Adatgyűjtés a Comedi segítségével

Létezik egy szabványos alaprendszer, amely számos különböző adatgyűjtő kártyához egységes API-t biztosít. Aki akarja, akár egy normál számítógép párhuzamos kapuján keresztül is kipróbálhatja.

A legtöbb kutató és mérnök imádja az adatokat. Minél több adatot kapnak, annál szélesebb mosoly ömlik szét az arcukon. Ezeknek az embereknek a mérés a mindene. A folyamatok felismeréséhez, a rendellenes jelenségek felfedezéséhez és a végső következtetések levonásához a laborokban dolgozóknak ügyelniük kell arra, hogy teljes adathalmazokat gyűjtsenek.

Az adatgyűjtés fogalma számos ötletet és fogalmat ölel fel. A legtöbb kutató és mérnök ugyanakkor egyetért abban, hogy az adatgyűjtés valamilyen természetes folyamat jellemzőinek mérését jelenti. Lehet szó egyszerű hőmérsékletmérésről, vagy akár olvadt acél szennyeződéseinek vizsgálatáról.

A számítógépek világában az adatgyűjtés általában valamilyen feszültség mérésével történik. Ilyenkor rendelkeznünk kell valamilyen érzékelővel vagy műszerrel, amely a számítógép számára mérhető feszültségeket állít elő. Fontos persze az is, hogy ismerjük a mért jellemző és az érzékelő feszültségkimenete közötti összefüggést. Jó esetben a kapcsolat lineáris. Ilyen például az, ha egy foknyi hőmérsékletkülönbséghez mindig 0,1 voltos feszültségeltérés tartozik.

A korszerű alaplapokon beépített érzékelők találhatók, amelyek képesek a teljes rendszer állapotáról képet alkotni. Ilyen eszköz például a ***National Semiconductor LM78*** jelű terméke. Ezek az érzékelők a gyakorlatban általában a hűtőventilátorok fordulatszámát, a processzormag feszültségét és hőmérsékletét, illetve a merevlemezek fordulatszámát mérik. Az adatokat maga a lapka gyűjti össze, a processzorhoz pedig egy soros buszon keresztül juttat el őket. A nyílt forrású ***lm_sensors*** projekt (➲ ***secure.netroedge.com/~lm78***) programja az alaplapok működésének figyelését számos szempont szerint teszi lehetővé.

A normál személyi számítógépek ettől függetlenül nem rendelkeznek általános, analóg adatok begyűjtésére alkalmas csatolófelülettel. Így ha külső feszültségeket akarunk mérni velük, akkor új felületre van szükségünk. Pontosan ilyen célra tervezik a ***PCI*** és ***ISA*** buszos kivitelben egyaránt létező adatgyűjtő (data acquisition, DAQ) kártyákat, amelyek számos gyártó termékkatalógusaiban megtalálhatók.

1. táblázat *Gyakoribb adatgyűjtő csatorna típusok*

Név	Leírás
Analóg bemenet	Külső jelek, például feszültségek mérése
Analóg kimenet	Változó jel küldése
Digitális be/kimenet	Diszkrét be/ki jel küldése, általában 0 volt a kikapcsolt, 5 volt a bekapcsolt állapot
Számláló	Impulzusok számlálására vagy frekvencia mérésére alkalmas
Időzítő	Két digitális impulzus beérkezése közötti időt méri

A Comedi projekt

A legtöbb Linux-felhasználó alighanem jól tudja, miféle gondokkal jár akár csak egy-egy nyomtató kapcsán a gyártók, a márkák, a típusok és az illesztőprogramok követése. Általános jelenség, hogy bármilyen szabványosítási törekvés csakhamar hatalmas fejlesztési projektté növi ki magát. Ha a tervezet kellő támogatást kap, akkor szabvány születik belőle. Vannak gyártók – ilyen például a ***National Instruments*** – amelyek eleve adtak ki linuxos illesztőprogramokat adatgyűjtő eszközeikhez, és vannak, amelyek nem.

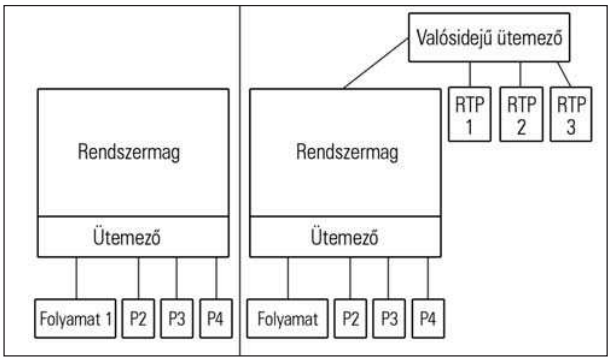
A ***Comedi***, vagyis a ***Control and Measurement Device Interface*** (vezérlő- és mérőeszköz felület) a linuxos adatgyűjtő illesztőprogramok és könyvtárak szabványos készlete. Fejlesztését 1996-ban ***David Schleef***, kezdte meg.

A ***Comedi*** célja több kártyagyártó és -modell támogatása egyetlen közös felületen keresztül. Az API tervezésekor lényegében a modularitás és a bonyolultság között kellett egyensúlyt teremteni. A többi linuxos illesztőprogram fejlesztéséhez hasonlóan a munka egy részéhez rengeteget kell búnyi a különféle eszközök leírásait, más részéhez visszafejtéseket kell végezni, bár természetesen a gyártók által a ***Comedi*** fejlesztéséhez a saját termékükre kiterjedően nyújtott támogatást sem szabad elfeledni.

Hogyan működik?

A ***Comedi*** két részből áll. Maga a ***Comedi*** illesztőprogramok gyűjteménye, amelyeket a rendszermag térbe kell betölteni, a ***comedilib*** pedig ezekhez az illesztőprogramokhoz nyújt

1. kép Légáramm mérő eszköz



2. ábra A normál és a valós idejű linuxos folyamatütemezés összehasonlítása

hozzáférést a felhasználói térből. A ***Comedi*** átlátszósága a ***comedilib*** szolgáltatásainak köszönhető. ***Comedire*** épülő programokat *C* vagy *C++* nyelven lehet írni, ezen kívül ***Perl*** és ***Python*** kötések is léteznek hozzá.
A ***Comedi*** mindent csatornákra, aleszközökre és eszközökre oszt fel. A csatorna a legalacsonyabb mérési vagy vezérlési szint. Több azonos típusú csatorna együtt egy aleszközt alkot, a teljes eszköz pedig ilyen aleszközökből áll össze.
A ***Comedi*** használatakor először egy ***Comedi*** illesztőprogram töltődik be a memóriába. Ezután a `/usr/sbin/comedi_config` fut le, amely létrehozza az illesztőprogram kötését a megfelelő ***Comedi*** eszközhöz, ez lehet például a */dev/comedi0*. Végül a DAQ-kártya által biztosított szolgáltatásokat a ***comedilib*** függvényei révén lehet elérni.

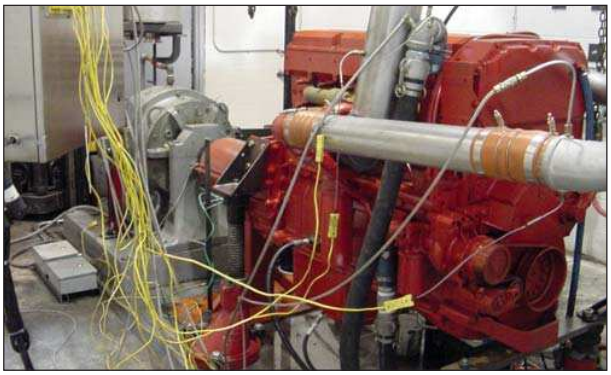
Laboratóriumi példa

DAQ és a ***Comedi*** együttes használatára többek közt az ***Analytical Engineering, Inc. (AEI)*** aerodinamikai laboratóriumában láthatunk példát. Itt egy ventilátor légáramát különféle méretű nyílásokon vezetik keresztül, a technikusok pedig egy egyedi program segítségével követni tudják a nyílásokban létrejövő nyomás alakulását. A mérés alapján ki lehet számítani a nyílásokon átáramló légtömeg nagyságát. Ezek a mérések és számítások alapvető jelentőséggel bírnak, mivel a technikusok ezek alapján tudják ellenőrizni, hogy a különféle műszerek kalibrálása helyes-e.

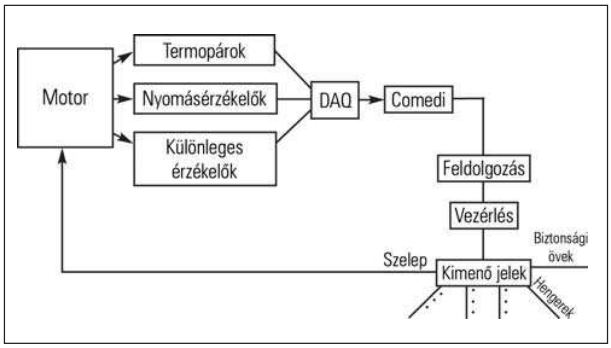
1. kódrészlet Példaprogram feszültségszint lekérdezésére az 1-es csatornáról

```
#include <stdio.h>
#include <comedilib.h>
const char *filename = "/dev/comedi0";
int main(int argc, char *argv[])
{
  lsampl_t data;
  int ret;
  comedi_t *device;
  /* A kártya melyik eszközét akarjuk
     használni? */
  int subdevice = 0;
  /* A csatorna kiválasztása */
  int channel   = 0;
  /* Az elérhető tartományok egyikének
     kiválasztása */
  int range     = 0;
  /* Mérések végzése földhivatkozással */
  int analogref = AREF_GROUND;
  device = comedi_open(filename);
  if(!device){
    /* Nem tudtuk megnyitni az eszközt - hiba
       történt */
    comedi_perror(filename);
    exit(0);
  }
  /* Adat beolvasása */
  ret=comedi_data_read(device,subdevice,
    channel,range,analogref,&data);
  if(ret<0){
    /* Valamilyen hiba történt */
    comedi_perror(filename);
    exit(0);
  }
  printf("A következő adatot kaptuk: %d\n",
         data);
  return 0;
}
```

A tényleges tömegáramlást még nehezebb pontosan kiszámítani. Meghatározásához kétféle légnyomást, három légáram hőmérsékletet, a nedvességtartalmat, a légköri nyomást és a magasságot kell ismerni. Mindezeket a mennyiségeket hétköznapi kereskedelemben kapható eszközökkel is át lehet alakítani feszültségekké. Az egyik legnépszerűbb csatolófelület az ***5B***. Ilyen típusú moduláris blokkokat használva mindezeket a mennyiségeket a DAQ kártya által olvasható feszültségekké lehet formálni. A ***Comedi*** segítségével mindezeket a feszültségeket könnyedén, a `comedi_data_read` függvény meghívásával tudjuk beolvasni. Ez a kiválasztott csatornát figyelve egyszerűen egy értéket ad vissza, amely lehet például 3421. De vajon mit jelent ez a szám?
A DAQ kártyák meghatározott felbontással dolgoznak. A leggyakoribb a 12 bites pontosság. Mindegyikhez egy adott feszültségtartományban dolgozik, amelyre a mérési eredményeket leképezi. Mivel 12 biten a 0-4095 értéktarto-



3. kép Motor vizsgálat közben



4. ábra A motorok vezérlése és jellemzőinek Comedi segítségével végzett mérése

mány ábrázolható, könnyen kiszámíthatjuk, hogy a 3421 jelentése: 3421/4095 *(a teljes mérési tartomány). Ha például eszközünk a [0, 5] feszültségtartományban dolgozik, akkor a 3421 értéknek 4,177 voltos feszültség felel meg. Mindezen adatok birtokában, illetve tudva, hogy az 5B blokk a [0 volt, 5 volt] feszültségtartományra a [0 °C – 100 °C] hőmérséklettartományt képezi le, néhány pillanat alatt ki tudjuk számítani, hogy a fenti feszültség 83,56 °C-os hőmérsékletet jelez. Gyúrjuk össze a szükséges méréseket, csapjunk hozzá egy csinos grafikus felületet, és ismételjük meg a mérést minden másodpercben.

Természetesen ennél bonyolultabb adatgyűjtéseket is végezhetünk. Az adatok begyűjtésekor fontos szempont a sebesség. Elég gyorsnak kell lennünk ahhoz, hogy a minták között semmilyen fontos adatot ne hagyjunk figyelmen kívül. Ennek elérésére a ***Comedi*** egy parancsfelületet is biztosít, amellyel szinkronizált mintavételt lehet végezni.

A DAQ-kártya képességeitől függően az időzítésről program alapú és a kártya által biztosított megszakítások egyaránt gondoskodhatnak.

A ***Comedi*** a legtöbb adatgyűjtési feladatot kiválóan képes ellátni. Lényegében a ***Comedi*** szolgáltatásait csak a futtatására használt hardver képességei korlátozzák. Az olcsóbb kártyák általában kisebb mintavételi gyakorisággal dolgoznak. Ha gyorsabb adatgyűjtést akarunk végezni, akkor drágább kártyát kell választanunk. Ezek DMA-kezelésre is képesek, az adatgyűjtést pedig saját processzoruk vezérli. A ***Comedi*** ilyenkor csak a pufferelt adatok továbbítását irányítja.

A gyors beolvasás ugyanakkor nem jelent egyben gyors feldolgozást is. A közönséges ***Linux*** rendszermag viselkedése időzítési szempontból kiszámíthatatlan, ezért az adatgyűjtés mellett valós idejű feldolgozást is végezni gyakorlatilag lehetetlen. Ehhez a folyamatok ütemezését szigorú szabályok alapján kellene végezni. Természetesen erre is van megoldás. A ***Linux Real-Time Application Interface*** (***RTAI***, valósidejű alkalmazásfelület) és az ***RTLinux*** csupán két példa a számos lehetőség közül, amelyekkel feljavítható a rendszermag időzítésvezérlése. Mindkét csomaghoz létezik ***Comedi*** felület. A valós idejű felületek mögötti alapötlet roppant egyszerű. A rendszermagot nem monolitikus folyamatként kell futtatni, hanem egy kis méretű és hatékony ütemező gyermekfolyamataként. Így a rendszermag nem tudja blokkolni a megszakításokat, illetve szükség esetén el lehet venni tőle az erőforrásokat. Ezután bármely a rendszer felett valós idejű vezérlési lehetőséget igénylő alkalmazás bejegyezheti magát az ütemezőnél, és olyan gyakran foszthatja meg a rendszermagot az erőforrásoktól, amilyen gyakran csak szeretné.

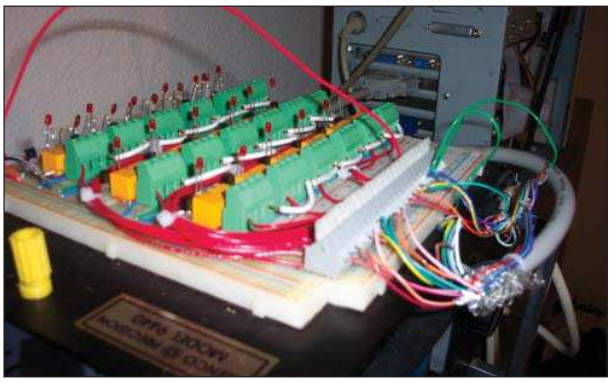
Vissza a laboratóriumi példához

Az ***AEI*** több tesztkamrát is fenntart dízelmotorok teszteléséhez. Minden cellában egy motor található, számos hőmérséklet- és nyomásmérő műszerrel felszerelve. Be szoktak építeni egy frekvenciamérőt is, amely a motor fordulatszámát figyeli. Végül a motort egy erőmérőhöz csatlakoztatják, amely valós vezetési helyzeteket szimulál úgy, hogy változó ellenállást fejt ki a működő motorral szemben. A létrejövő nyomatékot szintén mérjük.

A motoradatok beolvasási sebessége viszonylag kicsi, másodpercenként mindössze 20 mintát veszünk. Ha az adatok begyűjtése volna az egyetlen feladat, akkor az egész rendszer meglehetősen egyszerű lenne. Csakhogy az újabb adatkészletek beolvasásakor különféle beállításokat kell finomhangolni és szabályozni. A motor sebességét adott értéken kell tartani, ehhez változtatni kell a fojtószelep állását, illetve az erőmérő által adott terhelést. A cellának a hűtőfolyadék áramlását szabályozó szelepeit ugyancsak folyamatosan állítani kell, mert a hűtőfolyadék hőmérsékletét állandó értéken kell tartani. Biztonsági méréseket ugyancsak kell végezni, ezekkel ellenőrizhető, hogy katasztrófához vezető körülmények nem alakultak-e ki.

Ráadásul minden ellenőrzést és az új értékek beállítását még az előtt kell elvégezni, hogy egyéb teendői elvégzésére a rendszermag átvenné a vezérlést. Ha a ***Linux*** rendszermag maga gondoskodna erről az ütemezésről, nagy valószínűséggel minden megfelelően működne. Sajnos azonban nem lehet előre megmondani, hogy a teljes folyamat egyes lépései mikor kerülnek végrehajtásra. Az említett valós idejű kiterjesztések használatával ugyanakkor ez a probléma automatikusan megoldódik.

A valós idejű rendszermagoknak hátrányaik is vannak. Amikor a valós idejű ütemező meghatározott időszeletben futtat egy folyamatot, a ***Linux*** rendszermag futása lényegében megakad. A valós idejű folyamatnak ezért gyorsnak és hatékonynak kell lennie, és a lehető leghamarabb vissza kell adnia a vezérlést a rendszermagnak. Ha ezt elmulasztja, akkor a rendszer egyéb, nem valós idejű részei eléggé akadozva fognak futni. Ha pedig a valós idejű folyamatban valami hiba történik, és a rendszermag soha nem kapja vissza a vezérlést, akkor a teljes rendszer összeomolhat.



5. kép Csatoló áramkör a lámpák párhuzamos kapuról végzett vezérléséhez

Gyakorlati példa

A laboratóriumok mellett a ***Comedi*** akár otthon is használható. Egy egyszerűbb adatgyűjtő kártyához már 20-60 ezer forint körüli áron hozzájuthatunk, a gyártó nevétől, a kártya bonyolultságától és mintavételi gyakoriságtól függően. Egy ilyen eszközzel lakásunk különféle pontjain mérhetjük a hőmérsékletet, vagy mágneses érzékelővel figyelhetjük, hogy nem felejtettük-e nyitva a garázsajtót.

A személyi számítógépek érdekes jellegzetessége, hogy a párhuzamos kapuk vonalait külön is lehet vezérelni. A ***Comedi*** segítségével ezeket a vonalakat roppant egyszerűen tudjuk be- és kikapcsolni. Megfelelő relével párosítva pedig ezekkel gyakorlatilag bármit ki-be tudunk kapcsolgatni. Bár a párhuzamos kapuk az említett 0-5 voltos feszültségszintekkel dolgoznak, kellő áramerősséget nem képesek leadni. Párhuzamos kaput tehát nem szabad közvetlenül a vezérelni kívánt készülékhez csatlakoztatni, mindig valamilyen köztes áramkört kell beiktatni. Ilyen áramkörök készítéséhez számos weboldalon találhatunk leírásokat.

Én egy öreg 486-oson futtatom a ***Comedit***, és két párhuzamos kapu segítségével hozom létre az évi nyaralásunkkor szokásos fényjátékot. A házban a lámpák a szokásos módon vannak felszerelve, de mindegyikhez külön érpár vezet, amelyek a vezérlőszobában futnak össze (esetemben ez egy üresen álló hálószoba). Az érpárok egyedi áramköri laphoz csatlakoznak. Ezen találhatók azok a mechanikus relék, amelyek áramot adnak a lámpáknak, ha 5 voltos jelet kapnak a párhuzamos kapuról. Egy egyszerű C programot használok, amely ***Comedi*** függvényhívásokkal egyenként vezérli a párhuzamos kapu vonalait, vagyis be- és kikapcsolja a világítótesteket. Azt, hogy az egyes lámpákat mikor kell be- és kikapcsolni, egy egyszerű szöveges fájlban tudom megadni. A szomszédom nagy örömére...

Összefoglalás

Az adatgyűjtés lehetősége a laboratóriumokban rendkívül értékes. A ***Comedi*** által biztosított általános felülettel a ***Linux*** a legkülönfélébb DAQ-kártyákkal is kiválóan használható. Ahogy a ***Linux*** egyre népszerűbbé válik, úgy a ***Comedihez*** hasonló felületek fontossága is növekedni fog.

Az egyszerű DAQ-kártyák lassanként megfizethetővé válnak, a ***Linux*** alapú adatgyűjtés pedig valószínűleg egyre inkább elnyeri majd a területtel csak hobbiként vagy barkácsolási céllal foglalkozók tetszését. Ami korábban költséges hardver és programok vásárlását igényelte, az ma már – a legkülönfélébb területeken – bárki számára hozzáférhető.

2. kódrészlet Parancsok és ravaszok segítségével végzett, az előbbinél bonyolultabb pásztázást szemléltető kód

```
/* Cél: A Comedi beállítása kétcsatornás adat-
gyűjtésre és mindkét adatkészlet kétszeri beol-
vasása. Az adatgyűjtést egy digitális vonalon
érkező ravaszjel hatására indítjuk el.
  */
  comedi_cmd c, *cmd=&c;
  unsigned int chanlist[2];
  /* A CR_PACK egy különleges Comedi makró,
     amely a csatorna, a tartomány és
     a földhivatkozás beállítására szolgál.
  */
  chanlist[0] = CR_PACK(0,0,0);
  chanlist[1] = CR_PACK(1,0,0);
  /* Melyik aleszközt kell használnunk? */
  /* A legtöbb kártyán a nullás aleszköz egy
     analóg bemenet. */
  cmd->subdev        = 0;
  cmd->chanlist      = chanlist;
  cmd->chanlist_len = n_chan;
  /*A parancs elindítása a külső digitális vo-
    nalon érkező jelzés hatására. A start_arg ál-
    tal megadott digitális csatornát használjuk.
  */
  cmd->start_src = TRIG_EXT;
  cmd->start_arg = 3;
  /* A ravaszjel után azonnal megkezdjük a pász
     tázást. */
  cmd->scan_begin_src = TRIG_FOLLOW;
  cmd->scan_begin_arg = 0;
  /* A pásztázás után azonnal elvégezzük az
     átalakítást. */
  cmd->convert_src = TRIG_NOW;
  /* A scan_end_arg csatorna lekérdezése után
     befejezzük a pásztázást.
  */
  cmd->scan_end_src =      TRIG_COUNT;
  cmd->scan_end_arg =      2;
  /* stop_arg számú pásztázás után a parancsot
     leállítjuk. */
  cmd->stop_src =          TRIG_COUNT;
  cmd->stop_arg =          2;
  /* A parancs indítása */
  comedi_cmd(device, cmd);
```

Linux Journal 2004. augusztus, 124. szám

Caleb Tennis 1996 óta használ Linuxot. Ő volt a KDevelop Project egyik vezetője, jelenleg elsősorban a KDE Gentoo alatti változatának karbantartásával foglalkozik. Egy dízelmotor-tesztlabor mérnöki munkáit felügyeli, s eközben részidőben egy helyi főiskolán linuxos témákban oktat.