RT-THREAD 星火 1 号教育套件模块例程

RT-THREAD 文档中心

WWW.RT-THREAD.ORG

Wednesday 20th September, 2023

版本和修订

Date	Version	Author	Note
2023-09-20	V1.1.0	RT-Thread	SDK V1.1.0 版本增加本文档内容

目录

第 1 章

按键输入例程

1.1 简介

本例程主要功能是通过板载的 PMOD1 接口和转接板，将 4*4 矩阵按键接入开发板，PMOD2 也可以用，但是需要改引脚信息。

1.2 硬件说明

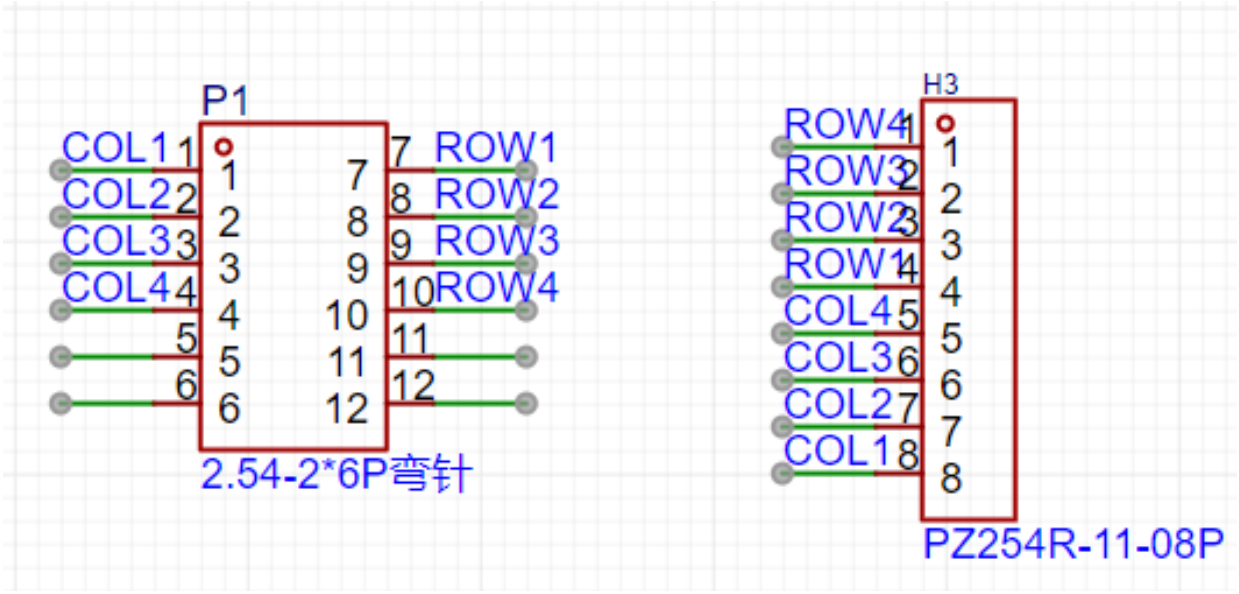

图 1.1: *image-20230918142454847*

PMOD1 接口在开发板中的位置如下图所示：

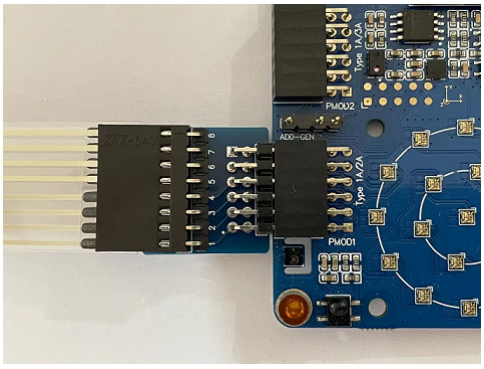

图 1.2: *image-20230918143358650*

1.3 软件说明

矩阵按键对应的单片机引脚定义。

```
#define PIN_COL1        GET_PIN(E,2)      // PE2:  COL1
#define PIN_COL2        GET_PIN(E,5)      // PE5 : COL2
#define PIN_COL3        GET_PIN(E,3)      // PE3 : COL3
#define PIN_COL4        GET_PIN(E,4)      // PE4:  COL4
#define PIN_ROW1        GET_PIN(A,4)      // PA4:  ROW1
#define PIN_ROW2        GET_PIN(A,7)      // PA7 : ROW2
#define PIN_ROW3        GET_PIN(A,6)      // PA6 : ROW3
#define PIN_ROW4        GET_PIN(A,5)      // PA5:  ROW4
```

按键输入的源代码位于 \projects\07_module_key_matrix/applications/main.c 中。流程如下，将四个行引脚放到一个数组中，四个列引脚放到另一个数组中，初始化引脚，行引脚设为输出模式，列引脚设为上拉输入模式，用双层 for 循环先扫描行再扫描列。返回键值，在主函数打印出按键按下对应的数值。

```
/* 定义列的四个IO在一个数组*/
unsigned int ROW_PINS[ROWS] = {PIN_ROW1, PIN_ROW2, PIN_ROW3, PIN_ROW4};
unsigned int COL_PINS[COLS] = {PIN_COL1, PIN_COL2, PIN_COL3, PIN_COL4};

/* 八个IO口模式设置  */
void key_scan_init()
{
    for (int i = 0; i < ROWS; i++)
    {
        /* 配置行IO为输出模式*/
        rt_pin_mode(ROW_PINS[i], PIN_MODE_OUTPUT);
        /* 初始化行IO为高电平 */
        rt_pin_write(ROW_PINS[i], PIN_HIGH);
    }
    for (int j = 0; j < COLS; j++)
    {
        /* 配置列IO为上拉输入模式*/
        rt_pin_mode(COL_PINS[j], PIN_MODE_INPUT_PULLUP);
    }
```

```
}
/* 按键扫描处理函数 */
int key_scan()
{
    /* 依次扫描行的四个IO */
    for (int row = 0 ; row < ROWS ; row++)
    {
        /* 扫描的行置低电平，其余为高电平 */
        rt_pin_write(ROW_PINS[row], PIN_LOW);
        /* 依次扫描列的四个IO */
        for (int COL = 0 ; COL < COLS ; COL++)
        {
            if (rt_pin_read(COL_PINS[COL]) == PIN_LOW)
            {
                rt_thread_mdelay(15);

                while (rt_pin_read(COL_PINS[COL]) == PIN_LOW);
                /* 返回扫描到的键值，键值和按键数值对应如下 */
                /* 1 2 3 A 4 5 6 B 7 8  9  C  *  0  #  D  */
                /* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 */
                return (row * COLS + COL) + 1;
            }
        }
        /*  扫描的行置高电平  */
        rt_pin_write(ROW_PINS[row], PIN_HIGH);
    }
    return -1;
}
/*  按键处理事件  */
int main(void)
{
    unsigned int count = 1;
    /* 按键初始化 */
    key_scan_init();

    while (count > 0)
    {
        int key = key_scan();
        switch (key)
        {
        case 1:
            LOG_D("1");
            break;
        case 2:
            LOG_D("2");
            break;
        case 3:
            LOG_D("3");
            break;
```

```
        case 4:
            LOG_D("A");
            break;
        case 5:
            LOG_D("4");
            break;
        case 6:
            LOG_D("5");
            break;
        case 7:
            LOG_D("6");
            break;
        case 8:
            LOG_D("B");
            break;
        case 9:
            LOG_D("7");
            break;
        case 10:
            LOG_D("8");
            break;
        case 11:
            LOG_D("9");
            break;
        case 12:
            LOG_D("C");
            break;
        case 13:
            LOG_D("*");
            break;
        case 14:
            LOG_D("0");
            break;
        case 15:
            LOG_D("#");
            break;
        case 16:
            LOG_D("D");
            break;
        default:
            break;
        }
        count++;
        rt_thread_mdelay(15);
    }
    return 0;
}
```

1.4 运行

1.4.1 编译 & 下载

- RT-Thread Studio：通过 RT-Thread Studio 导入工程，执行编译。
- MDK：首先双击 mklinks.bat，生成 rt-thread 与 libraries 文件夹链接；再使用 Env 生成 MDK5 工程；最后双击 project.uvprojx 打开 MDK5 工程，执行编译。

编译完成后，将开发板的 ST-Link USB 口与 PC 机连接，然后将固件下载至开发板。### 运行效果

按下复位按键重启开发板，在当按下按键，则在串口中打印出对应的数值。

此时可以在 PC 端使用终端工具打开开发板的 ST-Link 提供的虚拟串口，设置 115200 8 1 N 。开发板的运行日志信息即可实时输出出来。

```
[D/main] 1
[D/main] 2
[D/main] 3
[D/main] A
[D/main] 4
[D/main] 5
[D/main] 6
[D/main] B
[D/main] 7
[D/main] 8
[D/main] 9
[D/main] C
[D/main] *
[D/main] 0
[D/main] #
[D/main] D
```

1.5 注意事项

暂无。

1.6 引用参考

- 设备与驱动：PIN 设备

第 2 章

ENC28J60 以太网模块例程

2.1 简介

本例程的主要功能是让星火 1 号通过 ENC28J60 连接互联网。

2.2 硬件说明

ENC28J60 是带 SPI 接口的独立以太网控制器，兼容 IEEE 802.3，集成 MAC 和 10 BASE-T PHY，最高速度可达 10Mb/s。

ENC28J60 是通过板子上的 PMOD 插座连接单片机的，利用 SPI1 和单片机进行通讯。原理图和实物图如下所示：

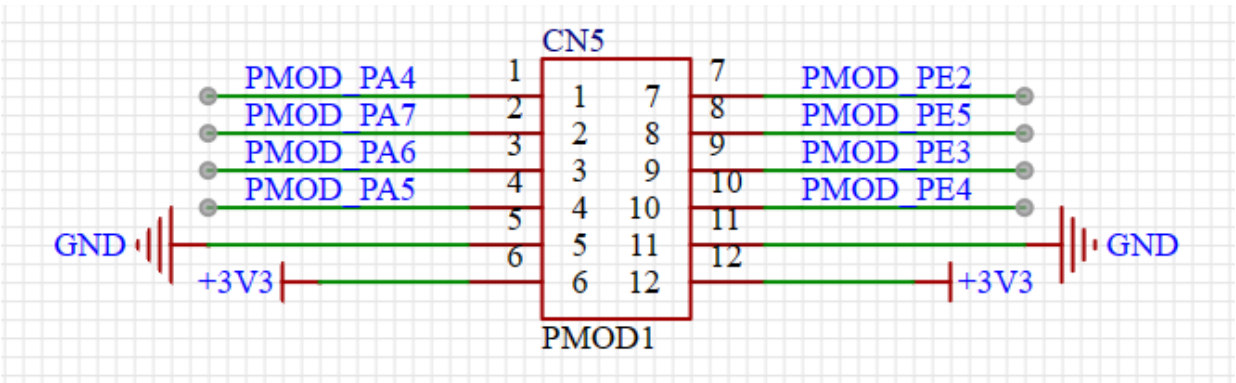

图 2.1: *ENC28J60 接口原理图*

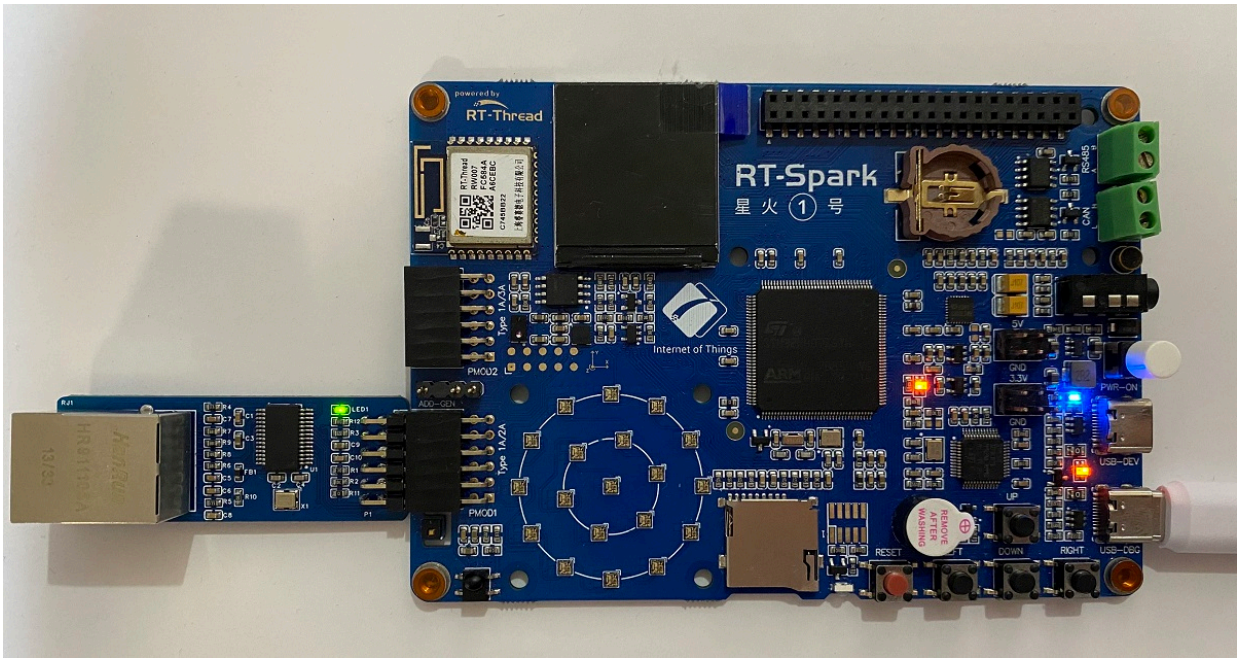

图 2.2: *ENC28J60 实物连接图*

2.3 软件说明

本例程的源码位于 /projects/07_module_spi_eth_enc28j60。ENC28J60 初始化的源代码位于 libraries/Board_Drivers/drv_enc28j60.c 中。

因为 ENC28J60 是通过 SPI 和单片机进行通讯的，所以需要通过 enc28j60_attach() 函数将 ENC28J60 连接到 SPI 设备上，星火 1 号提供了专门的接口，对应 SPI11 设备。这样，就能利用 RT-Thread 的 SPI 框架和 ENC28J60 进行通讯了。

然后就是将 ENC28J60 的中断处理函数通过 rt_pin_attach_irq() 函数绑定到对应的管脚上去，这里用到的是 **PE2** 号管脚。

最后利用 RT-Thread 的 INIT_COMPONENT_EXPORT 宏定义，将 enc28j60_init() 函数加入开机自动初始化。这样，板子上电后，就会自动执行 ENC28J60 的初始化函数，无需用户手动调用。

```
#include <rtdevice.h>
#include <enc28j60.h>
#include "drv_spi.h"
#include "board.h"

#define PIN_NRF_IRQ             GET_PIN(E,2)

int enc28j60_init(void)
{
    __HAL_RCC_GPIOD_CLK_ENABLE();
    rt_hw_spi_device_attach("spi1", "spi11", GPIOA, GPIO_PIN_4);

    /* attach enc28j60 to spi. spi11 cs - PA4 */
    enc28j60_attach("spi11");
```

```
    /* init interrupt pin */
    rt_pin_mode(PIN_NRF_IRQ, PIN_MODE_INPUT_PULLUP);
    rt_pin_attach_irq(PIN_NRF_IRQ, PIN_IRQ_MODE_FALLING, (void(*)(void *))
        enc28j60_isr, RT_NULL);
    rt_pin_irq_enable(PIN_NRF_IRQ, PIN_IRQ_ENABLE);

    return 0;
}
INIT_COMPONENT_EXPORT(enc28j60_init);
```

2.4 运行

2.4.1 编译 & 下载

- RT-Thread Studio：在 RT-Thread Studio 的包管理器中下载 STM32F407-RT-SPARK 资源包，然后创建新工程，执行编译。
- MDK：首先双击 mklinks.bat，生成 rt-thread 与 libraries 文件夹链接；再使用 Env 生成 MDK5 工程；最后双击 project.uvprojx 打开 MDK5 工程，执行编译。编译完成后，将开发板的 ST-Link USB 口与 PC 机连接，然后将固件下载至开发板。

编译完成后，将开发板的 ST-Link USB 口与 PC 机连接，然后将固件下载至开发板。

2.4.2 运行效果

按下复位按键重启开发板，可以看到板子会打印出如下信息：

```
 \ | /
- RT -     Thread Operating System
 / | \     4.1.1 build Jul  7 2023 14:54:52
 2006 - 2022 Copyright by RT-Thread team
lwIP-2.0.3 initialized!
[I/sal.skt] Socket Abstraction Layer initialize success.
```

在 msh 中输入 ping 命令

```
msh />ping www.rt-thread.org
60 bytes from 112.15.37.103 icmp_seq=0 ttl=55 time=126 ms
60 bytes from 112.15.37.103 icmp_seq=1 ttl=55 time=73 ms
60 bytes from 112.15.37.103 icmp_seq=2 ttl=55 time=38 ms
60 bytes from 112.15.37.103 icmp_seq=3 ttl=55 time=106 ms
```

2.5 注意事项

drv_enc28j60.c 里面并没有真正的初始化代码，只是调用了 RT_Thread 提供的 enc28j60.c 文件里的函数来初始化的，想了解 ENC28J60 初始化和中断函数详情的，可以查看 /rt-thread/components/drivers/spi/enc28j60.c 文件来学习。

2.6 引用参考

- 设备与驱动：PIN 设备
- 设备与驱动：SPI 设备
- 文档中心：RT-Thread 文档中心

第 3 章

超声波测距例程

3.1 简介

本例程主要功能是外接 SR-04 超声波模块实现测距功能。

3.2 硬件说明

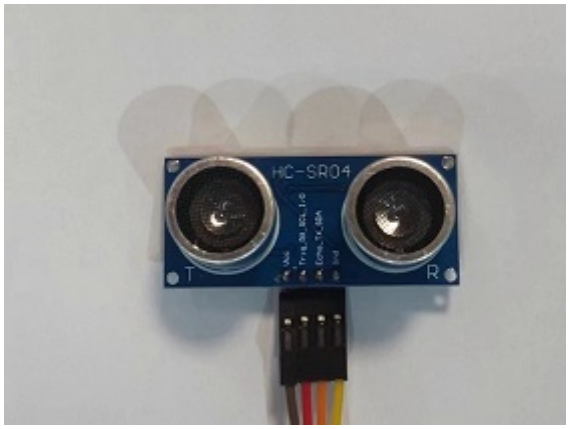

图 **3.1:** *SR-04 外观*

通过杜邦线将超声波模块接于 PMOD1 接口，Trig 引脚对应 PA5，Echo 引脚对应 PA6。

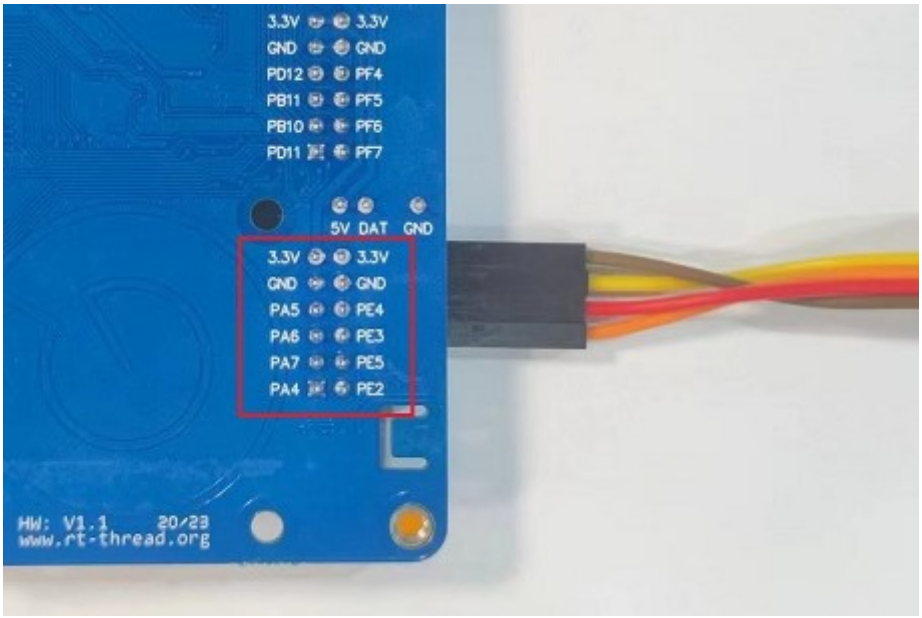

图 3.2: *连接图*

3.3 软件说明

超声波测距例程的示例代码位于 /projects/07_module_ultrasonic_sr04 下的 applications/ultrasonic.c 中，主要流程：初始化 GPIO 引脚。通过 MSH 启动测距代码，输入 ultrasonic run 启动测距，输入 ultrasonic pause 暂停测距。LED 闪烁代表处于工作状态。

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM3)
    {
        if (TIM3_CH1_Edge == 0) // 打开输入捕获
        {
            TIM3_CH1_Edge++;
                                                                    // 进入
                捕获下降沿状态
            __HAL_TIM_SET_CAPTUREPOLARITY(&htim3, TIM_CHANNEL_1,
                TIM_ICPOLARITY_FALLING); // 设置捕获极性为下降沿
            __HAL_TIM_SET_COUNTER(&htim3, 0);
                                                        // 设置定时器CNT计数器的
                值为0
        }
        else // 关闭定时器3
        {
            HAL_TIM_IC_Stop_IT(&htim3, TIM_CHANNEL_1);
                                                // 关闭定时器3
            TIM3_CH1_Edge++;
                                                                    // 进入到
                主函数状态
            TIM3_CH1_VAL = HAL_TIM_ReadCapturedValue(&htim3, TIM_CHANNEL_1);
```

```
                            // 读取捕获通道的值
            __HAL_TIM_SET_CAPTUREPOLARITY(&htim3, TIM_CHANNEL_1,
                TIM_ICPOLARITY_RISING); // 设置捕获极性为上降沿
        }
    }
}

void thread_entry(void *parameter)
{
    while (1)
    {
        rt_pin_write(PIN_Trig, PIN_HIGH);
        rt_hw_us_delay(10);
        rt_pin_write(PIN_Trig, PIN_LOW);

        rt_pin_write(PIN_LED_B, PIN_HIGH);
        rt_thread_delay(100);
        rt_pin_write(PIN_LED_B, PIN_LOW);
        rt_thread_delay(100);

        if (TIM3_CH1_Edge == 2)
        {
            TIM3_CH1_Edge = 0; // 复位状态计数值
            time = TIM3_CH1_VAL;
            distance = time * 342.62 / 2 / 10000; // distance=t*c/2 （c为室温在20摄
                氏度时声速）
            if (distance > 450)
                distance = 450;
            printf("The high leval last %d us    The distance=%.2f cm\r\n", time,
                distance);
            HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1); // 打开输入捕获
        }
        else
        {
            rt_kprintf("The ultrasonic module is not connected or connected
                incorrectly.\r\n");
            rt_thread_delay(5000);
        }
        if (rt_sem_take(ultrasonic_sem, 5) == RT_EOK)
        {
            break;
        }
    }
}
```

3.4 运行

3.4.1 编译 & 下载

- RT-Thread Studio：在 RT-Thread Studio 的包管理器中下载 STM32F407-RT-SPARK 资源包，然后创建新工程，执行编译。
- MDK：首先双击 mklinks.bat，生成 rt-thread 与 libraries 文件夹链接；再使用 Env 生成 MDK5 工程；最后双击 project.uvprojx 打开 MDK5 工程，执行编译。

编译完成后，将开发板的 ST-Link USB 口与 PC 机连接，然后将固件下载至开发板。

3.4.2 运行效果

输入ultrasonic run启动测距，输入ultrasonic pause暂停测距。

此时也可以在 PC 端使用终端工具打开开发板的 ST-Link 提供的虚拟串口，设置波率：115200，数据位：8，停止位：1 无校验。开发板的运行日志信息即可实时输出来。

```
msh >ultrasonic run
The high leval last 862 us    The distance=14.77 cm
The high leval last 717 us    The distance=12.28 cm
The high leval last 396 us    The distance=6.78 cm
The high leval last 205 us    The distance=3.51 cm
The high leval last 491 us    The distance=8.41 cm
The high leval last 780 us    The distance=13.36 cm
The high leval last 1017 us    The distance=17.42 cm
The high leval last 1168 us    The distance=20.01 cm
The high leval last 1369 us    The distance=23.45 cm
The high leval last 1744 us    The distance=29.88 cm
The high leval last 2167 us    The distance=37.12 cm
The high leval last 1927 us    The distance=33.01 cm
The high leval last 2050 us    The distance=35.12 cm
The high leval last 2941 us    The distance=50.38 cm
The high leval last 10518 us    The distance=180.18 cm
The high leval last 10540 us    The distance=180.56 cm
The high leval last 10540 us    The distance=180.56 cm
The high leval last 10540 us    The distance=180.56 cm
The high leval last 10516 us    The distance=180.15 cm
The high leval last 10513 us    The distance=180.10 cm
The high leval last 10513 us    The distance=180.10 cm
The high leval last 10512 us    The distance=180.08 cm
```

3.5 引用参考

- 设备与驱动：PIN 设备