项目 5

智 能 安 防

本项目通过鸿蒙 App 控制 Hi3861 开发板，连接 HC-SR-04 超声波传感器，实现智能安防系统。

项目 5

5.1 总体设计

本部分包括系统架构和系统流程。

5.1.1 系统架构

系统架构如图 5-1 所示，Hi3861 开发板与外设引脚连线如表 5-1 所示。

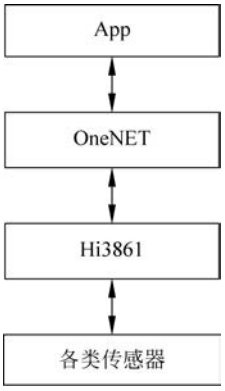

图 5-1 系统架构

表 5-1 Hi3861 开发板与外设引脚连线

Hi3861 开发板	HC-SR04 超声波	光敏传感器	蜂鸣器	OLED 串口屏
5V	VCC	VCC	VCC	VCC
GND	GND	GND	GND	GND
GPIO11	Trig	/	/	/
GPIO12	Echo	/	/	/
GPIO10	/	DO	/	/
GPIO5	/	/	I/O	/
GPIO13	/	/	/	SCL
GPIO14	/	/	/	SDA
板载 LED	/	/	/	

5.1.2 系统流程

系统流程如图 5-2 所示。

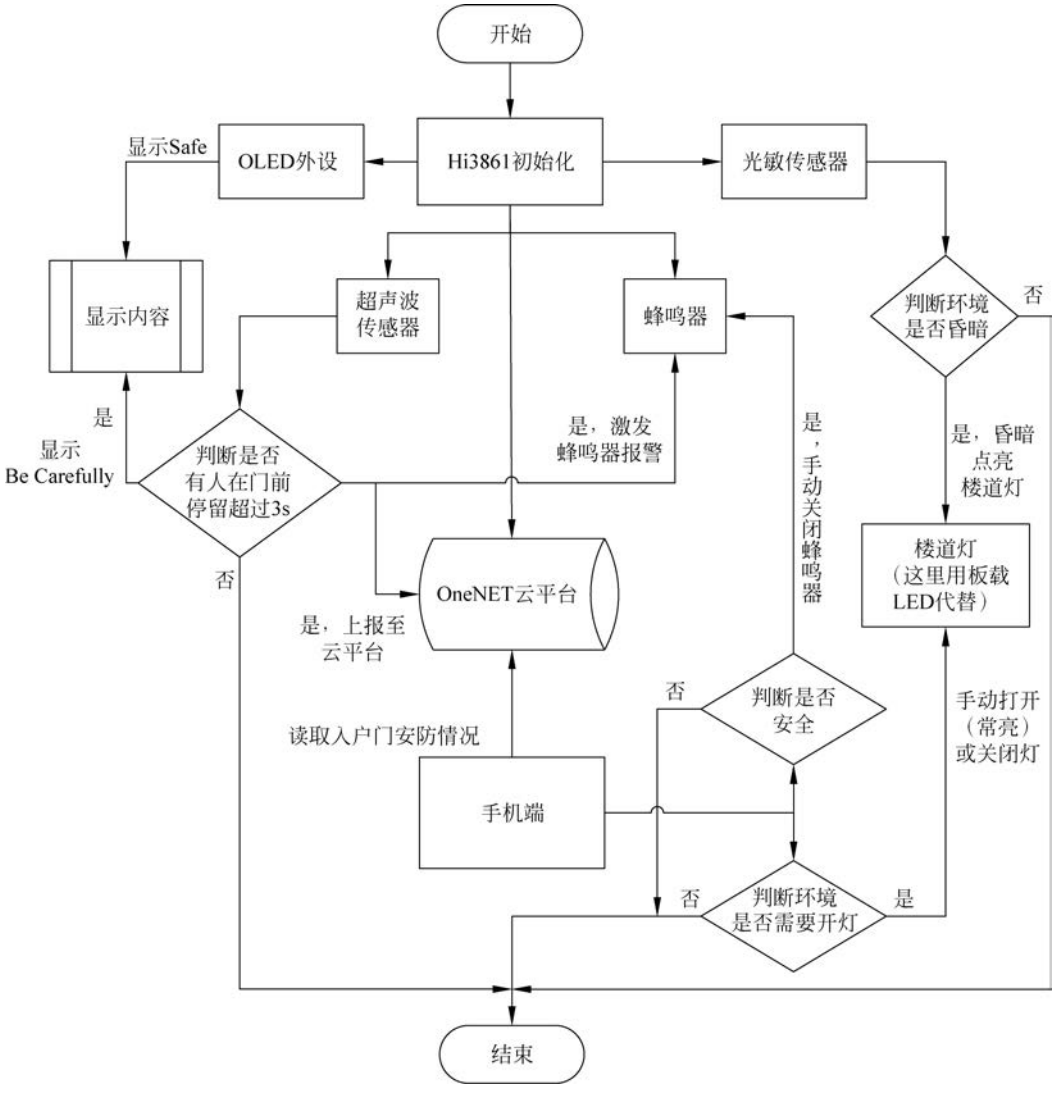

图 5-2 系统流程

5.2 模块介绍

本项目由 VSCode 和 DevEco Studio 开发，包括超声波测距、光敏感知、蜂鸣器控制、OLED 显示、WiFi 模块、OneNET 云平台和前端模块。下面分别给出各模块的功能介绍及相关代码。

5.2.1 超声波测距

该模块主要利用超声波模块进行测距，将 GPIO11 设置为输出，每个周期给 Trig 引脚发送 20～50μs 的脉冲，激发超声波模块发出 8 组超声波，当接收到返回信号时，Echo 产生相应

的高电平脉冲，脉冲的宽度、声波与返回时间有关，之后便可以计算距离，相关代码如下：

```
float GetDistance  (void)
{
    static unsigned long start_time = 0, time = 0;
    float distance = 0.0;
    IotGpioValue value = IOT_GPIO_VALUE0;
    unsigned int flag = 0;
    //超声波传感器 Trig 为 GPIO11,Echo 为 GPIO12
    IoTGpioSetDir(HI_IO_NAME_GPIO_12, IOT_GPIO_DIR_IN);
    IoTGpioSetDir(HI_IO_NAME_GPIO_11, IOT_GPIO_DIR_OUT);
    //触发 Trig
    IoTGpioSetOutputVal(HI_IO_NAME_GPIO_11, IOT_GPIO_VALUE1);
    hi_udelay(20);
    IoTGpioSetOutputVal(HI_IO_NAME_GPIO_11, IOT_GPIO_VALUE0);
    //检测 Echo 高电平时间
    while (1) {
        IoTGpioGetInputVal(HI_IO_NAME_GPIO_12, &value);
        if ( value == IOT_GPIO_VALUE1 && flag == 0) {
            start_time = hi_get_us();
            flag = 1;
        }
        if (value == IOT_GPIO_VALUE0 && flag == 1) {
            time = hi_get_us() -start_time;
            start_time = 0;
            break;
        }
    }
    //计算距离
    distance = time *0.034 / 2;
    printf("[Distance]:%f.\r\n",distance);
    return distance;
}
```

5.2.2 光敏感知

通过光敏电阻感知外界光强，当外界光强低于某个阈值时，则传感器的 DO 口会给出一个高电平，根据是否接收到高电平判断板载 LED 亮灭。

```
#define LED_TEST_GPIO 9                          //使用 hispark_pegasus 开发板
void GetLight  (void)
{
    IotGpioValue value = IOT_GPIO_VALUE0;
    //初始化 LED 的 GPIO
    IoTGpioInit(LED_TEST_GPIO);
    //设置为输入/输出
    IoTGpioSetDir(LED_TEST_GPIO, IOT_GPIO_DIR_OUT);
    //设置光敏传感器的 DO 口
    IoTGpioSetDir(HI_IO_NAME_GPIO_10, IOT_GPIO_DIR_IN);
        if(led_flag==2){
        IoTGpioSetDir(LED_TEST_GPIO, 1);     //LED 亮
    }
    else{
        IoTGpioGetInputVal(HI_IO_NAME_GPIO_10, &value);
        if ( value == IOT_GPIO_VALUE1 ) {
```

```
            IoTGpioSetDir(LED_TEST_GPIO, 1); //LED亮
            led_flag = 1;
            printf("on\n");
        }
        if (value == IOT_GPIO_VALUE0 ) {
            IoTGpioSetDir(LED_TEST_GPIO, 0); //LED灭
            led_flag = 0;
            printf("off\n");
        }
    }
}
```

5.2.3 蜂鸣器控制

通过简单的高低电平控制声响。

```
void BeepRing(void)
{
    //初始化 GPIO
    IoTGpioInit(HI_IO_NAME_GPIO_5);
    //设置为输出
    IoTGpioSetDir(HI_IO_NAME_GPIO_5, IOT_GPIO_DIR_OUT);
    //GPIO 复用
    hi_io_set_func(HI_IO_NAME_GPIO_5,HI_IO_FUNC_GPIO_5_GPIO);
    //输出低电平
    IoTGpioSetOutputVal(HI_IO_NAME_GPIO_5,0);
    beep_flag=0;
    printf("BeepRing\n");
}
void BeepStop(void)
{
    //初始化 GPIO
    IoTGpioInit(HI_IO_NAME_GPIO_5);
    //设置为输出
    IoTGpioSetDir(HI_IO_NAME_GPIO_5, IOT_GPIO_DIR_OUT);
    //GPIO 复用
    hi_io_set_func(HI_IO_NAME_GPIO_5,HI_IO_FUNC_GPIO_5_GPIO);
    //输出高电平
    IoTGpioSetOutputVal(HI_IO_NAME_GPIO_5,1);
    beep_flag=1;
    printf("BeepStop\n");
}
```

5.2.4 OLED 显示

SSD1306 逻辑功能代码如下：

```
void OLED(void)
{
    IoTGpioInit(HI_IO_NAME_GPIO_13);
    IoTGpioInit(HI_IO_NAME_GPIO_14);
    hi_io_set_func(HI_IO_NAME_GPIO_13, HI_IO_FUNC_GPIO_13_I2C0_SDA);
    hi_io_set_func(HI_IO_NAME_GPIO_14, HI_IO_FUNC_GPIO_14_I2C0_SCL);
    IoTI2cInit(0, OLED_I2C_BAUDRATE);
    usleep(20*1000);
```

```
    ssd1306_Init();
    ssd1306_Fill(Black);
    ssd1306_SetCursor(0, 0);
    if(distance_flag==0){
        ssd1306_DrawString("Be Carefully!", Font_11x18, White);
        ssd1306_UpdateScreen();
        printf("test1");
    }
    else{
        ssd1306_DrawString("Safe", Font_11x18, White);
        ssd1306_UpdateScreen();
        printf("test2");
    }

}
```

5.2.5　WiFi 模块

WiFi 连接相关代码如下：

```
void wifi_wpa_event_cb(const hi_wifi_event *hisi_event)
{
   if (hisi_event == NULL)
      return;
   switch (hisi_event->event) {
    case HI_WIFI_EVT_SCAN_DONE:
        printf("WiFi: Scan results available\n");
        break;
    case HI_WIFI_EVT_CONNECTED:
        printf("WiFi: Connected\n");
        netifapi_dhcp_start(g_lwip_netif);
        wifi_ok_flg = 1;
      break;
   case HI_WIFI_EVT_DISCONNECTED:
      printf("WiFi: Disconnected\n");
      netifapi_dhcp_stop(g_lwip_netif);
      hi_sta_reset_addr(g_lwip_netif);
      break;
   case HI_WIFI_EVT_WPS_TIMEOUT:
      printf("WiFi: wps is timeout\n");
      break;
   default:
      break;
 }
}
int hi_wifi_start_connect(void)
{
 int ret;
 errno_t rc;
 hi_wifi_assoc_request assoc_req = {0}
 rc = memcpy_s(assoc_req.ssid, HI_WIFI_MAX_SSID_LEN + 1, "OCF24", 8);
  if (rc != EOK) {
  return -1;
  }
   //热点加密方式
```

```
 assoc_req.auth = HI_WIFI_SECURITY_WPA2PSK;
 //热点密码
 memcpy(assoc_req.key, "xxxxxxxx", 8);
 ret = hi_wifi_sta_connect(&assoc_req);
 if (ret != HISI_OK) {
    return -1;
  }
 return 0;
}
int hi_wifi_start_sta(void)
{
  int ret;
  char ifname[WIFI_IFNAME_MAX_SIZE + 1] = {0};
  int len = sizeof(ifname);
  const unsigned char wifi_vap_res_num = APP_INIT_VAP_NUM;
  const unsigned char wifi_user_res_num = APP_INIT_USR_NUM;
  unsigned int  num = WIFI_SCAN_AP_LIMIT;
  ret = hi_wifi_sta_start(ifname, &len);
  if (ret != HISI_OK) {
    return -1;
   }
  ret = hi_wifi_register_event_callback(wifi_wpa_event_cb);
  if (ret != HISI_OK) {
    printf("register wifi event callback failed\n");
  }
    g_lwip_netif = netifapi_netif_find(ifname);
    if (g_lwip_netif == NULL) {
    printf("%s: get netif failed\n", __FUNCTION__);
    return -1;
    }
   //开始扫描附近的 WiFi 热点
   ret = hi_wifi_sta_scan();
   if (ret != HISI_OK) {
   return -1;
   }
  sleep(5);
  hi_wifi_ap_info *pst_results = malloc(sizeof(hi_wifi_ap_info) *WIFI_SCAN_AP_
LIMIT);
  if (pst_results == NULL) {
    return -1;
  }
  //将扫描到的热点结果进行存储
  ret = hi_wifi_sta_scan_results(pst_results, &num);
  if (ret != HISI_OK) {
     free(pst_results);
     return -1;
  }
  //打印扫描到的所有热点
  for (unsigned int loop = 0; (loop < num) && (loop < WIFI_SCAN_AP_LIMIT); loop++) {
    printf("SSID: %s\n", pst_results[loop].ssid);
  }
  free(pst_results);
  //开始接入热点
  ret = hi_wifi_start_connect();
  if (ret != 0) {
```

```
        return -1;
    }
    return 0;
}
void wifi_sta_task(void *arg)
{
 arg = arg;
 //连接热点
 hi_wifi_start_sta();
  while(wifi_ok_flg == 0)
 {
   usleep(30000);
 }
usleep(2000000);
```

5.2.6　OneNET 云平台

本部分包括创建账号、创建产品、添加设备和相关代码。

1. 创建账号

登录网页 https://open.iot.10086.cn/passport/reg/，按要求填写注册信息后进行实名认证。

2. 创建产品

进入 Studio 平台后，在全部产品服务中选择多协议接入。单击“添加产品”按钮，在弹出页面中按照提示填写产品信息。本项目采用 MQTT 协议接入，如图 5-3 所示。

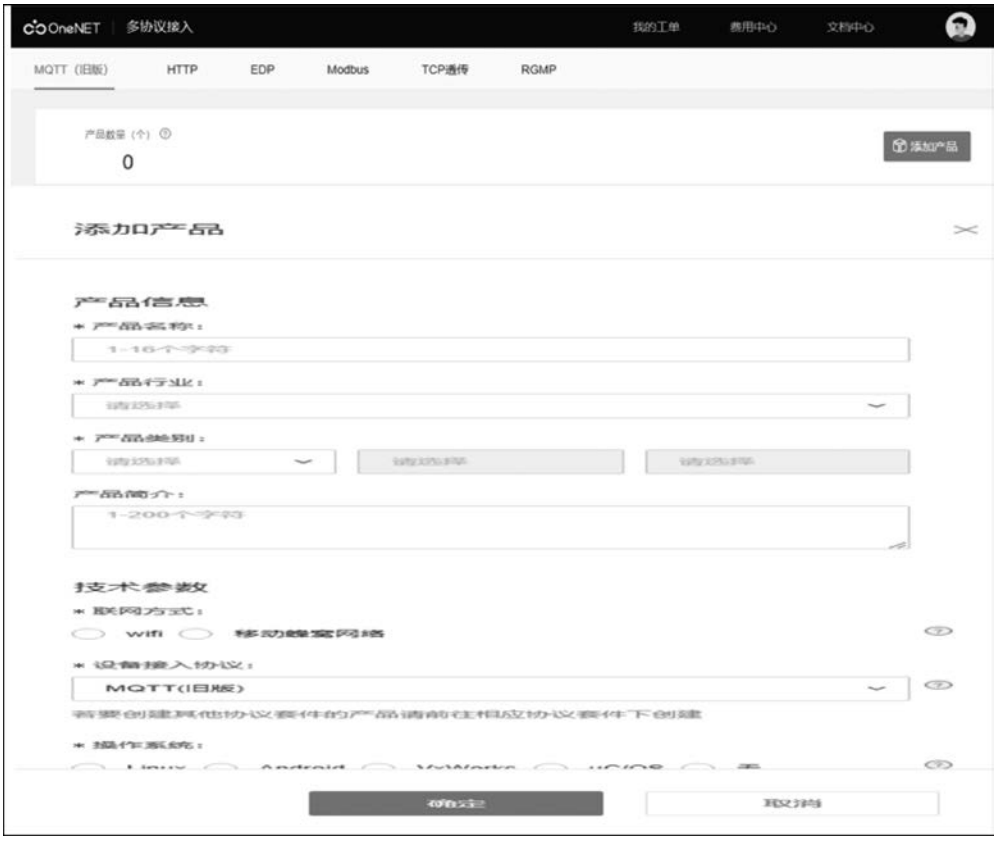

图 5-3　创建产品

3. 添加设备

单击“设备管理”，选择“添加设备”，按照提示填写相关信息，如图 5-4 所示。

图 5-4　添加设备

4. 相关代码

下面给出实现连接 OneNET 云平台及整个系统的逻辑控制代码。

```
void onenet_cmd_rsp_cb(uint8_t *recv_data, size_t recv_size, uint8_t **resp_data,
size_t *resp_size)
{
    printf("recv data is %.*s\n", recv_size, recv_data);
    strcpy(temp1, (char *)(recv_data));      //暂存 recv data
    int messageCode = (temp1[2] -'0') ;
    printf("messageCode is %d\n", messageCode);
    if(messageCode==0){
        BeepStop();
        printf("Beep\n");
    }
    if(messageCode==1){
        IoTGpioSetDir(LED_TEST_GPIO, 1);     //LED 亮
        led_flag = 2;
        printf("LED ON\n");
    }
    if(messageCode==2){
        IoTGpioSetDir(LED_TEST_GPIO, 1);     //LED 灭
        led_flag = 0;
        printf("LED OFF\n");
    }
    *resp_data = NULL;
    *resp_size = 0;
}
int onenet_test(void)
{
        device_info_init(ONENET_INFO_DEVID, ONENET_INFO_PROID, ONENET_INFO_AUTH,
ONENET_INFO_APIKEY, ONENET_MASTER_APIKEY);
    onenet_mqtt_init();
```

```
    onenet_set_cmd_rsp_cb(onenet_cmd_rsp_cb);
    IoTWatchDogDisable();
    while (1)
    {
        GetLight();
        temp = GetDistance();
        OLED();
        //连续判断,防止错判
        if(temp<3){
            count++;
        }
        else{
            count=0;
        }
        if(temp<3&&count>3){
            BeepRing();                          //蜂鸣器响
            printf("[Beep]:%d.\r\n",beep_flag);
            distance_flag=0;
            printf("[person]:%d.\r\n",distance_flag);
        }
        else{
            BeepStop();                          //蜂鸣器不响
            printf("[person]:%d.\r\n",distance_flag);
            distance_flag=1;
        }
        //上传至 OneNET
        if (onenet_mqtt_upload_digit("person", distance_flag) < 0)
        {
            printf("upload has an error, stop uploading");
            //break;
        }
        else
        {
            printf("buffer : {\"person\":%d} \r\n", distance_flag);
        }
        sleep(1);
    }
    return 0;
}
```

5.2.7　前端模块

本部分包括界面设计和发送 POST 请求。

1. 界面设计

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:alignment="horizontal_center"
    ohos:orientation="vertical">
    <Text
        ohos:id="$+id:text_type_get"
        ohos:height="60vp"
```

```
        ohos:width="match_parent"
        ohos:text_alignment="center"
        ohos:layout_alignment="horizontal_center"
        ohos:text="GET 请求"
        ohos:text_size="30vp"
        />
    <Text
        ohos:id="$+id:text_type_post"
        ohos:height="60vp"
        ohos:text_alignment="center"
        ohos:width="match_parent"
    ohos:background_element="#ed6262"
        ohos:layout_alignment="horizontal_center"
        ohos:text="POST 请求"
        ohos:text_size="30vp"
        />
    <ScrollView
        ohos:height="match_parent"
        ohos:width="match_parent"
        >
        <DirectionalLayout
            ohos:height="match_parent"
            ohos:width="match_parent">
            <Text
                ohos:id="$+id:text_result"
                ohos:height="match_parent"
                ohos:multiple_lines="true"
                ohos:text_alignment="center"
                ohos:width="match_parent"
                ohos:layout_alignment="horizontal_center"
                ohos:text="显示结果"
                ohos:text_size="20vp"
                />
</DirectionalLayout>
</ScrollView>
</DirectionalLayout>
```

2. 发送 POST 请求

```
package com.example.afinal.slice;
import com.example.afinal.ResourceTable;
import ohos.aafwk.ability.AbilitySlice;
import ohos.aafwk.content.Intent;
import ohos.agp.components.Component;
import ohos.agp.components.Text;
import okhttp3.*;
import java.io.IOException;
public class MainAbilitySlice extends AbilitySlice {
    //private static AbilityForm textResult;
    Text TextGet, TextPost, textResult;
    private boolean status;
    final String cloud_url =
            "http://api.heclouds.com/cmds?device_id=962845387";
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
```

```
        super.setUIContent(ResourceTable.Layout_ability_main);
        TextGet = findComponentById(ResourceTable.Id_text_type_get);
        TextPost = findComponentById(ResourceTable.Id_text_type_post);
        textResult = findComponentById(ResourceTable.Id_text_result);
        TextGet.setClickedListener(new Component.ClickedListener() {
            @Override
            public void onClick(Component component) {
                OkHttpClient client = new OkHttpClient();
                Request.Builder requestBuilder = new Request.Builder();
                requestBuilder. addHeader ( " api - key "," CT0J7NvniU4E5hsdH4emD =
FzKs0=");
                 HttpUrl.Builder urlBuilder = HttpUrl.parse("http://api.heclouds.
 com/devices/962845387/datapoints").newBuilder();   //todo 接口链接
                urlBuilder.addQueryParameter("datastream_id", "temperature");
                                                                    //参数
                urlBuilder.addQueryParameter("limit","4");   //密钥
                requestBuilder.url(urlBuilder.build());
                Call call = client.newCall(requestBuilder.build());
                call.enqueue(new Callback() {
                    @Override
                    public void onFailure(Call call, IOException e) {
                        //todo 失败回调需要回到主线程显示结果
                        getUITaskDispatcher().asyncDispatch(new Runnable()
 {
                            @Override
                            public void run() {
                                textResult.setText(e.getMessage());
                            }
                        });
                    }
                    @Override
                    public void onResponse(Call call, Response response) throws
 IOException {
                        if (response.isSuccessful()) {
                            String result = response.body().string();
                            //处理界面需要切换到界面线程处理
                            getUITaskDispatcher().asyncDispatch(new Runnable() {
                                @Override
                                public void run() {
                                    textResult.setText(result);
                                }
                            });
                        }
                    }
                });
            }
        });
        TextPost.setClickedListener(new Component.ClickedListener() {
            @Override
            public void onClick(Component component) {
                textResult.setText("post can be pressed");
                OkHttpClient client = new OkHttpClient();
                FormBody body = new FormBody.Builder()
                        .add("temperature", "77")//日期参数
                        .build();
```

```
                Request request = new Request.Builder()
                        .header("api-key", "CT0J7NvniU4E5hsdH4emD=FzKs0=")
                        .url("http://api.heclouds.com/cmds?device_id=962845387")
                        .post(body)
                        .build();
                Call call = client.newCall(request);
                call.enqueue(new Callback() {
                    @Override
                    public void onFailure(Call call, IOException e) {
                        //失败回调需要回到主线程显示结果
                        getUITaskDispatcher().asyncDispatch(new Runnable()
{
                            @Override
                            public void run() {
                                textResult.setText(e.getMessage());
                            }
                        });
                    }
                    @Override
                    public void onResponse(Call call, Response response) throws
IOException {
                        if (response.isSuccessful()) {
                            String result = response.body().string();
                            //处理界面需要切换到界面线程处理
                            //失败回调需要回到主线程显示结果
                            getUITaskDispatcher().asyncDispatch(new Runnable() {
                                @Override
                                public void run() {
                                    textResult.setText(result);
                                }
                            });
                        }
                    }
                });
            }
        });
    }
```

5.3 产品展示

Hi3861 开发板的实现效果如图 5-5 所示，串口监视器效果如图 5-6 所示，鸿蒙 App 的实现效果如图 5-7 所示。

图 5-5 Hi3861 开发板实现效果

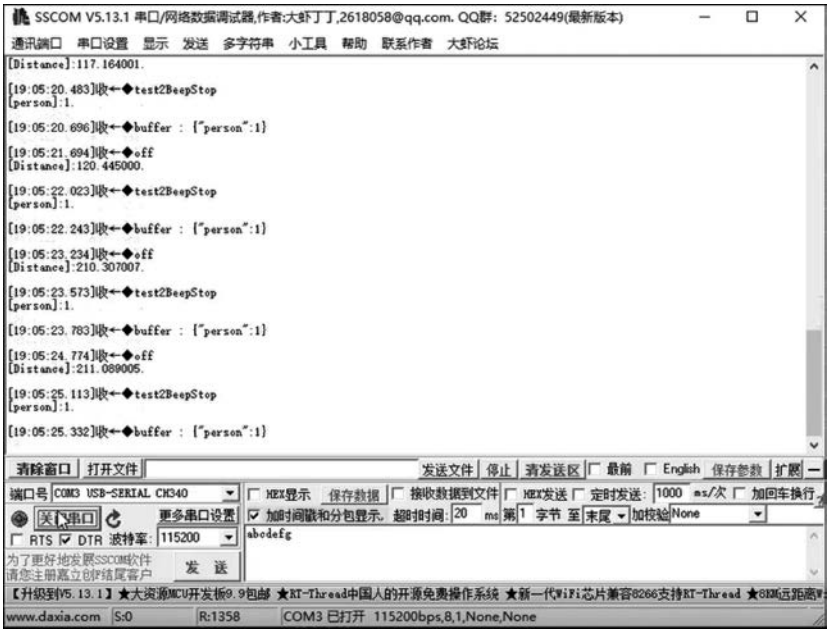

图 5-6　串口监视器效果

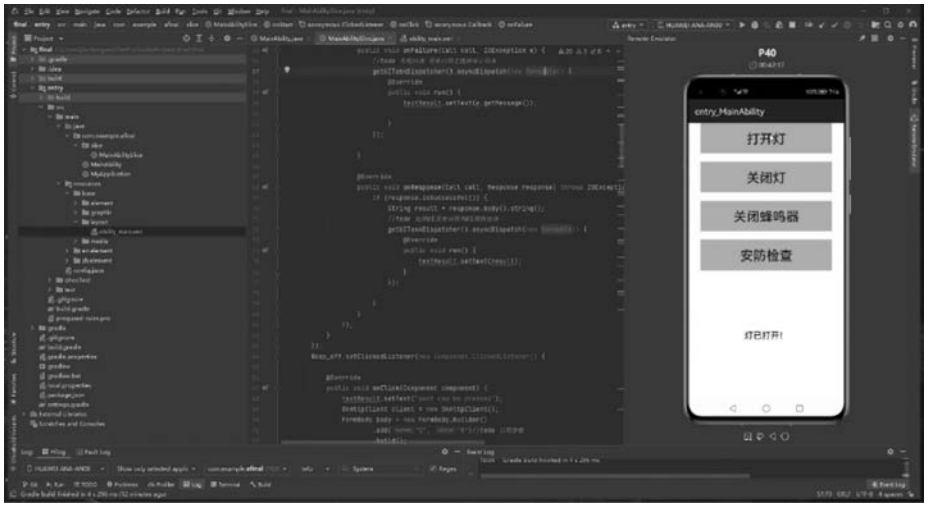

图 5-7　App 页面效果

5.4　元件清单

完成本项目所需的元件及数量如表 5-2 所示。

表 5-2　元件清单

元件/测试仪表	数　量	元件/测试仪表	数　量
面包板	1 个	光敏模块	1 个
Hi3861	1 个	蜂鸣器	1 个
HC-SR04 超声波传感器	1 个	OLED 串口屏	1 个