Konzept-OpenEMS_GridOp+BESS+PV+OpenEMS.md

Konzept-OpenEMS_GridOp+BESS+PV+OpenEMS.md 10.42 KiB

Konzept: Regelbare Anlage

Dieses Konzept ist aus dem Use-Case einer Regelbaren Anlage bestehend aus den folgenden Teileinheiten hervorgegangen:

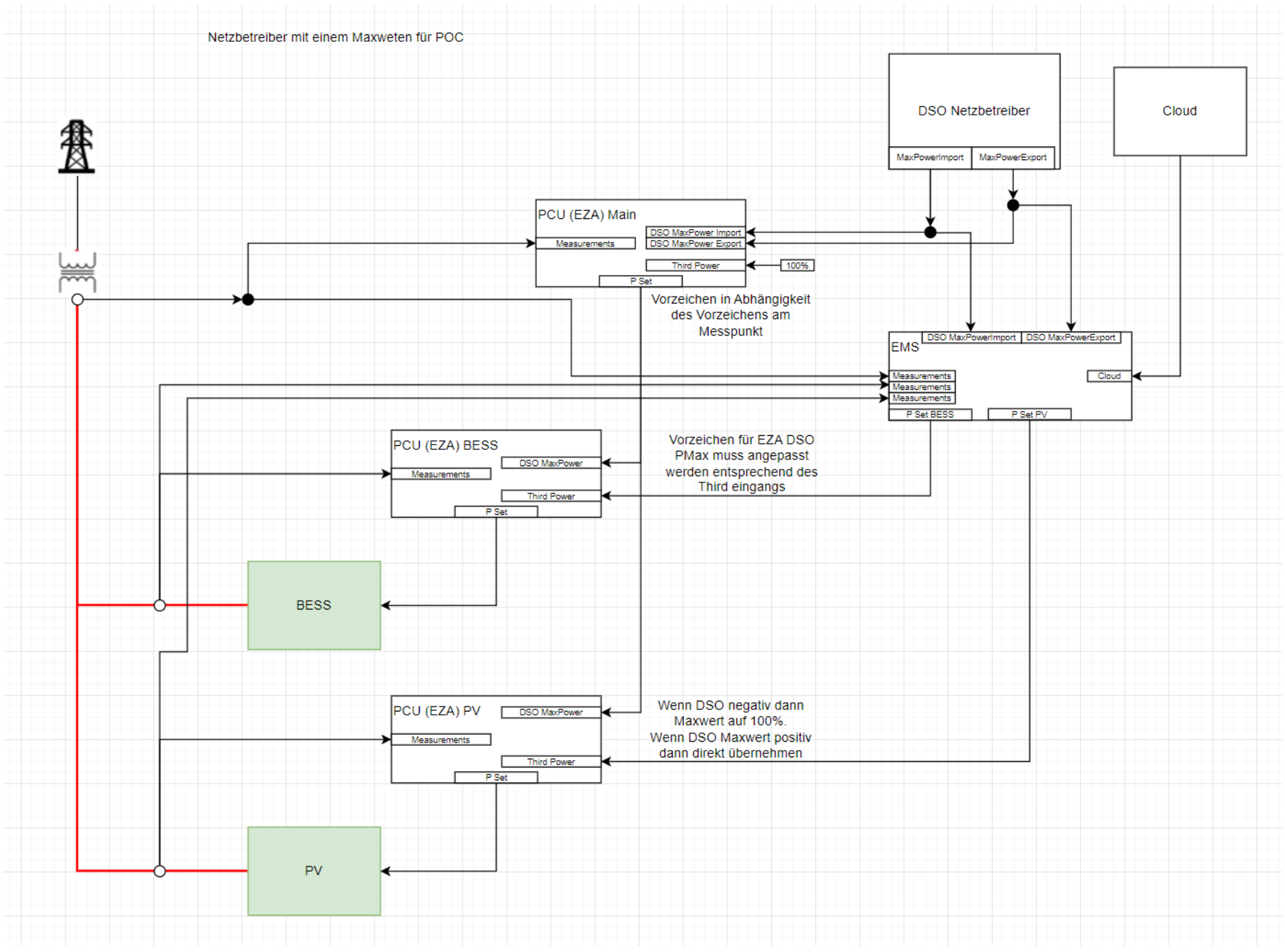
- Grid-Op Limiter: EZA-Regler
- (Managed-Symmetric) ESS
- (Managed-Symmetric) PV-Inverter
- OpenEMS

Zentrale Aufgabe des OpenEMS ist es, die Anlage optimal zu steuern und dabei die Vorgaben von Netzbetreiber (und ggf. Direktvermarkter oder Strombörse) zu berücksichtigen. Dabei läuft das OpenEMS auf einer Phoenix Contact Steuerung an einem definierten Ort, d.h. auch das ESS und die PV-Anlage.

1. Gewünschter technischer Aufbau

Die mit OpenEMS zu steuernde Anlage besteht aus den folgenden Bestandteilen. Das Schaubild unten zeigt schematisch den Aufbau.

- Netzanschlusspunkt: Übergabe-/Bezugspunkt zum/vom Netzbetreiber
- EZA-Regler (Main): EZA-Regler am Netzanschlusspunkt
- EZA-Regler (BESS): EZA-Regler vor dem Batteriespeicher
- BESS: Batteriespeicher mit Wechselrichter
- EZA-Regler (PV): EZA-Regler vor der PV-Anlage
- PV-Anlage: Die PV-Anlage mit Wechselrichter
- und natürlich das OpenEMS



2. Szenarien

Innerhalb dieses technischen Aufbaus gibt es recht unterschiedliche Szenarien, die abzudecken sind. Für dieses Konzept sind jedoch beiden Szenarien 5 und 6 des zugrundeliegenden Use-Cases ausgewählt worden, da diese fachlich recht eng zusammenhängen und eine Ausweitung auf Direktvermakter bzw. Strombörse zulassen. Im zugrundeliegenden Use-Case gibt es noch deutlich mehr Szenarien, daher bitte nicht an der Nummerierung stören.

Szenario 5: *Netzbetreiber regelt Einspeisung auf N Watt ab, ESS hat Kapazität*

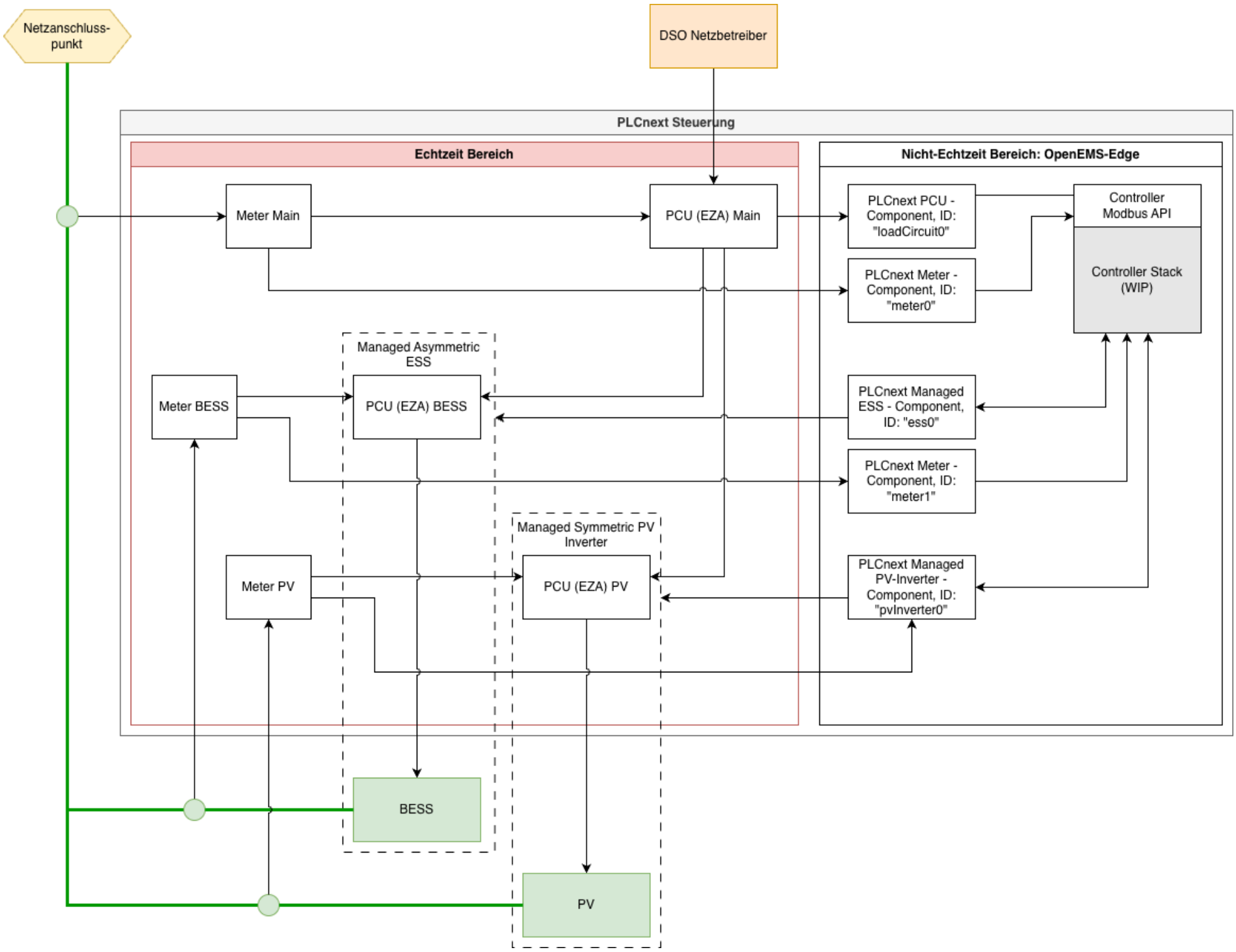
- EMS prüft die Kapazitäten des ESS und passt die Ladeleistung entsprechend an.
- EMS prüft die erzeugte Leistung der PV-Anlage und regelt die Leistung herunter, die nicht an das ESS geschickt werden kann und auch nicht im Netz ankommen darf.

Szenario 6: *Netzbetreiber regelt Einspeisung auf N Watt ab, BESS hat keine Kapazität*

- EMS regelt das Aufladen des ESS mit der von der PV-Anlage erzeugten Energie komplett ab.
- EMS prüft die erzeugte Leistung der PV-Anlage und regelt die Leistung herunter, die nicht im Netz ankommen darf.

3. Komponentenaufbau des Use-Cases

Das folgende Bild zeigt Abbildung des im ersten Unterabschnitt aufgeführten technischen Aufbaus für die ausgewählten im vorhergehenden Unterabschnitt skizzierten Szenarien. Es ist zu sehen, dass jede PLCnext Komponente von einem "Zwilling" im OpenEMS abgebildet wird. Die in OpenEMS mittels Controllern (und natürlich Schedulern) realisierte Business-Logic ist mit dem grauen Kasten "Controller Stack (WIP)" angedeutet, auf die in den folgenden Unterabschnitten eingegangen wird.



Im Bild zu sehen sind die drei Ebenen des Use-Cases:

- Netzanschlusspunkt: Mit Zähler (= Meter) und EZA-Regler (= PCU (EZA) Main)
- Batteriespeicher (BESS): Mit Zähler (= Meter), EZA-Regler (= PCU (EZA) BESS) und dem gemanagten Speicher selbst
- Photovoltaik-Anlage: Mit Zähler (= Meter) EZA-Regler (= PCU (EZA) PV) und dem PV-Wechselrichter (= PV-Inverter) selbst

Für die Umsetzung der Business-Logik dieses Use-Cases ist es wichtig die folgenden Hintergrund Informationen zu berücksichtigen.

- Das verwendete OpenEMS Modul "OpenEMS Edge" arbeitet in sogenannten "Zyklen".
- Ein "Zyklus" besteht aus den folgenden Phasen, die sequentiell abgearbeitet werden.
  1. Eingabe
  2. Einfrieren der Werte (aus der Phase "Eingabe")
  3. Verarbeitung

4. Ausgabe

- Während der Phase "Eingabe" werden die Werte aller angeschlossenen Geräte in die sogenanngen "Read-Channels" eingelesen.
- Das Lesen der Werte aller angeschlossenen Geräte erfolgt mittels asynchroner Kommunikation.
- Während der Phase "Einfrieren der Werte" werden nur die aktuell vorliegenden Werte eingefroren! (siehe Anmerkung unten)
- Ein "Scheduler" steuert die Reihenfolge der Ausführung der sogenannten "Controller" während der Phase "Verarbeitung".
- Die "Controller" selbst enthalten die abzubildende Business-Logik.
- Jeder "Controller" arbeitet auf den gleichen eingefrorenen Werten plus potentieller Ausgabewerte eines zuvor ausgeführten "Controllers".
- Stellt ein "Controller" fest, dass es für diesen nichts zu tun gibt, wird dess Business-Logik übersprungen. Der "Scheduler" führt dann den nächsten "Controller" aus, sofern einer konfiguiert ist.
- Die Weitergabe von Daten an einen nachfolgenden "Controller" ist über sogenannte "Write-Channels" möglich.
- "Write-Channels" werden auch dafür genutzt, um Daten an angeschlossene Geräte auszugeben, z.B. Set-Points für ESS oder PV-Inverter.
- Es ist möglich eine Abfolge von "Controllern" laufen zu lassen.
- Beim Priorisieren der Ausgabe-Werte mehrer laufender "Controller" gewinnt die Regel mit der stärksten Restriktion bzw. des "Controllers" der früher ausgeführt worden ist bei fehlender Unterscheidung über die Restriktion.

Anmerkung: In der Phase "Einfrieren der Werte" ist zu beachten, dass aufgrund der asynchronen Kommunikation mit den Geräten ggf. nicht alle Werte während Phase "Eingabe" eingelesen werden konnten. D.h. es steht immer nur ein Teil der Werte in unterschiedlicher Aktualtität zur Verfügung. Im folgenden "Zyklus" sind zwar dann alle Werte da, ein Teil kann jedoch schon aktualisiert sein.

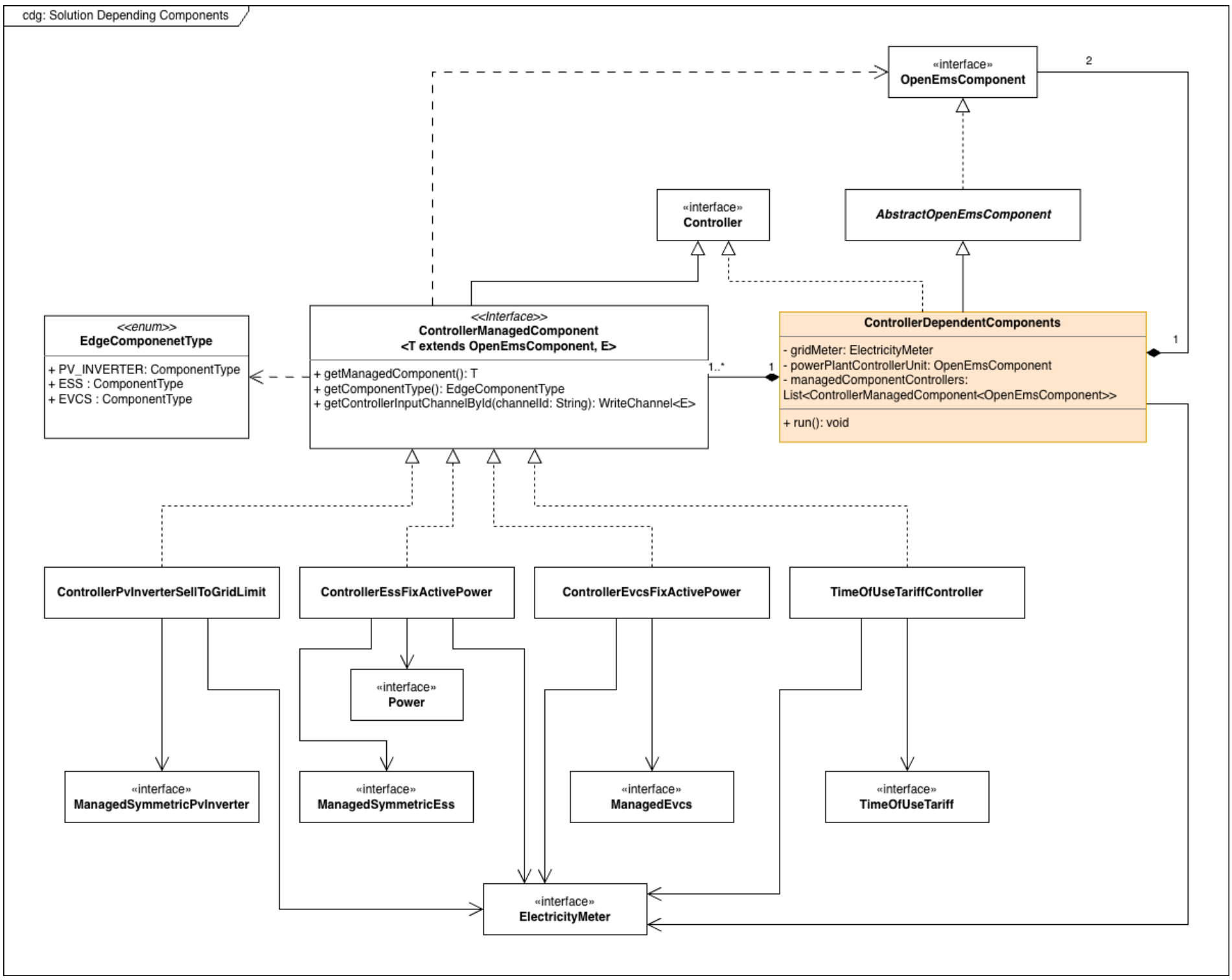
In der Architektur Dokumentation der "OpenEMS Edge" (hier das wesentliche Schaubild) sind die oben genannten Punkte zum Teil visualisiert.

4. Lösungsansatz für ausgewählte Szenarien

Dieser Abschnitt beschreibt die Umsetzung der Business-Logik für die ausgewählten Szenarien "Netzbetreiber regelt die Einspeisung ab". Laut Aussage der FENECON ist der skizzierte Use-Case mit der aktuellen Version von OpenEMS und der darin enthaltenen Controller noch nicht abbildbar. Es wird jedoch darüber nachgedacht diese Art von Use-Cases zu implementieren. Daher wird dieses Konzept der Community als Proposal für Feedback zur Verfügung gestellt.

Für eine Business-Logic, die voneinander getrennte PV-Inverter und ESS managen kann ist ein neuer "Controller" zu implementieren. In den beiden folgenden Schaubildern ist dieser Controller als `ControllerDependentComponents` bezeichnet. Aufgabe dieses Controllers ist es, unterstütze Controller für PV-Anlage und ESS mit dem EZA-Regler und Grid-Meter zusammen zu bringen.

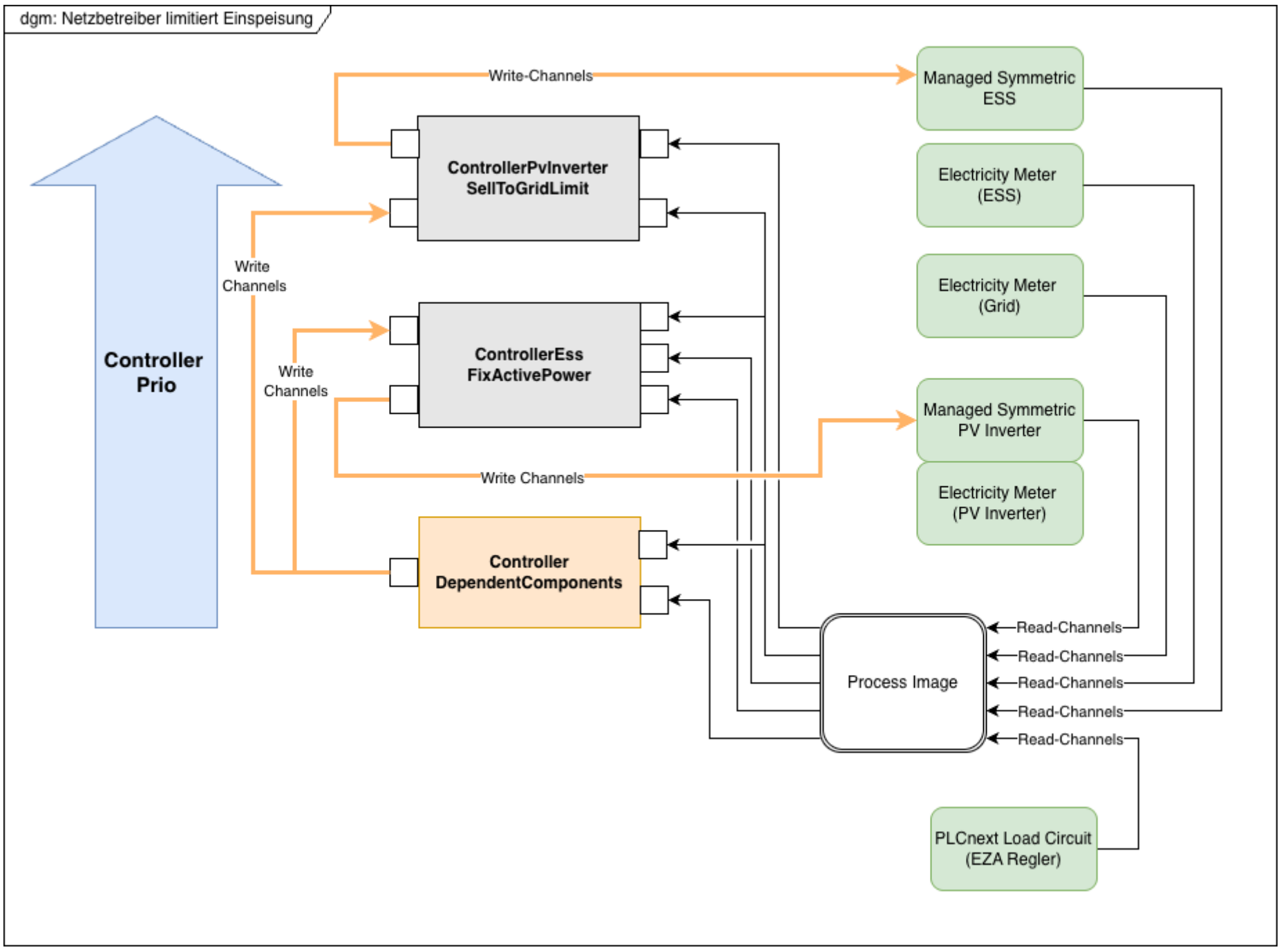
Wie das theoretisch funktionieren kann, zeigt das folgende Klassendiagramm. Es wird ein neues Interface namens `ControllerManagedComponents` (markiert in rot) als Erweiterung des `Controller` Interfaces im Modul `io.openems.edge.controller.api` implementiert. Dieses dient dazu unterstützende, konfigurierte und aktive Controller anhand ihrer ID mit dem neuen `ControllerDependentComponents` zu verknüpfen. Damit kann der neue Controller die benötigten `Channel` der jeweiiligen gemanagten Komponenten abfragen. Gleichzeitig wird eine Möglichkeit bereitgestellt den verknüpften Controllern über einen oder mehrere `WriteChannel` Daten mitzugeben. Da die Business-Logik der injizierten Controller jeweils nach dem neuen Controller laufen, können diesen so ggf. benötigte Vorgaben gemacht werden. Damit ein Controller das Gruppieren im neuen Controller unterstützt, muss dieser das Interface `ControllerManagedComponent` implementieren.



Eine Übersicht der ausgeführten Business-Logik in Summe zeigt das folgende Schaubild. Dieses zeigt den `ControllerPvInverterSellToGridLimit` als verantwortlichen Controller zum Steuern des `PV-Inverter`, sowie den `ControllerEssFixActivePower` als verantwortlichen Controller zum Steuern des `ESS`. Darüber hinaus zeigt es die mittels PLCnext Treibern implementierten digitalen Zwillinge zur Logik auf der PLCnext Steuerung. Auch hier ist der neue `ControllerDependentComponents` in rot markiert. Der Prozess selbst läuft ab, wie folgt.

1. Während der Phase "Eingabe" liefern die digitalen Zwillinge ihre Messwerte bei der OpenEMS Edge ab.
2. In der anschließen Phase "Einfrieren der Werte" werden diese Werte in ihren jeweiligen Kanälen fixiert.
3. Darauf folgt in Phase "Verarbeitung" das Abarbeiten der konfigurierten Controller.
    1. Der `ControllerDependentComponents` prüft zunächst den Inhalt seiner `Channel` und der Eingangskanäle der beiden anderen verknüpften Controller, ob eine Abregelung der Gesamtanlage am Netzanschlusspunkt notwendig ist. Dabei prüft dieser auch anhand der vorliegenden Daten, wie diese potentielle Abregelung der angeschlossenen Komponenten ausgestaltet werden muss. Das ermittelte Ergebnis wird per benötigten `WriteChannel` der jeweiligen injizierten Controller Instanz bekanntgegeben.
    2. Als nächstes läuft der `ControllerEssFixActivePower` (als Beispiel gewählt), der seine Konfiguration und den Inhalt seines `WriteChannel` prüft, der ggf. vom zuvor gelaufenen `ControllerDependentComponents` gefüllt sein kann. Auf dieser Basis, sowie der verfügbaren Messwerte in den `Channel` des verknüpften ESS und Meters erledigt der Controller vorschriftsmäßig seine Arbeit. Abregelungen und weitere Steuerungsdaten gibt dieser Controller in die benötigten `WriteChannel` des verknüpften ESS.
    3. Im Anschluss daran läuft der `ControllerPvInverterSellToGridLimit` (als Beispiel gewählt) der auch die Messwerte der `Channel` seines verknüpften `PV-Inverter`, sowie die vom `ControllerDependentComponents` gefüllten `WriteChannel` abfragt und entsprechend die PV-Anlage über die `WriteChannel` des verknüpften `PV-Inverter` steuert.
4. In der abschließenden Phase "Ausgabe" werden die Inhalte der `WriteChannel` von ESS und `PV-Inverter` mit dem zugehörigen Treiber an die Hardware übermittelt.

Im Anschluss beginnt ein neuer Zyklus bei 1.



HINWEIS: Für eine klare Priorisierung der Controller wird der Einsatz des `Fixed Order Scheduler` empfohlen.