ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

ABB RobotStudio Gyakorlat

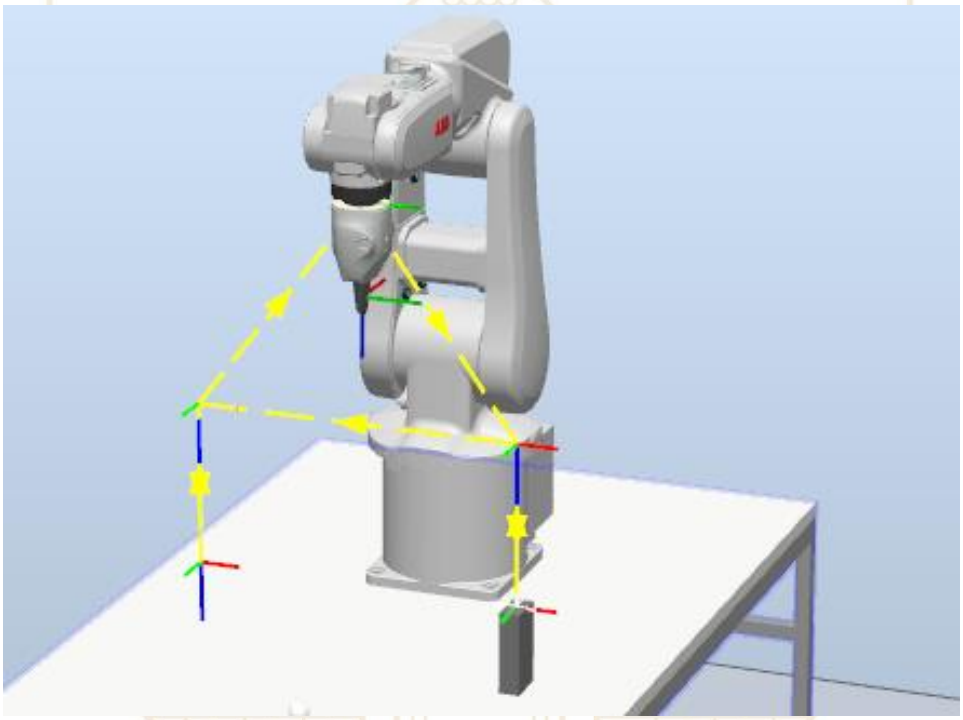
III. Mérési Feladat

„Pick and Place" feladatok implementálása RobotStudioban

Mérés Célja:

A mérés során a hallgatónak egy egyszerű „Pick and Place" robotprogramot kell elkészítenie. A manipulátor az 1. ábrán látható munkadarabot fogja áthelyezni az asztal egyik feléről a másikra.

A „Pick and Place" típusú feladatok az egyik leggyakrabban automatizált műveltek a gyártási folyamtokban. A mérés célja, hogy a hallgató megismerkedjen az ilyen típusú feladatok elvégzéshez szükséges robotprogram felépítésével és így a későbbiekben képes legyen önállóan hasonló programok készítésére. A bemutatott példán keresztül a hallgató elsajátíthatja, azon alapokat amelyeket a későbbiekben felhasználva akár komplexebb programokat is fejleszthet (pl.: szerszámgép kiszolgálás, palettázási feladatok megvalósítása).



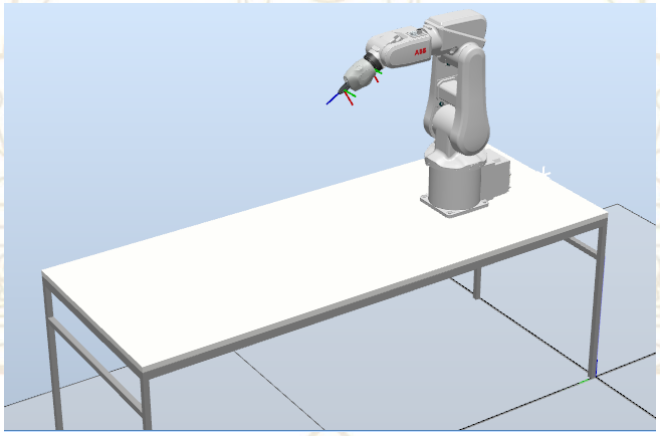
1. ábra Bejárni kívánt pálya

A mérésről a hallgatónak jegyzőkönyvet kell késztenie, amelyben az egyes részfeladatok végén található kérdéseket kell megválaszolni.

1. feladat – Robotcella felépítése

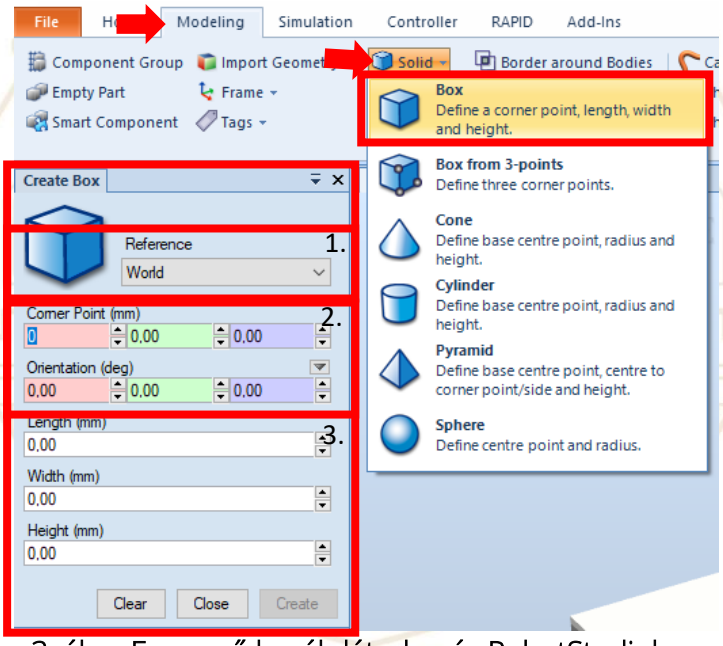
A szimulációt egy új virtuális cella segítségével fogjuk elkészíteni. Ehhez építse meg a 2. ábrán látható elrendezést, melyhez a következő elemeket használja:

- Robot: ABB IRB120_3_0.58_G_01
- Megfogó: ABB Smart Gripper-Servo, Fingers (ABB Library – Equipment)
- Asztal: Table (Add-Ins)



2. ábra Cella felépítése

A munkadarabunk ezesetben egy egyszerű hasáb lesz, amelyet nem szükséges külső modellből beimportálni, mivel ilyen egyszerű geometriákat a RobotStudio segítségével is elkészíthetünk. Ehhez válasszuk felül a **Modeling** fület, majd nyissuk le a **Solid** menüpontot és válasszuk a **Box** opciót. A létrehozni kívánt hasáb paramétereit a bal felső sarokban megjelenő ablakban adhatjuk meg.

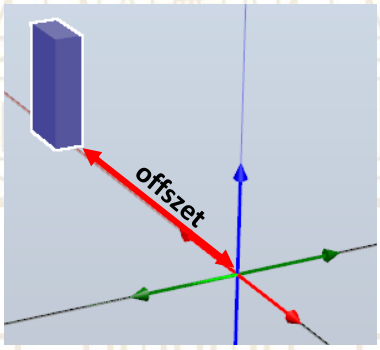


3. ábra Egyszerű hasáb létrehozás RobotStudioban

A hasáb létrehozása során a következő paramétereket kell megadnunk:

1. Referencia koordináta rendszer
2. Pozíció és orientáció
3. Hasáb méretei

Hozzon létre egy 50x25x100mm-es (Sz x H x M) hasábot! Fontos megjegyezni, hogy amennyiben itt adjuk meg a hasáb pozícióját a geometria megfelelő helyre kerül, azonban a RobotStudio az új alkatrészek alap koordináta rendszerét, mindig a világ koordináta rendszer origójába hozza létre, ez azt fogja eredményezni, hogy a geometria és az alap koordináta rendszer között egy offszet fog megjelenni, ami megnehezítheti a későbbi munkánkat (4. ábra). Ezért javasolt a hasáb pozíciójának és orientációjának későbbi beállítása.



4. ábra Hasáb és az alapkoordináta rendszerének elhelyezkedése

Helyezze el a munkadarabot az asztal felületén úgy, hogy az a robot munkaterébe essen!

Kérdések:

1. Hogyan hozná létre az alábbi alkatrészt a RobotStudio segítségével?

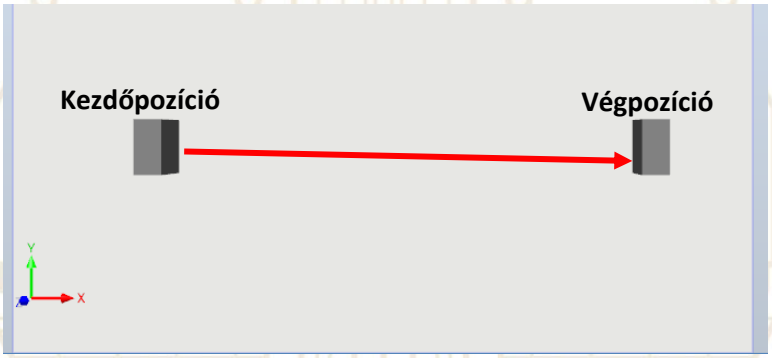
Írja le a lépések menetét! Segítségül tanulmányozza a felső szalagmenüt a Modeling fül kiválasztása esetén.

2. feladat – Robotmozgások programozása

A robotmozgások konfigurációja előtt, először a pályapontok és a **workobejct**ek felvételét kell elvégeznünk. Jelen feladat esetében azonban a **wobj0** alapértelmezett **workobject**et is használhatjuk. Ez esetben ez azért lehetséges mivel egy rendkívül egyszerű programot készítünk, viszonyleg kevés pályaponttal. Továbbá nem tudunk olyan **workobject**et kijelölni, amely segítséget nyújtana számunkra a programozási feladat megoldása során, mivel munkadarabunkhoz nem tudjuk hozzárendelni a koordináta rendszerünket, hiszen az a szimuláció során mozogni fog. Egy valós cella esetén a munkadarabot valamilyen adagoló vagy pozícionáló berendezésből vennénk fel és így ahhoz rendelhetnénk egy **workobject** koordináta rendszert. Most ettől eltekintünk a feladat egyszerűsítése érdekében.

Tehát így már elkezdhetjük a pályapontok felvételét, majd a robotmozgások konfigurációját. Az előző mérési feladat során ismertetett szempontok alapján vegye fel a szükséges pályapontokat (**Target, Joint Target**) és tervezze meg a robot mozgáspályáját! Továbbá a pályatervezés során a következő szempontokat vegye még figyelembe:

- A robot **home** pozícióból kezdje meg a munkát és a feladat befejeztével oda álljon vissza
- A robot vegye fel az asztal egyik széléről a munkadarabot és helyezze át az asztal másik szélére, az eredeti pozícióval egyvonalba (5. ábra)
- Figyeljen a szerszám és a munkadarab megfelelő kiemelésére a mozgatás után és a munkadarab helyes megközelítésére!
- A közelítési adatok (**zonedata**) megfelelő beállításával optimalizálja a mozgásokat.
- Ügyeljen a stop pozíciók és a **fly-by** pontok megfelelő konfigurációjára! Ne feledje **fly-by** pontok esetében a robot nem áll pontosan pozícióba csak megközelíti azt!
- Jelenleg csak a mozgáspályát kell elkészítenie!



5. ábra Hasáb mozgatása

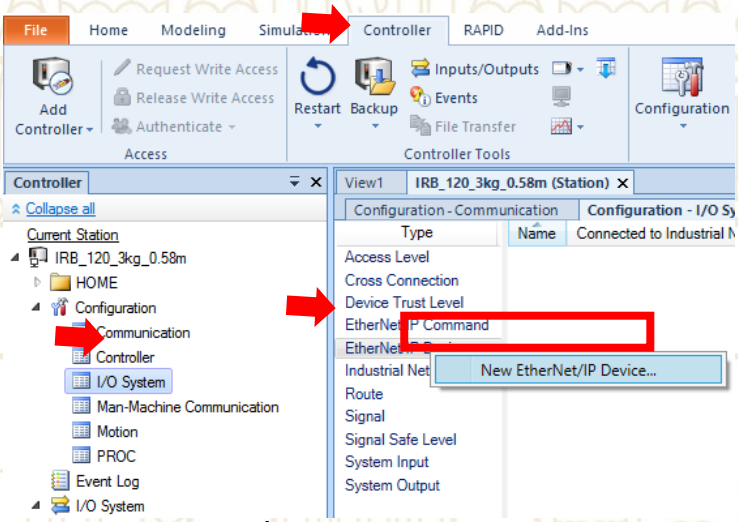
Kérdések:

2. Mutassa be milyen Targeteket vett fel a mozgáspálya megvalósításához!
3. Mutassa be, hogyan konfigurálta az egyes mozgásokat. Indokolja miért az adott mozgástípust és megközelítési beállításokat használta!

3. feladat – Ki- és bementek vezérlése

A robotunk megfogójának vezérléséhez digitális kimeneteket fogunk használni. Az IRC5 vezérlő esetében például DSQC 1030 I/O egységet használhatunk hasonló vezérlési célokra. Ez az eszköz **Ethernet/IP** protokollt használva kommunikál a robotvezérlőnkkel, így egyéb hardver vagy szoftver opció telepítése nélkül használható a vezérlőnkkel.

A digitális kimeneteink beállításához, először az I/O eszköz konfigurációját kell elvégeznünk. Ehhez felül válasszuk a **Controller** fület, majd a baloldali menüben válasszuk a **IRB_120_3kg_0.58m>Configuration>I/O System** menüpontot és végül középen kattintsunk jobb egérgombbal a **Ethernet/IP Device** menüpontra és válaszuk a **New Ethernet/IP Device** opciót. Itt tudjuk hozzáadni a rendszerünkhöz a DSQC 1030 I/O modult.

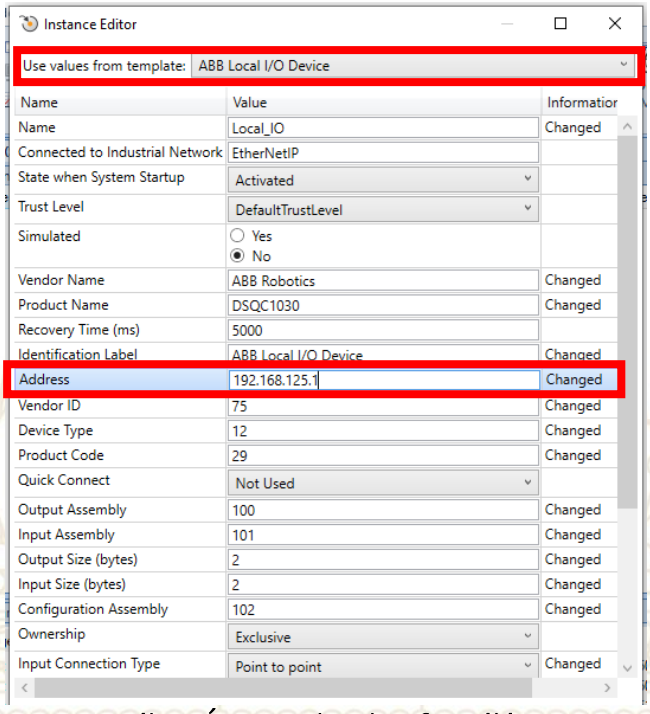


6. ábra Új IO eszköz hozzáadása

Jól látható, hogy a felugró ablakban rengeteg paraméter megadása szükséges egy új I/O eszköz konfigurációjához. Azonban mivel mi most nem valós hardverrel dolgozunk és az eddig bemutatott lépések is elhagyhatók szimuláció esetén, ezen paraméterek részleteiben nem kerülnek ismertetésre. Így egyszerűen legfelül a legördülő menüből válasszuk ki a **ABB Local I/O Device** sablont. A paraméterek részletes megismeréséhez tekintse meg az **ABB Technical Reference Manual – System Parameters,** illetve az **Application manual Local I/O** kézikönyveket. Az egyetlen módosítás, amit el kell végeznünk, hogy az **Address** paramétert a következőre módosítjuk: 192.168.125.1 (mivel nincs valós eszköz csatlakoztatva csak egy véletlenszerű IP címet adunk meg).

Figyelem!

Amennyiben nem valós hardverrel dolgozunk, az eddigiekben bemutatott lépések elvégzése nem kötelező. Ez esetben úgynevezett szimulált I/O jelet hozunk létre, a jel nem lesz egyik I/O eszközhöz sem hozzárendelve.



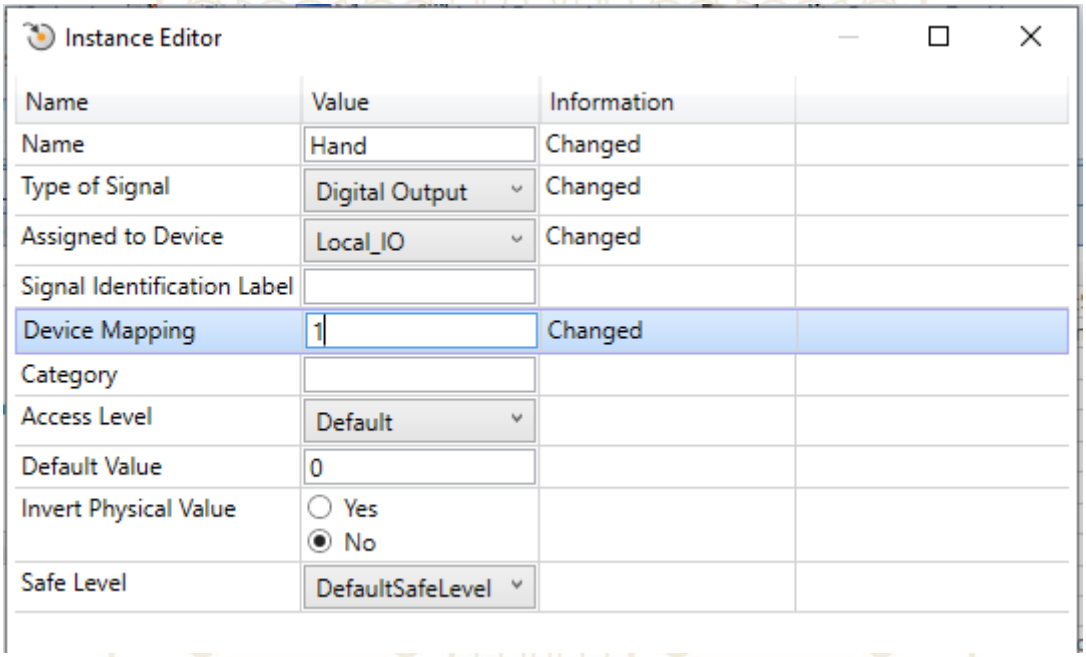
7. ábra Új IO eszköz konfigurálása

Ha ezzel megvagyunk elkezdhetjük az I/O jelünk konfigurációját. Ehhez a középső ablakban, ahol a kiválasztottuk a **Ethernet/IP Device** opciót is, kattintsunk jobb egérgombbal a **Singal** menüpontra és válasszuk ki a **New Signal** menüpontot. A jelünk konfigurálásánál a következő paramétereket kell megadnunk:

- **Name:** A jel neve, a későbbiekben majd ezzel névvel hivatkozhatunk a jelre például a Rapid kódunkban.
- **Type of Signal:** Jel típusa lehet Analóg ki- vagy bemenet, Digitális ki- vagy bemenet, illetve Csoportos ki- vagy bemenet. A csoportos ki- vagy bemenet több I/O-ból áll, amelyeket együttesen kiolvasva kaphatjuk meg a szükséges információt.
- **Assigned to Device:** Itt azt adhatjuk meg, hogy mely I/O eszközhöz szeretnénk hozzárendelni a jelünket (pl.: DSQC 1030 vagy DSQC 652 I/O modul). Szimulált jelek esetében nincs szükség a megadásra.
- **Signal Idetification Label:** Opcionálisan megadható paraméter, itt például megadhatjuk annak a jelvezetéknek a megnevezését, amely fizikálisan az IO-hoz van csatlakoztatva.
- **Device Mapping:** Itt adhatjuk meg mely fizikális IO-t akarjuk a jelünkhöz rendelni (szimulált jelek esetében nincs szükség a megadásra)
- **Catergory**: Opcionálisan megadható paraméter, itt adhatjuk meg mely csoportba szeretnénk a jelünket besorolni.
- **Access Level**: Itt adhatjuk meg mely kliensek írhatják vagy olvashatják a jelünket. A **Default** beállítás esetén csak a helyi kliensek (pl.: betanító panel) vagy Rapid utasítások írhatják a jelünket. **Read only** beállítás esetén a jel nem írható. **All** beállítás esetén pedig bármely kliens írhatja a jelünket.

- **Default Value**: Alapérték.
- **Invert Physical Value**: A jel fizikai reprezentációja (elektromos feszültség) a logikai jel (0 vagy 1) negáltja legyen.
- **Safe Level**: **DefaultSafeLevel** beállítás esetén a jel az alapértelmezett értékét használja, amikor a rendszer elindul, és amikor a jel nem érhető el, majd amikor a jel hozzáférhetővé válik és a rendszer leáll, a jel az utolsó írott értéket veszi fel. **SignalSafeLevel** beállítás esetén pedig, a jel az alapértelmezett értékét használja a rendszer indításakor, illetve amikor a jel hozzáférhetővé válik vagy nem elérhető, majd a rendszer leállításakor a jel az utolsó írott értéket veszi fel.

Végezzük el a jel konfigurációját a 8. ábrának megfelelően.



8. ábra Jel konfigurálása

Ha elvégeztük a megfelelő beállításokat kattintsunk az ok gombra és indítsuk újra a virtuális vezérlőnket a változtatások mentése érdekében.

A következő lépés a Rapid programunk módosítása lenne, ehhez nyissuk meg a már előzőekben megírt programunkat. Rapidban digitális kimenetet a **SetDO** utasítással tudunk írni. Az utasításnak két paramétere van. Az első a jel megnevezése, amelyet írni szeretnénk, majd ezt követően jön vesszővel elválasztva a beállítani kívánt érték (0 vagy 1).

```
10        SetDO Hand, 1;
```

9. ábra SetDO utasítás szintaktikája

Módosítsuk a programunkat úgy, hogy a program legelején (még az első mozdulat megtétele előtt) nyissuk ki a megfogót. Ezt követően a felvételi pontban zárjuk össze a megfogót, majd a munkadarab letételénél nyissuk ki újra. A megfogó összezárásához a **Hand** kimenetet 0-ba kell állítanunk, szétnyitásához pedig 1-be.

A program módosítása előtt még vegyük figyelembe, hogy a fizikai folyamatok lezajlásához időre van szükség (pl.: a megfogó összezárása néhány másodpercet vesz igénybe). Tehát a kimenet írását követően várnunk kell néhány másodpercet, amíg a megfogó összezár vagy kinyílik, különben a robot túl hamar kezdené meg a következő mozgásutasítást. Az **ABB Technical Reference Manual – Rapid Functions, Instructions and Data Types** kézikönyv segítségével keressen olyan utasítást, amellyel 2 másodperces késleltetést tud beállítani a szükséges helyeken.

A program módosítást elvégezve szinkronizáljuk vissza a módosításokat a szimulációs térbe, majd futtassa a programot!

Kérdések:

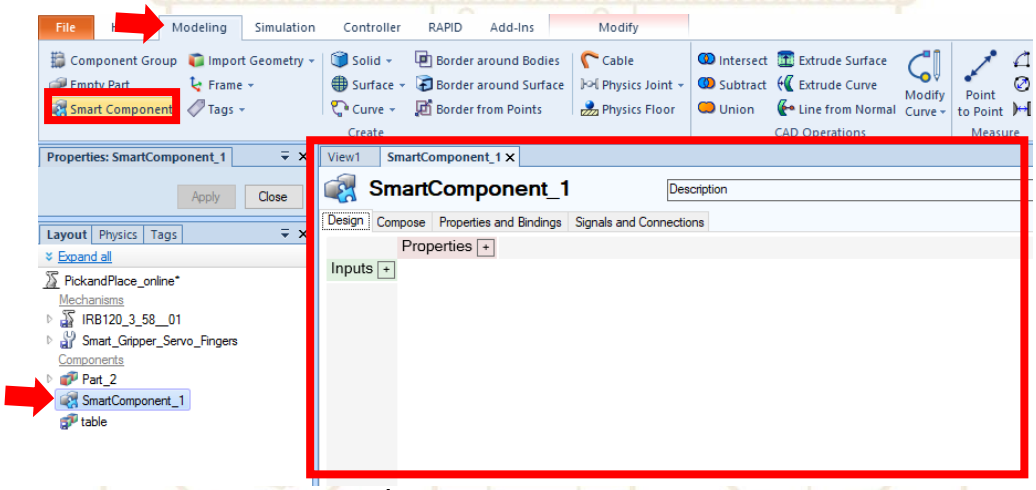
4. Mely utasítás segítségével tudta megoldani a késleltetést?
5. Mutassa be, hogyan módosította a programot! Megoldását indokolja.
6. Mit tapasztal a szimulációt futtatását követően?

4. feladat – Smart komponensek

RobotStudioban a **Smart** komponensek beépített logikával és tulajdonságokkal rendelkező blokkok melyeket a virtuális vezérlőtől független komponensek manipulálására, szimulációjára használunk (pl.: megfogó ujjainak mozgatása, munkadarab mozgatás a szimulációs térben)

Az előző részfeladat alapján jól látható, hogy habár a munkadarab mozgatására alkalmas robotprogramot elkészítettük, a szimulációs térben nem nyílik ki a megfogó és nem emeli fel a robot a munkadarabot. Ezen folyamatok szimulációjára fogjuk használni a **Smart** komponenseket.

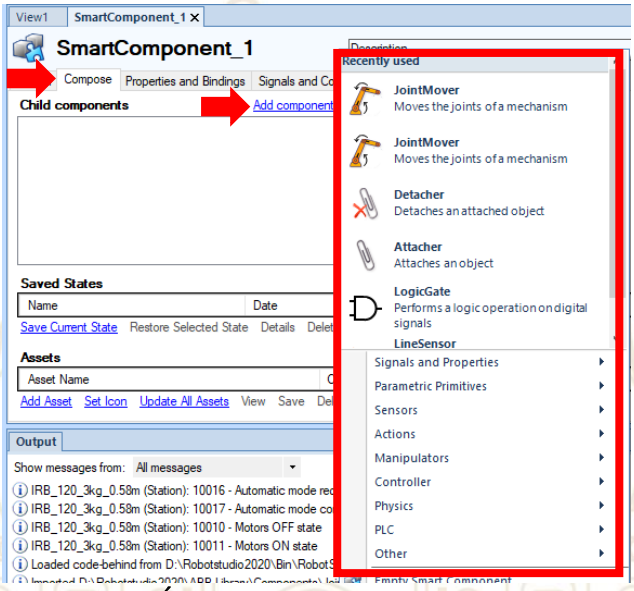
Új **Smart** komponens létrehozásához felső fülek közül válasszuk a **Modeling** opciót, majd a szalagmenüben kattintsunk a **Smart Component** menüpontra.



10. ábra Új Smart komponens létrehozása

A gombra kattintva a baloldali menüben, a **Layout** fül alatt megjelenik egy **SmartComponent_1** nevű új komponens, a középső ablakban pedig a **Smart** komponensük szerkesztésére alkalmas felület (10. ábra). Amennyiben a későbbiekben, újból szerkeszteni akarjuk majd a **Smart** komponensünket, jobb egérgombbal kattintsunk a nevére és válasszuk az **Edit Component...** menüpontot.

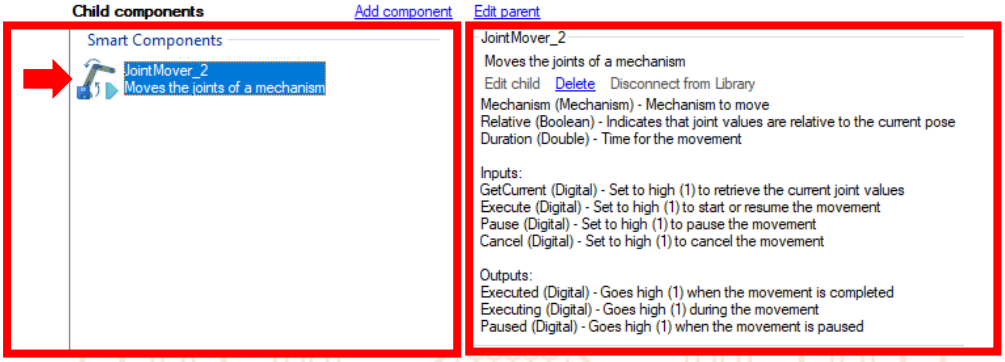
A saját **Smart** komponensünket egy grafikus programozási felület segítségével konfigurálhatjuk, a RobotStudioban található néhány alap komponens felhasználásával. Az alap komponensek különböző egyszerű funkciók megvalósítására képesek (pl.: Logikai kapuk, mozgások, parametrikus modellezés stb.). Ezen alap komponensekből építhetjük meg, saját bonyolultabb **Smart** komponensünket. Új komponens hozzáadáshoz válasszuk középen a **Compose** fület és kattintsunk az **Add component** menüpontra.



11. ábra Új Smart komponens létrehozása

A legördülő menüben csoportokba rendezve találhatjuk az alapkomponenseket. Adjunk hozzá egy **Manipulators>Jointmover** blokkot a saját **Smart** komponensünkhöz!

Középen a **Child Components** ablakban tekinthetjük, meg a saját **Smart** komponensünkhöz hozzáadott alap blokkokat. Bármely komponens nevére rákattintva jobboldalt egy rövid leírást találunk a komponens működéséről, tulajdonságairól és be- és kimeneteiről (12. ábra).



12. ábra Saját Smart komponenst felépítő alap blokkok és azok leírása

A **JointMover** blokk segítségével egy mechanizmus csuklóit mozgathatjuk.

Tulajdonságai:

- **Mechanism:** A mozgatni kívánt mechanizmus
- **Relative:** Itt azt adhatjuk meg, hogy relatív (jelenleg pozícióhoz viszonyítva) vagy abszolút módon kívánjuk megadni a célkoordinátákat
- **Duration:** Mozgás időtartama
- **Joint values:** Csuklókoordináták (Ez a tulajdonság csak a mechanizmus kiválasztását követően jelenik meg)

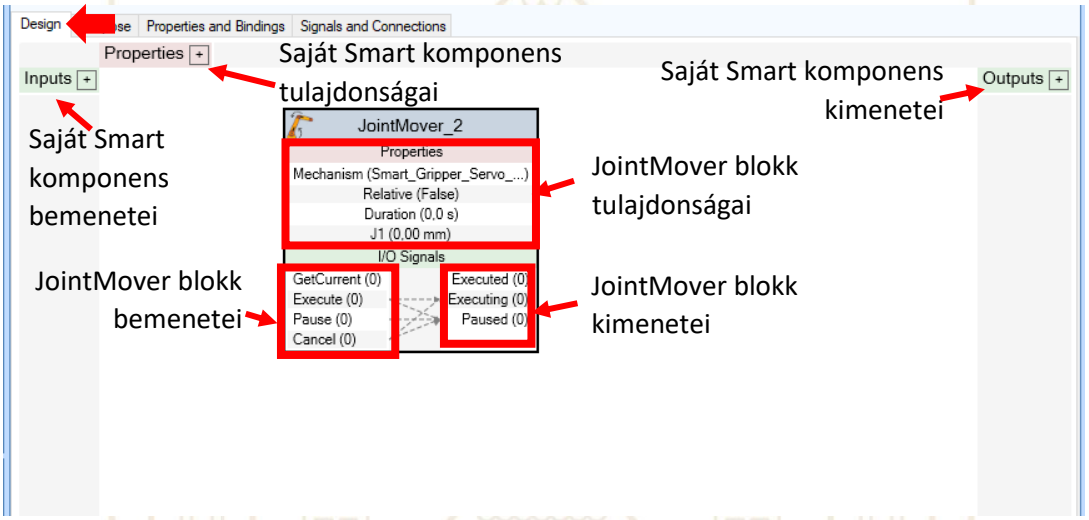
Bemenetek:

- **GetCurrent:** Ezt a bementet magas szintre kell hozni az aktuális csukló koordináták kiolvasásához
- **Execute:** A bemenet magas szintre állításával indíthatjuk el a mechanizmusunk mozgását
- **Pause:** Magas szintre állításával megállíthatjuk, szüneteltethetjük a mechanizmus mozgását
- **Cancel:** Magas szintre állításával megszakíthatjuk és törölhejük az aktuális mozgásfolyamatot

Kimenetek:

- **Executed:** A kimenet magas szintre vált, ha a mozgás befejeződött (elértük a célkoordinátákat)
- **Executing:** A kimenet magas jelszinten van, amíg a mozgás folyamatban van
- **Pause:** A kimenet magas jelszinten van amíg szüneteltetjük a mozgást

Amennyiben a **JointMover** utasítást hozzáadtuk saját **Smart** komponensünkhöz a **Design** fülre váltva, jól látható, hogy a szerkesztő felületen is megjelent az utasítást megtestesítő blokk (13. ábra).



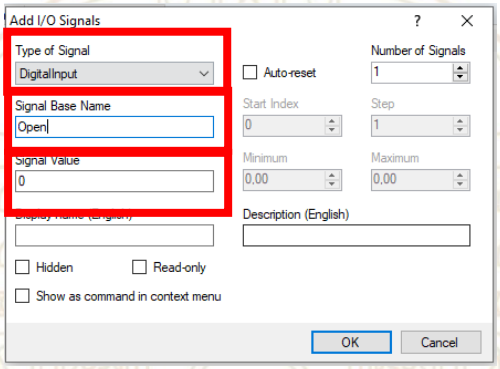
13. ábra Saját Smart komponenst szerkesztő felület

A szerkesztő felületen a fehér hatterű mezőbe kerülnek a saját **Smart** komponensünket felépítő alap blokkok. Továbbá látható, hogy a saját **Smart** komponensünk esetében is konfigurálhatunk tulajdonságokat, illetve be- és kimeneteket. A grafikus programozás során a Saját komponensünk és az egyes alap blokkok tulajdonságainak, illetve be is kimeneteinek összekötésével készíthetjük el saját szimulációs programunkat.

Adjunk hozzá egy bementet a saját **Smart** komponensünkhöz. Ehhez a kattintsuk rá baloldalon az **Inputs** felirat mellet található + jelre. A felugró ablakban válasszuk ki a jel típusát, nevezzük el bemenetünket és állítsuk a kezdőértéket 0-ra.

Figyelem!

Tulajdonság átadás csak abban az esetben lehetséges, amennyiben a két tulajdonságot leíró változó azonos típusú (pl.: mechanism, boolean).



14. ábra Saját Smart komponens digitális bemenet konfigurációja

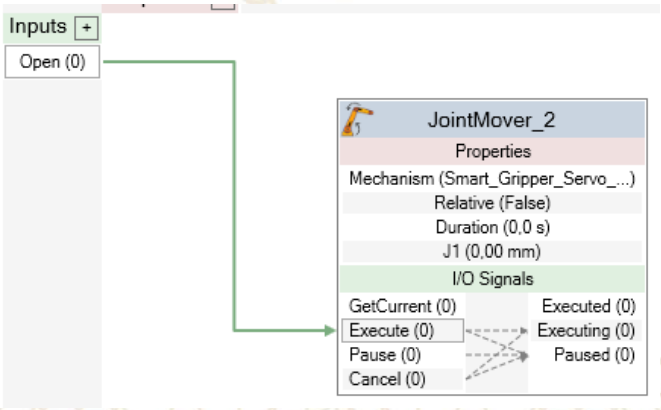
A jel típusa természetesen digitális bemenet lesz, hiszen a **Smart** komponensünket (a megfogó nyitásának és zárásának szimulációját) a **Hand** digitális kimenettel szeretnénk majd vezérelni.

Következő lépésként pedig konfiguráljuk a **JointMover** blokkot. Ehhez kattintsuk a komponensre a szerkesztő felületen. A blokk tulajdonságait a baloldalon megjelenő ablakban adhatjuk meg.

A következő beállításokat adhatjuk meg:

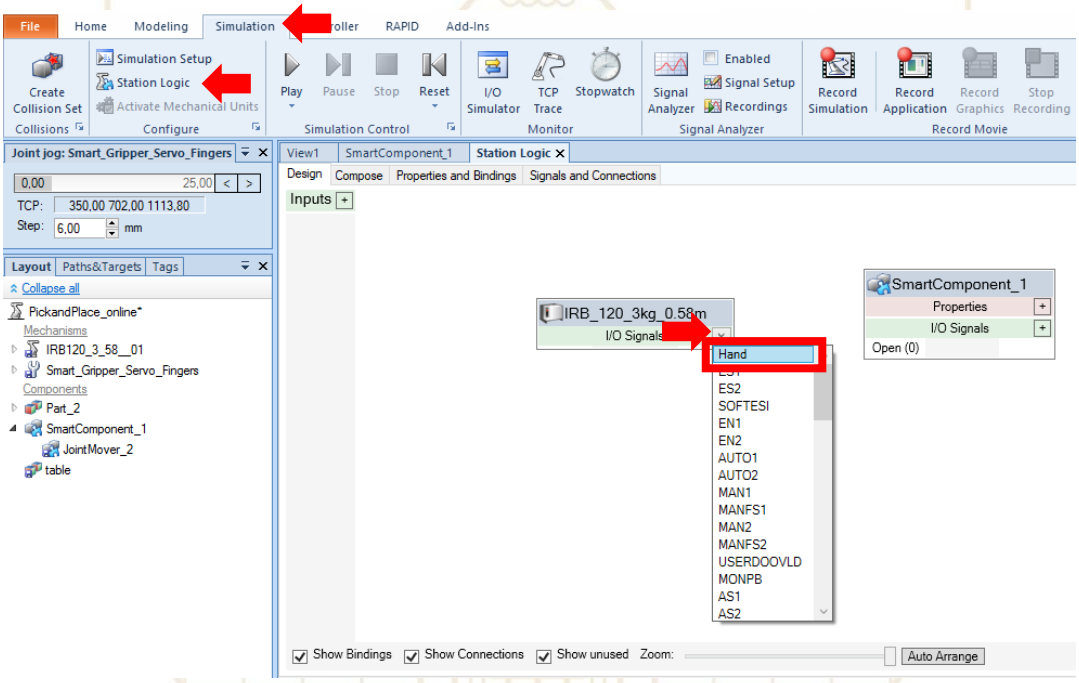
- **Mechanism:** Itt válasszuk ki a megfogót (a RobotStudio a megfogót is ugyanúgy mechanizmusként kezeli, mint mondjuk a robotot vagy egy futószalagot)
- **Relative:** Az egyszerűség kedvéért a koordinátákat abszolút módon adjuk meg, így ezt a tulajdonságot nem kell kipipálni.
- **Duration:** A megfogó ujjainak szétnyitására, állítsunk be egy 1s-os időintervallumot.
- **J1:** A csukló koordináta meghatározásához kattintsunk jobb egérgombbal a megfogó nevére a baloldali menüben. Majd a legördülő listából válasszuk a **Mechanism Joint Jog** utasítást. Ha középen kiválasztjuk a **View1** fület, jól látható, hogy a csukló koordináta növelésével kinyílik a megfogó. Nézze meg a csúszka segítségével meg a csukló koordináta értékét a megfogó teljesen nyitott állapotában és ezt az értéket adja a **J1** paraméternek.

Ha megvagyunk a **JointMover** blokk paraméterezésével, a **Smart** komponens szerkesztő felületén kössük össze az **Open** bemenetet, a **JointMover Execute** bemenetével (15. ábra).



15. ábra JointMover blokk vezérlése

Amennyiben megvagyunk a **Smart** komponensünk szerkesztésével, a következő lépés, hogy kialakítsuk a kapcsolatot a virtuális vezérlő és a **Smart** komponens között. Ehhez felül válasszuk a **Simulation** fület és kattintsunk a **Station Logic** menüpontra. Itt középen a szerkesztő felületen két blokkot látunk. Az egyik az általunk létrehozott **Smart** komponens (**SmartComponent_1**), másik a virtuális vezérlőnk (**IRB_120_3kg_0.58m**). A virtuális vezérlőnk ki- és bemeneteinek megjelenítéséhez nyissuk le a blokk alsó sarkában található nyilat és válasszuk ki a használni kíván jelet (16. ábra).



16. ábra Station Logic

Kössük össze a virtuális vezérlőnk **Hand** kimenetét a **Smart** komponensünk **Open** bemenetével. Innentől kezdve a **Smart** komponensünket a **Hand** digitális jel segítségével vezérelhetjük.

Módosítsa úgy az elkészített **Smart** komponenst, hogy

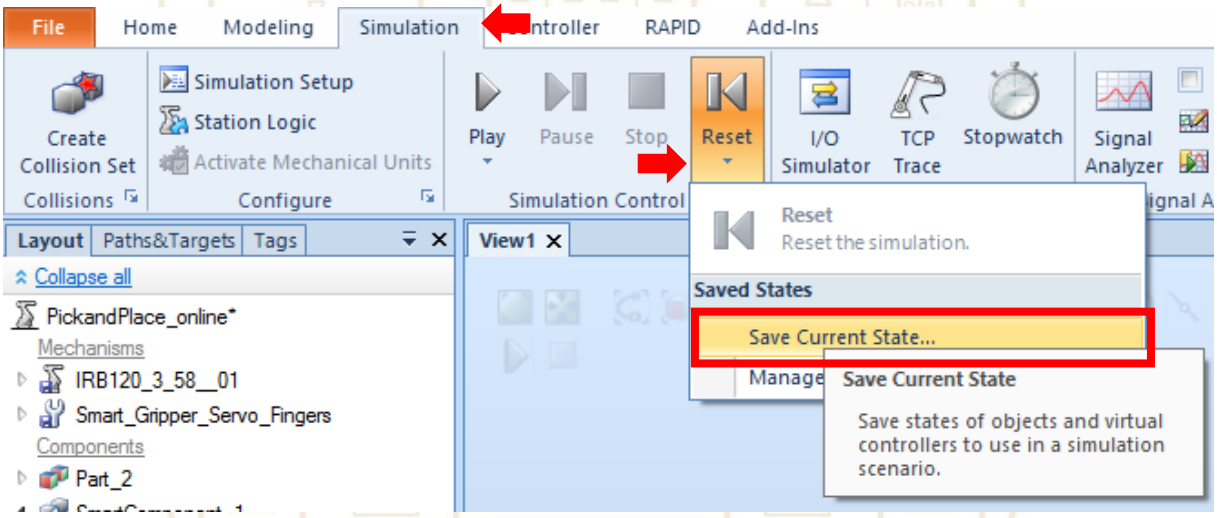
1. A **Hand** kimenet 0-ba állítása esetén összezár
2. Miután összezárt a megfogó a munkadarabot hozzá kényszerezzük a megfogóhoz
3. A **Hand** kimenet 1-be állítása esetén a megfogó teljesen kinyílik
4. Abban a pillanatban, hogy elkezd szétnyílni a megfogó töröljük a kényszereket a munkadarabról.

A **Smart** komponens megvalósításához a következő blokkokat adja még hozzá a rendszerhez:

- **JointMover**
- **LogicGate (NOT)**
- **Attacher**
- **Deattacher**

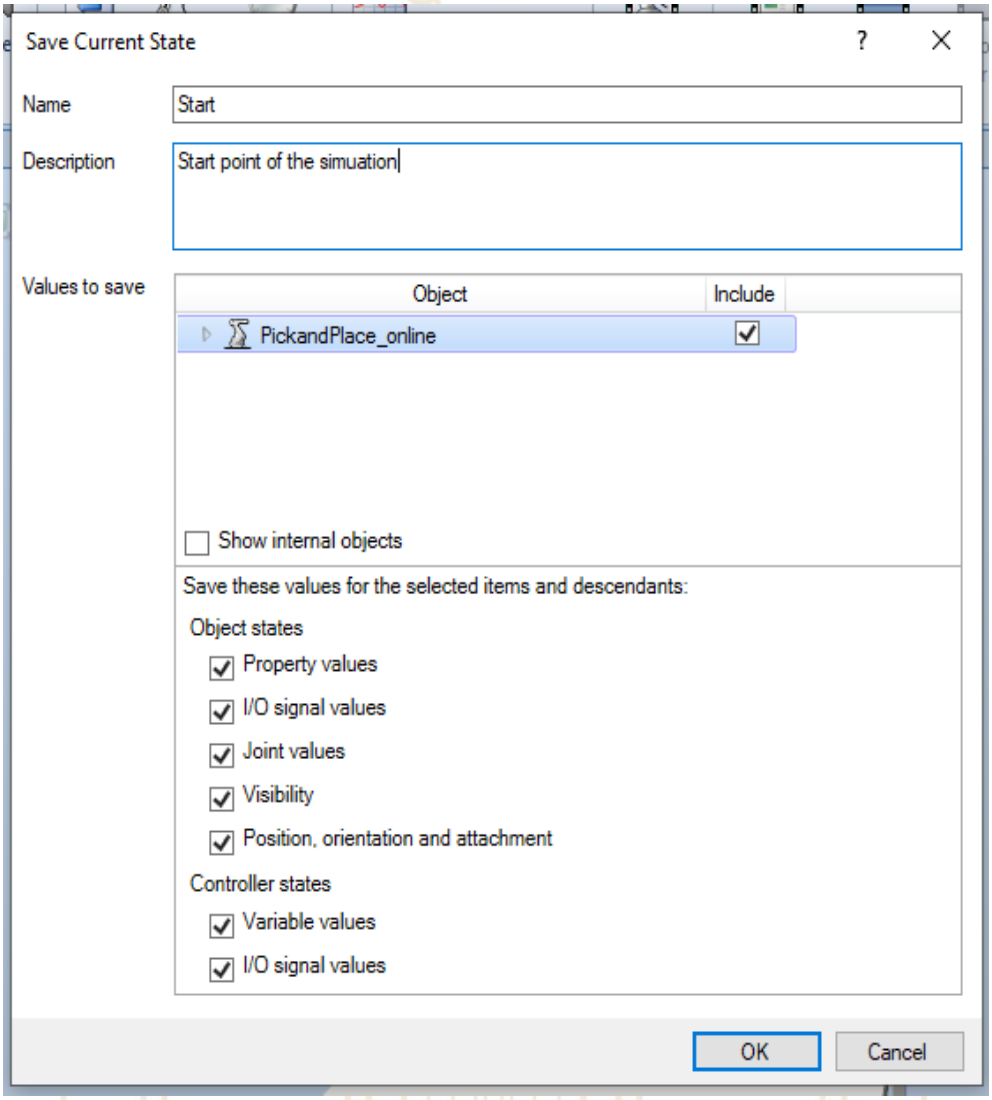
Konfigurálja az egyes blokkokat és alakítsa ki a szükséges logikai kapcsolatokat a komponensek között.

Mielőtt futtatnánk a szimulációt érdemes elmenteni a rendszer jelenlegi állapotát. Ehhez válasszuk felül a **Simulation** fület és a szalagmenüben nyissuk le a **Reset** menüpontot és kattintsunk a **Save Current State** opcióra.



17. ábra Szimuláció állapotának mentése

Végezzük el a 18. ábrának megfelelő beállításokat!

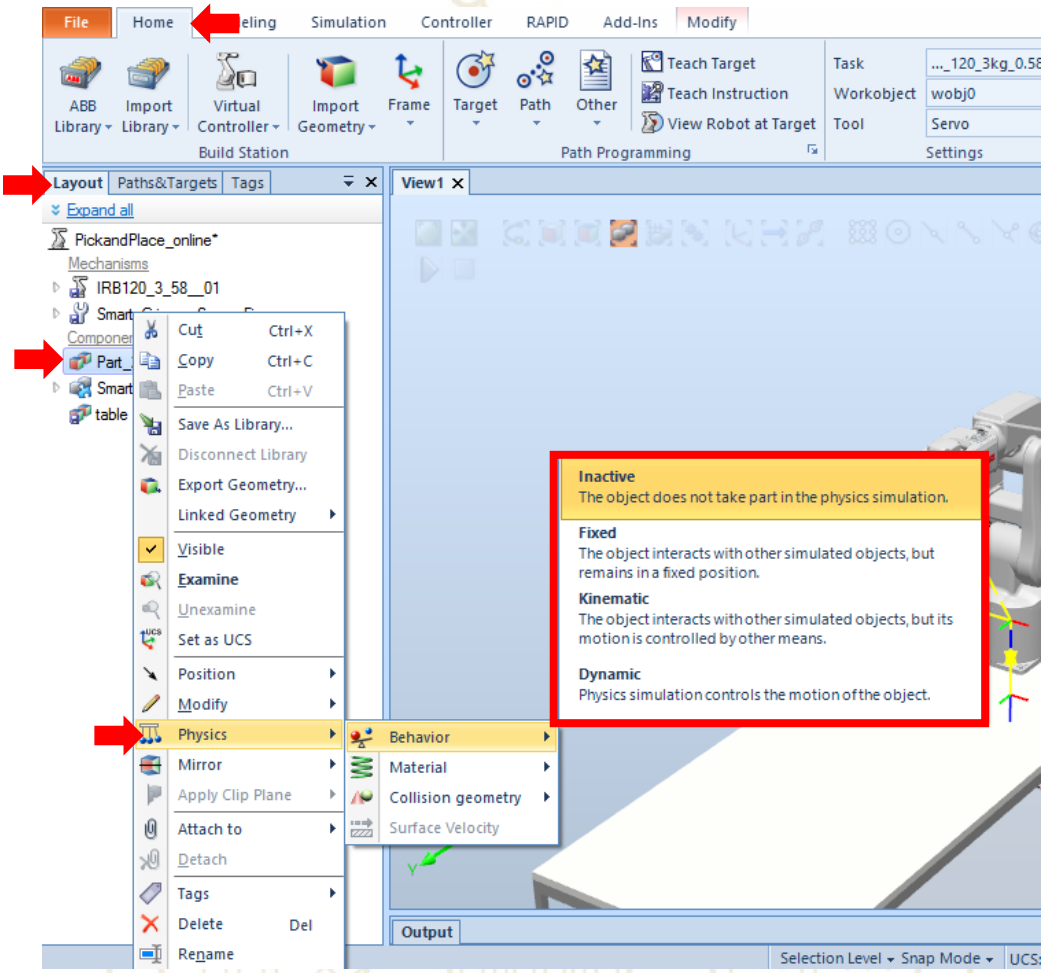


18. ábra Szimuláció állapotának mentése

Ettől kezdve a **Simulation** fület kiválasztva felül, majd a **Reset** gombra kattintva a rendszer a jelenlegi állapotába kerül vissza (a munkadarab visszakerül jelenlegi pozíciójába).

Futtassa le a szimulációt!

A **Smart** komponensek segítségével fizikai szimulációkat is készíthetünk a RobotStudioban. Ehhez minden objektum esetében be kell állítani a viselkedését. Kattintsunk felül a **Home** fülre, majd a baloldali menüben kattintsunk jobb egérgombbal a munkadarabunk nevére. A legördülő menüben válasszuk a **Physics>Behavior** opciót. Itt adhatjuk meg a munkadarabunk viselkedését (19. ábra). A **Smart** komponensen belül pedig a **Physics>PhysicsControl** segítségével készíthetünk szimulációkat!



19. ábra RobotStudio physics

Kérdések:

7. Mutassa be az Attacher smart komponenst! (Működés, tulajdonságok, be- és kimenetek)
8. Mutassa be az Ön által készített Smart komponens működését!

6. feladat – Egyéni feladat

Módosítsa úgy a szimulációt, hogy a Robot a munkadarabot az eredeti pozíciójához képest 90°-kal elforgatva tegye le az asztalra! Továbbá adjon hozzá egy kimenetet a **Smart** komponenséhez, amely a megfogó állapotát jelzi vissza (zárt vagy nyitott), majd konfiguráljon fel egy digitális bementet a virtuális vezérlőjén és kösse rá a **Smart** komponens kimenetét. Módosítsa úgy a programot, hogy a **SetDO** utasítást követően késleltesse addig a program futását ameddig a megfogó a megfelelő pozícióba nem állt.

Kérdések:

9. Mutassa be milyen módosításokat végzett a robotprogramban, illetve a Smart komponensben!