基于 ble 的手势控制 PPT

1 介绍

两块开发板通过 ble 来传递控制信息，一块开发板连接 PAJ7620，通过 iic 总线提供手势信息，另一块开发板使用 ble 和 USB HID，ble 用来接收数据，USB HID 则模拟键盘输入从而控制 ppt

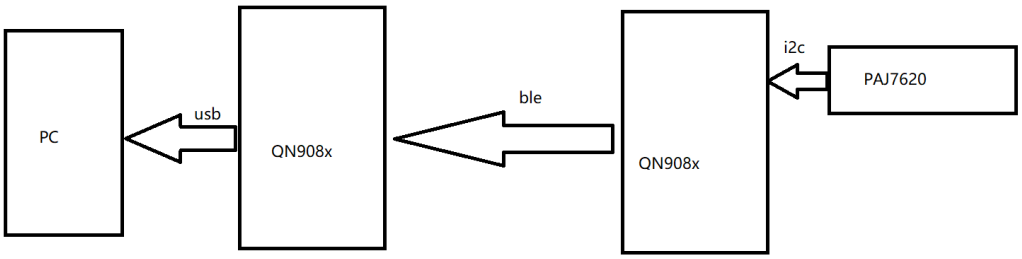

图 1

2 准备

两块开发板 QN908x, 手势控制 paj7620, IDE 使用 IAR, 使用的例程 temperature_sensor，以及 temperature_collector，sdk 版本是 2.2.3

3 代码

3.1 temperature_sensor 代码的编写

我们要实现 iic 读取手势信息以及将数据发送。

iic 使用的引脚是 PA6 和 PA7

将代码中的 iic 读写代码进行简单封装，创建 i2c_operation.c 和其对应的头文件。在里面实现 iic 初始化，以及读写寄存器函数

```
bool LPI2C_ReadRegs(I2C_Type *base, uint8_t device_addr,
                        uint8_t reg_addr, uint8_t *rxBuff,
                        uint32_t rxSize)
{
    i2c_master_transfer_t masterXfer;
    status_t reVal = kStatus_Fail;

    memset(&masterXfer, 0, sizeof(masterXfer));
    masterXfer.slaveAddress = device_addr;
    masterXfer.direction = kI2C_Read;
    masterXfer.subaddress = reg_addr;
    masterXfer.subaddressSize = 1;
    masterXfer.data = rxBuff;
    masterXfer.dataSize = rxSize;
    masterXfer.flags = kI2C_TransferDefaultFlag;

    /*  direction=write : start+device_write;cmdbuff;xBuff; */
    /*  direction=recive : start+device_write;cmdbuff;repeatStart+device_read;xBuff; */

    reVal = I2C_MasterTransferNonBlocking(BOARD_ACCEL_I2C_BASEADDR, &g_m_handle, &masterXfer);
    if (reVal != kStatus_Success)
    {
        return false;
    }
    /*  wait for transfer completed. */
    while ((!nakFlag) && (!completionFlag))
    {
    }

    nakFlag = false;

    if (completionFlag == true)
    {
        completionFlag = false;
        return true;
    }
    else
    {
        return false;
    }
}
```

图 2 读寄存器

```
bool LPI2C_WriteReg(I2C_Type *base, uint8_t device_addr, uint8_t reg_addr, uint8_t value)
{
    i2c_master_transfer_t masterXfer;
    status_t reVal = kStatus_Fail;

    memset(&masterXfer, 0, sizeof(masterXfer));

    masterXfer.slaveAddress = device_addr;
    masterXfer.direction = kI2C_Write;
    masterXfer.subaddress = reg_addr;
    masterXfer.subaddressSize = 1;
    masterXfer.data = &value;
    masterXfer.dataSize = 1;
    masterXfer.flags = kI2C_TransferDefaultFlag;

    /*  direction=write : start+device_write;cmdbuff;xBuff; */
    /*  direction=recive : start+device_write;cmdbuff;repeatStart+device_read;xBuff; */

    reVal = I2C_MasterTransferNonBlocking(BOARD_ACCEL_I2C_BASEADDR, &g_m_handle, &masterXfer);
    if (reVal != kStatus_Success)
    {
        return false;
    }

    /*  wait for transfer completed. */
    while ((!nakFlag) && (!completionFlag))
    {
    }

    nakFlag = false;

    if (completionFlag == true)
    {
        completionFlag = false;
        return true;
    }
    else
    {
        return false;
    }
}
```

图 3 写寄存器

3.1.1 拥有这些函数以后开始编写手势识别的代码，首先添加两个空白文件 paj7620.c 和 paj7620.h

选择 bank 寄存器区域函数

```
void paj7620u2_selectBank(bank_e bank)
{
    switch(bank)
    {
        case BANK0: LPI2C_WriteReg(I2C0,PAJ7620_ID,PAJ_REGITER_BANK_SEL,PAJ_BANK0);break;//BANK0 area
        case BANK1: LPI2C_WriteReg(I2C0,PAJ7620_ID,PAJ_REGITER_BANK_SEL,PAJ_BANK1);break;//BANK1 area
    }
}
```

图 4

唤醒 paj7620 读取设备状态函数

```
//PAJ7620U2 wakeup
uint8_t paj7620u2_wakeup(void)
{
    uint8_t data = 0;
    paj7620u2_selectBank(PAJ_BANK0);
    LPI2C_ReadRegs(I2C0,PAJ7620_ID,0,&data,1);
    if(data!=0x20)
        return 0;
    return 1;
}
```

图 5

初始化设备

```
//PAJ7620U2 init
//return: 0:fail 1:success
uint8_t paj7620u2_init(void)
{
    uint8_t status;
    uint32_t i;
    status = paj7620u2_wakeup();//wakeup PAJ7620U2
    if(!status)
        return 0;
    paj7620u2_selectBank(BANK0);//enter BANK0 area
    for(i=0;i<INIT_SIZE;i++)
    {
        LPI2C_WriteReg(I2C0,PAJ7620_ID,init_Array[i][0],init_Array[i][1]);
    }
    paj7620u2_selectBank(BANK0);//
    return 1;
}
```

图 6

手势测试函数

```
uint16_t Gesture_test(void)
{
    uint8_t status;
    uint8_t i;
    uint8_t index;
    uint8_t data[2]={0x00};
    uint16_t gesture_data;
    if(paj_init == false)
    {
        paj7620u2_selectBank(BANK0);//enter BANK0 area
        for(i=0;i<GESTURE_SIZE;i++)
        {
            status = LPI2C_WriteReg(I2C0,PAJ7620_ID,gesture_arry[i][0],gesture_arry[i][1]);
            if(status == true)
            {
              index++;
            }
        }
        if(index == GESTURE_SIZE)
        {
         //   AppPrintString("write ok\r\n");
        }
        paj7620u2_selectBank(BANK0);//
        paj_init = true;
    }

    status = LPI2C_ReadRegs(I2C0,PAJ7620_ID,PAJ_GET_INT_FLAG1,data,2);
    if(status)
    {
        gesture_data =(uint16_t)data[1]<<8 | data[0];
        return gesture_data;
    }
    return 0;
}
```

图 7

3.1.2 准备好读取设备信息以后，在 BleApp_Init 函数中初始化 iic 和 paj7620

```
/*****************************************************************************
void BleApp_Init(void)
{
    /* Initialize application support for drivers */
    BOARD_InitAdc();
    I2C_init(BOARD_ACCEL_I2C_BASEADDR);
    if(paj7620u2_init())
    {
         AppPrintString("PAJ7620 init success.\r\n");
    }
```

图 8

原则上我们需要为 PAJ 设备创建一个自定义的服务，但是我们节省一下，替换例程中的温度数据作为我们手势控制的数据。如果要创建自定义服务参考该链接。[custom profile]

3.1.3 创建一个定时器，这个定时器定时发送手势数据。

在 temerature_sensor.c 文件中，
定义定时器 ID，static tmrTimerID_t dataTimerId;
分配定时器 dataTimerId = TMR_AllocateTimer();
定义定时器的回调函数。

```
static void dataTimerCallback(void* pParam)
{
    uint16_t gesture_data = Gesture_test();
    if(gesture_data)
    {
        (void)Tms_RecordTemperatureMeasurement((uint16_t)service_temperature, gesture_data*100);
    }
}
```

图 9

启动定时器当链接成功以后

```
static void BleApp_ConnectionCallback (deviceId_t peerDeviceId, gapConnectionEvent_t* pConnectionEvent)
{
    /* Connection Manager to handle Host Stack interactions */
    BleConnManager_GapPeripheralEvent(peerDeviceId, pConnectionEvent);

    switch (pConnectionEvent->eventType)
    {
        case gConnEvtConnected_c:
        {
            /* Advertising stops when connected */
            mAdvState.advOn = FALSE;
            (void)TMR_StopTimer(appTimerId);

            /* Subscribe client*/
            mPeerDeviceId = peerDeviceId;
            (void)Bas_Subscribe(&basServiceConfig, peerDeviceId);
            (void)Tms_Subscribe(peerDeviceId);

            AppPrintString("Connected!\r\n");
//                (void)TMR_StartLowPowerTimer(appTimerId,
//                          gTmrLowPowerSecondTimer_c,
//                          TmrSeconds(gGoToSleepAfterDataTime_c),
//                          DisconnectTimerCallback, NULL);
            TMR_StartLowPowerTimer(dataTimerId,gTmrLowPowerIntervalMillisTimer_c,
                                   TmrMilliseconds(1),
                                   dataTimerCallback,NULL);
            /* Set low power mode */
```

图 10

将低功耗关闭，#define cPWR_UsePowerDownMode 0

这样服务器端代码就完成编写。

3.2 temperature_collector 代码编写

在这里最重要的是将 USB HID 移植进这里。我们使用的 usb 例程是 usb 键盘鼠标的例程。

3.2.1 把例程下的 osa，usb 文件夹加入工程目录，并且按照原例程的文件结构，将文件拷贝到对应文件夹。

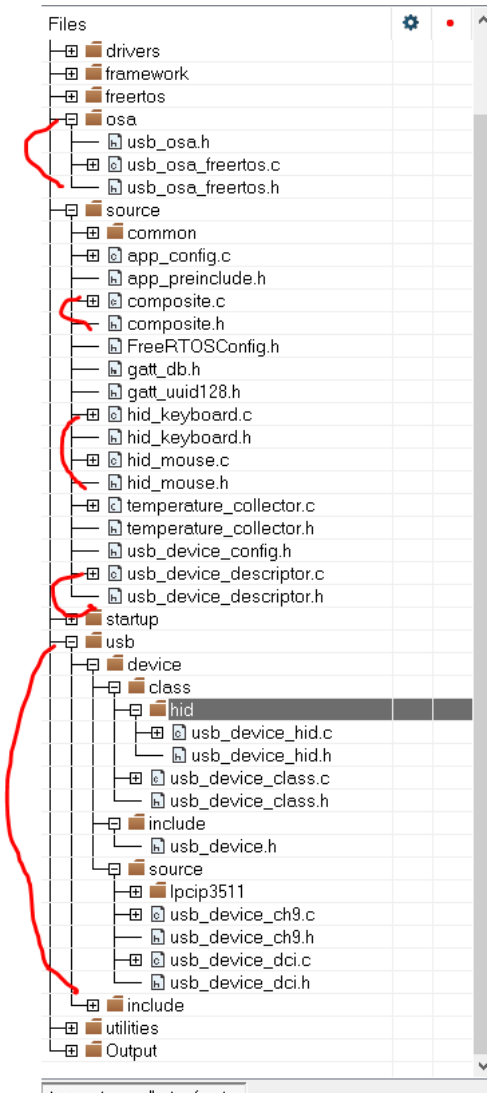

图 11

3.2.2 完成以后添加头文件目录

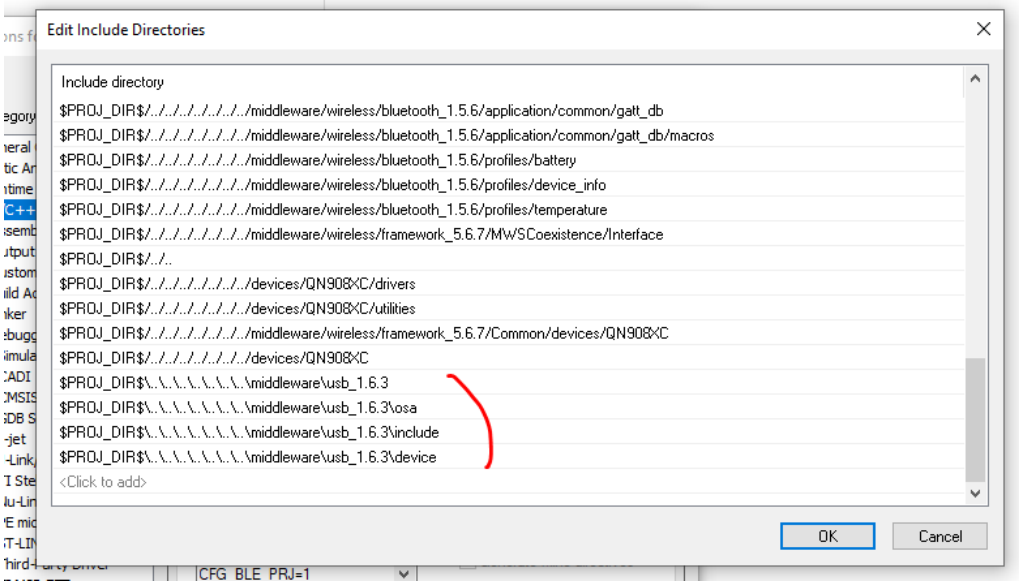

图 12

同时在该选项卡里，添加两个宏定义
USB_STACK_FREERTOS_HEAP_SIZE=16384
USB_STACK_FREERTOS

3.2.3 usb 例程本身时候 main 函数和初始化的我们需要进行修改，在 composite.c 文件中有 main 函数，我们做如图修改。

```
#if defined(__CC_ARM) || defined(__GNUC__)
int usb_main(void)
#else
void usb_main(void)
#endif
{
//    BOARD_InitPins();
//    BOARD_BootClockRUN();
//    BOARD_InitDebugConsole();
    POWER_DisablePD(kPDRUNCFG_PD_USBPLL); /* Turn on USB PLL */

    /* Disable USB PHY standby */
    SYSCON->USB_CFG = ((1 << SYSCON_USB_CFG_DPPUEN_B_PHY_POL_SHIFT) | (1 << SYSCON_USB_CFG_DPPUEN_B_PHY_SEL_SHIFT) |
                       (1 << SYSCON_USB_CFG_USB_VBUS_SHIFT) | (0 << SYSCON_USB_CFG_USB_PHYSTDBY_SHIFT) |
                       (0 << SYSCON_USB_CFG_USB_PHYSTDBY_WEN_SHIFT));

    SYSCON->MISC &= ~(1 << SYSCON_PIO_CFG_MISC_TRX_EN_INV_SHIFT);

    /* USB 48M clock calib */
    CLOCK_EnableClock(kCLOCK_Cal);
    CALIB_CalibPLL48M();

    APP_task(NULL);

//    if (xTaskCreate(APP_task,                                /* pointer to the task */
//                    "app task",                              /* task name for kernel awareness debugging */
//                    5000L / sizeof(portSTACK_TYPE),          /* task stack size */
//                    &g_UsbDeviceComposite,                   /* optional task startup argument */
//                    4U,                                      /* initial priority */
//                    &g_UsbDeviceComposite.applicationTaskHandle /* optional task handle to create */
//                    ) != pdPASS)
//    {
//        usb_echo("app task create failed!\r\n");
//#if (defined(__CC_ARM) || defined(__GNUC__))
//        return 1U;
//#else
//        return;
//#endif
//    }
//
//    vTaskStartScheduler();
```

图 13

它调用了 APP_task，这个函数也做修改

```
void APP_task(void *handle)
{
    USB_DeviceApplicationInit();
//#if USB_DEVICE_CONFIG_USE_TASK
//    if (g_UsbDeviceComposite.deviceHandle)
//    {
//        if (xTaskCreate(USB_DeviceTask,                      /* pointer to the task */
//                        "usb device task",                   /* task name for kernel awareness debugging */
//                        5000L / sizeof(portSTACK_TYPE),      /* task stack size */
//                        g_UsbDeviceComposite.deviceHandle,   /* optional task startup argument */
//                        5U,                                  /* initial priority */
//                        &g_UsbDeviceComposite.deviceTaskHandle /* optional task handle to create */
//                        ) != pdPASS)
//        {
//            usb_echo("usb device task create failed!\r\n");
//            return;
//        }
//    }
//#endif

//    while (1U)
//    {
//    }
}
```

图 14

3.2.4 找到 hid_mouse.c，将 USB_DeviceHidMouseAction 函数注释掉

找到 hid_keyboard.h，定义手势信息

```
/*gesture*/
#define BIT(x) 1<<(x)
#define GES_UP                BIT(0) //向上
#define GES_DOWM              BIT(1) //向下
#define GES_LEFT              BIT(2) //向左
#define GES_RIGHT             BIT(3) //向右
#define GES_FORWARD           BIT(4) //向前
#define GES_BACKWARD          BIT(5) //向后
#define GES_CLOCKWISE         BIT(6) //顺时针
#define GES_COUNT_CLOCKWISE BIT(7) //逆时针
#define GES_WAVE              BIT(8) //挥动
```

图 15

找到 hid_keyboard.c 函数，我们需要修改 USB_DeviceHidKeyboardAction 所要执行的任务。改成如图

```
static usb_status_t USB_DeviceHidKeyboardAction(void)
{
    if(gesture_from_server)
    {
        USB_DeviceGesture(gesture_from_server);
        gesture_from_server = 0;
    }
    else
    {
        s_UsbDeviceHidKeyboard.buffer[2] = 0;
        return USB_DeviceHidSend(s_UsbDeviceComposite->hidKeyboardHandle, USB_HID_KEYBOARD_ENDPOINT_IN,
                                 s_UsbDeviceHidKeyboard.buffer, USB_HID_KEYBOARD_REPORT_LENGTH);
    }
      return 0;
```

图 16

其中还要实现如下函数，当检测上挥手势就播放上一张 ppt，下挥就是下一张 ppt，左挥退出 ppt，向前则播放 ppt

```
extern uint16_t gesture_from_server;
usb_status_t USB_DeviceGesture(uint16_t gesture)
{
  if(gesture)
  {
      switch(gesture)
      {
          case GES_FORWARD: s_UsbDeviceHidKeyboard.buffer[2] = KEY_F5;break;
          case GES_UP:  s_UsbDeviceHidKeyboard.buffer[2] = KEY_LEFTARROW;break;
          case GES_DOWM:s_UsbDeviceHidKeyboard.buffer[2] = KEY_RIGHTARROW;break;
          case GES_LEFT:s_UsbDeviceHidKeyboard.buffer[2] = KEY_ESCAPE;break;
          default:break;
      }
      return USB_DeviceHidSend(s_UsbDeviceComposite->hidKeyboardHandle, USB_HID_KEYBOARD_ENDPOINT_IN,
                               s_UsbDeviceHidKeyboard.buffer, USB_HID_KEYBOARD_REPORT_LENGTH);
  }
  return 0;
}
```

图 17

这里面还引用了一个外部变量 gesture_from_server，这个变量定义在 temperature_colloctor.c 中，用来获取手势信息。

3.2.5 完成以上以后，我们到 BleApp_StateMachineHandler 函数中，在 case 为 mAppRunning_c 中调用 usb_main 来初始化 USB HID

```
        case mAppRunning_c:
        {
            if (event == mAppEvt_GattProcComplete_c)
            {
                usb_main();
                if (mpCharProcBuffer != NULL)
                {
                    (void)MEM_BufferFree(mpCharProcBuffer);
                    mpCharProcBuffer = NULL;
                }
```

图 18

3.2.6 最后在 BleApp_PrintTemperature 中，将收到的数据放到 gesture_from_server 里

```
static void BleApp_PrintTemperature
(
    uint16_t temperature
)
{

    shell_write("Temperature: ");
    gesture_from_server = (uint32_t)temperature / 100UL;
    shell_writeDec((uint32_t)temperature / 100UL);

    /* Add 'C' for Celsius degrees - UUID 0x272F.
       www.bluetooth.com/specifications/assigned-numbers/units */
    if (mPeerInformation.customInfo.tempClientConfig.tempFormat.unitUuid16 == 0x272FU)
    {
        shell_write(" C\r\n");
    }
    else
    {
        shell_write("\r\n");
    }
}
```

图 19