Traction Control System (TCS) Simulation

PLCnext | Python 3.9 | OPC UA

Ali Çift
230702032

Project Purpose & Scope

The Challenge

Electric vehicles generate instant torque, leading to wheel slip on low-friction surfaces. Testing this on real hardware is dangerous and expensive.

The Solution

Design a **Digital** simulation where:

- **PLCnext** acts as the Electronic Control Unit.
- **Python** simulates the vehicle physics and environment.
- **OPC UA** provides real-time, secure communication.

Technology Stack

PLCnext Engineer

IEC 61131-3 Programming (Structured Text) & HMI Design.

Python 3.9

Physics engine, motor torque and friction simulation.

OPC UA & UaExpert

Secure industrial communication protocol and testing tool.

Signal Mapping (I/O)

Signal Name	Direction	Type	Description
HMI_GasPedal	PLC → Python	REAL (0-100)	Driver acceleration request.
HMI_BrakePedal	PLC → Python	REAL (0-100)	Braking force input.
HMI_RoadFriction	PLC → Python	REAL (0-100)	Environmental condition (0=Dry, 100=Ice).
Py_VehicleSpeed	Python → PLC	REAL	Simulated actual vehicle speed.
CMD_MotorTorque	PLC → Python	REAL	Final torque output after TCS intervention.

PLC Algorithm (Structured Text)

```
System_Active := (HMI_Start OR System_Active) AND NOT HMI_Stop;

temp_MaxTorque := SEL(System_Active, (0.0), (100.0));

    Kp := 10.0;



temp_TargetTorque := (HMI_GasPedal / 100.0) * temp_MaxTorque;

temp_SlipError := Py_WheelSpeed - Py_VehicleSpeed;
Monitor_Slip := temp_SlipError;

B:= (temp_SlipError > 5.0);

A := HMI_TCS_Active AND B;

Monitor_Status := A;

temp_Reduction := SEL(A, (0.0), (temp_SlipError * Kp));

CMD_MotorTorque := temp_TargetTorque - temp_Reduction;

CMD_MotorTorque1 := MAX(0.0, CMD_MotorTorque);

```

Traction Control Logic

The core logic detects the difference between wheel speed and vehicle speed.

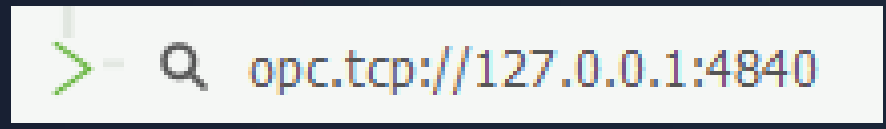
HMI Design


Motor Torque
0 10 20 30 40 50 60 70 80 90 100
Wheel Speed
0 20 40 60 80 100 120 140 160 180 200 220
100 90 80 70 60 50 40 30 20 10 0
100 90 80 70 60 50 40 30 20 10 0
0 20 40 60 80 100 120 140 160 180 200 220
km/h
100 90 80 70 60 50 40 30 20 10 0
TCS
Stop
Start
Slipperiness
Brake
Gas

OPC UA Expert

Secure Communication Tool

1. Server Setup: Created an OPC UA Server on the PLC to expose industrial data tags.
2. Python Connectivity: I developed a Python client to establish a secure connection with the PLC's endpoint.
3. Data Exchange: Enabled real-time data synchronization Python and PLCNext Engineer.



VALUE	DATATYPE
0	Float
0	Float
false	Boolean
false	Boolean
0	Float
0	Float
0	Float

Python Physics Engine

Connection with Python

1. Find the OPC UA server and enter.
2. Read the variable values coming from PLCNext.
3. Send back calculated values to HMI.

```
PLC_URL = "opc.tcp://127.0.0.1:4840"
PLC_USER = "admin"
PLC_PASS = "plcnext"
```

```
node_read_torque = client.get_node("ns=6;s=Arp.Plc.Eclr/MainInstance.CMD_MotorTorque")
```

```
node_write_veh_speed = client.get_node("ns=6;s=Arp.Plc.Eclr/Py_VehicleSpeed")
node_write_wheel_speed = client.get_node("ns=6;s=Arp.Plc.Eclr/Py_WheelSpeed")
```

Python Physics Engine

What Python Does?

Python calculates the "Real World" effects that the PLC cannot see.

1. **Air Drag:** Resistance increases with speed.
2. **Inertia:** Car mass delays acceleration.
3. **Road Friction:** If the road is icy, Python spins the wheels but keeps the car slow.

Test Results

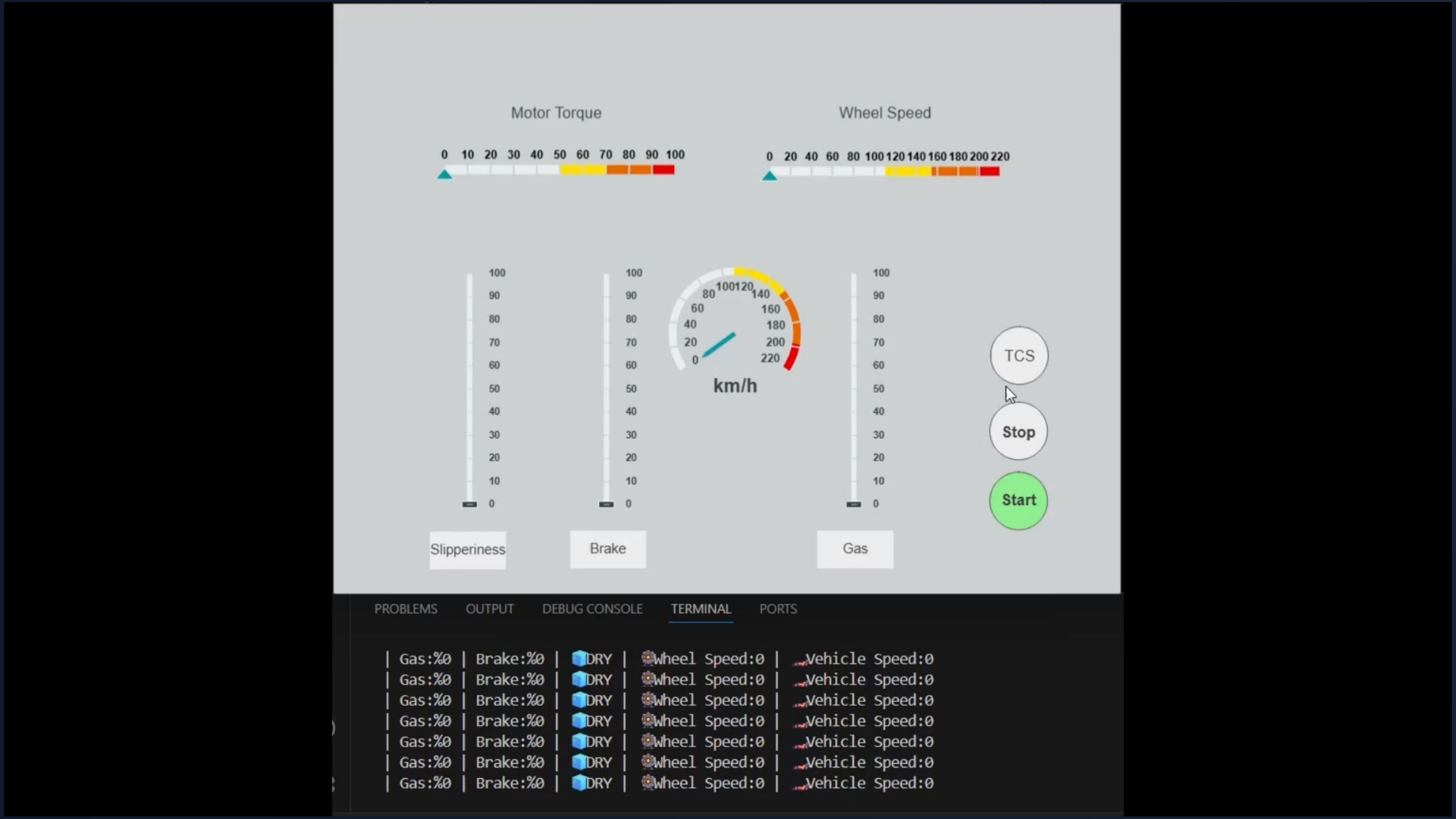
Scenario A: TCS off

1. Instant torque applies on wheels.
2. Result: Wheel speed is faster than vehicle speed. Wheels spin.

Scenario B: TCS on

1. TCS activates
2. Result: Cuts motor torque automatically. Wheel speed matches vehicle speed.

Test Results



Project Success

Seamless integration of Industrial Automation and Software Engineering.

- ✓ Full Digital Implementation
- ✓ Advanced Physics Modeling
- ✓ Secure Communication

Thank You!

QUESTIONS?