Konzepte

Die Konzepte im Ebenenmodell dienen als Design-Pattern und Nomenklatur. Sie geben Beispiel für die Implementierung und Abbildung auf eine individuelle Problemstellung

- SaldoId
- Bilanzierung einer Ebene (Saldo)
- Zweistufige Prognoseerstellung
- Ebenen GrünstromIndex

SaldoId

Eines der Ziele jeder Ebene ist die Bildung einer Energiebilanz. Damit auch asynchron mit Sensordaten gearbeitet werden kann, wird eine SaldoId verwendet.

saldo_id_trigger.png
Image not found or type unknown

Auf jeder Ebene wird die SaldoId mit Hilfe eines *Inject* einmal zum Start des Flows und dann in regelmäßigen Intervallen neu definiert und als Flow Variable den Komponenten der jeweiligen Ebene (Flow) zur Verfügung gestellt. Gleichzeitig werden die Summen des Flows zurückgesetzt.

```
flow.set("Consumption",0);
flow.set("Generation",0);
flow.set("SaldoID",new Date().toLocaleString());
return msg;
```

Hinweis

Die SadoId sollte nicht zu häufig aktualisiert werden, damit bei der Bildung der Bilanz für die Ebene alle Messwerte berücksichtigt werden.

Bilanzierung einer Ebene (Saldo)

Jede Ebene bildet für sich eine Bilanz (Saldo). Hierbei wird in der Einheit Watt gerechnet (nicht WattStunde!), da die Sensoren so oft wie möglich einen Messwert liefern sollen.

saldo_ebene0.png
Image not found or type unknown

```
msg.payload = flow.get('Consumption') - flow.get('Generation');
flow.set('Saldo',msg.payload);
node.status({text:"P: "+msg.payload+" W"});
return msg;
```

Zweistufige Prognoseerstellung

Die Vorhersagen in den Ebenen werden in zwei Schritten durchgeführt. Zunächst wird eine generische Abbildung auf Abhängige Werte vorgenommen und im Anschluss in einen Graphen (=konkrete Zeitreihe) überführt.

Schritt 1: Abbildung

vorhersage_schritt1.png
Image not found or type unknown

```
// Create empty array if not exists
let hours = {};
for(let i=0;i<24;i++) {
    hours["h"+i] = {
        latest: {},
        outcome: {
            sum: 0,
            cnt: 0,
            mean: 0,
            min: 9999999,
            max: 0,
            min_gsi: 999999
        }
    };
}

for(let i=0;i<msg.payload.length;i++) {
   let hour = new Date(msg.payload[i].time).getHours();
   hours["h"+hour].latest.gsi = msg.payload[i].mean + 1;
}

msg.hours = hours;
return msg;
```

In diesem Schritt wird ein Array mit einem Element für jede Stunde des Tages gebildet (Zeile 3-15). Auf Basis der Werte aus der Vergangenheit wird nun ein Wert für jede Stunde prognostiziert.

Ist die Vorhersage von einem anderen Wert als die Tageszeit abhängig, wie zum Beispiel bei der Sonneneinstrahlung, so wird im Anschluss die vorhandenen Bedingungen (Beispiel: Sonneneinstrahlung) zum jeweiligen historischen Wert gemappt.

Schritt 2: Graphen


```
let a = [];


for(let i=0;i<msg.payload.gsi.forecast.length;i++) {
    let item = {
        'x':msg.payload.gsi.forecast[i].timeStamp,
        'y':msg.payload.gsi.forecast[i].esolar
    }
    a.push(item);
}


msg.graph = [{
    'series': ['GenerationForecast'],
    'data': [a]
}];
return msg;
```

Der Graphen bildet die Zeitstempel für die kommenden 36 Stunden (=Vorhersagezeitraum) ab. Er wird bereits so aufgebaut, dass er am Ende der Verarbeitung in der InfluxDB gespeichert und mittels der Grafana Dashboards visualisiert werden kann.

Ebenen GrünstromIndex

ebenen_gsi.png
Image not found or type unknown

Jede Ebene berechnet einen eigenen GrünstromIndex. Dieser liegt im Wertebereich zwischen 0 und 100, wobei ein hoher Indexwert ein Indikator für viel Strom und ein niedriger Indexwert einer für wenig Strom ist. Der GrünstromIndex wird in der regel für die kommenden 36 Stunden berechnet.

gsi_ebenen4.png
Image not found or type unknown

In den Ebenen 0 und 1 wird der Wert auf Basis der Prognosen erstellt. In der Ebene 3 wird der für die Postleitzahl des Objektes gültige öffentliche GrünstromIndex verwendet.

```
let saldo = flow.get("forecastSaldo");
let min = 9999999999;
let max = -999999999999;

for(let i=0; i<saldo.length;i++) {
    saldo[i].measurement = "e0gsi";
    if(saldo[i].fields.w > max) max = saldo[i].fields.w ;
    if(saldo[i].fields.w < min) min = saldo[i].fields.w ;
}

let delta = max - min;

for(let i=0; i<saldo.length;i++) {
    saldo[i].fields.gsi = 100-Math.round(((saldo[i].fields.w  - min) / delta)*100);
}



flow.set("gsi",saldo);
global.set("e0gsi",saldo);
msg.payload = saldo;

return msg;
```