|  |
| --- |
| **多功能盲人拐杖的设计和实现** |

|  |
| --- |
| Design and implementation of multifunctional blind crutches |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学生姓名 | 林广海 | 学号 | | 201611911516 | | |
| 所在学院 | 电子与信息工程 | | 班级 | | | 传感1166 |
| 所在专业 | 电子科学与技术 | | | | | |
| 申请学位 | 学士 | | | | | |
| 指导教师 | 田秀云 | | 职称 | | 讲师 | |
| 副指导教师 |  | | 职称 | |  | |
| 答辩时间 | 年 月 日 | | | | | |

目 录

[摘 要 I](#_Toc40885042)

[abstract II](#_Toc40885043)

[1 绪论 3](#_Toc40885044)

[1.1 研究的目的和意义 3](#_Toc40885045)

[1.2 国内外发展状况 3](#_Toc40885046)

[1.3 相关的技术状况 3](#_Toc40885047)

[1.4 发展趋势 4](#_Toc40885048)

[2 硬件系统设计 4](#_Toc40885049)

[2.1 控制器模块 4](#_Toc40885050)

[2.1.1 STC89C54单片机简介 5](#_Toc40885051)

[2.1.2 STC89C51单片机引脚功能 5](#_Toc40885052)

[2.1.3 STC89C51单片机最小系统 7](#_Toc40885053)

[2.2 超声波测距模块 7](#_Toc40885054)

[2.2.1 HC-SR04超声波测距模块电路图 8](#_Toc40885055)

[2.2.2 HC-SR04超声波测距模块驱动方式 9](#_Toc40885056)

[2.3 光线亮度感应模块 10](#_Toc40885057)

[2.3.1 光线亮度感应模块电路图 10](#_Toc40885058)

[2.4 语音播放模块 11](#_Toc40885059)

[2.4.1 语音播放模块驱动方式 11](#_Toc40885060)

[2.5 GPS定位模块 12](#_Toc40885061)

[2.5.1 GPS定位模块电路图 12](#_Toc40885062)

[2.5.2 GPS定位模块驱动方式 13](#_Toc40885063)

[2.6 GSM无线通讯模块 14](#_Toc40885064)

[2.6.1 GSM无线通讯模块电路图 14](#_Toc40885065)

[2.6.2 GSM无线通讯模块驱动方式 14](#_Toc40885066)

[2.7 系统完整电路图 16](#_Toc40885067)

[2.8 实物完成图 16](#_Toc40885068)

[3 软件系统设计 17](#_Toc40885069)

[3.1 语音避障功能设计 17](#_Toc40885070)

[3.2 自动亮灯提醒路人避让功能设计 18](#_Toc40885071)

[3.3 播放语音提示功能设计 19](#_Toc40885072)

[3.4 GPS远程定位功能设计 20](#_Toc40885073)

[3.4.1 单片机端的程序设计 21](#_Toc40885074)

[3.4.2 服务器端的程序设计 23](#_Toc40885075)

[3.4.3 手机端的软件设计 24](#_Toc40885076)

[3.5 完整的程序运行结果 25](#_Toc40885077)

[4 结论 26](#_Toc40885078)

[鸣谢 27](#_Toc40885079)

[参考文献 28](#_Toc40885080)

[附 录 29](#_Toc40885081)

摘 要

本研究旨在设计出适用于盲人拐杖的多功能电子电路系统。它能够实现语音提示避障，黑夜里自动亮灯提醒路人避让, GPS远程定位等的功能。为了便于本系统的开发和后期维护，本系统采用了以单片机为核心，各个功能模块化设计的思路。单片机负责逻辑运算和协调各个模块的正常运作，各个模块负责实现相应的独立功能。在本系统中，其核心采用的是STC的80C54型号单片机，使用C语言为其编写运行程序，外围的主要功能模块有超声波测距模块、光线亮度感应模块、语音播放模块、GPS定位模块和GSM无线通讯模块。它们共同组成了一个完整的系统，为使用者提供一个良好的使用体验。

关键词：拐杖；单片机；测距；亮度；远程定位；

abstract

The purpose of this study is to design a multi-functional electronic circuit system for the blind crutches. It can realize voice prompt to avoid obstacles, automatic light in the night to remind passers-by to avoid obstacles, GPS remote positioning and other functions. In order to facilitate the development and later maintenance of the system, the system adopts a single-chip microcomputer as the core, the idea of modular design of each function. Single chip microcomputer is responsible for logical operation and coordination of the normal operation of each module, each module is responsible for the realization of the corresponding independent functions. In this system, the core of STC is 80c54 single chip microcomputer, which uses C language to write running program. The main peripheral function modules are ultrasonic ranging module, light brightness sensing module, voice playing module, GPS positioning module and GSM wireless communication module. They together constitute a complete system to provide users with a good use experience.

Keywords: Crutches; Single chip; Ultrasound wave; Brightness; Remote location;

**多功能盲人拐杖的设计和实现**

电子科学与技术，201611911516，林广海

指导教师：田秀云

# 绪论

## 研究的目的和意义

如今，随着科学技术的繁荣发展，各式各类的电子产品不断地出现在我们的生活中，为我们的生活水平带来了阶跃性的提高，某些传统的生活产品也正快速地被它们所替代，大到洗衣机，洗碗机，小到体温计，热水器。不可否认，它们极大地便利了我们的生活，甚至到了离不开它们的地步了。然而，对于某些特殊人群，如视觉障碍患者，他们的生活中必不可少的拐杖依旧是几十年前的老样子，没有因为科技的发展而有所改进，就好像是被时代抛弃了那样，或许是商家觉得这方面的产品市场不大，利润低，不值得花大力气去做研发，又或者是认为特殊人群的消费能力低，不值得去投资，去冒这个险。所以至今为止，市场上都还没有一款像样的多功能盲人拐杖。所以此研究将使拐杖使其相比于传统的普通拐杖，能为有视觉障碍的人群提供更加优秀的体验，减少更多的不必要的麻烦，让他们也能享受到科技进步为生活带来的便利。

## 国内外发展状况

国内和国外在盲人拐杖的设计和研究方面的发展状况相差不大，大多数的研究人员都把相关的注意力集中在了如何恢复视觉等这些高端的医疗领域，所以在普通的相关产品方面的关注度普遍不高。虽然如今的盲人拐杖相比传统那些手工制作的拐杖有了很大的进步，但这些改进大多是体现在了它的制作材料和机械结构上，它们的设计理念在不断地追求着更轻、更便捷以及更耐用。不可否认，这些外在的变化对有视觉障碍的人群也起到了很大的帮助，然而，即使它们的外表看起来多么的“高科技”，但它们相对于以普通的盲人拐杖依旧没有本质的改变，没能有效的融入现代电子技术是没有内在灵魂的，普通的拐杖甚至是一根用着比较顺手的木棍都可以轻易地把它们替代掉。虽然偶尔会有那么几款拐杖宣称其已实现了智能化，可是在研究后就会发现，其所谓的“智能”主要是指能够播放音乐和收听广播，对部分人来说，这或许很有创意。但是，这些偏娱乐化的功能似乎已经偏离了一个盲人拐杖的该有职责，所以它们并没有被社会所认可。

## 相关的技术状况

电子技术的快速发展，给我们的生活带来了极大的便利，尤其是集成电路的发明，更是引起了时代的变革，促使我们进入了信息时代。集成电路因为其具有体积小、能耗低、可靠性高等优点而获得了不可思议的发展速度，它使曾经庞然大物般的计算机微缩到了如今可托付于掌心的笔记本电脑那般大小，其中的变化不得不让人为之惊叹。现代的电子技术已经影响到了我们生活的方方面面，家里大大小小的电器里都会有它们的身影，我们的日常生活已经无法离开对它们的应用。因为电子技术的高度发达，使得我们可以在拐杖中内置一套电子系统为其实现多功能而不需要对其外形做出多大的改变，更不会因此而牺牲它们本该有的便携性，这将有助于本研究的顺利进行。

## 发展趋势

据统计，我国现有超过1700万的视觉障碍人士，相当于我们中每80人里就有一个“黑暗中的独行者”。然而，如今的医疗水平无法为它们提供有效的治疗，在相对有效的医疗手段还没有发明之前，拐杖依旧是他们出行的最直接最有帮助的工具之一。他们中曾有人坦言，如果没有拐杖在手，他们心里就会不踏实，甚至是不敢出门。所以，设计出拥有更加实用的多功能拐杖，将能在很大程度上帮助那些视觉障碍人士的日常出行。随着社会对他们关注度的增高，将会有越来越多的新技术融入到他们赖以出行的拐杖中，帮助他们与黑暗作斗争。

# 硬件系统设计

单片机

STC89C51

语音播放模块

超声波测距模块

光线亮度感应模块

GPS定位模块

GSM无线通讯模块

图 2‑1

本系统的硬件总体框架图如图 2‑1所示，单片机作为本系统的核心部件，负责处理来自超声波测距模块、光线亮度感应模块和GPS定位模块的输入信号，并根据处理后的数据做出判断是否需要驱动语音播放模块播放指定的提示音或者通过GSM无线通讯模块把GPS定位信息发送到服务器。

## 控制器模块

根据本系统的设计要求，我们决定采用STC的89C54型号单片机作为该系统的核心控制器。该类型的单片机经过多年的发展，相关的技术已经非常的成熟，而且学习资料多，这对我们的研究将会有很大的帮助。

### STC89C54单片机简介

* STC89C54是STC推出的新一代单片机，具有速度快、功耗低、抗干扰能力强等优点；
* 工作电压：5.5V-3.3V；
* 工作频率范围：0-40MHz，实际工作频率可达48MHz；
* 用户应用程序空间：16KB；
* 片上RAM：1208字节；
* 共有35个通用I/O口，复位后为除了P0口是开漏输出，其他的P1/P2/P3口都是准双向口/弱上拉；
* ISP（在系统可编程）/IAP（在应用可编程），通过串口（RXD/P3.0，TXD/P3.1）就可以在数秒的时间内完成用户程序的下载；
* 集成了EEPROM功能，其大小为45KB；
* 具有内部看门狗功能；
* 共有3个16位的定时器/计数器，其中定时器0还可以当作2个8位的定时器使用；
* 共有2个外部中断源，下降沿触发或低电平触发。
* 通用异步串行口(UART)，可通过定时器用程序模拟出多个UART；
* 工作温度范围：-40~+85℃(工业级）/0~75℃(商业级)；
* 封装形式：PDIP-40；

### STC89C51单片机引脚功能

STC89C54的外部引脚分布如图 2‑2所示，部分具有特殊功能的引脚说明如表格 2‑1所示。

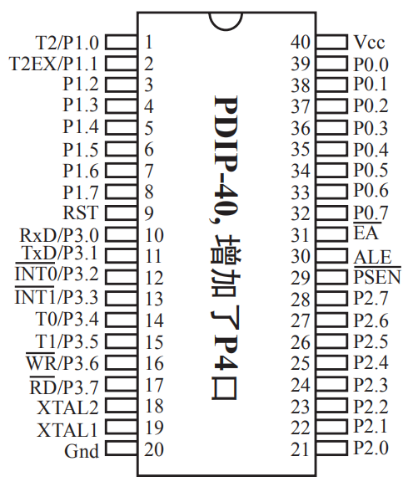


图 2‑2

|  |  |  |
| --- | --- | --- |
| 管脚 | 管脚编号 | 说明 |
|  |  | 外接上拉电阻的位准双向口或低位地址线的、数据线的 |
|  |  | 标准口或定时器/计数器外部输入端 |
|  |  | 标准口或T2捕捉/重装模式的触发控制端 |
|  |  | 标准口 |
|  |  | 位准双向口或高位地址总线 |
|  |  | 标准口或串口接收端 |
|  |  | 标准口或串口发送端 |
|  |  | 标准口或外部中断触发端 |
|  |  | 标准口或外部中断触发端 |
|  |  | 标准口或定时器/计数器外部输入端 |
|  |  | 标准口或定时器/计数器外部输入端 |
|  |  | 标准口或外部数据存储器写脉冲 |
|  |  | 标准口或外部数据存储器读脉冲 |
| P4.4/ |  | 标准口或外部程序存储器选通信号输出端 |
|  |  | 标准口或地址锁存允许信号输出端/编程脉冲输入端 |
|  |  | 标准口或内外存储器选择端 |
|  |  | 复位脚 |
|  |  | 内部时钟电路反相放大器的输入端。当直接使用外部时钟源时，此引脚是外部时钟源的输入端。 |
|  |  | 内部时钟电路反相放大器的输出端。当直接使用外部时钟源时，此引脚可悬空，此时的输出等于的输入。 |
|  |  | 电源正极 |
|  |  | 电源负极、接地端 |

表格 2‑1

### STC89C51单片机最小系统

单片最小系统指能使单片机正常工作的最小硬件单元电路，它们通常包括了单片机、时钟电路和复位电路。在本系统中，所使用的时钟电路和复位电路如图 2‑3所示。



图 2‑3

时钟电路：STC89C54单片机内部具有一个构成振荡器的高增益反相放大器。我们在XTALl和XTAL2这两个引脚间跨接石英晶体和两个补偿电容构成自激振荡器。在该电路中，晶振的频率为11.0592MHz，补偿电容为47pF的瓷片电容。

复位电路：STC89C54单片机要求向RST引脚发送持续2us以上的高电平脉冲来实现复位操作。这里采用了上电复位和手动复位两种方式。上电复位使单片机在通电源后，自动实现复位操作，其主要是通过电容C3的充电来实现。手动复位使单片机在运行期间，可以通过按键的开关来实现单片机的复位操作，其主要是通过按键将电容C3短路来实现。

## 超声波测距模块

为了实现语音避障的功能，我们需要提前知道拐杖使用者前方是否有障碍物。由于超声波具有指向性强，能量消耗缓慢，在介质中传播的距离较远等优点，因而经常用于测量与物体的距离。所以，我们可以使用超声波测距模块来感知拐杖使用者前方是否有障碍物，并通过它返回的数据计算出使用者与障碍物之间的距离，以此来判断是否需要驱动语音播放模块提醒使用者小心前方障碍物。HC-SR04是目前使用最为广泛的一款超声波测距模块（图 2‑4），它具有性价比高、测试距离远、精度高、测量一致性好、稳定可靠等优点。其主要技术参数如表格 2‑2所示：

|  |  |
| --- | --- |
| 电气参数 | 新款 HC-SR04 |
| 超声波测距芯片 | CS100 |
| 工作电压 | DC 3V-5.5V |
| 工作电流 | 5.3mA |
| 工作温度 | -40℃-85℃ |
| 输出方式 | GPIO |
| 感应角度 | 小于 15 度 |
| 探测距离 | 2cm-600cm |
| 探测精度 | 0.1cm+1% |



图 2‑4

表格 ‑2

### HC-SR04超声波测距模块电路图

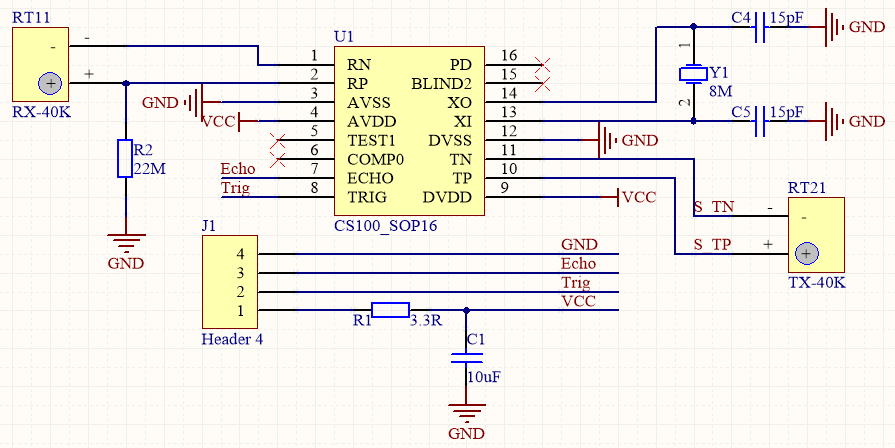


图 2‑5

图 2‑5为HC-SR04超声波测距模块的电路设计图。在该电路中，使用了一款型号为CS100\_SOP16的工业级超声波测距芯片，其内部集成了超声波发射电路，超声波接收电路，数字处理电路等，配合使用 40KHZ 的开放式超声波探头，只需要一个 22MR 的下拉电阻和 8M 的晶振，即可实现高性能测距功能单芯片即可完成超声波测距，测距结果通过脉宽的方式进行输出。其中 R1，C1 组成电源滤波电路。Y1 为 8MHZ 的晶体振荡器。RT21 为 40KHZ 的发射探头，RT11 为 40KHZ 的接收探头。R2 为 22 兆欧的下拉电阻，可以调节测距灵敏度。增大这个电阻，可以得到更远的测量距离，但也会对周围小物体的回波信号更加敏感。

### HC-SR04超声波测距模块驱动方式

该模块有4个引脚，从左到右依次编号 1,2,3,4，它们的功能定义如下：

1号 Pin：接 VCC 电源（直流 3V-5.5V）。

2 号 Pin：接外部电路的 Trig 端，向此管脚输入一个10uS 以上的高电平，可触发模块开始测距。

3 号 Pin：接外部电路的 Echo 端，当测距结束时，此管脚会输出一个高电平，电平宽度为超声波往返时间之和。

4号 Pin：接外部电路的地。

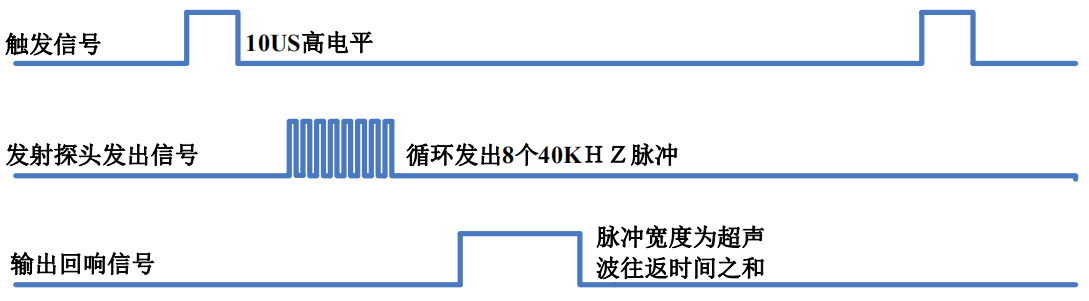
该模块的测距时序图如下：

图 2‑6

由图 2‑6可知，只需要在 Trig 管脚输入一个持续10US 以上的高电平脉冲，便可触发模块发出 8 个 40kHz 的超声波脉冲，在检测到回波信号后，便会通过 Echo 管脚输出脉冲宽度与超声波往返时间相等的高电平脉冲。因此，我们可以通过检测输出脉冲的持续时间来计算出与障碍物的距离。如果与障碍物的距离超出了测量范围，则此时返回的脉冲宽度最长，约为66ms，所以在这样的情况下可认为前方无障碍物。图 2‑7为该模块测试的数据，测试时与障碍物的距离为400cm，所得的结果虽然与实际有点偏差，但依旧符合本系统的设计要求。



图 2‑7

## 光线亮度感应模块

为了实现能够在黑夜里自动亮灯提醒路人避让的功能，我们需要实时监测当前环境的光线亮度，并以此来判断是否需要驱动提示灯亮起来。因为光敏电阻具有体积小、灵敏度高、反应速度快、光谱特性好等特点，并且能很好地符合我们的设计要求，所以我们采用光敏电阻式的光线亮度感应模块（图 2‑8）来感知当前外部环境光线亮度的变化。

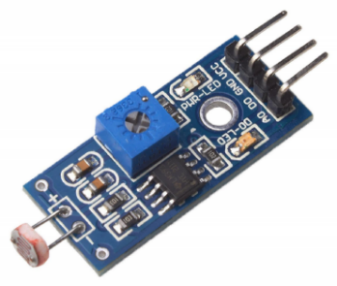


图 2‑8

### 光线亮度感应模块电路图



图 2‑9

图 2‑9为光线亮度感应模块的电路设计图。在该模块中有四个输入输出端口，它们分别为VCC端、GND端，DO端和AO端，VCC端用来接电源的正极，GND端用来接电源负极，DO端为开关量输出端（0 和 1），AO 端为模拟量输出端（电压）。LDR1为灵敏型光敏电阻，用来感知外部环境光线强度的变化，其电阻值随着光线强度的变化而发生改变。LM393为电压比较器，用来控制DO端的输出高电平或低电平，当其正输入端的电压高于负输入端时，DO端输出为高电平，当其正输入端电压低于负输入端时，DO端输出为低电平，所以滑动变阻器RV1可以用来控制DO端的输出灵敏度。AO端通过与AD 模块相连，经过AD转换，便可获得环境光线强度的精准数值。发光二极管C2为电源指示灯，发光二极管D2为DO端的开关指示灯。该模块可使用5V电源供电。

## 语音播放模块

为了实现在遇到障碍物或外部光线强度发生变化时播放语音提示的功能，我们需要一个能够通过串口指令控制的语音播放模块。图 2‑10为我们所使用的语音播放模块。在该模块中，使用的是一款型号为MH3028M-24SS串口语音芯片，它集成了 MP3、WAV 音频文件的硬解码，同时软件支持工业级别的串口通信协议。该模块以TF 卡为存储介质，通过简单的串口指令，不需要繁琐的底层操作即可完成播放指定语音等功能，使用方便，稳定可靠。该模块的参数说明如表格 2‑3所示：



图 2‑10

|  |  |
| --- | --- |
| 名称 | 参数 |
| MP3文件格式 | 支持所有比特率11172-3和 ISO13813-3 layer3音频解码，支持8/11.025/12/16/22.05/24/32/44.1/48(KHZ)采样率 |
| 输入电压 | 3.3V-5V |
| 额定电流 | 10MA[静态] |
| 低功耗电流 | <200uA |
| 功放功率 | 驱耳机，功放 |
| 工作温度 | -40℃-80℃ |
| 湿度 | 5% ~ 95% |

表格 2‑3

### 语音播放模块驱动方式

该模块共有四个引脚（GND，VCC，TX，RX），GND为接地端，VCC是电源端，TX是MP3芯片的TX串口数据发送端，RX是MP3芯片的RX串口数据接收端。

在该模块中，我们通过串口发送指令控制模块的运行，使用的指令格式为：$S VER Len CMD Feedback para1 para2 $O，其中$S为起始位，固定为0x7E，VER为版本信息，Len为有效字节数，CMD为命令字节，Feedback为是否需要反馈信息，para1和para2为指令的参数，$O为结束位，固定为0xEF。控制指令以十六进制的形式发送，通讯时使用的波特率为9600bps。通常用到的控制指令如表格 2‑4所示。如果想要播放指定的文件，TF卡中的音频文件应该按图 2‑11所示的目录结构存储，比如想要播放里面的《010-0.5米.mp3》文件，则使用的指令应该为7E FF 06 0F 00 01 0A EF，其中的01代表TF卡根目录的文件夹“01”，0A代表“01”文件夹里的带有“010”前缀的音频文件，因为十进制的010转换为十六进制就是0x0A。

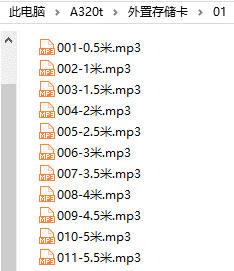


图 2‑11

|  |  |
| --- | --- |
| 发送的指令 | 作用 |
| 7E FF 06 0D 00 00 00 EF | 播放 |
| 7E FF 06 0E 00 00 00 EF | 暂停 |
| 7E FF 06 04 00 00 00 EF | 播放音量加一级 |
| 7E FF 06 05 00 00 00 EF | 播放音量减一级 |
| 7E FF 06 06 00 00 \*\* EF | 指定播放音量 |
| 7E FF 06 0F 00 \*\* \*\* EF | 播放指定的MP3文件 |
| 7E FF 06 0A 00 00 00 EF | 进入休眠模式 |

表格 2‑4

## GPS定位模块

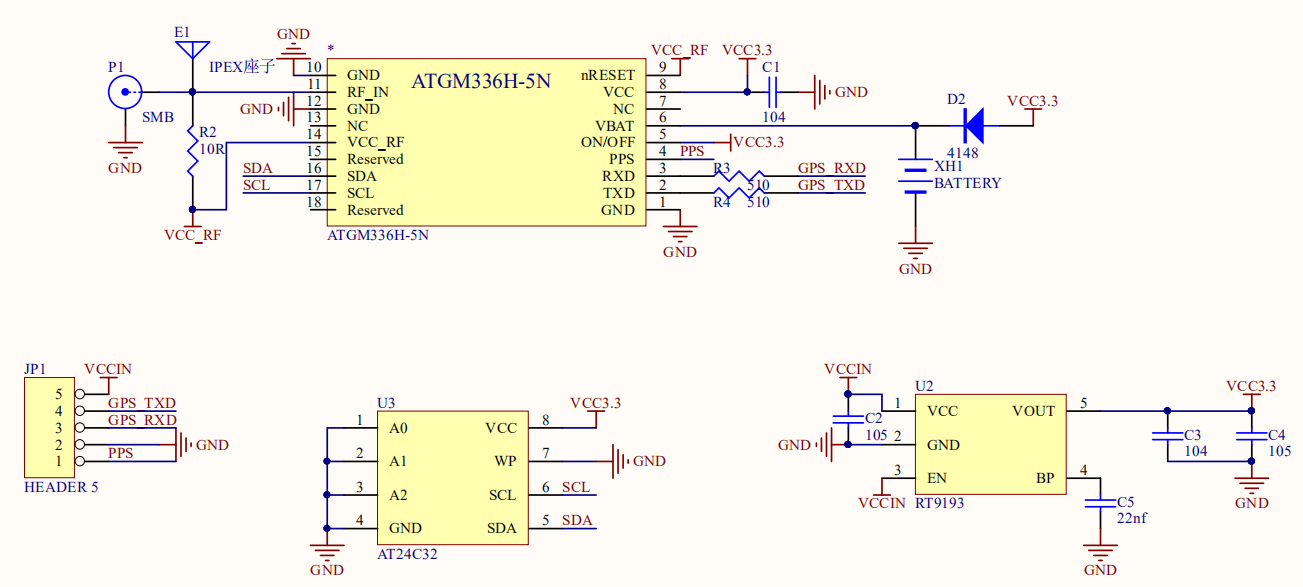
为了实现GPS远程定位的功能，我们需要使用GPS定位模块来获取如经度、纬度、海拔高度等这些地理信息。图 2‑12为本系统所使用的GPS定位模块，该模块使用的是中科微的ATGM336H芯片，该芯片支持我国的 BDS北斗导航系统和美国的 GPS导航系统，具有体积小、高灵敏度、低功耗、成本低等优点，其输出协议遵循《CASIC 多模卫星导航接收机协议规范》。



图 2‑12

### GPS定位模块电路图

图 2‑13



该模块的具体使用参数如下：

* 模块供电 3.3-5V；
* 具备 SMA 天线接口和 IPEX 天线接口；
* 板载 E2PROM 可设置保存波特率等信息；
* 板载 XH414 充电电子，加速热启动搜星；
* 支持 A-GNSS；
* 冷启动捕获灵敏度：-148dBm；
* 跟踪灵敏度：-162dBm；
* 定位精度：2.5 米（CEP50，开阔地）；
* 首次定位时间：32 秒（也有可能是几分钟，要看具体环境而定）；
* 低功耗：连续运行<25mA；
* 内置天线检测及天线短路保护功能；
* 板子尺寸 13.1mm x 15.7mm；

### GPS定位模块驱动方式

该模块通过 UART 作为主要输出通道，其共有5个输入输出引脚，它们分别为电源VCC端、接地GND端、串口通讯发送端TX和接收端RX，还有一个比较少用的PPS端，串口通讯的波特率可自由设置，我们使用默认的9600bps。该模块会以1Hz的固定频率向串口发送定位数据，无需额外的触发操作。当其定位成功时，输出的定位信息例子如下所示：

$GNGGA,084852.000,2236.9453,N,11408.4790,E,1,05,3.1,89.7,M,0.0,M,,\*48

其中$GNGGA为消息ID，084852.000为定位点的UTC时间，2236.9453为纬度，N为纬度方向，11408.4790为经度，E为经度方向，89.7为海平面高度，M为高度单位，其他位的数据用得比较少。经过测试，该模块的输出如图 2‑14所示。

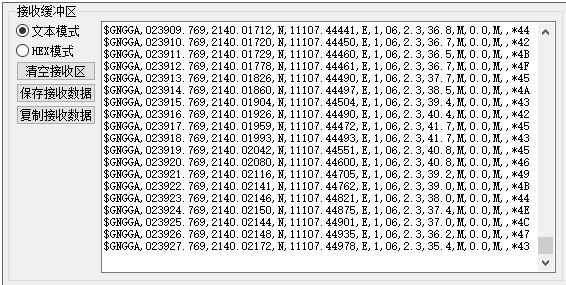


图 2‑14

## GSM无线通讯模块

为了实现远程定位功能，除了需要使用GPS定位模块获得定位数据外，还需要使用GSM无线通讯模块把数据发送到服务器。图 2‑15为我们使用的GSM无线通讯模块，一个工业级、多功能的四频段 GSM/GPRS 模块。其使用的芯片型号为SIM800C，支持移动和联通的2G网络，可通过AT命令集进行网络业务的交互，实现呼叫、短信、电话本、数据业务、传真等方面的控制。



图 2‑15

### GSM无线通讯模块电路图

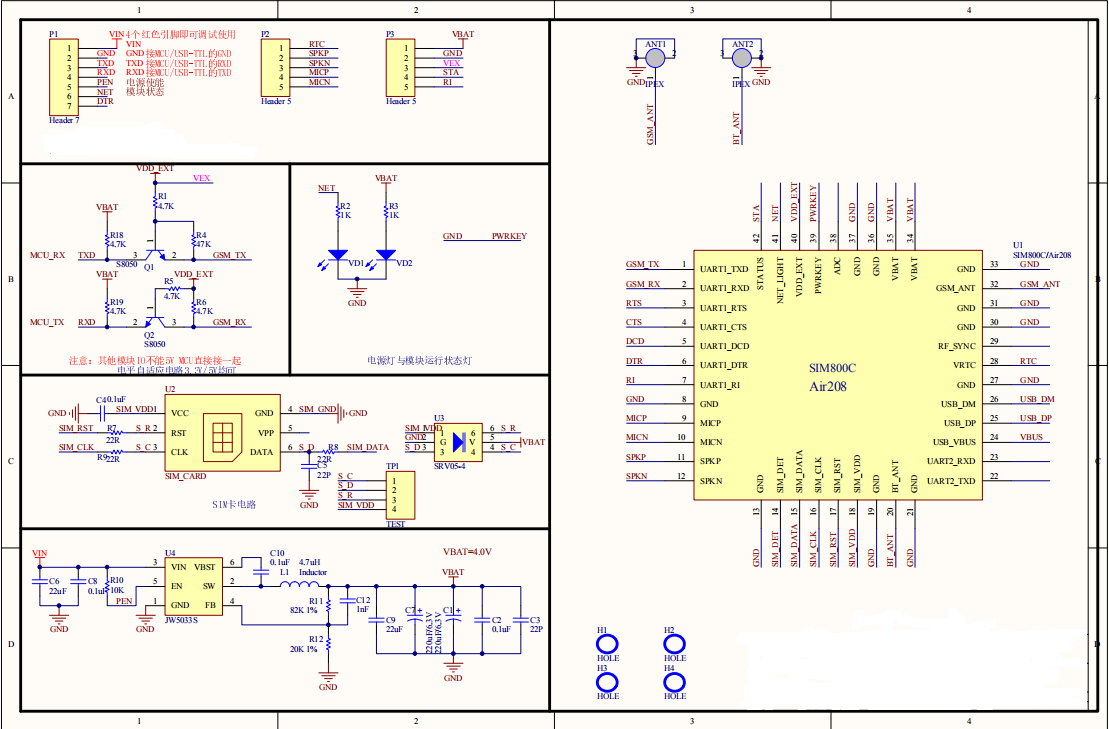


图 2‑16

### GSM无线通讯模块驱动方式

在本系统中，我们主要使用的是该模块的TCP/IP通讯功能。通过串口通讯向其发送AT指令使其成为TCP客户端，再把GPS定位模块产生的定位数据通过移动的2G网络发送到TCP服务器。表格 2‑5为在单链路非透传模式下建立一次TCP连接所需要的AT指令，除了“0x1A”这个指令外，其他的指令均要在末尾加上回车符“\n”。默认使用的波特率为9600 bps。图 2‑17为完成一次数据发送的AT指令连接过程。

|  |  |
| --- | --- |
| AT+CGATT? | 检查 GPRS 附着状态 |
| AT+CSTT=”CMNET” | 开始任务，设置 APN。移动为“CMNET”，联通为“UNINET” |
| AT+CIICR | 建立无线链路 (GPRS 或者 CSD) |
| AT+CIFSR | 获得本地 IP 地址 |
| AT+CIPSTART=”TCP”,” 120.78.203.170”,“12777” | 建立TCP连接，TCP为协议，120.78.203.170为服务器的地址，12777为服务器的端口号 |
| AT+CIPSEND | 发送数据到远端服务器，用户必须要等到响应“>”后才输入数据。 |
| 0x1A | 发送数据后的结束标志位，以16进制发送，不需要加回车符 |
| AT+CIPCLOSE | 主动关闭TCP连接 |
| AT+CIPSHUT | 关闭移动场景，如需再次连接，重复以上操作流程 |

表格 ‑5

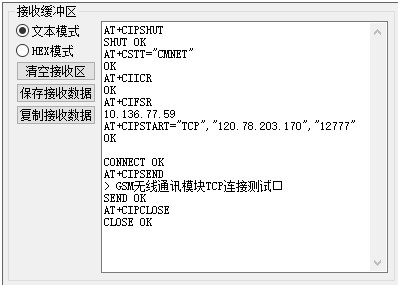
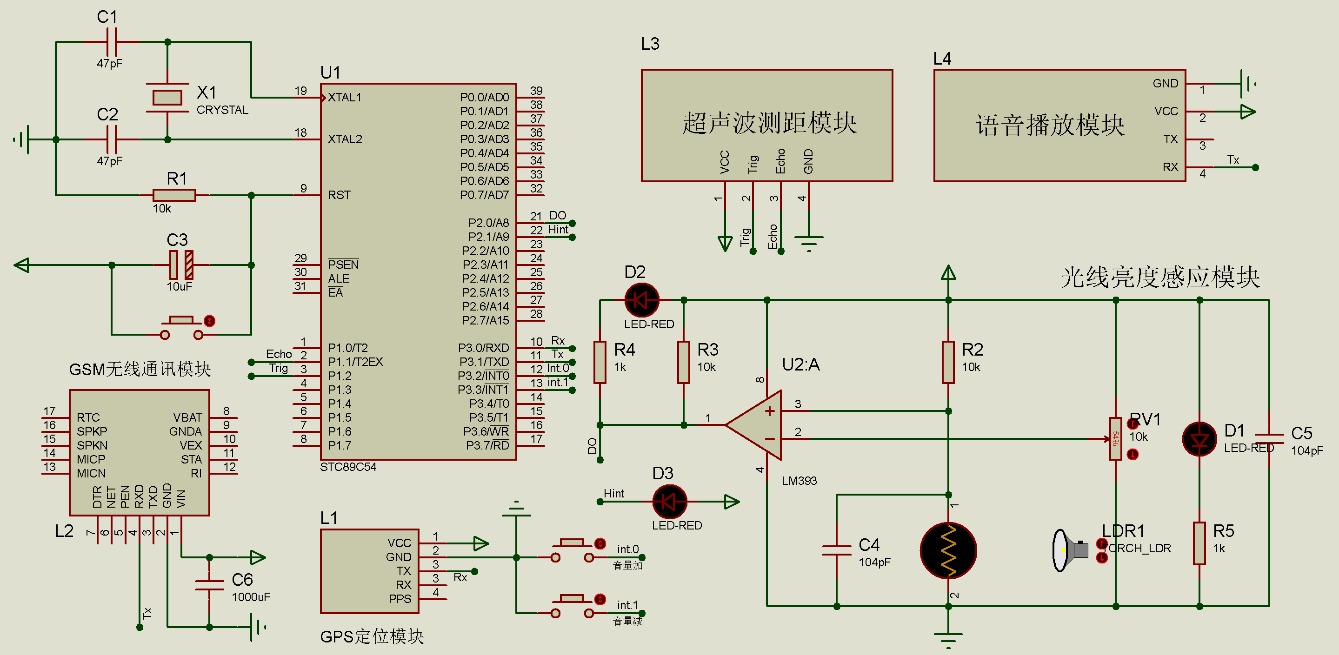


图 ‑17

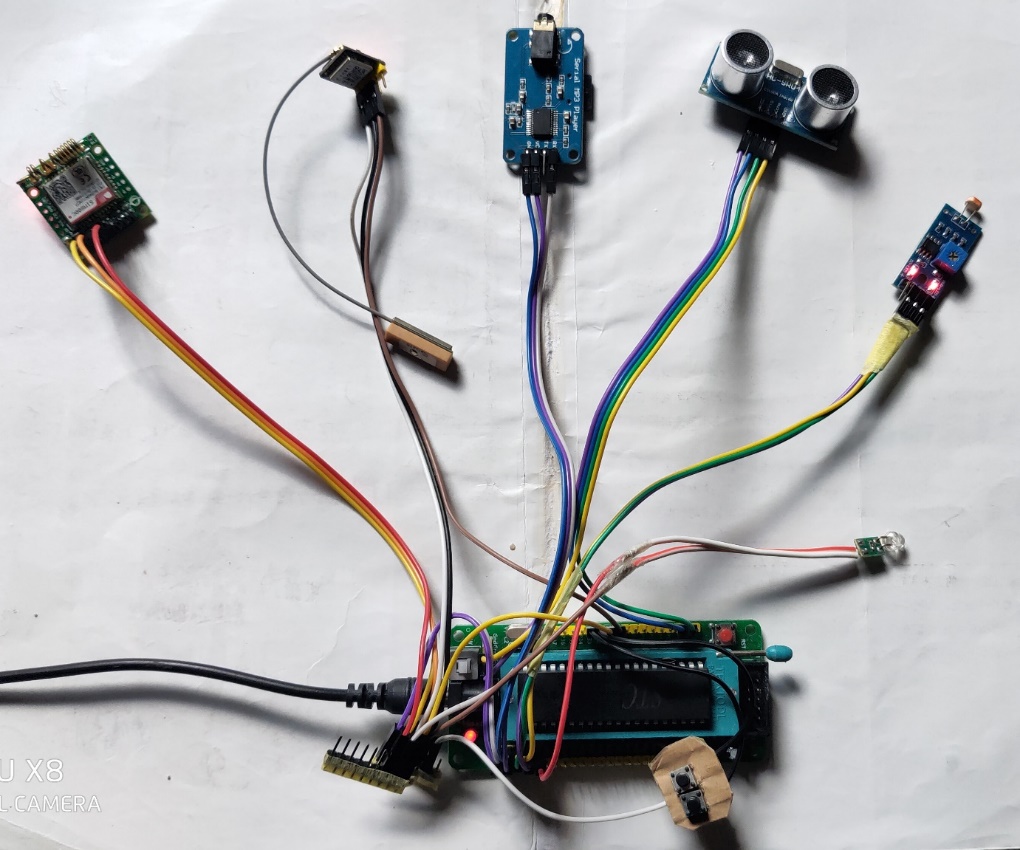
## 系统完整电路图

图 2‑18



## 实物完成图

按图 2‑19中的标号，它们分别为：①单片机控制器模块、②GSM无线通讯模块、③GPS定位模块、④语音播放模块、⑤超声波测距模块、⑥光线亮度感应模块、⑦提示灯、⑧音量加减键、⑨电源线。



**⑤**

**⑥**

**⑦**

**⑧**

**②**

**⑨**

图 2‑19

# 软件系统设计

系统想要运行起来，一套完整的运行程序是必不可少的。运行程序不仅需要处理每个模块返回的数据，还要协调它们之间相互工作。没有良好的运行程序，再优秀的电路系统也不会有优秀的使用体验，所以拥有一套能够良好运行的程序对本系统非常重要。

## 语音避障功能设计

实现该功能的程序流程图如图 3‑1所示，单片机会以固定的频率向超声波测距模块的Trig引脚发送持续10us以上的高电平脉冲，使其每隔一段时间就向前方发出测距的超声波信号，并在接收到回波信号后检测其Echo引脚的高电平脉冲的持续时间，以此来计算出与障碍物的距离，如果经过一系列的判断后认为前方确实存在障碍物且还没有被提示过，则驱动语音模块播放相关的语音提示，否则直接在延迟一定的时间后开始下一轮的障碍物测量。

为了应对不断变化的外部环境和某些不确定的因素，程序会用当前的测量数据与之前的作比较，如果差距比较大，说明外部情况有所变化，因此把变量n置0，并把当前的数据保存，否则变量n自增1。如果变量n连续自增次数累计达到2次以上，就可以认为前方确实存在障碍物，而不是某个快速路过的行人。这样能尽可能地减少误判情况的发生。

为了能够精确地得到超声波测距模块的Echo引脚输入的高电平持续时间，我们需要用到STC89C54单片机定时器T2的捕获模式。在该模式下，当外部输入T2EX由1变0时，自动将定时器2中TL2和TH2的当前值各自捕获到RCAP2L和RCAP2H。所以在单片机T2EX引脚为高电平时将TL2和TH2置0，并在T2EX引脚发生负跳变时触发T2的中断程序，在中断程序中读取RCAP2L和RCAP2H的值，以此来计算出与障碍物的具体距离。

因为T2定时器使用的计数速率是12T模式，每12个时钟得到一个计数脉冲，使用的晶振频率为11.0592MHz，所以超声波测距模块的Echo引脚高电平持续时间

取超声波在空气中传播的速度为V=340m/s，则可以通过以下公式算出与障碍物的距离

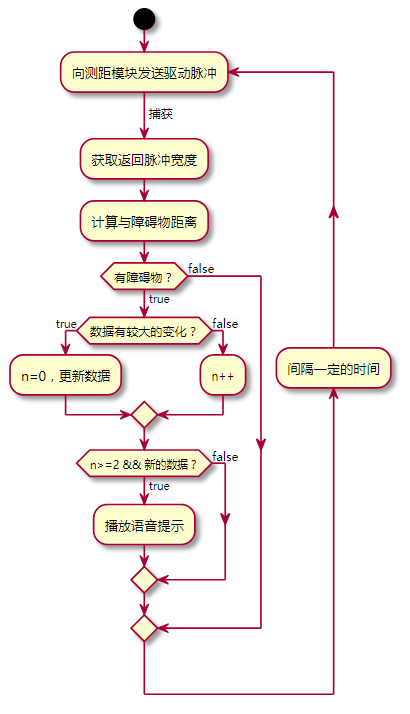
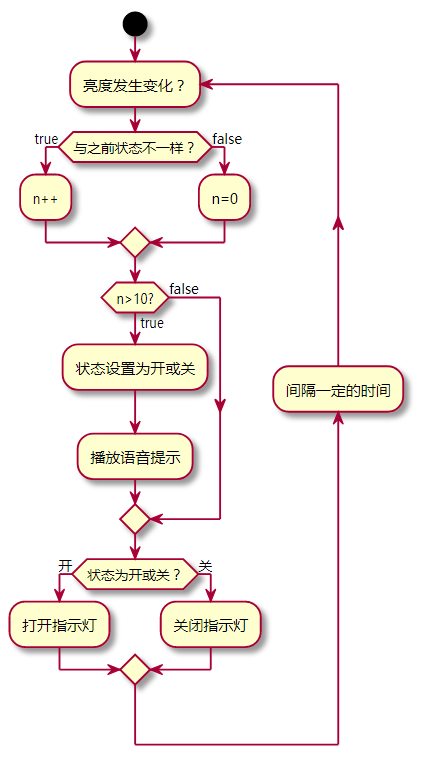


图 3‑1

## 自动亮灯提醒路人避让功能设计

实现该功能的程序流程图如图 3‑2所示。该程序会以循环的方式检查外部光线亮度的变化，如果当前的亮度与之前的状态不一样了，则使变量n自增1，否则变量n置0。如果变量n连续自增次数累计达到10次以上，则认为外部环境的光线亮度确实是发生了变化，而不是因为不经意的遮挡或其他外部干扰导致的误判。在确认亮度确实发生了变化后，会根据实际情况将提示灯的状态设置为开或关，并播放相关的语音提示。

图 3‑2



## 播放语音提示功能设计

实现该功能的程序流程图如图 3‑3所示，在系统启动时会从单片机的内部EPPROM里读取上次保存的音量值，然后通过指令把语音模块的音量初始化为上一次的状态，模块初始化完毕后，会处于等待指令的状态，其可响应的指令有播放指定文件和调节音量的大小，在响应完相关的指令后返回等待指令的状态。

在本系统中，我们分别使用89C54单片机的外部中断INT0和外部中断INT1来实现语音模块的音量加减功能，外部中断使用的是下降沿的触发方式。在中断函数中，会把当前的音量值加1或减1，然后通过串口通讯发送相关的指令使语音模块音量加1或减1，同时把当前的音量值重新写入单片机内部的EPPROM，方便下次模块初始化时使用。

因为在发送指令时需要使用串口通讯，所以需要设置合适的波特率。在与模块通讯时，使用的是串行口工作模式1，此模式为8位UART格式，一帧信息为10位：1位起始位，8位数据位(低位在先)和1位停止位，波特率可变，这里选用定时器1的工作方式2（8位自动重装）作为波特率的溢出率。当单片机工作在12T模式时，定时器/计数器T1的溢出率：

串行口工作模式1的波特率计算公式：

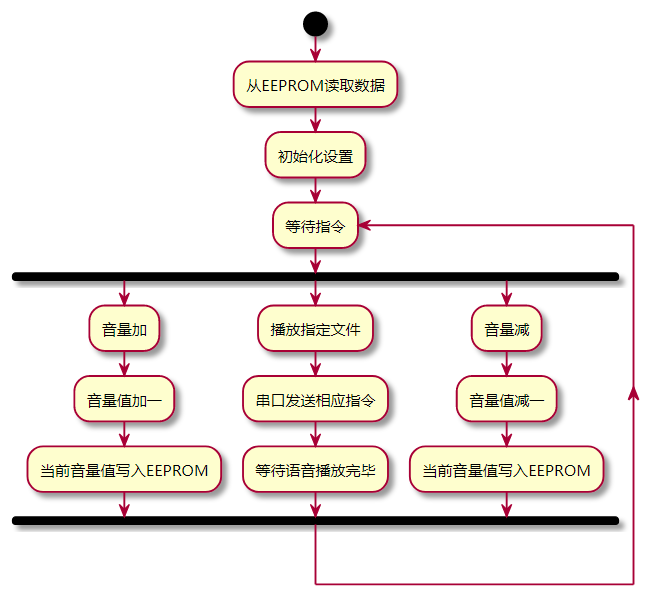
所以在晶振频率为11.0592MHz，波特率为9600，SMOD=1时，定时器T1装载初值：

图 3‑3

## GPS远程定位功能设计

实现GPS远程定位功能，除了需要本系统外，还需要TCP服务器和手机端的应用软件。它们的关系如图 3‑4所示，GSM无线通讯模块负责把GPS定位数据发送到服务器，服务器在接收到GSM无线通讯模块发送的数据后，将其保存起来，当手机端向服务请求GPS定位数据时，服务器把GPS定位数据发送给手机端。它们共同实现了远程定位功能。

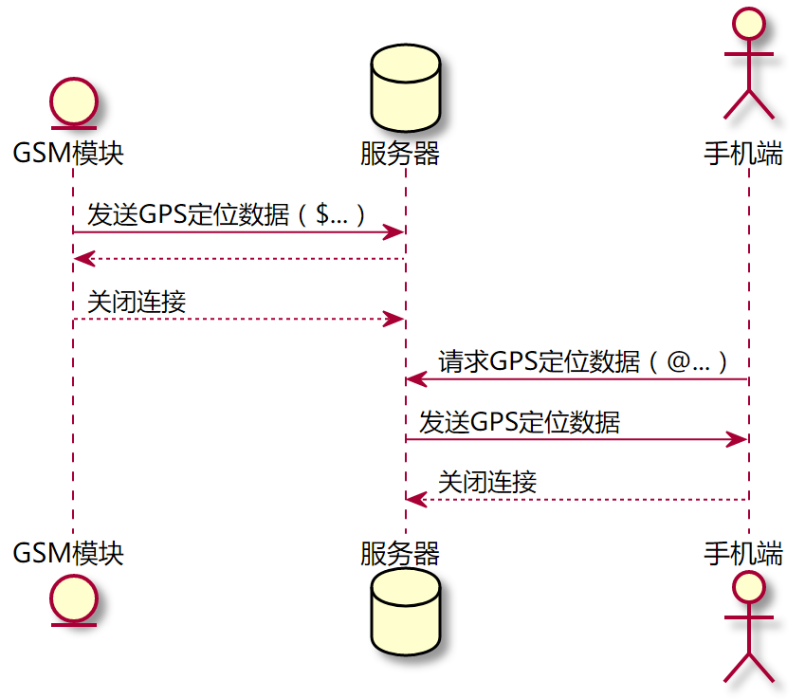


图 3‑4

### 单片机端的程序设计

在单片机端的程序主要处理的是接收和保存来自GPS定位模块的定位数据，并把定位数据通过GSM无线通讯模块发送到TCP服务器。它们的程序流程图分别如图 3‑5和图 3‑6所示。在图 3‑5中，receive数组用于保存接收到的临时数据，当接收到回车符时，说明一条数据接收完毕，此时需要判断该条数据是否为有效的GPS定位数据，如果是则把其保存到gpsSave数组中，否则直接开始新一轮的数据接收。图 3‑6为循环向GSM无线通讯模块发送AT指令的过程，每完成一次循环，就会建立一次TCP连接，并把gpsSave数组中的定位数据发送出去。

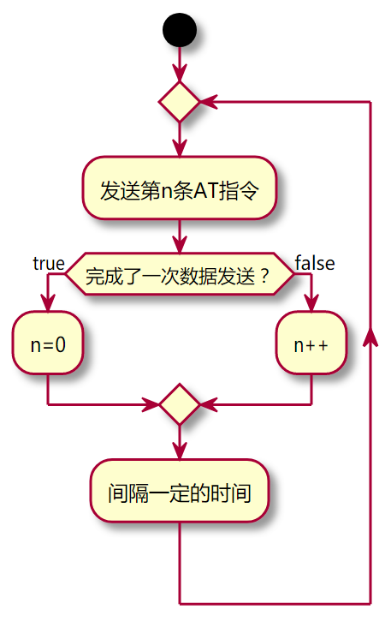
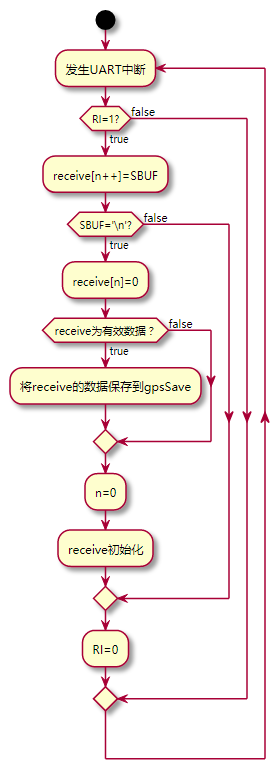


图 3‑5

图 3‑6

### 服务器端的程序设计

服务器端的程序流程图如图 3‑7所示，它根据发送过来的数据判断当前的连接是来自GSM无线通讯模块还是来自手机应用端，并以此做出相应的响应。其中GSM模块发送的定位数据以“$”字符开头，手机端应用发送的请求数据以“@”字符开头。

假设我们有多个多功能拐杖，同时也有多个手机应用端，那么为了使服务器端的程序能够准确无误地为它们提供定位功能，我们需要分别为每个拐杖设置不同ID编码来区分它们，并在发送定位数据时一起发送给服务器。这里我们使用“#”字符把定位数据和ID编码拼接起来。

为了方便数据的管理，我们还需要用到数据库把GSM模块发送过来的定位数据保存起来。图 3‑8为我们所设计的数据表结构，它包含了DeviceId、GpsData和UpdateTime这三个字段，分别用于保存一条数据中的ID编码、GPS定位数据和最近定位时间这三条信息。

该程序的实现语言为Java，使用了Netty客户端/服务器框架和mysql数据库，运行在阿里云的ECS云服务器上，公网IP为 “120.78.203.170”，操作系统为CentOS 7.4 64位，Java运行环境为JRE1.9。

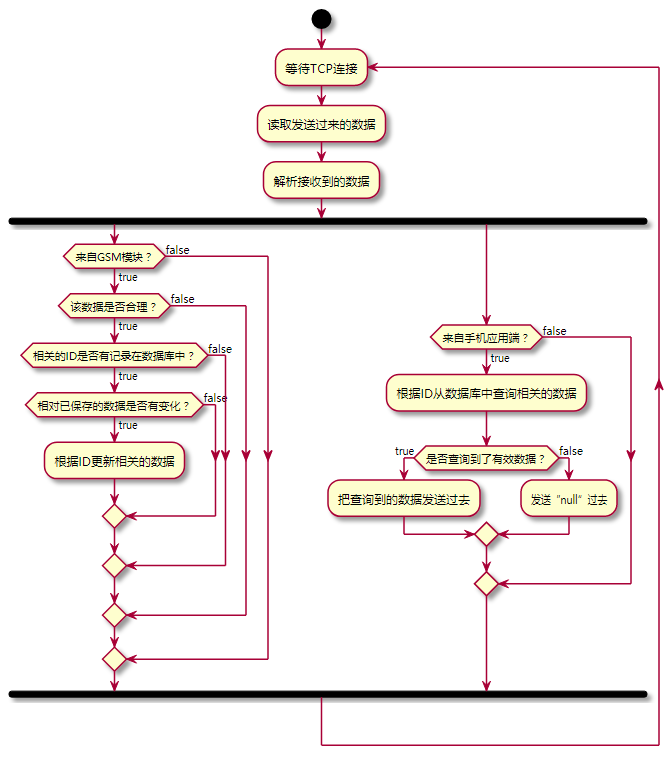


图 3‑7

### 手机端的软件设计

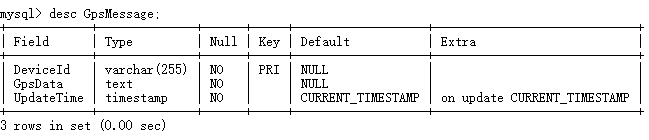


图 3‑8

通过手机应用端，我们可以向服务器查询拐杖的最新定位信息，并把查询到的数据解析后可视化地在地图上显示出来，定位软件的实现截图如图 3‑9所示。在开始定位之前，我们还需要先设置好需要定位的拐杖ID编号，其设置页面如图 3‑10所示，唯有该ID编号正确，才能从服务器上获取到对应的定位信息。



图 3‑10



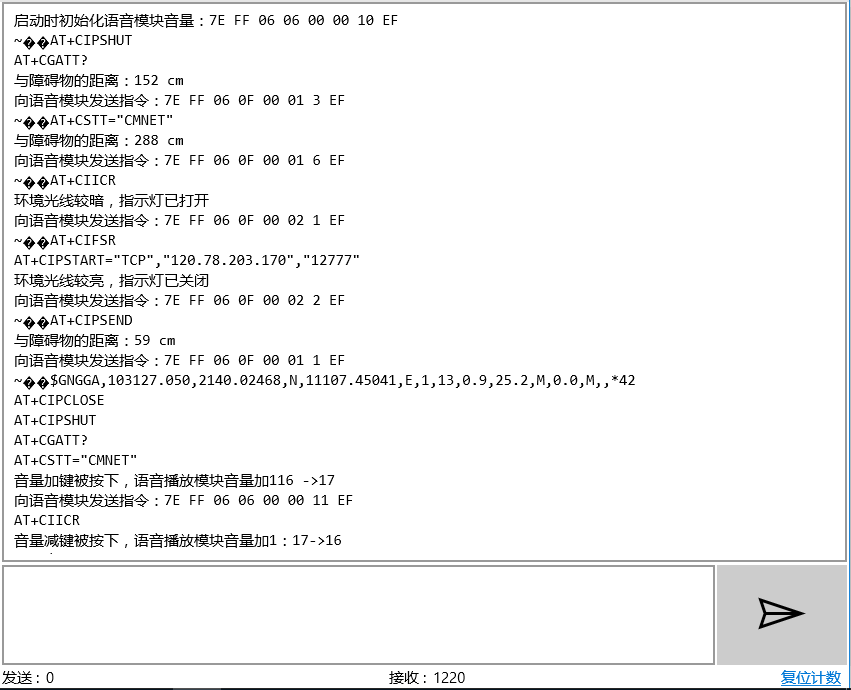
图 3‑9

在图 3‑9所示的页面中，通过点击下方的“更新定位”控件，可获取到最新的定位信息，通过点击右上角的第一个控件，可以打开设置ID编号的窗口，通过点击右上角第二个控件，可以自由切换地图的类型，这里提供了普通模式、卫星模式、夜景模式、导航模式、公交模式这五种类型。

该应用的实现语言为Java，集成开发环境为AndroidStudio 3.5，使用了高德开放平台提供的地图SDK。

## 完整的程序运行结果

为了能够比较直观的了解程序的运行状态，我们在程序代码中的关键部位加入了相应的打印代码。图 3‑11为单片机程序通过串口调试助手打印出的运行过程，其中里面乱码的部分实际为驱动语音播放模块的指令，因为其格式为16进制，所以在文本的接收模式下无法正常显示。在测试的时候，我们模拟了各种的使用场景，得到的运行结果均达到了预期的设计要求。图 3‑12为Windows系统里的Power Shell终端，我们通过它登陆阿里云的ESC服务器，并在服务器端的程序启动后打印出相关运行的过程。



循环发送的AT指令

发送由GPS模块获取的定位信息

GSM模块与服务器连接

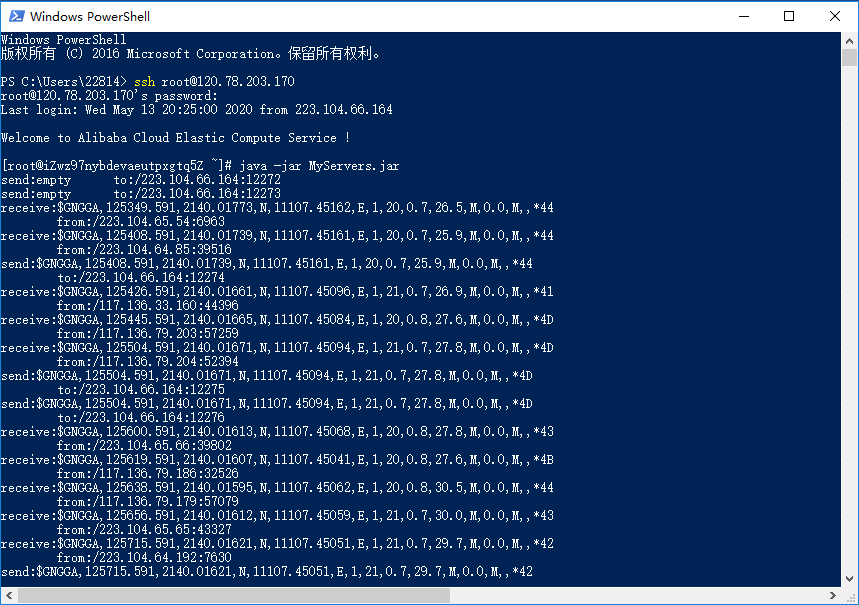
由超声波测距模块获得的与障碍物的距离

按下音量加减键时发送相关指令

根据环境光线亮度开启或关闭提示灯

图 3‑11

# 结论



登陆ESC服务器令

开始运行服务器端的程序令

接收到来自GSM模块的定位数据

当接收到手机端的数据请求时，把定位数据发送过去

图 3‑12

本系统在经过不断的调试和改进后，体验效果基本达到了设计要求，并且能够长时间地稳定运行。对于某些可能会引起误判的情况，也在程序的设计上特意做了优化，加入了不少的判断逻辑，使得误判的情况较少发生，几乎不影响使用。

本系统能够自动检测与障碍物得距离，自动检测当前环境光线亮度的变化，从而为盲人拐杖提供语音避障、亮灯提醒路人避让等功能。它提供的信息能较好地帮助使用者“感知”外部环境的变化，做到心里有数，放心出行。同时，它还提供了远程定位的功能，其他人可以实时地掌握使用者地位置信息，提供多一层地安全保障，让家里人安心。

本系统具有使用简单、成本低、抗干扰能力强等优点，但也避免不了有缺点，比如功耗较大这个问题就还没能很好地解决，同时定位功能也还不够完善，在环境条件不好时会出现定位失败或数据上传失败的情况。

从开始的不知所措，到最后任务的顺利完成，此次的毕业设计，给我带来的收获颇大。为了顺利完成毕业设计，我查阅了大量的资料，不仅完成了对以前所学知识的梳理，加深了对它们的理解，还学习到了不少更加专业的知识，特别是对技术文档的阅读能力有了很大的进步，这将对我以后的工作有很大的帮助。

鸣谢

时光匆匆如流水，转眼间便到了大学毕业季。然而，今年与往年有所不同，主角们由师兄师姐变成了我们自己，回忆起在校园里的过往岁月，真是感慨颇多。

在本论文的撰写期间，田秀云老师给予了我大量的指导意见，同时为我指点迷津，开阔研究思路。无论是论文的选题，框架的设计，还是论文的修改定稿，田老师都倾注了大量的时间和心血。在此，向田老师表示衷心的感谢！

当然，还要感谢曾经教导过我的老师们。四年以来，他们监督我学习了相关的专业知识，一步一步地丰富了我的知识体系，让我在大学里学有所成。如果没有他们无私的奉献，我的毕业论文也难以顺利地完成。

最后，特别感谢我的父母，他们给以了我大量的支持和鼓励，让我能够在学校里专心完成学业。

参考文献

[1]王骥，肖明明，杜爽，王立臣.模拟电路分析与设计(第二版)[M]. 北京: 清华大学出版社，2016.

[2]阎石. 数字电子技术基本教程[M]. 北京:清华大学出版社，2007.

[3]林立，张俊亮. 单片机原理及应用(第4版)[M]. 北京: 电子工业出版社，2018.

[4]张道德，杨光友. 单片机接口技术(C51版)[M]. 北京: 中国水利水电出版社，2007

[5]史蒂芬·普拉达. C Primer Plus 第6版 中文版[M].姜佑，译. 北京: 人民邮电出版社，2016

[6]江文杰，施建华，谢文科，马浩统，曾学文. 光电技术(第二版)[M]. 北京: 科学出版社，2014

[7]吴建平，彭颖，覃章健. 传感器原理及运用[M]. 北京: 机械工业出版社，2016

[8]埃克尔. Java编程思想(第4版)[M]. 陈昊鹏，译. 北京: 机械工业出版社，2007

[9]李刚. 疯狂Android讲义(第4版)[M]. 北京: 电子工业出版社，2019

附 录

单片机端程序代码：

#include <reg52.h>

#include <intrins.h>

#include <string.h>

#define ID "#0000000001"

#define IAP\_ADDR 0x4000 //内部REPPROM存储语音模块音量值的地址

#define uchar unsigned char

#define uint unsigned int

sfr WDT\_CONTR = 0xE1; //定义看门狗控制寄存器

sfr IAP\_DATA = 0xE2; //定义操作EPPROM相关的特殊功能寄存器

sfr IAP\_ADDRH = 0xE3;

sfr IAP\_ADDRL = 0xE4;

sfr IAP\_CMD = 0xE5;

sfr IAP\_TRIG = 0xE6;

sfr IAP\_CONTR = 0xE7;

sbit Echo = P1 ^ 1; //定义各类引脚

sbit Trig = P1 ^ 2;

sbit P2\_0 = P2 ^ 0;

sbit P2\_1 = P2 ^ 1;

bit lightStatus = 0; //定义各类标志位和辅助变量

bit distanceNew = 0;

uchar distanceNum = 0;

uchar lightNum = 0;

uchar lightStatusNum = 0;

uchar distanceNewNum = 0;

uchar distanceLastRange = 0x00;

uint distanceLast = 0;

uchar mediaCommand[] = {0x7E, 0xFF, 0x06, 0x00, 0x00, 0x00, 0x00, 0xEF}; //语音播放模块的指令

uchar volume = 0x0f; //语音模块的当前声音大小

uchar \*gsmCmd[10] = {"AT+CIPSHUT\n", "AT+CGATT?\n", "AT+CSTT=\"CMNET\"\n",

"AT+CIICR\n", "AT+CIFSR\n", "AT+CIPSTART=\"TCP\",\"120.78.203.170\",\"12777\"\n",

"AT+CIPSEND\n", "gps", "\x1A", "AT+CIPCLOSE\n"}; //循环发送的AT指令

uchar gsmNum = 0, gsmIndex = 0; //控制AT指令的发送顺序

uint receiveCount = 0;

uchar receive[100], gpsSave[150]; //接收和保存GPS定位数据

void delay(uint i) //延时函数

{

uint n;

for (n = 0; n < i; n++)

{

\_nop\_();

}

}

void sendData(uchar str[], uchar lenth) //向串行口发送数据

{

uint n;

for (n = 0; n < lenth; n++)

{

SBUF = str[n];

while (TI == 0)

;

TI = 0;

}

}

void sendAt() //发送AT指令

{

sendData(gsmCmd[gsmIndex], strlen(gsmCmd[gsmIndex]));

gsmNum = 0;

gsmIndex++;

gsmIndex %= 10;

delay(1000);

}

void IapIdle() //进入待机模式，无ISP/IAP操作

{

IAP\_CONTR = 0;

IAP\_CMD = 0;

IAP\_TRIG = 0;

IAP\_ADDRH = 0x80;

IAP\_ADDRL = 0;

}

uchar IapReadByte(uint addr) //对内部RPPROM进行读取数据操作

{

uchar dat;

IAP\_CONTR = 0x81;

IAP\_CMD = 0x01;

IAP\_ADDRL = addr;

IAP\_ADDRH = addr >> 8;

IAP\_TRIG = 0x46;

IAP\_TRIG = 0xb9;

\_nop\_();

dat = IAP\_DATA;

IapIdle();

return dat;

}

void IapProgramByte(uint addr, uchar dat) //对内部REPPROM进行写入数据操作

{

IAP\_CONTR = 0x81;

IAP\_CMD = 0x02;

IAP\_ADDRL = addr;

IAP\_ADDRH = addr >> 8;

IAP\_DATA = dat;

IAP\_TRIG = 0x46;

IAP\_TRIG = 0xb9;

\_nop\_();

IapIdle();

}

void IapEraseSector(uint addr) //擦除内部EPPROM的指定扇区

{

IAP\_CONTR = 0x81;

IAP\_CMD = 0x03;

IAP\_ADDRL = addr;

IAP\_ADDRH = addr >> 8;

IAP\_TRIG = 0x46;

IAP\_TRIG = 0xb9;

\_nop\_();

IapIdle();

}

void Time0() interrupt 1 using 0 //定时器T0中断函数

{

if (++lightNum >= 5) //检查外部光线强度的变化

{

if (P2\_0 != lightStatus)

{

lightStatusNum++;

}

else

{

lightStatusNum = 0;

}

if (lightStatus)

{

P2\_1 = !P2\_1; //指示灯闪烁

}

lightNum = 0;

}

gsmNum++;

distanceNum++;

TH0 = 0x4C; //重新装载初值

TL0 = 0x00;

}

void Time2() interrupt 5 using 1 //定时器T2的中断函数

{

if (EXF2) //使用定时器T2的捕获模式

{

double rcap = (RCAP2H << 8 | RCAP2L); //获取发生中断时的TH2和TL2的值

uint distance = (rcap \* (12 / 11.0592) \* 0.034) / 2; //计算出与障碍物的距离

EXF2 = 0;

if (distance < (distanceLast - 50) || distance > (distanceLast + 50)) //判断与障碍的距离是否发生了较大的变化

{

distanceLast = distance;

distanceNewNum = 0;

distanceNew = 1;

}

else

{

distanceNewNum++;

}

}

}

void serial\_4() interrupt 4 using 3 //UART中断，主要用来接收GPS模块的定位数据

{

if (RI)

{

uchar temp = SBUF;

RI = 0;

receive[receiveCount++] = temp;

if (temp == '\n')

{

receive[receiveCount] = 0;

if (receive[0] == '$' && receive[3] == 'G' && receive[4] == 'G' && receive[5] == 'A' && receiveCount >= 60 && receiveCount < 100)

{

strcpy(gpsSave, receive);

strcat(gpsSave, ID);

}

receiveCount = 0;

receive[0] = 0;

}

}

}

void external\_0() interrupt 0 using 2 //外部中断INT0的中断函数，语音模块音量加1

{

if (volume < 30)

{

mediaCommand[3] = 0x06;

mediaCommand[6] = ++volume;

sendData(mediaCommand, 8);

IapEraseSector(IAP\_ADDR);

IapProgramByte(IAP\_ADDR, volume);

}

}

void external\_1() interrupt 2 using 2 //外部中断INT1的中断函数，语音模块音量减1

{

if (volume > 5)

{

mediaCommand[3] = 0x06;

mediaCommand[6] = --volume;

sendData(mediaCommand, 8);

IapEraseSector(IAP\_ADDR);

IapProgramByte(IAP\_ADDR, volume);

}

}

void main()

{

EA = 1; //开总中断

ET0 = 1; //开T0中断

TR0 = 1;

TMOD |= 0x01;

TH0 = 0x4C;

TL0 = 0x00;

TR1 = 1; //设置定时器T1为波特率发生器

TMOD |= 0x20;

TL1 = 0xFD;

TH1 = 0xFD;

ET2 = 1; //开定时器/计数器T2的捕获模式

TR2 = 1;

EXEN2 = 1;

CP\_RL2 = 1;

ES = 1; //开UART中断，设置串行口通讯的工作模式

SM0 = 0;

SM1 = 1;

REN = 1;

EX0 = 1; //开外部中断INT0和INT1

EX1 = 1;

IT0 = 1;

IT1 = 1;

WDT\_CONTR = 0x37; //开看门狗，在受到干扰使程序跑飞时自动复位

memset(receive, 0, 100);

memset(gpsSave, 0, 100);

gsmCmd[7] = gpsSave;

delay(20000); //等待语音模块启动完毕后

volume = IapReadByte(IAP\_ADDR);

mediaCommand[3] = 0x06;

mediaCommand[6] = volume;

sendData(mediaCommand, 8); //初始化语音模块的音量大小

while (1)

{

if (distanceNum >= 10) //在间隔一定的时间后向测距模块发送高电平脉冲

{

Trig = 0;

delay(10);

Trig = 1;

while (!Echo)

; //等待测距模块返回高电平信号

TH2 = 0x00;

TL2 = 0x00;

distanceNum = 0; //喂看门狗

WDT\_CONTR = 0x37;

}

if (distanceNew && distanceNewNum >= 2) //如果测距的数据发生新的变化

{

uchar distanceCurRange = 0x00;

if (distanceLast >= 25 && distanceLast < 75) //根据新的测距数据确定要播放的语音文件

{

distanceCurRange = 0x01;

}

if (distanceLast >= 75 && distanceLast < 125)

{

distanceCurRange = 0x02;

}

if (distanceLast >= 125 && distanceLast < 175)

{

distanceCurRange = 0x03;

}

if (distanceLast >= 175 && distanceLast < 225)

{

distanceCurRange = 0x04;

}

if (distanceLast >= 225 && distanceLast < 275)

{

distanceCurRange = 0x05;

}

if (distanceLast >= 275 && distanceLast < 325)

{

distanceCurRange = 0x06;

}

if (distanceLast >= 325 && distanceLast < 375)

{

distanceCurRange = 0x07;

}

if (distanceLast >= 375 && distanceLast < 425)

{

distanceCurRange = 0x08;

}

if (distanceLast >= 425 && distanceLast < 475)

{

distanceCurRange = 0x09;

}

if (distanceLast >= 475 && distanceLast < 525)

{

distanceCurRange = 0x0A;

}

if (distanceLast >= 525 && distanceLast < 575)

{

distanceCurRange = 0x0B;

}

if (distanceCurRange != 0x00 && distanceCurRange != distanceLastRange) //如果前方有障碍物或已经还没提示过了，则发送相关指令

{

distanceLastRange = distanceCurRange;

mediaCommand[3] = 0x0F;

mediaCommand[5] = 0x01;

mediaCommand[6] = distanceCurRange;

sendData(mediaCommand, 8); //向语音模块发送指令

delay(13500);

WDT\_CONTR = 0x37; //喂看门狗

}

distanceNew = 0;

distanceNewNum = 0;

}

if (lightStatusNum >= 10) //如果外部环境亮度发生了变化，且持续了一段时间

{

lightStatus = P2\_0;

P2\_1 = !P2\_0;

mediaCommand[3] = 0x0F;

mediaCommand[5] = 0x02;

mediaCommand[6] = lightStatus ? 0x01 : 0x02;

sendData(mediaCommand, 8); //向语音模块发送指令

delay(13500);

WDT\_CONTR = 0x37; //喂看门狗

lightStatusNum = 0;

}

switch (gsmIndex) //分别控制每条AT指令的发送间隔时间

{

case 0: //AT+CIPSHUT

if (gsmNum > 10)

{

sendAt();

}

break;

case 1: //AT+CGATT?

if (gsmNum > 10)

{

sendAt();

}

break;

case 2: //AT+CSTT="CMNET"

if (gsmNum > 10)

{

sendAt();

}

break;

case 3: //AT+CIICR

if (gsmNum > 50)

{

sendAt();

}

break;

case 4: //AT+CIFSR

if (gsmNum > 50)

{

sendAt();

}

break;

case 5: //AT+CIPSTART="TCP","120.78.203.170","12777"

if (gsmNum > 10)

{

sendAt();

}

break;

case 6: //AT+CIPSEND

if (gsmNum > 150)

{

sendAt();

}

break;

case 7: //gps定位数据

if (gsmNum > 50)

{

sendAt();

}

break;

case 8: //0x1A

if (gsmNum > 5)

{

sendAt();

}

break;

case 9: //AT+CIPCLOSE

if (gsmNum > 10)

{

sendAt();

}

break;

}

}

}

服务器端程序代码：

import io.netty.channel.nio.\*;

import io.netty.buffer.\*;

import io.netty.channel.socket.\*;

import io.netty.bootstrap.\*;

import io.netty.channel.socket.nio.\*;

import io.netty.util.\*;

import io.netty.channel.\*;

import java.net.\*;

import java.nio.charset.\*;

import java.sql.\*;

public class MyServers {

private int port;//定义需要用到全局字段

private EventLoopGroup doConnect;

private EventLoopGroup doHandle;

private PreparedStatement getData;

public MyServers(int port) {

this.port = port;

try {

Class.forName("com.mysql.cj.jdbc.Driver");//加载mysql数据库驱动

//连接到mysql数据库

Connection toMysql = DriverManager.getConnection("jdbc:mysql://localhost:3306/lghDataBase", "user", "password");

// 预编译SQL查询语句

getData = toMysql.prepareStatement("SELECT \* FROM GpsMessage WHERE DeviceId=?;", ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

} catch (Throwable e) {

e.printStackTrace();

}

}

public void run() {

try {

doConnect = new NioEventLoopGroup();// 主线程组, 处理连接；

doHandle = new NioEventLoopGroup();// 从线程组, 处理任务；

ServerBootstrap bootLin = new ServerBootstrap();//netty服务器创建，服务器启动器

bootLin.group(doConnect, doHandle)// 设置主从线程组

.channel(NioServerSocketChannel.class)// 设置nio的双向通道

.childOption(ChannelOption.SO\_KEEPALIVE, true)

.childHandler(new ChannelInitializer<SocketChannel>() {

@Override

protected void initChannel(SocketChannel sc) {

//添加处理连接的适配器类

sc.pipeline().addLast(new MyChInHandlerAdapter());

}

});

ChannelFuture chFu = bootLin.bind(port).sync();// 启动server，并且设置端口号，同时启动方式为同步

chFu.channel().closeFuture().sync();// 监听关闭的channel，设置位同步方式

} catch (InterruptedException e) {

e.printStackTrace();

} finally {

doConnect.shutdownGracefully();

doHandle.shutdownGracefully();

}

}

private class MyChInHandlerAdapter extends ChannelInboundHandlerAdapter {

//该方法在接收到连接时自动调用

public void channelRead(ChannelHandlerContext chc, Object msgLin) {

ByteBuf inMsg = (ByteBuf) msgLin;

InetSocketAddress address = (InetSocketAddress) chc.channel().remoteAddress();

try {

String str = inMsg.toString(StandardCharsets.UTF\_8);//把接收的数据转化为字符串

// 如果接收到的数据以"$"开头，则是来自拐杖的定位数据

if (str.startsWith("$")) {

String[] temMsg = str.split("#");

//检查数据的合法性

if (temMsg.length == 2) {

String temGpsDate = temMsg[0].trim();//提取GPS定位信息

String temDeviceId = temMsg[1].trim();//提取设备ID号

getData.setString(1, temDeviceId);

ResultSet resultSet = getData.executeQuery();//执行查询语句

//如果该设备ID在数据库中存在且获得的定位信息与之前的不一样，则更新数据

if (resultSet.next() && !resultSet.getString(2).equals(temGpsDate)) {

resultSet.updateString(2, temGpsDate);

resultSet.updateTimestamp(3, new Timestamp(System.currentTimeMillis()));

resultSet.updateRow();

System.out.println("receive:" + temMsg[0] + "\tfrom:" + address.toString());//向控制台打印接受事件

}

}

}

// 如果接收到的数据以"@"开头，则是手机端在请求定位数据

if (str.startsWith("@")) {

getData.setString(1, str.substring(1).trim());//根据请求的设备ID号查询数据

ResultSet dataSet = getData.executeQuery();

//如果查询到了相关数据，格式化后发送到手机端,否则发送“null”过去

if (dataSet.next()) {

String temSend = dataSet.getString(2) + "," + dataSet.getTimestamp(3).getTime();

chc.writeAndFlush(Unpooled.buffer().writeBytes(temSend.getBytes()));

System.out.println("send:" + temSend + "\tto:" + address.toString());//向控制台打印发送事件

} else {

chc.writeAndFlush(Unpooled.buffer().writeBytes("null".getBytes()));

}

dataSet.close();

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

chc.close();//关闭连接通道

ReferenceCountUtil.release(msgLin);//释放资源

}

}

//该方法在发生异常时自动调用

public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) {

cause.printStackTrace();//打印异常链

}

}

public static void main(String[] args) {

MyServers myServers = new MyServers(12777);//启动服务，端口号为12777

myServers.run();

}

}

Android应用端程序代码：

import androidx.annotation.NonNull;

import android.app.\*;

import android.content.\*;

import android.os.\*;

import android.view.\*;

import android.widget.\*;

import com.amap.api.services.core.\*;

import com.amap.api.maps.\*;

import com.amap.api.services.geocoder.\*;

import com.amap.api.maps.model.\*;

import java.io.\*;

import java.net.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

public class GpsActivity extends Activity {

private MapView viewMapGd;

private AMap aMapGd;

private String deviceId;

private SharedPreferences shareGd;

private ScheduledExecutorService executorGd;

private String lastGpsMessage;

private Handler handlerGd;

private Runnable runnableGd;

private Marker markerGd;

private ProgressBar waitBar;

private GeocodeSearch searchGd;

private SimpleDateFormat dateFormatGd;

@Override

public void onCreate(Bundle s) {

super.onCreate(s);

setContentView(R.layout.activity\_gps);

viewMapGd = findViewById(R.id.GdMap);//获取地图控件引用

waitBar = findViewById(R.id.waitBar);//获取“更新定位”控件的引用

viewMapGd.onCreate(s); //创建地图

shareGd = getSharedPreferences("AppData", Context.MODE\_PRIVATE);

deviceId = shareGd.getString("DeviceId", "");

//创建用于TCP连接的单线程池

executorGd = Executors.newSingleThreadScheduledExecutor();

//用于日期格式化的对象

dateFormatGd = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss a", Locale.ENGLISH);

//地址解析器

searchGd = new GeocodeSearch(this);

aMapGd = viewMapGd.getMap();

//初始化标记点

markerGd = aMapGd.addMarker(new MarkerOptions().alpha(0.9f).title("查询中..."));

UiSettings settings = aMapGd.getUiSettings();

settings.setZoomControlsEnabled(false);

//初始化地图类型

aMapGd.setMapType(shareGd.getInt("Type", 1));

aMapGd.setOnMarkerClickListener(new AMap.OnMarkerClickListener() {

@Override

public boolean onMarkerClick(Marker m) {

if (m.isInfoWindowShown()) {

m.hideInfoWindow();

} else {

m.showInfoWindow();

}

return true;

}

});

//handler处理器，可通过它安全的改变UI线程里的控件状态

handlerGd = new Handler(new Handler.Callback() {

@Override

public boolean handleMessage(@NonNull Message msg) {

switch (msg.what) {

case 0x00:

toastShow("定位成功");

break;

case 0x01:

toastShow("定位未变");

break;

case 0x02:

toastShow("定位失败");

case 0x03:

waitBar.setVisibility(View.VISIBLE);

break;

case 0x04:

waitBar.setVisibility(View.GONE);

break;

case 0x05:

toAddress((String) msg.obj);

break;

case 0x06:

toastShow("请先设置设备ID编号");

break;

}

return true;

}

});

//用于从服务器获取定位数据

runnableGd = new Runnable() {

@Override

public void run() {

try {

if (deviceId.isEmpty()) {

handlerGd.sendEmptyMessage(0x06);

return;

}

handlerGd.sendEmptyMessage(0x03);//显示进度条

Socket socket = new Socket();//设置服务器地址，端口号

socket.connect(new InetSocketAddress("120.78.203.170", 12777), 5000);

socket.setSoTimeout(5000);

Scanner scanner = new Scanner(socket.getInputStream());

PrintWriter writer = new PrintWriter(socket.getOutputStream());

writer.println("@" + deviceId);//发送请求标志位和设备ID

writer.flush();

StringBuilder builder = new StringBuilder();

//读取来自服务器的数据

while (scanner.hasNextLine()) {

builder.append(scanner.nextLine());

}

scanner.close();//关闭资源

writer.close();

socket.close();

Message message = new Message();

message.what = 0x05;

message.obj = builder.toString();

handlerGd.sendMessage(message);

} catch (Throwable e) {

e.printStackTrace();

handlerGd.sendEmptyMessage(0x02);//发生异常时提示“定位失败”

} finally {

handlerGd.sendEmptyMessage(0x04);//最终隐藏精度条

}

}

};

Button update = findViewById(R.id.update\_gps);

//点击“更新定位”控件时更新数据

update.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

executorGd.schedule(runnableGd, 0, TimeUnit.MILLISECONDS);

}

});

}

@Override

public void onStart() {

super.onStart();

//打开app时自动更新数据

executorGd.schedule(runnableGd, 0, TimeUnit.MILLISECONDS);

}

@Override

public void onResume() {

super.onResume();

viewMapGd.onResume();

}

@Override

public void onPause() {

super.onPause();

viewMapGd.onPause();

}

@Override

public void onDestroy() {

super.onDestroy();

viewMapGd.onDestroy();

}

@Override

public void onSaveInstanceState(Bundle s) {

super.onSaveInstanceState(s);

viewMapGd.onSaveInstanceState(s);

}

@Override

public boolean onCreateOptionsMenu(Menu m) {

getMenuInflater().inflate(R.menu.my\_menu, m);

//获取ToolBar上的控件引用

MenuItem type = m.findItem(R.id.type);

final MenuItem id = m.findItem(R.id.id);

//用于切换地图类型

type.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {

@Override

public boolean onMenuItemClick(MenuItem i) {

int m = aMapGd.getMapType();

if (++m > 5) {

m = 1;

}

aMapGd.setMapType(m);

//将当前的地图类型保存，下次启动时初始化为该类型

shareGd.edit().putInt("Type", aMapGd.getMapType()).apply();

return true;

}

});

id.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {

@Override

public boolean onMenuItemClick(MenuItem i) {

final LayoutInflater laIn = LayoutInflater.from(GpsActivity.this);

View idInputView = laIn.inflate(R.layout.view\_input\_id, null);

final EditText editID = idInputView.findViewById(R.id.devId);

Button cancel = idInputView.findViewById(R.id.cancel);

Button sure = idInputView.findViewById(R.id.sure);

final AlertDialog idInputDialog = new AlertDialog

.Builder(GpsActivity.this)

.setView(idInputView)

.setCancelable(false)

.create();

editID.setText(deviceId);

editID.setSelection(deviceId.length());

cancel.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

idInputDialog.dismiss();

}

});

sure.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String input = editID.getText().toString().trim();

if (input.isEmpty()) {

toastShow("设备ID未设置");

}

deviceId = input;

shareGd.edit().putString("DeviceId", input).apply();

idInputDialog.dismiss();

}

});

idInputDialog.show();

return true;

}

});

return true;

}

//解析从服务器上获取的定位数据,并根据数据更新地图

public void toAddress(final String newGpsMessage) {

try {

if (newGpsMessage.equals("null")) {

toastShow("未发现相关设备");

return;

}

final String[] messageArray = newGpsMessage.split(",");

if (messageArray.length != 16) {

handlerGd.sendEmptyMessage(0x02);

return;

}

//坐标转换器，把gps的坐标格式转成适用于高德地图的坐标格式

CoordinateConverter laLnConvert = new CoordinateConverter(this);

laLnConvert.from(CoordinateConverter.CoordType.GPS);

final LatLng laLnGd = laLnConvert

.coord(new LatLng(convert(messageArray[2]), convert(messageArray[4])))

.convert();

markerGd.setPosition(laLnGd);//设置定位坐标点

//地图中心移向定位点

aMapGd.moveCamera(CameraUpdateFactory.newCameraPosition(new CameraPosition(laLnGd, 14, 0, 0)));

final String time = dateFormatGd.format(new Date(Long.valueOf(messageArray[15])));

markerGd.setTitle("定位时间：" + time);

searchGd.setOnGeocodeSearchListener(new GeocodeSearch.OnGeocodeSearchListener() {

//根据坐标获取地址信息

@Override

public void onRegeocodeSearched(RegeocodeResult re, int i) {

String address = re.getRegeocodeAddress().getFormatAddress();

if (!address.isEmpty()) {

markerGd.setTitle(address + "\n\n" + "定位时间：" + time);

}

if (newGpsMessage.equals(lastGpsMessage)) {

handlerGd.sendEmptyMessage(0x01);

} else {

handlerGd.sendEmptyMessage(0x00);

lastGpsMessage = newGpsMessage;

}

}

@Override

public void onGeocodeSearched(GeocodeResult gr, int i) {

}

});

RegeocodeQuery reQue = new RegeocodeQuery(new LatLonPoint(laLnGd.latitude, laLnGd.longitude), 200, GeocodeSearch.AMAP);

searchGd.getFromLocationAsyn(reQue);//发起异步查询请求

} catch (Throwable e) {

handlerGd.sendEmptyMessage(0x02);

e.printStackTrace();

}

}

//用于坐标转换

private double convert(String xy) {

int index = xy.indexOf('.') - 2;

return Double.valueOf(xy.substring(0, index)) + Double.valueOf(xy.substring(index)) / 60;

}

//用于显示提示

private void toastShow(String msg) {

Toast.makeText(this, msg, Toast.LENGTH\_SHORT).show();

}

}