西 南 交 通 大 学

本科毕业设计（论文）

平衡轮腿机器人地形自适应控制设计

THE TOPOGRAPHY ADAPTIVE DESIGN OF A BIPEDAL LEG-WHEELED ROBOT

年　　级：2018 级

学　　号：2018110101

姓　　名：唐绍武

专　　业：机械电子工程

指导老师：宋兴国

2022 年 6 月

摘要

传统的轮式机器人和足式机器人由于自身结构上的缺陷，无法同时满足高能量效率、高移动能力、高通行能力的现代化实际需求。而两足轮腿机器人兼备了足式机器人的灵活性能与轮式机器人的移动性能，具有较强的环境适应能力和障碍通行能力，被认为是拥有广泛应用前景的新型移动机器人。近年来，越来越多的国内外专家学者关注到两足轮腿机器人领域并展开相关研究。然而，由于两足轮腿机器人的发展起步较晚，在结构设计和控制方案等方面的研究还不够成熟，使得两足轮腿机器人离实际应用仍然有一定距离。本设计从结构方案设计、运动学建模、机电系统设计、运动控制和软件设计四个方面展开研究，设计制作了一款两足轮腿机器人样机，并基于样机开展了一系列测试实验。

在结构方案设计与运动学分析中，首先从自由度数目、机构构型、整体尺寸三个方面展开分析，提出了 4 自由度轮腿融接型的两足轮腿机器人结构方案。基于结构方案对腿部结构进行运动学分析和静态动力分析，为机器人的硬件选型和样机制作提供理论基础。

在机电系统设计中，根据机器人结构方案，从机械结构设计、动力系统硬件设计、控制系统硬件设计三个方面展开介绍。建立了机器人的三维 CAD 模型并完成了髋关节电机、主控芯片、传感器等电子硬件的选型。完成了两足轮腿机器人机电系统的设计。

在运动控制设计与软件设计中，首先介绍了 PID 控制方案及其控制原理。然后基于 PID 控制设计了机器人运动控制器，包括直立控制器、直行控制器、高度调节控制器等。最后通过软件设计来充分调用主控制板、传感器、控制硬件、动力硬件，用计算机程序实现了机器人的各功能需求和运动控制。

在完成了机器人的结构方案设计、运动学建模、机电系统设计、运动控制以及软

件设计后，本文进行了一系列测试实验和实验分析，验证了两足轮腿机器人机电系统设计和运动控制方法的正确性。

关键词：两足轮腿机器人；运动学建模；机电系统设计；运动控制

Abstract

Due to their structural defects, traditional wheeled robots and bipedal robots cannot meet the modern practical needs of high energy efficiency, high mobility, and high traffic capacity at the same time. The bipedal leg-wheeled robot combines the flexibility of a bipedal robot with the mobility of a wheeled robot, and has strong environmental adaptability and obstacle-passing ability. It is considered to be a new type of mobile robot with wide application prospects. In recent years, more and more domestic and foreign experts and scholars have paid attention to the field of bipedal leg-wheeled robots and carried out related research. However, due to the late development of bipedal leg-wheeled robots, the research on structural design and control scheme is not mature enough, which makes bipedal leg-wheeled robots still have a certain distance from practical application. In this design, a prototype of a bipedal leg-wheeled robot is designed from the aspects of structural scheme design, kinematic analysis, electromechanical system design, motion control and software design.

In the design and kinematics analysis of the structure scheme, the three aspects of the number of degrees of freedom, the mechanism configuration and the overall size are analyzed first, and the structure scheme of the bipedal leg-wheeled robot with 4 degrees of freedom leg-wheeled fusion type is proposed. Then, the kinematics analysis and static dynamic analysis of the leg structure are carried out based on the structural scheme, which provides a theoretical basis for the hardware selection and prototyping of the robot.

In the electromechanical system design, according to the robot structure scheme, three aspects are introduced: mechanical design, power system hardware design, and control system hardware design. The three-dimensional CAD model of the robot is established and the selection of electronic hardware such as hip motor, main control chip and sensor is

completed.

In motion control design and software design, PID control and its control principle are first introduced. Then the robot motion controller is designed based on PID control, including upright controller, straight controller, controller, height adjustment controller and so on. Finally, the main control board, sensors, control hardware and power hardware are fully invoked through software design, and the functions and motion control of the robot are realized by the program.

After completing the structural design, kinematic analysis, electromechanical system design, motion control and software design, a series of test experiments and experimental analysis were carried out to verify the correctness of the electromechanical system design and motion control method of the biped leg-wheeled robot.

Keywords： bipedal leg-wheeled robot, kinematics modeling, mechatronics design, locomotion control

目录

第1章 绪论

1.1 课题研究背景和意义

21 世纪以来，机器人技术受到了国内外著名企业、高等院校、科研机构的广泛关注，被认为是未来新兴产业技术之一。移动机器人作为机器人领域分支之一，它融合了计算机技术、信息技术、机械设计技术和控制理论，是一个完整而复杂的机电一体化系统。其在物流分拣、军事侦察、社会服务、工业巡检等领域有较强的应用前景[1]。

近年来，移动机器人的发展趋于多元化，各种创新的设计方案和控制方法层出不穷。将移动机器人按照运动方式分类，可以分为履带式机器人、足式机器人、轮式机器人以及轮腿复合式机器人四种类型[2]。其中，履带式移动机器人通过履带与地面之间的摩擦力进行移动。由于履带与地面的接触面积较大且履带结构具有一定挠性，使得履带式机器人拥有较强的地形适应能力、越障能力和负载能力，通常被用来执行野外或复杂环境任务。但同时履带式机器人也受限于自身结构，无法解决移动速度较慢、转向困难等缺点而没有得到快速的发展。足式机器人一般设计有特殊的仿生腿部结构，拥有一定程度的生物移动能力，在复杂环境中拥有较强的机动性和适应性。足式机器人通过足端与地面的交替接触，来实现在地面上的移动。当为其融合传感器技术、机器视觉技术后可以拥有强大的地形通行能力。但足式机器人有自由度多、控制难度大的缺点，往往会有较高的开发成本。同时由于足式机器人系统的复杂性和高耦合度，实际表现效果往往欠佳，目前还无法胜任复杂的地形通行任务。轮式机器人通过驱动轮表面与地面的滚动摩擦力来实现在地面上的移动，拥有远高于履带和足式机器人的移动速度。轮式机器人的驱动轮一般由减速电机直接驱动，具有较高的能量效率，被认为是最高效的地面移动机器人类型[3]。但轮式机器人应用场景十分有限，它们的越障能力与驱动轮半径直接相关。即使用上减震技术、悬挂技术也无法完全弥补结构上

的缺点，很难在凹凸不平的复杂地形环境上执行任务。

轮腿复合机器人技术的发展是建立在轮式机器人和足式机器人基础之上的。随着人类生活的不断发展和科学技术的更新迭代，单一的轮式机器人和足式机器人由于自身结构上的固有缺陷，无法同时满足非结构化环境[4]、高能量效率、高移动能力、高通行能力的实际需要，移动机器人技术瓶颈亟待突破，轮腿机器人应运而生。轮腿机器人将足式机器人的腿部结构和轮式机器人的轮部结构结合于一身[5]，拥有足式机器人高通行能力和轮式机器人高移动能力两方面优点，具有更加广泛的应用前景。轮腿机器人技术在近年来得到了迅速发展，成为国内外移动机器人领域学者的研究热点。

轮腿机器人按照结构类型的不同，可以分为串联轮腿融接型和并联轮腿独立型两种；按照腿部末端滚轮是否存在动力，可以分为从动轮式腿轮机器人和主动轮式腿轮机器人。另外，根据机器人腿部的数量还可以分为两足、多足轮腿机器人。目前对于轮腿机器人的研究大部分集中在四足[6]、六足[7]等多足轮腿机器人，然而实际环境中往往有许多狭窄过道、台阶，这使得多足轮腿机器人的应用受到限制。因此，对拥有更小体积的两足轮腿机器人进行研究具有特殊意义。近年来，国内外许多研究团队都展开了对两足轮腿机器人的控制方法研究和机器人样机制作，并取得了丰硕的成果，但是离两足轮腿机器人的实际应用还有一段距离。本设计拟展开对两足轮腿机器人的机械结构、控制硬件、系统建模、控制软件等相关技术的研究，设计并制作一款可行的两足轮腿机器人。本设计将有助于拓展两足轮腿机器人在结构方案、运动学模型和控制方法等方面的研究内容。

1.2 两足轮腿机器人国内外研究现状

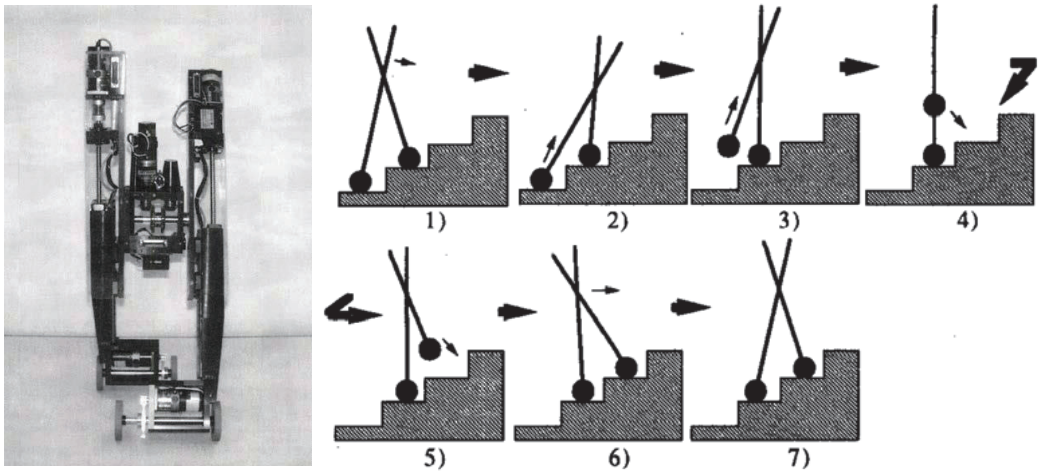
两足轮腿机器人的研究目标是综合足式机器人的地形适应能力和轮式机器人的运动能力两方面优点于一身，使机器人具有更加广泛的应用前景。国内外许多有机器

人研究机构及专家学者曾对两足轮腿机器人展开过详细研究，并取得了丰富的研究成果。本节针对他们的研究成果，从设计结构、动力学模型以及控制方法等方面进行介绍分析。

1.2.1 国外研究现状

（1）日本学者 Osamu Matsumoto 等人在 1998 年研发了一款名为 Bird type 的串联轮腿融合型机器人，如图 1-1a 所示。该机器人的总重量 10.0kg，有效载荷 2.0kg，宽度为 230mm，高度在 575mm~740mm 内可调节。全身共设 5 个自由度，从上到下分别是腿部上端安装的 2 个伺服电机，通过滚珠丝杆来传递运动，实现腿部下端的伸出和缩回；腰部安装的 1 个伺服电机用以调节机器人在俯仰方向上两腿之间的相对角度；以及足部安装的两个直流电机直接驱动轮组，保证机器人在平稳地形上的高速移动。

Bird type 机器人可以在轮式运动模式和腿式运动模式之间自由切换，轮部的轮组保证了机器人单足触地时在横滚方向上的稳定。另外，将陀螺仪和角加速度计获取机器人的俯仰角速度和角加速度信息作为机器人调整俯仰姿态的依据。图 1-1b 是学者 Osamu Matsumoto 及其团队为 Bird type 机器人开发的一整套攀爬阶梯步态[8][9]。



a）Bird type 机器人样机　　b）Bird type 机器人攀爬阶梯过程

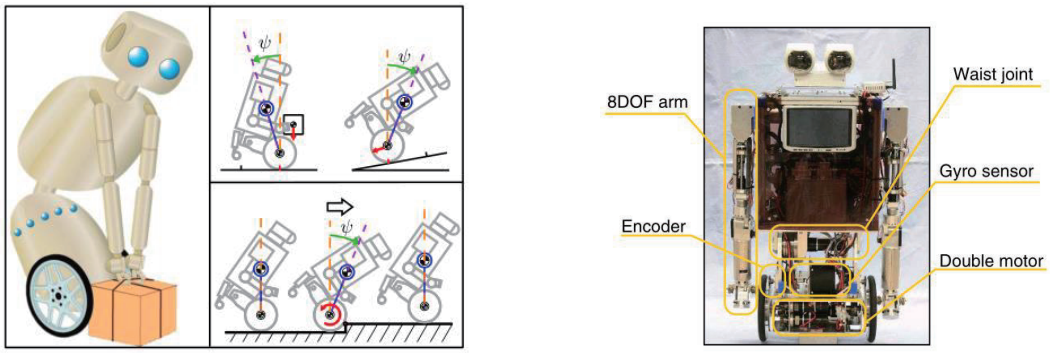
图 1-1 Bird type 机器人

（2）日本学者 Kenji Hashimoto 等人于 2005 年成功研制了一款串联型两足轮腿机器人 WS-2，如图 1-2 所示。该机器人是在 WL-16 机器人原型的基础上改进而来的，整体结构分为机体、腿部和脚掌三个部分。左右腿部都分别安装有多根伺服推杆，通过伺服推杆之间的相对长度来改变脚掌的倾斜角度，以实现切换轮式和足式运动形态的功能。当地面平整稳定时，腿部伺服推杆内侧收缩，外侧伸出，两外侧轮子着地，通过轮子转动来实现轮式运动；当地面凹凸崎岖时，伺服推杆相应变化使得内侧橡胶垫着地，切换为足式运动模式。在这种模式下，采用 ZMP 轨迹规划控制来保证机器人行走的动态稳定。WS-2 成功融合了轮式机器人和腿式机器人的两方面优点，两种模式之间的相互配合，大大提高了机器人对环境的适应能力和运行性能[10]。

图 1-2 WS-2 机器人

（3）葡萄牙学者 Luis Canete 等人在 2014 年研制了一款名为 IPENTAR 的机器人，如图 1-3 所示。该机器人总重量为 38kg，高度为 900mm，宽度为 300mm，整体机械结构上采用仿人型设计，拥有双臂双腿、膝肘关节等，而足端用轮子代替。IPENTAR 全身总共设有 19 个自由度，其中每条手臂设有 8 个自由度、腰部设有 1 个自由度以及左右两轮各设有 1 个自由度。该轮腿机器人的创新在于手臂引入，使得机器人除了通过前后转动轮子之外也可以通过手臂的前后摆动来改变重心的位置，拥有 2 种实现

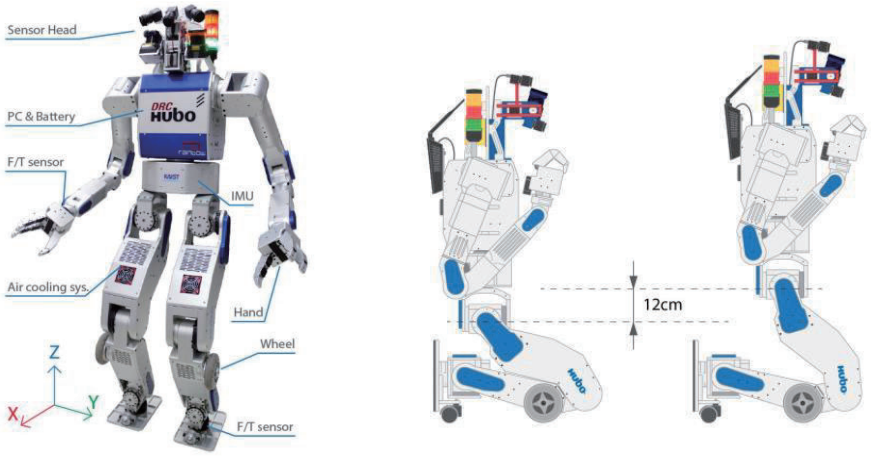
平衡的调整方式。另外，Luis Canete 等人也基于 IPENTAR 开发了补偿系统、反馈系统等，大大提高了机器人的运动精度和稳定性[11][12]。



a）IPENTAR 机器人平衡原理　　b）IPENTAR 机器人结构方案

图 1-3 IPENTAR 机器人

（4）韩国学者 Hyoin Bae 等人于 2015 年在 Hubo 双足机器人的基础上研发了一款名为 DRC-Hubo 的仿人并联轮腿机器人，如图 1-4a 所示。Hyoin Bae 等人巧妙地将主动轮设置在 DRC-Hubo 机器人的膝关节处，从动万向轮设置在足尖处，当切换为轮式运动模式后，其姿态与人类“跪姿”相似，可调节高度为 12cm，如图 1-4b 所示。在该模式下，整个机器人可以看作是一个四轮移动平台[13]，相较于两轮移动平台有着更好的稳定性能，同时也大大简化了运动开发的难度。此外，由于 DRC-Hubo 机器人自由度全面、传感器完备、功能强大、控制稳定性良好等优点，在 2015 年美国 DARPA 机器人挑战赛上获得了的冠军[14][15][16]。



a）DRC-Hubo 机器人结构方案　　b）DRC-Hubo 机器人轮式运动模式

图 1-4 DRC-Hubo 机器人

（5）美国波士顿动力公司（Boston Dynamics）在 2017 年发布了两足轮腿机器人 Handle 一代，如图 1-5a 所示。该机器人总重量 105kg，最大负载 45kg，总高度近 1.9m，关节采用液电混合驱动，有较好的运载能力。Handle 机器人基于 Atlas 机器人二次开发，运动能力大大提升，最大移动速度由原来的 5km/h 升级到 24km/h，最大跳跃高度近 1.2m。2019 年 3 月，波士顿动力公司又推出了第二代 Handle 机器人，如图 1-5b 所示。其在一代的基础上进行了结构和控制优化，同时增加了一个机械手臂。第二代 Handle 机器人的设计专注于仓库物流及货物搬运方面，相比于第一代 Handle 有着更好的越障、快速转弯、抗干扰和动态调节能力[17]。

a）Handle 一代机器人

b）Handle 二代机器人

图 1-5 波士顿动力 Handle 机器人

（6）苏黎世联邦理工学院的 Victor Klemm 等 9 名本科生于 2018 年研制了一款两足轮腿跳跃机器人 Ascento，如图 1-6a 所示。该机器人的最大运行速度为 8km/h，最大跳跃高度为 0.4m，本体高度可通过髋关节电机调节，可调节范围为 310mm~660mm。整体机械机构经过拓扑优化设计，全身大部分结构件由轻量化的尼龙材料 3D 打印制造，总重量仅有 10.4kg。Ascento 全身共有 4 个自由度，包括左右轮部、髋关节各 2 个自由度。Ascento 的腿部结构采用非线性设计，通过髋关节电机转动来调节腿部的屈伸状态，从而调节机器人高度。Ascento 内置 Intel NUC i7 主机和视觉传感器，拥有强大计算力，可以实现机器人障碍识别、自主导航等功能。在控制方面，该机器人采用的是线性二次调节器（LQR），在运动学模型的基础上设计了反馈跟踪和前馈控制器，

有较好的稳定性。该机器人的实际运行表现良好，在平稳道路上能稳定快速通行，在崎岖不平的道路上也能自动调节姿态，保持稳定。另外，Victor Klemm 等人为 Ascento 设计开发了跳跃控制器，使其采用跳跃的方式跨越阶梯等高度大于轮子半径的障碍。2021 年末，Victor Klemm 等人又发布了 Ascento Pro 机器人，如图 1-6b 所示。相比于上一代，该机器人主要升级在驱动轮尺寸和控制器优化方面，拥有更强的地形适应能力和越障能力[18][19]。

a）Ascento 机器人　　b）Ascento Pro 机器人

图 1-6 Ascento 和 Ascento Pro 机器人

1.2.2 国内研究现状

（1）哈尔滨工业大学机器人技术与系统国家重点实验室在 2018 年研发了一款名为 WLR 的两足轮腿机器人，如图 1-7 所示。WLR 机器人采用液电混合驱动方式，在结构上采用仿人型设计，总高度约 160mm，重约 50kg。WLR 机器人全身共具备 16 个自由度，从上到下分别是上肢两手臂各 3 个自由度、腰部 2 个自由度以及下肢两腿部各 4 个自由度。该项研究成果发表在 IROS 会议目上，展示了机器人平地移动、快速转向、上下斜坡、地形适应等功能[20][21][22]。

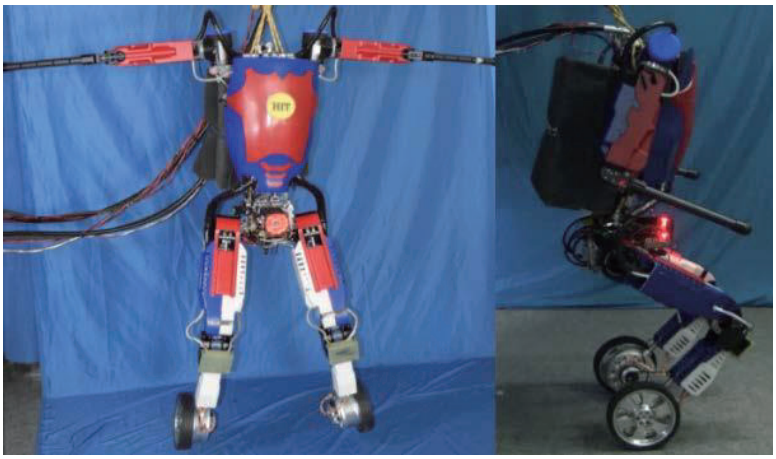


图 1-7 WLR 机器人

（2）腾讯公司的 Robotics X 实验室在 2021 年发布了一款两足轮腿机器人 Ollie，如图 1-8 所示，其研究成果发表在 2021 年 ICRA 会议上。Ollie 机器人全身具备 10 个左右对称的自由度，单边分别是髋关节处 2 个主动转动自由度，膝关节处 2 个被动转动自由度以及轮子处 1 个转动自由度。前 4 个自由度用于调整机器人高度，俯仰、横滚方向的倾角，轮部自由度由电机驱动机器人前进后退等。在控制策略上，系统计算倒立摆模型倾角和实际俯仰角的差值，采用线性系统的输出调节对该差值进行估计和滚动优化，再通过轮部电机转动来调整机器人姿态而保持平衡。Ollie 机器人行动灵活，稳定性好，可以实现高度调节、地形适应以及跳跃、前后空翻等复杂功能[23]。

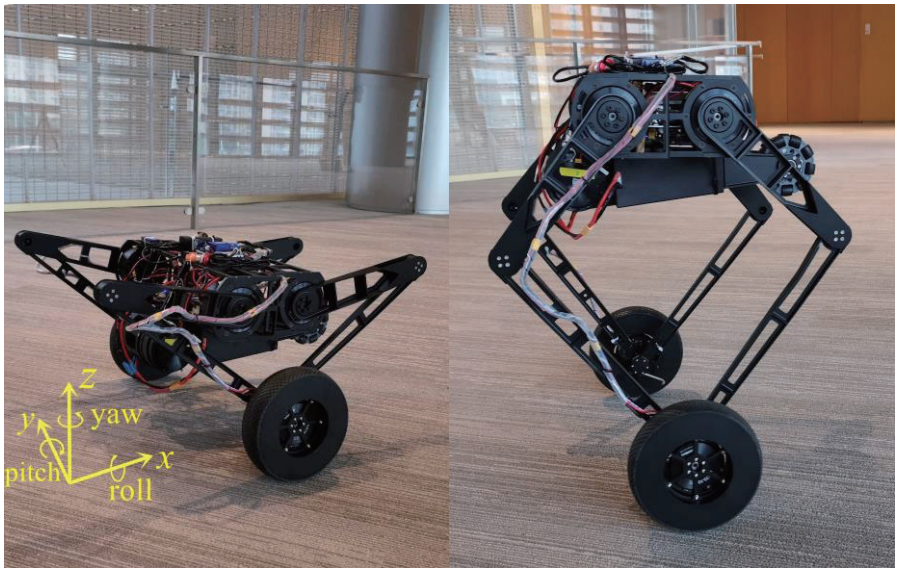


图 1-8 Ollie 机器人

1.2.3 国内外研究现状分析

早期国内外学者对于轮腿复合机器人的研究主要聚焦于四足、六足等多足领域，至今已积累了丰富的研究成果。自 2017 年波士顿动力公司发布了 Handle 机器人后，许多学者意识到两足结构的优势，开始将目光转移到对两足轮腿机器人的研究上，两足轮腿机器人迎来了蓬勃发展时期。尤其是近几年，国内外众多高校和公司，美国麻省理工学院、苏黎世联邦理工学院、哈尔滨工业大学、波士顿动力公司、腾讯公司等都积极开展相关研究。除了研究机构和研究人员数量激增以外，两足轮腿机器人近年快速飞速发展也得益于机器人硬件基础和控制理论逐渐成熟，为两足轮腿机器人完成多线程、高难度、大计算等复杂任务提供可能。

经过多年发展，虽然国内外学者在两足轮腿机器人的研究上已经取得了众多成果，但是离机器人的实际应用还有很长一段距离。一方面两足轮腿机器人本质上属于多模态复合机器人，目前已发布两足轮腿机器人的机械结构件和控制系统硬件未达到产业化水平；另一方面，相比于四足、六足等，两足轮腿机器人本体结构与地面仅有两线接触，无法实现俯仰方向的平衡，属于天然的非稳定系统，工作中必须依据控制理论持续施加动态控制，调整重心与接触线的位置关系来保持机器人平衡[24][25]。如何实现这一动态调整以及如何在保持平衡的同时执行工作任务、抵抗外部干扰、地形自动适应等都是关键技术问题。

1.3 本文主要研究内容与目标

本设计的目标是设计制作一款两足轮腿机器人样机，并基于该样机展开轮腿机器人的平衡、移动、姿态、地形自适应控制设计，期望该机器人能够结合足式和轮式机器人的两方面优点。针对这一目标，本设计将从结构方案设计与运动学分析、机电系统设计、运动控制研究与软件设计三个方面展开研究，如图 1-9 所示。

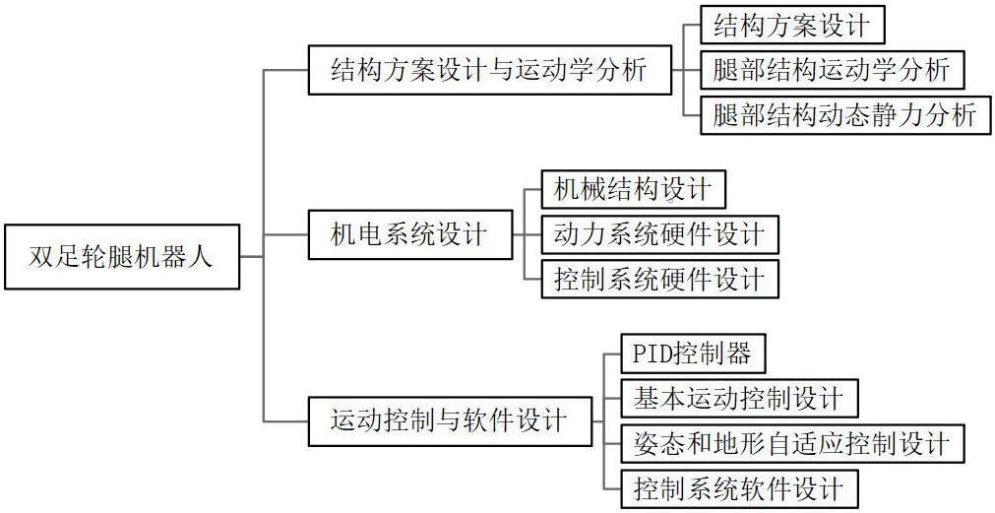


图 1-9 本文主要研究内容

结构方案设计与运动学分析：分析两足轮腿机器人功能需求，按照功能需求提出总体结构设计方案，并根据方案进行运动学分析。基于机器人平衡、地形自适应等功能需求，对机器人进行结构选型、自由度配置、尺度综合等设计，提出两足轮腿机器人总体设计方案。根据设计方案对机器人的腿部机构进行运动学分析、动态静力分析，为机器人的系统设计和样机制作提供理论基础。

机电系统设计：根据总体设计方案，完成相应的机电系统设计。基于两足轮腿机器人设计方案和功能需求，对动力系统硬件和控制系统硬件进行选型，包括髋关节电机、驱动轮电机、电机驱动器、主控芯片、传感器以及通信硬件，并对选定的电子硬件进行功能测试。功能模块测试完成后，进一步设计选型系统拓展板，将系统硬件集成封装起来，形成完整的两足轮腿机器人机电系统。

运动控制与软件设计：根据结构方案和机电系统对两足轮腿机器人进行运功控制研究。根据已经完成的结构方案和机电系统，设计关节驱动、传感器数据处理、硬件通信、人机交互等软件方案。基于软件方案开发平衡、移动、地形自适应控制方法，并不断优化提高控制方法稳定性和准确性，最终通过实验测试和数据分析来验证机器人结构方案和运动控制方法的可行性。

第2章 结构方案设计与运动学分析

2.1 引言

本章根据两足轮腿机器人的功能和运动需求、设计指标进行总体结构方案设计，包括自由度数目配置、结构构型设计、整体尺寸确定。在确定总体结构方案后，根据理论力学和机械原理相关知识对机器人进行运动学分析、动态静力分析，建立髋关节、膝关节与驱动轮之间的位置和速度关系；分析由自身重力而产生的反作用力对机器人本身的影响，为机器人样机制作提供理论基础。

2.2 结构方案设计

本设计的目标是设计制作一款两足轮腿机器人，其能够将轮式和腿式机器人两方面优点结合起来，在货物分拣、环境巡检、室内服务等领域发挥作用。因此该机器人需要具备关节控制、速度控制、地形自适应等运动能力，同时需要具备一定的人机交互功能。为了实现这一系列任务目标，本节从自由度数目、结构构型、整体尺寸三个方面具体分析，最终确定两足轮腿机器人结构设计方案。

自由度数目：为了降低系统的设计难度，机器人的自由度数目应该在满足任务需求前提下尽可能减少。参考简化人体 6 自由度的腿部结构，髋关节 3 个自由度来完成大腿的前后摆、左右摆以及转动；膝关节 1 个自由度来完成小腿前后摆动；踝关节 2 个自由度来调节脚掌与地面的接触角度。人体腿部结构十分灵活，在整体腿部屈伸时，往往是髋关节、膝关节、踝关节相互配合、协同运动，共同实现蹲下、站立、抬腿等动作。本设计的两足轮腿机器人的腿部结构无需和人类腿部一样灵活，但单腿至少设置 1 个驱动轮自由度保证机器人的移动和运行能力，1 个腿部自由度完成屈伸动作，

同时通过左右驱动轮转速差的方式来实现机器人转向。因此本设计将将单腿 2 自由度作为两足轮腿机器人的自由度数目方案，其中轮部和腿部各 1 个自由度。

机构构型：在确定了机器人单腿 2 自由度的设计方案后，需要合理的设置相应关节和腿部结构。轮腿机器人的轮腿复合方式有轮腿融接型和轮腿独立型两种[26]，其中轮腿独立型机器人的两种运动方式独立、相互之间不影响，如 DRC-Hubo 机器人。这种结构虽然比较简单直接，可以减少机构耦合，但是灵活性比较差，运动模式之间的切换需要一定的时间才能完成。轮腿融接型机器人轮部与腿部相耦合，如 Ascento 机器人，这种结构灵活性较强，但是需要持续施加控制来动态调整姿态以保持平衡。综上，本设计选择灵活性更强的轮腿融接构型。

整体尺寸：在确定自由度数目和结构构型后，需要进一步确定机器人的整体尺寸。该机器人的应用场景主要是在货物分拣、环境巡检、室内服务等领域，所以整体尺寸不能太大。同时由于关节力矩有限，腿部结构构件长度不能过长，避免驱动过载。结合外部因素和自身因素两个方面，将机器人长宽高尺寸初步设定在 200×200×250mm 以内，具体尺寸设计将在第 3.3 节机械结构设计中详细介绍。

综合自由度数目、结构构型、整体尺寸三方面因素，确定了 4 自由度轮腿融接型的两足轮腿机器人，图 2-1 是机器人结构的左视图。该机器人结构左右对称，主要包括腰身、腿部、轮部三个部分。

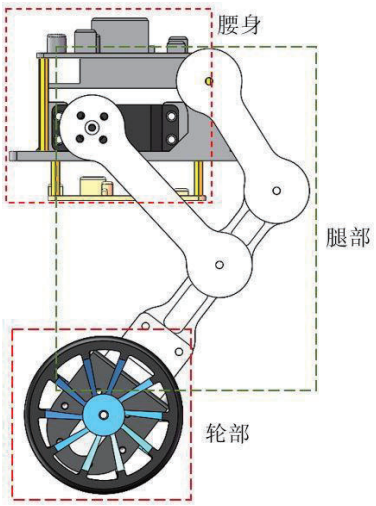


图 2-1 两足轮腿机器人结构方案

2.3 腿部结构运动分析

为了准确描述机器人腿部髋关节角度和轮部重心的位置、速度、加速度关系，本文根据两足轮腿机器人的结构方案，如图 2-2 在腿部结构上建立 *A-xy* 直角坐标系，其中图中 *A-E* 表示关节点，θ_1-θ_3表示杆件方位角。

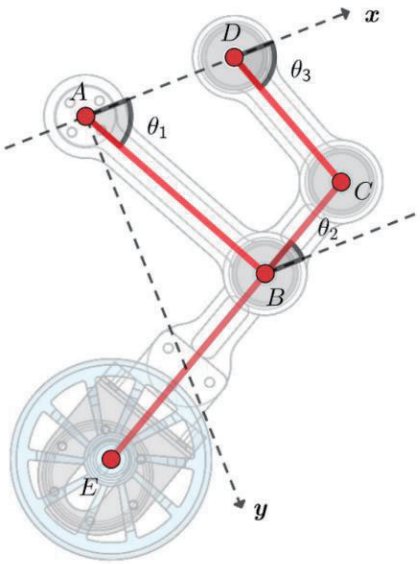


图 2-2 单腿 *A-xy* 坐标系

在机构设计时，常用矢量法来分析平面机构运动关系[27]。使用该方法首先需要将构件用矢量表示，设杆 *AD* 的长度为 l_{AB},其方位角为θ_1，$\vec{l}$为杆 *AD* 的杆矢量。则多边形 *ABCDA* 形成封闭杆矢量，在这个封闭杆矢量中有：

$$\vec{l}_{AB}+\vec{l}_{BC}-\vec{l}_{CD}-\vec{l}_{DA}=0 \tag{2-1}$$

因为髋关节舵机直接驱动 *A* 处转动副，所以θ_1为已知输入。而要想得到机器人腿部姿态就需要求得两个未知方位角θ_2和θ_3。

（1）位置分析

将封闭矢量方程式(2-1)用复数矢量形式表示：

$$l_{AB}e^{i\theta_1}+l_{BC}e^{i\theta_2}=l_{DA}+l_{CD}e^{i\theta_3} \tag{2-2}$$

应用欧拉公式$e^{i\theta}=cos\,\theta+i\,sin\,\theta$分离方程式(2-2)的实部和虚部，得：

$$\begin{cases} l_{AB}cos\theta_1+l_{BC}cos\theta_2=l_{DA}+l_{CD}cos\theta_3 \\ l_{AB}sin\theta_1+l_{BC}sin\theta_2=l_{CD}sin\theta_3 \end{cases} \tag{2-3}$$

由三角函数公式$cos\theta^2 + sin\theta^2 = 1$消去上式中$\theta_2$，得：

$$l_{BC}^2 = l_{CD}^2 + l_{DA}^2 + l_{AB}^2 - 2l_{AB}l_{CD}\,cos(\theta_3 - \theta_1) - 2l_{AB}l_{DA} - \theta_1 \tag{2-4}$$

将式(2-4)简化整理为：

$$A\,sin\,\theta_3 + B\,cos\,\theta_3 + C = 0 \tag{2-5}$$

式中 $A = 2l_{AB}l_{CD}\,sin\,\theta_1$；

$B = 2l_{CD}l_1\,(\cos\theta_1 - l_{DA})$；

$C = l_{BC}^2 - l_{AB}^2 - l_{CD}^2 - l_{DA}^2 + 2l_{AB}l_{DA}\,sin\,\theta_1$。

解得：

$$tan\left({}^{\theta_3}/_{2}\right) = \left(A \pm \sqrt{A^2 + B^2 - C^2}\right) / (B - C) \tag{2-6}$$

在求得θ_3后，利用式(2-3)可求得θ_2。式有两个解，根据实际安装情况和运动连续性来确定式中“$\pm$”号的正负。

（2）速度分析

将式(2-2)对时间 *t* 求导，可得：

$$l_{AB}\omega_{AB}\mathrm{e}^{\mathrm{i}\theta_1} + l_{BC}\omega_{BC}\mathrm{e}^{\mathrm{i}\theta_2} = l_{CD}\omega_{CD}\mathrm{e}^{\mathrm{i}\theta_1} \tag{2-7}$$

应用欧拉公式分离上式，再联解可求得两个位置角速度ω_{BC}和ω_{CD}，即：

$$\omega_3 = \omega_1\,sin(\theta_1 - \theta_2)/\,[l_{CD}\,\mathrm{sin}(\theta_3 - \theta_2)] \tag{2-8}$$

$$\omega_2 = -\omega_1\,sin(\theta_1 - \theta_3)/\,[l_{BC}\,\mathrm{sin}(\theta_2 - \theta_3)] \tag{2-9}$$

（3）加速度分析

将式(2-7)对时间 *t* 求导，可得：

$$il_{AB}\omega^2{}_{AB}\mathrm{e}^{\mathrm{i}\theta_1} + l_{BC}\alpha_{BC}\mathrm{e}^{\mathrm{i}\theta_2} + il_{BC}\omega^2{}_{BC}\mathrm{e}^{\mathrm{i}\theta_2} = l_{CD}\alpha_{CD}\mathrm{e}^{\mathrm{i}\theta_3} + il_{CD}\omega^2{}_{CD}\mathrm{e}^{\mathrm{i}\theta_3} \tag{2-10}$$

应用欧拉公式分离上式，再联解可求得两个位置角加速度α_{BC}和α_{CD}，即：

$$\alpha_{CD} = \frac{l_{AB}\omega^2{}_{AB}\,cos(\theta_1 - \theta_2) + l_{BC}\omega^2{}_{BC} - l_{CD}\omega^2{}_{CD}\,cos(\theta_3 - \theta_2)}{l_{CD}\,\mathrm{sin}(\theta_3 - \theta_2)} \tag{2-11}$$

$$\alpha_{BC} = \frac{-l_{AB}\omega^2{}_{AB}\,cos(\theta_1 - \theta_3) - l_{BC}\omega^2{}_{BC}\,cos(\theta_2 - \theta_3) - l_{CD}\omega^2{}_{CD}}{l_{CD}\,\mathrm{sin}(\theta_2 - \theta_3)} \tag{2-12}$$

以上过程求出了机构中所有构件的角位移、角速度、角加速度。对于杆 *CB* 延长线上点 *E* 的速度和角速度，其求解如下。

设 E 点在坐标系 A-xy 坐标系中的绝对位置矢量$\vec{l}_E = \overrightarrow{AE}$，则：

$$\vec{l}_E = \vec{l}_{AB} + \vec{l}_{BD} \tag{2-13}$$

即：

$$\vec{l}_E = l_{AB}\mathrm{e}^{\mathrm{i}\theta_1} + l_{BD}\mathrm{e}^{\mathrm{i}\theta_2} \tag{2-14}$$

将式(2-14)对时间 t 分别一次求导和二次求导，经变换整理可得$\overrightarrow{v_E}$和$\overrightarrow{\alpha_E}$的矢量表达式，即：

$$\begin{aligned}\overrightarrow{v_E} = &-(\omega_{AB}l_{AB}\sin\theta_1 + \omega_{BC}l_{BE}\sin\theta_2)\\ &+\mathrm{i}(\omega_{AB}l_{AB}\cos\theta_1 + \omega_{BC}l_{BE}\cos\theta_2)\end{aligned} \tag{2-15}$$

$$\begin{aligned}\overrightarrow{\alpha_E} = &-(\omega_{AB}{}^2 l_{AB}\cos\theta_1 + \alpha_{BC}l_{BE}\sin\theta_2 + \omega_{BC}{}^2 l_{BE}\cos\theta_2)\\ &+i(-\omega_{AB}{}^2 l_{AB}\sin\theta_1 + \alpha_{BC}l_{BE}\cos\theta_2)\\ &-\omega_{BC}{}^2 l_{BE}\sin\theta_2\end{aligned} \tag{2-16}$$

2.4 腿部结构动态静力分析

当机器人平衡运行时，根据牛顿第三定律，杆 CE 会受到由自身重力所引起的外力，这个外力经过杆件传递最终产生一个干扰 A 处驱动电机的正常运行的阻力矩，影响机器人平衡。设驱动电机输出力矩 M 可以消除外力 F 引起的影响，如图 2-3 单腿受力图建立坐标系。

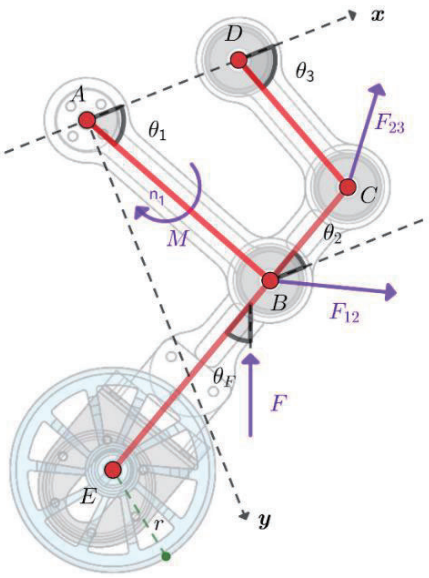


图 2-3 单腿受力图

（1）计算F_{23}（F_{23x}和F_{23y}）：取杆 *CD* 为分离体，对点 *D* 取矩，根据$\sum M_D = 0$，并应用欧拉公式可得：

$$\begin{aligned}\overrightarrow{l_{CD}^{t}} \cdot \overrightarrow{F_{23}} &= l_{CD}\mathrm{e}^{\mathrm{i}(90^{\circ}+\theta_3)} \cdot \left(F_{23x} + \mathrm{i}F_{23y}\right) \\ &= -l_{CD}F_{23x}\sin\theta_3 - l_{CD}F_{23y}\cos\theta_3 \\ &\quad + i\left(l_{CD}F_{23x}\cos\theta_3 - l_{CD}F_{23y}\sin\theta_3\right) \\ &= 0 \end{aligned} \tag{2-17}$$

由式(2-17)实部等于零，可得：

$$-l_{CD}F_{23x}\sin\theta_3 - l_{CD}F_{23y}\cos\theta_3 = 0$$

同理，取杆 *CE* 为分离体，对 *B* 点取矩，根据$\sum M_D = 0$，可得：

$$\begin{aligned}-\overrightarrow{l_2^{t}} \cdot \overrightarrow{F_{23}} + \overrightarrow{l_{BE}^{t}} \cdot \vec{F} &= l_{BC}\mathrm{e}^{\mathrm{i}(90^{\circ}+\theta_2)} \cdot \left(F_{23x} + \mathrm{i}F_{23y}\right) \\ &\quad + l_{BE}\mathrm{e}^{\mathrm{i}(90^{\circ}+\theta_2)} \cdot F\mathrm{e}^{\mathrm{i}\theta_F} \\ &= 0 \end{aligned} \tag{2-18}$$

由式(2-18)实部等于零，可得：

$$l_{BC}F_{23x}\sin\theta_2 + l_{BC}F_{23y}\cos\theta_2 - l_{BE}Fsin(\theta_2+\theta_F) = 0$$

联立上式可得：

$$F_{23x} = \frac{1}{\sin(\theta_2-\theta_3)}\left[\frac{F\cos\theta_3}{l_{BC}} \cdot l_{BE}\sin(\theta_2-\theta_F)\right] \tag{2-19}$$

$$F_{23y} = \frac{1}{\sin(\theta_2-\theta_3)}\left[\frac{F\sin\theta_3}{l_{BC}} \cdot l_{BE}\sin(\theta_2-\theta_F)\right] \tag{2-20}$$

（2）计算F_{12}（F_{12x}和F_{12y}）：根据杆 *BC* 平衡条件$\sum F = 0$，可得：

$$\begin{aligned}\overrightarrow{F_{12}} - \overrightarrow{F_{23}} + \vec{F} &= F_{12x} + iF_{12y} - F_{23x} - iF_{23y} + F\mathrm{e}^{\mathrm{i}\theta_F} \\ &= F_{12x} - F_{23x} + Fcos\theta_F \\ &\quad + i\left(F_{12y} - F_{23y} + Fsin\theta_F\right) \\ = 0 \end{aligned} \tag{2-21}$$

由上式实部和虚部分别相等，可得：

$$F_{12x} = F_{23x} - Fcos\theta_F \tag{2-22}$$

$$F_{12y} = F_{23y} - Fsin\theta_F \tag{2-23}$$

（3）计算平衡力矩 *M*：取杆 *AB* 为分离体，对点 *A* 取矩，根据$\sum M_A = 0$，可得：

$$\begin{aligned} M = \overrightarrow{l_{AB}^{t}} \cdot \overrightarrow{F_{12}} &= l_{AB} e^{i(90^{\circ}+\theta_1)} \cdot \left(F_{12x} + iF_{12y}\right) \\ &= -l_{AB}F_{12x}\sin\theta_1 - l_{AB}F_{12y}\cos\theta_1 \\ &\quad + i\left(l_{AB}F_{12x}\cos\theta_1 - l_{AB}F_{12y}\sin\theta_1\right) \\ &= 0 \end{aligned} \tag{2-24}$$

由式(2-24)实部等于零，可得：

$$M = l_{AB}F_{12x}\sin\theta_1 - l_{AB}F_{12y}\cos\theta_1 \tag{2-25}$$

联立式(2-19)、式(2-20)、式(2-22)、式(2-23)、式(2-25)，消除中间变量既可以得到驱动电机输出力矩 M 与外力 F 之间的关系。

2.5 本章小结

本章首先从自由度数目、结构构型、整体尺寸三个方面完成对两足轮腿机器人总体结构设计，提出了单腿 2 自由度轮腿融接型两足轮腿结构设计方案。然后基于理论力学和机械原理知识对机器人腿部结构进行运动分析和动态静力分析，建立了髋关节、膝关节与轮关节之间的位置和速度关系，分析了由自身重力而产生的反作用力对机器人本身的影响。

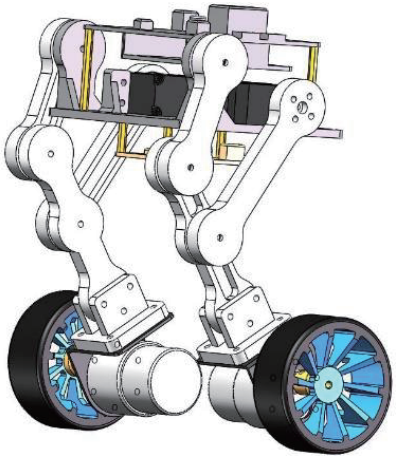
第3章 两足轮腿机器人机电系统设计

3.1 引言

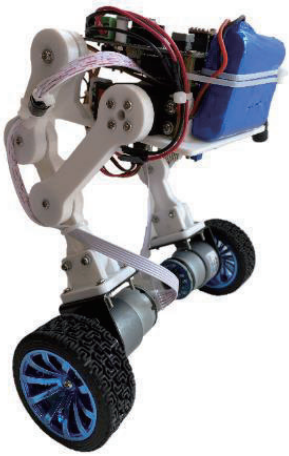
第二章确定了本设计4自由度轮腿融接型的两足轮腿机器人的结构方案并对该方案进行了运动学分析，建立了机器人的运动学模型和动态静力模型。在此基础上，本章从机械结构设计、动力系统硬件设计和控制系统硬件设计三个方面详细介绍机器人的机电系统设计。其中机械结构设计部分包括机器人腰身结构设计、腿部结构设计、轮部结构设计；动力系统部分主要介绍动力硬件设计参数及其选型依据；控制系统硬件设计将对主控芯片、传感器、通信硬件和系统拓展板进行介绍和选型。

3.2 机械结构设计

本设计两足轮腿机器人主要应用于货物分拣、环境巡检、室内服务等领域，同时具备一定的人机交互功能，运动上具备动态平衡、地形适应等能力。根据第二章确定的结构方案，该机器人主要有三个部分组成，包括腰身、腿部和轮部，其中腿部的髋关节和驱动轮各一个转动副，其三维CAD模型和样机如图3-1所示。



a）两足轮腿机器人三维CAD模型



b）两足轮腿机器人样机

图3-1 两足轮腿机器人

结构方案决定了机器人的平衡方式，4 自由度轮腿融接型机器人通过两驱动轮的同步动态调整实现直立平衡和移动，并采用左右驱动轮差速的方式来完成转向。同时，通过髋关节电机转动改变关节方位角，实现腿部的屈伸，达到地形自适应的目的。在三维模型设计方面，本设计使用达索公司的 SolidWorks 软件，该软件稳定性好、功能强大、简单易学，大大提高了机器人三维建模的效率。

腰身结构设计：机器人腰身部分结构如图 3-2 所示，该部分集中了机器人的控制硬件和关节电机。PLA 隔板是腰身的基础，左右两个对称的关节电机固定安装在隔板上；主控板和稳压电路板均通过定位孔和铜柱固定在 PLA 隔板上。通过这种隔板与铜柱结合的方式，可以很好的将电机、控制板等从空间上合理分层，降低安装和走线难度。整个腰身长宽高尺寸为：112×100×80cm。

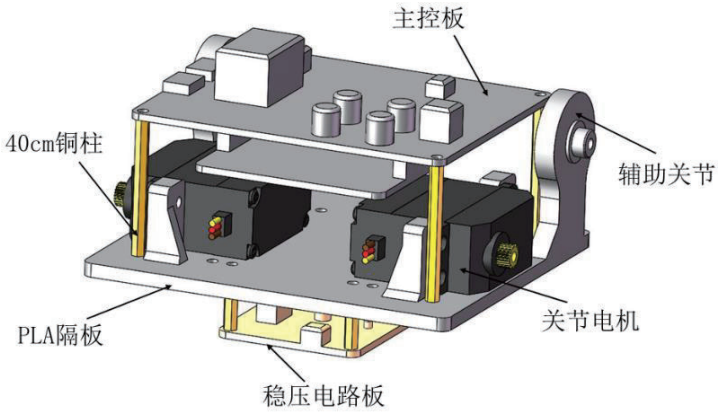


图 3-2 腰身结构设计

腿部结构设计：腿部结构采用平面四杆机构模型，如图 3-3 所示。单自由度单驱动的方式大大简化了机器人控制系统的复杂度。机器人大腿、小腿和辅助连杆均使用 PLA 材料通过 FDM 方式 3D 打印制造，比金属材料有更轻的重量，比机加工方式更加方便快捷且可以制造复杂构件。同时，通过轻量化减材设计可以进一步降低构件质量，减轻关节电机和驱动轮电机的运行负担。整个腿部结构包括大腿、小腿、辅助连杆和轴承。大腿作为平面四杆机构的主动件，其髋关节处通过舵盘于腰身髋关节电机连接，由髋关节电机直接驱动。小腿与大腿通过轴承在内关节处相连接，辅助连杆通

过轴承与小腿在膝关节处连接，膝关节和辅助关节作为被动关节，其运动规律服从平面四杆机构机械原理。尺寸方面，大腿、小腿、辅助关节厚度均为 9mm，其中大腿长 80mm，小腿长 80mm，辅助连杆长 55mm，内关节到膝关节长为 40mm。膝关节、内关节、辅助关节均采用 8×22×7mm 轴承作为中间连接件，有效降低了连杆之间相对转动的阻力，使得机器人腿部屈伸更加流畅。

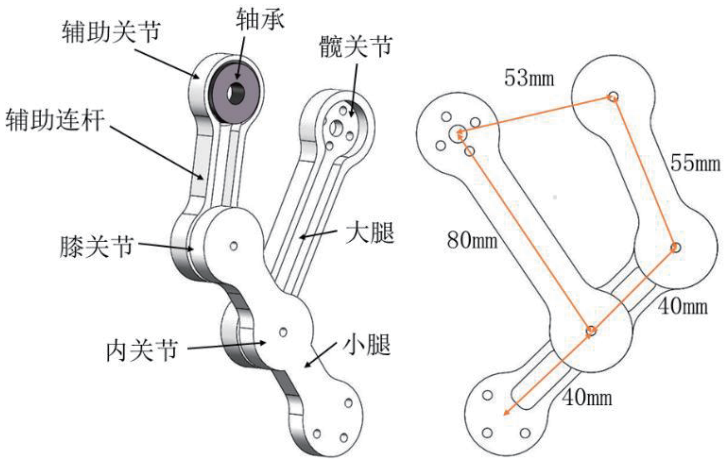


图 3-3 腿部结构设计

轮部结构设计：轮部结构驱动电机、驱动轮、电机支架等，如图 3-4 所示。驱动电机的 D 形输出轴与驱动轮连接，搭配紧定螺钉可以保证机器人在正常运行时，驱动轮的转速始终与驱动电机保持同步。电机支架是连接驱动电机与机器人小腿的关键零件，同时该零件承受机器人腰身和腿部的全部重量，因此采用 3mm 铝合金板材来保证刚度和强度。

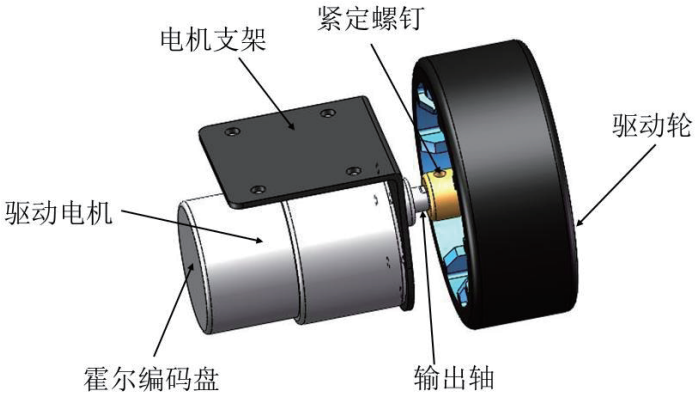


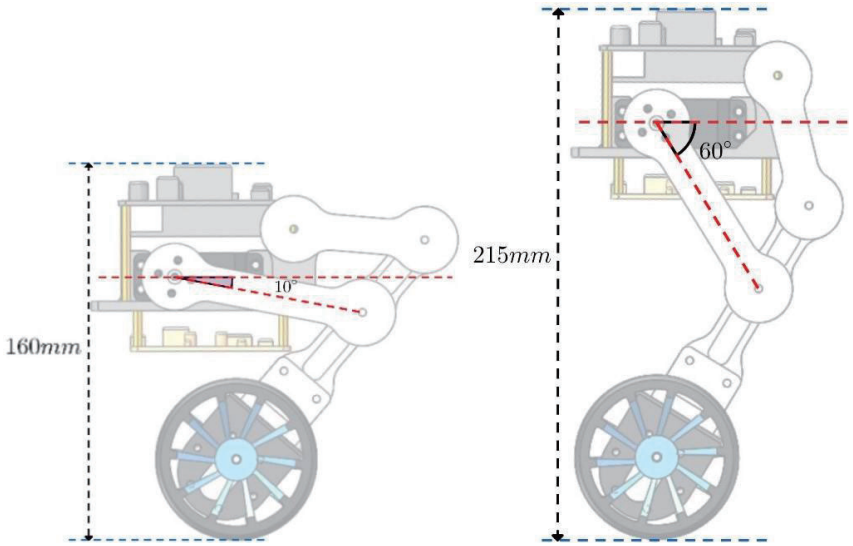
图 3-4 轮部结构设计

本设计两足轮腿机器人通过驱动轮转动，动态调整机器人位姿来保持平衡和移动

和转向；通过髋关节电机转动来改变大腿方位角来实现调整机器人高度和地形自适应。前文腿部结构设计中指出了大腿、小腿、辅助连杆的长度，根据 Grashof 定理，有

$$l_{max} + l_{min} = 80cm + 53cm > l_1 + l_2 = 80cm + 55cm \quad (3\text{-}1)$$

式(3-1)说明在腿部平面四杆机构中，最长杆件与最短杆件之和大于其余两杆。所以本设计的髋关节为摆转副，即转动范围小于 360° 。为了避免腿部结构进入死区，本设计将机器人髋关节电机转动范围限制在 10° 到 60° 之间，如图 3-5 所示。当髋关节电机在 10° 时，机器人处于高度最小位置，此时高度为 160mm，髋关节电机在 60° 时，机器人处于高度最大位置，此时高度为 215mm。两者的差值即为机器人高度调节范围。



a）机器人最小高度　　b）机器人最大高度

图 3-5 机器人最小高度和最大高度

机器人机械结构设计基于简单实用、轻量化的原则，设计构件大量使用 PLA 材料通过 FDM 方式 3D 打印制造。该材料本身具有轻量特点，同时结合轻量化减材设计，大大降低机器人整体质量，减小驱动电机、髋关节电机的负载。

根据第二章确定的单腿 2 自由度轮腿融接型两足轮腿机器人结构方案，本节分别对机器人腰身、腿部、轮部结构设计进行了详细介绍。该机器人的详细机械结构设计参数如表 3-1 所示。

表 3-1 机器人机械结构参数表

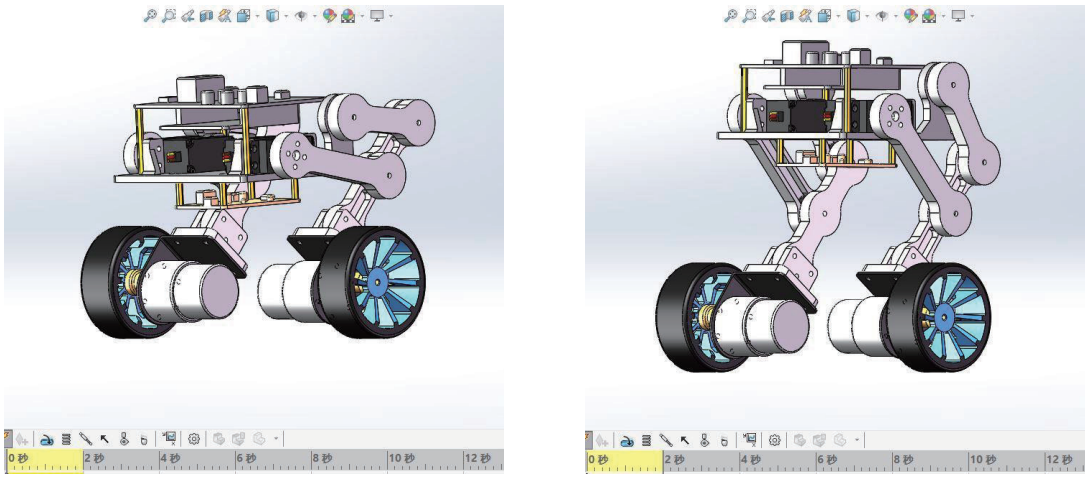
结构参数	数值	单位
腰身宽度	210	mm
大腿长度	80	mm
小腿长度	80	mm
最小高度	160	mm
最大高度	215	mm
髋关节电机调节范围	10~60	度
驱动轮半径	35	mm
整体质量	1.112	kg

3.3 动力系统硬件设计

3.3.1 髋关节电机

由于任务和功能需求，当两足轮腿机器人通过传感器感知到异常位姿或通过蓝牙接收到姿态指令时，关节执行器需要快速做出反应以保持平衡或改变姿态。因此，该机器人的关节执行器需要具备位置控制的功能，同时对快速性、准确性和稳定性也有一定要求。常见的关节执行器主要有步进电机和伺服电机两种类型。其中步进电机执行器因为其控制简单、无累计误差且拥有良好的启停和反转特性等优点被广泛应用在数控机床、工业自动化等领域。而伺服电机执行器拥有控制精度高、矩频特性好、稳定性好的优点被广泛应用于火花机、机械臂等。

本设计在髋关节执行器的选型和设计中，采用 SolidWorks 软件中的 Motion 分析来对髋关节扭矩进行物理仿真，如图 3-6 髋关节扭矩 Motion 仿真所示，整个仿真过程是由图 3-6a 所示的腿部收缩状态到图 3-6b 所示的伸展状态，仿真时间为 0~2s。首先将机器人三维 CAD 模型中各个零件添加材料属性，赋予机器人与真实世界相同的物理环境。然后新建 Motion 运动算例，设置好时间参数、引力、摩擦和关节执行器运动参数等后开始进行仿真。



a）仿真第 0s 时的机器人状态　　b）仿真第 1s 时的机器人状态

图 3-6 髋关节扭矩 Motion 仿真分析

仿真完成后，从“结果和图解”中得到时间-马达扭矩的关系曲线，如图 3-7 所示，其中 0~1s 是从机器人腿部收缩状态到伸展状态过程中的动态关节力矩，而 1~2s 是机器人腿部保持伸展状态是的关节力矩。

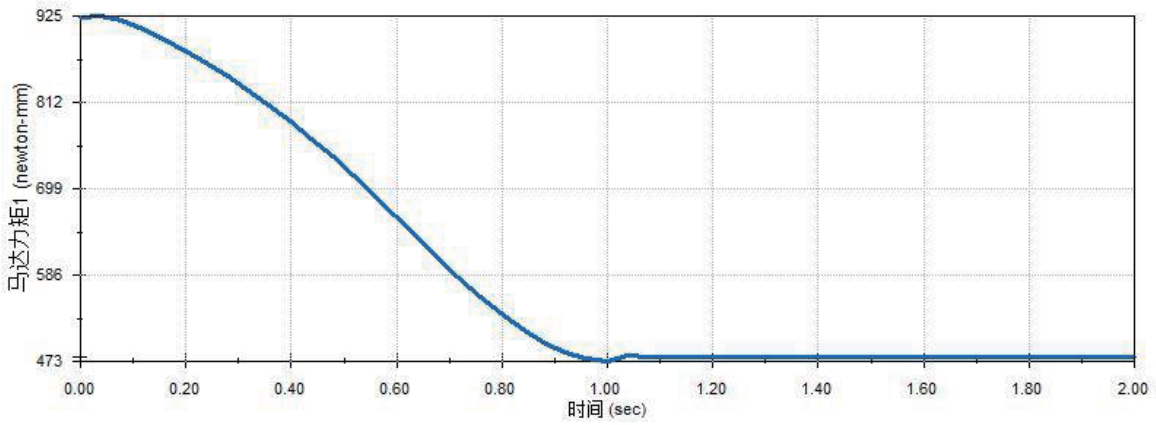


图 3-7 髋关节力矩 Motion 仿真结果

从图中可以看到，从 0~2s 仿真过程中，髋关节电机最大力矩约为 925N·mm，最小力矩约为 473N·mm，静力矩略大于最小力矩，保持在 480N·mm 左右。根据驱动力矩、尺寸、重量等因素，本设计最终选择达盛科技公司的 DS3115 伺服电机作为关节执行器，其外形如图 3-8 所示。

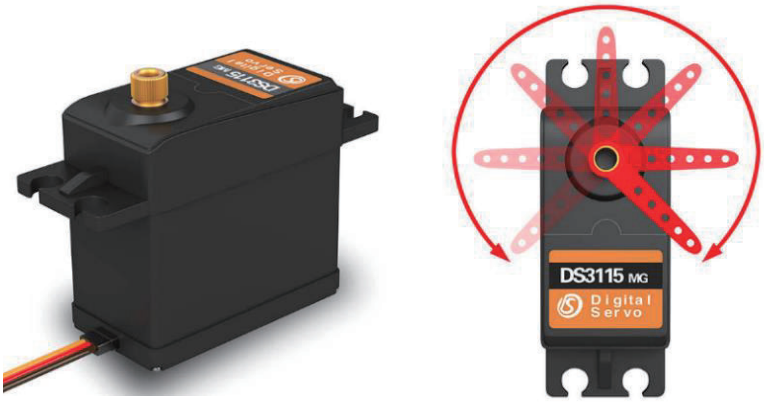


图 3-8 DS3115 电机外形

该伺服电机工作电压为4.8~6.8V，堵转扭矩为15kg，尺寸长宽高为20×40×40mm，重量为58g，内置直流电机、减速齿轮组、控制电路板等。DS3115共有三条输入线，分别是电源线、地线和信号线，其中信号线是电机与控制板通信的物理接口。该电机的控制信号为周期20ms的脉冲调制信号，控制原理如图3-9所示，脉冲宽度0.25ms~2.5ms与电机输出角度-90°~90°之间为线性关系，即电机的输出角度与输入脉冲宽度直接相关。

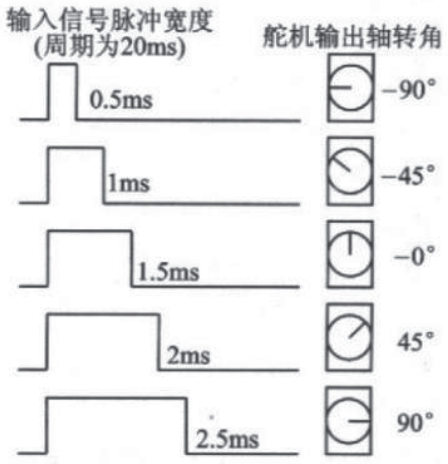


图 3-9 舵机角度控制原理

DS3115伺服电机的详细参数如表3-2所示。

表 3-2 DS3115 伺服电机参数表

型号	精度	工作电压	堵转扭矩	空载电流	质量
DS3115	0.3°	4.8~6.8V	15kg	100mA	58g

3.3.2 驱动轮电机

两足轮式机器人运行时，需要轮部驱动轮电机持续动态调整转动方向和转动速度以维持机器人平衡状态[28]，所以对驱动轮电机有一定的功率和扭矩要求，且驱动电机必须拥有频繁换向和快速响应的能力。在具体驱动轮电机选型之前，首先对机器人建立系统力学模型，对机器人驱动轮电机的功率和扭矩进行初步分析。将两足轮式机器人运动过程中的位姿状态建立力学模型，如图 3-10 所示。

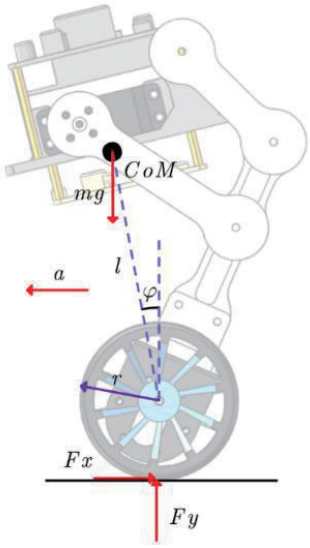


图 3-10 两足轮腿机器人力学模型

假设机器人因受到外部扰动而前倾φ角，为了回到平衡状态，机器人必须通过驱动轮转动产生一个向前的回复加速度，由此回复加速度产生的惯性力需满足：

$$mgl\,sin\,\varphi = mal \cdot cos\,\varphi \tag{3-2}$$

式中 l—质心到轮心的距离；

a—回复加速度；

m—机器人质量（除去两驱动轮）。

对式(3-2)进行化简，可得：

$$a = gtan\,\varphi \tag{3-3}$$

得到回复加速度a后，需要进一步分析产生该回复加速度的驱动力矩。

驱动轮线速度和角速度之间有：

$$v = \omega \cdot r \tag{3-4}$$

对式(3-4)两侧同时对时间t求导，可得：

$$\frac{\mathrm{d}\omega}{\mathrm{d}t}=\frac{1}{r}\cdot\frac{\mathrm{d}v}{\mathrm{d}t}=\frac{a}{r} \tag{3-5}$$

设机器人绕轮心的转动惯量为J_m，驱动轮绕轮心的转动惯量为J_r，则机器人动能为$\frac{1}{2}J_m\omega^2$，其中

$$J_m=mr^2 \tag{3-6}$$

$$J_r=m_r r^2 \tag{3-7}$$

驱动轮所受滚动阻力矩M_f为：

$$M_f=F_y\cdot r=\mu\cdot(m+m_r)\cdot g\cdot r \tag{3-8}$$

综上可以得到机器人驱动轮的总驱动力矩 M 为：

$$M=(J_m+2J_r)\cdot\frac{\mathrm{d}\omega}{\mathrm{d}t}+2M_f=(m+m_r)\cdot g\cdot r\cdot(2\mu+tan\,\varphi) \tag{3-9}$$

每个驱动轮提供的驱动力矩M_1为总驱动力矩的一半，即：

$$M_1=\frac{M}{2} \tag{3-10}$$

根据机器人的机械结构和运动需求，将机器人质量$m=1.012\mathrm{kg}$，驱动轮质量$m_r=0.1\mathrm{kg}$，重力加速度$g=9.8\mathrm{m/s^2}$，驱动轮半径$r=35\mathrm{mm}$，滚动阻力系数$\mu=0.0165$和最大回复角$\varphi=8°$带入上式可得：单个驱动轮的驱动力矩$M_1=0.033\mathrm{N\cdot m}$设机器人最大移动速度$v=1.2\mathrm{m/s}$，对于驱动轮的角速度$\omega=34.28\mathrm{rad/s}$，则可以得出驱动轮电机输出功率为：

$$P=M_1\cdot\omega=1.13\mathrm{W} \tag{3-11}$$

将驱动轮电机输出功率和扭矩作为主要参考指标进行选型，综合质量、尺寸、安装方式等因素，本设计最终选择了 GB37-520 直流编码电机作为两足轮腿机器人的驱动电机，其外形如图 3-11 所示。

图 3-11 GB37-520 直流电机外形

该直流编码电机额定电压为 12V，额定功率为 4.8W，额定扭矩为 0.1N·m，减速比为 1：30，外载 AB 两相增量式霍尔编码器，通过编码器可以获取电机的转动信息，是反馈控制器设计的基础。GB37-520 电机详细参数如表 3-3 所示：

表 3-3 GB37-520 直流电机参数表

型号	额定电压	额定功率	额定扭矩	减速比	接口
GB37-520	12V	4.8W	0.1N·m	1：30	PH2.0

3.3.3 电机驱动器

髋关节电机稳压硬件：前文在髋关节电机选型中，确定了 DS3115 伺服电机作为本设计的髋关节电机。该电机控制精度高、响应速度快、可提供较大扭力。由于该电机工作电压为 4.8~6.8V，需要为其搭配一款稳压模块来保证正常运行。根据电压范围和尺寸大小，本设计选择了基于 LM2596 芯片的稳压模块，如图 3-12 稳压模块所示。该稳压模块输入 12V 电压，提供 5 个 3V、5 个 5V 固定电压以及 5 个可调电压输出端。同时内置 4 个自恢复保险丝，可以有效保护控制电路。

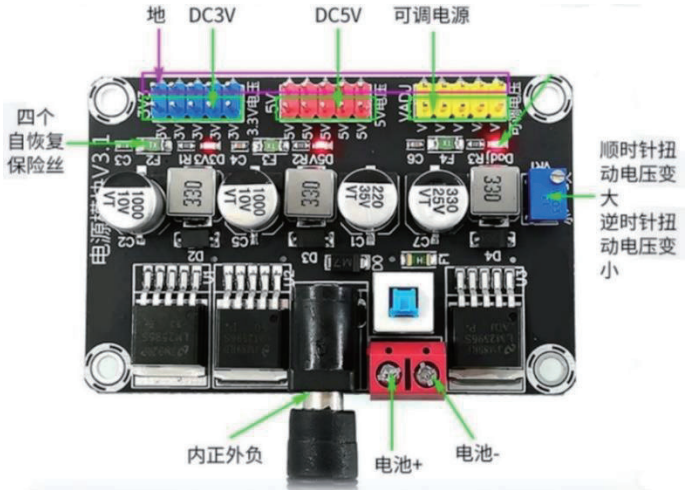


图 3-12 稳压模块

驱动轮电机驱动硬件：本设计的驱动轮电机 GB37-520 是一款直流有刷编码电机，驱动电压为 12V。结合电机参数，本设计选择了 TB6612FNG 驱动器作为 GB37-520 电机的驱动器，如图 3-13 所示。

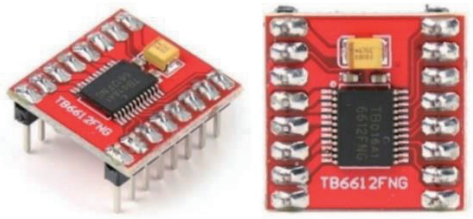


图 3-13 TB6612FNG 驱动器

TB6612FNG 驱动器基于大电流 MOSFET-H 桥结构，其效率高于晶体管 H 桥驱动器。该驱动器支持 100kHz 的 PWM 信号，保证 PWM 信号占空比调节范围，拥有较好的电机控制品质。同时模块尺寸较小，约为 20×20mm，有利于机器人控制系统的集成。另外该驱动器有双通道电路输出，可以同时驱动两个 GB37-520 电机，与本设计的需求贴合。TB6612FNG 的引脚共有 16 个，其引脚说明如表 3-4 所示：

表 3-4 TB6612FNG 驱动器引脚说明表

引脚名称	引脚功能
VM	电机驱动电压输入端（4.5~10V）
VCC	模块供电电压输入端（2.7~5.5V）
GND	电源地端
STBY	使能端
PWMA/PWMB	PWM 信号输入端
AIN1/BIN1/ AIN2/BIN2	电机控制模式输入端
AO1/BO1/AO2/BO2	电机驱动输出端

3.4 控制系统硬件设计

两足轮腿机器人的控制系统硬件设计包括主控芯片、传感器、通信硬件以及系统拓展板。其中主控芯片是整个控制系统的信息处理与控制中心，将来自传感器检测信息和外部输入指令进行处理、计算、分析和加工，再根据处理结果向执行单元发送相应的指令。传感器是机器人感知单元，对外界和机器人本身环境的状态信息进行检测，并将其转换成电信号，传输给信息处理单元。电源系统部分包括电源电路设计、动力系统供电以及保护电路设计。该机器人硬件系统框架如图 3-14 所示。

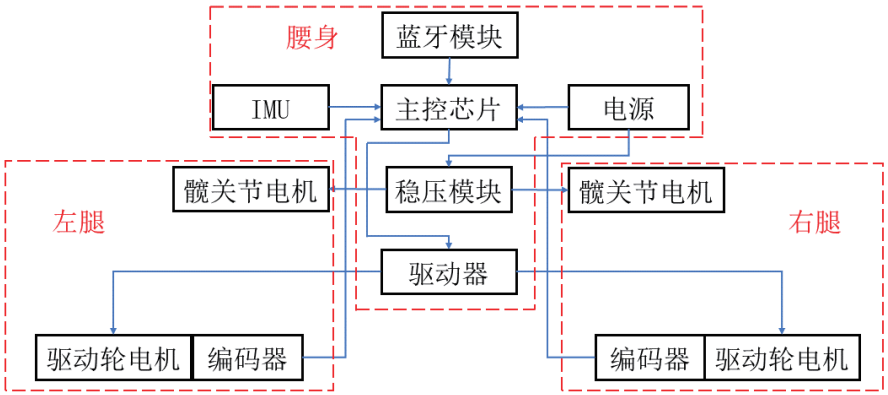


图 3-14 机器人硬件系统框架

3.4.1 主控芯片

主控芯片是机器人控制系统的信息处理与控制单元，在本设计中，主控芯片主要有三个任务：第一是将 IMU 传感器检测的姿态数据、驱动电机编码器的脉冲信号集中计算和处理，将所需要的数据进行记录和存储；第二是通过蓝牙模块接受来自操作人员的控制指令，包括前进、后退、调整高度等；第三是向执行机构发送指令，主控芯片按照预设程序，向驱动器发送指令，调整机器人姿态，使机器人可以保持直立平稳运行状态。

基于以上三个任务，本设计的主控芯片必须满足 5 个条件：1、有能够存一定量预设程序的 ROM 只读空间以及实时存取数据的 RAM 随机存取空间；2、有足够的计算能力，保证数据处理的实时性；3、有多个数字 I/O 通道、多个模拟输入通道和 PWM 通道，通过这些通道接受和发送数据；4、支持 I2C、SPI 等多种通信方式，可以建立起主控芯片与模块单元之间的信息传输通道；5、提供多个中断，包括定时器中断、外部中断等，保证程序执行的优先级。综合这 5 个条件，本设计选择了基于 ATmega328P 芯片的 Arduino Uno 开发板作为两足轮腿机器人的主控板，如图 3-15 所示。

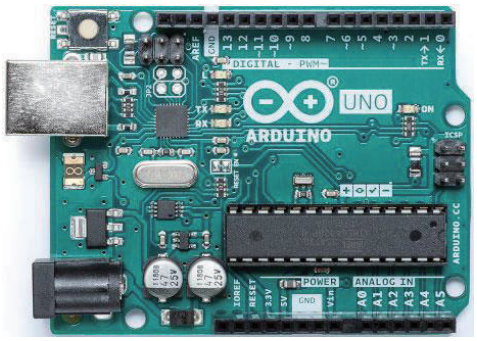


图 3-15 Arduino Uno

Arduino 作为开源硬件平台，经过多年发展，如今拥有大量的用户群体，积累了许多简单实用的库文件，让开发者不用阅读大量的芯片手册就可以进行软件开发。尤其是配合 Arduino IDE 集成开发环境，更加方便的管理和调用库文件，大大提高了软件开发效率。本设计选择的 Arduino Uno 控制板工作电压为 5V，内置 16MHz 晶体振荡器，拥有 1KB 的 EEPROM 空间、2KB 的 SRAM 空间和 32KB 的 Flash 空间；引脚方面：提供 6 个模拟输入引脚和 14 个数字 I/O 引脚，其中 6 个支持 PWM 输出。和单片机一样，Arduino Uno 控制板的大部分引脚都有第二功能，例如 pin2 和 pin3 可以输出外部中断信号、pin4 和 pin5 可以当作定时器使用等，详细的引脚功能如图 3-16 所示。

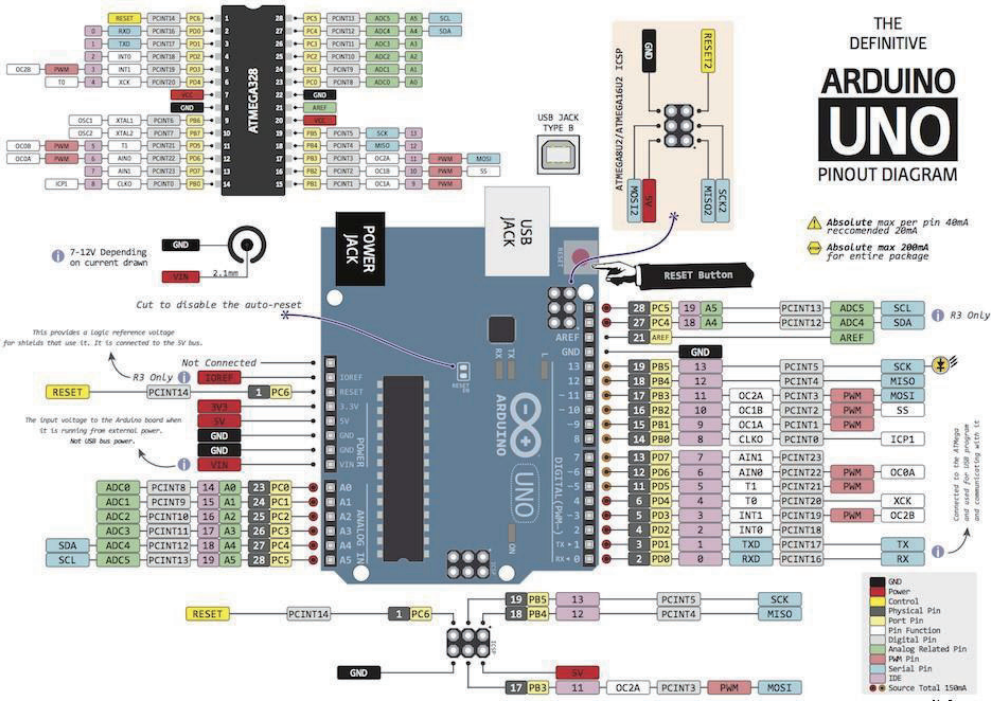


图 3-16Arduino Uno 引脚功能图

3.4.2 传感器

两足轮腿机器人保持平衡的原理是由主控芯片读取机器人的姿态信息和驱动轮的速度信息并进行计算、处理，根据信息处理结果向驱动轮发送指令使其做出相应调整，维持直立平衡状态。而完成调整后，机器人的姿态信息发生变化，主控芯片又读取新的传感器信息，如此形成反馈控制。在这个过程中，读取机器人姿态信息和速度信息都是重要环节，本小节针对机器人的功能需求，对姿态传感器和速度编码传感器进行设计和选型。

机器人姿态传感器：由于机器人的姿态信息数据传输量较大且有一定的实时性，需要姿态传感器内置有足够强计算能力的芯片，综合通信方式、外形尺寸等因素，本设计选择了 MPU6050 模块作为机器人的姿态传感器。该传感器的尺寸为 21×16mm，其外形如图 3-17 所示。

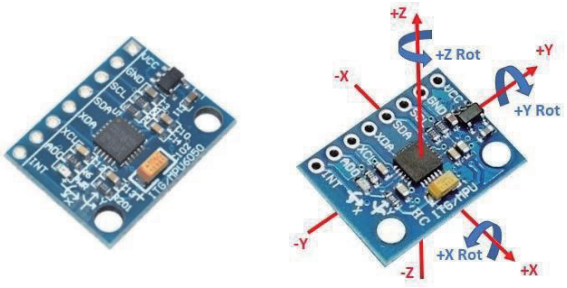


图 3-17 MPU6050 姿态传感器

MPU6050 芯片由 InvenSense 公司生产，是一款集成了三轴陀螺仪、三轴加速度计以及一个 DMP 数字运动处理器的六轴姿态传感器，其中三轴陀螺仪测量 XYZ 三轴方向的角速度，测量范围为每秒钟±250/±500/±1000/±2000°；三轴加速度计测量 XYZ 轴加速度，测量范围为±2/±4/±8/±16g；DMP 数字运动处理器接收陀螺仪和加速度计数据并做初步处理，再传输给主控芯片。在通信方面，该传感器采用 400kHz 的 I2C 接口，具有较高的传输速率和稳定性。MPU6050 模块共有 8 个引脚，其引脚名称和功能如表 3-5 所示：

表 3-5 MPU6050 引脚说明表

引脚名称	引脚功能
VCC	供电电压输入端
GND	电源地端
SCL	从机 I2C 串行时钟线
SDA	从机 I2C 串行数据线
XDA	主机 I2C 串行数据线
XCL	主机 I2C 串行时钟线
AD0	地址控制端
INT	外部中断

驱动轮速度传感器：在前文驱动轮选型中，本设计选择的 GB37-520 直流有刷电机附带霍尔编码器，形状为圆形，大小与电机相近，安装在驱动电机端部，如图 3-18 所示。

图 3-18 驱动电机霍尔编码器

霍尔传感器是以霍尔效应为基础的磁传感器[29]，通过检测磁场变化来反应环境状态。该霍尔编码器配有 11 线强磁编码盘，在 AB 双相输出和 1：30 减速比的共同作用下，电机输出轴每转动 1 圈，霍尔传感器的磁铁就会随之转动 30 圈，而磁铁每转动一圈，AB 两相位各产生 11 个脉冲信号。因此，驱动轮每转动一圈总共会产生 660 个脉冲信号。这样高频的脉冲信号保证了霍尔传感器的检测精度，使得驱动轮的速度信息更加准确可靠。在物理接口上，该霍尔编码器采用 6 针 PH2.0 接头与拓展版相连接，街头引脚说明如表 3-6 所示。

表 3-6 霍尔编码器引脚说明表

引脚名称	引脚功能
A	传感器信号线 A 相
B	传感器信号线 B 相
VCC	供电电压输入端
GND	电源地端
M+	电机电源线+
M-	电机电源线+

3.4.3 通信硬件

在机器人的人机交互设计中，常常 WIFI 和蓝牙两种通信方式，前者有高带宽、长距离等优势，而后者有更低的功耗、更小的体积以及更低的成本。综合考虑实际需求，选择了 HC-06 蓝牙模块作为本设计通信硬件。操作人员使用手机发送控制指令，指令通过 HC-06 蓝牙模块传输给 Arduino Uno 主控板，在主控板-蓝牙模块-手机之间建立如图 3-19 所示的数据传输路径。

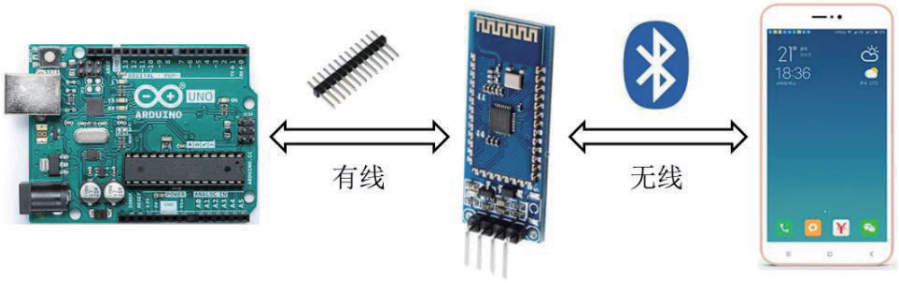


图 3-19 人机通信路径

HC-06 模块基于 BC417143 芯片开发而来，支持蓝牙 2.1 和 EDR 规范，工作电压为 3.3~6V，支持主从机模式，使用简易。本设计基于该模块开发了蓝牙串口数据通信协议，从而实现机器人的远程控制。HC-06 模块共有 6 个引脚，各个引脚名称及说明如表 3-7 所示。

表 3-7 HC-06 模块引脚说明表

引脚名称	引脚功能
KEY	从机连接端
引脚名称	引脚功能
VCC	供电电压输入端
GND	电源地端
TXD	数据发送端
RXD	数据接收
STATE	连接状态端

3.4.4 拓展板

前文介绍了机器人硬件部分的主控芯片、传感器、驱动器等各个模块的设计和选型，要想把这些功能模块合理集成起来组成机器人硬件系统，还需要一款设计合理的拓展板。该拓展板不仅要为各个功能模块提供物理接口，还需要将这些接口在空间上进行合理布置。根据机器人的实际需要，本设计选择了亚博智能科技公司开发的 Arduino Uno 拓展板，如图 3-20 所示。

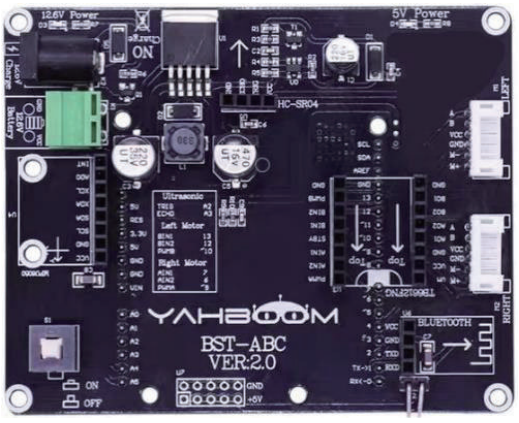


图 3-20 Arduino Uno 拓展板

这款拓展板设计了基于 LM2596 芯片设计的 12V 稳压电路，该电路包含 150kHz 频率振荡器和 1.23V 基准稳压电路，具有完善的保护电路、电流限制等作用。作为 Arduino Uno 的拓展板，提供了蓝牙模块、MPU6050 传感器、GB37 电机等接口，通过将各个功能模块集成在一起，大大简化了机器人硬件系统的线路布置，使机器人样

机在外观上更加简介。亚博智能科技公司开发的这款拓展板原理图如图 3-21 所示。

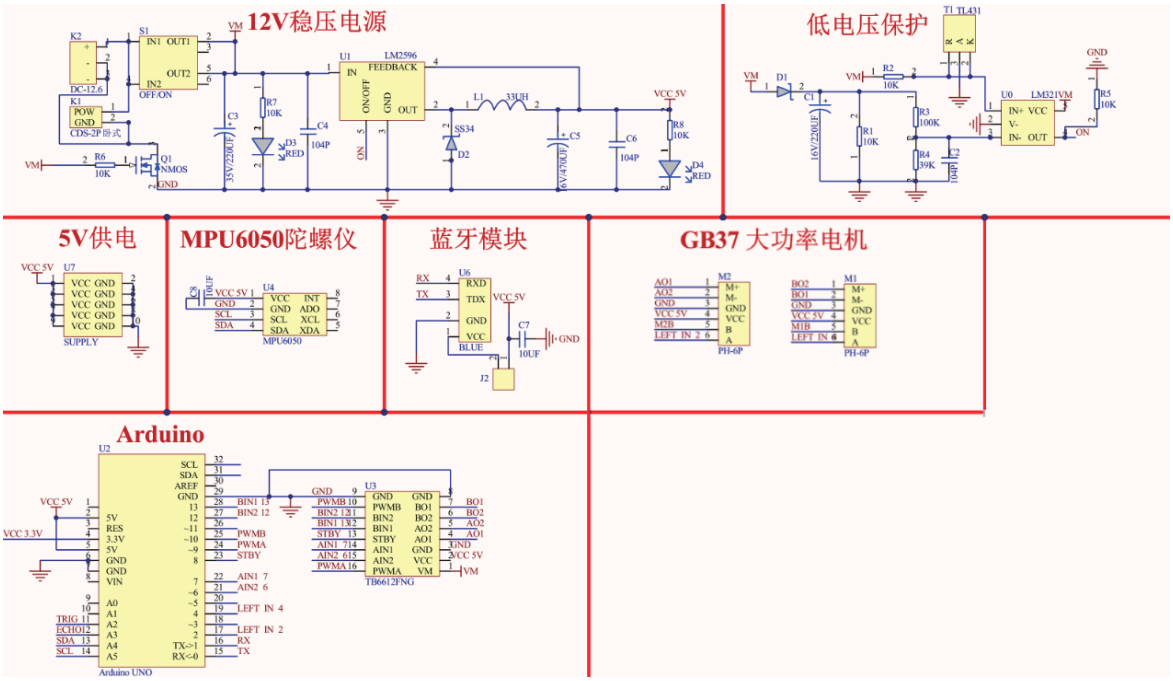


图 3-21 拓展板电气原理图

综上，本设计两足轮腿机器人的硬件系统设计完毕，包括动力系统硬件关节驱动轮、关节电机、驱动器和控制系统硬件 MPU6050 传感器、蓝牙模块，最后使用拓展板将各个功能模块集成起来。本设计机器人硬件系统的详细硬件名称和型号如表 3-8 所示。

表 3-8 机器人硬件表

硬件名称	型号
髋关节电机	DS3115
驱动轮电机	GB37-520
髋关节电机稳压板	LM2596 稳压模块
驱动轮电机驱动器	TB6612FNG
主控芯片控制板	Arduino Uno
六轴姿态传感器	MPU6050 模块
蓝牙模块	HC-06 模块
电源	12V 锂电池组
系统拓展板	亚博智能科技拓展板

3.5 本章小结

本章基于结构方案和动力学模型，从机械结构设计、动力系统硬件设计、控制系统硬件设计三个方面完成了机器人整个机电系统的结构设计和硬件选型。机械结构部分主要从尺寸设计、零件材料、设计思想等方面介绍了腰身、腿部、轮部的设计过程；动力系统硬件部分完成了髋关节电机、驱动轮电机及其驱动硬件的设计和选型；控制系统部分详细介绍了主控芯片、传感器、通信硬件的选型和设计。最后使用拓展板将动力系统硬件和控制系统硬件集成在一个电路板，使得机器人机电系统更加紧凑和简洁，为后期样机制作和测试实验提供方便。

第4章 两足轮腿机器人运动控制与软件设计

4.1 引言

在第 3 章中完成了机器人的机电系统设计和硬件选型后，需要进一步设计机器人的运动控制方法以及软件程序来充分利用各个功能模块，发挥它们的潜力，实现两足轮腿机器人的稳定控制。在控制方法方面，本设计主要利用了 PID 控制，因此本章首先对 PID 控制器进行介绍，然后分别对机器人的基本运动控制设计、机器人姿态和地形自适应控制设计以及控制系统的软件设计进行介绍。

4.2 PID 控制器

4.2.1 PID 控制原理

在连续控制系统中，PID 控制是应用最为广泛的调节器控制之一，其按照系统偏差量的比例、积分、微分规律进行组合控制。PID 控制距今已有近 70 年历史，由于它原理简明、结构简单、稳定性好、调整方便等优点，对于系统传递函数不需要深入了解，容易实现对模拟量的控制且控制性能较好而受到广泛应用[30]。

典型 PID 控制系统的原理框图如图 4-1 所示：

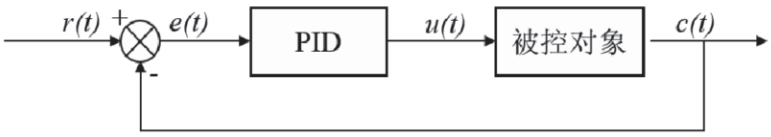


图 4-1 PID 控制原理框图

为了使系统稳定，则给定值$r(t)$与系统输出量$c(t)$的偏差量$e(t)$应趋于零，$u(t)$是关于时间的函数，是$e(t)$比例项、积分项、微分项的和。当$u(t)$施加在被控对象后会产生新的系统输出量$c(t+\Delta t)$，从而形成反馈控制，其控制规律可以总结为如下形式：

$$u(t) = K_P\left(\mathrm{e}(t) + \frac{1}{T_I}\int_0^t \mathrm{e}(\tau)\,\mathrm{d}\tau + T_D\frac{\mathrm{de}(t)}{\mathrm{d}t}\right) \tag{4-1}$$

式中 K_P—比例系数；

T_I—积分时间常数；

T_D—为微分时间常数。

为了将 PID 控制算法应用于计算机，还需要将式(4-1)离散化为如下形式：

$$u(k) = K_P\mathrm{e}(k) + K_i\sum_{j=1}^{k}\mathrm{e}(j) + K_d[\mathrm{e}(k), -\mathrm{e}(k-1)] \tag{4-2}$$

式中 K_p—比例控制器参数；

K_i—积分控制器参数；

K_d—微分控制器参数，

K_p、K_i、K_d均为常数，且三者之间相互独立。

4.2.2 比例、积分、微分控制

比例控制：在比例控制中，控制器的输出与偏差量成比例关系。比例控制的作用是放大或缩小偏差量幅值，控制效果取决于比例系数K_P的大小。当K_P较大时，偏差量将会被放大多倍而导致系统出现震荡；当K_P较小时，系统较为稳定，但由于控制效果较差而出现稳态误差，因此比例控制一般不单独使用。

积分控制：在积分控制中，控制器的输出与偏差量对时间的积分成比例关系。系统偏差量对时间的积分可以将误差量累计起来，会随着时间的增加而增大。这样即使系统偏差量很小，也会累计在积分项中，使得系统稳态误差进一步减小指导趋于零，也正因为积分项这一特性，常常与比例控制配合使用，消除系统的稳态误差。但积分控制会使系统的响应速度变慢，且较强的积分项会使系统超调量增大而破坏系统的稳定性。

微分控制：在微分控制中，控制器的输出与偏差量的积分呈正比关系。在控制系统中，有时存在惯性环节，即系统的输出滞后于反馈检测偏差的变化，从而使得系统振荡或失稳。而系统偏差量对时间的积分能预测偏差量，具有提前校正、改善超调的

作用，使系统趋于稳定，提高系统动态性能。但微分控制往往会放大噪声信号，因此一般也不单独使用[31]。

在 PID 控制系统中，往往根据实际情况，采用比例、积分、微分三种控制器相互组合的方式，选择合适的参数来保证系统响应的快速性、稳定性和准确性。

4.3 机器人基本运动控制设计

4.3.1 机器人直立控制设计

在保持直立平衡方面，四足机器人的四个足端与地面接触，形成四边形支撑；足式机器人拥有宽大的脚掌，形成脚掌与地面的面面接触。这两类机器人，由于其自身结构的优势，仅通过机械结构就可以保持直立平衡；而两足轮腿机器人是由左右两个驱动轮与地面之间的线面接触，单从机械结构上无法保持直立平衡，所以该类型机器人本身是一个非稳定系统。两足轮腿机器人保持平衡的条件是机器人质心与驱动轮中心线在同一个竖直平面内，当质心与驱动轮中心线不在竖直平面时，机器人会向质心偏离的方向加速倾倒。

由于本设计的两足轮腿机器人髋关节是由电机主动驱动，所以当髋关节角度不变时，机器人的整体姿态不会发生变化，此时可以将机器人简化为一阶倒立摆模型[32]，如图 4-2 所示，在简化模型的同时，将驱动轮的驱动力等效为外力F。

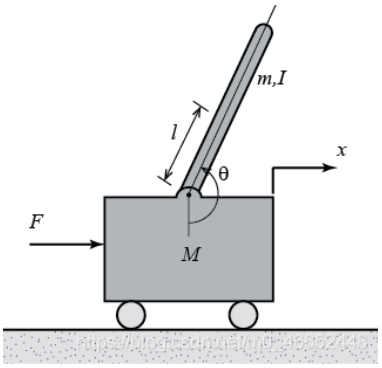


图 4-2 一阶倒立摆模型

其中，m为机器人腰身和腿部质量；M为机器人轮部质量；l为机器人质心到驱动轮中心线的距离；θ为机器人偏角；F为回复力。

将上述模型拆分为小车和杆件两个部分，如图 4-3 所示，二者水平和竖直方向上的相互作用力分别为N和P，按照牛顿-欧拉方法分别对两个分离体进行受力分析。

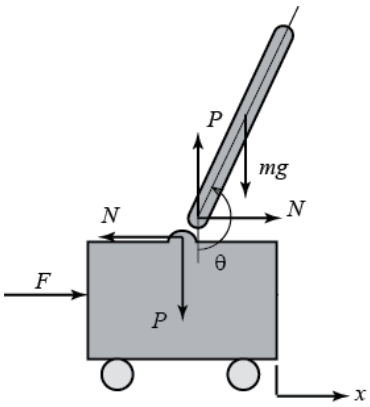


图 4-3 一阶倒立摆受力分析

由于外部回复力F的作用，小车会朝向F的方向移动，杆的倾角也会发生相应变化。假设小车的位移为x，杆与水平方向夹角为θ，则杆质心的水平和竖直位置分别表示为：

$$x = x_0 + l\sin\theta \tag{4-3}$$

$$y = l\cos\theta \tag{4-4}$$

对上述两式二次求导，得到杆质心在水平和竖直方向上的加速度为：

$$\ddot{x} = \ddot{x}_0 - l\sin\theta\dot{\theta}^2 + l\cos\theta\ddot{\theta} \tag{4-5}$$

$$\ddot{y} = -l\cos\theta\dot{\theta}^2 - l\sin\theta\ddot{\theta} \tag{4-6}$$

根据牛顿第二定律，可以得到杆在水平和竖直方向的受力为：

$$N = m\ddot{x} = m(\ddot{x}_0 - l\sin\theta\dot{\theta}^2 + l\cos\theta\ddot{\theta}) \tag{4-7}$$

$$mg - V = m\ddot{y} = m(-l\cos\theta\dot{\theta}^2 - l\sin\theta\ddot{\theta}) \tag{4-8}$$

又因为杆的倾角发生变化，则杆的转动方程为：

$$I\ddot{\theta} = Pl\sin\theta - Nl\cos\theta \tag{4-9}$$

对小车在水平方向受力分析，可得：

$$F - N = M\ddot{x} \tag{4-10}$$

联立上述(4-7)~(4-10)的 4 个方程，化简后可以得到倒立摆模型的动态方程为：

$$\ddot{x} = \frac{(I+ml^2)F + ml(I+ml^2)\sin\theta\ddot{\theta} - m^2l^2g\sin\theta\cos\theta}{(I+ml^2)(M+m) - m^2l^2\cos^2\theta} \tag{4-11}$$

$$\ddot{\theta}=\frac{ml^2cos\theta\cdot F+m^2l^2gsin\theta cos\theta\ddot{\theta}-(M+m)mlgsin\theta}{m^2l^2cos^2\theta-(M+m)(I+ml^2)} \tag{4-12}$$

由于实际工作时，倾角θ只会在很小的范围内摆动，可以基于这个条件对式(4-11)和(4-12)进行线性处理，近似认为$\dot{\theta}=0$，$sin\theta=\theta$，$cos\theta=1$，带入上述动态方程可以得到：

$$\ddot{x}=\frac{(I+ml^2)F-m^2l^2g\theta}{I(M+m)-Mml^2} \tag{4-13}$$

$$\ddot{\theta}=\frac{(M+m)mlg\theta-mlF}{I(M+m)+Mml^2} \tag{4-14}$$

将(4-13)、(4-14)写成标准的系统形式：

$$\dot{X}=\begin{bmatrix}x\\ \dot{x}\\ \theta\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & \frac{-m^2l^2g}{I(M+m)-Mml^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & \frac{(M+m)mlg}{I(M+m)+Mml^2} & 0\end{bmatrix}\times\begin{bmatrix}x\\ \dot{x}\\ \theta\\ \dot{\theta}\end{bmatrix}+\begin{bmatrix}0\\ \frac{(I+ml^2)}{I(M+m)-Mml^2}\\ 0\\ \frac{-ml}{I(M+m)+Mml^2}\end{bmatrix}F \tag{4-15}$$

设$a=\frac{-m^2l^2g}{I(M+m)-Mml^2}$、$b=\frac{(M+m)mlg}{I(M+m)+Mml^2}$、$c=\frac{(I+ml^2)}{I(M+m)-Mml^2}$、$d=\frac{-ml}{I(M+m)+Mml^2}$，则有：

$$C_0=\begin{bmatrix}0 & c & 0 & ad\\ c & 0 & ad & 0\\ 0 & d & 0 & bd\\ d & 0 & bd & 0\end{bmatrix} \tag{4-16}$$

C_0的行列式为：

$$\begin{vmatrix}0 & c & 0 & ad\\ c & 0 & ad & 0\\ 0 & d & 0 & bd\\ d & 0 & bd & 0\end{vmatrix}=b^2c^2d^2-2abcd^3+a^2a^d=(bcd-ad^2)^2\geq 0 \tag{4-17}$$

而$bcd\neq ad^2$，所以式(4-17)等号不成立，行列式不为 0。所以一阶倒立摆模型在外力F的作用下可以使杆件保持在直立平稳状态，系统可控。因此两足轮腿机器人系统也可以在驱动轮的调整下保持直立平衡，如图 4-4 所示。

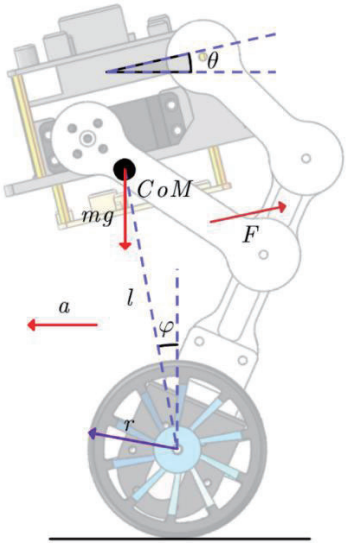


图 4-4 机器人平衡模型

为了简化控制过程，此处引入控制率：

$$a = K_p\varphi + K_d\dot{\varphi} \tag{4-18}$$

式中 a—驱动轮加速度；

φ—质心倾角；

K_p、K_d—比例系数。

因此，只需满足$mg\varphi < K_p\varphi + K_d\dot{\varphi}$就可以实现机器人的直立平衡控制。驱动轮的加速度$a$与电机输入 PWM 信号直接相关，机器人质心偏角$\varphi$和角速度$\dot{\varphi}$由 IMU 姿态传感器测量得到。根据上式可建立 PD 直立控制器进行反馈控制，其原理框图如图 4-5 所示。

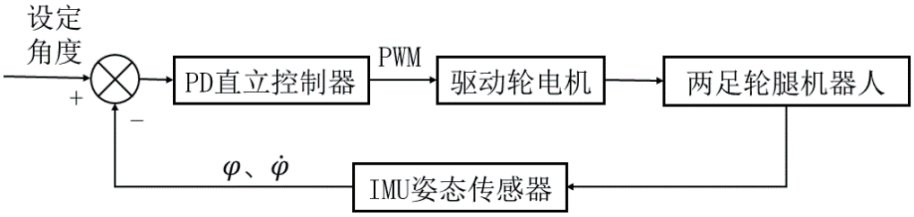


图 4-5 直立控制原理框图

4.3.2 机器人速度控制设计

在实现机器人直立平衡控制的基础上，还需要进一步设计速度控制器，对机器人

的直行速度进行控制。机器人的速度控制需要在机器人保持平衡的前提下进行，但速度控制不是将平衡控制器和速度控制器进行简单叠加。所以直接对驱动轮施加控制速度控制信号对直立平衡控制器产生影响，造成机器人系统失稳。这使得两足轮腿机器人的速度控制相比普通四轮移动机器人更加复杂。目前在双闭环控制系统中常用的是串级 PID 控制方法，将该方法应用到本设计速度控制器设计中，其系统原理框图如图 4-6 所示。

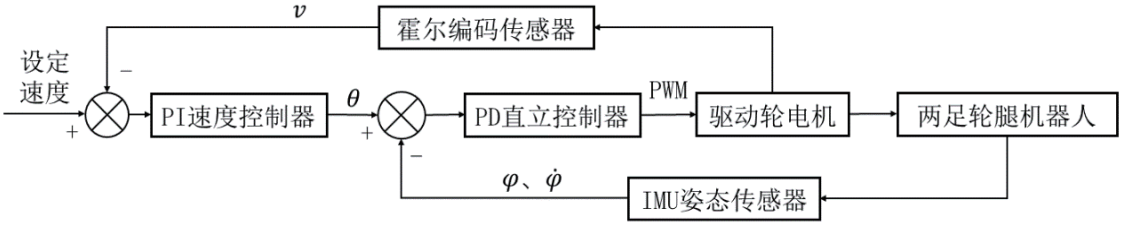


图 4-6 速度控制原理框图

为了保证机器人不同控制器的优先级，将直立控制器设置在内环，速度控制器设置为外环。当机器人移动时，首先需要速度控制器主动输出一个质心偏角θ，同时该质心偏角θ将作为直立控制器的输入，因此直立控制器的输入将不再是图 4-5 所示的设定角度。正由于直立控制器的输入是偏角θ，机器人要想维持偏角θ状态的同时保持直立而不倾倒，就必须偏角方向加速移动[33][34][35]。

根据上述分析可知，机器人的速度与机器人的偏角是正相关的，即机器人偏角越大，机器人的速度也越快。通过这样一个串级控制设计，间接达到机器人速度控制的目的。根据文献[36]，驱动轮在较高转速下，电机编码器可能会受到噪声干扰，而微分项往往会放大传感器噪声而造成系统振荡或失稳。因此在机器人的串级 PID 控制系统中，外环速度采用 PI 控制器来进行反馈控制。此处再次引入控制率：

$$\theta = K_1 \mathrm{e}(k) + K_2 \sum_{j=1}^{k} \mathrm{e}(j) \tag{4-19}$$

式中 θ—速度控制器的输出；

K_1、K_2—比例系数；

$e(k)$—系统速度偏差。

根据式(4-18)，当设定角度不在是 0° 而是速度控制器偏角θ时， PD 直立控制器也应做相应修正：

$$a = K_p(\varphi - \theta) + K_d\dot{\varphi} \tag{4-20}$$

将式(4-19)带入式(4-20)，可得：

$$a = K_p(\varphi - K_1\mathrm{e}(k) + K_2\sum_{j=1}^{k}\mathrm{e}(j)) + K_d\dot{\varphi} \tag{4-21}$$

化简得到：

$$a = K_p\varphi + K_d\dot{\varphi} - K_p\left[K_1\mathrm{e}(k) + K_2\sum_{j=1}^{k}\mathrm{e}(j)\right] \tag{4-22}$$

由上式可知，机器人的串级控制器可以看作是一个直立负反馈控制器和一个速度正反馈控制器的组合，根据这个结论将如图 4-6 所示的系统原理框图进行解耦，建立如图 4-7 所示的控制形式。

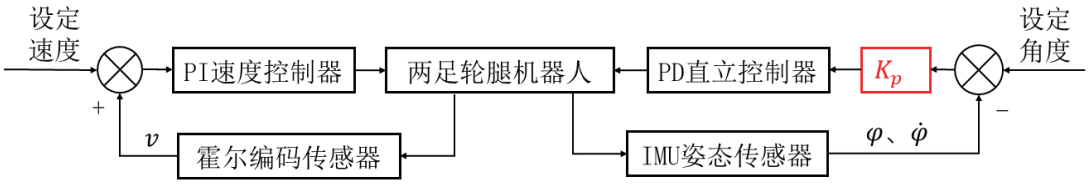


图 4-7 串级控制解耦框图

将机器人直立控制和速度控制解耦简化后可以看作两个独立的控制器，大大降低了计算机实现串级控制 PID 控制系统的难度，更加适用于实际工程。另外，在实际工程中，机器人的实际质心位置与理论位置通常会有微小误差，且 IMU 姿态传感器的安装通常有微小机械误差。这两种情况都会造成机器人产生机械偏差角度，如果此时系统中只有直立控制器，机器人会因为由于机械偏差角而持续向同一个方向加速，造成系统失稳；当引入速度控制器后，速度控制器的输出偏角θ会覆盖机器人的机械偏差角度。因此，速度控制器除了控制机器人的速度之外，还具有抵消机械误差的作用，进一步提高机器人控制的稳定性和鲁棒性。

4.3.3 机器人转向控制设计

本设计两足轮腿机器人的转向控制采用左右两轮差速的方式来实现，在机器人实现直立和速度控制的基础上，转向控制器的设计仅涉及左右两轮的相对速度控制。本

设计中，转向控制器的设计方法主要有如图 4-8 所示的 3 种：

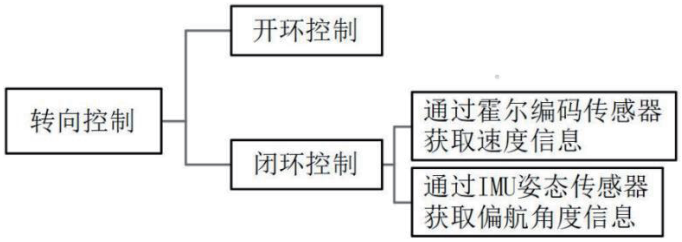


图 4-8 转向控制方法

在开环控制和闭环控制的选择中，虽然开环控制器的设计更加简单，但闭环控制有更高的控制精度和稳定性，所以选择闭环控制更加适合本设计的两足轮腿机器人。闭环控制有两种方式获取机器人的状态信息，若采用霍尔编码传感器检测机器人左右轮的速度，虽然可以实现闭环反馈控制，但当驱动轮出现打滑时，机器人设定的转动角度和实际转动角度会出现较大偏差；而通过 IMU 姿态传感器来获取机器人偏航角度信息，将偏航角作为反馈量可以有效的防止驱动轮打滑而造成的转向误差。

综上分析，本设计两足轮腿机器人采用通过 IMU 姿态传感器获取的偏航角为反馈量的方式来设计转向控制器。对系统转角偏差进行 P 比例运算，将结果作用于电机驱动器，实现对机器人转动角度的控制，该控制过程如图 4-9 所示。这种方式的有点是控制效果比较好，可以有效防止驱动轮打滑等的问题。同时，若将设定角度为 0 度，还可以解决机器人在直行过程中左右驱动轮不同步的问题，提高机器人直立和直行的实际运行效果。

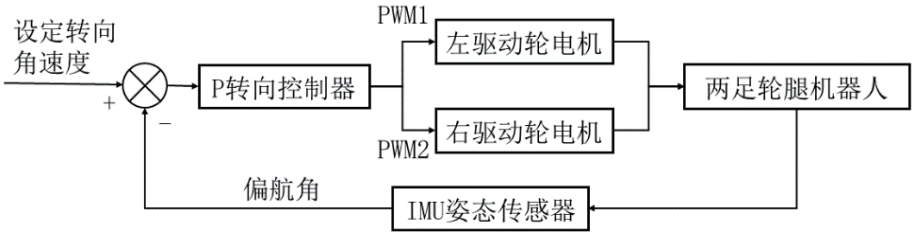


图 4-9 转向控制原理框图

以上分析了机器人的直立控制、速度控制和转向控制，虽然是三个不同的控制器，但它们之间不是相互独立的，都是通过传感器读取机器人运动状态来进行 PID 反馈控制，最终对驱动电机施加 PWM 信号，即三个控制器输出的执行单元都是驱动器和驱

动轮。因此可以将这三个控制器的输出组合起来，如图 4-10 所示，同时实现机器人的直立、速度和转向控制。

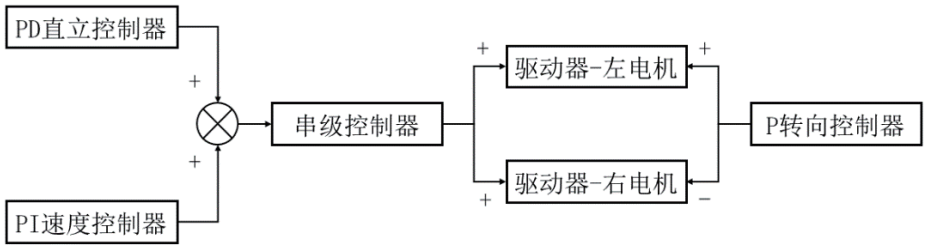


图 4-10 基本运动综合控制器

4.4 机器人姿态和地形自适应控制设计

实现基本运动控制后，需要在此基础上进一步展开对机器人的姿态和自适应控制的设计，提高机器人的功能和自适应能力。该部分主要包括机器人的高度调节控制设计、跳跃控制设计以及地形自适应控制设计。

4.4.1 机器人高度调节

在第 2 章中完成了 4 自由度轮腿融接型两足轮腿机器人结构方案设计和在第 3 章完成了机器人机电系统设计后，确定了机器人平面四杆的腿部结构。其驱动轮在平面中的位置与髋关节角度呈非线性关系，但非线性系统往往有死区、饱和等现象，给系统控制带来困难。为了避免这些现象，本设计忽略了较小影响的非线性因素，将机器人的髋关节角度和驱动轮位置建立线性关系，同时也是将髋关节角度和机器人高度建立了线性关系。

由于在实际中髋关节角度和机器人高度之间是非线性关系，将其线性化则会导致系统一定程度的失真。为了降低失真带来的影响，需要将机器人几个的关键高度位置作为系统的稳态点，保证机器人在这几个关键高度位置不会有误差。综合实际需求，本设计将机器人如图 4-11 所示的最低位、中间位、最高位三个高度位姿作为稳态点，对机器人的高度调节进行线性化处理。

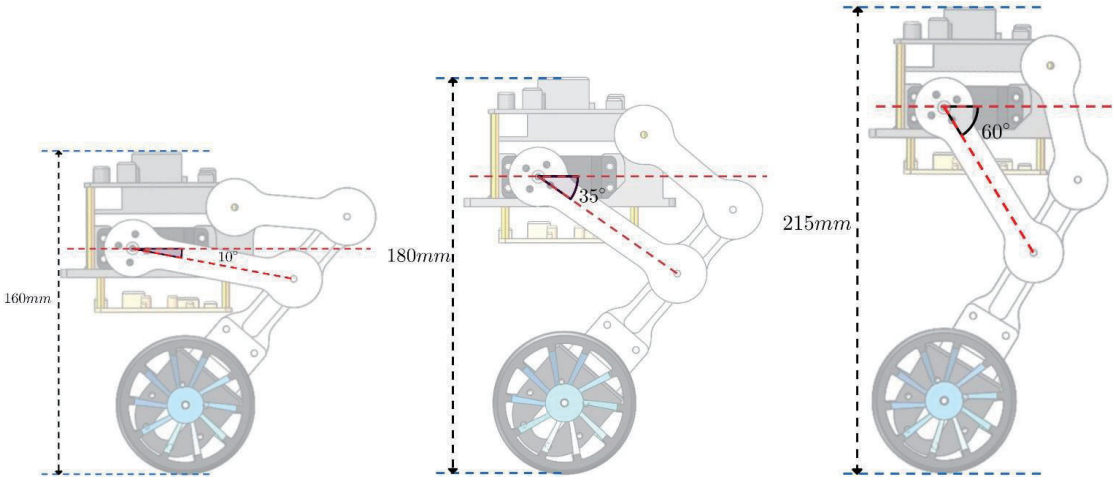


图 4-11 机器人的三个高度位姿

使用小偏差方法对系统线性化的过程分为三个步骤，设这一高度调节系统的非线性函数为：

$$y = f(x) \tag{4-23}$$

（1）假设x，y在稳态点(x_0, y_0)附近增量变化，即：

$$x = x_0 + \Delta x \tag{4-24}$$

$$y = y_0 + \Delta y \tag{4-25}$$

（2）近似处理，在稳态点(x_0, y_0)处以曲线的切线代替曲线，且对平衡点的变化求导，可以得到近似式：

$$\Delta y = \frac{\mathrm{d}f(x)}{\mathrm{d}x}\Big|_{x=x_0} \cdot \Delta x \tag{4-26}$$

（3）将非线性函数$f(x)$在x_0处的泰勒级数的一阶增量项并化简可以得到：

$$y - y_0 = \frac{\mathrm{d}y}{\mathrm{d}x}\Big|_{x=x_0} \cdot (x - x_0) \tag{4-27}$$

根据上述分析，设髋关节角度为变量x，单位是弧度；机器人高度为变量y，单位是毫米。带入三个稳态点，可以将机器人髋关节角度和机器人高度线性化为如下关系式：

$$\begin{cases} y = \frac{4}{5}x + 152, & 10 \le x \le 35 \\ y = \frac{7}{5}x + 131, & 35 < x \le 60 \end{cases} \tag{4-28}$$

为了更直观展示机器人髋关节角度与高度线性化关系，将上述关系式绘制如图 4-12 所示的函数图像，其中横坐标是机器人髋关节角度，纵坐标是机器人高度。

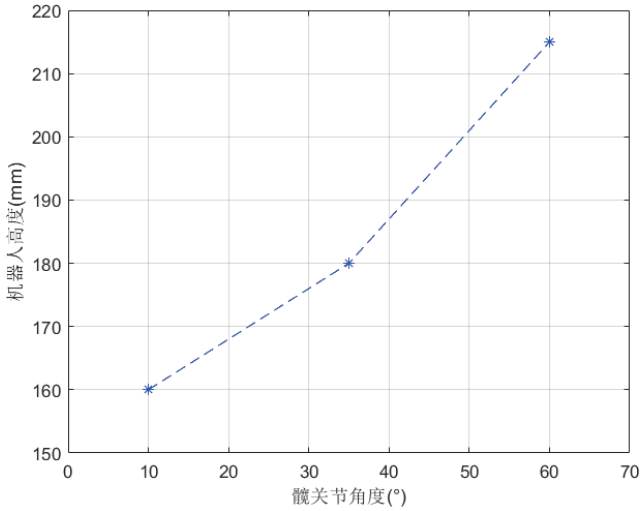


图 4-12 髋关节输出线性化

在高度调节中，机器人的左右腿是同步的，即左右髋关节角度相同。因此在实际控制中，只需对左右髋关节发送相同的控制信号。根据第 3 章机电系统设计中介绍的髋关节电机工作原理，本文设计了机器人的高度调节控制器，其原理框图如图 4-13 所示。其中设定角度由操作人员发送给机器人，通过 P 比例角度控制器的调整，最终将机器人髋关节角度稳定在设定角度附近，实现机器人的高度调节。

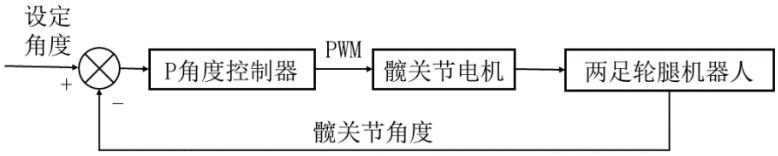


图 4-13 高度控制原理框图

4.4.2 机器人跳跃控制设计

前文介绍了机器人的基本运动控制设计，可以完成在平坦地形直行、转弯等，但当机器人在行驶途中遇到高度大于驱动轮半径的障碍物时，用常规的直行方式很难通过，这就需要对机器人进行跳跃控制设计，使其能够适应更加复杂的地形。本文设计的机器人跳跃控制过程包括如图 4-13 所示的五个状态、四个过程。

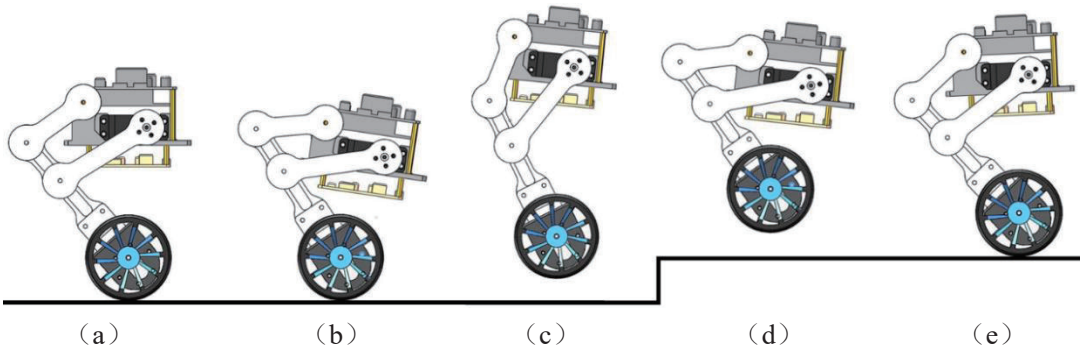


图 4-14 机器人跳跃过程

a-b）**缩腿：**在执行跳跃任务之前，机器人保持直立状态。当接收到跳跃令后，机器人进入跳跃控制中，机器人的髋关节角度减小，腿部收缩，重心下移，为跳跃蓄力。

b-c）**蹬腿：**使用髋关节比例控制器来实现机器人腿部的快速伸展，完成伸展后，由于机器人有向上的速度，机器人整体会向上跳跃起一定高度，直到机器人向上的速度为 0 时达到最高位置。

c-d）**收腿：**当机器人达到最高位置时，通过再次收腿这一动作可以增加驱动轮与地面的距离，进一步提高机器人的跳跃越障能力。

d-e）**调整：**机器人跳跃到空中后，会由于重力的影响重新回到地面，经过一段时间的调整后，机器人在次保持直立平衡状态。

在机器人跳跃过程中，总共需要 4 个如图 4-13 所示的髋关节角度控制器，它们的执行时机由控制指令和机器人状态所决定。

4.4.3 机器人地形自适应控制设计

当机器人在平地运动时，左右驱动轮基本处于同一水平线上，机器人只需要通过驱动轮前后调整就可以保持直立；但是当机器人在崎岖地形环境运动时，可能会出现左右驱动轮不在同一水平线的情况而引起机器人在横滚方向上的倾斜或侧翻。因此，两足轮腿机器人想要在是复杂多变的现实环境中执行任务，仅拥有基本的保持直立、直行等运动能力是远远不够的，还必须拥有感知复杂地形并主动调整自身姿态以适应复杂地形的能力。

本设计的两足轮腿机器人由于其本身的结构特点，横滚方向上有两个驱动轮支撑，本身具有稳定性，因此，横滚方向的控制器设计难度要低于俯仰方向。根据这一实际情况，本设计采用如图 4-15 所示的 PD 横滚角度控制器来实现机器人对复杂地形的自适应。

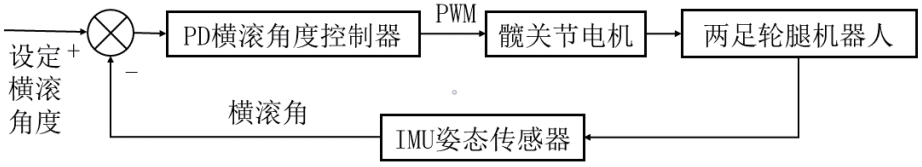


图 4-15 地形自适应控制原理框图

4.5 控制系统软件设计

在第 3 章中完成了机器人的机电系统设计和在第 4 章中完成了机器人的运动控制方法设计后，需要进一步在主控板上编写软件程序来处理传感器信息、驱动硬件等，从而综合调度机器人机电系统各个模块，实现机器人的运动控制。该部分将首先介绍机器人的软件总体结构、主程序和中断子程序，然后分别介绍机器人运行中，主控芯片执行的主要任务，包括传感器信息处理和系统通信。

4.5.1 控制系统软件总体结构

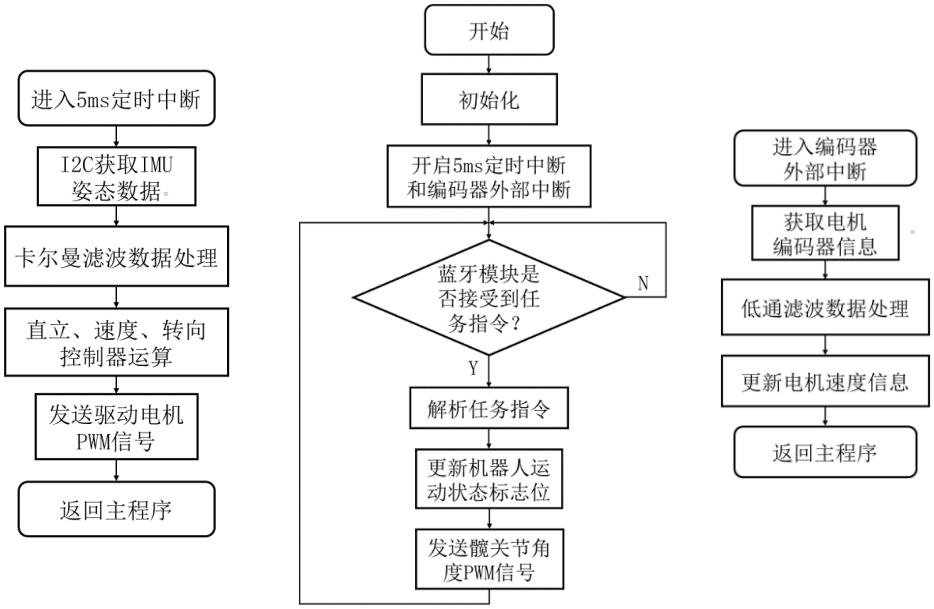
为了保证机器人的正常运行，控制系统软件程序需要处理众多信息，包括 IMU 姿态传感器测量的俯仰角信息、横滚角信息，蓝牙数据通信等等。为了方便程序设计，结合机器人的机电系统硬件设计和运动控制的需求，将机器人运行时主控板需要处理的任务汇总，这些任务名称及处理方式如表 4-1 所示。

表 4-1 控制系统任务处理表

序号	任务名称	处理方式
1	直立 PD 控制器	数字 PID 运算
2	速度 PI 控制器	数字 PID 运算

序号	任务名称	处理方式
3	转向 P 控制器	数字 PID 运算
4	髋关节电机 P 控制器	数字 PID 运算
5	IMU 姿态信息读取	I2C 接收姿态信息
6	驱动电机编码器信息读取	外部中断存储编码器信息
7	驱动器控制	发送开关信号和 PWM 信号
8	远程遥控和监控	蓝牙串口通信

在机器人嵌入式设计中，系统接通电源立即进入主程序中，并且会循环执行主程序，直到系统电源关闭。对于本设计任务多、关系复杂，且不同任务有不同优先级的程序设计需求，比较合理的是在控制主程序中设计多个中断子程序，分别执行任务。同时，通过这样的设计方式还可以将程序模块化，提高程序开发效率。本设计两足轮腿机器人的控制系统软件总体结构主要为三个部分：主程序、5ms 定时中断子程序和编码器外部中断子程序，其流程图如图 4-16 所示。其中 5ms 定时中断子程序的优先级大于编码器外部中断子程序。



a）5ms 定时中断子程序　　b）控制系统主程序　　c）外部中断子程序

图 4-16 控制系统流程图

主程序的主要任务是通过蓝牙接收来自操作人员的控制指令。当主控板接收到控制指令后立即对该指令进行解析，根据指令内容更新机器人的运动状态标志位。另外，

髋关节角度控制指令也在主程序中发出，髋关节电机根据角度指令进行腿部伸缩，实现机器人的姿态调整。

5ms 定时中断子程序是机器人保持平衡的关键，每次进入中断子程序后都会获取一次机器人的姿态信息，这些信息通常会因为零漂和温漂现象导致失真，因此需要对其中的关键数据进行卡尔曼滤波处理。再对处理后的姿态信息进行控制器运算，并根据运算结果向驱动电机发送 PWM 信号以动态调整机器人姿态，保证机器人的稳定性。

在机器人的正常工作中，驱动电机的运行状态也是重要参数之一。但因为驱动轮的转速较快，必须放在单独的外部中断子程序中，避免数据丢失。该子程序的进入标志是编码器脉冲信号的上升沿。

根据上述分析，本设计的两足轮腿机器人主要任务分为两个方面：传感器信号处理和机器人通信。以下对这两个方面的软件程序设计分别进行介绍。

4.5.2 传感器信息处理

传感器信息处理部分主要包括 IMU 姿态传感器测量的机器人姿态信息和霍尔编码器测量的驱动电机旋转信息两个方面。

IMU 姿态数据读取与处理：MPU6050 芯片内置三轴加速度计三轴陀螺仪，由 I2C 总线与主机相连接，并通过写寄存器的方式进行硬件配置和数据传输，相关的配置寄存器主要有以下几个。

• 电源管理寄存器（0×6B）：该寄存器用于配置电源模式和时钟源，同时附带一个复位和一个温度传感器开关。

• 陀螺仪配置寄存器（0×1B）：该寄存器用来出发陀螺仪自检和配置陀螺仪的量程范围。

• 加速度计配置寄存器（0×1B）：该寄存器的功能和陀螺仪配置寄存器类似，用来加速度计自检和量程范围配置。

• 陀螺仪采样率分频寄存器（0×19）：该寄存器用来产生 MPU6050 的采样率，是寄存器输出、FIFO 输出、DMP 采样和运动检测的基础。

• 配置寄存器（0×1A）：该寄存器为陀螺仪和加速度计配置外部帧同步引脚采样和数字低通滤波器。

•加速度计数据输出寄存器（0×3B~0×40）：这 6 个寄存器存储分别存储了 X/Y/Z 轴三个轴的加速度值，每个加速度值有 16 位且高字节位高 8 位，低字节位低 8 位。

• 陀螺仪数据输出寄存器（0×43~0×48）：这 6 个寄存器存储分别存储了 X/Y/Z 轴三个轴的角速度值，其数据的存储方式和加速度计数据输出存储器相同。

根据以上寄存器的功能对 MPU6050 模块进行参数配置，并将本设计中所需要的机器人角速度、角加速度数据进行封装，设计相应的通信协议。由于 MPU6050 模块内置的三轴加速度计和三轴陀螺仪数据均分别由两个 16 位的二进制数存储，其表示范围为-32767~32768，而实际机器人在工作中的最大角度范围为-180°~180°。因此需要在 16 位的二进制数和机器人实际角度之间建立线性映射关系。另外，通过如图 4-17 所示的 8 字节的串口通信协议来实现数据在 MPU6050 模块和主控板之间的传输。

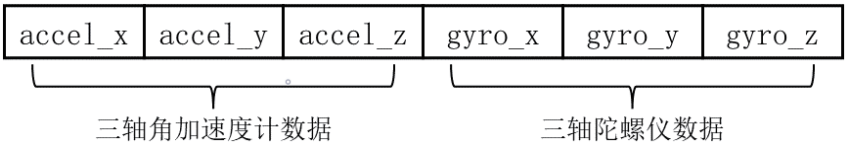


图 4-17 IMU 数据通信协议

当主控板接收到 IMU 姿态信息后，需要对这些数据信息进行滤波处理，降低噪声干扰，比较常用且有效的是卡尔曼滤波器。卡尔曼滤波根据线性系统的状态方程，观测系统的输入输出来进行滚动优化，最终得到所需要的机器人状态信息。实际测试中，卡尔曼滤波器稳定性高，使用效果良好。

霍尔编码器信息读取与处理：驱动轮电机 GB37-520 外置的 AB 两项 11 线编码盘，通过读取 A 相和 B 相的布尔值、计数和频率来得到驱动轮转动的方向、速度信息，其原理如图 4-18 所示。

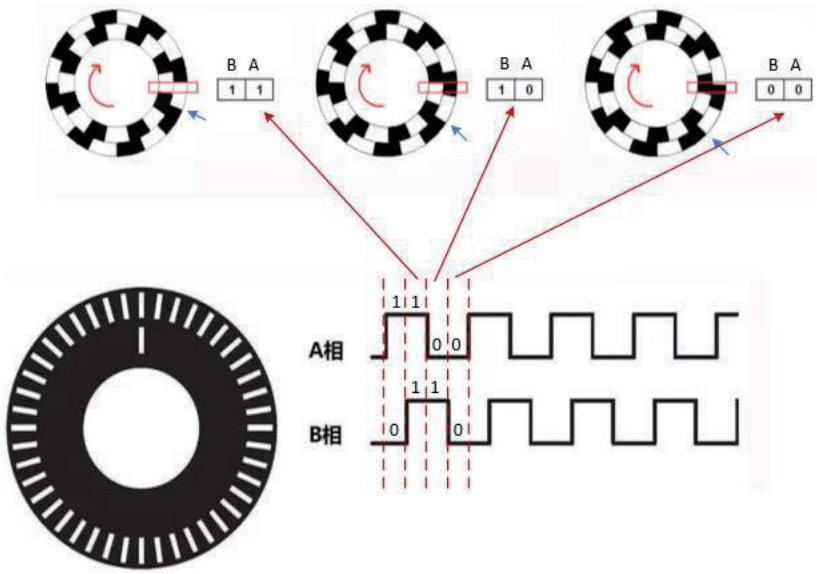


图 4-18 霍尔编码器原理图

正反转区分方面，结合上图所示编码方式，本设计以 A 相脉冲信号上升沿为标志位，读取此时 B 相的值，若为高电平则说明驱动电机正转，否则反转。转速计算方面，霍尔编码测速的方法有很多种，由于驱动轮的转速较快，比较适合的是周期测速法，其原理如图 4-19 所示。

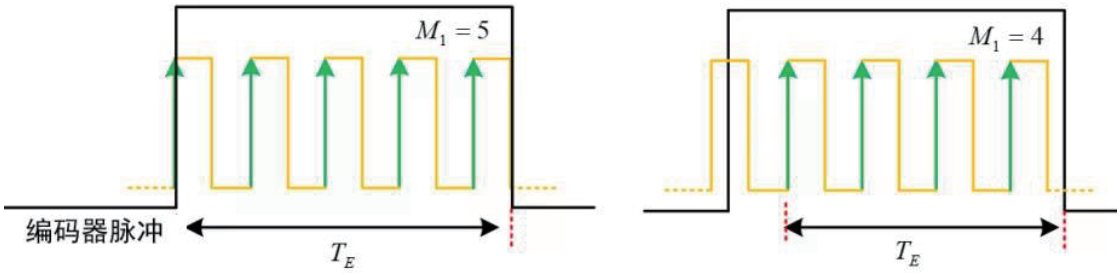


图 4-19 周期测速法

根据图 4-19 展示的周期测试法原理可知，驱动轮转速仅通过单相编码盘即可计算得到，满足如下关系式：

$$n = \frac{1}{T_E \cdot C} \tag{4-29}$$

式中 n—驱动轮电机转速；

T_E—1 秒内的脉冲个数；

C—编码器单圈总脉冲数。

通常驱动轮霍尔编码器容易受到高频噪声干扰，因此设计了一个简易的一阶低通滤波器来减小噪声干扰的影响。

4.5.3 机器人通信

IMU 姿态传感器 I2C 通信：I2C 是一种两线式的串行通信方式，通过一条串行数据线 SDA 和一条串行时钟线 SCL 来连接控制器和外围设备，实现数据的传输，I2C 总线时序如图 4-20 所示。

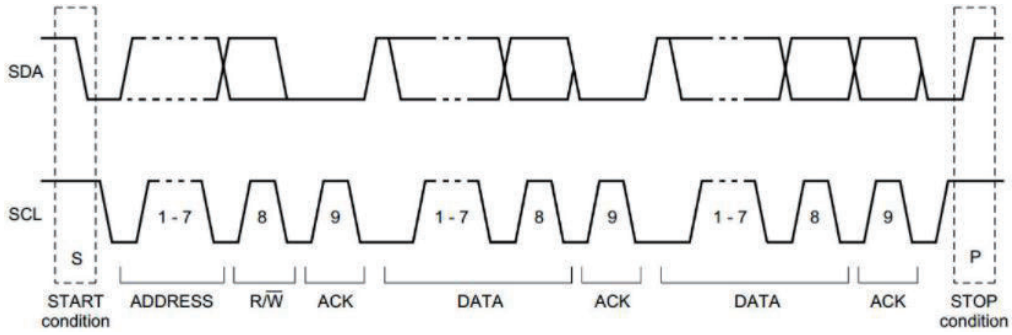


图 4-20 I2C 总线时序图

从 I2C 总线时序图中可知，当 SCL 为高电平、SDA 出现下降沿时表示起始位；当 SCL 为高电平、SDA 出现上升沿时表示停止位；而起始位和停止位之间的是通信位。其中通信位包括 ADDRESS、R/$\overline{W}$、ACK 和 DATA 四个部分。

• ADDRESS：地址帧，存储从机设备 7 位地址；

• R/$\overline{W}$：读/写位，低电平时主设备向从设备写数据，高电平时向从机读数据；

• ACK：确认位，通信位内的每一帧都有一个数据确认位，根据 ACK 位的状态来判断设备是否接收到了地址帧和数据帧；

• DATA：数据位，主从设备之间需要传输的数据。

根据 I2C 的通信时序，主设备在通信周期首先发送 7 位从设备地址和 1 个读写位到总线，从机匹配到地址和读写信息后返回一个确认位，主机再根据确认位给从机发送相应的数据。这样的通信方式具有接口线少、控制方式简单、通信效率高等优点，很适合应用在本设计的两足轮腿机器人中。

蓝牙无线通信：在第 3 章机器人机电系统设计中选择了 HC-06 模块作为本设计的蓝牙通信模块，通过 TXD 和 RXD 引脚实现与主控板的物理连接和数据传输，具有低功耗、高效率的特点。由于在机器人的实际运行中，控制信息较多，蓝牙传输数据量较大，因此本设计将这些数据信息进行打包，并设计了相应的通信协议，如图 4-21 所示。

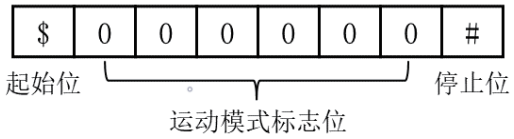


图 4-21 蓝牙通信协议

在这个 8 位通信协议数据包中，包括 1 个字符型的起始位、1 个字符型的停止位和 6 个整型的运动模式标志位。操作人员通过设置运动模式标志位内容来改变机器人的运动状态，从而实现机器人的远程控制。

4.6 本章小结

本章首先介绍了 PID 控制的原理以及比例项、积分项、微分项的功能和作用。再根据 PID 理论完成了机器人的基本运动控制设计，包括机器人直立控制、速度控制和转向控制。但由于机器人的实际工作环境复杂多变，仅有基本运动控制器是不够的，因此在 4.4 节中介绍了机器人的姿态控制和地形自适应控制进行设计，包括高度调节、跳跃控制和横滚方向的地形自适应控制。最后对控制系统的软件部分进行了详细介绍，包括控制系统的总体结构、传感器信息处理以及机器人通信三个部分。

第5章 实验研究

5.1 引言

前文完整介绍了两足轮腿机器人的结构设计方案与运动学建模，再根据运动需求进行了机电系统硬件设计和控制系统软件设计，完成了机器人样机的制作。本章将利用样机进行一系列运动实验，包括直立平衡实验、直行实验、高度调节实验等。通过实际的实验测试及数据分析来验证两足轮腿机器人系统设计和控制方法的合理性。

5.2 机器人基本运动控制实验

5.2.1 机器人直立平衡实验

直立平衡是两足轮腿机器人完成复杂任务的基础，因此本文首先对直立平衡控制器的设计进行验证。在本实验中，打开机器人机电系统电源开关，让机器人 0~10s 在内保持直立动态平衡状态，读取机器人的俯仰角度、角速度信息和电机驱动器接收到的 PWM 信号。图 5-1 是 0~10 秒内机器人直立动态平衡过程，图 5-2 记录了实验过程中机器人姿态信息和 PWM 调整信号。

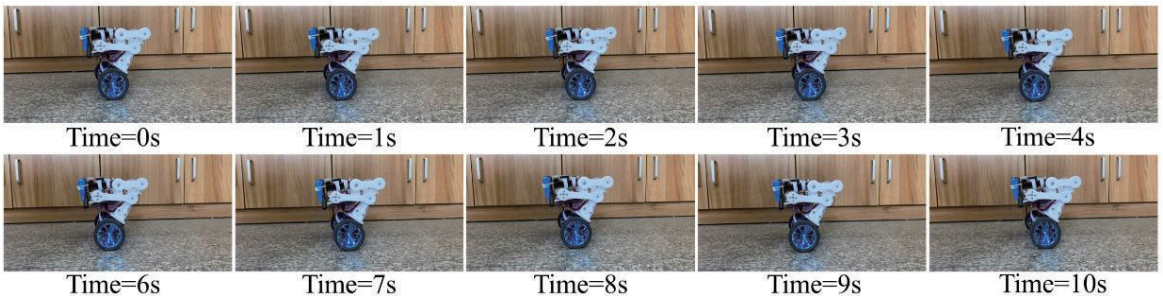


图 5-1 机器人直立平衡实验

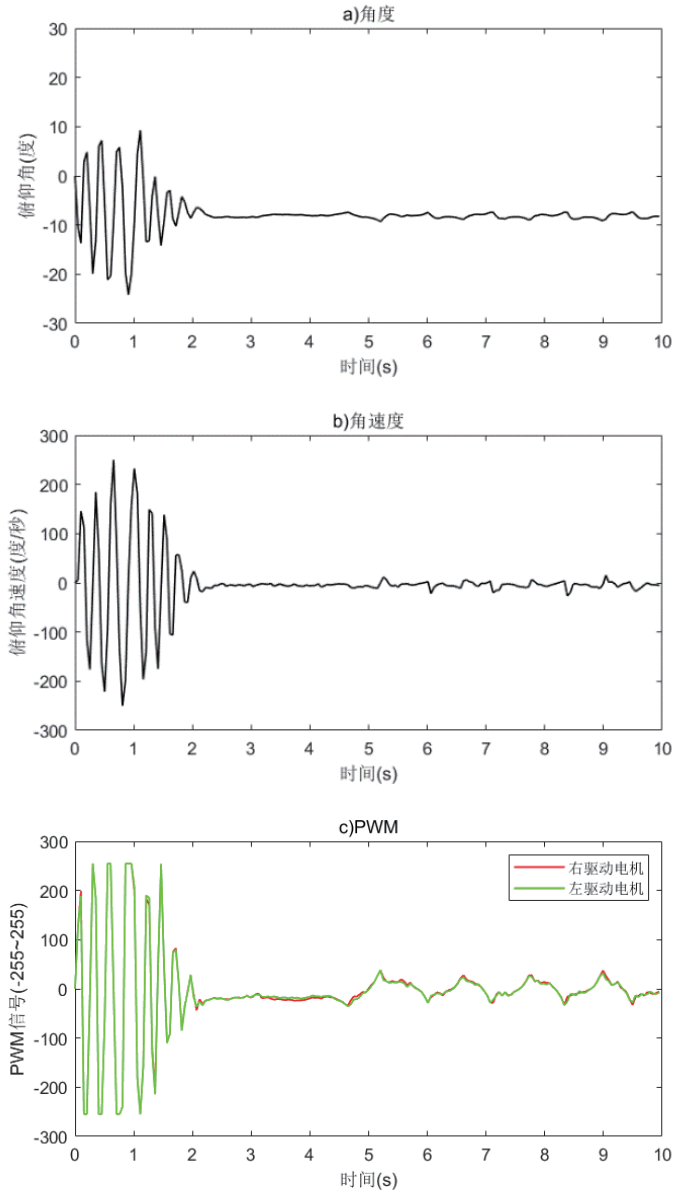


图 5-2 机器人直立平衡实验姿态变化

从图 5-2 中可以看出，机器人接通电源后大约需要 2s 到达动态平衡状态，由于驱动电机存在转动间隙，机器人无法达到完全静止。约第 2s 后机器人俯仰角速度幅值较小，保持动态调整，直立平衡效果良好。

5.2.2 机器人直行实验

机器人直行实验整个过程持续 10s，包括前进阶段 1~4s 和后退阶段 5~8s，其余时间段是机器人的缓冲过程。该实验机器人的运动状态和相关数据分别如图 5-3 和图 5-4 所示。

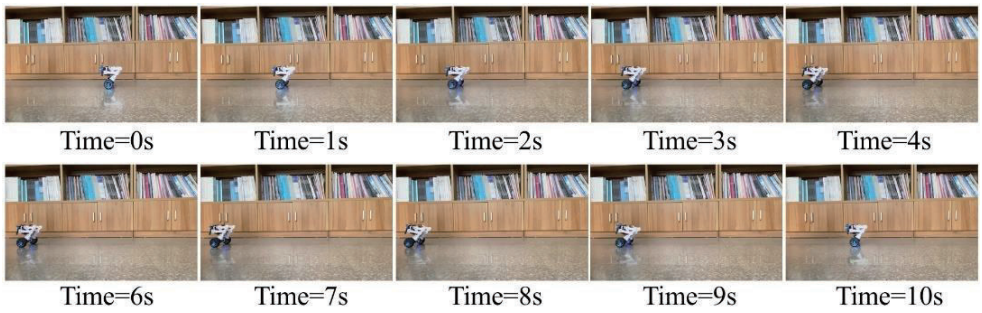


图 5-3 机器人直行实验

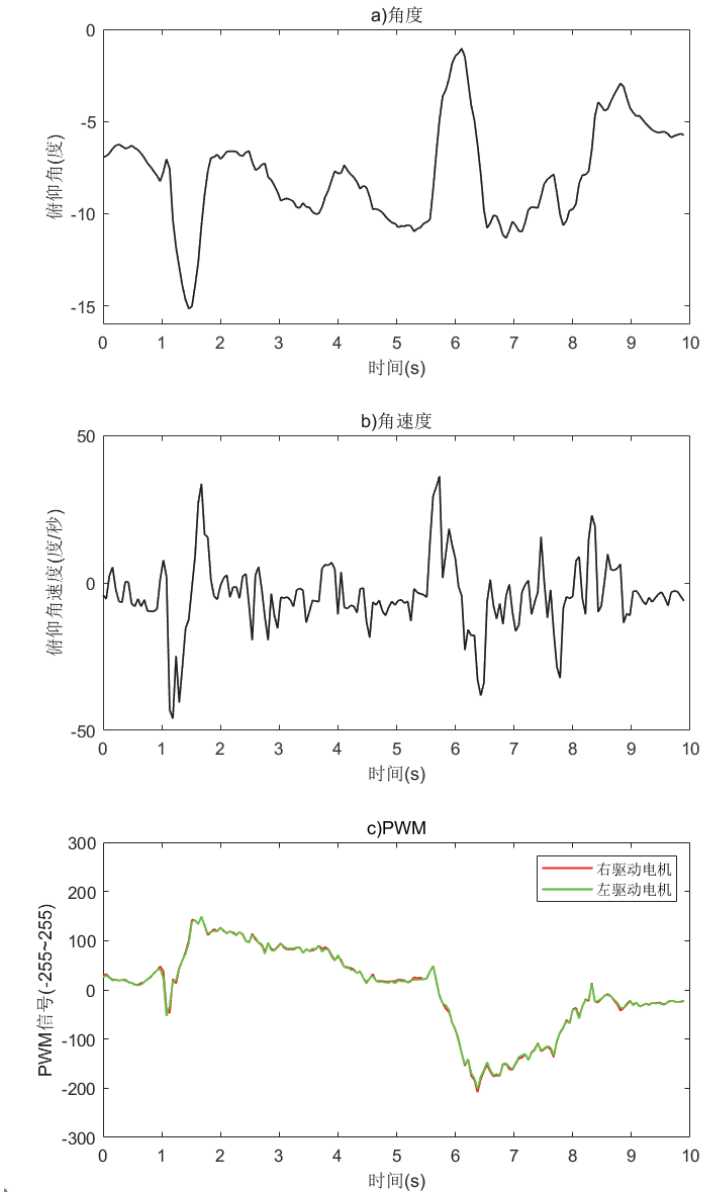


图 5-4 机器人直行实验姿态变化

从图 5-4c 中机器人前进阶段 1~4s 内可以看出，机器人要想前行，首先驱动轮向后加速，使得机器人产生一个向前的倾角而获得向前倾倒的趋势，此时机器人的直立控制器为了平衡这种倾倒趋势，会对驱动轮施加相应的 PWM 信号，使得机器人向前

加速，整个过程实现了机器人的前进控制，5~8s 内的后退控制同理。此处从实验数据分析也印证了第 4 章中速度控制器设计的正确性。

5.2.3 机器人转向实验

机器人的转向实验时间共 10s，其中包括 3~4s 内的左转控制和 6~7s 内的右转控制，其余时间段是机器人的缓冲过程。通过对机器人左驱动轮施加负的 PWM 信号，对右驱动电机施加正的 PWM 信号实现机器人的左转向；对机器人左驱动轮施加正的 PWM 信号，对右驱动电机施加负的 PWM 信号实现机器人的右转向。整个过程中机器人的运行状态和相关参数分别如图 5-5 和图 5-6 所示。

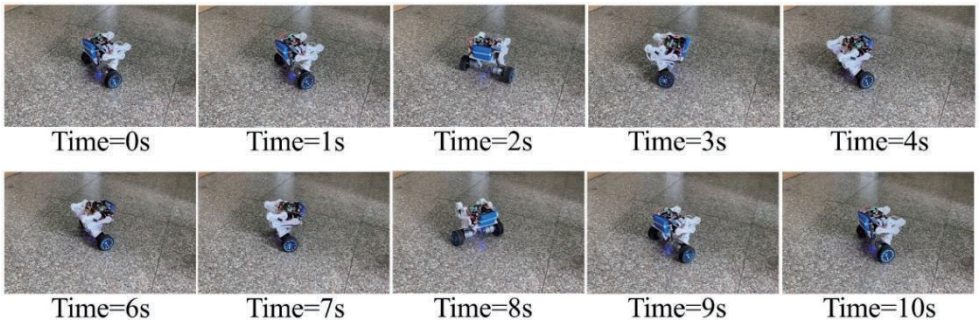


图 5-5 机器人转向实验

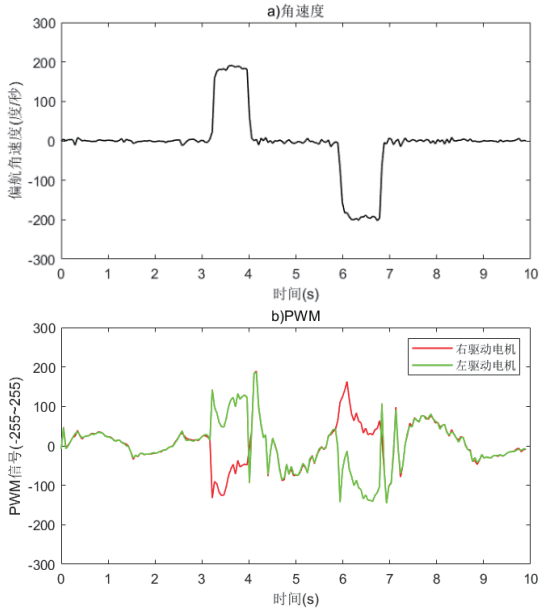


图 5-6 机器人转向实验姿态变化

从图 5-5a 可以看出，在实际运行中，机器人的偏航角速度在转向和直立过程中都比较稳定，左右转动的角速度在 200 度/秒左右。

5.3 机器人姿态和地形自适应控制实验

5.3.1 机器人高度调节实验

在第 4 章中设计的机器人高度控制器，将机器人的高度分为 160mm、180mm 和 215mm 三个高度，分别对应髋关节电机的 10°、35°和 60°。本文根据机器人的三个高度设计了高度调节实验，整个实验时间共 10s，包括髋关节电机 10°到 35°、35°到 60°两个升高过程和 60°到 35°、35°到 10°两个降低过程。由于髋关节电机角度变化通过延时函数来完成，在高度调整过程中，左右髋关节电机变化有先后循序，因此在高度调节过程中有机器人横滚方向上的角度变化。整个实验中机器人状态和相关参数分别如图 5-7 和图 5-8 所示。

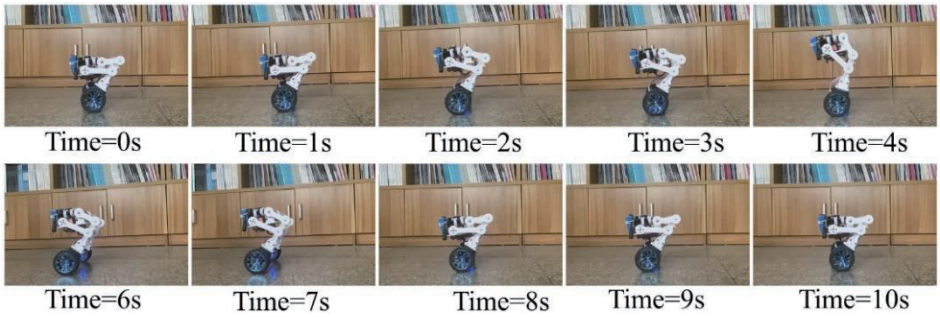


图 5-7 机器人高度调节实验

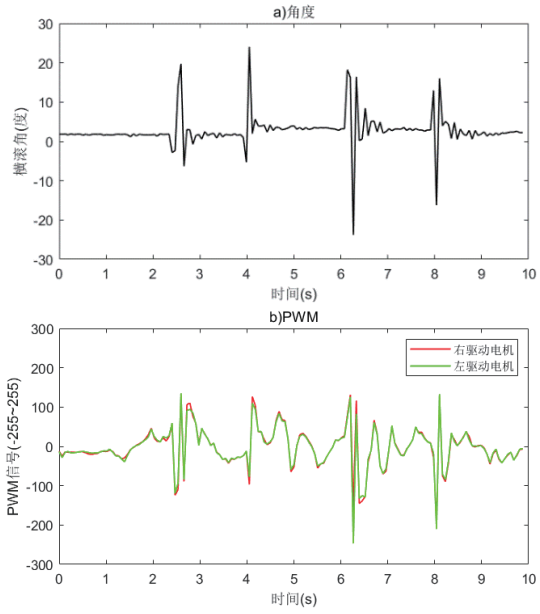


图 5-8 机器人高度调节实验姿态变化

由于机器人在高度调节的同时，伴随着重心的上下变化和机械倾角大小的调整，这两者都是机器人稳定性的重要因素。因此，从图 5-8 b 中可以看出，在高度调节实验中，左右驱动轮的 PWM 信号在不断变化，每一次改变高度都伴随有一段缓冲过程。

5.3.2 机器人单腿越障实验

在机器人单腿越障实验中，选取厚度为 15mm板材作为障碍物，障碍物与机器人的位置和大小关系如图 5-9 所示。

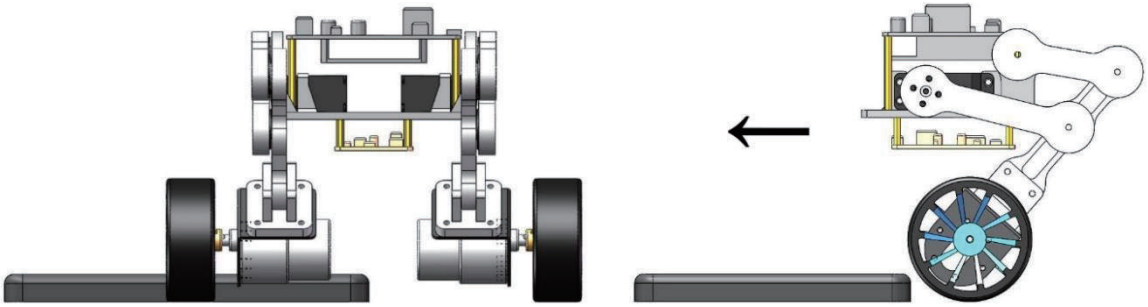


图 5-9 机器人与障碍物件的位置和大小关系

该实验的时间共有 5s，其中机器人左驱动轮在约 3.5s 时越上障碍物，约 4.5s 时机器人成功通过障碍物，回到平衡状态。在单腿越障实验中，机器人身体的侧向倾角将是判断控制器好坏的关键参数，因此把机器人横滚角作为测试参数进行读取，整个实验过程中，机器人的运行状态和横滚角数据分别如图 5-10 和图 5-11 所示。

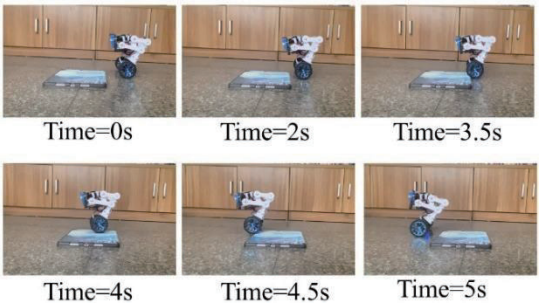


图 5-10 机器人单腿越障实验

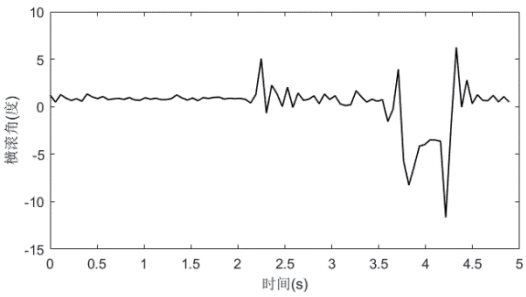


图 5-11 机器人单腿越障实验姿态变化

从实验结果中可以看出，机器人在单腿越障时，不论是进入障碍还是离开障碍，首先都会在横滚方向上出现较大的角度变化，但随后由于控制器的调整作用，机器人会快速回到平衡状态，展现了机器人较好的地形适应能力。

5.3.3 机器人抗冲击实验

在抗冲击实验中，本文利用重量为50g的塑料球和长度为600mm的细绳，通过塑料球的单摆运动对机器人产生冲击力，如图5-12所示。

在不考虑空气阻力的情况下，由能量守恒定律得：

$$mgl = \frac{1}{2}mv^2 \tag{5-1}$$

式中 m——塑料球的质量；

l——细绳长度；

v——塑料球在单摆运动最低点时的速度。

假设塑料球与机器人之间的冲力为F，作用时间为t，且发生冲击后塑料球的速度为0。根据冲量定理：

$$Ft = mv \tag{5-2}$$

将式1带入式2，化简可得：

$$F = m \cdot \sqrt{2gl} \tag{5-3}$$

将$t = 0.01s$、$m = 50g$、$l = 600mm$、$g = 9.8m/s^2$带入上式可得到，塑料球通过单摆而产生的冲力$F \approx 17N$。

本实验在机器人保持直立平衡时进行，通过该实验可以观察机器人平衡控制器的抗冲击和抗干扰能力。整个实验过程共持续 10s，塑料球对机器人的冲击在约第 3.8s时产生，0~10s内机器人的相关参数如图5-13所示。

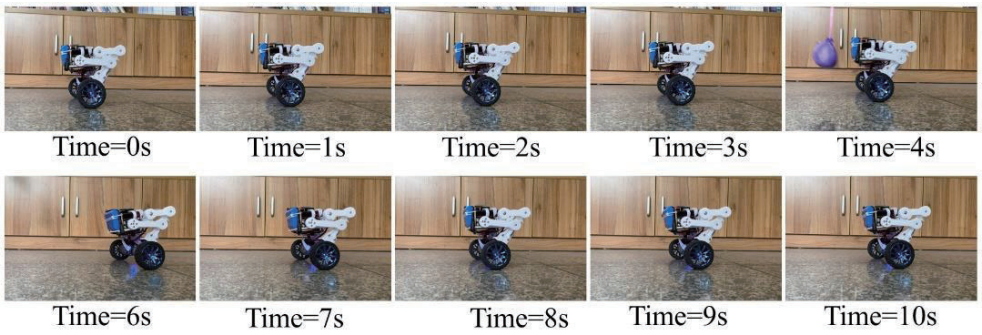


图 5-12 机器人抗冲击实验

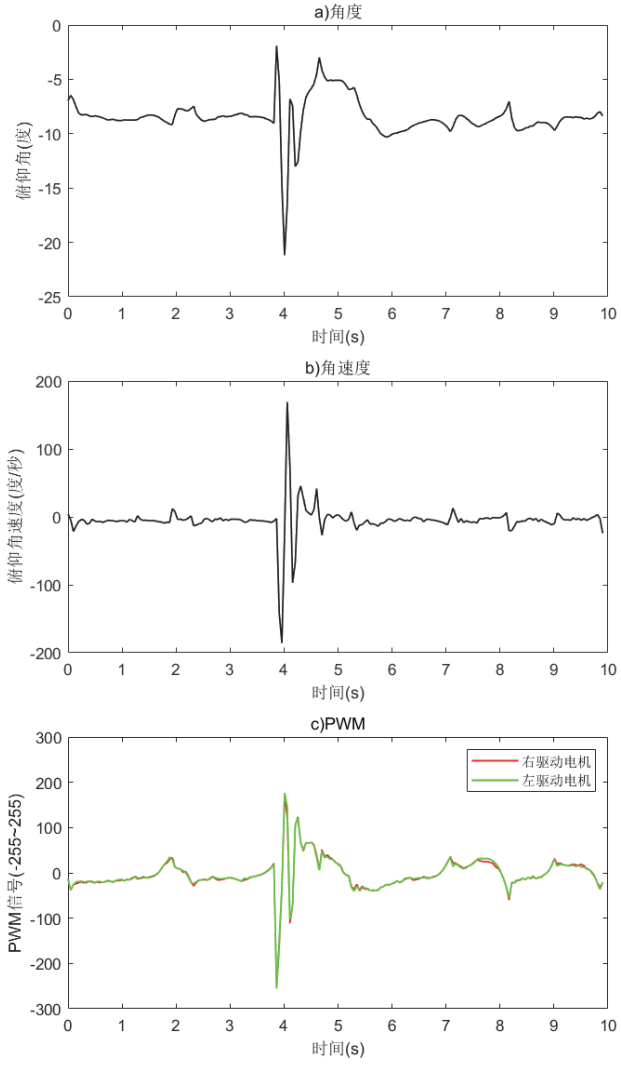


图 5-13 机器人抗冲击实验姿态变化

根据前文分析可知，塑料球将会对机器人产生约 17*N* 的冲击力，在该冲击力的作用瞬间，机器人平衡状态被打破，机器人的俯仰角和俯仰角速度发生较大变化，平衡控制器捕捉到这一状态信息后对驱动轮施加相应的 PWM 信号，使机器人做出相应调整来应对外部冲击力。从图中可以看到，大约 1s 后，机器人完成了自身调整而再次回到平衡状态，说明机器人的平衡控制器拥有较好的稳定性能。

5.3.4 机器人复杂地形通行实验

本实验选择了地面凹凸不平的林地作为实验环境，实验过程共 20s。实验中，读取机器人的俯仰角、横滚角、偏航角速度和驱动轮 PWM 信号数据，综合分析机器人的通行能力及稳定性。本次实验和相关数据分别如图 5-14 和图 5-15 所示。

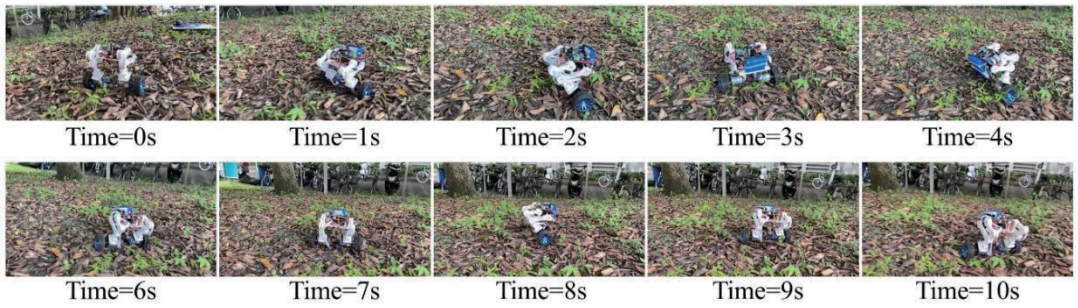
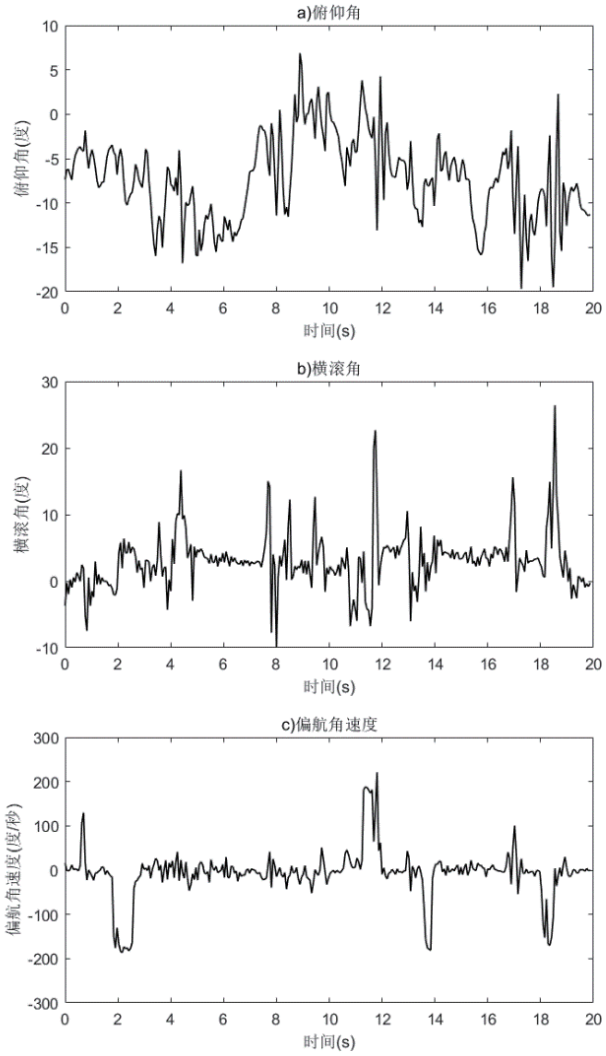


图 5-14 复杂地形通行实验



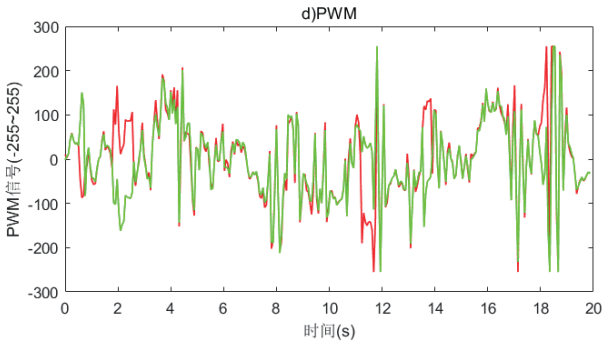


图 5-15 机器人复杂地形通行实验姿态变化

本次实验测试了机器人在复杂地形保持平衡、直行、转向、地形自适应 4 种运动能力，虽然机器人可以通过自身调整保证不倾倒，但在面对较为复杂地形时，系统仍然会有较为剧烈的振荡。这从图 5-15 的机器人姿态变化中也可以看出，机器人在三个轴方向上的姿态数据均有较大幅值和较高频率的变化。从实验结果分析，机器人在林地通行的表现不胜完美，但其能够通过各自身控制器的调整而保证系统不失稳或者倾倒，已然验证了机电系统设计和控制方法的合理性。

5.4 本章小结

为了验证两足轮腿机器人系统设计和控制设计的合理性，本章基于机器人样机进行一系列测试实验。包括直立平衡实验、直行实验、转向实验、高度调节实验、单腿越障实验、抗冲击实验以及复杂地形通行实验。实验中，机器人的实际运行效果展现了较强的平衡能力及地形适应能力，证明了机器人系统和控制方法的合理性。

结论

本设计的目标设计并制作一款可行的两足轮腿机器人样机。首先从自由度数目、机构构型、整体尺寸三个方面展开分析，提出了 4 自由度轮腿融接型的两足轮腿机器人结构方案。基于结构方案展开了对腿部结构的运动学分析和静态动力分析，为机器人的硬件选型和样机制作提供理论基础。再根据机器人运动要求，建立了机器人的三维 CAD 模型并完成了髋关节电机、主控芯片、传感器等电子硬件的选型。在运动控制设计与软件设计中，首先介绍了 PID 控制及其控制原理。然后基于 PID 控制设计了机器人运动控制器。通过软件程序设计来充分调用主控制板、传感器、控制硬件、动力硬件，用程序实现了机器人的各功能和运动控制。最后基于样机进行了一系列测试实验和实验分析，验证了两足轮腿机器人机电系统设计和运动控制方法的正确性。本设计主要完成了如下工作：

（1）提出了一种两足轮腿机器人的结构方案，并建立了其腿部结构的运动学模型和动态静力模型。根据机器人的运动功能需求，秉承简洁的设计原则，从腰身结构、腿部结构和轮部结构三方面展开，将结构方案进行了完整的设计。其中机器人腿部设计采用单输入平面四杆机构模型，仅需要一个驱动关节即可调整单个腿部结构的姿态。根据机械原理知识，对腿部的平面四杆机构进行运动学和动态静力分析和建模，为机器人的样机制作提供理论基础。

（2）设计了一种两足轮腿机器人机电系统，制作了一台两足轮腿机器人样机。根据结构方案，从机械结构、动力硬件设计和控制系统硬件三方面展开设计。设计了两足轮腿机器人的三维 CAD 模型，完成了相关电子硬件的选型并选用合适的系统拓展板，将各个功能硬件模块集成起来，简化了电路布线的同时也提高了机器人的稳定性能。最后用 3D 打印技术打印关键结构零件并将各零件、电路模块、拓展板进行装配，完成了两足轮腿机器人的机电系统设计和样机制作。

（3）开展了两足轮腿机器人的运动控制方法研究。根据 PID 控制理论和机电系统特点，设计了 PD 直立控制器、PI 速度控制器、P 转向控制器以及 P 髋关节角度控制器。然后对各控制器进行耦合，实现了两足机器人的基本运动控制、姿态控制和地形自适应控制等。最后设计了机器人的主程序和中断子程序，用计算机语言的方式充分调用主控芯片和控制硬件，完成了两足轮腿机器人的运动控制方法研究和相应的程序设计。

（4）基于两足轮腿机器人样机进行了一系列测试实验。在完成了结构方案设计、运动学分析、机电系统设计、运动控制以及软件设计后，利用机器人样机进行了一系列测试实验和实验分析，验证了两足轮腿机器人机电系统设计和运动控制方法的正确性。

对后续研究工作的展望：

（1）建立机器人腿部结构的正逆运动学模型。由于本设计两足轮腿机器人的腿部结构采用的是平面四杆机构模型，属于闭链机构，无法用常规的开链机构分析方法进行分析。而闭链机构的正逆运动学分析需要复杂的数学知识，不是本设计的研究重点。而本设计采用的平面四杆机构运动学分析方法在腿部结构的精确控制上有较大误差。

（2）改进机电系统动力硬件和控制硬件。性能更好的硬件通常会用更好的表现，例如：步进电机的精度高于直流电机；无刷电机的响应速度比伺服电机更快、扭矩更大；WIFI 比蓝牙更稳定、控制距离更远等。硬件的升级是机器人后续版本更新迭代中的首要问题。

（3）采用更先进和准确的控制理论。本设计基于 PID 理论设计的控制器，虽然机器人的实际运行表现不错，但仍然有很大的提升空间。采用更先进和准确的控制理论，如 LQR 二次型调节器控制、MPC 模型预测控制等，可以进一步提高机器人运行时的稳定性和鲁棒性。

致谢

本设计从选题、任务规划、内容研究、实验测试到论文撰写都得到了宋兴国老师的悉心指导，其渊博的专业知识和严谨的治学态度都深深感染着我。在生活中，宋老师时刻践行着“精勤求学，敦笃励志，果毅力行，忠恕任事”的交大校训，他用自己的行动为我辈学子了树立良好榜样。在此，谨向宋老师表示真诚的感谢和崇高的敬意。

感谢西南交通大学，母校对我的培养让我受益终生。

感谢我的室友和同学，本科四年的朝夕相处让我们如战友一般亲切。他们对我鼓励和帮助让我从容面对困难，祝他们前程似锦。

感谢我的家人和朋友，他们是我的坚强后盾，是我的心灵港湾，是我的动力源泉。他们的存在令我所畏惧，永远珍惜并深爱他们。

参考文献

[1] 贾硕,张文昌,吴航,陈炜,张永梅.救援机器人研究现状及其发展趋势[J].医疗卫生装备,2019,40(08):90-95.

[2] 民,王硕.机器人技术研究进展[J].自动化学报,2013,39(07):963-972.

[3] Nandhini, V. Krithika and K. Chittal, Design of four pedal quadruped robot, 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 2017:2548-2552

[4] 王一波,柳建.非结构环境下移动机器人定位研究综述[J].机床与液压,2021,49(08):176-181.

[5] Nagano and Y. Fujimoto, A control method of low speed wheeled locomotion for a wheel-legged mobile robot, 2014 IEEE 13th International Workshop on Advanced Motion Control (AMC), 2014:332-337

[6] De Viragh Y, Bjelonic M, Bellicoso C D, et al. Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Using Linearized ZMP Constraints [J]. IEEE Robotics and Automation Letters, 2019, 4(2): 1633-1640.

[7] Li Y. Dynamic Simulation Analyses of a Six-Leg-Wheel Hybrid Mobile Robot under Uneven Terrains [C]//Proceedings of the International Conference on Intelligent Networks and Intelligent Systems, 2010: 308-311.

[8] Matsumoto O, Kajita S, Saigo M, et al. Dynamic Trajectory Control of Passing Over Stairs by A Biped Type Leg-Wheeled Robot With Nominal Reference of Static Gait [C]//Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 1998: 406-412.

[9] Matsumoto O, Kajita S & Komoriya K. Flexible Locomotion Control of A Self-

Contained Biped Leg-Wheeled System [C]//Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002: 2599-2604.

[10] Hashimoto K, Hosobata T, Sugahara Y, et al. Realization by Biped Leg-wheeled Robot of Biped Walking and Wheel-driven Locomotion[C]//Proceedings of the IEEE/RSJ International Conference on Robotics and Automation, 2005: 2970-2975.

[11] Canete L & Takahashi T. Development of A Single Controller for The Compensation of Several Types of Disturbances During Task Execution of A Wheeled Inverted Pendulum Assistant Robot [C]//Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014: 2414-2420.

[12] Canete L & Takahashi T. Development of A Strain Gauge Based Disturbance Estimation And Compensation Technique for A Wheeled Inverted Pendulum Robot [C]//Proceedings of the International Conference on Robotics and Automation (ICRA), 2019: 2613-2619.

[13] Jung T, Lim J, Bae H, et al. Development of the Humanoid Disaster Response Platform DRC-HUBO+ [C]//Proceedings of the IEEE Transactions on Robotics, 2018, 34: 1-17.

[14] Zhang Y, Luo J, Hauser K , et al. Motion Planning And Control of Ladder Climbing on DRC-Hubo for DARPA Robotics Challenge [C]//Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2014: 2086-2086.

[15] Bae H, Lee I, Jung T, et al. Walking-wheeling Dual Mode Strategy for Humanoid Robot, DRC-HUBO [C]//Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016: 1342-1348.

[16] Wang H, Zheng Y F, Jun Y, et al. DRC-Hubo Walking on Rough Terrains [C]//Proceedings of the IEEE International Conference on Technologies for Practical Robot Applications (TePRA), 2014: 1-6.

[17]刘京运.从 Big Dog 到 Spot Mini:波士顿动力四足机器人进化史[J].机器人产业,2018(02):109-116.

[18]Klemm V, Morra A, Salzmann C, et al. Ascento: A Two-Wheeled Jumping Robot [C]//Proceedings of the International Conference on Robotics and Automation (ICRA), 2019:7515-7521.

[19]Klemm V , Morra A , Gulich L , et al. LQR-Assisted Whole-Body Control of a Wheeled Bipedal Robot With Kinematic Loops [J]. IEEE Robotics and Automation Letters, 2020, 5(2): 3745-3752.

[20]Li X, Zhou H, Feng H, et al. Design and Experiments of a Novel Hydraulic Wheel-legged Robot (WLR) [C]//Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2018: 3292-3297.

[21]Li J, Zhou H, Feng H, et al. Gain Scheduling Control of Wheel-Legged Robot LPV system Based on HOSVD [C]//Proceedings of the International Conference on Mechatronics and Automation, 2019: 2481-2486.

[22]Li X, Zhou H, Zhang S, et al. WLR-II, a Hose-less Hydraulic Wheel-legged Robot [C]//Proceedings of the International Conference on Intelligent Robots and Systems (IROS), 2019:4339-4346.

[23]Wang S, Cui L, Zhang J, et al. Balance Control of a Novel Wheel-legged Robot: Design and Experiments [C]//Proceedings of the International Conference on Robotics and Automation, 2021: 6782-6788.

[24]梁光胜,杜梦楠,周子豪,刘春旭,文月.基于互补滤波的两轮自平衡车姿态控制[J]. 测控技术, 2015, 34(05): 72-74.

[25]陆军.基于 PID 和 LQR 控制的两轮自平衡小车研究[D]. 西南交通大学, 2012.

[26]徐子力.双足溜冰机器人动力学建模与步态研究[D]. 上海交通大学, 2007.

[27] 孙桓,陈作模,葛文杰主编.机械原理[M]. 高等教育出版社,2013.

[28] 姜利彬. 一种双轮自平衡车的设计和实现[D]. 东南大学, 2016.

[29] 张玲娜. 基于霍尔传感器的电机转速测量与调速系统设计和研究[D].西安石油大学,2016.

[30] 肖世德,唐猛,孟祥印,黄慧萍.机电一体化系统监测与控制.上册[M].西南交通大学出版社,2011.

[31] 何芝强. PID 控制器参数整定方法及其应用研究[D].浙江大学,2005.

[32] 付任. 两轮自平衡倒立摆式机器人的建模与控制[D].大连海事大学,2016.

[33] 汪宇辰. 智能轮式机器人平台软硬件设计和系统控制[D]. 重庆大学.

[34] JianWei Zhao and XiaoGang, The control and design of Dual - wheel upright self-balance Robot, 2008 7th World Congress on Intelligent Control and Automation, 2008: 4172-4177.

[35] Yong Qin, Xizhe Zang, Yubin Liu, Jie Zhao, Xiaoyu Wang and Hegao Cai, Research of Trajectory Tracking Control of Two-Wheeled Self-Balance Robot, 2006 6th World Congress on Intelligent Control and Automation, 2006:8948-8952.

[36] Ngo H Q T, Nguyen T P, Huynh V N S, et al. Experimental Comparison of Complementary Filter and Kalman Filter Design for Low-cost Sensor in Quadcopter[C]//Proceedings of the International Conference on System Science and Engineering, 2017: 488-493.

附录

（1）Banlance_Robot.ino 文件

```
#include "PinChangeInt.h" //提供中断方法，在setup()中用到
#include "MsTimer2.h"
#include "BalanceCar.h" //提供speedout() turnout() pwma()三个方法，
#include "KalmanFilter.h" //
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
MPU6050 mpu; //实例化一个 MPU6050 对象，对象名称为 mpu
BalanceCar balancecar;
KalmanFilter kalmanfilter;
int16_t ax, ay, az, gx, gy, gz;   //三轴加速度计值和三轴角速度计值
#define AIN1 7  //TB6612FNG 驱动模块控制信号
#define AIN2 6
#define BIN1 13   //区分左右
#define BIN2 12   //PWMA--right--M2
#define PWMA 9    //PWMB--left--M1
#define PWMB 10
#define STBY 8

#define PinA_left 2  //中断0  左右编码器中的A 相，有转动就会产生高低电平，进而进入中断函数
#define PinA_right 4 //中断1

#define servopin_left A2   //舵机引脚
#define servopin_right A3
int servo_angle_right = 10;  //角度值
int servo_angle_left = 80;  //角度值

//声明自定义变量
float pwm_speed = 0;   //各个环节的pwm 值，在pwma()函数中汇总
float pwm_turn = 0;
```

```
31.
32. float kp =38, kd = 0.48;                          //直立环PID，保证小车直立
33. float kp_speed =3.8,  ki_speed = 0.11;   // 速度PID，保证小车稳定在原地，或者说保证小车速度为0
34. float kp_turn = 28, kd_turn = 0.29;                        //旋转PID
35.
36. float speed_setpoint = 0.00;   //想要的角度值和速度值setpoint,其中用 脉冲 pulse 间接表示 速度speed
37. float Pitch_setpoint = -7.0;
38. float Pitch_setpoint_10 = -7.0;    //不同的腿部高度机械倾角不同，此处将腿部10-60度分为3段，设置三个倾角
39. float Pitch_setpoint_35 = -11.50;
40. float Pitch_setpoint_60 = -13.0;
41.
42. //***************Kalman_Filter*********************
43. float Q_angle = 0.001, Q_gyro = 0.005; //角度数据置信度,角速度数据置信度
44. float R_angle = 0.5 , C_0 = 1;
45. float timeChange = 5; //滤波法采样时间间隔毫秒
46. float K1 = 0.05; // 对加速度计取值的权重
47. float dt = timeChange * 0.001; //注意：dt 的取值为滤波器采样时间
48.
49. volatile long count_right = 0;//左右编码器脉冲数，同时作为中间变量。使用volatile long 类型是为了外部中断脉冲计数值在其他函数中使用时，确保数值有效
50. volatile long count_left = 0;
51.
52. #define run_car   '1'//按键前
53. #define back_car  '2'//按键后
54. #define left_car  '3'//按键左
55. #define right_car '4'//按键右
56. #define stop_car  '0'//按键停
57.
58. //***************蓝牙数据传输*********************
59. int Mode = 0;
60. int incomingByte = 0;          // 接收到的 data byte
61. String inputString = "";         // 用来储存接收到的内容
62. boolean newLineReceived = false; // 前一次数据结束标志
```

```
63. boolean startBit  = false;  //协议开始标志
64.
65. int forwordflag = 0;//前进标志      //各个步态标志，蓝牙控制时用到
66. int backflag = 0;//后退标志
67. int turnleftflag = 0;//左转标志
68. int turnrightflag = 0;//右转标志
69.
70. void ResetCarState() //重置指令参数
71. {
72.   forwordflag = 0;
73.   backflag = 0;
74.   turnleftflag = 0;
75.   turnrightflag = 0;
76.   pwm_turn = 0;
77. }
78.
79. ////////////////////////////////////////////脉冲计算
////////////////////////////////////////////
80. void Code_left() {
81.   count_left++;
82. }
83. void Code_right() {
84.   count_right++;
85. }                    //左右编码器脉冲计数函数，使用外部中断的方式进入
86.
87. int pluseright_temp = 0;
88. int pluseleft_temp = 0;   //中间变量,目的是将 脉冲数 count_left 传递
给 balancecar.pulseleft
89.
90. void countpluse()  //主要功能是区分正负脉冲并汇总，定时中断每 5ms 进入 inter()
函数时，同时也进入了 countpluse()
91. {
92.   pluseleft_temp = count_left;
93.   pluseright_temp = count_right;
94.
95.   count_left = 0;
```

```
96.   count_right = 0;
97.
98.   if((balancecar.pwm_right < 0) && (balancecar.pwm_left < 0))       //小车运动方向判断 后退时（PWM 即电机电压为负） 脉冲数为负数
99.   {
100.        pluseright_temp = -pluseright_temp;
101.        pluseleft_temp = -pluseleft_temp;
102.      }
103.      else if((balancecar.pwm_right > 0) && (balancecar.pwm_left > 0))  //小车运动方向判断 前进时（PWM 即电机电压为正） 脉冲数为负数
104.      {
105.        pluseright_temp = pluseright_temp;
106.        pluseleft_temp = pluseleft_temp;
107.      }
108.      else if((balancecar.pwm_right < 0) && (balancecar.pwm_left > 0))  //小车运动方向判断 左旋转 左脉冲数为负数 右脉冲数为正数
109.      {
110.        pluseright_temp = pluseright_temp;    //由于实际安装是反安装，这里与实际安装相符
111.        pluseleft_temp = -pluseleft_temp;
112.      }
113.      else if((balancecar.pwm_right > 0) && (balancecar.pwm_left < 0))  //小车运动方向判断 左旋转 左脉冲数为负数 右脉冲数为正数
114.      {
115.        pluseright_temp = -pluseright_temp;
116.        pluseleft_temp = +pluseleft_temp;
117.      }
118.
119.      balancecar.pulseright += pluseright_temp;   //将汇总脉冲传递给 balancecar.pulseright，用于速度环
120.      balancecar.pulseleft += pluseleft_temp;
121.    }
122.    ////////////////////////////////////////////脉冲计算 ////////////////////////////////////////////
123.
124.
```

```
125.    //////////////////直立环，角度PD//////////////////由于用到了kalmanfilter.h且简单，直接放在主程序中
126.    void angleout()
127.    {
128.
129.      if(servo_angle_right>=10&&servo_angle_right<25) Pitch_setpoint = Pitch_setpoint_10;
130.      else if(servo_angle_right>=25&&servo_angle_right<45) Pitch_setpoint = Pitch_setpoint_35;
131.      else Pitch_setpoint = Pitch_setpoint_60;
132.      balancecar.pwm_angle = kp * (kalmanfilter.angle - Pitch_setpoint) + kd * kalmanfilter.Gyro_x;//PD 角度环控制
133.    }
134.    //////////////////角度PD//////////////////
135.
136.    //////////////////定时中断，每5ms进入//////////////////
137.    int speedcount = 0;
138.    int turncount = 0;// 进入速度环和旋转环的计数器
139.    void inter()
140.    {
141.      sei(); //开启中断函数
142.      noInterrupts();
143.      countpluse();  //脉冲汇总子函数
144.      interrupts();
145.      mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  //IIC 获取MPU6050 六轴数据 ax ay az gx gy gz
146.      kalmanfilter.Angletest(ax, ay, az, gx, gy, gz, dt, Q_angle, Q_gyro, R_angle, C_0, K1); //获取angle 角度和卡曼滤波
147.
148.    ////////////PID，先由angleout(), speedout(), turnout()得到pwm_angle等，再由pwma()进行电机驱动////////////
149.      angleout(); //角度环PD 控制，得到pwm_angle
150.
151.      speedcount++;
152.      if (speedcount >= 8)   //40ms 进入速度环控制,得到pwm_speed
153.      {
```

```
154.        pwm_speed = balancecar.speedout(kp_speed, ki_speed, forwordflag, backflag, speed_setpoint);
155.        speedcount = 0;
156.      }
157.
158.      turncount++;
159.      if (turncount > 4) //20ms 进入旋转控制,得到pwm_turn
160.      {
161.        pwm_turn = balancecar.turnout(turnleftflag, turnrightflag, kp_turn, kd_turn, kalmanfilter.Gyro_z);
162.        turncount = 0;
163.      }
164.      balancecar.pwma(pwm_speed, pwm_turn, kalmanfilter.angle, AIN1, AIN2, BIN1, BIN2, PWMA, PWMB);   //小车总 PWM 输出
165.    ////////////PID，先由 angleout(), speedout(), turnout()得到pwm_angle 等，再由 pwma()进行电机驱动/////////////
166.    }
167.    ////////////////////定时中断，每 5ms 进入//////////////////////
168.
169.
170.    ////////////////////舵机驱动函数//////////////////////
171.    void SetServoAngle(int servopin, int angle)//定义一个脉冲函数,这个函数的频率是 50hz 的，20000us=20ms=50hz
172.    {
173.      int pulsewidth=(angle*11)+500;  //将角度转化为 500-2480 的脉宽值
174.      digitalWrite(servopin,HIGH);     //将舵机接口电平至高,反过来也是可以的
175.      delayMicroseconds(pulsewidth);  //延时脉宽值的微秒数
176.      digitalWrite(servopin,LOW);      //将舵机接口电平至低
177.      delayMicroseconds(20000-pulsewidth);
178.    }
179.    ////////////////////舵机驱动函数//////////////////////
180.
181.    // ===     初始设置 setup()      ===
182.    void setup() {
183.      pinMode(servopin_left,OUTPUT);//设定舵机接口为输出接口
184.      pinMode(servopin_right,OUTPUT);
```

```
185.      pinMode(AIN1, OUTPUT);   //控制电机 2 的方向，01 为正转，10 为反转      //区分左右
186.      pinMode(AIN2, OUTPUT);                                                  //PWMA--right--M2
187.      pinMode(BIN1, OUTPUT);   //控制电机 1 的方向，01 为正转，10 为反转      //PWMB--left--M1
188.      pinMode(BIN2, OUTPUT);
189.      pinMode(PWMA, OUTPUT);   //右电机 PWM
190.      pinMode(PWMB, OUTPUT);   //左电机 PWM
191.      pinMode(STBY, OUTPUT);   //TB6612FNG 使能端
192.
193.      digitalWrite(AIN1, 0);  //初始化电机驱动模块
194.      digitalWrite(AIN2, 1);
195.      digitalWrite(BIN1, 1);
196.      digitalWrite(BIN2, 0);
197.      digitalWrite(STBY, 1);
198.      analogWrite(PWMA, 0);
199.      analogWrite(PWMB, 0);
200.
201.      pinMode(PinA_left, INPUT);  //测速码盘输入
202.      pinMode(PinA_right, INPUT);
203.
204.      Wire.begin();   //加入 I2C 总线序列
205.      delay(500);
206.      mpu.initialize();  //初始化 MPU6050
207.      delay(2);
208.      balancecar.pwm_right = 0;  //初始左右速度为 0
209.      balancecar.pwm_left = 0;
210.
211.      //5ms 定时中断设置  使用 timer2  注意：使用 timer2 会对 pin3 pin11 的 PWM 输出有影响，因为 PWM 使用的是定时器控制占空比，
212.      //所以在使用 timer 的时候要注意查看对应 timer 的 pin 口。
213.      MsTimer2::set(5, inter);   //每 5ms 进入 inter()函数--------平衡主要程序
214.      MsTimer2::start();
215.
```

```
216.      attachInterrupt(digitalPinToInterrupt(PinA_left), Code_left, CHANGE
);       //不能同时用两个attachInterrupt()
217.      attachPinChangeInterrupt(PinA_right, Code_right, CHANGE);   //pinCh
angeInt 库内的方法
218.
219.      Serial.begin(9600);
220.      Serial.println("CLEARDATA");
221.      Serial.println("LABEL,Date,Time,Millis,Angle_x,Angle_y,Gyro_x,Gyro_
z,PWM_left,PWM_right,");
222.    }
223.
224.    // ===        主循环程序体 loop()        ===
225.    void loop() {
226.      if(newLineReceived)
227.      {
228.        switch (inputString[1]) //[0]是$，第二位数据是前后左右标志位
229.        {
230.          case run_car:   ResetCarState();forwordflag = -250; break;
231.          case back_car:  ResetCarState();backflag = 250; break;
232.          case left_car:  turnleftflag = 1; break;
233.          case right_car: turnrightflag = 1; break;
234.          case stop_car:  ResetCarState(); break;
235.          default: ResetCarState(); break;
236.        }
237.        if(inputString[5]=='1')Mode = 1;//1：开启高度调节；2：关闭高度调节；
3：自平衡。相当于 2 既不进入 1 也不进入 3，只有前后左右
238.        if(inputString[5]=='2')Mode = 2;
239.        if(inputString[5]=='3')Mode = 3;
240.        if(Mode==1)
241.        {
242.          switch (inputString[3])
243.          {
244.            case '1':  SetServoAngle(servopin_right, 10); SetServoAngle(s
ervopin_left, 80);delay(100);servo_angle_right=10;break;
245.            case '2':  SetServoAngle(servopin_right, 35); SetServoAngle(s
ervopin_left, 55);delay(100);servo_angle_right=35;break;
```

```
246.           case '3':  SetServoAngle(servopin_right, 60); SetServoAngle(s
ervopin_left, 30);delay(100);servo_angle_right=60;break;
247.           default: SetServoAngle(servopin_right, 10); SetServoAngle(ser
vopin_left, 80);delay(100);servo_angle_right=10;break;
248.         }
249.       }
250.
251.       inputString = "";
252.       newLineReceived = false;
253.     }
254.     if(Mode==3)
255.     {
256.       if(kalmanfilter.angle6 > 3)
257.       {
258.         servo_angle_right--;
259.         servo_angle_left--;
260.       }
261.       if(kalmanfilter.angle6 < -3)
262.       {
263.         servo_angle_right++;
264.         servo_angle_left++;
265.       }
266.
267.       if(servo_angle_right>60)servo_angle_right=60;
268.       if(servo_angle_right<10)servo_angle_right=10;
269.       SetServoAngle(servopin_right, servo_angle_right);
270.       SetServoAngle(servopin_left, servo_angle_left);
271.       delay(100);
272.     }
273.
274.     Serial.println((String) "DATA,DATE,TIME," + millis() + "," + kalman
filter.angle + "," + kalmanfilter.angle6 + "," + kalmanfilter.Gyro_x + "," +
 kalmanfilter.Gyro_z + "," + balancecar.pwm_left+ "," + balancecar.pwm_right
);
275.   }
276.
```

```
277.    int BTTrans_flag = 0;
278.    void serialEvent()
279.    {
280.      while (Serial.available())
281.      {
282.        incomingByte = Serial.read();              //一个字节一个字节地读,
下一句是读到的放入字符串数组中组成一个完成的数据包
283.        if (incomingByte == '$')
284.        {
285.          BTTrans_flag = 0;
286.          startBit = true;
287.        }
288.        if (startBit == true)
289.        {
290.          BTTrans_flag++;
291.          inputString += (char) incomingByte;     // 全双工串口可以不用在下
面加延时，半双工则要加的//
292.        }
293.        if (startBit == true && incomingByte == '#')
294.        {
295.          newLineReceived = true;
296.          startBit = false;
297.        }
298.
299.        if(BTTrans_flag >= 80)
300.        {
301.          BTTrans_flag = 0;
302.          startBit = false;
303.          newLineReceived = false;
304.          inputString = "";
305.        }
306.      }
307.    }
```

（2）Banlance_car.h 文件

```
#ifndef BalanceCar_h
#define BalanceCar_h

#if defined(ARDUINO) && (ARDUINO >= 100)
#include <Arduino.h>
#else
#include <WProgram.h>
#endif


class BalanceCar
{
public:
  double speedout(double kp_speed,double ki_speed,int forwordflag,int  backflag,double speed_setpoint);
  float turnout(int turnleftflag,int turnrightflag, double kp_turn,double kd_turn,float Gyroz);
  void pwma(double pwm_speed,float pwm_turn,float angle, int Pin1,int Pin2,int Pin3,int Pin4,int PinPWMA,int PinPWMB);
 int pulseright = 0;
 int pulseleft = 0;
  float speeds;
 double pwm_angle=0,pwm_left = 0, pwm_right = 0;
private:
 float speeds_filterold;
 float positions;
 int turnmax = 0;
 int turnmin = 0;
 float turnout_para = 0;
};
#endif
//
// END OF FILE
//
```

（3）Banlance_car.cpp 文件

```
#include "./BalanceCar.h"

double BalanceCar::speedout(double kp_speed,double ki_speed, int forwordflag,int backflag,double speed_setpoint) //形参和实参可以同名
{
  speeds = (pulseleft + pulseright) * 1.0;
  pulseright = pulseleft = 0;
  speeds_filterold *= 0.7;
  float speeds_filter = speeds_filterold + speeds * 0.3;          //滤波算法?
  speeds_filterold = speeds_filter;
   positions += speeds_filter;
   positions += forwordflag;
   positions += backflag;
   positions = constrain(positions, -3000,3000);

  double pwm_speed = ki_speed * (speed_setpoint - positions) + kp_speed * (speed_setpoint - speeds_filter);
  return pwm_speed;
}

float BalanceCar::turnout(int turnleftflag,int turnrightflag, double kp_turn,double kd_turn,float Gyroz)
{
  int spinonce = 0;
  float turnspeed = 0;
  float rotationratio = 0;

  if (turnleftflag == 1 || turnrightflag == 1)
  {
    if (spinonce == 0)
    {
      turnspeed = ( pulseright + pulseleft);
      spinonce++;
    }
```

```
32.     if (turnspeed < 0)
33.     {
34.       turnspeed = -turnspeed;
35.     }
36.     if(turnleftflag==1||turnrightflag==1)
37.     {
38.      turnmax=5;
39.      turnmin=-5;
40.     }
41.     rotationratio = 5 / turnspeed;
42.     if (rotationratio < 0.5)rotationratio = 0.5;
43.     if (rotationratio > 5)rotationratio = 5;
44.   }
45.   else
46.   {
47.     rotationratio = 0.5;
48.     spinonce = 0;
49.     turnspeed = 0;
50.   }
51.   if (turnleftflag == 1)
52.   {
53.     turnout_para += rotationratio;
54.   }
55.   else if (turnrightflag == 1)
56.   {
57.     turnout_para -= rotationratio;
58.   }
59.   else turnout_para = 0;
60.   if (turnout_para > turnmax) turnout_para = turnmax;
61.   if (turnout_para < turnmin) turnout_para = turnmin;
62.
63.   double pwm_turn = -turnout_para * kp_turn - Gyroz * kd_turn;
64.   return pwm_turn;
65. }
66.
```

```
67. void BalanceCar::pwma(double pwm_speed,float pwm_turn,float angle, int P
in1,int Pin2,int Pin3,int Pin4,int PinPWMA,int PinPWMB)
68. {
69.   pwm_right = -pwm_angle - pwm_speed - pwm_turn; //pwma()的关键代码
70.   pwm_left = -pwm_angle - pwm_speed + pwm_turn;
71.
72.   if (pwm_right > 255) pwm_right = 255;
73.   if (pwm_right < -255) pwm_right = -255;
74.   if (pwm_left > 255) pwm_left = 255;
75.   if (pwm_left < -255) pwm_left = -255;
76.
77.   if (angle > 30 || angle < -30)  //角度大于 30°表示摔倒
78.   {
79.     pwm_right = 0;
80.     pwm_left = 0;
81.   }
82.
83.   if (pwm_right >= 0) {
84.     digitalWrite(Pin2, 0);
85.     digitalWrite(Pin1, 1);
86.     analogWrite(PinPWMA, pwm_right);
87.   }
88.   else {
89.     digitalWrite(Pin2, 1);
90.     digitalWrite(Pin1, 0);
91.     analogWrite(PinPWMA, -pwm_right);
92.   }
93.   if (pwm_left >= 0) {
94.     digitalWrite(Pin4, 0);
95.     digitalWrite(Pin3, 1);
96.     analogWrite(PinPWMB, pwm_left);
97.   }
98.   else {
99.     digitalWrite(Pin4, 1);
100.        digitalWrite(Pin3, 0);
101.        analogWrite(PinPWMB, -pwm_left);
```

```
102.      }
103.    }
```

（4）Kalman_Filter.h 文件

```
1. #ifndef KalmanFilter_h
2. #define KalmanFilter_h
3.
4. #if defined(ARDUINO) && (ARDUINO >= 100)
5. #include <Arduino.h>
6. #else
7. #include <WProgram.h>
8. #endif
9. class KalmanFilter
10. {
11. public:
12.   void Yiorderfilter(float angle_m, float gyro_m,float dt,float K1);
13.   void Kalman_Filter(double angle_m, double gyro_m,float dt,float Q_angle
,float Q_gyro,float R_angle,float C_0);
14.   void Angletest(int16_t ax,int16_t ay,int16_t az,int16_t gx,int16_t gy,i
nt16_t gz,float dt,float Q_angle,float Q_gyro,
15.           float R_angle,float C_0,float K1);
16.    float Gyro_x,Gyro_y,Gyro_z;
17.    float accelz = 0;
18.    float angle;
19.    float angle6;
20. private:
21.   float angle_err,q_bias;
22.   float Pdot[4] = { 0, 0, 0, 0};
23.   float P[2][2] = {{ 1, 0 }, { 0, 1 }};
24.   float  PCt_0, PCt_1, E, K_0, K_1, t_0, t_1;
25.   float angle_dot;
26. };
27. #endif
28. //
29. // END OF FILE
30. //
```

（5）Kalman_Filter.cpp 文件

```
1. //Yiorderfilter()一阶滤波算法，和卡尔曼滤波算法同级
2. //Kalman_Filter()真正的卡尔曼滤波
3. //Angletest()数据转换，设置想要的数据，同时内置 Kalman_Filter()
4. //主程序中 angle data 和 Kalman_Filter 里面的数据好像不用改？
5.
6. ////////////////////调用方法////////////////////
7. //在 Angletest()里面设置需要的数据，在主程序中直接调用 Angletest()
8.
9. #include "KalmanFilter.h"
10.
11. ////////////////////////Yiorderfilter//////////////////
12. void KalmanFilter::Yiorderfilter(float angle_m, float gyro_m,float dt,fl
oat K1)   //angle6 是 x 轴角度
13. {
14.   angle6 = K1 * angle_m + (1 - K1) * (angle6 + gyro_m * dt);
15.  // return angle6;
16. }
17.
18. ////////////////////////kalman/////////////////////////
19. void KalmanFilter::Kalman_Filter(double angle_m, double gyro_m,float dt,
float Q_angle,float Q_gyro,float R_angle,float C_0)
20. {
21.   angle += (gyro_m - q_bias) * dt;
22.   angle_err = angle_m - angle;
23.   Pdot[0] = Q_angle - P[0][1] - P[1][0];
24.   Pdot[1] = - P[1][1];
25.   Pdot[2] = - P[1][1];
26.   Pdot[3] = Q_gyro;
27.   P[0][0] += Pdot[0] * dt;
28.   P[0][1] += Pdot[1] * dt;
29.   P[1][0] += Pdot[2] * dt;
30.   P[1][1] += Pdot[3] * dt;
31.   PCt_0 = C_0 * P[0][0];
32.   PCt_1 = C_0 * P[1][0];
33.   E = R_angle + C_0 * PCt_0;
```

```
34.   K_0 = PCt_0 / E;
35.   K_1 = PCt_1 / E;
36.   t_0 = PCt_0;
37.   t_1 = C_0 * P[0][1];
38.   P[0][0] -= K_0 * t_0;
39.   P[0][1] -= K_0 * t_1;
40.   P[1][0] -= K_1 * t_0;
41.   P[1][1] -= K_1 * t_1;
42.   angle += K_0 * angle_err;
43.   q_bias += K_1 * angle_err;
44.   angle_dot = gyro_m - q_bias;
45. }
46. ////////////////////////kalman/////////////////////////
47.
48. ////////////////////////////// Angle test//////////////////////////////////
49. void KalmanFilter::Angletest(int16_t ax,int16_t ay,int16_t az,int16_t gx
,int16_t gy,int16_t gz,float dt,float Q_angle,float Q_gyro,
50.                               float R_angle,float C_0,float K1)
51. {
52.   float Angle = atan2(ay , az) * 57.29578;    //这部分是卡尔曼滤波的调
用
53.   Gyro_x = (gx) / 131.00;
54.   Kalman_Filter(Angle, Gyro_x, dt, Q_angle, Q_gyro, R_angle, C_0);

55.
56.   if (gz > 32768) gz -= 65536;   //Gyro_z,acclz 数据转换，直接调
用
57.   Gyro_z = -gz / 131.00;
58.   accelz = az / 16.4;
59.
60.   float angleAx = atan2(ax, az) * 57.29578;    //这部分是一阶滤波的调
用
61.   Gyro_y = -gy / 131.00;
62.   Yiorderfilter(angleAx, Gyro_y, dt, K1);
63. }
```