Content

[Project overview 2](#_Toc154486471)

[Results display 3](#_Toc154486472)

[Networking mode 4](#_Toc154486473)

[The gyroscope mode 4](#_Toc154486474)

[Oscilloscope mode 5](#_Toc154486475)

[User mode 5](#_Toc154486476)

[Encounter problems 5](#_Toc154486477)

[Integral design 6](#_Toc154486478)

[Peripheral device 6](#_Toc154486479)

[SHT40 6](#_Toc154486480)

[JY931 8](#_Toc154486481)

[TLC5615 8](#_Toc154486482)

[ESP8266 8](#_Toc154486483)

[RGB3528 8](#_Toc154486484)

[SSD1306 8](#_Toc154486485)

[SC662K 8](#_Toc154486486)

[KEY 8](#_Toc154486487)

[PCB design 8](#_Toc154486488)

[Shell design 9](#_Toc154486489)

[Code implementation 9](#_Toc154486490)

[Application layer 10](#_Toc154486491)

[Protocol layer 13](#_Toc154486492)

[Driver layer 20](#_Toc154486493)

[Appendix 21](#_Toc154486494)

# Project overview

This design aims to realize a miniaturized integrated multifunctional intelligent terminal, so it is proposed that the design is capable of accomplishing the following functions:

* High-precision attitude sensing capability
* Intelligent device control terminal
* Wireless external sensor
* Intelligent temperature detection
* Portable DDS system
* Simple Oscilloscope

In order to realize the ability of high-precision attitude perception. We need to introduce the gyroscope as the attitude sensor, at the same time, in order to allow the user to perceive the current state in real time, we also need to introduce a display module for real-time display of the current state of the device, so we introduced the keys and the display as the medium of interaction, taking into account the pursuit of a smaller volume while taking into account the characteristics of the oscilloscope, so we gave up the use of the LCD screen but the use of OLED screen As the display terminal.

In order to realize the effect of A/D input and output, we choose PCF8591 as the auxiliary chip for ADC, and use TLC5615 as the output chip for DAC, so that the 51 microcontrollers has gained the ability of analog signal input and output.

In order to realize the intelligent terminal of the Internet of Everything, we use ESP8266 as a peripheral to access the network, and at the same time, we carry out secondary development in the Internet of Things server platform of Aliyun, so as to realize the detection of various real-time states of the terminal on the cell phone side.

For the temperature sensor, we use SHT40, which can directly read the temperature in combination with ADC, and also equipped with EEPROM as the memory device.

This design follows top-down design, starting from the function, designing the circuit board, then designing the product shell, and finally debugging the code to realize the function.

电子仪器

描述已自动生成

Figure 1 Rendering (computing)

In Figure 1, a rendering of the project is depicted using Fusion 360 modeling software. This rendering serves as a visual representation of the design or model created within the Fusion 360 environment. Key aspects to consider when describing Figure 1 include:

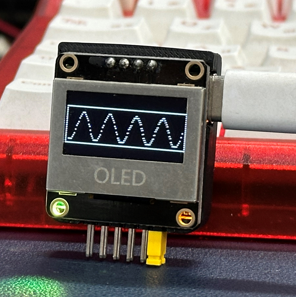


Figure 2 Product Picture

The final implementation and realization of the project are illustrated in Figure 2.

# Results display

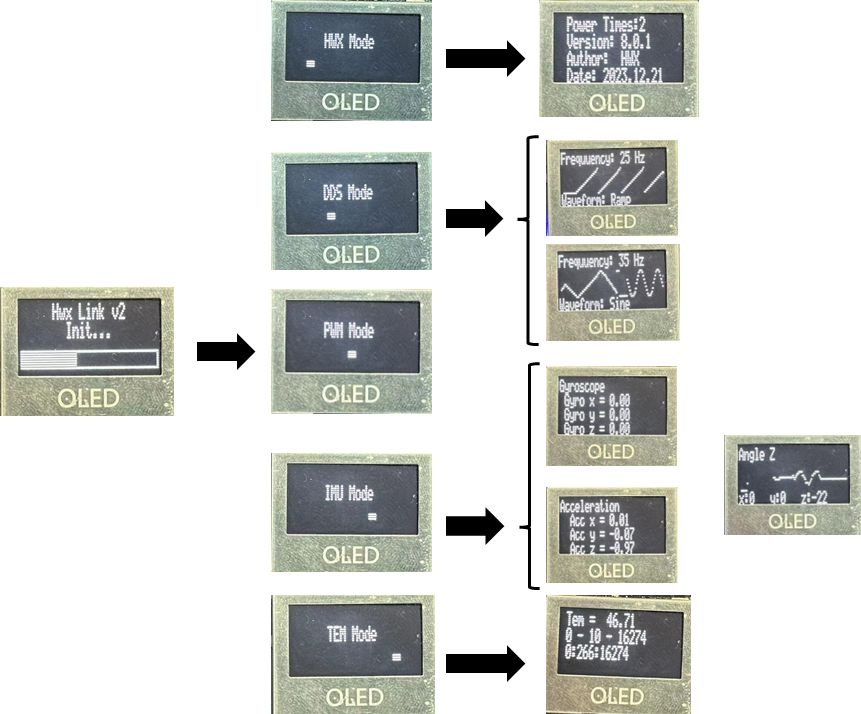


Figure 3 Functional diagram

The figure shows the individual modes that have been implemented, as well as the human-computer interactions

## Networking mode

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Figure 4 Real-time status updates

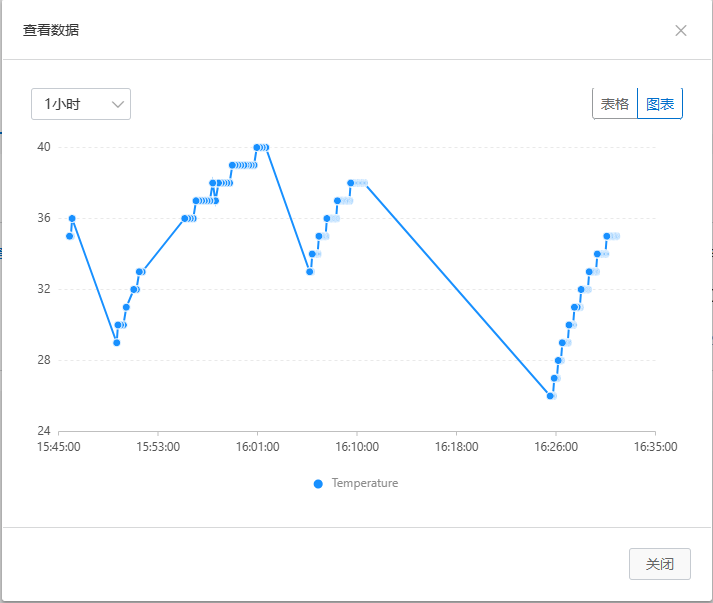


Figure 5 Cloud Server Detection

## The gyroscope mode

## Oscilloscope mode

## User mode

# Encounter problems

#### 1. Type-C Interface Soldering and CC Protocol:

**Soldering Challenges:**

* Discuss the specific challenges faced due to the small spacing between the 12 pins of the Type-C interface.
* Emphasize the difficulty in achieving precise soldering and how it impacted the overall assembly process.

**CC Protocol Understanding:**

* Explain the lack of understanding regarding the CC (Communication Channel) protocol in the initial version of the board.
* Highlight the consequences of not having access to the 5.1kΩ pull-down resistor on CC1 and CC2 pins, leading to the reversal of Type-C's VCC and GND pins.

#### 2. TLC5615 Output Level Issue:

**Reference Source Requirement:**

* Elaborate on the discovery made after reading the TLC5615 chip manual about its need for a reference source during operation.
* Discuss how the absence of the reference source resulted in incorrect output levels from the TLC5615.

#### 3. OLED Screen Frame Rate Optimization:

**Initial Frame Rate Issues:**

* Detail the initial challenges with the OLED screen, particularly the low frame rate impacting the interactive experience.
* Mention user experience concerns and potential usability issues.

**Optimization Efforts:**

* Describe the steps taken to optimize the IIC communication rate and improve the algorithm for displaying text and graphics.
* Highlight the positive impact of these optimizations on the frame rate and overall user experience.

#### 4. Serial Port Message Handling:

**Function Packaging Issue:**

* Explain the challenge with the serial port functionality caused by packaging the initialization into a function that adjusted the baud rate.
* Discuss the array overrun issue in Saitama and how it affected proper message handling.

**Debugging with OLED Screen:**

* Share how the OLED screen was utilized as a debugging device to identify and address the array overrun issue.
* Emphasize the importance of debugging tools in resolving complex issues.

#### 5. ESP8266 Connectivity and Pass-Through Mode:

**Anomalous WiFi Connection:**

* Discuss the abnormal ESP8266 connectivity issues and the inability to set the pass-through mode.
* Mention the reliance on the ESP official website for relevant instructions and the subsequent successful resolution.

#### 6. MQTT Protocol Understanding:

**Message Reception Issue:**

* Explain the problem with the server not receiving uploaded messages, attributing it to a lack of familiarity with the MQTT protocol.
* Detail the process of checking and resetting the MQTT protocol to address the communication issue.

# Integral design

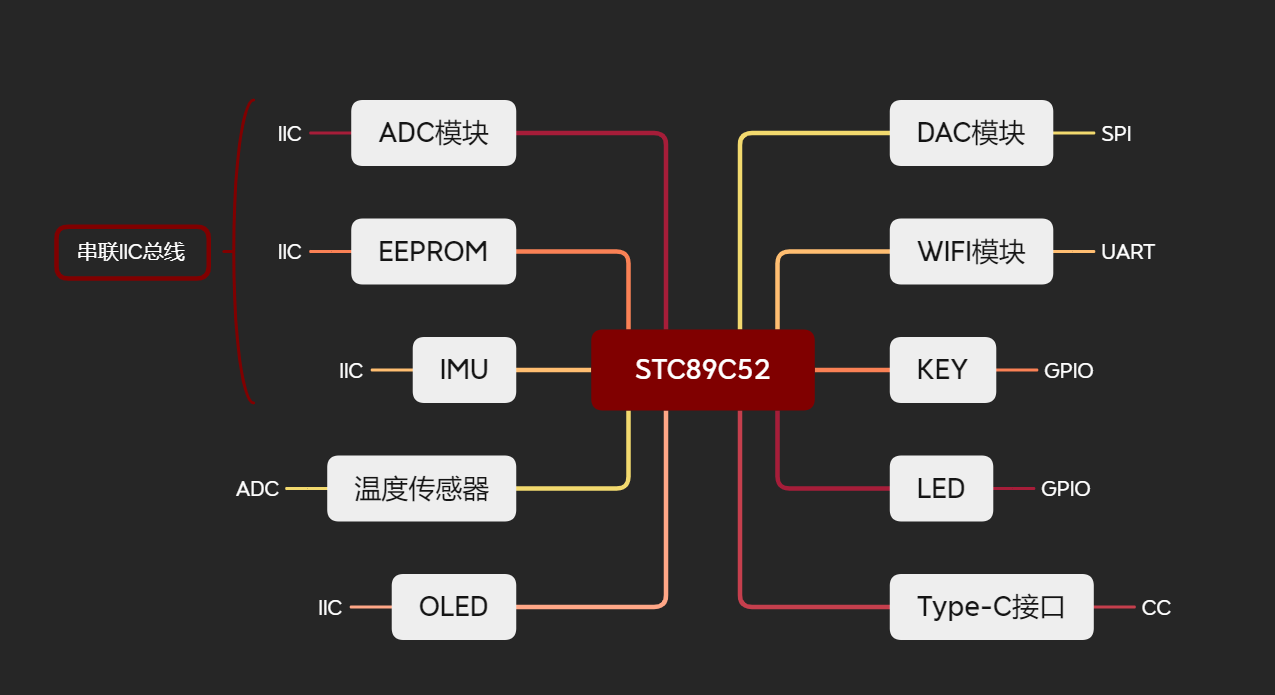


Figure 7 Embedded Architecture

## Peripheral device

图示

描述已自动生成

Figure 6 Product Composition

### SHT40

The SHT40 is a digital humidity and temperature sensor belonging to Sensirion's SHT series of products.In this product, we obtain the temperature and send it back to the MCU by means of ADC.



**Functionality and Purpose:**

Humidity Measurement: The SHT40 can measure the humidity level in the surrounding environment, providing highly accurate humidity data.

Temperature Measurement: In addition to humidity, the SHT40 can accurately measure the ambient temperature, providing temperature data.

**Digital Sensor:**

The SHT40 is a digital sensor that communicates with microcontrollers or other digital systems through a digital interface, typically I2C. This facilitates convenient data acquisition and processing.

High Accuracy and Stability:

The SHT40 is known for its high accuracy and stability, making it suitable for applications with stringent humidity and temperature requirements, such as weather measurements, industrial automation, and electronic devices.

**Temperature and Humidity Calibration:**

Sensors are typically calibrated during the manufacturing process to ensure accurate humidity and temperature readings. Some models also support user calibration during runtime.

**Low Power Consumption:**

The SHT40 generally has low power consumption during operation, making it suitable for battery-powered devices requiring extended operation.

**Applications:**

The SHT40 sensor finds widespread application in various scenarios that demand monitoring of environmental humidity and temperature, including but not limited to weather stations, greenhouses, industrial process monitoring, and consumer electronic devices.

图示

描述已自动生成

### TLC5615

The TLC5615 is a Digital-to-Analog Converter (DAC) produced by Texas Instruments. Here are some basic details about the TLC5615:Uses SPI to communicate with microcontroller

Features:

**Digital-to-Analog Conversion**:

The TLC5615 is primarily used to convert digital signals into analog voltage output. It transforms digital input data into corresponding analog voltages, suitable for driving various analog circuits and sensors.

**Resolution:**

The TLC5615 typically has a resolution of 12 bits, indicating that it can generate 2^12 (4096) different output voltage levels.

**SPI Interface:** It typically communicates using the Serial Peripheral Interface (SPI), making integration with digital systems like microcontrollers convenient.

Reference Voltage: The TLC5615 usually requires an external reference voltage to ensure the accuracy of the digital-to-analog conversion.

**Applications:**

Analog Signal Generation: Mainly used for generating various analog signals, such as audio waveforms, control voltages, or analog outputs from sensors.

Industrial Automation: In industrial control systems, the TLC5615 can be used to generate precise control analog voltages.

**Instrumentation:**

In measurement and testing equipment, the TLC5615 is employed to produce accurate analog outputs.

**Operation Principle:**

The TLC5615 utilizes serial input to receive digital data, which is then converted into the corresponding analog output voltage. Typically, it includes a voltage output buffer to ensure accuracy and stability in the output.

**Supply Voltage:**

The TLC5615 usually operates within a specific supply voltage range to ensure optimal performance.

## PCB design

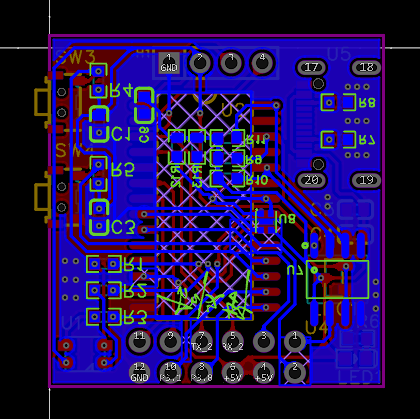
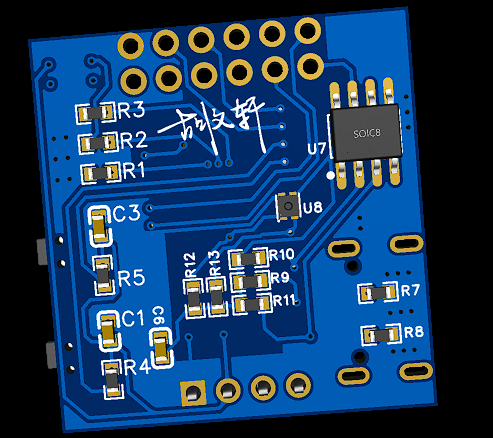
 

Figure 8 PCB

**Compact Size and Rich Interface:**

The design prioritizes a small form factor while maximizing interface versatility.

Despite its compact size, the PCB incorporates a rich set of interfaces to accommodate various peripherals and devices.

**ESP8266 Integration for IoT Development:**

The inclusion of the ESP8266 module empowers the PCB with networking capabilities, facilitating seamless IoT (Internet of Things) development.

This integration allows for wireless communication, enabling the PCB to connect to networks and exchange data in IoT applications.

**High-Performance IMU (Inertial Measurement Unit):**

The integration of the JY931 high-performance IMU enhances the PCB's capabilities for precise motion sensing and orientation calculations.

This feature is particularly beneficial for applications requiring accurate pose estimation and motion tracking.

**Stable Power Supply with LDO 5 to 3.3V Conversion:**

The design incorporates a stable power supply system featuring an LDO (Low Drop-Out) regulator for efficient voltage conversion from 5V to 3.3V.

The utilization of the SC662K chip ensures a reliable and stable operating voltage, crucial for consistent and uninterrupted performance.

**Versatile GPIO, ADC, UART, and Type-C I/O:**

The PCB offers multiple GPIO (General Purpose Input/Output) pins, accommodating a variety of external devices and sensors.

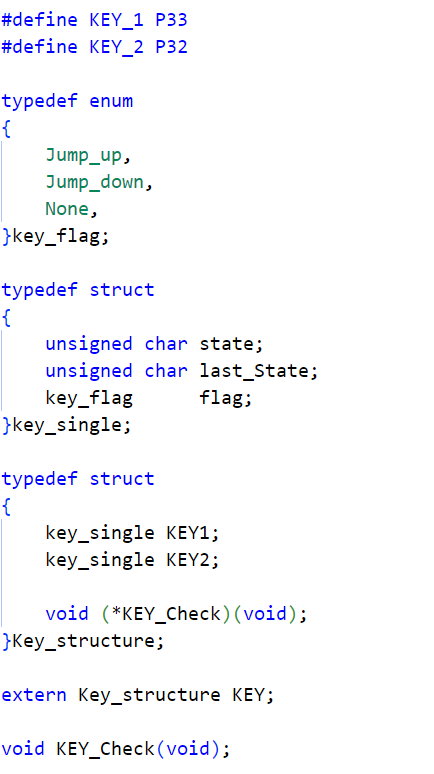
Two ADC (Analog-to-Digital Converter) channels enable analog signal processing, while one UART (Universal Asynchronous Receiver-Transmitter) interface facilitates serial communication.

The inclusion of a Type-C input/output interface adds modern and versatile connectivity, enhancing the PCB's compatibility with contemporary devices.

In summary, my PCB design embodies the principles of compactness, connectivity, and performance. The integration of advanced components, such as the ESP8266, JY931 IMU, and SC662K chip, enables the PCB to excel in IoT applications, motion sensing, and stable power management. The diverse set of interfaces further enhances the versatility of the design, making it well-suited for a wide range of electronic projects.

# Code implementation

Use C language to realize object-oriented ideas, encapsulate the module into a structure, to facilitate secondary development and subsequent calls.



图示

描述已自动生成

Figure 9 Code framework schematic

The design code implementation framework is shown in the figure above.

## Application layer

#### Main Function

int main()

{

/\*初始化开始\*/

GPIO\_Init();

OLED\_Init();

ADC\_Init();

USART1\_Init();

ESP8266\_Init();

Queue\_Init(&ADC\_queue);

/\*记录上电次数\*/

EEPROM\_read\_n(0x00,&PowerOnTimes[0],1);

// \_nop\_();

PowerOnTimes[0]++;

// \_nop\_();

EEPROM\_write\_n(0x00,&PowerOnTimes[0],1);

OLED\_bar(85);

/\*初始化完成\*/

OLED\_Clear();

while(1)

{

switch (mode)

{

case MAIN\_MODE:

if(KEY\_1\_cnt % CNT\_MODE == MAIN\_MODE)

{

unsigned char ch[22] = {0};

/\*预处理\*/

LED\_BLUE\_OFF();

LED\_GREEN\_OFF();

LED\_RED\_OFF();

/\*标志位清零\*/

first2bmi = 0;

first2tem = 0;

if(KEY\_2\_cnt % 2 == 1)

{

sprintf(ch, " Power Times:%d ", PowerOnTimes[0]);

OLED\_ShowString(0, 0, (uint8\_t \*)ch, 20);

sprintf(ch, " Version: 8.0.1");

OLED\_ShowString(0, 1, (uint8\_t \*)ch, 20);

sprintf(ch, " Author: HWX");

OLED\_ShowString(0, 2, (uint8\_t \*)ch, 20);

sprintf(ch, " Date: 2023.12.21");

OLED\_ShowString(0, 3, (uint8\_t \*)ch, 20);

}

else

{

sprintf(ch, " HWX Mode ");

OLED\_drawblock(20);

OLED\_ShowString(0, 1, (uint8\_t \*)ch, 20);

}

}

else if(KEY\_1\_cnt % CNT\_MODE == ADC\_MODE)

{

unsigned char ch[22] = {0};

sprintf(ch, " DDS Mode ");

EA = 0;

OLED\_drawblock(40);

OLED\_ShowString(0, 1, (uint8\_t \*)ch, 20);

}

else if(KEY\_1\_cnt % CNT\_MODE == PWM\_MODE)

{

unsigned char ch[22] = {0};

sprintf(ch, " PWM Mode ");

OLED\_drawblock(60);

OLED\_ShowString(0, 1, (uint8\_t \*)ch, 20);

}

else if(KEY\_1\_cnt % CNT\_MODE == IMU\_MODE)

{

unsigned char ch[22] = {0};

sprintf(ch, " IMU Mode ");

/\*预处理\*/

EA = 1;

first2bmi = 1;

OLED\_drawblock(80);

OLED\_ShowString(0, 1, (uint8\_t \*)ch, 20);

}

else if(KEY\_1\_cnt % CNT\_MODE == TEM\_MODE)

{

unsigned char ch[22] = {0};

sprintf(ch, " TEM Mode ");

/\*预处理\*/

EA = 1;

first2tem = 1;

OLED\_drawblock(100);

OLED\_ShowString(0, 1, (uint8\_t \*)ch, 20);

}

break;

case ADC\_MODE:

ADC\_mode\_control();

MODE\_Return();

break;

case PWM\_MODE:

PWM\_mode\_control();

MODE\_Return();

break;

case IMU\_MODE:

BMI\_mode\_control();

MODE\_Return();

break;

case TEM\_MODE:

TEM\_mode\_control();

MODE\_Return();

break;

default:

break;

}

}

return 0;

}

## Protocol layer

#### ESP8266 Protocol

#include "esp8266.h"

#include "JY901.h"

#include "key.h"

uint8\_t AT\_ok\_flag = 0;

extern int16\_t Pitch\_,Yaw\_,Roll\_,Tem\_;

uint32\_t wifi\_cnt = 0;

void ESP8266\_Init(void)

{

unsigned char ch[22] = {0};

sprintf(ch, " Hwx Link v2 ");

OLED\_ShowString(0, 0, (uint8\_t \*)ch, 16);

sprintf(ch, " Init... ");

OLED\_ShowString(0, 1, (uint8\_t \*)ch, 16);

OLED\_drawBOX();

delay\_250ms();

PrintString2("ATE0\r\n"); //发送AT测试指令

while(!AT\_ok\_flag);

AT\_ok\_flag = 0;

OLED\_bar(20);

PrintString2("AT+CWMODE=1\r\n");

while(!AT\_ok\_flag); //发送AT测试指令

AT\_ok\_flag = 0;

OLED\_bar(40);

PrintString2("AT+CWJAP=\"HWX\",\"2021280479\"\r\n");

while(!AT\_ok\_flag)

{

if(wifi\_cnt++ == 50000)

{

PrintString2("AT+CWJAP=\"HWX\",\"2021280479\"\r\n");

LED\_BLUE\_ON();

wifi\_cnt = 0;

}

KEY\_Check();

if(KEY.KEY1.flag == Jump\_down)

{

goto herehere;

}

;

}

AT\_ok\_flag = 0;

OLED\_bar(50);

PrintString2("AT+CIPSTART=\"TCP\",\"a1Me6XseN3i.iot-as-mqtt.cn-shanghai.aliyuncs.com\",1883\r\n");

while(!AT\_ok\_flag);

AT\_ok\_flag = 0;

OLED\_bar(60);

PrintString2("AT+CIPMODE=1\r\n");

while(!AT\_ok\_flag);

AT\_ok\_flag = 0;

OLED\_bar(70);

PrintString2("AT+CIPSEND\r\n");

while(!AT\_ok\_flag);

AT\_ok\_flag = 0;

Tcp\_Connect();

OLED\_bar(75);

}

#define WIFI\_Send(x) PrintString2(x)

#define Send\_Uart(x) TX2\_write2buff(x)

void Tcp\_Connect()

{

Send\_Uart(0x10); //固定报头

Send\_Uart(0x70); //剩余长度=0X10＋客户端ID长度+用户名长度+密码长度的16进制

Send\_Uart(0x00); //可变报头

Send\_Uart(0x04); //可变报头

WIFI\_Send("MQTT"); //可变报头

Send\_Uart(0x04); //可变报头

Send\_Uart(0xC2); //可变报头

Send\_Uart(0x00); //可变报头

Send\_Uart(0x64); //可变报头

Send\_Uart(0x00); //客户端ID长度00 28 即40字节

Send\_Uart(0x27);

WIFI\_Send("esp01|securemode=3,signmethod=hmacsha1|"); //客户端ID

Send\_Uart(0x00); //用户名长度00 12 即18字节

Send\_Uart(0x11);

WIFI\_Send("esp01&a1Me6XseN3i"); //用户名

Send\_Uart(0x00); //密码长度00 28 即40字节

Send\_Uart(0x28);

WIFI\_Send("a5c0c2dae092876cf199d0a1e479bb7bcae29e90"); //密码

}

u8 tabR[4] = {0}; //温度缓存单元

u8 tabP[4] = {0}; //湿度缓存单元

u8 tabY[4] = {0}; //湿度缓存单元

u8 tabT[4] = {0}; //温度缓存单元

void Up\_Report\_RP()

{

// Roll\_ = 120;

// Pitch\_ = 143;

tabR[0] = Roll\_ / 100 + 0x30;

tabR[1] = Roll\_ / 10 % 10 + 0x30;

tabR[2] = Roll\_ % 10 + 0x30;

tabP[0] = Pitch\_ / 100 + 0x30;

tabP[1] = Pitch\_ / 10 % 10 + 0x30;

tabP[2] = Pitch\_ % 10 + 0x30;

Send\_Uart(0x30); //固定报头

Send\_Uart(0x9A); //剩余长度 9A

Send\_Uart(0x01);

Send\_Uart(0x00); //可变报头

Send\_Uart(0x30);

WIFI\_Send("/sys/a1Me6XseN3i/esp01/thing/event/property/post");

WIFI\_Send("{\"method\":\"thing.event.property.post\",\"id\":\"395041230\",\"params\":{\"");

WIFI\_Send("Pit"); //标识符Tem

WIFI\_Send("\":");

WIFI\_Send(&tabP);

WIFI\_Send(",\"Rol"); //标识符Hum

WIFI\_Send("\":");

WIFI\_Send(&tabR);

WIFI\_Send("},\"version\":\"1.0.0\"}");

}

void Up\_Report\_YT()

{

// Yaw\_ = 150;

// Tem\_ = 50;

tabY[0] = Yaw\_ / 100 + 0x30;

tabY[1] = Yaw\_ / 10 % 10 + 0x30;

tabY[2] = Yaw\_ % 10 + 0x30;

tabT[0] = Tem\_ / 100 + 0x30;

tabT[1] = Tem\_ / 10 % 10 + 0x30;

tabT[2] = Tem\_ % 10 + 0x30;

Send\_Uart(0x30); //固定报头

Send\_Uart(0x9A); //剩余长度 9A

Send\_Uart(0x01);

Send\_Uart(0x00); //可变报头

Send\_Uart(0x30);

WIFI\_Send("/sys/a1Me6XseN3i/esp01/thing/event/property/post");

WIFI\_Send("{\"method\":\"thing.event.property.post\",\"id\":\"395041230\",\"params\":{\"");

WIFI\_Send("Yaw"); //标识符Tem

WIFI\_Send("\":");

WIFI\_Send(&tabY);

WIFI\_Send(",\"Tem"); //标识符Hum

WIFI\_Send("\":");

WIFI\_Send(&tabT);

WIFI\_Send("},\"version\":\"1.0.0\"}");

}

#### IMU Protocol

int16\_t sReg[REGSIZE];

static uint32\_t s\_uiWitDataCnt = 0, s\_uiProtoclo = 0, s\_uiReadRegIndex = 0;

static volatile char s\_cDataUpdate = 0, s\_cCmd = 0xff;

static uint8\_t s\_ucWitDataBuff[WIT\_DATA\_BUFF\_SIZE];

extern uint8\_t YY,MM,DD,HH,MN,SS;

static uint8\_t \_\_CaliSum(uint8\_t \*data\_, uint32\_t len)

{

uint32\_t i;

uint8\_t ucCheck = 0;

for(i=0; i<len; i++) ucCheck += \*(data\_ + i);

return ucCheck;

}

void WitSerialDataIn(uint8\_t ucData)

{

uint16\_t usCRC16, usTemp, i, usData[4];

uint8\_t ucSum;

s\_ucWitDataBuff[s\_uiWitDataCnt++] = ucData;

switch(s\_uiProtoclo)

{

case WIT\_PROTOCOL\_NORMAL:

if(s\_ucWitDataBuff[0] != 0x55)

{

s\_uiWitDataCnt--;

memcpy(s\_ucWitDataBuff, &s\_ucWitDataBuff[1], s\_uiWitDataCnt);

return ;

}

if(s\_uiWitDataCnt >= 11)

{

ucSum = \_\_CaliSum(s\_ucWitDataBuff, 10);

if(ucSum != s\_ucWitDataBuff[10])

{

s\_uiWitDataCnt--;

memcpy(s\_ucWitDataBuff, &s\_ucWitDataBuff[1], s\_uiWitDataCnt);

return ;

}

usData[0] = ((uint16\_t)s\_ucWitDataBuff[3] << 8) | (uint16\_t)s\_ucWitDataBuff[2];

usData[1] = ((uint16\_t)s\_ucWitDataBuff[5] << 8) | (uint16\_t)s\_ucWitDataBuff[4];

usData[2] = ((uint16\_t)s\_ucWitDataBuff[7] << 8) | (uint16\_t)s\_ucWitDataBuff[6];

usData[3] = ((uint16\_t)s\_ucWitDataBuff[9] << 8) | (uint16\_t)s\_ucWitDataBuff[8];

CopeWitData(s\_ucWitDataBuff[1], usData, 4);

s\_uiWitDataCnt = 0;

}

break;

}

if(s\_uiWitDataCnt == WIT\_DATA\_BUFF\_SIZE)s\_uiWitDataCnt = 0;

}

static void SensorDataUpdata(uint32\_t uiReg, uint32\_t uiRegNum)

{

int i;

for(i = 0; i < uiRegNum; i++)

{

switch(uiReg)

{

// case AX:

// case AY:

case AZ:

s\_cDataUpdate |= ACC\_UPDATE;

break;

// case GX:

// case GY:

case GZ:

s\_cDataUpdate |= GYRO\_UPDATE;

break;

// case HX:

// case HY:

case HZ:

s\_cDataUpdate |= MAG\_UPDATE;

break;

// case Roll:

// case Pitch:

case Yaw:

s\_cDataUpdate |= ANGLE\_UPDATE;

break;

default:

s\_cDataUpdate |= READ\_UPDATE;

break;

}

uiReg++;

}

}

static void CopeWitData(uint8\_t ucIndex, uint16\_t \*p\_data, uint32\_t uiLen)

{

uint32\_t uiReg1 = 0, uiReg2 = 0, uiReg1Len = 0, uiReg2Len = 0;

uint16\_t \*p\_usReg1Val = p\_data;

uint16\_t \*p\_usReg2Val = p\_data+3;

uiReg1Len = 4;

switch(ucIndex)

{

case WIT\_ACC: uiReg1 = AX; uiReg1Len = 3; uiReg2 = TEMP; uiReg2Len = 1; break;

case WIT\_ANGLE: uiReg1 = Roll; uiReg1Len = 3; uiReg2 = VERSION; uiReg2Len = 1; break;

case WIT\_TIME: YY = (uint8\_t)(\*p\_data) >> 8;

MM = (uint8\_t)(\*p\_data) & 0xff;

DD = (uint8\_t)(\*(p\_data + 1)) >> 8;

HH = (uint8\_t)(\*(p\_data + 1)) & 0xff;

MN = (uint8\_t)(\*(p\_data + 2)) >> 8;

SS = (uint8\_t)(\*(p\_data + 2)) & 0xff;

break;

case WIT\_GYRO: uiReg1 = GX; uiLen = 3;break;

case WIT\_MAGNETIC: uiReg1 = HX; uiLen = 3;break;

case WIT\_DPORT: uiReg1 = D0Status; break;

case WIT\_PRESS: uiReg1 = PressureL; break;

case WIT\_GPS: uiReg1 = LonL; break;

case WIT\_VELOCITY: uiReg1 = GPSHeight; break;

case WIT\_QUATER: uiReg1 = q0; break;

case WIT\_GSA: uiReg1 = SVNUM; break;

case WIT\_REGVALUE: uiReg1 = s\_uiReadRegIndex; break;

default:

return ;

}

if(uiLen == 3)

{

uiReg1Len = 3;

uiReg2Len = 0;

}

if(uiReg1Len)

{

memcpy(&sReg[uiReg1], p\_usReg1Val, uiReg1Len<<1);

SensorDataUpdata(uiReg1, uiReg1Len);

}

if(uiReg2Len)

{

memcpy(&sReg[uiReg2], p\_usReg2Val, uiReg2Len<<1);

SensorDataUpdata(uiReg2, uiReg2Len);

}

## Driver layer

#### Key Driver

void Timer2\_it() interrupt TIMER2\_VECTOR

{

KEY\_Check();

if(KEY.KEY1.flag == Jump\_down)

{

KEY\_1\_cnt++;

if(mode == MAIN\_MODE)

{

OLED\_Clear();

}

}

if(KEY.KEY2.flag == Jump\_down)

{

KEY\_2\_cnt++;

if(mode == MAIN\_MODE)

{

mode = KEY\_1\_cnt%CNT\_MODE;

OLED\_Clear();

}

}

}

#### LED Driver

sbit RGB\_B = P3^6;

sbit RGB\_R = P3^5;

sbit RGB\_G = P3^4;

#define LED\_RED\_ON() (RGB\_R = 0)

#define LED\_RED\_OFF() (RGB\_R = 1)

#define LED\_GREEN\_ON() (RGB\_G = 0)

#define LED\_GREEN\_OFF() (RGB\_G = 1)

#define LED\_BLUE\_ON() (RGB\_B = 0)

#define LED\_BLUE\_OFF() (RGB\_B = 1)

#define BLACK\_ON() LED\_RED\_OFF(); LED\_GREEN\_OFF();LED\_BLUE\_OFF()

#define RED\_ON() LED\_RED\_ON(); LED\_GREEN\_OFF();LED\_BLUE\_OFF()

#define GREEN\_ON() LED\_RED\_OFF(); LED\_GREEN\_ON();LED\_BLUE\_OFF()

#define BLUE\_ON() LED\_RED\_OFF(); LED\_GREEN\_OFF();LED\_BLUE\_ON()

#define YELLOW\_ON() LED\_RED\_ON(); LED\_GREEN\_ON();LED\_BLUE\_OFF()

#define PINK\_ON() LED\_RED\_ON(); LED\_GREEN\_OFF();LED\_BLUE\_ON()

#define LIGHT\_ON() LED\_RED\_OFF(); LED\_GREEN\_ON();LED\_BLUE\_ON()

#define WHITE\_ON() LED\_RED\_ON(); LED\_GREEN\_ON();LED\_BLUE\_ON()

# Appendix