目　录

物联网科研演示平台

智能交通

智能楼宇

智能交通

第一章 智能交通实验指导书

1.1 大唐移动 UI-IOT-ITS 智能交通实验指导书

UI-IOT-ITS 智能交通物联网实训系统实验指导书

1、 系统概述

1.1、 智能交通的概念

智能交通系统(IntelligentTransportationSystem，简称 ITS)是未来交通系统的发展方向，它是将先进的信息技术、数据通讯传输技术、电子传感技术、控制技术及计算机技术等有效地集成运用于整个地面交通管理系统而建立的一种在大范围内、全方位发挥作用的，实时、准确、高效的综合交通运输管理系统。

ITS 可以有效地利用现有交通设施、减少交通负荷和环境污染、保证交通安全、提高运输效率，因而，日益受到各国的重视。

中国物联网校企联盟认为，智能交通的发展跟物联网的发展是离不开的，只有物联网技术概念的不断发展，智能交通系统才能越来越完善。智能交通是交通的物联化体现。

21 世纪将是公路交通智能化的世纪，人们将要采用的智能交通系统，是一种先进的一体化交通综合管理系统。**在该系统中，车辆靠自己的智能在道路上自由行驶，公路靠自身的智能将交通流量调整至最佳状态，借助于这个系统，管理人员对道路、车辆的行踪将掌握得一清二楚。**

智能交通：智能交通是一个基于现代电子信息技术面向交通运输的服务系统。它的突出特点是以信息的收集、处理、发布、交换、分析、利用为主线，为交通参与者提供多样性的服务。

1.2、 系统组成

智能交通物联网实训系统由以下几个主要功能单元组成：（如下图）

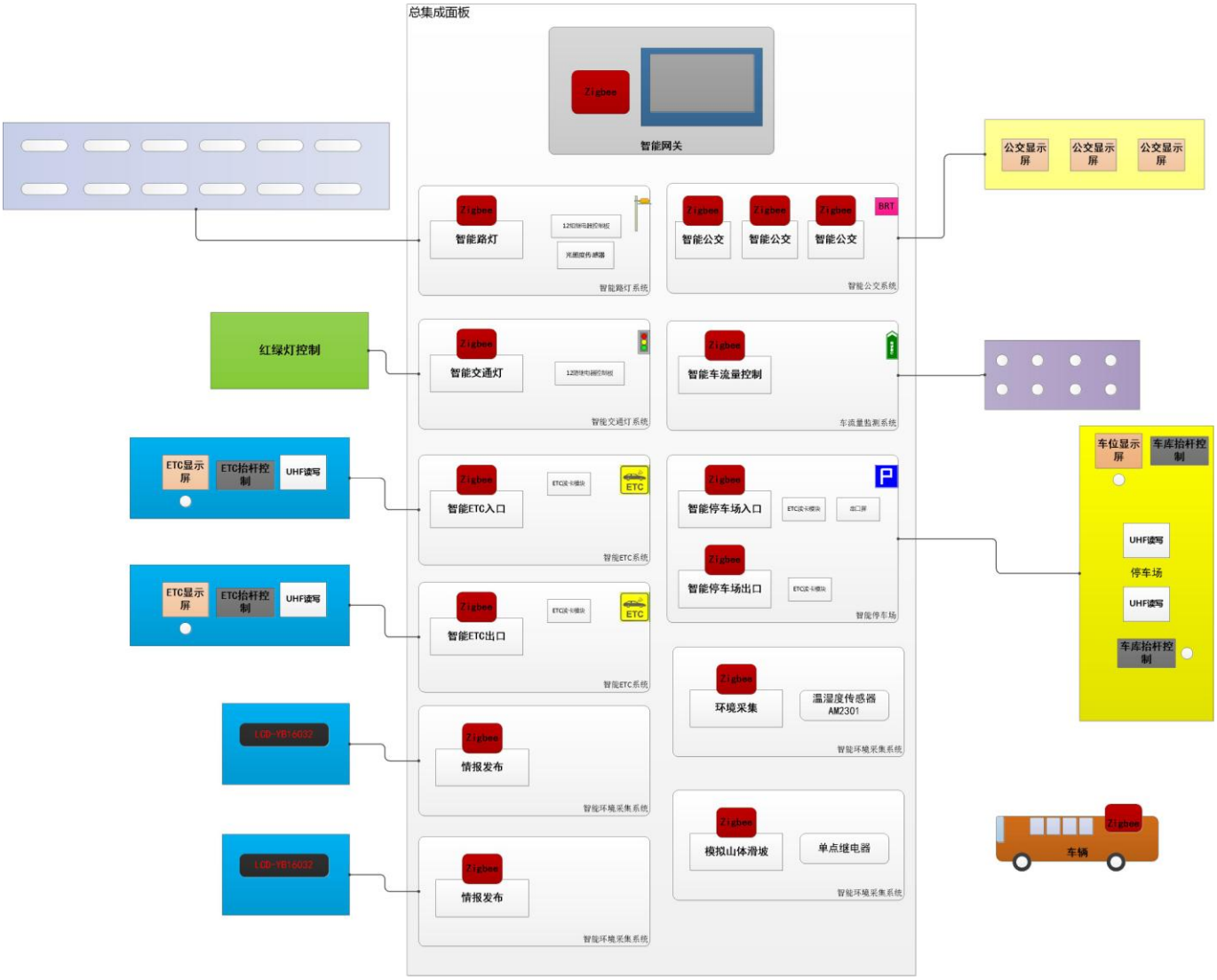


图 1.1 系统框图

1. 智能交通灯
2. 智能路灯
3. 智能公交
4. 智能车流量
5. 智能停车场
6. 智能 ETC
7. 智能网关/调度中心

每个单元由一个通用控制节点进行控制，该节点采用 STM32F103 作为主控芯片，引出了 GPIO 端口和 2 个串口，并集成了 Zigbee 无线通信模块。这些节点通通集成在沙盘下方的抽屉上，如下图。模块与抽屉可以手动连接线路，这种开放式的设计可以方便学生进行二次开发学习。

图 1.2 沙盘集成控制抽屉

2、 智能车系统

2.1、 智能小车简介

智能小车 UI-SmartCar 为我司重点研发攻克的项目，高度集成，包括磁导航传感器、RFID 读卡器、红外传感器、zigbee 通信模块等，下面两张图片为智能小车的底盘和架构框图。

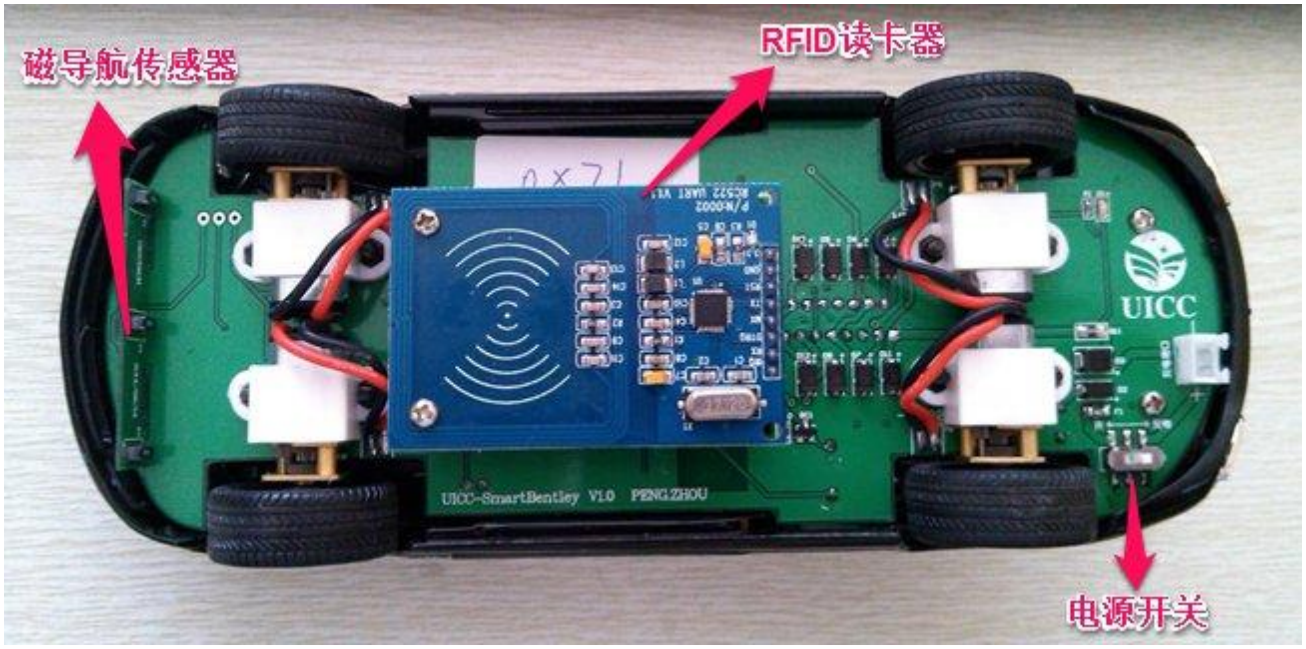


图 2.1 小车底盘

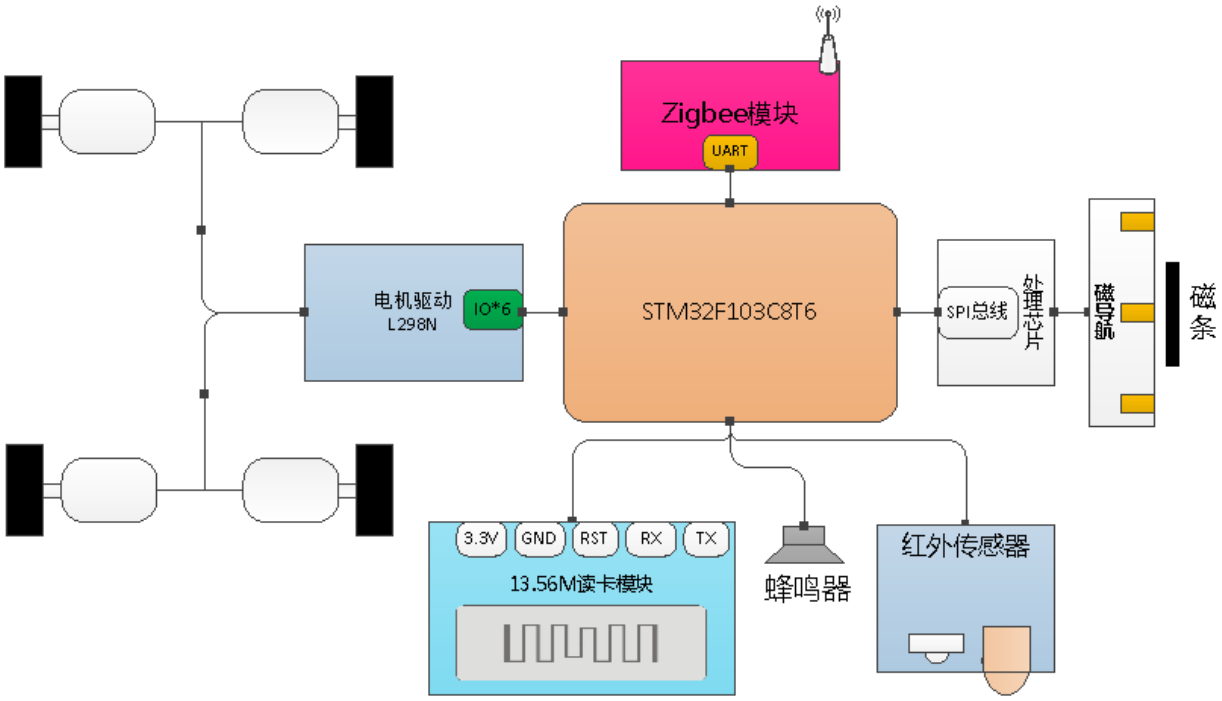


图 2.2 智能小车架构图

小车的程序和电路全部开源，可以供用户进行二次开发。

2.2、 Zigbee 通信实验

2.2.1、 Zigbee 简介

ZigBee 是基于 IEEE802.15.4 标准的低功耗个域网协议。根据这个协议规定的技术是一种短距离、低功耗的无线通信技术。这一名称来源于蜜蜂的八字舞，由于蜜蜂(bee)是靠飞翔和“嗡嗡”(zig)地抖动翅膀的“舞蹈”来与同伴传递花粉所在方位信息，也就是说蜜蜂依靠这样的方式构成了群体中的通信网络。其特点是近距离、低复杂度、自组织、低功耗、高数据速率。主要适合用于自动控制和远程控制领域，可以嵌入各种设备。简而言之，ZigBee 就是一种便宜的，低功耗的近距离无线组网通讯技术。

ZigBee 网络主要特点是低功耗、低成本、低速率、支持大量节点、支持多种网络拓扑、低复杂度、快速、可靠、安全。ZigBee 网络中的设备可分为协调器(Coordinator)、汇聚节点(Router)、传感器节点(EndDevice)等三种角色。

2.2.2、 动手操作

1. 使用 USB 转 RS232 线连接无线数据采集节点，如下图：
2. 查看电脑中相应的串口号，打开 commix10.exe 串口工具。

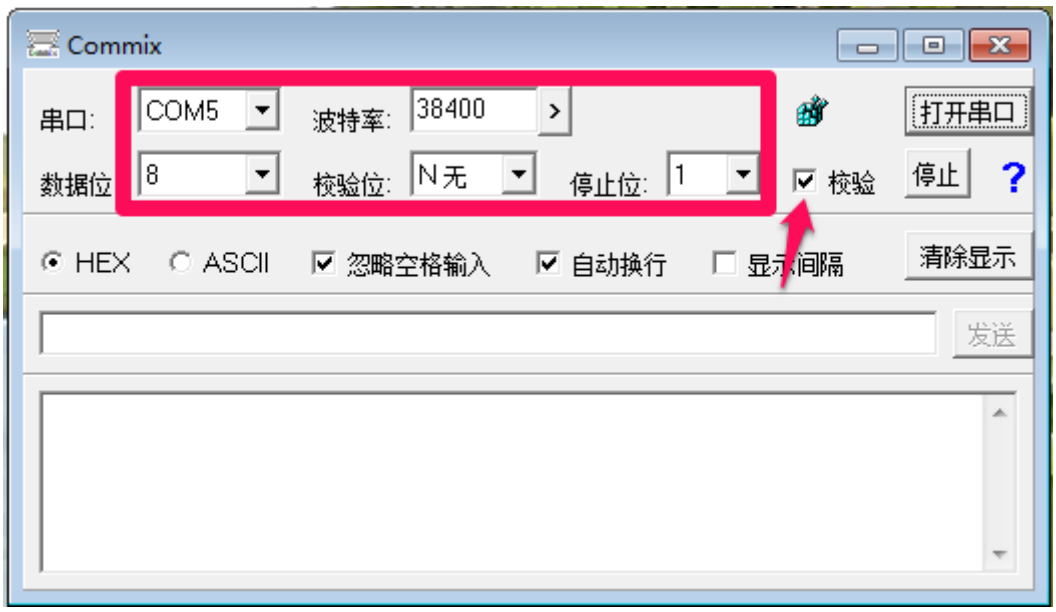


图 2.3 串口调试界面

3. 点击校验前门的对勾，按照下图设置参数。（自动计算每帧数据的校验位）设置完后打开串口。

图 2.4 串口调试界面

4. 单独给 1 号小车上电，观察串口接收到的数据是否跟下图一样。按照“智能交通通信协议.doc”解析该条指令的含义。

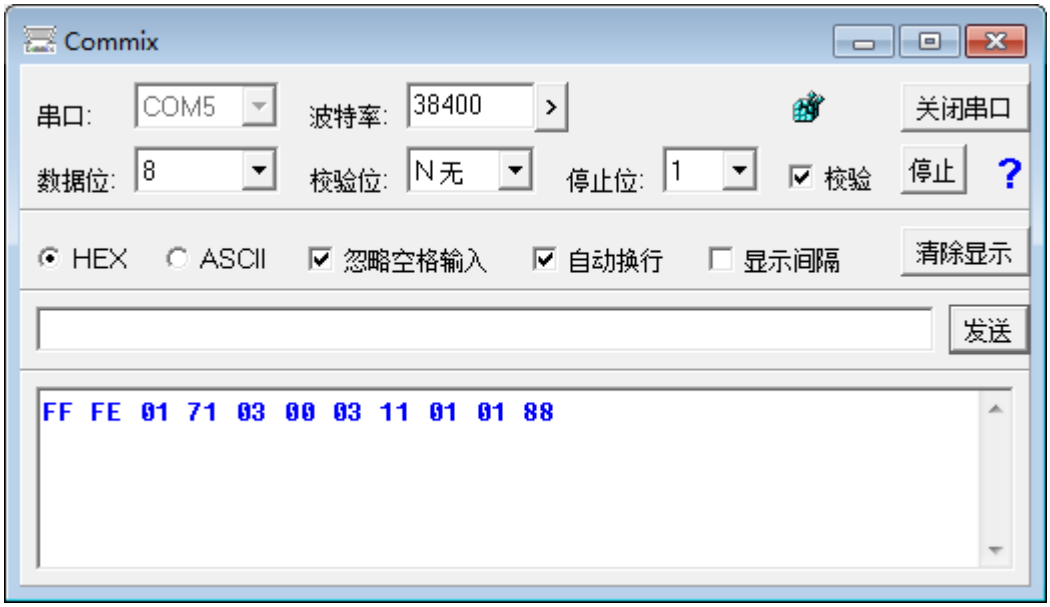


图 2.5 串口调试界面

5. 串口发送“FF FE 71 01 01 02 11”，查询小车信息。如下图

图 2.6 串口调试界面

6. 可按照通信协议，发送其他指令以进一步熟悉上下位机之间的通信协议。

2.3、 用智能小车 RFID 读写卡实验

1. 小车上电，用串口调试助手发“FF FE 71 01 06 01 17 00 01 41 32 03”，指令含义为将 41 32 03 即位置标记“A2-3”写入标签中，小车会回执“FF FE 01 71 01 00 01 17 88”，如下图，用附件带的 RFID 标签放在小车的 RFID 模块天线区域。

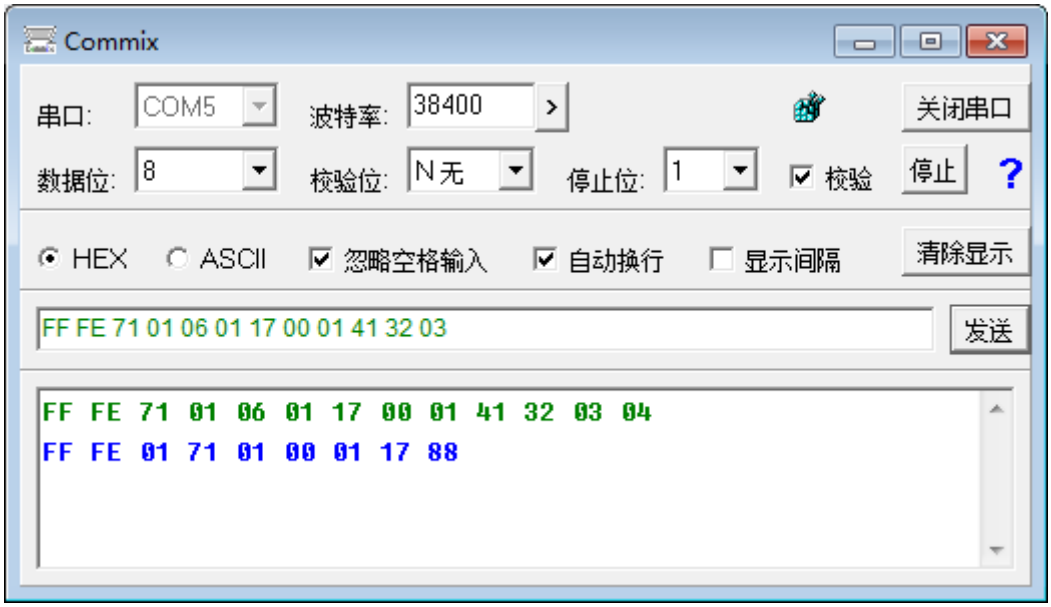


图 2.7 串口调试界面

2. 小车发出一声“嘀”长音，串口返回“FF FE 01 71 04 00 03 14 41 32 03 00”，这里注意，返回的数据 41 32 03 为写卡完成后读取数据，若数据跟写入数据一致，则代表写卡成功。

图 2.8 串口调试界面

2.4、 智能小车路径设置实验

小车上电，放入车位中，小车停车并发出“嘀”的一声，代表小车停车成功。

用手拿起小车，用串口连接协调器，打开串口调试助手，按照“xxx”节设置好串口收发参数。

用电脑发指令“FF FE 71 01 02 01 12 05 SUM”

小车的路径设置通过一个 5 字节二进制编码实现（参考智能交通通信协议），每一个二进制位代表了一次小车检测到关键点的决策，0 代表右转，1 代表直行。每当小车从车库出来时由上位机发送给小车，要实现小车按下图的 2 号路径（浅蓝色）行走，上位机给小车发送 FF FE 71 01 06 01 15 04 B0 00 00 00 SUM

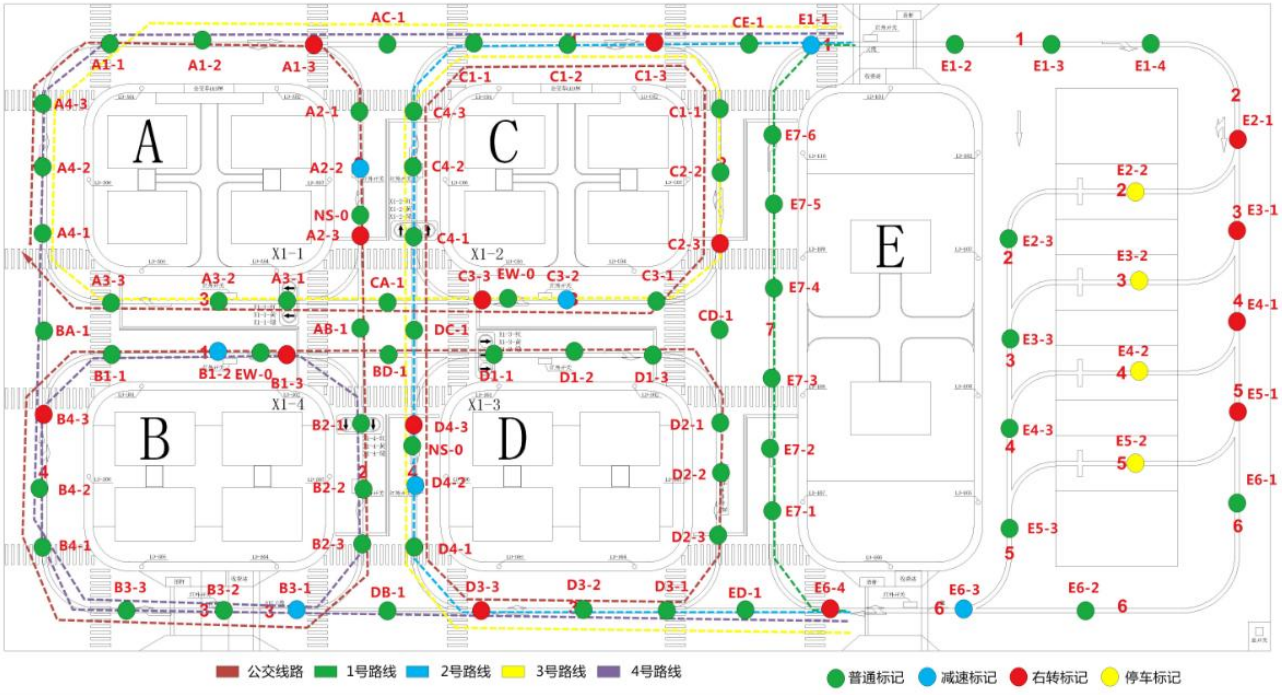


图 2.9 路径规划示意图

上图中，小圆点代表路基下方贴的 RFID 标签，用来标识所处沙盘的位置，绿色圆点代表普通位置标记，红色点表示既可以右转右可以直行的点，蓝色点代表需要减速的点。

小车行进过程中，若监测到红色的点，即检索上位机下发的路径编码，0 则右转，1 则继续直行，例如上面发的指令中， 04 B0 00 00 00 为路径编码，04 代表该路径编码有四个有效位，B0 00 00 00 为实际编码，转换为二进制即为 10110000 00000000 00000000 00000000，由于只有四个有效位，故只取高四位为路径编码。将该路径执行过程分解如下：

1. 小车出库，监测到第 1 个关键点“E6-4”，该位路径编码为“1”，直行，路径编码整体左移一位。

2. 监测到第 2 个关键点，“D3-3”，该位路径编码为“0”，右转，路径编码整体左移一位。

3. 监测到第 3 个关键点，“D4-3”，该位路径编码为“1”，直行，路径编码整体左移一位。

4. 监测到第 4 个关键点，“C1-3”，该位路径编码为“1”，直行，路径编码整体左移一位，小车入库，整个路径直行完毕。

2.5、 智能小车自主导航

小车上电，放于道路上方，观察小车前方的 MicroMA03 磁导航传感器，寻找磁条待 MicroMA03 磁导航传感器上的红色 LED 亮起，小车便会自主循迹导航，在智能网关 ITS 软件不打开的情况下，默认小车为漫游状态，最终会停入停车场的默认车位。

2.6、 WiFi 摄像头使用说明

1. 电脑上安装附带资料“Ai-Ball Multiview”文件夹中的 Setup.exe 软件；

2. 确保路由器上电，打开小车上的摄像头，如图所示，将开关拨到“ON”一档；（关于摄像头更多的详细资料，请看附带资料中的“AiBall 使用手册.doc”）

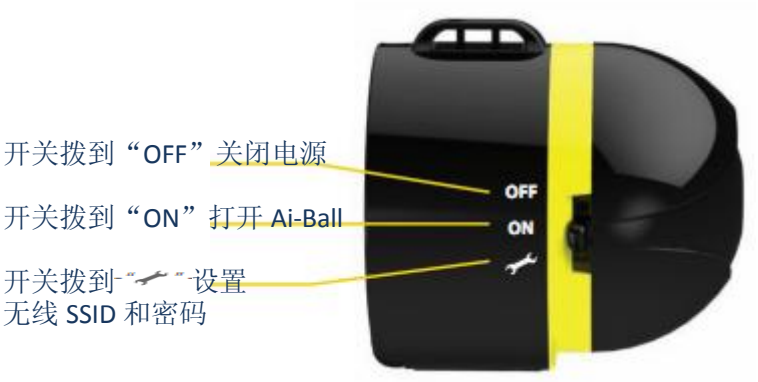


图 2.10wifi 摄像头

3. 使用笔记本连接我们附带的路由器，SSID：Creator Password：uicc99866

4. 打开“Ai-Ball Multiview”软件，界面如下图：

图 2.11 Ai-Ball Multiview 界面

5. 点击  图标，搜索局域网中连接的 wifi 摄像头（wifi 摄像头均已设置好 IP 地址为 192.168.1.151~157），如果有摄像头在线，就会在界面中显示摄像头获取的图像，点击其中一个图像既可以放大查看或录制视频，点击“File->Save video files to”设置录制视频的保存位置。

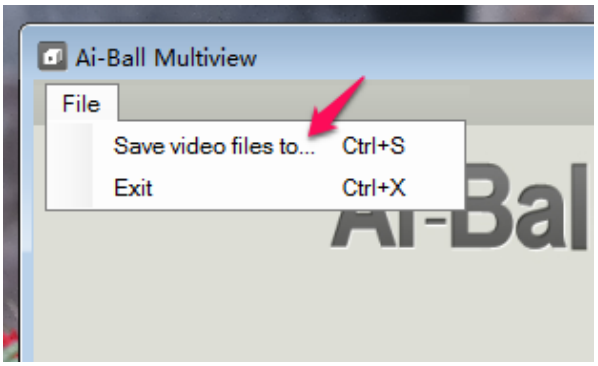


图 2.12 视频保存路径设置

3、 功能模块实验

智能交通每个单元控制的节点都是采用下图中的通用控制节点

图 3.1 智能通用节点

该节点充分利用 STM32F103C8T6 的片内资源，引出 12 个 GPIO、1 个 TTL 串口、2 个 RS232 串口、红外接口、舵机 PWM 接口、温湿度单总线接口以及 3.3V 和 5V 供电接口，以方便不同种类外设的连接。其系统设计框图如下：

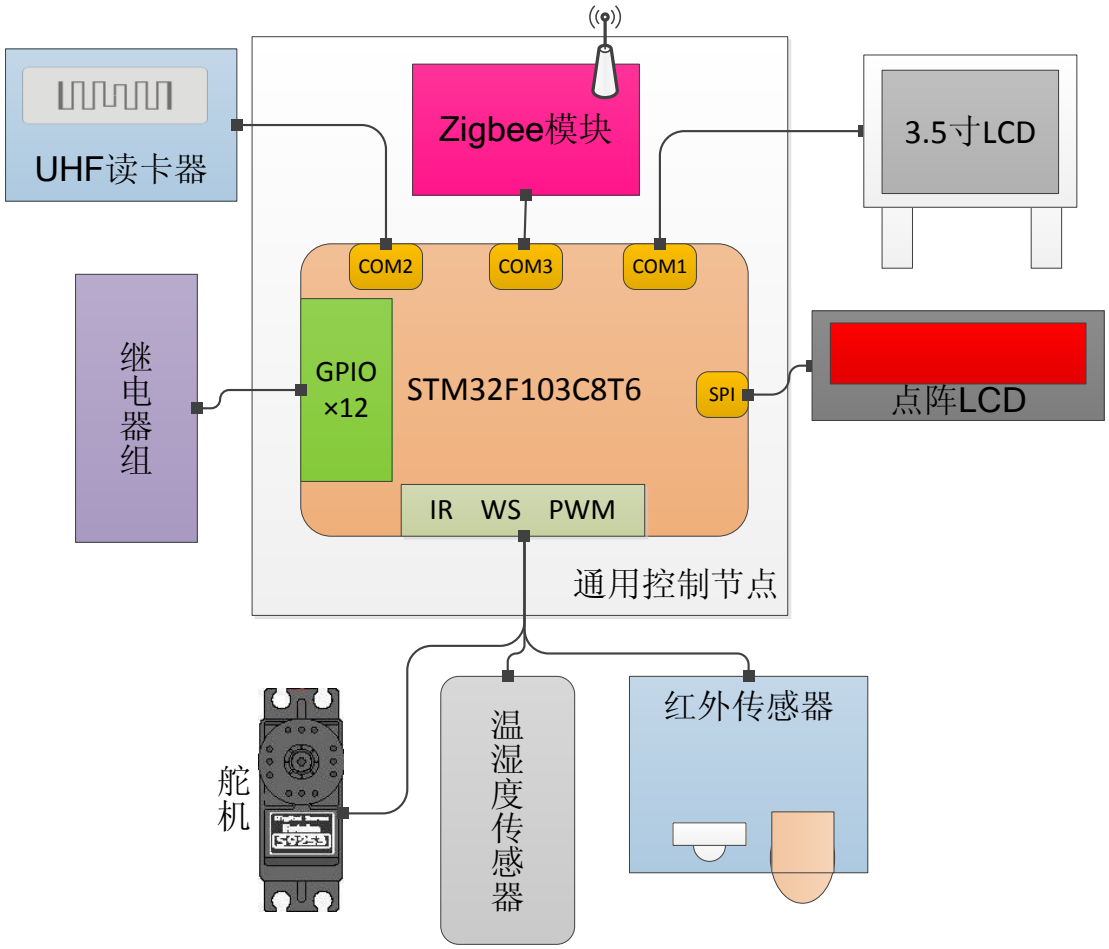


图 3.2 通用节点设计框图

对于不同的节点，只需要配置不同的外设就可以。下表为不同节点的配置情况。

表 3.1

	Zigbee 模块	3.5 寸 LCD	点阵 LCD	UHF 读卡器	舵机	温湿度传感器	红外传感器	继电器
交通灯模块	√							√
路灯模块	√							√
车流量模块	√						4-8	
ETC 模块	√	√		√	√		√	
停车场模块	√	√		√	√		√	
公交站模块	√	√						
情报板模块	√		√					
环境监测模块	√					√		

3.1、 交通灯模块

图 3.3 交通灯示意图

功能介绍

交通灯模块控制沙盘十字路口的红绿灯状态，有三种模式可供选择，分别为固定模式、公交优先模式、动态调整模式。特点如下：

1.固定模式下，两个方向红绿灯固定间隔 10S 变化一次，为现实道路中最普通的情况。

2.公交优先模式下，当公交车快要到达路口，红绿灯优先为其变绿，保障公共交通行驶畅通。

3.动态调整模式下，上位机根据车流量模块返回的实时道路情况，优化某个方向的通行时间，缓解拥堵路段的交通压力。

线路连接

表 3.2

模块端口	继电器输入	继电器输出	抽屉插头
PA1	IN1	OUT1	JTD-EW-R
PA6	IN2	OUT2	JTD-EW-Y
PA7	IN3	OUT3	JTD-EW-B
PB0	IN10	OUT10	JTD-NS-B
PB1	IN11	OUT11	JTD-NS-Y
PB12	IN12	OUT12	JTD-NS-R

指令控制

1. 参照 3.2 节，连接好协调器并打开串口，单独给红绿灯节点上电。观察随着红绿灯的变化，红绿灯模块给上位机上报的数据，参照“智能交通通信协议”进行解析。

2. 串口发指令“FF FE 10 01 02 01 11 01”将交通灯模块设置成公交优先模式。

3. 串口发指令“FF FE 10 01 02 01 12 00 ”改变将南北方向变为绿灯。

3.2、 路灯模块

图 3.4 路灯示意图

功能介绍

路灯模块控制沙盘所有的路灯亮灭，有两种模式可供选择，分别是整体控制模式和节能控制模式。

1.整体控制模式下，模块根据环境光决定是否打开路灯，也是现实道路采用的方式。

2.节能控制模式下，模块根据上位机指令控制路灯亮灭，实现道路上有车辆才打开路灯，节省能量。

线路连接

表 3.3

模块端口	继电器输入	继电器输出	抽屉插头
PA1	IN1	OUT1	LD-1
PA6	IN2	OUT2	LD-2
PA7	IN3	OUT3	LD-3
PB0	IN4	OUT4	LD-4
PB1	IN5	OUT5	LD-5
PB12	IN6	OUT6	LD-6
PB13	IN7	OUT7	LD-12
PB14	IN8	OUT8	LD-11
PB15	IN9	OUT9	LD-10
PA8	IN10	OUT10	LD-9
PA11	IN11	OUT11	LD-8
PA12	IN12	OUT12	LD-7

指令控制

1. 参照 3.2 节，连接好协调器并打开串口，单独给路灯节点上电。用手去堵住电路板上的光敏传感器，之后松开，观察随着路灯的变化，路灯模块给上位机上报的数据，参照“智能交通通信协议”进行解析。

2. 串口发指令“FF FE 30 01 02 01 11 01”将路灯模块设置成节能控制模式。

3. 串口发指令“FF FE 30 01 03 01 12 01 01”打开第 1 号路灯。

3.3、 车流量模块

图 3.5 车流量示意图

功能介绍

车流量模块主要功能是统计道路车流量，一般为其连接 4-8 个红外传感器或者地磁线圈，如上图中为红外传感器。每当车辆经过时，该模块就可以捕捉到一个信号，统计 20S 内的流量值，上报给智能网关。

线路连接

表 3.4

模块端口	抽屉插头
不接	LL-GND
不接	LL-5V
PB12	IR1
PB13	IR3
PB14	IR5
PB15	IR7

指令控制

1. 参照 3.2 节，连接好协调器并打开串口，单独给路灯节点上电。观察该节点串口上报的数据，参照“智能交通通信协议”进行解析。

2. 用手去扫过路口的红外传感器，观察该节点给上位机上报的数据，对比数据的变化。

3.4、 ETC 模块

图 3.6 ETC 收费站模型

功能介绍

ETC(Electronic Toll Collection) 不停车收费系统模拟目前世界上最先进的路桥收费方式。通过安装在车辆挡风玻璃上的 915MhzRFID 超高频车载电子标签与在收费站 ETC 车道上的陶瓷天线之间的 RFID 读写通信，ETC 模块通过 Zigbee 节点与上位机通信结算扣费，从而达到车辆通过路桥收费站不需停车而能交纳路桥费的目的。

整个系统由红外传感器、3.5 寸 LCD、UHF 读卡器、舵机组成，简单来说，小车到达 ETC 收费站，触发红外传感器，ETC 模块控制 UHF 读卡器读取车上粘贴的 RFID 标签，与上位机通信，UHF 读卡器写 RFID 标签扣费，控制舵机抬杆，从而就完成了一个收费流程。

线路连接

表 3.5

模块端口	抽屉插头
5V	ETC-5V
GND	ETC-GND
GND	UHF-R-GND
IR	ETC-IR
PWM	ETC-PWM
232_1_TX	UHF-R-RX
232_1_RX	UHF-R-TX
232_2_TX	ETC-RX
232_2_RX	ETC-TX

指令控制

此部分用指令控制略显复杂，为避免冗长篇幅，用户可根据通信协议自行实验。

3.5、 停车场模块

图 3.7 停车场模型

功能介绍

智能停车场模拟无人值守的自助停车收费系统，系统组成与 ETC 模块类似，当有车辆到达，通过 UHF 超高频读卡器读取标签内的小车信息和余额，并自动为车辆分配车位，车辆根据分配到的车位号自动索引至该车位并停车入库。

线路连接

表 3.6

模块端口	抽屉插头
5V	入口-5V
GND	入口-GND
GND	UHF-R-GND
IR	入口-IR
PWM	入口-PWM
232_1_TX	UHF-R-RX
232_1_RX	UHF-R-TX
232_2_TX	入口-RX
232_2_RX	入口 TX

指令控制

此部分用指令控制略显复杂，为避免冗长篇幅，用户可根据通信协议自行实验。

3.6、 公交站模块

图 3.8 公交站模型

功能介绍

公交站模块主要就是一块 3.5 寸液晶屏的公交站，通过接收上位机的指令显示公交车的信息，方便乘客出行。

线路连接

表 3.7

模块端口	抽屉插头
5V	5V
GND	GND
232_2_TX	RX
232_2_RX	TX

指令控制

1. 参照 3.2 节，连接好协调器并打开串口，单独给公交节点上电。
2. 串口发指令“FF FE 20 01 03 01 11 01 03”，液晶屏上就会显示 1 号车辆距离 3 公里

3.7、 情报板模块

图 3.9 情报发布板

功能介绍

情报板模块用于模拟，真实道路的情报发布牌，用于接收上位机发布的路况和环境信息，每 10S 更新一次。

线路连接

表 3.8

模块端口	液晶接口
5V	红线
GND	黑线
SID	黄线
SCLK	蓝线

指令控制

1. 参照 3.2 节，连接好协调器并打开串口，单独给情报发布节点上电。

2. 例如我们要发布一条“湿度：50%，温度 25 度；前方路段行驶通畅”的信息，首先用附带的“汉字十六进制转换工具”将“前方路段行驶通畅”转换成 GB2312 的 16 进制编码，如图所示：

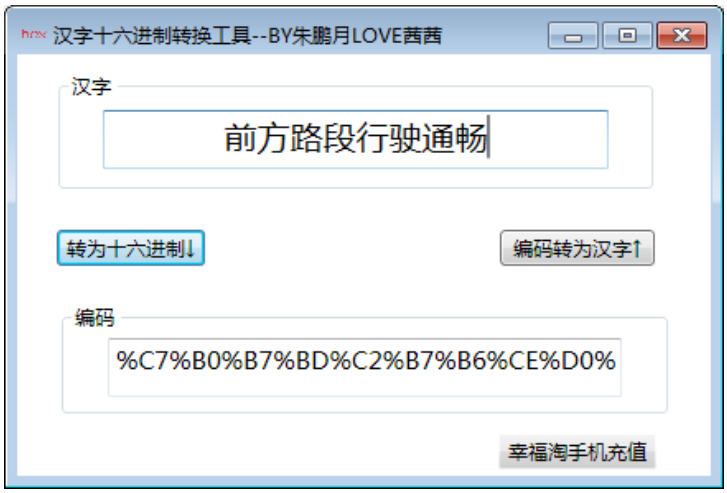


图 3.10 文字转换工具

3. 将转换得到的编码粘贴到记事本中，用空格替换掉中间的百分号，得到如下一串数据：

C7 B0 B7 BD C2 B7 B6 CE D0 D0 CA BB CD A8 B3 A9

4. 按照协议格式，将上面一段数据粘贴到一帧指令中，变成了“FF FE 80 01 13 01 02 01 10 C7 B0 B7 BD C2 B7 B6 CE D0 D0 CA BB CD A8 B3 A9”，用串口发送出去。观察情报发布板上内容的变化。

3.8、 环境监测模块

功能介绍

环境监测模块用于采集当前环境的 AM2321 温湿度值，并将数据上报给上位机。

线路连接

表 3.9

模块端口	AM2321
5V	VCC
GND	GND & SCL
WS	SDA

指令控制

1. 参照 3.2 节，连接好协调器并打开串口，单独给公交节点上电。
2. 查看串口接收到的数据，结合智能交通通信协议进行解析。

4、 智能网关 ITS 软件及演示操作

4.1、 ITS 软件使用前准备

1. ITS 软件可以通过 ZigBee 和智能交通沙盘,智能小车进行通信;对智能交通沙盘，智能小车进行控制。所有，在使用前，首先需要连接好 ZigBee 协调器。
2. 网关与协调器之间通过串口进行数据的传递，使用串口二，进行协调器与网关的连接(使用者也可以自己通过修改代码换用其他的串口)。

3. 智能 ETC 模块，智能停车场模块，实现了车辆的监控拍照功能。所以，在使用前，首先需要连接好摄像头拍照模块（ETC 模块摄像头连接高清摄像头接口一，停车场出口模块摄像头连接高清摄像头接口二），连接方式如下图：

图 4.1ETC 功能模块相机连接方式

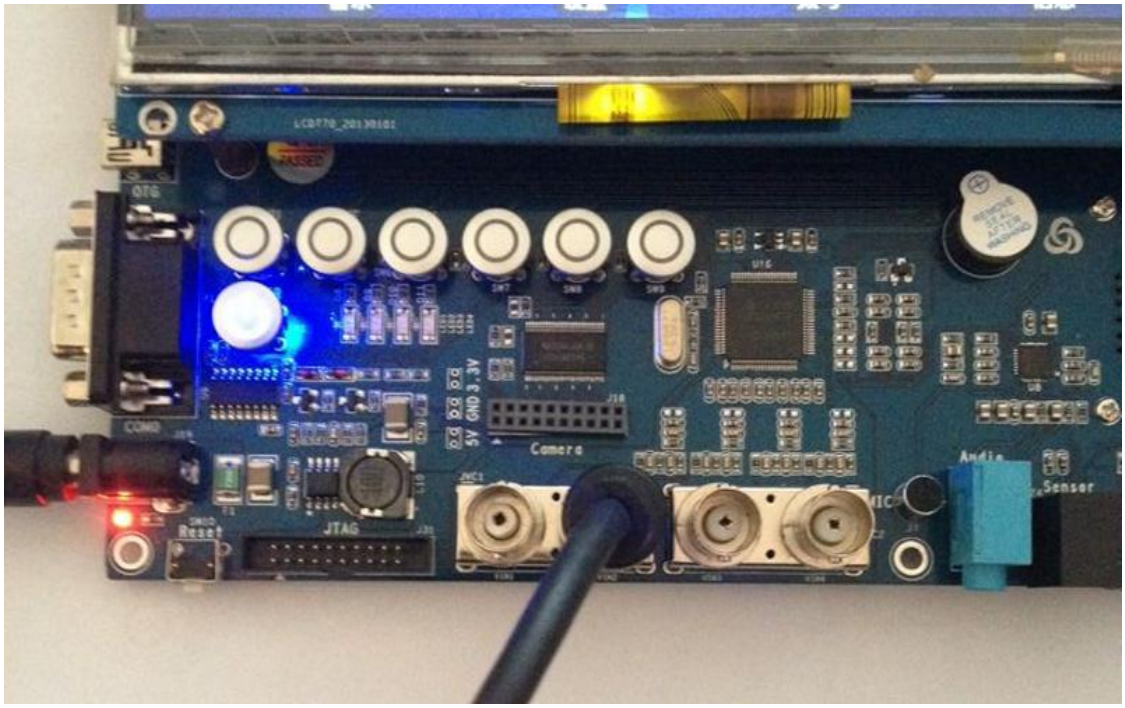


图 4.2 停车场功能模块相机连接方式

4. 给沙盘上电，检查所有节点电源是否打开，Zigbee 模块天线是否安装。

5. 长按网关上的电源键，待出现机器人图标时松开，网关启动成功后，打开 ITS 软件，入智能交通控制软件如图所示。

图 4.3 网关桌面 ITS 图标

6. 手拿几辆小车，上电后放入车位，待“嘀”的一声长鸣后小车停车成功。此时查看智能网关上软件的界面会看到相应车位出现小车图标。软件界面如下图所示，主要包括，智能红绿灯，智能路灯，智能停车场，智能 ETC，智能车流量等功能模块

图 4.4　ITS 程序主界面

7. 点击屏幕向右滑动菜单功能选项，如下图所示：

图 4.5　ITS 程序菜单功能项

4.2、 监控智能小车

1. 如果正确连接设备，节点都正常的运行，即可在界面上看到红绿灯的当前状态。如下图所示：

图 4.6 红绿灯状态显示

2. 分别给私家车上电，放在停车场相对应的车位（按车位，车号停放），即可在界面上显示小车在停车场的位置图，如下图所示：

图 4.7 小车上电入库显示

3. 界面右滑，滑出菜单功能项，点击停车场管理条目，进入停车场小车管理，如下图所示：

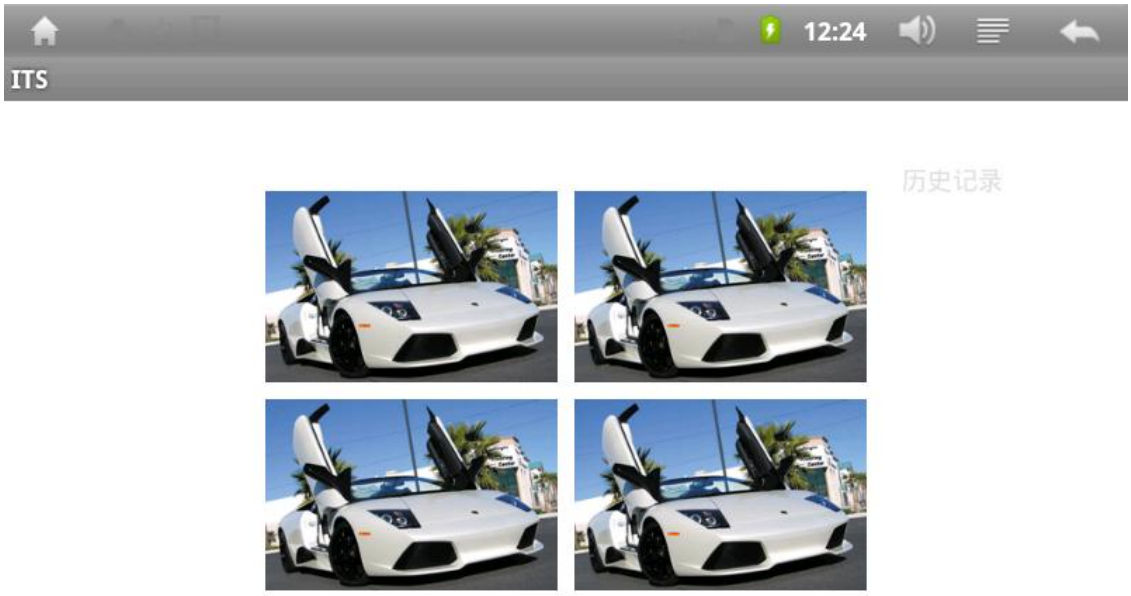


图 4.8 停车场管理界面

4. 点击左上角小车图片，让停放在一号停车位的小车驶出停车场(左上角图片对应一号停车位，右上角图片对应二号停车位，左下角图片对应三号停车位，右下角图片对应四号停车位)，如下图所示：

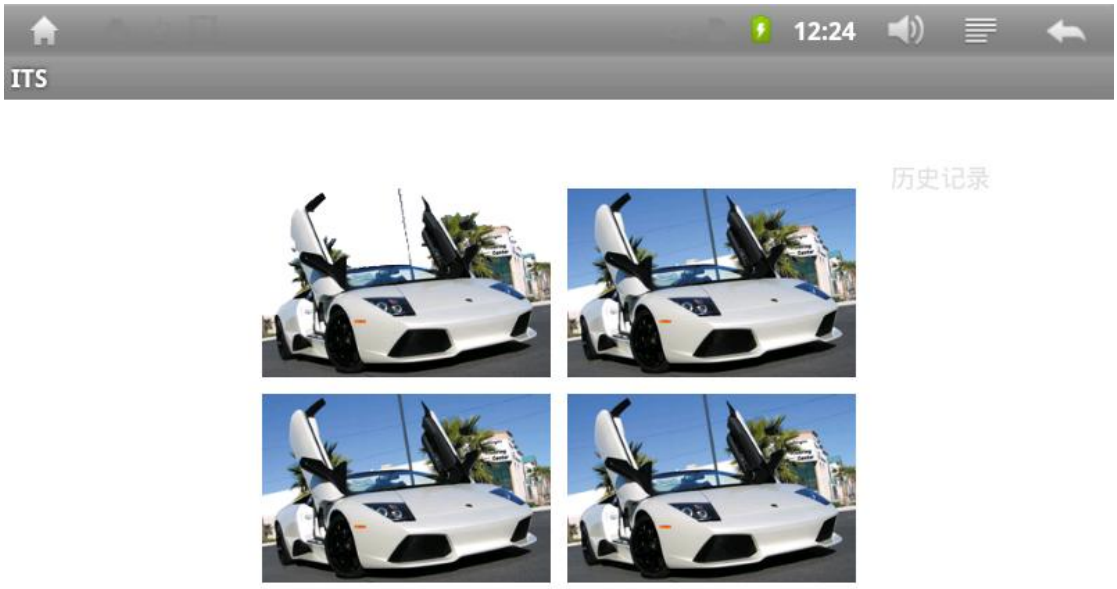


图 4.9 小车出库控制

5. 此后小车就会按照一定的路线运行(小车的路径已经在网关软件中定义个几条路径，开发者也可以自行规定其他的行车路径)(当小车再次进入停车场时，就会进行判断，如果停车场内存在车辆大于

2 的话，就会自行启动一辆小车驶出停车场；反之则不会有小车驶出停车场，使用者必须通过手动行驶相应位置的小车)，软件界面如下图所示：

图 4.10 ITS 运行界面

4.3、 路灯模式设置

1. 滑动软件界面可以进行路灯管理，设置路灯工作模式（默认是整体控制模式），点击条目选择不同的路灯工作模式，如下图所示：

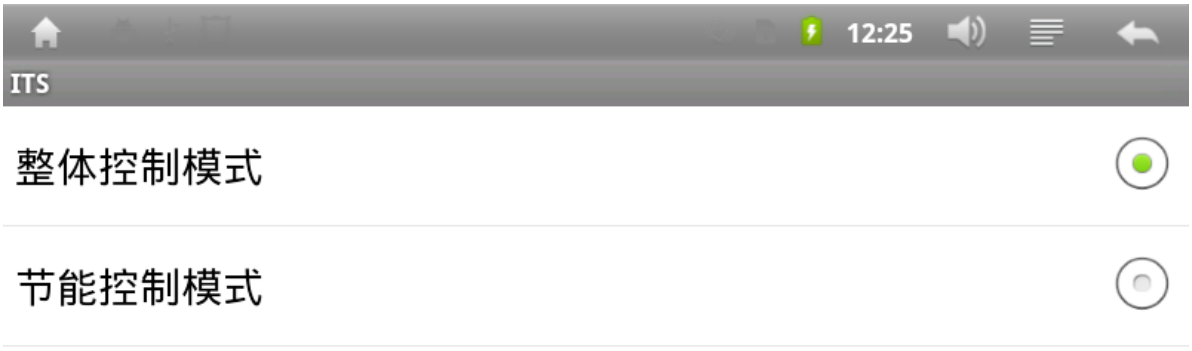


图 4.11 路灯模式设置

2. 当选择节能控制模式的时候小车行驶到哪里，路灯就会打开相应的路灯（如果设置的是整体控制，就会根据当前光照强度进行路灯的打开与关闭），如下图所示：

图 4.12 路灯状态显示

图 4.13 路灯整体控制模式

4.4、 红绿灯模式设置

1. 用户可以设置红绿灯的工作模式（默认是固定工作模式），在固定工作模式下所有的车遵循正常的红绿灯通行法则，设置工作模式如下图所示：

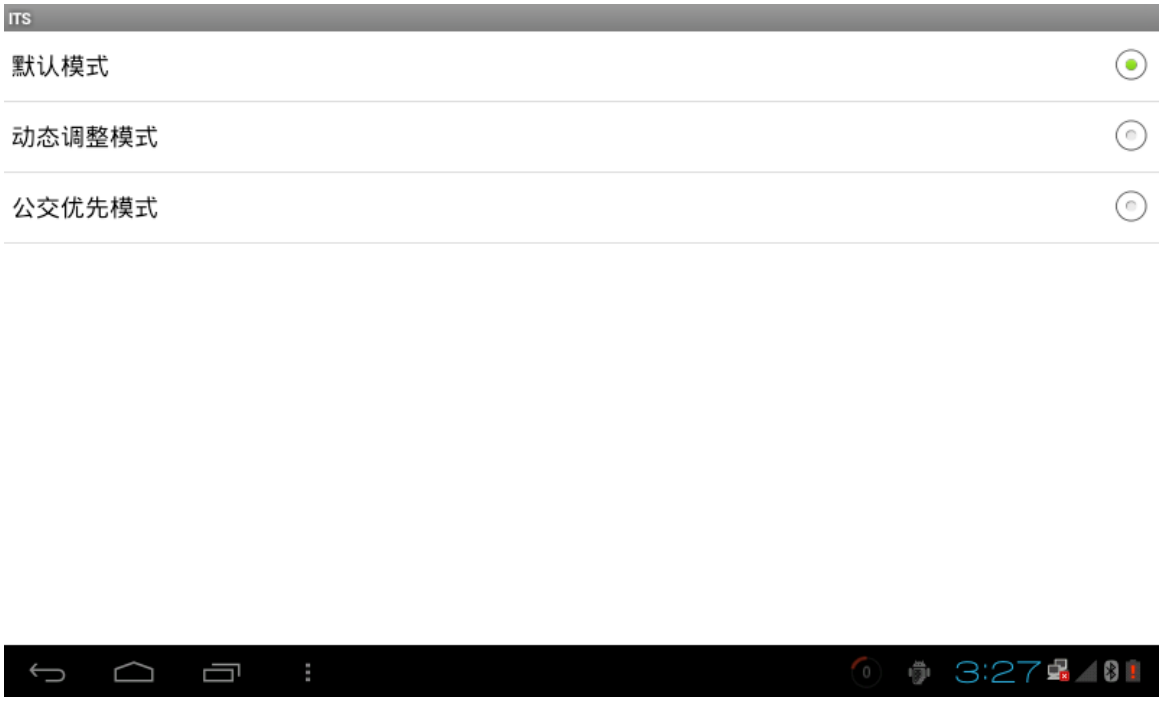


图 4.14 红绿灯模式设置

2. 用户可以设置红绿灯的工作模式，设置动态调整模式(设置动态模式之后，红绿灯就会根据南北方向以及东西方向车流量的多少，控制红绿灯的红绿灯时间，增加车流量多的绿灯通行时间)，设置工作模式方法如下：

点击如下所示进入绿灯延长时间的设置(在红绿灯原先的基础上增加绿灯的时间，建议设置为十以内的数字，因为原先的时间为十秒变化一次)

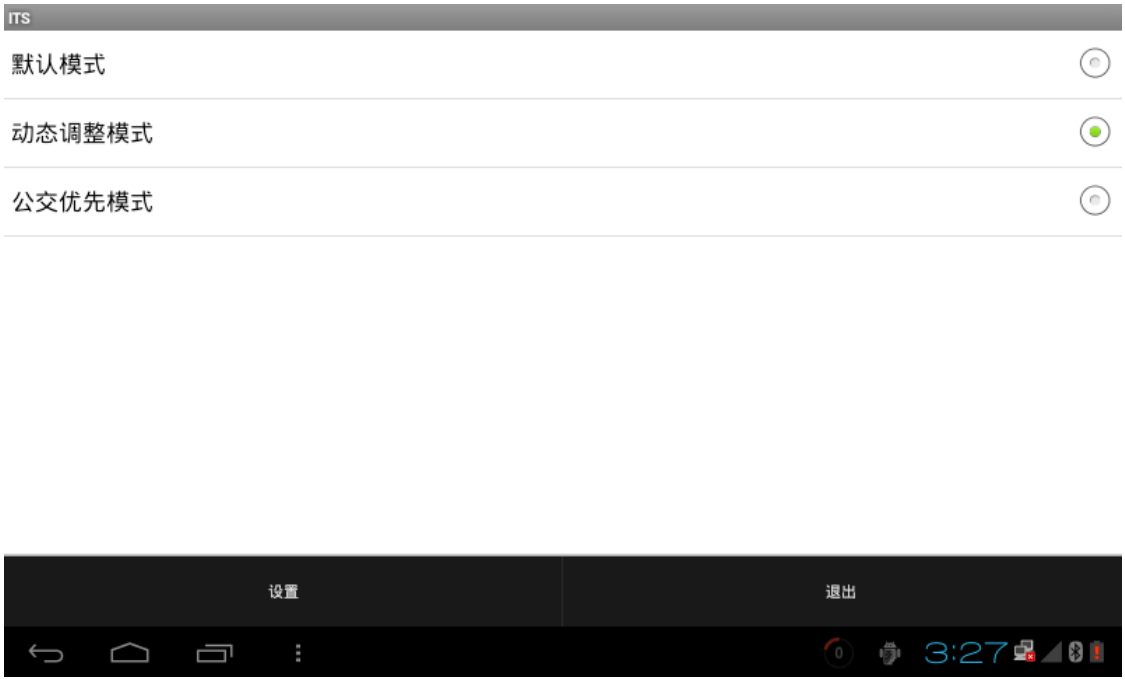


图 4.15 红绿灯模式设置

图 4.16 自定义红绿灯时长

4.5、 停车场历史记录

1. 用户可以查看驶出停车场的历史记录，达到停车场出口的车辆监控，操作时进入停车场管理，点击界面的历史记录，便可查看停车场出口小车监控，如下图所示：

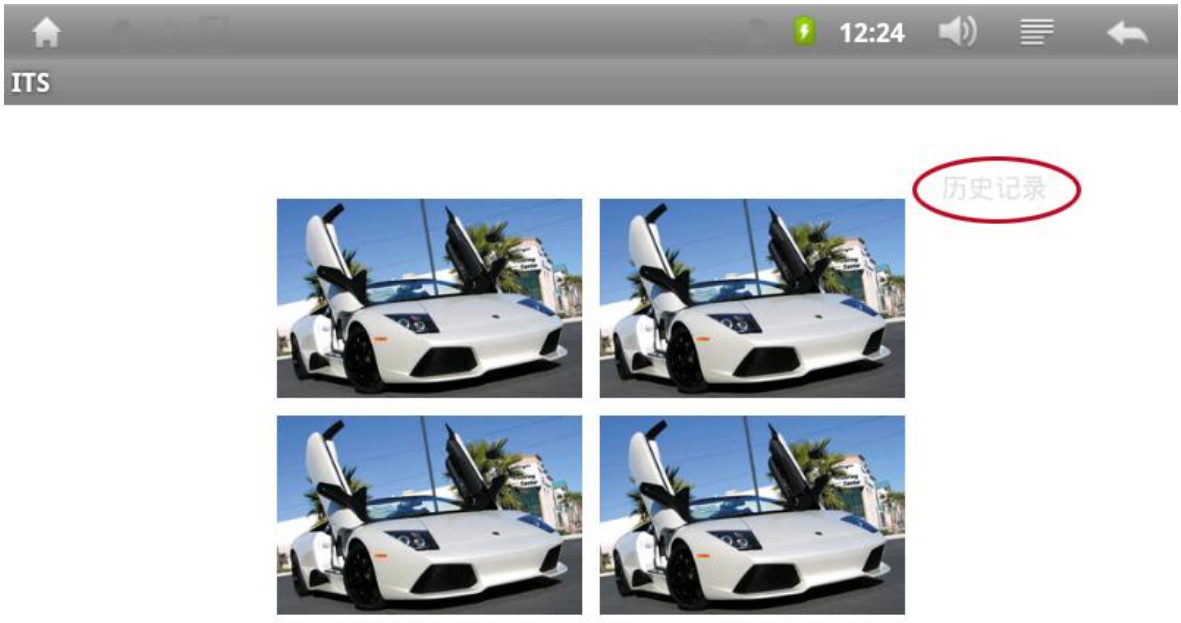


图 4.17 停车场历史记录

2. 使用者可以查看所有通过停车场和 ETC 的图片，路径在“mnt/sdcard/如果有历史图片就会存在一个 UICC 目录的文件夹，在 UICC 文件夹下存放这智能停车场出口 PARk 文件夹，智能 ETC 的小车监控图片 ETC 文件夹，（可以通过 ES 文件浏览器查看文件是否存在），ITS 软件启动的时候都会把以前的历史记录清空，只会保存当前的监控图片，使用者也可以通过修改程序代码达到不同的功能效果。

4.6、 车流量信息

1. 使用者可以查看每条道路的车流量信息，监控每条道路的路况，点击主界面右上角的车流量显示区域，还选择不同的路段查看如下图所示：

图 4.18 车流量显示

图 4.19 路段选择

智能交通

第二章 android 开发

2.1 智能交通物联网实训系统 Android 实验指导书

2.2 Android 开发环境的配置

1、 智能交通实验系统介绍

1.1、 智能交通平台主要功能

本实验系统为教学提供：智能小车，智能红绿灯，智能路灯，智能车流量，智能停车场，智能 ETC，智能网关(调动中心)等模块。使得学生能够充分理解智能交通系统的实现过程及实现细节。最终达到使学生能够构建自己的智能交通系统的教学目的。

智能交通实训系统通过沙盘、控制系统、无线传感网、Android 嵌入式网关、Android 智能交通实训软件，实现以下功能：

1. 智能红绿灯管理
   - 固定时间模式：采用此种模式时，每个路口红灯和绿灯的间隔时间是固定的。不随车流量变化而变化。
   - 动态调整模式：可以根据车流量大小来动态调整红绿灯变化时间。比如南北方向车辆多，东西方向车辆少，则适时调整南北方向的红绿灯时间。
2. 智能公交
   - 公交优先：进入公交优先模式时，当公交车快到达路口事，红绿灯会自动提前变绿，让公交车优先通过。
   - 公交运行信息实时显示系统：每个公交站台都有一个公交运行信息显示屏，会实时显示每路公交车的运行状态。比如某路公交车距离当前站点的距离、公交车抛锚、公交车人员情况等。
3. 智能路灯
   - 整体控制模式：根据光照传感器进行整体控制。光照流明度低于某一个值时，路灯整体打开。
   - 节能控制模式：在没有车辆的时候路灯熄灭，当有车辆即将进入本区域，本区域的灯光亮起。
4. ETC 系统
   - 智能 ETC 系统模拟真实的 ETC 自动收费系统，实现以下功能：

   车辆进入收费通道后，收费站 ETC 系统根据车上携带的 ETC 卡，确定是否放行，符合条件自动放行，不符合条件的提出警示信息。

   车辆进入出口收费站时，收费站 ETC 系统自动计算车辆的缴费数额，并在 ETC 卡上自动扣费，根据扣费情况进行车辆放行。

   1) 车辆通过 RFID 地标，获知进入收费站，就类似人通过眼睛知道进入收费站，车辆自动降低速度。

2) 车辆会触发 ETC 入口处的红外传感器，红外传感器会唤醒 UHF RFID，进行车辆上的 RFID 标签读取。并拍照。
3) 如果 RFID 标签内存储 ETC 数据，抬起 ETC 栏杆，并通过 ZigBee 告知车辆可以通过（如果没有这步，车辆不知道栏杆已经抬起）。如果没有存储 ETC 数据，则不抬起 ETC 栏杆。车辆停止。
4) 关键点是，车辆进入 ETC 通道后，会缓行一小段时间，在这小段时间内，ETC 系统必须完成 RFID 卡识别、抬杆及车辆放行指令发送。这样才能保证车辆不停下，能够流畅的过 ETC 收费站。
5) 如果车辆没有 ETC 卡，可通过手动放行。中央控制器给 ETC 系统发送一条指令，作出抬杆动作，并给车辆发出继续行走指令。
6) 车辆第二次来到 ETC 收费站，收费站将自动改为出口模式。通过车型确定收费数目（路程一样）。
7) 如果 ETC 卡余额足够，自动扣费。信息屏显示扣费金额，并播放扣费成功，一路平安语音。
8) 如果 ETC 卡余额不足，要求充值，车辆没有收到中央控制器继续行走指令，停车缴费。充值成功之后，扣费放行，显示和播放扣费信息。

5. 智能停车场

智能停车场包含以下功能：

1) 在停车场门口有信息屏，展示停车位空余情况。
2) 车辆进入停车场入口，读取 RFID 地标传感器，停车等待。
3) 停车场门禁通过 UHF RFID 读写器读取车辆信息，写入停车时间，通过 ZigBee 告知车辆具体停车位。
4) 控制器控制信息屏，显示具体停车导引信息。
5) 控制器控制闸门打开。车辆驶入。通过地标传感器判断转弯路口。
6) 控制器通过 ZigBee 告知车辆出场，车辆接收指令之后离开停车场。
7) 到达停车场出口时，车辆停车，红外传感器触发，停车场控制器通过 UHF 读写器读取车辆 RFID 卡数据，并扣费。
8) 停车场通过 ZigBee 告知车辆可以驶出，并抬杆放行。

6. 车流量检测

在路口的每个车道上都有红外传感器，可进行车流量统计。可根据车流量进行车辆智能管控和调度。

7. 磁导线与 RFID 地标全局定位及导航

2、 智能小车定位实验

2.1、 实验目的

1. 熟悉 UI-IOT-ITS 智能交通物联网技术教学实验平台的使用方法。

2. 熟悉 Android 应用程序开发的编译下载流程

3. 学习 UI-IOT-ITS 智能交通物联网通信协议。

2.2、 实验内容

通过 UI-IOT-ITS 智能交通物联网技术教学实验平台实现对智能小车在沙盘上的定位。

2.3、 实验仪器

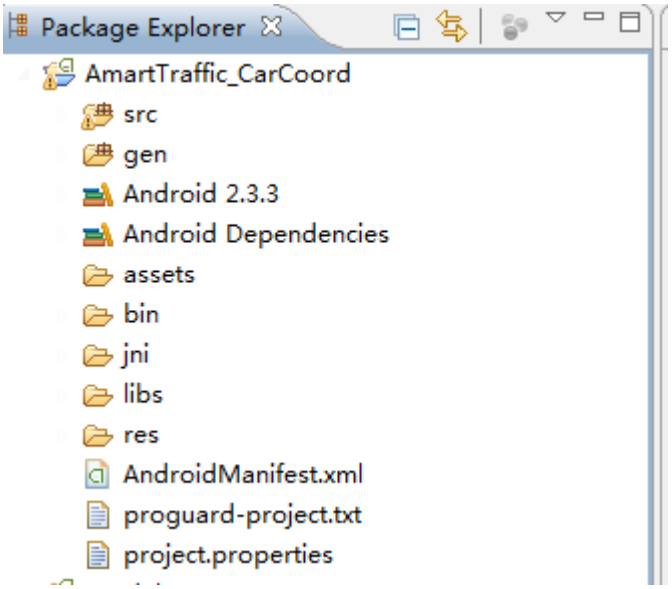
1. 本实验所需仪器：UI-IOT-ITS 智能交通物联网技术教学实验平台。

2. 安装了 Eclipse 和 Android SDK 的电脑一台

2.4、 实验步骤

1. 建一个 Android 工程

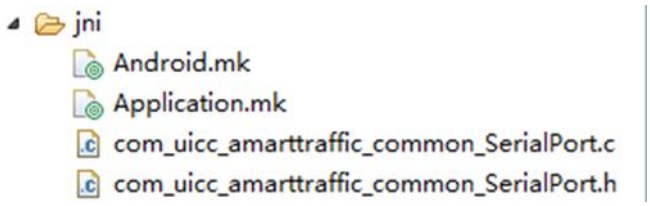
在Eclipse中，点击菜单[File] – [New] – [Project…]。弹出新建工程窗口。选择工程类型为“Android Project”，点击“Next >”按钮，进入工程信息配置窗口，项目结构如下图所示：



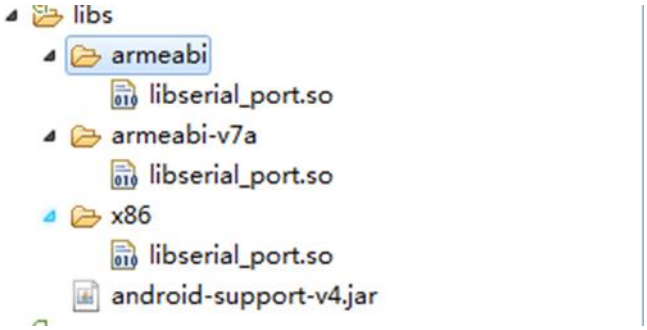
2. 通过 JNI 方法调用 C 语言写的打开,关闭串口方法。

1) 在新建的项目中新建一个jni的Folder文件夹，往里面导入如下文件，可从提供的实例的

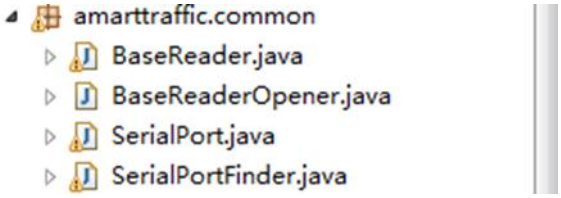
相应位置copy进去



2) 在项目的libs文件夹中导入如下文件，可从提供的实例的相应位置copy进来



3) 在项目src文件下，新建一个package，包名为com.uicc.amarttraffic.common；在本包名下导入如下java文件(串口的打开，关闭，读写操作的实例化)



3. 编辑程序界面

例如：双击工程子目录[res]-[layout]-[activity_main.xml]打开资源文件activity_main.xml。

点击右侧编辑区下方的选项卡“activity_main.xml”进入文本编辑模式，修改文件中的标签内容

可参照提供实训案例AmartTraffic_CarCoord中activity_main.xml编写。

4. 软件编程

1) 通过编写MyApp.java文件extends application，实例化串口类，并打开串口；并在清单文件AndroidManifest.xml中增加如下指令android:name=".MyApp"

```
<application
    android:name=".MyApp"
    android:allowBackup="true"
    android:icon="@drawable/log"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
```

MyApp.java 文件 com/uicc/amarttraffic/activity/MyApp.java

```
public class MyApp extends android.app.Application
{
    public static MyApp myApp;

    public BaseReaderOpener  br;
    @Override
    public void onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();
        br=new BaseReaderOpener();
        myApp=this;
        onOpenClick();//打开串口
        File file=new File("mnt/sdcard/UICC");
        new Util().DeleteFile(file);//删除所有的历史监控图片
    }
```

2) 在主界面创建的时候启动一个线程读取通过串口发送过来的数据，并做出相应的数据处理操作HomeThread.java 文件com/uicc/amarttraffic/mode/HomeThread.java

a) 启动线程读取串口数据(读取的数据存储在一个中间缓存中)：

Thread myThread = **new** Thread(**new** MyThread());

myThread.setPriorityThread.*MAX_PRIORITY*);

TheedPoolManager.*getInstance*).addTaskmyThread;

```
try {
    if(reader.baseReader.mInputStream!=null){
        while ((size = reader.baseReader.mInputStream.available()) > 0) {
            buffer = new byte[size];
            reader.baseReader.mInputStream.read(buffer, 0,size);
            data=buffer;
            if(data != null){
                Restr.append(utility.bytesToHexString(data).toString());
                Log.e(TAG,utility.bytesToHexString(data).toString());
            }
        }
    }else{}
    Thread.sleep(10);
} catch (Exception e) {
    e.printStackTrace();
```

b) 对串口接收到的数据进行有效处理，解析判断是否是一帧有效地数据

```
//================================
if(Restr.length()>25){//判断符合一帧数据结构的数
    if(Restr.toString().startsWith("ff fe")){
        int dataLength=Integer.parseInt(Restr.toString().trim().substring(12, 14),16)
        int end=(dataLength+8)*2+dataLength+8;
        if(Restr.length()>=end){
            str=Restr.substring(0,end);
            Restr.delete(0, end);
            DataFunction(str);
            if(!Restr.toString().contains("ff fe")){
                Restr.delete(0,Restr.length());
            }
        }
    }else {//如果不是帧头开始,判断符合一帧数据结构的数
        if(Restr.toString().contains("ff fe")){
            Restr.delete(0,Restr.indexOf("ff fe"));
            if(utility.Stringlocation(Restr)-Restr.indexOf("ff fe")<20){
                Restr.delete(0,utility.Stringlocation(Restr));
            }
        }else{
            Restr.delete(0,Restr.length());
        }
        continue;
    }
}else{};
```

c) 对读到的串口数据进行相应的功能操作(判断是不是小车节点发送的数据)：

private synchronized void DataFunction(String str){}

```
rivate synchronized void DataFunction(String str){
    Dataaddress=Integer.parseInt(str.substring(9, 11), 16);
    if(0x70<=Dataaddress&&Dataaddress<=0x7f){
        //小车位置显示
            CarCoord=str.toString().split(" ");
            //车辆信息
            if(CarCoord.length>=11&&CarManager.JCar(CarCoord[3])&&CarCoord[5].equalsIgr
                Nodeaddress=CarCoord[3];//小车节点地址
                strCarId=CarCoord[8];//小车ID号
                strCarType=CarCoord[9];//小车类型
                CarManager.CarNode.put(Nodeaddress, strCarId+"-"+strCarType);//根据节点ID
                CarManager.CarIndex.put(strCarId+"-"+strCarType, Nodeaddress);//根据小车I
```

d) 进行小车位置的绘制(存储小车上电发送的小车节点地址，并根据小车发送的坐标地址进行小车位置的绘制)：

private synchronized void DrawCar(String Nodeaddress,String CarCoord,String Cartype){}

```
/*
 * 绘制界面图片，小车的具体位置
 */
private synchronized void  DrawCar(String Nodeaddress,String CarCoord,String Ca
    CarUI.add(Nodeaddress);//存储时间内的所有小车
    if(!Cardata.contains(Nodeaddress)){//判断是否已经存在车信息
        BackBitmap(Nodeaddress);
        if(bitmap!=null){
        Cardata.add(Nodeaddress);//第一次出现的小车地址保存下来
        car = new Car(bitmap, -100, -100);
        MyView.carmanager.addCar(car);//第一次出现小车添加到车管理中

        }
    }else{}
    DrawPathCar(Nodeaddress,CarCoord,Cartype);
}
```

e) 智能小车信息对象化，坐标信息对象化（便利与界面绘制）

编写一个管理类进行小车信息的管理，坐标位置的管理。

- 创建小车管理类CarManager.java文件

com/uicc/amarttraffic/manager/CarManager.java

a) 添加小车个数(接收到小车上电数据)

public void addCar(Car car){}

b) 界面绘制小车(在自定义View中绘制出小车)

public synchronized void drawAllCars(Canvas canvas){}

```
/*
 * 绘制车
 */
public synchronized void drawAllCars(Canvas canvas){
    for(Car c : cars){
        c.drawSelf(canvas);
    }
}
```

c) 判断是不是小车节点地址

public static boolean JCar(String address){}

```
/**
 * 判断是不是小车节点地址
 */
public static boolean JCar(String address){
    int type=Integer.parseInt(address,16);
    if(127>=type&&type>=112){
        return true;
    }
    return false;
    }
```

- 创建坐标管理类CoordManager.java文件 com/uicc/amarttraffic/manager/

CoordManager.java

```
private static final String TAG="CoorManager";
private int width,hight;
private Coord coord;
private List<Coord> Coords=new ArrayList<Coord>();
public void addCoord(Coord coord){
    Coords.add(coord);
}
public List<Coord> getAllCoord(){
    return Coords;
}
private CoordManager() {
    // TODO Auto-generated constructor stub
}
private static CoordManager coordManager=new CoordManager();
public static CoordManager getCoordManager(){
    return coordManager;
}
```

a) 添加沙盘标签位置信息(添加每个坐标位置)

public void addCoord(Coord coord){}

b) 把所有的坐标位置映射(提前映射坐标位置 ,便于提取绘制小车当前坐标位置)

public void addAllCoord(){}

```
/
 * 添加所有的小车坐标
 */
public void addAllCoord(){
    width=MainActivity.sScreenWidth;
    hight=MainActivity.sScreenHeight;
    coord=new Coord((width/31)*2.5f,(hight/16)*0.9f);//车道信息 A1-1
    addCoord(coord);
    coord=new Coord((width/31)*4,(hight/16)*0.9f);//A1-2
    addCoord(coord);
    coord=new Coord((width/31)*6,(hight/16)*0.9f);//A1-3
    addCoord(coord);
    coord=new Coord((width/31)*8,(hight/16)*0.9f);//A1-4
    addCoord(coord);
    coord=new Coord((width/31)*10,(hight/16)*0.9f);//A1-5
    addCoord(coord);
```

c) 获得映射好坐标集合

```
public List<Coord> getAllCoord(){}
```

3) 自定义View（把小车的坐标实时绘制在界面上）

MyView.java 文件 com/uicc/amarttraffic/activity/ MyView.java

```
public MyView(Context context, AttributeSet set) {
    super(context, set);
    paint=new Paint();
    carmanager=new CarManager();
    new Mythread().start();
}
@Override
protected void onDraw(Canvas canvas) {
    // TODO Auto-generated method stub
    super.onDraw(canvas);
        carmanager.drawAllCars(canvas);
}
```

1) 具体代码请参考项目案例 AmartTraffic_CarCoord 工程

3、 智能 ETC 模块实验

3.1、 实验目的

1. 熟悉 UI-IOT-ITS 智能交通物联网技术教学实验平台的使用方法。
2. 熟悉 Android 应用程序开发的编译下载流程

3. 学习 UI-IOT-ITS 智能交通物联网通信协议。

3.2、 实验内容

通过 UI-IOT-ITS 智能交通物联网技术教学实验平台实现对智能小车通过智能 ETC 模块操作流程。

3.3、 实验仪器

1. 本实验所需仪器：UI-IOT-ITS 智能交通物联网技术教学实验平台。

2. 安装了 Eclipse 和 Android SDK 的电脑一台

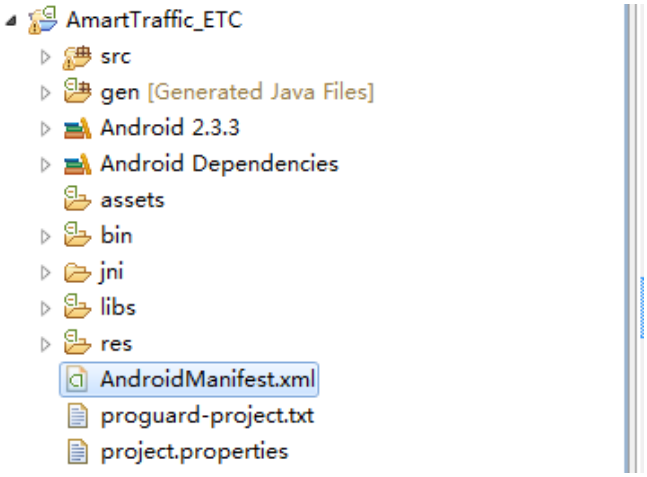
3.4、 实验步骤

1. 建一个 Android 工程

在Eclipse中，点击菜单[File] – [New] – [Project…]。弹出新建工程窗口。

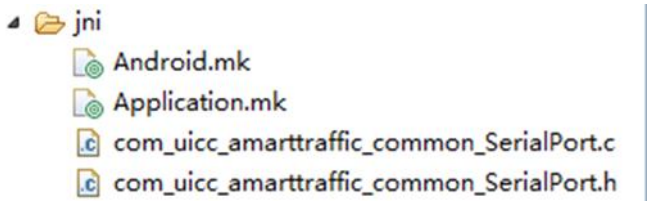
选择工程类型为“Android Project”，点击“Next >”按钮，进入工程信息配置窗口

项目结构如下图所示：

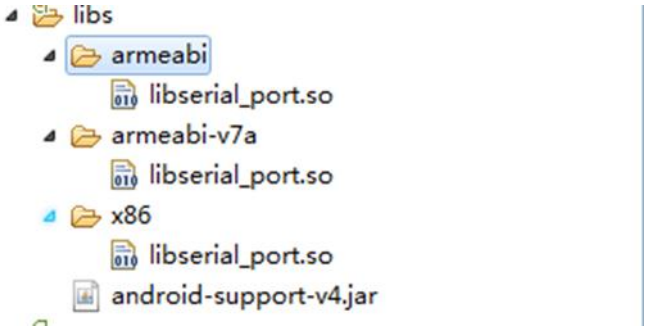


2. 通过JNI方法调用C语言写的打开,关闭串口方法。

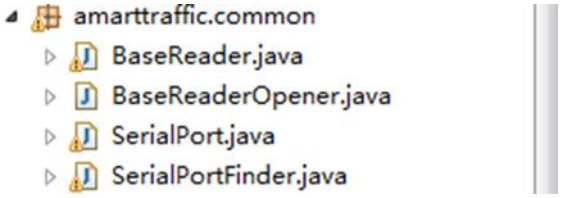
1) 在新建的项目中新建一个jni的Folder文件夹，往里面导入如下文件，可从提供的实例的相应位置copy进去



2) 在项目的libs文件夹中导入如下文件，可从提供的实例的相应位置copy进来



3) 在项目src文件下，新建一个package，包名为com.uicc.amarttraffic.common；在本包名下导入如下java文件(串口的打开，关闭，读写操作的实例化)



3. 编辑程序界面

例如：双击工程子目录[res]-[layout]-[activity_main.xml]打开资源文件activity_main.xml。

点击右侧编辑区下方的选项卡“activity_main.xml”进入文本编辑模式，修改文件中的标签内容

可参照提供实训案例AmartTraffic_ETC中activity_main.xml编写。

4. 软件编程

1) 小车位置的获取以及串口数据的获取上面以作讲解这里不在重复，见认知实验<智能小车定位>

2) 在清单文件中增加相应权限（AndroidManifest.xml）

```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="8" />
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

3) 小车是否进入智能ETC系统（通过判断是否是ETC节点发送的数据进行处理）

HomeThread.java 文件 com/uicc/amarttraffic/mode/HomeThread.java

private synchronized void DataFunction(String str){}

```
private synchronized void DataFunction(String str){
    Dataaddress=Integer.parseInt(str.substring(9, 11), 16);
    if(0x50<=Dataaddress&&Dataaddress<=0x5f){
        if(share.getETC()){//智能ETC
            strETC=str.toString().split(" ");
            if(strETC.length>=17&&strETC[3].equalsIgnoreCase("50")&&strETC[5].equals
                //人口模式
                Nodeaddress=strETC[3];//ETC节点地址
                strCarId=strETC[9];//小车ID号
                strCarType=strETC[10];//小车类型
                Money=strETC[11];//小车余额
                StrMessage.append(strCarId);
                StrMessage.append("-");
                StrMessage.append(strCarType);
                StrMessage.append(">");
                StrMessage.append(Util.TimeNow());
                StrMessage.append(">");
                StrMessage.append(Money);
                mMyHandler.obtainMessage(Present.MESSAGE_MODE_ETC_IN,StrMessage.toSt
```

4) 通过Handler把接收到的数据传递给相应的函数进行处理。

HomeThread.java 文件 com/uicc/amarttraffic/mode/HomeThread.java

private Handler mMyHandler(){}

```
 * 根据不同的条件处理不同的数据
 */
private Handler mMyHandler(){
    if(mMyHandler==null){
        mMyHandler=new Handler(){
            public void handleMessage(Message msg) {
                switch (msg.what) {
                case Present.MESSAGE_MODE_ETC_IN:
                    if(etcMode==null){
                        etcMode=new EtcMode(context, mMyHandler);
                        String etcData=msg.obj.toString();
                        etcMode.etcData=etcData;
                        etcMode.In_etc();//第一次进入ETC
                    }else{
                        String etcData=msg.obj.toString();
                        etcMode.etcData=etcData;
                        etcMode.In_etc();//第一次进入ETC
                    }
                    break;
                case Present.MESSAGE_MODE_ETC_OUT:
                    if(etcMode==null){
                        etcMode=new EtcMode(context, mMyHandler);
                        String car_etc=msg.obj.toString();
                        etcMode.out_etc(car_etc);//第二次进入ETC
                    }else{
                        String car_etc=msg.obj.toString();
                        etcMode.out_etc(car_etc);//第二次进入ETC
```

5) 对进入智能ETC的进行相应的操作、

EtcMode.java 文件 com/uicc/amarttraffic/ mode/ EtcMode.java

a) 智能小车通过智能ETC入口数据的处理函数(判断卡内余额是否充足（余额不足自动充值），如果余额充足网关控制中心下发小车的进入时间)

public void In_etc(){}

```
/*
 * 对通过ETC系统车辆进行设置 查看车辆是否有金额，如果有车辆通过提示客户扣费金额；没有提示充值，起杆，车辆放行。
 * 小车人口模式进入ETC
 */
public void In_etc() {
    Log.d(TAG, "In_etc="+etcData);
    in_state=true;
    new Thread(new Runnable() {
        @Override
        public void run() {
            // TODO Auto-generated method stub

                if (share.getETC()) {
                    if (etcData != null) {
                        Data = etcData.split(">");
                        if(Data.length>=3){
                            NowTime =Util.TimeNow().split(":");
                        if(!NowTime[0].isEmpty()&&!NowTime[1].isEmpty()&&!NowTime[2
                            Hour = Integer.parseInt(NowTime[0]);
                            Minute = Integer.parseInt(NowTime[1]);
                            Second=Integer.parseInt(NowTime[2]);
                        }
                        InCaraddress=CarManager.CarIndex.get(Data[0]);//获得人口模式下小
                        if (!Data[2].isEmpty()&&Integer.parseInt(Data[2], 16) > 10)
                            mhandler.obtainMessage(Present.MESSAGE_DATA_HINT_ETC,(w
                        } else {
                            System.out.println("in_state--------Money");
```

b) 智能小车通过智能ETC出口数据的处理函数(判断卡内余额是否充足（余额不足自动充值），如果余额充足网关控制中心下发小车的驶出时间以及消费金额)

public void out_etc(final String car_etc){}

```
/*
 * 小车出口模式进入ETC
 */
public void out_etc(final String car_etc){
    Log.d(TAG, "out_etc="+car_etc);
    out_state=true;
    new Thread(new Runnable() {
        @Override
        public void run() {
            // TODO Auto-generated method stub
            out_etc=car_etc.split(">");
            if(!out_etc[0].isEmpty()&&!out_etc[2].isEmpty()){
                OutCaraddress=CarManager.CarIndex.get(out_etc[0]);//获得出口模式下小车节点
                money= Integer.parseInt(Long.toString(Time(out_etc[2])));
                inTime=Util.TimeNow().split(":");
                if(!inTime[0].isEmpty()&&!inTime[1].isEmpty()&&!inTime[2].isEmpty()
                    Hour=Integer.parseInt(inTime[0]);
                    Minute=Integer.parseInt(inTime[1]);
                    Second=Integer.parseInt(inTime[2]);
                }
                Log.e(TAG, "ETC消费金额："+EMonetary(money)+" 剩余金额:"+Integer.parseInt
                if(Integer.parseInt(out_etc[1],16)-EMonetary(money)>10){//余额是否大于0
                        mhandler.obtainMessage(Present.MESSAGE_DATA_HINT_ETC,(width
                        mReader.Out_Time(0x50,Hour,Minute,Second,EMonetary(money),EI
                }else{
                    System.out.println("Out_state--------Money");
                    mhandler.obtainMessage(Present.MESSAGE_DATA_HINT_ETC,(width/31)
```

6) 智能ETC入口实现监控拍照功能

调用模拟摄像头进行拍照功能（MainActivity.java 进行摄像头的实例化以及图片的存

储）：

```
case Present.MESSAGE_DATA_CAMERA:
    String id=msg.obj.toString();
    setCameraId(Integer.parseInt(id));
    mCameraID = Integer.parseInt(id);
    Util.ThreadSleep();
    takepicture(id);
    break;
```

```
/**
 * 拍照并保存图片
 */
public void takepicture(final String str){
    mCamera.takePicture(null, null, new PictureCallback() {

        public void onPictureTaken(byte[] data, Camera camera) {
            // TODO Auto-generated method stub
            if(str.equalsIgnoreCase("1")){
                File file=new File("mnt/sdcard/UICC/ETC");
                if(!file.exists()){
                    file.mkdirs();
                }
                File filejpg=new File("mnt/sdcard/UICC/ETC/UICC"+System.currentT
                savePictuer(filejpg, data);
```

7) 具体代码请参考项目案例 AmartTraffic_ETC 工程

4、 智能情报板实验

4.1、 实验目的

1. 熟悉 UI-IOT-ITS 智能交通物联网技术教学实验平台的使用方法。
2. 熟悉 Android 应用程序开发的编译下载流程
3. 学习 UI-IOT-ITS 智能交通物联网通信协议。

4.2、 实验内容

通过 UI-IOT-ITS 智能交通物联网技术教学实验平台实现对情报版模块操作，当接收到环境采集模块数据，从而进行情报板数据的更新。

4.3、 实验仪器

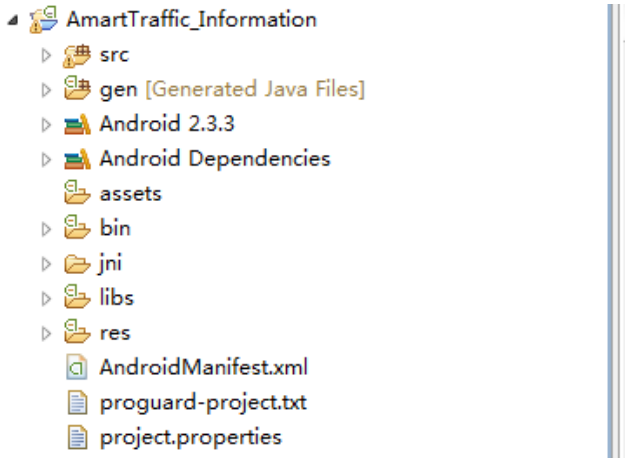
1. 本实验所需仪器：UI-IOT-ITS 智能交通物联网技术教学实验平台。
2. 安装了 Eclipse 和 Android SDK 的电脑一台

4.4、 实验步骤

1. 建一个 Android 工程

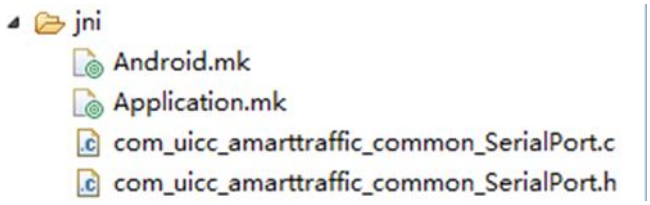
2. 在 Eclipse 中，点击菜单[File] – [New] – [Project…]。弹出新建工程窗口。

3. 选择工程类型为“Android Project”，点击“Next >”按钮，进入工程信息配置窗口，项目结构如下图所示：



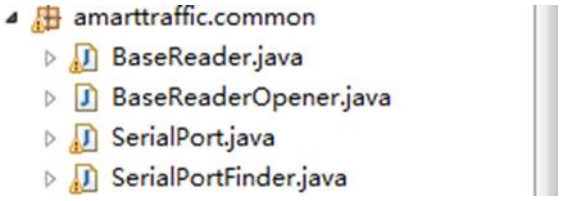
4. 通过 JNI 方法调用 C 语言写的打开,关闭串口方法。

4) 在新建的项目中新建一个jni的Folder文件夹，往里面导入如下文件，可从提供的实例的相应位置copy进去



5) 在项目的libs文件夹中导入如下文件，可从提供的实例的相应位置copy进来

6) 在项目src文件下，新建一个package，包名为com.uicc.amarttraffic.common；在本包名下导入如下java文件(串口的打开，关闭，读写操作的实例化)



5. 编辑程序界面

例如：双击工程子目录[res]-[layout]-[activity_main.xml]打开资源文件activity_main.xml。

5、 点击右侧编辑区下方的选项卡“activity_main.xml”进入文本编辑模式，修改文件中的标签内容可参照提供实训案例AmartTraffic_Information中activity_main.xml编写。

6. 软件编程

1) 通过串口对数据读取上面已作讲解，这里就不再重复请参照上述认知实验

2) 把需要发送的温馨提示语，用一个管理类来管理（温馨提示语轮流发送）

InformationManager.java 文件

com/uicc/amarttraffic/manager/InformationManager.java

情报板一行显示的数据不能超过10个中文

```
public InformationManager() {
    // TODO Auto-generated constructor stub
    Information_one="珍爱生命，勿疲劳驾驶";
    Information_two="保护环境，低碳出行 ";
    Information_three="珍爱生命，请减速慢行";
    Information_four="前方路段行驶通畅  ";
    addInformation(Information_one);
    addInformation(Information_two);
    addInformation(Information_three);
    addInformation(Information_four);
}
private void addInformation(String information){
    Information.add(information);
}
/*
 * 返回规划好的友情提示
 */
public List<String> getAllInformation(){
    return Information;
}
```

3) 当接收到环境节点发送的数据就更新情报版数据（发送到相应的处理方法中处理）

HomeThread.java 文件 com/uicc/amarttraffic/mode/HomeThread.java

private synchronized void DataFunction(String str){}

```
vate synchronized void DataFunction(String str){
 Dataaddress=Integer.parseInt(str.substring(9, 11), 16);
 if(0x80<=Dataaddress&&Dataaddress<=0x8f){
     //情报版
 }else if(0x90<=Dataaddress&&Dataaddress<=0x9f){
     //环境采集
     Environment=str.toString().split(" ");
     //温湿度
     if(Environment.length>=11&&Environment[3].equalsIgnoreCase("90")&&Environment[5]
         mTemprature=Environment[8];//温度
         mHumility=Environment[9];//湿度
         StrMessage.append(mTemprature);
         StrMessage.append(">");
         StrMessage.append(mHumility);
         mMyHandler.obtainMessage(Present.MESSAGE_DATA_ENVIRONMENT,StrMessage.toStrir
     }else{}
 }else {}
```

把接收到的数据发送给相应的函数进行处理：

private Handler mMyHandler(){}

```
/*
 * 根据不同的条件处理不同的数据
 */
private Handler mMyHandler(){
    if(mMyHandler==null){
        mMyHandler=new Handler(){
            public void handleMessage(Message msg) {
                switch (msg.what) {
                case Present.MESSAGE_DATA_ENVIRONMENT:
                    if(informationMode==null){
                        informationMode=new InformationMode(context,mMyHandler);
                        String data=msg.obj.toString();
                        informationMode.SendInformation(data);
                    }else{
                        String data=msg.obj.toString();
                        informationMode.SendInformation(data);
                    }
                default:
                    break;
                }
            }
        };
    }
    return mMainHandler;
}
```

4) 把接收到的环境节点数据和温馨提示语发送给情报板（进行数据相应的处理）

InformationMode.java 文件com/uicc/amarttraffic/mode/InformationMode.java

```
/*
 * 进行温湿度的处理，给情报板发送当前温湿度的信息
 */
public void SendInformation(String data){
    Information=data.split(">");
    mTemprature=Integer.parseInt(Information[0],16);
    mHumility=Integer.parseInt(Information[1],16);
    if((mTemprature&0x80)==0){//判断当前的温湿度，是否为零下
        if((mTemprature&0x7F)>=10){
            content="温度:"+(mTemprature&0x7F)+" ℃湿度:"+mHumility+"%RH";
        }else{
            content="温度:"+(mTemprature&0x7F)+"℃湿度:"+mHumility+"%RH";
        }
    }else{
        if((mTemprature&0x7F)>=-10){
            content="温度:-"+(mTemprature&0x7F)+" ℃湿度:"+mHumility+"%RH";
        }else{
            content="温度:-"+(mTemprature&0x7F)+"℃湿度:"+mHumility+"%RH";
        }
    }
    GetInformation();//设置友情提示
    try {
        SendContent=contenthint.getBytes("GBK");
        TextContext=content.getBytes("GBK");
    } catch (UnsupportedEncodingException e) {
```

```
for(int i=0;i<2;i++){//给情报板发送数据℃
    Reader.SendInformationData(0x80+i,0x01,0x01, Utility.Bytes2Ints(TextContext));
    try {
        Thread.sleep(200);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    Reader.SendInformationData(0x80+i,0x02,0x01, Utility.Bytes2Ints(SendContent));
    Util.ThreadSleep();
}
```

5) 具体代码请参考项目案例 AmartTraffic_Information 工程

6、 智能红绿灯实验

6.1、 实验目的

1. 熟悉 UI-IOT-ITS 智能交通物联网技术教学实验平台的使用方法。
2. 熟悉 Android 应用程序开发的编译下载流程
3. 学习 UI-IOT-ITS 智能交通物联网通信协议。

6.2、 实验内容

通过 UI-IOT-ITS 智能交通物联网技术教学实验平台实现对智能小车在沙盘上运行通过智能红绿灯的相关操作流程

6.3、 实验仪器

（1） 本实验所需仪器：UI-IOT-ITS 智能交通物联网技术教学实验平台。

（2） 安装了 Eclipse 和 Android SDK 的电脑一台

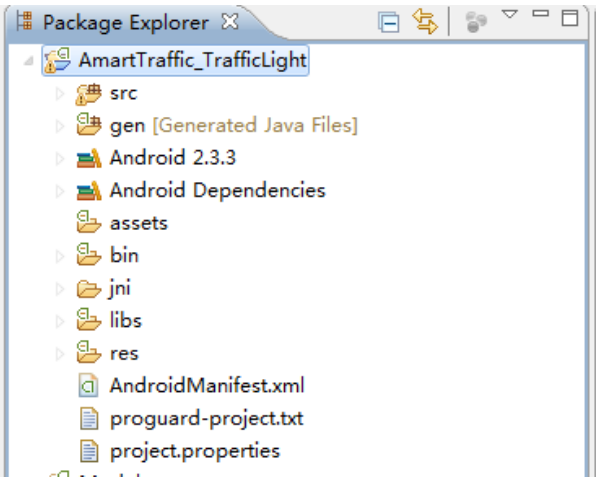
6.4、 实验步骤

1. 建一个Android工程

在Eclipse中，点击菜单[File] – [New] – [Project...]。弹出新建工程窗口。

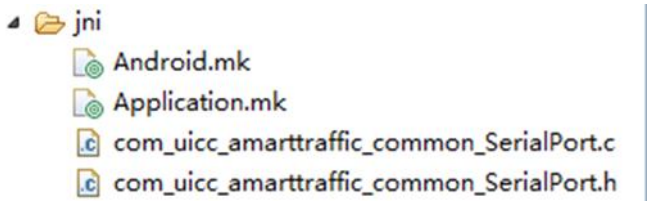
选择工程类型为“Android Project”，点击“Next >”按钮，进入工程信息配置窗口

项目结构如下图所示：



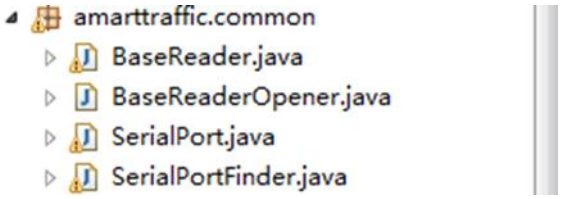
2. 通过JNI方法调用C语言写的打开,关闭串口方法。

1) 在新建的项目中新建一个jni的Folder文件夹，往里面导入如下文件，可从提供的实例的相应位置copy进去



2) 在项目的libs文件夹中导入如下文件，可从提供的实例的相应位置copy进来

3) 在项目src文件下，新建一个package，包名为com.uicc.amarttraffic.common；在本包名下导入如下java文件(串口的打开，关闭，读写操作的实例化)



3. 编辑程序界面

例如：双击工程子目录[res]-[layout]-[activity_main.xml]打开资源文件activity_main.xml。

点击右侧编辑区下方的选项卡“activity_main.xml”进入文本编辑模式，修改文件中的标签内容

可参照提供实训案例AmartTraffic_TrafficLight中activity_main.xml编写。

4. 软件编程

1) 小车位置的获取以及串口数据的获取上面以作讲解这里不在重复，见认知实验<智能小车定位>

2) 根据红绿灯节点发送的数据进行红绿灯在界面的绘制（默认情况固定模式，遵循正常的红绿灯工作模式）

HomeThread.java 文件 com/uicc/amarttraffic/mode/HomeThread.java

private synchronized void DataFunction(String str){}

```
rate synchronized void DataFunction(String str){
 Dataaddress=Integer.parseInt(str.substring(9, 11), 16);
 if(0x10<=Dataaddress&&Dataaddress<=0x1f){
     //判断红绿灯当前状态
         TrafficLight=str.toString().split(" ");
         if(TrafficLight.length>=9&&TrafficLight[3].equalsIgnoreCase("10")&&TrafficLi
             mMyHandler.obtainMessage(Present.MESSAGE_DATA_HINT_TRAFFICLIGHT, Traffic
         }else{
         }
         if(TrafficLight.length==9&&TrafficLight[3].equalsIgnoreCase("10")&&TrafficLi
             if(TrafficLight[7].equalsIgnoreCase("11")){
                 Log.e(TAG, "红绿灯工作模式设置成功");
             }else if(TrafficLight[7].equalsIgnoreCase("12")){
                 Log.e(TAG, "控制红绿灯通行方向成功");
                 mMyHandler.obtainMessage(Present.MESSAGE_DATA_TRAFFICLIGHT).sendToTa
             }
         }else{

         }
```

3) 通过Handler把接收到的数据传递给相应的函数进行处理。

HomeThread.java 文件 com/uicc/amarttraffic/mode/HomeThread.java

private Handler mMyHandler(){}

```
private Handler mMyHandler(){
    if(mMyHandler==null){
        mMyHandler=new Handler(){
            public void handleMessage(Message msg) {
                switch (msg.what) {
                case Present.MESSAGE_MODE_TRAFFICLIGHT:
                    if(trafficLightMode==null){
                        trafficLightMode=new TrafficLightMode(context,mMyHandler
                        trafficLightMode.TlightData=msg.obj.toString();
                        trafficLightMode.TrafficLight();
                    }else{
                        trafficLightMode.TlightData=msg.obj.toString();
                        trafficLightMode.TrafficLight();
                    }
                    break;
                case Present.MESSAGE_MODE_TRAFFIC:
                    if(trafficMode==null){
                        trafficMode=new TrafficMode(context, mMainHandler);
                        String data=msg.obj.toString();
                        trafficMode.JData(data);
                    }else{
                        String data=msg.obj.toString();
                        trafficMode.JData(data);
                    }
                    break;
```

4) 判断小车是否到达红绿灯路口(并把小车的节点地址存储下来 ,控制小车的启动与停止)

TrafficLightMode.java 文件 com/uicc/amarttraffic/mode/TrafficLightMode.java

a) 存储到达红绿灯路口的小车信息

public synchronized void AddCarSet(int NodeAddress, String CarCoord) {}

```
/*
 * 所有即将通过红绿灯小车信息
 */
public synchronized void AddCarSet(int NodeAddress, String CarCoord) {
    if (CarCoord.equalsIgnoreCase("NS-1")|CarCoord.equalsIgnoreCase("NS-2")) {
        if(!NSCar.contains(NodeAddress)){
            NSCar.add(NodeAddress);
        }
        CarMoveTrafficLight(NodeAddress,CarCoord);// 小车通过红绿灯
    } else if (CarCoord.equalsIgnoreCase("EW-1")|CarCoord.equalsIgnoreCase("EW-2"
        if(!EWCar.contains(NodeAddress)){
            EWCar.add(NodeAddress);
        }
        CarMoveTrafficLight(NodeAddress,CarCoord);// 小车通过红绿灯
    } else {
    }

}
```

b) 根据不同的红绿灯状态绘制不同的界面

public void TrafficLightState(String state){}

```
/*
 * 根据红绿灯不同状态绘制界面
 */
public void TrafficLightState(String state){
    NodeState=state;
    MyView.trafficlightmanager.changeAllTrafficLights();//清空以前绘制红绿灯状态
    try {
        Thread.sleep(50);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    if(state.equalsIgnoreCase("00")){//南北方向通行
        Myhandler.obtainMessage(Present.MESSAGE_DATA_NS_TRAFFICLIGHT).sendToTarget(
    }else if(state.equalsIgnoreCase("01")){//南北方向黄灯
        Myhandler.obtainMessage(Present.MESSAGE_DATA_NSY_TRAFFICLIGHT).sendToTarget
    }else if(state.equalsIgnoreCase("02")){//东西方向通行
        Myhandler.obtainMessage(Present.MESSAGE_DATA_EW_TRAFFICLIGHT).sendToTarget(
    }else if(state.equalsIgnoreCase("03")){//东西方向黄灯
        Myhandler.obtainMessage(Present.MESSAGE_DATA_EWY_TRAFFICLIGHT).sendToTarget
    }else{}
    CarMoveTrafficLight(0,null);// 小车通过红绿灯
}
```

c) 判断当前的红绿灯状态，进而判断小车的运行情况

```
private synchronized void CarMoveTrafficLight(int NodeAddress,String CarCoord)
{}
```

```
/*
 * 根据小车节点地址，让小车通过红绿灯路口(根据红绿灯状态)
 */
private synchronized void CarMoveTrafficLight(int NodeAddress,String CarCoord) {
    if (!NodeState.equalsIgnoreCase("-1")) {
        if (NodeState.equalsIgnoreCase("00")
                || NodeState.equalsIgnoreCase("01")) {
            // 如果南北通行，南北方向小车前进，东西方向小车停止
            if(!EWCar.isEmpty()){
                EwState_stop=true;
            }
            if(!EWCar.isEmpty()){//，东西方向小车停止
                new Thread(new Runnable() {

                    @Override
                    public void run() {
                        // TODO Auto-generated method stub
                        while(EwState_stop){
                            for(int i=0;i<EWCar.size();i++){
                                mReader.CarMoveData(EWCar.get(i), 0x01);
                                EwCarState_stop=true;
                            }
                            Util.ThreadSleep();
                        }
                    }
                }).start();
```

d) 选择不同的红绿灯工作模式，进行不同的红绿灯工作状态 如动态调整模式（根据车流量控制红绿灯的通行时间）。

```
/*
 * 动态调整模式，进行红路灯时间调节
 */
public void CarFlow(String direction){
    if(share.getTRAFFICLIGHT()==1){
        if(direction.equalsIgnoreCase("NS")){
            mReader.TrafficLightTime(0x10,0x0A+share.getLENGTHEN_TIME(),0x0A);
        }else if(direction.equalsIgnoreCase("EW")){
            mReader.TrafficLightTime(0x10,0x0A,0x0A+share.getLENGTHEN_TIME());
        }else if(direction.equalsIgnoreCase("EQUAL")){
            mReader.TrafficLightTime(0x10, 0x0A, 0x0A);
        }else{}
    }else{}
}
```

5) 红绿灯对象化，有利于红绿灯绘制控制

TrafficLightManager.java 文件 com/uicc/amarttraffic/ manager/TrafficLightManager.java

```
/*
 *红绿灯界面管理
 */
public class TrafficLightManager {
    public static List<Traffic_Light> TrafficLights =new ArrayList<Traffic_Light>();
    private List<Traffic_Light> lists = new ArrayList<Traffic_Light>();
    public void addTrafficLights(Traffic_Light TrafficLight){
        TrafficLights.add(TrafficLight);
    }
    public List<Traffic_Light> getAllTrafficLights(){
        return TrafficLights;
    }
    /*
     * 绘制红绿灯
     */
    public synchronized void drawAllTrafficLights(Canvas canvas){
        for(Traffic_Light t : TrafficLights){
            t.drawSelf(canvas);
        }
    }
```

6) 通过自定义 View，绘制出当前红绿灯的状态

MyView.java 文件 com/uicc/amarttraffic/ activity/MyView.java

```
public MyView(Context context, AttributeSet set) {
    super(context, set);
    paint=new Paint();
    carmanager=new CarManager();
    trafficlightmanager=new TrafficLightManager();
    new Mythread().start();
}
@Override
protected void onDraw(Canvas canvas) {
    // TODO Auto-generated method stub
    super.onDraw(canvas);
        carmanager.drawAllCars(canvas);
        trafficlightmanager.drawAllTrafficLights(canvas);
}
```

7) 具体代码请参考项目案例 AmartTraffic_TrafficLight 工程

7、 智能路灯实验

7.1、 实验目的

（1）熟悉 UI-IOT-ITS 智能交通物联网技术教学实验平台的使用方法。

（2）熟悉 Android 应用程序开发的编译下载流程

（3）学习 UI-IOT-ITS 智能交通物联网通信协议。

7.2、 实验内容

通过 UI-IOT-ITS 智能交通物联网技术教学实验平台实现对智能小车在沙盘上运行对智能路灯的相关操作流程

7.3、 实验仪器

（3） 本实验所需仪器：UI-IOT-ITS 智能交通物联网技术教学实验平台。

（4） 安装了 Eclipse 和 Android SDK 的电脑一台

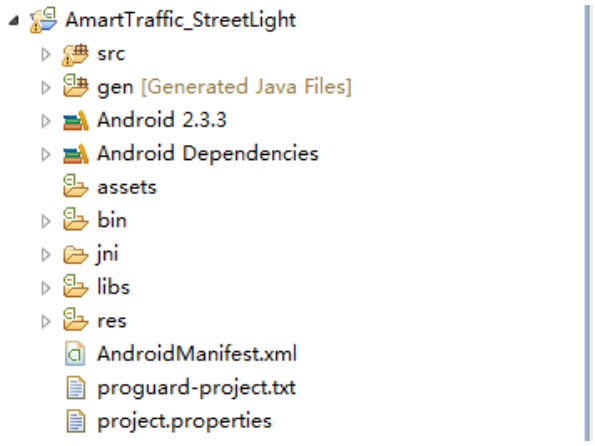
7.4、 实验步骤

1. 建一个Android工程

在Eclipse中，点击菜单[File] – [New] – [Project...]。弹出新建工程窗口。

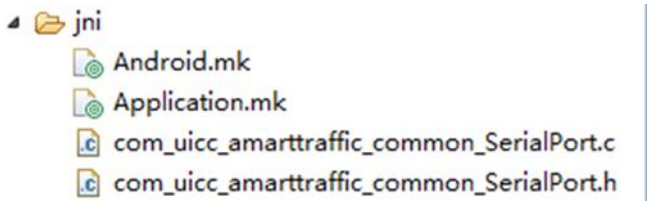
选择工程类型为“Android Project”，点击“Next >”按钮，进入工程信息配置窗口

项目结构如下图所示：



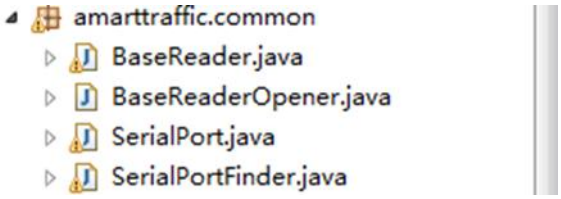
2. 通过JNI方法调用C语言写的打开,关闭串口方法。

1) 在新建的项目中新建一个jni的Folder文件夹，往里面导入如下文件，可从提供的实例的相应位置copy进去



2) 在项目的libs文件夹中导入如下文件，可从提供的实例的相应位置copy进来

3) 在项目src文件下，新建一个package，包名为com.uicc.amarttraffic.common；在本包名下导入如下java文件(串口的打开，关闭，读写操作的实例化)



3. 编辑程序界面

例如：双击工程子目录[res]-[layout]-[activity_main.xml]打开资源文件activity_main.xml。

点击右侧编辑区下方的选项卡“activity_main.xml”进入文本编辑模式，修改文件中的标签内容

可参照提供实训案例 AmartTraffic_StreetLight 中activity_main.xml编写。

4. 软件编程

1) 小车位置的获取以及串口数据的获取上面以作讲解这里不在重复，见认知实验<智能小车定位>

2) 根据智能路灯节点发送数据进行，智能路灯的工作控制(默认情况下是整体控制模式)

HomeThread.java 文件 com/uicc/amarttraffic/mode/HomeThread.java

private synchronized void DataFunction(String str){}

```
ivate synchronized void DataFunction(String str){
  Dataaddress=Integer.parseInt(str.substring(9, 11), 16);
  if(0x30<=Dataaddress&&Dataaddress<=0x3f){
      //判断路灯当前状态
          streetlight=str.toString().split(" ");
          if(streetlight.length>=11&&streetlight[3].equalsIgnoreCase("30")&&streetlig
              StrMessage.append(streetlight[8]);
              StrMessage.append(">");
              StrMessage.append(streetlight[9]);
              mMyHandler.obtainMessage(Present.MESSAGE_DATA_SSTREERLIGHT_STATE,StrMes
          }
          if(streetlight.length==9&&streetlight[3].equalsIgnoreCase("30")&&streetligh
              if(streetlight[7].equalsIgnoreCase("11")){
                  Log.e(TAG, "路灯工作模式设置成功");
                  mMyHandler.obtainMessage(Present.MESSAGE_DATA_SSTREERLIGHT_LD_OUT).
              }
          }
```

3) 通过Handler把接收到的数据传递给相应的函数进行处理。

HomeThread.java 文件 com/uicc/amarttraffic/mode/HomeThread.java

private Handler mMyHandler(){}

```
根据不同的条件处理不同的数据
/
ivate Handler mMyHandler(){
  if(mMyHandler==null){
      mMyHandler=new Handler(){
          public void handleMessage(Message msg) {
              switch (msg.what) {
              case Present.MESSAGE_MODE_STREETLIGHT:
                  if(streetLightMode==null){
                      streetLightMode=new StreetLightMode(context, mMyHandler);
                      streetLightMode.SlightData=msg.obj.toString();
                      streetLightMode.StreetLight();
                  }else{
                      streetLightMode.SlightData=msg.obj.toString();
                      streetLightMode.StreetLight();
                  }
                  break;
              case Present.MESSAGE_DATA_SSTREERLIGHT_STATE:
                  if(streetLightMode==null){
                      streetLightMode=new StreetLightMode(context, mMyHandler);
                      String id_state=msg.obj.toString();
                      String[] StreetLight=id_state.split(">");
                      if(StreetLight.length==2){
                          streetLightMode.StreetLight_state(StreetLight[0], StreetLi
                      }
                  }else{
```

4) 如果是整体控制模式，根据智能路灯节点发送数据进行路灯状态的显示

StreetLightMode.java 文件 com/uicc/amarttraffic/mode/StreetLightMode.java

public void StreetLight_state(String StreetLightId,String StreetLightState){}

```
/*
 * 路灯的整体控制工作状态
 */
public void StreetLight_state(String StreetLightId,String StreetLightState){
    if(StreetLightId.equalsIgnoreCase("00")){
        if(StreetLightState.equalsIgnoreCase("00")){
            Myhandler.obtainMessage(Present.MESSAGE_DATA_SSTREERLIGHT_LD_OUT).sendT
        }else if(StreetLightState.equalsIgnoreCase("01")){
            Myhandler.obtainMessage(Present.MESSAGE_DATA_SSTREERLIGHT_LD_IN).sendTo
        }
    }
}
```

5) 如果是节能控制模式，则根据小车的坐标位置进行路灯的打开与关闭操作（通知界面绘制路灯状态）

StreetLightMode.java 文件 com/uicc/amarttraffic/mode/StreetLightMode.java

private synchronized int JSlightData(String SlightData){}

```
     * 如果整体控制模式，则不要控制路灯；节能控制模式，则要控制路灯
     */
    public void StreetLight (){
        if(!share.getSTREETLIGHT()){
                Log.e(TAG, "整体控制模式");
        } else {
            Data=JSlightData(SlightData);
        }
    }
    /*
     * 判断小车在哪个道路位置，控制路灯打开与关闭
     */
    private synchronized int JSlightData(String SlightData){
        System.out.println("JSlightData="+SlightData);
        if(SlightData.equalsIgnoreCase("A1-4")||SlightData.equalsIgnoreCase("A1-3")){
            mReader.StreetLightData(0x30,0x01,0x01);//控制路灯组一打开
            mReader.StreetLightData(0x30,0x0A,0x01);//控制路灯组十打开
            Myhandler.obtainMessage(Present.MESSAGE_DATA_SSTREERLIGHT_LD1_IN).sendToTarg
            Myhandler.obtainMessage(Present.MESSAGE_DATA_SSTREERLIGHT_LD10_IN).sendToTar
            if(SlightData.equalsIgnoreCase("A1-4")){
                mReader.StreetLightData(0x30,0x02,0x00);//控制路灯组二关闭
                Myhandler.obtainMessage(Present.MESSAGE_DATA_SSTREERLIGHT_LD2_OUT).sendT
            }
            return 0x01;
        }else if(SlightData.equalsIgnoreCase("A1-12")||SlightData.equalsIgnoreCase("A1-1
            mReader.StreetLightData(0x30,0x02,0x01);//控制路灯组二打开
```

6) 创建路灯管理类把路灯对象化，有利于路灯绘制管理

StreetLightManager.java 文件 com/uicc/amarttraffic/ manager/ StreetLightManager.java

```
public class StreetLightManager {
    public static List<Street_Light> StreetLights =new ArrayList<Street_Light>();
    private List<Street_Light> lists = new ArrayList<Street_Light>();
    public void addStreetLights(Street_Light StreetLight){
        StreetLights.add(StreetLight);
    }
    public List<Street_Light> getAllStreetLights(){
        return StreetLights;
    }
    /*
     * 绘制路灯
     */
    public synchronized void drawAllStreetLights(Canvas canvas){
        for(Street_Light s : StreetLights){
            s.drawSelf(canvas);
        }
    }
    /*
     * 清空路灯界面
     */
    public synchronized void changeAllStreetLights(int[] number){
        for(int i=0;i<number.length;i++){
            lists.add(StreetLights.get(number[i]));
        }
        StreetLights.removeAll(lists);
    }
```

7) 通过自定义 View 把，路灯的当前状态绘制在界面

```
public MyView(Context context, AttributeSet set) {
    super(context, set);
    paint=new Paint();
    carmanager=new CarManager();
    streetlightmanager=new StreetLightManager();
    new Mythread().start();
}
@Override
protected void onDraw(Canvas canvas) {
    // TODO Auto-generated method stub
    super.onDraw(canvas);
        carmanager.drawAllCars(canvas);
        streetlightmanager.drawAllStreetLights(canvas);
}
```

8) 具体代码请参考项目案例 AmartTraffic_StreetLight 工程

8、 智能停车场实验

8.1、 实验目的

（1）熟悉 UI-IOT-ITS 智能交通物联网技术教学实验平台的使用方法。

（2）熟悉 Android 应用程序开发的编译下载流程

（3）学习 UI-IOT-ITS 智能交通物联网通信协议。

8.2、 实验内容

通过 UI-IOT-ITS 智能交通物联网技术教学实验平台实现对智能小车通过智能 ETC 模块操作流程。

8.3、 实验仪器

（5） 本实验所需仪器：UI-IOT-ITS 智能交通物联网技术教学实验平台。

（6） 安装了 Eclipse 和 Android SDK 的电脑一台

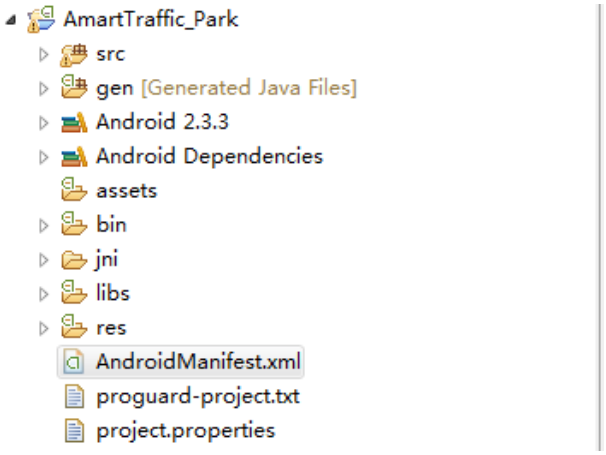
8.4、 实验步骤

1. 建一个Android工程

在Eclipse中，点击菜单[File] – [New] – [Project...]。弹出新建工程窗口。

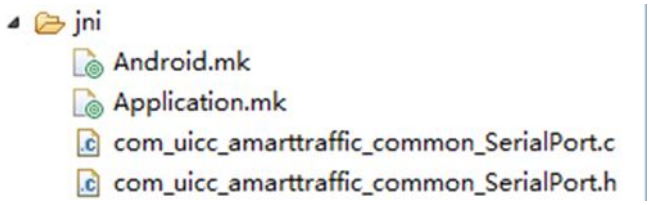
选择工程类型为“Android Project”，点击“Next >”按钮，进入工程信息配置窗口

项目结构如下图所示：



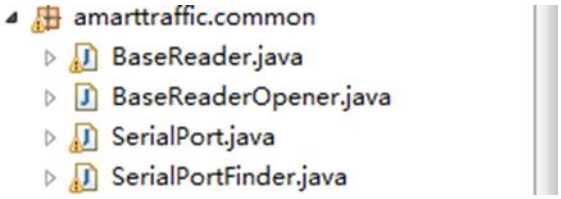
2. 通过JNI方法调用C语言写的打开,关闭串口方法。

1) 在新建的项目中新建一个jni的Folder文件夹，往里面导入如下文件，可从提供的实例的相应位置copy进去



2) 在项目的libs文件夹中导入如下文件，可从提供的实例的相应位置copy进来

3) 在项目src文件下，新建一个package，包名为com.uicc.amarttraffic.common；在本包名下导入如下java文件(串口的打开，关闭，读写操作的实例化)



3. 编辑程序界面

例如：双击工程子目录[res]-[layout]-[activity_main.xml]打开资源文件activity_main.xml。

点击右侧编辑区下方的选项卡“activity_main.xml”进入文本编辑模式，修改文件中的标签内容

可参照提供实训案例 AmartTraffic_Park 中activity_main.xml编写。

4. 软件编程

1) 小车位置的获取以及串口数据的获取上面以作讲解这里不在重复，见认知实验<智能小车定位>

2) 在清单文件中增加相应权限（AndroidManifest.xml）

```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="8" />
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

3) 小车是否进入智能停车场系统（通过判断是否是停车场节点发送的数据进行处理）

HomeThread.java 文件 com/uicc/amarttraffic/mode/HomeThread.java

private synchronized void DataFunction(String str){}

```
private synchronized void DataFunction(String str){
    Dataaddress=Integer.parseInt(str.substring(9, 11), 16);
    if(0x60<=Dataaddress&&Dataaddress<=0x6f){
        //智能停车场
            park=str.toString().split(" ");
            //智能停车场入口
            if(park.length>=15&&park[3].equalsIgnoreCase("60")&&park[5].equalsIgnore
                Nodeaddress=park[3];//智能停车场入口节点地址
                strCarId=park[8];//小车ID号
                strCarType=park[9];//小车类型
                Money=park[10];//小车余额
                StrMessage.append(strCarId);
                StrMessage.append("-");
                StrMessage.append(strCarType);
                StrMessage.append(">");
                StrMessage.append(Money);
                mMyHandler.obtainMessage(Present.MESSAGE_MODE_PARK_IN,StrMessage.toS
                mMyHandler.obtainMessage(Present.MESSAGE_DATA_CAMERA,2).sendToTarget

            }
            //智能停车场出口
            if(park.length>=15&&park[3].equalsIgnoreCase("61")&&park[5].equalsIgnore
                Nodeaddress=park[3];//智能停车场出口节点地址
                strCarId=park[8];//小车ID号
                strCarType=park[9];//小车类型
```

4) 通过Handler把接收到的数据传递给相应的函数进行处理。

HomeThread.java 文件 com/uicc/amarttraffic/mode/HomeThread.java

private Handler mMyHandler(){}

```
private Handler mMyHandler(){
    if(mMyHandler==null){
        mMyHandler=new Handler(){
            public void handleMessage(Message msg) {
                switch (msg.what) {
                case Present.MESSAGE_MODE_PARK_IN:
                    if(parkMode==null){
                        parkMode=new ParkMode(context, mMyHandler);
                        String inPark=msg.obj.toString();
                        parkMode.In_Park(inPark);
                    }else{
                        String inPark=msg.obj.toString();
                        parkMode.In_Park(inPark);
                    }
                    break;
                case Present.MESSAGE_MODE_PARK_OUT:
                    if(parkMode==null){
                        parkMode=new ParkMode(context, mMyHandler);
                        String outPark=msg.obj.toString();
                        parkMode.Out_Park(outPark);
                    }else{
                        String outPark=msg.obj.toString();
                        parkMode.Out_Park(outPark);
                    }
                    break;
```

5) 对进入智能停车场的进行相应的操作

ParkMode.java 文件 com/uicc/amarttraffic/ mode/ ParkMode.java

a) 小车通过智能停车场入口数据的相关操作（接收到智能停车场入口节点数据，判断卡内是余额充足（余额不足自动充值），停车场是否存在停车位；如果余额充足，存在停车位（停车场内存放小车数量大于 2 就会自动启动其中的一辆驶出），网关中心给停车场入口发送当前时间以及小车的停车位位置，并给小车发送启动小车进入停车场以及小车停车位位置;同时保持下小车的节点地址以及相应的停车位信息，小车驶出停车场时用到）

public void In_Park(final String inPark){}

```
/*
 * 停车场进入口数据处理
 */
public void In_Park(final String inPark){
    in_state=true;
    new Thread(new Runnable() {

        @Override
        public void run() {
            // TODO Auto-generated method stub

                if(inPark!=null){
                    in_Park=inPark.split(">");
                    inCaraddress=CarManager.CarIndex.get(in_Park[0]);//获得入口处小车节
                    ParkListAdapter.add("欢迎:"+in_Park[0]+" "+Util.TimeNow());
                    inTime=Util.TimeNow().split(":");
                    if(!inTime[0].isEmpty()&&!inTime[1].isEmpty()&&!inTime[2].isEm
                        Hour=Integer.parseInt(inTime[0]);//小时
                        Minute=Integer.parseInt(inTime[1]);//分钟
                        Second=Integer.parseInt(inTime[2]);//秒
                    }
                    if(inCaraddress!=null){
                    if(!in_Park[1].isEmpty()&&Integer.parseInt(in_Park[1],16)>0){
                        if(BarkParkNumber()!=0){//判断是否有停车位
                            number= BarkParkNumber();//车位编号
```

b) 停车场内存放小车数量大于2就会自动启动其中的一辆驶出，根据进入停车场的进入顺序驶出（先给小车发送小车的行驶路线，再启动小车驶出停车位，并给智能停车场入口模块更新停车场停车位信息）

```
//=======控制启动小车===============
if(parks.size()>=2){
    autoStart=parks.get(0);
    car_number_state=true;
    CarPath(parks.get(0));//给小车发送行车路径
    util.ThreadSleep();
    new Thread(new Runnable() {

        @Override
        public void run() {
            // TODO Auto-generated method stub
            while(car_number_state){
                mReader.CarMoveData(parks.get(0), 0x05);//启动小车驶出停车位
                util.ThreadSleep();
                car_number_send=true;
            }
        }
    }).start();
}else{
}
```

c) 当小车上报小车停靠车位成功命令，只能网关发送给智能停车场节点信息更新智能停车场停车位信息。

```
public synchronized void OutParkingSpace(final String Caraddress){}
```

```
/*
 * 小车在车位上
 */
public synchronized void OutParkingSpace(final String Caraddress){
    if(Caraddress!=null){
        if(!parks.contains(Integer.parseInt(Caraddress,16))){
            parks.add(Integer.parseInt(Caraddress,16));//存储在停车成功的小车节点信息
            }
        parkNumber--;
        if(parkNumber<=0){
            parkNumber=0;
        }else if(parkNumber>4){
            parkNumber=4;
        }
        mReader.RemainingSpace(0x60, parkNumber);
        System.out.println("剩余"+parkNumber+"停车位");
        if(homeThread==null){
            homeThread=new HomeThread();
        }
    }
}
```

d) 小车通过智能停车场出口时数据的相关操作（接收到智能停车场出口节点数据，判断卡内是余额充足（余额不足自动充值）；如果余额充足网关中心给停车场出口发送当前时间以及消费金额，并给小车发送驶出停车场出口命令。

```
/*
 * 停车场出口数据处理
 */
public void Out_Park(final String outPark){
    out_state=true;
    new Thread(new Runnable() {

        @Override
        public void run() {
            // TODO Auto-generated method stub

            if(outPark!=null){
                out_Park=outPark.split(">");
                outCaraddress=CarManager.CarIndex.get(out_Park[0]);//获得出口模式下小
                money= Integer.parseInt(Long.toString(Time(out_Park[1])));
                inTime=Util.TimeNow().split(":");
                if(!inTime[0].isEmpty()&&!inTime[1].isEmpty()&&!inTime[2].isEmpt
                    Hour=Integer.parseInt(inTime[0]);
                    Minute=Integer.parseInt(inTime[1]);
                    Second=Integer.parseInt(inTime[2]);
                }
                Log.e(TAG, "PARK消费金额："+EMonetary(money)+" 剩余金额:"+Integer.pars
                if(Integer.parseInt(out_Park[2],16)-EMonetary(money)>10){//余额是否
                        mReader.OutParkTime(0x61, Hour, Minute,Second, EMonetary
                }else{
```

e) 小车停放在智能停车场停车位置上，通过手动启动小车驶出停车场（先给小车发送小车的行驶路线，再启动小车驶出停车位，并给智能停车场入口模块更新停车场停车位信息）

ParkActivity.java 文件 com/uicc/amarttraffic/ ui/ ParkActivity.java

通过触发控件的点击事件，进行小车的运行状态控制。

public void onClick(View v){}

```
public void onClick(View v) {
    // TODO Auto-generated method stub
    switch (v.getId()) {
    case R.id.park_one:
        if(ParkMode.PackID!=null&&ParkMode.PackID.get(1)!=null){
            Log.e(TAG, "一号停车位 小车驶出"+ParkMode.PackID.get(1));
            park_one_state=true;
            address=Integer.parseInt(ParkMode.PackID.get(1),16);//获得停放在一号位的小
            CarPath(ParkMode.PackID.get(1));//给停放的小车发生，运动路径
            Util.ThreadSleep();
            new Thread(new Runnable() {

                @Override
                public void run() {
                    // TODO Auto-generated method stub
                    while(park_one_state){
                        mReader.CarMoveData(address, 0x05);//启动停放在一号位的车，驶出停
                        Util.ThreadSleep();
                        park_one_send=true;
                    }
                }
            }).start();
        }else{
            Toast.makeText(this, "没有小车停放一号位！",Toast.LENGTH_SHORT).show();
        }
        break;
```

6) 智能小车驶入智能停车场，进行监控拍照并把图片保存

调用模拟摄像头进行拍照功能（MainActivity.java 进行摄像头的实例化以及图片的存储）

```
case Present.MESSAGE_DATA_CAMERA:
    String id=msg.obj.toString();
    setCameraId(Integer.parseInt(id));
    mCameraID = Integer.parseInt(id);
    Util.ThreadSleep();
    takepicture(id);
    break;
```

```
/**
 * 拍照并保存图片
 */
public void takepicture(final String str){
    mCamera.takePicture(null, null, new PictureCallback() {

        public void onPictureTaken(byte[] data, Camera camera) {
            // TODO Auto-generated method stub
            if(str.equalsIgnoreCase("2")){
                File file=new File("mnt/sdcard/UICC/PARK");
                if(!file.exists()){
                    file.mkdirs();
                }
                File filejpg=new File("mnt/sdcard/UICC/PARK/UICC"+System.curr
                savePictuer(filejpg, data);
            }else {}
        }
    });
}
```

7) 智能停车场监控图片，历史记录查看（通过图片的存储位置把监控到的监控图片显

示出来）

ParkHistoryActivity.java　文件 com/uicc/amarttraffic/ ui/ ParkHistoryActivity.java

```
/*
 *停车场历史记录
 */
public class ParkHistoryActivity extends Activity {
    GridView gv;
    private ParkListAdapter adapter;
    List<String> lis;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.park_history_main);
        ActivityManager.getActivityManager().addActivity(this);
        lis=new Util().getSD("PARK");//把存储图片的地址取出来
        if(lis.isEmpty()){
            Toast.makeText(this,"没有历史记录",Toast.LENGTH_SHORT).show();
        }
        GridView gv = (GridView) this.findViewById(R.id.photo_gridView);
        adapter = new ParkListAdapter(this);
        gv.setAdapter(adapter);// 为GridView设置数据适配器
        gv.setOnItemClickListener(new AdapterView.OnItemClickListener() {

                public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                        long arg3) {
                    // TODO Auto-generated method stub
                    String map=lis.get(arg2);
```

8)　具体代码请参考项目案例 AmartTraffic_Park 工程

2.1 Android 开发环境的配置

工具/原料

- Eclipse 、Java Jdk6、Android SDK

步骤/方法

1. **安装 Java Jdk**

第二步：(可选)修改 Java Jdk 安装路径，这样便于环境变量配置。(也可以按照默认安装路径安装，只是配置环境变量时按照此路径即可)

第三步：点击“下一步”

第四步：(可选)修改 jre 安装路径，这样便于环境变量配置。(也可以按照默认安装路径安装，只是配置环境变量时按照此路径即可)。

第五步：点击“下一步”

第六步：点击“完成


Java(TM) SE Development Kit 6 Update 10 - 许可证
许可证协议
请仔细阅读下面的许可证协议。
Pre-Release Software Evaluation Agreement
SUN MICROSYSTEMS, INC. ("SUN") IS WILLING TO LICENSE
THE JAVA SE DEVELOPMENT KIT (JDK), VERSION 6
PRE-RELEASE SOFTWARE TO LICENSEE ONLY UPON THE
CONDITION THAT LICENSEE ACCEPTS ALL OF THE TERMS
CONTAINED IN THIS LICENSE AGREEMENT ("AGREEMENT").
PLEASE READ THE TERMS AND CONDITIONS OF THIS AGREEMENT
CAREFULLY. BY DOWNLOADING OR INSTALLING THIS SOFTWARE,
LICENSEE ACCEPTS THE TERMS AND CONDITIONS OF THIS
拒绝(D)
接受(A) >


Java(TM) SE Development Kit 6 Update 10 - 自定义安装
自定义安装
选择要安装的程序功能。
请从下面的列表中选择要安装的可选功能。安装完成后，您可以使用"控制面板"中的"添加/删除程序"实用程序来更改您选择的功能
功能说明
Java(TM) SE Development Kit 6 Update 10，包括专用 JRE 6 Update 10。这将需要 300 MB 的硬盘驱动器空间。
开发工具
演示程序及样例
源代码
公共 JRE
Java DB
安装到：
C:\Program Files\Java\jdk1.6.0_10\
更改(A)...
< 上一步(B)
下一步(N) >
取消


Java(TM) SE Development Kit 6 Update 10 - 进度
正在安装
正在安装您选择的程序功能。
安装向导正在安装 Java(TM) SE Development Kit 6 Update 10，请稍候。
这可能需要持续几分钟时间。
状态：
正在复制新文件
Baidu 经验
jingyan.baidu.com


Java(TM) SE Development Kit 6 Update 10 - 进度
正在安装
正在安装您选择的程序功能。
安装向导正在安装 Java(TM) SE Development Kit 6 Update 10，请稍候。
这可能需要持续几分钟时间。
Java 安装 - 目标文件夹
目标文件夹
单击"更改..."以将 Java 安装到其他文件夹
安装到：
C:\Program Files\Java\jre6\
更改(A)...
取消
下一步(N) >
Baidu 经验


Java(TM) SE Development Kit 6 Update 10 - 进度
正在安装
正在安装您选择的程序功能。
安装向导正在安装 Java(TM) SE Development Kit 6 Update 10，请稍候。
这可能需要持续几分钟时间。
Java 安装 - 进度
正在安装 Java
这可能会花费几分钟时间
状态：
现在，您可以免费拥有一个与 Microsoft Office 兼容的功能全面的办公套件
• 一组功能强大、集成在一起的字处理、电子表格、演示文稿、绘图和数据库应用程序
• 读取、编辑和保存 Microsoft Office 文件
• 支持 70 多种语言以及 Solaris、Windows、Linux 和 Mac 操作系统
• 使用行业标准的开放文件格式（OpenDocument）作为默认文件格式
• 内置的一键式 PDF 导出
Writer
Calc
Impress
Base
Draw
Math
OpenOffice.org


Java(TM) SE Development Kit 6 Update 10 - 完成
Java(TM) SE Development Kit 6 Update 10
已成功安装
产品注册是免费的，您将获得如下增值服务：
* 获得新版本、修补程序和更新的通知服务
* 获得有关 Sun 开发者产品、服务和培训的优惠
* 获得对早期版本和文档的访问权限
当您单击"完成"后将收集产品与系统信息，同时显示 JDK 产品注册表单。如果您不注册，则不保存以上信息。
有关注册所收集的数据以及这些数据的管理和使用方式的更多信息，
请参见"产品注册信息"页面。
产品注册信息(P)
完成(F)

2. **配置 Java Jdk**

右击 “我的电脑” ->属性->高级->环境变量->系统变量->新建

第一步：

变量名：JAVA_HOME

变量名：C:\jdk1.6.0_10

第二步：

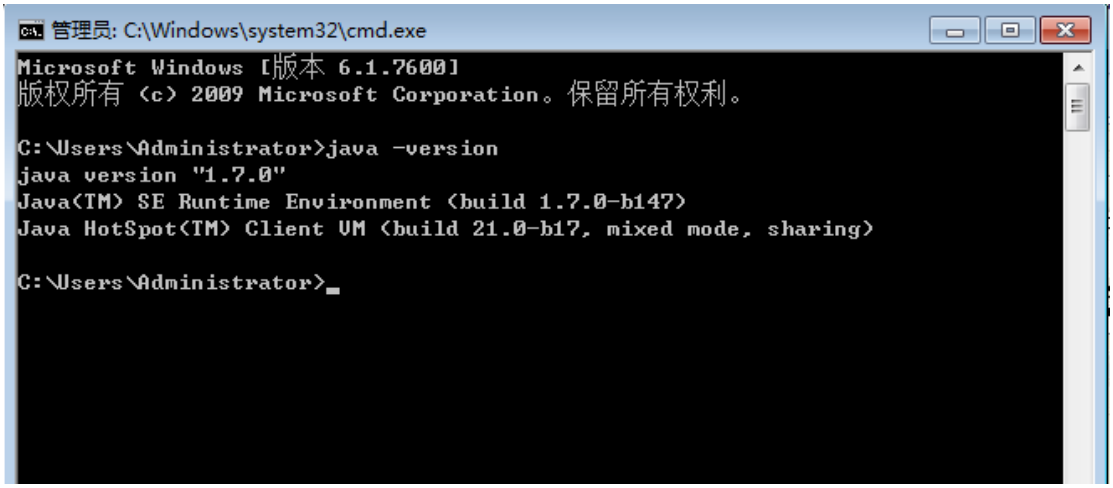
变量名：Path

变量值：%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin

(变量值中如果有内容，用;隔开)

3. **测试 Java Jdk 安装是否成功**

打开 cmd 窗口，输入 java –version 查看 JDK 的版本信息。



4. **安装 Android SDK**

本次安装使用集成安装包（SDK，ADT，Eclipse）

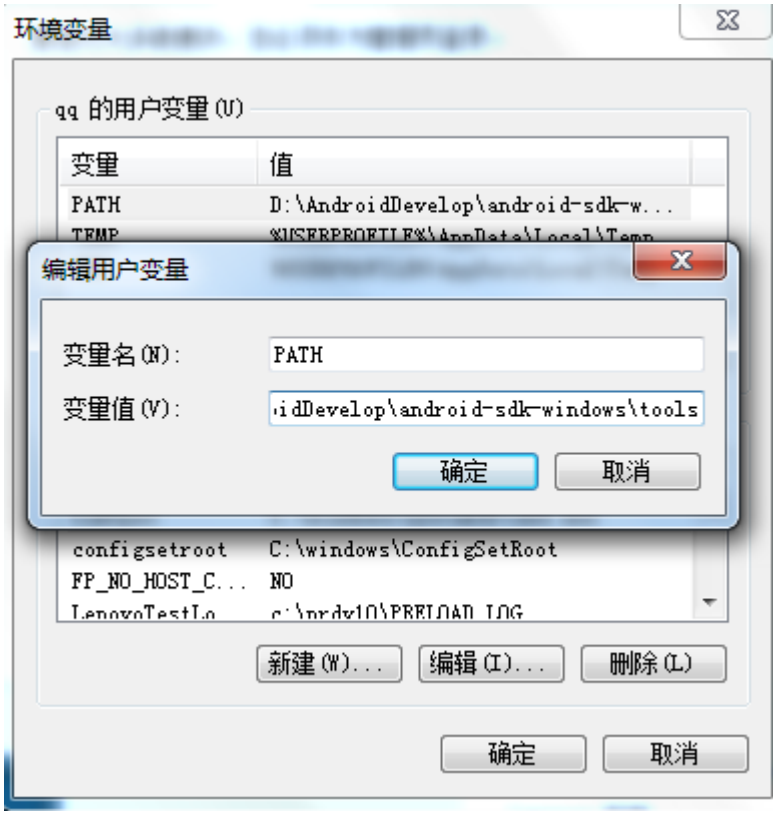
解压提供的 adt-bundle-windows.zip，使用者也可以自行下载安装包

5. **配置 Android SDK**

右击 “我的电脑” ->属性->高级->环境变量->系统变量

变量名：Path

变量值：Android SDK 中的 tools，platform-tools 绝对路径（如本机为 D:\AndroidDevelop\android-sdk-windows\tools）。

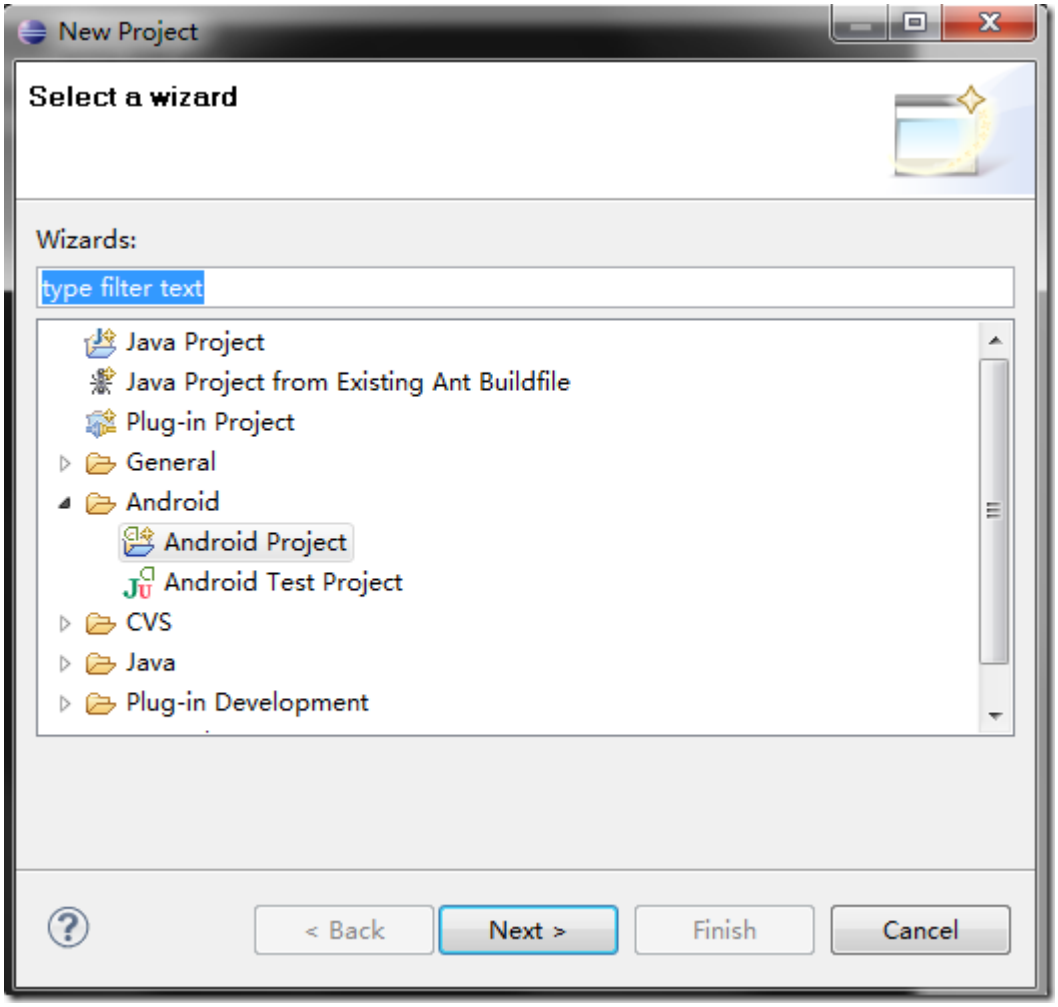


(变量值中如果有内容，用;隔开)

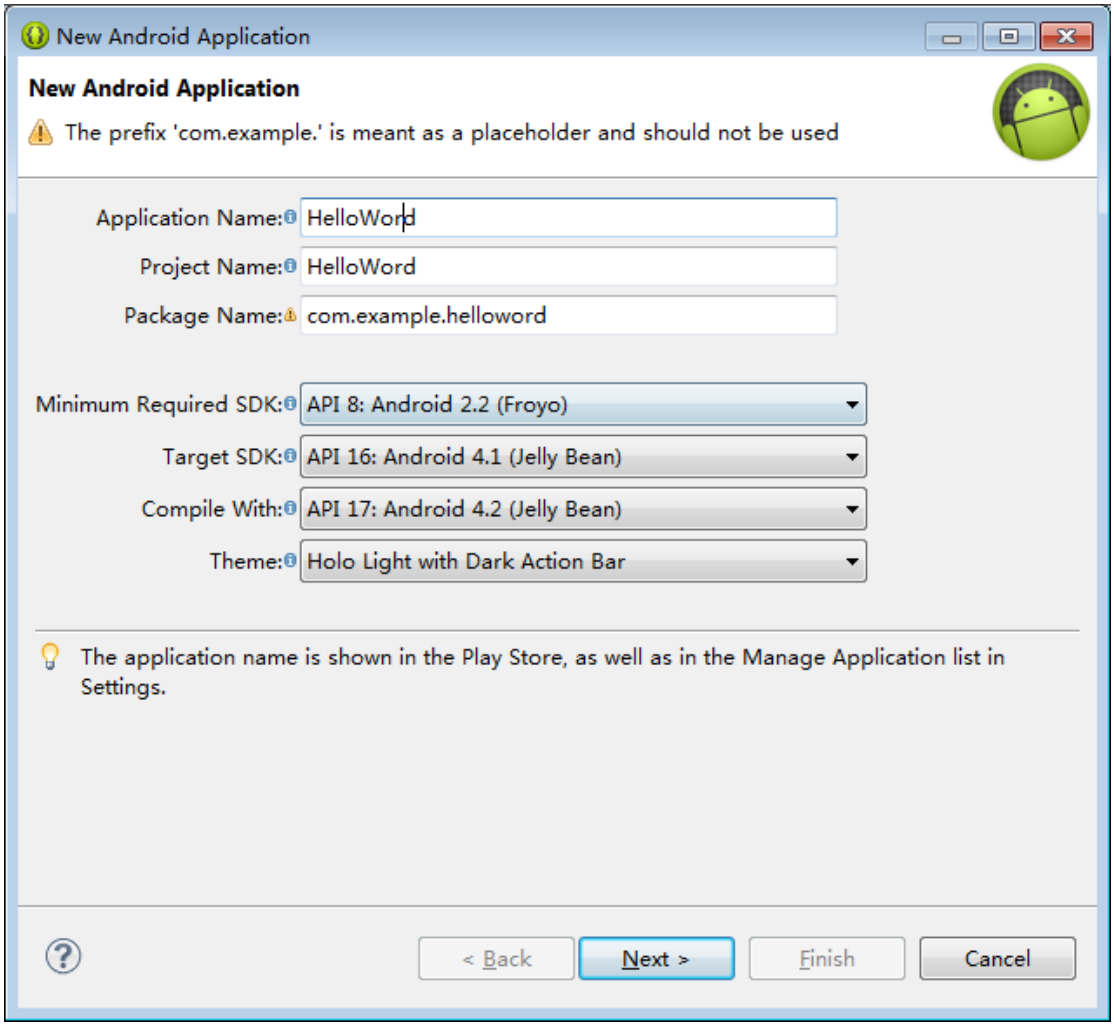
“确定”后，重新启动计算机

6. 新建 Android 工程:

1. 【File】-> 【new】 –> 【Project】->【Android Project】


New Project
Select a wizard
Wizards:
type filter text
Java Project
Java Project from Existing Ant Buildfile
Plug-in Project
General
Android
Android Project
Android Test Project
CVS
Java
Plug-in Development
< Back
Next >
Finish
Cancel

2. 输入项目名称



按顺序介绍以上红色框内内容

a. Project name: 你建立的工程的名称

b. Contents: 主要用于制定你的工程代码的存放路径

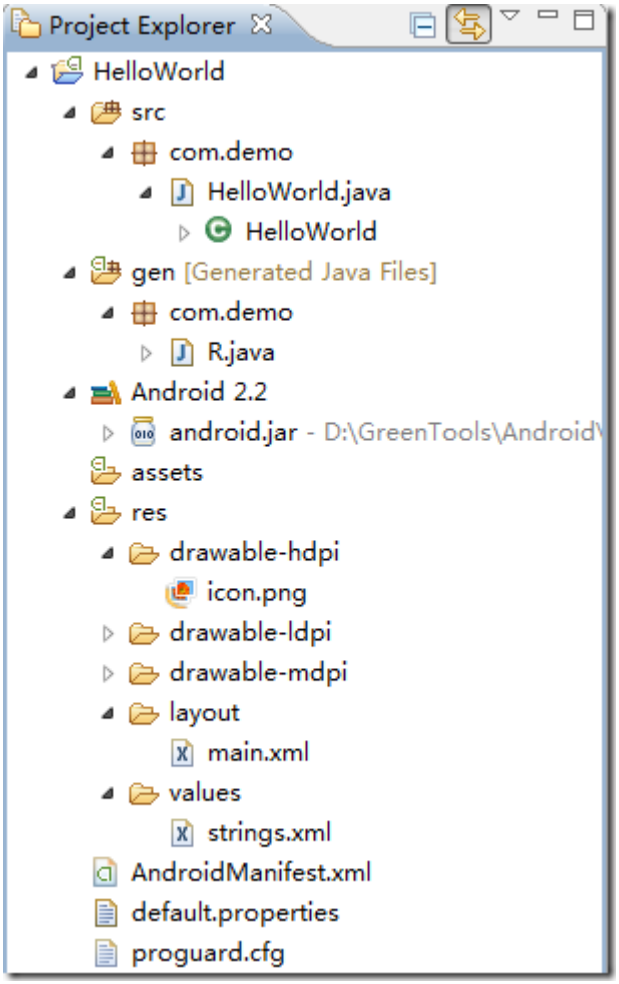
c. Build Target: 说明你要开发基于 Android 那个版本的应用程序

d. Application name: 这个是你开发出来的程序，安装到设备中之后，那个图标下面显示的名字.

e. Package name: 包名称

f. Create Activity: 选择了这个就会生成一个默认的类,

3. 创建完成之后, Eclipse 中就会打开



至此, Android 工程已经建立完毕.

二. 认识项目中各个文件及文件夹

需要注意的组件包括：

src 文件夹

包含示例应用程序的包，即

com.demo

。

gen 文件夹

所有自动生成的文件包含在这里面, 如 R.java

R.java

Android Developer Tools 自动创建这个文件，它提供访问 Android 应用程序的各种资源所需的常量。后面会详细讨论 R 类与资源之间的关系。 此文件是在构建时自动创建的，所以不要手工修改它，因为所有修改都会丢失。

HelloWorld.java

应用程序的主活动类的实现。

Referenced libraries

包含 android.jar，这是 Android SDK 中的 Android 运行时类的 jar 文件。

res 文件夹

Drawables

这个文件夹包含图形文件，比如图标和位图, Drawable-xdpi (hdpi, ldpi, mdpi)分别对应不同分辨率的屏幕, hdpi 对应的是800 * 480及以上,当你的应用程序安装到不同分辨率的机器上的时候他会到对应的文件夹中去读取.

Layouts

这个文件夹包含表示应用程序布局和视图的 XML 文件。后面会详细研究这些文件。

Values

这个文件夹包含 strings.xml 文件。这是为应用程序实现字符串本地化的主要方法。Android UI 遵循了 MVC 开发模式, UI 上用到的字符名称都定义再这个文件夹下。

AndriodManifest.xml

示例应用程序的部署描述符。

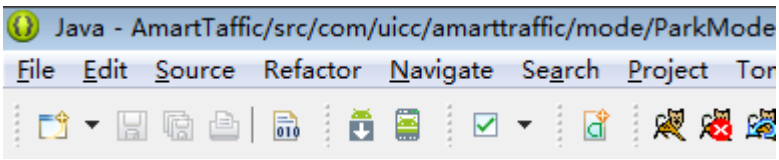
用 **Eclipse** 开发第一个 **Android** 应用程序 **HelloWorld(**下篇**)**

简单调试运行

一. AVD 的创建

AVD 就是指 Android 模拟器

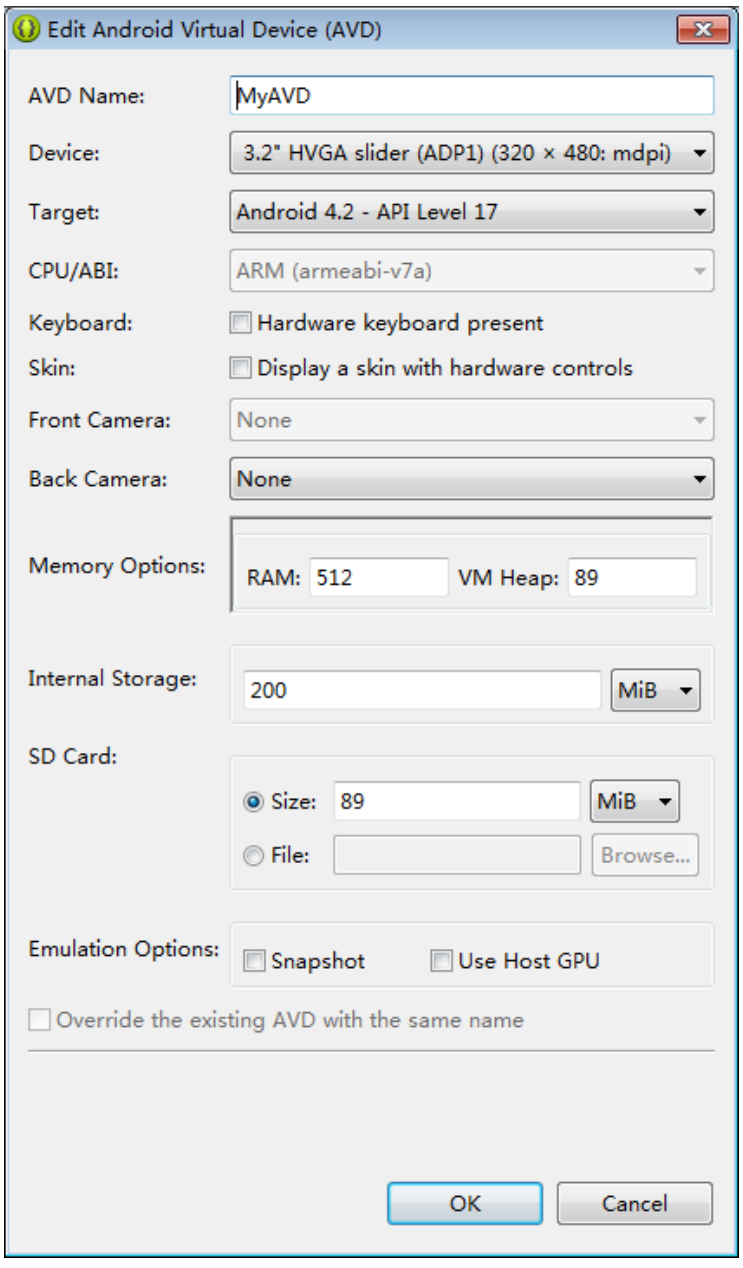
1. 通过 Eclipse 菜单【Window】->【Android SDK and AVD Manager】或者



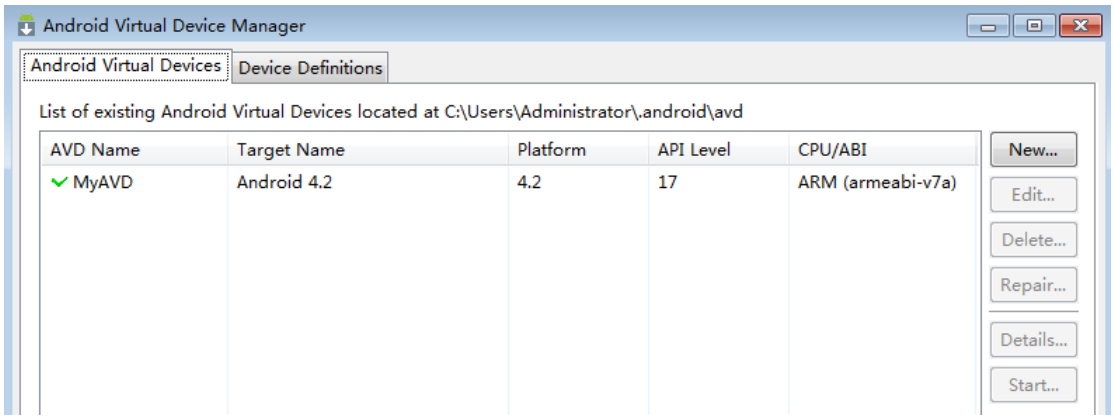
2. 在没有新建过的情况下里面应该是空白的, 以上是我自己新建的两个, 我们在新建的时候基本是按照所对应的 Android 版本来建立, 比如你要开发2.2的程序,则建立2.2对应的 AVD 就好.

Name: 指你建立的模拟器的名称

Target: 对应的 Android SDK 版本



创建完成之后就会多出来一条，如下图:



等待一会儿，让模拟器启动，就可以看见如下界面：

二. 程序的运行

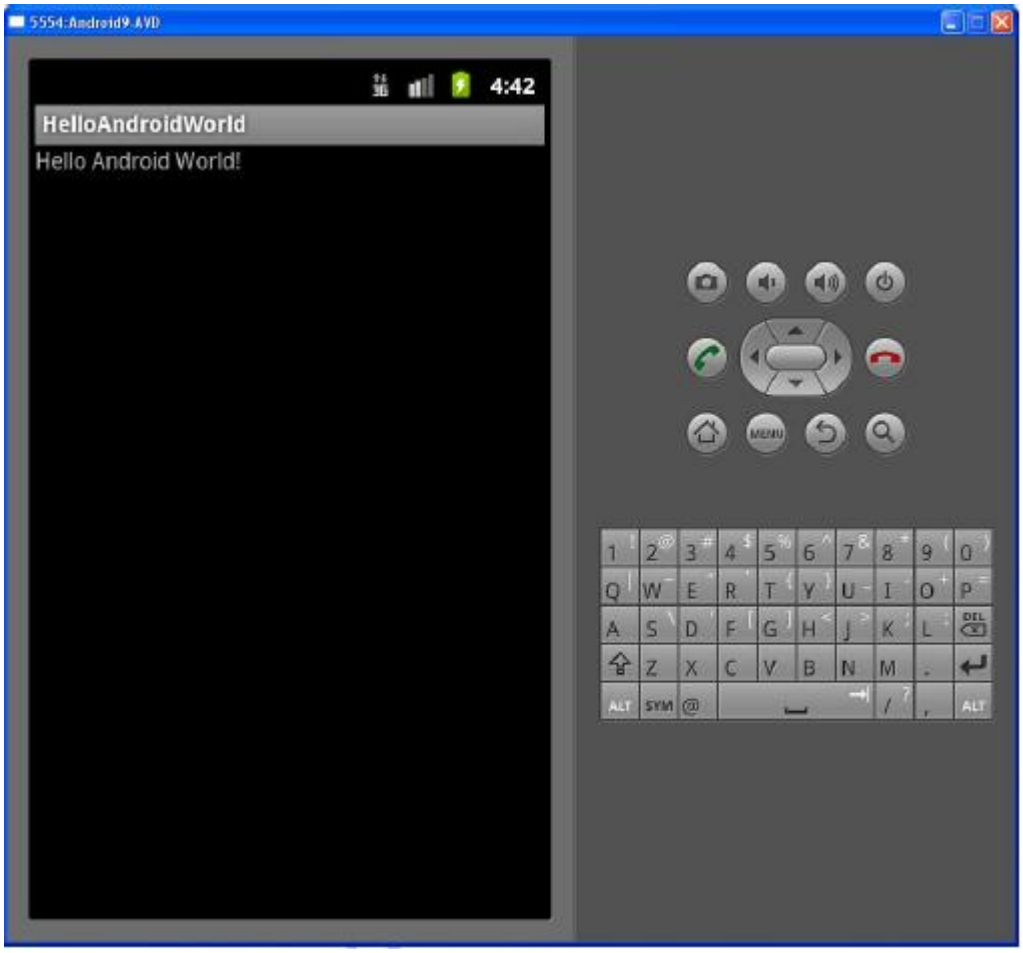
1. 在Project Explorer中右击项目名【Run As】->【Run Configurations...】, 打开 Run Configurations 对话框

使用模拟器的时候有以下几点需要注意:

模拟器开机时间有点长, 所以每天第一次运行需要等很久,具体时间取决于开发电脑额配置,

我们第一遍运行完成之后不需要急着关闭模拟器, 第二次运行程序你只要点 Run 他就会自动把新的程序安装到模拟器,

我们先来看看运行起来的样子吧



智能交通

第三章 AiBall 使用

3.1 AiBall 使用手册

3.1 AiBall 使用手册

Ai-Ball
Quick Start Guide

设备概况

1. Ai-Ball 操作

开关拨到“OFF”关闭电源

开关拨到“ON”打开 Ai-Ball

开关拨到“ ”设置无线 SSID 和密码

2.开始设置 Ai-Ball

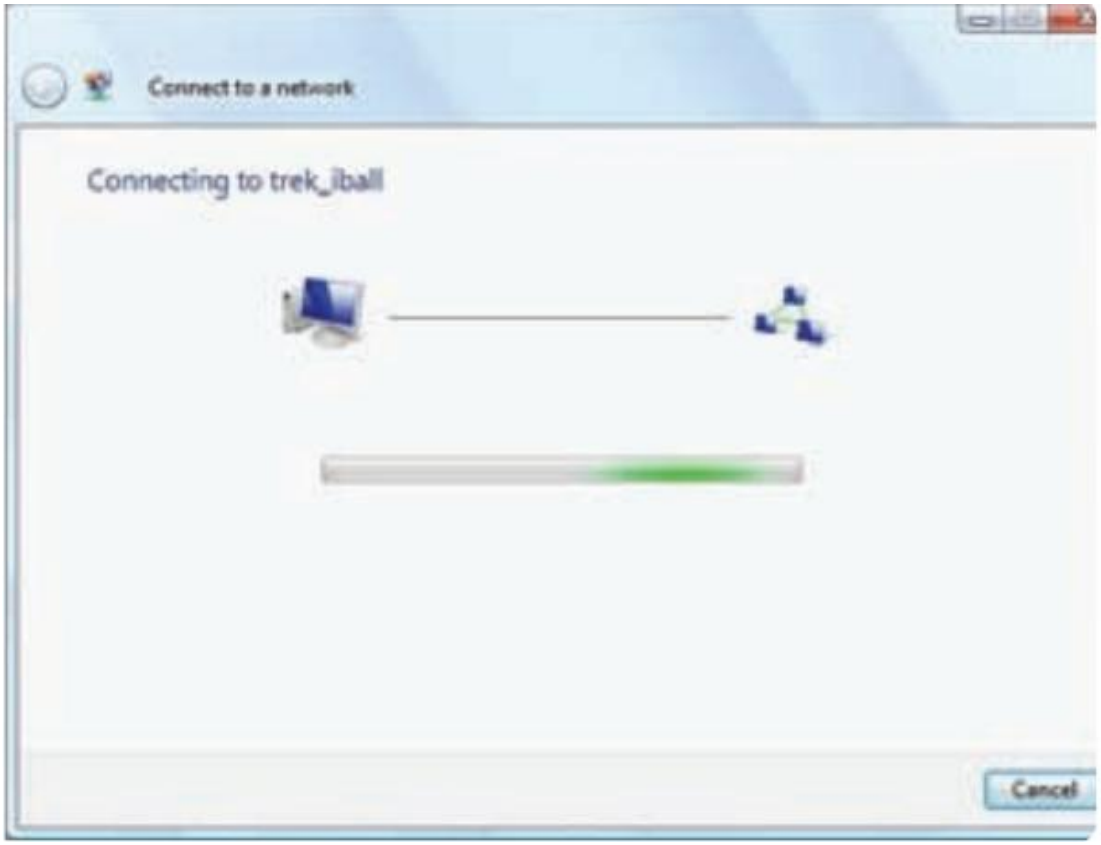
电脑或者带 WIFI 的手机连接 Ai-Ball

1） Ai-Ball 开关拨到″ON″，等待约 10 秒，打开无线网络连接，点击刷新网络列表找到 **SSID：Trek_Ai-Ball**

，

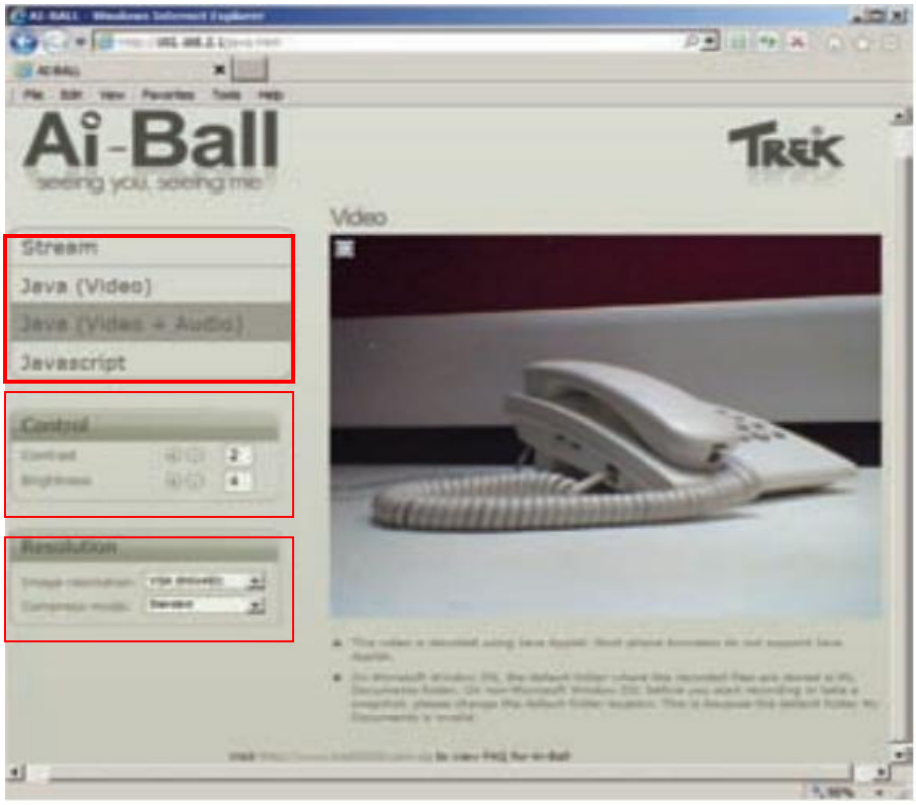
2） 选择 **SSID：Trek_Ai-Ball** 点击**连接**

3）**电脑连接 Ai-Ball**



4）.连接成功后，在 IE 浏览器里输入 http://ai-ball.com 或 http://192.168.2.1 打开以下网址

注意：电脑必须安装 JAVA JDK1.5 版本以上，最好使用火狐浏览器。



3.看和玩

1）依赖你的手机或者电脑配置，选择 Stream ,Java, Javascript 查看视频；

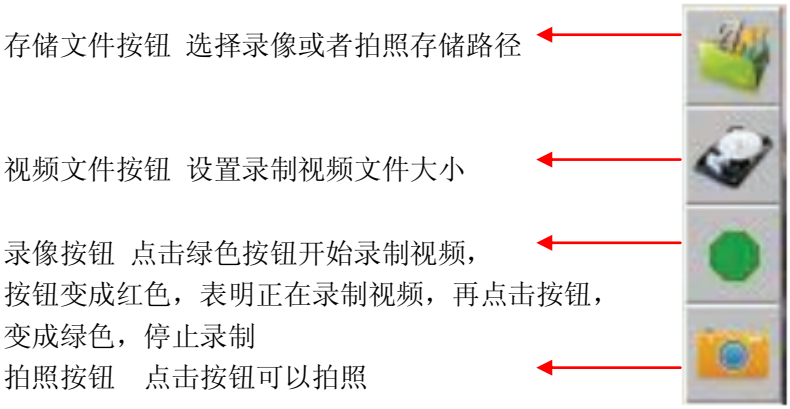
2）你可以选择**控制按钮**，调整对比度和亮度；

3）可以**选择分辨率**：默认图像是 **VGA** 和压缩模式**标准**

4 录像与拍照

1）在浏览器里浏览视频。选择 **Java(Video and Audio)** 方法，查看视频

2）在视频画面的左边，移动鼠标到 图标，看到如下按钮：



配置

1.Ai-Ball 有三种连接方式

1） Micro AP (也叫 uAP)模式

2） Ad-Hoc 模式

3） Infrastructure 模式

默认连接方式是 uAP 模式，在这种模式下可以连接 PC，Android 手机，iPhone 手机或者其他带 Wi-Fi 设备

2. uAP 和 Ad-Hoc 设置

1） 开关拨到设置状态，等待 10 秒左右，PC 或者手机搜索 Ai-Ball 信号

2） 找到 Trek iBall 信号后，点击连接

3） 连接成功后，在浏览器里面输入 http://192.168.2.1 打开网页设置页面

4） 在 Wireless mode 选择 uAP 或者 Ad-Hoc

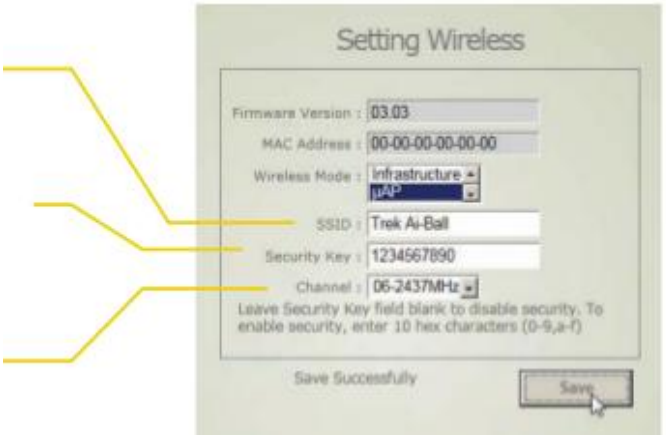
5） 点击 Save 按钮，保存设置。

6） 再把开关拨到 OFF 状态，在拨到 ON 状态

SSID:设置连接时显示的名称（默认 Trek iBall）

密码设置：当你设置密码后，连接 Ai-Ball 时需要输入密码。如果你不需要设置密码，请保留空白。

频段选择：如果你不清楚频段，请使用默认频段



3. Infrastructure 局域网模式设置

1） 开关拨到设置状态，等待 10 秒左右，PC 或者手机搜索 Ai-Ball 信号

2）找到 Trek iBall 信号后，点击连接

3）连接成功后，在浏览器里面输入 http://192.168.2.1 打开网页设置页面

4）在 Wireless mode 选择 Infrastructure

5）点击 Save 按钮，保存设置。

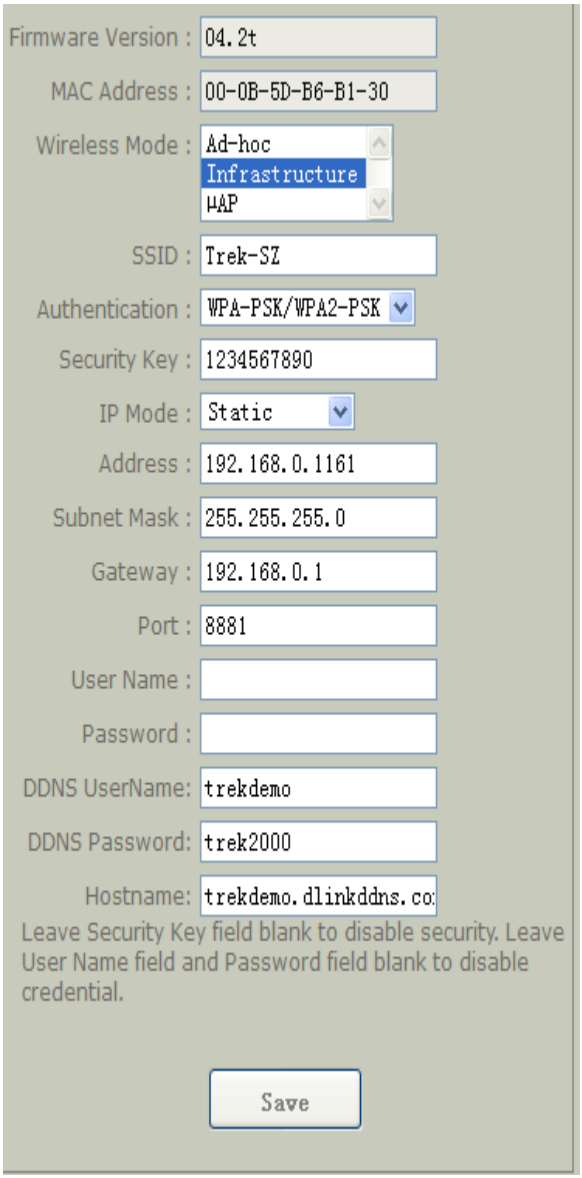
6）再把开关拨到 OFF 状态，在拨到 ON 状态

Firmware Version: 固件版本
MAC Address: MAC 地址
Wireless Mode: Ai-Ball 工作模式，
选择 Infrastructure 模式
SSID:设置 Ai-Ball 要连接路由器的名称（如 Dlink）
Authentication:无线路由器加密认证方式（可以在路由器设置里面查看）
Security Key：设置连接路由器需要的密码

IP Mode:选择 Static 静态 IP

Address:为 Ai-Ball 设置 IP 地址

Port:默端口设置大于 2000
User name / Password:
设置用户名和密码后，在网页里浏览视频需要输入用户名和密码
DDNS UserName:设置在 DDNS 网站注册的用户名
DDNS Password:设置在 DDNS 网站注册的密码
Hostname:DDNS 网站的域名，可以使用这个域名远程访问摄像头



高级配置

Infrastructure 远程模式设置

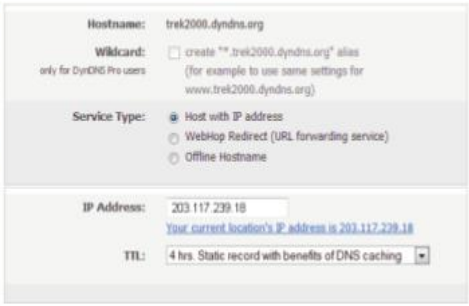
在这种模式下，AI-Ball 通过 AP 接入路由器。可以通过远程访问 Ai-Ball.

1) 设置同 Infrastructure 局域网模式设置

但是 Port 端口不要设置为 80，设置端口范围 2000 到 64000。

User name / Password:不需要设置，保留空白。

2） 你需要申请一个 DDNS 账号（我们使用的是免费的 DynDNS 服务）



3）在 DynDNS 里点击 My Server 配置你的 DDNS；

4）在你的路由器里面配置 DDNS;

5）路由器里面设置虚拟服务器。注意：此处静态 ip，端口号必须和 DynDNS 设置相同

6）在远程你就可以输入 http://trek2000.dyndns.org:8888 访问 Ai-Ball

注意：8888 是端口号

软件更新

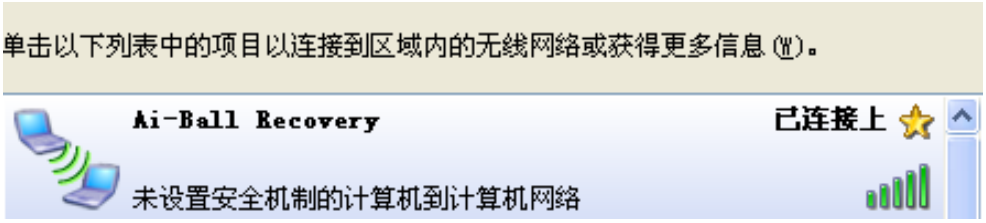
通过 Wi-Fi 连接更新软件

1） 在 www.thumbdriver.com 或者访问技术支持网页下载最新软件。

2） 开关拨到设置状态，等待 10 秒，刷新无线网络连接。

3） 同开始设置 2 到 4 步 选择 Firmware Upgrade 点击 Switch 按钮（如下图）

4） 设置完后，把开关拨到“OFF”，再把开关拨到“ ”，等待约 10 秒；在电脑无线网络连接里找到 SSID 为” Ai-Ball Recover”,点击连接（如下图所示）

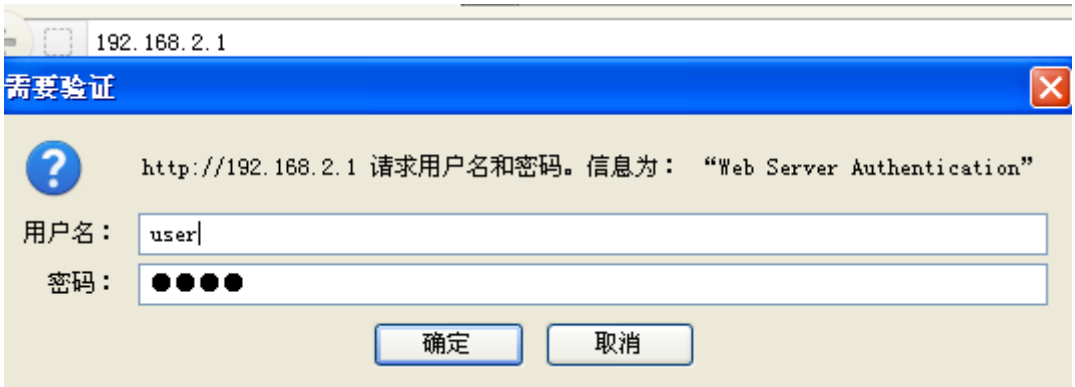


5） **连接成功后，在 IE 浏览器里输入** http://ai-ball.com 或 http://192.168.2.1 打开网页，

输入用户名：user,

密码：pass

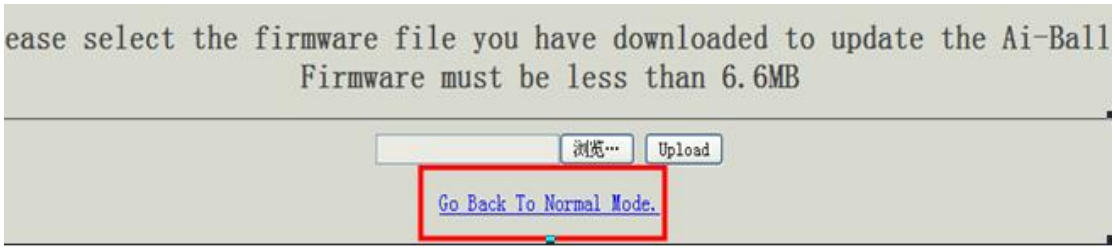
点击”Ok”



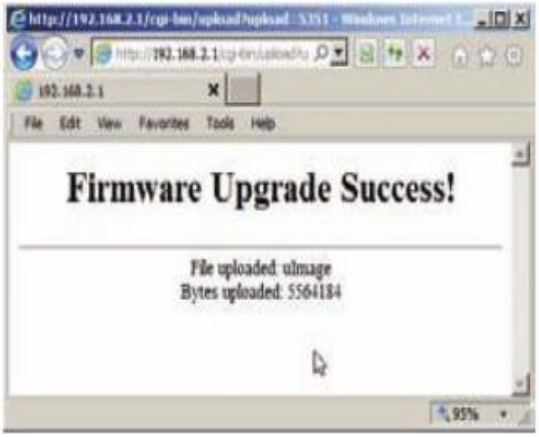
点击“确定”按钮

6） 点击“Browse”浏览按钮，找到从我们网站下载到的软件，例如下载的软件为 uImage，选中 uImage，点击“Open”打开。

7）点击”浏览”，找到你的固件程序，再点击“Upload”开始更新软件，你会看到软件更新进度条，更新完后，你会看到“Firmware Upgrade Success”表明软件已经更新成功。



如果你不想更新固件，可以点击“Go Back To Normal Mode”按钮恢复正常模式



注意：在更新过程中，不要关闭电源。

8）更新完后，再关闭电源，把开关拨到“ON”,Ai-Ball 就会运行新的软件。

智能交通

第四章 通用控制节点

4.1 大唐移动通用控制节点软硬件开发指南

4.2 大唐移动 SmartCar 软硬件开发指南

智能交通通用控制节点软硬件开发指南

1、 系统硬件介绍

软件部分，基于 uVision4 KEIL 4.1 开发，STM32 固件库版本 3.5，Keil4 的安装和使用说明请参考文档《Keil_uVision4 安装详细图解和破解方法.doc》。

智能小车详细的源代码，请打开“ETC 入口 v1.0”中的 KEIL 工程。以下仅对各模块的关键驱动代码作介绍。

1.1、 主控单元

主控单元是 STM32F103C8T6 芯片及其外围元件，关于 STM32F103 的详细资料，请参考 ST 官方文档。

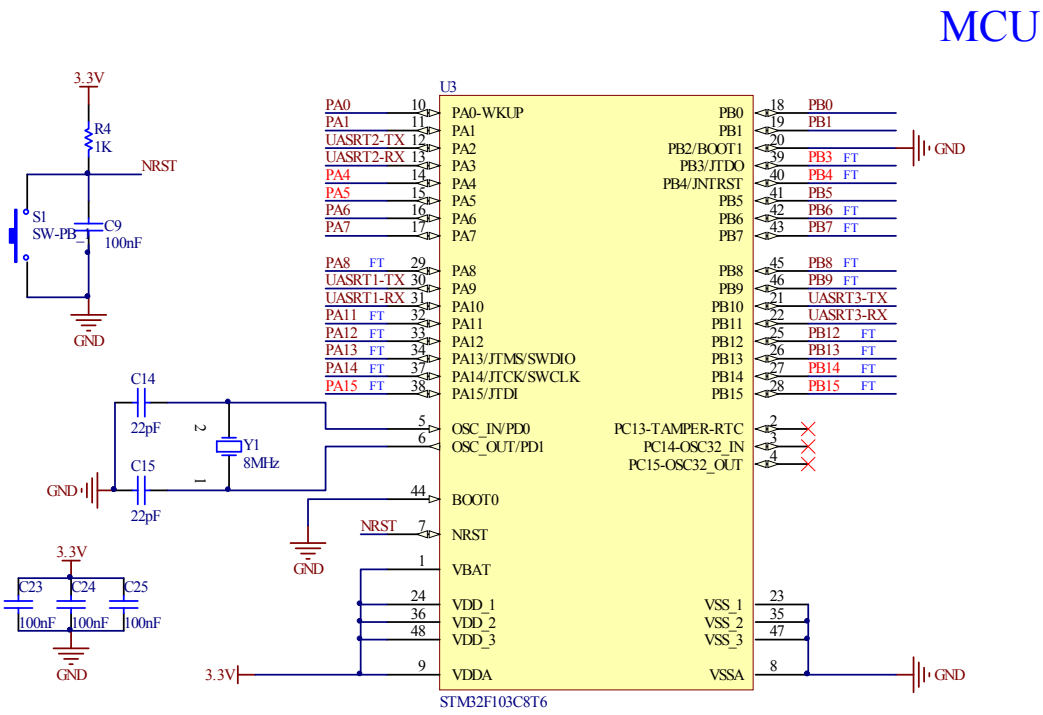


图 1.1 主控电路

在文件 SystemConfig.c 中，系统时钟初始化代码如下：

```
void RCC_Configuration(void)
{
  ErrorStatus HSEStartUpStatus;   //定义外部高速晶体启动状态枚举变量
  /* RCC system reset(for debug purpose) */
  RCC_DeInit();   //复位 RCC 外部设备寄存器到默认值

  /* Enable HSE */
  RCC_HSEConfig(RCC_HSE_ON);   //打开外部高速晶振

  /* Wait till HSE is ready */
```

```
    HSEStartUpStatus = RCC_WaitForHSEStartUp();    //等待外部高速时钟准备好

    if(HSEStartUpStatus == SUCCESS)    //外部高速时钟已经准别好
    {
      /* HCLK = SYSCLK */
      RCC_HCLKConfig(RCC_SYSCLK_Div1); //配置 AHB(HCLK)时钟等于==SYSCLK

      /* PCLK2 = HCLK */
      RCC_PCLK2Config(RCC_HCLK_Div1);     //配置 APB2(PCLK2)钟==AHB 时钟

      /* PCLK1 = HCLK/2 */
      RCC_PCLK1Config(RCC_HCLK_Div2);    //配置 APB1(PCLK1)钟==AHB1/2 时钟

//      /* Flash 2 wait state */
//      FLASH_SetLatency(FLASH_Latency_2);
//      /* Enable Prefetch Buffer */
//      FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);

      /* PLLCLK = 8MHz * 9 = 72 MHz */
      RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);     //配置 PLL 时钟  ==  外部高速晶体时钟
*9

      /* Enable PLL */
      RCC_PLLCmd(ENABLE);    //使能 PLL 时钟

      /* Wait till PLL is ready */
      while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET)    //等待 PLL 时钟就绪
      {
      }

      /* Select PLL as system clock source */
      RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);     //配置系统时钟  = PLL 时钟

      /* Wait till PLL is used as system clock source */
      while(RCC_GetSYSCLKSource() != 0x08)     //检查 PLL 时钟是否作为系统时钟
      {}
    }
      //RCC_AHBPeriphClockCmd(RCC_AHBPeriph_FSMC, ENABLE);
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE); //用 EXTI 的时候必须开此中断
      RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3, ENABLE);
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB , ENABLE);
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA , ENABLE);
```

```
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
}
```

1.2、 电源单元

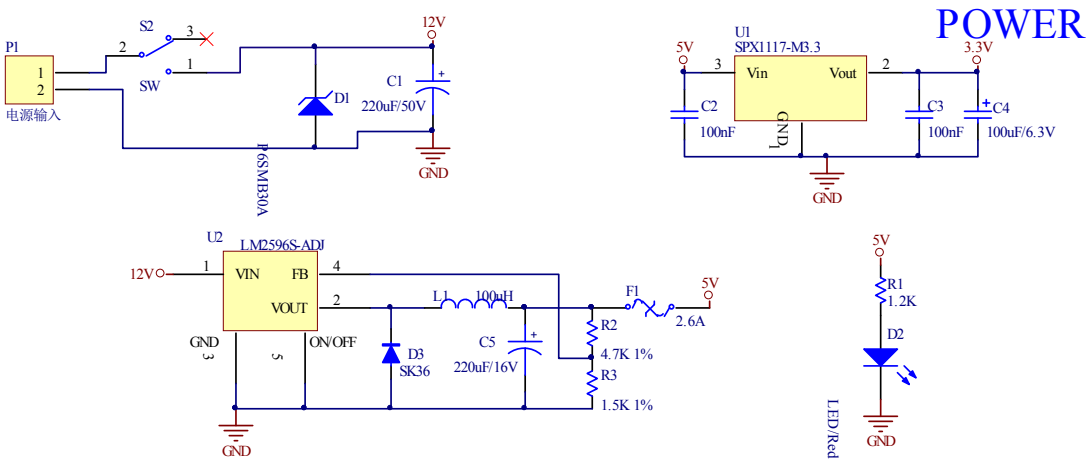


图 1.2 电源电路

电源部分采用 LM2596ADJ 开关电源芯片作为第一级电源转换，将输入电压变为 5V，供 Zigbee 模块、L298 电机驱动等，采用 LM1117-3.3 低压差电源芯片再将 5V 转成 3.3V，以供 STM32、磁导航、RFID 等。

电源输入端使用了自恢复保险、反向并联 P6SMB30A TVS 管，用于保护下级电路。

1.3、 Zigbee 模块

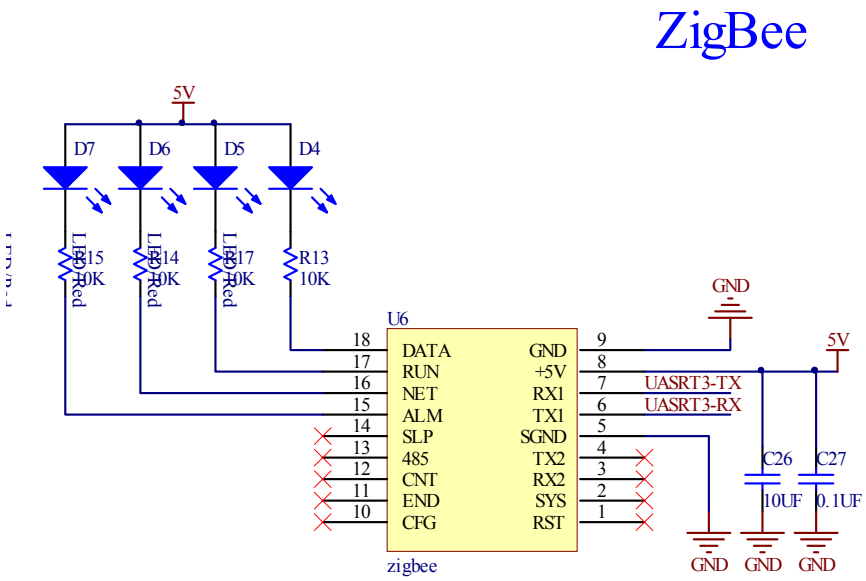


图 1.3 Zigbee 无线通信模块

Zigbee 无线通信模块采用串口透传方案，即 STM32 的 UART 端口直接交叉连接 Zigbee 模块的 UART 口，当 stm32 向 Zigbee 模块发送数据时，上位机的 Zigbee 协调器可以直接通过串口接收到数据。

由于 Zigbee 串口透传部分要实现上位机指令协议解析，所以采用了协议栈的形式来处理发送和接收数据。

在接收上用中断的方式，发送用 DMA 的方式。串口初始化代码如下：

```
/*************************************************************
*    函数名称：void COM1_Init(void)
*    功      能：串口初始化
*    入口参数：无
*    出口参数：无
*************************************************************/
void Com1Init(void)
{
    USART_InitTypeDef USART_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;

    USART_InitStructure.USART_BaudRate = 115200;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
    USART_InitStructure.USART_Parity = USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;

    USART_Init(USART1, &USART_InitStructure);
    USART_Cmd (USART1, ENABLE);

    /*使能 DMA 模式的发送，以及中断方式的接收*/
    USART_DMACmd(USART1,USART_DMAReq_Tx,ENABLE);
    USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);

      /*串口 GPIO 初始化*/
    //TXD 设置为复用推挽输出
    GPIO_InitStructure.GPIO_Pin =    GPIO_Pin_9 ;
    GPIO_InitStructure.GPIO_Mode =   GPIO_Mode_AF_PP;            // 复用推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    //RXD 设置为复用功能浮空输入
    GPIO_InitStructure.GPIO_Pin =    GPIO_Pin_10 ;
    GPIO_InitStructure.GPIO_Mode =   GPIO_Mode_IN_FLOATING; // 浮空功能输入
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    Com1_BufInit();
    //flash_initial();//初始化硬件信息
}
```

可以看到函数最后调用了 Com1_BufInit();函数，这是环形链表初始化函数，串口收发采用环形链表的形式可以提高设备在突发高速率通信情况下的可靠性。

```
/************************************************************
*    函数名称：void Com1_BufInit(void)
*    功      能：初始化发送与接收缓冲区,并构缓冲区环型链表
*    入口参数：无
*    出口参数：无
************************************************************/
void Com1_BufInit(void)
{
    unsigned char i;
    for ( i=0; i<TXD_BUF_NUM; i++ )
    {
        TXD_BUF[i].command = 0;
        TXD_BUF[i].length = 0;
        TXD_BUF[i].sum = 0;
                TXD_BUF[i].state = 1;        //初始化的时候表示该发送缓冲区已经发送完数据，可以使用
        if(i == (TXD_BUF_NUM - 1))
        {
            TXD_BUF[i].next = &     TXD_BUF[0] ;   //最后一个的尾指向第一个的头
        }
        else
        {
            TXD_BUF[i].next = &     TXD_BUF[i+1] ;
        }
    }

    for ( i=0; i<RXD_BUF_NUM; i++ )
    {
        RXD_BUF[i].command = 0;
        RXD_BUF[i].length = 0;
        RXD_BUF[i].sum = 0;
                RXD_BUF[i].state = 0;      //初始化的时候表示全部接受没有完成，可以接收
        if(i == (RXD_BUF_NUM - 1))
        {
            RXD_BUF[i].next = &     RXD_BUF[0] ;
        }
        else
        {
            RXD_BUF[i].next = &     RXD_BUF[i+1] ;
```

```
            }
        }
    }
```

在 com1_ZB.h 中定义了数据结构 DATA_BUF，在 com1_ZB.c 中对链表收尾进行了关联。代码如下：

```
    struct DATA_BUF                                    //定义数据缓冲区结构体
    {
      unsigned char uart0;
      unsigned char uart1;
      unsigned char target_id;                         //目标地址
      unsigned char my_id;                             //自身地址
      unsigned char data[DATA_BUF_NUM];                    //数据缓冲区
      unsigned char command;                           //命令字
      unsigned char length;                            //实际通讯的字节数
      unsigned char sum;                               //所有的和
      unsigned char state;                             //0 (TRUE)：表示一个新的数据帧接收完，还未
处理
      unsigned char error_flag;                        //错误标志位
      struct DATA_BUF *next;                            //链表连接指针，指向下一个表头
    };

Com1_ZB.c 文件中
    struct DATA_BUF RXD_BUF[RXD_BUF_NUM];                    //定义 RXD_BUF_NUM 个接收缓冲区
的个数
    struct DATA_BUF TXD_BUF[TXD_BUF_NUM];                    //定义 TXD_BUF_NUM 个发送缓冲区
的个数

    struct DATA_BUF *RXD_BUF_P_FRONT = &RXD_BUF[0];          //接收缓冲队列头
    struct DATA_BUF *RXD_BUF_P_REAR = &RXD_BUF[0];           //接收缓冲队列尾

    struct DATA_BUF *TXD_BUF_P_FRONT = &TXD_BUF[0];          //发送缓冲队列头
    struct DATA_BUF *TXD_BUF_P_REAR = &TXD_BUF[0];           //发送缓冲队列尾
```

在 stm32f10x_it.c 中有串口接收的中断服务函数和 DMA 发送结束函数。

```
    void DMA1_Channel4_IRQHandler(void)              //串口 DMA 方式发送完成中断
    {
        DMA_ClearFlag(DMA1_FLAG_TC4);
        TXD_BUF_P_FRONT = TXD_BUF_P_FRONT->next;
        TXD_BUF_P_FRONT->state = 1;
        DMA_Cmd(DMA1_Channel4,DISABLE);//发送完毕关闭 DMA，直到下次有数据发送时打开 DMA
    }
```

```
void USART1_IRQHandler(void)          //串口 1 接收中断服务程序
{
    if(USART_GetITStatus(USART1, USART_IT_RXNE)!= RESET)
    {
        Com1_Rxd();         //接收数据到缓冲队列
        Com1_ProtocolAnalysis();     //处理数据
        USART_ClearITPendingBit(USART1,USART_IT_RXNE);
    }
}
```

追踪 Com1_ProtocolAnalysis();函数，如果接收到一个成功帧，可以看到最终接收到的数据在

```
    switch(com_pk->command)
        {
            case 0x01:
                Com1_WriteMemory(com_pk);
                break;
            case 0x02:
                Com1_ReadMemory(com_pk);
                break;
            default:
                com_pk->error_flag = 0x01;//无效指令
                Com1_SendBack(com_pk);
                break;
        }
```

例如，进入 Com1_ReadMemory();函数，可以看到数据在这里进行了最终解析，

```
    /*****************************************************************************
*   函数名称：void Com1_ReadMemory(struct DATA_BUF *readone_pk)
*   功    能：响应上位机的读指令，将读取的数据送入发送缓冲区
*   入口参数：缓冲区结构体
*   出口参数：
*****************************************************************************/
void Com1_ReadMemory(struct DATA_BUF *readone_pk)
{
    u8 start_id;
    start_id = readone_pk->data[0];
    TXD_BUF_P_REAR->uart0 = readone_pk->uart0;
    TXD_BUF_P_REAR->uart1 = readone_pk->uart1;
    TXD_BUF_P_REAR->target_id = readone_pk->my_id;
    TXD_BUF_P_REAR->my_id = NODE_ID;
    TXD_BUF_P_REAR->command = readone_pk->command;          //设置返回命令
```

```
        TXD_BUF_P_REAR->error_flag = 0;
        TXD_BUF_P_REAR->state = 0;
        TXD_BUF_P_REAR->data[0] = readone_pk->data[0];
        if((start_id >= INFO_START) && (start_id <= INFO_END))
        {
            switch(readone_pk->data[0])
            {
                case 0x11:      //车辆信息
                    TXD_BUF_P_REAR->length = 3;
                    TXD_BUF_P_REAR->data[1] = CAR_ID;
                    TXD_BUF_P_REAR->data[2] = CAR_TYPE;
                    break;
                case 0x12:          //运行命令,该指令读无意义
                    TXD_BUF_P_REAR->error_flag = 0x02; //无效数据
                    TXD_BUF_P_REAR->length = 0;
                    break;
                case 0x13:              //运行状态
                    TXD_BUF_P_REAR->length = 2;
                    TXD_BUF_P_REAR->data[1] = CarMoveStatus;
                    break;
                case 0x14:              //位置编号
                    TXD_BUF_P_REAR->length = 4;
                    TXD_BUF_P_REAR->data[1] = AreaID_ASC2;
                    TXD_BUF_P_REAR->data[2] = StreetID_ASC2;
                    TXD_BUF_P_REAR->data[3] = RfidCardNum_HEX;
                    break;
                case 0x15:              //路径编码
                case 0x16:                  //车位编号
                default:
                    TXD_BUF_P_REAR->error_flag = 0x02;
                    break;
            }
        }
        else
            TXD_BUF_P_REAR->error_flag = 0x02;

            Com1_SetSum(TXD_BUF_P_REAR);                                    //设置校验和
            Com1_SendingData();
    }
```

最后将对应读取到的数据调用回执函数对上位机进行数据回复。至此完成了一个完整的读数据过程。

1.4、 接口电路

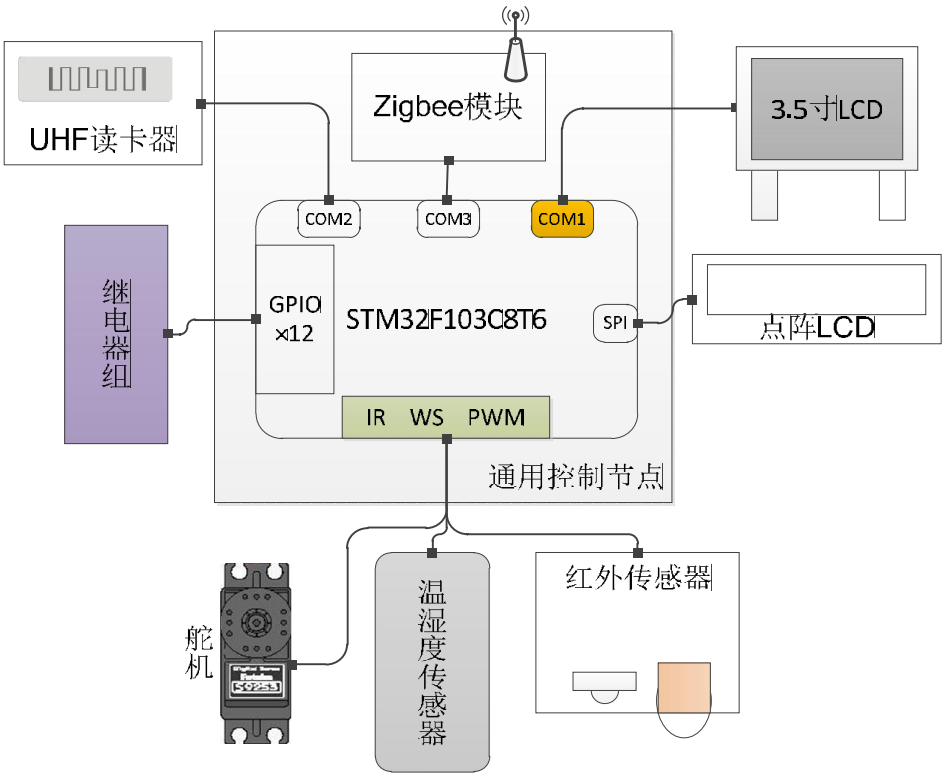


图 1.4 通用节点设计框图

根据上图我们得知，通用控制要实现不同的功能，需要选配不同的外设，这些外设有可能是 IO 量、RS232、PWM、IIC、SPI、OneWire 等，所以要求通用控制节点能提供这些接口。下图是接口电路。

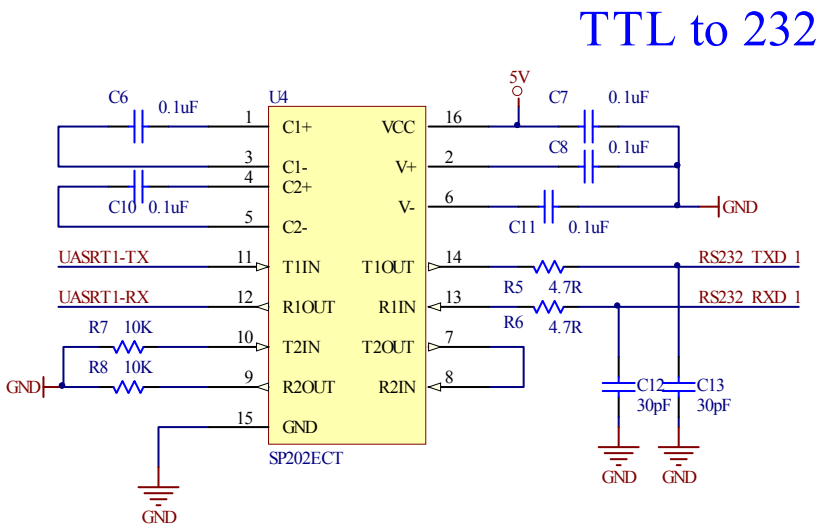


图 1.5 TTL 转 RS232 电路

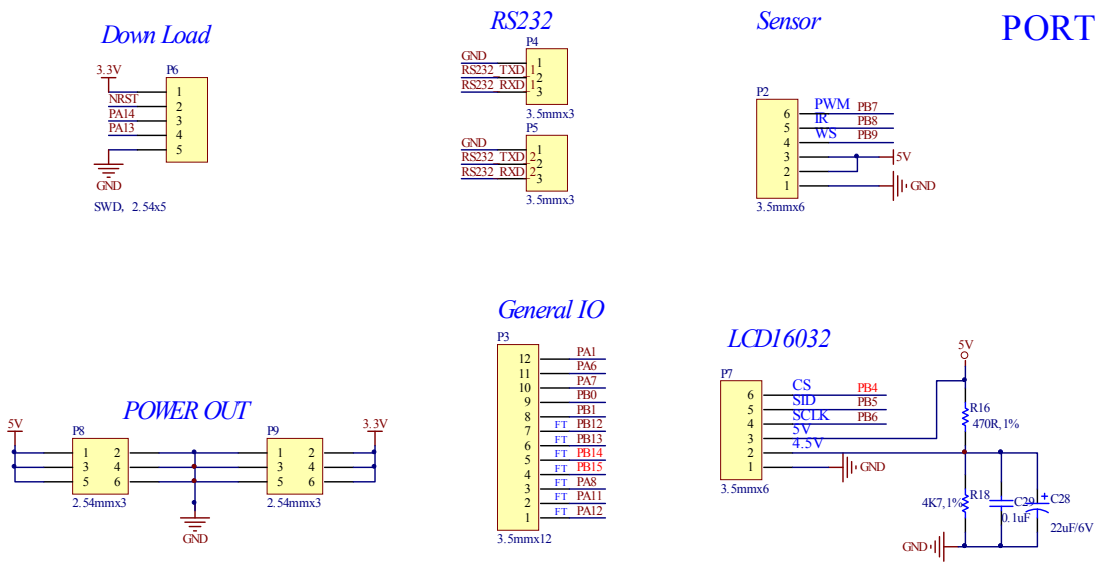


图 1.6 通用控制节点的接口

2、 外设驱动说明

2.1、 舵机驱动

舵机在 ETC 系统中模拟了抬杆作用，一般的舵机是以下的结构和引线形式，接线定义请参考《大唐移动 UI-IOT-ITS 智能交通实验指导书 V2.7.pdf》表 3.5。

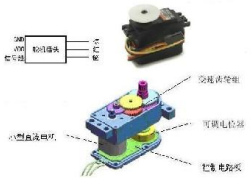


图 2.1 舵机原理图

舵机的控制一般需要一个 20ms 左右的时基脉冲，该脉冲的高电平部分一般为 0.5ms-2.5ms 范围内的角度控制脉冲部分，总间隔为 2ms。以 180 度角度伺服为例，那么对应的控制关系是这样的：

0.5ms----------0 度；

1.0ms----------45 度；

1.5ms----------90 度；

2.0ms----------135 度；

2.5ms----------180 度；

首先，我们配置一个 PWM 端口，使之可以输出周期为 20ms 的脉冲方波。

```
/*****************************************
 *功能：舵机产生 PWM 方波初始化
```

```
    *描述：产生周期为 20ms 的 PWM 波，脉宽可调，计数 20000
          次为 20ms，依次类推，
    *输入：无
    *输出：无
    ****************************************/
    void Servos_PWM_Config(void)
    {
        TIM_TimeBaseInitTypeDef    TIM_TimeBaseStructure;
      TIM_OCInitTypeDef            TIM_OCInitStructure;

      TIM_TimeBaseStructure.TIM_Period = 20000;                          //定时 20ms
      TIM_TimeBaseStructure.TIM_Prescaler = 72;                          //设置预分频：72 分频，即为
72MHz/72 = 1MHz
      TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1 ;           //设置时钟分频系数：不分频
      TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;   //向上计数模式
      TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);

      /* PWM1 Mode configuration: TIM2_CH2 */
      TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;                  //配置为 PWM 模式 1
      TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
      TIM_OCInitStructure.TIM_Pulse = 1850;                              //设置跳变值，当计数器计数到这
个值时，电平发生跳变
      TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;      //当定时器计数值小于 CCR1_Val 时为
高电平
      TIM_OC2Init(TIM4, &TIM_OCInitStructure);                           //使能通道 2
      TIM_OC2PreloadConfig(TIM4, TIM_OCPreload_Enable);
      TIM_ARRPreloadConfig(TIM4, ENABLE);                                // 使能 TIM4 重载寄存器
ARR
      /* TIM2 enable counter */
      TIM_Cmd(TIM4, ENABLE);
    }
```

实现抬杆落杆的两个个函数：

```
    void Servos_Open(void)
    {
        for(;Servos_Now_PWM >= 1100;Servos_Now_PWM = Servos_Now_PWM - 1)
        {
          TIM_SetCompare2(TIM4, Servos_Now_PWM);
          Delay_ms(1);
        }
    }
    void Servos_Close(void)
```

```
{
        for(;Servos_Now_PWM <= 1850;Servos_Now_PWM = Servos_Now_PWM + 1)
    {
     TIM_SetCompare2(TIM4, Servos_Now_PWM);
     Delay_ms(1);
    }

}
```

2.2、 LCD 驱动

智能控制节点与液晶屏通过 RS232 接口连接，接线定义请参考《大唐移动 UI-IOT-ITS 智能交通实验指导书 V2.7.pdf》表 3.5。

例如 ETC 系统上电，要在屏幕上显示初始化信息：

```
void LCD_ShowETCInfo(void)
{
    ClearAll();
    ShowStr(ONE,40,5,"      ETC 入口");
    ShowStr(TWO,15,60,"进入时间");
    ShowStr(TWO,15,100,"车辆牌照:");
    ShowStr(TWO,15,140,"卡内余额:        元");
    ShowStr(TWO,15,190,"操作状态:");
}
```

2.3、 红外传感器

在 ETC 系统中，红外传感器检测有无车辆触发超高频读卡器的读卡操作，接线定义请参考《大唐移动 UI-IOT-ITS 智能交通实验指导书 V2.7.pdf》表 3.5。

在使用中，我们直接配置成上拉输入，并在系统主循环中查询端口状态就可以了。

```
/************************************
 *功能：红外端口描述
 *描述：无
 *输出：无
 *输入：无
************************************/
void Infrared_GPIO_Config(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB , ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;//上拉输入
  GPIO_Init(GPIOB, &GPIO_InitStructure);
```

```
}
```

2.4、 UHF 读卡器驱动（900MHz 超高频读卡器）

UHF 读卡器与通用控制节点也是采用 RS232 的方式连接，接线定义请参考《大唐移动 UI-IOT-ITS 智能交通实验指导书 V2.7.pdf》表 3.5。

在 ETC 应用中，只对固定的区块进行读写操作：

```
/**************************************
 *功能：读取标签内存
 *描述：无
 *输入：无
 *输出：改变全局变量 TagMEM,将读到的数据存
                入该缓冲区中
**************************************/
unsigned char ReadTagMEM(void)
{
    unsigned char i;
    unsigned char MEMcmd[25] = {0};
    unsigned char TagMEM[8]   = {0};
    unsigned char CRC_MSB,CRC_LSB;
    u16                CRC_Val;
    unsigned char TimeCNT = 0;

    GetMultipleRead();
    VUM_Delay(10);

    MEMcmd[0] = 0x18;
    MEMcmd[1] = 0x00;
    MEMcmd[2] = 0x02;
    MEMcmd[3] = 0x06;
    for(i=0;i<12;i++)
        MEMcmd[i+4] = EPC[i];
    MEMcmd[16] = 0x03;//User 区
    MEMcmd[17] = 0x00;
    MEMcmd[18] = 0x04;
    MEMcmd[19] = 0x00;
    MEMcmd[20] = 0x00;
    MEMcmd[21] = 0x00;
    MEMcmd[22] = 0x00;
    CRC_Val = uiCrc16Cal(MEMcmd,23);
    CRC_MSB = (u8)(CRC_Val >> 8);
    CRC_LSB = (u8)(CRC_Val & 0xff) ;
```

```
        MEMcmd[23] = CRC_LSB;
        MEMcmd[24] = CRC_MSB;

        DMAflag = DMABusy;
        VUM_DMA_Cfg(0x0e);     //打开 DMA，接收 14 个字节的数据是产生中断
        for(i=0;i<25;i++)
              VUM_SendUartData(MEMcmd[i]);
        VUM_Delay(2);

        while(DMAflag == DMABusy && TimeCNT < 50)
        {
            VUM_Delay(2);
            TimeCNT ++;
        }
        if(RxBuffer[0] == 0x0d && RxBuffer[2] == 0x02)
        {
            for(i=0;i<8;i++)
            {
                TagMEM[i] = RxBuffer[i+4];
            }
            ETCData_R.CarID          = TagMEM[0];          //车辆 ID
        ETCData_R.CarType        = TagMEM[1];        //车辆类型
        ETCData_R.Balance        = TagMEM[2];        //卡内余额
        ETCData_R.InOutFlag      = TagMEM[3];      //进出标志
        ETCData_R.EntTimHou      = TagMEM[4];      //进入时间:Hou
        ETCData_R.EntTimMin      = TagMEM[5];      //            Min
        ETCData_R.EntTimSec      = TagMEM[6];      //            Sec
            ETCData_R.ETCWayLen      = TagMEM[7];
            return V_TRUE;
        }
        else
        {
            for(i=0;i<8;i++)
                TagMEM[i] = 0;
           return V_FALSE;
        }

}
/*************************************
 *功能：写标签内存
 *描述：向标签的 EPC 存储区写入指定长度的
                指定数据
```

```
 *输入：Data 要写入的数据结构体，DatLen 要
                    写入的数据的长度(保留未用)
 *输出：无
*************************************/
unsigned char WriteTagMEM(ETCDataType Data,u8 DatLen)
{
    unsigned char WriteCMD[33] = {0};
      unsigned char i;
      u16                CRC_Val;
      u8                 CRC_MSB,CRC_LSB;
      unsigned char TimeCNT = 0;

      GetMultipleRead();
    VUM_Delay(10);

      WriteCMD[0] = 0x20;          //Len = 32
      WriteCMD[1] = 0x00;          //Address
      WriteCMD[2] = 0x03;          //cmd
      WriteCMD[3] = 0x04;       //word to write num
      WriteCMD[4] = 0x06;       //EPC num(word)
      for(i=0;i<12;i++)
          WriteCMD[i+5] = EPC[i];
      WriteCMD[17]= 0x03;       //User Area
      WriteCMD[18]= 0x00;       //Word ptr

      WriteCMD[19]= Data.CarID;
      WriteCMD[20]= Data.CarType;
      WriteCMD[21]= Data.Balance;
      WriteCMD[22]= Data.InOutFlag;
      WriteCMD[23]= Data.EntTimHou;
      WriteCMD[24]= Data.EntTimMin;
      WriteCMD[25]= Data.EntTimSec;
      WriteCMD[26]= Data.ETCWayLen;          //保留
      WriteCMD[27]= 0x00;                //password
      WriteCMD[28]= 0x00;
      WriteCMD[29]= 0x00;
      WriteCMD[30]= 0x00;

      CRC_Val = uiCrc16Cal(WriteCMD,31);
      CRC_MSB = (u8)(CRC_Val >> 8);
      CRC_LSB = (u8)(CRC_Val & 0xff) ;
```

```
        WriteCMD[31] = CRC_LSB;
        WriteCMD[32] = CRC_MSB;

        DMAflag = DMABusy;
        VUM_DMA_Cfg(6);     //打开 DMA，接收 6 个字节的数据时产生中断
        for(i=0;i<33;i++)
              VUM_SendUartData(WriteCMD[i]);
        VUM_Delay(2);

        while(DMAflag == DMABusy && TimeCNT < 50)
        {
                VUM_Delay(2);
                TimeCNT ++;     //超时判断，100 毫秒
        }
        if(RxBuffer[0] == 0x05 && RxBuffer[2] == 0x03)
        {
      return V_TRUE;
        }
        else
        {
          return V_FALSE;
        }

}
```

图 2.2 系统执行流程

所有通用控制节点的程序具有共通性，我们仅以 ETC 入口节点为例进行说明，举一反三即可。

打开 main.c 找到 main();主函数，首先是上电时的初始化。

```
        Servos_Now_PWM = 1850;
        Servos_Init();              //舵机初始化函数          //LCD 初始化函数
        Servos_Close();             //关闭舵机
        IR_Delay(500);              //系统上电延时，躲过舵机上电大电流冲击

        LCD_Init();
        RCC_Configuration();     //配置系统时钟
        NVIC_Configuration();    //配置系统中断管理，这里指管理了 ZigBee 所用到的中断
        Com1Init();                //ZigBee 初始化函数
        IR_Init();                 //红外初始化函数
        SetColor(WHITE,BLUE);      //设置液晶背景色为蓝色，前景色为红色
        LCD_ShowETCInfo();         //在液晶上显示提示信息
        IR_Delay(10);
```

```
    VUMInit();                  //UHF 高频 RFID 初始化函数
    DogConfig();
```

接着就进入 while(1)循环，首先查询是否有车辆触发红外传感器，之后进行一系列的操作。

```
while(1)
    {
        if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_8) == 0)//车来到
        {
            IR_Delay(50);
            if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_8) == 0)
            {
                //读取卡片信息，判断车辆 ID
              //    ShowStr(TWO,15+24*5,70,"正在检查车位...");
                ReadTimes = 4;//超时判断,如果读取的次数超过 10 次，就放弃读取
                do{
                    ReadTagStatus = ReadTagMEM();
                  ReadTimes --;
                  VUM_Delay(10);
                }while((V_FALSE == ReadTagStatus) && ((ReadTimes) > 1));//读取卡片内存信息
                if(V_FALSE == ReadTagStatus)
                    SysReset();
                if(ReadTagStatus == V_TRUE)
                {
                    ETCData_R.EntTimHou = 0;                    //上报入口时间，没
用，所以就填写 0
                    ETCData_R.EntTimMin = 0;
                    ETCData_R.EntTimSec = 0;

                    ETCtoReport[0] = 1;                         //公路号
                    ETCtoReport[1] = ETCData_R.CarID;           //车辆 ID
                    ETCtoReport[2] = 1;                          //车辆类型
                    ETCtoReport[3] = ETCData_R.Balance;         //卡内余额
                    ETCtoReport[4] = 0;                          //进出标志
                    ETCtoReport[5] = ETCData_R.EntTimHou;        //进入时间 hou
                    ETCtoReport[6] = ETCData_R.EntTimMin;        //          min
                    ETCtoReport[7] = ETCData_R.EntTimSec;        //          sec

                    Com1_Reporting(0x11,(u8*)&ETCtoReport,8);    //主动上报
                    ShowStr(TWO,15+24*5,190,"读卡成功          ");
                    IR_Delay(20);//缓冲一下，等待上位机发送指
                    ShowStr(TWO,15+24*5,100,"京 A 8888");
                    ShowData_int(TWO,15+24*9,100,ETCData_R.CarID);
```

```
                    ShowData_int(TWO,15+24*5,140,ETCData_R.Balance);

                    if(ETCData_R.Balance > 0)
                    {

                        Com1_Reporting(0x15,(u8*)&LG_State[1],1);                    //主动上报,可以通
行
                        Servos_Open();
                        while(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_8) == 0)IR_Delay(400);//等待
小车离开
                        IR_Delay(1000);//等待车离开后两秒后关闭栏杆
                        Servos_Close();
                    }
                    else
                    {
                        ShowStr(TWO,15+24*5,190,"余额不足          ");
                        while(ETCData_W.Balance == 0)IR_Delay(5);//等待充值,在中断函数里面执行
充值操作
                        ShowStr(TWO,15+24*5,190,"充值成功          ");

                        Com1_Reporting(0x15,(u8*)&LG_State[1],1);                    //主动上报,可以通
行
                        Servos_Open();
                        while(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_8) == 0)IR_Delay(400);//等待
小车离开
                        IR_Delay(1000);//等待车离开后两秒后关闭栏杆
                        Servos_Close();
                        LCD_ShowETCInfo();
                    }

                //  ShowStr(TWO,15,60,"进入时间     :   : ");
                    while(TimeCMDFlag == UnRxTimeCMD)ShowStr(TWO,15+24*5,190,"等待下发时间
");;
                    ShowData_int(TWO,15+24*5,60,ETCData_W.EntTimHou); //小时
                    ShowStr(TWO,15+24*6,60,":");
                    ShowData_int(TWO,15+24*7,60,ETCData_W.EntTimMin);//分钟
                    ShowStr(TWO,15+24*8,60,":");
                    ShowData_int(TWO,15+24*9,60,ETCData_W.EntTimSec);//秒钟
                }//if(V_TRUE == ReadTagStatus)
                else
                    ShowStr(TWO,15+24*5,190,"读卡失败          ");
            }//if(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_8) == 0)
```

```
        }//if(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_8) == 0)
        else
        {
            ShowStr(TWO,15+24*5,190,"没有车辆");
            Servos_Close();
        }
        FeedDog();
    }
```

智能交通 SmartCar 软硬件开发指南

1、 系统硬件及软件综述

小车采用 STM32F103C8T6 作为主控芯片，集成了 RFID、Zigbee、电机驱动、磁导航、红外传感器等模块。其通过磁导航模块跟随铺设于路基下方的磁条循迹，同时用 RFID 技术读取路基下方的地标 RFID 标签辅助路径规划，Zigbee 模块与上位机通信，从而模拟实现了无人驾驶的未来交通工具特征。下图为智能小车的系统框图。原理图请参考《智能小车原理图.pdf》。

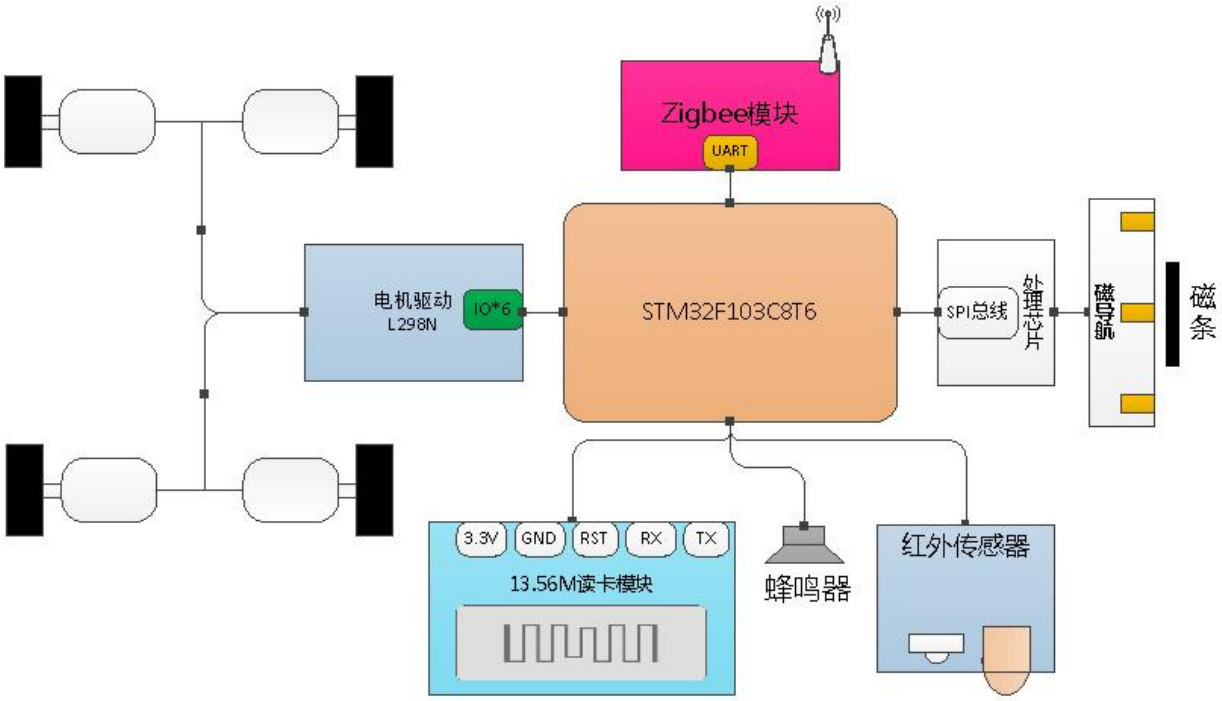


图 1.1 小车系统框图

软件部分，基于 uVision4 KEIL 4.1 开发，STM32 固件库版本 3.5，Keil4 的安装和使用说明请参考文档《Keil_uVision4 安装详细图解和破解方法.doc》。

智能小车详细的源代码，请打开“SmartCar2014-5-8 实例代码”中的 KEIL 工程。以下仅对各模块的关键驱动代码作介绍。

1.1、 主控单元

主控单元是 STM32F103C8T6 芯片及其外围元件，关于 STM32F103 的详细资料，请参考 ST 官方文档。

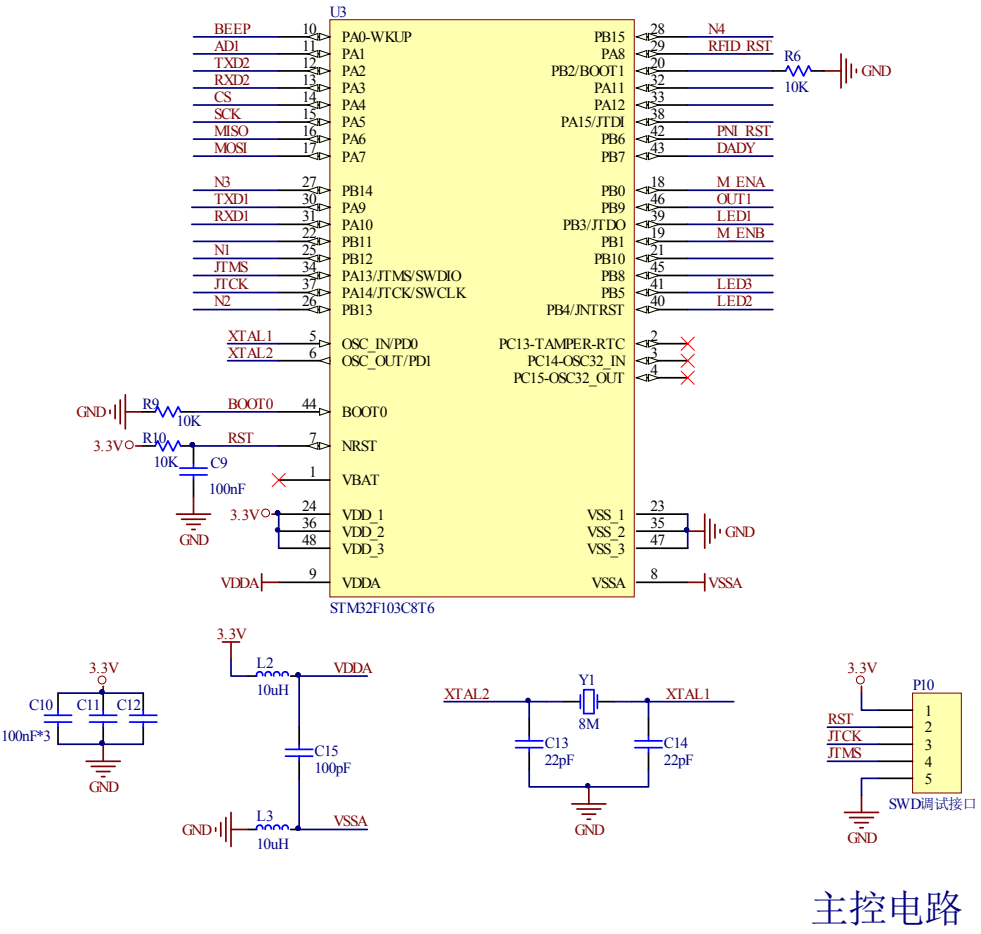


图 1.3 主控电路

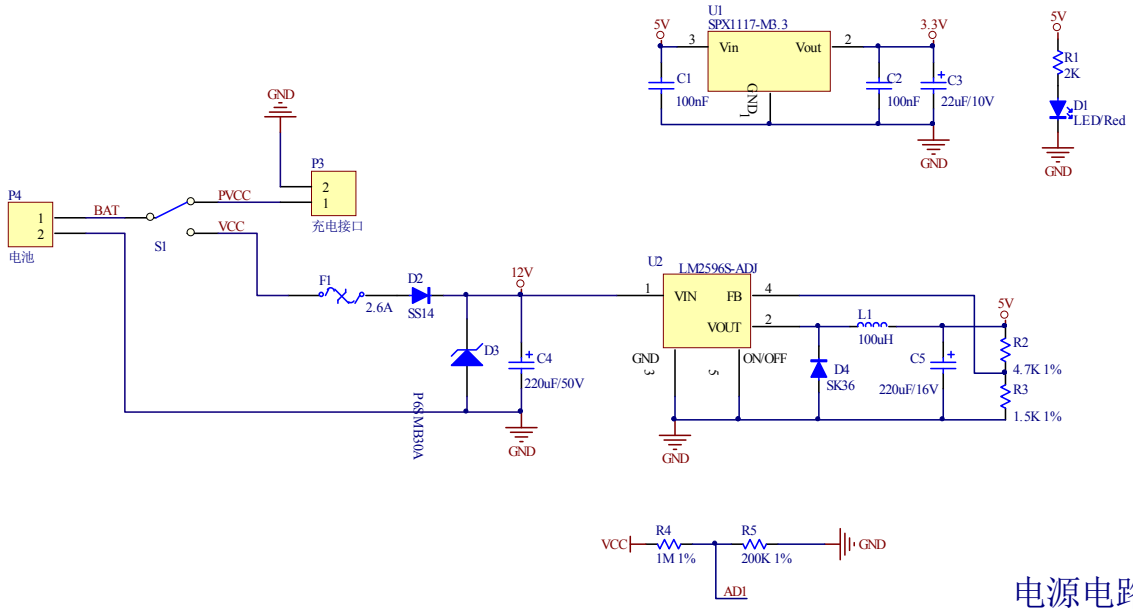
系统时钟初始化代码如下：

```
void RCC_Configuration(void)
{
  ErrorStatus HSEStartUpStatus;   //定义外部高速晶体启动状态枚举变量
  /* RCC system reset(for debug purpose) */
  RCC_DeInit();   //复位 RCC 外部设备寄存器到默认值

  /* Enable HSE */
  RCC_HSEConfig(RCC_HSE_ON);   //打开外部高速晶振

  /* Wait till HSE is ready */
  HSEStartUpStatus = RCC_WaitForHSEStartUp();   //等待外部高速时钟准备好

  if(HSEStartUpStatus == SUCCESS)   //外部高速时钟已经准别好
  {
    /* HCLK = SYSCLK */
    RCC_HCLKConfig(RCC_SYSCLK_Div1); //配置 AHB(HCLK)时钟等于==SYSCLK
```

```
        /* PCLK2 = HCLK */
        RCC_PCLK2Config(RCC_HCLK_Div1);     //配置 APB2(PCLK2)钟==AHB 时钟

        /* PCLK1 = HCLK/2 */
        RCC_PCLK1Config(RCC_HCLK_Div2);    //配置 APB1(PCLK1)钟==AHB1/2 时钟

    //      /* Flash 2 wait state */
    //      FLASH_SetLatency(FLASH_Latency_2);
    //      /* Enable Prefetch Buffer */
    //      FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);

        /* PLLCLK = 8MHz * 9 = 72 MHz */
        RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);     //配置 PLL 时钟  ==  外部高速晶体时钟
*9

        /* Enable PLL */
        RCC_PLLCmd(ENABLE);    //使能 PLL 时钟

        /* Wait till PLL is ready */
        while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET)    //等待 PLL 时钟就绪
        {
        }

        /* Select PLL as system clock source */
        RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);     //配置系统时钟  = PLL 时钟

        /* Wait till PLL is used as system clock source */
        while(RCC_GetSYSCLKSource() != 0x08)     //检查 PLL 时钟是否作为系统时钟
        {}
      }
        //RCC_AHBPeriphClockCmd(RCC_AHBPeriph_FSMC, ENABLE);
        //以下所有时钟均为程序中用到的硬件，放在一起开时钟便于维护
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE); //用 EXTI 的时候必须开此中断
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI1, ENABLE);                    //使能 SPI 时钟
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1, ENABLE);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2,ENABLE);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
```

```
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,ENABLE);

    GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);//释放 PA15，PB3，PB4
}
```

1.2、 电源单元



电源电路

图 1.2 电源电路

电源部分采用 LM2596ADJ 开关电源芯片作为第一级电源转换，将输入电压变为 5V，供 Zigbee 模块、L298 电机驱动等，采用 LM1117-3.3 低压差电源芯片再将 5V 转成 3.3V，以供 STM32、磁导航、RFID 等。

电源输入端使用了自恢复保险、防反接的二极管 SS14，以及反向并联 P6SMB30A TVS 管，用于保护下级电路。在 2.0 版本中加入了电压采样功能，通过 STM32 的 AD 端口实时读取电池电压，并在电压低的情况下鸣响蜂鸣器报警。

AD 端口初始化，采用 DMA 的方式获取采样值

```
void BatteryVoltageAdConfig(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    ADC_InitTypeDef ADC_InitStructure;          //ADC 初始化结构体声明
    DMA_InitTypeDef DMA_InitStructure;          //DMA 初始化结构体声明

    /* DMA1 channel1 configuration ----------------------------------------------*/
    DMA_DeInit(DMA1_Channel1);
```

```
    DMA_InitStructure.DMA_PeripheralBaseAddr = ADC1_DR_Address;      //DMA 对应的外设基地址
    DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&ADValue;     //内存存储基地址
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;  //DMA 的转换模式为 SRC 模式，由外设搬移到内存
    DMA_InitStructure.DMA_BufferSize = 1;               //DMA 缓存大小，1 个,单位为 DMA_MemoryDataSize
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;     //接收一次数据后，外设地址禁止后移
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;  //接收一次数据后，禁止目标内存地址自增
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;  //定义外设数据宽度为 16 位
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;  //DMA 搬数据尺寸，HalfWord 就是为 16 位
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;     //转换模式，循环缓存模式。
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;     //DMA 优先级高
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;              //M2M 模式禁用
    DMA_Init(DMA1_Channel1, &DMA_InitStructure);
    /* Enable DMA1 channel1 */
    DMA_Cmd(DMA1_Channel1, ENABLE);  //这句放在 main 中执行

    /* ADC1 configuration ------------------------------------------------------*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;   //独立的转换模式
    ADC_InitStructure.ADC_ScanConvMode = ENABLE;          //开启扫描模式
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;    //开启连续转换模式
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None; //ADC 外部开关，关闭状态
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;    //对齐方式,ADC 为 12 位中，右对齐方式
    ADC_InitStructure.ADC_NbrOfChannel = 1;   //开启顺序进行规则转换的 ADC 通道数
    ADC_Init(ADC1, &ADC_InitStructure);

    /* ADC1 regular channel10 configuration ADC 通道组， 第 n 个通道 采样顺序 1，转换时间 */
    ADC_RegularChannelConfig(ADC1, ADC_Channel_1, 1, ADC_SampleTime_239Cycles5);
    /* Enable ADC1 DMA */
    ADC_DMACmd(ADC1, ENABLE);      //ADC 命令，使能
    /* Enable ADC1 */
    ADC_Cmd(ADC1, ENABLE);   //开启 ADC1

    /* Enable ADC1 reset calibaration register */
```

```
        ADC_ResetCalibration(ADC1);          //重新校准
        /* Check the end of ADC1 reset calibration register */
        while(ADC_GetResetCalibrationStatus(ADC1));   //等待重新校准完成
        /* Start ADC1 calibaration */
        ADC_StartCalibration(ADC1);        //开始校准
        /* Check the end of ADC1 calibration */
        while(ADC_GetCalibrationStatus(ADC1));          //等待校准完成
        /* Start ADC1 Software Conversion */
        ADC_SoftwareStartConvCmd(ADC1, ENABLE);//连续转换开始，ADC 通过 DMA 方式不断的更新 RAM
区。      //放在 main 中执行
    }
```

在 main.c 中的 int main(void)函数中，实现了电压低就报警的作用。ADC1 的精度为 12 位，所以最大值为 4095，也就是能将 3.3V 分成 4095 份，在电池电压采样电路中用到了分压电路，实际测量的电压为电池的 1/6，假设电池最低保护电压为 9.6V，9.6/6=1.6,1.6/3.3=0.49,4095*0.48=1966；所以将采到的 ADValue 值跟 1966 做比较，如果低于此值说明电池电压低，需要关闭系统，并鸣响蜂鸣器报警。

```
        if(ADValue < 1966)//设定 9.6V 为放电截止电压，低于此值开始报警,实测 9.5 左右
            {

                Stop();
                TIM_Cmd(TIM1, DISABLE);//强制停止运行
                BeepOn();
                Delayms(500);
                BeepOff();
                Delayms(500);
                BeepOn();
                Delayms(1000);
                BeepOff();
    //          ADFlag = 0;
            }
```

1.3、 RFID 单元

RFID 读卡器采用 RC522 模块方案，RC522 是应用于 13.56MHz 非接触式通信中高集成度读写卡系列芯片中的一员。是 NXP 公司针对“三表”应用推出的一款低 电压、低成本、体积小的非接触式读写卡芯片，是智能仪表和便携 式手持设备研发的较好选择。

STM32 通过串口指令操作 RC522 读卡器，完成寻卡及读写卡操作，当检测到关键的 RFID 标签或者停车成功时，会通过端口输出一个高电平打开 BC817 三极管，鸣响一次蜂鸣器。

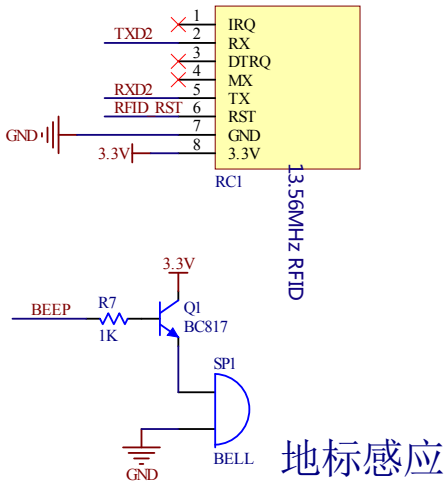


图 1.4 RFID 单元

关于 RC522 的驱动程序，可以参见“RC522.c”文件，下面仅对寻卡、读卡、写卡三个函数做介绍。

```
/*********************************************************************
*函数名：u8 SreachCard(u8 *pCardNum)
*功能：在天线所在区域内寻找 RFID 标签
*参数：存放 4 字节卡号的首地址
*返回值:执行结果：MI_OK：寻卡成功    MI_ERR：寻卡失败
*********************************************************************/
s8 SreachCard(u8 *pCardNum)
{
        s8 stat;
        u8 Tagtype[2],i; //卡类型暂存
        u8 selectedsnr[4]={0,0,0,0};
        stat= PcdRequest(REQ_ALL,Tagtype);
        if( stat == MI_OK)   //寻卡成功
          {
              stat = PcdAnticoll(selectedsnr);   //防碰撞
              if( stat == MI_OK)                    //防碰撞执行成功
              {
                  stat=PcdSelect(selectedsnr);
                  for(i=0;i<4;i++)    //输出卡号
                  {
                      *pCardNum++ = selectedsnr[i];
                  }
              }
              else
              {
                  stat = MI_ERR;
              }
```

```
            }
     return stat;
    }

    /*****************************************************************
    *函数名：u8 ReadCard(u8 *CardNum,u8 sector,u8 piece,u8 *pDataBuff)
    *功能：读取天线区域内的标签的数据
    *参数：CardNum：获取 4 字节标签序列号的首地址；
    *       sector：读取数据的扇区编号(S50 卡取值 0-15，共 16 扇区)
            piece ：读取数据的块编号（S50 卡取值 0-3 共 4 个块）
            pDataBuff：读取到的 16 个字节数据存放的首地址
    *返回值:执行结果：MI_OK：读卡成功    MI_ERR：读卡失败
    *****************************************************************/
    s8 ReadCard(u8 *CardNum,u8 sector,u8 piece,u8 *pDataBuff)
    {
        s8 stat;
        u8 i,buf[16];
        unsigned char   R_DefaultKey[6] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF} ;
          stat = PcdAuthState(KEYA, (sector*4+piece), R_DefaultKey, CardNum);// 校验 0 扇区密码
          if(stat == MI_OK)//密码验证成功
            {
              /*******************读卡测试*******************************************/
              stat = PcdRead((sector*4+piece), buf);          //读取 sector 扇区 piece 块数据
              if(stat == MI_OK)
                 {
                    for(i=0;i<16;i++)                  //读卡成功并且通过串口输出
                      {
                        // USART_SendByte(USART2,buf[i]);   //测试使用，将读到的数据通过串口打
印出来
                        *pDataBuff++ =     buf[i] ;
                      }

                 }
               else
                 {
                   stat == MI_ERR;
                 }
            }//验证密码
          else
            {
                stat == MI_ERR;
            }
```

```
        return    stat;
    }


    /*****************************************************************
    *函数名：u8 WriteCard(u8 *CardNum,u8 sector,u8 piece,u8 *pWriteData)
    *功能：写数据到天线区域内的标签
    *参数：CardNum：获取 4 字节标签序列号的首地址；
    *       sector：读取数据的扇区编号(S50 卡取值 0-15，共 16 扇区)
            piece ：读取数据的块编号（S50 卡取值 0-3 共 4 个块）
            pDataBuff：需要写入标签的 16 个字节数据的首地址
    *返回值:执行结果：MI_OK：写卡成功    MI_ERR：写卡失败

    注：分为 16 个扇区，每个扇区为 4 块，每块 16 个字节,以块为存取单位
    每个扇区由 4 块（块 0、块 1、块 2、块 3）组成，（我们也将 16 个扇区的 64 个块按绝对地址编号为
0~63，
    第 0 扇区的块 0（即绝对地址 0 块），它用于存放厂商代码，已经固化，不可更改。
    *****************************************************************/
    s8 WriteCard(u8 *CardNum,u8 sector,u8 piece,u8 *pWriteData)
    {
        s8 stat;                //执行状态
        u8 i,buf[16];
        unsigned char   W_DefaultKey[6] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF} ;

        stat = PcdAuthState(KEYA, (sector*4+piece), W_DefaultKey,CardNum);//  校验写扇区密码
         if(stat == MI_OK)     //密码验证成功
           {
            stat = PcdWrite((sector*4+piece), pWriteData);      //写 sector 扇区第 piece 块数据
                if(stat == MI_OK)
                 {
                 //--------------------仅供测试使用----------------------------------------------
    //                stat = PcdRead((sector*4+piece), buf);        //读扇区 1 扇区第 0 块数据
    //                if(stat == MI_OK)
    //                  {
    //                      for(i=0;i<16;i++)                       //读卡成功并且通过串
口输出   ,
    //                          {
    //                              USART_SendByte(USART2,buf[i]);
    //                          }
    //                  }
                 //-------------------------------------------------------------------------
```

```
                        }
                        else
                        { stat = MI_ERR; }

            }//验证密码
            else
            {
              stat = MI_ERR;
            }
    return stat;
 }
```

例如在 main()函数中通过调用 PositionCheck()函数来寻地标，在 PositionCheck()中有以下关键的代码：

```
RC522Status = SreachCard(SelectedSnr);        //寻卡
```

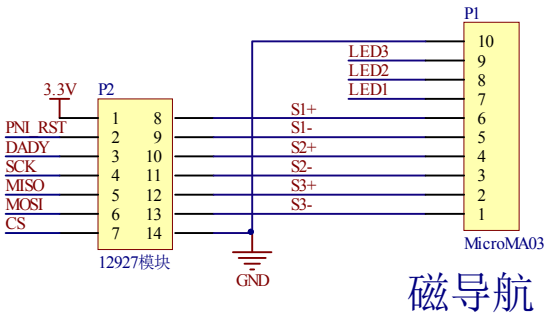
当确定寻到卡后，将寻到的卡号放在变量 SelectedSnr 中，然后作为参数读卡

```
 temp1 = ReadCard(SelectedSnr,RfidSector,RfidPiece,BlockBuf);
```

之后将读取到的特定地标值送入相关函数进行解析。

```
/**********根据关键地标确定修订值**********/
            if((TrackStatus == GARAGE_OUT)&&(Cartypefo == 0x01)) //私家车
            {
                RouteRevise = RfidAssistMobile();
            }
            else if(Cartypefo == 0)    //公交车
            {
                RouteRevise = RfidAssistBus(); //附加判定是否该停车
            }
            else if(TrackStatus == GARAGE_IN)//未入库成功执行停车指令
            {
                RouteRevise = RfidAssistPark();
            }
/****************修正完毕*****************/
```

1.4、 磁导航单元



磁导航

图 1.5 磁导航模块及接口电路

STM32 通过 SPI 总线读取对应线圈的磁场强度，并与程序中设定的阈值相比较，从而点亮对应顺序的 LED。

在函数 void MagnetoMeasure(void)中实现了此功能，通过将读取回来的磁场值跟 Threshold 进行比较，高于此值就点亮对应的 LED。

```
u16 Threshold = 1200;

void MagnetoMeasure(void)//完成一次磁导航测量
{
    //static u8 temp = 0;
    u8 i = 0;
    for(i = 0;i < 3; i ++)
    {
        Magneto_Data_Updata(i+1);
        if(Magneto_xyz_Data[i] > Threshold)
            LEDn_Set(i+1,LED_ON);
        else
            LEDn_Set(i+1,LED_OFF);
    }
}
```

1.5、 红外传感器

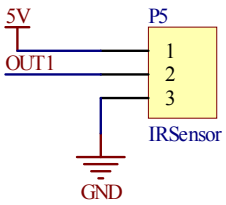


图 1.6 红外接口

红外传感器检测到前方障碍物，给 STM32 端口输出低电平，触发外部中断，STM32 控制电机停止，直到障碍物消失才再次启动小车。

在 SystemConfig.c 中，有中断及优先级相关配置的代码，红外端口配置如下

```
/*****************************************************************
* 函数名称：void IRInit(void)
* 功能描述：红外测距输入端口初始化
* 入口参数：无
* 出口参数：无
*****************************************************************/
void IRInit(void)
{
```

```
    GPIO_InitTypeDef GPIO_InitStructure;

    GPIO_InitStructure.GPIO_Pin = IR_Pin;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;     //通用浮空输入
    GPIO_Init(IR_PORT, &GPIO_InitStructure);
    IR_EXIT_Init();
}
/*************************************************************
* 函数名称：void IR_EXIT_Init(void)
* 功能描述：红外测距输入端口初始化
* 入口参数：无
* 出口参数：无
*************************************************************/
void IR_EXIT_Init(void)
{
    EXTI_InitTypeDef EXTI_InitStructure;
    GPIO_EXTILineConfig(IR_EXTI_PORT_SOURCE,IR_EXTI_PIN_SOURCE);
    /* Configure Button EXTI line */
    EXTI_InitStructure.EXTI_Line = IR_EXTI_LINE;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;//下降沿触发中断
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);
    EXTI_ClearFlag(EXTI_Line9);
}
```

在 stm32f10x_it.c 中有中断服务程序

```
void EXTI9_5_IRQHandler(void)
{
    if(EXTI_GetITStatus(EXTI_Line9) != RESET)    //确定触发中断
    {
        Stop();
        EXTI_ClearITPendingBit(EXTI_Line9);
    }
}
```

可以看到，这里仅仅是将小车停下，如果障碍物消失怎样重新启动呢？请看 main()中如下代码

```
/**********************查询障碍标志*************************/
        if((GPIO_ReadInputDataBit(IR_PORT,IR_Pin) != Bit_RESET) && (CarMoveStatus == NORMAL_STOP))
        {
            Delayms(4);//消抖
```

```
            if(GPIO_ReadInputDataBit(IR_PORT,IR_Pin) != Bit_RESET)
            {
                CarMoveStatus = NORMAL_FORWARD;
                //Advance(Percentage);
            }
        }
/*************************结束*****************************/
```

可以看到，这里采用了延时消抖的方法，

1.6、 Zigbee 模块

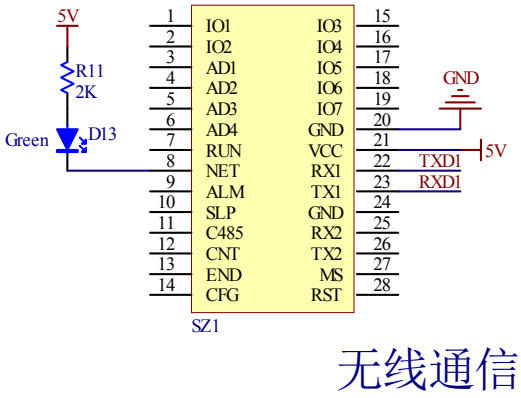


图 1.7 Zigbee 无线通信模块

Zigbee 无线通信模块采用串口透传方案，即 STM32 的 UART 端口直接交叉连接 Zigbee 模块的 UART 口，当 stm32 向 Zigbee 模块发送数据时，上位机的 Zigbee 协调器可以直接通过串口接收到数据。

由于 Zigbee 串口透传部分要实现上位机指令协议解析，所以采用了协议栈的形式来处理发送和接收数据。

在接收上用中断的方式，发送用 DMA 的方式。串口初始化代码如下：

```
/*************************************************************
*    函数名称：void COM1_Init(void)
*    功      能：串口初始化
*    入口参数：无
*    出口参数：无
*************************************************************/
void Com1Init(void)
{
    USART_InitTypeDef USART_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;

    USART_InitStructure.USART_BaudRate = 115200;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
```

```
    USART_InitStructure.USART_Parity = USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;

    USART_Init(USART1, &USART_InitStructure);
    USART_Cmd (USART1, ENABLE);

    /*使能 DMA 模式的发送，以及中断方式的接收*/
    USART_DMACmd(USART1,USART_DMAReq_Tx,ENABLE);
    USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);

      /*串口 GPIO 初始化*/
    //TXD 设置为复用推挽输出
    GPIO_InitStructure.GPIO_Pin =    GPIO_Pin_9 ;
    GPIO_InitStructure.GPIO_Mode =   GPIO_Mode_AF_PP;            // 复用推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    //RXD 设置为复用功能浮空输入
    GPIO_InitStructure.GPIO_Pin =    GPIO_Pin_10 ;
    GPIO_InitStructure.GPIO_Mode =   GPIO_Mode_IN_FLOATING; // 浮空功能输入
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    Com1_BufInit();
    //flash_initial();//初始化硬件信息
}
```

可以看到函数最后调用了 Com1_BufInit();函数，这是环形链表初始化函数，串口收发采用环形链表的形式可以提高设备在突发高速率通信情况下的可靠性。

```
/*************************************************************
*    函数名称：void Com1_BufInit(void)
*    功      能：初始化发送与接收缓冲区,并构缓冲区环型链表
*    入口参数：无
*    出口参数：无
*************************************************************/
void Com1_BufInit(void)
{
    unsigned char i;
    for ( i=0; i<TXD_BUF_NUM; i++ )
    {
        TXD_BUF[i].command = 0;
        TXD_BUF[i].length = 0;
```

```
            TXD_BUF[i].sum = 0;
                    TXD_BUF[i].state = 1;        //初始化的时候表示该发送缓冲区已经发送完数据，可以
使用
            if(i == (TXD_BUF_NUM - 1))
            {
                TXD_BUF[i].next = &     TXD_BUF[0] ;   //最后一个的尾指向第一个的头
            }
            else
            {
                TXD_BUF[i].next = &     TXD_BUF[i+1] ;
            }
        }

        for ( i=0; i<RXD_BUF_NUM; i++ )
        {
            RXD_BUF[i].command = 0;
            RXD_BUF[i].length = 0;
            RXD_BUF[i].sum = 0;
                    RXD_BUF[i].state = 0;      //初始化的时候表示全部接受没有完成，可以接收
            if(i == (RXD_BUF_NUM - 1))
            {
                RXD_BUF[i].next = &     RXD_BUF[0] ;
            }
            else
            {
                RXD_BUF[i].next = &     RXD_BUF[i+1] ;
            }
        }
    }
```

在 com1_ZB.h 中定义了数据结构 DATA_BUF，在 com1_ZB.c 中对链表收尾进行了关联。代码如下：

```
    struct DATA_BUF                                 //定义数据缓冲区结构体
    {
       unsigned char uart0;
       unsigned char uart1;
       unsigned char target_id;                     //目标地址
       unsigned char my_id;                         //自身地址
       unsigned char data[DATA_BUF_NUM];                 //数据缓冲区
       unsigned char command;                       //命令字
       unsigned char length;                        //实际通讯的字节数
       unsigned char sum;                           //所有的和
       unsigned char state;                         //0 (TRUE)：表示一个新的数据帧接收完，还未
```

```
处理
    unsigned char error_flag;                          //错误标志位
    struct DATA_BUF *next;                             //链表连接指针，指向下一个表头
};

Com1_ZB.c 文件中
struct DATA_BUF RXD_BUF[RXD_BUF_NUM];                  //定义 RXD_BUF_NUM 个接收缓冲区
的个数
struct DATA_BUF TXD_BUF[TXD_BUF_NUM];                  //定义 TXD_BUF_NUM 个发送缓冲区
的个数

struct DATA_BUF *RXD_BUF_P_FRONT = &RXD_BUF[0];        //接收缓冲队列头
struct DATA_BUF *RXD_BUF_P_REAR = &RXD_BUF[0];         //接收缓冲队列尾

struct DATA_BUF *TXD_BUF_P_FRONT = &TXD_BUF[0];        //发送缓冲队列头
struct DATA_BUF *TXD_BUF_P_REAR = &TXD_BUF[0];         //发送缓冲队列尾
```

在 stm32f10x_it.c 中有串口接收的中断服务函数和 DMA 发送结束函数。

```
void DMA1_Channel4_IRQHandler(void)          //串口 DMA 方式发送完成中断
{
    DMA_ClearFlag(DMA1_FLAG_TC4);
    TXD_BUF_P_FRONT = TXD_BUF_P_FRONT->next;
    TXD_BUF_P_FRONT->state = 1;
    DMA_Cmd(DMA1_Channel4,DISABLE);//发送完毕关闭 DMA，直到下次有数据发送时打开 DMA
}

void USART1_IRQHandler(void)          //串口 1 接收中断服务程序
{
  if(USART_GetITStatus(USART1, USART_IT_RXNE)!= RESET)
  {
      Com1_Rxd();          //接收数据到缓冲队列
      Com1_ProtocolAnalysis();     //处理数据
      USART_ClearITPendingBit(USART1,USART_IT_RXNE);
  }
}
```

追踪 Com1_ProtocolAnalysis();函数，如果接收到一个成功帧，可以看到最终接收到的数据在

```
    switch(com_pk->command)
        {
            case 0x01:
                Com1_WriteMemory(com_pk);
```

```
                break;
            case 0x02:
                Com1_ReadMemory(com_pk);
                break;
            default:
                com_pk->error_flag = 0x01;//无效指令
                Com1_SendBack(com_pk);
                break;
        }
```

例如，进入 Com1_ReadMemory();函数，可以看到数据在这里进行了最终解析，

```
    /*****************************************************************************
*   函数名称：void Com1_ReadMemory(struct DATA_BUF *readone_pk)
*   功      能：响应上位机的读指令，将读取的数据送入发送缓冲区
*   入口参数：缓冲区结构体
*   出口参数：
*****************************************************************************/
void Com1_ReadMemory(struct DATA_BUF *readone_pk)
{
    u8 start_id;
    start_id = readone_pk->data[0];
    TXD_BUF_P_REAR->uart0 = readone_pk->uart0;
    TXD_BUF_P_REAR->uart1 = readone_pk->uart1;
    TXD_BUF_P_REAR->target_id = readone_pk->my_id;
    TXD_BUF_P_REAR->my_id = NODE_ID;
    TXD_BUF_P_REAR->command = readone_pk->command;              //设置返回命令
    TXD_BUF_P_REAR->error_flag = 0;
    TXD_BUF_P_REAR->state = 0;
    TXD_BUF_P_REAR->data[0] = readone_pk->data[0];
    if((start_id >= INFO_START) && (start_id <= INFO_END))
    {
        switch(readone_pk->data[0])
        {
            case 0x11:      //车辆信息
                TXD_BUF_P_REAR->length = 3;
                TXD_BUF_P_REAR->data[1] = CAR_ID;
                TXD_BUF_P_REAR->data[2] = CAR_TYPE;
                break;
            case 0x12:          //运行命令,该指令读无意义
                TXD_BUF_P_REAR->error_flag = 0x02; //无效数据
                TXD_BUF_P_REAR->length = 0;
                break;
```

```
                case 0x13:              //运行状态
                    TXD_BUF_P_REAR->length = 2;
                    TXD_BUF_P_REAR->data[1] = CarMoveStatus;
                    break;
                case 0x14:              //位置编号
                    TXD_BUF_P_REAR->length = 4;
                    TXD_BUF_P_REAR->data[1] = AreaID_ASC2;
                    TXD_BUF_P_REAR->data[2] = StreetID_ASC2;
                    TXD_BUF_P_REAR->data[3] = RfidCardNum_HEX;
                    break;
                case 0x15:              //路径编码
                case 0x16:                  //车位编号
                default:
                    TXD_BUF_P_REAR->error_flag = 0x02;
                    break;
            }
        }
        else
            TXD_BUF_P_REAR->error_flag = 0x02;

            Com1_SetSum(TXD_BUF_P_REAR);                                    //设置校验和
            Com1_SendingData();
    }
```

最后将对应读取到的数据调用回执函数对上位机进行数据回复。至此完成了一个完整的读数据过程。

1.7、 L289N 电机驱动

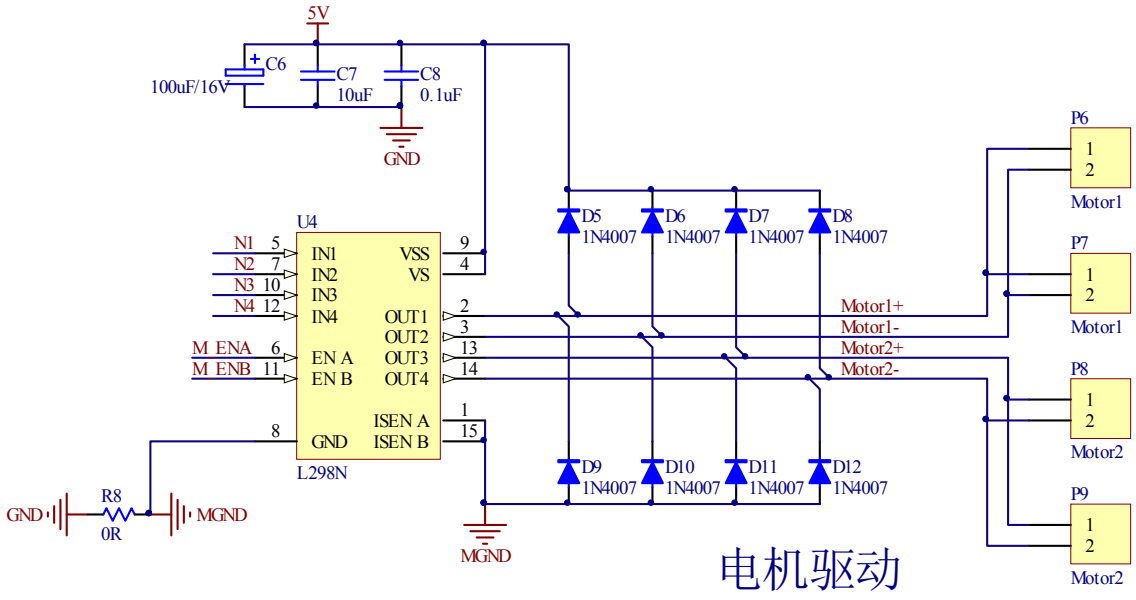


图 1.8 L298N 电机驱动电路

智能小车采用 L298N 作为电机的驱动芯片，L298N 是专用驱动集成电路，属于 H 桥集成电路，可以驱动两路电机，改变输入端的逻辑电平就可以改变电机的正反转。请对比以下前进和后退的代码

```
/*****************************************************************
* 函数名称：void Advance(u8 DutyCycle)
* 功能描述：小车前进
* 入口参数：速度
* 出口参数：无
*****************************************************************/
void Advance(u8 DutyCycle)
{

    u16 pulse = 0;
    GPIO_ResetBits(GPIOB,RCC_M1_N1_Pin);        //0
    GPIO_SetBits(GPIOB,RCC_M1_N2_Pin);          //1
    GPIO_ResetBits(GPIOB,RCC_M2_N3_Pin);        //0
    GPIO_SetBits(GPIOB,RCC_M2_N4_Pin);          //1
    pulse = (u16)DutyCycle * 9; //与 900 的比值
    SetMotorPWM(pulse,pulse,ENABLE);//通过改变两个电机的 PWM 占空比调速
    CarMoveStatus = NORMAL_FORWARD;//执行成功，改写当前运行状态
}
/*****************************************************************
* 函数名称：void Retreat(u8 DutyCycle)
* 功能描述：小车后退
* 入口参数：指定速度值,周期为 1s 的方波占空比，值为 1-1000
* 出口参数：无
*****************************************************************/
void Retreat(u8 DutyCycle)
{
    u16 pulse = 0;
    GPIO_SetBits(GPIOB,RCC_M1_N1_Pin);          //1
    GPIO_ResetBits(GPIOB,RCC_M1_N2_Pin);        //0
    GPIO_SetBits(GPIOB,RCC_M2_N3_Pin);          //1
    GPIO_ResetBits(GPIOB,RCC_M2_N4_Pin);        //0
    pulse = (u16)DutyCycle * 9;
    SetMotorPWM(pulse,pulse,ENABLE);
}
```

L298N 除了要用端口逻辑设置方向外，需要对 EN 端口加 PWM 来调节对应的速度，PWM 初始化代码如下：

```
/*****************************************************************
* 函数名称：void SetMotorPWM(u16 m1;u16 m2,FunctionalState NewState);
```

```
    * 功能描述：改变 PWM 占空比对电机调速
    * 入口参数：指定速度值,周期为 20ms 的方波占空比，值为 1-20
    * 出口参数：无
    ***********************************************************************/
    void SetMotorPWM(u16 m1,u16 m2,FunctionalState NewState)
    {
        TIM_OCInitTypeDef    TIM_OCInitStructure;
        /* Output Compare Active Mode configuration: Channel3 */
        TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1; //选择定时器模式:TIM 脉冲宽度调制模
式 2
        TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable; //比较输出使能
        TIM_OCInitStructure.TIM_Pulse = m1;//450                    //设置待装入捕获比较寄存器的脉
冲值，初始的占空比
        TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High; //输出极性:TIM 输出比较极性高
        TIM_OC3Init(TIM3, &TIM_OCInitStructure);    //根据 TIM_OCInitStruct 中指定的参数初始化外设
TIMx,PB0
        TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Enable);    //使能 TIMx 在 CCR2 上的预装载寄存器

        TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
        TIM_OCInitStructure.TIM_Pulse = m2;
        TIM_OC4Init(TIM3, &TIM_OCInitStructure);//PB1
        TIM_OC4PreloadConfig(TIM3, TIM_OCPreload_Enable);    //使能 TIMx 在 CCR2 上的预装载寄存器

        //上面两句中的 OC2 确定了是 channle 几，要是 OC3 则是 channel 3

        TIM_ARRPreloadConfig(TIM3, ENABLE); //使能 TIMx 在 ARR 上的预装载寄存器

        /* TIM3 enable counter */
        TIM_Cmd(TIM3, NewState);    //使能 TIMx 外设
    }
```

这样的话，就可以改变两边电机的 PWM 占空比调节左右两边的速度差，从而实现转向，例如右转代码如下：

```
    /***********************************************************************
    * 函数名称：void TurnRight(u8 DutyCycle,u8 Difference)
    * 功能描述：小车右转修正
    * 入口参数：弯道基速，百分比的差值
    * 出口参数：无
    ***********************************************************************/
    void TurnRight(u8 DutyCycle,u8 Difference)
    {
        u16 pulse1 = 0,pulse2 = 0,temp = 0;
```

```
        pulse1 = (u16)DutyCycle * 9;
        temp = Difference * 9;
        if(350 < temp)
        {
            GPIO_SetBits(GPIOB,RCC_M1_N2_Pin);
            GPIO_ResetBits(GPIOB,RCC_M1_N1_Pin);
            GPIO_SetBits(GPIOB,RCC_M2_N3_Pin); //M2 反转
            GPIO_ResetBits(GPIOB,RCC_M2_N4_Pin);
            if(temp < 500)
                pulse2 = InitialCornerSpeed * 10;
            else
                pulse2 = temp;
            pulse1 = pulse2;
        }
        else
        {
            GPIO_SetBits(GPIOB,RCC_M1_N2_Pin);
            GPIO_ResetBits(GPIOB,RCC_M1_N1_Pin);
            GPIO_SetBits(GPIOB,RCC_M2_N4_Pin);
            GPIO_ResetBits(GPIOB,RCC_M2_N3_Pin);
            pulse2 = pulse1 - temp;
            if(GarageOutFlag == 0)
                pulse1 += temp;
        }
        SetMotorPWM(pulse1,pulse2,ENABLE);//通过改变两个电机的 PWM 占空比调速
    }
```

2、 系统执行流程

打开 main.c 找到 main();主函数，首先是上电时的初始化，然后上报自身信息，并打开 Tim1 进行 50ms 一次的循环伺服周期。

```
        RCC_Configuration(); //时钟初始化
        NVIC_Configuration();
        BatteryVoltageAdConfig();
        CarInit();//小车所有硬件初始化
        Delayms(500);       //上电延时

    //  DingTest(500);
        Com1_Reporting(CAR_INFO,&info[0],2);//主动上报自身信息
        Tim1Init(50);//设置伺服采样周期为 50ms,开始自主运动
```

打开 stm32f10x_it.c 文件，可以找到 TIM1 的计数中断服务程序 TIM1_UP_IRQHandler(void)，每隔 50ms 读取一次磁导航数据 MagnetoMeasure();，并根据磁导航数据改变自身的运动状态 AutoMobile();

```
void TIM1_UP_IRQHandler(void)   //小车导航全部在这里完成
{
    static temp = 0;
    if ( TIM_GetITStatus(TIM1, TIM_IT_Update) == SET)
    {
        MagnetoMeasure();
//        temp ++;
//
//        if(temp >= 3)
//        {
            AutoMobile();
//            temp = 0;
//        }
        TIM_ClearITPendingBit(TIM1, TIM_FLAG_Update);
    }
}
```

接着还是回到 main()函数中，继续往下看，会进入 while(1)系统主循环，主要分为三个部分

1. 查询红外传感器的状态
2. 读取 RFID 地标
3. 判断电池电压是否过低

```
while(1)
    {
        /**********************查询障碍标志*************************/
        if((GPIO_ReadInputDataBit(IR_PORT,IR_Pin) != Bit_RESET) && (CarMoveStatus == NORMAL_STOP))
        {
            Delayms(4);//消抖
            if(GPIO_ReadInputDataBit(IR_PORT,IR_Pin) != Bit_RESET)
            {
                CarMoveStatus = NORMAL_FORWARD;
                //Advance(Percentage);
            }
        }
        /**************************结束*****************************/

        /**************************读地标*************************/
        if(RfidWriteFlag == 1)    //检查写卡标志位是否需要写卡操作
        {
            RfidTest();
        }
        else
```

```
        PositionCheck();

    if(GarageSuccessFlag == GARAGE_IN_SUCCESS)      //延时上报入库成功，防止数据帧丢失
    {
        DelayTemp ++;
        if(DelayTemp >= 10)
        {
            Com1_Reporting(MOVE_STATUS,&GarageSuccessFlag,1);
            GarageSuccessFlag = GARAGE_IN_FAIL;
            DelayTemp = 0;
        }
    }
    /*****************************地标结束**********************/

    /************************电池电压判定**************************/
    if(ADValue < 1966)//设定 9.6V 为放电截止电压，低于此值开始报警,实测 9.5 左右
    {

        Stop();
        TIM_Cmd(TIM1, DISABLE);//强制停止运行
        BeepOn();
        Delayms(500);
        BeepOff();
        Delayms(500);
        BeepOn();
        Delayms(1000);
        BeepOff();
        //ADFlag = 0;
    }
    /*********************************电压判定结束******************/
}
```

智能交通

第五章 视频定位

5.1 ITS 智能交通——视频定位系统用户手册

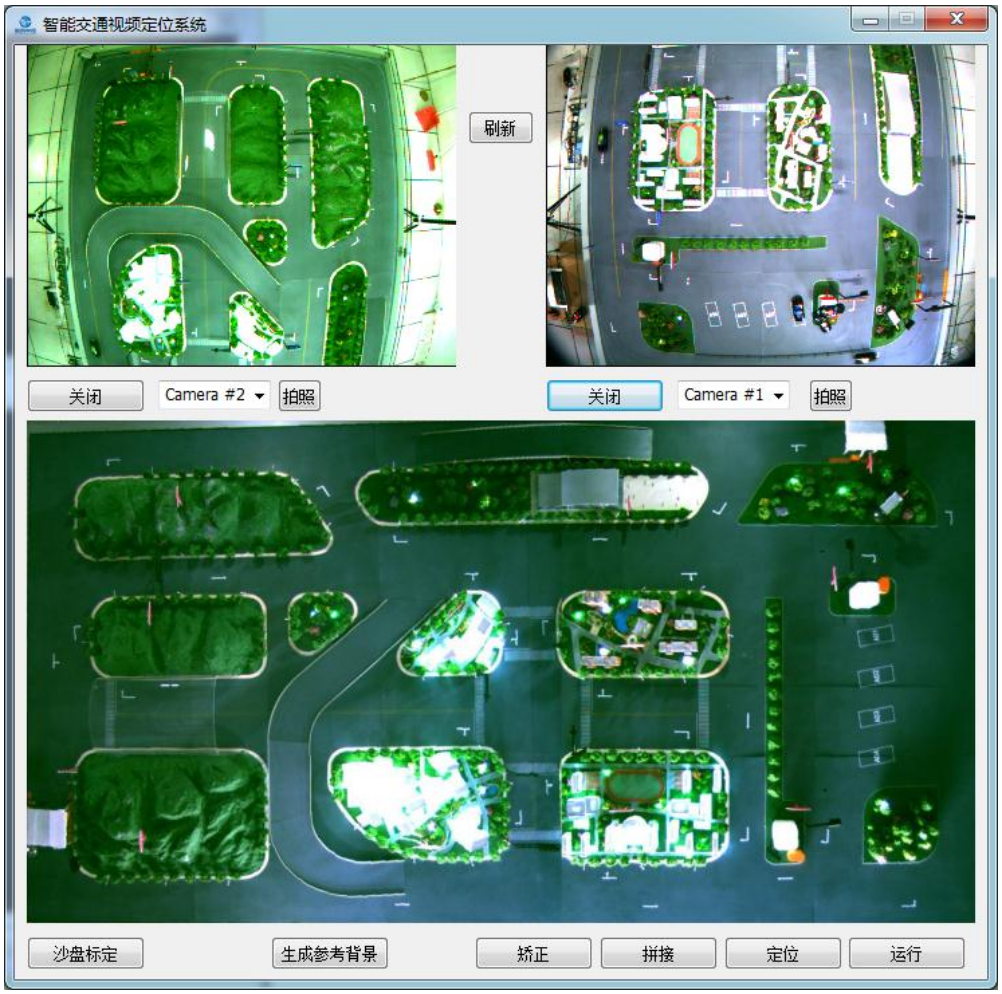
智能交通——视频定位系统用户手册

1、 视频定位软件使用指南

1.1、 使用前准备

使用之前，请搭建好摄像头支架，保证摄像头镜头和沙盘相对垂直，保证两个摄像头同时位于上盘的中心线上。

固定好支架后，把摄像头通过 USB 接入 PC，然后打开视频定位软件，如下图：



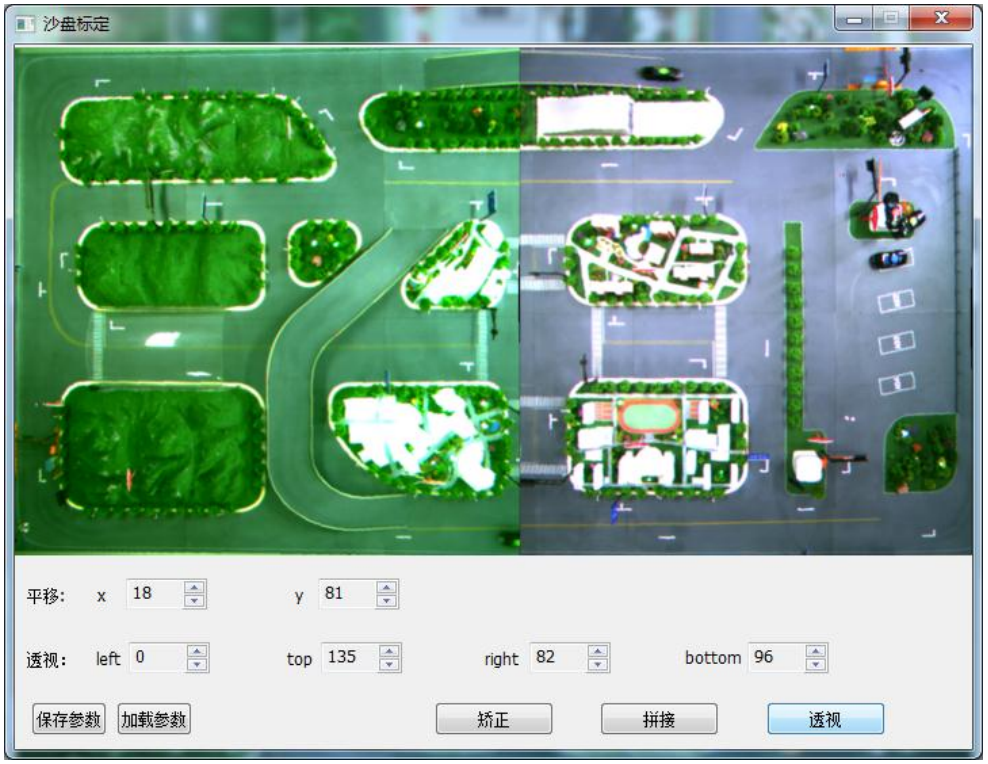
点击“打开”按钮，依次打开左右两个相机。需要注意的是软件中左侧子窗口需要对应的是左侧相机，右侧子窗口需要对应的是右侧相机。

打开相机后，可以根据窗口的图像再适当调整相机支架，让图像位于窗口的正中，左右间距相等。另外，在可能的情况下调整室内灯光，让沙盘比较明亮，拍摄的图片比较清晰。

1.2、 沙盘标定

做好上一步的准备后，需要进行沙盘标定。所谓沙盘标定，就是定位智能交通沙盘在摄像头视场中的位置。

点击“沙盘标定”按钮，弹出沙盘标定对话框，如下图：



首先来调整平移参数。所谓平移，是指右相机相对于左相机图像的位移量。由于两个相机拍摄的图像有重叠，需要通过这一步来实现图像的对接。X 标识水平的位移，Y 标识垂直方向的位移。通过点击响应的上下箭头，您可以看到图像的变化，不断调整直至图像达到比较好的拼接效果。

然后来调整透视参数，left、top、right、bottom 标识左、上、右、下的边界，您可以通过箭头来调节。通过透视参数，就可以切除窗口中沙盘边界以外的内容。

调整过程中，您可以通过点击“矫正”、“拼接”、“透视”来观察不同状态下的图像。调整结束后，您可以点击保存参数来保存调整好的参数，您也可以点击“加载参数”来加载之前保存好的参数。

标定结束后，关闭窗口返回到程序主窗口。

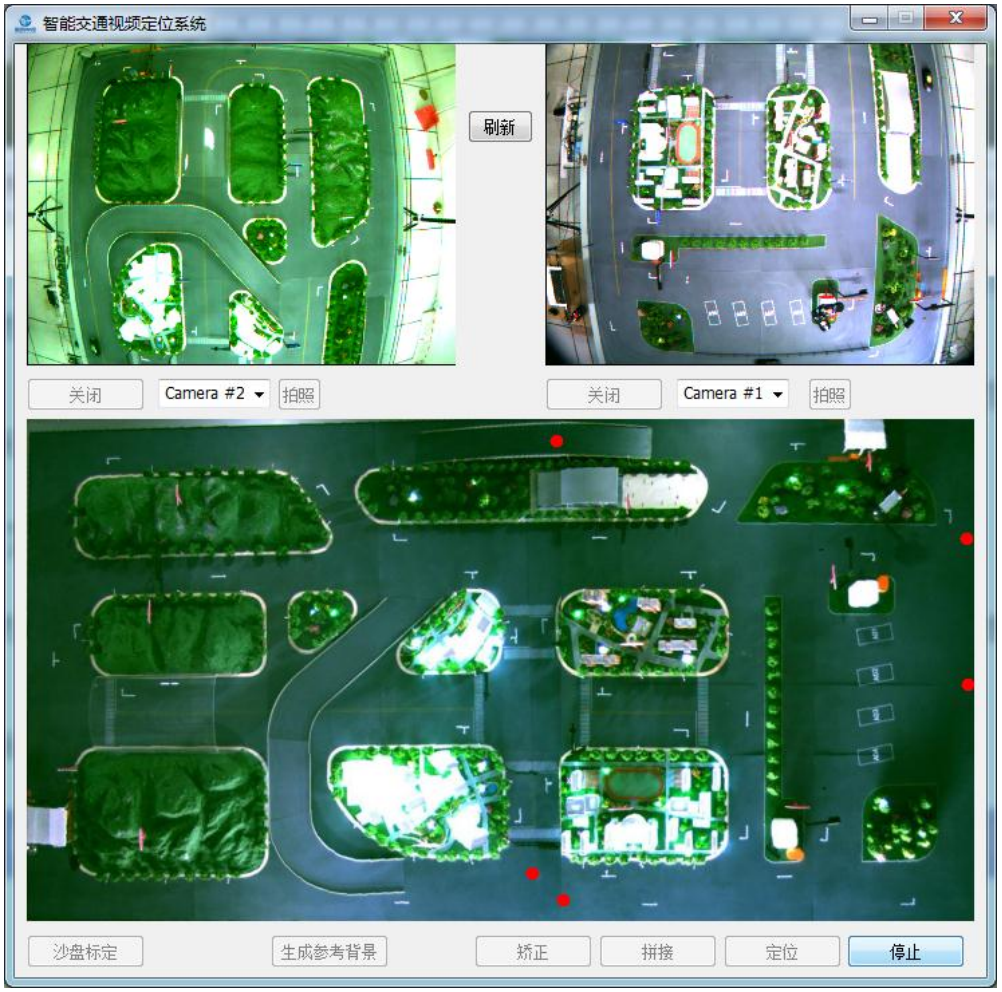
1.3、 视频定位运行

标定结束后，视频定位系统已经确定了执行的基本参数。这时就可以开始视频定位的运行了。

在开始运行之前，需要先点击“生成参考背景”按钮。这样就会在后台生成一副背景图片。需要说明的是，背景图片生成时，必须保证环境光源和实际运行时一致，另外要清理掉沙盘上任何不想关的物品，包括车辆。这是因为，视频定位运行时，是根据参考背景来计算车辆的位置的，如果光线或者别的干扰存在，会导致系统误识别。

生成背景后，可以在沙盘上运行一辆小车，然后依次点击“矫正”、“拼接”、“定位”三个按钮，如果前面的步骤都执行正确，则此时会在主窗口下方的窗口里出现小车的图标，标识小车定位成功。您可以再次点击按顺序点击这三个按钮，观察小车在窗口中位置的变化。

如果可以确认系统运行正确，则可以点击“运行”按钮，视频定位系统就会自动运行。这时您会看到小车的位置在不断的变化，窗口中的位置会和小车实际相对于沙盘的位置对应。

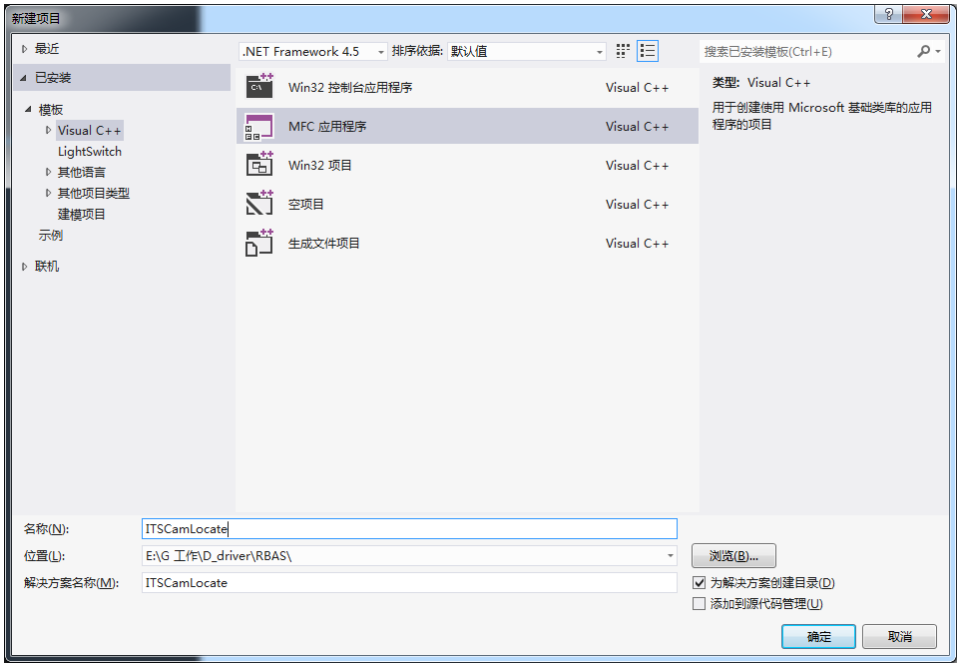


2、 视频定位软件开发实例

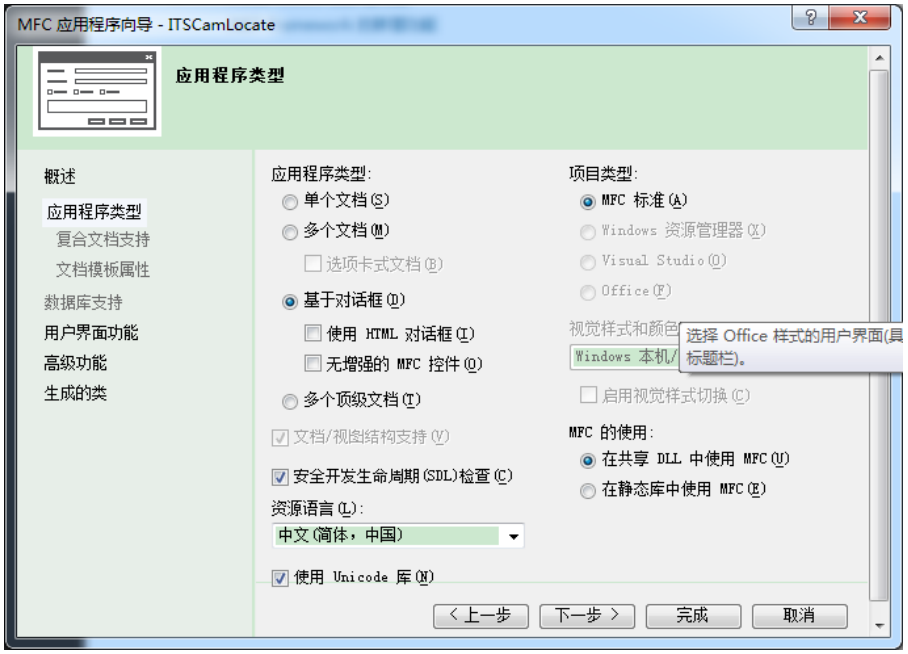
视频定位软件基于 VS2012 开发。在进行本实例的学习前，请先安装 Visual Studio 2012。

2.1、 新建工程

新建基于对话框的工程 ITCCamLocate，如下图：



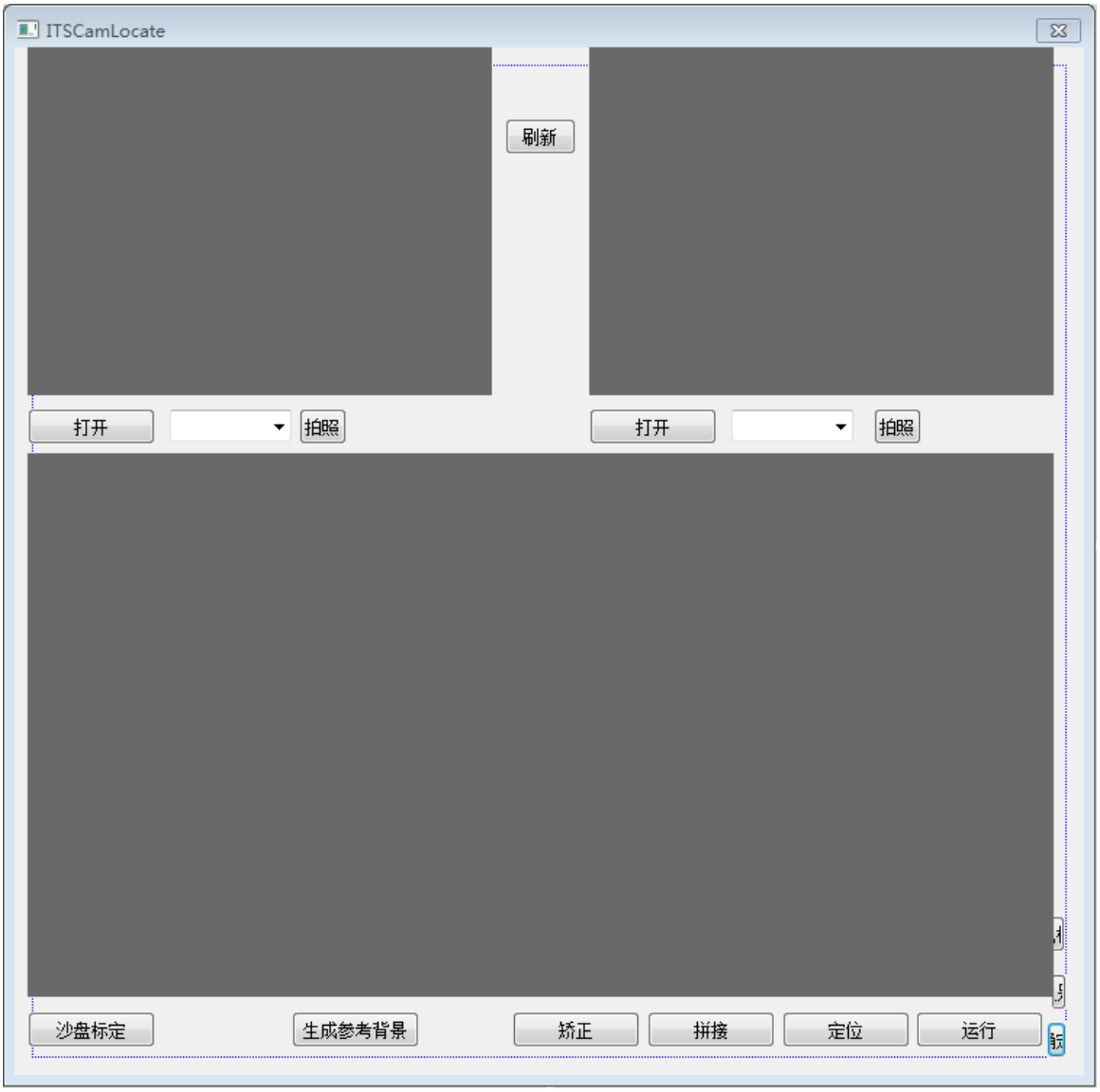
在上图的新建工程对话框中，输入功成名，选择工程的路径，然后点击确定，再点击下一步，进入应用程序类型选择，选择“基于对话框”，如下图所示：



点击“完成”按钮，即可完成工程的创建。

2.2、 编辑主界面

编辑主对话框，如下图，分别设置各个控件的 ID。



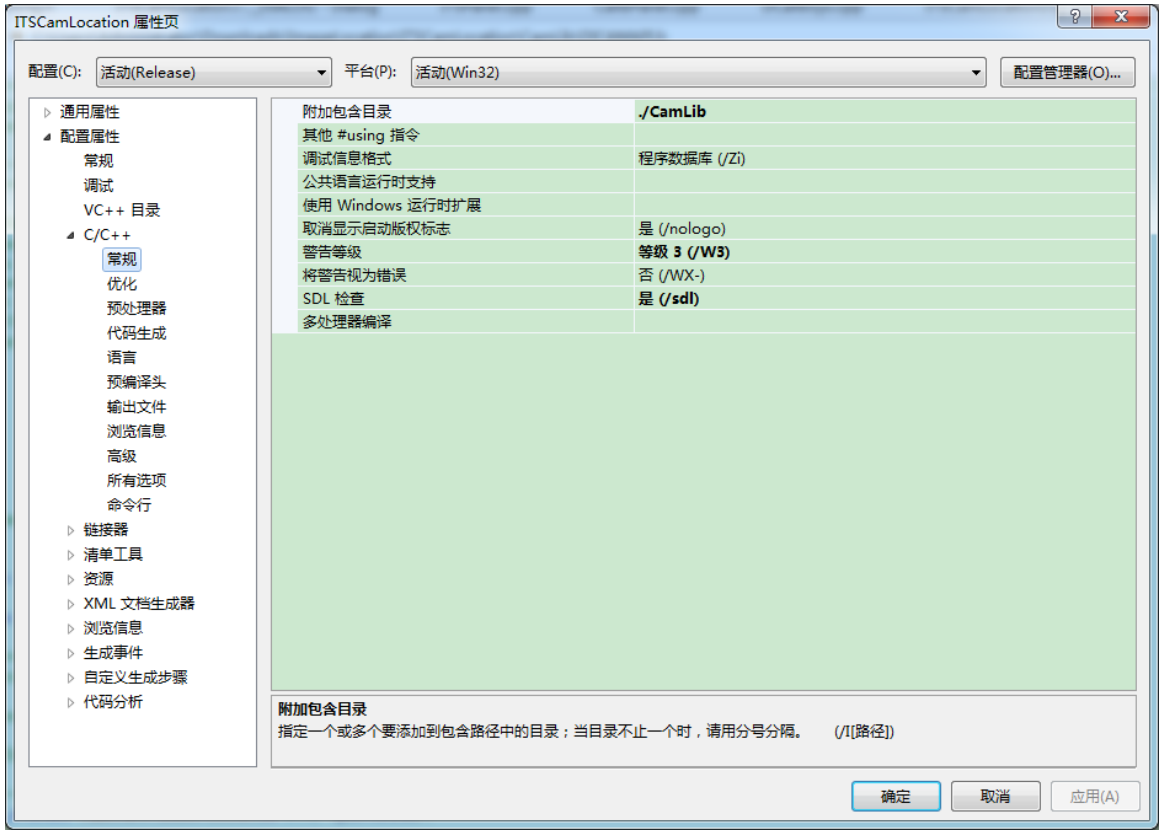
2.3、 实现相机的操作

添加相机操作的类、库文件，实现相机的打开关闭功能；

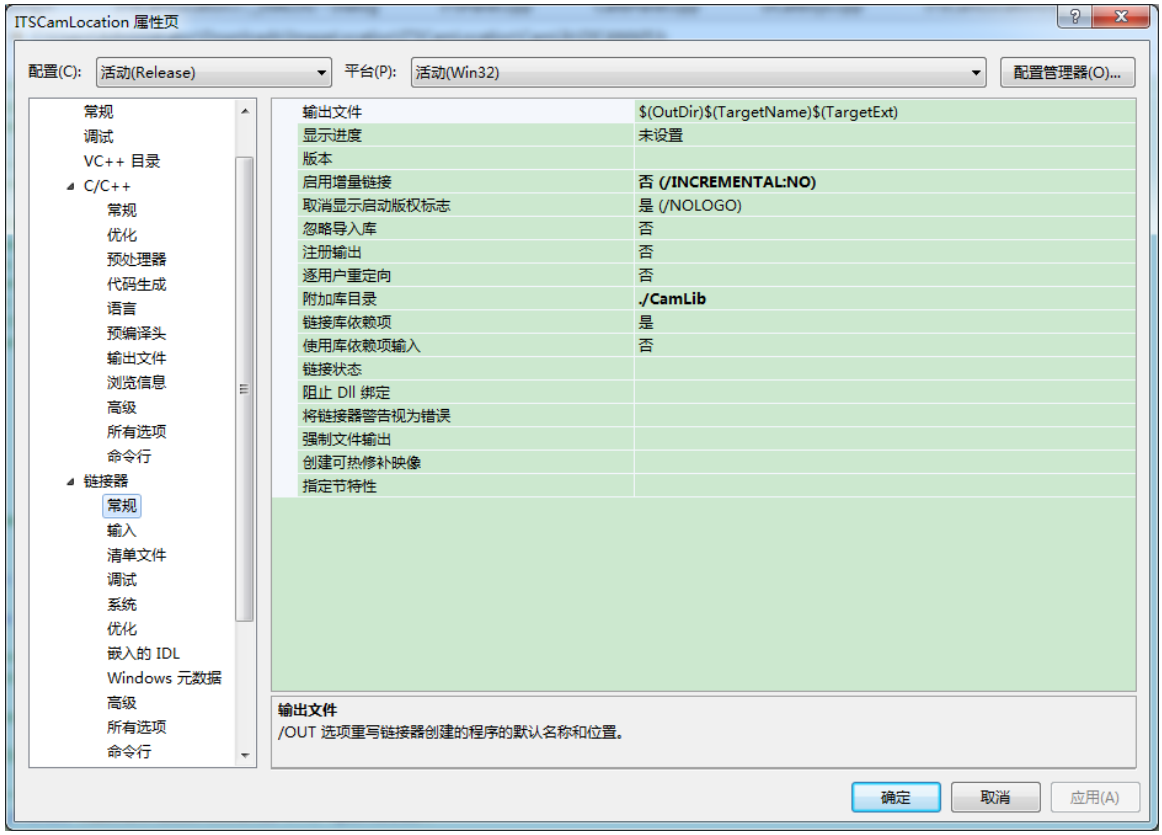
在 ITSCamLocateDlg.h 文件添加以下包含内容，引入相机操作的库函数头文件：

```
#include "DICAMAPI.h"
#include "DICAM2API.h"
```

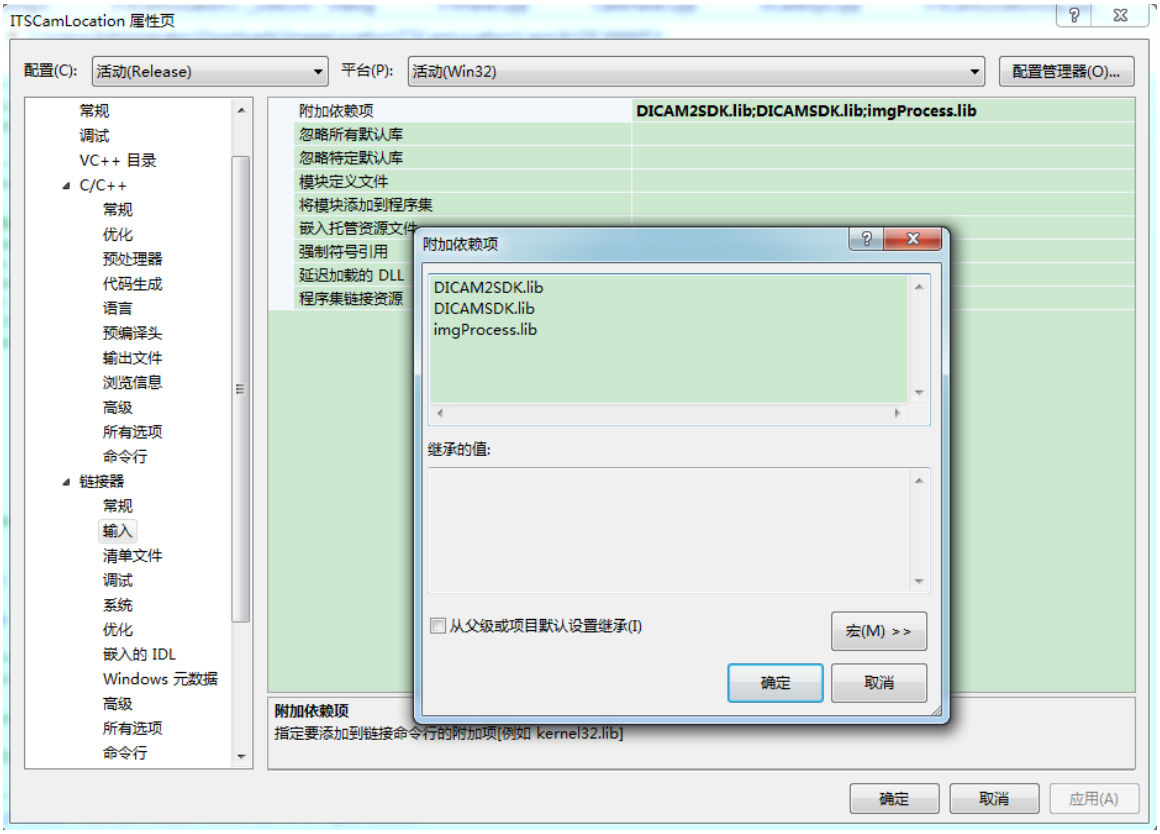
把实例工程中的 CamLib 文件夹拷贝到当前工程文件夹下，在工程属性中设置“附加包含目录”，如下图：



在链接器->常规中，设置附加库目录，如下图：



在链接器->输入中，输入如下图所示的三个 lib 文件，其中前两个为相机操作，后一个为图像处理的库文件。



在 ITSCamLocateDlg.h 文件中，添加如下代码。其中 CamList1 和 CamList2 对应对话框中的下拉框控件，可以通过添加关联变量的形式添加。

```
private:
 void SetDisplayWindow();
 void ProcessImage(int camIndex,BYTE * pbuf);
 void Process();
 void ScanCamera();
 BOOL IsCameraOpen();

 static int CALLBACK SnapThreadCallback(BYTE *pBuffer,    DI_DATA_TYPE Type, LPVOID lpContext);
 static int CALLBACK SnapThread2Callback(BYTE *pBuffer,    DI_DATA_TYPE Type, LPVOID lpContext);

 static UINT ProcessThread(LPVOID pParam);
private:
 BYTE m_RunMode;
 BYTE m_RunMode2;
 CComboBox m_CamList1;
 CComboBox m_CamList2;
```

```
DI_RESOLUTION m_CaptureResolution;
DI_RESOLUTION m_Resolution;

//CImageProc m_ImageProc;

CWinThread      *       m_pProcessThread;
BOOL                    m_bThreadRun;
```

在 ITSCamLocateDlg.cpp 中添加如下宏定义：

```
#define ROI_HOFFSET    640
#define ROI_VOFFSET    480
#define ROI_WIDTH      1024
#define ROI_HEIGHT     768
```

在构造函数中增加如下红色代码：

```
CITSCamLocateDlg::CITSCamLocateDlg(CWnd* pParent /*=NULL*/)
 : CDialogEx(CITSCamLocateDlg::IDD, pParent)
{
 m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);

 m_RunMode = RUNMODE_STOP;
 m_RunMode2 = RUNMODE_STOP;
 m_Resolution = R1024_768;
 m_CaptureResolution = R1024_768;

 m_bThreadRun = FALSE;
 m_pProcessThread = NULL;
}
```

在 ITSCamLocateDlg.cpp 中添加如下函数定义：

```
void CITSCamLocateDlg::SetDisplayWindow()
{
 GetDlgItem(IDC_STA_V1)->SetWindowPos(NULL, 0,0, 320, 240,
        SWP_NOMOVE | SWP_SHOWWINDOW);
 GetDlgItem(IDC_STA_V2)->SetWindowPos(NULL, 0,0, 320, 240,
        SWP_NOMOVE | SWP_SHOWWINDOW);
}

int CALLBACK CITSCamLocateDlg::SnapThreadCallback(BYTE *pBuffer,    DI_DATA_TYPE Type, LPVOID lpContext)
{
 CITSCamLocateDlg *pCtrl = (CITSCamLocateDlg *)lpContext;
 if(Type==DATA_TYPE_RGB24)
```

```
 {
        pCtrl->ProcessImage(1,pBuffer);
 }

 return TRUE;
}

int CALLBACK CITSCamLocateDlg::SnapThread2Callback(BYTE *pBuffer,    DI_DATA_TYPE Type, LPVOID lpContext)
{
 CITSCamLocateDlg *pCtrl = (CITSCamLocateDlg *)lpContext;
 if(Type==DATA_TYPE_RGB24)
 {
        pCtrl->ProcessImage(2,pBuffer);
 }
 return TRUE;
}

void CITSCamLocateDlg::ProcessImage(int camIndex,BYTE * pBuf)
{
 if (g_pImageProc==NULL)
 {
        return;
 }
 g_pImageProc->SaveData(camIndex,pBuf);
}
void CITSCamLocateDlg::ScanCamera()
{
 BYTE CamAllNum = 0;
 BYTE Resolution = 0;
 BYTE CapRes = 0;
 CameraGetMultiCameraNumber(&CamAllNum, &Resolution, &CapRes);

 m_CamList1.ResetContent();
 m_CamList2.ResetContent();
 for(int i=0; i<CamAllNum; i++)
 {
        CString xtmp;
        xtmp.Format(TEXT("Camera #%d"), i+1);
        m_CamList1.AddString(xtmp);
        m_CamList2.AddString(xtmp);
 }
```

```
 m_CamList1.SetCurSel(CamAllNum-2);
 m_CamList2.SetCurSel(CamAllNum-1);
}
```

这里引入了全局的图像处理对象 g_pImageProc，为了便于全局访问，需要把它定义在 stdafx.cpp 中，如下所示。

```
CImageProc * g_pImageProc = NULL;
```

然后在头文件 stdafx.h 中进行声明。

```
#include "ImageProc.h"
extern CImageProc * g_pImageProc;
```

在 CITSCamLocateApp 的构造函数中添加如下代码，完成对全局对象的初始化。

```
 if (g_pImageProc == NULL)
 {
        g_pImageProc = new CImageProc();
 }
```

紧接着，从实例工程中拷贝 ImageProc.h 和 ImageProc.cpp 文件到当前工程目录，然后在项目中添加这两个文件。

接下来，添加上图中各按钮的响应函数，在左侧的第一个“打开”按钮的响应函数中，添加如下代码：

```
void CITSCamLocationDlg::OnBnClickedBtnPlay1()
{
 if (m_RunMode == RUNMODE_PLAY)
 {
        CameraStop();
        CameraUnInit();
        SetDlgItemText(IDC_BTN_PLAY1, TEXT("打开"));
        //GetDlgItem(IDC_SETUP)->EnableWindow(FALSE);
        //GetDlgItem(IDC_RESOLUTION)->EnableWindow(TRUE);
        m_RunMode = RUNMODE_STOP;
 }
 else
 {
        BYTE CamNum = m_CamList1.GetCurSel()+1;
        if (m_CamList1.GetCount() == 0)
        {
                CamNum = 0;
        }
        if (R_ROI == m_Resolution)
        {
                CameraSetROI(ROI_HOFFSET, ROI_VOFFSET, ROI_WIDTH, ROI_HEIGHT);
```

```
            }
            if (STATUS_OK != CameraInit(CamNum,m_Resolution,GetDlgItem(IDC_STA_V1)->m_hWnd, SnapThreadCallback,
this) )
            {
                MessageBox(TEXT("相机初始化失败！"), TEXT("错误"), MB_OK | MB_ICONSTOP);
                return ;
            }

            //set AWB window
            {
                INT width = 0;
                INT hight = 0;
                CameraGetImageSize(&width, &hight);
                CameraSetWBWindow(width>>2, hight>>2, width>>1, hight>>1);
            }
            SetDlgItemText(IDC_BTN_PLAY1, TEXT("关闭"));
            //GetDlgItem(IDC_SETUP)->EnableWindow(TRUE);
            //GetDlgItem(IDC_RESOLUTION)->EnableWindow(TRUE);

            //CameraSetMessage(this->m_hWnd, WM_USERDEFMSG);
            SetDisplayWindow();

            CameraPlay();
            m_RunMode = RUNMODE_PLAY;
        }
        //      GetDlgItem(IDC_STA_V1)->UpdateWindow();
    }
```

添加右侧的“打开”按钮的响应函数中，添加如下代码：

```
    if (m_RunMode2 == RUNMODE_PLAY)
    {
        Camera2Stop();
        Camera2UnInit();
        SetDlgItemText(IDC_BTN_PLAY2, TEXT("打开"));
        m_RunMode2 = RUNMODE_STOP;

        GetDlgItem(IDC_STA_V1)->Invalidate();
    }
    else
    {
        BYTE CamNum = m_CamList2.GetCurSel()+1;
        if (m_CamList2.GetCount() == 0)
```

```
        {
            CamNum = 0;
        }
        if (R_ROI == m_Resolution)
        {
            Camera2SetROI(ROI_HOFFSET, ROI_VOFFSET, ROI_WIDTH, ROI_HEIGHT);
        }
        if (STATUS_OK != Camera2Init(CamNum,m_Resolution,GetDlgItem(IDC_STA_V2)->m_hWnd, SnapThread2Callback,
this) )
        {
            MessageBox(TEXT("相机初始化失败"), TEXT("Error"), MB_OK | MB_ICONSTOP);
            return ;
        }

        //set AWB window
        {
            INT width = 0;
            INT hight = 0;
            Camera2GetImageSize(&width, &hight);
            Camera2SetWBWindow(width>>2, hight>>2, width>>1, hight>>1);
        }
        SetDlgItemText(IDC_BTN_PLAY2, TEXT("关闭"));
        SetDisplayWindow();

        Camera2Play();
        m_RunMode2 = RUNMODE_PLAY;
    }
```

添加刷新按钮的响应代码，如下：

```
    CString str1,str2;
    GetDlgItemText(IDC_BTN_PLAY1,str1);
    GetDlgItemText(IDC_BTN_PLAY2,str2);

    if ((str1==L"关闭")||(str2==L"关闭"))
    {
        return;
    }
    ScanCamera();
```

在 OnInitDialog 中添加 ScanCamera 的调用。

接下来，编译整个工程，如果您的操作无误，此时可以编译通过没有错误。

如果您需要查看程序的运行效果，一定要把 DICAM2SDK.dll 和 DICAMSDK.dll 拷贝到可执行程序目录下才可以运行。

2.4、 添加 GDI+支持

为了方便界面绘图，我们在工程中引入 GDI+的库和声明。

首先，在 stdafx.h 中，添加如下代码：

```
#include <gdiplus.h>
using namespace Gdiplus;
```

在 ITSCamLocate.h 中，增加如下类成员和函数：

```
private:
 ULONG_PTR    m_GdiplusToken;
 GdiplusStartupInput m_GdiplusInput;
public:
 virtual int ExitInstance();
```

在 ITSCamLocate.cpp 的函数 CITSCamLocationApp 中，添加如下红色的代码：

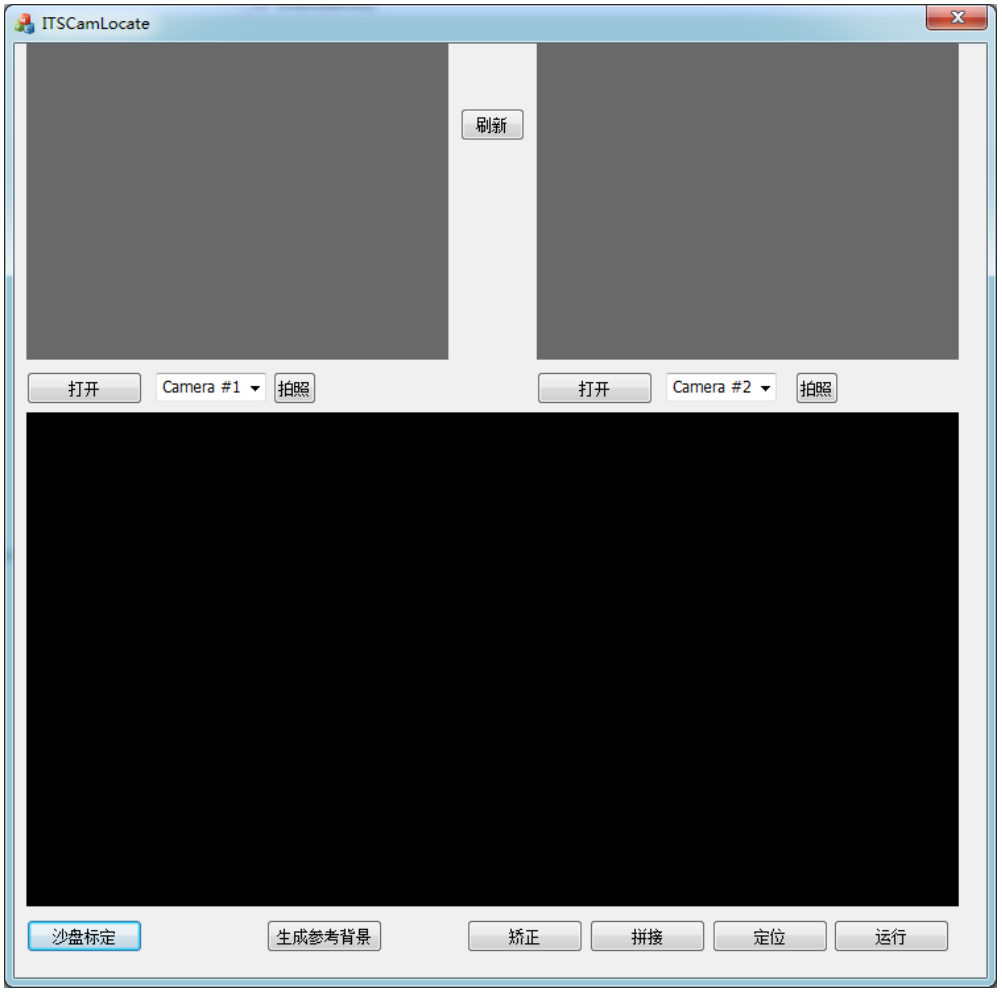
```
BOOL CITSCamLocateApp::InitInstance()
{
 // 如果一个运行在 Windows XP 上的应用程序清单指定要
 // 使用 ComCtl32.dll 版本 6 或更高版本来启用可视化方式，
 //则需要 InitCommonControlsEx()。否则，将无法创建窗口。
 INITCOMMONCONTROLSEX InitCtrls;
 InitCtrls.dwSize = sizeof(InitCtrls);
 // 将它设置为包括所有要在应用程序中使用的
 // 公共控件类。
 InitCtrls.dwICC = ICC_WIN95_CLASSES;
 InitCommonControlsEx(&InitCtrls);

 CWinApp::InitInstance();

 GdiplusStartup(&m_GdiplusToken,&m_GdiplusInput,NULL);

 AfxEnableControlContainer();

 // 创建 shell 管理器，以防对话框包含
 // 任何 shell 树视图控件或 shell 列表视图控件。
 CShellManager *pShellManager = new CShellManager;

 // 激活“Windows Native”视觉管理器，以便在 MFC 控件中启用主题
 CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS(CMFCVisualManagerWindows));
```

```
    // 标准初始化
    // 如果未使用这些功能并希望减小
    // 最终可执行文件的大小，则应移除下列
    // 不需要的特定初始化例程
    // 更改用于存储设置的注册表项
    // TODO: 应适当修改该字符串，
    // 例如修改为公司或组织名
    SetRegistryKey(_T("应用程序向导生成的本地应用程序"));

    CITSCamLocateDlg dlg;
    m_pMainWnd = &dlg;
    INT_PTR nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: 在此放置处理何时用
        //  “确定”来关闭对话框的代码
    }
    else if (nResponse == IDCANCEL)
    {
        // TODO: 在此放置处理何时用
        //  “取消”来关闭对话框的代码
    }
    else if (nResponse == -1)
    {
        TRACE(traceAppMsg, 0, "警告: 对话框创建失败，应用程序将意外终止。\n");
        TRACE(traceAppMsg, 0, "警告: 如果您在对话框上使用 MFC 控件，则无法 #define
_AFX_NO_MFC_CONTROLS_IN_DIALOGS。\n");
    }

    // 删除上面创建的 shell 管理器。
    if (pShellManager != NULL)
    {
        delete pShellManager;
    }

    // 由于对话框已关闭，所以将返回 FALSE 以便退出应用程序，
    //  而不是启动应用程序的消息泵。
    return FALSE;
}
```

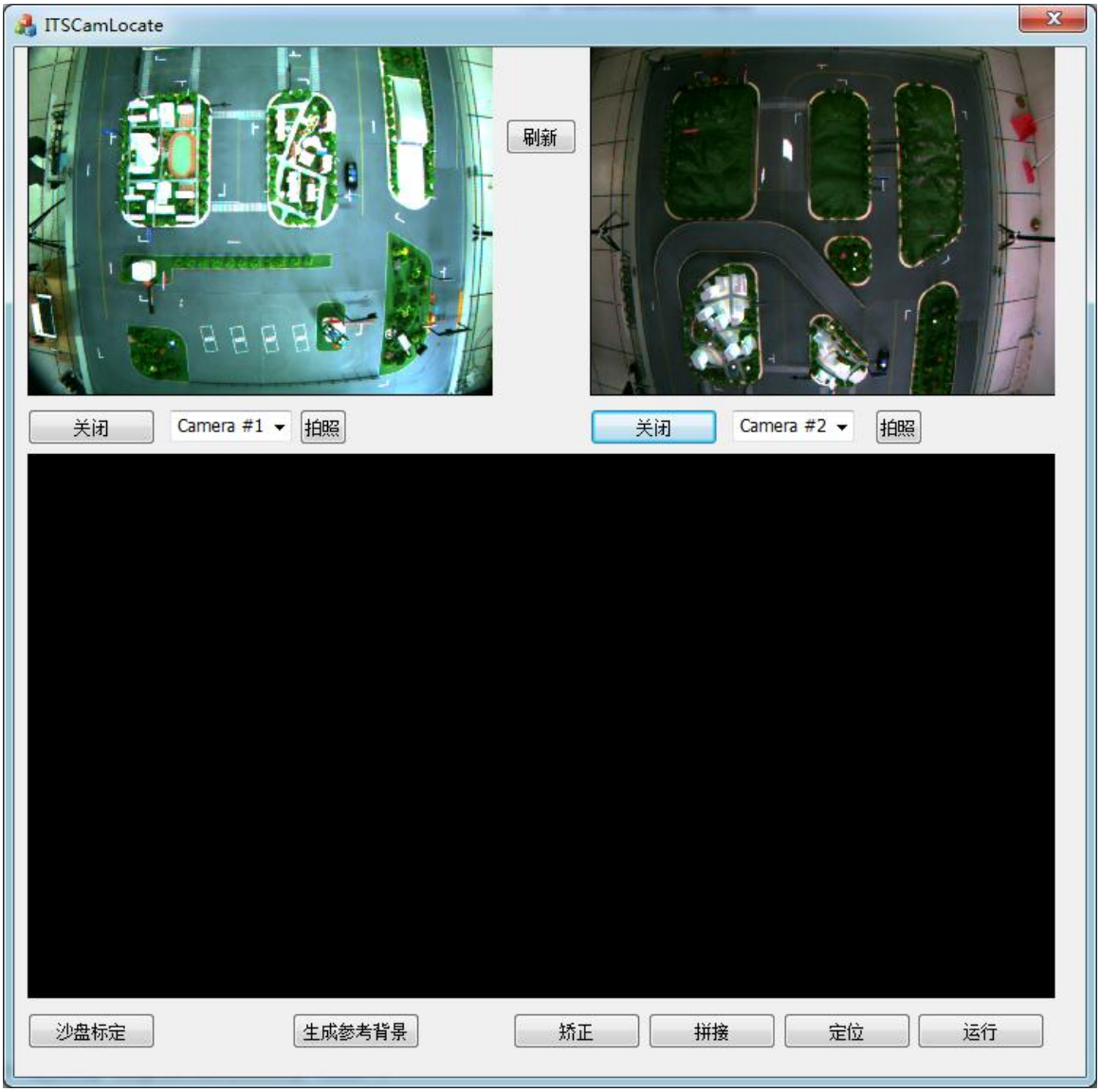
在 ITSCamLocate.cpp 的增加函数如下：

```
int CITSCamLocateApp::ExitInstance()
{
 // TODO: 在此添加专用代码和/或调用基类
 GdiplusShutdown(m_GdiplusToken);
 return CWinApp::ExitInstance();
}
```

编译运行程序，如下图所示：



点击打开相机，即可看到相机的实时图像：

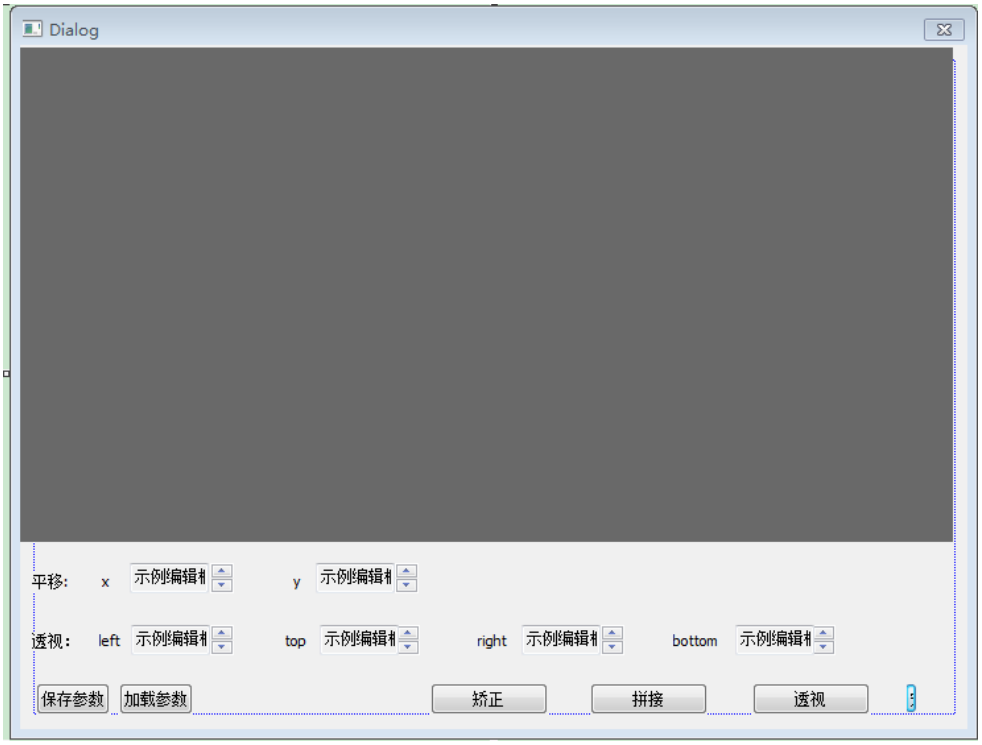


2.5、 沙盘标定

接下来，需要对沙盘进行标定，所谓标定，就是设置视频窗口中沙盘的区域，删除无关的区域。

这里的标定有两种：2 个相机的相对位置标定和透视标定。两个相机的相对位置标定是为了消除相机的差异性带来的误差，透视标定是为了去除不相关的区域。

在工程中添加一个对话框，如下图所示，



根据这个对话框生成对应的类，如下：

```
#pragma once
#include "calibpanel.h"
#include "afxcmn.h"
// CDCalibSys 对话框

class CDCalibSys : public CDialogEx
{
 DECLARE_DYNAMIC(CDCalibSys)

public:
 CDCalibSys(CWnd* pParent = NULL);    // 标准构造函数
 virtual ~CDCalibSys();

// 对话框数据
 enum { IDD = IDD_DLG_CALIBC };

protected:
 virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV 支持

 DECLARE_MESSAGE_MAP()
public:
 afx_msg void OnBnClickedBtnCorrect();
 afx_msg void OnBnClickedBtnStitch();
```

```
 afx_msg void OnDeltaposSpinXpos(NMHDR *pNMHDR, LRESULT *pResult);
 afx_msg void OnDeltaposSpinYpos(NMHDR *pNMHDR, LRESULT *pResult);
 afx_msg void OnDeltaposSpinLeft(NMHDR *pNMHDR, LRESULT *pResult);
 afx_msg void OnDeltaposSpinTop(NMHDR *pNMHDR, LRESULT *pResult);
 afx_msg void OnDeltaposSpinRight(NMHDR *pNMHDR, LRESULT *pResult);
 afx_msg void OnDeltaposSpinBottom(NMHDR *pNMHDR, LRESULT *pResult);
 afx_msg void OnBnClickedBtnPersp();
 virtual BOOL OnInitDialog();

 CCalibPanel m_CalibPanel;
 CSpinButtonCtrl m_spinXpos;
 CSpinButtonCtrl m_spinYpos;
 CSpinButtonCtrl m_spinLCut;
 CSpinButtonCtrl m_spinTCut;
 CSpinButtonCtrl m_spinRCut;
 CSpinButtonCtrl m_spinBCut;
 afx_msg void OnBnClickedBtnSave();
 afx_msg void OnBnClickedBtnLoad();
};

// DCalibSys.cpp : 实现文件
//

#include "stdafx.h"
#include "ITSCamLocation.h"
#include "DCalibSys.h"
#include "afxdialogex.h"


// CDCalibSys 对话框

IMPLEMENT_DYNAMIC(CDCalibSys, CDialogEx)

CDCalibSys::CDCalibSys(CWnd* pParent /*=NULL*/)
 : CDialogEx(CDCalibSys::IDD, pParent)
{

}

CDCalibSys::~CDCalibSys()
{
```

```
}

void CDCalibSys::DoDataExchange(CDataExchange* pDX)
{
 CDialogEx::DoDataExchange(pDX);
 DDX_Control(pDX, IDC_STA_V3, m_CalibPanel);
 DDX_Control(pDX, IDC_SPIN_XPOS, m_spinXpos);
 DDX_Control(pDX, IDC_SPIN_YPOS, m_spinYpos);
 DDX_Control(pDX, IDC_SPIN_LEFT, m_spinLCut);
 DDX_Control(pDX, IDC_SPIN_TOP, m_spinTCut);
 DDX_Control(pDX, IDC_SPIN_RIGHT, m_spinRCut);
 DDX_Control(pDX, IDC_SPIN_BOTTOM, m_spinBCut);
}


BEGIN_MESSAGE_MAP(CDCalibSys, CDialogEx)
 ON_BN_CLICKED(IDC_BTN_CORRECT, &CDCalibSys::OnBnClickedBtnCorrect)
 ON_BN_CLICKED(IDC_BTN_STITCH, &CDCalibSys::OnBnClickedBtnStitch)
 ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN_XPOS, &CDCalibSys::OnDeltaposSpinXpos)
 ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN_YPOS, &CDCalibSys::OnDeltaposSpinYpos)
 ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN_LEFT, &CDCalibSys::OnDeltaposSpinLeft)
 ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN_TOP, &CDCalibSys::OnDeltaposSpinTop)
 ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN_RIGHT, &CDCalibSys::OnDeltaposSpinRight)
 ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN_BOTTOM, &CDCalibSys::OnDeltaposSpinBottom)
 ON_BN_CLICKED(IDC_BTN_PERSP, &CDCalibSys::OnBnClickedBtnPersp)
 ON_BN_CLICKED(IDC_BTN_SAVE, &CDCalibSys::OnBnClickedBtnSave)
 ON_BN_CLICKED(IDC_BTN_LOAD, &CDCalibSys::OnBnClickedBtnLoad)
END_MESSAGE_MAP()


// CDCalibSys 消息处理程序


void CDCalibSys::OnBnClickedBtnCorrect()
{
 if (g_pImageProc==NULL)
 {
        return;
 }
 //畸变矫正
 g_pImageProc->Cam1Correct();
```

```
 g_pImageProc->Cam2Correct();
}


void CDCalibSys::OnBnClickedBtnStitch()
{
 if (g_pImageProc==NULL)
 {
        return;
 }
 g_pImageProc->StitchImage(true);

 m_CalibPanel.UpdateStitchBmp();
}


void CDCalibSys::OnDeltaposSpinXpos(NMHDR *pNMHDR, LRESULT *pResult)
{
 LPNMUPDOWN pNMUpDown = reinterpret_cast<LPNMUPDOWN>(pNMHDR);

 if (g_pImageProc==NULL)
 {
        return;
 }
 int value = m_spinXpos.GetPos32();
 g_pImageProc->SetXOffset(value);

 OnBnClickedBtnStitch();

 *pResult = 0;
}


void CDCalibSys::OnDeltaposSpinYpos(NMHDR *pNMHDR, LRESULT *pResult)
{
 LPNMUPDOWN pNMUpDown = reinterpret_cast<LPNMUPDOWN>(pNMHDR);

 int value = m_spinYpos.GetPos32();
 g_pImageProc->SetYOffset(value);

 OnBnClickedBtnStitch();
```

```
 *pResult = 0;
}


void CDCalibSys::OnDeltaposSpinLeft(NMHDR *pNMHDR, LRESULT *pResult)
{
 LPNMUPDOWN pNMUpDown = reinterpret_cast<LPNMUPDOWN>(pNMHDR);
 int value = m_spinLCut.GetPos32();
 g_pImageProc->SetLCut(value);

 OnBnClickedBtnPersp();
 *pResult = 0;
}


void CDCalibSys::OnDeltaposSpinTop(NMHDR *pNMHDR, LRESULT *pResult)
{
 LPNMUPDOWN pNMUpDown = reinterpret_cast<LPNMUPDOWN>(pNMHDR);
 int value = m_spinTCut.GetPos32();
 g_pImageProc->SetTCut(value);

 OnBnClickedBtnPersp();

 *pResult = 0;
}


void CDCalibSys::OnDeltaposSpinRight(NMHDR *pNMHDR, LRESULT *pResult)
{
 LPNMUPDOWN pNMUpDown = reinterpret_cast<LPNMUPDOWN>(pNMHDR);
 int value = m_spinRCut.GetPos32();
 g_pImageProc->SetRCut(value);
 OnBnClickedBtnPersp();

 *pResult = 0;
}


void CDCalibSys::OnDeltaposSpinBottom(NMHDR *pNMHDR, LRESULT *pResult)
{
 LPNMUPDOWN pNMUpDown = reinterpret_cast<LPNMUPDOWN>(pNMHDR);
```

```
 int value = m_spinBCut.GetPos32();
 g_pImageProc->SetBCut(value);
 OnBnClickedBtnPersp();

 *pResult = 0;
}

void CDCalibSys::OnBnClickedBtnPersp()
{
 if (g_pImageProc==NULL)
 {
        return;
 }
 g_pImageProc->PerspCorrect(true);

 m_CalibPanel.UpdatePersBmp();
}


BOOL CDCalibSys::OnInitDialog()
{
 CDialogEx::OnInitDialog();

 m_spinXpos.SetRange32(0,200);
 m_spinXpos.SetPos32(82);

 m_spinYpos.SetRange32(0,200);
 m_spinYpos.SetPos32(90);

 m_spinLCut.SetRange32(0,200);
 m_spinLCut.SetPos32(0);

 m_spinRCut.SetRange32(0,200);
 m_spinRCut.SetPos32(0);

 m_spinTCut.SetRange32(0,300);
 m_spinTCut.SetPos32(0);

 m_spinBCut.SetRange32(0,300);
 m_spinBCut.SetPos32(0);
```

```
 OnBnClickedBtnLoad();
 return TRUE;  // return TRUE unless you set the focus to a control
 // 异常: OCX 属性页应返回 FALSE
}


void CDCalibSys::OnBnClickedBtnSave()
{
 CFile file;
 if(file.Open(L"./data/crc.txt",CFile::modeCreate|CFile::modeWrite))
 {
        CArchive ar(&file,CArchive::store);

        int value = m_spinXpos.GetPos32();
        ar<<value;

        value = m_spinYpos.GetPos32();
        ar<<value;

        value = m_spinLCut.GetPos32();
        ar<<value;

        value = m_spinRCut.GetPos32();
        ar<<value;

        value = m_spinTCut.GetPos32();
        ar<<value;

        value = m_spinBCut.GetPos32();
        ar<<value;

        ar.Close();
        file.Close();
 }
}


void CDCalibSys::OnBnClickedBtnLoad()
{
 int xpos = 0;
```

```
int ypos = 0;
int lc = 0, rc = 0, tc = 0,bc = 0;
CFile file;
if(file.Open(L"./data/crc.txt",CFile::modeRead))
{
	CArchive ar(&file,CArchive::load);

	ar>>xpos;
	m_spinXpos.SetPos32(xpos);

	ar>>ypos;
	m_spinYpos.SetPos32(ypos);

	ar>>lc;
	m_spinLCut.SetPos32(lc);

	ar>>rc;
	m_spinRCut.SetPos32(rc);

	ar>>tc;
	m_spinTCut.SetPos32(tc);

	ar>>bc;
	m_spinBCut.SetPos32(bc);

	ar.Close();
	file.Close();

	if (g_pImageProc!=NULL)
	{
		g_pImageProc->SetXOffset(xpos);
		g_pImageProc->SetYOffset(ypos);

		g_pImageProc->SetLCut(lc);
		g_pImageProc->SetRCut(rc);
		g_pImageProc->SetTCut(tc);
		g_pImageProc->SetBCut(bc);
	}
}
}
```

从实例工程拷贝 calibPanel.h 和 CalibPanel.cpp 文件到当前工程并添加到工程。

在 ITSCamLocateDlg.cpp 中包含头文件“DCalibSys.h”，添加“沙盘标定”按钮响应函数和检查相机是否打开的函数，如下：

```
void CITSCamLocateDlg::OnBnClickedBtnCalib()
{
 if (!IsCameraOpen())
 {
        return;
 }

 CDCalibSys dlg;
 if (IDOK==dlg.DoModal())
 {
 }
}

BOOL CITSCamLocationDlg::IsCameraOpen()
{
 CString str1,str2;
 GetDlgItemText(IDC_BTN_PLAY1,str1);
 GetDlgItemText(IDC_BTN_PLAY2,str2);

 if ((str1==L"打开")||(str2==L"打开"))
 {
        MessageBox(L"请先打开两个相机！",L"提示");
        return FALSE;
 }
 else
 {
        return TRUE;
 }
}
```

2.6、 视频定位运行

在标定完成后，就可以开始视频定位运行。

依次添加“生成参考背景”，“矫正”，“拼接”，“定位”，“运行”这些按钮的响应函数，代码如下：

```
void CITSCamLocateDlg::OnBnClickedBtnGenbk()
{
 if (g_pImageProc==NULL)
 {
        return;
```

```
 }

 if (!IsCameraOpen())
 {
        return;
 }

 g_pImageProc->Cam1Correct();
 g_pImageProc->Cam2Correct();

 g_pImageProc->GenBkImage();

 m_ItsPanel.UpdateBackGround();
}


void CITSCamLocateDlg::OnBnClickedBtnCorrect()
{
 if (g_pImageProc==NULL)
 {
        return;
 }
 if (!IsCameraOpen())
 {
        return;
 }

 //畸变矫正
 g_pImageProc->Cam1Correct();
 g_pImageProc->Cam2Correct();
}


void CITSCamLocateDlg::OnBnClickedBtnStitch()
{
 if (g_pImageProc==NULL)
 {
        return;
 }

 if (!IsCameraOpen())
```

```
 {
        return;
 }

 g_pImageProc->StitchImage(false);
 g_pImageProc->PerspCorrect(false);
}


void CITSCamLocateDlg::OnBnClickedBtnLoc()
{
 if (g_pImageProc==NULL)
 {
        return;
 }

 if (!IsCameraOpen())
 {
        return;
 }

 g_pImageProc->ExtractCars();

 m_ItsPanel.Invalidate();
}


void CITSCamLocateDlg::OnBnClickedBtnRun()
{
 if (!IsCameraOpen())
 {
        return;
 }

 CString str;
 GetDlgItemText(IDC_BTN_RUN,str);
 if(str==L"运行")
 {
        m_bThreadRun = TRUE;
        if (m_pProcessThread==NULL)
        {
```

```
            m_pProcessThread = AfxBeginThread(ProcessThread,(LPVOID)this,THREAD_PRIORITY_NORMAL);
        }
        SetDlgItemText(IDC_BTN_RUN,L"停止");

        GetDlgItem(IDC_BTN_CCALIB)->EnableWindow(FALSE);
        GetDlgItem(IDC_BTN_GENBK)->EnableWindow(FALSE);
        GetDlgItem(IDC_BTN_CORRECT)->EnableWindow(FALSE);
        GetDlgItem(IDC_BTN_CALIB)->EnableWindow(FALSE);
        GetDlgItem(IDC_BTN_STITCH)->EnableWindow(FALSE);
        GetDlgItem(IDC_BTN_LOC)->EnableWindow(FALSE);

        GetDlgItem(IDC_BTN_PLAY1)->EnableWindow(FALSE);
        GetDlgItem(IDC_BTN_PLAY2)->EnableWindow(FALSE);

        GetDlgItem(IDC_BTN_SNAP1)->EnableWindow(FALSE);
        GetDlgItem(IDC_BTN_SNAP2)->EnableWindow(FALSE);
    }
    else
    {
        m_bThreadRun = FALSE;
        if (m_pProcessThread!=NULL)
        {
            WaitForSingleObject(m_pProcessThread->m_hThread,INFINITE);
            m_pProcessThread = NULL;
        }
        SetDlgItemText(IDC_BTN_RUN,L"运行");

        GetDlgItem(IDC_BTN_CCALIB)->EnableWindow(TRUE);
        GetDlgItem(IDC_BTN_GENBK)->EnableWindow(TRUE);
        GetDlgItem(IDC_BTN_CORRECT)->EnableWindow(TRUE);
        GetDlgItem(IDC_BTN_CALIB)->EnableWindow(TRUE);
        GetDlgItem(IDC_BTN_STITCH)->EnableWindow(TRUE);
        GetDlgItem(IDC_BTN_LOC)->EnableWindow(TRUE);

        GetDlgItem(IDC_BTN_PLAY1)->EnableWindow(TRUE);
        GetDlgItem(IDC_BTN_PLAY2)->EnableWindow(TRUE);
    }
}

UINT CITSCamLocateDlg::ProcessThread(LPVOID pParam)
{
```

```
 CITSCamLocateDlg * p = (CITSCamLocateDlg*)pParam;
 while (p->m_bThreadRun)
 {
        p->Process();
        Sleep(10);
 }

 return 0;
}

void CITSCamLocateDlg::Process()
{
 if (g_pImageProc==NULL)
 {
        return;
 }

 //TRACE("debug:Process1\r\n");
 //畸变矫正
 g_pImageProc->Cam1Correct();
 g_pImageProc->Cam2Correct();

 //TRACE("debug:Process2\r\n");
 //拼接
 g_pImageProc->StitchImage(false);

 //TRACE("debug:Process3\r\n");
 //透视矫正
 g_pImageProc->PerspCorrect(false);

 //TRACE("debug:Process4\r\n");
 //提取
 g_pImageProc->ExtractCars();

 //TRACE("debug:Process5\r\n");

 m_ItsPanel.Invalidate();
}
```

至此，整个工程就完成了，您可以在打开两个相机后，点击“运行”即可开始整个系统的运行。

3、 关键接口介绍

3.1、 相机的操作

```
DI_API DI_CAMERA_STATUS _stdcall  CameraInit(BYTE CamNum,
                                             DI_RESOLUTION uiResolution,
                                             HWND hWndDisplay,
                                             DI_SNAP_PROC pCallbackFunction,
                                             LPVOID lpContext
                                    );
```

CamNum:相机编号，1，2，3……

uiResolution:相机分辨率

hWndDisplay：显示图像的窗口；

pCallbackFunction：相机视频流处理回调函数；

lpContext：上下文对象指针；

函数的功能：按照制定的分辨率打开相机，在制定的窗口显示图像；

```
DI_API DI_CAMERA_STATUS _stdcall CameraSetGain(IN USHORT RGain, USHORT GGain, USHORT BGain);
```

RGain: R 值；

GGain: G 值；

BGain：B 值；

函数的功能：设置相机的白平衡。

更多的相机的函数接口，请参考《显微科技工业相机开发文档.pdf》。

3.2、 矫正接口

```
void Cam1Correct();
```

函数的功能：相机 1 图像矫正。由于广角镜头产生了鱼眼畸变，所以需要通过矫正来恢复原始图像。

```
void Cam2Correct();
```

函数的功能：相机 2 图像矫正。

3.3、 拼接接口

```
void StitchImage(bool bCalib);
```

bCalib：是否保存拼接后的图像，图像的保存路径是“./calibImage/bk_s.bmp”。

函数的功能：把左右两个相机的图像进行拼接，拼接后的图像在内存中，供后续处理使用。

```
void PerspCorrect(bool bCalib);
```

bCalib：是否保存拼接后的图像，图像的保存路径是“./calibImage/bk.bmp”。

函数的功能：利用四个标志点将图片拉伸为与现实沙盘比例相同的正方形图像。

3.4、 定位接口

```
void ExtractCars();
```

函数的功能：提取车辆的位置。

```
int GetCarSegSum();
```

函数的功能：获取提取到的车辆数量；

```
CarPos GetCarSeg(int i);
```

函数的功能：获取提取到的第 i 个车辆的位置；CarPos 为结构体，标识位置。

智能楼宇

第一章 门禁系统

1.1 门禁一体机安装手册

1.2 门禁用户操作手册

智能楼宇平台安装手册

1. 1.硬件连接

（1） 参照随机光盘中资料中的接线图示将门禁电源箱与门禁控制器相连接，然后依次将门禁磁力锁、门禁指纹机、门禁读卡器连接到门禁控制器上。

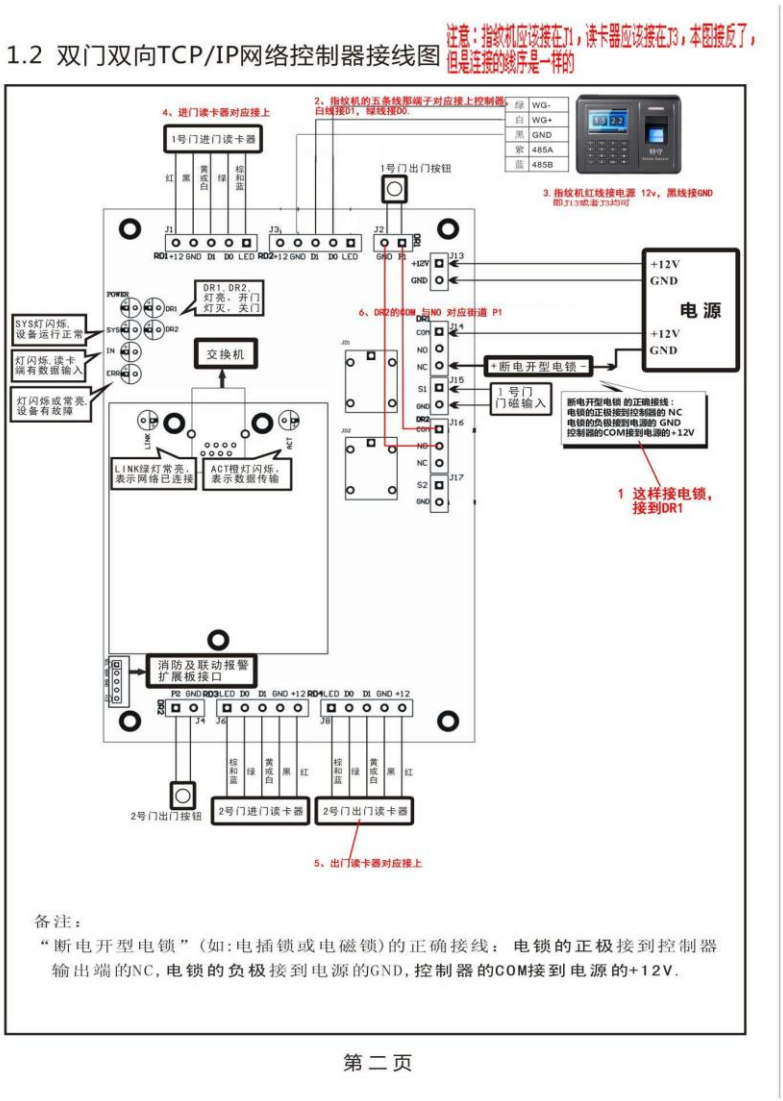


图 1 门禁控制器接线连接

（2） 然后将门禁一体机通过路由器与 pc 进行连接。

（3） 最后将门禁电源箱上电，控制器上电后个刷卡器自动启动，进行操作使用。

2.软件操作

2．1 RFID 门禁卡操作

2.1.1 设备管理

首先登陆上位机，进入到门禁管理系统。点击设备管理图标，弹出门禁设备配置窗口，进行设备配置。单击搜索按钮，进行搜索门禁控制器（如下图所示）。

搜索门禁控制器成功后，单击配置按钮，进行门禁设备配置。配置好门禁设备单击关闭按钮，退出设备配置窗口。

图 2 设备配置

2.1.2 人员注册

点击人员注册图标，进入到人员注册系统（如下图 3 所示）。点击新增按钮，然后在人员信息栏中添加人员信息（红框圈出部分）（如下图 4 所示）。

图 3 人员注册（一）

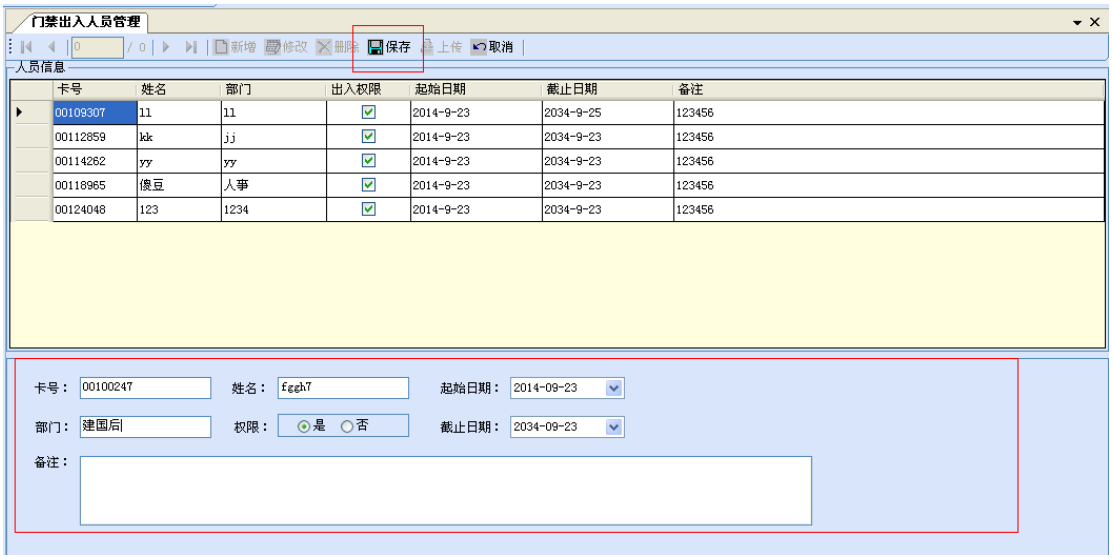


图 4 人员注册（二）

输入完人员信息之后，点击保存按钮，进行人员信息保存。弹出信息保存成功窗口后，点击上传按钮（如下图 5 所示），将人员信息上传到数据库。注意：如若仅保存人员信息没有上传，则该人员卡不具有门禁权限。至此人员注册全部完成。

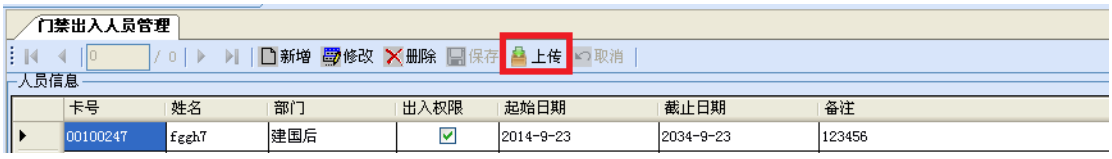


图 5 人员注册（三）

2.1.3 门禁控制

点击门禁控制图标，进入门禁控制系统（如下图 6 所示）。

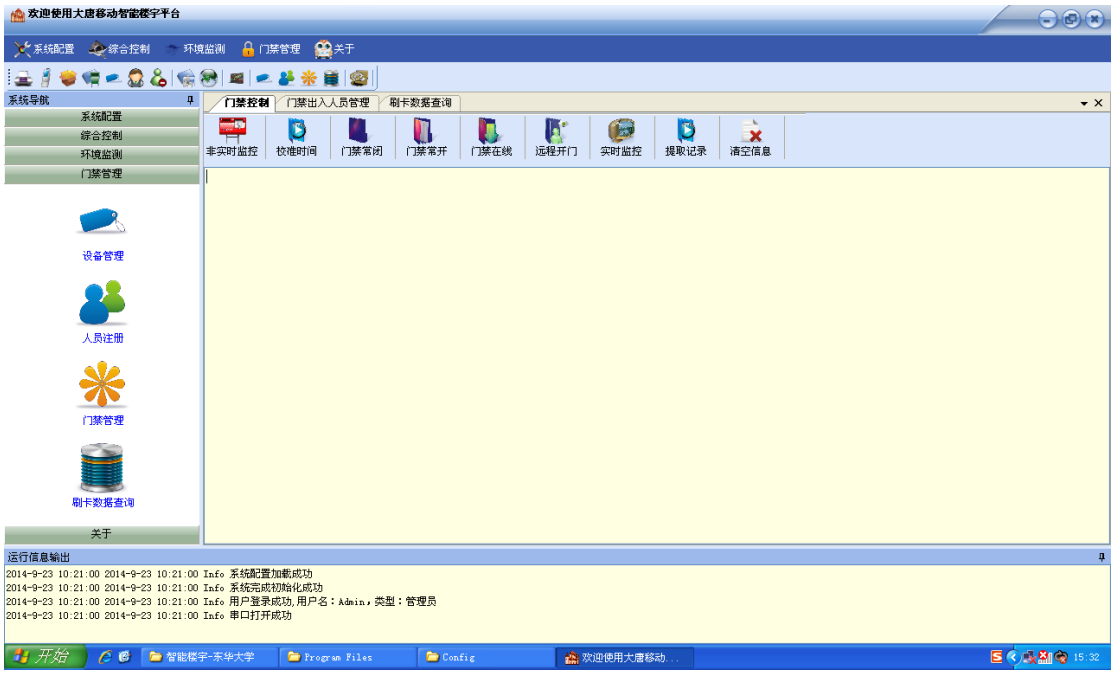


图 6 门禁控制

单击非实时监控图标，查看门磁控制器信息状态（如下图所示）。

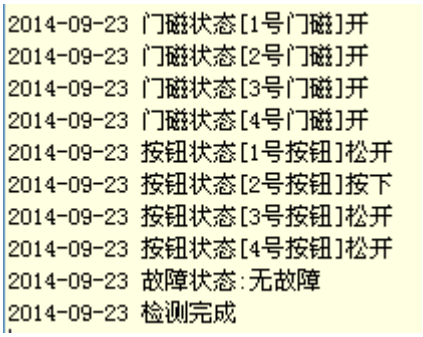


图 7 非实时监控

单击校准实际按钮，进行实际校准（如下 8 图所示）。

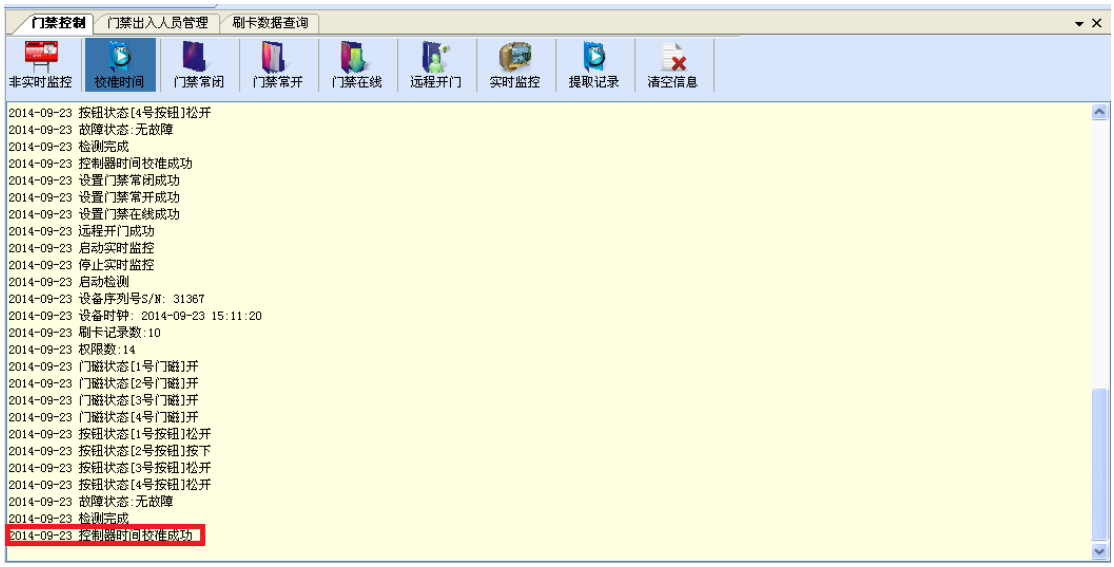


图 8 实际校准

单击门磁常闭按钮，进行控制磁力锁常闭。同时日志栏中输出门磁常闭设置成功（如下图 9 所示）。此时磁力锁长久锁在一起，无法打开。

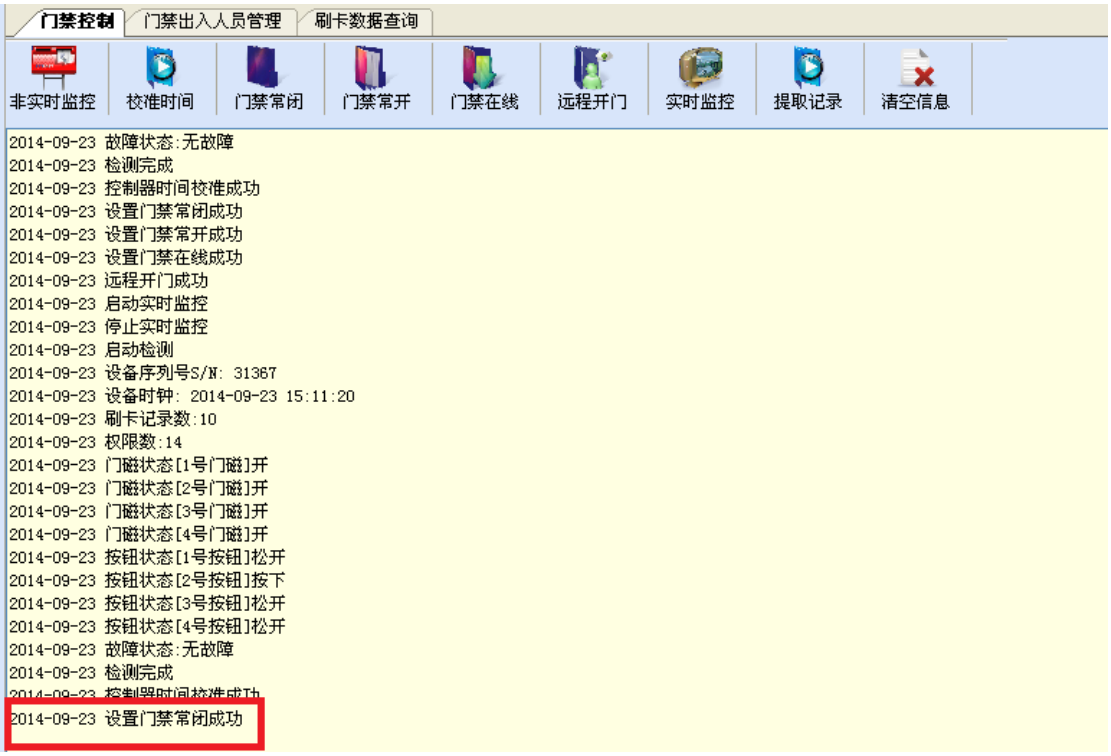


图 9 门磁常闭

单击门磁长开按钮，控制磁力锁保持常开状态。同时日志栏中输出门磁常开设置成功（如下图 10 所示）。此时磁力锁长久打开状态。

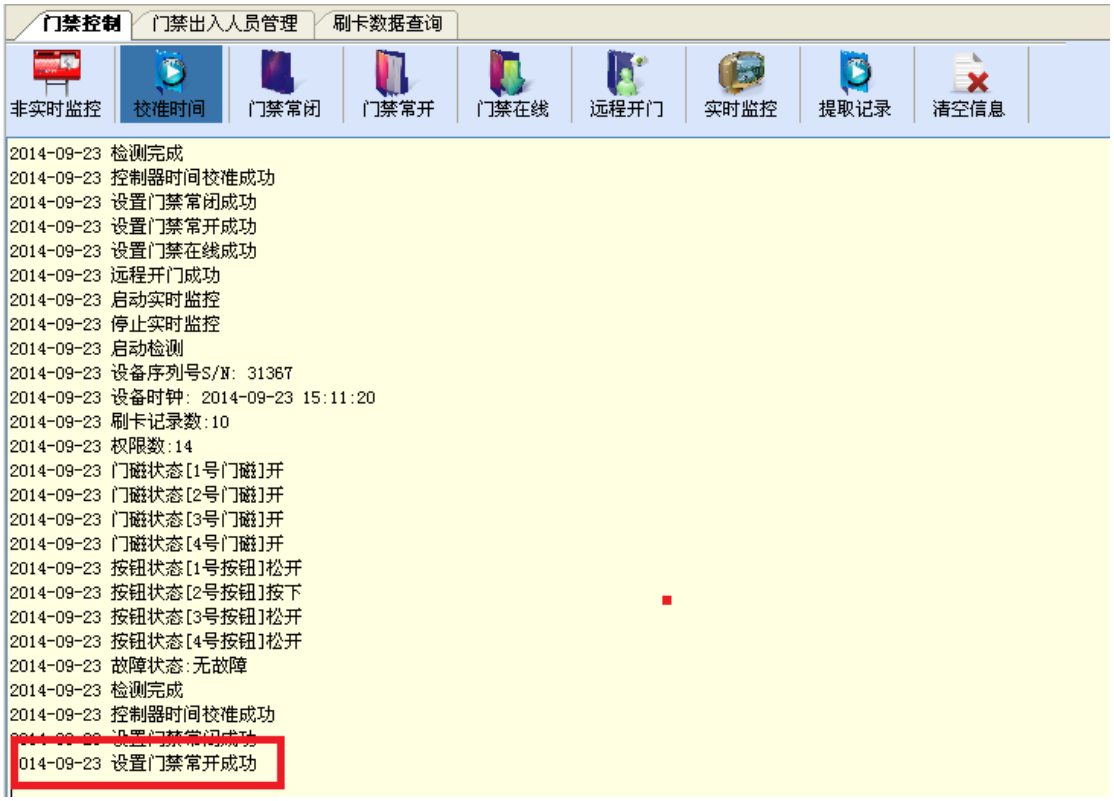


图 10 门磁常开

单击门磁在线按钮，设置门磁在线控制状态。该状态下已注册过的人员信息卡可以通过刷卡将磁力锁打开，同时上位机中日志栏输出设置门磁在线成功。

图 11 门磁在线

单击远程开门按钮，实现远程打开磁力锁的功能，同时上位机中日志栏输出设置门磁远程成功。

图 12 远程开门

单击实时监控按钮，将已注册过的人员信息卡在读卡器上刷一下，获取刷卡人员的全部注册信息（如下图所示）。获取全部信息之后，再次单击实时监控按钮，退出该监控系统，上位机中日志栏输出退出实时监控。

注意：人员注册信息时，如果仅保存人员信息并没有上传，则没有门禁权限，无法打开磁力锁（若下图所示）。

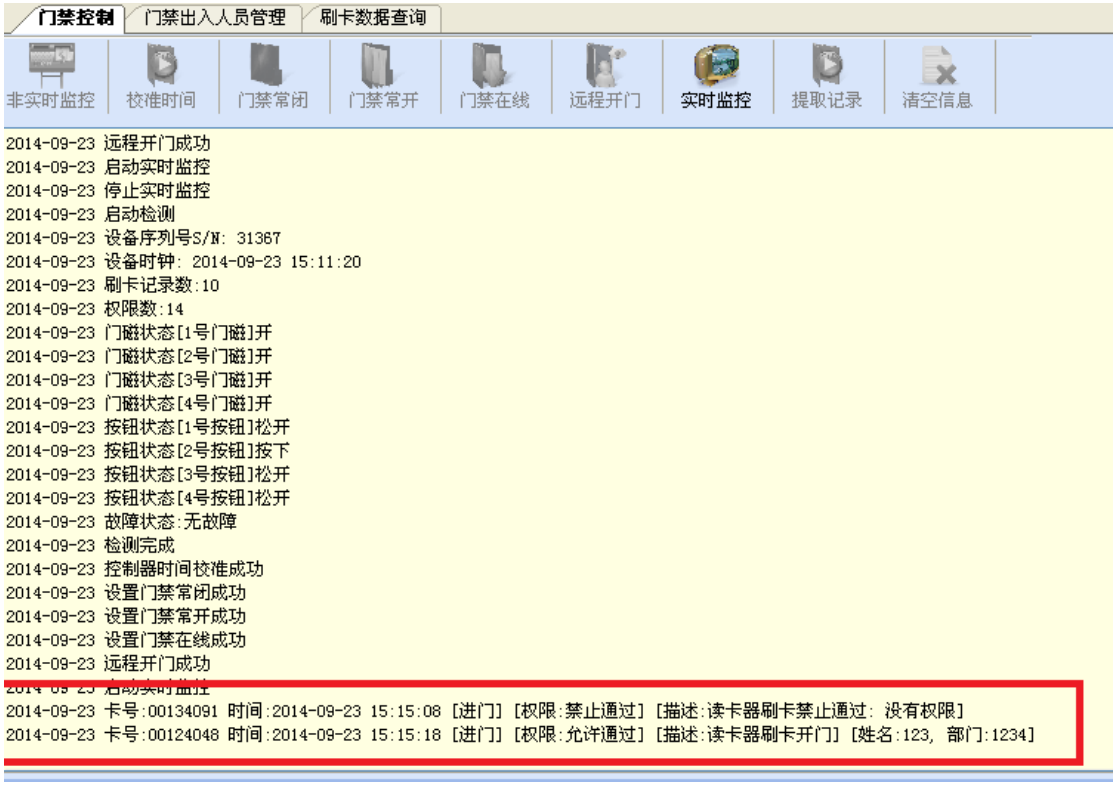


图 13 实时监控

单击提取信息按钮，获取近期所有刷卡记录（如下图 14 所示）。

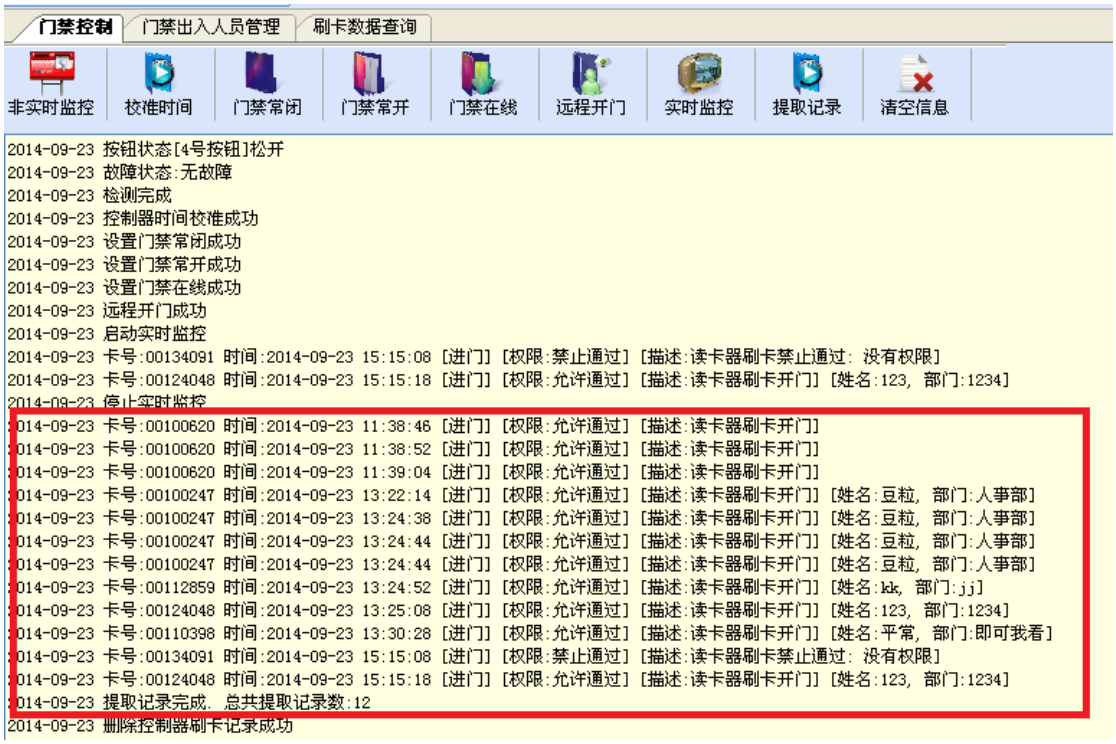


图 14 实时监控

单击清空信息按钮，上位机日志栏中所有信息全部被清空,如下图 15 所示。

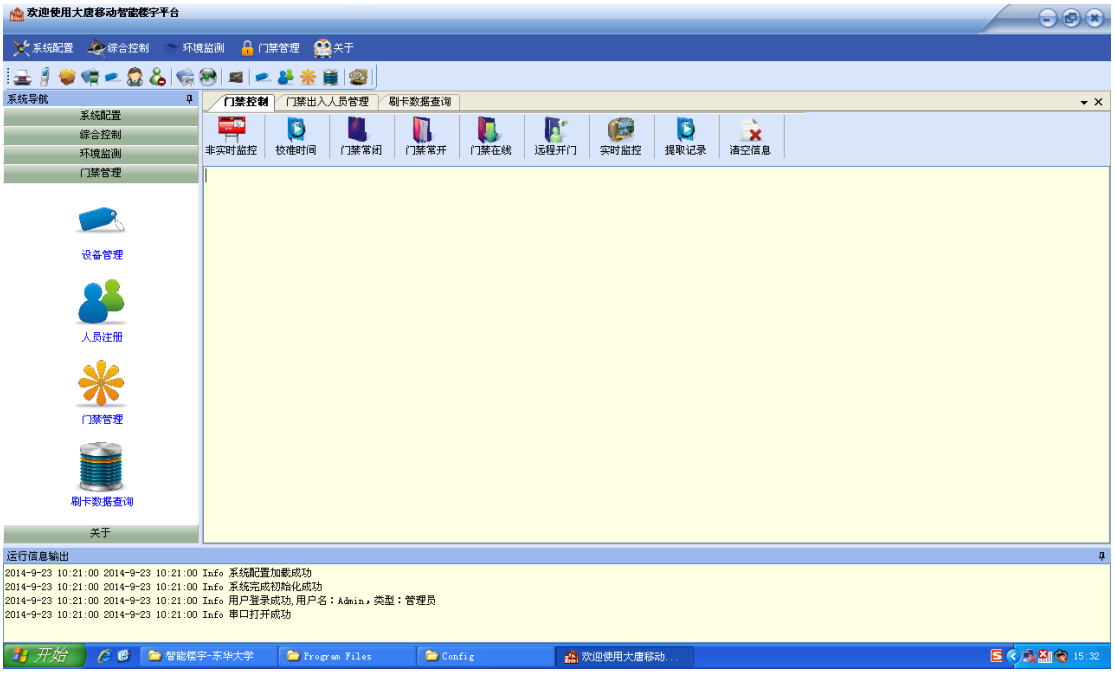


图 15 清空信息

2.1.4 刷卡数据查询

点击数据查询图标，进入数据查询系统（如下图所示）。

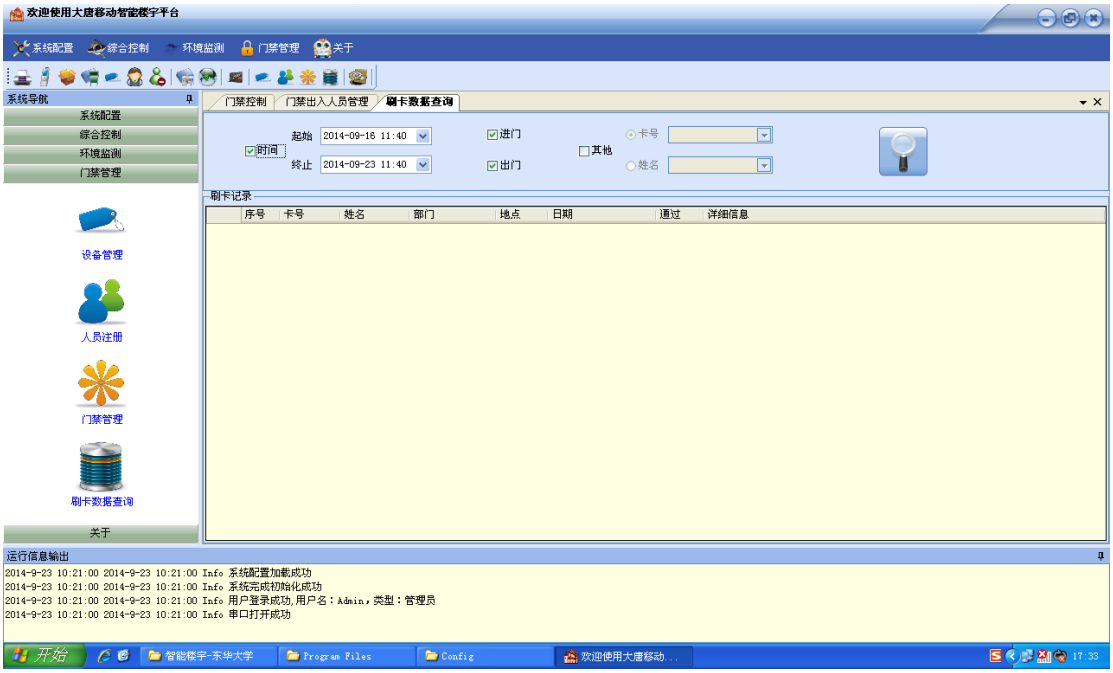


图 16 数据查询

点击时间按钮，进行设置所要查询的信息数据的时间范围。设置好时间范围之后，点击右侧放大镜查询图标，开始进行数据查询。注：如不进行时间设置则查询到的是所有时间段数据信息。

图 17 设置查询数据时间范围

点击姓名按钮，选择所要查询人员的姓名，开始查询该人员的全部出入情况的数据信息。设置好之后，点击右侧放大镜查询图标，开始进行数据查询。

序号	卡号	姓名	部门	地点	日期	通过	详细信息
1	00109307	11	11	进门	2014-09-23 11:29:30	☑	读卡器刷卡开门
2	00109307	11	11	进门	2014-09-23 11:06:50	☑	读卡器刷卡开门
3	00109307	11	11	进门	2014-09-23 11:06:36	☑	读卡器刷卡开门
4	00109307	11	11	进门	2014-09-23 11:06:22	☑	读卡器刷卡开门
5	00109307	11	11	进门	2014-09-23 11:06:12	☑	读卡器刷卡开门
6	00109307	11	11	进门	2014-09-23 11:05:58	☑	读卡器刷卡开门
7	00109307	11	11	进门	2014-09-23 11:05:52	☑	读卡器刷卡开门
8	00109307	11	11	进门	2014-09-23 11:05:46	☑	读卡器刷卡开门
9	00109307	11	11	进门	2014-09-23 11:05:40	☑	读卡器刷卡开门
10	00109307	11	11	进门	2014-09-23 11:05:38	☑	读卡器刷卡开门
11	00109307	11	11	进门	2014-09-23 11:05:30	☑	读卡器刷卡开门
12	00109307	11	11	进门	2014-09-23 11:05:08	☑	读卡器刷卡开门
13	00109307	11	11	进门	2014-09-23 11:05:06	☑	读卡器刷卡开门
14	00109307	11	11	进门	2014-09-23 11:05:00	☑	读卡器刷卡开门
15	00109307	11	11	进门	2014-09-23 11:00:30	☑	读卡器刷卡开门
16	00109307	11	11	出门	2014-09-23 11:00:24	☑	读卡器刷卡开门

图 18 选择姓名数据查询

点击卡号按钮，选择所要查询信息的卡号，开始查询该卡的全部出入情况的数据信息。设置好之后，点击右侧放大镜查询图标，开始进行数据查询。

图 19 卡号刷卡数据查询

点击出门/进门按钮，开始查询的全部出门/进门情况的数据信息。设置好之后，点击右侧放大镜查询图标，开始进行数据查询。

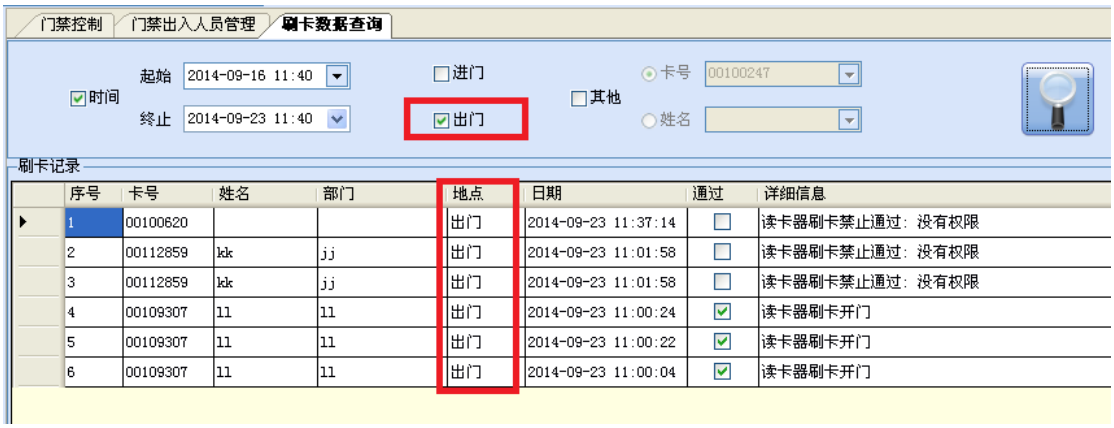


图 20 出/进门刷卡数据查询

点击其他按钮，开始查询的全部的刷卡数据信息。点击右侧放大镜查询图标，开始进行数据查询。

序号	卡号	姓名	部门	地点	日期	通过	详细信息
43	00114262	yy	yy	进门	2014-09-23 11:11:20	☑	读卡器刷卡开门
44	00100247	shadou	市场部	进门	2014-09-23 11:11:18	☐	读卡器刷卡禁止通过：没有权限
45	00100247	shadou	市场部	进门	2014-09-23 11:11:16	☐	读卡器刷卡禁止通过：没有权限
46	00100247	shadou	市场部	进门	2014-09-23 11:11:12	☐	读卡器刷卡禁止通过：没有权限
47	00100247	shadou	市场部	进门	2014-09-23 11:09:36	☐	读卡器刷卡禁止通过：没有权限
48	00100247	shadou	市场部	进门	2014-09-23 11:09:30	☐	读卡器刷卡禁止通过：没有权限
49	00100247	shadou	市场部	进门	2014-09-23 11:09:28	☐	读卡器刷卡禁止通过：没有权限
50	00100247	shadou	市场部	进门	2014-09-23 11:09:24	☐	读卡器刷卡禁止通过：没有权限
51	00109307	11	11	进门	2014-09-23 11:06:50	☑	读卡器刷卡开门
52	00109307	11	11	进门	2014-09-23 11:06:36	☑	读卡器刷卡开门
53	00109307	11	11	进门	2014-09-23 11:06:22	☑	读卡器刷卡开门
54	00109307	11	11	进门	2014-09-23 11:06:12	☑	读卡器刷卡开门
55	00109307	11	11	进门	2014-09-23 11:05:58	☑	读卡器刷卡开门

图 21 全部刷卡数据查询

2.2 指纹识别操作

1. 上电使用前，请确认指纹识别器与控制器正确连接，参考随机光盘中资料中的接线图示。

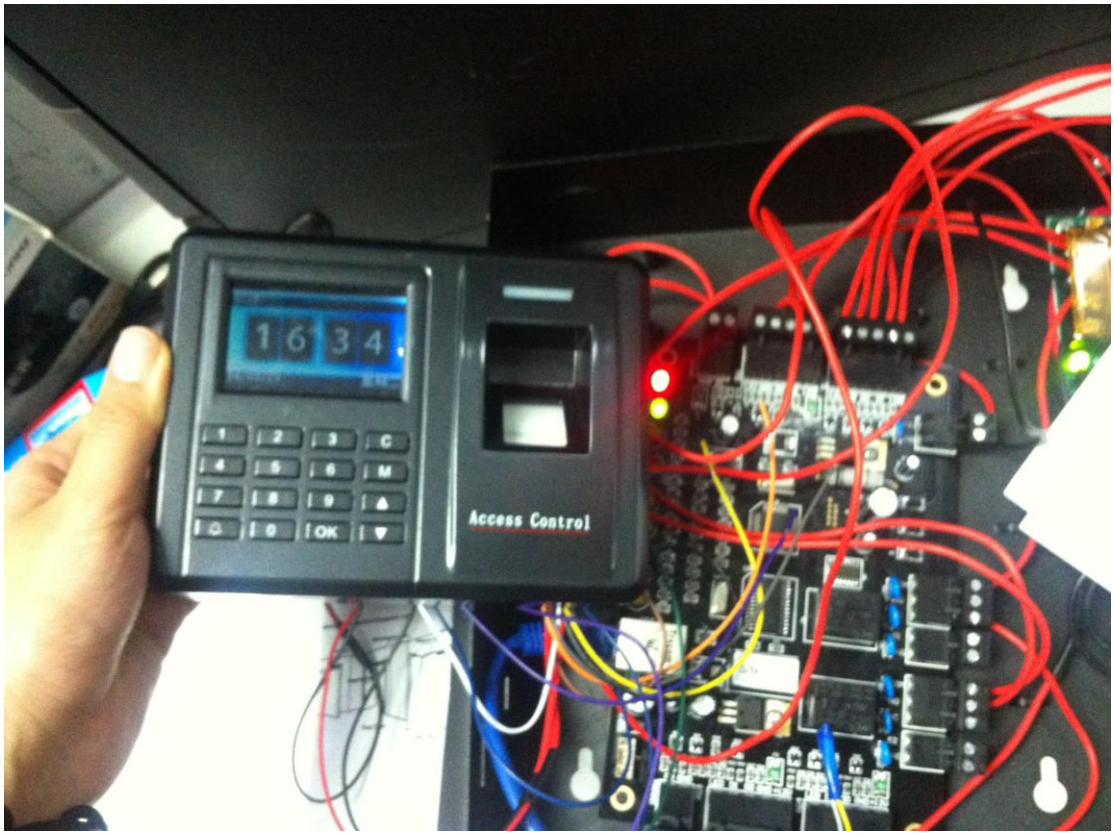


图 22 指纹识别器接线（一）

1. 控制器上电后个刷卡器自动启动，等待指纹刷卡器通过自检后，点击“M”键，进入菜单选择界面（如下图 24 所示）。

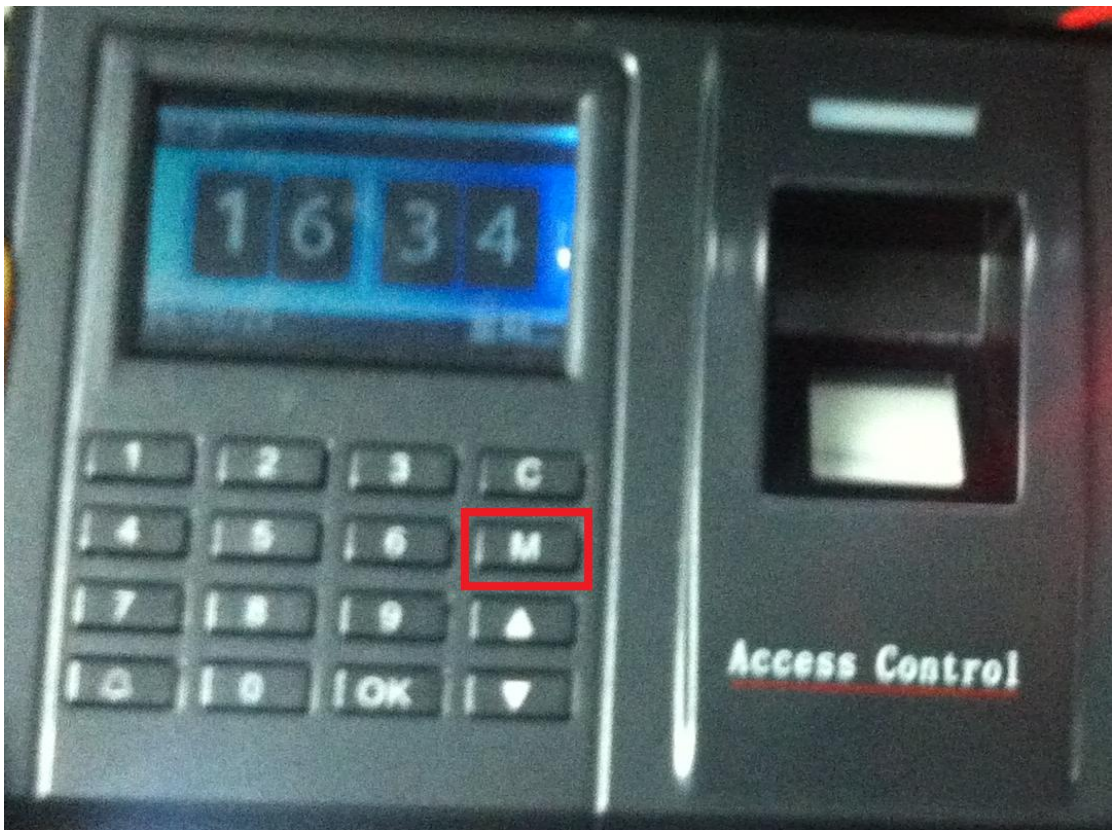


图 23 指纹识别器接线（二）

图 24 指纹识别器接线（三）

2. 选择“用户管理”，点击“OK”键进入，选择“新用户”->“新注册”->“卡注册”进入卡片注册画面。

图 25 卡注册

图 26 注册卡号

3. 在号码输入框框中输入卡号（如下图中的前半部分：99627），输入完成后点击“OK”键，提示“请拍卡”。

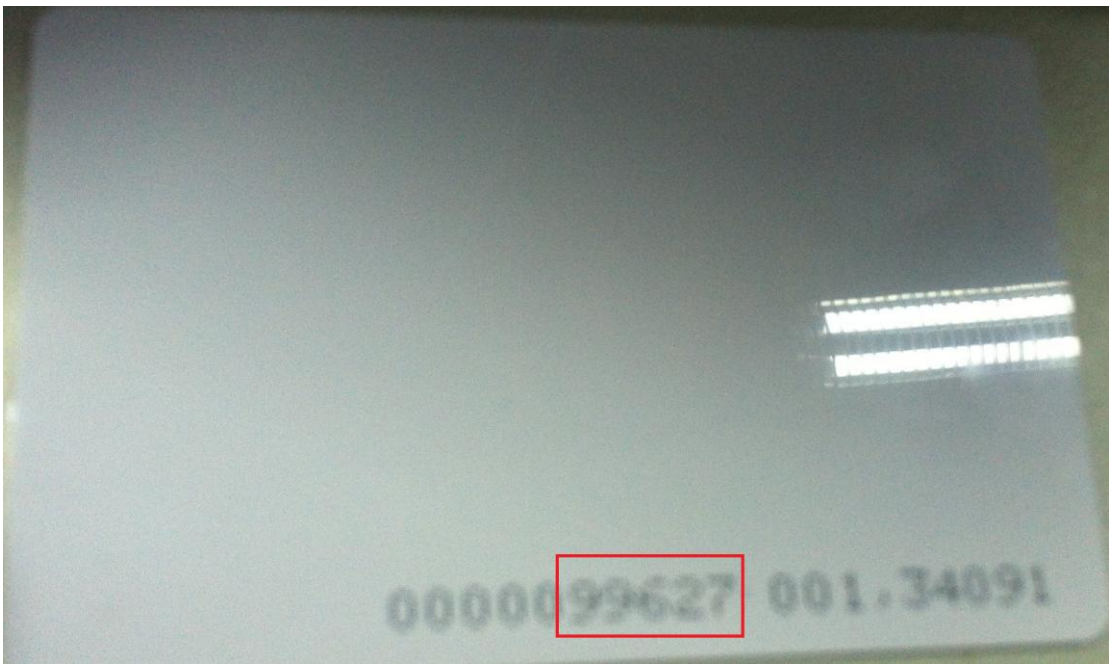


图 27RFID 卡号

4. 当前需要注册的卡片靠近指纹刷卡器，听到提示后，显示卡号“99627”，点击“OK”键确认完成信息。

图 28 完成注册卡

5. 上位机执行用户权限注册（卡片与用户绑定，并给与开门权限操作）后，刷卡后在上位机显示卡号（如上图中的后半部分：34091）。

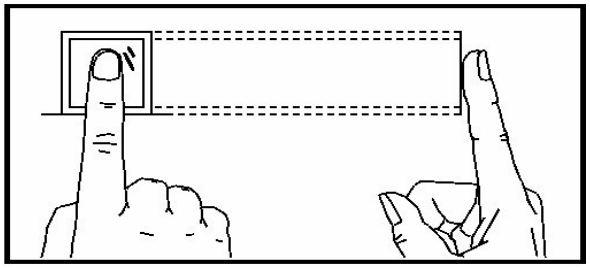
1.2　门禁用户操作手册

一、 使用需知

1.1　按指纹的方式

推荐手指：食指、中指或无名指；尽可能的避免使用大拇指和小拇指（因为它们按压指纹窗口时通常很笨拙）

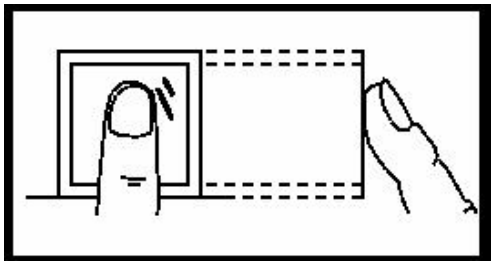
正确的按压方式



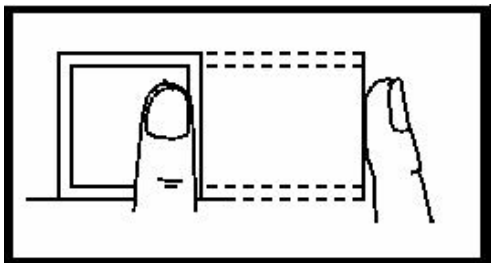
手指压平于指纹采集窗口上
手指纹心尽量对正窗口中心

几种错误的按压方式：

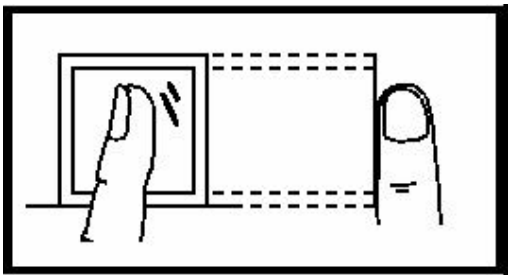
垂直未完全接触上



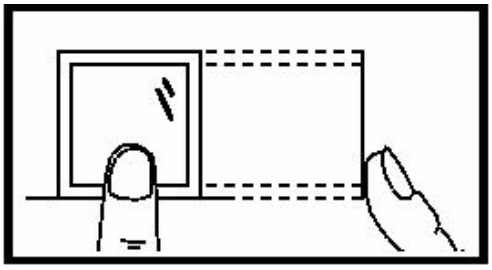
太偏



倾斜



太靠下



注：请采用正确的指纹按压方式进行登记和比对，本公司不承担由于用户操作不当而导致的识别性能降低等后果，本公司对此保留最终的解释权和修改权。

1.2.安装注意事项：

* 避免将设备安装在强光直照的地方。强光对指纹采集有着明显的影响，可能会导致指纹验证无法通过。
* 建议客户使用设备的温度范围在0℃－60℃，在 此范围内设备可以达到最佳使用效果。长期在室外使用时，过高或过低的温度易使设备工作受到影响，反应可能会变慢，，建议使用遮阳伞和散热设备，冬天采用保暖设施对设备进行保护。
* 安装设备时，请先连接好其他连线后再连接电源线，如果发现设备不能正常运行，请先断掉电源总开关后再进行必要的检查，我们提醒您：一切带电操作都有可能损怀机器，这将不在我们的正常保修范围内。

二、 基本概念

2 基本概念的定义和描述，包括：

- ◆ 用户的注册
- ◆ 用户的验证
- ◆ 用户的 ID 号码
- ◆ 权限级别

设备的最重要的两个功能是用户的登记和验证。

2.1 用户的注册

同一个 ID 号码可以登记 2 枚不同的指纹，（这样用户可以拥有多种验证的选择。

从理论上来说，每只手上的每个手指都应当被登记。这样用户可以使用登记的任一手指验证，也避免因忘记登记了哪一个手指而导致识别不方便的情况。一般情况下，推荐至少登记两枚手指，如：左右食指，这样，当用户的一个手指受到伤害时，可以使用备用的手指进行正常的比对。

2.2 用户的验证

当用户在设备的采集器上按压指纹，或者输入一个 ID 号码后，再输入密码

或按压指纹，通过储存的模板与当前扫描到的指纹进行比对。这个指纹模板被用来确认用户的身份，在设备上登记了的用户可以在该设备上使用指纹考勤，这个工作流程就是进行验证。系统在验证流程结束后将给出成功或是失败的信息，并将成功比对的记录储存到设备中。

2.3 用户的 ID 号码

在开始登记时，用户被分配一个未使用过的 ID 号码。当用户开始验证身份时，这个号码被用来关联指纹模板或密码。

2.4 权限级别

本系列设备有三个权限级别：

用户：是指那些因为某种目的而需要验证身份的人，诸如通过设备去开门，或者记录他们的出入记录。

管理员：除了不能进行高级设置外，可以进行其它所有操作。

超级管理员：是能访问所有系统功能的使用者，能更改系统的所有设置。

三、 快速操作说明

3.1 按键布局说明

1	2	3	C
4	5	6	M
7	8	9	▲
	0	OK	▼

注：部分机型可能布局有差异。

按键功能说明：

0…9　　输入数值
C　　ESC 键的缩写，在操作菜单时用作返回或取消
M　　即 MENU　进入菜单管理
▲　　上翻键
▼　　下翻键

有线门铃按键

3.2 主菜单

在初始界面，按菜单键“M”可以打开主菜单，如下图示：

菜单
& 用户管理
& 通讯设置
& 门禁功能
& 系统设置
& 系统信息
退出-ESC　　设置-OK

用户管理：对用户的基本信息进行注册、删除操作，并设置管理权限；
通讯设置：设置设备与 PC 通讯的相关参数，包括 U 盘下载、波特率、IP 地址、网关、子网掩码等。
门禁功能：设置设备门禁相关功能，如：时段、用户门禁定义、开门组合、报警等信息
系统设置：设置本机相关信息，如：设备机号、记录、时间、响铃等信息；
系统信息：查看本机相关信息，如：注册容量、记录详情、产品参数信息等

四、人员管理

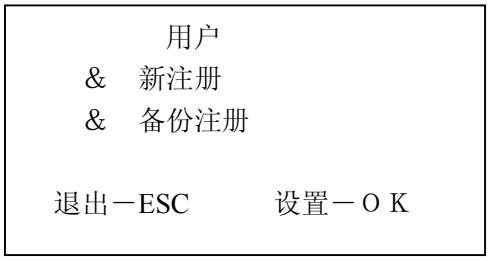
4.1 用户注册

在“用户管理”中包含了“新用户”“管理员”“超级用户”“删除用户”“U 盘上传”五个子项目；如下图：

用户管理
& 新用户
& 管理员
& 超级用户
& 删除用户
& U 盘上传
退出－ESC　　设置－OK

备注：在管理用户菜单里，主要是对人员的管理。下面说明注册用户流程。

在“用户”中包含“新注册”、“备份注册”二个子项目，如下图所示：

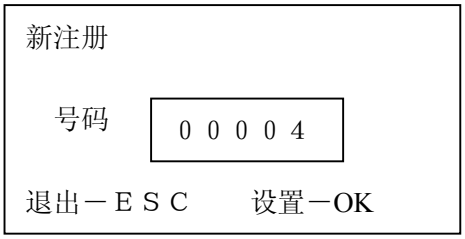


新注册：主要是注册指纹、密码、卡等信息。
备份注册：主要是对已注册的信息进行备份注册。

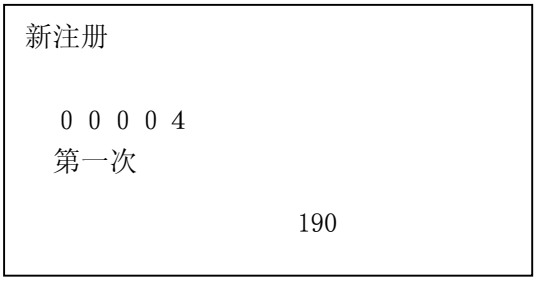
4.2 注册指纹用户流程

步骤一：在“新注册”菜单下选择用“▲”与“▼”键选择“指纹注册”，按 OK 键确认。如下图：

步骤二：按 OK 键确认，系统进入下一步，需要输入 ID 号。如下图：



步骤三：在输入完 ID 号，按 OK 键确认，进入指纹注册界面如下图：按提示按压指纹采集器两次，即可完成注册



如果登记指纹成功，设备提示“是否继续”，如下图：

如果选择 OK，则进入下一用户，如果选择 ESC 则返回至上一级，如下图：

注：关于指纹 ID 号说明，一个指纹 ID 号可以注册两枚指纹，在注册的时候注意。

4.3 卡注册流程

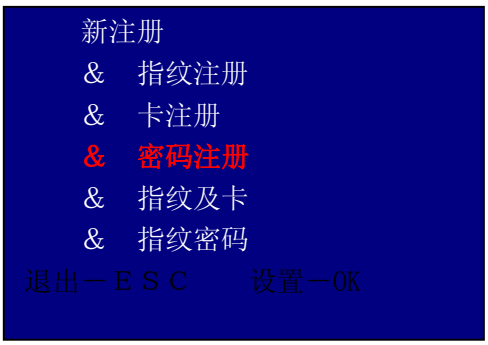
步骤一：在“新注册”菜单下选择“卡注册”，按 OK 键确认。如下图：

步骤二：按 OK 键确认，系统进入下一步，需要输入 ID 号。如下图：

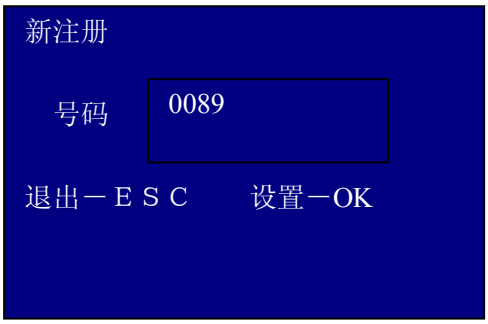
步骤三：在输入完 ID 号，按 OK 键确认，进入卡注册界面如下图：按提示刷卡，

4.4 密码注册流程

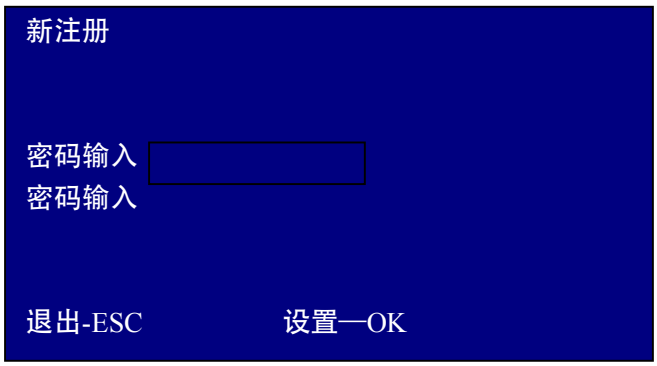
步骤一：在“新注册”菜单下选择“密码注册”，按 OK 键确认。如下图：



步骤二：按 OK 键确认，系统进入下一步，需要输入 ID 号。如下图：



步骤三：在输入完 ID 号，按 OK 键确认，进入密码注册界面，如下图：



4.5 备份注册

“备份注册”的操作方法和“新注册”的操作方法基本相同。

注册用户界面选择备份注册，选择要注册的类型，输入需要注册备份的用户注册号按OK键进行注册（备份注册仅能给已注册用户进行注册）

4.6 管理员注册

管理员注册是为了对进行管理操作的管理者的授权。方法同用户注册，请参照用户注册。

区别：管理者注册菜单项下注册的用户是管理员，用户注册菜单项下注册的用户是普通用户。同理，备份注册的权限依照注册时所选注册菜单。

本机注册的管理员也可视为普通用户做日常使用。同时，可进行管理操作。管理者所作的操作记录会记录在设备内。

4.7 超级用户注册

超级用户注册是为了对进行管理操作的次级管理者的授权，仅拥有部分管理权限。方法同用户注册，请参照用户注册。

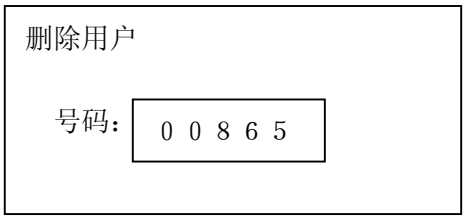
区别：超级用户注册菜单项下注册的用户是超级用户，用户注册菜单项下注册的用户是普通用户。同理，备份注册的权限依照注册时所选注册菜单。

本机注册的超级用户也可视为普通用户做日常使用。同时，可进行部分管理操作。管理者所作的操作记录会记录在设备内。

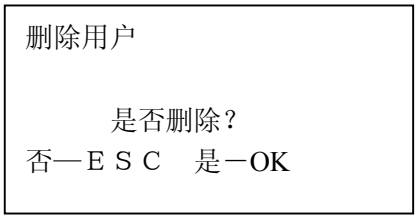
五、删除用户流程：

步骤一：在【用户管理】项中选择【删除用户】，按“OK”键进入【删除用户】菜单。

步骤二：输入需要删除的ID号，如下图：　　点击“OK”，



步骤三：使用OK键与ESC键确认是否删除，如下图；



完毕之后，按ESC退出。

六、通讯设置

选择【菜单】→【通讯设置】，进入数据下载菜单，此菜单包含“U 盘下载”“RS485 下载”“以太网下载”三项子菜单，如下图所示：

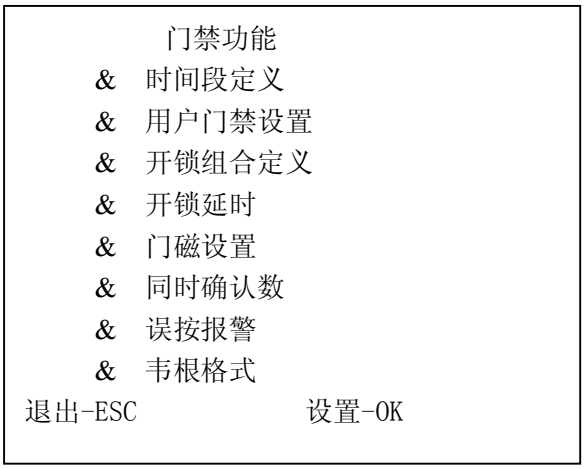
【U 盘下载】 下载开门记录和下载注册数据两项功能，分别实现设备内注册数据和记录的下载功能。该功能视具体机型而定。

【RS485 下载】菜单是用来设置设备与电脑通讯速度的，进入该项后通过选择键可选择通讯速度。

【以太网下载】菜单用来设置设备以太网各项参数，分别有 IP 地址，子网掩码、网关，用选择键进入， 按网络实际环境输入即可。

七、门禁功能

选择【菜单】→【门禁功能】，进入门禁功能菜单，此菜单包含“时间段定义”“用户门禁设置”“开锁组合定义” “开锁延时”“门磁设置 ”“同时确认数”“ 误按报警”“韦根格式”八项子菜单，如下图：



下面将每个子菜单作进一步的说明：

7.1时间段定义

时间段里分天时间段和周时间段两种！

天时段

根据用户的出入规律、制度，将每天的通行时间设置成相应时间段，总共可设置8种天时段，供周时段选择。例如一种早上6点到8点可开门，晚上17点到19点可开门的设置，如下图【天时段1】的设置情况，若需要设置全天候开门，则如下图【天时段2】的设置情况：

例如：【天时段 1】设置为

1	06:00	08:00
2	17:00	19:00
3	00:00	00:00
4	00:00	00:00
5	00:00	00:00

例如：【天时段 2】设置为

1	00:00	23:59

2	00:00	00:00
3	00:00	00:00
4	00:00	00:00
5	00:00	00:00

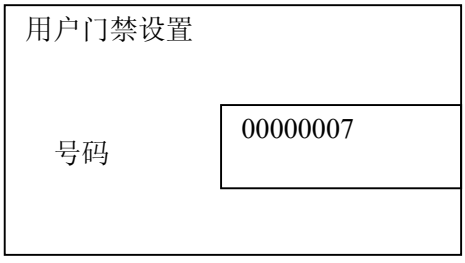
周时段

根据用户的出入规律、制度，将每个星期的通行时间设置成相应天时间段。例如：星期到星期五适用上述天时间段规则，星期六星期天全天可开门。如下图周时段 1的设置情况：

星期一	1
星期一	1
星期二	1
星期三	1
星期四	1
星期五	1
星期六	2
星期日	2

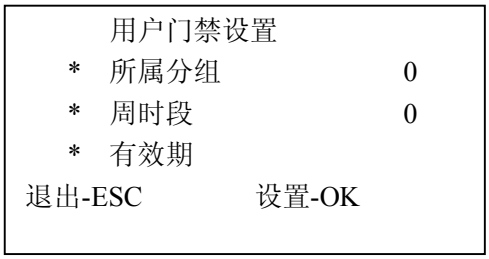
备注：指纹设备的【周时间段 0】默认为全天候开门。其他所有时间段都是全天候不开门的时段。在不更改天时段定义情况下，其他时段也为默认全天候开门。

7.2 用户门禁设置



步骤一：输入需要设置的用户ID号后，按OK键确认后，进入下一步：

步骤二： 设置相关信息，如下图：



所属分组：设备一共可将用户划分为10个用户组

周时段：可定义用户适用于哪种开门规律。

有效期：可定义用户开门的权限从什么时候起到什么时候截止

注：

每个用户都有一个所属分组，相应时间段，若没设置，则默认为0，全天候可开门。

当某用户验证身份（指纹或密码）的时候，指纹设备判断员工的时间段，如果满足条件，则开门；如果不满足条件则不开门。

例如：某员工登记号为：00000001，其所属分组为1，时间段分别设置为1，如下图所示。

用户门禁设置	
1. 所属分组	1
2. 周时段	1
3. 有效期	1
退出-ESC	设置-OK

如登记号为00000001的员工按指纹，设备判断该员工所属分组是否有开门权限（在“见开锁组合定义”中设置），如果有，则判断现在时间是否在此员工所允许的时间段内，如果在，则开门，如果不在，则不开门。

7.3 开锁组合定义

组合1	0
组合2	0
组合3	0
组合4	0
组合5	0

开锁组合定义可设置开锁组合，例如设置组合为12，则上节用户门禁定义中的定义为1组和2组的两组人员中各一个打卡可开门。

举例1：同一组的单人开锁设置

组合1　　1

组合2　　0

…

组合5　　0

如上设置，只有组合1做了设置，包含了【组1】，也就是只说所属分组为【组1】的员工拥有开门权限。

举例2：同一组的多人组合开锁设置

组合1　　111

组合2　　否

…

组合10　　否

如上设置，只有“组合1”设置“111”，表示在有效时间段内，所属分组为【组1】的任意三名人员同时验证（验证顺序不分先后）通过，才能够开锁。

举例3：不同组的单人组合开锁设置

组合1　　12

组合2　　否

…

组合10　　否

如上设置，只有“组合1”设置了内容，包含了【组1】、【组2】，表示所属分组为【组1】的员工和所属分组为【组2】的员工共同验证通过（验证顺序不分先后）才可以开锁。

所属分组为【组1】的员工和所属分组为【组2】的员工都不具有单独开锁的权限。

【注意】

- 如果【开锁组合定义】项目没有任何设置，则所有员工都不具有开锁权限。
- 指纹设备的默认设置是【开锁组合1】设置了包含【组1】。也就是说所有分组为【组1】的员工具有单独的开门权限。

7.4开锁延时

开锁延时设置的是锁控制继电器启动之后，恢复到常规状态的时间间隔。值范围为：1-255，单位为：秒。默认值为：5秒。

7.5门磁设置

A．门磁类型

设置了输入门磁的类型，默认状态下为：否（不开启门磁功能）。

B．门磁延时

锁控制继电器恢复到正常状态之后，经过多长时间未检测到门关闭既开始报警。值范围为：1-255，单位为分钟。

举例：当门磁类型设置为闭型，门磁延时设置为1时，则常规状态下，门磁为闭合状态，开门时断开。当开门状态持续时间超过1分钟，则设备报警。

注：非法开门报警

若没有验证身份而直接将门打开，即被视为：非法开门。非法开门将会立即启动警报。

7.6 同时确认数

同时确认数：可定义开门所需人数。例如设置为2则需要2个不同的用户打卡才可以开门。

7.7 误按报警

误按报警：误按报警可设置输入错误之后多少次报警。例如设置为5，则输入密码或输入指纹或卡无法辨认或辨认失败达到5次时机器报警。

7.8 韦根格式

可定义韦根端口输出模式，默认输出模式为 26，可更改为 26 或 34.

八、 系统设置

在主菜单界面下，选择【系统设置】“系统设置下有“本机设置” “记录设置”“时间设置”“定时响铃”共四个子菜单。如下图：

8.1 本机设置：用来设置设备的参数，下面列表解释了每一项设置的内容、范围、默认值。

设置内容	说 明	范 围	默认值
设备号	软件中识别设备的唯一号码	1-255	1
管理员总人数	设置设备管理者的总人数	1-10	1 0
显示语言	菜单显示的语言	多种	中文
验证方式	多种识别试	多种	指纹、密码、卡
上传界面	修改设备界面	空	否
恢复出厂值	将设备参数恢复到出厂状态	空	空
清除管理员权限	清除设备上的所有管理员权限	空	空

确认方式：本设备允许多种识别模式组合验证：比如卡+指纹，需要用户先验证感应卡片，再输入指纹才可以通过验证。

F/P/C	指纹、感应卡、密码都任何一种验证模式可以验证通过
C	感应卡可以验证通过，无其他验证模式
F+C	必须先刷卡，再按指纹
P+P	必须先输入登记号、密码，再按指纹

8.2 记录设置

8.2.1 管理记录警告

设备管理员每次进行一次操作（比如增加指纹，删除指纹等）都会保存成一条管理记录，设备最多可以保存1000 条管理记录。

当未被采集的管理记录差一定的数量满1000条的时候，设备就会提醒“超出管理记录警告”。

【管理记录警告】就是设置差多少条记录满1000条的时候开始报警的值。

比如【管理记录警告】设置为100，则未被采集的管理记录到900条的时候就开始提示“超出管理记录警告！”

8.2.2 出入记录警告

操作步骤与管理记录警告设置操作步骤完全相同。当未被采集的设备记录超过一定的值满 100000 条的时候，开始提示“超出记录！”。

例如：若此值设为 1500，当出入记录达到 98500 条时，从第 98501 条记录开始，用户验证时将显示“超出记录”，同时提示音为“谢谢！”；当设备记录超过 99999 条时，显示“超出记录”界面后，并且记录不予存储，即记录无效。

此值用户可根据需要设置（1-1500 之间的任意整数），设备将根据设定值告知“超出记录”。

8.2.3 重复确认时间

检测用户是否在相应的时间内反复验证，建议用户将此值设为 5(分钟)。

若用户在设定时间内反复验证，则“设备”告知用户已签到。这时不重复存储考勤记录。（需要考勤时，可以用到）

8.3 时间设置

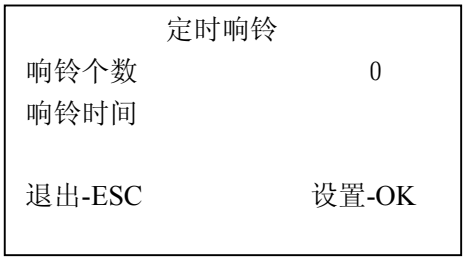
在【系统设置】菜单中选择【时间设置】，进入时间设置界面。

设备在出厂前已经调整为北京时间，但是不排除在客户购买过程中因为测试功能造成了时间变化。

开始使用之前，须要按照当前时间调整设备时间。

8.4 定时响铃

在【系统设置】菜单中选择【定时响铃】，进入响铃设置界面。

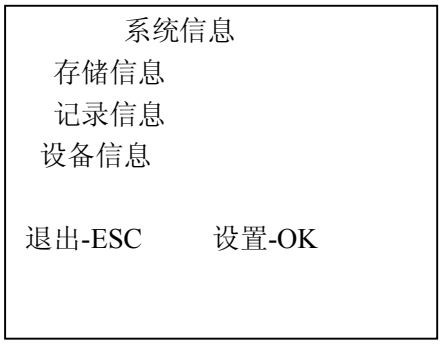


【响铃个数】：响铃响闹的次数；

【响铃时间】：设置响铃响闹的时间，最多可以设置8个时间点。

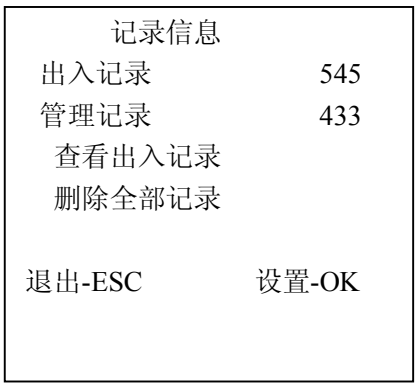
九、系统信息

【系统信息】菜单用于查询设备的一些信息，包括“存储信息”、“记录信息”、“系统信息”如下图：

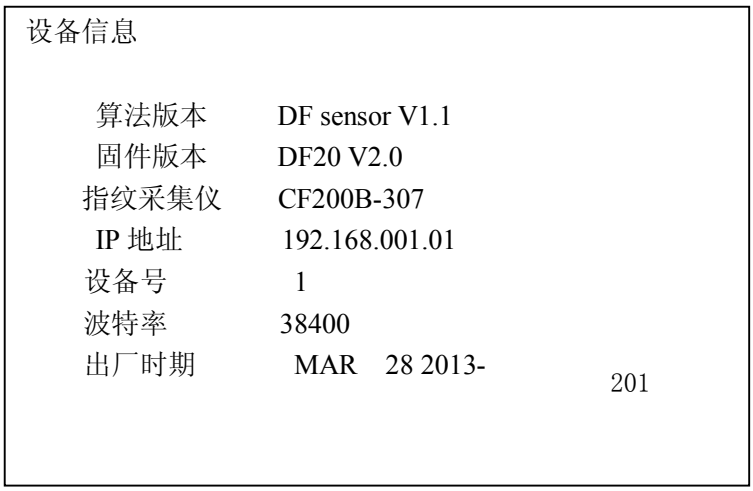


9.1 存储信息：显示设备上已注册信息，用上下键移动光标至“存储详情”按 OK 键，或按数字键“1”，进入查看。如下图,用上下键或数字键可选择要查询的注册信息。

9.2 记录信息：显示设备各种记录条数以及删除机器内所有记录，用上下键移动光标至 “记录详情”项按 OK 键，



9.3 设备信息：显示设备上的相关信息。如下图：



智能楼宇

第二章 视频监控

2.1 大唐移动 NETSDK 及例程简要说明

大唐移动 NETSDK 及例程简要说明

一、 概要说明

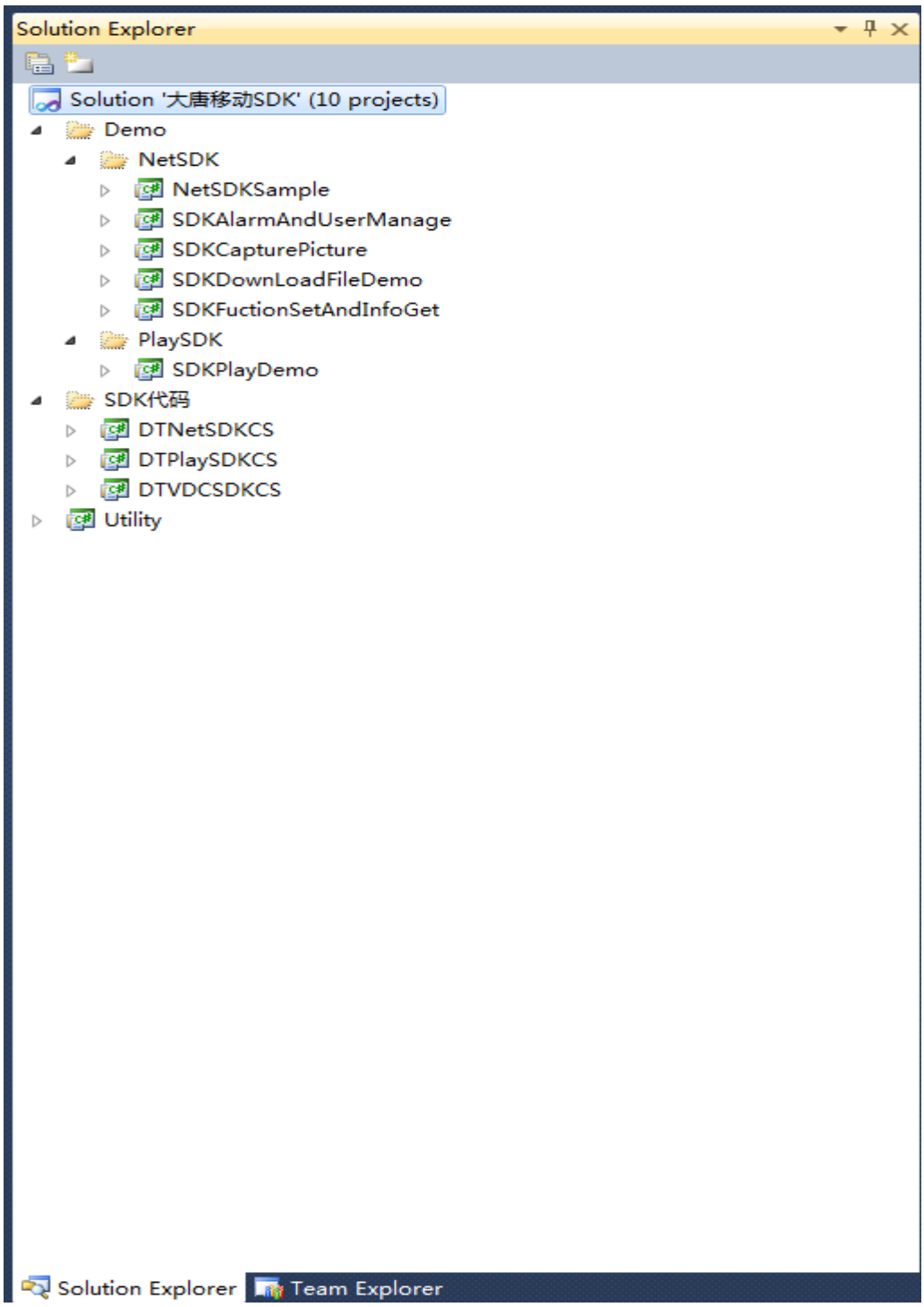
NETSDK 是对大唐“网络 SDK”、“播放 SDK”使用 C#进行的封装调用，使用该 SDK 可以接入视频设备。

该 SDK 封装了 C++版本 SDK 的主要功能接口，客户可在此基础上进行增减。

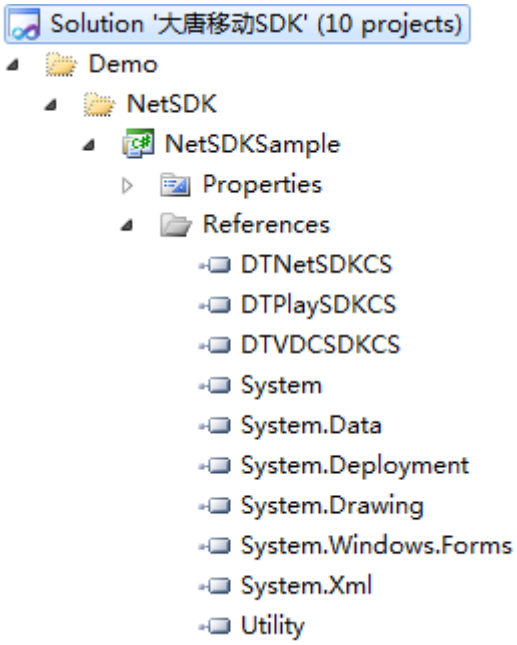
使用 IDE 的工具：VS2010

框架：.net framework 2.0

二、 例程解决方案图


Solution Explorer
Solution '大唐移动SDK' (10 projects)
Demo
NetSDK
NetSDKSample
SDKAlarmAndUserManage
SDKCapturePicture
SDKDownLoadFileDemo
SDKFuctionSetAndInfoGet
PlaySDK
SDKPlayDemo
SDK代码
DTNetSDKCS
DTPlaySDKCS
DTVDCSDKCS
Utility
Solution Explorer
Team Explorer

三、 工程项目引用



- DTNetSDKCS

网络 SDK

- DTPlaySDKCS

播放 SDK

- DTVDCSDKCS

解码卡 SDK

用户在使用时请先引用这三个动态库。

四、 编码说明

1. 软件环境

DEMO 程序使用 VS2010 进行编码。

2. 命名空间

using DTNetSDK

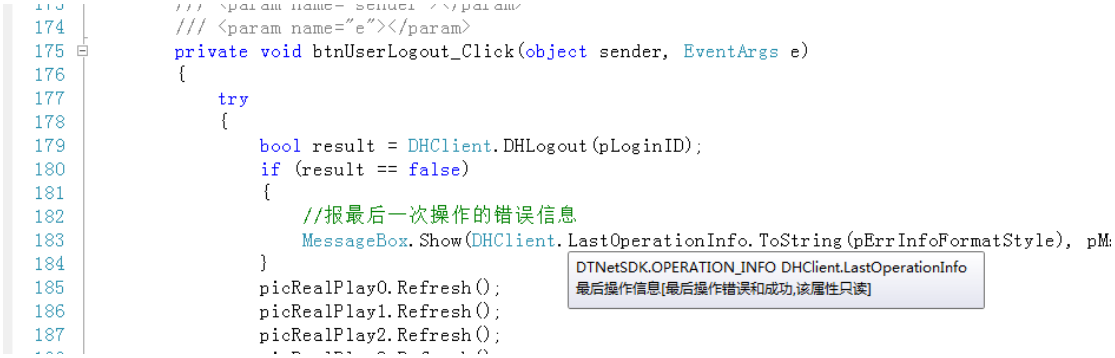
using DTPlaySDK

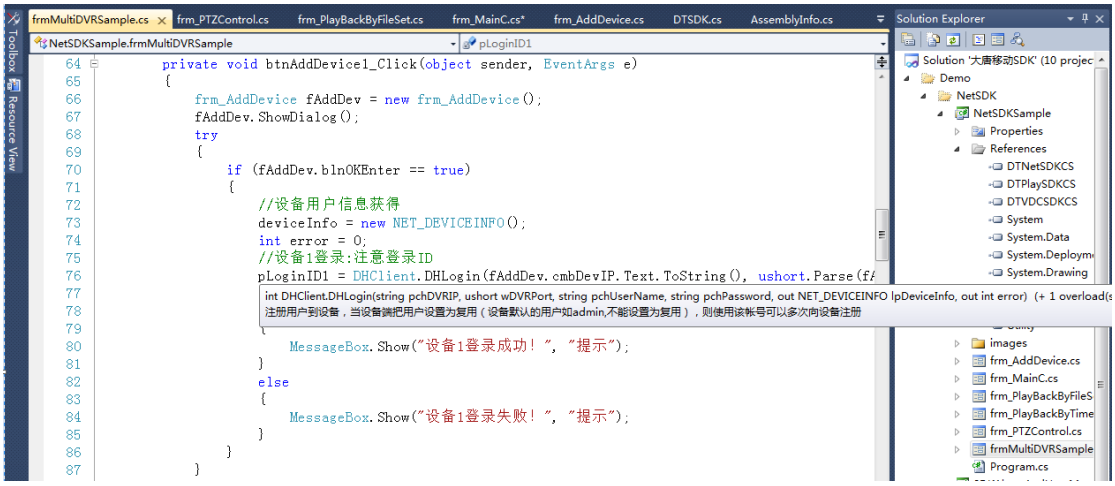
using DTVDCSDK

3. 编码智能提示

编码过程中每个类及其方法、属性的使用都会有详细的智能中文提示说明，无需查询相应的 SDK 文档。

例如：





五、 例程说明

1. 工程名：NetSDKSample

- 功能列表：
- 用户登录
- 实时监视
- 按文件回放
- 按时间回放
- 云台控制

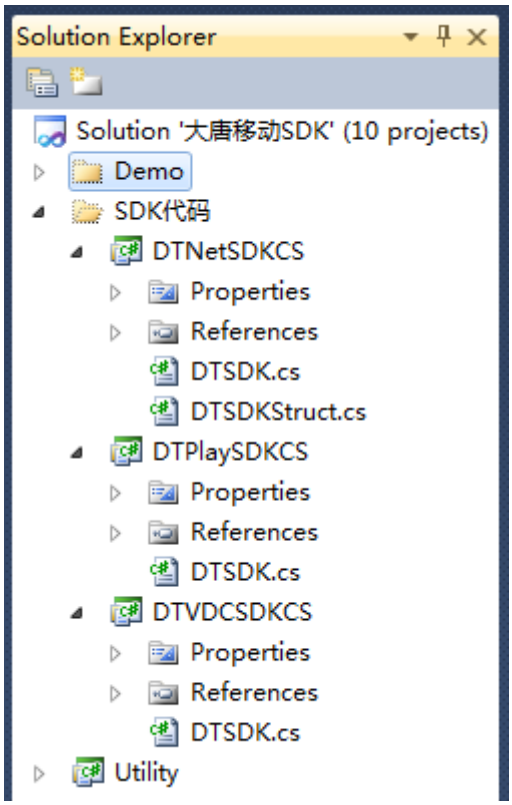
- 扩展云台控制
- 抓图
- 多 DVR 同时播放

2. 工程名：SDKAlarmAndUserManage
   - 用户管理
   - 用户组管理
   - 报警信息获取和显示
3. 工程名：SDKCapturePicture
   - 远程抓图设置
   - 手动抓图
   - 定时抓图
4. 工程名：SDKDownLoadFileDemo
   - 按照时间下载文件
   - 安装文件下载文件
5. 工程名：SDKFuctionSetAndInfoGet
   - DVR 参数的取得验证
   - DVR 参数的保存验证
6. 工程名：SDKPlayDemo
   - 按文件播放录像数据
   - 播放控制
   - 局部放大
   - 单帧播放

- 快速定位
- 播放时抓图
- 数据流录像【录像格式转为 AVI】
- 音量控制
- 颜色参数调整
- 自定义叠加内容

六、 SDK 代码说明

该部分代码对原始 C++开发的 SDK 代码进行 C#封装。



DTNetSDKCS 工程对网络 SDK 进行封装。

DTPlaySDKCS 工程对播放 SDK 进行封装。

DTVDCSDKCS 工程对解码卡 SDK 进行封装。

智能楼宇

第三章 视频发布

3.1 多媒体信息发布系统

多媒体信息发布系统 V5.0 操作手册

第一章：系统简介

第一节、系统介绍

多媒体信息发布系统是利用显示屏将企业宣传、实时通知全方位展现出来的一种高清多媒体显示技术。系统是将音视频、电视画面、图片、动画、文本、文档、网页、流媒体、数据库数据等组合成一段段精彩的节目，并通过网络将制作好的节目实时的推送到分布在各地的媒体显示终端，从而将精彩的画面、实时的信息资讯在各种指定场所全方位的完美展现在所需的群众眼前。

第二节、系统组成

多媒体信息发布系统由：服务器控制端、网络平台、终端网络播放器、显示设备四部分组成。

1. 服务器控制端：

   节目制作、节目管理、节目发布、终端管理、系统管理等

2. 网络平台：

   终端访问服务器的网络的通道， 系统支持多个网络平台，主要支持：局域网、广域网、DDN 专网、无线 WI-FI、3G、ADSL 等网络

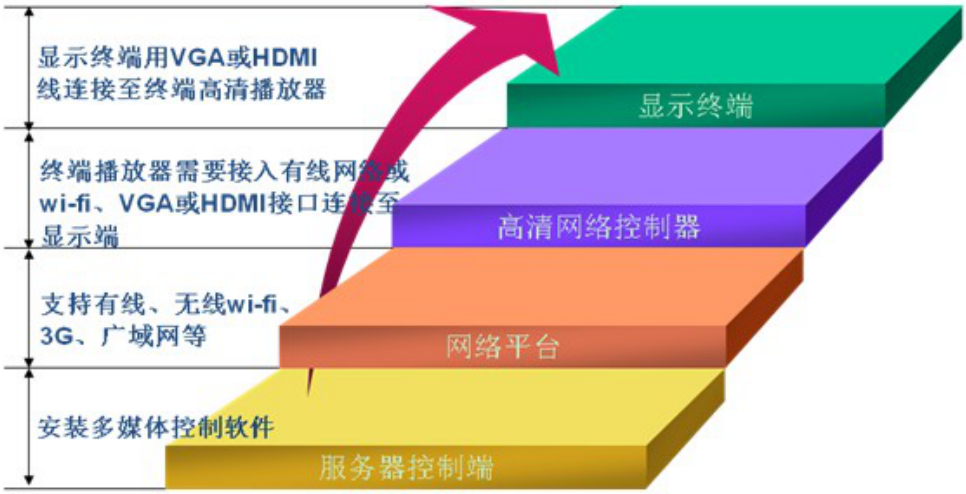
3. 网络多媒体播放器：

   播放服务器端发送的，视频、图片、文字、网页，office 文档，PDF 文档、FLASH、

天气等多种素材

4．显示设备:

包括液晶显示器/电视（LCD)、等离子电视（PDP)、微显示电视（DLP)、 全彩 LED 大屏幕、CRT 电视、投影仪、多屏幕拼接电视墙等



第三节：系统运行环境

1、服务器硬件运行环境

- CPU ：Intel 双核 2.0 GHz 或以上
- 内存：2GB 内存或以上
- 硬盘：320GB 硬盘空间大小或以上

2、服务器软件运行环境

- 操作系统： windowsXP/2003/ Win7/（推荐 windows server 2003）
- IE 浏览器：IE 8/9（推荐 IE8）

第二章：系统安装

第一节、终端安装调试

一、嵌入式终端

播放终端设置由供应商出厂默认安装，用户根据网络环境可设置本机 ip 和服务器 ip 即可，其他不同设置。

1.设置以太网静态 ip 地址

终端默认为自动获取 IP，插上 USB 键盘，等待设备完全启动后，按键盘数字”2”键，进行终端网络设置。

(1) 按键盘→键选择“以太网”,按键盘回车键进入

(2) 按键盘↓键选择高级配置，进入 IP 设置

(3) 因为默认为动态分配、设置静态 ip 时，需要取消动态分配。取消动态分配后，完整填写下面 IP 地址,网关,字码掩码即可(如需播放互联网信息则需要设置相应网络 DNS)。回车键进行操作，全部操作完成，按 ESC 键退回播放画面或重启播放终端。

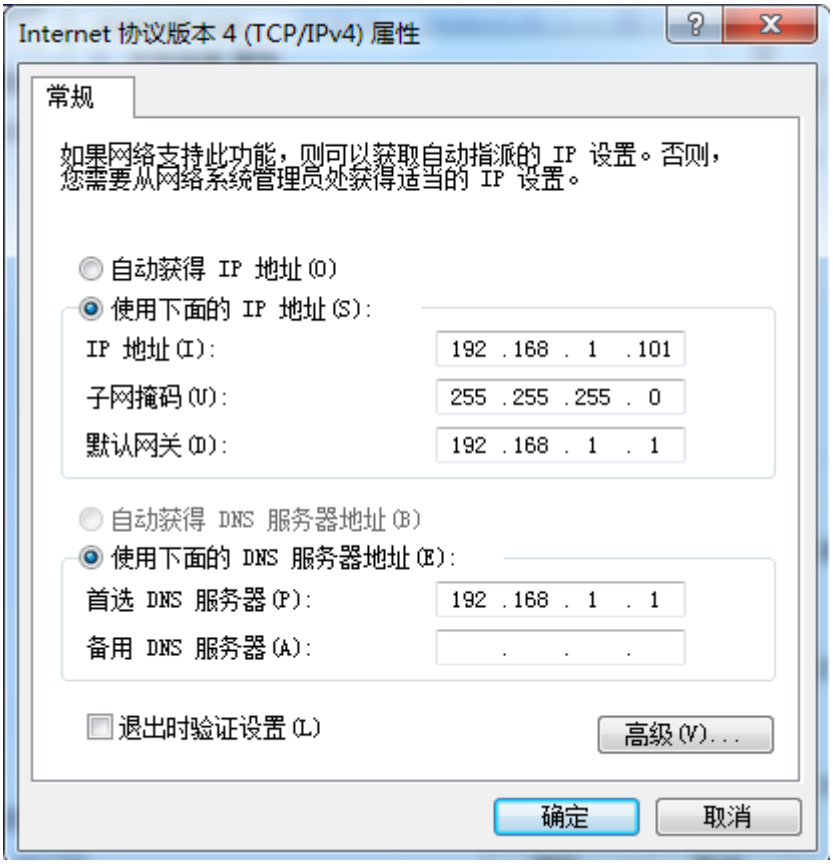
2.设置连接服务器 ip 地址

终端通过服务器 IP 和服务器建立连接，按数字“1”键设置连接服务器 IP，填写服务器电脑的 IP，设置完成后保存即可。

二、X86 播放终端

1.设置以太网静态 ip 地址

打开网卡属性 -----设置以太网静态 ip



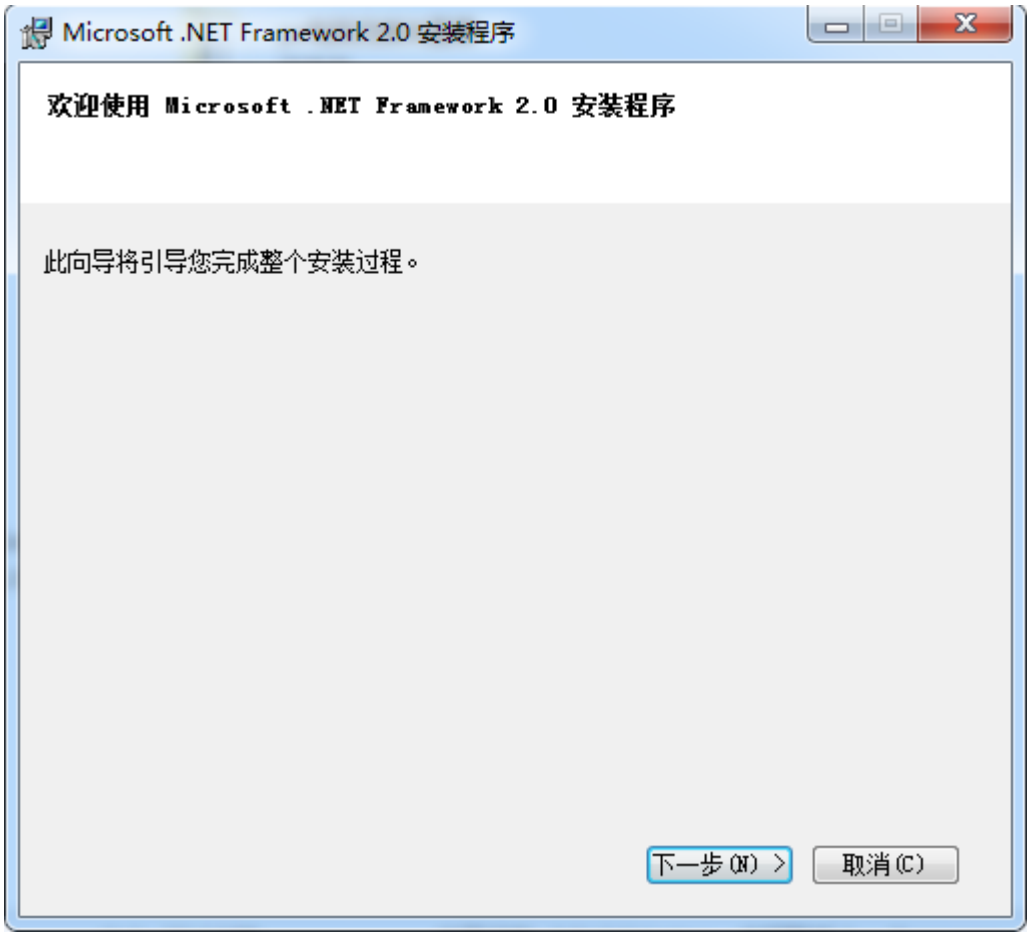
2.设置连接服务器 ip 地址

多媒体播放程序启动后，在白屏界面下按“F1”快捷键设置服务器 ip 地址，按“F2”键关闭多媒体播放程序，按“ESC”键最小化应该程序

第二节、控制系统安装

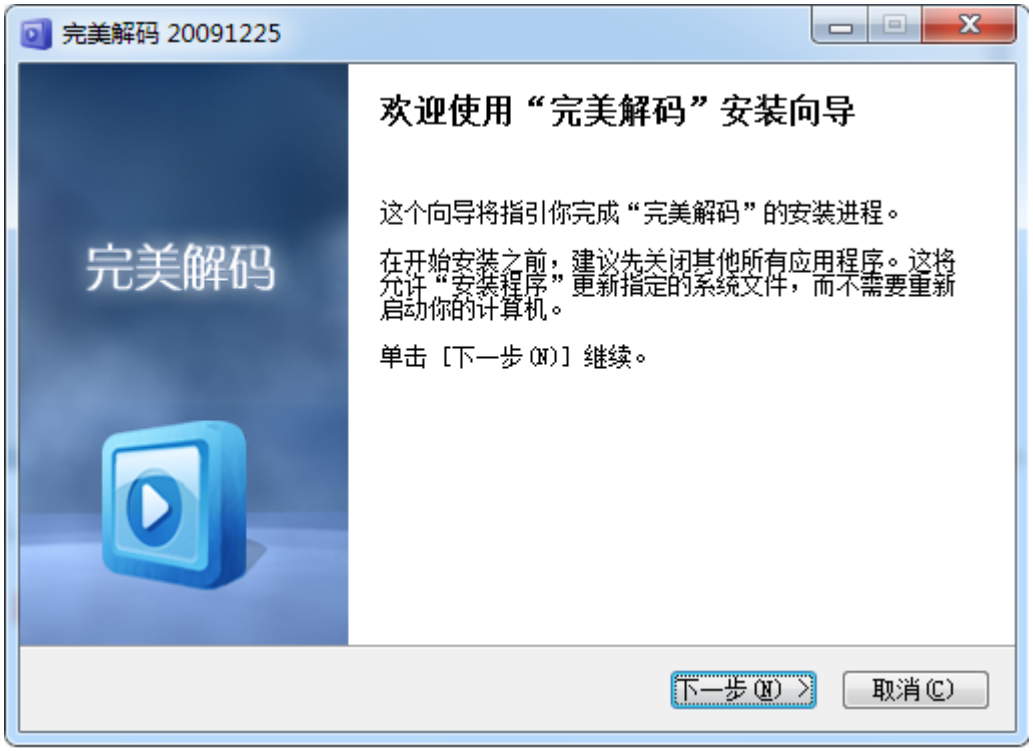
1. 安装系统补丁 Microsoft .NET Framework2.0

服务器系统需要安装 Microsoft .NET Framework，xp 和 2003 系统安装 Microsoft .NET Framework2.0 即可，win7 系统默认已经安装更高的.net 系统补丁，所以不用安装，安装时选择默认的“下一步”完成安装即可。



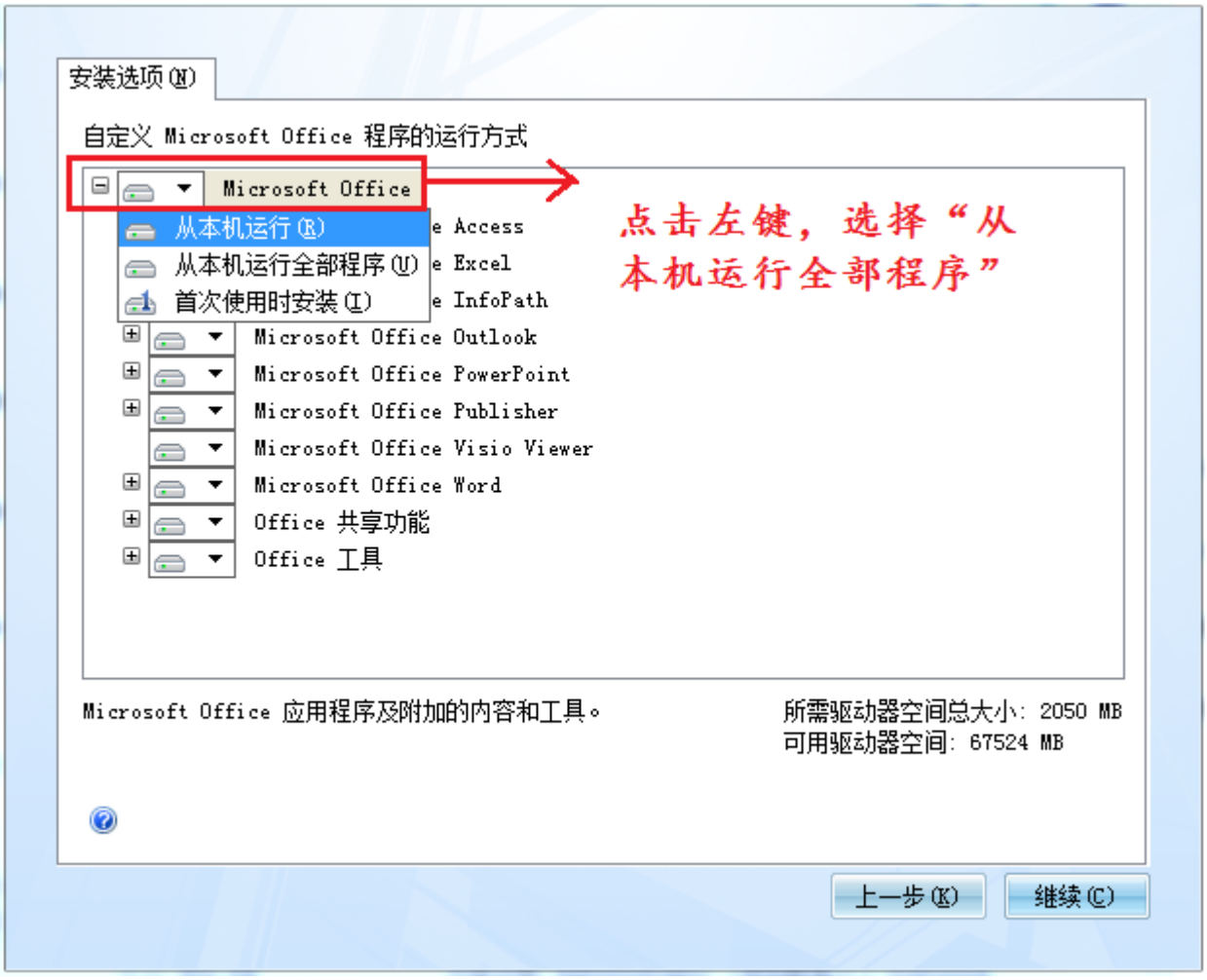
2. 安装视频解码器

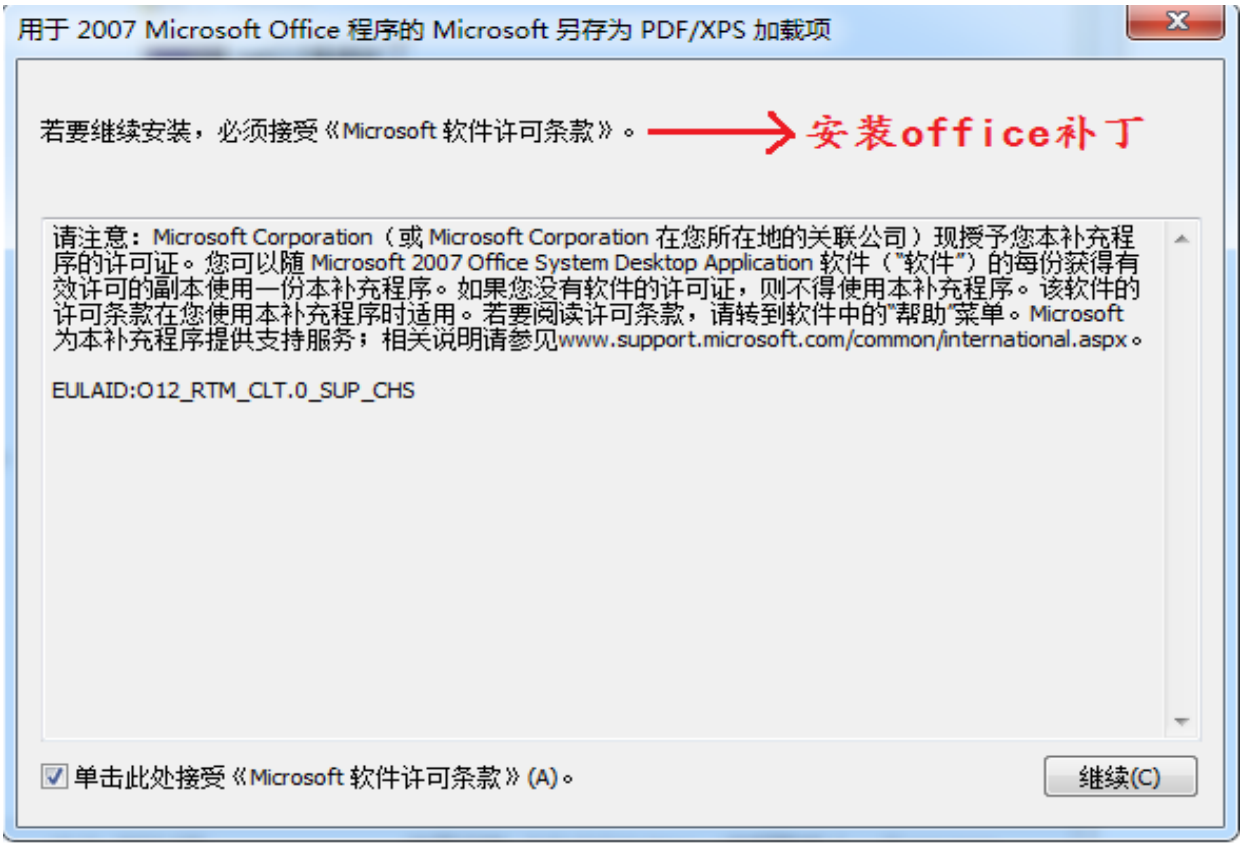
此视频解码器为服务器端预览视频节目时调用的播放器，如果未安装预览视频节目视频无法显示，但不影响发送或终端播放视频，因为终端有单独的硬件解码器，安装此程序是选择默认“下一步”完成安装即可。



3. 安装 office2007 及 office2007 系统补丁

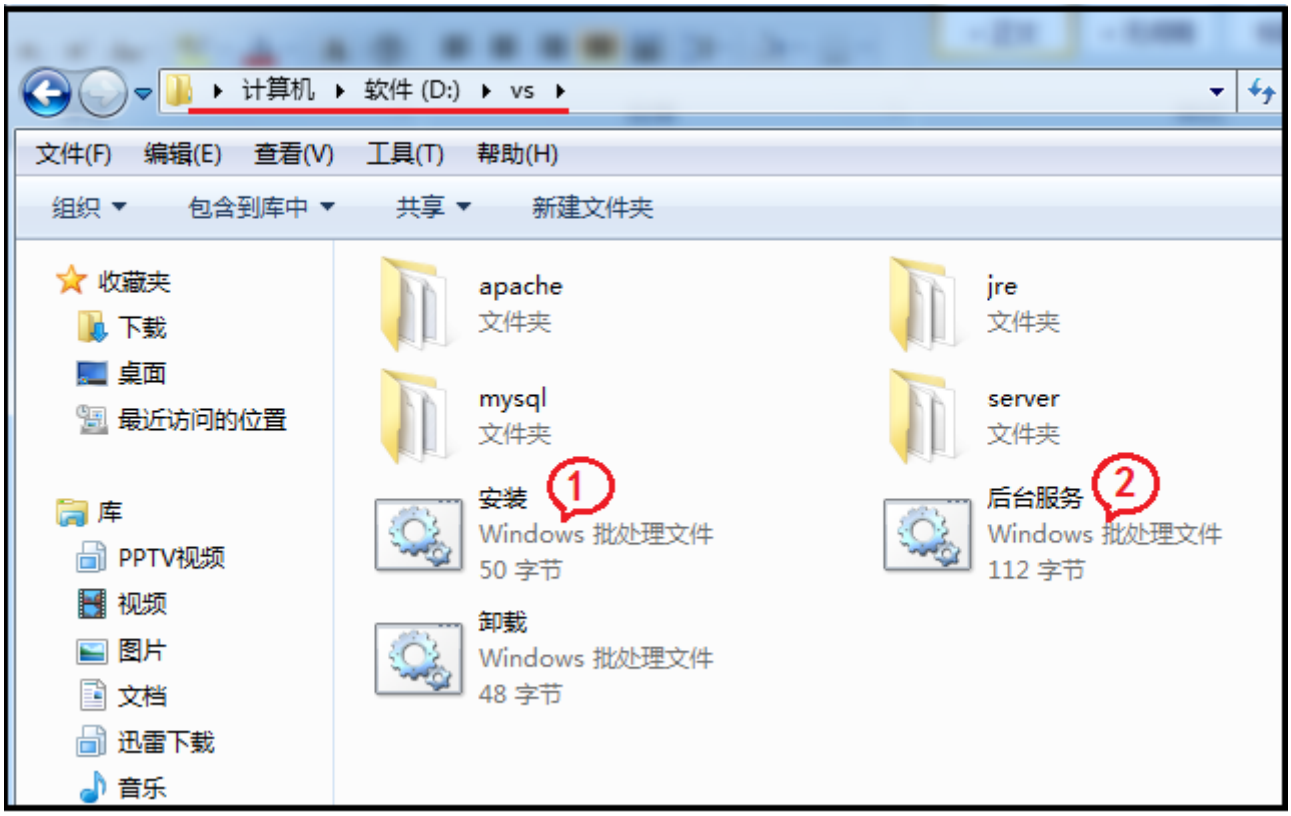
如需播放 office 文档 ppt、word、excel，则需要安装 office2007 版本，播放 office 文档时，会调用 office2007 系统模块，所以必须安装此软件，否则无法播放 office 文档内容。安装 office 软件时，需要选择“自定义安装模式”——“从本机运行全部程序”，然后选择“下一步”到完成即可，如果不选择“从本机运行全部程序”可能导致某些文档不能正常播放。


安装选项(N)
自定义 Microsoft Office 程序的运行方式
Microsoft Office
从本机运行(R)
从本机运行全部程序(U)
首次使用时安装(I)
Microsoft Office Outlook
Microsoft Office PowerPoint
Microsoft Office Publisher
Microsoft Office Visio Viewer
Microsoft Office Word
Office 共享功能
Office 工具
点击左键，选择“从本机运行全部程序”
Microsoft Office 应用程序及附加的内容和工具。
所需驱动器空间总大小: 2050 MB
可用驱动器空间: 67524 MB
上一步(K)
继续(C)

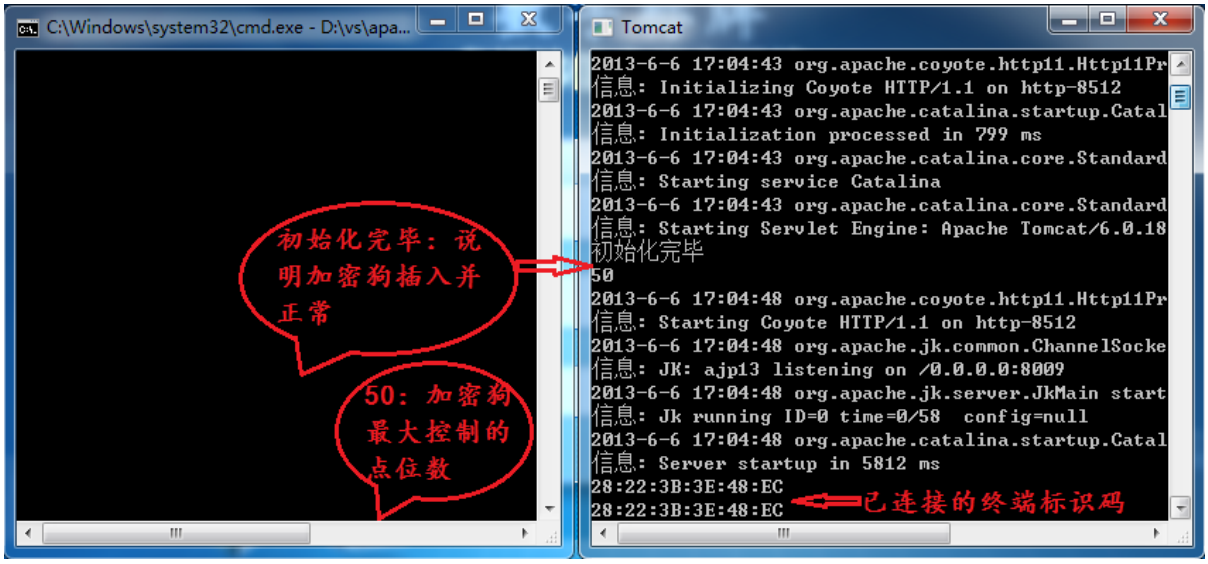

用于 2007 Microsoft Office 程序的 Microsoft 另存为 PDF/XPS 加载项
若要继续安装，必须接受《Microsoft 软件许可条款》。
安装office补丁
请注意：Microsoft Corporation（或 Microsoft Corporation 在您所在地的关联公司）现授予您本补充程序的许可证。您可以随 Microsoft 2007 Office System Desktop Application 软件（"软件"）的每份获得有效许可的副本使用一份本补充程序。如果您没有软件的许可证，则不得使用本补充程序。该软件的许可条款在您使用本补充程序时适用。若要阅读许可条款，请转到软件中的"帮助"菜单。Microsoft 为本补充程序提供支持服务；相关说明请参见www.support.microsoft.com/common/international.aspx。
EULAID:O12_RTM_CLT.0_SUP_CHS
单击此处接受《Microsoft 软件许可条款》(A)。
继续(C)

4. 安装多媒体控制系统

安装控制软件：本系统属于绿色安装版,只要将光盘“控制软件”目录下面的/vs 整个目录拷贝到/D 盘即可,系统数据库及其系统文件都在 vs 目录下。第一次使用只需要点击“**安装**”即可，会出现黑色 DOS 窗口闪过.说明数据库已经安装完毕.（win7 系统需要点击右键——以管理员身份运行）



运行控制软件：先将加密狗插入服务器 USB 接口上、然后点击 “**后台服务**”会启动两个黑色窗口,这个就是后台服务不能关闭,



5. 登录多媒体控制系统

本系统采用 B/S 架构，直接在浏览器输入登录地址即可进入系登陆地址

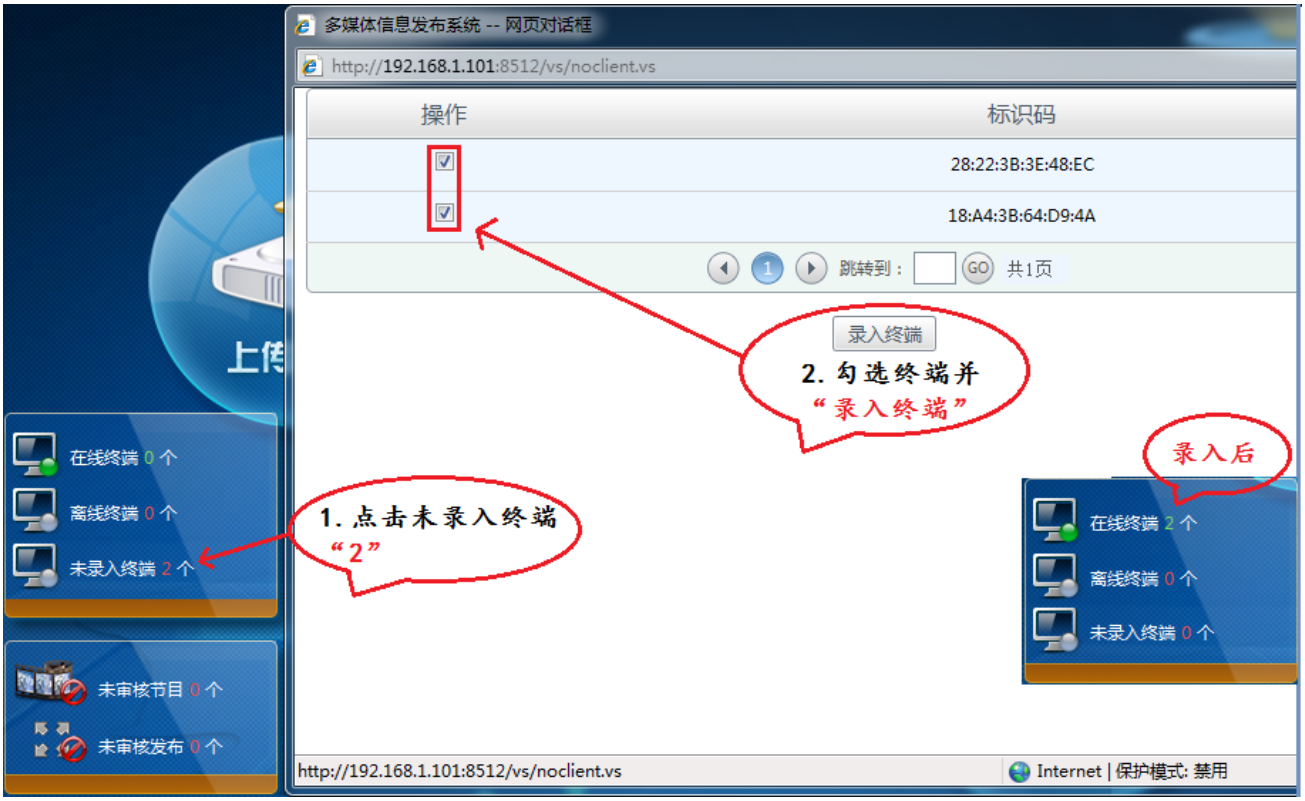
http://服务器 ip 地址:8512/vs

注意事项：软件安装过程中如果系统防火墙、杀毒软件、安全软件如 360 安全卫士、金山卫士等出现提示框，请选择允许程序的操作这一选项即可；本系统软件已通过金山、瑞星、卡巴、NOD、360、小红伞等杀毒软件的安装检测，无病毒无插件，请用户放心安装使用。

第三章：快速使用系统

一、录入终端

系统安装完成后，首先需要将调试好的终端录入到系统，否则将无法控制终端。如下图所示、首次连接到系统的终端在“首页——未录入终端”栏显示，点击未录入的终端数“2”——“勾选所有终端”——“录入终端”



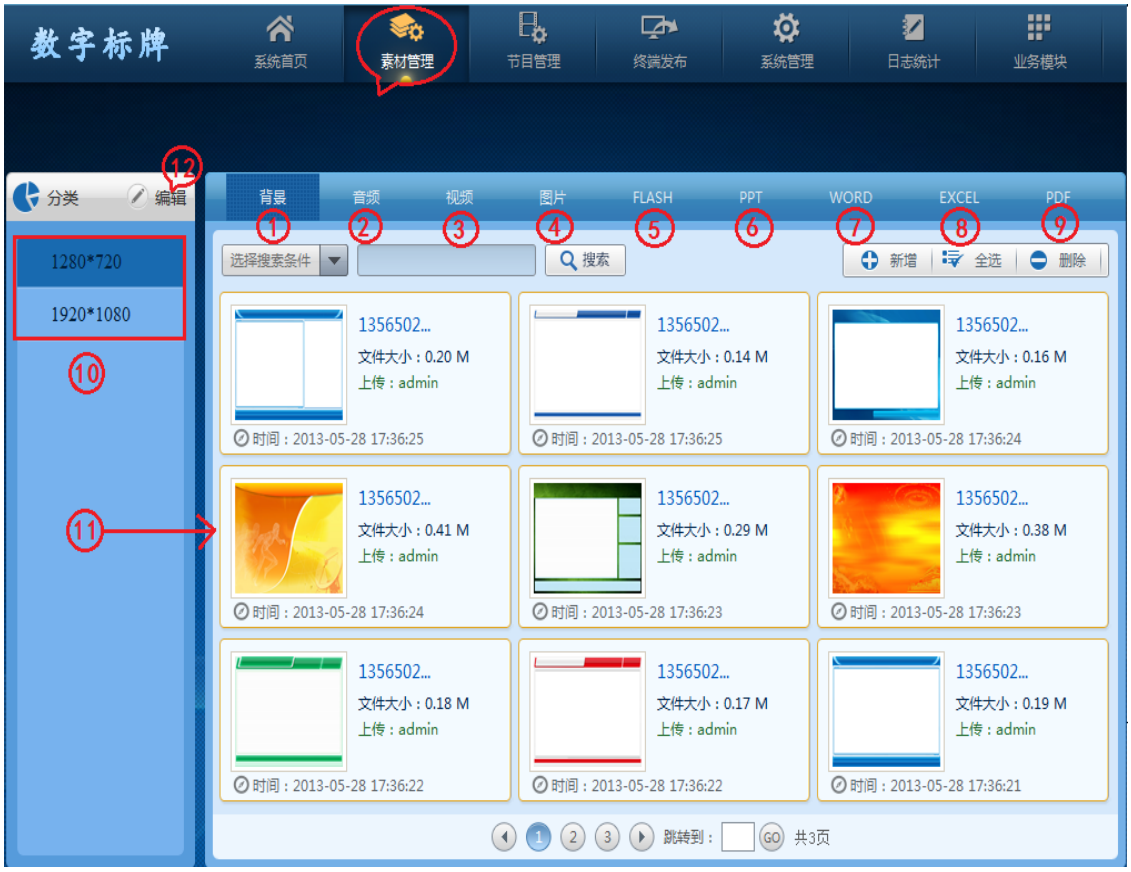
二、 上传素材

上传所有需要用到的多媒体素材、例如：图片、视频、ppt 等，可在主页面点击 

或直接点击主界面的 

进入添加素材窗口，如下图所示


数字标牌
系统首页
素材管理
节目管理
终端发布
上传素材
节目制作
在线终端 2 个
离线终端 0 个
未录入终端 0 个


数字标牌
系统首页
素材管理
节目管理
终端发布
系统管理
日志统计
业务模块
分类
编辑
1280*720
1920*1080
背景
音频
视频
图片
FLASH
PPT
WORD
EXCEL
PDF
选择搜索条件
搜索
新增
全选
删除
1356502...
文件大小：0.20 M
上传：admin
时间：2013-05-28 17:36:25
文件大小：0.14 M
时间：2013-05-28 17:36:25
文件大小：0.16 M
时间：2013-05-28 17:36:24
文件大小：0.41 M
时间：2013-05-28 17:36:24
文件大小：0.29 M
时间：2013-05-28 17:36:23
文件大小：0.38 M
时间：2013-05-28 17:36:23
文件大小：0.18 M
时间：2013-05-28 17:36:22
文件大小：0.17 M
时间：2013-05-28 17:36:22
文件大小：0.19 M
时间：2013-05-28 17:36:21
跳转到：
GO
共3页

1、**背景**：背景为节目的底板，可以做些模板样式作为节目的背景，这样节目编辑更美观，制作节目时不选择背景图片，则节目底板为白色，不影响其他使用。

2、**音频**：可添加 mp3、wav、wma 等格式音频，此音频可以用在节目里面作为背景音乐。

3、**视频**：可添加多种格式视频、此系统支持所有常用视频格式 AVI、3GP、MP4、FLV、RM、RMVB、MOV、VOD 等常用视频格式

4、**图片**：可添加多张图播放

5、**FLAHS**：添加 flahs 素材

6、**PPT**：可添加 ppt 文档

7、**WORD**：可添加 word 文档

8、**EXCEL**：可添加 excel 文档

9、**PDF**：可添加 pdf 文档

10、**1280*720**：已添加 1280*720 背景文件夹

1920*1080：已添加 1920*1080 背景文件夹

11、**编辑**：新键和删除背景文件夹

12、**素材显示区域**

下面举例说明添加素材：

1.添加/删除背景：

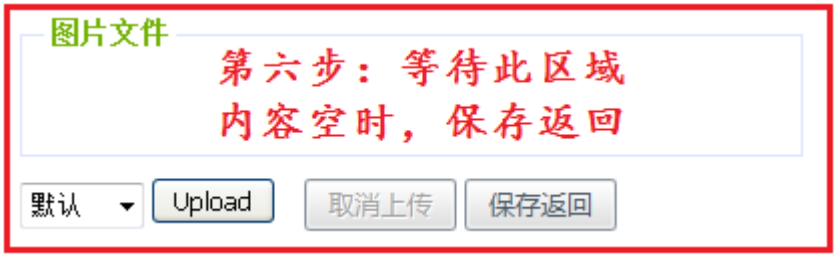
添加背景文件夹：点击“编辑”按钮——“填写新分辨率”—— “新增分组”、如下图所示添加一个 1024*768 分辨率名称完成。

删除背景文件夹：点击“编辑”按钮——“选择分辨率”——“删除”

2.添加背景：

选择“背景栏目”——“选择分辨率大小”——“新增或删除背景模板”

——在上传窗口点击“Upload”——“选择背景图片”——等待上传内容完成后点击“保存返回”，这样背景图片上传完成



3.添加视频：

选择素材管理“视频”——点击视频上传窗口中的“新增”——选择上传按钮“Upload”——上传完成后点击“保存返回”即可

4.添加其他素材（同上）：添加的素材可显示素材名称、素材大小、时间长度、上传用户、上传时间等

三、制作节目

需要播放的素材上传完成之后、接下来就可以制作节目，操作步骤如下：点击主菜单“ 

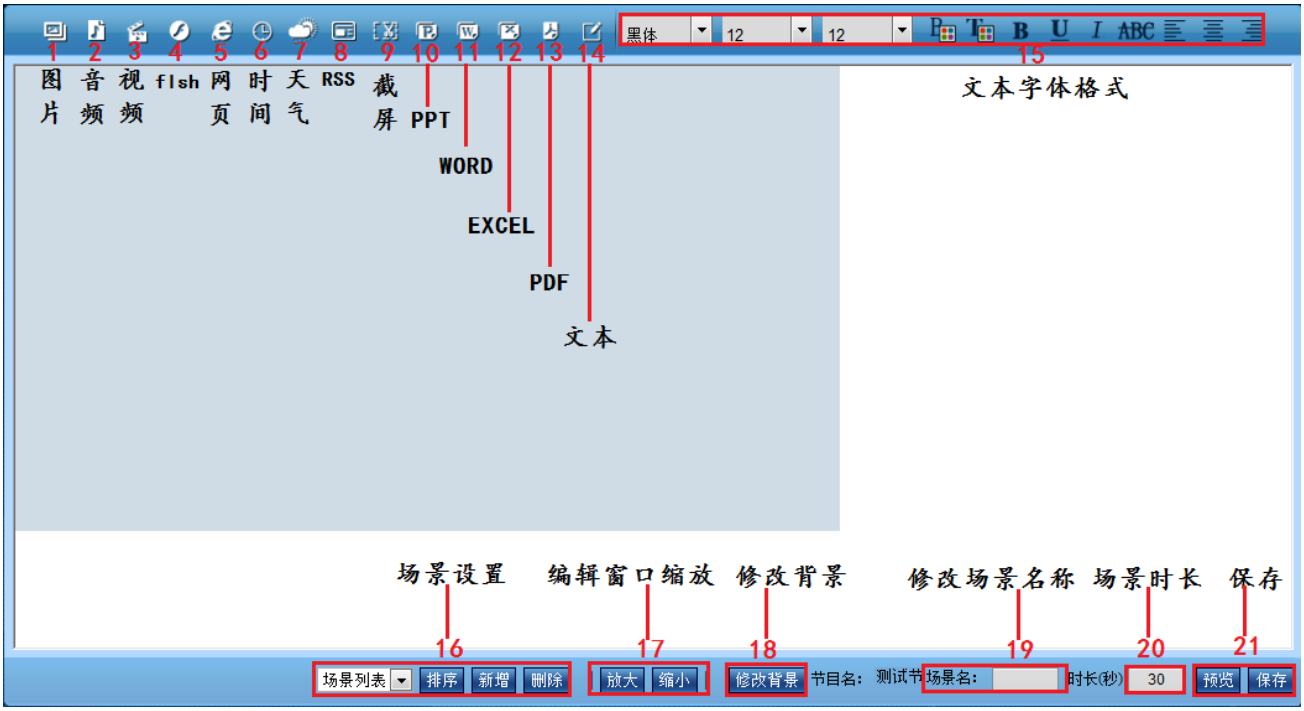
”模块或者直接点击主页上“ 
”按钮进入节目管理窗口、如下图所示：

1. 节目列表：显示已经制作的节目、节目名称、节目大小、节目时长、制作时间、节目是否已审核。
2. 新增：新建一个新的节目。
3. 全选：选择所有节目
4. 修改：修改制作的节目
5. 复制：复制一个节目、并提醒另存为新节目
6. 删除：删除节目
7. 导出节目：可将制作好的节目导出、然后可将导出的节目导入到终端播放，此功能可用在网络不通的单机模式下使用。（后续章节详细介绍）

制作节目举例说明

点击节目管理栏中“新增”按钮——输入“节目名称”——选择“节目分辨率”——“新增”如下图所示

在节目编辑窗口添加素材、制作节目。



制作好节目后可以预览、保存

四、节目发布

选择主页上的“ 

”或“ 

”进入发布界面——1 选择“发布节目”—— 2.“选择要发布的终端”——3.点击“选择节目发布”按钮——4.“选择要发布的节目”——5.“提交发送”

五、状态监控

“状态监控”可以监控所发送的节目下载情况和指令执行情况、根据执行结果可以清楚的看到操作是否成功。选择主页终端发布菜单下的“执行结果”或者直接点击主页的“状态监控”即可。可以看到终端在线状态、终端操作、下载百分比、操作等信息

第四章：系统首页介绍

登录系统进入系统首页如以下图所示，显示所有功能模块。

1. 系统首页：系统登录显示的主界面，包含所有功能菜单模块，一目了然。

2. 素材管理：管理所有需要播放的素材。

3. 节目管理：管理所有节目。

4. 终端发布：发布所有节目到终端、修改终端参数

5. 系统管理：终端管理、用户管理、系统设置

6. 日志统计：管理日至。

7. 业务模块：预览模块用在其他模块开发。

系统首页

1、系统 logo：可更改软件系统 logo

2、修改密码：用户可以修改本身登陆密码（密码不能为空）

3、快捷菜单：此 4 个快捷菜单分别代表，素材管理、节目管理、终端发布、执行结果四个菜单选项。

4、在线终端 1 个：显示已连接系统的终端数量、点击数字“1”可以监控终端播放画面

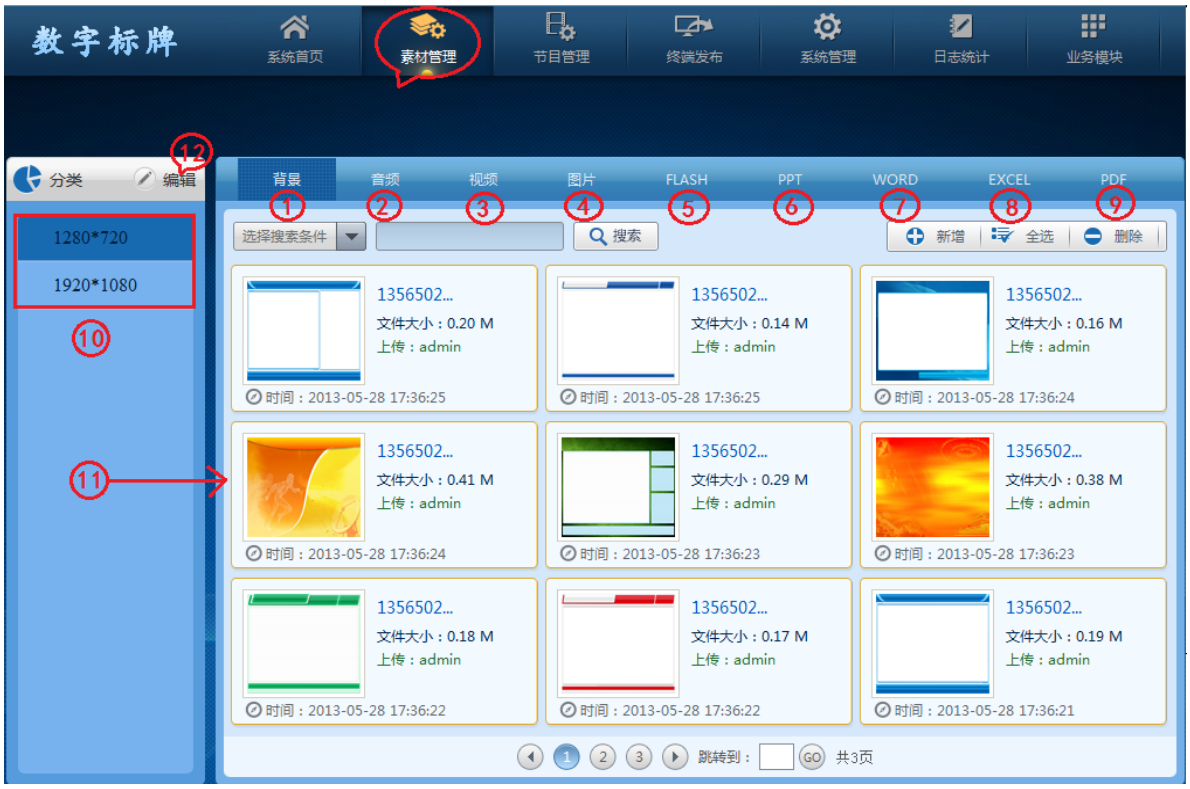
5、离线终端 0 个：显示未连接系统的终端数量

6、未录入终端 0 个：显示已连接后台且未录入终端的数量

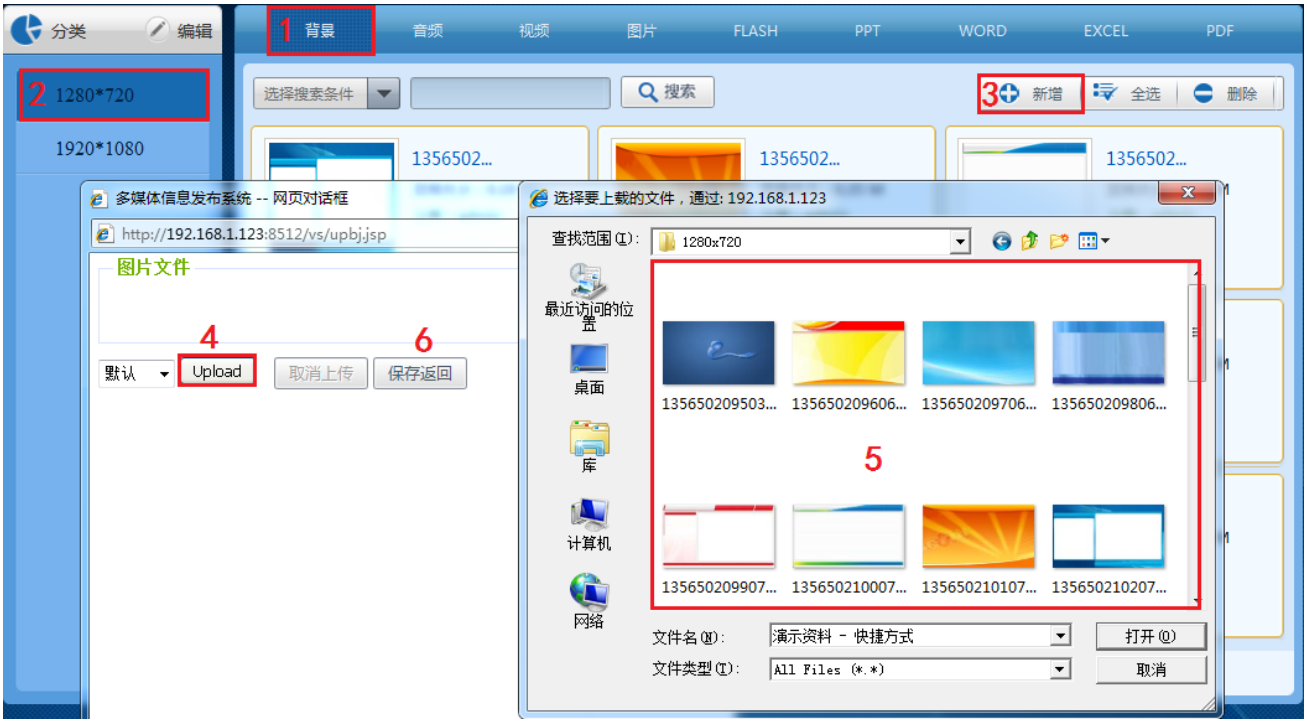
7、未审核节目 0 个：显示未审核的节目数量

8、 未审核发布 0 个：显示未审核发布的数量

第五章、素材管理



1、 **背景**：背景为节目的底板，可以做些模板样式作为节目的背景，这样节目编辑更美观，制作节目时不选择背景图片，则节目底板为白色，不影响其他使用。操作步骤：1 选择“背景”——2 选择对应“分辨率”——3“新增”——4 点击上传“Upload”——5 选择“背景图片”——6 添加完成后“保存返回”



2、**音频**：可添加 mp3、wav、wma 等格式音频，此音频可以用在节目里面作为背 景音乐。操作（同 3）

3、**视频**：可添加多种格式视频、此系统支持所有常用视频格式 AVI、3GP、MP4、FLV、RM、RMVB、MOV、VOD 等常用视频格式。操作如下：选择素材管理“视频”——点击视频上传窗口中的“新增”——选择上传按钮“Upload”——上传完成后点击“保存返回”即可

4、**图片**：可添加多张图播放（同上）

5、**FLAHS**：添加 flahs 素材（同上）

6、**PPT**：可添加 ppt 文档（同上）

7、WORD：可添加 word 文档（同上）

8、EXCEL：可添加 excel 文档（同上）

9、PDF：可添加 pdf 文档（同上）

10、1280*720：已添加 1280*720 节目分辨率

1920*1080：已添加 1920*1080 节目分辨率

11、编辑：新键和删除分辨率（同上）

12、素材显示区域

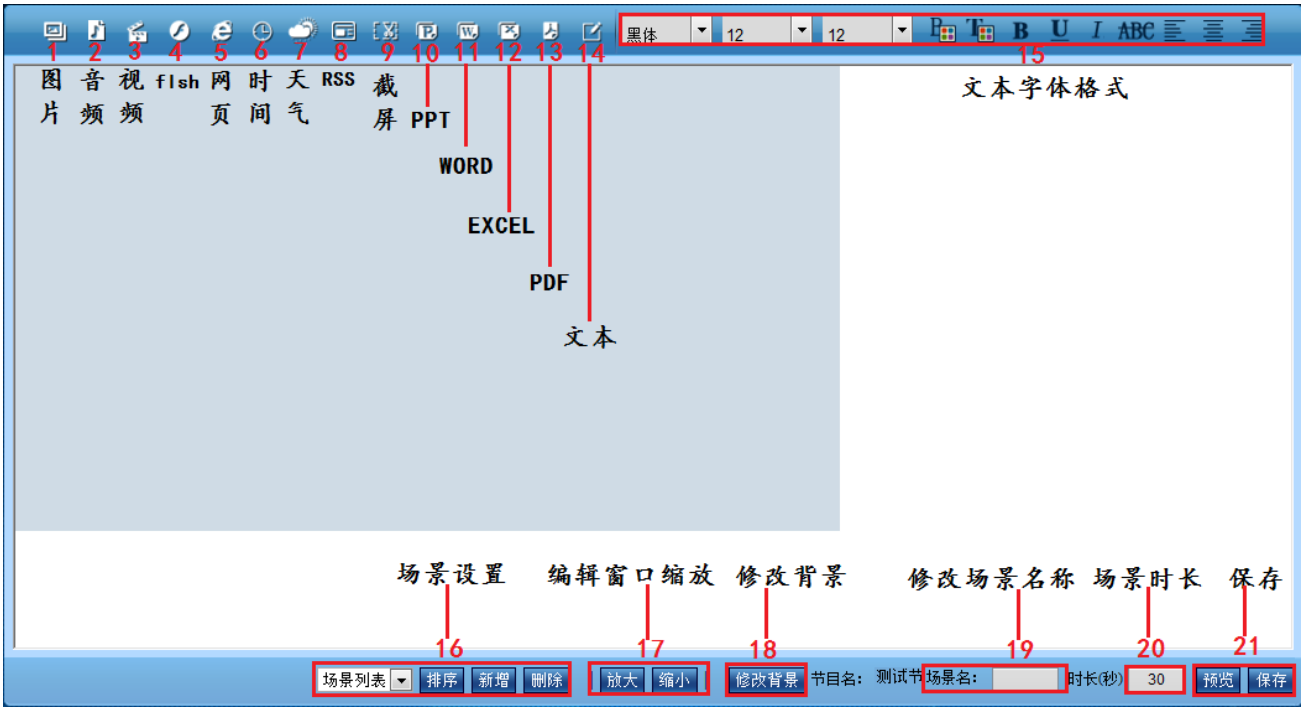
第六章、 节目管理

6.1 节目列表：用户可以查看本身和上级制作的节目信息

6.2 新增：制作节目

点击节目管理栏中“新增”按钮——输入“节目名称”——选择“节目分辨率”——“新增”如下图所示

在节目编辑窗口添加素材、制作节目，制作好节目后可以预览、保存。



1．添加图片

（1）“添加”：添加所需要播放的图片

（2）“全选”：选择所有图片

（3）“新增图片”：当列表中没有所需图片时，可直接从本地上传图片素材。

（4）“上移、下移、删除”：可排列图片顺序、删除已经选择的图片

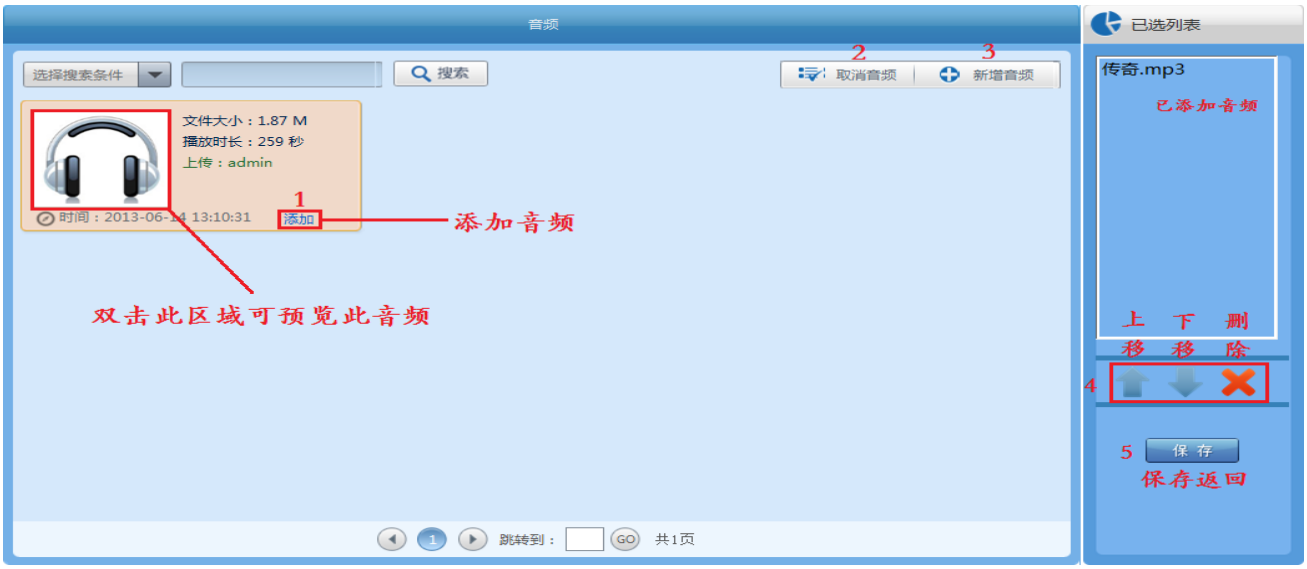
（5）默认图片大小：默认添加图片大小、可以更改

（6）图片间隔：两张图片播放时间间隔时长

（7）特效：选择特效后、图片播放会有特效功能、例如：渐变、百叶窗等效果

（8）保存：保存添加的图片

2. 添加音频



（1） 添加音频：添加音频文件到节目

（2） 取消音频：取消添加音频到节目

（3） 新增音频：当所需音频文件为上传时、可在此处直接上传音频素材到软件

（4） 上移、下移、删除：多个音频文件可以排列播放顺序、可删除已选择音频

（5） 保存：保存返回添加页面

3. 添加视频

（1） 添加：添加视频到节目

（2） 新增视频：当视频列表中没有需要添加的素材是、可直接上传新的素材到软件

（3） 已添加到节目中的视频

（4） 上移、下移、删除：设置多个视频的播放顺序、删除已选择的视频

（5） 视频默认大小

（6） 添加流媒体播放地址：此功能用在流媒体直播上、通过流媒体采集设备将流媒体编码成 rtsp 协议的 ip 流、将此地址添加到这里即可。

（7） 保存返回

4. 添加 flash

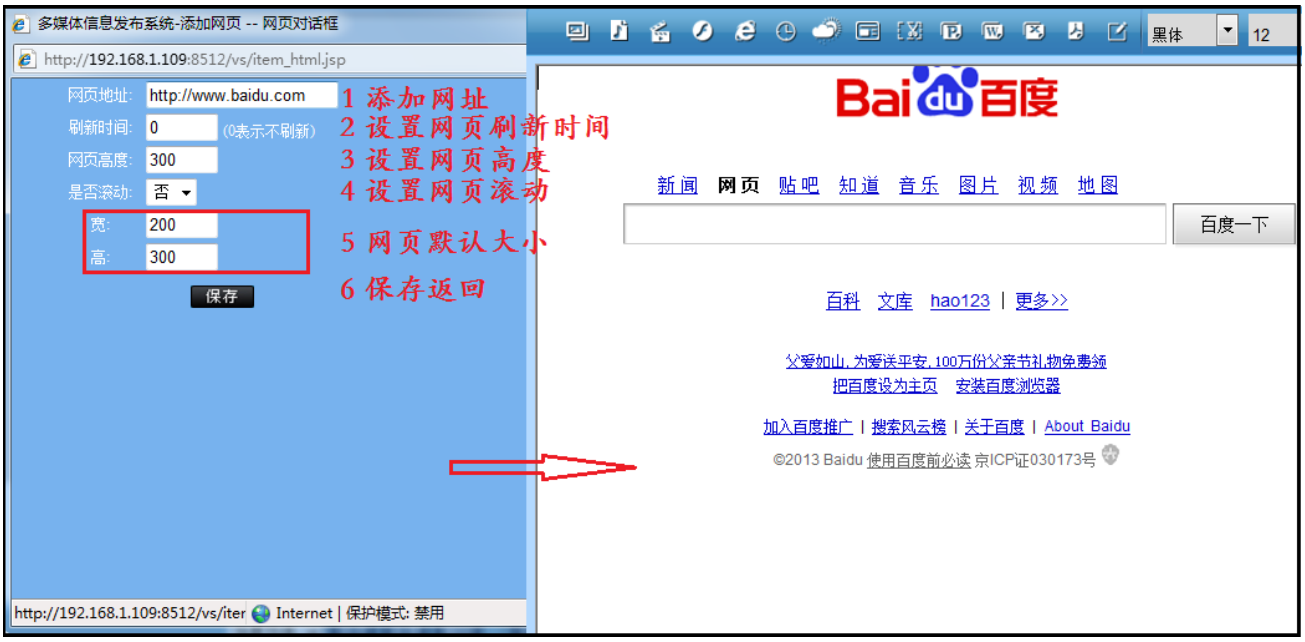
（1） 选择 flash：选择需要的 flash 素材

（2） 默认大小：添加的 flahs 默认大小、可在编辑节目制作窗口中任意更改

（3） 添加 flash：将选择的 flahs 添加到节目中

（4） 新增 flash：当需要添加的 flash 素材没有时、可直接从此处上传 flaah 素材

5. 添加网页



（1） 网页地址：添加网址，当和第三方系统对接时、只需要将第三方系统制作的网页添加到这里即可。例如：汇率系统、排队叫号系统、会议预定系统等

（2） 刷新时间：设置时间后、播放的网页会间隔时间去网站同步更新一次

（3） 网页高度：不同网页播放时高度不同，如果播放出来有部分不显示、可设置此高

度、高度大于 300 以上。

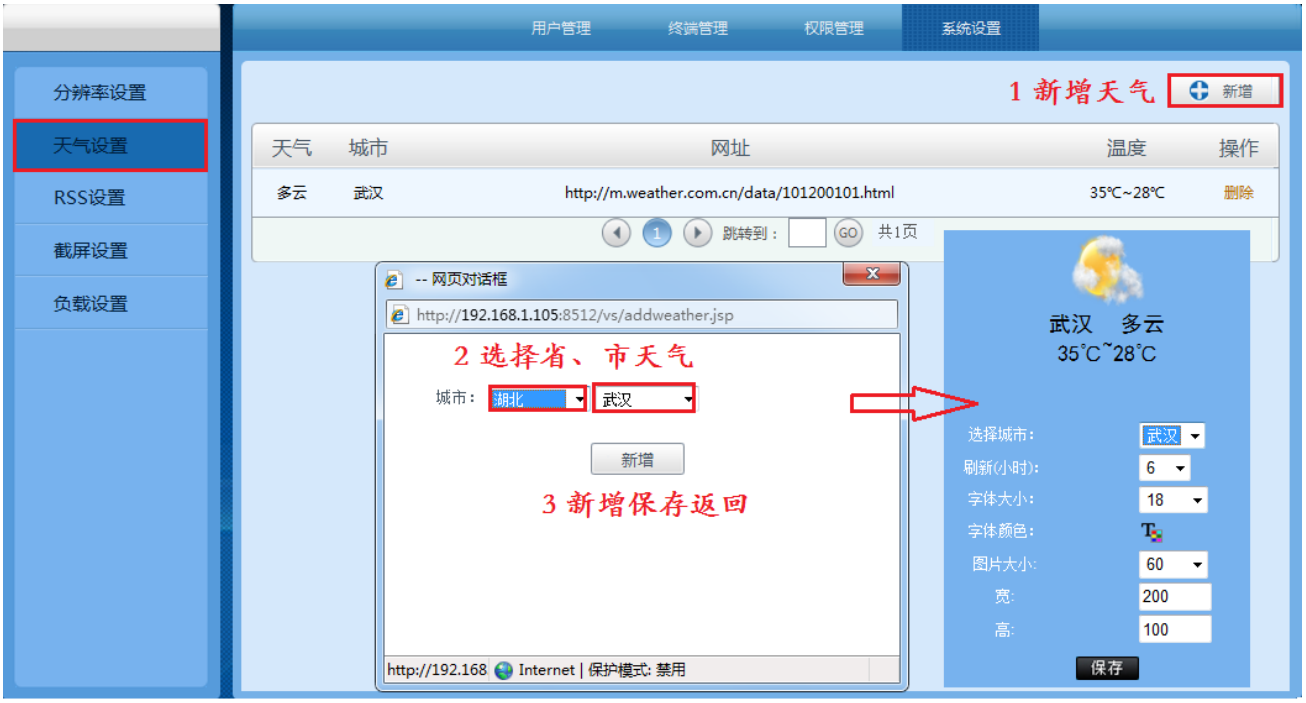
（4） 是否滚动：当网页页面比较大时、不能完全播放，可设置滚动播放

（5） 宽 高：设置网页的宽高度、此处为默认，可在节目编辑窗口中拉伸修改大小

（6） 保存：保存返回按钮

6. 添加天气

（1） 新增城市天气:“系统管理”——“系统设置”——“天气设置”——“新增”——“输入需要播放的城市”——“新增”（保存返回）



（2） 添加天气

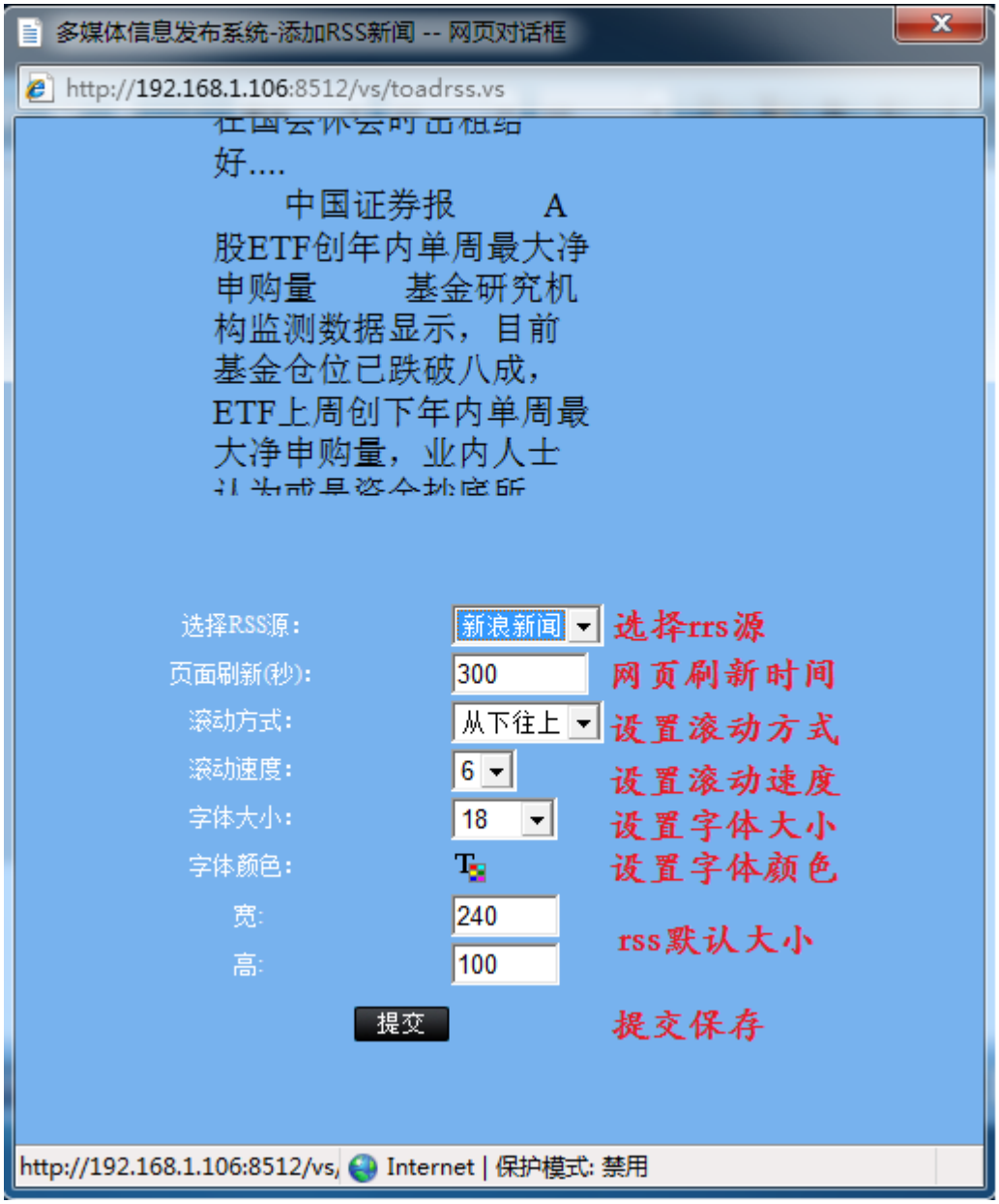

多媒体信息发布系统-添加天气 -- 网页对话框
http://192.168.1.109:8512/vs/toadtq.vs
上海 小雨
24℃-26℃
实时天气
选择城市:
上海
选择城市天气
刷新(小时):
6
选择刷新时间
字体大小:
18
选择字体大小
字体颜色:
字体颜色
宽:
200
高:
100
天气大小
提交
http://192.168
Internet | 保护模式: 禁用

7. 添加 rss


用户管理
终端管理
权限管理
系统设置
①
新增
③
分辨率设置
天气设置
RSS设置
②
截屏设置
负载设置
地址
内容
刷新时间
名字
操作
跳转到:
GO
共1页
新建用户 -- 网页对话框
http://192.168.1.109:8512/vs/addrss.jsp
名称:
新浪新闻
填写rss名称
刷新时间:
5
设置刷新时间
地址:
http://rss.sina.com.cn/new
填写rss源地址
新增
保存返回
http://192.168.1.109:8512/vs/ad
Internet | 保护模式: 禁用

（1） 新建 rss：操作如上图所示，选择“系统设置”——“rss 设置”——“新增”填写 rss 名称、设置刷新时间、滚动方式、滚动速度、字体大小、字体颜色、宽高度等

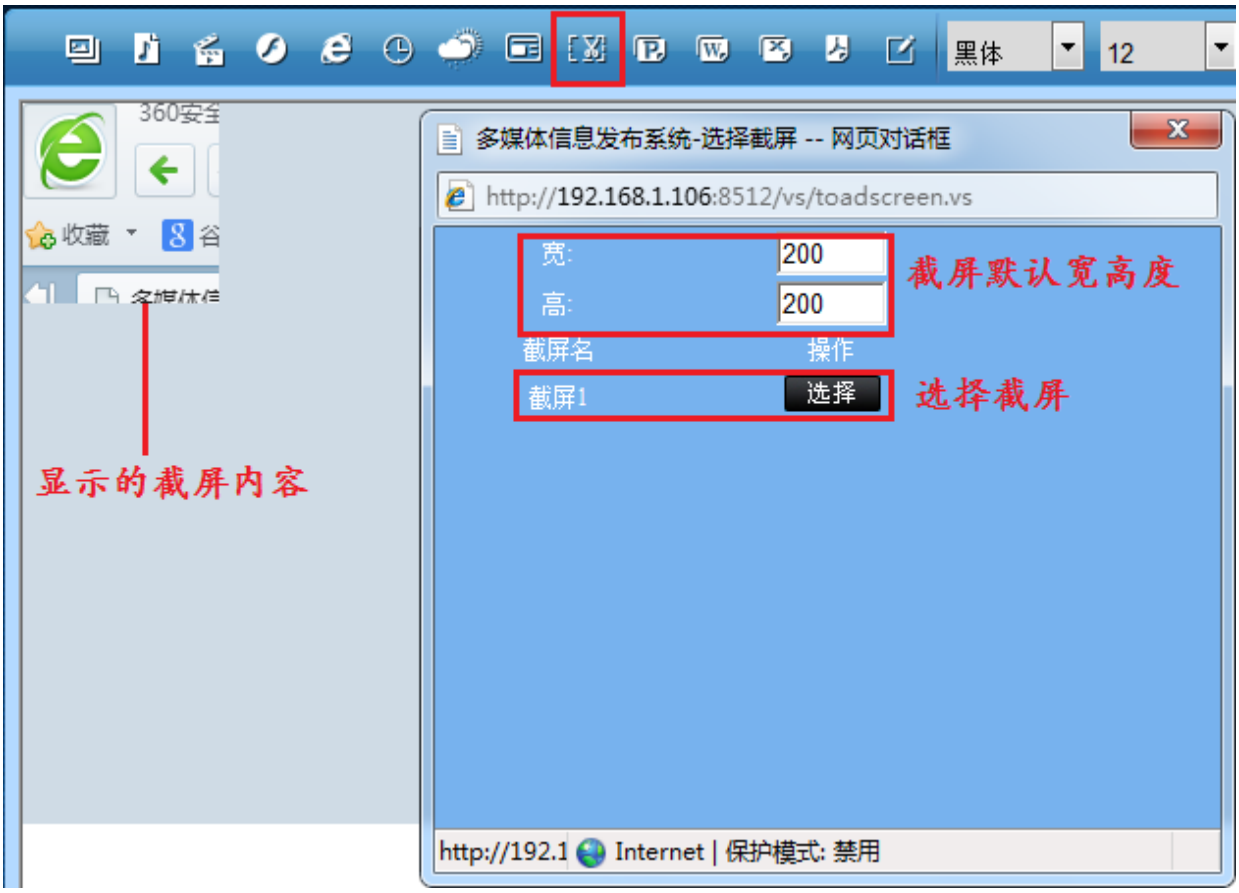
（2） 添加 rss:



8. 添加截屏

（1） 新增截屏：进入“系统设置”——“截屏设置”——“新增”——“选择截屏坐标位置”

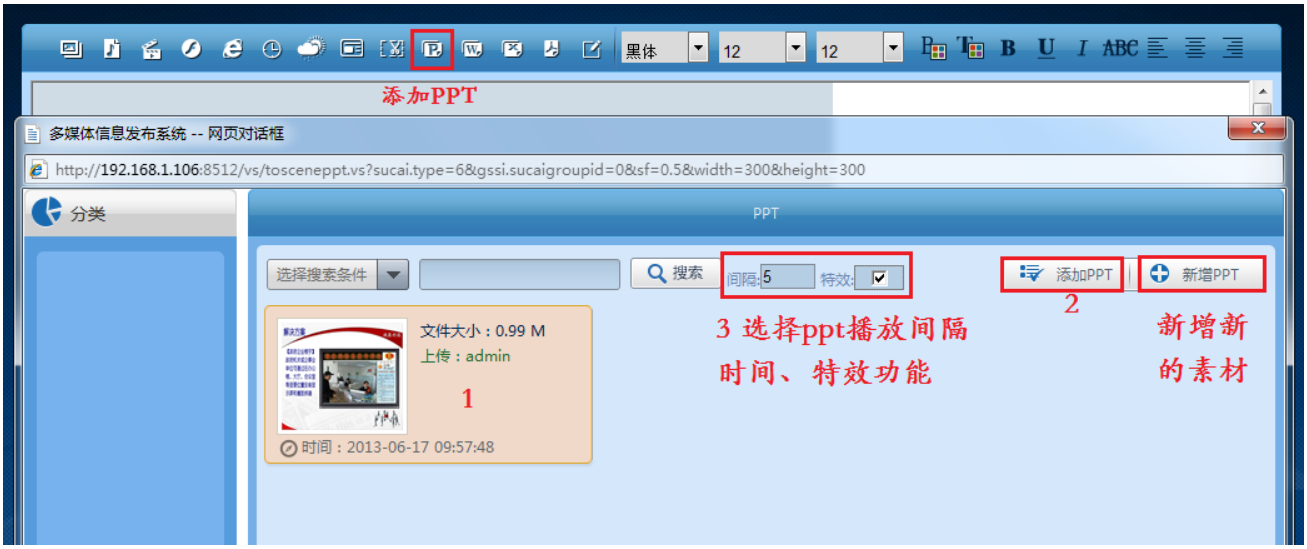
（2）添加截屏：



9. 添加 ppt

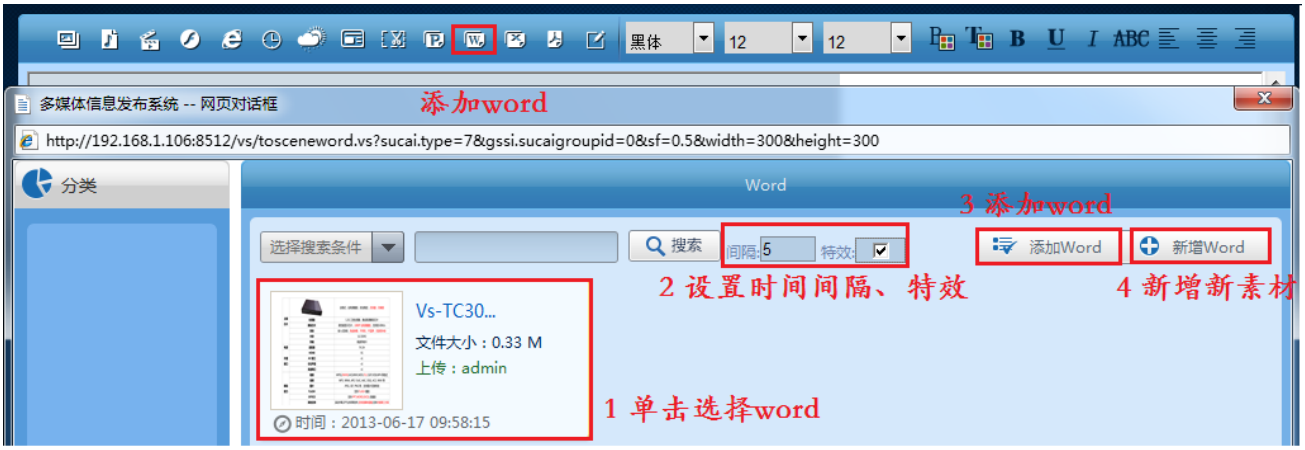
进入添加 ppt 窗口、选择所需要播放的 ppt 素材、然后点击添加 ppt 即可、可设置

ppt 播放时间间隔（单位秒）、可以设置 ppt 播放是否具备特效、可添加新 ppt 素材



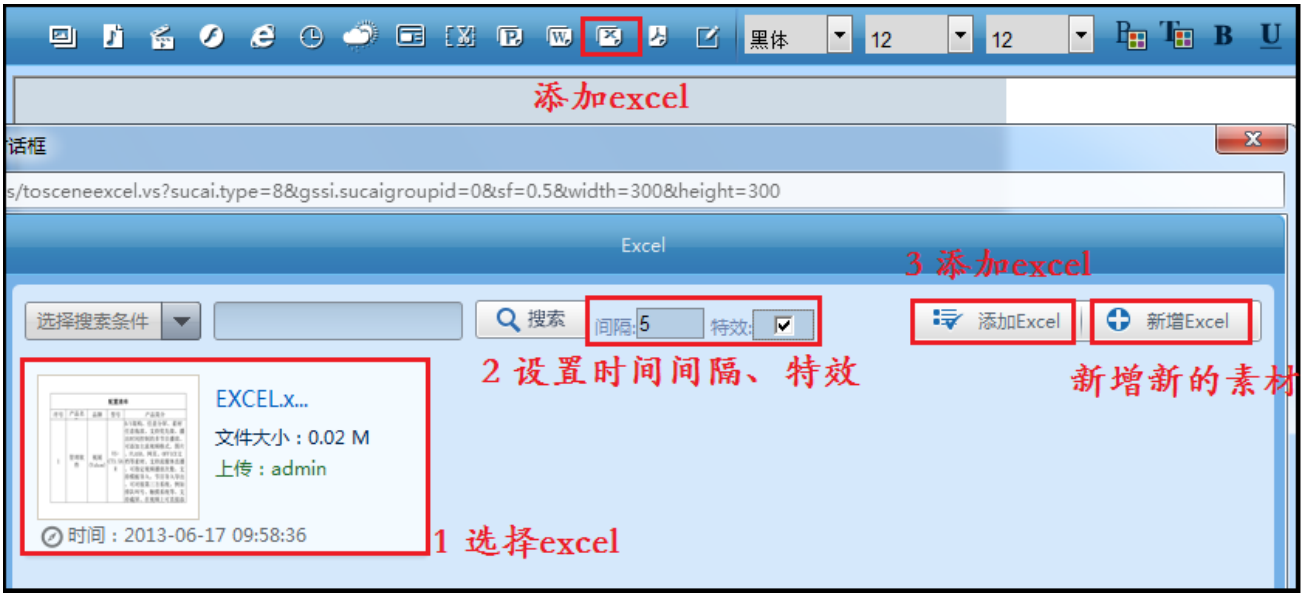
10. 添加 word

进入添加 word 编辑窗口——设置时间间隔、特效——添加 word（新增 word：添加新的素材）

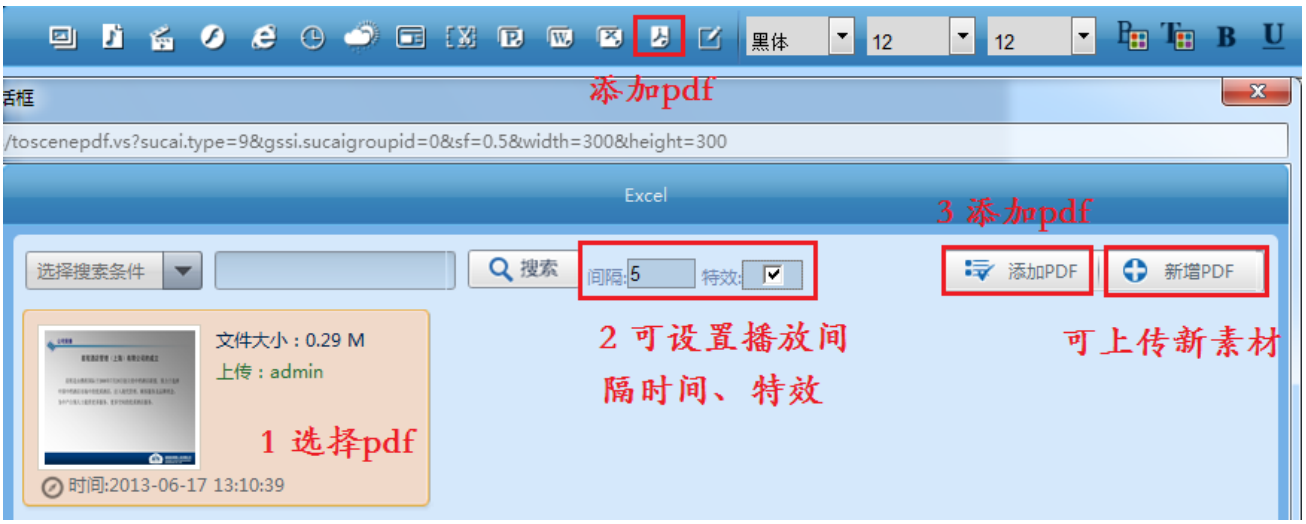


11. 添加 excel

系统可以播放 excel 文档、进入 excel 编辑窗口、选择所需要播放的 excel 文档素材——选择“添加 excel”即可、可以设置播放间隔时间、特效功能等。



12. 添加 pdf



13. 添加文本

文本可以添加静态和动态文本、动态文本可设置上下左右四个方向滚动。可设置文本的滚动速度、文本框大小等。如下图所示

6.3 全选：选择所有节目

6.4 修改：可直接修改编辑节目

6.5 复制：将制作好的节目复制成新的节目、相当于另存为节目。

此功能可将节目另存为新节目：选择需要复制的节目——点击“复制”——填写新节目名称然后点击“复制”

6.6 删除：可删除制作好的节目

6.7 导出节目：将制作好的节目导出、并且导入到终端之间播放，此功能可用作单机播放是将整个节目完整导出复制到终端播放器 SD 卡目录下的 off 文件夹里面，终端机会自动识别这个节目，并且播放。此时 off 文件夹下的节目优先播放，不播放发布的节目，如需要播放发

布的及时节目，需要在软件里面清空删除终端节目。

或者通过 u 盘自动导入节目到播放器、先在 u 盘根目录下建一个 vsfile 文件夹、然后将导出节目压缩包解压后里面的所有文件拷贝到 vsfile 文件里，将 u 盘插入播放器 usb 接口后，重启播放器等待启动后正常播放导出的节目，即自动导入成功。

6.8 搜索：可输入节目名称字符搜索节目

第七章、终端发布

此模块分 9 个区域功能：发布节目、轮播节目、插播节目、发布通知、同步时间、开关设置、清理终端、参数设置、执行结果

1 发布节目

发布节目：为立即发布模式、只要终端现在没有插播和轮播，发布节目会立即播放。选择需要发送的终端——然后选择“节目发送”——选择需播放节目——“提交发送”

全选：可选择所有终端

清理发布的节目：可清除终端的“发布节目”

查看记录：可以查看终端发送的节目是否成功下载、查看终端的播放计划列表等信息

返回：返回上一级

2 轮播节目

轮播节目：轮播节目是周期播放模式、即可以将节目定时在每天或者每周几什么时间点开始播放到几点结束。具体操作如：选择要发送的终端后——点击“选择节目发送”——选择轮播时间段——“提交发送”即可

全选：选择全部终端

选择节目发送：选择需要发布的节目

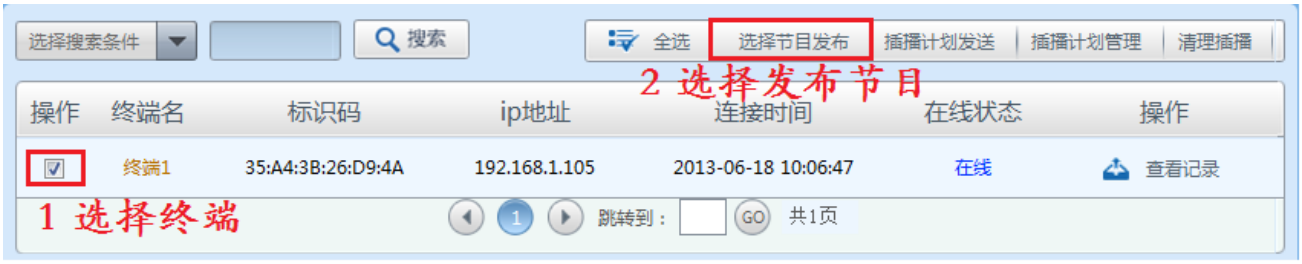
清除轮播：清楚终端播放器中的所有轮播节目

轮播计划发送：可以预先制作好轮播放松计划、发送时只需发送此轮播计划即可

轮播计划管理：编辑轮播计划、可预先设置好轮播计划、然后将计划发送到终端、终端会根据计划来播放节目。操作如下：进入“轮播计划管理”——“轮播计划管理”——“新增”然后添加轮播时间段、选择轮播节目、填写计划名称，如下图所示

3 插播节目

在临时和紧急没有预先编排计划任务时、可以再正常播放计划时，在一段时间内插入其他的节目计划。一次可以插入多组插播计划，可以将时间插播在将来的时间段。具体操作如：选择要发送的终端后——点击“选择节目发送”——>



“选择插播时间段”——“选择需要发布的节目”——“提交发送”

全选：选择所有终端发送节目

插播计划发送：可预先制作好插播计划、发送时只要发送此计划即可

插播计划管理：可以添加、删除管理插播计划

清理插播：清楚终端所有的插播节目

4 发布通知

可以通过软件后台发布及时的新闻通知信息、默认在屏幕最下方从右往左滚动播放，可对通知进行管理编辑等、操作如下：进入“发布终端”——选择需要发布的终端——“发布新通知”

发布已有通知：可以直接发布已经存在的通知

通知管理：可对通知进行修改、删除操作

标题：新增通知标题

内容：输入通知内容

提交发送：发送编辑好的通知信息

取消通知：删除终端的通知信息

5 同步时间

可手动同步终端和服务器之间的时间、终端和服务器时间不准会影响节目播放、如果时间不准时手动同步终端时间即可。“选择终端”——“同步时间”（发送终端同步时间）

6 开关设置

重启：可以通过后台重启终端

唤醒：通过后台远程开机

待机：通过后台远程关机

取消计划：可删除终端的定时开关机时间计划

定时开关机：通过软件后台可以设置终端定时开关机、可设置多轮。操作如下：选择“定时开关机”——选择“选择每天和周几”——选择“开关机时间”——“添加定时计划”——“提交发送终端”

7 终端清理

清理终端：是将终端所有播放内容清空、操作如下：“选择终端”——“终端清理”

8 参数设置

参数设置：可设置终端的参数信息、选择“终端”——进入“参数设置”

9 执行结果

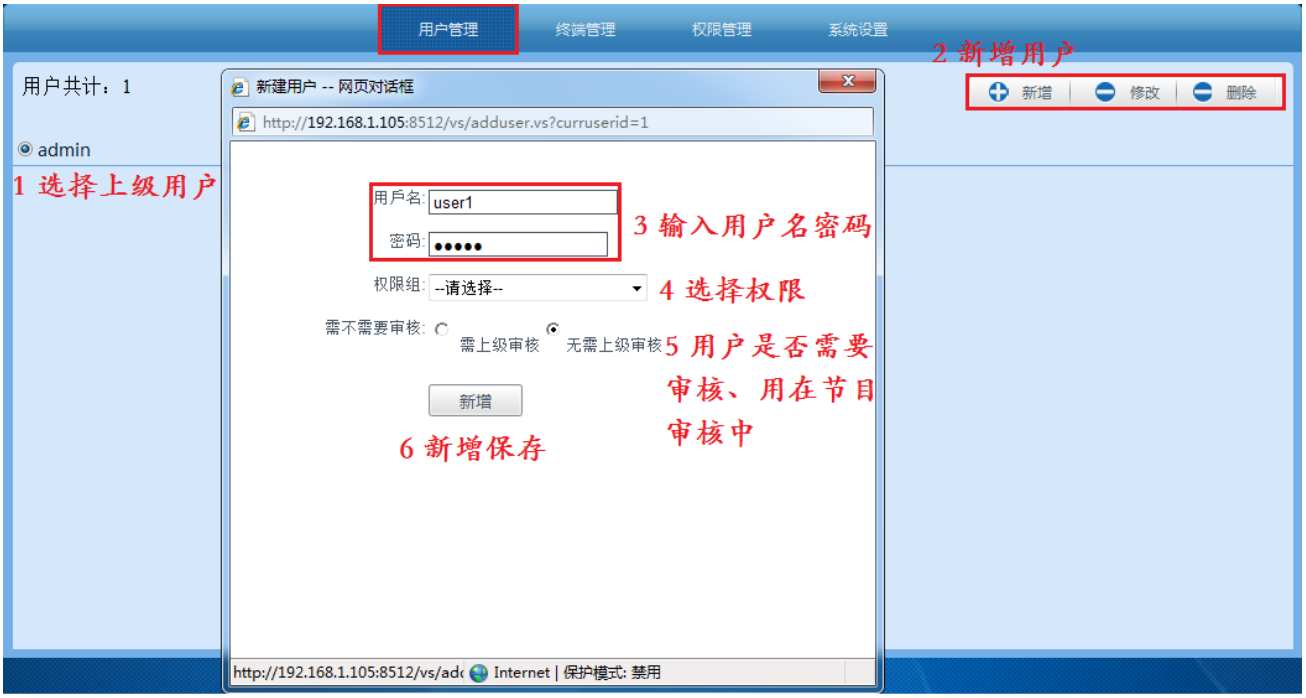
查看执行结果：可以查看软件后台发送到终端所有的指令是否成功被终端接收、执行，根据执行结果可以快速判断发送指令情况。

第八章、 系统管理

系统管理分为 4 个模块：1 用户管理、2 终端管理、3 权限管理、4 系统设置，主要对系统用户、权限、终端进行管理设置。

1. 用户管理

用户管理：可新增/删除用户、可设置用户权限、设置用户是否需要上级审核等



操作如下：1“选择上级用户”——2 点击“新增”——3“输入用户名密码”——4 选择“权限组”——“设置审核”（默认为无审核）——“新增”

新增用户时，必须选择上级用户，权限组权限需要在后面的“权限管理”模块中设置。“需不需要审核”如果设置了需要审核、此用户做的节目需要上一级审核才能使用、并且发布节目也需要审核后才能发布。

2. 终端管理

分配终端：给用户分配终端、用户就可以在权限范围内去控制终端，如果用户未分配终端则不能控制此终端、操作如下：“选择用户”——“选择需要分配的终端”——“分配终端”（取消分配：取消用户对终端的管理权限）

新增终端：可以直接手动添加新的终端、只需要知道终端的标识码、ip 地址即可、此功能在二级服务器中用到，需要在一级服务器手动添加终端。

删除终端：将终端从软件后台删除

修改终端名：可修改终端名称

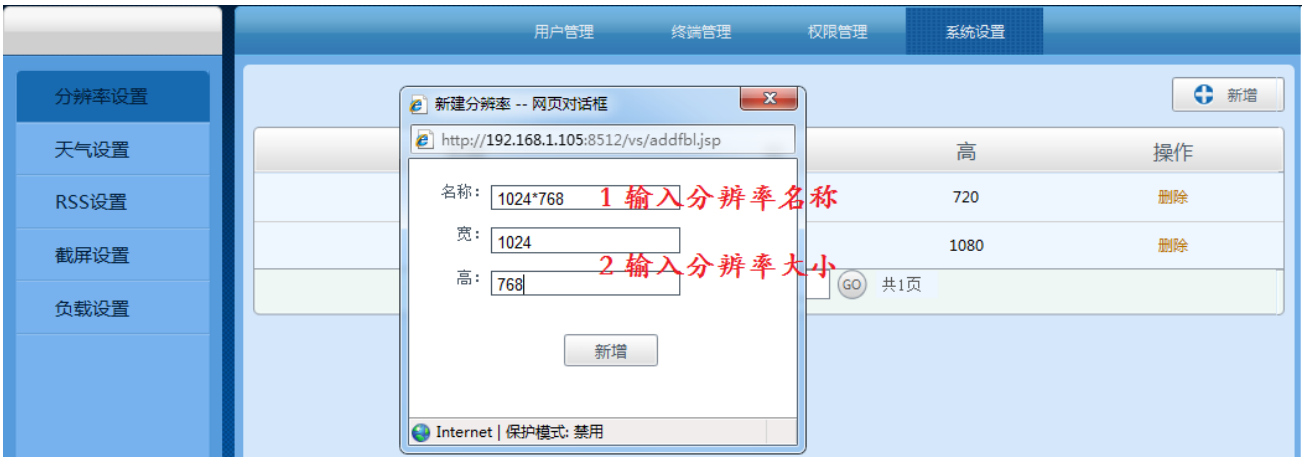
3. 权限管理

权限管理：可以添加用户权限组、修改、删除用户权限组等，操作如下：1“新增”——“输入权限组名称”——“添加权限”——“新增”保存返回

4. 系统设置

系统设置模块：分为 5 个小区域、(1) 分辨率设置、(2) 天气设置、(3) RSS 设、(4) 截屏设置、(5) 负载设置。

(1) 分辨率设置:



点击“新增”——“输入分辨率名称”——“输入分辨率大小”

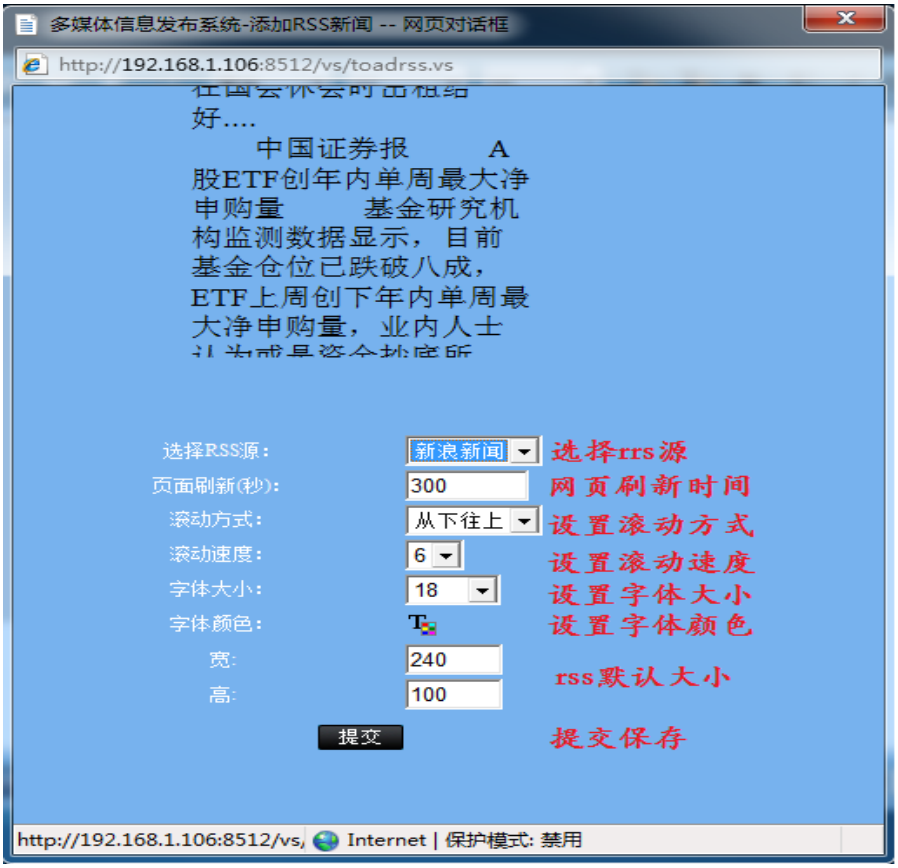
（2） 天气设置

新增全国城市天气预报、如下图所示：(1)“新增”——“选择城市天气”——“新增”保存返回

（3） RSS 设置

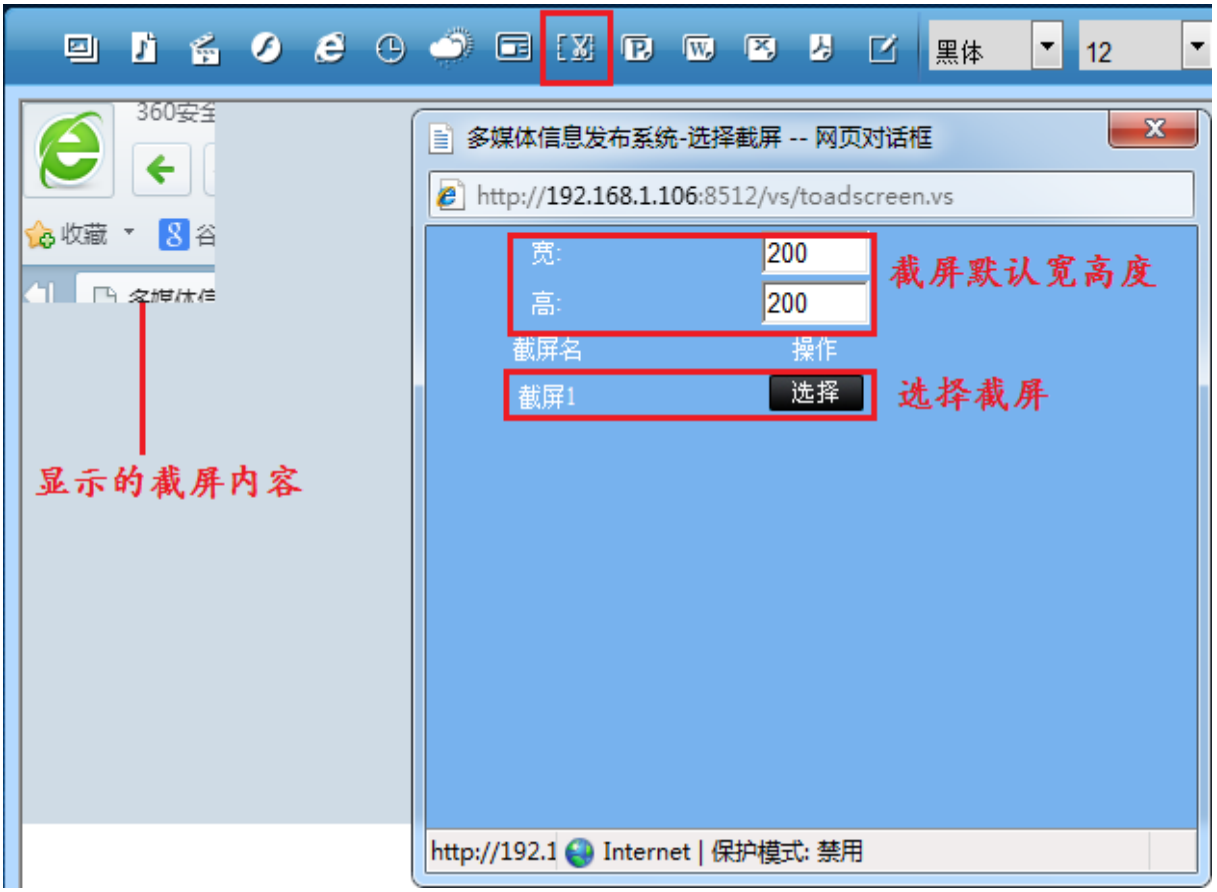
新建 rss：操作如上图所示，选择“系统设置”——“rss 设置”——“新增”

填写 rss 名称、设置刷新时间、滚动方式、滚动速度、字体大小、字体颜色、宽高度等



（4） 截屏设置

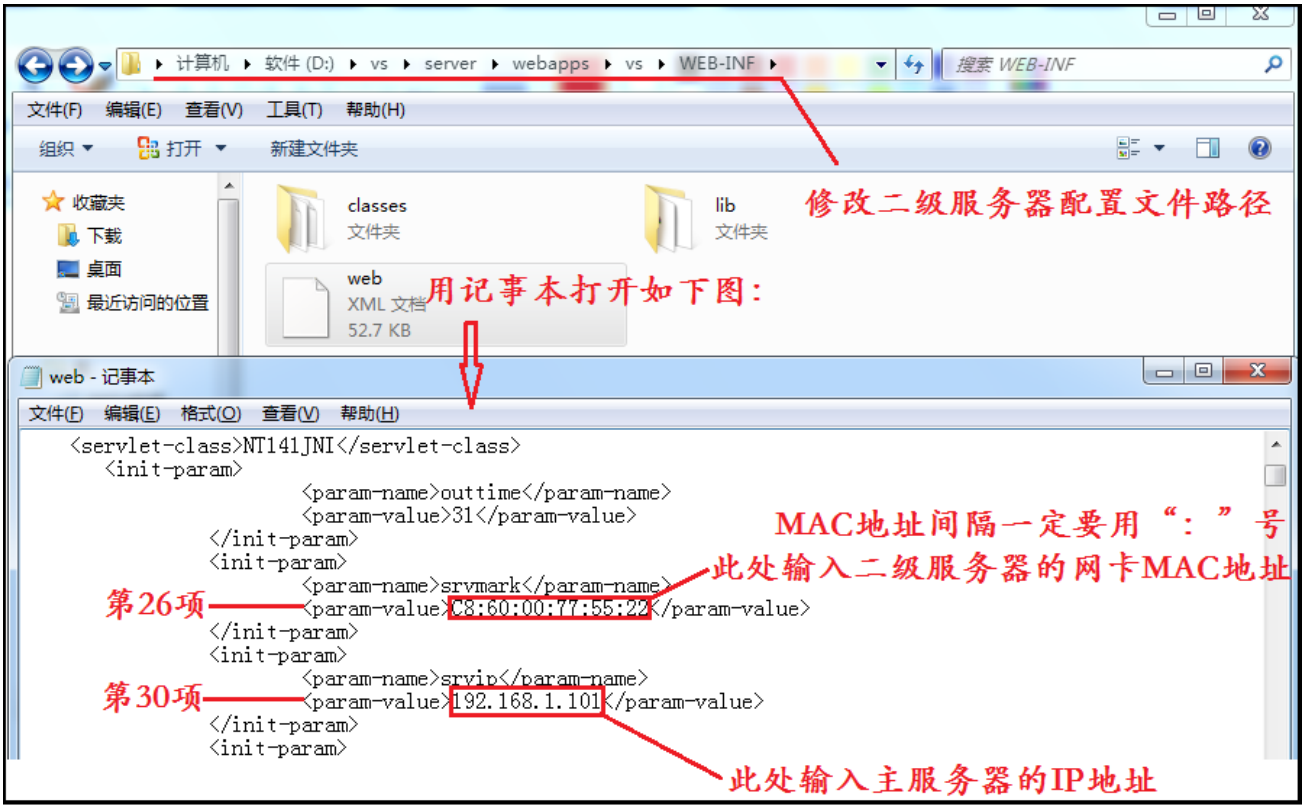
截屏：可以实时的截取服务器桌面当前显示的画面，操作如下：“新增”——“选择截屏坐标位置”、截屏间隔时间、截取图片类型等



（5） 负载设置

负载设置是为了解决因终端众多影响带宽问题、用负载设置添加二级服务器可以让多服务器负载分发、这样大大的解决了因终端多数据量大带来的瓶颈问题，启用负载二级服务器后、可以将终端分配到二级服务器去下载内容、从而起到负载分发功能。操作如下

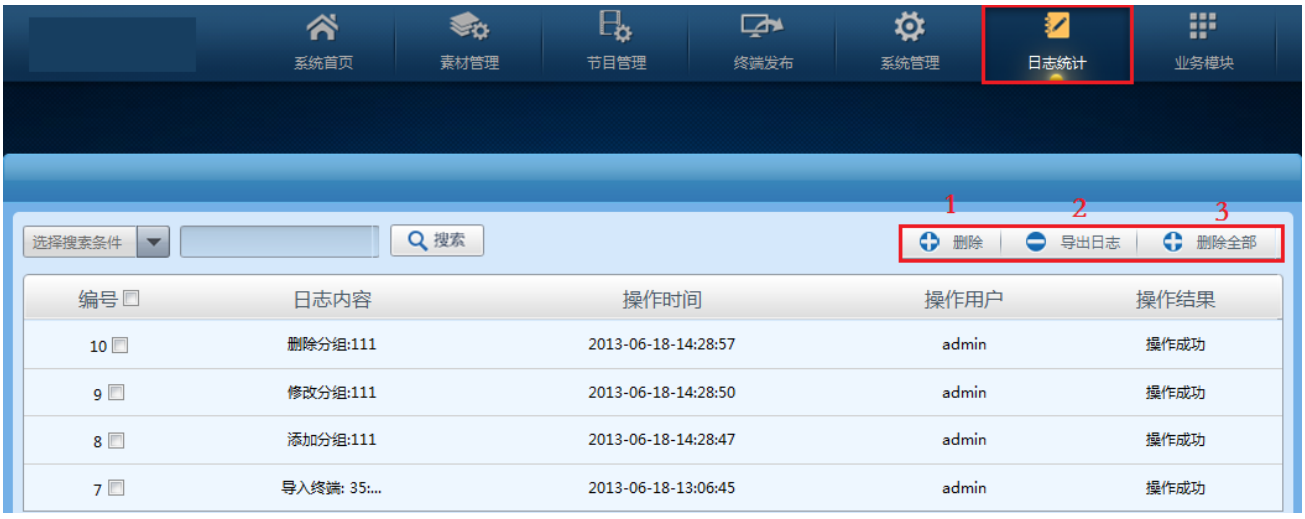
二级服务器设置：进入二级服务器的配置文件、修改对应 MAC 和主服务器 IP 地址。（此 MAC 地址为二级服务器本身网卡地址、IP 地址为主服务器公网 ip，如果主服务器为局域网络服务器、则需要将此服务器映射到互联网，输入互联网公网 ip）



一级服务器设置（主服务器）：首先添加二级服务器点击 “新增”——如下图所示：输入二级名称、ip 地址、标识码（二级服务器 MAC 地址）

第九章、 日志统计、

日志统计：可以通过此功能查看用户在什么时间做了什么操作、操作指令是否成功执行等信息，可对日志进行删除、导出，导出是以 Excel 表格方式保存的。



第十章、 业务模块

未开通模块