# **Rapport: Utveckling av Custom Component "Avancerad Elbilsladdning" för Home Assistant**

**Datum för denna sammanställning:** 2025-05-29

**Tidigare relevant dokument:** "Rapport: Utveckling av Custom Component 'Avancerad Elbilsladdning' för Home Assistant" daterat 2025-05-23.

**Syfte:** Att dokumentera nuvarande status, design, logik och utvecklingsprocess för den anpassade Home Assistant-integrationen "Avancerad Elbilsladdning". Denna rapport bygger vidare på och ersätter tidigare versioner för att ge en komplett bild av projektet och underlätta framtida utveckling och referens. Den reflekterar kodbasen och funktionaliteten upp till och med de ändringar som diskuterats kring version 0.1.35.

## **0. Versionshantering av Kodfiler**

För att säkerställa spårbarhet och underlätta felsökning samt utveckling har en princip om att alla kodfiler ska versionsmarkeras införts. Detta görs genom en kommentar i början av varje Python-fil (t.ex. coordinator.py, switch.py, etc.) med formatet:

# File version: ÅÅÅÅ-MM-DD <MANIFEST\_VERSION>

Där ÅÅÅÅ-MM-DD är datumet för den senaste signifikanta ändringen och <MANIFEST\_VERSION> är den övergripande versionen av integrationen som specificeras i manifest.json. Detta gör det enkelt att identifiera vilken specifik kodversion som diskuteras eller används. Den övergripande versionen i manifest.json uppdateras vid varje release.

## **1. Introduktion**

### **1.1. Syfte med Integrationen**

Målet med denna integration är att skapa en avancerad, flexibel och intelligent styrning för laddning av en elbil via en Easee-laddbox (eller potentiellt andra laddboxar med liknande styrningsmöjligheter) integrerad i Home Assistant. Integrationen ska möjliggöra automatisk laddning baserat på en kombination av flera dynamiska kriterier, primärt:

* **Elpris:** Aktuellt spotpris.
* **Tidsscheman:** Användardefinierade scheman för när laddning är tillåten eller prioriterad.
* **Solenergiöverskott:** Tillgänglig effekt från egna solpaneler efter att husets behov har täckts.
* **Bilens Laddningsnivå (SoC):** Möjlighet att sätta en övre gräns för hur mycket bilen ska laddas.

Utöver automatisk styrning ska integrationen ge användaren detaljerad kontroll över de smarta funktionerna och en god översikt över aktivt styrningsläge.

### **1.2. Övergripande Funktionalitet**

Integrationen tillhandahåller två huvudsakliga smarta laddningslägen som kan aktiveras oberoende av varandra, men med en inbördes prioritet, samt en övergripande laddningsgräns (SoC):

1. **Laddningsgräns (SoC - State of Charge) (Högst Prioritet):** Om konfigurerad, kommer denna funktion att förhindra all smart laddning om bilens aktuella laddningsnivå har nått eller överskridit den av användaren inställda gränsen.
2. **Pris/Tid-styrd Smartladdning (Andra Prioritet, om SoC tillåter):** Laddar bilen när användardefinierade tidsscheman är aktiva och det aktuella elpriset (endast spotpris) är under ett av användaren inställt maxvärde.
3. **Solenergiladdning (Tredje Prioritet, om SoC och Pris/Tid tillåter):** Om varken SoC-gränsen hindrar eller Pris/Tid-styrd laddning är aktiv (antingen för att dess villkor inte är uppfyllda eller för att den är avstängd), prioriterar detta läge laddning med överskott från egna solpaneler. Detta sker när ett specifikt schema för solenergiladdning är aktivt och det finns tillräckligt med överskott under en viss tid (SOLAR\_SURPLUS\_DELAY\_SECONDS).

För att möjliggöra detta skapar integrationen egna entiteter i Home Assistant (switchar för smarta lägen, nummerinmatningar för tröskelvärden som maxpris, solbuffert och SoC-gräns, samt en sensor för aktivt styrningsläge). Den hanterar även proaktiv kontrolltagning av laddaren vid anslutning av bil och styrning av laddarens huvudströmbrytare. Sensorer för sessionsenergi och sessionskostnad har tagits bort för att förenkla och undvika potentiella felkällor i beräkningarna.

## **2. Systemkrav och Externa Beroenden**

För att denna integration ska fungera korrekt krävs följande i användarens Home Assistant-miljö:

* En fungerande Home Assistant-installation.
* **Easee EV Charger-integrationen:** Korrekt installerad och konfigurerad.
* Nödvändiga externa sensorer och hjälpare (konfigureras via UI):
  + **Obligatoriska:**
    - CONF\_CHARGER\_DEVICE: Easee-laddarenheten (väljs från en lista över Easee-enheter).
    - CONF\_STATUS\_SENSOR: Sensor för laddarens status (t.ex. sensor.min\_laddbox\_status).
    - CONF\_CHARGER\_ENABLED\_SWITCH\_ID: Huvudswitch för laddboxen (t.ex. switch.min\_laddbox\_aktiverad).
    - CONF\_PRICE\_SENSOR: Sensor för aktuellt el-spotpris.
  + **Valfria (men rekommenderade för full funktionalitet):**
    - CONF\_SURCHARGE\_HELPER: Påslag (t.ex. nätavgift, moms). Kan vara en sensor eller input\_number. Används *endast* för kostnadsberäkning om sådan funktionalitet återinförs; påverkar f.n. inte laddningsbeslut.
    - CONF\_TIME\_SCHEDULE\_ENTITY: schedule-hjälpare för Pris/Tid-laddning.
    - CONF\_HOUSE\_POWER\_SENSOR: Sensor för husets totala effekt (W/kW). Nödvändig för solenergiladdning.
    - CONF\_SOLAR\_PRODUCTION\_SENSOR: Sensor för solproduktion (W/kW). Nödvändig för solenergiladdning.
    - CONF\_SOLAR\_SCHEDULE\_ENTITY: schedule-hjälpare för solenergiladdning.
    - CONF\_CHARGER\_MAX\_CURRENT\_LIMIT\_SENSOR: Sensor för laddboxens faktiska max strömbegränsning (A). Används för att bestämma maximal ström vid Pris/Tid-laddning. Faller tillbaka på MAX\_CHARGE\_CURRENT\_A\_HW\_DEFAULT (f.n. 16A) om ej angiven.
    - CONF\_EV\_POWER\_SENSOR: Sensor för elbilens aktuella laddeffekt (W/kW). Används f.n. inte aktivt i styrlogiken efter borttagning av sessionsenergi/kostnad, men kan vara användbar för framtida funktioner eller manuell övervakning.
    - CONF\_EV\_SOC\_SENSOR: Sensor för bilens aktuella laddningsnivå (%). Används för SoC-gränskontroll.
    - CONF\_TARGET\_SOC\_LIMIT: Numeriskt värde (%) för önskad maximal laddningsnivå. Används tillsammans med CONF\_EV\_SOC\_SENSOR.

## **3. Filstruktur för Custom Component**

Installeras under <HA\_config\_mapp>/custom\_components/smart\_ev\_charging/:

* manifest.json (Definierar integrationen, version, beroenden etc.)
* const.py (Konstanter som används inom integrationen)
* \_\_init\_\_.py (Huvudfil för integrationens uppsättning och avlastning)
* config\_flow.py (Hanterar konfigurationsflödet via UI)
* coordinator.py (Hjärnan i integrationen, innehåller SmartEVChargingCoordinator)
* switch.py (Definierar switch-entiteter)
* number.py (Definierar number-entiteter)
* sensor.py (Definierar sensor-entiteter)
* translations/sv.json (Svenska texter för UI)
* HELP.md (Användarmanual)
* README.md (Översikt och versionshistorik)

## **4. Konfiguration av Integrationen (via UI)**

Konfigurering sker helt via Home Assistants användargränssnitt. Både initial uppsättning (async\_step\_user i config\_flow.py) och senare ändringar via "Alternativ" (SmartEVChargingOptionsFlowHandler i config\_flow.py) hanteras.

Användaren väljer de externa sensorer och hjälpare som listats ovan. Fält markerade med asterisk (\*) i UI:t är obligatoriska vid initial konfiguration. Valfria fält kan lämnas tomma. Om en valfri sensor som är nödvändig för en specifik funktion (t.ex. solsensorer för solenergiladdning) inte anges, kommer den funktionen inte att vara fullt operativ och varningar kan loggas.

Ett alternativ för CONF\_DEBUG\_LOGGING finns också för att aktivera detaljerad loggning för felsökning.

## **5. Entiteter Skapade av Integrationen**

Integrationen skapar följande entiteter automatiskt, baserat på konfigurationspostens unika ID:

* **Switch (...\_smart\_charging\_enabled)**: "Avancerad Elbilsladdning Smart Laddning Aktiv"
  + Slår PÅ/AV den pris/tid-styrda smartladdningen.
* **Switch (...\_solar\_surplus\_charging\_enabled)**: "Avancerad Elbilsladdning Aktivera Solenergiladdning"
  + Slår PÅ/AV laddning med solenergiöverskott. (Notera att suffixet kan vara solar\_charging\_enabled beroende på exakt konstant i const.py)
* **Number (...\_max\_charging\_price)**: "Avancerad Elbilsladdning Max Elpris"
  + Ställer in det maximala spotpriset (kr/kWh) för Pris/Tid-laddning.
* **Number (...\_solar\_charging\_buffer)**: "Avancerad Elbilsladdning Solenergi Buffer"
  + Ställer in en buffert (Watt) som reserveras för husets behov innan solenergi används.
* **Number (...\_min\_solar\_charging\_current)**: "Avancerad Elbilsladdning Minsta Laddström Solenergi"
  + Ställer in den minsta strömstyrka (Ampere) för solenergiladdning. (Notera att suffixet kan vara min\_solar\_charge\_current\_a beroende på konstant)
* **Sensor (...\_active\_control\_mode)**: "Avancerad Elbilsladdning Aktivt Styrningsläge"
  + Visar det faktiska styrningsläget som för närvarande är aktivt och kontrollerar laddningen: "PRIS\_TID", "SOLENERGI" eller "AV".

*Sensorer för sessionsenergi och sessionskostnad har tagits bort.*

## **6. Kärnlogik och Styrningsstrategier (SmartEVChargingCoordinator)**

All styrningslogik hanteras av SmartEVChargingCoordinator-klassen i coordinator.py. Den ärver från DataUpdateCoordinator och uppdateras periodiskt baserat på CONF\_SCAN\_INTERVAL samt reagerar på tillståndsförändringar från externa entiteter via lyssnare.

### **6.1. Initialisering (\_\_init\_\_)**

* Koordinatorn tar emot hass, entry (ConfigEntry), och scan\_interval\_seconds.
* self.config skapas genom att slå samman entry.data och entry.options, där options har företräde. Detta säkerställer att de senaste inställningarna från användaren alltid används.
* Diverse interna tillståndsvariabler initieras, t.ex. active\_control\_mode, should\_charge\_flag, target\_charge\_current\_a, session\_start\_time\_utc.
* \_setup\_listeners() anropas för att börja lyssna på relevanta externa entiteter.

### **6.2. Hantering av Entiteter och Lyssnare**

* **\_resolve\_internal\_entities()**: Denna metod ansvarar för att hämta entity\_id för de switchar och nummer-entiteter som denna integration själv skapar. Detta är nödvändigt för att koordinatorn ska kunna läsa deras tillstånd. Metoden anropas vid behov, t.ex. under \_async\_first\_refresh och i början av \_async\_update\_data om entiteterna inte redan är lösta.
* **\_setup\_listeners()**: Sätter upp asynkrona lyssnare (async\_track\_state\_change\_event) för alla relevanta *externa* entiteter som konfigurerats av användaren (t.ex. laddarstatus, elpris, solproduktion, scheman, SoC). När någon av dessa entiteter ändrar tillstånd, anropas \_handle\_external\_state\_change.
* **\_handle\_external\_state\_change()**: Denna callback-metod tar emot händelsen och begär en omedelbar uppdatering av koordinatorn via self.async\_request\_refresh(). Detta säkerställer snabb reaktion på förändringar i omvärlden.
* **\_remove\_listeners() och cleanup()**: Används för att korrekt ta bort alla lyssnare när integrationen avlastas eller Home Assistant stängs ner, för att förhindra resursläckor.

### **6.3. Första Uppdatering (\_\_init\_\_.py och coordinator.py)**

* I \_\_init\_\_.py (under async\_setup\_entry) skapas koordinatorn. Direkt därefter anropas await coordinator.async\_config\_entry\_first\_refresh(). Detta tvingar DataUpdateCoordinator att köra \_async\_first\_refresh (om definierad) eller direkt \_async\_update\_data en första gång.
* I coordinator.py finns ingen explicit \_async\_first\_refresh längre, men \_resolve\_internal\_entities() anropas i början av \_async\_update\_data om det behövs, och en initial fördröjning (INITIAL\_SETUP\_DELAY\_SECONDS) kan implementeras i koordinatorns \_\_init\_\_ eller \_async\_update\_data vid första körningen för att låta andra entiteter initialiseras.

### **6.4. Huvudlogik (\_async\_update\_data)**

Detta är hjärtat i koordinatorn och körs periodiskt samt vid begäran från lyssnare.

1. **Läs in senaste konfiguration:** self.config = self.entry.data | self.entry.options.
2. **Lös interna entiteter:** Anropar self.\_resolve\_internal\_entities() om det inte redan är gjort.
3. **Hämta tillstånd:** Läser av aktuella tillstånd från alla relevanta externa sensorer (laddarstatus, elpris, SoC, solproduktion, husförbrukning, scheman etc.) och interna entiteter (smart-switchar, maxpris-nummer etc.).
4. **Huvudströmbrytare:** Om ett smartläge är aktivt och laddarens huvudströmbrytare (CONF\_CHARGER\_ENABLED\_SWITCH\_ID) är AV, slås den PÅ.
5. **Beslutslogik (prioriterad ordning):**
   * **SoC-gräns:** Om bilens SoC (CONF\_EV\_SOC\_SENSOR) är >= CONF\_TARGET\_SOC\_LIMIT, sätts should\_charge\_flag = False. All vidare utvärdering för laddning avbryts.
   * **Pris/Tid-styrd laddning:**
     + Villkor: smart\_charging\_enabled-switchen PÅ, CONF\_TIME\_SCHEDULE\_ENTITY aktiv (om konfigurerad), och aktuellt spotpris (CONF\_PRICE\_SENSOR) <= inställt maxpris (max\_charging\_price-nummer).
     + Om uppfyllt: active\_control\_mode\_internal = CONTROL\_MODE\_PRICE\_TIME, should\_charge\_flag = True, target\_charge\_current\_a sätts till laddarens hårdvarumaximum (från CONF\_CHARGER\_MAX\_CURRENT\_LIMIT\_SENSOR eller default).
   * **Solenergiladdning:**
     + Körs endast om Pris/Tid-laddning *inte* är aktiv och should\_charge\_flag fortfarande är False.
     + Villkor: solar\_surplus\_charging\_enabled-switchen PÅ, CONF\_SOLAR\_SCHEDULE\_ENTITY aktiv (om konfigurerad).
     + Beräkna tillgängligt solöverskott: Solproduktion (CONF\_SOLAR\_PRODUCTION\_SENSOR) - Husförbrukning (CONF\_HOUSE\_POWER\_SENSOR) - Solenergi Buffer (solar\_charging\_buffer-nummer). En dämpningsmekanism kan användas för att undvika att bilens egen laddeffekt felaktigt räknas med i husförbrukningen om en solsession redan pågår.
     + Om tillräckligt överskott finns för att driva laddaren med minst min\_solar\_charging\_current-nummer (och minst MIN\_CHARGE\_CURRENT\_A), och detta överskott har varit stabilt under SOLAR\_SURPLUS\_DELAY\_SECONDS (hanteras via \_solar\_surplus\_start\_time och \_solar\_session\_active):
       - active\_control\_mode\_internal = CONTROL\_MODE\_SOLAR\_SURPLUS, should\_charge\_flag = True.
       - target\_charge\_current\_a beräknas baserat på överskottet, avrundas nedåt, och begränsas av laddarens hårdvarumaximum.
     + Om villkoren inte uppfylls (t.ex. för lite överskott, fördröjning inte uppnådd), sätts should\_charge\_flag = False och relevanta solenergi-tillstånd återställs.
   * **Ingen smart styrning:** Om inga smarta switchar är aktiva, sätts active\_control\_mode\_internal = CONTROL\_MODE\_MANUAL och should\_charge\_flag = False. Laddaren lämnas då till manuell styrning eller Easee-appens scheman.
6. **Styr laddaren:** Metoden \_control\_charger() anropas med should\_charge\_flag och target\_charge\_current\_a.
7. **Uppdatera sensor för aktivt styrningsläge:** self.active\_control\_mode (som matar sensorn ...\_active\_control\_mode) sätts till self.active\_control\_mode\_internal om laddning faktiskt pågår och styrs av integrationen, annars till CONTROL\_MODE\_MANUAL ("AV").
8. **Sessionsdata:** self.session\_start\_time\_utc återställs när en ny laddningssession initieras av integrationen eller när en pågående session avslutas (t.ex. laddaren kopplas från, SoC-gräns nås, eller smart styrning stängs av). Beräkning av energi och kostnad har tagits bort.

### **6.5. Styrning av Laddare (\_control\_charger)**

Denna metod tar emot beslutet från \_async\_update\_data och utför de faktiska tjänsteanropen till Easee-integrationen:

* Om should\_charge\_flag är True:
  + Anropar easee.set\_dynamic\_charger\_circuit\_current (eller motsvarande tjänst från const.py, t.ex. EASEE\_SERVICE\_SET\_DYNAMIC\_CURRENT) för att sätta den beräknade target\_charge\_current\_a. Detta görs oavsett om strömmen redan är den önskade (för "PRIS\_TID") för att säkerställa korrekt inställning. För "SOLENERGI" kan en kontroll finnas för att undvika onödiga anrop om strömmen är oförändrad.
  + Om laddaren inte redan laddar (status är t.ex. ready\_to\_charge, paused), anropas easee.resume\_charging (eller EASEE\_SERVICE\_RESUME\_CHARGING).
* Om should\_charge\_flag är False:
  + Om laddaren aktivt laddar, anropas easee.pause\_charging (eller EASEE\_SERVICE\_PAUSE\_CHARGING).
* Hanterar även att slå av huvudströmbrytaren för laddboxen om alla smarta funktioner är avstängda men huvudströmbrytaren fortfarande är på.

### **6.6. Hjälpfunktioner**

* **\_get\_number\_value(), \_get\_spot\_price\_in\_kr(), \_get\_power\_value()**: Används för att säkert hämta och tolka värden från konfigurerade sensorer och hjälpare, med felhantering och normalisering av enheter.
* **\_reset\_session\_data()**: Återställer session\_start\_time\_utc.
* Andra interna hjälpfunktioner kan finnas för specifik logik.

## **7. Kända Problem / Fortsatt Utveckling (per 2025-05-29)**

* **Problem med Options Flow (Rensade fält):** Ett kvarstående problem är att när ett tidigare ifyllt entitetsfält rensas i Alternativ-flödet ("Konfigurera"), så sparas inte ändringen korrekt. Istället behålls det gamla värdet. Detaljerad loggning av user\_input (den data som tas emot från Home Assistants frontend) har visat att frontend skickar det gamla värdet till backend istället för ett tomt värde som indikerar att fältet rensats. Detta problem ligger därmed med hög sannolikhet i hur Home Assistants frontend hanterar detta scenario och är svårt att åtgärda enbart i denna integrations backend-kod.
* **Fas-hantering:** Behöver fortfarande verifieras fullt ut för att säkerställa korrekt strömberäkning och kommandon för 1-fas vs 3-fas laddning, särskilt för solenergiladdningens effekt-till-ström-omvandling. (Från tidigare rapport, relevans bör utvärderas).
* **Stabilitet kring awaiting\_start:** Även om state listenern nu fokuserar på ready\_to\_charge, kan awaiting\_start fortfarande vara ett känsligt läge. (Från tidigare rapport, relevans bör utvärderas).
* **Tröskel för Solproduktions-listener:** Överväg en konfigurerbar tröskel eller "debounce"-logik för att minska antalet uppdateringar om solproduktionssensorn är "brusig". (Från tidigare rapport, relevans bör utvärderas).
* **Hantering av "Ej hittade entiteter":** Loggningen för ej hittade/otillgängliga entiteter har förbättrats för att minska loggspam, men ytterligare användarfeedback (t.ex. reparationsvarning i HA) kan övervägas.

## **8. Bilaga: Kodfiler**

Alla kodfiler (\_\_init\_\_.py, config\_flow.py, const.py, coordinator.py, number.py, sensor.py, switch.py, translations/sv.json, manifest.json, HELP.md, README.md) utgör tillsammans integrationen och bör hanteras med versionskontroll (t.ex. Git).

Denna rapport syftar till att ge en aktuell och detaljerad översikt av "Avancerad Elbilsladdning". Den bör uppdateras kontinuerligt i takt med att integrationen utvecklas vidare.