检测技术与计量校准课程设计报告

叶开

1. 任务需求

任务名称为：DS18B20 数字温度计设计

任务分为两部分，基本任务和扩展功能：

基本任务要求

\* 能实现测温范围-5~125度，误差+0.1度;

\* 能实现信号实时数据与历史曲线显示;

\* 能用按键设置控制的期望值范围，超出声光报警。

扩展功能

\* 具有与计算机数据通讯和传送功能;

\* 能设置PI控制，并计算输出控制;

\* 能进行位式或连续温度调节;

\* 能进行温度无线传输显示。

1. 任务分析

基本一为读取ds18b20传感器数据，根据ds18b20的onewire通讯协议读取，较为简单。

基本二为在LCD上显示数值。

基本三为按键输入、蜂鸣器输出等GPIO的控制。

扩展一为串口外设的使用。

扩展二为PID算法的设计与PWM的输出。

扩展三为程序的设计。

扩展四为无线方案的选择与设计。

1. 软硬件方案选型

单片机方面我选择ESP32。ESP32是一颗集成了2.4G射频的无线高性能SoC，它有两个基于Xtensa® 32-bit LX6的内核，可运行在240MHz上。它还有着较为丰富的外设，自带2.4G射频可方便地完成扩展四。同时价格相较于STM32等ARM单片机更为廉价，性能相较于8051内核单片机更高。

单片机编程框架我选择arduino。乐鑫官方为ESP32制作了一套基于esp官方框架esp-idf的arduino兼容层，使我们能够方便地使用arduino的软件库，同时也能方便地调用esp-idf的底层函数。同时我选择platformIO+vscode作为本次开发的开发环境。PlatformIO是一个专业的嵌入式开发环境，能自动地根据已经选择好的芯片和框架和自行编写的配置文件，选择和下载配置适合的软件库、框架、工具链、下载工具等工具，相比于arduinoIDE更方便开发。VSCode是是一款由微软开发且跨平台的免费源代码编辑器。相比于Keil、IAR等传统的嵌入式IDE，VSCode小巧，却又在插件的帮助下有强大的功能。在 2019 年的 Stack Overflow 组织的开发者调查中，Visual Studio Code 被认为是最受开发者欢迎的开发环境。

版本控制软件我选择git。Git 是一个开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。Vscode和Github Desktop也能很好地支持git，让我进行图形化地提交、管理、回滚版本等操作。

图像显示我选择了一块2.4英寸的TFT-LCD屏幕。这块屏幕使用ST7789驱动芯片，SPI的驱动方式，可以使用TFT\_eSPI库进行方便地驱动。由于预算的限制，我只选择了2.4英寸的屏幕大小，显示内容稍显拥挤，但能显示任务所需内容。

GUI框架我选择了LVGL。LVGL是一个免费和开源的图形库，它提供了创建嵌入式GUI所需的一切，具有易于使用的图形元素，美丽的视觉效果和低内存占用。同时可以使用NXP的GUI-Guider方便的设计GUI生成代码。

无线传输方案我选择了ESP-Now。ESP-NOW 是由乐鑫开发的另一款协议，可以使多个设备在没有或不使用 Wi-Fi 的情况下进行通信。这种协议类似常见于无线鼠标中的低功耗 2.4GHz 无线连接——设备在进行通信之前要进行配对。配对之后，设备之间的连接是持续的、点对点的，并且不需要握手协议。ESP-Now能方便地让我在两个ESP32之间无线传输温度数据。

开源许可证方面我选择GPLv3。由于部分引用的库的许可证为GPLv3，故选择GPLv3。

1. 程序框架设计

程序基于arduino框架，包含了FreeRTOS。这次课程设计，我大体上设计了以下线程：LVGL心跳线程、按键扫描线程、传感器数据读取线程，以及在主线程中根据flag判断报警或是计算pid。

主线程框架：

图示

描述已自动生成

按键扫描线程：

图示

描述已自动生成

传感器读取和LVGL心跳线程较为简单，仅为执行任务后延时。

从机程序较为简单，为ESP-NOW接收主机发来的数据配合u8g2库显示在OLED上。

1. GUI界面设计

GUI主要分为3页，主界面、报警设置界面和PID参数设置界面：

图形用户界面

描述已自动生成

图形用户界面

描述已自动生成图形用户界面

描述已自动生成

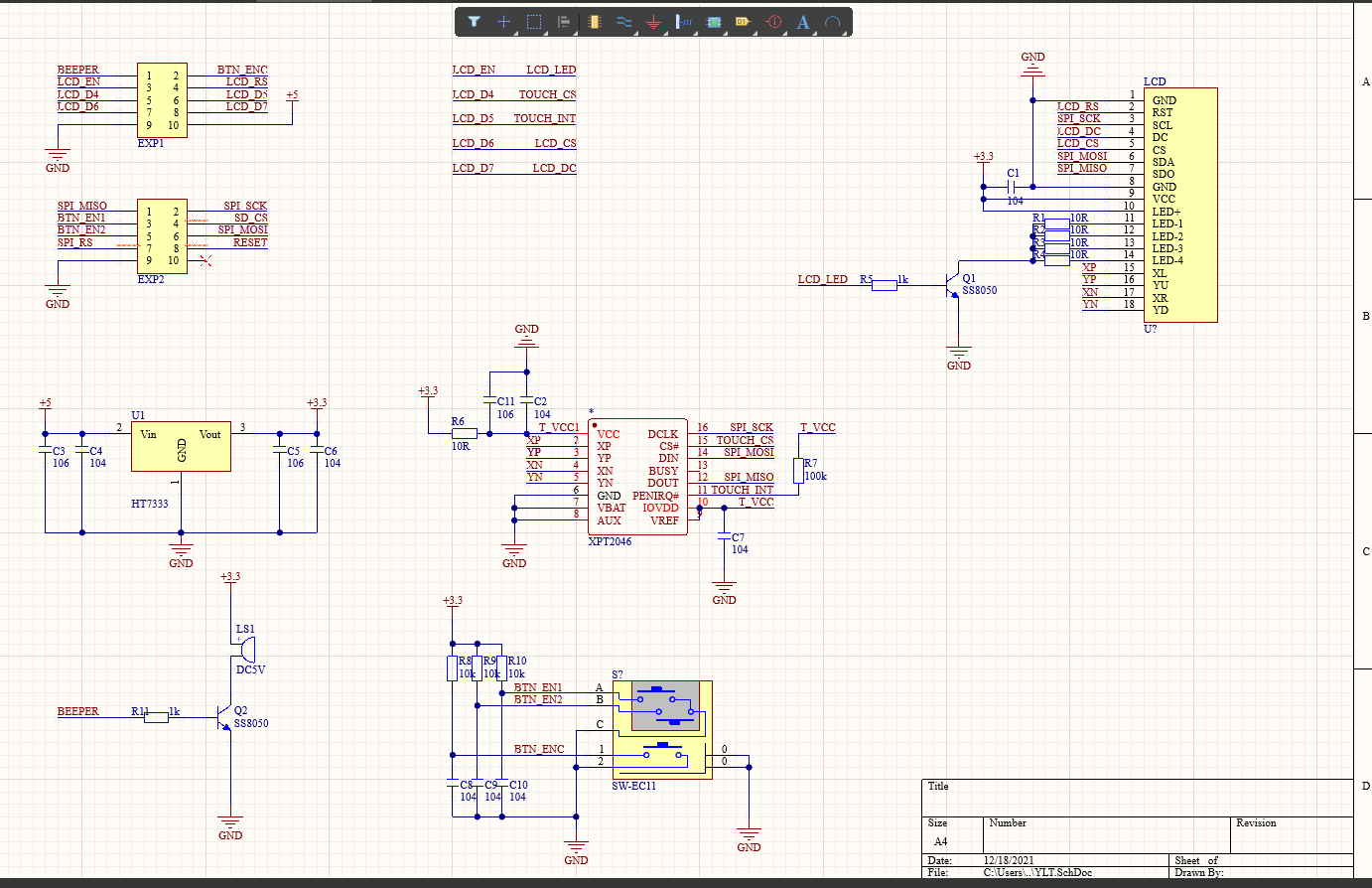
1. 主要连线

由于调试时间较为不充裕，故我并没有选择打板，而是使用洞洞板进行锡线连接。主要连线如下：

图示, 示意图

描述已自动生成

TFT屏幕为我之前给3D打印机设计的底板配上裸屏，由于要求使用按键，本次仅用到了屏幕而暂未使用触摸和旋转编码器，原理图如图：



热床为立创开源团队GeekStart作品，在此表示感谢。

文本, QR 代码

描述已自动生成

1. 遇到的问题及（不完美的）解决方案

1、热床电阻过低导致电源保护：串联水泥电阻增大电阻、改用5V电源代替12V电源。

2、可能由于电源纹波等质量问题，在pwm输出较为波动时刷新屏幕可能造成刷新错误等花屏现象：减缓了pwm调整周期，从10ms延长至200ms。

1. 体会及收获

在短暂的时间内进行一个完整项目的开发，这是一次很好的锻炼。尽管我更熟悉的似乎还是STM32系列单片机，尽管我并没有完美地解决一些问题，尽管因为种种原因充满了妥协，还是跌跌撞撞地在两三天里面调完了程序。虽然队友免掉课设而我并没有，没有了团队合作，但全栈工程师一个人就是一支队伍，不是吗？快速开发的经历让我深刻体会到了平时的积累和快速读懂文档的重要性，我以后也会朝着这些方向努力。当然，我能这么快速地开发出来还是靠开源大佬们写的库，我也在努力为开源世界添砖加瓦。

1. 参考资料

[1] LVGL LLC, documentation of LVGL, [DB/OL] <https://docs.lvgl.io/7.11/>

[2] NXP Semiconductors, GUI Guider User's Guide [DB/OL] GUI-Guider安装目录下

[3] PlatformIO, documentation of PlatformIO, [DB/OL] https://docs.platformio.org/en/latest/

[4] espressif esp-idf github.com[DB/OL] <https://github.com/espressif/esp-idf>

[5] espressif Arduino-esp32 github.com[DB/OL] https://github.com/espressif/arduino-esp32

[6] espressif esp32技术规格书 [DB/OL] <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_cn.pdf>

[7] espressif esp32技术参考手册 [DB/OL] <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_cn.pdf>

[8] milesburton Arduino-Temperature-Control-Library github.com[DB/OL] https://github.com/milesburton/Arduino-Temperature-Control-Library.git

[9] PaulStoffregen OneWire github.com[DB/OL] https://github.com/PaulStoffregen/OneWire

[10] bodmer TFT\_eSPI github.com[DB/OL] https://github.com/bodmer/TFT\_eSPI

1. 附录

一些代码：

Main.cpp:

1. **void** setup()
2. {
3. alarm\_value = 50;
4. alarm\_flag = **false**;
5. kp = 1;
6. ki = 1;
7. target = 0;
8. pid\_flag = **false**;
9. i\_enabled\_flag = **true**;
11. Beeper\_Init();
12. Heater\_Init();
13. PID\_Init();
15. Serial.begin(115200);
16. // Serial.println("HelloWorld");
18. GUI\_Init();
19. setup\_ui(&guider\_ui);
20. screen\_chart\_1\_0 = lv\_chart\_add\_series(guider\_ui.screen\_chart\_1, lv\_color\_make(0x00, 0x00, 0x00));
21. xTaskCreate(GUI\_Run, "GUI\_Task", 4096 \* 4, NULL, 2, NULL);
23. Btn\_Init();
24. xTaskCreate(Btn\_Scan, "Btn\_Scan\_Task", 4096, NULL, 3, NULL);
26. ESPNow\_Init();
28. Sensor\_Init();
29. xTaskCreate(Sensor\_Read, "Sensor\_Read\_Task", 4096, NULL, 1, NULL);
30. }
32. **void** loop()
33. {
34. **if** (alarm\_flag && (temperature > alarm\_value))
35. {
36. Beeper\_On();
37. }
38. **else**
39. {
40. Beeper\_Off();
41. }
43. **if** (pid\_flag)
44. {
45. auto heat = PID\_Calculate(temperature);
46. **if** (heat < 0)
47. {
48. heat = 0;
49. }
50. **else** **if** (heat > 100)
51. {
52. heat = 100;
53. }
54. Heater\_SetDuty(heat);
55. **char** ch[30];
56. sprintf(ch, "pidout:%d", **int**(heat));
57. lv\_label\_set\_text(guider\_ui.screen\_state, ch);
58. }
59. **else**
60. {
61. Heater\_SetDuty(0);
62. }
64. esp\_task\_wdt\_reset();
65. delay(200);
66. }

Btn.cpp:

1. **void** Btn\_Init()
2. {
3. pinMode(BTN\_1\_Pin, INPUT\_PULLUP);
4. pinMode(BTN\_2\_Pin, INPUT\_PULLUP);
5. pinMode(BTN\_3\_Pin, INPUT\_PULLUP);
6. }
8. **void** Btn\_Scan(**void** \*parm)
9. {
10. **int** btn1, btn2, btn3;
11. **int** pid\_set\_state = 0;
12. **int** pid\_kp, pid\_ki, pid\_target;
13. **while** (**true**)
14. {
15. btn1 = digitalRead(BTN\_1\_Pin);
16. btn2 = digitalRead(BTN\_2\_Pin);
17. btn3 = digitalRead(BTN\_3\_Pin);
18. **if** ((btn1 == LOW) || (btn2 == LOW) || (btn3 == LOW))
19. {
20. delay(10);
21. btn1 = digitalRead(BTN\_1\_Pin);
22. btn2 = digitalRead(BTN\_2\_Pin);
23. btn3 = digitalRead(BTN\_3\_Pin);
24. Serial.printf("btns' states : %d %d %d\n", btn1, btn2, btn3);
25. **if** ((btn1 == LOW) || (btn2 == LOW) || (btn3 == LOW))
26. {
27. auto now\_scr = lv\_scr\_act();
28. **if** (now\_scr == guider\_ui.screen)
29. {
30. **if** (btn1 == LOW) // 跳转到alarm
31. {
32. lv\_scr\_load(guider\_ui.alarm);
33. alarm\_value = **int**(temperature);
34. lv\_linemeter\_set\_value(guider\_ui.alarm\_lmeter\_1, alarm\_value);
35. **char** ch[30];
36. sprintf(ch, "%d", alarm\_value);
37. lv\_label\_set\_text(guider\_ui.alarm\_set\_temperature, ch);
38. alarm\_flag = **false**;
39. pid\_flag = **false**;
40. }
41. **else** **if** (btn2 == LOW) // 跳转到pid
42. {
43. // Serial.printf("goto pid\n");
44. lv\_scr\_load(guider\_ui.pid);
45. pid\_set\_state = 0;
46. pid\_kp = 1;
47. pid\_ki = 1;
48. pid\_target = **int**(temperature);
49. **char** ch[30];
50. sprintf(ch, "%d", pid\_kp);
51. lv\_textarea\_set\_text(guider\_ui.pid\_P\_value, ch);
52. sprintf(ch, "%d", pid\_ki);
53. lv\_textarea\_set\_text(guider\_ui.pid\_I\_value, ch);
54. sprintf(ch, "%d", pid\_target);
55. lv\_textarea\_set\_text(guider\_ui.pid\_target, ch);
56. lv\_textarea\_set\_cursor\_hidden(guider\_ui.pid\_P\_value, **false**);
57. lv\_textarea\_set\_cursor\_hidden(guider\_ui.pid\_I\_value, **true**);
58. lv\_textarea\_set\_cursor\_hidden(guider\_ui.pid\_target, **true**);
59. alarm\_flag = **false**;
60. pid\_flag = **false**;
61. }
62. }
63. **else** **if** (now\_scr == guider\_ui.alarm)
64. {
65. // not tested yet
66. **if** (btn1 == LOW) // 减小报警数值
67. {
68. alarm\_value--;
69. lv\_linemeter\_set\_value(guider\_ui.alarm\_lmeter\_1, alarm\_value);
70. **char** ch[30];
71. sprintf(ch, "%d", alarm\_value);
72. lv\_label\_set\_text(guider\_ui.alarm\_set\_temperature, ch);
73. }
74. **else** **if** (btn2 == LOW) // 增加报警数值
75. {
76. alarm\_value++;
77. lv\_linemeter\_set\_value(guider\_ui.alarm\_lmeter\_1, alarm\_value);
78. **char** ch[30];
79. sprintf(ch, "%d", alarm\_value);
80. lv\_label\_set\_text(guider\_ui.alarm\_set\_temperature, ch);
81. }
82. **else** **if** (btn3 == LOW) // 保存返回
83. {
84. alarm\_flag = **true**;
85. lv\_scr\_load(guider\_ui.screen);
86. }
87. }
88. **else** **if** (now\_scr == guider\_ui.pid)
89. {
90. **if** (btn1 == LOW) // 减小数值
91. {
92. **switch** (pid\_set\_state)
93. {
94. **case** 0:
95. pid\_kp >= 1 ? pid\_kp-- : pid\_kp;
96. **break**;
97. **case** 1:
98. pid\_ki >= 1 ? pid\_ki-- : pid\_ki;
99. **break**;
100. **case** 2:
101. pid\_target >= 1 ? pid\_target-- : pid\_target;
102. **break**;
103. **default**:
104. **break**;
105. }
106. **char** ch[30];
107. sprintf(ch, "%d", pid\_kp);
108. lv\_textarea\_set\_text(guider\_ui.pid\_P\_value, ch);
109. sprintf(ch, "%d", pid\_ki);
110. lv\_textarea\_set\_text(guider\_ui.pid\_I\_value, ch);
111. sprintf(ch, "%d", pid\_target);
112. lv\_textarea\_set\_text(guider\_ui.pid\_target, ch);
113. }
114. **else** **if** (btn2 == LOW) // 增加数值
115. {
116. **switch** (pid\_set\_state)
117. {
118. **case** 0:
119. pid\_kp <= 100 ? pid\_kp++ : pid\_kp;
120. **break**;
121. **case** 1:
122. pid\_ki <= 100 ? pid\_ki++ : pid\_ki;
123. **break**;
124. **case** 2:
125. pid\_target <= 100 ? pid\_target++ : pid\_target;
126. **break**;
127. **default**:
128. **break**;
129. }
130. **char** ch[30];
131. sprintf(ch, "%d", pid\_kp);
132. lv\_textarea\_set\_text(guider\_ui.pid\_P\_value, ch);
133. sprintf(ch, "%d", pid\_ki);
134. lv\_textarea\_set\_text(guider\_ui.pid\_I\_value, ch);
135. sprintf(ch, "%d", pid\_target);
136. lv\_textarea\_set\_text(guider\_ui.pid\_target, ch);
137. }
138. **else** **if** (btn3 == LOW) // 下一项/保存返回
139. {
140. **switch** (pid\_set\_state)
141. {
142. **case** 0:
143. pid\_set\_state++;
144. lv\_textarea\_set\_cursor\_hidden(guider\_ui.pid\_P\_value, **true**);
145. lv\_textarea\_set\_cursor\_hidden(guider\_ui.pid\_I\_value, **false**);
146. lv\_textarea\_set\_cursor\_hidden(guider\_ui.pid\_target, **true**);
147. **break**;
148. **case** 1:
149. pid\_set\_state++;
150. lv\_textarea\_set\_cursor\_hidden(guider\_ui.pid\_P\_value, **true**);
151. lv\_textarea\_set\_cursor\_hidden(guider\_ui.pid\_I\_value, **true**);
152. lv\_textarea\_set\_cursor\_hidden(guider\_ui.pid\_target, **false**);
153. **break**;
154. **case** 2:
155. pid\_set\_state = 0;
157. kp = pid\_kp;
158. ki = pid\_ki;
159. target = pid\_target;
160. PID\_Reset();
161. lv\_scr\_load(guider\_ui.screen);
162. esp\_task\_wdt\_reset();
163. delay(100);
164. pid\_flag = **true**;
165. **break**;
166. **default**:
167. **break**;
168. }
169. }
170. }
171. }
172. **while** ((btn1 == LOW) || (btn2 == LOW) || (btn3 == LOW))
173. {
174. btn1 = digitalRead(BTN\_1\_Pin);
175. btn2 = digitalRead(BTN\_2\_Pin);
176. btn3 = digitalRead(BTN\_3\_Pin);
177. esp\_task\_wdt\_reset();
178. delay(50);
179. }
180. }
181. esp\_task\_wdt\_reset();
182. delay(50);
183. }
184. }

Gui.cpp:

1. TFT\_eSPI tft;
2. **static** lv\_disp\_buf\_t disp\_buf;
3. **static** lv\_color\_t buf[LV\_HOR\_RES\_MAX \* 10] = {0};
5. **void** my\_disp\_flush(lv\_disp\_drv\_t \*disp, **const** lv\_area\_t \*area, lv\_color\_t \*color\_p);
6. **void** my\_print(lv\_log\_level\_t level, **const** **char** \*file, uint32\_t line, **const** **char** \*fn\_name, **const** **char** \*dsc);
8. **void** GUI\_Init()
9. {
10. lv\_init(); // 初始化LVGL
12. lv\_log\_register\_print\_cb((lv\_log\_print\_g\_cb\_t)my\_print); // 注册log输出组件
14. tft.begin(); // 初始化TFT
15. tft.setRotation(3);
16. tft.invertDisplay(0);
17. tft.fillScreen(TFT\_WHITE);
19. lv\_disp\_buf\_init(&disp\_buf, buf, NULL, LV\_HOR\_RES\_MAX \* 10); // 注册buffer
21. // 注册屏幕组件
22. **static** lv\_disp\_drv\_t disp\_drv;
23. lv\_disp\_drv\_init(&disp\_drv);
24. disp\_drv.hor\_res = TFT\_HEIGHT;
25. disp\_drv.ver\_res = TFT\_WIDTH;
26. disp\_drv.flush\_cb = my\_disp\_flush;
27. disp\_drv.buffer = &disp\_buf;
28. lv\_disp\_drv\_register(&disp\_drv);
29. }
31. **void** GUI\_Run(**void** \*parm)
32. {
33. **while** (**true**){
34. lv\_task\_handler();
35. esp\_task\_wdt\_reset();
36. delay(10);
37. }
38. }
40. **void** my\_print(lv\_log\_level\_t level, **const** **char** \*file, uint32\_t line, **const** **char** \*fn\_name, **const** **char** \*dsc)
41. {
42. Serial.printf("%s(%s)@%d->%s", file, fn\_name, line, dsc);
43. Serial.println("");
44. Serial.flush();
45. }
47. **void** my\_disp\_flush(lv\_disp\_drv\_t \*disp, **const** lv\_area\_t \*area, lv\_color\_t \*color\_p)
48. {
49. uint32\_t w = (area->x2 - area->x1 + 1);
50. uint32\_t h = (area->y2 - area->y1 + 1);
52. tft.startWrite();
53. tft.setAddrWindow(area->x1, area->y1, w, h);
54. tft.pushColors((uint16\_t \*)&color\_p->full, w \* h, **true**);
55. tft.endWrite();
57. lv\_disp\_flush\_ready(disp);
58. }

Sensor.cpp:

1. OneWire oneWire(SENSOR\_Pin);
2. DallasTemperature sensors(&oneWire);
4. **float** temperature;
5. uint8\_t addr[8];
7. **void** Sensor\_Init()
8. {
9. sensors.begin();
10. sensors.setWaitForConversion(**false**);
11. }
13. **void** Sensor\_Read(**void** \*parm)
14. {
15. uint8\_t retry = 0;
16. **while** (**true**)
17. {
18. sensors.requestTemperatures(); // Send the command to get temperatures
19. esp\_task\_wdt\_reset();
20. delay(500);
21. esp\_task\_wdt\_reset();
22. delay(500);
23. **while** (retry < 10)
24. {
25. auto temp = sensors.getTempCByIndex(0);
26. **if** ((temp != DEVICE\_DISCONNECTED\_C) && (temp != 85))
27. {
28. **char** ch[30];
29. sprintf(ch, "%.2f", temp);
30. Serial.print("Temperature for the device 1 (index 0) is: ");
31. Serial.printf("%f\n", temp);
32. temperature = temp;
33. sprintf(ch, "%.2f", temperature);
34. SendTemperature(temperature);
35. lv\_chart\_set\_next(guider\_ui.screen\_chart\_1, screen\_chart\_1\_0, int16\_t(temperature));
36. lv\_label\_set\_text(guider\_ui.screen\_temp\_label, ch);
37. **break**;
38. }
39. **else**
40. {
41. retry++;
42. esp\_task\_wdt\_reset();
43. delay(100);
44. Serial.println("Error: Could not read temperature data");
45. }
46. }
47. retry = 0;
48. esp\_task\_wdt\_reset();
49. delay(500);
50. }
51. }

Pid.cpp:

1. **float** sum\_err;
2. **float** last\_err;
3. **float** tao;
5. **void** PID\_Init()
6. {
7. kp = 1;
8. ki = 1;
9. target = 0;
10. sum\_err = 0;
11. last\_err = 0;
12. tao = 0.5;
13. }
15. **void** PID\_Reset()
16. {
17. sum\_err = 0;
18. last\_err = 0;
19. }
21. **float** PID\_Calculate(**float** present)
22. {
23. auto err = target - present;
24. sum\_err += err;
25. last\_err = err;
27. auto out = kp \* (err + tao / ki \* sum\_err);
28. **return** out;
29. }

Heat.cpp

1. **void** Heater\_Init()
2. {
3. ledcSetup(0, 100, 10);
4. ledcAttachPin(HEAT\_Pin, 0);
5. ledcWrite(0, 0);
6. }
8. **void** Heater\_SetDuty(**float** duty)
9. {
10. ledcWrite(0, **int**(duty / 100 \* 1024));
11. }

详细代码可查看仓库：https://github.com/kaidegit/TempertureSensorWithPIDController