# Smart home

**Introduction**

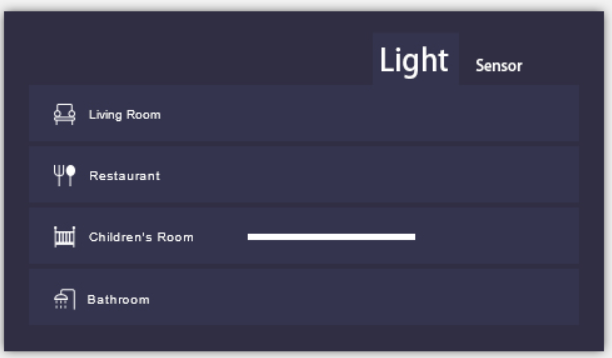
Today, I got STONE's serial port drive display, which can communicate through MCU's serial port, and the UI logic design of this display can be designed directly by using the VGUS software provided on STONE's official website, which is very convenient for us. So I plan to use it to make a simple appliance controller, which includes the control of various lights (living room, kitchen, children's room, bathroom).At the same time, indoor and outdoor temperature, humidity and air quality can be collected. This is just a simple demo, and you can conduct secondary development through the code I provided.Some basic tutorial about STONE screen, can go to website: <https://www.stoneitech.com/>

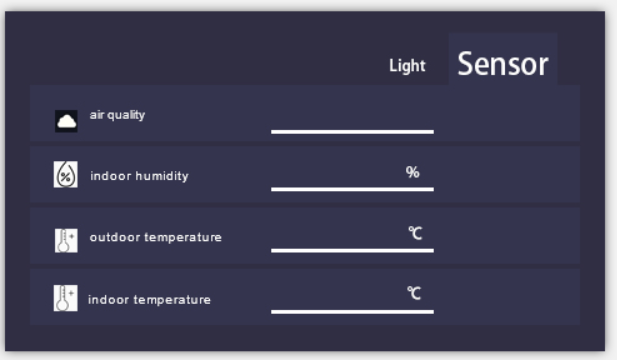
The website has a variety of information about the model, user and design documentation, as well as video tutorials.I won't go into too much detail here.

## UI interface design

## Photoshop

I designed the following two UI pages with photoshop:

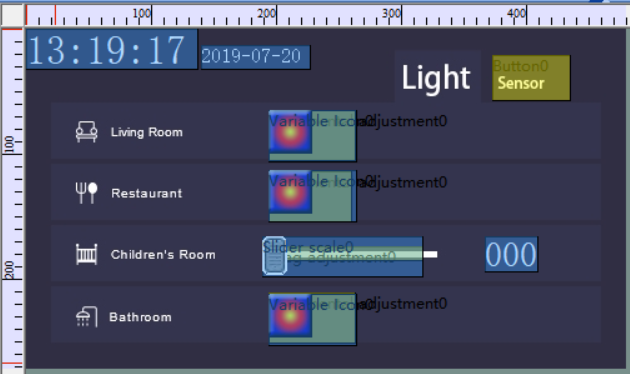


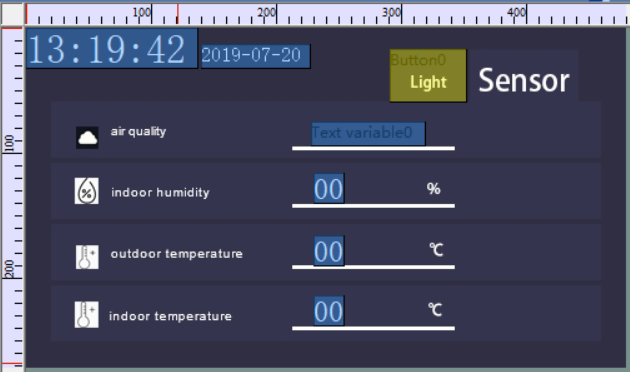


This project has the above two pages in total. "Light" and "Sensor" in the upper right corner are the switch buttons of these two pages.

In the "Light" page, you can control all kinds of lights in your home. In the "Sensor" page, you can check the values detected by various sensors.

After the design of the above two pages, we can conduct button logic design through the VGUS software provided on STONE's official website.





It is worth noting that the clock source used for the time display here is the clock source of the display screen, not the MCU clock source.

## TAB page switching effect

No TAB page switching component was found in the VGUS software, so I thought of another method to achieve TAB page switching effect.

Through the observation I provide two UI images can be found that the two images above are "Light" and "Sensor" text, the difference is their pixel size is different, so we only need to put the two pixel position is set to the same text, and then through the upper left corner of the time and date for reference, you can achieve the TAB to switch effect.

## Button logic

Take the "Living Room" button as an example. When the user presses this button, the STONE serial port display screen will send corresponding protocol instructions through the serial port. After receiving this instruction, the user's MCU will parse the protocol to control the switching state of the lights connected with the MCU.

## Sensor acquisition

Take "air quality" for example: if you want to get the indoor air quality, we must have a MCU to collect air quality, air quality sensor when the MCU numerical collected through algorithm comparing the pros and cons of air quality, and then the MCU sent via a serial port to display the storage area of "Good" or "Bad", to change "Text variable0" display content, and then the user can intuitively see the merits of the quality control.

These are explained later in the MCU code.

# MCU communication

STM32 is the MCU that everyone is familiar with, and it is a common MCU model in international. Therefore, the specific model of STM32 MCU I used in this project is STM32F103RCT6.

There are many series of STM32, which can meet various demands of the market. The kernel can be divided into cortex-m0, M3, M4 and M7, and each kernel can be divided into mainstream, high performance and low power consumption.

Purely from the perspective of learning, you can choose F1 and F4, F1 represents the basic type, based on the cortex-m3 kernel, the main frequency is 72MHZ, F4 represents the high performance, based on the cortex-m4 kernel, the main frequency is 180M.

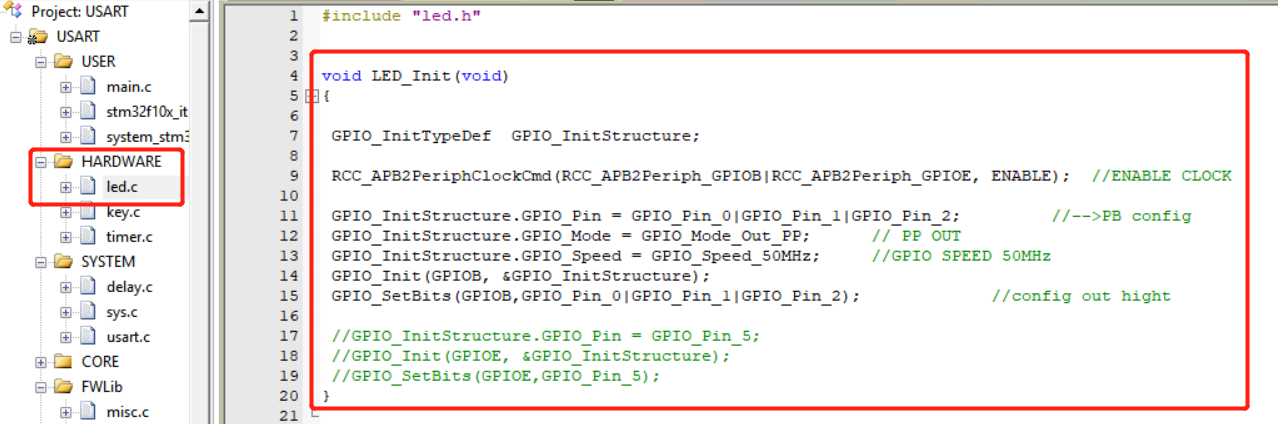
As for F1, F4 (429 series and above), apart from different kernel and improvement of main frequency, the obvious feature of upgrade is LCD controller and camera interface, support for SDRAM, this difference will be given priority in project selection. However, from the perspective of university teaching and users' initial learning, F1 series is still the first choice. Currently, the STM32 of F1 series has the largest amount of materials and products in the market.

About STM32 SCM development environment installation and program download method I will not do the introduction.

## GPIO initialization

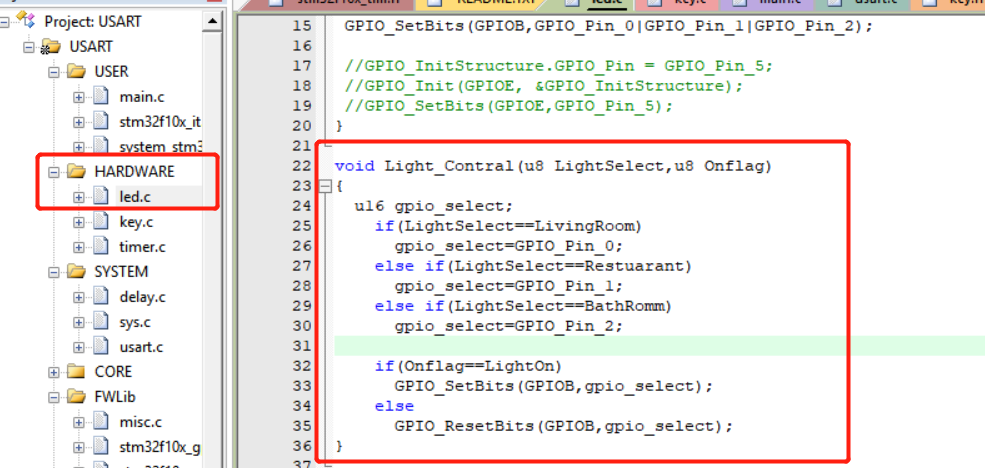
In this project, we used a total of 4 GPIO, one of which is PWM output pin.

Let's first look at the initialization of three ordinary GPIO ports:



This function initializes the PB0\PB1\PB2 of STM32F103C8 as the output pin and calls it from the main function.

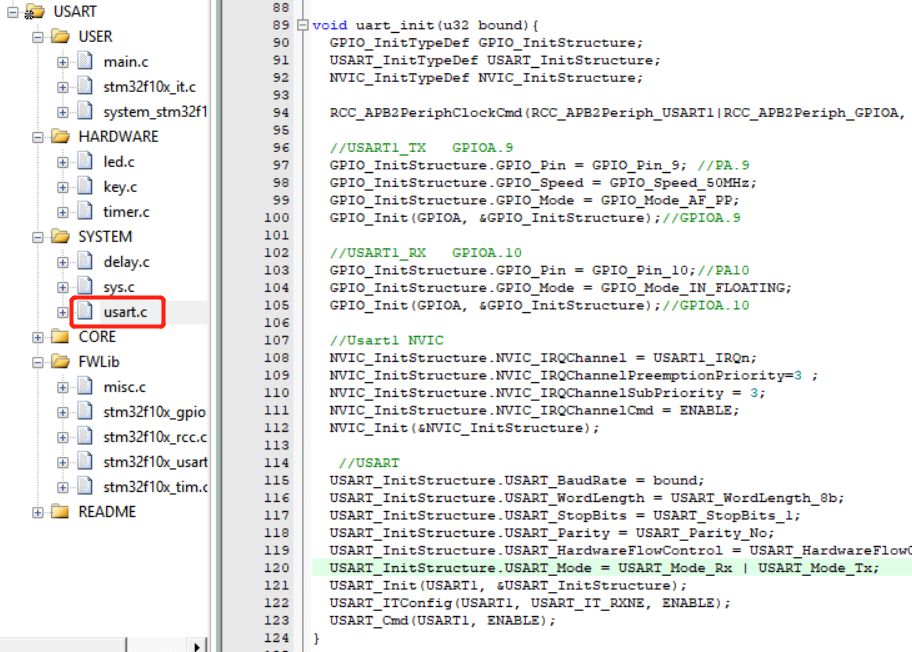
After initialization, we need to have a logic to control the output state, high and low level of these GPIO, so I wrote the function as below:



This is a function that you can intuitively understand by the name of the variable.

## Serial port initialization

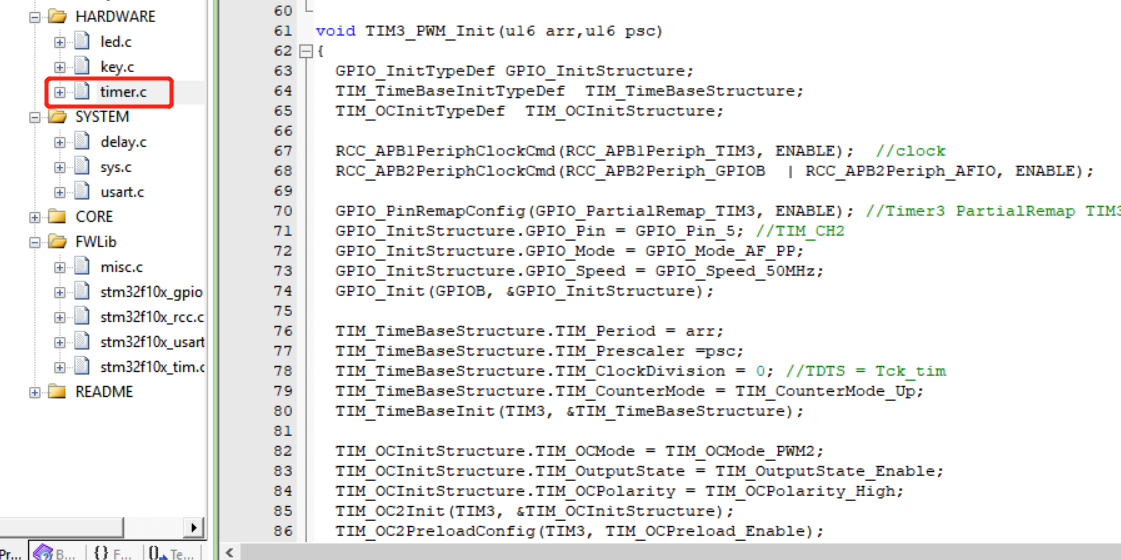
The initialization part of serial port is in uart.c:



Then call uart\_init in the main function to initialize the serial port baud rate of 115200.

Pins use PA9/PA10

## PWM initialization



Specific steps:

1. Set RCC clock;

2. Set GPIO clock;The GPIO mode should be set to GPIO\_Model\_AF\_PP, or to the GPIO\_PinRemapConfig() function if pin remap is required.

3. Set relevant registers of TIMx timer;

4. Set PWM related register of TIMx timer;

A. Set PWM mode

B. Set duty cycle (formula calculation)

C. Set the output comparison polarity (previously introduced)

D. Most importantly, enable the output state of TIMx and enable PWM output of TIMx;

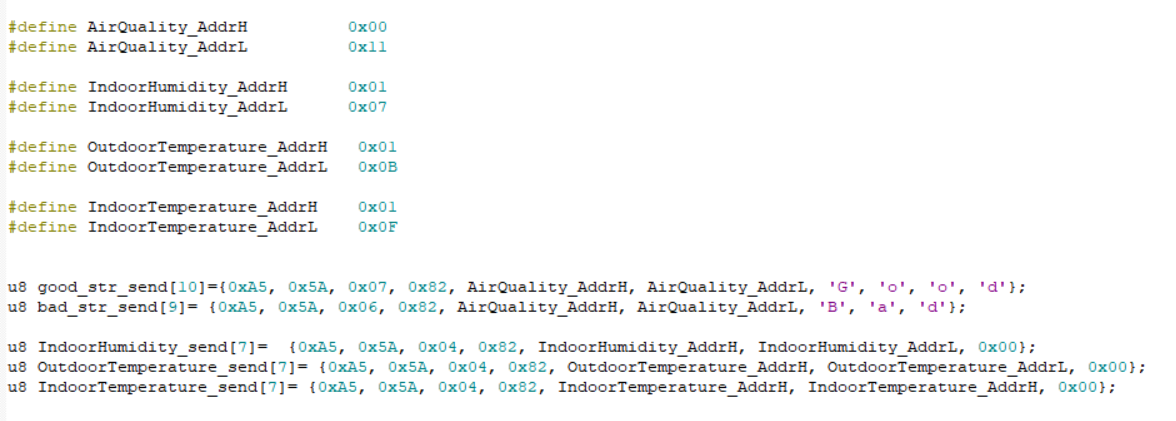
After the relevant Settings are completed, TIMx timer is turned on by TIMx\_Cmd () to obtain PWM output.

Call this TIM3\_PWM\_Init from the main function.

# Logic code writing

## Display component address definition

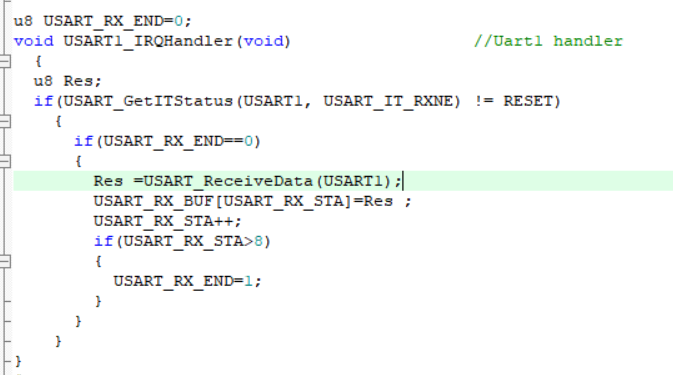
The components of the display have separate addresses, and here I've written them all as macro definitions:



## Serial data reception

Looking at information about STONE display, you can see that when the button is pressed, the serial port on the display sends protocols in the appropriate format, which the user MCU can receive and parse.When the button is pressed, the serial port on the display sends nine bytes of data, including user data.

Serial data reception is written in Handler:

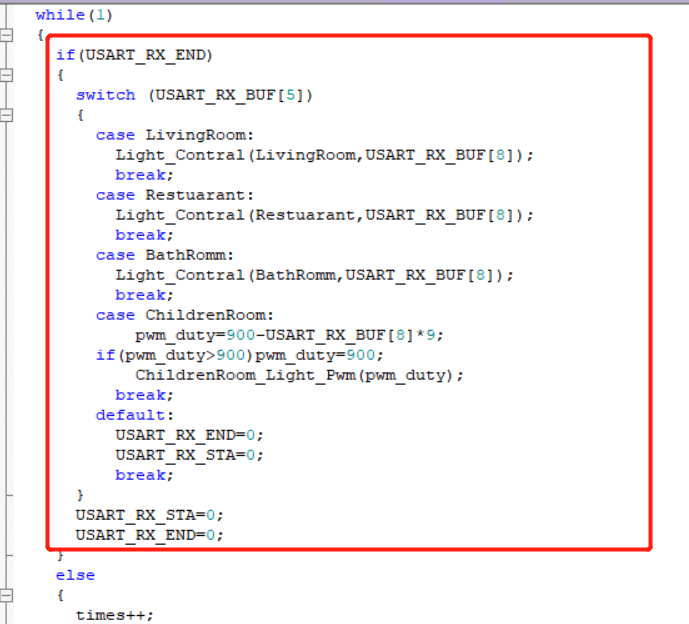


The received data is stored in the "USART\_RX\_BUF" array.

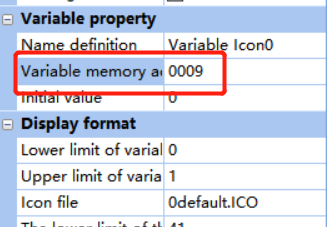
In this project, the receiving length is fixed. When the receiving length is more than 9 bytes, the receiving end is judged.

## Control the switching state of the lamp

In the main function, I wrote some logic code to control the switch state of the lamp:

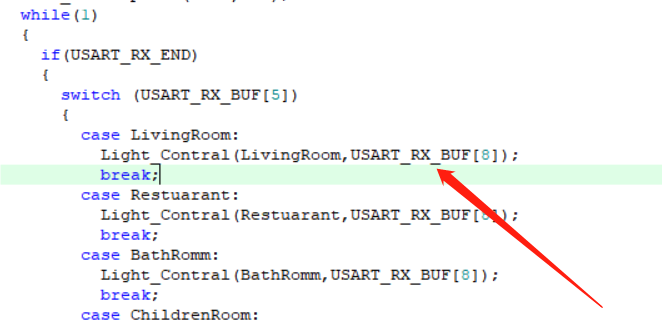


As we can see, the code first determines whether the serial port data is received, and when the serial port data is received, determines which button the user presses on the display screen.Different buttons on the display have different addresses, which can be seen in the VGUS software:



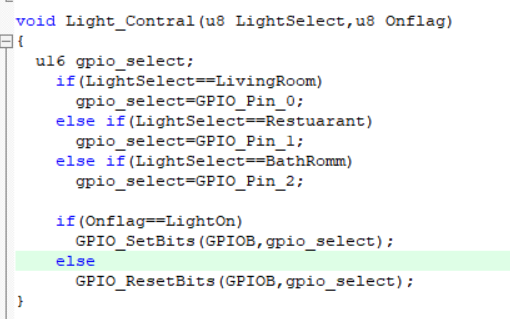
When the user presses the "Living Room" button, the fourth and fifth bits of the data sent out by the serial port of the display screen are the address of the button. Since the fourth bit of all the buttons set here is 0x00, we can judge which button the user presses by directly judging the data of the fifth bit.

After obtaining the button pressed by the user, we need to judge the user data received when the button is pressed, which is the eighth digit of the data sent from the display screen. Therefore, we make the following control:



write the button address parameter and user data into the "Light\_Contral" function to control the on-off state of the light.

Light\_Contral function entity is as follows:



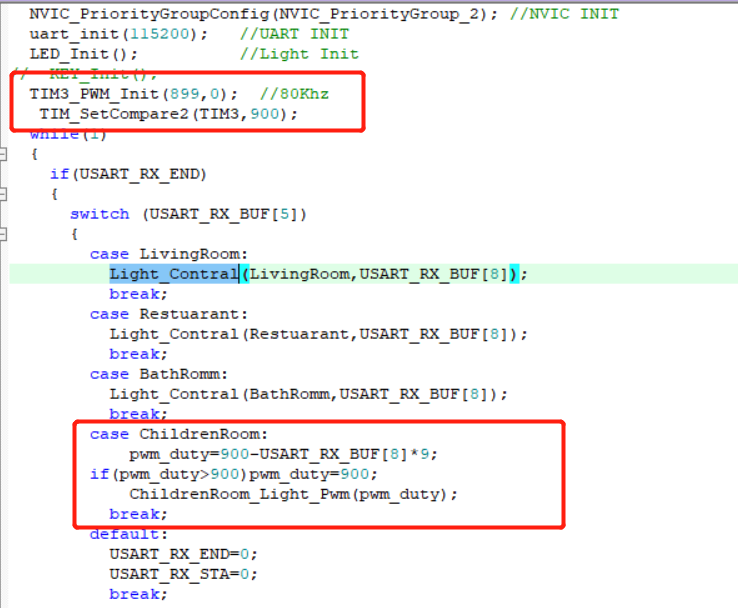
As you can see, if the button address is "Living Room" and the user data is "LightOn", then the PB0 pin of the MCU is set to high level output and the light is on.

The other three buttons are similar, but I won't go on here.

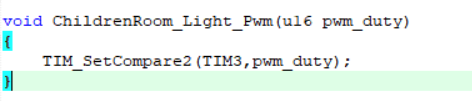
## PWM output

In the UI designed by me, there is a sliding regulator, which is used to control the brightness of the light of "Children Room". MCU is implemented by PWM.PWM output pin is PB5.

The code is as follows:



The slide adjuster is set to a minimum value of 0x00 and a maximum value of 0x64.When sliding, the serial port of the display screen will also send relevant addresses and data, and then set the duty ratio of PWM output by calling the following function:



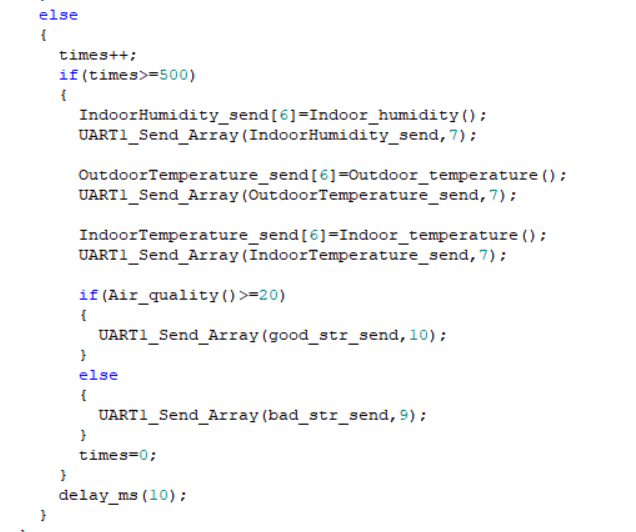
## Sensor acquisition

In the page of "Sensor" of the display screen, there are four Sensor data.

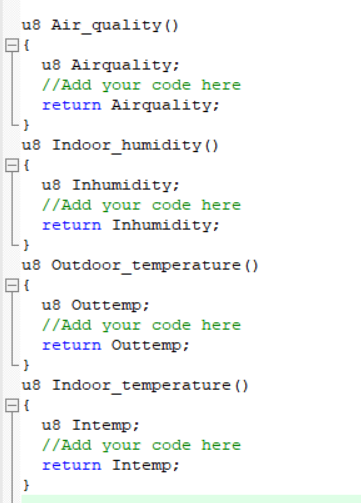


The data also has a storage address in the display, and we can change the real content by simply writing data to these addresses through the serial port of the MCU.

Here I made a simple code implementation:



The display data is updated every 5 seconds, and I only wrote a simple demo of the relevant sensor collection function, because I don't have these sensors in my hand.



In real project development, these sensors may be data collected by ADC, or data collected by IIC, UART, and SPI communication interfaces. All you need to do is write these data into the corresponding function as return value.

# Actual operation effect

