

**计算机科学与工程学院**

**“嵌入式系统”实验报告书**

**题目：ex5\_922106840127\_ I2C**

**学号：922106840127**

**姓名：刘宇翔**

**成绩**

**日期： 2025年 4月 8日**

# 1 题目要求

1. **题目设计要求**

（1）实验内容

通过I2C总线读写AP3216C传感器采集的环境光数据和距离数据，并将它们显示在LCD上，数据保留一位小数。

要求：采用I2C2的HAL库驱动函数(不是GPIO口模拟I2C总 线)进行操作AP3216C；

（2）完成要求

工程名称命名：ex5\_学号\_I2C，并打包成：ex5\_学号\_ I2C.rar压缩文件夹

实验报告PDF格式：ex5\_学号\_ I2C.pdf

1. **拟实现的具体功能**

本次实验拟实现一个基于STM32CubeMX硬件设计平台的I2C传感器数据采集系统，通过I2C2总线读写AP3216C传感器采集环境光与距离数据，并实时将采集结果显示在LCD屏上，数据显示精确到一位小数。系统采用I2C2的HAL库驱动函数进行AP3216C的操作，确保数据采集的准确性和实时性。

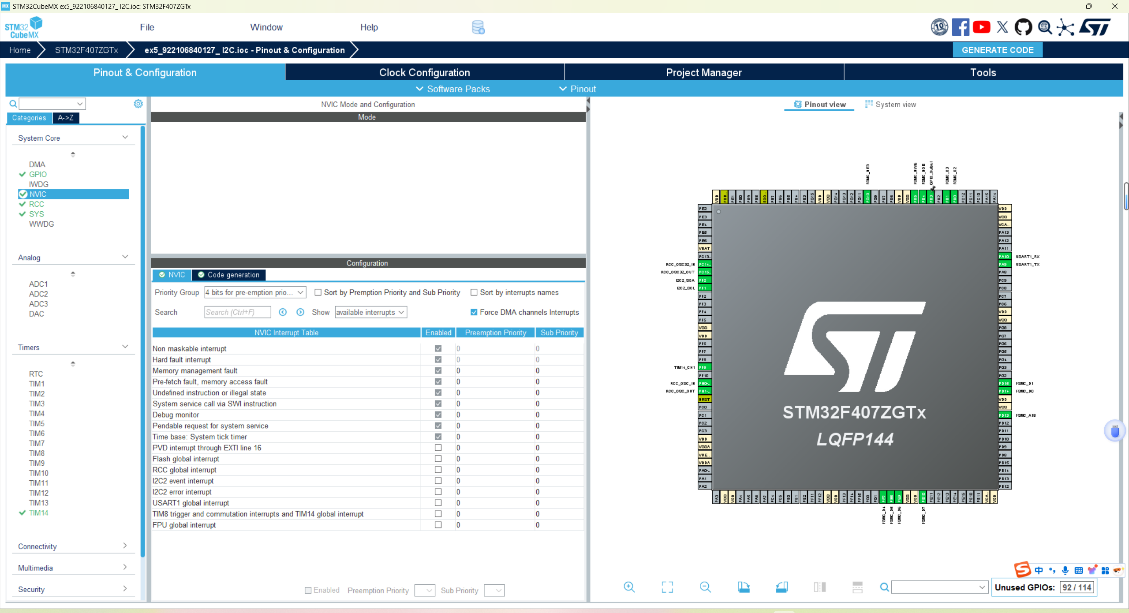
实验中，首先完成各项硬件初始化和外设配置，包括I2C2接口、LCD显示模块及其它必要的资源；接着通过AP3216C的寄存器操作完成传感器初始化，并周期性读取环境光数据及距离数据；最后，将读取到的数据经过格式化处理后显示在LCD上，同时通过串口输出调试信息，为后续数据交互和功能扩展打下基础。

# 2 总体设计

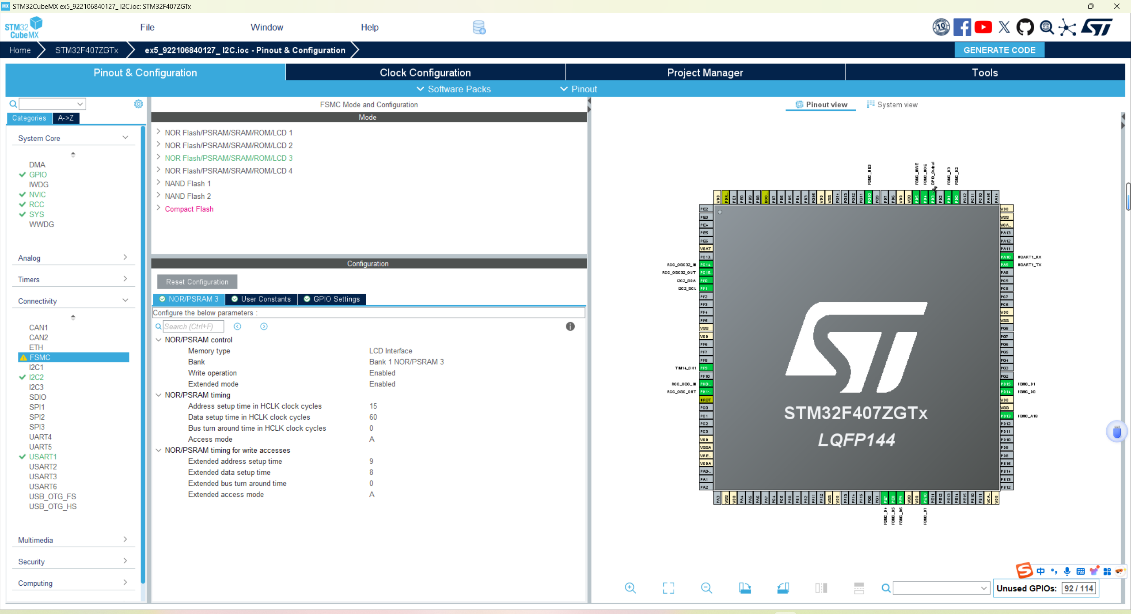
## 2.1硬件设计

**1.硬件设计思路**

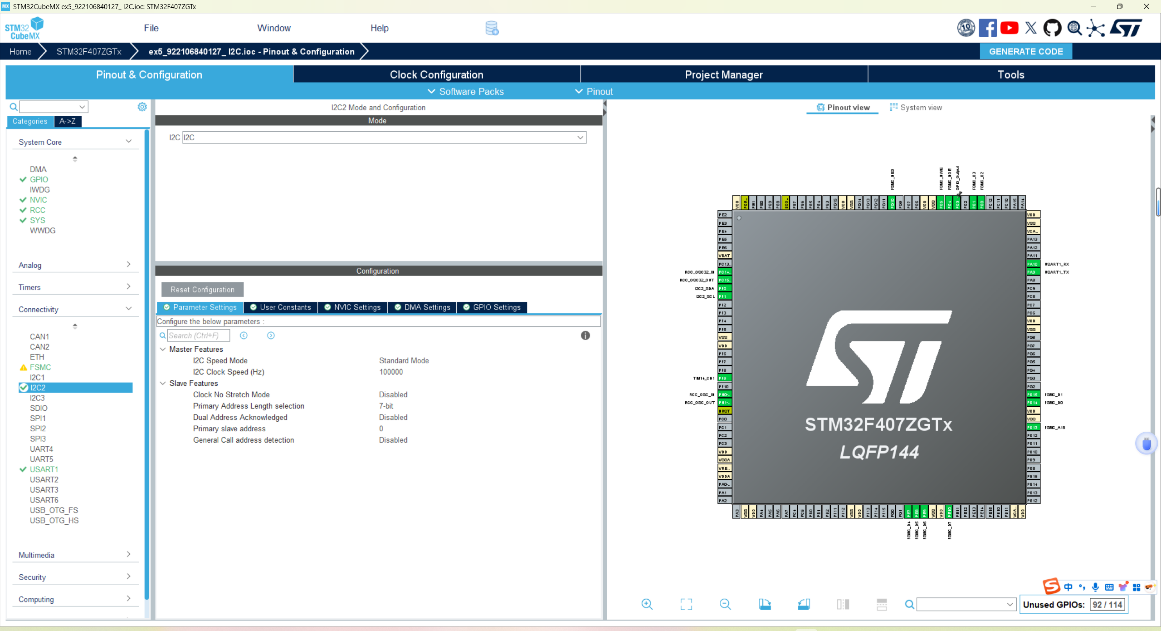
我通过查询相关开发板原理图确定了本次实验所需要的有关 GPIO 端口、 NVIC 设置、USART通信端口与TIM相关的设置，具体的代码工程配置硬件设 计如下：



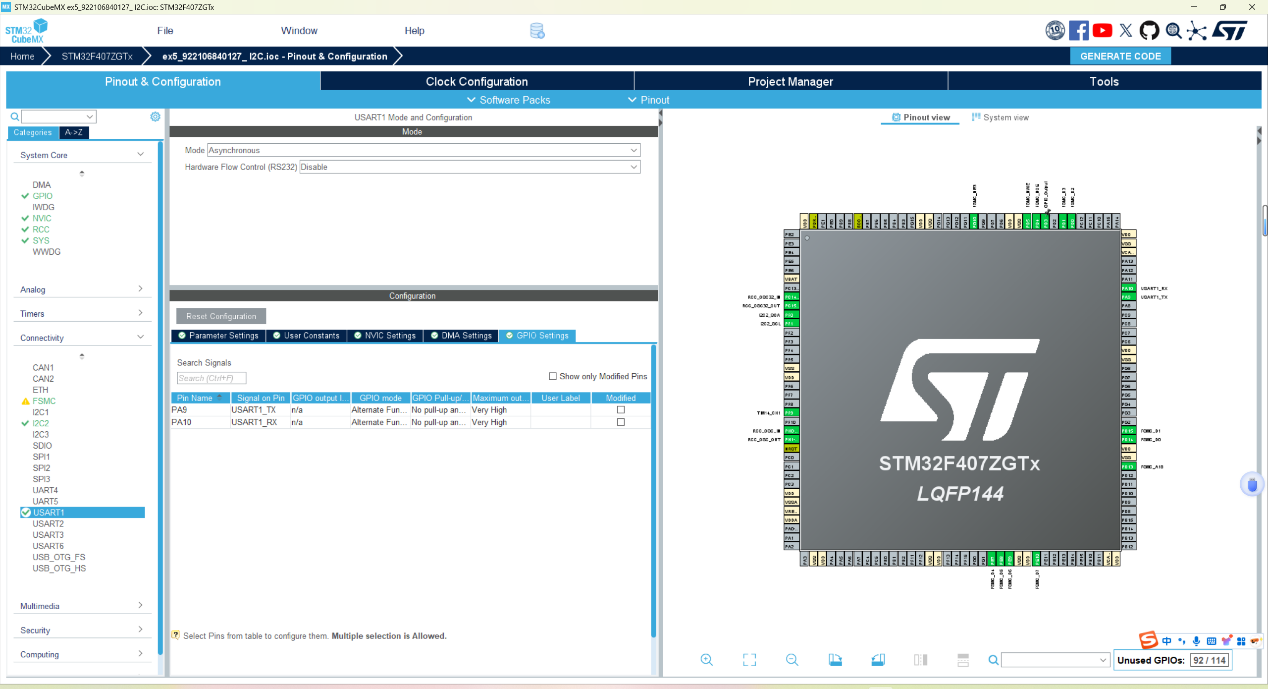
NVIC设置



FSMC设置



I2C设置



USART设置

如图所示，我在NVIC的相关设置界面对几种中断的相关属性做了具体的规 定，USART 的通信模式采用的是异步通信，FSMC 是设置 LCD 相关的内容， I2C的配置是本次实验的重点内容。

其他设置与前几次实验相同，如RCC采用Crystal/Ceramic Resonator，使用 Serial Wire 用于调试对应的debug接口，对Clock Configuration 的时钟配置进行 规范，选择了MDK-ARM V5.32作为编译工具链，其他内容此处省略。 以上配 置图是我作为 STM32CUBEMX 进行的配置设置，设置后点击 “Generate Code” 生成初始化代码。

## 2.2 软件设计

**1.软件设计概述**

本软件主要实现基于I2C总线的AP3216C传感器数据采集与LCD显示功能，通过STM32单片机利用I2C2接口读写AP3216C传感器，采集环境光数据和距离数据，并将经过格式化处理后的结果实时显示在LCD屏幕上，数据精度严格保留一位小数。系统基于HAL库进行外设初始化，采用模块化设计方式，整体结构清晰、易于维护且具备良好的实时性与扩展性。具体设计内容如下：

1. **系统初始化模块**

在main函数中，系统首先调用HAL\_Init()完成底层硬件初始化，并通过SystemClock\_Config()配置系统时钟，确保各外设的工作时序正确；随后依次初始化GPIO、FSMC、I2C2、TIM14以及USART1，并调用drv\_lcd\_init()初始化LCD显示模块，通过lcd\_clear()设置初始显示状态。

1. **AP3216C传感器操作模块**

利用I2C2接口，通过HAL\_I2C\_Mem\_Read/Write函数实现对AP3216C传感器各寄存器的操作，在AP3216C\_Init()函数中完成传感器的复位和使能设置。通过AP3216C\_Read\_PS()与AP3216C\_Read\_ALS()分别读取距离数据和环境光数据，同时根据传感器的量程设置进行单位换算，确保采集数据的正确性。

1. **数据显示与格式化模块**

在主循环中，系统周期性调用Display\_Sensor\_Data()函数，将采集到的距离数据以及经过RoundToOneDecimal()函数处理后的环境光数据进行格式化，生成严格保留一位小数的字符串，再调用lcd\_show\_string()将数据显示在LCD上；同时通过printf()输出调试信息，便于实时监控数据变化。

1. **辅助数据格式化模块**

为保证LCD上显示的环境光数据始终严格保留一位小数，系统新增RoundToOneDecimal()函数，对原始浮点数据进行四舍五入处理，使所有显示数据的精度符合要求，避免因数值变化导致多余小数位或显示异常。

总体而言，本软件依托I2C总线及HAL库驱动AP3216C传感器，实现了环境光与距离数据的实时采集和精确显示，模块划分合理、逻辑清晰，为后续进一步扩展如数据记录、报警控制等功能奠定了坚实基础。

**2.软件流程分解**

A. 初始阶段  
开始 → 系统初始化  
程序启动后，首先完成各项硬件与外设的初始化配置，包括：  
• 系统底层初始化（调用 HAL\_Init() 对 MCU 进行复位与基础设置）  
• 系统时钟配置（通过 SystemClock\_Config() 配置主频与外设时钟，确保各模块正常运行）  
• 初始化 GPIO、FSMC、I2C2、TIM14 以及 USART1（分别调用 MX\_GPIO\_Init()、MX\_FSMC\_Init()、MX\_I2C2\_Init()、MX\_TIM14\_Init() 与 MX\_USART1\_UART\_Init()，为传感器通信、LCD显示及调试信息输出提供硬件支持）  
• 初始化 LCD（调用 drv\_lcd\_init() 配置液晶模块，并通过 lcd\_clear() 清屏设置初始显示状态）  
• 初始化AP3216C传感器（调用 AP3216C\_Init() 进行复位及使能设置，保证传感器进入正常工作状态）

B. 主循环结构  
初始化完成 → 进入主循环  
系统进入主循环后，周期性地读取AP3216C传感器采集的环境光和距离数据，并调用显示函数刷新LCD屏幕，同时通过串口输出调试信息。

C. I2C数据采集流程  
传感器数据读取 → 通过I2C2接口完成数据采集  
利用HAL库提供的I2C2驱动函数，在AP3216C\_Read\_PS()和AP3216C\_Read\_ALS()函数中，依次从AP3216C的各数据寄存器读取两字节数据，再进行数据解码和单位换算，得到实际的距离值与环境光亮度。

D. 数据处理与格式化流程  
读取到的环境光数据经过传感器量程选择后换算出实际亮度值后，再调用辅助函数（如 RoundToOneDecimal()）对数据进行四舍五入处理，确保输出结果严格保留一位小数，从而使数据格式统一、显示规范。

E. LCD显示模块  
周期性刷新LCD → 调用 Display\_Sensor\_Data()  
主循环中每隔一定时间（如2000ms）调用 Display\_Sensor\_Data() 函数，将处理后的距离数据和环境光数据格式化为字符串，使用 lcd\_show\_string() 显示在LCD屏的指定区域，使用户直观查看传感器采集结果。

F. 循环机制  
数据采集及显示完成 → 返回主循环  
每次读取与显示数据完成后，系统返回主循环，等待下一次数据采集与刷新，保证整个系统运行连续、稳定，实现实时监控AP3216C传感器采集的环境光和距离数据，并确保数据输出始终保留一位小数。

**3.μvisoin详细代码**

#include "main.h"

#include "i2c.h"

#include "tim.h"

#include "usart.h"

#include "gpio.h"

#include "fsmc.h"

#include "stdio.h"

#include "./BSP/LCD/drv\_lcd.h"

#include "./BSP/LCD/rttlogo.h"

#define AP3216C\_CONFIG\_REG 0x00

#define AP3216C\_IR\_DATA\_L 0x0A

#define AP3216C\_ALS\_DATA\_L 0x0C

#define AP3216C\_PS\_DATA\_L 0x0E

#define AP3216C\_ALS\_CONFIG\_REG 0x10

#define AP3216C\_ADDR 0x3C

extern I2C\_HandleTypeDef hi2c2;

void SystemClock\_Config(void);

enum als\_range {

AP3216C\_ALS\_RANGE\_20661, //Resolution = 0.35 lux/count(default)

AP3216C\_ALS\_RANGE\_5162, //Resolution = 0.0788 lux/count

AP3216C\_ALS\_RANGE\_1291, //Resolution = 0.0197 lux/count

AP3216C\_ALS\_RANGE\_323 //Resolution = 0.0049 lux/count

};

//fprintf()

int fputc(int ch, FILE \*f) {

HAL\_UART\_Transmit(&huart1, (uint8\_t \*)&ch, 1, 1000);

return ch;

}

uint8\_t AP3216C\_WriteOneByte(uint8\_t reg, uint8\_t data) {

return HAL\_I2C\_Mem\_Write(&hi2c2, AP3216C\_ADDR, reg, I2C\_MEMADD\_SIZE\_8BIT, &data, 1, 100);

}

uint8\_t AP3216C\_ReadOneByte(uint8\_t reg) {

uint8\_t data;

HAL\_I2C\_Mem\_Read(&hi2c2, AP3216C\_ADDR, reg, I2C\_MEMADD\_SIZE\_8BIT, &data, 1, 100);

return data;

}

float RoundToOneDecimal(float value) {

return ((int)(value \* 10 + 0.5)) / 10.0f;

}

void AP3216C\_Init(void) {

AP3216C\_WriteOneByte(AP3216C\_CONFIG\_REG, 0x04); // Reset

HAL\_Delay(50);

AP3216C\_WriteOneByte(AP3216C\_CONFIG\_REG, 0x03); // Enable ALS + PS + IR

if (AP3216C\_ReadOneByte(AP3216C\_CONFIG\_REG) == 0x03) {

printf("AP3216C Init OK!\r\n");

} else {

printf("AP3216C Init Failed!\r\n");

}

}

uint16\_t AP3216C\_Read\_PS(void) {

uint32\_t buf[2];

uint32\_t read\_data;

uint16\_t proximity = 0;

for (int i = 0; i < 2; i++) {

buf[i] = AP3216C\_ReadOneByte(AP3216C\_PS\_DATA\_L + i);

}

read\_data = buf[0] + (buf[1] << 8);

proximity = (read\_data & 0x000F) + (((read\_data >> 8) & 0x3F) << 4);

return proximity;

}

float AP3216C\_Read\_ALS(void) {

uint32\_t buf[2];

uint32\_t read\_data;

float brightness = 0;

for (int i = 0; i < 2; i++) {

buf[i] = AP3216C\_ReadOneByte(AP3216C\_ALS\_DATA\_L + i);

}

read\_data = buf[0] + (buf[1] << 8);

uint8\_t range = (AP3216C\_ReadOneByte(AP3216C\_ALS\_CONFIG\_REG) >> 4) & 0x03;

switch (range) {

case AP3216C\_ALS\_RANGE\_20661: brightness = 0.35 \* read\_data; break;

case AP3216C\_ALS\_RANGE\_5162: brightness = 0.0788 \* read\_data; break;

case AP3216C\_ALS\_RANGE\_1291: brightness = 0.0197 \* read\_data; break;

case AP3216C\_ALS\_RANGE\_323: brightness = 0.0049 \* read\_data; break;

default: break;

}

return brightness;

}

void LCD\_Init\_Display(void) {

drv\_lcd\_init();

lcd\_clear(WHITE);

lcd\_set\_color(WHITE, BLACK);

}

void Display\_Sensor\_Data(uint16\_t ps\_data, float als\_data) {

char buffer[32];

float als\_formatted = RoundToOneDecimal(als\_data);

sprintf(buffer, "PS: %.1f", (float)ps\_data);

lcd\_show\_string(30, 70, 32, buffer);

sprintf(buffer, "ALS: %.1f", als\_formatted);

lcd\_show\_string(30, 110, 32, buffer);

printf("PS: %d\r\n", ps\_data);

printf("ALS: %.1f\r\n", als\_formatted);

}

int main(void) {

HAL\_Init();

SystemClock\_Config();

MX\_GPIO\_Init();

MX\_FSMC\_Init();

MX\_I2C2\_Init();

MX\_TIM14\_Init();

MX\_USART1\_UART\_Init();

HAL\_TIM\_Base\_Start\_IT(&htim14);

HAL\_TIM\_PWM\_Start(&htim14, TIM\_CHANNEL\_1);

LCD\_Init\_Display();

AP3216C\_Init();

while (1) {

uint16\_t ps = AP3216C\_Read\_PS();

float als = AP3216C\_Read\_ALS();

Display\_Sensor\_Data(ps, als);

HAL\_Delay(2000);

}

}

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 4;

RCC\_OscInitStruct.PLL.PLLN = 168;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 4;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5) != HAL\_OK)

{

Error\_Handler();

}

}

void Error\_Handler(void)

{

\_\_disable\_irq();

while (1){ }

}

#ifdef USE\_FULL\_ASSERT

void assert\_failed(uint8\_t \*file, uint32\_t line){}

#endif /\* USE\_FULL\_ASSERT \*/

# 3 实验结果分析与总结

代码工程编译完成上板运行，可以看到在不加外部强光下ALS值很小，仅为2.5，后续加了强光，ALS值飙升值242甚至600，传感器显示无误，完成实验要求。详细内容请见附带的视频：

