Meine IPSView-Oberfläche mit Claude bauen: Band, Seiten, Pop-ups, Schalter und Meldungen (iPad)

Stand: Oktober 2026 · IP-Symcon 9 · IPSView auf dem iPad quer (1366 × 846 px). Alle Bilder zeigen neutrale Demo-Daten. Du musst dafür keinen Code schreiben oder lesen: Claude baut, du beschreibst und gibst frei.

1. Worum geht es?

Ich habe zu Hause die komplette IPSView-Oberfläche für das iPad als **HTML-Seiten in HTML-Boxen** gebaut, und zwar zusammen mit **Claude** (dem KI-Assistenten von Anthropic), der direkt an IP-Symcon angebunden ist. Oben liegt ein **Band** (Kennzahlen, Raum-Buttons, Schalter, Meldungen), darunter und in **Pop-ups** die einzelnen Seiten. Alles zeigt Live-Daten aus IP-Symcon und sieht auf jedem iPad gleich aus.

Der Kern dieser Anleitung: **Du schreibst keinen Code.** Du sagst Claude in normalen Sätzen, was du sehen willst. Claude sucht die passenden Variablen in deinem IP-Symcon, baut die Seite zuerst in einem **Flächen-Simulator** (damit du sie vorher ansehen kannst), legt sie nach deiner Freigabe in IP-Symcon ab und sagt dir, was du im IPSView-Editor noch einsetzen musst. Das ist nur das Platzieren von HTML-Boxen und transparenten Flächen.

Das Besondere (so habe ich es im Forum noch nicht gesehen):

- Ein **Flächen-Simulator** im Browser: Jede Seite wird vorher am Rechner gebaut und getestet, ohne bei jeder Änderung in den IPSView-Editor zu müssen.
- **Überwachung und Meldungen** direkt in den Buttons: gelbes „!“ für Hinweise, rotes pulsierendes „!“ für Warnungen, Stummschaltung per Glocke.
- **Anzeigen im Button**: Klimaanlage (heizt oder kühlt) und offene Fenster.
- **Schalter und Pop-ups** über unsichtbare IPSView-Flächen, die genau über der HTML-Box liegen.
- **Zwei Wege** zur Einbindung: eine einzige HTML-Box oder zwei Boxen (Band und Fenster).

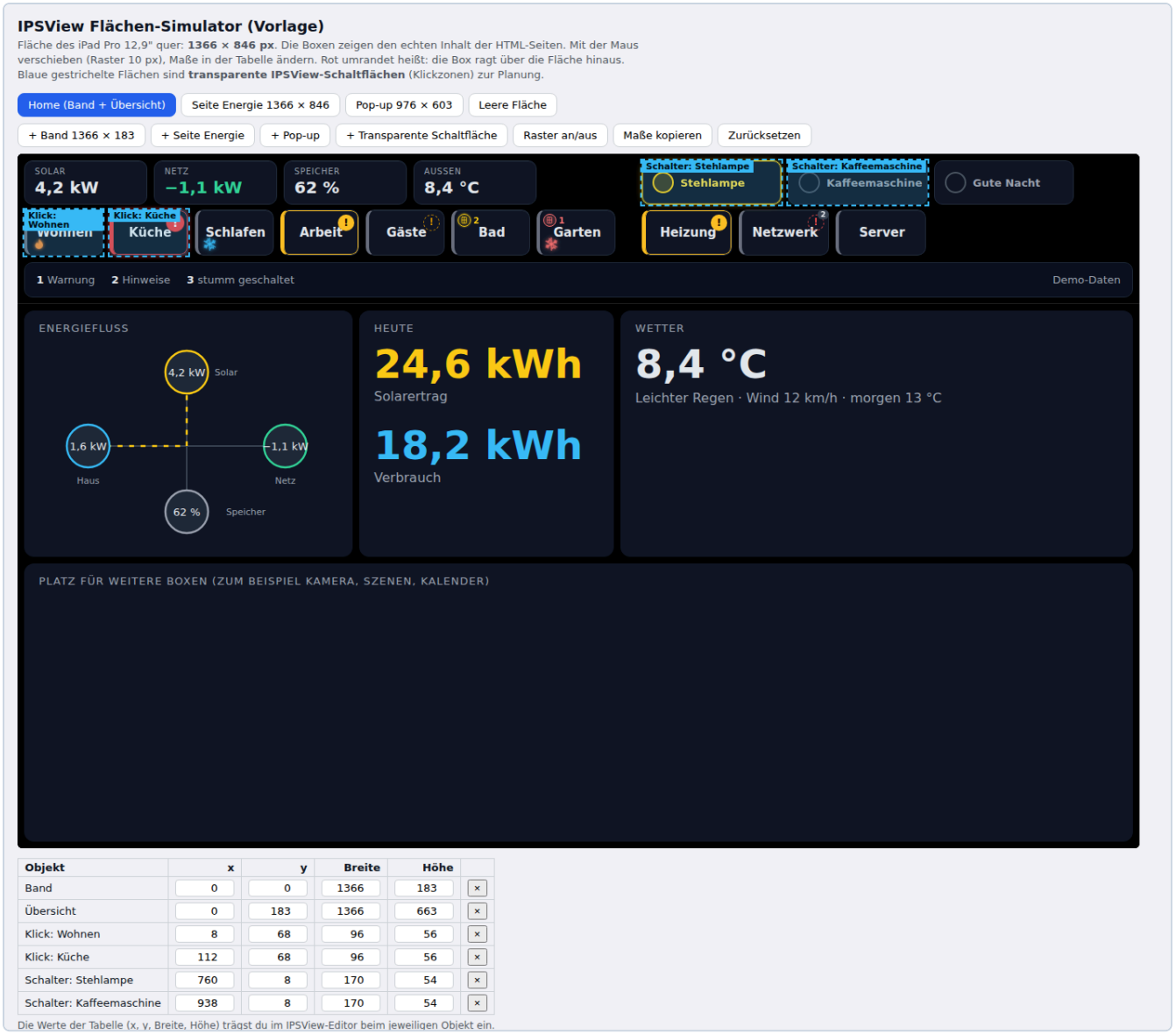


Bild 1: Der Flächen-Simulator mit Band, Übersicht und blauen Klickzonen für die transparenten IPSView-Flächen.

2. Was du brauchst

- **IP-Symcon** (bei mir Version 9) mit einem laufenden Server und **IPSView** auf dem iPad.
- Einen **Claude-Zugang** mit der Claude-Desktop-App (Mac oder Windows) und der Möglichkeit, einen MCP-Server einzubinden. Welches Abo dafür nötig ist, bitte bei Anthropic nachsehen, das ändert sich.
- Einen **Rechner im selben Netz** wie IP-Symcon, auf dem die Claude-Desktop-App läuft.
- Einen **Ordner für Notizen** auf diesem Rechner. Claude legt dort ab, was es sich merken soll (IDs, Entscheidungen, Sicherungen).
- Einen **Browser** zum Ansehen des Simulators.

3. Claude mit IP-Symcon verbinden

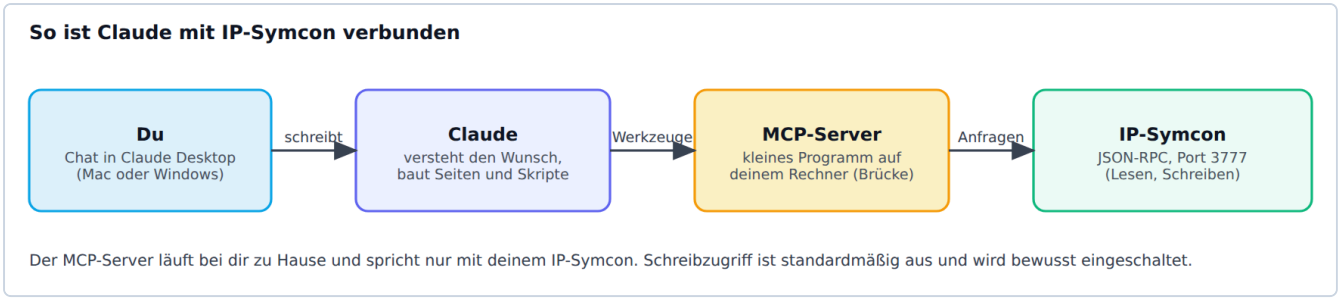


Bild 2: Du sprichst mit Claude, Claude nutzt über eine kleine Brücke (MCP-Server) die JSON-RPC-Schnittstelle von IP-Symcon.

Claude kann nicht von selbst auf dein IP-Symcon zugreifen. Dafür gibt es eine Brücke, einen sogenannten **MCP-Server**. Ich nutze das freie Projekt ipsymcon-mcp-server von Schimmilab. Es läuft auf deinem Rechner und spricht über die **JSON-RPC-Schnittstelle** von IP-Symcon (standardmäßig `http://<IPS-Adresse>:3777/api/`) mit dem Server. Nichts davon musst du selbst programmieren. Du kannst Claude bitten, dich durch die Einrichtung zu führen.

Die Einrichtung in vier Schritten:

1. **Benutzer in IP-Symcon anlegen** (Systemsteuerung, Benutzerverwaltung). Am besten einen eigenen Benutzer nur für Claude, nicht dein Haupt-Konto.
2. **MCP-Server installieren** nach der Anleitung des Projekts (Python-Umgebung, eine kleine Konfigurationsdatei mit Adresse, Benutzer und Passwort). Das Passwort bleibt in dieser Datei auf deinem Rechner und gehört nie in einen Chat.
3. **In Claude Desktop eintragen**, damit die Werkzeuge im Chat erscheinen. Das Projekt beschreibt den Befehl für Claude Code. Für Claude Desktop trägt man den Server in die MCP-Einstellungen der App ein. Auch dabei kann Claude helfen.
4. **Testen:** Schreibe „Lies mir den Wert der Variable XY“ oder „Zeig mir die Kategorien unter der Wurzel“. Wenn Claude antwortet, steht die Verbindung.

Lesen und Schreiben sind getrennt. Der Server startet standardmäßig **nur lesend**. Schreiben (Skripte anlegen, Variablen setzen, Hooks eintragen) schaltest du bewusst ein. Dann kann Claude alles, was ein Benutzer mit diesen Rechten in IP-Symcon kann, auch Skripte ausführen. Deshalb gilt:

- Lege vor dem ersten Schreiben eine **Sicherung** von IP-Symcon an.
- Beginne mit **nur Lesen**: Claude schaut sich dein System an und schlägt vor, was möglich ist.
- Schalte Schreiben erst frei, wenn du den Vorschlag gesehen hast.
- Gib Claude die **Spielregeln** aus dem nächsten Kapitel. Das ist nur eine Anweisung an Claude, kein technischer Schutz, deshalb zählt die Sicherung.

4. Die Spielregeln für Claude

Schreib diese Regeln am Anfang in den Chat (oder in die Projekt-Anweisungen von Claude). Claude hält sich daran und merkt sie sich über die Notizen im Ordner. Das sind meine, passe sie an:

Meine Regeln für unser IPSView-Projekt

1. Alles zuerst im Flächen-Simulator bauen und mir zeigen. In IP-Symcon übernimmst du nur, wenn ich es sage.
2. Vor jedem Schreiben in IP-Symcon: Sicherung der betroffenen Objekte anlegen und mir sagen, was du ändern willst. Ich antworte mit „ja“. Ein Testergebnis oder ein alter Plan ist keine Freigabe.
3. Der Simulator muss immer genau dem Stand in IP-Symcon entsprechen.
4. Berechnungen gehen vor Optik. Wenn eine Anzeige nicht zur Berechnung passt, sag es mir, statt die Anzeige still anzupassen.
5. Nutze vorhandene Variablen und Berechnungen, statt neue zu erfinden. Frag nach, wenn etwas unklar ist.
6. Farben für Meldungen: gelb = Hinweis, rot = Warnung. Die Glocke schaltet stumm.
7. Lege alles Wichtige (Objekt-IDs, Namen, Entscheidungen, Sicherungen) in meinem Notizordner ab und halte es aktuell.
8. Änderungen an Netzwerk, Geräten oder Systemen nur nach ausdrücklicher Bestätigung.

5. So läuft die Zusammenarbeit

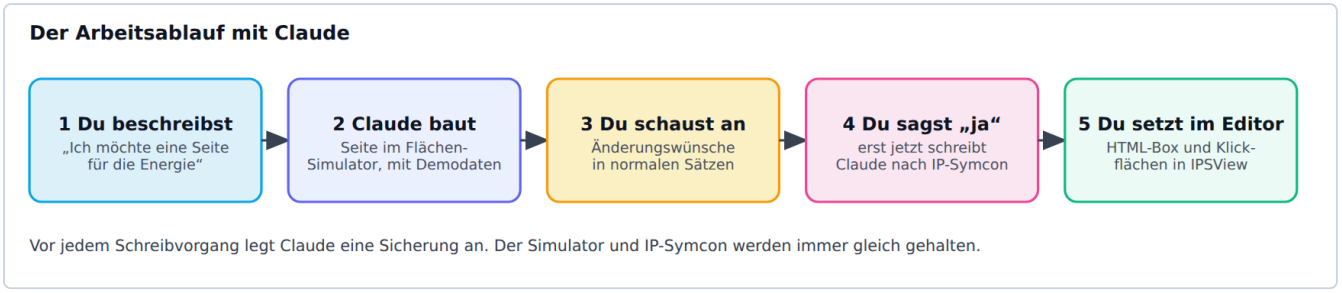


Bild 3: Fünf Schritte, nur der letzte passiert von Hand im IPSView-Editor.

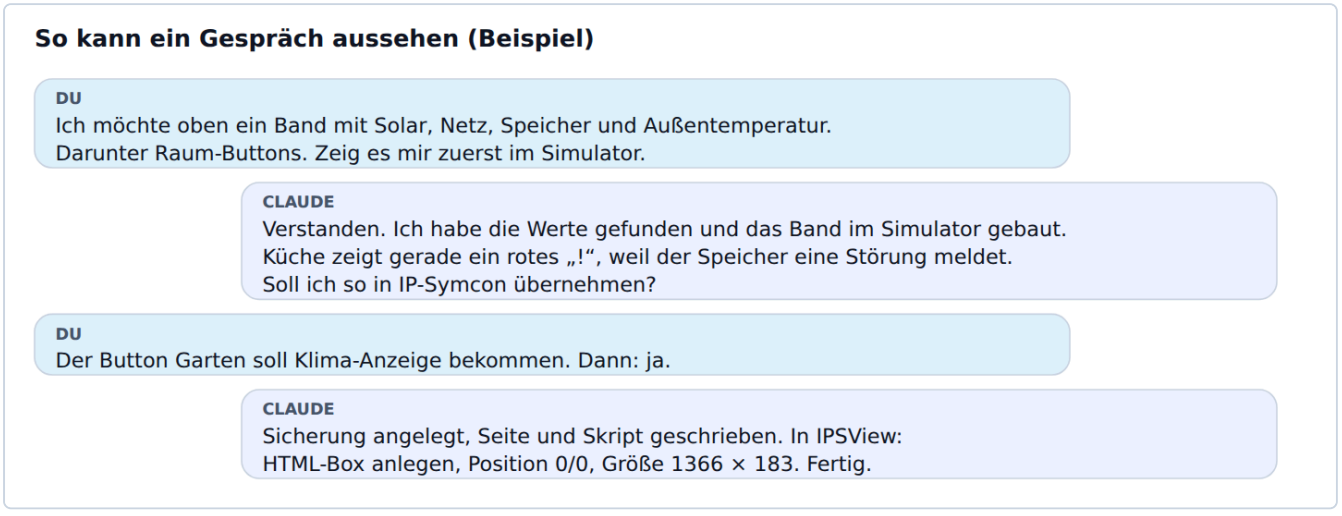


Bild 4: So kann ein Gespräch aussehen.

1. **Du beschreibst**, was du willst, so wie du es einem Kollegen sagen würdest.

2. **Claude liest** in IP-Symcon die passenden Variablen, baut die Seite und zeigt sie im Simulator.
3. **Du schaust sie an** und sagst, was anders sein soll („Der Schriftzug ist zu klein“, „Zeig die Temperatur oben“). Das wiederholt ihr, bis es passt.
4. **Du sagst „ja“.** Dann sichert Claude, legt alles in IP-Symcon an und prüft im Browser, dass die Seite dort läuft.
5. **Du setzt im IPSView-Editor** das, was Claude nicht kann: die HTML-Box auf die Fläche, bei Bedarf die transparenten Klickflächen. Claude nennt dir Variable, Position und Größe.

6. Der Flächen-Simulator: Claude baut ihn für dich

Der Simulator ist **eine einzige HTML-Datei**, die du per Doppelklick im Browser öffnest. Er zeigt die iPad-Fläche **1366 × 846** maßstabsgetreu mit Raster, jede Seite als echte HTML-Box, du kannst Boxen verschieben, Maße ablesen, zu große Boxen werden rot umrandet, und transparente Klickflächen sind als blaue Zonen eingeplant. So sieht ein Pop-up aus, bevor du irgendetwas in IPSView anlegst.

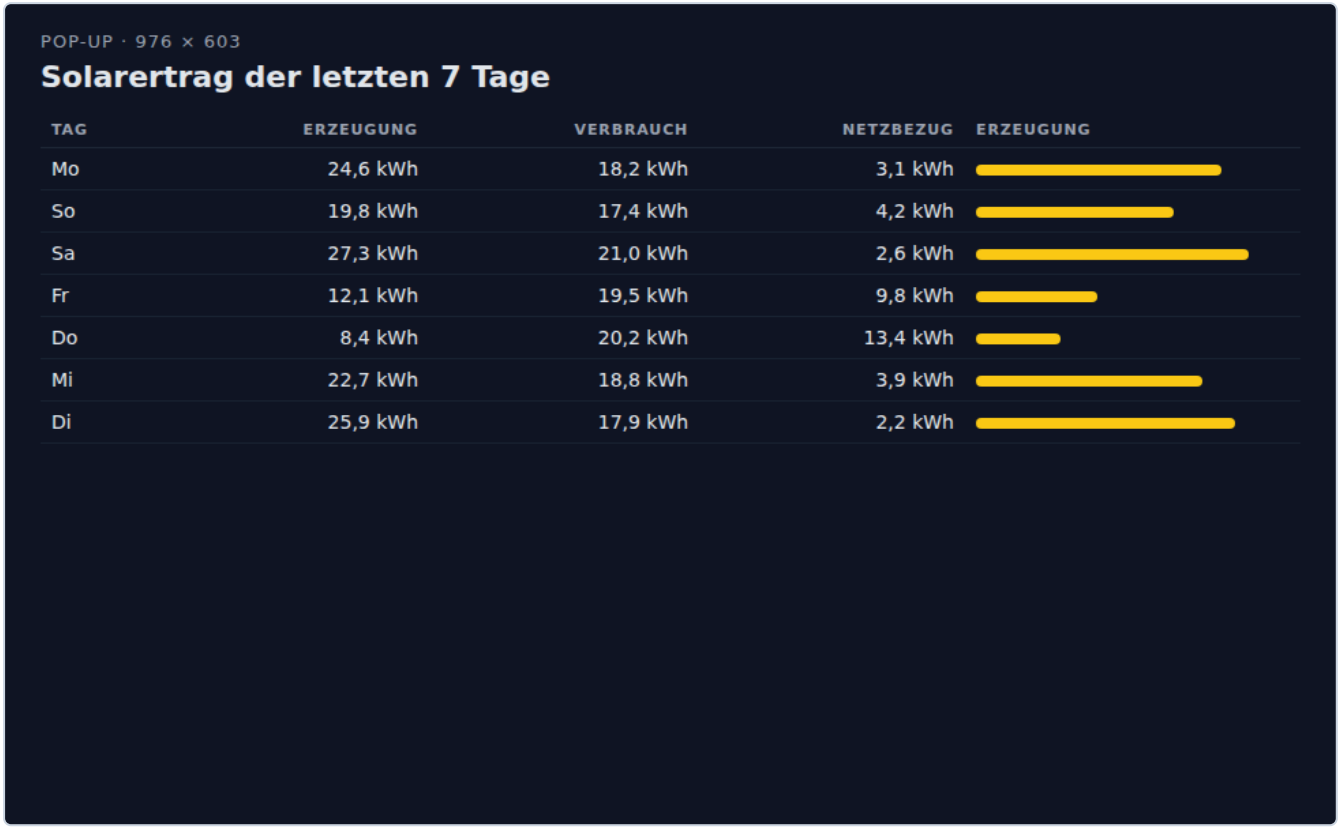


Bild 5: Eine Pop-up-Seite (976 × 603), wie sie später in IPSView erscheint.

Zwei Wege zum eigenen Simulator:

- **Mit der Vorlage:** Du lädst die Datei `flaechensimulator_vorlage.html` aus dem Anhang in den Chat und schreibst: „Das ist ein Flächen-Simulator für IPSView. Bau mir daraus meinen eigenen mit meinem Band und meinen Seiten. Alle Änderungen machen wir ab jetzt dort.“
- **Von Grund auf:** Schreib einfach: „Bau mir einen Flächen-Simulator für IPSView auf dem iPad, 1366 × 846, mit Raster, Maßtabelle und Pop-up-Fenstern.“ Claude erzeugt die Datei und legt sie in deinem Notizordner ab.

Danach arbeitet ihr nur noch im Simulator: „Zeig mir die Seite Energie im Simulator“, „Schieb das Pop-up weiter nach rechts“. Claude passt die Datei an und du lädst sie neu.

7. Was Claude in IP-Symcon ablegt

Du musst das nicht selbst anlegen. Es hilft aber zu wissen, was Claude wo ablegt, damit du den Objektbaum übersichtlich hältst und deine Sicherungen darauf abstimmen kannst. Für jede Seite entsteht dieselbe kleine Gruppe von Objekten, am besten gesammelt in **einer eigenen Kategorie** (zum Beispiel „Visualisierung“) mit einer Unterkategorie je Seite:

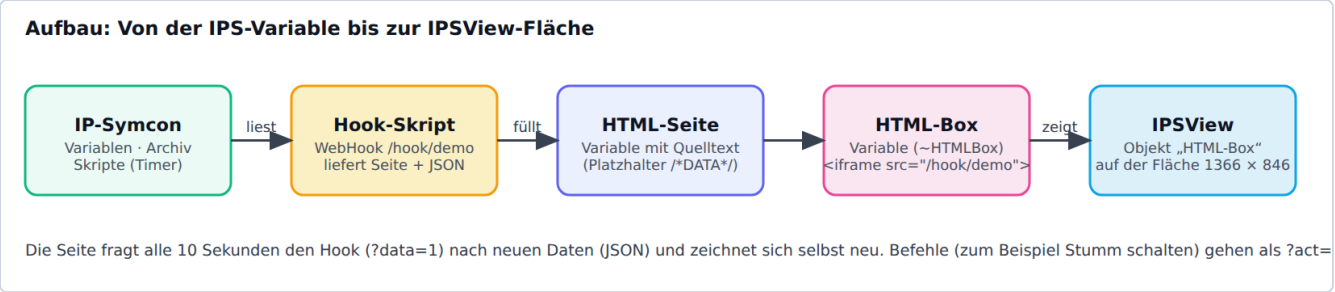


Bild 6: Von der IPS-Variable bis zur IPSView-Fläche.

Objekt in IP-Symcon	Art	Wozu
Seite (zum Beispiel „Energie_Seite“)	String-Variable	enthält die fertige HTML-Seite als Text
Hook (zum Beispiel „Energie_Hook“)	Skript	liefert die Seite aus und gibt Live-Daten und Befehle weiter
Eintrag im WebHook Control	Eintrag in der Kern-Instanz „WebHook Control“	macht die Adresse `/hook/energie` erreichbar
HTML-Box (zum Beispiel „Energie_Box“)	String-Variable mit Profil `~HTMLBox`	zeigt die Seite an, das ist das, was du in IPSView einhängst
Prüf-Skript (nur bei Meldungen)	Skript mit Timer (zum Beispiel 60 s)	prüft Werte und erzeugt Hinweise und Warnungen
Schnappschuss (nur bei Meldungen)	String-Variable	aktuelle Meldungen als Liste
Stumm-Liste (nur bei Meldungen)	String-Variable	merkt sich, welche Meldungen stumm geschaltet sind

Dazu gehört auch, was Claude außerhalb von IP-Symcon ablegt:

- **Sicherungen** der betroffenen Skripte und Variablen vor jeder Änderung, im Notizordner auf deinem Rechner.
- **Notizen**: Objekt-IDs, Namen, getroffene Entscheidungen, offene Punkte. Damit macht Claude im nächsten Gespräch genau dort weiter.
- Die **Simulator-Datei** mit allen Seiten, immer auf dem aktuellen Stand.

Tipp: Bitte Claude, nach jeder Seite eine kurze Übersicht zu geben: „Welche Objekte hast du angelegt und wie heißen sie?“ Dann siehst du sofort, was neu ist.

8. Einbindung in IPSView über HTML-Boxen

Das ist der einzige Teil, den du selbst im **IPSView-Editor** machst. Claude kann den Editor nicht bedienen, sagt dir aber genau, was du eingeben sollst.

1. **Seite im Browser prüfen.** Claude öffnet die Hook-Adresse im Browser und zeigt dir ein Bild. Du kannst es auch selbst tun: `http://<IPS-Adresse>:3777/hook/energie`. Erst wenn die Seite dort richtig mit Live-Daten erscheint, geht es weiter.
2. **HTML-Box in die IPSView-Seite setzen.** Im Editor das Objekt **„HTML-Box“** auf die Fläche ziehen.
3. **Variable verknüpfen:** die HTML-Box-Variable, die Claude dir nennt (zum Beispiel „Energie_Box“).
4. **Position und Größe eintragen:** die Zahlen aus der Maßtabelle im Simulator. Beispiel: Band 0/0 mit 1366 × 183, Hauptseite 0/183 mit 1366 × 663.
5. **Projekt speichern und aufs iPad übertragen.**

Mehr ist es nicht. Der Inhalt der Box (die Seite) liegt in IP-Symcon. Wenn Claude später etwas an der Seite ändert, siehst du es nach dem Neuladen auf dem iPad, **ohne das IPSView-Projekt zu ändern**. Du fasst den Editor nur an, wenn sich Position, Größe oder Klickflächen ändern.

Hinweis: Die Bezeichnungen im IPSView-Editor können je nach Version etwas abweichen. Wichtig sind Variable, Position und Größe.

9. Eine Box oder zwei Boxen?

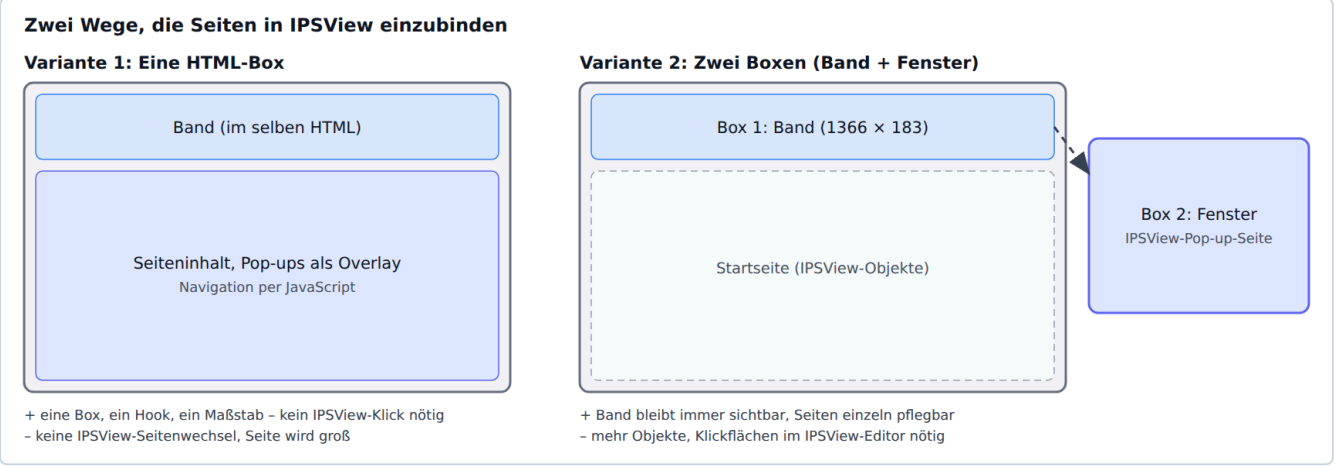


Bild 7: Die beiden Wege im Vergleich.

Variante 1: Eine einzige HTML-Box. Band, Seiteninhalt und Pop-ups liegen im selben HTML. Die Navigation macht die Seite selbst, ein Pop-up ist eine eingeblendete Fläche. In IPSView brauchst du **ein** Objekt und keine Klickflächen. Das ist der einfachste Weg, wenn du eine Hauptseite hast.

Sag Claude: *„Bau alles in eine einzige HTML-Box für 1366 × 846. Ein Tipp auf einen Button öffnet die Seite in der Box.“*

Variante 2: Zwei Boxen (Band und Fenster). Das Band liegt als eigene HTML-Box (1366 × 183) oben auf der IPSView-Startseite. Jede Seite ist eine eigene IPSView-**Pop-up-Seite** mit

eigener HTML-Box (zum Beispiel 976 × 603). Aufgerufen wird sie durch eine transparente Fläche über dem Band-Button (nächstes Kapitel). Vorteil: Das Band bleibt immer sichtbar und jede Seite lässt sich einzeln pflegen.

Sag Claude: *„Bau das Band als eigene Box und jede Seite als eigene Pop-up-Box. Sag mir, wo ich im Editor die Flächen anlegen muss.“*

In beiden Fällen ist der Inhalt gleich. Du entscheidest nur, wie du ihn in IPSView einhängst. Ein späterer Wechsel von einer Variante zur anderen ist möglich, Claude setzt dieselben Seiten nur anders zusammen.

10. Pop-ups und Schalter über transparente Flächen



Bild 8: Die HTML-Box zeigt nur an. Unsichtbare IPSView-Schaltflächen liegen genau über dem Button.

Die HTML-Box ist **nur eine Anzeige**. Damit ein Tipp etwas auslöst, legst du in IPSView eine **Schaltfläche ohne Beschriftung und ohne sichtbaren Hintergrund** (transparent) exakt über den Button und hängst die Aktion daran. Claude gibt dir pro Fläche Position und Größe, zum Beispiel „Fläche Küche: x 112, y 68, 96 × 56“. Im Simulator sind die Flächen als blaue Zonen eingezeichnet, du siehst also vorher, wo sie liegen.

- **Pop-up öffnen:** Aktion „Seite öffnen“ als Pop-up. Ziel ist die Pop-up-Seite mit der HTML-Box der jeweiligen Seite.
- **Schalten:** Aktion „Variable schalten“ auf die Schaltvariable (Lampe, Steckdose, Szene). Den Zustand zeigt der Button im Band an, zum Beispiel leuchtet die Kachel gelb, wenn die Lampe an ist.
- **Größe des Pop-ups:** Es muss nicht die ganze Fläche füllen. Meines ist 976 × 603 und liegt mittig.

Der Gedanke dahinter: Die **Optik** und die Anzeige des Zustands kommen aus dem HTML (das Claude baut). Die **Aktion** bleibt in IPSView. Dadurch kann die HTML-Box nichts in deinem System verstellen.

Sag Claude: *„Ich möchte drei Schalter im Band: Stehlampe, Kaffeemaschine und Gute Nacht. Der Zustand soll leuchten. Gib mir die Maße für die transparenten Flächen.“*

11. Überwachung und Meldungen im Button

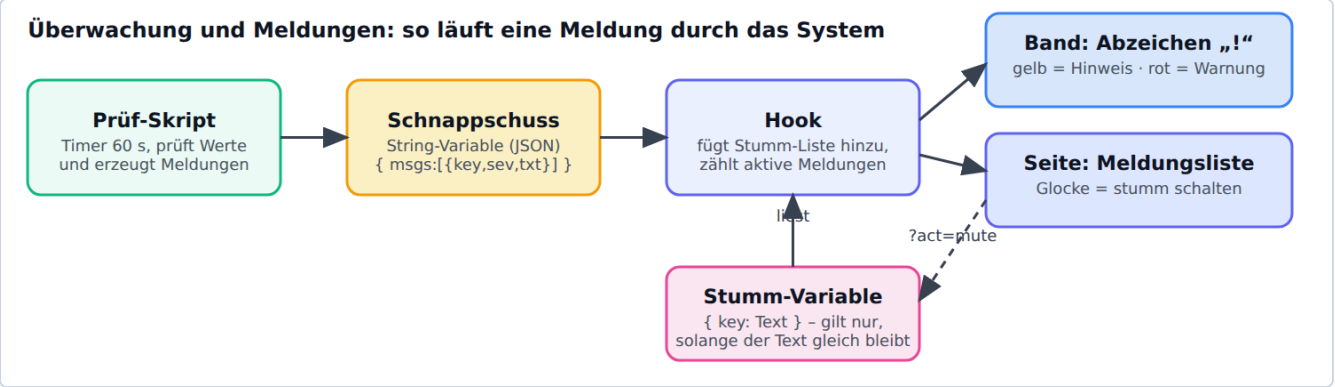


Bild 9: So läuft eine Meldung vom Prüf-Skript bis ins Band und in die Seite.

Das ist der Teil, auf den ich am meisten Wert lege. Ziel: Du siehst im **Band auf einen Blick**, ob irgendwo etwas nicht stimmt, und kannst Meldungen, die du kennst, ruhig stellen.

Die Regeln (gelten für alle Buttons gleich):

- **Gelb = Hinweis**, **Rot = Warnung oder Störung.**
- Am Button erscheint oben rechts ein **„!“-Abzeichen**. Rote Warnungen pulsieren, gelbe Hinweise nicht.
- In der Seite steht die **Meldungsliste** mit Text, Dauer und einer **Glocke**. Ein Tipp auf die Glocke schaltet die Meldung **stumm**: Sie bleibt in der Liste sichtbar, zählt aber nicht mehr für das „!“ im Band.
- Stumm geschaltete Meldungen bleiben am Button als **schwaches Merkzeichen** sichtbar: gestrichelter Ring mit „!“, in abgedunkelter Farbe, mit Zähler bei mehreren. So vergisst du sie nicht.
- Die Stummschaltung gilt **nur, solange der Meldungstext gleich bleibt**. Ändert sich die Meldung, ist sie wieder aktiv, damit nichts Neues verschluckt wird.

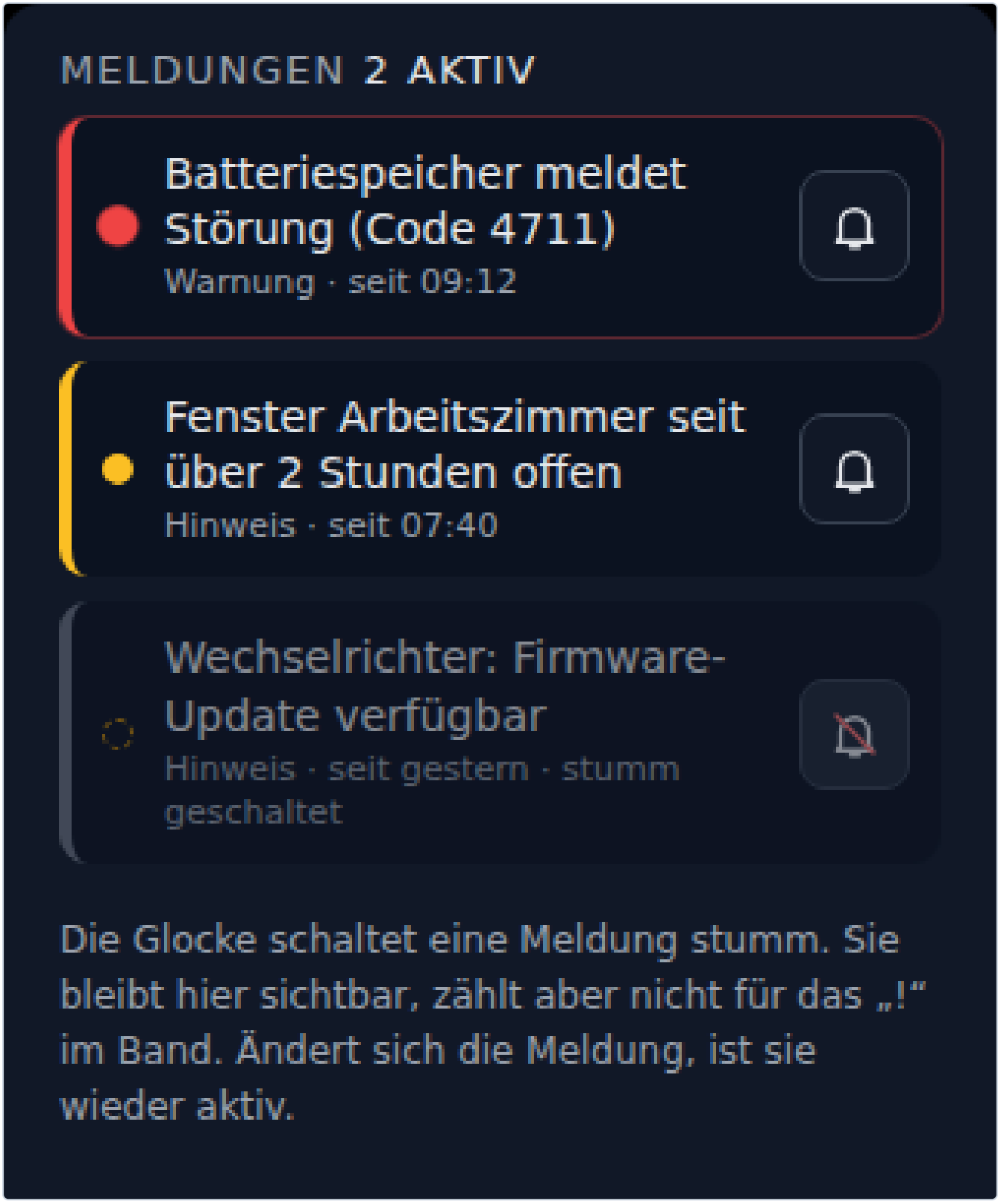


Bild 10: Das Meldungsfenster einer Seite: eine Warnung (rot), ein Hinweis (gelb) und eine stumm geschaltete Meldung.

Wie du das bei Claude bestellst: Du sagst nur, was überwacht werden soll und wie ernst es ist. Claude findet die Variablen, legt das Prüf-Skript an und verbindet es mit dem Band. Beispiele:

- *„Ich möchte eine Warnung, wenn der Batteriespeicher eine Störung meldet.“*
- *„Ein Hinweis, wenn ein Fenster länger als zwei Stunden offen ist.“*
- *„Eine Warnung, wenn der Server seit 30 Minuten keine Messwerte liefert.“*

- *„Alle Meldungen vom Wechselrichter sollen im Button Heizung/Energie erscheinen.“*
- *„Die Meldung zum Firmware-Update ist mir egal. Schalte sie stumm, bis sich etwas ändert.“*

Claude zeigt dir die Meldung zuerst im Simulator mit Demowerten (gelbes „!“, rotes „!“, stumm), damit du die Wirkung siehst, bevor irgendetwas in IP-Symcon geschrieben wird.

Wichtig: Die Schwellen („länger als zwei Stunden“) sind deine Entscheidung. Claude schlägt sie vor, du bestätigst. So gibt es keine versteckten Grenzwerte.

12. Klimaanlage und Fenster im Button



Bild 11: Die Zustände eines Raum-Buttons im Band.

Jeder Raum-Button zeigt zusätzlich zum Namen kleine Symbole:

Zustand	Anzeige im Button
Klima **heizt**	kleine orange Flamme
Klima **kühlt**	blaue Schneeflocke
Fenster offen	gelber Kreis mit der Anzahl offener Fenster
Klima läuft bei offenem Fenster	rot blinkende Warnung
Hinweis	gelbes „!“
Warnung	rotes pulsierendes „!“
Meldung stumm	gestrichelter Ring mit „!“

Die Daten dafür stammen aus deinen vorhandenen Variablen (Klimagerät Betriebsart, Fensterkontakte). Du sagst Claude nur, **welcher Raum welche Geräte hat**. Beispiel: *„Im Arbeitszimmer gibt es eine Klima und zwei Fensterkontakte. Zeig im Button, wenn die Klima heizt oder kühlt und wenn ein Fenster offen ist. Wenn beides gleichzeitig ist, soll es rot blinken.“* Claude sucht die Variablen heraus, legt dir die Zuordnung zur Kontrolle vor und baut sie in das Band ein.

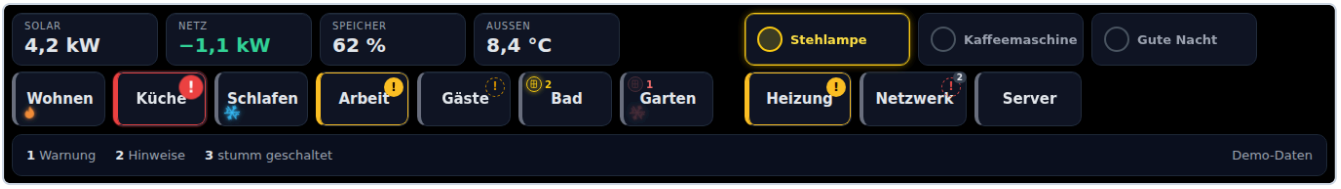


Bild 12: Das Demo-Band mit Kennzahlen, Raum-Buttons, Schaltern und Meldungsleiste.

13. Beispiele aus meinem Projekt

Zum Schluss vier echte Seiten aus meinem Haus, jede ist im Gespräch mit Claude entstanden. Namen von Personen und Räumen sind für die Bilder durch neutrale Bezeichnungen ersetzt, die Werte sind Momentaufnahmen.

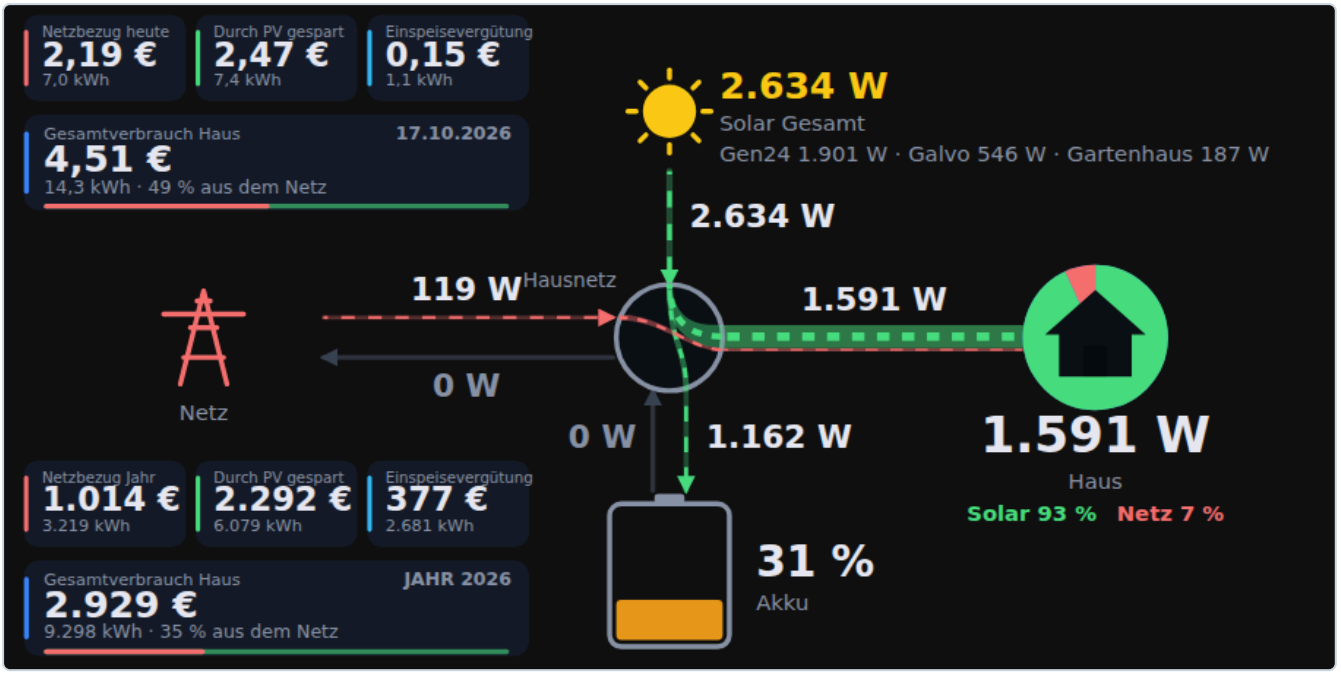


Bild 13: Flussdiagramm Energie. Solar, Hausnetz, Netz, Akku und Haus mit animierten Flüssen und Tages- und Jahreskosten. Bestellt mit: „Zeig mir den Energiefluss wie ein Flussdiagramm. Die Linien sollen in die Richtung fließen, in die die Energie geht, und mit der Leistung dicker werden.“

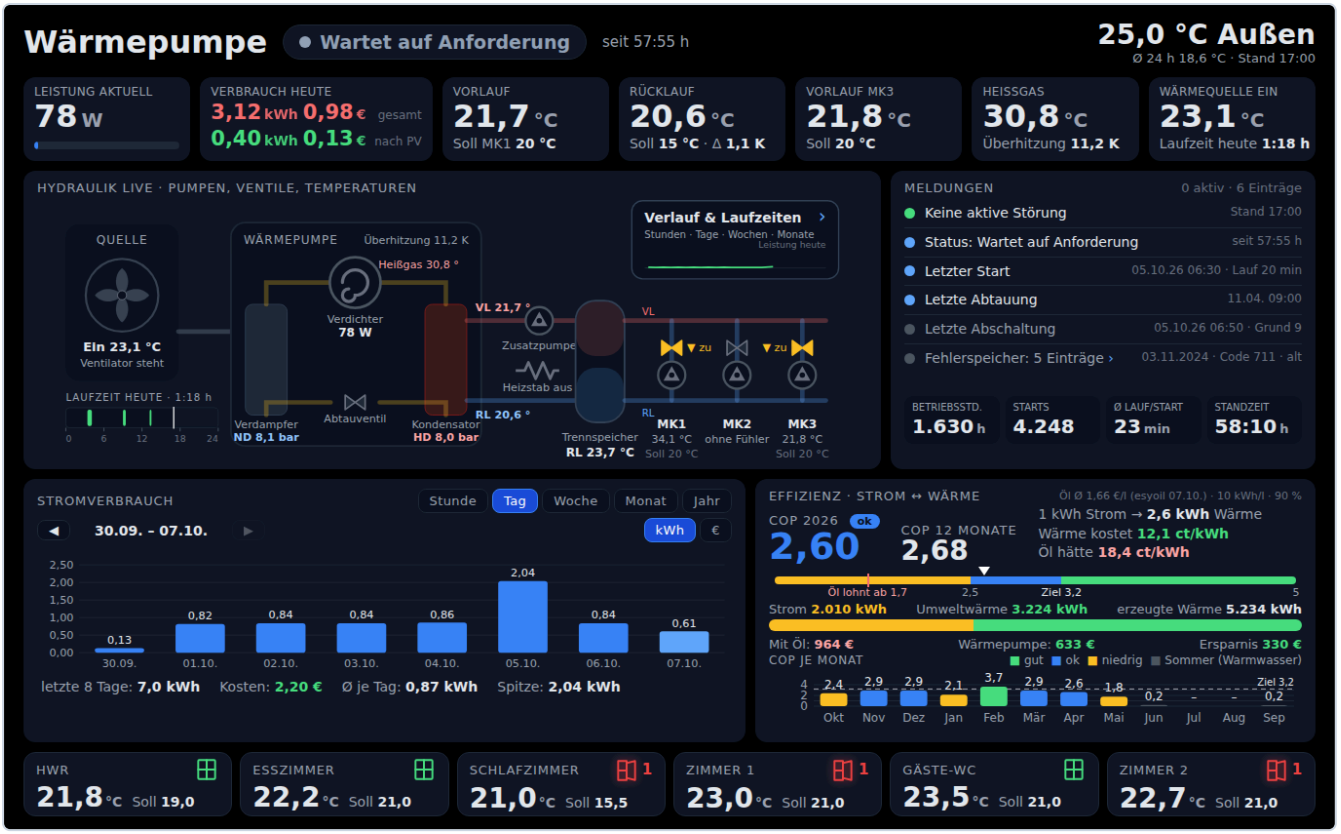


Bild 14: Wärmepumpe. Betriebszustand, Temperaturen, Hydraulik live, Meldungen, Stromverbrauch und Effizienz (COP) und unten die Raumtemperaturen mit Fensteranzeige. Bestellt mit: „Mach eine Wärmepumpen-Seite mit Hydraulikschema und Effizienz. Zeig, ob sich die Wärmepumpe gegenüber Öl lohnt.“



Bild 15: Meldungsseite für KNX und Homematic. Busspannung, Netzteile, Heizungsaktoren, Geräteüberwachung und alle nicht erreichbaren Homematic-Geräte, oben die Hinweise mit Glocke zum Stummschalten. Bestellt mit: „Ich will auf einer Seite sehen, ob am KNX-Bus und bei Homematic alles in Ordnung ist. Meldungen sollen im Band als gelbes oder rotes Ausrufezeichen erscheinen.“

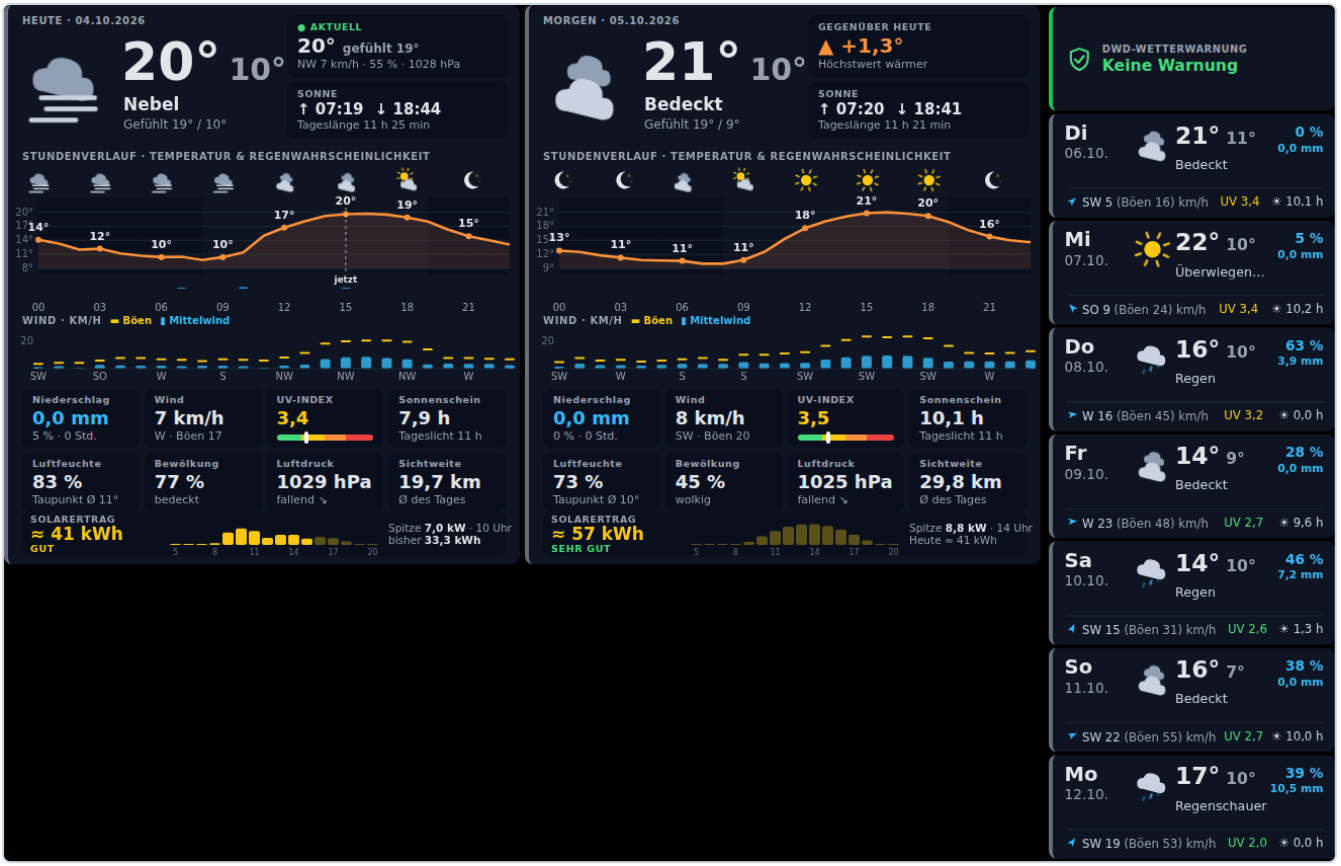


Bild 16: Wetterseite. Heute und morgen mit Stundenverlauf, Wind, Niederschlag, UV, Sonne und erwartetem Solarertrag, rechts die Vorhersage für sieben Tage und die DWD-Warnung. Bestellt mit: „Ich möchte eine Wetterseite mit Stundenverlauf und dem Solarertrag, der morgen zu erwarten ist.“

14. Formulierungen, die gut funktionieren

- **Erst schauen, dann ändern:** „Schau dir nur an, welche Variablen es zum Thema Energie gibt, und mach mir einen Vorschlag. Ändere nichts.“
- **Konkret beschreiben:** „Oben links die Solarleistung in Gelb, groß. Daneben der Netzbezug, grün bei Einspeisung.“
- **Rückfragen erlauben:** „Wenn dir eine Variable nicht eindeutig ist, frag mich.“
- **Freigabe geben:** „Ja, übernimm das in IP-Symcon.“ Erst danach schreibt Claude.
- **Kontrollieren lassen:** „Prüfe, ob die Seite in IP-Symcon genau dem Simulator entspricht.“
- **Rückgängig machen:** „Stelle den Stand von vor der letzten Änderung wieder her.“ Das geht, weil Claude vorher sichert.
- **Aufräumen:** „Zeig mir alle Objekte, die du für diese Seite angelegt hast.“

15. Grenzen, Datenschutz und Stolpersteine

- **Claude macht Fehler.** Deshalb: erst Simulator, dann Freigabe, immer Sicherung. Prüfe wichtige Werte (Kosten, Zähler) gegenüber deinen eigenen Zahlen.
- **Datenschutz:** Was Claude liest, wird Teil des Gesprächs mit Anthropic. Gib keine Passwörter, Zugangsdaten oder Personendaten in den Chat. Nutze für Claude einen eigenen IP-Symcon-Benutzer. Lies die Datenschutzhinweise von Anthropic.

- **Schreibzugriff ist mächtig.** Er erlaubt auch Skripte. Schalte ihn bewusst ein und lass ihn aus, wenn du nur Fragen stellst.
- **Der IPSView-Editor bleibt Handarbeit.** Claude baut die Inhalte, du setzt Boxen und Flächen.
- **Alles nur im Heimnetz.** Die Seiten laufen über den IP-Symcon-Webserver. Wer sie von außen nutzen will, braucht seinen eigenen sicheren Zugang. Das ist nicht Teil dieser Anleitung.
- **Nur iPad.** Diese Anleitung beschreibt nur die iPad-Fläche (1366 × 846). iPhone-Ansichten sind ein eigenes Thema.
- **Nach Updates prüfen.** Nach einem IP-Symcon- oder IPSView-Update die Seiten einmal ansehen. Claude kann das mit dir durchgehen.

16. Checkliste

1. IP-Symcon-Sicherung angelegt, eigener Benutzer für Claude angelegt.
2. MCP-Server eingerichtet, Test „Variable lesen“ klappt.
3. Spielregeln an Claude gegeben, Notizordner festgelegt.
4. Flächen-Simulator gebaut (Vorlage oder von Grund auf).
5. Band und erste Seite im Simulator fertig, von dir abgenommen.
6. „Ja“ gesagt, Claude hat gesichert und in IP-Symcon abgelegt.
7. Hook-Adresse im Browser geprüft.
8. HTML-Box im IPSView-Editor gesetzt (Variable, Position, Größe).
9. Transparente Flächen für Pop-ups und Schalter angelegt (nur bei Variante 2).
10. Meldungen bestellt und im Simulator geprüft.

17. Anhang (Dateien)

Datei	Inhalt
`flaechensimulator_vorlage.html`	Simulator zum Nachbauen: Doppelklick im Browser, Seiten und Klickflächen verschieben
`demo_band.html`	Demo-Band 1366 × 183 mit Raster, Buttons, Schaltern, Meldungen
`demo_energie.html`	Demo-Seite 1366 × 846 mit Kacheln, Diagramm und Meldungsliste
`demo_popup.html`	Demo-Pop-up 976 × 603
`demo_home.html`	Übersicht 1366 × 663 für die Startseite
`demo_komplett.html`	Variante 1: alles in einer Box

Die Demo-Dateien lädst du in den Chat und sagst: *„So ähnlich soll es bei mir aussehen. Bau es mit meinen Variablen.“* Claude nutzt sie als Vorlage.

Ich freue mich über Rückmeldungen und Fragen. Wenn Interesse besteht, zeige ich gern weitere Seiten (Raumseite mit Heizung und Rollläden, Wärmepumpe, Netzwerk-Übersicht, Solar).