# A XXI. század menekítő csapata

*LegoRockers*

*Bálint Karola, Gilicze Kristóf, Sebők Attila*
*Csongrádi Batsányi János Gimnázium, Szakképző Iskola és Kollégium*
*6640 Csongrád, Kossuth tér 1.*

## 1. Bevezetés

Tegyük fel, hogy valahol valamilyen katasztrófa történt, mondjuk egy földrengés következtében megsérült egy ház, vagy egész városrész, ipari létesítmény esetleg egy atomerőmű. Ha fontos ezeknek a területeknek az átjárhatósága (pl. stratégiai útvonal, utánpótlás szállítás), akkor a megváltozott terepviszonyok között új útvonalat kell keresni.

Ezt a feladatot robotokkal célszerű megvalósítani, mert így nem kockáztatunk emberéletet. A cél az volt, hogy gyorsan rövid útvonalat találjunk, ami a katasztrófa sújtotta területen átvezet. A mi megoldási ötletünk arra épül, hogy több robot hatékonyabban oldja meg ezt a problémát. A teszteléshez készítettünk egy modellt, ahol a terepet labirintus formájában ábrázoltuk. A robotok egymás között bluetooth-szal kommunikálnak.

## 2. Probléma megoldásának menete

### 2.1. A labirintus

A labirintus derékszögű, egyforma falapokból lett építve egy bejárattal és egy kijárattal, ami az 1. ábra középső képén jól látható. .Az elrendezése tetszőlegesen variálható, a labirintuscellák közepén egy fekete vonal segíti a robot közlekedését. Ez a modell első megközelítése a valóságos helyzetnek.

### 2.2. A robotok

Három robot (egy master, két slave) kommunikál egymással. A két slave robot felépítése (1. ábra, baloldali fotó) megegyezik: kisméretűek, mozgékonyak, három ultrahangos távolságérzékelővel és egy fényérzékelővel tájékozódnak.

A masternek (1. ábra, jobboldali fotó) nincs szüksége távolságérzékelőre, mert a slave-ek által küldött információk alapján határozza meg saját útvonalát.

**1. ábra. A slave robot, a labirintus és a master robot**

### 2.3. A stratégia

A két slave robot egymás után elindul a labirintusban, minden labirintusegységben megállnak, információt gyűjtenek környezetükről, ezt továbbítják a masternek. A master robot tárolja, kiértékeli, nyilvántartja a bejárt pozíciókat és a feldolgozott adatok alapján felváltva továbbhaladási utasítást küld a két slave robotnak. Az egyik robot esetén a jobb oldalt, a másiknál a bal oldalt részesíti előnyben. Miután mindkét slave robot kijut a labirintusból, a master kiválasztja a rövidebb útvonalat, és ezt követve, de az esetleges zsákutcákat kihagyva, áthalad a labirintuson.

Többféle optimalizálása létezik a problémának: a legrövidebb úton, vagy a lehető leghamarabb átjutni a labirintuson.

### 2.4. Az algoritmus

A labirintus egységeket koordináta párokkal (x;y) azonosítjuk. Ezek értéke segít meghatározni az aktuális pozíciót és a labirintus végét.

A slave robot minden labirintusegységben megvizsgálja az adott pozícióból a továbbhaladási lehetőségeket. Az utat 1-essel, a falat pedig 0-val jelzi. Ezt a kódhármast küldi a masternek, mely válaszként egy karaktert (b,e,j,h) küld. A slave robot ennek megfelelően balra, előre, vagy jobbra megy. A h egy zsákutcát jelöl, ilyenkor a slave megfordul, és előre megy. Lényegében ez ismétlődik, amíg a slave ki nem jut a labirintusból.

A master kiválasztja az optimalizált útvonalat, majd a következő zsákutcakezelést alkalmazza: minden pozíciót megvizsgál, hogy szerepel-e még egyszer a tervezett útvonalban. Ha szerepel, akkor a pozíció utolsó előfordulására ugrik.

### 2.5. Adatszerkezet

Az adatok tárolásához két kétdimenziós tömböt használtunk. Egyikben a jobbra tartó slave, a másikban a balra tartó slave robot által küldött, valamint a master robot által feldolgozott információk találhatók, ahogyan ez az 1. számú táblázatban látható.

| X | **Y** | Falak | Mozgás |
|---|---|---|---|
| 1 | 1 | 000 | J |
| 1 | 2 | 101 | E |
| 1 | 3 | 110 | B |
| 2 | 3 | 111 | H |

**1. táblázat. Koordináták és utasítások**

### 2.6. A programozás

A programokat RobotC nyelvben fejlesztettük. Összesen két program szükséges, az egyik a masteren fut, a másik a slave roboton (mindkettőn ugyanaz a programkód).

### 2.7. Problémák

Kezdetben NXTG-ben programoztunk, de végül „zsákutcába” jutott fejlesztésünk, mert a file-ban eltárolt adatokat körülményesen tudtuk kezelni. Ekkor döntöttük el, hogy RobotC-ben írjuk meg a programot.

Ebben a fejlesztő környezetben a bluetooth kommunikációt kettő robot között is nehéz volt összeegyeztetni, a harmadik robot pedig már igazi kihívást jelentett.

A külső körülmények (megvilágítás, ragasztott pályaelemek, súrlódás) nehezítették az algoritmus tesztelését. Végül ezt is sikerült megoldanunk.

## 3. Elért eredmények

A labirintus-bejárási feladatok másfél-két éve kerültek a látóterünkbe, ez idő alatt többféle szempontból megközelítettük a problémát:

- Megismerkedtünk tisztán elméleti bejárási stratégiákkal (rekurzív, és nem rekurzív megoldások).
- Foglalkoztunk teljes labirintus bejárással, és a bejárathoz visszajutó a robot algoritmusával.
- A gyakorlatban történő teszteléshez mindig robotokat használtunk.
- Többféle méretű labirintusban, több robotkonstrukcióval valósítottuk meg a fenti algoritmusokat.

Az ilyen képességű robotok, a katasztrófahelyzetekben alkalmazott un. menekítő robotok, amelyek vészhelyzetben emberi életet, és egyéb értéket menthetnek, szerepet kaphatnak a szennyezett területek mentesítésében is.

Az általunk megírt algoritmust továbbfejlesztve egy olyan menekítő csapatot lehetne létrehozni, mely hatékonyan és gyorsan működik. Kettő, vagy akár több robot feltérképezi a területet, megtalálja a legoptimálisabb útvonalat, majd egy további robot ezen az útvonalon egyszerűen végighalad, és megteszi a szükséges intézkedéseket. Ez lényegesen gyorsabb és biztonságosabb, mint ha minden emberekre lenne bízva.