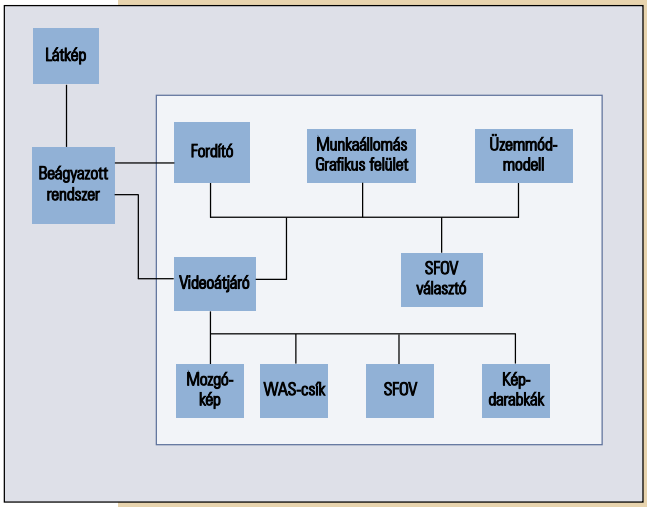
RSTA-MEP és a Linux-munkaállomás

Önműködően észleli az ellenséget a sötétben, és tájékoztatja a saját erőket annak helyéről.

Nemrég fejeztük be a Linuxra épülő Felderítő, megfigyelő és célbefogó küldetés felszereléscsomag (Reconnaissance, Surveillance and Target Acquisition Mission Equipment Package, azaz RSTA-MEP) munkaállomásának a kifejlesztését. Írásunkban röviden bemutatjuk a teljes rendszert, majd a munkaállomást részletesebben is tárgyaljuk.

A Raytheon RSTA-MEP programja módot ad a harctéri helyzet gyors értékelésére, amit az egyesített fedélzeti és nem fedélzeti érzékelők által nyújtott valós idejű adatok tesznek lehetővé. Az érzékelők és a programok fejlődése lehetővé teszi a széles területű keresésen (wide-area-search – WAS) alapuló képalkotást és az önműködő célészlelést (Automatic Target Detection – ATD), valamint a támogatott célfelismerést (Aided Target Recognition – AiTR). Ezek a képességek a kezelőszemélyzetet valós idejű adatokkal látják el, beleértve a cél helyzetét, osztályozását és elsőbbségének meghatározását. Ezt az amerikai hadsereg harcászati hálózatával (US Army's Tactical Internet) kombinálva lehetővé teszi a kezelők számára, hogy részt vegyenek a saját és az ellenséges erőkről alkotott átfogó hadműveleti kép kialakításában. Ez a jármű egy technológiai bemutató eszköz, amely azt szemlélteti, hogy milyen új képességekkel lehet felruházni a jelen és a jövő felderítő járműveit.



A munkaállomás számítógépes kapcsolatai és moduljai

A jármű jelenlegi kialakításában hőkép-előállító, éjjellátó berendezéssel van ellátva. Az elsődleges érzékelők az árbocon egy nagy hatótávolságú előrenéző infravörös érzékelőnek (Forward Looking Infrared – FLIR), egy inerciális (tehetetlenségi) navigációs rendszernek (Inertial Navigation System – INS) és egy helymeghatározó rendszernek (Global Positioning System – GPS) a vevői, ezen felül számos Raytheon NighSight éjjellátó infravörös érzékelő van a járműre erősítve, hogy a kezelőszemélyzet többi tagja is figyelemmel tudja kísérni a jármű közvetlen környékét.

Az árboc négy méter magas, ehhez adódik még a jármű magassága, így összesen öt méterre lehet kitolni. A járműben három főnyi személyzet utazik: a vezető, a parancsnok és a megfigyelőkezelő. A vezető szintén használni tudja az éjjellátó érzékelőket, hogy éjszaka, sötétben is tudjon vezetni és biztonsági okokból körbe tudjon nézni. A parancsnok feladata a harcászati hálózat kapcsolatának fenntartása és a két másik kezelő irányítása, valamint ő is hozzáfér az NightSight éjjellátó érzékelőkhöz. A megfigyelőkezelő használja a Linux-munkaállomást, amivel az árbocon lévő érzékelőket és a hozzájuk tartozó beágyazott rendszereket kezeli.

Az RSTA-MEP rendszer az árbocra telepített érzékelőkből, a beágyazott számítógépekből és a Linuxot futtató PC-s munkaállomásból áll, és mindez egy H1 Hummer terepjáróra lett telepítve. A részegységek az *ábrán* látható módon kapcsolódnak egymáshoz.

A beágyazott oldal

A beágyazott számítógépek digitális jelfeldolgozó processzorok, amelyek az érzékelők mechanikáját és elektronikáját vezérlik (például beirányozzák vagy lehűtik a detektort), emellett a képfeldolgozásban is részt vesznek. Ezek VxWorksöt futtató PowerPC-kártyák, Microsoft Windows NT-t és Sun Solarist futtató egykártyás gépek. A rajtuk futó alkalmazások között megtalálható a Force XXI Battle Command Brigade and Below (FBCB2), ami egy amerikai katonai digitális utasító- és irányítórendszer, célkereső és azonosító, valamint képfeldolgozó és kapcsolattartó program is. A csomag tartalmaz egy GPS-vevőt, tehetetlenségi navigációs rendszert és digitálistérkép-szolgáltatást. A beágyazott rendszerek optikai kábelen keresztül, ethernet és Virtual Interface (VI) protokoll segítségével tartják egymással a kapcsolatot.

Kapcsolódás a munkaállomáshoz

A Linux-munkaállomás prototípusa egy már létező rendszer utóda. Az eszményi eset az lett volna, ha munkaállomásunk pontosan beillett volna az előző helyére. A VI és az optikai kábel használatára irányuló első kísérleteink kudarcot vallottak. A beágyazott rendszerekkel foglalkozó csoportunknak már tekintélyes tapasztalataik voltak a beszállítók együttműködő-képességéről, vagyis az optikai kábelt választva csupán azokra a beszállítókra hagyatkozhattunk, akik VxWorks és Linux alatt működő kártyákat és meghajtóprogramokat egyaránt tudtak kínálni. Ezek közül egyet sem találtunk, aki a VI protokollt támogatta volna. Második kísérletként megpróbáltunk lemezemulációt használni és annak a látszatát kelteni, hogy egy merevlemezt csatlakoztattunk az optikai kábelre, így legalább ugyanazon az adathordozón belül maradhattunk.

Az eredmény szintén nem volt kielégítő, így áttértünk gigabit ehternetre. Az ethernet tudná hordozni egyrészt a videójelet az érzékelőktől, másrészt a parancs- és állapotadatokat a munkaállomás és a beágyazott rendszerek között. Gigabit ethernet megoldás keresésekor négy dolgot kell meghatároznunk: a csomagméretet,

az átviteli közeget, az összekapcsolás módját és a hálózati csatolókártyát.
A hagyományos ethernet legnagyobb csomagmérete 1500 bájt. Az egyre inkább terjedő gigabit ethernet 9000 bájtnyi legnagyobb csomagméretet engedélyez, ezt nevezik „jumbo" csomagnak. A mi projektünkben a beágyazott oldal és a linuxos oldal közötti megfelelőség miatt a hagyományos csomagméret mellett döntöttünk.
A következő, amit meg kell fontolnunk, a média. A gigabit ethernetkártyák két változatban kaphatók: réz- és optikai kábellel. A réz az elektromágneses interferenciára (EMI) érzékeny, míg az optikai kábel mechanikailag sérülékeny. Végül az ára miatt a rezet választottuk. Ha az elektromágneses interferencia gondot okozna, bármikor optikai kártyára lehet váltani a program módosítása nélkül. Azért is választottuk a rezet, mert könnyen tudtuk illeszteni laboratóriumunk meglévő berendezéseihez. Meglévő hálózatunk (a 10/100-as) is rézvezetékes volt.
A harmadik szempont az összekapcsolás módja. A helyzet nem sokat változott a 10/100-as ethernethez képest: vannak kapcsolók (switch), jelelosztók (hub) és átkötő kábelek. A kapcsolók irányítják a forgalmat, amit így csak a címzett kap meg. Kezelik a különböző sebességű, egy- és kétirányú kapcsolatokat, és villogó fényekkel jelzik a működést, segítve a hibakeresést. A kapcsolók hátulütői az áruk, és hogy felügyelhető hálózati kapcsolóra van szükség, ha csomagfigyelőt (packet sniffer) szeretnénk használni.
A másik lehetőség a jelelosztók használata. Előnyük, hogy olcsóbbak, mint a kapcsolók, és nekik is vannak állapotjelző lámpácskáik. A gond az, hogy eddig nem találkoztunk gigabit ethernetes jelelosztóval (csak kapcsolóval), ha pedig 10/100-as jelelosztót használunk, akkor fel kell áldoznunk a sebességet. A jelelosztók ráadásul az összes csomagot minden vonalra elküldik, ami jó, ha meg akarjuk figyelni a csomagokat, de rossz, ha korlátozni akarjuk a csatolón átfolyó forgalom mennyiségét.
A legegyszerűbb megoldás az átkötőkábel használata. Ez a legolcsóbb, nem igényel más eszközt és biztosak lehetünk benne, hogy külső forrásból nem érkezik csomag. Az is igaz viszont, hogy nincsenek villogó lámpácskák, nincs mód kívülről megfigyelni a csomagokat, és ha egy csatoló leáll (például újraindul a beágyazott gép), akkor megáll a másik is.
Végül a kapcsolók mellett döntöttünk, bár a kapcsolók és átkötőkábelek közötti választás még mindig „vallási" vita tárgyát képezi. Szintén nagy gondot fordítottunk a gigabit-kábelezésre. A profi módon előállított CAT 5e és CAT 6 kábelek használata előnyösebb a házilag készített kábeleknél.
A negyedik eldöntendő kérdés a hálózati csatolókártya választása. Ezek 32 vagy 64 bitesek lehetnek. A 64 bites kártyák jellemzően jobb teljesítményt nyújtanak és kevésbé terhelik le a PCI-sín erőforrásait. Bár nem végeztünk piackutatást az elérhető termékek körében, az Intel Pro/1000 Server Adapterre esett a választásunk.
A protokollok közül a TCP/IP mellett döntöttünk.

Bár a TCP lassabb, mint az UDP, megbízható, kijavítja az eldobott, kétszerezett vagy nem sorrendben érkező csomagok által okozott hibát. Szerettük volna a legjobb videóminőséget elérni a lehetséges elektromágneses interferencia ellenére is, ezért úgy véltük, hogy a beépített hibajavítás megléte alapvető. Ezenkívül az sem elhanyagolható, hogy így az utasító- és állapotadatok

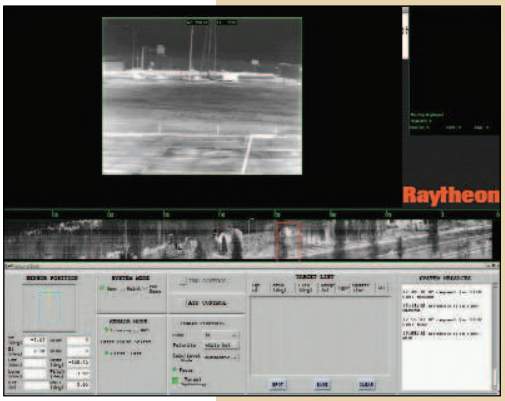
1. kép
Hummerre telepített RSTA-MEP rendszer kitolt árboccal. Az érzékelők az árboc tetején vannak, a beágyazott rendszerek hátul, a fehér dobozokban. A munkaállomás számítógép a jármű belsejében található

is megbízhatók maradnak. Amikor a foglalat- (socket) réteget kódoltuk hozzá, be kellett hangolnunk a foglalat küldő és fogadó átmeneti tárának méretét (a `setsockopt`-ot használtuk a `SOL_SOCKET SO_RCVBUF` és `SO_SNDBUF` kapcsolókkal), hogy az áteresztőképessége elég legyen a videó számára. Ezenkívül kikapcsoltuk a Nagle-algoritmust (a `setsockopt IPPROTO_TCP`-vel és a `TCP_NODELAY`-jel), hogy csökkentsük a késleltetést a munkaállomás és a beágyazott rendszer között, amivel ez utóbbit érzékenyebbé tettük a munkaállomáshoz csatlakoztatott vezérlőkarok által adott érzékelő-irányzó parancsokra.

Munkaállomás-alkalmazói program

Ez a munkaállomás-prototípus a Raytheon Tiger szimulátorból nőtte ki magát – a beágyazott oldallal ellentétben, amely az amerikai hadsereg fegyverek és rendszerek technikai kialakítását végző munkacsoport (Weapons Systems Technical Architecture Working Group – WSTAWG) által kidolgozott általános kezelői környezet (Common Operating Environment – COE) átültetése.
Bár mind a munkaállomás, mind a beágyazott oldal rendszere üzenettovábbító alapú, ezek egymással nem működnek együtt. Ezért egy fordítómodult szükséges a kettő közé illeszteni. A késleltetés és a processzorterhelés csökkentésére ez a fordítófolyamat (process) két szálra van szétválasztva, a Posix-szálak (threads) függvénykönyvtárának a használatával. Az egyik szál a beágyazott oldalon vár üzenetre, és azt lefordítva a munkaállomás elemei által használt megosztott memóriaterületre helyezi. A másik szál ugyaninnen veszi a

beágyazott rendszereknek szóló üzeneteket, hogy fordítás után továbbítsa azok bemenetére. A fordítómunka két szálra osztásával és a program javításával (optimizations) a késleltetést a legkisebb értéken lehet tartani.



2. kép
A munkaállomás mozgó videomódban

A videóátjátszó modul egy, csak ennek a feladatnak szentelt külön gigabit ethernet hálózati kapcsolaton keresztül olvas. Meghatározza, hogy melyik videóablakban kell megjeleníteni a képet, és oda továbbítja.
A munkaállomáson a kezelőfelület vezérlőpanelje a Builder's Xcessoryval lett létrehozva. Három fő szempont vezérelte a kialakítását: a korlátozott képernyőméret, a kész felület tükrözze a beágyazott oldal állapotát, valamint egér vagy hanyatt egér (trackball) helyett vezérlőkart (grip) lehessen használni.
Az első fő tervezési nehézség, amivel szembekerültünk, a képernyő területe volt. Egyetlen monitoron zajlik minden megjelenítés, ezért csak a képernyő alsó harmada áll a kezelőfelület rendelkezésére. A rendszer üzemmódja és ennek az üzemmódnak a vezérlőelemei tehát ebben a harmadban jelennek meg. A rendszernek két fő üzemmódja van: a WAS-mód és a hagyományos kameramód. WAS-módban az érzékelő gyorsan pásztázza a kezelő által meghatározott területet, miközben a vezérlőkarokkal ki lehet választani egy részt, amit szuper látómezőként (Super Field Of View – SFOV) lehet megjeleníteni. Kameramódban élő videókép látható és a vezérlőkarokkal lehet az érzékelőt mozgatni. Egyik üzemmódban sincs szükség a másik mód vezérlőelemeire, ezért két elemkészletet (widgets) terveztünk ugyanarra a képernyőterületre. A kezelőfelület-érzékelő üzemmód panelje a *2.* és *3. képen* látható. Amikor az egyik készletet használjuk, a másik el van rejtve.
Egyéb szolgáltatások, mint például az önműködő célészlelő program kezelőszervei, egy másik ablakban kaptak helyet, és a fő kezelőfelületről egy gombnyomással elérhetők. Ezek az ablakok a képmegjelenítő terület fölé ugranak be.
A képernyőterület szűkössége egy másik kérdést is felvet: szükség van a felhasználói utasításokra válaszoló rendszer azonnali, látható visszajelzésére csakúgy, mint a beágyazott rendszer pillanatnyi állapotának jelzésére. Külön elemek helyett ugyanazokat alkalmaztuk vezérlő- és állapotjelző objektumokra. Amikor a rendszer kezelője használ egy elemet, a rendszer a kiadott parancsát önműködően visszajelzi a kezelőfelületen, közben az elem visszahívó kódja indításra kerül. Ez a kért változással elküld egy üzenetet az üzemmódmodellnek. Ez a kérés átkerül a beágyazott érzékelő oldalra, amely egy állapotjelentést ad vissza. Amennyiben az állapot eltér a kéréstől, az üzemmódmodell értesíti a kezelőfelületet, amely frissíti az elemet, hogy a pillanatnyi állapotértéket mutassa.
A harmadik tervezési nehézséget az egér nélküli környezet iránti igény jelentette. A jármű mozgása és a tényleges munkaasztal hiánya nehézzé teszi az egér, a trackball vagy az érintőképernyő használatát. Egy billentyűzet rendelkezésre áll ugyan, de csekély mennyiségű adatbevitelre használatos. Ebből kifolyólag úgy terveztük, hogy a kezelőfelületet kézi vezérlőkarral lehessen kezelni.
Az egér nélküli módot a kezelőfelület korai változatában kézi elembejárási útvonalak és gombnyomásesemények hozzáadásával értük el. A vezérlőkaron lévő sapkakapcsoló (hat switch) mozgatása az `XmProcessTraversal` hívásával változtatta az elem fókuszálást. A kiválasztó (Select) gomb megnyomása egy, az alábbihoz hasonló `XEvent`-et határozott meg és küldött el:

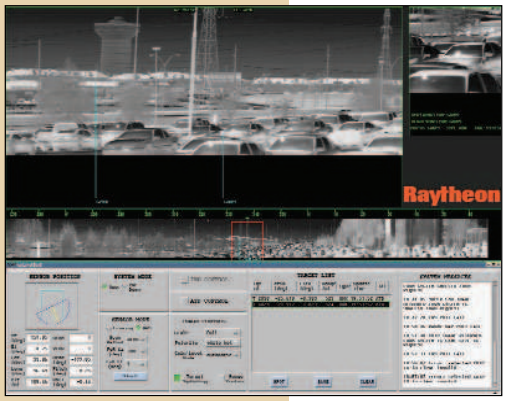
```
/* sending key press events */
#include <X11/keysym.h>

XKeyEvent ev;
Window     rootWin;
int        x,y;
int        root_x,root_y;
Window     win;

rootWin =  RootWindowOfScreen
➥(guiScreen);

win = findPointerWindow(rootWin,
➥&x, &y, &root_x, &root_y);

ev.type = (long) KeyPress;
ev.send_event = True;
ev.display = display;
ev.window = win;
ev.root = rootWin;
ev.subwindow = 0;
ev.time = CurrentTime;
ev.x = 1;
ev.y = 1;
ev.x_root = 1;
ev.y_root = 1;
ev.state = 0;
ev.same_screen = True;

ev.keycode =
➥XKeysymToKeycode(display,XK_space);

XSendEvent(display, window, True,
➥KeyPressMask, (XEvent *)&ev);
```

A kezelőfelület jelenlegi változatától eltérően az előző változat mindössze egy `topLevelShell`-ből állt, amely csak egyszerű elemeket tartalmazott, például ilyen a `PushButtons` és a `ToggleButtons`. A jelenlegi

kezelőfelületben több héj (felugró ablakok) és összetett elemek találhatók, mint például az *OptionMenus*.
Az `XmProcessTraversal` egyszerű meghívása a fókusz megváltoztatására nem működik héjak között. Egy gombnyomást az *OptionMenu*-n elküldve a menü felugrik, mindazonáltal egy második gomb megnyomása nem választ ki egy újabb lehetőséget, és nem tünteti el a menüt.
Néhány jó tanács olvasóinknak – nem árt, ha az alábbiakról nem feledkezünk meg:

1. Az ablakkezelő a főnök. Amikor több héjjal akad dolgunk, emlékezzünk rá, hogy az ablakkezelők nem igazán adják át a fókuszt vagy bármilyen hasonló feladat vezérlését.
2. Az elemhierarchia hatása: egy elemkészlet csoportjában a bejárási út rendjét részben az a sorrend határozza meg, ahogyan a kódban meg lettek adva (declared).
3. Bánjunk óvatosan a színfalak mögötti kóddal! Nézzünk egy `RadioBox`-ot, amely két `ToggleButton` gyermeket tartalmaz, ezek közül az A van kiválasztva. Amikor egy, a B-t kiválasztó üzenet érkezik, a gyermek `XmNset` erőforrások értékeinek egyszerű felcserélése a képernyőn helyesnek látszik. A szülőelem eközben még mindig azt hiszi, hogy az A van kiválasztva, ami nem várt `Button` működéshez vezethet.

Projektünk jelenlegi változatában egy külön folyamat foglalkozik a kézi vezérlőkarról érkező bemenettel, és vezérli az egérmutatót – az `XWarpPointer` és az X-kiszolgáló `XTest` bővítményének kombinációját használva (lásd a *Kapcsolódó címeket*). A munkaállomás a beágyazott oldalról küldött adatokból létrehozott videójelet is megjelenít. A videóátviteli folyamat ezt egy foglalatból olvassa ki és egy ablakba továbbítja. Négy ablak van: mozgó videó, WAS, SFOV és képdarabok.
Mint már említettük, a mozgó videó az élő képadat.
A WAS ablak képe egy összenyomott képcsík, ami egy állandó területet gyorsan pásztázó érzékelőtől érkezik. A WAS-csík jelrendszere többek között jelzi azokat a helyeket, amelyeken a célzó rendszer szerint célpontok találhatók. Az SFOV egy több részletet mutató nagyobb nézet a WAS-csík egy részéről, amit a kezelő választhat ki. Ezen láthatók a célzó szimbólumok és adatok a digitális térképről. A képdarabok a terep azon részeit mutatják, amelyeken a célzórendszer valami érdekeset talált. Ezeket megmutatja a kezelőnek értékelésre és jelenti más, a járművön kívül lévő rendszereknek. A 2. képen egy mozgó videómódban készített külső nézet látható a kezelőfelülettel, a videóablakkal és a WAS-csíkkal.
A 3. képen a rendszer WAS-módban van, látható rajta a WAS-csík, az SFOV ablak, a kezelőfelület és egy képszeletablak.
A videót OpenGL-ben egy poligonon lévő textúraként alakították ki, a videóból érkező adat erre a textúrára kerül. Amikor a poligon képernyőre kerül, akkor a videó is látható (lásd a példakódot az 53. CD Magazin/RSTA könyvtárában ami ezt a technikát szemlélteti). Azért választottuk az OpenGL-t a videóhoz, mert számos lehetőséget kínál az adat feldolgozására és megjelenítésére. A kép átméretezhető vagy elforgatható, ha más tájolásban készült, mint ahogy meg van jelenítve. Az OpenGL számos primitívet tartalmaz a szimbólumok képekre rajzolásához; a villogásmentes frissítéshez képfeldolgozó képessége és kettős átmeneti tárazása révén járul hozzá. Az OpenGL hordozható és jól dokumentált, ráadásul a processzorról egy csomó munkát átirányíthatunk a grafikus kártyára.



3. kép
A munkaállomás WAS-módban

Az SFOV kiválasztója vezérli, hogy a WAS-csík mely része kerüljön megjelenítésre az SFOV ablakban. Ezenkívül azt is szabályozza, hogy a piros négyszög a WAS-csík ablakán belül hova kerüljön kirajzolásra.
A munkaállomás egy külön vezérlő- és módválasztó modullal bír. Ahelyett, hogy a logikai egységek szét lennének szórva a rendszeren belül különböző modulokra, ebben a modulban koncentrálódnak. Ez a kialakítás a rendszer többi részét egyszerűbbé és könnyen újra felhasználhatóvá teszi. Ugyanakkor a módválasztó modul nagyon összetett. A módmodellnek meg kell tudnia valósítania mindazt az ismeretet, hogy miként hatnak egymásra az egységek és hogyan tükrözik a beágyazott rendszer és a munkaállomás állapotát. Lehetővé teszi a munkaállomásnak, hogy az összegyűjtött adatokra alapozva engedélyezett akciókat végezzen, és hogy a beágyazott oldalt hibák és a helyzet váratlan megváltozása után kutatva megfigyelje.

Milyen gyors az elég gyors?

A munkaállomás az éppen valós idejű kategóriába esik: a rendszer nem hibázik, ha valami késik. Mivel ez egy emberi tényezőt is tartalmazó próbarendszer, amelynek a legtöbb időkritikus összetevője a beágyazott rendszerekben rejlik, csak olyan gyorsan kell futnia, hogy a kezelő végre tudja hajtani a feladatokat. Ezért nem használjuk a valós idejű Linux-keretrendszerek egyikét sem, ehelyett nagy teljesítményű alkatrészekkel oldottuk meg a feladatokat: SCSI merevlemezzel, elég memóriával, hogy kiküszöböljük a lapozást, és egy GeForce4 grafikus kártyával a gyors OpenGL miatt, valamint két 2,4 GHz-es processzorral a Microwaytől.
Egy korlát emelkedett a rendszer két része, a videó és az érzékelő célzása elé. Az élő videó RS-170 sebességgel kerül betáplálásra a munkaállomásra, egy félképnyi képsor minden 1/60-ad másodpercben. Ezeket olyan gyorsan kell összerakni, és megjeleníteni, hogy állandó se-

bességet lehessen elérni és fenntartani. Hogy ezt lehetővé tegyük, biztosítottuk, hogy elegendő hálózati sávszélesség legyen a videójel szállítására és elégséges processzor- és grafikus teljesítmény a megjelenítés frissítésére. A monitor frissítését 60 Hz-re beállítva helyben is voltunk (lásd az nVidia meghajtóprogramhoz mellékelt *README.txt* fájlt, a legfrissebb a ➲ http://dowload.nvidia.com/XFree86_40/1.0-4194/README címről tölthető le).

Az érzékelő irányítása hasonló kihívást jelentett. Ennek elég érzékenynek kell lennie a vezérlőkarokra, hogy a kezelő a cél elvesztése nélkül tudjon vele célozni. Míg a videó leginkább a sávszélességtől függ, az érzékelő irányítása a késletetésen múlik. Egy hosszú üzenetlánc késleltetéshez vezet. Például egy gombnyomás vagy a vezérlőkar-kezelőfelületről származó elfordulás parancs eljut a vezérlőfolyamathoz, hogy megállapítsa, melyik bemenet érvényes, majd továbbmegy a fordítóhoz, hogy EO formájú üzenet legyen belőle. Ezután keresztülhalad a gigabit ethernet hálózaton egy beágyazott folyamatba, amely fogadja az üzenetet, majd továbbítja az OE-be és a beágyazott rendszer kódjába, onnan a működtető szerkezetre kerül, végül az eredmény visszaérkezik a videoadatfolyamba. A fordítófolyamatok két szálra választása és a teljes optimalizálással történő fordítás (`-Wall -ansi -O3 -ffast-math -mpentiumpro`) megtette a magáét.

A `gprof` profilert használtuk, hogy megkeressük, vannak-e kényes helyek a kódban (lásd a `gprof` információs oldalát). Itt a videókód profileozása közben komoly nehézségbe futottunk: amikor X-időzítőt használtunk (`XtAppAddTimeOut`), nem került időzítési adat a profileba. (Talán a profiler és az `XtAppAddTimeOut` ugyanazt a jelet használták, és egymást zavarták?) A másik egyszerűsítési lehetőség, amire rájöttünk: a videóforráshoz a páros és a páratlan sorok – két kisebb helyett – egyetlen paranccsal történő továbbítása a hálózaton.

Előnyök, kelepcék és következtetések

A Linux használata néhány gondot is okozott, például nem találtunk olyan beszállítót, aki PCI Mezzanine kártyát tudott volna szállítani PowerPC-hez VxWorks-meghajtóval, vagy PCI kártyákat Linux-meghajtóval, illetve aki kezelni tudta volna a VI protokollt. Végül el kellett vetnünk az optikai kábel használatának ötletét.

Számos alkalommal viszont azt tapasztaltuk, hogy a Linux használata előnyt jelentett. Mivel a rendszert merevlemezről indítottuk, nem kellett EEPROM-ba égetni, mint a beágyazott rendszereknél. Amikor ott EEPROM-ba kerül a kód, sokkal nehezebbé válik a hibakeresés. Ezenkívül a Linux magfájlokat kínál a hibakeresés segítésére, amit a VxWorks nem tesz meg. A Linux-munkaállomás jóval masszívabb felépítésű és jobb képminőséget nyújt, mint elődje. Végül a műhelyben történő összeállítás és az egységek kipróbálása könnyebb Linux alatt, mivel a kereskedelmi forgalomban kapható PC-k jóval elterjedtebbek, mint a beágyazott PowerPC-k. Arra számítunk, hogy a jövőben a Linux, az X és az OpenGL-környezet teljesítménye és rugalmassága egyre kifizetődőbb lesz, ahogy több módot és eszközt adunk prototípusunkhoz.

Linux Journal 2003. október, 114. szám

George Koharchik (g-koharchik@raytheon.com)
A Raytheon's Visualization és Simulation Lab (VSL) munkatársa, szabadidejében a biztosítótű mechanikáján elmélkedik.

Quintelle Griggs
(Quintelle_Y_Griggs@raytheon.com)
A Raytheon's VS munkatársa.

Sonja Gross (sonja_gross@raytheon.com)
2001-től a Raytheon's VSL munkatársa, miután megszerezte baccalaureátusi fokozatát számítástechnikából a Louisiana Tech Universityn.

Kathy Jones (kajones@raytheon.com)
Programfejlesztő a Raytheon's VSL-nél. Motif VAPS kezelői felületeket és egyéb programeszközöket készít.

John Mellby (j-mellby@raytheon.com)
A Raytheon's VSL munkatársa, virtuális szimulációval foglalkozik, rövid biográfiákat ír, és megméri a texasi napot a tavaszi napéjegyenlőség idején.

Joe Osborne (joe.osborne@smiths-aerospace.com)
A Smiths Aerospace munkatársa a Michigan állambeli Grand Rapidsben.

KAPCSOLÓDÓ CÍMEK

BX ➲ http://www.ics.com/products/bxpro
Integrated Computer Solutions ➲ http://www.ics.com
nVidia ➲ http://www.nvidia.com
Microway ➲ http://www.microway.com
Eric F. Johnson és *Kevin Reichard* Power Programming...Motif; Management Information Source, Inc. Második kiadás Version 1.2, 1993., New York. ISBN: 1-55828-319-6.
Raytheon ➲ http://www.raytheon.com
NightSight ➲ http://www.raytheon.com/products/tiger
Nem hivatalos VxWorks és Tornado GYK
➲ http://www.xs4all.nl/~borkhuis/vxworks/vxfaq.html
A VxWorks és a Tornado a Wind River Systems termékei
➲ http://www.windriver.com
VxWorks, illetve Tornado II GYK (különösen a 4.6-os rész a foglalatokról)
➲ http://www.xs4all.nl/~borkhuis/vxworks/vxworks.html
WSTAWG weboldal ➲ http://wstawg.army.mil/index.asp
XTest bővítmények XFree86-hoz
➲ http://xfree86.org/pub/XFree86/4.2.0/doc/xtestlib.TXT