华 清 远 见

FARSIGHT

始 于 2 0 0 4

嵌入式 ARM 开发平台

FS4412_M4

使用手册

华清远见教育集团 • 研发中心

 170-9108-5953

 2306275952/2668462267

 http://dev.hqyj.com

 support@farsight.com.cn

目 录

第 1 章　平台概述

华清远见 ARM FS4412M4 开发平台（下文简称 FS4412M4）是由华清远见研发中心专为培训教学和项目研发定制的高性能的 ARM Cortex-A9 开发平台和控制性能优越的 ARM Cortex-M4 平台。FS4412 采用三星 ARM Exynos 4412 四核处理器，运行主频可高达 1.5GHz，其处理速度和节能能力比起双核大幅提高。对比上一代的双核处理器，四核能提供翻倍的处理能力以及减半的功耗，这也为精细的显示效果、1080P 拍摄及播放、以及各方面的超高流畅度运行莫定了基础。FS4412 提供了丰富的板载资源以及扩展接口，搭配华清远见研发中心自主研发的 FS-JTAG ARM 仿真器，无论是用于教学、自学还是项目研发，都是非常完善的开发平台。Cortex-M4 在硬件上新增了硬件 FPU 单元以及 DSP 指令，同时，M4 的主频也提高了很多，达到 168MHZ，这使得其尤其适用于需要进行浮点运算或 DSP 处理的应用。

1.1　平台特色

1）采用 ARM 微控制器+ARM 微处理器多核心方式设计；

2）微控制器标配 ARM Cortex-M4，可选配核心板接口兼容的：Cortex-M0 核心板、Cortex-M3 核心板、51 单片机核心板、AVR 单片机核心板、430 核心板等；

3）微处理器标配 ARM Cortex-A9，可选配兼容的 ARM Cortex-A8 系统，和 Cortex-A53 系统；

4）多种 Arduino 扩展资源，微控制器和微处理器都可以直接访问；

5）可选配多种无线传感网络模块，包含：ZigBee、WiFi、IPv6、Bluetooth BLE、433M；

6）可选配多种 RFID 模块，包含：125K、13.56M、915M、NFC、指纹识别、2.4G 等；

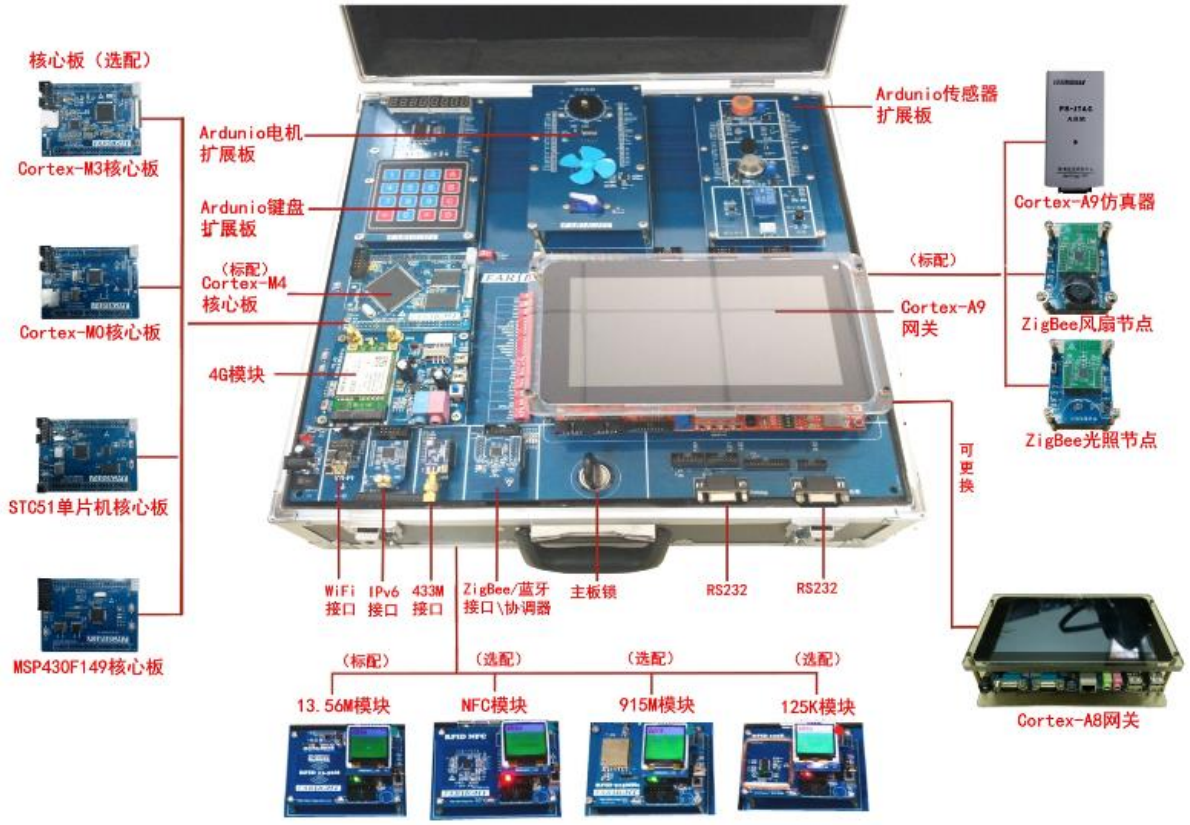
7）系统的 ARM 微控制器裸机和 ARM 微处理器裸机实验，标配相关仿真器；

8）完整的 Linux 移植、Linux 驱动、Android 底层实验；

9）多个综合嵌入式、物联网实训案例；

10）完善的实验指导书及配套教材、配套培训服务。

1.2 Cortex-A9 ARM 系统硬件资源介绍



	功能部件	型号参数
核心配置	CPU	- Samsung Exynos 4 Quad（四核处理器） - 32nm HKMG - 1433 MHz（最多可以达 1.6GHz）
	GPU	- Mali-400MP（主频可达 400MHz）
	屏幕	- LVDS 40 Pin 显示接口 - 7 寸 1024 x 600 高分辨率显示屏 - 多点电容触摸屏
	RAM 容量	- 1GB DDR3（可选配至 2GB）
	ROM 容量	- 4GB eMMC（可选配至 16GB）
	多启动方式	- eMMC 启动、MicroSD(TF)/SD 卡启动 - 通过控制拨码开关切换启动方式 - 可以实现双系统启动
板载接口	存储卡接口	- 1 个 MicroSD(TF)卡接口 - 1 个 SD 卡接口 - 最高可扩展至 64GB
	摄像头接口	- 20 Pin 接口，支持 OV3640 300 万像素摄像头
	HDMI 接口	- HDMI A 型接口 - HDMI v1.4a - 最高 1080p@30fps 高清数字输出
	JTAG 接口	- 20 Pin 标准 JTAG 接口 - 支持 FS-JTAG Cortex-A9 ARM 仿真器 - 独家支持详尽的 ARM 裸机程序
	USB 接口	- 1 路 USB OTG - 3 路 USB HOST 2.0（可扩展 USB-HUB）
	音频接口	- 1 路 Mic 接口 - 1 路 Speaker 耳机输出 - 1 路 Speaker 立体声功放输出（外置扬声器）
	网卡接口	- DM9000 百兆网卡
	RS485 接口	- 1 路 RS485 总线接口
	CAN 总线接口	- 1 路 CAN 总线接口
	串口	- 1 路 5 线 RS232 串口 - 2 路 3 线 RS232 串口 - 1 路 TTL 串口

	扩展 I/O 接口	- 1 路 I2C（已将 1.8V 转换为 3.3V） - 1 路 SPI（已将 1.8V 转换为 3.3V） - 3 路 ADC（1 路含 10K 电阻） - 多路 GPIO、外部中断（已将 1.8V 转换为 3.3V）
板级资源	按键	- 1 个 Reset 按键 - 1 个 Power 按键 - 2 个 Volume（+/-）按键
	LED	- 1 个电源 LED - 4 个可编程 LED
	蜂鸣器	- 1 个无源 PWM 蜂鸣器
	红外接收器	- 1 个 IRM3638 红外接收器 - 可选配红外遥控器在 Android 下使用
	温度传感器	- 1 个 DS18B20 温度传感器
	ADC	- 1 路电位器输入（Android 下可模拟电池电量）
	RTC	- 1 个内部 RTC 实时时钟
选配模块	Wi-Fi 模块	- 802.11 a/b/g/n/ac 无线网络
	4G 模块（5 模）	-支持 Android 系统下语音通话、短信、GPS 定位、4G 上网
	蓝牙模块	- USB 蓝牙模块
	VGA 模块	- 高质量的 VGA 输出
	RFID 模块	- FS_RC522 13.56MHz RFID 模块
	ZigBee 模块	- FS-CC2530 ZigBee 模块 - 配套可以选择各类传感器节点
	Bluetooth 模块	- FS-CC2540 Bluetooth Low Energy (Bluetooth 4.0)模块 - 配套可以选择各类传感器节点
	IPv6 模块	- FS-STM32W108 IPv6 模块 - 配套可以选择各类传感器节点
	低功耗 Wi-Fi 模块	- 低功耗串口 Wi-Fi 模块 - 配套可以选择各类传感器节点
	操作系统支持	- Android4.0、Linux3.0

1.3 FS4412M4 系统软件资源介绍

1.3.1 ARM 体系结构与接口技术

FS4412M4 开发平台配合华清远见研发中心开发的 FS-JTAG Cortex-A8/A9 仿真器 ,可以实现接近 ARM 官方仿真器的功能。FS-JTAG 支持 Windows XP/7/8/8.1 32bit/64bit 全系列 Windows 平台，而且仿真器使用全套开源软件开发环境，IDE 为使用相当广泛的 Eclipse，开发者可以轻易上手；编译器则使用 GNU GCC，代码可以和 Linux 下的代码实现无缝衔接，使开发者的学习难度大大降低，学习关联性紧密连接。

ARM 体系结构与接口技术部分实验配套有《ARM 体系结构与接口技术》实验手册作为指导，实验的设置由简入深包含了 GPIO、中断、A/D 转换、定时器 (看门狗、PWM、RTC 等) 、串口通讯等基础实验、I^2C、SPI 等总线接口实验和内存管理 MMU 实验等等。部分该手册不仅可以让用户一步一步做出实验现象，还有对相关知识原理的解读 ,让新手学习的同时 ,也能让有经验的开发者对体系更加清晰 ,对知识更加深入。

实验类别	实验名称
ARM 体系结构与接口技术	1、FS-JTAG 开发环境搭建实验
	2、ARM 汇编实验
	3、ARM 寄存器读写实验
	4、GPIO(LED)控制实验
	5、Interrupt 按键中断驱动
	6、PWM 蜂鸣器实验
	7、RTC 实时时钟实验
	8、Alarm 闹钟实验
	9、ADC 转换实验
	10、Uart 串口通信实验
	11、WatchDog 复位实验
	12、WatchDog 中断实验
	13、DS18B20 温度传感器实验
	14、I^2C 重力感应实验
	15、SPI 总线实验
	16、SPI CAN 总线通信实验
	17、MMU 内存管理实验

1.3.2 Linux 移植驱动及应用开发

Linux 移植驱动及应用开发包含三个部分，有上到下为 Linux 应用程序开发、Linux 移植部分、Linux 驱动部分。

实验类别	实验名称
Linux 系统部分	1、Linux 开发环境搭建实验
	2、Linux 常用命令使用实验
	3、Linux 系统 vi 编辑器使用实验
	4、Linux 系统 GCC 编译器的使用实验
	5、Linux 系统 GDB 调试工具使用实验
	6、Linux 系统 Makefile 编写实验
	7、Linux 系统 I/O 编程实验
	8、Linux 系统文件目录操作编程实验
	9、Linux 系统文件信息的遍历实验
	10、Linux 系统 fork 等系统调用实验
	11、fork 等函数编写执行命令实验
	12、Linux 系统守护进程实验
	13、Linux 系统无名管道通信实验
	14、Linux 系统有名管道通信实验
	15、Linux 系统信号机制实验
	16、Linux 系统信号量实验
	17、Linux 系统共享内存通信实验
	18、Linux 系统 TCP 网络协议编程实验
	19、Linux 系统 UDP 网络协议编程实验
	20、Linux 系统 select I/O 复用实验
	21、Linux 系统消息队列实验
	22、Linux 系统多线程实验
	23、Linux 串口通信实验
	24、RFID 读写卡实验

实验类别	实验名称
Linux 系统移植部分	1、Linux 系统配置 TFTP 实验
	2、Linux 系统配置 NFS 实验
	3、BootLoader（Uboot）开发实验
	4、全新 Linux 内核编译实验
	5、以太网卡驱动移植（网络驱动开发实验）
	6、eMMC 驱动移植
	7、USB 驱动移植
	8、SD 卡驱动移植

	9、LCD 驱动移植
	10、根文件系统开发实验
	11、Cramfs 文件系统制作实验
	12、Ramdisk 文件系统制作实验

实验类别	实验名称
Linux 驱动实验	1、简单字符设备驱动实验
	2、pipe 驱动实验
	3、poll 驱动实验
	4、异步通知驱动实验
	5、秒表字符设备驱动驱动
	6、tasklet 实验驱动
	7、工作队列实验驱动
	8、利用 udev、sys 动态创建设备结点实验驱动
	9、GPIO(LED)驱动实验
	10、按键中断驱动实验
	11、PWM 蜂鸣器驱动实验
	12、ADC 转换驱动实验
	13、Watch Dog 看门狗驱动实验
	14、温度传感器驱动实验

1.3.3 Android 底层及应用开发

实验类别	实验名称
界面编程	1、 线性布局
	2、 框架布局
	3、 相对布局
	4、 表格布局
	5、 TextView 的使用
	6、 AutoCompleteTextView
	7、 按钮的使用
	8、 图片按钮
	9、 单选按钮
	10、 开关按钮
	11、 复选框的使用

	12、 日期选择器
	13、 时间选择器
	14、 文本编辑框
	15、 进度条
	16、 打分进度条
	17、 警告对话框
	18、 进度对话框
	19、 日期和时间选择对话框
	20、 菜单
	21、 菜单 2
	22、 菜单 3
	23、 上下文菜单
	24、 子菜单
	25、 下拉列表 Spinner
	26、 下拉列表 Spinner 2
	27、 Toast
	28、 图像视图（ImageView）的使用
	29、 列表视图
	30、 选择列表视图
	31、 复杂列表视图
	32、网格视图
	33、 选项卡
	34、 滚动视图
	35、 画廊
	36、 自定义样式
	37、 使用 selector 来实现点击效果
	38、 ViewFlipper 的使用
	39、 TextSwitcher 的使用
	40、 ImageSwitcher 的使用
	41、 逐帧动画
	42、 在 xml 文件中定义逐帧动画
	43、 补间动画
	1、 利用 ActivityManager 获得系统信息
	2、 使用 Intent 在组件间传递数据
	3、 使用系统 Intent
组　用	4、使用 PendingIntent 来实现闹钟

线程进程	5、 状态栏提醒
	6、 启动服务
	7、 绑定服务
	8、使用 BroadcastReceiver
	9、 接收系统消息
	1、 使用 Handler 处理消息
	2、 使用 Handler 异步更新列表视图
	3、 使用 AsyncTask 执行异步任务
数据存取	1、 取得 Android 的缓存文件夹
	2、 使用 SharedPreference 来存取数据
	3、 使用 SQLite 数据库
	4、 使用 ContentResolver 来读取设置
	5、 读取联系人列表
	6、 往联系人列表中写入数据
	7、 使用 DOM 读取 XML 文件内容
	8、 使用 SAX 读取 XML 文件
图形图像设计	1、使用 Canvas 绘图
	2、 Paint 的用法
	3、 设置字体
	4、 使用 Path 来指定绘制图形的路径
	5、 使用 Shader 效果
	6、 使用 Shader 实现的放大镜效果
	7、 使用 Matix 来实现图片缩放
	8、 使用 Movie 播放动画图片（gif）
	9、 SurfaceView 的使用
事件处理	1、 Activity 中的事件处理方法
	2、 处理组件之间的跳转顺序
	3、 手势处理
	4、 通过手势缩放图片
体 媒	1、 AudioManager 的使用
	2、 使用相机拍照（迷你图片查看器）
	3、 使用相机录像

	4、 录制音频
	5、 播放音频（随身听实验）
	6、 一个更加完善的播放器
	7、 视频播放
	8、 设置铃声
	9、 通过 MediaStore 获得音频信息
传感器编程和桌面组件	1、 列出所有的传感器
	2、 温度计
	1、 获取网络状态（TCP 实验）
	2、 访问网页（UDP 实验）
网络编程	3、 从网络读取数据
	4、 提交数据到网络
	5、 使用 URLConnection 下载数据
	1、 获得当前的国家/地区
	2、 输出当地货币
Android 应用程序国际化	3、日期和时间的格式化
	4、 资源的格式化
Google 服务	1、 Google 地图服务
	1、实现 SurfaceView 类
	2、实现绘图线程类
	3、实现图层中的素材类
Android 游戏编程基础	4、绘制素材
	5、使用矩阵改变素材
	6、剪裁素材
	7、播放音乐
Android NDK 编程	1、 创建 NDK 项目
	1、大容量存储 USB 硬盘实验
	2、资源管理器（文件浏览器）
	3、语音识别实验
	4、Android 四大组件实验

其它 Andorid 实验及项目	5、智能网络家电实验（智能家居实验）
	6、RSS 阅读器
	7、文本编辑器实验
	8、日程管家实验（闹钟）
	9、新浪微博实验
	10、电子书阅读器
	11、网络浏览器
	12、文件和进程管理器
	13、I'm Here 实验（GPS 定位）
	14、3D 演示实验
	15、基于 Android 平台的社交类应用程序（我们约会吧）
	16、军旗
	17、推箱子实验
	18、连连看
	19、记忆卡片
	20、天气预报

实验类别	实验名称
Cortex-M4 实验部分	1、跑马灯实验
	2、蜂鸣器实验
	3、直流电机驱动实验
	4、舵机实验
	5、步进电机实验
	6、UART 串口 1 数据收发实验
	7、继电器实验
	8、SYSTick 系统滴答实验
	9、按键扫描数码管显示实验
	10、ADC 模数转换实验
	11、IR 红外接收实验
	12、4.3 寸 TFT 彩屏液晶屏实验
	13、4.3 寸 TFT 彩屏触摸屏实验
	14、温度采集实验
	15、光电开关实验
	16、UCOSII 操作系统创建 1 个任务实验
	17、UCOSII 操作系统创建多个任务实验
	18、UCOSII 操作系统-信号量实验
	19、UCOSII 操作系统移植实验
	20、UCOSII 操作系统创建 1 个任务实验
	21、UCOSII 操作系统创建多个任务实验

	22、UCOSII 操作系统-信号量控制灯实验
	23、综合测试例程

1.3.4 华清远见开发环境

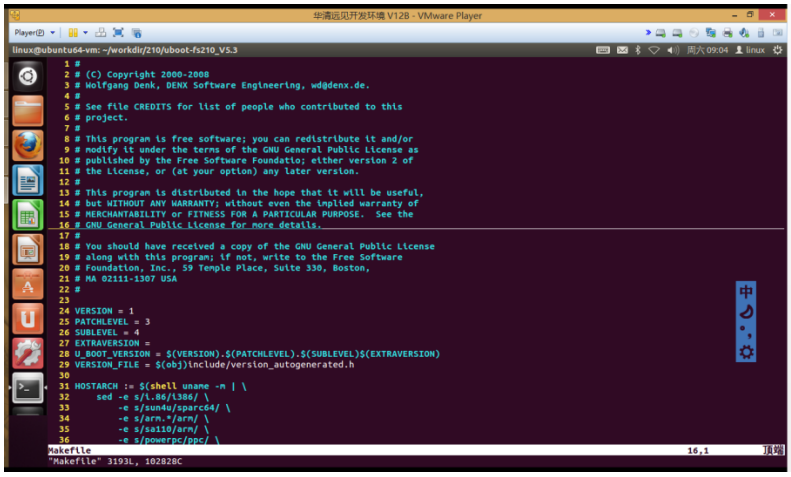
华清远见开发环境是基于 Ubuntu 12.04 LTS 64-bit 操作系统搭建的，使用 VMware Player（免费版）作为虚拟机工具软件（用户也可以使用 VMware 公司所提供的付费版虚拟机软件 VMware Workstation 代替 VMware Player）。本开发环境可用作嵌入式 Linux 和 Android 的编译与开发。

本开发环境在 Ubuntu 12.04 64-bit LTS 基础上，安装了编译调试 Bootloader、Linux、Android 系统所需要的工具和依赖的库，用户可以在无需额外操作的基础上，直接使用本开发环境进行嵌入式的学习和工作。

本开发环境在 Ubuntu 12.04 64-bit 基础上，安装配置了如下工具：

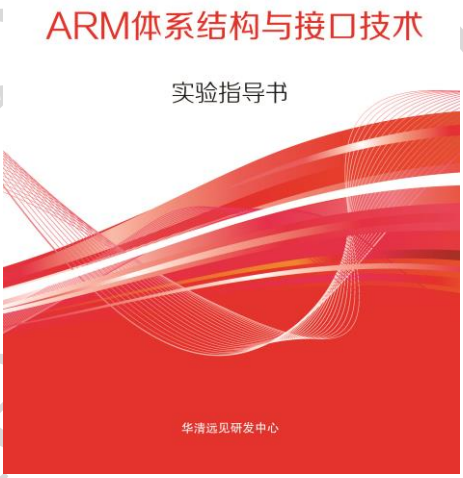
- **将 GCC、G++编译器版本从 4.6 降至 4.4；**
- **安装了 Android 编译所需要的工具和库（source.android.com）；**
- **安装 SUN JAVA JDK 6；**
- **安装内核编译所依赖的工具包；**
- **解决了 libncurses 32 位和 64 位不能同时安装导致编译 Android 和配置内核软件冲突的问题；**
- **安装制作安卓文件系统 yaffs2 格式 mkyaffs 工具；**
- **添加了常用的 arm-linux 交叉工具链，版本号为 4.3.2、4.4.6、4.5.1；**
- **安装 Vim、Ctags；**
- **安装 Vim 常用插件；**
- **安装配置 TFTP；**
- **安装配置 NFS 网络文件系统服务；**
- **安装 SSH 工具网络服务程序；**
- **安装 Kermit 串口调试工具；**
- **安装 Sogou 输入法；**
- **关闭 Ubuntu 更新提示；**

下图是使用华清远见开发环境编译源码：

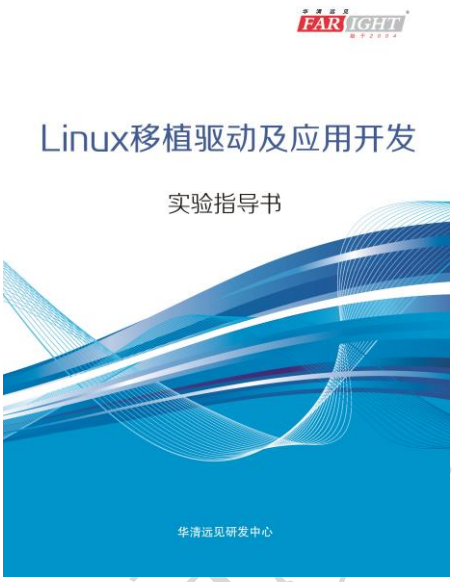


1.4 其他资源

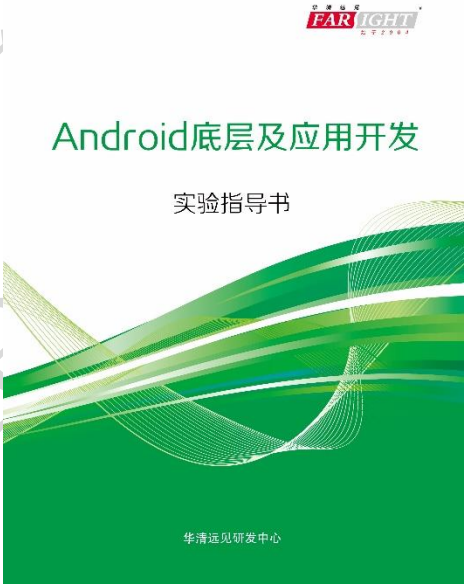
1.4.1 《ARM 体系结构与接口技术-实验指导书》



1.4.2 《Linux 移植驱动及应用开发-实验指导书》



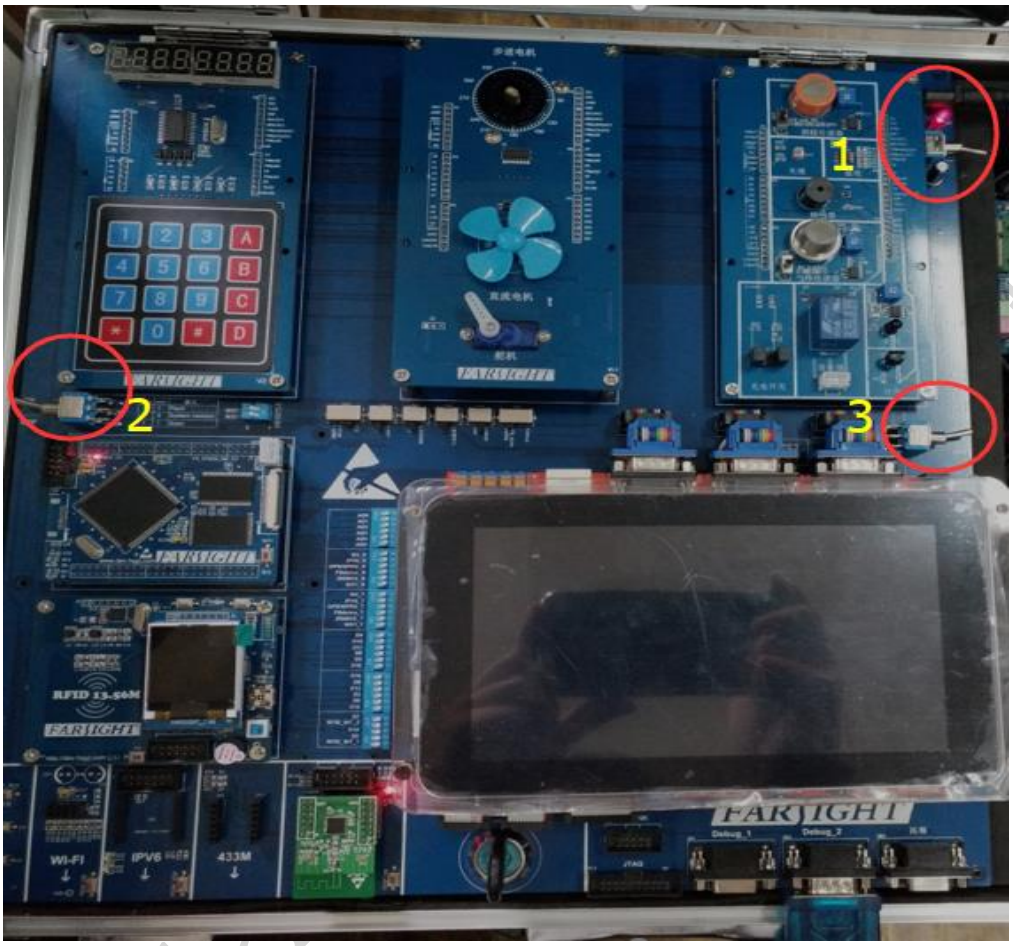
1.4.3 《Android 底层及应用开发-实验指导书》



第 2 章　Android 下功能演示

平台出厂默认在 EMMC 中运行 Android 系统。SD 卡中运行 Ubuntu 系统。

2.1　系统开机



如图，打开 3 个电源开关。

1 号电源开关为设备的总电源开关；

2 号电源开关为 4412 部分的电源开关

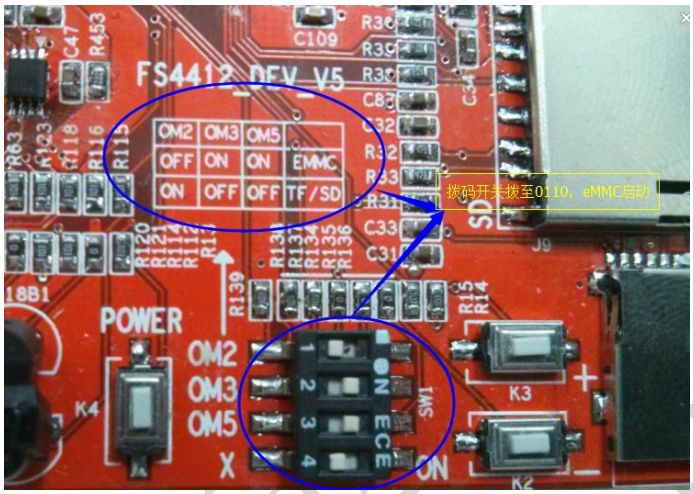
3 号电源开关为 M4 部分的电源开关

FS4412 系统功能演示之前，需要将 1 号、2 号电源开关打开。

2.1.1 eMMC 启动

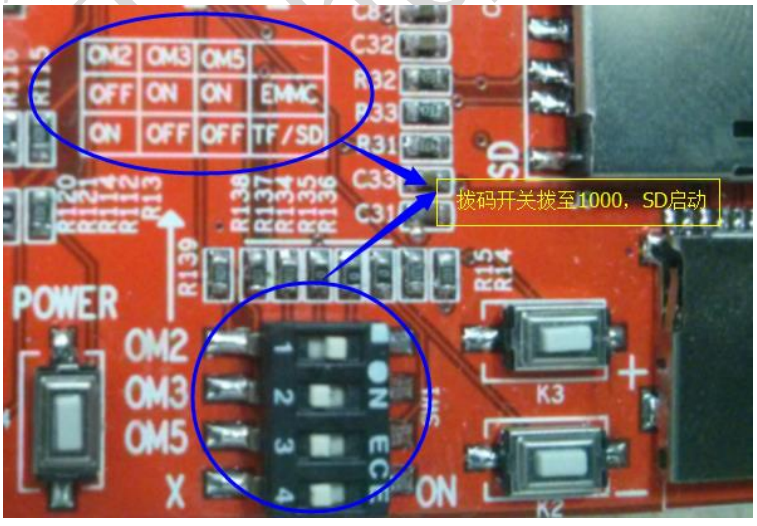
FS4412 开发板有两种启动方式，eMMC 启动和 TF/SD 卡启动。

拨动拨码开关至【0110】，如下图所示，系统开机即从 eMMC 启动。

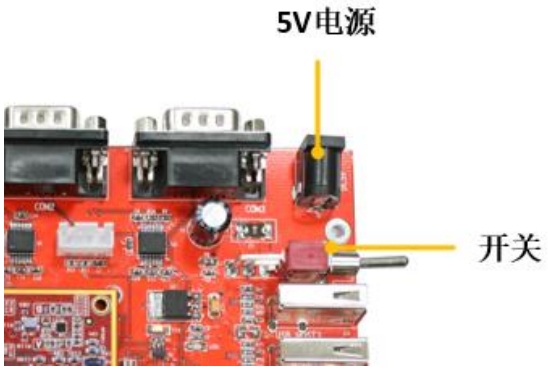


2.1.1 TF/SD 卡启动

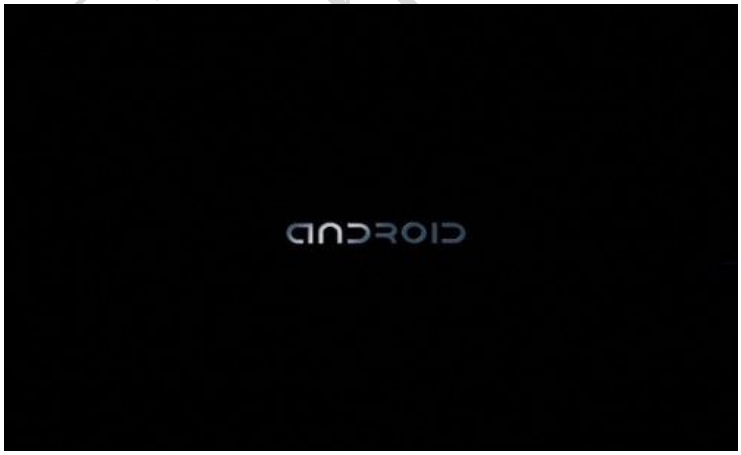
拨动拨码开关至【1000】，如下图所示，系统开机即从 TF/SD 卡启动。

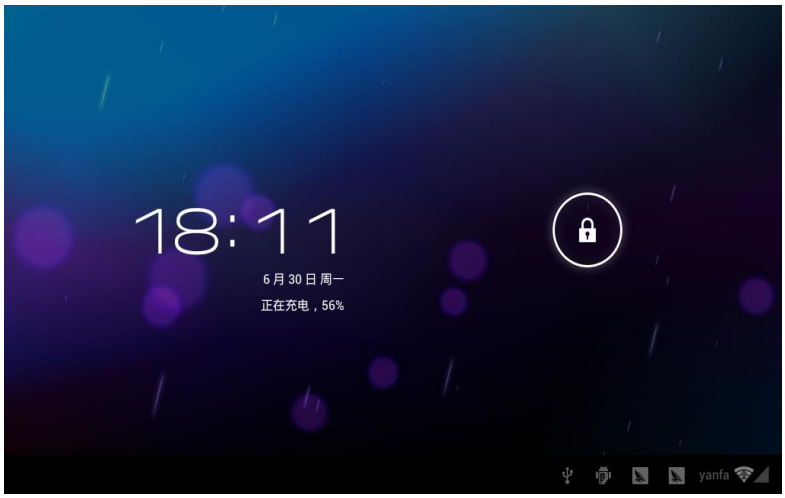


连接 5V 电源适配器至开发板右上侧电源接口处，并将电源开关拨至开启处（向上开启，向下关闭），系统即启动。



在启动过程中，系统会分别显示华清远见开机 Logo 和 Android 启动界面。

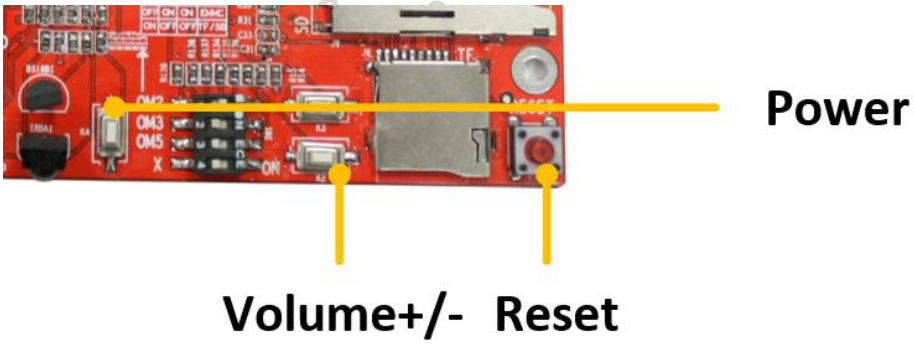




2.2 Reset/Power/Volume 按键

FS4412 开发平台包含 4 个按键。包含 1 个 Reset 复位按键（长按复位的模式），1 个 Power 电源按键、2 个音量控制按键。

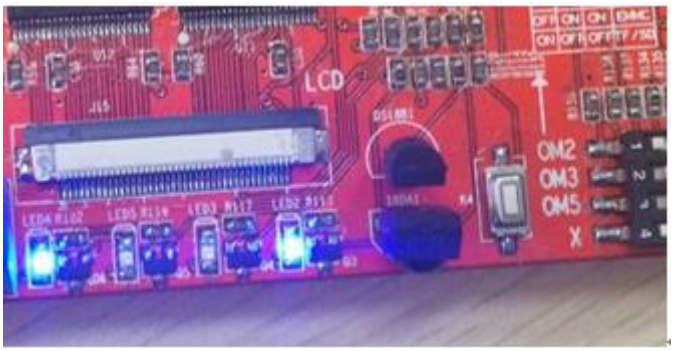
其中 Reset 复位按键的设计方便了开发人员调试，而 Power 和 Volume+/-按键的则完全按照现在主流的手机和平板的设计。其中 Power 键作为开发板的电源按键，可以使开发板唤醒或者睡眠，Volume+/-按键则可以调节 Android 下的音量。而且，FS4412 平台还具备类似同时按住 Power 键和 Volume - 按键，可以对 Android 的当前屏幕截图等实用的功能。



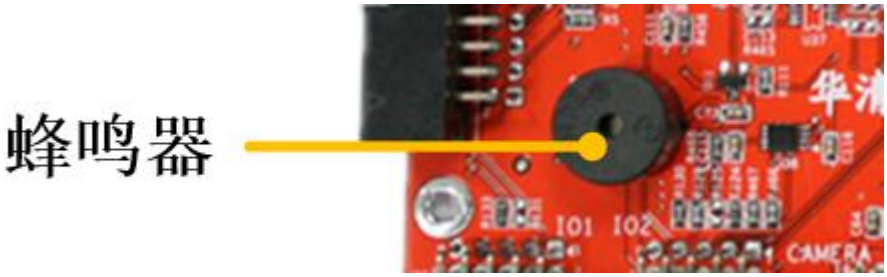
2.3 可编程 LED 灯

在 Android 下打开【功能演示】应用程序，界面如下图所示。

点击 LED1-LED4，点亮 LED 灯



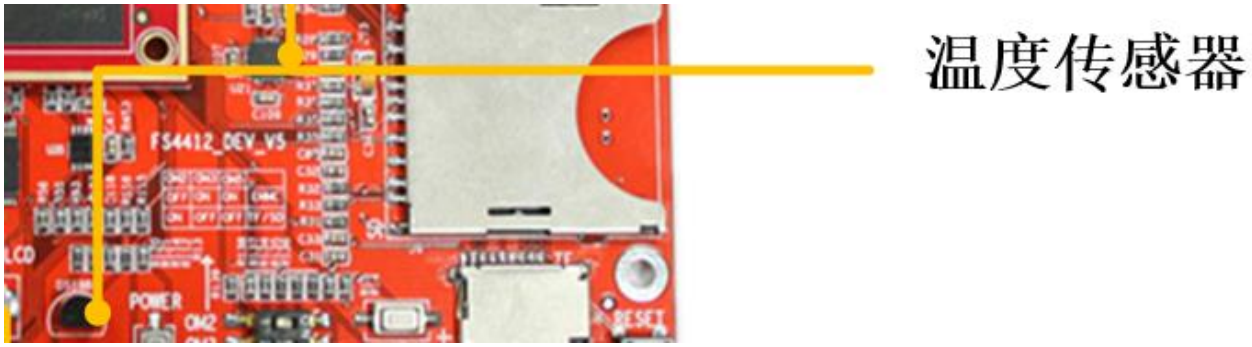
2.4 蜂鸣器



在 Android 下打开【功能演示】应用程序，界面如下图所示。

拖拽蜂鸣器调节蜂鸣器频率。

2.5 温度传感器



在 Android 下打开【功能演示】应用程序，界面如下图所示。

尝试触摸温度传感器，可以看到温度升高。

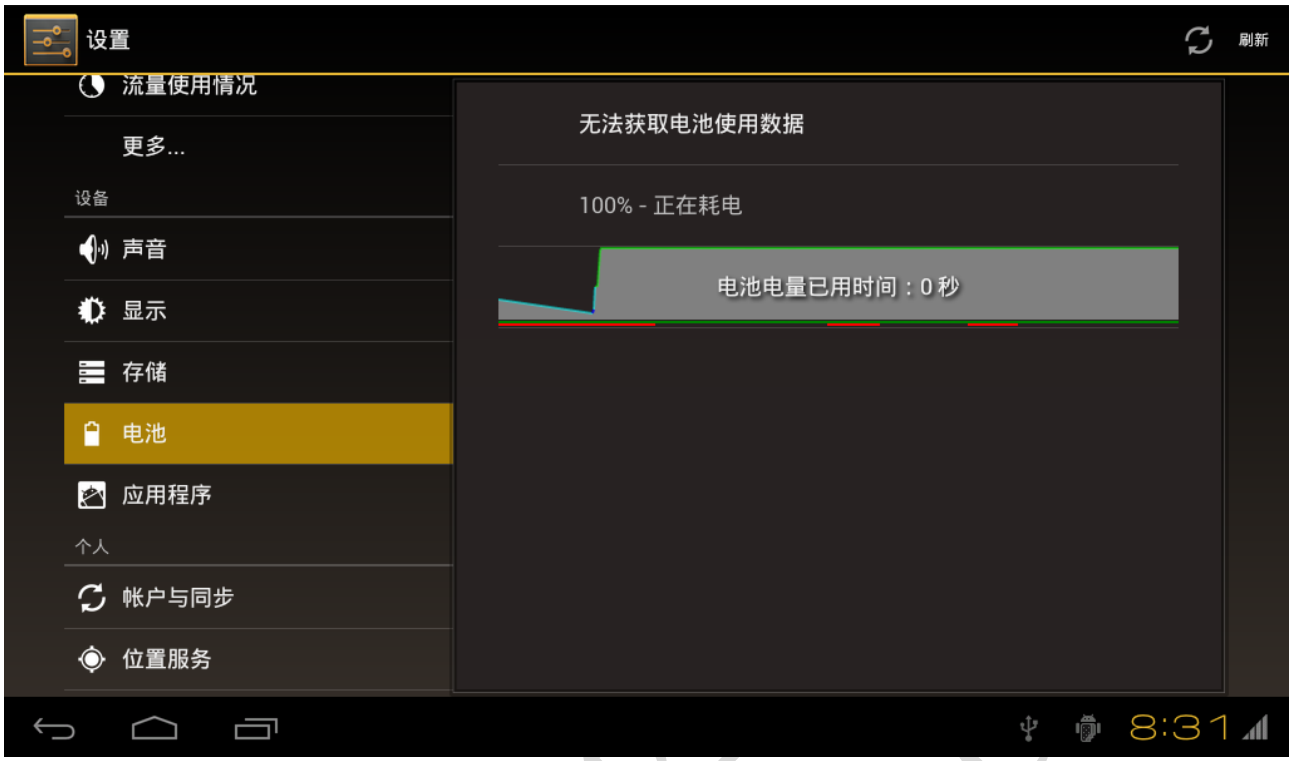
2.6 ADC（模拟电量）

电位器

在 Android 下打开【功能演示】应用程序，界面如下图所示。

旋转电位器，可以看到电量变化（模拟量转换为数字量），范围从 0-100。

同样的，从 Android 的系统设置界面中可以看到这个值同样在变化。



2.7 微信远程控制设备

本设备支持使用微信控制相关设备硬件，具体操作如下所示。

在 Android 下打开【功能演示】应用程序，界面如下图所示。

在右上角微信控制输入框输入 ID，然后点击生成二维码

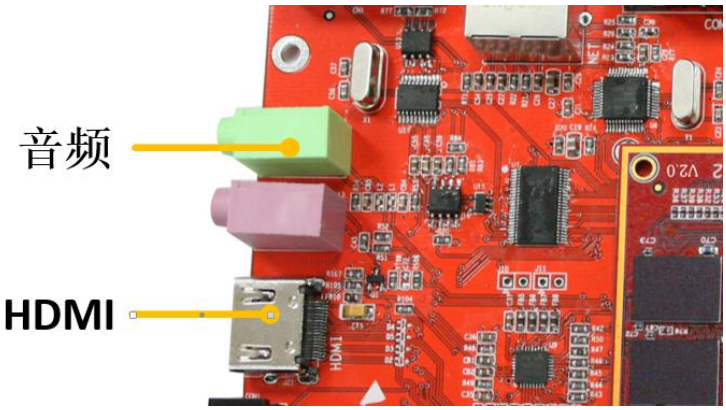
使用手机微信扫描右上角二维码，关注微信公众号

在手机端微信的下部可以查看当前设备状态，并对设备硬件进行控制等

2.8 媒体播放

FS4412 开发平台可以流畅的播放 1080P 的高清视频，以及高清游戏。连接耳机及麦克风至下图的音频接口，和 HDMI 接口，即可把开发板像高清播放器一样使用。

拷贝文件到 SD 卡的 FAT 分区或者使用 Android 同步工具（如豌豆荚等）同步视频到开发板内，即可以使用 Android 下的图库应用（图库仅仅可以播放硬解码格式，使用 MX Player 或者 Mobo Player 等媒体播放软件可以播放更多的格式）播放视频。



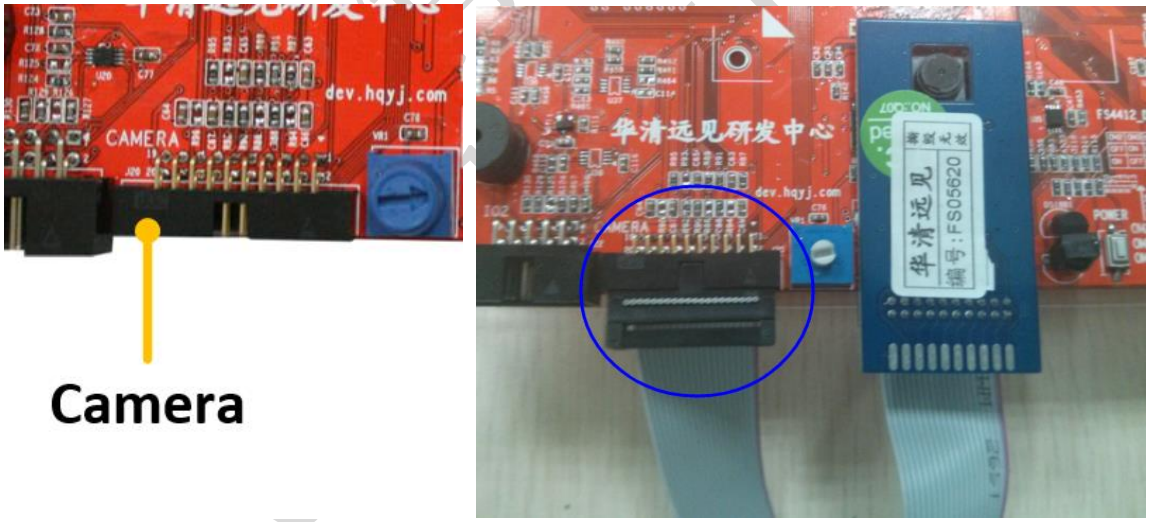
2.9 重力感应/陀螺仪

FS4412M4 开发平台内置重力感应/陀螺仪模块，最简单的测试方法就是在 Android 设置中的显示菜单下勾选【自动旋转屏幕】功能，然后上下左右转动开发板，可以看到开发板的屏幕会按照转动的方向旋转角度。

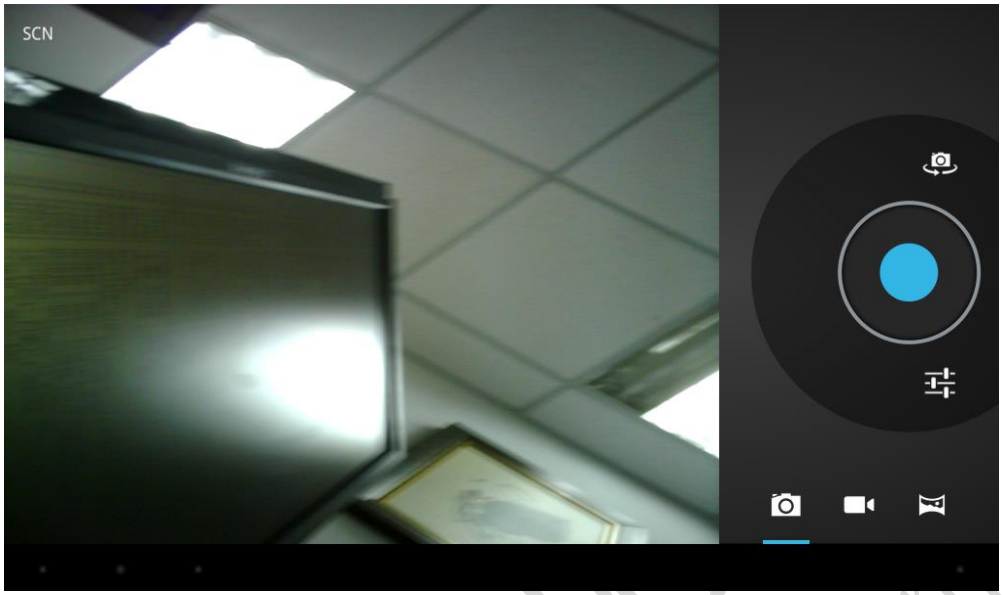
也可以使用相关的测试应用来测试 FS4412 开发平台的重力感应/陀螺仪模块，如下图所示的 SensorDimo 应用。转动开发板，可以看到界面 x、y、z 轴数值在不断变化。

2.10 拍照和摄像

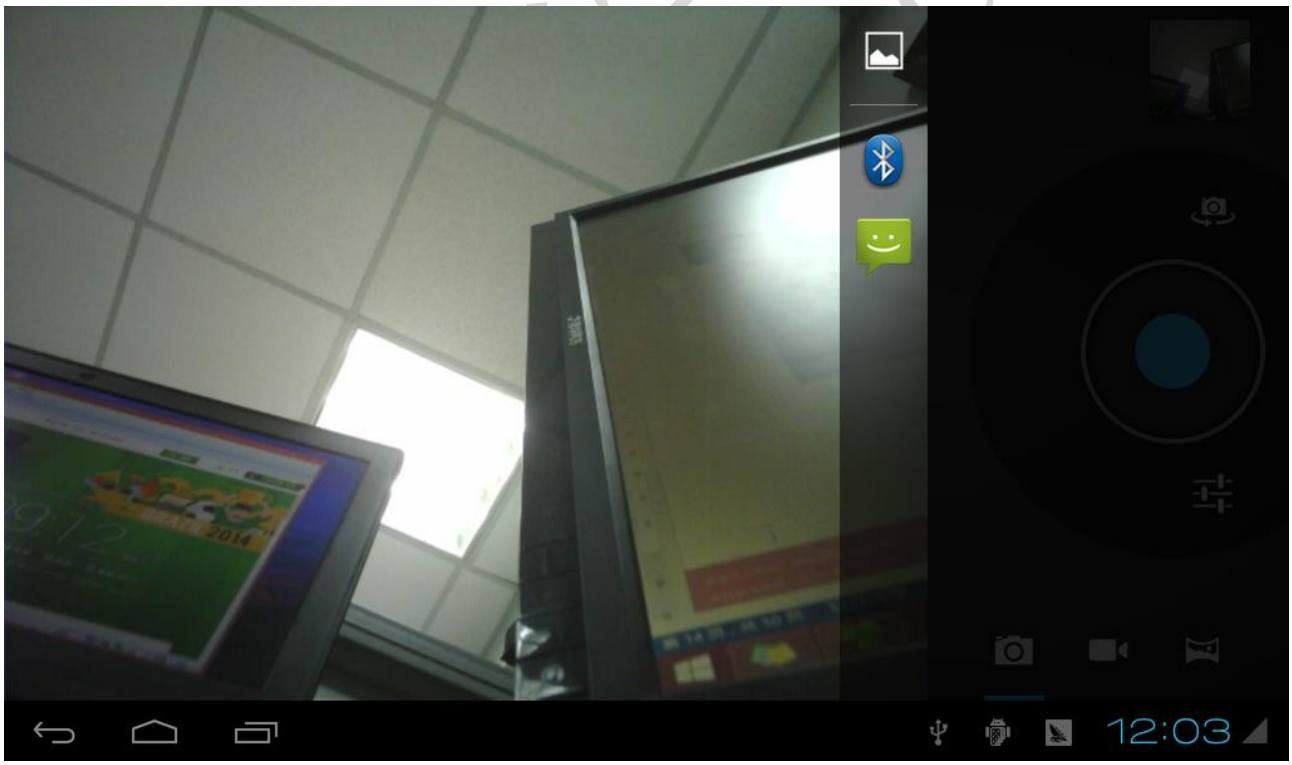
按照下图所示连接开发板和摄像头模块，开启开发板。



打开相机应用，可以看到摄像头预览。



可以进行拍照和摄像功能，可以在相册中查看我们拍到的内容。

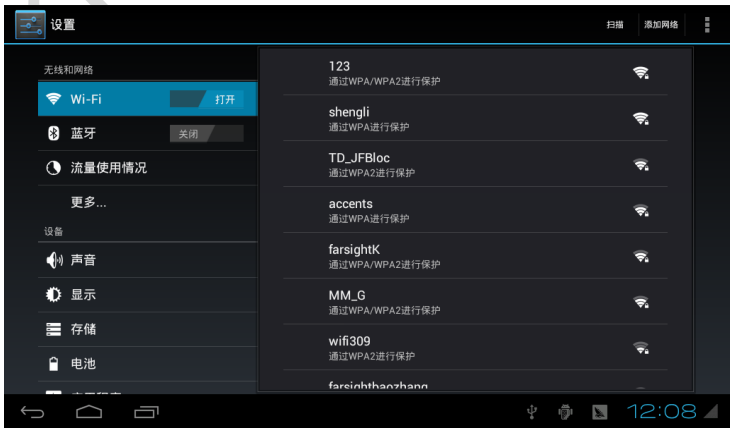


2.11 Wi-Fi 无线上网

将 USB WI-Fi 模块插入开发板的 USB 接口处。

打开 Android 下的系统设置。

可以查看到当前范围内的 Wi-Fi 热点和信号强度。



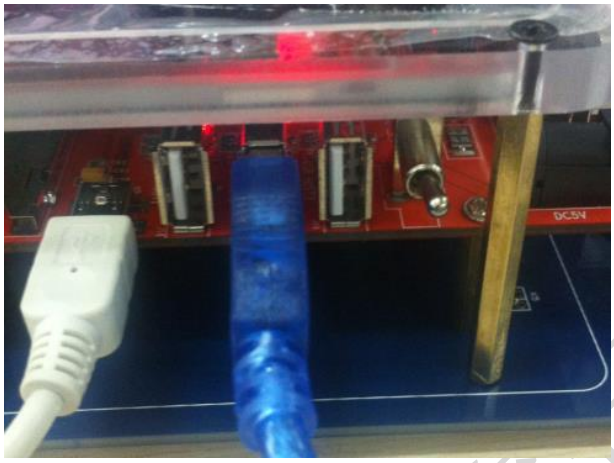
选择我们要连接的 Wi-Fi。

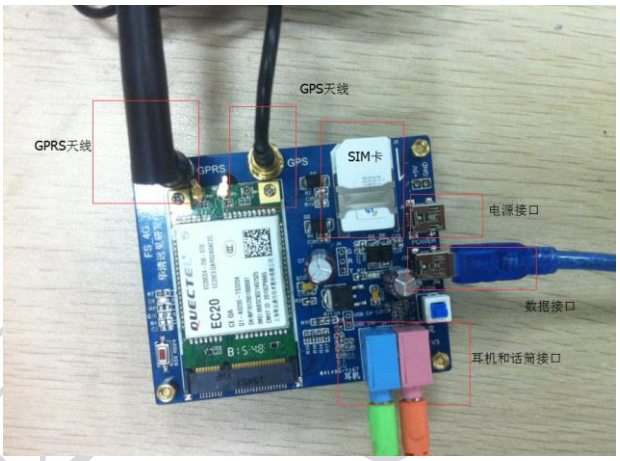
如果验证正确则可以看到【已连接】的字样。

打开浏览器，就可以访问互联网了。

2.12 4G 测试

系统出厂镜像默认为 4G 镜像，用户可以直接接入 4G 模块，若用户烧写过其他镜像，请重新烧写 4G 镜像，位置为【华清远见-CORTEXA9 资料:/烧写镜像/Adndroid 烧写镜像】，烧写方法参考“镜像烧写下的烧写系统章节”。

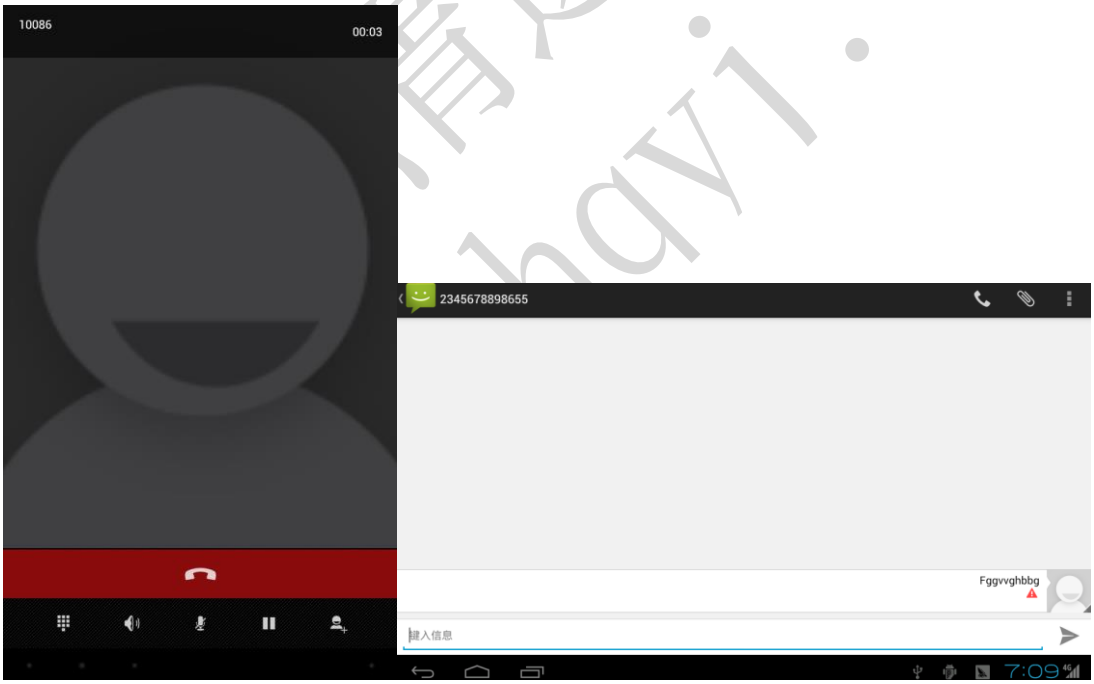




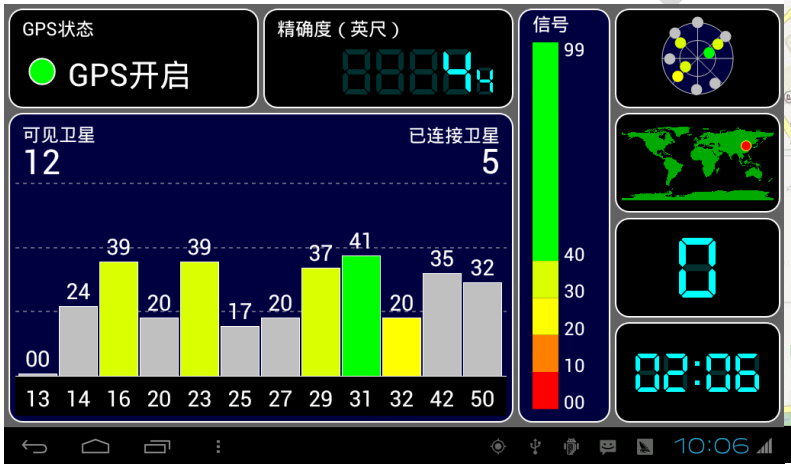
Android 启动之后插入 4G 模块，等待 4G 模块连接成功。可以看到信号状态。

尝试使用网络上网

尝试拨打电话和发送接收短信。



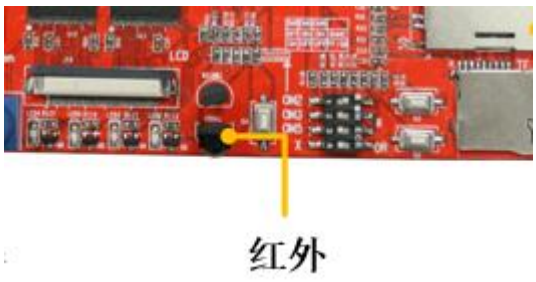
使用定位系统。



FS4412 开发平台不仅支持 GPS 定位系统，还支持北斗定位系统。

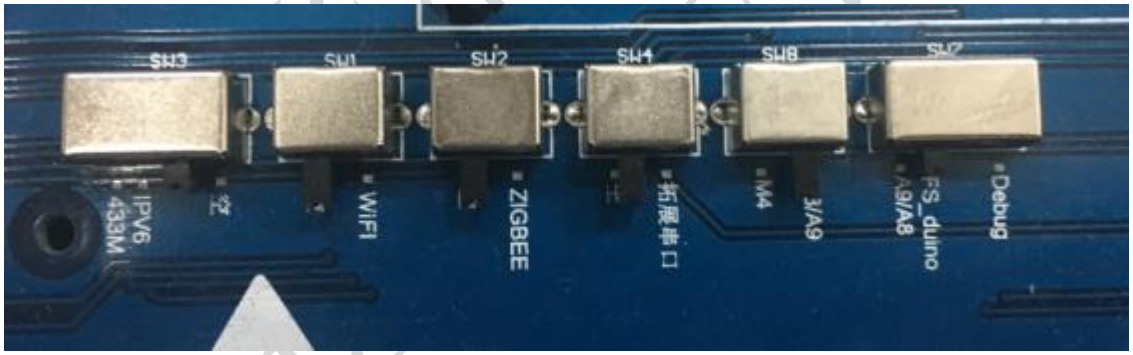
2.13 红外遥控器

FS4412 平台支持红外遥控器操作 Android，下图所示的遥控器已经完美支持。



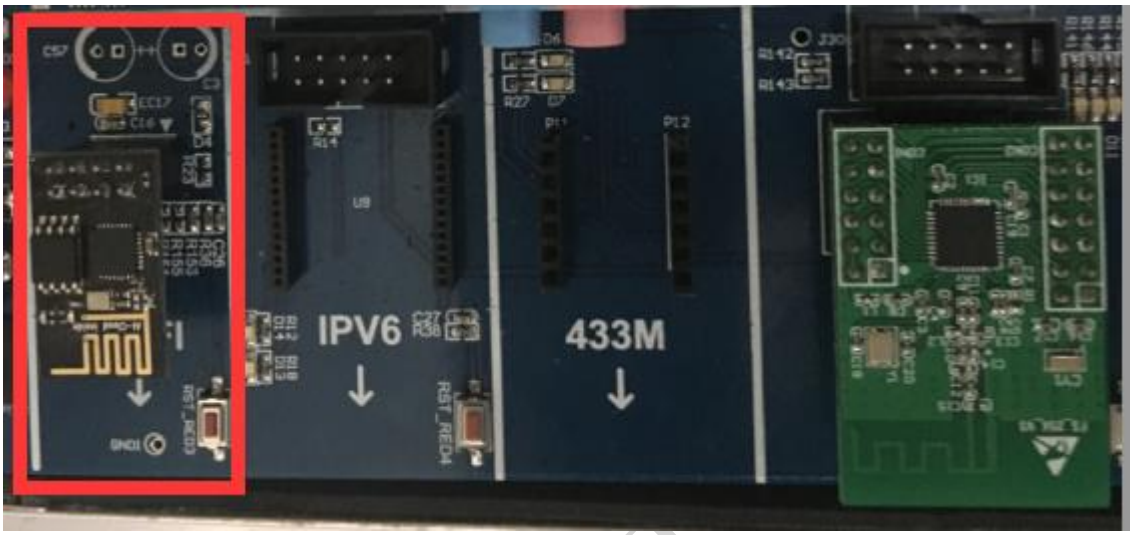
2.14 无线通讯实验

用户在使用该平台的串口时，需要保证横排拨码开关状态为（由左侧至右侧）SW3 拨空、SW1 拨空、SW2 拨空、SW4 拨空、SW8 拨 A8/A9、SW7（该拨码开关有两段，需特别注意）拨 A8/A9，完成后方可进行该章节的测试，如下图。



2.14.1 使用方法-WiFi 模式（选配）

将 WiFi 节点接入下图的位置，同时将其它的 WiFi 节点开关打开，用户需要观察 WiFi 节点背面贴有 PANID 的标签，该 PANID 我们稍后会用到。



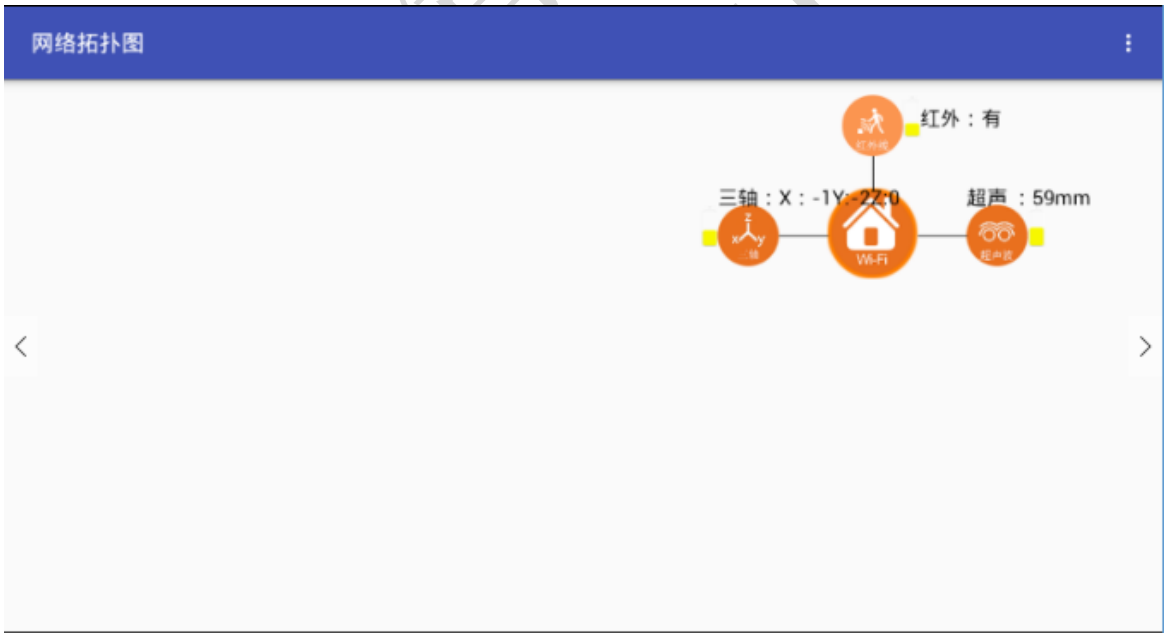
测试 WiFi 节点时，用户需将拨码开关中的 WIFI_R 和 WIFI_T 拨到 ON，其余拨码拨到 OFF 的位置，如下图。

完成后点击下图的网络拓扑图软件。

进入后，界面如下图，用户需要首先选择串口类型，然后输入从 WiFi 节点背面得到的 PANID，完成后，点击中间的 UART 图标。

进入之后，点击右上角的按钮，选择使用的串口，在该平台中使用的串口为 ttySAC3.

选择完成后，等待几秒钟，即可看到 WiFi 节点的数据，下图的 WiFi 节点仅供参考，用户的节点类型不完全一致。



节点上方显示的文字含义：节点地址+节点类型+节点状态（或采集得到的数值）。

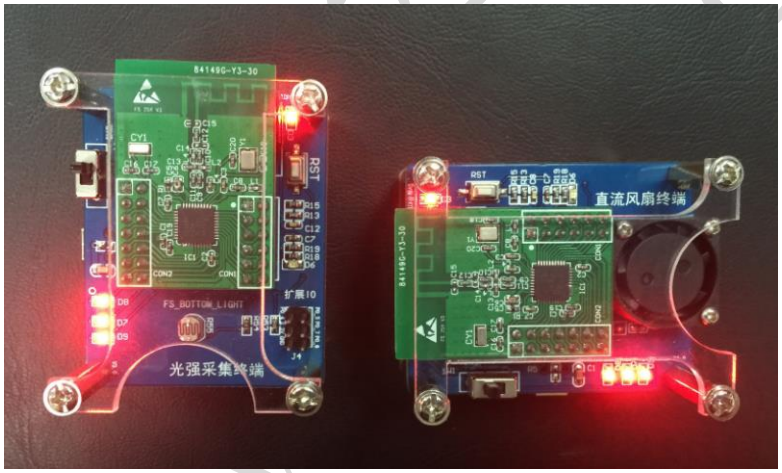
屏幕上除根节点外其他任何子节点都可以拖动以变换位置。

当数据更新的时候，子节点会通过闪烁来提示。

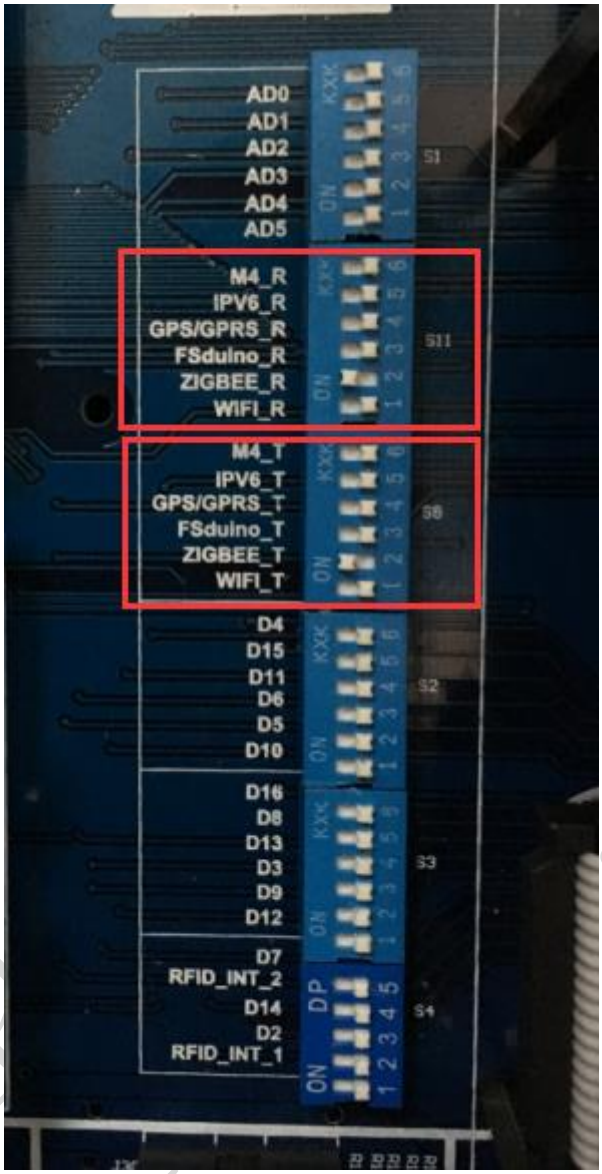
2.14.2 使用方法-UART 模式

将 ZigBee 协调器节点插到串口处，当这三个灯的第二个在闪烁的时候说明已经连接上了；

打开各个模块节点的开关；



测试 ZigBee 的时候，拨动拨码开关，将 ZigBee_R 和 ZigBee_T 拨到左边，其余拨到右边，如下图所示：


AD0
AD1
AD2
AD3
AD4
AD5
S1
M4_R
IPV6_R
GPS/GPRS_R
FSduino_R
ZIGBEE_R
WIFI_R
S11
M4_T
IPV6_T
GPS/GPRS_T
FSduino_T
ZIGBEE_T
WIFI_T
S8
D4
D15
D11
D6
D5
D10
S2
D16
D8
D13
D3
D9
D12
S3
D7
RFID_INT_2
D14
D2
RFID_INT_1
S4
ON
DP

打开图中所示软件：

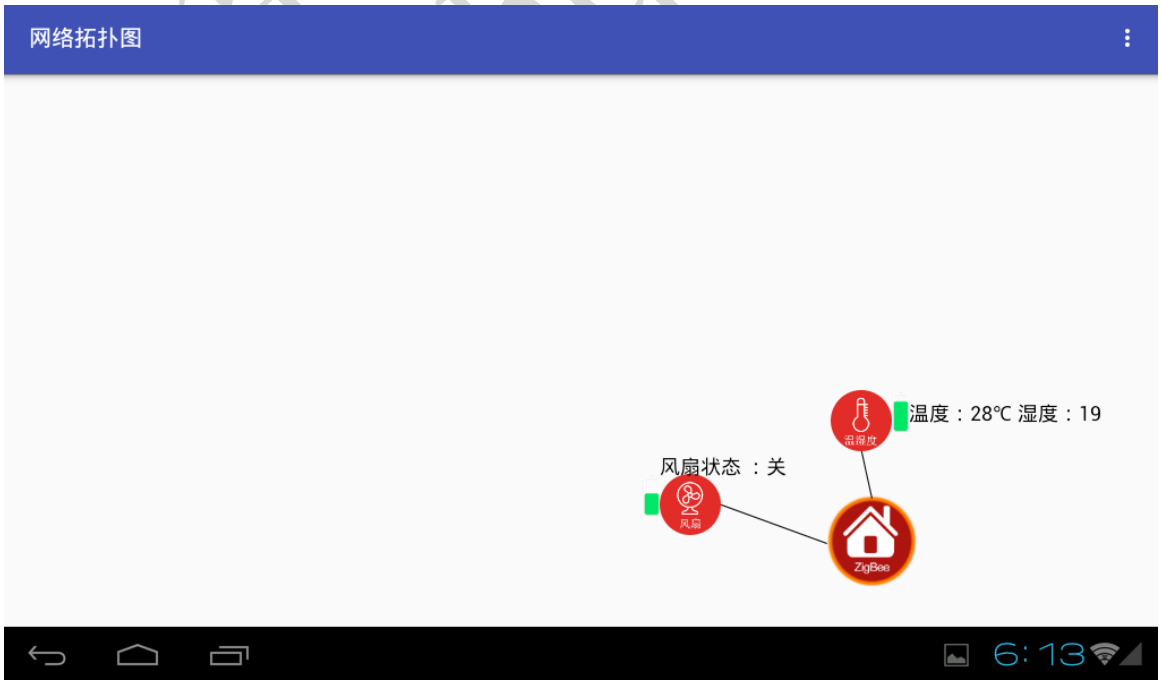
点击图标后进入程序，以下是主界面，根据你连接的设备，进行相应的选择：根据使用串口或者是 USB 口，选择相应的接口，并输入 PANID（PANID 是使用单独的 wifi 模块时，是来进行透传模式的开启，用来设置 wifi 名称的一个序号），然后点击中间的 UART 图片进入下一个界面。在这里选择串口：

点击红里面的图标，进入下一个界面之后，根据所连接具体哪一个接口进行选择：

选择错误的时候会提示错误，提示用户更改接口，依次打开实验相中各个节点的开关等待几秒钟后，程序界面会显示相应的节点信息。

节点上方显示的文字含义：节点类型+节点状态（或采集得到的数值）。

屏幕上除根节点外其他任何子节点都可以拖动以变换位置，点击根节点用来复位。



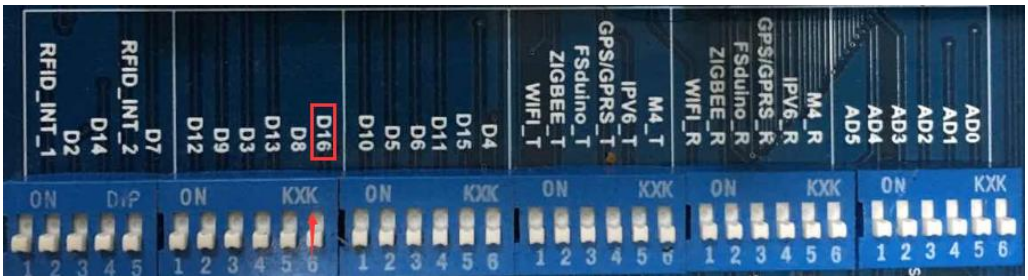
当数据更新的时候，子节点会通过闪烁来提示。

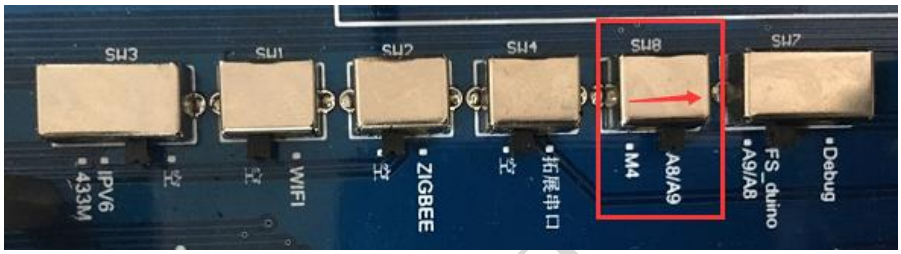
当风扇节点状态显示‘关’的时候，此时风扇处于关闭，点击节点，风扇会打开；当风扇节点状态显示‘开’的时候，此时风扇处于打开状态，点击节点，风扇会关闭。

2.15 arduino 板测试

点击如图所示的图标进入程序

下面是拨码开关，在使用时，先要拨动拨码开关使与之对应。





2.15.1 酒精传感器

将拨码开关中的 AD1 拨至 ON，其他拨码全为 OFF，如下图

当周围环境酒精浓度越大，值越大，结果如上图所示。

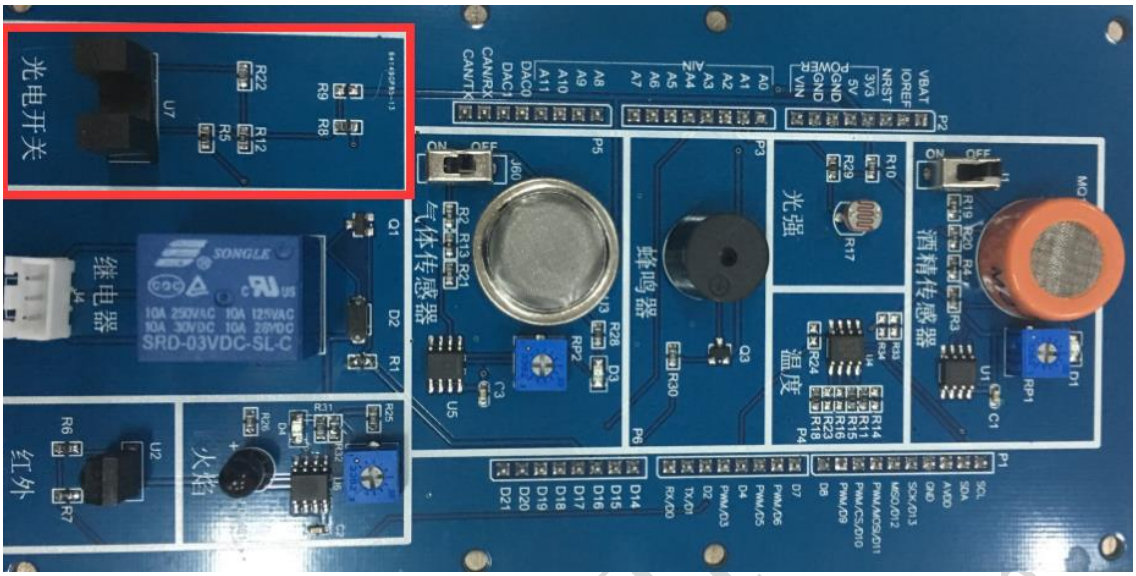
酒精传感器在模块中的位置如下图所示：

2.15.2 光电闸

将拨码开关中的 D14 拨至 ON，其他拨码全为 OFF，如下图

可以用不透明的东西挡住进行测试，结果如上图所示。

光电闸在模块中的位置如下图所示：

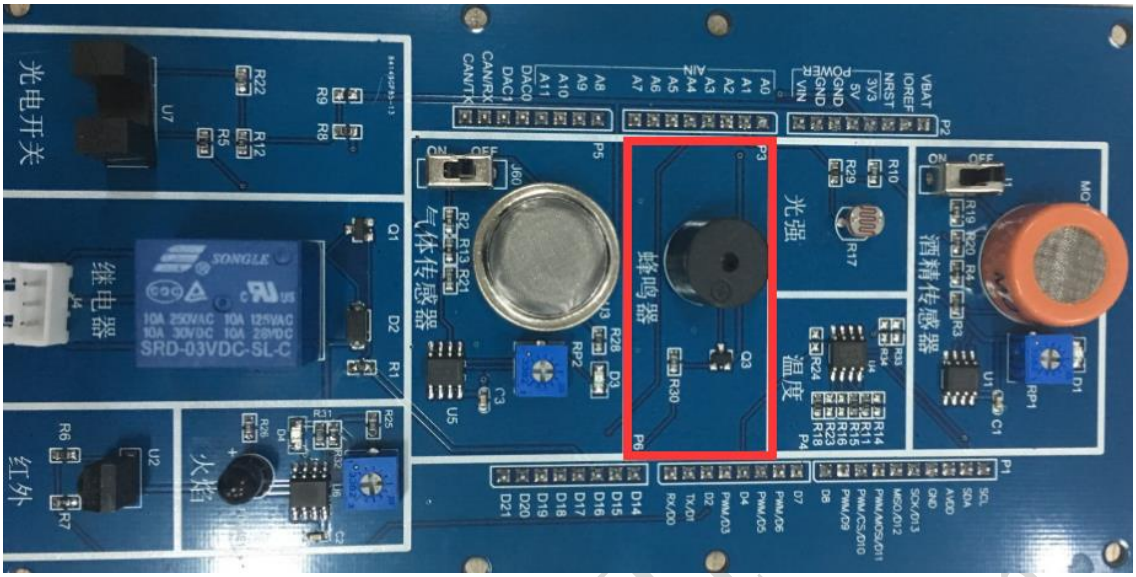


2.15.3 蜂鸣器

将拨码开关中的 D15 拨至 ON，其他拨码全为 OFF，如下图

点击上面的“打开蜂鸣器”和“关闭蜂鸣器”分别打开和关闭蜂鸣器，打开的时有响声。

蜂鸣器在模块中的位置如下图所示：

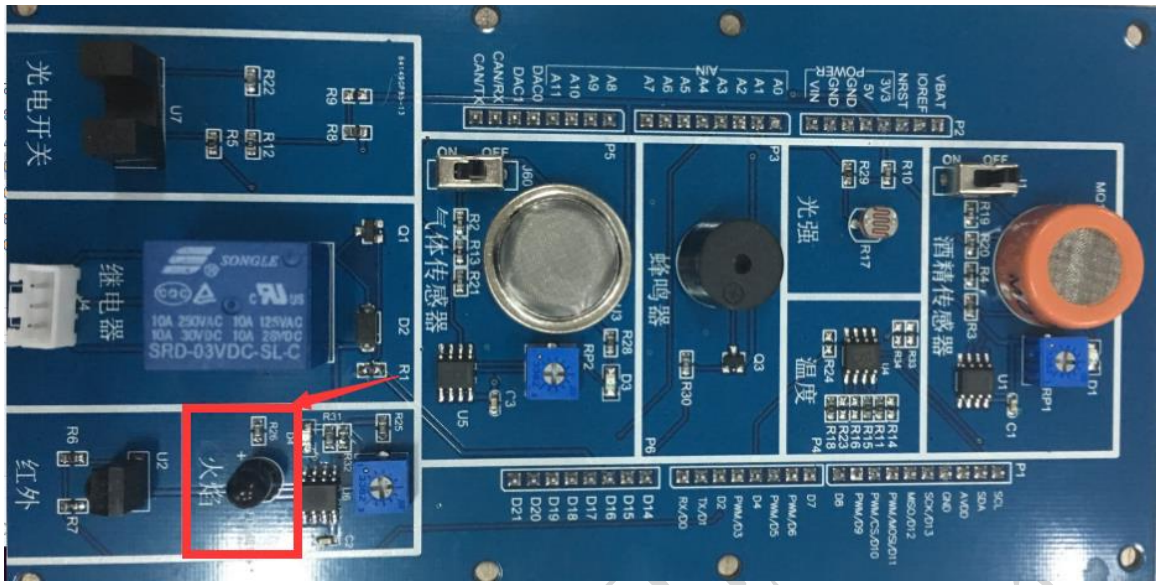


2.15.4 火焰

将拨码开关中的 AD3 拨至 ON，其他拨码全为 OFF，如下图

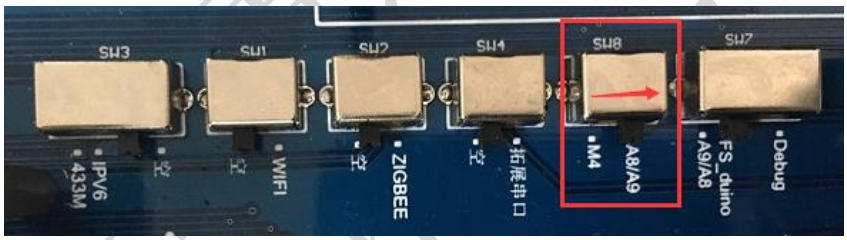
通过打火机或其他明火在火焰传感器上方照射，可以在软件中获取到传感器采集的数据。

火焰传感器在模块中的位置如下图所示：



2.15.5 直流电机

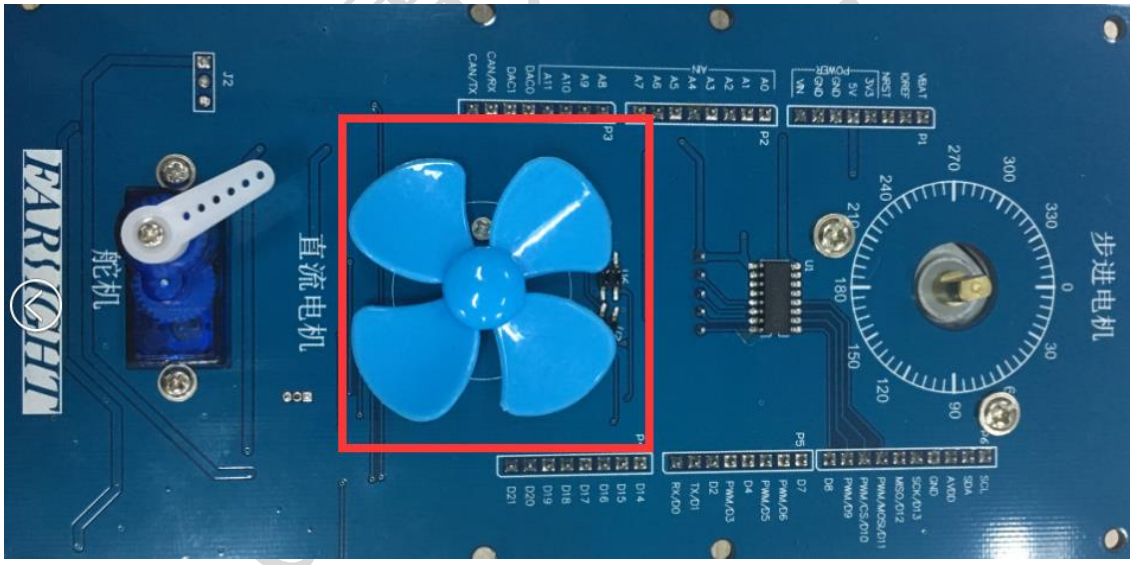
将 SW8 拨至 A8/A9 处，如下图：



将拨码开关中的 D12、D13 拨至 ON，其他拨码全为 OFF，如下图

这里有三个按钮，左边的按钮是让步进电机反转，中间是停止按钮，右边是正转按钮。

直流电机在模块中的位置如下图所示：

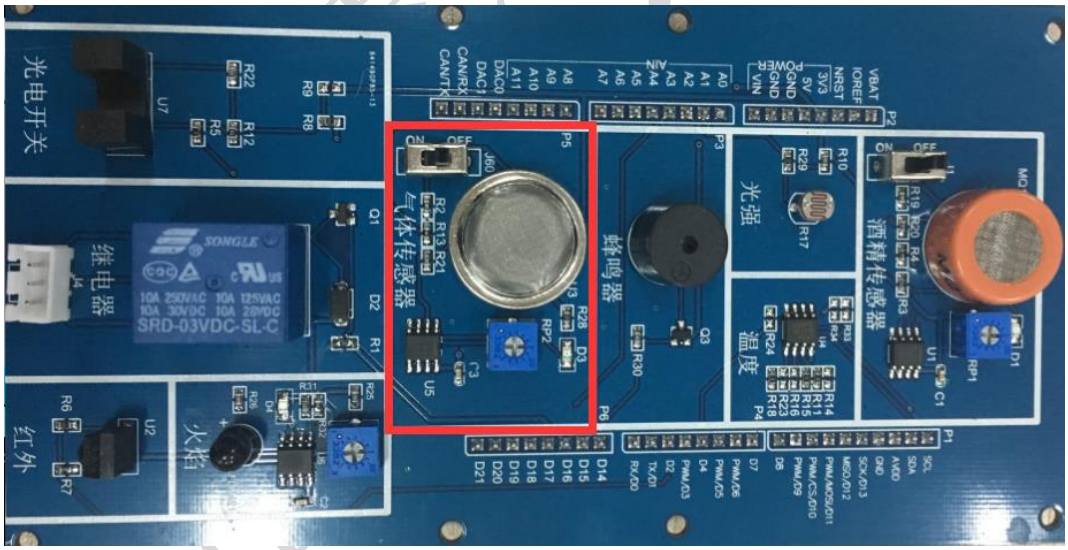


2.15.6 可燃气体传感器

将拨码开关中的 AD4 拨至 ON，其他拨码全为 OFF，如下图

用打火机的气体测试，当气体浓度越大，值越大，结果如上图所示。

可燃气传感器在模块中的位置如下图所示：

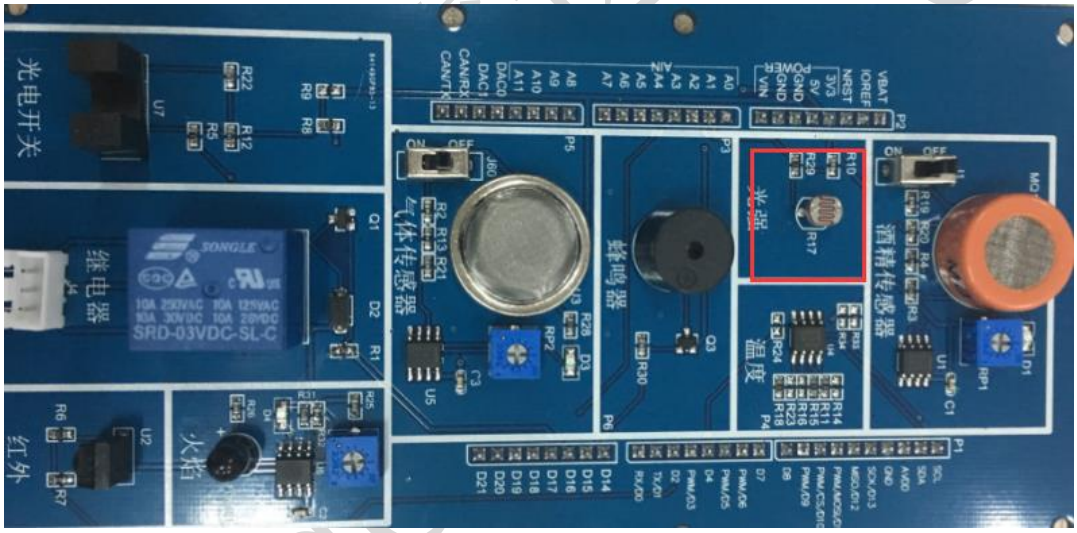


2.15.7 光敏传感器

将拨码开关中的 AD2 拨至 ON，其他拨码全为 OFF，如下图

周围关照强度越高，显示数字越大，如上图所示。

光照传感器在模块中的位置如下图所示：

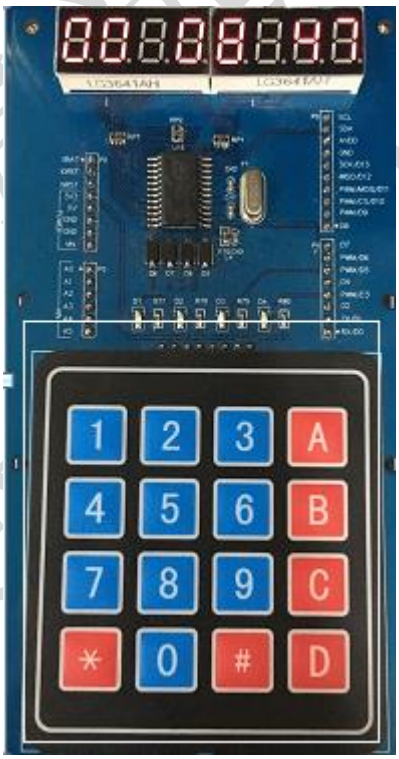


2.15.8 矩阵键盘

该实验只需要注意 SW8 拨码拨至 A8/A9，因为在硬件设计时，将 SW8 作为 MCU 和 CPU 的 i2c 主机切换开关，如下图

在键盘上输入的数字会显示在屏幕上面，如上图所示“等待输入。。。”

矩阵键盘在模块中的位置如下图所示：

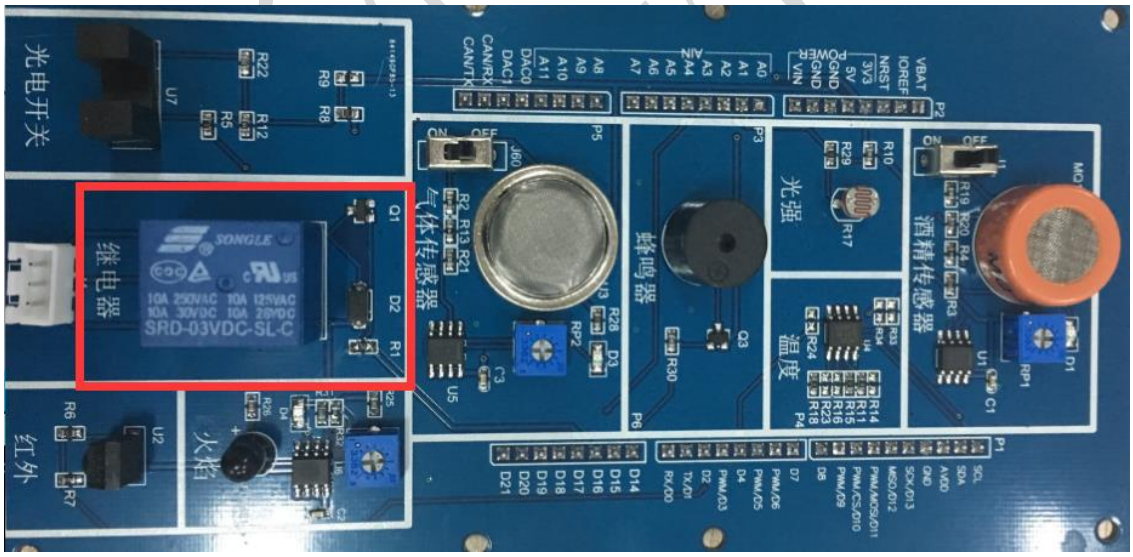


2.15.9 继电器

将拨码开关中的 D16 拨至 ON，其他拨码全为 OFF，如下图

如图所示两个按钮“继电器开”和“继电器关”分别用来对继电器的“开关”进行操作。继电器打开的时候回响一声。

继电器在模块中的位置如下图所示：

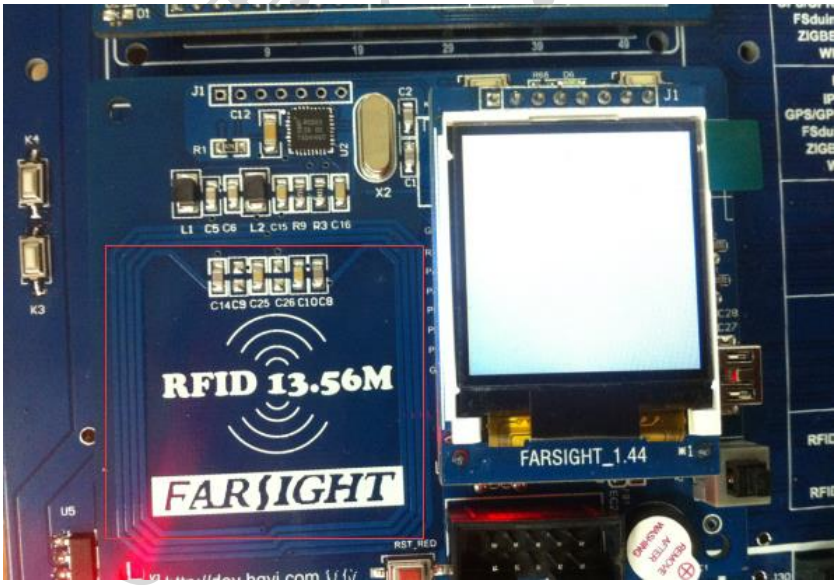


2.15.10 RFID 模块

该实验只需要注意 SW8 拨码拨至 A8/A9，因为在硬件设计时，将 SW8 作为 MCU 和 CPU 的 i2c 主机切换开关，如下图

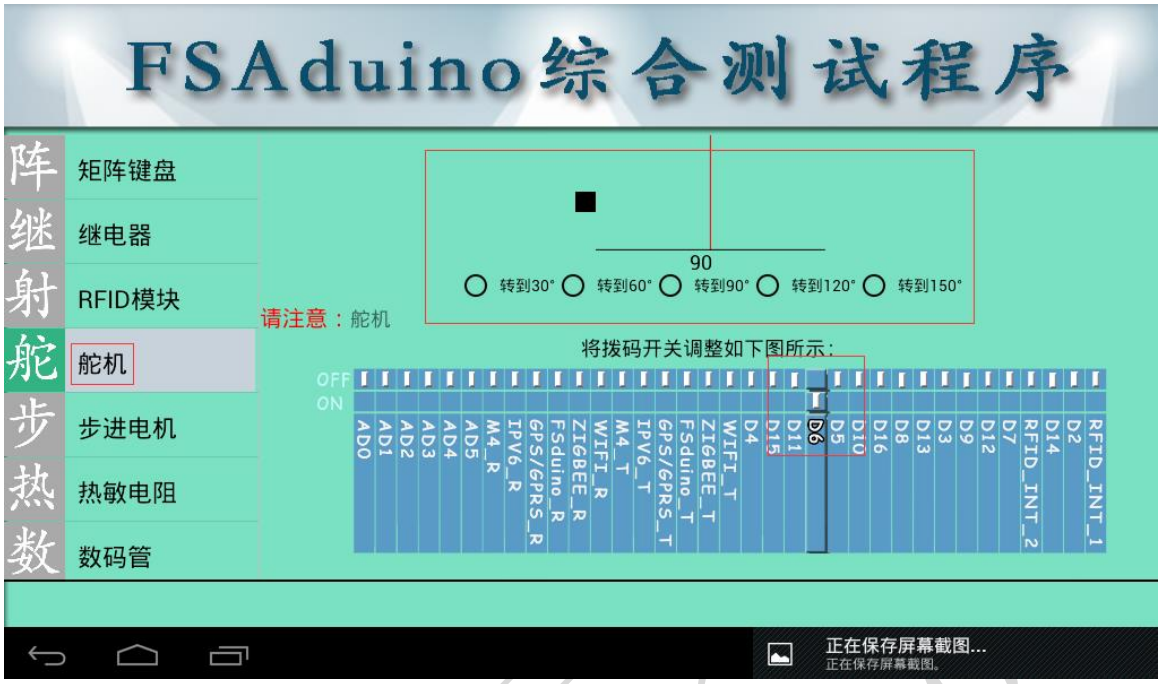
RFID 模块读到数据之后会显示在上图所示的框中。上面显示的是“00 00 00 00”

RFID 在模块中的位置如下图所示：



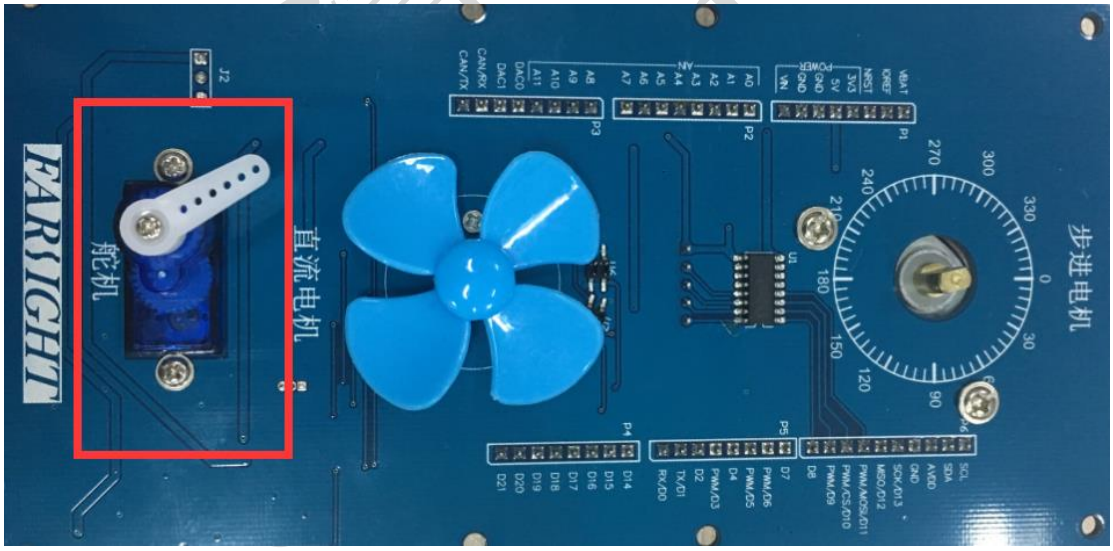
2.15.11舵机

将拨码开关中的 D6 拨至 ON，其他拨码全为 OFF，如下图



如图所示上面有旋转多少度的按钮，点击之后可以看到舵机旋转的位置。

舵机在模块中的位置如下图所示：

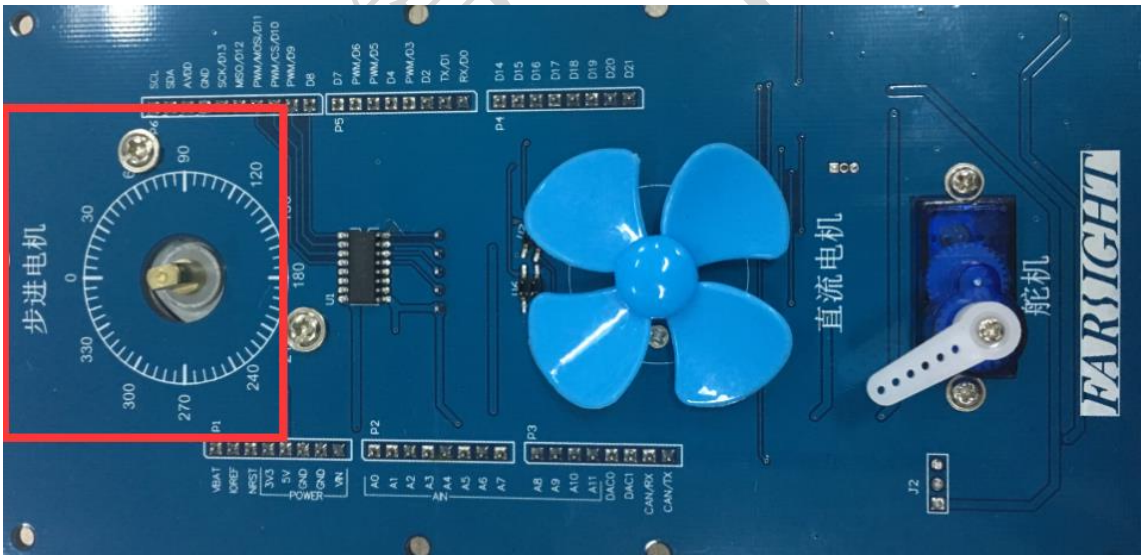


2.15.12步进电机

将拨码开关中的 D8、D9、D10、D11 拨至 ON，其他拨码全为 OFF，如下图

步进电机转分为 9 档，依次加快，可以看到步进电机在旋转。

步进电机在模块中的位置如下图所示：

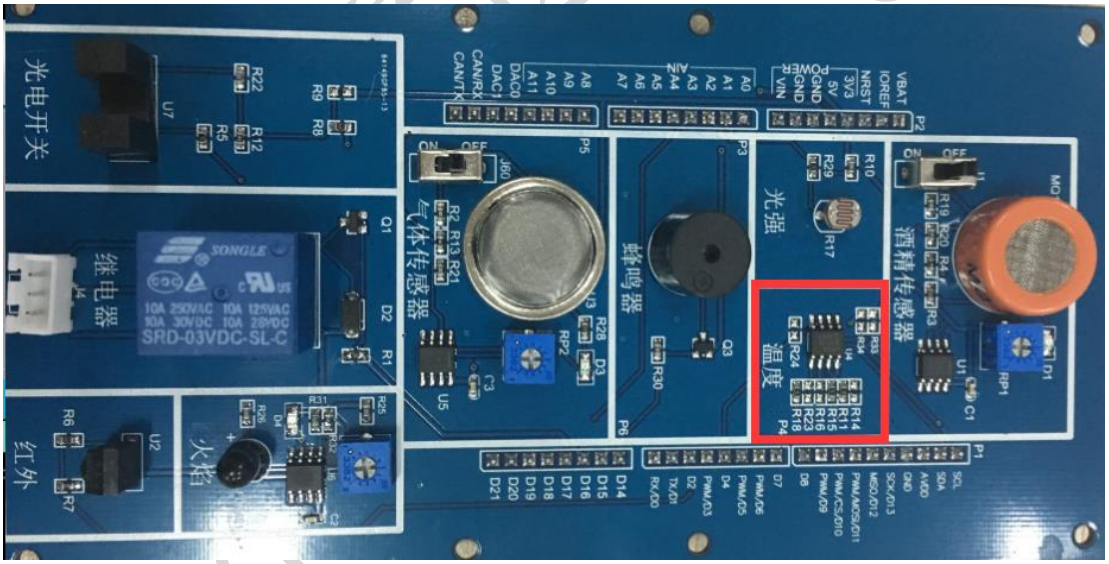


2.15.13温度传感器

该实验只需要注意 SW8 拨码拨至 A8/A9，因为在硬件设计时，将 SW8 作为 MCU 和 CPU 的 i2c 主机切换开关，如下图

可以用手或者和室温有差别的，触摸一会，温度传感器会采集当前的温度值。

热敏电阻在模块中的位置如下图所示：



2.15.14数码管

该实验只需要注意 SW8 拨码拨至 A8/A9，因为在硬件设计时，将 SW8 作为 MCU 和 CPU 的 i2c 主机切换开关，如下图

点击键盘，在数码管里面会显示出点击的数字，并且会依次向后排列。

数码管在模块中的位置如下图所示：

第 3 章　Linux 下功能测试

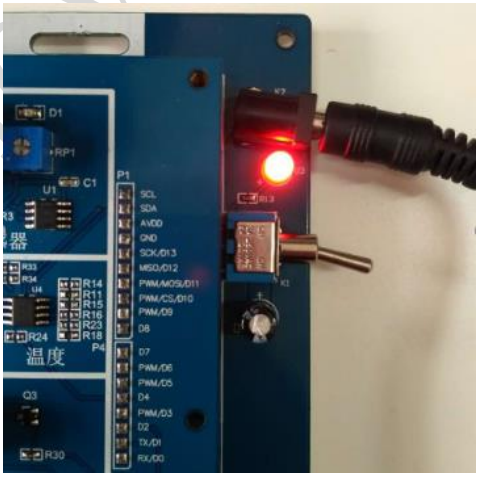
本章节的测试环境为 2010.03 版本 uboot ,linux3.0 内核。要实现本章节的测试 ,需要用户自行烧写 Linux 系统。（请参考：嵌入式 Linux 移植及应用驱动的第 3 章节，交叉开发环境搭建）

3.1　烧写 linux 测试系统

镜像目录位于【华清远见-CORTEXA9 资料:/烧写镜像/Linux3.0 烧写镜像】这里将 Linux3.0 烧写镜像目录下的文件拷贝到存放目录【E:\ ShareVMare】（目录用户可以根据自己的环境，自定义，不推荐使用中文路径）目录下

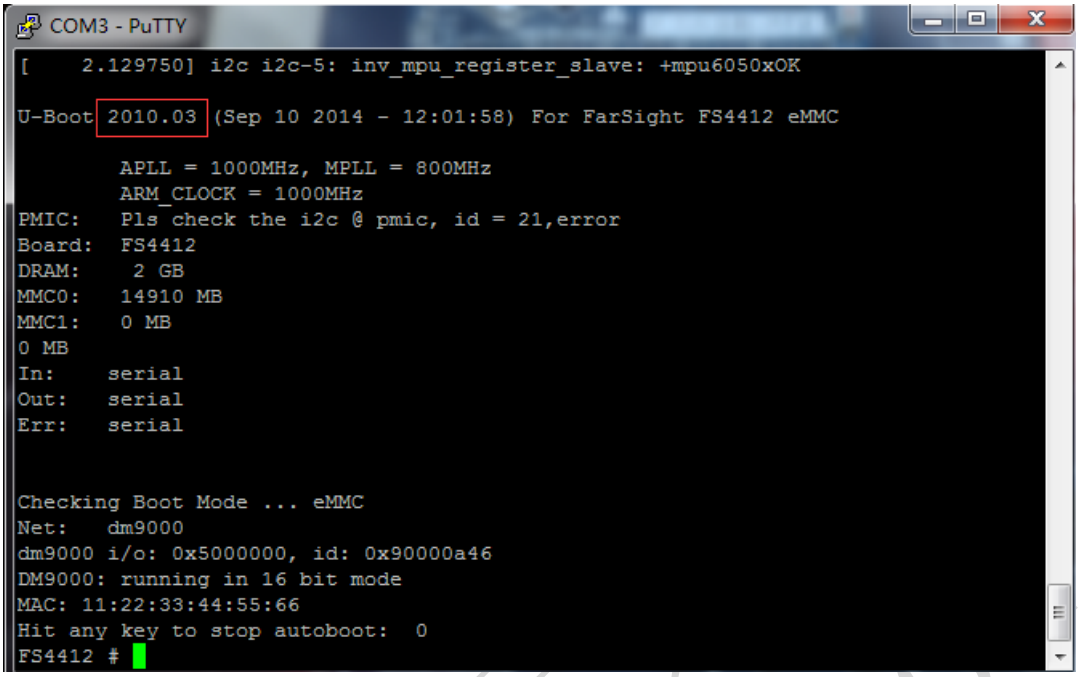
名称	修改日期	类型	大小
system.img	2016/5/24 9:29	光盘映像文件	65,165 KB
u-boot-fs4412.bin	2016/5/23 19:35	BIN 文件	515 KB
zImage	2016/5/23 19:34	文件	3,694 KB

准备开始进行烧写，烧写过程中开发板的接线如下图：



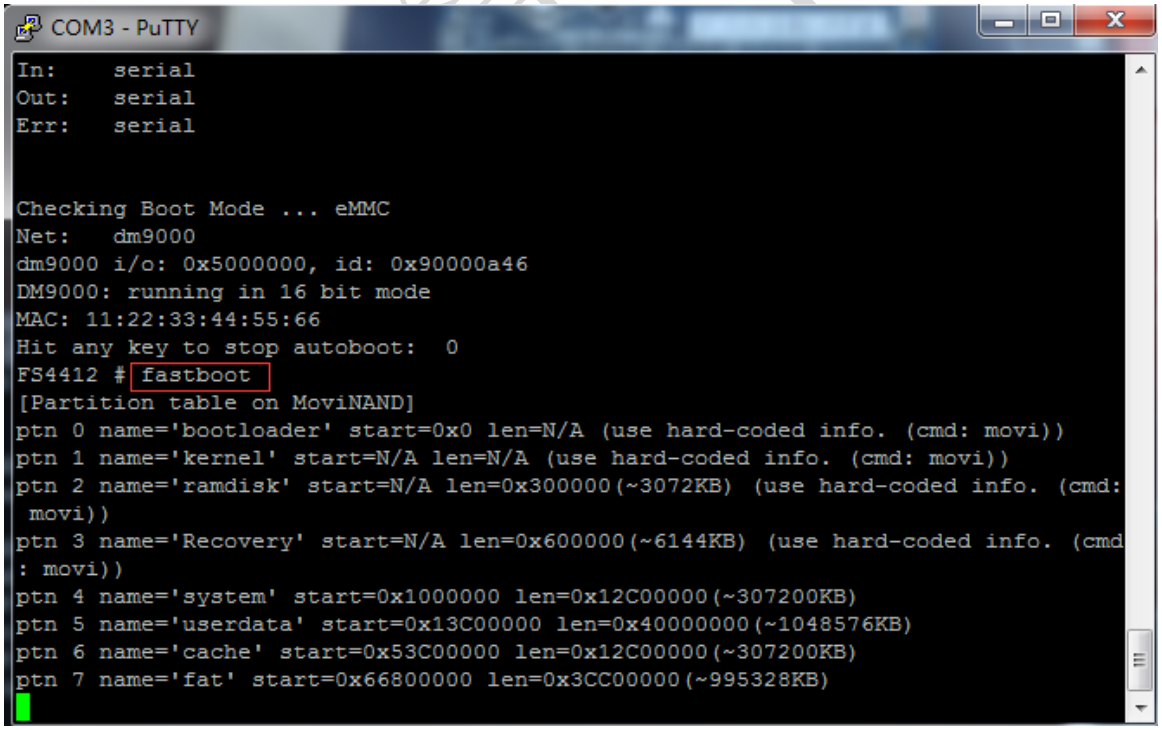
连线接好后，启动串口调试助手 putty 并对开发板进行上电，如何设置串口调试助手见”镜像烧写”章节的设置调试串口助手。

启动板子，putty 在倒数计时的过程中，按任意键停止；

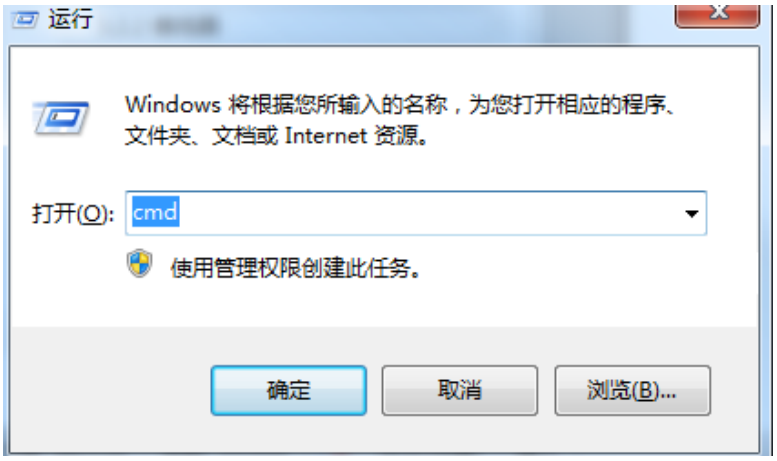


上图框内是当前开发板的 uboot 版本

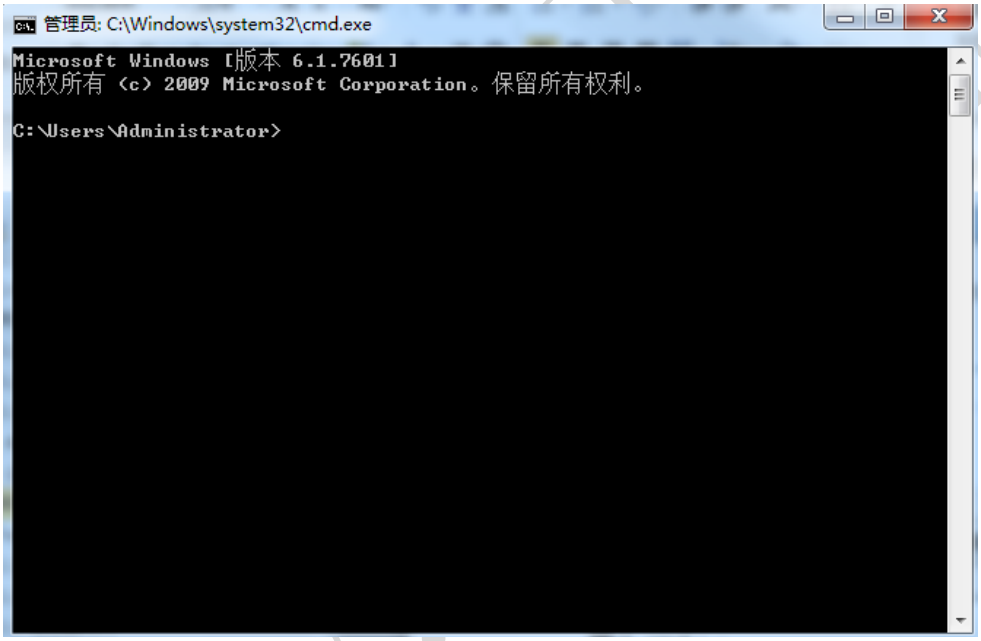
然后输入 fastboot，会出现以下现象：



在 window 系统下面进行烧写，点击 window+R：



输入 cmd：



进入烧写镜像的文件夹里面进行烧写，这里放在【E:\ ShareVMware】件夹下：

输入 E：+回车

输入 cd ShareVMare+回车

在 fastboot 环境搭建好的情况下进行如下操作，fastboot 环境搭建请参考:【第 5 章/5.4 环境配置】

烧写 uboot

```
Fastboot flash bootloader u-boot-fs4412.bin
```

```
E:\ShareVMware>fastboot flash bootloader u-boot-fs4412.bin
sending 'bootloader' (514 KB)...
OKAY [  0.072s]
writing 'bootloader'...
OKAY [  0.179s]
finished. total time: 0.253s
```

烧写 zImage

```
Fastboot flash kernel zImage
```

```
E:\ShareVMware>fastboot flash kernel zImage
sending 'kernel' (3693 KB)...
OKAY [  0.449s]
writing 'kernel'...
OKAY [  0.267s]
finished. total time: 0.719s
```

烧写文件系统

```
Fastboot flash system system.img
```

```
E:\ShareVMware>fastboot flash system system.img
sending 'system' (65164 KB)...
OKAY [  7.738s]
writing 'system'...
OKAY [  9.748s]
finished. total time: 17.487s
```

到这里为止 linux 系统所需要的镜像全部烧写完毕，重启开发板并在倒数计时的时候按任意键，并使用在 putty 里面输入【print】命令查看当前的环境变量。

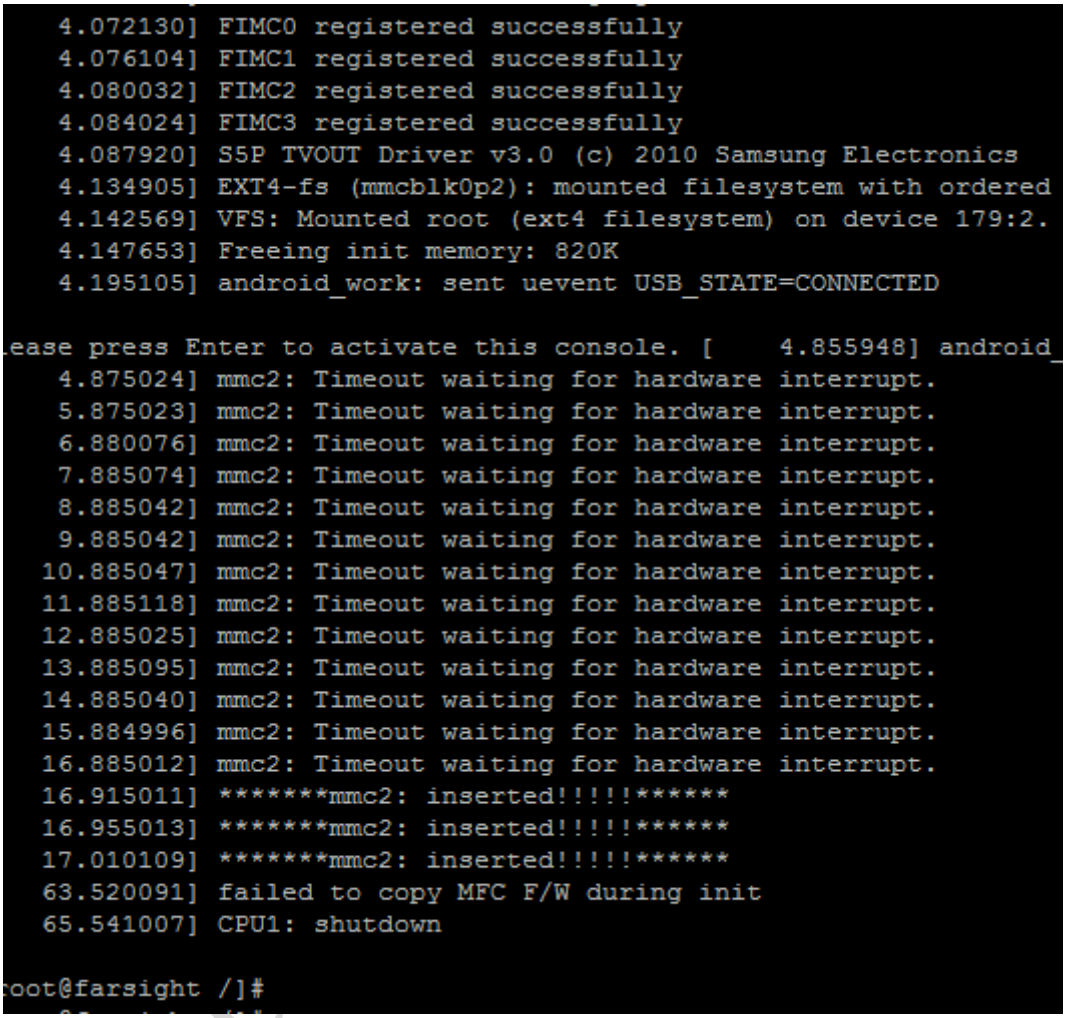
```
FS4412 # print
bootdelay=5
baudrate=115200
ethaddr=11:22:33:44:55:66
ethact=dm9000
ipaddr=192.168.1.192
serverip=192.168.1.133
gatewayip=192.168.1.1
bootargs=root=/dev/mmcblk0p2 rootfstype=ext4 init=linuxrc console=ttySAC2,115200
bootcmd=movi read kernel 40008000;bootm 40008000
stdin=serial
stdout=serial
stderr=serial

Environment size: 305/16380 bytes
FS4412 #
```

我们需要设置 bootcmd 和 bootargs 参数，命令如下：(该命令为一行命令，每一个单词数据之间均有

空格）

```
setenv bootargs root=/dev/mmcblk0p2 rootfstype=ext4 init=/linuxrc console=ttySAC2,115200


setenv bootcmd movi read kernel 40008000\;bootm 40008000
save
```

重新启动开发板：



3.2 外部模块功能测试

3.2.1 直流电机

```
cd /fs4412_arduino_driver
insmod  fs4412_dynamo.ko
mknod /dev/dynamo c 800 3
./DC_dynamo
```

将拨码开关中的 D12、D13 拨至 ON，其他拨码全为 OFF，如下图

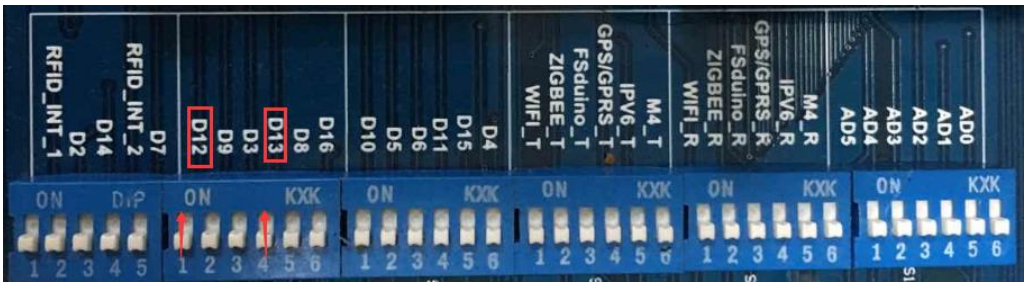


图 拨码开关示意图

如果以上步骤正确完成，可以看到直流电机每隔数秒变换一次转向，如果想要停止该程序，在 Putty 中使用 Ctrl + c 键即可。

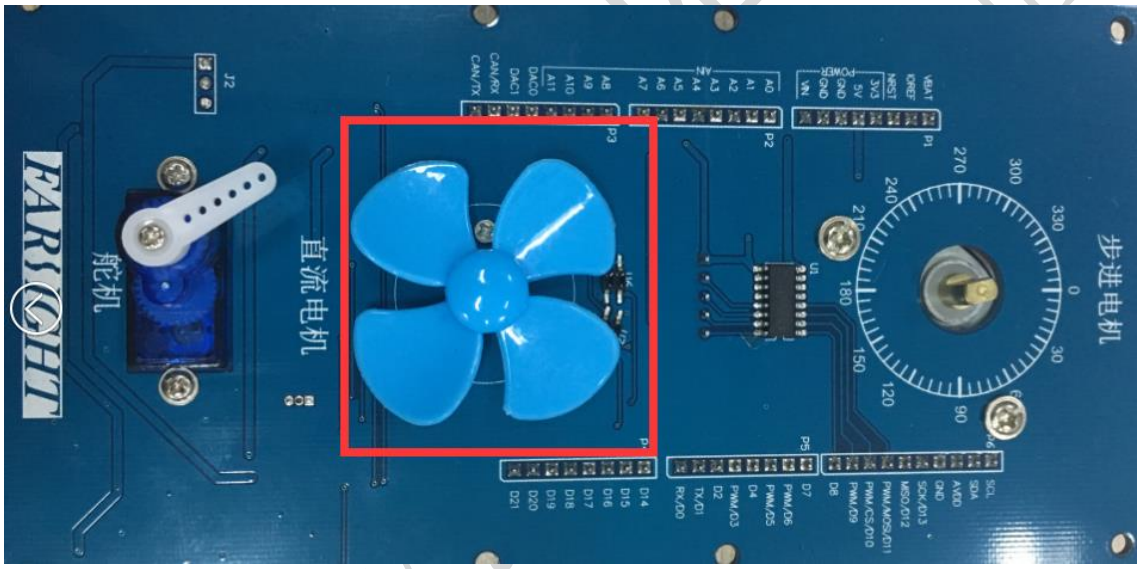


图 实验现象图

试验结束后需卸载模块：

```
lsmod
rmmod fs4412_dynamo
```

3.2.2 步进电机

```
cd /fs4412_arduino_driver
insmod  fs4412-stepper.ko
mknod /dev/stepper c 800 2
./stepper
```

将拨码开关中的 D8、D9、D10、D11 拨至 ON，其他拨码全为 OFF，如下图

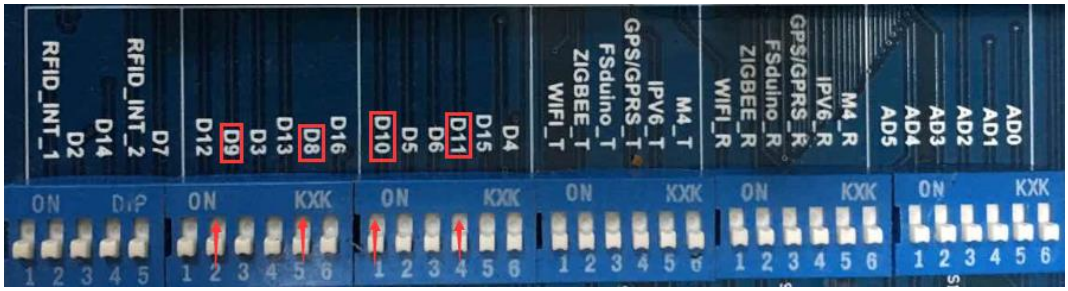


图 拨码开关示意图

如果正确按照上面的步骤操作完成，会看到步进电机开始转动：

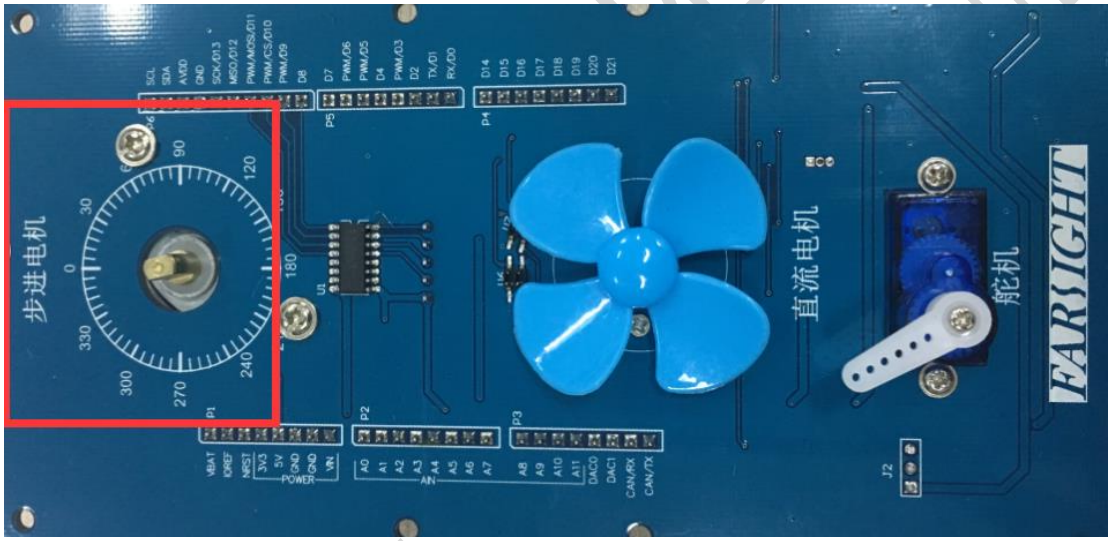


图 实验现象图

试验结束后需卸载模块：

```
lsmod
rmmod fs4412_stepper
```

3.2.3 舵机驱动

```
cd /fs4412_arduino_driver
insmod  fs4412_servo.ko
mknod /dev/servo c 501 0
./servo
```

将拨码开关中的 D6 拨至 ON，其他拨码全为 OFF，如下图

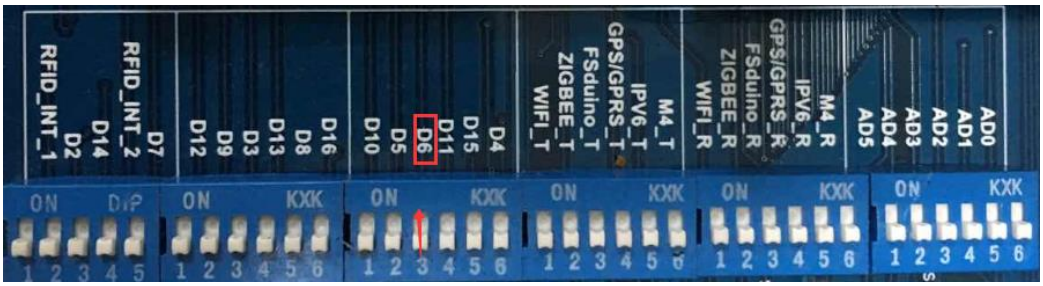


图 拨码开关示意图

如果按照上面的步骤操作正确的话，串口输入 0-180 度分为的角度，可以看到舵机转动。

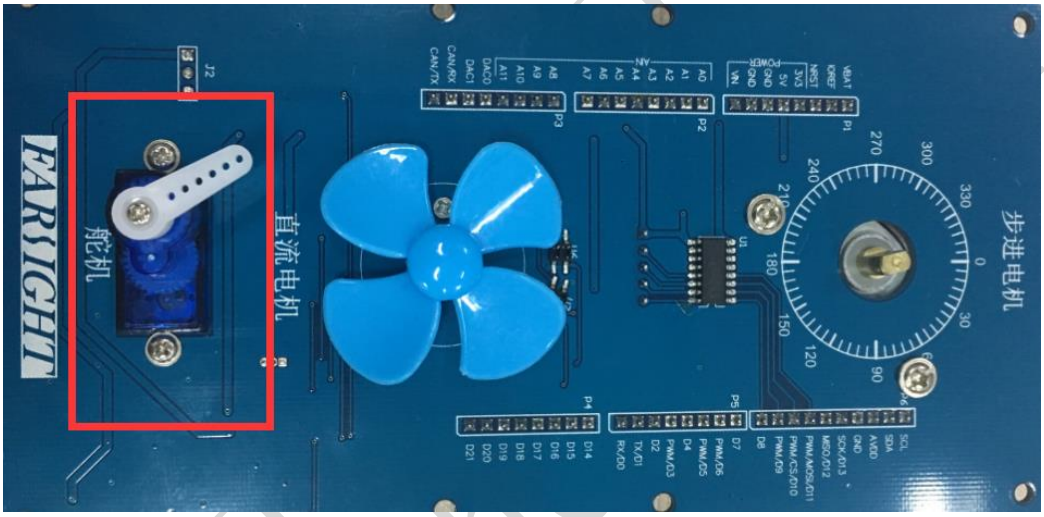


图 实验现象图

试验结束后需卸载模块：

```
lsmod
rmmod fs4412_servo
```

3.2.4 继电器驱动

```
cd /fs4412_arduino_driver
insmod  fs4412_gpio.ko
mknod /dev/gpio c 800 6
./gpio
```

将拨码开关中的 D16 拨至 ON，其他拨码全为 OFF，如下图

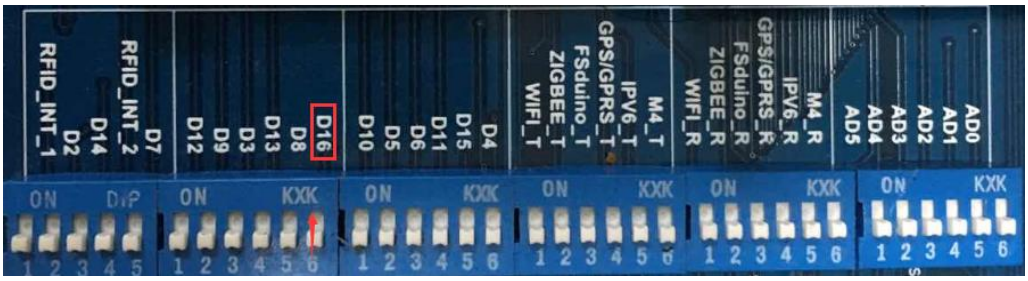
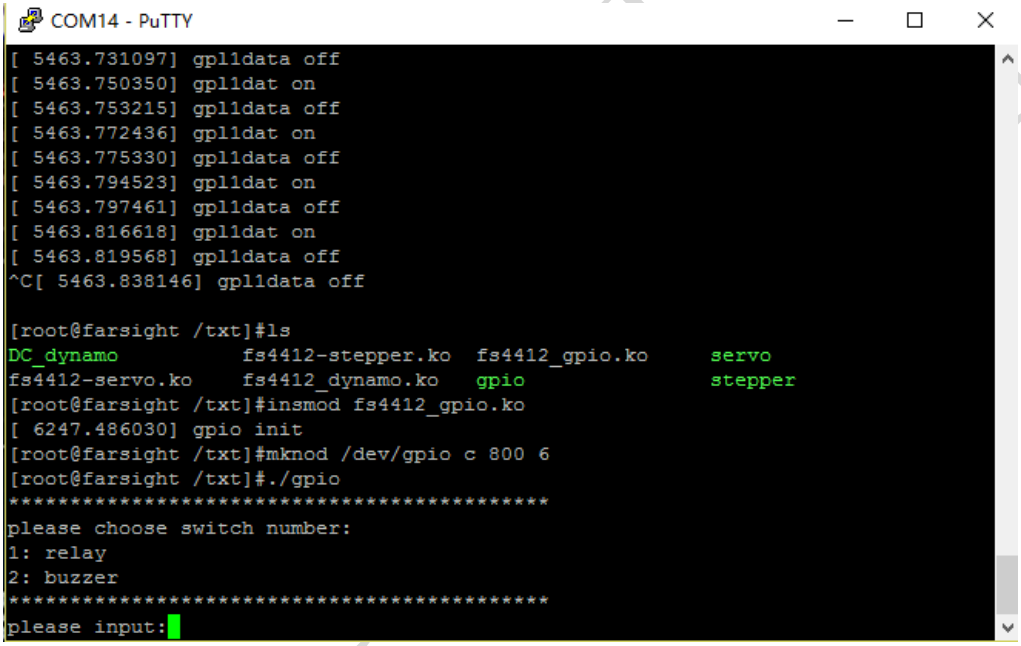
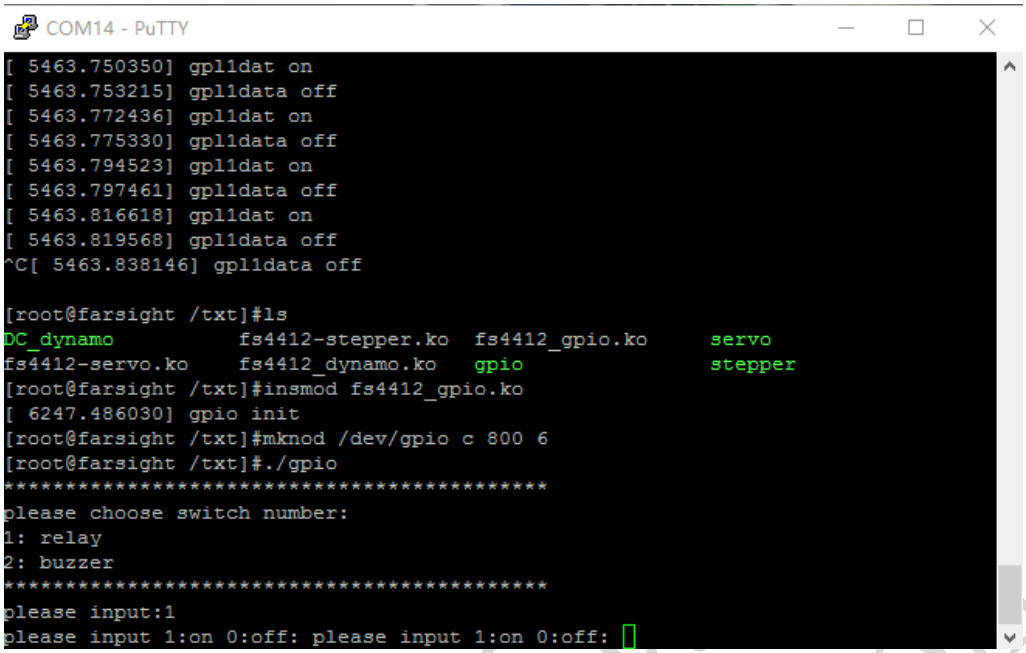


图 拨码开关示意图

如果按照上面的步骤正确操作，在 Putty 上会出现如下界面：



这里测试的是继电器，所以输入 1，可以看到如下提示：



输入 1 是打开，输入 0 是闭合，在这个过程中可以听到继电器发出的声音。

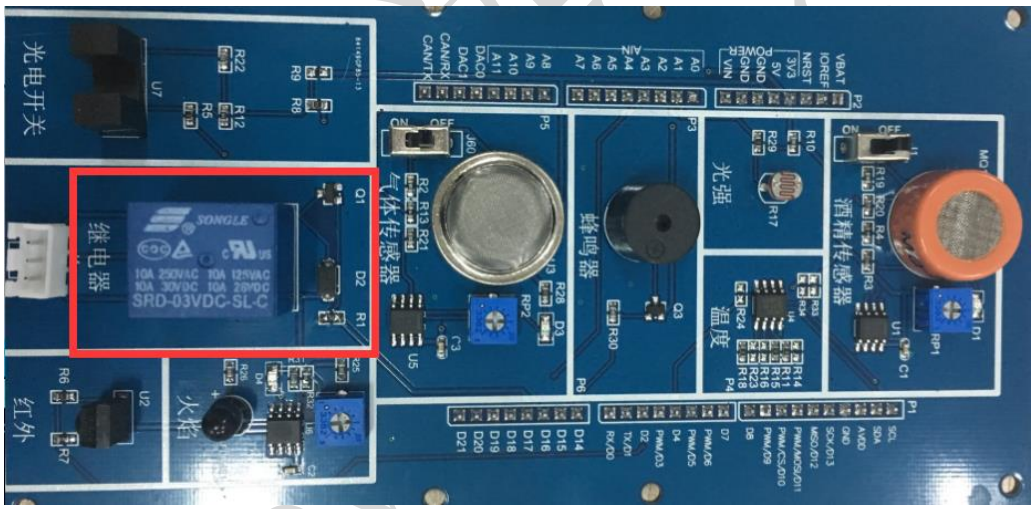


图 实验现象图

试验结束后需卸载模块：

```
lsmod
rmmod fs4412_gpio
```

3.2.5 蜂鸣器驱动

```
cd /fs4412_arduino_driver
insmod  fs4412_gpio.ko
mknod /dev/gpio c 800 6
```

```
./gpio
```

将拨码开关中的 D15 拨至 ON，其他拨码全为 OFF，如下图

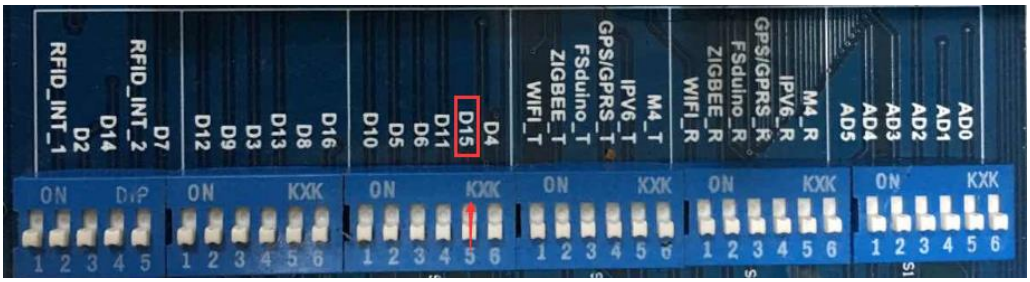
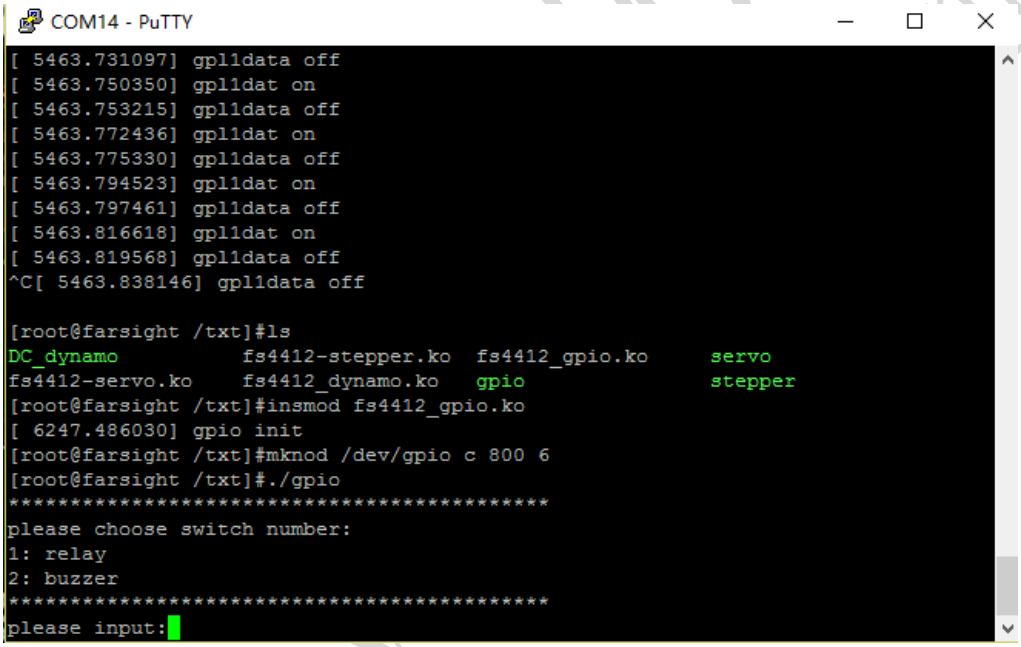
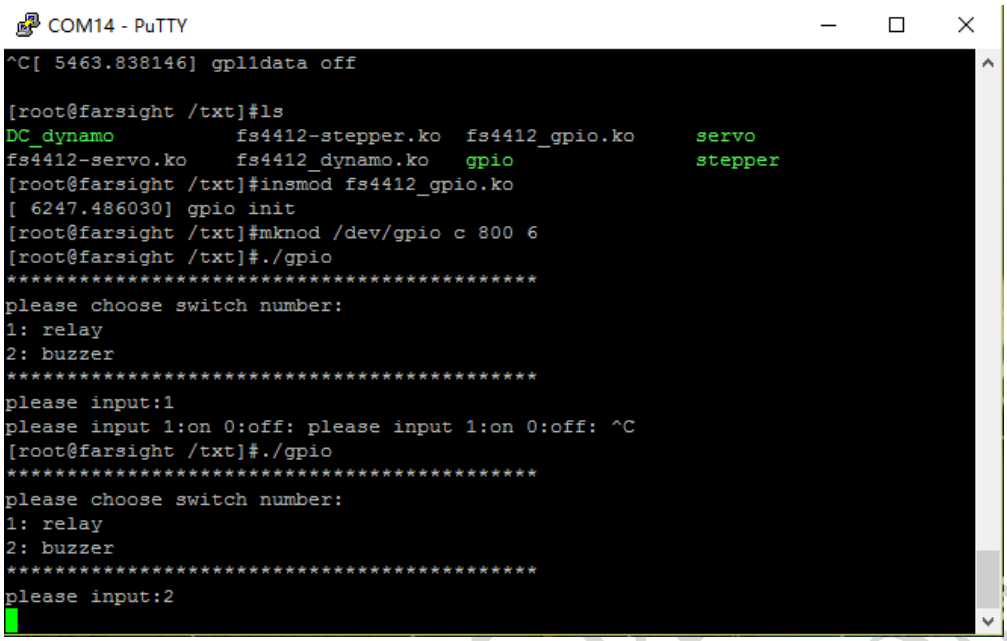


图 拨码开关示意图

如果按照上面的步骤正确操作，在 Putty 上会出现如下界面：



这里测试的是蜂鸣器，所以输入 2，可以看到如下界面，同时听到蜂鸣器发出声音：



如果想要关闭蜂鸣器，只需用组合键 Ctrl + C 键。

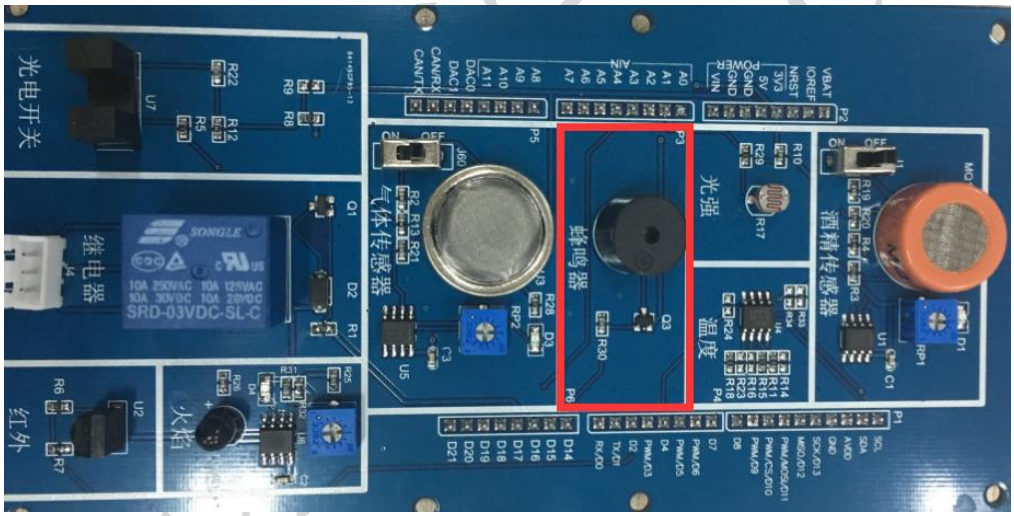


图 实验现象图

试验结束后需卸载模块：

```
lsmod
rmmod fs4412_gpio
```

3.2.6 酒精传感器驱动

```
cd /fs4412_arduino_driver
insmod  fs4412_adc.ko
```

```
mknod /dev/adc c 800 5
./adc
```

将拨码开关中的 AD1 拨至 ON，其他拨码全为 OFF，如下图

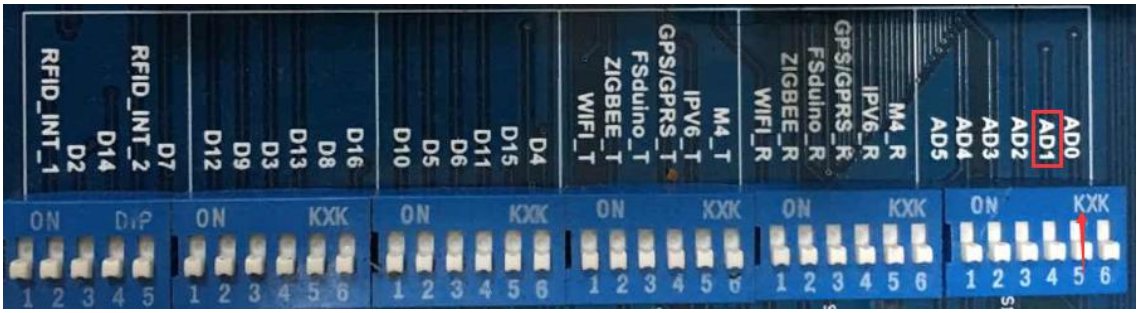


图 拨码开关示意图

出现如下界面：

```
[root@farsight fs4412_arduino_driver]# ./adc
******************[ 2050.652943] fs4412_adc_open:48
***************
please change Sensor type:
A: alcohol sensor
L: light sensor
F: flame sensor
G: gas sensor
********************************
input type:A
```

输入大写字母“A”，可以看到不停的显示当前采集到的模拟电压值，用打火机的气体测试，气体浓度越大，值越大：

```
********************************
input type:A
[ 2053.024902] fs4412_adc_ioctl:88
Vol: 0.76V
Vol: 0.85V
Vol: 0.92V
Vol: 0.97V
Vol: 1.01V
Vol: 1.03V
Vol: 1.04V
^C[ 2059.121439] fs4412_adc_release:54
```

酒精传感器在模块中的位置如下图所示：

图 实验现象图

试验结束后需卸载模块：

```
lsmod
rmmod fs4412_adc
```

3.2.7 光敏传感器驱动

```
cd /fs4412_arduino_driver
insmod  fs4412_adc.ko
mknod /dev/adc c 800 5
./adc
```

将拨码开关中的 AD2 拨至 ON，其他拨码全为 OFF，如下图

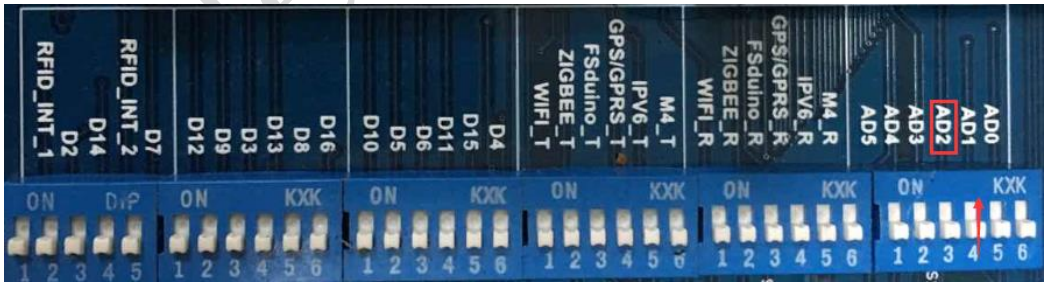


图 拨码开关示意图

出现如下界面：

```
[root@farsight fs4412_arduino_driver]# ./adc
******************[ 2101.535974] fs4412_adc_open:48
**************
please change Sensor type:
A: alcohol sensor
L: light sensor
F: flame sensor
G: gas sensor
*********************************
input type:L
```

输入大写字母“L”，可以看到不停的显示当前采集到的模拟电压值，当用手电筒照射光敏传感器时值会变小：

```
*********************************
input type:L
[ 2172.289114] fs4412_adc_ioctl:88
Vol: 0.41V
Vol: 0.37V
Vol: 0.34V
Vol: 0.25V
Vol: 0.24V
Vol: 0.54V
Vol: 0.52V
^C[ 2179.063164] fs4412_adc_release:54
```

关照传感器在模块上的位置如下图所示：

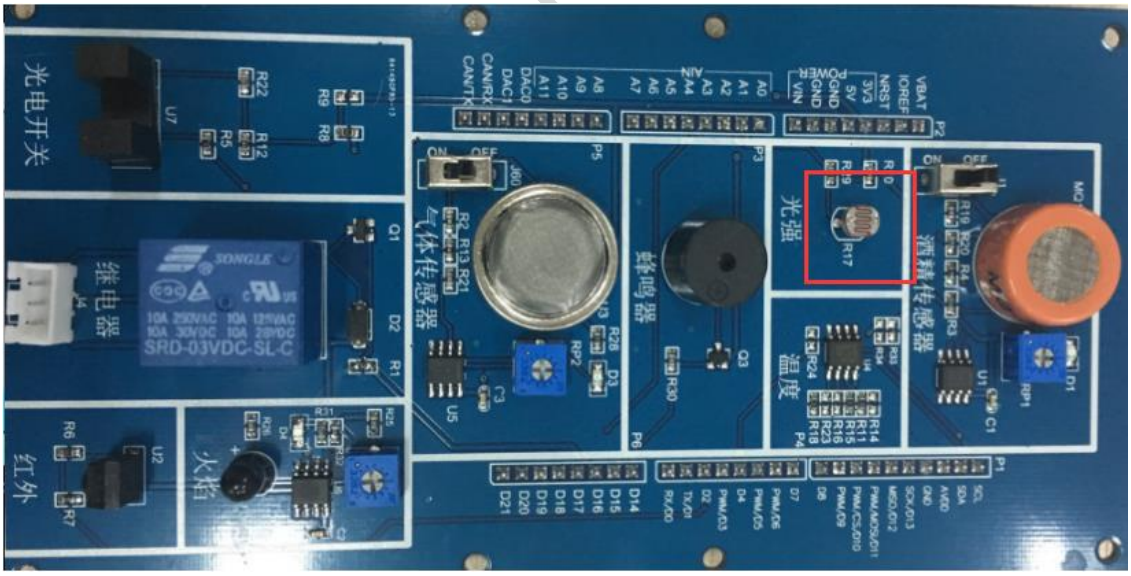


图 实验现象图

试验结束后需卸载模块：

```
lsmod
rmmod fs4412_adc
```

3.2.8 火焰传感器驱动

```
cd /fs4412_arduino_driver
insmod  fs4412_adc.ko
mknod /dev/adc c 800 5
./adc
```

将拨码开关中的 AD3 拨至 ON，其他拨码全为 OFF，如下图

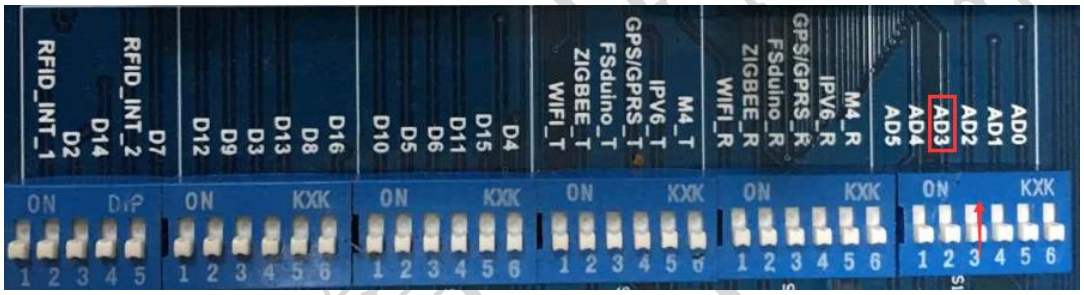


图 拨码开关示意图

出现如下界面：

```
[root@farsight fs4412_arduino_driver]# ./adc
**********[ 1519.289536] fs4412_adc_open:48
**********
please change Sensor type:
A: alcohol sensor
L: light sensor
F: flame sensor
G: gas sensor
**********
input type:F
```

输入大写字母“F”，可以看到不停的显示当前采集到的模拟电压值，用打火机火焰靠近火焰传感器会发现电压值升高：

```
input type:F
[ 1526.185125] fs4412_adc_ioctl:88
Vol: 1.41V
Vol: 1.42V
Vol: 0.59V
Vol: 1.00V
Vol: 1.41V
Vol: 1.35V
Vol: 0.00V
^C[ 1533.114989] fs4412_adc_release:54
```

热敏传感器在模块上的位置如下图：

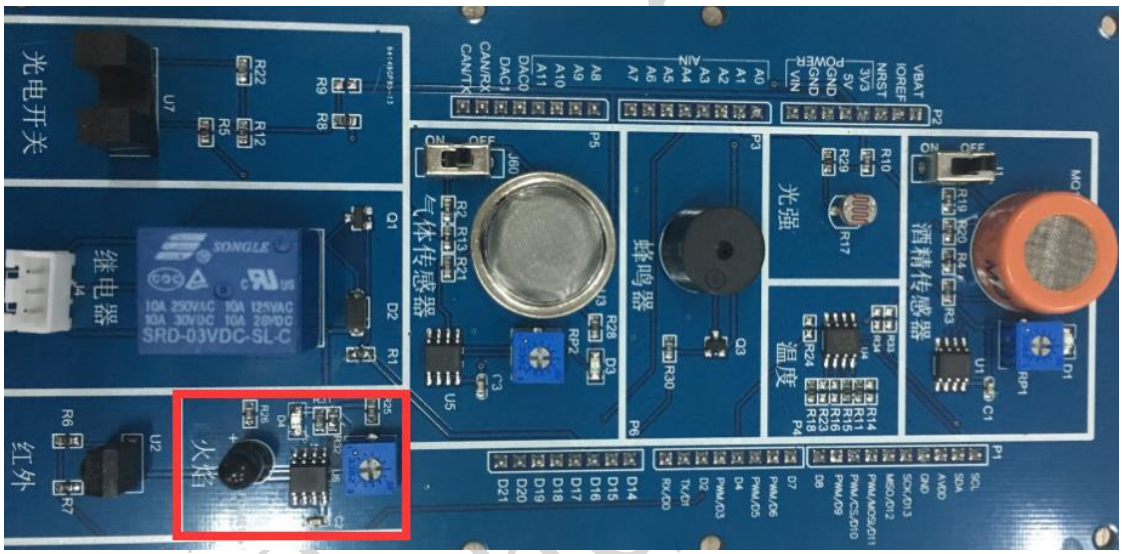


图 实验现象图

试验结束后需卸载模块：

```
lsmod
rmmod fs4412_adc
```

3.2.9 可燃气体传感器驱动

```
cd /fs4412_arduino_driver
insmod  fs4412_adc.ko
mknod /dev/adc c 800 5
./adc
```

将拨码开关中的 AD4 拨至 ON，其他拨码全为 OFF，如下图

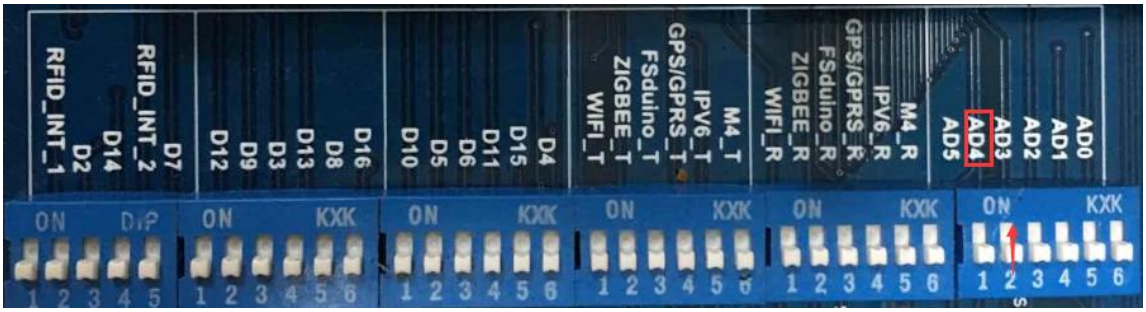


图 拨码开关示意图

出现如下界面：

```
[root@farsight fs4412_arduino_driver]# ./adc
******************[ 2236.904379] fs4412_adc_open:48
***************
please change Sensor type:
A: alcohol sensor
L: light sensor
F: flame sensor
G: gas sensor
*********************************
input type:G
```

输入大写字母“G”，可以看到不停的显示当前采集到的模拟电压值，用打火机的气体模拟检测，当检测到气体浓度越大，值越大：

```
*********************************
input type:G
[ 2243.313776] fs4412_adc_ioctl:88
Vol: 0.73V
Vol: 0.75V
Vol: 0.76V
Vol: 0.77V
Vol: 0.77V
Vol: 0.78V
Vol: 0.78V
^C[ 2250.209983] fs4412_adc_release:54
```

气体传感器在模块中的位置如下图所示：

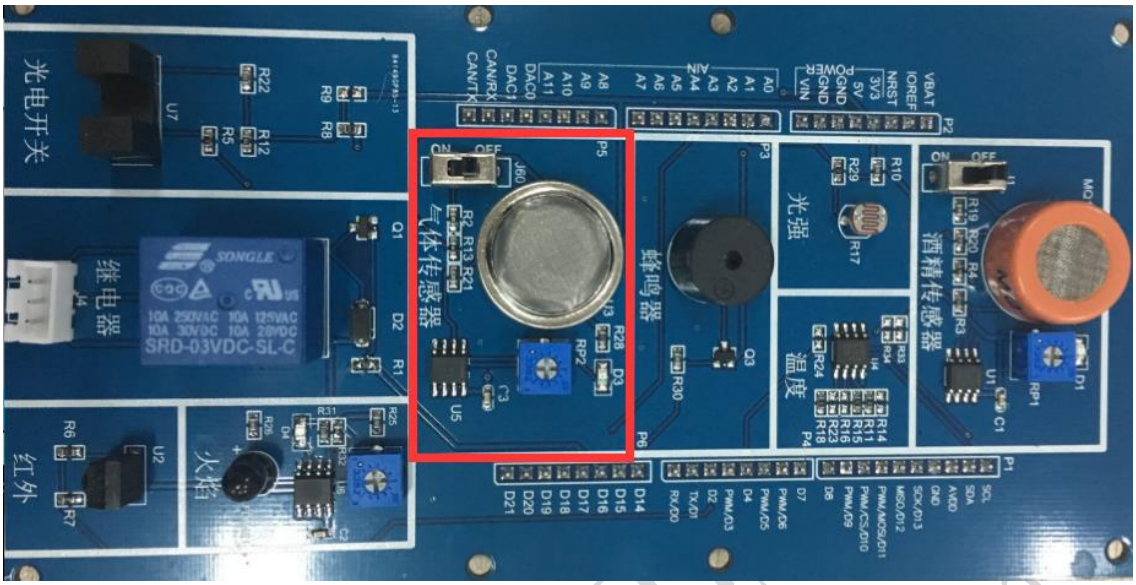


图 实验现象图

试验结束后需卸载模块：

```
lsmod
rmmod fs4412_adc
```

3.2.10 温度传感器驱动

```
cd /fs4412_arduino_driver
./temp
```

该实验只需要注意 SW8 拨码拨至 A8/A9，因为在硬件设计时，将 SW8 作为 MCU 和 CUP 的 i2c 主机切换开关，如下图

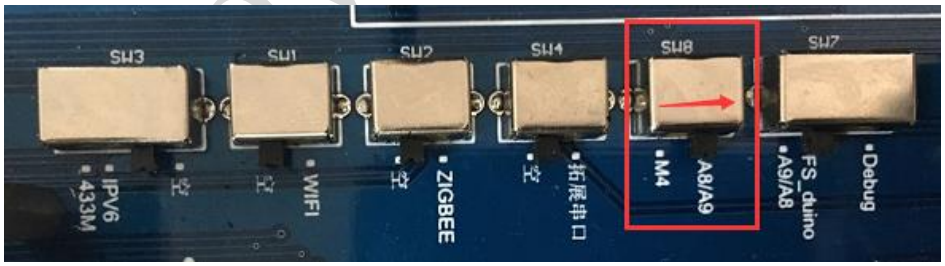


图 拨码开关示意图

```
[root@farsight fs4412_arduino_driver]# ./temp
temp =  30.500000
temp =  30.500000
temp =  30.500000
temp =  31.000000
temp =  31.500000
temp =  32.000000
temp =  32.500000
temp =  32.000000
^C
```

温度传感器在模块上的位置如下：

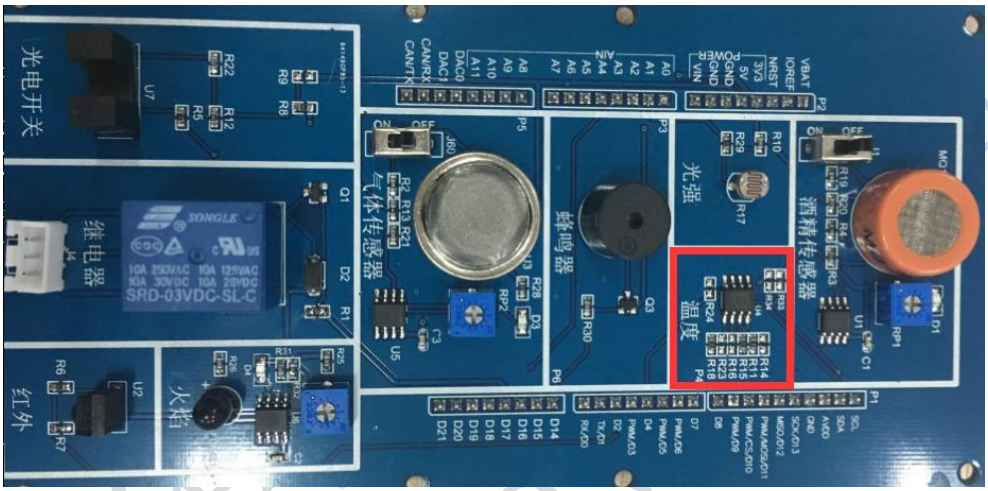


图 实验现象图

3.2.11 按键扫描数码管显示驱动

```
cd /fs4412_arduino_driver
insmod  zlg7290.ko
mknod /dev/zlg7290 c 800 1
./7led
```

该实验只需要注意 SW8 拨码拨至 A8/A9，因为在硬件设计时，将 SW8 作为 MCU 和 CUP 的 i2c 主机切换开关，如下图

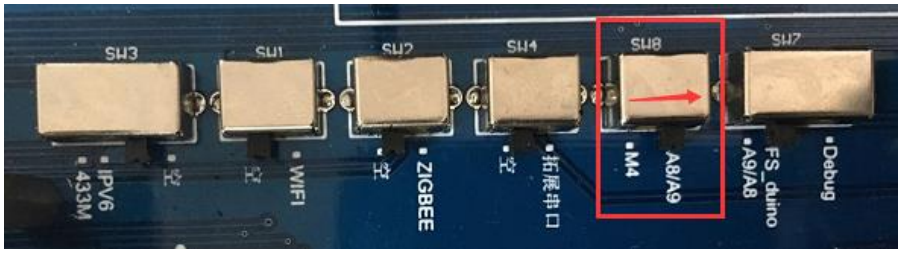


图 拨码开关示意图

可以看到在 Arduino 模块数码管上显示如下信息：

图 实验现象图

试验结束后需卸载模块：

```
lsmod

rmmod zlg7290
```

第 4 章　源码编译

该部分使用的开发环境安装过程请参考【华清远见开发环境-光盘资料】。

建议用户首先在虚拟机的工作目录下建立自己的工作目录，此处就以【/home/linux/workdir】为例，在此目录下新建 3 个的目录，作为此开发板的工作目录。关于 FS4412M4 开发平台的所有代码就在此目录下完成。

```
$ mkdir /home/linux/workdir/bootloader
$ mkdir /home/linux/workdir/kernel
$ mkdir /home/linux/workdir/android
```

```
linux@ubuntu64-vm:~/workdir$ ls
android  bootloader  kernel
```

首先将 FS4412M4 开发平台的源码拷贝到虚拟机的共享目录【D:\share】下。

此时，我们就可以在虚拟机中查看到拷贝的源码了。

```
$ ls    /mnt/hgfs/share/
```

```
linux@ubuntu64-vm:~/workdir$ ls /mnt/hgfs/share/
android4.0-fs4412.tar.xz  linux-3.0-fs4412_v9.tar.xz  u-boot-2010.03-FS4412_v5.tar.xz
```

4.1　编译 Bootloader 源码

4.1.1　拷贝源码到开发环境的工作目录

首先在 FS4412M4 开发平台的工作目录下建立的 bootloader 目录，作为 bootloader 开发目录，然后将共享目录下的 bootloader 源码拷贝至此。

```
$ cd    /home/linux/workdir/bootloader

$ cp    /mnt/hgfs/share/u-boot-FS4412_vX.tar.xz ./      // X 代表版本号，随着版本升级会有区别
```

4.1.1 解压源码

```
$ tar xvf u-boot-FS4412_vX.tar.xz                       // X 代表版本号，随着版本升级会有区别
```

```
u-boot-2010.03-FS4412/lib_generic/sha256.c
u-boot-2010.03-FS4412/lib_generic/addr_map.c
u-boot-2010.03-FS4412/lib_generic/crc32.c
u-boot-2010.03-FS4412/lib_generic/gunzip.c
u-boot-2010.03-FS4412/lib_generic/strmhz.c
u-boot-2010.03-FS4412/lib_generic/bzlib_crctable.c
u-boot-2010.03-FS4412/lib_generic/bzlib_private.h
u-boot-2010.03-FS4412/lib_generic/string.c
u-boot-2010.03-FS4412/lib_generic/div64.c
u-boot-2010.03-FS4412/lib_generic/zlib.c
linux@ubuntu64-vm:~/workdir/bootloader$
```

4.1.2 配置源码

对于 U-Boot 来说，一般对其配置主要修改的是相关平台的配置文件，对于 FS4412M4 开发平台，配置文件的位置为【include/configs/fs4412.h】

```
$ cd    /home/linux/workdir/bootloader/u-boot-2010.03-FS4412/
```

查看或者修改配置文件。

```
linux@ubuntu64-vm:~/workdir/bootloader/u-boot-2010.03-FS4412$ vim include/configs/fs4412.h
```

```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)                    周一 23:16 linux
181 //#define CONFIG_CLK_800_330_165
182 /* APLL : 1GHz  */
183 #define CONFIG_CLK_1000_400_200
184
185
186 #define CONFIG_SPARSEMEM    1       /* Sparsemem for 32 kernel */
187
188 #define BOOT_ONENAND    0x1
189 #define BOOT_NAND       0x40000
190 #define BOOT_MMCSD      0x3
191 #define BOOT_NOR        0x4
192 #define BOOT_SEC_DEV    0x5
193 #define BOOT_EMMC43     0x6
194 #define BOOT_EMMC441    0x7
195
196
197 #define CONFIG_BOOTM_LINUX      1
198
199 /* skip to load BL2 */
200 #define FAST_BOOT       1
201
202 #define CONFIG_SYS_NO_FLASH
203 #define MEMORY_BASE_ADDRESS 0x40000001
204
205 /* input clock of PLL */
206 #define CONFIG_SYS_CLK_FREQ 24000000    /* the SMDKC210 has 24MHz input clock */
207
208 /* MMU Setting  */
209 #define CONFIG_ENABLE_MMU
include/configs/fs4412.h [+]                                  203,38        29%
```

4.1.3 编译源码

修改交叉工具链的路径（我们在打包源码之前都会默认使用开发环境中交叉工具链的路径，一般不用修改，如有必要按下图路径更改）。

```
$vim Makefile
```

修改 162 行代码如下图所示。

```
162    CROSS_COMPILE = /usr/local/toolchain/toolchain-4.5.1/bin/arm-none-linux-gnueabi-
```

```
162 CROSS_COMPILE = /usr/local/toolchain/toolchain-4.5.1/bin/arm-none-linux-gnueabi-
```

执行编译脚本编译 U-Boot。

```
linux@ubuntu64-vm:~/workdir/bootloader/u-boot-2010.03-FS4412$ ls
api                          common     doc        lib_avr32      lib_microblaze  lib_sparc    nand_spl       rules.mk
board                        config.mk  drivers    lib_blackfin   lib_mips        MAINTAINERS  net            sdfuse
build_uboot.sh               COPYING    examples   libfdt         lib_nios        MAKEALL      onenand_ipl    sdfuse_q
CHANGELOG                    cpu        fs         lib_generic    lib_nios2       Makefile     post           tags
CHANGELOG-before-U-Boot-1.1.5 CREDITS   include    lib_i386       lib_ppc         mkconfig     README         tc4_cmm.cmm
CodeSign4SecureBoot          disk       lib_arm    lib_m68k       lib_sh          mkuboot.sh   readme.fs4412  tools
```

```
$ make distclean
$ ./build_uboot.sh
```

```
/usr/local/toolchain/toolchain-4.5.1/bin/arm-none-linux-gnueabi-objcopy -O srec u-boot u-boot.srec
/usr/local/toolchain/toolchain-4.5.1/bin/arm-none-linux-gnueabi-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
make[1]: 正在进入目录 `/home/linux/workdir/bootloader/u-boot-2010.03-FS4412/sdfuse_q'
gcc -o  chksum  chksum.c
gcc -o  add_sign  add_sign.c
gcc -o  add_padding  add_padding.c
make[1]:正在离开目录 `/home/linux/workdir/bootloader/u-boot-2010.03-FS4412/sdfuse_q'
bl2aa file size= 14336B
before padding uboot.bin file size= 250204B
35668 B written


linux@ubuntu64-vm:~/workdir/bootloader/u-boot-2010.03-FS4412$
```

如上图所示即为编译成功。下图所示【u-boot-fs4412.bin】即为编译生成的 u-boot 二进制文件。

```
linux@ubuntu64-vm:~/workdir/bootloader/u-boot-2010.03-FS4412$ ls
api                          COPYING   include       lib_microblaze  MAKEALL      README          tools
board                        cpu       lib_arm       lib_mips        Makefile     readme.fs4412   u-boot
build_uboot.sh               CREDITS   lib_avr32     lib_nios        mkconfig     rules.mk        u-boot.bin
CHANGELOG                    disk      lib_blackfin  lib_nios2       mkuboot.sh   sdfuse          u-boot-fs4412.bin
CHANGELOG-before-U-Boot-1.1.5 doc      libfdt        lib_ppc         nand_spl     sdfuse_q        u-boot.lds
CodeSign4SecureBoot          drivers   lib_generic   lib_sh          net          System.map      u-boot.map
common                       examples  lib_i386      lib_sparc       onenand_ipl  tags            u-boot.srec
config.mk                    fs        lib_m68k      MAINTAINERS     post         tc4_cmm.cmm
linux@ubuntu64-vm:~/workdir/bootloader/u-boot-2010.03-FS4412$
```

4.2 编译 Linux 内核源码

4.2.1 拷贝源码到开发环境的工作目录

首先在 FS4412M4 开发平台的工作目录下建立的 kernel 目录，作为内核的开发目录，然后将共享目录下的 Linux 源码拷贝至此。

```
$ cd /home/linux/workdir/kernel
$ cp /mnt/hgfs/share/linux-3.0-fs4412_vX.tar.xz    ./     // X 代表版本号，随着版本升级会有区别
```

```
linux@ubuntu64-vm:~/workdir/kernel$ ls
linux-3.0-fs4412_v9.tar.xz
```

4.2.2 解压源码

```
$ tar xvf linux-3.0-fs4412_vX.tar.xz                    // X 代表版本号，随着版本升级会有区别
```

```
linux-3.0-fs4412_v9/samples/hidraw/Makefile
linux-3.0-fs4412_v9/samples/hidraw/hid-example.c
linux-3.0-fs4412_v9/samples/tracepoints/
linux-3.0-fs4412_v9/samples/tracepoints/Makefile
linux-3.0-fs4412_v9/samples/tracepoints/tp-samples-trace.h
linux-3.0-fs4412_v9/samples/tracepoints/tracepoint-probe-sample.c
linux-3.0-fs4412_v9/samples/tracepoints/tracepoint-sample.c
linux-3.0-fs4412_v9/samples/tracepoints/tracepoint-probe-sample2.c
linux@ubuntu64-vm:~/workdir/kernel$
```

4.2.3 配置源码

进入到内核的源码路径下。

```
$ cd /home/linux/workdir/kernel/linux-3.0-fs4412_VX
```

修改交叉工具链的路径（我们在打包源码之前都会默认使用开发环境中交叉工具链的路径，一般不用修改，如有必要按下图路径更改）。

```
$ vim Makefile
```

修改 198 行代码如下图所示。

```
198    CROSS_COMPILE = /usr/local/toolchain/toolchain-4.5.1/bin/arm-none-linux-gnueabi-
```

```
198 CROSS_COMPILE   ?= /usr/local/toolchain/toolchain-4.5.1/bin/arm-none-linux-gnueabi-
```

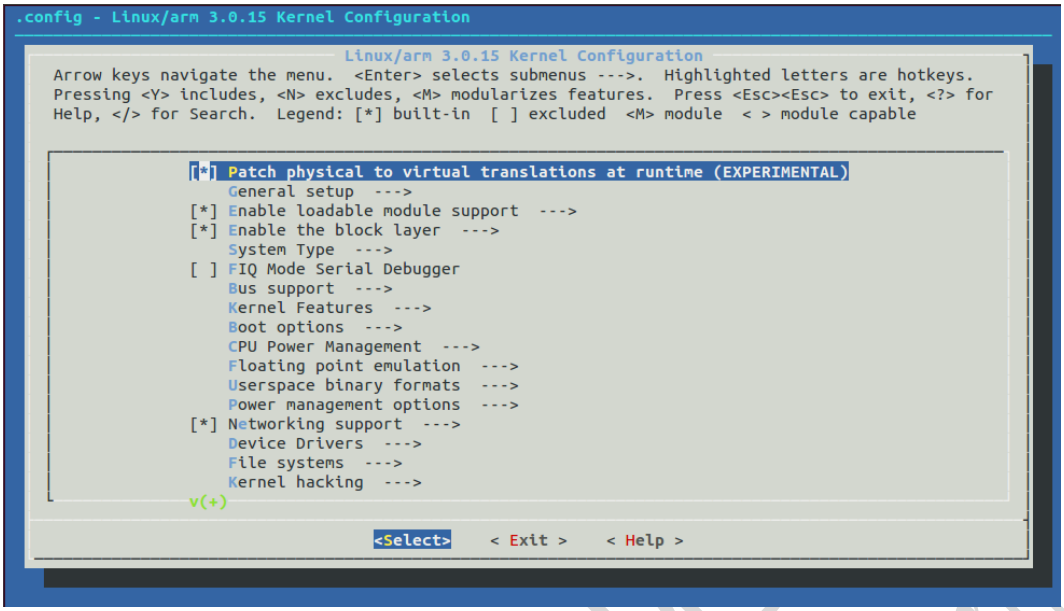
Linux 内核通常使用 menuconfig 图形界面配置内核编译选项，配置更改的内容会保存在内核源码目录下的【.config】文件中。首先拷贝 FS4412M4 开发平台的标准配置文件为【.config】。

```
$ cp arch/arm/configs/fs4412_vX_deconfig .config          // X 为版本号，请拷贝最新版本的配置文件
```

```
~/workdir/kernel/linux-3.0-fs4412_v9$ cp arch/arm/configs/fs4412_v8_defconfig .config
```

在终端下输入下列命令可以进入到 Linux 内核配置图形界面。

```
$ make menuconfig
```



Menuconfig 菜单使用【Enter】键进入下级菜单；使用【Space】键选中或者清除选项；使用【？】查看此菜单的帮助文件，使用连击【ESC】两次后退至上级菜单。

4.2.4 编译源码

```
$ make zImage –jX                    // X 为编译时使用的 CPU 线程数，建议此数与环境搭建中 CPU 个数一致
```

编译内核源码大概需要几分钟的时间，此时 CPU 会高负荷运转，电脑性能可能会有下降，此为正常现象，此时应避免其他 CPU 高负荷工作。

如下图所示，内核源码编译成功。

```
  GZIP    arch/arm/boot/compressed/piggy.gzip
  AS      arch/arm/boot/compressed/piggy.gzip.o
  CC      arch/arm/boot/compressed/misc.o
  CC      arch/arm/boot/compressed/decompress.o
  SHIPPED arch/arm/boot/compressed/lib1funcs.S
  AS      arch/arm/boot/compressed/lib1funcs.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
linux@ubuntu64-vm:~/workdir/kernel/linux-3.0-fs4412_v9$
```

查看编译生成的内核二进制文件。

```
$ ls arch/arm/boot/zImage -al
```

```
linux@ubuntu64-vm:~/workdir/kernel/linux-3.0-fs4412_v9$ ls arch/arm/boot/zImage -l
-rwxrwxr-x 1 linux linux 3810728  8月 15 14:16 arch/arm/boot/zImage
```

4.3 编译 Android 源码

4.3.1 拷贝源码到开发环境的工作目录

首先在 FS4412M4 开发平台的工作目录下建立 android 目录，作为内核的开发目录，然后将共享目录下的 Android 源码拷贝至此。

```
$ cd /home/linux/workdir/android

$ cp /mnt/hgfs/share/android4.0-fs4412_vX.tar.xz ./   // X 代表版本号，随着版本升级会有区别
```

```
linux@ubuntu64-vm:~/workdir/android$ cp /mnt/hgfs/share/android4.0-fs4412.tar.xz ./
linux@ubuntu64-vm:~/workdir/android$ ls
android4.0-fs4412.tar.xz
```

4.3.2 解压源码

```
$ tar xvf android4.0-fs4412_vX.tar.xz                    // X 代表版本号，随着版本升级会有区别
```

```
android4.0-fs4412/vendor/uart/uart_stub/
android4.0-fs4412/vendor/uart/uart_stub/module/
android4.0-fs4412/vendor/uart/uart_stub/module/uart.c.bak
android4.0-fs4412/vendor/uart/uart_stub/module/uart.c
android4.0-fs4412/vendor/uart/uart_stub/module/cpso.sh
android4.0-fs4412/vendor/uart/uart_stub/module/Android.mk
android4.0-fs4412/vendor/uart/uart_stub/include/
android4.0-fs4412/vendor/uart/uart_stub/include/uart.h
linux@ubuntu64-vm:~/workdir/android$
```

4.3.3 配置源码

进入到 Android 的源码路径下，执行编译脚本即可编译 Android 源码。

```
$ cd /home/linux/workdir/android
```

编译【fs4412_build.sh】文件，修改编译 Android 所使用的线程数。（建议配置一般的电脑线程数为虚拟机的 CPU 线程的数量，配置较高的电脑可以选择在[CPU 数量+1， CPU 数量 x2]的区间内，盲目过多的的编译线程设置不仅不会增加编译效率，反而会拖慢虚拟机以及主机的运行）

```
$ vim fs4412_build.sh
```

在第 4 行修改编译 Android 使用的线程数。

```
1 #!/bin/bash
2
3 CPU_JOB_NUM=$(grep processor /proc/cpuinfo | awk '{field=$NF};END{print field+1}')
4 CPU_JOB_NUM=(4)
5 CLIENT=$(whoami)
6
```

如果不确定怎么设置编译 Android 使用的线程数，请用【#】注释掉第 4 行的内容，编译脚本会依照规则计算线程数。

```
1 #!/bin/bash
2
3 CPU_JOB_NUM=$(grep processor /proc/cpuinfo | awk '{field=$NF};END{print field+1}')
4 #CPU_JOB_NUM=(4)
5 CLIENT=$(whoami)
6
```

4.3.4 编译源码

执行【fs4412_build.sh】编译 Android 源码。

Android 代码非常庞大，使用虚拟机编译 Android 源码所需要的时间会相当长，期间有可能出现机器卡死或者假死的情况，在编译的过程中请不要轻易操作虚拟机。

```
make -j1 PRODUCT-full_smdk4x12-eng

============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.0.3
TARGET_PRODUCT=full_smdk4x12
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a-neon
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=IML74K
============================================
```

下图所示即为编译 Android 成功。

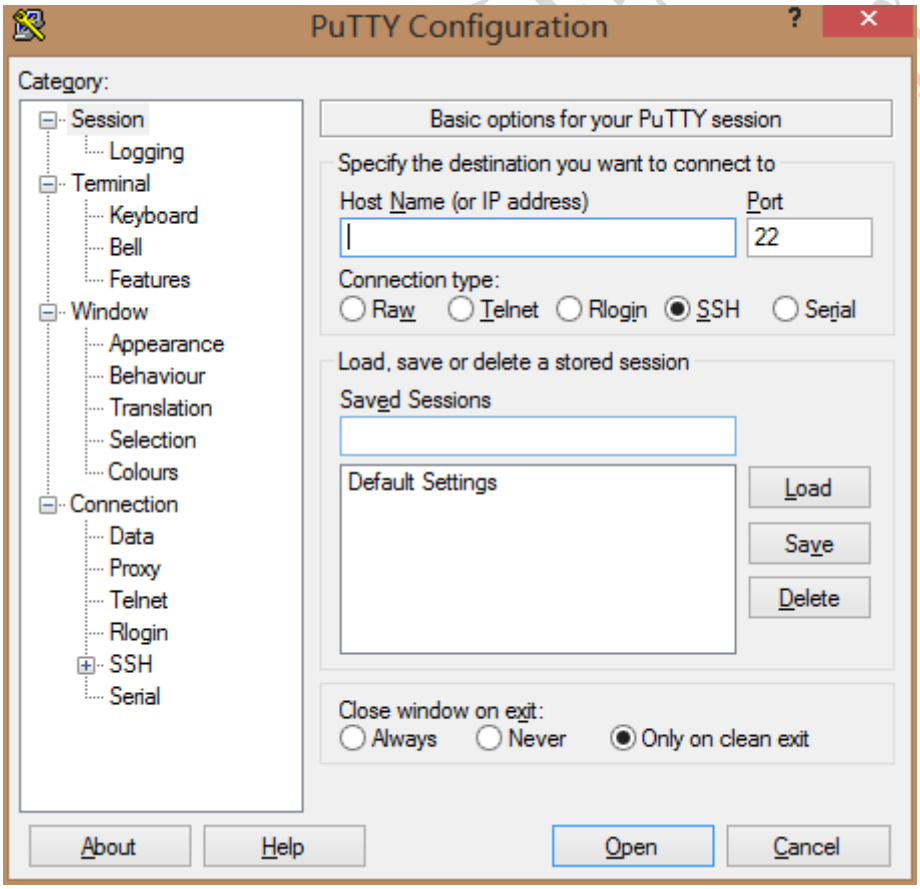
```
Add resources to package (out/target/product/smdk4x12/obj/APPS/android.core.tests.libcore.package.libcore_interm
# javalib.jar should only contain .dex files, but the harmony tests also include
# some .class files, so get rid of them
Total compile time is 12991 seconds

[[[[[[[ Make ramdisk image for u-boot ]]]]]]]

Image Name:   ramdisk
Created:      Tue Jul  1 04:24:39 2014
Image Type:   ARM Linux RAMDisk Image (uncompressed)
Data Size:    921076 Bytes = 899.49 kB = 0.88 MB
Load Address: 40800000
Entry Point:  40800000


[[[[[[[ Make additional images for fastboot ]]]]]]]

No zImage is found at /home/linux/workdir/fs4412/android/fs4412/../../kernel/linux-3.0-fs4412_V2//arch/arm/boot
  Please set KERNEL_DIR if you want to make additional images
  Ex.) export KERNEL_DIR=~ID/android_kernel_smdk4x12

ok success !!!
```

```
linux@ubuntu64-vm:~/workdir/android$ ls out/target/product/smdk4x12/ -al
总用量 211212
drwxrwxr-x  7 linux linux      4096  7月   1 04:24 .
drwxrwxr-x  3 linux linux      4096  7月   1 00:30 ..
-rw-rw-r--  1 linux linux        19  7月   1 00:32 android-info.txt
-rw-rw-r--  1 linux linux     17897  7月   1 00:48 clean_steps.mk
drwxrwxr-x  5 linux linux      4096  7月   1 02:16 data
-rw-rw-r--  1 linux linux     44343  7月   1 04:16 installed-files.txt
drwxrwxr-x 14 linux linux      4096  7月   1 04:15 obj
-rw-rw-r--  1 linux linux       566  7月   1 00:48 previous_build_config.mk
-rw-rw-r--  1 linux linux    921140  7月   1 04:24 ramdisk-uboot.img
drwxrwxr-x  9 linux linux      4096  7月   1 02:13 root
drwxrwxr-x  5 linux linux      4096  7月   1 02:12 symbols
drwxrwxr-x 13 linux linux      4096  7月   1 02:31 system
-rw-r--r--  1 linux linux 208464988  7月   1 04:18 system.img
-rw-r--r--  1 linux linux   6787388  7月   1 02:16 userdata.img
```

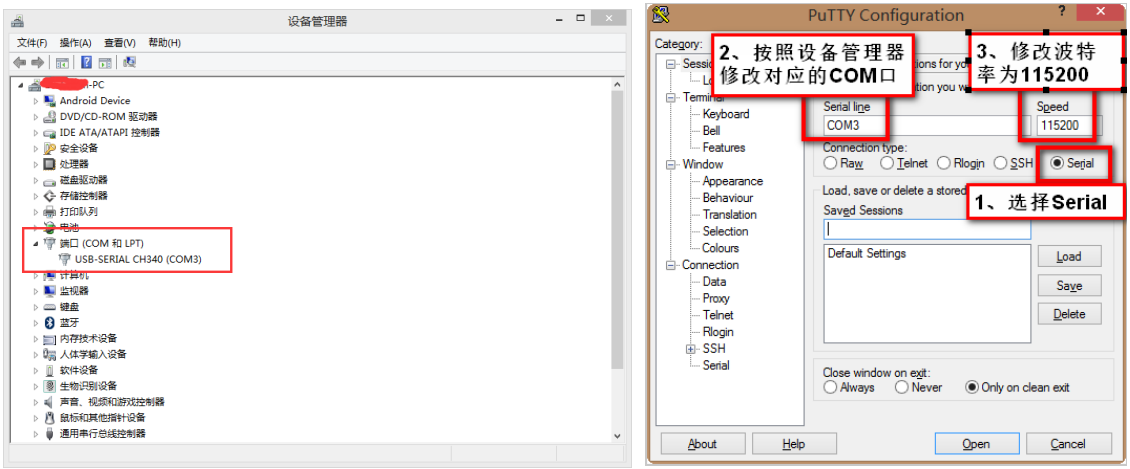
第 5 章　镜像烧写

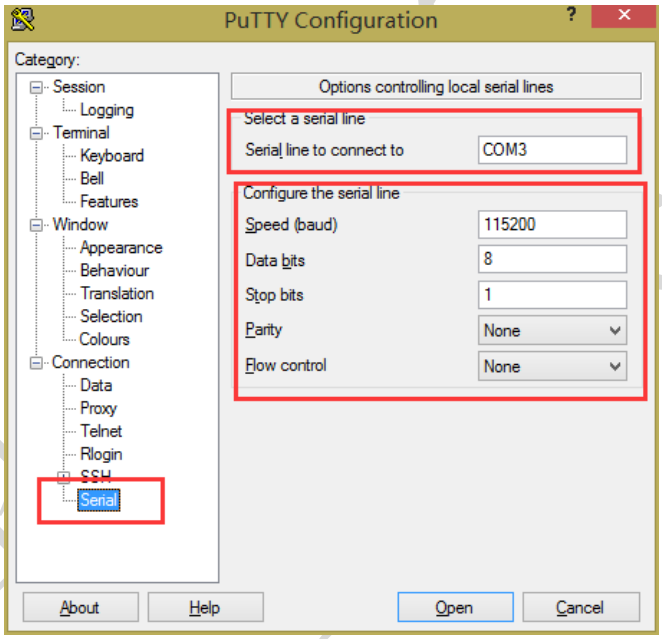
5.1　设置串口调试工具

打开【华清远见-CORTEXA9 资料\工具软件\Windows\串口调试工具\putty.exe】文件。



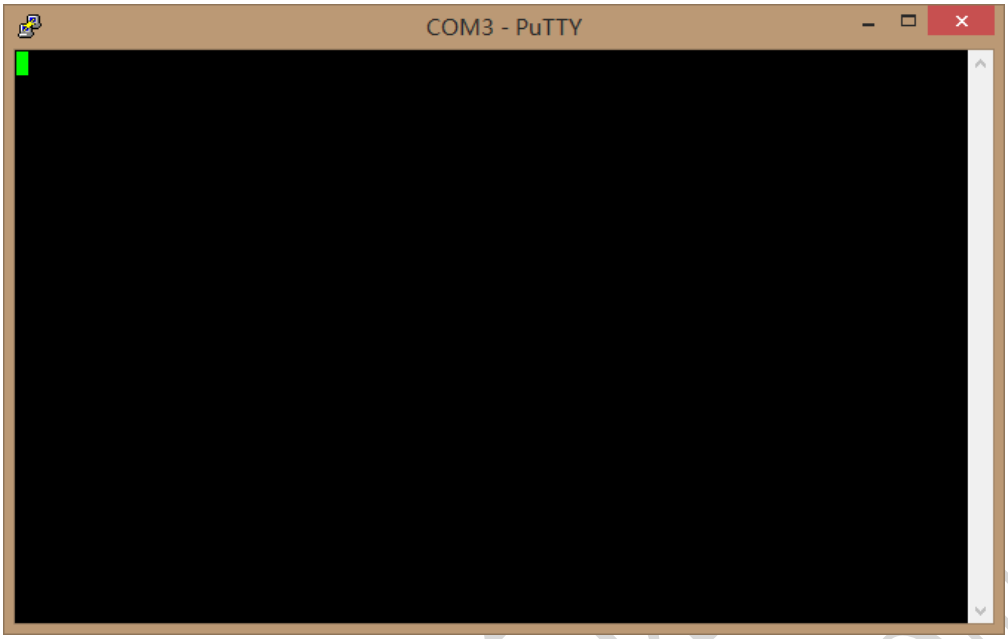
选择串口（Serial）连接方式：





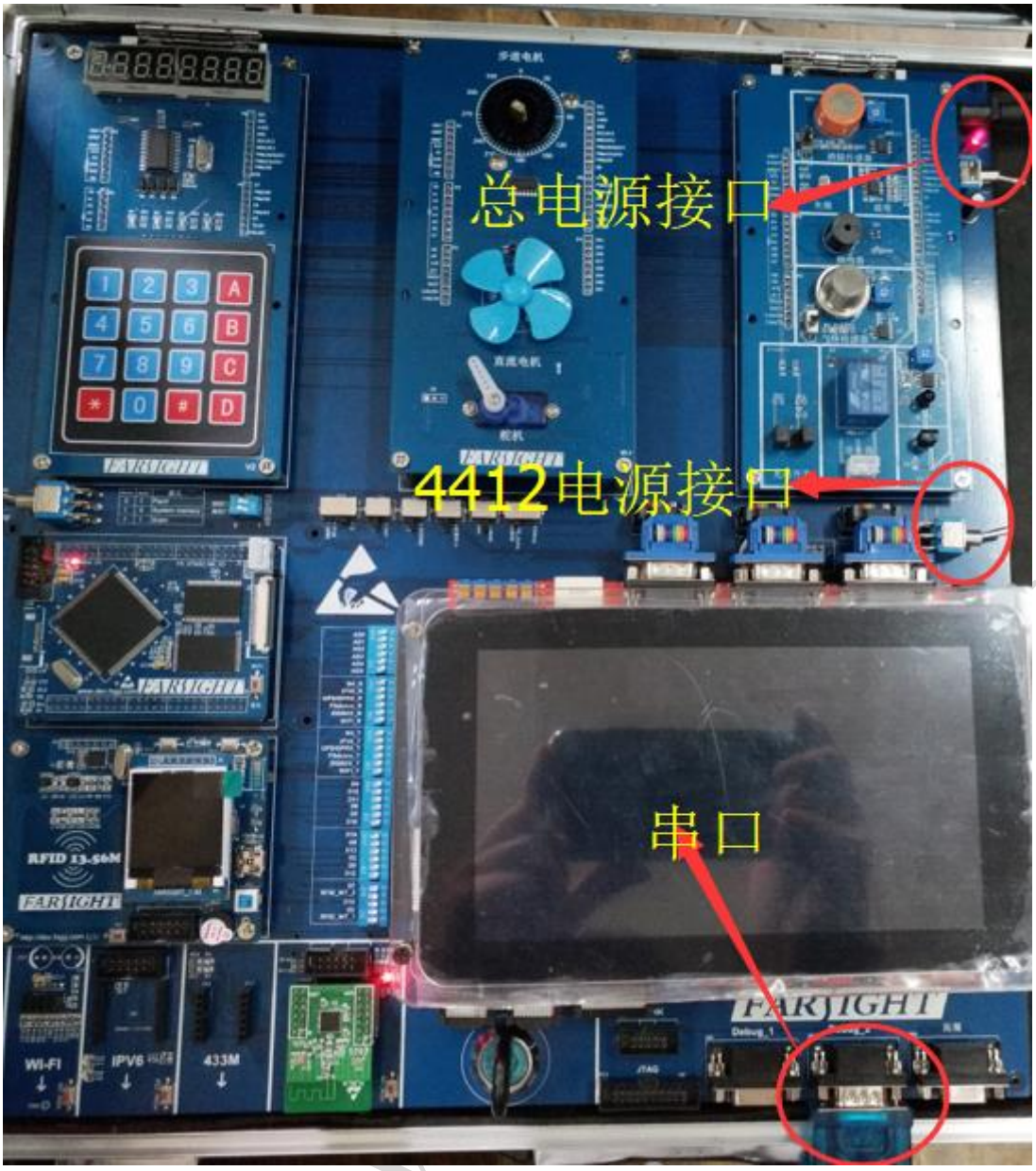
选中第一步方框内的 Serial,再点击第二步中的 Serial,进入串口设置的对话框（注意：这里如果使用台式机物理串口一般为 COM1 ;使用 USB 转串口线需要提前安装驱动程序 ,光盘中提供常用的 CH340 和 PL2303 两种驱动，路径为【华清远见-CORTEXA9 资料\工具软件\Windows\USB 转串口驱动】，如果使用其他 USB 转串口线请自行安装驱动程序）:

点击 open 打开串口。



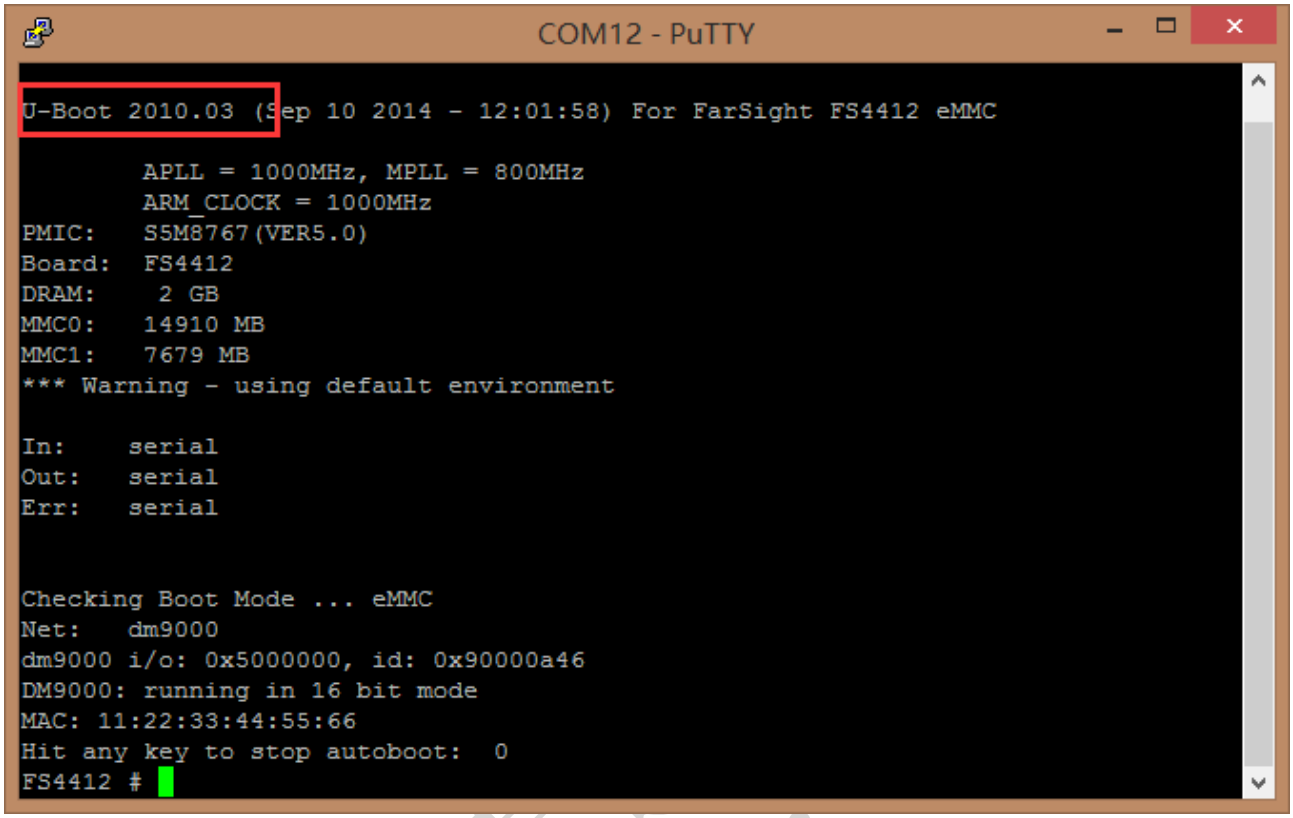
5.2 连接开发板

需要使用串口线连接以及接通电源。



5.3 制作 SD 卡启动盘（只需在开发板不是 UBoot 2010 时做）

如果启动开发板，U-Boot 处如下图红框所示相同，**可以省略此章节步骤**。



5.3.1 拷贝代码

```
$ cd ~
```

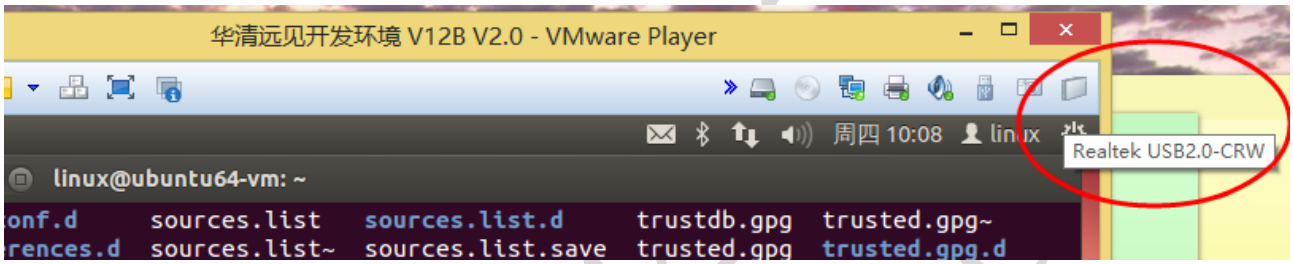
将【华清远见-CORTEXA9 资料\程序源码\Linux 移植实验源码\ SD 卡启动制作工具】录下的“sd_fusing”拷贝到虚拟机 Ubuntu 的共享目录下。


```
$ cp /mnt/hgfs/share/sdfuse_q/ ./ -a
```

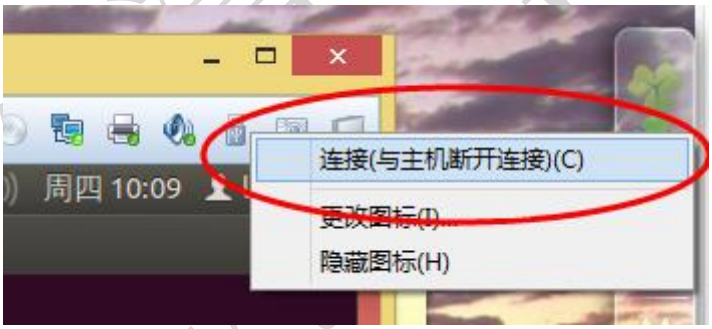
```
$ cd sdfuse_q        //进入 sdfuse_q 目录

$ make               //执行编译命令
```

```
linux@ubuntu64-vm:~/workdir$ cd ./sdfuse_q/
linux@ubuntu64-vm:~/workdir/sdfuse_q$ ls
add_padding  add_padding.c  add_sign  add_sign.c  chksum  chksum.c  Makefile  mkuboot.sh  sd_fusing_exynos4x12.sh  u-boot-fs4412.bin
linux@ubuntu64-vm:~/workdir/sdfuse_q$ make
gcc -o  chksum  chksum.c
gcc -o  add_sign  add_sign.c
gcc -o  add_padding  add_padding.c
linux@ubuntu64-vm:~/workdir/sdfuse_q$
```

用读卡器将 SD 卡插入电脑，虚拟机识别到 SD 读卡器。



右键点击图标，选择【连接】



查看生成的设备节点，笔者 SD 卡在 Ubuntu 系统中的设备节点是/dev/sdb，这里提供一种方式查看设备节点，首先输入 ls /dev/sd*【*代表匹配所有符合 sd 的选项】，sd*最后的设备为 sdb。

```
linux@ubuntu64-vm:~/workdir/sdfuse_q$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda5  /dev/sdb  /dev/sdb1
linux@ubuntu64-vm:~/workdir/sdfuse_q$
```

使用 df –Th 命令，下图就是整个 SD 卡被 ubuntu 识别之后所产生的设备节点。(从容量来说和 SD 卡容

量对等，从挂载点来说符合一般的 SD 卡挂载点)。

```
linux@ubuntu64-vm:~/workdir/sdfuse_q$ df -Th
文件系统       类型      容量  已用  可用 已用% 挂载点
/dev/sda1      ext4       78G  7.6G   66G   11% /
udev           devtmpfs  486M  4.0K  486M    1% /dev
tmpfs          tmpfs     198M  796K  197M    1% /run
none           tmpfs     5.0M     0  5.0M    0% /run/lock
none           tmpfs     495M  200K  495M    1% /run/shm
.host:/        vmhgfs    531G   52G  480G   10% /mnt/hgfs
/dev/sdb1      vfat      7.3G  652K  7.3G    1% /media/CE59-2E38
```

NOTE：关于 SD 卡或者 U 盘在 ubuntu 下识别顺序的问题，有如下的规则：在插入的 SD 卡或者 U 盘设备被 ubuntu 识别之后，会依次识别成 b，c，d.....如果在插入需要制作的 SD 卡后没有其他的 SD 卡或者 U 盘设备插入，那么插入的 SD 卡会被识别成为/dev/sd*下的最后一个纯字母的设备(如下图中即被识别成为 sdb)，即类似于下图：

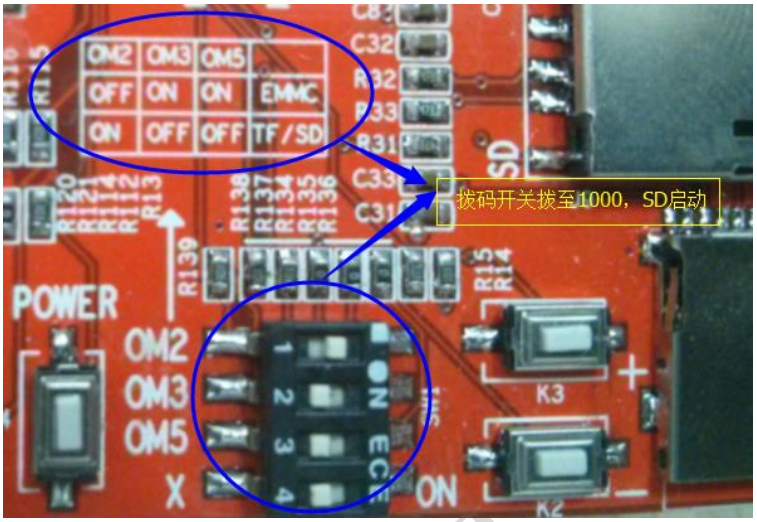
```
linux@ubuntu64-vm:~/workdir/sdfuse_q$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda5  /dev/sdb  /dev/sdb1
linux@ubuntu64-vm:~/workdir/sdfuse_q$
```

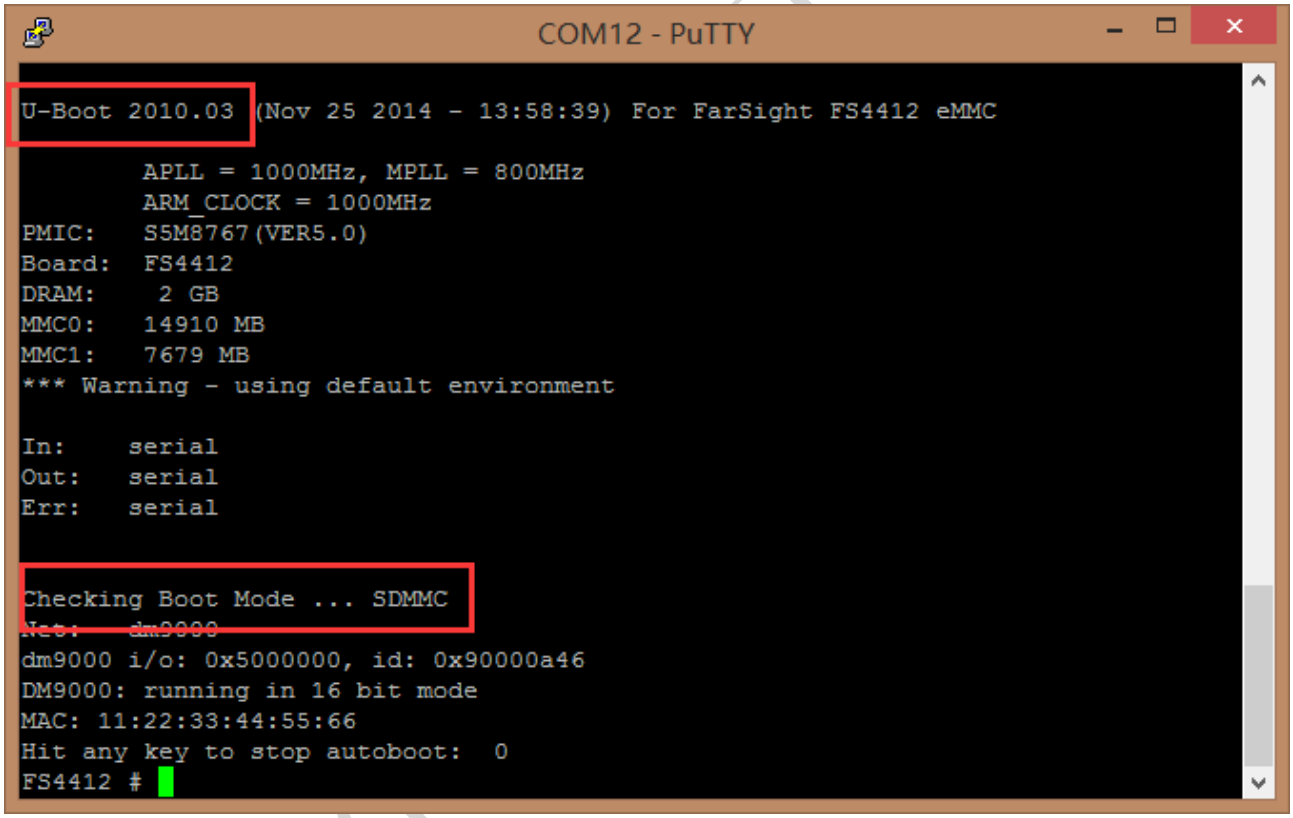
在确定了设备节点之后，使用如下的命令制作 SD 卡，如果识别的节点不是 sdb，则需要更换为识别的节点。

```
$ sudo ./mkuboot.sh /dev/sdb          //将 uboot 烧写到 sd 卡中
```

```
linux@ubuntu64-vm:~/workdir/sdfuse_q$ sudo ./mkuboot.sh /dev/sdb
[sudo] password for linux:
Fuse FS4412 trustzone uboot file into SD card
/dev/sdb reader is identified.
u-boot-fs4412.bin fusing...
记录了1029+1 的读入
记录了1029+1 的写出
527104字节(527 kB)已复制，4.88725 秒，108 kB/秒
u-boot-fs4412.bin image has been fused successfully.
Eject SD card
linux@ubuntu64-vm:~/workdir/sdfuse_q$
```

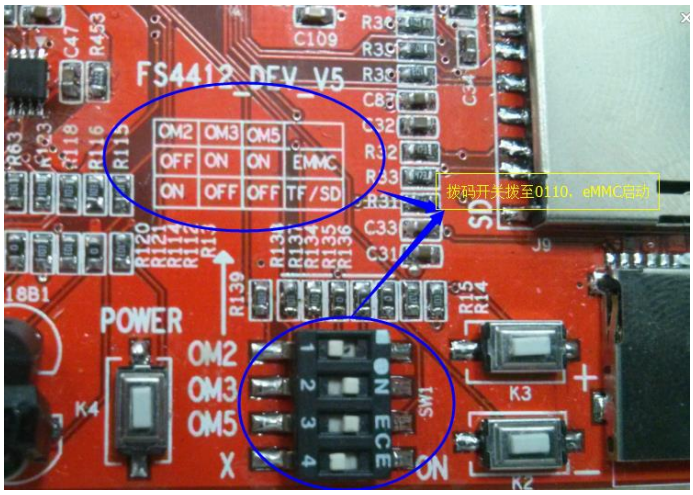
将 SD 卡插入开发板 SD 卡槽内，拨码拨至 1000，启动开发板。

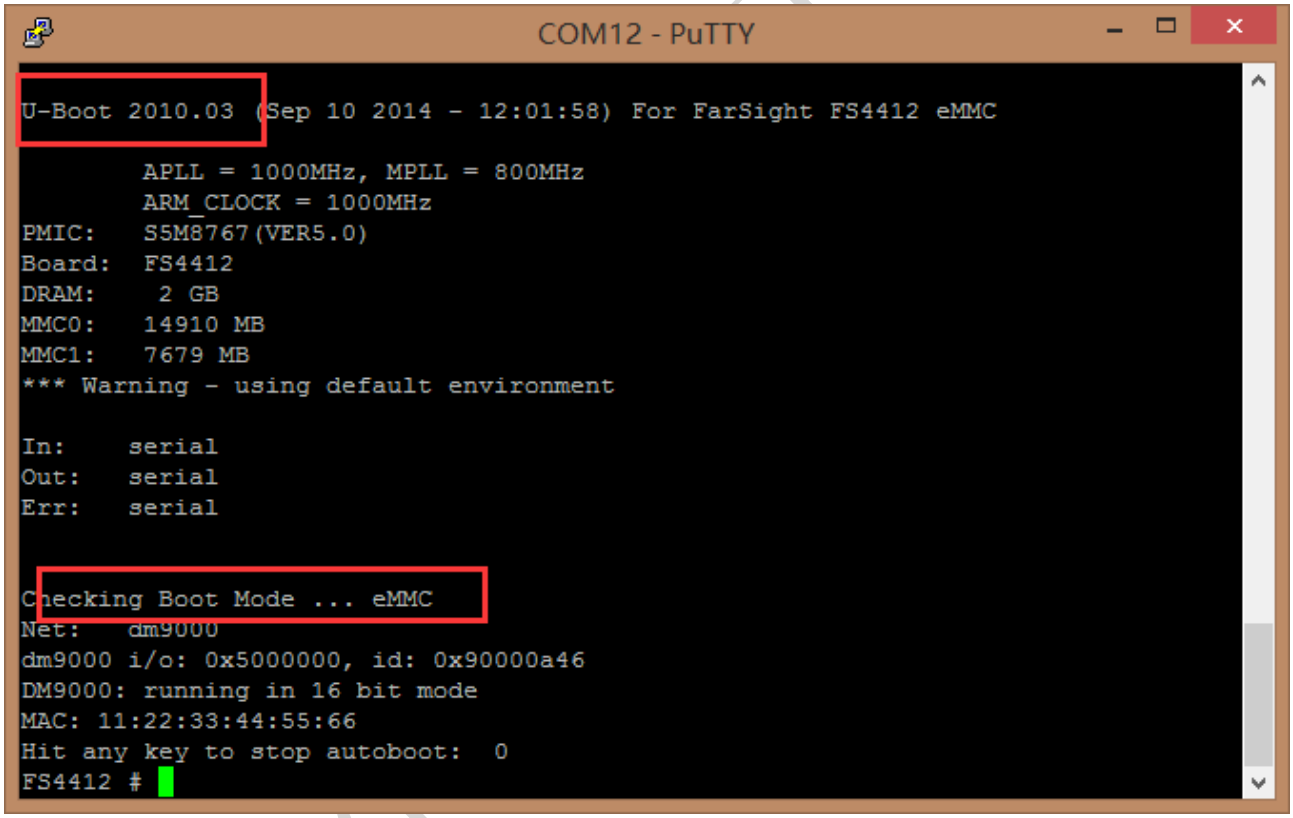




5.3.2 启动开发板

如果没有做 SD 卡启动章节，拨动拨码开关至【0110】，如下图所示，系统开机即从 eMMC 启动。





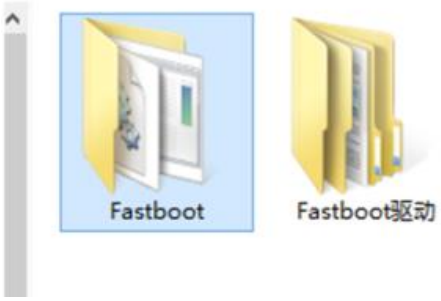
5.4 环境配置

在安卓手机中 Fastboot 是一种比 Recovery 更底层的刷机模式。Fastboot 是一种线刷，就是使用 USB 数据线连接手机的一种刷机模式。相对于某些系统卡刷来说，线刷更可靠，安全。

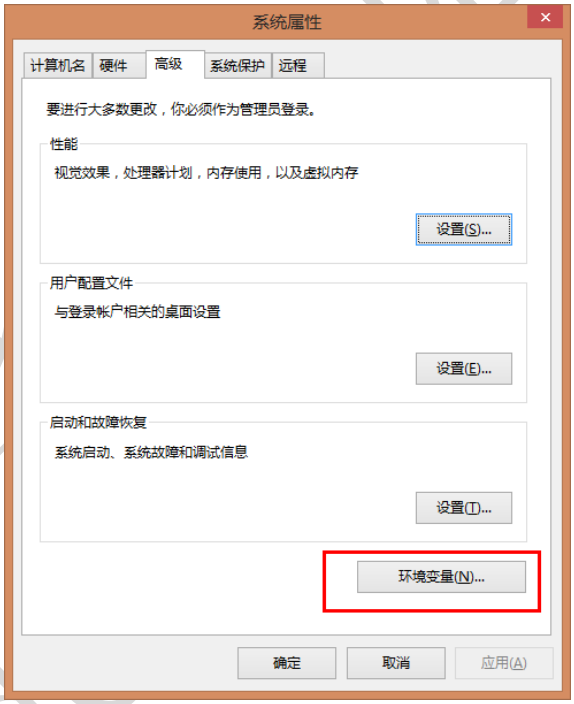
Fastboot 工具在光盘的【华清远见-CORTEXA9 资料\工具软件\Windows\Fastboot\Fastboot】，为了方便把他解压到 D 盘。

5.4.1 拷贝 Fastboot 工具

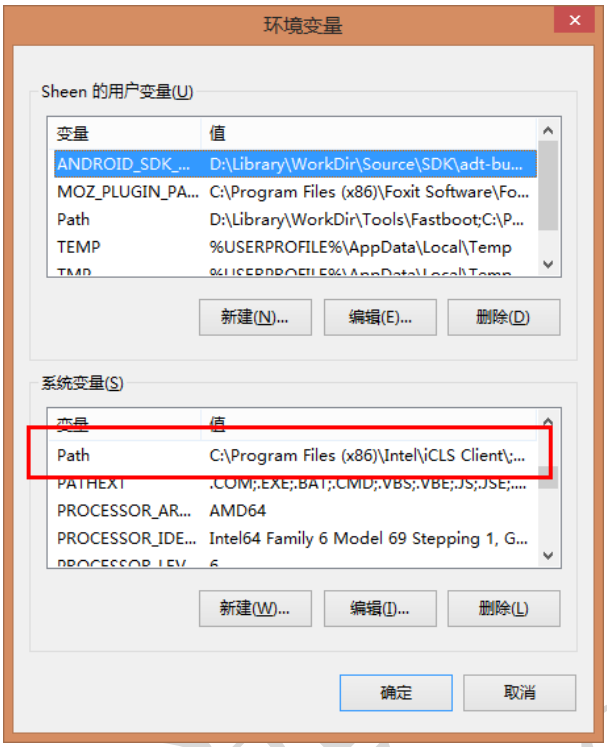
拷贝【华清远见-CORTEXA9 资料\工具软件\Windows\Fastboot\Fastboot】目录到 D 盘下。



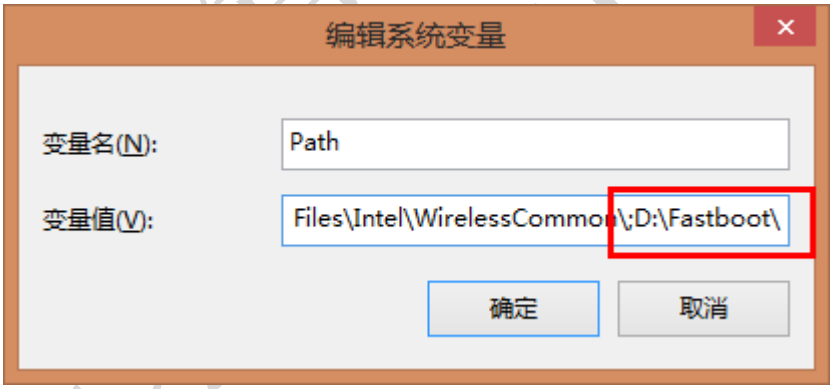
打开【计算机属性】下得而【系统属性】，查看【高级】，选择环境变量。



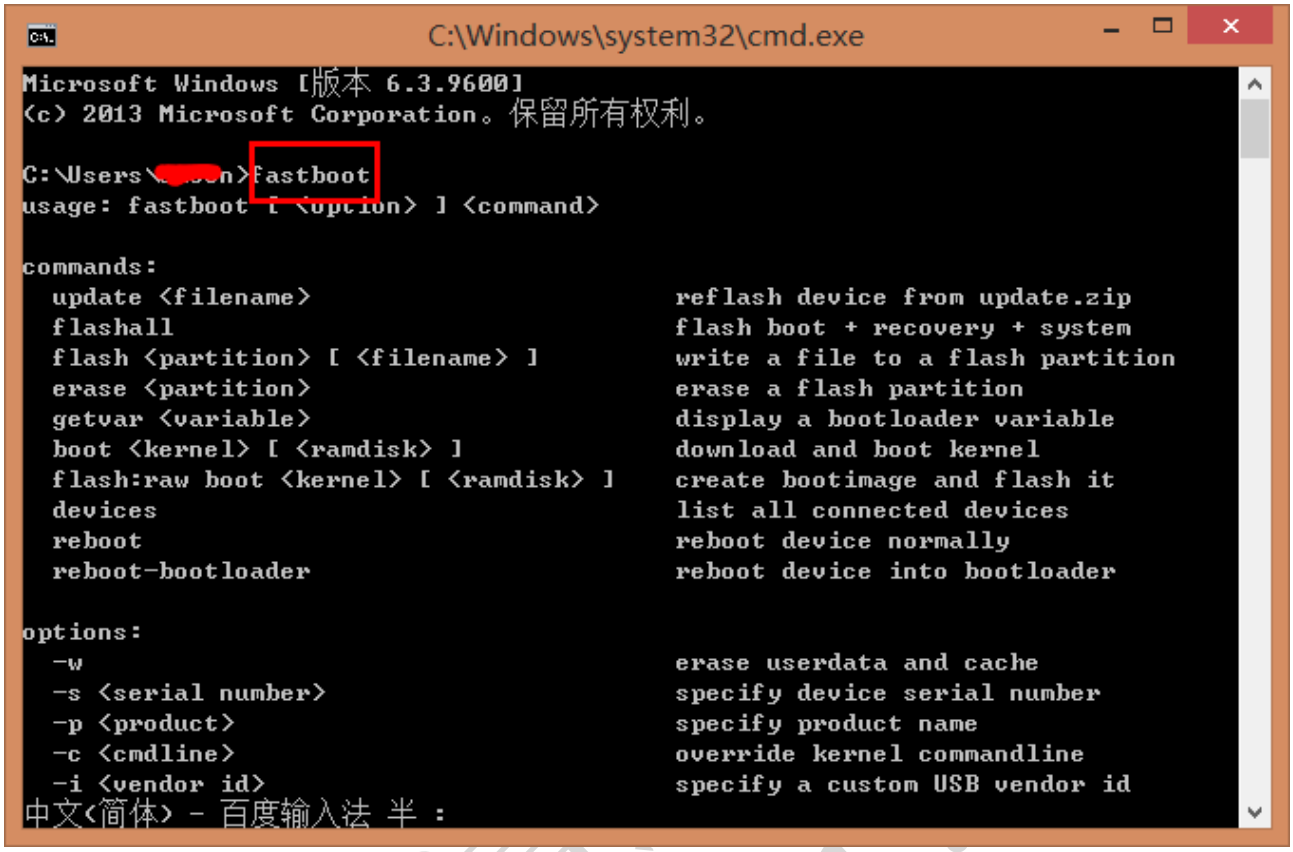
编译【系统变量】的【Path】项。



在【变量值】最后添加【;D:\Fastboot\】(注意开始的分号)。



点击【确定】，打开【cmd】，输入【fastboot】测试环境变量是否添加成功。



如上图所示即添加环境变量成功。

5.4.2 安装 Fastboot 驱动

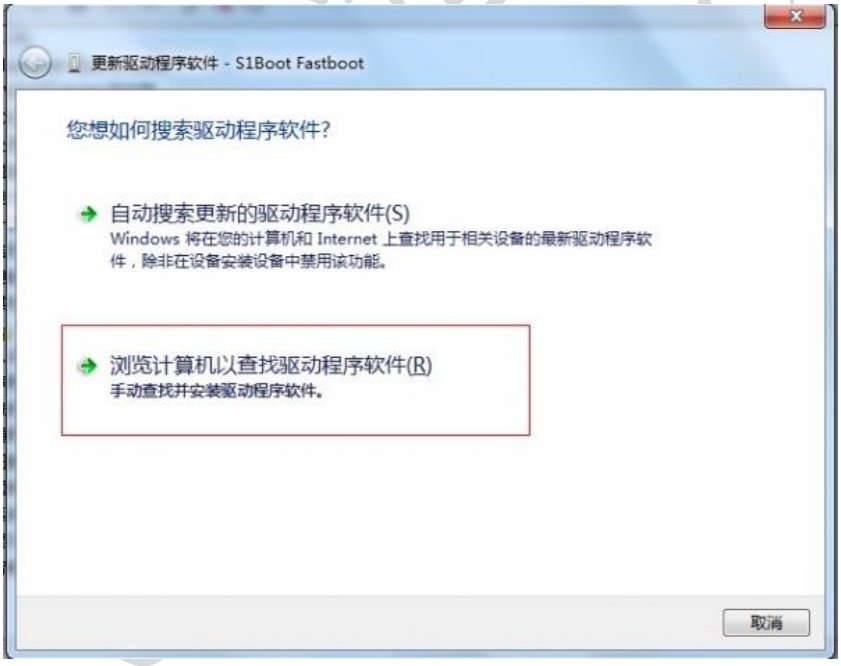
第一次使用 Fastboot 需要安装驱动，驱动位置在【华清远见-CORTEXA9 资料\工具软件\Windows\Fastboot\Fastboot 驱动】下。连接开发板串口和 USB 口并启动安卓系统。

此时，如果系统安装了驱动，通过 USB 线连接开发板和电脑，重启开发板，保证开发板 U-Boot 版本为 2010.03，在 u-boot 终端输入【fastboot】如下图所示。

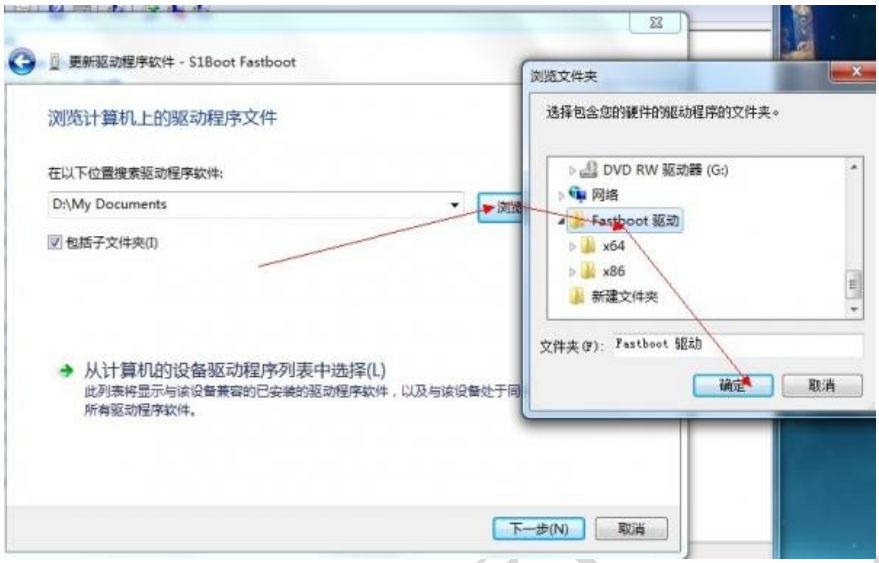
```
FS4412 # fastboot
[Partition table on MoviNAND]
ptn 0 name='bootloader' start=0x0 len=N/A (use hard-coded info. (cmd: movi))
ptn 1 name='kernel' start=N/A len=N/A (use hard-coded info. (cmd: movi))
ptn 2 name='ramdisk' start=N/A len=0x300000(~3072KB) (use hard-coded info. (cmd: movi))
ptn 3 name='Recovery' start=N/A len=0x600000(~6144KB) (use hard-coded info. (cmd: movi))
ptn 4 name='system' start=0x1000000 len=0x12C00000(~307200KB)
ptn 5 name='userdata' start=0x13C00000 len=0x40000000(~1048576KB)
ptn 6 name='cache' start=0x53C00000 len=0x12C00000(~307200KB)
ptn 7 name='fat' start=0x66800000 len=0x3CC00000(~995328KB)
```

此时，如果系统安装了驱动，在设备管理器中应该如下显示：

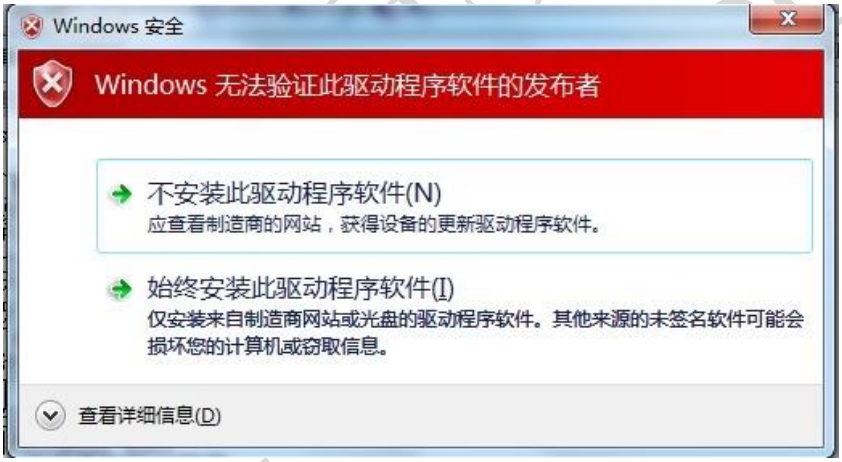
如没有安装驱动，需要手动安装，通过 USB 线连接开发板和电脑，重启开发板，保证开发板 U-Boot 版本为 2010.03，在 u-boot 终端输入【fastboot】，此时，电脑会识别到新设备。在设备管理器右击未识别的设备【Android1.0】选择“更新驱动程序软件”弹出如下窗口，点击“浏览计算机以查找驱动程序软件”：



点击浏览选择驱动所在文件夹，此处选择 “usb_driver_r03-windows”并勾选“包括子文件夹”即可。



选择“始终安装此驱动程序软件”



安装成功后，选择关闭

注：XP 下也是如此操作，只不过某些地方名称不一样！第五步如果确定按钮是灰色的话，选择到 X86 里面的 usbwin 就行了，如果是 64 位 XP 就选择 X64 的。WIIN8 下要关闭强制驱动签名才能安装上驱动。

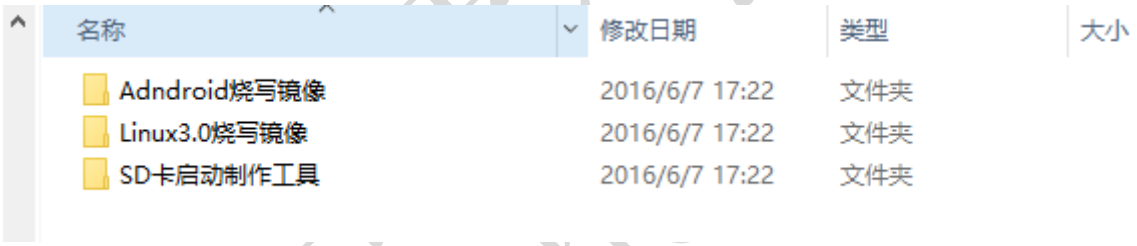
5.5 烧写系统

打开开发板，保证开发板 U-Boot 版本为 2010.03，终端输入【fdisk –c 0】对 eMMC 分区，然后输入“fastboot”，如下图所示。

```
# fdisk  –c  0          // 对 eMMC 分区
# fastboot
```

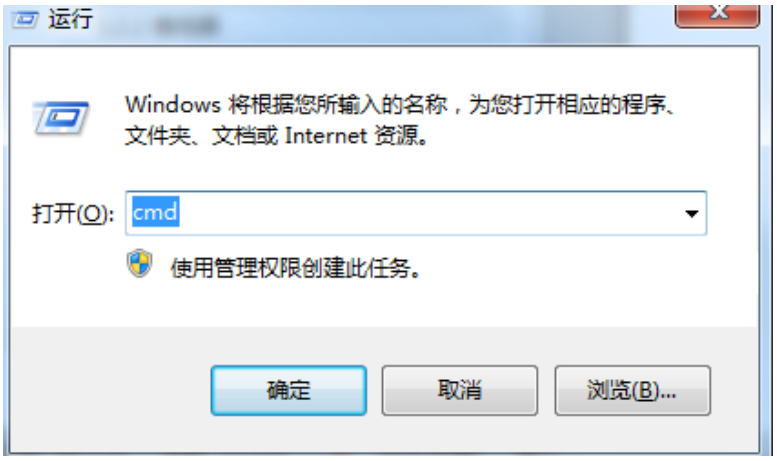
```
Hit any key to stop autoboot:  0
FS4412 # fdisk -c 0
.fdisk is completed

partion #    size(MB)     block start #    block count    partition_Id
   1         13260          3358720        27156480          0x0C
   2           300            32768          614400          0x83
   3          1024           647168         2097152          0x83
   4           300          2744320          614400          0x83
FS4412 # fastboot
[Partition table on MoviNAND]
ptn 0 name='bootloader' start=0x0 len=N/A (use hard-coded info. (cmd: movi))
ptn 1 name='kernel' start=N/A len=N/A (use hard-coded info. (cmd: movi))
ptn 2 name='ramdisk' start=N/A len=0x300000(~3072KB) (use hard-coded info. (cmd: movi))
ptn 3 name='Recovery' start=N/A len=0x600000(~6144KB) (use hard-coded info. (cmd: movi))
ptn 4 name='system' start=0x1000000 len=0x12C00000(~307200KB)
ptn 5 name='userdata' start=0x13C00000 len=0x40000000(~1048576KB)
ptn 6 name='cache' start=0x53C00000 len=0x12C00000(~307200KB)
ptn 7 name='fat' start=0x66800000 len=0x3CC00000(~995328KB)
Insert a OTG cable into the connector!
```

打开【华清远见-CORTEXA9 资料\烧写镜像\】目录。

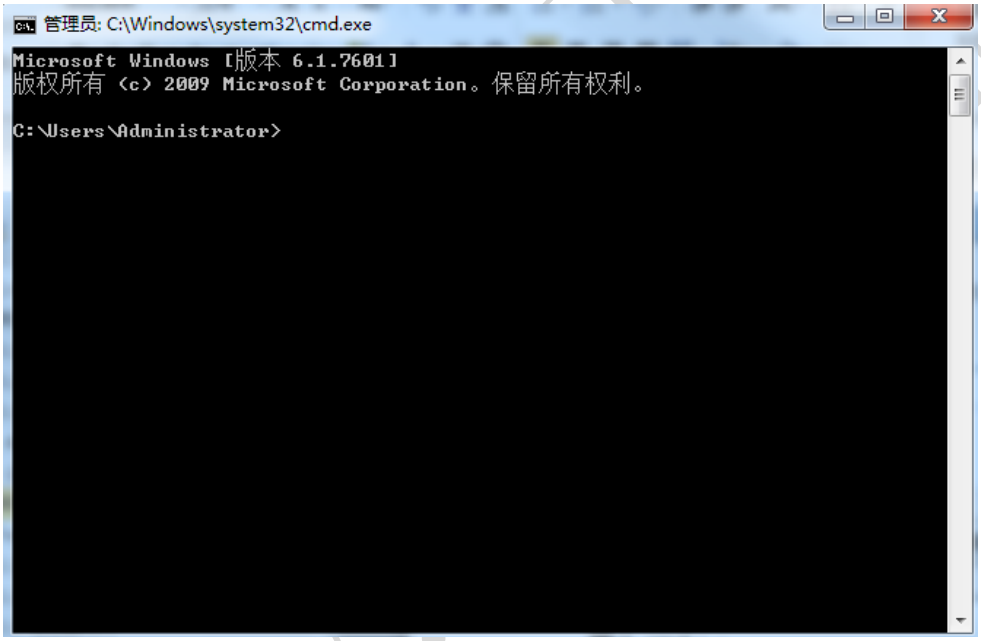


选择需要使用的镜像，如“Android 烧写镜像”的，打开目录。将该目录拷贝到【E:\ShareVMare】文件下面

在 window 系统下面进行烧写，点击 window+R：



输入 cmd：



进入烧写镜像的文件夹里面进行烧写，这里放在【E:\ ShareVMare】件夹下：

输入 E：+回车

输入 cd ShareVMare/+回车

在 fastboot 环境搭建好的情况下进行如下操作，fastboot 环境搭建请参考:“第 5 章 5.4 环境配置”

烧写 uboot

```
Fastboot flash bootloader u-boot-fs4412.bin
```

```
E:\ShareVMware>fastboot flash bootloader u-boot-fs4412.bin
sending 'bootloader' (514 KB)...
OKAY [  0.072s]
writing 'bootloader'...
OKAY [  0.179s]
finished. total time: 0.253s
```

烧写 kerne

```
Fastboot flash kernel zImage
```

```
E:\ShareVMware>fastboot flash kernel zImage
sending 'kernel' (3693 KB)...
OKAY [  0.449s]
writing 'kernel'...
OKAY [  0.267s]
finished. total time: 0.719s
```

烧写 ramdisk

```
Fastboot flash ramdisk ramdisk-uboot.img
```

```
E:\ShareVMare>fastboot flash ramdisk ramdisk-uboot.img
sending 'ramdisk' (900 KB)...
OKAY [  0.117s]
writing 'ramdisk'...
OKAY [  0.276s]
finished. total time: 0.395s
```

烧写文件系统

```
Fastboot flash system system.img
```

```
E:\ShareVMware>fastboot flash system system.img
sending 'system' (65164 KB)...
OKAY [  7.738s]
writing 'system'...
OKAY [  9.748s]
finished. total time: 17.487s
```

烧写 userdata

```
Fastboot flash userdata userdata.img
```

```
E:\ShareVMare>fastboot flash userdata userdata.img
sending 'userdata' (113768 KB)...
OKAY [ 13.506s]
writing 'userdata'...
OKAY [  6.390s]
finished. total time: 19.898s
```

到这里为止系统所需要的镜像全部烧写完毕，重启开发板并在倒数计时的时候按任意键，并使用在

putty 里面输入【print】命令查看当前的环境变量。

```
FS4412 # print
bootdelay=5
baudrate=115200
ethaddr=11:22:33:44:55:66
ethact=dm9000
ipaddr=192.168.1.192
serverip=192.168.1.133
gatewayip=192.168.1.1
bootargs=root=/dev/mmcblk0p2 rootfstype=ext4 init=linuxrc console=ttySAC2,115200
bootcmd=movi read kernel 40008000;bootm 40008000
stdin=serial
stdout=serial
stderr=serial

Environment size: 305/16380 bytes
FS4412 #
```

我们需要设置 bootcmd 参数，命令如下：(该命令为一行命令，每一个单词数据之间均有空格)

```
    setenv bootcmd movi read kernel 40008000\;movi read rootfs 40d00000 100000\;bootm 40008000
40d00000

    setenv bootargs

    save
```

重新启动开发板，会进入到 Android 系统，就可以在界面操作 Android 的应用。

第 6 章　修改环境变量

u-boot 的环境变量是使用 u-boot 的关键，其中大部分是 u-boot 自己定义的，大部分情况下我们只需要使用这些默认支持的命令就可以了，用户更改这些名字会出现错误，下面的表中我们列出了一些常用的环境变量：

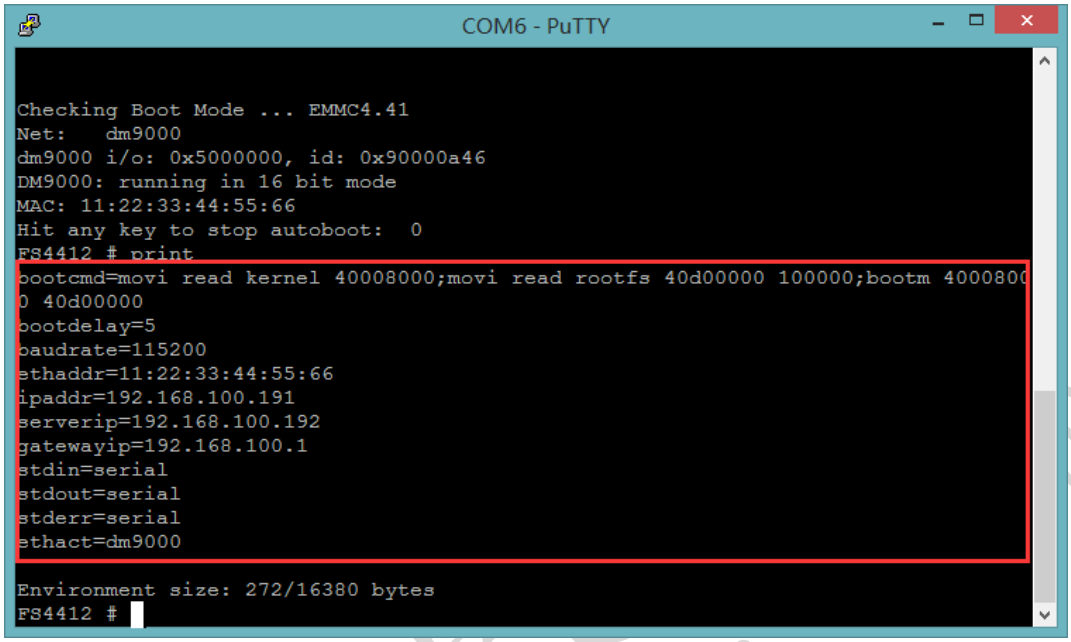
环境变量	描述
bootdelay	执行自动启动的等候秒数
baudrate	串口控制台的波特率
netmask	以太网接口的掩码
ethaddr	以太网卡的网卡物理地址
bootargs	传递给内核的启动参数
bootcmd	自动启动时执行的命令
serverip	服务器端的 ip 地址
ipaddr	本地 ip 地址

如果用户在使用过程中使做过移植的操作或其它操作，通过 setenv 命令修改过 uboot 的环境变量，即使是用户重新烧写 uboot，原来手动修改过的某些环境变量也不会使用默认的环境变量，所以这个时候用户需要做一些操作保证环境变量的正确性。

6.1　eMMC 启动 Android 环境变量

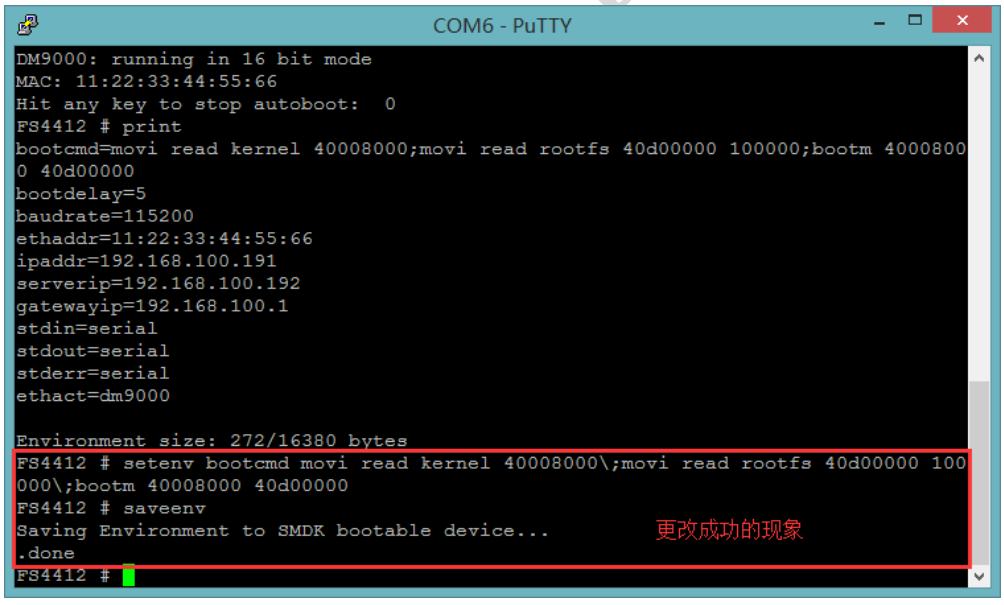
使用 print 查看当前的环境变量，如果用户在使用过程中修改了其中的参数，可以按照如下的值来设定：(使用 setenv 修改环境变量的值，修改完成之后使用 saveenv 保存环境变量)

使用 **print** 查看当前的环境变量，下图是使用 eMMC 启动 Android 系统时的默认参数，android 系统启动时不需要 bootargs。



如果用户在使用过程中修改了其中的参数，可以按照如下的值来设定：**(使用 setenv 修改环境变量的值，修改完成之后使用 saveenv 保存环境变量)**

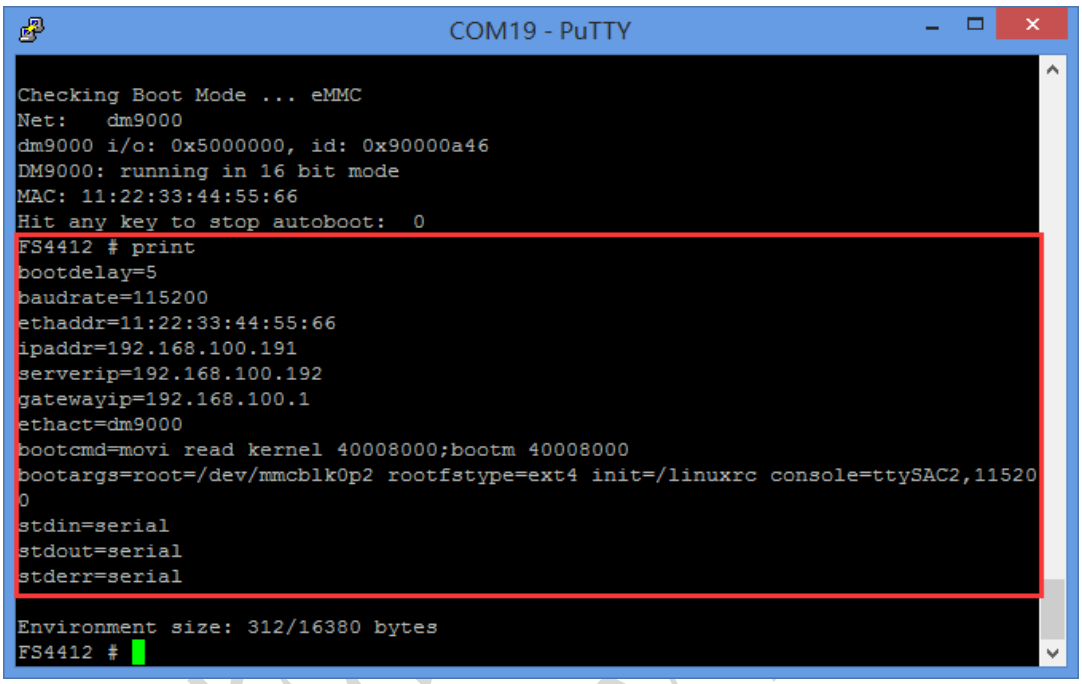
```
setenv bootcmd movi read kernel 40008000\;movi read rootfs 40d00000 100000\;bootm 40008000 40d00000
setenv bootargs
saveenv
```



上边的命令一定需要保证不能够出现错误，否则会出现系统引导失败 !!!!!!!!!!!

6.2 eMMc 启动 Linux 环境变量

使用 print 查看当前的环境变量，如果用户在使用过程中修改了其中的参数，可以按照如下的值来设定：(使用 setenv 修改环境变量的值，修改完成之后使用 saveenv 保存环境变量)



设置 bootcmd 参数，在命令行输入如下命令：

```
setenv bootcmd movi read kernel 40008000\;bootm 40008000
saveenv
```

设置 bootargs 参数，在命令行输入如下命令：

```
setenv bootargs root=/dev/mmcblk0p2 rootfstype=ext4 console=ttySAC2,115200 init=/linuxrc
saveenv
```

上边的命令一定需要保证不能够出现错误，否则会出现系统引导失败 !!!!!!!!!!!

6.3 总结

环境变量是系统能否成功启动的重要标志，尤其是 bootcmd 和 bootargs，用户可以通过其他资料了解两个命令的具体信息，前两节只给出了两种启动方式启动不同系统的默认环境变量，如果用户遇到系统无

法启动的现象，可以尝试首先使用 print 查看当前的环境变量值，如果与上图不符，则手动修改一下。这里不介绍直接擦除环境变量的办法，用户有需要的话可以自己查找相关的资料。

第 7 章　内核和文件系统说明

本章节根据用户使用不同的介质，对内核读取位置和文件挂载方式的参数进行介绍和说明。

7.1　内核启动方式

我们可以在 uboot 中使用不同的命令对不同介质中存储的内核进行读取和写入。在开发中我们一般把内核的读取和内核的启动写入到 uboot 的【bootcmd】参数中，这些 uboot 在上电后，会在几秒钟的倒计时后自动执行启动内核等相应的步骤，大大提高了效率。

设置 bootcmd 参数的命令如下所示，启动参数可以为多个命令，命令与命令之间用【\;】分割。一般最后一个命令总为启动内核，如【bootm 20008000】。

```
# setenv bootcmd cmd1 \; cmd2 \; cmd3
```

7.1.1　MMC 介质

对于 MMC 介质，uboot 提供了 movi read 和 movi write 两组命令对其进行读写，kernel 表示内核保存的分区，40008000 这个参数为读入的内存地址。bootm 是启动内核命令，40008000 表示内核在 40008000 这个内存地址。

```
# movi read kernel 40008000

# movi write kernel 40008000
```

7.1.2　TFTP

在嵌入式开发的过程中，底层开发者需要调试内核或者 ramdisk 等比较小型且独立的文件，这里提供一种 TFTP 服务的方式 ,可以让开发板从开发主机下载的方式 ,运行或者调试系统。40008000 为内存地址；在内核调试中，filename 一般为 zImage 或者 uImage。详细的 tftp 环境配置和使用参见《嵌入式 Linux 移植驱动及应用开发》的内容，这里不再赘述。

```
# tftp 40008000 filename
```

7.2 文件系统挂载方式

在 uboot 中，bootloader 向内核传递参数使用【bootargs】这个参数来实现的，一般 bootargs 中包含了文件系统的位置和格式、第一个启动的文件、console 调试终端信息和其他信息等属性。

设置 bootargs 参数的命令如下所示，devicename 表示文件系统所在的位置，filename 表示此文件系统中的哪一个文件作为第一个启动，ttySACx 的 x 为调试串口号，一般为 0,1,2,3 等，baudrate 表示调试串口的波特率，一般为 115200，etc…表示其他参数，比如在 nfs 网络文件系统中还要有 IP 地址等信息。

```
# setenv bootargs root=devicename init=/filename console=ttySACx,baudrate etc…
```

7.2.1 MMC 设备挂载 ext4 文件系统

现在很多嵌入式设备，尤其是手机，用 eMMC 代替了原来的 Nand Flash，虽然本质还是 Nand Flash 作为存储介质，但因为集成了主控，所以有了更为普遍的操作形式，我们使用的 SD 卡也是应用了此种形式，即为 MMC 设备。对于 MMC 来言，ext 文件系统是最常用的，我们一般使用下面的参数作为文件系统在 MMC 而且文件系统类型为 ext4 的时候。

```
# setenv bootargs root=/dev/mmcblk0p2 rootfstype=ext4 init=/init console=ttySAC2,115200
```

7.2.2 NFS 挂载网络文件系统

NFS 方式是开发板通过 NFS 挂载放在主机（PC）上的根文件系统。此时在主机在文件系统中进行的操作同步反映在开发板上；反之，在开发板上进行的操作同步反映在主机中的根文件系统上。实际工作中，我们经常使用 NFS 方式挂载系统，这种方式对于系统的调试非常方便。我们一般使用下面的参数作为文件系统为 nfs 的时候。（注意，此处环境的具体搭建方式参见《嵌入式 Linux 移植驱动及应用开发》的内容，

这里不再赘述。)

```
# setenv bootargs root=nfs nfsroot=192.168.100.192:/source/rootfs ip=192.168.100.191:::::eth0:off
init=/linuxrc console=ttySAC2,115200        //此参数为一行代码，尽量不要使用复制到串口终端
```

我们为 Linux3.0 内核提供了 1 个 Linux 文件系统供用户挂载（用户也可以使用自己的文件系统），位置【华清远见-CORTEXA9 资料:/烧写镜像/Linux3.0 烧写镜像/rootfs.tar.xz】中。