江 西 理 工 大 学

本 科 毕 业 设 计（论文）

题　　目：基于 KS16 的无线飞鼠设计

专题题目：

学　　院：电气工程与自动化学院

专　　业：自动化

班　　级：自动化 122 班

学　　号：20122961

学　　生：龙冠荣

指导教师：　韩树人　　职称：讲师

指导教师：　　职称：

时间：2016 年 6 月 5 日

摘　要

在各种人机交互设备，鼠标是最常用的一种设备，它以小巧，灵活受到很大的欢迎。但是目前市面流行使用的鼠标只能在一块较平整的平面才能正常工作，在一些特殊场合使用时将会受到限制。随着电子技术的飞速发展，各类半导体的芯片在性能不断提高的同时价格却越来越便宜，采用加速度陀螺仪传感器等相关传感器、RF 射频模块和微处理器构成的无线飞鼠，已经成为继光电鼠标之后的新一代鼠标发展的趋势。作为一种新的输入设备，无线飞鼠（空中鼠标）不需要借助任何平面，就可以直接在空间中实现鼠标的功能。

本文的主要任务是研究并设计一款无线飞鼠（空中鼠标）。在本文中，详细阐述了无线飞鼠软硬件系统的设计过程，同时介绍了飞鼠所采用的手势姿态解算算法。在这基础上实现了从加速度与陀螺仪传感器数据采集与处理，手势姿态解算，无线数据通过蓝牙 BLE 的收发，数据发送与接收的编解码，鼠标 USB HID 人机交互功能等一系列与飞鼠实现控制计算机过程中所使用到的关键技术难点，并且给出了系统的低功耗处理方法。

通过对系统的调试与功能、性能对比测试得出，此次飞鼠系统的设计是可行的。在设计中它具有功耗低、低成本，并具有高定位精度和平滑性的特点，使用的灵活性大大增强了。

关键词：KS16 与 KL25；姿态解算；四元数；蓝牙 BLE；MPU6050；USB HID 人机交互功能；低功耗；无线飞鼠

ABSTRACT

In a variety of interactive devices, a mouse is the most commonly used equipment, with its small, flexible popular. But currently the popular use of the mouse only in a relatively flat plane can work normally in the use of some special occasions will be limited. With the rapid development of electronic technology, all kinds of semiconductor the chip in the continuous improvement of performance and price is cheaper, the acceleration sensor gyroscope sensor, wireless flying RF module and microprocessor, has become a new generation of electric optical mouse development trend. As a new input device, a flying squirrel (wireless air mouse) do not need to use any plane, you can directly realize the mouse in the space.

The main task of this paper is research and design of a wireless flying squirrel (air mouse). In this paper, a detailed explanation of the flying squirrels wireless hardware and software system design process, and introduces the flying squirrel, the hand gesture algorithm. On the basis of the from the acceleration and gyro sensor data acquisition and processing, hand gesture solution, wireless data by Bluetooth transceiver, data transmitting and receiving, encoding and decoding Mouse USB HID human-computer interaction function series and flying squirrels to achieve control computer in the process of the use of the key technical problems, and the low power consumption of the system is given the processing method.

Through the debugging and function of the system, the performance comparison tests showed that, the flying squirrel system design is feasible. In design it has low power consumption, low cost, and has high positioning precision and smoothness of the characteristics and the flexibility of use greatly enhanced.

Keywords： KS16 and KL25；attitude algorithm；four element；Bluetooth BLE；MPU6050；USB HID interactive function；low power consumption；wireless mouse

目 录

第一章　绪论

1.1 课题来源与意义

鼠标作为最重要的人机交互设备之一，以其快捷、准确、直观的屏幕定位和控制能力，方便了用户与计算机之间的交互，简化了用户使用计算机各种强大功能的步骤[13]。传统的鼠标，无轮是机械鼠标还是光电鼠标，都需要一个桌面（平面）作为依托，才能够灵活的控制光标，极大的限制了使用的场所。随着智能电视，机顶盒，平板电脑等很多小型携带设备的兴起，传统的基于桌面（平面）控制方式的鼠标已不足以满足新的需求，人们需要一种在三维空间（空中）就可以控制设备。为了解决这个问题，得益于近年来无线与遥控技术、半导体及微电子技术等相关技术的发展，一种采用加速度与陀螺仪传感器等相关传感器、RF 射频模块和微处理器 MCU 构成的无线飞鼠（空中鼠标）系统应运而生。由于飞鼠（空中鼠标）可以通过功能的扩展，可以实现操作 PPT，玩 3D 体感游戏等，可以满足移动办公，娱乐的需要，具有较强的实用性。

飞鼠（空中鼠标）作为一种全新形式的人机交互设备，仅仅需要通过在三维立体空间中挥动飞鼠，就可以根据挥动的手势来实现准确控制计算机等设备了。其原理主要是飞鼠中内置的加速度与陀螺仪等相关传感器可以感知手势姿态（方向和速度）的变化，所以不需要像传统的键盘和鼠标那样，需要一个借助一个桌面（平面）才可以使用计算机等设备。飞鼠可以直接在三维空间中实现鼠标的功能，大大增强了灵活性，舒适性，具有现实的研究意义与实用价值。

1.2 国内外研究现状

自从 1968 年美国的 Douglas 博士设计了第一台鼠标以来，经过 45 多年的发展，总体而言，大概经历了以下三个阶段：

- 有线桌面鼠标

它的优点主要质量好灵敏度高，缺点为受线长限制离开桌面不能够灵活使用；

- 无线桌面鼠标

它的优点主要没有线的缠绕，使控制计算机等设备的距离大大增加，缺点也是离开桌面不能够灵活使用；

以上这两种类型的鼠标也经历了从机械滚轮到光电检测的的发展历程。

- 飞鼠空中鼠标

飞鼠（空中鼠标）是进入 21 世纪后逐渐兴起来的一个全新概念的鼠标，经过近 10 多来的发展，技术也越来越成熟，给用户带来了全新的体验。

对于飞鼠(空中鼠标)的设计，应用的技术一般有 RF 射频技术（包括 NRF、NFC、BLE 等），传感器检测技术等，采用的姿态解算方法有无损卡尔曼滤波，欧拉角与四元数，互补滤波等。而这种以微处理，加速度与陀螺仪传感器，RF 射频为硬件基础的飞鼠（空中鼠标）在国内外均有一定的研究基础。

国外的一些高校，早在 2001 年，美国密歇根大学的 Seungbae. Lee 首次提出将用两轴加速度传感器应用在鼠标中，并且设计出基于两轴加速度传感器的鼠标；2003 年，英国的伯明翰大学设计了一款可以在电脑 3D 环境下使用的三维鼠标[13]。

国内也有相应的研究，一些开源的社区，如正点原子里面就有很多发烧友对飞鼠（空中鼠标）研究。许多高校如华侨大学、湘潭大学、大连理工大学，武汉大学，武汉理工大学等也相继提出一系列基于加速度或陀螺仪的空中鼠标系统的设计方法。例如上海交大的姜晓波与钱莉博士提出的基于加速度传感器技术空中鼠标的研究，不但讨论了基于微加速度传感器的无线鼠标的软硬件系统设计和构成，还给出了 Matlab 中的 simulink 模型和算法研究[13]；武汉理工大学的成虎超硕士还研究一款出空中鼠标，不但可以操作计算机，还可以遥控机顶盒与智能电视等，同时给出了基于安卓系统的空中鼠标的设计与实现；武汉大学江朝强，石睿等人提出一款以 MEMS 技术为基础的指环式三维无线鼠标，还给出一些低功耗处理的方法思路。

在商业上，加拿大 Deanmark 公司首次提出了空中鼠标的这一概念，现在市面上比较流行与用户体验较好的产品以国外罗技公司的 Mx Air 系列空中鼠标为代表，国内的产品有深圳享爱数码的 Fly Mouse 系列，乐帆飞鼠系列，双飞燕公司飞鼠系列的产品等。

采用加速度、陀螺仪等相关传感器、RF 射频模块和微处理器 MCU 为硬件基础而构成的无线飞鼠（空中鼠标）系统，因其具有智能化、低成本、低功耗、并具有高定位精度和平滑性的特点，已经成为继光电鼠标之后的新一代鼠标发展的趋势。

1.3 研究目标及主要内容工作

1． 研究目标及系统的设计实现过程

本文研究的是基于主控制器为 KS16 的无线飞鼠的实现。研究的目标与功能主要有以下几个部分：

1、飞鼠手持端需要实现手部动作识别和数据格式编码，无线 BLE 传输；无线转 USB 接收端要实现无线接收和 USB HID 设备的识别；

2、电脑光标的变化需要跟随手部挥动姿势的变化，做到手部动作跟随性好，

具有较高的平滑性；

3、系统采用低功耗设计；

4、可以进行功能扩展，实现键鼠一体化，可以操作 PPT 等。

为实现系统的这些目标与功能，整个系统的设计过程主要拆分为以下几个部分来展开：

1、先对系统的设计要求进行细致的分析，进而勾略出飞鼠要实现的功能；

2、查阅资料，结合理论知识，根据飞鼠需要实现的功能，确定出飞鼠整个系统设计中可能遇到的难点，描绘出飞鼠系统的设计思想；

3、在确定设计思想，结合自身与公司的情况，确定实现飞鼠所需使用的硬件，在硬件基础上考虑使用合适的方案去实现飞鼠系统；

4、确定系统方案后，分别进行系统软硬件设计的实现；

5、系统软硬件设计完成后，首先对飞鼠的各个软硬件部分进行单元调试，单元调试通过后再对整个飞鼠系统进行系统级的、全面性的调试。在所有调试通过的基础上，最后进行飞鼠系统功能与性能的测试。

2． 研究内容及主要工作

根据上一小节所述，本文按照以下编排结构及章节内容介绍了整个研究的过程及研究的主要内容：

第一章绪论详细的介绍了这次毕设课题的来源与研究的意义，概括了国内外的研究现状以及飞鼠实现的功能目标与研究的内容；

第二章系统构思及分析详细介绍无线飞鼠的设计思想，以此来确定飞鼠系统的实现方案与硬件基础，最后确定飞鼠实现的性能指标及技术难点分析；

第三章详细介绍了飞鼠手势姿态解算的基础理论；

第四章硬件系统设计主要介绍了飞鼠硬件系统每个硬件部分之间的关系。同时详细的阐述了硬件设计的方法，包括每一部分的原理图设计及 PCB 设计;

第五章软件系统设计主要介绍了飞鼠软件系统的结构框图及程序流程图，同时还详细各个子软件部分的具体实现；

第六章系统调试与测试阐述了软硬件调试的方法及介绍了一些系统功能与性能的单元测试用例；

第七章是总结与展望，主要归纳了这次飞鼠系统的设计工作及论文编写的总结，同时还在展望中提出了一些改进之处。

1.4 本章小结

本章主要概括的介绍了本课题的国内外研究现状，可以得出，现在鼠标正在

朝着多键鼠标、3D 鼠标等新型鼠标发展，它已经替代光电鼠标成为新一代鼠标的发展趋势。除此之外，还介绍了这一课题的来源与研究意义，总结性的描述了系统设计的功能与目标，同时还对主要研究内容及工作进行了详细的叙述。

第二章　系统构思及分析

2.1 设计思路

无线飞鼠设计思想的核心，是通过加速度与陀螺仪检测出手部运动的方向，再通过姿态解算算法(四元数)解算出手势变化，从而让光标随着手势的变化而变化。由于 MPU6050 内部支持数字运动处理，在姿态解算方面十分有优势。它的设计思想如下图 2.1 所示。

图 2.1　无线飞鼠设计思想

2.2 性能指标

本次飞鼠的设计，最主要的性能指标有三个，如下所描述所述：

1、平滑性好，有较好的用户体验；

2、系统采用低功耗设计；

3、鼠标抗静态漂移；

4、键鼠一体化，既可以用于鼠标操作，也可以用于实现键盘的常用功能。

2.3 技术分析

在这次飞鼠的设计当中，发现此次飞鼠设计中主要有几个设计难点，会影响 2.2 小节性能指标的实现，如下所示：

1、按键按下时防止鼠标光标抖动；

2、手势姿态数据传输的时滞解决；

3、USB HID 人机交互功能的实现；

4、蓝牙 BLE 协议栈的运用以及蓝牙串口透传的实现；

5、硬件系统设计的合理性，这主要关系到低功耗的实现。

2.4 本章小结

根据前面本章前面三个小节的描述，得出飞鼠（空中鼠标）的系统设计是以微控制器，加速度与陀螺仪传感器，RF 射频芯片为核心硬件，无线传输采用蓝牙 BLE 技术来设计实现的。飞鼠手持端整个系统采用低功耗设计，以便使其能工作更长的时间。所以飞鼠手持端采用飞思卡尔 KS16 芯片作为主控制器，这款主控芯片具有 9 种功耗模式，在功耗控制管理方面表现卓越。而用于手势姿态检测的传感器方面，选择整合有陀螺仪与加速度于一体的传感器芯片 MPU6050，这款芯片在市面上价格较为实惠，而且内部支持了数字运动处理引擎—DMP 库，库的核心算法为用四元数表示飞鼠手势姿态，即三个方向的欧拉角。而蓝牙无线传输芯片采用昆天科的 QN902X，该芯片无线传输时消耗的电流最多为 9 毫安，静态电流低可致 2uA～4 uA，功耗表现方面十分优异。同时该芯片的蓝牙应用开发提供蓝牙协议栈，可以加快蓝牙透传的开发。而无线转 USB 接收端的主控采用 KL25 采用主控制器，用以实现 USB HID 人机交互功能，无线数据接收依然采用昆天科的蓝牙芯片 QN902X。具体的硬件选型如下表 2-1 所示。

表 2-1　硬件选型表

硬件模块	芯片	原因
电源	3.7V 锂电池	可充电多次使用，经济环保
稳压芯片	LDO 稳压芯片 (SP6201)	3.7V 电源输出输出 3.3V 给主电路供电，压降低，静态时电流为 28 微安，进而损耗功率低
蓝牙	QN902X	M0 内核，功耗低，可以利用蓝牙协议栈开发
发射端主控	KS16	利用 AMetal 软件包可缩短开发周期
传感器	MPU6050	整合加速度与陀螺仪，姿态解算更方便
接收端主控	KL25	带有 USB 控制器，可实现 HID

对于按下时光标抖动的问题，可以通过按键按下时给发给光标的位移清零来实现，对于传输的时滞性，可以通过减少蓝牙每次连接事件发生时间间隔来增强

实时性。

综上所述，采用 KS16 作为飞鼠手持端的主控制器，采用 MPU6050 加速度陀螺仪传感器，无线转 USB 接收端采用 KL25 作为主控制器，而无线数据的收发采用 QN902X 芯片为硬件基础的设计方案足以满足系统目标的设计与系统功能的实现。

第三章　飞鼠姿态解算基础理论

3.1 坐标系及姿态角表示

1． 坐标系表示

在运动力学中，对于一个物体的运动，为了方便研究，通常可以抽象成一个坐标系来表示这个物体，这个运动的物体可以是飞鼠（空中鼠标），飞行航模等。由于物体的运动是相对的，所以描述一个的物体的运动姿态，必须指明一个参考的坐标系才能很好的描述物体的运动状况。根据惯性导航理论，这里主要描述两个在导航与姿态解算中常用的坐标系。

(1) 地理坐标系

地理坐标系是指它的原点是位于运载体所在的地球表面的坐标系。它的原点 O 选取在物体的重心。对于地理坐标系来说，坐标轴取向的不同决定了其坐标系的名称有不同的命名方法来贴切表示，如东北天、北东地等。如下图 3.1 所示，X 轴沿当地纬线指向东，Y 轴沿当地子午线指向北，Z 轴沿着当地地理垂线的指向并与 X，Y 轴构成右手直角坐标系。

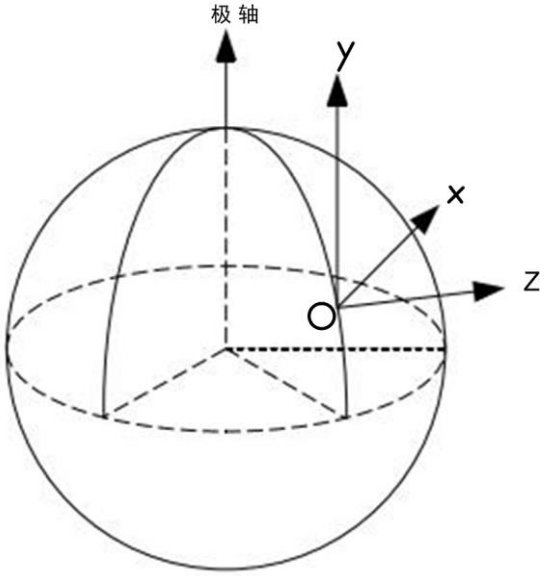

图 3.1　地理坐标系

(2) 参考坐标系

参考坐标系用于描述这个物体相对于它运动时的状态信息。一般来说，对于导航及物体的空间三维姿态来说，参考坐标系一般选择可以为地理坐标系。

(3) 机体坐标系

机体坐标系的原点 O 与其物体的质心重合。对于飞鼠、飞行航模、轮船等载体，X 轴与载体横轴方向重合并且指向载体的右方，Y 轴与载体纵轴方向重合并

且指向载体的前方，Z 轴和载体竖轴方向重合且指向载体的上方。X 轴、Y 轴和 Z 轴构成右手坐标系。如下图 3.2 中的坐标系所示。

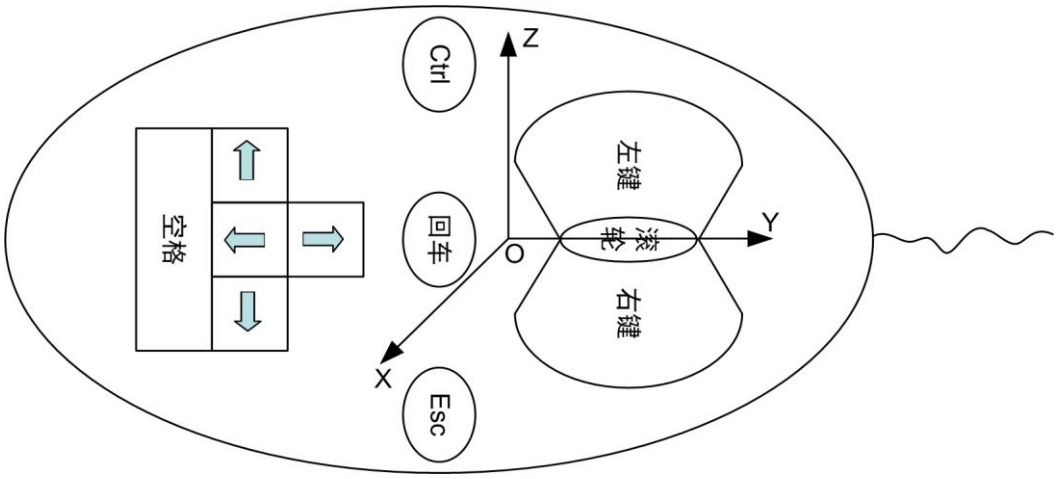

图 3.2　机体坐标系

2. 飞鼠三维姿态角定义

飞鼠的三维姿态角是根据机体坐标系相对参考坐标系的转角来表示是一种常用的表示方法。若飞鼠的参考坐标系为地理坐标系，位于机体坐标系中的飞鼠分别绕机体坐标系的 Z 轴旋转 ψ 角度、X 轴旋转 θ 角度、Y 轴旋转 γ 角度，可以定义为飞鼠（手势）空间姿态角，分别是：航向角 Yaw 、俯仰角 Pitch 和 横滚角 Roll。如下图 3.3 的红色标志部分所示。

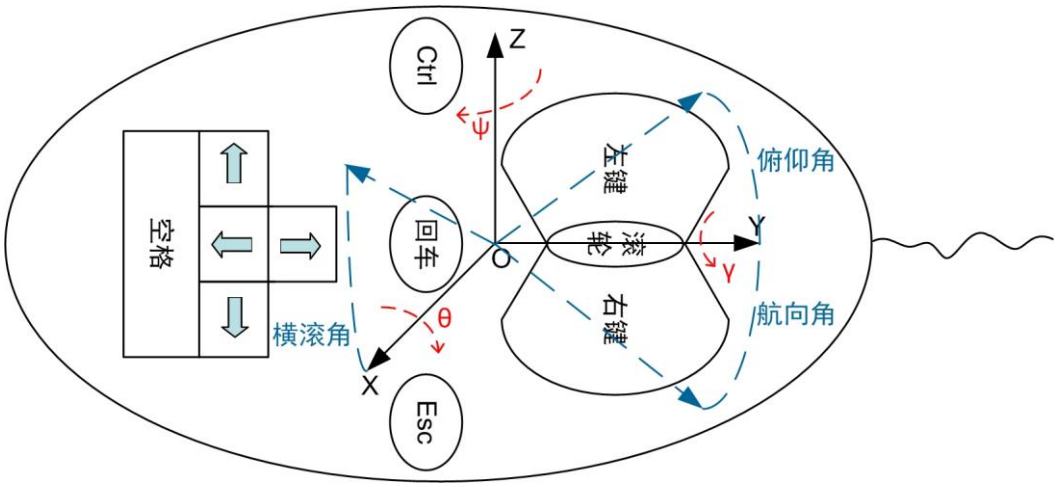

图 3.3　姿态角定义

飞鼠的三个姿态角粗略表示如上图 3.3 青蓝色标记部分，值得注意的是图中每一个姿态角的表示都是在只有一个轴转动，而其它轴保持不变的情况下表示的，每姿态角的含义如下所示：

1、航向角（Yaw）：飞鼠绕机体坐标系 Z 轴旋转后在水平面平面内投影与纵向轴 Y 之间的夹角，夹角范围为 -180°~180°；

2、俯仰角（Pitch）：飞鼠绕机体坐标系 X 轴旋转后飞鼠的纵向 Y 轴在当地水平面(这个水平面可认为相对于参考坐标系的水平面来说）之间的夹角，夹角范围为-90°~90°；

3、横滚角（Roll）：飞鼠绕机体坐标系 Y 轴旋转后飞鼠的横向 X 轴在当地水平面(这个水平面可认为相对于参考坐标系的水平面来说）之间的夹角，夹角范围为-180°~180°。

3.2 姿态解算与互补滤波原理

1. 姿态解算原理

对于三维空间一个确定的矢量，用不同的坐标系表示它时，所表示的大小和方向一定是相同的。但是由于这两个坐标系的姿态旋转矩阵存在误差，那么当一个矢量经过这么一个有误差存在的旋转矩阵后，在另一个坐标系中的得到实际值肯定和理论值是有偏差的，我们可以用这个误差不断的来修正这个旋转矩阵。如果这个旋转矩阵的元素是四元数，若修正了四元数，姿态矩阵也就被修正了。

2. 互补滤波原理

陀螺仪传感器的高频时比较灵敏，有较好的动态响应特性，但积分计算姿态角度时会产生累积误差，加速度传感器与地磁传感器测量姿态时没有产生累积误差，静态时测量值比较准确，但高频燥声干扰很严重，动态响应较差，为此它们两者在频域上特性刚好可以互相校正。因此这三种传感器测量出来的数据可以采用互补滤波器进行融合，进而达到消除干扰，提高测量精度、平滑度和系统的动态性能的目的。

3.3 姿态解算算法及姿态矩阵

1. 姿态解算算法介绍

姿态解算常用的算法有无损卡尔曼滤波，欧拉角法，四元数法等。无损卡尔曼滤波算法理解与运用起来很是费劲。欧拉角法存在这样一种情况，在其姿态解算的过程中存在奇点，不能用于全姿态解算。而四元数法，则不存在奇点这种情况，其运算量小，解算速度比较快，可以满足飞鼠在运动过程中的姿态实时解算。所以本文采用的算法为四元数姿态解算算法。

2. 姿态矩阵推导

根据惯性导航捷联式惯导系统理论可知，在三维空间姿态解算与测量系中，通常采用方向余弦矩阵来对机体坐标系和地理坐标系之间的这种相对转动关系进行数学现物理方程的表述，方向余弦矩阵也就是所谓的姿态矩阵。

所以飞鼠的手势姿态解算也可以参考捷联惯导的理论，它的手势姿态可以是根据机体坐标系相对于参考坐标系（地理坐标系）相对应轴旋转一定的角度来确

定，如下图 3.4 所示，也就是分别是绕地理坐标系的 Z 轴转 ψ 角度，X 轴转 θ 角度，Y 轴转 γ 角度来确定。其中 n 坐标系为地理坐标系（即飞鼠的参考坐标系），三个轴的方向分别指向东，指向北，指向天，b 坐标系是与飞鼠固联的机体坐标系，三个轴的方向指向机体右方，指向机体前方，指向机体上方。

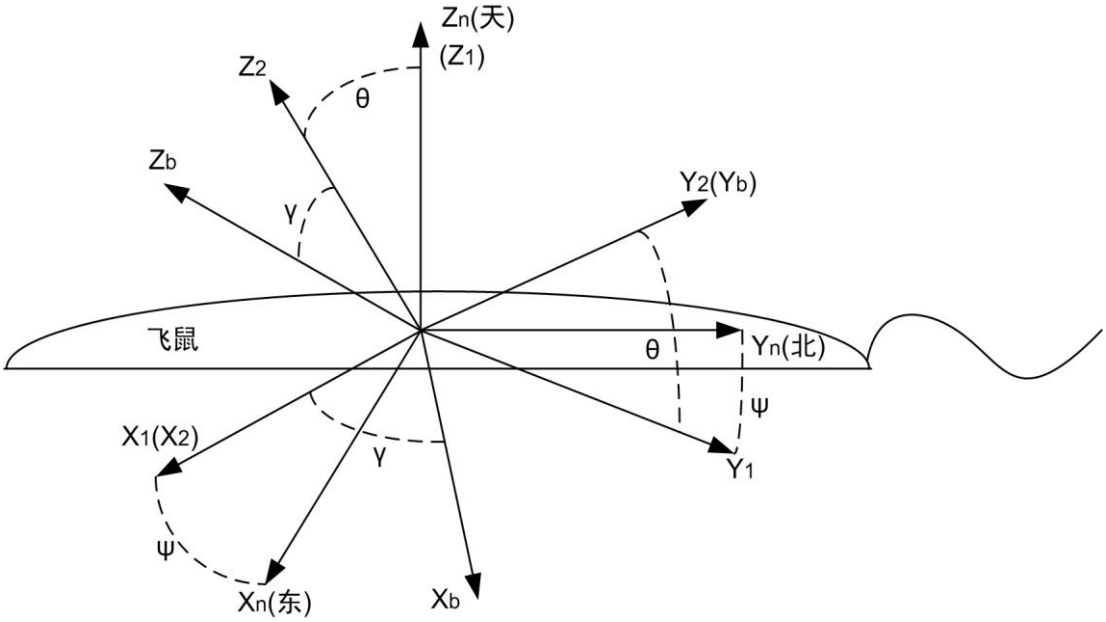

图 3.4　飞鼠姿态演变

对于上图 3.4 中所示，刚开始时，机体坐标系 b 和参考坐标系 n(地理坐标系)是重合的，飞鼠从参考坐标系按照下图 3.5 所示面三次先后顺序进行旋转，最终得到新的坐标系，即机体坐标系。

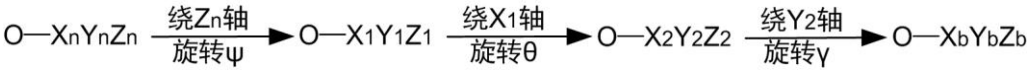

图 3.5　姿态旋转方向

这样，通过改变 ψ，θ，γ 这三个角，就可以任意将地理坐标系和机体坐系两者进行数学变换。三次顺序旋转所对应的变换矩阵依次分别为：

$$C_n^1=\begin{bmatrix}\cos\Psi & -\sin\Psi & 0\\ \sin\Psi & \cos\Psi & 0\\ 0 & 0 & 1\end{bmatrix}\quad C_1^2=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & \sin\theta\\ 0 & -\sin\theta & \cos\theta\end{bmatrix}\quad C_2^b=\begin{bmatrix}\cos\gamma & 0 & -\sin\gamma\\ 0 & 1 & 0\\ \sin\gamma & 0 & \cos\gamma\end{bmatrix}$$

根据上面这三个公式可得机体坐标系与参考坐标系之间姿态变换矩阵为如下公式(3-1)所示。

$$C_n^b=C_2^bC_1^2C_n^1=\begin{bmatrix}\cos\gamma & 0 & -\sin\gamma\\ 0 & 1 & 0\\ \sin\gamma & 0 & \cos\gamma\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & \sin\theta\\ 0 & -\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}\cos\Psi & -\sin\Psi & 0\\ \sin\Psi & \cos\Psi & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$=\begin{bmatrix}\cos\gamma\cos\Psi+\sin\gamma\sin\Psi\sin\theta & -\cos\gamma\sin\Psi+\sin\gamma\cos\Psi\sin\theta & -\sin\gamma\cos\theta\\ \sin\Psi\cos\theta & \cos\Psi\cos\theta & \sin\theta\\ \sin\gamma\cos\Psi-\cos\gamma\sin\Psi\sin\theta & -\sin\gamma\sin\Psi-\cos\gamma\cos\Psi\sin\theta & \cos\gamma\cos\theta\end{bmatrix} \quad (3\text{-}1)$$

注意，姿态矩阵的表现形式与旋转顺序有关，不同的旋转顺序代表不同的三维姿态空间，所以它的姿态转换矩阵的表现形式就会有所不同。

记 $C_b^n=\begin{bmatrix}T_{11} & T_{12} & T_{13}\\ T_{21} & T_{22} & T_{23}\\ T_{31} & T_{32} & T_{33}\end{bmatrix}$，由于参考坐标系（n 系）至机体坐标系（b 系）的旋转过程中坐标系始终保持直角坐标系，所以 C_b^n 为正交矩阵

$$C_n^b=(C_b^n)^T=\begin{bmatrix}T_{11} & T_{21} & T_{31}\\ T_{12} & T_{22} & T_{32}\\ T_{13} & T_{23} & T_{33}\end{bmatrix} \quad (3\text{-}2)$$

比较公式(3-1)与公式(3-2)，可得

$$\begin{cases}\theta=\arcsin(T_{32})\\ \gamma=\arctan\left(-\dfrac{T_{31}}{T_{33}}\right)\\ \Psi=\arctan\left(\dfrac{T_{12}}{T_{22}}\right)\end{cases} \quad (3\text{-}3)$$

至此，可以通过式(3-3)计算出飞鼠的手势姿态的三个角度，在本文的设计中，只用到了航向角 Yaw，用于光标的左右位移，俯仰角 Pitch，用于光标的上下位移。

3. 四元数与姿态矩阵之间的关系

由 4 个元构成的数被称为四元数，如式(3-4)所示，其中 q_0,q_1,q_2,q_3 表示实数，i,j,k 既是互相正交的的单位向量，又是虚单位 $\sqrt{-1}$ 。

$$Q(q_0,q_1,q_2,q_3)=q_0+q_1i+q_2j+q_3k \quad (3\text{-}4)$$

根据四元数的表示方法可知，任何三维立体空间的向量可以等效为实部为零的四元数。实部为零的四元数可以被等效成是三维立体空间向量在四维空间中的一个映像。所以可以用四元数的性质和方法来研究三维空间。

由于 n 系和 b 系均为直角坐标系，各轴之间始终保持直角，所以可以将坐标系理解成刚体，当只研究两个坐标系间的角位置关系时可对一个坐标系作平移，使其原点与另一个坐标系的原点重合[8]。因此两个坐标系间的空间角位置关系可

以理解成刚体的定点转动。所以说刚体可以通过绕一定点的某个轴的特定角度可达到任何姿态，四元数Q代表了绕该点转动的角度与轴向。

如果矢量 R 相对参考坐标系旋转，并且该旋转可以用四元数 Q 表示，新矢量为 R’， 则 R 与 R’之间的变换可以表示成下述四元数运算，如式(3-5)所示，其中，矢量 R 在这里就可以被当成一个实数部分q_0为零的四元数。

$$R' = Q \otimes R \otimes Q^{-1} \quad (3\text{-}5)$$

根据式(3-5)可以看出，矢量 R 相对于参考坐标系转动了一个四元数Q，得到一个新的坐标系，且矢量 R 在新坐标系的投影为 R’，则可以计算出四元数与方向余弦矩阵的关系表达式，如式(3-6)所示，其中分别为方向余弦，四元数，欧拉角用于表示姿态矩阵时不同的表示形式。

$$C_b^n = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\gamma\cos\Psi+\sin\gamma\sin\Psi\sin\theta & \sin\Psi\cos\theta & \sin\gamma\cos\Psi-\cos\gamma\sin\Psi\sin\theta \\ -\cos\gamma\sin\Psi+\sin\gamma\cos\Psi\sin\theta & \cos\Psi\cos\theta & -\sin\gamma\sin\Psi-\cos\gamma\cos\Psi\sin\theta \\ -\sin\gamma\cos\theta & \sin\theta & \cos\gamma\cos\theta \end{bmatrix} \quad (3\text{-}6)$$

上述分析说明，如果 n 系（参考坐标系）至 b 系（机体坐标系）的旋转四元数 Q 已确定，则按式(3-6)可计算出姿态矩阵C_b^n，再按式(3-3)可确定出机体（飞鼠）的航向角 Yaw、俯仰角 Pitch 和横滚角 Roll。因此，四元数 Q 包含了飞鼠运动过程中所有的姿态信息，所以在捷联惯导理论中的姿态解算实质上是如何计算四元数 Q。

对于四元数 Q 表示的姿态矩阵的求解，可以用龙格库塔法或毕卡法等实时更新四元数，具体的求解方法可以参考秦永元编写的《惯性导航》一书中的第九章的内容。

3.4 四元数姿态解算步骤

经过前面的分析，大概明白了用四元数来怎么表示其姿态矩阵，那么怎么用四元数来求解姿态角度，主要是经过以下几个步骤：

(1) 加计重力加速度归一化

把加速度计的三维向量单位化称为加速度计数据归一化。因为是单位矢量到参考性的投影，所以要把加速度传感器测出来的数据单位化。本质上加计向量归一化后其方向并没有改变，仅仅改变了这三个向量的长度，归一化的目的之一可

以说是为了与单位四元数对应。

(2) 提取四元数的等效余弦矩阵中的重力分量

利用四元数将地理的重力加速度旋转至飞行器上面为v_x,v_y,v_z，再与加速度传感器读出来的值a_x,a_y,a_z（已做归一化处理）做外积，得出误差；

(3) 向量叉积得出姿态误差

a_x,a_y,a_z是机体坐标系上加速度传感器实际测出来的重力向量，而v_x,v_y,v_z是根据以陀螺仪传感器测出来的角速度积分后的姿态矩阵来推算提取出来的重力向量，它们都表示机体坐标系上的重力向量。那它们之间的误差向量，就是陀螺仪角速度积分后从姿态矩阵里面提取出来的地理坐标系的重力向量到机体坐标系上的重力分量与机体坐标系加速度计测出来重力向量这两者之间的姿态误差。向量误差可以向量叉积来表示，e_x,e_y,e_z就是它们两者之间的向量叉积误差。

可知，得出来的向量叉积误差其大小与陀螺积分误差成正比，更为重要的是向量叉积误差e_x,e_y,e_z依然位于机体坐标系上，所以用这个向量叉积误差对陀螺累积误差的纠正就直接体现在对机体坐标系的纠正了。也就是说，如果陀螺按这个叉积误差的轴，转动叉积误差的角度（也就是转动三轴投影的角度）那就能把加计和重力向量的误差消除掉（具体可看向量叉积的定义）。如果完全按两者向量叉积误差转过去，那就是完全信任加速度传感器测出来的数据。如果一点也不转，那就是完全信任陀螺仪传感器测出来的数据。

(4) 对误差进行积分

对从当前姿态矩阵中把地理坐标系重力向量提取出来至机体坐标系中的重力分量，与当前加速度计测得的重力向量的差值进行积分消除误差。

(5) 互补滤波，姿态误差补偿到角速度上，修正角速度积分漂移

四元数的q_0,q_1,q_2,q_3系数不停地被陀螺积分更新，也不停地被误差修正，它和公式所代表的姿态也在不断更新。将该误差e_x,e_y,e_z输入 PI 控制器（互补滤波器）后与本次姿态更新周期中陀螺仪测得的角速度值相加，最终得到一个修正的角速度值，将其输入四元数微分方程，更新四元数。

(6) 解四元数微分方程，更新四元数姿态矩阵

利用龙格库塔法，可以求解四元数微分方程，实时更新四元数的姿态矩阵。

(7) 四元数归一化

描述刚体旋转的四元数是规范化的四元数，但是由于计算时产生的精度误差等因素，四元数在计算过程中会逐渐失去规范化特性，因此必须重新对四元数做规范化处理。

(8) 求取姿态角

利用式(3-7)，求取姿态角。

$$\begin{cases} \theta = \arcsin(-2q_1q_3 + 2q_0q_2) \\ \gamma = \arctan((2q_2q_3 + 2q_0q_1)/(-2q_1^2 - 2q_2^2 + 1)) \\ \Psi = \arctan((2q_1q_2 + 2q_0q_3)/(-2q_2^2 - 2q_3^2 + 1)) \end{cases} \quad (3\text{-}7)$$

算法实现代码详见程序清单 3-1 所示,其中程序清单中代码注释的编号对应上面的四元数姿态解算的步骤。

程序清单 3-1　四元数姿态解算算法

```
/* 加速度校正陀螺仪 */
void am_imu_ag_updata (float gx, float gy, float gz, float ax, float ay, float az)
{
    float norm;
    float vx, vy, vz;
    float ex, ey, ez;

    float q0q0 = q0*q0;
    float q0q1 = q0*q1;
    float q0q2 = q0*q2;
    float q1q1 = q1*q1;
    float q1q3 = q1*q3;
    float q2q2 = q2*q2;
    float q2q3 = q2*q3;
    float q3q3 = q3*q3;

    if (ax * ay * az == 0) {
        return;
    }

    /* (1) 测量正常化 把 MPU6050 加速度计的三维向量转成单位向量     */
    arm_sqrt_f32(ax * ax + ay * ay + az * az, &norm);
    ax = ax / norm;
    ay = ay / norm;
    az = az / norm;

    /* (2) 提取重力向量的分量到机体坐标系上                          */
    vx = 2 * (q1q3 - q0q2);
    vy = 2 * (q0q1 + q2q3);
    vz = q0q0 - q1q1 - q2q2 + q3q3;

    /*
     * (3) 向量叉乘，得到机体坐标系的合加速度（这里是 MPU6050 的加速度的测量值与地理坐标系
     * 到机体坐标系的加速度理论值）的大小与方向, 因为 MPU6050 测量的数值不包括重力的
```

```
     */
    ex = (ay * vz - az * vy);
    ey = (az * vx - ax * vz);
    ez = (ax * vy - ay * vx);

    /* (4) 计算和应用积分反馈                                    */
    exInt = exInt + ex * Ki ;
    eyInt = eyInt + ey * Ki ;
    ezInt = ezInt + ez * Ki ;

    /* (5) 校正陀螺仪测量值, 用叉积误差来做 PI 修正陀螺累积误差 */
    gx = gx + Kp * ex + exInt;
    gy = gy + Kp * ey + eyInt;
    gz = gz + Kp * ez + ezInt;

    /* (6) 整合四元数率, 四元数微分方程, 四元数更新算法，一阶龙格库塔法 */
    q0 = q0 + (-q1*gx - q2*gy - q3*gz)* halfT;
    q1 = q1 + (q0*gx + q2*gz - q3*gy) * halfT;
    q2 = q2 + (q0*gy - q1*gz + q3*gx) * halfT;
    q3 = q3 + (q0*gz + q1*gy - q2*gx) * halfT;

    /* (7) 四元数规范化 */
    arm_sqrt_f32(q0*q0 + q1*q1 + q2*q2 + q3*q3, &norm);
    q0 = q0 / norm;
    q1 = q1 / norm;
    q2 = q2 / norm;
    q3 = q3 / norm;

    /* (8) 转换为欧拉角 */
    g_angle.Pitch = asin(-2 * q1 * q3 + 2 * q0* q2)* 57.3f;                    /* pitch 俯仰角 */
    g_angle.Roll  = atan2(2 * q2 * q3 + 2 * q0 * q1,   -2 * q1 * q1 - 2 * q2 * q2 + 1)* 57.3f;   /* roll 横滚角 */
#if 0
g_angle.Yaw    = -atan2(2 * q1 * q2 + 2 * q0 * q3, -2 * q2 * q2 - 2 * q3 * q3 + 1)* 57.3f; /* yaw 偏航角   */
#endif /* 0 */
    g_angle.Yaw    += gz * 57.3 * 0.0025;
}
```

3.5 本章小结

四元数法求解飞鼠的姿态时，计算量比较小，仅仅需要求解四个一阶未知量的线性微分方程，而不需要很多复杂的三角运算与矩阵运算，从而姿态解算的速度较快。因此，得到飞鼠姿态实时性更能反应真实的运动情况，可以得到更好的控制效果。与此同时，四元数在姿态解算过程中只需要进行简单的四元数规范化

处理就能保证姿态矩阵的正交性，是工程运用中一种比较实用的姿态解算算法。而本次采用的陀螺仪加速度传感器为 InvenSense 公司的一款有六个自由度的传感器芯片 MPU6050，这个传感器芯片内部自带支持运动数字处理，只需要移植好官方的 DMP 库，就可以很快得到想要的姿态数据。而 DMP 库的核心理论类似于四元数姿态解算这一种算法，它可以将原始的加速度与陀螺仪数据在库里面运算完成，然后直接转换成四元数输出。故可以很方便的计算出欧拉角，也就是得到了航向角 Yaw，俯仰角 Pitch，横滚角 Roll，即飞鼠的手势姿态。

第四章　系统硬件设计

4.1 硬件系统模块框架设计

飞鼠硬件系统主要是两个部分，飞鼠手持发射端硬件设计与无线转 USB 接收端硬件设计两大部分。本文中，飞鼠手持端的硬件系统电路设计主要包括锂电池充电电路的设计，系统稳压供电电路设计，KS16 最小系统电路的设计，传感器与矩阵键盘电路的设计与蓝牙发射模块电路设计。无线转 USB 接收端由于主控内带 KL25 稳压 3.3V 电源输出，所以硬件系统电路设计主要包括 KL25 最小系统电路的设计，USB 外围电路的设计（包括 USB 供电与 USB 数据传输两部分）以及与蓝牙接收模块电路的设计。整个硬件系统的原理框图如下图 4.1 所示。

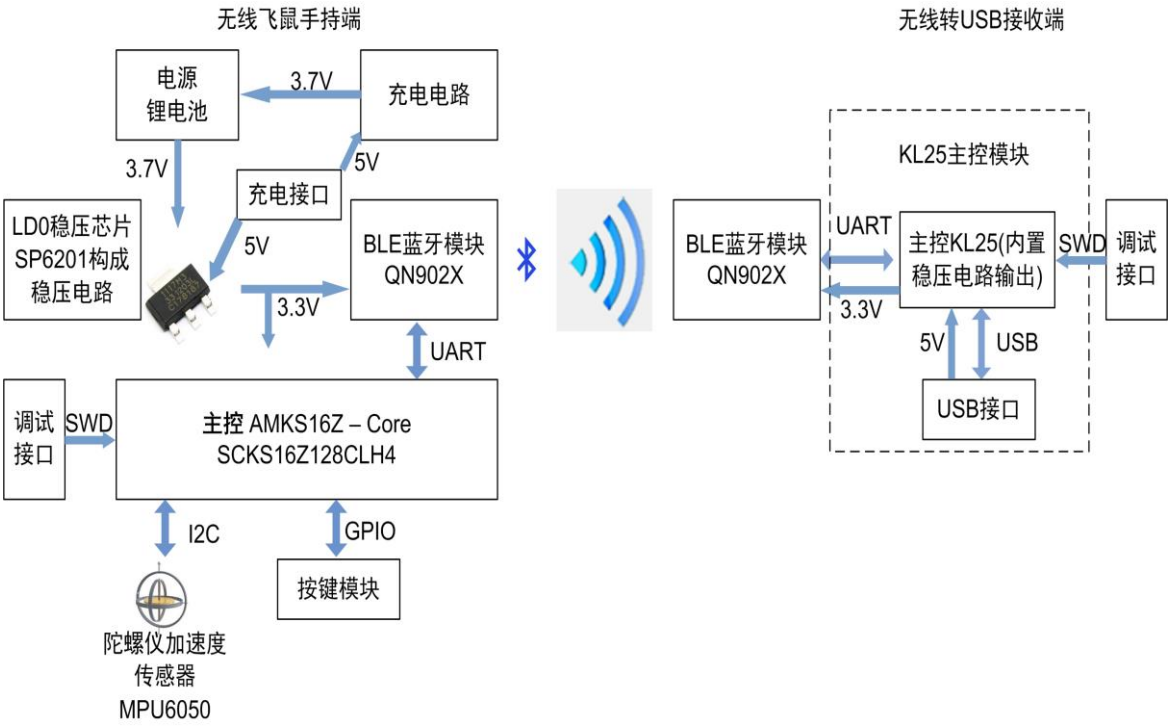

图 4.1　硬件系统原理框图

4.2 模块硬件电路设计与分析

4.2.1 充电电路与稳压电路设计

充电电路采用 TI 公司的 BQ24040 芯片，稳压电路采用 SP6201 作为稳压芯片，这两种芯片具有稳定性高，静态电流小的特点，具体电路原理图设计如下图 4.2 所示。

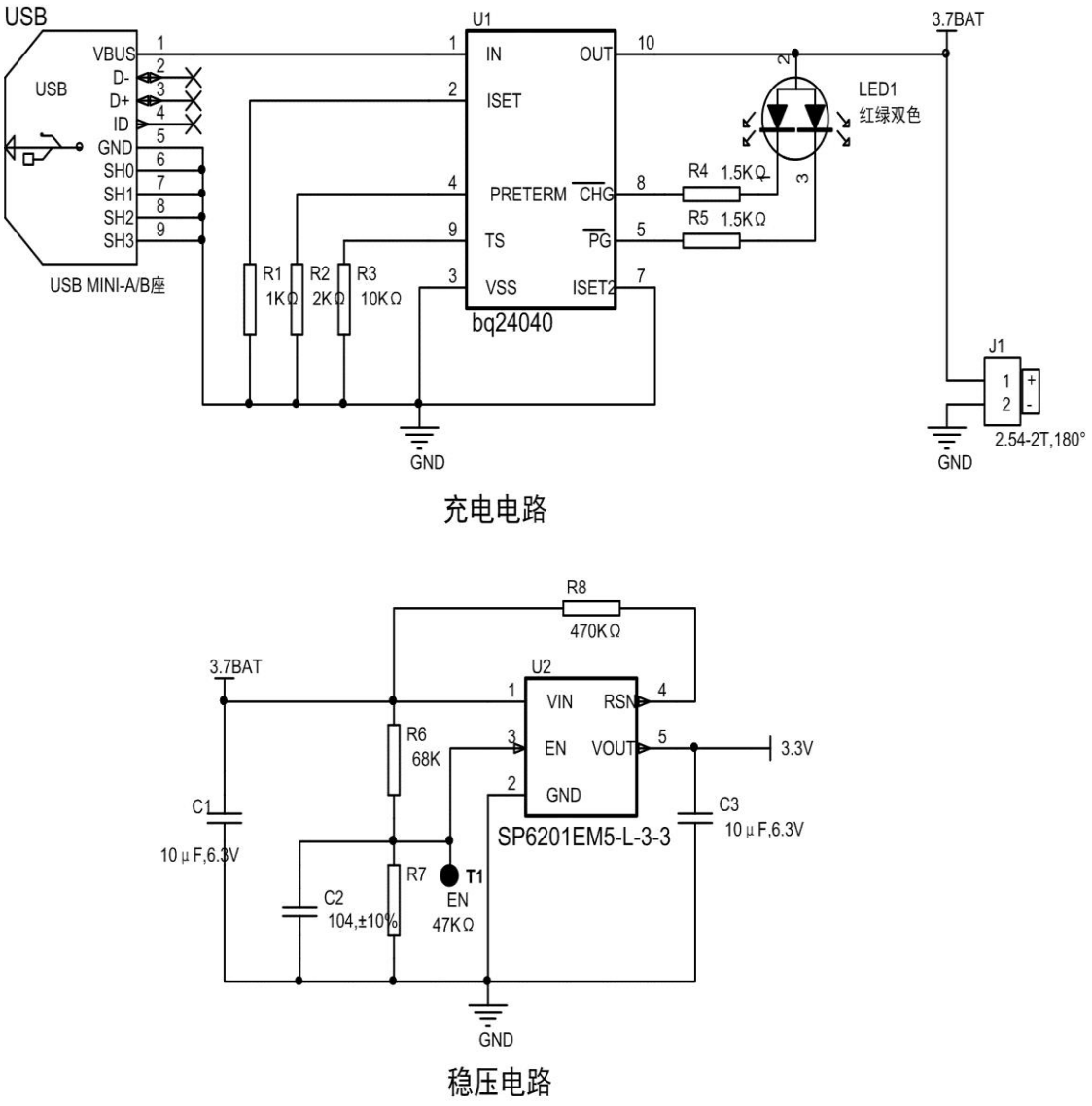

图 4.2　充电电路与稳压电路设计图

4. 2. 2 KS16 最小系统电路设计

主要包括有复位电路，调试接口电路，为了降低功耗，没有使用外部晶振。原理图设计如下图 4.3 所示。

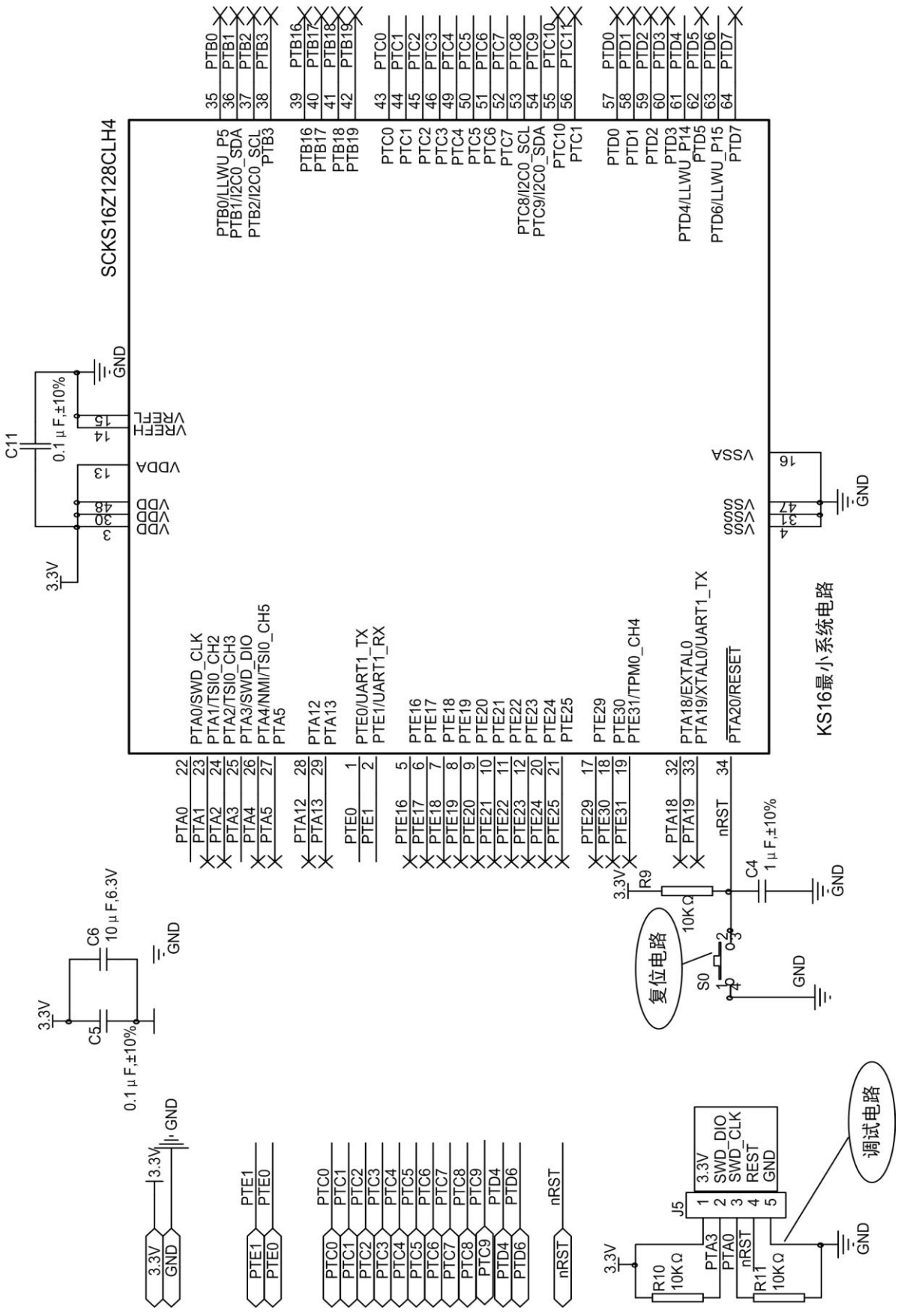

图 4.3　KS16 最小系统设计电路

4.2.3 矩阵键盘硬件电路设计

按键部分的电路设计采用矩阵键盘的设计，可以节省 IO 端口，在每个按键上都加上二极管，是为了识别多个按键按下，同时防止多个按键按下时烧坏 GPIO 通道。在程序时面，一个按键状态用 1 个位的状态来表示它，逻辑电平 0 代表没有按键按下，逻辑电平 1 代表有按键按下。原理图设计如下图 4.4 所示。

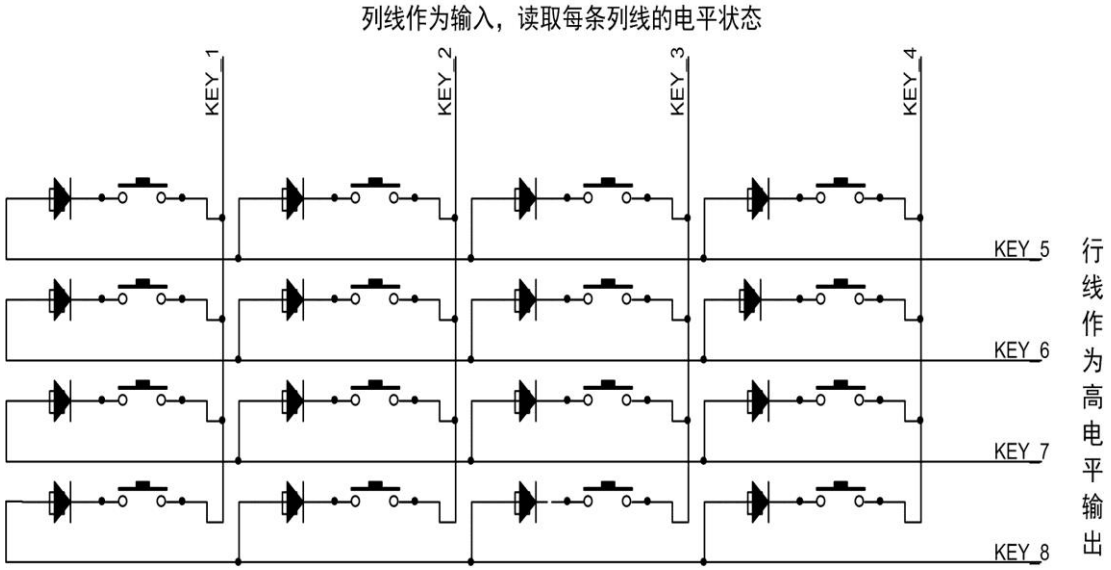

图 4.4　矩阵键盘硬件电路设计

4.2.4 传感器电路设计

飞鼠手持端用 KS16 作为主控，与 MPU6050 传感器之间是通过 I2C 总线通信，所以 KS16 与加速度与陀螺仪的接口电路如下图 4.5 所示。其中 PTC8_SCL 为 I2C 的同步时钟总线，PTC9_SDA 为 I2C 的数据总线。这两条总线，需要给上拉电阻才能正常工作，上拉电阻一般可以选择 4.7K 或 10K。

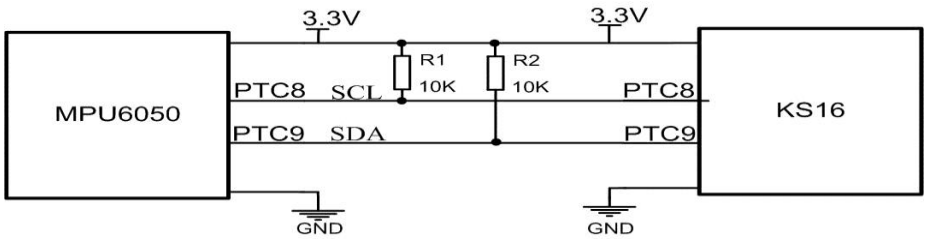

图 4.5　MPU6050 传感器接口电路设计

4.2.5 其它部分硬件电路设计

本次设计的硬件电路的设计，无线转 USB 接收端的硬件电路设计主要由队友负责，在这里不再进行详细的描述。

4.3 PCB 设计

在这次硬件系统的 PCB 布线了，主要注意有以下几个方面：

1、电源的走线尽可能粗一些，最好不要使用过长的电源走线，如给电源线绕圈。不同的电源线不要相互交叉重叠，以免造成电源供电质量不好；

2、尽量加宽地线和减小地线环路的面积，达到增大地线电流阈值和降低电路感应噪声的目的。同时，注意模拟地与数字地的分离；

3、接收端 USB 的布线原则应该要短而直，两条线的差分阻抗尽量一致，尽量符合其差分规则，这样可以避免 USB 传输不稳；

4、蓝牙天线的衬底不要铺铜，以免影响无线传输的质量；

5、I2C 总线的 PCB 布线尽量少打过孔，过多的过孔会产生寄生电感，影响数据传输。

4.4 PCB 设计展示

原理图与 PCB 布线完成后，这次硬件系统的 PCB 测试板如下图 4.6 所示。

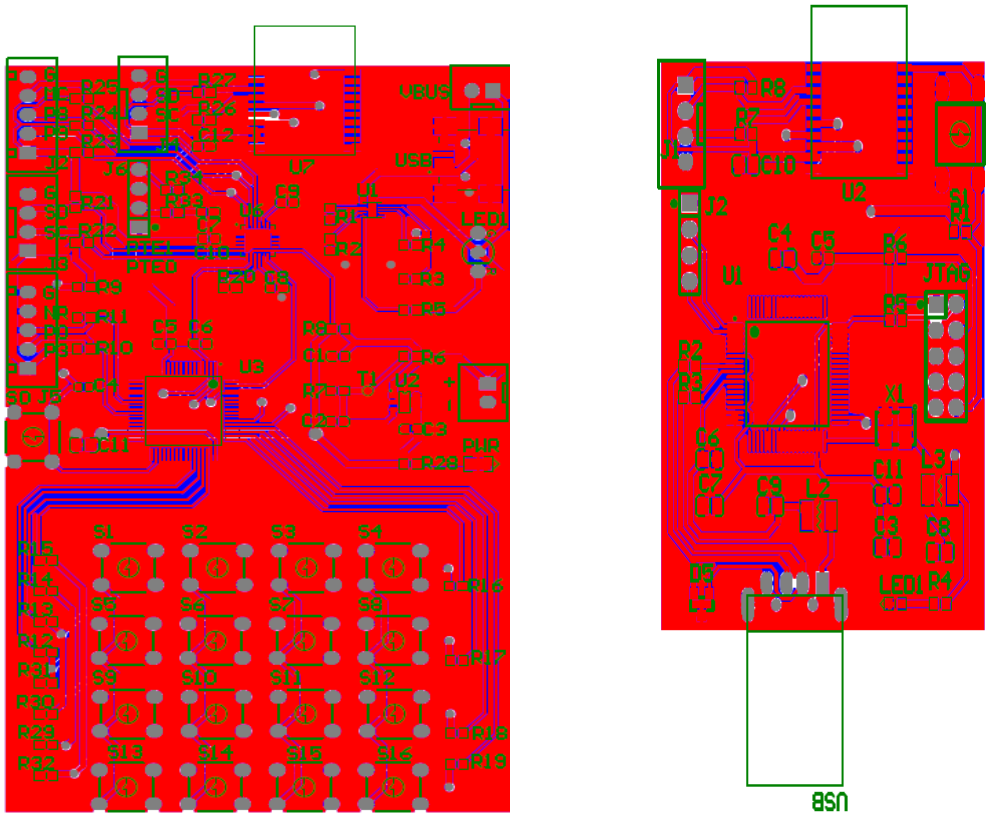

图 4.6　飞鼠 PCB 测试板

注：上图 4.6 左图是飞鼠手持端测试板，右图是无线转 USB 测试板。

4.5 本章小结

本文中电路设计的亮点在于，主要有以下几个方面：

1、可充电，一次充电可以使用半个月左右；

2、完整的系统电路设计，不是利用一个个硬件模块，用“搭积木”的方式来构建整个无线飞鼠硬件系统；

3、矩阵键盘支持识别多个按键同时按下，同时还具有保护主控 KS16 GPIO 的通道功能；

4、硬件的成本较低，各种芯片材料加起来的价格不超过 80 元，而市面上飞鼠的价格一般都在 100 元以上。

第五章　系统软件设计

5.1 软件系统框架设计

在硬件系统设计实现的基础上，无线飞鼠系统的软件系统也可以分为两部分，一部分为飞鼠手持端，另一部分为无线转 USB 接收端。飞鼠手持端的软件系统应该有手势姿态这一软件部分（包括读取传感器数据与对数据进行处理，利用处理后的数据进行姿态解算），功耗管理软件部分，手势姿态数据与按键状态数据编码软件部分，蓝牙透传（从机）软件实现部分等。无线转 USB 接收端的软件系统主要包括 USB HID 功能固件的实现部分，蓝牙透传（主机）软件实现部分，接收数据的校验与解码软件部分等。整个无线飞鼠系统各个软件部分的关系框图如下图 5.1 所示。

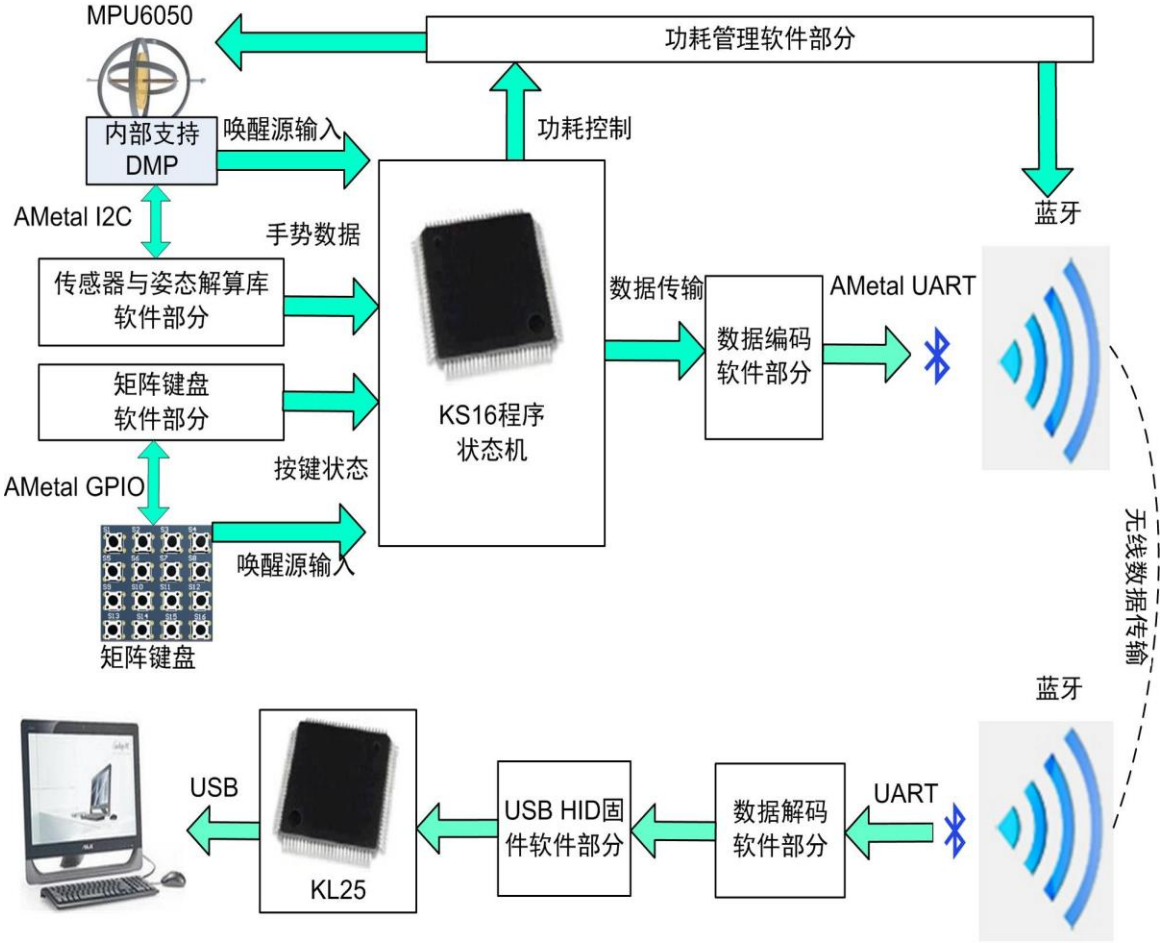

图 5.1　软件系统原理框图

5.2 软件实现的思路

上一小节中描述了各软件模块部分之间的关系，它们之间的工作方式主要如

下所描述的所示：

1、飞鼠手持端程序先运行时会进行一系列的初始化工作，然后 KS16 程序的状态机会通过 AMetal 的 I2C 标准接口不断读取姿态的数据，通过 GPIO 不断的读取按键的状态信息。状态机会根据这些信息做出反应，如果一段时间，获取到的数据没有明显变化，就让传感器的 DMP 关闭与蓝牙进入深度睡眠。进入睡眠后，可以通过传感器运动中断检测与按键唤醒程序状态机。此时程序状态机又开始重新运行；

2、无线转 USB 接收端的程序会进行初始化后，通过蓝牙的串口把蓝牙接收到的无线数据送给 KL25。主控 KL25 把接收到的数据进行校验与解码后，把数据封装成 HID 的数据格式（报告数组），然后通过 USB HID 固件里面的接口函数实现通过 USB 非零端点与电脑进行通信，从而可以控制电脑。

5.3 软件程序设计实现

5.3.1 飞鼠手持端软件的实现

1. KS16 状态机程序流程图

无线飞鼠（空中鼠标）手持端的系统软件是基于 ARM 的 Keil MDK5.15 开发编译环境进行开发的。整个软件设计包括传感器数据获取、利用 DMP 库进行手势姿态解算、按键状态信息扫描获取、手势姿态数据与按键信息数据编码，功耗管理四个软件子部分。这些软件子部分都是由 KS16 程序状态机控制与管理，从而保证光标移动和按键控制的实时性和准确性与各个功能的实现。整个飞鼠手持端的软件程序流程图如下一页的图 5.2 所示。

2. KS16 各个软件模块实现

(1) 传感器软件实现

传感器软件的实现流程主要包括两个方面：MPU6050 官方 DMP 驱动库的移植。移植成功后，进行 MPU6050 的初始化并利用 DMP 库的接口取得姿态信息。

1、传感器 DMP 的移植

本次设计中传感器为 InvenSense 公司的 MPU6050，这款传感器芯片最大的好处时，内部支持数字运动处理，可以很快的实现姿态解算，得到手势的姿态。InvenSense 公司提供 6 个文件，但其中与移植关系密切主要有 4 个文件：inv_mpu.c 与 inv_mpu.h，inv_mpu_dmp_motion_driver.c 与 inv_mpu_dmp_motion_driver.h 文件。其中在两个 C 文件中需重新移植实现的函数如下所述：需要实现 I2C 总线的读写函数，如程序清单 5-1 所示；一个延时函数与一个获取延时的函数（其实这个函数的函数体移植的时候为空），如程序清单 5-2 所示。

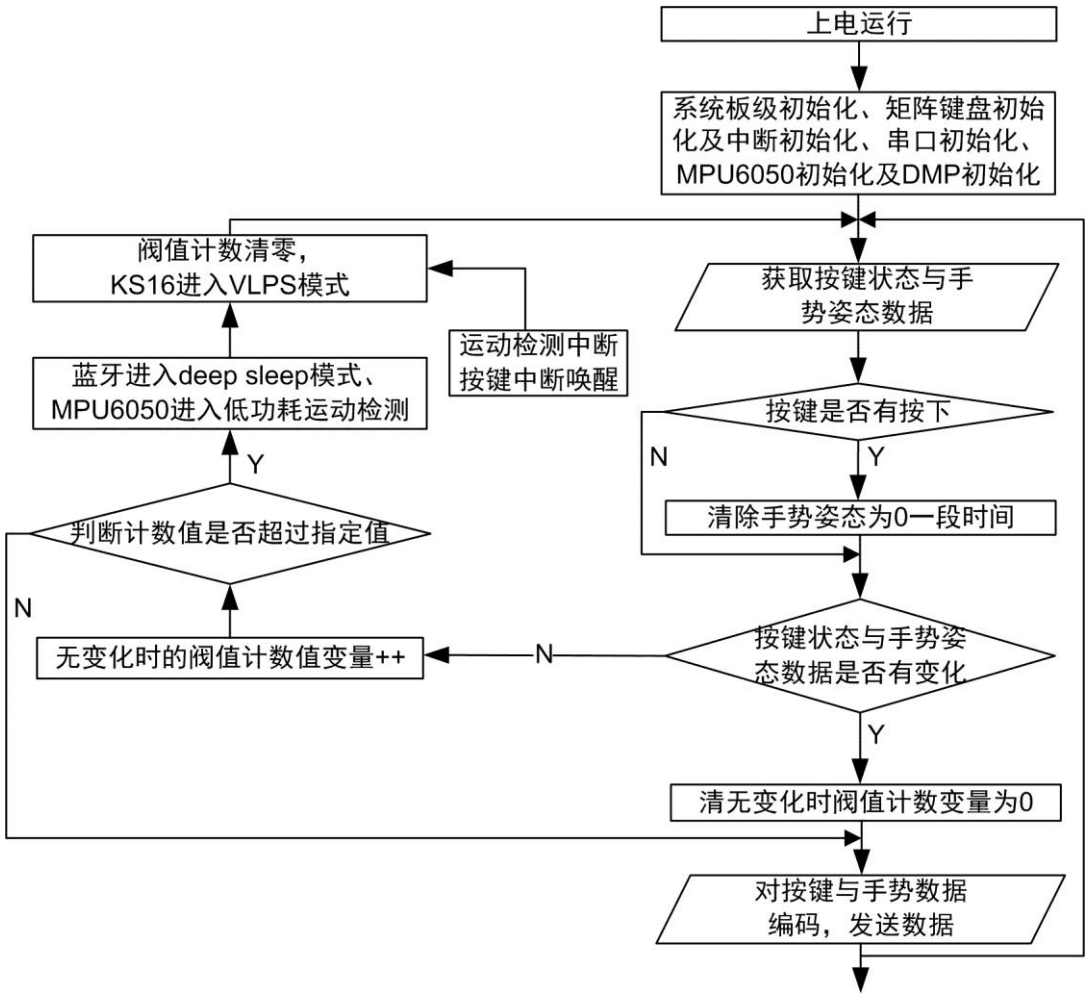

图 5.2　飞鼠手持端软件程序流程图

注：当按键按下时，手势姿态数据清 0 主要是用于当按键按下时，防止桌面光标出现抖动而导致不能正确的点击图标。

程序清单 5-1　MPU6050 DMP I2C 读写函数

```
/*   DMP 通过 I2C 总线连续读 MPU6050   */
int am_mpu_read_len (uint8_t addr,uint8_t reg,uint8_t len,uint8_t *p_buf)
{
    int status = 0;
    am_mpu6050_dev_t *p_dev = (am_mpu6050_dev_t *) mpu6050_handle;

    if (p_dev == NULL ||   p_dev->handle == NULL) {
        return -AM_ERROR;
    }
    (void) addr;
    if (len > 1) {
        status = am_i2c_read(&p_dev->i2c_device, reg, p_buf, len);
    } else {
        status = am_i2c_read(&p_dev->i2c_device, reg, p_buf, 2);
```

```
    }
    return status;
}

/*  DMP 通过 I2C 总线连续写 MPU6050  */
int am_mpu_write_len (uint8_t addr, uint8_t reg, uint8_t len, uint8_t *p_buf)
{
    int status = 0;
    am_mpu6050_dev_t *p_dev = (am_mpu6050_dev_t *)mpu6050_handle;
    if (p_dev == NULL ||  p_dev->handle == NULL) {
        return -AM_ERROR;
    }
    (void) addr;
    status = am_i2c_write(&p_dev->i2c_device, reg, p_buf, len);
    return status;
}
```

程序清单 5-2　MPU6050 DMP 延时与获取延时函数

```
/* DMP 获取延时函数  */
void get_ms(unsigned long *time)
{
}
/* DMP 延时函数  */
void delay_ms(unsigned long time)
{
    am_mdelay(time);
}
```

2、传感器初始化与获取姿态信息

MPU6050 的官方 DMP 驱动库移植完毕之后，在实现 MPU6050 初始化及 DMP 库初始化中取得操作句柄后，可以通过该句柄来调用 DMP 库的接口来实现手势姿态信息的获取。

(2) 矩阵键盘软件的实现

矩阵键盘采用行扫描法的原理，每次扫描时，参照 4.2.3 小节的图 4.4 矩阵键盘电路示意图，依次给四行 KEY_5、KEY_6、KEY_7、KEY_8 其中一行一个高电平，而其它三行为低电平，然后读取列线 KEY_1、KEY_2、KEY_3、KEY_4 这四列状态。如果其中一列有按键按下，这一列所代表的逻辑电平为 1，否则逻辑电平为 0。一个扫描周期下来，根据不同的行与列组合时便可确定的按键信息，这样就可以得到整个矩阵键盘的信息。其软件实现是先经过矩阵键盘的初始化后得到其句柄，再通过句柄调用其矩阵键盘接口函数来得按键的状态信息。

(3) 串口发送软件实现

串口的实现主要是先经过初始化后，再调用接口函数即可实现，串口软件的实现主要用于与蓝牙通过串口进行数据的交互。

(4) 功耗管理软件实现

功耗管理软件的实现主要是根据传感器输出数据的变化与按键状态信息，然后 KS16 的程序状态机根据这些信息的变化做出处理，从而让飞鼠手持端是否进入休眠或运行状态，其实现的流程如图 5.3 所示。

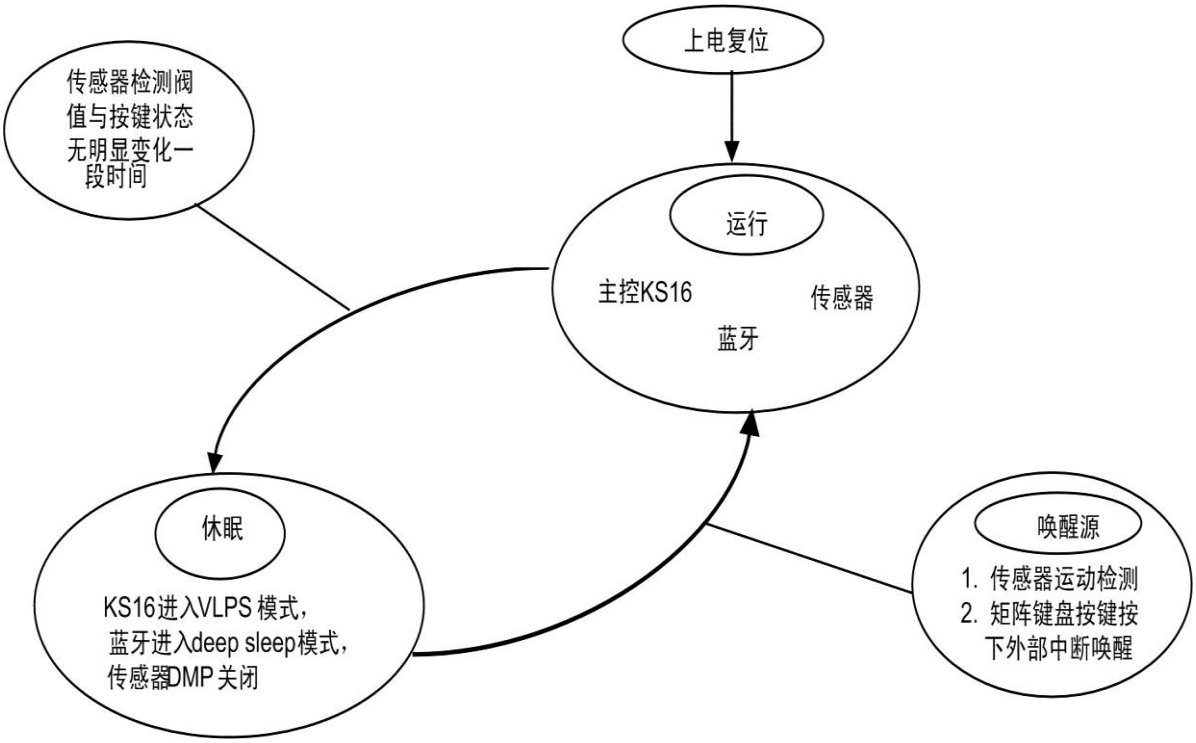

图 5.3　功耗管理软件实现流程

(5) 数据编码软件实现

数据编解码软件的实现主要是通过构造一帧用户自定义的数据，然后通过主控 UART 与蓝牙串口交互给蓝牙后，蓝牙利用协议栈，利用 RF 射频天线把数据无线传送出去。构造用户自定义数据的目的是让其无线转 USB 接收端在接收到这一帧数据的时候，可以方便对这一帧数据进行解码与校验，提高数据传输成功率，防止传输误码的出现。本次的设计中，数据帧的格式定义如下表 5-1 所示。

表 5-1　数据帧格式定义

帧头	帧头	用户数据				求和校验
0xa5	0xa5	X 轴坐标	Y 轴坐标	按键信息高字节	按键信息低字节	SUM

5. 3. 2 无线转 USB 接收端软件实现

1． KL25 状态机程序流程图

无线飞鼠的无线转 USB 接收端系统程序是基于 KDS3.0 开发环境设计的，在

程序中主要包括串口数据接收及数据解码软件实现，USB HID 软件固件实现。为了防止意外的情况出现，加上看门狗功能软件实现。所以在无线转 USB 接收端主要有三个软件子部分，整个系统的程序流程图如图 5.4 所示。

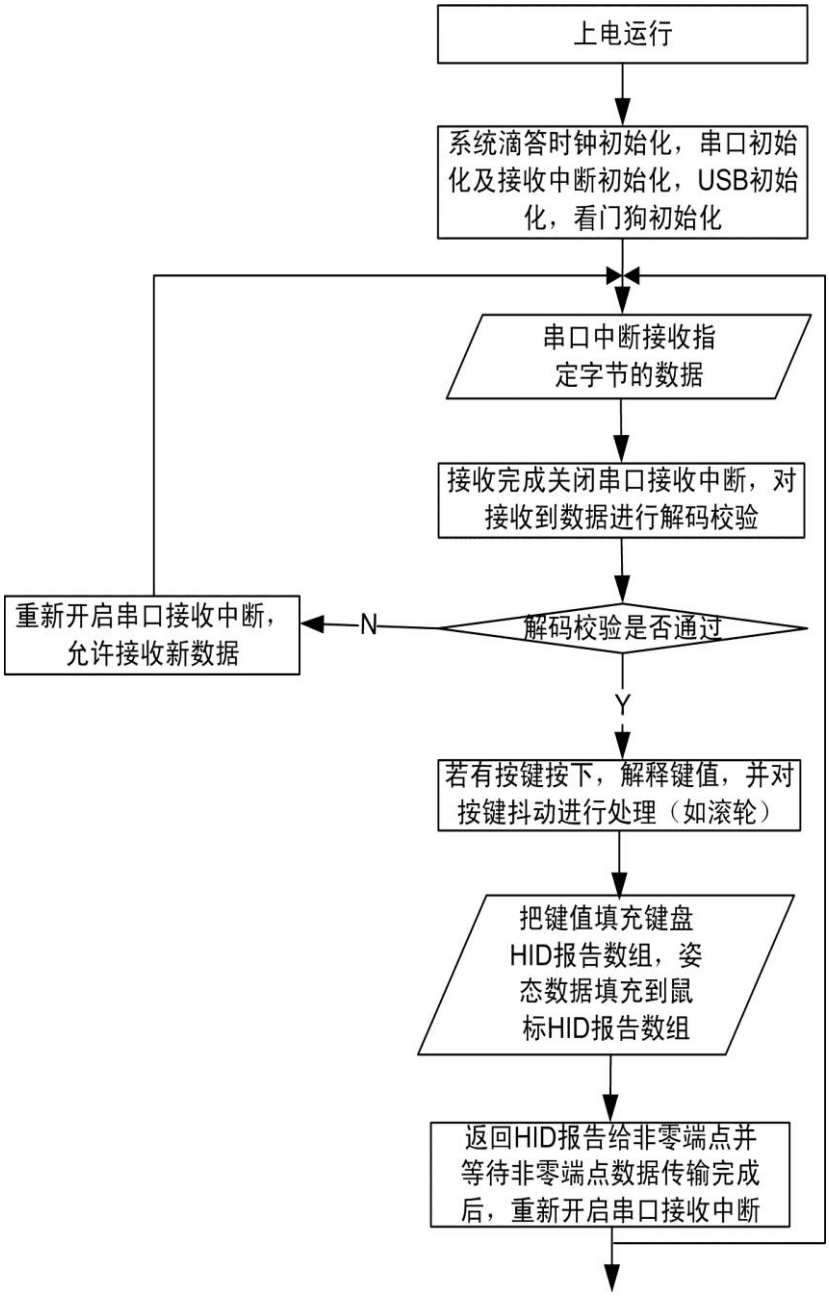

图 5.4　无线转 USB 接收端软件程序流程图

2． USB HID 枚举过程

根据 USB 协议，一个 USB HID 从机设备要与主机通信时，必须先经过枚举过程，让主机识别是哪一种设备，从而为其加载相应的驱动程序，这样才可以正常使用，一个完整的枚举过程一般如图 5.5 所示。

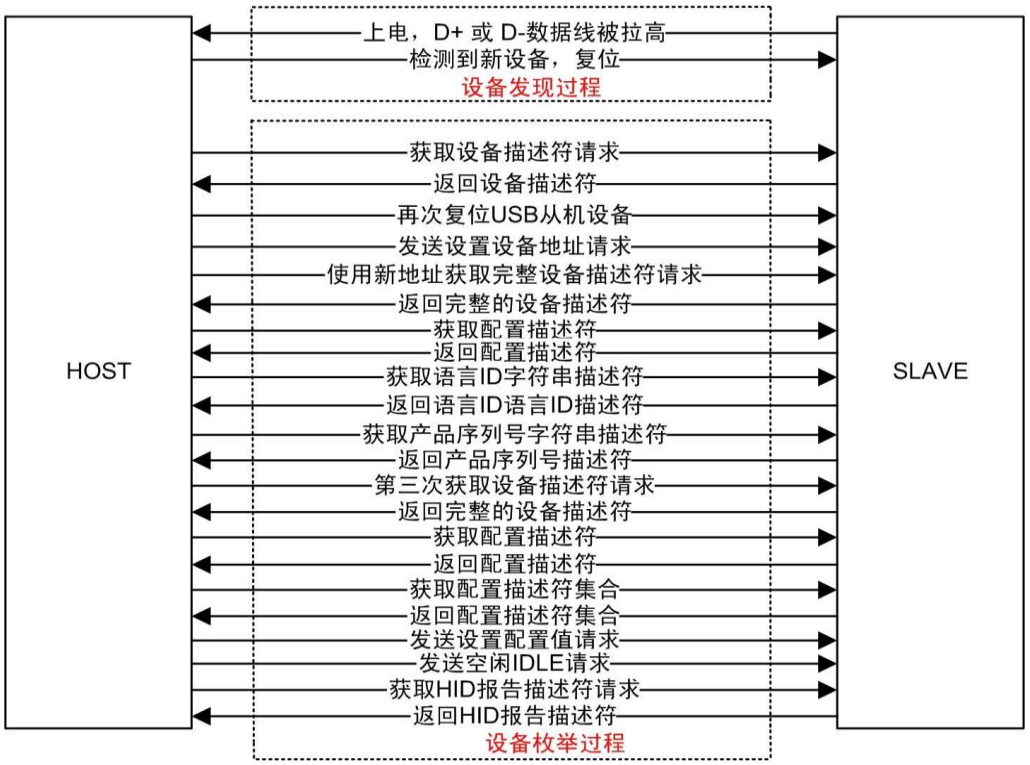

图 5.5　USB HID 的枚举过程

3．KL25 各个软件模块实现

(1) USB HID 人机交互功能的实现

KL25 是通过缓冲区描述符表 BDT 来高效地管理 USB 端点通信，其软件实现主要有以下几个关键函数，如程序清单 5-3 所示

程序清单 5-3　USB HID 人机交互主要功能软件实现

```
void usb_init(void)                /*  初始化函数          */
void usb_reset_handler(void)   /*  复位中断处理函数  */
void usb_handler(void)             /*  令牌中断处理函数  */
```

这三个函数的作用与软件实现流程如图 5.6 所示。

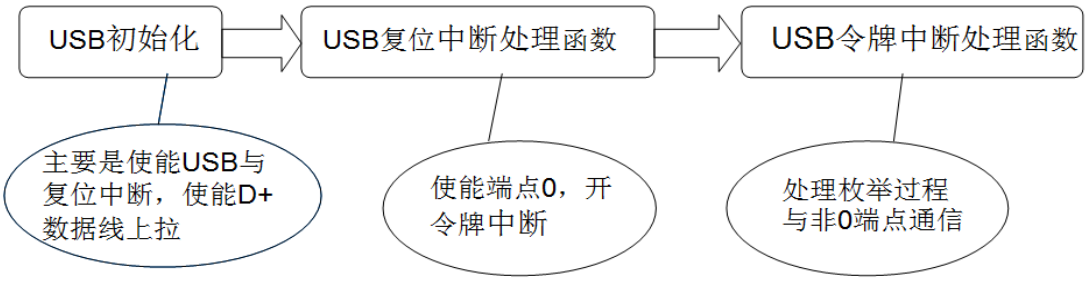

图 5.6　USB HID 人机交互功能软件流程图

5.3.3 蓝牙串口透传软件的实现

1. 蓝牙 BLE 技术原理与特点

蓝牙 4.0 BLE 工作在 2.4GHz ISM 频段，仅使用了 3 个广播道道，使用所有的蓝牙规范版本通用的自适应跳频技术，最大程度地减少与其它 2.4GHz ISM 频段无线电波的串扰。故而通信的质量可靠性高。

蓝牙 4.0 BLE 技术是蓝牙发展历史上一个革命性突破，它是三位一体的蓝牙技术。它将传统蓝牙、高速蓝牙和低功耗蓝牙技术融合在一起，这三个规格可以组合使用也可以单独使用，可以根据需要的不同在各个场合灵活使用，并且它最重要的特色就是低功耗技术。正是如此，蓝牙 4.0 BLE 技术在消费电子市场得到广泛应用与无限的应用空间。

2. 蓝牙协议栈介绍

Qn902x 的 BLE 协议栈分为 LL、L2CAP、SMP、ATT、GATT、GAP、Profiles 以及 APP 八个层，如下图 5.7 所示。

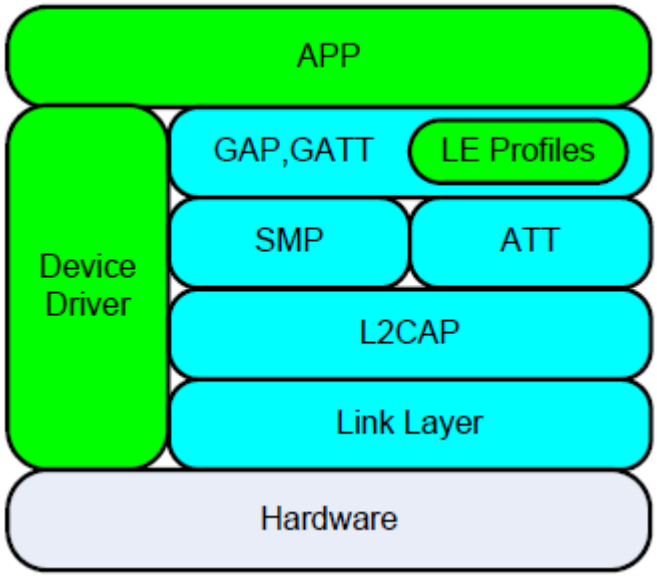

图 5.7　QN902X 蓝牙 BLE 协议栈架构

它的每一层的作用大概如下所述：

(1) 硬件 Hardware 层

这一层主要实现 1Mbps 自适应跳频的 GFSK 射频，工作于免许可证的 2.4GHz ISM 频段[18]。

(2) Link Layer 链路层

用于控制设备的射频状态，主机设备将会处于以下几种射频状态之一：初始化、等待、扫描、发起连接请求、连接；从机设备将会处于以下几种射频状态之

一：初始化、广播、回应扫描、接收连接请求、连接。

(3) Device Driver 硬件驱动层

主要是在协议栈里面提供 QN902X 的硬件底层接口。

(4) L2CAP 逻辑链路控制及自适应协议层

这一层主要为协议栈上层提供数据的封装服务，允许逻辑上的点对点通信。

(5) SMP 安全管理层

这一层主要定义了蓝牙设备的配对和密钥分配方式，加密无线数据在不同设备之间的传输。

(6) ATT 属性协议层

这一层允许一个蓝牙设备向另一个蓝牙设备展示特定的一块数据，这块数据称为“属性”。在 ATT 这一层，作为服务器的蓝牙设备会展示“属性”，客户端指的是与服务器配对连接的蓝牙设备。一般来说，多数情况下，主机设备是客户端，从机设备是服务器。

(7) (GAP,GATT Profile)配置文件层

这一层包括 GAP 角色/安全配置文件、GATT 配置文件，处于协议栈的顶层。GAP 层定义了使用 ATT 服务的框架，GATT 规定了配置文件的结构。在蓝牙 BLE 4.0 中，被蓝牙协议的 Profile 或服务用到的数据都称之为“特性”。

(8) 应用层

这一层主要是给用户编写应用相关的程序与直接使用配置文件层。

可见，QN902X 蓝牙 BLE 4.0 的协议栈每一个层都是一个单独的状态机，有独立的任务描述器和状态形式。为了支持层与层之间的数据和消息交换，Qn902x 的蓝牙协议栈实现了一个非常基本的消息驱动型内核，是一个小型高效的实时操作系统（RTOS）。操作系统的主要任务就是实现内存分配、层与层之间的消息通信和事件处理以及定时器功能。

任务发出的消息会保存在消息队列中，由操作系统负责管理。操作系统会根据消息的 ID 号查找到相应的消息处理 hander 函数并执行。如果消息队列被清空，内核一般会进入空闲（idle）状态。

用户在使用的时候可利用“事件”、“消息”、“任务”等手段来完成应用程序之间的协调运行。

3. 蓝牙通信过程

一次蓝牙通信一般需要以下步骤，包括发现设备、连接、配对和绑定等 4 个主要部分。如下图 5.8 所示。

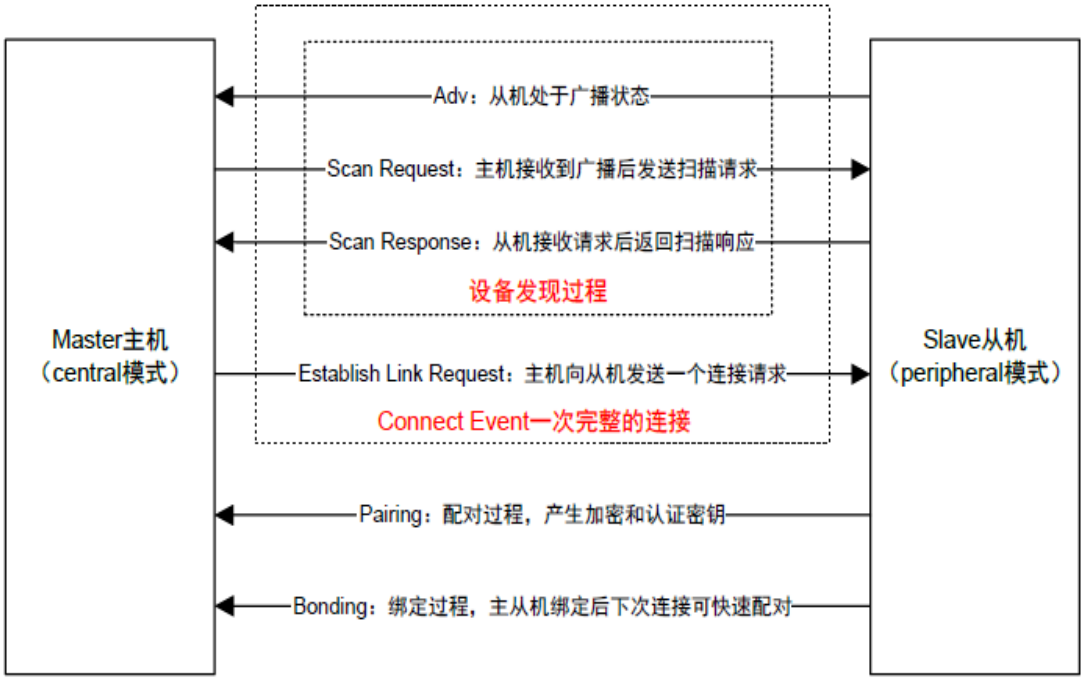

图 5.8　蓝牙通信连接过程分析

4． 蓝牙上电自连接

根据蓝牙通信过程可知，结合 QN902X 蓝牙协议栈的，从机上电后会不断的对外广播，主机接收到广播会发起扫描请求，扫描到设备会产生一个消息事件 GAP_DEV_INQ_RESULT_EVT，由此得到蓝牙上电实现快速链接的思路：

每次扫描到一个设备后，在产生事件相对应的事件消息处理函数中判断当前扫描设备的 MAC 地址是否是指定设备的地址，如果是，则发起链接请求，同时取消扫描，连接完成后，可以选择进行配对与绑定。如下图 5.9 是蓝牙主从机上电自连接的过程显示。

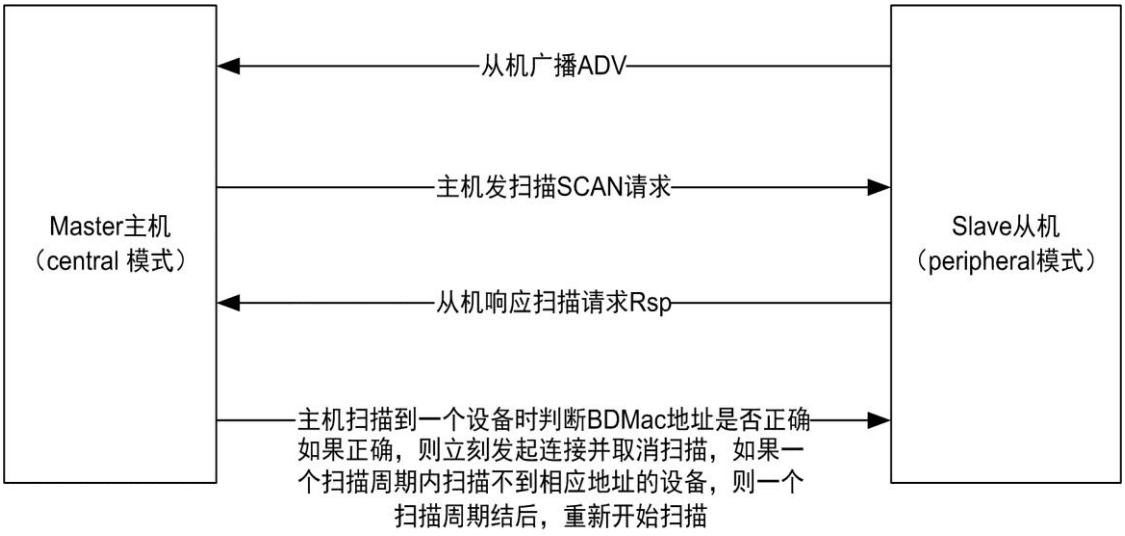

图 5.9　蓝牙主从机上电自连接过程分析

具体代码实现如程序清单 5-4 所示，其实本质上这段函数代码是一个事件状态机函数。

程序清单 5-4　蓝牙上电自连接实现函数代码

```
void app_task_msg_hdl(ke_msg_id_t const msgid, void const *param)
{
    uint32_t i = 0;
     switch (msgid) {
     case QPPC_DATA_IND:
          for (i = 0; i < ((struct qppc_data_ind *)param)->length; i++){
               UartPutc(((struct qppc_data_ind *)param)->data[i]);
          }
          break;
     case GAP_DISCON_CMP_EVT:
          /* 断开连接事件     */
          APPSetBLEDisConnect();                                    /* 清除连接标志位，发送消息给
                                                                       主控表明连接已经断开          */
          uart_rx_int_enable(APP_UART, MASK_DISABLE);       /* 断开连接后串口接收中断禁能 */
          led_set(4,    LED_OFF);                                   /* 熄灭连接指示灯                */
         app_gap_dev_inq_req(GAP_GEN_INQ_TYPE, QN_ADDR_TYPE); /*断开连接的话重新开始扫描*/
         break;
     case GAP_DEV_INQ_RESULT_EVT:
           /* 发现设备事件 */
          if (app_env.inq_addr[app_env.inq_idx].addr[0] == 0x0D) {
              /* 如果找到的设备地址匹配，则取消扫描，并发起链接             */
              app_gap_dev_inq_cancel_req();
              app_env.select_idx = app_env.inq_idx;                 /* 记住当前连接的设备序列号 */
              app_gap_le_create_conn_req(&app_env.inq_addr[app_env.select_idx],
                                         app_env.addr_type[app_env.select_idx],
                                         QN_ADDR_TYPE,
                                         GAP_INIT_CONN_MIN_INTV,
                                         GAP_INIT_CONN_MAX_INTV,
                                         GAP_CONN_SUPERV_TIMEOUT);
           }
         break;
     case GAP_DEV_INQ_REQ_CMP_EVT:
          /* 扫描周期完成事件 */
          if (app_env.inq_idx == 0){
               /* 如果没有找到设备，那么重新开始扫描   */
               app_gap_dev_inq_req(GAP_GEN_INQ_TYPE, QN_ADDR_TYPE);
          } else {
               for (i = 0; i < app_env.inq_idx; i++) {
                    if (app_env.inq_addr[i].addr[0] == 0x0D ) {
                         app_env.select_idx = i;                    /* 记住当前连接的设备序列号   */
```

```
                        app_gap_le_create_conn_req(&app_env.inq_addr[i], app_env.addr_type[i],
                                                   QN_ADDR_TYPE,
                                                   GAP_INIT_CONN_MIN_INTV,
                                                   GAP_INIT_CONN_MAX_INTV,
                                                   GAP_CONN_SUPERV_TIMEOUT);
                        break;
                    }
                }
                if ( i >= app_env.inq_idx) {
                    /* 找到设备的设备，地址不对，那么重新开始扫描  */
                    app_gap_dev_inq_req(GAP_GEN_INQ_TYPE, QN_ADDR_TYPE);
                }
            }
        break;
    case GAP_LE_CREATE_CONN_REQ_CMP_EVT:
        /* 连接完成事件处理 */
        led_set(4, LED_ON);                         /* 点亮 LED，表明已经连上 */
        break;
    case GAP_CHANGE_PARAM_REQ_CMP_EVT:
        /* 更新连接参数请求完成 */
        {
          uint16_t conhdl = app_get_conhdl_by_idx(app_env.select_idx);
          if (0xFFFF != conhdl && app_env.select_idx < BLE_CONNECTION_MAX
              && false == app_get_qpp_client_service_status(app_env.select_idx))
          {
              app_qppc_enable_req(NULL, conhdl);                    /* 使能请求 notify  */
          }
        }
        break;
    default:
        break;
}
```

5. 蓝牙串口透传软件

蓝牙成功连接后，根据从机 QPPS 工程里面的 profile 文件定义了的三个 notify 特征值。在蓝牙协议栈里面，主从机之间应用数据的传输都是通过“特性”（特征值）来实现的。当在主机使能了从机的 notify 特征值通知后，如果在从机中改变了这个 notify 的特征值，主机端就会产生通知类事件，表明从机改变了这个特征值，主机可以从这个特征值读取从机传过来的数据，再通过串口把数据打印出来，软件流程如下图 5.10 所示。

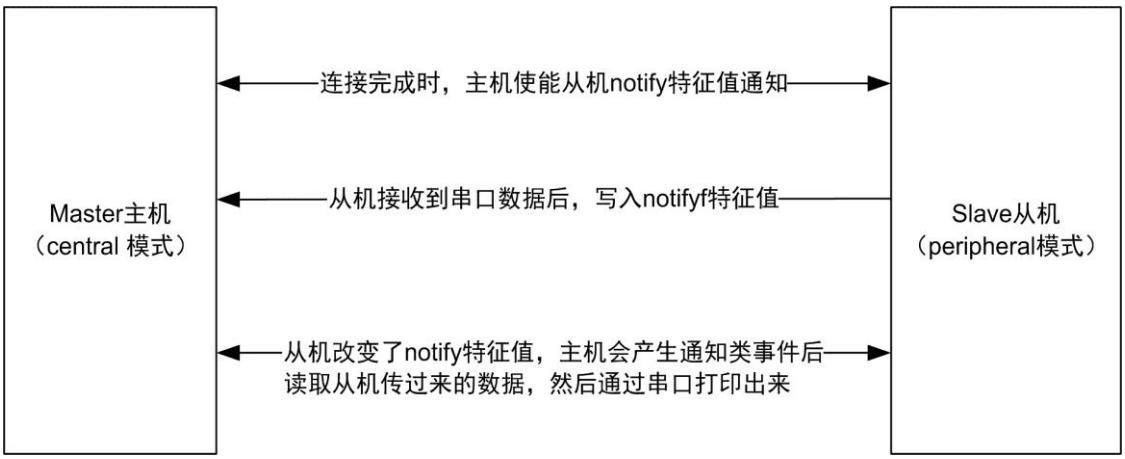

图 5.10　蓝牙透传软件实现分析

5.3.4 鼠标光标滑动位移软件

为了使鼠标的每一次移动变得更平滑，没有产生跳跃感，数据处理时，每次得到的姿态数据先扩大 100 倍再与上一次的姿态数据作差，这样可以防止因精度丢失而造成的鼠标光标跳跃。通过对差值进行比例系数的放大与缩小，可以调节光标移动的灵敏度。软件实现如程序清单 5-5 所示。

程序清单 5-5　鼠标光标映射编码软件实现

```
for (i = 0; i < 3; i++) {
    g_pst[i] = (g_bufferr[i] * 100    - g_last_bufferr[i] * 100) / 4 ; /*  取得当前光标的位移      */
    g_last_bufferr[i] = g_bufferr[i];                                  /*  记住上一次的姿态数据  */
}
```

5.4 本章小结

本章详细的讨论无线飞鼠的手挂端与无线转 USB 接收端，蓝牙无线数据传输的软件实现。在无线飞鼠的手持端，实现了姿态数据获取，按键状态获取，数据编码，串口数据发送与蓝牙透传从机软件的实现，在无线转 USB 接收端了实现了 USB HID 人机交互功能，数据解码，主控 KL25 串口数据接收与蓝牙透传主机软件的实现。经过验证，本文中所编写的程序满足无线飞鼠功能的实现。

总的来说，这次无线飞鼠的软件设计从结构与开发周期时间上看，具有以下优势：

1、KS16 采用面向对象编程的思想，利用句柄操作，上公司的 Aworks 系统有优势；

2、KL25 采用模块化编程，藕合性小；

3、蓝牙使用 BLE 协议栈，只需关注应用的开发，可以加快开发的进度。

第六章　系统调试与测试

6.1 系统的调试

6.1.1 软件调试

整个软件系统调试的难点在于：

1、USB HID 人机交互功能的实现，调试 USB 时，可以采用 BUS Hound 一个总线检测软件来枚举的过程；

2、蓝牙串口透传的实现，需要对蓝牙协议栈有一个运行机制的理解，懂得运用协议栈，可以利用串口打印蓝牙状态信息与抓包工具调试；

3、对数据一些处理与对传输延时造成的滞后性处理，光标平滑性及按键按下时光标抖动的处理；

4、功耗管理软件的实现，怎么让系统进入休眠状态，唤醒的时候要迅速，这是软件系统实现中的难点。

6.1.2 硬件调试

1、用烙铁焊接完无线飞鼠 PCB 板后，需要检查 PCB 电路板的焊接是否存在短路的现象，特别留意电源与地之间是否短路现象。同时，细心检查虚焊、漏焊和错焊（如电阻用错阻值、二极管极性焊反、芯片位置焊错）等现象；

2、检查完成后，使用清洗液和专业工具对电路板残留杂质如焊锡渣、松香等助焊剂及多余针脚等进行清理。

3、清理工作结束完成后，在确保焊接无误的情况下，尝试电路板上电，对电路板进行上电检测。根据设计的电路原理图与 PCB 走线分布，分别对各硬件部分进行电路的供电电压进行检查，检测各个模块留出来的测试点供电是否正常，最后检查整个 PCB 板的硬件系统是否正常；

4、如果一切正常，则可以通过软件来测试硬件的功能，先用一些单元测试用例测试各个硬件模块是否能正常工作，如传感器，矩阵键盘，蓝牙，USB HID 功能，当一切功能正常后，再用整个系统的软件来测试。

6.2 功能测试

1． 手势姿态数据分析

由于加速度与陀螺仪传感器内部支持数据运动处理，只要移植好 DMP 库，就可以调用库的接口函数来得到手势姿态数据了，如下图 6.1 与图 6.2 分别为其

静止与运动时的姿态数据波形。

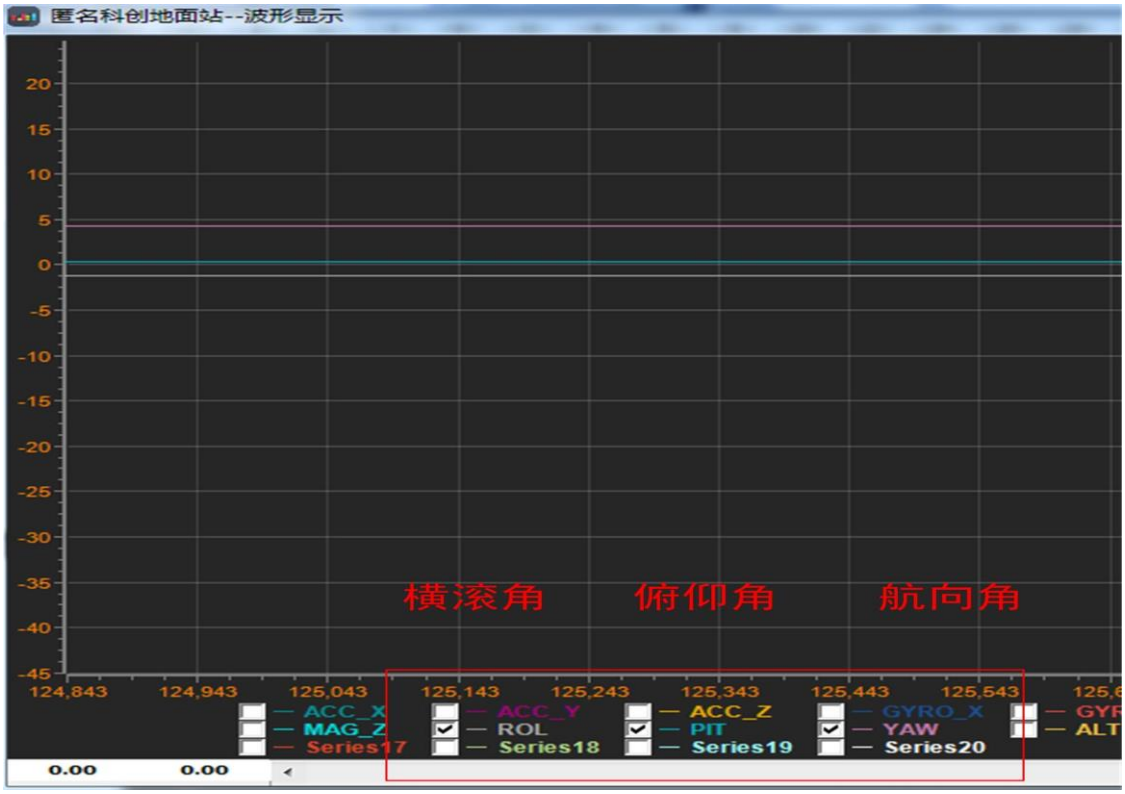

图 6.1　静止时姿态数据

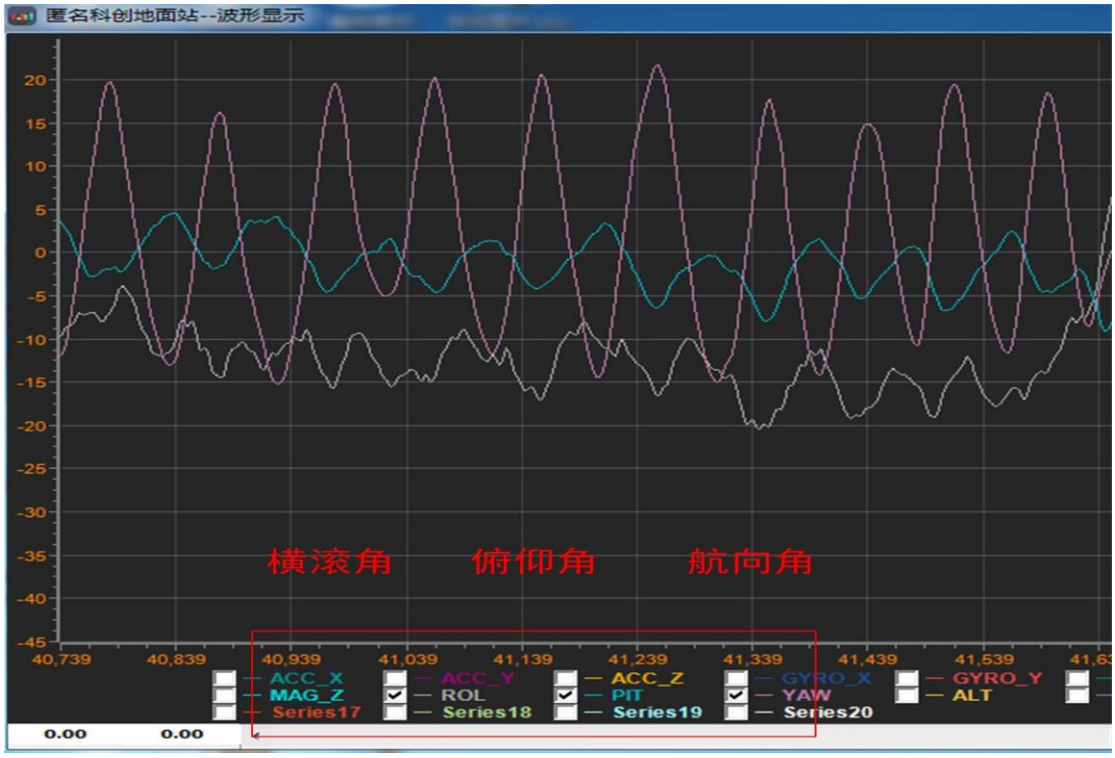

图 6.2　运动时的姿态数据

注：观察手势姿态的上位机为匿名飞控上位机软件。

根据上述图片所示，可见通过 DMP 库读出来的手势姿态数据静态时，漂移值很小，而运动时，其手势姿态数据的平滑性又非常高，足以满足这次飞鼠的控制要求。

2. HID 功能（键鼠一体化）

烧好 KL25 程序后，将无线转 USB 接收端插到电脑 USB 口后，在电脑为其安装驱动的效果如图 6.3 所示，正在安装驱动的设备名称正是字符串描述符定义的。

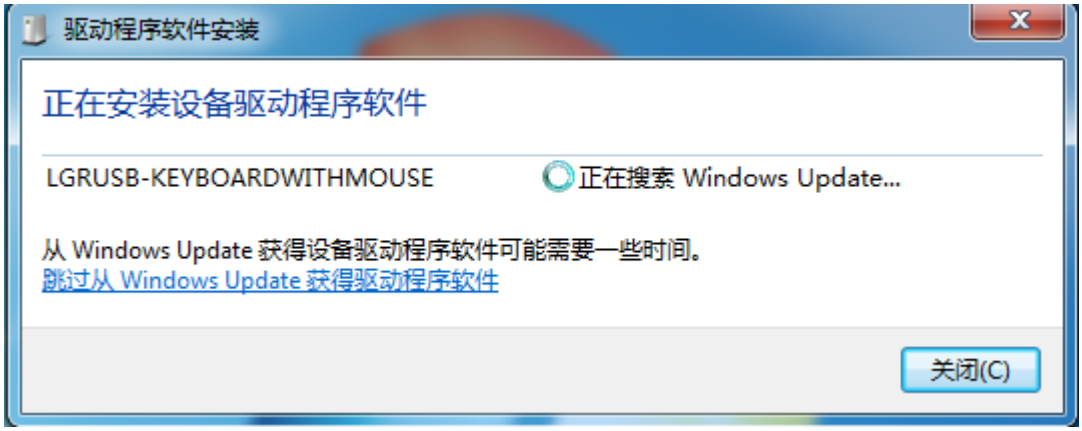

图 6.3　无线转 USB 接收端驱动安装过程

待电脑成功安装驱动后，电脑端会多出一个键盘设备与鼠标设备，如下图 6.4 中椭圆圈圈中内容所示。

图 6.4　键鼠 HID 设备实现

3. 实物展示

飞鼠手持端的 PCB 测试板，无线转 USB 接收端 PCB 测试板，分别如图 6.5、图 6.6 所示。

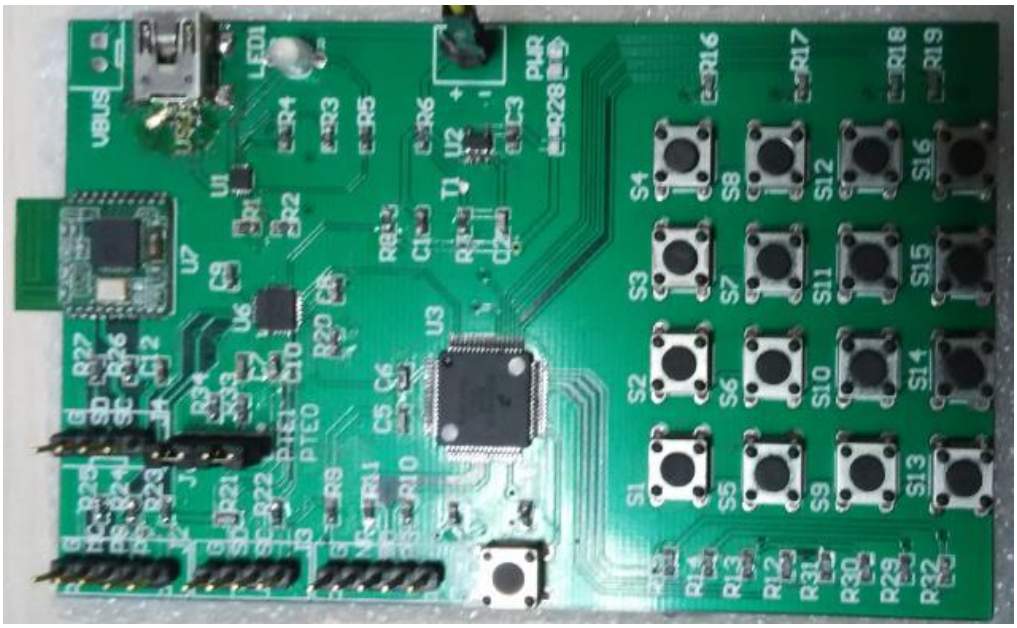

图 6.5　飞鼠手持端 PCB 测试板

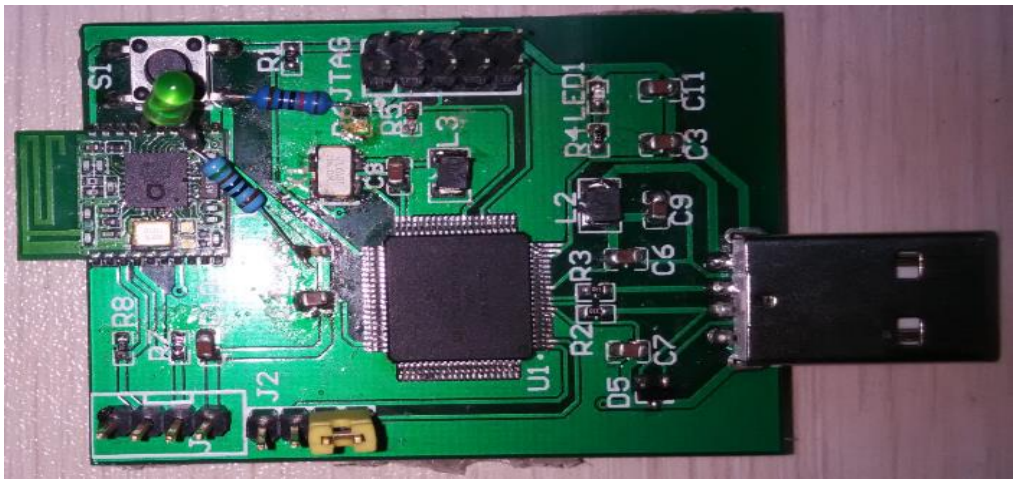

图 6.6　无线转 USB 接收端 PCB 测试板

4． 功耗对比

飞鼠手持端在正常运行时的功耗电流约为 11 毫安，在休眠时电流约为 0.9 毫安，加上可充电的锂电池充电电路设计，锂电池一次充电可以持久使用，续航能力十分强大。如图 6.7 是它们的功耗对比。

图 6.7　功耗对比示意图

注：在图 6.7 中左图是系统运行时的电流消耗，右图是休眠时的电流消耗.

6.3 性能对比测试

如果想全面了解飞鼠性能的好坏，实物演示是最能体现出飞鼠性能的途径，其次视频演试也能比较客观的反映无线飞鼠的真实性能。但是由于论文电子稿的局限性，因此本文中只好采用简单的测试截图来对飞鼠的性能进行测试。

(1) 测试无线飞鼠的基本功能

通过移动计算机资源管理器文件夹对飞鼠矩阵键盘中指定的按键模拟普通鼠标的左右键功能和通过手势控制电脑光标移动的功能进行测试。如下图 6.8 所示，可以得本文设计的无线飞鼠可以实现对电脑鼠标光标移动控制和鼠标左右按键操作的基本功能。

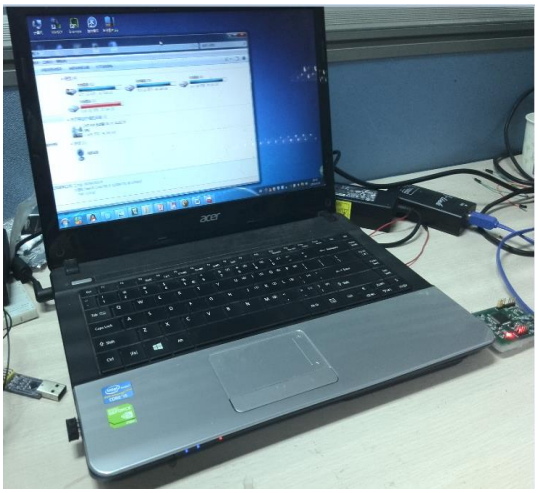

图 6.8　左右移对比测试

6.4 可靠性测试

本次设计的基于 KS16 的无线飞鼠，经过测试，连续长时间都可以正常工作，在广州致远电子公司答辩时小组队友们用完成的无线飞鼠作品来操控讲解小组每一个的 PPT 就是一个很好的印证。在休眠时，经过很长时间，唤醒后还可以正常使用，稳定性比较高。

6.5 本章小结

通过对无线飞鼠的软硬件系统的调试与进行的一系列测试，包括功能、性能测试等。可以得出，本次基于 KS16 的无线飞鼠所确定的设计方案是可行的，达到了设定的目标与既定功能及性能的实现。

第七章　总结与展望

7.1 总结

在本次基于 KS16 的无线飞鼠设计，体积相对较小，节能，飞鼠手持端部分采用锂电池供电，可进行多次充电使用，从而减少电池的购买，经济环保。无线转 USB 接收端直接通过计算机的 USB 端口供电，一个 USB 接口就可以同时用于数据传输和给电路板供电，减少了计算机 USB 端口的占用。在此次毕设过程当中，主要完成了以下几项工作：

1、完成了查阅专业学术期刊、撰写专业技术论文等的工作，并利用现有的参考资料与知识基础来开展研究工作；

2、利用飞思卡尔 KDS、ARM Keil MDK5.15 等开发工具完成了软件系统的设计；

3、利用 Altium Designer PCB 设计工具，完成了硬件原理图的设计及 PCB 的设计，同时焊接了 PCB 测试板；

4、了解了 USB 协议规范，学会了一些 USB 驱动固件的开发，同时理解了蓝牙 BLE 协议栈的工作，学会了在蓝牙协议栈上面开发一些应用程序。

总的来说，本文设计的无线飞鼠可以灵活的在三维立体空间中实现普通鼠标所具备的功能，同时还扩展一些键盘的功能，比如说一键播放 PPT，一键影音播放等，还增加组合键灵敏度调节，重新校准传感器的功能，可以摆脱桌面进行随心所欲的控制，对实际应用具有一定的意义。同时对于 MPU6050 和 KS16 来说，这两款芯片结合使用还有更大的使用空间，远远没有榨干 KS16 主控芯片的性能。所以还可以用它们实现更多功能与改进用户体验。

7.2 工作展望

由于时间关系，此次设计的飞鼠系统在性能和操控体验还存在一些可以改进的地方，后面的改进工作可以从以下几个方面进行：

1、硬件设计时合理选用适当的材料，上拉电阻应适当加大，二极管选用静态电流小的型号，从硬件上设计降低功耗；

2、在下一次重新做 PCB 板的时候，在 Altium Designer 中可以采虑采用 PCB 多层板的布线技术，使其 PCB 的走线布局更加合理化，使设计出来的飞鼠 PCB 板体积更小，特别是使无线转 USB 接收端的体积更小； .

3、设计中应该更多的考虑到在实际操控中的方便性和用户体验手感等问题，改进飞鼠手持端的外形，让其更符合人体输入学。

4、可以根据需要对无线飞鼠手持端的无线通信协议进行扩展，加入姿态角及其它控制信息，从而使得无线鼠标手持端可以控制更多遵循该协议的接收端，如飞行器接收端、智能小车接收端等等，实现用无线飞鼠去控制飞行器的飞行，智能小车的运行等功能；

5、系统从唤醒状态时，蓝牙重新连接时可以更快捷与稳定；

6、可以把红外遥控与学习功能加入飞鼠当中，实现与遥控一体化。

附　录

包括放在正文内过分冗长的公式、以备他人阅读方便所需的辅助性数学工具、重复性的数据图表、论文使用的符号意义、单位缩写、程序全文及有关说明等。

参考文献

[1] 周立功单片机．蓝牙 QN902X 软件开发指南[J]．广州：2015.9：1~56

[2] 恩智浦(NXP)．QN902X 蓝牙用户手册[J]．广州：2015.5：20~40

[3] 飞思卡尔．USBHID 例程指南[J]．苏州：2014：1~20

[4] 飞思卡尔．飞思卡尔单片机快速上手指南[J]．苏州：2014：1~44

[5] 飞思卡尔．飞思卡尔 KS16 用户手册[J]．2012：16~700

[6] InvenSense 公司．MPU-6000 and MPU-6050 Register Map and Descriptions[J]．2013：1~45

[7] InvenSense 公司．Embedded Motion Driver(DMP)第 5 版教程[J]．2013：1~9

[8] 秦永远．惯性导航[M]．北京：中国科学出版社．2006.5：288~305

[9] 秦永远．卡尔曼滤波与组合导航原理[J]．北京：西北工业大学出版社．2007：30~40

[10] 飞思卡尔．经典互补滤波器[J]．苏州：2007：1~3

[11] 无穷开源飞控．大话多旋翼飞行器--欧拉角与四元数[J]．2012：1~7

[12] Sebastian O.H Madgwick．An efficient orientation filter for inertial and inertial/magneticsensor arrays(融合算法)[D]．2010

[13] 周获．基于 MEMS 技术的无线空中鼠标的研究[D]．华侨大学：2013.7：2~15

[14] 江朝强． 基于 MEMS 指环式低功耗无线三维鼠标的设计[D]． 武汉大学：2013.5：2~4

[15] 姜晓波，钱莉．基于微加速度计的 AIR—MOUSE 的研究[D]．上海交大：2008.8：2~4

[16] 腾飞．基于 STM32F103 的空中鼠标的设计与论述[J]．湖南邵阳： 2015：2~5

[17] 王宜怀．嵌入式技术基础与实践[M]．苏州：清华大学出版社．2013：2~300

[18] 欧阳骏．物联网开发与技术实践蓝牙 4.0[M]．北京：化学工业出版社．2013：2~146

[19] 梁建宏．基于 ARM 与低成本 MEMS 器件的 AHRS 设计[J]．北航 2012.5 单片机与嵌入式应用第 5 期：2~4

[20] 刘荣．圈圈教你学 USB[M]．广州：北京航空航天大学出版社．2013.4：1~176

[21] 周立功单片机．USB 硬件电路设计原则[J]．广州：2015.5：1~10

[22] Altium 有限公司．Altium Designer 使用手册[J]．广州：2009.5：20~53

[23] 四元数定义与姿态解算的关系 http://wenku.baidu.com/link?url=e2INeby4vzWrtxZnKIFBdYl-5SlyXFuMHd-1jkzsvSDapJefDdMttHcYIgW0Q1Cd066r6y2zpQOYAhJz8Xu6hCo6mRufIvODBOk8reUmMH_

[24] 四元数姿态解算完全解析 http://wenku.baidu.com/link?url=UhIiIEnDpkRwvYIBz4-41S7jg48l8XNI5qto82QkKOmpFjh9wTtAlL8H8IkbHUXh-JA3NTIa3BmzMDO6npiK7DbjAg8zq6JZ4NLbGk2_Uoi

外文资料

KL25 Universal Serial Bus OTG Controller（USB OTG）

1. USB

The USB is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

USB software provides a uniform view of the system for all application software, hiding implementation details making application software more portable. It manages the dynamic attach and detach of peripherals.

There is only one host in any USB system. The USB interface to the host computer system is referred to as the Host Controller. There may be multiple USB devices in any system such as joysticks, speakers, printers,etc. USB devices present a standard USB interface in terms of comprehension, response,and standard capability.

The host initiates transactions to specific peripherals, whereas the device responds to control transactions. The device sends and receives data to and from the host using a standard USB data format. USB 2.0 full-speed /low-speed peripherals operate at 12Mbit/s or 1.5 Mbit/s.For additional information, see the USB 2.0 specification.

2. USB On-The-Go

USB is a popular standard for connecting peripherals and portable consumer electronic devices such as digital cameras and hand-held computers to host PCs. The On-The-Go(OTG) Supplement to the USB Specification extends USB to peer-to-peer application. Using USB OTG technology consumer electronics, peripherals, and portable devices can connect to each other to exchange data. For example, a digital camera can connect directly to a printer, or a keyboard can connect to a Personal Digital Assistant to exchange data.

With the USB On-The-Go product, you can develop a fully USB-compliant peripheral device that can also assume the role of a USB host. Software determines the role of the device based on hardware signals, and then initializes the device in the appropriate mode of operation (host or peripheral) based on how it is connected. After connecting the devices can negotiate using the OTG protocols to assume the role of host or peripheral based on the task to be accomplished. For additional information, see the On-The-Go Supplement to the USB 2.0 Specification.

3. KL25 USB-FS

- USB 1.1 and 2.0 compliant full-speed device controlle

- 16 bidirectional end points
- DMA or FIFO data stream interfaces
- Low-power consumption
- On-The-Go protocol logic

To efficiently manage USB endpoint communications the KL25 USB-FS implements a Buffer Descriptor Table (BDT) in system memory. The BDT resides on a 512-byte boundary insystem memory and is pointed to by the BDT Page Registers. Every endpoint direction requires two 8-byte Buffer Descriptor (BD) entries. Therefore, a system with 16 fully bidirectional endpoints would require 512 bytes of system memory to implement the BDT. The two BD entries allows for an EVEN BD and ODD BD entry for each endpoint direction. This allows the microprocessor to process one BD while the USB-FS is processing the other BD. Double buffering BDs in this way allows the USB-FS to transfer data easily at the maximum throughput provided by USB.

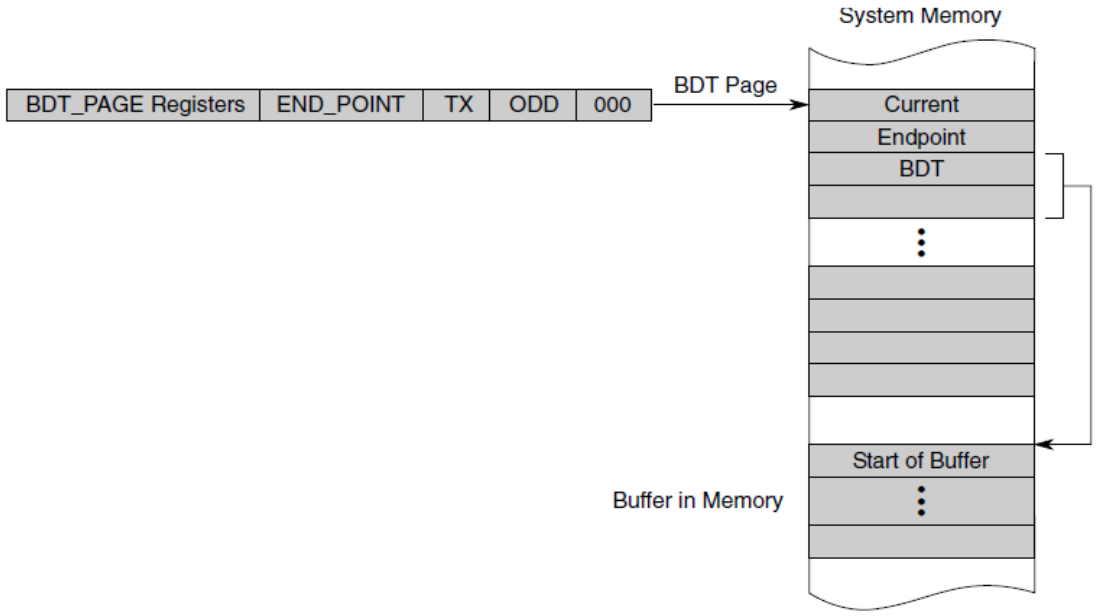

Figure 35-3. Buffer descriptor table

The software API intelligently manages buffers for the USB-FS by updating the BDT when needed. This allows the USB-FS to efficiently manage data transmission and reception, while the microprocessor performs communication overhead processing and other function dependent applications. Because the buffers are shared between the microprocessor and the USB-FS, a simple semaphore mechanism is used to distinguish who is allowed to update the BDT and buffers in system memory. A semaphore, the OWN bit, is cleared to 0 when the BD entry is owned by the microprocessor. The microprocessor is allowed read and write access to the BD entry and the buffer in system memory when the OWN bit is 0. When the OWN bit is set to

1, the BD entry and the buffer in system memory are owned by the USB-FS. The USB-FS now has full read and write access and the microprocessor must not modify the BD or its corresponding data buffer. The BD also contains indirect address pointers to where the actual buffer resides in system memory. This indirect address mechanism follows diagram as shown in the figure above.

An understanding of the addressing mechanism of the Buffer Descriptor Table is useful when accessing endpoint data via the USB-FS or microprocessor. Some points of interest are:

- The BDT occupies up to 512 bytes of system memory.
- 16 bidirectional endpoints can be supported with a full BDT of 512 bytes.
- 16 bytes are needed for each USB endpoint direction.
- Applications with less than 16 endpoints require less RAM to implement the BDT.
- The BDT Page Registers (BDT_PAGE) point to the starting location of the BDT.
- The BDT must be located on a 512-byte boundary in system memory.
- All enabled TX and RX endpoint BD entries are indexed into the BDT to allow easy access via the USB-FS or MCU core.

When a USB token on an enabled endpoint is received, the USB-FS uses its integrated DMA controller to interrogate the BDT. The USB-FS reads the corresponding endpoint BD entry to determine whether it owns the BD and corresponding buffer in system memory.

Table 35-2. BDT address calculation fields

Field	Description
BDT_PAGE	BDT_PAGE registers in the Control Register Block
END_POINT	END POINT field from the USB TOKEN
TX	1 for transmit transfers and 0 for receive transfers
ODD	Maintained within the USB-FS SIE. It corresponds to the buffer currently in use. The buffers are used in a ping-pong fashion.

When the USB-FS transmits or receives data, it computes the BDT address using the address generation shown in "Addressing Buffer Descriptor Entries" table.

If OWN =1, the following process occurs:

1. The USB-FS reads the BDT.

2. The SIE transfers the data via the DMA to or from the buffer pointed to by the ADDR field of the BD.

3. When the TOKEN is complete, the USB-FS updates the BDT and, if KEEP=0,

changes the OWN bit to 0.

4. The STAT register is updated and the TOK_DNE interrupt is set.

5. When the processor processes the TOK_DNE interrupt, it reads from the status register all the information needed to process the endpoint.

6. At this point, the processor allocates a new BD so that additional USB data can be transmitted or received for that endpoint, and then processes the last BD.

The following figure shows a timeline of how a typical USB token is processed after the BDT is read and OWN=1.

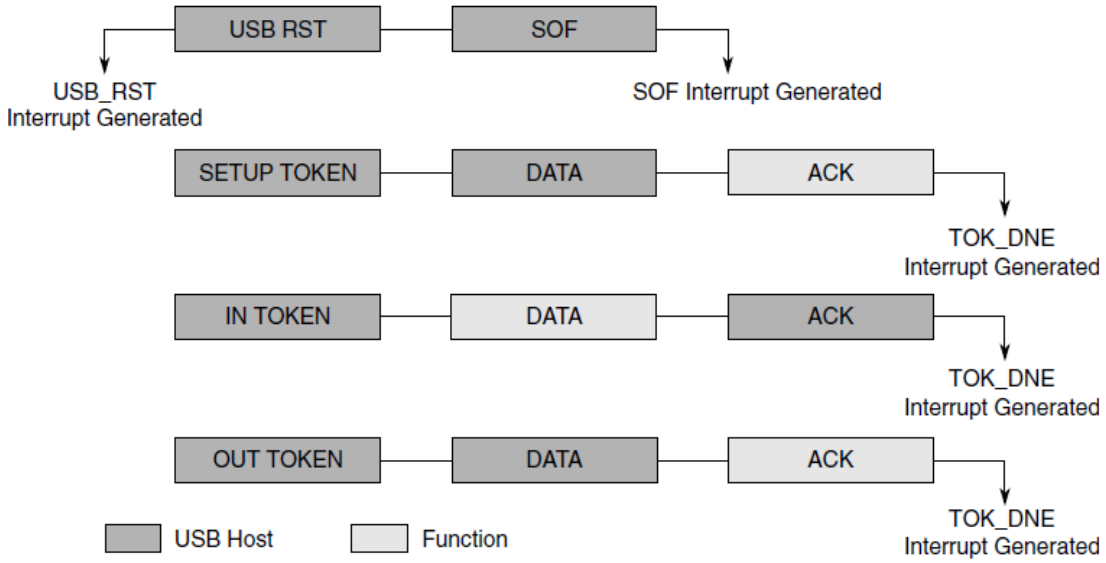

Figure 35-4. USB token transaction

KL25 通用串行总线 USB OTG 控制器

1． USB

USB 是一种支持主机与多个数据交换总线电缆可同时访问外设的范围。连接的外设共享 USB 通过主机的带宽调度，基于令牌的协议。总线允许外设连接，配置，使用，和分离，而主机和其他外围设备操作。

USB 软件提供了所有应用软件系统的统一视图，隐藏实施细则，使应用软件更轻便。它管理外设的动态连接和分离。

在任何 USB 系统中只有一个主机。USB 接口的主机系统被称为主机控制器。有可能在任何系统如操纵杆、音箱有多个 USB 设备、打印机，USB 设备等方面的理解，一个标准的 USB 接口，响应，以及标准能力。

由 USB 主机启动同 USB 设备的通信，然后 USB 设备对主机的控制信息进行回复。USB 设备通过标准的 USB 数据格式与主机进行数据交互。 USB2.0 支持全速与低速传输，其速度分别为 12Mb/s、1.5Mb/s。更多的信息参考 USB2.0 协议规范。

2． USB OTG 功能

USB 是一种通用标准用来连接外设以及手持电子产品，比如：数码相机、手持电脑、PC 机。USB 协议规范增加的 USB OTG 功能扩展了 USB 端到端的应用，使用 OTG 技术消费电子产品、外设以及手持设备能够连接到其他的设备用来交换数据，比如一个数码相机可以直接连接一台打印机，键盘可以直接与 PDA 相连。

作为具有 OTG 功能的 USB 设备，该设备必须能够完全兼容 USB 协议，因为该设备有时需要充当 USB 主机的角色。基于硬件信号由软件来决定设备的角色，然后依据设备是如何连接的用相应的操作模式来初始化设备。连接上之后，设备可以根据所要完成的任务使用 OTG 协议来协商自己的角色，是作为主机还是作为 USB 从机。有关详细的信息，可以参考“On-The-Go Supplement to USB 2.0”协议规范。

3． KL25 USB-FS

USB-FS 2.0 全速/低速模块通过状态寄存器、控制寄存器以及内存中的数据接口与处理器核心进行通信。KL25 的 USB-FS 具有以下特点：

（1）支持 USB1.1 以及 USB2.0 全速设备控制器；

（2）16 个双向端点；

（3）DMA 或 FIFO 数据流接口；

（4）低功耗；

（5）支持 OTG 协议。

KL25 中为了高效地管理端点之间的通信，USB-FS 在系统内存中实现了一个称之为缓冲区描述符表的数据结构。BDT 表占据系统内存的一个连续的 512 字节空间，该空间的起始地址由 BDT 页面寄存器指向。每个端点的方向都需要 2 个 8 字节的缓冲区描述符实体。因此一个具有 16 个双向端点的系统一共需要 512 字节的内存空间用来实现 BDT。端点的每个方向上的 2 个缓冲区描述符实体分别为 EVEN BD 以及 ODD BD。这样允许控制器在操作其中一个 BD 的同时 USB-FS 操作另外一个 BD。采用双缓冲区描述符的方法允许 USB-FS 很容易地从 USB 总线上以最大的速度传输得到的数据。

软件 API 在需要时通过更新 BDT 来智能地为 USB-FS 管理缓冲区。这样 USB-FS 可以高效地管理数据的发送与接收，同时微控制器负责处理和通信以及其他的功能性程序开销。因为缓冲区是由 USB-FS 与微控制器所共享的，因此需要有一个简单的信号量机制来决定允许谁来更新 BDT 以及系统中的缓冲区。当微控制器正在操作 BD 实体时，信号量比特 OWN 位被清 0。当 OWN 位为 0 时，微处理器可以读写 DB 实体以及系统内存中的缓冲区。当 OWN 位为 1 时，说明 USB-FS 具有对 BD 实体以及系统中缓冲区操作权，此时 USB-FS 拥有对 BD 表项以及相应的数据缓冲区的读写权限，而微控制器不能对该表项以及相应内存中的缓冲区进行任何操作。BD 描述符具有指向内存中缓冲区地址的指针，这种间接地址机制可以同下图 1 中看出来。

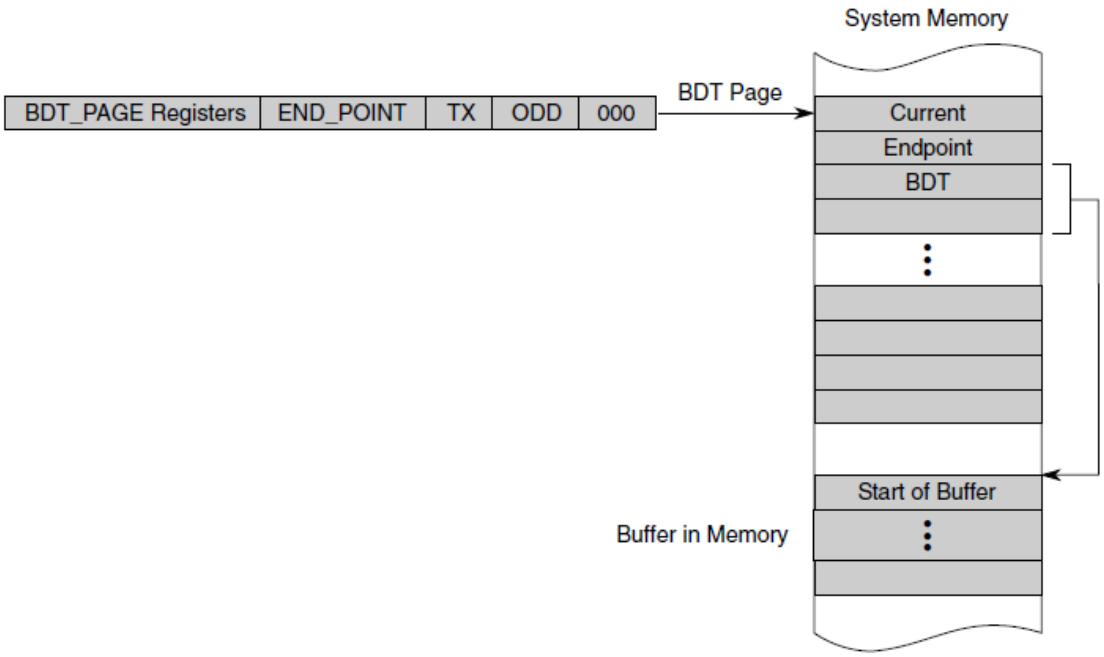

图 1　BDT 缓冲区描述表

当通过 USB-FS 或者微控制器来访问端点数据时，理解缓冲区描述符表的寻

址机制是很有用的。一些要点如下：

（1）缓冲区描述符表占据整个系统内存的 512 字节；

（2）16 个双向端点完全可以通过整个 512 字节来存储；

（3）每个端点的一个方向需要 16 个字节；

（4）如果应用程序中的端点少于 16 个，那么可以通过更少的空间（少于 512 字节）来实现 BDT；

（5）BDT 页面寄存器指向 BDT 表的起始地址；

（6）BDT 表必须被定位在系统的一个连续 512 字节区间；

（7）所有端点的 TX 以及 RX 描述符表最好通过索引的方式存到 BDT 表内，这样 USB-FS 以及微控制器可以很轻松的对其进行访问。

当一个开启某个端点的 USB 令牌到达时，USB-FS 通过其内部集成的 DMA 控制器来查询 BDT 表。USB-FS 通过对相应端点的 BD 实体来决定他是否拥有该描述符表项以及该表项在内存中的缓冲区。

为了计算实体在当前 BDT 表中的位置，BDT_PAGE 寄存器将当前的 endpoint（令牌命令的端点地址）、TX 字段、ODD 字段连在一起组成一个 32 位的地址。这种地址机制由下表 1 所示。

表 1　BDT 地址计算字段

字段	描述
BDT_PAGE	控制寄存器块中的 BDT_PAGE 寄存器
END_POINT	USB 令牌命令中的端点地址
TX	1：表示发送 0：表示接收
ODD	该比特由 USB-FS 的 SIE 位决定。相当于当期正在使用的缓冲区。表示是 DATA0 还是 DATA1。

当 USB-FS 发送护着接收数据时，它将通过 35.3.3 节给出的计算地址机制来计算 BDT 的地址。 如果 OWN =1，则进行下列操作：

（1）USB-FS 读取 BDT 表；

（2）SIE 通过 DMA 向由 BD 计算得到的地址存取数据；

（3）当令牌完成时，USB-FS 更新当前的 BDT 表。如果 KEEP=0，将 OWN 位清 0；

（4）STAT 寄存器被更新，并且 TOK_DNE 中断标志位置 1；

（5）当微控制器处理 TOK_DNE 中断时，会从状态寄存器中读取有关要处理端点的所有信息；

（6）于此同时，微控制器重新开辟一个 BD 保证后续的数据能被正确存取

在当前端点，然后开始处理最后的那个 BD。 如下图 2 显示了一个当 BDT 被读取并且 OWN=1 时一个 USB 令牌被处理过程。

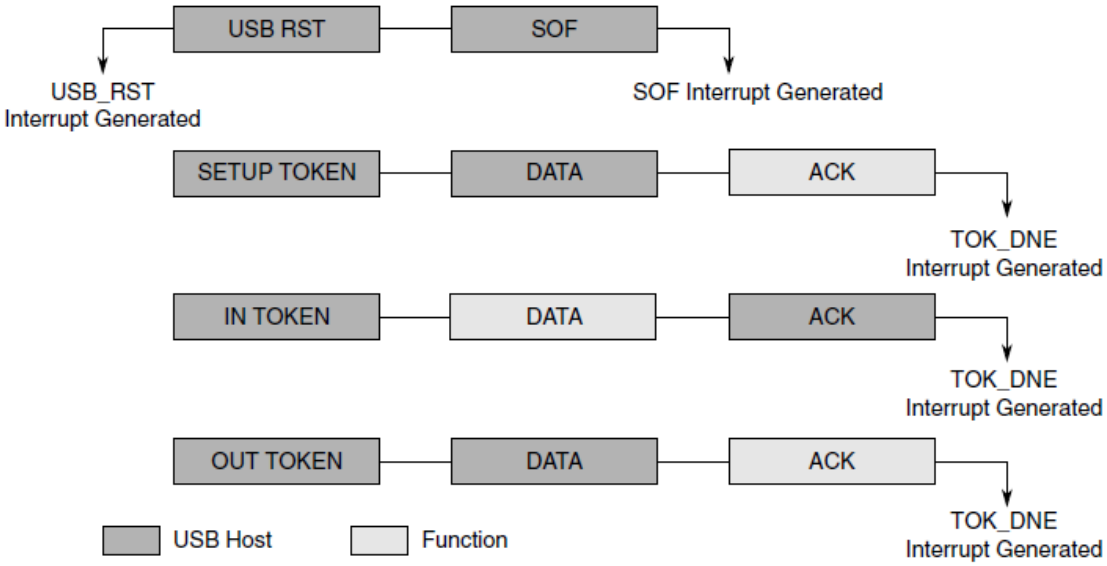

图 2　USB 令牌传输过程

致　谢

在公司毕业设计这一段时间以来，在长达两个多月的时间里，让我增长了很多见识，学到了 USB 与蓝牙方面的知识，增长了我对公司 AMetal 的认识与 Aworks 系统的了解。更为重要的是，在这一过程锻炼了我与队友，同事之间打交道的能力与分工合作的能力，在一点点中慢慢的改变了我偏内向的性格，增强了自信心。在这忙碌的时间里，增强了我对工作责任感的认识，提高了工作的抗压能力。同时在工作中应当主动出击，虚心向老前辈学习，一路坚持下来总是会有收获的。

在学校这四年来，很怀念与各位同学一起学习的日子，感谢你们，在我求学路上充满欢声笑语。特别感谢宿舍室友在我校外做毕业设计的这一段时间以来，帮我打理学校的一些事务。感谢各位自动化的老师，这三年多来他们辛勤的上课为我打下了比较扎实的专业基础。

最后，感谢广州致远电子的公司领导对于实习生在毕业设计与毕业实习这一阶段的重视，提供了很多毕业设计相关的仪器仪表、相关的芯片器件给我做毕设研究，同时为我们营造了良好的实习环境。感谢我在公司的导师李庆大哥，他为我们小组毕业设计过程当中给予了我们方向引导与技术支持。感谢小组队友之间的相互扶持，相互打气，正是由于你们，我们才是一个整体，才能顺利的完成毕设。感谢我在学校的导师韩老师，她为我毕业设计涉及到的课程打下了坚实的基础以及在毕业论文写作期间对我耐心的指导，感谢各位为自动化专业默默付出的老师，是他们教会了我很多专业知识。感谢学校，是母校培养了我。感谢父母，是他们在我求学路上一直默默的支持我。

在此，对于毕业设计期间给予我帮助的队友们，各位同事，各位领导，导师以及学校的各位同学，自动化专业的任课老师表示感谢，谢谢大家！