

**硬件课程设计报告（个人）**

**题目： 触控感应键鼠的设计**

**院（系）： 电子信息与通信学院**

**学 号： U202114208**

**姓 名： 吴永浩**

**指导老师： 吴建新**

**报告日期： 2024.6.12**

# 硬件课设报告

目录

[1. 项目概述 5](#_Toc169183948)

[2. 设计与实现背景 5](#_Toc169183949)

[3. 项目功能指标 5](#_Toc169183950)

[3.1 功能指标 5](#_Toc169183951)

[3.2 硬件指标 6](#_Toc169183952)

[3.3 性能指标 6](#_Toc169183953)

[3.4 软件指标 7](#_Toc169183954)

[4. 团队分工 8](#_Toc169183955)

[5. 系统框图 9](#_Toc169183956)

[5.1. 触控键盘 9](#_Toc169183957)

[5.2. 信号采集电路 9](#_Toc169183958)

[5.3. Nexys4板子 9](#_Toc169183959)

[5.4. 蓝牙发送模块 10](#_Toc169183960)

[5.5. 蓝牙接收模块 10](#_Toc169183961)

[5.6. 终端显示 10](#_Toc169183962)

[6. 各模块原理 10](#_Toc169183963)

[6.1 TTP226触控模块 10](#_Toc169183964)

[6.2 HC-08蓝牙模块 10](#_Toc169183965)

[6.3 Nexys4DDR开发板 11](#_Toc169183966)

[7. 理论分析 12](#_Toc169183967)

[8. 关键技术 13](#_Toc169183968)

[8.1 电容触控检测技术 13](#_Toc169183969)

[8.2 蓝牙通信技术 13](#_Toc169183970)

[8.3 FPGA编程技术 13](#_Toc169183971)

[8.4 蓝牙模块-端口转换技术 14](#_Toc169183972)

[8.5 上位机C#程序部分 14](#_Toc169183973)

[8.6 数据处理与优化技术 14](#_Toc169183974)

[8.7 系统集成与调试技术 14](#_Toc169183975)

[9. 模块测试 15](#_Toc169183976)

[9.1 TTP226模块测试 15](#_Toc169183977)

[9.2 HC-08模块测试 15](#_Toc169183978)

[9.3 Nexys4DDR开发板测试 16](#_Toc169183979)

[9.4 综合系统测试 16](#_Toc169183980)

[10. 仿真图 17](#_Toc169183981)

[10.1 电容式触控传感器 17](#_Toc169183982)

[10.2 HC-08蓝牙模块 18](#_Toc169183983)

[11. 器件清单 19](#_Toc169183984)

[11.1 电容式触控传感器 19](#_Toc169183985)

[11.2 FPGA NEXYS 4 ddr开发板 19](#_Toc169183986)

[11.3 蓝牙模块 19](#_Toc169183987)

[12. 总结及心得体会 20](#_Toc169183988)

[13. 参考资料 21](#_Toc169183989)

[14. 附录 21](#_Toc169183990)

[14.1 键鼠模拟程序（部分） 21](#_Toc169183991)

[14.2 键鼠模拟程序说明（部分） 26](#_Toc169183992)

[变量说明 26](#_Toc169183993)

[线程说明 26](#_Toc169183994)

[函数说明 26](#_Toc169183995)

## 1. 项目概述

本项目旨在利用TTP226触控模块、Nexys4DDR开发板及HC-08蓝牙发送与接收模块实现键盘-鼠标模拟功能。项目通过触控模块获取用户输入，在nexys4DDR开发板上处理触控模块输入数据，并将数据发送至蓝牙输出模块。最后数据由蓝牙接收模块收到，并通过USB数据传输线连接到pc的端口，在pc上通过代码实现对鼠标、键盘的模拟。

## 2. 设计与实现背景

在现代社会中，触控技术已经广泛应用于各种电子设备中，如智能手机、平板电脑和触控笔记本等。然而，传统的物理键盘和鼠标在便携性和交互体验上存在一定的局限性。我们提出一个创新的设计——基于Xilinx FPGA以及触控和蓝牙技术的结合的触控感应键鼠，通过检测微小的电容变化，将其转换为数字信号，实现按键的触控感应。此设计不仅可以用作无线游戏，还可以一定程度上代替键盘和鼠标。

## 3. 项目功能指标

### 3.1 功能指标

1.触控感应：

检测触摸板上的按键区域，并将触摸信息转换为相应的数字信号。TTP226触控模块能够识别8个独立触摸按键，通过电容变化感应触摸，实现高灵敏度和稳定性的触控检测。

2.数据处理：

使用Nexys4DDR开发板上的FPGA对采集到的电容按键数据进行实时处理。FPGA通过编写硬件描述语言（如Verilog或VHDL）实现对数据的解析和转换，然后将处理后的数据传输到蓝牙发送模块。数据处理包括去抖动、按键状态识别和信号编码等步骤。

3.无线传输：

通过HC-08蓝牙模块，将处理后的数据无线传输到上位机（PC）。蓝牙模块使用串口通信协议，与FPGA进行数据交互，并通过蓝牙BLE标准实现稳定和低功耗的无线通信，确保数据传输的准确性和实时性。

4.游戏控制：

在上位机上，使用Unity引擎编写的程序和游戏，使得可以使用触摸板直接操控游戏。触控板的按键输入通过蓝牙传输到上位机，Unity引擎中的脚本程序接收并解析这些信号，实现游戏角色的移动、攻击等操作，增强用户的互动体验。

5.模拟键鼠：

在上位机上，使用Visual Studio平台编写C#窗体程序，处理蓝牙接收模块收到的信号。C#程序将触控板的输入信号映射为键盘和鼠标的操作，实现如下功能：

* 光标移动和点击：触控板上的特定区域可以控制鼠标光标的移动和点击操作。
* 打字输入：触控板的按键可以模拟键盘输入，实现打字功能。

这些功能使得触控板不仅可以作为输入设备，还能够在特定应用中替代传统的键盘和鼠标，提供更灵活和多样化的用户交互方式.

### 3.2 硬件指标

1.检测处理电路：

包括电容感应电路，将电容变化转换为数字信号；以及信号转换电路，将数字信号通过引脚发送至FPGA开发板。

2.FPGA控制平台：

使用Xilinx FPGA NEXYS4DDR开发板进行数据处理和控制。

3.蓝牙模块电路：

实现数据的无线传输功能。

蓝牙传输波特率：9600；

使用协议：低功耗蓝牙BLE

蓝牙发送模块连接到FPGA开发板，由其提供供电和数据输送。

蓝牙接收模块连接一块转换板为其提供供电和传输数据，并使得接收模块可以通过usb传输线连接到上位机端口。

### 3.3 性能指标

1.响应速度：

触控感应信号的响应时间应足够快，以保证流畅的用户体验。

2.精度：

能够准确检测到微小的电容变化，确保触控操作的精确性。

3.可靠性：

触控板发送的信息需要即时、无损地发送到FPGA开发板。

4.稳定性：

上位机程序需要对端口接收到的数据及时的处理、判断，确保正确的按键才能触发相应的事件；除此之外，还要考虑到用户的使用体验：应当适当的设置数据接收时间间隔，避免用户一次按键触发多次按键事件。

### 3.4 软件指标

1.上位机程序：

1.1.数据接收和处理

* **数据接收**：上位机程序使用C#编写，通过串口通信接收来自HC-08蓝牙模块的数据。程序应确保数据接收的实时性和准确性，能够处理丢包和错误数据的情况。
* **数据解析**：对接收到的数据进行解析，将其转换为相应的键盘或鼠标指令。解析过程中应考虑数据的格式和编码方式，确保正确还原触控板的操作。
* **数据处理**：处理数据包括去抖动、状态识别和动作触发等步骤，确保上位机能够准确响应触控板的输入。

1.2. 与终端设备交互：

* **模拟键盘和鼠标**：使用Windows API或其他库模拟键盘和鼠标的操作。程序应能够根据解析后的数据触发相应的键盘按键或鼠标移动、点击等操作。
* **游戏控制**：与Unity引擎编写的游戏程序进行交互，传递触控板的输入信号，实现游戏角色的控制。程序应设计良好的接口，确保数据的快速传递和响应。

2.蓝牙协议实现：

2.1.蓝牙通信协议

* **协议规范**：按照HC-08蓝牙模块的通信协议规范，实现数据的发送和接收。协议应包括连接建立、数据传输和连接断开的过程。
* **数据帧格式**：设计并实现适用于蓝牙传输的数据帧格式，包括起始位、数据位、校验位和结束位等部分，确保数据的完整性和可靠性。

2.2. 数据传输机制

* **串口通信**：在Nexys4DDR开发板上，通过UART接口与HC-08蓝牙模块进行通信。FPGA程序应能够将处理后的数据按照指定的帧格式发送至蓝牙模块。
* **数据接收与反馈**：上位机程序应能够接收并解析蓝牙模块传输的数据，并根据需要发送反馈信号，以确保数据传输的双向性和确认机制。

3. 虚拟键盘界面

3.1..界面设计

* **用户友好界面**：设计一个直观且易于操作的虚拟键盘界面。界面应包含所有标准键盘按键，并能够根据需要进行自定义布局，以适应不同的应用场景。
* **游戏控制界面**：设计一个专门用于游戏控制的界面，包含常用的游戏操作按键，如方向键、动作键等。界面应能够实时响应触控板的输入，实现流畅的游戏控制。

3.2.功能实现

* **虚拟按键映射**：将触控板的输入映射到虚拟键盘的按键上。程序应能够识别触控板的不同按键，并将其转换为相应的虚拟按键操作。
* **触控反馈**：实现触控反馈功能，在用户触摸按键时给出视觉或声音反馈，增强用户体验。例如，当用户点击虚拟按键时，按键会在界面上显示按下状态，同时播放点击音效。
* **可配置性**：提供界面的自定义配置功能，使用户能够根据个人需求调整按键布局和功能。例如，用户可以自定义某个按键的功能或位置，以适应特定的应用或游戏需求。

## 4. 团队分工

* **上位机编程----吴永浩**：

负责整体系统设计和架构规划。

设计和实现蓝牙模块的集成。

完成电脑端键鼠输入信号处理的设计。

* **FPGA编程----李鑫瑞**：

负责触控板信号的输入处理。

负责触控板性能的升级。

搭建测试平台和环境，对系统进行验证和调试。

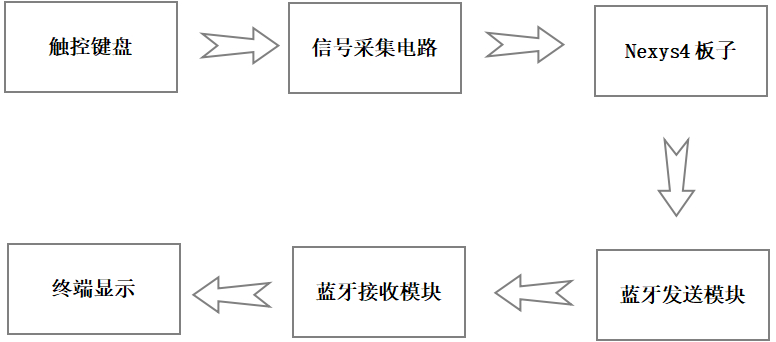
* **系统调试----王梦婷**：

负责处理触控板输入信号并传输至蓝牙发送端。

负责对硬件和软件进行系统联调。

搭建测试平台和环境，对系统进行验证和调试。

## 5. 系统框图



其中信号采集电路已集成到触控键盘模块。

**系统框图详细描述**

### 5.1. 触控键盘

触控键盘（TTP226模块）是整个系统的输入设备。用户通过触摸键盘上的按键来输入指令。触控键盘通过电容变化感应触摸信号，并将这些信号转换为相应的电信号输出。

### 5.2. 信号采集电路

信号采集电路负责接收来自触控键盘的电信号，并对这些信号进行初步处理。该电路可以包括信号放大、滤波和模数转换等功能，以确保信号的准确性和稳定性。处理后的信号将传输到Nexys4板子进行进一步处理。

### 5.3. Nexys4板子

Nexys4板子是整个系统的核心处理单元。该板子基于Xilinx Artix-7 FPGA，具有强大的处理能力和丰富的外设接口。Nexys4板子接收并处理来自信号采集电路的触控信号，通过FPGA编写的程序实现对信号的解析、处理和编码。处理后的数据将通过串口发送到蓝牙发送模块。

### 5.4. 蓝牙发送模块

蓝牙发送模块（HC-08模块）与Nexys4板子通过串口通信。它负责将处理后的数据无线传输到蓝牙接收模块。该模块遵循蓝牙4.0协议，具有低功耗和长传输距离的特点，确保数据的稳定传输。

### 5.5. 蓝牙接收模块

蓝牙接收模块接收来自蓝牙发送模块的数据。它将无线接收的数据转换为串口信号，传输到终端显示设备进行进一步处理和显示。蓝牙接收模块同样遵循蓝牙4.0协议，确保数据接收的准确性和实时性。

### 5.6. 终端显示

终端显示设备（上位机）接收并处理来自蓝牙接收模块的数据。上位机运行一个C#编写的程序，该程序负责解析接收到的数据，并根据这些数据执行相应的操作，如控制鼠标光标移动、模拟键盘按键等。此外，终端显示设备还可以运行Unity引擎编写的游戏，通过触控键盘控制游戏中的角色和操作。

## 6. 各模块原理

### 6.1 TTP226触控模块

TTP226是一种电容式触摸传感器IC，具有8个独立的触摸按键输入。其工作原理基于电容检测，通过感应电荷变化实现触摸检测。当用户触摸某个按键时，触摸点的电容值发生变化，TTP226检测到这一变化并将其转换为电信号。触控模块内部的处理电路对信号进行处理，去除干扰和噪声，然后通过引脚并行输出触摸信号。每个按键对应一个独立的输出引脚，确保各个按键的触摸信号能够被准确区分和传输。

#### 工作步骤：

1. **电容感应**：每个按键区域有一个电容，当用户触摸时，电容值发生变化。
2. **信号处理**：TTP226内部的电路检测电容变化，并将其转换为电信号。
3. **输出信号**：处理后的信号通过并行输出引脚传输，确保各个按键信号的准确性。

### 6.2 HC-08蓝牙模块

HC-08是一款低功耗蓝牙模块，支持蓝牙4.0 BLE协议。系统中使用了两个HC-08模块，分别作为数据的发送和接收端。发送端HC-08模块通过串口与Nexys4DDR开发板连接，接收端HC-08模块通过串口-USB转换架连接到PC端口的USB传输线。数据传输过程中，TXD引脚负责发送数据，RXD引脚负责接收数据。HC-08模块在BLE协议下进行通信，具有低功耗、传输距离长和数据传输稳定的特点。

#### 工作步骤：

1. **初始化**：HC-08模块上电后初始化，设置波特率、角色（主/从）等参数。
2. **连接建立**：发送端与接收端通过蓝牙协议进行配对和连接建立。
3. **数据传输**：发送端通过TXD引脚发送数据，接收端通过RXD引脚接收数据。数据以串口数据帧格式进行传输，包括起始位、数据位、校验位和停止位。
4. **连接断开**：数据传输完成后，蓝牙模块可以按照协议断开连接，进入低功耗模式。

#### 硬件连接：

* **发送端**：HC-08发送模块的TXD引脚连接到Nexys4DDR开发板的UART接收引脚（RX），RXD引脚连接到Nexys4DDR开发板的UART发送引脚（TX）。
* **接收端**：HC-08接收模块的TXD引脚连接到串口-USB转换架的RX引脚，RXD引脚连接到串口-USB转换架的TX引脚。串口-USB转换架通过USB传输线连接到PC端口。

### 6.3 Nexys4DDR开发板

Nexys4DDR开发板是一款基于Xilinx Artix-7 FPGA的开发板，提供丰富的外设接口和强大的逻辑资源。该开发板具有以下主要功能和特点：

* **FPGA资源**：内置Xilinx Artix-7 FPGA，具有强大的逻辑处理能力，支持用户编写Verilog或VHDL代码实现复杂的硬件逻辑功能。
* **外设接口**：板载提供多种外设接口，包括UART、SPI、I2C、GPIO等，可以与各种外部设备进行连接和通信。
* **存储资源**：板载具有DDR2 SDRAM、Flash存储器等，支持数据的存储和访问。
* **开发环境**：支持Xilinx的Vivado设计套件，通过编写硬件描述语言，实现FPGA内部逻辑功能。

在本项目中，Nexys4DDR开发板负责接收和处理来自TTP226触控模块的触摸信号。通过编写FPGA代码，实现对触摸信号的解析、处理和编码。处理后的数据通过UART接口传输到HC-08蓝牙发送模块，实现数据的无线传输。Nexys4DDR开发板在整个系统中充当核心处理单元，负责信号的实时处理和传输。

## 7. 理论分析

项目的核心在于电容触控检测和蓝牙通信的实现。电容触控检测依赖于TTP226模块，通过检测电容变化来感应触摸信号。电容触控技术的基本原理是，当用户的手指靠近或接触电容触摸传感器时，会改变传感器表面的电场分布，导致电容值的变化。TTP226模块内部集成了高灵敏度的电容检测电路，能够实时感应这些变化，并将其转换为电信号输出。触控信号通过并行引脚传输，确保各个按键的触摸信号能够被准确区分和传输。

信号经TTP226处理后，通过信号采集电路传输至Nexys4DDR开发板。Nexys4DDR基于Xilinx Artix-7 FPGA，提供强大的逻辑处理能力和丰富的外设接口。FPGA通过编写Verilog或VHDL代码，实现对触控信号的进一步处理。处理过程包括信号去抖动、按键状态识别和信号编码等步骤。去抖动技术用于消除按键抖动造成的误触发，确保信号的准确性；按键状态识别则用于判断按键的按下和松开状态；信号编码则将处理后的触控信号转换为特定的格式，便于后续的传输和解析。

处理后的信号通过UART接口传输至HC-08蓝牙发送模块。HC-08是一款低功耗蓝牙模块，支持蓝牙4.0 BLE协议。蓝牙通信的实现需要遵循特定的协议规范，确保数据的稳定传输。HC-08模块通过串口与Nexys4DDR开发板连接，利用TXD和RXD引脚进行数据的发送和接收。数据传输过程中，TXD引脚负责将处理后的触控信号发送至蓝牙接收模块，而RXD引脚则用于接收来自蓝牙接收模块的反馈信号。

蓝牙接收模块通过BLE协议接收到的数据通过串口-USB转换架传输至PC端口。BLE（Bluetooth Low Energy）协议是蓝牙4.0标准中的一个重要部分，具有低功耗、传输距离长和数据传输稳定的特点，适用于低功耗设备间的无线通信。串口-USB转换架将蓝牙接收模块的串口信号转换为USB信号，方便PC端口进行数据接收和处理。

在PC端，使用C#编写的上位机程序负责接收和处理来自蓝牙接收模块的数据。程序通过串口通信接收数据，并对数据进行解析，将其转换为相应的键盘或鼠标指令。解析后的数据可以用于控制鼠标光标的移动、点击和键盘的按键操作。此外，PC端还可以运行使用Unity引擎编写的游戏程序，通过触控信号实现对游戏角色的控制，增强用户的互动体验。

项目的实现涉及电容触控检测、信号处理、蓝牙通信和数据解析等多个环节。各个模块的协同工作，确保了系统的稳定性和实时性，实现了触控键盘的功能扩展和应用。通过合理的设计和实现，本项目在用户交互和应用扩展方面具有重要意义。

## 8. 关键技术

### 8.1 电容触控检测技术

TTP226模块利用电容变化检测触摸信号，具有高灵敏度和稳定性。电容触控技术的核心在于电容值的变化。当用户触摸触控面板时，电容值会发生变化，TTP226模块通过检测这些变化来识别触摸事件。该模块内部集成了高精度的电容检测电路，能够实时感应微小的电容变化，并将其转换为数字信号输出。高灵敏度的电容检测技术确保了触控操作的准确性和响应速度，适用于各种触控应用场景。此外，TTP226模块具有较高的抗干扰能力，能够在复杂电磁环境下稳定工作，保证触控信号的可靠性。通过合理的电路设计和优化的信号处理算法，TTP226模块在实际应用中展现出了卓越的性能。

### 8.2 蓝牙通信技术

HC-08模块采用蓝牙BLE（Bluetooth Low Energy）标准，具有低功耗和较长的通信距离。BLE技术特别适合低功耗设备间的短距离通信，通过简化的协议栈和优化的传输机制，显著降低了功耗。HC-08模块支持多种工作模式，包括广播模式和连接模式，能够灵活适应不同的应用需求。

在本项目中，两个HC-08模块分别作为发送端和接收端，通过串口与其他设备连接，实现了稳定可靠的数据传输。蓝牙通信的实现需要遵循严格的协议规范，确保数据传输的完整性和准确性。通过合理的协议栈设计和优化的传输机制，HC-08模块在低功耗和高效传输之间找到了平衡点，适用于各种无线通信场景。

### 8.3 FPGA编程技术

Nexys4DDR开发板基于Xilinx Artix-7 FPGA，具有强大的逻辑处理能力和丰富的外设接口。通过编写Verilog或VHDL硬件描述语言，可以实现复杂的硬件逻辑功能。

在本项目中，FPGA负责接收并处理来自TTP226触控模块的并行信号，将其转换为串行信号，并以一定的时间间隔发送到蓝牙发送模块的输入端口。FPGA编程技术在信号处理、数据传输和逻辑控制方面发挥了关键作用，确保了系统的稳定性和实时性。通过灵活的硬件编程，FPGA能够实现高度并行的信号处理和实时的数据传输，为系统的性能优化提供了强有力的支持。FPGA的高可编程性使其在各种复杂应用场景中得到了广泛应用，展现出了优异的适应性和扩展性。

### 8.4 蓝牙模块-端口转换技术

蓝牙接收模块通过串口-USB转换架连接到PC的USB端口，实现了蓝牙到USB端口的转换。转换模块将蓝牙接收模块接收到的串口信号转换为USB信号，使PC能够通过COM端口接收数据。该转换技术不仅扩展了蓝牙模块的应用范围，还简化了与PC的连接方式，确保了数据传输的便捷性和兼容性。串口-USB转换架在数据传输过程中起到了桥梁作用，通过有效的数据帧格式转换和传输协议匹配，确保了数据传输的稳定性和高效性。该技术在实际应用中具有广泛的适用性，能够满足各种数据传输需求，提升了系统的兼容性和灵活性。

### 8.5 上位机C#程序部分

上位机程序使用C#编写，负责接收并处理来自蓝牙接收模块的数据，实现键鼠模拟功能。程序通过调用System.IO.Ports命名空间中的串口类监听COM端口，将接收到的数据存储为字节数组。然后，利用C#库中的鼠标和键盘事件处理机制，对接收到的数据进行解析和判断，触发相应的鼠标和键盘操作。该程序能够模拟鼠标光标的移动、点击和键盘按键等功能，为用户提供了自然直观的交互方式。在实际应用中，上位机程序通过优化的数据解析算法和高效的事件处理机制，确保了数据的实时响应和准确执行。通过合理的界面设计和用户体验优化，C#程序在提升用户交互体验方面发挥了重要作用。

### 8.6 数据处理与优化技术

在整个系统中，数据处理与优化技术贯穿始终。TTP226触控模块产生的触摸信号需要经过FPGA的处理，包括去抖动、按键状态识别和信号编码等步骤。通过去抖动技术，可以有效消除按键抖动带来的误触发问题，保证信号的稳定性和准确性。按键状态识别技术则用于判断按键的按下和松开状态，确保每一次触控操作都能够被准确识别。信号编码技术将处理后的触摸信号转换为特定的格式，便于后续的传输和解析。在蓝牙通信过程中，数据传输需要经过协议栈的处理，确保传输的完整性和准确性。通过优化传输机制和数据帧格式，可以有效提升数据传输的效率和稳定性。

### 8.7 系统集成与调试技术

项目的成功实现离不开各个模块的系统集成与调试。在系统集成过程中，需要将TTP226触控模块、Nexys4DDR开发板、HC-08蓝牙模块和上位机程序进行有效的连接和协调。通过合理的系统架构设计和接口定义，确保各个模块之间的数据传输和信号处理能够顺畅进行。在系统调试过程中，通过逐步验证和优化各个模块的功能，发现并解决潜在的问题，确保系统的整体性能和稳定性。系统集成与调试技术在项目开发中起到了关键作用，确保了各个模块的协同工作和系统的最终实现。

## 9. 模块测试

### 9.1 TTP226模块测试

#### 测试方法

TTP226触控模块的测试主要通过单片机读取触控按键信号，并将这些信号通过串口输出到PC上进行验证。具体步骤如下：

1. 将TTP226模块连接到单片机的输入引脚，确保每个触摸按键的输出引脚都连接到单片机的对应引脚。
2. 编写单片机程序，实时读取每个触摸按键的状态，并将这些状态数据通过串口发送到PC。
3. 在PC上编写一个简单的串口接收程序，接收并显示来自单片机的按键状态数据。
4. 手动按下TTP226模块上的每个触摸按键，观察PC端显示的按键状态变化，验证触摸按键的响应情况。

#### 测试结果

经过多次测试，所有触摸按键都能正常响应，且输出数据准确。具体测试结果如下：

* 每个触摸按键在被按下时，PC端显示的按键状态会立即发生变化，表明触控信号能够实时传输。
* 按键状态的变化与实际按键操作完全一致，没有出现误触发或漏触发的情况。
* 在不同的触摸力度和环境条件下，TTP226模块的灵敏度和稳定性表现良好，能够准确检测到每一次触摸操作。

### 9.2 HC-08模块测试

#### 测试方法

HC-08蓝牙模块的测试主要通过与手机蓝牙配对，发送和接收数据进行验证。具体步骤如下：

1. 将HC-08蓝牙模块与单片机连接，通过串口进行通信。
2. 编写单片机程序，设置HC-08模块进入配对模式，并等待手机蓝牙的连接请求。
3. 使用手机蓝牙功能搜索HC-08模块，并进行配对连接。
4. 在手机上编写一个简单的蓝牙通信应用，发送数据到HC-08模块，并接收从HC-08模块发送的数据。
5. 通过单片机程序发送测试数据到手机，并接收手机发送的数据，验证数据传输的正确性和稳定性。

#### 测试结果

经过多次测试，HC-08蓝牙模块的通信稳定，数据传输准确。具体测试结果如下：

* 手机能够成功搜索到HC-08蓝牙模块，并顺利完成配对连接。
* 手机应用发送的数据能够被HC-08模块接收，并通过串口传输到单片机，单片机能够正确处理接收到的数据。
* 单片机发送的数据能够通过HC-08模块传输到手机，手机应用能够准确接收并显示这些数据。
* 在不同的通信距离和环境条件下，蓝牙通信的稳定性和传输速度表现良好，没有出现数据丢失或传输错误的情况。

### 9.3 Nexys4DDR开发板测试

#### 测试方法

Nexys4DDR开发板的测试主要通过编写简单程序，测试开发板的输入输出功能进行验证。具体步骤如下：

1. 编写一个简单的Verilog或VHDL程序，控制开发板上的LED灯和按键。
2. 将程序下载到Nexys4DDR开发板上，观察开发板的运行情况。
3. 手动操作开发板上的按键，观察LED灯的状态变化，验证按键输入和LED输出的正确性。
4. 编写更复杂的程序，测试开发板的其他外设接口，如VGA、UART、SPI等，验证其功能和性能。

#### 测试结果

经过多次测试，Nexys4DDR开发板的各项功能正常，能够正确执行程序。具体测试结果如下：

* 简单程序测试中，开发板上的LED灯能够根据按键的操作变化状态，按键输入和LED输出的响应速度和准确性都表现良好。
* 更复杂的程序测试中，开发板的其他外设接口，如VGA、UART、SPI等，均能够正常工作，数据传输和处理稳定。
* 在不同的测试场景和条件下，Nexys4DDR开发板的稳定性和性能表现出色，能够胜任各种复杂应用需求。

### 9.4 综合系统测试

#### 测试方法

综合系统测试旨在验证各模块集成后的整体性能。具体步骤如下：

1. 将TTP226触控模块、Nexys4DDR开发板、HC-08蓝牙模块以及上位机程序进行系统集成。
2. 编写FPGA程序，处理来自TTP226模块的触控信号，并通过HC-08模块将数据传输到PC。
3. 在PC上运行C#上位机程序，接收并处理蓝牙模块传输的数据，模拟键鼠功能。
4. 手动操作触控模块，观察PC上的鼠标和键盘响应情况，验证整体系统的性能和稳定性。

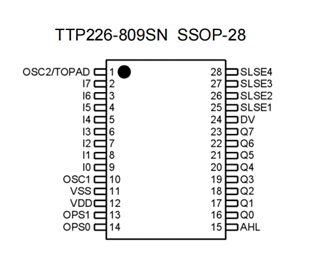
#### 测试结果

综合系统测试结果表明，各模块集成后的整体系统性能良好。具体测试结果如下：

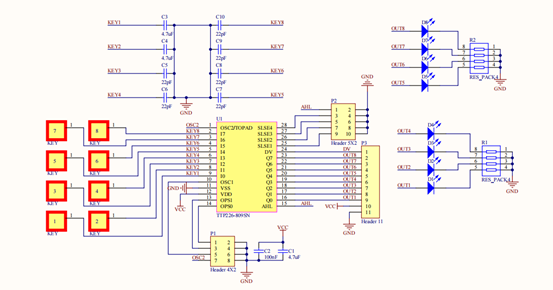
* 触控模块的触摸信号能够准确传输到FPGA，并通过蓝牙模块发送到PC。
* PC端的上位机程序能够实时接收并处理来自蓝牙模块的数据，模拟键鼠功能。
* 系统在不同的操作条件下表现稳定，没有出现数据传输延迟或错误的情况。
* 整体系统的响应速度快，用户操作体验良好，能够满足实际应用需求。

## 10. 仿真图

### 10.1 电容式触控传感器

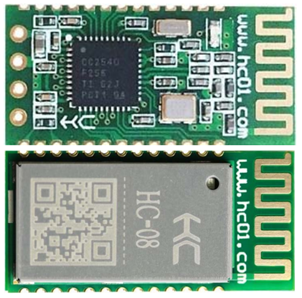


*TTP226模块芯片封装*

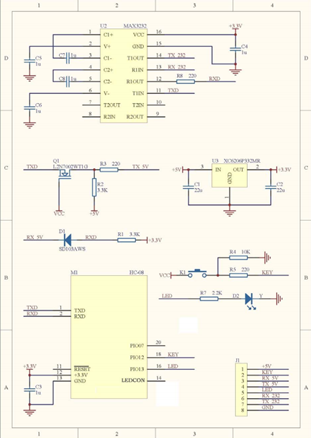


*TTP226电路图*

### 10.2 HC-08蓝牙模块



*HC-08蓝牙模块实物图*



*HC-08蓝牙模块电路图*

## 11. 器件清单

### 11.1 电容式触控传感器

一块TTP226 8路电容式触摸开关模块，及其连接电路

### 11.2 FPGA NEXYS 4 ddr开发板

使用Xilinx nexys4DDR开发板，进行电源和接口设计。

编写FPGA逻辑程序，完成电容模块数据的处理。

设计与蓝牙模块的数据交互，确保与蓝牙模块的通信稳定。

### 11.3 蓝牙模块

选择两块HC-08主从一体蓝牙模块分别作为发送模块和接收模块。

设计接口电路，实现与FPGA以及上位机（个人电脑）的无缝连接。

## 12. 总结及心得体会

通过本次项目，我们深入了解了电容触控技术和蓝牙通信技术的原理及应用。电容触控技术通过检测电容变化来识别触摸操作，这种技术在触控设备中具有广泛的应用前景。我们通过TTP226模块，成功实现了高灵敏度的触摸检测，并将触摸信号转换为数字信号进行处理。在蓝牙通信技术方面，HC-08模块采用低功耗蓝牙（BLE）标准，通过串口与其他设备进行通信，实现了数据的稳定传输。我们通过该模块，成功实现了蓝牙数据的发送和接收，将触控信号传输到上位机。

项目中，我们面对了硬件连接、驱动编写、通信协议实现等多方面的挑战。在硬件连接方面，我们需要确保各个模块之间的信号传输稳定，特别是TTP226模块与FPGA开发板、HC-08模块与单片机之间的连接。在驱动编写方面，我们需要编写FPGA程序来处理触控信号，并通过串口发送数据。在通信协议实现方面，我们需要确保蓝牙模块能够稳定进行数据传输，并在上位机中正确接收和处理数据。

在项目的不同阶段，我们遇到了许多问题。例如，在硬件连接过程中，由于电路设计的不完善，曾出现触控信号不稳定的情况。通过不断尝试和调整电路设计，我们最终解决了这一问题。在驱动编写过程中，由于对FPGA编程不够熟练，曾遇到程序运行不正常的情况。通过查阅资料，我们逐渐掌握了FPGA编程的技巧，成功编写了稳定的驱动程序。在通信协议实现过程中，由于对蓝牙通信协议不够了解，曾出现数据传输不稳定的情况。通过深入研究BLE协议，并进行多次调试，我们最终实现了稳定的蓝牙数据传输。

团队合作在项目中发挥了关键作用。每个成员都充分发挥了自己的专长，共同解决了项目中遇到的各种问题。在项目的初期阶段，我们通过分工协作，明确了各自的任务，并定期进行交流和讨论，确保项目进展顺利。在项目的实施过程中，我们通过协作解决了硬件连接、驱动编写和通信协议实现等方面的问题。每次遇到困难时，团队成员都积极参与讨论，共同寻找解决方案。通过团队合作，我们不仅提升了项目的效率，也增进了彼此的理解和信任。

此次项目经历不仅提升了我们的硬件设计与调试能力，也增强了我们对团队合作重要性的认识。在硬件设计方面，我们通过实践掌握了电容触控技术和蓝牙通信技术的原理及应用，提高了电路设计和信号处理的能力。在调试方面，我们通过多次调试和优化，解决了项目中遇到的各种问题，增强了故障排除和系统优化的能力。在团队合作方面，我们通过分工协作和共同努力，顺利完成了项目，提高了团队协作和沟通的能力。

这次项目不仅是一次技术上的提升，更是一次宝贵的学习和成长经历。我们不仅学到了很多新的知识和技能，还在项目中体验到了团队合作的力量。通过这个项目，我们更加深刻地认识到，在实际工程项目中，技术和团队合作同样重要。只有将技术和团队合作相结合，才能成功完成复杂的工程项目。我们也认识到，不断尝试和学习是解决问题的关键，只有通过不断的努力和探索，才能在技术领域不断进步。此次项目经历为我们的学习和职业发展打下了坚实的基础，也为我们今后的学习和工作提供了宝贵的经验和启示。

## 13. 参考资料

* + TTP226-809SN 8按键触摸检测芯片的详细中文数据手册
  + Nexys 4 DDR Reference Manua
  + Nexys4 开发板用户手册
  + Unity User Manual 2022.3 (LTS)
  + HC-08蓝牙传输模块用户手册
  + MicroSoft Visual Studio 文档

## 14. 附录

### 14.1 键鼠模拟程序（部分）

此部分为键鼠模拟程序的部分代码及其解释，省略了按钮等控件部分。

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO.Ports;

using System.Threading;

namespace comController

{

    public partial class Form1 : Form

    {

        private string portName = "COM5";

        private int baudRate = 9600;

        private Parity parity = Parity.Odd;//奇校验

        private int dataBits = 8;

        private StopBits stopBits = StopBits.One;//停止位

 // 串口对象

        private SerialPort sp = null;//创建一个串口对象

        // 读取数据线程对象

        private Thread dataReceiveThread = null;//创建一个线程

        private Thread writeReceived = null;//创建一个线程,用于写入接收到的数据

        // 是否接收数据

        private bool canRecieveMsg = true;

        // 接收到的数据

        public string strReceived;//接收到的数据

        public byte[] bytesReceived;//接收到的字节数组

        public int bytesLength;//接收到的字节数组长度

        private bool IsOpenSerial = false;

        public string receivedData = string.Empty;

        [System.Runtime.InteropServices.DllImport("user32")]//引入user32.dll鼠标库

        private static extern int mouse\_event(int dwFlags, int dx, int dy, int cButtons, int dwExtraInfo);

        //移动鼠标

        const int MOUSEEVENTF\_MOVE = 0x0001;

        //模拟鼠标左键按下

        const int MOUSEEVENTF\_LEFTDOWN = 0x0002;

        //模拟鼠标左键抬起

        const int MOUSEEVENTF\_LEFTUP = 0x0004;

        //模拟鼠标右键按下

        const int MOUSEEVENTF\_RIGHTDOWN = 0x0008;

        //模拟鼠标右键抬起

        const int MOUSEEVENTF\_RIGHTUP = 0x0010;

        //模拟鼠标中键按下

        const int MOUSEEVENTF\_MIDDLEDOWN = 0x0020;

        //模拟鼠标中键抬起

        const int MOUSEEVENTF\_MIDDLEUP = 0x0040;

        //标示是否采用绝对坐标

        const int MOUSEEVENTF\_ABSOLUTE = 0x8000;

        public int moveSpeed = 10;//鼠标移动速度

        public int moveSleep = 30;//鼠标移动延时

        public int keybdSleep = 15;//键盘输入延时

        public int controlMode = 0;//控制模式，0为鼠标，1为键盘

        public Form1()

        {

            InitializeComponent();

            Control.CheckForIllegalCrossThreadCalls = false;

        }

        private void Form1\_Load(object sender, EventArgs e)//窗体加载时

        {

            stateTEXT.Text = "COM5";

            textBox1.Text = "9600";

            sp = new SerialPort(portName, baudRate, parity, dataBits, stopBits);

            strReceived = string.Empty;

        }

        public void ClosePort()

        {

            try

            {

                sp.Close();

                Console.WriteLine("close success");

                stateTEXT.Text = "关闭成功";

                //Debug.Log("close success");

                dataReceiveThread.Abort();

                writeReceived.Abort();

            }

            catch (System.Exception ex)

            {

                Console.WriteLine(ex.Message);

                //Debug.Log(ex.Message);

            }

        }

…………

        void DataReceiveFunction()

        {

            try

            {

                while (canRecieveMsg)

                {

                    // 设定读取间隔

                    Thread.Sleep(25);//25ms读取一次

                    if (!sp.IsOpen)

                        return;

                    int datalength = sp.BytesToRead;

                    if (datalength == 0)

                    {

                        bytesLength = 0;

                        continue;

                    }

                    byte[] bytes = new byte[datalength];

                    bytesReceived = new byte[datalength];

                    bytesLength = datalength;

                    sp.Read(bytes, 0, datalength);

                    bytes.CopyTo(bytesReceived, 0);

                    //strReceived = System.Text.Encoding.Default.GetString(bytes);//将字节数组转换为字符串

                    textBox1.Text = BitConverter.ToString(bytesReceived);

                    if(controlMode == 0)

                    {

                        nowMODE.Text = "鼠标模式";

                        mousecontroller(bytesReceived);

                    }

                    else if(controlMode == 1)

                    {

                        nowMODE.Text = "键盘模式";

                         keyboardcontroller(bytesReceived);

                    }

                    //receivedData = bytesReceived.ToString();

                }

            }

            catch (System.Exception ex)

            {

                if (ex.GetType() != typeof(ThreadAbortException))

                {

                }

                Console.WriteLine(ex.Message);

                //Debug.Log(ex);

            }

        }

……

        private void mousecontroller(byte[] comIN)//控制鼠标

            if (comIN[0] == 0x05)//左键单击，按下数字键4

            {

                mouse\_event(MOUSEEVENTF\_LEFTDOWN | MOUSEEVENTF\_LEFTUP, 0, 0, 0, 0);

            }

            else if (comIN[0] == 0x02)//右键单击，2

            {

                mouse\_event(MOUSEEVENTF\_RIGHTDOWN | MOUSEEVENTF\_RIGHTUP, 0, 0, 0, 0);

            }

            else if (comIN[0] == 0x04)//鼠标向左移动，7

            {

                mouse\_event(MOUSEEVENTF\_MOVE, -moveSpeed, 0, 0, 0);//相对当前鼠标位置x轴分别移动50像素

                Thread.Sleep(moveSleep);//延时30ms

            }

            else if (comIN[0] == 0x01)//鼠标向右移动，3

            {

                mouse\_event(MOUSEEVENTF\_MOVE,  moveSpeed,0, 0, 0);//相对当前鼠标位置x轴分别移动50像素

                Thread.Sleep(moveSleep);//延时30ms

            }

            else if (comIN[0] == 0x07)//鼠标向上移动，6

            {

                mouse\_event(MOUSEEVENTF\_MOVE, 0, -moveSpeed, 0, 0);//相对当前鼠标位置x轴分别移动50像素

                Thread.Sleep(moveSleep);//延时30ms

            }

            else if (comIN[0] == 0x06)//鼠标向下移动，5

            {

                mouse\_event(MOUSEEVENTF\_MOVE, 0, moveSpeed, 0, 0);//相对当前鼠标位置x轴分别移动50像素

                Thread.Sleep(moveSleep);//延时30ms

            }

            else if (comIN[0] == 0x00)//模式切换为键盘，8

            {

                controlMode = 1;

                Thread.Sleep(200);//延时200ms

            }

        }

…………

\*省略部分\*

…………

    }

}

### 14.2 键鼠模拟程序说明（部分）

以下为程序中部分重要变量、线程、函数的简述：

### 变量说明

1. **portName**: 串口名称，默认为"COM5"。
2. **baudRate**: 波特率，默认为9600。
3. **parity**: 校验方式，设置为奇校验。
4. **dataBits**: 数据位，默认为8位。
5. **stopBits**: 停止位，设置为一个停止位。
6. **sp**: 串口对象，用于串口通信。
7. **dataReceiveThread**: 用于接收串口数据的线程对象。
8. **writeReceived**: 用于处理接收到数据的线程对象。
9. **canRecieveMsg**: 标志是否接收数据。
10. **strReceived**: 接收到的字符串数据。
11. **bytesReceived**: 接收到的字节数组。
12. **bytesLength**: 接收到的字节数组长度。
13. **IsOpenSerial**: 标志串口是否打开。
14. **receivedData**: 接收到的数据，默认为空字符串。
15. **moveSpeed**: 鼠标移动速度。
16. **moveSleep**: 鼠标移动延时。
17. **keybdSleep**: 键盘输入延时。
18. **controlMode**: 控制模式，0为鼠标模式，1为键盘模式。

### 线程说明

1. **dataReceiveThread**:
   * 用途: 接收串口数据。
   * 说明: 该线程在后台运行，通过串口对象读取数据，并将数据处理后进行显示或控制鼠标/键盘。
2. **writeReceived**:
   * 用途: 写入接收到的数据。
   * 说明: 该线程在后台运行，用于实时处理和显示接收到的数据。

### 函数说明

1. **Form1()**:
   * 用途: 构造函数，初始化组件和变量。
   * 说明: 初始化窗体组件，设置控件跨线程调用检查为false。
2. **Form1\_Load(object sender, EventArgs e)**:
   * 用途: 窗体加载时执行的函数。
   * 说明: 设置初始串口参数并实例化串口对象。
3. **ClosePort()**:
   * 用途: 关闭串口。
   * 说明: 尝试关闭串口，并终止相关线程。
4. **whileRunning()**:
   * 用途: 持续运行的循环，用于处理串口打开后的操作。
   * 说明: 在串口打开时持续运行，可用于调试或其他后台处理。
5. **connect\_Click(object sender, EventArgs e)**:
   * 用途: 连接按钮点击事件处理。
   * 说明: 打开串口并启动数据接收线程。
6. **textBox1\_TextChanged(object sender, EventArgs e)**:
   * 用途: 文本框内容改变事件处理。
   * 说明: 可用于处理文本框内容变化的逻辑。
7. **DataReceiveFunction()**:
   * 用途: 数据接收函数。
   * 说明: 线程循环读取串口数据，并根据控制模式调用相应的鼠标或键盘控制函数。
8. **clear\_Click(object sender, EventArgs e)**:
   * 用途: 清除按钮点击事件处理。
   * 说明: 清空文本框内容。
9. **DisConnect\_Click(object sender, EventArgs e)**:
   * 用途: 断开连接按钮点击事件处理。
   * 说明: 调用关闭串口函数。
10. **Form1\_FormClosing(object sender, FormClosingEventArgs e)**:
    * 用途: 窗体关闭事件处理。
    * 说明: 在窗体关闭时停止接收数据并关闭串口。
11. **stateTEXT\_TextChanged(object sender, EventArgs e)**:
    * 用途: 状态文本框内容改变事件处理。
    * 说明: 可用于处理状态显示的逻辑。
12. **mousecontroller(byte[] comIN)**:
    * 用途: 控制鼠标操作。
    * 说明: 根据接收到的数据执行鼠标点击或移动操作。
13. **keyboardcontroller(byte[] comIN)**:
    * 用途: 控制键盘操作。
    * 说明: 根据接收到的数据执行键盘按键操作。
14. **nowMODE\_TextChanged(object sender, EventArgs e)**:
    * 用途: 控制模式文本框内容改变事件处理。
    * 说明: 可用于处理模式显示的逻辑。