摘要

本文针对目前下肢瘫痪人员由于久卧在床，肌肉萎缩并且需要他人持续帮助的缺点，在已有智能康复设备的基础上，利用CAD设计基础机械并利用Arduino程序设计单片机控制，并通过串口与PC通信；利用Unity3D编程制作动画软件实现及时反馈患者的动作轨迹并对患者今后的训练提出建议，利用步进电机实现下肢三轴自由度康复训练的智能康复机器人，并含有多种训练模式，让患者能在床边独自完成康复训练，并提出其可能存在的缺点和今后的改进方向。

引言

随着经济的发展，人们迫切希望生活水平的提高，特别是一些需要特殊照顾的人群，如可能遭遇意外事故而下肢发生部分瘫痪的患者，由于行动不便，不仅自身生活不便，也对家人的生活产生了很多麻烦，有些患者由于长期身体不活动造成肌肉萎缩，脂肪增加，体重严重超标，严重的甚至翻身都需要家人帮助，因此在这个电子技术高度发展的时代，完全可以利用电子控制制作一种能让他们自行进行康复训练的智能机器人，通过腿脚的不断运动，逐渐使肌肉有力，学会平衡控制和走路节奏，有利于他们的康复，并减轻了家人的负担。这种机器人通过控制患者的腿脚路径并实时给予反馈，让患者对自身状态有直观的认识。如果患者在长期的康复训练下得到恢复，不仅方便了他们的生活，而且经过康复训练的他们对生活更加有信心，对整个家庭的心理上的恢复都是大有裨益的。

这款机器人基础机械部分采用CAD设计，利用Ardunio单片机实现电机转速、方向调节和模式选择，通过串口与PC通信，每个关节部位采用3台步进电机实现三轴任意方向控制。实时动作反馈采用Unity3D软件制作。

患者脚底压力数据

压力传感器

Arduino Uno控制器

控制

步进电机

PC

控制

电机运动状态数据

游戏界面

电机运动状态数据

硬件设计

步进电机

步进电机是将电脉冲信号转变为角位移或线位移的开环控制电机，是现代数字程序控制系统中的主要执行元件，应用极为广泛。在非超载的情况下，[电机](https://baike.baidu.com/item/%E7%94%B5%E6%9C%BA/117901)的转速、停止的位置只取决于脉冲信号的频率和脉冲数，而不受负载变化的影响，当步进驱动器接收到一个脉冲信号，它就驱动步进电机按设定的方向转动一个固定的角度，称为“步距角”，它的旋转是以固定的角度一步一步运行的。可以通过控制脉冲个数来控制角位移量，从而达到准确定位的目的；同时可以通过控制[脉冲频率](https://baike.baidu.com/item/%E8%84%89%E5%86%B2%E9%A2%91%E7%8E%87/9841161)来控制电机转动的[速度](https://baike.baidu.com/item/%E9%80%9F%E5%BA%A6/5456)和[加速度](https://baike.baidu.com/item/%E5%8A%A0%E9%80%9F%E5%BA%A6/3764)，从而达到调速的目的。步进电机采用开路控制，不需要反馈电路。

步进电机是一种感应电机，它的工作原理是利用电子电路，将直流电变成分时供电的，多相时序控制电流，用这种电流为步进电机供电，步进电机才能正常工作，驱动器就是为步进电机分时供电的，多相时序控制器。

本品选用的步进电机步距角为1.8度，配合A4988驱动器可实现1,1/2,1/4,1/8,1/16

步距角实现更精确的控制，电源采用5VDC通过Arduino控制器转换成脉冲波信号传递给A4988驱动器。

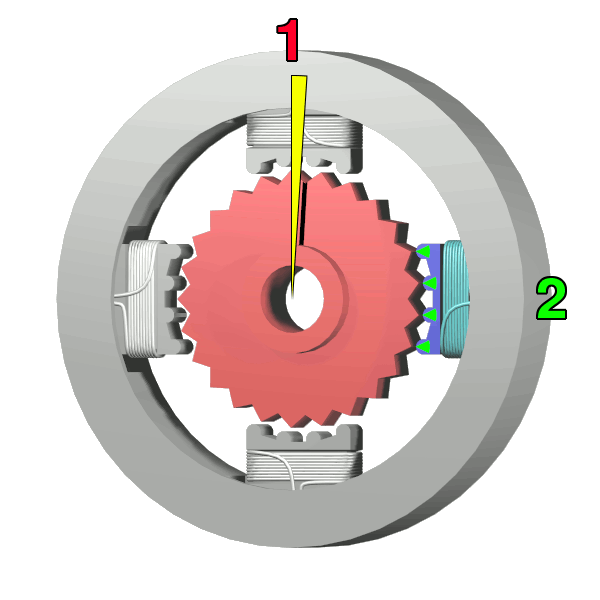
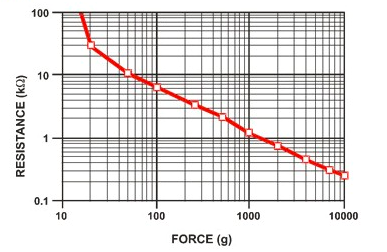


图 步进电机步级角构造

FSR402 压力传感器

本品采用FSR402压力传感器采集患者双角压力数据。其原理是变化的压力使内部对压力敏感的半导体元件的电阻值发射变化，从而检测出压力的变化。机械参数等如下图。

电阻值和压力关系如下图。



利用Unity3D实现人物移动

Unity3D是由Unity Technologies开发的一个让玩家轻松创建诸如三维视频游戏、建筑可视化、实时三维动画等类型互动内容的多平台的综合型游戏开发工具，是一个全面整合的专业游戏引擎。可用于开发各种平台的游戏，还被广泛用于建筑可视化和实时三维动画等类型互动内容的综合型创作工具。本品利用Unity工具实现电机的位移与PC中人物的移动同步，使患者能即时知道自己的姿态，对身体运动有更感性的认识。并可通过模式控制患者的运动距离和运动方式，并在最后有结束奖励机制使患者体会到康复训练的成就感。通过此设计，患者不再在康复训练中呆板地看着天花板，而是看到动态的画面，给患者生活增添了信心和乐趣。

在利用Unity3D实现人物和电机位移同步时，需要首先准备好人物预制件和环境预制件(.prefab)，以便随时调用。在控制腿部运动的过程中需要制作AnimatorController，在腿部各运动节点需要脚本文件（.cs）采集来自电机的信息以控制其运动，并可设置其运动距离，在到达时停止运动。在编辑人物这个层次(hierarchy)时还需调用更多元素（components）,如刚体（rigid），使其能和现实中的物体一样能受重力作用等。还需要控制照相机（camera）和人物同步运动，并用脚本文件实施控制。

系统初始化

选择模式

压力传感器读数大于零

电机不转

电机按设定模式运行

No

Yes

人的运动控制

有了人物和环境预制件后，就应考虑如何控制人物的运动。人的腿放在机器上时，电机就随之按规定模式转动。电机运动与人物同步。控制人物运动的代码如下：

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using System.Text;  public class PlayerMovement : MonoBehaviour {  public float speed = 6f; // The speed that the player will move at.  Vector3 movement; // The vector to store the direction of the player's movement.  Animator anim; // Reference to the animator component.  Rigidbody playerRigidbody; // Reference to the player's rigidbody.  int floorMask; // A layer mask so that a ray can be cast just at gameobjects on the floor layer.  float camRayLength = 100f; // The length of the ray from the camera into the scene.  void Awake()  {  // Create a layer mask for the floor layer.  floorMask = LayerMask.GetMask("Floor");  // Set up references.  anim = GetComponent<Animator>();  playerRigidbody = GetComponent<Rigidbody>();  }  void FixedUpdate()  {  // Store the input axes.  float h = Input.GetAxisRaw("Horizontal");  float v = Input.GetAxisRaw("Vertical");  // Move the player around the scene.  Move(h, v);  // Turn the player to face the mouse cursor.  Turning();  // Animate the player.  Animating(h, v);  }  void Move(float h, float v)  {  // Set the movement vector based on the axis input.  movement.Set(h, 0f, v);  // Normalise the movement vector and make it proportional to the speed per second.  movement = movement.normalized \* speed \* Time.deltaTime;  // Move the player to it's current position plus the movement.  playerRigidbody.MovePosition(transform.position + movement);  }  void Turning()  {  // Create a ray from the mouse cursor on screen in the direction of the camera.  Ray camRay = Camera.main.ScreenPointToRay(Input.mousePosition);  // Create a RaycastHit variable to store information about what was hit by the ray.  RaycastHit floorHit;  // Perform the raycast and if it hits something on the floor layer...  if (Physics.Raycast(camRay, out floorHit, camRayLength, floorMask))  {  // Create a vector from the player to the point on the floor the raycast from the mouse hit.  Vector3 playerToMouse = floorHit.point - transform.position;  // Ensure the vector is entirely along the floor plane.  playerToMouse.y = 0f;  // Create a quaternion (rotation) based on looking down the vector from the player to the mouse.  Quaternion newRotation = Quaternion.LookRotation(playerToMouse);  // Set the player's rotation to this new rotation.  playerRigidbody.MoveRotation(newRotation);  }  }  void Animating(float h, float v)  {  // Create a boolean that is true if either of the input axes is non-zero.  bool walking = h != 0f || v != 0f;  // Tell the animator whether or not the player is walking.  anim.SetBool("IsWalking", walking);  }  } |

2、camera的移动

在游戏界面中，需要游戏相机和人物同步移动，这也需要脚本文件（.cs）控制其运动，给患者增强现实之感。其运行代码和注释如下：

|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class CameraFollow : MonoBehaviour  {  public Transform target; // The position that that camera will be following.  public float smoothing = 5f; // The speed with which the camera will be following.  Vector3 offset; // The initial offset from the target.  void Start()  {  // Calculate the initial offset.  offset = transform.position - target.position;  }  void FixedUpdate()  {  // Create a postion the camera is aiming for based on the offset from the target.  Vector3 targetCamPos = target.position + offset;  // Smoothly interpolate between the camera's current position and it's target position.  transform.position = Vector3.Lerp(transform.position, targetCamPos, smoothing \* Time.deltaTime);  }  } |

3、进度条的设置

进度条告知患者离目标运动距离还有多远，给患者直观的目标提示，增强其康复决心，是一种UI（User Interface，用户界面）设置。

在unity界面中的window下拉菜单中新建一个画布（Canvas），将空画布新建子对象（child）并命名为DistanceUI，并调整其尺寸以适应游戏界面。再在DistanceUI中创建DistanceSlider，根据游戏需要调整其设置后进入脚本文件控制进度条。脚本文件代码如下：

|  |
| --- |
| using UnityEngine;  using UnityEngine.UI;  using System.Collections;  using UnityEngine.SceneManagement;  public class PlayerHealth : MonoBehaviour  {  public int startingHealth = 100;  public int currentHealth;  public Slider healthSlider;  public Image damageImage;  public AudioClip deathClip;  public float flashSpeed = 5f;  public Color flashColour = new Color(1f, 0f, 0f, 0.1f);  Animator anim;  AudioSource playerAudio;  PlayerMovement playerMovement;  PlayerShooting playerShooting;  bool isDead;  bool damaged;  void Awake ()  {  anim = GetComponent <Animator> ();  playerAudio = GetComponent <AudioSource> ();  playerMovement = GetComponent <PlayerMovement> ();  playerShooting = GetComponentInChildren <PlayerShooting> ();  currentHealth = startingHealth;  }  void Update ()  {  if(damaged)  {  damageImage.color = flashColour;  }  else  {  damageImage.color = Color.Lerp (damageImage.color, Color.clear, flashSpeed \* Time.deltaTime);  }  damaged = false;  }  public void TakeDamage (int amount)  {  damaged = true;  currentHealth -= amount;  healthSlider.value = currentHealth;  playerAudio.Play ();  if(currentHealth <= 0 && !isDead)  {  Death ();  }  }  void Death ()  {  isDead = true;  playerShooting.DisableEffects ();  anim.SetTrigger ("Die");  playerAudio.clip = deathClip;  playerAudio.Play ();  playerMovement.enabled = false;  playerShooting.enabled = false;  }  public void RestartLevel ()  {  SceneManager.LoadScene(0);  }  } |



游戏成品图