BLUETOOTH KOMMUNIKÁCIÓ ÉS NXT

BLUETOOTH COMMUNICATION AND NXT

Tóth Attila János
Mechatronikai mérnök hallgató
Debreceni Egyetem, Műszaki Kar, Villamosmérnöki és Mechatronikai Tanszék
4032 Debrecen, Böszörményi út 68 E 4/A
attila.taj@gmail.com

Málnás Péter
Mechatronikai mérnök hallgató
Debreceni Egyetem, Műszaki Kar, Villamosmérnöki és Mechatronikai Tanszék
4079, Debrecen Fancsika Tanya 124/e
malnas.peter@gmail.com

Kivonat: Az emberiség a világ kezdete óta törekszik a gyors információáramlásra, mert az információ olyan tudás, ismeret, amivel a változásokra tudunk reagálni. A mi pályamunkánk, egy vezeték nélküli *adatcserére használt szabvány a Bluetooth. A világon, mindenhol használják, majdnem minden informatikai eszközben alapfelszereltségnek tekintik. Munkánk során, az NXT játékrobotokkal dolgoztunk, amelyeket az egyetemünk tett elérhetővét számunkra. Ha eltekintünk attól, hogy az NXT egy eredetileg játéknak szánt robot, mellyel csak modelleztünk egy irányítandó eszközt, akkor látszik az, hogy ezzel a technológiával bármit tudunk vezérelni. Így az iparban egy ilyen Bluetooth modullal, azt akármilyen gépbe integrálva távirányíthatunk, valamint adatokat kérdezhetünk le nagy sebességgel, viszonylag valós időben és viszonylag olcsón kialakítva. A Bluetooth platformfüggetlenségét is bizonyítjuk azzal, hogy az NXT robotot három különböző eszközzel irányítjuk: mikrovezérlővel, mobiltelefonnal és végül számítógéppel.*

Kulcsszavak: *Bluetooth, adatcsere, platformfüggetlenség*

IDEGEN NYELVŰ ÖSSZEFOGLALÓ: *Humanity has always been eager to transfer information fast since the beginning of time, as information is such knowledge, cognition that enables us to react on changes. Our competitive design is a wireless standard: the Bluetooth. It is widely used all over the world, considered almost serial in almost every IT device. During our work we used the NXT toy robot, which were made available for us by our university. If we disregard that the NXT, through which we only modelled the device to be remote controlled, is originally a robot for playing, we can see that it is possible to remote control eventually anything. So, with a Bluetooth module like this, integrated in any machinery, we can remote control, download data at high speed in relatively real time and with quite low costs to build. We even prove the platform independence of Bluetooth technology by controlling the NXT with three different utilities: microcontroller, mobile phone and finally computer.*

Keywords: *Bluetooth, datachanges, platform independence*

1. BEVEZETÉS

A mechatronikában önállóan működő, intelligens gépek tervezése a feladatunk. Gyakorlatban azonban többször előfordul olyan helyzet, amelyre a rendszer nincs felkészítve és külső beavatkozás szükséges. Az is előfordulhat, hogy valamilyen folyamat nem automatizálható és emberi irányítás szükséges a végrehajtásához. Erre a két problémára nyújthat megoldást a távvezérlés. A problémára ma már számtalan megoldás létezik: vezetékes technológiák (USB, soros- és párhuzamos kapcsolat, ethernet, vezetékes modem) és vezeték nélküliek (RF-modulok, infra, Bluetooth, wifi, GSM). A felsorolt eszközök közül a Bluetooth-t választottuk. Nem kell vezetékezni, a mozgás szabadabb, de az alacsony hatótáv ennek korlátot szab (kb. 10 m). További előnyei még a nagyfokú elterjedtség, platformfüggetlenség, valamint az integrálhatóság. Vezérelendő eszköznek a LEGO által forgalmazott MINDSTORMS NXT robotokat választottuk. A platformfüggetlenséget is bizonyítjuk azzal, hogy az NXT robotot három különböző eszközzel irányítjuk: mikrovezérlővel, mobiltelefonnal és végül

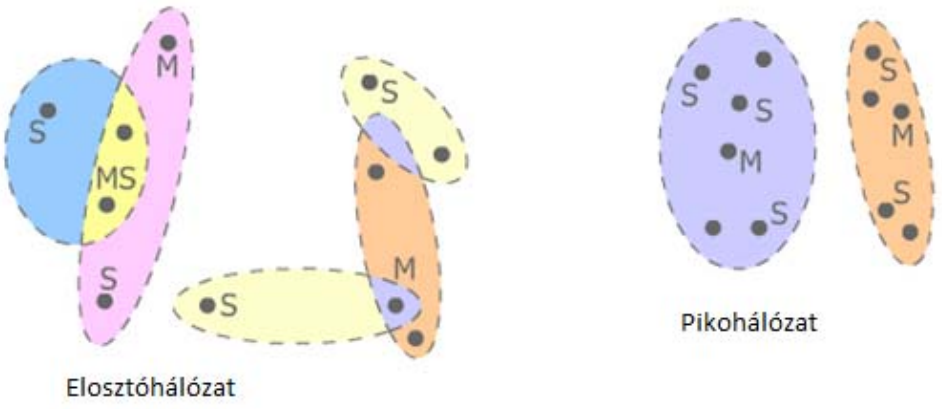
számítógéppel. A mikrovezérlős távirányító egy általunk megépített és felprogramozott mikrovezérlőt tartalmaz, amely egy Bluetooth modulon keresztül csatlakozik az NXT-hez és billentyűzettel irányíthatjuk a robotot. A mobiltelefonos távvezérlőnél az eszközbe beépített Bluetooth és gyorsulásmérő modult használjuk a robot mozgatására. Végül számítógépes vezérléssel, egy joystick-al irányíthatunk akár több NXT-t is egyszerre.

2. BLUETOOTH

A Bluetooth adatcserére használt vezeték nélküli szabvány. Frekvenciasávja a világszerte szabadon elérhető 2,4 GHz, amelyen belül, néhány ország kivételével (pl. Japán és Franciaország) 79 db 1 MHz-es csatornákon dolgozik (2,402-2,480 GHz). A kommunikáció ezeken a csatornákon véletlenszerűen ugrálva történik, ezáltal biztosított több eszköz egyidejű kommunikációja valamint a zavarszűrés is. Az adat átviteli sebessége az 1.2-es verziónál 1 Mbps, a 2.0-ásnál pedig 3 Mbps. [1]
A Bluetooth eszközöket három csoportra oszthatjuk hatótáv szempontjából:

1. osztály: 100 m
2. osztály: 10 m
3. osztály: 1 m

Minden eszköznek egyedi, 48 bites címe van. A Bluetooth kapcsolat mester-szolga alapú, azaz egy mesterhez egyidejűleg 7 szolga eszköz csatlakozhat. A szolga csak a mesterrel kommunikálhat és csak akkor, amikor a mester engedélyezi. Az így kialakuló hálózatot pikohálózatnak (piconet) nevezzük. A Bluetooth rendszer 10 pikohálózat egyidejű működését teszi lehetővé. Egy egység több pikohálózathoz is tartozhat, több hálózatban lehet szolga, de csak egy hálózatban lehet mester. Ezt úgy érheti el, hogy megváltoztatja a csatornaparamétereket, valamint ha szükséges, szerepet is vált (mesterről szolgára vagy fordítva).



1. ábra: Bluetooth hálózatok [1]

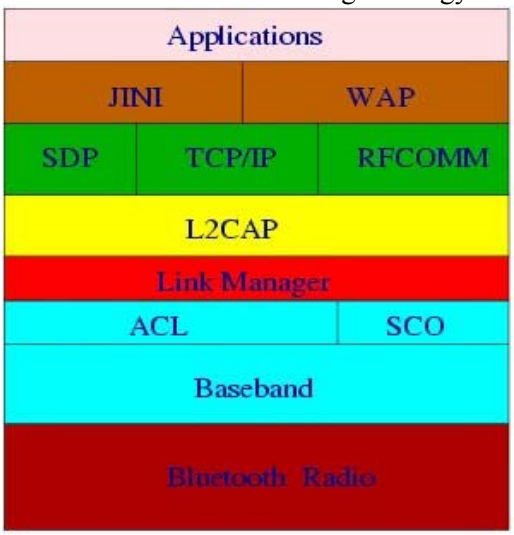
2.1. Bluetooth Alkalmazási profilok és protokolok

Az korábban leírtak csupán az adatátvitel fizikai megvalósítását határozzák meg. Az 1.1-es Bluetooth specifikáció a 2. ábrán látható 13 alkalmazási lehetőséget támogatja, ami azóta már tovább bővült.

Név	Leírás
Alap hozzáférés	Eljárások a kapcsolatok kezelésére
Szolgálatfelfedezés	Protokoll a felkínált szolgálatok felfedezésére
Soros port	A soros port kábelét helyettesíti
Alap objektumcsere	Ügyfél-kiszolgáló kapcsolatot definiál az objektumok mozgatásához
LAN-hozzáférés	Protokoll a mozgó számítógép és a rögzített LAN között
Betárcsázós hálózat	Lehetővé teszi a hordozható számítógépről mobiltelefonon keresztül indított hívásokat
Fax	Lehetővé teszi, hogy a mozgó faxgép faxot küldjön/fogadjon egy mobiltelefonon keresztül
Zsinór nélküli telefónia	Összeköti a kézi beszélőt a helyi bázisállomással
Interkom	Digitális kézi adó-vevő (walkie-talkie)
Fejhallgató	Lehetővé teszi a kéz használata nélküli beszélgetéseket
Objektumpumpa	Egyszerű objektumok cseréjét biztosítja
Állományátvitel	Általánosabb állományátviteli szolgáltatást biztosít
Szinkronizáció	Lehetővé teszi, hogy egy PDA szinkronizálja magát egy másik számítógéppel

2. ábra: Bluetooth alkalmazások [2]

Az alábbi ábrán látható a Bluetooth rétegeinek egymásra épülése:



3. ábra: Bluetooth protokollok [2]

Az alsóbb rétegekben találhatók azok a protokollok, amelyek minden alkalmazásban működnek (ezek a Bluetooth fizikai rétegei) pl: Baseband, ACL, SCO. Ezen belül a Link Manager protokoll gondoskodik az eszközök kapcsolódásáról, a titkosításról és a hitelesítésről. Az L2CAP (Logical Link Control and Adaptation Protocol) tartja a kapcsolatot az alsó és középső rétegek között. Feladata a Bluetooth sajátosságainak elrejtése, ezáltal támogatva a platformfüggetlenséget.

A középső rétegben találhatóak a speciális felhasználói alkalmazásokat támogató protokollok. Az SDP (Service Discovery Protocol) a környező Bluetooth eszközök szolgáltatásainak felderítésére szolgál, azaz kideríti hogy a 4. ábrán szereplő protokollok közül melyek érhetőek el az eszközön. Az OBEX bináris adatok átadására szolgál, amelyet például fájlátvitelre használhatunk. Erre más protokollok is épülnek. Az egyik legfontosabb az SPP (Serial Port Profile) amely virtuális soros port emulációra szolgál és az RFCOMM-ra (Radio Frequency Communication) épül. Erre is számos magasabb szintű protokoll épül és az NXT kommunikáció során, ezt fogjuk használni.

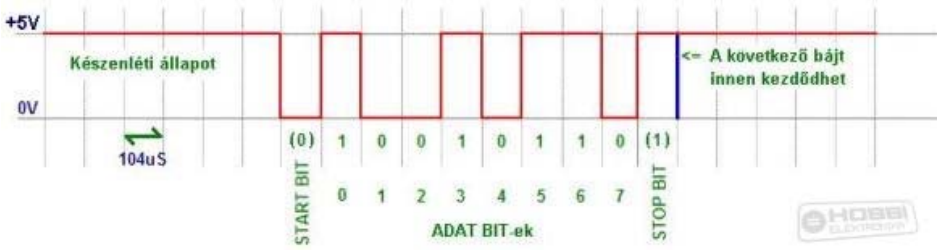
2.2. BTM-112 Modul

4. ábra: BTM-112 modul

A 6. ábrán látható egy 2.0-ás specifikációjú, másodosztályú (10 m hatótáv), SPP protokollt támogató Bluetooth modul. Működése hasonló az NXT-ben lévő modulhoz, azonban ez az eszköz egyszerre csak egy eszközzel tud kommunikálni az SPP protokoll miatt. Mind mester, mind szolga módban működik. A modul az áramkörhöz soros porton kapcsolódik. Ezen keresztül történik az eszköz vezérlése és az adatkommunikáció is. Bár ki van alakítva rajta az USB és az SPI port is, ezek csak firmware frissítésre és lekérdezésre szolgálnak. Táplálása 3,3 V-ról történik, de nem TTL jelszint toleráns, azaz nem működtethető 5 V-ról. Az eszköz vezérlése AT parancsokkal történik. Ezek ismertetése előtt bemutatjuk a soros kommunikáció alapjait, amelyet a mikrovezérlő és a Bluetooth modul között használtunk.

2.3. Soros kommunikáció

A soros kommunikáció alkalmával az átvinni szándékozott bájt bitjeit egymás után, sorosan juttatjuk el egyik eszköztől a másikhoz. Két fajtája ismert és használt: soros szinkron (SPI) [3] és soros aszinkron (UART) [4].
Az SPI kommunikáció mester-szolga alapú, a kommunikáció a mester eszköz által generált órajellel szinkronizált. A buszon több szolga eszköz is lehet, melyek közül a mester egy külön vezetéken engedélyezi az eszközt, amellyel éppen kommunikálni kíván. Ez a kommunikáció van implementálva az NXT-ben a Bluetooth modul, az LCD (szolgák) és a mikrovezérlő (mester) között.
Az UART kommunikációban két eszköz vesz részt, melyek egyenrangúak. Itt nincs szinkronizáló órajel, vagyis az adatátvitel értelmezése az időzítésen alapul. Az információcsere két vezetéken zajlik: minden eszköznek van RX és TX lába. Az RX láb az eszköz bemenete, a TX pedig a kimenete. Az első eszköz TX lábát a második eszköz RX lábára kell kötni és fordítva. A sikeres kommunikációhoz elengedhetetlen hogy mindkét eszköz azonos átviteli sebességre legyen beállítva, máskülönben a kommunikáció nem értelmezhető.



5. ábra: Soros adatátvitel [4]

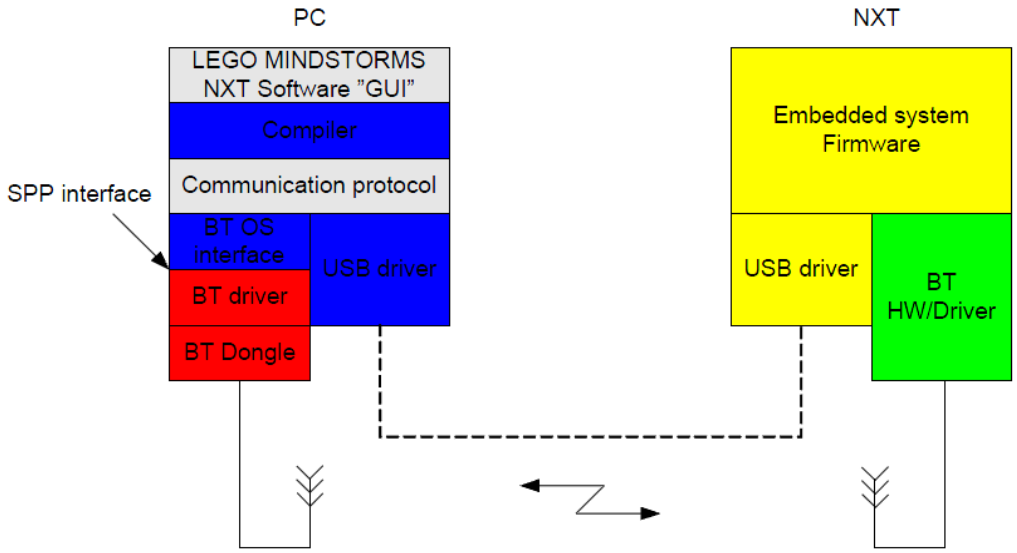
Vegyük például a 9600 bps-os (bit per secundum) sebességet. A kommunikáció itt TTL szinten történik, azaz a logikai 1-esnek 5 V, a 0-nak 0 V felel meg. Mivel a sebesség 9600 bps, egy bit elküldéséhez szükséges idő 1/9600, azaz 104 µs. Ha az egyik eszköz adatot kíván küldeni, a készenléti állapotból (5 V) a TX lábat 0 V-ra kapcsolja 104 µs-ig, majd elküldi az adatbiteket (esetünkben 8 db-ot). Ezután egy paritás bitet küld, amely az átvitt csomag hibaszűrését szolgálja (esetünkben ez nem használt) és végül egy stop bitet, amely lezárja a csomagot.
Ebből látszik, hogy a kommunikáció megkezdése előtt a két eszközt ugyanazokkal a beállításokkal kell ellátni (átviteli sebesség, bitek száma, paritás bit használata, stop bitek száma, stb.) különben a kommunikáció sikertelen lesz.

2.4. Bluetooth modul beállítása- AT parancsokkal

Beállítás után számítógépről kipróbáltuk a Bluetooth modul vezérlését. A modul soros kimenetét összekötöttük a PC soros kimenetével, azonban mivel a PC soros portja +12 V és -12 V-os jelszinttel működik, átalakító IC-t alkalmaztunk: MAX232 soros átalakító IC, amely a PC jelszintjeit TTL jelszintté alakítja és fordítva. A PC felől érkező TX lábon így 5 V-ot kaptunk magas jelszintnél, amelyet feszültségosztó ellenállássokkal csökkentettünk le, hogy a Bluetooth modul maximális 3,6 V-os feszültségértékét ne lépjük túl. Ezután már tudtunk kommunikálni számítógépről a modullal. Az eszköz adatlapjából megismerkedtünk az AT parancsokkal, azaz soros porton elküldjük az eszköznek az „AT" karaktereket, majd a kiválasztott utasítás betűjelét és paramétereit (pl: eszközök keresése: „ATF", csatlakozás eszközhöz: „ATA" stb.). A modult úgy kellett beállítani, hogy a soros kommunikáció paraméterei egyezzenek a mikrovezérlőjével, majd be kellett állítani a PIN kódot, és a modul szerepét a kommunikációban (mester).
A Bluetooth modul első indítása a gyári alapértelmezett beállításokkal történt. A modul szolga módban volt, a soros port sebessége 19,2 kbps. Ezt át kellett átállítani úgy, hogy a mikrovezérlővel képes legyen kommunikálni és csatlakozni az NXT-hez.
A soros port sebességét az „ATL1" paranccsal állítottuk 9600 bps-ra, „ATR0"-val állítottuk mester módba, „ATO1" paranccsal pedig kikapcsoltuk az automatikus csatlakozást.
Az „ATF" paranccsal indíthatjuk el a Bluetooth modul kereső funkcióját, amely szövegesen kilistázza a talált eszközök Bluetooth nevét és címét. Miután már ismert az eszköz Bluetooth címe, csatlakozni is tudunk hozzá. A modulnak az „ATD=…." paranccsal kiválasztjuk, hogy melyik eszközhöz szeretnénk csatlakozni. Ezt az eszközcímet a modul elmenti és az „ATA" parancs kiadása után csatlakozik hozzá. A legtöbb Bluetooth eszköz használ PIN kódot. Ezért csatlakozás előtt beállítjuk az eszközön a kódot az „ATP=…." paranccsal. A kapcsolat létrejötte után, a modul LINK lábára kapcsolt LED folyamatosan világít, ami a kommunikáció működését jelzi vissza. Mivel a BTM-112-es modul csak az SPP protokollt támogatja, olyan eszközökkel képes kommunikálni, amelyek szintén támogatják ezt a protokollt.
A Bluetooth egyik nagy előnye, hogy elfedi az eszközök közötti különbséget, azaz platformfüggetlen. Munkánk esetében ez abból látszik, hogy a mikrovezérlő és a BTM-112 modul 9600 bps-os sebességel, míg az NXT és a BlueCore modul 19200 bps-al kommunikál. Az eszközök pufferelik az adatokat, így a sebességkülönbség miatt nincs információvesztés.

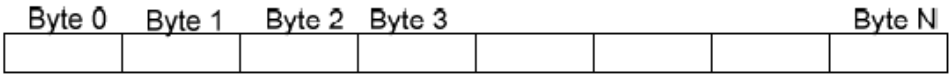
3. KOMMUNIKÁCIÓS PROTOKOL

A 6. ábrán látható az NXT és egy külső eszköz (jelen esetben egy PC) között megvalósuló kommunikáció [5].



6. ábra: NXT-PC kapcsolat [5]

A PC legfelső rétege a fejlesztőkörnyezet. Ebben írhatjuk meg az NXT-n futó programokat. A programok különböző programnyelven írhatók meg: NXT-G, LabVIEW, RobotC, NXC, NBC, Java, stb. A fordító szorosan kapcsolódik a fejlesztőkörnyezethez. Ez állítja elő az adott nyelven megírt forráskódból a gépi kódot, amit bele kell írni az NXT memóriájába. A következő réteg feladata ezt a gépi kódot átjuttatni a PC-ről az NXT-re. Az áttöltés történhet vezetéken keresztül (USB) vagy vezeték nélkül (Bluetooth). A következőkben a kommunikációs protokoll szerkezetét mutatjuk be.



7. ábra: Közvetlen parancs formátum [5]

A kommunikáció a 7. ábra szerinti csomagokból áll össze. Az első bájt (Byte 0) határozza meg, hogy az aktuális csomag rendszer- vagy közvetlen parancs. Ezen felül ez a bájt tartalmazza, hogy az adott parancsra a küldő eszköz vár-e választ, valamint azt is, hogy ez a csomag egy előző csomagra küldött válasz-e.
A bájtok hexadecimális számok formájában vannak bemutatva, a „0x” előtag jelzi ezt.
Az első bájt a következő öt értéket veheti fel:

- 0x00: Közvetlen parancs, választ vár
- 0x01: Rendszerparancs, választ vár
- 0x02: Válaszcsomag
- 0x80: Közvetlen parancs, válasz nélkül
- 0x81: Rendszerparancs, válasz nélkül

A távirányítók a közvetlen parancsokat használják.
A második bájt (Byte 1) tartalmazza magát a parancsot. A következő bájtok a parancshoz tartozó paraméterek. Ezeknek a száma attól függ, hogy melyik parancsot választjuk. Egy parancs maximálisan 64 bájtot tartalmazhat összesen. Erre a megszorításra azért van szükség, mert az NXT USB-s puffere ennyi adatot tud fogadni egyszerre. Ha a parancsot Bluetooth-on keresztül küldjük, a csomag elé még két bájtot kell csatolnunk, ezt azonban a 64 bájtba nem kell beleszámolni. Az első bájt a csomag hosszát tartalmazza (a csomagban szereplő bájtok darabszámát értve ez alatt, ezen a két bájton kívül), a második bájt mindig 0, mert a maximális csomaghossz egy bájton is ábrázolható.
Egy közvetlen parancs felépítésekor mindig a LEGO által kiadott dokumentációból [5] indultunk ki. Ez tartalmazza a bájtok számát, jelentését és lehetséges értékeit.

A következő példákon keresztül mutatjuk be, hogy hogyan lehet egy közvetlen parancsot felépíteni:

Kimeneti portok beállítása: SET_OUTPUT_STATE
Ezzel a paranccsal a szervo motorokat lehet szabályozni. A parancs 12 bájtból áll. Mivel a parancsok kiküldése Bluetooth-on fog történni, nem szabad elfelejteni a plusz két bájtot, amely a csomag hosszát takarja. Az **első** bájt tehát 0x0C, a **második** 0x00 lesz. A parancs **harmadik** bájtja jön, amely a csomag típusát hivatott eldönteni. Ebben az esetben ez közvetlen parancs lesz, választ nem kérünk az NXT-től, tehát 0x80. A **negyedik** bájt 0x04, ez maga a parancs, ezzel tudatjuk az NXT-vel, hogy a kimeneti portok állapotát kívánjuk beállítani. Az **ötödik** bájt a port sorszámát takarja (0x00 az A portot, 0x01 a B portot, 0x02 a C portot, sepicálisan a 0xFF az összes portra alkalmazza a következő paramétereket). A robot kanyarodását a két port különböző sebességre állításával érjük el, ezért a két portot külön parancsban kezeljük. A **hatodik** bájt a sebességet takarja. Értéke -100-tól 100-ig terjed, az előjel a forgási irányt határozza meg, a szám pedig a sebesség nagyságát jelenti százalékban. **Hetedik** bájt, a módbájt, ezzel mondhatjuk meg, hogy kívánunk-e bármilyen szabályozást a motorok forgására nézve (motorok szinkronizálása, elfordulás mérés, stb.). Ennél az egyszerűség kedvéért a 0x01-et választottuk, azaz nincs semmiféle szabályozás, egyszerűen csak elindítja az adott porton lévő motort. Az ezt követő bájtok a szabályzásra vonatkoznak. Ezeket nem használtuk, ezért erre nem térnünk ki, a tizedik bájt értéke 0x20, a **többi** mind 0.

A bemutatott paranccsal azt értük el, hogy a megadott portra kötött motor elkezd forogni a megadott irányba a megadott sebességgel és mindaddig nem áll meg, amíg nem adunk ki egy ugyanilyen parancsot, melyben a sebesség értéke 0.

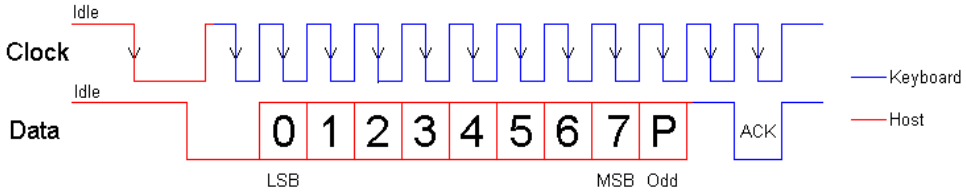
Elemfeszültség lekérdezése: GET_BATTERY_LEVEL
Hasonlóképpen, mint az előző parancsnál, az **első két** bájt a parancs hossza. Most 2 bájtból áll a csomag, vagyis az első két bájt 0x02, 0x00. A **harmadik** bájt a csomag típus bájtja, ami a közvetlen parancs válasszal: 0x00. Nyilván nem küldhetjük a 0x80-as csomagtípussal, mert minket éppen a válaszcsomag érdekel. Az **utolsó** bájt a parancs bájtja, 0x0B. Ez a parancs nem tartalmaz semmilyen paramétert. A válasz csomag hasonló szerkezetű ehhez, legalábbis a fejlécben. Az **első két** bájt a 0x05, 0x00, vagyis ezeken kívül még öt bájtot kell fogadni, **harmadik** a 0x02, vagyis válaszcsomag, **negyedik** a 0x0B, az előző csomagban szereplő parancs kódja. Ezt követi egy **státusz bájt**, melynek értéke 0x00 szerencsés esetben (ez azt jelenti, hogy nem történt hiba), minden más esetben hiba történt. A hiba oka felderíthető, ha a státusz bájt értékének jelentését kikeressük a dokumentációban lévő táblázatból. Az **utolsó két** bájt pedig a mért elemfeszültséget takarja milli voltban. A két bájtot bitműveletek segítségével össze kell fűzni, az így kapott előjel nélküli egész szám a mért feszültség.

4. NXT VEZÉRLÉSE TÁVIRÁNYÍTÓVAL

A legnagyobb kihívást a mikrovezérlős távirányító megépítése jelentette, ugyanis nagyon sok részfeladatot kellett összehangolni egy áramkörben. Először is biztosítani kellett a tápellátást, 5 V-ot a mikrovezérlőnek és az LCD modulnak valamint 3,3 V-ot a Bluetooth modulnak. Meg kellett oldani a feszültség szintillesztést a Bluetooth modul és a mikrovezérlő között.
További feladatot jelentett az áramkör bedobozolása, és a mikrovezérlőben futó program megírása.

4.1. Billentyűzet kezelése mikrovezérlővel

A távirányító készítésekor legelső részfeladat az NXT motorjainak elindítása volt. Miután sikerült, rájöttünk, hogy a közvetlen parancsokkal elérhető funkciók száma sokkal nagyobb, mint ahány szabad bemenet maradna a mikrovezérlőn. Körülményes lett volna saját nyomógombpanelt létrehozni és vezetékezni, ezért döntöttünk úgy, hogy PS/2-es billentyűzetet fogunk használni a vezérléshez.
A 8. ábrán látható a PS/2-es billentyűzet kommunikációja:

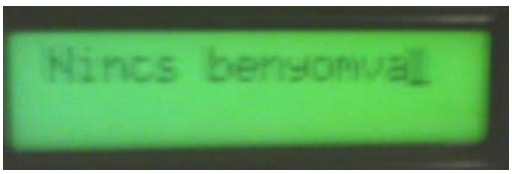


8. ábra: Kommunikáció billentyűzettel

A PS/2 billentyűzet kommunikációja hasonlít a soros szinkron kommunikációhoz. Egy billentyű lenyomásakor a CLOCK vezetéken a billentyűzet 11 órajelet generál. A kommunikáció egy start bittel kezdődik, vagyis a DATA vezeték 0 V-ra van húzva. A DATA vezetéken lévő feszültségértéket akkor kell leolvasni, amikor az órajel magas állapotból alacsonyra vált. Ezután következik 8 adatbit, majd egy paritás bit és végül a stopbit. A kommunikáció kétirányú is lehet, azonban a távirányító vezérléséhez elég a billentyűzetről a mikrovezérlő felé történő adatátvitel.
A billentyűzet CLOCK lábát a mikrovezérlő külső megszakítás generáló lábára kötöttük. Amint megnyomunk egy gombot a billentyűzeten, a CLOCK lábon megjelenik az órajel. Ekkor az eső órajel hatására megszakítás generálódik a mikrovezérlőben, így a főprogram futása abbamarad és a megszakításkezelő rutin hajtódik végre. Ekkor nem csinálunk mást, mint a DATA lábon lévő értéket (bitet) elmentjük egy változóba. Egy globális változóban sorszámozzuk a megszakításokat, így a rutin tudja, hogy a meghíváskor hányadik megszakítás történt és azt is tudja, hogy a beérkező új bit az adatbit hányadik bitje. A paritás bitet, a start és a stop bitet figyelmen kívül hagyjuk.

Ha lenyomunk egy billentyűt, a billentyűzet folyamatosan küldi a gomb kódját a mikrovezérlőnek. Egyes speciális gomboknak több kódja van. Például a négy irányba mutató nyilakat ábrázoló gombok alaphelyzetben egy kódot küldenek. Ha megnyomjuk a NumLock gombot, a billentyű felengedésekor a billentyűzet egy ún. „break” kódot küld. Ezt felhasználtuk a robot mozgatásakor, úgymond beragadó billentyű funkcióként.

4.2. Távirányító funkciói



9. ábra: A Végállás érzékelő állapota I

10. ábra: A Végállás érzékelő állapota II

11. ábra: Ultra hangos érzékelő működés közben

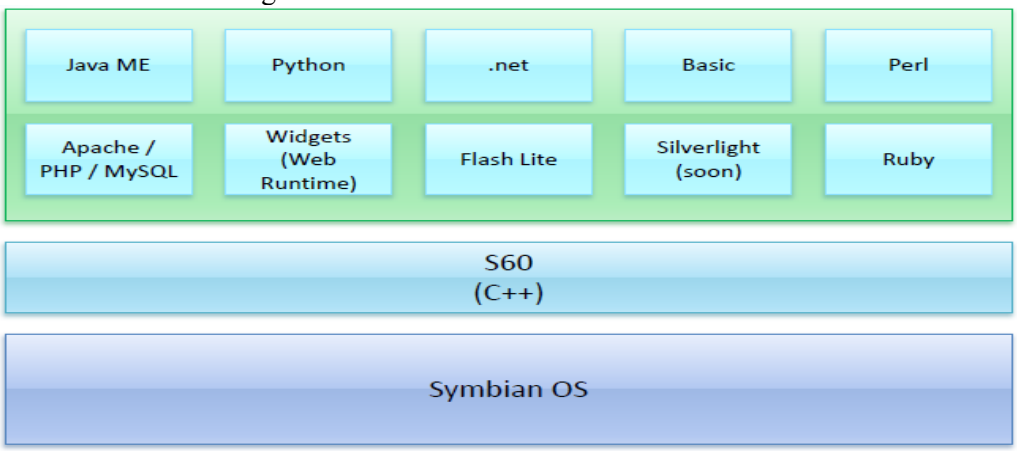
12. ábra: Elem feszültség lekérdezés

Miután csatlakoztunk az NXT-hez, billentyűzet segítségével irányíthatjuk azt. A négy kurzormozgató billentyű segítségével mozgathatjuk a robotot. A programszerkezet kialakítása miatt egyidejűleg csak egy gombot érzékel a mikrovezérlő, így az íves mozgások nem megvalósíthatók (például előre és jobbra). Alaphelyzetben, ha lenyomjuk az egyik kurzormozgató billentyűt, felengedés után a robot nem áll meg, úgymond „beragad” a billentyű. Megállítás bármely egyéb gomb lenyomásával érhető el. Ez akkor hasznos, ha hosszasan egy irányba akarjuk terelni a robotot. Amennyiben ezt a funkciót nem akarjuk használni, a „NumLock” billentyű megnyomása után a robot a kurzormozgató gomb felengedéséig mozog. A sebesség alaphelyzetben 100%, a numerikus rész „+” és „-” billentyűivel szabályozható. Az „U” billentyű lenyomása az ultrahang szenzorral való távolságmérést indítja el. Az eredményt a távirányító LCD kijelzőjén olvashatjuk le, ahogy az a 20. ábrán látható. Amennyiben a mérendő távolság meghaladja a szenzor által mérhető maximális távolságot, az LCD-n az „érvénytelen” szöveg olvasható. A „G” gomb lenyomásával lekérdezhető az ütköző állapota: benyomott esetben a kijelzőn „benyomva” szöveg, egyébként a „nincs benyomva” szöveg olvasható a 9. és 10. ábra szerint. A „B” gomb lenyomásával az elemfeszültség kérdezhető le. A feszültség nagysága ugyancsak az LCD-n jelenik meg, 12. ábra mutatja. Az „S” gomb lenyomása egy sípolást eredményez. A „PrintScreen” gombbal kapcsolgathatjuk az analóg színérzékelőn elhelyezett három

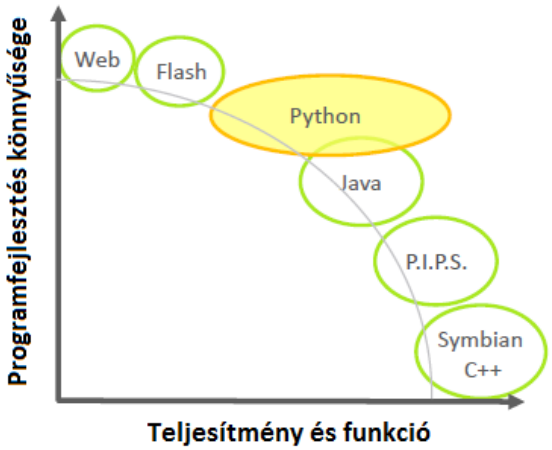
LED-et. Első lenyomásra a piros, másodikra a zöld, harmadikra a kék, majd mindhárom egyszerre világít, végül minden LED-et kikapcsol.

5. NXT VEZÉRÉLSE MOBILTELEFONNAL

Az NXT vezérlését egy Nokia gyártmányú N95, Symbian operációs rendszerű, S60-as platformmal, rendelkező mobiltelefonnal végeztük el.



13. ábra: Symbian OS



14. ábra: Symbian alatt használható programnyelvek

A Symbian operációs rendszer egyfelhasználós, grafikus, többfeladatos rendszer. [7] Előnye az operációs rendszerrel nem rendelkező telefonokkal szemben, hogy fejleszthető szoftverekkel. A Symbian a Nokia által fejlesztett keretrendszerrel rendelkezik. A rendszert C++ nyelven fejlesztették, ezáltal a leghatékonyabb programok is ezen a nyelven írhatók meg. Ezen felül futtathatók rajta Java nyelven, valamint Python nyelven íródott programok is. A Symbian C++ nyelv az ANSI C++ -hoz képest nagyon eltérő szerkezetű [8]. Szigorú névadási szabályok jellemzik, hiányzik a kivételkezelés, helyette egy nehézkesen érthető ún. „leave”-elést megvalósító függvényeket kell létrehozni, a dinamikus memória-felszabadítás hiányzik (garbage collector), helyette a felhasználóra hárul a már nem használt memóriaterületek felszabadítása (stack pointer segítségével). A memória-felszabadítás problémája nagyon fontos az ilyen, viszonylag kevés memóriával rendelkező eszközök esetében, hisz egy mobiltelefon sokkal ritkábban van újraindítva, mint egy asztali számítógép, ezért egy rosszul

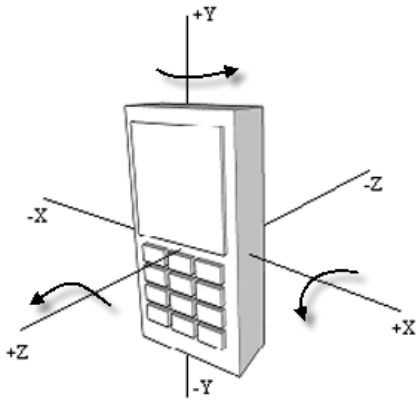
megírt program nagyon lelassíthatja az eszközt. Mindezekből következik, hogy a Symbian C++ nyelven történő programozás gyakorlott és tapasztalt programozókat kíván.
A 14. ábrán láthatók a programnyelvek teljesítmény és nehézségi fok alapján csoportosítva. Itt látható, hogy a Symbian C++ a leghatékonyabb és egyben a legnehezebben elsajátítható. Ehhez képest a Python nyelv teljesítmény szempontjából elmarad, azonban tanulhatósága és a fejlesztés gyorsasága jelentősen nagyobb, mint az előbb említetteké.

5.1. Miért Python?

A Python egy objektumorientált, script alapú, kivételkezelést támogató, dinamikus adattípusokat használó, hatalmas könyvtárkészlettel rendelkező, egyszerű szintaktikájú nyelv [9]. A Symbianos, S60-as szériákon használható script-értelmező alkalmazást PyS60-nak nevezik. A programot egy egyszerű szöveges szerkesztőben lehet megírni, amit „ .py” kiterjesztéssel kell elmenteni, majd a telefonra feltelepített Py S60-as alkalmazással lehet megnyitni.
A programon belül valós időben, a telefon billentyűzetét használva is lehet interaktívan programozni, vagy ezeket az előre megírt script-fájlokat futtatni. Egyik nagy hátránya, hogy a szintaktikai hibák csak futási időben derülnek ki, hisz a program soronként értelmezi, fordítja és hajtja végre az utasításokat.

5.2. Gyorsulásmérő

Az N95-ös mobiltelefonban található egy beépített gyorsulásmérő. Ennek segítségével a telefon érzékeli, hogy milyen helyzetben áll a térben. Úgy döntöttünk, hogy egy kicsit rendhagyó módon irányítjuk a robotot, amihez egy kis kézügyesség is szükséges. A három tengely közül csak az X és Y tengelyeket fogjuk használni. Ha a telefon vízszintes helyzetben van, akkor ezeknek, a koordinátáknak az értékei nullák.



15. ábra: N95 gyorsulásmérő tengelyei

5.3. A gyorsulásérzékelős program

A Python szotvernek rengeteg előre megírt könyvtára van. Ezek közül a gyorsulásmérőhöz tartozó „axyz” és a Bluetooth-hoz kapcsolódó „socket” könyvtárat használtuk. A teljes program mindössze 53 sorból áll, ez mutatja a Python nyelv nagyszerűségét.
Nézzünk meg néhány részletet a teljesség igénye nélkül a programból:
„address, services = bt_discover()”
Ez a függvény a Bluetooth eszközök keresését indítja el. Előhozza a telefon Bluetooth eszközöket kezelő menüjét, majd a menüből kiválasztott eszközöknek a címét és a szolgáltatás portjának számát visszaadja a hívás helyére.
“nxt=socket(AF_BT,SOCK_STREAM)
nxt.connect((address,services.values()[0]))”

Létrehoztunk egy „socket” típusú objektumot, „nxt” néven. Paraméterként, megadtuk, hogy a Bluetooth-t használtuk: „AF_BT”, valamint, hogy folyamatos adatkommunikációt használtunk: „SOCK_STREAM”.
Miután létrehoztuk az objektumot, a „connect” tagfüggvénnyel csatlakozhatunk a kiválasztott eszközhöz. Paraméterként a metódus vár egy Bluetooth címet és egy portot. Ezeket a „bt_discover” függvénnyel már meghatároztuk.
Ezek után a program folyamatosan lekérdezi a telefon helyzetét az „axyz.connect(read_xyz)” függvénnyel, majd az „nxt.send(motor1)” függvénnyel elküldi a közvetlen parancsot az NXT-nek.
A “send” metódus egy szöveges változót vár paraméternek. Emiatt akadtak a program fejlesztése során nehézségek. A függvény ugyanis a beadott karakter ASCII kódját küldi el az NXT-nek, vagyis ha a közvetlen parancs első bájtja a 0x00, vagyis decimálisan 0 lenne, akkor a “send” függvény ennek a karakternek az ASCII kódját fogja kiküldeni, ami a 48. A probléma első kísérleti megoldása az volt, hogy kikerestük az ASCII táblából a karakterkódokat és az azoknak megfelelő karakterekből fűztük össze a szöveges változót. Ekkor merült fel még egy probléma, mégpedig az, hogy vannak olyan karakterek, amelyek nem adhatók meg billentyűzetről, mint például a 0x07-es, ami a számítógépen egy sípolást eredményez. Erre a megoldást a “chr()” függvény jelentette, amely a beadott kódból létrehozta a kódnak megfelelő karaktert. Az X és Y koordinátákból kiszámolt sebességeket is ezzel a függvénnyel alakítottuk át karakterré.
Egy másik probléma a negatív számok átküldése volt. Ha az X és Y koordinátákból számított sebesség negatív értékre jött ki, azt jelentette, hogy az aktuális motornak hátrafele kellene forognia. A „chr” függvény azonban csak pozitív számot fogad el argumentumnak, vagyis ki kellett találni, hogy ilyenkor az NXT mit ért negatív számon. Némi utánajárás után rájöttünk, hogy az NXT a bájt 8 bitje közül az elsőt előjelbitnek tekinti, hisz 7 bit elég a sebesség ábrázolására (2^7=128). Negatív szám esetén bitműveletekkel az első bitet 1-esre állítottuk, ezáltal a motor forgási iránya megváltozott.
A program fontos részét képezi az X és Y koordináták értékét vizsgáló szekvencia. Ez a rész feltételvizsgálatokból és egyszerű műveletekből áll, melyek eredményei meghatározzák a „motor1” és „motor2” szöveges változókba kódolt sebességértéket.
A bemutatott részekből látható, hogy a Python nyelv mennyire egyszerűen használható mobilalkalmazások fejlesztésére és hogy mennyire egyszerűen kezelhető vele a Bluetooth.

6. NXT VEZÉRLÉSE SZÁMÍTÓGÉPPEL

Ez a téma rendkívül szerteágazó, és rendkívül érdekes. Egy mechatronikai mérnők alapvető feladata a problémák felismerése és a problémák kiküszöbölése. A felhasználó minden esetben meg fogja találni az esetleges hibákat a futó programban. Ezért mindig felhasználó barát és 100 százalék hatásfokú programot kell létrehozni. Az általunk írt és futtatott programokat hosszas tesztelés és program „debug” után készítettük el.

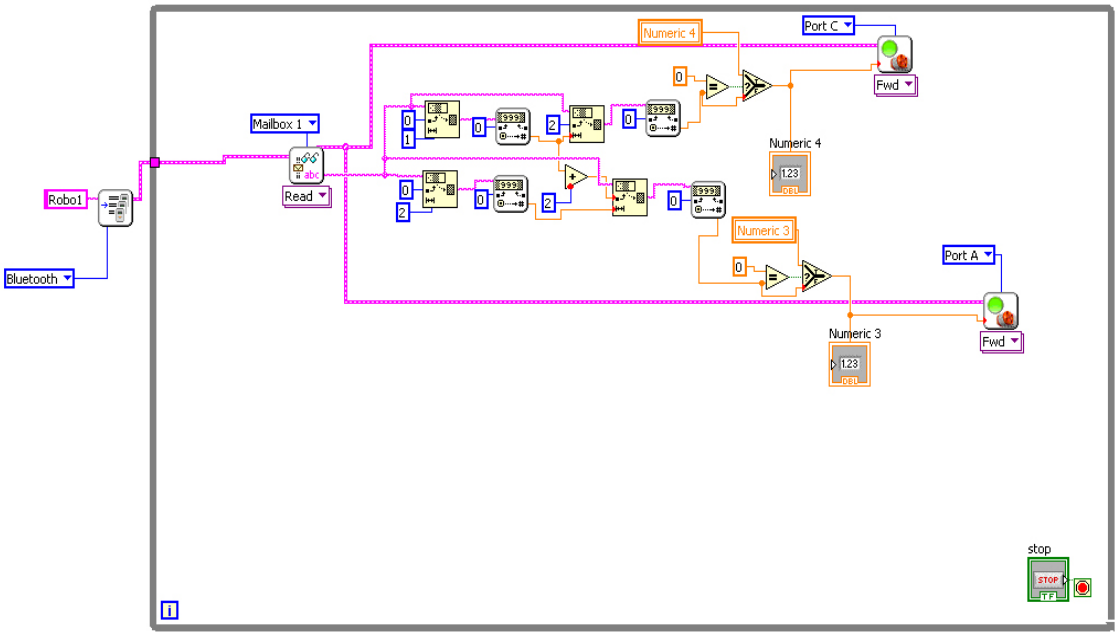
6.1. Választott programozási felületek

Az általunk választott programozási nyelv egy grafikus felületű program. A LabVIEW-t választottuk, mert a tanulmányaink folyamán már sikerült elsajátítani az alapvető ismereteket.
A karakteres felületű programozáshoz az NXC-t, C-re épülő programnyelvet választottuk. Ez a program nem csak a megértésben, de a programok működésében is nagy segítséget nyújtott.

6.2. NI LabVIEW 9.0

A NI LabVIEW egyre nagyobb segítséget nyújt a mérések, vizsgálatok terén. Az iparban egyre nagyobb szerepe van. Az oktatás terén is átütő sikerei vannak, mivel sok felsőoktatási intézmény alapkövetelményei között van az ismerete.
Az NI LabVIEW grafikus felületű programnyelv. Az alap programon túl használnunk kellett egy kiegészítő csomagot is. Ez a csomag beépül az NI LabVIEW 9.0-ba és lehetőséget ad nekünk az előre megirt NXT programok használatára. Az NI LabVIEW programban közel 700 darab matematikai

függvény van. Ezeknek a segítségével gyakorlatilag minden matematikai problémára tudunk megoldást találni. A webes csatlakozási lehetőségek gyors és ötletes megoldásokat mutathatnak be az esetleges hibakeresésnél. **A Master (mester) programját a számítógép processzor kihasználása érdekében az alap (blank) VI-ban kell létrehozni.** A program futtatása esetében az **NXT Shell nevű programot fogja futtatni a téglán a számítógép. Ez a számítógépes irányítást jelenti.** Itt tudunk beleszólni a programba a felhasználó által.
Az itt futtatott számításokat a számítógép végzi, így nem kell terhelni az NXT-t a számításokkal.
A második lehetőség az NXT objektum (Blank VI targeted to NXT) kiválasztása. Ezzel a lehetőséggel fogunk dolgozni a segéd (slave) program megírásához. Ebben az esetbe nem kell csak a kész programot futtatni az NXT-n.



16. ábra: Joystick Slave (segéd) program

6.3. NXT vezérlése USB-s eszközzel

Sokaknak eszébe jutott, hogy több robot azonos időben azonos mozgásokat hajtson végre. Ennek a vezérlésnek legnagyobb problémája az átviteli sebesség. Ez a vezérlés nem csak látványos de nagy teljesítménynövekedést is eredményezhet a gyárakban. Abban az esetben, ha egy olyan programot vagyunk képesek létrehozni, amelyik kiszűri a hibákat és még az anyagmozgatást is átláthatóvá és ütemessé teszi, akkor létrehoztunk egy jól működő, jó logisztikájú, versenyképes termelésű gyárat.

7. ÖSSZEFOGLALÁS

Dolgozatunkban bemutattuk, hogyan tudtuk használni különböző eszközök Bluetooth-ait. Rávilágítottunk, hogy a Bluetooth nem platformfüggő, elterjedt, tudásához képest nem is drága és viszonylag könnyen kezelhető. A dolgozatban bemutatott alkalmazásokban csupán az SPP profilt használtuk, a Bluetooth azonban ennél jóval többet tud. Ezzel az egy protokollal is az NXT összes funkcióit elértük. Ha eltekintünk attól, hogy az NXT egy eredetileg játéknak szánt robot, mellyel csak modelleztünk egy irányítandó eszközt, akkor látszik az, hogy ezzel a technológiával bármit tudunk vezérelni. Így az iparban egy ilyen Bluetooth modullal, azt akármilyen gépbe integrálva távirányíthatunk, valamint adatokat kérdezhetünk le nagy sebességgel, viszonylag valós időben és viszonylag olcsón kialakítva. A tapasztalt problémák nagy része a Bluetooth átvitel jellege miatti késés. A mikrovezérlős távirányító folyamatosan „elárasztja” parancsokkal az NXT robotot, vagyis a kommunikáció folyamatos. Mivel az AVR mikrovezérlő lassabban kommunikál a Bluetooth modullal, mint az NXT BlueCore modulja az ARM mikrovezérlővel, ezért az NXT puffere nem telítődik túl. A

folyamatos utasítás miatt azonban az NXT folyamatosan értelmezi az adatokat, vagyis jóval nagyobb áramot vesz fel. Ezáltal az akkumulátor is hamarabb lemerül. Küldő oldalról nézve a fogyasztás nem jelent problémát, mert a tápot a hálózatról kapja a távirányító. Kisebb késés figyelhető még meg a billentyűk értelmezése miatt. Az eredeti tesztpanelen egy gyorsabb mikrovezérlőt használtunk, de ez a késés alig észlelhető. Néha előfordul, hogy a Bluetooth modulnak több időre van szüksége a kapcsolat létrehozásához, mint általában. Mivel a mikrovezérlő nem figyeli a Bluetooth modul állapotát, ilyen esetekben elkezdi az adatküldést, és emiatt úgymond „elcsúsznak" a parancs bájtjai. Ilyenkor a legjobb megoldás az újraindítás. Ezek a problémák szinte mind orvosolhatók, fejlesztések után ki lesznek javítva.
A telefonos távirányítás viszonylag hibamentesnek mondható. A Python nyelet csak kismértékben ismertük meg ezért még nagyon sok dolgot lehet finomítani rajta.
A legtöbb probléma a PC-s távirányításnál jelentkezett. Mivel a robotok vezérlése egy USB-n a PC-hez csatlakozó NXT tégla Bluetooth-án keresztül történik, jelentős késések tapasztalhatóak. A LabVIEW kiegészítő csomagja magas szintű függvényeket és eljárásokat tartalmaz, ugyanakkor ezek lassúak is. A joystick kezelése is nehézkes LabVIEW alól, a tengelyek értékei pontatlanok, és középállásban nem nullázódik ki az értéke.

8. FEJLESZTÉSE LEHETŐSÉG

A dolgozat készítése során találkoztunk néhány olyan területtel, amit még lehet fejleszteni. A mikrovezérlős távirányító esetében, a felsorolt problémák megoldásán kívül pontosabb motorvezérlés is kialakítható. Az útvonal elmentése és visszajátszása is megoldásra vár. Ehhez külső memória vagy tároló (SD/MMC kártya, EEPROM) illesztése szükséges, mivel az AVR memóriájának közel 50%-át a program elfoglalja. A program struktúrájának átalakításával megoldható lenne az NXT tehermentesítése, vagyis csak akkor küldene adatot a távirányító, amikor valamilyen billentyűzet esemény következik be. Ezzel a távirányító és az NXT fogyasztását is csökkenteni lehetne. A szenzorok időleges lekérdezésével megakadályozható az ütközés: például, ha az ultrahang szenzor akadályt érzékel a robot előtt, nem engedélyez előre irányuló mozgást. A színérzékelő szenzor kezelése is fejleszthető, így nem csak világításra, hanem színek érzékelésére is alkalmas lenne.
A mobiltelefonos távirányító esetében a kanyarodások finomabb kidolgozása segíthet a pontosabb irányításban. A szenzorok ugyanúgy kezelhetőek mint a mikrovezérlős távirányítóval.
A PC-s irányítás előnye, hogy nemcsak egy NXT-t tudunk egyszerre irányítani, hanem többet is. Ebben az esetben azonban késések lépnek fel a Bluetooth miatt. Ezek a késések már két NXT esetében is feltűnőek, ezért itt mindenképp fejlesztésre van szükség.

9. FELHASZNÁLT IRODALOM

[1] A Bluetooth rendszer
http://studio.pataky.hu/edu/14p/tavkozlesi_halozatok/Bluetooth.pdf letöltve: 2010.10.04.

[2] Bluetooth
http://www.mcl.hu/~fazek/mobil_infokom_oravazlat/oravazlat_bluetooth.ppt letöltve: 2010.10.04.

[3]Avinash Gupta - Synchronous Serial Communication Tutorial – The Basics of I2C and SPI
http://extremeelectronics.co.in/avr-tutorials/synchronous-serial-communication-tutorial-–-the-basics-of-i2c-and-spi/ letöltve: 2010.06.23.

[4] Neveri Gábor – Egyszerű soros kommunikáció AVR-el (UART)
http://www.hobbielektronika.hu/kapcsolasok/egyszeru_soros_kommunikacio_avr-rel_uart_oldal2.html letöltve: 2010.07.22.

[5] Lego Mindstorms Bluetooth Developer Kit
http://cache.lego.com/upload/contentTemplating/Mindstorms2SupportFilesDownloads/otherfiles/downloadFF7F8010EB6D176857CB6CBF82A8B567.zip letöltve: 2010.07.08

[6] Az AT billentyûzet (AT Keyboard)
http://www.vfx.hu/info/atkeyboard.html letöltve: 2010.08.03.

[7] Andreas Jakl - Symbian OS: Python/PyS60
http://symbianresources.com/tutorials/general/python/Python.pdf letöltve: 2010.10.27.

[8] Charaf Hassan, Csúcs Gergely, Forstner Bertalan, Marossy Kálmán: Symbian alapú szoftverfejlesztés Szak kiadó, ISBN: 963-9131-66-0 2006

[9] A Python élete
http://www.hotdog.hu/magazin/magazin_article.hot?m_id=29143&a_id=602550&h_id=90719 letöltve: 2010.10.27.

AZ ELVÉGZETT MUNKÁT ÉS A MEGJELENÉST AZ OKTATÁSÉRT KÖZALAPÍTVÁNY TÁMOGATTA AZ NTP-OKA-XXII-038 PÁLYÁZAT ALAPJÁN.

ÖSSZEFOGLALÓ A TDK MUNKÁRÓL

Az elkészült TDK dolgozat címe: **Bluetooth kommunikáció és NXT**, amit **Málnás Péter** és **Tóth Attila János** másodéves mechatronikai mérnökhallgatók készíttetettek. A mechatronikai mérnökök munkáját Magyarországon egyre jobban kezdik megismerni. A mechatronikai mérnöki munka széleskörű ismeretet igényel. Ötvözi a gépészet, az elektronika és informatika szakágakat, melyek igen keresett és megbecsült szakmák önmagukban is. Minden mérnöki képzésben résztvevő hallgatónak el kell sajátítania nagyon sok olyan tudást, amit később a munkájában vagy további tanulmányaikban alkalmazni tud. A folyamatosan fejlődő ipar egyre nagyobb kihívások elé állítja a hallgatókat és a tanárokat egyaránt. Tanárként az a feladatom, hogy az általam megszerzett ismereteimet a lehető legjobban továbbadjam.

A két hallgatómat a mikrokontrollerek használata és az elektronika magas szintű elsajátítása fogta meg. A 2010-ben a hallgatóim kéréssel fordultak hozzám, hogy nyári gyakorlat céljából szeretnének bejárni az egyetemre. Érdeklődésüket egy NXT robot keltette fel. Az elképzeléseiket alkalmasnak láttam arra, hogy két darab NXT- vel az ötleteiket kipróbálhassák. Lehetőséget biztosítottam számukra, hogy kezdeti lépések elsajátítása után a feladatukat otthon önállóan folytathassák. A nyár folyamán folyamatos visszajelzéseket kértem és kaptam a hallgatóktól. Tóth Attila János ötlete egy NXT távirányító volt, amivel adatokat tudott küldeni és lekérni. A távirányítót meg is építette. A hallgatók az NXT Bluetooth kommunikációjával is foglalkoztak és egyedi programokat, játékokat is készítettek C nyelven. Az elkészült programok közül kiválasztottunk egy néhányat, ami a későbbi fejlesztés alapját képezte. Az ipari alkalmazásokat továbbfejlesztették, és lehetőségeket láttam abban, hogy a közös ötletünket hogyan lehetne akár ipari alkalmazásként továbbfejleszteni. A későbbi programozás alapja a LabVIEW programnyelv lett, ami a mai ipari mérés- adatgyűjtés, irányítástechnika megkerülhetetlen eszközévé vált. A LabVIEW programozási nyelvet kedvezményesen tudjuk a hallgatóknak biztosítani, otthoni használatra.

Hallgatókat a bemutatott munkájuk alapján érdemesnek találtam egy TDK pályamunkára. A biztatásnak meg is lett az eredménye, hiszen Kari TDK megmérettetésen első helyezést értek el, mellyel továbbjutottak az OTDK versenyre is. A konferenciára már egy esztétikus hardvert sikerült bemutatniuk.

A hallgatók megismerkedtek az NI LabVIEW Robotics Starter Kit-el is ahol a TDK munkájuk során megszerzett tapasztalataik alapján sikerült egyedi vezérlőt építeniük.

A hallgatókat támogattuk a hardver megvalósításához szükséges eszköz beszerzésekkel. A támogatás jó helyre került, mert ezek a hallgatók önszorgalomból rengeteget dolgoztak a projektük sikeréért. Meg vagyok róla győződve, hogy a hallgatók rengeteg szakmai tapasztalatot szereztek, megtanulták hogyan lehet másokkal együtt dolgozni, milyen érzés egy közös célért egymást támogatva sikereket elérni. Ez is egy jó példa arra, hogy a befektetett energia a tudás formájában megtérül.

A két hallgató a TDK első hely miatt + 10 pontot kap az MSc-re való jelentkezéskor. A mechatronikai mérnök BSc elvégzése után lehetőségük nyílik az MSc diploma megszerzésére is.

Munkahelyi kilátásaik elég kecsegtetők, hiszen a jól képzett mérnök szakemberek könnyen el tudnak helyezkedni akár a régiós akár országos szinten. A jól teljesítő mérnököknek nemzetközi karrier is van lehetőségük a multinacionális gyáraknál.

Mind a hallgatók és mind az eszközök folyamatos fejlesztése igen fontos. Csak úgy lehet lépést tartani az ipari alkalmazásokkal, ha folyamatos beruházásokat, fejlesztéseket hajtunk végre a versenyképes tudás megszerzéséhez. Az egyetem a fejlett laboratóriumai, eszközei, tudásbázisa tehetségeket tud felkarolni.

Erre a legjobb példa a két hallgatóm. Tudásuk, felkészültségük másodévesként is eljutatta őket az OTDK-ra.

A TDK-HOZ BESZERZETT ESZKÖZÖK:

BTM-112 modul	3500 Ft
Atmega8-16 PU modul	1875 Ft
CAB Műszerdoboz	2450 Ft
RC-1602-B LCD kijelző	1565 Ft

Dr. Tóth János
sk.