

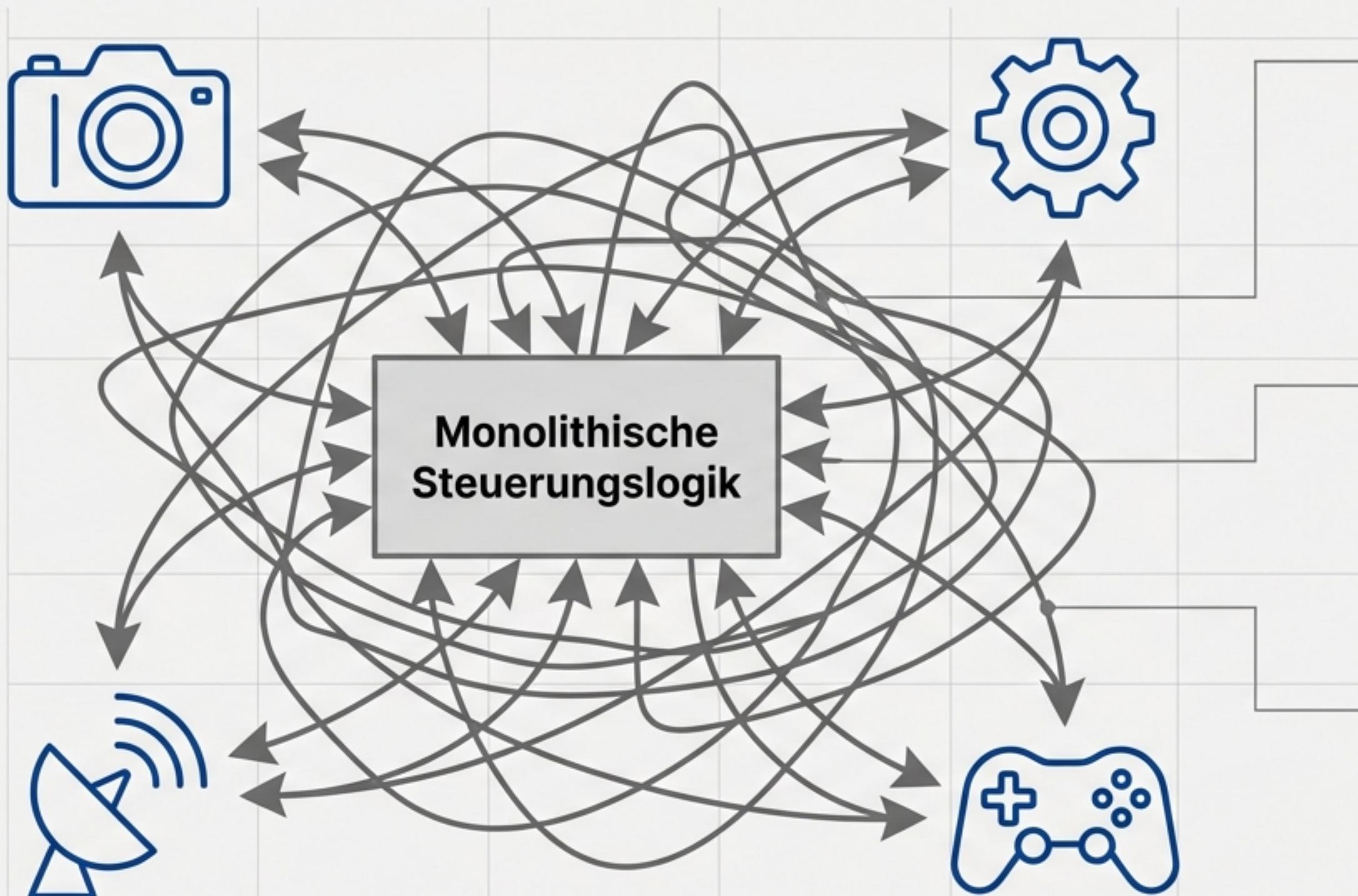
Das HTL Wien West Robotics Framework

Ein modularer, nachrichtenbasierter Ansatz für den Robocup



HTL Wien West

Die Herausforderung: Die wachsende Komplexität eines Roboters



Starre Verbindungen

Jede Komponente ist fest mit der zentralen Logik verdrahtet.
Änderungen sind aufwendig.

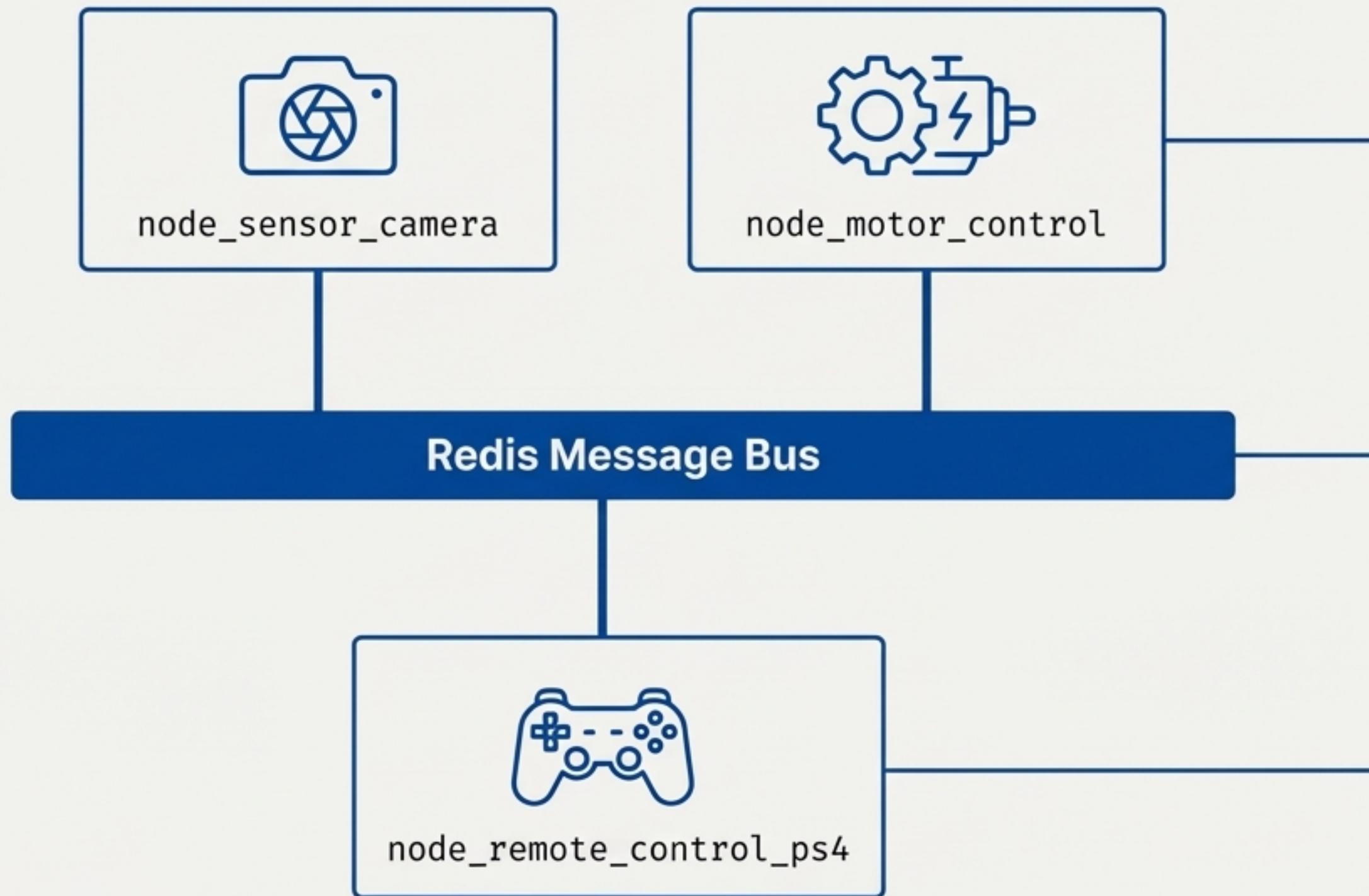
Schwer zu testen

Einzelne Teile (z.B. ein neuer Sensor)
können nicht isoliert getestet werden.
Das gesamte System muss laufen.

Geringe Wiederverwendbarkeit

Code für eine bestimmte Hardware
ist schwer für andere Roboter oder
Projekte anzupassen.

Unsere Philosophie: Entkopplung durch eine zentrale Nachrichten-Bus-Architektur



Flexible Module (Nodes)

Jeder 'Node' ist ein unabhängiges Programm. Er kann jederzeit hinzugefügt, entfernt oder ausgetauscht werden.

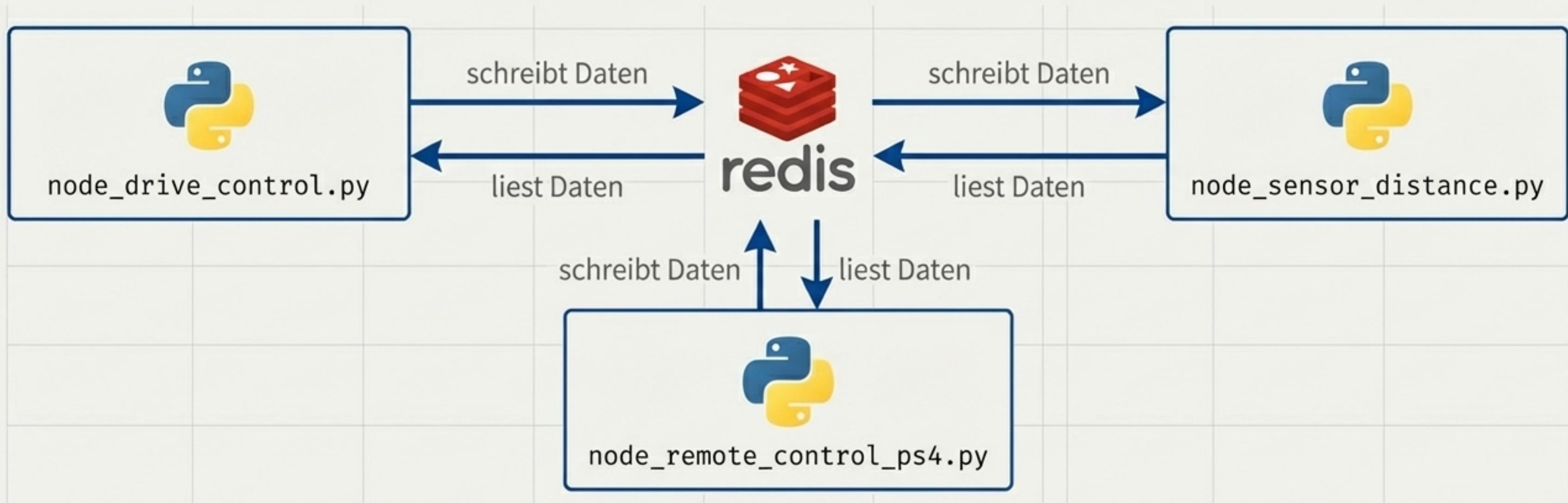
Unabhängige Entwicklung & Tests

Teams können parallel an verschiedenen Nodes arbeiten und diese isoliert testen.

Einfache Skalierbarkeit

Ein neuer Sensor? Einfach einen neuen Node schreiben, der mit dem Bus kommuniziert.

Das technische Fundament: Python Nodes & Redis Server



Nodes:

Unabhängige Python-Prozesse, die eine spezifische Aufgabe erfüllen (z.B. einen Motor ansteuern, einen Sensor auslesen).

Redis Server:

Ein extrem schneller In-Memory-Datenspeicher, der als zentraler "Nachrichten-Broker" oder "Schwarzes Brett" fungiert.

Kommunikation:

Nodes kommunizieren niemals direkt direkt miteinander. Sie schreiben und lesen Informationen ausschließlich auf bzw. vom Redis Server.

Die Sprache des Systems: Kommunikation über Schlüssel-Wert-Paare (Keys)



- Ein Node (z.B. der PS4-Controller) schreibt einen Wert ('100') unter einem eindeutigen Schlüssel ('vel_linear_x') in die Redis-Datenbank.
- Ein anderer Node (z.B. die Antriebssteuerung) liest den Wert von genau diesem Schlüssel und kann darauf reagieren.
- Dieses Prinzip ermöglicht eine vollständige Entkopplung: Der Controller-Node weiß nicht, wer seine Daten verwendet, und der Antriebs-Node weiß nicht, woher die Daten stammen.

Code in der Praxis: Ein Signal senden

Ein Sender-Node stellt eine Verbindung zu Redis her und verwendet den Befehl `r.set()`, um einen Wert zu veröffentlichen.

```
1 # node_transmitter_example.py (vereinfacht)
2
3 import redis
4
5 # Verbindung zum lokalen Redis-Server herstellen
6 r = redis.Redis(host='localhost', port=6379, db=0)
7
8 # Den Modus des Roboters auf 'autonom' setzen (1)
9 r.set('mode', 1) —————— Schlüssel 'mode' mit
10                                Wert '1' wird gesetzt.
11 # Eine Vorwärtsgeschwindigkeit von 80 setzen
12 r.set('vel_linear_x', 80)
```

Code in der Praxis: Ein Signal empfangen

Ein Empfänger-Node liest kontinuierlich Werte von Redis mit dem Befehl `r.get()`, um auf Änderungen im System zu reagieren.

```
# node_receiver_example.py (vereinfacht)

import redis
import time

r = redis.Redis(host='localhost', port=6379, db=0)

while True:
    # Lese den aktuellen Wert des Schlüssels 'vel_linear_x'
    velocity = r.get('vel_linear_x')

    if velocity is not None:
        # Code zur Ansteuerung der Motoren...
        print(f"Empfangene Geschwindigkeit: {velocity.decode('utf-8')}")

    time.sleep(0.01)
```

Wert vom Schlüssel
'vel_linear_x' wird
gelesen.

Die Bausteine des Systems: Vorgefertigte Beispiel-Nodes

Das Framework enthält eine Sammlung von einsatzbereiten Nodes für gängige Roboter-Aufgaben.



node_motor_control



node_sensor_distance



node_sensor_linear-angular



node_sensor_color



node_sensor_camera



node_remote_control_ps4



node_drive_control



node_touchdisplay_control



node_navigation_control

Die gemeinsame Sprache: Definierte Schlüssel-Typen (Keys)

Eine standardisierte Nomenklatur für Schlüssel sorgt für Interoperabilität zwischen den Nodes.

Bewegungssteuerung

vel_linear_x / y / z (-100 bis 100)

vel_angular_x / y / z (-100 bis 100)

Status & Modus

move_state (0: stop, 1: drive)

mode (0: manual, 1: autonom)

Distanzsensorik

sensor_range_front / left / right / back

IMU-Sensorik

sensor_linear_x / y / z

sensor_angular_x / y / z

sensor_angular_abs_x / y / z (0 bis 359)

Navigation

map_current_position (x,y)

map_goal_position (x,y)

Inbetriebnahme Schritt 1: Installation der Redis-Datenbank

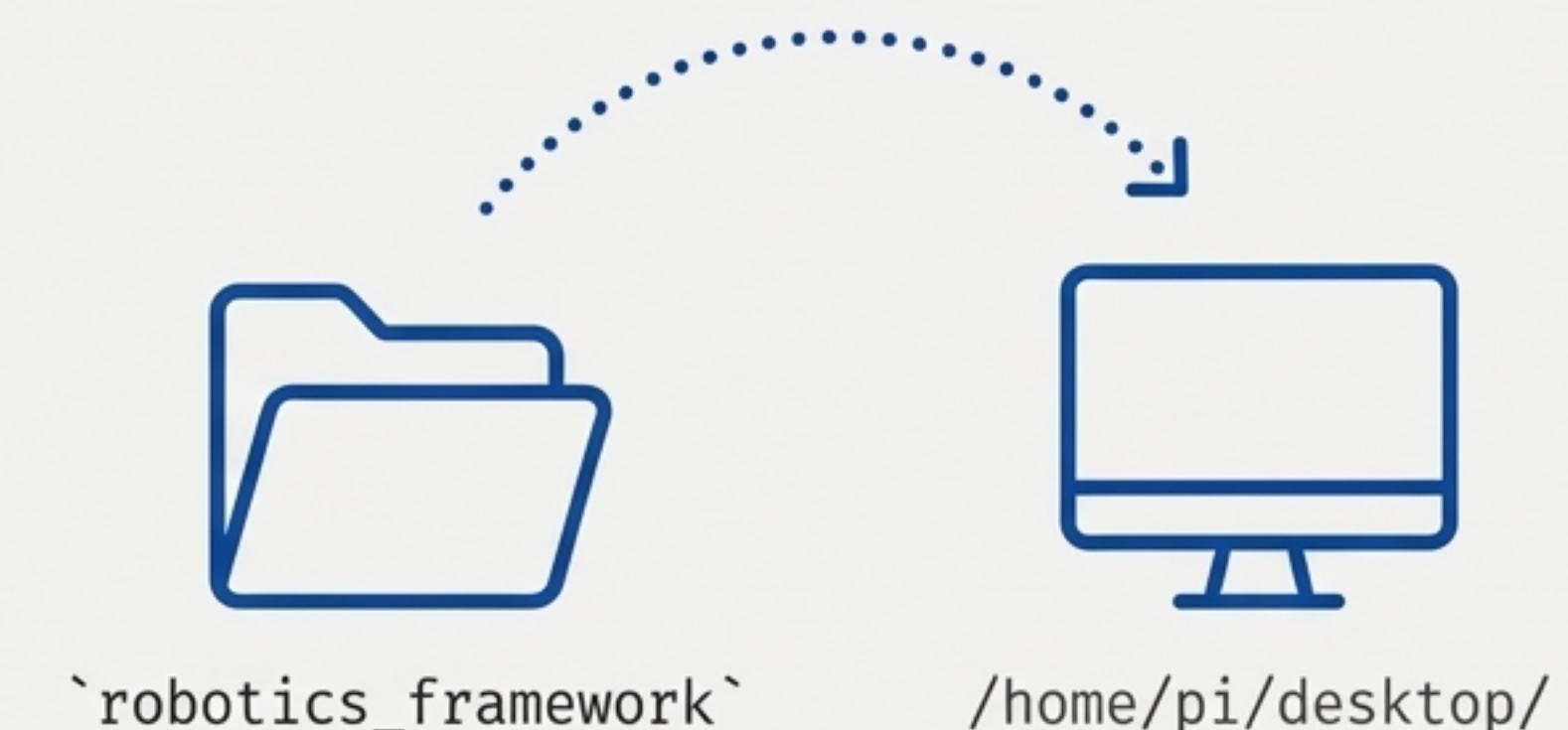
Das Framework benötigt einen laufenden Redis-Server als zentrale Kommunikationsschnittstelle. Die Installation erfolgt über den Paketmanager.

```
# System aktualisieren  
$ sudo apt update  
$ sudo apt upgrade  
  
# Redis-Server installieren  
$ sudo apt install redis-server  
  
# Redis-Dienst starten und für den Systemstart aktivieren  
$ sudo systemctl start redis  
$ sudo systemctl enable redis
```

Inbetriebnahme Schritt 2: Python-Bibliothek und Framework-Dateien

Installieren Sie die Python-Client-Bibliothek für Redis und kopieren Sie die Framework-Dateien an einen geeigneten Ort, z.B. auf den Desktop.

```
>_
# Python-Bibliothek für Redis installieren
$ pip install redis
```



Inbetriebnahme Schritt 3: Automatischer Start der Nodes beim Booten

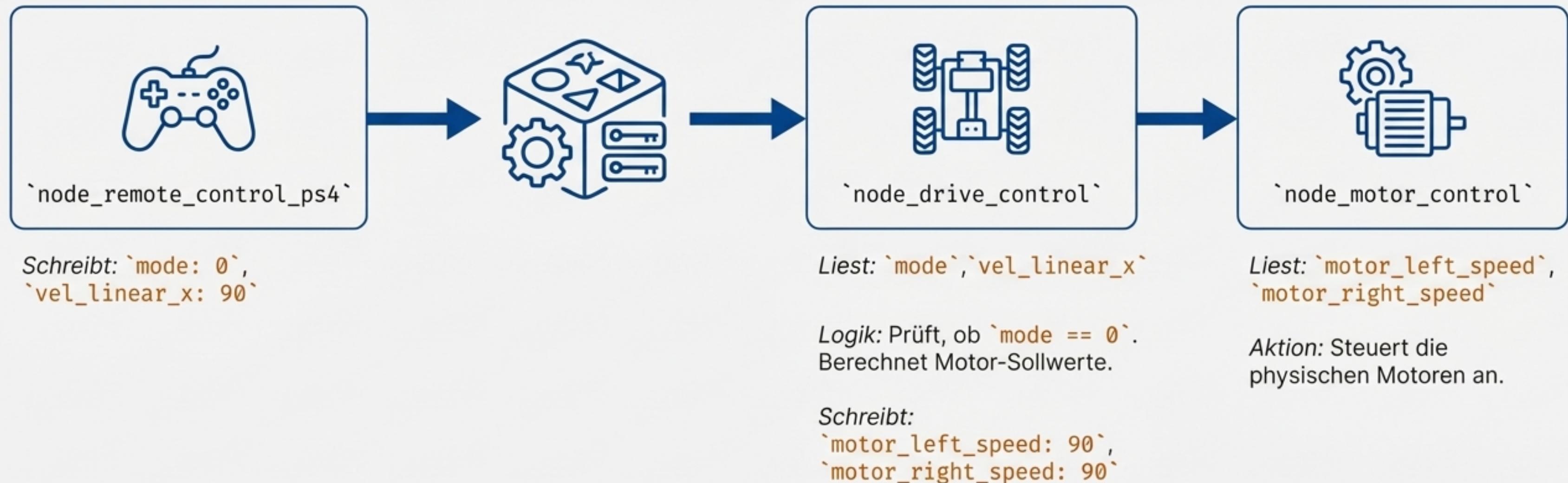
Damit der Roboter nach dem Einschalten sofort einsatzbereit ist, wird ein Autostart-Eintrag für den 'Node Starter' ('starter.py') erstellt.

```
$ sudo nano ~/.config/autostart/nodestarter.desktop

[Desktop Entry]
Type=Application
Name=Node Starter
Exec=python3 /home/pi/Desktop/robotics_framework/starter.py
Terminal=true
```

 Der Exec-Pfad muss an den tatsächlichen Speicherort des Frameworks angepasst werden.

Alles im Zusammenspiel: Ein typischer Anwendungsfall im manuellen Modus



Zusammenfassung und Ausblick

Kernvorteile



Modular: Einfacher Austausch und Erweiterung von Funktionalität.



Robust & Testbar: Unabhängige Entwicklung und Fehlersuche in isolierten Nodes.



Skalierbar: Problemlose Integration neuer Sensoren und Aktoren für komplexe Aufgaben.

Nächste Schritte & Vision

Integration weiterer Sensorarten (LIDAR, Tiefenkameras).

Entwicklung von KI-basierten Navigations- und Entscheidungs-Nodes.

Aufbau einer Dokumentation und Community zur Förderung der Zusammenarbeit.

Erkunden Sie den Code und werden Sie Teil des Projekts



Scannen Sie den Code oder besuchen Sie uns auf GitHub, um den Quellcode einzusehen, Probleme zu melden oder eigene Ideen beizutragen.

``htlwienwest-novak/robotics_framework``



htlwienwest-novak/robotics_framework