摘　要

报告简单地记录了本人与小组成员于 2020.6.15~2020.6.20 期间在**公司进行了达 6 天的线上嵌入式培训。在***老师的指导下，我们第一天先将所需的软件（keil5，proteus）装好并学会使用，搭建一个工程。第二天学习了 GPIO 口的使用，以及时钟树，了解了 STM32 的时钟源，STM32F401 启动文件中一些语句的意思，还有外部中断，EXIT 控制器等等。第三天我们学习 GPIO 口的复用功能--串口通信，有 GPIO 结构体和串口结构体，使能串口时钟和 GPIO 时钟，将 GPIO 复用位串口收发端，配 GPIO 口复用功能（推挽模式）并生效到寄存器，配置串口信息（波特率、数据位、停止位、校验位、硬件流控、模式<收/发>）并生效到寄存器，使能串口工作，使用数据发送来验证串口是否工作正常等。了解了实体的 DHT11 后，我们进行了一个简单的功能设计（温湿度数据获取）。第四天学习了工程代码模块化，即将各功能函数进一步封装成独立的.c 语言文件，将独立的模块化文件添加工程中即可使用对应的功能，以及 STM32F401 定时器(timer)使用。第五天学习了定时器输出 PWM 波和光敏传感器数据采集以及传感器的配置，数码管的配置显示以及连接到相应模块。第六天老师给我们讲解了整合模块的一些细节。

最终以三个功能模块的设计以及各功能模块整合结束此次培训。经历了这六天的学习，我们更深入地了解了嵌入式，上手设计使得理论更具体，但可惜的是无法真正在实体开发板上设计开发。

目　　录

1 温湿度获取模块概述 1
1.1 DHT11 1
1.1.1 原理及原理图 1
1.1.2 代码思路 4
1.2 蜂鸣器模块 5
1.2.1 原理及原理图 5
1.2.3 代码思路 5
2 光照获取与距离检测模块 7
2.1 光照获取 7
2.1.1 原理及原理图 7
2.1.2 代码思路 7
2.2 光照结果数码管及串口显示 7
2.2.1 代码思路 7
2.3 距离检测 8
2.3.1 原理及原理图 8
2.3.2 代码思路 8
2.3.3 遇到的困难 9
3 气体（滑动变阻）检测模块 9
3.1 气体（滑动变阻）检测 9
3.1.1 原理及原理图 10
3.1.2 代码思路 10
3.1.3 遇到的困难 11
3.2 检测结果串口打印及数码管显示 11
3.2.1 代码思路 11
Abstract 11
参考文献 12

1 温湿度获取模块概述

1.1 DHT11

1.1.1 原理及原理图

首先先了解其中所需的传感器 DHT11。DHT11 数字温度传感器是一款含有已校准数字信号输出的温湿度符合传感器。其实物图和原理图如图 1、2 所示，其特性如图 3 所示。

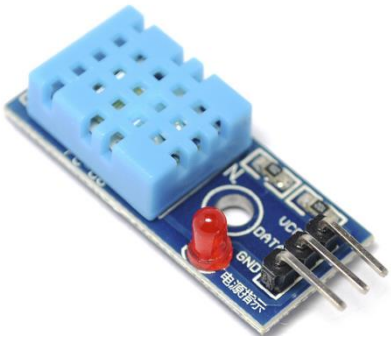

图 1 DHT11 实物图

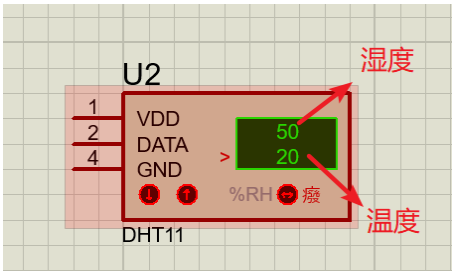

图 2 DHT11 原理图

数字温湿度传感器

DHT11

►相对湿度和温度测量
►全部校准，数字输出
►卓越的长期稳定性
►无需额外部件
►超长的信号传输距离
►超低能耗
►4 引脚安装
►完全互换

图 3 DHT11 的特性

DHT11 和微处理器之间采用串行接口的方式进行通信，采用单总线数据格式，一次完整的数据传输为 40bit，高位先出。DHT11 和微处理器之间的通讯过程如下图 4 所示。

先主机（微处理器）发出低电平信号，至少持续 18ms，之后主机拉高电平，并转换为输入模式，等待 DHT11 的响应，若 DHT11 输出 80us 的低电平，即 GPIO 输入 80us 的低电平，表示 DHT11 发出响应，之后 GPIO 再输入 80us 的高电平，之后开始传送数据。

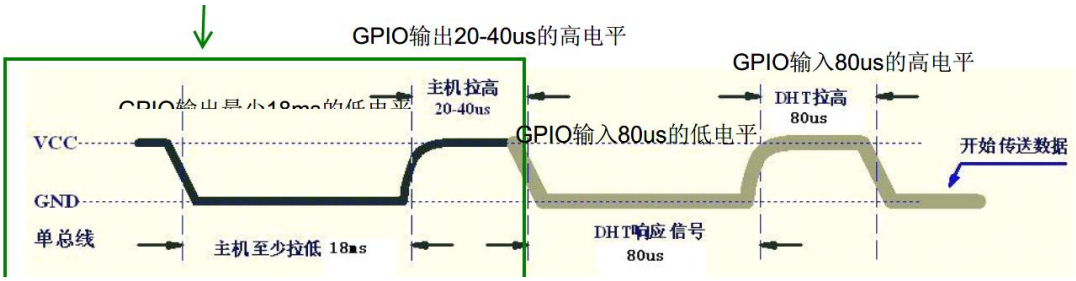

图 4 DHT11 和微处理器通信过程

开始传送数据后，GPIO 输入高电平，主机会根据输入高电平的时间长度来判断为 1 或 0。当输入的数据位为 0 时，持续高电平的输入时间长度在 26us~28us 之间，便开始输入低电平，表示该数据位结束，其时序过程如图 5 所示。

数字0信号表示方法如图4所示

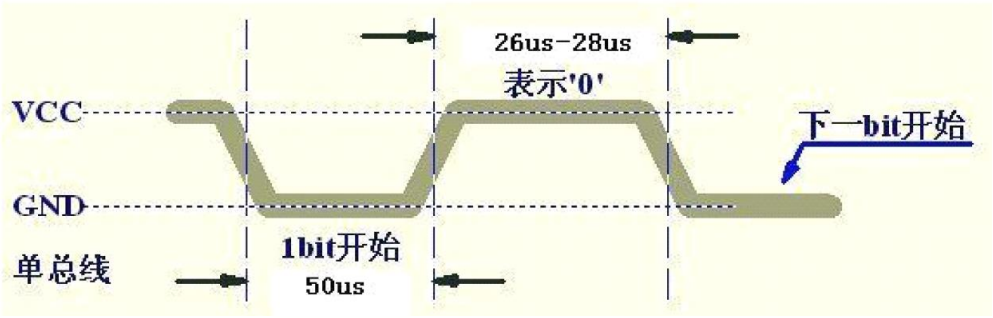

图 5 数字 0 信号传输时序图

数据 1 的传输相比数据 0 的传输在于高电平的输入时间达到 80us，远远超过 28us，传输完成便转为低电平。

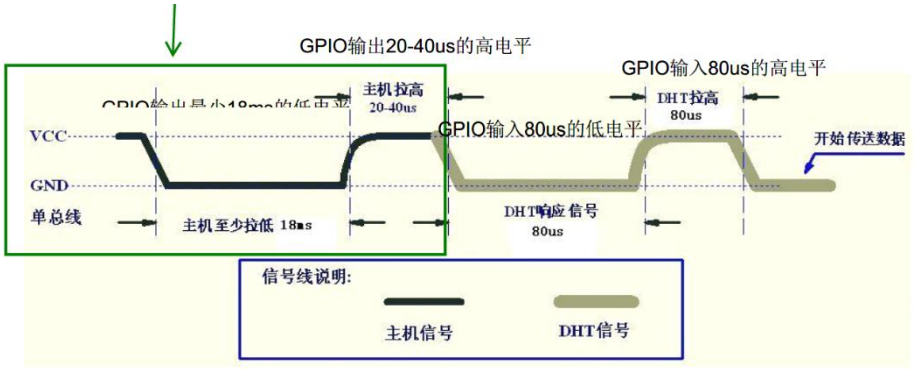

图 6 数字 1 信号传输时序图

其总的时许图如图 7 所示。

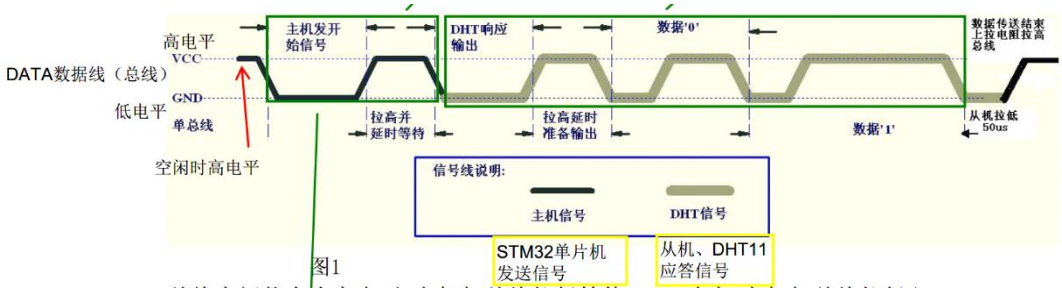

图 7 DHT11 与主机传输总时序图

1.1.2 代码思路

根据时许图所示，先把 GPIOD12 配置为输出模式，输出低电平至少 18ms 给 DHT11，发出信号，然后再发出 30us 的高电平信号。接着 GPIOD 配置为输入模式，等待 DHT11 的低电平的响应信号，在等待完一个低电平信号、一个高电平信号、一个低电平信号的完结后，开始接受数据，通过高电平的时间长度判断数据为 0 或 1。再通过第五位校验位来判断数据是否正确。

```c
void Read_DHT11_Data(void)
{
  uint8_t ret,i,DHT11_data[5];

  //1、主机发送起始信号
  DHT11_Start();
  //2、从机应答
  ret = DHT11_Response();
  if(ret == 0)
  {
    //3、读取数据(读5次)
    for(i=0; i<5; i++)
    {
      DHT11_data[i] = Read_DHT11_DataByte();
    }
    //4、校验数据
    if(DHT11_data[0]+DHT11_data[1]+DHT11_data[2]+DHT11_data[3] == DHT11_data[4])
    {
      if(flag == 1)
      {
        printf("————————————————\r\n");
        printf("当前环境温度：%d ℃\r\n", DHT11_data[2]);
        while(flag == 1)
        {
          //温度超于28或低于18，蜂鸣器响
          if( !(18<=DHT11_data[2] && DHT11_data[2]<=28) )
          {
            PDout(0) = 1;
          }
```

图 8 DHT11 与主机通信部分代码

1.2 蜂鸣器模块

1.2.1 原理及原理图

此蜂鸣器采用 SPEAKER 进行模拟，参照原理图画出连接电路，其连接电路图如图 4 所示。

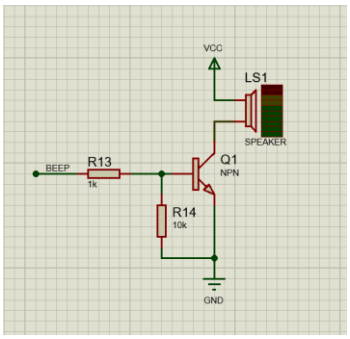

图 9 蜂鸣器原理图

1.2.3 代码思路

先配置一个 GPIO 口作为输出，当需要蜂鸣器响时，输出 1 即可。其配置 GPIO 口的代码如图 10 所示。

```c
#include "beep_init.h"

void beep_init(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure; //定义GPIO结构体

  /* GPIOD Peripheral clock enable 使能GPIOA时钟*/
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

  /* Configure PD0 in input  PullUp mode */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;          //引脚编号：0号
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;       //GPIO模式：输出模式
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_High_Speed;   //IO速度：高速
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;       //使用上拉电阻:给IO一个确定的电平
  GPIO_Init(GPIOD, &GPIO_InitStructure); //按初始化结构体的要求对GPIO进行初始化

  PDout(0) = 0;
}
```

图 10 蜂鸣器配置代码

在 DHT11 检测到数据后，若温湿度高于或低于一定值，则使输出口为高电平，蜂鸣器响。其代码如图 11 和图 12 所示所示。

```c
if(flag == 2)
{
  printf("————————————————\r\n");
  printf("当前环境湿度：%d %%\r\n", DHT11_data[0]);
  while(flag == 2)
  {
    //湿度高于70%或低于30%，蜂鸣器响
    if( !(30<=DHT11_data[0]&&DHT11_data[0]<=70) )
    {
      PDout(0) = 1;
    }
```

图 11 湿度部分蜂鸣器响应代码

```c
 if(flag == 1)
 {
   printf("————————————————\r\n");
   printf("当前环境温度：%d ℃\r\n", DHT11_data[2]);
   while(flag == 1)
   {
     //温度超于28或低于18，蜂鸣器响
     if( !(18<=DHT11_data[2] && DHT11_data[2]<=28) )
     {
       PDout(0) = 1;
     }
```

图 12 温度部分蜂鸣器响应代码

2 光照获取与距离检测模块

2.1 光照获取

2.1.1 原理及原理图

通过与光敏传感器相连接，利用光敏元件将光信号转换为电信号，再通过计算将获得的电信号换算为光信号便可得光照值。由于疫情影响没有实物操作，通过 proteus 的光敏电阻实现仿真，由于光敏电阻的变化获得光照的强度变化，使用 ADC 转换获取转换值，通过转换值计算得出实际电压值。光照越强电压越小，光照越弱电压越大。

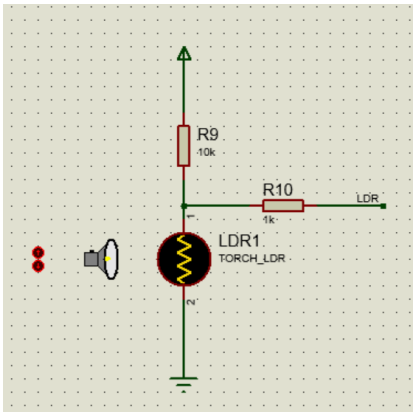

图 13 光照获取原理图

2.1.2 代码思路

初始化 ADC 转换光敏电阻函数，开始 ADC 转换，当 ADC 转换结束时，获取转换值，失能 ADC。

```
LDR_Init1();
uint16_t  LDR_tmp, LDR_value;

/* Start ADC1 Software Conversion */
ADC_SoftwareStartConv(ADC1);

while(ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC) == RESET);//等待转换结束

LDR_tmp = ADC_GetConversionValue(ADC1); //获取转换值
LDR_value = LDR_tmp * 3300/4096;

//光照越强电压越小，光照越若电压越大
printf("转换值是：%d 实际电压：%dmv \r\n",LDR_tmp, LDR_value);

//失能ADC1
ADC_Cmd(ADC1, DISABLE);
```

图 14 光照获取代码

2.2 光照结果数码管及串口显示

2.2.1 代码思路

先通过 printf 函数串口输出光照结果，而数码管需要一直循环位选段选输出光照值，定义一个判断函数，若还在此模块则一直循环输出光照值，每

个位选之间需要一点延迟来确保数码管可见。

```
//光照越强电压越小，光照越若电压越大
printf("转换值是：%d 实际电压：%dmv \r\n",LDR_tmp, LDR_value);

//失能ADC1
ADC_Cmd(ADC1, DISABLE);
//循环数码管显示电压
while(flag == 3)
{
  display(1,LDR_value/1000%10);
  delay_ms(20);
  display(2,LDR_value/100%10);
  delay_ms(20);
  display(3,LDR_value/10%10);
  delay_ms(20);
  display(4,LDR_value%10);
  delay_ms(20);
}
```

图 15 串口及数码管输出代码

2.3 距离检测

2.3.1 原理及原理图

通过超声波测距模块（HC-ST04）发出超声波，检测返回的超声波所消耗的时间，计算所得检测的距离所长。超声波测距模块采用 IO 触发测距，给至少 10us 的高电平信号，模块自动发送 8 个 40khz 的方波，自动检测是否有信号返回，有信号返回，通过 IO 输出一高电平，高电平持续的时间就是超声波从发射到返回的时间，测试距离=（高电平时间*声速（340M/S）/2。

图 16 超声波测距模块实物

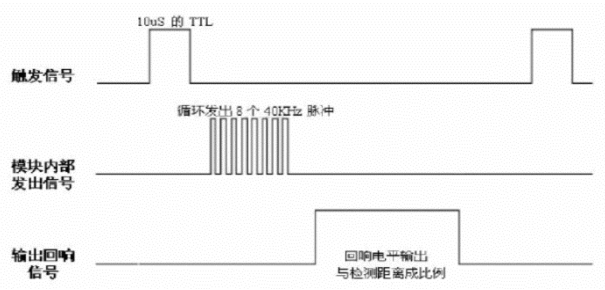

图 17 超声波时序图

2.3.2 代码思路

根据图 5 超声波时序图，配置 GPIOD6 口作为输出，GPIOD7 作为输入。先从 GPIOD6 输出一个高电平持续时间 10us 给超声波测距模块，超声波测

```
uint32_t SR04_Getdis(void)
{
  uint32_t dis;
  uint32_t tmp=0;

  //超声波测距模块初始化
  HCSR04_Init();

  PDout(6) = 1;
  delay_us(11);

  while(PDin(7) == 0);//容易卡住

  while(PDin(7) == 1)
  {
    delay_us(8); //约 3mm
    tmp++;
  }

  dis = tmp*3/2;

  return dis;
}

void GETdis(void)
{
  uint32_t dis;

  dis = SR04_Getdis();

  printf("————————————————\r\n");
  printf("距离值是: %d \r\n",dis);

}
```

图 18 测距相关代码

距模块收到高电平后，会自动循环发出 8 个 40KHz 脉冲并检测回波，一旦检测到有回波信号则输出回响信号，通过 GPIOD7 输入数据，再通过计算便可的所得距离。

先初始化超声波测距模块，在输出 PDout(6)=1，延时 10us，然后等待超声波测距模块的高电压返回信号，再开始收集数据。

2.3.3 遇到的困难

在仿真测试当中，经常发生“卡住”现象。在按下按键 3 后进行光照和距离检测时，中断服务函数启动成功后，但是并不显示光照和距离值。

经过排查分析，发现程序经常在 SR04_Getdis(void)函数中，即在 GPIOD6 发送高电平信号给测距模块后，测距模块一直没有返回回响信号，程序处在 while(PDin(7) == 0);无限循环中，一直等待测距模块的回响信号，使程序发生“卡住”现象。

对此我们小组经过讨论，认为这种情况可能是由于 proteus 仿真所出现的 bug，这种情况时有发生时无发生。

3 气体（滑动变阻）检测模块

3.1 气体（滑动变阻）检测

3.1.1 原理及原理图

原本此模块用于检测家居中气体，由于 proteus 中并没有气体检测模拟元件，则采用滑动变阻来代替气体，其原理都是大同小异的。用 ADC 转换将气体（滑动变阻）转换为转换值，再通过计算所得气体（滑动变阻）的值。

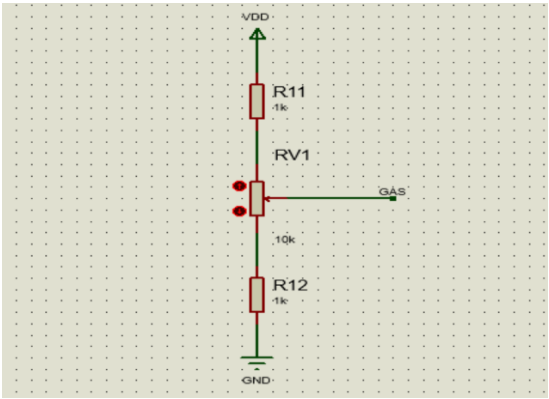

图 19 气体（滑动变阻）原理图

3.1.2 代码思路

由于之前在检测光照时已经使用了 ADC1，而这次又需要使用 ADC 检测气体（滑动变阻）。在初步时有三种方案。

第一种方案：在 ADC1 中建立两条通道，采用扫描、连续转换模式不断扫描两条通道的数据。

第二种方案：在 ADC1 外再建立一个 ADC2，在此 ADC2 里建立一个通道。

第三种方案：建立两个 ADC1 分别绑定一个通道，若要检测光照，则失能检测气体（滑动变阻）通道的 ADC1，使能检测光照通道的 ADC1；反之亦然。

对于这三种方案，首先排除了方案二，由于仿真中没有 ADC2，所以我们并不能使用 ADC2。在方案一和方案三中，两种都通过了尝试，但是只有

方案三最终实现了我们的需求。

```
LDR_Init2();
uint16_t GAS_tmp, GAS_value;

/* Start ADC1 Software Conversion */
ADC_SoftwareStartConv(ADC1);

while(ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC) == RESET);//等待转换结束

GAS_tmp = ADC_GetConversionValue(ADC1) ; //获取转换值
GAS_value = GAS_tmp * 10000/4096;

printf("————————————\r\n");
//光照越强电压越小，光照越若电压越大
printf("气体检测结果为：%d  \r\n",GAS_value);

//失能ADC1
ADC_Cmd(ADC1, DISABLE);
//循环数码管显示气体检测结果
while(flag == 4)
{
  display(1,GAS_value/1000%10);
  delay_ms(20);
  display(2,GAS_value/100%10);
  delay_ms(20);
  display(3,GAS_value/10%10);
  delay_ms(20);
  display(4,GAS_value%10);
  delay_ms(20);
}
```

图 20 气体（滑动变阻）代码

3.1.3 遇到的困难

在 proteus 当中，根据老师提供的原理图如图 9 所示，但是在测试过程中，只要一接上此原理图，key1 和 key2 的中断服务程序都会失效，但是源码并没有任何改变。为此，我们改变了上下两个电阻的大小，由原来的 0R 变为 1KR，于是，proteus 又继续正常工作了。

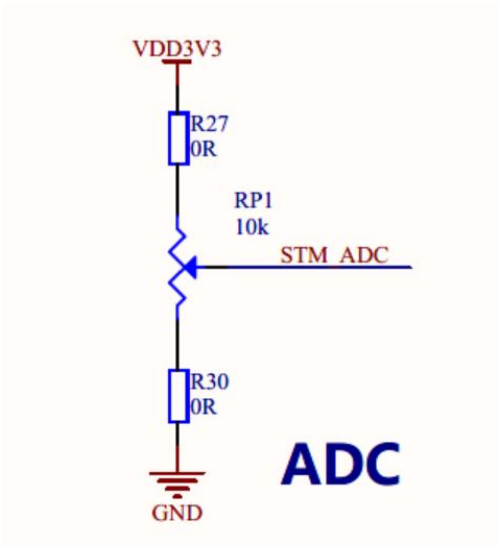

图 21 老师提供的滑动变阻原理图

3.2 检测结果串口打印及数码管显示

3.2.1 代码思路

先通过 printf 函数串口输出气体（滑动变阻）结果，而数码管需要一直循环位选段选输出气体（滑动变阻）值，所以可以先定义一个判断函数，若还在此模块则一直循环输出气体（滑动变阻）值，每个位选之间需要一点延

迟来确保数码管可

Abstract

The report simply records that I and my team members conducted online embedded training in ** Company for up to 6 days from June 15 to June 20.Under the guidance of Teacher ***, on the first day, we installed the required software (Keil5, Proteus) and learned how to use it to build a project.The next day, I learned the use of GPIO port and clock tree, understood the clock source of STM32, the meaning of some statements in STM32F401 startup file, external interrupt, EXIT controller and so on.The third day, we study the GPIO port multiplexing capabilities - a serial port communication, has the GPIO structure and serial structure, can make serial clock and GPIO clock, the GPIO reuse a serial originated, with GPIO port multiplexing function and effective to register, configure a serial port information. And into the register, so that the serial port work, use data to send to verify whether the serial port is working properly, etc.After understanding the physical DHT11, we performed a simple functional design.On the fourth day, I learned the modularization of engineering code, which further encapsulates each function into independent. C language files, and the corresponding functions can be used when adding the independent modularization files to the project, as well as the use of STM32F401 timer.On the fifth day, I learned the timer output PWM wave and the data acquisition of photosensitive sensor as well as the configuration of the sensor, the configuration display of the digital tube and the connection to the corresponding module.On the sixth day, the teacher explained to us some details of the integration module.

Finally, the training ended with the design of three functional modules and the integration of each functional module. After six days of study, we have a deeper understanding of embedment. Hands-on design makes the theory more specific, but unfortunately, we cannot really design and develop on the physical development board.

参考文献