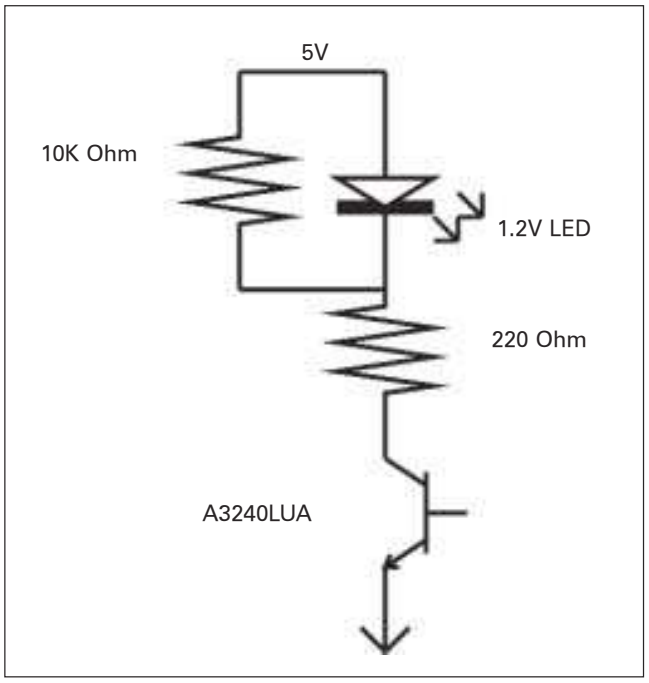
Szennyvízszivattyú a weben, beágyazott Linuxszal

Egy egyszerű áramkörrel bármilyen váltakozó árammal üzemelő elektromos készülékről megállapíthatjuk, hogy be van-e kapcsolva, és az adatokat weben keresztül is elérhetővé tehetjük.

Feleségemmel 1996 nyarán vásároltuk meg a házunkat. 1997 kora tavaszán az alagsor tele lett vízzel, rá egy évre a jelenség megismétlődött. 2001 tavaszán az árvíz újfent ismét elöntötte az alagsort. Ekkor egy kissé kezdtük unni a dolgot, és elhatároztuk, hogy felszerelünk egy szennyvízszivattyút. 2003-ban, amikor rápillantottunk arra a több mint egy méter magas hókupacra, amely csakis arra várt, hogy elolvadva a mi pincénkbe szivárogjon be, eszünkbe jutott, hogy végül is a szivattyú felszerelésére azóta sem került sor. Ekkor keményen elhatároztuk, hogy cselekedni fogunk, és meg is tettük. Mindössze egy hétig kellett fúrókalapáccsal ügyeskedni a pincében, és máris körbecsövezett pincével, valamint egy csinos kis szivattyúval dicsekedhettünk, amely akkumulátoros tápot és tetszetős borítást is kapott. Mivel kételkedtünk abban, hogy kedves kis szivattyúnk képes-e megküzdeni a hólével, kissé rögeszmévé válva egyre többet forogtak a gondolataink a betörő víz körül. Jómagam tízpercenként ellenőriztem a vízszintet. Éjjelente azért keltem fel, hogy a vizet nézzem. A munkahelyemről rendszeresen hazatelefonáltam, hogy helyzetjelentést kérjek – szóval, az egész kezdett komédiába illő lenni.
Az a szerkezet, amit a saját nyugalmam érdekében állítottam üzembe, egyre jobban felemésztett. Végül úgy döntöttem, egy olyan megoldás kell, amivel távolról is figyelni tudom a szivattyú működését. Meghatároztam házi tervezetem fő céljait:

1. ne égjen le a ház;
2. a szivattyú működését a figyelőeszköz nem gátolhatja meg;
3. tanulni akarok valami újat.

Első és második számú célommal összhangban úgy határoztam, hogy a szivattyú áramellátásával semmit nem szabad sorba kötnöm. Nyilvánvaló volt, hogy a saját tervezésű áramkörömön – biztonsági okokból, például a jelek feldolgozását végző processzor leválasztásának és védelmének nehézségei miatt – inkább ne folydogáljanak 10 amperes áramok. Az egyik lehetőség az volt, hogy a szivattyú elektromos vezetékére egy másik vezetéket tekercselek. Finomhangolás után a tekercsben indukált áramot a processzor érzékelni tudta volna. Sajnos ezt a rendszert megépíteni – figyelembe véve az otthon rendelkezésemre álló eszközöket – túl sokáig tartott volna.
Jó néhány ötlet elvetése után a Google-höz folyamodtam segítségért. Kutakodásom közben véletlenül felidéződött bennem a Hall-hatásnak nevezett jelenség. A Hall-hatás a mágneses mezőben áramló elektronokra ható Lorentz-erő megnyilvánulása. A Lorentz-erő az elektromos és a mágneses mezőre egyaránt merőleges, hatására és irányában az elektronok eloszlása egyenetlenné válik. A vezető felületén indukált ez irányú feszültség arányos a mágneses mező erősségével, így alkalmas az erősségének a mérésére. A Hall-hatást rengeteg különféle, a kereskedelemben könnyen beszerezhető műszerrel mérni lehet, ezek a belső jelfeldolgozás milyenségében és a mágneses erők érzékelésének finomságában térnek el egymástól. Saját céljaimra az Allegro Microsystems A3240LUA típusa tűnt megfelelőnek. Ez egy kellően érzékeny, egypólusú érzékelő, adatlapját a ➲ http://www.allegromicro.com/sf/3240 címen meg lehet tekinteni. Az egypólusú érzékelők alapjában véve olyan NPN tranzisztorként viselkednek, amelynek akkor van bázisárama, ha az eszköz közelében déli mágneses pólus van jelen. Próbálgatás céljából beszereztem néhány ilyen érzékelőt.
Az volt az elképzelésem, hogy a távoli érzékelő mindössze a Hall-hatásra épülő eszközből állna, és ez a jelfeldolgozó általános be- és kiviteli lábára (GPIO) csatlakozna. A programoldal hibáinak a keresését különálló áramkörrel akartam megkönnyíteni, amely a szivattyú működését egy LED-del jelezte volna. Így legalább arról meg tudtam volna bizonyosodni, hogy az érzékelő valóban érzékeli-e az átfolyó áramot. Így készítettem el az *ábrán* látható áramkört.
Csatlakoztattam az elemeket, megmozgattam egy mágnest az érzékelő előtt, és a LED várakozásaimnak megfelelően kigyulladt. A következő lépés a szivattyú elindulásának a megvárása, majd az érzékelőnek a tápkábel mellett való mozgatása volt, ezzel megbizonyosodtam arról, hogy valóban képes a közelében lévő mező érzékelésére. Vártam, mozgattam, semmi… Vártam, mozgattam, még mindig semmi. Úgy tűnt, hogy a mező belső és külső határai közelebb voltak a vezetékhez, mint hittem. A fázis- és a nullavezetékek mágneses mezői elég erősek voltak ahhoz, hogy kioltsák egymást, így képtelen voltam mérni őket. Teljesen mindegy volt, hogy hova helyeztem az érzékelőt, az áramot nem lehetett mérni. Természetesen nem adtam fel ilyen könnyen, inkább módosítottam a terveimen. Egy régebbi, 15 amperre méretezett hosszabbítóval és egy lágyvas maggal felszerelkezve nekiláttam a mező felerősítésének: tízszer körbetekertem a nullavezetékkel a vasmagot. Művemet néhány kábelkötegelővel és némi ragasztóval tettem teljessé. Módosított kábelemmel ezt követően újra megindultam a pince felé. Csatlakoztattam a szivattyút a hosszabbítóhoz, majd vártam, amíg működésbe lépett. Miután a szivattyú újra bekapcsolt, végre sikerült az érzékelőt elég közel tenni a vasmaghoz, és érzékeltetni vele a mezőt. Kevéske falap, hangyányi forrasztás, még egy kis ragasztó, és máris készen állt a végleges érzékelő. A *képen* ez látható.
Következő lépésként el kellett döntenem, milyen processzort használjak a jelek feldolgozására. A fő szempont az ár, a beépített ethernetcsatoló, a használható GPIO-k és a Linux futtatásának lehetősége volt. Némi kutakodás után arra jutottam, hogy rengeteg olyan beágyazott mikrokontroller van, amely ethernetcsatolóval és a feladathoz elegendő teljesítménnyel egyaránt rendelkezik, ám a legtöbbnél nem említették kifejezetten a Linuxot. A kínálat másik végén a PC/104 osztályú beágyazott személyi számítógépek kellették magukat, igaz, jóval nagyobb teljesítménnyel és magasabb árral, mint amennyit én gondoltam el erre a célra. Végül a Soekris Engineering Net4501

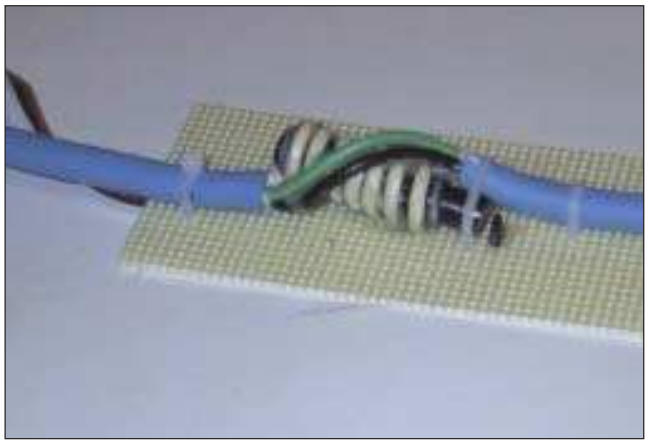


A próbaáramkör vázlata

jelzésű, egyetlen áramköri lapból álló számítógépe mellett döntöttem, amely CompactFlash foglalattal, 64 MB RAM-mal, AMD Elan processzorral és beépített ethernetcsatolóval rendelkezik. Érdekessége, hogy a Preboot Execution Environment (PXE) megoldás segítségével hálózatról is képes indítani az operációs rendszert.

A Soekris webhelyén (➲ http://www.soekris.com) elérhető leírásból megtudtam, hogy az Elan GPIO lábainak jelentős része egy csatlakozó révén – egy +5 voltos tápvonal társaságában – könnyedén hozzáférhető. Az ára elfogadható volt, és mellékeltek hozzá egy tápegységet, valamint egy pofás fémdobozt, amibe be lehet szerelni az áramkört. Az Elan GPIO lábai belső behúzással vagy elengedéssel működnek. Egy belső behúzással rendelkezőt választottam magamnak, mivel így az érzékelőt további alkatrészek beépítése nélkül, önmagában is rákötöttem a processzorra.

Ezután készítettem egy hálózati rendszerindításra képes (2.4.19-es) rendszermagot, amelyen szinte mindent letiltottam. A magot modulok nélkül fordítottam le, mindössze a Natsemi Ethernet illesztőprogramot, a gyökér NFS-t, a soros konzolt és az SC520 figyelő időzítőt hagytam meg. A normál beállítások megadása mellett egy további módosítást is végre kellett hajtanom a rendszermagon. A 2.4-es sorozatba tartozó rendszermagoknál x86 alapú gépekhez az alapértelmezett időzítőmegszakítás 100 Hz-re van állítva. Mivel tudtam, hogy egy közel ekkora frekvenciájú (60 Hz-es) jelet kell majd mintavételeznem, úgy döntöttem, hogy növelem az időzítőfrekvenciát. A megszakító időzítőfrekvenciáját az ***asm/param.h*** fájlon belüli Hz-érték szabályozza. Ennek felső határa 2000; én az 1500-as érték mellett döntöttem, vagyis másodpercenként 1500 megszakítást akartam kapni. Mivel a gépen más nem nagyon futott, nem volt komoly esély arra, hogy a megnövelt frekvencia miatt a megszakításokra alapuló eljárások összevesznének egymással. Az így létrehozott rendszermag elérhetővé tételének feladatát DHCP-kiszolgálómra és a PXELinuxra bíztam. Ekkor már csak a gyökérfájlrendszernek a TFTP-kiszolgálóra való átmásolása volt hátra. A kezdeti futtatókörnyezet létrehozásához lefordítottam a legújabb uClibc, BusyBox, TinyLogin és utelnetd csomagokat. Mindhárom futtatható állományt állandó jelleggel csatoltam az uClibc-hez. A BusyBox-féle init alapállapotban egy héjat indít a konzolkapun. A további szolgáltatásokat saját ***/etc/inittab*** állományom segítségével adtam hozzá a rendszerhez. Ez engedélyezi a konzolhéjat, meghív egy egyszerű parancsfájlt, amely (újra)csatlakoztatja a gyökérfájlrendszert, engedélyezi az időzítőt, majd a `telnetd` indításával lehetővé teszi, hogy távolról is beléphessek a gépre. Tehát egy terminált csatlakoztattam a soros konzolkapura, majd újraindítottam az eszközt. A konzolkapun keresztül figyelemmel tudtam követni a rendszer betöltésének a folyamatát, majd megjelent előttem a BusyBox-féle héj.



A kész érzékelő

Miután a rendszer életre kelt, figyelmemet az új illesztőprogramok felé irányítottam. Esetemben rendszermagterületen futó eszközillesztőre egyetlenegyre volt szükség, mégpedig annak a GPIO lábnak a figyeléséhez, amelyhez az érzékelőt csatlakoztattam. Mivel a rendszermag programozásában még nem voltam jártas, úgy döntöttem, hogy a lehető legkisebbre próbálom meg szorítani annak a valószínűségét, hogy a rendszermagban valami hiba lépjen fel, és ezért a lehető legkevesebb kódot írom meg. Ennek szellemében tehát egy ***/proc*** fájlrendszerbeli illesztőprogram írása mellett határoztam, amelynek feladatául a szivattyú be- és kikapcsolt állapotának a jelzését szántam. Ha ez az alacsonyabb szinten futó illesztőprogram elkészült, akkor egy felhasználói területen futó programmal már könnyedén lekérdezhető.

Az illesztőprogram init függvénye három lényeges műveletet hajt végre. Először a `create_proc_entry` hívással bejegyzi magát ***/proc*** fájlrendszerbeli modulként. A `proc_dir_entry` visszatérési értékeként két fontos adatszerkezetet kapunk, ezek a fájl- és fájlleíró műveleti táblák. Értékül két állandó, a megfelelő értékekkel feltöltött adatszerkezetet kapnak. Mivel meglehetősen egyszerű modulról van szó, a két adatszerkezet bejegyzései leginkább NULL értékeket tartalmaznak. A ***proc*** bejegyzés létrejötte után az `init` eljárás néhány, az Elan processzorra egyedileg jellemző regiszter értékének megadásával a kívánt GPIO-t állítja be bemenetként.

Az indítóeljárás utolsó lépésként egy időzítőt indít el, amely biztosítja a bemeneti láb rendszeres lekérdezését. Az időzítőfüggvényt érdemes jobban is megvizsgálni. (A listát lásd az 54. CD Magazin/Szennyviz könyvtárában.)

Mivel a mágneses mező – illetve emiatt az érzékelő jele is – oszcillál, nem lehetett egyszerűen mintavételezni a lábon bejövő jelet, és ezt a szivattyú állapotának jelzéseként kezelni.

Mivel el akartam kerülni, hogy a rossz időzítésű mintavételezés miatt zajok jelenjenek meg a statisztikákban, egy egyszerű integrátort valósítottam meg. Amikor az időzítő elindul, a mintaszámláló a periódusonként vételezni kívánt minták számával egyenlő értéket vesz fel. Alapesetben ennek értéke Hz/60, vagyis másodpercenként 25 minta. Ne feledjük, a Hz a rendszermag időzítő megszakításának a frekvenciája, amelyet a rendszermag lefordításakor 1500-ra állítottam.
A 25 gyorsminta beolvasása után újra beállítom az időzítőt, hogy a mintacsoport vételezési időtartamának a végén ismét lejárjon. Alapesetben öt másodpercenként olvasok be egy-egy mintacsoportot. Az időzítő újra történő beállításakor integrálom is (összeadom) a gyorsminták számát. Mivel a mintavételezés elég gyorsan történik, biztosan meg tudom határozni, hogy a szivattyú üzemel-e. Ha úgy tűnik, hogy a szivattyú be van kapcsolva, megváltoztatom a `pump_state` változó értékét. A modul adatainak olvasásakor (ezt a `pump_output` függvény végzi) mindössze ennek a változónak az állapotát vizsgálom meg és adom tovább. Ez a módszer lehetővé teszi, hogy úgy játszadozzak a mintavételezéssel, hogy közben ne kelljen az illesztőprogram válaszidejének a megváltozásától tartanom.
Az illesztőprogram hibáinak keresésekor egy `verbose` kapcsolót is beépítettem, amely különféle adatok kiírását teszi lehetővé a naplófájlba. A modult a `verbose=1` kapcsolóval futtatva például a mintavételi átmeneti tár tartalma íratható ki, amikor a szivattyú működni látszik. Így egy egyszerű oszcilloszkópfüggvénnyel a naplófájl tartalma alapján rendszeresen ellenőrizni tudom, hogy nem kapok-e hibás eredményt. Egy másik, ugyancsak érdekes lehetőség is kínálkozik. Mivel ismerem az érzékelő kioldási pontját és a pontok közötti fázisszöget (időt), ki tudom számolni a mágneses mező erősségének csúcsértékét az érzékelőnél. 1500 Hz-es órajelnél az érzékelő öt mintavétel idejéig van bekapcsolva. A mező 60 Hz-es frekvenciával oszcillál, vagyis a minták között 0,4 PI radián fázisszög van.
Az érzékelők kioldási pontjaira a legrosszabb esetet feltételezve – 50 Gauss felső és 5 Gauss alsó határérték – és az alábbi egyenleteket megoldva az amplitúdó csúcsértékére 51,4 Gausst kapok. Általános értékeket véve (35 és 25) a mező csúcsértéke nagyjából 38 Gauss:

```
A * sin( theta ) = 35
A * sin( theta + 0,4PI ) = 25
```

Ugyancsak a hibakeresést segítette egy másik illesztőprogram, amely egy másik GPIO-hoz csatlakoztatott LED-et kapcsol be, ha erre utasítom. Ez az illesztőprogram hasonló a szivattyúéhoz, ám ennek kimeneti függvénye – amely a modulról való olvasást teszi lehetővé – nem tesz szükségessé kifinomult mintavételezést, valamint bemenettel is rendelkezik, amely lehetővé teszi, hogy a felhasználó a modulra írva beállítsa a GPIO láb állapotát. Ez az új függvény (`led_input`) egy felhasználói területen található átmeneti tárból olvas, majd ennek alapján dönti el, hogy mire kell állítania a láb állapotát. A függvény bejegyzése a `file_operations` adatszerkezettel történik.
Ez az illesztőprogram a szivattyúhoz készítettől szerkezetileg annyiban tér még el, hogy a fájlra vonatkozó engedélyek között (ezeket a `create_proc_entry` hívásnál adjuk meg) lehetővé kell tenni az írási hozzáférést. Az illesztőprogram egy egyszerű parancsfájllal társítva alkalmas arra, hogy a szivattyúnál tájékoztatást adjon a programok működéséről: ha a LED be- és kikapcsolása követi a szivattyúét, akkor minden rendben van és működik.
Miután az alapvető illesztőprogramok a helyükre kerültek, a többi építőelem beillesztésével összeállhatott a rendszer. Szükség volt egy felhasználói térben futó démonra, amelynek segítségével távolról is le lehetett kérdezni a szivattyú állapotát. Mivel a gyökérfájlrendszer NFS-en keresztül van a központi kiszolgálómról befűzve, elég lett volna egy parancsfájlt készítenem, amely egy időre mindig elaludt volna, majd időnként ellenőrizte volna a */proc/pump* állapotát, és egy valódi fájlba írta volna a kapott eredményt. Ezúttal azonban a könnyebb megoldás helyett a `pumpserv` megírása mellett döntöttem.
A `pumpserv` egy egyszerű démon, amely az 5678-as kapun fogadja a kapcsolatokat, majd a */proc/pump* teljes tartalmát a másik fél felé másolja át. A csővezeték másik végén a `pumpwatch` helyezkedik el. A `pumpwatch` egy másik démon, amely a gazdagépen fut, és rendszeresen ellenőrzi a szivattyú állapotát, valamint rögzíti az állapotváltozások időpontját. Az átmenetekhez időbélyeget csatol, majd az adataikat egy naplófájlba írja. A naplófájlt később további programokkal fel lehet dolgozni, tartalma alapján kimutatásokat lehet készíteni, illetve valamilyen webhelyre feltöltve az egész világon elérhetővé tehető.
A rendszer 2003 áprilisa óta folyamatosan működik. Mivel a jelek szerint hibátlanul üzemel, nyugodtan sikerként könyvelhetném el, és túlléphetnék ezen a gondon, ám nem tudok szabadulni a `pumpserv2` megírásának a gondolatától. Ha valaha is nekiállok, egy pár dolgot másként fogok csinálni. A jelenlegi rendszer egyik súlyos hibája, hogy a gyökérfájlrendszert egy NFS-kiszolgálóról veszi, az adatokat pedig egy a kiszolgálón futó démon veszi át. Üzembiztosabb megoldást kaptam volna, ha a gyökérfájlrendszer helyi lenne, és mivel a Soekris Net4501 rendelkezik CompactFlash foglalattal, ennek megoldása nem is volna lehetetlen. Jó lenne, ha az adatok naplózását a szivattyú oldalán helyben végezné egy démon, majd kérésre elérhetővé tenné őket. Így a központi kiszolgáló leállása nem okozna adatvesztést.
Ugyanezzel a rendszerrel – apróbb módosítások árán – más eszközöket is lehetne figyelni, feltéve, hogy az érzékelő számára észlelhető nagyságú áramot vesznek fel. Néhány példa a sok közül: légkondicionálók, hűtőgépek, izzócsöves fűtőgépek, búvárszivattyúk. A leghasznosabb – és legfontosabb – alkalmazási lehetőség talán az irodai kávéautomatában található kávé mennyiségének a követése. A fűtőszálak bekapcsolásának száma ugyanis fordítottan arányos a még lefőzhető kávé mennyiségével.
Ha valaki a nyomdokomban haladva hasonló rendszert szeretne építeni, akkor az indító parancsfájlt és az illesztőprogramok, a `pumpwatch` és a `pumpserv` forráskódját a ➲ http://pumps.oldtools.org/src címről érheti el. A szivattyúfigyelő fájlrendszere ***.tar*** állományba tömörítve szintén hozzáférhető. Ha valaki tudni szeretné, hogy a pincémet fenyegeti-e árvíz, szivattyúm állapotát a ➲ http://pumps.oldtools.org címen ellenőrizheti.
A forráskód a Linux Journal FTP-helyéről, az ➲ http://ftp.ssc.com/pub/lj/listings/issue113/6827.tgz címről ugyancsak letölthető.

Linux Journal 2003. szeptember, 113. szám

Tad Truex
Napközben Alpha processzorok tervezésén dolgozik a HP-nál. Éjjel két gyermeke és számos hobbija között próbálja megosztani az idejét.