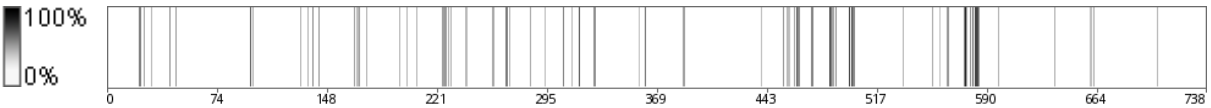
PaperPass旗舰版检测报告
简明打印版

比对结果(相似度):

总　体： 6% (总体相似度是指本地库、互联网的综合对比结果)
本地库： 6% (本地库相似度是指论文与学术期刊、学位论文、会议论文、图书数据库的对比结果)
期刊库： 4% (期刊库相似度是指论文与学术期刊库的比对结果)
学位库： 5% (学位库相似度是指论文与学位论文库的比对结果)
会议库： 1% (会议库相似度是指论文与会议论文库的比对结果)
图书库： 3% (图书库相似度是指论文与图书库的比对结果)
互联网： 1% (互联网相似度是指论文与互联网资源的比对结果)

编　号：5AFEBE51C2FCA3ULC
版　本：旗舰版
标　题：[降重]LoRaWAN节点开发板设计
作　者：曹佳禾
长　度：18510字符(不计空格)
句子数：738句
时　间：2018-5-18 19:51:45
比对库：学术期刊、学位论文、会议论文、书籍数据、互联网资源
查真伪：http://www.paperpass.com/check

句子相似度分布图:



本地库相似资源列表(学术期刊、学位论文、会议论文、书籍数据):

1.相似度： 1% 篇名：《基于FPGA的小型CPU中通信协议的研究及IP Core的开发》
来源：学位论文 山东大学 2009
2.相似度： 1% 篇名：《单片机C语言应用技术》
来源：书籍数据 西安电子科技大学出版社 2012-02-01
3.相似度： 1% 篇名：《单片机接口技术》
来源：书籍数据 西安电子科技大学出版社 2010-09-01

互联网相似资源列表：

1.相似度： 1% 标题：《I2C总线是由数据线SDA和时钟SCL构成的串行...》
http://www.docin.com/p-691965004.html

全文简明报告:

1　引言

物联网Internet　Of　Things(IO-T)是一个非常广泛的概念。　从字面上看，它是相

互连接并访问网络来交换和传递信息的客体和客体。 物联网已经大规模的进入到我们的生活中来了。 例如共享单车就是物联网的一个典型应用。

共享单车的一个简单的开锁过程由三个部分组成： 用户的手机，单车公司的云端以及单车的锁。 单车的锁内部一般有SIM卡来和基站通讯。 用户手机扫码后可以获取单车的编号，然后手机将单车的编号发送给单车公司的云端，云端收到单车编号后下发开锁指令。开锁后单车再想服务器发送开锁成功的信息，再由服务器向手机发送开锁成功的信息。 一些单车智能锁还配有蓝牙模块用于和手机通信，方便在内置SIM卡信号不好的情况下与手机通信，通过手机向云端发送信息。 再由手机接收解锁码等信息来解锁。

从上面的例子可以看出，物联网大致由这些部分组成： 云端，节点，客户端。 目前主流的节点通信方式大多为蓝牙，Wifi，ZigBee等方式。 这些方式能被普遍采用自然由很多优点，比如连接方便，价格便宜，传输速度快等。 但是它们也存在着能耗高。 通讯距离短等问题。 所以我的毕设就研究采用一种相对冷门的物联网技术----LoRaWan技术。该技术主要的优势在于功耗低，传输距离远，可同时连接设备数量多的特点，但是该技术的带宽窄， 所以适用于需要传输的数据不是很大并且电池供电的情况下使用。

本设计本毕业设计是为了响应学校搭建智能网络的号召，作为搭建智能网络系统的马前卒，主要设计一个LoRaWAN节点开发板。 {61%：整个节点开发板主要由三大部分组成：} 一是各种所需的传感器，二是单片机MCU控制系统，三是LoRaWan传输模块。 传感器负责收集各种各样智能网络所需要获取的信息，来代替人工监视实时情况。 {50%：微控制器负责获取传感器捕获的信息并进行初步处理，然后将处理后的信息发送给LoRaWan节点。} LoRaWan节点负责将单片机发送来的信息传送给LoRa网关。 最后实现人在网页上就可以监视各个传感器的信息并且下发相关命令，实现智能网络。

2 概述

2.1 LoRaWAN节点开发板系统以及LoRa技术概述

{44%：该设计主要模拟在真实环境中部署的每个传感器的物联网系统。} 整个系统的以单片机STM32C8T6做为核心控制器。 通过单片机控制整个节点系统，接收数据，处理数据，控制LoRa模块发送数据。 传感器的选取是根据实际的需要，并且能够再实际环境中部署的传感器。 本设计选取了： 1.温湿度传感器（型号： DHT11）2.烟雾传感器（型号：MQ-2）3.光照强度传感器（型号： GY-30）4.红外传感器（HC-SR501）。 其他主要部分：OLED屏，全彩RGBLED，按键，LED灯，有源蜂鸣器。 LoRa传输模块选择的是 M100 D-T模块，该模块使用方便，配置好要配对的网关参数后即可自动寻找配对， {57%：单片机发送数据的方式是串行通信。} 且一次不能超过8个字节。 如果把单片机比作系统的大脑，那么传感器就是你的眼耳鼻，用来收集各种信息， LoRa模块就是你的嘴巴，用来发送你收集并处理好的信息。

{53%：LoRa技术是一种用于未授权频谱的低功率广域网技术。} 这种技术具有网络搭建的快速灵活，技术种类丰富，商品化方便等特点。 LoRa芯片是由semtech公司在2013年8月首先提出的。 LoRa技术是面向物联网技术提出的，与同期的启发业界顶级芯片相比，LoRa芯片的接收信息的灵敏度达到了-148 dB， 提高了将近20 dB，使得信息的接收更加灵敏与顺畅，也能拥有更大的通信范围同时有助于网络效率的提高以及故障率的降低。

LoRa采用的是线性扩频调制技术，它的链路预算能够达到157 dB，这使得 LoRa的理论通信距离可以达到15 km以上， 这使得城市级网关有了实现的可能。 同样，LoRa节点的通信距离也远远大于其他类型的物联网芯片。 LoRa的另外一个特点是采用数据速率自

适应的策略，这能够最大程度的优化每一个节点的数据发送接收的速率，节点的能耗，输出功率等。 这使得LoRa芯片在接收发送信息时的电流可以低至10mA，休眠时的电流甚至可以低至200nA。 这样的工号完全可以支持电池供电，为系统的设计和使用提供了方便，加速LoRaWan网络的部署。

2.2 本设计方案思路

本设计的核心就是一块以单片机为核心的，利用传感器采集数据并通过LoRa模块发送的系统，系统框图如图2-1所示。

图2-1 开发板系统框图

首先调查需求，首先是最基本也是最重要的火灾预警，所以选择了烟雾传感器（型号：MQ-2）。 然后是对气候监测的需求，温度情况，湿度情况，以及是否下雨等。 所以我综合以上考虑，选取了温湿度传感器以及光强传感器，可以综合湿度以及光强判断晴天，阴天，下雨等情况。 温湿度传感器（型号： DHT11），光强传感器（型号： GY-30）。 最后是监测是否有人经过，来控制灯的亮灭，所以我选取了红外传感器（型号： HC-SR501）。 选择传感器后，选择另一个关键模块 - LoRa模块。 LoRa模块我和老师商量之后最后决定选取 M100 D- T模块，该模块使用方便， 配置好要配对的网关参数后即可自动寻找配对，单片机发送数据的方式是串口通信。 且一次不能超过8个字节。

在安排好这些模块和单片机连接的引脚后，我发现还多出来了不少引脚没有被用到，本着不能浪费的原则，我选择给这些空闲引脚加上别的功能，技能丰富开发板功能， 也能让开发板不仅只是‘ LoRa’开发板。 我添加的器件有有源蜂鸣器，按键，RGB全彩LED，单色LED。

程序的下载方面，除了单片机自带的 st- link下载口之外，为了方便演示需要以及哪些没有 Keil的用户， 我还设计了串口下载的方式，利用 ch340 e芯片和 micro USB来实现这个功能。

开发板程序设计方面，由于我制作的开发板，所以暂时只打算提供各个模块以及LoRa的例程。 程序部分设计思路下文会提及。

2.3 研发方向和技术关键

（1）传感器传回信息的读取和分析；

（2）LoRa模块和网管的连接与通信；

（3）单片机和LoRa模块的通信以及数据的发送；

（4）传感器的选择以及性能。

2.4 主要实现的功能

（1）单片机的正常工作

（2）传感器数据的正常读取

（3）LoRa模块能发送信息给网关

（4）能在云端看到开发板发送的信息

3 总体设计

电路板由四个主要部分组成： 1.供电系统。 电源是整个电路的发动机，是最基础的部分，纯净，稳定的电源能够让整个电路的运行更加安全，持久。 2.单片机最小系统。{72%：单片机的最小系统是可以让单片机工作的基本条件。} 包括晶振，复位等部分。{48%：3.各种传感器，传感器负责收集各种信息。} 4.LoRa模块，负责发送数据。

3.1 供电系统

有两种方式为整个电路板供电，一种是通过 USB，另一种是通过 st- link下载程序，通过电脑的 USB口和板子相连， 此时板子的供电则会通过 st- link口的VCC和 GND来提供。

根据两种不同的情况，电源的稳压及滤波显的尤其重要，要兼顾两种不同的供电方式。一般来说，首要保证由USB输入的电压的稳定。 由于电脑的 USB口的电压是5 V，所以我在 USB电压输入端设计了电容来滤波， 然后滤波的电源连接到 LM1117进行调节，输出3.3 V. LM1117最大供电电流可以达到1.5 A左右， 而板子上的各器件最大电流分别为： 1.光强传感器： 最大电流7mA。 2.温湿度传感器模块： 最大电流2.5mA。3.烟雾传感器： 最大电流： 180mA。 4.红外传感器： 最大电流忽略不计。 4.液晶屏： 最大电流25mA。 5.RGB全彩LED： 5.单片机STM32C8T6： 最大电流100mA左右。总计375mA。 所以LM1117完全可以达到要求。 并且器件也不会一直工作在满负荷状态，所以LM1117是足够正常使用的而且也不会过热。 经过稳压后还需要3.3V电压的滤波，所以在3.3V输出的附近也需要加上贴片电容进行滤波。 由于整个供电回路较长，基本上环绕了整个板子，所以在路线末端可能会出现电压衰减的情况， 所有我在每个用电的器件的供电处根据消耗电流的大小都设置了不同容值的电容来保证各个器件供电电压的稳定以使之稳定工作。

整个开发板的地我也进行了隔离。 因为开发板总电流并不大，正常工作可能也就2，300mA，所以我并没有进行模拟地和数字地的隔离。 {46%：但是，烟雾传感器输出一个电压信号，需要单片机的ADC来采集它。} ADC是一个对电压变化很敏感的部分，所以我对ADC的地和开发板其他部分的地进行了隔离。 两种地的隔离方法是单点接地，然后用0欧姆电阻进行连接。 这样能最大限度的保持单片机ADC地线的纯净，从而使测得的电压更加准确。

3.2 单片机最小系统

{46%：单片机最小系统的单片机工作的基本保证，只有最小系统中工作的单片机才能稳定工作。} STM32C8T6单片机最小系统包括四个部分： 1.晶振电路。 {58%：晶体振荡器电路包括晶体振荡器和启动电容器。} 在我设计的开发板上，我设计的是8MhZ的晶振以及22pF的起振电容接地。 这是最基础的晶振电路，非常容易起振。 2.复位电路。{55%：复位电路的核心是将RESET引脚拉低以复位微控制器。} 为了防止电压跳变过快，引发错误，所以并联一个电容，使电压不会跳变。 3.电源。 已经详细描述了电源部分3.1，并且将不再描述次数。 4.程序下载口。 我设计了两个程序下载的方式。 一是通过单片机的PA13： SWDIO口和PA14： SWCLK口进行下载。 所以需要拉出四个脚VCC，SWDIO，SWCLK，GND来通过ST-Link进行程序下载。 SWDIO口需要一个10k的电阻进行上拉。 另一种下载的方式是通过串口进行下载，我用单片机的 PA9和 PA10作为 TX和 RX然后加上串口转 USB芯片 CH340 e来和板子的 micro USB进行连接。 这样就使得单片机

也可以直接同过串口下载HEX文件。

3.3 各种传感器

传感器是开发板的重头戏，所有的信息和数据都是通过各种各样的传感器接收而来。并且传感器怎么和单片机相连也是一个难点。 既要考虑单片引脚是否能接收传感器的信息，也要考虑到绘制PCB时的布线方便。

1. 光强传感器GY-30： 光强传感器与单片机通信的方式是IIC通讯，IIC具有所需导线少，传输速度快等优势。 我将传感器的SCL，SDA分别连接到单片机的PB11和PB12。

2. 温湿度传感器DHT11： 温湿度传感器和单片机通讯的方式是通过一个DATA脚，根据一定的时序发送高低电平的信号来向单片机传递信息。 DATA脚需要上拉。 {50%：所以我设计将DATA引脚连接到微控制器的PB5引脚。}

3. 红外传感器HC-SR501： {43%：红外传感器与单片机之间的通信相对简单，传感器和无人传感器的信号引脚分别输出高低电平。} {53%：所以我将红外数据引脚连接到微控制器的PA15引脚。} 通过判断销是高还是低，判断它是否被人通过。

4. 烟雾传感器MQ-2： 烟雾传感器的输出由两种方式，一种是和温湿度传感器一样的TTL输出，另一种方式是输出一个电压，电压越大说明烟雾浓度越高。 但是该模块的输入电压是5V，所以理论上当烟雾浓度很大时，输出的电压可能会接近甚至达到5V。 {46%：微控制器ADC的最大电压只能接收3.3 V，这可能会损坏微控制器。} 所以在这个地方我对传感器的输出电压进行了分压处理，以保护单片机 ADC。 我设计将分压后的电压输入到单片机的PA1脚。 这个引脚可以复用为ADC。

5. 液晶屏OLED12864： 液晶屏和单片机通信的方式是SPI。 我设计PB13，PB14，PB15和PA8分别连接液晶屏的SPI_SCK，SPI_DC，SPI_MOSI和SPI_RES。

3.4 LoRa模块

LoRa模块使用的是M100D-T。 该模块内部有个单片机，内置了程序，也可以自行下载程序，所以将其下载口拉出。 该模块还有两个串口，一个是连着LoRa芯片，用来配置LoRa芯片的各个参数的。 另外一个是与LoRa模块的单片机相连。 单片机内部内置了一些程序使得你只需要按照他的规则利用单片机通过这个串口给模块内的单片机发送数据， LoRa模块就会将这些数据通过 LoRa发送给已经配对上的网关等设备。

3.5 其他说明

系统分为硬件部分和软件部分。 硬件部分包括电路设计以及传感器选取等。 软件部分包括各个模块例程以及LoRa芯片的配置（此部分属于怎么使用别人公司的特例产品，故在本论文中不作详细介绍）等

4 硬件设计

4.1 供电电路

上面详细描述了整个电路板的功率要求。 由于有单片机这种对电源比较敏感的器件，加上整块电路板的电流加起来不过3，400 mA， {45%：所以在开关电源和线性电压电源方面，我选择线性稳压电源芯片 LM1117作为供电装置。}

供电电路的另一个要点就是保持整条供电线电压的恒定，所以需要增加储能元件电容来保证电压不会突变。 还有就是地线的设计。 这些都在上文已有提及。 此处不再赘述。{45%：本节主要介绍具体硬件电路的设计和所用芯片的原理。}

4.1.1 集成稳压芯片（LM1117）

LM1117是集成线性稳压芯片。 相比于开关电源的电压变换，线性稳压得到的电压更加的稳定，几乎不会有纹波的存在。 这样能够使得整个系统的供电更加纯净，稳定，使得各个器件得以正常的工作。

线性稳压的核心器件是个MOS管或者三极管。 LM1117内部使用的MOS。 {45%：线性稳压原理是使芯片内部的MOS管工作在放大区域，相当于一个电阻。} 让多余的电压消耗在这个电阻上。 这个电阻的阻值大小是可以通过Vgs进行控制的，所以在输出端有一个反馈电路，保证消耗在管子上的电压大小随着负载的变化而相应变化。 从而使得输出的电压爆出一个稳定值。 由原理可见，线性稳压的输出不会像开关电源一样有无法消除的纹波存在。但同时，多余的能量直接消耗在管子上，所以这种稳压方式的电流不能太大，不然会使得芯片过热从而损坏芯片。 但在本设计中，1.5A的电流已经足够使用，所以选择线性稳压进行供电。

LM1117的基本原理示意图如图4-1所示。

图4-1

我这里以三极管为例来说明线性稳压芯片的内部结构。 整个结构由四部分组成：1.R2和D7组成基准电压电路。 D7是齐纳二极管，工作在反向击穿状态时，D7上面的电压时一个固定的值，用来作为基准电压的参考。 2.采样电路： R3和R4形成一个分压输出电压的采样电路。 3.比较电路： 比较电路采用了一个运放。 {52%：运算放大器正端的参考电压连接，运放的负端负电压输出分压。} {52%：运算放大器然后放大这两个电压之间的差异。} {54%：运算放大器的输出连接到晶体管的基极。} 4.三极管。 {47%：晶体管是该电路的核心器件，也是负反馈控制器件。} 之所以这个电路能够实现降压稳压的效果，核心就在于这个三极管工作在放大区，当成一个电阻， {47%：这个电阻的阻值是可变的，并且会随着运算放大器的输出电压而变化。} 晶体管电阻然后与负载串联连接，从而过量的电压在晶体管中耗散。 这是线性稳压没有纹波的原因，同时也是不能通过大电流的原因。

4.1.2 开发板供电电路图

图4-2 供电部分电路（1）

图4-3 供电部分电路（2）

供电部分电路图如图4-2，4-3所示。 图4-2是5V转3.3V电路。 5V和3.3V处都分别加了滤波电容，容值从大到小，为了更好的滤除各个频率的杂波。 图4-3左侧电路时单片机的单独滤波。 {44%：STM32C8T6单片机有3个电源端口，目的是使单片机工作更加稳定。} 即使有一路两路的供电出了问题也能继续工作。 所以在这3路供电每一路都要加上滤波电容。图4-3右侧是单片机ADC的供电以及地线隔离。 ADC的地和其它的地通过R27这个0欧姆电阻通过单点接地的方式连接在一起。

4.1.3 供电电路PCB设计思路

电源电路输入，微型USB是喇叭口，所以它必须放在电路板的边缘。 电容一定要按照电

流的方向根据容值由大到小放置。 有些传感器的供电是5V，有些则是3.3V，这也是设计PCB是需要考虑的。 每个器件的储能电容的放置最好靠近器件的VCC，这样能达到最好的效果。

LM1117是整块板子发热最厉害的器件，所以要对 LM1117的3.3 V处进行铺铜， 为了达到最好的效果，我在正反面均进行了铺铜，用过孔连接。

由于供电的电路路线较长，为避免产生太长的环路，以及为了获得稳定的地，本开发板采用了铺地的设计。 电源线上的电流可以快速返回到最近的地线。 我在板子的正面和背面都大面积铺铜。 通过大量的过孔连接。 但是需要注意，LoRa模块下面不能铺铜，不然会对信号的发射产生影响。

4.2 单片机系统

{49%：单片机最小系统电路设计的第一点是便于每个设备的连接。} 因为此开发板器件较多，连线比较复杂，而电路板由于能力及经费有限，最多只能做双层板。 所以一旦出现传感器和单片机的连线出现太多的交叉绕弯等情况，会导致绘制PCB的极大不便。 同时也需要考虑单片机的哪些引脚能与对应器件相连。 例如液晶屏是SPI通信，所以原则上需要和单片机的引脚中能用来进行SPI通信的引脚相连。 但是本设计中的IIC通信和SPI通信均采用软件模拟，所以这个问题不是什么大麻烦。 软件模拟具体怎么实现会在软件设计篇详细讲述。

单片机最小系统中的晶振需要离单片机越近越好，并且晶振的两个脚到单片机两个脚的电路距离要一样。 STM32C8T6可以有两个晶振，本设计由于用不到，所以选择单晶振设计。{63%：微控制器的最小系统电路图如图4-4所示。}

图4-4

{46%：微控制器复位时微控制器BOOT0和BOOT1的电平状态决定了微控制器在复位后开始从该区域执行程序。} 当BOOT0为0时，无论BOOT1是高还是低，程序都从用户闪存启动，即正常工作模式。 当BOOT0为1且BOOT1为0时，它从系统存储器开始。 当BBOT0和BOOT1均为1时，从内置SRAM开始，此模式可用于调试。 一般来说我们不太会采用从内置SRAM启动。因为SRAM掉电后数据就会丢失。

根据 BOOT0和 BOOT1的功能可以看到，大多数情况下 BOOT0和 BOOT1都应该是0，但是考虑到这是开发板，所以我设计成 BOOT0和 BOOT1默认接地， 但是留出了排针给使用者自行决定是否接高电平。 电路图如图4-5所示。

图4-5

4.3 传感器及其他外设电路设计

每个传感器及外设的通信方式不同，需要的单片机支持的引脚也不相同。 这些在我们硬件电路的设计中是需要考虑的头等大事。 {47%：在第一个设计中，我将IIC器件连接到单片机的硬件IIC引脚，SPI器件连接到单片机的硬件SPI引脚。} 但是这不仅给电路图的绘制带来了困难，整个板子跳线跳的乱七八糟， 更让人糟心的是写程序的时候才知道STM32单片机的一个坑。 ST公司是没有IIC的版权的，所以STM32单片机的IIC脚是伪IIC。就是说硬件IIC在STM32上是行不通的。 最后还是采用了软件模拟的方法才得以成功。 后来第二板的时候，我决定都是用软件模拟的方式。 这使得电路的设计，器件的放置顺畅了很多。 不会出现很多跳线的情况。

其它外设方面，RGBLED也给我带来了困难。 {43%：据称，RGB LED芯片是将三个LED

灯珠封装在一起，使用相同的电源。} 我第一次设计时认为这三个灯都是一样的，所以只在电源输入端加了一个限流电阻。 后来实验的时候发现蓝灯和绿灯都是正常的，但是只要程序设定红色的灯亮，其他颜色的灯就不会亮。 这个问题困扰了我很久，排查了很多问题。最后才知道，原来三个灯珠的参数并不一样，由于红色的灯光波长段，如果在输入端只有一个限流电阻， 相应 LED的等效电阻很小，只要红灯亮，所有电流都从红色 LED流出，而不是通过另外两个二极管。 所以只会有红灯亮。 正确的接发应该是在每个发光二极管的负极连接一个限流电阻，这样就能保证每个管子都有一样的电压，每个管子都能发光。 其他的外设比较简单，没有带来什么问题。

4.3.1 烟雾传感器电路图

图4-6

图4-6是烟雾传感器的电路图。 {59%：传感器的输出有两种输出方式：} DO和AO。DO是指数据以高低电平代表1和0的方式，以一定的编码进行数据的传输。 即数字传输。AO代表模拟传输，烟雾浓度由电压值的变化表示。 但是烟雾传感器的 VCC是5 V，但是单片机的 ADC脚并不是5 V兼容的，但是理论上在烟雾浓度大的时候， {44%: AO引脚的电压将超过3.3 V，这可能会损坏单片机的内部 ADC。} 以保证 ADC能够正常安全的工作。 单片机方面，选择PA1作为ADC口。

4.3.2 温湿度传感器电路图

图4-7

{74%：温湿度传感器电路如图4-7所示。} 传感器通过DATA脚以一定的规则发送信息给单片机，座椅对单片机的引脚没有特别的要求。 所以选择PB5作为IO口来接收信息。

4.3.3 红外传感器电路图

图4-8

图4-8是红外传感器的电路图。 红外传感器的信息传输也非常简单，传感器有一个输出口，输出口的高低电平表达了是否有人经过。 单片机方面我选择PA15脚作为IO口来监测传感器信号。

4.3.4 光强传感器电路图

图4-9

{57%：图4-9是光强度传感器的电路原理图。} 用于在多个设备安装在一个IIC总线上时选择与主机通信的设备。 由于本设计中这一条IIC总线上只有这一个设备，所以将ADDR脚接地即可。 VCC和GND就是正常的3.3V供电。 IIC的SCL和SDA需要用电阻上拉，我选则的是R19和R20两个5.1K欧姆的电阻进行这项工作。 在单片机的引脚选择上，如上文所述，STM32的硬件 IIC是伪 IIC， 无法使用，所以我选择单片机的 PB11和 PB12脚来软件模拟 IIC来读取数据。

4.3.5 OLED12864液晶显示屏电路图

图4-10

图4-10是OLED12864液晶显示屏的电路图。 SPI通信需要四根通信线： 1.SPI_SCK是

为SPI通信提供时钟信号的脚。 2.SPI_MOSI是信息传输线。 信号通过这条线传输。 3.SPI_RES这根线的作用是复位。 微控制器可以通过该线路的高电平和低电平来重置LCD屏幕。 4. SPI_DC，SPI总线和IIC总线可以将许多设备安装在同一总线上。 SPI_ DC就是用来选择当时和主机通信的是哪一个期间按，避免产生混乱。

液晶屏这个器件当时我在选型的时候有两种可选，一种是SPI的，另一种的IIC的。 为什么有只需要两根线的IIC不用却选择一个SPI呢？ 我是考虑到这个节点同时也是单片机开发板，而开发版的作用之一就是让使用者禁可能多的用到单片机的不同功能。 既然光强传感器和单片机的通信方式已经是IIC了，所以这里的液晶屏我就选择了SPI。 这样也能“迫使”开发学习者区了解这两者的区别与联系。

4.3.6 有源蜂鸣器电路图

蜂鸣器有有源蜂鸣器和无缘蜂鸣器之分。 蜂鸣器并不是给一个电压它就会响，而是需要一个一定频率的PWM波，这样，它才会根据PWM波的频率发出声音。 所以有源无源的源指的是振动源而不是电源。 有源蜂鸣器有自己的内部振动源，所以不需要提供PWM。 被动蜂鸣器可根据程序设置发出不同的频率。

{41%：蜂鸣器驱动电路需要单独设计，不能简单地连接到微控制器的引脚。} 因为单片机的引脚能通过的电流是有限的并且很小，达不到蜂鸣器的电流，所以需要驱动电路来带动蜂鸣器。 我选择用三极管来驱动蜂鸣器。 三极管能放大基极电流，所以只要将单片机的引脚连接到三极管的基极，就可以弥补单片机引脚通过电流比较小的问题， {66%：使用三极管的发射极和集电极驱动蜂鸣器。} 晶体管的基极连接到MCU的PB9引脚。 R26既能使单片机引脚在浮空时不让蜂鸣器误开启，也能在在单片机输出为1时接收大部分电压时三极管开启。整个电路图如图4-11所示。

图4-11

4.3.7 RGBLED电路图

RGBLED芯片的最关键的问题就是需要使用三个分开的限流电阻，而不能只使用一个。不然会只亮红灯而其他几个灯不会亮。 RGBLED电路图如图4-12所示。

图4-12

GREEN引脚连接到MCU的PB10，RED引脚连接到MCU的PB1，BLUE引脚连接到MCU的PB4。

4.3.8 按键电路图

微控制器监控按键的原理是查看连接按键的引脚的高低电平是否改变。 我在按钮上设计了拉力阻力，按钮未按下时默认为高，按下时为默认值。 按键电路图如图4-13所示。

图4-13

4.4LoRa模块

LoRa模块是LoRaWan节点的核心器件之一，有了它才能远距离地向LoRa

网关发送信息LoRa模块一共有12个引脚。 模块内还有一个单片机，负责和外部单片机通信，并且根据LoRa芯片协议将外部发送进来的数据通过LoRa模块发送出去。 这使得LoRa模块的使用变得容易许多。 由于LoRa模块的通信方式是串口，我们只需要使用外部单片机

向LoRa模块内的单片机通过串口发送一定规则的数据即可。 该模块将单片机的下载口，直接和LoRa芯片通讯的串口以及和模块内单片机通讯的串口全部拉出。 个人能力有限，所以只能使用模块的基本功能。 至于向模块内单片机下载程序等工作就交给更有能力的使用者来完成。 因此我将所有接口全部拉出，酌情使用。 {51%：模块中的模块BOOT脚处理方法与上述相同，在此不再赘述。} LoRa模块电路图如图4-14所示。

图4-14

4.5串口电路

此处串口电路指的是和电脑相连的串口，即USB转串口。 这是为了方便没有KEIL的用户直接通过HEX文件进行程序的下载。 我采用的串口芯片是CH340e芯片。 该芯片管脚少，使用方便。 比美信的串口芯片使用起来方便很多。 电脑通过 MICRO USB传输进来的信号是通过两个脚进来的，分别为 D+和 D-， CH340 E就是将 D+， D-的信号转换成 TX， RX，方便单片机的接收处理。 给单片机下载程序的串口的脚是确定的，即PA9和PA10。 只有连在这两个脚上才能正确下载程序。 CH340E芯片还有一个TNOW脚，这个脚的作用是起指示作用。 将这个脚连一个发光二极管到地，每当有信息在转换的时候发黄二极管就会发出提示光，说明芯片正在工作，电脑有信息正在发送而来。 串口电路如图4-15所示。

图4-15

硬件电路图如图4-16所示。

图4-16

5 软件设计

5.1 总体方案

软件设计方面，本毕业设计主要设计了各个模块的例程，以方便使用者快速上手。 我已经制作好了一整套的HARDWARE.c文件，使得使用可以简单明了的了解到程序如何工作，如何在MAIN函数中控制各个模块而不用管具体的底层接口。 例如，如果要控制LED灯的开和关，只需在MAIN功能中写入LED1 = ON即可。 这样会大大方便开发者。

5.2 各模块及单片机外设资源例程设计

各个模块的例程只是说明了各个模块的基本功能，主要工作是将模块的底层接口封装起来，使得使用在在编写程序时可以 将注意力放在程序本身要实现的功能逻辑上而无须在意具体的接口。

5.2.1 OLED液晶屏例程设计

首先介绍一下OLED模块初始化的过程， OLED模块的控制器是SSD1306。

图5-1

图5-1是SSD1306初始化的框图。

如果你直接将OLED显示屏接上电源屏幕是不会亮的。 这并不是说明屏幕坏了，而是初始化的时候会清空缓存，所以显示器默认显示的就是空，即没有内容。 我们需要另外的函

数来对液晶屏进行控制。

OLED的显存是一个128x8的空间，对应屏幕的像素点，显存空间的0，1代表了屏幕上像素点的亮灭。 所以程序控制的原理就是首先在单片机内存中开辟一个128 x8的内存空间，然后在单片机内存中先修改， 直到内存中的数据是自己想要显示的内容然后再一次性更新到 OLED中进行显示。 下面介绍几个主要的函数及其作用。

（1）OLED_INIT： 此功能的功能是初始化与OLED模块通信的相关引脚。 单片机与OLED显示屏的通信方式是SPI，我采用的是软件模拟的方式，所以再INIT函数中将相关引脚设置为推挽输出模式，速度为50MHz。 然后初始化片选脚为1，RESET脚为0。 即SPI总线一直与显示屏进行通讯（因为SPI总线上只有这一个器件）。 INIT函数里还带有初始显示字样的程序，这些就需要用到其他函数了。

（2）OLED_DrawPoint： 这个函数是其他所有显示函数的基础，这个函数直接修改单片机与OLED显存对应的内存，修改特定点的数值。 OLED_ Display_ On（开启 OLED显示）， OLED_ ShowChar（显示字符）， OLED_ ShowNum（显示数字）， OLED_ ShowString（显示字符串）等函数都需要用到 OLED_ DrawPoint来实现功能。

（3）OLED_Refresh_Gram： 这个函数的作用很简单也很重要，就是将内存中的数据推送给液晶屏，实现显示的效果。 每次执行这个函数相当于刷新一次屏幕。

{41%：使用上述功能，您可以使用液晶显示屏来显示所需的内容。} 程序流程是首先OLED_INIT（），然后使用SHOW函数写入想显示的内容，最后REFRESH一下就可以达到想要的效果。 主程序关键代码如下：

```
OLED_INIT（）；

OLED_ShowString(0，24， ”0.96’ OLED TEST”，16)；

OLED_ShowString(0，40，”ATOM 2018/3/7”，12)；

OLED_ShowString(0，52，”ASCII: ”，12)；

OLED_ShowString(64，52，”CODE: ”，12)；

OLED_Refresh_Gram()； //更新显示到OLED
```

最终显示效果图如图5-2所示。

图5-2

5.2.2 红外模块例程

{53%：该模块基于红外传感器自动控制模块，使用的探头是德国的LHI778。} 这个模块在探头上套了一个散光罩，用来探测更大范围内的红外信号。 {45%：该模块有两个可调旋钮，可分别调节传感器的感应距离和角度范围。} {43%：一旦有人通过红外模块的感应范围，模块的输出引脚变高，如果没有人为低。} {42%：我们可以连接单片机的脚和模块的输出引脚，判断人是否通过判断引脚是高电平还是低电平。}

这个模块有两种触发方式可以设置： 1. 不可重复触发方式: {49%：也就是说，在检测输出为高电平后，一旦延迟时间结束，输出将自动从高电平变为低电平。} 2. 可重复

触发方式： {72%：传感器输出高电平后，在延迟期间，如果人体在其感应范围内有效，其输出将保持高电平。} {58%：在人离开之前，它延迟将高电平转换为低电平（感测模块在检测到人体的每个活动之后将自动推迟延迟时间，} 最后一次活动的时间是延迟时间的起点）。本设计选择的是可重复触发模式。 下面介绍红外模块例程几个主要函数的作用：

（1）INFRARED_Init： 这个函数是初始化函数，即初始化与模块输出脚连接的单片机引脚PA15。 这个函数非常简单。

（2）INFRARED_Scan： 这个函数是引脚电平监视函数，可以放于主循环，也可放于终端服务函数中。 {60%：建议将其放入定时器中断服务功能中。} 因为这样的话可以实现每隔一段时间自动监测一次是否有人而不需要考虑主循环的时长，使得程序简单易维护。这个函数内部还自带了OLED显存更新程序。 使用时只需要在使用此函数之后再加一行OLED_Refresh_Gram()命令即可。

有了这两个函数，主函数就非常简单了，关键代码如下：

```
OLED_Init();

INFRARED_Init();

INFRARED_Scan();

OLED_Refresh_Gram();
```

最终显示效果如图5-3所示：

图5-3

5.2.3 温湿度传感器例程

{82%：温湿度传感器DHT11是一款带有校准数字信号输出的温湿度复合传感器。} {63%：其内部传感器是电阻式湿度传感器和NTC温度传感器。} {50%：该传感器与DATA引脚中的微控制器进行通信。} 一次通信的时间大约在4ms左右。 {57%：传感器发送的数据分为整数和小数两部分。} 在我的设计中，我将小数部分进行了屏蔽，为保诚程序的简洁，小数部分可以给用户自行扩展发挥。 具体数据格式如下：

一次完整的数据传输为40bit，高位先来。

数据格式： 8bit湿度整数数据+8bit湿度小数数据

+8bi温度整数数据+8bit温度小数数据

+8bit校验和

数据正确传输时，校验和数据相同\\ {78%：“ 8位湿度整数数据+ 8位湿度十进制数据+ 8bi温度整数数据+ 8位温度十进制数据\\ ”} 所得结果的末8位。

{78%：用户MCU发出启动信号后，DHT11从低功耗模式切换到高速模式，等待主机启动信号结束。} {59%：DHT11发送响应信号，发送40位数据，并触发信号采集，用户可以选择读取部分数据。} DHT11接收到启动信号以触发温度和湿度采集。 通讯时序图如图5-4所示。

图5-4

主要程序函数说明：

（1）dht11_init: 没什么可说的，和之前的初始化函数一样，用来配置单片机引脚工作在推挽输出模式。

（2）dht11_reset: 复位函数。 初试化温湿度传感器功能时，除了需要初始化单片机引脚外，最好也能重置一下模块。 需要注意的一点是，这个模块并非插上电就能准确的读出数值，而实需要一定时间的预热， 所以在程序的主循环之前，预留一定的时候给模块进行预热。

（3）dht11_check: 检查函数，这个函数是用来检查模块是否正常工作的，函数返回值为0时说明模块正常工作，1表示不正常。

（4）dht11_read_bit: 这个函数时模块使用的核心函数。 这个函数配合dht11_read_data一起使用。 read_data用到的就是read_bit函数。 通过这个函数我们会把这个值放在一个变量中，然后在主程序中进行转换，显示。 主函数关键代码如下：

```
SystemInit();

delay_init();    //延时函数初始化

OLED_Init();

dht11_test();
```

dht11_test()函数是我根据前面几个常用函数组合起来的函数，我将它们封装为一个函数，同时也加入了OLED显示的代码。 使得程序更加简洁。 最终效果如图5-5所示。

图5-5

5.2.4 烟雾传感器例程

在这个例程中，我们将使用 STM32 F103 C8 T6的 ADC1通道1来采样 MQ-2的电压（烟雾越厚， 电压越高），并在 OLED模块上显示出来。 首先说明一下本设计使用的单片机STM32C8T6的ADC系统。 {45%：微控制器有两个ADC，每个ADC有18个通道，分为16个外部通道和2个内部通道。} 它可以测量16个外部来源和2个内部来源。 两个ADC可以独立使用或以双模式使用（这可以提高采样率，从而提高精度）。 ADC的采样阈值时3.3V，精度为16位。 单片机也可以使用外部ADC，本设计采用的时内部自带的ADC。 下面是使用单片机内部ADC的程序步骤。

1.开启PA口和ADC1的时钟，设置PA1为模拟输入。 STM32F103C8T6的ADC通道1在PA1上，因此我们需要首先使能PORTA时钟和ADC1时钟，然后将PA1设置为模拟输入。 使能 GPIOA 和 ADC 时钟用 RCC_APB2PeriphClockCmd 函数，设置 PA1 的输入方式，使用 GPIO_Init 函数即可。

2. 复位 ADC1，同时设置 ADC1 分频因子。 打开ADC1时钟后，我们需要复位ADC1。 分频因子是确保ADC1时钟（ADCCLK）不超过14 MHz。 这个我们设置分频因子位6，时钟为 72/6=12MHz。

3.初始化ADC1参数，设置ADC1的工作模式以及有关规则顺序的信息。 设置分频因子后，我们可以启动ADC1模式配置。 同时，我们还需要设置ADC1规则序列的相关信息。

4.使能ADC并校准。

5.读取ADC的数值。

主要函数介绍：

（1）ADC_Init： {48%：初始化函数，该函数的作用是根据上述步骤初始化ADC。}次设计中，我们只用到了ADC1，所以就开启了ADC1的通道1。

（2）Get_Adc： 该功能的功能是获取主ADC的值。

（3）Get_Adc_Average： {41%：此功能用于多次获取ADC值，平均值以提高准确度。}主函数关键代码如下：

```
Adc_Init();

NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);  //设置NVIC中断分组2:
```

{53%：2位抢占优先权，2位响应优先权}

```
adcx=Get_Adc_Average(ADC_Channel_1，10);

temp=(float)adcx*(3.3/4096);  //换算成实际给用户看到的数值
```

最终运行图如图5-6所示。

图5-6

可以看到当前环境下 ADC值很小（最大为4096）因为我室内环境下进行测试的，因此，烟雾传感器不检测烟雾，所以输出电压也非常低。

5.2.5 光强传感器例程

在本次例程中，我们使用 STM32上的 PB11和 PB12来模拟 IIC总线，来实现和BH1750之间的双向通信， 读取 BH1750光感传感器采集的光照强度值，并且通过通用定时器 TIM3产生中断来不断的更新所采集的数据， 在 OLED上实现显示实时的光照强度值。

{61%：IIC总线是PHILIPS公司开发的用于连接微控制器及其外围设备的双线串行总线。}{94%：它是由数据线SDA和时钟SCL组成的串行总线，可以发送和接收数据。} {47%：在CPU和受控IC之间以及IC和IC之间，执行双向传输，并且高速IIC总线通常达到400kbps或更高。}IIC总线在传递信息时的信号可以分为三种： {54%：1.启动信号，即SCL线保持高电平，SDA线从高电平转换为低电平，数据传输开始。} 2. 结束信号： {69%：当SCL为高电平时，SDA从低电平转换为高电平，结束数据传输。} 3. 应答信号： {80%：接收数据IC在接收到8位数据后，向发送数据的IC发出特定的低电平脉冲，以指示数据已被接收。} {77%：CPU向受控单元发送信号后，等待受控单元发出响应信号。} {55%：如果没有收到响应信号，则确定受控单元发生故障。} 这只是写入信号中的起始信号所必需的，其他信号可能是不必要的。 IIC总线时序图如图5-7所示。

图5-7

主要函数介绍：

（1）IIC_Init：　该功能初始化PB11和PB12引脚，启用时钟，并设置为推挽模式。

（2）IIC_Start：　这个函数的作用是产生IIC的开启信号。

（3）IIC_Stop：　这个函数的作用与上一个相反，它的作用是产生一个IIC停止信号。

（4）IIC_Wait_Ack：　这个函数的作用等待应答信号的到来。　{46%：函数返回值1表示接收应答失败，返回值0表示接收应答成功。}

（5）IIC_Send_Byte：　这个函数的作用是通过IIC协议发送一个字节的信号。　返回值为0说明从机无应答，返回值为1说明从机有应答。

（6）IIC_Read_Byte：　这个函数的作用和上一个函数正好相反，放一个函数时发送一个字节，这个函数时接收一个字节。　这也体现了IIC总线的双向性。

主要代码如下：

```
IIC_Init();

IIC_Start();　//起始信号

IIC_Send_Byte(0x46);　//发送设备地址+写信号

while(IIC_Wait_Ack());

IIC_Send_Byte(cmd);　//内部寄存器地址

while(IIC_Wait_Ack());

BH1750_SendByte(REG_data);　//内部寄存器数据,

IIC_Stop();　//发送停止信号
```

最终程序达到的效果如图5-8所示。

图5-8

注：　LX代表光强的单位勒克斯。

5.3　LORA模块例程设计

这一节主要描述如何实现LoRa模块和网关的通信。　虽然说市面上存在的LoRa模块种类繁多，使用者不一定会用到我使用的这种，但是只要是LoRa芯片，都是大同小异。　并且我制作的开发板也制作了30块给学弟使用，所以详细说明一下也是很有必要的。　首先介绍一下模块引脚。　模块原理图如图5-9所示。

图5-9

我们可以清楚的看到模块由三部分组成，单片机，LoRa芯片，以及天线。　单片机负责接收并处理外界传来的信息并发送给LoRa芯片，LoRa芯片利用天线将信息发送给配对好的网关。

5.3.1 LoRa模块的配置

LoRa模块在刚买来的时候有一个默认配置，但是根据我的实验，在默认配置的情况下是无法连接上网关并且发送信息的。 在使用模块之前，您需要配置一系列模块。 模块的U1_TX和U1_RX是配置串口。 {47%：我们使用USB串口线将模块连接到计算机，然后打开SECURE_CRT，找到正确的端口并连接到我们的模块。} 然后在软件中输入“system”，然后再输入low power enable。 因为模块默认是工作在低功耗模式下的，需要单片机区唤醒，而对于我们例程来说是需要实时发送想要的数据，所以低功耗模式实属多此一举。其他设置可以根据芯片手册上的进行配置即可。 当我们由特殊需要时再进行调整。 设置完如图5-10所示。

图5-10

设置完成后，不要断开与电脑的连线，重置开发板及 LoRa模块然后再上电，看到SECURE_ CRT上出现如图5-11所示图样， 即显示“ join success”时说明模块与wangguan连接成功。 （连接成功的前提是要在网关的控制界面提前输入想要配对的 LoRa模块的 ID等信息， 才能正常配对，这部分内容属于网关的设置，与本设计无关，故在此不加赘述）

图5-11

5.3.2 LoRa模块通信软件设计

LoRa模块与单片机的通信方式是串口通信。 我一开始在设计这个串口通信的时候，由于模块说明不足，我无法通过说明文档有效的了解这个模块待敌应该怎么使用， 所以我采用了使用电脑当作串口发送短的方法，免去了反复调试程序的苦恼。 我将模块的U2_TX和U2_RX连接到连接到串行计算机的USB串行端口的另一端。 然后再上位机通过软件SSCOM直接向模块串口发送信息来测试模块的使用方法。 后来我弄清楚了模块应该怎样使用，所以我们家现在就可以放弃上位机，改用单片机对模块发送数据。

主要函数介绍：

（1）fputc： 这个函数的作用是修改函数定义，它使得printf这个大家都很熟悉的函数成为了串口发送函数。 发送的就是printf里面的内容。

（2）My_USART2_Init： {42%：串行端口初始化功能初始化串行通信所需的引脚，并设置为推挽模式。} 还有设置串口通信的参数。 {49%：波特率为9600，硬件流量控制关闭，停止位为1位，字长为8个字节。}

经过这样的设置后，发送信息就变得非常简单了，只需要先初始化一下，然后调用printf即可。 关键代码如下：

```
delay_init();

My_USART2_Init();

printf(”%d”，1);

delay_ms(50000);

printf(”%d”，2);
```

delay_ms(50000);

最终程序效果如图5-12所示： （SECURE_CRT内看到的结果）

图5-12

LoRa模块发送的是单片机发送数据的ASCII码，31和32分别是1和2的ASCII码。 我们的例程设定的就是间隔一段时间轮流发送1，2。 云端的交互界面上也显示了我们发送的数据，如图5-13.

图5-13

这说明通信成功，我们的用法是完全正确的。 此时，我们完成了整个项目的硬件和软件设计，并实现了预期的功能。

6 制作与调试

6.1 硬件电路的布线与焊接

6.1.1 总体特点

由于我们在电路设计中考虑到了布线方便的问题，将模块的信号脚尽量连接到离单片机近的脚，能用软件模拟协议的就使用软件模拟。 所以整体电路还算简洁，整个开发板设计在10x10（cm）的范围之内。 虽然再布线时经过了考虑，但是跳线在所难免。 我这回设计的电路板并非手工板，而是投板制作的双层板。 厂商的标准是线宽最低为6mil，这就大大方便了我的走线和打过控。 布线的大体原则是首先保证电源线和地，尽量短粗，减少消耗。 尽可能跟随顶部和底部线条，这可以减少干扰。

6.1.2 电路划分

为方便布线，我们将电路划分为几个大块分别设计PCB，最后再拼接起来，这样可以大大提升工作效率。 我将整个电路板分为： 1.单片机部分，2.电源部分，3.模块部分，4.LoRa部分。 四个部分分别设计电路，最后再进行拼接。

6.1.3 焊接

焊接方面，本毕业设计需要焊接30块同样的电路，工作量非常大，所以适当的技巧还是有必要的。 比如可以几块同时焊，就不用为了不同的器件找来找去。 还可以事先把电路原理图打印出来。 还可以一个部分一个部分的焊接，方便查找器件。

最终的电路图如图6-1所示。

图6-1

最终实物图如图6-2所示。

图6-2

6.2 调试

{43%：毕业设计的制作需要花费很多时间，并且在生产过程中遇到许多问题。} 我的最

终成品电路板是第三板，下面简单说一下前两块电路板的制作过程中出现的问题。

（1）基础的问题，第一板的打样回来，我发现有一个芯片封装对不上。 那个芯片的引脚是在芯片底下，，并没有伸到芯片外面来，所以根本无法焊到板子上。 这个芯片是6轴传感器芯片，如果用成熟的MPU6050模块的话就会大大超过预算，所以最后只得放弃这个念头。

（2）丝印的问题。 丝印对电路板的作用是很大的，能方便你的焊接以及告诉使用者什么是什么。 我第一次做板子的时候就没有考虑丝印的问题，只有一堆器件库离自带的丝印，看都看不清，对我的焊接造成了很大的麻烦，只能对着电脑的电路图一个一个找。 后来我不仅改进了这一点，而且还对单片机的每个引脚做了注释。 还对诸如LED1，LED2这种器件的标注。

（3）过孔数量，位置的问题。 这里的过孔主要连接到顶部和底部接地之间的过孔。我一开始的设计是打了很多的过孔，并且比较密集。 20个左右一组。 后来我了解到如果过孔太多太集中会有天线效应，电路板通电后会有很多个小天线，它们会往外辐射能量，造成不必要的损耗。 所以后来我全部改成4个一组。

7 结论

本设计方案达到了毕业设计题目的要求，能够实现在云端上查看传感器的实时状态的功能，LoRa的通信稳定，快速，实时。 传感器的功能也均正常。 开发板既实现了其作为一个普通单片机开发板的功能，也能当成一个LoRa节点使用。

由于时间、水平和经验有限，LoRa模块的功能并没有探索完全，按照原理来说，LoRa是可以实现双向收发功能的。 即开发板既可以向云端发送数据，同时云端也可以向开发板下发信息。 这也是个非常实用的功能，但是由于时间有限，我的例程中并没有这个功能的代码。不过这个功能的硬件上是一样的，所以只需要对软件进行开发就能实现。 这个任务就交给有心的使用者来完成了，

这次的毕业设计带给我完全不同于实验室里的体验。 由于该公司使产品与实验室的研究不同，因此这是为了获利。 所以成本这个在实验室里基本不去关注的东西成了首先需要考量的东西。 整块电路板尽量缩减成本，比如将PCB设计在10x10cm之内，选择双层板而非四层，单片机型号的选择等等。 还有就是包装的问题。 因为产品是用来卖的，没有良好的包装别人也不会买账，这些都是我之前从来没有关注过的。 最后就是焊接，由于这次的开发板需要30块，焊接也是一块很大的工作，这也考验了我的耐心和毅力。