# 基于 Python 和 UDP 协议的激光 AGV 模拟测试及 B/S 架构应用

尹鑫
（天津赛象云科技有限公司，天津　300392）

**摘要**：本文利用 Python 和 UDP 协议，模拟激光 AGV 测试，通过 B/S 架构实现了其在模拟环境下的导航、控制及通信。首先概述 AGV 概念与应用，然后详述了系统设计，涵盖数据处理、UDP 通信、路径规划及 B/S 架构搭建。接着，描述了系统的测试方案和测试结果，验证了系统的可行性和稳定性。最后，探讨了系统优化与未来研究方向。

**关键词**：激光 AGV；Python 编程；UDP 协议；B/S 架构；仿真测试
**中图分类号**：TQ330.493　　**文章编号**：1009-797X(2025)05-0019-04
**文献标识码**：B　　DOI:10.13520/j.cnki.rpte.2025.05.004

激光 AGV 是一种能够自主导航、避障并执行物料搬运任务的智能机器人，广泛应用于工厂、仓库等场景中，能够提高物流效率、降低人力成本。为了验证激光 AGV 系统的可行性和稳定性，在实际部署之前进行模拟测试十分必要。UDP 协议作为一种无连接的传输层协议，具有传输速度快、实时性好的特点，适用于对实时性要求较高的应用场景。本文基于 Python 编程语言和 UDP 协议，设计了一个激光 AGV 系统的模拟测试平台，并使用 B/S 架构实现了对该系统的远程控制与监测。

## 1　激光 AGV 系统设计与实现

### 1.1　数据处理

程序中通过 UDP 协议进行通信，使用 socket 模块实现。收发的数据采用字节流的形式，通过 struct 模块进行解析和打包。根据协议授权码、报文类型、通信序列号、命令码等字段，实现了命令的识别和解析。

### 1.2　UDP 通信协议实现

激光 AGV 系统中各个硬件设备之间需要进行实时通信，以便实现数据的传输和控制指令的发送（图 1）。我们选择 UDP 协议作为通信协议，因为 UDP 具有简单、高效的特点，非常适合实时通信场景。通过 Python 的 socket 库，我们实现了基于 UDP 的通信功能，确保了系统的实时性和稳定性。

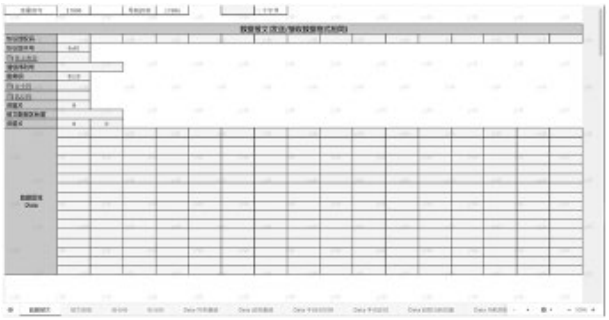

图 1　激光导航接口

### 1.3　路径规划

路径规划是激光 AGV 导航的关键步骤，程序中采用了贝塞尔曲线进行路径的生成。根据控制点的位置和路径长度，将路径等分成多个点，然后计算每个点的坐标和朝向角度，以实现平滑的运动轨迹。

### 1.4　B/S 架构搭建

为了实现对激光 AGV 系统的远程控制与监测（图 2），我们采用了 B/S 架构，即浏览器 / 服务器架构。通过 Java 的 Spark 框架搭建了服务器端，利用 HTML、CSS 和 JavaScript 编写了客户端的 Web 界面。用户可以通过浏览器访问该界面，实现对激光 AGV 系统的远程控制和监测。

**作者简介**：尹鑫（1986–），女，自动化测试工程师，副高级工程师，主要从事软件测试、自动化测试等方面的工作。

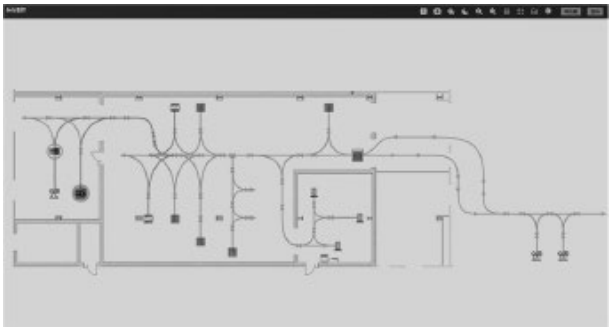

图 2　激光 AGV 运行监控系统

### 1.5　Python 代码实现细节

本文所使用的 Python 代码主要包括三个文件：LineUtils.py、readFile.py 和 AGV_simulation.py。其中，LineUtils.py 提供了计算直线点、贝塞尔曲线点和贝塞尔曲线弧度的方法；readFile.py 用于解析 XML 格式的地图数据；AGV_simulation.py 是主要的仿真测试程序。

#### 1.5.1　程序结构

AGV_simulation.py 是主程序，包括了激光 AGV 的模拟类 SimulatedAGV 和主函数 main()。

LineUtils.py 是一个工具类，提供了计算直线点、贝塞尔曲线点和贝塞尔曲线弧度的方法。这些方法可以用于在给定起点、终点和控制点的情况下生成一系列直线或贝塞尔曲线上的坐标点，以及计算这些点的弧度。它利用了数学公式来计算贝塞尔曲线上的点和弧度，以便在机器人导航系统中生成平滑的路径。

ReadFile.py 是一个读取 XML 文件并解析其中内容的类。它的主要功能是解析给定 XML 文件中的点、控制点和路径信息，并将其存储在相应的数据结构中。具体来说，它会提取 XML 文件中的点的名称、$x$ 和 $y$ 坐标，控制点的坐标，以及路径的长度和最大速度。这些数据被存储在字典中，以便后续的处理和使用。

#### 1.5.2　类与数据结构

SimulatedAGV 类：该类用于模拟 AGV 的行为。它包括了初始化方法 __init__() 用于初始化 AGV 的各项参数（包括 IP 地址、端口号、AGV ID 等）和网络连接。其中，points_data、control_points_data 和 path_data 分别存储了地图上的点坐标、控制点坐标和路线长度与最大速度等信息。在类的实例化过程中，会读取一个 XML 文件，解析其中的数据并传递给 __init__ 函数，以初始化 AGV。common_AGV_reply() 用于发送通用的 AGV 应答报文。listen_position() 用于监听 AGV 的位置信息。update_position() 用于更新 AGV 的位置信息。此外，还定义了一个内部的数据结构 Point 用于表示坐标点。

Point 类：这是一个命名元组，用于表示二维坐标点，包括 $x$ 坐标和 $y$ 坐标。

#### 1.5.3　方法详解

（1）初始化方法：在 __init__() 方法中，初始化了 AGV 的 IP 地址、端口号、ID 等信息，并创建了一个 UDP 套接字用于通信。同时，还初始化了一些路径信息和状态信息。

（2）发送应答报文方法：common_AGV_reply() 方法用于生成并发送通用的 AGV 应答报文。根据传入的参数，构造了不同类型的应答报文，并通过 UDP 协议发送给特定的地址。

（3）监听位置方法：listen_position() 方法是一个循环，持续监听来自 AGV 的位置信息。它解析收到的 UDP 数据包，并根据不同的命令码执行相应的操作（包括变量读写和导航控制）。其中，变量读写包括读变量、写变量；导航控制包括手自动切换、AGV 手动定位、获取 AGV 当前位置、导航控制、查询 AGV 运行状态 / 导航状态等。

下面对上述操作做详细说明。

读变量，目前已有的读变量包括告警信息、AGV 状态、AGV 叉臂高度、AGV 叉臂状态、电池状态。

写变量，目前已有的写变量包括设置叉臂高度、设置区域、服务器心跳、远程关机、吸合、重新下发导航、设定管控状态。

获取 AGV 当前位置，如果路径点 ID 为空，当前位置默认为 AGV 停车点。

导航控制，操作包括开始导航、取消导航、暂停导航、继续导航、创建导航任务并暂停导航 5 种。下面以“开始导航”为例进行说明，先获取指定路径的所有路径和路径点 ID，然后根据路径，读取 XML 文件，获取路径长度和最大速度。等分点的计算是通过路线长度 / 最大速度，先计算出运行时长。假设前端页面每隔 3 s 刷新一次，如果运行时长 <3 s，等分点固定为 1；否则，等分点 = 运行时长 /3。最后根据路径，获取控制点坐标，调用 LineUtils 工具类计算贝塞尔曲线上的点和弧度。

查询 AGV 运行状态，包括位置的 $X$ 坐标、位置的 $Y$ 坐标、位置的朝向角度等。

查询 AGV 导航状态，状态包括无到导航点任务、

等待、正在前往导航点、暂停、完成、失败、退出、等待开 / 关门操作 8 种。具体 AGV 导航状态，需要根据导航控制中的操作决定。下面以“正在前往导航点”和“完成”为例进行说明，如果导航控制中的操作是“继续导航”，进一步判断是否还有未经过的路径点，如果有，AGV 导航状态为“正在前往导航点”；如果没有，AGV 导航状态为“完成”。

（4）更新位置方法：update_position() 方法用于模拟 AGV 的运动过程。它通过不断地更新 AGV 的位置信息来模拟 AGV 在地图上的移动。在 AGV 开始导航时，根据指定的路径计算路线上的所有点的坐标，并按照预定的时间间隔更新当前位置。同时考虑了路线的等分点、控制点等因素，以及导航状态的变化。

#### 1.5.4 流程控制

整个项目的流程控制由 main 函数实现（图 3）。在 main 函数中，首先读取 XML 文件，解析其中的地图数据，并将解析得到的数据传递给 SimulatedAGV 类的实例化过程。然后，创建了多个线程分别监听多个不同端口的 AGV 的位置信息。每个线程会创建一个 SimulatedAGV 实例，并调用 listen_position 方法开始监听。这样，就实现了模拟多台 AGV 同时运行的效果。

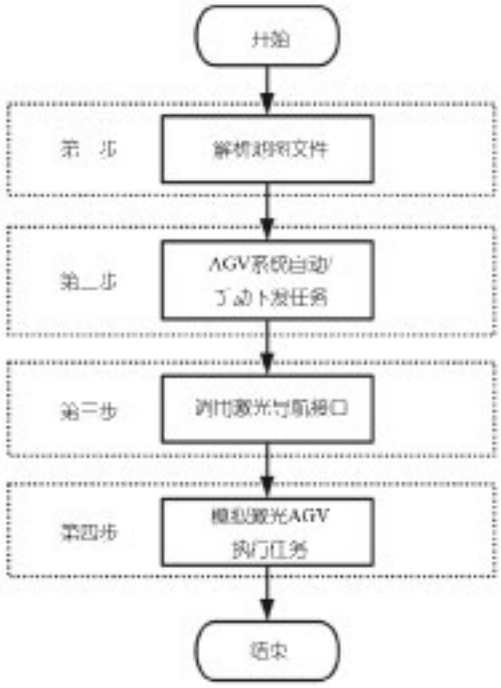


图 3　模拟激光 AGV 流程图

## 2　系统测试与结果分析

### 2.1　实验环境搭建

在本研究中，为了全面评估激光 AGV 系统的性能，我们精心搭建了一个模拟测试环境。该环境不仅模拟了实际生产场景中的多种复杂情况，还确保了测试结果的准确性和可靠性。该实验环境包括：

#### 2.1.1　硬件设备

本实验中，我们采用了一台高性能的计算机作为测试平台，其上安装了基于 Python 的模拟器软件。该模拟器能够精确模拟 AGV 的行驶轨迹、速度变化及避障等行为，并通过 UDP 协议与其他模拟设备进行通信。此外，我们还配置了必要的网络设备和接口，以确保数据传输的实时性和稳定性。

#### 2.1.2　软件依赖

软件层面，我们充分利用了 Python 的强大功能及其丰富的第三方库。具体而言，我们使用了 socket 库来实现网络通信，threading 库来支持多线程并发处理，struct 库来处理字节流数据的解析与封装，datetime 库来处理时间信息。此外，我们还开发了一系列自定义的工具类和模块，如地图解析、路线规划等，以辅助实现 AGV 的模拟控制。

#### 2.1.3　数据文件

为了模拟真实的生产环境，我们采用了 XML 格式的文件来存储地图的相关信息，包括点坐标、路线信息（包括控制点坐标、长度、最大速度等）、AMR 信息（包括读写变量 IP、读写变量端口、激光导航 IP、激光导航端口等）关键数据。

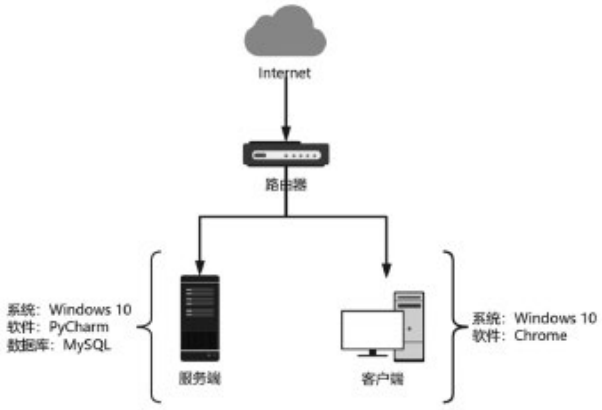


图 4　测试环境部署图

### 2.2　实验设计与实施

在实验设计阶段，我们根据实际需求制定了详细的测试计划，包括测试目标、测试方法、测试步骤以及预期结果等。为了确保测试结果的全面性和准确性，

我们设计了多个测试场景，涵盖了 AGV 在复杂环境中的行驶、避障、路径规划等多个方面。

在实验实施阶段，我们按照测试计划逐步进行操作。首先，我们读取并解析了 XML 文件中的地图信息，将其加载到模拟器中。然后，我们启动了多个线程来模拟多台 AGV 同时运行的情况，并通过 UDP 协议实现了它们之间的通信和协作。在测试过程中，我们记录了关键的运行数据和性能指标，如行驶速度、路径偏差、避障成功率等。

### 2.3 实验结果与分析

经过多次实验和测试，我们得到了丰富的实验数据和结果。以下是对实验结果的具体分析和总结：

#### 2.3.1 UDP 协议的可行性

实验结果表明，使用 UDP 协议进行激光 AGV 的模拟测试是完全可行的。UDP 协议具有传输速度快、实时性好的优点，能够满足 AGV 系统对数据传输的严格要求。同时，通过合理的错误处理和重传机制，我们可以有效地减少数据丢失和传输延迟对测试结果的影响。

#### 2.3.2 性能评估与优化

通过模拟测试，我们全面评估了激光 AGV 的性能表现。实验数据显示，AGV 在多数测试场景中能够稳定地行驶并准确地完成路径规划任务。然而，在部分复杂场景中，如障碍物密集或路径狭窄的情况下，AGV 的避障能力和路径规划效率仍有待提高。针对这些问题，我们提出了相应的优化策略和改进措施，如优化避障算法、调整路径规划参数等。

#### 2.3.3 实时性与稳定性

实验还验证了模拟系统的实时性和稳定性。在多次测试中，系统能够实时地接收和处理 AGV 发送的位置信息，并根据这些信息对 AGV 进行精确的控制。同时，系统能够稳定运行较长时间而不出现崩溃或异常现象，确保了测试结果的可靠性和准确性。

## 3 结论与展望

本文基于 Python 和 UDP 协议，设计了一个模拟激光 AGV 系统的测试平台，并使用 B/S 架构实现了对该系统的远程控制与监测。系统测试结果表明，该平台具有良好的可行性和稳定性，能够准确地模拟多台 AGV 同时运行的情况，并全面评估其性能表现，可以作为激光 AGV 系统实际部署之前的模拟测试工具。未来，我们将进一步优化系统的性能，提高 AGV 的避障能力和路径规划效率。同时，我们还将探索更多的应用场景和测试方法，以更好地满足实际生产的需求。最终，我们期望将该系统应用于实际生产中，为工业自动化的发展做出更大的贡献。

# Laser AGV simulation testing and B/S architecture application based on Python and UDP protocols

Yin Xin

(Tianjin Saixiang Cloud Technology Co. LTD., Tianjin 300392, China)

**Abstract:** This article uses Python programming language and UDP protocol to simulate and test laser AGV, and implements its navigation, control, and communication functions in the simulation environment through B/S (browser/server) architecture. Firstly, this article briefly introduces the concept and application areas of AGV (Automated Guided Vehicle). Next, the design scheme of the system was elaborated in detail, including data processing mechanism, application of UDP communication protocol, path planning method, and construction of B/S architecture. Subsequently, the article described the testing plan and results of the system, verifying its feasibility and stability. Finally, this article explores the optimization direction of the system and future research directions.

**Keywords:** laser AGV; Python programming; UDP protocol; B/S architecture; simulation test

(R-03)