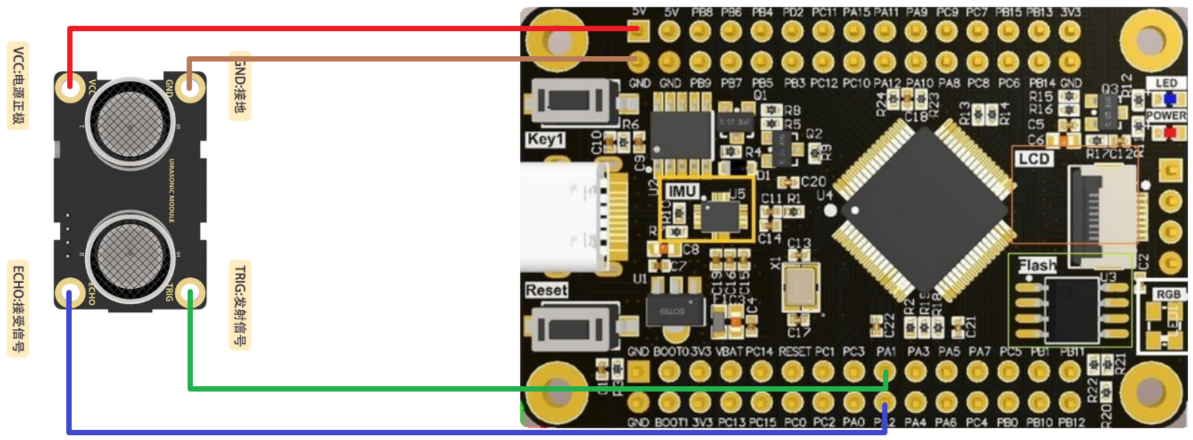
Ultrasonic ranging module: timer counting

Hardware wiring



Ultrasonic ranging module	STM32F103RCT6
VCC	5V/3.3V
TRIG	PA1
ECHO	PA2
GND	GND

Brief principle

Trigger ranging for the TRIG pin high-level signal, and give at least 10us high-level signal;

The module automatically sends 8 square waves of 40KHz and automatically detects whether there is a signal return;

There is a signal return, and the high level is output through the ECHO pin, and the time that the high level lasts is the time from the ultrasonic wave from emission to return.

Distance = (Time high * Speed of sound (340m/s)) / 2;

Main code

main.c

```
#include "stm32f10x.h"
#include "SysTick.h"
#include "UART.h"
#include "HC_SR04.h"
```

```
int main(void)
{

    SysTick_Init();//滴答定时器初始化
    UART1_Init();//UART1初始化
    TIM2_Count_Init();//定时器2功能初始化
    Trig_Echo_Init();//HC_SR04 TRIG ECHO引脚初始化 PA1 PA2

        printf("Start measuring distances!\n");

    while(1)
    {
        HC_SR04_Data();//Trig发送高电平触发测距 获取Echo高电平时间并转换成距离
    }
}
```

SysTick.c

```
#include "SysTick.h"

unsigned int Delay_Num;

void SysTick_Init(void)//滴答定时器初始化
{
    while(SysTick_Config(72));//设置重装载值 72 对应延时函数为微秒级
    //若将重装载值设置为72000 对应延时函数为毫秒级
    SysTick->CTRL &= ~(1 << 0);//定时器初始化后关闭，使用再开启
}

void Delay_us(unsigned int NCount)//微秒级延时函数
{
    Delay_Num = NCount;
    SysTick->CTRL |= (1 << 0);//开启定时器
    while(Delay_Num);
    SysTick->CTRL &= ~(1 << 0);//定时器初始化后关闭，使用再开启
}

void SysTick_Handler(void)
{
    if(Delay_Num != 0)
    {
        Delay_Num--;
    }
}
```

SysTick.h

```
#ifndef __SYSTICK_H__
#define __SYSTICK_H__

#include "stm32f10x.h"

void SysTick_Init(void);//滴答定时器初始化
void Delay_us(unsigned int NCount);//微秒级延时函数

#endif
```

UART.c

```
#include "UART.h"

void UART1_Init(void)//UART1初始化
{
    USART_InitTypeDef USART_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Enable GPIO clock */
    /* 使能GPIOA AFIO时钟 TXD(PA9) RXD(PA10) */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO, ENABLE);

    /* Enable USART1 Clock */
    /* 使能串口1 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1 , ENABLE);

    /* Configure USART1 Rx as input floating */
    /* 配置RXD引脚 PA10 浮空输入模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure USART1 Tx as alternate function push-pull */
    /* 配置TXD引脚 PA9 复用推挽输出模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* USART1 configured as follow:
    - BaudRate = 115200 baud
    - Word Length = 8 Bits
    - One Stop Bit
    - No parity
    - Hardware flow control disabled (RTS and CTS signals)
    - Receive and transmit enabled
    */
    USART_InitStructure.USART_BaudRate = 115200;//波特率设置
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;//8位数据位
    USART_InitStructure.USART_StopBits = USART_StopBits_1;//1位停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;//无奇偶校验位
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;//无需硬件流控
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;//全双工 即发送也接
受
    USART_Init(USART1, &USART_InitStructure);

    /* Enable USART1 Receive and Transmit interrupts */
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    USART_ITConfig(USART1, USART_IT_TXE, ENABLE);

    /* Enable the USART1 */
    /* 使能USART1 */
    USART_Cmd(USART1, ENABLE);
}

void NVIC_UART1_Init(void)//UART1 NVIC配置
{
    NVIC_InitTypeDef NVIC_InitStructure;

    /* Configure the NVIC Preemption Priority Bits */
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);

    /* Enable the USART1 Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}


void USART_SendString(USART_TypeDef* USARTx, char *pt)//给指定串口发送字符串
{
    while(*pt)
    {
        while(USART_GetFlagStatus(USARTx, USART_FLAG_TXE) == RESET);
        USART_SendData(USARTx,*pt);
        while(USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
        pt++;
    }
}

int fputc(int c, FILE *pt)//printf重定向
{
    USART_TypeDef* USARTx = USART1;
    while(USART_GetFlagStatus(USARTx, USART_FLAG_TXE) == RESET);
    USART_SendData(USARTx, c);
    while(USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
    return 0;
```

```
}

void USART1_IRQHandler(void)
{
    unsigned char ch;
    while(USART_GetFlagStatus(USART1, USART_FLAG_RXNE) == SET)//接收到数据
    {
        ch = USART_ReceiveData(USART1);
        printf("%c\n",ch);
    }
}
```

UART.h

```
#ifndef __UART_H__
#define __UART_H__

#include "stm32f10x.h"
#include "stdio.h"

void UART1_Init(void);//UART1初始化
void USART_SendString(USART_TypeDef* USARTx, char *pt);//给指定串口发送字符串
int fputc(int c, FILE *pt);//printf重定向
void NVIC_UART1_Init(void);//UART1 NVIC配置

#endif
```

HC_SR04.c

```
#include "HC_SR04.h"

uint16_t Time;//Echo高电平时间
float Distance;//转换之后的距离

void TIM2_Count_Init(void)//定时器2初始化
{
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;

    /* TIM2 clock enable */
    /* TIM2 时钟使能 */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

    /* Time base configuration */
    /* TIM2 配置 定时1s */
    TIM_TimeBaseStructure.TIM_Period = 0xFFFF;//设置计数值
    TIM_TimeBaseStructure.TIM_Prescaler = 72 - 1;//设置分频值(0-0xFFFF 0-65535)
72MHz/72 = 1MHz 1s计数1000000
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;//向上计数
    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

    /* TIM2 enable counter */
    /* TIM2使能计数 */
    TIM_Cmd(TIM2, ENABLE);
}

/* TRIG: PA1 输出  ECHO: PA2 输入 */
void Trig_Echo_Init(void)//HC_SR04 TRIG ECHO引脚初始化 PA1 PA2
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Enable GPIOA clock */
    /* 使能GPIOA时钟 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

    /* Configure PA1 in output pushpull mode */
    /* 配置PA1 推挽输出模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure PA2 in input floating mode */
    /* 配置PA2 浮空输入模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void HC_SR04_Data(void)//Trig发送高电平触发测距 获取Echo高电平时间并转换成距离
{
                TIM_Cmd(TIM2, ENABLE);//开启定时器

        /* 触发Trig测距，给至少 10us 的高电平信号 */
        Trig_H;
        Delay_us(20);
        Trig_L;

        /* 获得Echo信号高电平持续的时间 */
        while(Echo == 0);
        TIM_SetCounter(TIM2, 0);//开启计数
        while(Echo == 1);

        TIM_Cmd(TIM2, DISABLE);//关闭定时器
        Time = TIM_GetCounter(TIM2);//获取计数值 对应时间单位μs

        /* 将Echo信号高电平时间装换成距离 */
        Distance = Time*0.034/2;//340m/s 转换成 0.034cm/μs
        printf("Distance = %.2f cm\n",Distance);

                Delay_us(100000);//打印时间间隔 0.1s
```

```
}
```

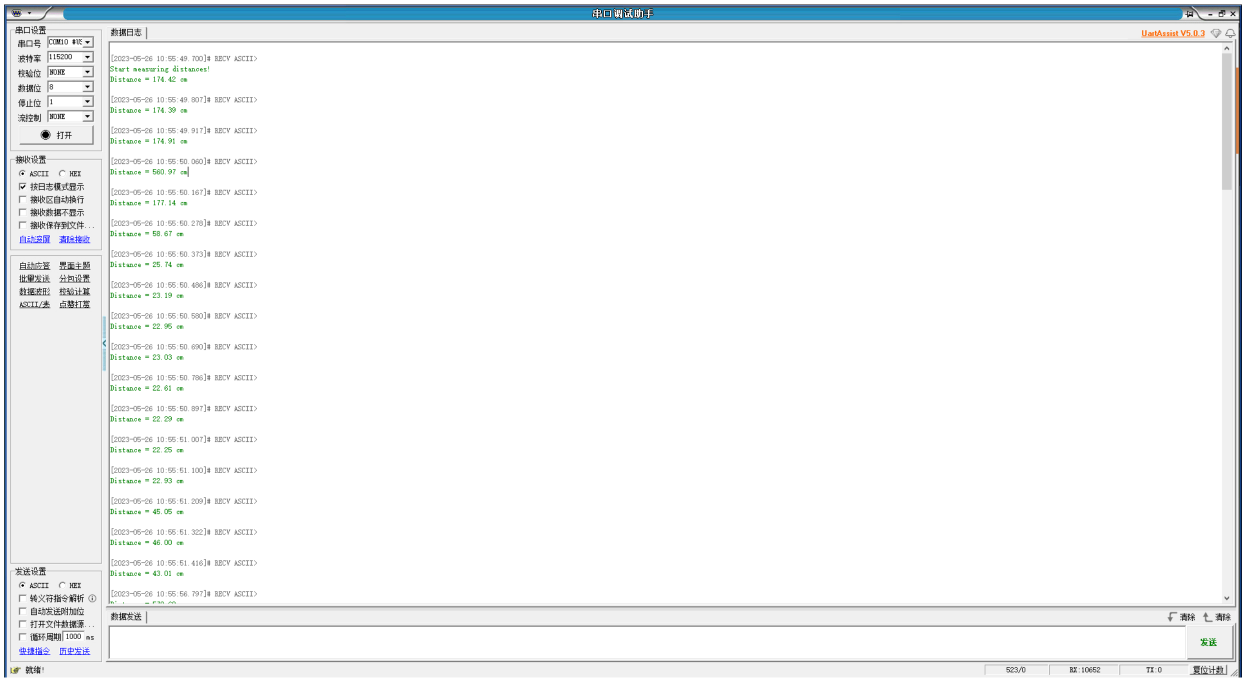
HC_SR04.h

```c
#ifndef __HC_SR04_H__
#define __HC_SR04_H__

#include "stm32f10x.h"
#include "SysTick.h"
#include "UART.h"

extern uint16_t Time;//Echo高电平时间
extern float Distance;//转换之后的距离

#define Trig_H GPIO_WriteBit(GPIOA, GPIO_Pin_1, Bit_SET)
#define Trig_L GPIO_WriteBit(GPIOA, GPIO_Pin_1, Bit_RESET)

#define Echo GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_2)

void TIM2_Count_Init(void);//定时器2初始化
void Trig_Echo_Init(void);//HC_SR04 TRIG ECHO引脚初始化 PA1 PA2
void HC_SR04_Data(void);//Trig发送高电平触发测距 获取Echo高电平时间并转换成距离

#endif
```

Phenomenon

After downloading the program, press the Reset key once, and the downloaded program will run.

At this time, the serial port will continuously print the distance measured by ultrasound.



```
[2023-05-26 10:55:49.700]# RECV ASCII>
Start measuring distances!
Distance = 174.42 cm

[2023-05-26 10:55:49.807]# RECV ASCII>
Distance = 174.39 cm

[2023-05-26 10:55:49.917]# RECV ASCII>
Distance = 174.91 cm

[2023-05-26 10:55:50.060]# RECV ASCII>
Distance = 560.97 cm

[2023-05-26 10:55:50.167]# RECV ASCII>
Distance = 177.14 cm

[2023-05-26 10:55:50.278]# RECV ASCII>
Distance = 58.67 cm

[2023-05-26 10:55:50.373]# RECV ASCII>
Distance = 25.74 cm

[2023-05-26 10:55:50.486]# RECV ASCII>
Distance = 23.19 cm

[2023-05-26 10:55:50.580]# RECV ASCII>
Distance = 22.95 cm

[2023-05-26 10:55:50.690]# RECV ASCII>
Distance = 23.03 cm

[2023-05-26 10:55:50.786]# RECV ASCII>
Distance = 22.61 cm

[2023-05-26 10:55:50.897]# RECV ASCII>
Distance = 22.29 cm

[2023-05-26 10:55:51.007]# RECV ASCII>
Distance = 22.25 cm

[2023-05-26 10:55:51.100]# RECV ASCII>
Distance = 22.93 cm

[2023-05-26 10:55:51.209]# RECV ASCII>
Distance = 45.05 cm

[2023-05-26 10:55:51.322]# RECV ASCII>
Distance = 46.00 cm

[2023-05-26 10:55:51.416]# RECV ASCII>
Distance = 43.01 cm

[2023-05-26 10:55:56.797]# RECV ASCII>
```