



中國石油大學 (华东)
CHINA UNIVERSITY OF PETROLEUM

本 科 毕 业 设 计（论文）

题 目：基于 Lora 的暗渠危险气体检测系统

学生姓名：刘静远

学 号：1507040220

专业班级：物联网工程 15-2 班

指导教师：宋继志

2019 年 6 月 13 日

基于 LoRa 的暗渠危险气体监测系统

摘 要

暗渠中一般都会充斥着各种有毒有害甚至易燃易爆气体，每年因为危险气体发生的危险事故也屡见不鲜。本毕业设计针对这种情况，运用传感器感知技术，Lora 窄带扩频通信技术，Arduino 单片机开发技术以及新浪云 sae 框架下服务器及数据库的搭建，实现了远程暗渠有害气体的感知，传输，数据可视化以及危险警告功能。本系统主要用到的编程语言为 C，python, php, html, javascript，开发工具为 Arduino IDE，pyCharm，云端服务器为新浪云。本论文详细介绍了传感器采集原理，lora 通信过程，网关服务器脚本的编写以及网页端可视化内容的具体实现。

本次毕业设计的关键任务是通过网页可视化界面实时检测节点危险气体的浓度，并进行危险预警，实现危险可燃气体的检测与防护。

关键词：Arduino；lora；危险气体；动态检测

Detection System of Dangerous Gas in Underground Channel Based on LoRa

Abstract

Underground canals are usually filled with toxic, harmful and even flammable and explosive gases. Every year, dangerous accidents caused by dangerous gases are also common. In view of this situation, this graduation project uses sensor sensing technology, Lora narrowband spread spectrum communication technology, Arduino MCU development technology and Sina cloud SAE framework server and database construction to realize remote dark channel harmful gas sensing, transmission, data visualization and danger warning functions. The main programming languages used in this system are C, python, php, html, javascript, development tools are Arduino IDE, pyCharm, and cloud server is Sina cloud. This paper introduces in detail the principle of sensor acquisition, Lora communication process, the writing of gateway server script and the concrete realization of Web visualization content.

The key task of this graduation project is to detect the concentration of dangerous gases in nodes in real time through the visual interface of web pages, and carry out dangerous early warning, so as to realize the detection and protection of dangerous flammable gases.

Keywords: Arduino; lora; dangerous gases; dynamic detection

目 录

第 1 章 引言	1
第 2 章 基本设计	2
2.1 Arduino 单片机系统开发	2
2.1.1 Arduino 单片机系统	2
2.1.2 节点电压采集模块	2
2.1.3 MQ-9 气敏传感器模块	3
2.2 LoRa 数据传输系统	4
2.2.1 LoRa 通信基本原理	4
2.2.2 SX1278LoRa 通信模块	5
2.3 Web 云服务器应用	6
2.3.1 新浪云 SAE 概述	6
2.3.2 云服务器的搭建与维护	6
第 3 章 主体设计	9
3.1 概要设计	9
3.1.1 硬件软件平台	9
3.1.2 系统实现流程	11
3.1.3 创新点与难点分析	12
3.2 详细设计	12
3.2.1 节点信息采集及封装	12
3.2.2 LoRa 通信流程	14
3.2.3 网关数据解包及上传	15
3.2.4 云数据库数据处理	16
3.2.5 注册登录界面的实现	17
3.2.6 信息可视化的实现	19
3.2.7 短信预警功能的实现	20
第 4 章 系统测试	22

4.1 软件及硬件测试.....	22
4.1.1 传感器节点及 LoRa 发送模块测试.....	22
4.1.2 LoRa 接收模块测试.....	22
4.1.3 网关数据上传测试.....	23
4.1.4 数据库数据写入测试.....	23
4.1.5 数据可视化测试.....	24
4.1.6 短信功能测试.....	24
4.2 测试总结.....	25
第 5 章 结论.....	27
致 谢.....	28
参考文献.....	29
附 录.....	30
附录 A 传感器节点数据封装及发送程序.....	30
附录 B 网关 Python 程序.....	34

第 1 章 引言

近些年随着经济的快速发展，城市化水平逐渐升高，地下管道作为城市必不可少的元素，越来越成为城市化过程中需要顾及的重要一环，城市排水系统，排污系统更是维持城市运转的重要因素。但是，近些年由城市地下暗渠引发的危险事故也层出不穷，屡见不鲜，越来越多的人也开始逐渐将注意力转移到暗渠有害气体检测上，研究发现，记录在册的大部分城市暗渠引发的危险事故大都是可燃气体引发的爆炸问题，据调查研究所知，由于暗渠中生化环境的特殊性以及暗渠中流体的特殊性，使得相关物质极易在微生物的作用下发生化学反应生成甲烷，硫化氢等可燃气体，甲烷，硫化氢等可燃气体的浓度达到一定的阈值再加之光照，温度，明火等外部条件极易产生爆炸事故，产生巨大的安全隐患。

据材料记载，2017 年重庆市因为暗渠气体燃烧爆炸引发的安全事故就有十几起，造成数千万的财产损失，假如将这个范围扩充到全国，可以大致估计一下因为暗渠气体爆炸引起的事故造成的财产损失可能数以亿计。然而，目前为止还没有相关的专门针对暗渠可燃有害气体的检测的相关理论系统，可以说，暗渠有害气体检测在城市化进程如此之快的今天仍然存在巨大的缺口，因此，一套专门针对暗渠可燃有害气体的检测预警系统迫在眉睫，特别是完全自动化，数字化，低功耗的暗渠有害气体检测系统具有广阔的发展前景。

城市暗渠具有排布密集，环境复杂的特点，因此采用传统的人工检测方案显然是不安全，也不现实的，基于 lora 通信的暗渠有害气体检测检测系统具有的重要意义，它不仅可以填补城市暗渠有害气体检测缺口，更会为国家和社会避免不必要的生命财产损失，基于 lora 通信的暗渠有害气体检测检测系统具有低功耗，低成本，高可靠性，高响应速度等特点，并可以通过后端数据实时汇总进行数据分析，制定更为可靠有效的解决方案，这些都是传统的传感器系统所不具备的。城市地下暗渠系统就像是城市的血液循环系统，而基于 lora 通信的暗渠有害气体检测检测系统的实施必然会成为维持城市“血液”正常运转的重要部分，对城市的运转及发展具有不可替代的作用，有重大的发展意义和光明的发展前景。

第2章 基本设计

2.1 Arduino 单片机系统开发

2.1.1 Arduino 单片机系统

Arduino 单片机系统是一类单片机系统的统称，是一类可用于简单软硬件开发的开源硬件平台，具有跨平台，简单易上手，开放性高，发展迅速等特点，一套完整的 Arduino 系统包括 Arduino 功能板以及用于程序编写的 Arduino IDE。

Arduino 单片机系统具有跨平台优势，由于 Arduino IDE 的加持使得 Arduino 的开发可以在 Mac, Windows, Linux 等主流操作系统上进行，而其他硬件开发平台大都只支持 Windows 平台的开发，这是 Arduino 相对于其他开发平台的重要优势。

Arduino IDE 基于 processing IDE 开发，与 Keil, IAR 等主流开发软件相比显得十分迷你，对于刚刚接触硬件开发的新手十分友好，Arduino 程序主要采用 C++ 编写，并且针对不同的硬件，世界各地的开发者们封装了各式各样的函数库，大大降低了新手进行程序开发的难度。除此之外，Arduino IDE 自带串口监视器及串口绘图器，可以直接在 IDE 中可视化的查看串口的数据，与传统的开发软件相比更加方便调试。

开源对促进软硬件系统的完善具有重要意义，Arduino 作为一个开源硬件开发平台，不仅实现了软件上的开源，还实现了硬件原理图，电路图的开源，开发者可以在此基础上进行二次开发，打造适合自己的开发板及函数库，这是 Arduino 近些年飞速发展的重要原因。

本系统采用 Arduino Uno 开发板，在 Arduino IDE 上基于 C++ 进行开发，引用了 Arduino 中串口及 Lora 函数库，极大的简化了传感器节点数据采集，汇总及数据通信的程序开发过程。

2.1.2 节点电压采集模块

对于任何一个电子器件，其功能的实现以及效率都与供电电压有着密不可分的关系，本系统主要应用于地下暗渠，用于危险气体的检测，传感器节点的灵敏度以及传感器节点是否可以及时将数据进行上传都关系着系统的正常运转。

为了在实际应用中避免传感器节点信息检测不准，信息上报不及时等众多隐患，本系统引入了传感器节点电压检测模块，图 2-1 为电压检测模块的电路原理图。

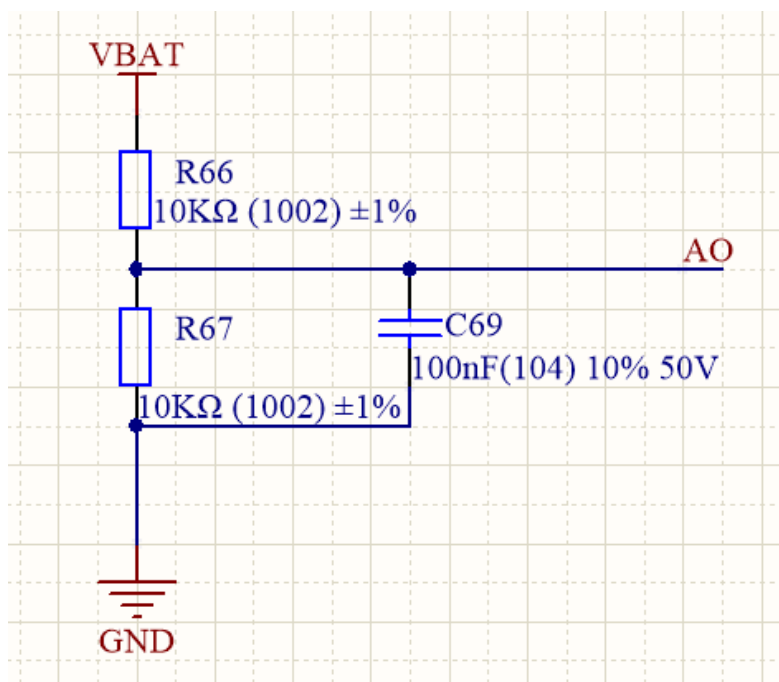


图 2-1 节点电压检测模块

如图可知，此模块主要由两个十千欧电阻以及一个一百纳法的滤波电容构成，模块有三个接线端，一端直接接到电池正极，另一端接地，信号输出端接到单片机模拟口，由两个电阻对电池电压进行分压，由于本模块采用 9V 电池供电，因此 A0 引脚分得一半的电池电压，输出值的正常范围大约在 4.20V-5V 之间，除此之外的电压值则可判定为异常电压。之后单片机会将电池电压作为一项参数封装与既定的数据包中进行上传，网关对收到的数据包进行拆解，判断电池电压是否属于正常范围，对处于正常范围的节点数据进行上传，丢弃不符合规定电压的数据包，实现了节点电压的实时监测，确保了数据的准确性以及系统的稳定性。

2.1.3 MQ-9 气敏传感器模块

MQ-9 气敏传感器是一款比较常用的可燃气体传感器，MQ-9 气体传感器所使用的气敏材料是在清洁空气中电导率较低的二氧化锡(SnO_2)。采用高低温循环检测方式，低温状态（1.5V 加热）用于检测一氧化碳，传感器的电导率随空气中一氧化碳气体浓度增加而增大，高温状态（5.0V 加热）检测可燃气体甲烷、丙烷并清洗低温时吸附的杂散气体。使用简单的电路即可将电导率的变化，转换为与该气体浓度相对应的输出信号。MQ-9 气体传感器对一氧化碳、甲烷、液化气的灵敏度高，这种传感器可检测多种含一氧化碳及可燃性的气体，是一款适合多种应用的低成本传感器。

MQ-9 的供电电压为 5V，具有双路信号输出，TTL 信号输出为低电平有效，模拟量信号输出为 0-5V 电压信号，浓度越高电压越高，除此之外，MQ-9 较高的灵敏度及快速的相应恢复特性，适用于持续性检测。

本系统采用 MQ-9 传感器，用节点电压来指示节点数据是否有效，用节点模拟量输出来判断节点有害气体浓度，节点有害气体浓度（主要为烷类浓度）正常范围值为 20-70，节点浓度超过 70 则可初步判断节点有害气体浓度过高，低于 20 则可判断节点电压电源电压处于较低状态，应及时更换电源，具有较高的可操作性及可靠性。

2.2 LoRa 数据传输系统

2.2.1 LoRa 通信基本原理

LoRa 是英语 Long Range Radio 的缩写，翻译成中文就是远距离无线电，从名字可以得出 LoRa 的两个性质，一个是适用于较远距离，而是 LoRa 属于无线电通信的范畴。LoRa 是由瑞士 semtech 公司提出的低功耗局域网无线标准，与传统的无线电通信相比，LoRa 最大的优势就是对于功耗的控制，在传统的概念中，低功耗则意味着传输速率的下降或者传输距离的缩短，而 LoRa 则对低功耗远程通信进行了突破，兼顾了功耗及通信距离两个衡量通信技术水平的重要指标，除此之外，LoRa 通信的传输速率也比较令人满意，可以满足当下物联网应用中的大部分传输任务(视频码流的实时传输除外)，因此 LoRa 在物联网领域，例如智能家居，智能交通等应用案例中具有很大的发展潜力。

LoRa 设计适用于用于功耗较低，距离较远，且对速率要求不高的传输场景，LoRa 的速率范围可以从 0.3kbps 到 50kbps，LoRa 采用 ADR(adaptive data rate)调度算法，此算法可以根据环境应用及用户需求修改数据的发送带宽及功率。LoRa 使用扩频 CSS 技术来发送数据，可有效对抗多普勒频移效应。在工作频率的选择方面，LoRa 使用未授权免费的公共 ISM 频率，为了减少设备间相互的信号干扰，终端采用伪随机的设备接入方式。LoRa 调制的最大耦合损耗(MCL)高达 148 dB-20 dB，以便实现距离可达数千米的通信范围，并一定程度上扩展了网络容量。LoRa 具有六个可以自适应数据速率的扩展因子，开发者可以根据环境及应用需要选择合适的扩展因子进行数据传输，这也可以使得 LoRa 能够在同一个信道频率上进行多个不同扩展的信号发送。

与传统的 FSK，OOK 调制技术相比，LoRa 调制解调器采用专利扩频调制和前向纠错技术，这也一定程度上扩展了 LoRa 链路的覆盖范围，同时提高了链路的鲁棒性。在

实际开发中,开发人员可以调整扩频因子和纠错率两个参数,从而平衡通信的带宽占用,通信速率,空中数据包的存活时间,以及抗干扰性等众多参数值指标。

2.2.2 SX1278LoRa 通信模块

SX1278LoRa 收发器主要采用 LoRa 远程调制解调器,用于超长距离扩频通信,抗干扰性强,能够最大限度降低电流消耗。SX1278LoRa 收发器还支持 WM-Bus 和 IEEE802.15.4g 等系统的高性能 FSK 模式,与同类器件相比,SX1276 在大幅降低功耗的基础上,还显著优化了相位噪声,选择性,接收机线性度,三阶输入截去点等各项性能,最大的链路预算可达 168dB,具有低至-148dBm 的高灵敏度,卓越的抗阻塞也行及 9.9mA 低接受电流和 200nA 的寄存器保持电流。SX1278 这些卓越的特性使得 SX1278 在自动抄表,智能家居,工业监视及安防系统中有着广泛的应用。

SX1278LoRa 调制解调器具有八种操作模式,模块所提供的功能和寄存器的访问范围取决于选定的操作模式,下面简单介绍 LoRa 的几种操作模式。

睡眠模式:又称为低功耗模式。此模式下,仅 SPI 和配置寄存器可以访问,其他寄存器都无法访问,FSK/OOK 模式与 LoRa 模式切换也需要经过睡眠模式。

待机模式:此模式下,射频和 PLL 部分会被关闭,晶振及 LoRa 的基带处于开启状态。

TX 模式:当模块切换为 TX 模式时,SX1278 的所有与发送相关的功能模块会被打开,功率放大器被打开,进行数据包发送,待数据发送完成后,切换为待机模式。

RX 连续模式:当模块切换为 TX 模式时,SX1278 的所有与接收相关的功能模块会被打开,进行数据接收。

RX 单一模式:SX1278 将打开接收所需的所有模块、在收到有效数据包前保持此状态、随后切换回待机模式。

CAD 模式:CAD 模式下,设备进行信道检测,用于检测 LoRa 的前导码信号。

在使用 SX1278 模块还有一点需要注意的是无论从 LoRa 发送模式到接收模式,还是从 LoRa 接受模式到发送模式,都必须中间先经过待机模式,否则会转换失败,导致 LoRa 通信失败。

本系统在使用 RX1278 模块时主要要用到了 TX 模式 RX 单一模式和待机模式,由传感器节点采集数据进行发送,再由汇聚节点收集数据统一上传网关的单向发送接受模式。

2.3 Web 云服务器应用

2.3.1 新浪云 SAE 概述

新浪云 SAE 是新浪信息研发中心推出的面向云计算和云服务器的分布式 Web 应用业务开发托管、运行平台。新浪云与其他云服务器相比具有高可靠，高拓展，面运维等显著特点，新浪云支持 php，python，java 等当下主流的 Web 语言，可满足多种应用场景。新浪云还提供了共有云数据库的接口，可满足小型用户的数据库操作的基本要求。除此之外，新浪云还支持代码包上传，开发者可以将本地开发好的代码包直接打包上传到新浪云服务器，新浪云经过一系列对用户透明的解压，编译即可成功部署到云端运行，十分简洁方便。

然而，新浪云 SAE 与其他云服务提供商最大的区别在于开发环境的集成度，在用户创建新浪云 SAE 时便会选择开发语言及环境，这些都会由云平台自动配置部署，这与其他云平台使用命令行一步一步部署开发环境相比，新浪云 SAE 对于刚刚接触 Web 开发的新用户是十分友好的。本系统所采用的开发环境是基于 php7.0，数据库采用新浪共享 mysql 数据库，代码管理采用 FTP 的方式。

2.3.2 云服务器的搭建与维护

本次毕业设计的服务器主机是由新浪云提供的基于 php7.0 开发环境的，500M 代码空间的云主机。下图为创建新浪云 SAE 的具体操作。云服务器配置如图 2-2。

应用列表 / 创建新应用

开发语言 ☒ PHP ☐ Python ☐ Java ☐ Go ☐ NodeJS ☐ Docker ☐ 容器虚拟机

代码运行环境 ☒ 云空间环境 ☐ 标准环境 [查看区别](#)

PHP版本 ☐ 5.3 ☐ 5.6 ☒ 7.0

代码管理方式 ☒ FTP

代码空间套餐 ☐ 500M ☒ 1G ☐ 2G ☐ 5G ☐ 50G

代码空间套餐时长 ☐ 一月 ☐ 二月 ☐ 三月 ☐ 四月 ☐ 五月 ☐ 六月 ☐ 七月 ☐ 八月 ☐ 九月 ☒ 一年 (优惠中) ☐ 两年 (优惠中) ☐ 三年 (优惠中)

套餐自动续费周期 ☐ 1个月 ☒ 1年 ☐ 2年 ☐ 3年 ☐ 不自动续费

* 一级域名 请输入一级域名前缀 仅允许由数字、字母组成 长度为4到18位

图 2-2 云服务器配置

如图可见，新浪云 SAE 的创建过程是完全的可视化操作，用户只需根据自己的需求尽心选择便可完成云平台的搭建，十分简单方便。

新浪云还提供有在线的代码包管理界面，可进行代码包上传，下载，编译等操作，界面如图 2-3。

版本列表

版本	版本访问链接	调试状态	XHProf	操作
1 默认版本	http://1.gasdetect.applinzi.com http://gasdetect.applinzi.com https://gasdetect.applinzi.com	✔ 开启 关闭	! 未开启 开启	在线编辑 上传代码包 下载 清空

任务列表

任务编号	添加时间	最后更新时间	任务类型	状态
19167096	2019-05-14 15:01:37	2019-05-14 15:01:42	代码导出	✔ 执行成功
18869982	2019-05-06 21:22:58	2019-05-06 21:23:04	代码导出	✔ 执行成功

图 2-3 新浪云代码管理

除此之外，新浪云还提供有专门的代码在线编辑工具，可以在云端进行代码的编辑，修改及运行，适用于后期对云应用进行简单的更新与修改，无需从本地重新上传代码包，大大节省了流量开支。图 2-4 为新浪云 SAE 自带的云代码编辑器。

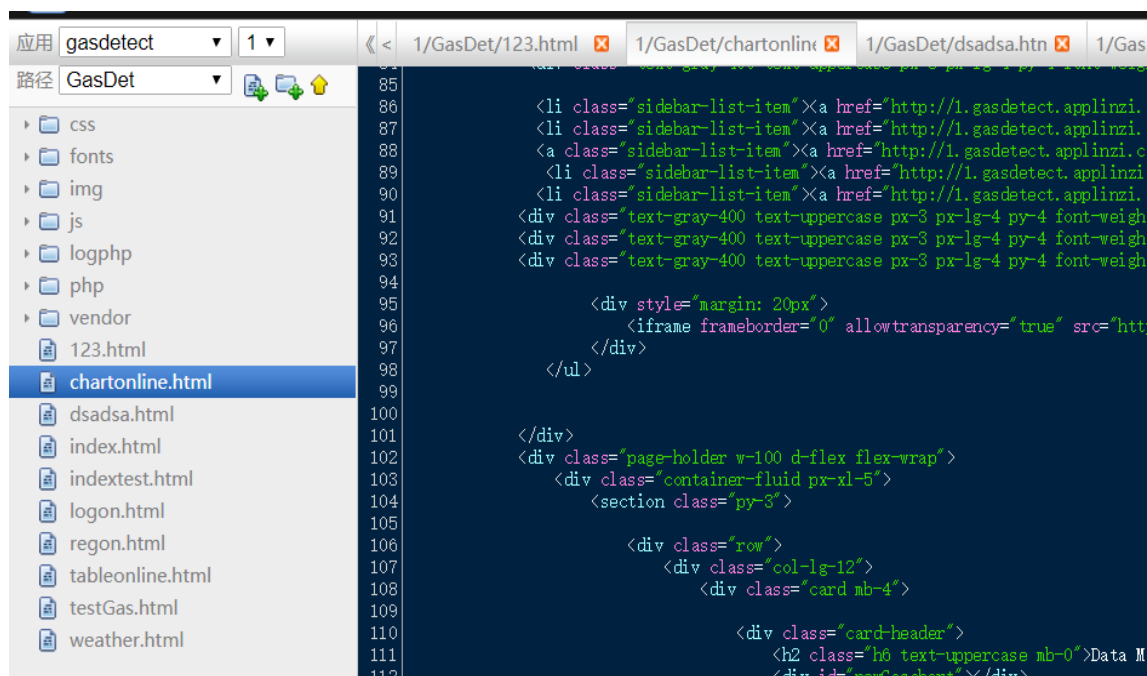


图 2-4 新浪云编辑器

本系统所采用的代码编辑方式主要使用 HBuilderX 进行本地编写，基于 Apache 进行网络及数据库环境的搭建，进行 Web 服务的本地测试，本地测试完成之后将工程文件进行打包上传到新浪云服务器部署运行，云数据库连接及调试的代码编写则使用了新浪云 SAE 自带的云代码编辑器。

第3章 主体设计

3.1 概要设计

3.1.1 硬件软件平台

本次毕业设计采用的硬件平台主要为 Arduino Uno 开发板，PC 机和新浪云服务器。本系统初步采用两块 Arduino Uno 开发板，一块位于传感器节点处，用于控制传感器节点进行数据的采集，并进行数据封装发送，另一块与网关（PC）相连，作为汇聚节点接收传感器节点传送的数据，并通过串口发送至网关，这也就实现了数据有节点汇聚到本地的过程。Arduino 开发板如图 3-1 所示。

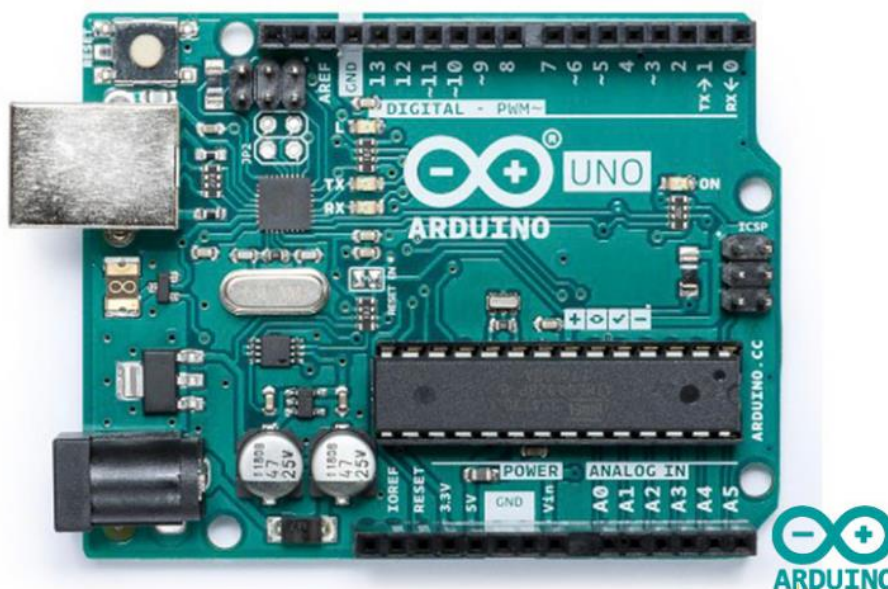


图 3-1 Arduino 开发板

本系统采用 PC 机作为网关节点，PC 机通过运行脚本程序实时接收有汇聚节点通过串口发送过来的数据，并进行帧校验和数据解包，若数据包格式符合事先约定的数据格式则接收，否则则丢弃数据包。网关的另一个作用是将符合要求的数据包进行解包分析，通过分析数据包的不同字段，恢复出节点的相关信息，并将信息进行分类再次打包，此次打包的格式为 json 格式，之后网关会将打包好的 json 数据包通过 post 请求递交给云服务器，再由云服务器端实时运行的 php 接收脚本将数据写入云数据库，这也就实现了数据从本地递交到云端的过程。

本系统采用新浪云服务器和新浪云共享数据库作为本系统的云端数据处理平台，云服务器在本系统中的作用主要体现在两个方面，第一，云服务器上运行的数据接收脚本

会实时接收由网关^[1]发送的 post 请求，并将 json 数据进行解包，在通过 sql 脚本写入新浪云共享数据库，实现本地数据的云端存储。第二，网站的运行离不开服务器的支持，本系统的可视化界面采用 Web 的形式，服务器前端运行的网页会实时向服务器后端发起 post 请求，后端的 php 则会响应前端的 post 请求，从数据库读取需要的数据并再次封装为 json 格式，以表单的格式提交给前端网页，网页端再次对 json 进行数据拆解，将需要实时展示的数据以曲线或者表格的形式显示在网页上，这也就实现了数据有数据库非可视化到网页端数据可视化的过程。新浪云服务器界面如图 3-2 所示。



图 3-2 新浪云服务器

本次毕业设计的软件平台主要有基于 C++ 的 Arduino IDE，基于 python 的 PyCharm 以及 Web 前后端开发平台。Python 开发环境如图 3-3。

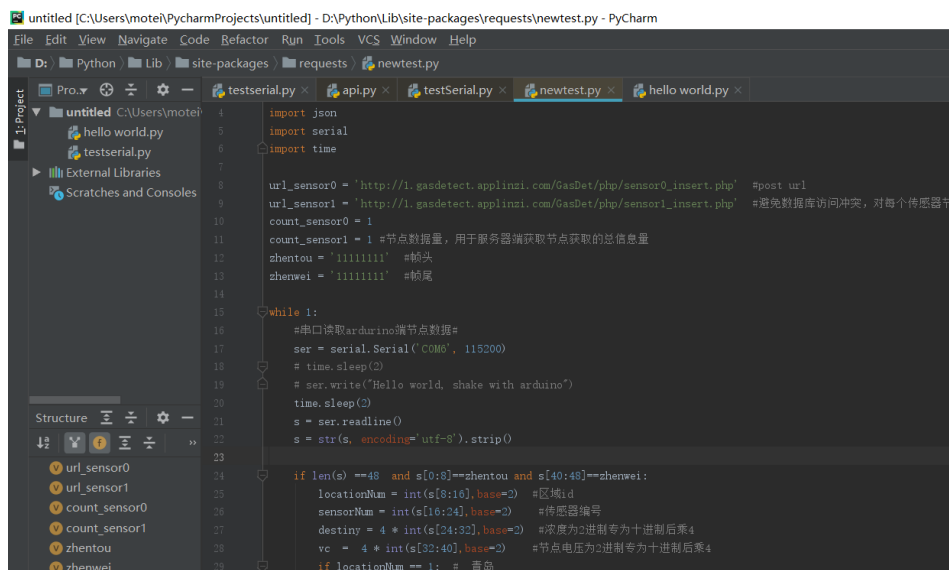


图 3-3 Python 开发环境

Web 开发环境主要是由前端的 html, javascript, css 以及后端的 php 脚本构成, 在本地主要借助 Apache 和 Hbuildx 进行开发测试, 云端则通过新浪云提供的云代码编辑器进行调试。HbuilderX 网站界面编写如图 3-4。



图 3-4 HbuilderX 网站界面编写

3.1.2 系统实现流程

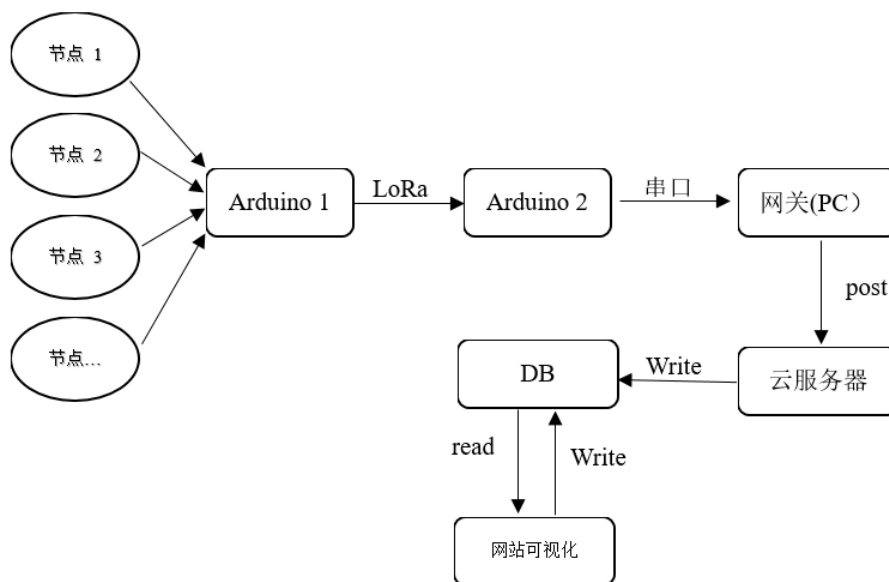


图 3-5 系统运转流程图

如 3-5 流程图所示, 网关为整个系统的中心环节, 不仅仅要对节点上报的数据进行接收, 分解并重新打包, 更要将重新封装好的数据重新通过 post 请求的方式递交给云服

务器，本系统中网关对整个系统的运行起到了承上启下的作用。通过上图，可以分析出本系统是极易扩展的，一般情况下只需要在传感层增加传感器节点，并将新增节点进行编号便可以扩展本系统。LoRa 通信可以通过信道频率进行区别，因此汇聚节点处的单片机可以通过设定不同的接收频率而分时接收来自不同节点的数据。本系统所采用的新浪云共享数据库作用主要体现在两个方面，一方面数据库需要接受网关所上传的传感器采集的数据，另一方面数据库需要响应网页实时的动态请求，读出所需的数据并实时动态的显示在网页上。

3.1.3 创新点与难点分析

本系统的创新点主要体现在三个方面，第一方面，由于本系统将数据封装成了规范的数据格式，类似计算机网络中 IP 的分配，对于数据包的每个字段都可以根据实际需要进行扩展和分解。第二个方面，本系统的网页采用 Ajax 异步加载的形式，可以根据用户需要更新界面的局部，而无需像传统的网页进行全局页面更新，不光节省了流量，更优化了系统的性能。第三方面，本系统创新性的加入了短信提醒功能，对于已经登陆的合法用户，每当有节点出现数据异常，服务器便会调用第三方接口通过短信的方式通知用户异常节点的编号，这一点在实际应用中具有重大意义，可以有效地避免传统检测系统通知不及时，有效降低事故发生风险。

本毕业设计的难点在于网关对于数据接收和数据发送的时序分配，由于本系统所租用的新浪云服务器内存较小，带宽较低，因此在进行实时数据传输时容易造成数据阻塞，甚至导致系统崩溃，这可以采用 python 的多线程或者通过安排合理的时序分配加以解决。第二个难点在于 ajax 异步加载图标，由于页面需要同时加载两者曲线图，这一过程是顺序进行的，而两张曲线图在读取进行读数据库请求时可能会发生阻塞而导致曲线加载出现问题，对此，我采取了两个解决方法，第一在两个任务中间加一定的延时，第二，当表单请求一直处于 pending 状态则判断发生了阻塞，脚本执行函数刷新功能，终止这次读取，防止系统一直处于数据阻塞状态。

3.2 详细设计

3.2.1 节点信息采集及封装

数据的采集是整个系统最基本的部分，数据采集的稳定性和高效性决定着整个系统的可靠性，本系统采用了 MQ-9 高灵敏度气敏传感器，可以及时的对空气中的危险可燃气体做出反应，本系统所采用的 Arduino Uno 开发板具有较高的系统稳定性，一般不会

发生死机复位现象，除了上述两点，本系统还加入了节点电量检测模块，实时汇报节点的电压值，确保数据的可靠性和系统的稳定高效运转。

数据传输离不开协议的制定，本系统针对应用场景的需要制订了一套专属的数据格式，整个数据帧采用二进制字符串的形式，由 48 位二进制字符串构成，从起始段到终端依次为帧头编号，区域标号，传感器编号，气敏传感器的数值，节点电压值以及帧尾，每个字段各占八位字符。

数据帧的具体格式如下：

11111111 00000001 00000001 00000000 00000000 11111111

帧头 区域 编号 数值 电压 帧尾

其中帧头，帧尾格式较为简单，都由八个 1 构成，用于进行初步的帧校验。

区域字段代表传感器节点所处的区域编号，此编号可以根据用户需要实现制定区域表，可以代表不同的城市或者同一城市的不同区域，因为总共为八位二进制编码，因此最多可以支持 255 个区域。

传感器编号字段对应为同一区域的不同传感器，具体到每个传感器节点，者这在实际生产生活中十分关键，其中后续的短信通知功能也是针对于每个传感器节点，因此，用户可以在第一时间获知发生异常的具体节点信息，从而有效避免事故的发生。由于本字段为八位二进制字符，因此每个区域最多安放 255 个节点。

气敏传感器的数值字段为本系统所需要的最关键的内容，它代表了某个节点处空气中有害气体的具体浓度。在实际的应用测试中发现，低浓度下气敏传感器所产生的电压数值大约在 0.2V-0.4V，高浓度下气敏传感器所产生的电压数值大约在 3V-3.5V，对于本系统所使用的 Arduino Uno 而言，模拟口 0-1023 的数值对应于实际电压 0V-5V，因此低浓度下单片机读取的模拟口数值大约在 40-80 之间，而高浓度下单片机读取的模拟口数值甚至会超过 600，这远远超出了八位二进制所能表示的范围，对此，本系统采取了将数据封装前进行除 4 操作，将处理后的数据进行封装，打包上传，这不仅节省了数据字段的位数，更在几乎不影响数据精度的前提下有效的封装了数据。

节点电压值字段可以有效反映出节点供电电压的有效值，节点的电压采集模块原理已在第二章详细概述，模块采用两个 10 千欧的分压电阻对电池进行分压，将其中一般的电压值通过滤波电容接入开发板模拟口，因为本系统传感器节点采用 9V 电池供电，因此节点电池范围的正常值大约在 4.5V 左右，对应于单片机 0-1024 的数值范围大约在 1000 左右，但实际测试中由于单片机内部自身的稳压处理以及电容滤波的影响导致实

际电压数值的正常范围大约在 600-700 左右，这也远远超出了八位二进制所能表示的数值范围，因此与气敏传感器数值字段的处理方式相同，采取数据封装前先除 4 的操作，再将数据进行打包上传。

3.2.2 LoRa 通信流程

本系统中 LoRa 通信主要为有传感器节点到汇聚节点之间的数据传输，采用 SX1278 模块，SX1278LoRa 模块采用 SPI 总线的方式与 Arduino 进行数据传输，这不同于普通的串口 LoRa 模块，这也就决定了 SX1278 必须借助支持 SPI 的单片机与网关进行数据传输，而不可以直接通过串口转 TTL 模块与网关串口直接通信。Arduino Uno 与 SX1278 模块的硬件接线图如下，SX1278 模块总共有 16 个引脚，其中用到的只有八个，需要特别注意的是 SPI 引脚的连接。单片机连线如图 3-6。

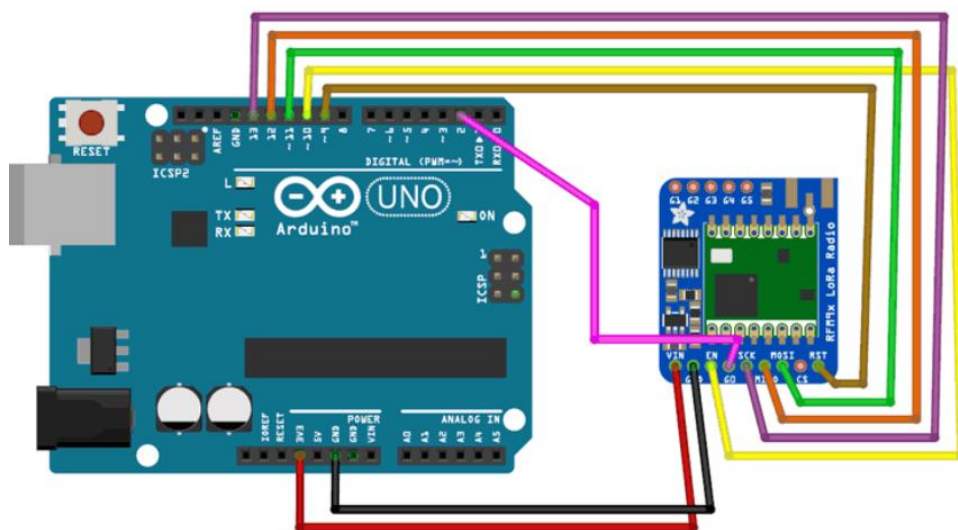


图 3-6 Arduino 与 SX1278 接线图

本系统 LoRa 发送端程序采取轮询的方式持续发送，而接收端处于低功耗的考虑采用了中断方式，SX1278 模块的 DIO0 引脚可以触发外部中断，每当模块检测到有数据包传来便会触发外部中断，将数据打入缓存并等待单片机进行接收，这与轮询方式相比既降低了功耗又不影响程序其他功能的执行，但在实际应用中发现经常会出现接收的数据包中字符串的个数正确，而字符串之间的顺序出现了颠倒，对于这个问题，采取在中断服务程序中加一小段延时加以解决。

LoRa 通信的具体程序实现借鉴了 LoRa 的官方例程，在 Arduino Uno 中可以将其库文件加入工程，直接调用其中的方法。但在调用其中初始化函数时出现了初始化失败的问题，查询了其库函数中的初始化程序发现，其初始化操作只有一次寄存器写入操作，

写入失败模块则会进入 While(1)死循环,对此,我将初始化寄存器写入的次数限定了 500 次的上限,解决了初始化失败的问题。Arduino 发送端部分函数如图 3-7 所示。

```

    LoRa.endPacket();
    delay(1000); //延时1000毫秒
    LoRa.beginPacket();
    delay(1000);
    LoRa.print(Destiny1_send); //串口输出temp的数据
    delay(1000); //延时500毫秒

    LoRa.endPacket();
    /*
    */
    /*
    static int counter;
    counter++;
    counter %=1000; //
    //uint8_t payload[4] = {0b000000001, 0b000000001, 0b63, 0b11111111};
    //单纯的发送只需要一下3步即可
    LoRa.beginPacket();
    delay(1000);
    //这里的write用来发送数组的
    //LoRa.write(payload, sizeof(payload));
    Serial.println(counter);
    // ... 这里省略了部分代码 ...

```

图 3-7 Arduino 发送端部分函数

由于库函数的支持, LoRa 通信简单了很多, 尤其是不需要手动配置寄存器, LoRa 模块在上电后未激活的状态一般会处于睡眠模式, 单片机通过调用 LoRa.begin()函数将模块由睡眠模式激活为待机模式, 若此时有发送任务, 单片机会调用 LoRa.beginpackage()函数将 LoRa 模块由待机模式转为发送模式, 执行数据发送的任务, 待任务完成之后, 执行 LoRa.endpackage()函数将 LoRa 模块由发送模式重新转为待机模式, 等待下一次发送任务。对于接收端, 与接收端类似, 先通过执行 LoRa.begin()函数进行模块的初始化, 由睡眠模式转入待机模式, 由于接收端采取中断触发方式, Lora 接受完成时会在 DIO0 引脚产生上升沿, 再通过单片机引脚去捕获这个上升沿, 调用中断服务程序 Lora.onReceive()执行相应的任务。本系统中中断服务程序主要有两个任务, 一个是读取 LoRa 缓存中的数据, 另一个任务是将读取的数据进行打包通过串口的方式发送到网关。

3.2.3 网关数据解包及上传

本系统采用 PC 机作为网关, 网管通过串口的方式从汇聚节点的单片机获取数据, 经过校验, 解包, 封装等一系列操作之后再通过 Post 请求的方式递交给服务器。本系统既定的数据包格式为 48 位二进制字符串, 通过 3.2.1 章节可以了解到, 数据帧的前八位为帧头, 最后八位为帧尾用于格式判断, 中间的 32 位为数据包的具体内容。

本系统的网关工作方式采用轮询的方式, 网关程序基于 python3.7 进行编写, 引入了 python 中 serial, requests, json 等函数库, 网管程序实现的大致流程可以概括为如下几个方面, 首先, 网关通过 serail.Serial()函数进行串口初始化, 通过 serail.readline()函数进行串口数据的读取, 并将数据转换为 utf-8 的格式; 第二步, 网关需要对从串口读到数据进行格式判断, 首先校验帧头及帧尾, 符合的数据包则接受, 不符合的数据包则丢

弃，对于符合要求的数据包，网关通过 `int(string[8:16],base=2)` 函数将各个字段的二进制字符串转为十进制 `int` 型变量，封装为 `json` 的格式。最后，网关需要将封装好的 `json` 包通过 `post` 请求的方式递交给新浪云服务器，由服务器端运行的 `php` 脚本接收数据并进行处理，此过程需要调用 `requests.post()` 函数，此函数的两个参数为服务器端 `php` 脚本的 `url` 以及网关封装好的 `json` 包。`json` 包格式如图。

```
elif locationNum == 2:
    Time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S') # 获取节点读取数据的时间，格式为年月日时分秒
    count_sensor1 = str(count_sensor1)
    data_sensor1 = {'Num': count_sensor1,
                   'SensorNum': sensorNum,
                   'Location': 'Weihai',
                   'Destiny': str(destiny),
                   'Time': Time,
                   'Voltage': str(v0)}

    count_sensor1 = int(count_sensor1)
    count_sensor1 += 1
    r1 = requests.post(url_sensor1, data=data_sensor1) # sensor0进行post请求
    time.sleep(1)
    # print('sensor0:' + str(count_sensor0) + 'r0' + str(r0))
    ser.close()
```

图 3-8 Json 字段内容

在每两次 `post` 请求之间都要加一定时间的延时函数，否则会造成服务器压力过大，导致系统瘫痪，除此之外，在读取串口的任务完成以后一定要通过 `ser.close()` 函数关闭串口，防止网关程序一直占用串口资源，影响其他程序对串口资源的调用。

3.2.4 云数据库数据处理

本系统的数据库采用新浪云共享数据库，使用 `phpAdmin` 进行数据库管理。本系统数据库主要有三张表，一张用于存放用户注册信息，另外两张用于存放从网关获取的节点信息。

`LogReg` 表用于存放用户注册信息，主要有 `username`, `password`, `email`, `regdate`, `phone` 几个属性，其中用户名为主码。

显示:	30	行, 开始行数:	0			
以	水平	▼	模式显示, 并且在	100	行后重复标题	
主键排序:	无	▼				
+ 选项						
←→	username	password	email	regdate	phone	
<input type="checkbox"/>	c994283977	f9a7395bf05fd9f1	994283977@qq.com	1556958870	17806286303	
<input type="checkbox"/>	liujingyuan3	f9a7395bf05fd9f1	994283977@qq.com	1556956903	17806286303	
<input type="checkbox"/>	liujingyuan4	f9a7395bf05fd9f1	994283977@qq.com	1557731871	17806286303	
<input type="checkbox"/>	liuyonghua	f9a7395bf05fd9f1	994283977@qq.com	1556881629	13455874369	
<input type="checkbox"/>	moteily	f9a7395bf05fd9f1	994283977@qq.com	1556881503	17806286303	
<input type="checkbox"/>	sunlin521	cf519b3271c3df56	994283977@qq.com	1557660751	17806286303	
<input type="checkbox"/> 全选 / <input type="checkbox"/> 全不选 选中项: <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>						

图 3-9 LogReg 表中字段

Sensor_Det 表用于存储从网关读取的传感器节点信息, 包括 Number, SensorNum, Location, Destiny, Time, DangerRelease, Voltage 几个属性, 其中 Number 为主码, Voltage 为节点电压值。Sensor_Det 表如图 3-10 所示。

←T→	Num	SensorNum	Location	Destiny	Time	DangerRelease	Voltage
<input type="checkbox"/>  	1	1	Qingdao	12	2019-05-13 15:16:59	1	452
<input type="checkbox"/>  	2	1	Qingdao	8	2019-05-13 15:17:07	1	456
<input type="checkbox"/>  	3	1	Qingdao	8	2019-05-13 15:17:48	1	456
<input type="checkbox"/>  	4	1	Qingdao	12	2019-05-13 15:17:57	1	452
<input type="checkbox"/>  	5	1	Qingdao	8	2019-05-13 15:18:05	1	456
<input type="checkbox"/>  	6	1	Qingdao	8	2019-05-13 15:18:13	1	456
<input type="checkbox"/>  	7	1	Qingdao	8	2019-05-13 15:18:21	1	452
<input type="checkbox"/>  	8	1	Qingdao	8	2019-05-13 15:18:29	1	456
<input type="checkbox"/>  	9	1	Qingdao	8	2019-05-13 15:18:38	1	456
<input type="checkbox"/>  	10	1	Qingdao	12	2019-05-13 15:18:46	1	452
<input type="checkbox"/>  	11	1	Qingdao	8	2019-05-13 15:18:54	1	456

图 3-10 Sensor_De 表中字段

云数据库数据的写入可概述为以下流程, 首先, 网关通过 post 的方式向服务器对应 php 脚本的 url 发送数据请求, 服务器端实时运行的 php 脚本通过\$_POST()函数获取 post 请求的字段名称, 若字段名称与 php 要接收字段名称相符合则进行接收, 并通过 sql 语句写入数据库。在进行数据库操作时需要注意, 每当数据写入或者读出完成时都要用 con->close()函数断开数据库连接, 否则会导致其他应用访问同一张表时出现数据库占用的情况。

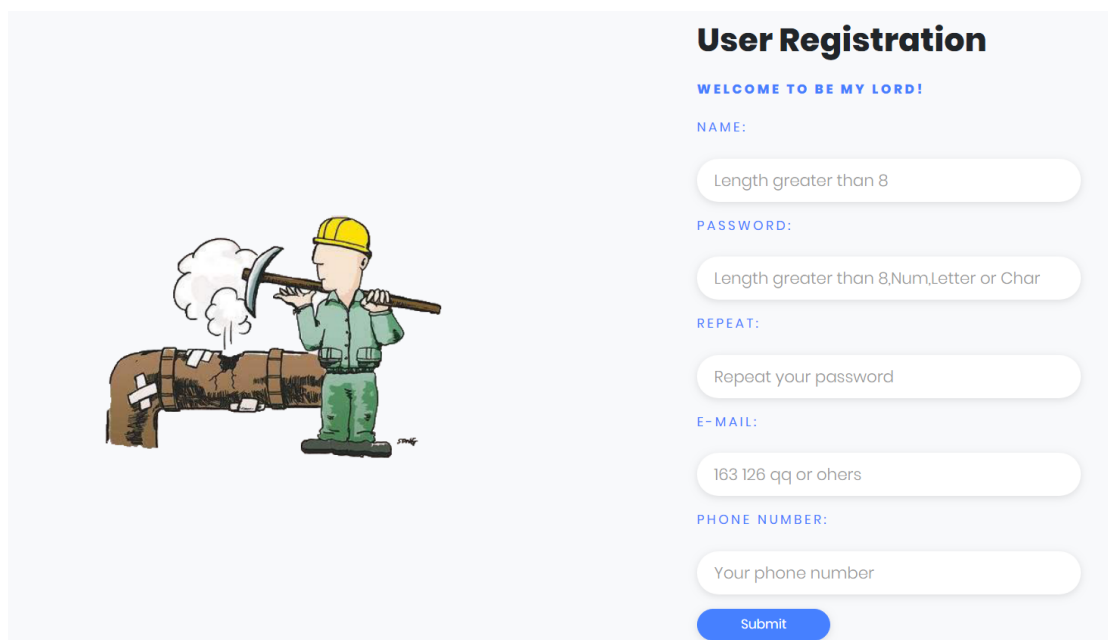
3.2.5 注册登录界面的实现

对于任何一个完整的系统, 尤其是对于涉及工业安全的系统一定要进行用户权限的设定, 只有管理员 (即注册用户) 才有权对系统进行访问或者进行相关操作。

本系统的注册及登陆功能的实现采用 post 及 session 的方式^[2], 并将用户的注册信息保存在云数据库中, 用户通过在注册界面的表单元素中填写用户名及密码, 此时会触发 InputCheck()函数, 此函数会验证用户的注册信息(用户名是否为空, 必要信息时候填写等), 当用户单击 submit 按钮是会触发服务器在后端运行的 php 脚本, 用于判断用户名及密码是否符合规范, 用户名是否被注册等, 当所有信息都符合要求时会将信息写入数据库的 RegLog 表中, 供用户登陆及相关预警功能调用。

登录功能的实现与注册相似, 首先用户在网页端进行用户名及密码的填写, 此时会触发 html 界面上运行的 InputCheck()函数用于判断用户名是否为空, 密码是否为空, 当用户单击 Log 按钮, 会触发服务器后端运行的 log.php 脚本, 脚本程序获取前端界面递交的登陆信息, 基于用户名进行数据库信息匹配, 首先检测用户名是否存在, 不存在则

会提示用户为空，用户名存在的话会再次检查密码是否正确，密码正确则会进入系统界面，并将用户的\$_SESSION()对用户的相关登陆信息进行保存。注册界面如图 3-11,登陆界面如图 3-12。



The registration form is titled "User Registration" and includes a welcome message "WELCOME TO BE MY LORD!". It features a cartoon illustration of a miner on the left. The form contains input fields for Name, Password, Repeat Password, E-Mail, and Phone Number, each with a placeholder text indicating requirements. A "Submit" button is at the bottom.

User Registration

WELCOME TO BE MY LORD!

NAME:

Length greater than 8

PASSWORD:

Length greater than 8,Num,Letter or Char

REPEAT:

Repeat your password

E-MAIL:

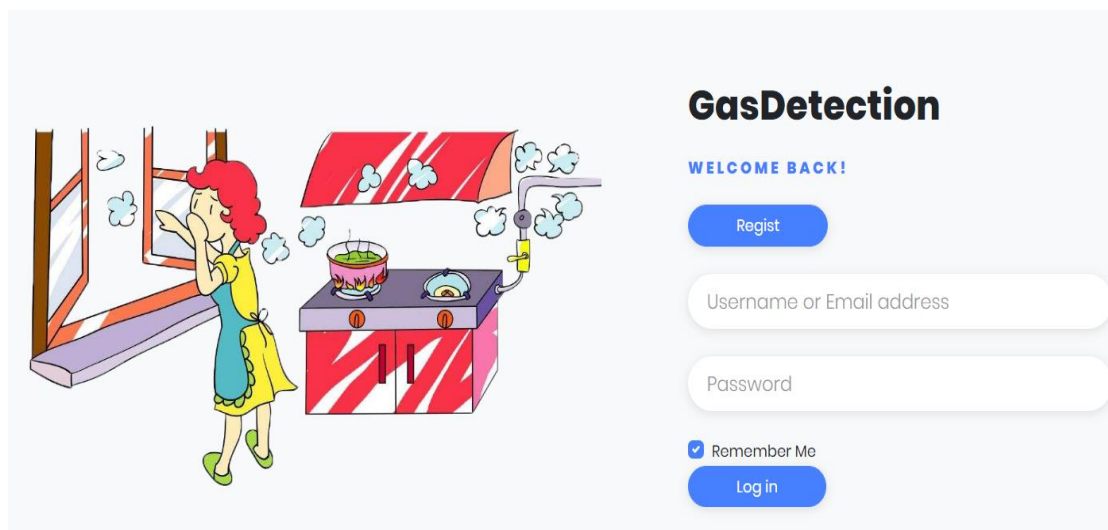
163 126 qq or others

PHONE NUMBER:

Your phone number

Submit

图 3-11 注册界面



The login form is titled "GasDetection" and includes a welcome message "WELCOME BACK!". It features a cartoon illustration of a woman in a kitchen on the left. The form contains input fields for Username or Email address and Password. There is a "Remember Me" checkbox and a "Log in" button.

GasDetection

WELCOME BACK!

Regist

Username or Email address

Password

☒ Remember Me

Log in

图 3-12 用户登陆界面

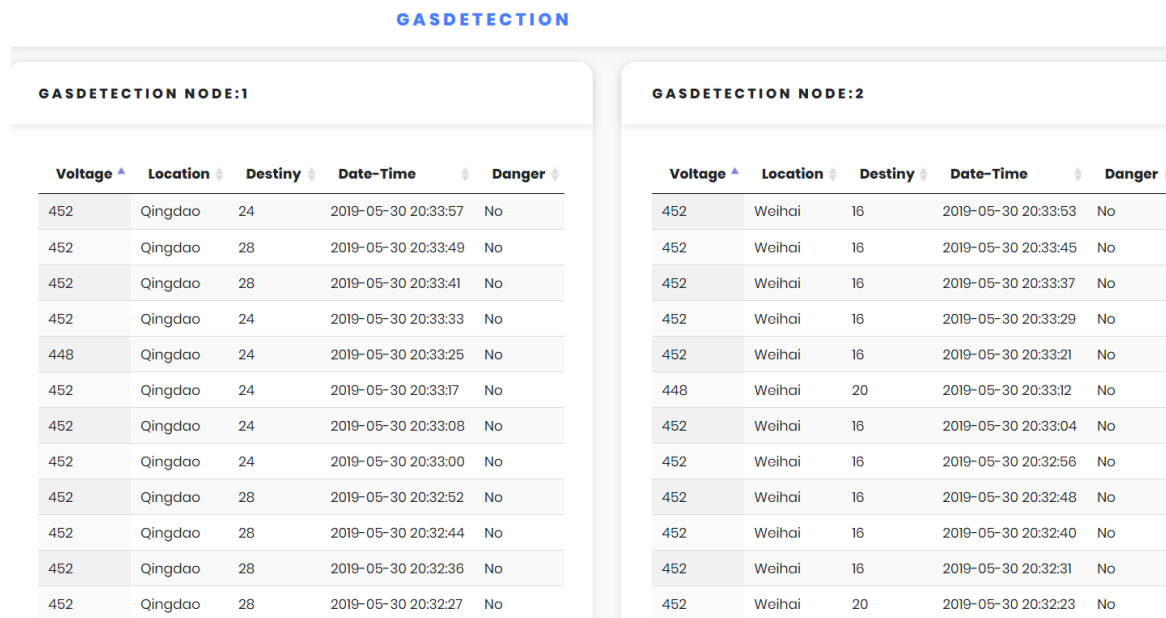
将用户的登陆信息通过 session 的形式存储下来是十分必要的，假如把账号密码作为用户登陆的钥匙，让你进入系统，但系统本身并不知道用户的具体信息，而 session 的角色相当于身份证，他可以让系统可以识别出登录用户的具体信息，以便于系统对当前用的进行相关的反馈。用户进行网站访问时，往往需要浏览许多界面，对于 php 作为后

端构筑的网站来说,用户进行网页访问时要出发许多 php 脚本,而 php 脚本中的常规变量都是具有作用域的,仅仅对此 php 有效,而 session 的作用就类似 C 语言中 static 类型的全局变量,同一个工程中的文件都可以对其进行调用和修改。session 数据存储不需要数据库的参与,而是存储与服务器端 php.ini 文件中。

对于登陆成功的用户,本系统中以 session 的形式对用户的用户名进行存储,当用户主要登陆时,服务器端会执行 unset()函数对 session 变量进行释放。

3.2.6 信息可视化的实现

信息的可视化对于一个数据监测系统是必不可少的,本系统采用网页端曲线加表格的形式对传感器端的数据进行实时展示。由于本系统的网页界面采用了 jQuery 框架,因此在在进行曲线绘制时调用了 jQuery 的 Morris 曲线库,实现表格显示时引用了第三方库函数 datatables。网站数据表格如图 3-13。



GASDETECTION NODE:1					GASDETECTION NODE:2				
Voltage	Location	Destiny	Date-Time	Danger	Voltage	Location	Destiny	Date-Time	Danger
452	Qingdao	24	2019-05-30 20:33:57	No	452	Weihai	16	2019-05-30 20:33:53	No
452	Qingdao	28	2019-05-30 20:33:49	No	452	Weihai	16	2019-05-30 20:33:45	No
452	Qingdao	28	2019-05-30 20:33:41	No	452	Weihai	16	2019-05-30 20:33:37	No
452	Qingdao	24	2019-05-30 20:33:33	No	452	Weihai	16	2019-05-30 20:33:29	No
448	Qingdao	24	2019-05-30 20:33:25	No	452	Weihai	16	2019-05-30 20:33:21	No
452	Qingdao	24	2019-05-30 20:33:17	No	448	Weihai	20	2019-05-30 20:33:12	No
452	Qingdao	24	2019-05-30 20:33:08	No	452	Weihai	16	2019-05-30 20:33:04	No
452	Qingdao	24	2019-05-30 20:33:00	No	452	Weihai	16	2019-05-30 20:32:56	No
452	Qingdao	28	2019-05-30 20:32:52	No	452	Weihai	16	2019-05-30 20:32:48	No
452	Qingdao	28	2019-05-30 20:32:44	No	452	Weihai	16	2019-05-30 20:32:40	No
452	Qingdao	28	2019-05-30 20:32:36	No	452	Weihai	16	2019-05-30 20:32:31	No
452	Qingdao	28	2019-05-30 20:32:27	No	452	Weihai	20	2019-05-30 20:32:23	No

图 3-13 数据的表格形式展示

实时性对于检测系统是十分重要的,因此,本系统中采用了 ajax 异步加载的形式,实时对曲线数据进行更新^[3]。数据曲线图的绘制可概括为以下几个步骤,首先网页端会像后端运行的 php 脚本发送 post 或者 get 请求,后端脚本对前端的请求进行回应,从数据库中读取前端需要的数据,并封装成 json 的形式,以网页表单的形式提交给前端曲线,前端曲线通过 json 中属性的关键字进行数据提取,并通过 ajax 异步加载以及 clearTimeout()函数刷新的方式保证数据的动态和实时性。曲线展示如图 3-14。

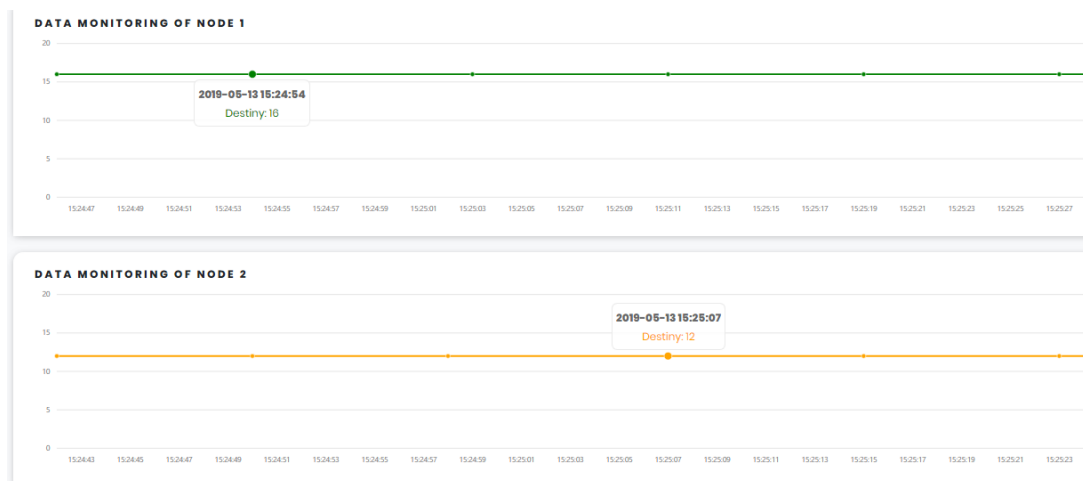


图 3-14 数据的曲线形式展示

表格的实现相比于曲线要复杂，主要体现在数据格式上，曲线绘制所需要的 json 数据只需要包含曲线中所需要的元素，例如时间，浓度等，而表格的绘制不光需要表格中需要体现的内容，更要在将表长，数据长等格式元素以 json 数组形式与传感器数据封装到一起。

500 ms 1000 ms 1500 ms 2000 ms 2500 ms 3000 ms 3500 ms 4						
Name	Status	Type	Initiator	Size	Time	
<input type="checkbox"/> sensor0_chart.php	200	xhr	jquery.min.js:2	514 B	119 ms	
<input type="checkbox"/> sensor1_chart.php	200	xhr	jquery.min.js:2	513 B	116 ms	
<input type="checkbox"/> sensor0_chart.php	200	xhr	jquery.min.js:2	514 B	116 ms	
<input type="checkbox"/> sensor1_chart.php	200	xhr	jquery.min.js:2	513 B	151 ms	
<input type="checkbox"/> sensor0_chart.php	200	xhr	jquery.min.js:2	514 B	117 ms	
<input type="checkbox"/> sensor1_chart.php	200	xhr	jquery.min.js:2	513 B	116 ms	

图 3-15 网页动态请求数据

在进行曲线和表格数据的记载时需要安排好时序，最好将各个曲线及表格对统一数据表的访问进行分时处理，否则可能会由于服务器性能有限导致数据请求一直出去 pending 状态导致信息加载失败。

3.2.7 短信预警功能的实现

本系统创新性的加入了短息预警通知的功能，本系统租用了第三方应用“短信宝”的服务器端接口^[4]，通过后端 php 脚本实施检测传感器节点^[5]传来的数据，只要出现异常数据便会触发短信报警，保证监测系统的实时性及安全性。

```

    "42" => "帐户已过期",
    "43" => "IP地址限制",
    "50" => "内容含有敏感词"
);
$smsapi = "http://www.smsbao.com/"; //短信网关
$user = "moteily"; //短信平台帐号
$pass = md5("460716687"); //短信平台密码
$content="【GasDet】 2号传感器节点附近有漏气现象, 请及时处理。"; //要发送的短信内容
$phone = $phonenumber;
$sendurl = $smsapi."sms?u=".$user."&p=".$pass."&m=".$phone."&c=".urlencode($content);
$flag = 1; //只发送一次短信;
$sql3 = "ALTER TABLE Sensor1_Det alter COLUMN DangerRelease SET DEFAULT 1"; //修改数据库DangerRelease变量
$result3 = $phpGasDet -> Query_Mysql($sql3, $conn) or die("查询失败");
/ALTER TABLE Sensor0_Det alter COLUMN DangerRelease SET DEFAULT 0 修改默认值为0;
file_get_contents($sendurl) ;

//$DangerRelease=0; // ALTER TABLE Sensor0_Det alter COLUMN DangerRelease SET DEFAULT 0 修改默认值为0

```

图 3-16 短信通知功能的实现

由于用户在注册时填入了手机号，且存入了 RegLog 数据表中，用户在登陆时，系统获取用户的用户名存为 session，并通过此 session 变量查询 RegLog 表获取用户的手机号，再调用第三方接口实现短信发送的功能。一旦出发短信预警，节点会一直处于 danger 状态，短信通知业务会被取消，需要管理员登入系统，通过点击系统主页“Danger Release”按钮进行危险解除，恢复节点的短信通知业务^[6]。

第4章 系统测试

4.1 软件及硬件测试

4.1.1 传感器节点及 LoRa 发送模块测试

传感器节点在 9V 电源供电下，传感器模块，电压测量模块及 LoRa 发送模块工作正常，从 Arduino IDE 的串口解释器中可以查看封装好的 LoRa 数据。LoRa 发送的字符串如图 4-1 所示。



图 4-1 发送端发送的字符串

4.1.2 LoRa 接收模块测试

LoRa 接收节点由 9V 电源供电，可几乎没有延迟的情况下接收 LoRa 发送端传来的字符串数据。单片机接收到的字符串如图 4-2 所示。

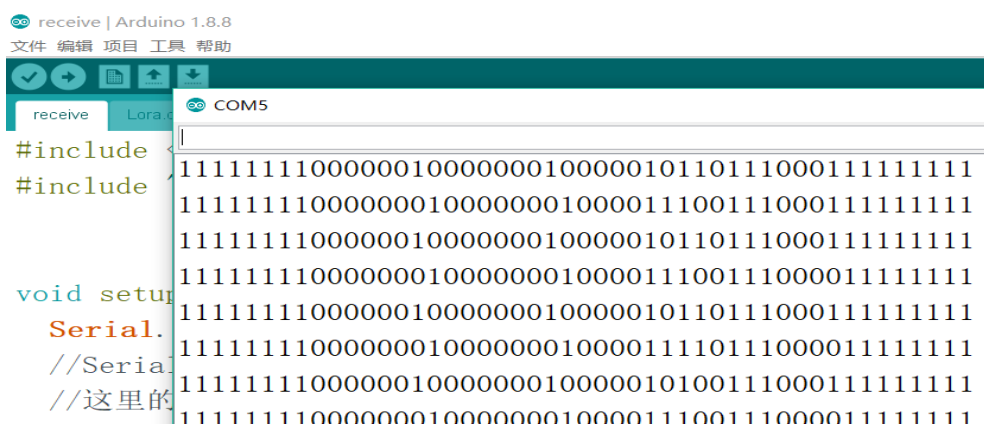


图 4-2 接收端接受的字符串

4.1.3 网关数据上传测试

PC 作为网关网关串口实时读取 LoRa 接收节点通过串口发送来的数据，并进行数据包解析，讲解析好的数据封装为 json，最后通过 Post 请求的方式上传云平台。

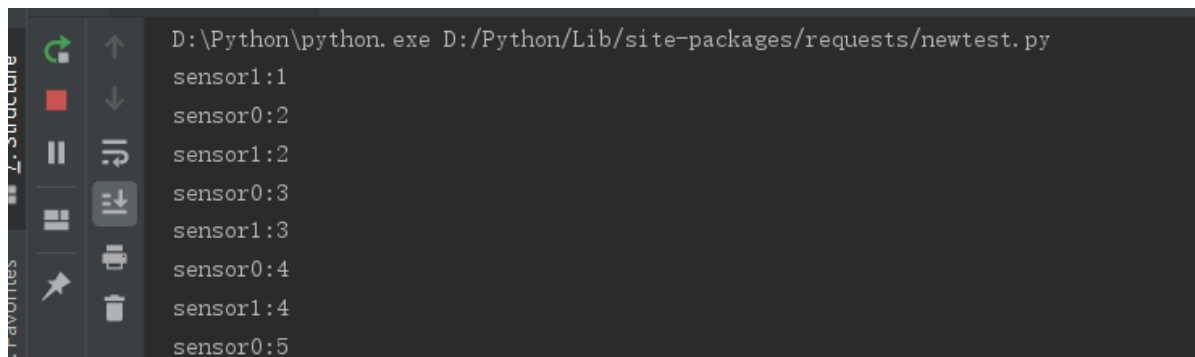


图 4-3 节点数据上传成功

4.1.4 数据库数据写入测试

数据库 Senser_Det 表中实时写入来自网关的 json 数据，下图通过 phpAdmin 查看数据库中数据写入情况。数据库写入如图 4-4。

			Num	SensorNum	Location	Destiny	Time	DangerRelease	Voltage
<input type="checkbox"/>		<input checked="" type="checkbox"/>	1	1	Qingdao	12	2019-05-13 15:16:59	1	452
<input type="checkbox"/>		<input checked="" type="checkbox"/>	2	1	Qingdao	8	2019-05-13 15:17:07	1	456
<input type="checkbox"/>		<input checked="" type="checkbox"/>	3	1	Qingdao	8	2019-05-13 15:17:48	1	456
<input type="checkbox"/>		<input checked="" type="checkbox"/>	4	1	Qingdao	12	2019-05-13 15:17:57	1	452
<input type="checkbox"/>		<input checked="" type="checkbox"/>	5	1	Qingdao	8	2019-05-13 15:18:05	1	456
<input type="checkbox"/>		<input checked="" type="checkbox"/>	6	1	Qingdao	8	2019-05-13 15:18:13	1	456
<input type="checkbox"/>		<input checked="" type="checkbox"/>	7	1	Qingdao	8	2019-05-13 15:18:21	1	452
<input type="checkbox"/>		<input checked="" type="checkbox"/>	8	1	Qingdao	8	2019-05-13 15:18:29	1	456
<input type="checkbox"/>		<input checked="" type="checkbox"/>	9	1	Qingdao	8	2019-05-13 15:18:38	1	456
<input type="checkbox"/>		<input checked="" type="checkbox"/>	10	1	Qingdao	12	2019-05-13 15:18:46	1	452
<input type="checkbox"/>		<input checked="" type="checkbox"/>	11	1	Qingdao	8	2019-05-13 15:18:54	1	456
<input type="checkbox"/>		<input checked="" type="checkbox"/>	12	1	Qingdao	372	2019-05-13 15:19:02	1	456
<input type="checkbox"/>		<input checked="" type="checkbox"/>	13	1	Qingdao	84	2019-05-13 15:19:10	1	456

图 4-4 数据库数据写入

<input type="checkbox"/>		<input checked="" type="checkbox"/>	c994283977	f9a7395bf05fd9f1	994283977@qq.com	1556958870	17806286303	
<input type="checkbox"/>		<input checked="" type="checkbox"/>	liujingyuan3	f9a7395bf05fd9f1	994283977@qq.com	1556956903	17806286303	
<input type="checkbox"/>		<input checked="" type="checkbox"/>	liujingyuan4	f9a7395bf05fd9f1	994283977@qq.com	1557731871	17806286303	
<input type="checkbox"/>		<input checked="" type="checkbox"/>	liuyonghua	f9a7395bf05fd9f1	994283977@qq.com	1556881629	13455874369	
<input type="checkbox"/>		<input checked="" type="checkbox"/>	moteily	f9a7395bf05fd9f1	994283977@qq.com	1556881503	17806286303	
<input type="checkbox"/>		<input checked="" type="checkbox"/>	sunlin521	cf519b3271c3df56	994283977@qq.com	1557660751	17806286303	

图 4-5 数据库用户信息

4.1.5 数据可视化测试

系统的可视化界面中，实时更新传感器节点的最新数据，并进行曲线绘制。曲线形式的展示如图 4-6。

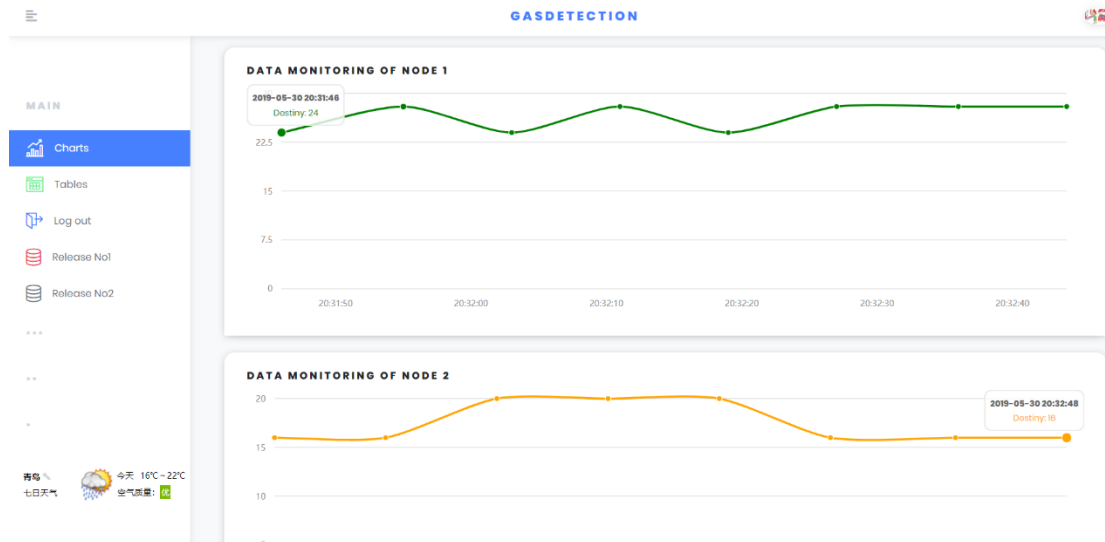


图 4-6 数据的曲线图形式

系统的可视化界面中，实时更新的数据通过表格写入的形式，以数据表的形式实时显示在网页上。表格形式的展示如图 4-7 所示。

The figure displays the same 'GASDETECTION' web interface, but with the 'Tables' option selected in the sidebar. The main content area shows two data tables. The left table, 'GASDETECTION NODE:1', contains 12 rows of data for 'Qingdao' with columns: Voltage (452), Location (Qingdao), Destiny (24 or 28), Date-Time, and Danger (No). The right table, 'GASDETECTION NODE:2', contains 12 rows of data for 'Weihai' with columns: Voltage (452), Location (Weihai), Destiny (16 or 20), Date-Time, and Danger (No). The sidebar and weather widget remain the same.

图 4-7 数据的表格形式

4.1.6 短信功能测试

当节点浓度达到 200，则会触发短信通知功能，用户需自己登陆系统取消节点预警状态^[7]，恢复短信通知业务。短信通知业务如图 4-8 所示。



图 4-8 短信通知业务



图 4-9 恢复短信通知业务

4.2 测试总结

经过一系列测试表明，本系统传感器较为灵敏，LoRa 通信比较稳定，在 1000 次数据包的传输过程中不存在丢包现象^[8]，网关数据上传几乎没有延时，数据库数据写入准确，短信通知业务计时，网页数据的可视化实时性较高。硬件连线如图 4-10 所示。

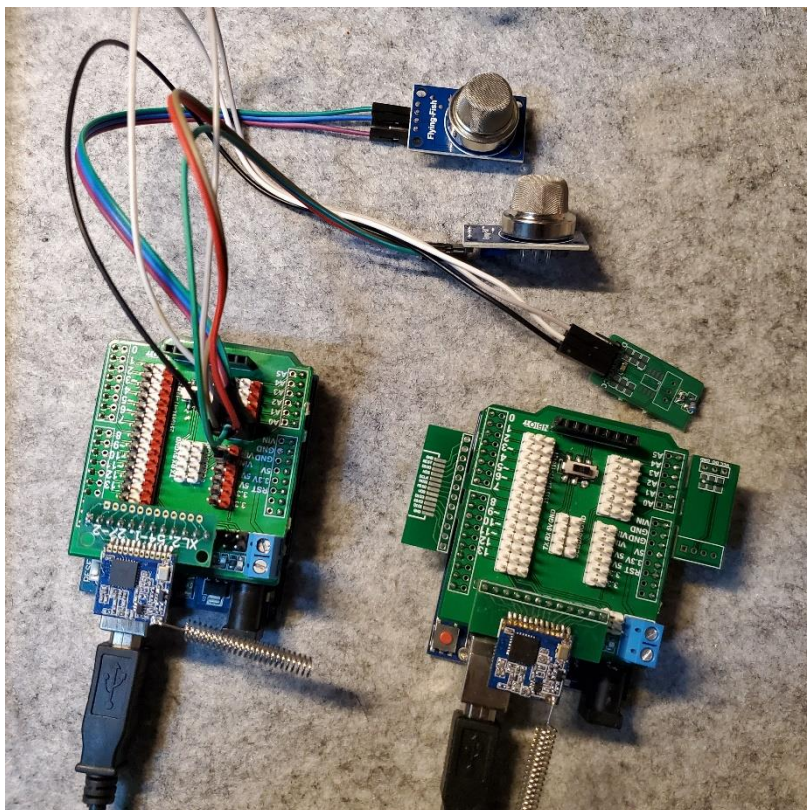


图 4-10 系统硬件模块

网站主题界面如图 4-11 所示，包括注册登陆及可视化展示。

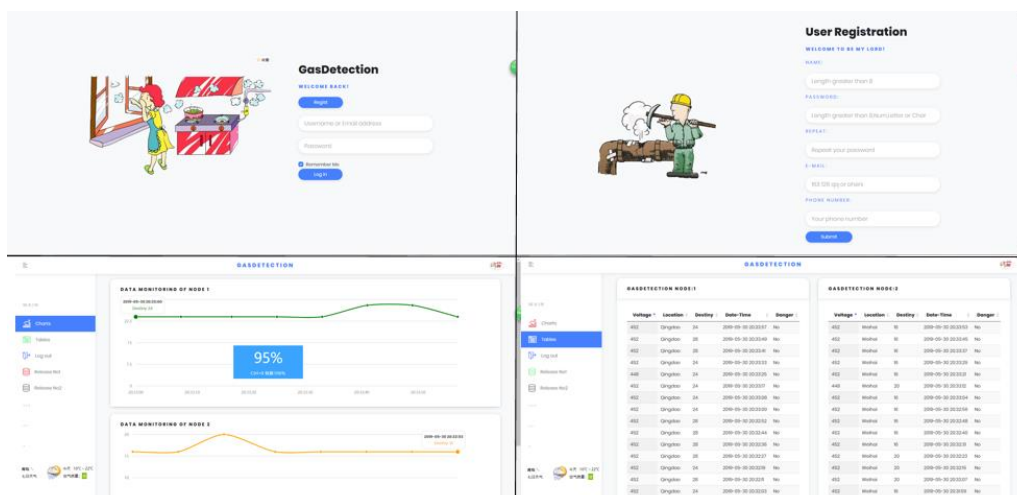


图 4-11 网站主体界面

第 5 章 结论

本系统验证了基于 LoRa 的暗渠有害气体监测系统的可行性，与传统的人工检测或者半自动化检测相比，LoRa，Arduino，云服务器等物联网技术的应用使得本系统更加高效，可靠，并且人力成本更低，具有广阔的应用前景。

本系统的应用具有重要的现实意义，目前由于暗渠气体爆炸造成的危险事故屡见不鲜，造成的财产损失数以亿计，这些都是亟待解决的问题，基于 LoRa 的暗渠有害气体检测系统可以实时监测暗渠中有害气体的浓度状态并第一时间对出现浓度异常的泄漏点进行报警通知，极大程度的减少此类事故造成的损失，具有重大的现实意义。

目前市面上也存在类似的暗渠气体检测装置，但是都是基于 WiFi，GPRS 等通信技术，LoRa 与这些通信技术相比，在功耗方面有着较大的优势^[9]，除此之外，LoRa 系统支持自组网不需要运营商支持，运营成本上也有较大优势。

当然，本系统中作为一款尚未面世的产品，仍存在众多不尽如人意，亟待改善的地方。例如，单片机采用 Arduino 功耗上仍是问题，可以采用功耗更低的 MSP430 系列；网页可视化界面中不支持地图显示节点位置，可以引入 gps 模块及高德地图等第三方接口加以解决；目前的短信通知量较少，因此租用第三方短信接口完全可以满足，但随着后期业务增大，可以考虑搭建自己的短信服务器，尤其是涉及危险气体浓度等敏感信息，防止第三方接口窃取短信内容也是十分必要的；最重要的一个问题，LoRa 通信在暗渠中信号受到干扰，造成丢包率升高，可以设计更高倍数的运算放大器，采用硬件的方式加以解决^[10]。总而言之，本系统有着广大应用前景和现实意义，但是仍有着众多的不足，需要开发者不断完善。

致 谢

文章写到最后，大学生也基本到尾声了，大学毕业的时间和高考的时间前前后后差不了几天，回想昨日，仿佛自己仍是那个即将踏入高考考场的少年，奈何光阴似箭，岁月如梭，四年时间弹指一挥，令人唏嘘。

四年里，离开父母，远离家乡，渐渐的习惯了也喜欢上了黄岛这片土地，习惯了晚饭后去唐岛湾吹吹海风的悠闲自在，习惯了考试前几天图书馆闭关修炼的撕心裂肺，习惯了考完试和舍友北门撸串的畅快淋漓.....有些东西只有失去的时候才学会珍惜，这些日子里，手机照片里多了许多黄岛记忆，石大记忆，金沙滩的日出，大珠山的日落，食堂两块钱一只的油闷大虾以”脏乱”的 222 宿舍，这些都会永远铭记在心里，无法割舍。

大学四年，对我的改变巨大，学到了专业知识，有了更强了自我学习能力以及独立思考解决问题的能力，这都离不开四年里各位老师及同学的帮助，尤其是物联网工程系的老师们，老师们不仅教会了我专业知识，更在潜移默化中教会了我做人，做事的技巧，老师们平日里真真正正走进了学生的生活，回想起来，这种亦师亦友的师生关系真的令人羡慕。

感谢是四年里朝夕相处的同学，一起学习，一起娱乐，在你们身上我看到了许许多多自己的影子，大学四年里我们共同进步，祝大家前程似锦。

最后，感谢我的母校中国石油大学（华东），感谢母校为我提供了舒适的学习及生活环境，优质的教育资源，优秀的发展平台，无论将来在学习还是工作中，我一定会谨记石大“惟真惟实”的作风，定不会令母校失望。

参考文献

- [1] [美]布兰登·罗德（Brandon Rhodes）.Python 网络编程.人民邮电出版社,2016.
- [2] [美]克罗克洛德.Javascript 语言精粹.电子工业出版社,2012.
- [3] 林凌.物联网与穿戴式应用中的传感器及其接口技术.电子工业出版社,2016.
- [4] 童诗白,华成英.模拟电子技术基础.高等教育出版社,2010.
- [5] 阎石编.数字电子技术基础.高等教育出版社,2010.
- [6] 明日科技.PHP 项目开发全程实录.清华大学出版社,2018.
- [7] 崔广伟.可燃和有毒气体检测报警器.中国石化出版有限公司,2017.
- [8] 彭澎,姜旭.可视化 H5 页面设计与制作.人民邮电出版社,2019.
- [9] 郁有文.传感器原理及工业应用.西安电子科技大学出版社,2019.
- [10] 徐琴.数据处理与知识发现.机械工业出版社,2019.

附 录

附录 A 传感器节点数据封装及发送程序

```
#include "Arduino.h"
#include<string.h>
#include<math.h>
#include "Lora.h"
#include <SPI.h>

#define Aout0 A0 //MQ-135 模拟输出 接 Arduino Uno A0, 节点 1
#define Aout1 A1 //MQ-135 模拟输出 接 Arduino Uno A1,节点 2
#define Aout2 A2 //测量电压模块 模拟输出 接 Arduino Uno A1,节点 2

// 11111111 00000001 00000001 00000000 00000000 11111111
// 帧头      区域      编号      数值      电压      帧尾

int Qingdao = 1;//区域 id, 青岛
int Weihai = 2; //区域 id, 威海
int sensorid1 = 1;//传感器编号
int sensorid2 = 2;//传感器编号
int Destiny1 = 0; //临时变量, 存储 A0 读取的数据
int Destiny2 = 0; //临时变量, 存储 A1 读取的数据
int vc1 = 0; //临时变量, 存储 A3 读取的电压数据
int vc2 = 0; //临时变量, 存储 A3 读取的电压数据
int temp = 0;
/*
十进制转二进制, a 为待转参数, i 为二进制位数
*/
String dec2bin(int num,int weishu) //
{
    int i=weishu-1;
    String s="00000000"; //数组初始化赋值
    int j=i;
    while (i >= 0) {
        if ((num >> i) & 1)
        {
            s[j-i]='1';
        }
        else
        {
            s[j-i]='0';
        }
    }
}
```

```

        --i;
    }
    return s;
}
/*
    帧封装函数，将节点数据进行格式封装发送，共六个字节，格式如下：
    11111111 00000001 00000001 00000000 00000000 11111111
        帧头      区域      编号      数值      电压      帧尾
*/
String package(int citynum,int sensornum,int destiny,int vc)    //
{
    //
    /*
    String[] packagedata = {"00000000","00000000","00000000","00000000","00000000", "00000000"};
    packageData[0] = "11111111"; //帧头
    packageData[1] = dec2bin(citynum,8); //区域 id
    packageData[2] = dec2bin(sensornum,8); //传感器 id
    packageData[3] = dec2bin(destiny,8); //节点数据，255 不够可除 10 转换
    packageData[4] = dec2bin(vc,8); //节点电压值
    packageData[5] = "11111111" //帧尾
    string s="00000000"; //数组初始化赋值
    */
    String packagedata="11111111"; //帧头
    String packageData1 = dec2bin(citynum,8); //区域 id
    String packageData2 = dec2bin(sensornum,8); //传感器 id
    String packageData3 = dec2bin(destiny,8); //节点数据，255 不够可除 10 转换
    String packageData4 = dec2bin(vc,8); //节点电压值
    String packageData5 = "11111111" ;//帧尾
    packagedata.concat(packageData1);
    packagedata.concat(packageData2);
    packagedata.concat(packageData3);
    packagedata.concat(packageData4);
    packagedata.concat(packageData5);

    return packagedata;
}

void setup()
{
    // put your setup code here, to run once:
    //Serial.begin(9600);//定义波特率
    pinMode(Aout0, INPUT);//定义 A0 为 INPUT 模式

```

```

pinMode(Aout1, INPUT); //定义 A0 为 INPUT 模式
pinMode(Aout2, INPUT); //定义 A0 为 INPUT 模式
Serial.begin(9600);
//Serial.println("Sender");
//这里的 LoRa 是 LoraClass 的一个实例，在 CPP 文件中我已经给声明了，直接使用即可
while(!LoRa.begin(433E6))
{
    Serial.println("Starting LoRa failed!");
}
//Lora 模块的引脚设置在这里只能修改这三个引脚，其中 ss 和 reset 可以任意选择 IO 口，
//dio0 只能选择 2 或 3，应为只有这两个引脚有外部中断的功能
//在 begin 中已经添加了引脚设置 ss->10 dio0->2 reset->9 sck->13 miso->12 mosi->11
//如果使用默认引脚的话就不需要调用下边的这个函数
//LoRa.setPins(int ss, int reset, int dio0)
Serial.println("Starting LoRa OK!");
}

void loop()
{

    Destiny1 = analogRead(Aout0); //读取 A0 的模拟数据
    Destiny2 = analogRead(Aout1); //读取 A0 的模拟数据
    Destiny1 = Destiny1/4; //
    Destiny2 = Destiny2/4; //8 位二进制为 0-255，浓度峰值为 700 左右，封装前除 4
    vc1 = analogRead(Aout2);
    vc2 = analogRead(Aout2); //Vc1 Vc2 暂时用一个电压模块
    vc1 = vc1/4; //
    vc2 = vc2/4; // 电压模块峰值 600 左右，封装以前除 4
    String package1 = package(Qingdao,sensorid1, Destiny1,vc1);
    String package2 = package(Weihai,sensorid2, Destiny2,vc2); //包 2
    LoRa.beginPacket();
    delay(1000); //延时 1000 毫秒
    if(!temp)
    {
        LoRa.print(package1); //串口输出 temp 的数据
        Serial.println(package1);
        temp = 1;
    }
    else if(temp)
    {
        LoRa.print(package2); //串口输出 temp 的数据
        Serial.println(package2);
        temp = 0;
    }
}

```

```

delay(1000); //延时 1000 毫秒
LoRa.endPacket();
/*
String Destiny0_send="GasDet0:";
String Destiny1_send="GasDet1:";
Destiny0 = analogRead(Aout0); //读取 A0 的模拟数据
Destiny0_send.concat(String(Destiny0));
delay(1000); //延时 1000 毫秒
Destiny1 = analogRead(Aout1); //读取 A0 的模拟数据
Destiny1_send = (Destiny1_send+String(Destiny1));
delay(1000); //延时 1000 毫秒
LoRa.beginPacket();
delay(1000);
LoRa.print(Destiny0_send); //串口输出 temp 的数据
delay(1000); //延时 500 毫秒

LoRa.endPacket();
delay(1000); //延时 1000 毫秒
LoRa.beginPacket();
delay(1000);
LoRa.print(Destiny1_send); //串口输出 temp 的数据
delay(1000); //延时 500 毫秒

LoRa.endPacket();
*/
/*
static int counter;
counter++;
counter %=1000;    //
//uint8_t payload[4] = {0b00000001, 0b00000001, 0b63,0b11111111};
//单纯的发送只需要一下 3 步即可
LoRa.beginPacket();
    delay(1000);
//这里的 write 用来发送数组的
//LoRa.write(payload, sizeof(payload));
Serial.println(counter);
//print()用来发送字符串[不要使用 println() 有 bug]
LoRa.print("Hello"+String(counter));
LoRa.endPacket();
//-----
delay(1000);
*/

```

附录 B 网关 Python 程序

```
import requests
import time
import datetime
import json
import serial
import time

url_sensor0 = 'http://1.gasdetect.applinzi.com/GasDet/php/sensor0_insert.php' #post url
url_sensor1 = 'http://1.gasdetect.applinzi.com/GasDet/php/sensor1_insert.php' #避免数据库访问冲突,对
每个传感器节点分别建表
count_sensor0 = 1
count_sensor1 = 1 #节点数据量,用于服务器端获取节点获取的总信息量
zhentou = '11111111' #帧头
zhenwei = '11111111' #帧尾

while 1:
    #串口读取 ardurino 端节点数据#
    ser = serial.Serial('COM6', 115200)
    # time.sleep(2)
    # ser.write("Hello world, shake with arduino")
    time.sleep(2)
    s = ser.readline()
    s = str(s, encoding='utf-8').strip()

    if len(s)==48 and s[0:8]==zhentou and s[40:48]==zhenwei:
        locationNum = int(s[8:16],base=2) #区域 id
        sensorNum = int(s[16:24],base=2) #传感器编号
        destiny = 4 * int(s[24:32],base=2) #浓度为 2 进制专为十进制后乘 4
        vc = 4 * int(s[32:40],base=2) #节点电压为 2 进制专为十进制后乘 4
        if locationNum == 1: # 青岛
            Time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S') # 获取节点读取数
            据的时间,格式为年月日时分秒
            count_sensor0 = str(count_sensor0)
            data_sensor0 = {'Num': count_sensor0,
                            'SensorNum': sensorNum,
                            'Location': 'Qingdao',
                            'Destiny': str(destiny),
                            'Time': Time,
                            'Voltage':str(vc)}

            count_sensor0 = int(count_sensor0)
            count_sensor0 += 1
```

```

    r0 = requests.post(url_sensor0, data=data_sensor0) # sensor0 进行 post 请求
    time.sleep(1)
    # print("sensor0:" + str(count_sensor0) + "r0" + str(r0))
elif locationNum == 2:
    Time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S') # 获取节点读取数
    据的时间，格式为年月日时分秒
    count_sensor1 = str(count_sensor1)
    data_sensor1 = {'Num': count_sensor1,
                    'SensorNum': sensorNum,
                    'Location': 'Weihai',
                    'Destiny': str(destiny),
                    'Time': Time,
                    'Voltage': str(vc)}

    count_sensor1 = int(count_sensor1)
    count_sensor1 += 1
    r1 = requests.post(url_sensor1, data=data_sensor1) # sensor0 进行 post 请求
    time.sleep(1)
    # print("sensor0:" + str(count_sensor0) + "r0" + str(r0))
ser.close()

```