　　echo out > /sys/class/gpio/gpio235/direction

调试灯 GPIO 输出高电平　　echo 1 > /sys/class/gpio/gpio235/value

调试灯 GPIO 输出高低平　　echo 0 > /sys/class/gpio/gpio235/value

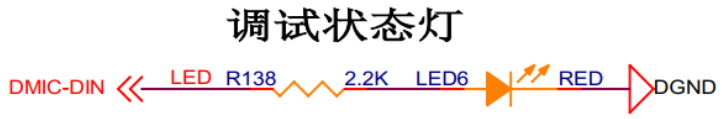
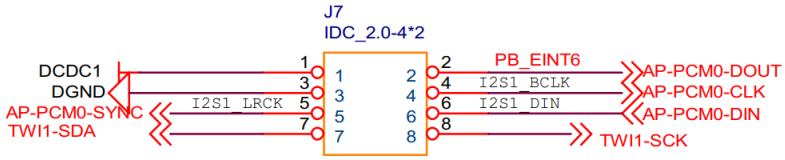
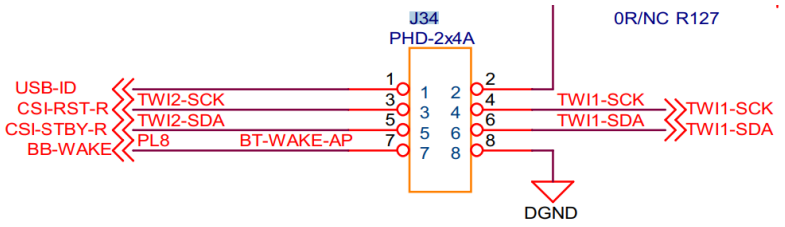
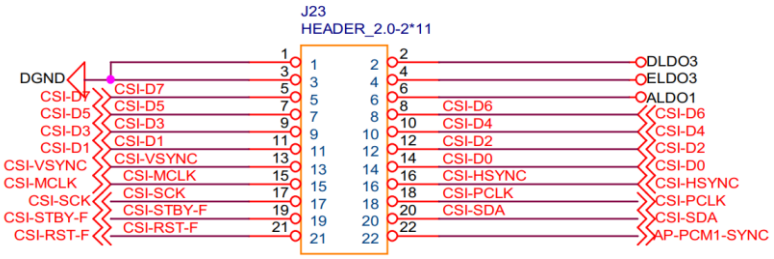
调试灯 GPIO 设置为输入　　echo in > /sys/class/gpio/gpio235/direction

读取调试灯 GPIO 输出输入电平　cat /sys/class/gpio/gpio235/value

当 GPIO 处于输出和输入模式时都可以读取，当设置为输入模式时读取的是 GPIO 实际电平，当设置为输出模式时读取的是设置的值（如果设置为高电平输出，外部将引脚电平拉低后，读取的值依然是 1）。

应用程序控制请参考源码下的 demo 程序源码

GPIO 对应控制目录列表			
丝印	接口	脚位	目录

LED6			/sys/class/gpio/gpio235
J7		2 脚	/sys/class/gpio/gpio38
		4 脚	/sys/class/gpio/gpio37
		5 脚	/sys/class/gpio/gpio36
		6 脚	/sys/class/gpio/gpio39
J34		1 脚	/sys/class/gpio/gpio233
		3 脚	/sys/class/gpio/gpio142
		5 脚	/sys/class/gpio/gpio143
		7 脚	/sys/class/gpio/gpio360
J23		5 脚	/sys/class/gpio/gpio139
		7 脚	/sys/class/gpio/gpio137
		8 脚	/sys/class/gpio/gpio138
		9 脚	/sys/class/gpio/gpio135
		10 脚	/sys/class/gpio/gpio136
		11 脚	/sys/class/gpio/gpio133
		12 脚	/sys/class/gpio/gpio134
		13 脚	/sys/class/gpio/gpio131
		14 脚	/sys/class/gpio/gpio132
		15 脚	/sys/class/gpio/gpio129
		16 脚	/sys/class/gpio/gpio130
		17 脚	/sys/class/gpio/gpio140
		18 脚	/sys/class/gpio/gpio128
		19 脚	/sys/class/gpio/gpio145
		20 脚	/sys/class/gpio/gpio141
		21 脚	/sys/class/gpio/gpio144
		22 脚	/sys/class/gpio/gpio202

五、WatchDog 看门狗编程参考

进入内核后默认会启动看门狗，内核崩溃等情况出现，会在 60 秒内自动复位主板。
上层应用程序打开看门狗后，内核将看门狗控制权交由上层应用程序控制，上层应用程序的喂狗间隔建议不少于 10 秒。
看门狗的使用流程为 打开看门狗→循环喂狗→停止喂狗→关闭看门狗
喂狗之前必须先打开看门狗，关闭看门狗之前需停止喂狗操作。
打开看门狗后如果 60 秒内没有喂狗或关闭看门狗，系统会自动复位。

命令行测试：

打开看门狗：echo 1 > /sys/class/gzpet/user/watch_dog
喂狗：　　echo 2 > /sys/class/gzpet/user/watch_dog
关闭看门狗：echo 0 > /sys/class/gzpet/user/watch_dog
应用程序控制请参考源码下的 demo 程序源码

六、串口 UART 编程参考

J8	串口/dev/ttyS1	PH2.0 4Pin	标配	默认为 RS232 串口，与蓝牙功能不可同时使用
J9	串口/dev/ttyS3	PH2.0 4Pin	标配	默认为 RS232 串口，可配置为 TTL 串口
J10	串口/dev/ttyS5	PH2.0 4Pin	标配	默认为 TTL 串口(只能接收数据)，与 GPS 功能不可同时使用
J13	串口/dev/ttyS5	PH2.0 5Pin	非标配	与 J10 复用且位置重叠，不能同时使用
J11	串口/dev/ttyS0	XH2.54 4Pin	标配	默认为 TTL 串口，调试串口
J49	串口/dev/ttyS2	PH2.0 4Pin	标配	默认为 TTL 串口，与 485 功能不可同时使用
J14	485 通讯 /dev/ttyS2	PH2.0 4Pin	标配	默认为 TTL 串口

调试串口修改为普通串口使用请参考安卓主板系统说明书进行源码修改与编译系统镜像文件。
安卓系统串口编程请参考以下链接：
https://github.com/Geek8ug/Android-SerialPort

七、动态隐藏/显示系统状态栏和导航栏

注意：仅在未将系统设置为强制全屏时有效。
隐藏状态栏和导航栏在应用 app 里面向系统发送广播
gzpeite.intent.systemui.hidenavigation 和 gzpeite.intent.systemui.hidestatusbar
显示状态栏和导航栏在应用 app 里面向系统发送广播
gzpeite.intent.systemui.shownavigation 和 gzpeite.intent.systemui.showstatusbar
测试命令如下：

```
am broadcast -a "gzpeite.intent.systemui.hidenavigation"
am broadcast -a "gzpeite.intent.systemui.hidestatusbar"

am broadcast -a "gzpeite.intent.systemui.shownavigation"
am broadcast -a "gzpeite.intent.systemui.showstatusbar"
```

八、静默安装/卸载应用

安装 APK 时，向系统发送 gzpeite.intent.action.install_apk 广播
卸载 APK 时，向系统发送 gzpeite.intent.action.uninstall_apk 广播
测试命令如下：

```
am broadcast -a "gzpeite.intent.action.install_apk" --es apk_path "/mnt/media_rw/0000-4823/GPSTest.apk"
am broadcast -a "gzpeite.intent.action.uninstall_apk" --es pkg_name "com.android.gpstest"
```

九、编译 Linux + QT5.9

注意 Linux+QT 环境下不支持 opengl，不支持视频硬件编解码，如果需要进行视频播放或复杂的图形显示等应用，建议用安卓系统或 QT for Android https://doc.qt.io/qt-5/android.html
请首先新开一个控制台进行编译操作。
首次编译请严格按照步骤进行内核、uboot、Rootfs 的编译，否则编译可能会出现错误。

1、编译内核

cd lichee
./build.sh -p sun50iw1p1_dragonboard -k linux-3.10
编译完成后正确提示如下

```
Copy boot.img to output directory ...
./scripts/build.sh

sun50iw1p1 compile Kernel successful


INFO: build kernel OK.
INFO: build rootfs ...
Regenerating dragonboard Rootfs...
extract dragonboard rootfs.tar.xz
INFO: Prepare dragonboard toolchain ...
cp: cannot stat 'extra/*': No such file or directory
generating rootfs...
blocks: 695M -> 1024M
Creating filesystem with parameters:
    Size: 1073741824
    Block size: 4096
    Blocks per group: 32768
    Inodes per group: 8192
    Inode size: 256
    Journal blocks: 4096
    Label:
    Blocks: 262144
    Block groups: 8
    Reserved block group size: 63
Created filesystem with 14921/65536 inodes and 186636/262144 blocks
e2fsck 1.44.1 (24-Mar-2018)
fsck.ext4: Bad magic number in super-block while trying to open rootfs.ext4
success in generating rootfs
Build at: Sun Apr  7 20:29:02 CST 2019
INFO: build rootfs OK.
INFO: ----------------------------------------
INFO: build lichee OK.
INFO: ----------------------------------------
```

2、编译 uboot

首次编译或修改 uboot 代码后需要执行这一步骤。

首先切换到 uboot 目录
cd lichee/brandy
./build.sh -p sun50iw1p1
编译完成后正确提示如下

```
/root/Work/A64_7.1.1_SKD_BASE/lichee/brandy/u-boot-2014.07/../gcc-linaro
opy  -O binary /root/Work/A64_7.1.1_SKD_BASE/lichee/brandy/u-boot-2014.0
14_sunxi_spl/sunxi_spl/sbrom/sboot.axf /root/Work/A64_7.1.1_SKD_BASE/lic
./../bootloader/uboot_2014_sunxi_spl/sunxi_spl/sbrom/sboot.bin
/root/Work/A64_7.1.1_SKD_BASE/lichee/brandy/u-boot-2014.07/../../bootloa
nxi_spl/../../tools/add_hash.sh -f /root/Work/A64_7.1.1_SKD_BASE/lichee/
/bootloader/uboot_2014_sunxi_spl/sunxi_spl/sbrom/sboot.bin -m sboot
build_sboot
'sboot_sun50iw1p1.bin' -> '/root/Work/A64_7.1.1_SKD_BASE/lichee/brandy/u-
ack/chips/sun50iw1p1/bin/sboot_sun50iw1p1.bin'
root@SeKeDe:~/Work/A64_7.1.1_SKD_BASE/lichee/brandy#
```

3、编译 QT Rootfs

首先切换到 pack 目录

cd lichee/tools/pack/

./pack -c sun50iw1p1 -p dragonboard -b p3

编译完成后正确提示如下

```
config.fex Len: 0xc400
split_xxxx.fex Len: 0x200
sys_partition.fex Len: 0xaa0
sunxi.fex Len: 0x14000
boot0_nand.fex Len: 0x8000
boot0_sdcard.fex Len: 0x8000
u-boot.fex Len: 0xd4000
toc1.fex Len: 0x8
toc0.fex Len: 0x8
fes1.fex Len: 0x4100
boot_package.fex Len: 0x130000
usbtool.fex Len: 0x23600
aultools.fex Len: 0x2847b
aultls32.fex Len: 0x24d23
cardtool.fex Len: 0x41600
cardscript.fex Len: 0x6de
sunxi_mbr.fex Len: 0x10000
dlinfo.fex Len: 0x4000
arisc.fex Len: 0x6
boot-resource.fex Len: 0x4f4c00
Vboot-resource.fex Len: 0x4
env.fex Len: 0x20000
Venv.fex Len: 0x4
boot.fex Len: 0xfa7800
Vboot.fex Len: 0x4
rootfs.fex Len: 0x2cc5b6dc
Vrootfs.fex Len: 0x4
BuildImg 0
Dragon execute image.cfg SUCCESS !
----------image is at----------

/root/Work/A64_7.1.1_SKD_BASE/lichee/tools/pack/sun50iw1p1_dragonboard_p3_uart0.img

pack finish
```

编译完成后会在 lichee/tools/pack 目录下生成 sun50iw1p1_dragonboard_p3_uart0.img 系统烧写镜像文件。

4、修改 Rootfs

完成首次编译后，rootfs 的所有文件位于 lichee\buildroot\target\dragonboard\rootfs 目录下。
如果需要修改或添加文件，需要将文件复制到 lichee\buildroot\target\dragonboard\extra 相同目录下，然后再修改，重新编译内核、uboot、rootfs 即可。

例如需要修改 rootfs/etc/init.d/S00peite 这个系统初始化设置脚本文件
cd lichee/buildroot/target/dragonboard
mkdir –p extra/etc/init.d
cp –rf rootfs/etc/init.d/S00peite extra/etc/init.d/S00peite
修改 extra/etc/init.d/S00peite 后重新编译即可生成新的烧写镜像文件

5、更换 Rootfs 为 Linux 或 Linux + QT

首先删除 lichee\buildroot\target\dragonboard\rootfs 目录。
将开发资料《源代码》目录下 rootfs 文件更名为 rootfs.tar.xz
复制 rootfs.tar.xz 到 lichee\buildroot\target\dragonboard 覆盖同名文件
重新编译即可
Rootfs 类型有：
Linux_Full --- Linux 全功能版，不包含 QT
Linux_Lite --- Linux 部分功能版，不包含 QT
QT_Full --- Linux+QT 全功能版
QT_Lite --- Linux+QT 部分功能版

6、交叉编译其他应用

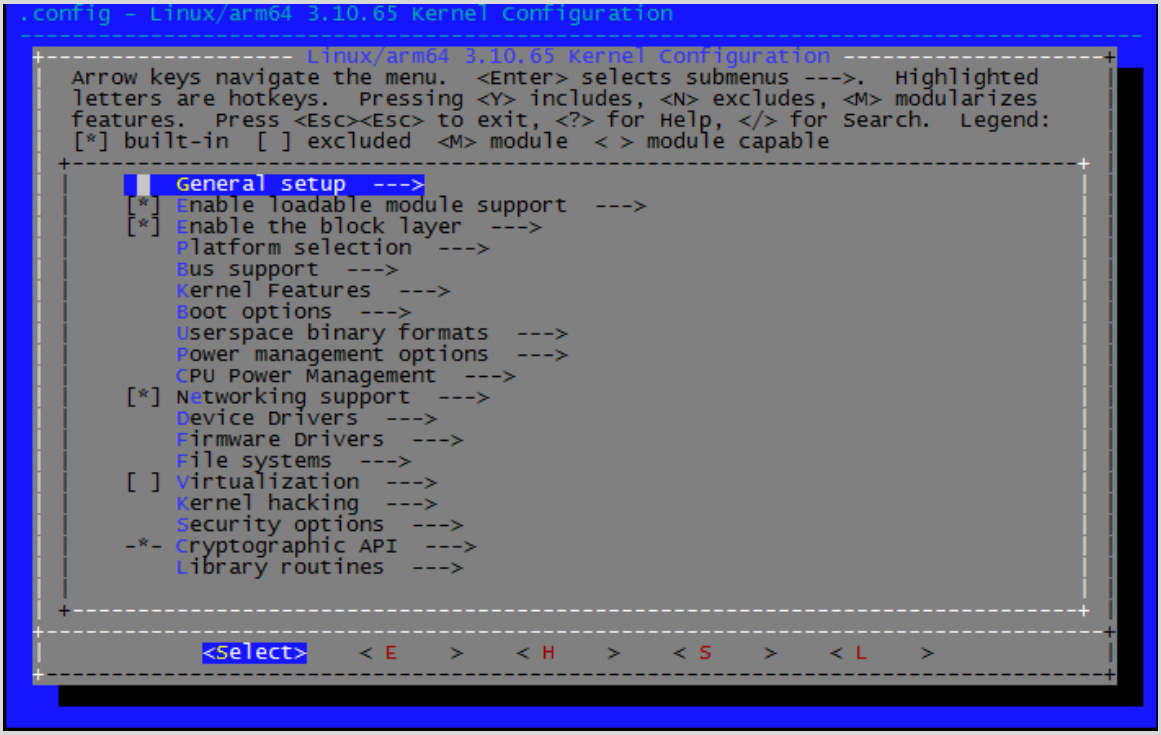
系统使用的交叉编译器位于开发资料《开发工具/交叉编译器》目录下:
gcc-linaro-7.2.1-2017.11-x86_64_aarch64-linux-gnu.tar.xz
编译所需的其他库文件是 sysroot_peite.tar.xz(交叉编译器目录内)可根据需要进行解压使用，客户可自行编译其他未包含的支持库、应用程序等。

十、修改 Linux 内核编译选项

首先切换到 linux 内核目录
cd lichee/linux-3.10/

加载默认配置
make sun50iw1p1smp_android_defconfig

启动内核配置
make menuconfig



修改内核选项时不要选择编译成模组文件，可以选择直接编译进内核。
完成配置后保存退出，
将内核根目录下的 .config 文件复制保存为
arch/arm64/configs/sun50iw1p1smp_android_defconfig
此步骤非常重要，如果不执行的话会自动恢复为默认配置。
cp .config arch/arm64/configs/sun50iw1p1smp_android_defconfig

完成内核配置修改后，从新编译 android 或 linux 即可。
注意 Android 和 Linux 共用同一个默认配置，修改内核选项对 android 和 Linux+QT 同时生效。

十一、镜像文件烧写

开发过程中，一般使用 PhoenixSuit 进行镜像文件的烧写，具体操作方式请参考开发工具目录下的《PhoenixSuit 使用说明文档.pdf》，除了 Android 系统我司的 Linux+QT 系统也支持这种烧写方式。

将开发板的 MicroUSb 接口连接到系统主机后，Linux+QT 系统检测到的设备信息如下：

烧写操作需要首先通过 Micro USB 数据线连接主机的开发板，在进行烧写时如果出现主机识别到新的设备没有正常安装驱动的情况时，需要手动安装设备驱动程序，驱动程序位于开发工具文件夹内。

注意，在点击烧写镜像后，设备会重启黑屏，如果没有开始烧写进程，此时需要在 PC 端的设备管理区中对黄色感叹号设备手动安装镜像烧写设备驱动（AW_Driver）。

十二、建立 QT 应用程序编译环境

所需工具位于开发资料的《开发工具/QT》目录下:

1、解压交叉编译器 gcc-linaro-7.2.1-2017.11-x86_64_aarch64-linux-gnu.tar.xz

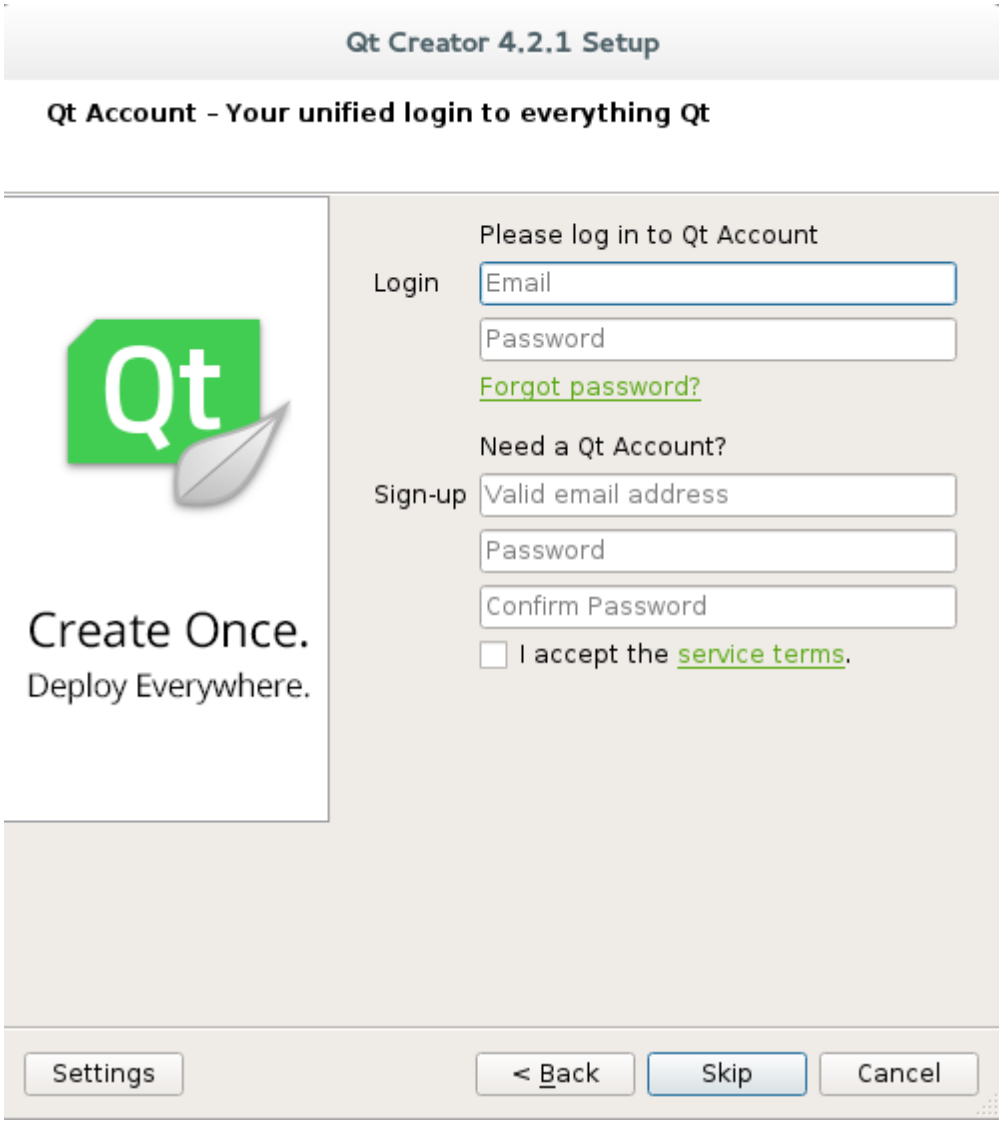
```
sudo tar -xJf gcc-linaro-7.2.1-2017.11-x86_64_aarch64-linux-gnu.tar.xz -C /usr/local
```

2、解压库文件 sysroot_peite_qt.tar.xz

```
sudo tar -xJf sysroot_peite_qt.tar.xz -C /usr/local
```

3、解压安装 qt-creator-opensource-linux-x86_64-4.4.0.tar.xz

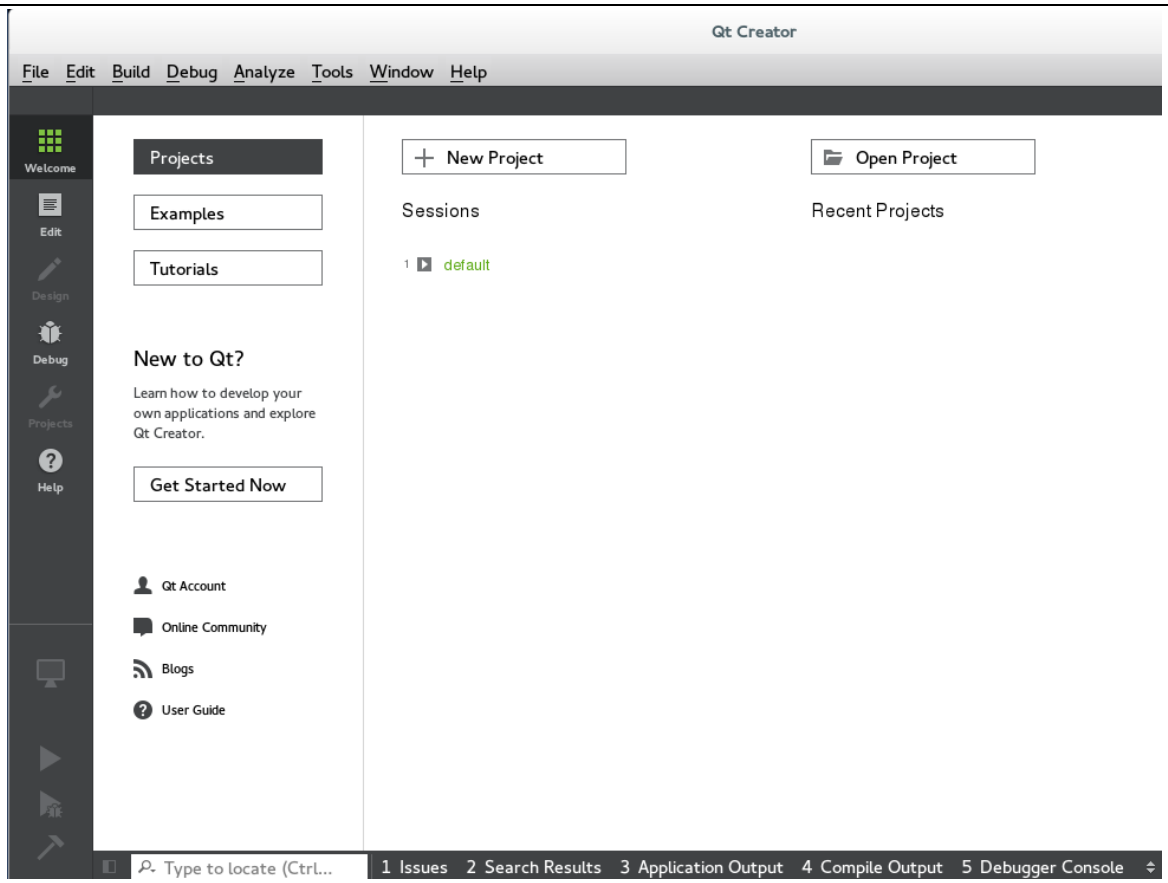
```
chmod +x qt-creator-opensource-linux-x86_64-4.4.1.run
./qt-creator-opensource-linux-x86_64-4.4.1.run
```



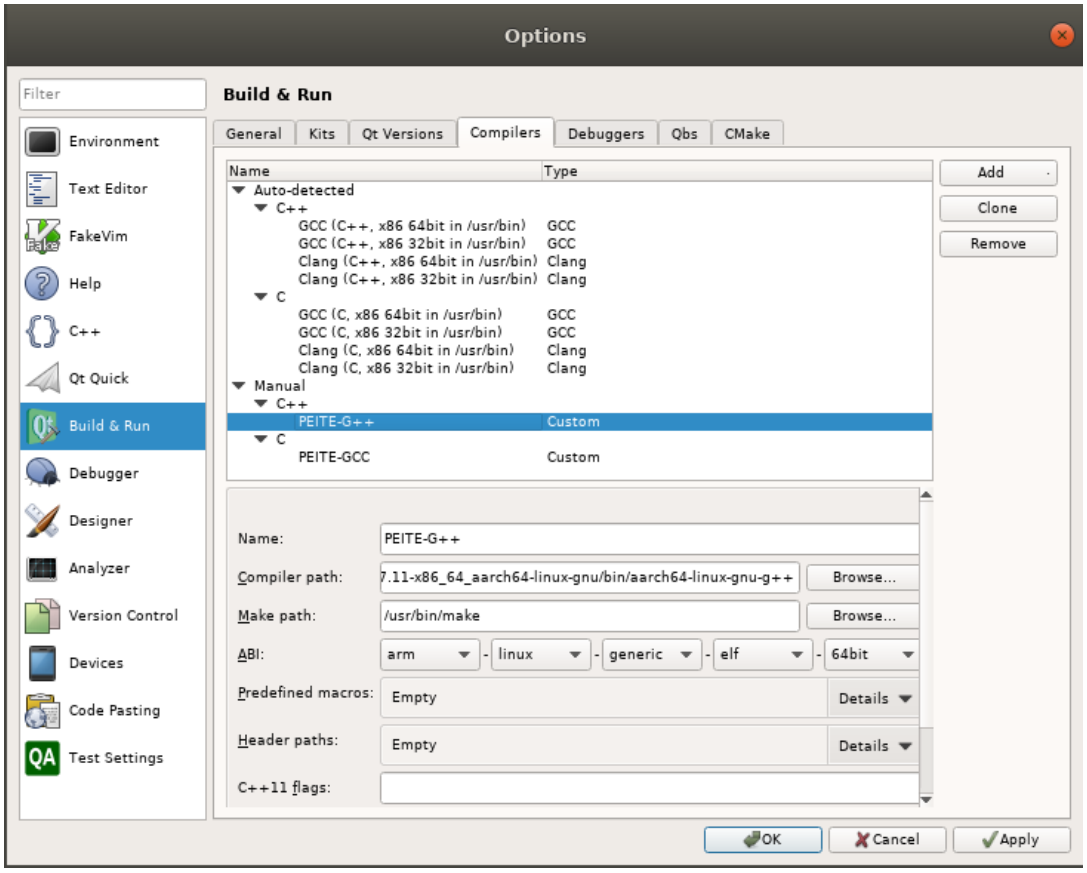
注意在上面这一步选择 Skip，其他直接选择 Next 即可

4、 启动 qt creator 设置交叉编译器和 QT 库文件路径。

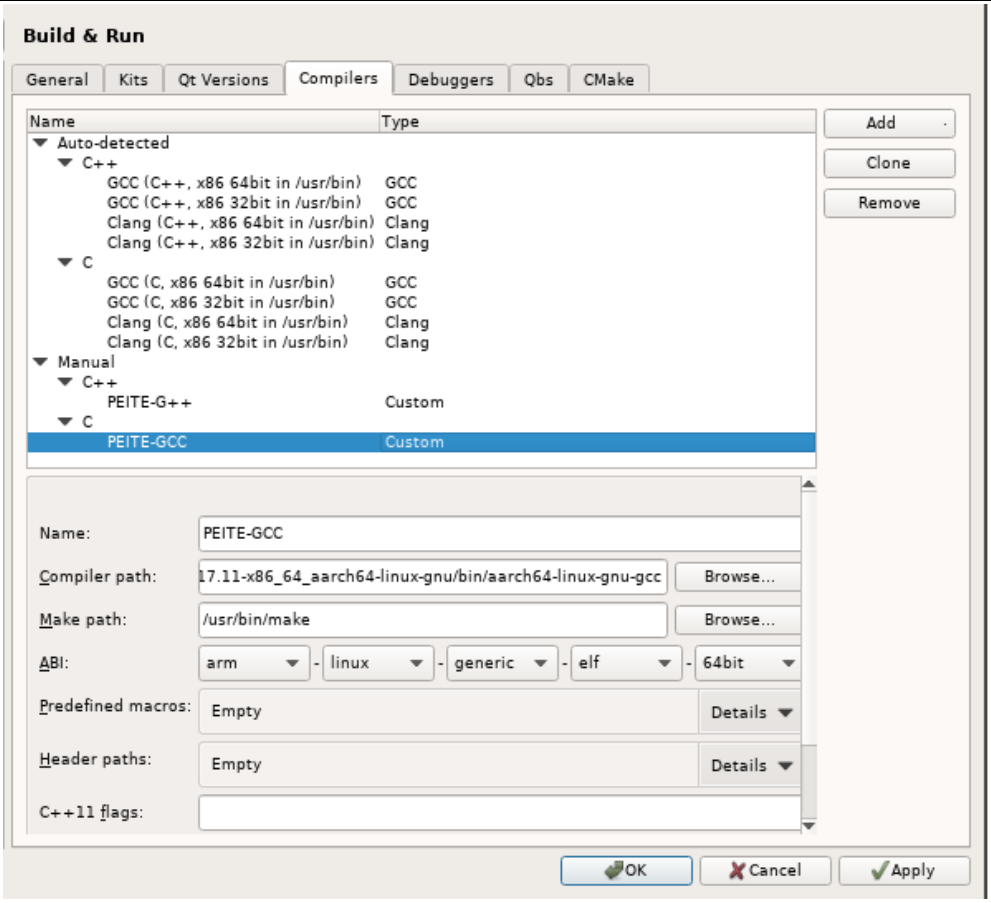
5、 /opt/qtcreator-4.4.1/bin/qtcreator



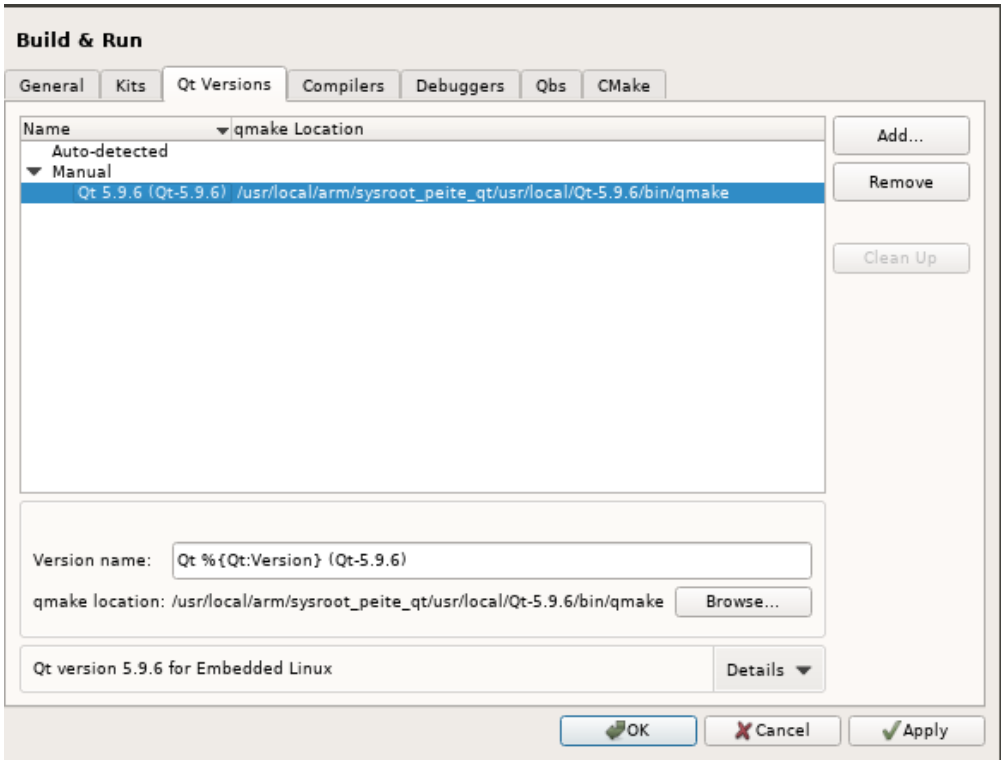
选择菜单 Tools->Options->Build & Run->Compilers，点击 Add ->Custom->C++ 按钮，添加 C++编译器，Complier path：
/usr/local/arm/gcc-linaro-7.2.1-2017.11-x86_64_aarch64-linux-gnu/bin/ aarch64-linux-gnu-g++



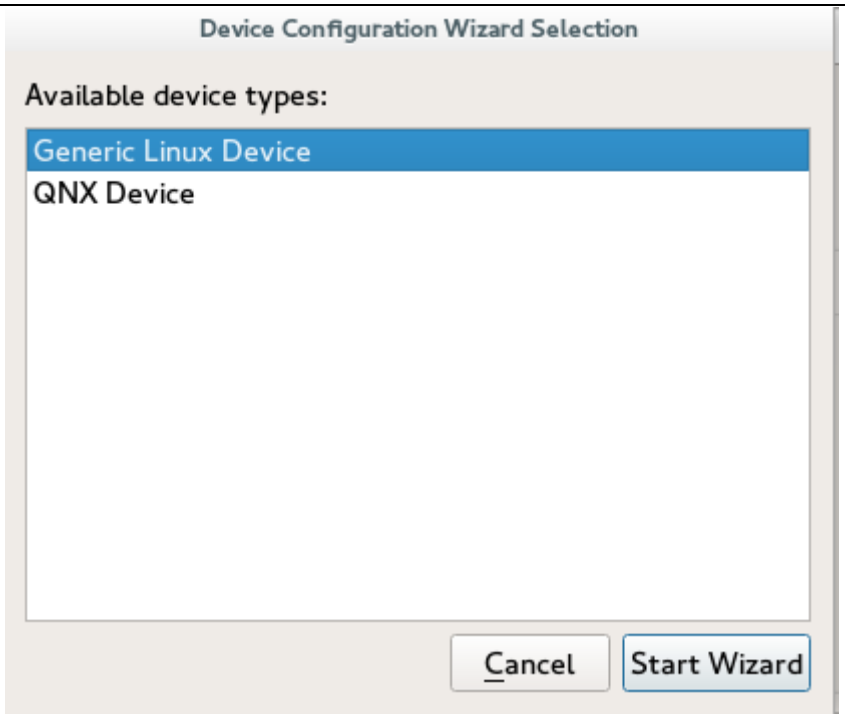
Add ->Custom->C 添加 C 编译器 Complier path：
/usr/local/arm/gcc-linaro-7.2.1-2017.11-x86_64_aarch64-linux-gnu/bin/ aarch64-linux-gnu-gcc



选择菜单 Tools->Options->Build & Run->Qt Versions，点击 Add 按钮，配置如下：

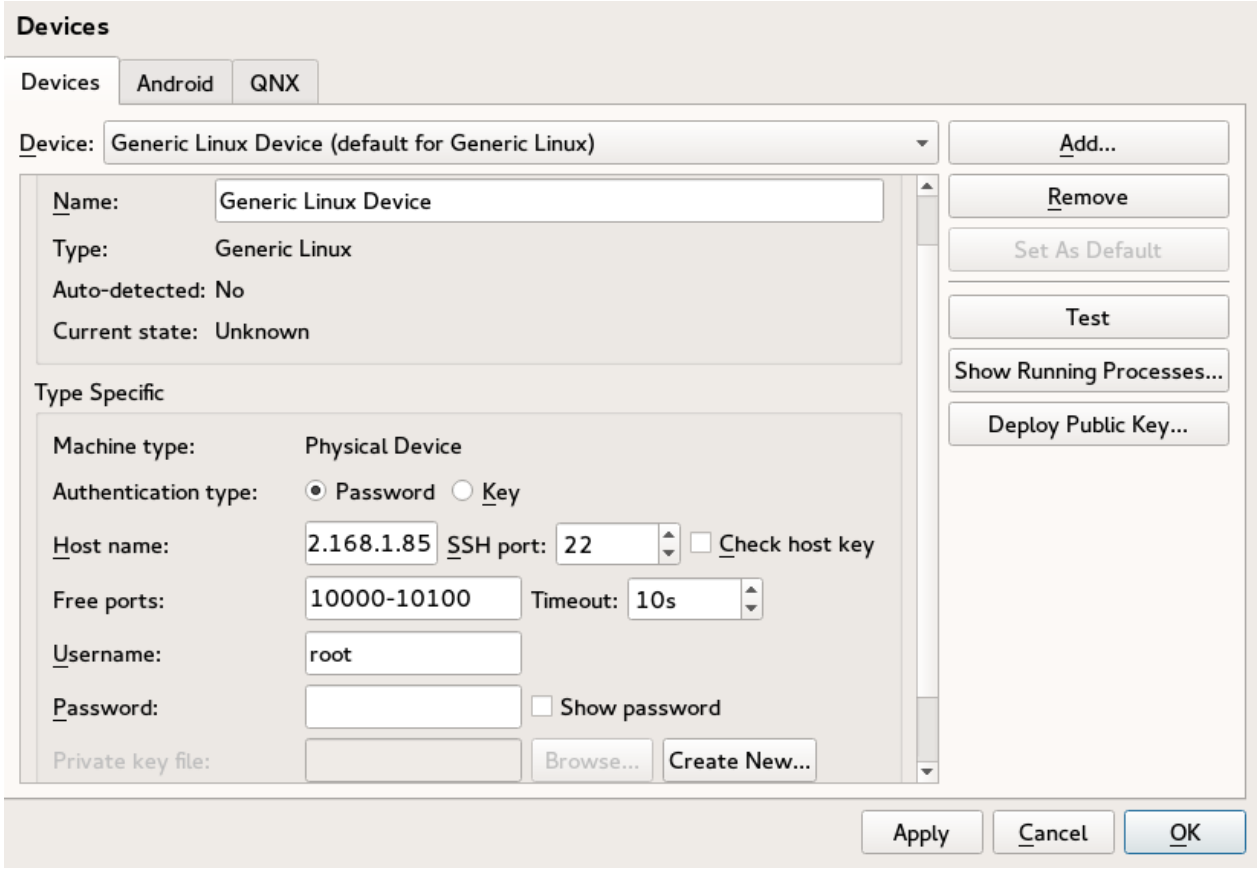


选择菜单 Tools->Options->Devices，点击 Add 按钮，配置如下：

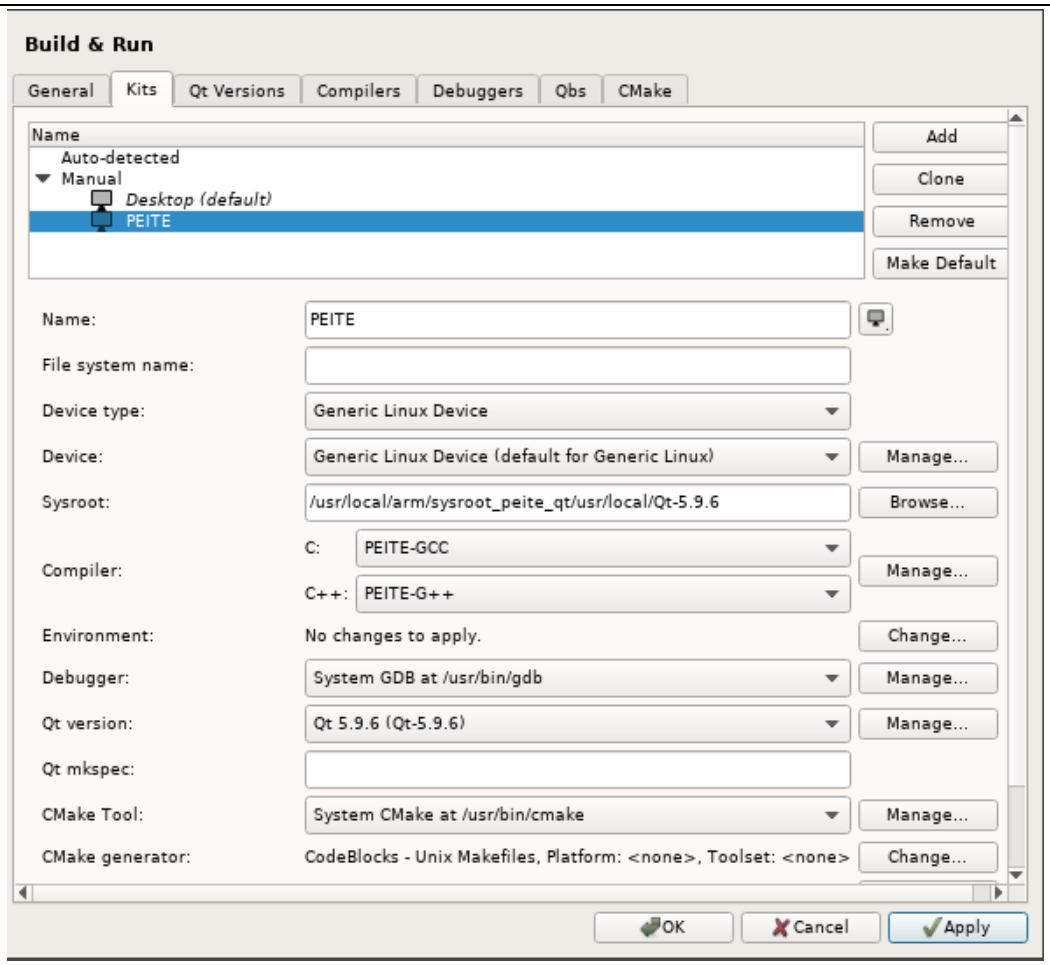


首先需要将开发板与主机在同一局域网内连接好，主机可以正常 ping 通开发板

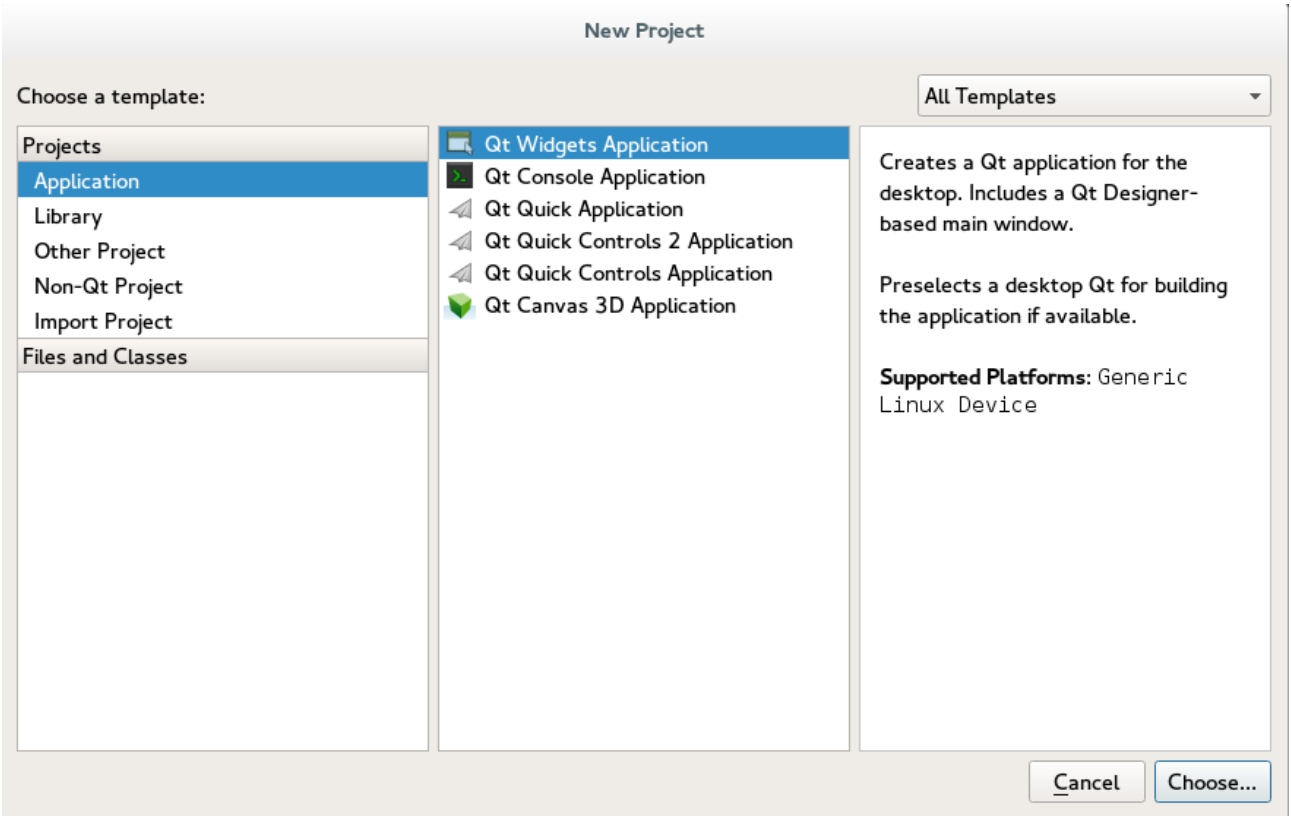
测试通过后的配置如下：

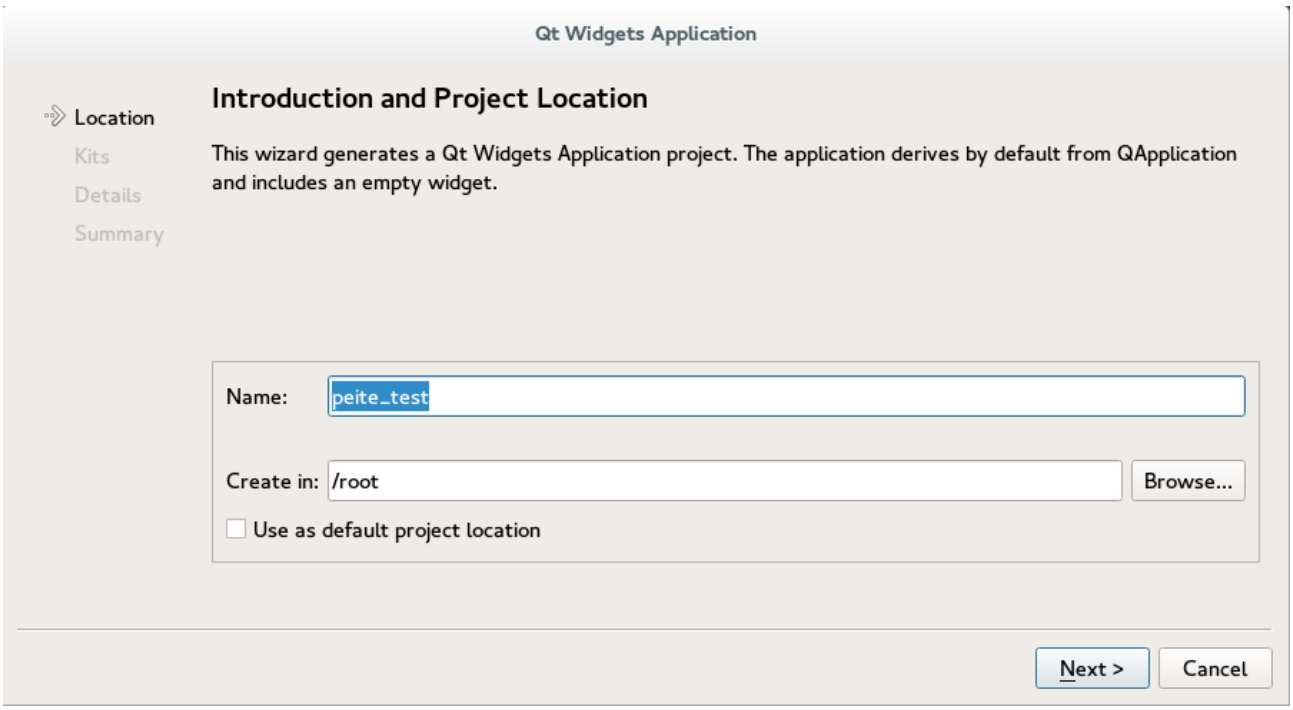


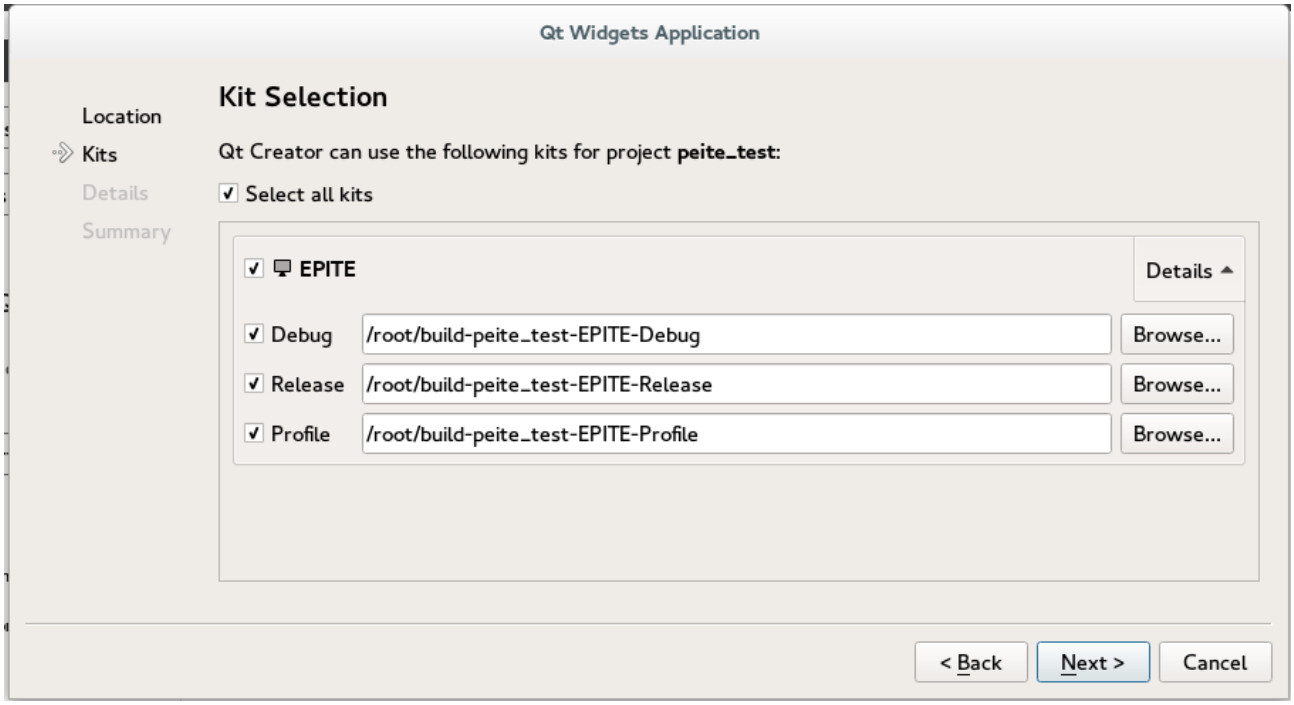
选择菜单 Tools->Options->Build & Run->Kits，点击 Add 按钮，配置如下：

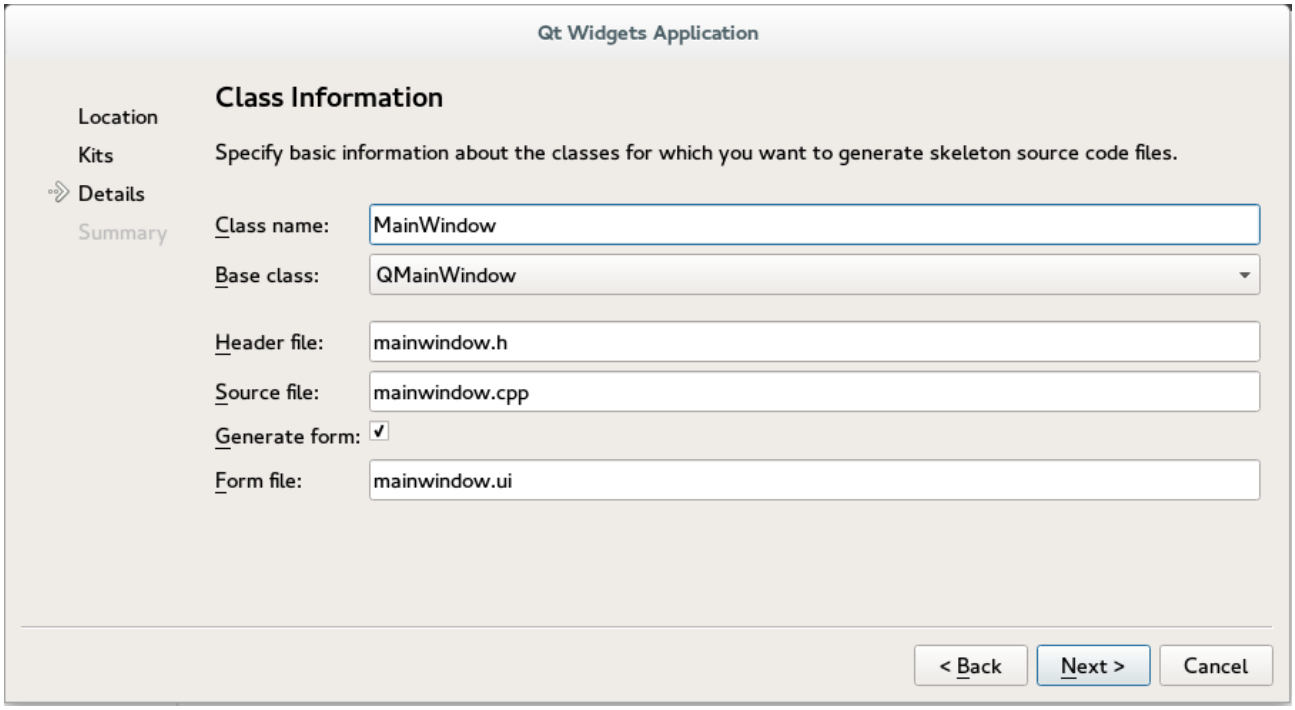

Build & Run
General
Kits
Qt Versions
Compilers
Debuggers
Qbs
CMake
Name
Auto-detected
Manual
Desktop (default)
PEITE
Add
Clone
Remove
Make Default
Name:
PEITE
File system name:
Device type:
Generic Linux Device
Device:
Generic Linux Device (default for Generic Linux)
Manage...
Sysroot:
/usr/local/arm/sysroot_peite_qt/usr/local/Qt-5.9.6
Browse...
Compiler:
C:
PEITE-GCC
C++:
PEITE-G++
Manage...
Environment:
No changes to apply.
Change...
Debugger:
System GDB at /usr/bin/gdb
Manage...
Qt version:
Qt 5.9.6 (Qt-5.9.6)
Manage...
Qt mkspec:
CMake Tool:
System CMake at /usr/bin/cmake
Manage...
CMake generator:
CodeBlocks - Unix Makefiles, Platform: <none>, Toolset: <none>
Change...
OK
Cancel
Apply

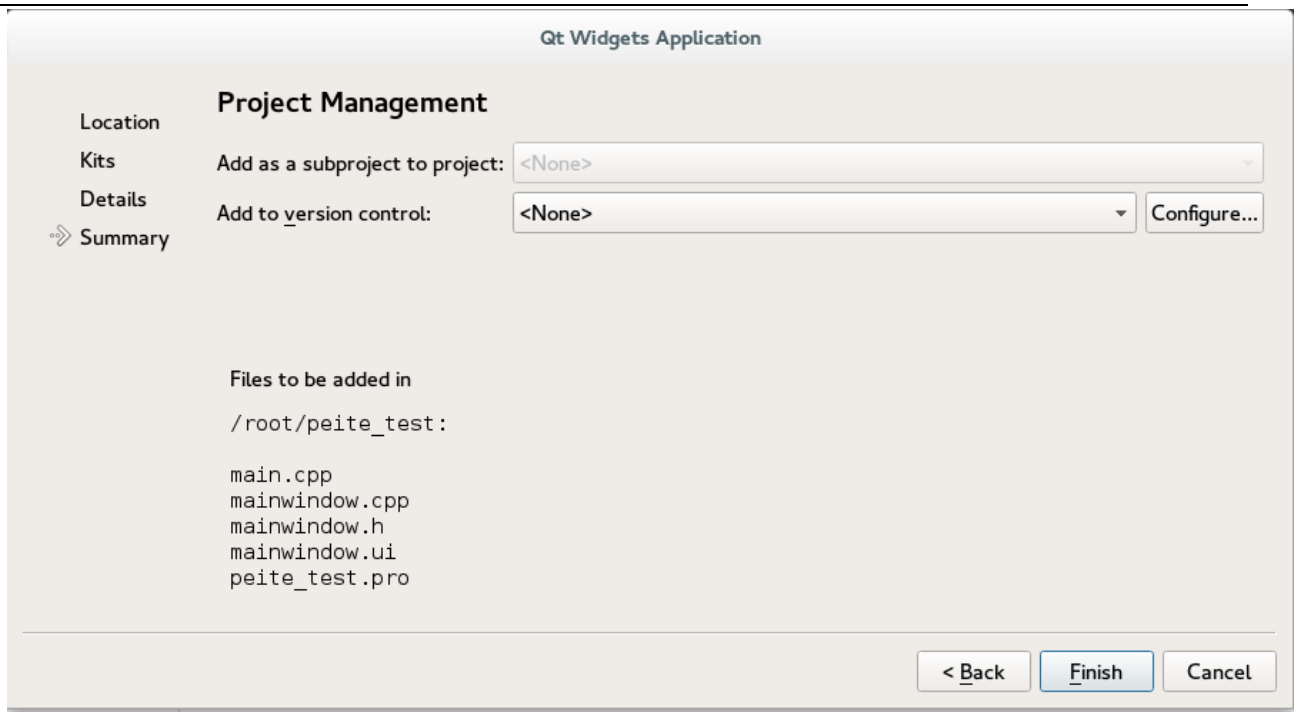
十三、创建并编译 QT 程序





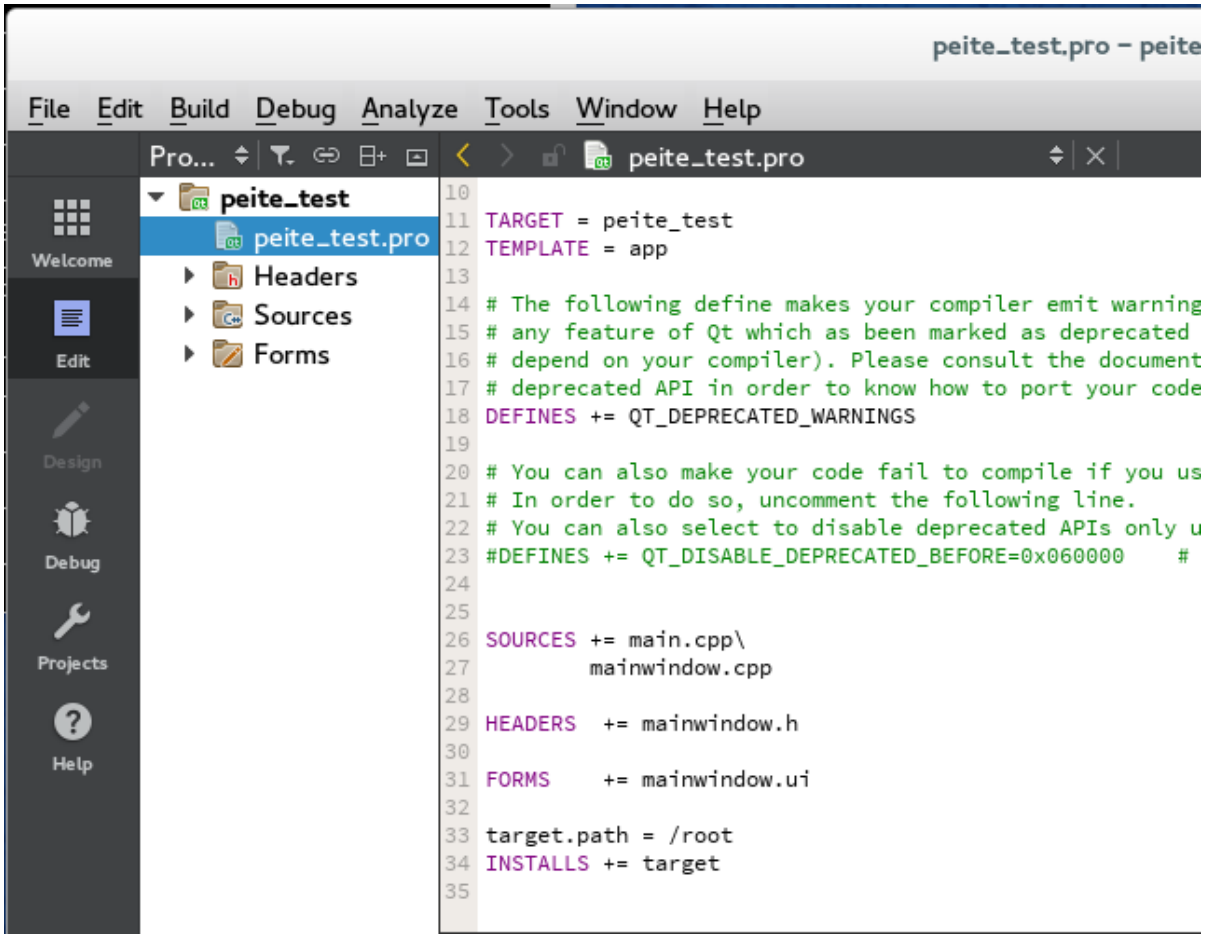

Qt Widgets Application
Location
Kits
Details
Summary
Kit Selection
Qt Creator can use the following kits for project peite_test:
Select all kits
EPITE
Details
Debug
/root/build-peite_test-EPITE-Debug
Browse...
Release
/root/build-peite_test-EPITE-Release
Browse...
Profile
/root/build-peite_test-EPITE-Profile
Browse...
< Back
Next >
Cancel


Qt Widgets Application
Location
Kits
Details
Summary
Class Information
Specify basic information about the classes for which you want to generate skeleton source code files.
Class name:
MainWindow
Base class:
QMainWindow
Header file:
mainwindow.h
Source file:
mainwindow.cpp
Generate form:
Form file:
mainwindow.ui
< Back
Next >
Cancel



完成创建后，需要修改工程目录下的 peite_test.pro 文件，在文件最后添加下面两行代码

```
target.path = /root
INSTALLS += target
```



然后在 QT Creator 中重新打开工程，编译、运行后，可以在开发板上查看运行效果。

十四、定制编译 QT 源码

客户可以自行编译 QT 的源码，可以对 QT 源码进行修改及定制，以下执行步骤需要 ROOT 权限。

1、 复制 QT_Source 目录及所有文件到编译主机。
2、 进入 QT_Source 目录，运行 config.sh 进行编译环境准备及选项配置。
3、 运行 build.sh 编译
4、 编译完成后的 QT 安装目录为 /usr/local/arm/sysroot_peite_qt/usr/local/Qt-5.9.6

十五、联系方式

地址 ：广州市天河区大观中路新塘大街鑫盛工业园 A1 栋 201
电话 ：020-85625526
传真 ：020-85625526-606
主页 ：http://www.gzpeite.net
淘宝店 ：https://shop149045251.taobao.com

核心板 ：王先生
移动电话：18926288206
电子信箱：18926288206@gzpeite.net
业务 QQ：594190286

定制研发：杨先生
微 信：18902281981
电子信箱：18902281981@gzpeite.net
业务 QQ：151988801

广州佩特电子科技有限公司

2020 年 8 月