

Diplomatervezési Feladat

Mikulás Ádám

villamosmérnök hallgató részére

Adatgyűjtő rendszer (DAQ) fejlesztése nagyenergiás fizikai detektorrendszerekhez

A nagyenergiás fizikai - illetve gyakran orvosi és ipari alkalmazásoknál használt – mérőberendezések alapvető feladata az adatgyűjtési funkciók megvalósítása, mely magában foglalja mind az aleggységektől érkező adatok gyors beolvasását, az összetartozó mérési eredmények csoportosítását (event building) és esetleges előfeldolgozását és azok adatrögzítő rendszer felé történő továbbítását. A hallgató feladata egy olyan kis fogyasztású, rugalmasan kezelhető adatgyűjtő berendezés kifejlesztése, mely képes ellátni a nagyenergiás fizikai detektorok alapvető adatgyűjtési feladatait, beleértve a detektor, illetve a PC felé történő kommunikációt. Fontos szempont, hogy a megvalósítás során költséghatékony megoldást válasszon.

Feladatok:

1. Ismertesse a detektorrendszerrel, illetve a személyi számítógéppel történő kommunikáció megvalósítási lehetőségeit.
2. Válasszon a feladathoz illeszkedő mikroszámítógépes architektúrát, becsülje meg a szükséges erőforrás igényeket.
3. A korábbi pontok alapján állítsa fel az eszköz rendszertervét.
4. Dolgozza ki az egyes részegységek részleteit, beleértve a kapcsolási rajzokat valamint a nyomtatott huzalozású lemezeket. Állítsa össze az adatgyűjtő berendezést.
5. Készítse el az adatgyűjtést végző szoftverkomponenseket, valósítsa meg mind a detektor mind a PC felé történő kommunikációt.
6. Végezzen méréseket a konkrét detektorrendszer segítségével.

Tanszéki konzulens: Fehér Marcell

Külső konzulens: Varga Dezső (MTA KFKI RMKI)

Budapest, 2012. március 9.

Dr. Vajk István
egyetemi tanár
tanszékvezető



Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Mikulás Ádám

**ADATGYŰJTŐ RENDSZER (DAQ)
FEJLESZTÉSE NAGYENERGIÁS
FIZIKAI
DETEKTORRENDSZEREKHEZ**

KONZULENSEK

Varga Dezső Fehér Marcell

BUDAPEST, 2012

Tartalomjegyzék

Összefoglaló	7
Abstract.....	8
1 Bevezetés	9
2 Nagyenergiás fizikai mérések	11
2.1 A nagyenergiás fizikai mérések célja	11
2.2 A nagyenergiás fizikai mérési összeállítás	12
2.3 Trigger és adatgyűjtő rendszerek	13
2.3.1 A tervezés kulcslépései	14
2.3.2 Klasszikus adatkiolvasó rendszerek.....	15
2.3.3 Lokális pufferek	16
2.3.4 Több szintű trigger.....	16
2.3.5 Több szintű trigger lokális pufferekkel.....	17
3 A REGARD csoport.....	18
3.1 A kutatócsoport detektorai	18
3.1.1 CCC kamra - közeli katódú detektor	19
3.1.2 Szintillációs detektor	21
3.1.3 „Small Gap Chamber” detektor	21
3.2 A kutatócsoport detektorrendszerei - projektek.....	21
3.2.1 Müon-tomográf detektorrendszer	21
3.2.2 HPTD trigger detektorrendszer.....	22
3.3 A kutató csoport általános mérési elrendezése	24
3.4 A kutatócsoport adatgyűjtő rendszerei.....	25
3.4.1 CAMAC.....	26
3.4.2 LPT	26
3.4.3 A müon-tomográf specializált DAQ rendszere	27
4 Feladatom részletezése	30
4.1 Motiváció	30
4.2 Megvalósítási lehetőségek	31
4.2.1 Saját fejlesztés.....	31
4.2.2 Raspberry PI	32
4.3 Mérlegelés és döntés	34

5 DAQ rendszer megvalósítása Raspberry Pi-vel.....	36
5.1 Az RPi kezdeti lépések	36
5.1.1 Beszerzés	36
5.1.2 Az operációs rendszer kiválasztása és konfigurálása	36
5.1.3 A PC-RPi kommunikációhoz felhasznált programok.....	38
5.1.4 RPi GPIO lábainak bemutatása.....	40
5.2 Kommunikáció a Front End elektronikával.....	43
5.2.1 A CLOCK jel megvalósítási lehetőségei	43
5.2.2 Adat kiolvasás megvalósítása	48
5.3 Kommunikáció a trigger elektronikával	49
5.3.1 LAM jel olvasása:	49
5.3.2 BUSY jel kiadás.....	50
5.4 Kommunikációhoz szükséges feszültség átalakítás.....	50
5.4.1 Breadboardon megvalósítva 3.3 V – 5 V – 3.3 V feszültség átalakító	50
5.4.2 NYÁK-on megvalósítva a 3.3 V – 5 V – 3.3 V feszültség átalakító	53
5.5 Webszerver	55
6 Mérések.....	58
6.1 Mérések szimulálása	58
6.1.1 A kommunikációhoz szükséges 4 alapjel (LAM, BUSY, CLOCK, DATA) tesztelése	58
6.1.2 Saját előállítású trigger jelek által vezérelt mérési elrendezés tesztelése	60
6.1.3 Különböző sebességű trigger jelek számlálása	64
6.1.4 Az RPi-val való vezérlés megvalósítása UART-on keresztül	68
6.2 Konkrét mérések	70
6.2.1 A REGARD által fejlesztett, ADC mérést végző front-enddel való kommunikáció	70
6.2.2 Szcintillátor mérése.....	72
6.2.3 SGC mérése:	74
6.3 A mért adatok értékelése.....	76
6.3.1 SGC 1 millió esemény	76
6.3.2 Szcintillátor	78
6.3.3 Időkülönbségek egyes mérések között	79
6.4 A megvalósult mérési összeállítás értékelése	79
6.5 Továbbfejlesztési lehetőségek	80

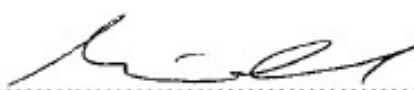
7 Összefoglalás.....	81
8 Köszönetnyilvánítás	83
Irodalomjegyzék.....	84
Ábrajegyzék.....	85
Függelék.....	87

HALLGATÓI NYILATKOZAT

Alulírott **Mikulás Ádám**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2012. 12. 05.


.....
Mikulás Ádám

Összefoglaló

A nagyenergiás fizikai - illetve gyakran orvosi és ipari alkalmazásoknál használt – mérőberendezések alapvető feladata az adatgyűjtési funkciók megvalósítása, mely magában foglalja mind az aleggységektől érkező adatok gyors beolvasását, az összetartozó mérési eredmények csoportosítását (event building), esetleges előfeldolgozását és azok adatrögzítő rendszer felé történő továbbítását. A jelenleg használt adatgyűjtő rendszerek (*Data Acquisition System* - DAQ) specifikusak, illetve a rendszert alkotó egységek bonyolult felépítésűek.

A feladatom egy általános, rugalmasan kezelhető adatgyűjtő rendszer megalkotása volt, amely bármely detektornál, mérési elrendezésnél használható, valamint képes ellátni a nagyenergiás fizikai detektorok alapvető adatgyűjtési feladatait, beleértve a detektor illetve a PC felé történő kommunikációt. Több követelménynek kellett megfelelni: alacsony fogyasztás, kis méret, költséghatékonyság. Az adatgyűjtés elkezdését egy trigger eseménynek kellett kiváltania, amelyet a körülményekhez mérten nagyon pontosan kellett beállítani. Továbbá lényeges szempont volt, hogy a PC felé továbbított adatok megfelelő formában jelenjenek meg a monitoron, mivel ezek a mérési eredmények elemzésre kerülnek a későbbiek során.

A kitűzött feladat megvalósítási lehetőségeit megvizsgálva és a konzulensemmel egyeztetve döntöttünk a Raspberry Pi féle megvalósítás mellett. Ennek a működését először szimulált környezetben teszteltem, majd a KFKI-ban lévő 2 különböző detektoron végeztem méréseket valós környezetben. A végső célt sikerült elérni, azaz egy olyan adatgyűjtő rendszert kifejleszteni, amely gyors, hatékony, olcsó és alacsony fogyasztású.

Abstract

The basic function of the measurement instruments of the High Energy Physics - as well as of the medical and industrial applications - is the realization of a Data Acquisition System (DAQ), meaning the scan of the incoming data from the subunits, event building, preprocessing and data transmission to the data record system. The actual DAQ systems are specific, the measurements and layouts are very complex and inflexible.

My challenge is to develop a general DAQ system which can be used at each detector, can communicate with them and which is able to fulfill the basic requirements of the High Energy Physics Detectors. There are some requirements: low consumption, small size, low cost. It is important to mention that in the course of a measure a huge data set exist but we have to select the data which is relevant for us. The start of the measurement is controlled by a trigger event, which we have to set very correct according to the conditions. It is significant too, that the frame of the transmitted data to the personal computer should visualize in a correct form on the screen, because with these data we would like to work on.

After I analyzed the possibilities of my challenge, my consultant and I decided to realize the challenge with Raspberry Pi. First I tested the device in simulated conditions, afterthat I measured with my data acquisition system on 2 kind of detector system in real conditions. I reached the final goal, what was a quick, cheap and low consumption system.

1 Bevezetés

A diplomamunkám az *MTA Wigner Fizikai Kutatóközpont Részecske- és Magfizikai Intézet* [1] által kiírt téma. Az Intézetben számos kísérletet végeznek párhuzamosan a kutatók, a legtöbbet különböző detektorrendszerek segítségével, amelyeket saját maguk fejlesztettek ki.

A feladatom megoldást találni a különböző detektorok által végzett részecske kísérletek adatrögzítésére, a jelenleginél jóval egyszerűbb, gyorsabb és költséghatékonyabb módon. Az adatgyűjtő rendszer (*Data Acquisition* – továbbiakban: DAQ) célkitűzése, hogy bizonyos paraméterek állításával az *RMKI - ELTE Gaseous detector Research and Development* (továbbiakban: REGARD, [2]) laborban megtalálható legtöbb detektorrendszerrel kompatibilis legyen. A fejlesztés által két helyen is időt nyerünk. Egyrészt univerzalitásának köszönhetően megtakarítja a fejlesztők számára azt az időt, melyet a különféle detektorrendszerekhez tartozó célfeldolgozó egységek tervezése igényelne. Másrészt relatív gyorsasága miatt felgyorsítaná a mérési adatok feldolgozását. Az adatgyűjtő rendszernek szintén célja, hogy később egy esetleges komplexebb méréshez is felhasználható legyen. Így további fontos szempont a tervezés során a rugalmasság és a bővíthetőség.

A dolgozat 2. fejezete általánosságban ismerteti a nagyenergiás fizikai mérések célját, a mérési összeállítást, melynek a dolgozat témáját érintő elemeit, a trigger rendszert, a detektoron található elsődleges érzékelő (front-end) eszközöket és az adatgyűjtő rendszert részletesebben is tárgyalom annak lehetséges megvalósításain keresztül. A fejezet átfogó képet kíván adni a nagyenergiás fizikai detektorrendszereknél történő adatgyűjtés kihívásáról.

A 3. fejezetben a REGARD csoport tevékenységét mutatom be. Ismertetem a laborban megtalálható detektorok típusait, hogy tisztában legyen az olvasó a mérések fizikai háttérével, illetve, hogy milyen méréshez tartozó adatokat kell feldolgozni. Ezután a futó projekteken keresztül az egyes detektorrendszereket sorolom fel. Végül ismertetem az azokhoz megvalósított adatgyűjtő rendszereket, melyek egyes hiányzó tulajdonságai adták feladatom motivációját.

A 4. fejezet a diplomamunkám lehetséges megoldási lehetőségeit vázolja és érvel a legmegfelelőbb megvalósítás mellett.

Az 5. fejezetben a választott út részletezése, elemzése következik. A *Raspberry Pi* (továbbiakban: RPi) mikroszámítógép tulajdonságainak ismeretében kiderül, hogy ez az eszköz kiválóan alkalmas mérésvezérlő és adatgyűjtő funkciók ellátására. Bemutatom az RPi ebből a szempontból fontos tulajdonságait, illetve konkrétan DAQ és a detektorrendszer közötti kommunikációhoz szükséges vezérlőjeleket, valamint a feszültség- és időzítési kompatibilitással kapcsolatos problémákat.

A 6. fejezetben a megvalósított DAQ rendszerhez vezető teszteket, szimulációkat mutatom be, majd két konkrét detektorrendszerrel való mérést, illetve kivitelezését ismertetem. Gyakorlati szempontokat is figyelembe vevő kritikai elemzésnek vetem alá a méréseket, összehasonlítva a népszerűnek számító, de már igen kevésbé korszerű *Computer Automated Measurement And Control* (továbbiakban: CAMAC) rendszerű mérésvezérlővel. Az eredmények kitekintést adnak a további fejlesztési lehetőségek irányába.

2 Nagyenergiás fizikai mérések

A diplomamunkám során olyan adatgyűjtő rendszer megvalósítása volt a feladatom, melyet nagyenergiás fizikai kísérletek esetén alkalmaznak. Ebben a fejezetben ismertetem egy általános nagyenergiás fizikai mérés célját, illetve a mérési összeállítást. A mérési elrendezés elemzése során a hangsúlyt az adatgyűjtő rendszerek és az azokhoz szorosan kapcsolódó trigger rendszer lehetséges megvalósításaira fektettem. Bemutatom a feladat mértékétől függő trigger és adatgyűjtő rendszerek sematikus működését, valamint kitérek azok korlátaira.

2.1 A nagyenergiás fizikai mérések célja

A részecskefizikai mérések célja az atommag belsejének, illetve az anyag jelenleg ismert legelemibb összetevőinek és ezek kölcsönhatásainak feltérképezése. Szerkezetéről az elmúlt század folyamán tisztázódott, hogy az atomok atommagból és elektronokból épülnek fel, az atommagok protonokat és neutronokat (közös néven nukleonokat) tartalmaznak, a nukleonok pedig kvarkokból állnak (1. ábra).

Az anyagi részecskék három családja (fermionok)

	I	II	III	
tömeg →	2.4 MeV	1.27 GeV	171.2 GeV	0
töltés →	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	0
spin →	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1
név →	u up	c charm	t top	γ foton
	Kvarkok			
	4.8 MeV	104 MeV	4.2 GeV	0
	$-\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	0
	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1
	d down	s strange	b bottom	g gluon
	< 2.2 eV	< 0.17 MeV	< 15.5 MeV	91.2 GeV
	0	0	0	0
	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1
	ν_e elektron-neutrínó	ν_μ műon-neutrínó	ν_τ tau-neutrínó	Z Z-bozon
	Leptonok			
	0.511 MeV	105.7 MeV	1.777 GeV	80.4 GeV
	-1	-1	-1	± 1
	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1
	e elektron	μ műon	τ tau	W[±] W bozon
				Bozonok (kölcsönhatások)

1. ábra Standard modell

A Standard Modell szerint [3] a kvarkok és elektronok tovább nem osztható úgynevezett elemi részecskék. Azonban számos megoldatlan kérdéssel kell még szembenéznünk, melyek megválaszolása újabb és újabb kihívások elé állítja a kutatókat.

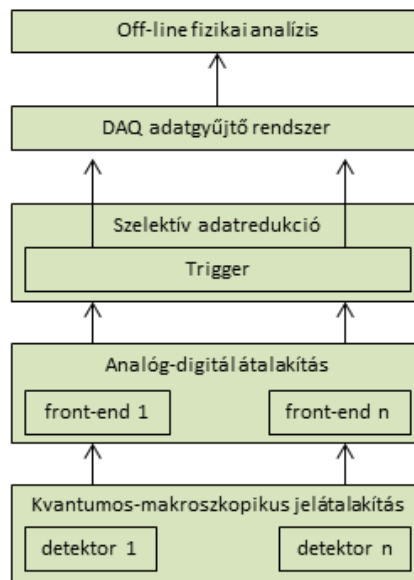
Az egyik nagy kérdés néhány hónapja oldódott meg, mely szerint a részecskék nyugalmi tömegét egy több évtizede hipotetikusán feltételezett, de most felfedezett részecske, a Higgs részecske adja [4]. Nem teljesen tisztázott továbbá a kvarkok közötti kölcsönhatás természete, illetve a fentieken kívüli további részecskék létezése.

2.2 A nagyenergiás fizikai mérési összeállítás

A részecskefizikai kísérletek során ismert részecskékkel ütközési kölcsönhatásokat hoznak létre és a szórt, keletkezett vagy visszaverődő alkotóelemeket részecske-detektoroknak nevezett berendezésekben próbálják összeszedni [5]. A teljes kísérleti folyamat sematikusán a 2. ábra mutatja.

Az első nagy feladat ezeknek a gyenge, tipikusan egy elemi töltéssel rendelkező részecskék által keltett jeleknek makroszkopikus jellé történő átalakítása. Ezt a feladatot maga a detektor látja el valamilyen fizikai folyamatra támaszkodva (például ionizáció vagy lyuk-elektron párok keltése félvezető anyagban). Ezek az előállított analóg jelek azonban jelentősen zajérzékenyek. Az elsődleges erősítésért, illetve nem ritkán az analóg-digitális átalakításért és az azt követő átmeneti adatrögzítésért felelős egység a front-end elektronika. Front-end eszköznek tulajdonképpen azt a részt nevezik a detektor elektronikus rendszerében, ami közvetlenül, fizikailag a detektoron van elhelyezve. A front-end tehát opcionálisan tartalmazhatja a digitalizálást végző egységet is, de van, hogy csak erősítést és jelformálást végez. Az egyes ütközések során óriási adatmennyiség keletkezik, elsősorban a részecske pályájának meghatározására szolgáló, nagyon sok elemi egységből álló detektorból. Képtelenség az összes adat elraktározása, ezért csak a kísérlet számára valamilyen, fizikai motiváció alapján jelentős eseményeket rögzítik adathordozóra, az összes többi törlésre kerül. A teljes eseményrendszerből a számunkra hasznos szűrését végző egység a trigger rendszer, melyet hierarchikusan, több szintben is megvalósítanak (első, második, illetve magasabb szintű trigger), attól függően mekkora átmeneti pufferrel rendelkezik a front-end, illetve milyen hatékonysággal tudják kiszűrni a lényegtelen eseményeket. Ha a megkülönböztetett esemény bekövetkezik a trigger jel hatására elkezdődik a konkrét mérés és az adatgyűjtés, melyet a DAQ egység vezérel. Az időzítés pontosabbá tételére és időnyerés miatt a trigger elektronikából megy egy vezérlőjel a front-endek felé (egyszerre nagyszámú, gyakran különböző típusú front-end is csatlakozhat ugyanazon detektorhoz és ennek hatására minden egyes front-end elektronikához tartozó átmeneti pufferbe

betöltődnek az adott front-endhez tartozó detektor részen lévő analóg jelek. A mérés indulásakor a trigger rendszer a DAQ rendszernek küld egy *Look At Me* (továbbiakban: LAM) jelet, melynek hatására a DAQ megkezdí a mérési adatok begyűjtését. Az összegyűjtött nyers adatokat később off-line módon analizálják a kutatók.



2. ábra A nagyenergiás fizikai mérési folyamat sémája [5].

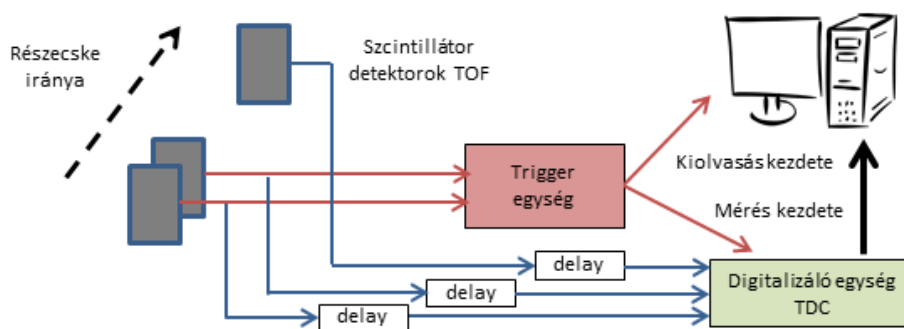
2.3 Trigger és adatgyűjtő rendszerek

A trigger és DAQ rendszer a részecskék kölcsönhatásai közül kiválasztják a fizikai feldolgozásra megfelelőt, illetve gondoskodnak a megfelelő adatok detektorból történő kiolvasásáról (átmeneti pufferből), az adatok megfelelő formátumba szervezéséről, valamint azok adathordozóra való mentéséről. Néhány esetben azonban a kalibrálhatóság érdekében szükség lehet a detektor és a front-end között történő kétirányú kommunikáció megvalósítására. A trigger rendszernek számos egymásnak ellentmondó követelménynek kell megfelelnie, úgymint nagy hatékonyság, erős háttérzaj elnyomás és gyors feldolgozás. A trigger rendszer egy előre definiált trigger menü alapján dönt egy esemény sorsáról. A trigger elektronika döntése alapján eltárolt adatokért felelős DAQ rendszernek meg kell akadályoznia az adatvesztést és adathibát, valamint minimalizálnia kell az adatok mentésére szükséges holtidőt, mely során értékes események detektálását veszíthetjük el.

2.3.1 A tervezés kulcslépései

Az ütközés után nem kívánt hátráltató tényezők miatt időbe telik, míg ki tudjuk választani azt az eseményt, ami érdekel bennünket és ennek az adatait tudjuk is rögzíteni. Késések adódnak, mert szükség van némi időre, amíg a trigger elektronika meghozza a döntést, a jeleket fizikailag továbbítani kell az elektronikához, olykor egy hosszú kábelon át (hiszen a trigger rendszer gyakran távoli eszközökből nyeri az információt). Ez és még számos más forrás útján befolyásolt holtidő az, ami az egyik legjelentősebb kérdés a trigger és adatgyűjtő rendszerek tervezése során, hiszen alatta a mérést felfüggesztik, azaz új eseményt nem tudunk detektálni.

A következőkben egy rövid és egyszerű mérési összeállításon keresztül mutatom be (3. ábra) a trigger és DAQ rendszer lehetséges változatait. Képzeljük el, hogy repülési időt mérünk három szcintillátor (lásd bővebben 3. fejezet) számlálóval, melyeket *Time-to-Digital Converterrel* (továbbiakban: TDC) olvasunk ki. Kettő közülük egymáshoz közel helyezkedik el, míg a harmadik elegendően távol ahhoz, hogy megállapítható legyen a részecske sebessége. Minden egyes detektort fotoelektronsokszorozó csővel figyelünk, melyek elektromos jelet juttatnak egy diszkriminátor áramkör felé. Az áramkör élesen felfutó digitális impulzust bocsát ki abban az esetben, ha egy töltött részecske haladt át a detektoron. A jel felfutó éle a részecske megjelenését követő határozott idő múlva jelenik meg. Trigger jelet akkor kapunk, ha a közeli detektorok egyszerre jeleznek (logikai ÉS trigger elektronika). A jelek egybeesésének detektálásával szűrjük a random zajjelenségeket. A háromcsatornás TDC segítségével a részecske sebességét úgy határozhatjuk meg, hogy mérjük az egyes detektorokon áthaladó időpillanatát a trigger jelhez relatívan. A TDC-be eljuttatott jeleket késleltetni kell, a trigger jel saját késése miatt. A trigger jelet a DAQ felé is szükséges továbbítani, hogy értesítsük a kiolvasás kezdetéről. A működés szempontjából fontos az a logika, mely megakadályozza egy további trigger jel megjelenését addig, amíg az TDC-ből az adatkiolvasás folyik.



3. ábra Mintapéllda: mérési elrendezés vázlata időmérés esetén [11].

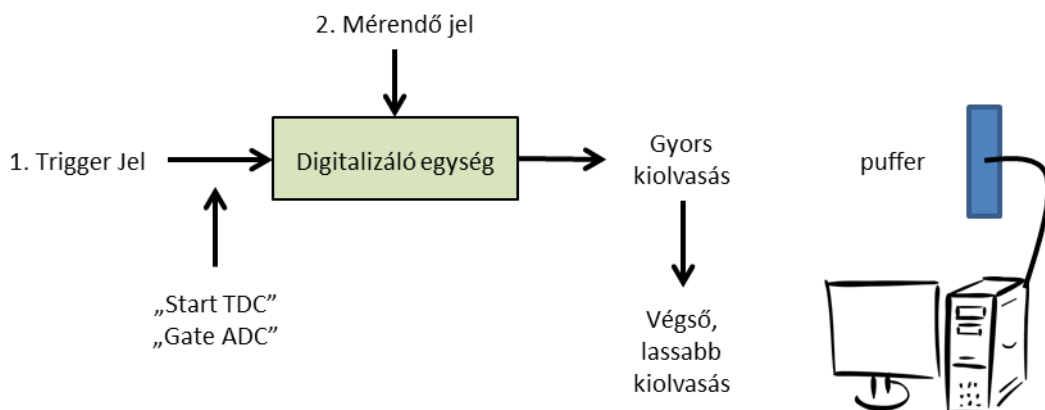
2.3.2 Klasszikus adatkiolvasó rendszerek

A nagyenergiás fizikai mérések kivitelezésére kialakultak szabványos digitalizáló és adatkiolvasó rendszerek, melyek közül egyesek mostanra jöllehet idejét múltak, de továbbra is használatban vannak, leginkább kisebb, fejlesztési célzatú laborokban. Az egyik ilyen a *Nuclear Instrumentation Module* (továbbiakban: NIM) [6], mely időzítési és nyers feldolgozási feladatokat végezhet, vagy a *Computer Automated Measurement And Control*, (továbbiakban: CAMAC) [7] szabványú elektronikai modulok, ahol lehetőség van rugalmas számítógépes kommunikációra is. Ezek már nagyon régi, de még mindig drága technológiák. Azok a kisebb laborok, amelyek rendelkeznek ilyen eszközzel, még mindig használják a CAMAC rendszert egyszerűsége miatt, de a legtöbb esetben saját maguknak fejlesztenek ki DAQ rendszereket. A nagyobb kollaborációk által üzemeltetett kísérleteknél szinte kivétel nélkül saját fejlesztésű, hatalmas költségű elektronikai rendszerekről van szó, optimalizálva a szenzorok hatalmas számára.

Több szempont korlátozza a fent bemutatott, egyetlen trigger jelet figyelembe vevő elrendezést. A trigger döntésnek rendkívül hamar meg kell érkeznie ahhoz, hogy a TDC időben digitalizálni tudja a jeleket. Ugyanez a helyzet az ADC átalakítók esetén is, melyek a jel amplitúdóját digitalizálják. További hátrány jelent, ha számítógép felé a TDC-ből való olvasás sebessége lassú. Ez magas triggerelési sebesség mellett nagy holtidőhöz vezethet. Ez a limit még inkább korlátoz nagyobb rendszerek esetén, ahol a számos csatornát egymás után kell kiolvasnunk minden egyes eseményre.

2.3.3 Lokális pufferek

Az előzőekben leírt megvalósítást úgy fejleszthetjük tovább, hogy egy lokális puffer tárolót integrálunk a rendszerbe (4. ábra), melybe gyorsan beíródnak az adatok a trigger jel érkezése után. A gyors kiolvasás csökkenti a holtidőt, melyet a triggerelési sebesség és a lokális kiolvasás szorzataként kapunk. Ez a megvalósítás azoknál a nagyobb rendszereknél jelentős, ahol az adattovábbítás párhuzamosítható a lokális pufferekkel (pl. minden elektronikus rekeszhez tartozik egy puffer). Ilyen módon a lokális kiolvasás történhet gyorsan nagy rendszerek esetén is. Ráadásul a lokális pufferbe írás sebessége sokkal gyorsabb lehet, mint a DAQ számítógépbe. A puffer segítségével csökkentettük az effektív kiolvasási időt, azonban a gyors trigger iránti igényt nem teljesítettük. A jeleket ugyanis késleltetnünk kell, amíg a trigger döntés megszületik a trigger logikán. Komplex döntéskritériumok ezzel a módszerrel nem vizsgálhatók.

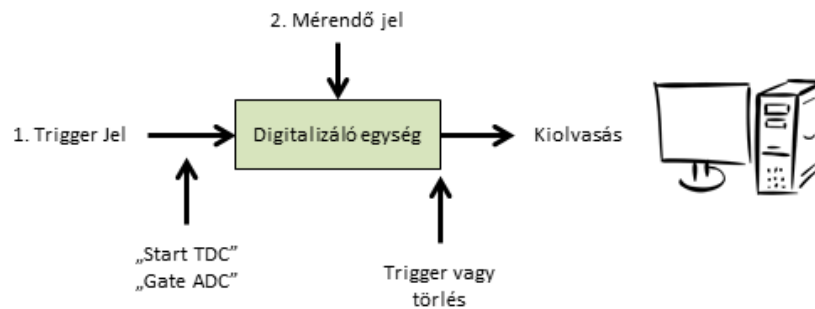


4. ábra Lokális pufferek [11].

2.3.4 Többszintű trigger

Nem mindig lehetséges egyidejűleg teljesíteni a triggerrel szemben támasztott fizikai követelményeket és mellette elérni egy rendkívül rövid lappangási időt (mialatt megszületik a trigger döntés és eléri a trigger jel a megfelelő elektronikához). A többszintű trigger erre kínál egy megoldást (5. ábra). Az első szintű trigger lappangási ideje rövid, hatékonysága nagy. A háttérzaj kiszűrését a magasabb szintű trigger végzi, mely lehet lassabb. Az előbbi a TDC felé küldi a start jelet, míg az utóbbi a kiolvasást indítja a regiszterekből akár komplexebb kalkulációkkal kísérve az adatszűrés. Azokban az esetekben, ha az első szintű trigger jelet nem erősíti meg magasabb szintű

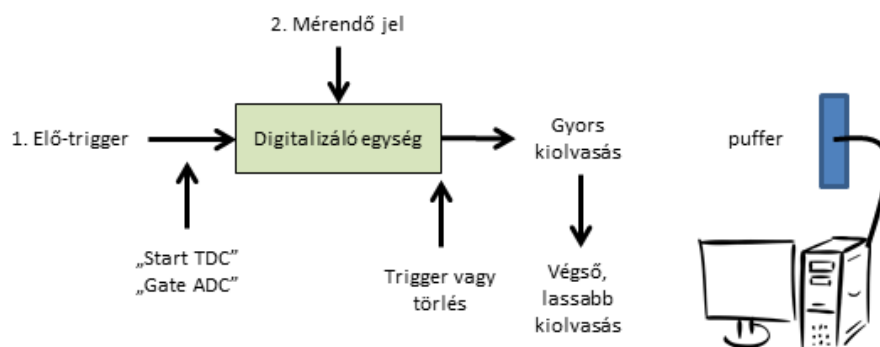
(master) trigger, egy gyors törlési művelet következik be, hogy újra aktiválhassuk a TDC-t (vagy ADC-t).



5. ábra Többszintű trigger [11].

2.3.5 Többszintű trigger lokális puffereléssel

A trigger szintek száma tovább növelhető. Az alacsonyabb szintű triggererek kevésbé szűrik a jeleket, de gyors működésűek. Épp ellenkezőleg a magasabb trigger szintek adatszűrése erősebb, sebessége azonban lassabb (6. ábra). A holtidőt azzal csökkenthetjük, hogy az adatokat átmeneti tárolókba mentjük, az előző fejezetben tárgyalt módszereknek megfelelően.



6. ábra Többszintű trigger lokális puffereléssel [11].

3 A REGARD csoport

A dolgozatomat a REGARD csoport keretében, a Wigner Fizikai Intézetben kialakított laboratórium méréseivel kapcsolatban írom. A csoport 2009 szeptemberében jött létre két magyar intézet, az Eötvös Loránd Tudományegyetem és az MTA Részecske és Magfizikai Kutatóintézete (mai nevén Wigner Fizikai Intézet) együttműködéséből. A megalakulás elősegítője a CERN (*Európai Nukleáris Kutatási Szervezet*) kezdeményezésére 2009-ben, a detektortechnika fejlesztése érdekében létrehozott RD51 kollaboráció, melynek mindkét intézet alapító tagja volt. A REGARD (RMKI ELTE Gaseous detector Research and Development) rövidítésből következtethetően fő feladatuk a gáztöltésű részecskefizikai detektorok kutatása és fejlesztése.

Jelen fejezetben szeretném ismertetni, hogy mivel is foglalkozik ez a részleg, milyen fejlesztési, kutatási területeken tevékenykednek az itt dolgozó emberek. Először bemutatom a kutatás-fejlesztésük eszközét képező detektorokat, hogy átfogó képet kapjunk arról, milyen méréshez tartozó adatokat kell feldolgozni. Ezután áttekintem a tárgyalt detektorokból kifejlesztett és összeállított detektorrendszereket, azaz a konkrét projekteket. Végül ismertetem feladatom motivációját a csoport jelenleg használatban lévő adatgyűjtési rendszereinek elemzésén keresztül.

3.1 A kutatócsoport detektorai

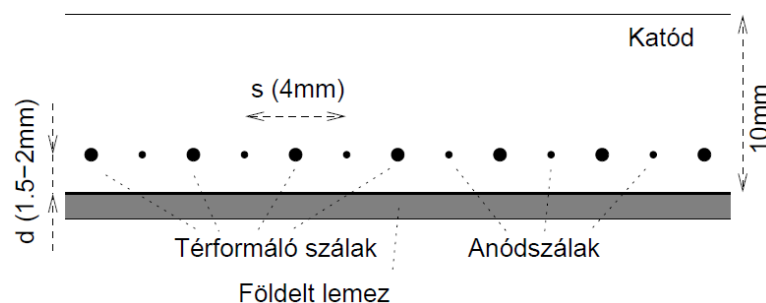
A REGARD kutatócsoport fő kutatási területe a gáztöltésű detektorok fejlesztése, valamint ezen a területen tudományos kutatások, kísérletsorozatok végzése. A részecskék detektálásának elvi alapja mindig a részecskék (sugárzás) és a detektor anyagának kölcsönhatása (legtöbbször elektromágneses kölcsönhatás, például ionizáció, cserenkov sugárzás). A REGARD csoport detektorait kozmikus részecskékkel (mion ~ 2 GeV), béta-sugárzással (elektron ~ 2 MeV), illetve a CERN-ben előállított és helyben használható célnyalábbal (pion nyaláb ~ 5 GeV) is vizsgálják.

Az alábbiakban néhány olyan detektortípusról adok áttekintést, amelyek jól példázzák az adatkiolvasási lehetőségeket és kihívásokat.

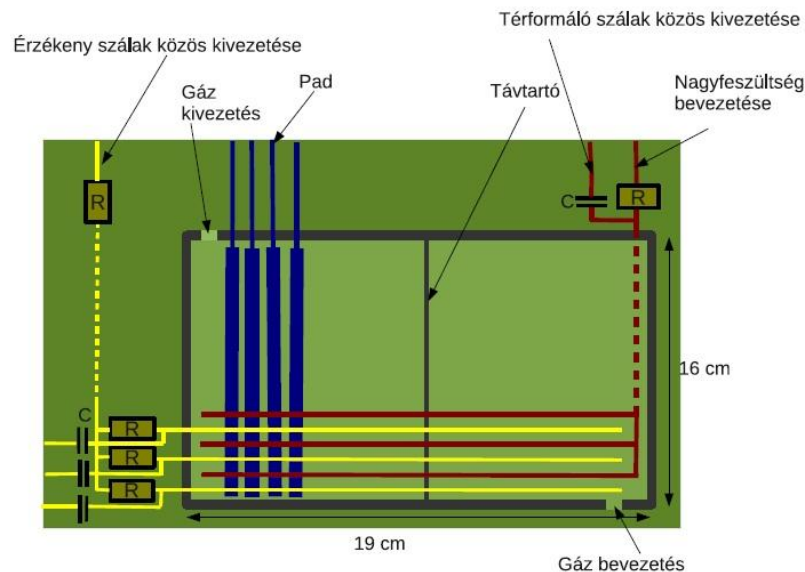
3.1.1 CCC kamra - közeli katódú detektor

A mérendő fizikai jel (részecske) indirekt detektálására elterjedt módszer az ionizáció jelensége, mely során atomi pályáikról kiszakított elektronok jönnek létre. Ezeket az elektronokat gyűjtik össze és alakítják át makroszkopikusan mérhető jellé. A megfelelő jelszint előállítása érdekében számos fejlesztés történt az elmúlt évtizedben. Az egyik legfrissebb a sokszálas proporcionális kamra *Multiwire Proportional Chamber*, (továbbiakban: MWPC) [8], melynek kifejlesztéséért és a nyolcvanas évekre elért tökéletesítéséért G. Charpak francia kutatót Nobel-díjjal jutalmazták. Az MWPC-k népes családjának a REGARD csoport által továbbfejlesztett változata a közeli katódú kamra *Close Cathode Chamber*, (továbbiakban: CCC) [8]. A detektor kialakításának köszönhetően egyetlen kamrával megállapítható a részecske áthaladásának kétdimenziós pozíciója. A csoport detektorrendszereinek jelentős része ilyen típusú kamrákból áll, ezért felépítését és a jelkiolvasás folyamatát részletesebben elemzem.

A CCC kamra egy légmentesen záró, nagy tisztaságú gázt tartalmazó tartály („kamra”), melyben a két katódlemez között (melyek közül az alsó földelve van, a felső pedig nagyfeszültségre ~ -500 V van kapcsolva) helyezkednek el egy síkban a szálak. A szimmetrikus proporcionális kamrákhoz képest különbséget jelent, hogy a szálsíkot a földelt katódhoz közelebb helyezték (7. ábra). Másrészt a pozitív potenciálú anódszálak közé térformáló, negatív feszültségű szálakat vezettek (8. ábra). A földpotenciálon lévő lemezt szegmentálták és ezzel bevezettek egy, a szálsík futási irányára merőleges „parketta” vagy pad-rendszert. Az újításokkal sikerült mechanikai változásokra ellenállóbbá tenni az eszközt. A bevitt anyag mennyisége és így a kamra költsége csökken, kivitelezése egyszerűsödik.



7. ábra CCC elrendezés oldalnézet [8]



8. ábra CCC elrendezés felülnézet [9]

A jelkiolvasás folyamata a következőképpen történik. A kamrán áthaladó részecske ionizálja az érzékeny térfogatot kitöltő gázt (Ar-CO₂ keverék 9:1 arányban áramoltatva). A leszakadt elektronok a detektorban kialakított elektromos tér hatására gyorsulva az anódszálak felé tartanak. Az anódszálak környezetében kialakításukból fakadóan nagy az elektromos térerősség. Az elektronok akkora energiára tesznek szert, hogy maguk is másodlagos ionizációt keltenek a gáztérfogatban. Ezzel a jelenséggel lavina alakul ki, tipikusan 10⁶ elektront tartalmazó jellé erősödik és detektálhatóvá válik. Az ellentöltéseknek (pozitív töltésű ionok alkotta ionfelhőnek) köszönhetően, mely kapacitívan csatolódik minden, a lavina környezetében levő elektródához, megjelenik jel, azaz detektálhatóvá válik a térformáló szálakról is és a szegmentált katódról is. Mivel az anódszálon megjelenő jel nagyobb, mint a többi elektródára szétoszló csatolt jel, ez utóbbit (minden anódszálat egymással összekötve) triggerelésre alkalmazzák (detektálják haladt-e át részecske a detektoron), míg ezek után a térformáló szálakról és az ezekre merőleges szegmentált katódról leolvasva megállapíthatják, hogy hol haladt át a kamrán (a kétirányú mérés adja az áthaladási pont két koordinátáját). Vegyük észre, hogy a több kamrából összeállított detektorrendszer saját magát triggereli.

A térformáló szálakról és a pad-ekről érzékelt jelet diszkriminátor segítségével digitalizálhatjuk (ennek az „egy bites” digitalizálásnak költséghatékonysági oka van csak) és tárolhatjuk az átmeneti pufferekben. A jel kiolvasására analóg-digitál

konvertert is használhatunk, ezen az úton pontosabb helyazonosítást érhetünk el [11], viszont adott esetben jelentősen növekednének a költségek.

3.1.2 Szcintillációs detektor

A szcintillátorok olyan anyagok, amelyekben a molekulák egy áthaladó töltött részecske vagy γ -foton hatására gerjesztett állapotba kerülnek, az alapállapotba való visszatérés során pedig gerjesztett fotonok, azaz fluoreszcenciafény bocsátódik ki. A szcintillátor anyagának természetesen átlátszónak kell lennie a benne keletkező fényre, hogy a detektálás megoldható legyen. Léteznek szerves és szervetlen anyagból készült szcintillátorok. Előbbiek, melyek nagyenergiás fizikában széles körben kapnak alkalmazást, általában plasztik anyagok, de készültek már kristály vagy folyadék halmazállapotú szerves szcintillátorok is. A plasztik (polikarbonát) szcintillátorok azért előnyösek, mert nem túl drágák, könnyen gyárthatók, mechanikailag jól formálhatók, a belőlük kapott fényjel pedig gyors (tipikusan 10 nanoszekundum vagy annál rövidebb az impulzus). A szervetlen szcintillátorok legjelentősebb képviselői NaI(Tl) kristályból készülnek, ezek nagyobb sűrűségük miatt hasznosak elsősorban gamma foton detektálásra.

3.1.3 „Small Gap Chamber” detektor

A *Small Gap Chamber* (továbbiakban: SGC) egy teljesen új fejlesztése a kutatócsoportnak. Dolgozatom szempontjából egy nagyon egyszerű proporcionális kamra, mivel 1 darab anód szál van benne, többi, aránylag összetett része nem játszott szerepet. Az eszköz egyik mérési lehetősége, hogy egy UV LED-ből érkező foton kiüt foto-effektussal egy elektront a kamra katód részéből (falából) és az egyetlen elektron által keltett lavinát, illetve jelerősítés után a feszültséget tudjuk mérni a kimeneten. Részletes mérési leírás a 6.2.3 fejezetben.

3.2 A kutatócsoport detektorrendszerei - projektek

Több projekt fut párhuzamosan a csoporton belül, saját fejlesztésre, általános céllal, illetve CERN-es kísérletre, konkrét céllal.

3.2.1 Müon-tomográf detektorrendszer

A müon-tomográf kísérlet egy saját célra alkotott detektorrendszer, melyet a kutatócsoport saját fejlesztésű CCC (lásd 3.1.1 fejezet) kamráiból építettek fel (több

detektor helyezkedik el egymás alatt). A müon-tomográfia során a detektorrendszer felett elhelyezkedő anyagon keresztülhaladó müonok intenzitáscsökkenését mérik.

A kísérletben detektálandó részecske, a müon, kozmikus eredetű (Tipikus energiája ~ 2 GeV, de ez időnként több száz GeV-et is elérhet). A Földünket bombázó nagyenergiás részecskék és a légkör felső rétegének kölcsönhatása által keltett részecskezapor eredményeként jut el a földfelszínig, illetve a földkéregbe is. A földfelszínre másodpercenként 5-10 részecske jut négyzetméterenként.

A mérési elv első történelmi vonatkozása a fizikai Nobel-díjas L.W. Alvarez nevéhez fűződik, aki a gízai Kefren-piramis rejtett kamráinak létét kutatta. Az eljárást napjainkban számos helyen alkalmazzák különböző célokra. Ezek közül megemlítek néhányat. Japánban az Asama vulkán tevékenységét vizsgálják emulziós ködkamra segítségével. A Los Alamosi Kutatóintézetben az eljárást a határon átkelő járművek radioaktív forrás alkalmazása nélküli átvizsgálásra adaptálják. A Mexikói Állami Egyetemen a Teotihuacanban található Nap-piramis belső szerkezetét kutatják sokszálas proporcionális kamrákból álló tomográfot alkalmazva [10].

A REGARD csoport tomográfjának célja földalatti üregek vagy kiterjedtebb közet-inhomogenitások keresése és vizsgálata. A fentebb említett alkalmazásokkal összehasonlítva a rendszer azoknál nagyobb precizitású, kisebb, mobilisebb és költséghatékonyabb. Ezek mellett kevesebb infrastruktúrát, energiafelhasználást és emberi felügyeletet eredményez.

A detektorrendszer folyamatosan fejlesztés alatt áll. Jelenleg a harmadik prototípusa készül. A példányok a CCC kamrák kivitelezésének minőségében (pad-rendszer nélkül vagy pad-rendszerrel), illetve az egymás alatt elhelyezkedő kamrák számában különböznek. Trigger esemény, azaz a mérés rögzítése akkor történik, ha több kamra együttesen jelzi részecske áthaladását. Ennek vizsgálatára kombinációs logikát fejlesztettek ki.

3.2.2 HPTD trigger detektorrendszer

A *High pT Trigger Detector* (továbbiakban: HPTD) detektorrendszer kifejlesztésére a CERN bízta meg a csapatot. A *Large Hadron Collider* (továbbiakban: LHC) nagy hadronütköztető egyik kísérlete, A *Large Ion Collider Experiment* (továbbiakban: ALICE) ugyanis bővíteni szeretné jelenlegi, hatalmas és komplex detektorrendszerének egyik aldetektorát. Ez a bővítés az aldetektorával azonos fizikai

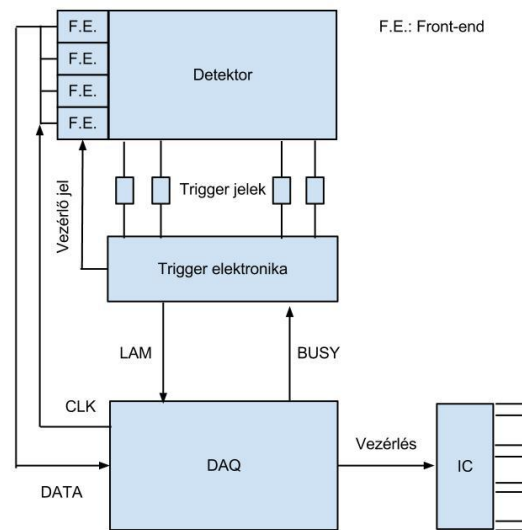
mennyiséget mérne (Cserenkov sugárzáson alapuló sebességmérés), azonban nagyobb energiaszinteken. A jelenlegi aldetektor a *High Momentum Particle Identification* (továbbiakban: HMPID) nevet viseli, mely nagy impulzusú részecske azonosítást jelent. A tervezett aldetektor a *Very High Momentum Particle Identification* (továbbiakban: VHMPID) a nagyon nagy impulzusú részecske azonosító.

Az LHC két üzemmódja közül (proton-proton vagy nehézion-nehézion ütközés) az ALICE-kísérlet az utóbbira lett optimalizálva. Az ólom-ólom ütközések során hatalmas adatmennyiség keletkezik. A másodpercenként bekövetkező 8000 esemény egyenként megközelítően 90 MB adatot eredményez. Másodpercenkénti tárolásra azonban csak 1.5 GB áll rendelkezésre. Azaz az eseményeknek mindössze a 0.25 %-át rögzíthetjük. Ennek az óriási mennyiségű adatnak a szűrését az 1. fejezetben bemutatott módon több szintű trigger és adatgyűjtő rendszerrel látják el. Ebbe a háromszintű trigger rendszerbe ékelődne (a 0-dik és 1. szint közé) a HPTD trigger detektor, mely a VHMPID mérő detektor működését szabályozná (0-dik szintű trigger érkezésekor mér, illetve képes 1-es szintű triggerrel jel kiadására) [2].

A HPTD detektorrendszert a kutatócsoport a müon-tomográfhoz teljesen hasonló módon szeretné megvalósítani. Azaz szintén CCC kamrákból álló rendszert épít fel. A prototípusok folyamatos fejlesztés alatt állnak. Várhatóan 2x4 rétegből fog felépülni. A HPTD detektornak a müon-tomográfhoz hasonlóan annyi a feladata, hogy megállapítsa, mely pontokon haladt át részecske. Kiolvasó, front-end elektronikája ezért szintén egyszerűbb.

3.3 A kutató csoport általános mérési elrendezése

A REGARD csoport által használt általános mérési elrendezést 9. ábra mutatja.



9. ábra Általános mérési elrendezés

A DAQ rendszer ebben a megvalósításban négy jelet kezel a mérési adatok kiolvasásához, és egyet a vezérléshez. Ezek közül kettő közvetlenül a detektorhoz csatlakozó front-end elektronikákkal valósít meg kétirányú kommunikációt. A másik két jel a trigger elektronikával hoz létre duális kapcsolatot. A csatornák nevét, feladatát és a DAQ rendszerhez vonatkoztatott kommunikáció irányát az alábbi felsorolás ismerteti, melynek sorrendje egyben utal a jelek időbeliségére is:

- LAM jel fogadása az adatgyűjtés indítása céljából (trigger elektronika),
- BUSY jel kiadása a trigger események szüneteltetésére (trigger elektronika),
- CLOCK jel kiadása az adatok kiolvasása céljából (front-end),
- DATA olvasás (front-end),
- vezérlés: SPI, UART, I²C

A mérési folyamat a 2. fejezetben ismertetett általános változathoz nagyon hasonlít. Ha egy részecske a detektorrendszeren áthalad különböző, időben gyorsan lejárászó fizikai folyamatok után elektromos jelet generál. Ez, az áthaladást követően gyorsan detektálható jel szolgál trigger jelként, melyről a trigger logika kiértékelése után a trigger egység dönti el, hogy mérjük-e az eseményt vagy sem. A trigger esemény meghatározása egyrészt a mérés céljától (hatékonyság, detektálandó jel minősége),

illetve a mérési összeállítástól függ (detektorok száma). Amennyiben a trigger egység döntése pozitív, küld egy vezérlőjelet a front-endek felé, hogy a detektor analóg kivezetésén lévő jelet olvassa be és alakítsa át digitálissá. Ehhez a front-end elektronikáknak körülbelül 100 ns – 1 μ s közötti idő áll rendelkezésükre, mivel ennyi ideig marad a detektor kivezetésein (a front-end belső késleltetése miatt) egy adott eseményhez tartozó jel. Ezzel párhuzamosan a trigger elektronika küld egy LAM jelet a DAQ rendszer felé, hogy kezdje meg az adatok kiolvasást a front-endek tárolóiból. Természetesen szinkronizálni szükséges a folyamatokat, hogy a pufferekből történő adatkiolvasás csak a mérés elvégzése után kezdődhessen. Ha a DAQ rendszer észlelte a LAM jelet akkor arra válaszként egy BUSY jelet küld vissza a trigger egységnek, hogy blokkolja a további trigger beérkezését addig, amíg ki nem olvassa az összes adatot a front-endekből. A BUSY jel kiadása után az adatgyűjtő egység küldi a CLOCK jelet a front-endek felé és olvassa be az adatokat. Ha beolvasta az összes adatot, akkor a CLOCK-ot és a BUSY-t nullába állítja és várja a következő LAM beérkezését.

A mérés közben igény szerint tudunk vezérelni eszközöket, amennyiben szükség van rá SPI, I²C illetve UART-on keresztül (pl. léptető motor mechanikai pozicionáláshoz vagy nagyfeszültség értékének beállítása).

3.4 A kutatócsoport adatgyűjtő rendszerei

Az alábbiakban ismertetek három, a csoport által alkalmazott adatgyűjtő rendszert. A rendszerek működését a müon-tomográf kísérlet alapján mutatom be, ahol mindegyik típus tesztelésre került. A leírásuk után felsorolom a rendszerek korlátait és ismertetem a feladatom motivációját.

elektronikák minden kapott órajel után küldenek egy bitet a PC-nek, amely közben blokkolja a triggeret. Miután véget ért a bitek átadása, a PC feloldja a trigger blokkolását és várja az új triggeret.

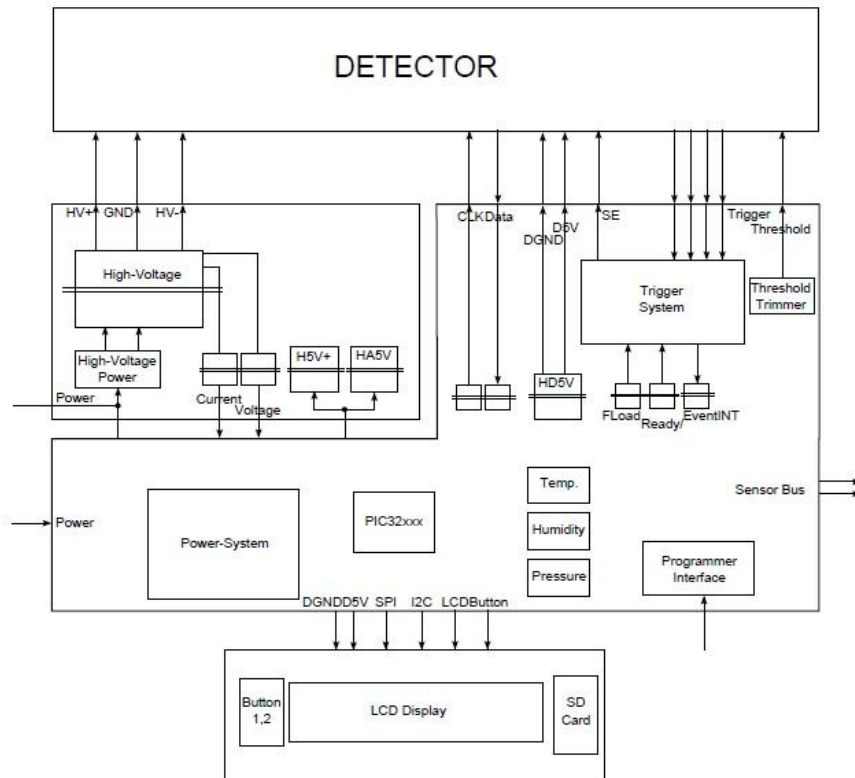
Az LPT alkalmazása során szükség van egy laptopra, amely közvetlen vezérlőként működik és ennek a printer portján keresztül kapcsolódik a detektorrendszerhez. 0.5 MHz frekvenciájú órajellel tud olvasni, tehát 500.000 bitet olvas be egy másodperc alatt, de a különböző időzítések, illetve bit beolvasáshoz szükséges idők miatt körülbelül 40 db CLOCK jel kell egy 16 bites adat beolvasásához, ami egy esemény. Így ez a fajta DAQ rendszer ~12.000 esemény / másodpercre képes.

3.4.3 A müon-tomográf specializált DAQ rendszere

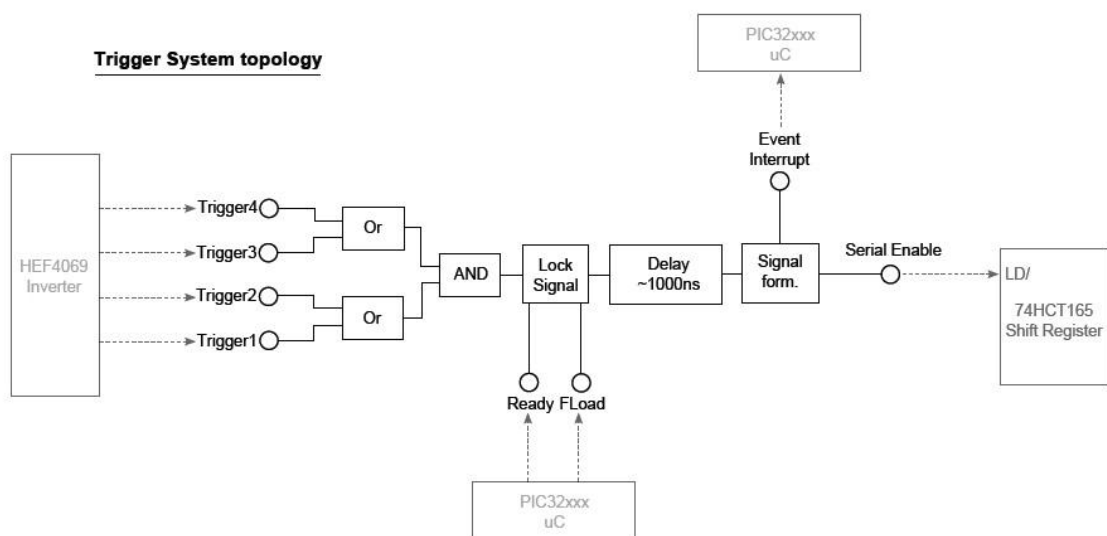
A REGARD csoport által működtetett, tényleges környezetfizikai (barlangkutató) mérésekben részt vett müon-tomográf adatkiolvasó rendszerét Melegh Hunor villamosmérnök tervezte és építette [12]. Az általa összeállított rendszermodell 11. ábra mutatja, részei:

- tápegység: a detektort ellátó feszültséget állítja elő
- adatgyűjtést megvalósító kártya
- kezelőfelület
- detektorrendszer

Az érintésvédelmi szempontoknak megfelelően galvanikus leválasztást kellett biztosítani. Ennél a mérésnél kialakított trigger rendszer feladata, hogy ha a felső- és alsó párból legalább egy detektor jelez (összesen 2, 12. ábra), akkor egy kisebb késleltetést követően, töltse fel a front-enden lévő shift regisztereket, amelyeket majd később kiolvasásra kerülnek. A triggerimpulzust követően csak akkor töltődhetnek fel újra adatokkal a regiszterek, ha már az összes adatot kiolvastuk. A rendszer elvi vázlata a következő rajzon látható:



11. ábra REGARD fejlesztésű DAQ [12]



12. ábra REGRAD fejlesztés trigger jel kiértékelése [12]

A mikrokontroller kiválasztása során csupán annyi megkötés volt, hogy a Microchip PIC32xxx fejlesztő rendszerrel kompatibilis legyen. A választás a PIC32 család egy egyszerűbb tagjára esett PIC32MX340F512H. Sem USB-vel, sem Ethernetnel nem rendelkezik, mivel ezekre nem is volt szükség, az adott rendszernél. 32 kbyte RAM-mal rendelkezik, amely az adatok ideiglenes tárolásához elegendő

nagyságú, végleges tárolásuk egy SD kártyára írással lett megvalósítva. A mikrokontroller rendelkezik a fontosabb perifériákkal, funkciókkal:

- SPI,
- I2C,
- UART,
- AD átalakító,
- megszakítás rendszer,
- timer.

Az adatgyűjtő kártyához tartoznak még különböző szenzorok, amelyekkel a hőmérsékletet, a páratartalmat illetve a nyomást tudjuk mérni. A tervek alapján legyártásra kerültek a nyomtatott huzalozású lemezek, amelyekre körülbelül 300 darab alkatrészt ültettek be kézzel.

Jól elhatárolható processzek állnak rendelkezésre, ezért egy állapotgépen alapuló szoftvermodell fut a mikrokontrolleren, amelyet C nyelven írtak meg.

A rendszer műszaki jellemzői:

- maximum 380 mA fogyasztás 12 V mellett, 5 W alatti a fogyasztása,
- adatok kiolvasása 0,8 milliszekundum idő alatt,
- SD kártyára történő írás ~2 másodperc,
- megvalósított kártyák méretei (13. ábra):

Név	Méret	Darabszám
uDatv1.0.2	141.63x89.53mm	1db
HighVoltage	116.84x71.12mm	1db
HMI.uDatv1.0.1	119.40x56.92mm	1db
SD.card	29.88x39.03mm	1db
Sensor	27.50x21.53mm	3db

13. ábra Megvalósított kártyák méretei [12]

4 Feladatom részletezése

4.1 Motiváció

A 3.3 fejezetben bemutatott mérési elrendezés DAQ részének megvalósítása a feladatom, úgy hogy az kompatibilis legyen a különböző kísérletekhez. Ezzel rendelkezésre állna egy olyan alap adatgyűjtő rendszer, melyet nem kell minden egyes alkalommal külön megtervezni, az adott kísérletre kibővíteni és szükség esetén könnyen lehessen specializálni. Továbbá biztosítsa a stabil, pontos működést és a begyűjtött adatokat később felhasználásra alkalmas formában tárolja el. A megjelenő bitmintákat a detektor digitális tárolóiból lehet sorosan kiolvasni. A kiolvasás általában valamilyen eseményvezérelt ütemezésre történik (kapott órajel fel- vagy lefutó éle).

Az általános adatgyűjtő rendszerrel szemben elvárt követelmények:

- a FE (front-end, itt valósul meg a detektorból jövő analóg jelek felerősítése, illetve digitalizálása) felől jövő digitális adatok, nagyon egyszerű és gyors összegyűjtése (ha lehet párhuzamosan 4-8 adatcsatornán, megfelelően választott órajellel 1-20 MHz).
- PC-nek küldhető adathalmazt kell tervezni (minden bejött adatot összerakni egybe - event building), hozzáfűzni a timestamp-et, egyéb adatokat, eseményszámot, stb
- elküldés a PC-nek
- egy trigger unitnak kell az olvasást indítania, vagyis hasznos, ha a trigger unit (vagy egy része) benne van a DAQ-ban, a trigger és a DAQ nem választható el élesen (együtt kell működniük)
- mindezt kis fogyasztással (néhány W), olcsón, kis méretben

A fentebb ismertetett kísérleteket a KFKI-ban állítják össze, majd amikor lehetőség adódik rá, akkor a CERN-ben is tesztelik azokat. A cél az, hogy egy könnyen összeállítható mérési elrendezés álljon a rendelkezésükre és feleslegesen ne mozgassanak hatalmas berendezéseket a kísérletek elvégzéséhez. Azért szükséges a mérésben használt eszközök kiszállítása a CERN-be, mert a bérletük nagyon drága.

A 3.4-ben bemutatott REGARD csoport által használt DAQ rendszerek között nincs olyan, amely univerzálisan mindegyik méréshez hatékonyan felhasználható lenne. A CAMAC rendszer hátránya, hogy nagyon lassú, nagyon drága, és maga a rendszer is nagyméretű. Emellett a fogyasztása is jelentős (~ 50 W) és szükséges 230 V-os feszültség a működéséhez. Előnye, hogy egyszerű és megbízható, így egy új DAQ rendszer működését könnyen tudjuk ellenőrizni vele. Az LPT hátránya, hogy egy laptop szükséges hozzá és annak teljes processzor idejét lefoglalja a mérés vezérlése. A sebessége ~ 12.000 db esemény / másodperc. A Hunor-féle megvalósítás tökéletesen alkalmas egy olyan detektorrendszerhez ahol másodpercenként csak néhány esemény következik be és az alacsony fogyasztás elsődleges szempont. Ez a DAQ rendszer azonban nem tudna lekezelni egy olyan mérést ahol másodpercenként több 10.000 db esemény következik be, továbbá a mikrokontroller programozása és működésének ellenőrzése is rendkívül körülményes és nehézkes. Ezért lenne szükség egy olyan rendszer megalkotására, amely ezek hátrányát kiküszöbölve, de előnyüket megtartva egy univerzális DAQ rendszert valósítana meg.

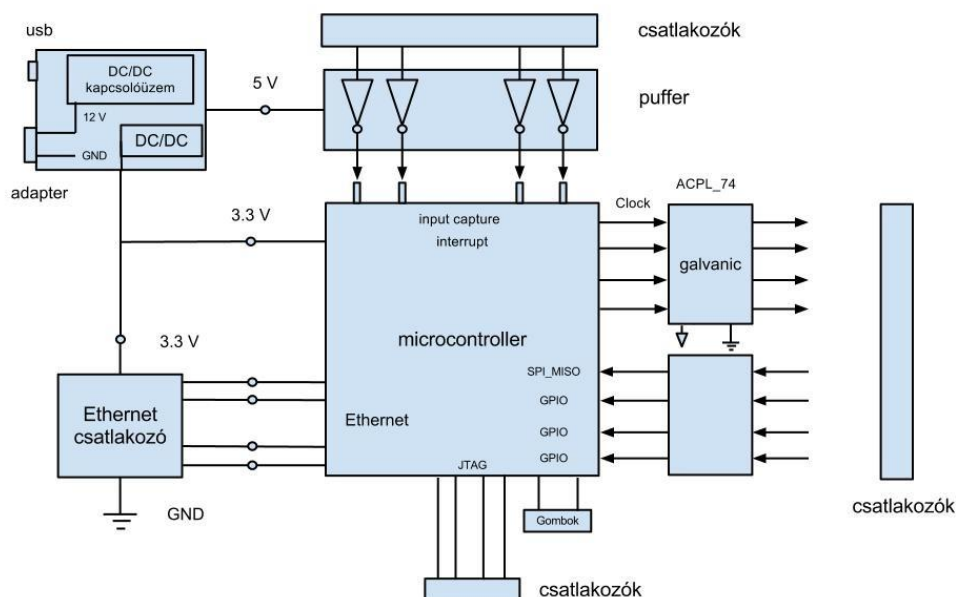
4.2 Megvalósítási lehetőségek

A rendszerrel kapcsolatos elvárások meghatározása valamint a kísérletekbe történő betekintés után konzulensemme, Varga Dezsővel - megvizsgálva az alternatív megoldásokat is - arra a következtetésre jutottunk, hogy 2 féle megvalósítás lehetséges. Az egyik a 3.4.4-ben bemutatott DAQ rendszer áttekintése, általánosítása úgy, hogy az megfeleljen a feladatkiírásban szereplő kritériumoknak. A másik - egy menetközben felmerülő lehetőség - egy újonnan piacra dobott készülék, a Raspberry Pi mini PC által megvalósított megoldás. Párhuzamosan mindkét lehetőséget szem előtt tartva döntöttünk végül közösen a végső megoldás mellett.

4.2.1 Saját fejlesztés

A 3.4.4-ben ismertetett DAQ rendszer, mint már korábban említettem egy müon-tomográfhoz készült. Az adatok mentése SD kártyára történik, amely szűk keresztmetszete a rendszernek, amennyiben nagy esemény számú kísérletet akarunk elvégezni. Ajánlott lenne Etherneten keresztül az adatok továbbítását megoldani, mivel ez egy nagyon elterjedt és gyors kommunikációs csatorna.

Ehhez a megvalósításhoz készült egy vázlat terv (14. ábra):



14. ábra Saját fejlesztési megvalósítás vázlati terve

Ennek a megvalósításnak a hátránya, hogy úgy kell megtervezni a rendszert, hogy később nem fogunk tudni rajta változtatni egy esetlegesen felmerülő változó körülmény miatt, illetve a legyártása is időigényes. Ennek a lehetőségnek a kidolgozási fázisáig jutottam (mikrokontrollerek kiválasztása, szükséges NYÁK lapok tervezésének megkezdése, stb.), de a következő pontokban leírt események miatt nem léptem ennél tovább.

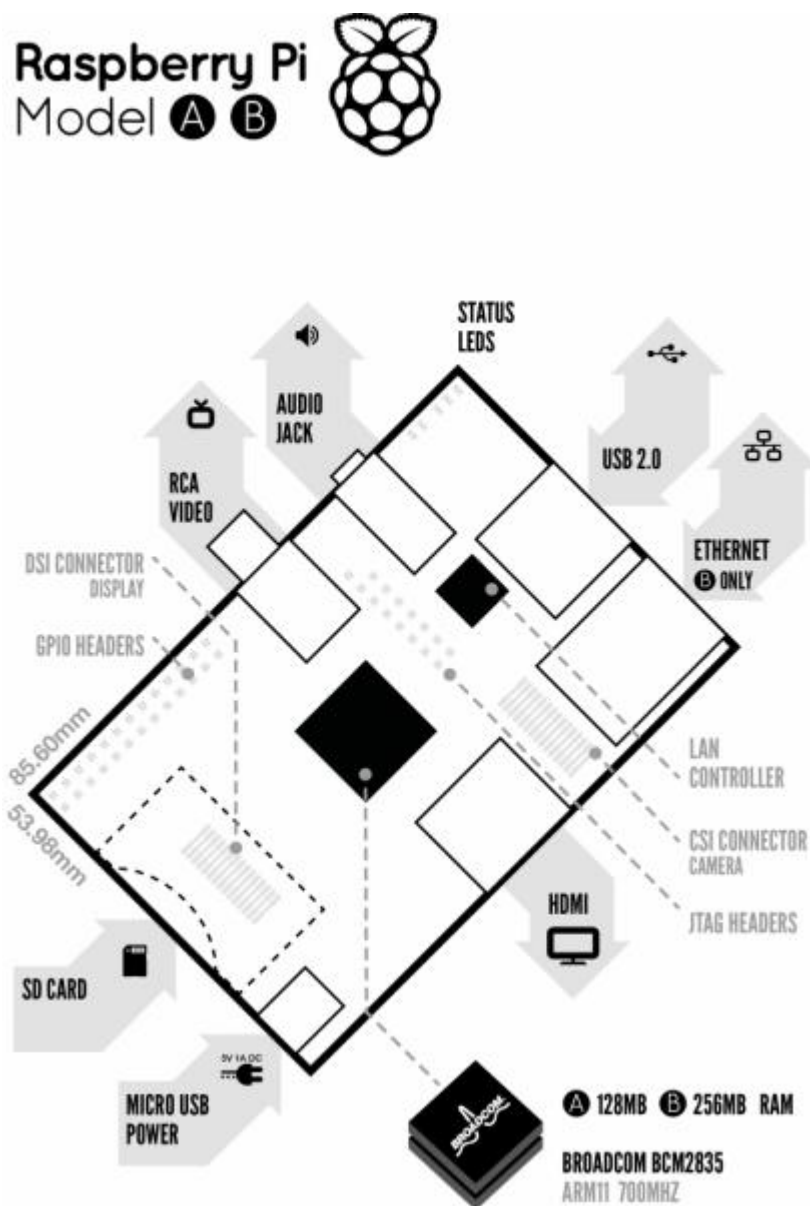
4.2.2 Raspberry PI

A másik út a *Raspberry PI* (továbbiakban: RPi, 15. ábra) alkalmazása. Ez az eszköz most került a piacra „olcsó és kicsi linux alapú PC” névvel. Dobozolatlan verzióban az alap („A”) változat 25\$, a fejlettebb („B”) változat 35\$ és mindössze bankkártya méretűek. Röviden az RPi-ről:

A CPU 700 MHz-en működik, ami önmagában kevés lenne, de a Broadcom VideoCore IV GPU-nak köszönhetően akár még 1080p-s (Full HD) videókkal is megbirkózik. A nagy különbség a két modell között a csatlakozókban és a memóriában rejlik (1. táblázat) [13].



15. ábra Raspberry Pi [13]



16. ábra RPi B modell [13]

	MODEL A	MODEL B
Ár:	25 USD (~5760 HUF)	35 USD (~8070 HUF)
SoC:	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM)	
CPU:	700 MHz ARM1176JZF-S mag (ARM11 család)	
GPU:	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 h.264/MPEG-4 AVC high-profile decode	
Memória (SDRAM):	128 MB	256 MB
USB 2.0 ports:	1	2
Video kimenetek:	Composite RCA	HDMI
Audio kimenetek:	3.5 mm jack	3.5 mm jack, HDMI
Adattárolás:	SD / MMC / SDIO kártyaolvasó	
Alaplapi hálókártya:	nincs	10/100 Ethernet (RJ45)
Fogyasztás:	500 mA (2.5 W)	700 mA (3.5 W)
Táplálás:	5 V MicroUSB	
Méretek	85.60 × 53.98 mm	
Támogatott operációs rendszerek	Debian GNU/Linux, Fedora, Arch Linux	

1. táblázat RPi modell A és B különbségek

Az RPi használatához mindössze USB-s klaviatúrára, egérre, megfelelő SD kártyára (minimum 4GB), operációs rendszerre, HDMI vagy RCA kábelre csatlakoztatható monitorra, micro USB tápellátásra (minimum 0,7 A, 5 V) és persze magára az RPi-ra van szükség.

A táblázat adataiból is látszik, hogy a B modell az optimális megoldás. Az alacsony fogyasztása mellett a mérete is kicsi, olcsó, Ethernet csatlakozóval és video kimenettel is rendelkezik, amely a rugalmasságát nagyban növeli. A számomra szükséges bejövő, illetve kimenő jelek feldolgozását a GPIO lábakon szoftveresen könnyen meg lehet valósítani.

Az eszközt nem tudtam kipróbálni, de az interneten található tesztek alátámasztották, hogy ezzel az eszközzel meg lehetne valósítani a kitűzött feladatot.

4.3 Mérlegelés és döntés

A saját fejlesztés (4.2.1) tanulmányozása után arra a következtetésre jutottam, hogy ezzel csak nagyon specifikus megvalósítást tudnék elérni, ezért ennél egy jóval általánosabb rendszerre lenne szükség. Kiindulásnak azonban megfelelő lehet. Azonban egy olyan megoldásra lenne szükség, mely rugalmasan tud reagálni egy adott mérés adott paraméter változására (mérések darab száma, mérete, órajel, vezérlés, stb.)

A konzulensemmel illetve a témavezetőkkel tartott megbeszélések során arra a megállapításra jutottunk, hogy a már meglévő 3.4.4 megvalósításhoz hasonló egyszerűbb mérésekhez újabb DAQ rendszert nem lenne célszerű kifejleszteni, érdekesebb lenne az RPi-s féle útba belekezdeni és abból kihozni a maximumot. Ez a megoldás egy teljesen új kaput nyithat a DAQ rendszerek világába. Ezért az RPi által vezérelt DAQ rendszer kifejlesztése mellett döntöttünk!

5 DAQ rendszer megvalósítása Raspberry Pi-vel

5.1 Az RPi kezdeti lépések

5.1.1 Beszerzés

Az RPi beszerzése sajnos nem volt egyszerű folyamat. A Raspberry Pi Foundation 10.000 db RPi-t gyártatott le első körben január vége felé. Nem számítottak akkora érdeklődésre, mint amekkora fogadta őket a piacon. Az alapító, Eben Upton, tervei szerint a célközönség a fiatalabb korosztály lett volna. Nekik szeretett volna egy olcsó célhardware-t nyújtani ahhoz, hogy könnyen és gyorsan megtanulhassanak programozni már az általános iskolákban is. Nehezítette a rendelést, hogy Ázsiában gyártották az RPi-kat, emiatt a szállítás és tesztelés tovább növelte a beszerzéshez szükséges időt. Az első 10.000 db értékesítése március elején kezdődött - pillanatok alatt elkelt mind - ezeknek a kiszállítása körülbelül június elején történt meg. Én júniusban rendeltem meg (először!) az RPi-t, rögtön 2 disztribútortól (RS Component, illetve Farnell Element14), 1-2 hónapos szállítási határidő mellett. Többszöri rendelési és telefonálási fázis után sikerült a Farnelltől október elején hozzájutnom a saját Raspberry Pi „B” modellemhez.

5.1.2 Az operációs rendszer kiválasztása és konfigurálása

Az RPi egy speciális hardware platform, amely nagyon hasonló egy okostelefon felépítéséhez. Emiatt a választható operációs rendszerek image fájlként állnak a felhasználó rendelkezésére, amelyet egy SD kártyára flashelve tudunk elérni az RPi-val. Minimum 2 GB-os SD kártya szükséges, de 4GB-os ajánlott. Windows 7-t használok, ezért le kellett tölteni a Win32DiskImager nevű programot az image fájl SD kártyára másolásához. A hivatalos honlapról [13] a diploma írás pillanatában az alábbi operációs rendszerek voltak elérhetőek:

- Raspbian „wheezy”: Ez az általánosan elterjedt operációs rendszer, amely a Debiannak egy RPi-ra optimalizált verziója. Ez a verzió tartalmaz Midorit, LXDE-t, fejlesztő eszközöket és példa kódokat a multimédiás funkciókhoz. (2012-10-28-as verzió)

- Soft-float Debian „wheezy”: A Raspbianhoz hasonló image, de a lassabb soft-float ABI-t használja. (2012-08-08-as verzió)
- Arch Linux ARM: Amint a neve is mutatja Arch Linuxon alapuló rendszerről van szó, amely teljes körű irányítást ad a felhasználó kezébe. Egyik előnye a gyors boot ~10 másodperc. (2012-09-18-as verzió)
- RISC OS. Castle Technology által készített, alapvetően ARM architektúrájú rendszerekre, grafikus felhasználói interfészen alapuló operációs rendszer.

Választásom az általános Raspbian 10-28-as verziójára esett, így ezt másoltam fel az említett Win32DiskImagerrel. A Raspbian image körülbelül 800 MB, kicsomagolva 2 GB.

Következő lépés a rendszer elindítása volt. Az SD kártyaolvasó az RPi alsó felén található. Szükség volt még egy USB-s klaviatúrára és egérre. Ethernet-en keresztül kapcsolódtam az internetre, amihez egy UTP kábelre, végül a monitor csatlakozáshoz egy HDMI kábel kellett. Micro USB-n keresztül, amely 5 V / 0,7 A-rel táplálja az RPi-t, adtam rá tápfeszültséget és elindult a rendszer. Egy On/Off kapcsolót és egy Reset gombot hiányolok az RPi-ről, hiszen kikapcsolni csak úgy lehet, ha kihúzom a micro USB-t. A bekapcsolás után röviddel bebootol a rendszer és az oldalon szereplő username / password: pi/raspberry párossal be tudunk lépni. Az első bootoláskor felugró ablakban (raspi-config, később a

```
# sudo raspi-config
```

paranccsal hívható elő, sudo = do as superuser) néhány alapvető konfigurációs beállítást tudunk állítani:

- *info*: általános információ a beállításokról
- *expand-rootfs*: ha ezt a beállítást engedélyezem, akkor az éppen használt SD kártya egész területét használja, nem csak az image fájlt. Ez akkor előnyös, ha nagyobb méretű software-eket akarunk telepíteni. Hátránya, hogy ha backup-ot akarunk csinálni az image-ről akkor több időbe telik.
- *overscan*: ennél a pontnál a megjelenítést tudjuk beállítani, azaz amikor pl.: tv-re kötjük az RPi-t akkor a tv általában levág a megjelenítendő képből. Ennek az engedélyezésével kicsinyíteni tudjuk a megjelenítendő képet.
- *configure-keyboard*: a billentyűzetünk típusát állíthatjuk be.

- *change_pass*: a jelszó megváltoztatása
- *change_local*: a betűk kódolásának beállítása
- *change_timezone*: belső óra beállítása a saját időzónánknak megfelelően
- *memory_split*: beállíthatjuk hogyan osztjuk meg a rendelkezésre álló RAM-ot a GPU és a CPU között. Választani lehet, hogy 32, 64 vagy 128 MB-ot adunk a GPU-nak.
- *ssh*: ssh (secure shell) server engedélyezése / tiltása. Az ssh kommunikációs protokoll segítségével be tudunk lépni más számítógépről az RPi-ra IP cím segítségével, pl.: putty terminál segítségével.
- *boot_behavior*: választhatunk a parancssoros vagy grafikus verziós boot-tal induljon a rendszer.
- *update*: a raspi-config-ot tudjuk vele upgradelni, amennyiben egy újabb verziót fejlesztenének ki

Kiválasztva a megfelelő beállításokat tovább tudtam lépni. Kipróbáltam a grafikus környezetet a

```
# startx
```

paranccsal. Egy teljesen szokványos Linux rendszer képpel találkozik a felhasználó. A Midori böngésző használatával lehet internetezni. A sebességét nem szabad összehasonlítani egy PC-jével, de kisebb várakozással minden szépen elindult és futott rajta.

5.1.3 A PC-RPi kommunikációhoz felhasznált programok

Az ssh egy nagyon praktikus és kényelmes megoldása annak, hogy ne kelljen külön monitort kötni az RPi-ra, hanem a saját PC-ről tudjak belépni. Ehhez be kell állítani a statikus IP címet, így a beállított címmel tudok mindig belépni. Ezt az interfészek között tudom megtenni:

```
# sudo nano etc/network/interfaces
```

A nano egy Unix rendszerekben használt szövegszerkesztő, aminek a célja, hogy egy egyszerű interfészen keresztül tudjunk szöveges fájlokat létrehozni, átírni, menteni, stb. Itt kell beállítani a következőket (/etc/network/interfaces):

```
auto lo
```

```

iface lo inet loopback
auto eth0
iface eth0 inet static
address xxx.xxx.xxx.xxx
gateway xxx.xxx.xxx.xxx
netmask 255.255.255.0

```

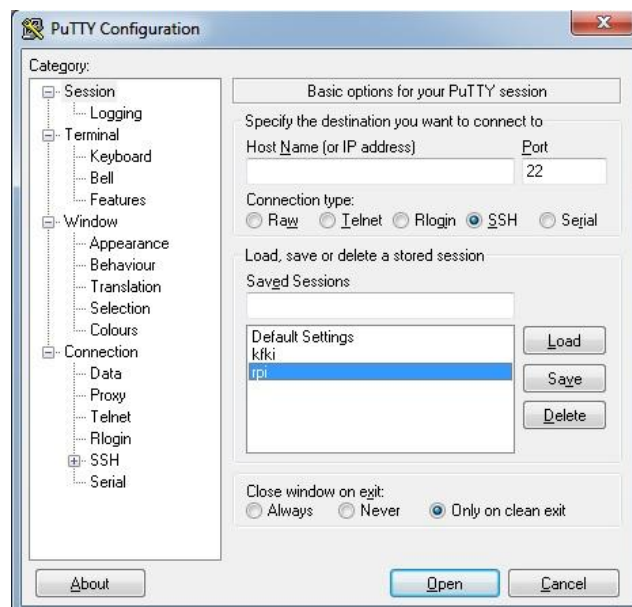
Ha végeztünk, akkor a Ctrl+O billentyű kombinációval tudjuk elmenteni a fájlt, majd a Ctrl+X kombinációval tudunk kilépni. A változtatások nem működnek egyből újra kell indítani a hálózatot:

```
# sudo /etc/init.d/networking restart
```

A beállítások érvényre jutását az

```
# ifconfig
```

paranccsal tudjuk ellenőrizni. Amennyiben mindent rendben találtunk, akkor elindíthatjuk a PC-n a **Putty** programot, amely egy ingyenes telnet és ssh kliens Windows és Unix rendszerre egyaránt. A 17. ábra a Putty program futás közben, értelemszerűen kell kitölteni a rublikákat, a statikus IP címet be kell írni, a 22-es portot kell használni és az ssh beállítást. Az Open gomb megnyomásával máris elénk tárul az RPi parancssoros módja, ahol a username/password megadása után folytathatjuk a munkát.



17. ábra Putty program

Rendelkezésemre állt egy Cisco Linksys Compact Wireless-G USB [21] adapter, így adott volt, hogy ezt is működésre bírjam, mivel így sokkal egyszerűbben lehet elérni az RPi-t, mint kábelén keresztül. A kényelem előtérbe helyezése a PC és az

RPi kommunikáció sebességének rovására megy. A Wi-Fi konfigurálásához szintén az /etc/network/interfaces fájlt kell megnyitnunk és kiegészíteni néhány sorral az Ethernet kapcsolat analógiája szerint:

```
# sudo nano etc/network/interfaces

iface wlan0 inet static
    wpa-ssid routernév
    wpa-psk password
    address xxx.xxx.xxx.xxx
    gateway xxx.xxx.xxx.xxx
    netmask 255.255.255.0
```

Elmentve a beállításokat és ellenőrizve az IP címet csatlakoztattam az USB portra a Wi-Fi adaptert, amelyet automatikusan felismert és rögtön csatlakozott a PC-hez **Connectify**-on keresztül, amely programmal egy saját Wi-Fi hotspotot tudok létrehozni router nélkül WPA2 titkosítással. A Putty-t megnyitva, ugyanúgy tudok csatlakozni most már az RPi-hez, csak vezeték nélkül.

Most, hogy minden összeállt az RPi-val való kommunikációhoz, tovább léphetünk a feladatom megoldása felé. A következő pontban ismertetem azokat a lépéseket, amiket a fejlesztés során szükséges volt megtenni ahhoz, hogy detektorból adatot tudjak kiolvasni. Előtte még telepítettem a fejlesztéshez szükséges könyvtárakat a

```
# sudo apt-get essential-build install
```

parancs segítségével, amely package különböző csomagokat tartalmaz, de ami számunkra a legfontosabb az a „gcc”, illetve a „make”.

5.1.4 RPi GPIO lábainak bemutatása

Mielőtt ismertetném a méréshez szükséges jelek generálását, néhány szót a GPIO lábokról. Ahogy azt a neve is mutatja General Purpose Input/Output, általános pinekről van szó, amiket a felhasználó tud felprogramozni futási időben annak megfelelően, hogy mire akarja használni a pineket. Minden egyes láb bemenet és kimenet módban is tud működni. A lábak értéke lehet magas (= 1), illetve alacsony (= 0). Az RPi-on 26 darab pin van (2 sorban, 13-13 darab, 18. ábra):

- 2db 3.3 V
- 2db 5 V
- 5db Ground

- 8 db GPIO
- 2db UART-hoz tartozó láb (Txd, Rxd), használható GPIO-ként
- 2db I²C-hez tartozó láb, használható GPIO-ként
- 5db SPI-hez tartozó láb, használható GPIO-ként



18. ábra RPi GPIO BCM lábkiosztás [14]

A GPIO lábak feszültség szintje 3.3 V kompatibilis, az 5 V-ot nem tolerálja, és nem rendelkezik túlfeszültség védelemmel, így nagyon figyelmesen kell bánni velük. Ha nem használjuk az alternatív funkcióit a lábaknak, akkor összesen 17 GPIO láb áll a rendelkezésünkre. A 3.3 V-os pinek maximum 50 mA, az 5 V-os pinek pedig 1 A nagyságú áramot adnak le. A detektorokkal alapvetően 5 V TTL jelekkel tudunk kommunikálni, azaz az alacsony szint 0 V - 0.8 V között, míg a magas szint 2.2 V – 5 V között. A köztes tartomány tiltott, ezért gyorsan kell váltani a két szint között.

A lábak számozására 3 féle módszer létezik, először is a legegyszerűbb, amikor az USB táp felfelé van és a pin sor bal felső lába az első, mellette a második és így tovább 26-ig. Létezik a BCM processzor kivezetett lábai által meghatározott számozás (18. ábra), illetve a wiringPi által használt számozás (19. ábra) A wiringPi [15] egy C nyelvű függvénykönyvtár, amellyel a processzor GPIO lábait tudjuk könnyen vezérelni. Eredetileg a GPIO lábak, adott memória címekre képződnek le és így lehet őket

közvetlenül elérni/vezérelni. A wiringPi-t használva nem kell alacsony szinten programoznunk az RPi-t, hanem lehetőség van már előre definiált függvényekkel programoznunk. A továbbiakban, ha külön nem mondom, hogy más GPIO számozás, akkor a wiringPi-s számozás szerintire értem, amely az alábbi ábrán látható (a revision 2-től a Do Not Connect (DNC) = Ground). Erre fokozottan figyelni kell a programozás közben, mivel nagy meglepetéseket tud okozni egy-egy elnézett láb száma.

wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin
–	–	3.3v	1 2	5v	–	–
8	R1:0/R2:2	SDA0	3 4	DNC	–	–
9	R1:1/R2:3	SCL0	5 6	0v	–	–
7	4	GPIO7	7 8	TxD	14	15
–	–	DNC	9 10	RxD	15	16
0	17	GPIO0	11 12	GPIO1	18	1
2	R1:21/R2:27	GPIO2	13 14	DNC	–	–
3	22	GPIO3	15 16	GPIO4	23	4
–	–	DNC	17 18	GPIO5	24	5
12	10	MOSI	19 20	DNC	–	–
13	9	MISO	21 22	GPIO6	25	6
14	11	SCLK	23 24	CE0	8	10
–	–	DNC	25 26	CE1	7	11
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin

19. ábra RPi GPIO wiringPi lábkiosztás [15]

17 darab GPIO láb áll rendelkezésünkre, ebből a korábban ismertetett jelek vezérlésére 3 fixen elmegy (BUSY, LAM, CLOCK) 14 darab marad az adatok beolvasásra és külön vezérlésre. Az alapvető elrendezésre 2 fajta lehetőséget láttam megvalósíthatónak:

- 8 biten párhuzamosan beolvasás és 6 biten tudok vezérelni
- 12 biten párhuzamosan beolvasás és 2 biten UART vezérlés

A második megoldás mellett döntöttem, mivel így párhuzamosan több lábról tudok adatot beolvasni, ennek hátránya, hogy IC-t kell majd beilleszteni, amellyel az UART jelet szétbontom 8 bitre, mert a laborban arra volt igény, hogy 8 külön biten tudjunk vezérelni.

5.2 Kommunikáció a Front End elektronikával

5.2.1 A CLOCK jel megvalósítási lehetőségei

A számomra releváns rész a fentebb említett mérési elrendezésnél az az ADC vagy a diszkriminátor után shift regiszterekben eltárolt digitális adatok. A shift regiszter, más néven léptető regiszter, egy nagyon egyszerű digitális áramkör, amely flip-flopokból épül fel. Ha több flip-flopot egymás után kötünk (kaszkádosítunk), akkor az elsőre adott léptető jel (órajel) hatására az első bit tovább lép egy pozícióval, melyet a legutolsó flip-flop kimenetén tudunk kiolvasni. Tehát az alapvető cél egy minél nagyobb frekvenciájú órajel generálása, amellyel minél gyorsabban ki tudjuk olvasni az eltárolt adatokat a regiszterekből.

Az első ötletem az volt, hogy ha már az RPi processzora 700 MHz-es órajellel működik, akkor a számomra szükséges körülbelül 1-20 MHz-es jel előállítása nem jelenthet problémát, amennyiben néhány osztással csökkentem a processzor órajelét. Erre az előállított órajelre pedig interruptolni szerettem volna, azonban a linux rendszerek nem valósídejű operációs rendszerek, így mire egy interrupt érvényesítésre jut, relatív sok idő telik el. A mérések során kiderült, hogy körülbelül 20.000 db interruptot képes lekezelni másodpercenként az RPi, az általam használt beállításokkal, ami számomra kevés, továbbá ezek nem egyenletes, hanem sztochasztikus időpontokban lépnek fel. Így a hardveres órajel generálás lehetőségét kizártam, maradt a szoftveres megvalósítás. Ennek hátránya, hogy jóval lassabb, mint amelyet a hardveressel elő lehet állítani, előnye viszont, hogy a működését és az arra való reagálást nagyon rugalmasan lehet kezelni a programozás során.

Több programozási és script nyelv állt rendelkezésemre, amelyeket összehasonlítottam, hogy melyikkel állítható elő nagyobb frekvenciájú és stabilabb jelalak egy GPIO lábon. A fájlok egyszerűbb kezelése érdekében használtam egy **WinSCP** nevű programot, amely egy ingyenes SFTP és FTP kliens Windowsra. Ebben is IP cím alapján lehet csatlakozni az RPi-ra, ahol a username/password megadása után egy szokványos két paneles fájlkezelőt láthatunk. A fájlokat **Notepad++**-szal szerkesztettem.

- **Shell script:**

Az első változat, amit teszteltem az a shell script. Ezt akár a parancssorba is írhatnám, de én létrehoztam egy fájlt ennek érdekében és a következő egyszerű programmal teszteltem a GPIO 17-es lábat, ez a számozás a BCM szerinti, mivel a wiringPi-s libraryt nem include-lom. Ehhez a már említett WinSCP-ben létrehozok egy új fájlt, akár kiterjesztés nélkül és a következő sorokat írom bele:

```
1.#!/bin/sh
2.
3.echo "17" > /sys/class/gpio/export
4.echo "out" > /sys/class/gpio/gpio17/direction
5.
6.while true
7.do
8.  echo 0 > /sys/class/gpio/gpio17/value
9.  echo 1> /sys/class/gpio/gpio17/value
10.done
```

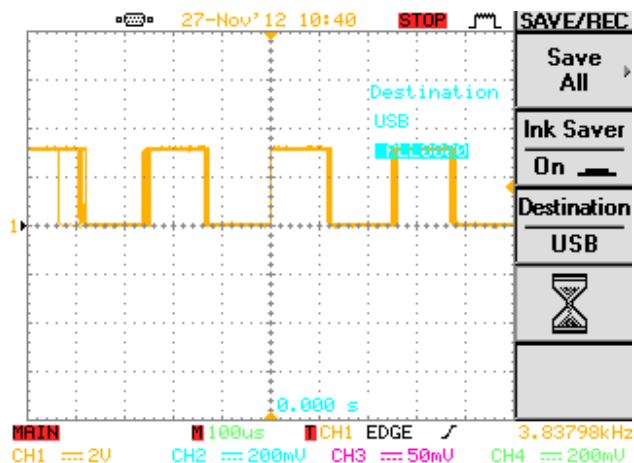
Az első sor azért szükséges, hogy az operációs rendszer tudja, hogy ezt a fájlt shell scriptként értelmezze. A GPIO sysfs interfészen keresztül vezérelhetjük a GPIO pineket a /sys könyvtárból. A bootoláskor a GPIO lábakat a kernel irányítja és alapvetően nem jelennek meg a sysfs rendszerben, amíg nincsenek exportálva. Ha az egyik GPIO lábat exportáljuk (pl.:17), akkor ennek hatására létrejönnek a következő fájlok:

- /sys/class/gpio/gpio17/active_low
- /sys/class/gpio/gpio17/direction
- /sys/class/gpio/gpio17/edge
- /sys/class/gpio/gpio17/power
- /sys/class/gpio/gpio17/subsystem
- /sys/class/gpio/gpio17/uevent
- /sys/class/gpio/gpio17/value

Jelen esetben nekünk elég, mindössze a direction (irány), illetve a value (érték) állítása. A rövid program tehát, annyit csinál, hogy a 17-es lábat beállítja output módba és a while ciklusban 1 és 0 értékeket, azaz 3.3 V-ot (1), illetve 0 V-ot (0) adunk rá. Amennyiben abban a könyvtárban vagyunk, amelyben a programunk van akkor a

```
# sudo ./programnév
```

paranccsal el tudjuk indítani a programunkat. Oszilloszkópon megvizsgálva a GPIO 7-es lábán lévő jelet (a földelést pedig az RPi egyik Ground lábára csatlakoztatva) az alábbi képet kapjuk:



20. ábra Shell script által generált CLOCK

A 20. ábra mutatja, hogy mindössze 3.8 kHz-es jelet kapunk amellyel, sajnos sokkal több mindent nem tudunk tenni, minthogy elfogadjuk, hogy nem shell scriptben írt programmal fogunk előállítani szoftveres órajelet.

A program elindulása után fellépett egy nem várt esemény, mégpedig az, hogy megszakadt a Wi-Fi kapcsolat, amint elindítottam a programot. Sikertelenül kideríteni, hogy az USB-s Wi-Fi adapterek viszonylag sok áramot fogyasztanak ezért, amikor a GPIO lábakat használom, akkor már nem elég az RPi-nak az 5 V / 0.7 A-es táplálás, ezért le kellett cserélni az adaptert egy másikra (jelenlegire), amelynek paraméterei 4,8 V / 1 A. Ezzel a működés problémamentes.

- Python:

A következő tesztelt programnyelv a Python. Ebben még nem volt alkalmam programozni, de rövid utánajárást követően egy nagyon egyszerű programmal sikerült tesztelni a GPIO 17-es lábat, itt is a BCM lábkiosztás szerinti számozást használtam. Ebben a nyelvben a kód tagolása a behúzáson alapul, így nem mindegy, hogy TAB-bal, vagy SPACE-szel strukturálja a felhasználó a kódot. A Python nyelvhez szükséges fejlesztő package a Raspbian tartalmazza, ezen kívül az RPi.GPIO package-t kell letölteni hozzá a már korábban említett paranccsal:

```
# sudo apt-get install RPi.GPIO
```

Ezután ismét létrehoztam egy újabb fájlt „.py” kiterjesztéssel és a következő tartalommal:

```
import RPi.GPIO as GPIO

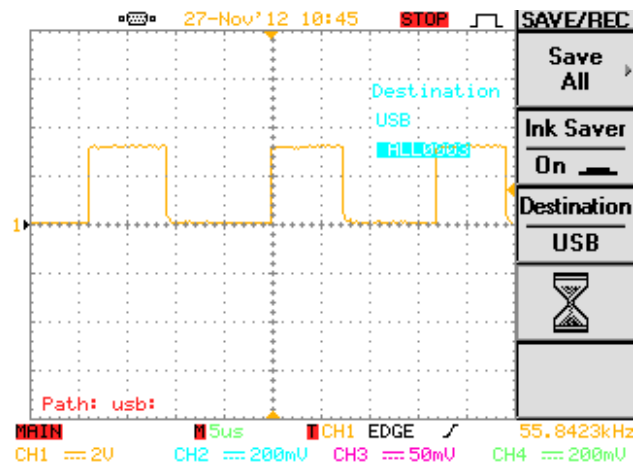
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)

while True:
    GPIO.output(17, False)
    GPIO.output(17, True)
```

Ezt a fájlt a következő paranccsal tudjuk futtatni, amennyiben abban a könyvtárban vagyunk, amelyben a programunk:

```
# sudo python programnév.py
```

Ismét az oszcilloszkópra tekintve a következőt látjuk (21. ábra):



21. ábra Python által generált CLOCK

Itt már egy 55.8 kHz-es órajelet figyelhetünk meg, amely körülbelül 15-szöröse a shell scriptben írt script által generált jelnek, de még mindig nem éri el az általunk kívánt néhány MHz körüli tartományt.

- C nyelv

A korábban telepített build-essential package-et felhasználva, most már minden további nélkül tudunk C nyelvű programokat fordítani. A fordítás után kapjuk meg a bináris fájlt, amelyet már tudunk futtatni. A már említett wiringPi-t letöltöttem, hogy használni tudjam az azokban található előre definiált függvényeket.

```
# cd /tmp
# wget http://project-downloads.drogon.net/files/wiringPi.tgz
```

Ezzel letöltöttük a wiringPi-t az RPi-re, ezután ki kell csomagolni és installálni:

```
# tar xzf wiringPi.tgz
# cd wiringPi/wiringPi
# make
# sudo make install
# cd ../gpio
# make
# sudo make install
```

A wiringPi-ben található néhány alapvető példa, amelyeket ki lehet próbálni, amennyiben igény van rá, jelen esetben elegendő egy alap C program megírása, amely a GPIO 7-es lábat felváltva állítja 0-ba és 1-be. Létrehoztam egy újabb fájlt a WinSCP-ben „.c” kiterjesztéssel, és a következőket írtam bele:

```
#include <wiringPi.h>
#include <stdlib.h>

int main() {
    if (wiringPiSetup () == -1)
        exit (1) ;

    pinMode(7, OUTPUT);

    while(1) {
        digitalWrite(7, 0);
        digitalWrite(7, 1);
    }
    return 0 ;
}
```

A wiringPi-hez tartozó headert include-oltam, mivel arra szükség van a fordításhoz, illetve a wiringPiSetup() függvénnyel inicializálom a GPIO lábakat, hogy a wiringPi számozásnak megfelelően lehessen rájuk hivatkozni, lehetőség van a BCM-es verziós számozás használatára a wiringPiSetupGpio() függvény meghívásával. A C fájl fordítása és futtatása a következő parancsokkal történik:

```
# gcc programnév.c -o programnév -l wiringPi
# sudo ./programnév
```

A GPIO 7-es lábon lévő jelről a következő képet kaptam az oszcilloszkópon (22. ábra):



22. ábra C nyelven által generált CLOCK

A 22. ábra mutatja, hogy 5.8 MHz órajelet kapunk a wiringPi használatával, amely megfelel az általam megoldandó feladathoz. Sikerült egy megfelelő nagyságú CLOCK jelet generálni, amellyel már megfelelő sebességgel tudjuk a shift regiszterekből a biteket léptetni, majd kiolvasni.

5.2.2 Adat kiolvasás megvalósítása

A beolvasás szorosan összefügg a CLOCK kiadásával, mivel a CLOCK hatására léptet a detektoron lévő shift regiszter, ezért egy órajele kiadás után tudunk beolvasni egy bitet, és ezt addig ismételjük, amíg az összes bitet be nem olvastuk. Ezeket a biteket el kell tárolni, majd a mérés végeztével kiírni a megfelelő adathordozóra, jelen esetben az SD kártyára. Ha egy eseményhez 8 csatornán jön az adat, akkor azt a következőképpen tudjuk olvasni:

```
uint8_t data[DATA_SIZE];
long index = 0;
pinMode(0, INPUT);
pinMode(1, INPUT);
pinMode(2, INPUT);
pinMode(3, INPUT);
pinMode(4, INPUT);
pinMode(5, INPUT);
pinMode(6, INPUT);
pinMode(7, INPUT);
pinMode(CLKPIN, OUTPUT);
.
.
.
while(!done) {

    digitalWrite(CLKPIN,1);
    int value = 0;
    value+=digitalRead(0)<<0;
    value+=digitalRead(1)<<1;
```

```

        value+=digitalRead(2)<<2;
        value+=digitalRead(3)<<3;
        digitalWrite(CLKPIN, 0);
        value+=digitalRead(4)<<4;
        value+=digitalRead(5)<<5;
        value+=digitalRead(6)<<6;
        value+=digitalRead(7)<<7;

        data[index] = value;
        index++;
        if (index>=DATA_SIZE) {
            index=0;
            done=1;
        }
    }
}

```

A value-ban eltárolom az egy eseményhez tartozó 8 bit értékét az adott láb értékével shiftelve, ezeket pedig a data tömb megfelelő elemébe mentem el, hogy később fel tudjuk dolgozni (event building).

Eredmény:

- 5.8 MHz-es CLOCK kiadása C nyelvű kóddal
- 8 GPIO lábról adat olvasása
- rövid tesztprogram összeállítása

5.3 Kommunikáció a trigger elektronikával

5.3.1 LAM jel olvasása:

A LAM jel olvasása az előző pontban ismertetett jelek előállításánál jóval egyszerűbben történik. Itt mindössze figyelniük kell a megadott GPIO láb feszültség változását. Ezt egy nagyon egyszerű módszerrel oldottam meg:

```

pinMode(x, INPUT);
while(!digitalRead(x)

```

Tehát az x GPIO láb értéke alapvetően nulla, ezért ha a programom elér oda, hogy vizsgálja az x pin értékét, akkor bekerül a while ciklusba, így addig ott marad, amíg nem jön egy LAM jel, ekkor az x láb 1-es értéket vesz fel és továbblépünk a while cikluson a következő parancsra.

5.3.2 BUSY jel kiadás

Ennek a jelnek a kiadása, a LAM jel beolvasása után történik, így amikor a while() ciklusból tovább lépünk, akkor a következő parancs ennek a lábnak az 1-be állítása:

```
pinMode(y, OUTPUT);  
digitalWrite(y,1)
```

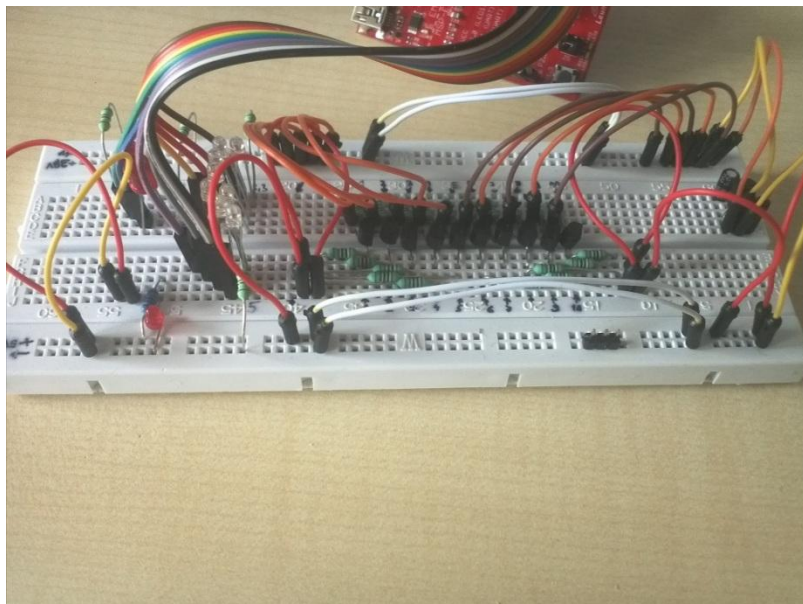
Eredmény:

- BUSY jel kiadása
- LAM jel beolvasása

5.4 Kommunikációhoz szükséges feszültség átalakítás

5.4.1 Breadboardon megvalósítva 3.3 V – 5 V – 3.3 V feszültség átalakító

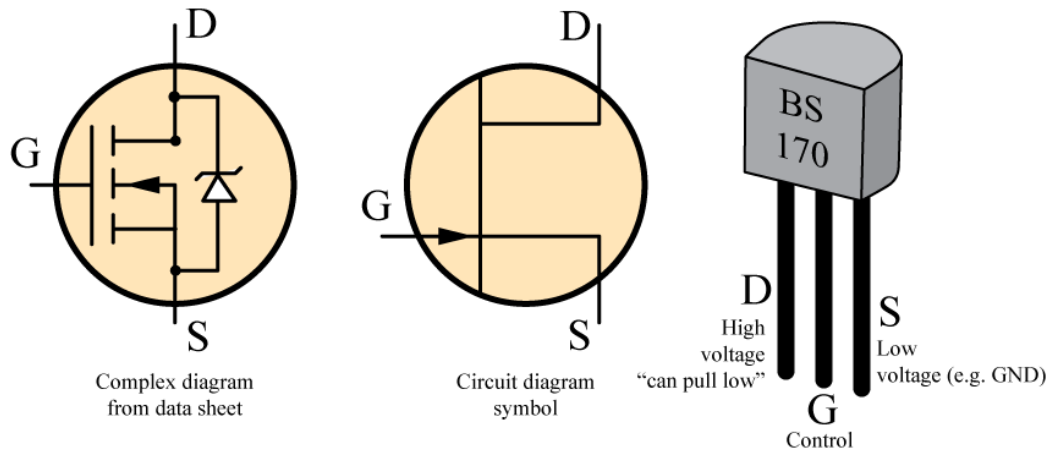
A detektorrendszer felől jövő jelek (LAM, Data) 5 V TTL típusú jelek, ezért szükség volt 5 V - 3.3 V feszültség átalakításra, illetve fordítva, amelyet breadboardon valósítottam meg és az alábbi képen látható:



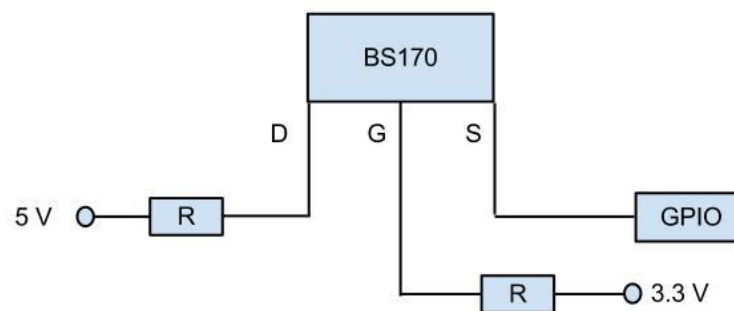
23. ábra 3.3 V – 5 V feszültség átalakító breadboardon

A mérési elrendezést egy Philips Application Note (AN97055) [18] szerint állítottam össze. Ennek lényege, hogy egy egyszerű MOS-FET-tel, egy kétirányú szintváltó áramkör segítségével, különböző tápfeszültségű eszközöket (3.3 V, 5 V)

tudunk rákapcsolni közös buszra. Különböző MOS-FET-ekkel különböző szintváltásokat tudunk elérni. Ennek egy egyszerűsített változatát valósítottam meg BS170-es MOS-FET-tel (24. ábra):

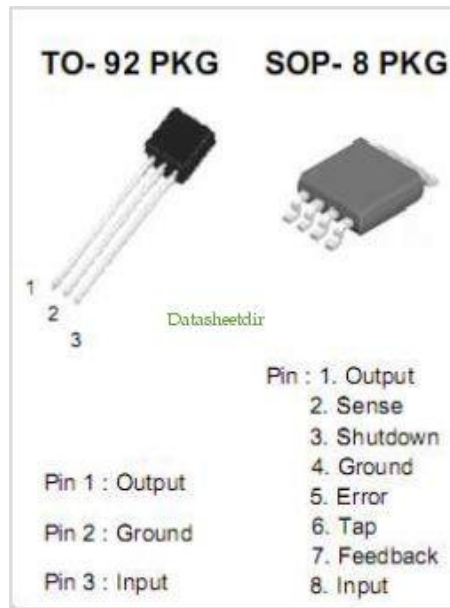


24. ábra BS170-es MOS-FET lábkiosztás [16]



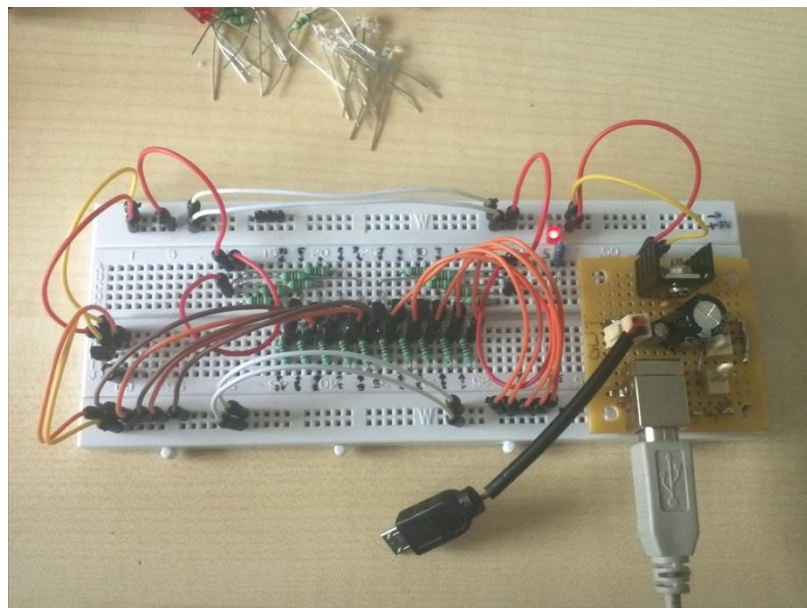
25. ábra BS170 lábakra csatlakoztatott jelek

Ahogy a 25. ábra is mutatja a MOS-FET-ek Drain és Ground lábaira állandó 5 V-ot, illetve 3.3 V-ot szükséges kapcsolni, a Source-ra pedig a GPIO lábakat. Nem akartam, hogy további feszültség problémák lépjenek fel, ezért nem közvetlenül az RPi-ről tápláltam (5 V output), hanem a breadboardot külön egy USB-B csatlakozón keresztül helyeztem feszültség alá, amely 5 V-ot biztosított a Drain oldalra, továbbá rákerült a breadboardra egy LM2905-ös feszültség regulátor, amellyel az 5 V-ból csináltam 3.3 V-ot (26. ábra).



26. ábra LM2905 feszültség regulátor lábkiosztása [17]

A végső kialakítás a következőképpen nézett ki (27. ábra):



27. ábra 3.3 V – 5 V feszültség átalakító megvalósítása breadboardon

A breadboardon található áramkör a félév végén lett megvalósítva, amelynek lépéseit a következő fejezetben ismertetek.

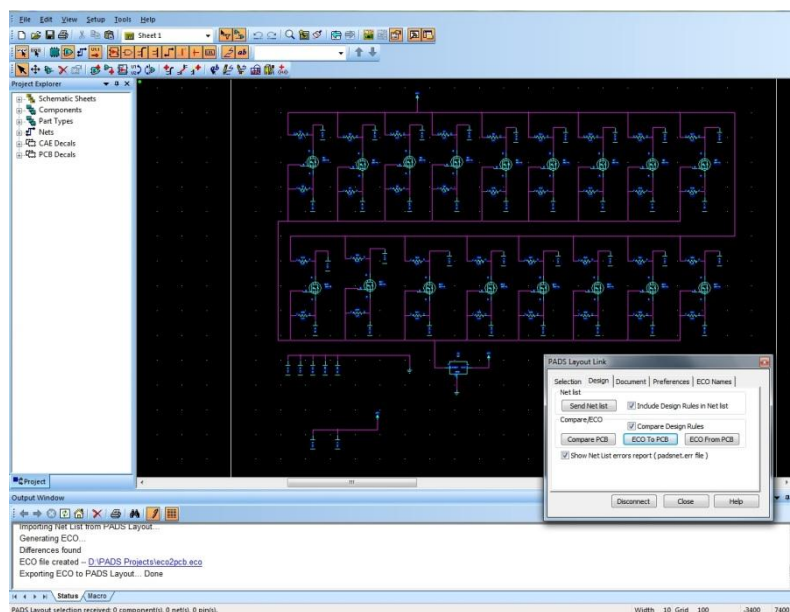
Eredmény:

- breadboardon összeállítottam egy 3.3 V – 5 V feszültség átalakítót, amely mindkét irányba használható

5.4.2 NYÁK-on megvalósítva a 3.3 V – 5 V – 3.3 V feszültség átalakító

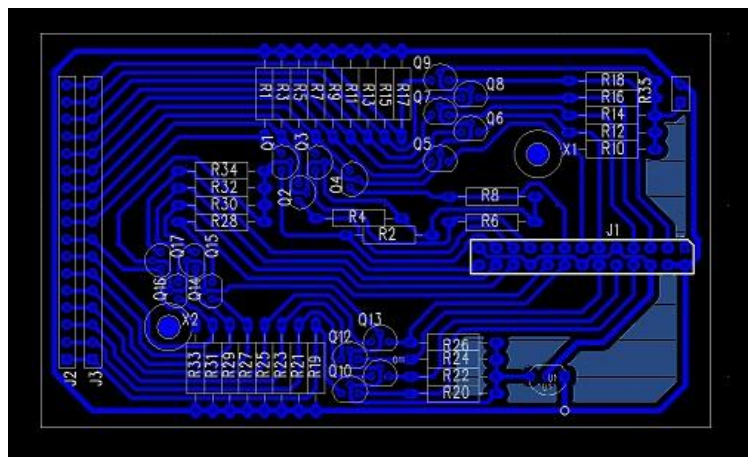
Az előző fejezetben összeállított feszültség átalakítót úgy gondoltam, hogy célszerű lenne legyártani NYÁK-on, mivel minden méréshez szükség van rá. A legyártásra csak a félév végén került sor, így a tesztek során a breadboardon lévő átalakítót használtam. Röviden ismertetem a megvalósítás lépéseit, amelyek „otthoni körülmények” között mentek végbe:

- Philips Application Note [18] (PAN) tanulmányozása
- Breadboardon megvalósítani az áramkört: megvásároltam a különböző alkatrészeket (ellenállás, MOS-FET, feszültség regulátor) hozzá, és a PAN szerint összekötöttem őket
- Mentor Graphics PADS Logic programot használtam a kapcsolási rajz sematikus ábrájának előkészítéséhez (28. ábra), amellyel az alkatrészek elhelyezését és az összeköttetések létrehozását tudtam megcsinálni:



28. ábra Mentor Graphics PADS Logic-kal a sematikus rajz elkészítése

- PADS Logic és PADS LAYout összelinkelése, Netlist (összeköttetések listája) és az alkatrészlista átküldése a PADS Layoutba
- PADS Layout programmal a fizikai layout (PCB board, alkatrészek elhelyezése, vezetékezés) megalkotása (29. ábra):



29. ábra Fizikai layout

- Az elkészült Layout terv TOP rétegének exportálása pdf fájlba, hogy nyomtatható legyen
- A gyártás folyamata:
 - Lézernyomtatóval kinyomtattam a layout tervet. Azért lézerrel és nem tintasugarassal, mert abban a festékben van műanyag, így a maratásnál nem marja le a számunkra lényeges részt.
 - Vasalóval először felmelegítettem a NYÁK lapot (60 x 100 mm), majd a kinyomtatott layout tervet ráhelyeztem a NYÁKra és a vasalóval 3-4 percig melegítettem, hogy rámásoljam a tervet. A papírt eltávolítva a NYÁKra felkerült a vezetékezés a réz felületre.
 - A NYÁK-ot Fe-III-Cl-ba (3 szabad vegyérték) kellett áztatni 5-6 percig. Ez a rézzel érintkezve oxidálja azt Fe-II-CL-dá, így csak a vezetékezés helyén lévő réz maradt a NYÁK-on. Az elkészült NYÁK-ot lefújtam Plastik 70-es univerzális bevonattal, hogy védjem az oxidációtól.
 - 0,4 mm-es fúróval kifúrtam az alkatrészek lábainak helyét, és 2 mm-es fúróval a távtartók helyét. Ezután beültettem és ráforrasztottam az alkatrészeket, valamint a tüskesorokat. A folyamatok képeken illusztrálva:



30. ábra Az elkészült feszültség átalakító NYÁK-on

A 30. ábra az elkészült NYÁK-ot mutatja, viszonyításképpen egy öngyújtóhoz és az RPi-hez. A NYÁK az RPi fölött lesz elhelyezve.

Eredmény:

- NYÁK legyártása, működése tesztelve

5.5 Webszerver

Felmerült bennem az az ötlet, hogy sokkal kényelmesebb lenne az, ha a mért adatokat a weben keresztül bárki elérhetné, így bárhol is végeznénk el a mérést a világon, a mérési eredmények rövid időn belül elérhetők lennének. Erre is lehetőség van az RPi-val, mivel egy egyszerű webszerverrel képesek lehetünk (pl.: xml formában) megjeleníteni az adatokat. Erre az nginx webszervert használtam, amely egy alacsony rendszerkövetelményű szerver, így ez számomra optimális megoldásnak tűnt. Az nginx installálás után, a tesztprogramot kicsit megváltoztatva úgy írtam meg, hogy az elmentett adatokat egy „.xml” fájlba mentse, amelyet a /var/www könyvtárba ment. A webszerver feladata az itt található fájlok kiszolgálása a http protokollon keresztül.

```
FILE *f1;
f1 = fopen ("/var/www/out.xml", "w");
int i;
fprintf(f1,"<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\n");
fprintf(f1,"<?xml-stylesheettype=\"text/xsl\" ref=\"xy.xsl\"?>\n");
fprintf(f1,"<pi>");
for(i = 0; i < MAX; i++)
{
```

```

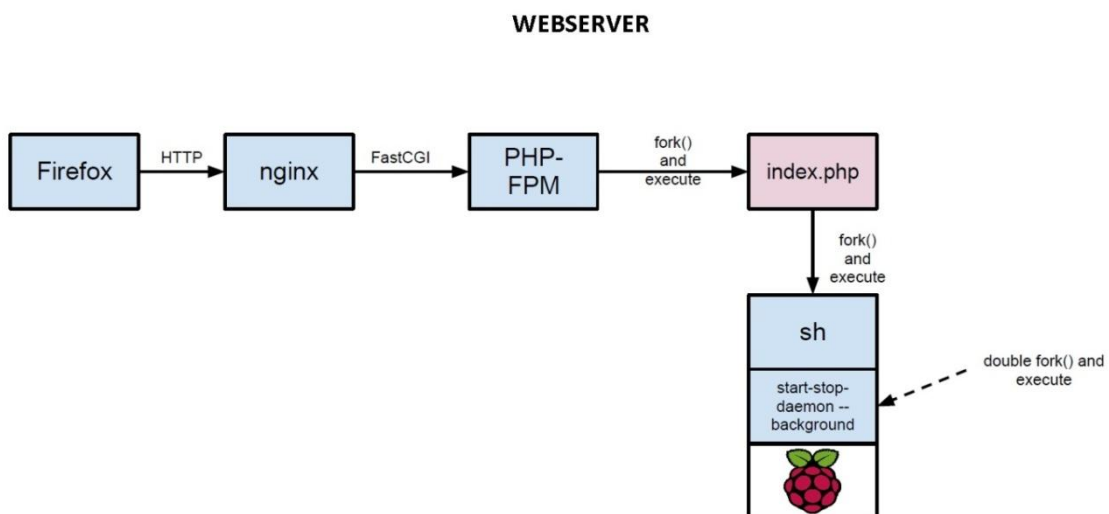
        fprintf(f1,"<measure><data>%d</data><diff>%d</diff></measure>\n",data
        [i],diff[i]);
    }
    fprintf(f1,"</pi>");
    fclose (f1);

```

A kódból látható, hogy az out.xml-hez egy xslt stíluslap fájlt is csatoltam, melynek segítségével a modern böngészők már elegáns formázott módon jelenítik meg az adatokat. Az xy.xsl-t fájlban lévő kód a mellékletben megtalálható.

Ezekkel a beállításokkal, ha megnyitjuk bármelyik böngészőt és beírjuk az RPi IP címét és a fájl nevét, amelyet meg akarunk nyitni, jelen esetben: <http://192.168.50.103/out.xml> -t akkor máris elérjük az elmentett adatokat.

Továbbá felmerült, hogy ha már a mért adatokat elérem bárhonnan, akkor akár az előkészített mérést is elindíthatnám bárhonnan. Tehát a következő célom az volt, hogy egy másik gépről elérve az RPi-t el tudjak indítani egy előre megírt és már lefordított C programot. Sajnos PHP-val nem foglalkoztam még eddigi tanulmányaim során, így elég sok utánajárást igényelt még ennek az alapszintű programnak a megírása is. Végül is sikerült megoldani a problémát és írni egy olyan egyszerű PHP programot, amely az nginx webservert segítségével kilistázza egy adott könyvtárban szereplő programokat és rákattintva egy program nevére elindítja azt. Ehhez szükség volt a már korábban említett nginx webserverre és pluszba telepíteni kellett a php5-fpm, php5-cgi package-eket. A megvalósított program kódja a mellékletben megtalálható, leegyszerűsített működését a 31. ábra mutatja:



31. ábra Webservert egyszerű vázlata

Eredmény:

- webszerveren keresztül elérhetővé váltak a mért adatok IP cím alapján
- egy adott könyvtárban lévő lefordított(!) C programokat el tudok indítani weben keresztül

6 Mérések

6.1 Mérések szimulálása

A különböző méréseket először szimulált, de realisztikus környezetben teszteltem, különböző eszközök segítségével. Ezeknek a méréseknek a célja az adatátvitel tesztelése, elérhető sebességének és időzítő lehetőségeinek felmérése. A beolvasási tesztek során egy-egy front-end modelljét hoztam létre, melyből mesterséges adatot lehetett kiolvasni.

6.1.1 A kommunikációhoz szükséges 4 alapjel (LAM, BUSY, CLOCK, DATA) tesztelése

A szimuláláshoz szükséges programhoz az 5. fejezetben ismertetett kódsorokat használtam fel, amely teljes egészében a mellékletben megtalálható, így a következő lépés a program tesztelése volt. Az első szimuláláshoz rendelkezésemre állt egy M430G2553 mikrokontroller, ami elsőre kissé bonyolultnak tűnhet, de ezzel már egy korábbi fejlesztés alkalmával volt már lehetőségem megismerkedni, így ennek a felprogramozása az MSP430 Launchpad segítségével könnyen megoldható volt. A mikrokontroller legfontosabb paraméterei: 16 MHz frekvencia, 16 KB flash memória, 512 B SRAM, 24 GPIO pin, 2 db 16 bites timer, UART, SPI, I²C, komparátor, 8 csatornán ADC (32. ábra).



32. ábra MSP430 Launchpad [19]

Az első tesztelés alkalmával a LAM, illetve BUSY jel működése LED-ekkel volt ellenőrizve. A mikrokontroller egyfajta front-end szerepet töltött be, amire egy 8 bites számláló programot írtam, amely minden bejövő órajelre növeli eggyel az adott (8 db) pineken található értéket, azért növekvő, hogy lássam, hogy minden érték beolvasása megtörténik-e. A mikrokontroller lábai össze voltak kötve az RPi GPIO lábaival, azaz 8-8 láb, csatorna az adatok miatt 1-1 a CLOCK miatt, ezen kívül még az RPi-ről kapott 3.3 V-ot, illetve megfelelő földpotenciált. A program a már korábban ismertetett lépésekből állt, azaz a LAM jel 1-be váltásakor (ezt manuálisan húztam fel magas (3.3 V) értékre), a BUSY jelet 1-be állítom és el kezdem küldeni a CLOCK-ot a mikrokontrollernek, majd az RPi-val a 8 darab GPIO lábon olvasom a bejövő adatokat. A mérési elrendezés a következőképpen nézett ki (33. ábra):



33. ábra Adatok kiolvasása mikrokontrollerről

Már az első tesztek során világossá vált az a probléma, hogy az RPi-vel megvalósított DAQ rendszer gyakran, például ebben az esetben is, jelentősen gyorsabb, mint a (szimulált vagy valódi) front-end. Ez arra mutat rá, hogy jóllehet bármilyen front-end kezelhetővé válik, érdemes a front-end sebességét úgy kiválasztani, hogy kihasználható legyen a DAQ teljes sávszélessége, a mikrokontrollernél ezt egyszerű késleltetéssel oldottam meg, konkrétan

```
delay(0);
```

függvényt használtam, amelynek a zárójelben lévő paraméterét milliszekundumban kell megadni, és az érték azt jelenti, hogy ekkora időtartamra

minimum feladja a futási jogot a program és az ütemezőre bízva mikor kapja meg a ismét a futási jogot. Az órajel 1-esbe, illetve 0-ba állítása után is egy-egy késleltetést berakva a program megfelelően működött, tehát a program egy növekvő számsort olvasott be 0 – 255 között.

Eredmény:

- RPi GPIO lábain keresztül sikerült adatokat beolvasni egy mikrokontrollerről, emellett a BUSY, LAM, illetve CLOCK jelek is megfelelően működtek.

6.1.2 Saját előállítású trigger jelek által vezérelt mérési elrendezés tesztelése

A mérés során megvalósítottam egy szimulált trigger jelet kiadó impulzusgenerátort, egy LAM – BUSY elvű trigger logikát, illetve egy fix adatbevitelt adó szimulált front-end tárolót. Ehhez a következő 3 db IC-t használtam fel:

- 74HCT14: Nagysebességű Si-gate CMOS eszköz, amely kompatibilis az alacsony schottky TTL jelekkel. 6 db Schmitt invertert tartalmaz, amelyek a lassan változó jeleket átalakítják gyorsan változó jelekké, azaz jitter mentes kimeneti jeleket generálnak.
- 74HCT00: Nagysebességű Si-gate CMOS eszköz, amely kompatibilis az alacsony schottky TTL jelekkel. 4 db 2 bemenetes NAND kaput tartalmaz.
- 74HCT165: Nagysebességű Si-gate CMOS eszköz, amely kompatibilis az alacsony schottky TTL jelekkel. 8 bites shift regiszter. Működése: ha a /PL (parallel load, default 1) bemeneten alacsony jelszint van akkor a D0-D7 csatornákon 8 bitnyi adat aszinkron betöltődik a regiszterbe. Ha a /PL magas jelszintű az adatokat sorosan ki tudom olvasni a Q7 lábon megfelelő Clock mellett (ezt a fajta tárolót több, ténylegesen használt front-end esetén alkalmazza a REGARD csoport.).

A tesztáramkör az alábbi képen (34. ábra) látható. A részeinek a működése a következőképpen történik. A trigger generátoros kapcsolás első fele egy oszcillátort helyettesít. Itt gyakorlatilag egy Schmitt-inverteres RC oszcillátort valósítottam meg. Az oszcillátor után következő One-shot kapcsolás az oszcillátor által előállított jelből egy ugyanakkora periódus idejű, de annál jóval kisebb kitöltési tényezővel rendelkező impulzusszerű jelet állít elő. Ezek a tüskék szolgálnak a trigger jel szimulálására,

értékük lehet 1 mikroszekundum vagy akár 10 nanoszekundum is attól függően, hogy a kapcsolásban mekkora az R és C értéke. Tehát a trigger generátor kimenetén megjelenik a TRIGGER IN jel, amely megfelel a trigger egység által kiadott, mérést indító jelnek. A BUSY jellel kell kialakítani a megfelelő kapcsolást úgy, hogy ha érkezik egy trigger jel, akkor az egy LAM = 1-t generáljon, majd a BUSY-t billentse be 1-be. A BUSY bebillenése viszont blokkolja a további triggerrek érvényre jutását addig, amíg aktív állapotban van a BUSY. Ez egy invertált bemenetű RS tároló kapcsolással lett megoldva, mivel ennek a segítségével tárolhatunk egy bemenő impulzust. Egy segédtábla mutatja a TRIGGER IN és /BUSY IN bemenetű Schmitt trigger igazságtábláját invertálás nélkül, mert később könnyebb az eredeti jelekkel számolni (2. táblázat):

TRIGGER IN	(BUSY IN) → /BUSY IN	TRIGGER IN & /BUSY IN
L	(H) → L	L
L	(L) → H	L
H	(H) → L	L
H	(L) → H	H

2. táblázat TRIGGER IN és /BUSY IN igazságtáblája

Az alábbi segédtábla (3. táblázat) az RS tároló működése a ponált jelekkel:

TRIGGER IN & /BUSY IN	BUSY IN	Q = LAM OUT
L	L	Q előző
H	L	H
L	H	L
H	H	X (határozatlan)

3. táblázat RS tároló működése a kapcsolásban szereplő ponált jelekkel

A trigger logika működés során fellépő esetek bemutatása (4. táblázat):

	TRIGGER IN	BUSY IN	Q = LAM OUT
--	------------	---------	-------------

1.	0	0	0 (Q előző)
2.	1	0	1
3.	0	0	1 (Q előző)
4.	0	1	0
5.	1	1	0
6.	1	0	1

4. táblázat Trigger logika lehetséges esetei

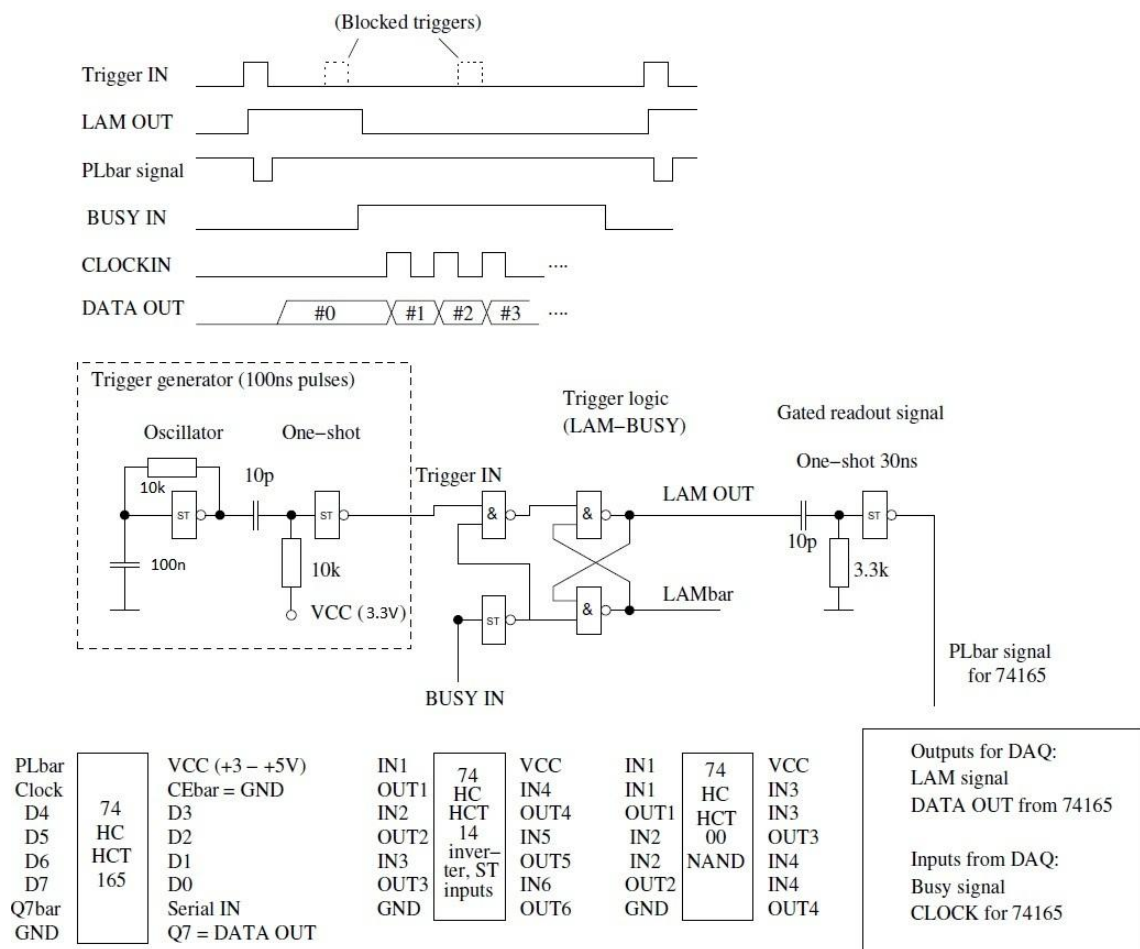
1. eset: triviális
2. eset: jön egy trigger, ennek hatására bebillen a LAM OUT (=1), aminek a hatására a DAQ megkezd a kiolvasást, illetve BUSY = 1 állapotba kerül
3. eset: LAM OUT (=1) jelet az RPi beolvassa és bebillenti a BUSY-t 1-be, de mielőtt ez megtörténik, a rövid trigger jel megszűnik, azaz a kimenet magas értéke jelenti a további triggerok blokkolását
4. eset: bebillent a BUSY, nem jön trigger, LAM OUT nullába
5. eset: BUSY aktív, jön egy (blokkolt) trigger a LAM OUT marad alacsony szinten
6. eset: lásd 2. eset

Az RS tároló segéd táblázatban (3. táblázat) látszik, hogy van olyan lehetőség miszerint a kimenetnek határozatlan az értéke, azonban ezt kiküszöböltük, mivel az RS tároló bemenetére nem kerülhet egyszerre mindkét jel H szinten.

A LAM OUT 1-es jelből ismét egy one-shot kapcsolást használva egy rövid impulzust generálunk. A PLbar jel default aktív jel és néha, amikor fellép a LAM OUT jel, akkor megjelenik egy rövid nulla impulzus, amely hatására a 74HCT165 shift regiszter 8 biten beolvas adatot (parallel load) majd ha beolvasta utána letiltja a felülírást a következő PLbar jelig. Amíg a BUSY jel be van billentve, addig nem kapunk újabb LAM OUT 1-t (ezáltal újabb alacsony /PLbar jelet sem) tehát megkezdhetjük a rögzített adatok kiolvasását a regiszterből az RPi-val adott CLOCK felfutó élére. Összességében az RPi-val a következő lépéseken kell végigmenni:

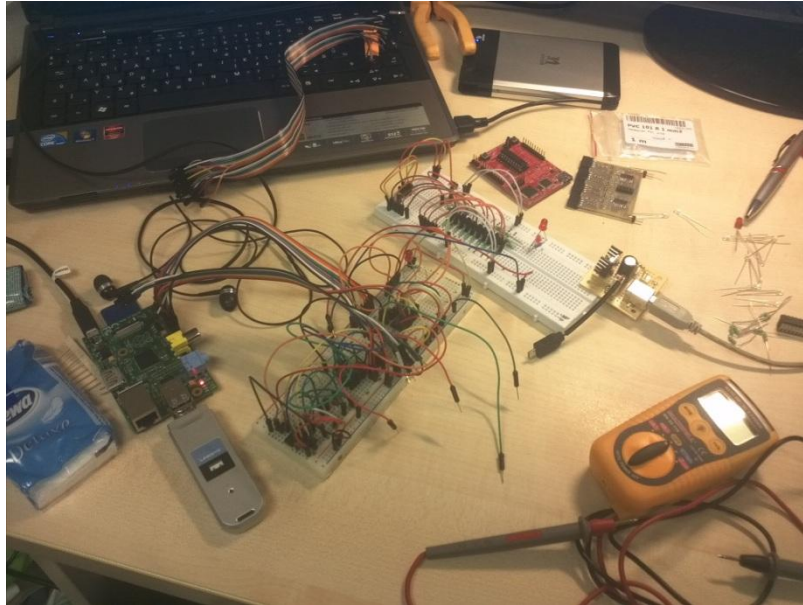
- (0). A LAM 1-ben van-e? Ha nem, akkor figyeld tovább. Ha igen, akkor lépj az (1)-re.

- (1). Vidd fel a BUSY-t 1-be.
- (2). Olvass be egy bitet
- (3). Vidd a CLOCK-ot 1-be, aztán vidd a CLOCK-ot 0-ba.
- (4). Ha nincs minden bit beolvasva, menj (2)-re. Ha megvan minden (fix számú, RPi-ben paraméterként megadott) bit, menj (5)-re.
- (5). Vidd le a BUSY-t 0-ba. Menj (0)-ra.



34. ábra Saját előállítású triggeres rendszer vázlata

A mérés megvalósítását a 35. ábra mutatja. Fix, állandó adatokat sikeresen olvastam be a shift regiszter D0-D7 lábairól.



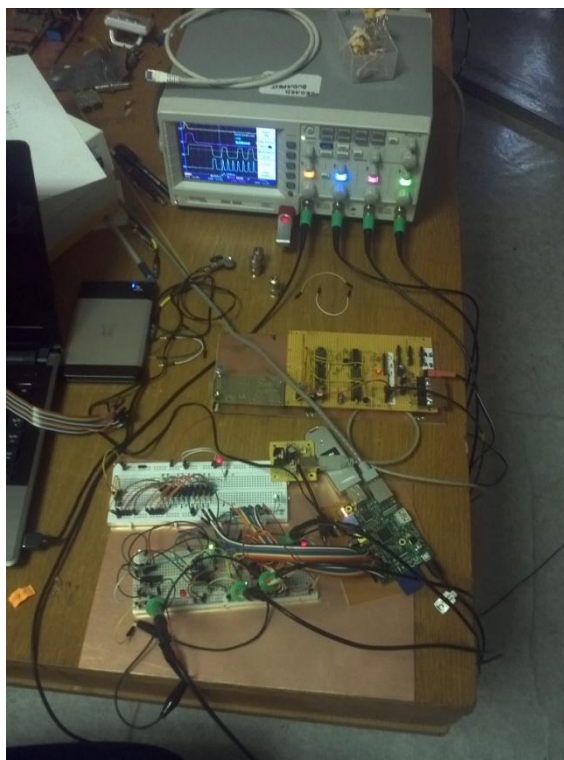
35. ábra Saját előállítású triggeres rendszer megvalósítása

Eredmény:

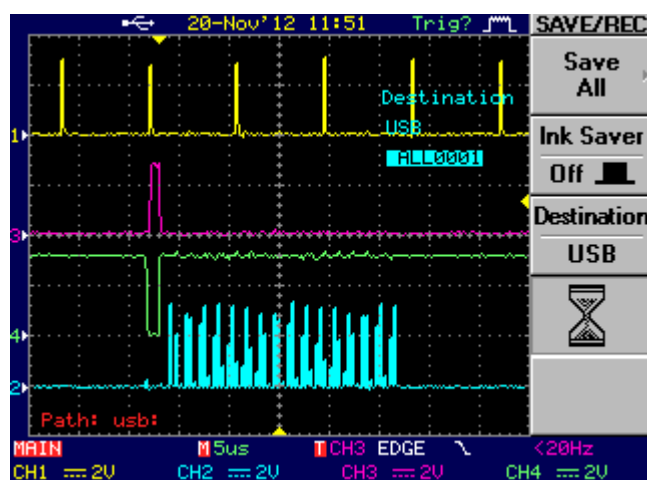
- egy saját trigger rendszer és egy kezdetleges front end sikeres megvalósítása, majd az ezekkel való kommunikáció megvalósítása az RPi által használt jelekkel, fix adatok sikeres beolvasása
- a megalkotott mérési elrendezés kifogástalanul működött, ehhez nagyon hasonló jelek kezelése a végső cél

6.1.3 Különböző sebességű trigger jelek számlálása

A 6.1.3 alfejezetben bemutatott szimuláció a KFKI-ban is tesztelésre került, oszcilloszkóppal ellenőrizve a működését, az alábbi képen (36. ábra) látható az összeállítás, illetve a szkóp képe (37. ábra):



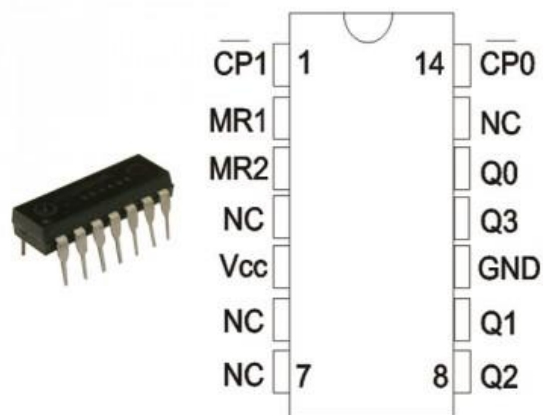
36. ábra Saját előállítású trigger rendszer



37. ábra Saját előállítású trigger rendszer jelei, CH1: trigger, CH2 CLOCK, CH3 LAM, CH4 /PLbar

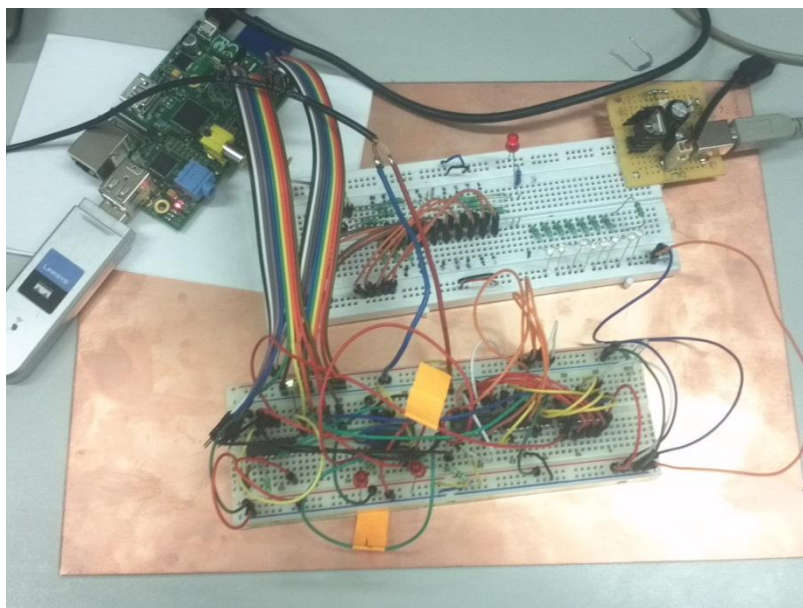
Az oszcilloszkóp képen (37. ábra) is jól látszanak a trigger jelek (CH1, sárga), amelyek impulzuszerűek, a LAM (CH3, lila), melyet egy bejövő trigger vált ki, az aktív /PLbar (CH4, zöld), amely kap egy nulla impulzust és a kiolvasás megkezdése, azaz az órajelek (CH2, kék) kiadása a shift regiszter felé.

A következő lépés az RPi beolvasási sebességének tesztelése volt. Felhasználtam két darab 74HCT93E nagysebességű CMOS 4 bites számlálót, kaszkádosítva, így egy 8 bites számlálót kaptam (38. ábra):



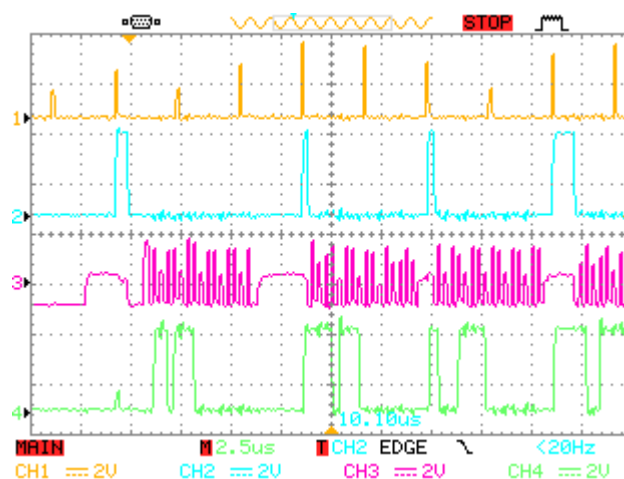
38. ábra 74HCT93E 4 bites számláló lábkiosztása [20]

A mérés lényege, hogy az előző pontban szereplő trigger generátor részt egy jelgenerátorral helyettesítettem, így könnyebben tudtam állítani a kiadott jel frekvenciáját és nem kellett ellenállásokat, illetve kondenzátorokat cserélni. A mérés célja az volt, hogy megvizsgáljam hány darab trigger jelet veszít a DAQ (nagyon nagy frekvenciájú trigger esetén). Ennek érdekében a jelgenerátorral előállított trigger jeleket a 8 bites számlálóra kötöttem, így minden beérkező trigger jel lépteti a számlálót, azaz a beérkező triggereket számolom (a trigger logika által blokkoltakat is). A NAND kapus IC-be is bekötöm a trigger jelet az eddig megszokott módon a TRIGGER IN helyére, a számláló outputjait pedig a shift regiszter data lábaihoz kötöttem. Az is kérdés volt, hogy megnézzem mekkora az a frekvencia, amelyen még az összes bejövő trigger jelet be tudom olvasni, anélkül, hogy egyet is elszalasztanék. Ezzel azt tudom lemérni, hogy mennyi idő alatt tudom elvégezni a mérést és ebből következően mekkora az a legkisebb időkülönbség, amelynek el kell telnie két részecske becsapódása között, hogy kivétel nélkül mindegyikre el tudjam végezni a mérést. A mérési összeállítás az alábbi képen látható (39. ábra). A DAQ 16 bitet olvasott be (ebből 8 bit volt a számlálónak megfelelően értékelhető).



39. ábra Trigger számlálós mérési elrendezés

Az eredmények azt mutatták, hogy 100 kHz frekvenciájú trigger mellett, gyakorlatilag még az összes beérkező jelre tudott reagálni és „lemérni” azokat (triggerenként 16 bitet beolvasni), 100 – 350 kHz között pedig minden másodikat (néha harmadikat) tudja elkapni, amely az alábbi képen (40. ábra) is látható:



40. ábra Trigger számlálós mérés jelei oszcilloszkópon CH1: trigger, CH2 LAM, CH3 CLOCK, CH4 adatok

Ez az eredmény tulajdonképpen dolgozatom lényegi konklúziója: 16 bites adatok esetén 100 kHz-es, azaz a tipikus CAMAC rendszernél 300-szor gyorsabb, az LPT-s megvalósításnál közel tízszer gyorsabb adatbeolvasás érhető el az RPi-re alapuló DAQ rendszerrel.

Eredmény:

- saját triggeres elrendezés helyes működésének ellenőrzése szkóppal
- jelgenerátorral sebesség tesztelése a bejövő triggerek számolásával, amelyet 2 db 4 bites számlálóval valósítottam meg
- 100 kHz-es adatbeolvasás (triggerenként 16 bitre) sikeres demonstrálása

6.1.4 Az RPi-val való vezérlés megvalósítása UART-on keresztül

A 4 alapjel után a fennmaradó egy darab vezérlő jel megvalósítására, mint már említettem az UART-on való kommunikációt választottam, mivel így több GPIO láb marad az adatok beolvasására és az RPi támogatja az UART-ot. Ez az adatátviteli típus soros átvitelnél aszinkron módon továbbítja a küldött adatot, mely adatok elé és mögé start, illetve stop bitet fűz. Az UART bekonfigurálásához a következő lépéseket kellett megcsinálni:

```
# sudo nano /boot/cmdline.txt
```

Ebben a fájlban a bootoláskor automatikus konfigurációkat tudjuk beállítani. Itt ki kell törölnünk a következő két sort:

- console=ttyAMA0,115200
- kgdboc=ttyAMA0,115200

Erre azért van szükség, mert defaultból a linux kernel használja az UART lábakat soros konzolként, így ezt ki kell kapcsolnunk ahhoz, hogy saját magunk tudjuk irányítani. Ezen kívül még egy helyről ki kell venni az UART-os konfigurációt:

```
# sudo nano /etc/inittab
```

Ebben a fájlban a következő sort kell kikommentelni:

- # 2:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

A változtatások után szükséges egy reboot, hogy az új beállítások érvényre jussanak. Az UART felélesztéséhez szintén segítséget nyújt a wiringPi. A szokásos wiringPi.h helyett a wiringSerial.h-t kell include-olni a programunk elején. A létrehozott új fájl a következőképpen néz ki:

```
#include <wiringSerial.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int handle = serialOpen ("/dev/ttyAMA0", 115200);
```

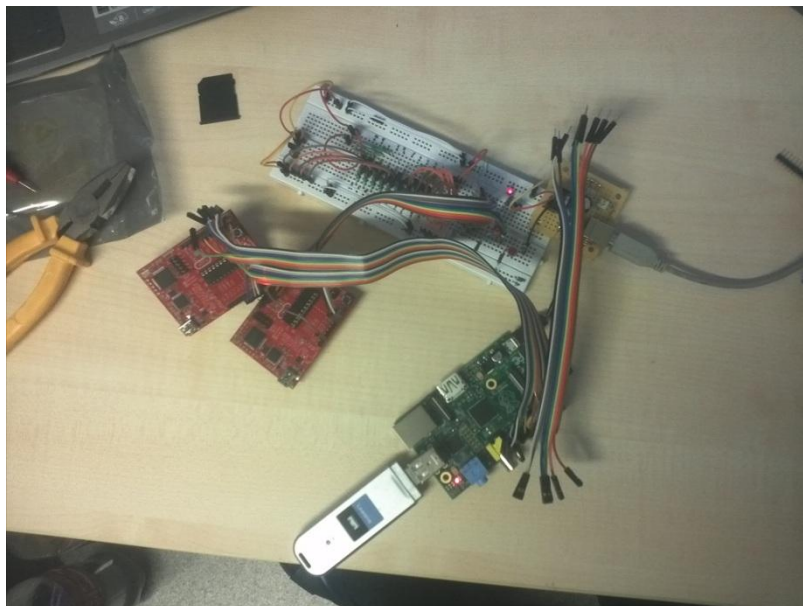
```

printf("input chars: %s\n",argv[1]);
serialPutchar(handle,atoi(argv[1]));
serialClose(handle);
}

```

Ezzel a megoldással parancssorból tudunk adni értéket 0 – 255 (8bit) között. Az UART-tal elérhető maximális sebessége 115200 baud/sec, amely bőven elegendő számunkra. Mint említettem ennek a megvalósításnak a hátránya, hogy szükség van egy IC-re, amely a soros UART jelet visszabontja 8 bit-re (vagy ennek többszörösére), mivel a laborban ilyen módon van megoldva a vezérlés. A gyakorlatban mégis jó megoldásnak tűnik ez a módszer, hiszen a vezérlés sebessége nem kell, hogy olyan nagy legyen, mint az adatbeolvasás.

Ennek a programnak a tesztelésére a 6.1.1 fejezetben szereplő eszközök mellett felhasználtam még egy M430G2553 mikrokontrollert, mivel ezen is van beépítve UART, valamint rendelkezésre áll több mint 10 szabad láb. A szimuláció úgy történt, hogy az RPi UART Txd kimenetére a mikrokontroller UART RXD bemenetét kötöttem és a mikrokontroller 8 lábát bekötöttem a breadboardra, hogy LED-ekkel ellenőrizni tudjam, hogy az RPi-ről kiküldött adat helyesen érkezett-e meg a mikrokontrollerhez és hogy az megfelelően alakította-e vissza az UART-os bitfolyamot 8 párhuzamos bitre. A tesztelés során semmilyen probléma nem lépett fel, meghatározott számú beolvasott adat után (az első mikrokontrollerről), meg lett hívva az uart.c program így felvillantak a megfelelő LED-ek. A mérési elrendezés az alábbi képen (41. ábra) látható:



41. ábra RPi UART-on való vezérlésének szimulációja

A fenti összeállítással gyakorlatilag sikerült megvalósítani egy nagyon speciális esetét a feladatnak. Fontos viszont, hogy a mikrokontrolleres megoldás nagyon lassú egy gyakorlatilag releváns front-endhez képest, tehát szükség volt egy egyszerűbb, pusztán logikai áramkörökből felépülő és ezért gyorsabb szimulációs környezetre.

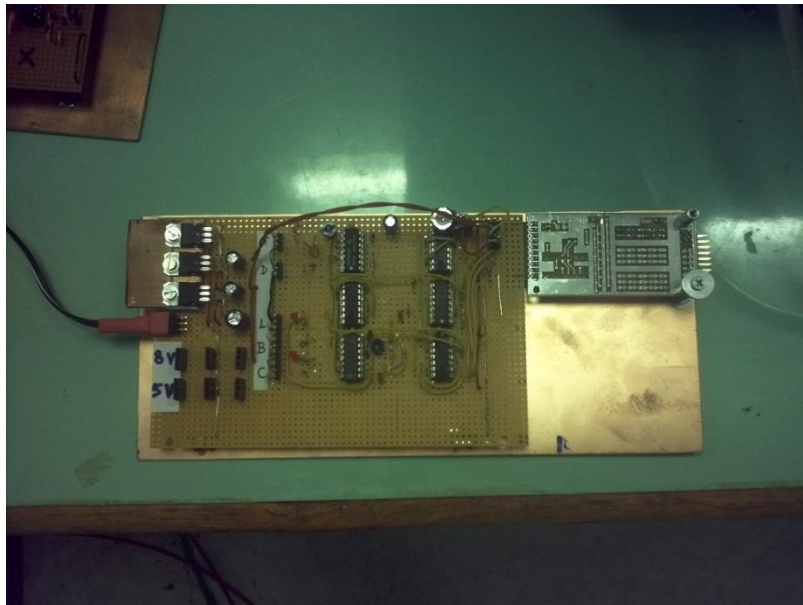
Eredmény:

- egy újabb mikrokontroller alkalmazása, amellyel a vezérlést szimuláltam sikeresen UART-on keresztül

6.2 Konkrét mérések

Két fajta detektornak a mérését sikerült valós körülmények között lemérnem. Az egyik egy szcintillátor, a másik pedig egy proporcionális kamra, a már említett „Small Gap Chamber”. Mindkét detektortípusról már volt szó a bevezetőben, de részletesebb működésüket a következő pontokban fejtem ki. Mindkét mérésnél a front end szerepét az alábbi képen (42. ábra) látható kártya töltötte be, így először ennek a felépítését mutatom be.

6.2.1 A REGARD által fejlesztett, ADC mérést végző front-enddel való kommunikáció



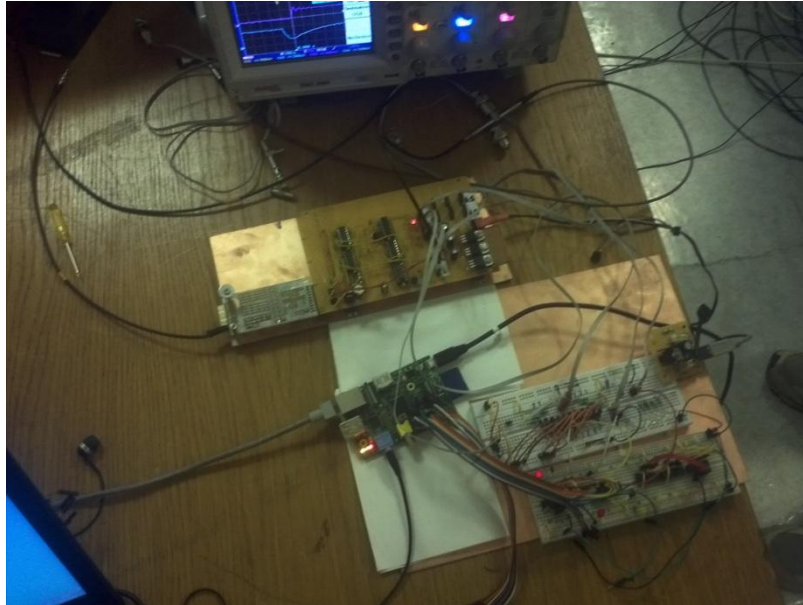
42. ábra REGARD front-end

A front-enden található:

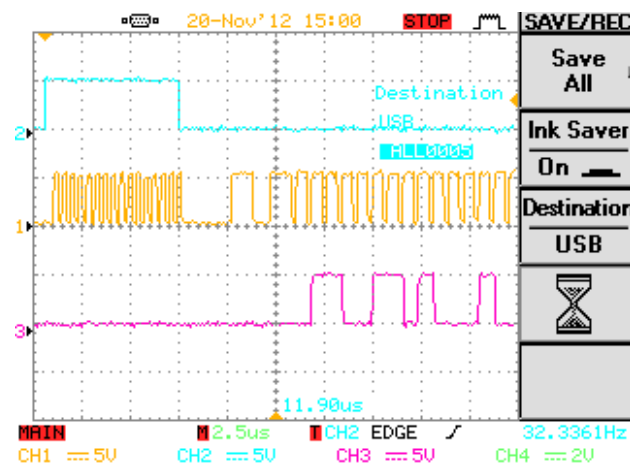
- trigger unit (BUSY, LAM, bejövő trigger kezelése)
- vezérlő jel kezelése és időzítésének állítása potméterrel
- ADC (National Semiconductor gyártmányú, ADC121S021 típus)
- shift regiszter (16 bites), ami automatikusan tárolja az ADC adatait
- konstans feszültség szint hozzáadása a beérkező analóg jelhez, potméterrel állítható

A front-end 5 V-os jeleket ad ki, így szükség van a feszültség átalakítóra, mivel az RPi csak 3.3 V toleráns. A két mérés általános összeállítása hasonló az eddigiekhez. A detektoron áthalad egy részecske és trigger jelet generál, amit a trigger egység feldolgoz. Ha a trigger jel bebillentette a LAM-ot 1-be a front-enden, akkor azt az RPi-val feldolgozom és bebillentem a BUSY-t aktív állapotba, hogy blokkoljam a következő trigger eseményeket a kiolvasás során. Majd elkezdődik az adatok kiolvasása a shiftregiszterekből (16 bit), azaz CLOCK-ot küldök a front-endnek és a Data vezetéken olvasom ki az adatokat. Természetesen minden jelet a feszültség átalakítón át vezetek, attól függően, hogy 3.3 V-ból akarok 5 V-ot vagy fordítva. Az első mérésnél a front-end-re nem kapcsoltam detektort, ilyenkor a konstans feszültség hozzáadását végző potméter értéket tudom kiolvasni. A front-endből érkező 16 bitből nem mind hasznos:

- 0.,1.,2. bit értéke fix 0
- 3-15 bit a hasznos adat
- 16. bit értéke fix 0



43. ábra REGARD front-end mérési elrendezése



44. ábra REGARD front-end mérése detektor nélkül

A szkóp képen (44. ábra) látható a vezérlőjel (CH2, kék), amely a mérés kezdetét jelenti, ekkor kezd el beolvasni a front-end az ADC analóg bemenetén lévő jeleket. A CH1, sárga jel első felén az ADC belső CLOCK-ja látható, ami az analóg jel beolvasásához szükséges majd a második felén az RPi-vel kiadott CLOCK látható. A CH3, lila jel a front-endből kiolvasott adat. Látható, hogy az első 3 bit értéke valóban nulla.

6.2.2 Szcintillátor mérése

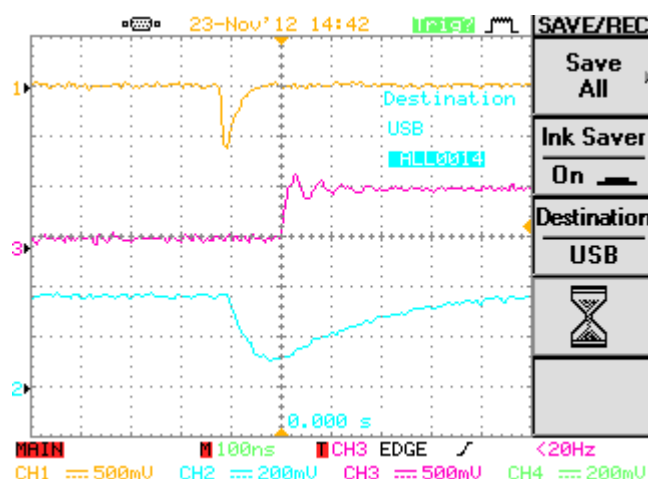
A képen (45. ábra) egy szcintillátort, illetve egy ^{90}Sr (Stroncium) izotópot láthatunk egy bronz hengerbe zárva, hogy ne sugározzon mindenfelé csak arra, amerre a kollimátor (lyuk) van rajta. Ez béta-sugárzást (elektronokat, $E \sim 2 \text{ MeV}$) idéz elő,

gyakorlatilag ezeknek a részecskéknél a detektorban hagyott energiáját mérjük, fényjel formájában. A szcintillátor 2 párhuzamos detektorból áll, az első, amely a forráshoz közelebb van 2 mm vastag, a távolabbi 4 mm vastag. A ^{90}Sr által kilőtt elektronok mindössze 3-4 milliméteren keresztül rendelkeznek a megfelelő energiával ahhoz, hogy át tudjanak haladni az első detektoron majd a másodikban el is nyelődnek, mivel hamar elveszítik energiájukat. A hátsó detektort fogjuk trigger jelként használni, azaz ha abba becsapódott egy részecske, akkor nagy valószínűséggel az elsőn is átment. Vannak kivételek persze, amikor nem a saját forrásunkból érkezik a részecske, hanem kozmikus sugárzás következtében egy olyan szögből érkezik egy részecske, hogy csak a második detektoron halad keresztül az elsőn pedig nem, ekkor a front-enden lévő konstans erősítés értéket (bemenőjel nélkülit) kell megkapnunk, mivel abban az időpontban az első detektoron 0 energia jelenik meg.

Az első detektorlapon megjelenő jelet késleltetnünk kell ahhoz, hogy a második lapról érkező trigger jel idejében elérjen a front-endre és kiváltsa a vezérlő jel kiadását, azaz a mérés kezdésének időpontját még mielőtt eltűnne az energiaszint az első detektorról. Ezt az a szerencsés körülmény oldja meg, hogy a front-end bemenetén is van egy erősítő-jelformáló kapcsolás, aminek a saját, 100 nanoszekundum nagyságrendű késleltetése jól kompenzálható a mérés pillanatának időzítésével.



45. ábra Szcintillátor, illetve az ^{90}Sr



46. ábra Szcintillátoros mérés oszcilloszkópon

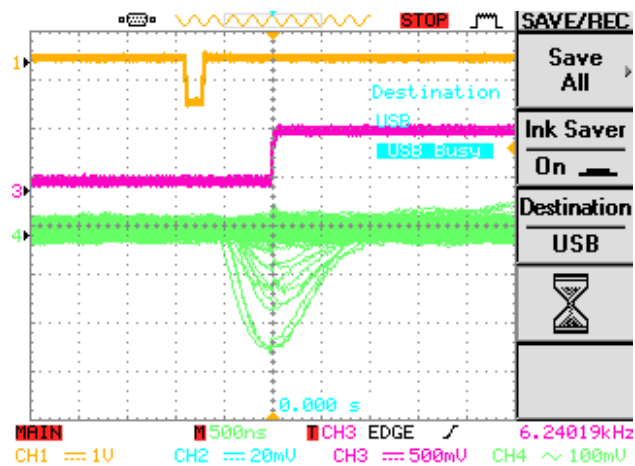
A szkóp képen (46. ábra) CH1, sárga jel a trigger, azaz a második detektor jele, a CH2 kék csatorna az első detektoron megjelenő jel, amelyet mérünk. A CH3 pedig a vezérlőjel kiadásának idejét mutatja, azaz a mérés kezdetét, ebben az időpillanatban mintavételez az ADC a detektor által jövő jelből. A vezérlőjel kiadásának az idejét tudjuk állítani a front-enden lévő másik potméterrel, mégpedig azért, hogy az első detektoron megjelenő jelet akkor mintavételezzük, amikor az a maximumán (optimális jel-zaj viszonynál) van.

6.2.3 SGC mérése:

A Small Gap Chamber speciális, az ionizáció természetét vizsgáló eszköz. A DAQ része egy az egyben megegyezik az előző pontban ismertetett szcintillátoros mérésben alkalmazottal. Az SGC detektornál szintén saját magunkra triggerelünk, ahol egy oszcillátorból kijövő jelek adják a trigger jeleket a rendszernek, de ezek az impulzusok be vannak vezetve egy erősítőn keresztül egy UV LED-be is. Minden triggerrel egy időben az UV LED fényimpulzust bocsát ki, ami az SGC felső katódjából kiszakít egy elektront, ami a kamrában lévő egyetlen anód szál felé kezd el mozogni. Itt a kialakuló, egyetlen elektron által indított lavina miatt egy adott feszültség szint jelenik meg az anód szálon, ezt a jelet vezetem a front-endbe a jelerősítő, illetve a jelformálón keresztül.



47. ábra SGC detektor

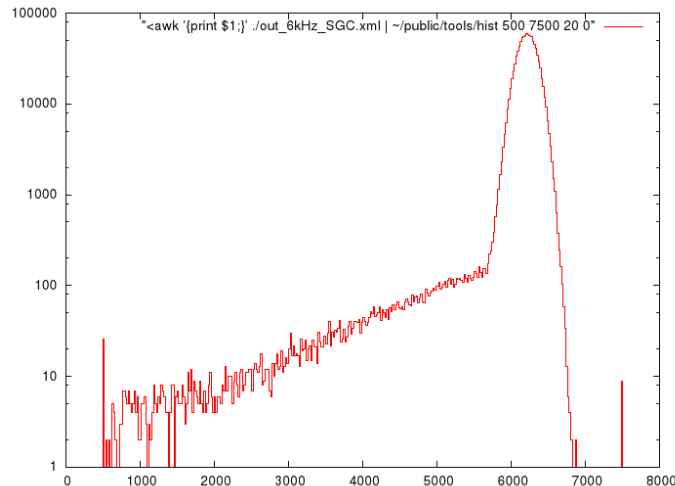


48. ábra SGC mérés oszcilloszkópon

A szkóp képen látható a CH1 sárga jel a trigger jel amely a saját oszcillátorunkból jön, a CH2 a detektor anód szálán megjelenő felerősített jel értéke, valamint a CH3 a Front End vezérlőjele, azaz a mérés kezdeti időpontja a megfelelő helyre állítva azaz a CH2 maximumához.

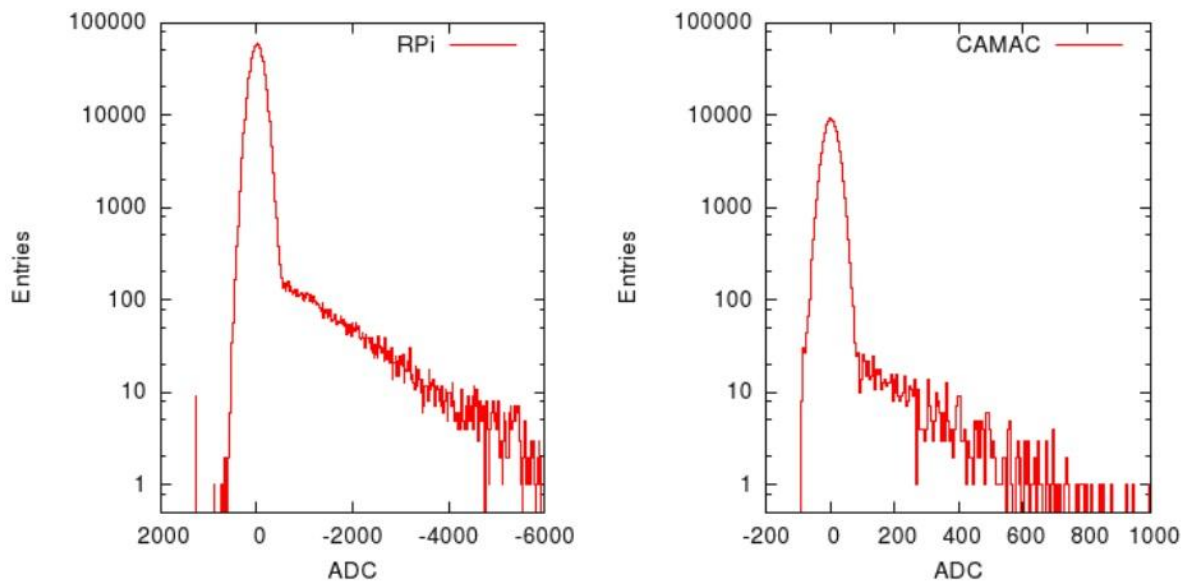
6.3 A mért adatok értékelése

6.3.1 SGC 1 millió esemény



49. ábra RPi SGC 1 millió esemény

Ez az eloszlás közvetlenül az RPi-vel mért adatokból készült az SGC mérés során, azaz a kiolvasott ADC értékek, ahol a nagy csúcs (ami a Gauss alakú zaj miatt jelenik meg) helye a konstans eltolás értéke, amit az ADC-vel hozzáadunk. Az eloszláson látszik, hogy a legtöbb esetben a fényimpulzus nem váltott ki egyetlen elektron kiütést sem, viszont sokszor megjelenik az elektron detektálását jelző lavina, melynek exponenciális az amplitúdó eloszlása. A mérést 23 másodperc alatt végezte el az RPi, amihez 130 kHz-es trigger jelet használtam, és mint azt a 49. ábra mutatja 1 millió eseményt rögzített.



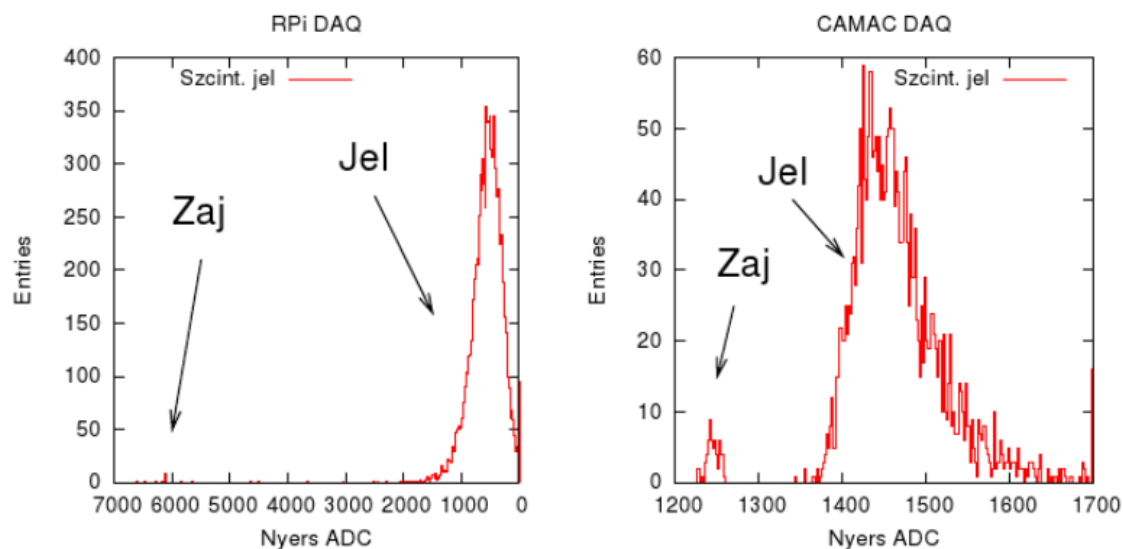
50. ábra RPi és CAMAC SGC mérési eredményei

Az 50. ábra bal oldalán az RPi által mért SGC mérés látszik. Ez megegyezik az 49. ábra adataival, csak nullára csökkentettem az eltolást, illetve előjelet cseréltem, hogy össze tudjam hasonlítani a CAMAC, 50. ábra jobb oldalán lévő hisztogram, által mért adatokkal. Mivel ez a mérés pontosan ugyanannak a detektornak (SGC), ugyanolyan beállításával készült, ezért látszik, hogy az RPi által mért eredmények reálisak, kompatibilisek a CAMAC által mértekkel. Tehát az RPi működése megfelelő! Összehasonlítás:

	Esemény (db)	Kiolvasási sebesség (Hz)	Méréshez szükséges idő (s)
CAMAC	100.000	300	333
RPi	1.000.000	43.000	23

Konklúzió: Az RPi által megvalósított DAQ tízszer több eseményt, körülbelül 15-ször gyorsabban olvas, mint a klasszikus CAMAC rendszer. Az is kiderült, hogy ezt még lehet gyorsítani(!), mivel késleltetéseket kellett beletenni a kiolvasó programba az ADC adattároló eszközeinek lassúsága miatt.

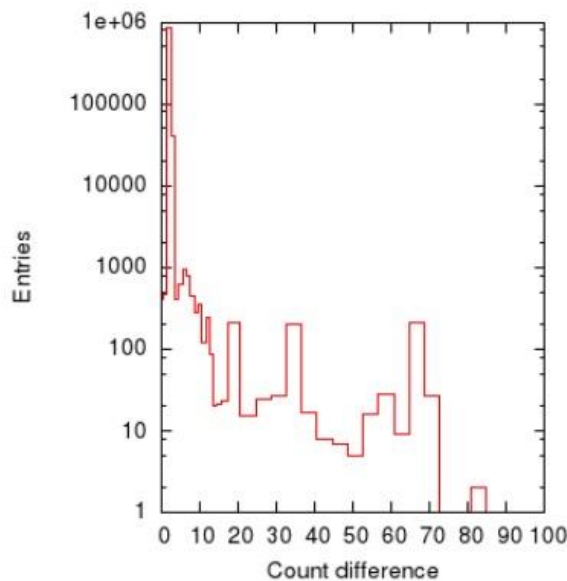
6.3.2 Szcintillátor



51. ábra RPi és CAMAC szcintillátor mérési eredményei

A szcintillátorral végzett mérés hasonló következtetésekre jogosít minket, mint a már tárgyalt SGC-vel felvett adatok. Ha nem ment át a béta-elektron a forráshoz közelebbi szcintillátoron, a detektor csak egy kis szórású „zaj”-nak megfelelő jelet ad. Ha átmegy a részecske, akkor a mért ADC érték ettől eltér, a detektor negatív feszültségű jele miatt a negatív ADC értékek felé (ezért van a skála fordított előjellel). Az eloszlás hasonlít a CAMAC-kal felvett adatsorra. Az RPi-vel megvalósított eszköz esetén a jel erőssége nagyobb volt az optimálisnál, ezért látható, hogy a 0-s ADC értéknél sok adat van – ez torzította az eloszlást (de így is konkluzív, nem került megismétlésre a mérés). Az is látható, hogy a CAMAC-kal végzett adatfelvételnél a zaj relatíve szélesebb, azaz az RPi-s megoldásnál a jel/zaj viszony kedvezőbb.

6.3.3 Időkülönbségek egyes mérések között



52. ábra RPi időkülönbségek mérése

A fentiekből látható volt, hogy körülbelül 100 kHz-es trigger frekvencia esetén nagyjából minden triggert mérni tud a rendszer. Ennek számszerűsítésére a következő mérést dolgoztam ki. Nagyon nagy frekvenciájú triggert adtam a bemenetre a 6.1.3 fejezetben tárgyalt összeállításban, és azt vizsgáltam, hogy hány darab triggert kap az eszköz két beolvasás között. Fontos, hogy ekkor az időnként előforduló nagyobb időkülönbségek is megfigyelhetővé válnak (ilyenek pl. memóriafrissítéskor vagy az operációs rendszer magasabb prioritású feladatainak végzésekor fordulhat elő, és személyi számítógépek DAQ-ként való alkalmazásánál gondot tudnak jelenteni).

321 kHz frekvenciás trigger jeleket generáltam jelgenerátorral és gyakorlatilag minden második triggert elkapta, tehát a beolvasási (esemény) frekvencia $321/2 = 160,5$ kHz. A mérés célja az volt, hogy milyen arányban találunk hosszú időkülönbséget két mérési pont között. Azt tapasztaltam, hogy ilyen nagyon ritkán fordul elő, 1 millió esetet vizsgálva néhány ezrelékében van 5 időegységnél (15 mikroszekundumnál) nagyobb holtidő. Időnként nagy sebességű olvasásnál előfordultak nem jól reprodukálható bithibák, és ez a problémakör vizsgálat alatt van.

6.4 A megvalósult mérési összeállítás értékelése

A feladat kiírásban szereplő kritériumoknak maximálisan megfelel az általam kifejlesztett DAQ rendszer, mivel egy alacsony fogyasztású, rugalmasan kezelhető

adatgyűjtő rendszert alkottam meg, amelynek költsége mindössze 35 \$. A DAQ rendszerem sokkal hatékonyabb megoldást nyújt, mint az eddig kifejlesztett, megvett rendszerek a laborban, így amellet, hogy a kitűzött feladatot sikeresen megoldottam, egy hasznos, univerzális, az eddigi rendszereknél jóval gyorsabb és bárki számára könnyen kezelhető megoldást sikerült kialakítani.

6.5 Továbbfejlesztési lehetőségek

A dolgozat eredményei sok részletkérdést hagytak nyitva, többek között azt, hogy miképpen érdemes úgy tervezni a front-endet, hogy az RPi által kínált teljesítményt optimálisan ki lehessen használni. Ha ezt a felhasználó nem akarja elérni, az RPi-vel megvalósított adatgyűjtő természetesen akkor is jól adaptálható. A jövőbeli tervek között szerepel, hogy a fejlesztés alatt lévő 5 detektor réteggel rendelkező müon-tomográf adatgyűjtését már a RPi féle megoldással történjen, illetve a laborban fejlesztés alatt lévő HPTD detektorrendszer elsődleges tesztjeihez az RPi-re épülő DAQ-ot használják.

Nagyobb detektorrendszereknél egyetlen RPi egység nem tudja teljesíteni a szükséges adatolvasási feladatot, de tekintve hogy az eszköz olcsó, megérheti akár több tíz egység párhuzamos üzemeltetése. Ez tehát laborfejlesztéseken túlmutató alternatívát is kínálhat.

A fentiek során nem tisztázódott az RPi-vel valóban elérhető maximális sebesség értéke, ami a regiszterek közvetlen, az operációs rendszer részben megkerülő – emiatt körültekintően kezelendő – címzésével várhatóan javítható.

7 Összefoglalás

A diploma írásom során lehetőségem nyílt a REGARD kutatócsoport (RMKI – ELTE Gaseous detector Research and Development) által végzett részecske kísérleteket megismerni. Alapvető tudást szereztem a nagyenergiás fizikai detektorrendszerekben végbemenő fizikai kölcsönhatásokról, folyamatokról és alaposabban megismerkedtem a detektorok működésével. A feladatom elvégzéséhez kétféle út vezetett, amelyek közül egy teljesen új megvalósítási lehetőséget választottam. A Raspberry PI mini PC-vel az alaptól indulva építettem fel egy komplett adatgyűjtő (DAQ – Data Acquisition) működéshez szükséges funkciókat.

Első lépésben megvalósítottam szoftveresen a trigger egységgel, illetve a front end elektronikával történő kommunikációt, majd az ehhez szükséges feszültség átalakítást is kifejlesztettem, amely később NYÁK formában legyártásra került.

Második lépésként a mért adatok könnyebb elérése, illetve a mérések elvégzéséhez szükséges programok indításának kezelését valósítottam meg webszerveren keresztül, hogy a világon bárhol el tudjuk érni a mérési eredményeket és el tudjuk indítani a mérést.

Harmadik lépésben az RPi megfelelő működését teszteltem szimulált körülmények között (mikrokontrollerekkel, saját trigger egységgel, számlálókkal), erre azért volt szükség, hogy megbizonyosodjak róla, hogy az elvárt működést tudja produkálni a rendszerem mielőtt valós detektoron mérést végzek vele. A tesztek jól sikerültek, ezért 2 detektorrendszeren végeztem méréseket a KFKI-ban. Az egyik egy szcintillátor, a másik pedig egy proporcionális detektor volt. A mért adatokat összevetve az ugyanezen a detektorokon korábban elvégzett klasszikus DAQ rendszerekkel történő méréssel megegyező eloszlást kaptam, de azoknál jóval több adatot tudtam, kevesebb idő alatt beolvasni. Az elért eredmények miatt sikeresnek ítélem meg az elvégzett munkát, mivel a kitűzött feladatot elértem és sikerült egy olyan új fejlesztésű DAQ rendszert megvalósítanom, amely univerzálisabb, gyorsabb, olcsóbb és rugalmasabban vezérelhetőbb, mint az eddigi rendszerek.

Külön kiemelném, hogy számomra egyéni sikert jelent, hogy a diplomamunkám során rengeteg terület működését kellett úgy összeillesztenem, szinkronizálnom, mint

például digitális technika (nagy sebességű kombinációs hálózatok konkrét kialakítása), speciális hardware-re optimalizált Linux operációs rendszer, GPIO periféria, mikrokontrollerek, webszerver üzemelése, feszültség átalakító legyártása NYÁK-on, részecskefizikai ismeretek, stb, hogy azok egy egységként az adott körülmények között a megfelelő működést produkálják. Továbbá az is fontos számomra, hogy a kifejlesztett rendszert valóban fel fogják használni a mérések során, így sikerült egy hatékony, olcsó, rugalmas és gyors rendszert megalkotnom. A megvalósított DAQ rendszer kiváló paraméterekkel rendelkezik, további lehetőségei nagyon széles körűek, reményeim szerint a közeljövőben nagyobb kaliberű mérésekhez is felhasználják majd.

8 Köszönetnyilvánítás

Köszönetet szeretnék mondani konzulensemnek, Varga Dezsőnek, aki szakmai tanácsaival, kreatív ötleteivel és végtelen türelmével hozzájárult a diplomamunkám elkészítéséhez, ellenőrzéséhez. Továbbá köszönettel tartozom barátaimnak, Juhász Sándornak, aki a programozás terén látott el hasznos tanácsokkal és jó szobatárshoz méltóan végigkíserte a munkámat, illetve Kovács Andrásnak a megvalósított NYÁK legyártása közben nyújtott segítségéért. Valamint szeretném megköszönni családomnak a tanulmányaim alatt nyújtott a töretlen támogatásukat.

Irodalomjegyzék

- [1] <http://www.rmki.kfki.hu/>
- [2] <http://regard.kfki.hu/>
- [3] Horváth Dezső: A Standard Modell: mi az, és mire jó?
- [4] Lederman, Teresi: Az isteni a-tom: Mi a kérdés, ha a válasz a világegyetem?
- [5] Dénes Ervin és Vesztergombi György: Magyar technikai export a CERN-be
- [6] <http://en.wikipedia.org/wiki/NIM>
(2012. november 28.)
- [7] http://en.wikipedia.org/wiki/Computer_Automated_Measurement_and_Control
(2012. november 28.)
- [8] G. Hamar, G. Kiss, D. Varga: Asymmetric Multiwire Proportional Chamber With Reduced Mechanical Tolerances. Nucl. Instrum. Meth. A648(2011) 163.
- [9] Oláh László: Föld alatti üregek vizsgálata kozmikus részecskék segítségével – 2010 (ELTE-TTK BSc szakdolgozat)
- [10] Barnaföldi G., Bencédi Gy., Hamar G., Melegh H., Oláh L., Surányi G., Varga D.: Kincskeresés kozmikus műonnal
- [11] N. Ellis: *Trigger and data acquisition*
- [12] Melegh Hunor: Műon-tomográf adatgyűjtő berendezésnek fejlesztése – 2011 (BME-VIK MSc Önálló laboratórium)
- [13] <http://www.raspberrypi.org/>
- [14] http://elinux.org/Rpi_Low-level_peripherals
- [15] <https://projects.drogon.net/raspberry-pi/wiringpi/>
- [16] http://wiki.mech.uwa.edu.au/upload/8/8e/BS170_pinout.gif
- [17] <http://pinout-circuits-images.dz863.com/432/LM2950.jpg>
- [18] Philips Application Note: Bi-directional level shifter for I²C-bus and other systems
- [19] <http://www.ti.com/graphics/tool/MSP-EXP30G2.jpg>
- [20] <http://www.kontel.iwebshop.hu/termek/thumbnails/74hct93-w.jpg>
- [21] <http://homesupport.cisco.com/en-us/support/adapters/WUSB54GC>

Ábrajegyzék

1. ábra Standard modell	11
2. ábra A nagyenergiás fizikai mérési folyamat sémája [5].....	13
3. ábra Mintapéllda: mérési elrendezés vázlata időmérés esetén [11].	15
4. ábra Lokális pufferek [11].	16
5. ábra Többszintű trigger [11].	17
6. ábra Többszintű trigger lokális pufferekkel [11].	17
7. ábra CCC elrendezés oldalnézet [8].....	19
8. ábra CCC elrendezés felülnézet [9]	20
9. ábra Általános mérési elrendezés.....	24
10. ábra Müon-tomográf CAMAC kiolvasása [9]	26
11. ábra REGARD fejlesztésű DAQ [12].....	28
12. ábra REGRAD fejlesztés trigger jel kiértékelése [12].....	28
13. ábra Megvalósított kártyák méretei [12].....	29
14. ábra Saját fejlesztési megvalósítás vázlati terve	32
15. ábra Raspberry Pi [13]	33
16. ábra RPi B modell [13]	33
17. ábra Putty program.....	39
18. ábra RPi GPIO BCM lábkiosztás [14].....	41
19. ábra RPi GPIO wiringPi lábkiosztás [15]	42
20. ábra Shell script által generált CLOCK	45
21. ábra Python által generált CLOCK.....	46
22. ábra C nyelven által generált CLOCK.....	48
23. ábra 3.3 V – 5 V feszültség átalakító breadboardon	50
24. ábra BS170-es MOS-FET lábkiosztás [16]	51
25. ábra BS170 lábakra csatlakoztatott jelek	51
26. ábra LM2905 feszültség regulátor lábkiosztása [17].....	52
27. ábra 3.3 V – 5 V feszültség átalakító megvalósítása breadboardon	52
28. ábra Mentor Graphics PADS Logic-kal a sematikus rajz elkészítése	53
29. ábra Fizikai layout	54
30. ábra Az elkészült feszültség átalakító NYÁK-on	55
31. ábra Webszerver egyszerű vázlata.....	56

32. ábra MSP430 Launchpad [19]	58
33. ábra Adatok kiolvasása mikrokontrollerről	59
34. ábra Saját előállítású triggeres rendszer vázlata	63
35. ábra Saját előállítású triggeres rendszer megvalósítása.....	64
36. ábra Saját előállítású trigger rendszer	65
37. ábra Saját előállítású trigger rendszer jelei, CH1: trigger, CH2 CLOCK, CH3 LAM, CH4 /PLbar	65
38. ábra 74HCT93E 4 bites számláló lábkiosztása [20]	66
39. ábra Trigger számlálós mérési elrendezés	67
40. ábra Trigger számlálós mérés jelei oszcilloszkópon CH1: trigger, CH2 LAM, CH3 CLOCK, CH4 adatok	67
41. ábra RPi UART-on való vezérlésének szimulációja.....	69
42. ábra REGARD front-end	70
43. ábra REGARD front-end mérési elrendezése	72
44. ábra REGARD front-end mérése detektor nélkül.....	72
45. ábra Szcintillátor, illetve az ^{90}Sr	73
46. ábra Szcintillátoros mérés oszcilloszkópon	74
47. ábra SGC detektor.....	75
48. ábra SGC mérés oszcilloszkópon	75
49. ábra RPi SGC 1 millió esemény	76
50. ábra RPi és CAMAC SGC mérési eredményei	77
51. ábra RPi és CAMAC szcintillátor mérési eredményei	78
52. ábra RPi időkülönbségek mérése	79

Függelék

M430G2553 mikrokontrollerrel 8 bites számláló megvalósítása:

```
#include <msp430.h>

unsigned int i;
int main(void)
{
    //P1.0 clock feedback led
    //P1.3 manual clock switch
    //P1.4-5 and P2.0-5 is the 8 output
    //P1.6 clock input

    WDTCTL = WDTPW + WDTHOLD;    // Stop watchdog timer
    P1DIR |= 0x31;                // Set P1.0 and P1.4 and P1.5 to output
direction
    P1DIR &= ~0x08;               // Set P1.3 to input direction
    P1DIR &= ~0x40;               // Set P1.6 to input direction
    P2DIR |= 0x3F;                // Set P2.0..5 to output direction

    P1IE  |= 0x08;                // P1.3 interrupt enabled
    P1IES  |= 0x08;                // P1.3 hi/lo edge
    P1IFG &= ~0x08;               // P1.3 IFG cleared
    P1REN  |= 0x08;                // P1.3 pullup

    P1IE  |= 0x40;                // P1.6 interrupt enabled
    P1IES &= ~0x40;                // P1.6 lo/hi edge
    P1IFG &= ~0x40;                // P1.6 IFG cleared

    //setting default state

    P1OUT |= (1<<0);

    P1OUT &= ~(0x30);
    P2OUT &= ~(0x3F);

    i = 0;

    __enable_interrupt();

    for (;;)
    {
        //infinite loop
    }
}
// Port 1 interrupt service routine
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    P1OUT ^= 0x01; //toggle the led
    i++;          //increment the counter

    //set the output
```

```

P1OUT |= (i << 4) & 0x30;
P1OUT &= (i << 4) | ~0x30;
P2OUT = i >> 2;
//clear interrupt
P1IFG &= ~0x08;           // P1.3 IFG cleared
P1IFG &= ~0x40;           // P1.6 IFG cleared
}

```

M430G2553 mikrokontrollerrel az UART-os vezérlés megvalósítása:

```

#include "msp430g2553.h"
unsigned char i;
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;           // Stop WDT
    BCSCTL1 = CALBC1_1MHZ;              // Set DCO
    DCOCTL = CALDCO_1MHZ;
    P1DIR = 0xFF;                       // All P1.x outputs
    P1OUT = 0;                           // All P1.x reset
    P1SEL = BIT1 + BIT2 + BIT4;          // P1.1 = RXD, P1.2=TXD
    P1SEL2 = BIT1 + BIT2;                // P1.4 = SMCLK, others GPIO

    //set button on P1.3
    P1DIR &= ~0x08;                      // Set P1.3 to input direction
    P1OUT |= 0x08;                       // P1.3 set
    P1IE |= 0x08;                        // P1.3 interrupt enabled
    P1IES |= 0x08;                       // P1.3 hi/lo edge
    P1IFG &= ~0x08;                      // P1.3 IFG cleared
    P1REN |= 0x08;                       // P1.3 pullup

    P2DIR = 0xFF;                        // All P2.x outputs
    P2OUT = 0;                           // All P2.x reset

    P3DIR = 0xFF;                        // All P3.x outputs
    P3OUT = 0;                           // All P3.x reset

    UCA0CTL1 |= UCSSEL_2;                // SMCLK
    UCA0BR0 = 8;                         // 1MHz 115200
    UCA0BR1 = 0;                         // 1MHz 115200
    UCA0MCTL = UCBRS2 + UCBRS0;          // Modulation UCBRSx = 5
    UCA0CTL1 &= ~UCSWRST;                // **Initialize USCI state
machine**
    IE2 |= UCA0RXIE;                    // Enable USCI_A0 RX interrupt
    __enable_interrupt();
    for (;;)
    {
        //infinte loop
    }
}
// Echo back RXed character, confirm TX buffer is ready first
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
    while (!(IFG2&UCA0TXIFG));           // USCI_A0 TX buffer ready?
    UCA0TXBUF = UCA0RXBUF;               // TX -> RXed character
    i=UCA0RXBUF;
    P2OUT = i;
    P1OUT |= (i & 0xC0);
    P1OUT &= (i | 0x3F);
}

```

```

    P1OUT ^= 0x01;
}
// Port 1 interrupt service routine
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    P1OUT ^= 0x01;    //toggle led
    P1IFG &= ~0x08;   // P1.3 IFG cleared
}

```

index.php:

```
<?php
```

```
$dir = '/home/pi/test';
```

```

ini_set('display_errors', 1);
error_reporting(-1);

```

```

if (isset($_GET['start'])) {
    start($_GET['start']);
} else if (isset($_GET['kill'])) {
    kill( (int)$_GET['kill']);
} else {
    listing($dir);
}

```

```

function start($path) {
    echo "Starting $path...";
    $r = system("/sbin/start-stop-daemon --start --background --exec
$path");

    echo '<a href="?">... OK!</a>';
}

```

```

function kill($pid) {
    system("kill $pid");
    header('Location: ' . $_SERVER['PHP_SELF']);
}

```

```

function listing($dir) {
    $dir = realpath($dir);

    echo "Executable files in $dir:";

    $d = new DirectoryIterator($dir);
    $execs = array();
    foreach($d as $f) {
        if ($f->isFile() and $f->isExecutable()) {
            $execs[] = $dir . '/' . $f->getFilename();
        }
    }

    echo '<ul>';
    foreach ($execs as $e) {

```

```

    echo '<li><a href="?start=' . $e . '">. $e . '</a></li>';
}
echo '</ul>';

echo 'Currently running programs:';

$x = strlen($dir);
$d = new DirectoryIterator('/proc');
echo '<ul>';
foreach(glob("/proc/*/exe") as $proc) {
    $exe = readlink($proc);
    if (substr($exe, 0, $x) === $dir) {
        $pid = substr($proc, 6, strlen($proc) - 6 - 4);
        echo '<li>' . $exe . ' -- <a href="?kill=' . $pid . '"
style="color: red;">EXTERMINATE DESTROY KILL</a></li>';
    }
}
echo '</ul>';
}

```

adat kiolvasás:

```

#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <wiringSerial.h>
#define DATA_SIZE 16
#define MAX 1000000
#define CLKPIN 14
#define BUSYPIN 11
#define LAMPIN 12

int main() {

    unsigned int diff[MAX];
    unsigned int data[MAX];
    int lam_cnt = 0;
    int last_data = 0;

    if (wiringPiSetup () == -1)
        exit (1) ;

    //CLK pin
    pinMode(CLKPIN, OUTPUT);

    //input pins
    pinMode(0, INPUT);
    pinMode(1, INPUT);
    pinMode(2, INPUT);
    pinMode(3, INPUT);
    pinMode(4, INPUT);
    pinMode(5, INPUT);
    pinMode(6, INPUT);
    pinMode(7, INPUT);
}

```

```

//Busy jel
pinMode(BUSYPIN, OUTPUT);

//LAM jel
pinMode(LAMPIN, INPUT);

printf("Starting\n");

digitalWrite(CLKPIN, 0);

while(lam_cnt < MAX)
{
    printf("Waiting for LAM signal\n");

    //Busy set 0
    digitalWrite(BUSYPIN, 0);

    while(!digitalRead(LAMPIN)){

        //Busy set 1
        digitalWrite(BUSYPIN, 1);

        int k;
        for(k = 0; k < 5; k++)
        {
            digitalWrite(BUSYPIN, 1);
        }

        int j;
        unsigned int one_data = 0;

        for(j = 0; j < DATA_SIZE; j++)
        {

            one_data += digitalRead(0)<<(DATA_SIZE-j-1);
            digitalWrite(CLKPIN, 1);
            digitalRead(1);
            digitalRead(2);
            digitalRead(3);

            digitalWrite(CLKPIN, 0);
            digitalRead(4);
            digitalRead(5);
            digitalRead(6);
            digitalRead(7);
        }

        data[lam_cnt]=one_data;
        diff[lam_cnt]=one_data-last_data;
        if (lam_cnt>0)
            printf("%d: %d - diff: %d\n", lam_cnt, one_data, one_data-
last_data);
        else
            printf("%d: %d\n", lam_cnt, one_data);

        last_data = one_data;
        lam_cnt++;

        //UART control

```

```

        if(lam_cnt > 10 && lam_cnt < 12)
            system("sudo ./uart 255");

    }

    printf("Writing to file...\n");
    FILE *f1;
    f1 = fopen ("/var/www/out.xml", "w");
    int i;
    fprintf(f1,"<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\n");
    fprintf(f1,"<?xml-stylesheet href=\"bambula.xsl\"?>\n" type="text/xsl\"
    href="\"bambula.xsl\"?>\n");
    fprintf(f1,"<pi>");
    for(i = 0; i < MAX; i++)
    {

        fprintf(f1,"<measure><data>%d</data><diff>%d</diff></measure>\n",data
        [i],diff[i]);
    }
    fprintf(f1,"</pi>");
    fclose (f1);
    printf("File created\n");

    return 0 ;
}

```

UART vezérles RPi-vel:

```

#include <wiringSerial.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int handle = serialOpen ("/dev/ttyAMA0", 115200);
    printf("input chars: %s\n",argv[1]);
    serialPutchar(handle,atoi(argv[1]));
    serialClose(handle);
}

```

xslt:

```

<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <html>
    <body>
    <h2>My data</h2>
    <table border="1">
        <tr bgcolor="#9acd32">
            <th>Data</th>
            <th>Diff</th>
        </tr>
        <xsl:for-each select="pi/measure">
            <tr>
                <td><xsl:value-of select="./data"/></td>
                <td><xsl:value-of select="./diff"/></td>
            </tr>
        </xsl:for-each>
    </table>
    </body>
    </html>
</template>

```

```
        </tr>
      </xsl:for-each>
    </table>
  </body>
</html>
</xsl:template>
```

```
</xsl:stylesheet
```