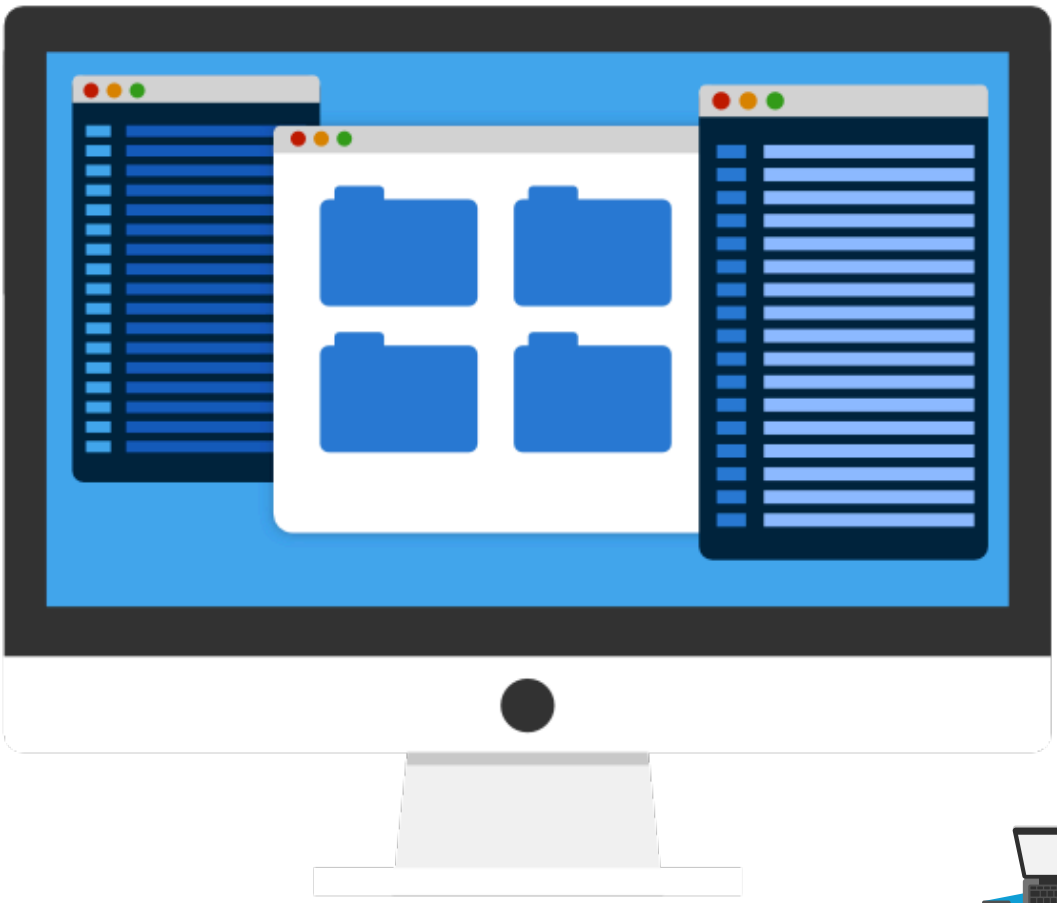


Ev Charging

Liu Yangfeng
Chen Yanan

Contenido

Arquitectura de proyecto

Comunicación entre los componentes

Comunicación por Sockets (TCP/IP)

Central ↔ Monitor

- Protocolo: Socket TCP/IP
- Puerto: Configurable (por defecto: 5000)
- Tipos de mensaje:
    - auth_request/auth_response - Autenticación
    - register_request/register_response - Registro
    - heartbeat_request/heartbeat_response - Latido (cada 30 segundos)
    - status_update - Actualización de estado
    - fault_notification - Notificación de fallo
    - start_charging_command - Comando para iniciar la carga
    - stop_charging_command - Comando para detener la carga
    - charging_data - Datos de carga (reenviados por el Monitor)

Monitor ↔ Engine

- Protocolo: Socket TCP/IP
- Puerto: Asignado dinámicamente (Engine escucha, Monitor se conecta) o configurar desde fichero *.env*
- Tipos de mensaje:
    - init_cp_id - Inicialización de CP_ID
    - health_check_request/health_check_response - Comprobación de estado (cada 30 segundos)
    - start_charging_command - Comando para iniciar la carga
    - stop_charging_command - Comando para detener la carga
    - charging_data - Datos de carga (Engine → Monitor)
    - charge_completion - Notificación de carga completada

Cola de Mensajes Kafka

Driver ↔ Central

- Tópicos (Topics):
    - driver_charge_requests - Solicitudes de carga del conductor
    - driver_stop_requests - Solicitudes para detener la carga
    - driver_cps_requests - Consultas de puntos de carga disponibles
    - driver_responses - Tópico de respuesta unificado (para todas las respuestas al Driver)

Engine → Central

- Topics:

- charging_session_data - Datos de la sesión de carga en tiempo real
- charging_session_complete - Notificación de sesión de carga completada

Descripción Detallada de los Componentes

Central (EV_Central)

Responsabilidades: Sistema de control central que coordina todos los componentes.

Funciones:

- Gestionar el registro y estado de los puntos de carga.
- Procesar las solicitudes de carga de los conductores.
- Gestionar las sesiones de carga.
- Guardar datos de forma permanente (SQLite).
- Proporcionar una interfaz de administración (AdminCLI).

Métodos de Comunicación:

- Servidor de Sockets (escucha conexiones del Monitor).
- Productor/Consumidor de Kafka (para comunicarse con Driver y Engine).

Monitor (EC_CP_M)

Responsabilidades: Monitorear el estado de los puntos de carga y servir de puente entre Central y Engine.

Funciones:

- Conectarse a Central y registrar los puntos de carga.
- Enviar latidos a Central (cada 30 segundos).
- Monitorear el estado del Engine (comprobación cada 30 segundos).
- Reenviar comandos de Central a Engine.
- Reenviar datos de Engine a Central.
- Detectar y reportar fallos.

Métodos de Comunicación:

- Cliente de Sockets (se conecta a Central).
- Cliente de Sockets (se conecta a Engine).

Engine (EV_CP_E)

Responsabilidades: Ejecutar la lógica de carga real.

Funciones:

- Recibir comandos de carga del Monitor.
- Ejecutar el proceso de carga (simulado).
- Enviar datos de carga en tiempo real (a través de Monitor y Kafka).
- Enviar notificaciones cuando la carga se completa.
- Proporcionar una CLI para simular operaciones del vehículo.

Métodos de Comunicación:

- Servidor de Sockets (espera la conexión del Monitor).
- Productor de Kafka (envía datos de carga a Central).

Driver (EV_Driver)

Responsabilidades: Aplicación cliente para el conductor.

Funciones:

- Consultar puntos de carga disponibles.
- Enviar solicitudes de carga.
- Ver el estado de la carga.
- Ver el historial de cargas.
- Enviar solicitudes para detener la carga.

Métodos de Comunicación:

- Productor/Consumidor de Kafka (para comunicarse con Central).

Pila Tecnológica (Tech Stack)

- Lenguaje de Programación: Python 3.8+
- Cola de Mensajes: Apache Kafka
- Base de Datos: SQLite

Protocolos de Comunicación:

- Socket TCP/IP (Central ↔ Monitor ↔ Engine)
- Cola de Mensajes Kafka (Driver ↔ Central, Engine → Central)
- Manejo de Concurrencia: Multihilo (Multi-threading)
- Framework de Interfaz: Rich (para interfaces de línea de comandos)

Diseño de Escalabilidad

Escalabilidad Horizontal: Se pueden iniciar múltiples instancias de Puntos de Carga (CP) y Conductores (Driver).

Diseño Desacoplado: El uso de Kafka permite una comunicación asíncrona, reduciendo la dependencia entre componentes.

Recuperación de Fallos:

- Mecanismo de latido (heartbeat) del Monitor (cada 30 segundos).
- Comprobación de estado del Engine (cada 30 segundos, con un tiempo de espera de 90 segundos).

Reconexión automática a través de un gestor de conexiones.

Gestión de Estado: Lógica clara de transición de estados para facilitar la recuperación ante fallos.

Despliegue

Requisito de entorno

Requisito de sistema

- Windows 10/11 o Linux
- Versión de python no inferior de 3.8

Requisito de software

1. Python 3.8+
2. Kafka
3. Docker

Instrucción de instalación de proyecto

Instalación de paquetes python

Dentro de la carpeta de proyecto ejecutar los siguientes comandos:

```
pip install -r requirements.txt
# o utilizar entorno virtual de python
python -m venv venv
# Windows:
venv\Scripts\activate
# Linux/macOS:
source venv/bin/activate
pip install -r requirements.txt
```

Listas de dependencias utilizada en el proyecto

```
colorama==0.4.6
colorlog==6.9.0
kafka_python==2.2.15
python-dotenv==1.2.1
rich==13.7.0
```

Instalación de kafka

```
# Arrancar Kafka con Docker
docker-compose up -d
# Estado de contenedores de Docker
docker-compose ps
# log de kafka
docker-compose logs -f broker
```

configuración de entorno utilizado en el proyecto(opcional)

Existe un fichero llamado *.env.copy dentro de la raíz de proyecto,* se debería de cambiar el nombre a *.env* para la ejecución(opcional).

```
# El fichero .env está en la raíz del proyecto
# .env ejemplo
# (true/false), true para muestrar log para desarrolladores y false para muestra información al usuario
DEBUG_MODE=False
# Dirección de Kafka Broker
BROKER_ADDRESS=localhost:9092
# Puerto de Central(se activa cuando DEBUG_MODE=True
```

```
LISTEN_PORT=5000
# Nombre de fichero de base de dato(sqlite)
DB_PATH=ev_central.db
# IP y puerto de Engine
IP_PORT_EV_CP_E=localhost:6000
# IP y puerto de Central
IP_PORT_EV_CP_CENTRAL=localhost:5000
# Tiempo máximo para simular una recarga de vehículo(expresado en segundo)
MAX_CHARGING_DURATION=30

```

Instrucciones de configuración

En este apartado se describe cómo configurar los argumentos del programa.

Argumentos de la línea de comandos

EV_Central

```
python Core/Central/EV_Central.py <listen_port> <broker_address>
# listen_port: Puerto de central
# broker_address: Dirección Kafka Broker
# ejemplo:
python Core/Central/EV_Central.py 5000 localhost:9092
```

EV_CP_E

```
python Charging_point/Engine/EV_CP_E.py <broker_address> [--debug_port PORT]
# broker_address: Dirección de Kafka Broker
# --debug_port: (opcional) Puerto de engine para que monitor pueda conectar a este puerto
# Ejemplo:
python Charging_point/Engine/EV_CP_E.py localhost:9092
python Charging_point/Engine/EV_CP_E.py localhost:9092 --debug_port 5003
Nota: En caso de que no definir --debug_port, Engine asigna un puerto de forma aleatoria y será demostrado por pantalla esa información
```

```
============================================================
  ENGINE LISTENING ON: localhost:3069
  Use this address to start Monitor:
  python EV_CP_M.py localhost:3069 <central_address:central port> <cp_id>
============================================================
```

EC_CP_M

```
python Charging_point/Monitor/EC_CP_M.py <engine_address> <central_address> <cp_id>
# engine_address: Dirección de Engine (IP:PORT)
# central_address: Dirección de Central(IP:PORT)
# cp_id: Identificador de punto de recarga(ejemplo: cp_001)
# ejemplo:
python Charging_point/Monitor/EC_CP_M.py localhost:5003 localhost:5000 cp_001
```

EV_Driver

```
python Driver/EV_Driver.py <broker_address> <driver_id>
# broker_address: Kafka Broker
# driver_id: identificador de conductor (ejemplo: driver_001)
# ejemplo:
python Driver/EV_Driver.py localhost:9092 driver_001
```

Desplegar

En este apartado, se asume que el desarrollador/usuario ejecuta todos los comandos en su único equipo, en caso de desplegar de forma separada, se debería de cambiar los argumentos de línea de comando de cada componente ajustado en su caso.

Para desarrollador

Crear fichero *.env* en la raíz del proyecto, se puede cambiar de nombre de .env.copy a *.env* o directamente crear una nueva con las siguientes variables de entornos.

```
1.    DEBUG_MODE=True
2.    BROKER_ADDRESS=localhost:9092
3.    LISTEN_PORT=5000
4.    DB_PATH=ev_central.db
5.    IP_PORT_EV_CP_E=localhost:6000
6.    IP_PORT_EV_CP_CENTRAL=localhost:5000
```

Arranque de sistema

Existe un fichero *.bat* situado dentro de **Common\tools\start_services_dev.bat**, este script te permite crear una central, un engine, un monitor y un driver. Para la ejecución se debe de comprobar previamente que Kafka está en marcha.

```
 1. @echo off
 2. chcp 65001 >nul
 3. title EV Charging System
 4. echo only available in developer environment(.env set debug_mode to true)
 5. echo [1/4] (EV Central)...
 6. start "EV Central" cmd /k "python Core\Central\EV_Central.py"
 7. timeout /t 3 /nobreak >nul
 8. echo [2/4] (Charging Point Engine)...
 9. start "CP Engine" cmd /k "python Charging_point\Engine\EV_CP_E.py"
10. timeout /t 2 /nobreak >nul
11. echo [3/4] (Charging Point Monitor)...
12. start "CP Monitor" cmd /k "python Charging_point\Monitor\EC_CP_M.py"
13. timeout /t 2 /nobreak >nul
14. echo [4/4] (EV Driver)...
15. start "EV Driver" cmd /k "python Driver\EV_Driver.py"
16. timeout /t 1 /nobreak >nul
```

Para usuario

No crear fichero *.env* o crear y poner DEBUG_MODE=False

```
1.    DEBUG_MODE=False
2.    BROKER_ADDRESS=localhost:9092
3.    LISTEN_PORT=5000
4.    DB_PATH=ev_central.db
5.    IP_PORT_EV_CP_E=localhost:6000
6.    IP_PORT_EV_CP_CENTRAL=localhost:5000
```

Arranque de sistema

Existe también otro script que te permite levantar una central, un engine, un monitor y un driver. Situado en **Common\tools\start_services_production.bat**

```
1. @echo off
2. chcp 65001 >nul
3. title EV Charging System
4. echo only available in production environment(.env set debug_mode to false)
```

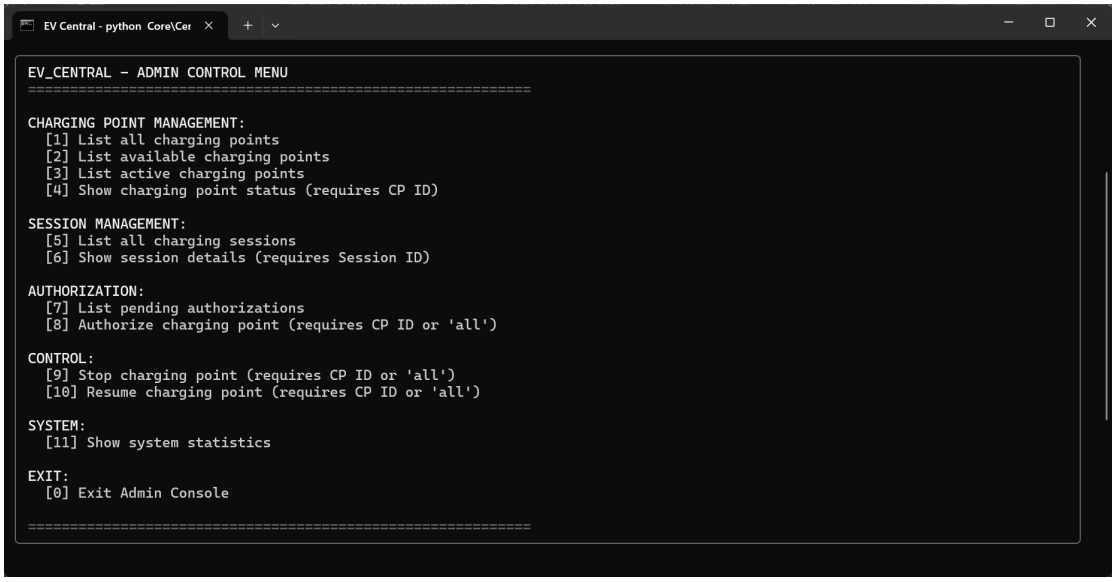
```
echo [1/4] (EV Central)...
start "EV Central" cmd /k "python Core\Central\EV_Central.py 5002 localhost:9092"
timeout /t 3 /nobreak >nul
echo [2/4] (Charging Point Engine)...
start "CP Engine" cmd /k "python Charging_point\Engine\EV_CP_E.py localhost:9092 --debug_port 5003"
timeout /t 2 /nobreak >nul
echo [3/4] (Charging Point Monitor)...
start "CP Monitor" cmd /k "python Charging_point\Monitor\EC_CP_M.py localhost:5003 localhost:5002 cp_001"
timeout /t 2 /nobreak >nul
echo [4/4] (EV Driver)...
start "EV Driver" cmd /k "python Driver\EV_Driver.py localhost:9092 driver_001"
timeout /t 1 /nobreak >nul
```

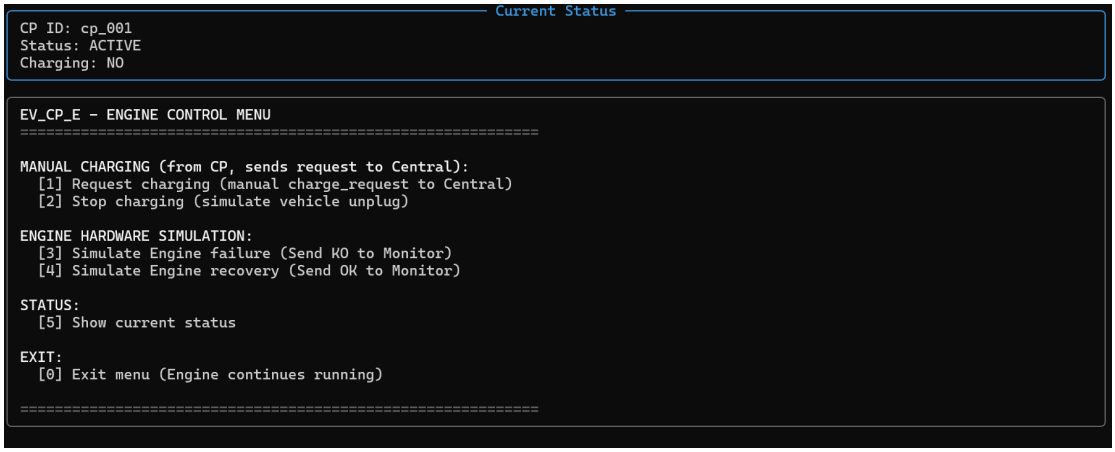
También se puede levantar múltiples puntos de recarga y conductores, están definido en **Common\tools\start_multi_driver.bat** y **Common\tools\start_multi_charging_points.bat**

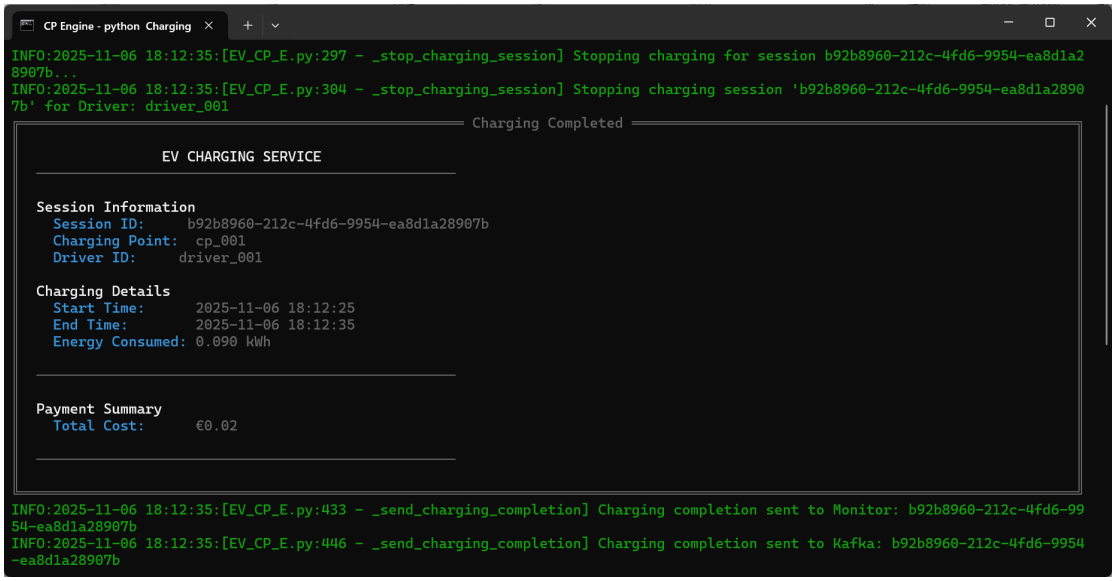
```
@echo off
setlocal enabledelayedexpansion
set /a loop_count=3
set /a start_port=5003
set start_id=cp_001
for /l %%i in (1,1,%loop_count%) do (
    set /a current_port=!start_port! + %%i - 1
    set current_id=!start_id:~0,-1!%%i
    start "EV_CP_E_%%i" cmd /k "python Charging_point\Engine\EV_CP_E.py localhost:9092 --debug_port !current_port!"
    timeout /t 1 >nul
    start "EC_CP_M_%%i" cmd /k "python Charging_point\Monitor\EC_CP_M.py localhost:!current_port! localhost:5002 !current_id!"
)
```

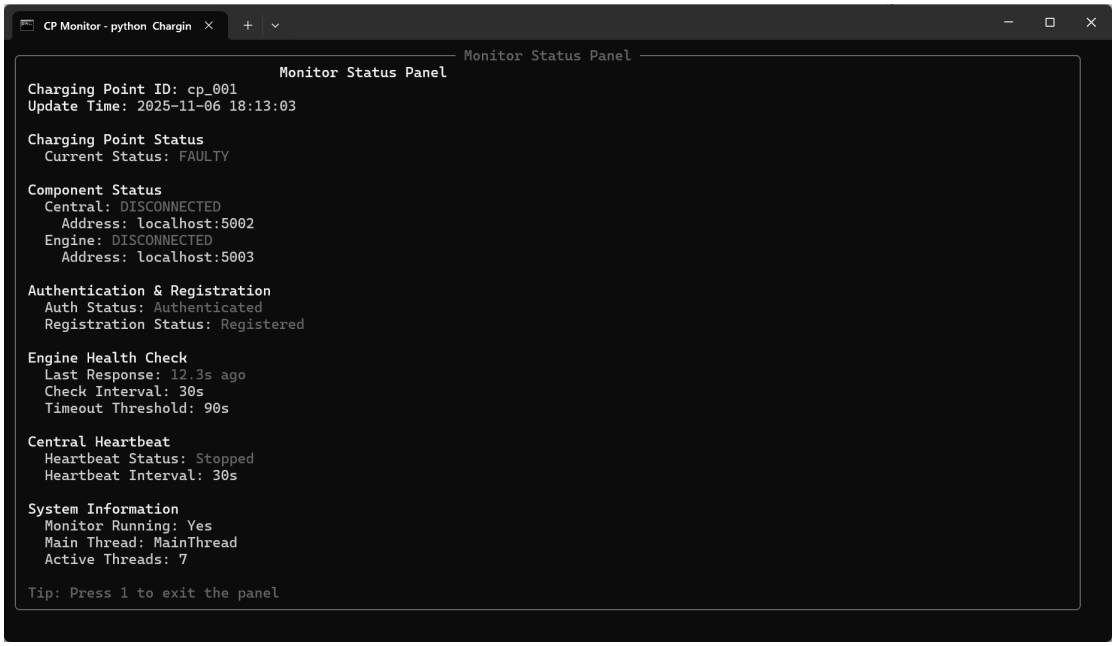
```
@echo off
setlocal enabledelayedexpansion
set loop_count=5
for /l %%i in (1,1,%loop_count%) do (
    set "driver_id=driver_00%%i"
    echo !driver_id!
    start "Driver_%%i" cmd /k "python Driver\EV_Driver.py localhost:9092 !driver_id!"
    timeout /t 1 >nul
)
```

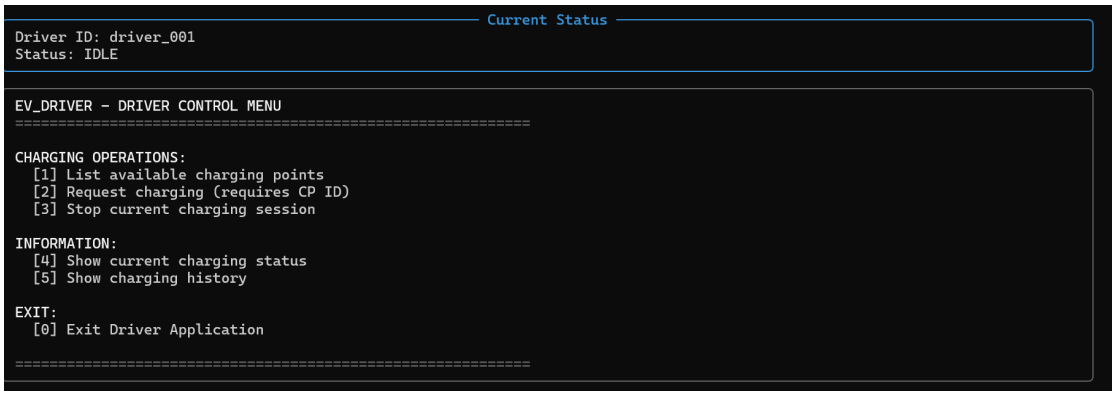
Algunas capturas de programa








CP Monitor - python Chargin
Monitor Status Panel
Monitor Status Panel
Charging Point ID: cp_001
Update Time: 2025-11-06 18:13:03
Charging Point Status
Current Status: FAULTY
Component Status
Central: DISCONNECTED
Address: localhost:5002
Engine: DISCONNECTED
Address: localhost:5003
Authentication & Registration
Auth Status: Authenticated
Registration Status: Registered
Engine Health Check
Last Response: 12.3s ago
Check Interval: 30s
Timeout Threshold: 90s
Central Heartbeat
Heartbeat Status: Stopped
Heartbeat Interval: 30s
System Information
Monitor Running: Yes
Main Thread: MainThread
Active Threads: 7
Tip: Press 1 to exit the panel


Current Status
Driver ID: driver_001
Status: IDLE
EV_DRIVER - DRIVER CONTROL MENU
CHARGING OPERATIONS:
[1] List available charging points
[2] Request charging (requires CP ID)
[3] Stop current charging session
INFORMATION:
[4] Show current charging status
[5] Show charging history
EXIT:
[0] Exit Driver Application