Robot operációs rendszerek és fejlesztői ökoszisztémák

ROS_Control, MATLAB ROS toolbox

Kiss Bálint

2025. október 27.

KUKA

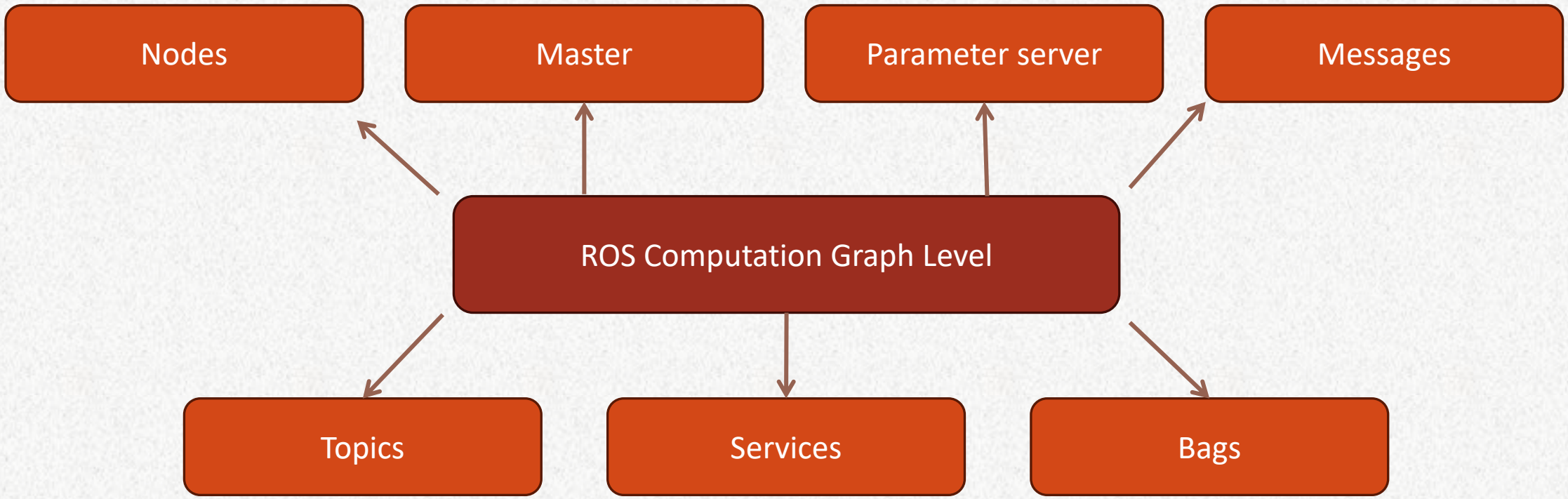
MŰEGYETEM 1782

iit

Tartalom

Visszatekintés

ROS “hálózat”, Computation graph

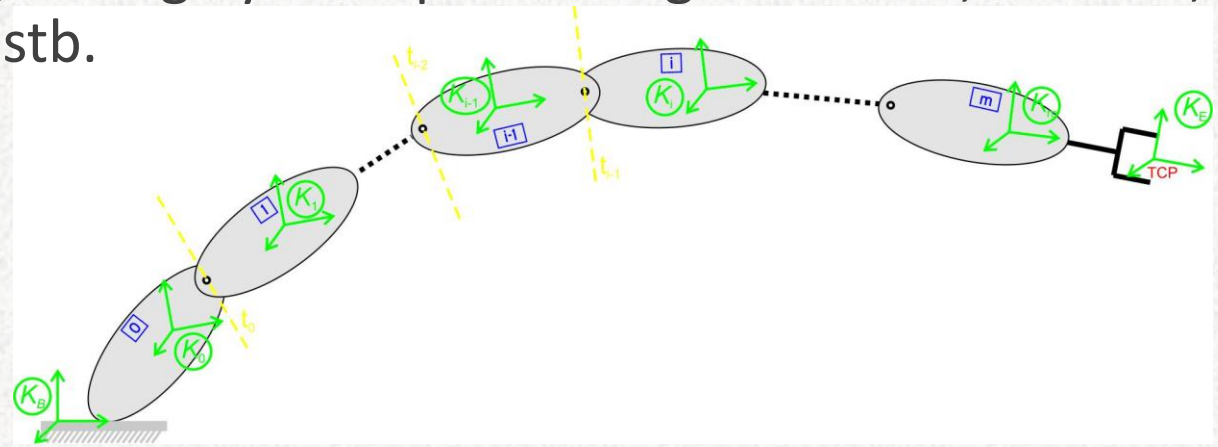


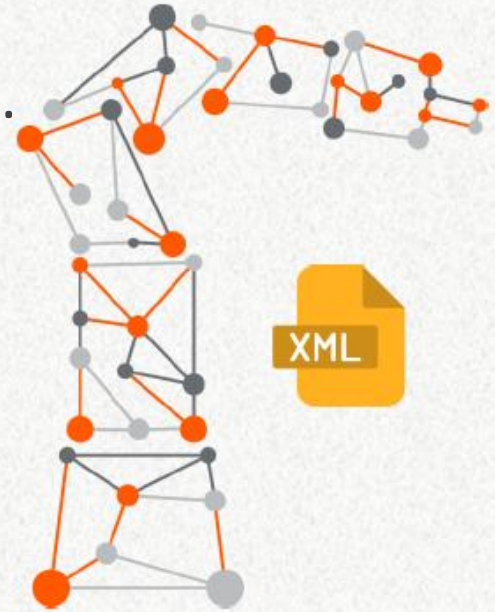
Ebbe a hálózatba kell beilleszteni a szabályozási körök számítási algoritmusait a robotunkhoz / mechanizmusunkhoz.

Robot (mechanizmus) leírása -URDF

Robotkarokhoz és mobilis robotokhoz egyaránt

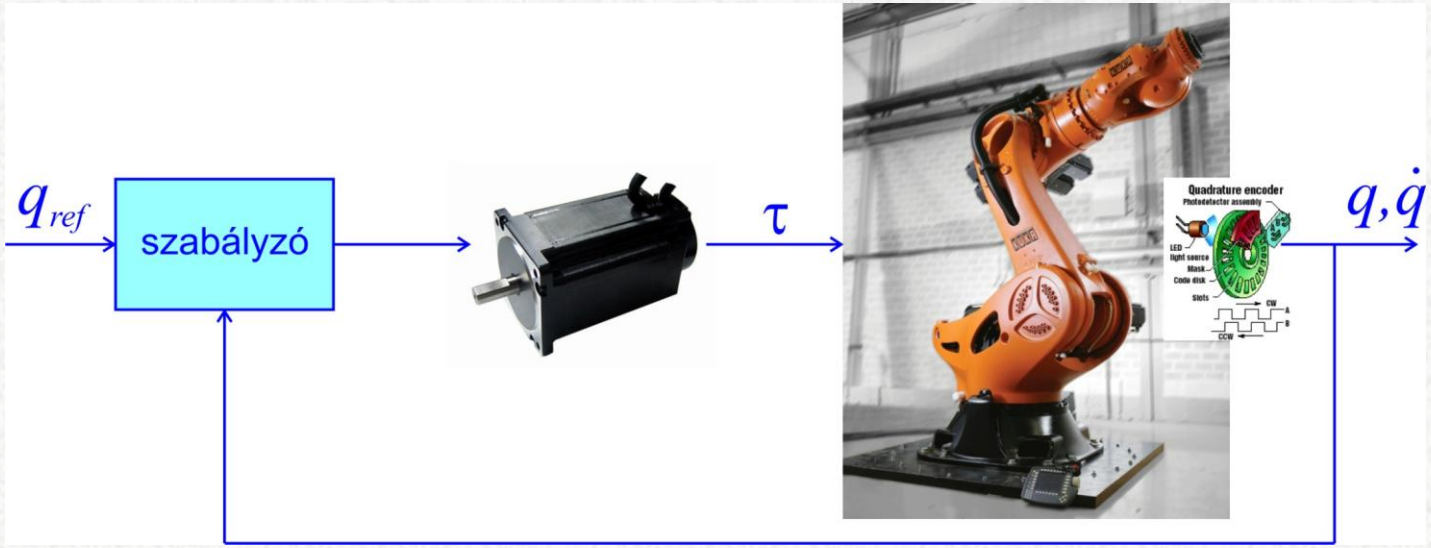
1. Szegmensek (link) – tömegközéppont, kiterjedés, megjelenés, inercia, burkolótest ütközésdetektáláshoz, ütközés dinamikus paraméterei, stb.
2. Csukló (joint) – 1DOF / 2DOF /3DOF relatív mozgás típusa (1DOF: rotációs/transzlációs), összekötött szegmensek, referencia pozíció, relatív mozgás tengelye a kapcsolt szegmenseken, határok, súrlódási együttható, stb.





ROS controls

Egy kis szabtech: az irányítandó szakasz a robot



1. Hagyományos irányítási hurok érzékelőkkel, algoritmussal (pl. PID szabályozó), beavatkozókkal.
2. A ros_control csomagok gyűjteménye ilyen hurkok algoritmusainak implementálására **is** jók.
3. Az irányítási hurkok hierarchikusak: nem mindegyiket kell a ROS-ban zárni.

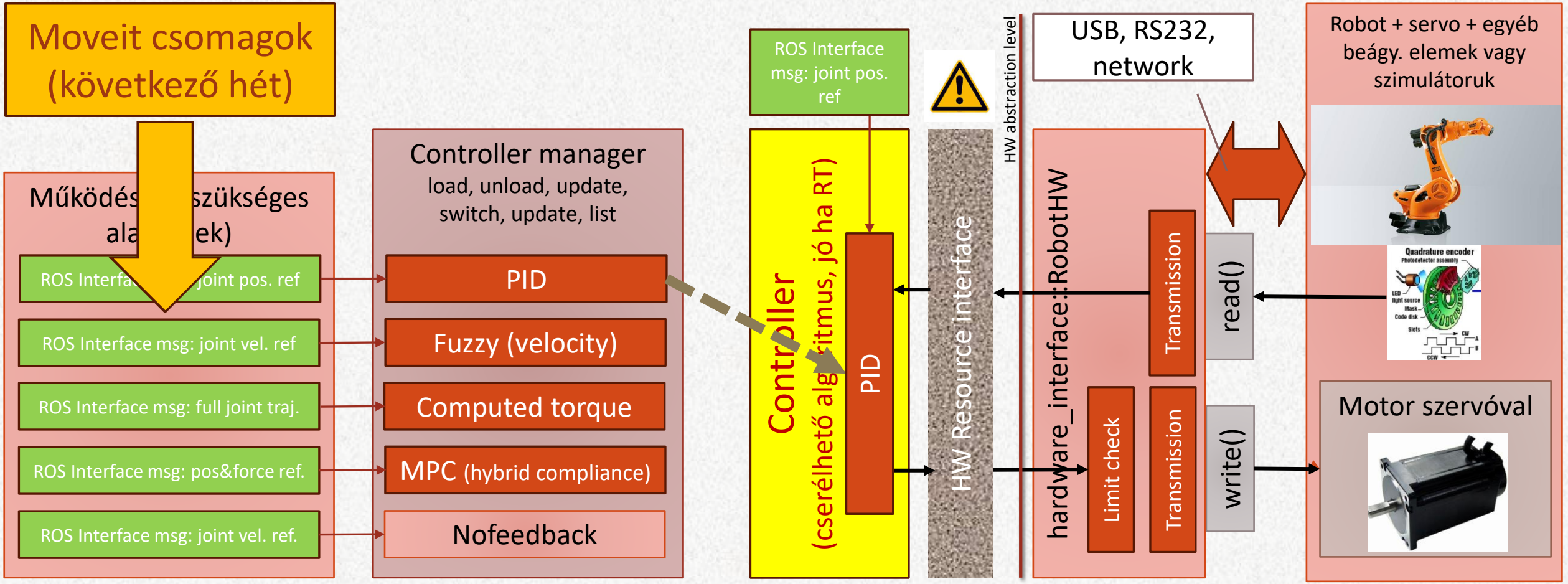
Célkitűzések

1. Robot hardverhez közeli rétegek leválasztása (minél közelebb a hardverhez).
2. Azonos architektúra szimulációhoz és valódi robotok irányításához.
3. Újrahasznosíthatóság: ugyan az a szabályozó több robothoz és több pályatervezési megoldáshoz is használható legyen.
4. Újrahasznosíthatóság: a szabályozók automatikusan hozzáférhessenek az aktuális robot paramétereihez (futási időben vagy fordítási időben)
5. Valósidejűség támogatása (RT OS-re fordítható kód).

Előzmények

Pr2_controller_manager – egy humanoid robot irányításához készült csomagok, ennek általánosított verziója a ros controls

ROS control és „barátai” (ROS-controls) Koncepció



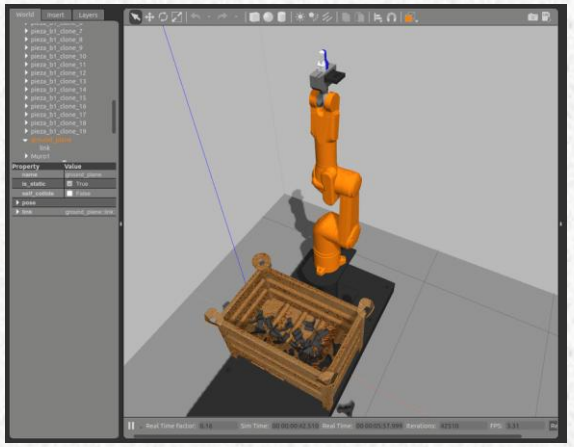
A ROS koncepció nem teljesen egyezik az irányítástechnika “control” koncepciójával

Control alatt azt értjük, hogy a robot egyes jellemzőit változtatva **befolyásoljuk**, beállítjuk a robot mozgását, ehhez nem kell zárni szabályozási hurkot.

Csuklók (link) pozíciója - position
Csuklók sebessége - velocity
Csuklónyomatékok - effort
Mobilis robot sebességállapota (twist)
Stb.

Egyszerű alkalmazások:

1. URDF-fel adott gemetriájú robotkar rotációs csuklóit egy grafikus felületről beállítjuk (nincs dinamika, “rajzolunk”)
2. URDF-fel adott mobilis robot kerekeinek szögsebességét beállítjuk sebességállapot alapján: egyszerű dinamika, twist számítás.



Szimulált robot (Gazebo)

Irányító rendszer + safety

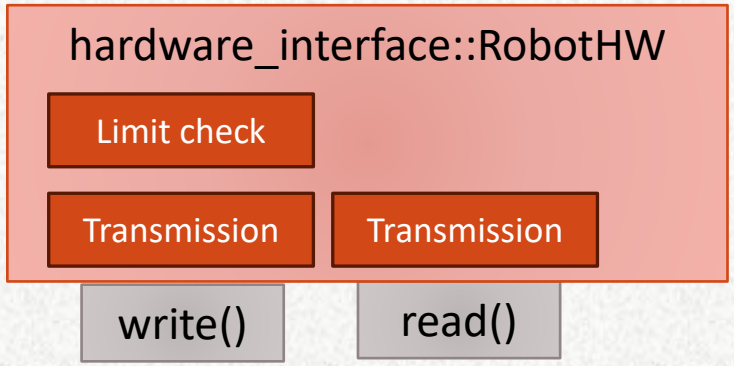
valódi robot

ROS-controls repo fő elemei

1. **ros_control** – az általános keretrendszer
2. ros_controllers – a ros-control keretrendszerbe illeszthető “szabályozók” (inkább elemek)
3. control_toolbox – eszközök controllerek létrehozására
4. Realtime tools – eszközök „kemény” valós idejű (hardRT) implementálásához
5. Control_msgs – üzenetek és műveletek (akciók) robotok irányításához
6. Gazebo_ros_control (ros-simulation/gazebo_rps_pkgs – nem a ROS-controls része)

A ROS2 elnevezések hasonlók

Robot hardware abstraction



Robot hardver leírás:

1. Csuklók,
2. Érzékelők (pozíció, sebesség, nyomaték)
3. “Beavatkozók” (előírt nyomaték, sebesség, pozíció)

Kinyerhető URDF-ből.

Limit check — Opcionális: határérték ellenőrzés

Transmission — Opcionális: áttételek, robot specifikus kinematika figyelembe vétele

write() — Implementálandó a robot kommunikációs lehetőségei szerint (ROS2-ben kész függvények)

read() — Implementálandó a robot kommunikációs lehetőségei szerint (ROS2-ben kést függvények

Q: Hogyan lehet beavatkozó jel a pozíció?
A1: A pozíció szabályozást nem a ROS-on belül valósítjuk meg, azt a robot melletti beágyazott HW végzi
A2: szimulált robotunk van dinamika nélkül, csak beállítjuk a pozíciót a következő szimulálási lépésben.

hardware_interface (docs)

HW Resource interface

1. Csak olvasható:
   - csuklóváltozók állapota (pozíció/sebesség),
   - IMU (inerciális szenzor)
   - erő-nyomaték érzékelő
2. Írható/olvasható csukló beavatkozások
   - Pozíció
   - Sebesség
   - Nyomaték
3. Nem engedi, hogy egy csuklón több szabályozó végezzen beavatkozást
4. Bővül

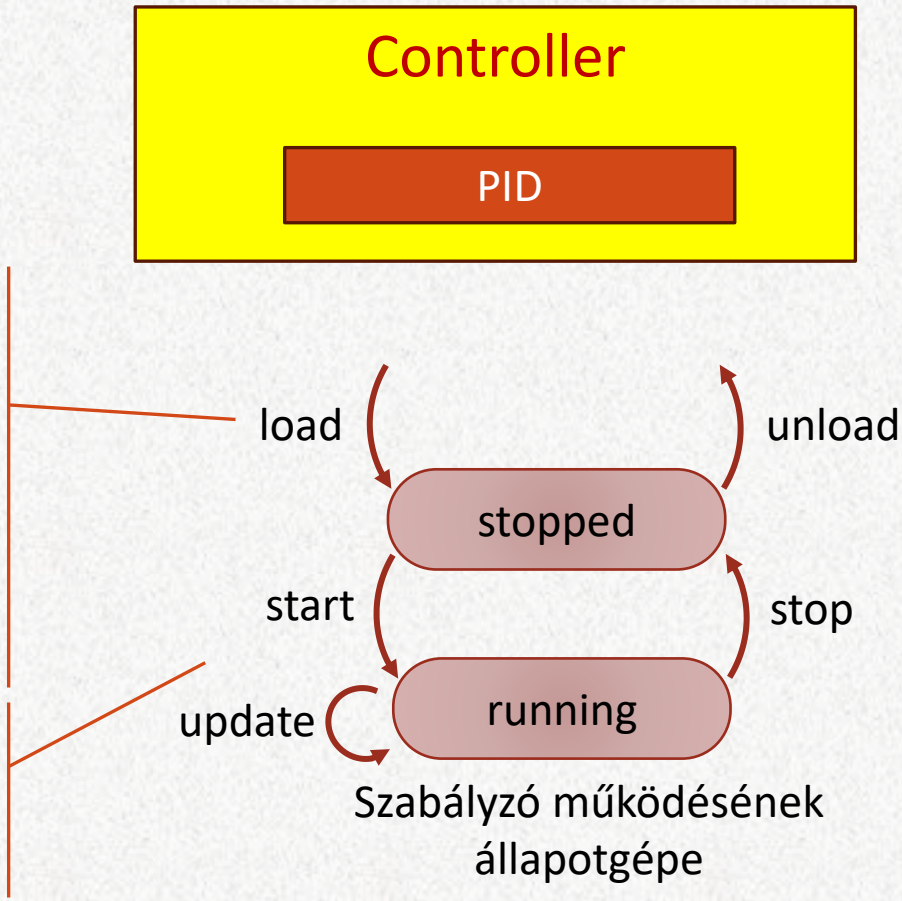
Szabályozók és működésük állapotgépe

1. Több Controller is működhet párhuzamosan (de egy tengelyt/csuklót, azaz robot erőforrást csak egy irányíthat)
2. RT és nem RT műveletek elválnak

load
- Betöltés és inicializálás
- Ellenörzés (pl. van-e olyan érzékelő és beavatkozó a HW interfészben, amivel a szabályzó működni képes
- ROS interfészek felállítása (pl. pályatervező felé)

unload
- törlés

start: első iterálás előtt végrehajtandó alaphelyzetbe állítás
stop: utolsó iterálás után végrehajtandó
update: iterálás (ide jön pl. a PID kódja)



Szabályzó működésének állapotgépe

ros_controllers repo

1. Érzékelők kiolvasása (reporting)
   - **Joint_state_controller** – csuklók állapotát tartalmazó üzeneteket küld a sensor_msgs/JointState topicra
   - **imu_sensor_controller** – inerciális érzékelő adatokat tartalmazó üzeneteket küld a sensor_msgs/imu topicra
   - **force_torque_sensor_controller** – erő/nyomatékmérő cella adatait tartalmazó üzeneteket küld a sensor_msgs/wrench topicra
2. Csuklónkénti beavatkozás
   - **position_controllers** – csukló pozícióját állítja (üzenetküldéssel)
   - **velocity_controllers** – csukló sebességét állítja (üzenetküldéssel)
   - **effort_controllers** – csukló nyomatékát állítja (üzenetküldéssel)
3. **Joint_trajectory_controller** – több csuklós együttes pályainterpolációja (MoveIt! Kompatibilis, mert megfelelő action interface-t implementál)

Bár controller-ek, nem avatkoznak be

Bavatkoznak

ros_controllers repo

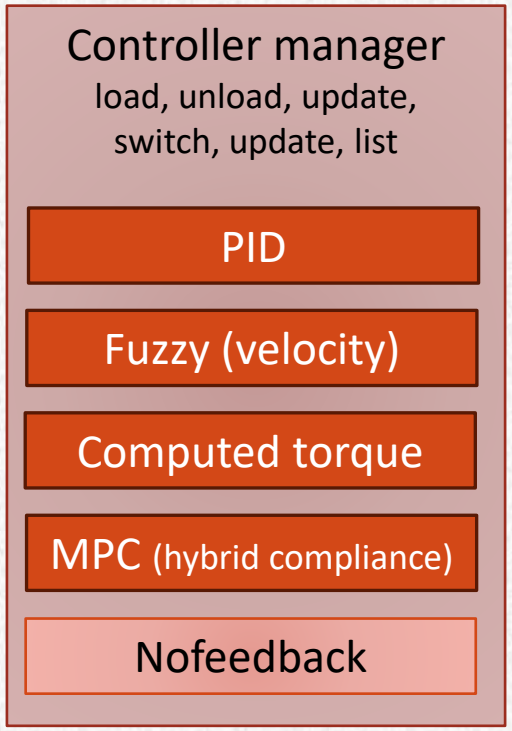
1. **diff_drive_controller**
   - Differenciális meghajtású robotokhoz
   - Odometria számítását elvégzi a szernzorok alapján, üzenetekben elküldi a nav_msgs/Odometry topicra
   - Sebesség és pozíció visszacsatolással is működik
   - A robothoz rögzített keretben adott sebesség parancsokat fogad a geometry_msgs/Twist topic-on, azt keréksebességbe átszámolja
2. **Ackermann_steering_controller**
   - Járműszerű moblis robotokhoz
3. **Gripper_action_controller**
   - 1 DOF megfogóhoz
   - MoveIt! komaptbilis

Bavatkoznak

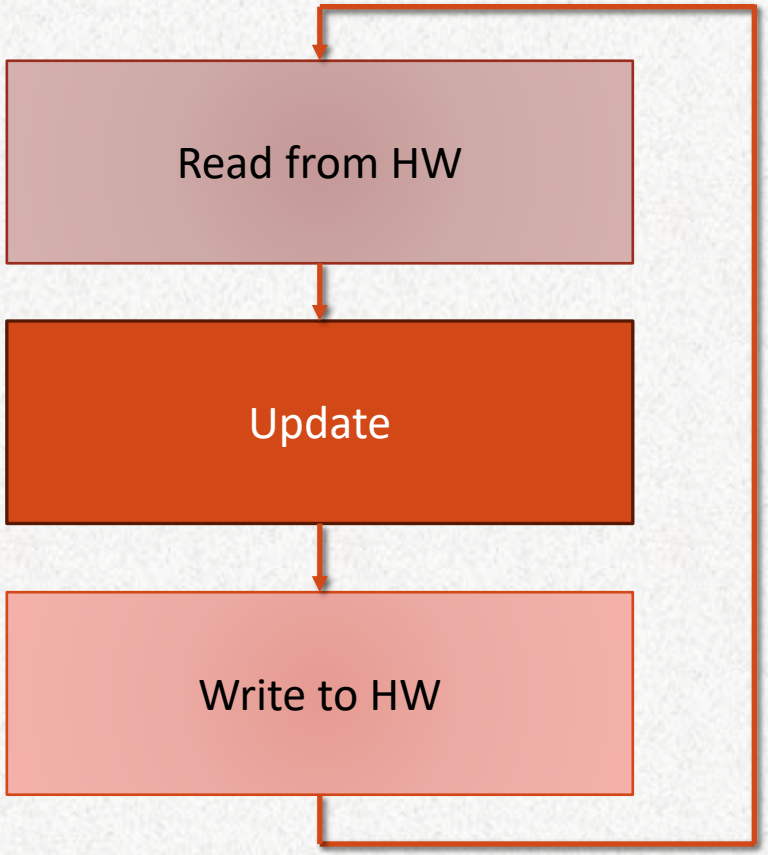
Controller_manager

Feladatai:

1. Robot erőforrások nyilvántartása
2. Ellenőrzi a controllerek és a robot erőforrások megfelelőségét
3. Ellenőrzi, hogy nincs-e erőforrás konfliktus (pl. egy tengelyt több controller akarnak vezérelni)
4. Controller állapotgép működtetése
5. Controller-ek update metódusainak periodikus hívása (adott periódusban sorban)
6. Switch_controller – szabályzó csere garantáltan egy ciklusban
7. Lekérdezhető a futó szabályozók és a rendelkezésre álló szabályozók halmaza, stb.



```
#include <ros/ros.h>
#include <myrobot/myrobot.h>
#include <control_manager/control_manager.h>

int main(int argc, char **argv) {
        // setup
        ros::init(argc, argv, 'my robot');

        MyRobot::MyRobot robot;
        controller_manager:controller_Manager cm(&robot);

        ros::AsyncSpinner spiner(0); // non RT thread
        spinner.start();

        // Control loop
        ros::Time prev_time = ros::Time::now();
        ros::Rate rate(10.0);

        while (ros::ok())     {
                const ros::Time      time = ros::Time::now();
                const ros::Duration period = time – prev_time;

                robot.read();
                cm.update(time, period);
                robot.write();

                rate.sleep(); // sleep for cycle leftorver
        }
}
```

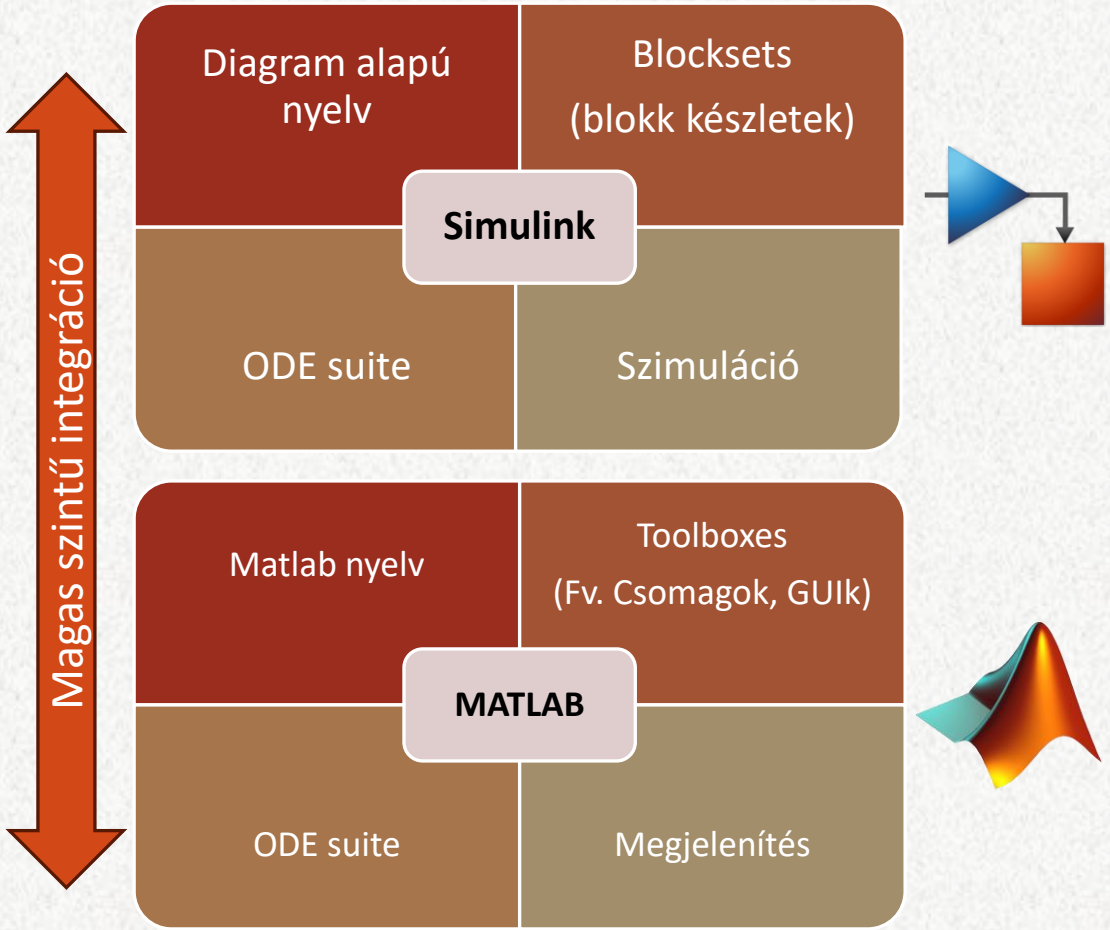


Do not trust a controller

- Használjunk korlátozásokat a jelekre: joint_limits_interface
- Adjuk meg őket az URDF-ben
- Ellenőrizzük őket
  - Update implementációban
  - HW interface-ben

Matlab ROS és Simulink ROS

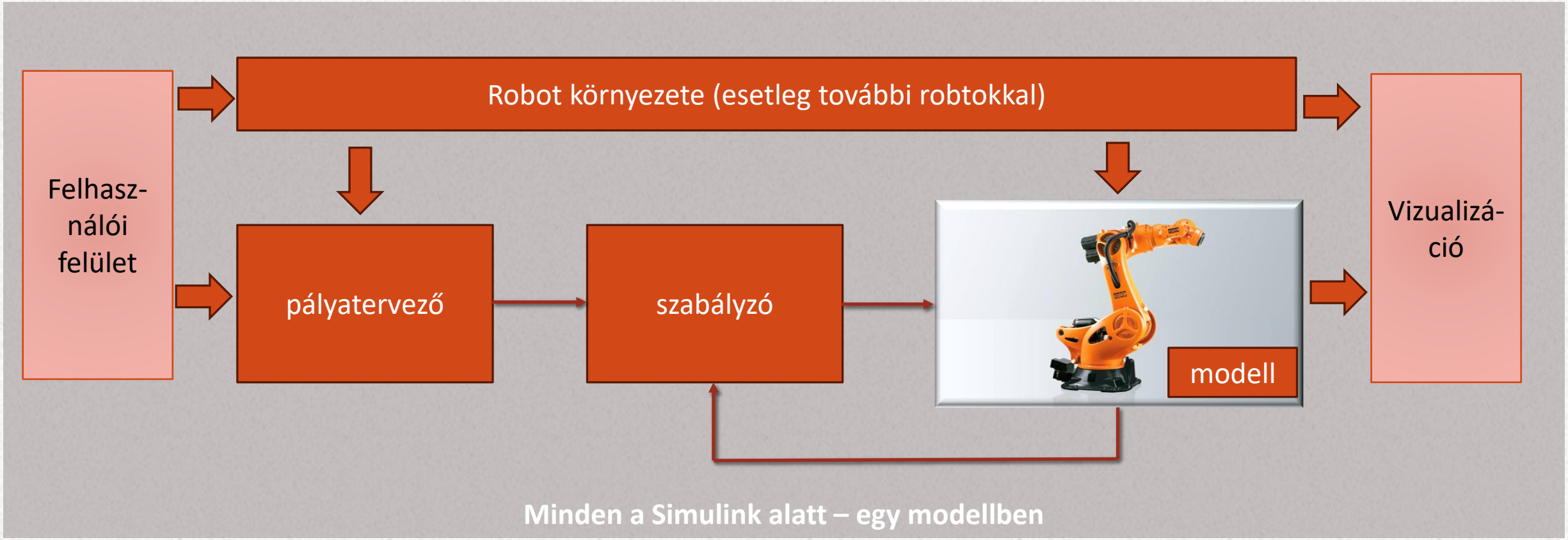
MATLAB – Simulink ökoszisztéma



1. Matlab “felett” fut, annak ODE megoldóját használja.
2. Dinamikus modellek felépítése (robot, környezet)
3. Összetett algoritmusok gyors, magas szintű, diagram alapú fejlesztés és vizsgálata (SIL)
4. Integrálás a Matlab környezettel: analízis, hangolás, tesztelés alacsony szintű programozás nélkül

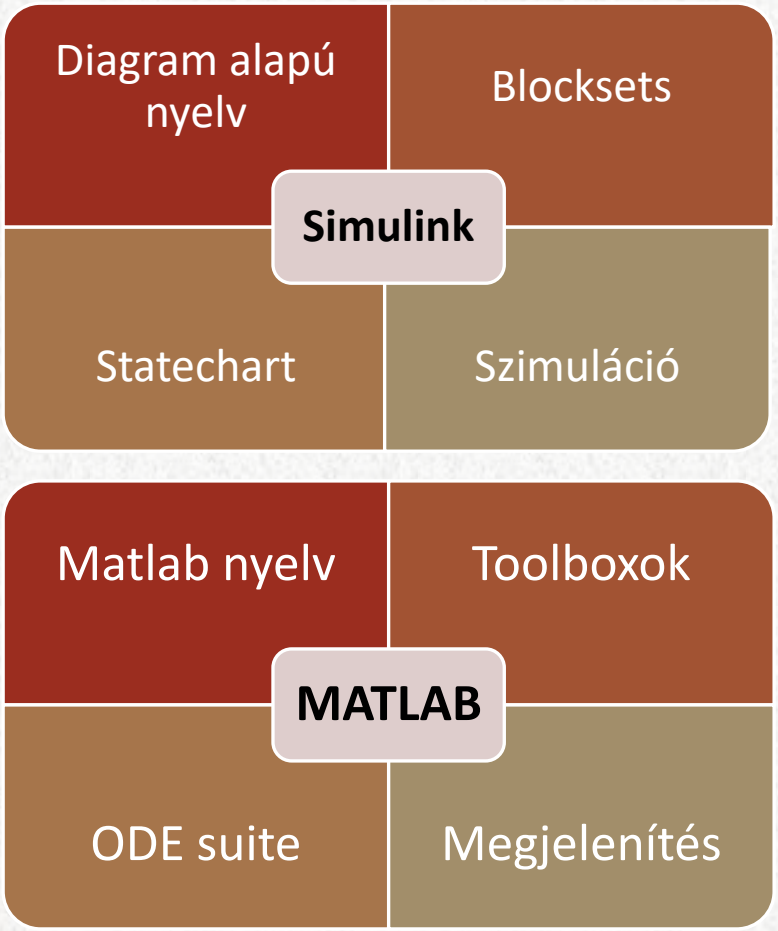
1. Stabil numerikus algoritmusok
2. Egyszerű, de objektumorientált „Matlab nyelv”
3. GUI-kon keresztül is elérhető Toolbox-ok autonóm funkciók fejlesztéséhez (mélytanulás, identifikáció és szabályozás, képfeldolgozás, stb.)
4. Adatelemzés és vizualizáció

A SIL megközelítés Simulinkben



SIL – Software-in-the-loop

Matlab-Simulink és kódgenerálás



Kódgenerálás (Simulink Coder)

Run-time interface

Kódgenerálás (Matlab Coder)

C/C++ kód
PLC kód
GPU kód
stb.
Lehet részleges (csak egyes függvények - gyorsítás)

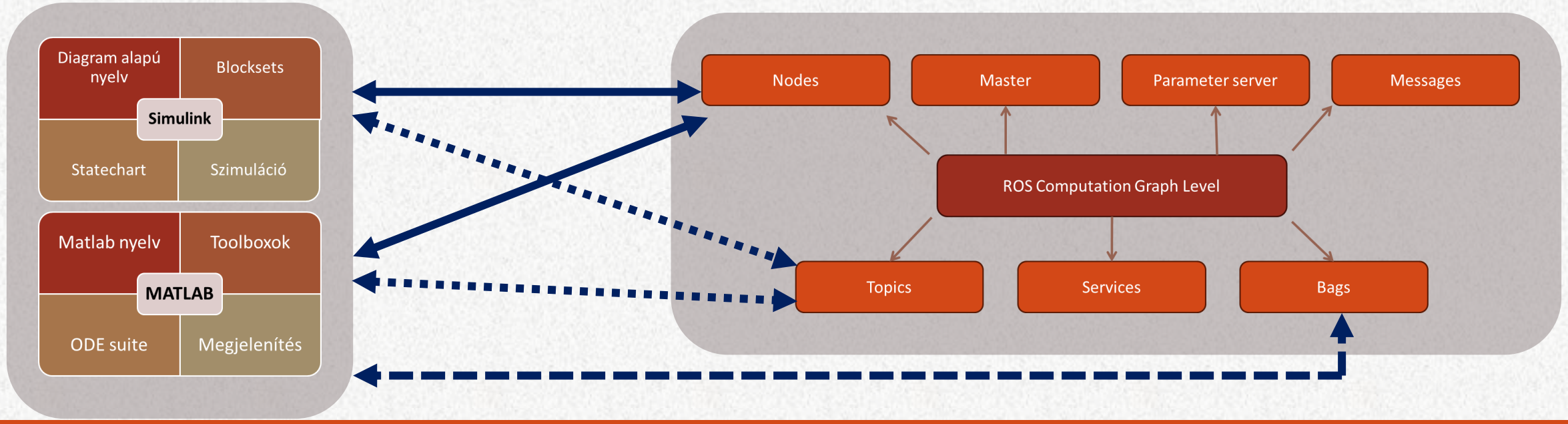
Matlab Executabe (MEX)
Standalone APP
C/C++ kód
stb.
Lehet részleges (gyorsítás)

Egyéb Coderek beágyazott HW-re: Embedded Coder, PLC Coder

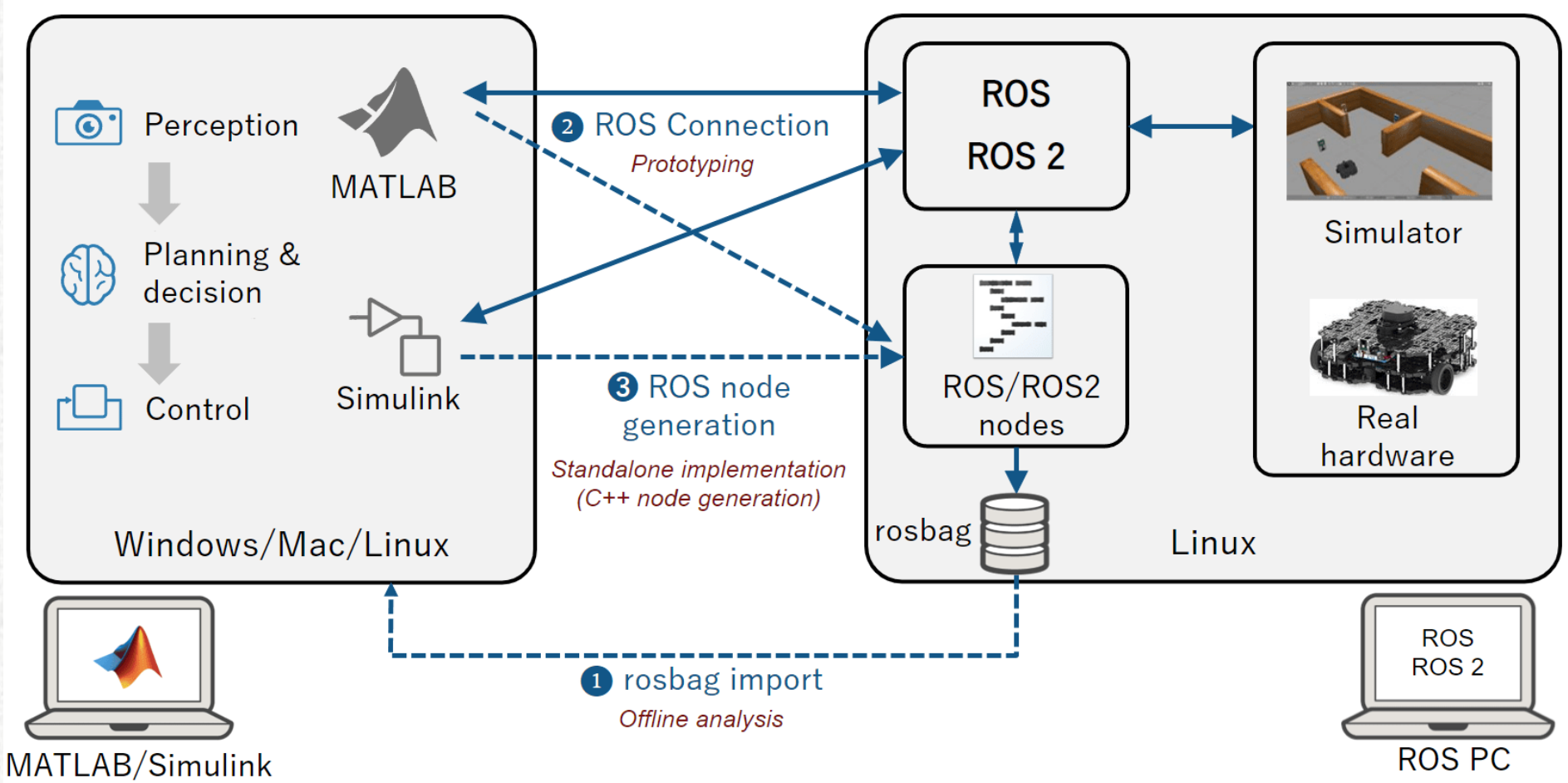
Mi lenne, ha ROS/ROS2 “kompatibilis” kódot is tudnánk generálni?

ROS Toolbox – kódgenerálás és egyebek

1. ROS által rögzített adatok (ROSbag) importálása, vizsgálata, lejátszása, modosítása, létrehozása (off-line) - analysis
2. Futási idejű kapcsolat a ROS rendszerrel - prototyping
3. Kódgenerálás a ROS rendszer felé (C++ node kódok) – prototyping & deployment



ROS Toolbox doksijának ábrájával



ROS Toolbox használati esetek

1. ROSbag és ROS2bag adat importálás és lejátszás:
   - Üzenetek (tipikusan csuklók és erő/nyomaték, távolság szenzorok), kameraképek, Lidar pontfelhők, foglaltsági térképek (mobilis robotoknál)
   - Importálás: átalakítás Matlab struktúrává (pl.: timeseries üzenetekből)
   - A Matlab toolboxai és azok segítségével készített algoritmusok, programok az importált adatokon dolgozhatnak, (képfeldolgozás, mélytanulás) és eredményeiket visszaküldhetik a ROS rendszerbe (üzenetekkel)
   - Importált adatok forrásként lejátszatók Simulink modellben
2. Run-time kapcsolat ROS rendszerrel – csatlakozás, üzenet küldése és fogadása
   - Matlab parancsokon keresztül
   - Simulink nyelőkön és forrásokon keresztül
   - Felhasználás: együttes vagy elosztott szimuláció (co-simulation), pl. Simulink + Gazebo

ROS Toolbox használati esetek

1. Coder: Matlab függvényből vagy Simulink modellből (annak egy blokkjából) képes kódot előállítani (pl. egy új controller-t a ros_controllers repo-ba.
2. A kód előállatása konfigurálható coder.config objektum
3. Beállítandó “célkörnyezet”: ROS/ROS2
4. Simulink modellből:
   - A generált Node a Simulink host gépén fut
   - A generált Node a távoli ROS eszközön (azok valamelyikén) fut.