

《物联网工程及应用》实验指导书

(试用版)

**五邑大学 智能制造学部
重庆大云端科技有限公司**

联合编著

2021.06

前 言

《物联网工程及应用》是五邑大学通信工程专业(物联网工程方向)的核心专业课,课内实验是帮助同学们从理论学习走向物联网项目开发的主要环节。本实验指导书是基于重庆大云端科技有限公司提供的物联网实训设备编写而成的,适用于我校物联网工程专业的学生在《物联网工程及应用》实验教学中使用。

本实验指导书基于物联网层级参考体系结构,共设计了4个实验。

实验1 认知物联网实训系统。主要是为了让同学们熟悉实验设备,对照物联网参考体系结构,熟悉物联网实训系统的硬件结构和硬件模块,掌握其工作原理和使用方法。

实验2 数据感知与无线传输实验。通过基础训练包的焊接、安装、调试,让同学们掌握温湿度感知节点的设计方法,包括硬件设计和软件编程方法;熟悉终端节点的无线接入方式和无线数据传输,掌握基于WiFi模块AT指令的编程方法,以及通过串口调试助手调试WiFi模块的基本方法。

实验3 网络传输与数据存储实验。主要是为了让同学们了解数据存储方法,掌握数据库MySQL的设计方法,及其增删改查的编程方法;编写服务端程序,接收实时采集数据并进行存储。

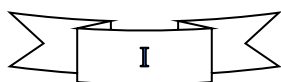
实验4 温湿度实时监测系统。在实验2~3的基础上,实现温湿度数据的实时采集、传输与储存;编写应用程序,实现温湿度实时采集数据的显示和历史数据查询。这部分内容是在前几个实验的基础上设计的综合实验,主要完成物联网项目——温湿度实时监测系统的整体搭建及其开发,培养学生进行物联网简单应用系统开发的基本能力。

为配合本实验指导书的使用,提供了实验设计的相关电子文档,包括本实验指导书、软件、工程模板、参考资料、参考代码等,全部电子文档都可以在工作站电脑中找到,供同学们在实验过程中参考。

由于时间仓促,在编写过程中难免会有错误纰漏,请读者批评指正。

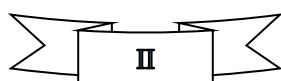
编 者

2021年6月



目 录

前 言.....	I
实验 1 认知物联网实训系统	1
实验 2 数据感知与无线传输实验	11
实验 3 网络传输与数据存储实验	29
实验 4 温湿度实时监测系统	45



实验1 认知物联网实训系统

【实验目的】

1. 熟悉物联网实训系统组成、工作原理和基本功能，掌握相关仪器设备的使用方法 & 注意事项。

2. 通过搭建环境监测系统，掌握物联网应用系统的层级架构，准确理解各物理对象与系统架构的对应关系，熟悉信息感知、数据传输、数据存储和数据应用各环节的工作原理。

【实验设备】

1. 物联网实训工作站	1 台
2. 智能物联终端	1 套
3. 环境感知物联训练装置	1 套
4. 应用交互技术训练装置	1 套
5. 物联网服务与应用技术训练平台（服务器）	1 套
6. 网络设计训练装置（路由器/交换机）	1 套

【实验要求】

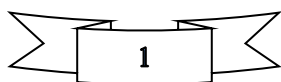
1. 用杜邦线、航空插头等连接线，连接环境监测系统所需硬件模块，构建环境监测硬件系统。

2. 通过 LCD 显示屏、iPad 和扬声器等观察系统运行结果。

【实验原理】

1. 物联网实训系统组成

物联网实训系统由实训工作站、智能物联终端、环境感知物联训练装置、应用交互技术训练装置、WiFi AP热点、路由器、交换机、服务器等组成。实训工作台面布局如图1.1所示，其它实验设备与装置见图1.2。



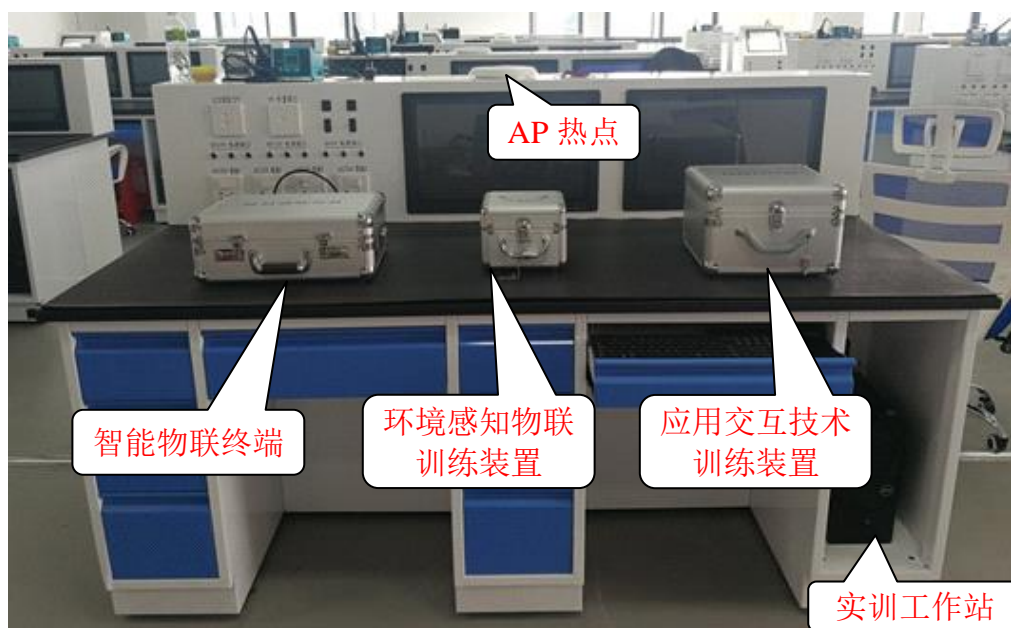


图1.1 实训工作站台面布局



图1.2 实验装置

2. 环境监测物联网应用系统的系统架构及工作原理

➤ 环境监测物联网应用系统架构及硬件总体框图如图1.3所示。各环境参数传感器通过不同的接口接入MCU，完成环境参数的信息采集；所采集的数据经单片机处理后，可以直接在LCD显示屏上实时显示出来。

➤ 采集数据上传至服务器，或服务器下传指令给终端节点，有2种接入方式：

(1) 通过以太网接口（网卡）接入；

(2) 通过 WiFi AP 热点接入。

➤ 用户终端iPad可以通过AP热点从服务器获取感知节点所采集的数据，实现环境参数的实时监测和历史数据查询。

➤ 当所监测环境参数超过设定阈值时，扬声器会发出报警声。

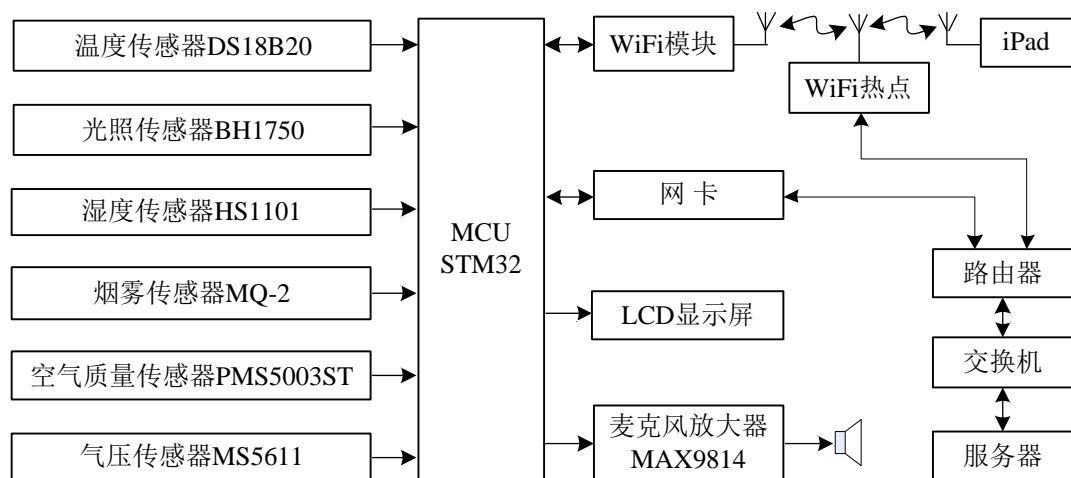


图1.3 环境监测系统架构及硬件总体框图

【实验步骤】

1. 按表 1.1~1.9，对照图 1.4 完成硬件连接。

表 1.1 PMS5003 空气质量模块连接表

PMS5003 空气质量模块	物联终端连接管脚
① — VCC	悬空
② — GND	① —GND （UART 接口）
③ — SET	悬空
④ — TX	③ —TX（UART 接口）
⑤ — RX	④ —RX（UART 接口）
⑥ — RESET	悬空

表 1.2 DS18B20 温度模块连接表

DS18B20 温度模块	物联终端连接管脚
① — GND	①—GND (GPIO 接口)
② — OUT	⑮—PE12 (GPIO 接口)

表 1.3 BH1750 光照模块连接表

BH1750 光照模块	物联终端连接管脚
① — ADDR	悬空
② — SDA	⑫—PG0 (GPIO 接口)
③ — SCL	⑪—PF14 (GPIO 接口)
④ — GND	③ —GND (GPIO 接口)

表1.4 HS1101湿度模块连接表

HS1101 湿度模块	物联终端连接管脚
① — OUT	悬空
② — OUT	⑤ —AD_4 (AD 接口)
③ — GND	④ —GND (AD 接口)
④ — GND	悬空

表 1.5 MQ-2 烟雾模块连接表

MQ-2 烟雾模块	物联终端连接管脚
① — OUT	悬空
② — AD	① —AD_1 (AD 接口)
③ — GND	④ —GND (AD 接口, HS1101⑤)
④ — GND	悬空

表 1.6 MS5611 气压模块连接表

MS5611 气压模块	物联终端连接管脚
① — GND	②—GND (GPIO 接口)
② — SCL	⑮—PE10 (GPIO 接口)
③ — SDA	⑯— PE7 (GPIO 接口)
④ — CSB	悬空
⑤ — SDO	悬空
⑥ — PS	悬空

表 1.7 MAX9814 声音采集模块连接表

MAX9814 声音采集模块	物联终端连接管脚
① — AR	悬空
② — OUT	⑥ — AD_2 (AD 接口)
③ — GAIN	悬空
④ — GND	④ — GND (AD 接口)

表 1.8 AD9834 可调波形模块连接表

AD9834 可调波形模块	物联终端连接管脚
① — RST	⑥ — MOSI (SPI 接口)
② — SDA	① — MISO (SPI 接口)
③ — CLK	⑤ — SCK (SPI 接口)
④ — FSY	② — NSS (SPI 接口)
⑤ — GND	③ — GND (SPI 接口)

表 1.9 MPU6050 加/角速度模块连接表

MPU6050 加/角速度模块	物联终端连接管脚
① — INT	悬空
② — SCL	⑬ — PF15 (GPIO 接口)
③ — SDA	⑭ — PG1 (GPIO 接口)
④ — RX	悬空
⑤ — TX	悬空
⑥ — ADO	悬空
⑦ — GND	⑤ — GND (GPIO 接口)

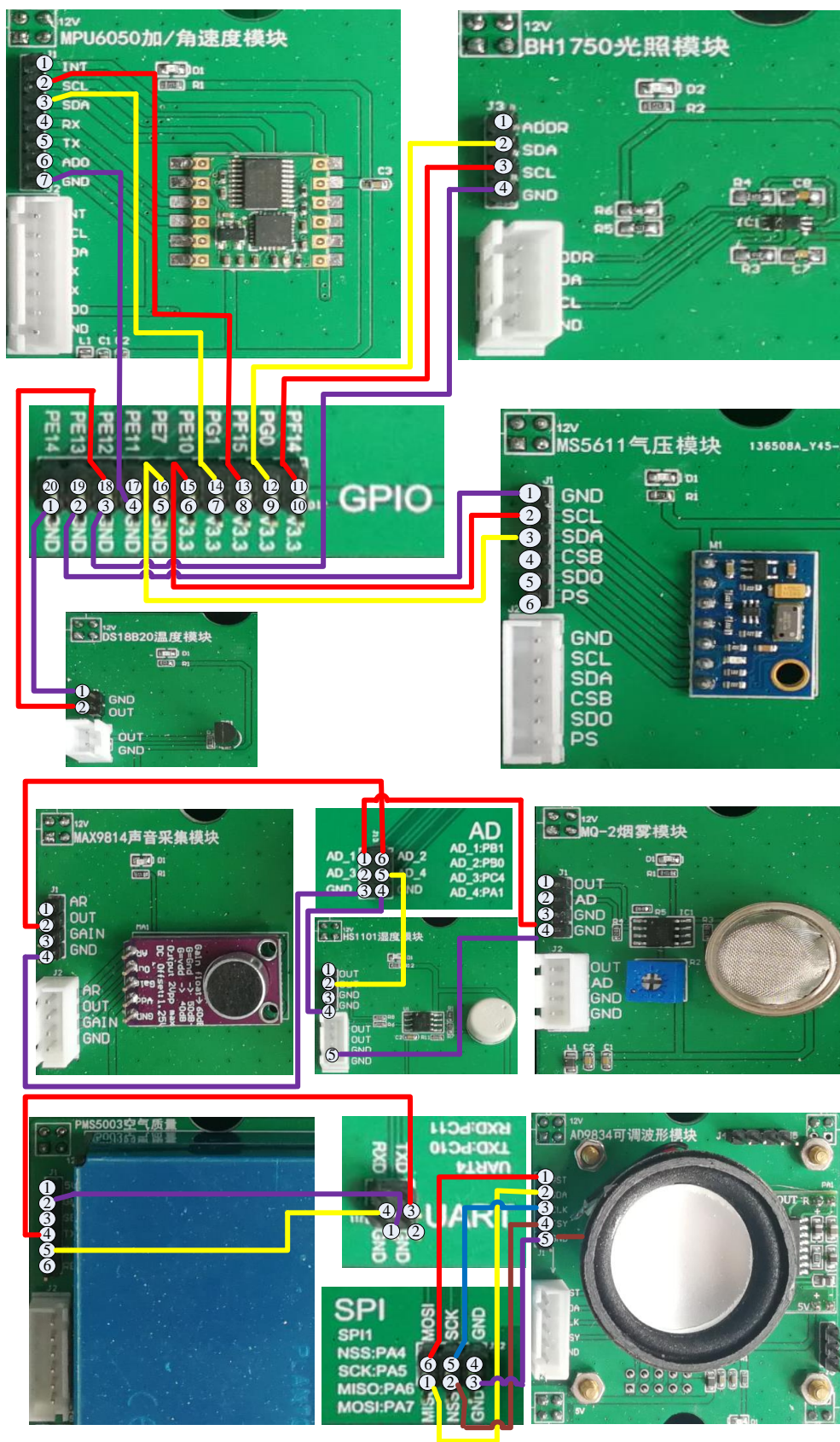


图1.4 实物接线图

2. 按照图 1.5 所示实训工作台和实验箱电源开关及插座，正确连接实验箱电源至工作站台面电源接口，并打开所用设备的电源开关。



图1.5 实验箱电源插座接口及开关

3. 安装 Keil5（已安装的可以跳过）

按照《Keil5安装与使用教程》中的方法安装Keil。

4. 安装 J-Link 驱动（已安装的可以跳过）

按照《Keil5安装与使用教程》中的方法安装J-Link驱动。

5. 在“...\物联网实验\工程模板”路径下，解压 E1_IoTSystem.zip，找到 PMT5003 工程文件，以 Keil5 打开编译后，下载到“智能物联终端”的单片机。注意，物联终端的单片机型号是。

6. 按照表 1.10 和图 1.6 连接 LCD 显示屏，观察实验现象，记录实验结果(照相)。

表 1.10 LCD 显示屏连接表

LCD 显示屏	物联终端连接管脚
航空插头(红黑线)	SDA（I2C 接口）
航空插头（红线）	SCL（I2C 接口）
航空插头（黑线）	GND（I2C 接口）

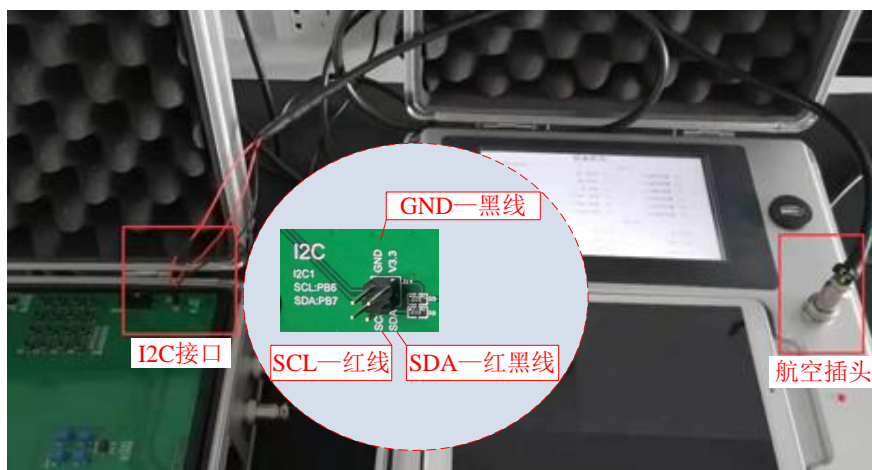


图1.6 LCD显示屏连接图

7. 开启 iPad，打开浏览器，输入网址：<http://192.168.2.221:81>，或在如图 1.7 所示 iPad 首页点击环境感知图标，进入环境感知系统登录界面，如图 1.8 所示。



图1.7 iPad首页界面



图1.8 环境感知系统登录界面

8. 在智能物联终端铭牌上找到“出厂编号”，如图 1.9 所示，该编号也是终端节点的 ID 号。然后在登录界面输入用户名：test，密码：123，进入环境感知系统设备监控界面，如图 1.10 所示。

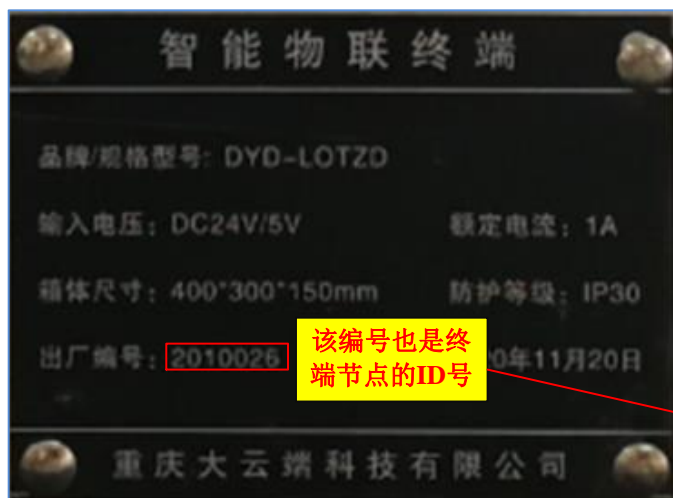


图1.9 物联终端ID号

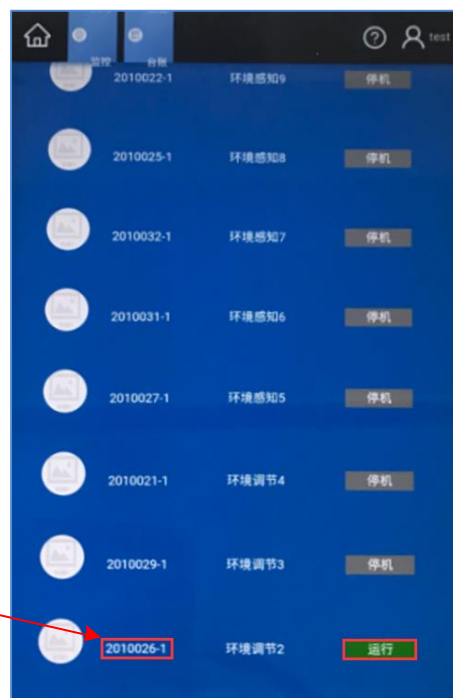


图1.10 环境感知系统实时监测界面

9. 如果你所使用 ID 号的终端节点处于“运行”状态，可以双击 ID 号，进入该 ID 号对应终端节点的实时监测界面，观察并记录实验结果（照相）。
10. 打开设备台账界面，改变环境参数的报警阈值，观察并记录实验现象。

【思考与练习】

1. 实验中各传感器模块与 MCU 的接口分别是什么？哪些是物理接口、哪些是虚拟接口？
2. 实验中的传感器哪些是数字传感器？哪些是模拟传感器？
3. 说明实验所构建的环境监测系统中，各物理对象与物联网层级系统结构的对应关系，以此说明感知/控制层、网络层和应用层的功能与作用。

【预习要求】

1. 认真阅读《Keil5 安装与使用教程》，在自己的电脑上安装 Keil5，并练习用 Keil5 建立工程，编辑程序代码，进行程序调试的方法。
2. 认真阅读《Keil5 安装与使用教程》，熟悉 J-Link 驱动的安装方法，了解在 Keil5 环境下使用 J-Link 下载程序的方法。
3. 认真阅读本实验指导书，熟悉实训系统的架构和实验步骤。

实验2 数据感知与无线传输实验

【实验目的】

1. 掌握物联网感知节点的组成、工作原理和基本功能，熟悉感知节点的设计方法。
2. 熟悉单片机集成开发环境 IDE（Integrated Development Environment），掌握在 Keil5 环境下以工程模板创建工程的方法。
3. 熟悉温湿度传感器 DHT11 的工作原理，初步掌握 DHT11 初始化程序的编写方法。熟悉 WiFi 模块 ESP8266 与 STM 单片机的通信接口，掌握其在程序设计中的管脚定义方法。熟练掌握 STM32 单片机程序的编译、下载与调试方法。
4. 掌握 ESP8266 的 Station 模式应用于物联网终端节点接入网的方法，掌握基于 ESP8266 及其 AT 指令实现数据无线传输的编程方法。
5. 掌握以串口调试工具 SSCOM 进行 WiFi 模块调试的方法。

【实验设备】

- | | |
|---------------|-----|
| 1. 物联网实训工作站 | 1 台 |
| 2. 基础训练包 | 1 套 |
| 3. J-Link 下载器 | 1 个 |

【实验要求】

1. 对照所给原理图和 PCB，找到温度传感器 DS18B20、温湿度传感器 DHT11 和烟雾传感器 MQ-2 与单片机的硬件接口，从而熟悉以 STM32F103C8 单片机为核心的环境参数（温度、湿度、烟雾）感知节点的硬件设计方法。在此基础上，以基础训练包提供的元器件，完成环境参数感知节点的焊接与安装。因为 ds18b20 的几个管脚靠得比较接近，焊接时可以先焊中间的管脚，再焊两边的，这样不容易连锡。

2. 根据所给原理图和 PCB，找到 WiFi 模块 ESP-12S，熟悉其与单片机的硬件接口。对照管脚图和管脚定义，通过工程模板提供的参考代码，掌握在程序设计中 ESP-12S 的管脚定义方法，以及与单片机连接的 STM32 管脚定义。

3. 套用工程模板新建工程，了解工程文件各组文件夹的内容和作用。通过所提供相关器件手册和资料，熟悉 ESP8266 的工作原理。阅读 dht11.c、dht11.h、esp8266.c 和 esp8266.h 源代码，补充编写 DHT11 温湿度传感器初始化程序 DHT11_GPIO_Config() 和 WiFi 模块部分代码。

4. 补充编写 main 程序入口函数，每隔 T 秒（可以自己在程序中设定）采集一次温湿度数据，通过调试窗口（watch）观察所采集的温湿度值；以 TCP 透传到服务端，通过串口调试助手 SSCOM 观察所采集的温湿度值，完成 STM32 系统下的 WiFi 模块调试。

【实验原理】

1. 感知节点组成及工作原理

感知节点组成框图如图2.1所示，主要由MCU、传感器/执行器、射频模块和电源模块组成。

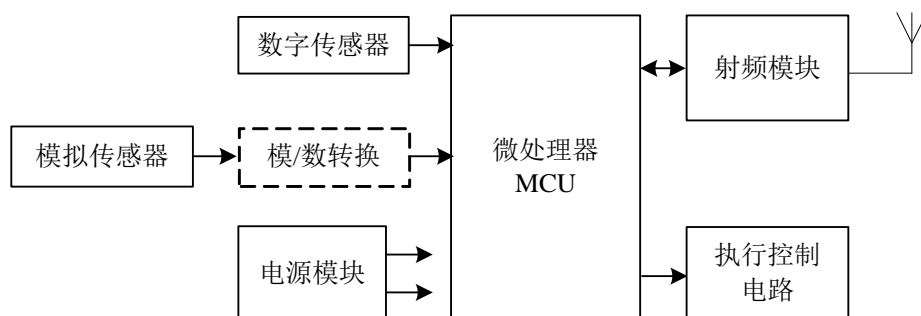


图2.1 典型物联网节点组成框图

（1）微处理器（MCU）：是典型物联网感知/控制节点的核心器件，负责整个节点的控制、任务分配与调度、数据处理等。

（2）传感器：包括各种数字传感器和模拟传感器，负责外部世界的信息感知。其中，数字传感器可以直接与MCU相关接口进行连接，模拟传感器需经模/数转换后与MCU连接。如果MCU本身具有A/D转换接口，模拟传感器也可接入该A/D转换接口。

（3）射频模块：所有感知数据通过射频模块无线接入基础网。

（4）执行控制电路：对终端设备进行控制。

（5）电源模块：为节点提供能量。

2. 环境参数感知节点原理图和 PCB 图

环境参数感知节点原理图、PCB和实物图分别如图2.2、2.3、2.4所示。

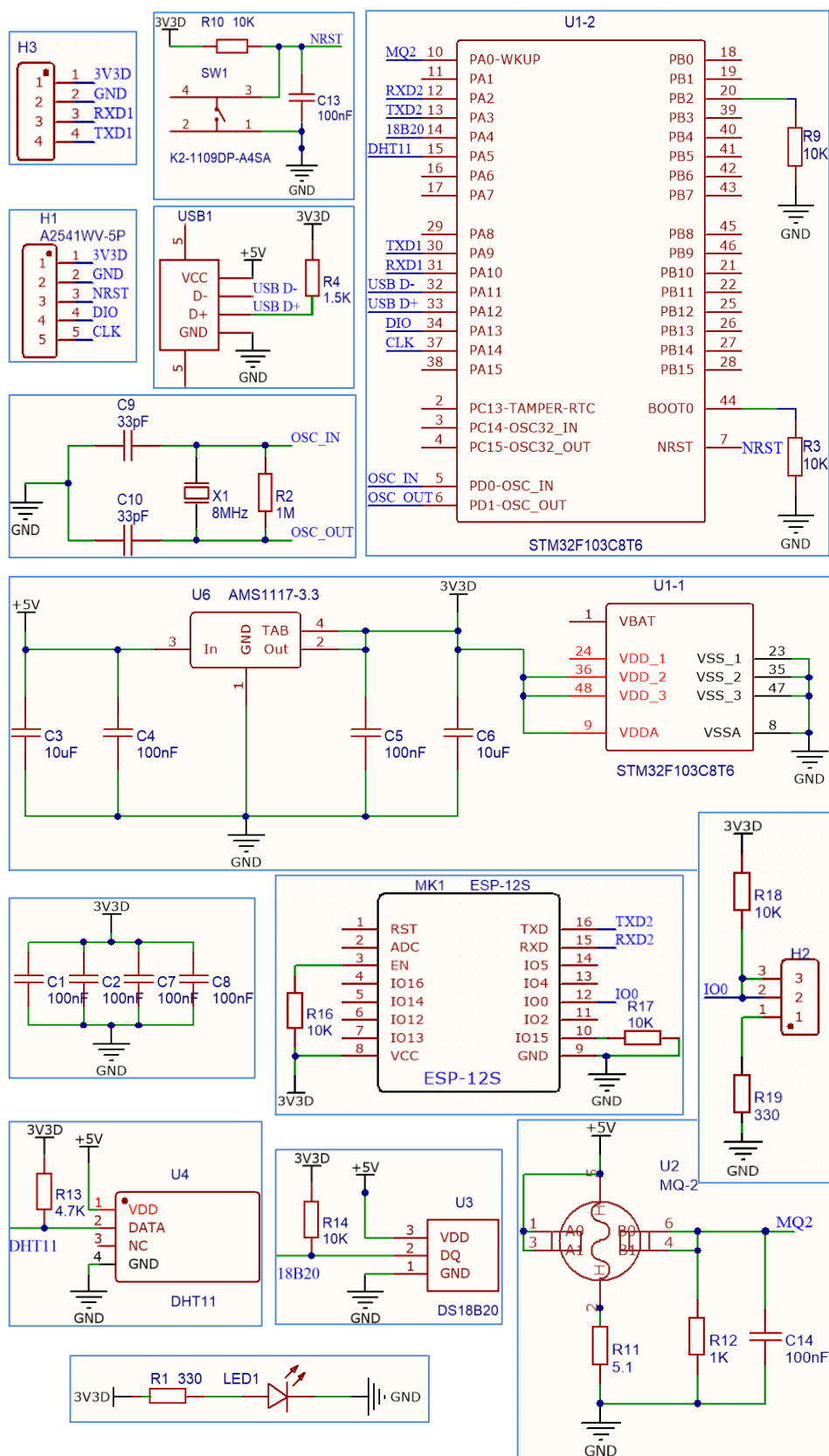


图2.2 环境参数感知节点原理图

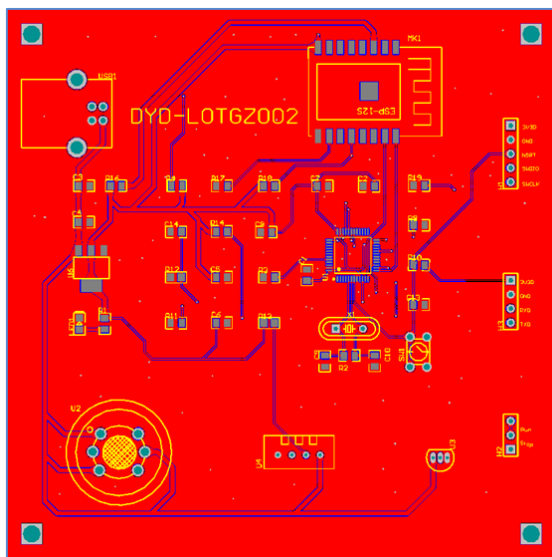


图2.3 环境参数感知节点PCB

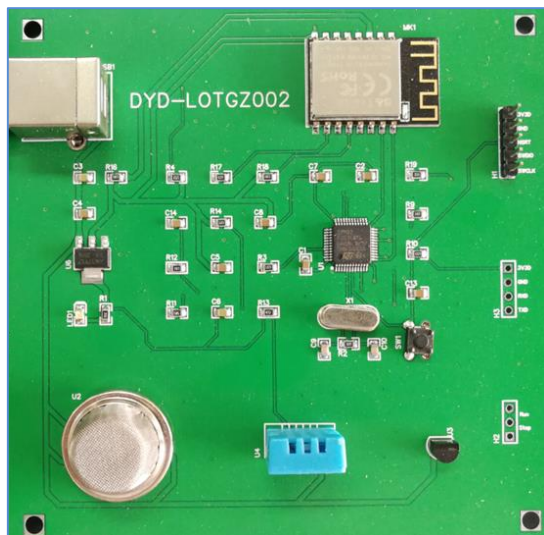


图2.4 环境感知节点实物图

3. 温湿度传感器 DHT11 工作原理及程序设计（参见《DHT11 数据手册》）

根据图2.2所示环境参数感知节点原理图可知，DHT11的数据线DATA与单片机的PA5连接。连接线长度短于20米时用5K上拉电阻，大于20米时根据实际情况使用合适的上拉电阻。

温湿度传感器程序设计包括头文件dht11.h和源文件dht11.c。

4. WiFi 模块工作原理（参见《ESP8266 规格书》和《ESP-12S 规格书》）

（1）WiFi模块ESP-12S

从图2.2可知，开发板WiFi模块采用的是ESP-12S。ESP-1/ESP-S/ESP-F等系列模块，均采用ESP8266EX主控芯片，各种模块的ESP8266源码程序可以原封不动进行移植。所以，实验工程模板提供的是ESP8266相关的程序代码。但各种模块引出的管脚有所不同，硬件设计和软件编程需要做相应修改。

ESP-12S的管脚图如图2.5所示，管脚定义如表2.1所示。

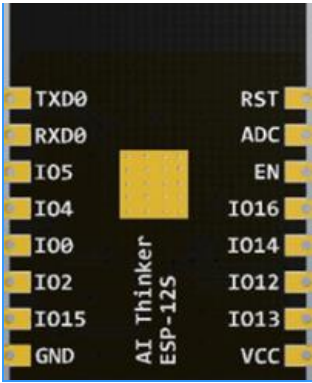


图2.5 ESP-12S管脚图

表2.1 ESP-12S管脚定义

序号	管脚名称	功 能 说 明
1	RST	复位引脚，低电平有效。
2	ADC	A/D转换结果，输入电压范围0~1V，取值范围：0~1024
3	EN	芯片使能端，高电平有效。
4	IO16	GPIO16，与RST管脚相连时可做深度睡眠（deep sleep）唤醒。
5	IO14	GPIO14；HSPI_CLK，IR_T；I2C_SCL；I2SI_WS
6	IO12	GPIO12；HSPI_MISO
7	IO13	GPIO13；HSPI_MOSI；UART0_CTS
8	VCC	3.3V VDD；外部供电电源，输出电流建议在 500mA 以上。
9	GND	接地
10	IO15	GPIO15；I2SO_BCK，HSPI_CS；UART0_RTS
11	IO2	GPIO2；UART1_TXD；I2C_SDA；I2SO_WS
12	IO0	GPIO0；下载模式：外部拉低；运行模式：悬空或外部拉高。
13	IO4	GPIO4
14	IO5	GPIO5；IR_R
15	RXD0	UART0_RXD；GPIO3；I2SO_DATA
16	TXD0	UART0_TXD；GPIO1

(2) WiFi模块ESP-12S与STM32单片机的串行通信

ESP-12S与STM32的串行通信接口为USART2，从图2.2可知，ESP-12S的RXD0和TXD0分别与STM32的PA2(GPIOA的第2个管脚TXD2)和PA3(GPIOA的第3个管脚RXD2)管脚连接。

(3) ESP8266的Station（客户端）模式

ESP8266有三种工作模式：Station——客户端模式；AP——接入点模式；Station+AP——客户端+接入点模式。本实验ESP8266工作于Station模式，终端节点（ESP8266）为客户端（Client），电脑为服务器端（Server），如图2.6所示。



图2.6 ESP8266工作于Station模式的典型应用

5. ESP8266 程序设计（参见《ESP8266 AT 指令集》和《ESP8266 AT 指令使用示例》）

通过串口向ESP8266发送AT指令，即可实现对ESP8266的各种操作。本实验工程模板，提供了向ESP8266发送AT指令的函数ESP8266_Cmd，该函数源代码在工程模板的esp8266.c中，具体格式如下：

```
bool ESP8266_Cmd ( char * cmd, char * reply1, char * reply2, u32 waittime )
```

输入：

- cmd—待发送的指令；
- reply1, reply2—期待的响应，为NULL表不需响应，两者为或逻辑关系；
- waittime—等待响应的时间。

返回：

- 1—指令发送成功；
- 0—指令发送失败。

范例：

```
ESP8266_Cmd ( "AT+CWMODE=1", "OK", "no change", 2500 ); //设置STA模式
```

【实验步骤】

1. 根据所给原理图和 PCB，按照《焊接基础知识》所介绍的方法，完成感知节点硬件系统的焊接、安装，检查无误后，才可以上电（USB 供电）。

2. 感知节点应用程序编写

（1）套用工程模板新建工程

➤ 为了快速、便捷建立STM32工程，可以直接套用工程模板。将实验所提供工程模板压缩文件E2_ProjectTemplate.zip解压，然后将其所有文件拷贝到你自建的工程文件夹中，比如E2_DataSensing，然后删除DebugConfig、Listings、Objects三个文件夹，这三个文件夹在编译时由系统自动生成，所以删不删问题不大。将工程名改为新建工程名E1_DataSensing，如图2.7所示。工程名以及下面的执行文件名改不改也无所谓，只是一个命名而已，当然，取一个合适的工程名或文件名可以使用户一目了然地了解工程的含义，比如工程名E2_DataSensing的含义是实验2_数据感知（Experiment—实验，Data Sensing—数据感知）。

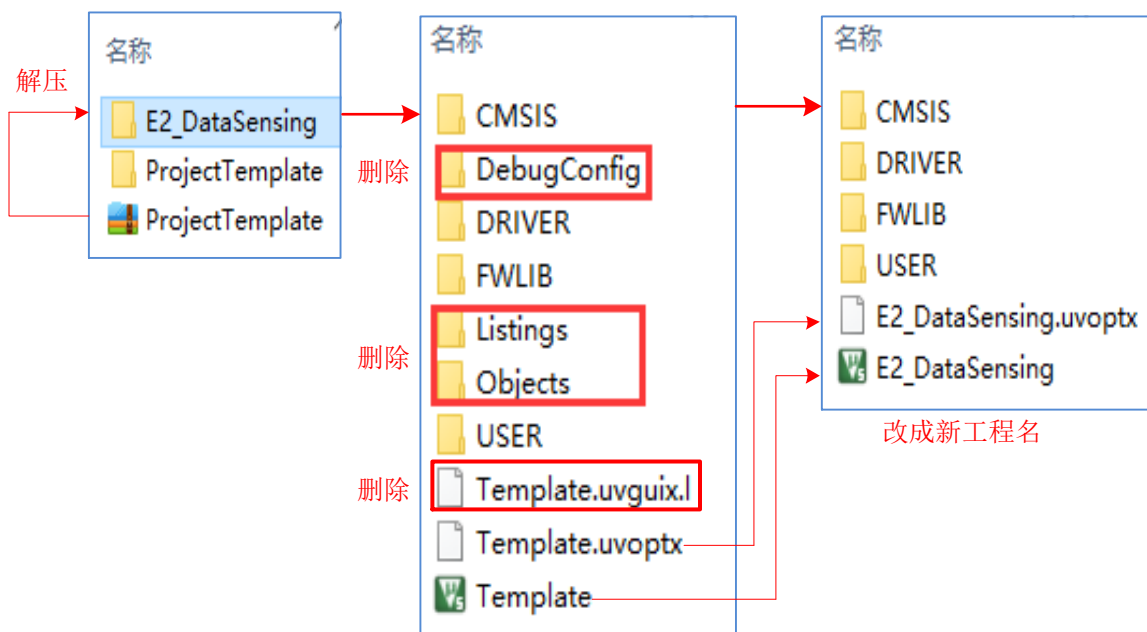


图2.7 套用工程模板创建新工程

剩下的4个组文件夹用来存放各种不同的文件，见表2.1。

表 2.1 工程内组文件夹存放内容

名 称	作 用
CMSIS	微控制器软件接口标准库
FWLIB	固件库
DRIVER	驱动库文件
USER	用户编写的文件 main.c、stm32f1xx_it.c：跟中断有关的函数等

FWLIB库：固件库，FW的单词是Firmware，即固件的意思。STM32的固件库包含C文件和H文件，主要用于存放STM32的一些寄存器的定义及一些底层驱动函数

CMSIS库：微控制器软件接口标准(CMSIS: Cortex Microcontroller Software Interface Standard)，CMSIS 可实现与处理器和外设之间的一致且简单的软件接口，从而简化软件的重用，缩短微控制器开发人员新手的學習过程，并缩短新设备的上市时间。

➤ 打开新建的工程文件，修改执行文件名，如图2.8所示。

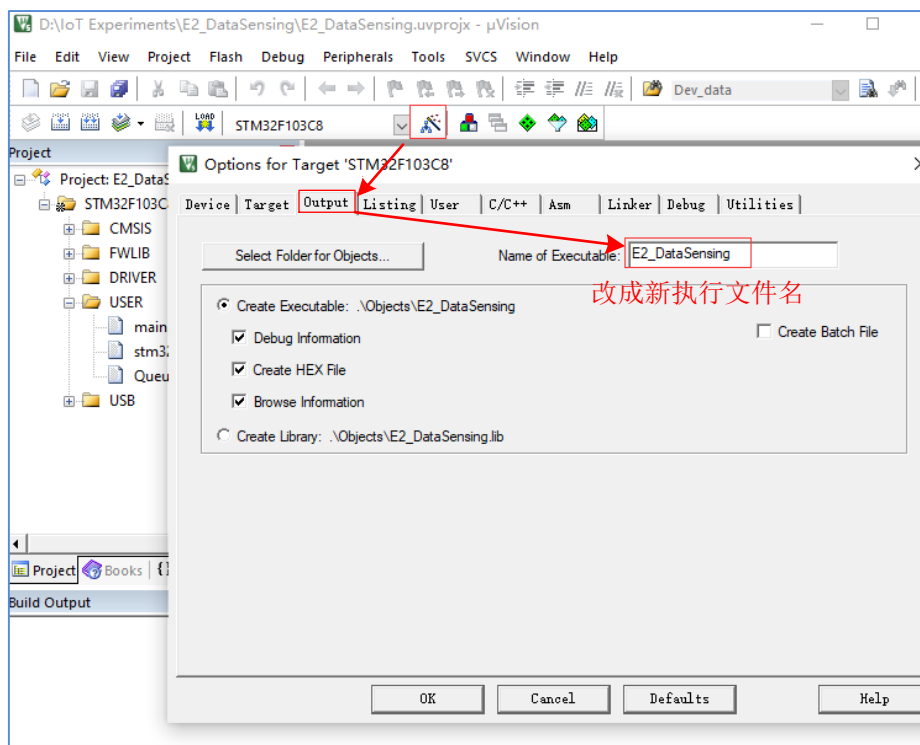


图2.8 修改执行文件名

➤ 编译通过，则成功建立工程，如图2.9所示。但main函数为空，需要用户自己编写应用程序。

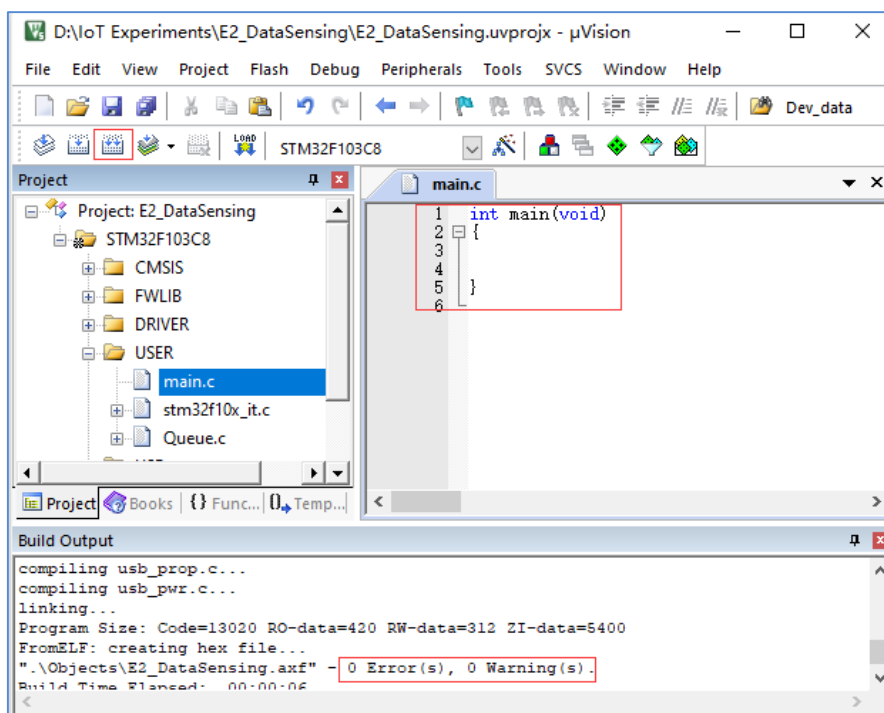


图2.9 成功建立工程

(2) 编写应用程序

➤ 补充编写DHT11初始化函数DHT11_GPIO_Config()

DHT11初始化函数DHT11_GPIO_Config()在DRIVER->dht11.c中, 需要根据工程模板已有代码和《DHT11数据手册》在红线处自行补充完整, 如图2.10所示。

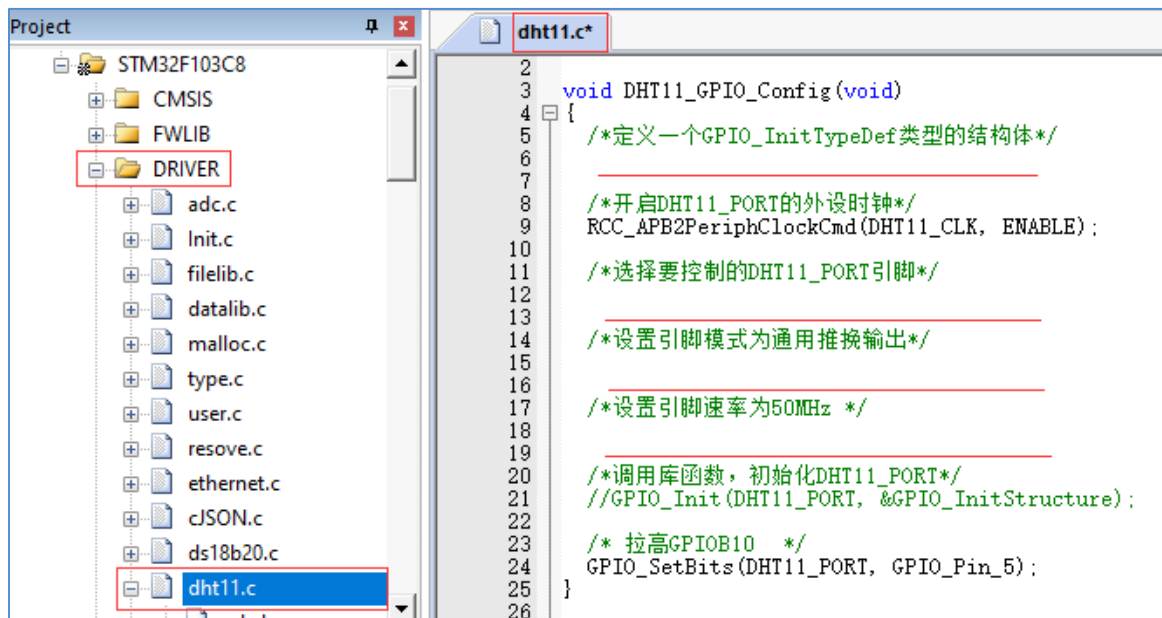
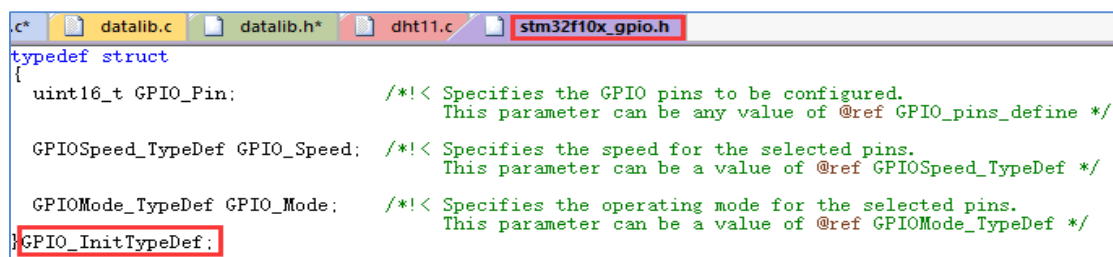


图2.10 补充DHT11_GPIO_Config()代码

程序中用到的GPIO_InitTypeDef结构体定义如图2.11所示。



```

typedef struct
{
    uint16_t GPIO_Pin;           /*!< Specifies the GPIO pins to be configured.
                                   This parameter can be any value of @ref GPIO_pins_define */

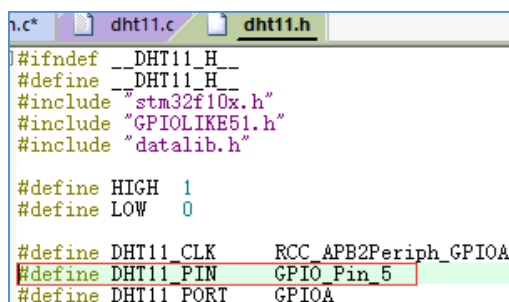
    GPIO_Speed_TypeDef GPIO_Speed; /*!< Specifies the speed for the selected pins.
                                   This parameter can be a value of @ref GPIO_Speed_TypeDef */

    GPIO_Mode_TypeDef GPIO_Mode;  /*!< Specifies the operating mode for the selected pins.
                                   This parameter can be a value of @ref GPIO_Mode_TypeDef */
}GPIO_InitTypeDef;

```

图2.11 GPIO_InitTypeDef结构体定义

DHT11与单片机的接口为单总线，它的DATA管脚接单片机PA5，在dht11.h中将PA5定义为宏名DHT11_PIN GPIO_Pin_5，如图2.12所示。程序中要控制的DHT11_PORT引脚，即为DHT11_PIN。



```

#ifndef __DHT11_H__
#define __DHT11_H__
#include "stm32f10x.h"
#include "GPIO_LIKE51.h"
#include "datalib.h"

#define HIGH 1
#define LOW 0

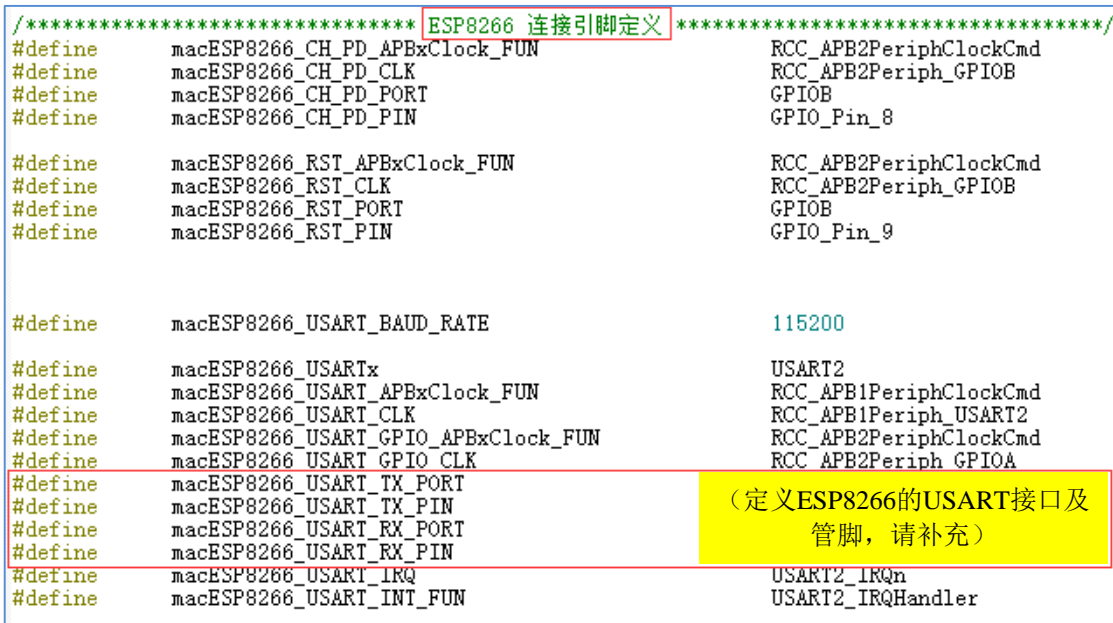
#define DHT11_CLK RCC_APB2Periph_GPIOA
#define DHT11_PIN GPIO_Pin_5
#define DHT11_PORT GPIOA

```

图2.12 DHT11管脚与单片机PA5连接宏定义

➤ ESP8266管脚定义

阅读esp8266.h和esp8266.c源程序代码，加深对ESP8266工作原理的理解。在esp8266.h文件中找到ESP8266管脚定义程序代码，如图2.13所示，根据ESP8266与STM32的硬件连接，补齐涂掉的部分代码。



```

/***** ESP8266 连接引脚定义 *****/
#define macESP8266_CH_PD_APBxClock_FUN RCC_APB2PeriphClockCmd
#define macESP8266_CH_PD_CLK RCC_APB2Periph_GPIOB
#define macESP8266_CH_PD_PORT GPIOB
#define macESP8266_CH_PD_PIN GPIO_Pin_8

#define macESP8266_RST_APBxClock_FUN RCC_APB2PeriphClockCmd
#define macESP8266_RST_CLK RCC_APB2Periph_GPIOB
#define macESP8266_RST_PORT GPIOB
#define macESP8266_RST_PIN GPIO_Pin_9

#define macESP8266_USART_BAUD_RATE 115200

#define macESP8266_USARTx USART2
#define macESP8266_USART_APBxClock_FUN RCC_APB1PeriphClockCmd
#define macESP8266_USART_CLK RCC_APB1Periph_USART2
#define macESP8266_USART_GPIO_APBxClock_FUN RCC_APB2PeriphClockCmd
#define macESP8266_USART_GPIO_CLK RCC_APB2Periph_GPIOA
#define macESP8266_USART_TX_PORT (定义ESP8266的USART接口及管脚，请补充)
#define macESP8266_USART_TX_PIN (定义ESP8266的USART接口及管脚，请补充)
#define macESP8266_USART_RX_PORT (定义ESP8266的USART接口及管脚，请补充)
#define macESP8266_USART_RX_PIN (定义ESP8266的USART接口及管脚，请补充)
#define macESP8266_USART_IRQ USART2_IRQn
#define macESP8266_USART_INT_FUN USART2_IRQHandler

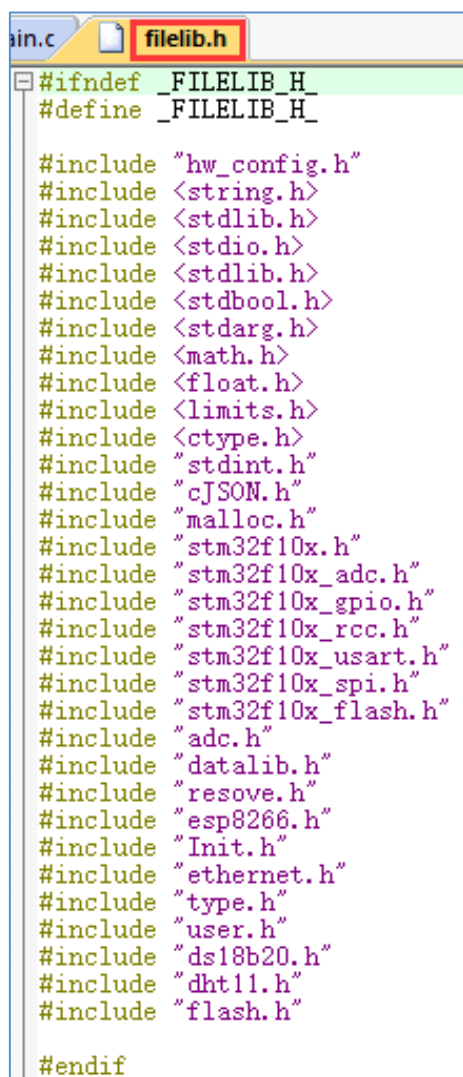
```

图2.13 ESP8266管脚定义

➤ main 程序入口函数

实验用到的filelib.h头文件，其内容如图2.14所示，需要包含于main.c中。#include后面有两种方式，<>和" "，前者在标准库中查找，后者先在path中查找，若找不到，则到标准头文件目录中查找。

工程模板已经定义了一个结构体T_Data，储存温湿度等数据，并将Dev_data声明为该结构体变量全局变量，如图2.15所示。在编程时可以直接采用结构体变量Dev_data.temp和Dev_data.humi存储采集的温湿度数据，并在Watch窗口观察它们的实时变化情况。另外一个结构体Lib_Data，用于存储WiFi名称等数据，并将P_data声明为该结构体变量。



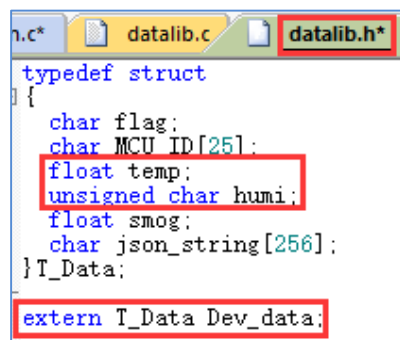
```

main.c  filelib.h
#include <filelib.h>
#ifndef FILELIB_H
#define FILELIB_H

#include "hw_config.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdarg.h>
#include <math.h>
#include <float.h>
#include <limits.h>
#include <ctype.h>
#include "stdint.h"
#include "cJSON.h"
#include "malloc.h"
#include "stm32f10x.h"
#include "stm32f10x_adc.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_usart.h"
#include "stm32f10x_spi.h"
#include "stm32f10x_flash.h"
#include "adc.h"
#include "datalib.h"
#include "resove.h"
#include "esp8266.h"
#include "Init.h"
#include "ethernet.h"
#include "type.h"
#include "user.h"
#include "ds18b20.h"
#include "dht11.h"
#include "flash.h"

#endif
  
```

图2.14 filelib.h头文件



```

n.c*  datalib.c  datalib.h*
typedef struct
{
    char flag;
    char MCU_ID[25];
    float temp;
    unsigned char humi;
    float smog;
    char json_string[256];
} T_Data;

extern T_Data Dev_data;
  
```

图2.15 T_Data结构体定义

main函数流程图如图2.16所示，终端节点建立与服务器的TCP连接后，每间隔一段时间T（可自行设定）采集一次温、湿度数据，然后透传给服务器。程序中用到的温度获取函数readtemp()和湿度获取函数Get_Humi_Value()已给出，直接调用即可。

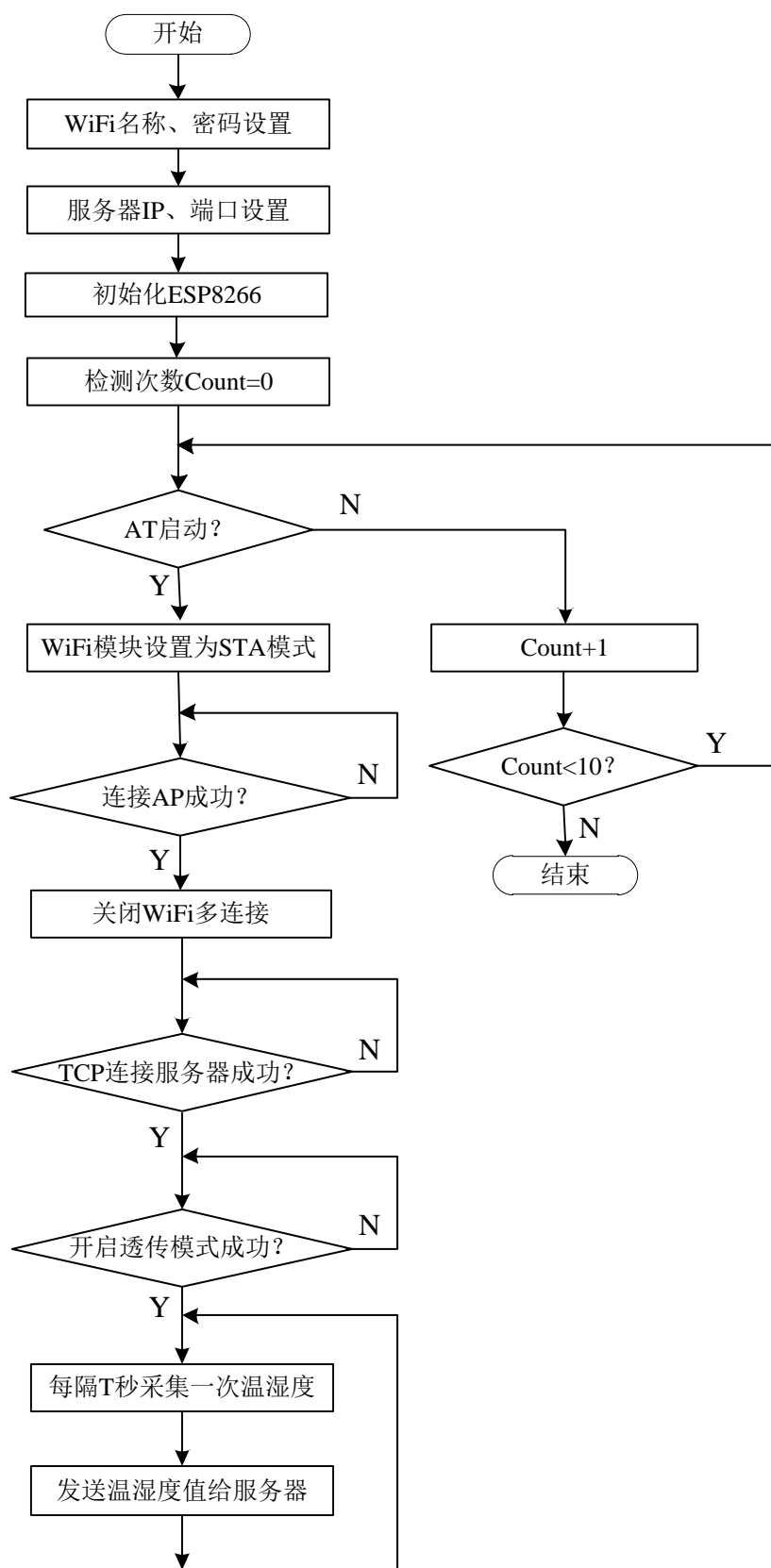


图2.16 main函数流程图

点击WiFi连接图标，找到电脑连接WiFi名称，如图2.17所示。



图2.17 本机WiFi连接

查看WiFi连接属性，如图2.18所示，记下WiFi的SSID和本机IP地址。

属性	
SSID:	ChinaNet-zkl
协议:	Wi-Fi 4 (802.11n)
安全类型:	WPA2-个人
网络频带:	2.4 GHz
网络通道:	4
链接速度(接收/传输):	144/144 (Mbps)
本地链接 IPv6 地址:	fe80::ce2:2d8c:51f9:1c71%19
IPv4 地址:	192.168.1.4
IPv4 DNS 服务器:	192.168.1.1
制造商:	Realtek Semiconductor Corp.
描述:	Realtek RTL8192EU Wireless LAN 802.11n USB 2.0 Network Adapter
驱动程序版本:	1030.38.328.2019
物理地址(MAC):	30-B4-9E-99-DC-D7

图2.18 本机WiFi连接属性

图2.19给出了main.c源代码，根据本机WiFi属性，补齐main.c中涂掉的代码。

```
#include "filelib.h"
int main(void)
{
    char *sendtxt = (char*)malloc(48);
    strcpy(P_data.wifi_ssid, " "); //设置连接 WiFi 名称
    strcpy(P_data.wifi_psd, " "); //设置连接 WiFi 密码
    strcpy(P_data.wifi_ip, " "); //设置连接服务器 IP
    strcpy(P_data.wifi_port, " "); //设置连接服务器端口
    ESP8266_Init (); //初始化 ESP8266
    macESP8266_CH_ENABLE(); //使能 ESP8266
    ESP8266_AT_Test(); //检测 ESP8266 的 AT 是否启动
    ESP8266_Net_Mode_Chose(STA); //设置 WiFi 模块为 STA 模式

    //连接 WiFi
    while (!ESP8266_JoinAP(P_data.wifi_ssid, P_data.wifi_psd))
    {
        //连接 WiFi 失败，重连
    }

    ESP8266_Enable_MultipleId(DISABLE); //关闭 WiFi 多连接

    //连接到远程服务器
    while (!ESP8266_Link_Server(enumTCP, P_data.wifi_ip, P_data.wifi_port, Single_ID_0))
    {
        //连接服务器失败，重连
    }

    while (!ESP8266_UnvarnishSend()); //开启穿透模式

    while(1)
    {
        Delay_ms(1500); //每隔 1500ms 获取一次温湿度
        Dev_data.temp = ; //获取温度
        sprintf(sendtxt, "%s%.2f%s", "温度: ", Dev_data.temp, "°C"); //转换为格式字符串
        ESP8266_SendString ( ENABLE, sendtxt, 0, Single_ID_0 ); //发送温度值到服务器
        Dev_data.humi = ; //获取湿度
        sprintf(sendtxt, "%s%u%s", "湿度: ", Dev_data.humi, "%RH"); //转换为格式字符串
        ESP8266_SendString ( ENABLE, sendtxt, 0, Single_ID_0 ); //发送温度值到服务器
    }
}
```

图 2.19 main.c 源代码

➤ AT指令在ESP8266程序中的应用

main.c中的ESP8266_AT_Test、ESP8266_Net_Mode_Chose和ESP8266_JoinAP函数的源代码在esp8266.c中，如图2.20所示。查阅《ESP8266 AT指令集》，补齐调用ESP8266_Cmd的函数参数。

```

void ESP8266_AT_Test ( void )
{
    char count=0;                                //检测次数变量赋初值
    macESP8266_RST_HIGH_LEVEL();                 //复位管脚置高电平
    Delay_ms ( 1000 );                           //延时 1000mS
    while ( count < 10 )
    {
        if( ESP8266_Cmd ( ) ) return; //发送 AT 启动测试指令
        ESP8266_Rst();                     //复位 ESP8266
        ++ count;                          //检测次数+1
    }
}

bool ESP8266_Net_Mode_Choose ( ENUM_Net_ModeTypeDef enumMode )
{
    switch ( enumMode )
    {
        case STA:          // STA 模式
            return ESP8266_Cmd ( );
        case AP:           // AP 模式
            return ESP8266_Cmd ( );
        case STA_AP:       // STA+AP 模式
            return ESP8266_Cmd ( );
        default:
            return false;
    }
}

bool ESP8266_JoinAP ( char * pSSID, char * pPassWord )
{
    char cCmd [120];
    sprintf ( cCmd, "AT+CWJAP=\"%s\",\"%s\"", pSSID, pPassWord ); //字符串格式化
    return ESP8266_Cmd ( ); //发送 AT 指令连接外部 WiFi
}

bool ESP8266_Link_Server ( ENUM_NetPro_TypeDef enumE, char * ip, char * ComNum, ENUM_ID_NO_TypeDef id)
{
    char cStr [100] = { 0 }, cCmd [120];
    switch ( enumE )
    {
        case enumTCP:
            sprintf ( cStr, "\"%s\",\"%s\",%s", "TCP", ip, ComNum ); //TCP 协议
            break; //字符串格式化
        case enumUDP:
            sprintf ( cStr, "\"%s\",\"%s\",%s", "UDP", ip, ComNum ); //UDP 协议
            break; //字符串格式化
        default:
            break;
    }
    if ( id < 5 )
        sprintf ( cCmd, "AT+CIPSTART=%d,%s", id, cStr ); //字符串格式化
    else
        sprintf ( cCmd, "AT+CIPSTART=%s", cStr ); //字符串格式化
    return ESP8266_Cmd ( ); //发送 AT 指令连接外部服务器
}

```

图 2.20 esp8266.c 部分源代码

3. 系统调试

(1) 程序下载

按照《Keil5安装与使用》中的方法，进行硬件连接，完成程序编译，编译无误后下载到单片机中，如图2.21所示。然后调试、运行程序，开发板会自动连接到本地WiFi。如果已经建立与服务器端的TCP连接，开发板采集的温湿度数据则上传到SSCOM中。

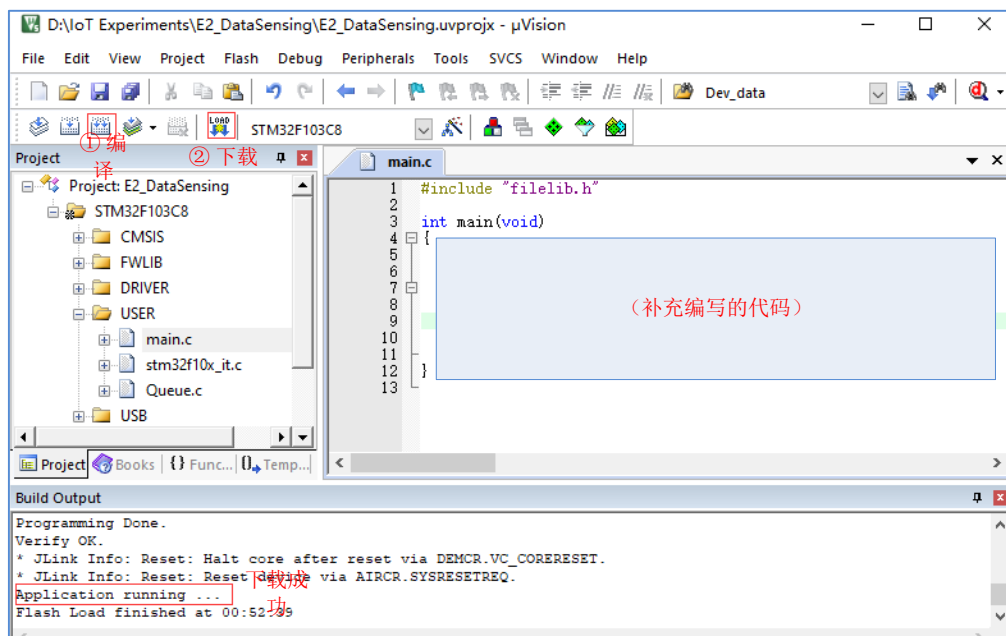


图2.21 程序编译下载

(2) 系统调试

➤ 打开串口调试工具SSCOM，如图2.22所示。实验以开发板作为客服端（Client），以电脑作为服务器端（Server）。所以，在SSCOM中，端口号选择TCPServer，服务器IP（本地IP）、端口设置与main函数中的一致。

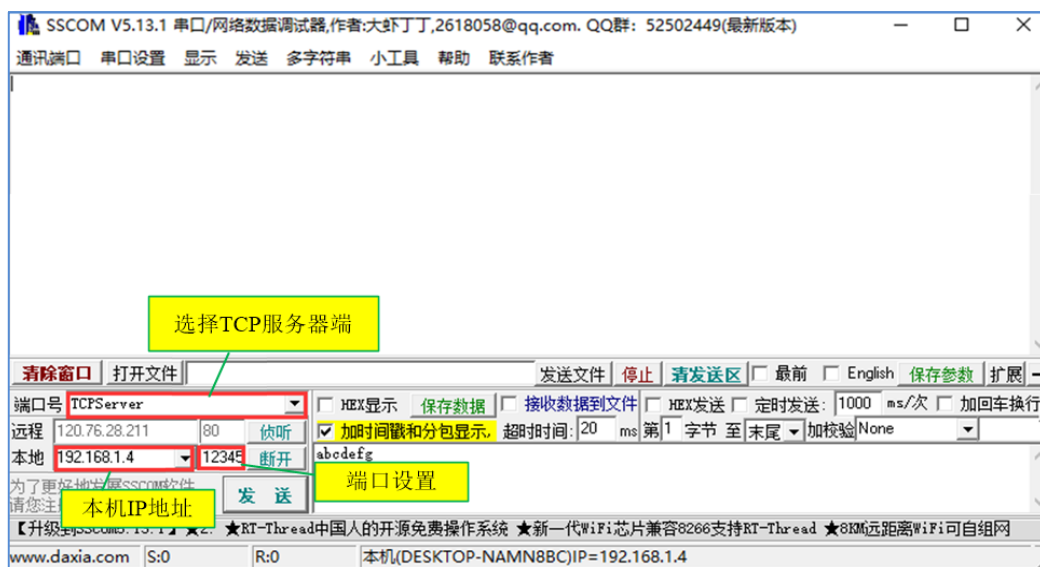


图2.22 SSSCOM设置

➤ 按下SSCOM“侦听”按钮，若系统显示“已连接”，表明开发板已经建立了与服务端端的TCP连接，则可以看到温湿度的实时监测数据，如图2.23所示，照相记录。

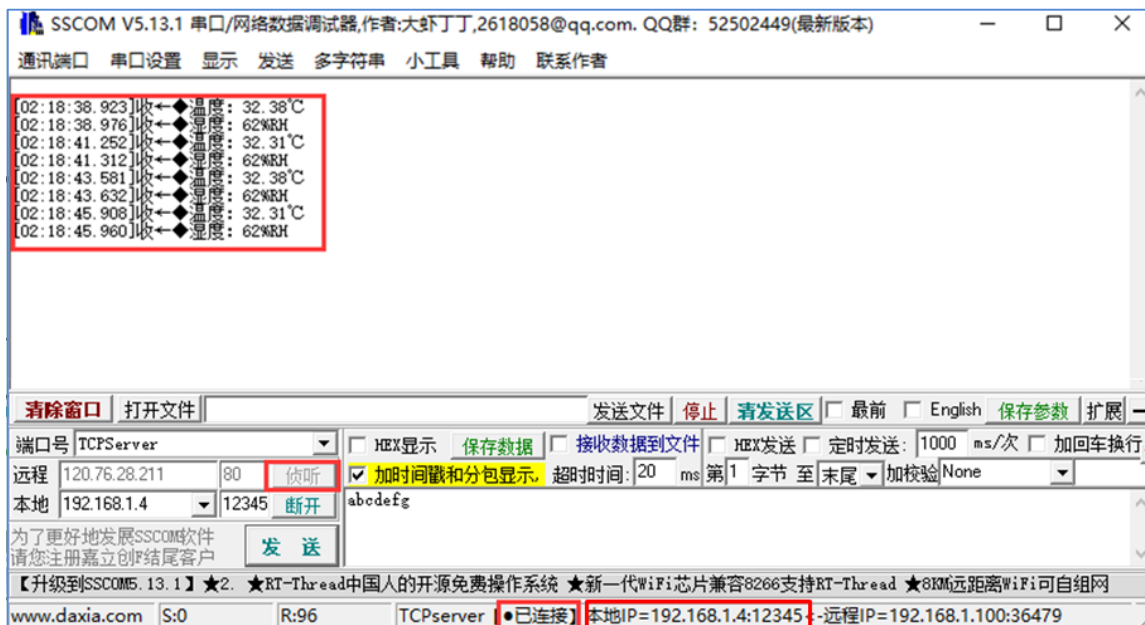


图2.23 调试结果

➤ 按照图2.24所示步骤，在Watch窗口查看所采集温湿度，记录实验结果（照相）。

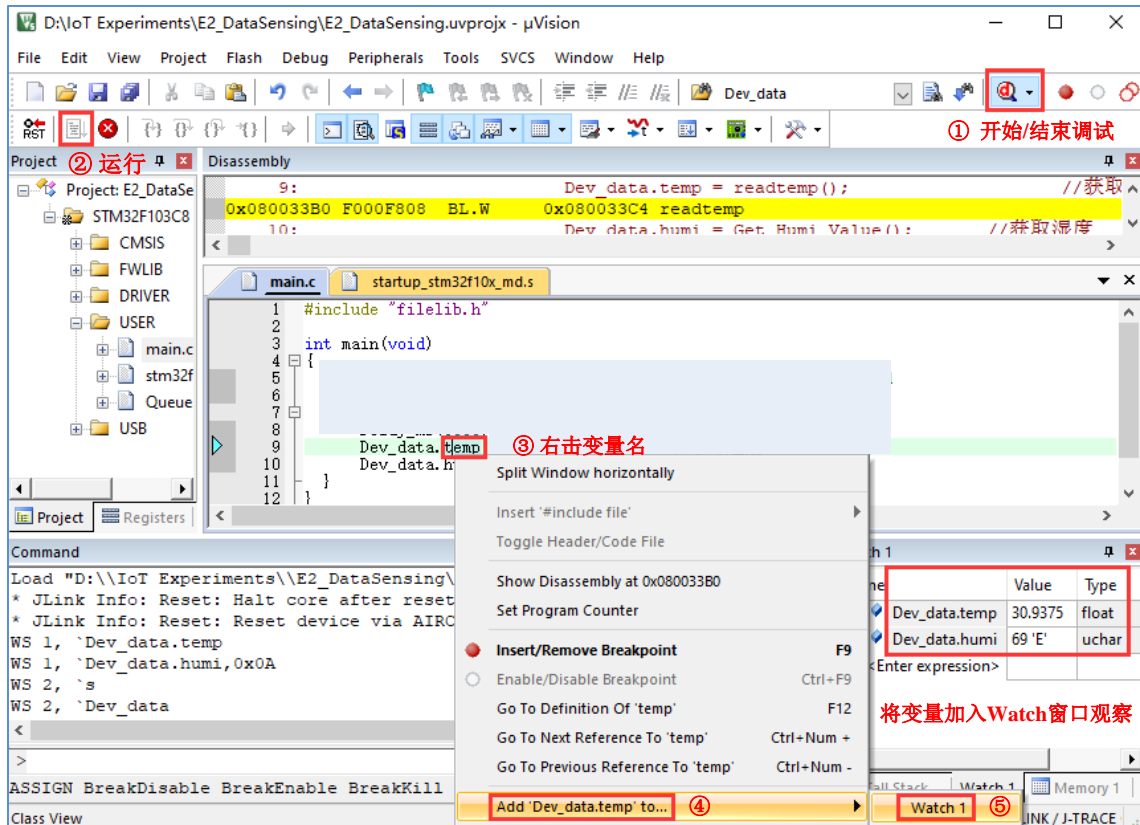


图2.24 实验结果观察

【思考与练习】

1. 实验中温湿度传感器 DHT11、WiFi 模块 ESP-12S 与 MCU 的接口分别是什么？是物理接口还是虚拟接口？管脚是怎么连接的？
2. 对照 DHT11 程序（dht11.c）和 ESP8266 程序（esp8266.c）说明各函数的作用，分别画出其程序流程图。
3. 给 DHT11 的头文件 dht11.h 和 ESP8266 的头文件 esp8266.h 每行代码加上注释（已有注释的行忽略）。
4. 把实验中的传感器分别换为温度传感器 DS18B20 和烟雾传感器 MQ-2，完成温度采集和烟雾浓度采集实验。
5. 若不采用串口调试工具 SSCOM，自行编写服务器端的应用程序，接收并显示开发板的温湿度实时监测数据。

【预习要求】

1. 根据实验指导书提供的方法，在 Keil5 开发平台下，自行练习套用工程模板新建工程。
2. 认真阅读《Keil5 安装与使用教程》，熟悉在 Keil5 环境下使用 J-Link 下载程序，进行单片机系统调试的方法。
3. 根据提供的参考资料，在单片机程序中定义 ESP8266 的 USART 接口及管脚。
4. 熟悉 ESP8266 的 AT 指令，编写实验相关程序代码。
5. 认真阅读本实验指导书，熟悉实验步骤和实验方法。

实验3 网络传输与数据存储实验

【实验目的】

1. 掌握网络传输与数据存储系统架构，掌握基于 Socket 实现数据 TCP 网络传输的编程方法。
2. 掌握 MySQL 数据库的命令操作方法，以及编程操作方法。
3. 掌握 Socket 服务端程序的设计方法。

【实验设备】

- | | |
|------------------|-----|
| 1. 物联网实训工作站 | 1 台 |
| 2. 开发板（已焊接基础训练包） | 1 块 |
| 3. J-Link 下载器 | 1 个 |

【实验要求】

1. 根据所给物联网实训系统网络拓扑结构，设计网络传输与数据存储系统架构。
2. 熟悉 MySQL 数据库的基本操作方法，新建数据库和数据表，实现感知数据的存储。
3. 改写 Socket 客户端程序，每隔 T 秒（可以自己在程序中设定）采集一次温湿度数据，基于 TCP 通信协议通过网络透传到服务端。
4. 改写 Socket 服务端程序，接收来自客户端的数据，解析并存入数据库。

【实验原理】

1. 网络传输与数据存储系统架构

物联网实训系统的网络拓扑结构，如图3.1所示。由于目前条件限制，服务器暂时不能够使用，所以，网络传输与数据存储系统只好把工作站电脑暂时当服务器使用。系统架构可以有2种形式，如图3.2所示。图3.2（a）是把工作站电脑既当服务器又当用户终端使用，图3.2（b）是把一台电脑当服务器使用，另外一台电脑当用户终端使用。

把图3.1中的PC作为用户终端，才是我们真正想搭建的系统架构。无论采用哪种系统架构，程序设计都相差无几，无非是部署的位置不同而已。

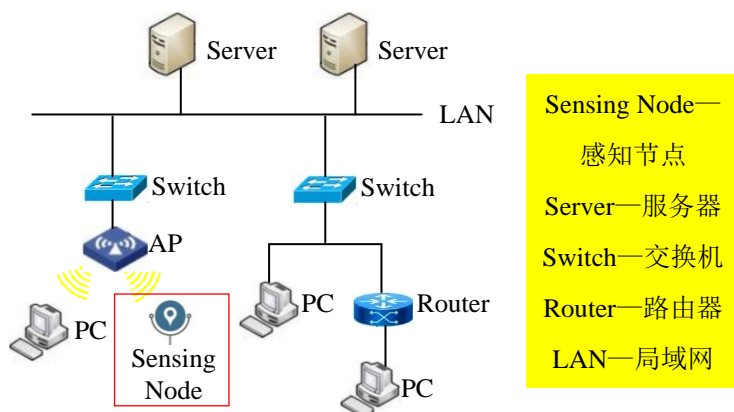


图3.1 物联网实训系统网络拓扑结构

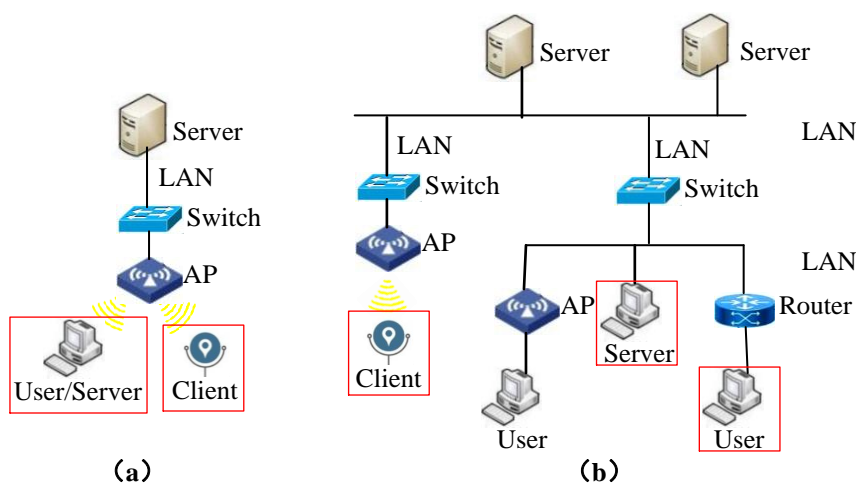


图3.2 网络传输与数据存储系统架构

2. 系统工作原理

网络传输与数据存储系统的感知节点作为客户端（Client），实时采集温湿度数据，然后通过WiFi模块，以TCP协议将采集数据实时无线传输给服务端（Server）存于数据库中。

3. 系统设计

（1）温湿度数据采集与传输

在实验2中，已经给出了温湿度感知节点硬件设计原理图和PCB，并完成了温湿度数据采集，同时实现了WiFi模块进行无线数据传输的软件调试。本实验的数据采集及无线传输原理和实验2的完全相同，不再赘述。

（2）Socket通信原理

本实验以Socket方式实现采集数据的TCP传输。Socket 的中文翻译过来就是“套接字”，它是在应用层和传输层之间的一个抽象层，它把 TCP/IP 层复杂的操作抽象为几个简单的接口，供应用层调用实现不同终端在网络中的通信。Socket通信模型是服务器

与客户端之间的通信，两端都建立了一个Socket对象，然后通过Socket对象对数据进行传输，其工作流程如图3.3所示。

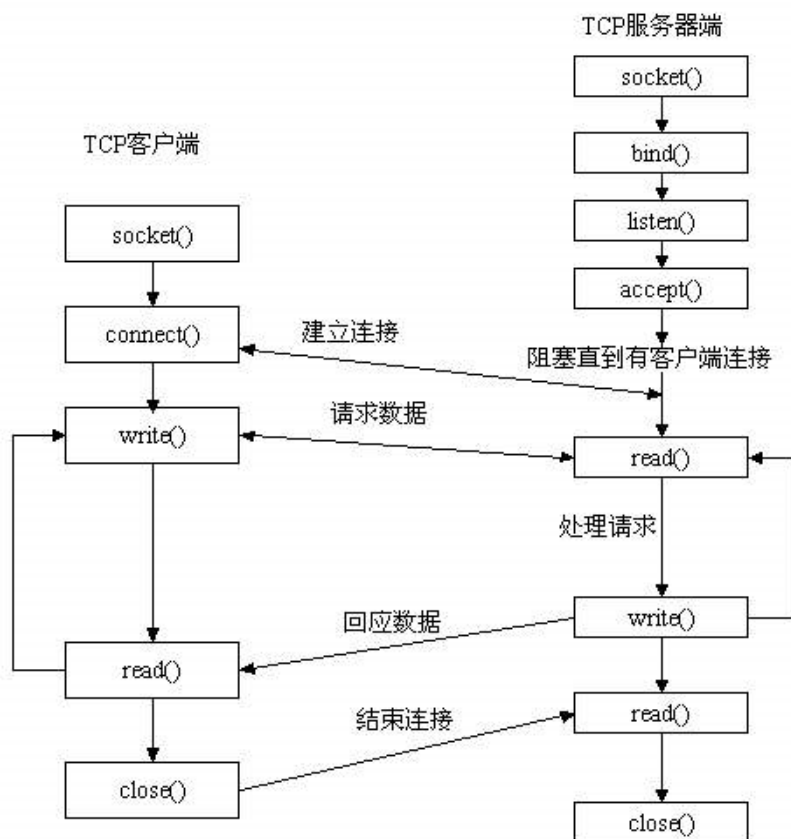


图3.3 Socket通信示意图

感知节点客户端（Client）程序在实验2中已经以AT指令编写完成，本实验只需要设计Socket服务端程序。

（3）数据库设计

本实验采用MySQL数据库进行数据存储。MySQL是一种关系型数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。本实验只存储温湿度数据和采集时间，数据表见表3.1。

表 3.1 采集数据表 acquisitiondata

字段名	字段类型	字段宽度	小数位数	是否为主键	备注
numRecord	int	11	0	是	记录号（自增列）
tempData	varchar	5	0	否	温度数据
humiData	varchar	3	0	否	湿度数据
datetimeDA	varchar	30	0	否	采集时间

【实验步骤】

1. 新建数据库和数据表

(1) 配置/安装MySQL（已经完成配置或安装的请忽略）

使用mysql-8.0.18-winx64.zip压缩包，按照《Windows下MySQL-8.0.18-winx64免安装设置教程》完成MySQL软件的安装。

如果选用安装版本的MySQL软件，请自行查阅安装方法进行安装。

(2) 连接本地数据库

- 以管理员身份打开命令窗口，改变当前路径为MySQL安装路径：

```
cd D:\Program Files (x86)\mysql-8.0.18-winx64\bin
```

- 键入以下命令，启动MySQL服务，注意配置MySQL时的服务名称：

```
net start localmysql
```

按照图3.4的方法，可以查询MySQL服务名称。

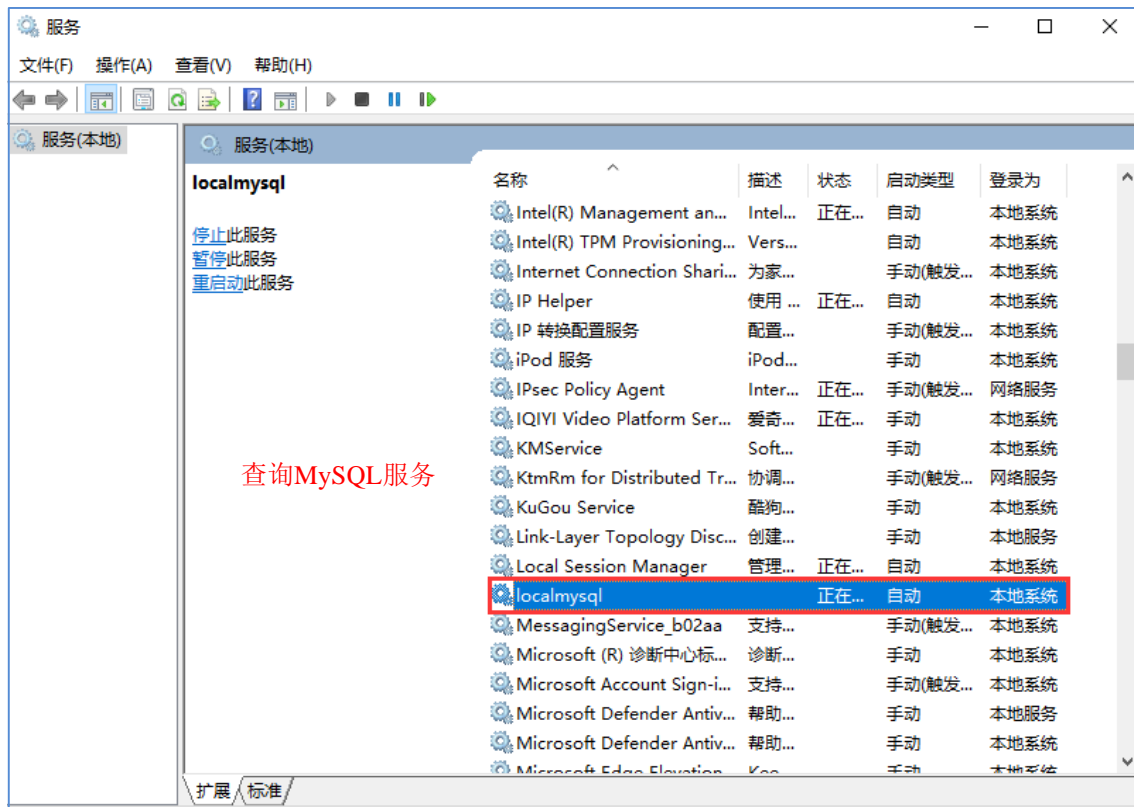


图3.4 查询MySQL数据库服务

- 键入以下命令，连接本地数据库：

```
mysql -u <用户名> -p
```

输入密码，出现mysql>，则连接上了本地数据库，如图3.5所示。

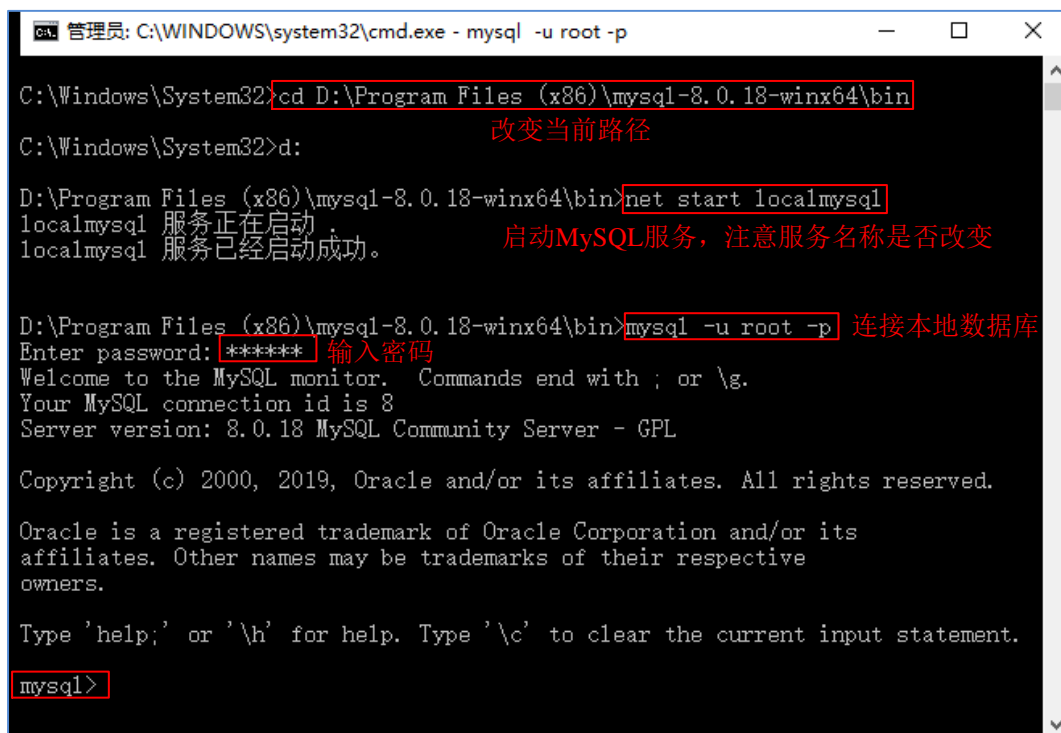


图3.5 连接本地数据库MySQL

(3) 新建数据库

如果我们不想在以mysql命名的数据库作为实验项目的数据库,可以重新建立一个数据库,如iotdatabase,如图3.6所示。有关数据库操作的命令可以查阅《MySQL命令大全》,键入以下命令,新建数据库iotdatabase:

```
create database iotdatabase;
```

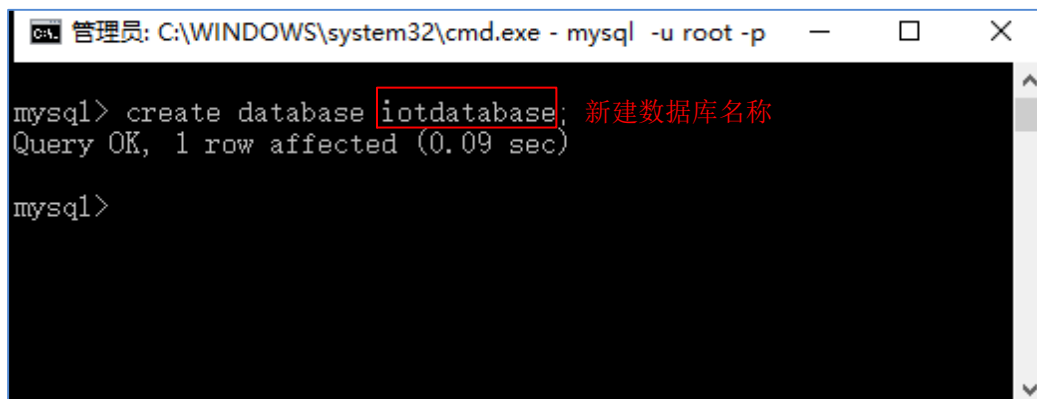


图3.6 新建数据库iotdatabase

(4) 新建数据表

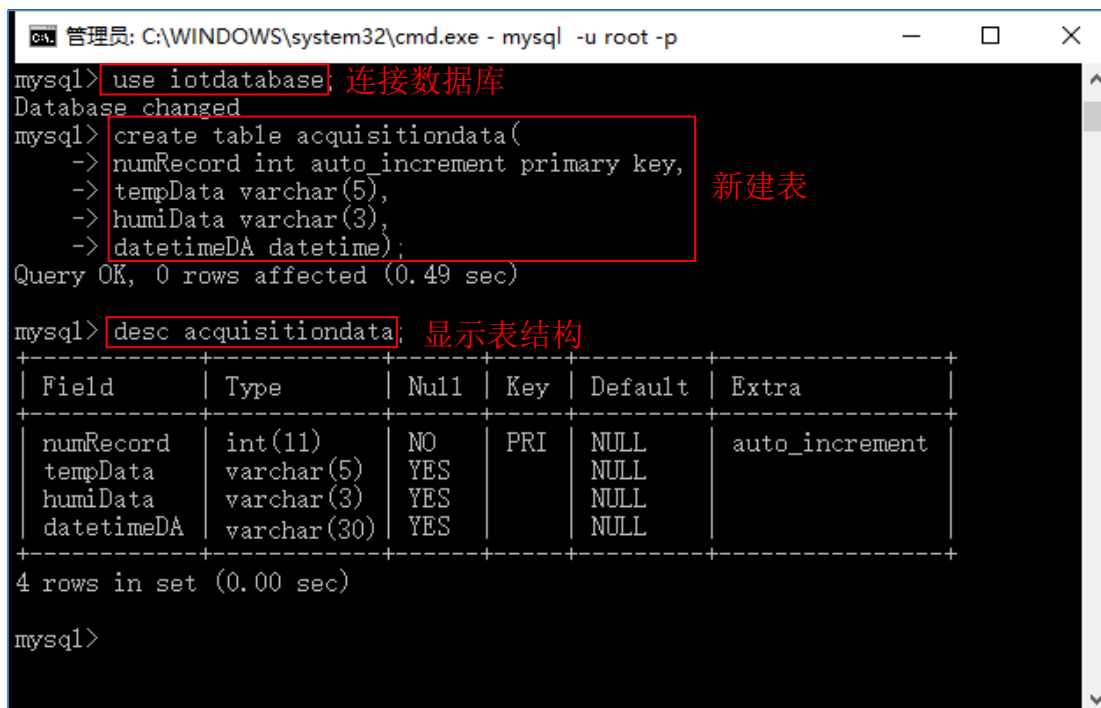
先连接所建数据库iotdatabase,然后键入以下命令,新建表 acquisitiondata,如图3.7所示。

表acquisitiondata的结构见表3.1。

```

use iotdatabase;
create table acquisitiondata(
    -> numRecord int auto_increment primary key,
    -> tempData varchar(5),
    -> humiData varchar(3),
    -> datetimeDA varchar(30));

```



The screenshot shows a MySQL command prompt window with the following commands and output:

```

mysql> use iotdatabase; 连接数据库
Database changed
mysql> create table acquisitiondata(
    -> numRecord int auto_increment primary key,
    -> tempData varchar(5),
    -> humiData varchar(3),
    -> datetimeDA datetime); 新建表
Query OK, 0 rows affected (0.49 sec)

mysql> desc acquisitiondata; 显示表结构
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| numRecord | int(11) | NO | PRI | NULL | auto_increment |
| tempData | varchar(5) | YES | | NULL | |
| humiData | varchar(3) | YES | | NULL | |
| datetimeDA | varchar(30) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

图3.7 新建数据表acquisitiondata

2. 编写 Socket 客户端程序

Socket客户端程序在实验2中已经编写完成，本实验中只需改写一下发送的字符串格式，满足本实验的通信协议即可，如图3.8标红的行。

所谓通信协议，是指服务器接收到客户端的数据后，要解析除感知数据和每次发送数据的结束标识符，需要与客户端约定收发数据的格式。本实验采用的通信协议如下：“温度数据+湿度数据#”。

```

while(1)
{
    Delay_ms(1500);                //每隔 1500ms 采集一次温湿度
    Dev_data.temp = readtemp();     //获取温度数据
    Dev_data.humi = Get_Humi_Value(); //获取湿度数据
    sprintf(sendtxt, "%.2f%s%s", Dev_data.temp, "+", Dev_data.humi, "#");
    //温湿度数据以“+”间隔，“#”为结束标识符
    ESP8266_SendString ( ENABLE, sendtxt, 0, Single_ID_0 ); //发送温湿度值到服务器
}

```

图 3.8 客户端程序 main.c 源代码改写

3. 编写 Socket 服务端程序

(1) 创建SeverProgram工程

本实验以VS2019作为开发平台，以C语言进行Socket服务端程序开发。先参照《使用VS2019编写C语言Socket服务端程序》的方法建立SeverProgram工程，如图3.9所示。

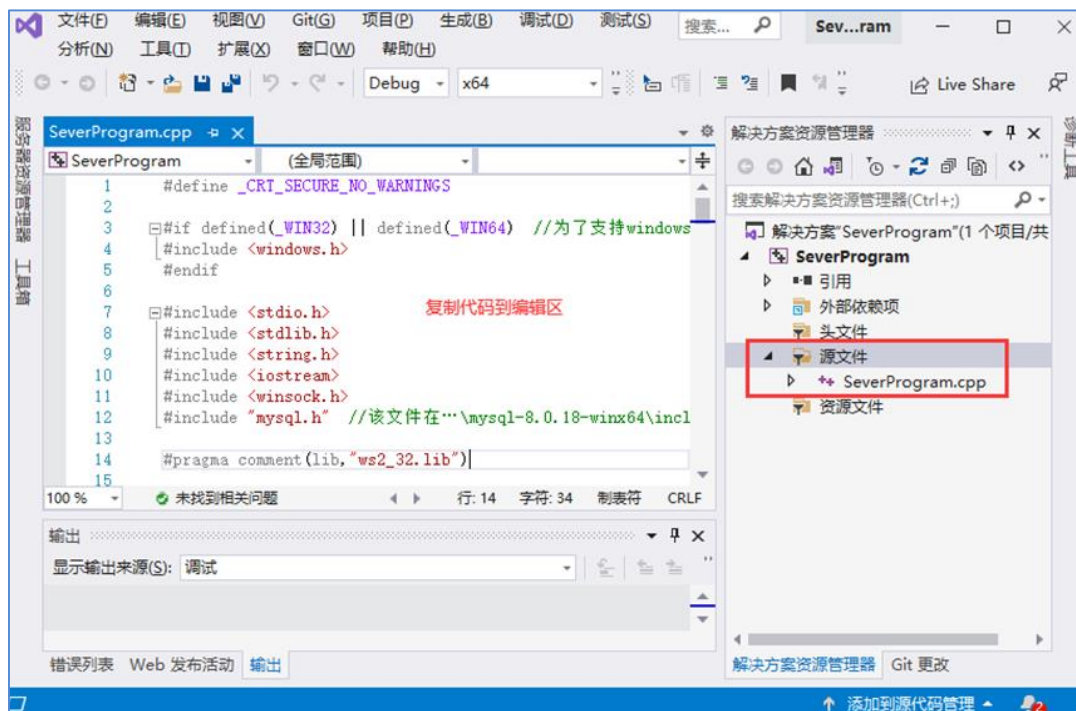


图3.9 新建Socket服务端工程

(2) 编写Socket服务端程序

Socket服务端程序主要接收来自感知节点的温湿度采集数据，并存入数据表 acquisitiondata中，其程序流程图如图3.10所示。

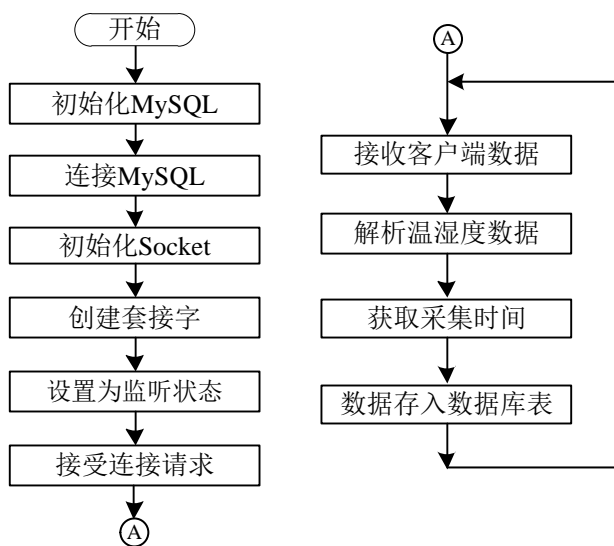


图3.10 Socket服务端程序main函数流程图

Socket服务端程序如图3.11所示，其中涂掉的部分是需要根据实际情况改写的。

```
#define _CRT_SECURE_NO_WARNINGS
#ifdef _WIN32 || defined(_WIN64) //为了支持windows平台上的编译
#include <windows.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <winsock.h>
#include "mysql.h" //该文件在...\mysql-8.0.18-win64\include下
#pragma comment(lib, "ws2_32.lib")

using namespace std;
void initialization();
int main() {
    //以下是MySQL数据库操作程序
    MYSQL mysql;
    const char* host = " "; //声明MySQL的句柄
    const char* user = " "; //数据库服务器IP（本地服务器host = "localhost"）
    const char* passwd = " "; //连接MySQL的用户名
    const char* db = " "; //连接MySQL的用户密码
    unsigned int port = 3306; //连接数据库的名字
    const char* unix_socket = NULL; //这是MySQL的服务器的端口，如果没有修改过的话就是3306
    unsigned long client_flag = 0; //unix_socket这是unix下的，在Windows下把它设置为NULL
    char query_str[200]; //这个参数一般为0
    mysql_init(&mysql); //MySQL命令字符串
    //连接MySQL
    if ((sock = mysql_real_connect(&mysql, host, user, passwd, db, port, unix_socket, client_flag)) == NULL)
    {
        exit(1); //exit(0);正常的关闭所有程序; exit(1):有错误的关闭所有程序
    }

    //以下是Socket通信服务端程序
    int recv_len = 0; //定义长度变量
    int len = 0;

    //定义服务端套接字，接受请求套接字
    SOCKET s_server;
    SOCKET s_accept;

    //服务端地址客户端地址
    SOCKADDR_IN server_addr;
    SOCKADDR_IN accept_addr;

    //填充服务端信息
    server_addr.sin_family = AF_INET; //协议簇类型：TCP/IP-IPv4
    server_addr.sin_addr.S_un.S_addr = inet_addr(" "); //服务器IP（阿里云ECS用内网IP）
    server_addr.sin_port = htons( ); //服务器端口

    //初始化套接字
    WORD w_req = MAKEWORD(2, 2); //版本号
    WSADATA wsadata;
    int err;
    err = WSAStartup(w_req, &wsadata); //初始化套接字
    if (err != 0) {
        WSACleanup(); //中止套接字
    }
}
```



```

//创建套接字
s_server = socket(AF_INET, SOCK_STREAM, 0);    //SOCK_STREAM—TCP流;
//SOCK_DGRAM—UDP数据报; SOCK_RAW—原始套接字
if (bind(s_server, (SOCKADDR*)&server_addr, sizeof(SOCKADDR)) == SOCKET_ERROR) {
    WSACleanup();
}

//设置套接字为监听状态
if (listen(s_server, SOMAXCONN) < 0) {
    WSACleanup();
}

//接受连接请求
len = sizeof(SOCKADDR);
s_accept = accept(s_server, (SOCKADDR*)&accept_addr, &len);
if (s_accept == SOCKET_ERROR) {
    WSACleanup();    //释放DLL资源
    return 0;
}

//接收数据
while (1) {
    char recv_buf[100] = { "0" };    //定义接受缓冲区，并赋初值Null（字符串结束）
    recv_len = recv(s_accept, recv_buf, 100, 0);    //接收数据长度
    if (recv_len < 0) {
        break;
    }
    else {
        //获取温湿度数据
        char* p1 = NULL;
        char* p2 = NULL;
        char* p3 = NULL;
        string tempData = strtok_s(recv_buf, "+", &p2);
        string humiData = strtok_s(p2, "#", &p3);

        //获取采集时间
        char str[50];
        SYSTEMTIME st;    //系统时间变量
        GetLocalTime(&st);    //获取本地时间
        sprintf(str, "%u-%u-%u %u:%u:%u", st.wYear, st.wMonth, st.wDay, st.wHour, st.wMinute, st.wSecond);
        //将采集时间转换为字符串
        string datetimeDA = str;

        //拼接MySQL查询命令字符串
        string sql = "insert into acquisitiondata(tempData,humiData,datetimeDA) values ('" + tempData + "','" +
humiData + "','" + datetimeDA + "')";
        strcpy_s(query_str, sql.c_str());

        //将采集温湿度数据插入MySQL数据表acquisitiondata
        mysql_real_query(&mysql, query_str, strlen(query_str));
    }
}

//关闭服务端套接字
closesocket(s_server);
closesocket(s_accept);
WSACleanup();    //释放DLL资源
return 0;
}

```

图 3.11 服务端程序 main.c 源代码改写

(3) 配置工程项目属性

➤ 项目一>属性一>平台，选择x64，如图3.12所示。

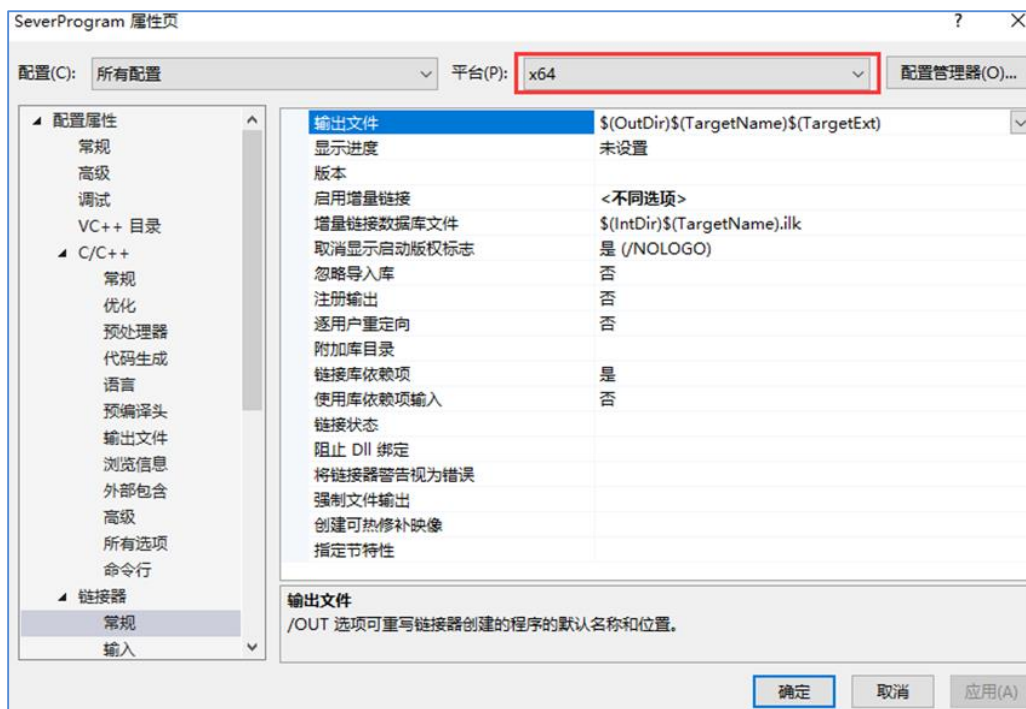


图3.12 项目平台配置

➤ 项目一>属性一>配置管理器一>活动解决方案平台，选择x64，如图3.13所示。

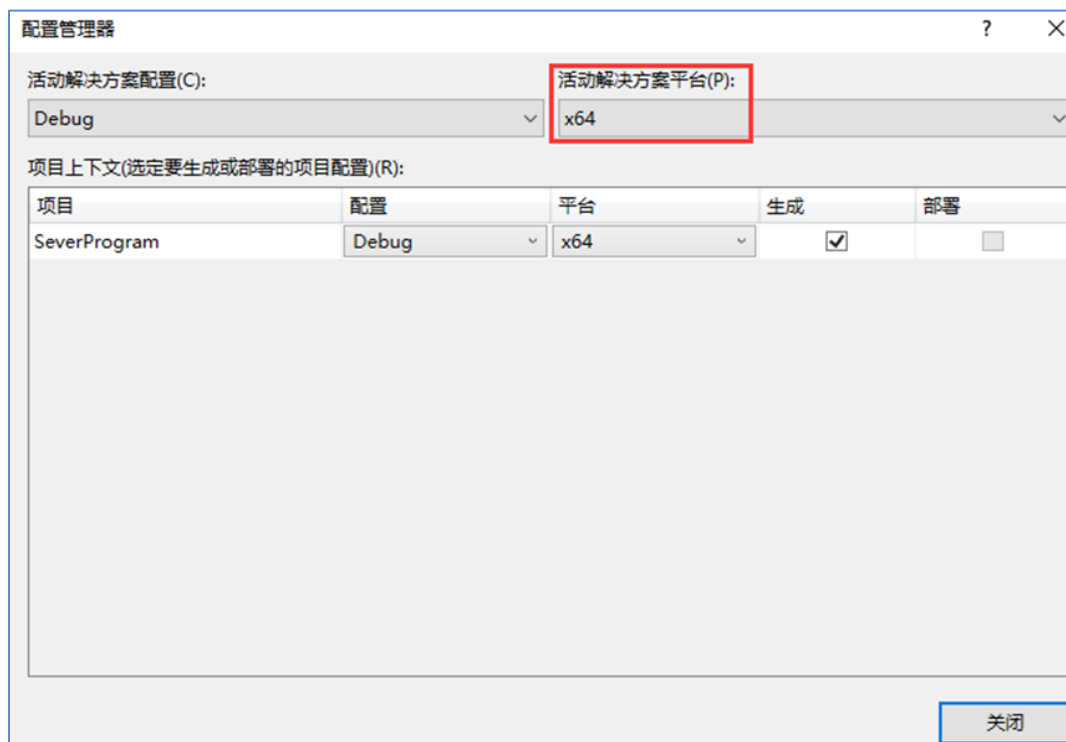


图3.13 配置管理器

➤ 项目一>属性一>C/C++ 一>常规一>附加包含目录里面添加 MySQL include文件
夹路径：D:\Program Files (x86)\mysql-8.0.18-winx64\include，如图3.14所示。

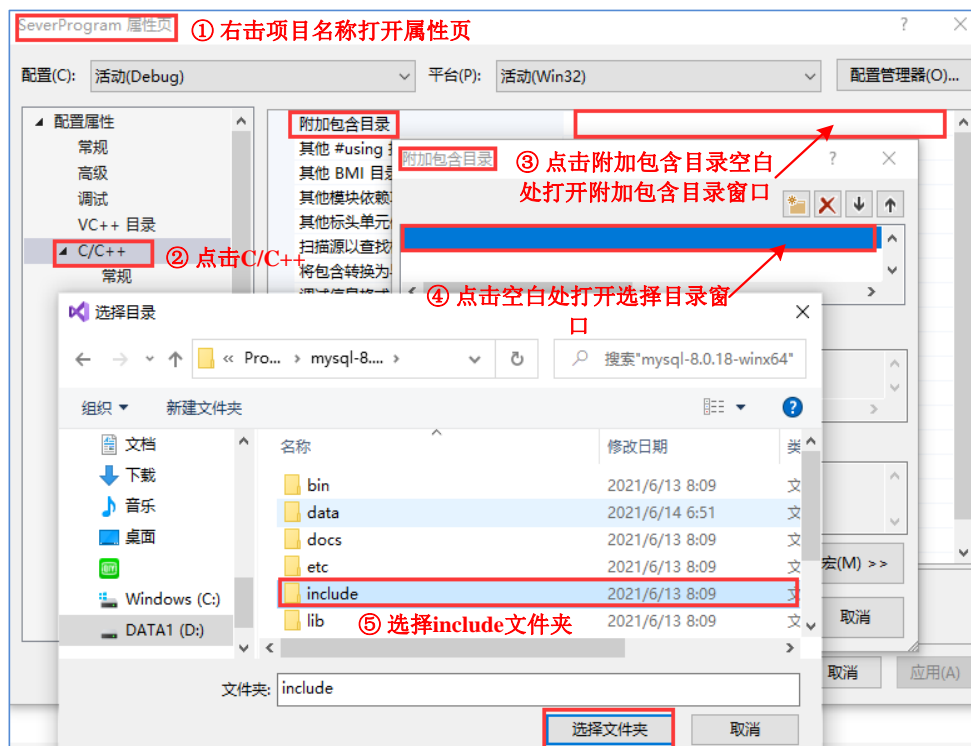


图3.14 添加 MySQL include文件夹路径

➤ 项目一>属性一> C/C++ 一>常规一>使用Windows运行时扩展，选“否”，如图3.15所示。

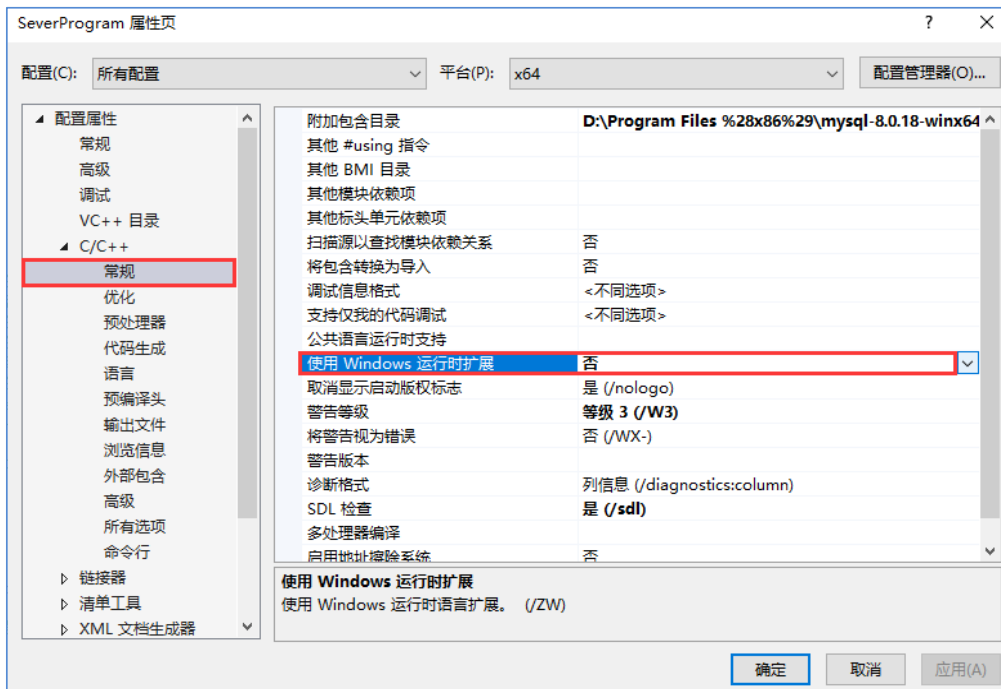


图3.15 使用Windows运行时扩展配置

➤ 项目一>属性一>链接器一>常规一>附加库目录，添加MySQL lib文件夹路径：
D:\Program Files\MySQL\MySQL Server 5.6\lib，如图3.16所示。

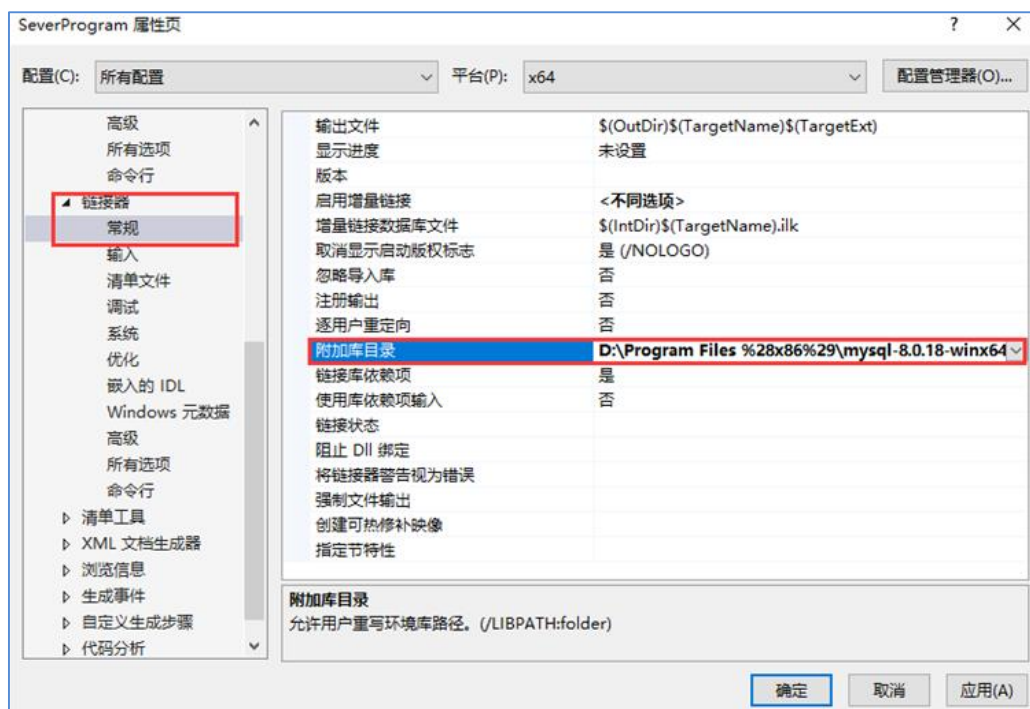


图3.16 添加MySQL lib文件夹路径

➤ 项目一>配置属性一>链接器一>输入一>附加依赖项，添加 libmysql.lib，如图3.17所示。

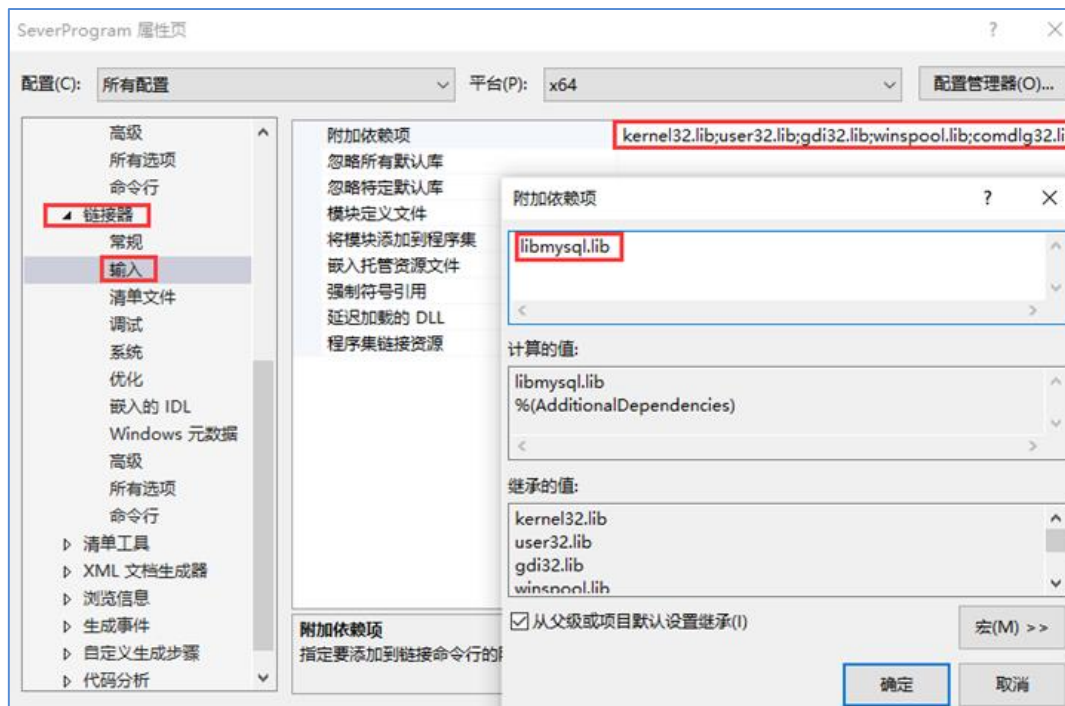


图3.17 添加 libmysql.lib附加依赖项

➤ 项目一>配置属性一>调试一>环境，把路径设置为MySQL的bin文件夹即可，注意末尾分号要用半角的，如图3.18所示。

PATH=D:\Program Files (x86)\mysql-8.0.18-winx64\bin;D:\Program Files (x86)\mysql-8.0.18-winx64\lib;

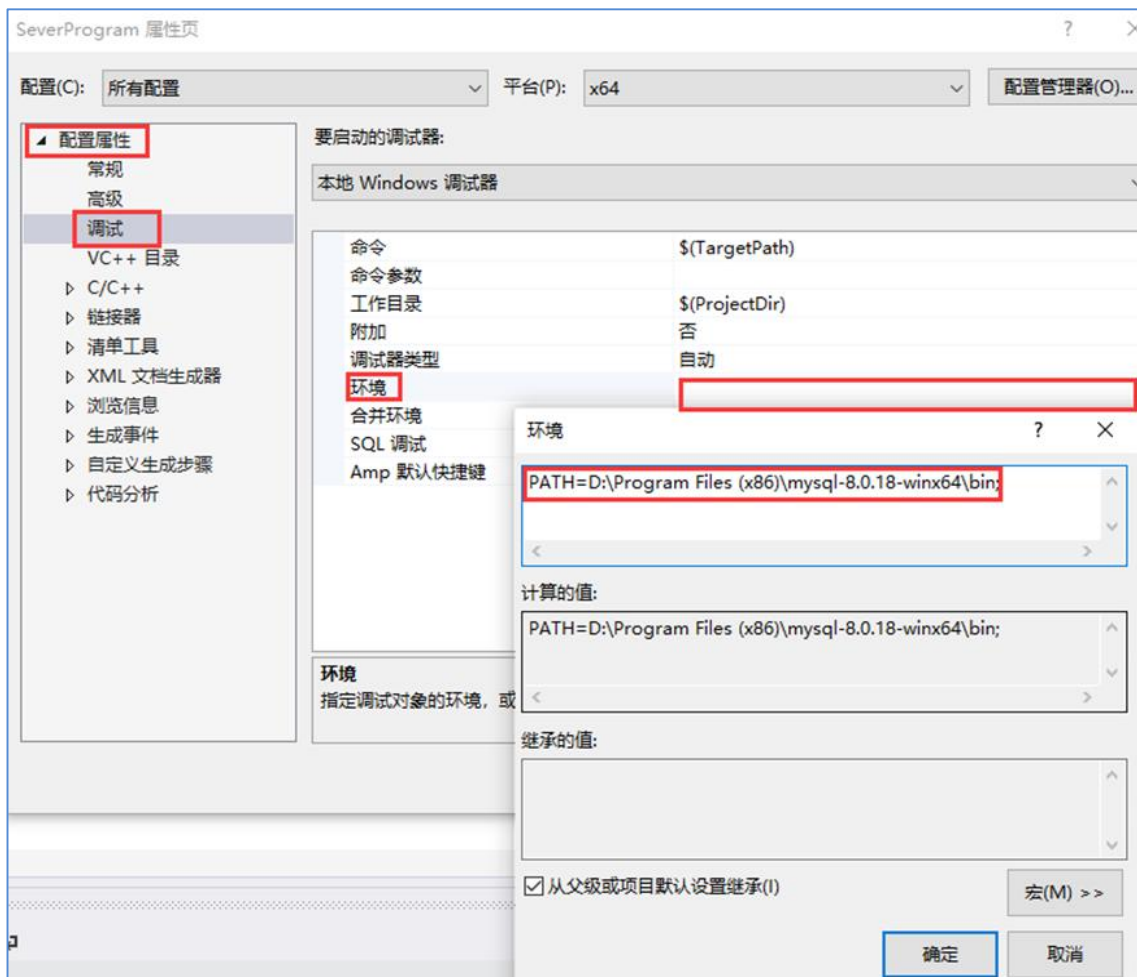


图3.18 调试环境设置

➤ 如果把得到的可执行文件ServerSocket.exe拿到另外一台电脑上运行，往往会提示缺少某些动态链接库（dll），而导致程序无法正常运行。这是因为VS编译器默认是动态编译，需要配置成静态编译，得到的可执行文件才能在另外一台电脑上正常运行。

- ✧ 动态编译：在程序编译时 dll 不被连接到目标代码中，而是在程序运行时才被载入。
- ✧ 静态编译：在程序编译时 dll 会被连接到目标代码中，程序运行时将不再需要该静态库。

在属性页面下选择“C/C++ -->代码生成-->运行库-->多线程调试（MTd），如图3.19所示。其中，MTd是静态编译；MDd是动态编译。

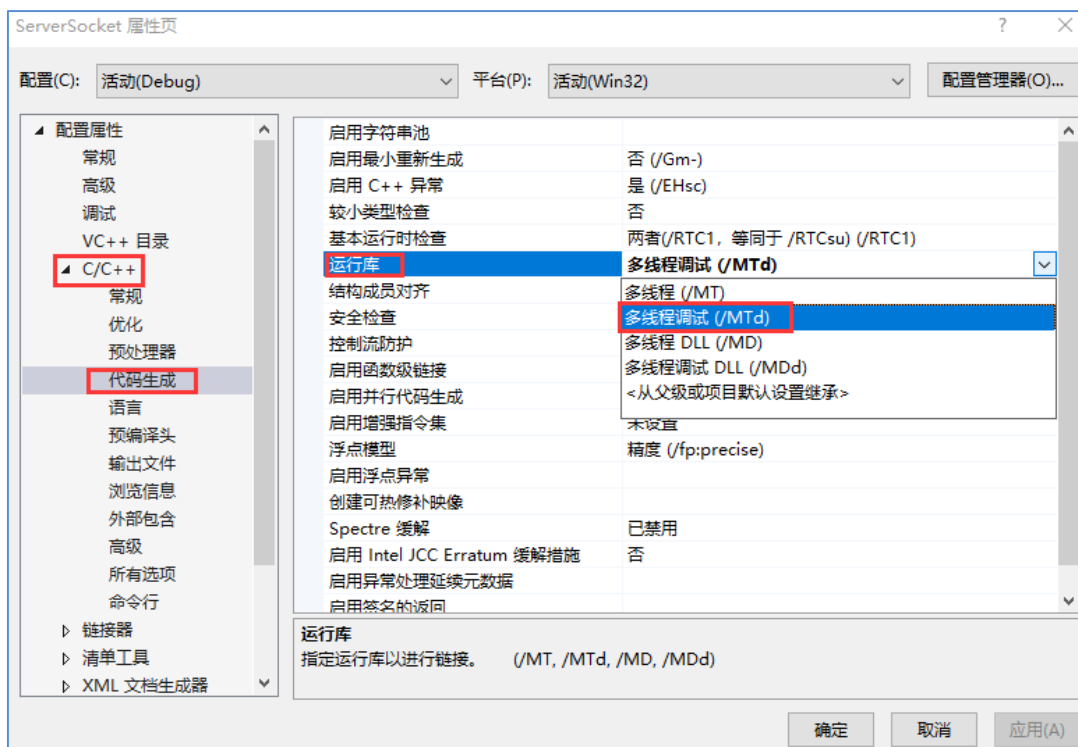


图3.19 设置运行库属性

(4) 调试工程项目：点击本地“Windows调试器”，如图3.20所示。

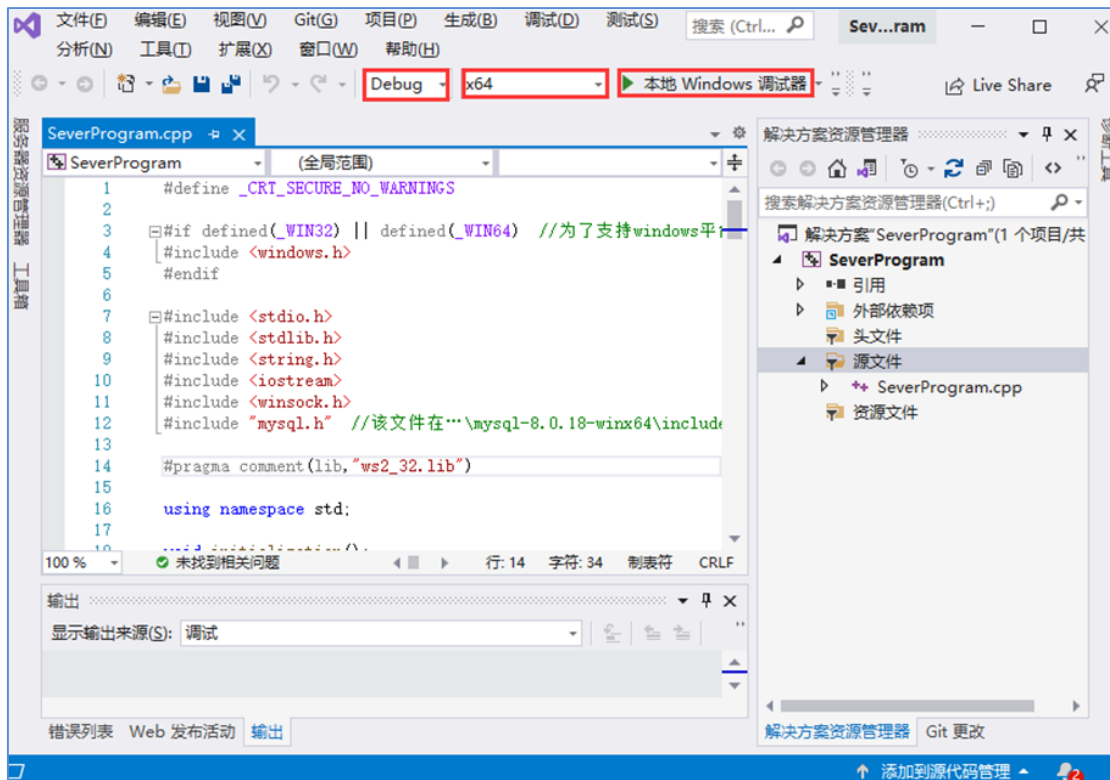


图3.20 调试工程项目

得到如图 3.21 所示调试结果。

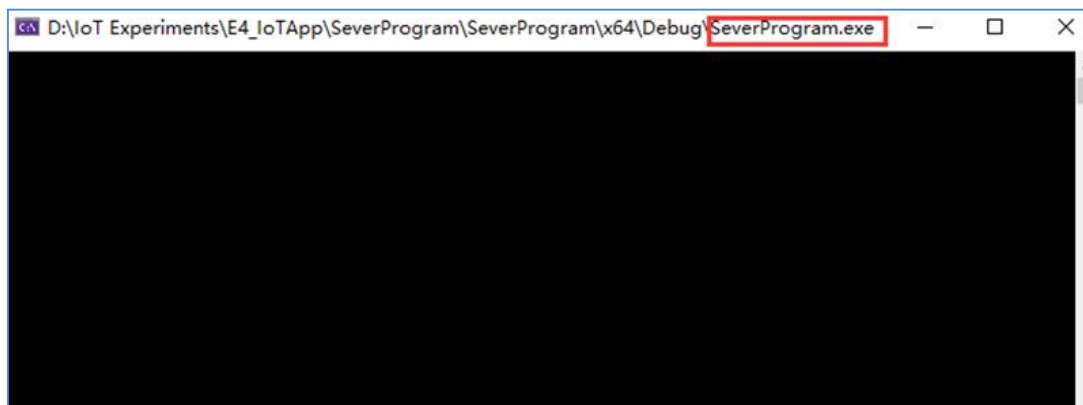


图3.21 工程项目运行结果

因为程序将部署在服务器后台运行，所以未输出任何数据，但可以开启终端节点，然后打开数据库，键入命令：“select * from acquisitiondata;”，查看数据存储结果，如图 3.22 所示，记录该实验结果（照相）。

numRecord	tempData	humiData	datetimeDA
1	30.06	95	2021-6-19 4:6:13
2	30.06	95	2021-6-19 4:6:15
3	30.06	95	2021-6-19 4:6:18
4	30.06	95	2021-6-19 4:6:20
5	30.06	95	2021-6-19 4:6:22
6	30.06	95	2021-6-19 4:6:25
7	30.13	95	2021-6-19 4:6:27
8	30.13	95	2021-6-19 4:6:29
9	30.13	95	2021-6-19 4:6:32
10	30.19	95	2021-6-19 4:6:34
11	30.19	95	2021-6-19 4:6:36
12	31.88	95	2021-6-19 4:58:3
13	31.88	95	2021-6-19 4:58:6

图3.22 查看数据库记录

（5）发布应用程序

在主界面把“Debug”改为“Release” 点击本地“Windows调试器”，生成发布应用程序。.dll文件是程序运行需要载入的动态链接库，VS中调试时可以通过“项目->属性->调试->环境”栏目添加.dll文件的path而成功调试，但在独立运行.exe程序是须将.dll文件放到同一目录下，如图3.23所示。

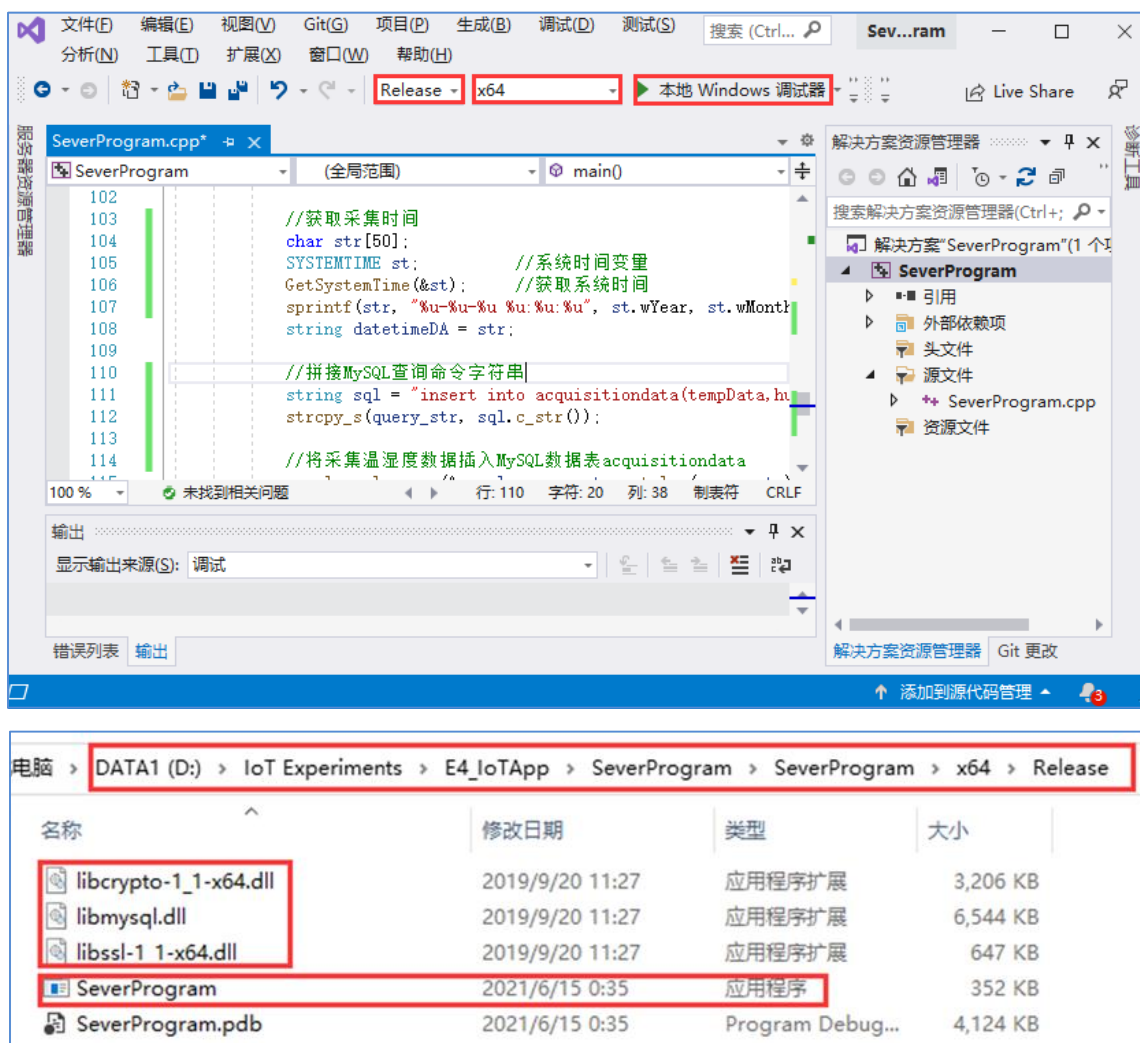


图3.23 生成发布程序

【思考与练习】

1. 如何将服务端程序部署在服务器，并在后台运行？请部署在云服务器中。
2. 客户端和服务端的自定义数据通信协议（数据格式）是如何设计的？
3. 温湿度采集数据一直增加，会导致数据库表的记录无限增长，会超过其存储空间吗？该如何解决？

【预习要求】

1. 参照相关资料，创建数据库和数据表，存储温湿度数据。
2. 改写源程序代码，按照自定义通信协议，实现数据的采集与上传。
3. 熟悉 Socket 服务端程序的编写方法，熟悉 VS2019 平台下 C++程序的设计方法。
4. 认真阅读本实验指导书，熟悉实验步骤和实验方法。

实验4 温湿度实时监测系统

【实验目的】

1. 熟悉 WiFi 模块 ESP8266 与 STM 单片机的通信接口，掌握其在程序设计中的管脚定义方法。
2. 掌握 ESP8266 的 Station 模式应用于物联网终端节点接入网的方法，掌握基于 ESP8266 的 AT 指令和 TCP 通信协议进行数据网络传输的编程方法。
3. 掌握以串口调试工具 SSCOM 进行 WiFi 模块调试的方法。

【实验设备】

- | | |
|------------------|-----|
| 1. 物联网实训工作站 | 1 台 |
| 2. 开发板（已焊接基础训练包） | 1 块 |
| 3. J-Link 下载器 | 1 个 |

【实验要求】

1. 根据所给原理图和 PCB，找到 WiFi 模块 ESP-12S，熟悉其与单片机的硬件接口。对照管脚图和管脚定义，通过工程模板提供的参考代码，掌握在程序设计中 ESP-12S 的管脚定义方法，以及与单片机连接的 STM32 管脚定义。
2. 套用实验 3 工程模板 E3_ProjectTemplate 新建工程，通过所提供相关器件手册和资料，熟悉 ESP8266 的工作原理。阅读 esp8266.c 和 esp8266.h 源代码，补充编写 WiFi 模块部分代码。
3. 编写 main 程序入口函数，每隔 T 秒（可以自己在程序中设定）采集一次温湿度数据，以 TCP 透传到服务器端，通过串口调试助手 SSCOM 观察所采集的温湿度值，完成 STM32 系统下的 WiFi 模块调试。

【实验原理】

1. 温湿度实时监测系统的系统架构

温湿度实时监测系统结构，如图4.1所示，其中图4.1（a）和图4.1（b）与实验3是一样的，都是把工作站电脑暂时当服务器使用。图4.1（c）把阿里云ECS作服务端，工作站电脑当用户终端，温湿度感知节点作客户端，是实际温湿度实时监测系统应该采用的

系统架构。

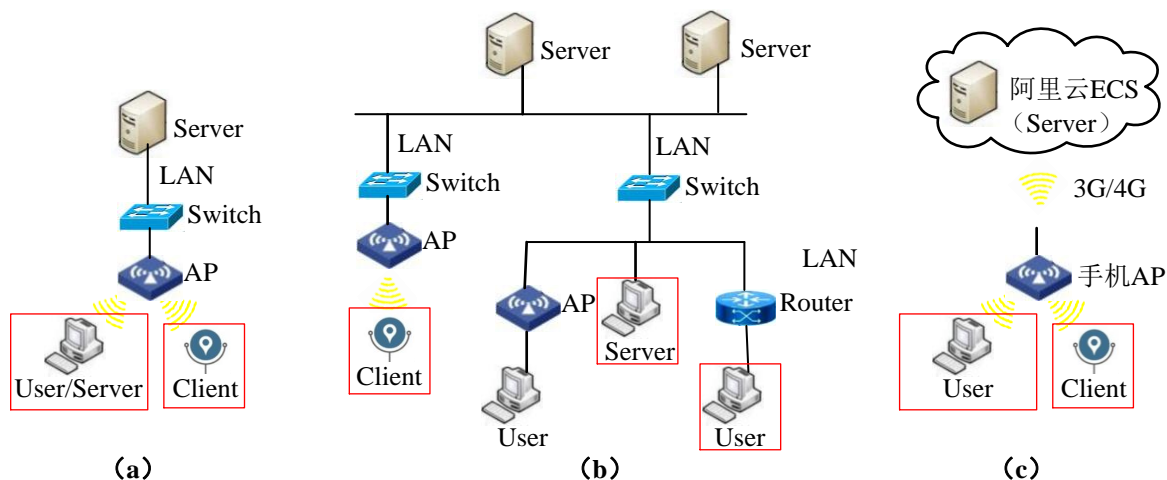


图4.1 温湿度监测系统架构

2. 系统工作原理

温湿度感知节点（Client），实时采集温湿度数据，然后通过WiFi模块，以TCP协议将采集数据实时无线传输给服务端（Server）；服务端将接收到的温湿度数据存于数据库中；用户终端（User）从服务端获取实时采集和历史数据，用以监测、分析、查询等应用。

3. 系统设计

用户应用程序界面如图4.2所示。

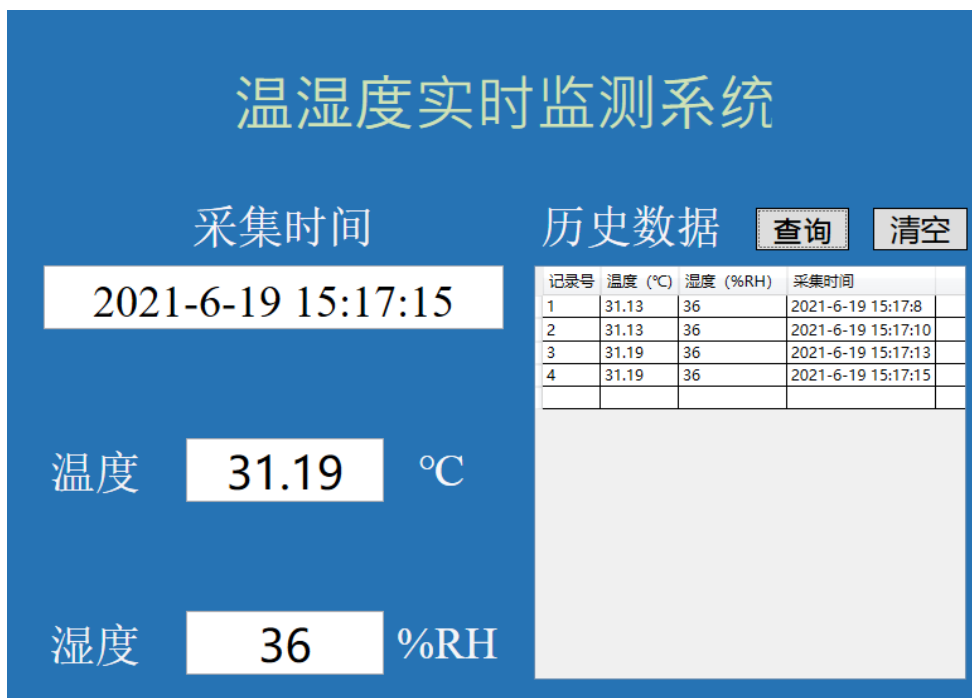


图4.2 用户应用程序界面

用户应用程序开发平台为Visual Studio 2019，采用C#编程实现（同学们也可以选择自己熟悉的编程语言和开发平台），需要从服务器的数据库获取实时采集数据和历史数据，进行显示和查询。

【实验步骤】

1. 客户端（Client）和服务端（Server）程序及数据库设计

在实验3中，通过客户端（Client）和服务端（Server）程序以及数据库的设计，已经实现了温湿度数据采集、传输与存储，本实验过程与之完全相同，不再赘述。

2. 编写用户应用程序

（1）创建UserApp工程

➤ 按照用户需求，需要实时监测温湿度数据和查询历史数据，所以用户程序需要建立WPF应用，如图4.3所示。



图4.3 创建UserApp工程

➤ 配置UserApp工程项目，如图4.4所示。

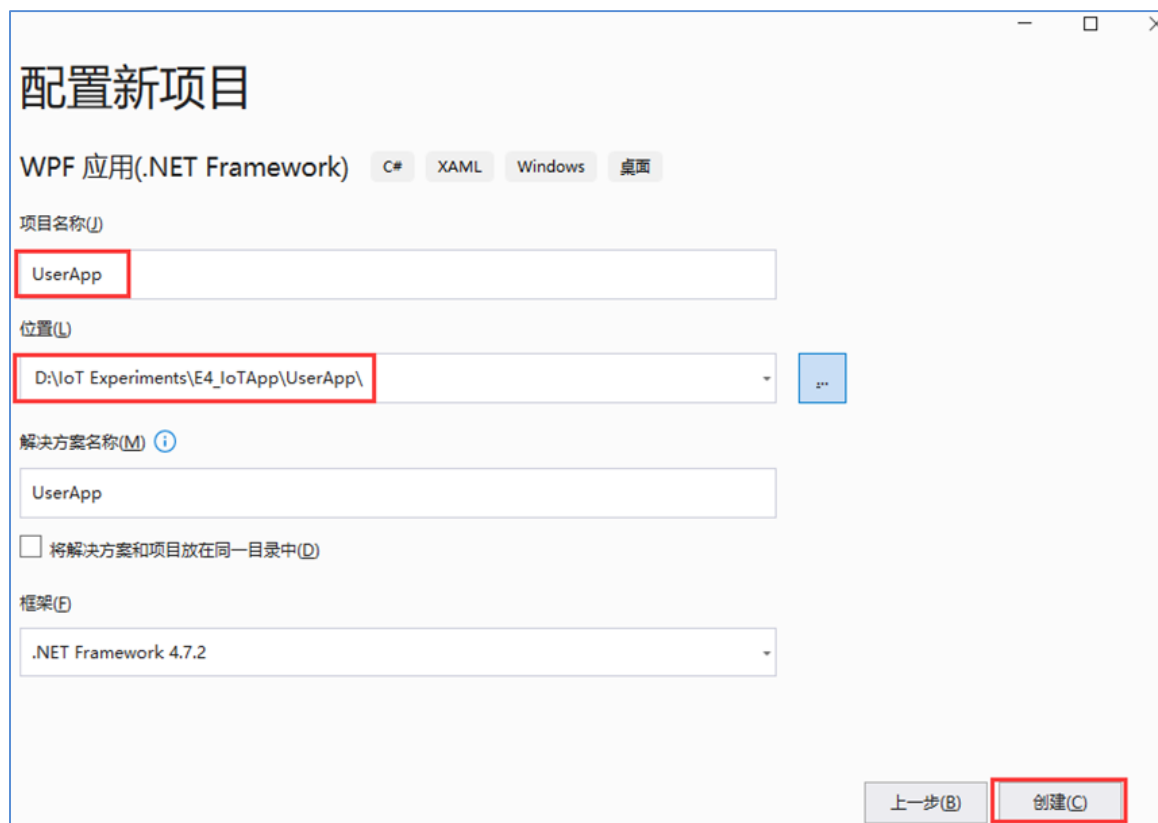


图4.4 配置UserApp工程

➤ 配置UserApp目标框架，如图4.5所示。

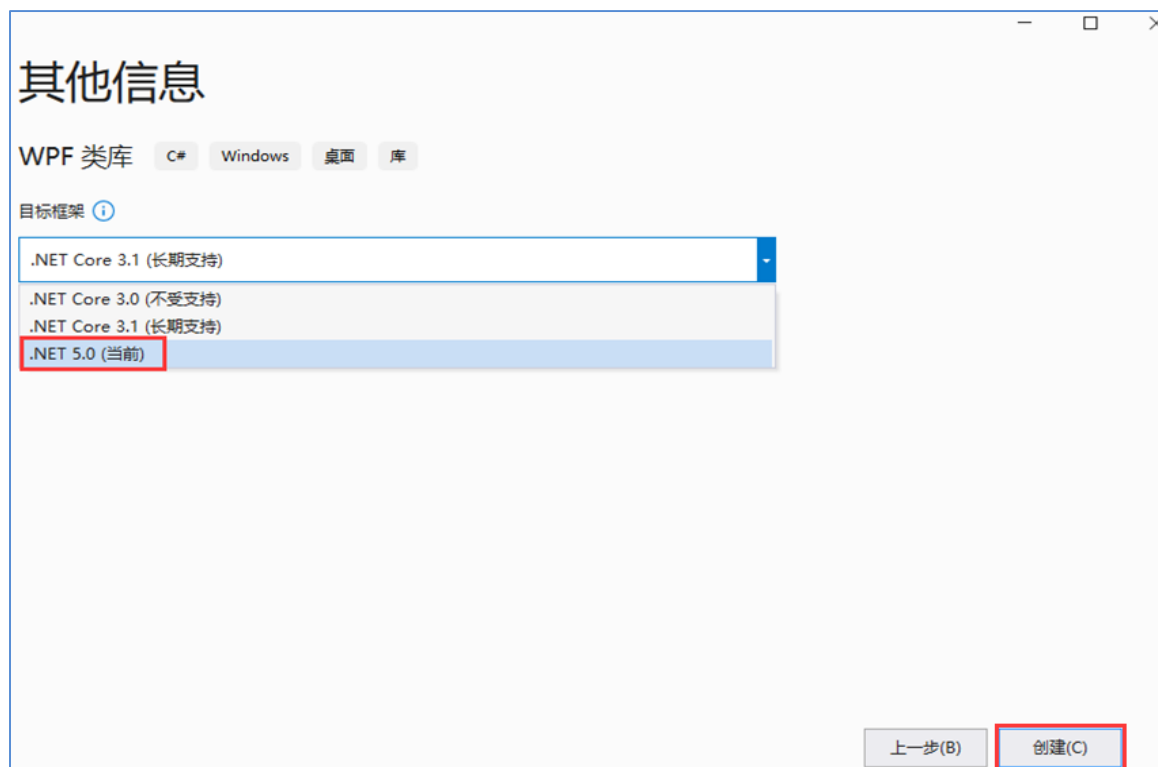


图4.5 配置UserApp目标框架

➤ 创建的工程项目如图4.6所示。

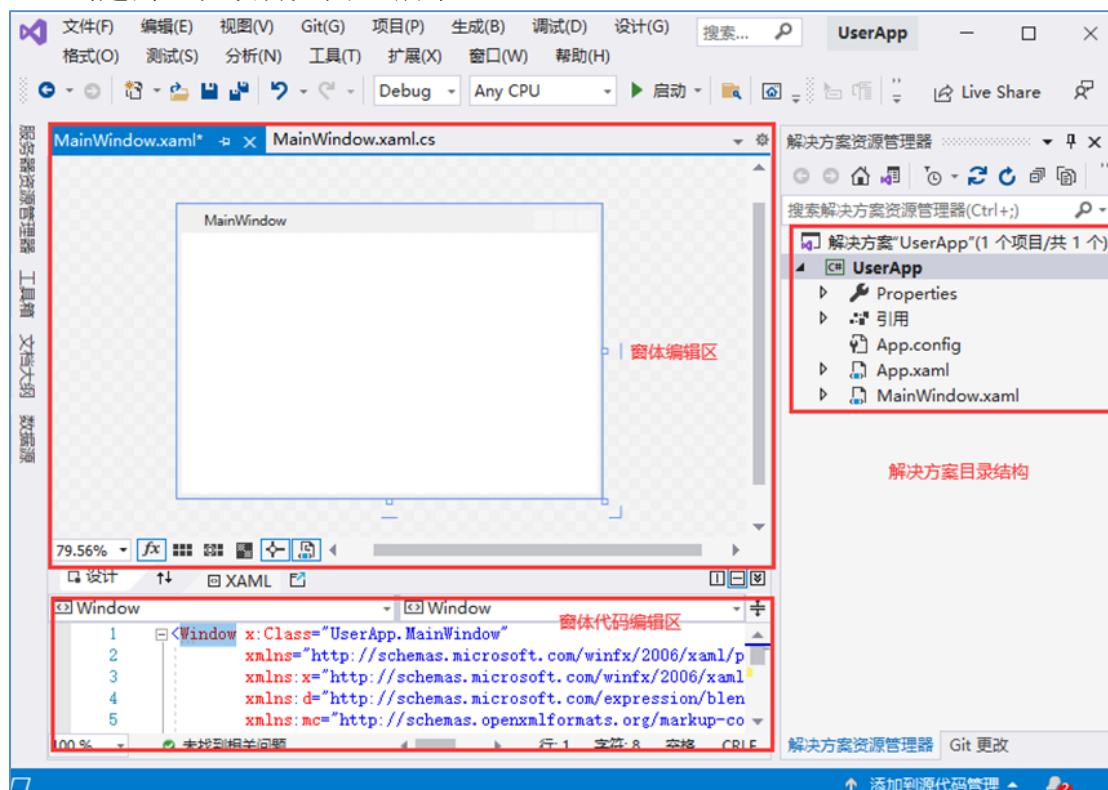


图4.6 UserApp工程编辑主界面

(2) UserApp界面设计

➤ 添加不同控件，设计如图4.7所示用户界面（界面风格可自行设计）。



图4.7 UserApp界面

- 其中显示历史数据的DataGrid要绑定数据表的各个字段，如图4.8所示。

```
<DataGrid x:Name="dataGridViewTH" HorizontalAlignment="Left" Height="331" Margin="412,205,0,0"
VerticalAlignment="Top" Width="343" AutoGenerateColumns="False">
    <DataGrid.Columns>
        <DataGridTextColumn Header="记录号" Binding="{Binding Path=numRecord}"/>
        <DataGridTextColumn Header="温度 (°C)" Binding="{Binding Path=tempData}"/>
        <DataGridTextColumn Header="湿度 (%RH)" Binding="{Binding Path=humiData}"/>
        <DataGridTextColumn Header="采集时间" Binding="{Binding Path=datetimeDA}"/>
    </DataGrid.Columns>
</DataGrid>
```

图4.8 DataGrid绑定数据表各字段

(3) UserApp程序设计

- UserApp程序需要从服务器获取温湿度实时采集数据和历史数据，并在用户界面显示实时采集数据和历史数据，其程序流程图如图4.9所示。

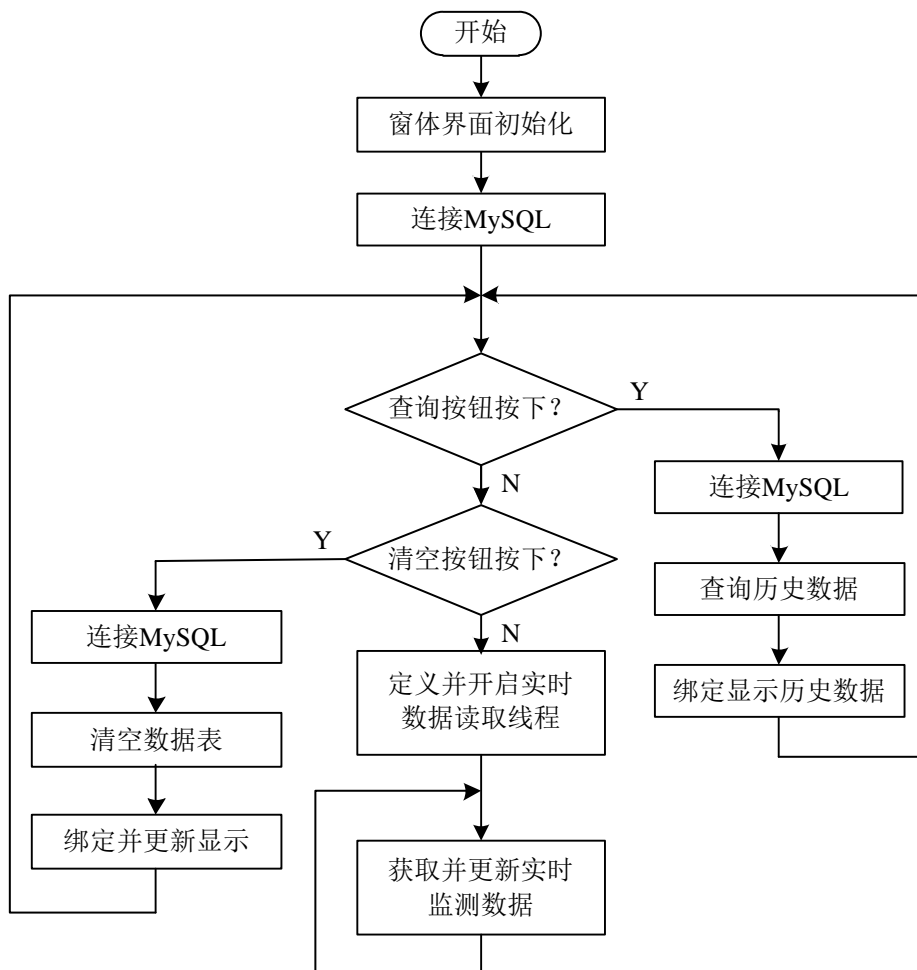


图4.9 用户程序main函数流程图

➤ UserApp程序代码如图4.10所示，其中涂掉的部分是需要根据实际情况改写的。

```
using System.Windows;
using MySql.Data.MySqlClient;
using System.Data;
using System.Linq;
using System.Threading;

namespace UserApp
{
    /// <summary>
    /// MainWindow.xaml 的交互逻辑
    /// </summary>
    public partial class MainWindow : Window
    {
        SynchronizationContext _syncContext = null;

        public MainWindow()
        {
            InitializeComponent(); //窗体界面定义初始化

            string connectionString = "server=;port=;database=;charset=utf8;user id=
            ;password=;pooling=false;"; //MySQL数据库连接字符串（本地服务器，server=localhost）
            MySqlConnection conn = new MySqlConnection(connectionString); //连接MySQL数据库
            conn.Open(); //打开连接
            DataTable dt = new DataTable(); //定义数据表
            string sqlstring = "SELECT * from acquisitiondata where numRecord = (SELECT max(numRecord) FROM
            acquisitiondata)"; //查询最后一条记录（实时数据）命令字符串
            MySqlCommand cmd = new MySqlCommand //定义MySQLCommand对象

            {
                Connection = conn,
                CommandText = sqlstring,
                CommandType = CommandType.Text
            };
            _syncContext = SynchronizationContext.Current; //获取UI线程同步上下文
            Thread LogThread = new Thread(new ThreadStart(DoService)); //定义线程
            LogThread.IsBackground = true; //设置线程为后台线程
            LogThread.Start(); //开启线程

            void DoService()
            {
                while (true)
                {
                    MySqlDataAdapter da = new MySqlDataAdapter(cmd); //获取最后一条记录
                    da.Fill(dt); //将实时数据填充到数据表dt
                    if (dt.Rows.Count == 0)
                    {
                        MessageBox.Show("尚未采集到数据！", "提示信息");
                        continue;
                    }
                    DataRow dr_last = dt.AsEnumerable().Last();
                    _syncContext.Post(SetTextBox1, dr_last[1].ToString()); //通过上下文更新textBoxTemp
                    _syncContext.Post(SetTextBox2, dr_last[2].ToString()); //通过上下文更新textBoxHumi
                    _syncContext.Post(SetTextBox3, dr_last[3].ToString()); //通过上下文更新textBoxTime
                    Thread.Sleep(1000);
                }
            }
        }
    }
}
```



```

void SetTextBox1(object text)
{
    txBoxTemp.Text = text.ToString();    //更新txBoxTemp
}
void SetTextBox2(object text)
{
    txBoxHumi.Text = text.ToString();    //更新txBoxHumi
}

void SetTextBox3(object text)
{
    txBoxTime.Text = text.ToString();    //更新txBoxTime
}

}

private void cxBtn_Click(object sender, RoutedEventArgs e)
{
    string connectionString = "server=localhost;port=3306;database=iotdatabase;charset=utf8;user
id=root;password=123456;pooling=false;";
    MySqlConnection conn = new MySqlConnection(connectionString);
    conn.Open();
    DataTable dt = new DataTable();

    string sqlstring = "SELECT * FROM acquisitiondata;";    //查询所有记录命令字符串
    MySqlCommand cmd = new MySqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = sqlstring;
    cmd.CommandType = CommandType.Text;
    MySqlDataAdapter da = new MySqlDataAdapter(cmd);
    da.Fill(dt);

    dataGridVTH.ItemsSource = dt.DefaultView;    //dataGridVTH绑定数据表dt

    //关闭数据库连接
    conn.Close();
}

private void btnClear_Click(object sender, RoutedEventArgs e)
{
    
}

}
}

```

图4.31 用户程序源代码

(4) 生成UserApp发布应用程序

➤ 按照图4.32所示方法，生成UserApp发布应用程序。

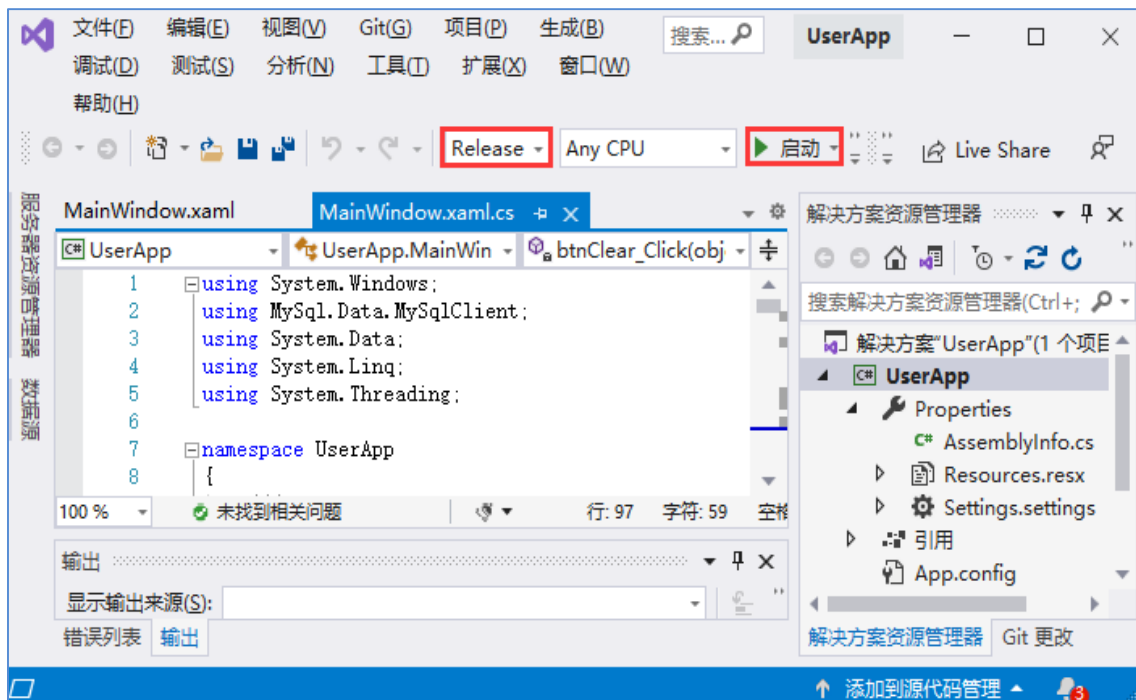


图4.32 生成UserApp发布应用程序

3. 系统联调

开启温湿度感知节点，打开服务端程序SeverProgram.exe，运行用户程序UserApp.exe，观察用户界面实时监测数据是否正常，“查询”、“清空”按钮功能是否正常。调试正常后，记录系统调试结果（照相）。

【思考与练习】

1. 用户程序怎样安装在另一台电脑中，才能够正常运行？请实践之。
2. 温湿度数据的实时监测是通过什么技术实现的？
3. DataGridView 控件如何绑定数据库中的数据？

【预习要求】

1. 熟悉 VS2019 环境下，创建 C# WPF 用户应用程序的方法。
2. 熟悉 VS2019 平台下 C#程序的设计方法，了解线程的编程方法。
3. 熟悉 VS2019 平台下 C#操作 MySQL 数据库的方法。
4. 认真阅读本实验指导书，熟悉实验步骤和实验方法。