Vezérlési szerkezetek (CONTROL programcsoport), szenzorok egyszerű használata (SENSORS programcsoport)

A programnyelveknél a „vezérlési szerkezetek” kifejezéssel a legtöbb esetben három olyan programozási konstrukciót jelölnek, amelyek az összetevők (utasítások) végrehajtási sorrendjét határozzák meg.

1. Szekvencia

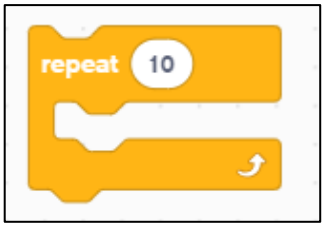
Az utasítások (blokkok) egymás utáni végrehajtása. Gyakorlatilag, amikor a programnyelvben fentről-lefelé egymás után fűzzük az utasításblokkokat, akkor ezzel a végrehajtási sorrendet is megadjuk.

2. Ciklus (iteráció)

Az utasítások egy feltételtől függően többször hajtódnak végre. A feltétel lehet például egy egyszerű számlálás, amikor előre megadjuk, hogy hányszor történjen a végrehajtás, vagy lehet egy esemény bekövetkezése.

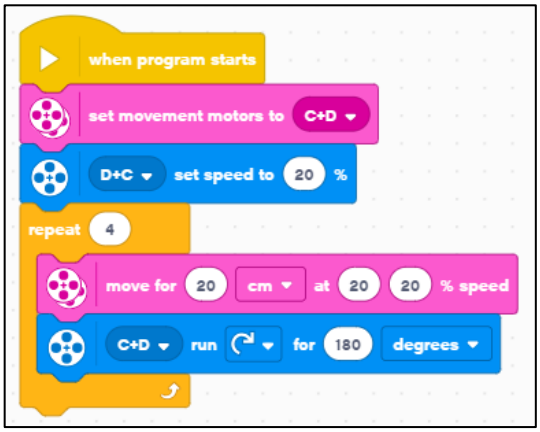
3. Elágazás (szelekció)

Egy feltételtől függően a programnak adott helyzetben mást kell elvégeznie. A feltétel sok esetben egy esemény vagy logikai állítás, amelynek bekövetkezése, vagy igazzá válása esetén a programnak mást kell csinálnia, mint egyébként. Bonyolultabb esetben nem csak kétféle kimenetelt lehet beállítani, hanem megszámlálhatóan sokat.



A ciklusok közül a legegyszerűbb az úgynevezett növekményes ciklus, amelynél előre tudjuk, hogy az utasításszálon szereplő blokkok egy csoportját hányszor kell egymás után végrehajtani. Minden ilyen ciklushoz tartozik egy számláló (ciklusváltozó), amelynek az értéke minden egyes végrehajtás során eggyel nő (0-tól kezdve a számlálást), így elérve a beállított értéket a ciklus futása leáll és a programszál következő utasításával folytatódik a végrehajtás. Az ismétlendő utasítások alkotják a ciklusmagot, amelyet a blokk két vízszintes eleme közé kell helyezni. A ciklusfejléc mutatja az ábrán, hogy jelenleg 10-szer fogja végrehajtani a program a ciklusmagban szereplő utasításokat.

Az előző fejezetben bemutatott 1. példa rövidíthető a növekményes ciklus használatával, mert vannak benne ismétlődő utasítások. Például a négyzet alakú pályán történő mozgás kódját mutatja a jobb oldali ábra.



Sok esetben nem tudjuk előre megmondani, hogy hányszor kell végrehajtani a ciklusmag utasításait. Az ismétlések száma valamilyen feltételtől, eseménytől függ. Ilyen eseményt generálhatnak például a szenzorokkal mért értékek.

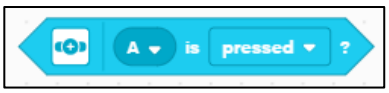
Jelenleg az alaprobotunkra három szenzor csatlakozik:

A port – ütközés/nyomás érzékelő
B port – fény/szín érzékelő
F port – távolság érzékelő

A robot szenzorai kétféle típusú értéket adhatnak vissza a programnak: logikai értéket vagy számot. A logikai érték igaz/hamis lehet, míg a számok akár előjeles valós számok is (de jellemzően nemnegatív egészek).

A SENSORS programcsoportban szereplő blokkok ennek megfelelően az alakjukban eltérnek. A logikai értéket visszadó blokkok hatszög alakúak, míg a számértéket visszaadó blokkok lekerekített végű téglalapok. Ezeket az utasításblokkokat tehát nem közvetlenül a programszálra kell kapcsolni, mint utasítást, hanem az általuk szolgáltatott értékeket lehet felhasználni más blokkok paramétereiként.

Az A portra kötött ütközésérzékelő által visszaadott logikai érték (be van nyomva/nincs benyomva → igaz/hamis).



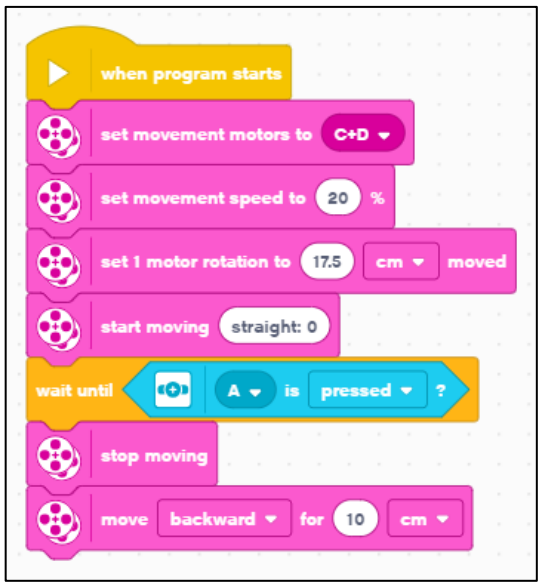
A B portra kötött fényszenzor által visszaadott fényintenzitás érték (a visszavert fény intenzitása → 0-100 közötti szám, minél sötétebb, mattabb a felület, annál kisebb).

A mért értékek felhasználhatók a vezérlési szerkezeteknél a ciklusok, elágazások feltételeinek megadására.

2. Példa – Akadály észlelés

Írjunk olyan programot, amelynél a robot egyenesen előre halad mindaddig, amíg az ütközésérzékelőjét be nem nyomták! Ekkor tolasson vissza 10 centimétert!

Megoldás



A cikluson belül most nincsenek végrehajtandó utasítások, mert a mozgást a ciklus előtt el lehet indítani. A ciklusnak csak annyi a szerepe, hogy figyelje az ütközésérzékelőt és várjon mindaddig, amíg be nem következik az az esemény, hogy megnyomták a szenzoron lévő gombot. Erre alkalmas a „*wait until*” szerkezet. Ennek hatására addig várakozik a blokknál a programszál utasításainak végrehajtása, amíg a beállított feltétel teljesül. Ezután a végrehajtás a következő utasítással folytatódik.

A motort a „*wait until*” blokk előtt kell elindítani (a mozgás „időtartamát” ne adjuk meg, hiszen nem tudjuk mikor fogják megnyomni az ütközésérzékelőt). A „*wait until*” blokk után pedig állítsuk meg a mozgást (*stop moving*). Az utolsó utasítás a 10 cm-es tolatás.

3. Példa – Adaptív sebesség radar

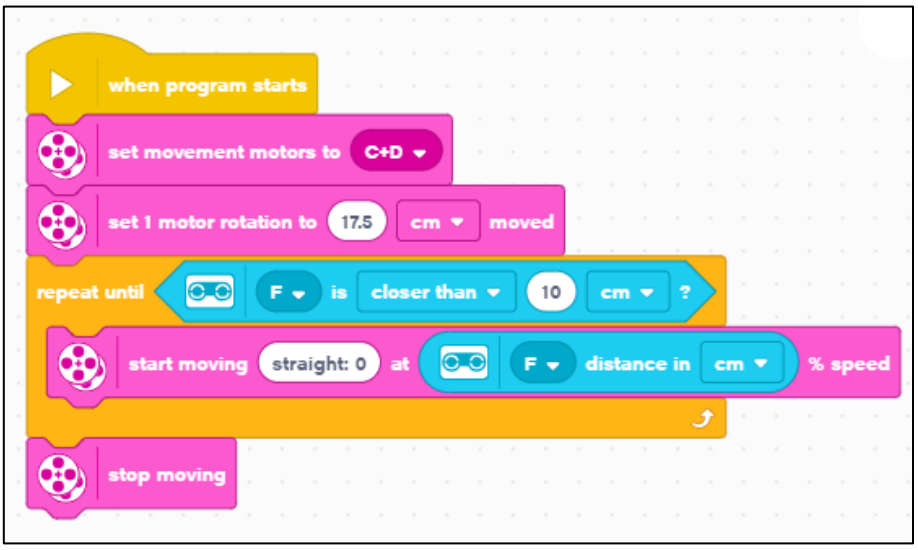
Írjon programot, amelyet a robot végrehajtva egy adaptív sebességtartó automatikát szimulál! Az autókba beépített adaptív sebességtartásnak a lényege, hogy ha akadály kerül (pl. lassabb jármű) az autó elé, akkor csökkenti a sebességet az ütközés elkerülése érdekében.

A feladat szerint a robot haladjon állandó sebességgel (ez legyen 100%), és az ultrahang szenzorával mért távolság függvényében változtassa ezt a sebességet! Ha 10 cm-re megközelítette az akadályt, akkor álljon meg!

Megoldás

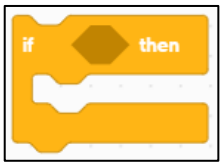
A szokásos „*set*” beállítások után egy úgynevezett előltesztelő ciklust használunk. Itt nem tudjuk előre, hogy hányszor kell végrehajtani a ciklusmagot. A leállítást egy szenzor által adott feltétel fogja garantálni. A ciklus futása során folyamatosan figyeljük a távolságérzékelő aktuális értékeit, és ha 10 cm-nél kisebb értéket érzékel a rendszer (*closer than 10 cm*), akkor a feltétel igaz lesz és a ciklus leáll. A ciklusmagban egyetlen utasítás szerepel. A MORE MOVEMENT csoportban található blokknál a mozgás sebességét lehet megadni. Szerencsére a motor sebessége nem címezhető túl. Ezt azt jelenti, hogy motorsebességként 0-100 közötti érték adható, de ha nagyobb számot kap a rendszer, akkor nem ad hiba üzenetet, hanem a maximális 100%-os sebességet tartja. Tehát a távolságmérő pillanatnyi értéke fogja meghatározni a robot sebességét. Egy fal felé közeledve ez egyre kisebb érték lesz, így a robot is folyamatosan lassul, míg eléri a 10 cm-es értéket. Ekkor megáll, mivel leáll a ciklus is.

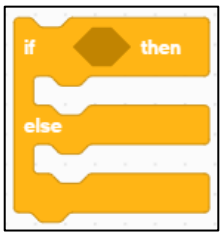
Az előltesztelő jelző a ciklusnál azt jelenti, hogy a ciklusmag utasításainak végrehajtása előtt megvizsgálja a feltételt. Ha az már a kezdéskor is igaz, akkor egyszer sem hajtja végre az utasításokat. Tehát ha a robot 10 cm-en belül áll az akadály előtt, akkor el sem indul.



Alapvetően megjegyezhető, hogy a blokkokon belül minden hatszög alakú szövegdobozba (logikai érték) betehető logikai feltétel, míg a kerek szélű téglalap alakú szövegdobozokba (szám típusú érték) olyan blokkok, amelyek számokat adnak eredményül.

A harmadik vezérlési szerkezet az elágazások. Itt egy logikai feltételtől függően más-más utasítást tud a robot végrehajtani. Lehet olyan elágazást készíteni, amelynél a feltételtől függ, hogy végrehajtsa-e az utasítást a robot, de ha nem teljesül a feltétel, akkor ne csináljon semmi mást, és lehet olyan elágazást létrehozni, ahol van egyébként ág, tehát ha a feltétel nem teljesül, akkor megmondhatjuk, hogy mi legyen a végrehajtandó utasítás.





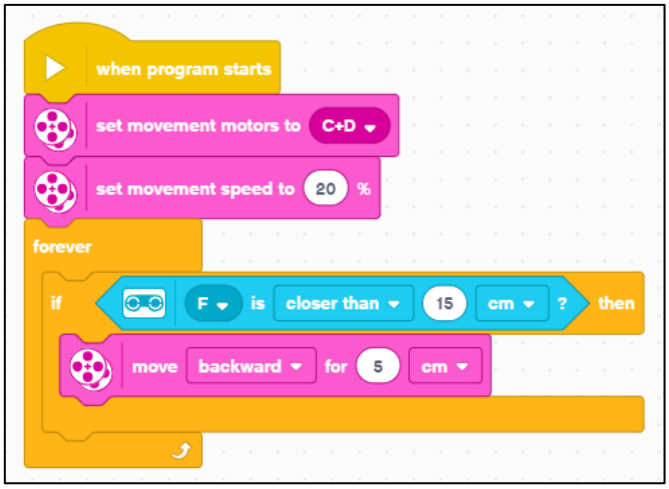
4. Példa – Menekülő robot

Írjon olyan programot, amelyet végrehajtva a robot tolat 5 cm-t, ha 15 cm belül észlel maga előtt valamilyen akadályt! Ha nincs ilyen akadály, akkor álljon egyhelyben.

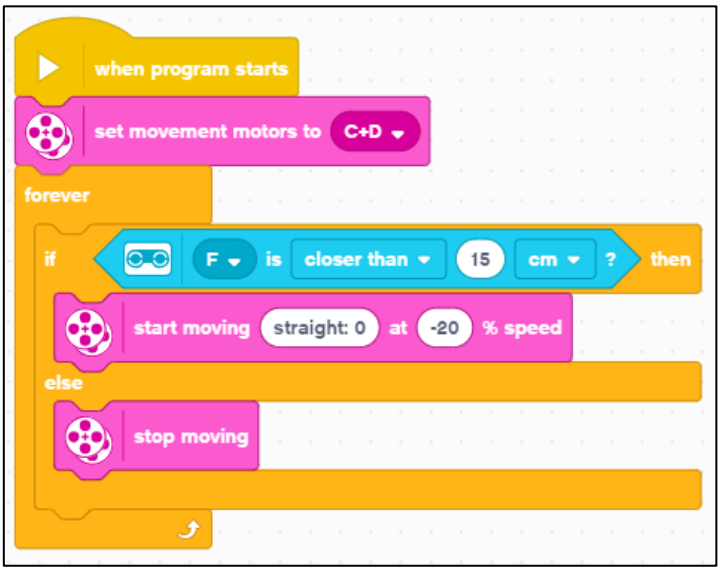
Megoldás

A feladatspecifikáció szerint csak akkor kell mozognia a robotnak, ha 15 cm-en belül akadályt lát. Így egy egyszálú elágazást lehet használni.

A feltételt az előző feladatnál már látott módon állítjuk be. Az utasítás, amit a feltétel igaz állapota esetén végre kall hajtani, egy 5 cm-es tolatás. Ha ebben a formában illesztenénk a kódot a programszálra, akkor csak egyszer hajtaná végre a robot. Ha 15 cm-en belül állna egy akadály előtt a programindítás pillanatában, akkor tolatna és utána véget is érne a programja. Ha úgy indítanánk a programot, hogy nincs a robot előtt akadály, akkor nem mozdulna, de mivel a feltételt megvizsgálta, ezért le is állna a programja. Amennyiben azt szeretnénk, hogy a feltételt folyamatosan figyelje és reagáljon, akkor egy ciklusba kell helyezni az elágazást. Ez lehet most egy végtelen ciklus (elvileg csak akkor áll le, ha direkt leállítjuk, vagy lemerül az akkumulátor). Így már működik.



A program futása nem szép, mert a robot nem folyamatosan tolat (menekül), hanem 5 cm-es szakaszokban. Ha azt szeretnénk, hogy a mozgása folyamatos legyen, akkor kétszálú elágazást érdemes használni. Az egyébként (*else*) ágra a *stop moving* blokkot tehetjük, az igaz ágon pedig csak bekapcsoljuk a motorokat.



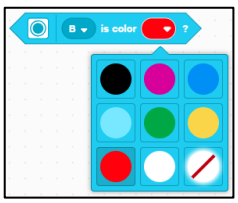
A *start moving* blokk esetén a negatív sebesség a tolatást eredményezi (ellentétes forgásirány a motoroknál).

5. Példa – Útvonalkövetés

Írjon programot, amelyet a robot végrehajtva fényszenzorával követi a fehér alapon lévő fekete színű vonalat!

Megoldás

A feladat bevezetéseként írjunk egy olyan programot, amelynél a robot balra fordul, ha a fényszenzora az indításkor fekete színt érzékel és jobbra egyébként.

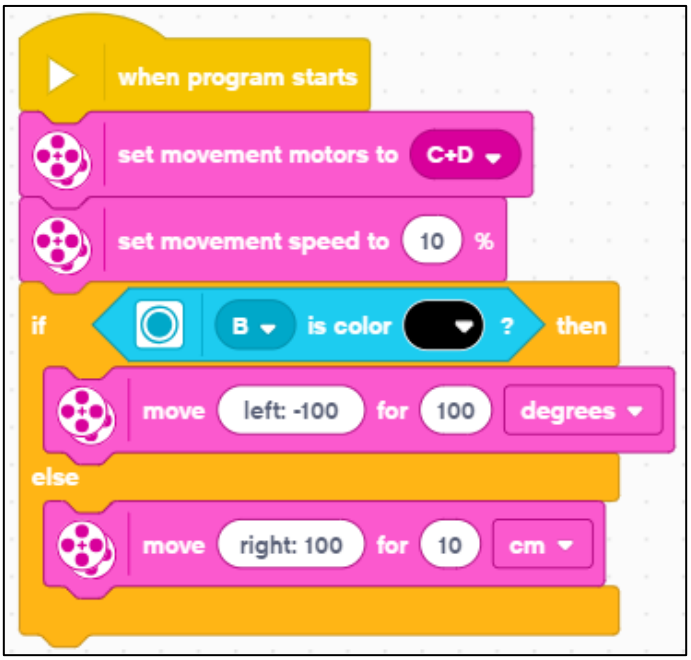


A fényszenzort a programban kétféle üzemmódban tudjuk használni. Színt képes érzékelni vagy a felületről visszavert fény intenzitását. Színérzékelés esetén nyolc különböző színt képes megkülönböztetni (a kilencedik az egyéb szín).

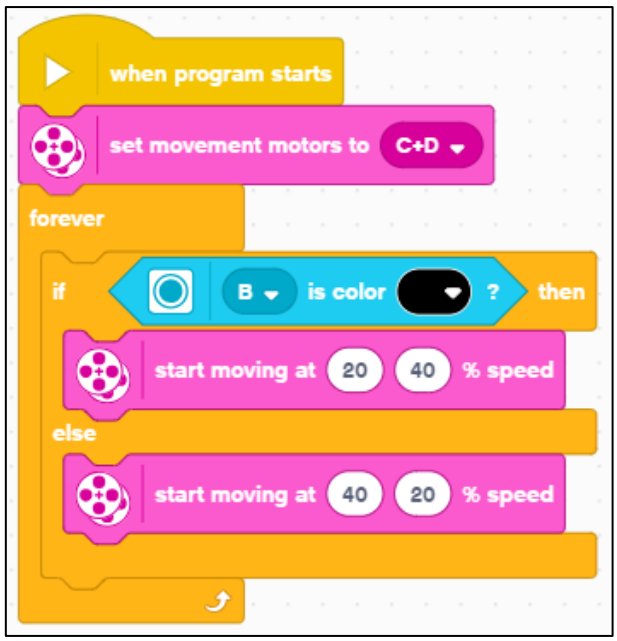
Fény üzemmódban egy 0-100 közötti értéket ad eredményül, a felület színétől, fényességétől függően. A sötét matt felületeknél alacsonyabb az érték.

A program elkészítéséhez a motorok sebességét állítsuk kicsire (pl. 10%)! Az elágazás vezérléséhez most a színszenzor logikai értéket visszaadó blokkját használjuk. A forráskódot az ábra szemlélteti.

A robot összesen 100 fokot fordít a kerekein. Az irány attól függ, hogy az indításnál fekete vagy nem fekete színt érzékelt-e a szenzora.



Ha a programunkat úgy alakítjuk át, hogy ne csak egyszer hajtsa végre az elágazás kiértékelését, hanem folyamatosan és ennek megfelelően reagáljon jobbra vagy balra fordulással, akkor egy olyan általánosan használható programhoz jutunk, amely képes követni egy fekete vonal jobb vagy bal szélét, kígyózó mozgással. Javítanunk kell még a programon annyit, hogy a forgások ne konstans értékűek legyenek, hanem csak különböző sebességgel forgassuk a motorokat. Ha konstanst állítunk be fordulási szögnek, akkor addig nem vizsgálja újra az elágazás feltételét, míg le nem forogta a beállított szöget.
Ciklusként használhatunk végtelen ciklust.



Érdemes a robot mozgásán megfigyelni az algoritmus működését. A jelenlegi programmal az útvonal menetirány szerinti bal szélét követi a robot, hiszen fekete szín érzékelésénél a jobb oldali motor forog gyorsabban, tehát balra fordul. Ha a szenzora lekerült a fekete felületről, akkor pedig jobbra fordul.

Ha az útvonal másik oldalát szeretnénk követni, akkor fel kell cserélni az elágazás két ágát, vagy meg kell fordítani a motorok sebességének előjelét.

Finomítani lehet az útvonalkövetés pontosságát, ha a motorok sebeségei közötti különbséget változtatjuk. A robot reakcióideje miatt, ha kicsi a sebességkülönbség, akkor nagy ívű lesz a fordulás és könnyen átkerül a szenzor az út rossz oldalára és elveszíti az útvonalat (megfordul). Tehát éles kanyarokat vagy keskeny útvonalat lassú sebességekkel és nagyobb sebességkülönbséggel érdemes követni. Mindig az adott útvonal dönti el a beállítandó értékeket.

Ha a végtelen ciklus helyett például a távolságérzékelőt használjuk az útvonalkövetés befejezésének érzékelésére, akkor komplex pályákat is be lehet járni a robottal. Például akadályig kövesse az útvonalat, majd ott forduljon meg és induljon el visszafelé. …

Az alábbi program két akadály között útvonalkövetéssel mozgatja a robotot amíg le nem állítjuk a programot. Akkor működik helyesen a program, ha a színszenzor kb. a robot hosszanti tengelyén van. Az alaprobotnál a jobb oldalon található, így az akadály észlelésekor, a 180 fokos fordulásnál a szenzor messze kerül az útvonaltól és nem fog rátalálni a visszafelé követésnél. Ezt úgy korrigálhatjuk, hogy az akadály elérésekor nem 180 fokos fordulót állítunk be, hanem kisebbet vagy nagyobbat a fényszenzor helyzetétől függően. A lényeg, hogy a fordulás után a fényszenzor az útvonal „jó” oldalán álljon, így, ha ismét elindul az útvonalkövetési ciklus, akkor automatikusan rá fog állni az útra.