HADMÉRNÖK

IV. Évfolyam 4. szám - 2009. december

Nagy Tibor István
nagy.tibor@nik.bmf.hu

A FELÜGYELET NÉLKÜLI SZENZORHÁLÓZATOK ÉS A PROGRAMOZÁSI NYELVEK KAPCSOLATA

Absztrakt

A felügyelet nélküli szenzorhálózatok a civilszférában és a szárazföldi harcászati felderítésben egyaránt fontos szerepet játszanak. A legtöbb informatikai eszközhöz hasonlóan a szenzorok működéséhez is szükségesek a hardverösszetevőkön túl szoftverkomponensek is, melyeknek elkészítéséhez valamilyen programozási nyelvre van szükség. Ez a publikáció áttekintést ad a felügyelet nélküli rendszereknél használt programozási nyelvekről, és azok jellemzőiről.

The unattended sensor networks play a significant role in both the civil applications and military ground surveillance and reconnaissance. Like most of the IT devices, the sensors also require hardware and software components. Software component are created with programming languages. This paper reviews the programming languages used with unattended systems, and their characteristics.

Kulcsszavak: *szenzor, felügyelet nélküli szenzorhálózat, intelligens szenzorhálózat, programozási nyelvek ~ sensor, unattended sensor network, intelligent sensor network, programming languages*

Bevezető

A felügyelet nélküli, intelligens szenzorhálózatok fontos szerepet játszanak napjaink haditechnikájában, azon belül is a harcászati felderítés legmodernebb és legbiztonságosabb eszközrendszerét biztosítják. A felderítés során az ellenséges erők helyzetét, mozgását, felszereltségük mértékét szükséges beazonosítani annak érdekében, hogy meg lehessen tervezni az offenzív vagy defenzív műveletek részleteit. A felderítés hagyományos módon humán erőforrás felhasználásával történik, amely a legköltségesebb (legfőképp elvesztése esetén). A szenzorhálózat automatizálja a felderítési területen végzendő veszélyes adatgyűjtést és biztosítja az adatok távolról történő feldolgozását, értékelését.

A szenzorhálózatok úgynevezett node-okból, vagy más néven mote-okból állnak, amelyek egy, vagy több – környezeti jellemző mérésére alkalmas – szenzort tartalmaznak. A node-ok képesek egymással rádiós kapcsolatot létesíteni és így egy hálózatot vezeték nélküli hálózatot

létrehozni. Működésükhöz emberi felügyelet nem szükséges, automatikusan képesek elvégezni feladatukat. A bonyolult műveletek miatt nem elég az elektronika, a hardver, szükség van az azt működtető szoftverre is, amely vezérli a hálózat kiépítését, a mért jellemzők begyűjtését, eltárolását, és az adatok továbbítását, azaz a felderítési ciklus két fázisában is jelentős szerephez jutnak: az adatszerzésben és adatfeldolgozásban egyaránt [1]. A szoftver elkészítéséhez pedig szükség van valamilyen programozási nyelvre. Célom feltárni a programozási nyelvek jellemzőit, elemezni a szenzorhálózatoknál jelenleg használt nyelvek tulajdonságait, illetve csoportosítani ezeket.

Csoportosítási szempontok

A programozási nyelveket sokféle szempont szerint fel lehet osztani. Ezek közül azokat veszem sorra, amelyeket a felügyelet nélküli szenzorrendszerekkel kapcsolatban a legfontosabbnak tartok, azaz a programozási nyelv alapját képező paradigmát, számítási modellt, a nyelv emberközeliségét, illetve a felhasználók körét.

Emberközeliség

Ebből a szempontból két csoportot szoktak megkülönböztetni. Az első az alacsony szintű, vagy más néven hardver közeli nyelvek csoportja. Ezeknek a jellemzője, hogy a programozó a hardver beépített utasításkészletét használhatja fel. Átfogó ismerettel kell rendelkeznie a hardver, illetve annak elemeinek működéséről. Kevés programszerkezeti elem áll rendelkezésre, az összetett feladatok megoldása bonyolult.

A másik csoportba a magas szintű nyelvek tartoznak. Ezek a hardvertől távolabb, a programozóhoz közelebb helyezkednek el. A programozó számára könnyebb a programkészítés, több szerkezeti elem áll rendelkezésre, egy-egy utasítás bonyolultabb feladatok végrehajtására is alkalmas.

Paradigmák

A programozási paradigma – vagy más néven a programozási nyelv számítási modellje – a programozási nyelv által használt szemléletmódot, alapelvet, illetve az ezzel kapcsolatos szabályokat, szabályrendszert jelenti. Erre a szemléletmódra, szabályrendszerre épül a nyelv, annak szerkezete, illetve meghatározza a benne készíthető programok felépítését, és a benne használható szerkezeti elemeket. A paradigma ezen túlmenően behatárolja azon feladatok körét, amelyek a legkönnyebben, legelegánsabban, vagy éppen a legoptimálisabban megoldhatók az adott nyelven. Kialakulásuk és fejlődésük a magas szintű programozási nyelvek megjelenéséhez köthetőek. Egyetértek a fent leírtakkal, de csak azzal a kiegészítéssel, hogy a programozási nyelveknél általában elmondható, hogy nem egy számítási modellt alkalmaznak, hanem több számítási modell ötvözetére, vagy egymás melletti együttélésére alapoznak. Az alábbiakban a fontosabb programozási paradigmák rövid jellemzése olvasható.

Automataelvű paradigma

Az automataelvű paradigmát használó programozási nyelvek végrehajtójának egy automatát tekintünk. Az automata rendelkezik állapotokkal, és az állapotok közti átmenetet leíró függvényekkel. Az állapotok állapotkomponensekből állnak, amelyeket a függvények tudnak megváltoztatni. Az automata rendelkezik egy kiinduló állapottal.

A program és az automata állapotai egymástól elkülönülve tárolódnak. A programban egyszerű vezérlési szerkezetek valósíthatóak meg.

Imperatív (utasításorientált) paradigma

A Neumann elvű számítógépek tulajdonságait kihasználó számítási modell. A Neumann elvű számítógépek legfontosabb jellemzője, hogy címezhető, írható/olvasható memóriával rendelkeznek, amelyben az adatok és a programok utasításai is eltárolásra kerülnek. Az utasítások a számítógép által értelmezhető és végrehajtható „egységnyi" tevékenységek, amelyek egy megadott funkció elvégzésre késztetik a számítógép processzorát. A programban lévő utasítások sorban egymás után, azaz szekvenciálisan hajtódnak végre. A címezhető memória biztosítja a változók használatának lehetőségét, amelyek egy-egy – a program működéséhez szükséges – adat tárolását teszik lehetővé. A program címezhető memóriában tárolásának következménye, hogy a programok végrehajthatóak egy megadott utasítástól kezdve és ezt kihasználva a szekvenciális végrehajtás kiegészíthető ciklusokkal, elágazásokkal is. [2]

A legismertebb képviselői a BASIC, C, Ada, Pascal.

Funkcionális (függvényorientált) paradigma

Egzakt matematikai alapokon nyugvó programozási paradigma. A programszerkezetet függvényekből és a függvények közti matematikai kapcsolatokból – függvénykompozícióból, alternatív függvényekből és rekurzióból – lehet összeállítani. Az adatok tárolásához változók nem állnak rendelkezésre; helyettesítésük függvényparaméterekkel és a függvények visszatérési értékeivel oldhatóak meg. Ebből következik, hogy változó-értékadás helyett paraméterátadást és rekurziót kell használni. A matematikai alapok miatt az ilyen számítási modellt használó programozási nyelvekben írt programok helyessége egyszerűen, matematikai apparátus felhasználásával bizonyítható, tehát megbízható és biztonságos programok készíthetőek velük. Ilyen nyelvet használtak például a párizsi metró irányítórendszerének elkészítéséhez is. [2]

Legismertebb képviselői a Logo, Lisp, ML nyelvek.

Logikai paradigma

Ez a paradigma is a matematika egy részterületének alapelveire támaszkodik. A programot logikai formulákból építi fel. A formulákhoz meghatározhatók következtetési szabályok, amelyek meghatározzák a logikai formula kiértékelését. A formula kitöltött és kitöltetlen – azaz nem meghatározott értékű – is lehet. A kiértékelés eredményei a kitöltetlen paraméterek különböző lehetséges értékei, illetve egy igaz vagy hamis érték lehet. A paradigma legfőbb felhasználási területe a mesterséges intelligencia. Működési alapelvéből adódóan azon feladatok megoldására legoptimálisabb, ahol visszalépéses keresést kellene alkalmazni. A visszalépéses keresés egy programozási tétel, amelynek megvalósítása nem logikai paradigmát alkalmazó nyelvek esetén nem könnyű feladat.

Legismertebb képviselője a Prolog programozási nyelv.

Deklaratív paradigma

Szinte minden számítási modell esetén a programban azt kell leírnia a programozónak, hogy az adott feladat megoldásához milyen lépések, utasítások végrehajtásán keresztül lehet eljutni. Ez

alól a deklaratív paradigma talán az egyetlen kivétel. Itt azt kell megfogalmazni, hogy milyen bemenő adatokból milyen eredményt akarunk elérni. Az eredmény elérésének miként jét a programozási nyelv elrejti a programozó elől, illetve automatikusan elvégzi. Az ilyen nyelveket leginkább a relációs adatbázisok kezelésénél használják.

Legismertebb képviselője az SQL lekérdező nyelv.

Objektum-orientált paradigma

Alapelve, hogy a valós világot objektumokra bontja és ezeknek az objektumoknak a számítógépes modelljeit lehet megvalósítani a számítógépes programban. Az objektumok a rájuk jellemző adatokat és az adataikon végzendő műveleteket fogják össze egy egységbe. Ez a paradigma felhasználja az utasításorientált számítási modell elveit is a műveletek elkészítésénél, illetve az adatok tárolásánál. Kialakulásukat a grafikus felhasználói felületű programok iránti felhasználói igény ihlette. Manapság ezt a paradigmát használó programozási nyelvek a legelterjedtebbek.

Legismertebb képviselői: C++, Java, C#.

Programozási nyelv felhasználói köre

A számítógépeknél használatos programozási nyelvek a felhasználók köre szerint két nagy csoportra bonthatók: az amatőr és a professzionális nyelvekre. Az amatőr nyelveket nem informatikus, illetve programozó szakembereknek, hanem „laikusoknak” készítik, az általuk történő felhasználásra optimalizálják őket, emiatt jellemzőjük az egyszerűség: egyszerű programszerkezettel rendelkeznek, emellett nagy utasításkészlettel, melynek segítségével kiveszik a programozó kezéből a nehezen megoldható, de gyakran használt feladatok elkészítését. A professzionális nyelvek kisebb utasításkészlettel rendelkeznek, de nagyobb szabadságot adnak a programozónak, bonyolultabb programszerkezet felépítését teszik lehetővé, biztosítják a programkód újrafelhasználhatóságát modulok, komponensek, vagy más építőelemek felhasználásával. [2]

A fenti helyett a szenzorhálózatok katonai alkalmazásainál a programozási nyelvek felhasználóit négy csoportra bontanám, figyelembe véve, hogy a felügyelet nélküli rendszerek tervezésénél, karbantartásánál és használatánál milyen ismeretekre van szükség:

- Laikusok: olyan szakemberek, akik valamely tudományterületnek (pl.: a hadtudományoknak) szakértői, viszont informatikai tudással nem, vagy csak alapszinten rendelkeznek, programozási tudás pedig szinte egyáltalán nincs birtokukban

- Elektronikai szakemberek: az elektronika, híradástechnika, rádiózás témakörében jártasak, és az ehhez szükséges programozási ismeretekkel is rendelkeznek, ami általában a mikrokontrollerek programozásához szükséges assembly, vagy a nagyintegráltságú áramkörök programozásához használható VHDL hardverleíró nyelv.

- Informatikai szakemberek: az informatikai rendszerek, hálózatok kiépítésében, karbantartásában jártas szakemberek, akik egy programozási nyelvet alap, vagy legfeljebb középszinten ismernek. A számítógépek és elektronikai eszközök összekapcsolásához szükséges alapszintű elektronikai ismeretekkel rendelkeznek.

- Programozók: több programozási nyelvben való magas szintű jártasság és a hozzájuk kapcsolódó egyéb ismeretek birtokában lévő szakemberek, akik az informatikai rendszerek, hálózatok tekintetében csak alapszintű ismeretekkel rendelkeznek, elektronikai ismereteknek pedig nincsenek birtokában.

Szenzorhálózatok programozási nyelvei

Nagyon sok programozási nyelv készült a felügyelet nélküli szenzorhálózatok különböző feladatainak vezérlésére. Az alábbiakban a teljesség igénye nélkül néhánynak a leírása olvasható.

Basic

A BASIC programozási nyelv az utasításorientált nyelvek közé tartozik. Könnyű megtanulni a használatát, a kezdő programozónak nem kell olyan adminisztrációs jellegű utasításokat megtanulnia, amelyeket más programozási nyelvek megkövetelnek (ilyen például a változódeklaráció). Egyszerűsége miatt a programírásban kevéssé, vagy egyáltalán nem járatos személyek számára is rövid idő alatt elsajátítható. Interpreter típusú nyelv, ami azt jelenti, hogy az elkészített program fordítása – BASIC nyelvről a node által végrehajtható utasításokra – a program futtatása közben, utasításonként történik. Ez a tulajdonság a programfutás sebességét csökkenti, azaz a feladatok elvégzéséhez szükséges időt növeli. A Northwestern egyetem Electrical Engineering and Computer Science tanszék kutatói készítettek egy BASIC implementációt, illetve integrált fejlesztőkörnyezetet kimondottan vezeték nélküli szenzorhálózatokhoz. A nyelvbe beépítettek néhány utasítást, amelyek a kommunikációt, energiatakarékos üzemmód vezérlését, az érzékelők által mért információk beolvasását teszik lehetővé. A kutatók egy felméréssel próbálták bizonyítani az általuk kidolgozott nyelv könnyű elsajátíthatóságát. A felmérésben 40 végzett, vagy végzős egyetemistát teszteltek, mindegyikük kevés, vagy egyáltalán semmilyen programozási tudással nem rendelkezett. Eredményként azt kapták, hogy a résztvevők 45-50%-a meg tudott oldani egyszerűbb – előre meghatározott – feladatokat. Vizsgálták ezen túlmenően az elkészített programok és a node-ok fogyasztásának kapcsolatát is. Eredményük szerint a BASIC-ben írt program által okozott fogyasztás – nem számításigényes programok esetén – mindössze 1,5%-kal nagyobb a natív C-ben megírtnál. Feltételezésük szerint nagyobb számítási kapacitást igénylő feladatok nem is igen fordulnak elő a szenzorhálózatoknál [3]. Én ezzel a megállapítással vitatkoznék, mivel az egyes node-ok pontos helyének ismerete hiányában közöttük hálózatot kiépíteni valamilyen szakértőirendszer-elmélet, gráfelmélet felhasználásával lehetséges, amelynek megvalósítása utasításorientált nyelvben meglehetősen számításigényes feladat. A felügyelet nélküli szenzorok harcászati felderítésben történő alkalmazásánál pedig pont ilyen ad- hoc hálózatok kiépítésére lehet szükség.

NESC

Komponens alapú programozási nyelv, a C nyelv egy speciális változata, amelyet kifejezetten a vezeték nélküli szenzorhálózatok korlátozott memóriakapacitásának figyelembe vételével fejlesztettek ki. A benne írt programok előre elkészített komponensekből tevődnek össze, melyek egymással interfészeiken keresztül kommunikálnak. Kész komponensek léteznek a legfőbb hardverfunkciók megvalósítására, mint a csomagkapcsolt kommunikáció, útvonalválasztás, érzékelők adatainak begyűjtése és tárolása.

JAVA

A Sun Microsystems SPOT projektje az intelligens vezeték nélküli hálózatoknak egy implementációja. A Sun SPOT device-ok kisméretű, különböző szenzorokat tartalmazó node-ok, amelyekből egy kis hatótávolságú vezeték nélküli hálózat építhető ki.

Ehhez a technológiához készítettek a Squawk elnevezésű project keretein belül egy olyan Java virtuális gépet, amelynek a lehető legnagyobb részét magát is Java nyelven írtak meg. A cél kis memóriakapacitással rendelkező eszközökhöz optimális futtatókörnyezet készítése volt.

Erre a virtuális gépre a Java MicroEdition egy speciális változata használható. Előnye, hogy magas szintű, objektum-orientált nyelven, kisebb költséggel, kényelmesebben, könnyebben tesztelhetően lehet programokat készíteni beágyazott rendszerekhez, mint az assembly, vagy C nyelvek használatával. Rendelkezésre állnak különböző fejlesztőeszközök is a programok elkészítéséhez, teszteléséhez, nyomkövetéséhez, mint például a teljesen ingyenes és a Java-s programozók körében elterjedt és népszerű NetBeans.

TINYSQL

Egy egyszerű, Java programozási nyelven írt relációs adatbázis-kezelő rendszer, amely dBase formátumban tárolja az adatbázisokat [4]. A szenzorhálózatokban történő alkalmazásánál egy táblában kerül eltárolásra a hálózat aktuális időpillanatbeli állapota. A tábla meghatározott időközönként frissül. A sorok az egyes node-okat, az oszlopok pedig a mért jellemzőket, illetve a node-ok bizonyos tulajdonságait jelentik. Az aktuális állapot lekérdezése a szokásos SQL szintaktikával lehetséges, míg a korábbi állapotok úgynevezett mentési pontok létrehozásával tárolhatók.

SWISSQM[1]

A SwissQM egy verem alapon működő virtuális gép vezeték nélküli szenzorhálózatokhoz kifejlesztve. Platform független, mivel a nyelvben megírt programból bájtkód készül. Biztosítja az eseménykezelés lehetőségét, függvények definiálását, állapotgépek, Turing-automata elvű programozást, illetve többféle magas szintű programozási nyelv használatát. [5]

WASP

A Northwestern és a Michigani egyetem kutatói által kifejlesztett nyelv, amelynek középpontjában az az elgondolás áll, hogy a szenzorhálózatokat nagy többségében nem programozók, illetve informatikusok használják, hanem valamilyen más tudományterület művelői, akiknek munkájukhoz és kutatásukhoz van szükségük ezekre a felügyelet nélküli eszközökre.

Az egyetem kutatói több alkalmazást és programozási nyelvet is megvizsgáltak. Arra a következtetésre jutottak, hogy a létező nyelvek többsége általános nyelv, laikusok számára nehezen megtanulható, ezért egy új nyelv kifejlesztését tűzték ki célul, amely könnyen elsajátítható és egy adott feladattípushoz használható fel. Ehhez beazonosították azokat az alkalmazás-jellemzőket, amelyek alapján feladatosztályokba lehet sorolni az alkalmazásokat. A jellemzők a következők:

[1] SwissQM - Scalable Wireless Sensor Query Machine

- **Mobilitás**: vannak olyan node-ok, amelyek működésük közben változtatják helyzetüket, és olyanok is, amelyeknek a pozíciója állandó.

- **Mintavételezés kezdeményezése**: meghatározza, hogy a mintavételezés milyen feltétel bekövetkezésekor indul. Lehet periodikus, eseményvezérelt, vagy a kettő ötvözete.

- **Adatátvitel kezdeményezése**: meghatározza, hogy a node-ok milyen feltétel teljesülése esetén kezdeményezhetnek adatforgalmat a hálózaton. Az előző jellemzőhöz hasonlóan ez is lehet periodikus, eseményvezérelt, vagy a kettő ötvözete.

- **Vezérlés**: meghatározza, hogy a szenzorhálózat létrehoz-e olyan jeleket, amelyekkel más hardvereszközök vezérelhetők.

- **Interakció**: meghatározza, hogy a szenzorhálózatnak kell-e tudnia reagálni, választ adni bizonyos parancsokra. Interakcióra tipikusan a telepítésnél, átprogramozásnál, karbantartásnál, illetve a működési paraméterek megváltoztatásánál lehet szükség.

- **Adatok értelmezése**: meghatározza, hogy szükség van-e az érzékelő adatok átalakítására, szűrésére a szenzorhálózaton belül.

- **Adatok egyesítése**: meghatározza, hogy szükség van-e több node-tól érkező adatok egyesítésére

- **Homogenitás**: meghatározza, hogy a hálózat node-jai mind ugyanazt a funkciót látják-e el.

Ezen jellemzők alapján a lehetséges feladatokat hét feladatosztályra bontották fel. Minden feladatosztályhoz egy-egy külön nyelv létrehozását tervezik, amelyből eddig kettőt készítettek el. Az első feladatosztályba a periodikus mintavételezésű, nyers adatokkal, adatok szűrésével és egyesítésével működő, nem mobil, homogén alkalmazások tartoznak. A programozási nyelv külön részben írja le az egyes node-ok, illetve külön másik részben a teljes hálózat működését. A nyelv fordítóprogramja NesC kódot állít elő, amelyből újabb fordítás után generálható a futtatható program. [6]

Szenzorhálózatok rogramozási nyelveinek jellemzői

Az alábbi táblázatban az előző fejezetben leírt nyelvek programozási paradigmák szerinti besorolása látható.

1. táblázat A programozási nyelvek paradigmák szerint

	Automataelvű	Utasításorientált	Funkcionális	Logikai	Deklaratív	Objektum-orientált
BASIC		✚				
NesC		✚				
Java		✚				✚
TinySQL					✚	
SwissQM	✚				✚	
WASP		✚			✚	

A vizsgált nyelvek közül a legtöbb – nem meglepő módon – az utasításorientált paradigmát használják. Hosszú ideig a PC-knél is ezek a nyelvek voltak a legnépszerűbbek, és a nagyintegráltságú áramkörök esetén még most is szinte ez az egyetlen használható számítási modell. Kicsit meglepőbb és érdekes megoldási mód a deklaratív számítási modell használata. A logikai paradigma használata a hálózatkiépítésnél indokolná a logikai paradigma alkalmazását, azonban a fenti nyelvek közül egyik sem tartozik ebbe a körbe. Az objektum-orientált paradigmát pedig csak a Sqawk használja, ami a végrehajtási sebesség és a felhasznált tárkapacitás szempontjából érthető is.

Összefoglaló

A felügyelet nélküli szenzorhálózatok fontos szerepet játszanak a harcászati felderítésben a humán erőforrás kiváltására, illetve a felderítő személyzet életének és biztonságának megvédésében. Mint minden számítástechnikai eszköznél, a node-ok esetében is lényeges elem a hardverösszetevőkön túl az azokat működtető szoftver is. A szoftverek elkészítéséhez valamilyen programozási nyelvre van szükség, melyeknek tulajdonságai meghatározzák a bennük készíthető szoftverek jellegét, megbízhatóságát, biztonságosságát, illetve a velük megoldható feladatok körét. Manapság a kutatások iránya olyan programozási nyelvek kifejlesztése, amelyeket nemcsak informatikai szakemberek, illetve programozók tudnak használni, hanem a felhasználási tudományterület képviselői is könnyen megtanulhatnak. Véleményem szerint a fejlődés iránya és a programozási nyelvek, illetve azok alkalmazási területei ezt a jövőt festik fel.

Irodalomjegyzék

[1] Kovács L.: *Az elektronikai felderítés korszerű eszközei, eljárásai és azok alkalmazhatósága a Magyar Honvédségben*. PhD értekezés, ZMNE, Budapest, 2003.

[2] Temesvári T., Szlávi P., Zsakó L.: *Mikrológia 22, Programozási nyelvek: Alapfogalmak*. NJSZT, 2004.

[3] J. S. Miller, P. A. Dinda, R. P. Dick: *GOTO Considered Helpful: A BASIC Approach to Sensor Network Node Programming*. Technical Report NWU-EECS-09-01, Northwestern Univertsity, 2009.01.31.

[4] http://www.jepstone.net (2009.11.21.)

[5] http://www.swissqm.inf.ethz.ch (2009.11.21.)

[6] L. S. Bai, R. P. Dick, P. A. Dinda: *Archetype-Based Design: Sensor Network Programming for Application Experts, Not Just Programming Experts*. IPSN'09, 2009.04.13-16.