Introduction

This is the implementation of the OCPP-1.6 protocol

Tools

- Charge point implementation: Python
- CMS Websockets server: Python
- Platform: Windows

Description of the OCPP protocol

OCPP protocol is a set of rules used to negotiate the relationship/communication between EV charge points and central management systems. It describes how messages are packaged, server responses, error handling schemes, state transitions and the like.
This protocol allows EV chargers to have remote monitoring, marriage with any vendor supporting OCPP, load balancing, control and payment schemes, among others.

Approach used to implement

For implementation of this protocol, I mapped out the necessary functionality I need to implement. Listed below are the functions needed to have a full working OCPP stack:

- connection to CSMS
- BootNotification
- Heartbeat
- Authorization
- StartTransaction
- StatusNotification
- FirmwareUpdate

This implementation builds only the first 3 of these.

To connect to the CSMS via websockets, we need to have a websocket client and server. The intended client was in C using `libwebsockets`.

I implement the client in python using websockets and asyncio, and using the ocpp library.

Once a websocket connection is SUCCESS, the client(charge point) can send messages to the server. These PDUs(protocol data units) are formatted as JSON objects to conform with the **OCPP-JSON-1.6 Specification**.

Boot notification

The boot notification request looks like shown below:

```
{
        "call_type":    2,
        "id":   "msg_001",
        "type": "BootNotification",
        "properties":   {
                "charge_point_vendor":  "VENDOR",
                "charge_point_model":   "CP001",
                "charge_point_serial_number":   "CJ-12345ABCD",
                "charge_box_serial_number":     "CJ-ABCD12345",
                "firmware_version":     "1.1",
                "iccid":        "1",
                "imsi": "1",
                "meter_type":   "dual-socket",
                "meter_serial_number":  "CJ-ABC123ABC"
        }
}
```

The server response is expected to have the following structure:

```
[3, "12345", {
    "status": "Accepted",
    "currentTime": "2025-02-17T12:00:00Z",
    "interval": 300
}]
```

Heartbeat

The interval value in the above server response represents the heartbeat interval set by the CSMS.

On sending the boot notification, the server processes the payload and responds with the packet shown below:

!()[]

Challenges encountered

- Windows envronment setup using C - libsockets
- Time taken to understand and implement a clean working OCPP protocol in C

Key OCPP concepts learned

- OCPP flow from when an EVSE boots all the way to completing/closing a charging session
- How the EV hardware ecosystem work especially in charging and integration with OCPP protocol

Most relevant sections of the OCPP specifications

Having used the OCPP-JSON-1.6 protocol, The most relevant sections are:

- **Section 3.1 - Connection** - explains how a connection is made to the CMS via websockets
- **Section 3.2 - Server response** - How the server responds to charge point requests depending on the message type
- **Section 4 - Message and Message types** - when sending or receiving message from server, the message type is critical to be known e.g CALL, CALLRESULT and CALLERROR. This section was heavily used when constructin the JSON packets to send to server and when receiving the same packets as responses from the CMS.

Link to the code

The code for python implementation can be found here: https://github.com/bytecod3/OCPP-simulation/tree/main/ocpp-protocol-python (https://github.com/bytecod3/OCPP-simulation/tree/main/ocpp-protocol-python)

The code for the python CMS mockup can be found here: https://github.com/bytecod3/OCPP-simulation/tree/main/python-web-socket-server (https://github.com/bytecod3/OCPP-simulation/tree/main/python-web-socket-server)

For C, though in progress can be found here: https://github.com/bytecod3/OCPP-simulation/tree/main/ocpp-protocol/src (https://github.com/bytecod3/OCPP-simulation/tree/main/ocpp-protocol/src)