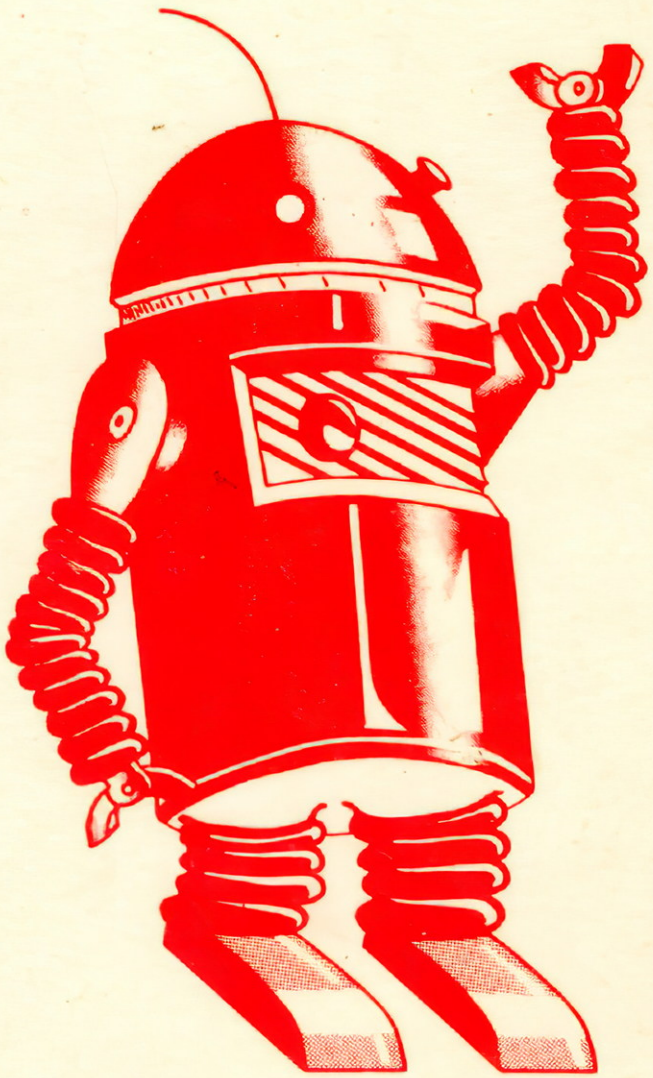
Steigers

A robotok és a Commodore 64



DATA BECKER – NOVOTRADE

Steigers

A robotok és a Commodore 64

DATA BECKER – NOVOTRADE

A könyv eredeti címe: Das Roboterbuch zum Commodore 64

Fordította: HARTYÁNYI MÁRIA
Lektorálta: ILA LÁSZLÓ
DR. LENGYEL JÓZSEF

A kiadásért felel: RÉNYI GÁBOR, a NOVOTRADE RT. igazgatója
Budapest, 1987
Műszaki szerkesztő: DÉVÉNYI ERIKA

Szedte a Nyomdaipari Fényszedő Üzem, Budapest
Készült a Somogy Megyei Nyomdaipari Vállalat kaposvári üzemében (8 A/5 ív)
Felelős vezető: MIKE FERENC igazgató

ISBN 963 02 4876 X

FONTOS TUDNIVALÓ

A jelen könyv keretén belül ismertetett kapcsolások, eljárások és programok nem tekinthetők szabadalmi oltalom alá eső ipari termékeknek. Ezek elsősorban amatőr és oktatási célokat szolgálnak. A szerzők rendkívül nagy gondot fordítottak a kapcsolások, műszaki adatok és programok helyességére, a részletek kidolgozása során többszöri ellenőrzést végeztek. Mindez azonban nem zárja ki az esetleges hibalehetőségeket.
Az előforduló hibákért és az ebből adódó következményekért a DATA BECKER cég sem szavatosságot, sem jogi felelősséget nem vállal. Az esetlegesen előforduló hibák közlését a szerzők hálásan fogadják.

TARTALOMJEGYZÉK

ELŐSZÓ

Tízéves lehettem, amikor valósággal megigéztek a tudományos-fantasztikus filmek és könyvek főszereplői, a robotok.

Lelkesedésem nem maradt meg a puszta csodálatnál, hanem egyre erősebben ösztökélt arra, hogy saját magam is készítsek a filmekben látottakhoz hasonlatos szerkezetet.

Minden tehetségemet és kézügyességemet latba vetve meg is építettem az első, általam robotnak nevezett valamit, ami lakásunkban ma is kitüntetett helyen díszeleg.

Természetesen roppantul bosszantott, hogy művem nem egészen úgy sikerült, mint amilyenek a filmek mesterséges főszereplői voltak.

Ahelyett azonban, hogy az elégedetlenség kedvemet szegte volna, tovább folytattam a kísérletezést, és az eredmény nem is maradt el: robotjaim egyre sokoldalúbbakká váltak. Az első fabábu – amely mindössze egy villanykörte bekapcsolására volt képes – hamarosan egész cselekvéssorozat elvégzésére alkalmas automatává fejlődött.

A cselekvéssorozatot előre elkészített program határozta meg. Persze ezt a programot nem úgy kell elképzelni, mint a mai személyi számítógépes programokat, hiszen akkoriban ezek a gépek még teljesen ismeretlenek voltak. A programot lyukszalagon, vagy egyéb, a maihoz képest technikailag igen kényelmetlen alkalmatosságon lehetett csak előkészíteni.

Így bár a robotgyártás nagyon hosszadalmas és sok vesződséggel járó feladatot jelentett számomra, ez sem csökkentette kísérletező kedvemet. Az a nevezetes nap, amikor az első programozható játék birtokába jutottam, alapvetően megváltoztatta, sőt túlzás nélkül állíthatom hogy forradalmasította a tevékenységemet.

Ma már az olcsó, programozható számítógépek egész sora áll az érdeklődők rendelkezésére, és közülük árát és képességeit tekintve az egyik legkiemelkedőbb a Commodore 64-es gép. Ez a gép kiválóan alkalmas különböző vezérlési feladatok megoldására, és óriási előnye, hogy sokoldalúan bővíthető. Bővíthetőségének szinte csak gazdája fantáziája szab határt.

Könyvemben szeretném bevezetni az Olvasót a robotépítés sajátos világába, és bízom abban, hogy sikerül segítséget nyújtanom ahhoz, hogy ki-ki az olvasottak nyomán született saját ötleteit meg is tudja valósítani.

Remélem, hogy az élmények és eredmények sokukból kitörlik majd az efféle foglalatoskodással szembeni előítéleteket.

Engedjük, hogy magával ragadjon bennünket a kísérletező kedv, amely mindennél jobban megmozgatja a fantáziát és felébreszti az alkotási vágyat!

Jó munkát és sok örömet kívánok a könyv olvasásához!

Jürgen Steigers

Viersen, 1985. április

1. BEVEZETÉS

Napjainkban egyre szélesebb körben terjednek a különböző feladatok ellátására kifejlesztett automaták és robotok. Alkalmazásuk célja elsősorban az emberi munka megkönnyítése, vagy az emberi munka egy-egy részének gazdaságos kiváltása.

A robotok egyik típusát gyakran illetik a nem túlságosan megtisztelő „bádogidióta" névvel, ami megkülönbözteti ezeket az egyszerű végrehajtó szerkezeteket újonnan fejlődésnek indult „intelligens" társaiktól, amelyek nem egyebek mint „végtagokkal" ellátott számítógépek.

Az utóbbiak élénk manókhoz hasonlíthatóak, amelyek képesek többé-kevésbé intelligens cselekedetek végrehajtására. Alkalmas programmal egy tárgyat elmozgatnak pl. x helyzetből y helyzetbe úgy, hogy közben beépített érzékelőikkel ügyesen kikerülik az útjukba kerülő akadályokat.

Manapság már az sem meglepő, hogy ha az effajta szerkezetek emberi hangot utánozva üdvözlik a családhoz érkező látogatókat.

Kézenfekvő a különbség az ipari robotok és a háztartásbeli robotok között. Míg az előbbiek a modern gyártási folyamatok lebonyolításának precíz és gyors eszközei, addig a házi robotok a modern technika sokkal szerényebb igényű, tanulási célokra vagy szórakoztatásra szánt termékei.

Ez a könyv tartalmazza a mechanika, elektronika és programozástechnika alapszintű ismereteit, amelyek elegendőek ahhoz, hogy a Commodore 64-es számítógépből házi robotot építsünk. Egyben némi betekintést igyekszik nyújtani a fiatal tudományág, a robotika területére. A leírtak megértése az elektronika és a forrasztás alapszabályain kívül más előismeretet nem igényel. A bemutatott kapcsolásokat általában könnyű elkészíteni, és egyikük megépítéséhez sem szükséges magát a Commodore 64-es alapgépet átalakítani.

Az elkészült kapcsolásokat az alapgépen a user portra (a felhasználói bemenetre), ill. az expansion portra (bővítő bemenetre) csatlakoztathatjuk, ill. ezeken keresztül vezérelhetjük.

Szó esik a könyvben arról is, hogy hogyan lehet egy robotot a Commodore 64-gyel összekötő kábel nélkül programozottan vezérelni.

Ugyancsak nem marad említés nélkül a kibernetikus vezérlés területe sem.

A kibernetikus modellek azok, amelyek leginkább képesek a szemlélőt lenyűgözni az intelligens lények benyomását keltve.

A még tapasztalatlan barkácsolók számára közöljük, hogy házilagos kivitelezés esetén mivel helyettesíthetők az egyes alkatrészek.

Bár a könyv alapvetően gyakorlatias szellemben készült, úgy gondoljuk, hogy az elméletről sem szabad tökéletesen megfeledkeznünk. Így némi betekintést nyújtunk egyebek között a kibernetika-, a vezérlés- és szabályozástechnika területére, és röviden összefoglaljuk az embert utánzó szerkezetek, az automaták történetét.

2. AZ ALAPOK

2.1 Mi a robot?

Ma már lehetetlenség pontosan meghatározni, hogy a történelem során mikor és hol merült fel először a mesterséges lények megépítésének gondolata, az azonban bizonyos, hogy a gondolat megszületése jóval megelőzte a modern technika kialakulásának korát.

Az ókori görögök legendáiban számos utalást találunk az ún. androidokra, amelyek a leírás szerint embert utánzó szerkezetek lehettek. Homérosz hőskölteményében az Iliászban olvashatunk például arról, hogy Hephaisztosz isten kívánságait olyan arany szolgálólányok teljesítették, amelyek nem hús-vér lények voltak.

A ránk maradt történetek szerint Ikarusz édesapja is kísérletezett mesterséges lények építésével.

Az ókori görögök legendáikban az isteneket és embereket egyaránt felruházták az androidok megalkotásának képességével. Egyáltalán nem meglepő, hogy Arisztotelész, a görög filozófus egész elméletet épített fel arról, hogy hogyan befolyásolja majd a társadalom szerkezetét az emberi munka elvégzésére alkalmas androidok elterjedése. A későbbi korokból származó írások tanúsága szerint az automaták megépítésének gondolata egyetlen történelmi kor emberét sem hagyta nyugodni.

Ma már persze sok esetben nehéz rekonstruálni, hogy a történetek főhősei – pl. a láthatatlan erő mozgatta szobor – valóban léteztek, vagy pusztán az emberi fantázia szüleményei voltak. Az azonban biztos, hogy a mesterséges lények alkotóit a középkorban istenkáromlással vádolták, hiszen a vallás szerint a teremtés képességével egyedül az Isten rendelkezett.

Először a felvilágosodás korában nyertek némi elismerést azok az emberek, akik erre az akkoriban hétköznapinak egyáltalán nem nevezhető foglalkozásra adták fejüket. A 18. században már megkülönböztetett tisztelet övezte az emberformájú zenélő automata, az írni tudó gépember és a sakkautomata alkotó mestereit.

Egyébként épp ez utóbbi, a sakkautomata volt az, amelyen lemérhetők az adott korban elérhető eredmények korlátai.

A 18. században már önmagában az is csodálatos teljesítmény volt, hogy ezeket a szerkezeteket tisztán mechanikus úton, fogaskerekek, bütykök, csapszegek és egyéb elemek bonyolult szövevényével vezérelni tudták.

A sakkautomata mesterséges intelligenciájának előállításához azonban a legtehetségesebb finommechanikai mester felkészültsége is kevésnek bizonyult.

A világhírű „Török" néven emlegetett sakkautomata helyett valójában a mechanikus szerkezetek mögé ügyesen elrejtett élő ember játszott, aki rejtekhelyéről jól láthatta a sakkasztalt, és a mechanika segítségével mozgatni tudta a bábukat.

Az iparosodás kezdetén egyre nőtt az érdeklődés a szórakoztató automaták iránt, felmerült ugyanis a lehetőség, hogy hasonló szerkezeteket az ipari gyártási folyamatokban is alkalmazni lehetne. Az új automaták továbbra is tisztán mechanikus elven működtek, de már képesek voltak bizonyos munkafázisokban az embert helyettesíteni. Az új meghajtó erő felfedezése azonban forradalmasította az egész ipart, és egyben elősegítette a mai ipari robotok őseinek világrajöttét.

Definíciószerűen megadni, hogy mit is értünk a robot fogalma alatt, meglehetősen nehéz feladat. Először 1921-ben K. Capek író kísérletezett a fogalom körülírásával, meghatározása azonban nem tekinthető általánosnak, sokkal inkább csak a filmek és könyvek fantáziaszüleményeire alkalmazható.
Szerinte a robot az ember mesterséges mása, felruházva az ember képességeinek egy töredékével, és képes a szemlélőben azt a benyomást kelteni, hogy önállóan mozog és cselekszik.

A modern robotokat fejlettségi szintjük szerint három csoportba sorolhatjuk.

Az első csoportba tartoznak az ún. telemanipulátorok, amelyeket az emberek közvetlenül kézzel irányítanak. Ezek a gépek elsősorban arra alkalmasak, hogy az ember helyett a nehéz vagy veszélyes terheket mozgassák. Ilyen robotokat alkalmaznak például a nukleáris kísérletekben, a radioaktív anyagok kezelésére.

A telemanipulátorok önálló mozgásra képtelenek, közvetlenül vezérlik minden mozdulatukat.

A második csoportba sorolhatjuk azokat az ipari robotokat, amelyek képesek egy rögzített program végrehajtására és így önállóan el tudják végezni valamely gyártási folyamat egy meghatározott részét. Nagy szerepük van az állandóan ismétlődő munkafázisok mechanikus végrehajtásában, mentesítve ezzel az embert az egyhangú fizikai és pszichikai megterhelés alól.

A harmadik generációhoz tartozó robotok a jövőt vetítik elénk, és ezek felelnek meg a leginkább, az említett meghatározásnak.

Ezek a robotok beépített szenzoraik segítségével kapcsolatot tudnak teremteni a környezetükkel. A betáplált programot nem gépiesen, hanem „rugalmasan"

hajtják végre, alkalmazkodva minden esetben a pillanatnyi körülményekhez. A robotok második és harmadik generációja közötti lényeges különbség érzékeltetésére nézzünk egy példát.

Ha a második típusú robot feladata bizonyos tárgyak átrendezése, akkor a művelet hibátlan végrehajtásához a tárgyaknak mindig ugyanazon a helyen kell lenniük, ha ugyanis az eredeti helyzet nem olyan, mint amelyre a robotot programozták, az átrendezés eredménye sem felel majd meg az elvárásoknak.

Ezzel szemben a harmadik generációs robot képes a tárgyakat átrendezés előtt azonosítani.
Az ilyen típusú robotnak nem kell minden művelet előtt azonos körülményeket teremtenünk, hiszen rugalmasan reagál a környezetbeli változásokra.

Persze az emberi fantázia képességei mögött még a harmadik típusú robotok is messze elmaradnak. A tudományos-fantasztikus filmek és könyvek főhőseinek képességei elérik, sőt esetenként túlszárnyalják az ember képességeit. Az ilyen robotok előállítására a legmodernebb technika sem eléggé fejlett.

A 18. század embere által megálmodott technikai csodák azonban ma már valósággá váltak, legalábbis a szórakoztató automaták tekintetében.

A modern elektronika fantasztikus távlatokat nyitott a robotok gyártásában. Ma már mind az ipari, mind a háztartási robotokat beépített számítógépek vezérlik. A háztartási robot tulajdonképpen a háztartási számítógép továbbfejlesztése, és mint ilyen a programozás felhasználásában is teljesen új területeket nyit. Annak ellenére, hogy a háztartási robotok még általában csak a szórakoztatásra alkalmasak, megjelenésük és terjedésük jelentőségét nem szabad lebecsülni. Építésük és programozásuk előtanulmányokat jelenthet későbbi, az emberi tevékenységet megkönnyítő fejlettebb változataik építéséhez és programozásához.

Sőt talán nem túlzás azt állítani, hogy működési elvüket tekintve a mai robotjátékszerek már nagyon közel állnak a jövő robotjaihoz, amelyekre az ember komoly segítőtársként számíthat.

2.2 Vezérlés és szabályozás

A robottechnika két alapfogalmát a vezérlést és szabályozást a mindennapi szóhasználatban gyakran nem különböztetjük meg egymástól, holott a szaknyelvben egészen eltérő jelentést hordoznak. Szakmai jelentésük alapvető fontossága kellő indok ahhoz, hogy tisztázásukra szánjunk néhány sort.

2.2.1 A VEZÉRLÉS

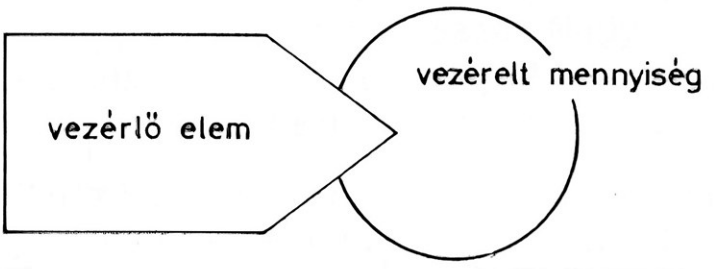
A két fogalom közül érdemes először az egyszerűbbel, a vezérlés fogalmával megismerkedni. Tegyük fel, hogy fürödni szándékozunk, és szándékunktól indíttatva vizet eresztünk a kádba. Természetesen eközben arra törekszünk, hogy a fürdővíz hőmérséklete a lehető legkellemesebb legyen, azaz ne legyen se túl meleg, se túl hideg.

Megnyitjuk a meleg- és hidegvizes csapot olyan mértékben, hogy a víz hőmérséklete legjobban megfeleljen elvárásainknak, és várunk míg a kád meg nem telik.

Mindezen felelősségteljes műveletek közepette nem feltétlenül gondoljuk végig azt a fizikai törvényszerűséget, miszerint a fürdővíz hőmérsékletét a meleg és hideg víz keverési aránya határozza meg.

Persze ha időközben bekövetkezik valami előre nem látható változás (túl nagy a huzat a fürdőszobában és a víz gyorsabban kihűl mint gondoltuk), az eredmény nem felel meg a várakozásunknak.

A vezérlés fogalmát a példa jól szemlélteti: a vezérelt mennyiséget (a víz hőmérsékletét) közvetlenül befolyásoljuk egy vezérlő elem (a vízcsap) segítségével.



1. ábra. A vezérlés alapelve

A vezérlés lényeges tulajdonsága, hogy vezérlés közben az előre nem látható, nem várt zavarókörülmények (pl. a huzat) hatását nem lehet kiküszöbölni.

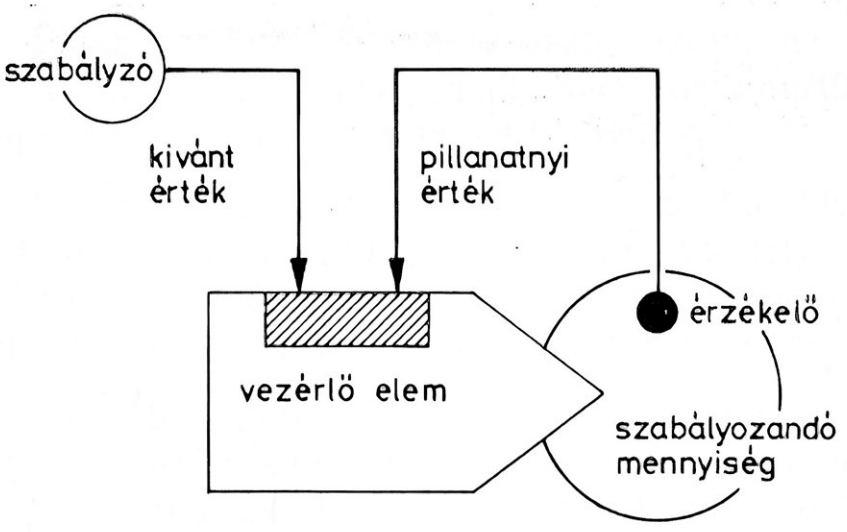
2.2.2 A SZABÁLYOZÁS

Most képzeljünk el előző példánkhoz kapcsolódva egy „okos" fürdőkádat, amely képes saját maga meghatározni a fürdővíz végső hőmérsékletét. Hogyan?

A feladat megoldásához szükséges intelligenciát egy a kádba beépített hőmérő szolgáltatja, amely folyamatosan méri a víz pillanatnyi hőmérsékletét, lehetőséget teremtve ezzel a kád számára, hogy az a meleg és hideg víz további keverési arányáról „döntsön".

Mármost ha időközben valaki a legjobb szándéktól vezérelve a kádba egy vödör hideg vizet lódít (pusztán azért, mert tudja, hogy a túlságosan meleg vízben való fürdés árt az egészségünknek), intelligens fürdőkádunk a zavaró körülményt azonnal észleli, és a hibát kellő mennyiségű forró víz adagolásával korrigálja.

A fentiek alapján kiderül, hogy a szabályozáshoz szükség van egy szabályzóra, amelyen a kívánt mennyiséget (a fürdővíz hőmérsékletét) beállíthatjuk, egy vezérlő elemre (esetünkben a vízcsapra), amellyel a kívánt mennyiséget (a víz hőmérsékletét) befolyásolhatjuk, végül egy érzékelőre, amely folyamatosan méri a beállítandó mennyiség aktuális értékét, a pillanatnyi hőmérsékletet). A szabályzó a vezérlő elemet állítja, ha eltérést észlel a pillanatnyi és a kívánt érték között.



2. ábra. A szabályozás alapelve

2.2.3 A SZABÁLYOZÁS ALAPELVE

A szabályozási folyamatot végigkíséri egy jelenség, nevezetesen a beállítandó mennyiség értékének ingadozása. Ez azt jelenti, hogy a pillanatnyi és a kívánt érték szabályozás közben sohasem egyezik meg pontosan, a pillanatnyi érték folyamatosan ingadozik a kívánt érték körül.

Az ingadozást, minthogy az egész szabályozó rendszer működését befolyásolja, előbb-utóbb mesterségesen meg kell szüntetni. Az optimális rendszert éppen az jellemezhetné, hogy képes ingadozás nélkül elérni a beállítandó mennyiség kívánt értékét, és ha az időközben valamely zavaró körülmény hatására ismét megváltozna, a különbséget képes kiegyenlíteni.

Valójában az „okos fürdőkád" nem egyéb mint egy önszabályozó rendszer, amelynek behatóbb vizsgálata a kibernetika területére esik.

2.3 Rövid bevezetés a kibernetikába

A kibernetikát, mint tudományágat az ötvenes években Norbert Wiener alapozta meg mozgó rendszerek modellszerű, matematikai leírásával.

Mozgó rendszerek alatt olyan önálló szabályozó rendszereket értünk, amelyekben állandó információáramlás zajlik le a vezérlő és a vezérelt egység között. Legjobb példa az önszabályozó rendszerekre az emberi test. A szemünk a rendszer megfigyelő objektuma, azon belül a szivárványhártya lehetővé teszi, hogy alkalmazkodjunk a különböző fényviszonyokhoz. Ha sötét helyiségben tartózkodunk, a pupilla kitágul, hogy minél több fényt befogadhasson. Erős napsütésben pedig összehúzódik, és csak annyi fényt ereszt át, amennyi a látáshoz feltétlenül szükséges.

A pupilla nyílása mindig megfelel a környezetbeli fényviszonyoknak, a pupilla tehát, mint a kibernetikus rendszer igen fontos része, képes alkalmazkodni a környezetbeli változásokhoz. A szabályozási folyamat – miközben egy világos helyiségből átmegyünk egy sötét helyiségbe – rendkívül egyszerű: az első pillanatokban szinte semmit nem látunk – hozzá kell szoknunk a sötétséghez – ami azt jelenti, hogy a kibernetikus rendszer, észlelve a változást, alkalmazkodik az új fényviszonyokhoz. Hasonló folyamat játszódik le akkor is, amikor a sötétből kimegyünk a napsütésre: eleinte a szivárványhártya túl sok fényt enged be, ami néhány másodpercre elvakítja az embert.

Pontosítsuk még egyszer a fenti példán keresztül azokat az alapfogalmakat, amelyeket a szabályozó rendszerrel kapcsolatban megismertünk. A szabályozási folyamat célja, hogy az optimális látást biztosítsa.

Az a fénymennyiség, amely ehhez szükséges, a mindenkori kívánt érték. A beállító az aktuális fénymennyiséget képes érzékelni és befolyásolni. Valahányszor változik a környezetbeli fényintenzitás, és a fénymennyiség eltér a kívánttól, a szivárványhártya a különbség kiegyenlítésére törekszik.

Azok számára, akik a biológiai példát túlságosan bonyolultnak ítélik meg, nézzünk még egy egyszerű példát a kibernetikus rendszerekre a mindennapi élet területéről, sőt az okos fürdőkád közvetlen szomszédságából: elemezzük hogyan működik a WC öblítőtartálya!

Az öblítőtartály esetében a beállítandó érték a tartályban található víz mennyisége. A zavaró tényező maga az öblítés – ekkor ugyanis a vízmennyiség a kívánt érték alá csökken. Ennek következtében működésbe lép a beállító egység (csap) és mindaddig ereszti a tartályba a vizet, amíg a vízmennyiség

pillanatnyi értéke el nem éri a kívánt értéket, azaz amíg a tartály meg nem telik. Ettől kezdve a rendszer állapota állandó – a következő öblítésig.

Reméljük, hogy a felsorolt egyszerű példákon keresztül sikerült érzékeltetni az Olvasóval, hogy mi is a kibernetika, mint tudományág tárgya. Kibernetikus rendszerekkel szinte minden más tudományágon belül – így a modern elektronikában is – találkozhatunk. A könyv további részeiben mi is megismerkedünk majd olyan kapcsolásokkal, amelyek már a kibernetika területére esnek. Ezek a kapcsolások a legfontosabb alapelemei az „intelligens" robotoknak, amelyek valóban képesek az „intelligencia" látszatát kelteni, hiszen képesek környezetükhöz alkalmazkodni.

De tegyünk ismét egy lépést előre!

3. ÖTLETEK – PROGRAMOK – KAPCSOLÁSOK

A rövid történeti áttekintés és a kibernetika alapfogalmainak tisztázása után ideje rátérni a robottechnika gyakorlati kérdéseire.

A kapcsolásokat olyan egyszerű elemekből építjük fel, amelyeket mindenki jól ismer, aki egy kicsit is foglalkozott elektronikával. Kipróbálásukhoz ugyancsak elegendő lesz egy egyszerű szimulációs modell. A szimulációs modellt a Commodore 64-es gép gazdag utasításkészletével kelthetjük életre. A robotépítésnek ez az a pontja, amely talán a legtöbb nehézséget jelenti majd a lelkes barkácsolók számára.

3.1 A motorvezérlés

A robot legfontosabb alkotórésze az elektromotor, amely képessé teszi a mozgásra.

A robottechnikában kétféle motortípust használnak: az egyik a hagyományos elektromotor, amely a legtöbb háztartási gépben megtalálható, a másik pedig az ún. léptető motor.

A hagyományos elektromotort motorvezérléssel tehetjük a robot alkalmas alkotórészévé.

A motorvezérlés két vezetéken keresztül befolyásolja a motor mozgását. Nyugalmi állapotban egyik vezetékben sem folyik áram. Ha az egyik vezetéket áram alá helyezzük, a motor a vezérlés iránya szerint balra vagy jobbra forog. Ha mindkét vezetéken áram folyik át, a motor ismét nyugalmi állapotba kerül.

A motorvezérlő alapkapcsolások két típusával ismerkedünk meg, melyek a fenti feltételeknek megfelelnek. Az első kapcsolás lényegi része egy relé, amely a motorok tulajdonképpeni kapcsolásáért felelős. Az ilyen típusú vezérlésnek vannak előnyei és hátrányai egyaránt: egyfelől a kapcsolási folyamatot vizuálisan követni tudjuk, és ugyanakkor meg tudjuk különböztetni egymástól a vezérlő és a kapcsoló áramkört, ami különösen fontos lehet nagy áramfelvételnél. Azonban a relés kapcsolás hátrányairól sem szabad megfeledkezni: a relék mint elektromechanikus elemek lényegesen több helyet foglalnak el, mint a hasonló célú tranzisztoros kapcsolások, és lassúbbak is.

A kapcsolások másik típusa tranzisztorokból építhető fel. A tranzisztoros kapcsolásoknál azonban gondosan kell ügyelni arra, hogy a tranzisztorok terhelhetőségét sehol se lépjük túl, hiszen egyébként azok tönkremennek.

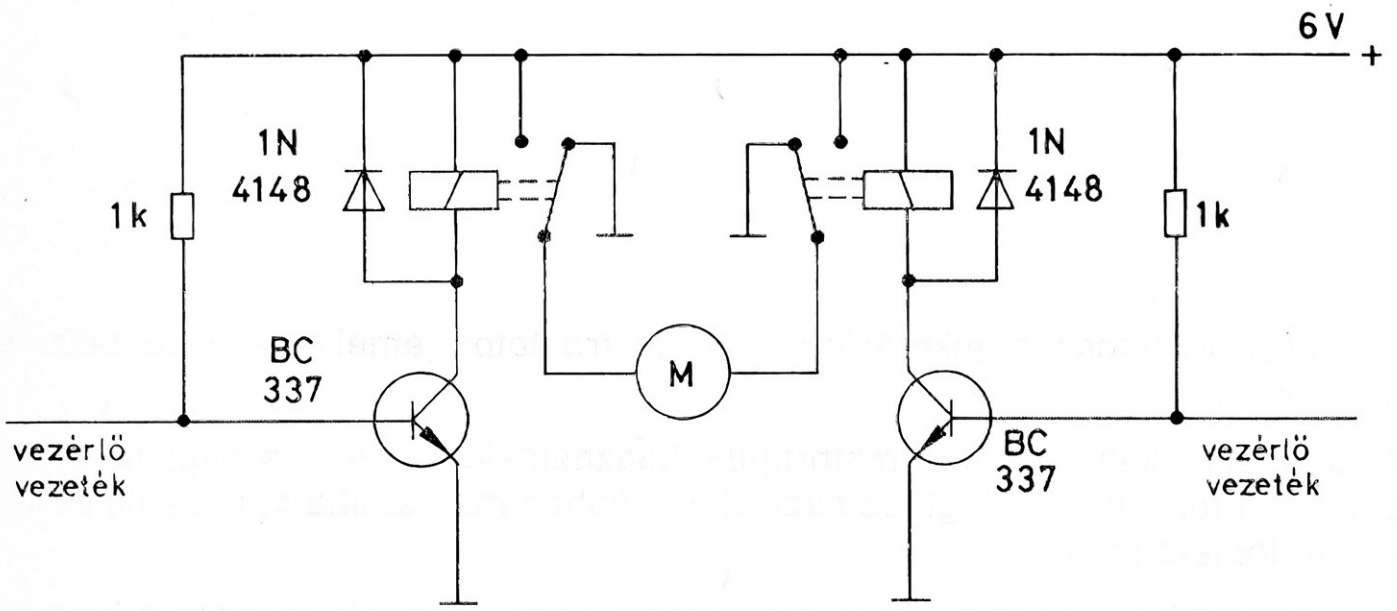
3.1.1 A RELÉS MOTORVEZÉRLÉS

A motorvezérlő kapcsolást két 6 V-os reléből és egy kapcsolóból építhetjük fel.

A relé nyugalmi érintkezője a földdel, munkaérintkezője pedig a pozitív tápfeszültséggel van összekötve.

A motor, amelyet a két relé közé kell bekötni, csak akkor működik, ha a két relé közül csak egy van meghúzott állapotban.

A relék vezérlését egy standard NPN tranzisztorral oldhatjuk meg.



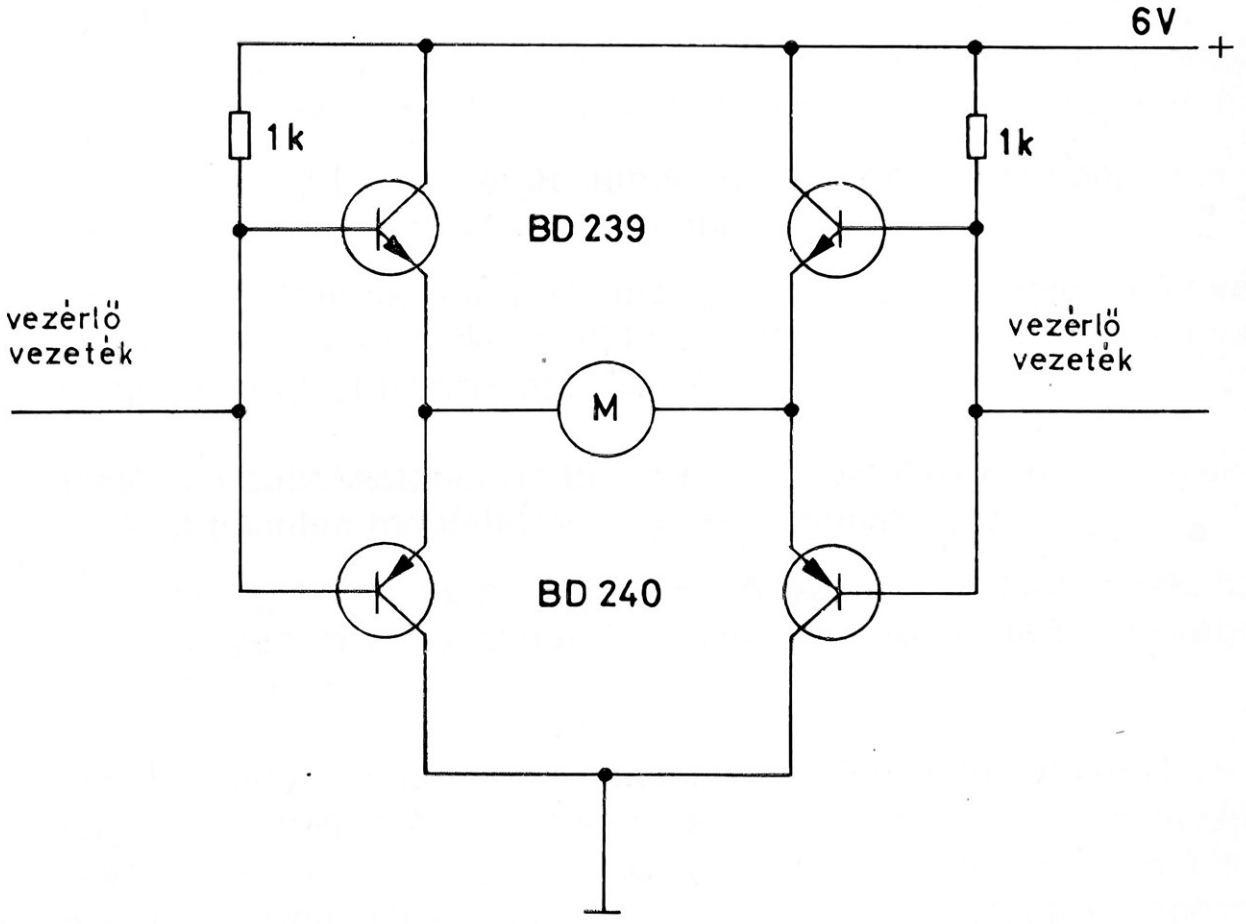
3. ábra. Relés motorvezérlés kapcsolási rajza

A kapcsolást kis ügyességgel megépíthetjük anélkül, hogy szerelőlapot használnánk a relék csatlakoztatásához. Ez a megoldás egyrészt munka- másrészt helymegtakarítással jár.

Amint azt a későbbiekben látni fogjuk a vezérlés megoldható egy egyszerű TTL integrált áramkörrel, ami ugyanakkor közvetlen csatlakozási lehetőséget is jelent a Commodore 64-eshez.

3.1.2 TRANZISZTOROS MOTORVEZÉRLÉS

Ez a kapcsolás kizárólag tranzisztorokból épül fel, mozgó alkatrészek nélkül, ezzel elkerülhetjük a biztosítékok alkalmazását is. Maga a vezérlés lényegesen kisebb méretű lesz mint az előző, relés megoldásnál. Kis helyigénye miatt különösen alkalmas arra, hogy egy robot építőeleme legyen. A robotépítésnél mindig az okozza a legnagyobb gondot, ha nincs már hely a bővítésre.



4. ábra. Tranzisztoros motorvezérlés kapcsolási rajza

A motorvezérlést két tranzisztorpárból álló kapcsolással valósítjuk meg. Ha az egyik tranzisztorpár *bázisát* vezérel, akkor a motor balra forog, ha a másikat, akkor pedig jobbra. Ha mind a négy tranzisztor bekapcsolt állapotban van, a motor éppúgy nyugalmi állapotban marad, mintha egyetlen tranzisztor sem kapott volna bázisfeszültséget. Az előzőhöz hasonlóan ezt a kapcsolást is vezérelhetjük TTL jelekkel.

Bár első látásra úgy tűnik, hogy a két kapcsolás cserélhető (ami nagyon fontos szerepet játszik a kapcsolások kiválasztásánál), egy apró különbségről sohasem szabad megfeledkezni.

Ha mindkét kapcsolást 6 V-os feszültséggel működtetjük, akkor a tranzisztoros motorvezérlésnél 4,6 V feszültséget kap a motor. A feszültségesés abból adó-

dik, hogy a tranzisztoros kapcsolásnál a motor nincs közvetlenül a tápfeszültségre kötve. A tranzisztoronkénti 0,7 V feszültségesés összesen 1,4 V esést eredményez. Nem kell tehát meglepődni azon, hogy a relés kapcsolás gyorsabban forgatja a motort mint a tranzisztoros.,

Ezt a tényt feltétlenül célszerű figyelembe venni, amikor a megfelelő motort kiválasztjuk.

3.2 A szimulációs modell

A sajátépítésű robot életrekeltéséig sok apró részletet kell átgondolnunk és kidolgoznunk.

A szükséges kapcsolások és programok megértését feltétlenül megkönnyíti, ha azokat először egy igen egyszerű modell elkészítésén keresztül mutatjuk be.

Építsünk először egy karosszériát, amelyet az előző fejezetben bemutatott elektromotorral mozgó járművé alakíthatunk.

Ezt a kis járművet szimulációs modellnek nevezzük. Az elegáns név valóban megilleti az egyszerű kis szerkezetet, hiszen működésével nagyrészt valóban szimulálhatjuk egy igazi házirobot működését.

A szimulációs modelleknek három típusát különböztetjük meg, amelyek mindegyike építését tekintve megfelel egy-egy robottípusnak is.

Ezek egymástól egy bizonyos ponton olyan lényegesen különböznek, hogy az alkotó igen könnyen el tudja dönteni, melyik típus felel meg leginkább az ő elképzeléseinek.

Mindhárom modellt két motorral hajtjuk meg, amelyek ellentétesen kapcsolhatóak, így a jármű képes lesz egy helyben is megfordulni. A motorokat két olyan vezérlőegység irányítja, amellyel az előző fejezetben megismerkedtünk. A dolog lényegét nem érinti, hogy a vezérlést relés- vagy tranzisztoros kapcsolással oldjuk meg. Mindenesetre első modellünk legyen tisztán mechanikus szerkezetű, beleértve a motor vezérlését is. Ami a modell méretét illeti, mindössze arra kell ügyelnünk, hogy belső terében elférjen egy pár elem vagy akkumulátor és néhány kis szerelőlap.

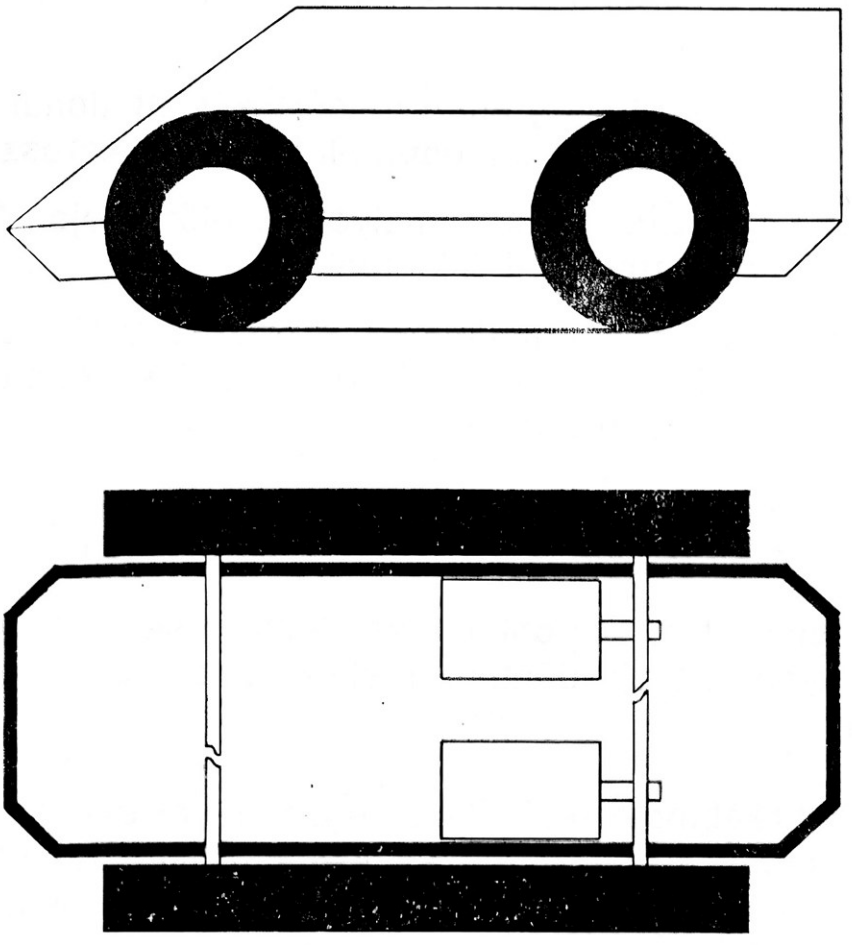
Azoknak, akiknek van otthon modellépítő barkácsdobozuk, nagyon könnyű dolguk lesz. Abban megtalálható minden eszköz még a vezérlőegység elkészítéséhez is.

3.2.1 I TÍPUSÚ SZIMULÁCIÓS MODELL

Az I típusú szimulációs modellünk egy egyszerű lánc-, ill. fogaskerék-meghajtású jármű, amelynek leglényegesebb része a hajtómű.

A két motort úgy kell egy rendszerbe kötni, hogy a jármű képes legyen kényelmes ütemben előrehaladni, ill. szükség szerint képes legyen üzem közben gyorsítani. A kerekeket a lánchoz, ill. a fogaskerékhez kell illeszteni úgy, hogy szabadon mozoghassanak. Ugyanis a jármű csak akkor tud helyben megfordulni, ha a két lánc szabadon és egymástól függetlenül mozgatható.

Végül a karosszériát bármilyen anyagból, fából, műanyagból elkészíthetjük.



5. ábra. I típusú szimulációs modell

Az I típusú modell igazi terepjáró.

Sajátos meghajtása, a lánc-, ill. fogaskerékmeghajtás ugyanis képessé teszi arra, hogy az akadályokat könnyedén leküzdje és az emelkedőkön is gond nélkül közlekedjen.

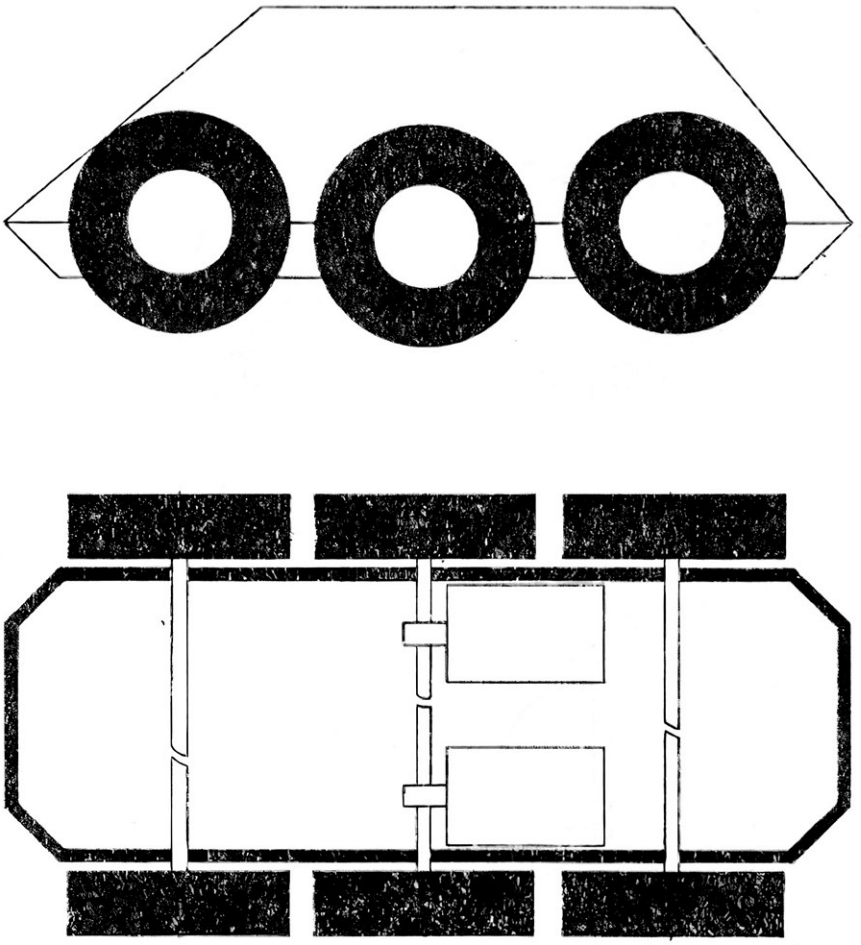
Persze mindez csak akkor valósulhat meg igazán, ha a kívánalmaknak a karosszéria is megfelel, azaz könnyű anyagból készül.

3.2.2 II TÍPUSÚ SZIMULÁCIÓS MODELL

A II típusú modell külsőleg nagyon hasonlít az I típusú modellre, technikailag azonban lényegesen eltér attól. Az I-es modell láncmeghajtását ennél a modellnél hat kerékkel felszerelt futóművel helyettesítettük. A hat kerék közül csak a középső kettőt hajtjuk meg. A kerekek súrlódás okozta kopását úgy kerülhetjük el, hogy a meghajtott kerekeket egy kicsit alacsonyabbra helyezzük mint a többit. Így azonban a talajjal csak a középső kerekek érintkeznek, tehát az egész szerkezetet ki kell egyensúlyozni, hogy a súlypont a középső kerékre kerüljön. Így haladás közben a súlypont helyzetének állandó változása miatt mindig pontosan négy kerék érintkezik a talajjal, ezen az apró trükkön múlik a jármű biztonságos mozgása.

Ez a modell már nem nevezhető jó terepjárónak, alkalmazkodó készségét azonban javíthatjuk, ha az első tengelyét úgy rögzítjük, hogy az a hosszanti tengelyhez képest kilenghessen.

A 6. ábrán látható az MB cég által gyártott kis játék meghajtó szerkezete. Pl. ez vagy egy hasonló kis játékszer kiváló alapja lehet épülő robotunknak.



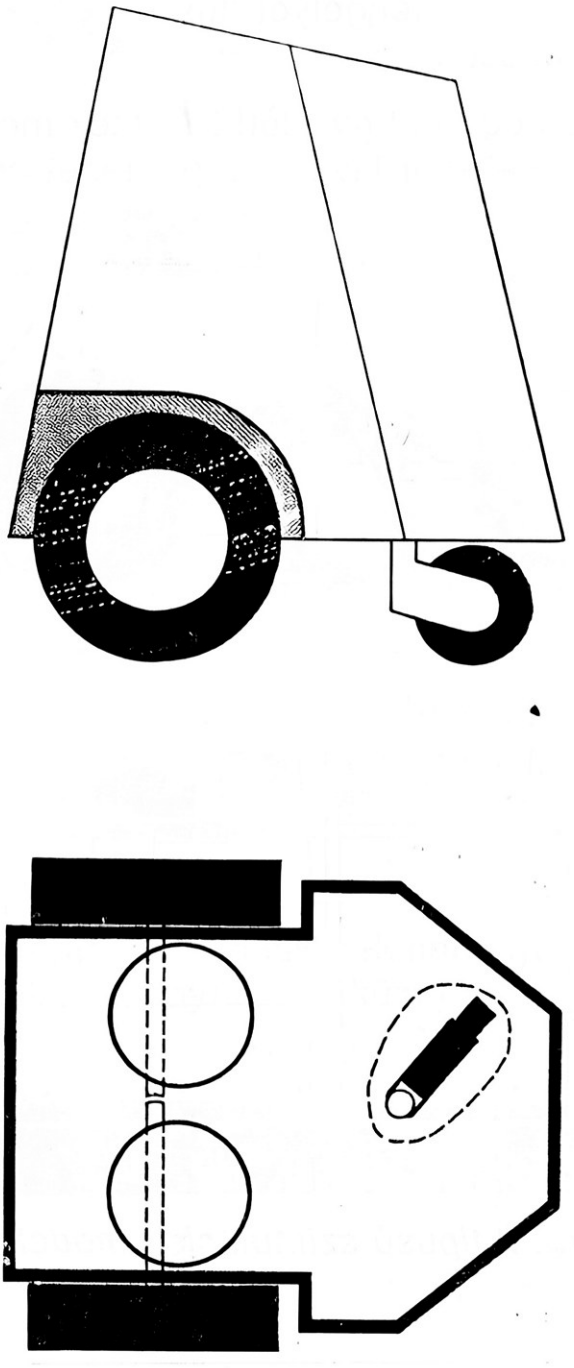
6. ábra. II típusú szimulációs modell

3.2.3 III TÍPUSÚ SZIMULÁCIÓS MODELL

Az eddig bemutatott modellek viszonylag alacsony építésű járművek voltak.

A III típus három kerékre támaszkodik, amelyek közül kettőt meghajtunk, a harmadik pedig támasztó kerékként szabadon fut. A támasztó kereket úgy kell elhelyezni, hogy ne érintkezzen a másik kettővel, de a lehető legközelebb legyen hozzájuk, és így biztosítsa a modell megfelelő egyensúlyi állapotát. A három kerék egymáshoz viszonyított helyzetét nagyon pontosan meg kell előre határoznunk különösen akkor, ha később szeretnénk a modellt egy igazi házirobothoz hozzáépíteni.

A 7. ábrán jól látható, hogy meghajtómotort az alaplapra merőlegesen kell beépíteni.



7. ábra. III típusú szimulációs modell

Hasonlóan fontos a fordulatszám pontos meghatározása, hiszen ez a modell nagyobb sebességnél könnyen elveszítheti az egyensúlyát.

Hogy végül a három modell közül melyiket választjuk, az csak egyéni ízlésünktől függ. A kapcsolások és programok, amelyekkel az Olvasót a későbbiekben megismertetjük mindhárom típusra egyaránt jól alkalmazhatóak. Ennek ellenére a választás nem lehet teljesen önkényes, hiszen egyáltalán nem mindegy, hogy mi a végső célunk az elkészített eszközzel.

Az is fontos szempont, hogy az eszközt később bővíteni lehessen, anélkül hogy az egészet át kellene építenünk. Ha valaki úgy fog hozzá a robotépítéshez, hogy a munkát nem tervezi meg kellő alapossággal, és a végső célt nem határozza meg kellő pontossággal, az eredménytelenség hamar megfoszthatja a kísérletező kedvétől.

3.3 Fénykapcsoló

A Commodore 64-es géppel megoldható vezérlési feladatokra első példaként egy fénykapcsolót szeretnénk bemutatni. A fénykapcsoló közbeiktatásával a Commodore 64-es gép alkalmassá tehető különböző külső kapcsolások vezérlésére, anélkül, hogy magát a számítógépet át kellene alakítani, vagy fizikailag csatlakozni kellene hozzá.

Az utóbbi feltehetően megnyugtatja azokat az Olvasóinkat, akik féltik számítógépüket a maguképítette szerkezetek romboló hatásától.

A fénykapcsoló a számítógéphez csatlakoztatott monitort vagy tv-t használja a kapcsolás, ill. a robot vezérlésére. A kapcsolatot a vezérlő és vezérelt egység között egy szenzor (érzékelő) teremti meg, amelyet egyszerű szigetelőszalaggal felragasztunk a képernyőre. A csatlakoztatás „bonyolult" mechanikája miatt természetesen erre a célra csak olyan tv-készülék felel meg, amelyen nincs mattüveg fedőlap. Egyébként ugyanis nagyon valószínű, hogy a kapcsolás vagy egyáltalán nem fog működni, vagy csak olyan precíz beállítás után, ami meghaladja az amatőr szintet.

Vegyük sorra a kapcsolás elkészítéséhez szükséges részeket. Kezdjük a szenzorral. Szenzorként LDR 05 típusú fényérzékeny ellenállást használhatunk, ami külső alakját tekintve is kifejezetten alkalmas erre a célra.

Az átlátszó műanyag házba épített LDR-t papírral vagy fóliával be kell borítani, hogy a fényhatások elől leárnyékoljuk, természetesen úgy, hogy a szenzor fényérzékeny oldala szabadon maradjon.

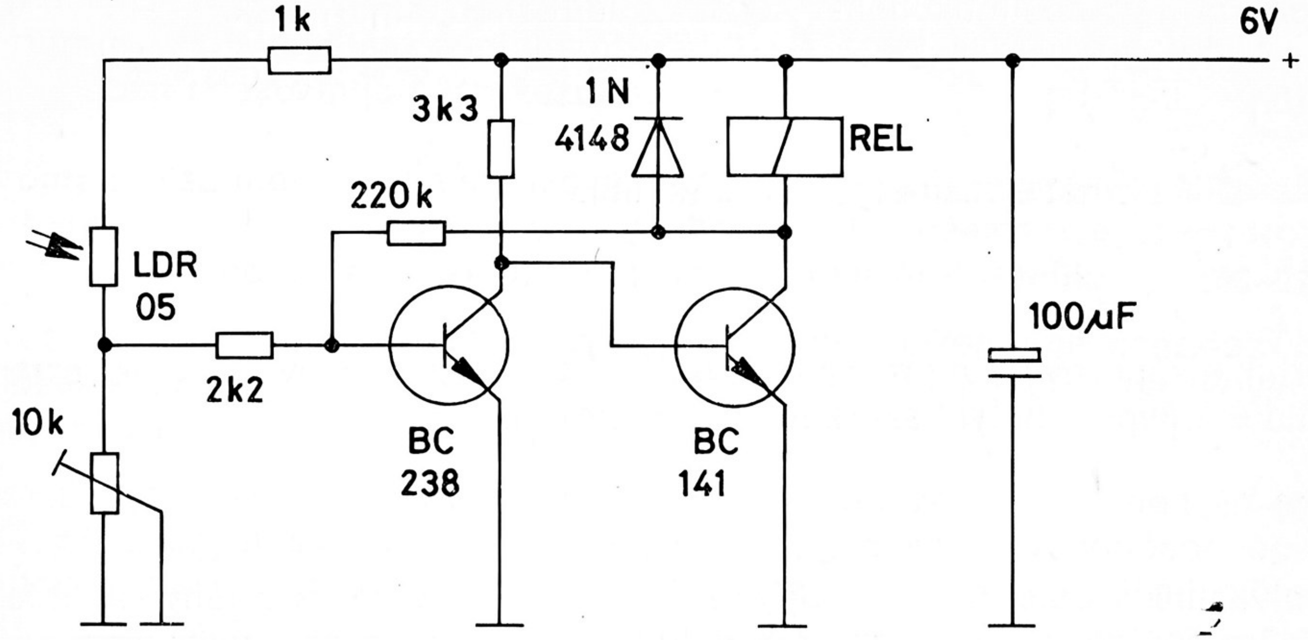
A vezérlő másik fontos része az elektronika.

Az elektronika gondoskodik arról, hogy a reléket a fény állapotváltozásainak megfelelően kapcsolgassa. A fény állapotváltozásait a Commodore 64-es gép váltja ki, és a szenzor érzékeli.

Az elektronika lényegi része két NPN tranzisztor, amelyek a tulajdonképpeni kapcsolásról gondoskodnak. A fényérzékenység beállítását a 10 kΩ-os beállító ellenállással végezhetjük. A relékhez bármilyen típusú (pl. 1N 4148) diódát használhatunk.

A kapcsolás megépítése rendkívül egyszerű, és anyagigénye is mindössze egy egyszerű raszterezett szerelőlap.

A mechanikus és elektronikus részek megtervezése után nem maradt más hátra mint a vezérlő program elkészítése.



8. ábra. Fénykapcsoló áramkör

3.3.1 EGYBITES FÉNYKAPCSOLÓ

Az „Egybites fénykapcsoló” egy kifejezetten egyszerű program, amellyel a fénykapcsoló működését szemléltethetjük. Tulajdonképpen a program a rendszer vezérlő egysége. Rendszer alatt persze nem kell jelen esetben összetett vezérlési rendszerre gondolni, itt mindössze egy elektromotor vagy egy villanykörte kapcsolásáról van szó.

A program indításkor feketére váltja a képernyőn a keret és háttér színét, hogy a fénykapcsoló vezérléséhez szükséges kontraszt elég éles legyen.

A bekeretezett fénypontot a programmal világosra, ill. sötétre válthatjuk, és ezt a képernyőre ragasztott fényérzékeny ellenállás közvetlenül érzékeli. A program az 1-es ill. 0-ás billentyű leütésének hatására a fénykapcsolót be-, ill. kikapcsolja.

```
5 REM *** P1 ***
10 POKE53280,0:POKE 53281,0
20 PRINT"████*** EGYBITES FENYKAPCSOLO ***███"
30 PRINT TAB(12);"BE/KI------->0/1"
35 PRINT TAB(23);"████████╭─╮"
36 PRINT TAB(13);"KAPCSOLO: | |":PRINT TAB(23);"╰─╯███";
40 GET E$:IF E$="0" THEN PRINT"█ ";:GOTO40
50 IF E$="1" THEN PRINT"█●";:GOTO40
60 GOTO40
```

A program pontos működését részletes leírásban ismertetjük.

10:

A keret- és a háttér színét feketére állítjuk.

20–36:

Kiírjuk a képernyőre a program nevét és a tájékoztató szöveget fehér színnel, majd a fénypont helyét szürkével bekeretezzük.

40–60:

Beolvasunk a billentyűzetről egy számértéket. Ha 1, akkor a fénypontot fehér színűre váltjuk (a szürke kereten belül), és mindaddig nem változtatjuk meg, amíg a billentyűzetről 0-át nem olvasunk. Előfordulhat, hogy az elkészült kapcsolás elsőre nem működik, pl. bizonytalanul vagy vibrálva kapcsol. Ebben az esetben az egész rendszert be kell szabályozni. A szabályozás jelentheti a fénypont sötét és világos színe közötti kontraszt, ill. a szerelőlapon a 10 kΩ-os potenciométer helyzetének megváltoztatását.

A külső egység vezérlésével csak akkor érdemes először kísérletezni, amikor a relés kapcsolás már biztonságosan, ingadozás és vibrálás nélkül működik.

3.3.2 NYOLCBITES FÉNYKAPCSOLÓ

Bár az „Egybites fénykapcsoló" c. program kiválóan alkalmas arra, hogy a fénykapcsoló működésének lényegét szemléltesse, komolyabb vezérlési feladatok megoldására, bonyolultabb szimulációs modellek vezérlésére valójában alkalmatlan. Az összetettebb vezérlési feladatoknak olyan fénykapcsolóra van szükség, amelynek több kimenete van. Ennek a megfontolásnak az eredményeként született meg a „Nyolcbites fénykapcsoló" program megírásának ötlete.

A nyolcbites fénykapcsoló nyolc, egymástól függetlenül programozható kimenetből áll. Természetesen nem feltétlenül kell mind a nyolc kimenetet minden vezérlési feladatban felhasználni, és így a programot egyszerűbb vezérlési feladatokban is használhatjuk.

A kapcsoló hardver része pontosan ugyanolyan mint az egybites változat hardver része, figyelembe véve természetesen azt, hogy most minden csatornához egy-egy kapcsolót kell építeni.

A nyolcbites fénykapcsoló programja jóval összetettebb, mint egybites őse. Ez a program többféle választási lehetőséget nyújt, amelyekről az induló képernyőmenüben tájékoztatja a gépkezelőt.

A program első menüpontja kirajzolja a képernyőre a fényérzékeny ellenállások helyét, pontosan megjelölve, hogy hová kell ragasztani a szenzorokat ahhoz, hogy mind a nyolc csatornát optimálisan vezérelhessük.

„A vezérlősor meghatározása" c. menüpont a tulajdonképpeni vezérlés programozása, amelyben rögzítjük, hogy mely szenzorok és mennyi ideig legyenek bekapcsolva.

A menüpont először a vezérlősornak megfelelő változótömb elemeit – 1-gyel tölti (ez jelzi ugyanis a beolvasás végét, ill. feldolgozáskor a vezérlősor végét). A feltöltés után következik az elemek beolvasása, amely két részből áll: először kiválasztjuk azokat a szenzorokat, amelyeket szeretnénk bekapcsolni, majd rögzítjük a bekapcsolt állapot időtartamát.

„A vezérlősor módosítása" menüpontban az előzőleg begépelt adatokat módosíthatjuk.

A vezérlősor meghatározása után meghívhatjuk „A vezérlés indítása" menüpontot, amelyben a program feldolgozza a begépelt sor tartalmát.

A feldolgozást csak a RUN/STOP billentyű leütésével lehet megszakítani. Ha megszakítás után a feldolgozást folytatni szeretnénk, a programot a GOTO 30 utasítással célszerű újraindítani. Ebben az esetben ugyanis a változók korábbi tartalma megmarad.

Az 5-ös és 6-os menüpontokkal a begépelt vezérlősort tárolhatjuk lemezen, ill. visszaolvashatjuk lemezről. Ezzel a lehetőséggel érdemes élni, hiszen így elkerülhetjük, hogy a vezérlősort a program indításakor mindig újra be kelljen gépelnünk.

```
5 REM *** P2 ***
10 POKE 53280,0:POKE 53281,0
15 DIM DZ(255),DL(255),DN$(15),DY(255)
30 PRINT"■■■■   ***   NYOLCBITES FENYKAPCSOLO   ***"
40 PRINT TAB(8);"■■■■A SZENZOROK KIRAJZOLASA     (1)
50 PRINTTAB(8);"■■A VEZERLOSOR MEGHATAROZASA  (2)"
60 PRINTTAB(8);"■■A VEZERLOSOR MODOSITASA     (3)"
70 PRINTTAB(8);"■■A VEZERLES INDITASA         (4)"
80 PRINTTAB(8);"■■A VEZERLOSOR TAROLASA       (5)"
90 PRINTTAB(8);"■■A VEZERLOSOR VISSZAOLVASASA (6)"
95 PRINTTAB(8);"■■A PROGRAM VEGE              (7)"
100 PRINTTAB(8);"■■■■*** KEREM VALASSZON! ***"
110 GET WA$:IF VAL(WA$)<1 OR VAL(WA$)>7 THEN GOTO110
120 ON VAL(WA$) GOSUB 200,300,400,500,600,700,800
```

```
130 GOTO30
200 PRINT"■■":PRINT TAB(7);"■*** A SZENZOROK ABRAZOLASA ***"
210 PRINT"■■■■■■■■■■■■■          7    6    5    4    3    2    1    0"
220 PRINT"■       ⌒    ⌒    ⌒    ⌒    ⌒    ⌒    ⌒    ⌒"
230 PRINT"        |●|  |●|  |●|  |●|  |●|  |●|  |●|  |●|"
240 PRINT"        ◡    ◡    ◡    ◡    ◡    ◡    ◡    ◡"
250 PRINT TAB(16);"□□□□□□□□□□□□□■KESZ(I/N)?"
260 GET E$:IF E$="" GOTO260
270 RETURN
300 PRINT"■■":PRINT TAB(4);"■*** A VEZERLOSOR MEGHATAROZASA ***"
310 FOR L=0 TO 255:DZ(L)=-1:NEXT L:I=0
320 PRINT TAB(17);"■■■■■■SORSZAM=";I
330 INPUT"■■■■ KEREK EGY DECIMALIS SZAMOT=";DZ(I)
340 IF DZ(I)<O OR DZ(I)>255 THEN RETURN
350 INPUT"■■ IDOTARTAM=";DL(I)
360 PRINT"□□□□□□□□□□□□□□□":I=I+1:GOTO320
400 PRINT"■■":PRINT TAB(9);"■*** A VEZERLOSOR MODOSITASA ***"
410 INPUT"■■■■■■SORSZAM=";I
420 PRINT TAB(10);"■■A REGI DEC.ERTEK=";DZ(I)
430 PRINT TAB(10); "A REGI IDOTARTAM=";DL(I)
440 PRINT TAB(12);"■■■MODOSITAS(I/N)"
445 GET E$:IF E$="" GOTO 445
450 IF E$<>"I" THEN RETURN
460 INPUT"■AZ UJ DEC.ERTEK=";DZ(I)
465 IF DZ(I)<0 OR DZ(I)>255 THEN PRINT"□":GOTO 460
470 INPUT"AZ UJ IDOTARTAM=";DL(I)
480 RETURN
500 PRINT"■■":PRINT TAB(9);"■*** A VEZERLOSOR INDITASA ***":I=0
505 DY(I)=DZ(I):IF DZ(I)<0 OR DZ(I)>255 THEN RETURN
510 FOR J=7 TO 0 STEP-1
520 IF DY(I)<2↑J THEN A$(J)=" "
530 IF DY(I)>=2↑J THEN A$(J)="□":DY(I)=DY(I)-2↑J
540 NEXT J
550 PRINT"■■■■■■■■■■■■■■■■ ";A$(7);"    ";A$(6);"    ";A$(5);"    ";
553 PRINTA$(4);"    ";
555 PRINTA$(3);"    ";A$(2);"    ";A$(1);"    ";A$(0);
    "□□□□□□□□□□□□□□□□"
560 FOR L=0 TO DL(I):NEXT L
570 I=I+1
580 GOTO 505
600 PRINT"■■":PRINT TAB(6);"■*** A VEZERLOSOR TAROLASA ***":I=0
610 INPUT "■■■■■■■■■■■■■■■■■■■■■■A FILE NEVE=";DN$
620 OPEN2,8,2,DN$+",S,W"
640 PRINT#2,DZ(I);CHR$(13);DL(I)
645 IF DZ(I)<0 OR DZ(I)>255 THEN CLOSE2:RETURN
650 I=I+1:GOTO 640
700 PRINT"■■":PRINT TAB(4);"■*** A VEZERLOSOR VISSZAOLVASASA
    ***":I=0
710 INPUT "■■■■■■■■■■■■■■■■■■■■■■A FILE NEVE=";DN$
```

```
720 OPEN 2,8,2,DN$+",S,R"
740 INPUT#2,DZ(I),DL(I)
745 IF DZ(I)<0 OR DZ(I)>255 THEN CLOSE 2:RETURN
750 I=I+1:GOTO740
800 PRINT"■■":PRINT TAB(8);"■*** A PROGRAM VEGE ***"
810 PRINT TAB(8);"■■■■■■■BIZTOSAN EZT KERI(I/N)"
820 GET E$:IF E$="" THEN 820
830 IF E$<>"I" THEN RETURN
840 SYS 64738
READY.
```

A program listáját igyekeztünk úgy felépíteni, hogy könnyen áttekinthető legyen. Az egyes lépések megértéséhez segítségül közöljük a program részletes leírását.

10:

A főmenü kiírása a képernyőre

110:

A választott menüponthoz tartozó számérték beolvasása a billentyűzetről.

120:

GOSUB utasítással elágazunk a kívánt programrészre.

130:

Visszatérés a főmenühöz a kiválasztott alprogram végrehajtása után.

200–270:

„A szenzorok ábrázolása" alprogram, amely kirajzolja a képernyőre a fényérzékeny ellenállások helyét. Az alprogramból tetszőleges billentyű leütésével kiléphetünk (260-as sor).

300–360:

„A vezérlősor meghatározása" programrészben a DZ változót feltöltjük −1-gyel (310-es sor).

A beolvasás közben az I futóindex értéke minden lépésben eggyel nő, és így minden lépésben a vezérlősor következő eleme kap értéket. A 330-as sorban a program egy 0 és 255 közé eső decimális szám begépelésére vár. A decimális szám bináris alakja pontosan megfelel egy vezérlősornak. A bináris bitmin-

tában nulla értéknek megfelelő fénypont nem világít, azaz a hozzárendelt kapcsoló „kikapcsolt" állapotba kerül. Így ha pl. a decimális szám 255, akkor 0-tól 7-ig minden bit értéke 1, tehát minden fénypont világít.

(A decimális/bináris átalakításhoz a 3.4 alfejezetben közlünk egy táblázatot).

Ha a begépelt szám kívül esik a 0–255 számtartományon, az alprogram automatikusan visszatér a főprogram végrehajtására.

A 350-es sorban a program bekéri az éppen begépelt vezérlőmintához tartozó időtartamot, azaz azt a számértéket, amely meghatározza, hogy a begépelt bitmintához tartozó állapotot a program mennyi ideig tartsa fenn.

400–480:

Ebben a programrészben a korábban begépelt vezérlősorok bármelyikét ismét feldolgozhatjuk vagy módosíthatjuk. A kívánt vezérlősort az index (l. 300–360) segítségével választhatjuk ki.

Az index begépelése után a program kijelzi a hozzá tartozó vezérlősort, amit „A vezérlősor módosítása" menüponttal igény szerint megváltoztathatunk. A programrészből a MODOSITAS (I/N) kérdésre adott N(em) válasszal léphetünk ki.

500–505:

„A vezérlősor indítása" című alprogram legfontosabb része a decimális szám binárissá alakítása.

A fénypontokat a program az egybites fénykapcsoló programjában megismert módon rajzolja ki. A program átszámítás előtt megvizsgálja, hogy a decimális szám a megengedett határok közé esik-e.

Ha nem, akkor az alprogram végrehajtása megszakad és visszatérünk a főprogramhoz. Egyébként átalakítás (510–540) és kijelzés következik. A kirajzolt fénypontok mindaddig a képernyőn maradnak, amíg le nem fut egy üres ciklus, amelynek végértéke a korábban megadott időtartam.

A ciklus lefutása után az index értéke eggyel nő, és a program feldolgozza a következő vezérlősort. A hibátlan futás érdekében az indexet az 500-as sorban nullára állítjuk.

600–650:

Ez a programrész az index nullázása után megkérdezi, hogy a vezérlősorokat milyen néven tárolja a lemezen. Válaszul természetesen olyan file-nevet kell megadni, amely még nem létezik a lemezen. Ha a nevet helyesen adtuk meg, a program az adott nevű soros file-ban tárolja az adatokat, majd lezárva a

megnyitott csatornát visszatér a főprogramhoz. Az adatok végét ismét egy a 0–255 számtartományon kívül eső érték, pl. −1 jelzi.

700–750:

Az alprogram a lemezen tárolt vezérlősorokat beolvassa a tárba. Működése hasonló a tárolást végző (600–650) alprogram működéséhez.

800–:

A program a biztonság kedvéért megkérdezi, hogy a futtatást valóban be akarjuk fejezni, vagy sem, és igenlő válasz esetén SYS 64738 utasítással alapállapotba hozza a számítógépet (RESET).

A program részletes leírása után még egy apróságra szeretnénk felhívni az Olvasó figyelmét.

Ahhoz, hogy a szimulációs modellt a nyolcbites fénykapcsolóval valóban vezérelni tudjuk, építenünk kell négy kapcsolóáramkört, amelyek a relékkel a szimulációs modellek motorjait közvetlenül vezérelhetik. A motorokat, a relés vezérléshez hasonlóan, a fénykapcsoló egyes kapcsolási helyzetének megfelelően kell a relékhez csatlakoztatni.

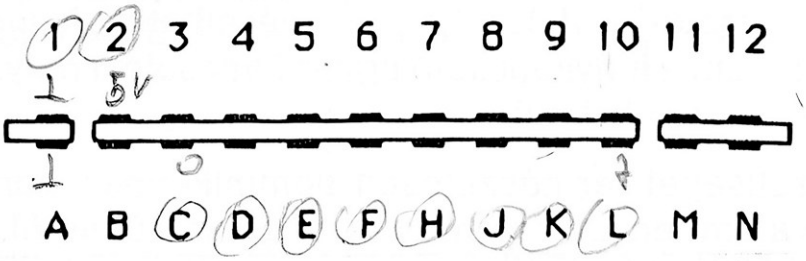
A szükséges feszültséget természetesen semmiképpen nem a Commodore 64-esről, hanem valami egyéb áramforrásról célszerű venni.

3.4 A USER PORT (felhasználó kapu)

A Commodore típusú gépeken megszokott user port a 64-esről sem hiányzik. A user port a külső egységek (pl. nyomtató) vezérlésére szolgáló, egyszerűen programozható kapu, amelyet minden nehézség nélkül felhasználhatunk a sajátépítésű kapcsolások vezérlésére is.

A user port a Commodore 64-es alapgép hátoldalán található, 24 pólusú csatlakozó 3,96 mm-es rasztertávolsággal. Az illesztéshez csak boltban vásárolt szabályos csatlakozót szabad használni, nehogy megsértsük a számítógép alapkártyáján lévő érintkezőket.

A 9. ábra a user portot ábrázolja, ahogy az a Commodore 64 hátoldalán látható.



9. ábra. USER PORT

1	*GND*	*2*	*+5 VOLT*	*3*	*RESET*
4	*CNT 1*	*5*	*SP 1*	*6*	*CNT 2*
7	*SP 2*	*8*	*PC 2*	*9*	*ATN IN*
10	*9 VOLT AC*	*11*	*9 VOLT AC*	*12*	*GND*

A	*GND*	*B*	*FLAG 2*	*C*	*PB 0*
D	*PB 1*	*E*	*PB 2*	*F*	*PB 3*
H	*PB 4*	*J*	*PB 5*	*K*	*PB 6*
L	*PB 7*	*N*	*PA 2*	*N*	*GND*

Mivel kapcsolásainkban csak az 1,2 és C–L lábakat fogjuk használni, a többi ismertetésére nem térünk ki. Az 1-es láb (GND) a számítógép földvezetéke, amelyet a géphez kapcsolt készülék földvezetékéhez kell csatlakoztatni. Természetesen erre a célra bármely másik GND jelű láb is alkalmas.

A 2-es lábon +5 V TTL tápfeszültséget találunk. A számítógépet túlterhelés ellen úgy védhetjük, hogy a lábat 100 mA-nél magasabb árammal nem terhel-

jük. Az 1-es és 2-es, ill. a 10-es és 11-es lábak közötti kivágás megakadályozza, hogy a csatlakozót fordítva dugjuk be és ezzel a user portot megsértsük. Általában a csatlakozókra ugyanezeken a helyeken kis darab műanyagot ragasztanak a fordítva feldugás elleni védelmül.

Bennünket elsősorban a C PB 0-tól L PB 7-ig terjedő lábak érdekelnek, ezeket fogjuk ugyanis használni vezérlési feladataink megoldásában.

Ezek a lábak a Commodore 64-es alkatrészei közül az egyik legfontosabbhoz, a CIA 6526-os chiphez vezetnek. Ez az alapelem kétszer nyolc input/output vezetékkel rendelkezik. A 0-tól 7-ig terjedő port bitek (C–L) adják a második nyolc input/output bitet. Ismerkedjünk meg e második nyolc bit szerepével.

3.4.1 AZ ADATIRÁNY- ÉS ADATREGISZTER

A CIA 6526-os elem második nyolc adatvezetékének programozásához némileg meg kell ismerkednünk a chip belső felépítésével. A CIA egyebek között tartalmaz két nyolcbites regisztert, az ún. adatirány- és adatregisztert.

Az adatirányregiszter tartalma határozza meg, hogy a nyolc bit közül melyek a bejövő és melyek a kimenő adatok.

Az adatirány regiszter rögzített értéke mellett a CIA adatregiszterének megfelelő vonalain megkezdődik az adatok olvasása, ill. írása.

Hogyan programozhatók ezek a regiszterek, és hogyan használhatók vezérlési feladatok megoldására?

Az adatirány regiszter

Az adatirányt mindig azonos módon rögzíthetjük. Ebből a célból be kell írni a regiszter címére egy 0 és 255 közé eső decimális számértéket. Ha a regiszter tartalma nulla, mind a nyolc adatvonal bemenetként, míg a 255-ös számérték mellett minden vonal kimenetként működik.

A 0 és 255 közé eső további számértékek az általuk képviselt bitminta szerint osztják fel a nyolc vonalat beolvasó és kiíró vonalakra.

Adatirányregiszter

1	1	1	1	0	0	0	0

= 240

Ha a bitmintában 1-es áll, a sorban ennek a bitnek megfelelő adatvonal kimenet, egyébként bemenet. Az előző példában az adatregiszter első négy bitjét bemenetre, utolsó négy bitjét kimenetre állítottuk.

Az adatirányregiszter tartalmát ugyan decimális alakban kell megadni, de ahhoz, hogy az adatátvitelről képet kapjunk, ezt a számot binárissá kell alakítani.

Az unalmas és fáradságos átalakítástól megment bennünket a következő táblázat.

Tízes	Kettes	Tízes	Kettes
0	00000000	33	00100001
1	00000001	34	00100010
2	00000010	35	00100011
3	00000011	36	00100100
4	00000100	37	00100101
5	00000101	38	00100110
6	00000110	39	00100111
7	00000111	40	00101000
8	00001000	41	00101001
9	00001001	42	00101010
10	00001010	43	00101011
11	00001011	44	00101100
12	00001100	45	00101101
13	00001101	46	00101110
14	00001110	47	00101111
15	00001111	48	00110000
16	00010000	49	00110001
17	00010001	50	00110010
18	00010010	51	00110011
19	00010011	52	00110100
20	00010100	53	00110101
21	00010101	54	00110110
22	00010110	55	00110111
23	00010111	56	00111000
24	00011000	57	00111001
25	00011001	58	00111010
26	00011010	59	00111011
27	00011011	60	00111100
28	00011100	61	00111101
29	00011101	62	00111110
30	00011110	63	00111111
31	00011111	64	01000000
32	00100000	65	01000001

Tízes	Kettes	Tízes	Kettes
66	01000010	105	01101001
67	01000011	106	01101010
68	01000100	107	01101011
69	01000101	108	01101100
70	01000110	109	01101101
71	01000111	110	01101110
72	01001000	111	01101111
73	01001001	112	01110000
74	01001010	113	01110001
75	01001011	114	01110010
76	01001100	115	01110011
77	01001101	116	01110100
78	01001110	117	01110101
79	01001111	118	01110110
80	01010000	119	01110111
81	01010001	120	01111000
82	01010010	121	01111001
83	01010011	122	01111010
84	01010100	123	01111011
85	01010101	124	01111100
86	01010110	125	01111101
87	01010111	126	01111110
88	01011000	127	01111111
89	01011001	128	10000000
90	01011010	129	10000001
91	01011011	130	10000010
92	01011100	131	10000011
93	01011101	132	10000100
94	01011110	133	10000101
95	01011111	134	10000110
96	01100000	135	10000111
97	01100001	136	10001000
98	01100010	137	10001001
99	01100011	138	10001010
100	01100100	139	10001011
101	01100101	140	10001100
102	01100110	141	10001101
103	01100111	142	10001110
104	01101000	143	10001111

Tízes	Kettes
144	10010000
145	10010001
146	10010010
147	10010011
148	10010100
149	10010101
150	10010110
151	10010111
152	10011000
153	10011001
154	10011010
155	10011011
156	10011100
157	10011101
158	10011110
159	10011111
160	10100000
161	10100001
162	10100010
163	10100011
164	10100100
165	10100101
166	10100110
167	10100111
168	10101000
169	10101001
170	10101010
171	10101011
172	10101100
173	10101101
174	10101110
175	10101111
176	10110000
177	10110001
178	10110010
179	10110011
180	10110100
181	10110101
182	10110110

Tízes	Kettes
183	10110111
184	10111000
185	10111001
186	10111010
187	10111011
188	10111100
189	10111101
190	10111110
191	10111111
192	11000000
193	11000001
194	11000010
195	11000011
196	11000100
197	11000101
198	11000110
199	11000111
200	11001000
201	11001001
202	11001010
203	11001011
204	11001100
205	11001101
206	11001110
207	11001111
208	11010000
209	11010001
210	11010010
211	11010011
212	11010100
213	11010101
214	11010110
215	11010111
216	11011000
217	11011001
218	11011010
219	11011011
220	11011100
221	11011101

Tízes	Kettes	Tízes	Kettes
222	11011110	239	11101111
223	11011111	240	11110000
224	11100000	241	11110001
225	11100001	242	11110010
226	11100010	243	11110011
227	11100011	244	11110100
228	11100100	245	11110101
229	11100101	246	11110110
230	11100110	247	11110111
231	11100111	248	11111000
232	11101000	249	11111001
233	11101001	250	11111010
234	11101010	251	11111011
235	11101011	252	11111100
236	11101100	253	11111101
237	11101101	254	11111110
238	11101110	255	11111111

Eddigi tanulmányaink alapján akár hozzá is foghatunk ismereteink hasznosításához.

Térjünk vissza egy pillanatra utolsó példánkhoz, amelyben a user port adatregiszterének első négy bitjét kimenetre, második négy bitjét pedig bemenetre állítottuk. A megfelelő decimális érték 240 volt. Ezt az értéket a következő utasítással írhatjuk be a CIA 6526-os adatirányregiszterébe:

```
POKE 56579,240
```

Az utasítás hatására az adatáramlás iránya mindaddig a megadott értelmű lesz, amíg újabb utasítással a regiszter tartalmát meg nem változtatjuk, ill. a gépet ki nem kapcsoljuk.

Az adatregiszter

Ha a user portot kimenetként használjuk, akkor az adatregiszter programozása hasonló az adatirányregiszter programozásához. Ha az adatregiszter valamely bitje nulla, akkor a hozzárendelt port-vonal LOW (alacsony), egyébként HIGH (magas) állapotba kerül. A hozzárendelés nagyon egyszerű: a user port PB 0 jelű lábához tartozik az adatirány-, ill. adatregiszter legelső bitje stb.

adatirányregiszter								adatregiszter							
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1

A regiszterek tartalmát a következő utasításokkal állíthatjuk a kívánt értékre:

```
POKE 56579,255

POKE 56577,1
```

A port összes vonalát kimenetre kapcsoltuk, a PB 0-ás lábat pedig magasra állítottuk.

Hasonlóan olvashatjuk le a user port tartalmát bevitel esetén, azzal a különbséggel, hogy az adatregiszterbe ebben az esetben nem írunk, hanem olvasunk belőle:

```
POKE 56579,0

PRINT PEEK (56577)
```

Minden vonalat bemenetre programoztunk és a hozzárendelt regiszter tartalmát kiírjuk a képernyőre. Eddigi feladatainkban a port-vonalak mindegyikét bemenetre vagy kimenetre programoztuk.

Most nézzünk egy példát az írás-olvasás kevert alkalmazására.

Állítsuk az első négy bitet bemenetre, a második négy bitet kimenetre:

```
POKE 56579,240
```

Ezzel az utasítással az adatáramlás irányát rögzítettük. Ahhoz azonban, hogy valóban csak az első négy bit értékét olvassuk be, a második négy bit értékét törölni kell:

```
PRINT (255-(PEEK(56577)OR 240))
```

Az utasítás eredményeként egy 0 és 15 közé eső számot kapunk, amelyet binárissá alakítva láthatóvá válik az egyes bitek pillanatnyi szerepe.

A

```
POKE 56577,32
```

utasítással a PB 5 jelű lábat magasra, a felső byte összes többi bitjét pedig alacsonyra állítjuk. Reméljük a leírtakat sikerült minden Olvasónak megértenie. A biztonságos alkalmazást azonban célszerű némi gyakorlással megerősíteni.

3.5 A meghajtó fokozat beépítése

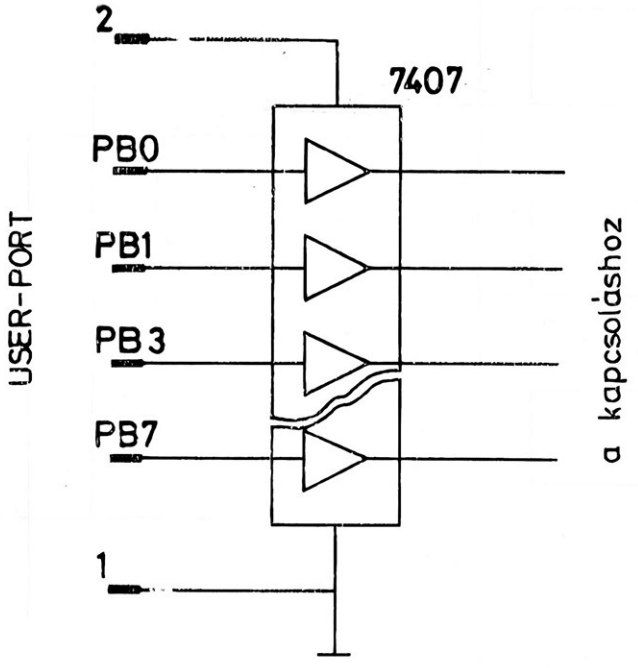
Mielőtt az első usert port alkalmazásával készült vezérlést kipróbálnánk, kénytelenek vagyunk egy közbenső kapcsolást építeni.

A CIA 6526-os chip ugyanis maximálisan 3,5 mA-rel terhelhető, ami viszont egyetlen kapcsolás vezérléséhez sem elegendő. Ahhoz, hogy a vezérlést is meg tudjuk oldani és ugyanakkor a számítógépet se tegyük tönkre, be kell iktatnunk egy meghajtó fokozatot, amely a kapott áramerősségnél nagyobbat szolgáltat, és ugyanakkor nem terheli túl a CIA 6526-os chipet.

Erre a célra kiválóan alkalmas egy TTL kapu IC, amely tápfeszültségét közvetlenül a Commodore 64-estől kaphatja. Különösen megfelelő pl. a 7407-es típusú elem, ez ugyanis hat egymástól független kaput tartalmaz.

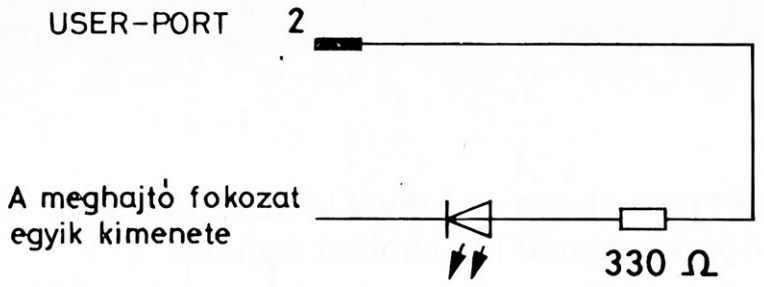
A 7407-es elem földjét a Commodore 64-es földjével (a user port 1-es lába) a pozitív tápfeszültséget pedig a user port 2-es lábával kell összekötni.

Ugyanakkor a user port kimeneteit össze kell kötni a közbenső meghajtó áramkör bemeneteivel. Ha a user port összes lábát kimenetként használjuk, akkor még egy 7407-es IC tokot kell a leírt módon közbeiktatni.



10. ábra. A meghajtó fokozat bekötése

A user port programozásának első kísérleteként kössünk egy fénydiódát megfelelő előtét ellenállással közvetlenül a meghajtófokozatra:



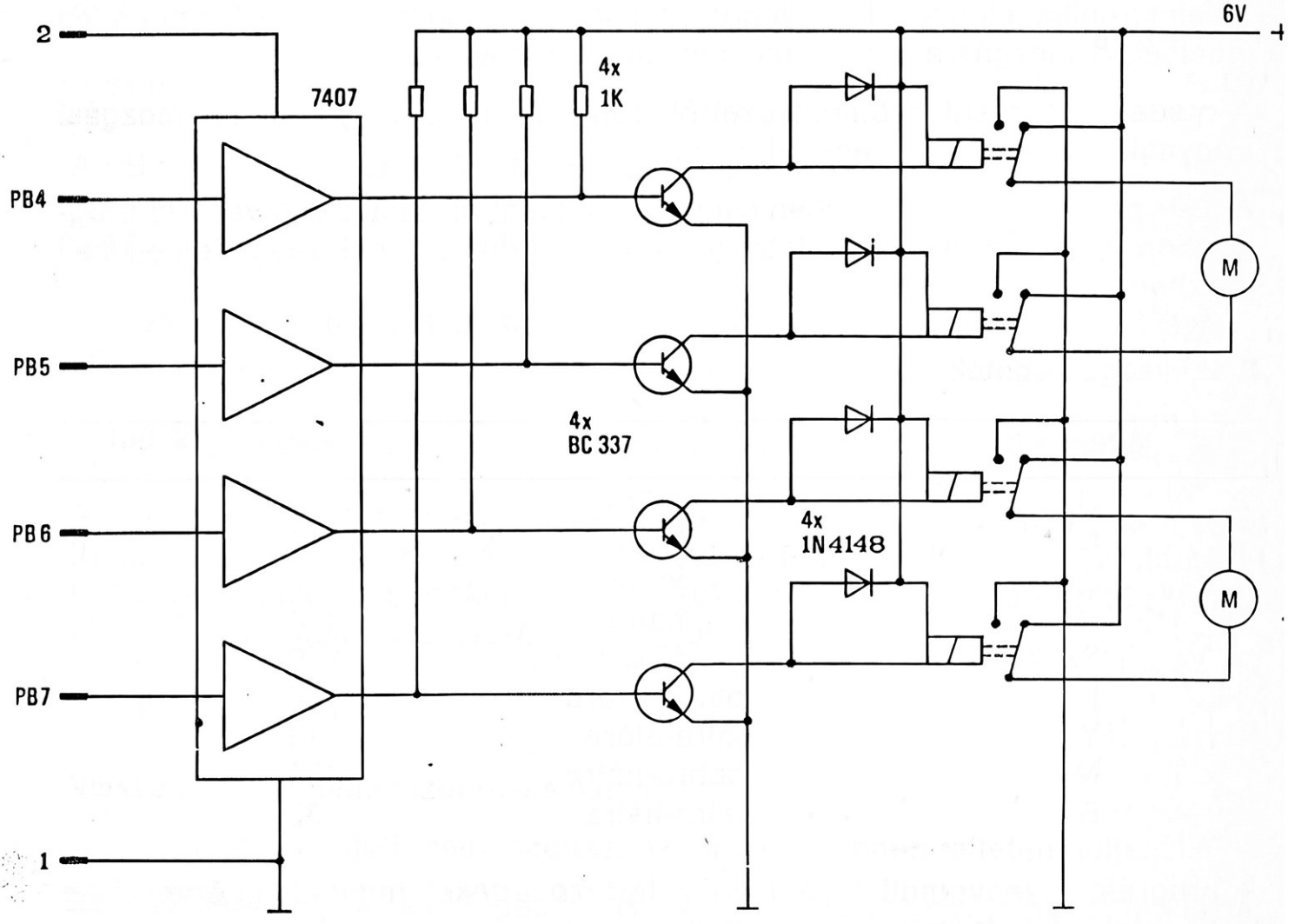
11. ábra. A meghajtó fokozat ellenőrzése fénydiódával

Mivel a kapcsolás áramfelvétele kevesebb 100 mA-nál, a Commodore 64-est biztosan nem károsítja.

3.6 A szimulációs modellek vezérlése

Az előző fejezetek áttanulmányozása után minden alapvető ismeretanyag az Olvasó birtokában van, amelyre a robot építése közben szüksége lehet. Kétféle motorvezérlés, háromféle szimulációs modell, a user port programozása, a meghajtó fokozat elkészítése, egyszóval valóban minden készen áll az igazi munkához. A készítendő modell vezérlésére a Commodore 64-es gép billentyűzetét fogjuk használni. Az előzőek alapján első feladatunk a vezérlő fokozat megépítése, amely a két módszer valamelyikével a szimulációs modellt fogja vezérelni. A modell tervét a következő oldalakon ismertetjük.

Még egyszer nyomatékosan felhívjuk az Olvasó figyelmét arra, hogy 5 V-nál nagyobb feszültséggel semmiképpen ne dolgozzon, az ugyanis a Commodore 64-re jutva tönkretenné a gép elektronikáját!!



12. ábra. Vezérlés kapcsolási rajza

A kapcsolást a PB 4-től a PB 7-ig terjedő lábakkal fogjuk vezérelni. Ezeket a lábakat, mint az már ismert, a meghajtó fokozattal kell összekapcsolni. Az első két kimenetet az első motorfokozathoz, míg a második kettőt a második motorfokozathoz kötjük. A 12. ábrán a relés motorvezérlés kapcsolási rajzát közöljük. Ezt természetesen bárki gond nélkül helyettesítheti tranzisztoros vezérléssel. Végül a modell motorjait összekapcsoljuk a motorvezérléssel. A modell – legalábbis ami az elektronikus részét illeti – máris működőképes.

Egyetlen dologra azonban még ki kell térnünk, nevezetesen az áramellátásra. A rajzon látható, hogy a kapcsolást egy 6 V-os külső áramforrás látja el. Ez a 6 V nem áll rendelkezésünkre a Commodore 64-ben. Áramforrásként hálózati stabilizátort használhatunk, de megfelel néhány szárazelem vagy egy akkumulátor is.

Természetesen az áramforrást ki kell kapcsolni, amikor az elkészült kapcsolást a számítógéphez illesztjük. Nagyon fontos, hogy eközben az összes földpotenciált összekössük (a C 64-es, a kapcsolás-, ill. a külső áramforrás földpotenciálját).

Ha mindennel készen vagyunk annak rendje s módja szerint, nekiláthatunk a vezérlő program megírásának.

Az elektronika felépítéséből adódóan a szimulációs modell képes lesz minden lehetséges mozgásra, amit a mechanikája lehetővé tesz.

A modell mozgását a billentyűzetről irányítjuk úgy, hogy minden mozgási iránynak megfeleltetünk egy billentyűt.

A user portot ebben az esetben csak kimenetként használjuk. A vezérlő programban az adatirány-, ill. adatregiszterek tartalmát ennek megfelelően kell rögzíteni.

A szükséges adatok:

Billentyű	Mozgásirány	Adatregiszter
U	előre	80
N	hátra	160
J	jobbra	96
H	balra	144
Szóköz	STOP	0
I	jobbra-előre	64
Y	balra-előre	16
M	jobbra-hátra	128
B	balra-hátra	32

A program a lenyomott billentyűhöz tartozó értéket mindaddig érvényben tartja, amíg új billentyűt le nem ütünk.

```
5 REM *** P3 ***
10 PRINT"[CLR]      **********************************"
20 PRINT"       ***          VEZERLES       ***"
30 PRINT"       *** A USER PORTON KERESZTUL***"
40 PRINT"       **********************************[DOWN]"
50 PRINT"       A PORTBITEK KIOSZTASA  :[M]"
100 PRINT"          PB 0-TOL  PB 7-IG    KIMENET"
110 POKE 56579,255
120 GET E$:IF E$="" THEN 120
130 IF E$="U" THEN POKE 56577,64+16:REM  ELORE
140 IF E$="N" THEN POKE 56577,128+32:REM HATRA
150 IF E$="J" THEN POKE 56577,64 +32:REM JOBBRA
160 IF E$="H" THEN POKE 56577,128 +16:REM BALRA
170 IF E$=" " THEN POKE 56577,0+0:REM STOP
180 IF E$="I" THEN POKE 56577,64+0:REM JOBBRA-ELORE
190 IF E$="Y" THEN POKE 56577,0+16:REM BALRA ELORE
200 IF E$="M" THEN POKE 56577,128+0 :REM JOBBRA HATRA
210 IF E$="B" THEN POKE 56577,0+32:REM BALRA HATRA
220 GOTO120
```

Bár a program annyira egyszerű, hogy listája önmagáért beszél, most is mellékeljük a részletes magyarázatot.

110:

A PB 0-tól PB 7-ig terjedő lábakra rögzítjük az adatirányt.

120:

Beolvassuk a lenyomott billentyűt. Ha nincs lenyomva egyetlen billentyű sem, várakozunk ezen a soron.

130–210:

Azonosítjuk a lenyomott billentyűt és az eredménytől függően más-más értéket írunk az adatregiszterbe. A programrészben felhasználtuk az előző táblázat tartalmát. A beírandó értéket úgy bontottuk két részre, ahogyan az a két (jobb, ill. bal oldali) motor vezérlését befolyásolja.

220:

Visszatérés a billentyűzet olvasásához.

A szimulációs modell megmozdításának immár minden feltétele teljesül.

Az első kísérletek, az esetleges hibák remélhetőleg számos hasznos tapasztalatot hoznak majd a következő feladatok megoldásához.

3.7 A szimulációs modellek kényelmes vezérlése

Az előző programot könnyen továbbfejleszthetjük a kapcsolás átalakítása nélkül úgy, hogy a vezérlés sokkal kényelmesebb legyen.

A továbblépés lényege az, hogy egyetlen betű beolvasása (begépelése) helyett, a program egy betűsorozatot olvas be, majd feldolgozásukat egymás után folyamatosan végzi el.

A vezérlőbetűk most is ugyanazok, mint amelyek az előző programban voltak. Egyetlen betűvel használunk többet; az A betű, mint vezérlőbetű, a mozgatás, ill. betűsor végét fogja jelezni.

A program egy sorozatban maximálisan 200 betűt fogad el. A begépelt betűk mindvégig láthatóak a képernyőn.

A továbbfejlesztett program listája:

```
1 REM *** P4 ***
5 DIM E$(199)
10 PRINT"■■■■      ****************************"
20 PRINT"       ***      KENYELMES        ***"
30 PRINT"       ***        VEZERLES       ***"
40 PRINT"       ***     A USER PORTON     ***"
50 PRINT"       ****************************■■"
60 PRINT"         A PORTBITEK KIOSZTASA: "
100 PRINT"       PB 0-TOL PB 7-IG KIMENET "
110 POKE 56579,255:POKE56577,0
120 PRINT"■      A VEZERLOSOR BEOLVASASA ■"
130 FOR I=0 TO199
140 GET E$(I):IF E$(I)="" GOTO140
145 PRINTE$(I);" ";
150 IF E$(I)="A" THEN 170
160 NEXT I
170 GET D$: IF D$="" THEN 170
180 FOR I=0 TO 198
190 GOSUB 230
200 FOR J=1 TO 500:NEXT J
210 NEXT I
220 POKE 56577,0:END
230 E$(I)="U" THEN POKE 56577,64+16:REM ELORE
240 E$(I)="N" THEN POKE 56577,128+32:REM HATRA
250 E$(I)="J" THEN POKE 56577, 64+32:REM JOBBRA
260 E$(I)="H" THEN POKE 56577,128+16:REM BALRA
```

```
270 E$(I)=" " THEN POKE 56577,  0+ 0:REM STOP
280 E$(I)="I" THEN POKE 56577, 64+ 0:REM JOBBRA-ELORE
290 E$(I)="Y" THEN POKE 56577,  0+16:REM BALRA-ELORE
300 E$(I)="M" THEN POKE 56577,128+ 0:REM JOBBRA-HATRA
310 E$(I)="B" THEN POKE 56577,  0+32:REM BALRA-HATRA
320 E$(I)="A" THEN POKE 56577,  0:END:REM MEGSZAKITAS
330 RETURN
```

Részletes leírás:

5:

Helyet foglalunk a lehetséges 200 betű számára. Azért választottuk ezt az értéket, mert így a vezérlő betűsorozat még éppen elfér a képernyőn.

110:

Rögzítjük az adatirányregiszter értékét és a user port kimeneteit alacsonyra állítjuk.

130–160:

Egy FOR-TO-NEXT ciklusban beolvassuk a vezérlő betűsorozatot. A beolvasás vagy a 200. betű beolvasása után, vagy az A betű leütésével ér véget.

A gép vár egy billentyű leütésére, majd megkezdi a feldolgozást.

180–210:

A feldolgozást tulajdonképpen egy alprogram végzi el. A 200-as sor határozza meg az adott irányú mozgás időtartamát. Az időtartamot az üres ciklus végértékének átjavításával megváltoztathatjuk.

220:

Az összes vezérlő betű végrehajtása után a program futása befejeződik úgy, hogy előtte a user port kimeneteit ismét alszintre állítja.

230:

Az előző programból már jól ismert feldolgozó programrész, kiegészítve az A betű vizsgálatával. Az A betű a vezérlő betűsor végét, azaz a feldolgozás befejezését jelzi.

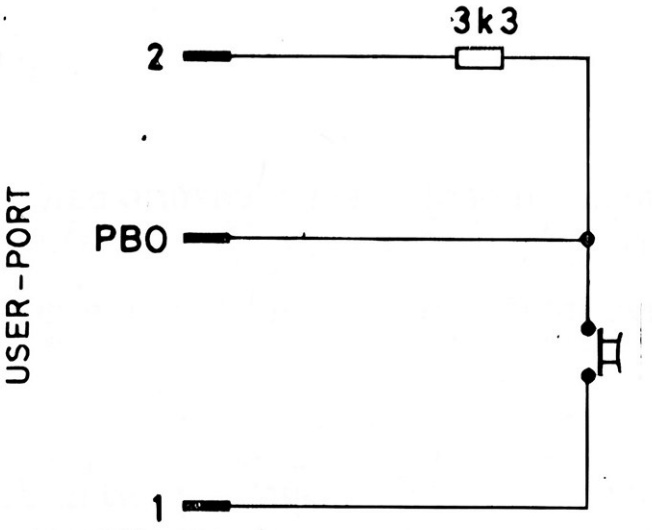
3.8 Adatbevitel a user porton keresztül

A user porton keresztüli adatbevitel programozásával már foglalkoztunk, de eddigi feladatainkban csak kivitelre volt szükségünk.

Következő példánkban az adatbevitel programozását szeretnénk szemléltetni.

Az adatbevitelhez a legegyszerűbb esetben egy billentyűs kapcsolóra (nyomógombra) van szükségünk, amelyet a külső kapcsolásba beépítve a user portra csatlakoztathatunk.

A user port adatregiszterének tartalmát leolvasva megállapíthatjuk, hogy a billentyű le van nyomva vagy nem. A billentyű csatlakozását a 13. ábra mutatja.



13. ábra. Billentyű csatlakoztatása a USER PORT-ra

Ha nem egy, hanem mondjuk nyolc billentyűt szeretnénk a 64-eshez csatlakoztatni, a 13. ábra kapcsolását egyszerűen nyolcszor meg kell ismételni, és egyenként a nyolc lábra kötni.

A mi kapcsolásainkban négynél több billentyűt nem fogunk használni.

Idézzük fel az adatbevitel programozására vonatkozó ismereteinket: első lépésként az adatirány-regiszter minden bitjét nullára kell állítanunk. Ha ez megtörtént, az egyes bitekhez rendelt lábakon megkezdődik az adatbevitel. Az adatregiszter tartalmát leolvasva egy 0 és 255 közé eső decimális számot kapunk, amelyet bináris számmá alakítva megállapíthatjuk, hogy a port mely lábai voltak egy, ill. nulla értékűek.

A következő kis program leolvassa a PB 0 láb értékét, és abból megállapítja, hogy a kapcsolásba épített billentyű le volt nyomva vagy sem.

```
5 REM. *** P5 ***
10 REM A PB 0-HOZ KAPCSOLT BILLENTYU LEOLVASASA
20 POKE 56579,0
30 E=PEEK (56577)
40 IF E=254 THEN PRINT"A BILLENTYU LE VAN NYOMVA!":GOTO30
50 PRINT"A BILLENTYU NINCS LENYOMVA!"
60 GOTO 30
READY.
```

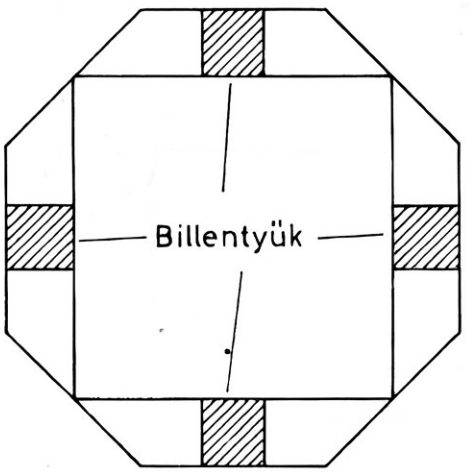
A program először bemenetre állítja a PB 0-tól PB 7-ig terjedő lábakat, majd leolvassa az adatregiszter tartalmát. Ha az adatregiszter tartalma 255, akkor a billentyű nincs lenyomva, azonban ha 254 akkor a PB 0-ra kötött billentyű le van nyomva. Az adatregiszter leolvasását a program mindaddig folytatja, amíg meg nem nyomjuk a RUN/STOP billentyűt.

3.9 Szimulációs modell reflexszel

Eddig olyan szimulációs modellekkel foglalkoztunk, amelyeket „előprogramozással" kellett vezérelni. Most megmutatjuk, hogy hogyan lehet a szimulációs modellt négy billentyű, négy ellenállás segítségével „önállósítani".

Ez a modell ideális esetben órákig képes egy zárt térben mozogni, anélkül, hogy egy sarokban elakadna.

A modellt négy billentyűvel kell felszerelni, mivel négy oldalról szeretnénk megvédeni a modellt az ütközéstől. Ahhoz, hogy a teljes frontfelületen keletkező ütközést észleljük, a billentyűket be kell építeni egy-egy lökhárító mögé. A lökhárítók elhelyezésével elérjük, hogy ha az ütközés oldalirányú, a billentyűk akkor is betöltik a szerepüket. A magyarázatnál többet árul el elképzelésünkről a 14. ábra:



14. ábra. A billentyűk elrendezése

Hogyan lesz a modellnek reflexe? Az alapelv nagyon egyszerű. A modell mindaddig halad egyenesen előre, amíg valamivel össze nem ütközik. Az ütközésről a billentyűkön keresztül értesül a számítógép, és a modellt kitérő manőverre készteti, hogy előtte meghatározza a kitérés lehetséges irányát.

Mechanikai és elektronikai szempontból nem támaszt különösen magas követelményeket a feladat megoldása.

A user port 0-tól 3-ig terjedő lábait a négy billentyűvel, míg a 4-től 7-ig terjedő lábakat a meghajtó fokozattal kell összekötni.

A legnehezebb feladatot talán az akadály kikerülésének programozása jelenti.

A program megírása előtt gondoljuk végig a reflexképesség logikáját. Képzeljük magunk elé azt a helyzetet, amikor a modell egyenesen előre haladva egy falba ütközik. Milyen kitérő manővert célszerű ilyenkor végrehajtani?

A hátrafelé haladás semmiképpen nem megoldás, hiszen egy bizonyos távolság megtétele után a modell ismét egyenesen előre szeretne haladni, és így előbb-utóbb a falba ütközne, a program pedig végtelen ciklusba esne. Arra azonban mindenképpen szükség van, hogy a jármű elszakadjon a faltól, vagyis egy kicsit hátrafelé mozduljon. Célszerű tehát a jobbra-hátra vagy balra-hátra irányt választani. Hogy a kettő közül melyik mellett döntünk, az teljesen mindegy, a lényeget nem érinti. A programba tizenhatféle lehetséges kitérő manővert iktattunk be, azaz minden elméletileg lehetséges irányt. Hogy melyek ezek, és milyen ütközések váltják ki, azt a következőkben összefoglaltuk:

Ütközés	**Reakció**
nincs	előre
előre	balra-hátra
hátra	előre
előre és hátra	nyugalmi állapot
jobbra	hátra-balra
előre és jobbra	hátra-balra
hátra és jobbra	előre-balra
előre, hátra és jobbra	nyugalmi állapot
balra	hátra-jobbra
előre és balra	hátra-jobbra
hátra és balra	előre-jobbra
előre, hátra és balra	nyugalmi állapot
jobbra és balra	hátra
előre, jobbra és balra	hátra
hátra, jobbra és balra	előre
előre, hátra, jobbra és balra	nyugalmi állapot

A táblázatból látható, hogy modellünket felkészítettük olyan helyzetekre is, amelyekből nincs kiút – ha pl. egyszerre több ellentétes irányba indulva akadályba ütközik – ilyenkor az egyetlen megoldás a helybenmaradás.

A dolog lényegét már áttekintettük, a program megírása előtt azonban még mindig érdemes néhány apróságon elgondolkodni.

Hogy a reflexszerű reakció hatékony legyen, az észlelt ütközés típusát tárolni kell, hogy a hozzá tartozó kitérő manővert megismételhessük. Ha ezt nem tennénk meg, és a kitérő manővert csak annyi ideig hajtanánk végre ameddig az ütközés fennáll, a jármű minden ütközés után egy helyben „remegni" kezdene. A remegés állandósulna, hiszen a modell alighogy elmozdul a faltól, ismét előre szeretne haladni, ami újra meg újra ütközést eredményez.

Ha az ütközés típusát tároljuk, a hozzá tartozó reakciót akkor is megismételtethetjük a járművel, amikor már nincs az útjában az akadály. Ha a kitérő manőver végrehajtása közben csak a tárolt típusú ütközés lép fel ismételten, vagy egyáltalán nincs ütközés, akkor a manővert egy idő elteltével befejezzük. Ezt az időtartamot a jármű mechanikus adottságai, felépítése alapján kell megállapítani. Egy másik típusú ütközés észlelésekor az előző reakciót befejezve áttérünk az új típushoz tartozó kitérő manőver végrehajtására. Ha programírás közben a leírtakat mind megszívleljük, szimulációs modellünk soha sem kerülhet kilátástalan helyzetbe – vagy ha mégis, megáll és türelmesen várja a kiszabadítást.

Lássuk a minden eshetőségre felkészített reflexprogramot:

```
5 REM *** P6 ***
10 PRINT"■■■       ****************************"
20 PRINT"        ***    REFLEX PROGRAM     ***"
30 PRINT"        ***     A USER PORTON     ***"
40 PRINT"        ****************************■■■"
50 PRINT"        A PORTBITEK KIOSZTASA:■"
100 PRINT"        PB 0-3:BEMENET-PB 4-7:KIMENET"
110 POKE 56579,240
120 EG=255-(PEEK(56577)OR 240)
130 GOSUB 200
140 FOR I=1 TO 30
150 ET=255-(PEEK(56577)OR 240)
160 IF(ET<>EG AND ET<> 0) THEN 120
170 NEXT I
190 GOTO120
200 REM KODTABLAZAT
210 IF EG= 0 THEN POKE 56577,64+16:REM ELORE
220 IF EG= 1 THEN POKE 56577, 0+32:REM BALRA-HATRA
230 IF EG= 2 THEN POKE 56577,64+16:REM ELORE
240 IF EG= 3 THEN POKE 56577, 0+0 :REM NYUGALMI ALLAPOT
250 IF EG= 4 THEN POKE 56577, 0+32:REM BALRA-HATRA
260 IF EG= 5 THEN POKE 56577, 0+32:REM BALRA-HATRA
270 IF EG= 6 THEN POKE 56577, 0+16:REM BALRA-ELORE
280 IF EG= 7 THEN POKE 56577, 0+ 0:REM NYUGA:LMI ALLAPOT
290 IF EG= 8 THEN POKE 56577,128+ 0:REM JOBBRA-HATRA
300 IF EG= 9 THEN POKE 56577,128+ 0:REM JOBBRA-HATRA
310 IF EG=10 THEN POKE 56577, 64+ 0:REM JOBBRA-ELORE
320 IF EG=11 THEN POKE 56577,  0+ 0:REM NYUGALMI ALLAPOT
330 IF EG=12 THEN POKE 56577,128+32:REM HATRA
340 IF EG=13 THEN POKE 56577,128+32:REM HATRA
350 IF EG=14 THEN POKE 56577, 64+16:REM ELORE
360 IF EG=15 THEN POKE 56577,  0+ 0:REM NYUGALMI ALLAPOT
380 RETURN
READY.
```

110:

Az adatirányregiszterbe 240-et írunk, így a PB 0-tól PB 3-ig terjedő lábakat bemenetre, a PB 4-től PB 7-ig terjedő lábakat pedig kimenetre állítjuk.

120:

Megvizsgáljuk, hogy az első négy lábra kapcsolt billentyűk valamelyike le van-e nyomva (volt-e ütközés). Ehhez leolvassuk az adatregiszter tartalmát, egy logikai OR művelettel leválasztjuk a felesleges biteket és az eredményt kivonjuk 255-ből.

A maradék 0 és 15 közé eső számból megállapíthatjuk a billentyűk állapotát.

130:

Ugrunk a kitérőmanővert végrehajtó programrészre.

140–170:

Ez a FOR-TO-NEXT ciklus határozza meg a kitérőmanőver időtartamát.

A 150-es sorban ismét leolvassuk a billentyűk állapotát. A 160-as sorban a kitérőmanőver megszakítási feltételét vizsgáljuk. Ha a billentyűk vizsgálata során új típusú ütközést észlelünk az előző manővert abbahagyjuk. Az ellenőrzéshez összehasonlítjuk a billentyűk előző és mostani állapotához tartozó értékeket.

A ciklusváltozó végértéket (a programban 30) a szimulációs modell szerkezetétől függően kísérleti úton kell megállapítani. Az ökölszabály: ha a jármű egy akadállyal frontálisan ütközik, a „jobbra-hátra" manőver az eredeti irányhoz képest 90°-os elforgatást kell hogy eredményezzen.

190:

Ugrás a billentyűk leolvasására.

200–300:

Ez a programrész nem egyéb mint a korábban bemutatott táblázat programozott változata.

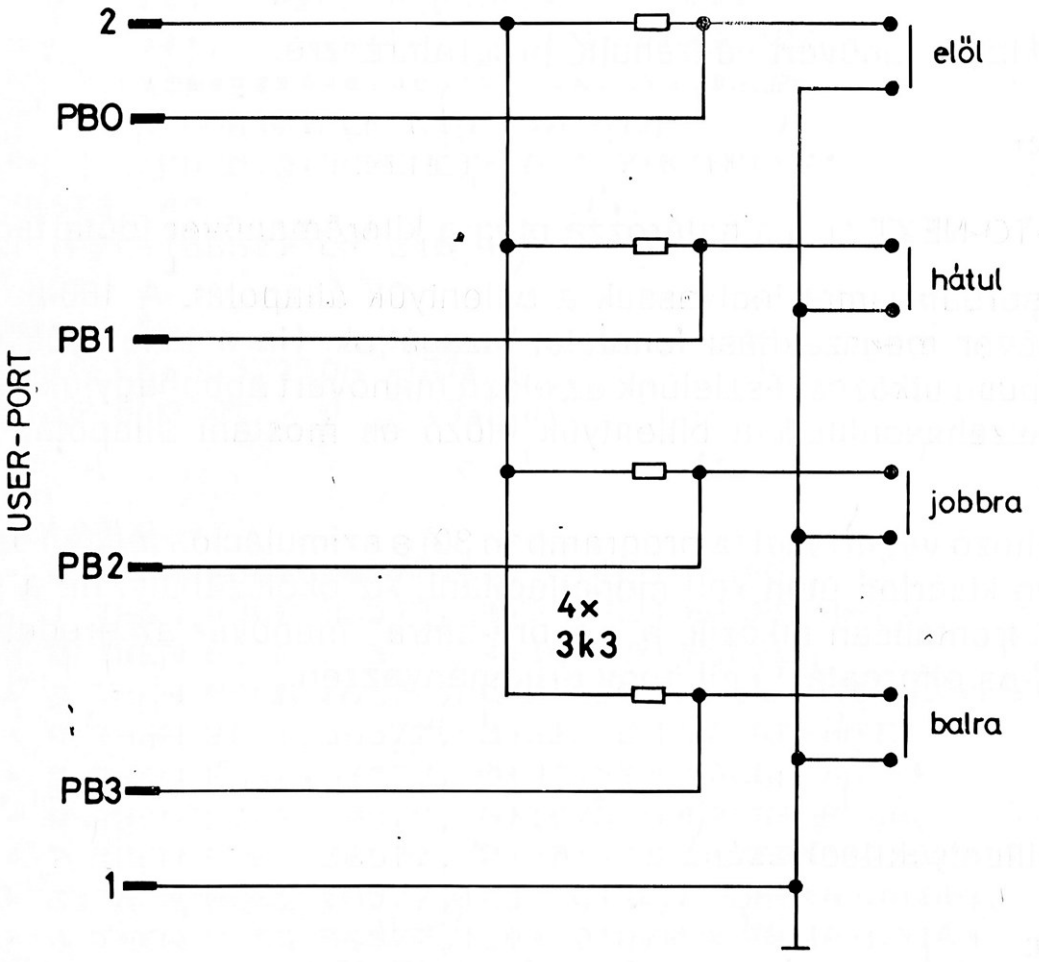
Az alprogram kiválasztja a billentyűk állapotának megfelelő manővert és aszerint mozgatja a meghajtókerekeket. Így a jármű mindig az adott helyzethez alkalmazkodva folytatja mozgását.

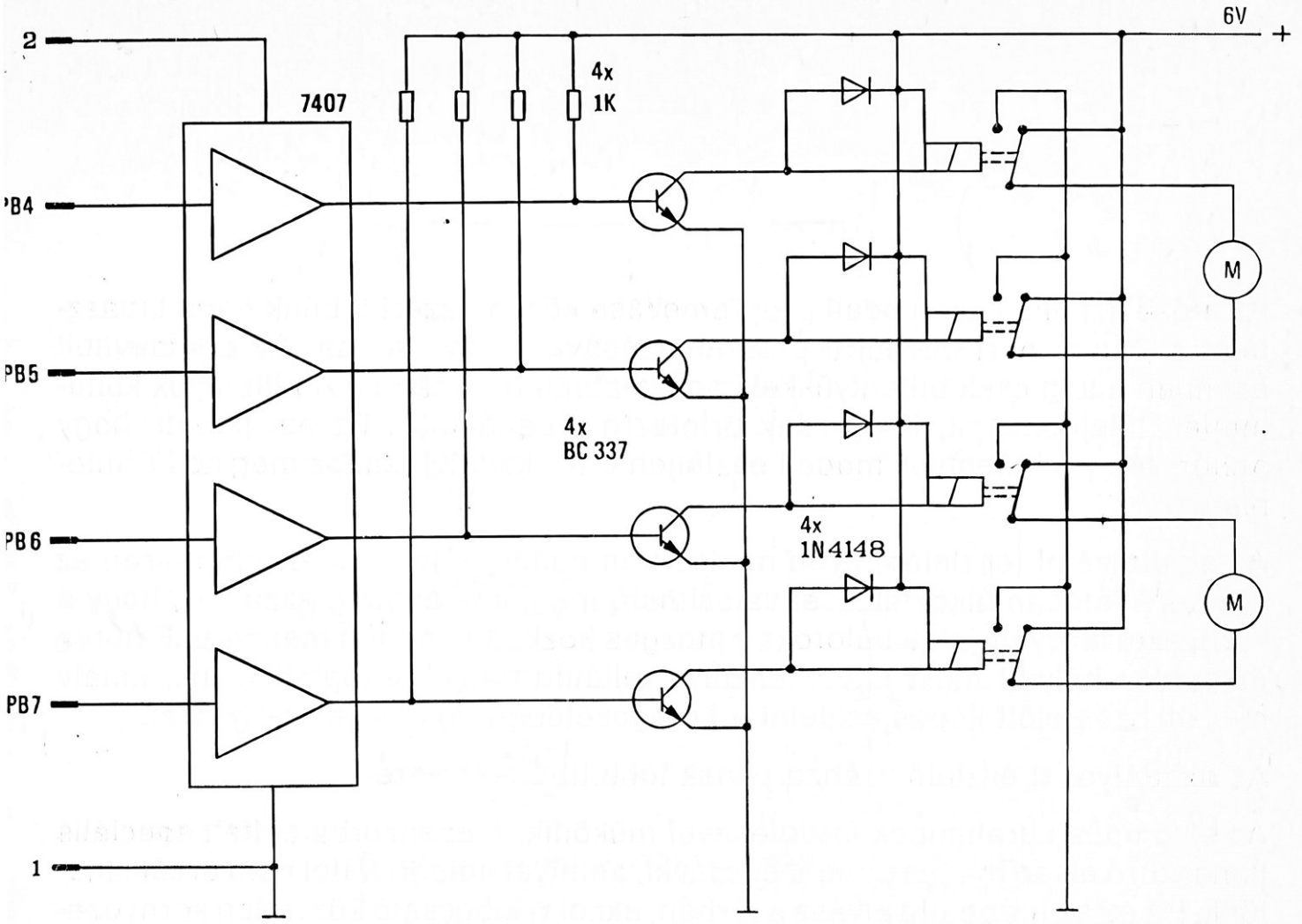
A program elemzése után térjünk vissza még egy rövid ideig a modell elektronikájához.

A teljes elektronika egymástól jól elkülöníthető egységekből épül fel, amelyekkel tulajdonképpen már egyenként megismerkedtünk. A modell vezérlése például pontosan követi azt a kapcsolást, amelyet a szimulációs modellek vezérlésének leírásánál ismertettünk.

Az adatbevitelt négy adatbeviteli egységből építettük fel, amit már korábban úgyszintén bemutattunk. A kiviteli lábak a 4-től 7-ig, a beviteli lábak a 0-tól 3-ig terjedő lábak.

Hasznosabb a további magyarázatnál a teljes szimulációs modell kapcsolásának figyelmes áttekintése.





15. ábra. Reflexszel rendelkező szimulációs modell rajza

Magától értetődik, hogy a relés motorkapcsolást most is helyettesíthetjük tranzisztoros kapcsolással. A két kapcsolás helyettesíthetőségéről már többször beszéltünk.

Ha valaki a modell különböző helyzetekhez való alkalmazkodásával mégis elégedetlen lenne, a vezérlő program módosításával befolyásolhatja a modell viselkedését, anélkül, hogy az elektronikát módosítania kellene.

3.10 Az infravörös szenzor

Az előző szimulációs modell programozása közben szert tettünk némi tapasztalatra a user port bevitelre programozásával kapcsolatban. Az adatbevitelt azonban eddig csak billentyűkkel kapcsolatban használtuk. A billentyűk kellemetlen tulajdonsága, hogy csak érintésre „reagálnak". Ez azt jelenti, hogy ahhoz, hogy a billentyűs modell észleljen egy akadályt, ahhoz meg kell érintenie azt.

Az adatfelvétel (észlelés) ilyen módja nem mindig a legkedvezőbb, hiszen az érintés általában ütköztetéssel valósítható meg, ami esetleg azzal jár, hogy a környezet tárgyait (pl. a bútorokat) mozgás közben a modell megsérti. E durva megoldás helyett most egy szenzorral ellátott modellel foglalkozunk, amely már ütközés előtt képes észlelni a környezetében elhelyezett tárgyakat.

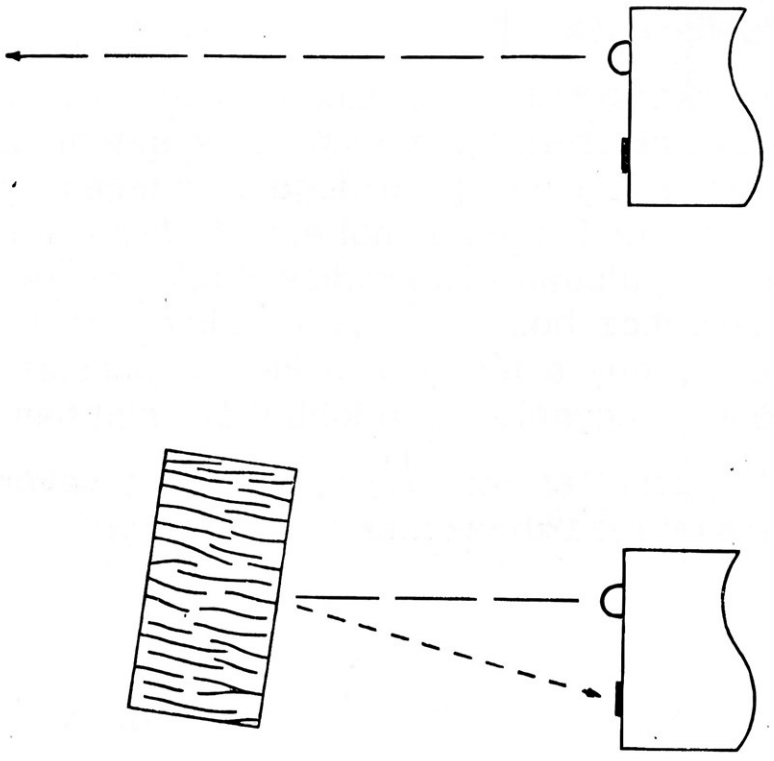
Az akadályokat észlelő szenzoroknak több típusa ismeretes.

Az első típus ultrahangok észlelésével működik. A szenzorba épített speciális hangszóró olyan magas hangot bocsát ki, amelyet emberi füllel nem érzékelhetünk. Ha ez az ultrahang elvész a térben, akkor a kibocsátó közvetlen környezetében nincs egyetlen tárgy sem. Az ultrahangszerzor másik alkotórésze egy mikrofonból és egy hangfrekvenciás kapcsolófokozatból áll. Az utóbbit úgy kell tervezni, hogy csak az adó frekvenciájára kapcsoljon. Az adót és vevőt közvetlenül egymás mellett kell elhelyezni, de úgy, hogy egymástól akusztikusan el legyenek szigetelve, azaz a testhang semmiképpen ne eredményezhessen átkapcsolást. Ez azt jelenti, hogy az adó által kibocsátott hang nem szabad, hogy közvetlenül – mielőtt az akadályt elérte volna – átkapcsolásra késztesse a vevőt. Így a szenzor az akadályt valóban az általa visszavert hanghullámok alapján azonosítja.

Az ultrahangszenzor óriási előnye, hogy több méter távolságból képes észlelni az akadályokat. Azonban ha valóban megpróbáljuk megépíteni, látni fogjuk, hogy az adó és vevő akusztikai elszigetelése egészen komoly gondot okoz. Éppen ezért, hogy az Olvasó választhasson, bemutatunk egy másik szenzortípust, amely infravörös fény segítségével „tájékozódik".

A szenzor legfontosabb alkotórészei most is az adó és a vevő, amelyeket egymáshoz nagyon közel, de optikailag jól árnyékolva kell elhelyezni. Tegyük fel, hogy az adó – esetünkben egy infravörös fénydióda – kibocsát egy fénysugarat. Ha a fénysugár elvész a térben és nem tér vissza a vevőhöz, akkor a szenzor környezetében nincs akadály. Ellenkező esetben a környezeti tárgy visszaveri a kibocsátott sugarat, és a szenzorba épített vevő ezt felfogva

tudomást szerez a tárgy jelenlétéről. Az észlelési távolság több tényezőtől függ. A legfontosabb tényező a kibocsátott fénysugár intenzitása, amelynek növelésével nyilván az észlelési távolság is nő.



16. ábra. Az infravörös érzékelő működése

Hasonlóan fontos szerepet játszik a tárgyak színe. A vakító fehér színű tárgyak sokkal jobban visszaverik a fénysugarakat, mint a matt fekete színűek. Úgyszintén meghatározó a tárgyak mérete is.

Mindezek valóban fontos szempontok lehetnek különösen akkor, ha a szenzort egy robotba szándékozunk beépíteni. Gondoljunk pl. arra, hogy világos padlón a vékony világoskék színű székláb a szenzor számára nagyon nehezen felismerhető, gyakorlati szempontból azonban valódi akadály.

Az IR (infravörös) szenzorok működésének lényegét ezek után már ismerjük, azonban még egy fontos dologra ki kell térnünk.

Ha a szenzor valóban az infravörös tartományba eső fénysugarakra reagál, akkor feltehető, hogy minden pillanatban akadályt jelez, hiszen a nappali fény is tartalmaz infravörös részt. Azt, hogy a szenzor csak az általa kibocsátott fénysugarakra reagáljon, úgy érhetjük el, hogy a fénysugarat nagyon gyorsan kapcsoljuk ki- és be. Természetesen a kiválasztott frekvencia nem lehet 50 Hz, hiszen ez esetben a szenzor minden szobai lámpára érzékenyen reagálna. Célszerű megoldás pl. a 10 kHz-es frekvencia választása, amely elég magas ahhoz, hogy a szenzort ne befolyásolhassák bizonyos zavaró tényezők, és ugyanakkor még beleesik a hallható tartományba, így a szenzort egy nagyohmos fülhallgatóval ellenőrizhetjük.

Összefoglalva: az IR szenzor egy kb. 10 kHz-es oszcillátorból, két IR LED-ből (egy adóból és egy vevőből) és egy hangfrekvenciás kapcsolófokozatból áll, amelyet az adó frekvenciájára kell beállítani.

Az IR szenzor működési elvének részletezése után szenteljünk néhány gondolatot az eszköz felépítésének.

Az eddigi kapcsolásokhoz képest ez esetben nagyon fontos szerepe van a toknak. Az IR-szenzort célszerű egy fémházba beépíteni, ami megvédi a környezetet a keletkező frekvenciától. Nem elegendő a teljes kapcsolást fémházba helyezni. Az egyes részeket is feltétlenül el kell egymástól választani, ha azt akarjuk, hogy a szenzor valóban jól működjék. A ház hosszanti mérete kb. 40-60 mm kell hogy legyen ahhoz, hogy mindkét részt kényelmesen elhelyezhessük. Végül nagyon fontos, hogy a fémtokot a kész kapcsolás földpotenciáljával összekössük, egyébként ugyanis a fém többet árt mint használ.

A tulajdonképpeni kapcsolást két részre, adóra és vevőre kell osztanunk. A szerelőlapon mindkét rész kb. azonos helyet igényel.

3.10.1 AZ ADÓ

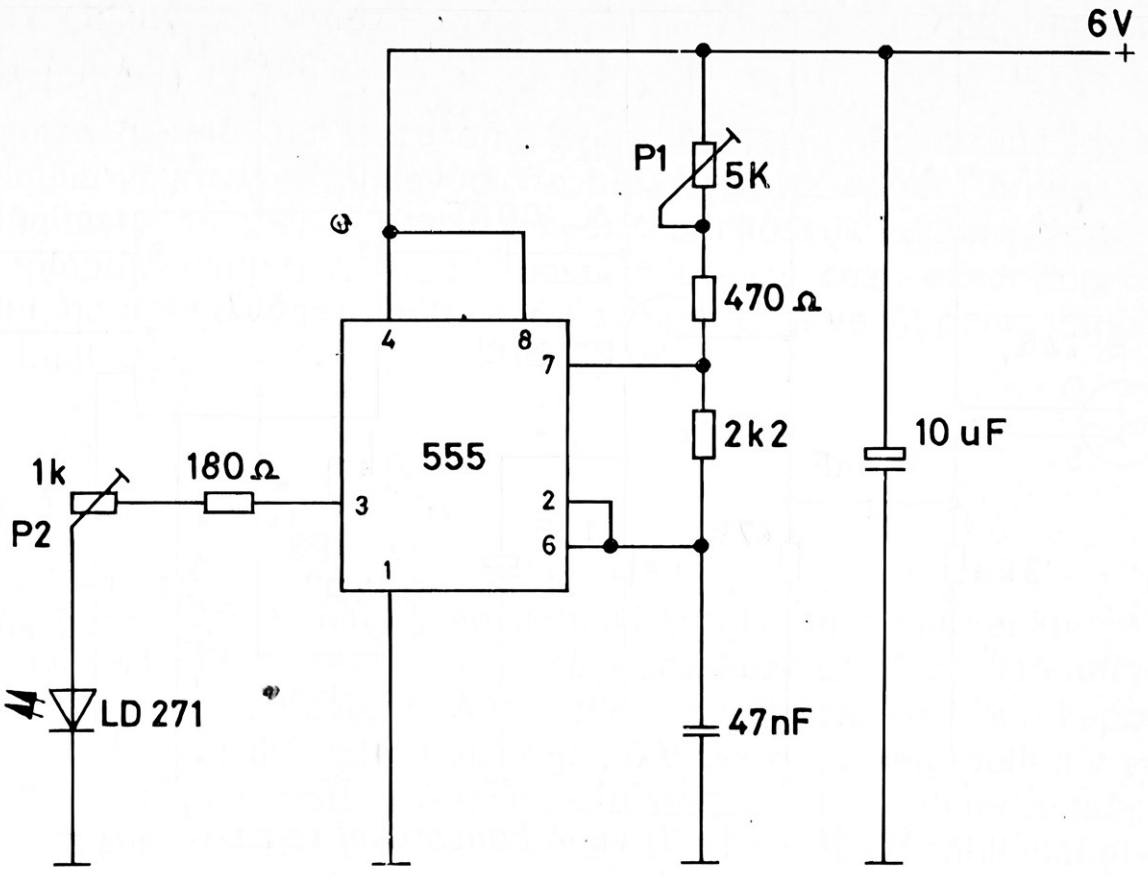
Az adó lelke az 555-ös típusú időzítő (timer). Ez az oszcillátor a mi kapcsolásunkban kb. 10 kHz-en rezeg, amelynek pontos értékét az 5 k-os potenciométerrel megváltoztathatjuk, hogy az adó és vevő frekvenciáit a lehető legpontosabban egymáshoz hangoljuk. Az 1 k-os potenciométerrel a fénydiódák világosságát, és ezáltal a szenzor hatótávolságát lehet szabályozni. Nagyon fontos, hogy az ellenállást, amelyet a P2-vel sorba kapcsoltunk, ne válasszuk alacsony értékűre (a nagyobb hatótávolság érdekében), mert ez tulajdonképpen a LED biztosítéka.

Az előellenállást úgy kell megválasztani, hogy a LED a potenciométer minden beállítása mellett gond nélkül üzemelhessen.

A kapcsolás eddig nem említett alkotórészei az oszcillátor frekvenciáját határozzák meg, kivéve az ELKO-t (elektrolit kondenzátort) amely az adó tápfeszültségét szűri.

Az adó kimenetére kapcsolt LD 271-es típusú fénydióda az emberi szem által nem látható infravörös tartományban bocsát ki sugarakat. Az adó hatótávolságát egy speciális LED reflektor beiktatásával, optikai úton lényegesen javíthatjuk.

Az IR LED-ek beépítésénél nagyon kell ügyelni a helyes polaritás megválasztására, hiszen a kapcsolás optikailag nem ellenőrizhető.



17. ábra. Az IR adó kapcsolási rajza

A kapcsolás megépítéséhez még egy javaslat: az időzítő elemet IC foglalatba szereljük, ez ugyanis mind az összeépítést, mind az esetleges szétszerelést nagyon megkönnyíti.

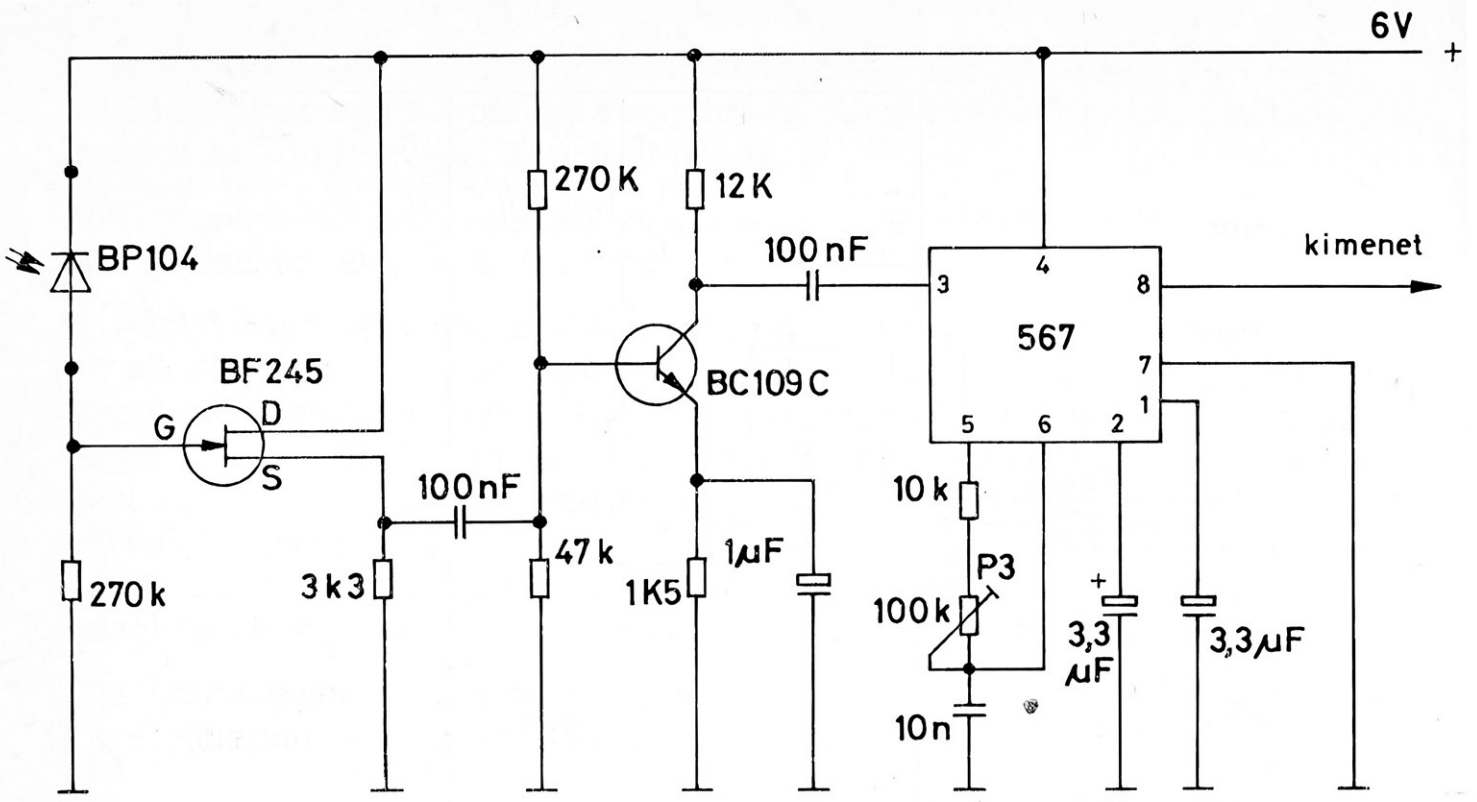
3.10.2 A VEVŐ

A szerelőlap másik felét elfoglaló vevő kapcsolása már messze nem olyan egyszerű mint az adóé volt. Kisebb odafigyeléssel azonban ezt is meg lehet érteni.

A vevő bemenetén egy BP 104-es típusú IR vevődióda található. Ez a dióda az adótól érkező fényt átalakítja nagyon gyenge árammá. A jel egy térvezérlésű és egy NPN típusú tranzisztorral kicsit felerősítve érkezik az 567-es PLL bemenetére. A felerősítésre feltétlenül szükség van, ugyanis a PLL elem észlelési alsó határa kb. 25 mV.

A PLL elem működése nagyon egyszerű.

Egyetlen oszcillátorból áll, melynek frekvenciája a 100 k-os potenciométerrel beállítható. A potenciométer finombeállításra szolgál, a 10 kHz körüli alapértéket az áramkör többi eleme határozza meg.



18. ábra. Az IR vevő kapcsolási rajza

Ha a jel frekvenciája megfelel az integrált oszcillátor frekvenciájának, akkor rezonancia frekvenciáról beszélünk. Rezonancia esetén az elem a kimenetet vezérli, amelyet maximálisan 100 mA-ral lehet terhelni. Mivel a beérkező frekvencia a PLL frekvenciájával pontosan sohasem egyezik, a PLL rögzít egy sávszélességet, amely meghatározza, hogy milyen mértékben lehet eltérni a beállított rezonancia-frekvenciától.

3.10.3 ÖSSZESZERELÉS

A közös lapon elhelyezett adót és vevőt el kell választani egymástól egy árnnyékolólemezzel. Az árnyékolót legjobban egy kis darab fóliázott lemezből lehet elkészíteni.

Meg kell azonban jegyezni, hogy az adó és vevő árnyékolására a FET erősítőfokozat a legalkalmasabb. Rögzítés és felragasztás után a fóliázott oldalt össze kell kötni a szenzor földpotenciáljával. Ha ellenőriztük, hogy rövidzárlat sehol sem léphet fel, a kapcsoló elvileg készen van, működőképes.

Mielőtt a kapcsolást beépítenénk a fémházba, ki kell fúrnunk a diódák és a kábelcsatlakozások helyét. A kábelek és az adódióda esetében ez nem okoz nehézséget, de a vevődióda esetében különleges eljárást igényel.

A doboz formája miatt a BP 104-est fel kell forrasztani egy kis fóliázott lemezre, majd a lemezt az árnyékolótokban a megfelelő helyre fel kell ragasztani. Mindkét diódát a lehető legrövidebb vezetékkel kell a laphoz kötni.

Az elkészült kapcsolást egy szigetelőként használt vékony papírlap segítségével építhetjük be a fémházba.

Rögzítsük a szerelőlapot feszesen a fémházba, majd helyezzünk egy kis darab fekete műanyaghengert a vevődióda elé, hogy az adóból érkező közvetlen megvilágítástól megóvjuk. Utolsó műveletként kössük össze a kapcsolás és a doboz földpotenciálját. A kész kapcsolás persze csak akkor fog hibátlanul működni, ha a teljes összeépítés előtt az adót és a vevőt pontosan összehangoljuk, beállítjuk.

3.10.4 BEÁLLÍTÁS

A kapcsolás beállítása nem egyéb, mint a három potenciométer kívánt mértékű elforgatása. A beállítás két egyszerű segédeszköze egy feszültségmérő és egy nagyohmos kristályfülhallgató. A feszültségmérőt összekötjük a kapcsolás kimenetével és a pozitív tápfeszültséggel. 6 V tápfeszültség mellett a voltmérőnek kb. 0.7 V-ot kell mutatnia. Rezonancia esetén a mutatott feszültség felugrik a tápfeszültség értékére. Hogyan kell tehát a kapcsolás beállítását elvégezni?

Először beállítjuk az adó frekvenciáját egy középértékre. A fülhallgatóval megállapíthatjuk az adófrekvenciát a föld és az időzítő 3-as lába között. Ha most a kezünkkel akadályt szimulálunk, a PLL elem 3-as lábán lehallgathatjuk a visszavert frekvenciát. Ha az adófrekvencia hallható, akkor az erősítőfokozat minden bizonnyal hibátlanul működik.

A P3 100 kΩ-os potenciométerrel beállíthatjuk az 567-es rezonanciafrekvenciáját, majd az 5-ös lábon lehallgathatjuk a PLL IC oszcillátorának frekvenciáját. Végül a P1 elforgatásával az adóban összehangolhatjuk a rezonanciafrekvenciát az adófrekvenciával.

Ha a 3-as és 5-ös lábakon hallható hangmagasságok azonosak, akkor a vevőt sikerült a megfelelő értékre beállítani. Vessünk egy pillantást még a voltmérőre: ha nincs a szenzor előtt akadály, akkor 0.7 V, egyébként 6 V feszültséget kell mutatnia.

Ha minden tökéletesen működik, felerősíthetjük a fémtokra a fedelet, ügyelve arra, hogy az IR szenzor árnyékolása is hibátlan legyen. Ejtsünk még néhány szót a szenzor elektronikus adatairól: nyugalmi állapotban, vagyis amikor nincs előtte akadály, a szenzor 6 V tápfeszültség mellett 17 mA-t vesz fel, egyébként 23.5 mA-t. A PLL elem 100 mA maximális kapcsolási áramot szolgáltat, amelyre már teljesítményfokozat nélkül rá lehet kötni néhány fogyasztót.

A csekély áramfelvétel lehetővé teszi, hogy az IR szenzort elemről vagy akkumulátorról működtessük.

Az IR szenzor már önmagában számos érdekes alkalmazási lehetőséget kínál. Alkalmazhatjuk pl. az ajtó felügyeletére, vagy az autó riasztóberendezéseként. Ezeknél azonban sokkal izgalmasabbak azok az alkalmazások, amelyek a szenzort a Commodore 64-esre csatlakoztatva használják.

Néhány ilyen megoldást ismertetünk a következő fejezetben.

3.11 Az IR szenzor illesztése a user portra

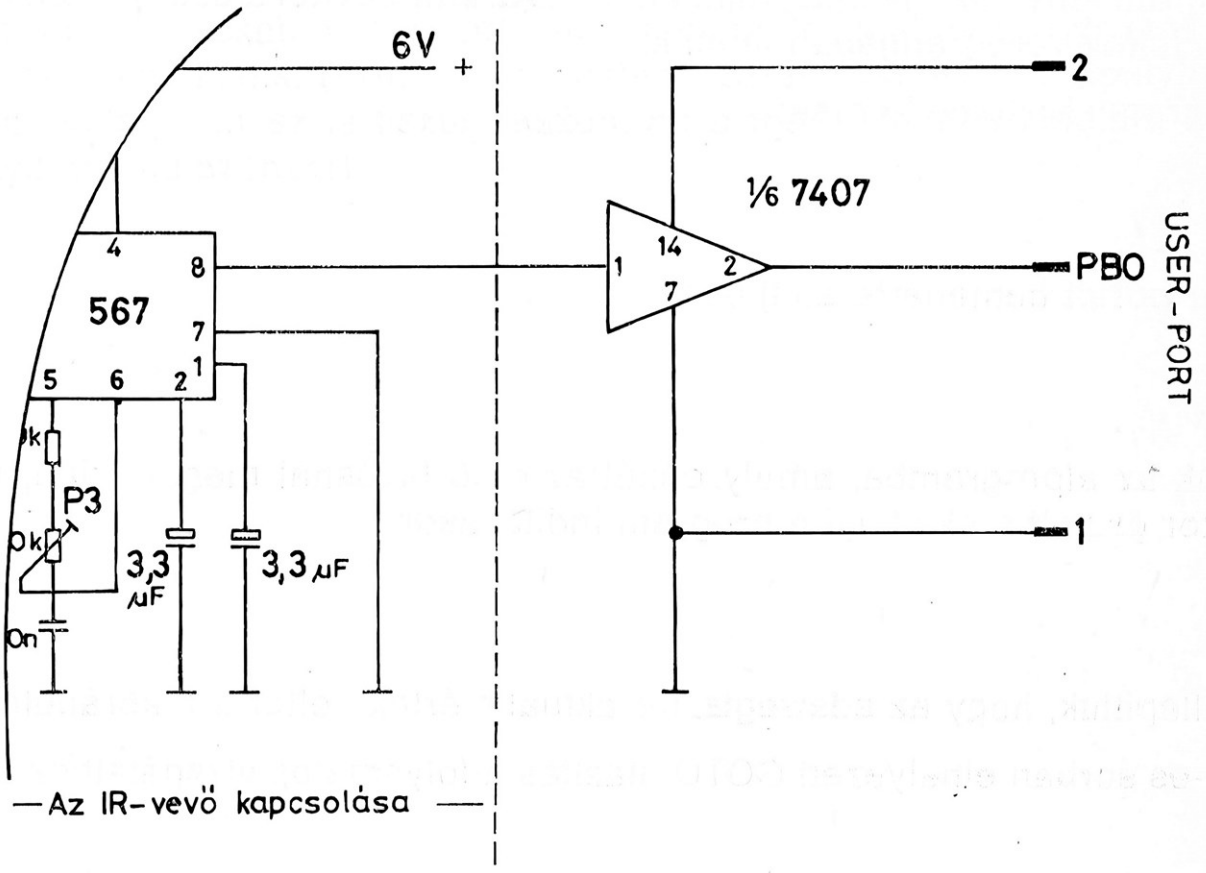
Az IR szenzor elméletileg készen áll arra, hogy a számítógéphez kapcsoljuk.

Hogyan?

Az összekapcsolás első pillantásra problematikusnak tűnhet, hiszen a szenzor és a gép különböző üzemi feszültségen működik.

Amint azt már tudjuk, a user portra nem szabad 5 V-nál magasabb feszültséget kötni. Ha azonban megmérjük a terhelt feszültséget a szenzor kimenetén, azt tapasztaljuk, hogy az lényegesen 6 V alatt van, és sohasem emelkedik 5 V fölé.

A csatlakoztatáshoz a már jól ismert 7407-es elemet használhatjuk, amely gondoskodik arról, hogy a szenzor által kibocsátott jeleket TTL jelekké alakítsa. A 7407-es beiktatása ugyanakkor biztosítja azt is, hogy a számítógép ne kapjon túlfeszültséget. A 19. ábrán bemutatjuk a csatlakoztatás egyszerű kapcsolását.



19. ábra. Az IR vevő összekötése a C 64-gyel

A TTL elem a szükséges tápfeszültségét megkaphatja a user portról. Mivel az elem hat meghajtót tartalmaz, összesen hat IR szerzort köthetünk vele a géphez.

A sikeres csatlakoztatás után már csak a programot kell elkészíteni, és a teljes berendezés működőképessé válik. A program minden pillanatban tájékoztat bennünket arról, hogy a szenzor észlelt-e akadályt vagy sem. Ehhez mindössze annyit kell tennünk, hogy leolvassuk a user port adatregiszterét. Ha a szenzort a PB 0 jelű lábra kötöttük, akkor 255 jelzi ha van, és 254 ha nincs akadály a szenzor előtt.

```
5 REM *** P7 ***
10 REM IR-SZENZOR A USER PORTON
20 POKE 56579,0
30 GOSUB 100
40 IF E<>PEEK(56577) THEN GOSUB 100
50 GOTO 40
100 E=PEEK(56577)
110 IF E=254 THEN PRINT"■   AKADALY!!!":RETURN
120 PRINT"■              NINCS AKADALY."
130 RETURN
```

A program lényege ugyanaz, mint amit a „Az adatbevitel a user porton keresztül" c. fejezet programjában láttunk.

A program részletes leírása:

20:

A user portot bemenetre állítjuk.

30:

Ugrunk az alprogramba, amely ennél az első hívásnál megállapítja, hogy a szenzor észlelt-e akadályt a program indításakor.

40:

Megállapítjuk, hogy az adatregiszter aktuális értéke eltér-e a korábbitól.

Az 50-es sorban elhelyezett GOTO utasítás a folyamatos vizsgálathoz szükséges.

Ha az adatregiszter tartalma megváltozott, ismét ugrunk az alprogramra, amely a képernyő feliratát megfelelően módosítja.

100–130:

Az alprogram tárolja az E változóban az adatregiszter tartalmát. A következő sorban megvizsgáljuk, hogy ez az érték 254, vagy nem. Ha igen, akkor az "AKADALY" egyébként a "NINCS AKADALY" szöveget írjuk a képernyőre. Mindez persze csak akkor helyes, ha a szenzort valóban a PB 0 jelű lábra csatlakoztattuk. Ha nem ez a helyzet, akkor a program konstansát (a 254-et) a bit helyzetének megfelelően módosítani kell.

Az IR szenzor alkalmazási lehetőségeit persze messze nem tükrözi a bemutatott program. Arra mindenesetre feltétlenül jó volt, hogy a programozási technikát szemléltesse. Ennek alapján könnyűszerrel készíthetünk pl. olyan programot, amely a szenzort ajtóőrként használja, és folyamatosan tájékoztat arról, hogy a megfigyelt térrészen tartózkodott-e valaki és ha igen, mennyi ideig.

Emlékezzünk azonban vissza kiindulópontunkra: a robotépítéshez olyan eszközre volt szükségünk, amely lehetővé teszi, hogy a robot fizikai érintkezés, ütközés nélkül adatokat gyűjtsön környezetéről.

Nyilvánvalóan a szenzor egyik legfontosabb alkalmazási lehetősége a reflexszel rendelkező szimulációs modell. A modell sokkal „intelligensebb" lesz, ha a billentyűket szenzorokkal helyettesítjük.

Szereljük ki a billentyűket és a hozzájuk tartozó ellenállásokat és építsük be helyettük szenzorokat. A programot nem kell módosítani, régi változata ebben az esetben is működik. Természetesen nem szükséges minden billentyűt szenzorra cserélni, már az is tiszta haszon, ha a modell elülső oldalára építünk billentyű helyett szenzort.

3.12 Az IR szenzor beépítése a szimulációs modellbe

Bár a billentyűk helyettesítése szenzorokkal valóban megoldható és nem okoz túlságosan sok gondot, ahhoz, hogy a csere eredményes legyen, nem árt néhány dologra odafigyelni.

A szenzorok alkalmazása alapvetően megváltoztatja a modell külső formájával szemben támasztott követelményeket. A szenzoros modellnek feltétlenül magasépítésűnek kell lennie. Minél magasabb a modell, annál jobb! Gondoljuk csak el, hogyan viselkedne a szenzoros robot egy világos színű padlón mozogva?

A világos padló folyamatosan nagymennyiségű fényt bocsát ki, amely a szenzort indokolatlanul átkapcsolásra készteti.

Az adó kimenőteljesítményét úgy kell beállítani, hogy a szenzor a világos padlót ne tekintse akadálynak, de a szék lábát igen.

Ha a szék lábát a program akadályként jelzi, mielőtt a modell nekiütközne, akkor a kísérlet sikerült.

A másik dolog, amiről nem szabad megfeledkezni: ha a modellbe egynél több szenzort építettünk be, ügyelnünk kell arra, hogy mindegyik más-más frekvencián működjön, egyébként ugyanis zavarni fogják egymást. Az egyes szenzorokat a már leírt módon kell beállítani, ügyelve arra, hogy egyik se reagáljon a másik frekvenciájára.

Lehetséges, hogy az előzőleg megépített modell egy-egy oldala túlságosan nagy ahhoz, hogy a szenzor ütközés előtt érzékelje az akadályt. Ekkor többféle megoldással javíthatunk a helyzeten. Az egyik lehetőség az, hogy egy oldalra több, legalább két szenzort szerelünk fel. Ez a megoldás egy kicsit költséges. A másik, feltehetően gazdaságosabb megoldás, ha kiegészítjük a modellt néhány további adó LED-del. Egy vagy több további adódiódával felszerelve a szenzort, a fényintenzitást többszörösére növelhetjük. Arra azonban vigyázzunk, hogy az 555-ös timer kimenetét nehogy 100 mA-nél jobban terheljük.

Egyébként a LED-eket célszerű a már meglévőkkel sorba kötni és az előtétellenállás értékét csökkenteni. Három LED-del már több, mint kétszer akkora felület védhető az ütközéstől.

3.13 Együttműködés a Commodore 64-essel köldökzsinór nélkül

A bemutatott robotvezérlések igen nagy hátránya:

mindegyiküket csak kábellel lehetett a számítógéphez csatlakoztatni, mintha egy köldökzsinór húzódna a modell és a gép között.

A vezérlést és áramellátást biztosító kábel nemcsak esztétikailag kifogásolható, de ellentmond a robot lényegének is. A robot lényege, hogy képes legyen mindenféle további berendezés segítsége nélkül adatokat gyűjteni a környezetéről.

Hogyan lehetne mindezt megoldani?

Sajnos magát a Commodore-t nem lehet kerekekkel felszerelve útnak indítani – bár sok mindenre jó – erre valóban nem készítették fel. Valami más megoldást kell keresni.

Ha szemügyre veszünk egy boltban vásárolt robotot, megállapíthatjuk, hogy mindegyikbe eleve beépítettek egy számítógépet, amely az energiát az összes többi fogyasztóval együtt egy akkumulátorból nyeri – így a robotot mozgásában valóban semmi sem gátolja. A beépített számítógépbe bekapcsolás után általában egy másik számítógép betölt egy programot, ami a robot intelligenciáját szolgáltatja. A program és az adatok áttöltése után a „vendéggépet" el lehet oldani a robottól, most már képes a kijelölt feladatot önállóan végrehajtani. Hogyan lehetne ezeket a tapasztalatokat szimulációs modellünk építése közben hasznosítani? Képzeljünk el egy tárolóegységet, amelybe a Commodore 64-essel be tudnánk tölteni egy vezérlő programot. Ezután a tárolót behelyeznénk a szimulációs modellbe, amely képes lenne annak tartalmát beolvasni és vezérlő programként kezelni. Így a modellt valóban függetleníthetnénk a számítógéptől egy előprogramozási fázis beiktatásával.

3.14 Vezérlés EPROM-mal

Amint az köztudott a számítástechnikában két egymástól alapvetően különböző tártípust használnak. Az egyik típus az ún. RAM a másik az EPROM. A RAM (Random Access Memory) írható és olvasható tár, amelyben programokat és adatokat tárolunk.

Az EPROM-ot a gép csak olvasni tudja, ezért ez az ún. „olvasható tár". Ebben tárolják az összes olyan programot (pl. az operációs rendszert), amelyet a számítógépnek kikapcsoláskor sem szabad elfelejtenie. Az egyszer beégetett programok üzemi feszültség alatt kiolvashatók, és akkor sem vesznek el, ha az EPROM-ot évekig szekrényben tároljuk. Ezzel szemben a RAM-ból minden információ elvész néhány tizedmásodperc áramszünet alatt.

Az EPROM-ba egy sajátos eszközzel, az ún. beégetővel rögzíthetjük az adatokat, hogy közben a beégető a számítógéphez van kapcsolva.

A tár tartalmát beégetés után árammal nem lehet törölni. A törléshez ultraibolya fényt használnak, amellyel egy bizonyos ideig meg kell világítani a chipet, hogy elveszítse információtartalmát.

Robotunkat tehát egy EPROM segítségével függetleníteni tudjuk a Commodore 64-estől. Ahhoz, hogy valaki ezt a robotot megépítse, nem kell feltétlenül rendelkeznie az EPROM beégető mellé egy EPROM törlő berendezéssel is. Elég ha van a raktárunkban egy olyan EPROM, amit az utolsó beégetés óta nem töröltek.

A kapcsolást, amit most ismertetünk kis jóindulattal már egypaneles számítógépnek is nevezhetjük. A jóindulatra azért van szükség, mert a kapcsolás nem tartalmaz mikroprocesszort, amely nélkül valójában nem számítógép a számítógép.

A kapcsolás középpontjában található az EPROM, a szimulációs modell vezérlésére, vagy egyéb célra készített programmal.

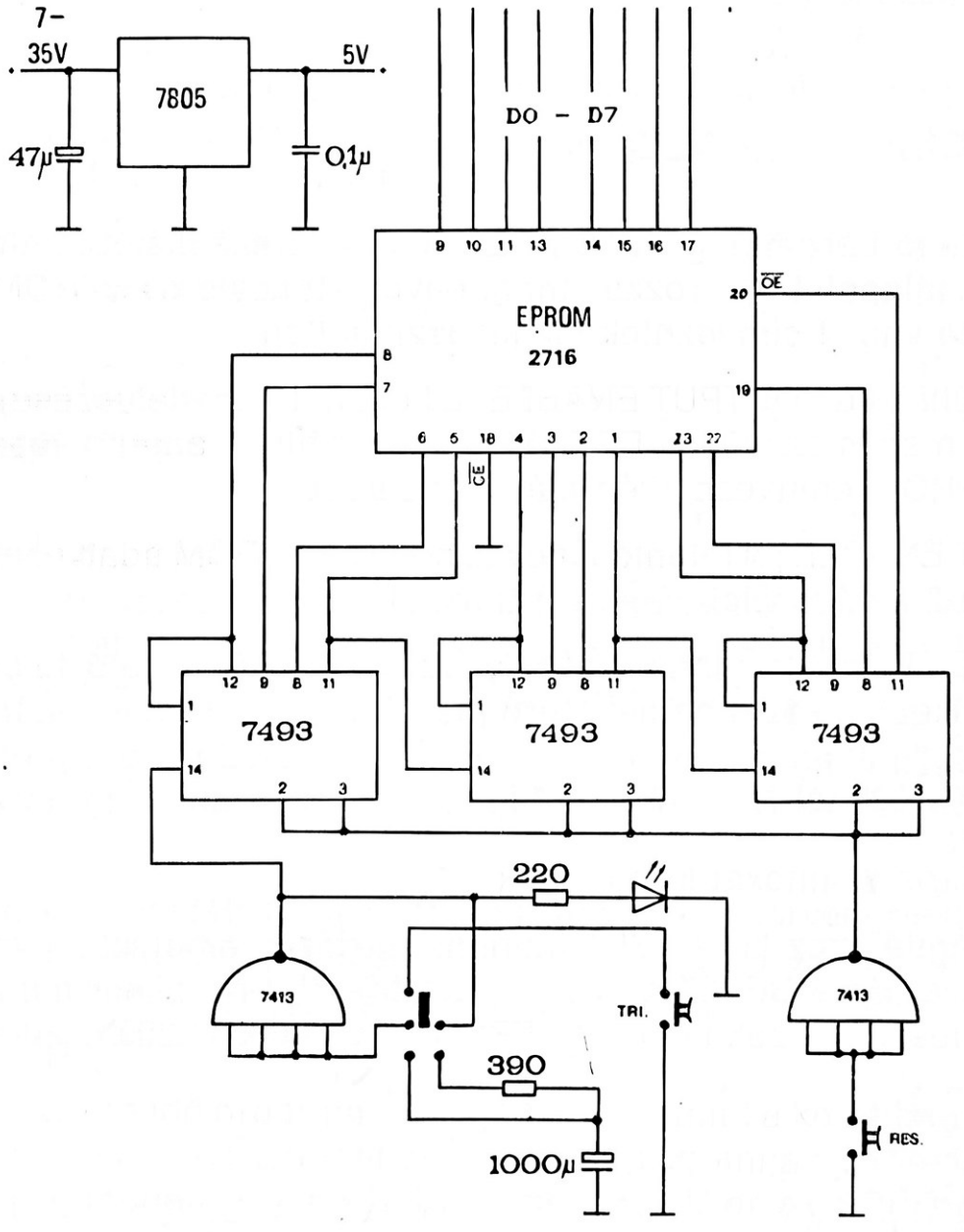
A kapcsolás további része lényegében négy IC-ből (integrált áramkörből) áll, amelyeknek az a feladata, hogy a beírt byte-ok tartalmát egymás után az adatvezetékre olvastassák.

A címbemenetek értékét három négybites számláló határozza meg, amelyek tartalmát automatikusan vagy kézzel lehet megváltoztatni. Az automatikus

számláló oszcillátora kb. minden fél másodpercben ad ki egy impulzust a számlálóknak, amelyek a következő EPROM címet adják. Ez azt jelenti, hogy a szimulációs modell minden fél másodpercben kaphat egy új vezérlőparancsot. Ha pl. egy 2716-os jelű 2 k-os EPROM-ot használunk, akkor a vezérlésre 2048 programlépést használhatunk fel. A teljes 2 kbyte-os program végrehajtása kb. 20 percig tart, ami már elég jelentős vezérlési időtartam. Ha az egyes programlépéseket ellenőrizni kívánjuk, akkor átkapcsolhatunk egylépéses üzemmódra (Single-Step-Modus). Ekkor a címvezeték tartalmát egy billentyű megnyomásával növelhetjük egyesével. A kapcsolás utolsó fontos része egy RESET kapcsoló, amellyel visszaállhatunk az EPROM-ban tárolt program első utasítására (a kiinduló állapotra).

A kapcsolás tényleges összeszerelése nem jelent különösebb nehézséget a vezérlési feladatok megoldása ezzel a módszerrel valóban gyerekjáték.

Hogy ez valóban ne maradjon puszta szó, foglalkozzunk mélyebben a kapcsolás részleteivel is.



20. ábra. EPROM-vezérlő kapcsolás

A kapcsolási rajzon négy egymástól jól elkülöníthető részt látunk. Ezek: az áramellátás, a címszámláló, a vezérlő egység és az EPROM.

3.14.1 AZ ÁRAMELLÁTÁS

A kapcsolás TTL elemekből épül fel, amelyek mindegyike 5 V tápfeszültséget igényel. Ha a tápfeszültség 4.75 V alá süllyed, semmi biztosítékot nem adhatunk az egyes TTL elemek további működésére. Ugyanakkor az 5,25 V-os feszültség már tönkreteheti a TTL elemeket. Ebből következik, hogy a kapcsolásba olyan áramforrást kell beiktatni, amely pontos feszültséget ad. Erre a célra pl. 7805-ös típusú 5 V szabályzót használhatunk, amely maximálisan 1 A áramerősséget szolgáltat. Ha a bemeneti feszültség 7...35 V között változik, a szabályzó mindig pontos TTL tápfeszültséget ad ki.

Az áramellátás részben találunk még két kondenzátort, amelyek az 5 V-os szabályzó feszültségeit szűrik.

3.14.2 A CÍMSZÁMLÁLÓ

A címszámláló három négybites 74 LS93-as jelű számlálóból áll, amelyek 12 címvezeték állapotát határozzák meg. Mivel a 2 kbyte-os EPROM-nak csak 11 címvezetéke van, 1 címvezeték így kihasználatlan.

Ez utóbbi tölti be az OUTPUT ENABLE (a kimenet engedélyezése) jel szerepét. Ha ez a jel magas szintű, az EPROM kimenete tiltott, ami pontosan azt jelenti, hogy az EPROM adatvezetékén már nincs adat.

Az OUTPUT ENABLE jel jelentősége az, hogy az EPROM adatvezetékét blokkolhatjuk ha a 2 k-ás terület végére értünk.

A számlálók láncba vannak kötve, azaz az alsó számláló túlcsordulása az utána következő fokozat bemenetére jut. A három összekapcsolt regiszterben

0000 0000 0000-tól 1111 111 1111-ig

terjedő bináris számokat tárolhatunk.

Minden számlálóhoz két RESET bemenet tartozik, amelyek egymással össze vannak kötve. Ha a RESET bemenet átvált 0-ról 1-re, akkor minden számláló tartalma 0 lesz, és ezek együtt az EPROM 0000 0000 0000 kezdőcímét adják.

3.14.3 A VEZÉRLŐ EGYSÉG

A vezérlő egység két billentyűt, egy kapcsolót és egy LED-et tartalmaz. A billentyűk egyike a RESET kapcsoló, amelyet egy 74LS13-as jelű Schmitt-trigger köt össze a számlálók RESET vezetékével. Ha megnyomjuk a RESET billentyűt, a kimenet logikai 0-ról logikai 1-re vált, ami a címszámláló tartalmának törlését eredményezi.

Mivel a 74LS13-as elem két Schmitt-triggert tartalmaz, a másikat felhasználhatjuk az oszcillátorhoz. A 100 µF-os ELKO és a 390 Ω-os ellenállás az oszcillátor mintegy 2 Hz-es rezgését eredményezi.

Az oszcillátor rezgését a LED láthatóvá teszi. A LED-et megfelelő előtét ellenállással a föld és a Schmitt-trigger kimenete közé kapcsoljuk.

A Schmitt-trigger kimenetét az első számláló bemenetére kapcsolva a cím minden impulzus hatására eggyel nő. A billenőkapcsolóval átválthatunk egylépéses üzemmódra. Az oszcillátor ilyenkor kikapcsol és a léptetés billentyűvel lehetséges.

Ebben az esetben nagyon fontos, hogy a billentyű jó minőségű legyen, egyébként ugyanis előfordulhat, hogy a számláló a billentyű egyszeri megnyomására a címet több értékkel továbbszámlálja.

A TTL elemek közül a kapcsoláshoz csekély áramfelvételük miatt az LS sorozat áramkörei a legalkalmasabbak.

3.14.4 AZ EPROM

Az EPROM programozásának módját minden esetben a feladat és a vezérlendő készülék típusa határozza meg. Bár maga a program szinte minden feladatra teljesen egyedi, a programozás folyamata változatlan.

Az EPROM-programozó készüléket hozzákapcsolva a számítógéphez, egy alkalmas segédprogrammal az adatokat rendre beégetjük az addig üres EPROM-ba.

Nézzünk erre egy példát: Nagyon fontos, hogy a motorvezérlést ne kössük közvetlenül az EPROM-ra, feltétlenül iktassunk közbe meghajtófokozatot, egyébként előfordulhatna, hogy az EPROM-ot túlterheljük. A meghajtófokozat az EPROM-panelről kaphatja az áramellátást.

Az EPROM-ba égetendő program logikája hasonló mint a korábban bemutatott, user portra csatlakoztatott szimulációs modell programja. Ha például azt szeretnénk, hogy a modell egy helyben maradjon, akkor az EPROM megfelelő

tárcímére nullát kell írni. Továbbmenve, a decimális 80 érték pl. az előre- míg a 160 a hátrafelé haladást eredményezi. Az egyes mozgásfajtákhoz rendelt számértékeket a 3.6 alfejezetben már felsoroltuk.

A program tényleges beégetése előtt egy már kész EPROM használatával meg kell határozni néhány állandó számértéket, amelyekre a vezérlés közben szükség van. Ilyenek például azoknak a ciklusváltozóknak a végértékei, amelyek megszabják hogy az egyes utasítások végrehajtását hányszor kell megismételni a vezérlési manőverek pontos időzítéséhez. Ezzel az EPROM beégetésének elvi kérdéseit tulajdonképpen tisztáztuk, nincs más hátra, mint az elvek gyakorlati alkalmazása.

Még egyszer felhívjuk azonban az Olvasó figyelmét arra, hogy meghajtófokozat beiktatása nélkül egyetlen áramkört sem célszerű az EPROM-mal összekötni!

Még egy apró ötlet:

Az EPROM hátrányos tulajdonsága, hogy a beégetett információt, az EPROM tartalmát meglehetősen nehéz törölni. Ez eléggé megnehezítheti a munkát, különösen eleinte, amikor a kiépítés még kezdeti stádiumban van, és a vezérlő programot is javítgatni kell.

Hasznos lehet ebben a stádiumban egy másik tárolótípus, az ún. szoft-ROM. A szoft-ROM tulajdonképpen egy RAM-szerű tár, amely rendelkezik az EPROM előnyös tulajdonságával, vagyis információtartalmát egy akkumulátor segítségével meg tudja őrizni. Ugyanakkor, RAM-jellegéből adódóan írható-olvasható tár, tehát tartalmát könnyen megváltoztathatjuk. Meg kell azonban jegyeznünk, hogy a szoft-ROM – bár kapható a kereskedelemben – jóval drágább az EPROM-nál.

3.15 A szimulációs modell mint fénykereső

A könyv bevezető részében megismerkedtünk a kibernetika legfontosabb alapfogalmaival.

Keltsük életre az elméletet ismét egy hasznos gyakorlati alkalmazással! Kibernetikai példánk egy olyan kapcsolás, amely a szimulációs modellt, egy „a fényre vágyakozó lénnyé" varázsolja.

A szimulációs modell sötét helyiségben semmiféle aktivitást nem mutat, türelmesen vár mindaddig, amíg fényt nem érzékel.

Érzékelői fényérzékeny ellenállások amelyek motorvezérlőként működnek. A rájuk eső fény csökkentve az LDR ellenállását – egy tranzisztorral kiold egy relét, amely a hozzárendelt meghajtómotort elindítja.

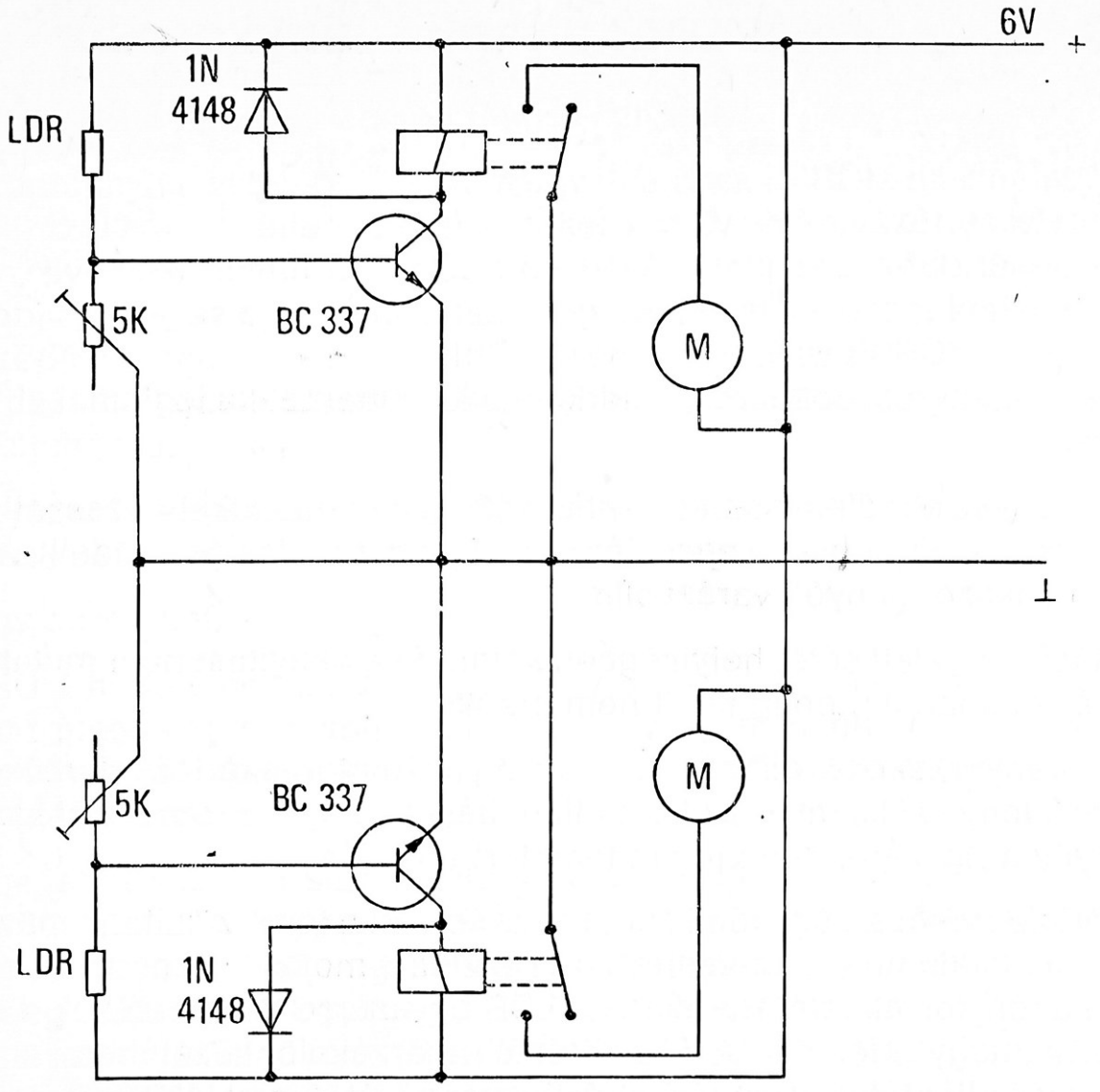
A szemlélőnek valóban úgy tűnik, mintha a modellt a fény csábítaná mozgásra, ugyanis amint valamelyik ellenállás fényt észlel, a motor bekapcsol és a jármű elmozdul a fényforrás felé. Ha mindkét LDR egyszerre fényt érzékel, a modell „feltételezi", hogy a fényforrás közvetlenül az érzékelői közelében van, ezért megfordul, és ellentétes irányban folytatja mozgását. Amint látótere ismét sötét lesz, motorja kikapcsol, és a modell ismét nyugalmi állapotba kerül.

Ezt az egyszerű kapcsolást kibernetikai tulajdonságai teszi érdekessé. De valójában mi is az, ami miatt egy modellt kibernetikai modellnek nevezünk?

Korábban megfogalmaztuk, hogy a kibernetika az önszabályozó rendszerekkel foglalkozó tudományág. Önszabályozó rendszer alatt viszont pontosan olyan típusú vezérlőkapcsolások összességét értjük, mint amilyen a mi modellünket mozgatja: a kapcsolás aktivizálja a modell meghajtómotorját fény hatására, majd a huzamosabb ideig tartó fényhatás ellentétesen hat: a motort ismét kikapcsolja. Ezáltal a modellünk valóban képessé vált arra, hogy a fényforrást „megkeresse". A fény így a modell mozgásának kiváltó ingere és egyben végcélja is.

A következő ábrán az elkészült „fénykereső" vezérlés kapcsolása látható.

A kapcsolás két azonos felépítésű részből áll, amelyek mindegyike megfelel egy-egy motorvezérlő egységnek.



21. ábra. A kibernetikus vezérlőegység kapcsolási rajza

A potenciométerek a két tranzisztor átkapcsolásának beállítására szolgálnak. A LDR-re eső fény csökkenti az ellenállást és a tranzisztorokat átkapcsolásra készteti. Ha a tranzisztorok kinyitnak, az áram átfolyik a relék tekercsein, és a relék meghúznak. A modell meghajtómotorjait közvetlenül a relék kapcsolják ki, ill. be. A motorok és relék 6 V üzemi feszültségen működnek.

A kapcsolás összeszerelése közben ügyelni kell arra, hogy a két LDR-t a modell „orrán" egymástól a lehető legtávolabb helyezzük el, olyan anyaggal elválasztva, amely a fényt nem engedi át. Ezzel érhetjük el azt, hogy a két fényérzékeny ellenállás egyszerre csak akkor érzékeljen egy pontszerű fényforrást, ha azzal a modell pontosan szemben áll.

A modell két térfelén elhelyezett motorokat keresztezve kell a kapcsoláson összekötni, különben a kibernetikus hatás a járművet eltávolítaná a fényforrástól. Ez pontosabban azt jelenti, hogy a bal oldali szenzor által érzékelt fénynek a jobb oldali motort kell bekapcsolnia és megfordítva.

A két részt most is a potenciométerek elforgatásával kell összehangolni.

Még egy ötlet azok számára, akik szívesen kísérleteznek:

Az előző fejezetekben megismerkedtünk egy reflexképességgel rendelkező modellel. Gondoljuk meg, milyen érdekes lenne, ha egy modellben ezt a képességet egyesíteni tudnánk a kibernetikai adottságokkal.

A végeredmény egy olyan robot lehetne, amely képes lenne a fényt, az útjába eső akadályokat ügyesen kikerülve is megkeresni!

Hogy hogyan lehet a kétféle képességgel egy robotot felruházni azt az Olvasó fantáziájára és ügyességére bízzuk, annyit azonban elárulunk, hogy az ötlet megvalósítható.

4. NÉMI FANTÁZIÁVAL...

Az előző oldalakon megismerkedtünk számos kapcsolással és eljárással, amelyek egy robot építéséhez alapvetően fontosak.

Ebben a fejezetben semmi egyebet nem teszünk, mint azt, hogy az összegyűlt tudásanyag elemeit újra, más módon illesztjük össze, hogy az egész végül is az igazi robot felépítéséhez vezessen minket.

Mindenkinek – akit ez a témakör foglalkoztat – megvan a saját elképzelése a robotról, vagy legalábbis arról, hogy hogyan kell egy robotnak kinéznie. Van aki robotként egyfajta morgó és brummogó szörnyet képzel maga elé, amint az fel- s alá rohangál, van akinek a fantáziájában a robot egy modern arculatú baba képe jelenik meg, végül van akit nem is annyira a robot külső megjelenési formája foglalkoztat, hanem az, hogy mekkora segítséget jelenthetne a háztartásban.

Ha valami képzeletben a tulajdonságok ennyire széles skálájával ruházható fel, arról nem lehet ma pontosan megmondani, hogy a jövőben miként valósul majd meg. Egyelőre ez az egyes emberek képzeletének szabadságára van bízva. Némi fantáziával és alapvető ismeretekkel felvértezve azonban mindenki elkészítheti a saját álmai robotját, amellyel szemben eleinte nem szabad túlságosan magas követelményrendszert támasztani. Annyi bizonyos, hogy az igazi kísérletező típusú ember nem nélkülözheti a fantáziát, és különösen nem a robotépítésben.

A fantázia ösztökéli a kísérletezőt újabb és újabb megoldások keresésére, miközben a korábbi gondolatok és ötletek újakat szülnek. Csak a fantázia szabadjára eresztése foszthatja meg gondolkodásunkat a konvekcióktól, amelyek béklyójában nincs mód az igazi alkotásra.

Éppen a robottechnika az a terület, amelyre az elmondottak a leginkább érvényesek. A robotok az új meghajtóelv feltalálásával kezdődnek és a szenzoron, a számítógépen vagy összefoglalva: az elektronikán át a szoftverig vezetnek, amely végső soron a vezérlés, a robot lelke.

Kísérelje meg valaki egy már meglévő eszköz tökéletes mását megépíteni, a saját kreativitását messze háttérbe kell szorítania.

A legfőbb tanács, amit végül is adhatunk azoknak akik robotot építenek:

ne szabjanak határt fantáziájuknak a technikai lehetőségek kihasználásában, és mielőtt nekifognának a szerelésnek, alaposan gondolják végig, hogy az egyes kapcsolások és programok milyen rejtett, első pillantásra nem látható lehetőségekkel szolgálhatnának.

A leírtak szellemében a könyv további fejezeteit nem is a robotépítés receptjeiként kínáljuk, amit pontról pontra követni kell, hanem kiindulópontnak, amely első ötletként szolgál a korlátlan lehetőségek felderítésében.

4.1 Alapötletek

A robotépítés első fázisa mindenképpen a tervezés. Elképzelésünket csak akkor tudjuk megvalósítani, ha minden részletet alaposan átgondolunk és lépésről lépésre megvizsgáljuk a megvalósítás feltételeit – a közben felmerülő kérdésekre egyenként válaszolva:

Milyen képességekkel rendelkezzen a robot?

Persze válaszul nem álmaink robotját kell bemutatnunk, hanem a rendelkezésünkre álló szerszámok, elektronika és programozási lehetőségek tárházát kell számba vennünk. A robot jövendőbeli képességeit ugyanis a felsoroltak bizonyos korlátok közé szorítják. Tervezéskor a megvalósításhoz szükséges idő, fáradtság és pénz sem utolsó szempont.

Hamar kedvét szegheti az alkotónak a kilátástalannak tűnő, hosszadalmas pepecselés.

Célszerű úgy tervezni a robotot, hogy építése közben viszonylag hamar sikerélményekhez jussunk, de ne zárjuk ki végleges megoldásokkal a továbbfejlesztés lehetőségét sem.

Érdemes megjegyezni; minél alaposabban terveztünk, annál könnyebb lesz a robotot újabb és újabb feladatok megoldására képessé tenni.

Az első kérdésből azonnal adódik a második, amelynek a végtermék szempontjából az elsővel azonos jelentőséget kell tulajdonítanunk, azonban csak akkor érdemes vele foglalkozni, amikor már az elsőre válaszoltunk.

Hogy nézzen ki a robot?

Itt ismét meg kell keresnünk a kompromisszumot elképzeléseink, a célszerűség és a technikai lehetőségek között. Javasoljuk az Olvasónak, hogy ezen a ponton egy rövid időre felejtse el mindazt, amit a tudományos-fantasztikus filmekben látott, és próbáljon a valóságos barkácsműhelyének talaján maradni.

Mindenesetre a valódi lehetőségek közül semmit sem szabad figyelmen kívül hagyni, nézzünk át mindent alaposan a barkácsműhelyben, sosem lehet tudni, hogy melyik félredobott kacat tehet most jó szolgálatot.

Még a fém- és műanyag-feldolgozásban gyakorlott mesterek számára is célszerű egy szerény karosszéria mellett dönteni, amelyet esetleg félkész darabokból könnyen össze lehet építeni. Játék- és modellező üzletekben általában

lehet kapni olyan félkész alkatrészeket, amelyek a robotépítésben nagyszerűen alkalmazhatóak, és ha a szükség úgy hozza, könnyen átalakíthatók.

Az utolsó, és egyben legfontosabb kérdés:

Milyen elektronikát célszerű beépíteni a készülő robotba?

A válasz természetesen erősen összefügg az első kérdésre adott válasszal: milyen képességekkel kívánjuk felruházni a robotot?

Az elektronikus részeket elsősorban az áramszükséglet szempontja alapján kell kiválasztani. Feltétlenül figyelembe kell venni, hogy a robot egy önálló rendszer, amely az áramot nem a hálózatról kapja, hanem egy magával hordozott áramforrásból.

A hordozható áramforrás mérete és súlya azonban a robot képességeivel arányosan nő – az egymással szembenálló tényező mindegyikével számolnunk kell. Pontosan ebből adódik a második alapszabály: célszerű minél könnyebb, kisebb méretű és teljesebb kapcsolásokat igénybe venni, hiszen minél kisebbek a kapcsolások, annál többet építhetünk be belőlük a robotba, ami növeli a robot intelligenciáját.

Az elektronika legfontosabb alkotórésze mindazonáltal maga a számítógép, ami még egy nagy vállalat keretei között is talán a legtöbb fejtörést okozhatja.

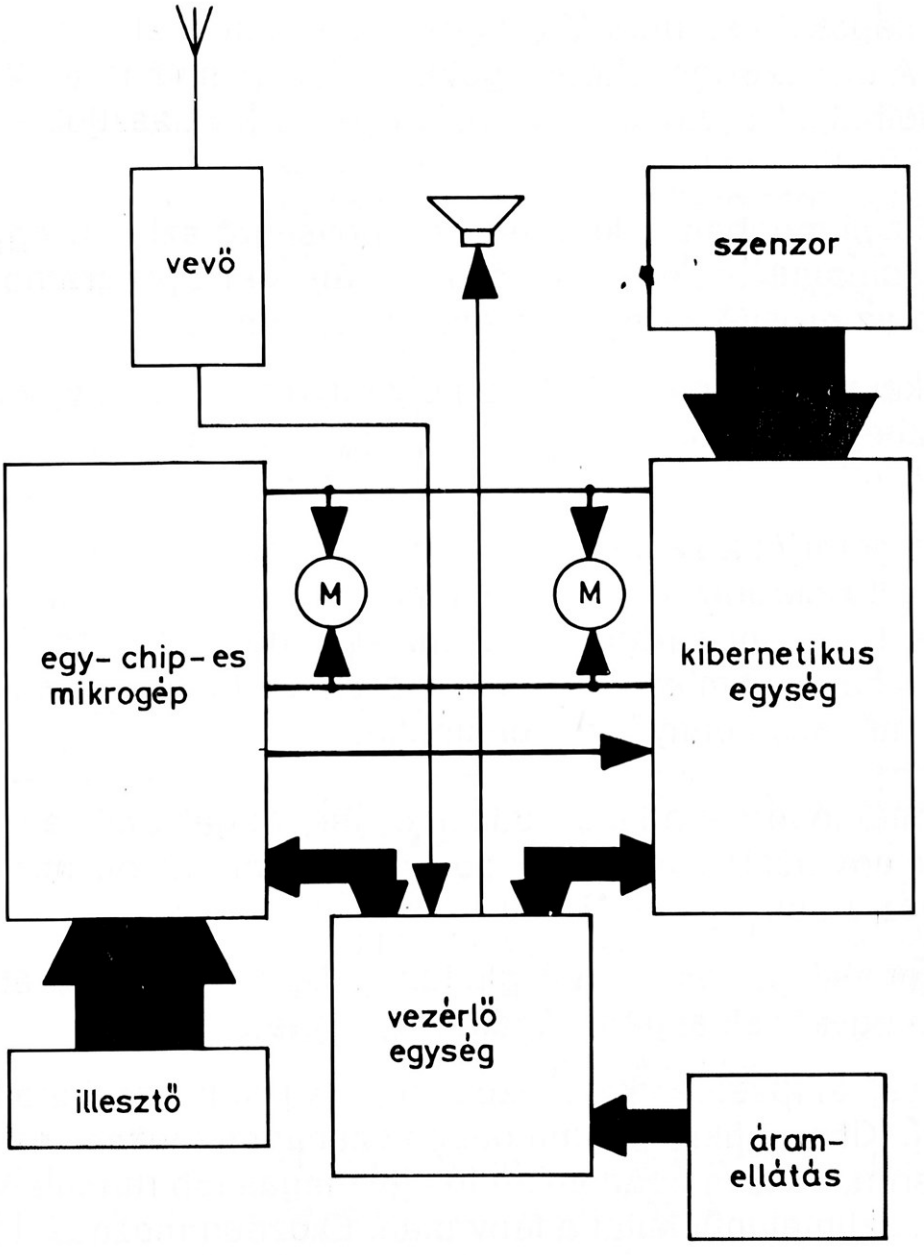
A számítógép kiválasztásánál nem ugyanazok a szempontok játszanak szerepet, mint egy háziszámítógép kiválasztása esetében. A legfontosabb szempontok, amelyeket a robotépítésnél figyelembe kell venni a sebesség, a portvezetékek száma, az illesztési lehetőségek, a processzor típusa, a tárterület mérete stb., és nem a kényelmes képernyőkezelés, ami egy családi számítógép esetében döntő fontosságú.

Ha valaki robotépítés közben az elmondottakhoz legalább nagyvonalakban igazodik, biztosan nem csalódik majd.

Valószínű, hogy sokkal izgalmasabb lenne a hosszadalmas tervezésnél, ha azonnal hozzákezdhetnénk a munkához. Ez azonban csak addig jelentene örömet, amíg bele nem ütköznénk az első nem várt akadályba, amit már csak mindent elölről kezdve lehet esetleg leküzdeni.

4.2 Ha javasolhatom, legyen a neve: HARO 5

A HARO 5, amint azt a neve is mutatja az ötödik a háztartási robotok családjában és bár jóval „tehetségesebb" elődeinél – még igen messze van álmaink sokoldalú háztartási robotjától. Elődeinek tudása mindössze néhány mozgásfajtára korlátozódott, és vezérlésük is meglehetősen házilagos kivitelezésű volt. Az utasításokat egymástól sötét vonallal elválasztva egy papírszalagon kellett előprogramozni. A vezérlő egységnek a sötét vonal jelezte, hogy új utasítás – új mozgásforma következik.



22. ábra. A sajátépítésű robot blokkvázlata

A HARO 5-öt az igazi HARO első változatának tekinthetjük. Technikailag jóval fejlettebb idősebb testvéreinél, bár tudása még meglehetősen korlátozott. Feladata, hogy a háztartási eszközöket egyik helyről a másikra szállítsa.

A modellt a „BIGTRAK" nevű programozható játékautóból készítettük, amely a Milton Bradely (MB) cég terméke.

A 22. ábrán jól látható, hogy a kapcsolás két fő része: a mikroszámítógép és a kibernetikai egység, melyek többféle módon össze vannak kötve egymással. Ugyancsak fontos szerepet tölt be a vezérlőegység, amely továbbítja a parancsot a különböző egységekhez.

A vezérlőegységre egy kis kapcsolópultot szereltünk, amely a tápfeszültséget bekapcsolja, ill. megszakítja. Amikor a főkapcsoló be van kapcsolva, a robot ún. STAND-BY üzemmódban van. Ebből az állapotból egy billentyű megnyomásával aktivizálhatjuk a mikroszámítógépet és a kibernetikus egységet.

A harmadik kapcsoló az ún. RUN kapcsoló, amellyel elindíthatjuk a mikrogép programját. A távvezérlés lehetőségét két tolókapcsoló teremti meg. Az egyikkel bekapcsolhatjuk a vevőt, a másikkal pedig kiválasztjuk az érzékenységi sávot.

STAND-BY üzemmódban a kibernetikus egységet szintén egy egycsatornás adóval kapcsolhatjuk be, és ugyanígy indíthatjuk el a programot is. A negyedik tolókapcsoló az erősítő ki-/bekapcsolására alkalmas.

A robot optikai vezérlője egy LED, amely egyrészt jelzi, hogy az akkumulátor tölt, másrészt egy 7 elemű kijelzővel mutatja, hogy mely részek vannak bekapcsolva.

A robot hajtómotorját a számítógép és a kibernetikus egység vezérli. A hajtás sajátossága a szinkronizáció: a két hajtómotort egy mágneskuplunggal összekötöttük, így, ha azonos irányú elfordulási parancsot kapnak, akkor a robot egyenletesen halad. Amikor viszont a motorok ellentétes irányba forognak, a kuplungot a mágnes könnyűszerrel kioldja.

A mikroszámítógépet és a kibernetikus egységet galvanikus úton tökéletesen elszigeteltük egymástól, így szabadon dönthetünk arról, hogy mikor melyik végezze a vezérlést.

Ezzel nagyvonalakban össze is foglaltuk a HARO 5 felépítését, rátérhetünk a legfontosabb egységek sajátosságainak ismertetésére.

A HARO „agya" alapvetően két részből áll, egy programozható és egy kibernetikus részből. Kibernetikus üzemmódban szabadon mozog „saját elektronikus akarata" szerint, azaz mindaddig amíg egy magasabb rendű elv nem kényszeríti más útra, szüntelenül kutat a fény után. Eközben mozgását a két fényérzékeny ellenállás szabályozza, amelyek a robot érzékelőiként működnek.

Ha mozgása közben valamilyen akadályhoz érkezne, akkor működésbe lép az IR-szenzor, amely kerülő útra kényszeríti. Abban persze semmiféle akadály nem gátolja meg, hogy szüntelenül, akkumulátorának kimerüléséig a fény felé igyekezzen.

Ha azonban átkapcsoljuk robotunkat program-üzemmódra, viselkedése egészen megváltozik. Szigorúan tartja magát ahhoz az utasítássorozathoz, amit tulajdonosa a számítógépébe programozott.

A hozzákapcsolható nagyobb gépen (pl. Commodore 64-esen) vagy közvetlenül a billentyűzetén keresztül lehet programozni. Az egyszerű mozgásformák programozásához már néhány utasítás elegendő.

Utasítás		Jelentés
Előre	:	mozgás előre,
Hátra	:	mozgás hátra
Jobbra	:	elfordulás jobbra
Balra	:	elfordulás balra
Szünet:	:	helybenmaradás egy meghatározott ideig
Váltás	:	átkapcsolás kibernetikus üzemmódra és vissza
Pulzus	:	szabadon programozható mozgásfajta
Ismétlés	:	nagyon fontos, tármegtakarító utasítás, jelentése: valamely programrész ismételt végrehajtása

Az utasításokat a kívánt mozgásfajtának megfelelően paraméterezhetjük. A „Balra 15" utasítás hatására pl. a robot 90°-kal balra fordul. Az egyes paraméterek jelentését persze előre rögzíteni kell.

A következő utasítások magát a programfejlesztést támogatják. Ezek az ún. szerkesztő vagy EDIT utasítások.

Edit utasítás		Jelentés
TEST	:	az utasítás kiváltja a robot öntesztelését
CLEAR MEMORIE	:	törli a gép tárát
CLEAR ENTRIE	:	törli az utoljára begépelt utasítást
CHECK	:	a program kipróbálásának eszköze. Hatására a robot végrehajtja az utoljára végrehajtott utasítást
RUN	:	a teljes program végrehajtása

A programot a billentyűzetről nagyon nehéz hiba nélkül begépelni, hiszen nincs optikai kijelzés, és így nem látjuk amit beírtunk.

A Commodore 64-es a programfejlesztésnek sokkal alkalmasabb eszköze, bár ehhez egy az expansion (bővítő) portra csatlakoztatható speciális illesztőre van szükség.

A HARO 5 áramellátását nagyon egyszerűen oldottuk meg, két ólomakkumulátorral. Mivel a kapcsolás egyes részei más-más üzemi feszültséget igényelnek, a 6 V-ot egyszerű kapcsolásokkal 9 V-ra, ill. kétszer 3 V-ra kell alakítani. Ha az üzemi feszültség 5.8 V alá süllyedne, az áramellátó egység figyelmeztető jelzést ad.

Már-már tökéletes robotunkat rádiótávvezérléssel tehetjük egy kicsit még tökéletesebbé. A távkapcsolóval magunkhoz szólítva, tetszés szerint kibernetikus, ill. programüzemmódra kapcsolhatjuk.

Munka közben robotunk „lelki állapotának" megfelelő hangokat ad beépített hangszórója használatával. Hangjelzéssel hozza pl. tudomásunkra, hogy útjába akadály került. Az adatbevitelt is hangjelzéssel kíséri – tudatva hogy az éppen begépelt utasítást felismerte vagy sem.

Ha kitartó csipogása és kopogása kellemetlenné válna, a hangerejét csökkenthetjük egy potenciométerrel, sőt fecsegését ha úgy tetszik teljesen el is némíthatjuk.

Azt nem állíthatjuk, hogy aki megépíti az elmondottak alapján a HARO 5 elnevezésű háztartási szuperrobotot, az a továbbiakban mentesül a nehéz háztartási munkák (pl. a mosogatás) terhe alól. Ennek ellenére ne becsüljük le szolgálatait, nem megvetendő, ha pl. egy fáradtságos partin van valaki, aki hoz tulajdonosának egy üdítő italt.

Amint azt többször hangsúlyoztuk, a HARO 5 még nem egészen az, amit a sci-fi filmek kedvelői megálmodtak.

Arra azonban feltétlenül alkalmas, hogy szemléltesse, mi mindent lehet létrehozni egyszerű technikai eszközökből némi fantázia és kísérletező kedv birtokában.

4.3 Az erőmű

A robotok képességeinek általában a szűkös áramellátás szab határt, hiszen szabad mozgásukhoz önellátó áramellátásra van szükségük. Legjobb lenne, ha a robotot saját „erőmű"-vel szerelhetnénk fel. Ebben a fejezetben az „erőmű"-kérdést – mint a robottechnika egyik legégetőbb problémáját – próbáljuk tisztázni.

A robotokkal szemben támasztott követelmények között elsőként soroltuk fel azt az elvárást, hogy ne legyen szükségük kábelre, ami szabad mozgásukat tökéletesen meggátolná. Az eddig ismertetett megoldásokban a független robot elemről vagy akkumulátorról „táplálkozott".

Igaz, hogy az elemek látszólag olcsó megoldást jelentenek, de óriási hátrányuk, hogy hamar kimerülnek. Ugyanakkor a tartósabb akkumulátorok ára pedig jóval borsosabb. Ennek ellenére, ha választanunk lehet, célszerűbb az utóbbi megoldás mellett dönteni.

Akkumulátor és akkumulátor között azonban meglehetősen nagy különbséget tapasztalhatunk. A döntés, hogy ólom- vagy nikkel-kadmium akkumulátort használjunk, még nem túlságosan nehéz, ha összehasonlítjuk árukat és teljesítményeiket: a mérleg egyértelműen az ólomakkumulátor felé billen. Legcélszerűbb teljesen zárt gáz- és savsűrűségű akkumulátort használni, hiszen ezek semmilyen karbantartást nem igényelnek.

Ha azonban a karosszéria mérete nem korlátozza a beépíthető akkumulátorok méretét, akkor célszerűbb olyan ólomakkumulátort választani, ami a kívánt feszültségérték mellett viszonylag magas áramfelvételt biztosít a fogyasztónak, és ugyanakkor a súlya még elviselhető.

A HARO 5 akkumulátora 680 g súlyú, 6 V-on 2,4 Ah kapacitású.

Sajnos ilyen bonyolult elektronika mellett szinte teljesen kizárt, hogy a kapcsolás egyes részei azonos üzemi feszültségen működjenek. Így az akkumulátor feszültségét általában át kell alakítani. Az átalakítás még mindig sokkal célszerűbb megoldás, mintha minden feszültségértékhez külön áramforrást építenénk be a robotba.

Amint azt már említettük a HARO 5 esetében az akkumulátor által szolgáltatott 6 V-ot 9 V, ill. 2 × 3 V feszültség értékre kellett alakítanunk.

Az átalakítást egy egyszerű kis kapcsolás végzi el. Az ún. DC–DC átalakító egy meghatározott egyenfeszültséget képes magasabb egyenfeszültséggé alakítani.

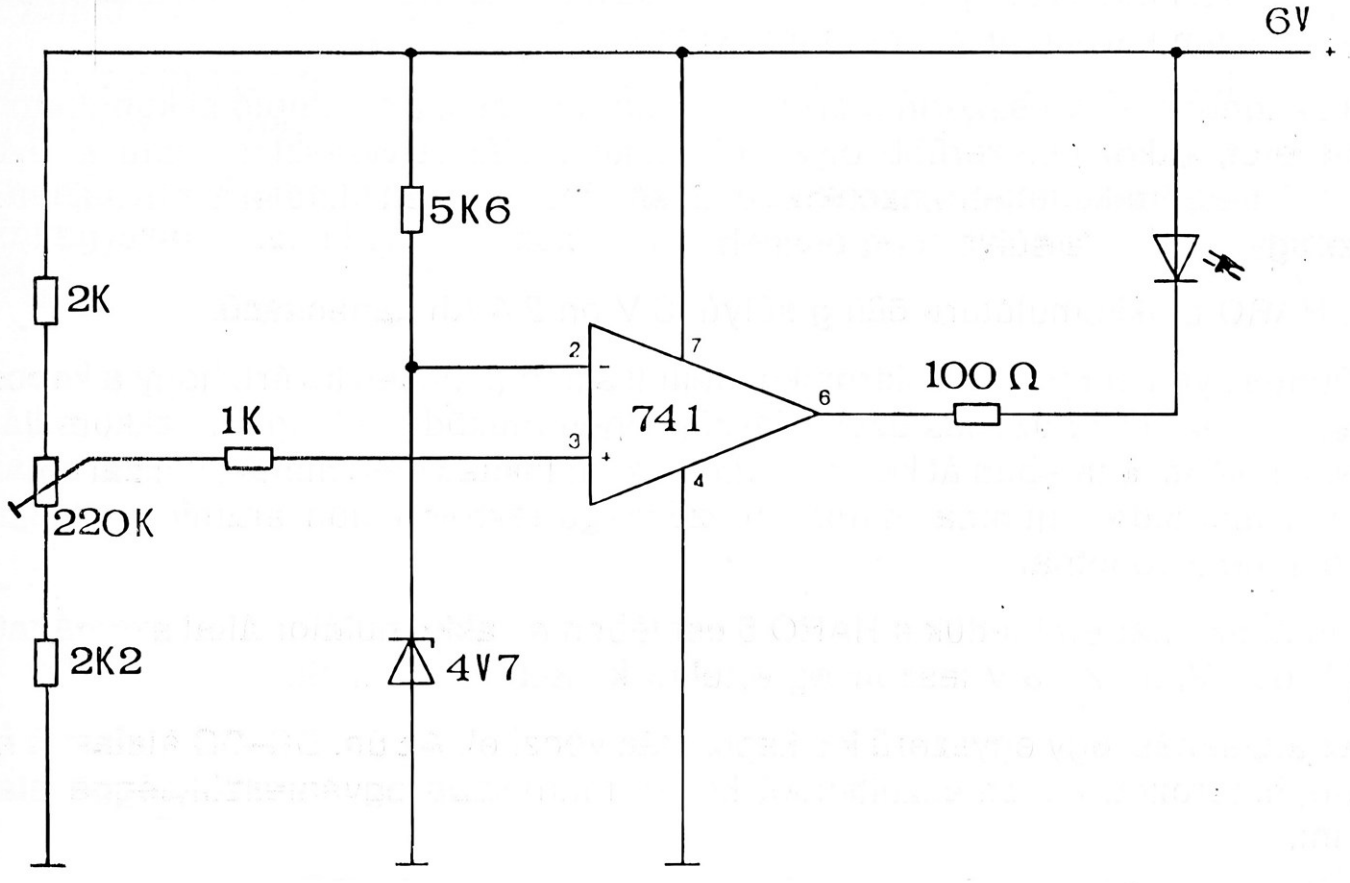
Egyszerűbb megoldás az ún. hasított áramellátás. Ebben az esetben egy műveleti erősítő 3 V-nál létrehoz egy mesterséges nullapontot. A mesterséges nullaponttól mérve az eredeti 0, −3 V, a 6 V pedig +3 V lesz.

Annak, hogy a különböző üzemi feszültségeket egyetlen áramforrásról nyerjük nemcsak az az előnye, hogy megoldható egyetlen akkumulátorral, hanem főként az, hogy a tápfeszültség ellenőrzését egyszerűvé teszi. Arról is esett már szó, hogy az egyes elektronikai elemek biztonságos működéséhez viszonylag állandó üzemi feszültségre van szükség. Az ellenőrzést egy műveleti-erősítővel ellátott egyszerű kapcsolás végzi. Jelzi az eltérést egy LED-en keresztül, vagy „jelenti" azt közvetlenül a számítógépnek.

A robotot a program akkor visszairányíthatja pl. az akkumulátortöltőhöz. Másik hasznos megoldás, ha ilyenkor a program a legfontosabb adatokat gyorsan tárolja egy biztonsági tárba, mielőtt a számítógép áramellátása is bizonytalanná válna.

Az üzemi feszültség felügyeletét mi egy 741-es típusú műveletierősítővel oldottuk meg. Ennek az elemnek van egy invertáló és egy nem invertáló bemenete. Az egyiken a referencia – a másikon a pillanatnyi tápfeszültség mérhető. A bemenetek felcserélésével túlfeszültség vagy a névlegesnél alacsonyabb feszültség észlelhető.

A következő kapcsolás a 6 V üzemi feszültség ellenőrzésére készült. Ha nem 6 V-ot akarunk mérni, a kapcsolást némileg módosítani kell, legalább még egy Zéner-diódát be kell kötni.



23. ábra. A feszültségőr kapcsolási rajza

Az optikai kijelzőt (LED-et) nagyon egyszerűen csatlakoztathatjuk a számítógéphez, amely az eltérésre megfelelőképpen reagál.

A kapcsolás központi része a 741-es műveleti erősítő, amely folyamatosan összehasonlítja a táp- és a referenciafeszültséget. Az ábrán közölt bekötésben az áramkör azt jelzi, hogy a tápfeszültség a referenciaérték alá csökkent. Ha ugyanis a műveleti erősítő a két feszültség között eltérést talál, a kimenetet (a 6-os lábon) átkapcsolja.

Azt, hogy az átkapcsolást mekkora feszültségeltérés váltsa ki, a 220 k-ás potenciométerrel állíthatjuk be. Természetesen ezt a határt úgy kell megszabni, hogy a figyelmeztető jel leadásának pillanatában az üzemi feszültség még elég magas legyen ahhoz, hogy a kapcsolás megfelelő részei a jelet fel tudják dolgozni.

4.4 Az agy

Az áramellátáson kívül a robotnak feltétlenül rendelkeznie kell egy központi vezérművel, egy aggyal, amely a folyamatosan érkező adatokat összegyűjti és feldolgozza.

Az agy szerepét a számítógép tölti be.

A számítógép számos input/output vonalán zajló adatforgalom biztosítja, hogy a robot a külvilágból a szenzorain keresztül érkező információt képes összegyűjteni, és azokra megfelelően reagálni. Így képes például bizonyos érzékenységi tartományon belül megállapítani, hogy milyen távol van tőle a hozzá legközelebb eső akadály. A külvilágból érkező információt a robot közvetlen formában nem tudja hasznosítani. A számítógép feladata éppen az, hogy az adatokat a robot számára is érthetővé alakítsa.

A számítógépet össze kell kapcsolni a robot szenzoraival, amelyek az információt észlelik. A gép és a szenzor közötti adatforgalmat nem minden esetben lehet egy egyszerű IR-szenzorral megvalósítani. Általában szükség van egy közbenső kapcsolásra, amely a szenzor kimenetére csatlakozik és az adatokat a számítógépnek előkészíti.

Gondoljunk pl. arra az esetre, amikor a szenzor analóg jelet bocsát ki. Minthogy a számítógép az analóg jelet közvetlenül nem tudja feldolgozni, be kell iktatni egy analóg-digitális átalakítót, amely a géphez a már érthető digitális jeleket továbbítja.

A számítógépnek beérkező adatok feldolgozásán kívül egy sor, a robot által feldolgozható kimenő adatról is gondoskodnia kell. A vezérlendő egységek természetesen nem tudják a számítógépből érkező jeleket közvetlenül értelmezni – gondoljunk pl. az elektromotorokat vezérlő motorfokozatokra. Egyébként az elektromotor-számítógép illesztés még viszonylag egyszerűen megoldható. A robot „agyának" működésében a hardver eszközökön kívül a szoftvernek is jelentős szerepe van. Az nyilvánvaló, hogy az input adatok feldolgozására, az output adatok előkészítésére csak egy szoftver, egy megfelelő program képes. Ráadásul ahhoz, hogy a robot kellő intelligenciával rendelkezzen nem is túlságosan egyszerű program szükségeltetik. A legtöbb barkácsolónak általában nem is a robotépítés technikai része, mint sokkal inkább a programok elkészítése jelenti a gondot. A robotvezérlés közben felmerülő feladatokat (pl. a port adatvonalainak folyamatos írását és olvasását) célszerű gépi kódban programozni, mivel a hardverközeli utasítások végrehajtása magasszintű

programnyelven általában nagyon időigényes. Általánosságban elmondhatjuk, hogy a robotépítésnek elengedhetetlen eszköze a gépi kódú programozás.

Ráadásul ha a robotnak egyszerre többféle tevékenységet kell elvégeznie, a vezérléshez már nem is elegendő egyetlen számítógép, többprocesszoros rendszert kell kiépíteni. A központi számítógép feladata ilyenkor az egyes processzorok tevékenységének összehangolása a résztevékenységek kiosztása.

A processzorok közötti munkamegosztásra számos lehetőség kínálkozik.

Célszerű megoldás például az adatgyűjtést és a feldolgozást elkülöníteni.

Az első processzor begyűjti a szenzorról érkező adatokat és előzetes szelekció után átadja a közvetlenül fölé rendelt processzornak. Az utóbbi eldönti, hogy mit kell tenni az adatokkal, és azokat a robot megfelelő alárendelt egységeihez továbbítja. Az alárendelt egységek szintén rendelkezhetnek egy processzorral, amely a kimenő adatok előkészítéséről gondoskodik. Ezen a ponton további megkülönböztetéseket tehetünk aszerint, hogy az adatok által generálandó feladat pl. egyszerű előremozgás, vagy egy összetett művelet (pl. egy tárgy felemelése) amely további bonyolult vezérlést igényel, esetleg egy újabb alprocesszor beiktatásával.

A robot működésének alapelveit áttekintve valóban megbizonyosodtunk arról, hogy gyors reagálási készsége nagymértékben függ az adatok feldolgozási sebességétől. A megfelelő reakcióidőt csak többprocesszoros rendszerrel érhetjük el, amelynek a gyorsaság mellett számos további előnye is van.

Az egyes processzorokat úgy kell hálózatba kapcsolni, hogy közöttük az adatforgalom zavartalan legyen. Ez azt jelenti, hogy a robot egy processzoros részegysége kiemelhető, és helyére egy másik építhető be.

A többprocesszoros rendszert úgy kell elképzelnünk, mint több önálló tárral és I/0 egységgel rendelkező számítógép összekapcsolását. Ha a műveletek végzése közben kiderült, hogy valamelyik egység működése nem elég hatékony, pl. a szenzor vezérlésével nem vagyunk elégedettek, akkor helyére egy javított egységet építhetünk be anélkül, hogy a többi egységre ez hatással lenne. Ehhez az szükséges, hogy a továbbítandó adatok azonos típusúak legyenek. Ha ez nem így van, akkor a központi számítógép programját kell megfelelően módosítani.

A többprocesszoros rendszer további előnye, hogy különböző típusú processzorokat is tartalmazhat. Az egyszerű feladatok elvégzésére elegendő adott esetben egy egybites vagy egy négybites processzor. Sőt a rendszerbe beépíthetünk olyan chip-eket is, amelyekbe egy konkrét feladat megoldására alkalmas programot égettek be. Természetesen a mértéket itt is célszerű betartani, hiszen a túlságosan sokféle processzor gépi nyelvének összehangolása nem mindig a legegyszerűbb feladat.

Visszatérve HARO 5 elnevezésű kedvenc robotunkra, megállapíthatjuk, hogy a rászabott feladatok ellátására tökéletesen elegendőnek bizonyult egy egyprocesszoros mikroszámítógép. Ebben nincs semmi meglepő, hiszen a HARO egyáltalán nem képes olyan bonyolult tevékenységre, amelyek összehangolásához komplex robotagyra lenne szükség. A HARO 5 „agyának" legfontosabb feladata az, hogy az egyes bemeneteket folyamatosan lekérdezze – ezt a feladatot pedig egy egyszerű programmal megoldottuk.

További feladata a hajtómotorok vezérlése, és a kerekek irányításához szükséges visszajelzés.

Végül az utolsó két kimenet lehetőséget ad a két üzemmód (kibernetikus és szabad vezérlésű) közötti átkapcsolásra.

Meglepő, hogy mennyi apróságra kell odafigyelni, még egy ilyen igazán szerény tudású robot építése közben is. Képzeljük el, hogyan sokszorozódnak a teendők bonyolultabb feladatok ellátására tervezett robot építése közben.

A HARO mikrogépe egy speciálisan erre a célra készített programot tartalmaz, amelynek nyelve is speciálisan ehhez a feladathoz készült.

A program egyszerű, így természetesen semmi másra nem is használható, mint a HARO 5 vezérlésére. Azoknak, akik a robotot nem saját maguk készítették, nagy könnyítést jelenthet, hogy a vezérlést egy igen egyszerű, gépközeli nyelven programozhatják.

Ennek a nyelvnek egy egyszerű eleme lehet pl. a következő utasítás:

ELŐRE 30

ami azt jelenti a robot számára, hogy mozogjon előre 30-szor olyan távolságra mint a saját hossza.

Az operációs rendszert természetesen fel kell készíteni arra, hogy az utasításokat értelmezni tudja, és a kijelölt műveletet (pl. a meghajtómotor bekapcsolását) a megfelelő egységgel elvégeztesse.

A kialakított programnyelv olyan egyszerű, hogy azt bárki nagyon könnyen elsajátíthatja anélkül, hogy a processzorról bármit is tudna.

A helyzet tulajdonképpen ugyanaz, mint bármely más magas szintű programnyelv esetén.

Ha valaki megtanulja pl. a BASIC nyelvet, minden olyan gépen tud programozni, a gépbe épített processzor típusától függetlenül, amelynek operációs rendszerében van BASIC értelmező.

4.5 Mi a helyzet a visszajelzéssel?

A mechanikus szerkezetek számítógépes vezérlése elképzelhetetlen visszajelzés nélkül. A számítógép a visszajelzés alapján határozza meg a mechanika pillanatnyi állapotát.

Azt, hogy a hajszálpontos vezérlés visszacsatolás nélkül szinte lehetetlen vállalkozás, azt már szimulációs modellünk esetében is tapasztalhattuk, amelynek mozgását csak nagyvonalakban sikerült előre meghatározni. Azonos program, többszöri kísérletben más és más mozgáspályát eredményezett. Az eltérések magyarázata egyszerűen az elektromotorok azon tulajdonságában rejlik, hogy kikapcsolás után még egy bizonyos – előre meghatározhatatlan ideig – tovább forognak. Igaz, hogy egy-egy útszakaszon csekély eltérés mutatkozik két különböző kísérletben, a csekély eltérések azonban egy hosszabb útvonal megtétele közben jelentős mértékben felhalmozódhatnak.

Képzeljünk most el egy ipari robotot, amelynek az a feladata, hogy az autó karosszériáját egy megadott helyen megforrassza. Ha a fentihez hasonló eltéréseket megengednénk, a forrasztási pont beállítása egy idő elteltével teljesen eltérne a kívánttól, és a robot a karosszériát mindenütt megforrasztaná, csak ott nem, ahol szükséges. A feladat megoldásához elengedhetetlenül fontos, hogy a robot képes legyen érzékelni már az 1/10 mm-es nagyságrendű eltérést is.

A visszajelzésnek többféle módja ismeretes. Mi a módszerek közül aszerint válogatunk, hogy alkalmazhatóak-e robotok vezérlésében, vagy sem.

4.5.1 A KAPCSOLÓ

A visszajelzés legegyszerűbb eszköze a kapcsoló. A kapcsoló állása jelezheti a számítógépnek, hogy a mechanika végrehajtotta a kijelölt feladatot, vagy sem.

Ezt a megoldást alkalmazhatjuk pl. a liftajtó kinyitására. A számítógép gondoskodik arról, hogy az ajtó mindaddig zárva maradjon, amíg a lift el nem érte végállomását.

A végállomást elérve az ajtók nekiütköznek egy kapcsolónak, amelynek benyomott állapota jelzi, hogy az ajtót ki lehet nyitni.

Ahhoz, hogy a számítógép a kapcsoló állapotát észlelni tudja, valamelyik bemenetén egy programmal minden pillanatban le kell azt kérdeznie. A kapcsoló és a Commodore 64-es illesztéséről és a számítógépes programról már szó esett a 3.8 „Adatbevitel a user porton keresztül" című alfejezetben.

4.5.2 OPTIKAI VISSZAJELZÉS

Kifejezetten finom mechanikai szerkezetek általában nem alkalmasak arra, hogy egy kapcsolót működtessenek.

Ekkor a mechanika helyzetét optikai úton figyelhetjük. Gondoljunk pl. a korábban bemutatott IR szenzorra, amely kiválóan alkalmazható erre a célra.

Tegyük fel pl. hogy meghajtókerék elfordulását kell megfigyelnünk. A kereket olyan lukas tárcsára rögzítjük, amely egyébként átláthatatlan anyagból készül. A megfigyelést egy adó- és vevődiódával ellátott IR szenzorral végezhetjük.

Valahányszor a kerék elfordulása közben az IR szenzor és a tárcsán található valamelyik rés egy vonalba esik, a szenzor fényt kap. A megfigyelés most már nagyon egyszerű. Ha pl. a tárcsán összesen 30 rés van, akkor egy körülfordulás alatt a szenzor 30 esetben észlel fényt.

A számítógép számlálva a szenzorból érkező jeleket, tetszőleges számú elfordulás után megállíthatja a meghajtómotort.

A számítógép természetesen annál pontosabban tudja vezérelni a kerék mozgását, minél sűrűbben helyezkednek el a tárcsán a lyukak.

Ha a tárcsán 30 lyuk van, akkor a mozgás legkisebb megfigyelhető egysége a kerék kerületének 1/30-ad része. 100 lyuk esetében a megfigyelés pontossága megháromszorozódik.

A HARO 5 szintén optikai visszajelzéssel működik. A visszajelzéshez szükséges lyukak a meghajtó fogaskerékben találhatóak, és ehhez van erősítve az IR szenzor is.

4.5.3 A POTENCIOMÉTER

A visszajelzés harmadik módja a potenciométeres visszajelzés, amelyet könnyen be lehet építeni pl. a robot karjába. Ez a megoldás – szemben az optikai megfigyeléssel – nem az elfordulást, hanem a helyzetváltoztatást jelzi.

A visszajelzés elve nagyon egyszerű. Fenti példánkban a robotkar csuklójához erősítünk egy potenciométert, amely a csukló mozgásával együtt mozog. A potenciométer meghatározza egy hanggenerátor frekvenciáját, amit a számító-

gép bemenő jelként fel tud dolgozni. A frekvencia változása követi a potenciométer, ill. ezáltal a kar helyzetváltoztatását. Persze a kar pontos helyzetét a számítógép csak akkor tudja minden pillanatban követni, ha minden csuklót felszerelünk visszajelző potenciométerrel.

A visszajelzésnek ez a módja támogatja legjobban a robot öntanulását. Ha a számítógép a mozgásról, a kar helyzetváltoztatásáról folyamatosan érkező adatokat tárolja egy alkalmas program segítségével, a mozgást bármikor „visszatudja játszani", azaz meg tudja ismételtetni. Így a megfigyelőrendszer azt is lehetővé teszi, hogy bárki, aki a programozáshoz egyáltalán nem ért – megtanithatja a robotot bizonyos feladatok elvégzésére.

4.5.4 LÉPTETŐ MOTOR

A visszajelzésnek a mikroelektronikában igen gyakran alkalmazott módja a léptető motor, amellyel különösen gyakran találkozhatunk a számítógépek külső egységeiben, pl. a lemezmeghajtóban vagy nyomtatóban.

Ebben az esetben nem is beszélhetünk a szó szoros értelmében vett visszajelzésről, mindössze arról van szó, hogy a számítógép minden pillanatban tudja a külső egység motorjának helyzetét.

A léptető motor belső felépítése alapvetően eltér a közönséges elektromotor belső felépítésétől, és éppen ez a különbség, ami a robottechnikában különösen alkalmazhatóvá teszi.

A léptető motor legfontosabb részei az álló- és a forgórész, amelyek közül az első meghatározott számú elektromágnest, míg a második ugyanannyi permanens mágnest tartalmaz. A rotor (forgórész) egy körülfordulása fel van osztva a mágnesekkel azonos számú szögelfordulásra.

A legkisebb szögelfordulás nagyságát a mágnesek száma szabja meg. A rotort elektromágneses vezérléssel hozhatjuk mozgásba. Az elmozdulás legkisebb egységét – egy lépést – két, a mozgás irányában egymást követő elektromágnes váltja ki.

Hogyan hasznosíthatjuk a léptető motort a robottechnikában?

A léptető motornak minden egyes lépésben egy impulzust kell kapnia a vezérlő számítógéptől. A számítógép egy számlálóban mindig pontosan tárolja a kiadott impulzusok számát. Mivel minden impulzus megfelel egy lépésnek, az impulzusok számából pontosan meghatározható a motor pillanatnyi helyzete. Ha valami miatt a léptető motor helyzetét kívülről kell meghatározni, módosítás előtt alaphelyzetbe kell hozni, azaz pontosan tudnunk kell, hogy a motor hány impulzus után kerül ismét alaphelyzetbe.

Ezt minden Commodore 64-es lemezegységgel rendelkező Olvasónak kellett tapasztalni lemezformálás közben. A formálás első lépéseként ugyanis az operációs rendszer a lemezegységet mindig alaphelyzetbe hozza.

Sok szó esett arról, hogy a léptető motorra alapozott visszajelzési módszer minden más módszert háttérbe szorít.

Azt azonban eddig még nem említettük, hogy egyben ez a módszer a legköltségesebb is.

Meghajtásához ugyanis legalább annyi energia szükséges, amennyi már a robot meghajtásához is elegendő lenne. Ez a magyarázata annak, hogy az amatőrök szívesen maradnak valamely másik megoldás mellett.

4.6 A kibernetikus egység

A robotnak van néhány olyan tevékenysége is, amelyek vezérléséhez nincs szükség feltétlenül számítógépre. Bizonyos feladatok megoldására egyszerűbb kapcsolások is elegendőek.

Ilyen egyszerűbb kapcsolásokat tartalmaz a kibernetikus egység, amellyel már a HARO 5 leírásánál találkoztunk.

A 22. ábra blokkvázlata alapján láttuk, hogy a HARO 5 meghajtómotorját a számítógép és a kibernetikus egység egyaránt vezérelheti. Természetesen egy adott időben a kettő közül csak az egyik lehet aktív. Valahányszor a számítógép bekapcsol, a kibernetikus egység áramellátása megszűnik. Ugyanakkor ha a robot éppen fénykereső állapotban van, a számítógép nem kap tápellátást.

A HARO 5 kibernetikus egysége többre képes, mint amennyiről a korábbiakban szó esett.

Azon túl, hogy érzékeli a fényforrásokat, a robot útjába eső akadályokat is képes felismerni.

Ehhez mindössze négy szenzorra van szüksége – két LDR-re a fénykereséshez és két infravörös szenzorra az akadályok felismeréséhez. A HARO 5 tökéletesen sötét térben vagy azonos fényviszonyok mellett egyenes irányban közlekedik.

Abban a szempillantásban azonban, amikor a térben megjelenik egy fényforrás, azt a fényérzékeny ellenállások észlelik, és a robotot a kibernetikus egység a fény felé irányítja.

A mozgás irányát úgy határozza meg, hogy a két szenzorra eső fény mennyisége azonos legyen. Ha a két LDR által mért fénymennyiség eltér, a kibernetikus egység a robot mozgásirányát azonnal módosítja.

A fény keresésével párhuzamosan az IR szenzor folyamatosan figyeli a robot környezetében levő akadályokat. Ha egy adott helyzetben a szenzorok akadályt észlelnek, a robot minden más tevékenységét felfüggeszti, és az akadály elkerülésére koncentrál.

Az akadály által kiváltott reflex egyrészt arra készteti a robotot, hogy egy bizonyos ideig hátrafelé haladjon, és így az akadály kikerüléséhez legyen elegendő helye. Ezután megvizsgálja, hogy az akadály tőle jobbra, vagy balra található. Ugyanis a tőle jobbra lévő akadályt balra kerüli meg, a balra lévőt pedig jobbra.

Ha a kikerülési manőver közben a kibernetikus egység újabb akadályt észlel, az előző manőver végrehajtását felfüggeszti és megpróbálja az uj akadályt kikerülni.

A HARO kibernetikus egységét közismert elemekből raktuk össze, felhasználva a csekély áramfelvétele alapján erre a célra különösen alkalmas MOS IC-t. A számítógépre a kibernetikus egységnek csak a bekapcsolásáig van szüksége (SHIFT parancs). A kibernetikus egységet és a számítógepet egy fénykapcsoló köti össze, ami mindig nagyon célszerű megoldas, ha két olyan kapcsolást – amelyeknek különböző tápfeszültségen keli működniük – galvanikusan el akarunk választani egymástól.

A kibernetikus egység egy billentyű lenyomásával, vagy néhány méter távolságból távvezérléssel kapcsolható be.

A kibernetikus egység részletes ismertetésével remélhetőleg meggyőztük az Olvasót arról, hogy nem kell minden egyszerű feladat elvégzésére számítógépet használni.

Mielőtt bármit is eldöntenénk, alaposan meg kell gondolni, hogy megoldható-e a feladat egyszerű kapcsolással, és csak akkor célszerű a számítógépet igénybe venni, ha az valóban indokolt. Mivel a számítógép csak digitális jeleket képes feldolgozni, analóg folyamatok ellenőrzését csak egy átalakító beiktatása árán tudjuk megoldani, ami nem mindig a legjobb megoldás. Ha van rá mód, érdemes egy egyszerű kapcsolás mellett dönteni.

4.7 Fontos tartozékok

Nem lehet minden, a robothoz szükséges elektronikus elemet a robotba beépíteni. Ilyen járulékos tartozékok azok az eszközök, amelyeket pl. a robot karbantartása igényel.

A boltban vásárolható robotok tartozékai általában a következők:
- akkumulátor töltő
- billentyűzet vagy egyéb eszköz a programozáshoz
- illesztő a nagyobb, könnyebben programozható számítógéphez
- távvezérlő

Az teljesen nyilvánvaló, hogy a távvezérlőt értelmetlen lenne beépíteni a robotba.

A robotot az illesztővel a Commodore 64-es géphez csatlakoztathatjuk, és így a 64-es gépen keresztül programozhatjuk, míg a billentyűzet a robot közvetlen programozhatóságát biztosítja. Az utóbbi esetben az egyes utasításokat a billentyűk megfelelő sorrendbeni leütésével írjuk be a robotba épített számítógép tárába.

A programozásnak ez a módja azonban rendkívül fáradságos és hosszabb programok esetén túlságosan sok hibalehetőséget rejt magában. Nagyobb programok kifejlesztésének eszköze az illesztő. Az illesztő egy MOS kapcsoló segítségével a billentyűzetet szimulálja. Az egyes kapcsolók vezérlését ez esetben a nagyobb számítógép – esetünkben a Commodore 64-es gép – végzi el. A Commodore 64-es gép egy programmal képes szimulálni a robotba épített kisszámítógépet és így az általunk fejlesztett programot fejlesztés közben ki is próbálhatjuk. Miután a program elkészült, rögzíthetjük azt a beépített számítógép tárában, és a robotot leválaszthatjuk a Commodore 64-esről, hiszen a tárolt program képessé teszi az önálló cselekvésre.

Hasonlóan semmi értelme nem lenne annak, hogy az ólomakkumulátorok töltőegységét beépítsük a robotba. Egyetlen esetben lenne a beépítésnek haszna, ha a robot képes lenne arra, hogy önállóan megkeresse az áramforrást és feltöltse akkumulátorait.

5. TEGYÜK PROFESSZIONÁLISSÁ!

Az előző fejezetek központi kérdése a robotok önálló megépítése volt.

Megterveztük a robot mechanikus részeit, felépítettük a kapcsolásokat és megírtuk a programokat. Végcélunk az volt, hogy az új ismeretek alkalmazásával valóban megalkossuk első tökéletes robotunkat.

Ebben a fejezetben az önálló összeszerelés alárendelt szerepet játszik.

Előtérbe kerülnek azonban a kereskedelemben kapható, a robotépítésben jól alkalmazható eszközök. Különösen fontosnak tartjuk, hogy kitérjünk az olyan vezérlési feladatok megoldására készített számítógépes egységekre, amelyek bővíthetik robotunk képességeinek sorát.

5.1 A számítógép hangja

Ebben az alfejezetben egy különösen érdekes külső egységgel, a beszédszintetizátorral szeretnénk olvasóinkat megismertetni.

Beszédszintetizátor alatt egy olyan elektronikus egységet értünk, amely egy számítógéphez, esetünkben a Commodore 64-es, géphez csatlakoztatható, és megfelelő program segítségével képes az emberi hangot utánozni.

A beszédszintetizátor megvalósítása még néhány évvel ezelőtt is igen nagy gondot jelentett a szakemberek számára, mivel nem sikerült erre alkalmas integrált áramkört szerkeszteni. A beszédszintetizátort hagyományos eszközökből kellett megépíteni, így az eredmény meglehetősen terjedelmes méretű lett, és a gramofon és a Diesel-motor hangjának keverékére emlékeztetett.

Ezek a készülékek azonban ma már múzeumi tárgyakká öregedtek. Az elektronikai szakemberek törekvése a nagy integráltságú elemek létrehozására megmaradt, és végül is sikerült megalkotni az első beszédgenerálásra alkalmas IC-t, amelynek teljesítménye többszöröse ősei teljesítményének. Ehhez az IC-hez már csak néhány apró elemre van szükség ahhoz, hogy tökéletes beszédszintetizátor váljék belőle, és ráadásul közvetlenül a számítógéphez csatlakoztatható. Az egyik ilyen elem az SP 0256–AL2 jelet viseli.

Az SP 0256–AL2 típusú áramkört szívesen alkalmazzák a beszédszintetizátor építésében, mert igen egyszerű kapcsolást igényel, és a programozása is nagyon egyszerű. Ráadásul a fonémák segítségével igen jó hangminőséget nyújt.

A fonémák, a beszéd egymástól különböző kiejtésű elemi egységei, amelyekből szavakat állíthatunk elő. Az utóbbi gondolat adja tulajdonképpen a beszédgenerálás legfontosabb alapelvét. Mivel a fonémákat az amerikai nyelvből vettük át, az SP 025–AL2-es „kiejtésén" is érezhető némi amerikai akcentus. Az elem megszólaltatásához hat adatvezetékére egy 0-tól 63-ig terjedő decimális szám bináris megfelelőjét kell adni. Ugyanis minden 0 és 63 közé eső számnak megfelel a beszéd egy elemi egysége a 2. táblázatban közölteknek megfelelően.

A táblázat a fonémának megfelelő decimális számok mellett tartalmaz példaként egy-egy szót is. A fonémához csak a nagybetűvel jelzett részek kiejtése tartozik hozzá. A jelzett időtartamokat a szintetizátor a fonéma megszólaltatásához használja.

Kód	Fonéma	Péld	Időtartam, ms
00	PA 1	Pause	10
01	PA 2	Pause	30
02	PA 3	Pause	50
03	PA 4	Pause	100
04	PA 5	Pause	200
05	DY	boY	290
06	AY	skY	170
07	Eh	End	50
08	KK 3	Comb	80
09	PP	Pow	150
10	JH	dodGe	100
11	NN 1	thiN	170
12	IH	sIt	50
13	TT 2	To	100
14	RR 1	Rural	130
15	AX	sUcceed	50
16	MM	Milk	180
17	TT 1	parT	80
18	DH 1	THey	140
19	IY	sEE	170
20	EY	bElge	200
21	DD 1	coulD	50
22	UW 1	tO	60
23	AD	AUght	70
24	AA	hOt	60
25	YY 2	Yes	130
26	AE	hAt	80
27	HH 1	He	90
28	BB 1	Busy	40
29	TH	THin	130
30	UH	bOOk	70
31	UW 2	fOOd	170
32	AW	DUt	250
33	DD 2	Do	80
34	GG 3	wiG	120
35	VV	Vest	130
36	GG 1	Guest	80
37	SH	SHip	120
38	ZH	aZure	130

Kód	Fonéma	Péld	Időtartam, ms
39	RR 2	bRain	80
40	FF	Food	110
41	KK 2	sKy	140
42	KK 1	Can t	120
43	ZZ	Zoo	150
44	NG	aNchor	200
45	LL	Lake	80
46	WW	Wool	140
47	XR	repAIR	250
48	WH	wHig	150
49	YY 1	Yes	90
50	CH	CHurch	150
51	ER 1	fIR	110
52	ER 2	fIR	210
53	OW	bEAU	170
54	DH 2	THey	180
55	SS	veSt	60
56	NN 2	No	140
57	HH 2	Hoe	130
58	OR	stORE	240
59	AR	alARm	200
60	YR	clEAR	250
61	GG 2	Got	80
62	EL	saddLE	140
63	BB 2	Busy	60

Az SP 0256–AL2 jelű IC az említett hat adatvezetéken kívül rendelkezik további három nagyon fontos vezetékkel.

Ezek közül az első a RESET vezeték, amellyel az IC-t alapállapotba hozhatjuk. A RESET vezetéket, a 28 pólusú ház 25. lábára kötötték, az IC bekapcsolása után egy rövid ideig logikai 1-ről 0-ra kell állítani.

A második vezeték az ún. ALD jel (a 20-as lábon), amelyen keresztül a számítógép jelzi, ha az elem bemeneteire érvényes adatot adott.

Hasonlóan az IC az SBY jellel hozza a számítógép tudomására, hogy egy fonéma feldolgozásával végzett, és készen áll a következő adat fogadására.

Az elmélet további részletezése helyett térjünk rá arra a fontos gyakorlati kérdésre, hogy lehet hozzájutni egy a Commodore 64-eshez csatlakoztatható beszédszintetizátorhoz.

Az AZTEC-Software cég gyártja és árusítja pl. a „Speakeasy" nevezetű beszédszintetizátort. Az egységet a Commodore 64-es gép user portjára kell csatlakoztatni és betöltve a működtetéséhez szükséges programot a gép máris átalakul beszédszintetizátorrá.

Az adatforgalom a számítógép és a külső egység között a leírt módon zajlik le. Az egység a számítógéptől érkező adatokat hat adatvezetéken, a RESET, ill. az ADL vonalakon kapja, és saját állapotáról az SBY jelű vonalon keresztül tájékoztatja a gépet.

Milyen program szükséges ahhoz, hogy a beszédszintetizátor egy tetszőleges szót kiejtsen?

A kérdés megválaszolásához tekintsük át először a következő program listáját, amely részben a Speakeasy egységhez tartozó szoftverből származik:

```
5 REM *** P8 ***
10 PRINT"■■              D E M O
20 PRINT"■■■■    A -SPEAKEASY- BESZEDGENERATORHOZ"
100 REM A GEPI KODU PROGRAM BASIC BETOLTOJE
110 FOR P=49152 TO 49206
120 READ A
125 POKE P,A
130 NEXT
140 SYS 49191
150 DATA 160,0,177,122,201,44,240,1,96,32,115,0
160 DATA 32,49,192,165,20,32,23,192,76,0,192,41
170 DATA 63,141,1,221,9,64,141,1,221,173,1,221
180 DATA 48,251,96,169,127,141,3,221,169,0,76
190 DATA 23,192,32,138,173,76,247,183
200 REM BESZADGENERALAS
210 SYS 49152,12,50,3:REM ICH
220 SYS 49152,28,12,11,4:REM BIN
230 SYS 49152,33,19,3:REM DIE
240 SYS 49152,37,17,12,16,7,4:REM STIMME
250 SYS 49152,33,26,55,1,55,3:REM DES
260 SYS 49152,8,24,16,9,49,31,13,52,55,4:REM COMPUTERS
```

Ránézve a listára azonnal szembetűnik, hogy a program két részből épül fel. A 100-as sortól a 190-es sorig terjedő rész egy gépi kódú program BASIC betöltőprogramja. A DATA sorokban látható gépi kódú programot a betöltő rész a 49152-es címtől a 49206-os címig terjedő területen tárolja. A gépi kódú program első hívása (a 140-es sorban) előállít egy RESET jelet, amely a hanggenerátort alapállapotba hozza. Erre azért van feltétlenül szükség, mert a Speakeasy bekapcsoláskor halk brummogó hangot hallat, és ez a hang csak a RESET jel hatására szűnik meg.

A gépi kódú program következő feladata, hogy a fonémák továbbításához az adatirányt az user porton kimenetre állítsa.

Az adatok továbbításában kap szerepet a BASIC program második része. Itt helyeztük el ugyanis a SYS utasítások paramétereként az adatokat. A következő gépi kódú programrész gondoskodik arról, hogy a szintetizátor csak akkor kapion új adatot, amikor az előző adatot már feldolgozta, azaz a decimális számhoz rendelt fonémát kiejtette.

Elvileg az egész feladat BASIC nyelven is megoldható, mégis célszerűbb gépi kódban programozni, elképzelhető ugyanis, hogy a BASIC program sebessége a beszédszintetizátor működésében némi zavart kelt.

A példa alapján nagyon könnyen készíthetünk más szavak kiejtésére alkalmas programot. Mindössze annyit kell tenni, hogy a 210-es sortól 260-as sorig terjedő programrészben kicseréljük a decimális számokat az új szavakban előforduló fonémáknak megfelelően.

A szükséges adatokat a táblázatból kell kikeresnünk. Arra azonban feltétlenül ügyelni kell, hogy a különálló szavakat mindig szünettel (pause) zárjuk le, egyébként ugyanis a generátor a szóban szereplő utolsó fonémát „elnyeli".

Könnyű programozhatósága mellett a Speakeasy felépítése is rendkívül egyszerű.

A beszédmodult nemcsak a Commodore 64-es, hanem bármely más számítógéppel is összekapcsolhatjuk. Ez a tény és ráadásul a kicsi méret (amely némi ötletességgel még tovább csökkenthető) a Speakeasy-t kifejezetten alkalmassá teszi arra, hogy a robotépítésben felhasználjuk. A napjainkban kapható robotok többségét felszerelték olyan beszédgenerátorral, amelyek általában alig különböznek a Speakeasy-től. A beszédgenerátorral a robot képes arra, hogy bizonyos helyzetekben néhány előre programozott szöveget „elmondjon".

A szöveg terjedelmét csak a tárolókapacitás és a programozó ideje korlátozza.

A beszédgenerátoron kívül van még néhány, robotunk szempontjából érdekes és hasznosítható eszköz, amelyekre érdemes figyelmet fordítani.

5.2 A robotkar

Nyitott szemmel járva a világban, közelebbről megvizsgálva a technikai csodákat, a különböző építményeket, gépeket és szerkezeteket, előbb-utóbb fény derül arra, hogy az ember zseniálisnak hitt alkotásainak többsége nem egyéb, mint a természet valamely részének gyenge utánzata.

Ki tagadhatja pl. a repülőgépek és madarak vagy a halak és hajók közötti rokonságot.

Az ember számos munkaeszközéhez a természetből vette a mintát, amely hihetetlen gazdag és kimeríthetetlen tárháza az újabb és újabb ötleteknek. Kétségtelenül az iparból jól ismert markoló kar sem egyéb, mint az emberi kar és kéz másolata.

Az ember a karját a vállától az ujjai hegyéig hihetetlenül sokféleképpen tudja mozgatni. Szinte leírni lehetetlenség a számtalan tevékenységet, amit kezünkkel és karunkkal el tudunk végezni, ezért nem meglepő, hogy még a felépítését végletesen leegyszerusítő technikai eszközök is sokoldalúan használhatoak.

A robotkar az emberi karral végezhető műveleteknek csak kicsiny hányadát képes utánozni, mégis hihetetlenül gazdag felhasználói területtel rendelkezik.

Milyennek kell lennie tulajdonképpen egy markoló karnak ahhoz, hogy a maga nemében az ember univerzális munkaeszköze lehessen? A kérdés megválaszolásához el kell döntenünk, hogy ehhez hányféle mozgás elvégzésére van szükség.

Először is alapvetően fontos, hogy a markolókar rögzítési pontja körül 360°-kal el tudjon fordulni, ez biztosítja ugyanis a szerkezet tetszőleges irányú mozgási szabadságát. Lehet, hogy számos munkaterületet találnánk az olyan robotkarnak is, amely csak 90°-os elfordulásra képes, az azonban messze nem tölthetné be az univerzális munkaeszköz szerepét.

Másodszor: a robotkar legyen képes a rögzítési pontjához viszonyított emelkedésre és süllyedésre – ezáltal válik ugyanis az előző síkbeli mozgás térbelivé. Ezt a két képességet az emberi kar számára a váll csuklóízületei biztosítják.

A robotkar harmadik mozgási síkja: a fenti mozgásfajtákat ne csak a rögzítési pontból kiindulva tudja elvégezni, hanem attól bizonyos távolságban lévő rögzítési ponthoz képest is (ezt a szerepet tölti be az emberi kar esetében a könyök).

A negyedik és ötödik mozgásfajta az emberi kéz mozgathatóságának analógja. A markoló legyen forgatható a saját tengelye körül, és ugyanakkor legyen hajlítható egy bizonyos mértékig.

Végül természetesen nagyon fontos, hogy a markoló valóban markoló legyen, azaz nyitni és zárni lehessen. Ebben a tekintetben is mintát vehetünk az emberi kézről, de az ujjakra – és arra hogy ezeket külön-külön mozgatni lehessen – nincsen szükség. Az emberi kéz esetében ezek elsősorban arra szolgálnak, hogy két kezünkkel egy időben két különböző tárgyat meg tudjunk fogni.

Ezzel valóban kimerítettük azokat a legegyszerűbb mozgási lehetőségeket, amelyre egy univerzális markolókarnak feltétlenül szüksége van.

Térjünk most ismét át a visszajelzéssel kapcsolatos kérdésekre.

Azt, hogy a markolókart elektromotorral vagy hidraulikával működtetjük, mindig az adott helyzettől, a robotkar térbeli elhelyezkedésétől, ill. a vezérlés típusától kell függővé tenni.

Induljunk ki most abból, hogy a markolót számítógép vezérli, és az a feladata, hogy különböző tárgyakat mozgasson egyik helyről a másikra. A munkaeszköz tervezése közben egészen szélsőséges helyzetekre is gondolnunk kell.

Elképzelhető, hogy az egyik tárgy törékeny üvegből készült, nagyon vékony és könnyű, míg a másik öntöttvasból van, nehezen kezelhető és rendkívül súlyos. Végül az is lehet, hogy a markolóval valami veszélyes anyagot, pl. tömény savat kell egyik helyről a másikra mozgatni. Az alkalmazási területtől függ, hogy a vezérlési folyamat mely pontjainál van szükség visszajelzésre.

A számítógép feladata az, hogy a robotkart úgy irányítsa, hogy az a tárgyat szorosan megragadhassa. A megfigyelő szenzorokat a vezérléshez és visszajelzéshez közvetlenül a robotkarba kell beépíteni. Ezek a szenzorok mérhetik pl. a kar tárgytól mért távolságát.

A számítógépnek abban a pillanatban kell a markolást elindítani, amikor a tárgytól mért távolság már majdnem nulla. Ebben a pillanatban nagyon fontos, hogy a munkaeszköz alkalmazkodni tudjon a tárgy anyagi tulajdonságaihoz. Másképpen kell megfogni az üveget, az öntöttvasat és a savval töltött edényt.

A kezelési sajátosságokat a szenzoroknak kell észlelniük és a számítógéphez továbbítaniuk, hogy az a markolót megfelelőképpen vezérelhesse. Ha például üveget kell a helyéről elmozdítani, akkor ügyelni kell arra, hogy a markolás ne legyen túlságosan erős, az öntöttvas formát ugyanakkor nagyon erősen kell megragadni, a savas edényt pedig óvatosan kell mozgatni, nehogy megsérüljön.

Tehát a robotkar helyes vezérléséhez nemcsak a tárgytól mért távolságot, hanem a kifejtendő erő nagyságát is pontosan meg kell határozni.

A robotkart felszerelhetjük speciális célú szerszámokkal. Pl. egy hegesztő készülékkel, amely így hegesztési munkákat is el tud végezni. De helyettesíthetjük a robot kezét pl. tapadókoronggal is, ha üveglapokat kell egyik helyről a másikra mozgatni.

Végül térjünk rá ismét a gyakorlatra. A következőkben ismertetjük egy robotkar összeszerelésének lépéseit. Ez a robotkar azonban nem általános célú, csak mágneses tárgyak mozgatására alkalmas. A kéz szerepét ebben az esetben elektromágnes tölti be.

A kar mozgási síkjai: szabadon forog rögzítési pontja körül, képes felemelkedni és lesüllyedni. A kar végére bármilyen speciális célú eszközt felszerelhetünk.

Az egyszerűség kedvéért visszajelzési lehetőséget nem építettünk be a robotkarba.

Minthogy a visszajelzésnek számos módjával ismerkedtünk meg, bárki megoldhatja a beépítést különösebb befektetés nélkül.

Az eddigiekben minden olyan elemmel megismerkedtünk, amelyre szükség lehet a robotkar és a Commodore 64-es összekapcsolása közben. A legfontosabb, hogy ne feledkezzünk meg a (relés vagy tranzisztoros) motorvezérlés elé meghajtó fokozatot beépíteni!

Az elektromágnes vezérlését a motorvezérléshez hasonlóan kell megoldani. A megoldásban a relés változatot használtuk, és annak csak egyik felét építettük be a karba. A relét az elektromágnes helyettesíti – így a mágnes elektronikus vezérlése máris készen van.

A programban az elektromágnest az user port 0-ás vezetékén keresztül vezéreljük. A motorvezérlést, amely a kar forgatásáért felelős, a PB 6-os és a PB 7-es vezetékekre kötjük.

A PB 4-es és PB 5-ös vonalak ugyanakkor a másik elektromotorhoz csatlakoznak.

Mechanikai és elektronikai szempontból a robotkar már működőképes, ami még hiányzik:

a vezérléshez szükséges számítógépes program.

A robotkar vezérléséhez sajátos programnyelvet alakítunk ki, amely minden szempontból igazodik a robotkar sajátosságaihoz.

A program elindítása után a kar mozgását az újonnan definiált utasításokkal irányíthatjuk, azaz az új nyelven programozhatjuk.

A robotkar vezérlésének utasításai:

KUP xx: Kar UP. Az utasítás hatására a kar xx magasságba emelkedik.

KDN xx: Kar DOWN. Az előzetesen xx magasságba emelt kart ismét a kiinduló helyzetbe mozgatja. Általában az itt megadott xx számérték kisebb mint az emelkedésnél megadott, ugyanis a súly tehetetlenségénél fogva bizonyos mértékben lejjebb mozgatja a kart a megadott értéknél.

TUR xx: TUrn Right. Az utasítás a kart xx mennyiséggel jobbra forgatja.

TUL xx: TUrn Left. Hatására a kar balra fordul xx mértékben. Ha egymásutáni jobbra és balra forgatásokkal a kart kiinduló helyzetbe akarjuk hozni, akkor a kétféle utasításban elvileg azonos xx értéket kell megadni. Ennek ellenére elképzelhető, hogy az elmélet nem követi tökéletesen a gyakorlatot, és a kar nem kerül vissza a kiinduló helyzetbe. Ekkor a kétféle elforgatás mértéke közötti különbséget gyakorlati úton kell meghatározni.

TAK: TAKe. Hatására az elektromágnes bekapcsol és felemeli a fémtárgyat. Az utasításnak nincs paramétere, hiszen csak a mágnes be- és kikapcsolása a feladata. Az eddig bemutatott utasításokat csak egymás után alkalmazhatjuk, azonban az utobbi utasítás hatása a többi végrehajtása közben is érvényben marad mindaddig, amíg ki nem adjuk a kikapcsolásra vonatkozó utasítást.

PUT: PUT. A TAK utasítással bekapcsolt elektromágnest ismét kikapcsolja.

GOT xx: GOTO. Az egyéb programnyelvekben megszokott feltétel nélküli vezérlésátadó utasítás. Segítségével egy-egy programrészt többször megismételtethetunk.

Az xx az a sorszám, amelyben a program végrehajtása folytatódik.

END: Hatására a vezérlőprogramból visszatérünk a főprogramra. Paraméterre itt nincs szükség.

A Commodore 64-esen elkészített program három részből áll. Az első menü pontban meghatározzuk a vezérléshez szükséges utasításokat. A második menüpontban a beírtakon javíthatunk.

Végül a harmadik menüponttal elindíthatjuk a vezérlő programot.

```
5 REM *** P9 ***
10 POKE53280,3
20 POKE53281,3
30 POKE 56579,255
40 POKE 56577,0
50 DIM UTS$(255)
60 DIM TAR(255)
70 FOR I=1 TO 255
80 UTS$(I)="END"
90 NEXT I
100 PRINT"▣▣▣      ╭──────────────────────────╮
104 PRINT TAB(6);"|";TAB(34);"|"
105 PRINT"      | ROBOTKART VEZERLO PROGRAM |"
106 PRINT TAB(6);"|";TAB(34);"|"
107 PRINT"      ╰──────────────────────────╯"
110 PRINT"▣▣      ╭──────────────────────────╮
120 PRINT TAB(6);"|";TAB(34);"|"
130 PRINT TAB(6);"|       MEGHATAROZAS    (1) |"
140 PRINT TAB(6);"|";TAB(34);"|"
150 PRINT TAB(6);"|       MODOSITAS       (2) |"
160 PRINT TAB(6);"|";TAB(34);"|"
170 PRINT TAB(6);"|       INDITAS         (3) |"
180 PRINT TAB(6);"|";TAB(34);"|"
190 PRINT"      ╰──────────────────────────╯"
200 PRINT"▣▣      ╭──────────────────────────╮
210 PRINT TAB(6);"|";TAB(34);"|"
220 PRINT TAB(6);"|     KEREM VALASSZON!      |"
230 PRINT TAB(6);"|";TAB(34);"|"
240 PRINT"      ╰──────────────────────────╯"
250 GET WA$
260 IF VAL(WA$)<1 OR VAL(WA$)>3 THEN GOTO 250
270 ON VAL(WA$) GOSUB 300,400,500
280 GOTO 100
300 REM MEGHATAROZAS
305 INDEX=1
310 PRINT"▣"
315 PRINT INDEX;".";
320 INPUT"   UTASITAS   ";UTS$(INDEX)
330 IF UTS$(INDEX)="END" THEN RETURN
335 INPUT"▣        IDOTARTAM ";TAR(INDEX)
340 INDEX=INDEX+1
350 PRINT
360 GOTO 315
400 REM MODOSITAS
405 INDEX=1
410 PRINT"▣"
420 PRINT INDEX,UTS$(INDEX),TAR(INDEX)
430 IF UTS$ (INDEX)="END" GOTO 450
435 INDEX=INDEX+1
```

```
440 GOTO 420
450 PRINT"■■■MODOSITAS(I/N)?"
452 GET E$:IF E$=""GOTO 452
460 IF E$="N"THEN RETURN
470 INPUT"■■■      INDEX    ";INDEX
480 INPUT"■      UTASITAS ";UTS$(INDEX)
485 IF UTS$(INDEX)="END" THEN RETURN
490 INPUT"■      IDOTARTAM  ";TAR(INDEX)
498 RETURN
500 REM INDITAS
501 INDEX=1
502 PRINT"■■■          ┌──────────────────────────────────────┐
504 PRINT TAB(6);"|";TAB(34);"|"
506 PRINT"          |    *** A PROGRAM FUT ***     |"
508 PRINT TAB(6);"|";TAB(34);"|"
510 PRINT"          └──────────────────────────────────────┘"
520 IF UTS$(INDEX)="END" THEN RETURN
530 IF UTS$(INDEX)="TUL"   THEN SUM=64 :GOSUB 1000
540 IF UTS$(INDEX)="TUR"   THEN SUM=128:GOSUB 1000
550 IF UTS$(INDEX)="KUP"   THEN SUM=16 :GOSUB 1000
560 IF UTS$(INDEX)="KDN"   THEN SUM=32 :GOSUB 1000
570 IF UTS$(INDEX)="TAK"   THEN POKE 56577,(PEEK(56577)+1)
580 IF UTS$(INDEX)="PUT"   THEN POKE 56577,(PEEK(56577)-1)
590 IF UTS$(INDEX)="GOT"   THEN INDEX=(TAR(INDEX)-1)
600 IF UTS$(INDEX)="PAU" THEN SUM=0  :GOSUB 1000
620 INDEX=INDEX+1
630 GOTO 520
1000 REM KESLELTETES-VEGREHAJTAS
1010 POKE 56577,(PEEK(56577)+SUM)
1020 FOR I=1 TO TAR(INDEX)
1030 NEXT I
1040 POKE 56577,(PEEK(56577)-SUM)
1050 RETURN
READY.
```

A program részletes ismertetése

10–20:

A háttér és keretszín beállítása

30:

A user port vezetékeit kimenetre állítjuk

40:

Az adatregiszterbe nullát töltünk, hogy előző tartalma törlődjön.

50–90:

Deklaráljuk az UTS$(x) és TAR)x) változókat, amelyekben később az utasítás-szavakat és az egyes utasítások időtartamát tároljuk.

Az UTS$ tömböt feltöltjük az „END" utasításszóval, a vezérlőprogram ugyanis ennek alapján ismeri fel a program végét.

100–240:

Kiírjuk a képernyőre a főmenüt a MEGHATAROZAS, MODOSITAS, INDITAS menüpontokkal.

250–270:

Várakozunk egy billentyű leütésére. A választott menüpont (1, 2 vagy 3) szerint ugrunk a megfelelő programrészre.

280:

Az alprogramból visszatérve ismét kiírjuk a képernyőre a főmenűt.

300–360:

A MEGHATAROZAS programrész. Az index kezdőértékét 1-re állítjuk, majd a CHR$(147) vezérlőkarakterrel töröljük a képernyőt, és a kurzort a képernyő bal felső sarkába mozgatjuk.

A 315-ös sortól a 360-as sorig terjedő ciklusban beolvassuk az utasításkészlet elemeit és az utasításokhoz rendelt időtartamokat. Az első ciklislépésben kiírjuk az utasítás sorszámát, amely minden lépésben eggyel nő. A vezérlő-program az utasításokat sorszámukkal azonosítja.

A beolvasást a 320-as, ill. 335-ös sorokba írtuk. A program nem ellenőrzi, hogy a beírt utasításszó benne van-e a szótárában. Erre adatbevitel közben a gépkezelőnek kell ügyelnie.

Az „END" szót a program utolsó utasításnak tekinti és a ciklust lezárva (330-as sor) visszatér a főprogramba.

Az időtartam beolvasása után (335-ös sor) a ciklusváltozót megnöveljük, kiírunk egy üres sort, és visszatérünk a ciklus elejére.

400–498:

A MODOSITAS alprogram.

Az alprogramban először kiírjuk a képernyőre az utasításszavakat és időtartamukat (420-as sortól a 440-es sorig terjedő ciklus), azzal a céllal, hogy módosítás előtt a gépkezelő áttekinthesse az utasításkészletet.

A 450-es sorban megkérdezzük, hogy a módosításra valóban szükség van-e. Ha a gépkezelő az N leütésével válaszol, visszatérünk a főprogramba. Egyébként a végrehajtás módosítással folytatódik.

A módosítás programrészben bekérjük a javítandó utasítás sorszámát, majd az új utasításszót. Ha válaszul a gépkezelő az END szót üti be, akkor a módosításból kilépünk.

Az utasításszó kijavítása után az új időtartamot is bekérjük.

A módosítások után visszatérünk a főprogramba és a képernyőn ismét megjelenik a menü.

500–630:

Az INDITAS alprogram.

Az alprogram A PROGRAM FUT szöveg kiírásával kezdődik, ugyanis minden további adatforgalom a user porton zajlik, aminek eredményeiről a képernyőn nem tájékozódhatunk.

Az INDEX változó értéke első lépésben 1 lesz, a következő IF utasítás tehát az UTS$(1) változó tartalmát vizsgálja. Ha az utasításszó ismeretlen, az INDEX értéke automatikusan nő eggyel, azaz rátérünk a második utasítás vizsgálatára.

Ha a vizsgált utasítás szerepel az utasításkészletben, akkor a hozzárendelt tevékenységet a program végrehajtja.

A különböző utasítások értelemszerűen eltérő tevékenységet eredményeznek.

A megszakítási feltétel itt is az END szó, hatására ismét visszatérünk a főprogramba.

Példaként elemezzük a KDN, TAK, PAU és GOT utasítások egymás utáni feldolgozását.

Amint a program felismeri a KDN utasítást a SUM változóba 32-t tölt és ugrik a 1000-es alprogramba. Az alprogram az adatregiszter pillanatnyi tartalmához hozzáadja a SUM változó értéket, jelen esetben 32-t. Ennek eredményeként az adatregiszter ötödik bitje 1 lesz, és így a robotkar lefelé mozog.

A mozgás időtartalmát az 1020-as és 1030-as sorok késleltető ciklusa határozza meg. A ciklus lefutása után az adatregiszter tartalmából ismét levonunk 32-t, majd visszatérünk a hívási ponthoz.

A TAK utasítás feldolgozása némileg eltér a KDN utasítás feldolgozásától. Az utasítás hatására az adatregiszter első bitjét 1-re állítjuk, és az állapot mindaddig változatlan marad, amíg a PUT utasítás vissza nem állítja az alaphelyzetet.

A be- és kikapcsolás közben folyamatosan feldolgozza a többi kijelölt utasítást.

A PAU utasítás az adatregiszter tartalmát nem módosítja, hatása mindössze egy késleltetés (1020-as és 1030-as sorok), amelynek időtartamát a TAR (INDEX) változó értéke szabja meg. Eközben a kar nyugalomban van.

A GOT utasítás sem az adatregiszter értékét nem módosítja, sem késleltetést nem vált ki.

Hatására csak az INDEX változó tartalma módosul, ami az utasítások feldolgozásának sorrendjét változtatja meg.

A többi utasítás feldolgozása a KDN-hez hasonló, attól csak annyiban tér el, hogy az adatregiszter értékét másképpen befolyásolja.

1000–1050:

Ez az alprogram hajtja végre az egyes utasításokat.

Ha a bemutatott program minden részlete világos, a tulajdonképpeni vezérlőprogram elkészítése rendkívül egyszerűvé válik.

Írjunk egy rövid vezérlőprogramot, amely hatására a robotkar áthelyez egy tárgyat egyik helyről a másikra.

```
 1  TAK
 2  KUP      1000
 3  TUR       800
 4  KDN       900
 5  PUT
 6  PAU       500
 7  KUP      1000
 8  TUL       770
 9  KDN       830
10  GOT         1
```

Mivel a robotkarba nem építettünk be visszajelzést, egészen biztos, hogy a program eredménye minden újabb kísérletben más és más lesz. Sőt a program eltérően fogja vezérelni a különböző motorral felszerelt robotkarokat.

A magyarázat a motorok eltérő futási adottságaiban rejlik. Következésképpen a programban szereplő állandókat minden robotkar esetében külön kell meghatározni.

Végül néhány hasznos ötlet:

Az elektromotor kikapcsolás utáni szabadonfutását erősen csökkenthetjük néhány izzó párhuzamos bekötésével. A dinamó-effektus ugyanis fékezően hat a motorra.

Ugyancsak megszívlelendő a következő javaslat: az elektromágnest célszerű egy ragasztószalaggal leragasztani, hogy maradjon egy kis hézag a mágnes és a tárgy között. Egyébként ugyanis a mágneses remanencia miatt előfordulhat, hogy a markoló kikapcsolás után sem engedi el a tárgyat.

5.3 Az egykártyás számítógép

Minden számítógép legfontosabb alapeleme a processzor, a ROM tár, a RAM tár és az input/output elemek.

A processzor a számítógép központi vezérlője. A processzor végzi el a számítógép legfontosabb műveleteit, az aritmetikai és logikai műveleteket, kezeli a tárat és kiszolgálja az input/output egységeket, és mindezt fantasztikus sebességgel teszi. Röviden a processzor a számítógép „igáslova".

A ROM tár több ROM és EPROM egységből épül fel, amelyekben előzetesen a gép operációs rendszerét tárolták. A RESET jel hatására a gép működése az operációs rendszer indításával kezdődik. Hogy az operációs rendszer a tár melyik területén helyezkedik el, az gépenként eltérő. Az operációs rendszer biztosítja mindazt a kényelmet, amelyet a számítógép használójának nyújtani képes.

Folyamatosan lekérdezi pl. a billentyűzetet, karbantartja a képernyőt stb.

Az operációs rendszer tehát a számítógép nélkülözhetetlen része. A ROM további területein találhatóak általában azok a hasznos programok és rutinok, amelyekre gyakran van szükség, és fontos, hogy gyorsan elérhetőek legyenek.

A ROM tár mellett, amelyből a felhasználó csak olvasni tud, a számítógép fontos tárterülete a RAM, amelybe írni is lehet.

A RAM területre még akkor is szükség van, ha célszámítógépről van szó, azaz olyan gépről, amely csak az előre beégetett programokkal dolgozik.

A fontosabb rendszerváltozókat (pl. a visszatérés címeket) ugyanis minden esetben tárolni kell. Nyilvánvalóan ha a gép használója az általa készített programokkal is dolgozni szeretne, akkor jóval nagyobb RAM területre van szükség mint az előző esetben.

A számítógép harmadik és nem kevésbé fontos részét jelentik az input/output elemek.

Input/output elemek nélkül a gép egy önmagába zárt egység, amely programokat dolgoz fel.

Az utóbbinak azonban semmi értelme nem lenne, ha a feldolgozás eredményeit a gép nem tudná közölni. Az I/0 elemek biztosítják a számítógép és a külvilág kapcsolatát. Rajtuk keresztül zajlik az adatforgalom a billentyűzet, a képernyő és pl. a lemezegység közvetítésével.

Minimálisan a leírt négy alapvető egység alkothat egy számítógépet. Ha mindezeket egyetlen lapra szerelték, akkor egykártyás számítógépről beszélünk.

Az egykártyás számítógép alapvetően különbözik az olyan számítógépektől, mint pl. a Commodore 64-es. Az egykártyás gépeket ugyanis általában vezérlési feladatok ellátására tervezik, és így számos ki- és bemenettel rendelkeznek, amelyeken keresztül különböző készülékekhez csatlakoztathatók. Ezeket a gépeket általában programozni is csak egy másik számítógépen keresztül lehet, vagy esetleg egy egyszerű hexadecimális billentyűzeten egy kijelzővel. Az egykártyás gépek programjait többnyire kényelmesen programozható nagyobb gépen fejlesztik és tesztelik, majd amikor a program elkészült, beégetik egy EPROM-ba, ami beépíthető a célszámítógépbe.

Ettől kezdve az egykártyás gép tudása bővül az EPROM-tartalmazta ismeretekkel. Az egykártyás számítógépek egyszerűségüknél fogva a robottechnika legfontosabb elektronikus eszközei. Az ilyen típusú számítógép számos I/0 vezetékével biztosítja, hogy a robot gazdag kapcsolatrendszert alakítson ki környezetével.

Tulajdonképpen sokak dolgát megkönnyítené, ha az egykártyás gépek a gépi kódú nyelven kívül valamely magas szintű programnyelven (pl. BASIC nyelven) is programozhatóak lennének, azonban ezzel a vezérlésre szánt gépeket a legfontosabb tulajdonságaitól fosztanánk meg. Meg kellene növelni a méretüket, le kellene mondani a gépi kódú programok gyors végrehajtási sebességéről stb. A vezérlésben épp az utóbbi tulajdonság az egyik legfontosabb, amiről semmiképpen nem szabad lemondani a kényelmes programozhatóság érdekében.

Az egypaneles gépek a vezérlés, és így a robottechnika rendkívül hasznos eszközei. Segítségükkel a robotok vezérlésének módosítása programozási kérdéssé egyszerűsödik. Ez olyan rugalmasságot biztosít a gyakorlat számára, amit semmiképpen nem szabad lebecsülni.

5.4 A látás és a hallás

A szenzorok minden változatukban a robotok legfontosabb részei, hiszen ezek teszik alkalmassá a külvilággal való kapcsolatteremtésre, és biztosítják, hogy a robotok bizonyos változásokra reagálni tudjanak.

Már a legegyszerűbb szenzorok, pl. a kapcsolók is számos értékes adatot képesek összegyűjteni a környező világról a robot számára.

A látás

Az infravörös szenzor – amellyel részletesen foglalkoztunk – az egyik példája annak, hogy hogyan tehetjük képessé a robotot optikai úton való tájékozódásra a környezetében. Az optikai érzékelést igen nagy túlzás lenne látásnak nevezni.

Éppen ezért most megismerkedünk egy olyan eszközzel, amellyel a számítógép ténylegesen olyan képeket nyerhet a közvetlen környezetéről mint az emberi szem.

Az elv, mint mindig, zseniálisan egyszerű: a dolog lényege egy módosított dinamikus RAM. Az IC-t megszabadítják tokjától, hogy a benne lévő chip szabadon hozzáférhető legyen. Ezt a chipet beépítik egy a fényképezésben használt lencse sugárterébe. A lencsével a környezetet leképezhetjük a dinamikus RAM-chip-re. A dinamikus RAM, hogy az információtartalmát ne veszítse el, bizonyos időnként ún. REFRESH (frissítő) impulzust kap a mikroprocesszortól.

Pontosan ezt a tulajdonságot tudjuk kihasználni jelen esetben. Megállapították, hogy a frissítő impulzus után a tár tartalma a chipet érő fényerőtől függően különböző idő alatt vész el teljesen, nevezetesen: erősen megvilágított helyen gyorsabban mint sötétben.

A megoldás tehát a következő: „fényképezés" előtt minden tárcímre 1-et írunk. Ezt követően a tár nem kap frissítő impulzust és így lassan elveszíti az információtartalmát. Ha egy bizonyos megvilágítási idő elteltével újra visszaolvassuk a tár tartalmát, a kevéssé megvilágított helyeken a korábban beírt értéket, a többi helyen pedig nullát kapunk.

Ha a sötét és világos részek közötti megkülönböztetéssel nem lennénk megelégedve, akkor egymás után többször kell a chipet megvilágítanunk, anélkül,

hogy közben újra egyeseket töltenénk bele. Így egy idő múlva valóban árnyalt képet kaphatunk.

Sajnos a hosszú megvilágítási idő (0.3 s-tól 0.7 s-ig) miatt mozgó tárgyat nem lehet ezen a módon megörökíteni, azonban a foltszerű felismerésre az eljárás kiválóan alkalmas.

Hogy a mesterséges szem látása milyen éles, azaz a kép felbontása milyen részletes, az a RAM tárkapacitásától függ, amelynek bittérképe adja a felbontás minőségét. Egy 64 kbyte-os RAM pl. 256 × 256-os képfelbontásra képes.

A dinamikus RAM belső felépítése miatt valójában sohasem egy teljes képet kapunk, hanem két félképet, amelyeket egy sötét sáv választ el egymástól. Ebben a sávban a gép nem tudja a külvilágot érzékelni. Ha valaki mégis teljes képet szeretne nyerni, ezért a felbontás minőségével kell fizetnie. Ugyanis a tárterületnek csak egy bizonyos részét használhatja optikai érzékelésre. 64 kbyte-os IC-vel pl. 32768 pontból álló összefüggő képet tudunk készíteni.

Amit elmondtunk az csak az elv. Ha valaki valóban „látóvá" szeretné tenni a számítógépet, annak nem javasoljuk, hogy dinamikus RAM-ot vásároljon. Az eredmény nem lépne túl a kísérletek kezdeti stádiumától várható eredményeken.

Erre a célra a kereskedelemben már számos jó minőségű, illesztővel ellátott készülék kapható.

Ez azonban még mindig nem nevezhető látásnak, hiszen a látáshoz a képalkotáson kívül még nagyon sokféle képesség – pl. az alak felismerésének képessége – tartozik.

A számítógépnek már a legegyszerűbb tárgyak felismeréséhez is rendkívül bonyolult szoftverre van szüksége.

Messze van még az az idő, amikor a gép megközelítheti az ember képességeit.

A hallás

Mielőtt a gépeket megtanítanánk hallani, meg kellene fejteni az emberi hallás mibenlétét. Nincs más kiindulópontunk, mint az emberi fül.

Aki valaha foglalkozott ennek az érzékszervnek a működésével, az tudja, hogy a fülbe érkező hang először a dobhártyába ütközik. A dobhártya felerősíti a hanghullámokat és mechanikus rezgésekké alakítja. A mechanikus rezgések a középfülben ismét mechanikus úton tovább erősödnek. Az ilyen módon átalakított hanghullámok a belső fülbe, a csigába jutnak. Itt a belső fül folyadékának segítségével idegingerré alakulnak. Így zajlik le a tulajdonképpeni hallás.

A hanghullámok magasságuk szerint más-más idegsejteket aktivizálnak a hallójáratban, melyek a kiértékelés eredményéről tájékoztatják az agyat. Ez a hallási folyamat egy gép esetében a következőképpen képzelhető el:

A hanghullámokat egy mikrofon fogadja és elektromos impulzusokká alakítja. A hullámok általában szinusz hullámok, és így ezeket számítógép nem tudja közvetlenül értelmezni.

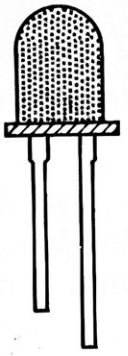
Az analóg jeleket a már jól ismert analóg/digitális átalakítóval a számítógép számára érthetővé kell tenni. Az AD átalakító a bemenő jelektől függő bitmintákat továbbít a számítógépnek. Éppúgy, mint a látás esetében, a leírt adatgyűjtést semmiképpen nem nevezhetjük hallásnak. A lényeg: a hangfelismerés ezzel még nincs megoldva.

Programozástechnikailag ez a következőképpen képzelhető el: a szavakat és mondatokat a hallókészüléken keresztül a számítógépben tárolni kell. Ha azt akarjuk, hogy a gép a szavakat kellő biztonsággal felismerje, akkor a tárolást többször meg kell ismételni. Ezáltal a gép képes lesz az eltérő hangzás ellenére felismerni az azonos szavakat.

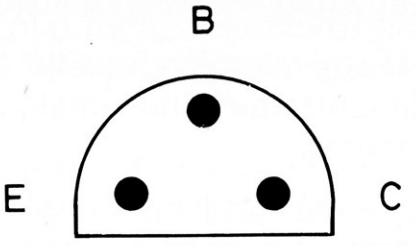
Persze az sem jó, ha ugyanazt a szót túlságosan sokszor, egymástól eltérő hangsúllyal megismételjük, hiszen ekkor előfordulhat, hogy az egyes szavak felismerési sávja túl széles lesz, és így a gép különböző szavakat azonosnak tekint. Arra mindenképpen célszerű ügyelni, hogy a különböző szavak a gép számára is világosan elkülöníthetőek legyenek.

A „tanfolyam" befejeztével joggal elvárhatjuk, hogy számítógépünk könnyedén felismerje hallás után a begyakorolt szavakat és szövegeket, és azokra az általunk meghatározott módon reagáljon.

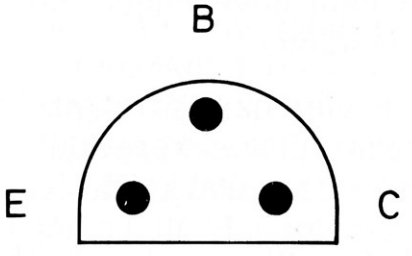
FÜGGELÉK



katód anód

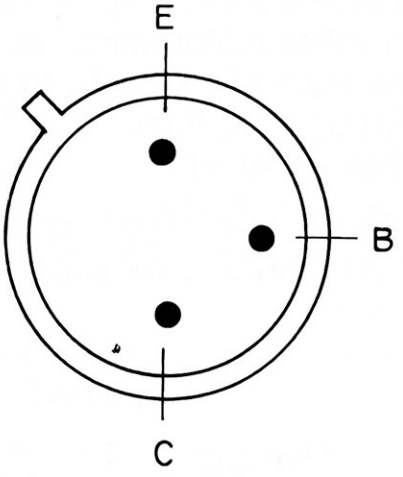


BC 337

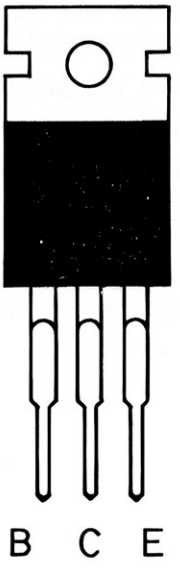


BC 238

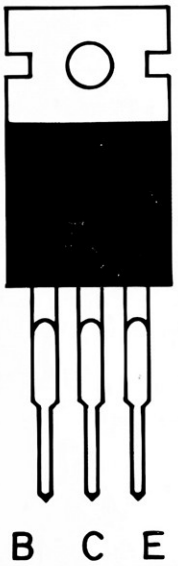
BC 109



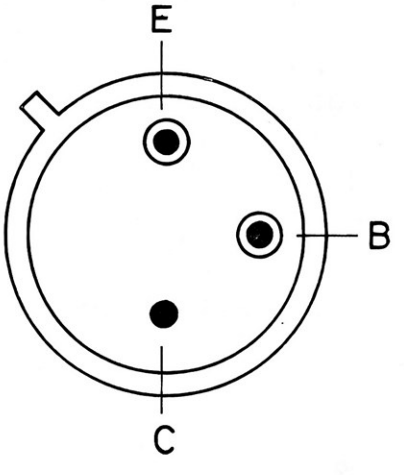
BD 240

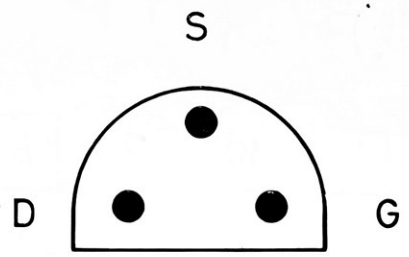


BD 239

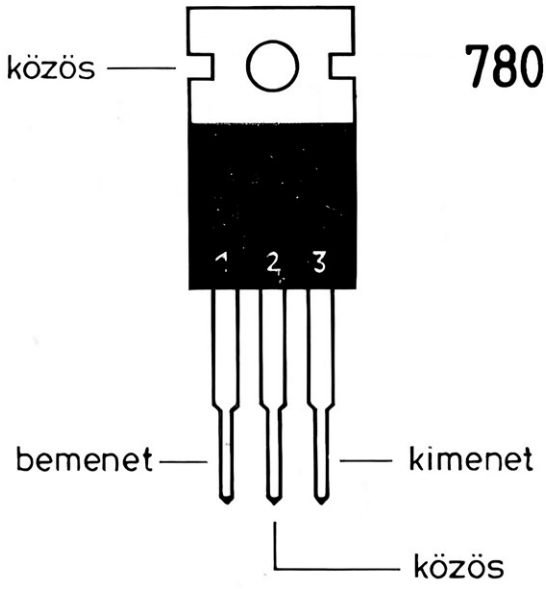


BC 141



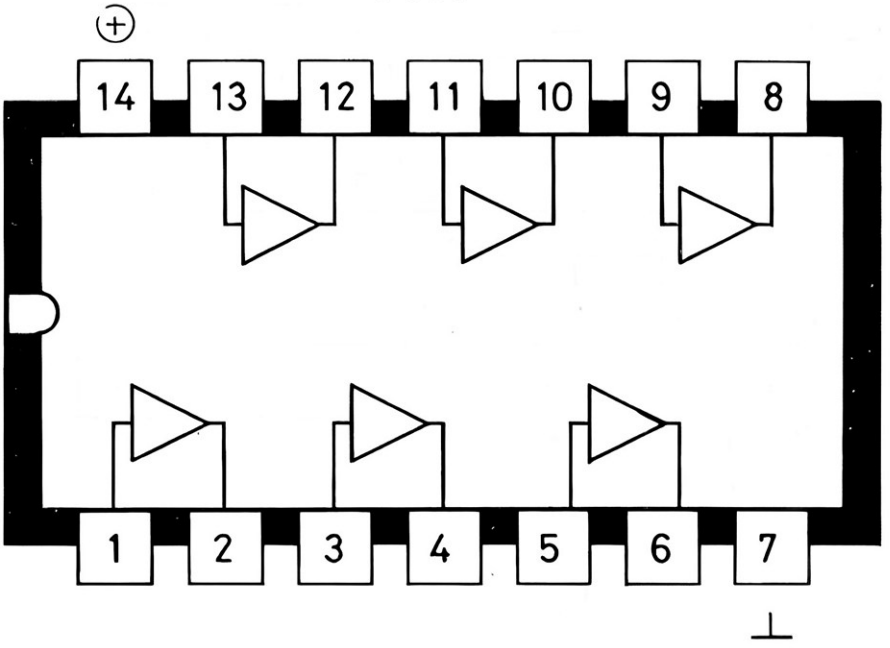


BF 245

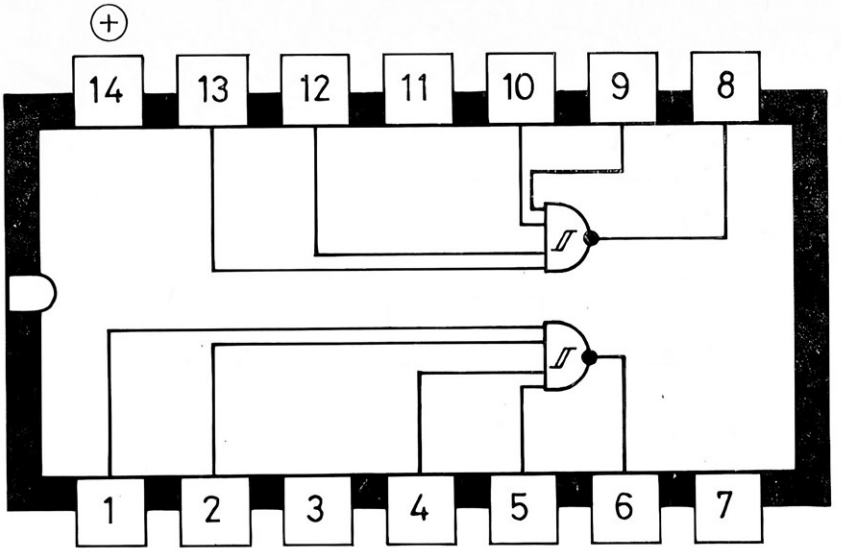


7805

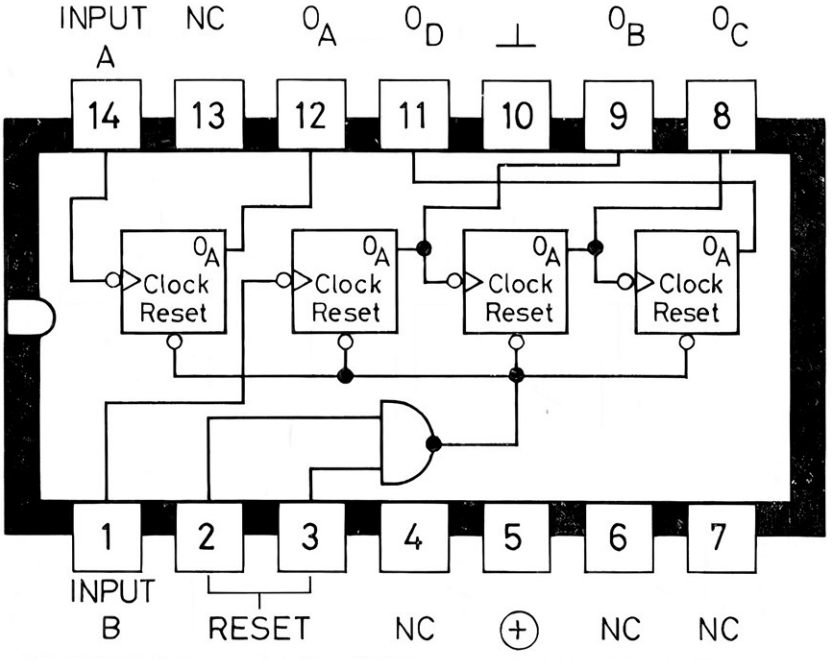
7407



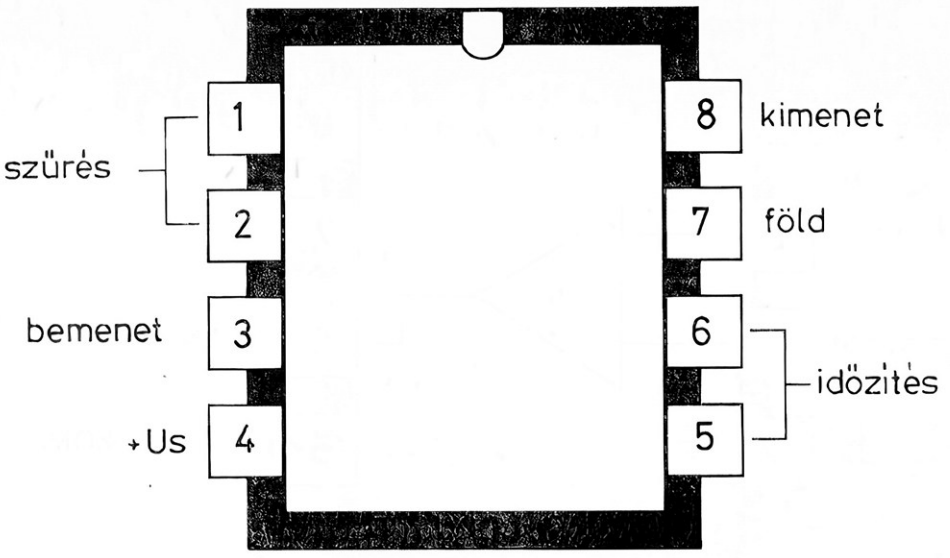
7413


⊕
14
13
12
11
10
9
8
1
2
3
4
5
6
7

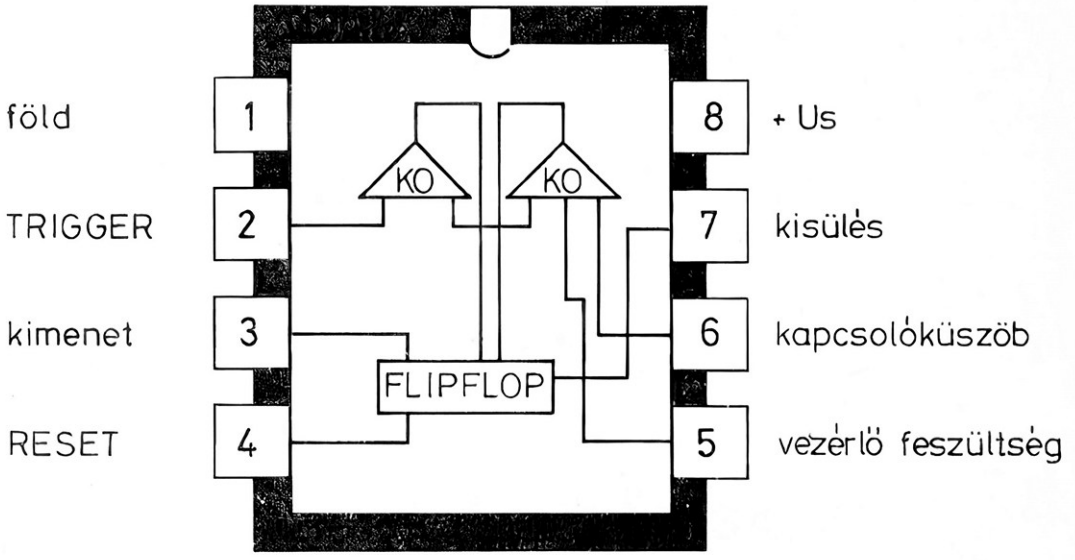
7493


INPUT A
NC
Q_A
Q_D
⊥
Q_B
Q_C
14
13
12
11
10
9
8
Q_A
Clock
Reset
1
2
3
4
5
6
7
INPUT B
RESET
NC
⊕
NC
NC

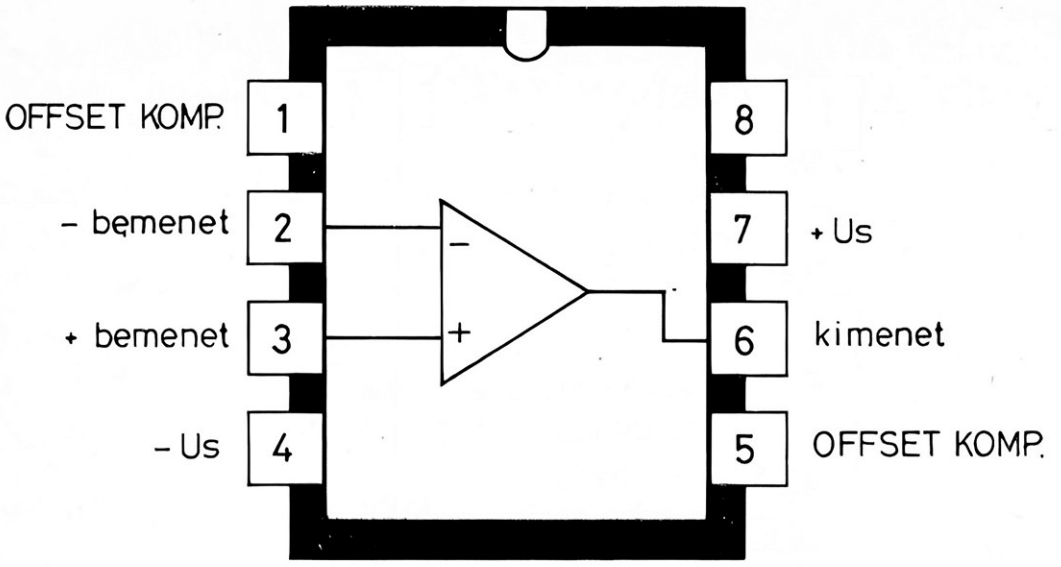
567


szürés
1
2
bemenet
3
+Us
4
8
kimenet
7
föld
6
időzítés
5

555


föld
1
TRIGGER
2
kimenet
3
RESET
4
KO
KO
FLIPFLOP
8
+ Us
7
kisülés
6
kapcsolóküszöb
5
vezérlö feszültség

741


OFFSET KOMP.
1
- bemenet
2
+ bemenet
3
- Us
4
8
+ Us
7
kimenet
6
OFFSET KOMP.
5
–
+

1987-ben megjelent
DATA BECKER-könyvek

A 68 000-ES MIKROPROCESSZOR
Technika és programozás

CAD
Bevezetés a számítógéppel segített
műszaki tervezésbe

AZ EPSON NYOMTATÓK KÖNYVE

COMPILER

SZÁMÍTÓGÉP ÉS SAKK

Ára: 249,– Ft

SZÁMÍTÁSTECHNIKA A KÖNYVESBOLTOKBAN

NOVOTRADE – 2 C ÁRUHÁZ
1136 Bp., Balzac u. 35. Tel.: 402-954

ÁLLAMI KÖNYVTERJESZTŐ V. – NOVOTRADE 2C

BUDAPEST

Táncsics Könyvesbolt
1073 Lenin krt. 17.
Telefon: 422–178

BUDAPEST

Műszaki Könyváruház
1061 Liszt Ferenc tér 9.
Telefon: 420-353

MŰVELT NÉP KÖNYVTERJESZTŐ V. – NOVOTRADE 2C

PÉCS

Zrínyi Miklós Könyvesbolt
7621 Jókai u. 25.
Telefon: 72-12835

VESZPRÉM

Kölcsey Ferenc
Könyvesbolt
8200 Cserhát út 7.

SZEGED

Tömörkény Könyvesbolt
6720 Lenin krt. 48.
Telefon: 62-21453

DEBRECEN

Szak- és ismeretterjesztő
Könyváruház
4024 Hunyadi u. 8.
Telefon: 52-23237

BÉKÉSCSABA

Radnóti M. könyvesbolt
5600 Tanácsköztársaság
út 2.
Telefon: 25-207

SZOLNOK

Szigligeti Könyvesbolt
5000 Ságvári krt. 35.
Telefon: 56-11133

SZOMBATHELY

Savaria Könyvesbolt
9700 Mártírok tere 1.
Telefon: 94-12341

GYŐR

Pattantyús Á. Géza Szak-
könyvesbolt
9021 Molnár Ferenc u. 9.

MISKOLC

Chip-kuckó
3530 Tanácsház tér 14.

KECSKEMÉT

Művelt Nép Könyváruháza
6000 Március 15. u.
Széchenyiváros
Telefon: 06-76-28157