ionpy Framework Erweiterungs-Spezifikation

Dieses Dokument beschreibt die geplanten Erweiterungen für das ionpy-Framework zur Verbesserung der Benutzerinteraktion, Datenstrukturierung und Automatisierung.

1. Synchronisations- & Sperrstrategie (Race Condition Protection)

Um Konflikte zwischen automatischem Polling und Benutzereingaben zu vermeiden, wird eine zweistufige Sperre implementiert.

1.1 Backend: Mute-Timer (AbstractDevice)

[cite_start]In der Basisklasse `AbstractDevice` wird ein Zeitstempel-Tracking eingeführt[cite: 117].

- **Mechanismus**: Bei Ausführung von `execute_command` wird ein Mute-Timer (default 3s) für die spezifische `entity_id` gesetzt.
- [cite_start]**Logik**: Die Methode `update_entity` verwirft eingehende Polling-Daten, solange der Timer aktiv ist[cite: 128].
- **Ziel**: Verhindert das Zurückspringen von Werten in der UI, während die Hardware den neuen Wert noch verarbeitet.

1.2 Frontend: Focus-Lock (Web-UI)

Universelle Sperre im JavaScript-Frontend (z.B. in `settings.html`).

- **Mechanismus**: Ein zentrales `Set` speichert die IDs aller Felder, die aktuell den Fokus haben (Event-Delegation via `focusin`).
- **Logik**: Eingehende WebSocket-Updates werden ignoriert, wenn die Element-ID im `Set` registriert ist.
- **Beispiel**: Während ein User “12.5” in ein Feld tippt, darf ein eintreffendes Sample das Feld nicht auf “10.0” zurücksetzen.

2. Strukturierte Daten: TableEntity

Einführung einer komplexen Entität zur Abbildung von Speicherplätzen, Profilen und Sequenzen.

- **Struktur**:
  - `columns`: Definition der Spalten (Name, Typ, Unit, Constraints).
  - `value`: Eine Liste von Dictionaries (Rows).
- **Beispiel (UDP3305 Sequenzer)**:
  - Spalten: `Step`, `Voltage (V)`, `Current (A)`, `Duration (s)`.
- **Interaktion**: Das Frontend sendet bei Änderung ein Koordinaten-Objekt: `{ “row”: 0,`

“col”: “volt”, “val”: 12.0 }.

3. Gamepad Integration (Human Interface Device)

Anbindung von Gamecontrollern zur haptischen Steuerung von Laborgeräten auf Basis von pygame.

3.1 GamepadManager & Discovery

Ein neuer System-Treiber scannt mittels pygame.joystick dynamisch nach angeschlossenen Controllern.

- [cite_start] **gamepad.py**: Erbt von AbstractDevice[cite: 116].
- **GamepadEntity**: Kapselt den gesamten Zustand (Achsen, Buttons, Hats) in einem einzigen Objekt.

3.2 Visualisierung

- [cite_start]**Achsen**: Darstellung als Fadenkreuz oder analoge Level-Balken (analog zu NumericEntity mit UIMode.LEVEL)[cite: 154].
- **Buttons**: Darstellung als virtuelle Status-LEDs.

4. LogicService & Automation Bridge

Der zentrale Dienst zur Verknüpfung von Events (z.B. Gamepad → Netzteil).

- [cite_start]**Funktionsweise**: Ein asynchroner Task in der Engine abonniert den EventBus[cite: 86, 427].
- **Regel-Engine**: Vergleicht eintreffende Samples mit einer JSON-basierten Regelliste.
- **Beispiel-Regel**:

```
{
  "trigger": { "device": "gamepad_0", "entity": "axis_1" },
  "action": { "device": "psu_1", "entity": "ch1_v_set", "scale": 32.0 }
}
```

- **Cross-Device-Talking**: Ermöglicht es Geräten, aufeinander zu reagieren (z.B. Senke folgt der Spannung des Netzteils), ohne die Treiber-Logik zu verändern.

5. Neue Entitätstypen (Katalog)

[cite_start]Erweiterung der structures/enums.py und structures/entities.py[cite: 421, 414].

Typ	Beschreibung	Beispiel
LogEntity	Chronologischer Ereignis-Feed pro Gerät	“OVP triggered”, “Calibration OK”
StatusIndicator	Visuelle “LED”-Anzeige (Farben/Blinken)	Alarm-Status, Power-LED
FileEntity	Schnittstelle für Up/Downloads	Firmware-Update, Log-Export
XYGraphEntity	Kennliniendarstellung (X vs Y statt Zeit)	U/I-Kurve, Bode-Plot
RangeEntity	Kapselung von Start, Stopp und Step	Sweep-Definition (Frequenz)
ScheduleEntity	Zeitbasierte Steuerbefehle	“Ausgang aus um 22:00 Uhr”

6. Technische Basis

- [cite_start]**Backend**: Python 3.10+, FastAPI (Asynchron), Pygame (Gamepad)[cite: 1, 437].
- [cite_start]**Kommunikation**: JSON-Samples über asynchronen EventBus[cite: 86, 424].
- [cite_start]**Frontend**: HTML5/JS (WebSockets) für Echtzeit-Visualisierung[cite: 52, 427].