目录

图目录

表目录

摘要

近几年，随着我国科技水平的提高，测距手段变得十分先进和多样，目前已经有红外测距、激光测距和超声波测距等多种方法，前两种方法制作难度较大，成本较高，一般应 用于军事测量和工业测量。超声波测距成本较低，操作简 单，指向性强，能量消耗缓慢，易于做到实时控制，在民用测 距的领域得到了广泛的应用。而且超声波测距可以在较差的环境中进行，对于被测物处于有灰尘、有毒、黑暗等恶劣环境下有一定的适应能力。超声波测距时要求超声波测距仪与被测物体不直接接触并相隔一定的距离，以便能够清晰地测量出结果。因此，超声波测距仪常应用于泊车辅助系统、智能导盲系统、移动机器人等[1]。

目前，常见的单片机有 51 系列和 MSP430 系列等。51 系列应用广泛、功能完备，但抗干扰能力不强；与上述相比，MSP430 系列具有功能强大、集成度高、抗干扰能力强、成本低、市场流通性大、能进入低功耗模式运行，保证产品的时间长、损耗低、精度高、稳定性佳等特点。

综上所述，本文将基于 MSP430G2553 单片机进行超声波测距仪的设计。

关键字：超声波；单片机；MSP430；测距

Abstract

In recent years, with the improvement of China's scientific and technological level, ranging has become very advanced and diverse, there are infrared ranging, laser ranging and ultrasonic ranging and other methods, the first two methods are difficult to make, high cost, generally should be used for military measurement and industrial measurement.

Ultrasonic ranging cost is low, operation simple, strong directivity, slow energy consumption, easy to achieve real-time control, has been widely used in the field of civil distance measurement.

And ultrasonic ranging can be carried out in a poor environment, for the measured objects in dust, toxic, dark and other harsh environment has a certain adaptability.

Ultrasonic ranging requires ultrasonic rangefinder and measured objects do not directly contact and a certain distance, in order to be able to clearly measure the results.

Therefore, ultrasonic rangefinder is often used in parking assistance system, intelligent guide system, mobile robot, etc.

At present, the common SCM has 51 series and MSP430 series.

Series 51 is widely used and has complete functions, but its anti-interference ability is not strong.

Compared with the above, MSP430 series has powerful functions, high integration, strong anti-interference ability, low cost, large market liquidity, can enter the low-power mode of operation, to ensure the product's long time, low loss, high precision, good stability and other characteristics.

To sum up, this paper will be based on MSP430G2553 MCU ultrasonic rangefinder design.

Keywords: ultrasonic; SCM; MSP430; ranging

超声波测距系统设计

1 设计题目

超声波测距系统设计。

2 设计目的

运用单片机原理及其应用等课程知识,根据题目要求进行软硬件系统的设计和调试，从而加深对传感器的理解，把学过的比较零碎的知识系统化，比较系统地学习开发单片机应用系统的基本步骤和基本方法，使应用知识能力、设计能力、调试能力以及报告撰写能力等有一定的提高。

3 设计内容及要求

设计一个测距系统，以超声波传感器模块为传感单元，测量范围 5cm-200cm，测距精度±5mm，显示分辨率 1 cm。

4 系统总体结构

此次超声波测距系统包括：超声波传感器模块，显示模块，控制模块。

系统上电，超声波模块的电源由 MSP430G2553 开发板上的 3.3V 电源提供。单片机的两个 IO 口用来与超声波模块进行通信，将超声波模块发送的方波信号通过 MSP430G2553 捕获，借助软件编程处理最后将距离显示在 128 段液晶显示屏上。系统总设计框图如图 4-1 下：

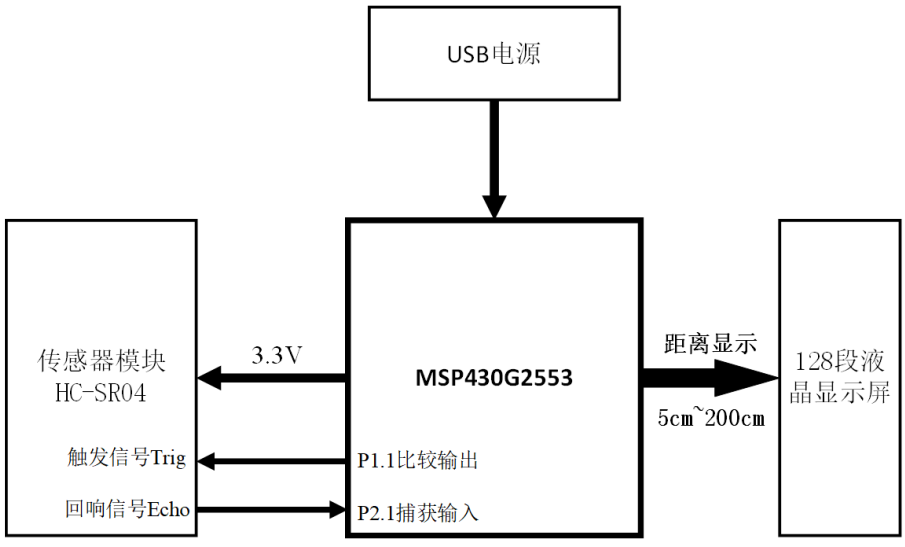


图 4-1 系统框图

5 硬件设计

5.1 元件选取

(一) MSP430G2553

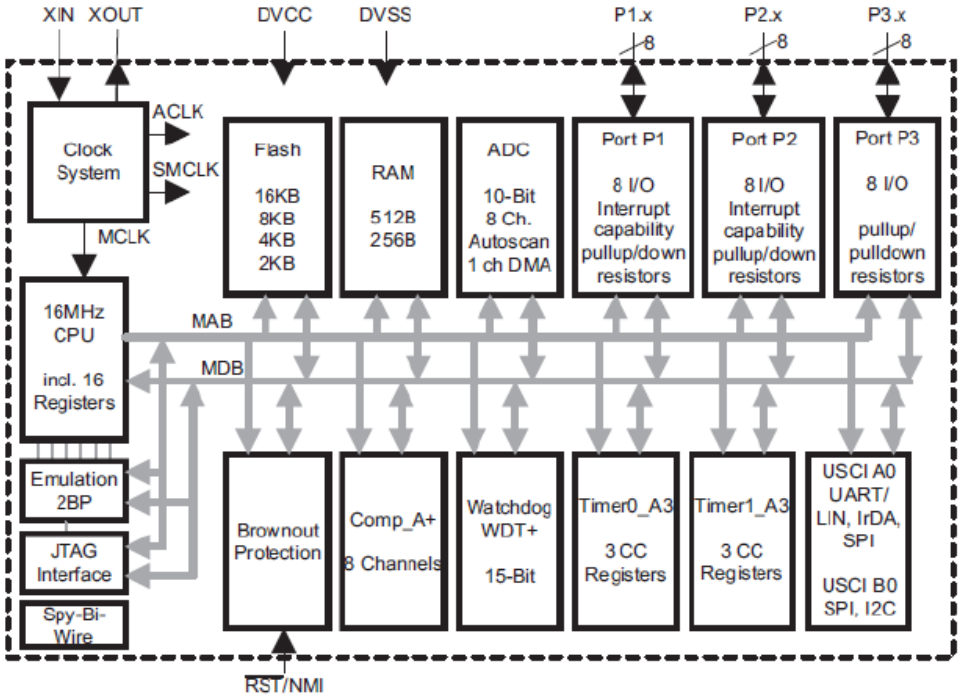


图 5-1 MSP430G2x53 微控制器结构

如图 5-1 所示，此次课程设计主要用到了 MSP430G2553 内部的定时器功能。

MSP430 单片机中 Timer_A 定时器就是一种辅助功能强大的定时器，具备捕获和 PWM 输出等极其有用功能。

MSP430x2xx 系列单片机的 Timer_A 模块的整体构造如图所示，包括 1 个 16 位定时器（Timer Block）和 3 个捕获比较模块（CCRx）。

1）16 位定时器的最大定时值 65535，当前计数值被存放在 TAR 寄存器中。

2）CCRx 的捕获模块 Caputre 由 1 个输入 IO 口（CCIx）控制，输入上升沿或下降沿均能触发比较模块动作，捕获发生后的瞬间 TAR 值被存入 TACCRx 寄存器。

3）CCRx 的比较模块 Comparator 控制 1 个输出 IO 口（TAx）去生成各种脉冲波形。当 TAR 计数值与预存入 TACCRx 寄存器的值相等时，比较模块动作，以某种预设规则控制 IO 电平，生成波形。

4）由于捕获模块 Caputre 和比较模块 Comparator 共用了 TACCRx 寄存

器，捕获 Capture 的功能是写 TACCRx，而比较 Comparator 的功能是读 TACCRx 模块，所以捕获和比较不能同时使用。

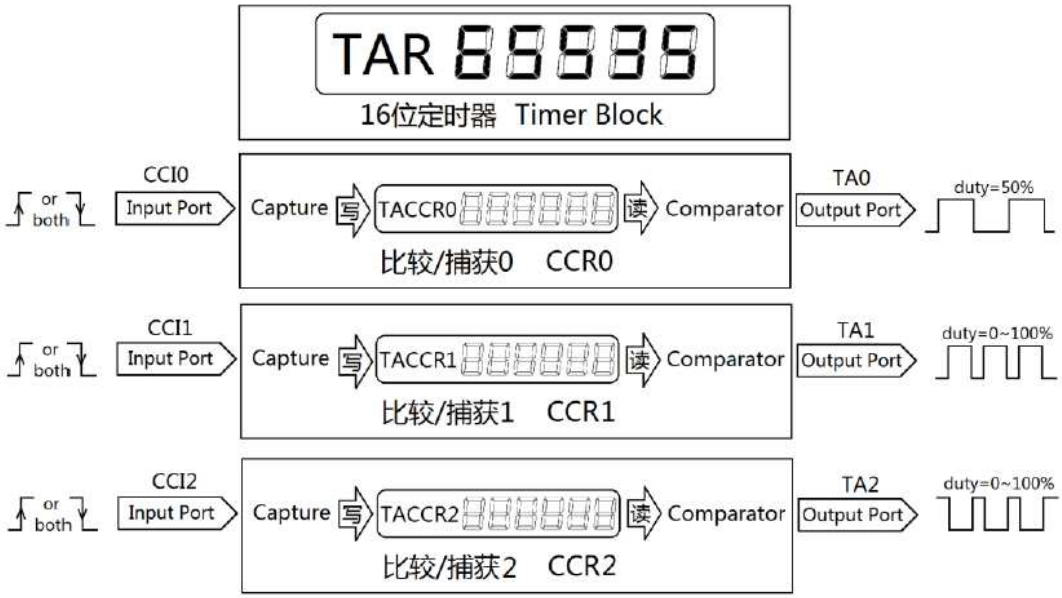


图 5-2 Timer_功能框图

捕获模块

将 CAP 设置为 1，CCRx 工作于捕获模式。主定时器一般设置为连续计数模式，当 CCRx 检测到 CCIx（某带捕获功能的 IO 口）的电平边沿时，瞬间读取 TAR 寄存器的值并写入 TACCRx 。CCRx 可以选择检测上升沿或下降沿，或者都检测。CCRx 用于测定信号脉宽时，只需要分别记录信号上升沿时刻和下降沿时刻，两时刻相减就是脉宽；而测量频率时，连续记录两次上升沿时刻，相减就是周期。

比较模块

当 CAP=0 时，CCRx 工作于比较模式。CCR0 在比较模式中，将用于设定定时器的周期，所以我们暂时当 CCR0“牺牲”了，只讨论 CCR1 和 CCR2 的工作情况。当 CCR1/2 发现 TAR 的值与 TACCR0 或它们自己的 TACCRx 相等时，便会自动改变输出 IO 口 TAx 的输出电平，从而生成波形。改变的规则由 OUTMODx 寄存器决定，共有 8 种规则。

这 8 种规则配合主定时器 TAR 的 3 种模式（连续计数、增计数、增减计数），可以无需 CPU 干预生成各种波形。

本次课题，我将 P1.1 复用为比较输出引脚，用于产生周期为 100ms，高电平持续时间为 20us 的 PWM 波。将 P2.1 复用为捕获输入引脚，用来捕获超声波传感器模块 Echo 引脚输入的高电平持续时间。

(二) HC-SR04

HC-SR04 超声波测距模块可提供 2cm-400cm 的非接触式距离感测功能，测距精度可达高到 3mm；模块包括超声波发射器、接收器与控制电路。

基本工作原理：

(1)采用 IO 口 Trig 触发测距，给最少 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。

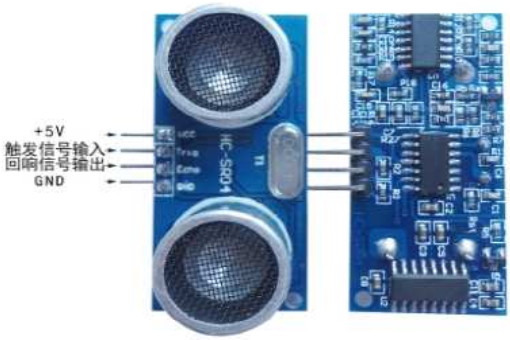


图 5-3 超声波模块实物

如图 5-3 接线，VCC 供 5V 电源，GND 为地线，Trig 触发控制信号输入，Echo 回响信号输出等四个接口端。下表格 5-1 为模块参数。

表格 5-1 HC-SR04 模块参数

工作电压	DC 5 V
工作电流	15mA
工作频率	40kHz
最远射程	4m
最近射程	2cm
测量角度	15 度
输入触发信号	10uS 的 TTL 脉冲
输出回响信号	输出 TTL 电平信号，与射程成比例

(三) 128 段 LCD 液晶屏

MSP430G2553 通过 I2C 协议 SCL、SDA，对应为 P1.6 和 P1.7 去控制扩展版上的 TCA6416A 芯片输出 4 个信号 CS、WR、RD、DATA，对应的引脚分别为 P1.4、P1.6、P1.5、P1.7，控制 LCD 驱动芯片 HT1621，来实现 128 段 LCD 的显示。下图 5-5、图 5-4 为 LCD 的控制流程和 LCD 原理图。

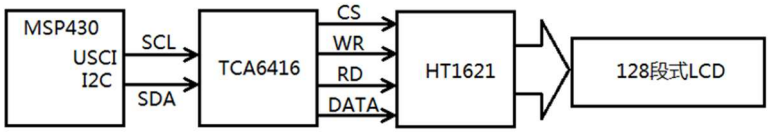


图 5-5 LCD 控制流程

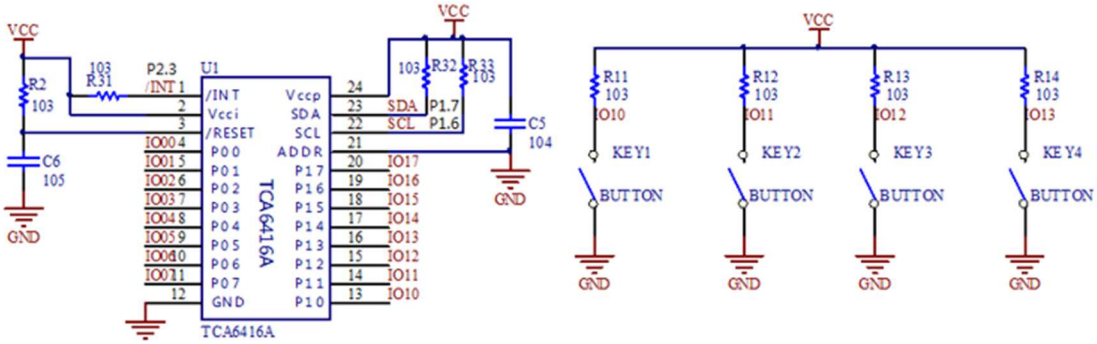


图 5-6 基于 I2C 的 IO 扩展

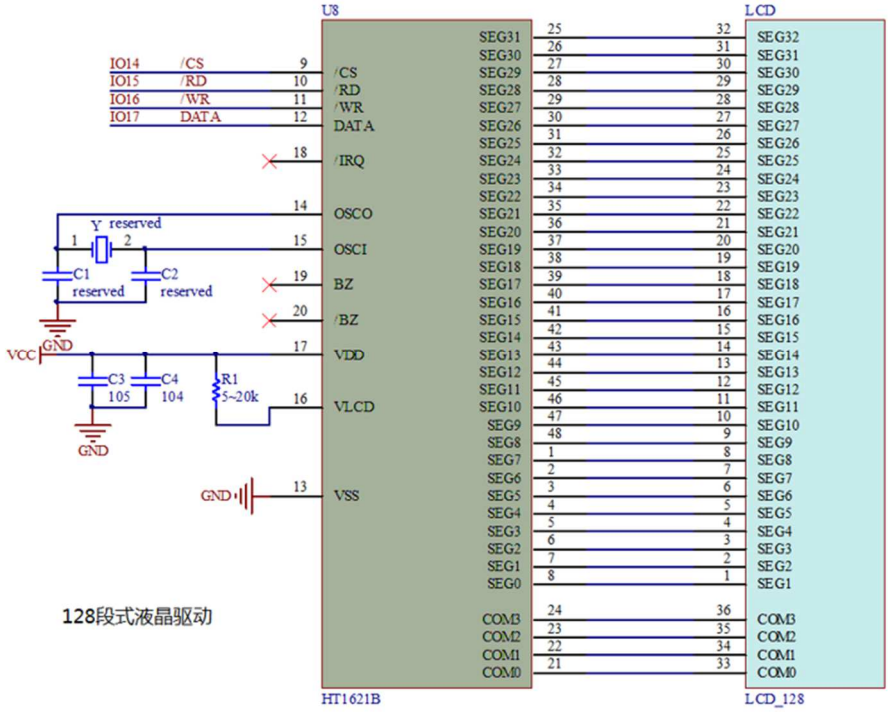


图 5-4 LCD 驱动模块

5.2 电路设计

(一) 超声波模块

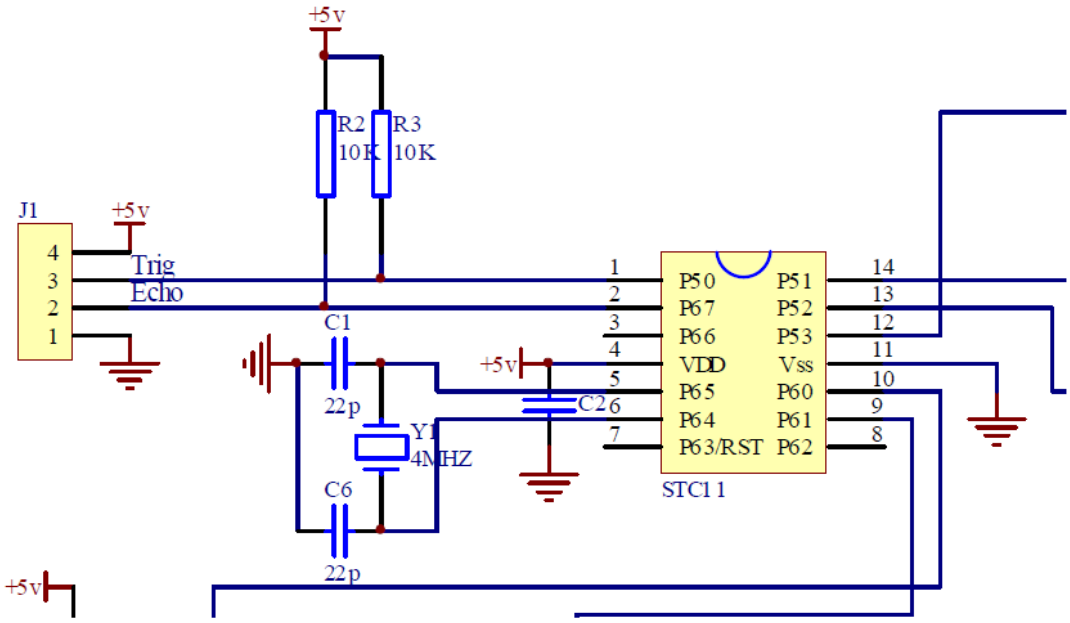


图 5-7

STC11 在这里的作用是根据收到的发送信号 TRIG，发送一段超声波波形给 MAX232；还有个作用是根据接收到的超声波波形，返回一段 ECHO 信号。

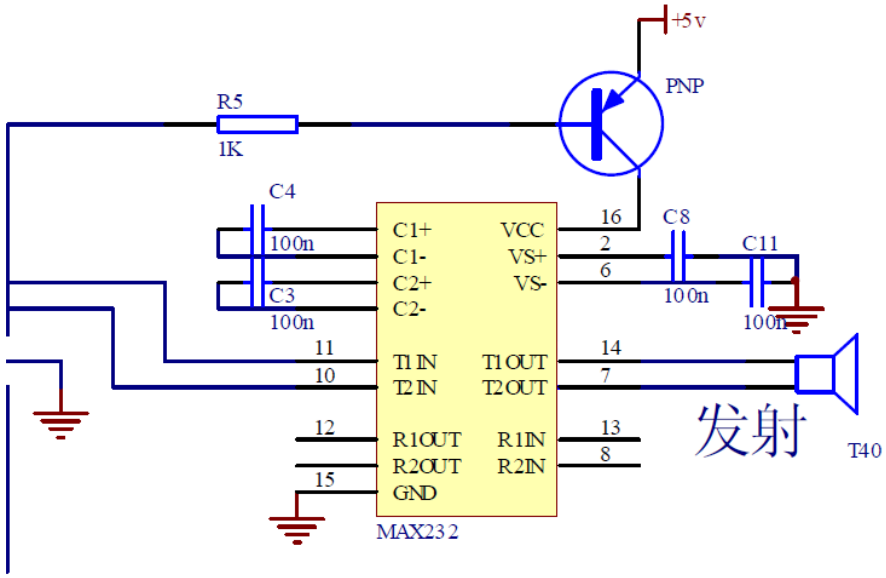


图 5-8

MAX232 在这里做电平转换，因为单片机给的波形是 TTL 波形，这里要转换电平，提高发射功率。

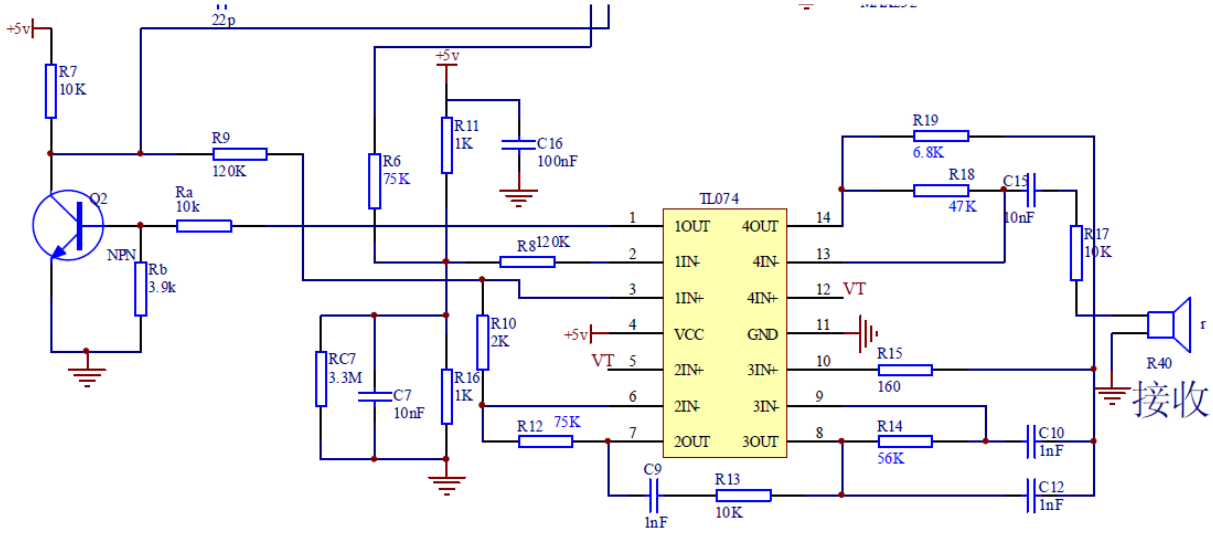


图 5-9

TL074 是对接收的超声波波形进行滤波、放大、解调之类的。

6 软件设计

6.1 中断流程图

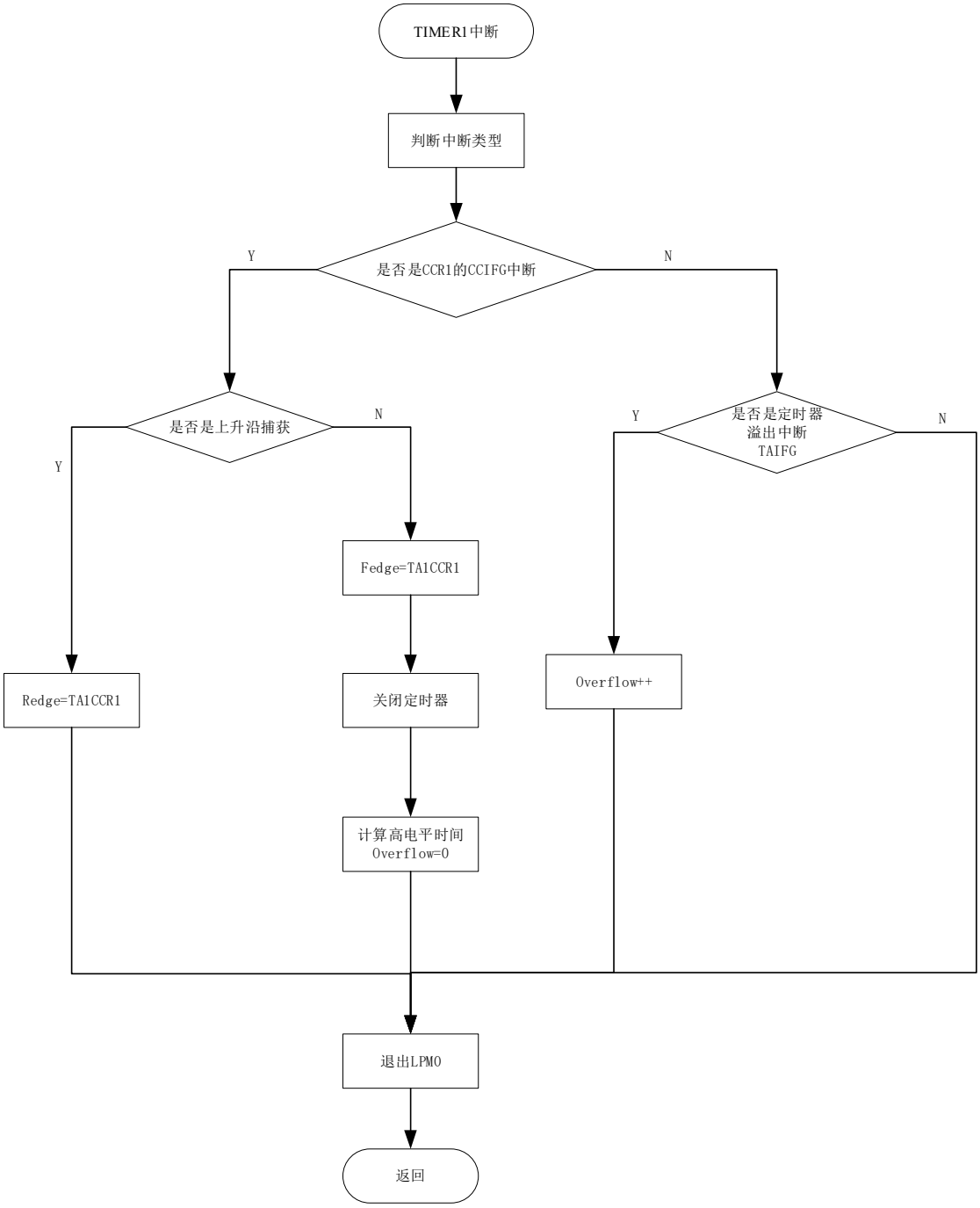


图 6-1 定时器中断流程图

如上图 6-1 所示，为定时器 TIMER1 的中断流程图，当发生定时器 TIMER1 中断时，进入中断服务函数判断引发的是哪种类型的中断，此次用到的是 CCIFG 捕获中断向量和 TAIFG 溢出中断向量。一旦引发 CCIFG 中断，通过查询 IO 引

脚的状态标志位 CCI 来判断发生的是上升沿捕获还是下降沿捕获。如果是上升沿捕获，读取 TA1CCR1 的值，赋值给变量 REdge；如果是下降沿捕获，读取 TA1CCR1 的值，赋值给变量 FEdge，关闭定时器，并计算高电平的时间。一旦发生定时器溢出中断 TAIFG，将溢出标志 overflow 累加。执行完中断服务函数里的子程序后，退出低功耗模式。

6.2 主函数流程图

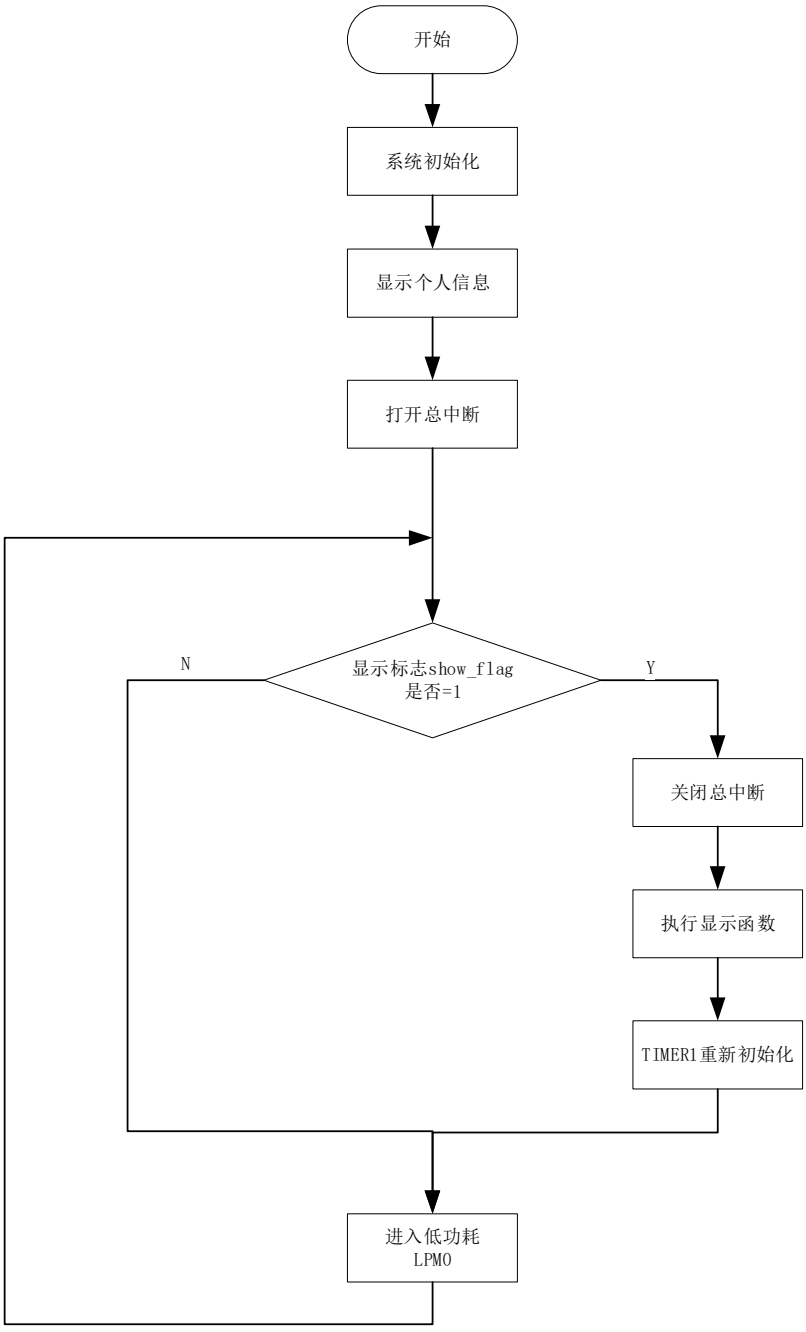


图 6-2 主函数流程图

如**错误!未找到引用源。**所示，为此次主程序的流程图，系统上电，系统程序进入初始化，显示个人信息，打开总中断，进入死循环。开始，判断显示标志show_flag=1，如果 show_flag=1，那么关闭总中断，执行显示函数，定时器初始化，进入低功耗等待中断函数退出低功耗模式 LPM0；如果 show_flag=0，那么直接进入低功耗模式 LPM0，等待中断函数退出低功耗模式 LPM0。

6.3 显示函数流程图

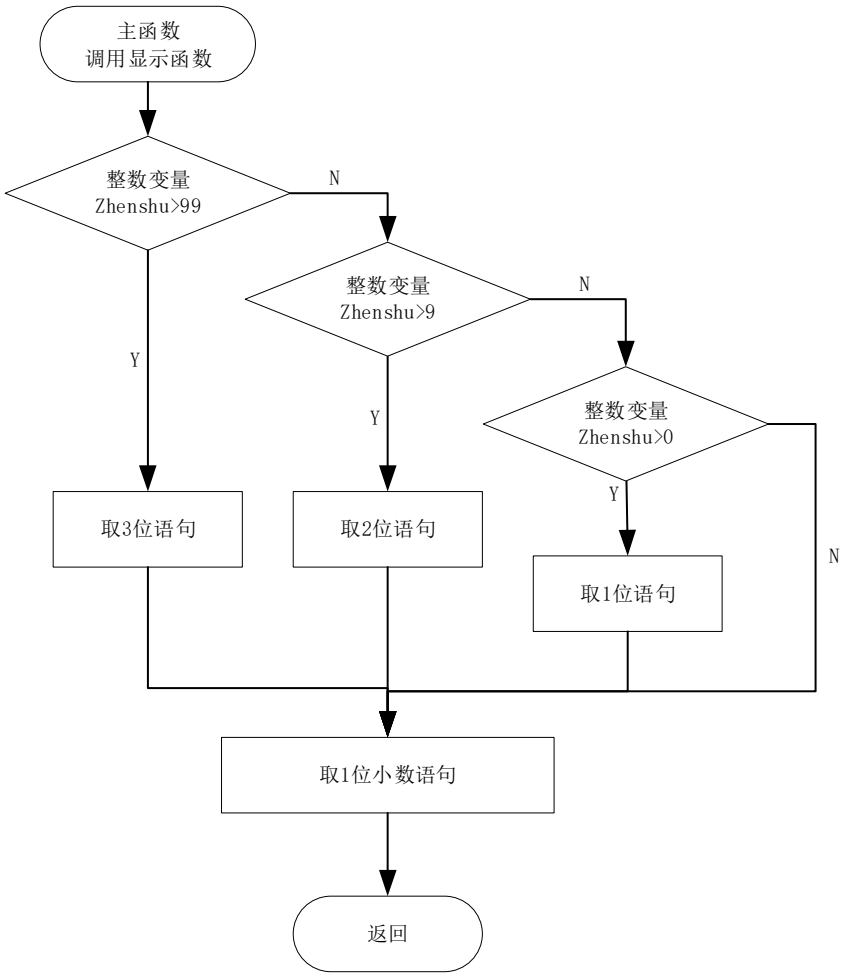


图 6-3 显示函数流程图

主函数调用显示函数之前，已经对距离的整数部分和小数部分完成处理，将整数部分保存在变量 zhenshu 中，将小数部分保存在变量 xiaoshu 里。调用显示函数时，首先判断整数部分的距离区间，然后再执行不同的取位函数，最后将整数和小数一同显示在 128 段液晶显示屏上。

7 硬件调试

7.1 距离测试

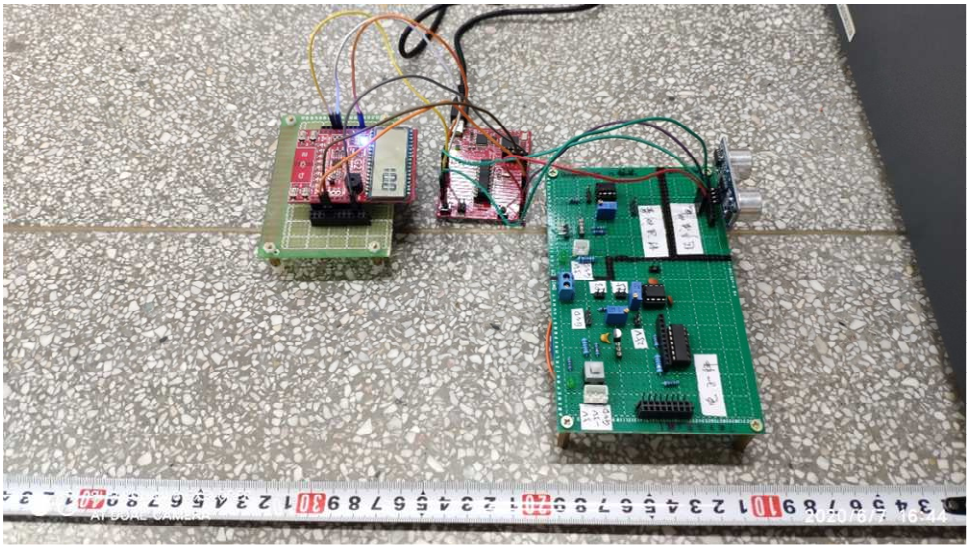
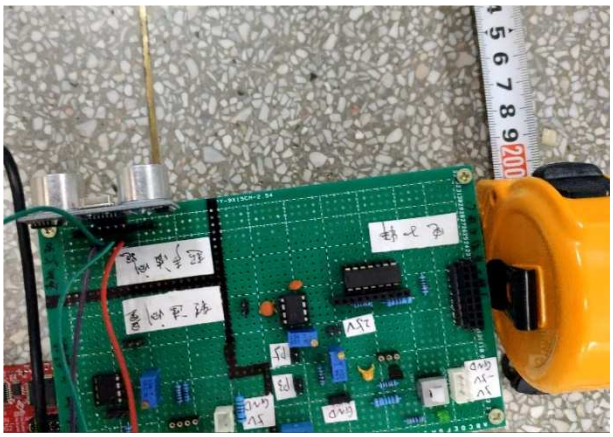


图 7-1 10cm 距离测试

如图 7-1 所示，为 10cm 距离测试，LCD 显示稳定，比较准确。



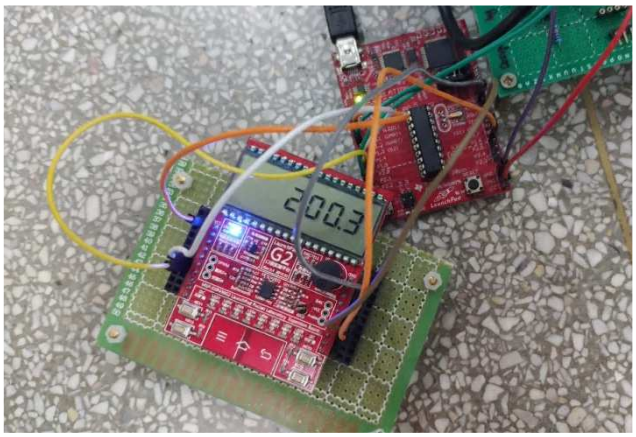
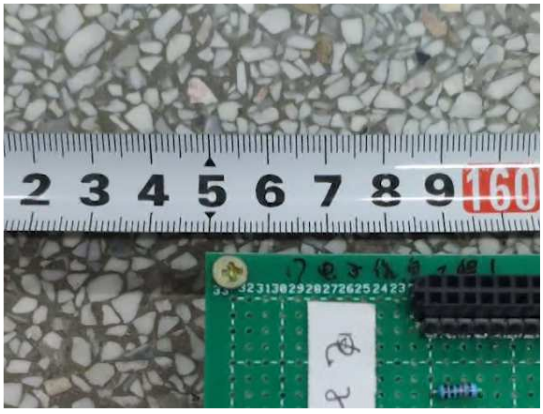


图 7-2 200cm 距离测试

如图 7-3 所示，为 200cm 距离测试，LCD 显示稳定，比较准确。



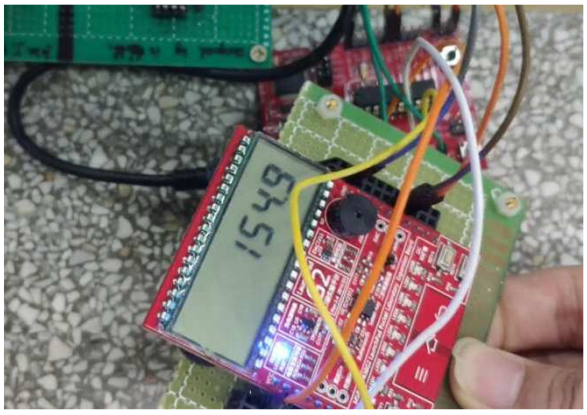


图 7-3 155cm 距离测试

如图 7-3 所示，为 155cm 距离测试，LCD 显示稳定，比较准确。

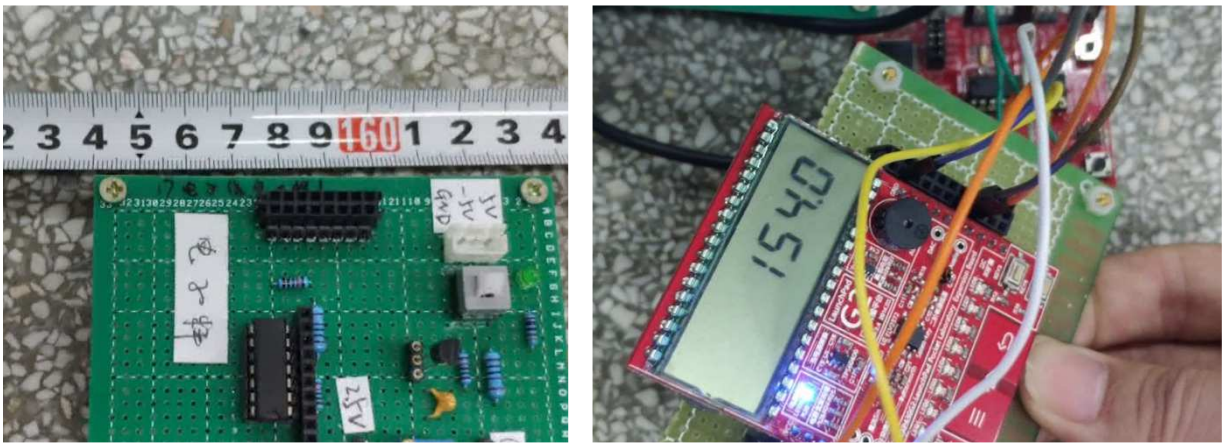


图 7-4 154cm 距离测试

如图 7-4 所示，为 154cm 距离测试，LCD 显示稳定，比较准确

图 7-5 61.5cm 距离测试

如图 7-5 所示，为 61.5cm 距离测试，LCD 显示稳定，比较准确。

图 7-6 104cm 测试

如图 7-6 所示，为 104cm 距离测试，LCD 显示稳定，比较准确。

7.2 数据记录

表格 7-1 调试数据记录

卷尺测量 单位 cm	LCD 第 1 次显示 单位 cm	拟合前误差	LCD 第 2 次显示 单位 cm	拟合后误差
5.0	4.1	-18%	4.7	**-6%**
10.0	9.8	-2%	10.3	3%
15.0	14.1	-6%	14.8	-1%
20.0	19.0	-5%	19.8	-1%
25.0	23.5	-6%	24.5	**-2%**
30.0	28.9	-4%	29.4	**-2%**
35.0	33.5	-4%	34.7	-1%
40.0	38.4	-4%	39.7	-1%
45.0	43.2	-4%	44.7	-1%
50.0	48.6	-3%	50.0	0%
55.0	53.2	-3%	55.1	0%
60.0	58.7	-2%	60.3	0%
65.0	63.7	-2%	65.0	0%
70.0	68.8	-2%	70.0	0%
75.0	73.6	-2%	75.5	1%
80.0	77.9	-3%	80.5	1%
85.0	83.2	-2%	85.7	1%
90.0	87.6	-3%	90.3	0%
95.0	92.8	-2%	95.1	0%
100.0	97.3	-3%	99.9	0%
105.0	102.1	-3%	104.7	0%
110.0	107.1	-3%	109.6	0%
115.0	111.5	-3%	114.9	0%
120.0	117.0	-3%	119.7	0%
125.0	121.6	-3%	124.7	0%
130.0	126.1	-3%	129.6	0%
135.0	131.5	-3%	134.9	0%
140.0	136.3	-3%	139.5	0%
145.0	141.3	-3%	144.9	0%
150.0	146.1	-3%	150.1	0%
155.0	150.8	-3%	154.7	0%
160.0	155.7	-3%	160.2	0%
165.0	159.8	-3%	165.2	0%

170.0	165.6	-3%	169.9	0%
175.0	170.8	-2%	175.0	0%
180.0	175.9	-2%	180.1	0%
185.0	180.4	-2%	185.0	0%
190.0	185.7	-2%	190.6	0%
195.0	190.8	-2%	195.0	0%
200.0	195.6	-2%	200.6	0%

实验测量数据如上表所示，第一次测量温度误差在 4%上下波动。数据拟合以 LCD 显示的距离为横轴，实际距离为纵轴，绘制二阶多项式曲线，如图 7-7 所示。

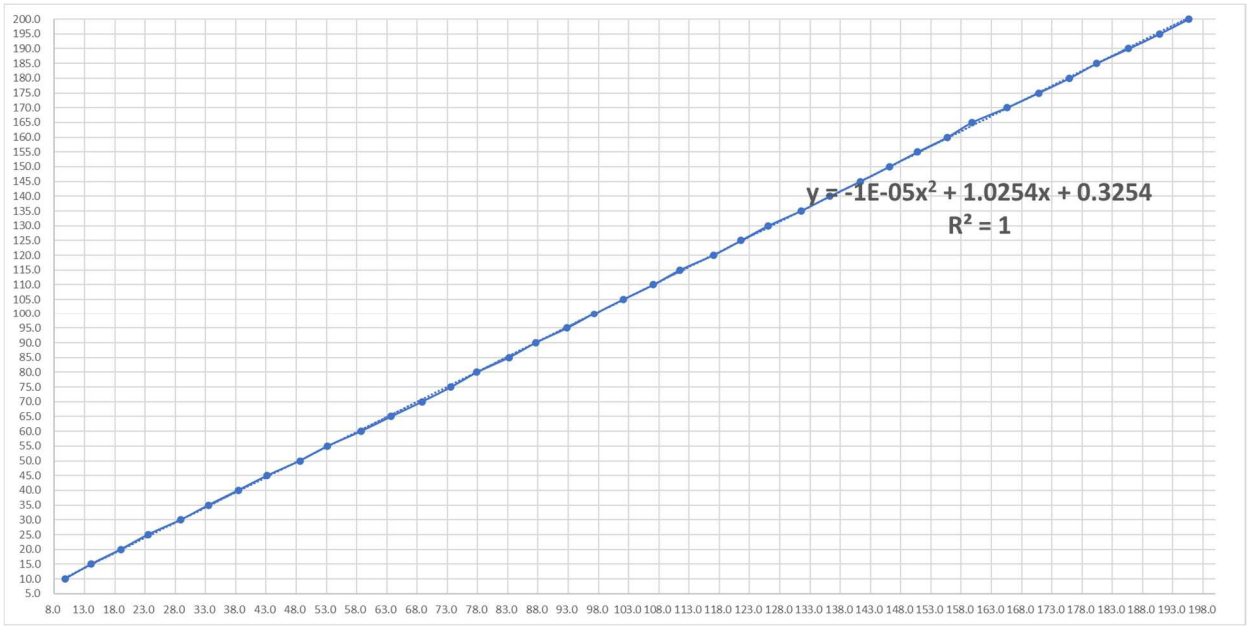


图 7-7 第一次拟合曲线

将第一次数据拟合的曲线公式：

$$y = -1 \times 10^{-5}x^2 + 1.0254x - 0.3254 \tag{7-1}$$

带入软件重新编译运行，记录第 2 组数据。由上表格 7-1 可以观察到，50cm 以下的距离误差控制在 3%左右，50cm~200cm 的误差能够控制在 1%以下。

8 设计小结

此次的超声波测距设计让我对超声波传感器有了些许了解。除了翻阅超声波模块的技术手册外，此次的波形捕获，我又一次加深了对 MSP430G2553 内部定时器模块的认识。波形捕获的关键之一在于中断函数的编写，将 IO 引脚初始化为波形捕获输入后，一旦发生沿的跳变，就会引发 CCIFG 中断。此次的课题我用到了 TIMER1 的 CCR1 作为波形的捕获寄存器，TIMER0 的 CCR0 和 CCR1 一起作用，比较输出 PWM 波。另外，周期的计算也是波形捕获的重中之重，需要考虑变量的类型、运算溢出、计数溢出等一系列问题。

此次的传感器的课程设计，感谢李老师和季老师的指导与帮助，不管是在硬件电路设计上还是程序设计上他们都给了我特别大的启发。

9 参考文献

[1]崔佳絮,陶俊杰.基于 MSP430G2553 超声波测距仪的设计[J].科教文汇(中旬刊),2020(05):82-83+95.

附件 1 源程序代码

```
#include <msp430.h>
#include "LCD_128.h"
#include"TCA6416A.h"
#include "HT1621.h"

int REdge=0,FEdge=0,zhengshu=0,xiaoshu=0;
int overflow=0;
unsigned char show_flag=0;
void LCD_Init();
void Distance_Display();
void display_my_info();
void cal_distance();
void timer1_stop();
void gpio_init();
void timer0_init();
void timer1_init();
int main(void)
{
   WDTCTL = WDTPW + WDTHOLD;
   BCSCTL2=DIVS_1;
   LCD_Init();
   TCA6416A_Init();
   display_my_info();
   gpio_init();
   timer0_init();
   timer1_init();
   _EINT();
   while(1)
```

```
    {
        if(show_flag==1)
        {
            _DINT();
            Distance_Display();
            timer1_init();
            show_flag=0;
            _EINT();
        }
        LPM0;
    }
}

void gpio_init()
{
    P1DIR |= BIT1; //P1.1 设置为输出
    P1SEL |=BIT1;//使用 P1.1 复用功能

    P2DIR &=~ BIT1; //P2.1 设置为输入
    P2SEL |=BIT1;//使用 P2.1 复用功能
}

void timer0_init()
{
    TA0CCR0=50000;//触发信号为周期 100ms
    TA0CCR1=10;//高点平为 20us
    TA0CTL=TASSEL_2+MC_1+TACLR;//时钟源选择 SMCLK,是经过 2 分频以后的 0.5MHz,计数模式为增模式，同时清除一下计数器
    TA0CCTL0=OUTMOD_3;
}
```

```
void timer1_init()
{
    TA1CCTL1 = CAP + CM_3 + CCIE + SCS + CCIS_0;
    TA1CTL |= TASSEL_1 + MC_2 + TACLR;          // SMCLK, Cont Mode;
start timer
}

void timer1_stop()
{
    TA1CTL |= MC_0 + TACLR; // stop
}

void cal_distance()
{
    static char count=0;
    static float sum=0;
    float ON_Period=0;

    ON_Period = (65536.0*overflow+FEdge-REdge)*17000/32768;
//扩大 100 倍
    sum=sum+ON_Period;
    count++;
    if(count==15)
    {
        show_flag=1;
        ON_Period=sum/15.0;
        //ON_Period=1.0229*ON_Period + 0.4164;
        ON_Period=-0.00001*ON_Period*ON_Period
+1.0254*ON_Period+0.3254;
```

```
            zhengshu=(int)ON_Period;
            xiaoshu=(ON_Period-zhengshu)*10;
            count=0;
            sum=0;
            overflow=0;
        }
    }

    #pragma vector = TIMER1_A1_VECTOR
    __interrupt void TIMER1_A1_ISR (void)
    {
      switch(TA1IV)//AIV 的值是在 0--10 内的偶数时才会执行 switch 函数内的语句
        {
            case  TA1IV_NONE:  break;                    // Vector  0:  No interrupt
            case  TA1IV_TACCR1:                          // Vector  2:  TACCR1 CCIFG
              if (TA1CCTL1 & CCI)                        // Capture Input Pin Status
              {
                  REdge= TA1CCR1;
              }
              else
              {
                  FEdge = TA1CCR1;
                  timer1_stop();
                  cal_distance();
              }
              break;
            case TA1IV_TACCR2: break;                    // Vector  4:  TACCR2
```

```
CCIFG
            case  TA1IV_6:  break;                          // Vector     6:
ON_Perioderved CCIFG
            case  TA1IV_8:  break;                          // Vector     8:
ON_Perioderved CCIFG
            case TA1IV_TAIFG:
                        overflow+=1;
            default:    break;
        }
        LPM0_EXIT;
    }

    void display_my_info()
    {
        LCD_DisplaySeg(4);
        LCD_DisplaySeg(9);
        LCD_DisplaySeg(10);//显示 J
        LCD_DisplayDigit(8, 2);
        LCD_ClearSeg(12);
        LCD_ClearSeg(19);//显示 H
        LCD_DisplayDigit(0, 3);
        LCD_ClearSeg(25);
        LCD_ClearSeg(26);
        LCD_ClearSeg(27);//显示 L
        LCD_DisplayDigit(1, 4);
        LCD_DisplayDigit(0, 5);
        LCD_DisplayDigit(7, 6);
        HT1621_Reflash(LCD_Buffer);
        __delay_cycles(1000000);
        LCD_Clear();
```

```
    LCD_DisplayDigit(0, 5);
    LCD_DisplayDigit(0, 6);
    LCD_DisplaySeg(_LCD_DOT4);
    HT1621_Reflash(LCD_Buffer);
}

void LCD_Init()
{
    HT1621_init();
    //相关硬件的初始化，其中 I2C 模块的初始化由 TCA6416A 初始化函数在内部完成了， LCD_128 库函数由 HT1621 初始化函数在内部引用了
    //-----显示固定不变的 LCD 段-----
}

void Distance_Display()
{
      LCD_DisplayDigit(LCD_DIGIT_CLEAR,3);
      LCD_DisplayDigit(LCD_DIGIT_CLEAR,4);
        //-----根据 ON_Period 拆分并显示数字-----
      if(zhengshu>99)//100~999（3 位）
      {
          LCD_DisplayDigit(zhengshu/100,3);
          LCD_DisplayDigit((zhengshu%100)/10,4);
          LCD_DisplayDigit(zhengshu%10,5);
          LCD_DisplayDigit(xiaoshu,6);
      }
      else if(zhengshu>9)//（2 位）
      {
          LCD_DisplayDigit(zhengshu/10,4);
          LCD_DisplayDigit(zhengshu%10,5);
```

```
            LCD_DisplayDigit(xiaoshu,6);
        }
        else
        {
            LCD_DisplayDigit(zhengshu,5);
        LCD_DisplayDigit(xiaoshu,6);
        }
        HT1621_Reflash(LCD_Buffer);//-----更新缓存，真正显示-----
}
```