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

**Datum för denna sammanställning:** 2025-05-23

**Tidigare dokument:** "Rapport: Utveckling av Custom Component 'Avancerad Elbilsladdning' för Home Assistant" daterat 2025-05-19.

**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.

## **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.X

Där ÅÅÅÅ-MM-DD är datumet för den senaste signifikanta ändringen och X är ett löpnummer för den dagen. Detta gör det enkelt att identifiera vilken specifik kodversion som diskuteras eller används.

## **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 laddningssessioner, kostnader och 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.

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 sensorer för sessionsdata och status). Den hanterar även proaktiv kontrolltagning av laddaren vid anslutning av bil och styrning av laddarens huvudströmbrytare.

## **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.
    - CONF\_STATUS\_SENSOR: Sensor för laddarens status (t.ex. sensor.uppfart\_status).
    - CONF\_CHARGER\_ENABLED\_SWITCH\_ID: Huvudswitch för laddboxen (t.ex. switch.uppfart\_laddboxen\_aktiverad).
    - CONF\_PRICE\_SENSOR: Sensor för aktuellt el-spotpris (integrationen försöker tolka enheter som öre/kWh, SEK/kWh, EUR/kWh eller /MWh-varianter och omvandlar internt till kr/kWh eller motsvarande basvaluta).
  + **Valfria (men rekommenderade för full funktionalitet):**
    - CONF\_SURCHARGE\_HELPER: Påslag (t.ex. nätavgift, moms). Kan vara en sensor eller input\_number. Förväntad enhet är kr/kWh eller öre/kWh (om entiteten har det specificerat). Används endast för kostnadsberäkning. Om utelämnad, antas påslag vara 0.
    - 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 max strömbegränsning (A).
    - CONF\_EV\_POWER\_SENSOR: Sensor för elbilens aktuella laddeffekt (W/kW). Används för sessionsdata.
    - 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
* const.py
* \_\_init\_\_.py
* config\_flow.py
* coordinator.py (Hjärnan i integrationen)
* switch.py
* number.py
* sensor.py

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

Konfigurering sker via Home Assistants gränssnitt. Både initial setup och senare ändringar (Options Flow) hanteras. Användaren väljer de externa sensorer och hjälpare som listats ovan. Fält för CONF\_SURCHARGE\_HELPER, CONF\_EV\_SOC\_SENSOR, CONF\_TARGET\_SOC\_LIMIT och scheman är valfria. 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.

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

* **Switch (...\_smart\_charging\_enabled):** "Avancerad Elbilsladdning Smart Laddning Aktiv"
* **Switch (...\_solar\_charging\_enabled):** "Avancerad Elbilsladdning Aktivera Solenergiladdning"
* **Number (...\_max\_charging\_price):** "Avancerad Elbilsladdning Max Elpris" (kr/kWh)
* **Number (...\_solar\_charging\_buffer):** "Avancerad Elbilsladdning Solenergi Buffer" (W)
* **Number (...\_min\_solar\_charge\_current\_a):** "Avancerad Elbilsladdning Minsta Laddström Solenergi" (A)
* **Sensor (...\_session\_energy):** "Avancerad Elbilsladdning Session Energi" (kWh)
* **Sensor (...\_session\_cost):** "Avancerad Elbilsladdning Session Kostnad" (SEK)
* **Sensor (...\_active\_control\_mode):** "Avancerad Elbilsladdning Aktivt Styrningsläge" (Visar "PRIS\_TID", "SOLENERGI" eller "AV" baserat på faktiskt pågående, kontrollerad laddning).

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

SmartEVChargingCoordinator hanterar all logik.

### **6.1. Övergripande Funktion**

Koordinatorn uppdateras både periodiskt (via scan\_interval\_seconds) och händelsestyrt via state listeners för laddarstatus, elpris, solproduktion och bilens SoC.

### **6.2. Hantering av Koordinatorns Initiala Tillstånd och Uppstart**

* **Explicit Återställning:** Vid koordinatorns allra första uppdatering (i \_async\_first\_refresh) nollställs interna tillståndsvariabler (active\_control\_mode\_internal, is\_smart\_charging\_controlled\_now, last\_commanded\_solar\_amps, \_proactive\_pause\_done\_this\_connection) för att säkerställa ett rent startläge och undvika att gamla värden från tidigare HA-sessioner påverkar den initiala beräkningen.
* **Fördröjd Första Uppdatering:** En fördröjning (f.n. 5 sekunder, definierad av INITIAL\_SETUP\_DELAY\_SECONDS) har lagts in i början av \_async\_first\_refresh. Detta ger andra entiteter och integrationer i Home Assistant tid att initialiseras fullt ut innan vår integration gör sin första omfattande datainhämtning och logikkörning, vilket minskar risken för att nödvändiga sensorer är otillgängliga initialt.

### **6.3. State Listeners för Omedelbar Reaktion**

* **\_handle\_charger\_status\_change (Laddarstatus):**
  + **Proaktiv Paus:** När status ändras till "ready\_to\_charge" (och ett smartläge är aktivt samt ingen paus redan gjorts för anslutningen), anropas \_execute\_assert\_control\_and\_pause() för att omedelbart skicka override\_schedule och pause. Detta förhindrar oönskad start innan integrationen hunnit utvärdera.
  + **Frånkoppling:** Vid "disconnected" återställs flaggor (\_proactive\_pause\_done\_this\_connection, is\_smart\_charging\_controlled\_now, last\_commanded\_solar\_amps, active\_control\_mode\_internal).
  + **Extern Paus/Stopp:** Om integrationen tror att den laddar men statusen ändras till något annat än "charging" (t.ex. "paused"), tolkas det som ett externt ingrepp. Kontrollflaggor och relevanta tillstånd återställs omedelbart.
  + Alla åtgärder följs av en async\_request\_refresh() för omedelbar omvärdering.
* **\_handle\_price\_change (Elpris):** Vid ändring av price\_sensor\_id, triggas en async\_request\_refresh().
* **\_handle\_solar\_production\_change (Solproduktion):** Vid ändring av solar\_production\_sensor\_id (och om solenergiläget är aktivt), triggas en async\_request\_refresh().
* **\_handle\_ev\_soc\_change (Bilens SoC):** Vid ändring av ev\_soc\_sensor\_id (och om SoC-kontroll är fullt konfigurerad), triggas en async\_request\_refresh().

### **6.4. Proaktiv Paus och Huvudswitch-hantering (i \_async\_update\_data)**

* **Huvudswitch:** Om ett smartläge är aktivt och huvudswitchen är AV, slås den PÅ.
* **Fallback Proaktiv Paus:** En fallback-logik finns kvar som kan anropa \_execute\_assert\_control\_and\_pause() om laddaren är "ready\_to\_charge" eller "awaiting\_start" och state listenern ännu inte har agerat.

### **6.5. Beslutslogik för Laddningslägen (i \_async\_update\_data)**

Beslutsprocessen sker i följande prioriterade ordning:

1. **SoC-Laddningsgränskontroll (Högst Prioritet):**
   * Om både CONF\_EV\_SOC\_SENSOR och CONF\_TARGET\_SOC\_LIMIT är konfigurerade:
     + Aktuell SoC från sensorn jämförs med den inställda gränsen.
     + Om aktuell\_SoC >= SoC\_gräns, sätts should\_charge\_flag = False och reason\_for\_action till "Laddningsgräns nådd...". All vidare utvärdering av Pris/Tid och Solenergi hoppas då över för denna cykel.
2. **Pris/Tid-styrd Smartladdning (om SoC tillåter):**
   * **Villkor:** Switchen PÅ, schema aktivt, spotpris <= maxpris.
   * **Åtgärd:** new\_intended\_active\_control\_mode = "PRIS\_TID", target\_charge\_current\_a = charger\_hw\_max\_current\_a, should\_charge\_flag = True.
   * Anledningen till utfallet (även om det inte leder till laddning) loggas separat.
3. **Solenergiladdning (om SoC och Pris/Tid tillåter):**
   * **Villkor:** Switchen PÅ, schema aktivt, tillräckligt solöverskott.
   * **Beräkning av överskott (dämpad):**
     + EVOffset\_Watt baseras på self.last\_commanded\_solar\_amps om en solsession fortsätter, annars 0W för en ny solsession. Detta förhindrar att en tidigare hög (nät-driven) laddeffekt felaktigt ökar beräknat solöverskott.
     + Tillgängligt\_för\_EV\_W = Solproduktion\_W - (HusTotal\_W - EVOffset\_Watt) - SolarBuffer\_W.
   * **Målström:** Beräknas baserat på överskottet, avrundas nedåt till heltal Ampere, och får aldrig vara under EASEE\_ABSOLUTE\_MIN\_AMPS (6A) om laddning ska ske. Användarens min\_solar\_current\_a respekteras också.
   * **Åtgärd:** new\_intended\_active\_control\_mode = "SOLENERGI", target\_charge\_current\_a = beräknad\_solström, should\_charge\_flag = True.
   * Anledningen till utfallet loggas separat.
   * self.last\_commanded\_solar\_amps uppdateras endast om laddning faktiskt initieras eller pågår i solenergiläge och strömgränsen ändras.

### **6.6. Styrning av Laddare via Hjälpfunktioner**

* **\_execute\_assert\_control\_and\_pause():** Skickar override\_schedule och pause. Sätter is\_smart\_charging\_controlled\_now = True, \_proactive\_pause\_done\_this\_connection = True.
* **\_execute\_stop\_charging():** Skickar pause (om status tillåter). Nollställer is\_smart\_charging\_controlled\_now = False och last\_commanded\_solar\_amps (om relevant).
* **\_execute\_start\_or\_adjust\_charging():**
  1. Skickar set\_charger\_dynamic\_limit (endast om det nya heltalsvärdet för solenergi skiljer sig från senast kommenderade).
  2. Om ny session/lägesbyte (och ej disconnected): skickar override\_schedule, nollställer sessionsdata.
  3. Uppdaterar last\_commanded\_solar\_amps om solenergiladdning startar/fortsätter och strömgränsen skickades.
  4. Skickar start om status tillåter (inkluderar awaiting\_start och ready\_to\_charge, men inte disconnected).
  5. Sätter is\_smart\_charging\_controlled\_now = True.
  6. Returnerar True om laddning initierades/var aktiv, annars False.

### **6.7. Reaktiva Åtgärder**

* **Reaktivt Stopp (i \_async\_update\_data):** Om should\_charge\_flag är False, men laddaren är "charging" och vi inte trodde oss ha kontroll, anropas \_execute\_assert\_control\_and\_pause().
* **Hantering av Extern Paus (via State Listener):** Beskrivet i 6.3.1.

### **6.8. Hantering av sensor.avancerad\_elbilsladdning\_aktivt\_styrningslage**

Variabeln self.active\_control\_mode (som matar sensorn) uppdateras strikt i slutet av \_async\_update\_data:

* Den sätts till det *avsedda* läget (self.active\_control\_mode\_internal, t.ex. "PRIS\_TID" eller "SOLENERGI") **endast om**:
  + self.is\_smart\_charging\_controlled\_now är True.
  + Laddarens status (current\_charger\_status\_for\_sensor\_check) är "charging".
  + Och charging\_actually\_started\_or\_continued (från \_execute\_start\_or\_adjust\_charging) är True.
* I alla andra fall sätts self.active\_control\_mode till None (vilket visas som "AV" i sensorn).

### **6.9. Sessionsdata**

Ackumuleras om is\_smart\_charging\_controlled\_now är True, laddaren är "charging", och ev\_power\_w > 0. Kostnad beräknas endast för Pris/Tid-läget.

## **7. Kända Problem / Fortsatt Utveckling (Uppdaterad 2025-05-23)**

* **Fas-hantering:** Behöver fortfarande verifieras fullt ut för att säkerställa korrekt strömberäkning och kommandon för 1-fas vs 3-fas laddning.
* **Stabilitet kring awaiting\_start:** Även om state listenern nu fokuserar på ready\_to\_charge, kan awaiting\_start fortfarande vara ett känsligt läge. Kontinuerlig övervakning av loggar för "Charger disconnected"-fel är viktig.
* **Tröskel för Solproduktions-listener:** Överväg en konfigurerbar tröskel för att minska antalet uppdateringar om sensorn är "brusig".
* **Avlastning av Listeners:** \_remove\_listeners finns, men anropet från async\_unload\_entry i \_\_init\_\_.py bör vara robust för att förhindra resursläckor. Den nuvarande versionen av async\_unload\_entry inkluderar detta.
* **Felsökning av konfigurationssparande:** Fortsatt uppmärksamhet på att config\_flow.py (särskilt OptionsFlowHandler) tillförlitligt sparar alla användarval och att coordinator.py korrekt läser in den senaste konfigurationen (data och options) vid uppstart och efter ändringar.
* **Hantering av "Ej hittade entiteter":** Förbättra loggning och eventuellt användarfeedback om konfigurerade sensorer inte kan hittas eller blir otillgängliga. Kanske en reparationsvarning i HA.

## **8. Bilaga: Kodfiler**

*Alla kodfiler ska versionsmarkeras med en kommentar i början av filen, t.ex. # File version: ÅÅÅÅ-MM-DD.X.*

**Kodfiler ligger separat bifogat.**

Denna rapport bör ge en god grund för att förstå integrationens nuvarande funktionalitet och de designval som gjorts.