X　X　X　X　大　学

嵌入式系统及应用实习报告

学生姓名：XXX

学　　号：XXXXXXXXXX

专　　业：电气工程及其自动化

班　　级：XXXX

2021年12月

1. **实习地点**

XXXX

1. **实习时间**

2021.12.13-2021.12.26

1. **实习项目要求**

采用学过的STM32微控制器芯片设计一个温度采集报警仪。该温度采集报警仪可以通过DHT11温湿度传感器采集并显示测量到的温度，可以设定温度报警值，温度报警后可以控制蜂鸣器报警。同时该温度采集报警仪时钟功能和远程通信功能，可以显示当前时间，可以将温度值发送至PC的串口调试软件中显示。

1. **必做功能：温度采集报警仪的基本功能（采集与报警）**
   1. 数码管的基本信息显示功能：在数码管上显示自己的学号。
   2. 温湿度采集和显示功能：通过DHT11温湿度传感器采集环境温度，并通过数码管进行显示。
   3. 温度报警功能：环境温度如果超过设定的温度高报警值，蜂鸣器报警。
   4. 温湿度报警值的修改功能：可以通过红外遥控器或按键调整温度高报警值。
2. **选做功能：温度采集报警仪的扩展功能 （时钟与通信）**
   1. 时钟功能：在数码管上显示当前的时间。
   2. 时间修改功能：可以通过红外遥控器或按键调整当前时间。
   3. 串口通信功能：温度采集报警仪通过串口将采集的温度值定时送PC机。
3. **实习项目设计方案**
   1. **项目设计方案描述**

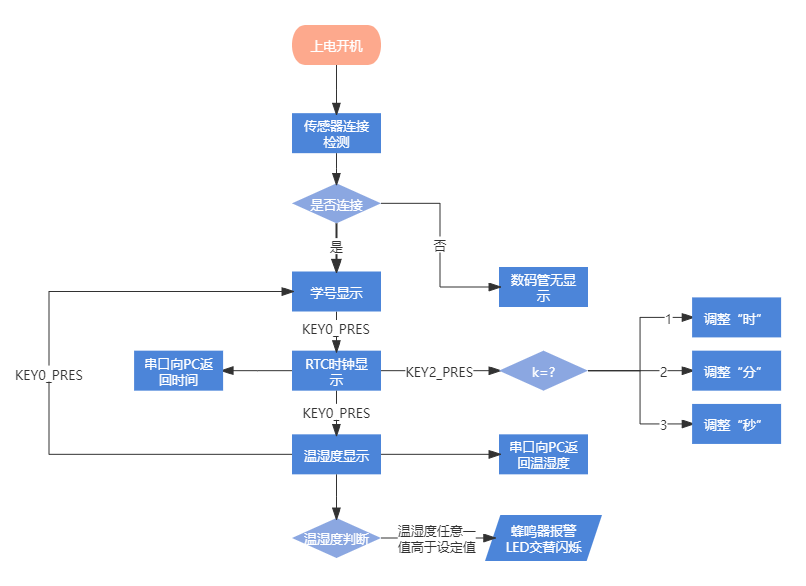


图1 功能流程图

功能：

1. 数码管可显示学号、时间、温湿度，通过按键key0在四个界面切换。
2. 时间界面按键key2可以进入设置模式，key2可以切换想修改的值，通过keyup（+1）和key1（-1）可分别对时、分、秒进行修改，进入设置模式后，可修改的值会闪动，对应的修改“时”（LED5），“分”（LED6），“秒”（LED7）。
3. 温湿度任意一值高于上限会进行声光报警，温湿度界面按键key2可以进入设置模式，key2可以切换想修改的值，通过keyup（+1）和key1（-1）可分别对温度、湿度上限进行修改，进入设置模式后，可修改的值会闪动，对应的修改 “温度”（LED6），“湿度”（LED7）。
4. 串口调试工具XCOM V2.0可根据选择的功能显示时间和温湿度，并在温湿度报警时显示警告。
   1. **硬件部分**

选择ALIENTEK NANO STM32F411开发板

其板载资源包括：

◆ CPU：STM32F411RCT6，LQFP64，FLASH：256K，SRAM：128K；

◆ 外扩SPI FLASH：W25Q16，2M字节

◆ 1个电源指示灯（蓝色）

◆ 7个状态指示灯（DS0-DS7：红色）

◆ 1个红外接收头，并配备一款小巧的红外遥控器

◆ 1个 EEPROM芯片，24C02，容量256字节

◆ 2个4位共阴极数码管（红色）

◆ 1个74HC595D串行转并行数据芯片

◆ 1个74HC138D 3-8译码器芯片

◆ 1路数字温湿度传感器接口，支持DS18B20 /DHT11等

◆ 1个USB JTAG接口，可用于程序下载、软件仿真和串口通信

◆ 1个USB SLAVE接口，用于USB通信

◆ 1个有源蜂鸣器

◆ 1个仿真器功能内/外选择开关

◆ 1个SWD 5P的外部调试下载接口

◆ 1个100K可调电位器

◆ 1组多功能端口（ADC/TPAD）

◆ 1组5V电源供应/接入口

◆ 1组3.3V电源供应/接入口

◆ 1个启动模式选择配置接口

◆ 1个RTC后备电池座，并带电池

◆ 1个复位按钮，可用于复位MCU

◆ 4个功能按钮，其中KEY\_UP兼具唤醒功能

◆ 1个电容触摸按键

◆ 1个电源开关，控制整个板的电源

◆ 除晶振占用的IO口外，部分IO口引出

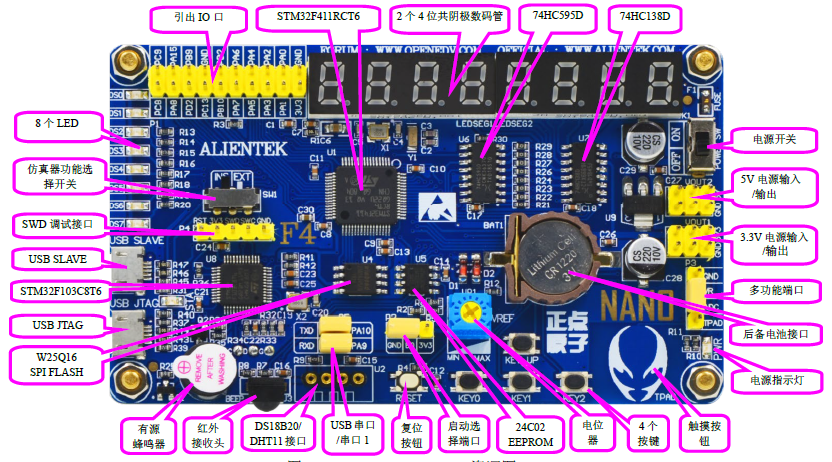


图2 NANO STM32F411资源图

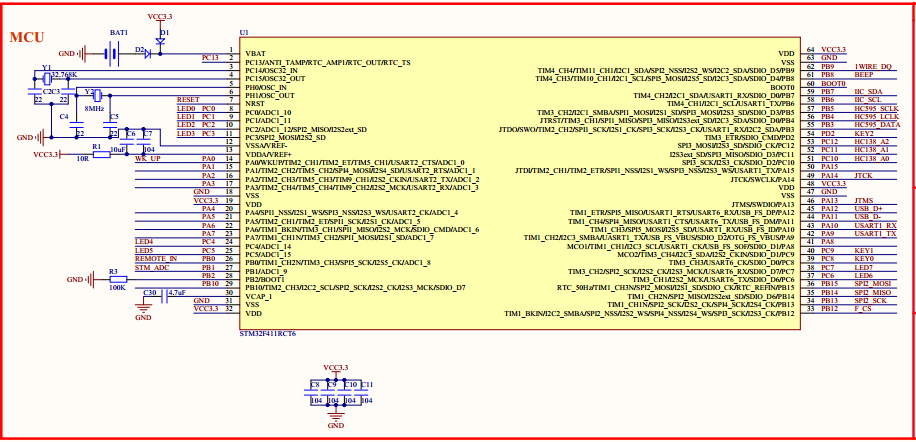


图3 MCU部分原理图

图3为MCU主IO引出口，排针共引出了17个IO口，STM32F411RCT6总共有50个IO，除去RTC晶振占用的2个，还剩48个，由于部分IO口被板载的功能芯片占用了，就没做引出，开发板只通过P1的排针引出了17个IO。

图4中为DHT11数字温湿度传感器原理图。1WIRE\_DQ是传感器的数据线，该信号连接在 MCU 的 PB9 上。

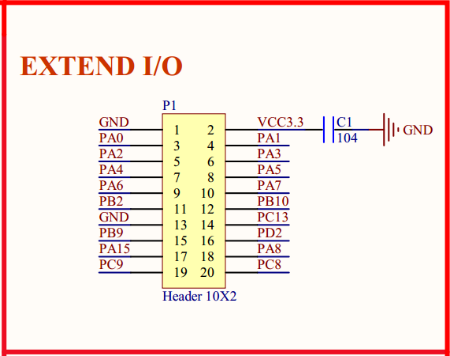
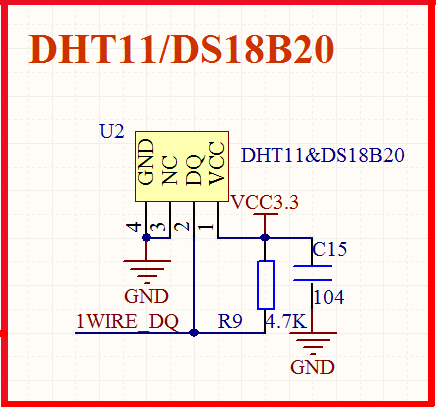
 

图4 引出口原理图 图5 DHT11原理图

ALIENTEK NANO STM32F411板载总共有8个LED，其原理图如图5所示。其中PWR是系统电源指示灯，为蓝色。LED0(DS0)~LED7(DS7)分别接在PC0~PC7上。LED0~LED7灯颜色都为红色。

ALIENTEK NANO STM32F411板载总共有4个输入按键，原理图如图6所示。KEY0、KEY1和KEY2用作普通按键输入，分别连接在PC8、PC9和PD2上，这里并没有使用外部上拉电阻，但是STM32的IO作为输入的时候，可以设置上下拉电阻，所以我们使用STM32的内部上拉电阻来为按键提供上拉。

KEY\_UP按键连接到PA0(STM32的WKUP引脚)，它除了可以用作普通输入按键外，还可以用作STM32的唤醒输入。（该按键为高电平触发）。

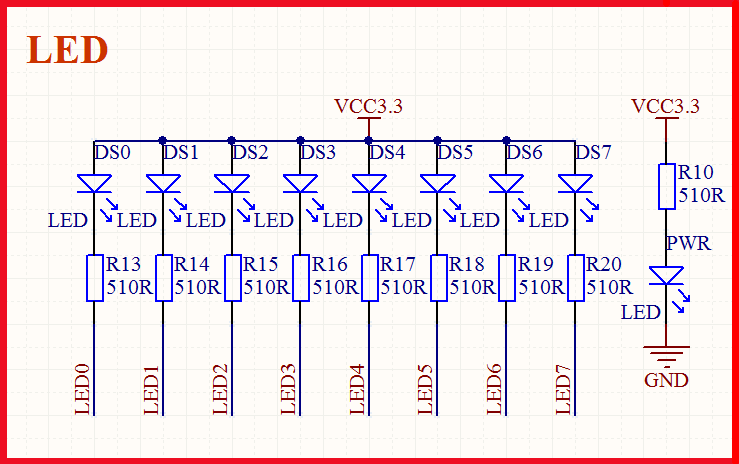
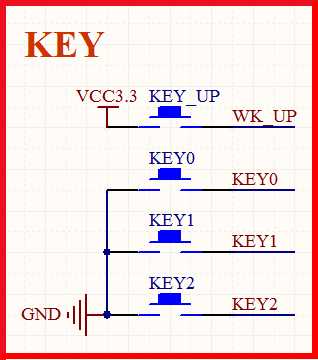
 

图6 LED原理图 图7 按键原理图

图7为两个4位的共阴极数码管LEDSEG1和LEDSEG2以及驱动电路的原理图，CH1~CH4为位选端，A~DOT为段选段。74HC138（U7）为3-8译码器芯片，这里我们使用它实现数码管的位选功能，其Y0~Y7连接到两个数码管的CH1~CH4位选端，通过U7的A0~A2三个地址线的控制，选择Y0~Y7的输出（低电平有效），A0连接在MCU的PC10，A1连接在PC11，A2连接在PC12。74HC595（U6）为串行转并行数据芯片，这里我们使用它实现数码管的段选功能，其QA~QH引脚连接在数码管的HC595\_QA~HC595\_QH公共端，通过SFTCLK和LCHCLK的时钟控制，将SDI输入的串行数据转换成并行数据，通过QA~QH端输出控制数码管的段选，SFTCLK连接在MCU的PB5，LCHCLK连接在PB4，SDI连接在PB3，而RST则连接在MCU的复位电路RESET端，上电或按下RESET按键可对74HC595进行复位。

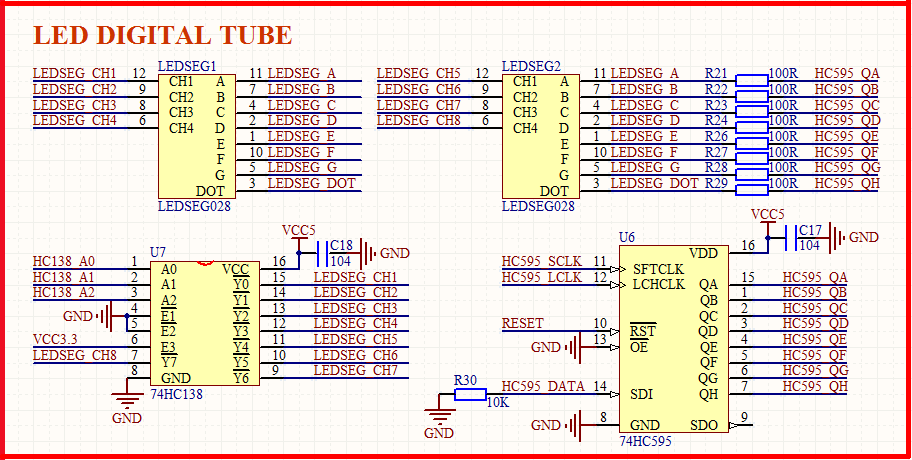


图8 数码管原理图

ALIENTEK NANO STM32F411板载了一个有源蜂鸣器，其原理图如图9所示，有源蜂鸣器是指自带了震荡电路的蜂鸣器，这种蜂鸣器一接上电就会自己震荡发声。图中Q1是用来扩流，R25则是一个限流电阻直接连接在BEEP信号端，BEEP信号端连接在MCU的PB8上面，PB8可以做PWM输出。

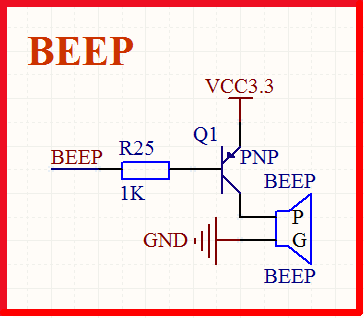


图9 蜂鸣器原理图

* 1. **软件部分**

1. #include "sys.h"
2. #include "delay.h"
3. #include "usart.h"
4. #include "led.h"
5. #include "rtc.h"
6. #include "smg.h"
7. #include "timer.h"
8. #include "usmart.h"
9. #include "key.h"
10. #include "dht11.h"
11. #include "beep.h"
13. //mode3 default num
14. //mode2 time
15. //mode1 temp
16. //mode0 set temp line
18. // 共阴数字数组
19. // 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F, .,全灭
20. u8 smg\_num[]={0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,0xfe,0xf6,0xee,0x3e,0x9c,0x7a,0x9e,0x8e,0x01,0x00};
22. RTC\_TimeTypeDef RTC\_TimeStruct;
23. RTC\_DateTypeDef RTC\_DateStruct;
24. **int** mode=3;
25. **int** c=0;
26. **int** i;
27. //for mode0
28. u8 smg\_wei2=2;//数码管位选
29. u8 num2=0;//数值
31. //for mode1
32. u8 smg\_wei1=2;//数码管位选
33. u8 num1=0;//数值
35. u16 led\_t=0;//led统计时间
36. u16 dht11\_t=0;//dht11采样时间
37. u8 temperature;//温度值
38. u8 humidity;//湿度值
40. //for mode2
41. u8 smg\_wei=0;//数码管位选
42. u8 num=0;//数码管数值
44. u8 time=0;//时间值
45. u8 LEDT=0x03;    //初始化映射LED0 LED1接口  (点灯)
46. u8 LEDF=0X00;    //初始化映射LED接口  (关灯)
48. //设定温湿度报警上限
49. **int** te=30;//温度上限(temperature)
50. **int** hu=80;//湿度上限(humidity)
52. **int** main(**void**)
53. {
54. **int** k=0;
55. vu8 key=0;
56. HAL\_Init();                       //初始化HAL库
57. Stm32\_Clock\_Init(96,4,2,4);     //设置时钟,96Mhz
58. delay\_init(96);                 //初始化延时函数
59. uart\_init(115200);              //初始化串口115200
60. LED\_Init();                     //初始化LED
61. LED\_SMG\_Init();                 //数码管初始化
62. TIM3\_Init(20-1,9600-1);         //数码管2ms定时显示
63. usmart\_dev.init(96);         //初始化USMART
64. KEY\_Init();
65. BEEP\_Init();                    //初始化蜂鸣器端口
67. //RTC状态串口通讯显示
68. printf("RTC TEST\r\n");
69. **while**(RTC\_Init())       //RTC初始化
70. {
71. printf("RTC ERROR!\r\n");
72. delay\_ms(800);
73. printf("RTC Trying...\r\n");
74. }
75. RTC\_Set\_WakeUp(RTC\_WAKEUPCLOCK\_CK\_SPRE\_16BITS,0); //配置WAKE UP中断,1秒钟中断一次
76. printf("RTC OK\r\n");
78. //DHT11状态串口通讯显示
79. printf("DHT11 TEST\r\n");
80. **while**(DHT11\_Init()) //DHT11初始化
81. {
82. printf("DHT11 Error\r\n");
83. delay\_ms(200);
84. printf("DHT11 Trying...\r\n");
85. }
86. printf("DHT11 OK\r\n");

89. **while**(1)
90. {
92. //按键判断
93. key=KEY\_Scan(0);
94. **switch**(key)
95. {
96. **case** KEY2\_PRES://改变所调时间位数
97. **if**(mode==2)
98. {
99. k++;
100. **switch**(k)
101. {
102. **case** 1:
103. LED5=!LED5;//亮起时为调节"时"
104. **break**;
105. **case** 2:
106. LED5=!LED5;
107. LED6=!LED6;//亮起时为调节"分"
108. **break**;
109. **case** 3:
110. LED6=!LED6;
111. LED7=!LED7;//亮起时为调节"秒"
112. **break**;
113. **case** 4:
114. k=0;
115. LED7=!LED7;//全部熄灭锁定调节
116. **break**;
117. }
118. }
119. **if**(mode==0)
120. {
121. k++;
122. **switch**(k)
123. {
124. **case** 1:
125. LED6=!LED6;//亮起时为调节“温度”
126. **break**;
127. **case** 2:
128. LED6=!LED6;
129. LED7=!LED7;//亮起时为调节“湿度”
130. **break**;
131. **case** 3:
132. k=0;
133. LED7=!LED7;//全部熄灭锁定调节
134. **break**;
135. }
136. }
137. **break**;
138. **case** KEY1\_PRES:
140. //时间调整
141. **if**(mode==2)
142. {
143. **if**(k==1) //"时"减少模块
144. {
145. **if**(RTC\_TimeStruct.Hours==0)
146. {
147. RTC\_Set\_Time(23,RTC\_TimeStruct.Minutes,RTC\_TimeStruct.Seconds,0);
148. }
149. **else**
150. {
151. RTC\_TimeStruct.Hours--;
152. RTC\_Set\_Time(RTC\_TimeStruct.Hours,RTC\_TimeStruct.Minutes,RTC\_TimeStruct.Seconds,0);
153. }
154. }
155. **if**(k==2) //"分"减少模块
156. {
157. **if**(RTC\_TimeStruct.Minutes==0)
158. {
159. RTC\_Set\_Time(RTC\_TimeStruct.Hours,59,RTC\_TimeStruct.Seconds,0);
160. }
161. **else**
162. {
163. RTC\_TimeStruct.Minutes--;
164. RTC\_Set\_Time(RTC\_TimeStruct.Hours,RTC\_TimeStruct.Minutes,RTC\_TimeStruct.Seconds,0);
165. }
166. }
167. **if**(k==3) //"秒"减少模块
168. {
169. **if**(RTC\_TimeStruct.Seconds==0)
170. {
171. RTC\_Set\_Time(RTC\_TimeStruct.Hours,RTC\_TimeStruct.Minutes,59,0);
172. }
173. **else**
174. {
175. RTC\_TimeStruct.Seconds--;
176. RTC\_Set\_Time(RTC\_TimeStruct.Hours,RTC\_TimeStruct.Minutes,RTC\_TimeStruct.Seconds,0);
177. }
178. }
179. }
181. //温度报警值调整
182. **if**(mode==0)
183. {
184. **if**(k==1) //"温度"减少模块
185. {
186. te--;
187. }
188. **if**(k==2) //"湿度"减少模块
189. {
190. hu--;
191. }
192. }
193. **break**;
195. **case** WKUP\_PRES:
197. //时间调整
198. **if**(mode==2)
199. {
200. **if**(k==1) //"时"增加模块
201. {
202. **if**(RTC\_TimeStruct.Hours==23)
203. {
204. RTC\_Set\_Time(0,RTC\_TimeStruct.Minutes,RTC\_TimeStruct.Seconds,0);
205. }
206. **else**
207. {
208. RTC\_TimeStruct.Hours++;
209. RTC\_Set\_Time(RTC\_TimeStruct.Hours,RTC\_TimeStruct.Minutes,RTC\_TimeStruct.Seconds,0);
210. }
211. }
212. **if**(k==2) //"分"增加模块
213. {
214. **if**(RTC\_TimeStruct.Minutes==59)
215. {
216. RTC\_Set\_Time(RTC\_TimeStruct.Hours,0,RTC\_TimeStruct.Seconds,0);
217. }
218. **else**
219. {
220. RTC\_TimeStruct.Minutes++;
221. RTC\_Set\_Time(RTC\_TimeStruct.Hours,RTC\_TimeStruct.Minutes,RTC\_TimeStruct.Seconds,0);
222. }
223. }
224. **if**(k==3) //"秒"增加模块
225. {
226. **if**(RTC\_TimeStruct.Seconds==59)
227. {
228. RTC\_Set\_Time(RTC\_TimeStruct.Hours,RTC\_TimeStruct.Minutes,0,0);
229. }
230. **else**
231. {
232. RTC\_TimeStruct.Seconds++;
233. RTC\_Set\_Time(RTC\_TimeStruct.Hours,RTC\_TimeStruct.Minutes,RTC\_TimeStruct.Seconds,0);
234. }
235. }
236. }
238. //温度报警值调整
239. **if**(mode==0)
240. {
241. **if**(k==1) //"温度"增加模块
242. {
243. te++;
244. }
245. **if**(k==2) //"湿度"增加模块
246. {
247. hu++;
248. }
249. }
250. **break**;
252. //模式切换
253. **case** KEY0\_PRES:
254. c++;
255. **if**(c==4)
256. {
257. c=0;
258. }
259. **switch**(c)
260. {
261. **case** 0 : //switch to num
262. k=0;
263. LED0=1;
264. LED1=1;
265. LED2=1;
266. LED5=1;
267. LED6=1;
268. LED7=1;
269. mode=3;
270. **break**;
272. **case** 1 : //switch to time
273. k=0;
274. LED0=1;
275. LED2=1;
276. mode=2;
277. **break**;
279. **case** 2 : //switch to temp
280. k=0;
281. LED0=1;
282. LED1=1;
283. LED5=1;
284. LED6=1;
285. LED7=1;
286. mode=1;
287. **break**;
289. **case** 3 : //switch to temp  line set
290. k=0;
291. LED0=1;
292. LED1=1;
293. LED2=1;
294. LED5=1;
295. LED6=1;
296. LED7=1;
297. mode=0;
298. **break**;
300. }
301. **break**;
302. }
304. //学号显示
305. **if**(mode==3)//溢出中断
306. {
307. smg\_wei=0;
308. LED\_Write\_Data(smg\_num[2],smg\_wei);//数码管显示
309. LED\_Refresh();
310. smg\_wei++;
311. LED\_Write\_Data(smg\_num[1],smg\_wei);
312. LED\_Refresh();
313. smg\_wei++;
314. LED\_Write\_Data(smg\_num[0],smg\_wei);
315. LED\_Refresh();
316. smg\_wei++;
317. LED\_Write\_Data(smg\_num[2],smg\_wei);
318. LED\_Refresh();
319. smg\_wei++;
320. LED\_Write\_Data(smg\_num[4],smg\_wei);
321. LED\_Refresh();
322. smg\_wei++;
323. LED\_Write\_Data(smg\_num[3],smg\_wei);
324. LED\_Refresh();
325. smg\_wei++;
326. LED\_Write\_Data(smg\_num[1],smg\_wei);
327. LED\_Refresh();
328. smg\_wei++;
329. LED\_Write\_Data(smg\_num[7],smg\_wei);
330. LED\_Refresh();
331. smg\_wei++;
332. **if**(smg\_wei==8) smg\_wei=0;
333. }
335. //温湿度警报（流水灯）与串口通讯
336. **if**((temperature>=te||humidity>=hu)&&mode==1)
337. {
338. printf("警告!温湿度过高!(温度高于%d,湿度高于%d)\r\n",te,hu);
339. **for**(i=0;i<7;i++)//正向循环
340. {
341. HAL\_GPIO\_WritePin(GPIOC,LEDT,GPIO\_PIN\_RESET);//点亮指定LED(置0)
342. HAL\_GPIO\_WritePin(GPIOC,LEDF,GPIO\_PIN\_SET);//熄灭指定LED    (置1)
343. HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_8,GPIO\_PIN\_RESET); //BEEP引脚拉低，响， 等同BEEP=0;
344. delay\_ms(100);//延时100 ms
345. HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_8,GPIO\_PIN\_SET);   //BEEP引脚拉高，不响，等同BEEP=1;
346. delay\_ms(100);//延时100 ms
347. LEDT<<=1;//左移指向下一个地址
348. LEDF<<=1;//左移指向下一个地址
349. LEDF++;
350. }
351. HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_8,GPIO\_PIN\_RESET); //BEEP引脚拉低，响， 等同BEEP=0;
352. delay\_ms(100);//延时100 ms
353. HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_8,GPIO\_PIN\_SET);   //BEEP引脚拉高，不响，等同BEEP=1;
354. delay\_ms(100);//延时100 ms
355. LEDT=0x60;//映射到LED5 LED6
356. LEDF=0x80;//映射到LED7
357. **for**(i=0;i<6;i++)//逆向循环
358. {
359. HAL\_GPIO\_WritePin(GPIOC,LEDT,GPIO\_PIN\_RESET);//点亮指定LED
360. HAL\_GPIO\_WritePin(GPIOC,LEDF,GPIO\_PIN\_SET);//熄灭指定LED
361. HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_8,GPIO\_PIN\_RESET); //BEEP引脚拉低，响， 等同BEEP=0;
362. delay\_ms(100);//延时100 ms
363. HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_8,GPIO\_PIN\_SET);   //BEEP引脚拉高，不响，等同BEEP=1;
364. delay\_ms(100);//延时100 ms
365. LEDT>>=1;//右移指向下一个地址
366. LEDF>>=1;//右移指向下一个地址
367. }
368. LEDT=0x03;
369. LEDF=0X00;//初始化所有LED映射准备下一循环
370. }
371. **else**
372. {
373. HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_8,GPIO\_PIN\_SET);
374. LED1=1;  //重置警报灯与蜂鸣器
375. }
376. }
377. }
379. //RTC时间显示与串口通讯
380. u16 print=0;//输出计时
381. **void** HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim) //回调函数，定时器中断服务函数调用
382. {
384. **if**(htim==(&TIM3\_Handler)&&mode==2)
385. {
386. print++;
387. **if**(print==500)
388. {
389. printf("当前的时间是：%d点%d分%d秒\r\n",RTC\_TimeStruct.Hours,RTC\_TimeStruct.Minutes,RTC\_TimeStruct.Seconds);
390. print=0;
391. }
392. HAL\_RTC\_GetTime(&RTC\_Handler,&RTC\_TimeStruct,RTC\_FORMAT\_BIN);
393. HAL\_RTC\_GetDate(&RTC\_Handler,&RTC\_DateStruct,RTC\_FORMAT\_BIN);
394. **switch**(smg\_wei)
395. {
396. **case** 0:  num = smg\_num[RTC\_TimeStruct.Hours/10]; **break**;//时
397. **case** 1:  num = smg\_num[RTC\_TimeStruct.Hours%10]; **break**;
398. **case** 2:  num = 0x02; **break**;
399. **case** 3:  num = smg\_num[RTC\_TimeStruct.Minutes/10]; **break**; //分
400. **case** 4:  num = smg\_num[RTC\_TimeStruct.Minutes%10]; **break**;
401. **case** 5:  num = 0x02; **break**;
402. **case** 6:  num = smg\_num[RTC\_TimeStruct.Seconds/10]; **break**; //秒
403. **case** 7:  num = smg\_num[RTC\_TimeStruct.Seconds%10]; **break**;
404. }
405. LED\_Write\_Data(num,smg\_wei);//写数据到数码管
406. LED\_Refresh();//更新显示
407. smg\_wei++;
408. **if**(smg\_wei==8) smg\_wei=0;
409. }
411. //温湿度串口通讯显示
412. **if**(htim==(&TIM3\_Handler)&&mode==1)
413. {
414. print++;
415. **if**(print==500)
416. {
417. printf("当前温度为：%d 湿度为%d(当前预警值：%d %d)\r\n",temperature,humidity,te,hu);
418. print=0;
419. }
420. dht11\_t++;
421. **if**(dht11\_t==500)//DHT11 1S采样
422. {
423. dht11\_t=0;
424. DHT11\_Read\_Data(&temperature,&humidity);//读取温湿度值
425. }
426. **switch**(smg\_wei1)
427. {
428. **case** 2:  num1 = smg\_num[temperature/10]; **break**;//温度值
429. **case** 3:  num1 = smg\_num[temperature%10];**break**;
430. **case** 6:  num1 = smg\_num[humidity/10]; **break**;//湿度值
431. **case** 7:  num1 = smg\_num[humidity%10];**break**;
432. }
433. LED\_Write\_Data(num1,smg\_wei1);//写数据到数码管
434. LED\_Refresh();//更新显示
435. smg\_wei1++;
437. //数码管位数跳变
438. **if**(smg\_wei1==4)  smg\_wei1=6;
439. **if**(smg\_wei1==8)  smg\_wei1=2;
440. led\_t++;
441. **if**(led\_t==250)//LED0每500ms闪烁
442. {
443. **if**(temperature<30&&humidity<80)
444. {
445. led\_t=0;
446. LED0=!LED0;
447. }
448. led\_t=0;
449. }
450. }
452. //温湿度报警值显示
453. **if**(mode==0)
454. {
455. **switch**(smg\_wei2)
456. {
457. **case** 2:  num2 = smg\_num[te/10]; **break**;//温度值
458. **case** 3:  num2 = smg\_num[te%10];**break**;
459. **case** 6:  num2 = smg\_num[hu/10]; **break**;//湿度值
460. **case** 7:  num2 = smg\_num[hu%10];**break**;
461. }
462. LED\_Write\_Data(num2,smg\_wei2);//写数据到数码管
463. LED\_Refresh();//更新显示
464. smg\_wei2++;
466. //数码管位数跳变
467. **if**(smg\_wei2==4)  smg\_wei2=6;
468. **if**(smg\_wei2==8)  smg\_wei2=2;
469. }
470. }
471. **实习效果**
     1. **实物展示**

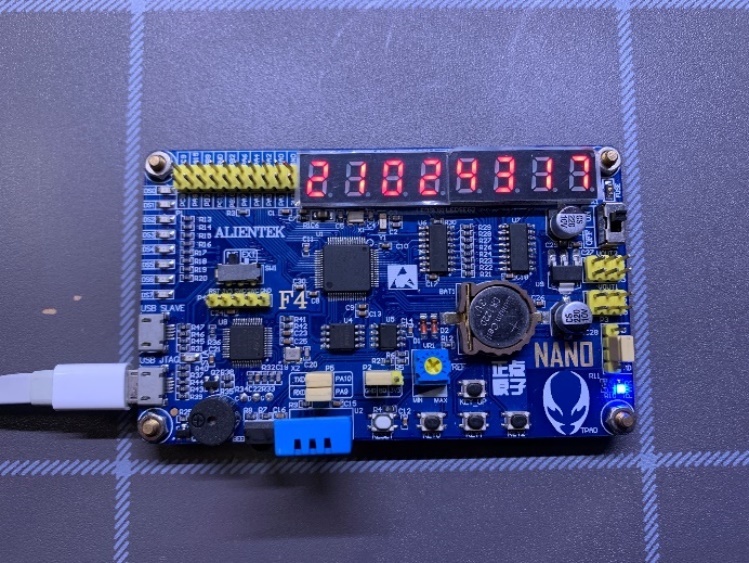
****

图10 学号显示

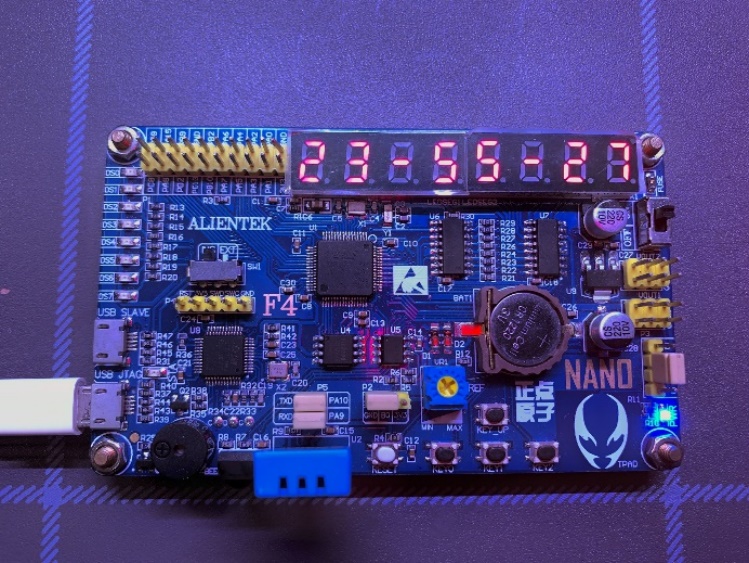
****

图11 时间显示

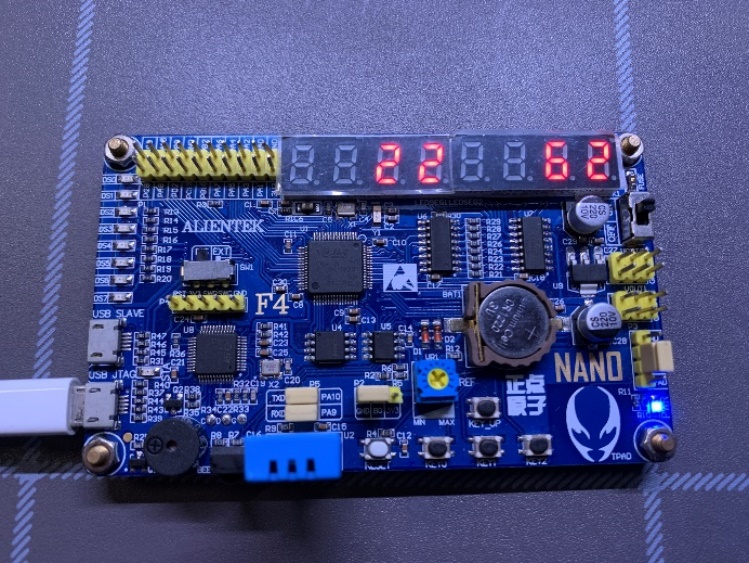
****

图12 温湿度显示

****

图13 温湿度报警值修改

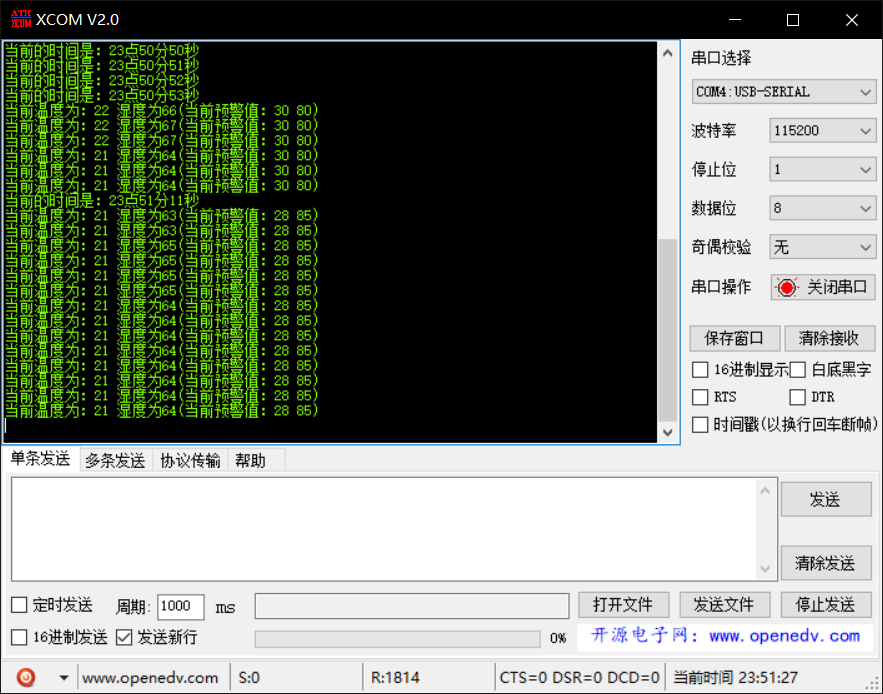


图14 PC端串口显示

* 1. **设计结果评价**

在本程序设计中，数码管能正确显示预设数字（学号），可以在修改时间和温湿度报警值时有对应的位数闪烁提示，且能实现对于过高的温、湿度的声光报警，完成了既定目标，程序设计成功。