**A feladat célja**

Ebben a gyakorlatban egy kétkerekes robot viselkedését fogjuk programozni és elemezni a Webots szimulációs környezetben. A feladat két fő részből áll:

1. **Reaktív vonalkövető vezérlő implementálása**, amely segítségével a robot képes végigmenni egy zárt pályán, majd a kiindulási vonalhoz érve megáll.
2. **Odometria számítása** a keréksebességek alapján, hogy megbecsüljük a robot által megtett teljes távolságot és a pálya végén elért orientációját.

**Előkészületek**

A gyakorlat elvégzéséhez szükséged lesz a számítógépedre telepített **Webots** szoftverre. Ezenkívül szükséged lesz a korábbi leckében létrehozott line\_following.wbt nevű világleírásra és a line\_follower.py programvázra. Amennyiben ezek a fájlok nem állnak rendelkezésedre, kérjük, pótold a korábbi gyakorlati anyagok alapján.

E-puck mobil robot leírása: [link](https://webots.cloud/run?version=R2025a&url=https%3A%2F%2Fgithub.com%2Fcyberbotics%2Fwebots%2Fblob%2Freleased%2Fprojects%2Frobots%2Fgctronic%2Fe-puck%2Fprotos%2FE-puck.proto)

**Fontos:** Ez a gyakorlat nem csupán az elméleti tudásod egyszerű ellenőrzése. Minden kérdés egy-egy konkrét kísérlet elvégzését és az eredmények mélyebb megértését igényli a Webots szimulátorban. Mivel a gyakorlat értékelésre kerül, ne adj meg választ, amíg nem vagy teljesen biztos benne, és nem tudod azt egy megbízható kísérlettel vagy elemzéssel alátámasztani.

**1. Rész: Vonalkövető vezérlő implementálása**

A vonal követéséhez a robot három, lefelé néző távolságérzékelőjét fogjuk használni. A gyakorlat során ezeket egy g nevű listában tároltuk, ahol g[0] a bal, g[1] a középső, g[2] pedig a jobb oldali szenzor.

* Amikor a szenzorok a **fehér padló** fölött vannak, körülbelül **800**-as értéket adnak vissza.
* Amikor a **fekete vonal** fölött vannak, az értékük körülbelül **300**-ra csökken.

A célod, hogy minden időlépésben kiszámítsd a bal és jobb oldali kerekek szögsebességét és a szenzorértékek alapján.

A vezérlődnek az alábbi struktúrát kell követnie (a saját vonalkövető logikáddal kiegészítve):

*# Konstans a maximális sebességhez*

*MAX\_SPEED = 6.28*

*# ... a szimulációs ciklusban ...*

*# Szögsebességek inicializálása*

*phildot = 0.0*

*phirdot = 0.0*

*if (g[0] > 500 and g[1] < 350 and g[2] > 500): # Haladás egyenesen a vonalon*

*phildot, phirdot = MAX\_SPEED, MAX\_SPEED*

*elif(g[2] < 550): # Jobbra kanyarodás (jobb szenzor vonalat lát)*

*phildot, phirdot = 0.25 \* MAX\_SPEED, -0.1 \* MAX\_SPEED*

*# ... ide további "elif" ágak kellenek a balra kanyarodáshoz és egyéb esetekhez ...*

*# Motorok sebességének beállítása*

*leftMotor.setVelocity(phildot)*

*rightMotor.setVelocity(phirdot)*

A robot megbízható működéséhez finomhangolnod kell a fenti küszöbértékeket, és további if/elif feltételekkel kell kiegészítened a kódot.

**Javasolt gyakorlatok:**

1. **Egyetlen beállítási pont:** A fenti kódrészlet a keréksebességeket csak egy helyen állítja be a kódban. Ez olvashatóbbá teszi a programot, és később könnyedén felhasználhatod ezeket az értékeket az odometria számításához.
2. **Állapotmentes (reaktív) vezérlő:** A vezérlő nem tárol korábbi állapotokat, kizárólag az aktuális szenzorértékekre reagál.
3. **Finom irányítás:** A vezérlő a kanyarodás során lelassít (0.25 \* MAX\_SPEED) és nem végez túl agresszív fordulókat. A megfelelő értékek megtalálása időbe telhet.

**Megjegyzés:** Ahhoz, hogy a robot ne érzékelje azonnal a startvonalat a szimuláció elején, mozgasd egy kicsit előrébb a pályán.

Akkor léphetsz tovább a következő részre, ha a robotod már képes megbízhatóan végigmenni a pályán. A hibakereséshez manuálisan is mozgathatod a robotot, és használhatod a "gyorstekerés" (fast forward) gombot a szimuláció felgyorsításához.

**2. Rész: Odometria számítása**

Megtanultuk, hogyan számítható ki a robot elmozdulása a két kerék elfordulásából. A Webots (és a legtöbb valódi robot) a keréksebességet **radián per másodpercben (rad/s)** adja meg. Ezt megszorozva egy időlépés hosszával (a Webots timestep értékével, ezredmásodpercben), megkapjuk az adott lépésben történt elfordulást radiánban.

A robot elmozdulása (Δx) egy időlépés (Δt) alatt:

Itt *r* a kerék sugara (**0.0201 m**), *Δt* pedig az időlépés hossza (**32/1000 s**). Hasonlóképpen, a robot elfordulása (Δωz) is kiszámítható:

ahol *d* a két kerék középpontja közötti távolság (**0.052 m**).

Ezek az egyenletek lehetővé teszik, hogy minden időlépésben kiszámítsd a robot elmozdulását és elfordulását.

* A robot által megtett **teljes távolság** kiszámításához össze kell adnod az összes Δx értéket.
* A **teljes elfordulás** kiszámításához pedig az összes Δωz értéket.

**Feladat:** Implementálj egy egyszerű "útvonal-számlálót" (odométert), amely kiírja a robot által megtett teljes távolságot, valamint a jelenlegi orientációját. A kezdőértékek legyenek: távolság 0, orientáció 0 fok.

Az eredményeid ellenőrzéséhez használd a Webots "Scene Tree" nézetét, de ne keverd össze a "robot" és a "világ" koordináta-rendszerét – a robot a Webots világ y-tengelye mentén halad előre. Gondold végig, mire számítasz: mekkora lesz a robot orientációja az első jobb kanyar után? Az olvashatóság érdekében a radiánban kapott szöget érdemes átváltani fokra: (szög / π) \* 180.

Ne haladj tovább az alábbi kérdésekre, amíg az odometriád nem működik, és a számított orientáció eltérése legfeljebb 10-20% a várt értékhez képest.

**Ellenőrző kérdések**

A lenti kérdések megválaszolásához futtasd a szimulációt a kész vonalkövető és odometria kóddal.

**1.Kérdés**  
Milyen orientációra számítasz (elméletileg), amikor a robot egy teljes kör után visszatér a célvonalhoz?

* 360 fok
* 0 fok
* 90 fok
* -360 fok

**2.Kérdés**  
A robot odometriáját használva (az elmozdulásokat összeadva a robot saját tengelye mentén), mekkora a pálya teljes hossza? Méterben add meg a választ (pl. 1.04, ami 104 cm-nek felel meg).

A kísérlet eredménye: 3.077m

**3.Kérdés**  
Mekkora a robot által becsült tényleges orientáció fokban, miután visszaért a célvonalhoz?

A kísérlet eredménye:

**4.Kérdés**  
Észrevehetted, hogy a robot hajlamos túlbecsülni az elfordulás szögét. Vizsgáld felül a szög kiszámításához használt egyenletet, és válaszd ki az alábbi lehetőségek közül az egyetlen lehetséges okot:

* A tengelytáv (d), amit használunk, túl nagy, és csökkenteni kell.
* A kerekek sugara (r), amit használunk, túl kicsi, és növelni kell.
* Eltérés van a robot által ténylegesen elért sebesség és a programban beállított sebesség között.

**5.Kérdés**  
Mi az odometriai hiba csökkentésének egy egyszerű módja? Válaszadás előtt próbáld ki a lehetőségeket a Webots szimulátorban.

* Lassabb haladás
* Az összesített értékek csökkentése egy konstans szorzóval.
* A keréksugár értékének csökkentése a képletben.