VR 工程应用拓展训练报告

1 引言

1.1 研究背景与意义

1.1.1 数字孪生在智能制造中的价值（实时监控、预测性维护）

1.1.2 ESP32 与 Unity 在低成本、高灵活度机械臂控制中的优势

1.2 实验目标

1.2.1 实现机械臂物理实体与 Unity 虚拟模型的实时双向同步

1.2.2 验证虚-实协调控制的精度与可靠性

2 系统设计与实现

2.1 硬件系统构建

2.1.1 机械臂结构

基于本实验所使用舵机型号 MG996R，在模型库（bambulab.cn/zh-cn）中选择合适的机械臂样品进行 3D 打印。

2.1.2 电路设计

实验采用 ESP32 舵机驱动电路、串口/蓝牙双模通信模块（由授课教师提供）

2.2 软件系统开发

实验采用 Unity 搭建控制系统，进行三维建模和调整，在教师提供的参考代码的基础上做一定的修改，编写 C#脚本实现对舵机的控制

3 实验原理和过程

3.1 硬件测试

将提供的示例代码传输至单片机及舵机电路中，进行舵机转动测试，确认硬件无故障。实验中，系统编译环境存在一定问题，示例代码运行异常。经过一段时间的 debug 之后成功解决开发环境问题，并成功运行代码，单片机串口通信功能正常，上位机数据交互正常。

3.2 机械臂结构设计

3.2.1 机械臂模型

在模型库（bambulab.cn/zh-cn）中选择合适的机械臂样品进行 3D 打印和拼装，最终产品为单自由度机械臂（如下图所示），结构相对简单，方便控制

3.2.2 机械臂运动模型

3.2.2.1 参数定义

L = 机械臂长度（单位：mm）

θ = 舵机旋转角度（单位：度）

(x_0, y_0) = 舵机旋转中心坐标

t = 时间（单位：s）

3.2.2.2 运动方程

a 位置方程：

$$\begin{cases} x(t) = x_0 + L \cdot \cos(\theta(t) \cdot \dfrac{\pi}{180}) \\ y(t) = y_0 + L \cdot \sin(\theta(t) \cdot \dfrac{\pi}{180}) \end{cases}$$

b 速度方程：

$$\begin{cases} v_x(t) = -L \cdot \dot{\theta}(t) \cdot \dfrac{\pi}{180} \cdot \sin(\theta(t) \cdot \dfrac{\pi}{180}) \\ v_y(t) = L \cdot \dot{\theta}(t) \cdot \dfrac{\pi}{180} \cdot \cos(\theta(t) \cdot \dfrac{\pi}{180}) \end{cases}$$

3.3 Unity 控制程序开发

3.3.1 功能模块实现

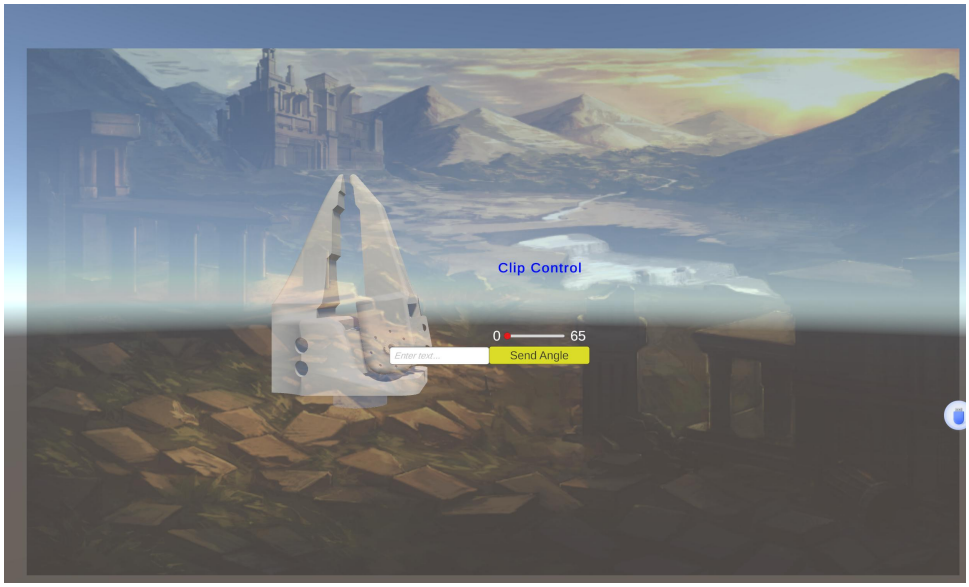
在 Unity 中创建一个新项目，导入机械臂模型，通过编写 C#脚本实现 Unity 与串口通信，使用串口监听返回数据，实现人机交互。

3.3.2 控制功能实现

借助示例代码和 AI 协作，创建 UI 文本输入框和滑块组件，实现舵机角度的双重控制。

3.3.3 UI 程序完善

设计控制程序 UI，搭建完整的控制程序平台，控制界面如下图所示。



3.4 示例代码与实际使用代码比较

3.4.1 通信协议

示例：

```
// 位置：SendAngle方法 (Line 80-94)
string command = "$"+angle+"#"; // 简单报头报尾
serialPort.WriteLine(command);
```

实际：

```
// 位置：ProcessData方法 (Line 154-186)
if (data.Contains("#")) {
    int endIndex=data.IndexOf("#");
    string packet = data.Substring(0, endIndex + 1);
    if(packet.StartsWith("$")) {
        string temp = packet.TrimStart('$').TrimEnd('#');
        // 完整的数据校验流程
    }
}
```

示例代码仅添加简单标识符，实施代码增加完整数据包解析、处理数据截断问题、添加类型转换验证。

3.4.2 数据接受

示例：

```
// 位置：整个文件
// 缺少数据接收功能
```

实际：

```
// 位置：Start方法 (Line 47-49)
receiveThread=new Thread(ReciveData);
receiveThread.IsBackground = true;
receiveThread.Start();

// 位置：ReciveData方法 (Line 118-152)
while (serialPort.IsOpen) {
    string receiveData = serialPort.ReadExisting();
    Debug.Log("舵机实际角度："+ProcessData(receiveData));
}
```

3.4.3 控制方式

示例：

```
// 位置：onSendBtnClick方法 (Line 64-77)
if (int.TryParse(AngleInput.text, out int angleValue))
{
    SendAngle(angleValue);
}
```

实际：

```
// 位置：Start方法 (Line 55)
AngleSld.onValueChanged.AddListener(value => OnSliderValueChanged(value));

// 位置：OnSliderValueChanged方法 (Line 85-88)
void OnSliderValueChanged(float value) {
    SendAngle((int)value);
    AngleInput.text = value.ToString(); // 双向同步
}
```

3.4.4 下位机处理

示例：

```
// 位置：loop函数 (Line 26-44)
servoB.write(180); // 硬编码测试
delay(2000);
```

实际：

```
// 位置：loop函数 (Line 20-35)
if (Serial.available()) {
    command = Serial.readString();
    command.trim(); // 关键数据清洗
    angle = command.toInt();
    servoB.write(angle); // 动态响应

    // 位置：数据回传 (需添加)
    Serial.print("$");
    Serial.print(angle);
    Serial.println("#");
}
```

4 结论与展望

实验过程中，主要遇到的困难有（括号内为解决方案）：3D 打印装配误差（通过孔径与壁厚测试修正模型）、舵机控制抖动（优化角度渐变算法）、Unity-单片机通信延迟（提升串口波特率至 115200bps）以及双向控制同步冲突（引入状态锁机制）等，针对这些困难，制定了有效解决方案并予以实施。通过本次“虚实结合”系统开发（涵盖 3D 打印、硬件驱动、Unity 交互全流程），我深入掌握了 Unity UI 系统与物理引擎的协同控制逻辑及相关代码原理，并深刻认识到 AI 在代码设计与需求实现中的重要作用，极大拓展了对未知领域的认知。未来优化可探索多舵机协同算法增加自由度、Unity 内实现轨迹规划与自主学习、以及采用蓝牙/WIFI 提升灵活性。此次 VR 工程应用实践有效拓展了我的专业前沿视野，为后续学习与实践奠定了坚实基础。