【移动气象站】

——YAN 网络应用层模块

项目总结报告

成员：张健宁 徐博涵

日期：2021/06/26

目录

1 项目背景

移动气象站是用于对风向、风速、雨量、气温、相对湿度、气压、太阳辐射、土壤温度、土壤湿度等气象要素进行全天候现场监测的仪器。由气象传感器、气象数据采集仪、电源系统、轻型百叶箱、野外防护箱和不锈钢支架等部分构成。PH 风速风向等传感器为气象专用传感器，具有高精度高可靠性的特点气象数据采集仪具有气象数据采集、实时时钟、气象数据定时通过多种通讯方法(有线、数传电台、GPRS 移动通讯等)传输到数据库中，用于统计分析和处理的功能。被广泛应用于气象、环保、机场、农林、水文、军事、仓储、科学研究等领域。本小组负责实现的移动气象站便是 YAN 网络应用层中的一个具体应用服务。

本小组要实现的基于 YAN 网络的移动气象站可用于对风向、风速、雨量、温度、气压进行采集测量。一方面通过显示屏将气象要素数据显示出来，便于现场直接观测，减少通过电脑检测数据带来的不便。另一方面将获得的气象数据按照约定的协议封装成数据包通过 YAN 网络定时发送给 APRS 服务器，以便用户能够在手持机上不受距离限制地对气象信息进行实时监测查询。

2 项目配置

2.1 硬件

- 主控芯片：STM32F103C8

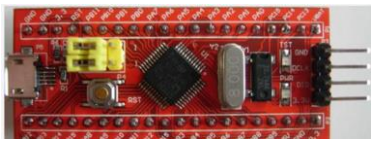

图 2- 1

- 风速/风向/雨量测量：BH4TDV51WS3 气象站

图 2- 2

- 温度/气压测量：BMP280 模块

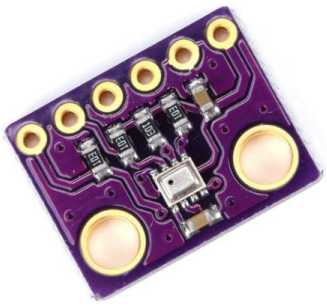

图 2- 3

- 数据显示：微雪 2.9 寸墨水屏

图 2- 4

- WiFi 模块：ESP-WROM-32

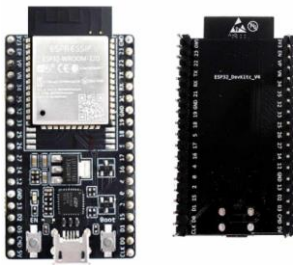

图 2- 5

2.2 软件

- 编译环境：STM32CubeIDE

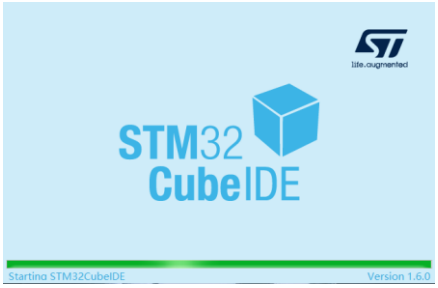

图 2- 6

- 串口调试助手：XCOM

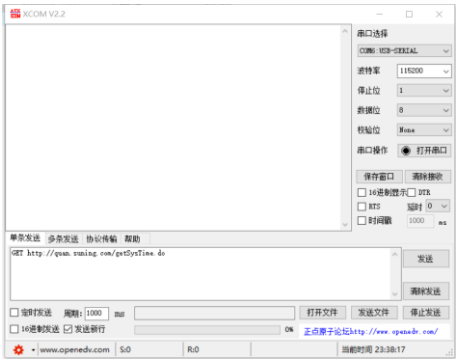

图 2- 7

- 网络调试助手：NetAssist

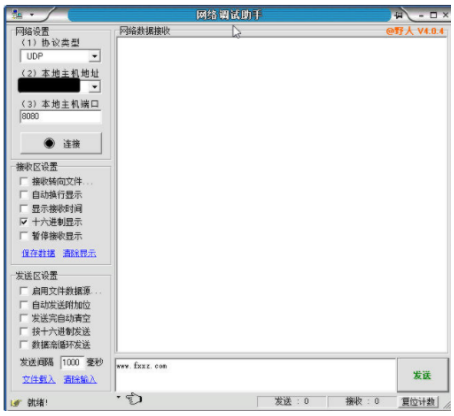

图 2- 8

3 项目进度总结

3.1 气象站元件测试

3.3.1 风速仪

根据器件手册的描述，风速仪是一个无源开关，扇叶每转动一圈，会触发两个脉冲(2Hz)。使用示波器测得的波形图印证了这个说明:

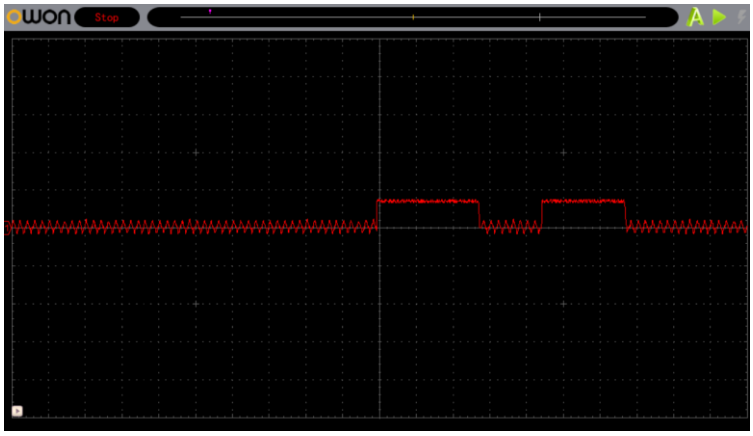

图 3- 1

3.3.1 风向仪

根据器件手册的描述，风向仪内部是一个电阻网络，相当于步进电位器。用户在使用的时候，需要外接一个上拉电阻分压。使用示波器测得的波形图印证了

这个说明:

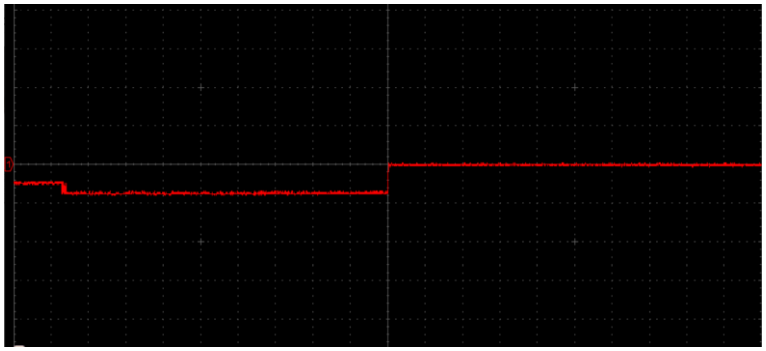

图 3- 2

3.3.1 雨量仪

雨量仪的内部结构如图所示，当雨量斗中的液体积攒到一定程度的时候，会因因重力的关系使雨量斗将液体倒出。

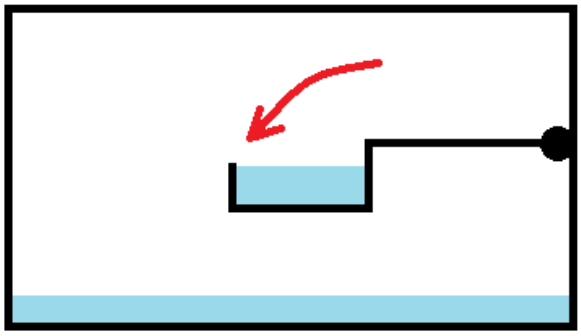

图 3- 3

根据器件手册的描述，雨量仪也是一个无源开关，雨量斗翻转一次会产生一个脉冲。使用示波器测得的波形图印证了这个说明:

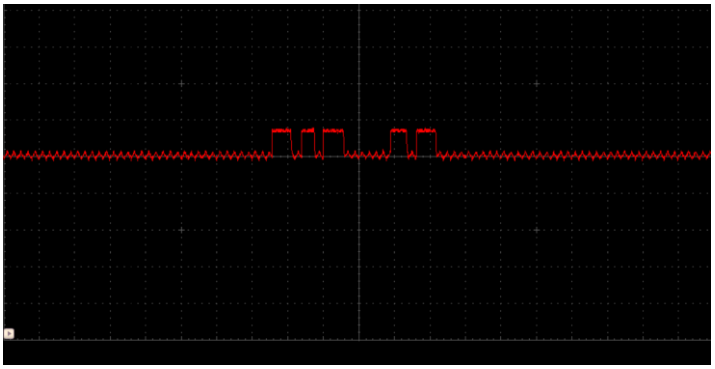

图 3- 4

3.2 气象站元件实现

3.2.1 器件线路原理

根据器件手册提供的气象套件二次开发接线参考图和器件实物。可以基本确定接线原理：

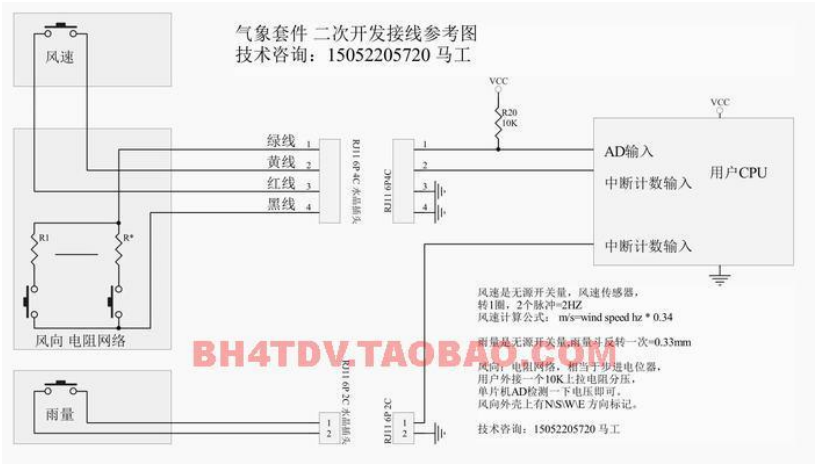

图 3- 5

风速仪通过水晶头接在风向仪上，由风向仪引出四条线(VCC、GND 和两条数据线)。雨量仪则是单独引出两条线(VCC 和一条数据线)。为了连接方便，将这六条线改装成了杜邦头。

图 3- 6

3.2.2 风速仪

根据器件手册的描述和示波器的波形印证，可以知道风速可以通过一定时间内触发脉冲数的一半乘上扇叶转动一圈运动的距离和时间相除得到。不过器件手册提供了风速的计算公式：

$$\mathbf{m/s = wind\ speed\ hz * 0.34}$$

观察公式可以发现，只需要计算触发的脉冲数即可计算出风速，而不需要通过风速仪扇叶转了多少圈。

因此设置引脚 PA2 为外部中断引脚，每当由低电平变高电平时，触发该外部中断，脉冲计数加 1。主要代码如下：

```
/**
  * @brief This function handles EXTI line2 interrupt.
  */
void EXTI2_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI2_IRQn 0 */
  wind_count ++;
  /* USER CODE END EXTI2_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_2);
  /* USER CODE BEGIN EXTI2_IRQn 1 */

  /* USER CODE END EXTI2_IRQn 1 */
}
```

图 3- 7

```
if(wind_count - last_wind_count >= 0.000001)
{
    speed = (wind_count - last_wind_count) * 0.34;
}
else
{
    speed = 0;
}
```

图 3- 8

由串口读取测试结果如下：

```
[22:55:54.793] Speed: 2.720000 m/s;
[22:55:54.809] Speed: 2.720000 m/s;
[22:55:55.809] Speed: 1.700000 m/s;
[22:55:56.809] Speed: 1.020000 m/s;
[22:55:57.809] Speed: 1.020000 m/s;
[22:55:58.810] Speed: 0.680000 m/s;
[22:55:59.809] Speed: 0.340000 m/s;
[22:56:00.808] Speed: 1.700000 m/s;
[22:56:01.808] Speed: 3.400000 m/s;
[22:56:02.809] Speed: 2.040000 m/s;
[22:56:03.809] Speed: 1.360000 m/s;
[22:56:04.808] Speed: 0.680000 m/s;
[22:56:05.809] Speed: 1.020000 m/s;
[22:56:06.808] Speed: 0.680000 m/s;
[22:56:07.808] Speed: 1.700000 m/s;
[22:56:08.808] Speed: 3.060000 m/s;
[22:56:09.808] Speed: 1.700000 m/s;
[22:56:10.808] Speed: 1.360000 m/s;
[22:56:11.808] Speed: 1.020000 m/s;
```

图 3- 9

3.2.3 风向仪

根据器件手册的描述和示波器的波形印证，可以知道测量风向实际是要开启 STM32F103C8 的 ADC 通道，测量出电压大小再结合仪器标注来确定方向。STM32CubeIDE 配置和主要代码如下：

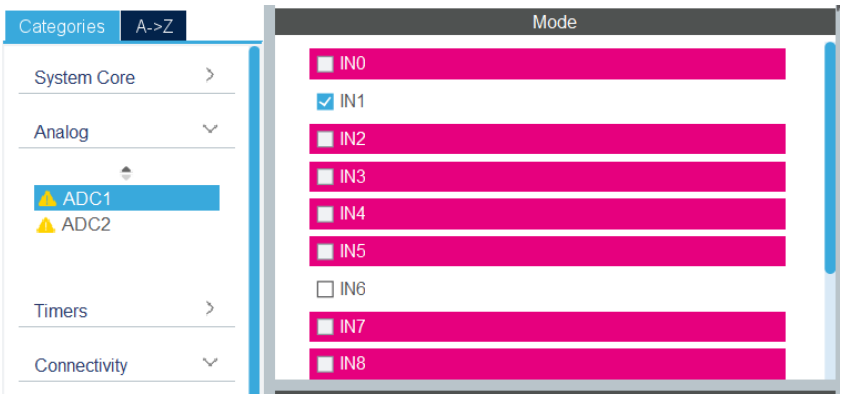

图 3- 10

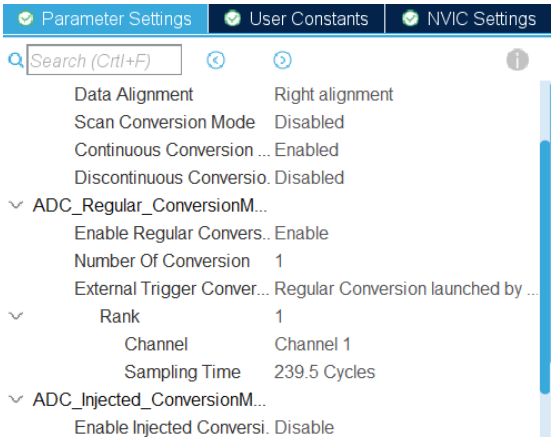

图 3- 11

```
HAL_ADC_Start_DMA(&hadc1, &ADC_Value, 100);
for(i = 0; i < 100; i++)
    ad1 += ADC_Value[i];
int tmp = (int)ad1/100;
switch(tmp)
{
case 402:case 401:dir = "W"; break;
case 199:case 400:dir = "WN"; break;
case 398:dir = "N"; break;
case 369:case 368:dir = "S"; break;
case 295:case 296:case 297:case 298:dir = "E"; break;
case 385:dir = "EN"; break;
case 394:dir = "WS"; break;
case 345:case 346:dir = "ES"; break;
};
```

图 3- 12

3.2.4 雨量仪

根据器件手册的描述和示波器的波形印证，可以知道测量雨量的原理实际和测量风速差不多。都是通过脉冲数和器件提供的公式进行计算。

$$mm = 0.33mm * hz$$

所以设置引脚 PA3 为外部中断引脚，每当由低电平变高电平时，触发该外部中断，脉冲计数加 1。只不过对数据处理方式会不同。雨量的量是叠加的。风速则是要不断初始化更新。主要代码如下：

```
void EXTI3_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI3_IRQn 0 */
    rain_count ++;
  /* USER CODE END EXTI3_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_3);
  /* USER CODE BEGIN EXTI3_IRQn 1 */

  /* USER CODE END EXTI3_IRQn 1 */
}
```

图 3- 13

由串口读取测试结果如下：

```
[22:56:46.510] Rain: 11.880000 mm;
[22:56:46.526] Rain: 11.880000 mm;
[22:56:46.806] Rain: 12.870000 mm;
[22:56:47.807] Rain: 13.530000 mm;
[22:56:48.806] Rain: 21.120000 mm;
[22:56:49.806] Rain: 30.690000 mm;
[22:56:50.805] Rain: 41.580000 mm;
[22:56:51.806] Rain: 47.520000 mm;
[22:56:52.805] Rain: 50.160000 mm;
[22:56:53.806] Rain: 82.830000 mm;
[22:56:54.805] Rain: 89.100000 mm;
[22:56:55.805] Rain: 92.400000 mm;
[22:56:56.805] Rain: 94.380000 mm;
[22:56:57.806] Rain: 97.020000 mm;
[22:56:58.805] Rain: 99.000000 mm;
[22:56:59.806] Rain: 99.660000 mm;
[22:57:00.806] Rain: 100.980000 mm;
[22:57:01.805] Rain: 100.980000 mm;
[22:57:02.806] Rain: 100.980000 mm;
```

图 3- 14

3.3 BMP280

BMP280 是一种专为移动应用设计的绝对气压传感器。该传感器模块采用极其紧凑的封装。得益于小尺寸和低功耗特性，这种器件可用在如移动电话、GPS 模块或手表等电池供电型设备中。

3.3.1 原理图

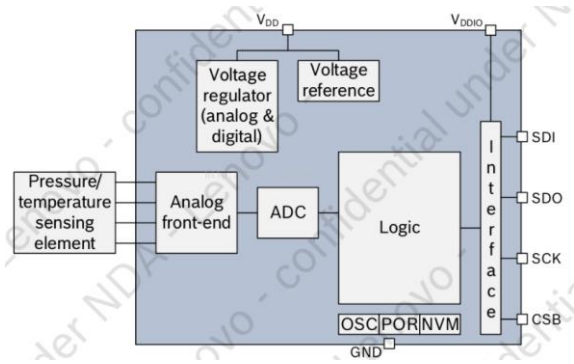

图 3- 15

3.3.2 通信模式

BMP280 支持 I2C 和 SPI 两种通信模式。根据片选(CSB)引脚决定，如果 CSB 拉低，则采用 SPI 模式。注意，一旦 CSB 引脚被拉低后，SPI 模式会一直开启，直到下一次重启硬件。项目中采用的通信模式是 I2C 通讯模式，引脚对应如下：

- SCL --> 时钟线
- SDI --> 数据线
- SDO --> 片地址的 LSB(最低加权位)
- CSB --> 拉高

要往 BMP280 中写数据，则需主机 STM32F103C8 向 BMP280 发送 0xEC，然后告知要写入的地址和数据。要从 BMP280 读取数据，则需要主机 STM32F103C8 先向主机发送 0xEC 和要读取的寄存器的地址，才能等待接收数据。

主要代码如下：

```c
uint8_t bmp280_Read(uint8_t addr)
{
    uint8_t ret = 255, isok = 0;
    HAL_I2C_Master_Transmit(&hi2c1, BMP280_ADDR, &addr, 1, 10);
    isok = HAL_I2C_Master_Receive(&hi2c1, BMP280_ADDR, &ret, 1, 10);
    if (isok != 0)
        return HAL_I2C_GetError(&hi2c1);
    return ret;
}

void bmp280_Write(uint8_t addr, uint8_t dat)
{
    uint8_t mydat[2] = {addr, dat};
    HAL_I2C_Master_Transmit(&hi2c1, BMP280_ADDR, mydat, 2, 10);
}
```

图 3- 16

BMP280 的测量流程主要是这样的：

- 将 BMP280 复位。
- 设置采集的数据精度、采样时间等配置。
- 读取补偿寄存器中的补偿值。
- 延时一小段时间，等待数据采集完毕放入寄存器中。
- 主机读取从机寄存器中的采集数据，然后根据器件手册提供的公式和读取到的补偿数据进行计算，得到温度和气压数据。

主要代码如下：

```c
uint8_t lsb = 0, msb = 0;
//复位BMP280
bmp280_Write(BMP280_RESET_REG, BMP280_RESET_VALUE);
HAL_Delay(1);
//获取CHIP IP
chipid = bmp280_Read(BMP280_CHIPID_REG);
//设置数据精度
bmp280_Write(BMP280_CTRLMEAS_REG, 0xff);
//设置采样时间
bmp280_Write(BMP280_CONFIG_REG, 0x20);

//读取补偿值 dig_XX
lsb = bmp280_Read(BMP280_DIG_T1_LSB_REG);
msb = bmp280_Read(BMP280_DIG_T1_MSB_REG);
dig_T1 = msb << 8 | lsb;
```

图 3- 17

```c
double var1 = 0, var2 = 0;
msb = bmp280_Read(BMP280_TEMPERATURE_MSB_REG);
lsb = bmp280_Read(BMP280_TEMPERATURE_LSB_REG);
xlsb = bmp280_Read(BMP280_TEMPERATURE_XLSB_REG);
adc_T = (msb << 12) | (lsb << 4) | (xlsb >> 4);
var1 = (((double)adc_T) / 16384.0 - ((double)dig_T1) / 1024.0) * ((double)dig_T2);
var2 = ((((double)adc_T) / 131072.0 - ((double)dig_T1) / 8192.0) * (((double)adc_T) / 131072.0 - ((double)dig_T1) / 8192.0))
t_fine = var1 + var2;
bmp_280_temperature = (var1 + var2) / 5120.0;
```

图 3- 18

```
msb = bmp280_Read(BMP280_PRESSURE_MSB_REG);
lsb = bmp280_Read(BMP280_PRESSURE_LSB_REG);
xlsb = bmp280_Read(BMP280_PRESSURE_XLSB_REG);
adc_P = (msb << 12) | (lsb << 4) | (xlsb >> 4);

var1 = ((double)t_fine / 2.0) - 64000.0;
var2 = var1 * var1 * ((double)dig_P6) / 32768.0;
var2 = var2 + var1 * ((double)dig_P5) * 2.0;
var2 = (var2 / 4.0) + (((double)dig_P4) * 65536.0);
var1 = (((double)dig_P3) * var1 * var1 / 524288.0 + ((double)dig_P2) * var1) / 524288.0;
var1 = (1.0 + var1 / 32768.0) * ((double)dig_P1);

if (var1 == 0.0)
{
    bmp_280_atmospressure = 0; // avoid exception caused by division by zero
    return;
}

pressure = 1048576.0 - (double)adc_P;
pressure = (pressure - (var2 / 4096.0)) * 6250.0 / var1;
var1 = ((double)dig_P9) * pressure * pressure / 2147483648.0;
var2 = pressure * ((double)dig_P8) / 32768.0;
pressure = pressure + (var1 + var2 + ((double)dig_P7)) / 16.0;
bmp_280_atmospressure = pressure;
```

图 3- 19

3.4 微雪墨水屏

墨水屏与常用的液晶显示屏等其他显示器件相比，其最主要的优势在于能够以极低的能耗在长时间下工作，刷新出文字后即使断电也能一直保持显示内容。

实现上采用画布绘制的方式，先建立一个二进制数组表示画布信息，将各个字母的图像以常量保存在程序中，绘制时用字母图像的二进制黑白信息替换画布的相应位置的图像。完成绘制后刷新图像时将整个二进制的图像数组传入墨水屏模块，以此来完成图像的显示。

在通信上，微雪墨水屏采用 spi 通信，并用片选信号来区分数据和指令，CS=0 时为数据信号，CS=1 时为指令信号。显示图像时先向墨水屏发送 0x26 指令，然后逐位发送数组中的二进制数据，最后依次发送 0x22 指令、0xF7 数据、0x20 指令命令墨水屏刷新

3.4.1 相关配置

Mode: Transmit Only Master

5.2.1. Parameter Settings:

Basic Parameters:

Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First

Clock Parameters:

Prescaler (for Baud Rate)	**64 ***
Baud Rate	**1.125 MBits/s ***
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge

Advanced Parameters:

CRC Calculation	Disabled
NSS Signal Type	Software

3.4.2 主要代码

初始化部分，闪两次然后显示一下微雪的图标，以确认正常工作

```
int EPD_init(void)
{
    MX_GPIO_Init();
    MX_USART1_UART_Init();
    MX_SPI1_Init();
    if(DEV_Module_Init()!=0){
        return -1;
    }

    printf("e-Paper Init and Clear...\r\n");
    EPD_2IN9_V2_Init();
    EPD_2IN9_V2_Clear();

    //Create a new image cache
    UWORD Imagesize = ((EPD_2IN9_V2_WIDTH % 8 == 0)? (EPD_2IN9_V2_WIDTH / 8 ): (EPD_2IN9_V2_WIDTH / 8 + 1)) * EPD_2IN9_V2_HEIGHT;
    if((BlackImage = (UBYTE *)malloc(Imagesize)) == NULL) {
        printf("Failed to apply for black memory...\r\n");
        return -1;
    }
    Paint_NewImage(BlackImage, EPD_2IN9_V2_WIDTH, EPD_2IN9_V2_HEIGHT, 90, WHITE);
    Paint_Clear(WHITE);

    Paint_NewImage(BlackImage, EPD_2IN9_V2_WIDTH, EPD_2IN9_V2_HEIGHT, 90, WHITE);
    Paint_SelectImage(BlackImage);
    Paint_Clear(WHITE);
    Paint_DrawBitMap(gImage_2in9);

    EPD_2IN9_V2_Display(BlackImage);
    DEV_Delay_ms(3000);
    Paint_NewImage(BlackImage, EPD_2IN9_V2_WIDTH, EPD_2IN9_V2_HEIGHT, 90, WHITE);
    printf("Drawing\r\n");
    return 0;
}
```

顶层显示函数，转换到字符串形式后逐字符输出

```
int EPD_write(float wind_speed,char* wind_direction, float rain_fall, float temp, float humi, float air_pressure)
{
    char tem[50];
    //Select Image
    Paint_SelectImage(BlackImage);
    Paint_Clear(WHITE);

    // Drawing on the image

    Paint_DrawString_EN(0, 0,"windspeed(m/s):", &Font20, WHITE, BLACK);
    sprintf(tem,"%.2f",wind_speed);
    Paint_DrawString_EN(205, 0,tem, &Font20, WHITE, BLACK);

    Paint_DrawString_EN(0, 20, "wind direction:", &Font20, WHITE, BLACK);
    Paint_DrawString_EN(210, 20, wind_direction, &Font20, WHITE, BLACK);

    Paint_DrawString_EN(0, 40,"rainfall(mm):", &Font20, WHITE, BLACK);
    sprintf(tem,"%.2f",rain_fall);
    Paint_DrawString_EN(180, 40,tem, &Font20, WHITE, BLACK);

    Paint_DrawString_EN(0, 60, "temperature(c):", &Font20, WHITE, BLACK);
    sprintf(tem,"%.2f",temp);
    Paint_DrawString_EN(210, 60,tem, &Font20, WHITE, BLACK);

    Paint_DrawString_EN(0, 80,"humidity:", &Font20, WHITE, BLACK);
    sprintf(tem,"%.2f",humi);
    Paint_DrawString_EN(120, 80,tem, &Font20, WHITE, BLACK);

    Paint_DrawString_EN(0, 100, "air pres(MPa):", &Font20, WHITE, BLACK);
    sprintf(tem,"%.2f",air_pressure);
    Paint_DrawString_EN(200, 100,tem, &Font20, WHITE, BLACK);
```

逐像素复制字符信息到画布上

```
uint32_t Char_Offset = (Acsii_Char - ' ') * Font->Height * (Font->Width / 8 + (Font->Width % 8 ? 1 : 0));
const unsigned char *ptr = &Font->table[Char_Offset];

for (Page = 0; Page < Font->Height; Page ++ ) {
    for (Column = 0; Column < Font->Width; Column ++ ) {

        if (FONT_BACKGROUND == Color_Background) {
            if (*ptr & (0x80 >> (Column % 8)))
                Paint_SetPixel(Xpoint + Column, Ypoint + Page, Color_Foreground);
        } else {
            if (*ptr & (0x80 >> (Column % 8))) {
                Paint_SetPixel(Xpoint + Column, Ypoint + Page, Color_Foreground);
            } else {
                Paint_SetPixel(Xpoint + Column, Ypoint + Page, Color_Background);
            }
        }
        if (Column % 8 == 7)
            ptr++;
    }
    if (Font->Width % 8 != 0)
        ptr++;
```

单个像素修改数组信息

```
    uint32_t Addr = X / 8 + Y * Paint.WidthByte;
    UBYTE Rdata = Paint.Image[Addr];
    if(Color == BLACK)
        Paint.Image[Addr] = Rdata & ~(0x80 >> (X % 8));
    else
        Paint.Image[Addr] = Rdata | (0x80 >> (X % 8));
```

显示数据

```
EPD_2IN9_V2_SendCommand(0x26);   //Write Black and White image to RAM
for(i=0;i<4736;i++)
{
    EPD_2IN9_V2_SendData(Image[i]);
}
EPD_2IN9_V2_TurnOnDisplay();
```

3.5 WiFi 模块——ESP-WROOM-32

3.5.1 为 ESP-WROOM-32 烧写 AT 固件

在 ESPRESSIF 官网下载 ESP32-AT 固件和烧写工具 flash_download_tool。连接好 ESP-WROOM-32，擦除已有固件后。根据提供的烧写配置文件 download.config 烧写 AT 固件。

```
--flash_mode dio --flash_freq 40m --flash_size 4MB
0x8000 partition_table/partition-table.bin
0x10000 ota_data_initial.bin
0xf000 phy_init_data.bin
0x1000 bootloader/bootloader.bin
0x100000 esp-at.bin
0x20000 at_customize.bin
0x24000 customized_partitions/server_cert.bin
0x39000 customized_partitions/mqtt_key.bin
0x26000 customized_partitions/server_key.bin
0x28000 customized_partitions/server_ca.bin
0x2e000 customized_partitions/client_ca.bin
0x30000 customized_partitions/factory_param.bin
0x21000 customized_partitions/ble_data.bin
0x3B000 customized_partitions/mqtt_ca.bin
0x37000 customized_partitions/mqtt_cert.bin
0x2a000 customized_partitions/client_cert.bin
0x2c000 customized_partitions/client_key.bin
```

图 3- 20

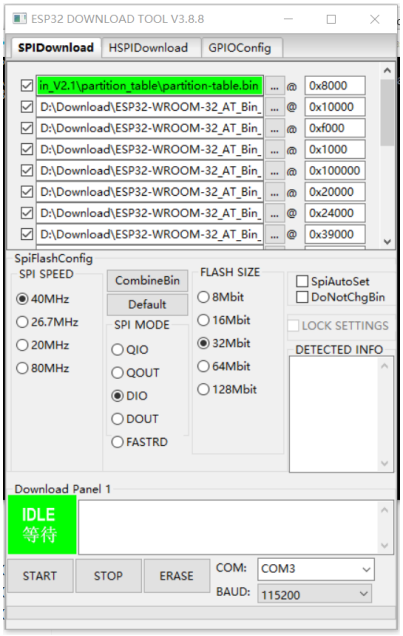

图 3- 21

3.5.2 AT 指令调试 ESP-WROOM-32 连接本地模拟的 TCP 服务器

打开手机热点作为 WIFI，运行 AT 指令让 ESP-WROOM-32 和 PC 都连接该热点。使用 NetAssist 设置一个本地模拟的 TCP 服务器，运行 AT 指令让 ESP-WROOM-32 连接 TCP 服务器，并尝试发布信息。运行指令和结果如下：

```
AT+RST
AT+CWMODE=1      //Station 模式
AT+CWLAP         //查看AP热点
AT+CWJAP="niu01","987654321"     //输入AP名称和密码
AT+CIPSTART="TCP","192.168.0.6",12345    //连接TCP服务器
AT+CIPSEND=11        //发送数据，11个字符
NBC Client!          //发送11个字符到TCP服务器
```

图 3- 22

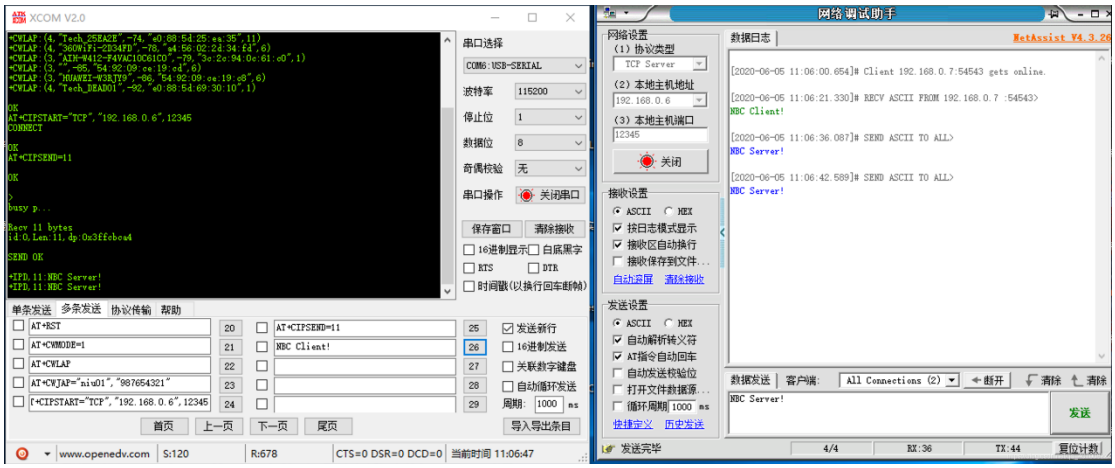

图 3- 23

3.5.3 AT 指令调试 ESP-WROOM-32 获得时间

由于 APRS 服务器的天气数据包要求有关时间的信息，因此尝试使用 AT 指令 GET 让 ESP32-WROOM-32 从公开的时间 API 中拉取时间数据。

时间 API 采用的是 http://quan.suning.com/getSysTime.do，返回数据如下：

```
{"sysTime2":"2021-06-23 09:19:04","sysTime1":"20210623091904"}
```

3.5.4 STM32F103C8 和 ESP-WROOM-32 的主从机整合

将 STM32F103C8 作为主机，ESP-WROOM-32 作为从机，由主机通过 UART 发送 AT 指令来控制从机。具体代码如下：

```
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)"AT+RST\r\n", 8);
HAL_Delay(3);
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)"AT+CWMODE=1\r\n", 13);
HAL_Delay(3);
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)"AT+CWJAP=\"zjn\",\"13536712286\"\r\n", 31);
HAL_Delay(3);
```

图 3- 24

```
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)"AT+CIPSTART=\"TCP\",\"192.168.43.47\",12345\r\n", 41);
HAL_Delay(100);
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)"AT+CWMODE=1\r\n", 13);
HAL_Delay(100);
buffer_length = sprintf(buffer, "AT+CIPSEND=%d\r\n", length);
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)buffer, buffer_length);
HAL_Delay(100);
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)aRxBuffer, length);
HAL_Delay(100);
buffer_length = sprintf(buffer, "AT+CIPSEND=%d\r\n", buf_length);
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)buffer, buffer_length);
HAL_Delay(100);
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)buf, buf_length);
HAL_Delay(100);
HAL_UART_Transmit_DMA(&huart1, (uint8_t*)"AT+CIPCLOSE\r\n", 13);
HAL_Delay(100);
```

图 3- 25

4 项目整合成果

4.1 连接原理图

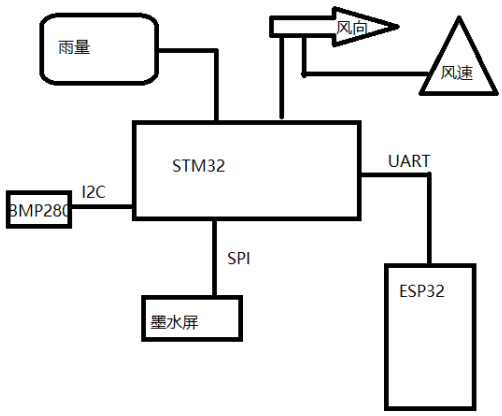

图 4 1

如上图所示，以 STM32F103C8 作为主控芯片，通过 I2C 的方式连接 BMP280，

SPI 的方式连接墨水屏，通过 UART 连接 ESP-WROOM-32。

4.2 实物图

根据连线原理图连接各个模块，组成实物图如下：

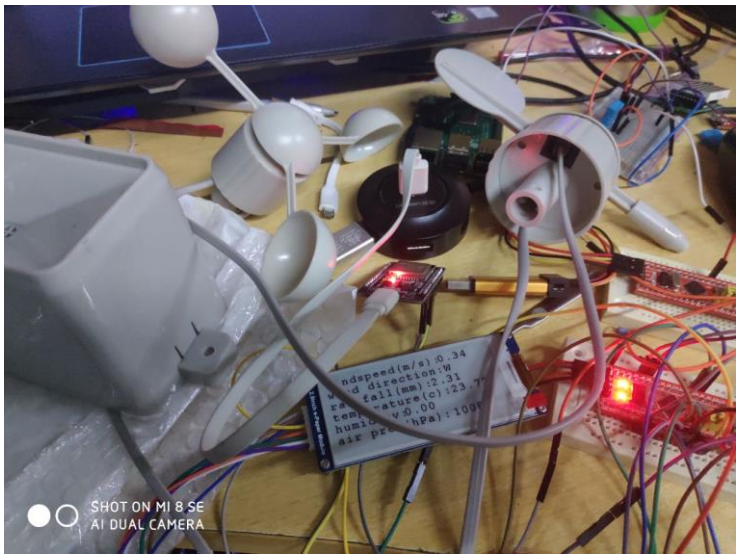

图 4 2

4.3 整合思路

4.3.1 引脚配置

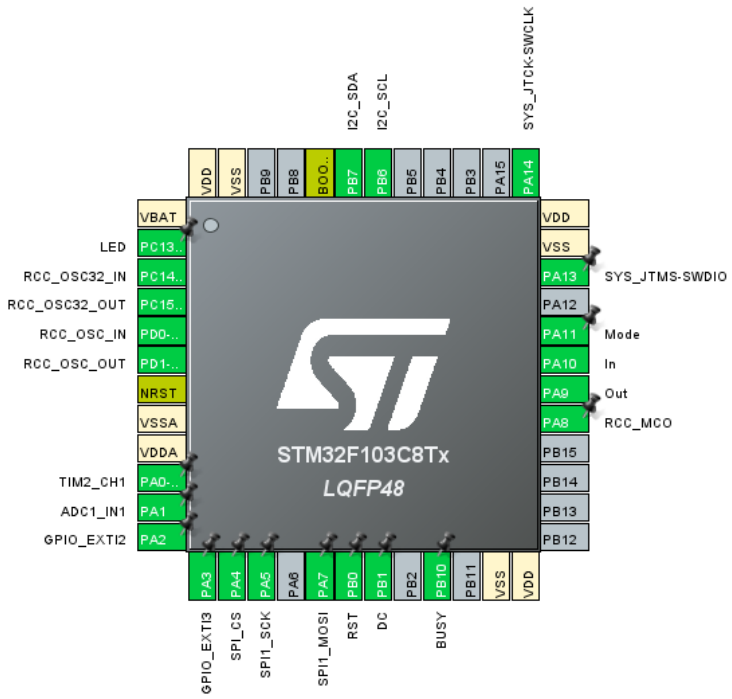

图 4 3

引脚配置如上图：PA1 引脚作为 ADC 通道用于读取风向。PA2 引脚作为外部中断引脚用于读取风速。PA3 引脚作为外部中断引脚用于读取雨量。PA0、PA4、PA5、PA7、PB0、PB1、PB10 引脚用于 SPI 连接墨水屏进行显示。PB7、PB8 引脚用于 I2C 连接 BMP280 读取温度和气压数据。PA9、PA10 引脚用于 UART 通信控制 ESP-WROOM-32 连接网络。

4.3.2 代码结构

- stm32f1xx_it.c 中主要是进行两个引脚外部中断的处理，即对风速和雨量的脉冲进行计数。
- bmp280.c 中主要实现了对 BMP280 温度和气压数据的读取。

- main.c 中主要是 main 函数，用于初始化和读取气象数据进行显示。

4.4 整合成果

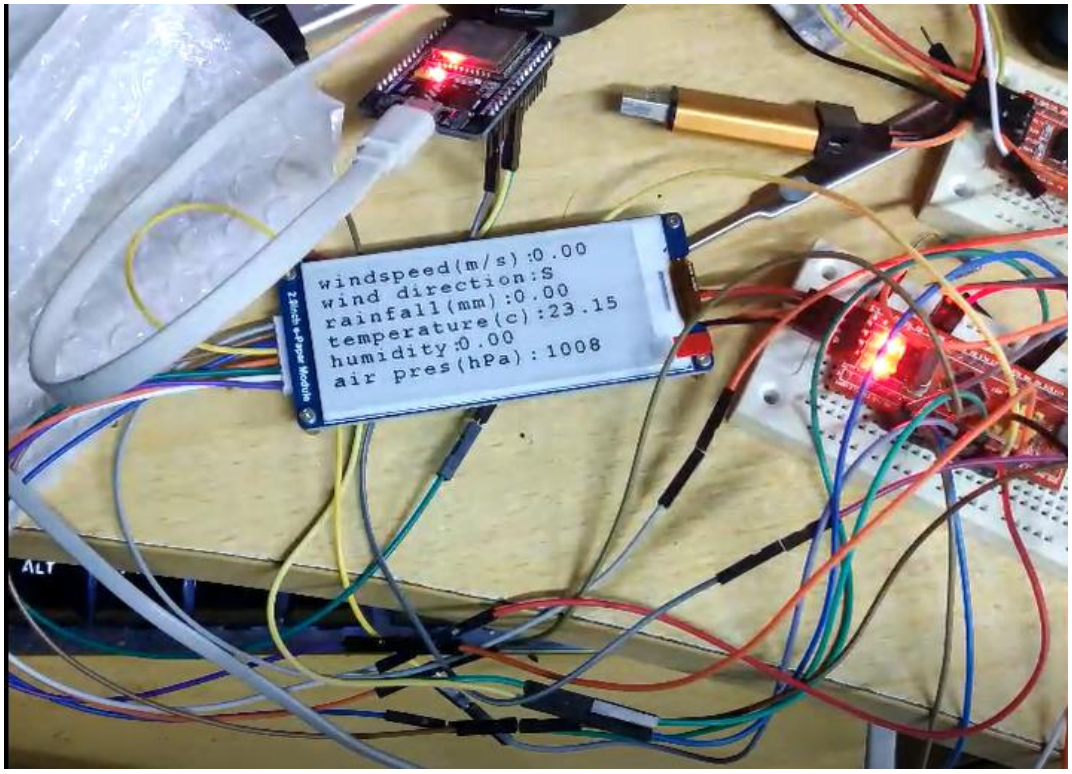

图 4 4

"sysTime1":"20210623023022"

temperature:23.25 'C; QNH :1008.03hPa; Wind_Direction: WS; Wind_Speed: 0.00 m/s; Rain_Count: 0.00 mm;

"sysTime1":"20210623023035"

temperature:23.27 'C; QNH :1008.04hPa; Wind_Direction: N; Wind_Speed: 0.00 m/s; Rain_Count: 0.00 mm;

"sysTime1":"20210623023047"

temperature:23.30 'C; QNH :1008.02hPa; Wind_Direction: WS; Wind_Speed: 0.00 m/s; Rain_Count: 0.33 mm;

"sysTime1":"20210623023100"

temperature:23.31 'C; QNH :1008.02hPa; Wind_Direction: WS; Wind_Speed: 20.06 m/s; Rain_Count: 0.33 mm;

"sysTime1":"20210623023113"

temperature:23.32 'C; QNH :1008.02hPa; Wind_Direction: WS; Wind_Speed: 20.40 m/s; Rain_Count: 0.33 mm;

"sysTime1":"20210623023126"

temperature:23.32 'C; QNH :1008.03hPa; Wind_Direction: WS; Wind_Speed: 5.44 m/s; Rain_Count: 0.33 mm;

"sysTime1":"20210623023138"

temperature:23.33 'C; QNH :1008.03hPa; Wind_Direction: WS; Wind_Speed: 1.36 m/s; Rain_Count: 0.33 mm;

"sysTime1":"20210623023151"

temperature:23.33 'C; QNH :1008.00hPa; Wind_Direction: W; Wind_Speed: 0.00 m/s; Rain_Count: 0.33 mm;

"sysTime1":"20210623023204"

temperature:23.33 'C; QNH :1008.02hPa; Wind_Direction: W; Wind_Speed: 0.34 m/s; Rain_Count: 1.32 mm;

"

temperature:23.33 'C; QNH :1007.99hPa; Wind_Direction: W; Wind_Speed: 0.00 m/s; Rain_Count: 1.98 mm;

图 4 5

5 经验总结

组员張健宁:

此次项目主要做的是应用层的工作。在完成项目的过程中,收获还是挺多的:

- 在处理风向仪、风速仪、雨量仪的接线时,使用工具将其改装成杜邦头,怀旧了一下。
- 在整合项目的时候，接触了几乎 STM32F103C8 所能使用的所有通信模式：UART、SPI、I2C、WiFi 模块。

项目目前来说实际还存在一些问题：

- 风向仪指向不准确。
- 整个项目采用的是前后台模式，在输出的时候感觉输出过慢。希望能够改成中断驱动模式。
- 虽然拉取了实时时间，但是 UART 接收数据的时候接收了很多冗余，后续可以对其进行优化。
- 项目在整合墨水屏的时候，引入了大量的链接库。但是感觉有很大部分是不需要用到的。这导致 STM32F103C8 的 Flash 占用率飙升。后续可以进行适当剪裁。
- 最终数据没有封装成 APRS 数据包发送到服务器上去进行测试。

组员徐伯涵：

墨水屏部分虽然例程给使用的开发环境和板子都略有不同，导致没有办法直接用，但是整体而言除了中文字体相关的部分都写得非常简单易懂，稍作修改之后就可以使用了