



# 数字孪生的编程应用

同济大学中德工程学院



Chinesisch-Deutsche Hochschule für Angewandte Wissenschaften (CDHAW) an der Tongji-Universität  
Sino-German College of Applied Sciences at Tongji University



# 目录

1

工业编程逻辑

2

数字孪生技术架构

3

相关软件及学习指南

4

Q&A



01  
01



## 智能工厂的技术架构



## 传统编程和AI编程的区别

### Traditional Programming

Traditional programming is a manual process—meaning a person (programmer) creates the program. But without anyone programming the logic, one has to manually formulate or code rules.



### Machine Learning

Unlike traditional programming, machine learning is an automated process. It can increase the value of your embedded analytics in many areas, including data prep, natural language interfaces, automatic outlier detection, recommendations, and causality and significance detection. All of these features help speed user insights and reduce decision bias.





## AI在工业领域的应用限制?

### 普通人工智能

试错导向性

发散性和机会导向

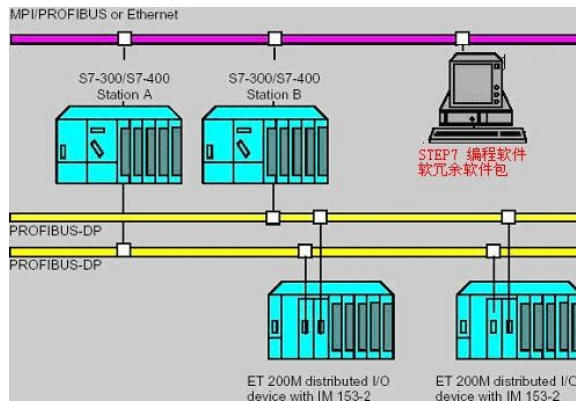


智能推荐系统

### 工业人工智能

系统性、快速性、可传承性

收敛性和效率导向



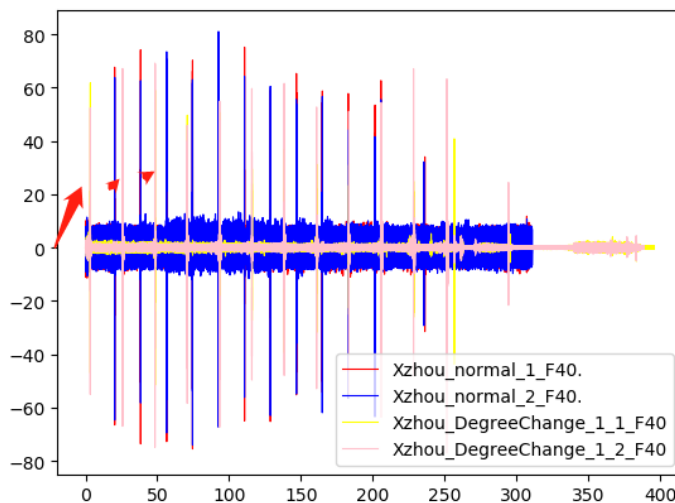
冗余系统



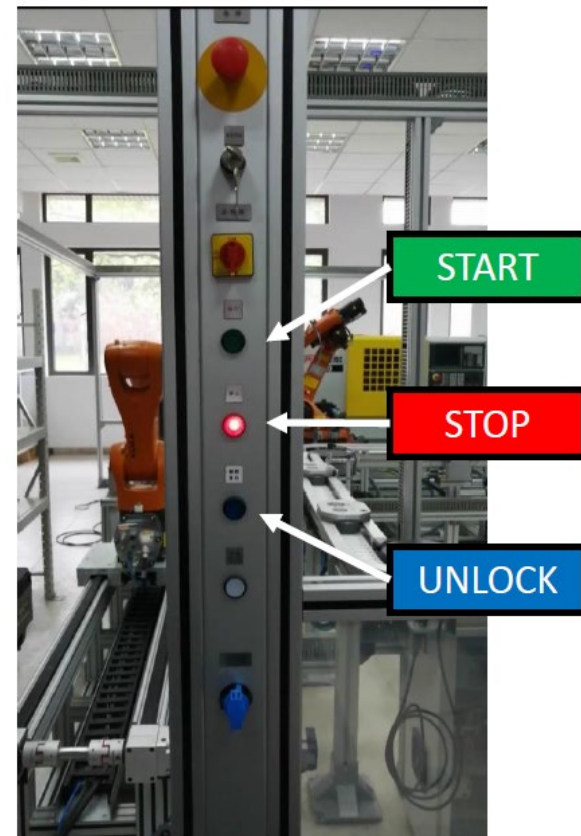
## 常见的工业人工智能应用



工业相机



预测性维护

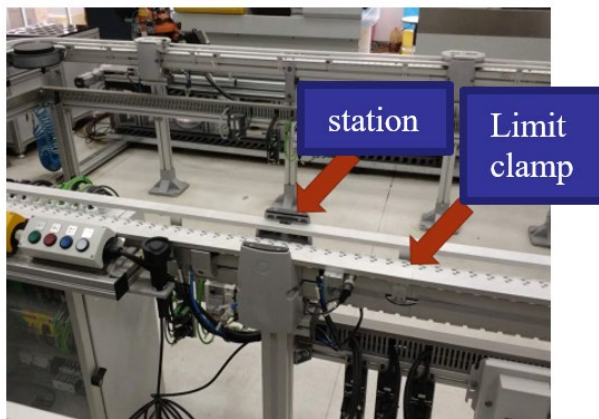
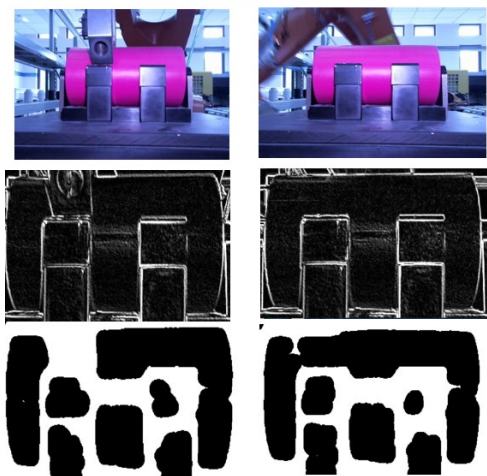


自动化编程与测试

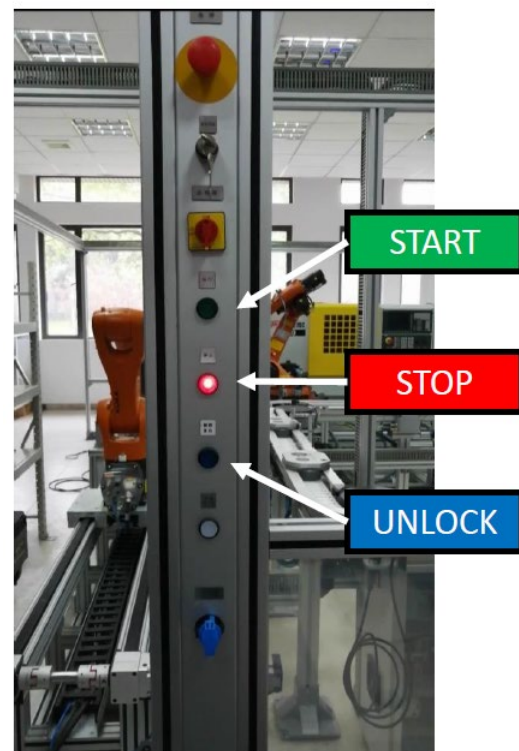




# 往届的案例

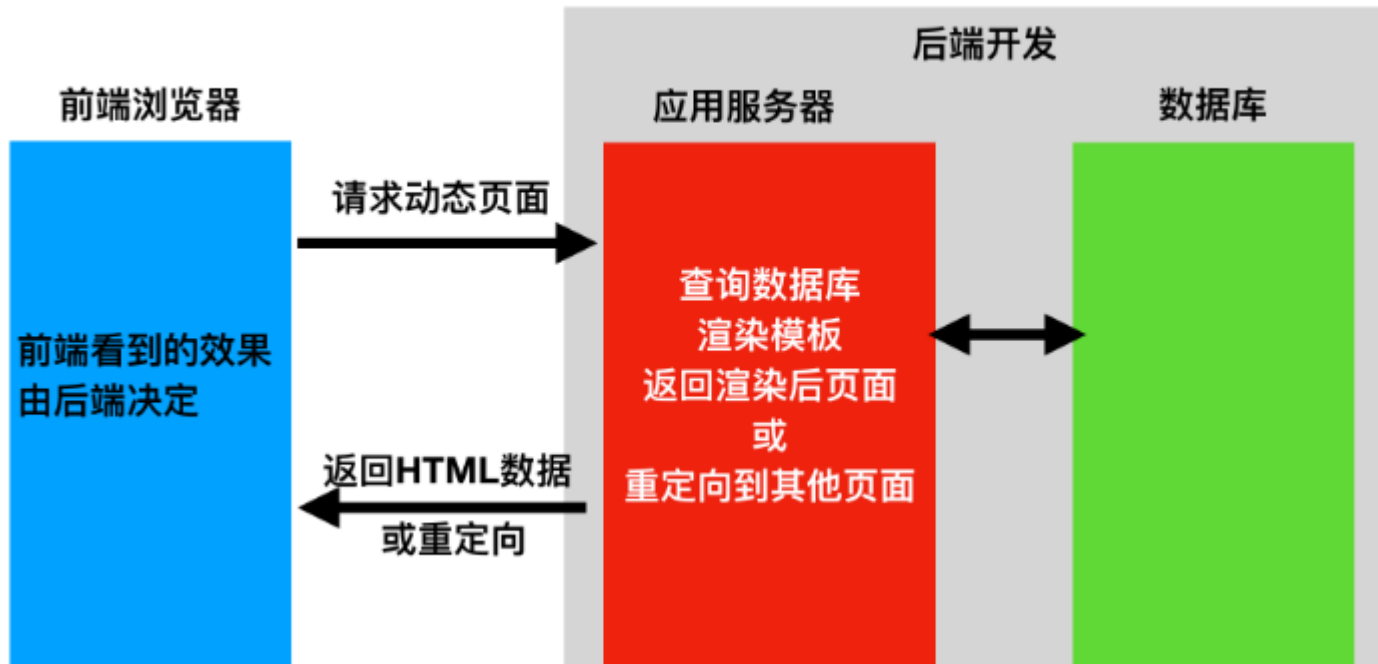


*The assembly line*





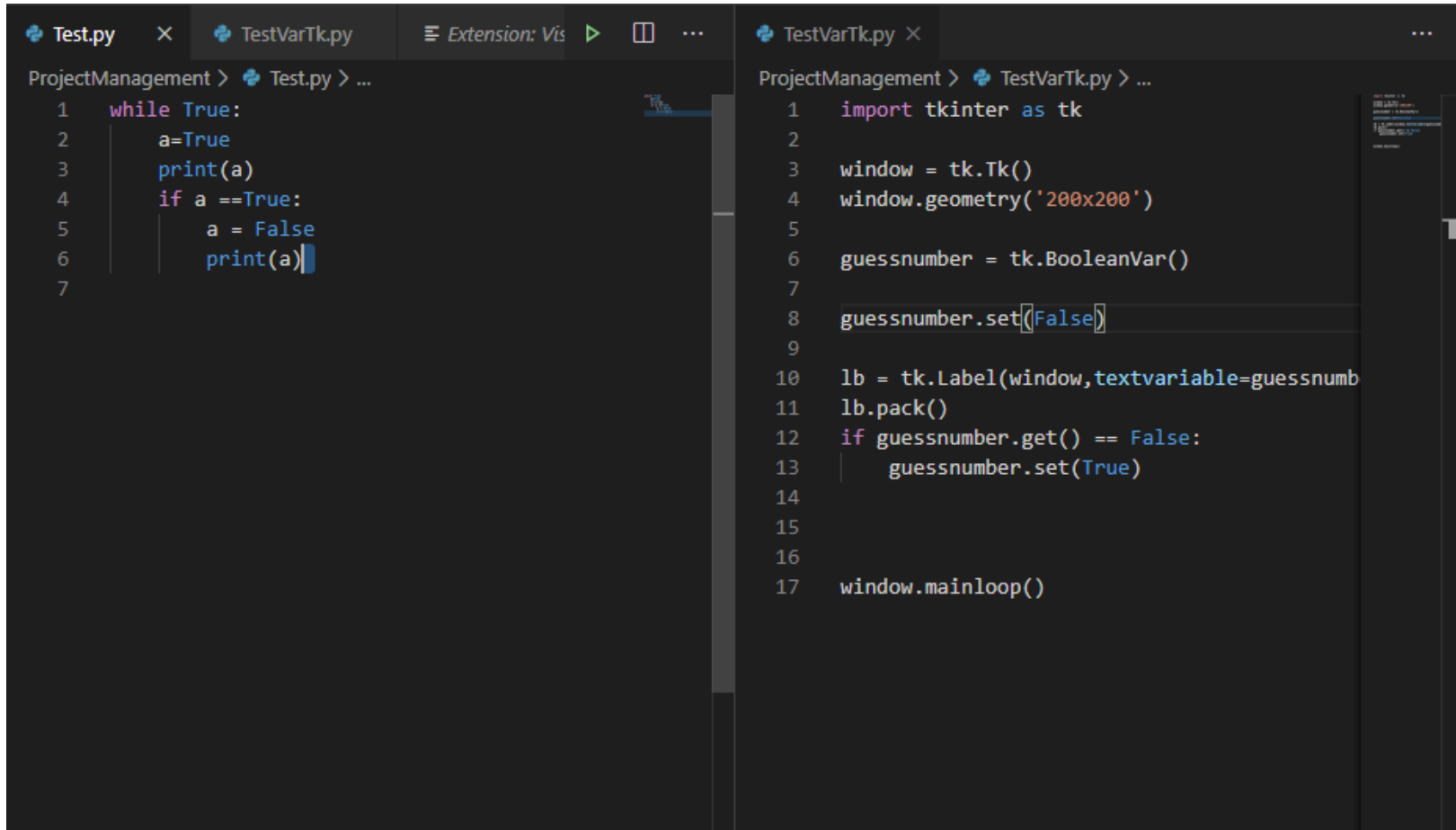
# 常见的用户交互程序架构







# GUI程序与命行程序的的不同



```
Test.py
1 while True:
2     a=True
3     print(a)
4     if a ==True:
5         a = False
6         print(a)
7

TestVarTk.py
1 import tkinter as tk
2
3 window = tk.Tk()
4 window.geometry('200x200')
5
6 guessnumber = tk.BooleanVar()
7
8 guessnumber.set(False)
9
10 lb = tk.Label(window,textvariable=guessnumb
11 lb.pack()
12 if guessnumber.get() == False:
13     guessnumber.set(True)
14
15
16
17 window.mainloop()
```



# 工业编程与命令行编程

工业编程语法IEC 61131-3 与高级语言的不同

while True:

    a=True

    print(a)

    if a ==True:

        a = False

        print(a)

a := True;

IF a= True Then

    a:=False;

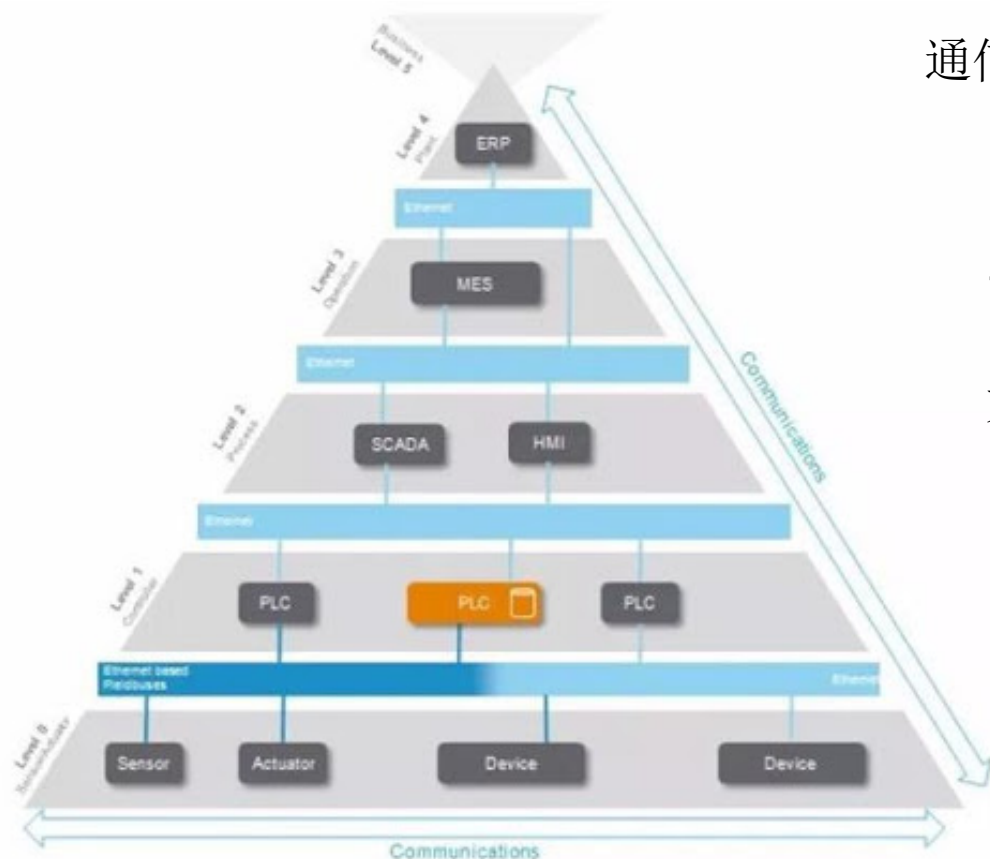
End\_IF



02  
05

## ▶ 数字孪生系统架构

# 工业控制金字塔



控制层级

通信时延

1分钟30次来回的往复运动

如果将通信的时延由20ms 降为5ms



# 工业控制金字塔

前导码	帧前定界码	目的地址	源地址	长度或类型	数据域	填充段(n字节)	CRC
7字节	1字节	6字节	6字节	2字节	46~1500字节		4字节

DSAP	SSSP	Ctrl	Org code	类型	数据
AA	AA	03	00	2字节	38 ~ 1492字节
1字节	1字节	1字节	3字节		

## TCP/IP数据帧

56 Bits	8 Bits	6 Byte	6 Byte	2 Byte	2 Byte	2 Byte	2 Byte	40 --- 1440 Bytes	2 Byte	1 Byte	1 Byte	4 Byte
Preamble	SYNCH	Dest Addr	Src Addr	Tag	Tag Control	Type 8892H	Frame ID	User data	Cycle Counter	Date Status	Transfer Status	FCS

← APDU-Status →

Tag(optional)			
Type 8100	Priority	0	VLAN-ID
2 byte	3 bit	1 bit	12 bit

**VLAN-Tag**  
Acc.to 802.1q  
(Usage is appl. specific)

## Profinet IO 数据帧



# Profinet 实时通信机理

- ✓ 通过软件的方法来完成实时通道的功能
- ✓ 去除一些协议层，减少文本长度；
- ✓ 提高通信双方传输数据的确定性，把数据传输准备就绪的时间减至最小；  
(Ether Type和Frame ID)
- ✓ 采用IEEE802.1q标准，增加对数据流传输优先级处理环节。(VLAN Tag)
- ✓ PROFINET把实现RT功能的VLAN标志嵌入到了以太网的帧结构中，VLAN由4字节组成,其中有表示优先级的3位。在RT帧中有两个最重要的协议元素，一个是以太网类型(EtherType),PROFINET使用以太网类型的Ox8892表示该帧是RT帧,该类型是由IEEE指定的区别于其他协议的惟一标准;另外一个帧ID码(FrameID),它用来编址两个设备间的特殊的通信通道。仅使用FrameID就可以快速选择和识别RT帧而不需要任何多余的帧头标志。

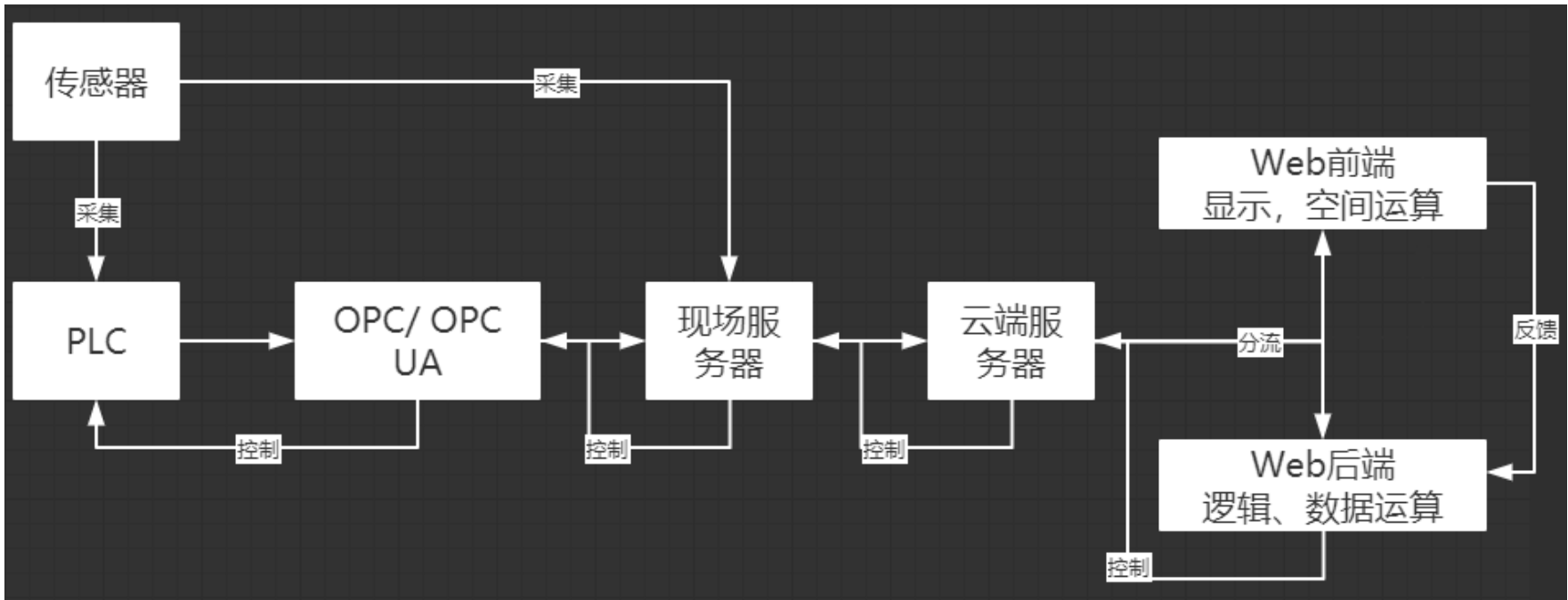
重点：减少识别帧头大小，加快数据帧的识别速度。





# 问题及解决方案

上位机与PLC的通信通常不能通过工业以太网协议来完成。

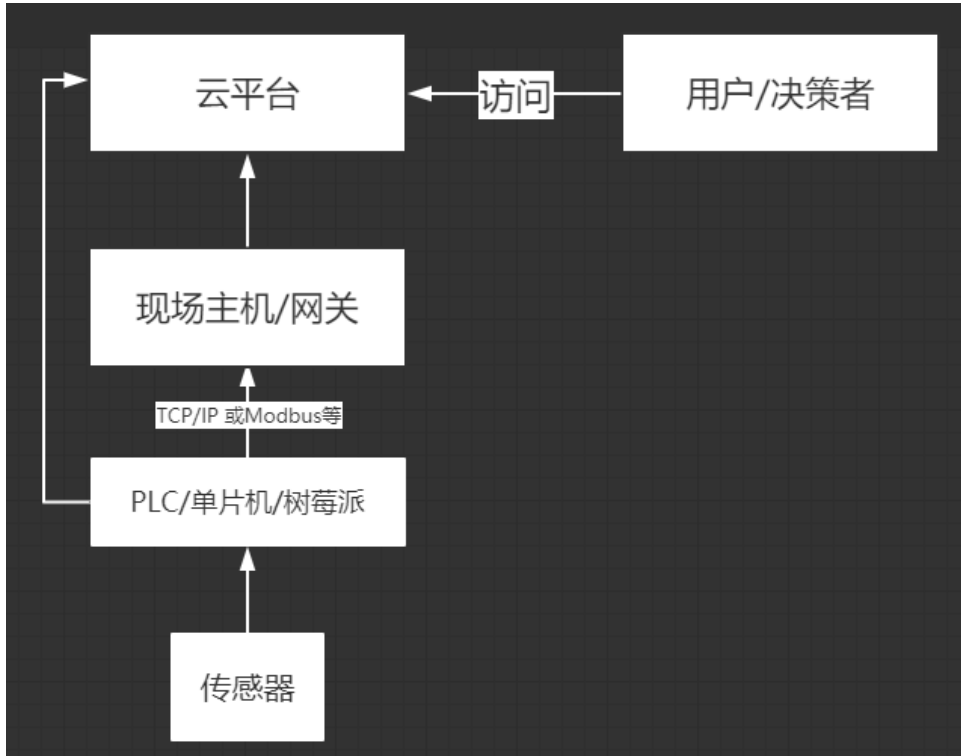


方案一：通过OPC/OPC UA建立上位机与PLC的连接



# 问题及解决方案

上位机与PLC的通信通常不能通过工业以太网协议来完成。



方案二：将上位机作为控制设备，使用TCP/IP、Modbus等协议通信

# TCP/IP UDP方案

```

UDPTestServer.py ×  UDPTestClient.py
ProjectManagement > UDPTestServer.py
1  #UDP_server.py
2  #coding=utf-8
3  #UDP 协议服务器代码
4  import socket
5
6  s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
7  s.bind(('192.168.1.250',6000))
8  while True:
9      data,addr=s.recvfrom(1024)
10     print(data.decode('utf-8'))
11

```

```

UDP_SOCKET1(ACTIVATE := InSock_Act TRUE, BIND_IP := InSock_BindIP '192.168.1.10', BIND_PORT := InSock_BindPort 5000,
HANDLE => OutSock_Handle 16#00000001, ACTIVE => OutSock_Act TRUE, BUSY => OutSock_Busy FALSE, ERROR => OutSock_Error FALSE,
STATUS => OutSock_Status 16#00000000, USED_PORT => OutSock_UsedPort 5000);

UDP_SEND1(REQ := InSend_Req TRUE, HANDLE := OutSock_Handle 16#00000001, DEST_IP := InSend_DestIP '192.168.1.250',
DEST_PORT := InSend_DestPort 6000, DATA_CNT := InSend_DataCNT 4,
DONE => OutSend_Done FALSE, BUSY => OutSend_Busy FALSE, ERROR => OutSend_Error FALSE, STATUS => OutSend_Status 16#0000,
DATA := InOutSend_Data 'hello');

```



# OPC UA 方案

```
from opcua import ua, Client
import time
import logging
import sys

logging.basicConfig(level=logging.INFO)
_logger = logging.getLogger('opcua')
dv_False = ua.DataValue(ua.Variant(False, ua.VariantType.Boolean))
dv_True = ua.DataValue(ua.Variant(True, ua.VariantType.Boolean))

if __name__ == "__main__":
    client = Client("opc.tcp://192.168.1.10:4840")
    client.set_user('admin')
    client.set_password('d0a904e3')
    client.set_security_string("Basic256Sha256,SignAndEncrypt,certificate-example.der,private-key-example.pem")

    try:
        client.connect()
        root = client.get_root_node()
        testvar = client.get_node("ns=5;s=Arp.Plc.Eclr/MainInstance.Web_MachineOn_Button")
        testvar.set_value(dv_True)
        # _logger.info('Objects node is: %r', root)
        #
        # objects = client.get_objects_node()
        # node = client.get_node("ns=5;s=Arp.Plc.Eclr/NewProgram1.INPUT")

        print(testvar.get_value())

    finally:
        client.disconnect()
```



03  
03

## ► 相关软件及学习指南



# Epic games 与 Unreal Engine

Unreal engine 4 为游戏引擎，是基于C++开发的大型模型加载、运算库

Epic games 中可以下载UE4的不同库版本，也可以在Market下载Unreal 的插件。

Unreal 本质是C++ 图形运算库文件，比WebGL在GPU利用率上性能更加强大，并且很多已经封装好的文件运行效率更高。

可以使用源码编译Unreal，也可以使用不同的IDE，或者使用Mac 以及Linux 系统。





# Unreal Engine 中的关键术语

关卡：场景关卡（**Level**）网格物体（**Static Mesh**），体积（**Volume**），灯光（**Light**）和蓝图（**Blueprint**）

**Actor**：**Actor**是支持三维转换（如平移、旋转和缩放）的泛型类。可通过游戏进程代码（**C++**或蓝图）创建（生成）及销毁**Actor**。

材质：物体的材料属性，包括颜色、透明度、金属光泽等。

蓝图：可以使用直观、基于节点的方式创建逻辑，或者设置一些变量数据。策划可以创建自定义的**Actor**、**Event**、函数等，蓝图也可以选择继承**C++**类，获取**C++**中定义的变量，调用**C++**中定义的函数，或者实现**C++**中定义的**event**。

**Widget**：用户交互界面，用于显示数据和操作，类似于人机交互界面



# Unreal Engine 中的模型处理

- 1、fbx文件导入
- 2、模型的调节（位置、原点）
- 3、物理特性设置（重力、移动特点、物理特性）
- 4、材质的创建与选择



# Unreal Engine 中的通信

- 1、以Unreal为核心搭建服务器
- 2、储存系统变量，用于显示
- 3、接收数据与发送数据
- 4、[https://blog.csdn.net/weixin\\_41738734/article/details/86221806](https://blog.csdn.net/weixin_41738734/article/details/86221806)



# Unreal Engine 中的碰撞预防

- 1、选定物理特征（Overlap 还是hit）
- 2、根据数据特征控制物体体积
- 3、提高画面帧率（设置、缩放）



**Q&A**

04  
04