```
#include <Adafruit_GFX.h>        // Core graphics library
#include <Adafruit_TFTLCD.h> // Hardware-specific library
#include <TouchScreen.h>
#include <OneWire.h>
#include <DS18B20.h>
#include <DS3231.h>
#include <ESP8266wifi.h>
#define esp8266_reset_pin 33 // Connect this pin to CH_PD on the esp8266, not reset. (let reset be unconnected)

//Gsm
bool bSmsSend = false;
bool bSmsWarn = false;
unsigned long tSmsStart = 0;

//暂停时间
String StrStopTime="20Min";

//摄像头
#define pinCameral1 28
#define pinCameral2 29

//屏幕密码锁
String PassWd = "1357";
Adafruit_GFX_Button Passbuttons[12];
bool bPassScreen = false;
bool bIsPass = false;
String PassWdp = "";
unsigned long tPassStart = 0;
unsigned long tPassEnd = 0;

//照明定时初始化
//平时
#define TimerStartHour 15
#define TimerEndHour 23
#define TimerStartDay 1
#define TimerEndDay 5
//周末
#define TimerStartHourW 13
#define TimerEndHourW 22
#define TimerStartDayW 6
#define TimerEndDayW 7
bool TimerAsWeekEnd = false;
bool bLight = true;
```

```
bool bLightOnTime = false;
bool bForceLight = false;

bool PumpStop = false;    //水泵是否关闭
                                    //WiFi
ESP8266wifi wifi(Serial2, Serial2, esp8266_reset_pin, Serial);//内部通信 wifi

//ESP8266wifi wifi2(Serial3, Serial3, esp8266_reset_pin, Serial);//外部通信 wifi

//滴定
#define pinDrop1    30
#define pinDrop2    32
#define pinDrop3    34
int pinDrop[3] = { pinDrop1 ,pinDrop2 ,pinDrop3 };

//蜂鸣器
#define pinBeep 41

//补水
#define pinWaterLow    A15
#define pinWaterHigh A13
#define pinWaterPump 37
bool bWaterFull = false;
bool bWaterLow = false;
int WaterLowVal = 0;
int WaterHighVal = 0;
String WaterLevel = "Nor";
bool bWaterLevelChange = false;
unsigned long TPumpStart = 0;

//WiFi
char col;
String data;
bool bWiFi = false;

//温度模块
DS18B20    Tempds(33);
uint8_t address[] = { 40, 255, 217, 164, 129, 22, 3, 47 };
uint8_t Tempselected;
float Tempreture;
float TempHigh = 27.5;
float TempLow = 23.5;
bool bTempWarning = false;
uint8_t TempExceptTimes = 0;
```

```
//时钟模块
DS3231 Clock;
bool Century = false;
bool h12;
bool PM;
byte ADay, AHour, AMinute, ASecond, ABits;
bool ADy, A12h, Apm;
byte year, month, date, DoW, hour, minute, second;
String Times;
//屏幕模块
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0
#define LCD_RESET A4 // Can alternately just connect to Arduino's reset pin


#define    BLACK      0x0000
#define BLUE        0x001F
#define RED         0xF800
#define GREEN       0x07E0
#define CYAN        0x07FF
#define MAGENTA 0xF81F
#define YELLOW    0xFFE0
#define WHITE       0xFFFF

// Color definitions
#define ILI9341_BLACK           0x0000          /*      0,      0,      0 */
#define ILI9341_NAVY            0x000F          /*      0,      0, 128 */
#define ILI9341_DARKGREEN       0x03E0          /*      0, 128,      0 */
#define ILI9341_DARKCYAN        0x03EF          /*      0, 128, 128 */
#define ILI9341_MAROON          0x7800          /* 128,      0,      0 */
#define ILI9341_PURPLE          0x780F          /* 128,      0, 128 */
#define ILI9341_OLIVE           0x7BE0          /* 128, 128,      0 */
#define ILI9341_LIGHTGREY       0xC618          /* 192, 192, 192 */
#define ILI9341_DARKGREY        0x7BEF          /* 128, 128, 128 */
#define ILI9341_BLUE            0x001F          /*      0,      0, 255 */
#define ILI9341_GREEN           0x07E0          /*      0, 255,      0 */
#define ILI9341_CYAN            0x07FF          /*      0, 255, 255 */
#define ILI9341_RED             0xF800          /* 255,      0,      0 */
#define ILI9341_MAGENTA         0xF81F          /* 255,      0, 255 */
#define ILI9341_YELLOW          0xFFE0          /* 255, 255,      0 */
#define ILI9341_WHITE           0xFFFF          /* 255, 255, 255 */
#define ILI9341_ORANGE          0xFD20          /* 255, 165,      0 */
```

```
#define ILI9341_GREENYELLOW 0xAFE5          /* 173, 255,    47 */
#define ILI9341_PINK                0xF81F

/****************** UI details */
#define BUTTON_X 40
#define BUTTON_Y 100
#define BUTTON_W 60
#define BUTTON_H 30
#define BUTTON_SPACING_X 20
#define BUTTON_SPACING_Y 20
#define BUTTON_TEXTSIZE 2

// text box where numbers go
#define TEXT_X 10
#define TEXT_Y 10
#define TEXT_W 220
#define TEXT_H 50
#define TEXT_TSIZE 3
#define TEXT_TCOLOR ILI9341_MAGENTA
// the data (phone #) we store in the textfield
#define TEXT_LEN 12

#define YP A1    // must be an analog pin, use "An" notation!
#define XM A2    // must be an analog pin, use "An" notation!
#define YM 7      // can be a digital pin
#define XP 6      // can be a digital pin

#define TS_MINX 100
#define TS_MAXX 920

#define TS_MINY 70
#define TS_MAXY 900
// We have a status line for like, is FONA working
#define STATUS_X 10
#define STATUS_Y 65

#define MINPRESSURE 10       //触摸屏按压参数
#define MAXPRESSURE 1000

unsigned long tstart = 0;
unsigned long tend = 0;
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
//#include <MCUFRIEND_kbv.h>
//MCUFRIEND_kbv tft;
```

```
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
// If using the shield, all control and data lines are fixed, and
// a simpler declaration can optionally be used:
// Adafruit_TFTLCD tft;

Adafruit_GFX_Button buttons[4];
/* create 15 buttons, in classic candybar phone style */
uint16_t buttoncolors[4] = { ILI9341_DARKGREEN, ILI9341_ORANGE, ILI9341_RED, ILI9341_BLUE };

void setup(void) {

    delay(5000);

    Serial.println(F("TFT LCD test"));

#ifdef USE_ADAFRUIT_SHIELD_PINOUT
    Serial.println(F("Using Adafruit 2.4\" TFT Arduino Shield Pinout"));
#else
    Serial.println(F("Using Adafruit 2.4\" TFT Breakout Board Pinout"));
#endif

    Serial.print("TFT size is "); Serial.print(tft.width()); Serial.print("x"); Serial.println(tft.height());

    tft.reset();

    uint16_t identifier = tft.readID();
    if (identifier == 0x9325) {
        Serial.println(F("Found ILI9325 LCD driver"));
    }
    else if (identifier == 0x9328) {
        Serial.println(F("Found ILI9328 LCD driver"));
    }
    else if (identifier == 0x4535) {
        Serial.println(F("Found LGDP4535 LCD driver"));
    }
    else if (identifier == 0x7575) {
        Serial.println(F("Found HX8347G LCD driver"));
    }
    else if (identifier == 0x9341) {
        Serial.println(F("Found ILI9341 LCD driver"));
    }
    else if (identifier == 0x7783) {
        Serial.println(F("Found ST7781 LCD driver"));
```

```
}
else if (identifier == 0x8230) {
      Serial.println(F("Found UC8230 LCD driver"));
}
else if (identifier == 0x8357) {
      Serial.println(F("Found HX8357D LCD driver"));
}
else if (identifier == 0x0101)
{
      identifier = 0x9341;
      Serial.println(F("Found 0x9341 LCD driver"));
}
else {
      Serial.print(F("Unknown LCD driver chip: "));
      Serial.println(identifier, HEX);
      Serial.println(F("If using the Adafruit 2.8\" TFT Arduino shield, the line:"));
      Serial.println(F("    #define USE_ADAFRUIT_SHIELD_PINOUT"));
      Serial.println(F("should appear in the library header (Adafruit_TFT.h)."));
      Serial.println(F("If using the breakout board, it should NOT be #defined!"));
      Serial.println(F("Also if using the breakout, double-check that all wiring"));
      Serial.println(F("matches the tutorial."));
      identifier = 0x7575;

}

tft.begin(identifier);
tft.setRotation(1);//设置屏幕方向
CleanScreen();

//滴定初始化
pinMode(pinDrop1, OUTPUT);
pinMode(pinDrop2, OUTPUT);
pinMode(pinDrop3, OUTPUT);


//时钟初始化
Wire.begin();
//温度初始化
Tempselected = Tempds.select(address);
if (!Tempselected)
{
      Serial.println("Device not found!");
}
```

```
// create 'text field'
//tft.drawRect(TEXT_X, TEXT_Y, 300, TEXT_H, ILI9341_WHITE);
String PrintText = "";
tstart = millis();
Times = ReadTime();
Tempreture = GetTemperature(0.1);
//tft.setCursor(TEXT_X, TEXT_Y + 10);
//tft.setTextColor(ILI9341_WHITE, ILI9341_BLACK);
//tft.setTextSize(2);
PrintText = PrintText + " TEMP:" + Tempreture + " Water:Nor";
Display(PrintText);
//tft.print(PrintText.c_str());

status("Starting wifi.....");

Serial.begin(9600);

//补水初始化
pinMode(pinWaterPump,OUTPUT);
pinMode(pinWaterHigh, INPUT);
pinMode(pinWaterLow, INPUT);
//蜂鸣器初始化
pinMode(pinBeep,OUTPUT);
//WiFi 初始化
Serial2.begin(9600);
Serial.println("Starting wifi");
wifi.setTransportToTCP();// this is also default
wifi.endSendWithNewline(false);
wifi.begin();
wifi.connectToAP("mywifi", "wifipass");
if(!wifi.connectToServer("192.168.4.1", "8080"))
{
      bWiFi=false;
      Serial.println("Connect Server Error");
      status("Wifi Start Error");
}
else
{
      bWiFi=true;
      status("Wifi Started");
}
//GSM 初始化
delay(60000);
Serial3.begin(115200);
```

```
        Serial3.println("AT+CMGF=1");
        delay(1500);
        Serial3.println("AT+CPMS=\"SM\",\"SM\",\"SM\"");
        delay(1500);

        DrawPressButton();
}
// or charstring
void status(String msg) {
        tft.fillRect(STATUS_X, STATUS_Y, 240, 8, ILI9341_BLACK);
        tft.setCursor(STATUS_X, STATUS_Y);
        tft.setTextColor(ILI9341_WHITE);
        tft.setTextSize(1);
        tft.print(msg.c_str());
}

void Display(String PRN)
{
        tft.drawRect(TEXT_X, TEXT_Y, 300, TEXT_H, ILI9341_WHITE);
        tft.setCursor(TEXT_X+8, TEXT_Y + 15);
        tft.setTextColor(ILI9341_WHITE, ILI9341_BLACK);
        tft.setTextSize(2);
        tft.print(PRN.c_str());
}

void CleanScreen()
{
        tft.fillRect(0, 0, 320, 240, ILI9341_BLACK);

}

int ButtonX = 40;

void DrawPassButton(int nPos)
{
        ButtonX = BUTTON_X + 40;
        /* create 15 buttons, in classic candybar phone style */
        char buttonlabels[12][5] = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "CLR", "OK", "#" };
        uint16_t buttoncolors[12] = {
                ILI9341_BLUE, ILI9341_BLUE, ILI9341_BLUE,
                ILI9341_BLUE, ILI9341_BLUE, ILI9341_BLUE,
                ILI9341_BLUE, ILI9341_BLUE, ILI9341_BLUE,
                ILI9341_DARKGREEN, ILI9341_DARKGREY, ILI9341_RED };
```

```
    for (uint8_t row = 0; row<4; row++) {
        for (uint8_t col = 0; col<3; col++) {
            Passbuttons[col + row * 3].initButton(&tft, ButtonX + col*(BUTTON_W + BUTTON_SPACING_X),
                43 + row*(BUTTON_H + BUTTON_SPACING_Y),      // x, y, w, h, outline, fill, text
                BUTTON_W, BUTTON_H, ILI9341_WHITE, buttoncolors[col + row * 3], ILI9341_WHITE,
                buttonlabels[col + row * 3], BUTTON_TEXTSIZE);
            Passbuttons[col + row * 3].drawButton();
        }
    }
}

void Timer()
{

}

void GsmReturnStatus()
{
    String ReData = "";
    ReData = GetSocketStatus();
    ReData = ReData + "Tempretur:" + Tempreture + ";WaterLevel:" + WaterLevel + ";";
    if (ReData != "")
    {
        Serial3.println("AT+CMGS=\"133xxxxxxxx\"");
        delay(2000);
        Serial3.print(ReData.c_str());
        delay(600);
        Serial3.print("\x01A");
        delay(1500);
    }
}

void GetGsmMessage()
{
    data = "";
    Serial3.println("AT+CMGR=1");
    delay(1500);
    while (Serial3.available() > 0)
    {
        col = Serial3.read();
        if(col!=10 && col!=13)
```

```
                data = data + col;
        }
        if (data != "")
        {
            GsmMessageProcess(data);
        }
        Serial3.println("AT+CMGD=1");
}

void SendGsmMessage()
{
        if (bSmsSend)
            return;
        bSmsSend = true;
        String sms="";
        if (bWaterFull) sms = "Warning: Water is Full";
        if (bWaterLow) sms = "Warning: Water is Low";
        if (bTempWarning) sms = sms+"Waring: Tempreture " + Tempreture;
        bSmsWarn = true;
        tSmsStart = millis();
        //发两次
        Serial3.println("AT+CMGS=\"133xxxxxxxx\"");
        delay(2000);
        Serial3.print(sms.c_str());
        delay(600);
        Serial3.print("\x01A");
        delay(30000);

        Serial3.println("AT+CMGS=\"133xxxxxxxx\"");
        delay(2000);
        Serial3.print(sms.c_str());
        delay(600);
        Serial3.print("\x01A");
        delay(30000);
}

void WarnCall()
{
        if (!bSmsSend)
            return;

        if (millis() - tSmsStart > 1800000)   //30 分钟无处理，即打电话预警
        {
            Serial3.println("ATD133xxxxxxxx");
```

```
            bSmsSend = false;
        }

}

void GsmMessageProcess(String msg)
{
        if (msg.indexOf("133xxxxxxxx") < 0)
        {
            return;
        }

        bSmsSend = false;

        if (msg.indexOf("ReturnStatus") >= 0)
        {
            GsmReturnStatus();
        }

        if (msg.indexOf("Cameral:ON") >= 0)
        {
            wifi.send(SERVER,"Con2ON");
        }

        if (msg.indexOf("Cameral:OFF") >= 0)
        {
            wifi.send(SERVER, "Con2OFF");
        }

        if (msg.indexOf("Pump:OFF") >= 0)
        {
            wifi.send(SERVER,"Con3:OFF");
            delay(500);
        }

        if (msg.indexOf("Pump:ON") >= 0)
        {
            wifi.send(SERVER,"Con3:ON");
            delay(500);
        }

        if (msg.indexOf("Wave:ON") >= 0)
        {
            wifi.send(SERVER,"Con5:ON");
```

```
        delay(500);
}

if (msg.indexOf("Wave:OFF") >= 0)
{
        wifi.send(SERVER,"Con5:OFF");
        delay(500);
}

if (msg.indexOf("Light:ON") >= 0)
{
        wifi.send(SERVER, "Con4:ON");
        delay(500);
}

if (msg.indexOf("Light:OFF") >= 0)
{
        wifi.send(SERVER, "Con4:OFF");
        delay(500);
}

if (msg.indexOf("Set:20Min") >= 0)
{
        wifi.send(SERVER, "20Min");
        delay(500);
}

if (msg.indexOf("Set:40Min") >= 0)
{
        wifi.send(SERVER, "40Min");
        delay(500);
}

if (msg.indexOf("Set:60Min") >= 0)
{
        wifi.send(SERVER, "60Min");
        delay(500);
}

if (msg.indexOf("Set:120Min") >= 0)
{
        wifi.send(SERVER, "120Min");
        delay(500);
}
```

```
}

void DrawPressButton()
{
    // 画 4 个按钮

    buttons[0].initButton(&tft, 85, 115, 100, 60,       // x, y, w, h, outline, fill, text
        ILI9341_WHITE, buttoncolors[0], ILI9341_WHITE,
        "Pump", BUTTON_TEXTSIZE);
    buttons[0].drawButton();

    buttons[1].initButton(&tft, 230, 115, 100, 60,       // x, y, w, h, outline, fill, text
        ILI9341_WHITE, buttoncolors[1], ILI9341_WHITE,
        "20Min", BUTTON_TEXTSIZE);
    buttons[1].drawButton();

    buttons[2].initButton(&tft, 85, 195, 100, 60,       // x, y, w, h, outline, fill, text
        ILI9341_WHITE, buttoncolors[2], ILI9341_WHITE,
        "Light", BUTTON_TEXTSIZE);
    buttons[2].drawButton();

    buttons[3].initButton(&tft, 230, 195, 100, 60,       // x, y, w, h, outline, fill, text
        ILI9341_WHITE, buttoncolors[3], ILI9341_WHITE,
        "Wave", BUTTON_TEXTSIZE);
    buttons[3].drawButton();

    GetSocketStatus();
}

void ChangeStopTime()
{
    if (StrStopTime == "20Min")
    {
        StrStopTime = "40Min";
        wifi.send(SERVER, StrStopTime.c_str());
    }
    else if (StrStopTime == "40Min")
    {
        StrStopTime = "60Min";
        wifi.send(SERVER, StrStopTime.c_str());
    }
    else if (StrStopTime == "60Min")
    {
```

```
            StrStopTime = "120Min";
            wifi.send(SERVER, StrStopTime.c_str());
        }
        else if (StrStopTime == "120Min")
        {
            StrStopTime = "20Min";
            wifi.send(SERVER, StrStopTime.c_str());
        }
        buttons[1].setLabel(StrStopTime.c_str());
        buttons[1].press(false);
        buttons[1].drawButton();
}

void Warning()
{
        if (bWaterFull || bWaterLow || bTempWarning)
        {
            tone(pinBeep, 2000, 5);
            SendGsmMessage();
        }
}

void TestWaterLevel()
{
        if (PumpStop)
            return;


        if (millis() - TPumpStart < 90000)
            return;

        String PrintText = "";

        WaterHighVal = analogRead(pinWaterHigh);
        WaterLowVal = analogRead(pinWaterLow);
        if (WaterHighVal>450)
        {
            if (!bWaterFull)
            {
                digitalWrite(pinWaterPump, HIGH);
                bWaterLevelChange = true;
            }
            bWaterFull = true;
        }
```

```
else
{
	if (bWaterFull)
	{
		digitalWrite(pinWaterPump, LOW);
		bWaterLevelChange = true;
	}
	bWaterFull = false;
}

if (WaterLowVal<500)
{
	if (!bWaterLow)
	{
		bWaterLevelChange = true;
	}
	bWaterLow = true;
}
else
{
	if (bWaterLow)
	{
		bWaterLevelChange = true;
	}
	bWaterLow = false;
}

if (!bWaterFull && !bWaterLow)
{
	WaterLevel = "Nor";
}
else if (bWaterFull)
{
	WaterLevel = "Ful";
}
else
	WaterLevel = "Low";

if (bWaterLevelChange)
{
	bSmsSend = false;
	PrintText = "";
	PrintText = PrintText + " TEMP:" + Tempreture + " Water:" + WaterLevel;
	Display(PrintText);
```

```
    }
}

void GetWifiMessage()
{
    data = "";
    while (Serial2.available()>0)
    {
        col = Serial2.read();
        data = data + col;
        delay(50);
    }

    if (data.indexOf("SocketStatus:") > 0)
        SetSockStatus(data);
}

String GetSocketStatus()
{
    data = "";
    String RecvData = "";
    if (bWiFi)
    {
        wifi.send(SERVER, "ReturnStatus");
        delay(2000);
        while (Serial2.available()>0)
        {
            col = Serial2.read();
            data = data + col;
            delay(50);
        }
        RecvData = data;
        Serial.println(RecvData);
        SetSockStatus(RecvData);
    }

    return RecvData;
}

void SetSockStatus(String RecvData)
{
    if (RecvData.indexOf("2:20;") >= 0|| RecvData.indexOf("2:40;") >= 0||
RecvData.indexOf("2:60;") >= 0|| RecvData.indexOf("2:120;") >= 0)
    {
```

```
        if (RecvData.indexOf("2:20;") >= 0)
            StrStopTime = "20Min";
        else if (RecvData.indexOf("2:40;") >= 0)
            StrStopTime = "40Min";
        else if (RecvData.indexOf("2:60;") >= 0)
            StrStopTime = "60Min";
        else if (RecvData.indexOf("2:120;") >= 0)
            StrStopTime = "120Min";
        buttons[1].setLabel(StrStopTime.c_str());
        buttons[1].press(false);
        buttons[1].drawButton();
}

if (RecvData.indexOf("2:O;") >= 0)
{
        buttons[0].press(false);
        buttons[0].drawButton();
        PumpStop = false;
        TPumpStart = millis();
}

if (RecvData.indexOf("3:O;") >= 0)
{
        buttons[2].press(false);
        buttons[2].drawButton();
        bLight = true;
}
if (RecvData.indexOf("4:O;") >= 0)
{
        buttons[3].press(false);
        buttons[3].drawButton();
}

if (RecvData.indexOf("2:F;") >= 0)
{
        buttons[0].press(true);
        buttons[0].drawButton(true);
        PumpStop = true;
}
if (RecvData.indexOf("3:F;") >= 0)
{
        buttons[2].press(true);
        buttons[2].drawButton(true);
        bLight = false;
```

```
        }
        if (RecvData.indexOf("4:F;") >= 0)
        {
                buttons[3].press(true);
                buttons[3].drawButton(true);
        }
}

void LightTimer()
{
        bLightOnTime = false;
        float tminute = 0;
        float thour = 0;
        float CurTime = 0;

        tminute = minute;
        thour = hour;

        CurTime = thour + tminute / 60;

        if (CurTime >= TimerStartHour && CurTime <= TimerEndHour)
        {
                if (DoW >= TimerStartDay && DoW <= TimerEndDay)
                {
                        bLightOnTime = true;
                        if (!bLight)
                        {
                                wifi.send(SERVER, "LightON");
                                bLight = true;
                                buttons[2].press(false);
                                buttons[2].drawButton();
                        }
                }
        }

        if (DoW >= TimerStartDayW && DoW <= TimerEndDayW || TimerAsWeekEnd)
        {
                if (CurTime >= TimerStartHourW && CurTime <= TimerEndHourW)
                {
                        bLightOnTime = true;
                        if (!bLight)
                        {
                                wifi.send(SERVER, "LightON");
                                bLight = true;
```

```
                    buttons[2].press(false);
                    buttons[2].drawButton();
                }
            }
        }

        if (!bLightOnTime)
        {
            if (bLight)
            {
                wifi.send(SERVER, "LightOFF");
                bLight = false;
                buttons[2].press(true);
                buttons[2].drawButton(true);
            }
        }

}

void PumpON()
{
    if (bWiFi)
    {
        wifi.send(SERVER, "Con3ON");
        PumpStop = false;
        TPumpStart = millis();
        //GetSocketStatus();
    }
}

void PumpOFF()
{
    if (bWiFi)
    {
        wifi.send(SERVER, "Con3OFF");
        PumpStop = true;
        bWaterFull = false;
        //GetSocketStatus();
    }
}

void SkimON()
{
    if (bWiFi)
```

```
	{
		wifi.send(SERVER, "Con3ON");
	}
}

void SkimOFF()
{
	if (bWiFi)
	{
		wifi.send(SERVER, "Con3OFF");
	}
}

void SkimForceOFF()
{
	if (bWiFi)
	{
		wifi.send(SERVER, "SkimOFF");
	}
}

void LightON()
{
	if (bWiFi)
	{
		wifi.send(SERVER, "Con4ON");
		bLight = true;
		if (!bForceLight)
			bForceLight = true;
		else
			bForceLight = false;
	}
}

void LightOFF()
{
	if (bWiFi)
	{
		wifi.send(SERVER, "Con4OFF");
		bLight = false;
		if (!bForceLight)
			bForceLight = true;
		else
			bForceLight = false;
```

```
    }
}

void OtherON()
{
    if (bWiFi)
    {
        wifi.send(SERVER, "Con5ON");
    }
}

void OtherOFF()
{
    if (bWiFi)
    {
        wifi.send(SERVER, "Con5OFF");
    }
}

String ReadTime()
{
    String TimeS = "";
    //int second,minute,hour,date,month,year,temperature;
    second = Clock.getSecond();
    minute = Clock.getMinute();
    hour = Clock.getHour(h12, PM);



    date = Clock.getDate();
    month = Clock.getMonth(Century);
    year = Clock.getYear();
    DoW = Clock.getDoW();

    TimeS = TimeS + "20" + (int)year + "-" + month + "-" + date + " " + hour + ":" + minute;
    //Serial.println(TimeS);
    return TimeS;
}

float GetTemperature(float OldTemp)
{
    float celsius;

    if (!Tempselected) //找不到温度传感器，返回-1
```

```
            return -1;
        celsius = Tempds.getTempC();

        if (celsius < 0 || celsius>100)
        {
            TempExceptTimes++;
            if (TempExceptTimes > 3)
                return celsius;
            else
                return OldTemp;
        }
        TempExceptTimes = 0;
        return celsius;
}

void ConnectWiFi()
{
        wifi.connectToAP("mywifi", "wifipass");
        if (!wifi.connectToServer("192.168.4.1", "8080"))
        {
            bWiFi = false;
            tft.setCursor(TEXT_X, TEXT_Y);
            tft.setTextColor(ILI9341_WHITE, ILI9341_BLACK);
            tft.setTextSize(2);
            tft.print("WiFi Connect Error");
        }
        else
            bWiFi = true;
}

bool TimeOut(unsigned Dual)
{
        tPassEnd = millis();
        if (tPassEnd - tPassStart >= Dual || tPassEnd - tPassStart < 0)
        {
            tPassStart = millis();
            return true;
        }
        else
            return false;
}

void TimeDrop(int &Start,int Interval,int Dual,int Pin)
{
```

```
    int Int1 = 0;

    bool bDrop = false;

    Int1 = hour - Start;

    if (Int1 >= Interval)
    {
        Start = hour;
        bDrop = true;
    }
    else if (Int1 < 0    &&    Int1+24 >= Interval)
    {
        Start = hour;
        bDrop = true;
    }

    digitalWrite(Pin, HIGH);
    delay(Dual * 1000);
    digitalWrite(Pin, LOW);
}

void TouchPin()
{
    digitalWrite(13, HIGH);
    TSPoint p = ts.getPoint();
    digitalWrite(13, LOW);
    //swap(p.x, p.y);
    pinMode(XM, OUTPUT);
    digitalWrite(XM, LOW);
    pinMode(YP, OUTPUT);
    digitalWrite(YP, HIGH);
    pinMode(YM, OUTPUT);
    digitalWrite(YM, LOW);
    pinMode(XP, OUTPUT);
    digitalWrite(XP, HIGH);

    if (p.z > MINPRESSURE && p.z < MAXPRESSURE)//检测是否按键
    {
        // scale from 0->1023 to tft.width
        p.x = (tft.width() - map(p.x, TS_MINX, TS_MAXX, tft.width(), 0));
        p.y = map(p.y, TS_MINY, TS_MAXY, tft.height(), 0);

        if (bIsPass)
```

```
        { }
        else
        {
                bIsPass = false;
                bPassScreen = true;
                tPassStart = millis();
                return;
        }
}

if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
        Serial.print("X = "); Serial.print(p.x);
        Serial.print("\tY = "); Serial.print(p.y);
        Serial.print("\tPressure = "); Serial.println(p.z);
}

// go thru all the buttons, checking if they were pressed
if (p.z > MINPRESSURE && p.z < MAXPRESSURE) //检测是否按键
{
        for (uint8_t b = 0; b<4; b++) {
                if (buttons[b].contains(p.x, p.y)) {
                        Serial.print("Pressing: "); Serial.println(b);
                        if (buttons[b].isPressed())
                        {
                                buttons[b].press(false);
                                switch (b)
                                {
                                case 0:       //
                                        PumpON();
                                        break;
                                case 1:       //
                                        SkimON();
                                        break;
                                case 2:       //
                                        LightON();
                                        break;
                                case 3:       //
                                        OtherON();
                                        break;
                                }
                                //delay(200);
                                //GetSocketStatus();
                        }
                        else
```

```
                    {
                        buttons[b].press(true);
                        switch (b)
                        {
                        case 0:       //
                            PumpOFF();
                            break;
                        case 1:       //
                            SkimOFF();
                            break;
                        case 2:       //
                            LightOFF();
                            break;
                        case 3:       //
                            OtherOFF();
                            break;
                        }
                        //delay(200);
                        //GetSocketStatus();
                    }
                }
            }

            // now we can ask the buttons if their state has changed
            for (uint8_t b = 0; b<4; b++) {
                if (buttons[b].justReleased()) {
                    // Serial.print("Released: "); Serial.println(b);
                    buttons[b].drawButton();    // draw normal
                }

                if (buttons[b].justPressed()) {
                    buttons[b].drawButton(true);    // draw invert!
                }
            }
        }
        delay(100);
}

void TouchNormalPin()
{
        digitalWrite(13, HIGH);
        TSPoint p = ts.getPoint();
        digitalWrite(13, LOW);
        //swap(p.x, p.y);
```

```
pinMode(XM, OUTPUT);
digitalWrite(XM, LOW);
pinMode(YP, OUTPUT);
digitalWrite(YP, HIGH);
pinMode(YM, OUTPUT);
digitalWrite(YM, LOW);
pinMode(XP, OUTPUT);
digitalWrite(XP, HIGH);

if (p.z > MINPRESSURE && p.z < MAXPRESSURE)//检测是否按键
{
    // scale from 0->1023 to tft.width
    p.x = (tft.width() - map(p.x, TS_MINX, TS_MAXX, tft.width(), 0));
    p.y = map(p.y, TS_MINY, TS_MAXY, tft.height(), 0);
}

if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
    Serial.print("X = "); Serial.print(p.x);
    Serial.print("\tY = "); Serial.print(p.y);
    Serial.print("\tPressure = "); Serial.println(p.z);
}

// go thru all the buttons, checking if they were pressed
if (p.z > MINPRESSURE && p.z < MAXPRESSURE) //检测是否按键
{
    for (uint8_t b = 0; b<4; b++) {
        if (buttons[b].contains(p.x, p.y)) {
            Serial.print("Pressing: "); Serial.println(b);
            if (buttons[b].isPressed())
            {
                buttons[b].press(false);
                switch (b)
                {
                case 0:      //
                    PumpON();
                    break;
                case 1:      //
                    ChangeStopTime();
                    break;
                case 2:      //
                    LightON();
                    break;
                case 3:      //
                    OtherON();
```

```
                        break;
                    }
                    //delay(200);
                    //GetSocketStatus();
                }
                else
                {
                    buttons[b].press(true);
                    switch (b)
                    {
                    case 0:       //
                        PumpOFF();
                        break;
                    case 1:       //
                        ChangeStopTime();
                        break;
                    case 2:       //
                        LightOFF();
                        break;
                    case 3:       //
                        OtherOFF();
                        break;
                    }
                    //delay(200);
                    //GetSocketStatus();
                }
            }
        }

        // now we can ask the buttons if their state has changed
        for (uint8_t b = 0; b<4; b++) {
            if (buttons[b].justReleased()) {
                // Serial.print("Released: "); Serial.println(b);
                if(b!=1) buttons[b].drawButton();    // draw normal
            }

            if (buttons[b].justPressed()) {
                if(b!=1) buttons[b].drawButton(true);    // draw invert!
            }
        }
        delay(2000);
    }
}
```

```
float Oldtemp = 0;
byte Oldsecond = 0;
byte Oldminute = 0;
byte Oldminute2 = 0;
byte OldMinute3 = 0;
bool bDrawPassButton = false;
int jjj = 0;

void loop(void) {

    Warning();   //蜂鸣器、短信报警

    WarnCall(); //电话报警

    GetWifiMessage();//获得开关发送信息

    tend = millis();
    if (tend - tstart>1000 || tend - tstart <0) //隔 1 秒获取一次时钟
    {
        Times = ReadTime();
        tstart = millis();
    }

    if (second % 10 == 0) //隔 10 秒获取一次水位
    {
        TestWaterLevel();
    }

    if (Oldminute2 != minute)//一分钟获取一次时钟,灯光定时
    {
        status(Times);
        Oldminute2 = minute;
        if (!bForceLight)   //强制开关灯时，不管定时
            LightTimer();
    }

    if (OldMinute3 != minute && minute % 5 == 0) //5 分钟发送一次数据，保持连接
    {
        OldMinute3 = minute;
        GetSocketStatus(); //获取开关状态
    }

    if (Oldminute != minute) //一分钟获取一次温度
    {
```

```
            Oldminute = minute;
            Tempreture = GetTemperature(Oldtemp);
            if (Tempreture >= TempHigh || Tempreture <= TempLow)
            {
                if (!bTempWarning)
                {
                    bTempWarning = true;
                    bSmsSend = false;
                }
            }
            else
                bTempWarning = false;
            GetGsmMessage();//获得 GSM 模块信息
        }
        String PrintText = "";

        for(int i=0;i<10;i++)
        {
            TouchNormalPin(); //响应屏幕触摸
            delay(10);
        }

        if (Oldtemp != Tempreture)
        {
            PrintText = "";
            PrintText = PrintText + " TEMP:" + Tempreture + " Water:"+WaterLevel;
            Oldtemp = Tempreture;
            Display(PrintText);
        }

        if (!bWiFi)
        {
            tft.setCursor(TEXT_X, TEXT_Y);
            tft.setTextColor(ILI9341_WHITE, ILI9341_BLACK);
            tft.setTextSize(2);
            tft.print("WiFi Connect Error");
        }

        delay(50); // UI debouncing
    }
```