第二十二章 超声波测距

本章主要介绍有关超声波测距的知识，本章分为如下几个部分：

8.1 超声波测距简介

8.2 硬件设计

8.3 软件设计

8.4 下载验证

8.1 超声波测距简介

HC-SR04 超声波测距模块工作原理：

1）采用 IO 口 Trig 触发测距，给至少 10us 的高电平信号；

2）模块自动发送 8 个 40KHz 的方波，自动检测是否有信号返回；

3）有信号返回，通过 IO 口 Echo 输出一个高电平，高电平持续的时间就是超声波从发射到返回的时间。测试距离=（高电平时间*声速（340M/S））/2；

4）本模块使用方法简单，一个控制口发一个 10us 以上的高电平，就可以在接收口等待高电平输出。一有输出就可以开定时器计时，当此口变为低电平时就可以读定时器的值，此时就为此次测距的时间，方可算出距离。如此不断的周期测，即可以达到你移动测量的值。

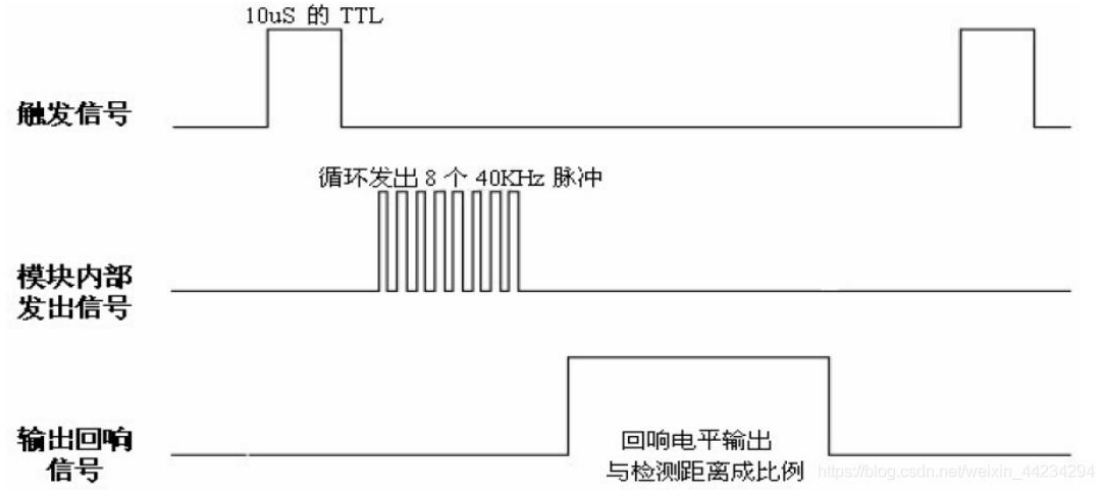

图 8.1.1 HC-SR04 超声波模块的时序触发图

编程步骤：

1. 配置好相应 GPIO，Trig 和 Echo 引脚，将 Trig 和 Echo 端口都置低；

2. 配置定时器，开启中断，并记录中断产生次数；

3. 给模块 Trig 端口发送大于 10us 的高电平信号，发出回响信号时，Echo 端呈现高电平，此时打开定时器计时；当收到回响信号时，Echo 端呈现低电平，此时关闭定时器；

4. 获取 Echo 高电平时间，利用相关公式计算出距离 =(高电平时间*声速（340M/S）)/2，可取平均值获取更精准数据。

因为单片机的定时器一般使用 us 进行计时，所以公式可以转换为$t/58$，单位为 cm。

我们作一下**单位换算**，$\frac{34000}{1000000}$，单位为 cm/us，即为：0.034cm/us

再换一个角度，1/（0.034 cm/us），即：29 us/cm， 1cm 就是 29us。

但是发送后到接收到回波，声音走过的是 2 倍的距离，所以实际距离就是 1cm，对应 58us。换成距离 cm，就是除以 58，即：t/58。

8.2 硬件设计

本章需要用到的硬件资源有：

1）极风 STM32 开发板

2）STLINK 下载器

3）HC-SR04 超声波模块

4）OLED 屏

下面介绍一下 STM32 开发板和 HC-SR04 超声波模块的连接，由于程序中使用的是 TIM2，图 8.2.1 中红框内为所需要用到的引脚，所以 HC-SR04 超声波模块的 Vcc、Trig、Echo、Gnd 分别连在 STM32 开发板的 3.3V、PA2、PA3、GND。总体连接实物图如图 8.2.2 所示：

引脚号	引脚名称	类型	I/O口电平	主功能	默认复用功能
1	VBAT	S		VBAT	
2	PC13-TAMPER-RTC	I/O		PC13	TAMPER-RTC
3	PC14-OSC32-IN	I/O		PC14	OSC32-IN
4	PC15-OSC32-OUT	I/O		PC15	OSC32-OUT
5	OSC_IN	I		OSC_IN	
6	OSC_OUT	O		OSC_OUT	
7	NIRST	I/O		NIRST	
8	VSSA	S		VSSA	
9	VDDA	S		VDDA	
10	PA0-WKUP	I/O		PA0	WKUP/USART2_CTS/ADC12_IN0/TIM2_CH1_ETR
11	PA1	I/O		PA1	USART2_RTS/ADC12_IN1/TIM2_CH2
12	PA2	I/O		PA2	USART2_TX/ADC12_IN2/TIM2_CH3
13	PA3	I/O		PA3	USART2_RX/ADC12_IN3/TIM2_CH4

图 8.2.1 引脚分配图

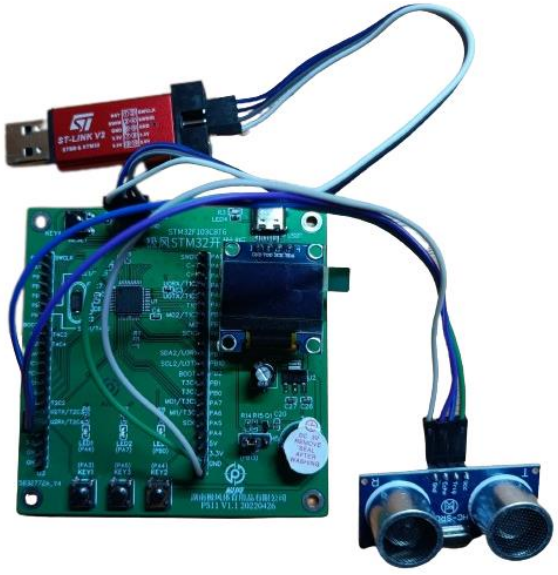

图 8.2.2 总体连接实物图

8.3 软件设计

（1）程序实现的功能

①利用 TIM2 来实现定时器计时来获取 Echo 高电平时间，并在 OLED 屏上显示平均距离数据（这里的平均距离数据是取 5 次距离数据的平均值）；

②还使用了 TIM3 来进行 1s 计时，可打开串口查看 1s 内测量距离的所有数据、平均距离数据以及平均距离数据的刷新次数。

（2）程序的实现

打开**实验 1 超声波测距\motor.uvprojx**，我们可以看到工程中拥有 9 个源文件，分别是 led.c、led.h、oled.c、oled.h、ultrasonsic.c、ultrasonsic.h、atim.c、atim.h、main.c。其中 ultrasonsic.c 存放 TIM2 和超声波模块配置的代码，实现程序功能①；atim.c 存放 TIM3 的驱动代码，实现程序功能②，main.c 文件存放应用代码。

打开 ultrasonsic.c 文件，代码如下：

```
#include "ultrasonsic.h"
#include "delay.h"

u8 msHcCount = 0;    // ms 计数

/**********************************
*  函数全称:
* void Hcsr04_Init()
*
*  函数作用:
*  初始化超声波模块和 TIM2
* **********************************/
void Hcsr04_Init()
{
      TIM_TimeBaseInitTypeDef    TIM_TimeBaseStructure;
      NVIC_InitTypeDef NVIC_InitStruct;
      GPIO_InitTypeDef GPIO_InitStructure;
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);

    /* IO 初始化  */
      GPIO_InitStructure.GPIO_Pin =GPIO_Pin_2; //  发送电平引脚 TX
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;    //  设置推挽输出
      GPIO_Init(GPIOA, &GPIO_InitStructure);
      GPIO_ResetBits(GPIOA,GPIO_Pin_2); //一开始低电平
```

```
    GPIO_InitStructure.GPIO_Pin =     GPIO_Pin_3;              // 返回电平引脚
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //浮空输入
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_ResetBits(GPIOA,GPIO_Pin_3);   // 默认低电平

    /* 定时器初始化 使用 TIM2 */
    RCC_APB1PeriphClockCmd(TIMx_CLK, ENABLE);   // 使能对应 RCC 时钟
    TIM_DeInit(TIMx); // 配置定时器基础结构体
  //自动重装载值寄存器的值，累计 TIMx_Period+1 个频率后产生一个更新或者中断
    TIM_TimeBaseStructure.TIM_Period = TIMx_Period; // 设置周期为 1000us
    TIM_TimeBaseStructure.TIM_Prescaler = TIMx_Prescaler; // 时钟预分频数
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1; // 时钟分频因子
    //计数器计数模式，设置向上计数
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIMx, &TIM_TimeBaseStructure);

    TIM_ClearFlag(TIMx, TIM_FLAG_Update);            //清除定时器的状态标志位
    TIM_Cmd(TIMx,ENABLE);
    TIM_ITConfig(TIMx,TIM_IT_Update,ENABLE);      //打开定时器更新中断

    NVIC_InitStruct.NVIC_IRQChannel = TIM2_IRQn;   //设置中断来源
    NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
    NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 1; //设置主优先级
    NVIC_InitStruct.NVIC_IRQChannelSubPriority = 1; //设置次优先级
    NVIC_Init(&NVIC_InitStruct);   //初始化
}

/**********************************
*  函数全称:
* void TIM2_IRQHandler(void)
*
*  函数作用:
*  定时器 2 中断服务程序
* **********************************/
void TIM2_IRQHandler(void)
{
    if(TIM_GetITStatus(TIMx,TIM_IT_Update) != RESET){
        TIM_ClearITPendingBit(TIMx,TIM_IT_Update); //清除更新中断标志位
        msHcCount++;   //溢出次数
    }
}
```

```
/**********************************
*  函数全称:
* u32 GetEchoTimer(void)
*
*  函数作用:
*  获取定时器时间
* ***********************************/
u32 GetEchoTimer(void)
{
    u32 time = 0;
    /*当回响信号很长是，计数值溢出后重复计数，msHcCount 用中断来保存溢出次数*/
    time = msHcCount*1000;//msHcCount 毫秒，time 微妙
    time += TIM_GetCounter(TIMx);//获取计 TIM2 数寄存器中的计数值，以便计算回响信号时间
    TIMx->CNT = 0;   //将 TIM2 计数寄存器的计数值清零
    delay_ms(50);
    return time;
}

/**********************************
*  函数全称:
* float Hcsr04GetLength(void)
*
*  函数作用:
*  获取距离数据
* ***********************************/
float Hcsr04GetLength(void)
{
    /*测 5 次数据计算一次平均值*/
    float length = 0;
    float t = 0;
    float sum = 0;
    u16 j = 0;
    while(j != 5){
        GPIO_SetBits(GPIOA,GPIO_Pin_2);   // Trig 拉高信号，发出高电平
        delay_us(20);   //持续时间超过 10us
        GPIO_ResetBits(GPIOA,GPIO_Pin_2);
        /* Echo 发出信号  等待回响信号  */
        /*  输入方波后，模块会自动发射 8 个 40KHz 的声波，与此同时回波引脚
            （Echo）端的电平会由 0 变为 1（此时应该启动定时器计时）；
            当超声波返回被模块接收到时，回波引脚端的电平会由 1 变为 0（此时应该停止定时器计数），定时器记下的这个时间即为超声波由发射到返回的总时长；*/
```

```
        while(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_3) == 0); //Echo 等待回响
        /* 开启定时器 */
        TIM_SetCounter(TIMx,0);   //清除计数器
        msHcCount = 0;
        TIM_Cmd(TIMx,ENABLE); //使能定时器
        j = j+1;   //每收到一次回响信号+1，收到 5 次就计算均值
        while(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_3) == 1);
        /* 关闭定时器 */
        TIM_Cmd(TIMx,DISABLE);
        /* 获取 Echo 高电平时间时间 */
        t = GetEchoTimer();
        length = (float)t/58;//单位为 cm
        printf("dis = %fcm\r\n",length);//打印当前计算出的距离数据
        sum += length;
    }
    length = sum/5;//五次平均值，求平均距离数据
    printf("dis_average = %fcm\r\n",length); //打印平均距离
    return length;
}
```

下面我们看看头文 ultrasonsic.h 的代码，代码中包含一些头文件定义，数据类型、函数声明，代码如下：

```
#ifndef _ULTRASONSIC_H
#define _ULTRASONSIC_H
#include "stdlib.h"
#include "stm32f10x.h"

/*TIM2*/
#define TIMx                TIM2
#define TIMx_CLK            RCC_APB1Periph_TIM2
#define TIMx_IRQn           TIM2_IRQn
#define TIMx_Period         (1000-1)
#define TIMx_Prescaler (72-1)

void Hcsr04_Init(void);        //定时中断初始化函数
float Hcsr04GetLength(void); //获取距离数据函数

#endif
```

打开 atim.c 文件，代码如下：

```
#include "atim.h"
#include "oled.h"
```

```
extern int i;

/**********************************
*  函数全称:
* void TIM3_init(uint16_t arr, uint16_t pre)
*
*  函数作用:
*  初始化 TIM3
* **********************************/
void TIM3_init(uint16_t arr, uint16_t pre)
{
	TIM_TimeBaseInitTypeDef	TIM_TimeBaseStructure;
	NVIC_InitTypeDef NVIC_InitStruct;

	/*  定时器初始化  使用 TIM3 */
	RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);	//使能对应
RCC 时钟
	TIM_DeInit(TIM3);//配置定时器基础结构体
   //自动重装载值寄存器的值，累计 TIMx_Period+1 个频率后产生一个更新或者中
断
	TIM_TimeBaseStructure.TIM_Period = arr;//设置周期为 1s
	TIM_TimeBaseStructure.TIM_Prescaler = pre;//时钟预分频数
	TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1; //时钟分频因子
	//计数器计数模式，设置向上计数，
	TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
	//重复计数器的值
	//TIM_TimeBaseStructure.TIM_RepetitionCounter=0 ;
	TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);

	TIM_ClearFlag(TIM3, TIM_FLAG_Update);		//清除定时器的状态标志位
	TIM_Cmd(TIM3,ENABLE);
	TIM_ITConfig(TIM3,TIM_IT_Update,ENABLE);	//打开定时器更新中断

	NVIC_InitStruct.NVIC_IRQChannel = TIM3_IRQn; //设置中断来源
	NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
	NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 1; //设置主优先级
	NVIC_InitStruct.NVIC_IRQChannelSubPriority = 1; //设置次优先级
	NVIC_Init(&NVIC_InitStruct);	//初始化
}

/**********************************
*  函数全称:
* void TIM3_IRQHandler(void)
```

```
*
* 函数作用:
* 定时器 3 中断服务程序
* ************************************/
void TIM3_IRQHandler(void)
{
    if(TIM_GetITStatus(TIM3,TIM_IT_Update) != RESET){
        TIM_ClearITPendingBit(TIM3,TIM_IT_Update);  //清除更新中断标志位
        printf("cnt=%d\n",i); //打印 1s 内平均距离数据的刷新次数
        i=0;
    }
}
```

下面我们看看头文 atim.h 的代码，代码中包含一些头文件定义，数据类型、函数声明，代码如下：

```
#ifndef _ATIM_H
#define _ATIM_H
#include "stdlib.h"
#include "stm32f10x.h"
#include "usart.h"
#include "ultrasonsic.h"

void TIM3_init(uint16_t arr, uint16_t pre);

#endif
```

最后，我们在 main 函数里编写应用代码，main.c 文件如下：

```
#include "stm32f10x.h"
#include <string.h>
#include <stdlib.h>

#include "stdio.h"
#include "usart.h"
#include "delay.h"
#include "oled.h"
#include "led.h"
#include "ultrasonsic.h"
#include "atim.h"

int i=0;
int main(void)
{
```

```
float length;
char str[32];

uart_init(115200);                          /*  初始化串口  */
LED_Init();                                 /*  初始化 LED */
OLED_Init();                                /*  初始化 OLED */
OLED_Clear();                               /*  清屏  */
Hcsr04_Init();
TIM3_init(10000 -1, 7200 -1);

{
    //蜂鸣器初始化
    GPIO_InitTypeDef    GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;              /* BEEP 引脚  */
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;    /*  推挽输出  */
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;    /*  高速  */
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    GPIO_ResetBits(GPIOB, GPIO_Pin_13);                     /*  关闭 BEEP */
}

while(1)
{
    i++;
    length=Hcsr04GetLength();    //获取距离
    OLED_ShowCHinese(0, 0, 6); //距
    OLED_ShowCHinese(16, 0, 7); //离
    OLED_ShowCHinese(32, 0, 5 ); //：
    sprintf(str,"%f",length);
    OLED_ShowString(40,1,str,8);    //OLED 屏显示平均距离数据

    if(length < 10)
      {//小于 10cm
        GPIO_ResetBits(GPIOA, GPIO_Pin_6);    //LED1 灭
        GPIO_SetBits(GPIOA, GPIO_Pin_7);      //LED2 亮
        GPIO_SetBits(GPIOB, GPIO_Pin_13);     /*  打开 BEEP    */
        delay_ms(300);
      }else
      {//正常距离
        GPIO_SetBits(GPIOA, GPIO_Pin_6);      //LED1 亮
```

```
                GPIO_ResetBits(GPIOA, GPIO_Pin_7); //LED1 灭
                GPIO_ResetBits(GPIOB, GPIO_Pin_13);/*  关闭 BEEP */
            }
        }
}
```

至此，我们的软件设计部分就结束了。

8.4 下载验证

在代码编译成功，下载程序完成后，可观察到 OLED 屏显示平均距离数据，打开串口可以查看 1s 内测量距离的所有数据、平均距离数据以及平均距离数据的刷新次数，即成功完成实验 1 超声波测距。