陇东学院

毕业论文（设计）

论文题目： 基于 STM32 环境下的驾校管理系统设计与实现

学　　院： 信息工程学院

年　　级： 2016 级

专　　业： 物联网工程

学　　号： 2016631124

姓　　名： 王富国

指导教师： 李芳芳

完成日期： 2020 年 4 月 20 日

作者声明

本毕业论文（设计）在李芳芳老师指导下由本人独立完成，没有剽窃、抄袭、造假等违反道德、学术规范和其他侵权行为。对本论文（设计）的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。因本毕业论文（设计）引起的法律结果完全由本人承担。

毕业论文（设计）成果归陇东学院所有。

特此声明。

作者专业　：　物联网工程

作者学号　：　2016631124

作者签名　：

年　月　日

目　录

基于 STM32 环境下的驾校管理系统设计与实现

王富国

（陇东学院　信息工程学院）

摘　要：现代社会是基于是互联网的时代，从当前物联网发展的社会潮流来看，这是一个永恒不变的话题。驾照已经成为了我们人生中必不可少的证件之一，是我们安全出行的保证。随着“考证”人数的越来越多，驾校遍布，种类繁多，业务杂乱，驾校的人员管理越来越乱。目前主流的驾校管理系统在一定程度上减轻了人力的负担，但是还存在很多问题，例如预约练车系统容易崩溃、系统页面卡死，打卡定位计时不准确，学员信息存储不够安全，系统可维护性差等，仅仅依靠 Web 端单一的管理方式，很难均衡。

本文基于物联网平台与 B/S 架构相结合设计了管理系统，将目前火热的物联网平台技术融入到驾校实际的业务场景下，一方面充分展示应用了物联网技术在当今社会场景下的应用，以及通过物联网技术来解决驾校系统中的一些由于纯软件平台技术缺陷造成的功能低下和系统的不够智能化问题，另一方面也是学习和了解物联网技术与软件平台结合下的应用与融合。本系统充分体现了将物联网技术应用到驾校这样一种社会化环境下的技术需求以及物联网平台与传统软件端数据交流交互起到了引导作用，由于物联网范畴强大，本系统可根据需求随时做软硬件结合的拓展。

关键词：物联网；STM32；HTTP 通信；射频识别；软硬件结合

Design and Implementation of Driving School Management System Based on STM32

Wang Fu guo

(School of Information Engineering，Longdong University)

Abstract: Modern society is based on the era of the Internet. Judging from the current social trend of Internet of Things, this is an eternal topic. The driver's license has become one of the indispensable documents in our lives and is a guarantee for our safe travel. With the increasing number of "testing", driving schools are all over, there are many types, and the business is messy, and the management of driving school personnel is becoming more and more chaotic. At present, the mainstream driving school management system has reduced

the burden of manpower to a certain extent, but there are still many problems, such as the booking car training system is easy to crash, the system page is stuck, the punching positioning timing is not accurate, the student information storage is not safe enough, and the system can be maintained Sex is not high, just rely on a single management method on the Web side, it is difficult to balance.

This article designs a management system based on the combination of the Internet of Things platform and the B / S architecture. It integrates the current hot Internet of Things platform technology into the actual business scenario of the driving school. On the one hand, it fully demonstrates the application of the Internet of Things technology in today's social scenarios. , And use the Internet of Things technology to solve some of the problems of low function and insufficient intelligence in the driving school system due to technical defects of the pure software platform. On the other hand, it is also to learn and understand the application and integration of the combination of Internet of Things technology and software platform. . This system fully embodies the technical requirements for applying the Internet of Things technology to a driving environment such as a driving school and the data exchange interaction between the Internet of Things platform and the traditional software terminal. Due to the strong scope of the Internet of Things, this system can be used at any time according to the needs. Expand the combination of hardware and software.

Key words: Internet of Things; STM32; HTTP communication; radio frequency identification; combination of hardware and software

1 绪论

人工智能，简称 AI，是指人类制造出来的机器表现出学习，演绎、推理、解决问题等的智能。[1]物联网在一定层次上起到了万物互联，通过硬件传感器设备收集数据的作用。人工智能与物联网是相辅相成的。在物联网大数据的支撑下，人工智能才能更加的智能便捷化。本课题研究的管理系统，在物联网的加持下，将进一步人性化驾校的管理。

1.1 概念解析

1.1.1 物联网概述

物联网在维基百科中是这么定义的：物联网（The Internet of Things，缩写 IOT），又称 IOT 技术，是互联网、传统电信网等的信息载体，让所有能行使独立功能的普通物体实现互联互通的网络。物联网将现实世界数字化，一体化，万物互联，应用十分广泛。[2]物联网拉近了世界万物间的距离，统整物与物之间分散的信息，主要应用领域有以下方面：智能环境（家居、办公、工厂）领域、物流运输领域、工业制造领域、医疗健康领域、个人以及社会公共安全领域等。[3]

1.1.2 STM32 系列 MCU

STM32 是使用了 ARM 核的 MCU，属于 32 位微控制器。ARM 公司在处理器 ARM11 以后将产品改用 Cortex 命名，并分为了 A、R 和 M 三类，全世界超过了 95%的手机和电脑都使用的是 ARM 架构的芯片。STM32F103 是首款基于 ARMV7-M 的精简指令集处理器，具备很高的代码运行效率。STM32 系列芯片基于高性能、低成本和低功耗的嵌入式 ARMCortex-M3 内核设计，按照性能可分为 F103 系列增强型和 F101 系列基本型。Cortex-M 内核无需操作系统驱动，可以向单片机一样使用 Keil 进行 C 语言的开发，极大的减少了开发工作量。[4]新一代的嵌入式处理器，除了具备基本的 ARM 体系结构外，还降低了开发平台的成本，大大缩减了引脚额数目，减少了系统功耗，同时还提供了强劲的计算性能和先进的中断响应系统。丰富的单片机硬件资源，使得 STM32F103 系列微控制器在多种领域被广泛使用。

1.1.3 Java EE 体系概述

Java EE，Java 平台企业版，原名 J2EE，2018 年 3 月更名为 Jakarta EE。[5]它是在 SUN 公司的领导下，多家公司参与共同制定的企业级分布式应用程序开发规范。是目前世界上主流的分布式应用平台解决方案。

该平台包含有 JDBC（Java 数据库连接）、Servlet（Java Servlet API）、WS（Web

Service）、JTA（Java 事务 API）等相关解决方案。

1.2 研究背景及意义

驾校，全称驾驶人训练班或驾驶学校，英文名称：Driving School，是目前世界各国为机动车驾驶人提供的统一培训和教授练习驾驶技术的场所。驾校的开设现象，在我国普遍流行，随着大众化消费水平的提高，国家经济的快速发展，机动车产业迅速增长，在 “人人有车”的社会前景下，驾校行业迅速扩大，服务内容也有所拓展，培训形式多种多样，最近几年兴起的练车预约、模拟驾考尤为流行，这些相对智能设备的辅助，让培训人在练车、学习等方面轻松了不少，也减轻了驾校的人力资源。

相对智能化的设备，例如语音模拟练车系统、灯光语音模拟考试系统等，相比传统纯人工监督的方式，更加的先进、便捷、智能化，然而随着人工智能的进一步发展，扩大物联网的范围，提供更多的数据支持，才能更好地利用互联网为社会造福。

1.3 国内外的研究现状及发展趋势

纵观国内外，属我国驾驶人培训行业形式最为严峻，我国交通管理条例全面，对驾驶人的要求严格，其他国家在驾驶人培训管理方面现存资料公开很少，本文将以我国为例分析驾校管理系统现状及发展趋势。我国在驾驶人培训方面机构条例都很完备，根据目前市面上存在的驾校管理系统，我将其分为了软件现状和硬件现状。

软件现状，软件形式的驾校管理系统，并且已经处于正常服务状态的系统在市面上有很多，这些系统都是单一的 B/S 架构模式的网络服务预约系统，通过编码方式对驾校的人员信息数据做了汇总，并且已经具备一整套的系统流程，比如网点报名、学员管理、练车预约、考试预约等功能，相对来说，功能完备，大大减轻了驾校管理方面绝大部分人工处理数据的工作量，但是也存在许多不足，比如许多系统技术落后，采用是陈旧的技术，JSP 服务端页面技术等，存在系统可移植性差，维护性不高，响应速度过慢等问题，而且 JSP 作为一种服务器端语言，应用到具体的业务场景下，大大增加了服务器的开销，给服务器造成了不小的压力，这项技术在当前的前后端分离开发体系下已经被淘汰。

硬件现状，“互联网+”概念的提出和物联网技术的发展，已经有一部分服务企业推出了物联网技术应用的驾校管理系统，例如安徽展瑞信息科技推出的驾校数据监控系统《一种基于物联网的驾校综合监控管理系统》，还有苏州木兰电子科技的驾校智能识别系统《驾校智能识别系统驾校考试管理有源电子标签》，这些系统充分运用了物联网平台的技术实现，GPS 定位，车速监测等。[6]并为用户以及企业管理人员提供了可视化的页面交互系统。

伴随着物联网技术的逐步发展，为了实现网络信息办公化，充分实现网络一体化服务替代人工劳动，物联网技术将会在驾校管理这一方面有着更为先进的突破。目前驾校综合业务的拓展和系统的可维护性依然停留在 B/S 架构，单纯网络服务的领域，应用到物联网场景下的技术很少，而且 Web 页面采用的是 JSP 技术，这种服务器端的页面框架已经成为过去式，对服务器的负荷重，维护成本高，目前 H5 已经占领了潮流，应用广泛，前后端分离开发模式已经占据了企业级应用的顶端。[7]借助物联网平台，再来研究驾校的综合管理，使用软硬件结合的方式，将在一体化办公，智能服务的领域更进一步。相信不久的将来，随着物联网技术的进一步发展，驾校的综合化、一体化管理服务将会变得更加人性、更加便捷。

1.4 研究本课题的目的和基本内容

目前社会上具备物联网技术并且是软硬件结合的驾校服务系统都有一个技术共性，物联网平台与 Web 服务交互数据采用的是 MQTT 协议的物联网数据上报平台，根据计算机网络中 OSI 七层网络模型的概念，物理网平台的数据可以采用 HTTP 协议直接与 Web 服务器交互，并不需要先通过 MQTT 协议将数据上报，然后转发到 Web 服务器。在七层网络模型中，网络数据的传输链路部分只需要一层即可完成相同的需求，对于绝大部分驾校场所，减少数据上报平台的资源投入，是非常有必要的。

本课题研究的驾校管理系统，主要采用物联网平台与 Web 网络服务平台相结合的方式，在解决驾校日常运转各项业务，提高员工工作效率的前提下，更好的为广大的驾校管理者服务。本课题主要研究在 STM32 开发板环境下，RFID 射频识别技术采集数据，经过 ESP8266 网络设备通过 HTTP 协议将数据转发给 Web 服务器进行数据分析处理，响应给 ESP8266 设备，再由 STM32 芯片处理数据，展示在其它硬件设施（LED 灯光、蜂鸣器等）上。摒弃现有平台不必要的数据上报处理过程，节省资源的同时，还能达到同样的目的。

本文系统中物联网平台基本内容如下：

（1） 采用 C 语言编码，驱动 STM32 芯片指挥各个硬件模块合理化工作。

（2） 通过 RC522 型号的 RFID 模块完成读写卡工作，来操作卡片内存在的数据信息。

（3） 使用 ESP8266 型号的网络模块将物联网平台的数据通过 HTTP 协议转发到 Web 服务器端进行数据交互。

（4） 通过板载按键，完成模式切换以及辅助完成其余功能。

（5） 通过 LED 作为系统提示，起到指示系统状态，提示用户的作用。

（6） 使用串口代替屏幕，打印输出系统调试信息以及对用户的提示语句。

（7） 使用 STM32 延时中断异常配合其它硬件完成功能。

本文系统中 Web 网络平台的基本内容如下：

（1） 服务器端采用 Java 语言开发的主流 Web 框架，由 Spring Boot 快速集成式搭建。

（2） 前后台分离式开发部署，后端使用 Spring Boot、Spring MVC 和 MyBatis 整合（简称 SSM 框架），前端使用数据驱动型框架 Vue 开发。

（3） 服务器端使用 maven 一键构建、运行、打包和部署。

（4） 公网服务器使用 nginx 或者 apache 服务器做分离式部署。

2 系统总体设计方案

2.1 系统需求分析

该系统主要包括由 C 语言编写的物联网平台、Java 语言编写的 Web 服务端和由前端 vue 框架搭建的 Web 网页端，还需要数据库来存储系统数据。

驾校是一个服务型机构，主要为培训人提供培训、考试服务，STM32 单片机的组合，能在驾校安全管理，人员控制方方面面起到辅助管理的作用，还能帮助驾校管理人员采集数据，减轻工作人员的劳动力。具体业务流程如图 1 所示。

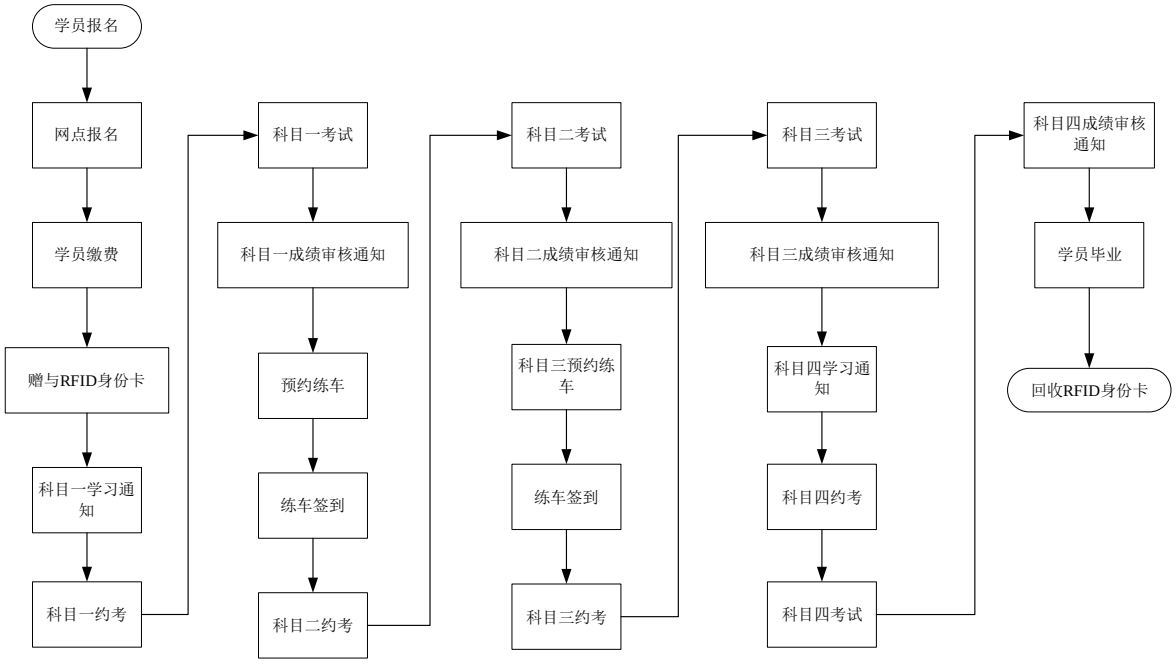


图 1　业务流程图

流程图中展示的大部分驾校数据信息处理的业务，在目前的各类驾校管理系统中

已经较为成熟，且本论文实现的处理方式也基本类似，本文主要讨论驾校数据管理业务结合物联网环境平台的应用，根据这个初衷，需要研究以下内容：

（1） 任何一套系统都离不开数据的支持，在现有驾校管理系统的基础上，融入物联网平台的内容，需要考虑物联网平台与 Web 服务器数据的衔接方式。[8]本论文将使用开源免费的 MySQL 关系型数据库，通过构建物理模型来展示数据的交互过程，在下一章节将会详细介绍。

（2） 本论文采用了物联网平台，需要用到开发板芯片处理数据，RFID 模块采集数据，读取生产厂商内置在射频卡中的卡片编号，由 ESP8266 模块更后台请求、交互和传递数据等，具体的实现流程将会在系统硬件设计章节具体阐述。

（3） 本系统的数据处理以及展示都采用了目前企业级应用开发的前后端分离架构处理，前端展示，后端处理，本论文着重介绍 STM32 在该系统下的应用。

（4） 本论文研究的系统涉及物联网平台、Web 服务器端，Web 数据展示端等，本设计从项目搭建开始就将采用分布式版本控制工具 Git 来管理项目文件。物联网作为实践类科目，大部分功能都需要通过代码来实现。本论文的重点在于研究 STM32 在该系统的应用和数据交互，以及平台要实现的功能，本项目代码需求量大，耗时周期长，在项目搭建初始，就要考虑到代码的版本的迭代，以及代码的维护、拓展等，由于代码维护工作量太大，为了专注于功能的实现，Git 分布式版本控制工具就担当了代码维护的重要角色。

2.2 系统总体架构设计

2.2.1 数据建模和数据库 E-R 图

数据建模是对现实世界中的各类数据抽象的组织起来，确定数据库需要管辖的数据范围和对数据的组织形式等，最终转化为现实中数据库的过程。一套完整的系统设计，必须要有数据库的支撑，对数据的业务处理，是系统运转的核心。本节主要讨论数据字典的建立到转换为物理模型，再到 E-R 关系图。

驾校是一个服务型企业，涉及到的用户主要有三类：学员（Student）、教练（Coach）、以及驾校管理层人员（Administrator）,服务类型主要是以教练和学员为中心的数据服务，三类用户都有权访问驾校系统中的资源，可将这三类用户统一为驾校的用户（User），驾校是提供付费培训的场所,支付费用才能获得对应的服务，因此每位用户都要有属于自己在驾校系统中的账户和密码。学员是驾校的用户主体，这三类人群构成了系统的三种用户角色（Role）。不同角色的用户可以在驾校系统中行使自己的权利，每个用户在系统中有着不一样的访问权限（Privilege），根据以上的逻辑关系，可

以构建用户表（driver_user）的物理模型,对应数据字典如表 1 所示。

表 1　用户表（driver_user）

字段名	数据类型	长度	允许为空	自动递增	主/外键	备注
id	BIGINT	255	NO	YES	P	主键 id
login_id	VARCHAR	36	YES	NO		登录用户名
telephone	VARCHAR	36	NO	NO		电话号码（可用于登录）
id_card	VARCHAR	20	NO	NO		身份证号（可用于登录）
email	VARCHAR	255	NO	NO		邮箱（可用于登录）
username	VARCHAR	30	YES	NO		用户昵称
password	VARCHAR	255	YES	NO		登录密码（初始密码身份证后 8 位）
role	VARCHAR	200	NO	NO		身份角色（默认为学员）
created_time	DATETIME		NO	NO		数据创建时间
enable	TINYINT	4	NO	NO		数据是否有效

系统共有三类用户，教练和学员属于典型的多对多关系，为了在系统中突出这一层关系，需要设计中间连接表来关联这一层关系，学员和教练作为驾校的主要人群，需要管理更多的用户信息，为了更好的实现多对多的关系，需要单独根据用户信息的需求设计学员、教练、管理员的信息表，由于管理员信息不是系统的主体业务，结构相对简单，不再列出，学员信息表（driver_student_archives）和教练信息表（driver_coach_archives）的数据字典分别如表 2 和表 3 所示。

表 2　学员信息表（driver_student_archives）

字段名	数据类型	长度	允许为空	自动递增	主/外键	备注

id	BIGINT	255	NO	YES	P	主键 id
user_id	BIGINT	255	NO		F	用户表主键
RFID_card_id	BIGINT	255	NO		F	RFID 卡片表主键
real_name	VARCHAR	50	YES	NO		真实姓名
birthday	DATETIME		YES	NO		出生日期
age	INT	4	YES	NO		年龄
gender	VARCHAR	10	YES	NO		性别
nation	VARCHAR	30	YES	NO		民族
address	VARCHAR	255	YES	NO		居住地址
paper_file_num	VARCHAR	255	YES	NO		纸质档案编号
register_id	BIGINT	255	NO		F	报名表主键
created_time	DATETIME		NO	NO		数据创建时间
enable	TINYINT	4	NO	NO		数据是否有效

表 3　教练表（driver_coach_archives）

字段名	数据类型	长度	允许为空	自动递增	主/外键	备注
id	BIGINT	255	NO	YES	P	主键 id
user_id	BIGINT	255	NO		F	用户表主键
car_info_id	BIGINT	255	NO		F	车辆信息表主键
RFID_card_id	BIGINT	255	NO		F	RFID 卡片表主键
real_name	VARCHAR	50	YES	NO		真实姓名
birthday	DATETIME		YES	NO		出生日期
age	INT	4	YES	NO		年龄
gender	VARCHAR	10	YES	NO		性别

nation	VARCHAR	30	YES	NO		民族
address	VARCHAR	255	YES	NO		居住地址
wage	NUMBER	10,2	YES	NO		工资
driver_type	VARCHAR	100	YES	NO		驾照类型
created_time	DATETIME		NO	NO		数据创建时间
enable	TINYINT	4	NO	NO		数据是否有效

通过中间桥表（student_associate_coach）的方式，将学员与教练建立多对多的关联关系，关联关系的数据字典，如表 4 所示。

表 4　学员教练关联表（student_associate_coach）

字段名	数据类型	长度	允许为空	自动递增	主/外键	备注
id	BIGINT	255	NO	YES	P	主键 id
student_id	BIGINT	255	NO		F	学员表主键
coach_id	BIGINT	255	NO		F	教练表主键
associated_time	DATETIME		NO	NO		关联时间
comment	VARCHAR	255	YES	NO		备注
created_time	DATETIME		NO	NO		数据创建时间
enable	TINYINT	4	NO	NO		数据是否有效

RFID 模块是本系统中数据传递的重要组成部分，考虑到系统中用户数据的安全性，RFID 卡片扇区中不存储用户的任何数据，采用只读卡，不写卡的方式，将 RFID 的唯一识别卡号在数据库中记录，并与对应的用户信息进行绑定，每次读卡操作都将使用网络通信的方式，向服务器中请求该卡片对应的用户数据，并将用户的操作信息发送至服务器端，保存在数据库中。由上所述，需要建立 RFID 卡片的数据字典，如表 5 所示，并通过外键与用户数据相绑定。

表 5　RFID 卡片表（driver_rfid_info）

字段名	数据	长度	允许	自动递增	主/外键	备注

	类型		为空			
id	BIGINT	255	NO	YES	P	主键 id
RFID_num	VARCHAR	255	YES	NO		RFID 卡片唯一编号
color	VARCHAR	10	YES	NO		卡片颜色
publish_time	DATETIME		YES	NO		卡片激活时间
comment	VARCHAR	255	YES	NO		备注
created_time	DATETIME		NO	NO		数据创建时间
enable	TINYINT	4	NO	NO		数据是否有效

除了上述的几张表之外，系统正常运行还需要管理员表、权限表、RFID 读写卡记录表、科目表、成绩表、练车预约记录表、教练车信息记录表、用户操作记录表等，其余表字段简单，关联性不强，这里不再详细列出，以上数据字典都是本系统核心业务运行的基础数据表。

通过软件自动化构建物理模型非常方便，使用软件 Power Designer 构建物理模型，只需要根据数据字典将表和各个字段一一对应，再画出来表跟表之间的对应关系，就能创建出物理模型，然后一键生成 MySQL 可执行脚本文件，再使用数据库运行脚本文件，就可以创建出数据库以及生成对应的 E-R 关系图。

2.2.2 系统主要硬件设计

本系统采用的主要硬件设备清单如表 6 所示。

表 6　主要硬件清单

型号	数量	用途	主要功能
STM32F103ZET6 系列开发板	1 个	处理数据，控制开发板上的各个硬件工作	实现开发板各个引脚功能复用，驱动串口工作
MFRC-522 RFID 射频读写卡模块	1 个	读写 S50 系列 IC 卡、异形卡数据	读写射频卡数据
IC 射频卡	3 张	提供数据源	监测用户行为，记录用户信息
ESP8266 WIFI 模块	1 个	网络通信	通过 HTTP 协议向远程服务器发送数据

本系统采用的开发板的 MCU 是基于 ARM 的 32 位通用增强型微控制器，引脚数目 144，闪存容量 512K 字节，具体型号为 STM32F103ZET6。这款开发板属于通用定制款，板子增强和拓展了官方提供的引脚定义，对部分通用引脚功能做了自动化配置。除了使用官方推荐的仿真器下载程序外，本开发板还可以通过 USB 直接下载程序，方便快捷，不再需要单独购买仿真器。只需要将开发板的 BOOT0 和 BOOT1 的电平位通过跳帽修改，切换启动模式到系统存储器启动模式下，通过串口直接下载程序到板载 FLASH 闪存中。使用好官方的固件库，更能减轻代码编写的复杂度，降低开发难度，提升开发效率。本论文所有程序基于从服务商网站下载到的官方 V3.5 版本的固件库开发，通过官方提供的示例程序来减轻开发难度。图 2 为本系统基于 STM32 微控制器提供的物联网平台原理图。

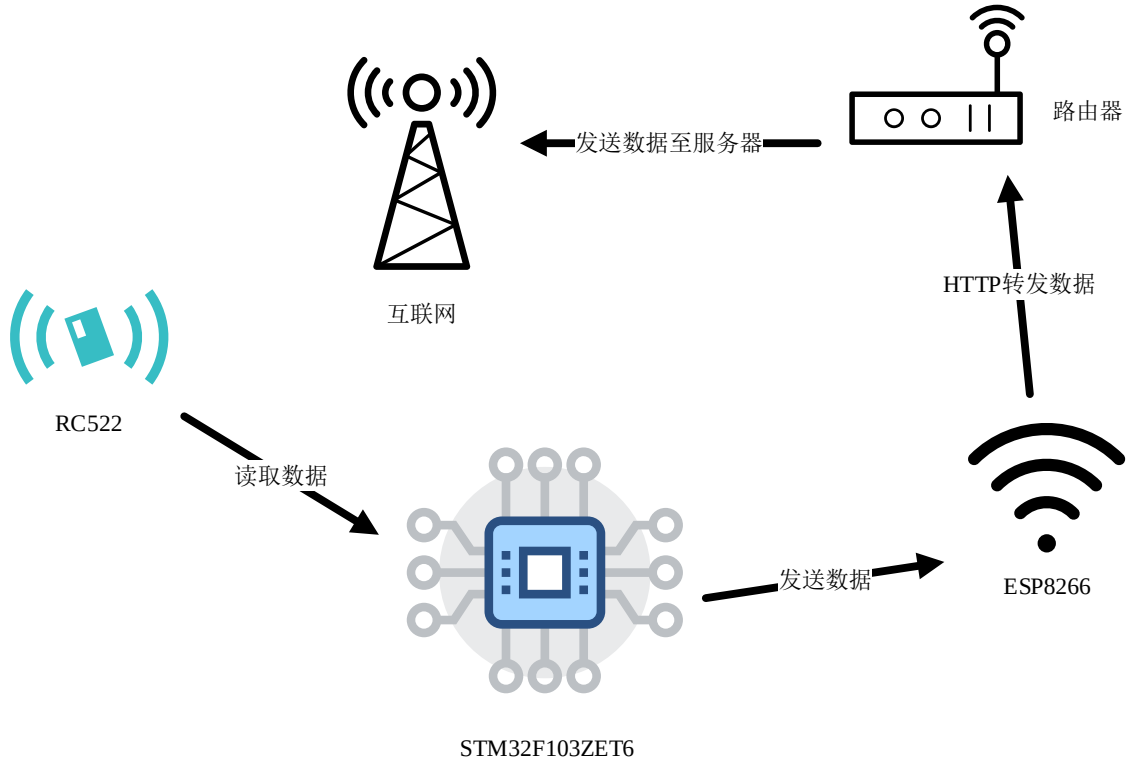


图 2　物联网平台原理图

2.2.3 系统 Web 端设计

Web 端主要分为两部分，Web 服务端和 Web 前端。Web 服务端是一台使用 Java 语言开发的服务器，用来接收 ARM 微控制器通过网络发送的数据，将接收到的数据分析之后存入数据库中，用户在 Web 前端查看对应的服务时，Java 服务器再将数据从数据库中读取出来，经过业务流程的筛选处理，渲染到前端页面上。Java 服务器基于轻量级开源框架 Spring 开发，能够完美支持多线程、高并发，提供了一站式解决方

案，包括依赖注入、动态加载、切面编程、事务管理等。Web 模块主要的功能原理如图 3 所示。

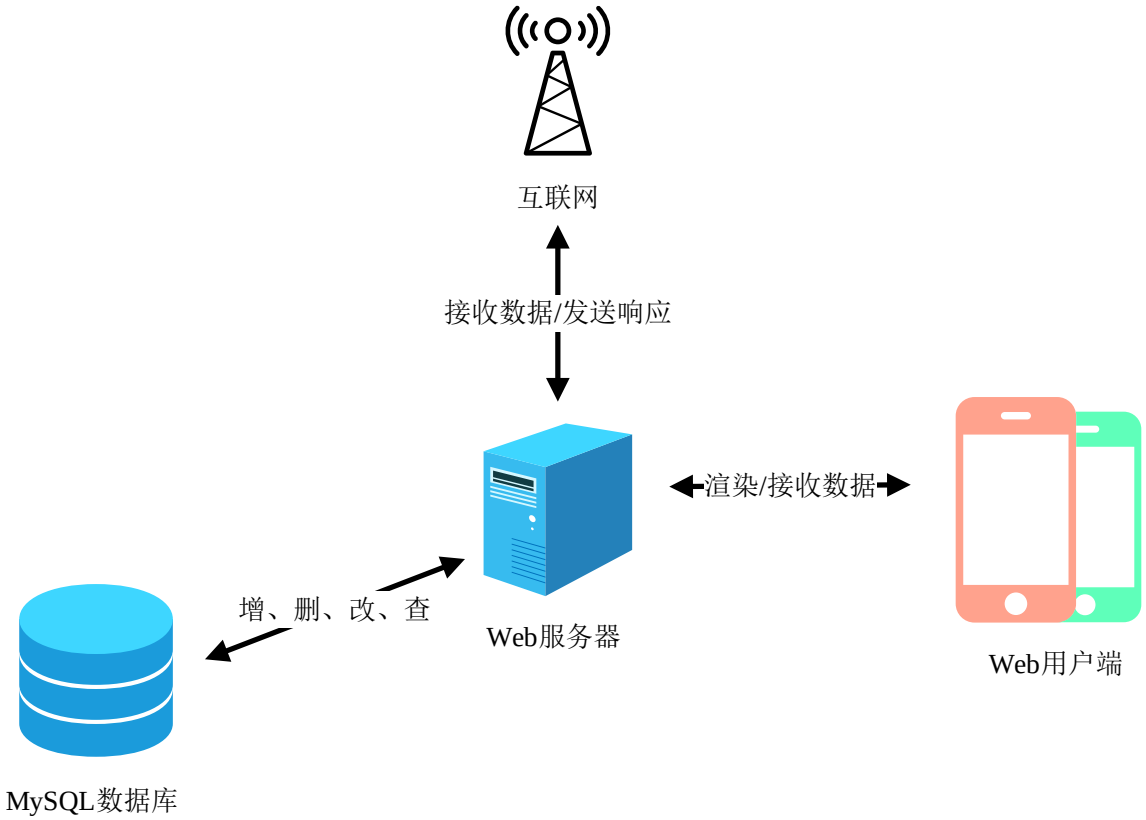


图 3　Web 模块原理图

3 系统详细实现方案

3.1 项目版本管理

本系统从项目创建到最终开发完成，都采用了分布式版本控制器 git 来管理代码，以及控制项目版本的迭代，git 不仅仅具备版本控制的功能，还支持良好的团队协作，以及远程仓库关联同步等功能。git 是一款开源免费的软件，不管有没有网络环境，都能正常工作，创建好本地仓库之后，没有远程仓库也能够在本地管理代码。[9]面前互联网上能够与 git 配合使用的免费仓库有 Github 和 Gitee（中文名码云），同时也可以使用 gitLab 自己搭建 git 远程仓库，本系统使用的远程仓库是 github。完成项目版本控制，需要搭建以下环境，如表表 7 所示。

表 7　版本控制器环境

名称	官网地址	性质	备注
Github 账户	https://github.com/	免费	注册账户 创建远程仓库

Git 客户端	https://git-scm.com/	免费	git 命令环境

安装好 gitBash 环境，同样需要配置 git 的环境变量，配置好环境变量，打开命令行终端，在终端里输入 git –version，看到 git 的版本信息，如图图 4 所示，就具备了 git 版本控制器的环境。

图 4　查看 git 版本

3.1.1 初始化本地项目仓库

为了方便管理系统的全部代码，在开始真正编码之前，先创建项目的目录结构，将系统所有的文件整理归类，让 git 来统一管理，git 虽然不管理空文件夹，但当文件夹下有了内容，git 自然会去管理。

首先新建一个文件夹，根据表 8 所示清单，创建好如图 5 所示的目录。

表 8　项目目录结构说明

文件/文件夹名称	用途
back_end	Java 后端服务器代码存放目录
front_end	Web 前端页面代码存放目录
database	数据库相关内容存放（模型、sql 文件等）
stm32_client	STM32 ARM 程序存放目录
doc	设计说明文档存放目录
README.md	git 仓库项目简要描述说明文档

图 5　项目结构目录

打开命令行终端，使用命令 git init 将目录初始化为一个 git 仓库，执行完成后，

该目录下会变成一个 git 管理的仓库，同时会产生.git 文件夹，如图 6 所示

```
d:\Documents\new\driver  $ git init
Initialized empty Git repository in d:/Documents/new/driver/.git/

d:\Documents\new\driver (master -> origin) $ ls -a
./  ../  .git/  back_end/  database/  doc/  front_end/  README.md  stm32_client/
```

图 6　初始化项目为 git 仓库

然后使用命令将初始化仓库提交到本地版本库，提交时，git 会询问操作的用户，使用命令配置用户名和邮箱，再次提交到本地版本库，相关命令如下。

```
$ git add .
$ git config user.name “”
$ git config user.email “”
$ git commit -m “message”
```

本系统的项目包括有 C 语言编写的 ARM 程序，Java 开发的 Web 服务器，前端框架开发的 Web 页面，版本库只需要管理这三个平台的源代码即可，例如 C 语言编译的过程文件，编译生成的可执行文件，下载到单片机开发板的 hex 文件，Java 编译后的 class 文件等，都可以从源代码再次编译产生，版本库需要忽略掉这些文件，需要在项目根目录下创建.gitignore 文件，在里面编写需要忽略追踪的文件的表达式，相关忽略规则，如表 9　git 忽略规则所示。

表 9　git 忽略规则

规则定义	规则说明
*.o	忽略所有文件后缀是 o 的文件
!*.c	所有后缀是 c 的文件都不能忽略
log/	log 目录下的所有文件全部忽略
/log/	任何位置的 log 目录及其子文件全部忽略

然后将.gitignore 文件同时提交到版本库中。

3.1.2 关联同步远程仓库

github 提供了免费的代码仓库管理服务，用户可根据自己的需求注册账户，创建远程代码版本仓库，仓库可以设置为公开或者是仅供自己访问，具备安全可靠，方便快捷等优势。github 已经被开发者们广泛认可，做为代码开源共享、版本控制管理的平台，github 已经成为了主流的版本控制解决方案。注册 github 账户之后，新建代码仓库如图 7 所示。

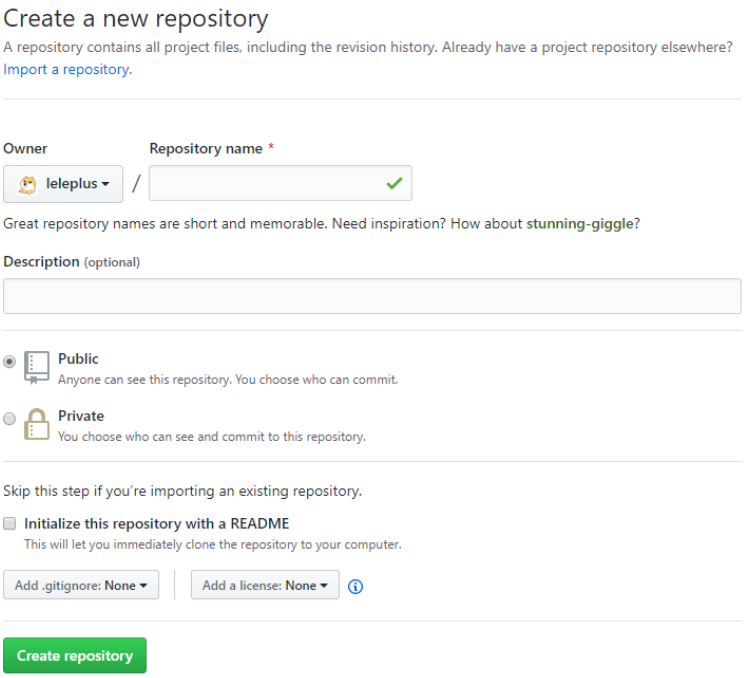


图 7　创建远程仓库

创建好之后，会得到一个远程仓库地址，如图 8 所示，有 SSH 和 HTTPS 两种协议的地址，我这里使用 SSH 协议地址

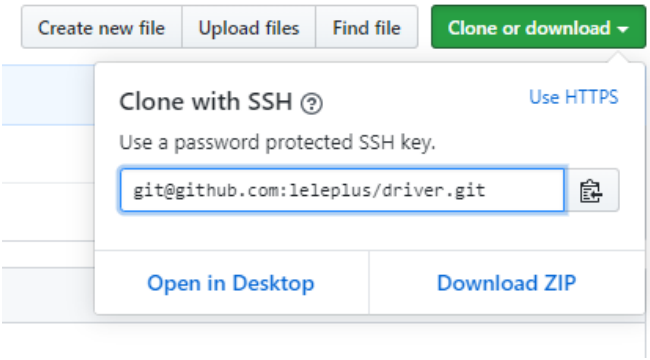


图 8　远程仓库地址

使用 SSH 协议之前，需要先将本地电脑的 SSH 公钥配置到 github 的 SSH Key 中，然后在创建好的本地仓库里，通过以下命令完成远程仓库的关联，以及代码推送。

```
$ git remote add origin git@github.com:leleplus/driver.git
$ git push -u origin master
```

后续将代码推送到远程只需要使用命令 git push。

3.1.3 更新同步项目代码

本系统的版本控制，主要管理的内容有三部分，STM32 相关的 Keil 代码，Web 服务器和 Web 页面的代码，还有系统文档，对应这三部分内容，分别使用 git 创建三

个分支，相关命令为 git checkout -b 分支名，在对应分支下编写对应模块的代码，编写完成后，将对应的分支使用 git push 推送到远程仓库管理，本地分支如图 9 所示。

```
D:\Documents\driver\driver (dev/doc -> origin) $ git branch
  dev/arm
* dev/doc
  dev/server
  master
```

图 9　项目分支

管理好项目仓库之后，每次编写代码，只需要使用命令 git push 将代码推送到远程分支，更新代码时，使用命令 git pull，切换对应模块所在的分支，使用命令 git checkout，最终完成项目之后，使用 git merge 命令将各个模块所在分支的代码合并到 master 分支即可。

3.2 数据持久层实现

3.2.1 系统数据建模实现

按照系统的需求分析流程，编写出了系统的数据字典文档，按照数据字典文档使用物理模型构建工具 Power Designer 创建物理模型，实现流程如下：

（1） 安装好 Power Designer 工具，运行软件，如图 10 所示，新建一个模型。

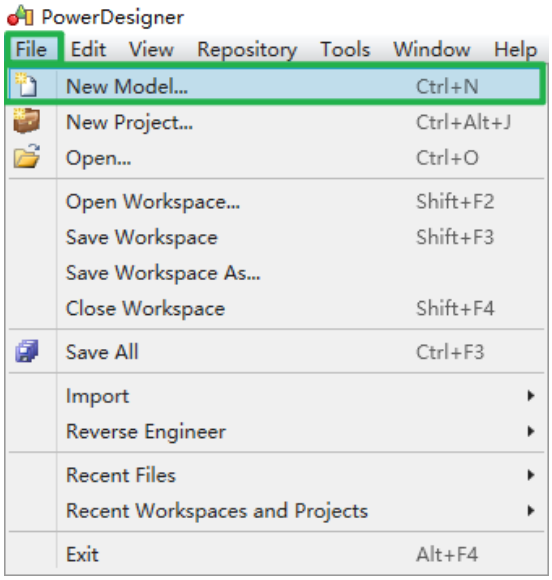


图 10　新建模型

（2） 选择模型的类型为物理模型，选择方式如图 11 所示。本系统使用的 MySQL 版本为 5.7.29，这里需要选择 MySQL 版本为 5.0。

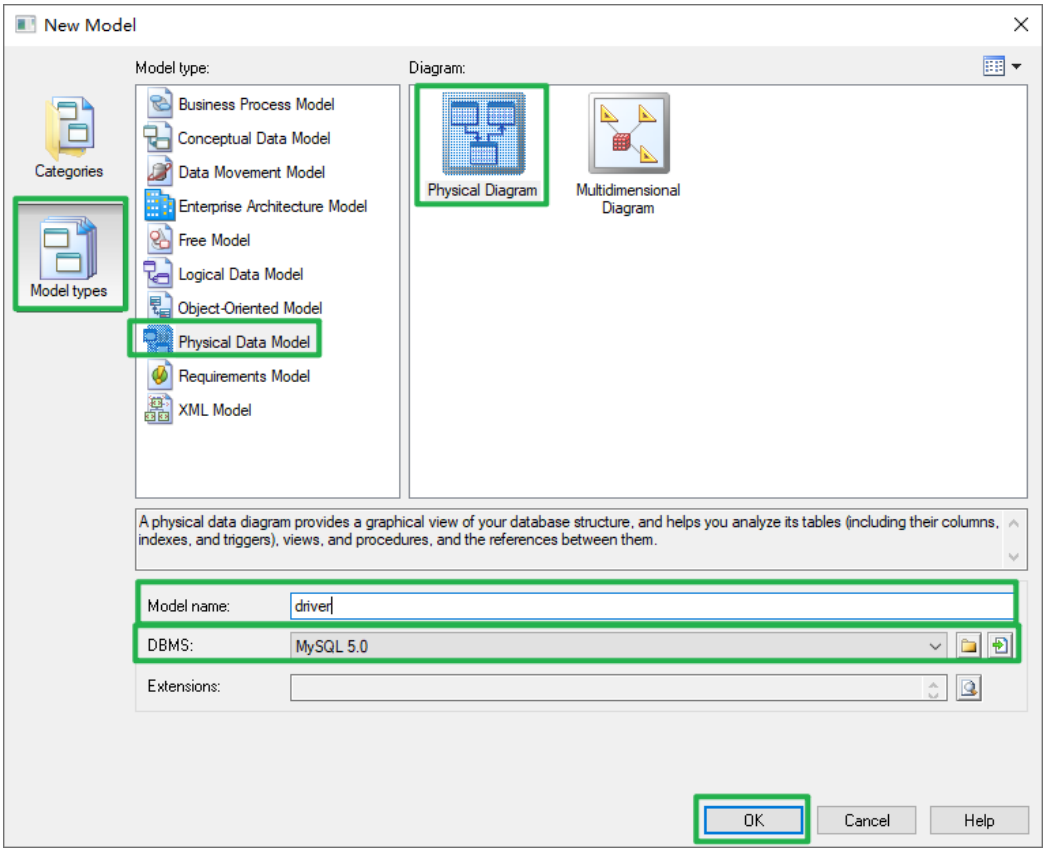


图 11　选择模型类型

（3） 工具提供了在图形化界面通过拖拉拽的方式快速构建模型，如图 12 所示，在 ToolBox 菜单下，有表（Table）和关联关系（Reference）的按钮，使用鼠标，点击这两个按钮，能够方便的构建出物理模型以及实体之间的关系。

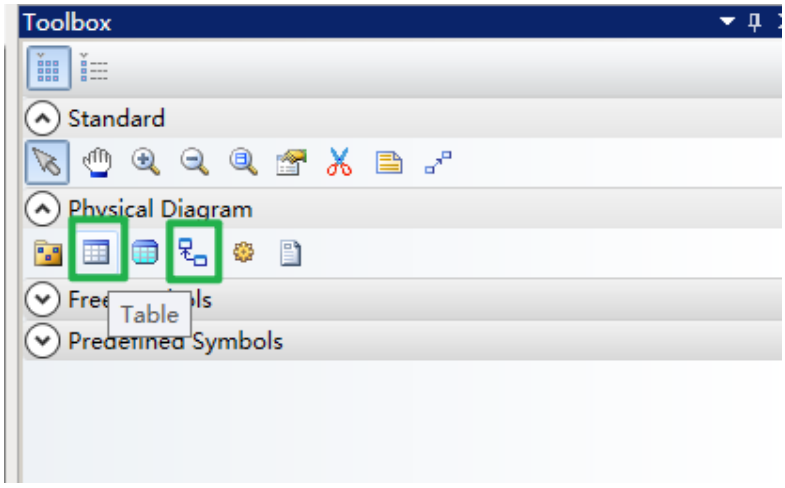


图 12　构建实体

（4） 双击每一个 Table，在弹出的窗口中，编辑添加表对应的字段、类型、长度、等属性，如图 13 所示。

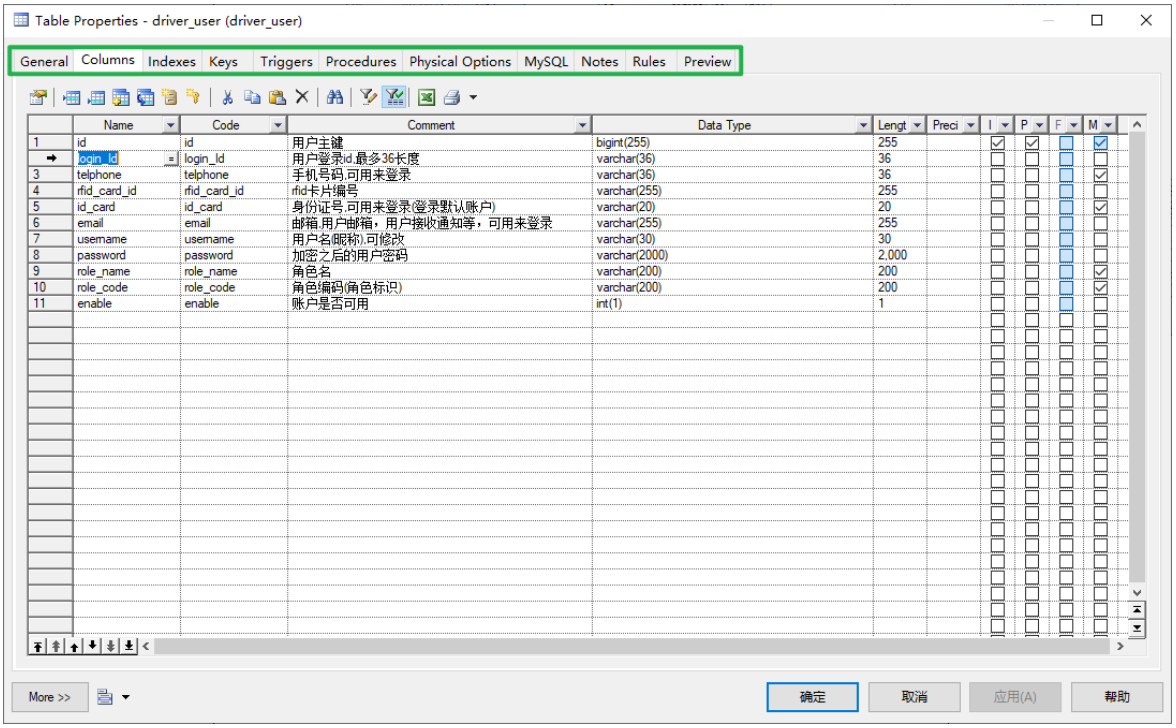


图 13　编辑 Table

（5） 创建完模型，按照图 14 所示方式，可直接预览物理模型对应的数据库 SQL 执行脚本，如图 15 所示。

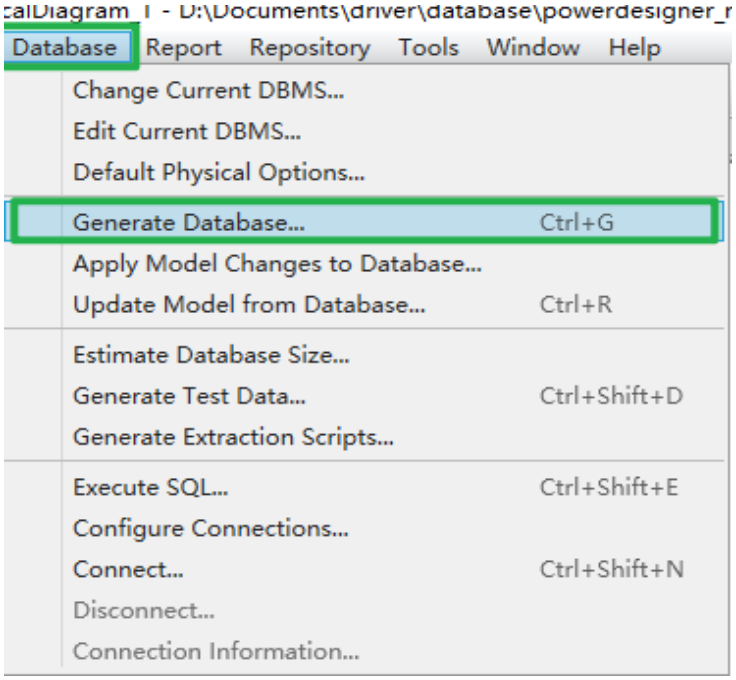


图 14　生成脚本

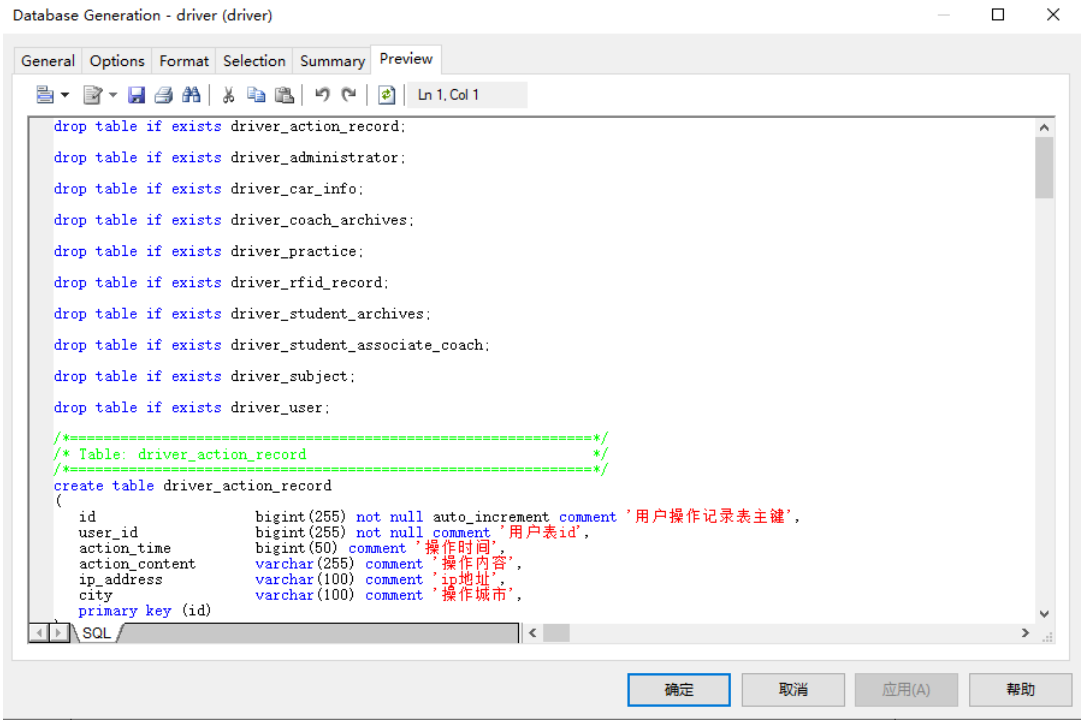


图 15　预览 SQL 脚本

（6） 设置编码，本系统使用的字符编码统一为 UTF-8，导出 SQL 脚本前设置脚本编码，如图 16 所示。

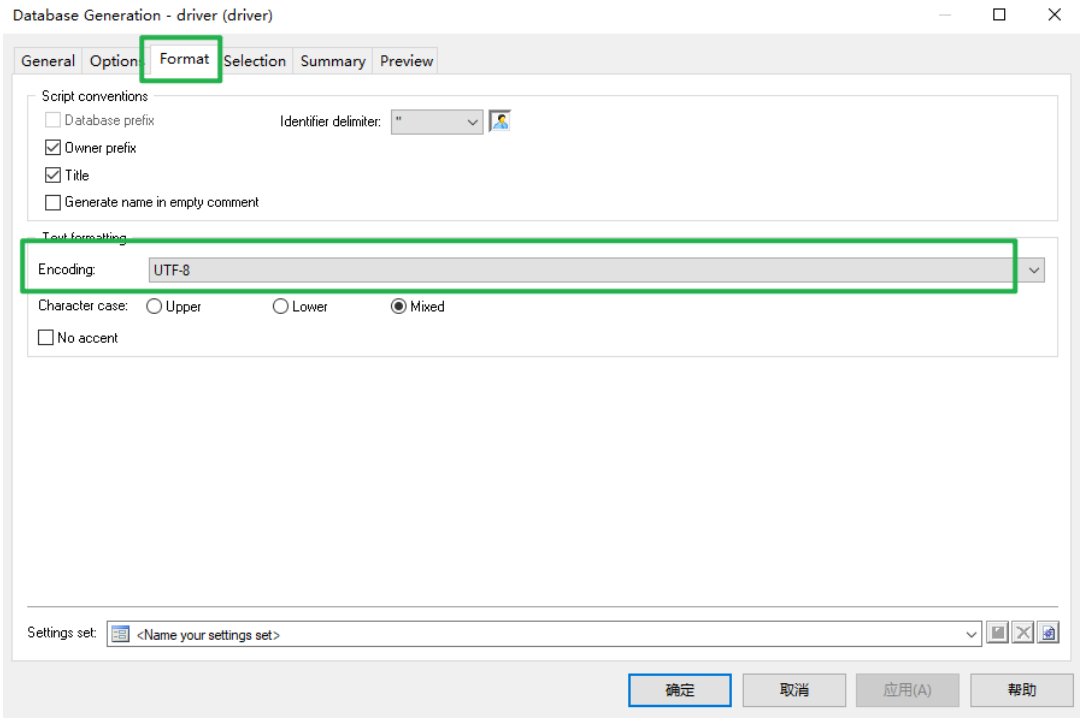


图 16　设置 SQL 脚本编码

（7） 选择导出 SQL 脚本的位置以及脚本文件的名称，如图 17 所示。点击确定之后，

就会在对应的目录下生成好文件。

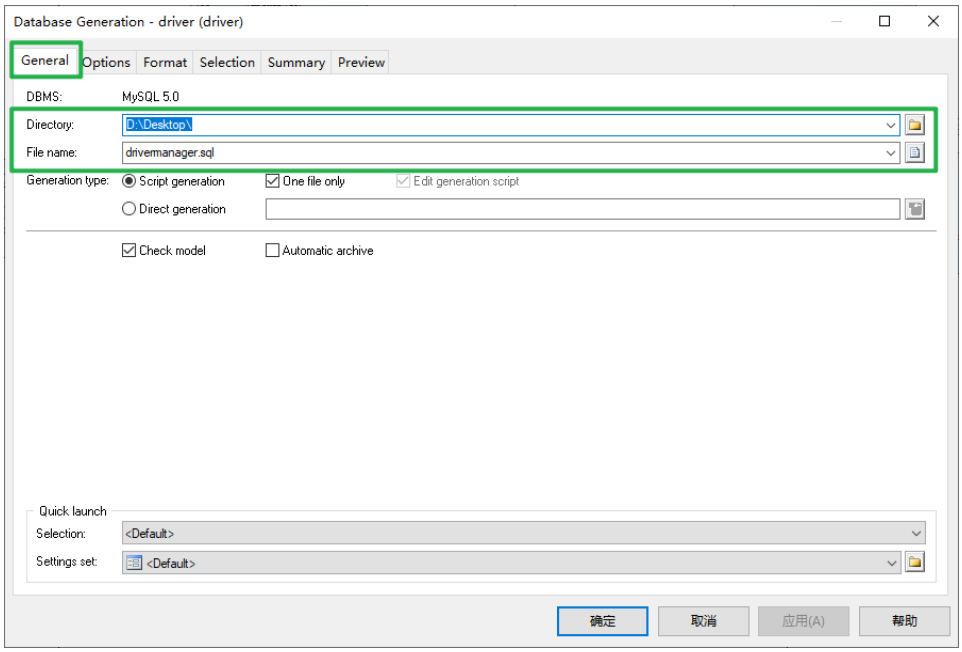


图 17　选择脚本位置设置文件名

3.2.2 数据库设计实现以及 E-R 图的生成

构建完物理模型，就已经有了数据库的脚本文件，直接在数据库中运行脚本文件，就会在数据库中创建好跟物理模型一样的数据库结构。完成数据库设计以及 E-R 生成需要如表 10 所示的软件清单。

表 10　数据库设计环境

软件名称	版本	用途	说明
JDK	1.8	Java 开发工具集以及 DBeaver 运行时环境	免费下载 需要配置环境变量
MySQL	5.7.29	数据库软件	免费开源 需要配置环境变量
DBeaver	7.0	数据库可视化管理工具	免费开源 基于 Java 开发，需要 JDK

具体实现步骤如下：

（1） 安装完软件，开始创建数据库，如图所示，打开数据库可视化管理工具 DBeaver，使用账户密码连接到数据库，如图 18 所示。

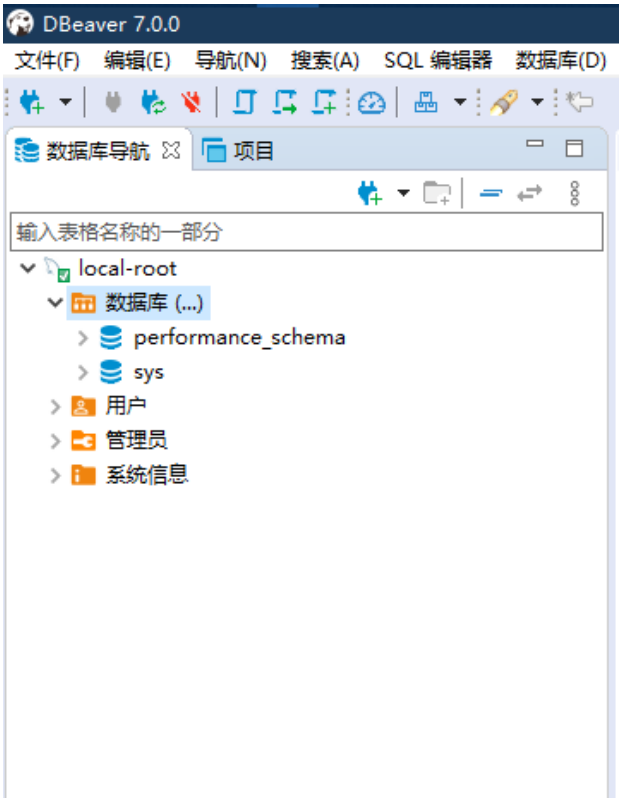


图 18　连接数据库

（2） 新建开发用的数据库，保证编码为 UTF-8，如图 19 所示。

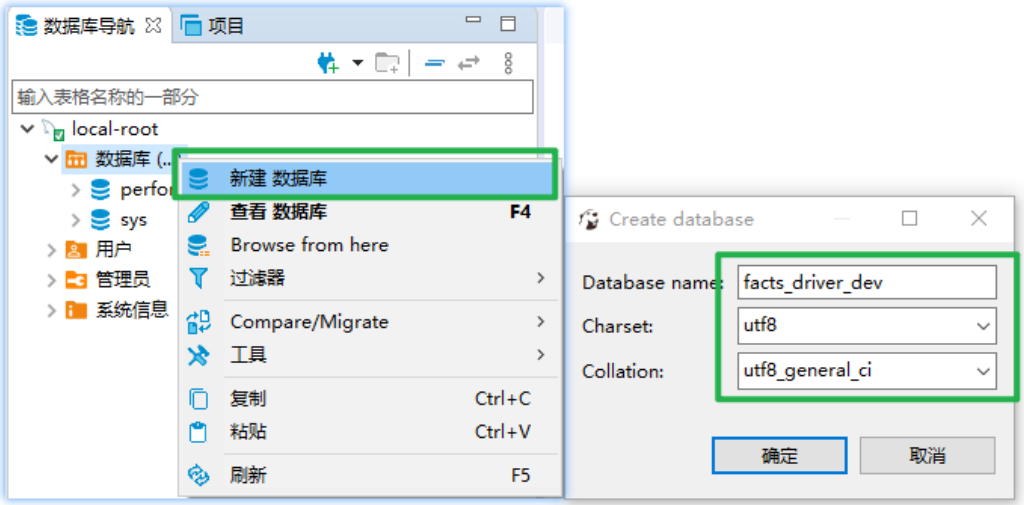


图 19　创建数据库

（3） 运行数据库脚本，按照图 20 所示的步骤，运行数据库脚本文件，如图 21 所示，需要指定 MySQL 的安装目录，以及物理模型生成的 SQL 脚本文件的位置，点击确定即可创建好数据库结构。

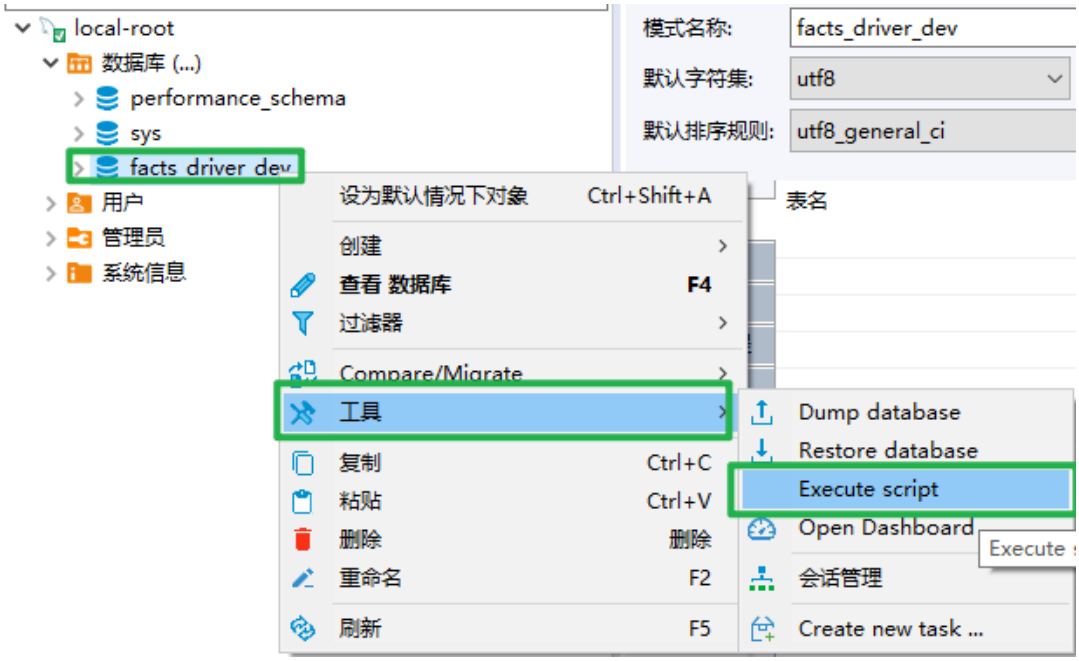


图 20　运行脚本

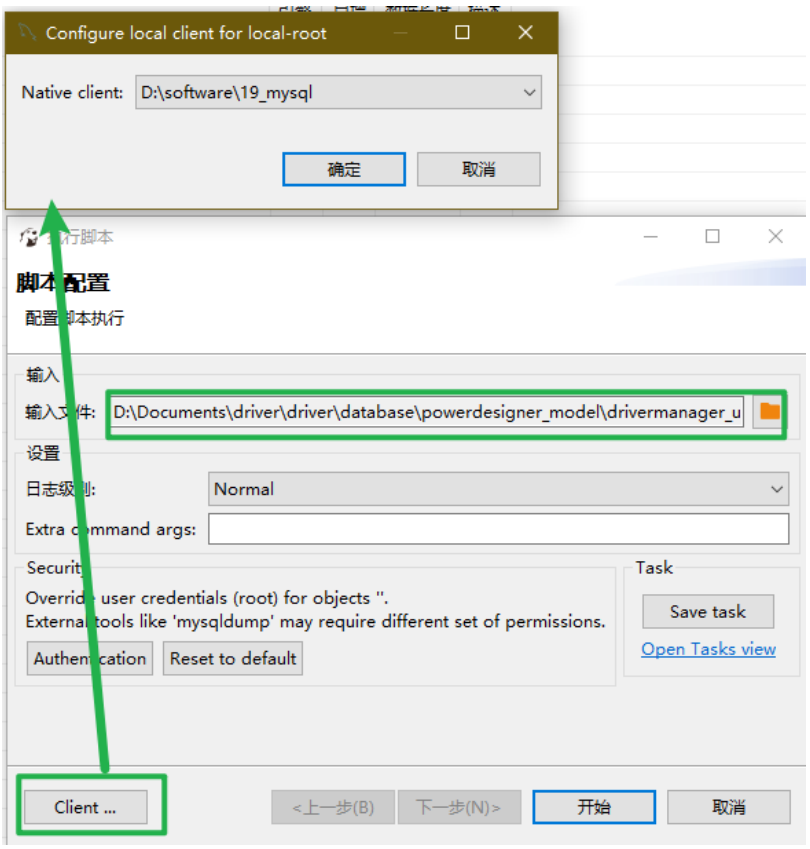


图 21　选择 mysql 客户端和脚本文件

（4） 数据库已经创建完成，创建好的表如图图 22 所示。

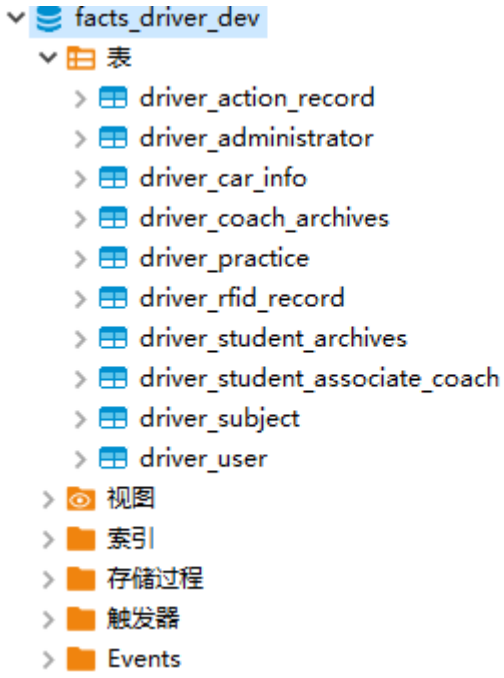


图 22　系统所需数据库的表

（5） 生成 E-R 关系图，双击打开数据库，切换到 E-R 图，即可看到 E-R 图，步骤如图 23 所示。可以导出为图片等。

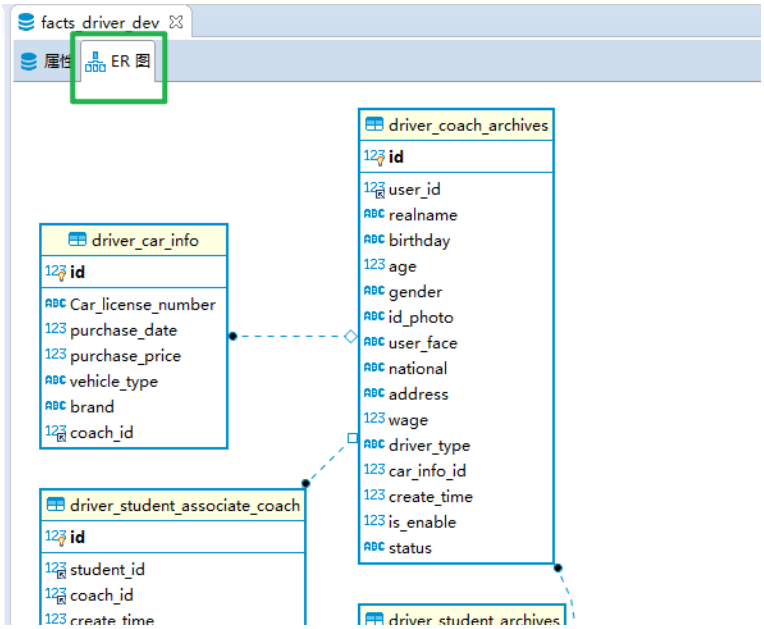


图 23　生成 E-R 图

系统最终使用的 E-R 关系图，见附录 1 所示。

3.3 物联网模块实现

3.3.1 物联网模块实现流程

本设计的物联网模块主要为系统提供读写卡支持，用户的数据是保存在系统后台的数据库中的，由数据库来关联用户与 RFID 卡片的数据，实物卡片中并不会存储用户的任何个人信息。在系统中 RFID 可以用来作为门禁，也可以用来作为打开教练车车门的凭证，这个卡片是由驾校在用户报名时提供给用户的，卡片由驾校负责人提前录入系统中，RFID 主要做卡片识别和数据交互功能。硬件环境主要分为两大模块，管理员模式和普通用户刷卡模式。

管理员模式流程图如图 24 所示。

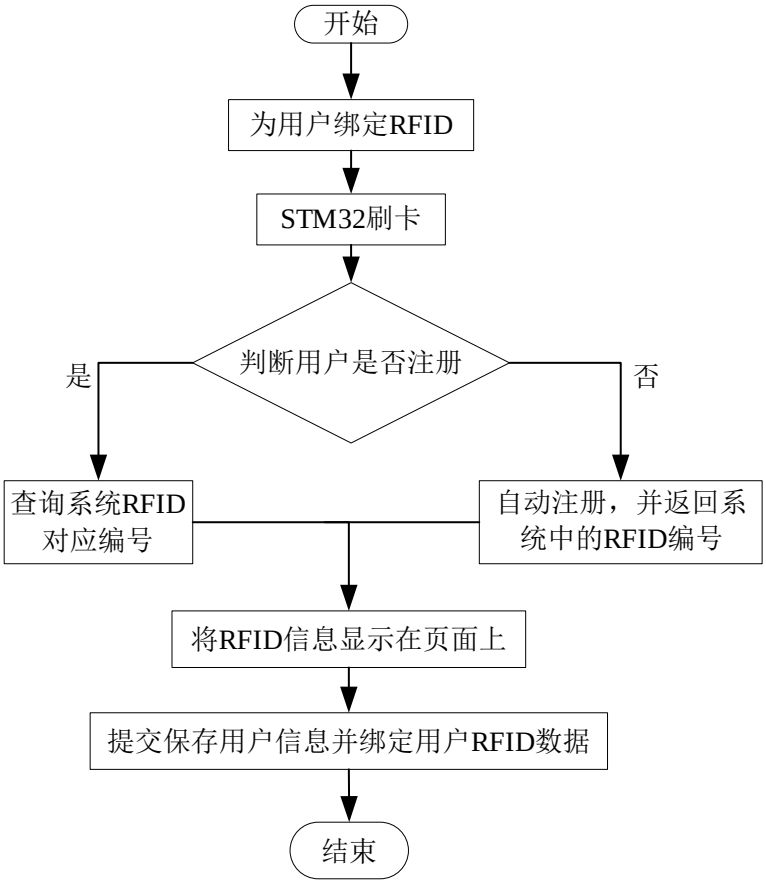


图 24　管理员模式

普通用户刷卡模式流程如图 25 所示

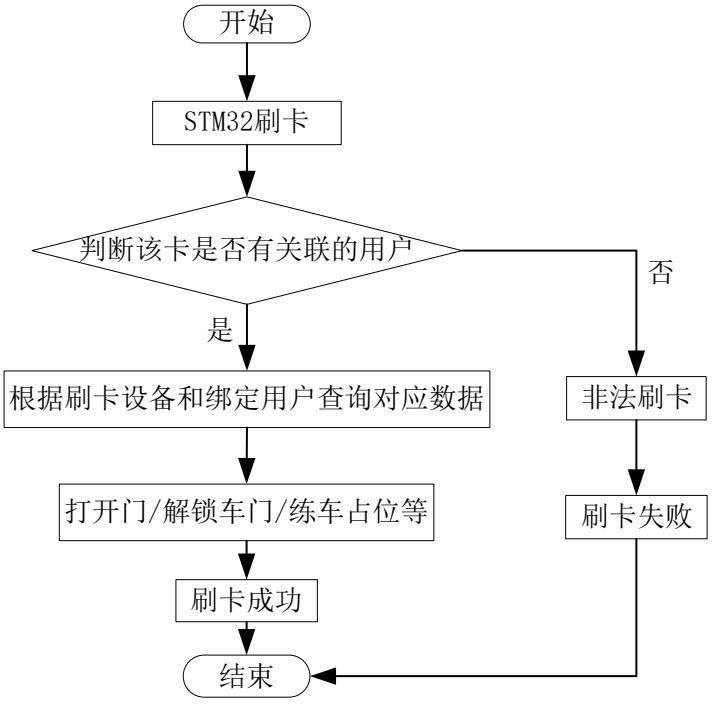


图 25　普通用户刷卡模式

3.3.2 物联网模块硬件产品介绍

物联网模块主要是在 ARM 上做 C 语言开发，使用 Keil5 作为开发环境，同时需要安装 STM32F103ZE 系列的软件包来编译运行 C 语言代码，完成本模块需要的开发环境如表 11 所示。[10]

表 11　STM32 开发环境清单

名称	用途	版本	来源
Keil-MDK	编程环境	V5.11	网络下载
STM32F1xx_DFP	编译执行 STM32F103 固件库代码	V2.1.0	网络下载
STM32F103ZE 固件原理图	查看引脚定义等	V1.1	随 STM32 购买附赠
STM32F103ZE 尺寸图	查看设备型号、接口等	V1.08	随 STM32 购买附赠
Micro-USB 下载线	下载程序		购买
杜邦线	连接模块和开发板		购买
固件库函数	编写 Keil 代码的底层参考程序	V3.5	随 STM32 购买附赠
CH340 驱动程序	下载程序驱动	V1.6.0	随 STM32 购买附赠
ISP 下载器 MCUISP	下载 HEX 文件到开发板	V0.99	随 STM32 购买附赠

3.3.2.1 STM32F103ZET6 核心板

本设计所使用的开发板，板载 MCU 为 STM32F103ZET6，主频为 72MHZ，闪存 512K,64K 的 SRAM，具备 4 个定时器，112 个 GPIO 引脚，5 个 UART，1 个 USB 接口，整体外观如图 26 所示。[11]板载三个 LED 灯，四个按键（一个复位按键，三个独立按键），支持 ISP 一键下载程序，同时具备了 SWD 仿真功能，本论文主要使用 ISP 下载程序，后面章节详细描述。

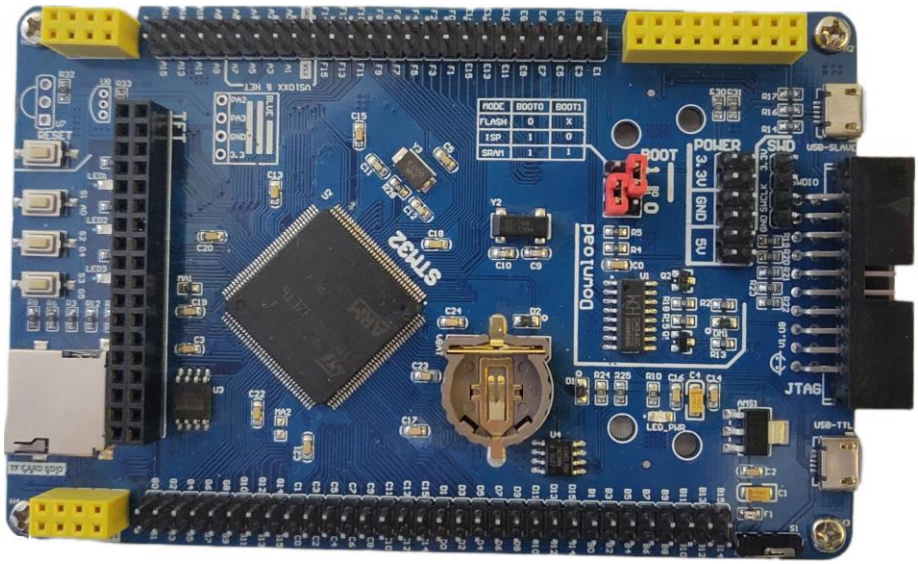


图 26　STM32F103ZET6 开发板外观图

3.3.2.2 ESP8266 网络通信模块

ESP8266 网络通信模块，尺寸为 5x5mm，模组需要的外围器件有：10 个电阻电容电感、1 个无源晶振、1 个 flash。工作温度范围：40~125℃。有 StandAlone 和 SIP 两种模式，CPU 主频支持 80M 到 160M，GPIO 输出电压 3.3V，具备两个 UART，UART0 和 UART1。[12]整体外观如图 27 所示。

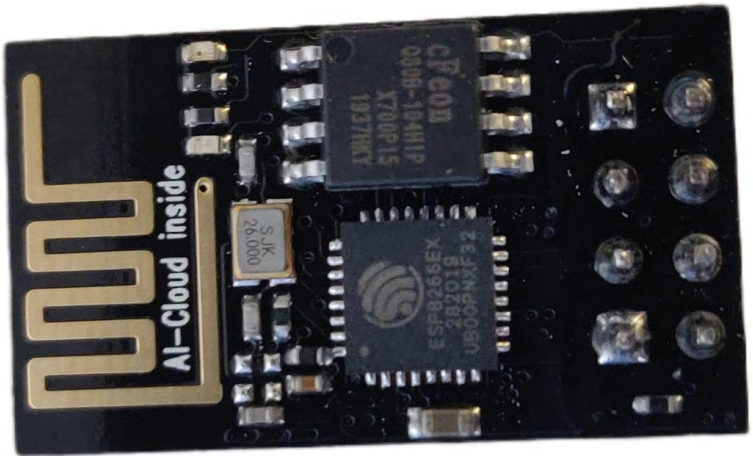


图 27　ESP8266 外观图

3.3.2.3 MFRC522 读写卡芯片

MFRC522 射频读写卡芯片，是一款非接触式的射频芯片，具有电压低，成本小、体积小的特点，该读写卡芯片采用先进的调制解调技术，完全集成了在 13.56 MHz 下所有类型的被动非接触式通信方式。[13]目前已经被广泛应用。MFRC522 模块与射频卡如图 28 所示。

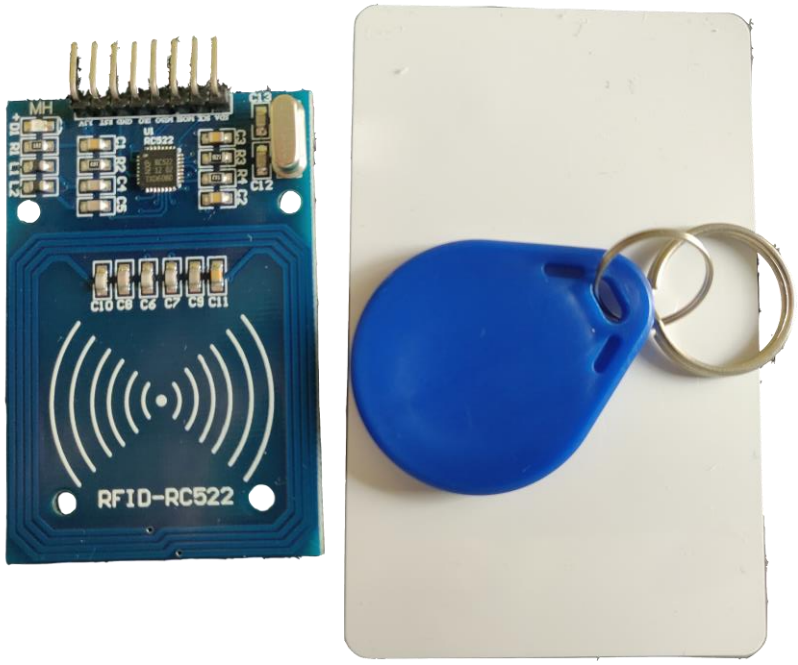


图 28　MFRC522 和射频卡

3.3.3 通过固件库创建工程

从零开始编写一套 ARM 程序，很显然任务量太重，本系统是基于官方固件库创建的工程，创建主要步骤如下

（1）下载好 STM32F10 系列的固件库源代码，如图 29 所示。

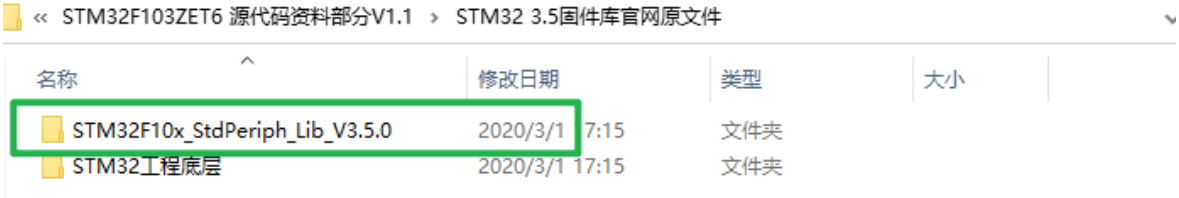


图 29　STM32V3.5 固件库

（2）新建文件夹 bin 和 src，bin 用来存放编译之后的文件，src 用来存放源代码，如图 30 所示。

图 30　源码和编译文件目录

（3）创建项目工程文件夹，具体作用如表 12 所示，创建完成后的效果如图 31 工程文件目录所示。

表 12　工程文件目录清单

文件夹名称	用途
CORE	存放 STM32 核心文件
FWLIB	各个外设的底层程序和源码
LIBRARY	自己定义引用的 STM32 库文件
STARTUP	STM32 启动文件
USER	自定义的程序以及主程序入口文件

图 31　工程文件目录

（4）再在每个文件夹下创建文件夹 inc 和 src，inc 用来存放.h 头文件,src 用来存放.c 文件，然后使用 Keil 创建工程，位置选择上一步创建的工程目录，如图 32 所示。

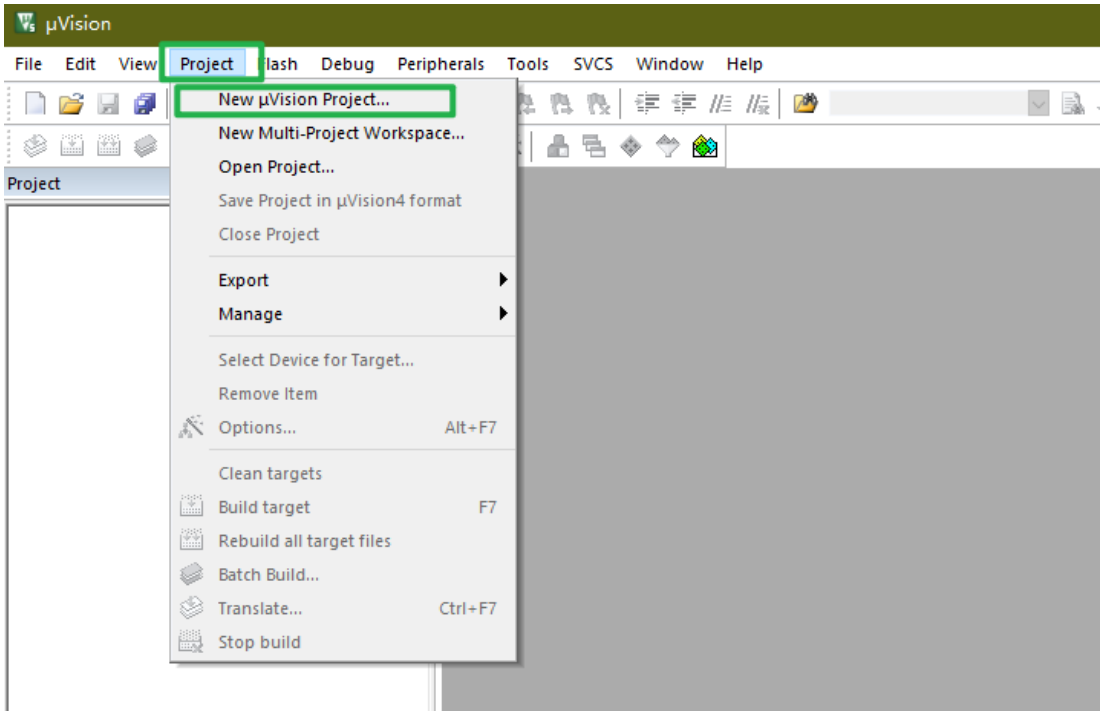


图 32　Keil 创建项目

（5）本系统使用的系统板为 STM32F103ZE 系列，新建项目需要使用对应的 Device 设备，如图 33 所示。

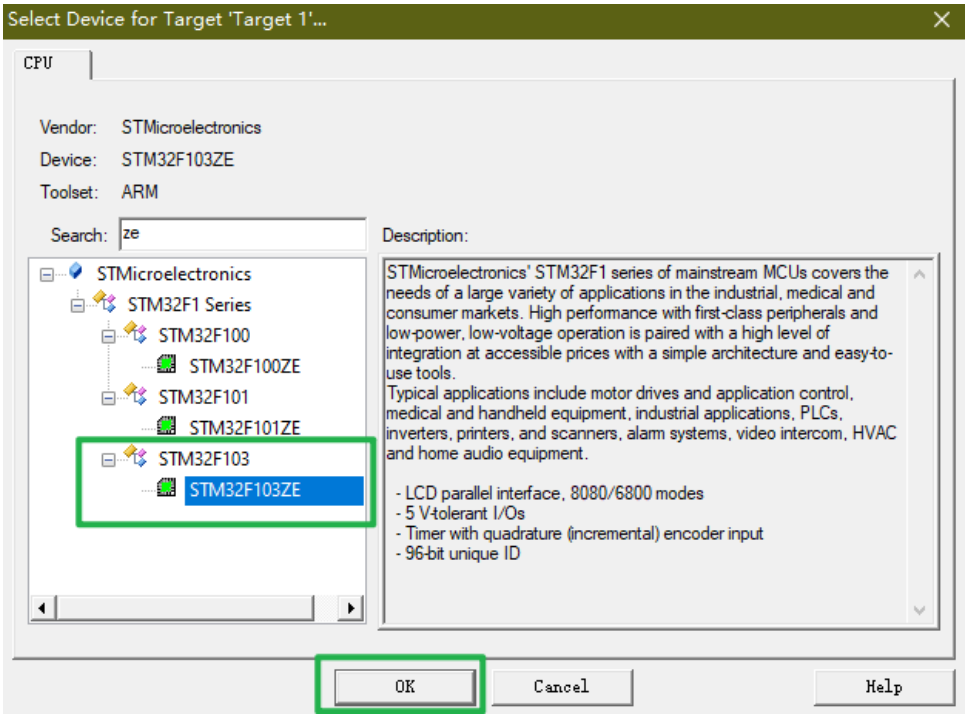


图 33　添加 STM32F103ZEDevice 到项目中

（6）将项目已有的程序文件关联到项目中，如图 34 所示。

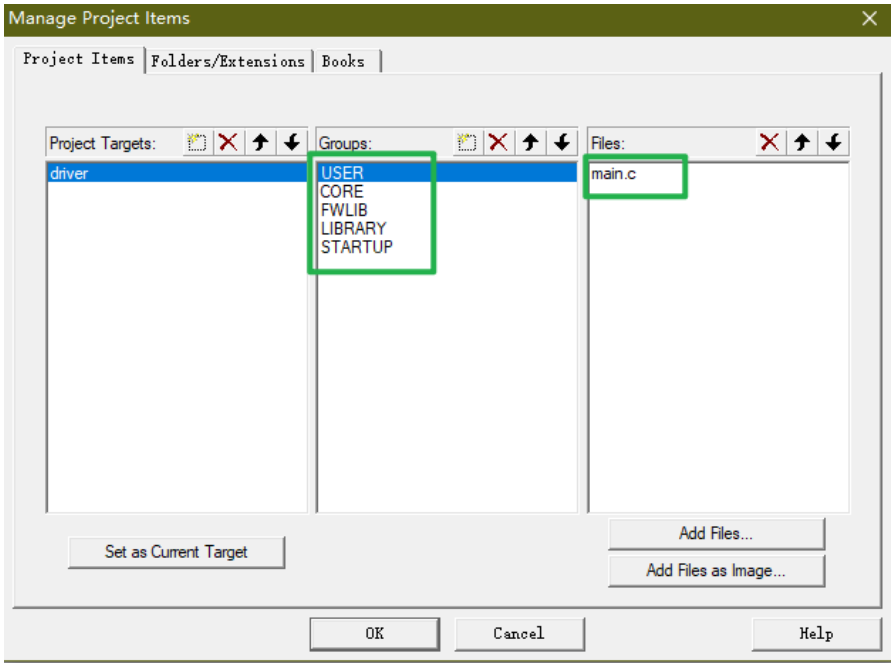


图 34　关联项目和工程

（7）关联完成后，整体项目目录结构如图 35 所示

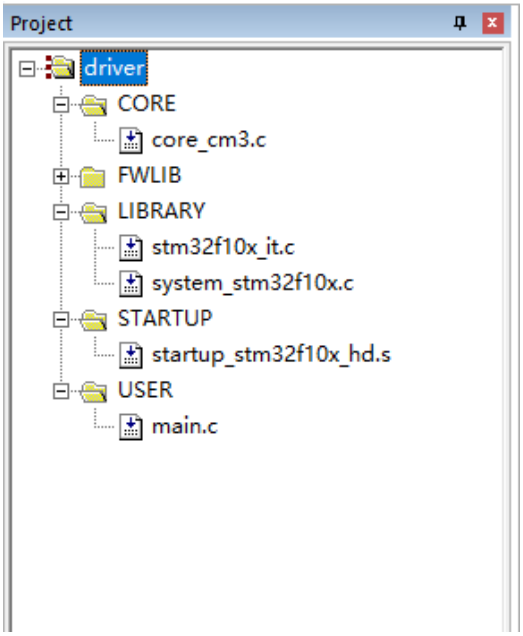


图 35　Keil 项目目录结构

（8）库函数在配置和选择外设的时候是通过宏定义来选择的，务必要配置一下系统全局的宏定义和头文件所在的目录，Keil5 的开发环境，已经预置了一部分宏定义，只需要配置 USE_STDPERIPH_DRIVER 就能完成编译，同时还要指定各个目录的 inc 文件夹位置，所有的头文件都存放在 inc 文件夹下，如图 36 所示。[14]

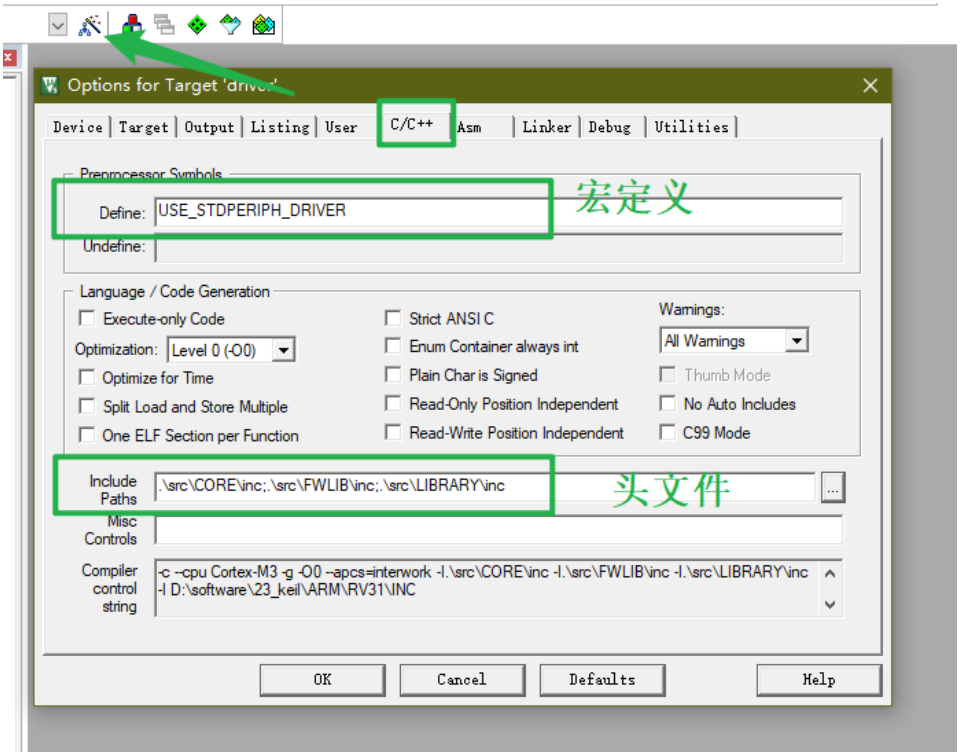


图 36　设置宏定义，关联头文件

3.3.4 ISP 串口下载程序

单片机下载程序的方法有好多种，本系统采用的是 ISP 方式下载，这种下载方式需要使用出厂预置在单片机内部的 BootLoader，所以需要更改单片机的启动模式。[15]STM32 启动模式在开发板图例和文档中都有介绍，更改启动模式，只需要更换 BOOT 跳帽的接入方式，切换高低电平即可。如表 13 所示为开发板配置启动模式的跳帽接入方式。

表 13 STM32 启动模式

启动模式	BOOT0	BOOT1
FLASH	0	
ISP	1	0
SRAM	1	1

串口 ISP 下载不需要购买单独的下载器，生活中使用的手机的 Micro USB 数据线就能完成程序的下载，下载完成后，程序直接在开发板上自动启动并且开始运行，不需要通过开发板上的 reset 键来复位程序。ISP 下载程序流程如下：

（1） 安装 CH340 驱动，驱动计算机识别开发板 COM 端口，如图 37 所示。

图 37　CH340 驱动识别单片机 COM 端口

（2） 设置 Keil 输出可以直接通过 ISP 下载的 HEX 文件，如图 38 所示，然后编译程序

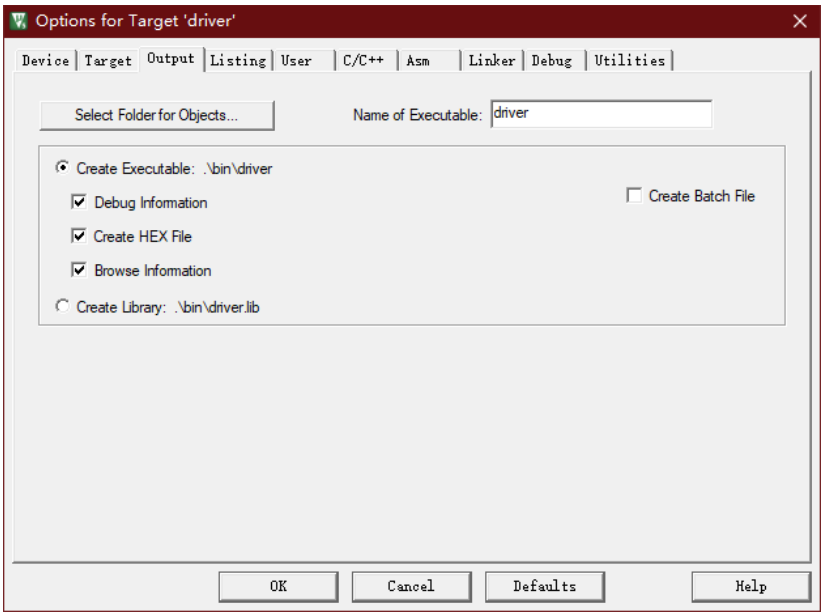


图 38　设置 Keil 输出 HEX 文件

（3） 打开 ISP 下载程序，将编译好的 HEX 文件下载到开发板中，如图 39 所示。

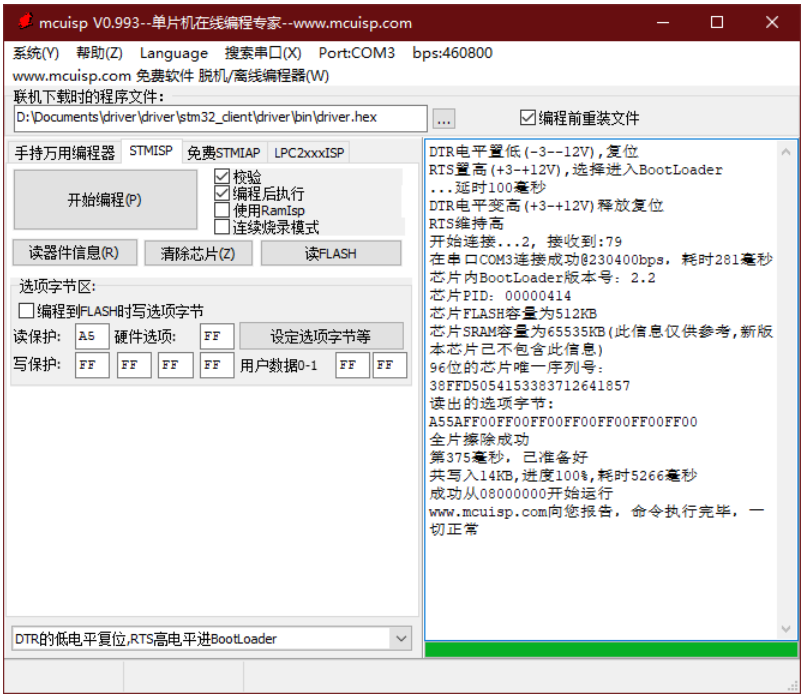


图 39 ISP 下载程序

3.3.5 板载硬件功能实现

本论文设计在开发核心功能前需要通过 STM32 开发板携带的 LED、按键等外设以及中断、串口来配合完成。

3.3.5.1 LED 功能实现

购买的开发板，板载三个 LED，相关电路图如图 40 所示

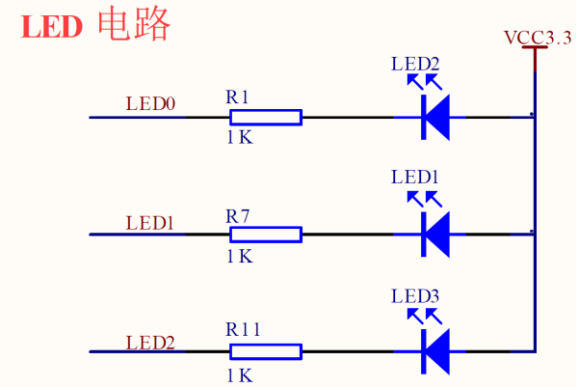


图 40 STM32 LED 电路图

通过查看 STM32F103ZE 原理图，查找到对应的 GPIO 引脚，如图所示

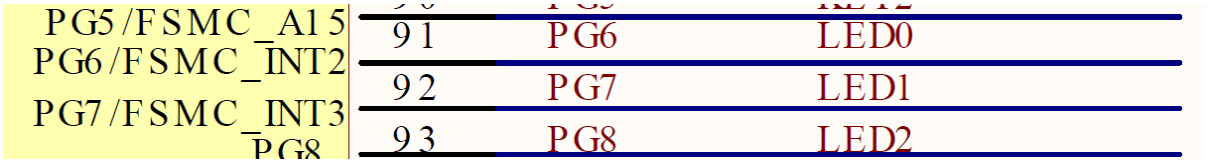


图 41 LED 引脚定义

查看官方文档中的介绍，通过使能 PA 端口时钟，配置 IO 口速率，初始化 GPIO 引脚，默认将 LED 对应引脚设置为低电平，完成 LED 设备初始化，初始化核心代码如下。

```
//定义 GPIO 结构体
GPIO_InitTypeDef  GPIO_InitStructure;

//使能 PA 端口时钟
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOG, ENABLE);
//LED-->PG6 7 8 端口配置
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7|GPIO_Pin_8;
//推挽输出
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
//IO 口速度为 50MHz

GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
//根据设定参数初始化 PG6 7 8
GPIO_Init(GPIOG, &GPIO_InitStructure);
//设置 PG 6 7 8 输出低，默认为关闭状态
GPIO_SetBits(GPIOG,GPIO_Pin_6|GPIO_Pin_7|GPIO_Pin_8);
```

初始化完 LED 引脚，通过将对应的 GPIO 引脚配置高低电平即可完成开关 LED 控制。

3.3.5.2 按键功能实现

开发板板载按键共四个，除了复位键，还有三个可以自定义功能，按键对应的的电路图如图 42 所示

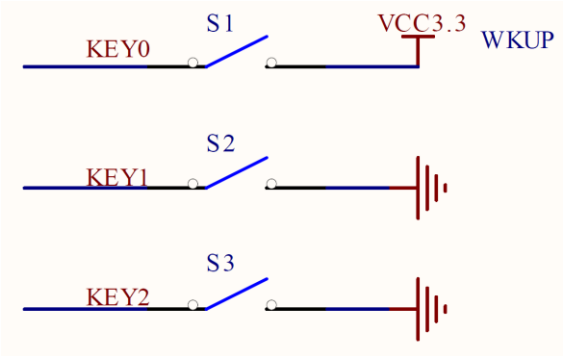


图 42 STM32 按键电路图

在原理图中查找按键对应的引脚定义，如图 43、图 44 所示

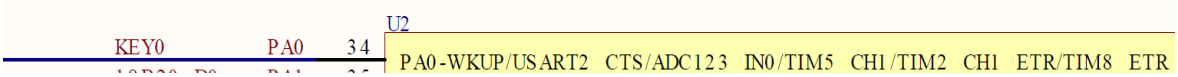


图 43　WKUP 按键引脚定义

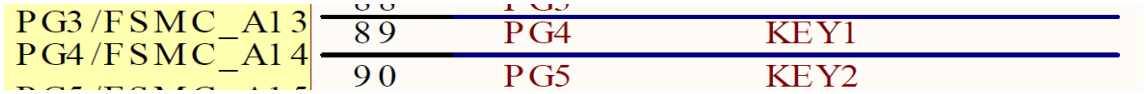


图 44　Key1 Key2 按键引脚定义

初始化按键外设，需要开启 GPIOA 和 GPIOG 时钟，将 WKUP 按键引脚默认下拉，Key1 和 Key2 默认上拉，核心初始化代码如下

```
//结构体定义
GPIO_InitTypeDef GPIO_InitStructure;
//开启 GPIOA GPIOD 时钟
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOG | RCC_APB2Periph_GPIOA,ENABLE);
/*************** GPIOG 设置 ******************/
//IO 设置
GPIO_InitStructure.GPIO_Pin =GPIO_Pin_4|GPIO_Pin_5;
//设置成上拉输入
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
//初始化 GPIOG
GPIO_Init(GPIOG,&GPIO_InitStructure);
/*************** GPIOA 设置 ******************/
// GPIOA0 初始化 WK_UP-->GPIOA.0　　下拉输入
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
//设置成输入，默认下拉
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
//初始化 GPIOA
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

初始化完成后，通过 GPIO_ReadInputDataBit 函数监听 GPIO 引脚电平变化，来检测键值。

3.3.5.3 SysTic 定时器与 NVIC

SysTic 定时器被困绑在 NVIC 中，用于产生 SYSTICK 异常。本设计使用的 STM32 基于 Cortex - M3 内核，不需要为操作系统产生滴答中断，该处理器有四个寄存器控制 SysTic 定时器。本系统采用外部时钟 HCLK 来完成延时函数的编写，秒延时函数为系统时钟的$\frac{1}{8}$。延时函数初始化核心代码如下。

```
//选择外部时钟　HCLK/8
SysTick_CLKSourceConfig(SysTick_CLKSource_HCLK_Div8);
//为系统时钟的 1/8
fac_us= SystemCoreClock / 8000000;
//代表每个 ms 需要的 systick 时钟数
fac_ms= (u16)fac_us * 1000;
```

3.3.6 RFID 射频芯片驱动实现

使用杜邦线将 MFRC522 模块与 STM32 连接，接线方式，如表 14 所示。

表 14　MFRC522 接线方式

MFRC522 引脚	STM32 引脚	作用
VCC3.3	VCC3.3	电源正极
GND	GND	电源负极
SDA	PA4	数据接口
SCK	PA5	时钟接口
MOSI	PA7	SPI 接口主出从入
MISO	PA6	SPI 接口主入从出
RQ	悬空	
RST	VCC3.3	

初始化 MFRC522 各个引脚，将 IO 速率设置为 50MHz，将 RST 引脚和 SDA 拉低，默认为关闭状态，核心代码如下

```
GPIO_SetBits ( macRC522_GPIO_RST_PORT, macRC522_GPIO_RST_PIN )
GPIO_SetBits ( macRC522_GPIO_CS_PORT, macRC522_GPIO_CS_PIN )
```

定义寻卡函数，寻找所有符合标准的 RFID 射频卡，核心代码如下

```
ClearBitMask(Status2Reg,0x08);
WriteRawRC(BitFramingReg, 0x07);
SetBitMask(TxControlReg,0x03);
ucComMF522Buf[0] = ucReq_code;
cStatus=PcdComMF522(PCD_TRANSCEIVE,ucComMF522Buf,1,ucComMF522Buf,&ulLen);
```

每张 RFID 射频卡共有 16 个扇区，每个扇区共有块 0、块 1、块 2 和块 3 组成，其中第 0 个扇区的块 0 在出厂时已经存放了厂商信息，并且不可更改，内置了 16 进制的卡号，并且每张卡的卡号都不相同。[16]利用这个特性来完成这次设计，编写以下的卡号读取函数。

```
int readPhyNumber ( void ){
    u8 ucArray_ID [ 4 ];
    u8 ucStatusReturn;
    if ((ucStatusReturn = PcdRequest (PICC_REQALL, ucArray_ID)) != MI_OK){
            ucStatusReturn = PcdRequest ( PICC  REQALL, ucArray  ID );
//ÈôÊ§°ÜÔÙ´ÎÑ°¿¨
    }
    if (ucStatusReturn == MI_OK){
        if (PcdAnticoll(ucArray_ID) == MI_OK){
            sprintf ( phyNumber, "%02X%02X%02X%02X", ucArray_ID[0], ucArray_ID[1], ucArray_ID[2],
ucArray_ID[3] );
            return 1;
        }
        return 0;
    }
    return -1;
}
```

3.3.7 ESP8266 网络通信实现

STM32 板载 WIFI 电路如图 45 所示

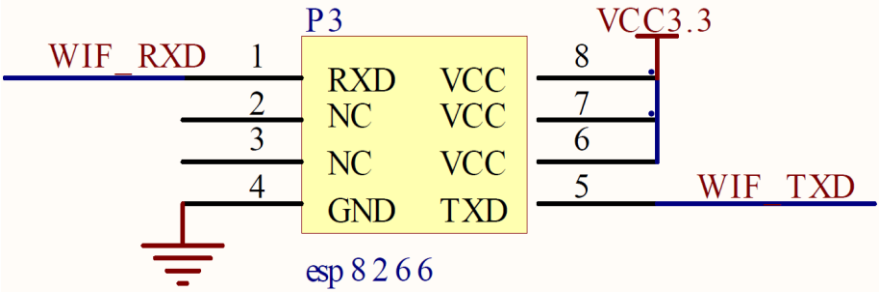


图 45　STM32 WIFI 电路

将 ESP8266 模块直插在 STM32 核心板上，由于 ESP8266 是通过串口 2 发送和接收数据的，需要初始化 USART2。USART2 引脚定义如图 46 所示

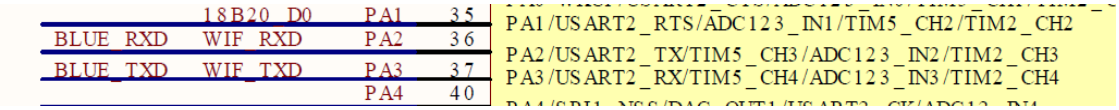


图 46　串口 2 引脚图

初始化 USART2，使能 GPIOA 时钟，使能 USART2 时钟将串口 2 的引脚复用推挽输出，配置收发模式和中断，核心代码如下。

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//使能 GPIOA,G 时钟
RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2,ENABLE);//使能 USART2 时钟
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;    //PA2
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP; //复用推挽
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);
RCC_APB1PeriphResetCmd(RCC_APB1Periph_USART2,ENABLE);//复位串口 2
RCC_APB1PeriphResetCmd(RCC_APB1Periph_USART2,DISABLE);//停止复位
//USART 初始化设置
USART_InitStructure.USART_BaudRate = bound;                    //波特率设置 一般设置为 9600;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;//字长为 8 位数据格式
USART_InitStructure.USART_StopBits = USART_StopBits_1;      //一个停止位
USART_InitStructure.USART_Parity = USART_Parity_No;         //无奇偶校验位
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;//无硬件数据流控制
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;     //收发模式    这里可以配置仅发或仅收
USART_Init(USART2, &USART_InitStructure); //初始化串口
//#if EN_USART2_RX          //如果使能了接收
NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=3 ;//抢占优先级 3
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;        //子优先级 3
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;           //IRQ 通道使能
NVIC_Init(&NVIC_InitStructure); //根据指定的参数初始化 VIC 寄存器
USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);//开启中断       接收到数据进入中断
USART_Cmd(USART2, ENABLE);                    //使能串口
```

通过串口接收函数 USART_ReceiveData 来接收串口收到的数据。

ESP8266 的波特率为 115200，通过串口来驱动 ESP8266 的运行，就需要将串口 2 的波特率配置为 115200,。ESP8266 模块是通过 AT 指令集来控制的，通过不同的 AT 指令，控制 ESP8266 完成对应的操作。本论文 ESP8266 模块工作流程如图 47 所

示。

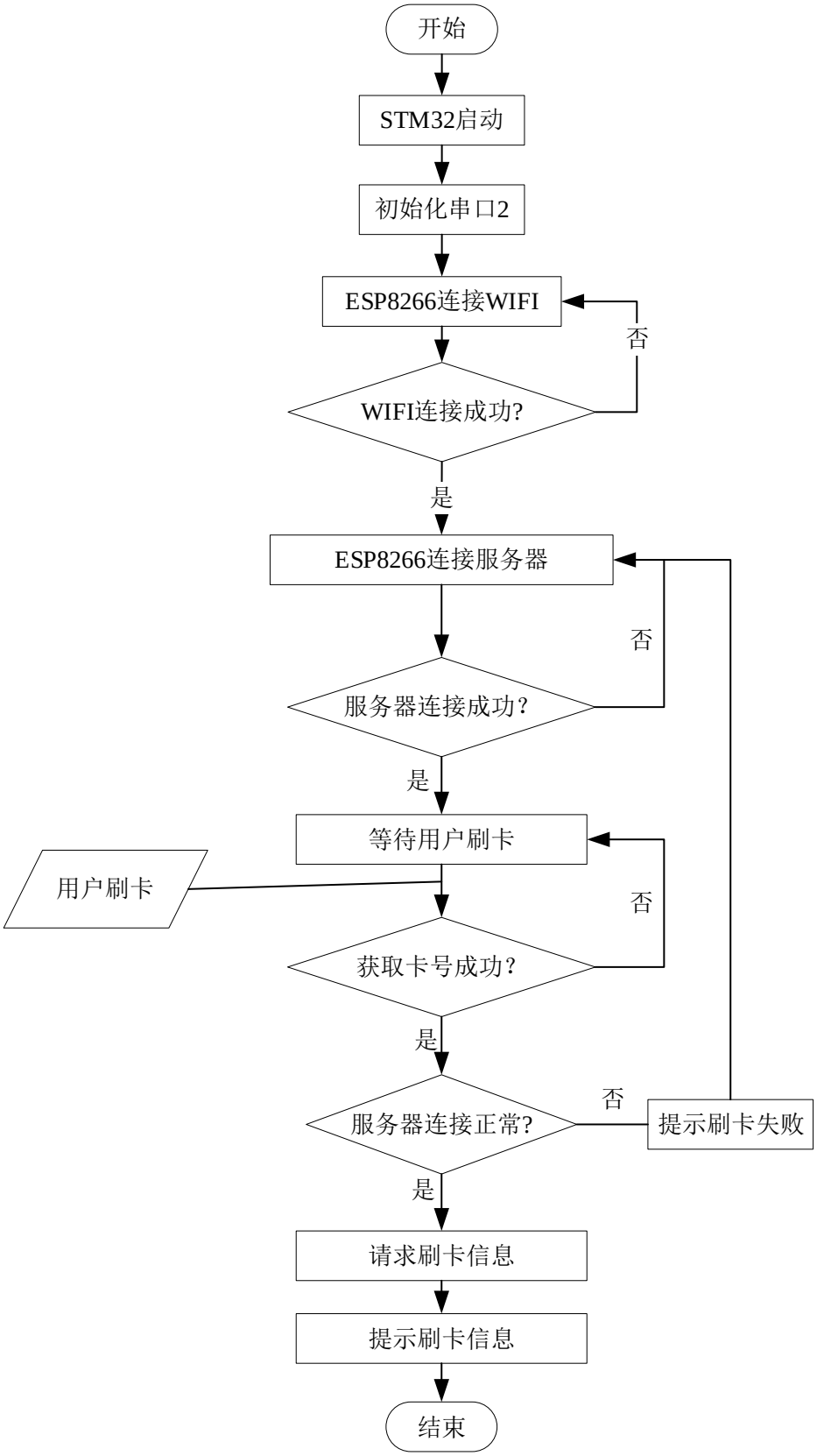


图 47　ESP8266 通信流程图

ESP8266 发送 HTTP 请求需要连接服务器，并且将读取到的卡号和当前的设备环境封装在请求行中携带过去，需要遵循 HTTP 协议，HTTP 请求有着固定的格式，如图 48 所示。[17]

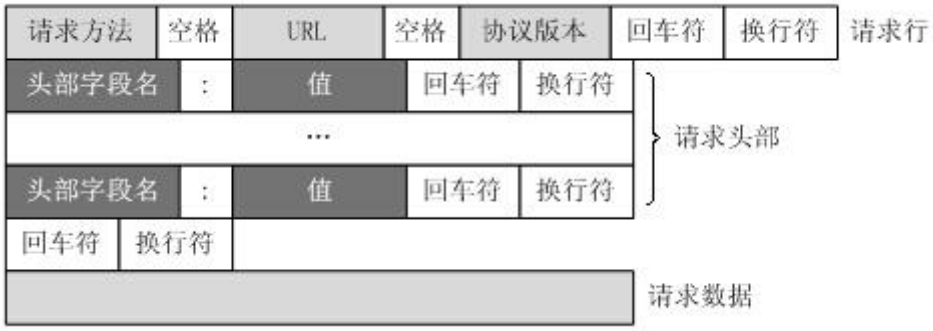


图 48　HTTP 请求格式

本设计 ESP8266 发送请求的后台接口属于 GET 请求，构成 GET 请求的最小请求格式需要包含请求行和 Host 主机地址，将获取到的卡号和设备参数拼接在请求地址后面即可完成 GET 请求，封装 HTTP 请求头的核心代码如下。

```
sprintf(get,"GET /rfid/swipe/%s/%s HTTP/1.1\r\nHost:192.168.31.198\r\n\r\n",type,cardId);
```

通过 AT 指令开启 ESP8266 的透传模式，将请求格式通过串口 2 发送，完成数据发送。[18]请求结束后，ESP8266 会将相应结果通过串口接收。发送请求的核心代码如下。

```
// 将要发送的字节数
Uart2SendStr("AT+CIPSEND\r\n");
delayMs(20);
if(strCompare("ERROR")){
    // 出现 ERROR，服务器掉线
    espConnectServer();
    return 0;
}
// 发送请求去刷卡
clearCache();
// 发请求
Uart2SendStr(get);
delayMs(800);
// 交给回调处理结果
call = 0;
response = esp8266CallBack();
flag = 1;
closeTansparentMode();
clearCache();
```

3.4 Web 模块实现

Web 服务端主要用到了 Java 语言，需要 JDK 的安装环境，首先需要安装 JDK，本设计使用的 JDK 版本为 JDK1.8，JDK 的安装非常简单，安装完成后需要配置环境变量，配置方式如图 11 所示，打开 Windows 运行框，输入 sysdm.cpl，确定后打开系

统属性。

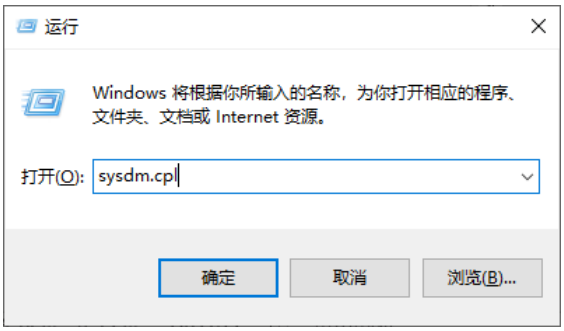


图 49　运行 sysdm.cpl

如图 12 所示，选择高级，环境变量。

图 50　设置环境变量

如图 13 所示，在系统环境变量下，依次新建环境变量，变量名 JAVA_HOME,变量值 JDK 安装路径下能看到 bin 的路径，变量名 CLASSPATH，变量值.。

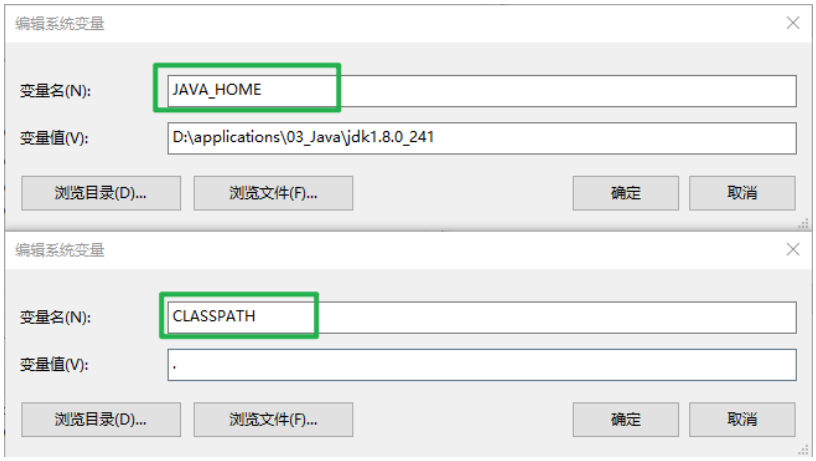


图 51　编辑环境变量

在 Path 变量下新建变量%JAVA_HOME%\bin，引用下 JAVA_HOME 的变量。如图 14 所示，运行 cmd，输入 java -version，能看到 Java 版本，JDK 安装成功。

图 52　JDK 安装成功验证

本系统采用的 Java 开发 IDE 为 IDEA，另外还使用了 maven、Tomcat、git 等工具部署管理项目

3.4.1 后端项目实现

本设计主要解决的是 STM32 网络模块与后台服务器的通信问题，后台服务器采用 Spring boot、Spring MVC 和 Mybatis 搭建。通过 Spring Initializr 来创建项目。使用 Mybatis-generator 逆向工程创建出数据库表对应的 Java Bean 以及数据增、删、改和查的 Mapper 接口和对应的 xml 实现文件。后端严格按照三层架构（控制层、服务层、数据交互层的方式开发）的结构开发项目，将项目所有的后台接口使用 swagger 创建出在线接口文档。

3.4.2 前端项目实现

本设计在实现了 STM32 与后台交互的情况下，还为用户提供了界面操作的功能。前端页面使用 vue elemen admin 模板创建，依赖于 vue 数据驱动的理念，同时搭配了 element 组件，丰富用户体验，使用 npm 作为前端项目第三方包管理器，使得前端项目的基础化搭建更为简单，通过组件的开发，将项目各个功能模块化，最终为用户提供了简单的图形化功能。

4 系统部署与应用

4.1 系统环境部署

4.1.1 硬件环境部署

4.1.1.1 ESP8266 配置

项目部署时，后端项目会使用服务器上的 IP 地址，硬件环境需要修改为服务器的地址信息，需要修改的代码为

```
u8 esp_tcp[] = "AT+CIPSTART=\"TCP\",\"47.103.215.243\",9999\r\n";
sprintf(get,"GET /rfid/swipe/%s/%s HTTP/1.1\r\nHost:47.103.215.243\r\n\r\n",type,cardId);
```

4.1.1.2 修改 STM32 启动模式

下载程序时，STM32 使用的是 ISP 下载程序，这种下载方式在下载完成之后，不需要通过复位键来启动程序，程序是直接开始运行的，在最后下载完程序后，将 STM32 启动模式修改为从 FLASH 启动，需要将 BOOT0 端口的跳帽接到低电平位置上，保证每次开发板都是从 FLASH 启动程序，RESET 按键也可以使程序从 FLASH 重新启动。

4.1.2 Linux 服务器环境部署

软件环境需要部署在有公网 IP 的操作系统下，外网设备通过公网 IP 地址能够正常访问服务器提供的数据，本论文将采用从阿里云购买的轻量级应用服务器来部署项目代码。服务器实例信息如图 53 所示。

图 53 服务器实例

阿里云的轻量级应用服务器面向学生优惠开放，可以自定义安装操作系统，支持 SSH 协议连接服务器，项目环境部署采用 Linux 操作系统下部署，Linux 操作系统为 Ubuntu 发行版 16.04。Ubuntu 作为服务器端操作系统，相比 Windows Server 有更好的性能，硬件要求低，系统可定制，软件安装简单，安全性高，更加适用于嵌入式环境的部署。

连接到远程服务器，需要使用 SSH 协议占用 22 端口号。目前在 Windows 操作系统下，优秀的 SSH 连接工具有 OpenSSH 和 XShell 等。XShell 工具属于商业软件，

不提供免费服务，本论文将使用 Shell 终端通过纯命令行的方式连接到服务器部署环境。

使用 SSH 协议连接到远程服务器的命令为

```
$ ssh lele@47.103.215.243
```

Ubuntu 更新软件源命令为

```
$ sudo apt-get upgrade
$ sudo apt-get update
```

4.1.3 后端环境部署

后端服务器环境属于 Java 项目，完全部署后端项目到服务器中，服务器需要具备后端项目核心运行环境。

4.1.3.1 后端项目运行环境部署

JAVA 项目解释器 JDK

Ubuntu 安装 JDK1.8 的相关命令为：

```
$ sudo tar -zxvf jdk-8u181-linux-x64.tar.gz -C /opt
$ cd /opt
$ mv jdk-8u181-linux-x64 jdk1.8.0
$ sudo vi /etc/profile
$ source /etc/profile
```

MySQL 数据库

Ubuntu 安装 MySQL5.7 的相关命令为：

```
$ sudo apt-get install mysql-server-5.7
$ service mysql start
$ mysql -u root -p
```

Redis 服务器

Ubuntu 安装 Redis 服务器的相关命令为：

```
$ wget http://download.redis.io/releases/redis-5.0.8.tar.gz
$ sudo tar xzf redis-5.0.8.tar.gz -C /opt
$ cd /opt
$ mv redis-5.0.8 redis
$ cd redis/
$ make
$ ./src/redis-server
```

4.1.3.2 后端项目部署

后端项目部署到服务器端，核心步骤主要有四部分：迁移数据库到服务器，修改项目启动配置文件，后端项目打包，服务器端运行项目。

（1） 服务端数据库部署

导出本地数据库为 SQL 脚本，在阿里云服务器的 MySQL 数据库中新建部署用的数据库，数据库相关配置，如图 54 所示。

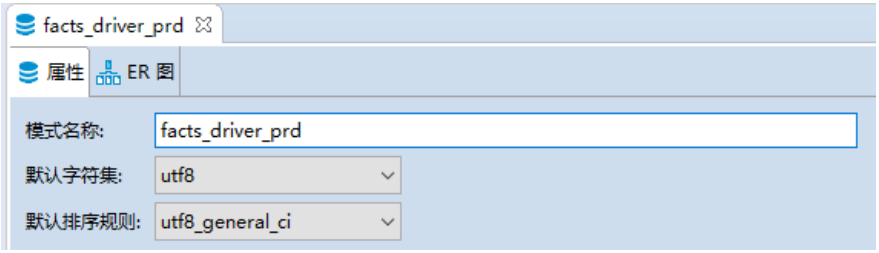


图 54　部署环境数据库信息

在生产环境的数据库中导入 SQL 脚本，添加基础数据，如图 55 所示。

rfid_card	InnoDB	1	16,384	RFID卡登记表
student_coach	InnoDB	1	16,384	学员教练关联表
swipe_rfid_record	InnoDB	1	16,384	刷卡记录表
sys_config	InnoDB	4	16,384	参数配置表
sys_dept	InnoDB	110	16,384	部门表
sys_dict_data	InnoDB	31	16,384	字典数据表
sys_dict_type	InnoDB	12	16,384	字典类型表
sys_job	InnoDB	4	16,384	定时任务调度表
sys_job_log	InnoDB	1	16,384	定时任务调度日志表
sys_logininfor	InnoDB	219	16,384	系统访问记录
sys_menu	InnoDB	2,001	16,384	菜单权限表
sys_notice	InnoDB	2	16,384	通知公告表
sys_oper_log	InnoDB	35	16,384	操作日志记录
sys_post	InnoDB	5	16,384	岗位信息表
sys_role	InnoDB	4	16,384	角色信息表
sys_role_dept	InnoDB	0	16,384	角色和部门关联表
sys_role_menu	InnoDB	0	16,384	角色和菜单关联表
sys_user	InnoDB	4	16,384	用户表
sys_user_info	InnoDB	2	16,384	用户信息表
sys_user_post	InnoDB	0	16,384	用户与岗位关联表
sys_user_role	InnoDB	0	16,384	用户和角色关联表
user_rfid	InnoDB	1	16,384	用户RFID关联表

图 55　生产环境数据库

新建生产环境的配置文件，将数据库的连接信息修改为服务器端的连接信息，如图 56 所示。

```
# 数据源配置
spring:
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource
    driverClassName: com.mysql.cj.jdbc.Driver
    druid:
      # 主库数据源
      master:
        url: jdbc:mysql://47.103.215.243:3306/facts_driver_prd?useUnicode=true&characterEncoding=utf8
        username: root
        password: lele
```

图 56　生产环境数据源配置信息

（2） 修改项目配置

修改生产环境下的后端项目启动端口号，尽量不要与 Linux 服务器中已有程序使用的端口号冲突，如图 57 所示。

```yaml
# 生产环境配置
server:
  # 服务器的HTTP端口，默认为8080
  port: 9999
  servlet:
    # 应用的访问路径
    context-path: /
  tomcat:
    # tomcat的URI编码
    uri-encoding: UTF-8
    # tomcat最大线程数，默认为200
    max-threads: 800
    # Tomcat启动初始化的线程数，默认值25
    min-spare-threads: 30
```

图 57　服务器端口配置

修改生产环境项目日志级别为 info，同时配置生成的日志文件的路径，如图 58 所示

```yaml
# 项目配置
driver:
  # 上传下载文件路径
  profile: ~/driver/driver_temp/files
  # swagger扫描哪些包中的接口
  swagger:
    package: com.leleplus.project.system.controller;com.leleplus.project.monitor.controller;
  # 日志文件路径
  log:
    path: ~/driver/driver_temp/logs
# 日志级别
logging:
  level:
    root: info
```

图 58　生产环境日志配置

（3） 项目打包

本项目是基于 Spring boot 创建的 maven 项目，maven 具备了优秀的依赖管理机制，通过 maven 生命周期的 package 阶段，即可轻松完成项目打包部署。首先指定打包方式为 jar,需要在 pom.xml 中加入代码

```
<packaging>jar</packaging>
```

然后配置 build 标签，使用 spring 提供的 maven 插件实现打包功能

```
<build>
    <finalName>driver-server-V${project.version}</finalName>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <!-- 配置 devtools 启用 -->
                <fork>true</fork>
            </configuration>
        </plugin>
    </plugins>
</build>
```

接着打开命令行，在能看到 pom.xml 的路径下执行以下打包命令

```
$ mvn clean package
```

如图 59 所示，maven 开始清理并打包项目，最终输出 BUILD SUCCESS 时，打包完成。

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.1.2:jar (default-jar) @ driver-manager-system ---
[INFO] Building jar: D:\Documents\driver\driver\back_end\driver-manager-system\target\driver-server-V2.2.1.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.2.5.RELEASE:repackage (repackage) @ driver-manager-system ---
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  24.888 s
[INFO] Finished at: 2020-04-13T17:35:36+08:00
[INFO] ------------------------------------------------------------------------
```

图 59　maven 打包项目

在项目的 target 目录下，可以看到打包之后的 jar 文件，如图 60 所示。

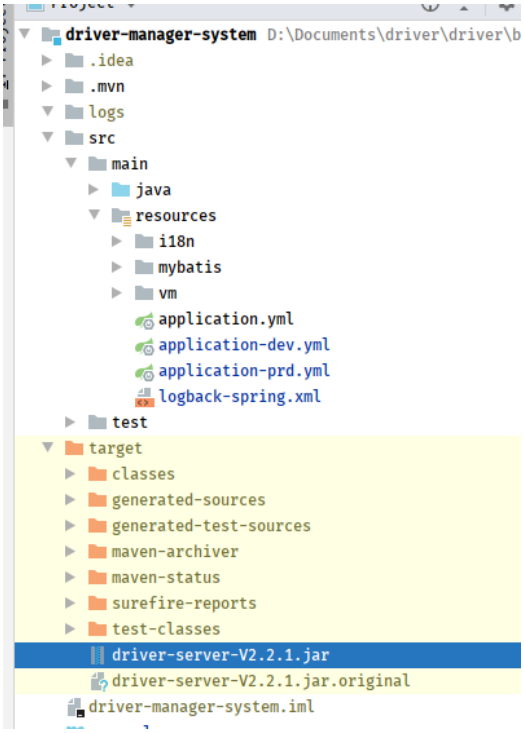


图 60　target 目录下的打包文件

（4） 服务器端运行

将打包好的 jar 文件，上传到服务器上，通过以下命令

```
$ scp target\driver-server-V2.2.1.jar lele@47.103.215.243:~/
```

连接到服务器端，启动 jar 项目，同时指定生产环境下的配置文件，并将任务改为后台进程，启动命令如下。

```
$ nohup java -jar -Dspring.profiles.active=prd driver-server-V2.2.1.jar &
```

查看启动过程的日志信息，使用命令

```
$ tail -f driver_temp/logs/driver-server.log
```

启动成功后，浏览器输入地址 http://47.103.215.243:9999/swagger-ui.html 访问接口，打开如图 61 所示页面，后端部署完成。

图 61　部署完成的接口页面

4.1.4 前端环境部署

前端环境需要部署在 Linux 服务器的 nginx 中，然后通过 nginx 的反向代理解决跨域问题，npm 可以轻松的构建 vue 项目，自动将 vue 项目文件转换为原生的 js、html、css 文件。

4.1.4.1 修改项目配置和设置 nginx 反向代理

在项目根目录下修改.env.production 文件，将 base_URL 路径改为生产环境的后端服务地址，并加上请求前缀，如图 62 所示

```
# 生产环境配置
ENV = 'production'

# 基于STM32的驾校管理系统/生产环境
 VUE_APP_BASE_API = 'http://47.103.215.243/driver-api'
```

图 62　前端生产环境配置接口地址

修改 vue.config.js 文件，配置静态资源的公共访问路径，添加如下代码

```
publicPath: process.env.NODE_ENV === "production" ? "/driver/" : "/",
```

配置 Linux 服务的 nginx 配置通过一下命令打开配置文件 nginx.conf

```
$ sudo vi /etc/nginx/nginx.conf
```

添加以下配置代码到 nginx 配置文件中

```
server {
        listen          80;
        server_name 47.103.215.243;

    location / {
        root /var/www/html;
        index index.html;
    }

    location ^~/driver-api/ {
        proxy_set_header Host $host;
        proxy_set_header   X-Real-IP            $remote_addr;
        proxy_set_header   X-Forwarded-For   $proxy_add_x_forwarded_for;
        proxy_set_header X-NginX-Proxy true;
        proxy_pass http://localhost:9999/;
    }
}
```

配置完成后重新加载 nginx 配置，使用命令

```
$ sudo nginx -s reload
```

4.1.4.2 前端项目部署

使用 npm 打包构建打包前端项目，将打包后的 dist 文件夹压缩后上传至 linux 服务器，解压缩到 nginx 部署根目录下，相关命令如下

```
$ npm run build:prod
$ scp dist.zip lele@47.103.215.243:~/
$ sudo mv ~/dist.zip /var/www/html/driver/
$ cd /var/www/html/driver
$ sudo unzip dist.zip
```

部署完成后，在本地客户机上打开浏览器，访问 http://47.103.215.243/driver,看到如图 63 所示页面，部署完成。

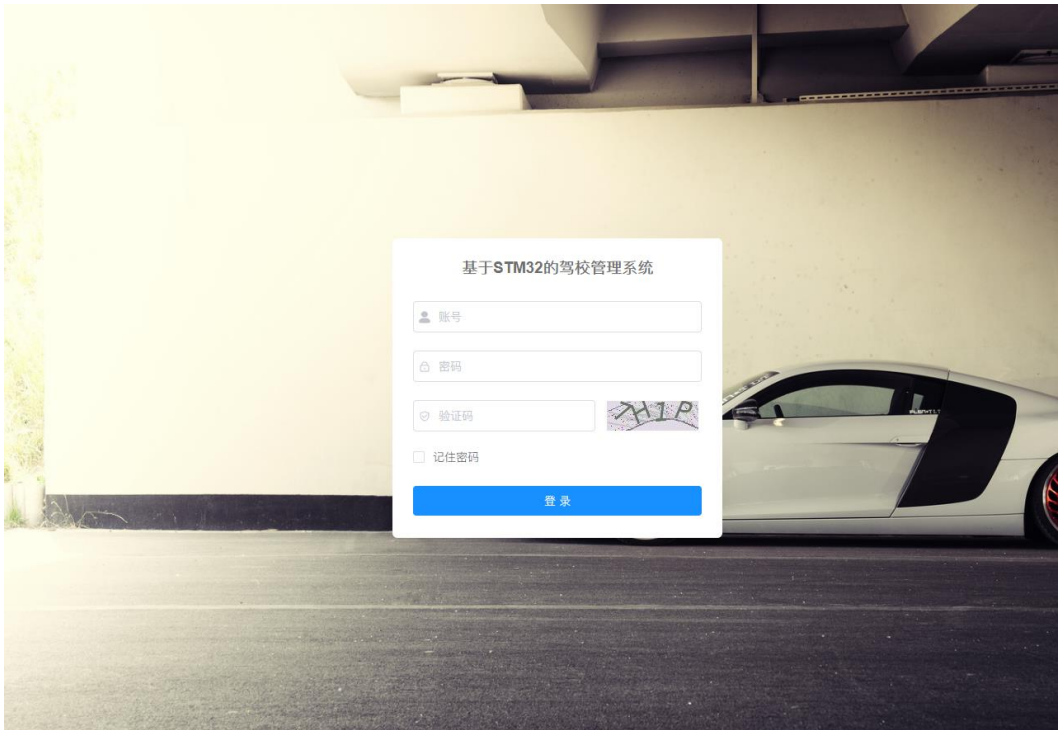


图 63　前端项目主页

4.2 系统测试和应用

4.2.1 系统整体测试

通过本地浏览器访问 Swagger 在线文档接口，接口页面如图 64 所示。

图 64　Swagger 在线接口文档

接口性能良好，没有报错，数据正常返回。

启动 STM32，查看输出日志信息，等到系统启动完成，系统指示灯正常闪烁，如图 65 所示。WIFI 连接连接成功，并且获取到了 IP 地址，连接服务器成功。串口打印出正常之后的菜单选项，如图 66 所示。

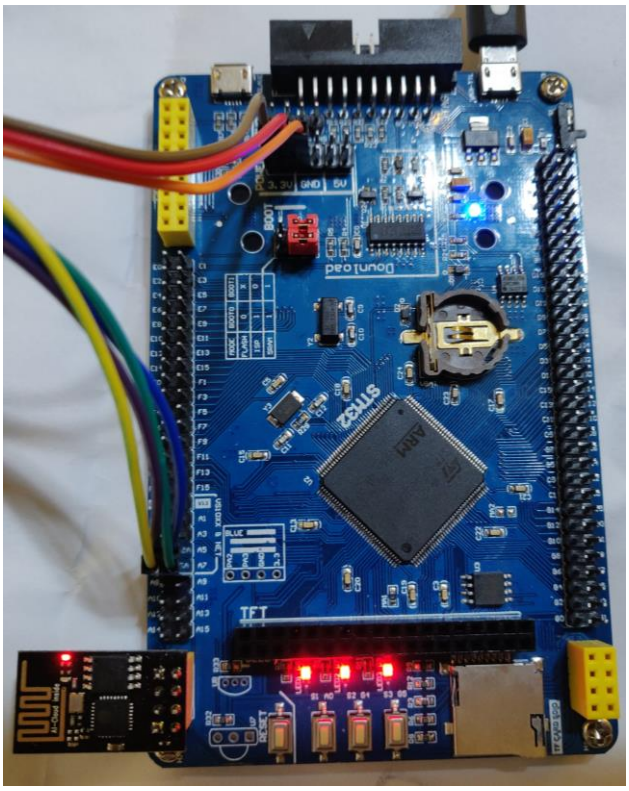


图 65　系统整体测试

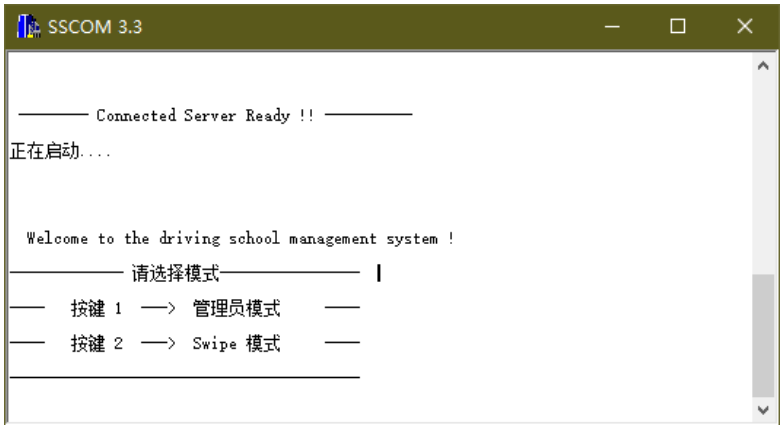


图 66　启动菜单

4.2.2 管理员注册模式测试

使用按键切换到管理员模式，此时管理员模式指示灯闪烁，并且串口打印出当前模式的提示信息以及退回主菜单的按键提示。系统状态如图 67 所示。串口提示信息如图 68 所示。

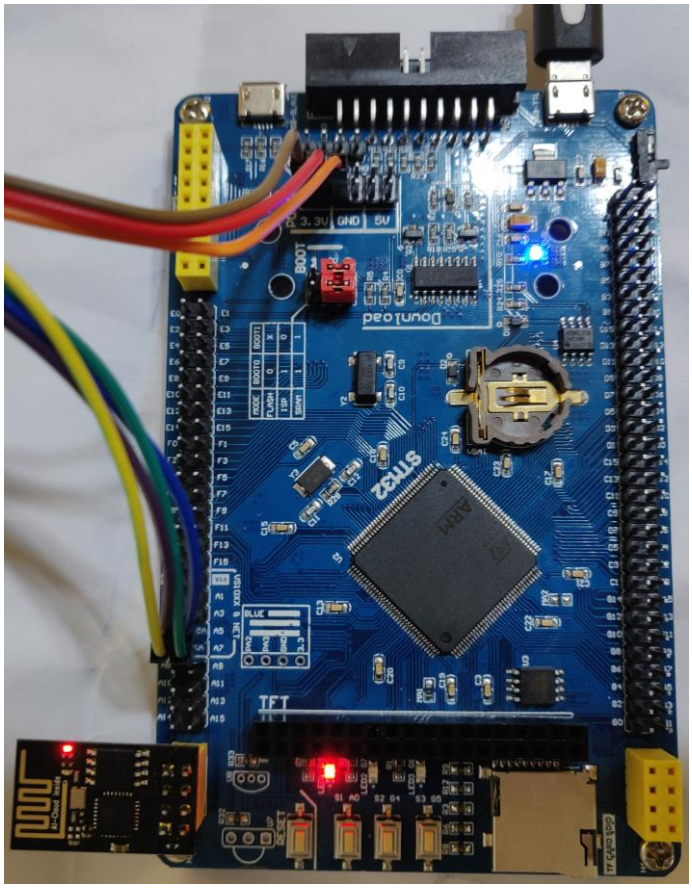


图 67　管理员模式测试

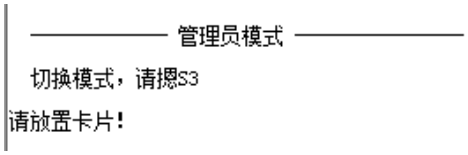


图 68　管理员模式提示信息

刷卡后会提示刷卡成功，并将数据保存在后台数据库中，同时将数据保存在 Redis 缓存中，缓存时间为管理员设置的时间。如图所示。

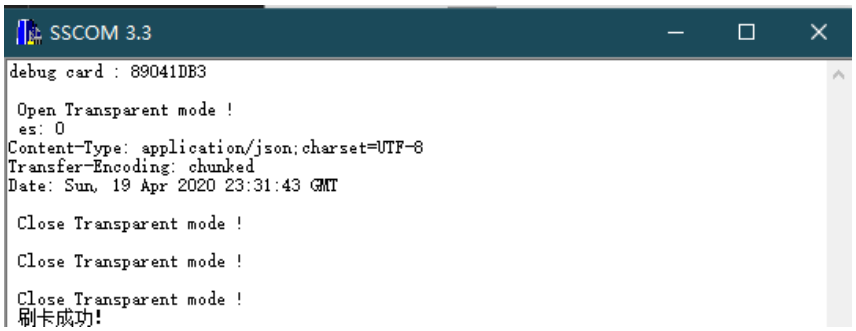


图 69　管理员刷卡成功

4.2.3 普通用户刷卡模式

通过 S3 按键返回主菜单，进入普通用户刷卡模式，刷卡模式指示灯闪烁，同时串口打印提示信息，如图 70 所示。

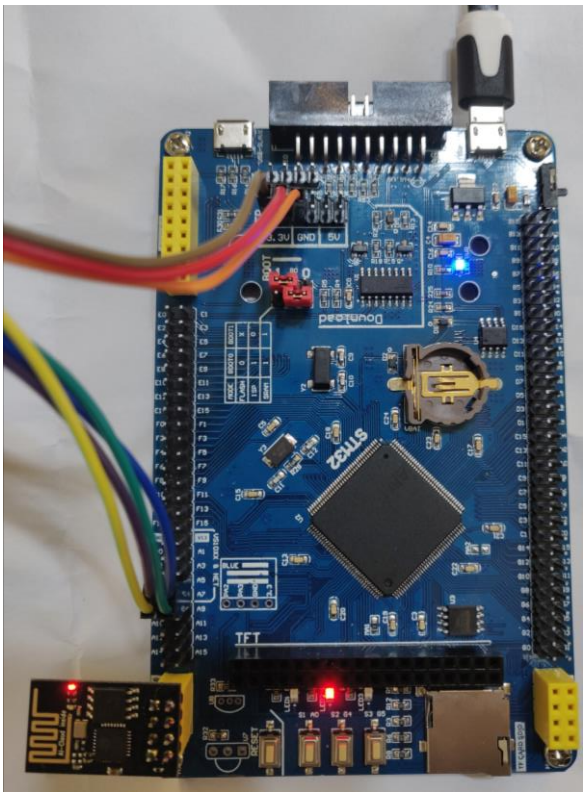


图 70　刷卡模式

刷卡后，后端收到刷卡请求，查询匹配用户数据，并记录刷卡信息

4.2.4 保持连接功能测试

系统每间隔十秒钟，自动会去检查与服务器的连接是否正常，如果失去了连接，会自动进行重连，直到连接到服务器为止，由于网络环境、设备等原因，重连次数以及连接时间受影响较大。经测试，此模块多数情况下，工作正常。

参考文献

[1] Luger, George. 人工智能：复杂问题求解的结构和策略. 由史忠植 等翻译 原书第 4 版. 北京: 机械工业出版社. 2004. ISBN 7-111-12944-X （中文）:142-143.

[2] 王保云. 物联网技术研究综述%Review on internet of things[J]. 电子测量与仪器学报, 2009, 023(012):1-7.

[3] 郭炯, 郝建江. 人工智能环境下的学习发生机制[J]. 现代远程教育研究, 2019, 31(5):4-7.

[4] 王铁流, 李宗方, 陈东升.基于 STM32 的 USB 数据采集模块的设计与实现[J]. 测控技术, 2009, 028(008):37-40.

[5] 肖茂兵, 卢振环. JavaEE 应用技术框架选型[J]. 华南金融电脑, 2006, 14(8):78-81.

[6] 孙书鹰, 陈志佳, 寇超. 新一代嵌入式微处理器 STM32F103 开发与应用[J]. 微计算机应用, 2010(12):61-65.

[7] 孙根. (0). 基于 SSM 框架的驾校管理平台的研究与实现. (Doctoral dissertation). 武汉邮电科学研究院. 2017:3-4.

[8] 罗国富, 刘海东, 姜宗品. 基于 RFID 的离散型制造物联实时数据采集系统的研究与开发[J]. 制造业自动化, 2015, v.37(21):141-146.

[9] 周伟,陈柳. Git 在软件版本管理实验教学中的应用[J]. 信息技术与信息化(5 期):97-100.

[10] 洪斯宝, 徐建明, 吴世名. 嵌入式数控系统 G 代码解释模块的设计与实现[J]. 机械设计与制造, 2012(11):42-44.

[11] 王芷郁, 王善伟, 曾胜艳. 基于 STM32F103ZET6 的无线语音控制小车设计与实现[J]. 电脑知识与技术, 2018, v.14(12):203-205.

[12] Ravi Kishore Kodali, SreeRamya Soratkal. *MQTT based home automation system using ESP8266*[C]. IEEE Region 10 Humanitarian Technology Conference. IEEE, 2016:15-19.

[13] 陈杰, 应时彦, 朱华. 基于 MFRC522 的 RFID 读卡器设计[J]. 浙江工业大学学报, 2014(06):30-34.

[14] 盛蒙蒙, 邱烨, 罗维, et al. 浅谈 C 语言中宏定义[J]. 中国科技纵横, 2009, 000(007):243.

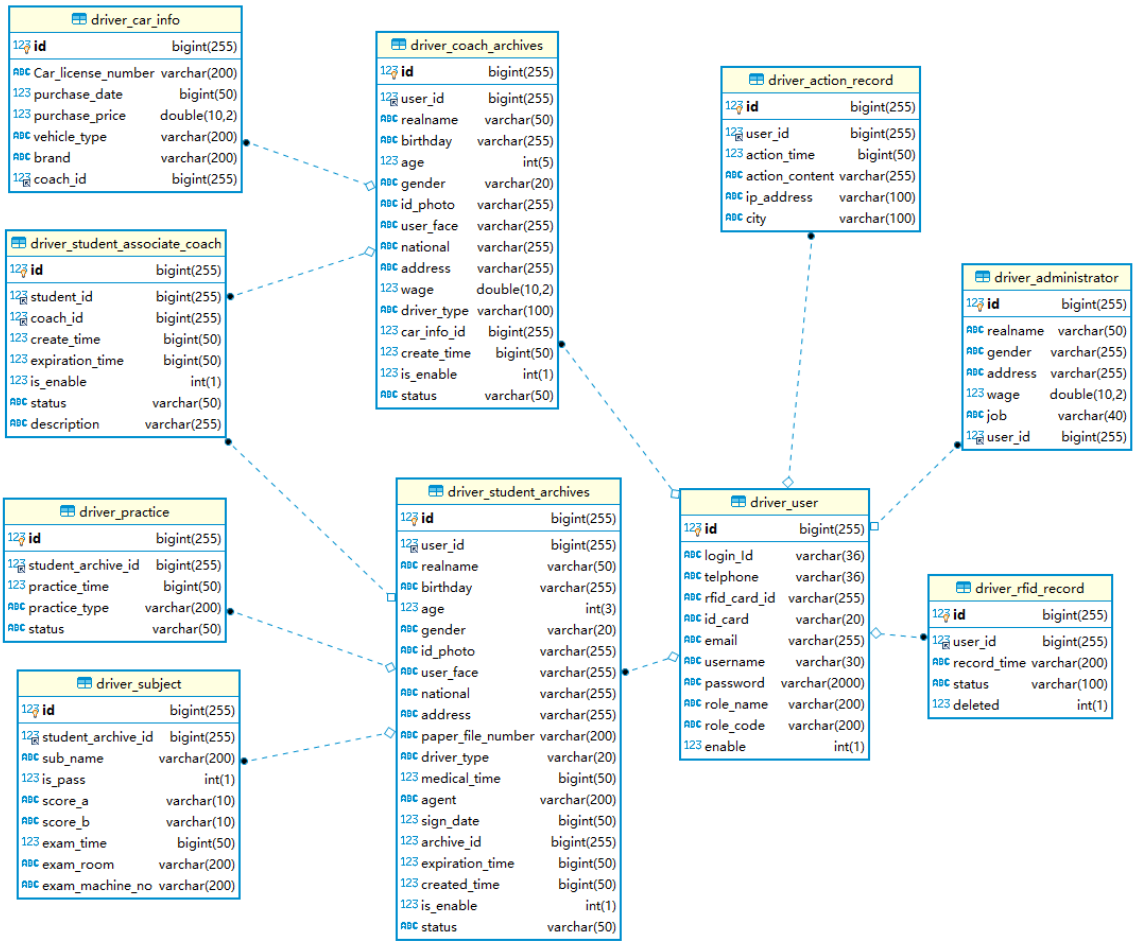
[15] Nation I S P . *Learning Vocabulary in Another Language*[J]. klett, 2001, 56(1):págs. 91-93.

[16] 张勇, 王锐. RFID 读卡器“作怪”干扰通信基站[J]. 中国无线电, 2016, 000(005):71.

[17] 王超, 胡晨, 刘新宁, et al. 嵌入式系统中 HTTP 协议的实现[J]. 电子器件, 2002, 25(1):93-96.

[18] 黄玉金, 杨越, 薛伟, et al. 无线模块的 AT 指令 UDP 透传设计[J]. 电子产品世界, 2018, v.25;No.349(01):38-41.

附录 1 数据库 E-R 关系图



附录 2 主要英文缩写语对照表

缩略语	中文全称	英文全称
ARM	微控制处理器	Advanced RISC Machine
ISP	在线系统编程	in-system programming
GPIO	通用输入输出端口	General Purpose Input Output
NVIC	向量中断控制器	Nested Vectored Interrupt Controller
LED	发光二极管	Light Emitting Diode
USB	通用串行总线	Universal Serial BUS

致　谢

时间转瞬即逝，大学生涯就要结束了，毕业设计也已经接近尾声，在老师的指导和各位同学的帮助下，我的设计最终如愿完成，感谢各位老师的付出和同学们的帮助。

在我个人看来，毕业设计就是一场考试，它帮我们总结了大学四年的收获，帮助我们认清自我，锻炼我们独立解决问题的能力。从最开始的选题，到查阅资料，编码实现，代码维护，项目部署测试等，锻炼的就是我们的编程和思维能力，同时也是考验我们的技术。即将步入社会、面临就业的我们，需要具备较硬的专业知识以及独立编码的能力。毕业设计在大学生活完美结束之前，让我们进行一次锻炼，对自己有一个认知。本此设计的完成，让我对编程有了一个全新的认识，编程并没有我们想象的那么复杂，它需要你去不断的研究，测试，读文档，查资料来探索语法的奥秘，每门语言都是相通的，各有各的规则和原理，需要你自己去实验，去理解。物联网是一个新奇的东西，在将来的生活中，我们将时刻接触到它，他将出现在我们生活的方方面面。

物联网专业是一个新兴专业，不仅要求我们有扎实的编程功底，还要对电路知识非常熟悉。本次在 RFID 读写卡设计中，手动焊接，查找电路图，手动编码，都是对我们大学三年来知识的一个综合性应用，充分考验了我们的独立设计能力。毕业论文是一项相对耗时的任务，也是我们大学生涯中接触的第一个独立研究的项目。完成这次任务的过程可能是艰辛的，但它带给我们的收获将成为我们即将步入社会的垫脚石，我相信我能完成任务。

在文章的结尾，我想感谢大学四年帮助照顾过我的老师和同学，感谢你们一路的陪伴和支持，希望大家都能始终坚持自己的梦想，永不放弃！

2020 年 4 月 20 日于陇东学院