Gesture control PPT based on ble

# 1 Introduction

Two development boards transmit control information through ble. One development board connects to paj7620 and provides gesture information through IIC bus. The other development board uses ble and USB HID. Ble is used to receive data, and USB HID is used to simulate keyboard input and control ppt

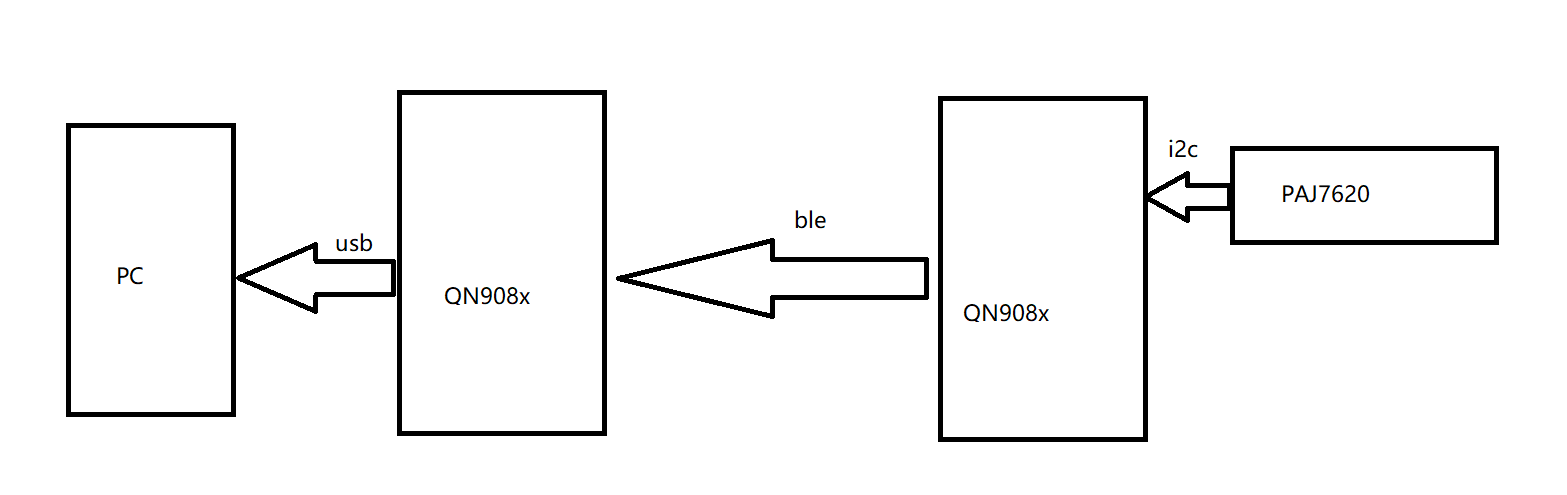


Figure 1

# 2 Preparation

We need two development boards qn908x and gesture control device paj7620. We use IAR as development enviroment.The example we use is temperature\_sensor, and

temperature\_ colloctor. The SDK version is 2.2.3

# 3 Code

3.1 temperature\_sensor code

We want to implement IIC to read gesture information from paj7620 and send data.

The pins used by IIC are PA6 and PA7

Simply encapsulate the IIC reading and writing code in the code to create i2c\_ operation.c and i2c\_ operation.h. Realize IIC initialization and reading / writing register function in it

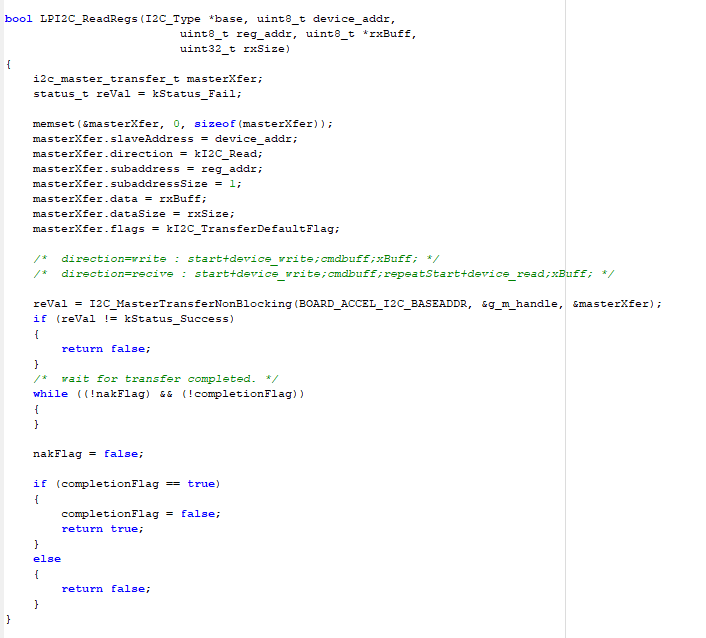


Figure 2

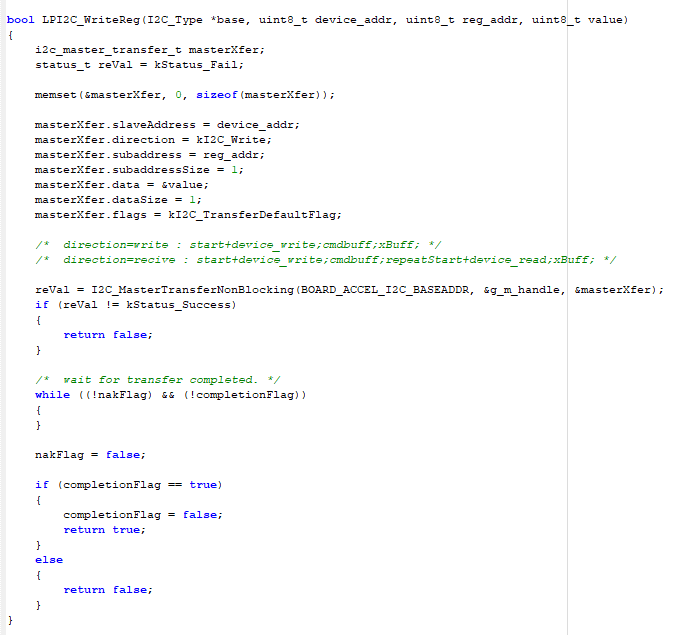


Figure 3

3.1.1 After having these functions, we begin to write gesture recognition code. First, we add two blank files paj7620.c and paj7620.h into our project.

Select bank register area

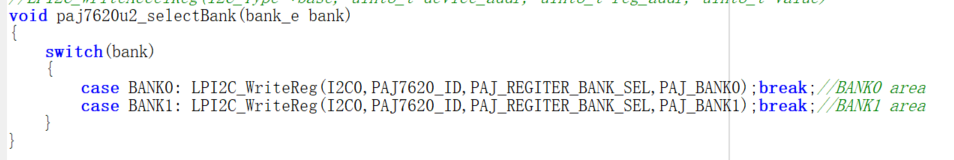


Figure 4

Wake up paj7620 to read device state

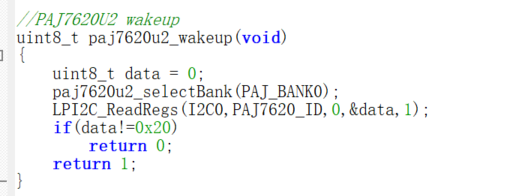


Figure 5

Initialize device

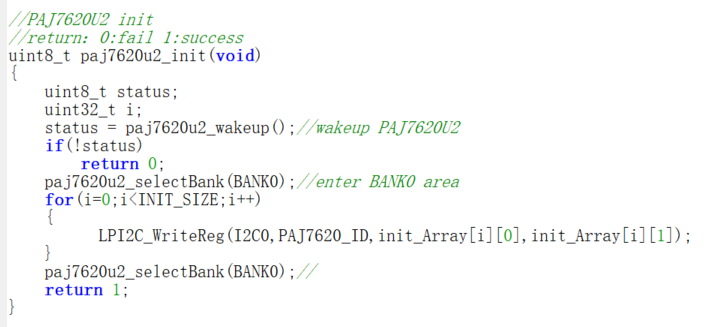


Figure 6

Gesture test function

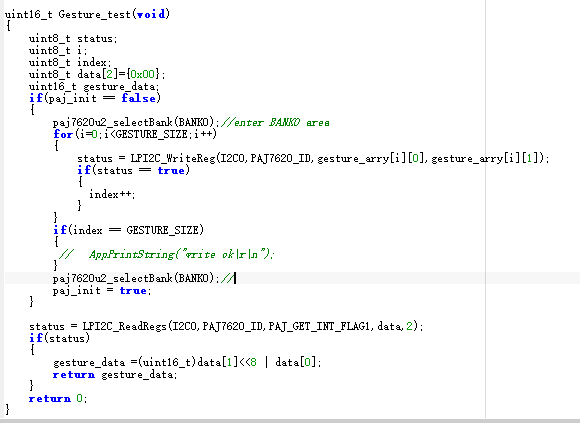


Figure 7

3.1.2 When you are ready to read the device information,

You should initialize IIC and paj7620 in BleApp\_Init function

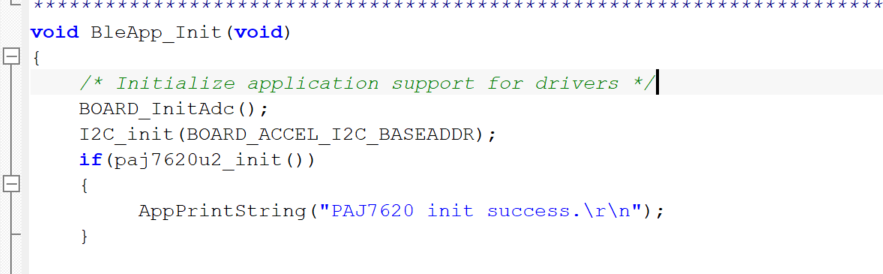


Figure 8

In principle, we need to create a custom service for the PAJ device, but we replace the temperature data as our gesture control data. If you want to create a custom service, refer to this link [custom profile](https://community.nxp.com/docs/DOC-332703)

3.1.3 Create a timer that sends gesture data regularly.

In file temerature\_sensor.c

Define a timer，static tmrTimerID\_t dataTimerId;

Allocate a timer, dataTimerId = TMR\_AllocateTimer();

Define the callback function of this timer

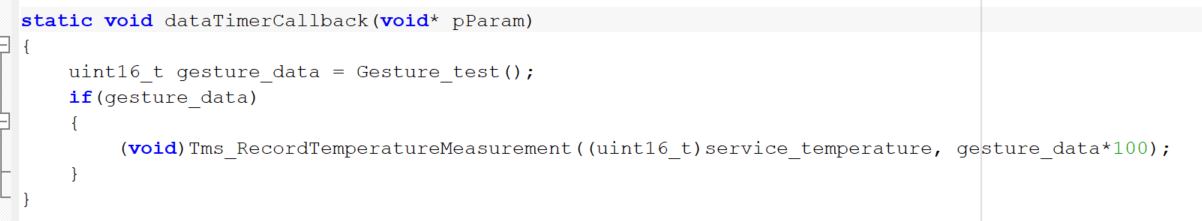


Figure 9

Start timer

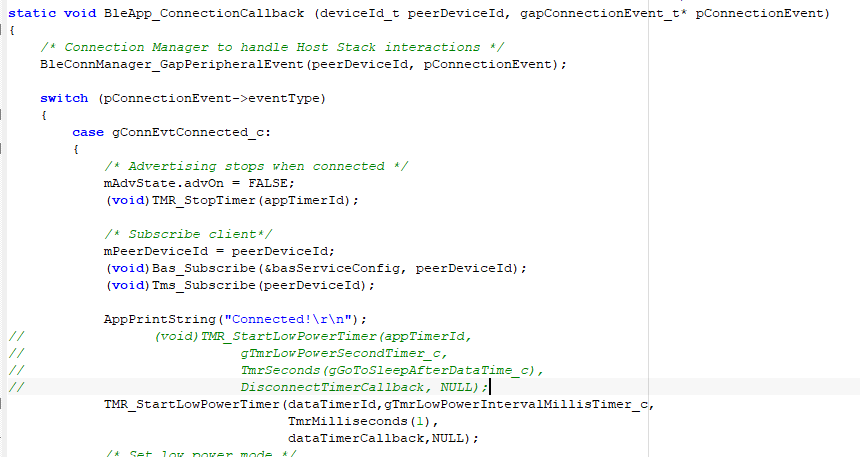


Figure 10

Close the low power mode. #define cPWR\_UsePowerDownMode 0

## 3.2 temperature\_collector code

The most important thing here is to port USB HID into our project. The USB example

we use is the USB keyboard and mouse.

3.2.1 Add the OSA and USB folder under the example to the project directory, and copy the file to the corresponding folder according to the file structure of the original example.

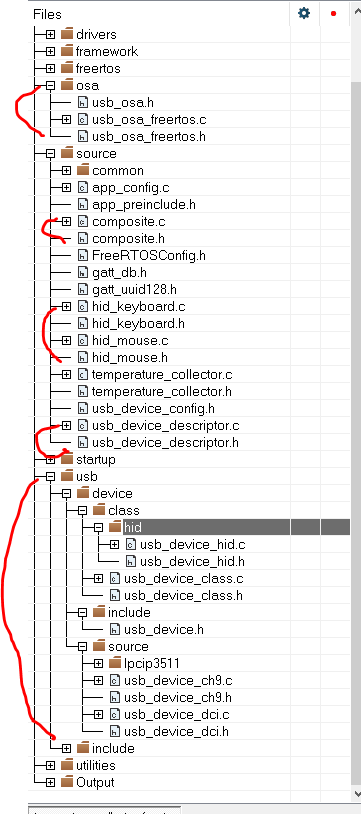


Figure 11

3.2.2 Add header file directory after completion

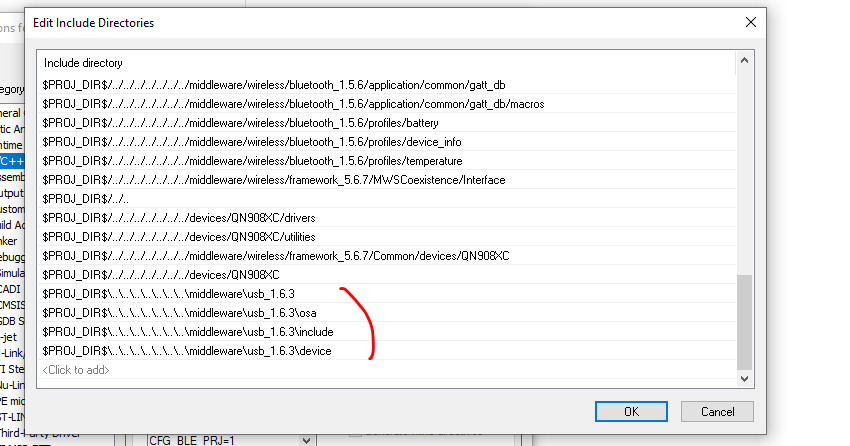


Figure 12

At the same time, in this tab, add two macro definitions

USB\_STACK\_FREERTOS\_HEAP\_SIZE=16384

USB\_STACK\_FREERTOS

3.2.3 Next, we need to modify the main function in usb example . Open composite.c file.

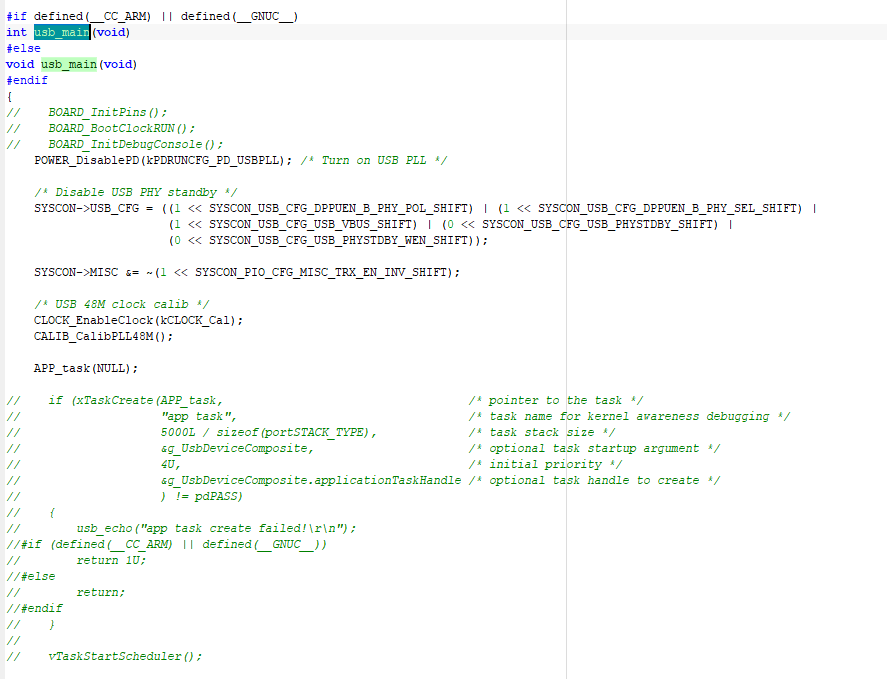


Figure 13

It calls the APP\_task. So this function also need to be modified.

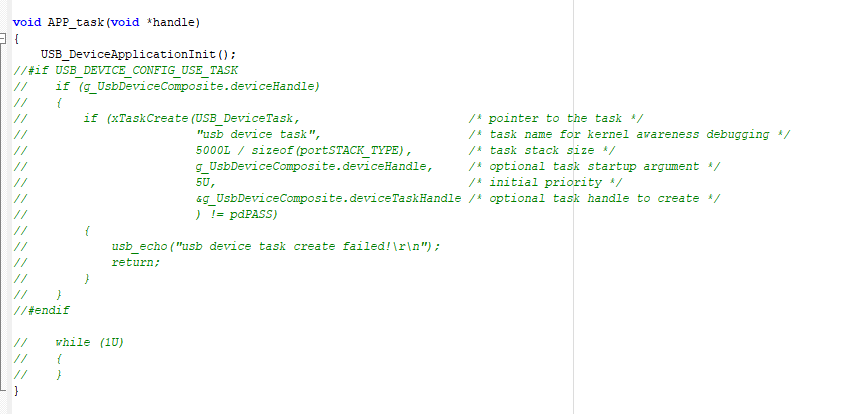


Figure 14

3.2.4Find hid\_mouse.c，Comment function USB\_DeviceHidMouseAction

Find hid\_keyboard.h. Define the gesture information.

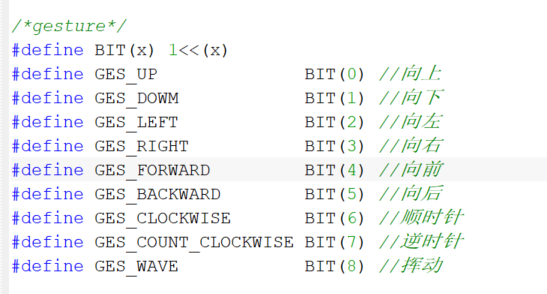


Figure 15

Find hid\_keyboard.c. We need to modify the function USB\_DeviceHidKeyboardAction as following figure.



Figure 16

Among them, we also need to implement the following function. When the up hand gesture is detected, the previous ppt will be played. The down hand gesture will be the next PPT, the left hand gesture will exit PPT, and the forward hand gesture will play ppt

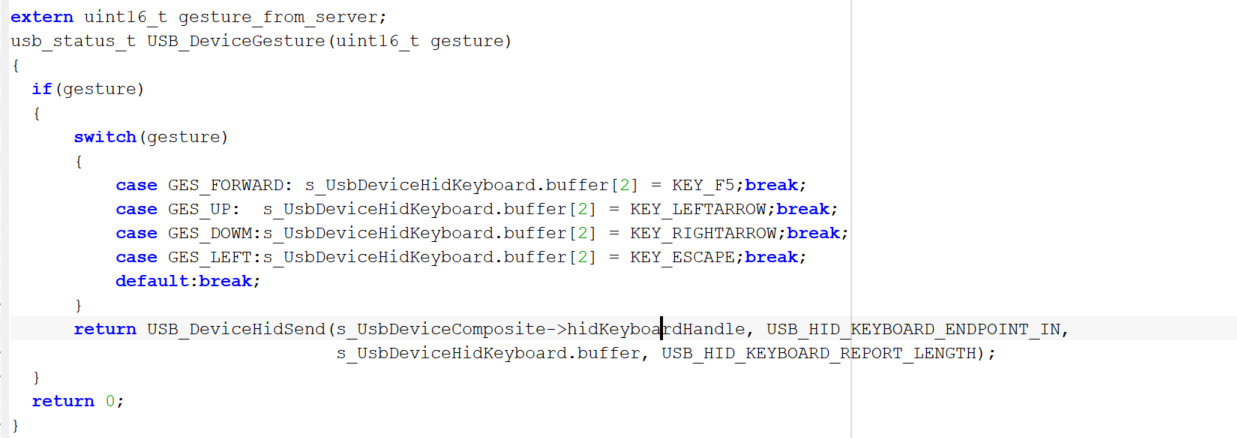


Figure 17

It also refers to an external variable gesture\_from\_server. The variable definition

is in file temperature\_ collocation.c,.

3.2.5 After that, let's go to BleApp\_Statemachinehandler function

in temperature\_colloctor.c.

In case mApppRunning\_c, we will call usb\_main to initialize USB HID

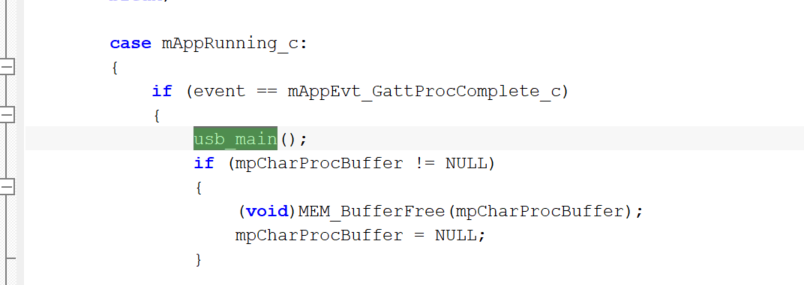


Figure 18

3.2.6 In BleApp\_PrintTemperature, we will save the gesture data to gesture\_from\_server



Figure 19

We finished the all steps.