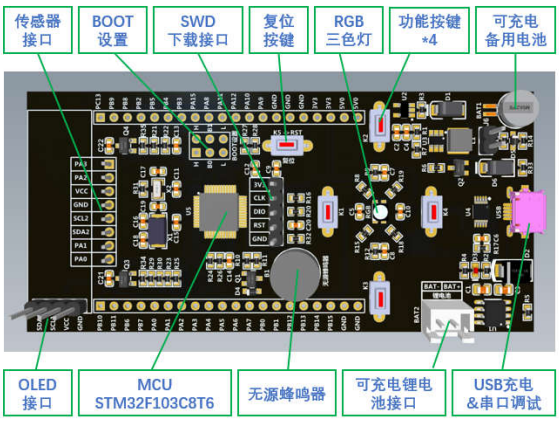
STM32 多功能开发板

使用手册

V1.0

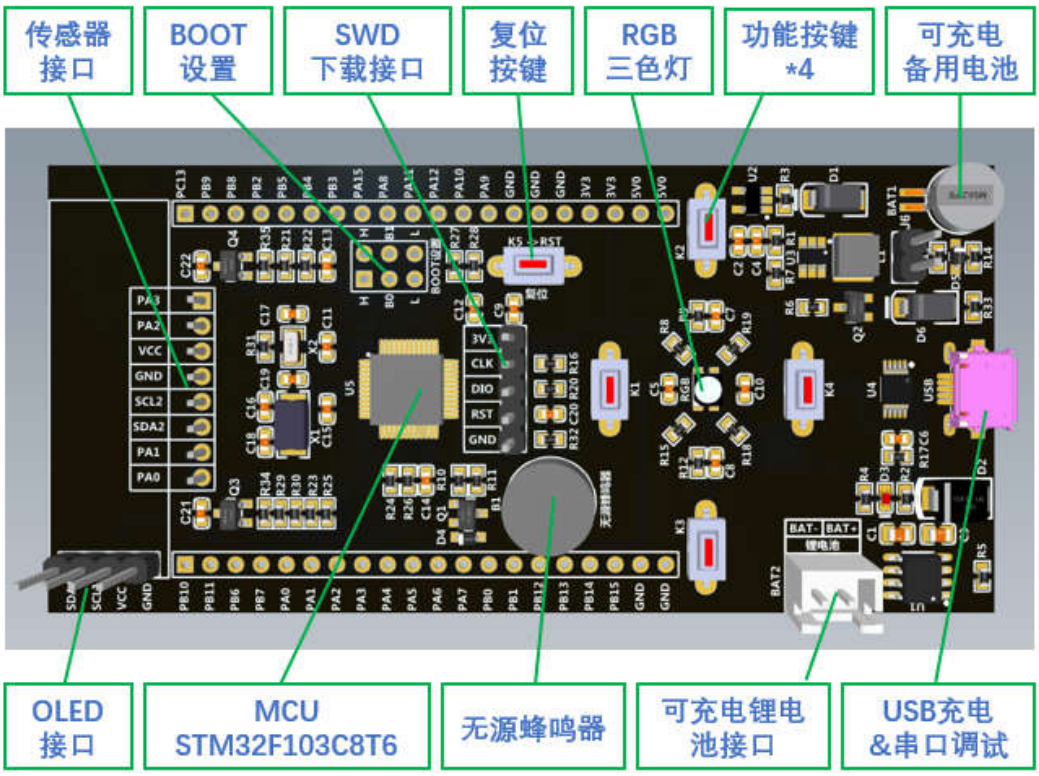
2020-04-09



一、 硬件篇

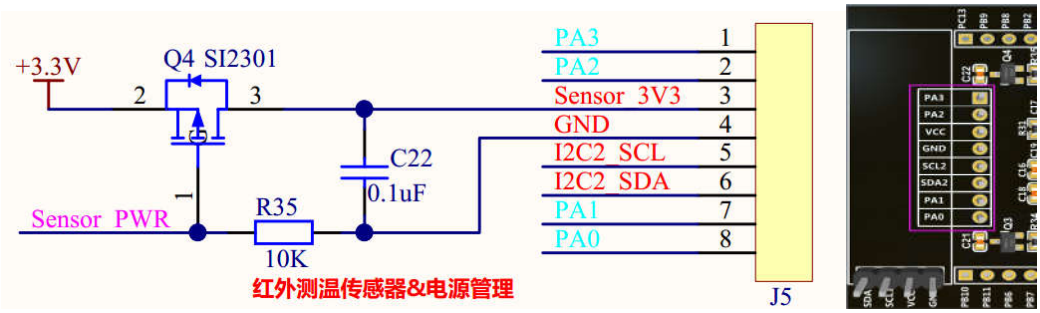
工欲善其事必先利其器。单片机的学习不仅需要书本的理论知识，更需要实际操作的硬件平台，否则一切是都是空中楼阁。除了硬件平台，还需软件开发工具，用于软件开发的有 Keil、IAR 等。 开发好的程序，还需下载器将其程序下载到 Flash 中，这时还需要下载器。同样，当我们在项目开发中，还会遇到各种各样的问题，此时还需调试器。

此开发板，是我们『芯知识学堂』成立以来，推出的首款开发板，主控芯片采用了 ST 的 STM32F103C8T6 这款处理器，板载详细功能如下图所示：

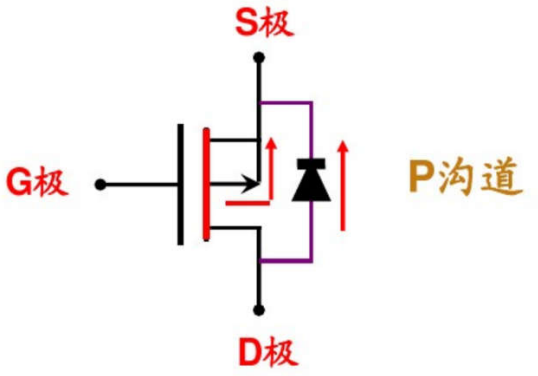


➢ **传感器接口：**

板载接口默认使用搭载了 MLX90614 这款红外测温传感器的 GY-906 模块，从板子背面直接连接，但是考虑到目前 MLX90614 这款传感器的短缺，以及方便后续用户做其他实验，特意预留了 4 路 ADC 接口。这里需要说明一下，此处的 PA0、PA1、PA2、PA3、SCL2、SDA2 和板子两侧的单排针上的 PA0、PA1、PA2、PA3、PB10、PB11 引脚是连在一起的，详见原理图。



传感器接口的电源部分，通过一个 P-MOS 管 SI2301 来作为传感器电源的控制开关，图中 SI2301 的 1 脚为 G 极（门极），2 脚为 S 极（源极），3 脚为 D 极（漏极）。



这里，我们需要明白 P-MOS 管导通和截止的条件是：当 P-MOS 管的 G 极与 S 极中间的电压差低于阈值时，P-MOS 管的 S 极和 D 极就会导通；反之，P-MOS 管的 S 极和 D 极就会截止。

根据 P-MOS 管的工作原理：

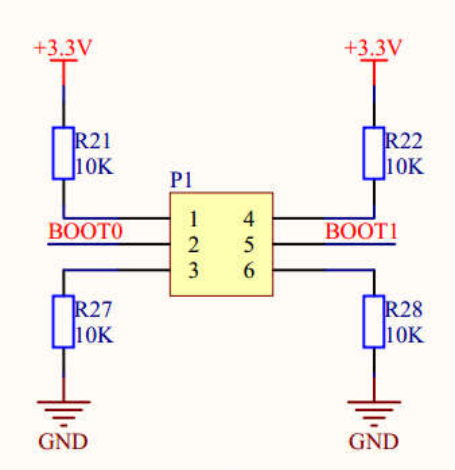
当 STM32 的 GPIO 口输出低电平时，P-MOS 管 SI2301 的 GS 电压差为 V_{GS}=-3.3V，此时，SI2301 处于导通状态；

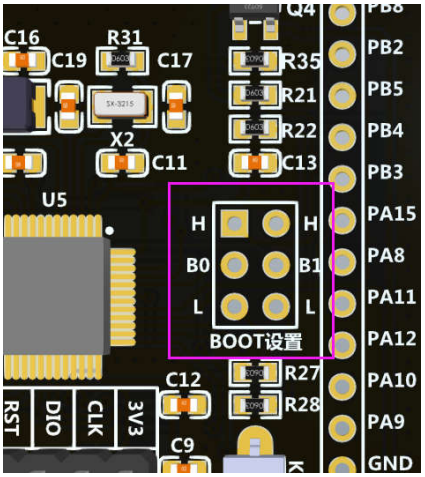
当 STM32 的 GPIO 口输出高电平时，P-MOS 管 SI2301 的 GS 电压差为 V_{GS}=0V，此时，SI2301 处于截止状态；

- **BOOT 设置：**

STM32F103 有多种启动模式可以选择。启动模式的理解是：微控制器允许从“不同地址”，比如内部 Flash（最常用）、RAM、系统代码，读取程序指令并执行，这个“不同地址”的选择是通过在为芯片供电（上电）后芯片会自动读取 BOOT0 和 BOOT1 引脚电平高低来决定的。这里的“不同地址”实际上是通过地址映射来实现，芯片总是从启动存储区开始执行程序的。举个类似的例子来帮助简单理解，比如，打听对应存储区 0，存放了一个房号，这个房号就是由 BOOT0 和 BOOT1 引脚决定的；房间 1 对应内部 Flash，房间 2 对应 RAM，房间 3 对应系统代码，在芯片上电时会现在大厅获取房号，然后找到对应房号，并执行对应的程序。

开发板通过排针+短接帽来配置 BOOT0 和 BOOT1，硬件原理图如下图所示：





当 B0/B1 与 PCB 板上的 **H** 短接时，就意味着将 B0/B1 **上拉**，此时 B0/B1 为**高电平（1）**；

当 B0/B1 与 PCB 板上的 **L** 短接时，就意味着将 B0/B1 **下拉**，此时 B0/B1 为**低电平（0）**；

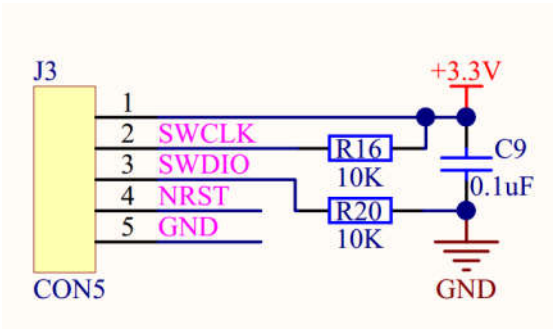
STM32F103 的启动模式和 BOOT0、BOOT1 的关系如下表所示：

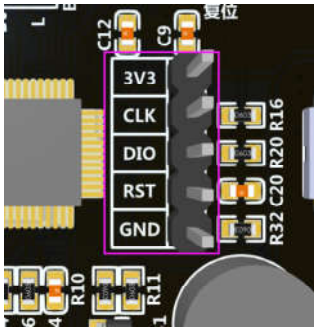
启动模式选择		
BOOT0	**BOOT1**	**启动模式**
0	任意	从 Flash 中启动（默认状态）
1	0	从系统代码（ISP）启动
1	1	从 RAM 中启动

一般程序都是存储在 Flash 空间的，默认在 Flash 启动运行；ISP 模式用于直接使用 USB 线进行串口下载；RAM 主要用于调试（需要先配置相关资源）。

- **SWD 下载接口**

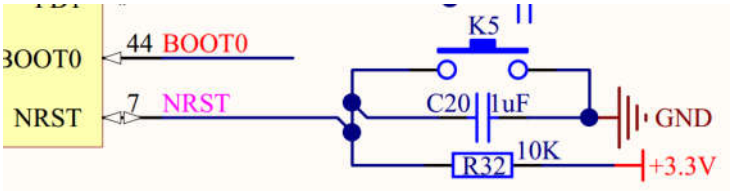
STM32 支持 JTAG 和 SWD 两种调试接口，这两种接口在调试功能上并没有差异。JTAG 接口是 ARM 早期的调试接口标准，需要 20 个引脚。SWD 接口只需要 6 个引脚。我们的开办板仅支持 SWD 调试接口（5 脚），不支持 JTAG 接口。

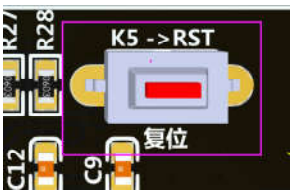




➢ **复位按键**

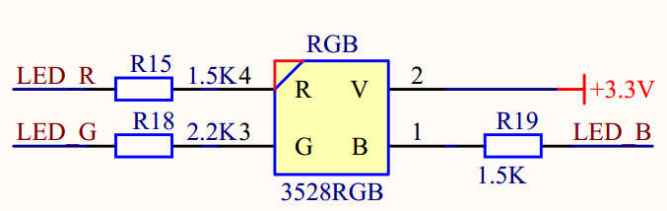
微控制器一般有软件复位和硬件复位两种方法，STM32F103 硬件复位信号为低电平有效，一般加上拉电阻（即另一端为高电平），保证一般情况下为高电平，如下图所示：

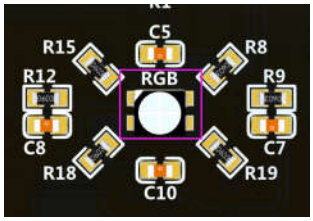




➢ **RGB 三色灯**

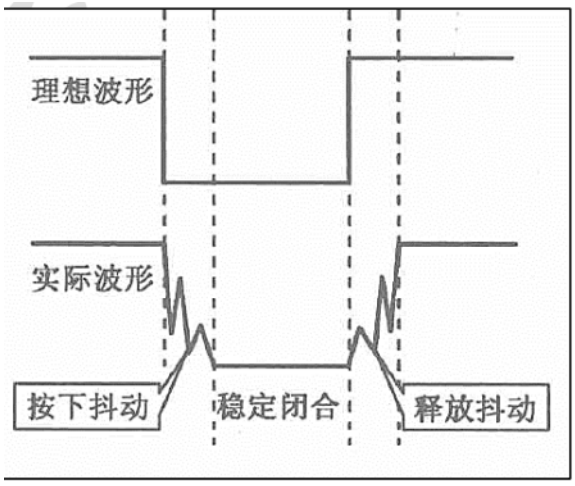
流水灯一直低入门学习微控制器的最简单、最经典的例子，体现了微控制器最基本的功能：控制引脚输出高低电平。此开发板上搭载了一颗 RGB 三色灯，三色灯采用共阳极接法，正极接到了 3.3V，负极通过一个限流电阻接到了 STM32F103 的 GPIO 口上，只要控制 STM32F103 的对应引脚输出低电平，就能点亮对应的灯，输出高电平，就能关闭对应的灯。通过一定的逻辑组合，就能实现流水灯的效果。



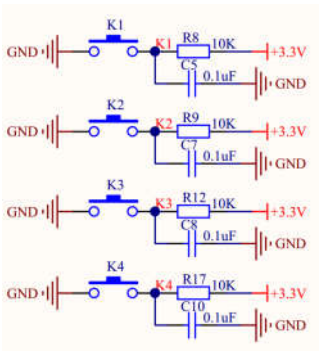


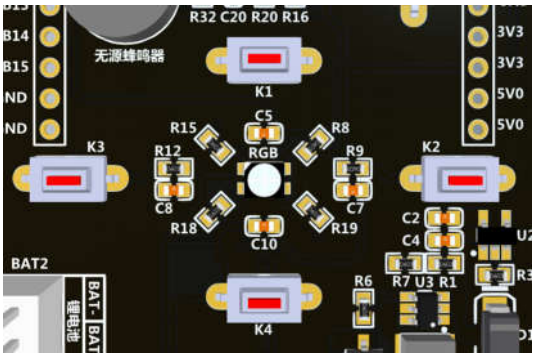
➢ **功能按键**

类似 RGB 灯，几乎每个开发板都有集成独立按键，因为从测试代码功能也好，实际应用也好，按键用处多多。此开发板也不例外，板载了 4 个功能按键可供用户使用。普通按键按下和弹开瞬间都有抖动过程，如下图所示：



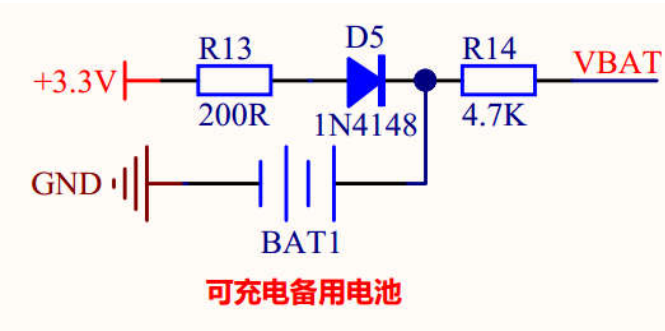
按键抖动过程一般为 5ms-10ms，有了抖动就需要消抖，不然很容易造成误操作。消抖方法可分为：硬件消抖和软件消抖。软件消抖在后续的软件部分会做相应的介绍，此处主要是介绍硬件消抖。此开发板按键部分主要通过一个上拉电阻和一个电容组成硬件消抖电路，其原理图及实物图如下：

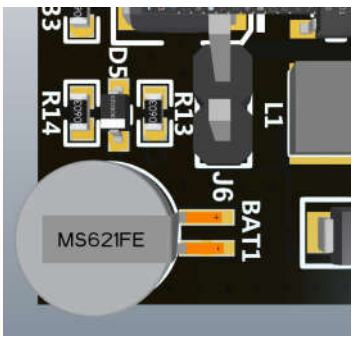




➢ **可充电备用电池**

备用电池的作用主要是在 STM32 断电时保持内部 RTC 始终正常运行，否则断电后 RTC 数据会丢失，重新启动后又从默认时间重新开始计数。一般开发板上都会带有一个备用电池，常用的要数 CR1220 这款纽扣电池了。由于我们板子空间比较紧凑，所以我们在选择备用电池的时候，也考虑到了电池的尺寸，从而选择了 MS621 这款容量为 5.5mAh 的可充电的锂电池来作为 STM32 的备用电池，其工作电路及实物图如下图所示：



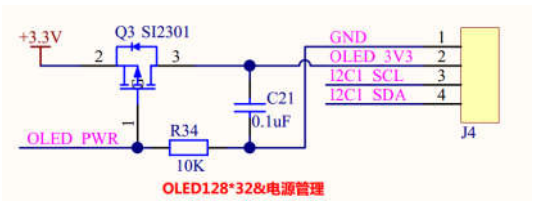


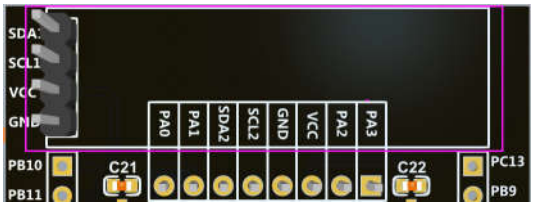
当外部电压 3.3V 存在时，外部 3.3V 通过限流电阻 R13 和整流二极管 D5 给 STM32 的 VBAT 供电，同时也可以给 MS621 充电。当外部 3.3V 段开始，STM32 的 VBAT 引脚由

MS621 供电。

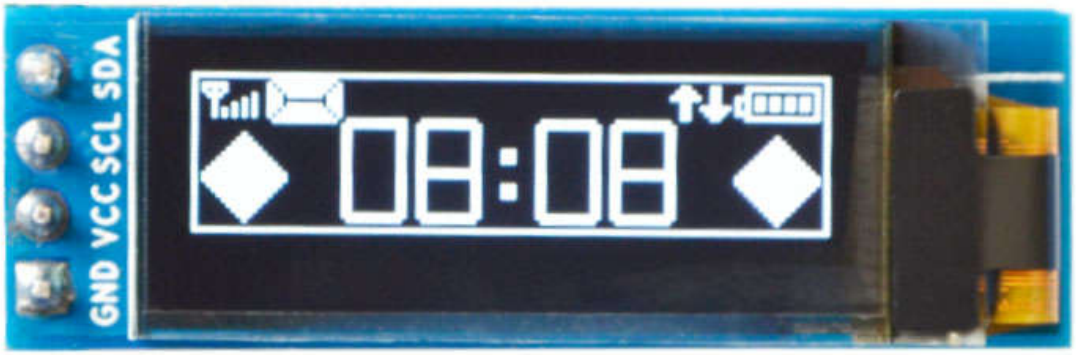
- **OLED 接口**

该开发板设计了一个 OLED 模块的接口，接口的通信方式是 IIC 通信，同传感器接口的电源一样，OLED 模块的电源部分也采用了一个 P-MOS 管 SI2301 来作为 OLED 模块电源的开关，控制原理前面已经讲过，这里就不再赘述了。





OLED 模块选用的是中景园电子的 0.91 寸的 OLED 模块，IIC 接口、分辨率为 128*32 像素。实物图如下：



关于 OLED 模块的使用，在后续软件部分会详细介绍，在这里就不再做介绍了。

- **无源蜂鸣器**

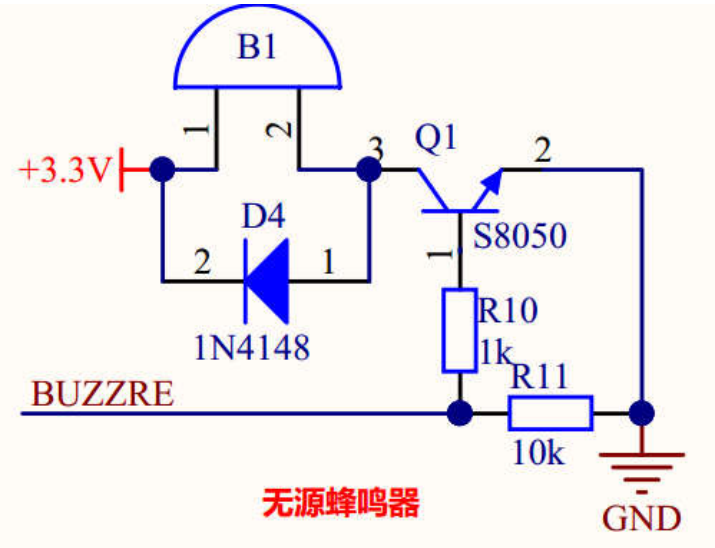
在单片机应用的设计上，很多方案都会用到蜂鸣器，大部分都是使用蜂鸣器来做提示或报警，比如按键按下、开始工作、工作结束或是故障等等。

有源蜂鸣器直接接上额定电源就可以联系发声；而无源蜂鸣器则和电磁扬声器一样，需要在音频输出电路中才能发声。

有源蜂鸣器和无源蜂鸣器的区别（注意：这里的“源”不是指电源，而是指震荡源。）：

- 有源蜂鸣器内部带有震荡源，所以只要一通电就会叫；
- 无源蜂鸣器内部不带震荡源，所以如果用直流信号无法令其鸣叫，必须使用一定频率的方波去驱动它（一般频率为 2kHz-5kHz）；
- 有源蜂鸣器一般比无源蜂鸣器贵，主要是因为内部多了个震荡电路。

此开发板上集成了一个无源蜂鸣器，电路设计参考图如下：



由于蜂鸣器的工作电流一般比较大，以至于 STM32 的 GPIO 口是无法直接驱动的，所以要利用放大电路来驱动，一般使用三极管来放大电流就可以了。

三极管的基极必须串接电阻，保护基极，即保护 STM32 的 GPIO 口。

基极和发射极需要串接电阻，即 R11 电阻，该电阻的作用是在输入呈高阻态时使三极管可靠截止，放置三极管受噪声信号的影响而产生误动作。三极管的基极不能出现悬空，当输入信号不确定时，加下拉电阻，就能使其有效接地。

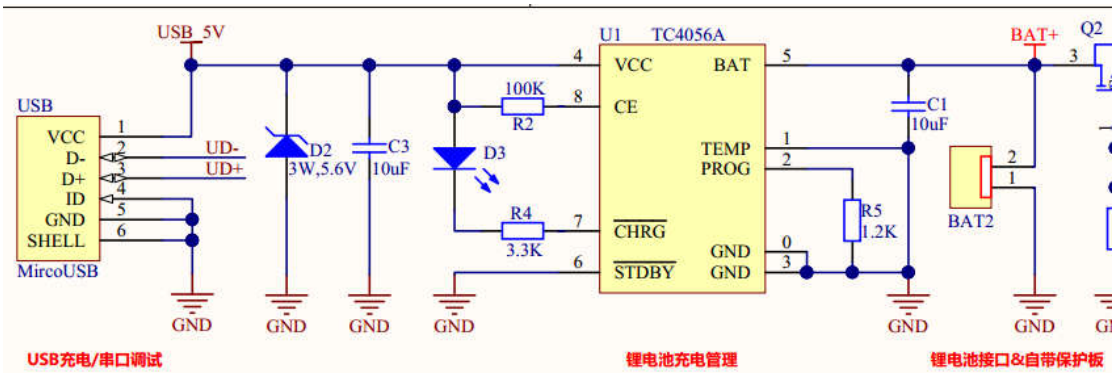
D4 1N4148 二极管的作用主要是续流，因为蜂鸣器是感性元件，当感性元件突然断电时会产生很大的反向感应电动势，可能会对电路中的其他电子元件造成损伤，因此，并联该二极管的目的是旁路掉此感应电动势，起保护作用。

- **锂电池充电电路**

此开发板搭载了一颗“单节锂电池充放电管理”的芯片 TC4056A，它是一款完整的单节锂电池采用恒定电流/恒定电压线性充电器，关于该芯片的详细参数，大家可以查阅手册，这款芯片的主要特点包括电池温度检测、欠压闭锁、自动再充电和两个用于指示充电、结束的 LED 状态引脚，可编程充电电流可高达 1000mA。

脚位	名称	说明
1	TEMP	NTC 引脚
2	PROG	充电电流控制脚
3	GND	地
4	VCC	电源正
5	BAT	充电电流输出端
6	STDBY	充满指示
7	CHRG	充电指示
8	CE	芯片使能引脚（高有效）

锂电池的充电管理电路如下图所示：



上图中，D3 主要是用来指示锂电池的充电状态，插入 USB 时，在 TC4056A 的作用下，开始给锂电池充电，指示灯 D3 常亮，当锂电池充满时，D3 常灭。R15 是用来限定 TC4056A 的最大充电电流，根据公式：I_{BAT}=（V_{PROG}/R_{PROG}）*1200，其中，V_{PROG} 在预充电阶段时的电压为 0.1V，此时电流 I_{BAT}=（0.1/1200）*1200=0.1A；V_{PROG} 在恒流充电阶段时的电压为 1V，此时电流 I_{BAT}=（1/1200）*1200=1A。

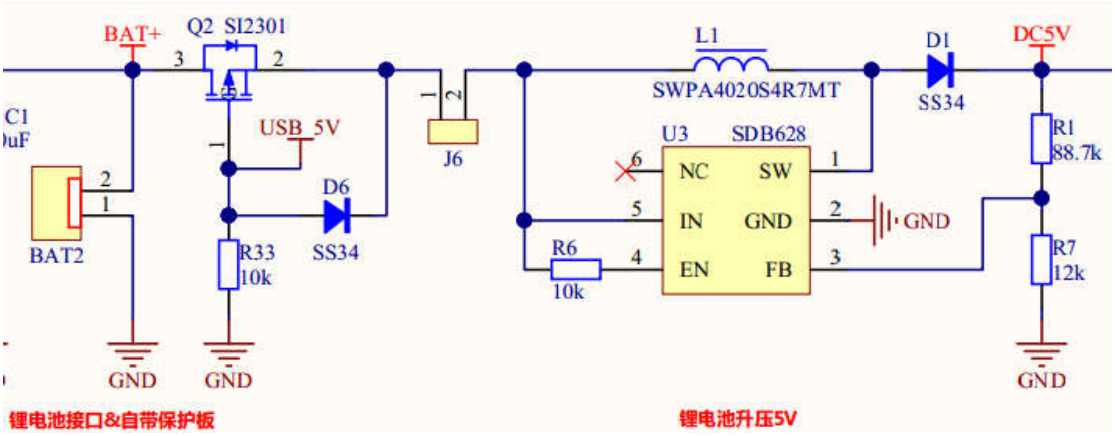
由于锂电池我们采用了自带保护板的铝包电池，所以我们在板子上没有再增加充电保护电路了，如果大家在更换电池的时候一定要注意选择带保护板的电池，否则有可能发生意想不到的事情。

- **升压电路**

考虑到我们的板子需要适应一些锂电池供电的应用，而标称为 3.7V 的锂电池的电压范围实际一般在 3.0V-4.2V，而对绝大部分常用的 LDO 或者 DC-DC 电源芯片来说，如果需要得到稳定的 3.3V 输出电压，那么输入电压必须要比 3.3V 高出至少 100mV 以上，这样的话，锂电池电压低于 3.4V 的时候就不能得到稳定的 3.3V 电压了，可能会造成某些元器件不能正常工作。

因此，我们开发板在电路设计的时候，采用了先升压、再降压的方案。当然咯，也许也有人会有疑问，这里为什么不直接用一款内部集成了同步升压、降压一体的电源芯片，这样岂不是更省事？这样做确实没毛病，也省事，但是，我们之所以选择先升压、再降压的方案，也是为了得到一个常用的电压—5V 电压，这样的话，用户在做别的实验需要用到 5V 的时候，就会很方便了有木有？

那么，我们先来看一下锂电池升压到 5V 的电路：



升压芯片采用的是 SDB628，这款芯片的输入电压范围为 2-24V，最高输出电压为 28V、最大输出电流为 2A，更多详细参数请大家查阅芯片手册。

这款芯片的可调节输出电压的公式为：$V_{OUT}=V_{REF}*(1+R1/R7)$，其中 V_{REF} 的典型值为 0.6V，那么根据原理图我们可以得到输出电压 $V_{OUT}=0.6V*(1+88.7K/12K)=5.035V$。大家如果需要其他的电压，只需根据公式计算，改变 R1 和 R7 的值就好了。

另外值得说一下的是，在锂电池的输出端，我们增加了一个 P-MOS 管 SI2301 用来作为锂电池输出的开关，细心的小伙伴们可能发现了，同样是 SI2301，但是用法似乎跟之前传感器电源控制和 OLED 电源控制有点不一样了，难道是我们设计错了吗？

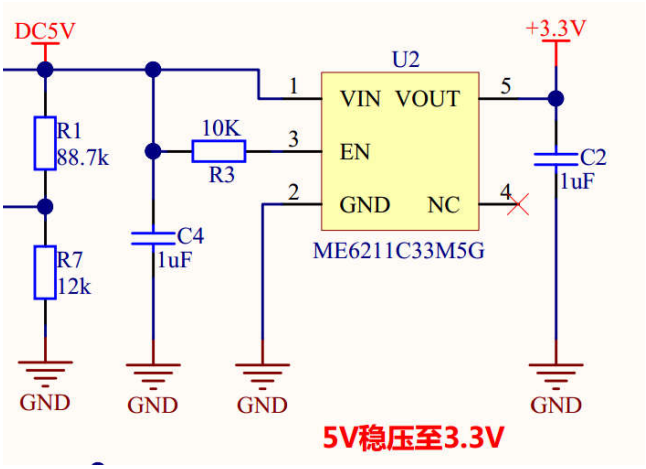
其实，我们这里并没有设计错，只是换了一下用法，且听笔者分析一下这部分的工作原理，大家就明白了。

- 当外部 USB 插入时，MOS 管 SI2301 的 1 脚（G 极）和 2 脚（S 极）之间的电压差等于 SS34 两端的电压，只有零点几伏，并未满足 SI2301 的 D 极和 S 极导通的条件，所以此时锂电池仅处于充电状态，并未放电，后极电路的电源完全由外部 USB 提供，也就是说，外部 USB 在给锂电池充电的同时，也在为后极电路供电；
- 当外部 USB 拔掉时，MOS 管 SI2301 的 1 脚（G 极）在下拉电阻 R33 的作用下，被拉低到了 GND，此时 MOS 管 SI2301 的 2 脚（S 极）的电压也几乎为 GND 电压，MOS 管 SI2301 的 D 极和 S 极是处于未导通状态的，但是，由于 MOS 管 SI2301 内部寄生二极管的作用，导致了 MOS 管 SI2301 的 D 极和 S 极之间直接形成了通路，这样又使得 MOS 管 SI2301 的 2 脚（S 极）的电压接近了电池的输入电压，而其 1 脚（G 极）在下拉电阻的作用下被拉低到了 GND 处电压，于是 G 极和 S 极的电压差又产生了，而且这个电压差高于 SI2301 的 G 极阈值，从而又导致 MOS 管 SI2301 的 D 极和 S 极导通了。

看了这个电路的工作原理，相信大家不会再质疑这部分的电路有问题了吧！？另外，图中的 J6 实际上是用一个短接帽连起来的，这里主要是为了在做低功耗应用测回路电流的时候提供了方便，其他也没有什么太大的用处。

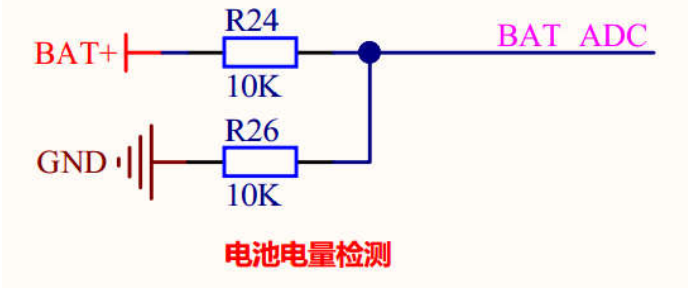
➢ 降压电路

开发板的降压部分，采用的是 ME6211C33M5G 这款 LDO，这款 LDO 的输入最大电压为 6V，输出电压为固定 3.3V，输出最大电流为 500mA。封装采用的是 SOT-23-5，也是比较省空间的，应用原理图如下图所示：



➢ **电池电量检测**

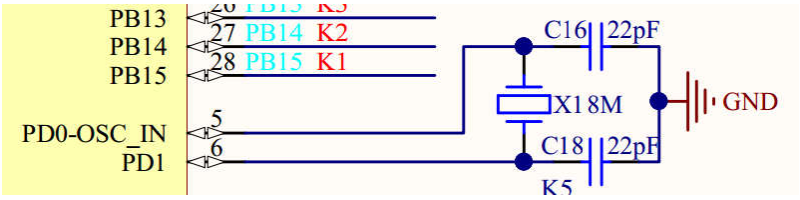
此开发板上电池电量的检测主要采用比较粗糙的直接测电池电压的方法，这种方法相对一些专用的电量芯片来说，还是比较简单，但是并不能检测到电池的真实电量，主要应用在对电池电量检测精度要求不高的场合。



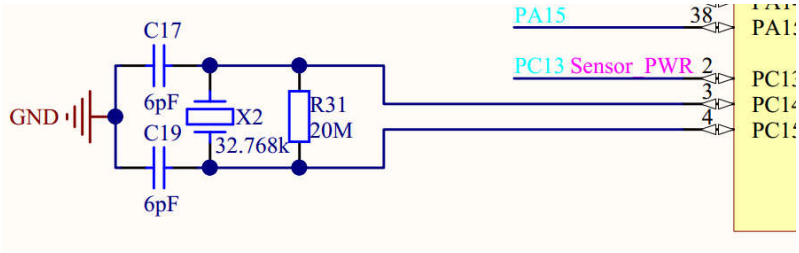
我们是直接将两个电阻分压后电压送入到 STM32 的 AD 口进行采集，计算出分压后的电压，然后再通过分压比例，计算出电池的电压。开发板上电池的电压为 AD 口采集到的电压的 2 倍。

➢ **外部时钟电路**

时钟是微控制器的脉搏，重要性可想而知，微控制器都需要一个基本的时钟节拍，即时钟基准，不然会导致时序错乱，完全乱套了，系统就崩溃了。一般，我们需要为 STM32F103 芯片提供一个外部时钟源，一般使用 8MHz 的无源石英晶振，如下图所示：

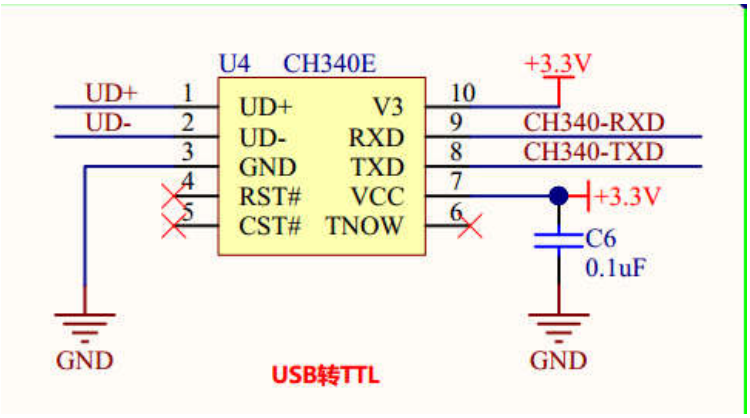


实际上，STM32F103 芯片内部也有一个主时钟源，也是 8MHz 的，但是一般我们不使用，而是使用外部晶振，主要是处于稳定性的考虑。另外，为使用 RTC（实时时钟、万年历）还需要提供 32.768 KHz 的时钟源，由于芯片内部提供时钟是 32KHz，不是 2 的 N 次方，无法得到准确的时间，所以，为得到精准的时间，我们需要用外部的 32.768 KHz 的晶振，电路设计如下图所示：



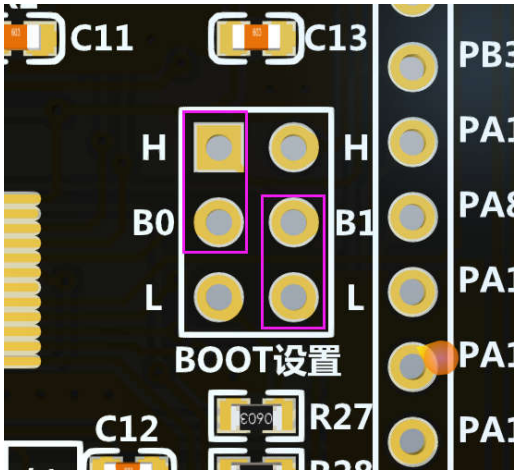
➢ **串口下载/调试**

此开发板上集成了一颗 USB 转 TTL 电平的芯片 CH340E，有了这款芯片，用户可以直接使用 ISP 串口下载程序，同时也可以用来做串口通信的实验。CH340E 的应用电路如下图所示：



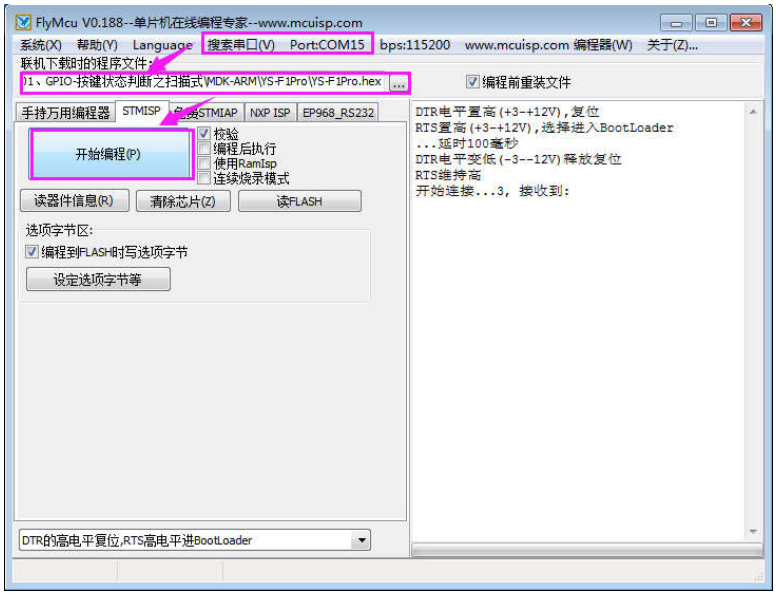
CH340E 这款芯片的外围电路相比于其他同类型芯片来说，简直是简单的不要不要滴！

在进行 ISP 下载时，我们首先要将 BOOT0 设置为高电平，BOOT1 设置为低电平，即：将板子上的 B0 跟 H 短接，B1 跟 L 短接，这样 STM32F103 就可以进入 ISP 下载模式了，如下图所示：



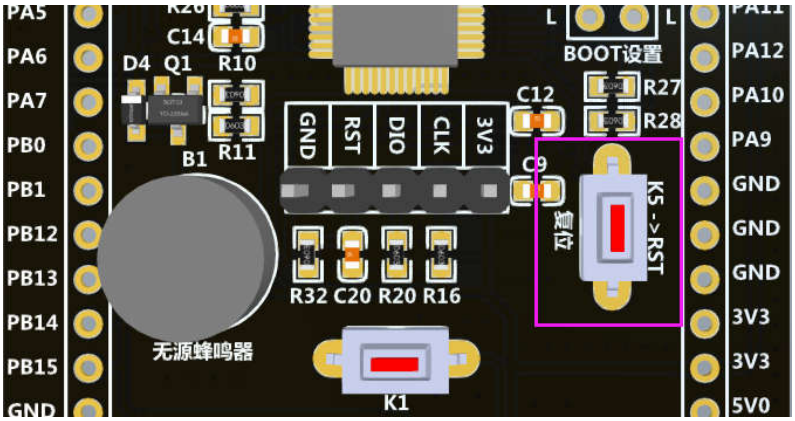
注意，这里我们讲的是“可以进入”而事实上并不会马上进入 ISP 下载模式，因为，只有系统复位后 MCU 才会读取 BOOT 引脚电平，才会决定启动模式。所以，在对 STM32F103 进行 ISP 下载的时候，需要对 STM32F103 进行一次手动复位。我们的开发板上给大家设计好了复位按键，可以让大家很方便的对 STM32F103 进行复位。

我们可以通过 FlyMcu 这款免费的软件，来对 STM32F103 进行程序的下载，软件界面如下图所示：

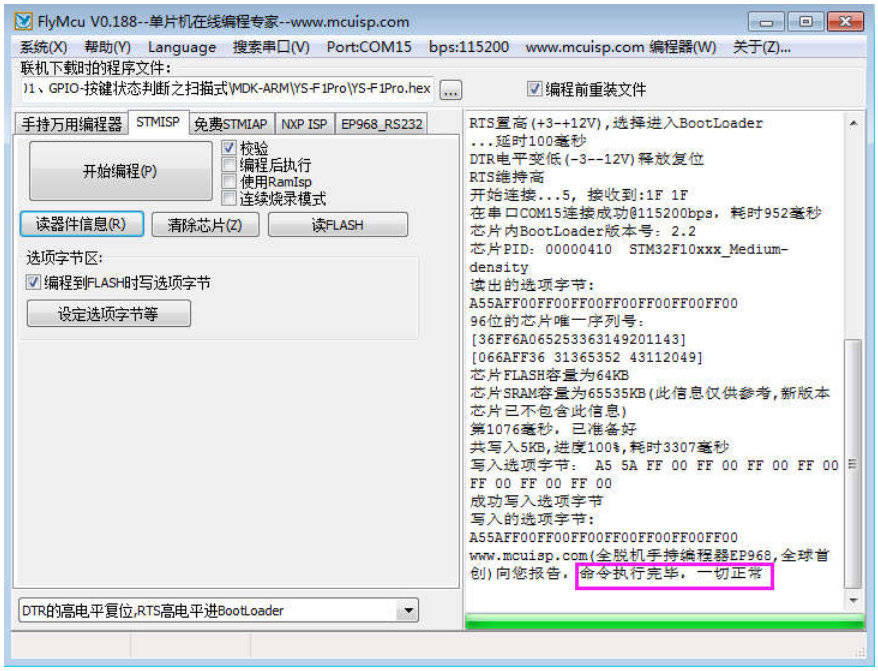


ISP 下载操作流程如下：

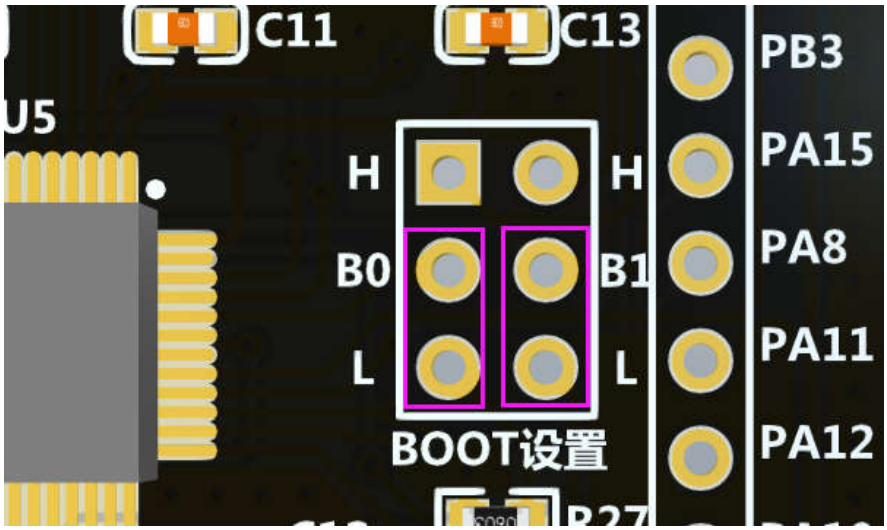
- 插入 USB 线，并跟电脑连接，安装好 CH340 的驱动；
- 打开 FlyMcu 软件，搜索串口，然后找到我们的设备 COM 口号；
- 加载程序的 hex 文件；
- 在 STMISP 选项中点击“开始编程”；
- 在出现“开始连接”提示后，手动按下开发板上的“复位”按键；



- 等待 FlyMcu 软件出现如下字样时，说明程序下载成功：



程序下载成功后，再将 BOOT0 设置为 0，BOOT1 设置为 0，并给板子复位或者重新上电，程序就能正常执行了。



关于板子的硬件部分就给大家介绍到这里了，如有疑问，请在我们的 QQ 群里面提问交流：

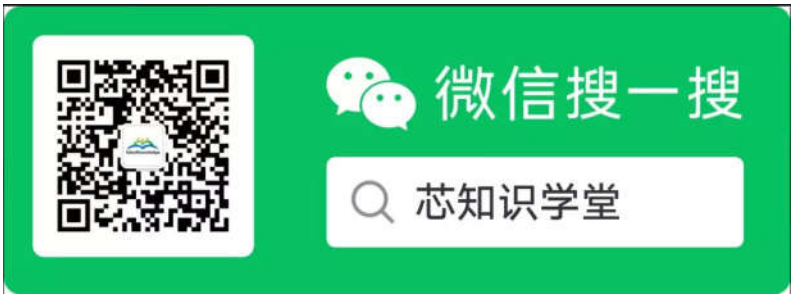
物联网&嵌入式开发群：

544101253

单片机&工控技术交流群：

82558344

也欢迎大家关注我们**『芯知识学堂』**的**微信公众号**，我们会不定期更新更多的技术干货，同时我们也会在公众号里分享大量的学习资料：



另外，如果想看更多学习视频的小伙伴们，也可以移步到我们『芯知识学堂』的 B 站（哔哩哔哩 bilibili），我们的 B 站链接为：https://search.bilibili.com/all?keyword=%E8%8A%AF%E7%9F%A5%E8%AF%86%E5%AD%A6%E5%A0%82，里面有很多款视频，总有一款是你想要的！