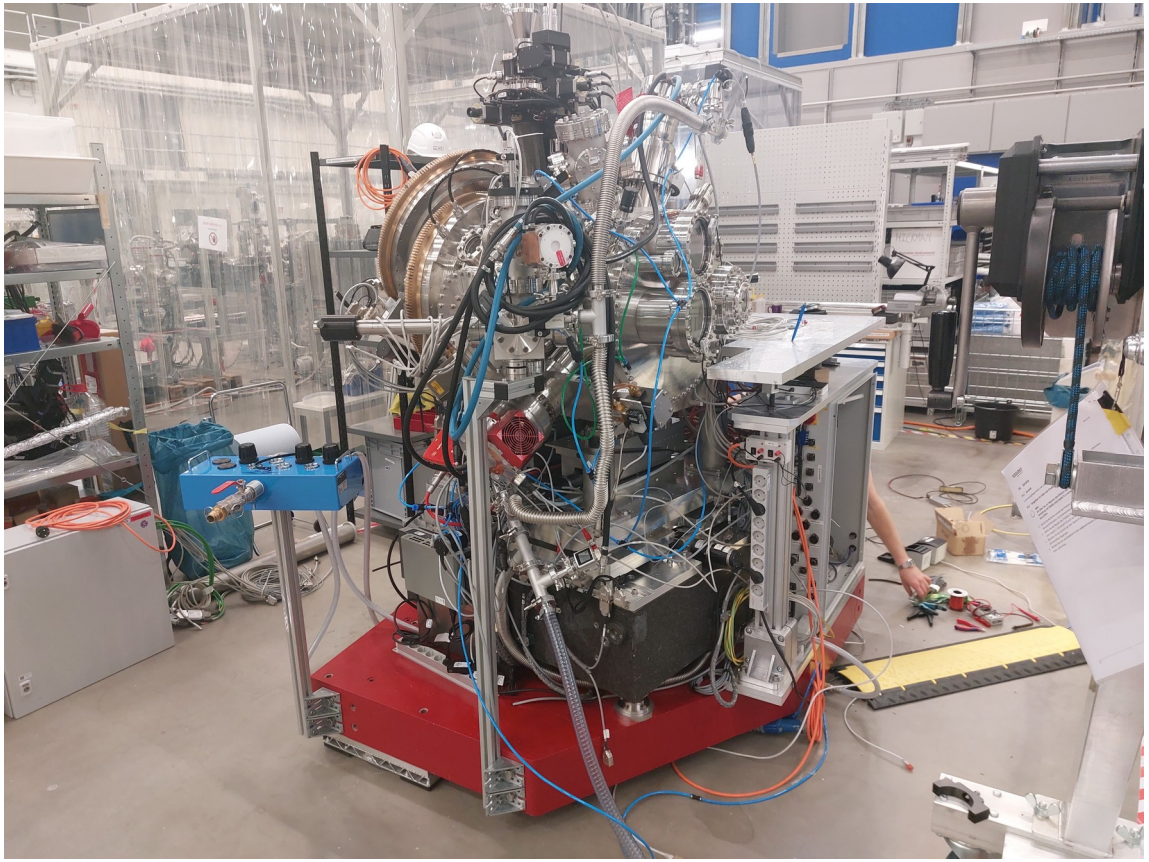
EBPFCat: Beckhoff ohne Beckhoff - Individuelle Lösungen mit Python programmiert

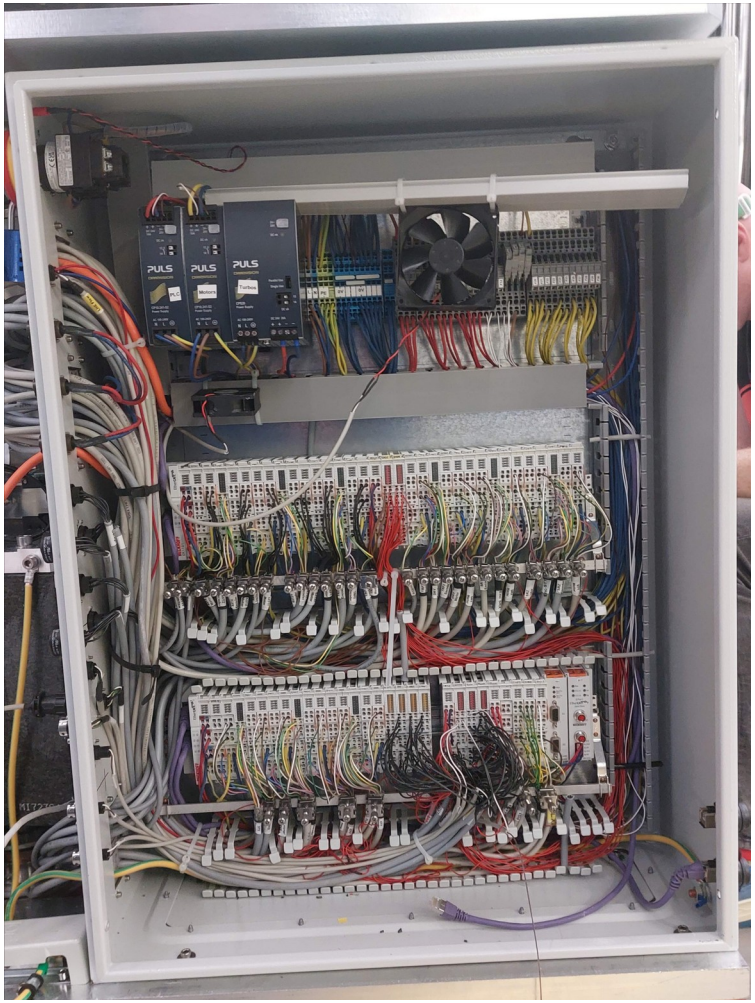
Martin Teichmann
SCS Instrument @ XFEL

Motivation



Ausserhalb von Strahlzeiten hat der experimentelle Aufbau keine Verbindung zu den XFEL SPS CPUs

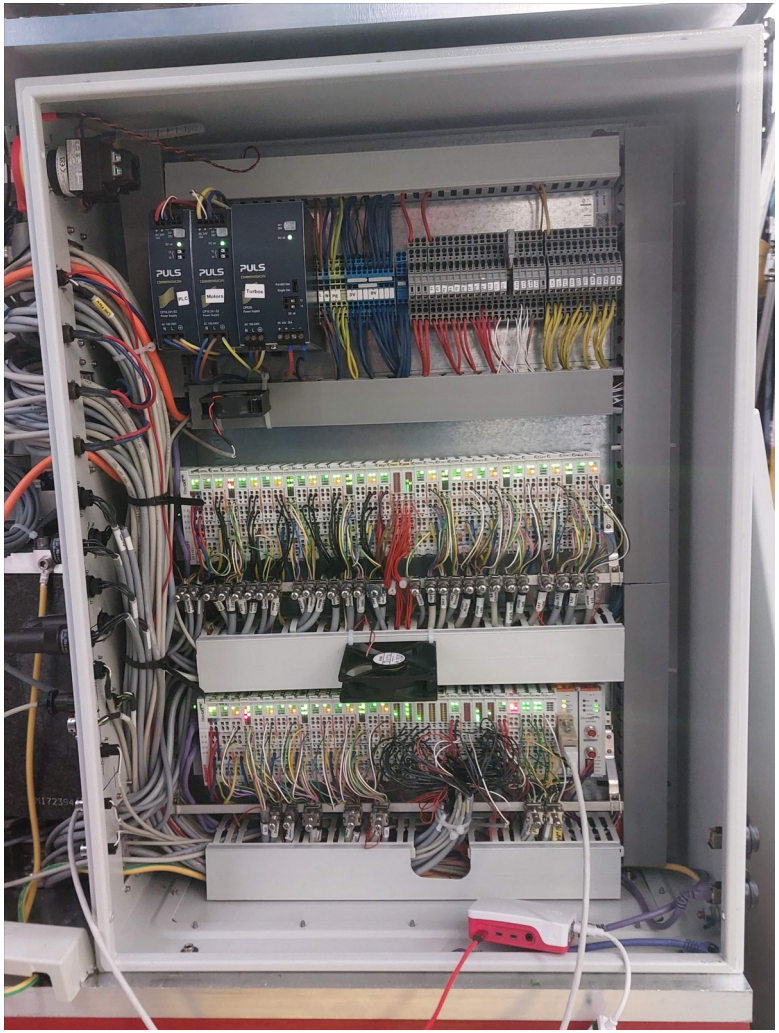
Problemstellung



Beim European XFEL verwenden wir EtherCAT-Klemmen von Beckhoff zur Experimentsteuerung.

Können wir die Elektronik ohne eine SPS CPU ansteuern?

Problemstellung

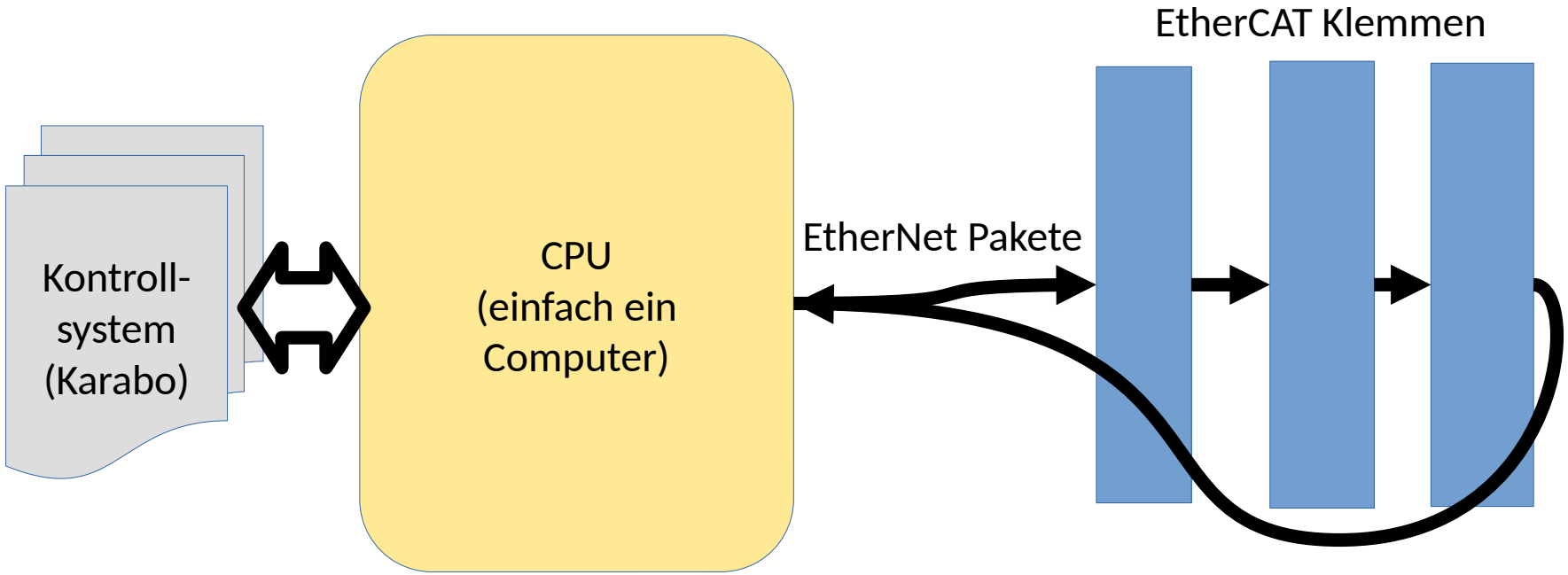


Können wir die Elektronik ohne eine SPS CPU ansteuern?

Ja!

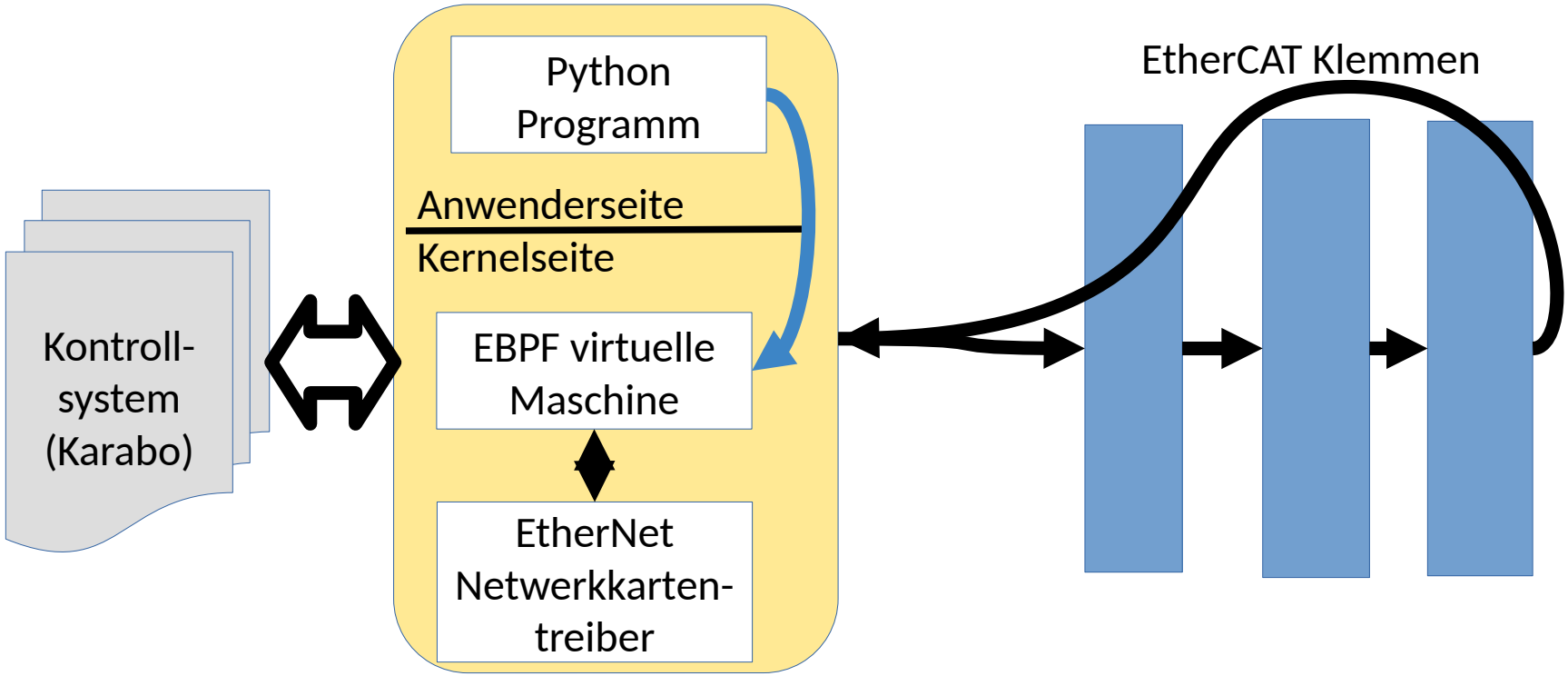
Dazu benötigen wir keinerlei Software von Beckhoff oder anderer Fremdfirmen – wir können alles mit unserem Projekt „EBPFCat“ machen

Wie funktioniert EtherCAT?



EtherCAT verwendet normale EtherNet Pakete, die von einer zur nächsten Klemme weitergereicht werden, um am Ende wieder am Anfang anzukommen.

Regelung mit EBPF



- EBPF (Extended Berkley Packet Filter) ist eine virtuelle Maschine im Linux-Kernel die direkt mit der Netzwerkkarte interagieren kann
- Wir können den Regelkreis für die SPS direkt in diese virtuelle Maschine injizieren

Schnelle und Langsame Regelung

Langsame Regelung direkt in Python

Regelschleifen mit etwa 100 Hz

Ventile
Druckmessröhren
Lasershutter
Pumpen

Auch alle Geräte, die per RS232 oder RS485 angesteuert werden

Schnelle Regelung mit EPBF

Regelschleifen bis etwa 10 kHz

Motoren (insbesondere Regelschleife zwischen Encoder und Motor)

Gasdruckregelung

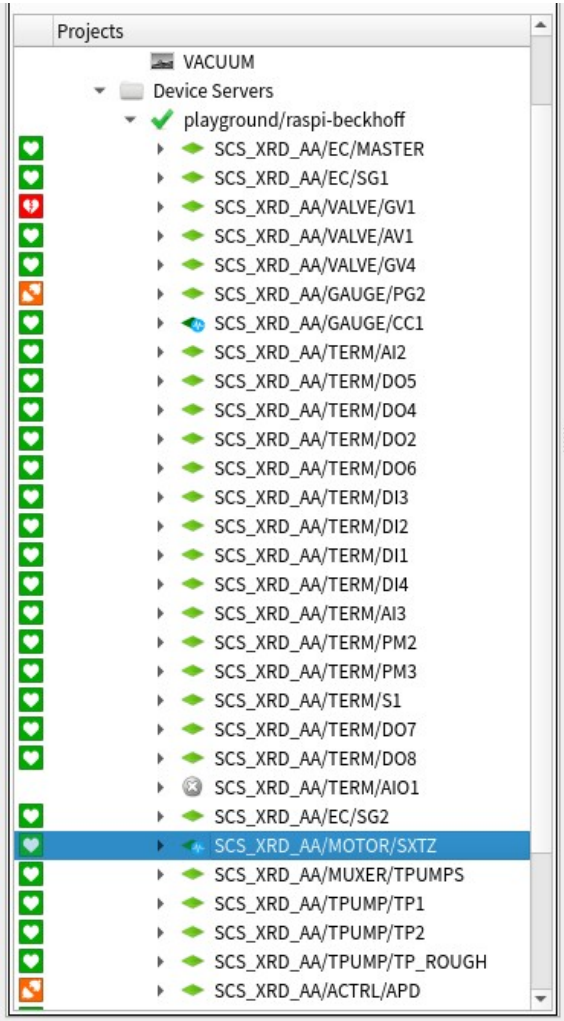
Laserstabilisierung

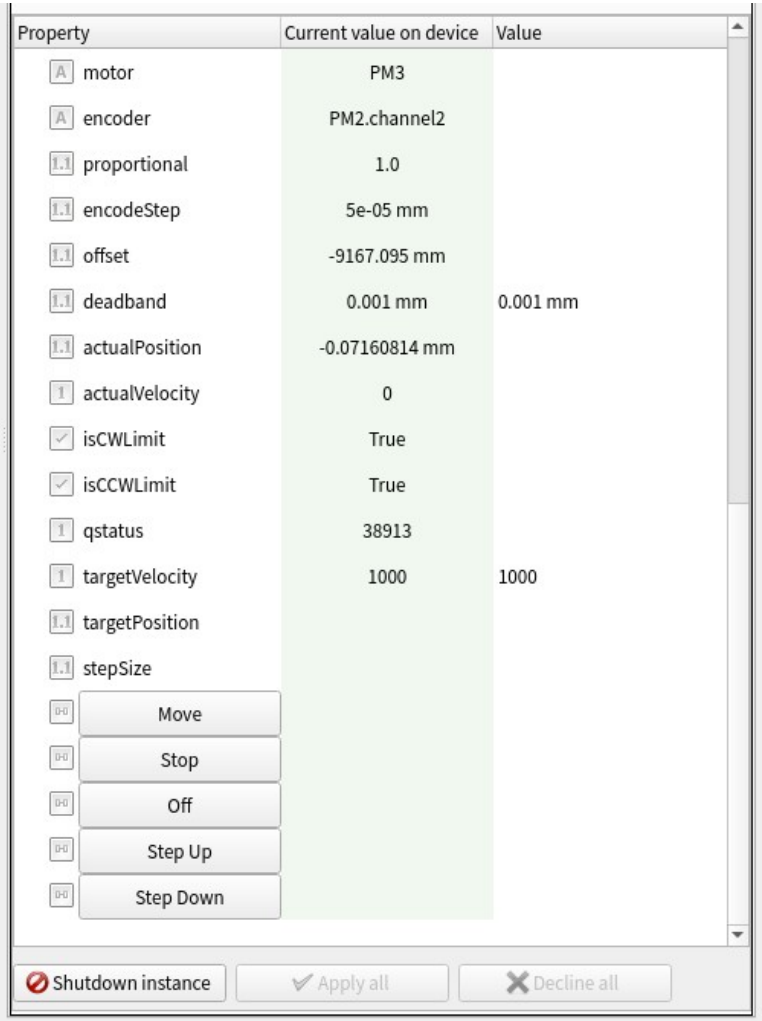
EBPF-Programme werden vom Linux-Kernel auf Korrektheit verifiziert

Adressierung via fester Nummer

- Normalerweise bestimmt allein die Position der Klemme ihre Funktion
- Dies macht Umbauten schwer: Software und Hardware müssen immer nachgeführt werden
- Eindeutige Nummer im EEPROM der Klemme: Software spricht immer mit derselben Klemme, unabhängig von ihrem Ort

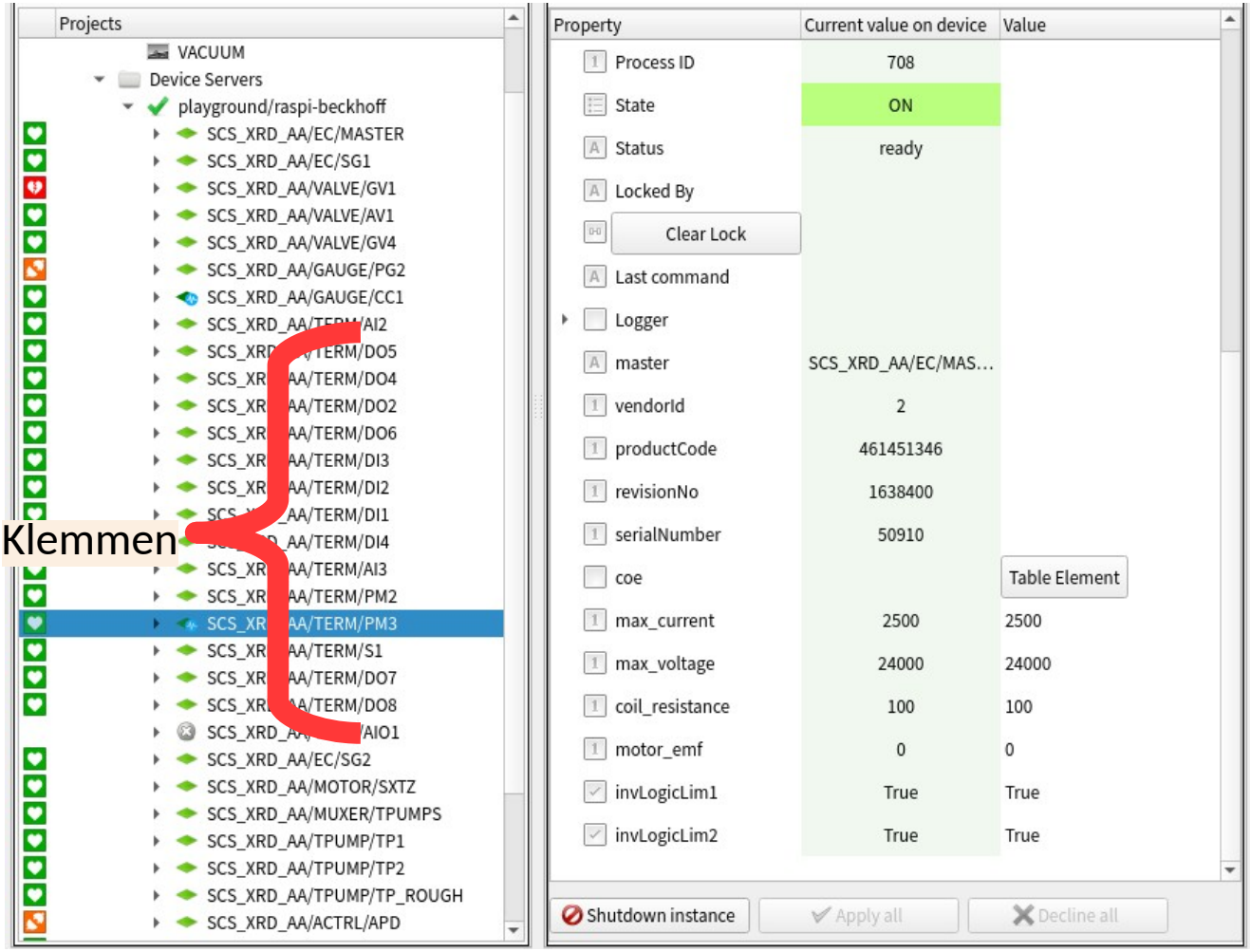
Benutzung im Kontrollsystem





Alle Funktionen können *vom Benutzer* des XFEL Kontrollsystems „Karabo“ eingerichtet werden.

Benutzung im Kontrollsystem



Für jede Klemme gibt es einen Eintrag im Kontrollsystem, sowie für jedes mit einer Klemme verbundene Gerät.

Programmierung eines Motors

```
class Motor(Device):
```

Programmierung eines Motors

```
class Motor(Device):
    velocity = TerminalVar()
    encoder = TerminalVar()

    target = DeviceVar()
    proportional = DeviceVar()
```

Programmierung eines Motors

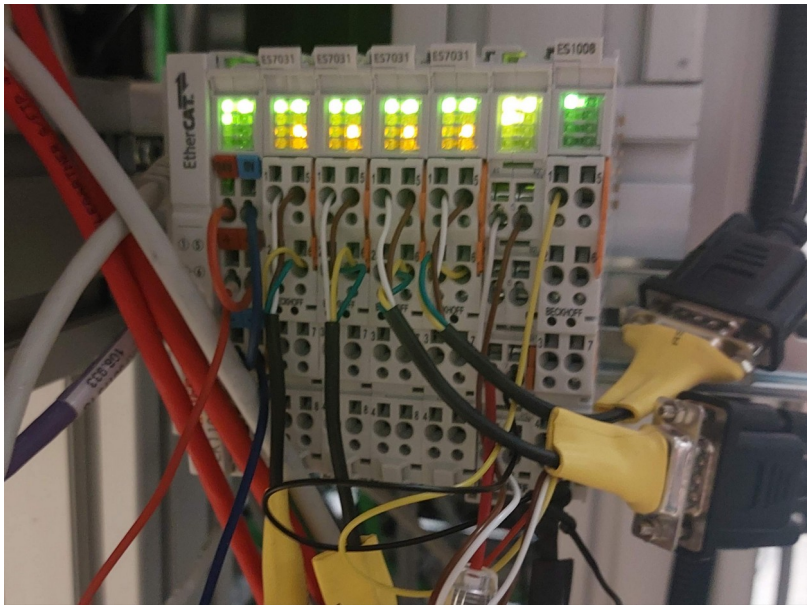
```
class Motor(Device):
    velocity = TerminalVar()
    encoder = TerminalVar()

    target = DeviceVar()
    proportional = DeviceVar()


    def program(self):
        self.velocity = (self.target – self.encoder)  \
                        * self.proportional
```

Programmierung eines Motors

```
class Motor(Device):
    velocity = TerminalVar()
    encoder = TerminalVar()

    target = DeviceVar()
    proportional = DeviceVar()
    max_velocity = DeviceVar()

    def program(self):
        self.velocity = (self.target – self.encoder)  \
                        * self.proportional

        with self.velocity > self.max_velocity:
            self.velocity = self.max_velocity
        with self.velocity < -self.max_velocity:
            self.velocity = -self.max_velocity
```

Voraussetzungen

- Ein Rechner auf dem Linux läuft
  - z.B. irgendein Laptop
  - Oder Raspberry Pi 4 - Erfahrungsgemäss das beste
- Administrator / „root“ – Rechte
- Ein Kontrollsystem obendrüber
  - Das XFEL Kontrollsystem „Karabo“ wird bereits unterstützt
  - Andere Kontrollsysteme (Tango, EPICS, DOOCS…) denkbar
- Auch sehr kleine Projekte sind einfach umsetzbar



Zusammenfassung

- Wir können Beckhoff EtherCAT Steuerungen ohne eine Beckhoff CPU steuern
- Langsame Regelkreise werden direkt in Python geschrieben, schnelle können direkt in den Linux Kernel injiziert werden
- Klemmen haben feste Nummern – das vereinfacht Änderungen

Alles ist offen verfügbar unter **`https://github.com/tecki/ebpfcat`**

Dank an meine Gruppe – SCS beim European XFEL, insbesondere Carsten Broers für die Hardware, und Jan Tolkiehn von EEE - XFEL