**单片机程序主函数：**

#include "Usermain.h"

extern char TEMP[45];

extern char HUMI[45];

extern unsigned char BMP1[];

extern uint8\_t Hum\_Mod;

extern uint8\_t SHT30Update\_Flag;

extern uint8\_t AntiBur\_Flag;

void Usermain(void)

{

UserInit();

while(1)

{

if( !AntiBur\_Flag ) User\_Update();//如果防烧标志位为0，则正常运行。

else{ //如果防烧标志位为1，触发防烧警报。

OLED\_ShowString(12, 0, "Press any key", 12);

OLED\_ShowString(20, 2, "to exit the ", 12);

OLED\_ShowString(15, 4, "anti-burning", 12);

OLED\_ShowString(42, 6, "state", 12);

BEEP\_Yell\_Times(10, (uint8\_t)200, (uint8\_t)200);

}

}

}

/\*\*

\* @brief 加湿器初始化

\* @param 无

\* @note 调整内部参数可调节声音和画面出现的顺序、蜂鸣器时间等

\*/

void UserInit(void)

{

OLED\_Init();

Humidifier\_Off();

SHT30\_Reset();

if(SHT30\_Init() == HAL\_OK)

printf("sht30 init ok.\n");

else

printf("sht30 init fail.\n");

BEEP\_Yell\_Times(3, (uint8\_t)100, (uint8\_t)100);

OLED\_DrawBMP(0,0,128,8,BMP1);

HAL\_Delay(300);

OLED\_Clear();

}

/\*\*

\* @brief OLED屏幕显示函数

\* @param 无

\* @note 调整内部参数可调节OLED屏幕显示样式

\*/

void OLED\_Display(void){

OLED\_ShowString(0, 1, "----------------", 12);

OLED\_ShowString(10, 2, "TEM:", 12);

OLED\_ShowString(5, 3, TEMP, 16);

OLED\_ShowString(45, 3, "`", 16);

OLED\_ShowString(10, 5, "HUM:", 12);

OLED\_ShowString(5, 6, HUMI, 16);

OLED\_ShowString(45, 6, "%", 16);

OLED\_ShowString(60, 2, "|", 16);

OLED\_ShowString(60, 4, "|", 16);

OLED\_ShowString(60, 6, "|", 16);

OLED\_ShowString(86, 2, "MOD:", 12);

switch(Hum\_Mod){

case Hum\_Stop:

OLED\_ShowString(78, 4, " ", 16);

OLED\_ShowString(110, 4, ">", 16);

break;

case Hum\_Run:

OLED\_ShowString(78, 4, "<", 16);

OLED\_ShowString(110, 4, ">", 16);

break;

case Hum\_Int:

OLED\_ShowString(78, 4, "<", 16);

OLED\_ShowString(110, 4, ">", 16);

break;

case Hum\_Auto:

OLED\_ShowString(78, 4, "<", 16);

OLED\_ShowString(110, 4, " ", 16);

break;

default:

break;

}

OLED\_ShowNum(94, 4, Hum\_Mod, 1, 16);

}

/\*\*

\* @brief 用户更新函数

\* @param 无

\* @note 用于更新Esp、SHT30、OLED和加湿器状态

\*/

void User\_Update(void){

Esp\_StatusConfirm();

if( SHT30Update\_Flag ) SHT30\_Update();

OLED\_Display();

Humidifier\_Play();

}

/\* printf重定向 \*/

#ifdef \_\_GNUC\_\_

#define PUTCHAR\_PROTOTYPE int \_\_io\_putchar(int ch)

#else

#define PUTCHAR\_PROTOTYPE int fputc(int ch, FILE \*f)

#endif

int fputc(int ch,FILE \*f)

{

uint8\_t temp[1]={ch};

HAL\_UART\_Transmit(&huart1,temp,1,10); //UartHandle设置

return ch;

}

PUTCHAR\_PROTOTYPE

{

HAL\_UART\_Transmit(&huart1,(uint8\_t\*)&ch,1,10);

return ch;

}

**加湿模式控制函数：**

#include "HumidifierControl.h"

uint8\_t Hum\_Mod = Hum\_Run;

uint8\_t AntiBur\_Flag = 0;

float Humidity\_Set = 40;

extern float humidity;

/\*\*

\* @brief 加湿器开

\* @param 无

\* @note 无

\*/

void Humidifier\_On(void){

HAL\_GPIO\_WritePin(HUM\_CON\_GPIO\_Port,HUM\_CON\_Pin,GPIO\_PIN\_SET);

}

/\*\*

\* @brief 加湿器关

\* @param 无

\* @note 无

\*/

void Humidifier\_Off(void){

HAL\_GPIO\_WritePin(HUM\_CON\_GPIO\_Port,HUM\_CON\_Pin,GPIO\_PIN\_RESET);

}

/\*\*

\* @brief 加湿器运行

\* @param 无

\* @note 停止模式-关闭；开始模式-开启；间歇开启模式-开5s关5s（500ms判断一次是否中途改变状态）；自动模式-湿度低于设定值开启，湿度高于设定值关闭；

\*/

void Humidifier\_Play(void){

if( Hum\_Mod == 1 )

Humidifier\_On();

else if( Hum\_Mod == 0 )

Humidifier\_Off();

else if( Hum\_Mod == 2){

if( Hum\_Mod == 2 ) Humidifier\_On();//开5s

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) Humidifier\_Off();//关5s

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

if( Hum\_Mod == 2 ) HAL\_Delay(500);

}

else if( Hum\_Mod == 3){

if(humidity < Humidity\_Set) Humidifier\_On();

else Humidifier\_Off();

}

}

**联网模块控制函数：**

#include "esp.h"

uint8\_t Esp\_Sta = 0; //Esp联网标志位

/\*\*

\* @brief Esp状态检测

\* @param 无

\* @note 用于更新Esp联网状态和加湿器联网控制状态

\*/

void Esp\_StatusConfirm(void)

{

//检测WIFI连接状态(0：连接;1：断开)

if( HAL\_GPIO\_ReadPin(ESP\_STA\_GPIO\_Port, ESP\_STA\_Pin) == RESET )

{

OLED\_ShowString(0, 0, "Online ", 12);

Esp\_Sta = 1;

}

else

{

OLED\_ShowString(0, 0, "Offline", 12);

Esp\_Sta = 0;

}

//检测加湿器工作命令(0：运行;1：停止)

if( Esp\_Sta && (HAL\_GPIO\_ReadPin(HUM\_STA\_GPIO\_Port, HUM\_STA\_Pin) == RESET) )

{

OLED\_ShowString(104, 0, "ON ", 12);

}

else if( Esp\_Sta && (HAL\_GPIO\_ReadPin(HUM\_STA\_GPIO\_Port, HUM\_STA\_Pin) == SET) )

{

OLED\_ShowString(104, 0, "OFF", 12);

}

else OLED\_ShowString(104, 0, "UNK", 12);

}

**蜂鸣器控制函数：**

#include "beep.h"

/\*\*

\* @brief 开启蜂鸣器

\* @param 无

\* @note 开启蜂鸣器

\*/

void BEEP\_Yell(void){

HAL\_GPIO\_WritePin(BEEP\_CON\_GPIO\_Port,BEEP\_CON\_Pin,GPIO\_PIN\_SET);

}

/\*\*

\* @brief 关闭蜂鸣器

\* @param

\* @note 关闭蜂鸣器

\*/

void BEEP\_Stop(void){

HAL\_GPIO\_WritePin(BEEP\_CON\_GPIO\_Port,BEEP\_CON\_Pin,GPIO\_PIN\_RESET);

}

/\*\*

\* @brief 间歇开启蜂鸣器

\* @param Times - 开启次数

\* @param On\_delay\_ms - 开启时间

\* @param Off\_delay\_ms - 关闭时间

\* @note 间歇开启蜂鸣器

\*/

void BEEP\_Yell\_Times(uint8\_t Times,uint8\_t On\_delay\_ms,uint8\_t Off\_delay\_ms){

uint8\_t i = 0;

for(i = 0;i<Times;i++){

HAL\_GPIO\_WritePin(BEEP\_CON\_GPIO\_Port,BEEP\_CON\_Pin,GPIO\_PIN\_SET);

HAL\_Delay(On\_delay\_ms);

HAL\_GPIO\_WritePin(BEEP\_CON\_GPIO\_Port,BEEP\_CON\_Pin,GPIO\_PIN\_RESET);

HAL\_Delay(Off\_delay\_ms);

}

}

**ESP-12E模块代码：**

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <DNSServer.h>

#include <ESP8266WebServer.h>

#include <WiFiManager.h>

//巴法云服务器地址默认即可

#define TCP\_SERVER\_ADDR "bemfa.com"

//服务器端口，tcp创客云端口8344

#define TCP\_SERVER\_PORT "8344"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*需要修改的部分\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//WIFI名称，区分大小写，不要写错

#define DEFAULT\_STASSID "TP-LINK\_DF5B"

//WIFI密码

#define DEFAULT\_STAPSW "yx990510"

//用户私钥，可在控制台获取,修改为自己的UID

#define UID "b9e4ceb2c84ab8e8368d1a88e352540b"

//主题名字，可在控制台新建

#define TOPIC "Humidifier003"

//单片机引脚值

const int Hum\_Pin = 4;//加湿器控制引脚(LOW:运行;HIGH:停止)

const int Con\_Pin = 5;//模块联网状态(LOW:连接;HIGH:断开)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//最大字节数

#define MAX\_PACKETSIZE 512

//设置心跳值30s

#define KEEPALIVEATIME 30\*1000

//tcp客户端相关初始化，默认即可

WiFiClient TCPclient;

String TcpClient\_Buff = "";

unsigned int TcpClient\_BuffIndex = 0;

unsigned long TcpClient\_preTick = 0;

unsigned long preHeartTick = 0;//心跳

unsigned long preTCPStartTick = 0;//连接

bool preTCPConnected = false;

//相关函数初始化

//连接WIFI

unsigned int WifiXTimes = 0;

void doWiFiTick();

void startSTA();

//TCP初始化连接

void doTCPClientTick();

void startTCPClient();

void sendtoTCPServer(String p);

//led 控制函数

void turnOnLed();

void turnOffLed();

/\*

\*发送数据到TCP服务器

\*/

void sendtoTCPServer(String p){

if (!TCPclient.connected())

{

Serial.println("Client is not readly");

return;

}

TCPclient.print(p);

Serial.println("[Send to TCPServer]:String");

Serial.println(p);

}

/\*

\*初始化和服务器建立连接

\*/

void startTCPClient(){

if(TCPclient.connect(TCP\_SERVER\_ADDR, atoi(TCP\_SERVER\_PORT))){

Serial.print("\nConnected to server:");

Serial.printf("%s:%d\r\n",TCP\_SERVER\_ADDR,atoi(TCP\_SERVER\_PORT));

char tcpTemp[128];

sprintf(tcpTemp,"cmd=1&uid=%s&topic=%s\r\n",UID,TOPIC);

sendtoTCPServer(tcpTemp);

preTCPConnected = true;

preHeartTick = millis();

TCPclient.setNoDelay(true);

}

else{

Serial.print("Failed connected to server:");

Serial.println(TCP\_SERVER\_ADDR);

TCPclient.stop();

preTCPConnected = false;

}

preTCPStartTick = millis();

}

/\*

\*检查数据，发送心跳

\*/

void doTCPClientTick(){

//检查是否断开，断开后重连

if(WiFi.status() != WL\_CONNECTED) return;

if (!TCPclient.connected()){//断开重连

digitalWrite(Con\_Pin,HIGH);//连接失败熄灭

if(preTCPConnected == true){

preTCPConnected = false;

preTCPStartTick = millis();

Serial.println();

Serial.println("TCP Client disconnected.");

TCPclient.stop();

}

else if(millis() - preTCPStartTick > 1\*1000)//重新连接

startTCPClient();

digitalWrite(Con\_Pin,HIGH);

}

else

{

digitalWrite(Con\_Pin,LOW);//连接成功亮灯

if (TCPclient.available()) {//收数据

char c =TCPclient.read();

TcpClient\_Buff +=c;

TcpClient\_BuffIndex++;

TcpClient\_preTick = millis();

if(TcpClient\_BuffIndex>=MAX\_PACKETSIZE - 1){

TcpClient\_BuffIndex = MAX\_PACKETSIZE-2;

TcpClient\_preTick = TcpClient\_preTick - 200;

}

preHeartTick = millis();

}

if(millis() - preHeartTick >= KEEPALIVEATIME){//保持心跳

preHeartTick = millis();

Serial.println("--Keep alive:");

sendtoTCPServer("cmd=0&msg=keep\r\n");

}

}

if((TcpClient\_Buff.length() >= 1) && (millis() - TcpClient\_preTick>=200))

{//data ready

TCPclient.flush();

Serial.println("Buff");

Serial.println(TcpClient\_Buff);

if((TcpClient\_Buff.indexOf("&msg=on") > 0)) {

turnOnHum();

}else if((TcpClient\_Buff.indexOf("&msg=off") > 0)) {

turnOffHum();

}

TcpClient\_Buff="";

TcpClient\_BuffIndex = 0;

}

}

void startSTA(){

WiFi.disconnect();

WiFi.mode(WIFI\_STA);

WiFi.begin(DEFAULT\_STASSID, DEFAULT\_STAPSW);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

WIFI

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

WiFiTick

检查是否需要初始化WiFi

检查WiFi是否连接上，若连接成功启动TCP Client

控制指示灯

\*/

void doWiFiTick(){

static bool startSTAFlag = false;

static bool taskStarted = false;

static uint32\_t lastWiFiCheckTick = 0;

if (!startSTAFlag) {

startSTAFlag = true;

startSTA();

Serial.printf("Heap size:%d\r\n", ESP.getFreeHeap());

}

//未连接1s重连

if ( WiFi.status() != WL\_CONNECTED ) {

if (millis() - lastWiFiCheckTick > 1000) {

lastWiFiCheckTick = millis();

WifiXTimes++;

}

if ( WifiXTimes >= 10){

WiFiManager wifiManager;

Serial.printf("AutoConnecting...\r\n");

wifiManager.autoConnect("SmartHumidifier");

}

}

//连接成功建立

else {

if (taskStarted == false) {

WifiXTimes = 0;

taskStarted = true;

Serial.print("Successful connection !!!\r\n");

Serial.print("ESP8266 Connected to ");

Serial.println(WiFi.SSID()); // WiFi名称

Serial.print("\r\nGet IP Address: ");

Serial.println(WiFi.localIP());

startTCPClient();

}

}

}

//打开加湿器

void turnOnHum(){

//Serial.println("Turn ON");

digitalWrite(Hum\_Pin,HIGH);

delay(50);

digitalWrite(Hum\_Pin,LOW);

}

//关闭加湿器

void turnOffHum(){

//Serial.println("Turn OFF");

digitalWrite(Hum\_Pin,LOW);

delay(50);

digitalWrite(Hum\_Pin,HIGH);

}

// 初始化，相当于main 函数

void setup() {

Serial.begin(115200);

pinMode(Hum\_Pin,OUTPUT);

pinMode(Con\_Pin,OUTPUT);

digitalWrite(Con\_Pin,HIGH);//GPIO0引脚初始值

digitalWrite(Hum\_Pin,LOW);//GPIO2引脚初始值

}

//循环

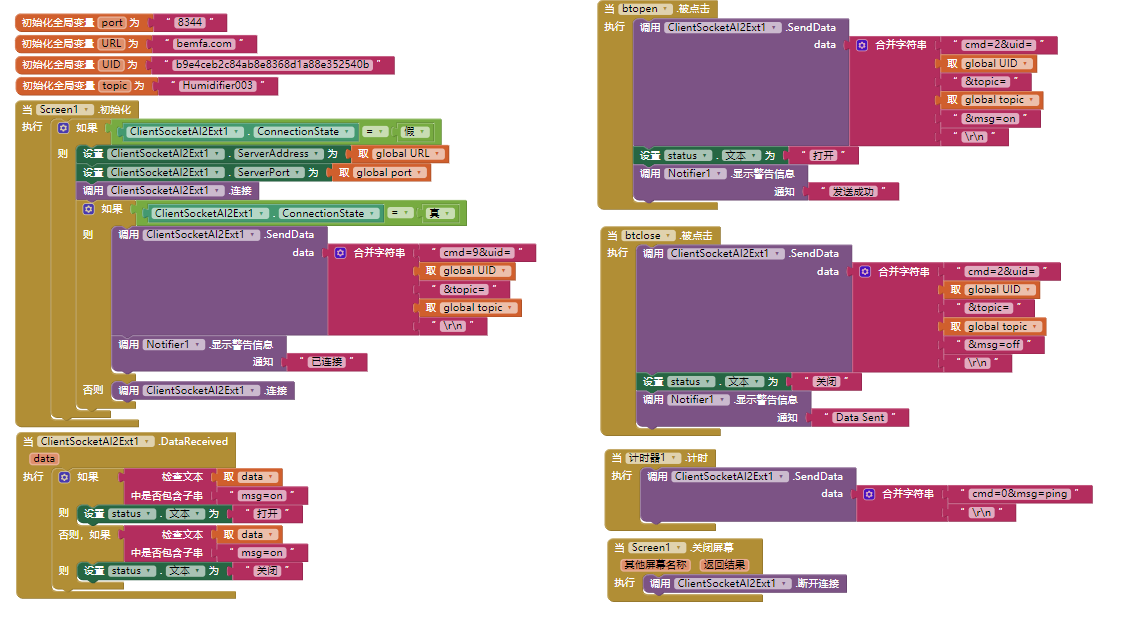
void loop() {

doWiFiTick();

doTCPClientTick();

}

**手机APP代码：**



**电路原理图：**

