Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Mérőeszköz IEEE 1588 és SyncE megoldások alkalmazási környezetben történő validációjára

Önálló laboratórium zárójegyzőkönyv
2015/16. II. félév

Badó Dávid

III. évf, villamosmérnök szakos hallgató
BSc Beágyazott információs rendszerek ágazat

Konzulens:

Kovácsházy Tamás
(Méréstechnika és Információs Rendszerek Tanszék)

Feladatkiírás:

Az IEEE 1588 óraszinkronizáció és a SyncE órajel elosztási megoldások lehetővé teszik, hogy elosztott beágyazott rendszerekben, beleértve az Internet of Things (IoT) megoldásokat is, nagy pontossággal azonos idő és órajel szerint járjanak a rendszer komponensei. Ez óriási előny, mert ez jelentősen egyszerűsíti az ilyen rendszerek rendszerarchitektúrát, például lehetségessé válik szinkronizált mintavétel és beavatkozás kiterjedt rendszerekben a kommunikációra használt hálózat felhasználásával (korábban erre nagyon drága dedikált hálózatokat és technológiáat, pl. IRIG használtak, ami sok alkalmazásban elérhetetlen volt annak ára és komplexitása miatt). A tanszéken korábbi fejlesztések során kidolgozásra kerültek ilyen megoldások, amelyek pl. Linux alatt szabadon hozzáférhetők, és 100 ns alatti pontossággal teszik lehetővé az elosztott rendszer óráinak szinkronizálását. Ugyanakkor az is kiderült a fejlesztések során, hogy az ilyen rendszerek tesztelése és validációja nem megoldott, arra speciális műszereket kell kifejleszteni.

Ennek megfelelően kidolgozásra kerültek olyan mérési eljárások, amelyek egyik oldalról figyelembe veszik az alkalmazások kötöttségeit, másik oldalról képesek a szükséges pontosságot nyújtani, várhatóan alacsony ár és egyszerű használhatóság mellett. Ezek mérési eljárások deszkamodell jellegű prototípusok felhasználásával kipróbálásra kerültek, és alkalmasnak tűnnek a felvetett feladat megoldásra. Vagyis a konkrét mérőműszer prototípusok kifejlesztése a következő feladat.

A témán dolgozó hallgatók feladata a mérőrendszer prototípusának egyes hardware és/vagy szoftver komponenseinek részletes specifikációjának kidolgozása, elkészítése és megvalósítása. A hardware tartalmaz(hat) nagyteljesítményű mikrovezérlőket (pl. a TI EK-TM4C1294XL Connected LaunchPad prototípus kártya felhasználásával), esetleg egykártyás számítógépet (pl. Raspberry PI), FPGA-át (idő és frekvencia mérés, stb.), stb. A szoftver komponensek részben a mikrovezérlőkön, részben Linux-on futhatnak (ARM vagy x86/amd64 platform).

A feladat több közösen dolgozó hallgatónak is kiadható (preferált konstrukció), vagy akár egymással nem kapcsolódó feladatok is kijelölhetőek. A témák folytathatók szakdolgozatként (BSc), vagy diplomatervként (MSc képzés), valamint TDK dolgozat készítésére is van lehetőség.

Kulcsszavak: Óraszinkronizáció, HW támogatott IEEE 1588 óraszinkronizáció, Ethernet és TCP/IP kommunikációs beágyazott rendszerekben, mikrovezélő.

Tartalomjegyzék

1. Bevezetés

1.1. Feladat értelmezése

A feladatkiírásból nagyvonalakban látható, hogy egy mérőrendszer megtervezése és előállítása a feladat. Az viszont nem teljesen világos, hogy mit és hogyan mérjen ez a mérőműszer, és hogy ezt milyen pontossággal, milyen tartományban tegye.

A mérőrendszer alapvetően az IEEE1588-cal szinkronizált eszközök által küldött PPS jelek közötti idő mérésére szükséges létrehozni. Ezen óraszinkronizásciós eljárások 100 ns alatti pontossággal teszik lehetővé az elosztott rendszer óráinak szinkronizálását, ezért a mérőműszerünk pontossága és felbontása 1-10 ns közötti tartományba kell, hogy essen. További elvárás, hogy több jelet is tudjon mérni, ezért konfigurálhatónak kell lennie, hogy mely jelek között mérjen.

A mérési eredmény semmit sem ér, hogy ha ez nincs valahol megjelenítve. Mivel alapvetően ennél a feladatnál elvárás, hogy a mérési eredmények folyamatosan elérhetőek és megállapíthatóak legyenek, ezért elvárás, hogy a műszer konfigurálása és a mérési eredmények elérése egy WEB-es felületen elérhető legyen.

1.2. Lehetséges megoldások

Ennek a feladatnak a megoldását hardware komponensek és software komponensek szempontjából is szemügyre kell venni.

Hardware komponensek:

- nagyteljesítményű mikrovezérlő
- egykártyás számítógépet (pl. Raspberry PI)
- FPGA

Software komponensek:

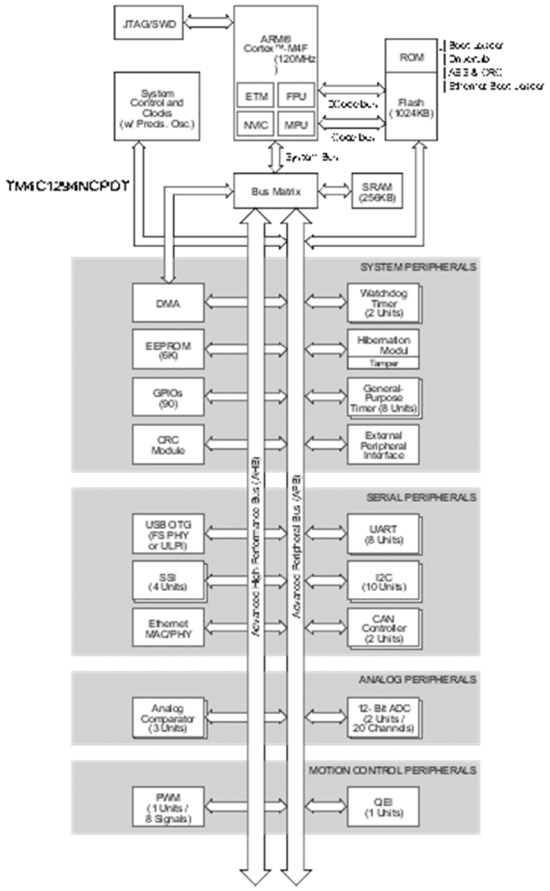
- Mikrovezérlőn futó
- Linux-on futó (ARM vagy x86/amd64 platform)

Az én választásom egy nagyteljesítményű mikrovezérlőre (Texas Instruments EK-TM4C1294XL Connected LaunchPad) és mikrokontrolleren futó software komponensre esett, amelyet a Code Composare Studio fejlesztőkörnyezet segítségével C nyelven hozok létre.

2. Ismerkedés

2.1. A mikrovezérlő és a fejlesztő környezet

A feladat megvalósításához Texas Instruments egyik fejlesztőkártyáját használtam, amely az EK-TM4C1294XL Connected LaunchPad névre hallgat. Ez a fejlesztőkártya egy nagyteljesítményű mikrovezérlőt (TM4C1294NCPDT Mikrokontroller) tartalmaz. Hardweres támogatottságát tekintve eléggé kitesz magáért ugyanis tartalmaz egy 32-bites ARM Cortex-M4 CPU-t , 1MB Flash, 256KB SRAM és 6KB EEPROM memóriákat továbbá Ethernet interfészt, 8 darab 32-bit-es timereket, dual 12-bit ADC-t, soros kommunikációs portokat (UART, I2C, CAN) stb.



1. ábra Tiva™ TM4C1294NCPDT Microcontroller Funcional Diagramm [1]

A fejlesztőkártya a mikrokontrolleren kívül még tartalmaz többek között Ethernet portot, 4 darab ledet, kettő darab gombot és két USB portot, amelyekből egyet programozásra fogunk használni.

Sokféle fejlesztő környezet közül a Code Composer Studiot (későbbiekben CSS) használtam. Ez egy Eclipse alapú fejlesztőkörnyezet, amely a Texas Instruments C/C++ compilerét használja. A házon belüli compailer illetve az a tény, hogy megvásárlás nélkül is elég jó lehetőségeket biztosít, ezért jutott a választásom erre a programra.

Szerencsére a mikrovezérlő bőséges dokumentációval [1] és szoftveres támogatottsággal rendelkezik, így csak a megfelelő dokumentumot kellett megnyitnom, hogy megtaláljam azt, amit kerestem. A CSS-hez letölthető a mikrokontroller softver package-e, ami sokféle mintaprogrammot tartalmaz. Ezen kívül a driverekhez már megírt alkalmazások leírását a Peripheral Driver Library tartalmazza [2].

A mintaprogramok és e dokumentum segítségével már belemélyedhettem a beágyazott C programozás rejtelmeibe. Viszont a soros kommunikációs port olvasásához még nem rendelkeztem programmal. Annak ellenére, hogy nem ismertem erre a Puttyot választottam. Ez jó döntésnek bizonyult, mert működését gyorsan megértettem és a használat során is ritkán volt vele gond.

2.2. Mintaprogramok működésének megismerése

Miután telepítettem a CSS-t és hozzáadtam a szükséges szoftver csomagot kipróbáltam néhány mintaprogramot.

2.2.1. blinky

A blinkyvel kezdtem. Ez egy nagyon primitív program. A második led villogtatását végzi. Ehhez két féle drivert használ: a General-Purpose Inputs/Outputs (GPIO) és System Controll (SysCtl) modulokat. A N0-ás port kimenetbe állításával vezéreli a második ledet. A ledek villogtatása szoftveresen van időzítve. Ez nagyon primitív és nem használatos megoldás. A későbbiekben az ilyeneket az időzítő áramkörök (timerek) konfigurálásával fogjuk megoldani.

2.2.2. hello

Ezután megnéztem a hello-t. Ez már egy fokkal jobb mintaprogram, mint a blinky. Az első led villogtatása mellett egyszer kiírja az UART-ra, hogy „Hello, world!” Ehhez több drivert is használ: GPIO, SysCtl és UART. Az N1-es porton vezérli a D1-es ledet. Az UART Rx és Tx vonalait az A0 és A1-es portra kapcsolja. Viszont ez a mintaprogram is még primitívnek mondható, hiszen az időzítés itt sem a timerek segítségével történik. Érdekesség, hogy ebben a programban a rendszerórajel 120Mhz-en működik, ami a mikrokontroller által elérhető maximális órajel sebesség. Ezt a sebességet a PLL segítségével állítja be.

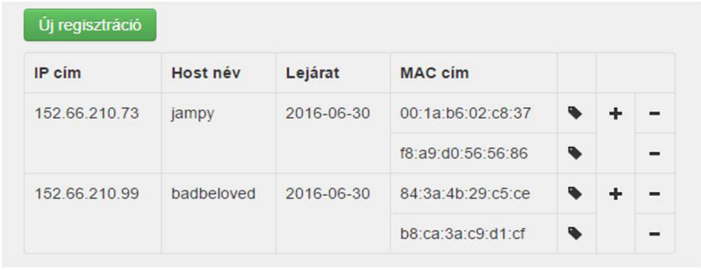
Ezután az Ethernetet használó enet_io, enet_lwip és enet_uip mintaprogramokat fedeztem fel. Egyből az enet_io felprogramozásánál és megismerésénél rájöttem, hogy ahhoz hogy lássam a program lényegi működését csatlakoztatnom kell a fejlesztő kártyát az internethez. Viszont az internet elérésében akadályokba ütköztem.

2.3. Csatlakozás a hálózatra, avagy hogyan kapok IP címet

Több-féle megoldás is van arra, hogy miképp tudjuk csatlakoztatni az eszközt a hálózathoz és elérni, hogy kapjon IP címet:

- Modemhez vagy routerhez való csatlakoztatása Ethernet kábellel
- laptophoz való csatlakoztatása amely wifin keresztül éri el az internetet és DHCP szerverként működve kioszt egy IP címet

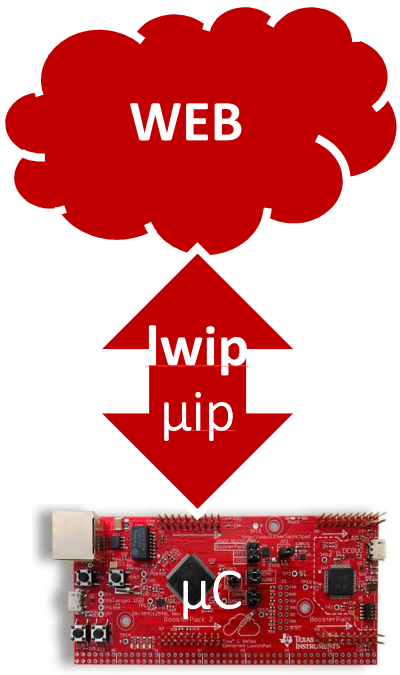
A munkát ebben a témában otthon, a tanszéken, de főleg a kollégiumban végeztem. Otthon nem volt nehéz elérnem, hogy IP címet kapjon a mikrovezérlő, mert az otthoni hálózat egy router segítségével működik, amihez Ethernet kábellel hozzá tudom csatlakoztatni az eszközt. Az egyetlen hátrány, hogy újracsatlakozáskor mindig új IP címet kap. Viszont a munka nagy részét a kollégiumban végeztem, ezért fontosabb volt, hogy az egyetemi hálózathoz hogyan tudom csatlakoztatni az eszközt. Ezzel egy kissé rosszul jártam ugyanis az egyetemi hálózatról csak úgy nem lehet IP címet kapni. Hallgatónként két-két eszközzel lehet elérni az egyetemi épületeken belül az internetet méghozzá úgy, hogy a megfelelő helyen megadjuk készülékünk MAC címét. Ez több okból is nem volt járható út. Egyrészről be kellett áldoznom az egyik eszközöm internet hozzáférését (laptop vagy okostelefon) másrészt hiába írtam be a kártya MAC címét így sem mindig sikerült megfelelő IP címet kapnom. Ha volt vezetékes hozzáférés az adott teremben (pl.: Önálló labor IE320) akkor működött, viszont ha nem akkor hiába csatlakoztattam hozzá a laptopomhoz Ethernet kábellel és osztottam meg az internetet a laptop Ethernet kártyájával a mikrokontroller sajnos nem kapott olyan IP címet, ami elérhető az internetről. A kollégiumi hálózat úgy működik, hogy minden egyes hálózati regisztrációhoz tartozik egy előre kiosztott IP cím. Ez alá több MAC címet is be lehet írni, de egyszerre csak egy eszközzel lehet csatlakozni a hálózatra. Az én esetemben ez nem a legjobb hiszen a fejlesztés során a laptopomnak és a kártyának egyszerre van szüksége IP címre. Szerencsére idén két hálózati regisztrációt is csináltattam, így ezzel az adott MAC cím beírásával könnyen megoldottam a problémát. A vezetékes internet hozzáféréssel sem volt gond, mert az egyetemmel ellentétben, a kollégiumban a hálózathoz való hozzáférés nagyrészt vezetékesen történik. Ebben az esetben még előny is volt ez a rendszer, mert itt mindig ugyanazt az IP címet kapta az eszköz.



2. ábra kollégiumi IP cím kiosztás

2.4. IP stack megválasztása

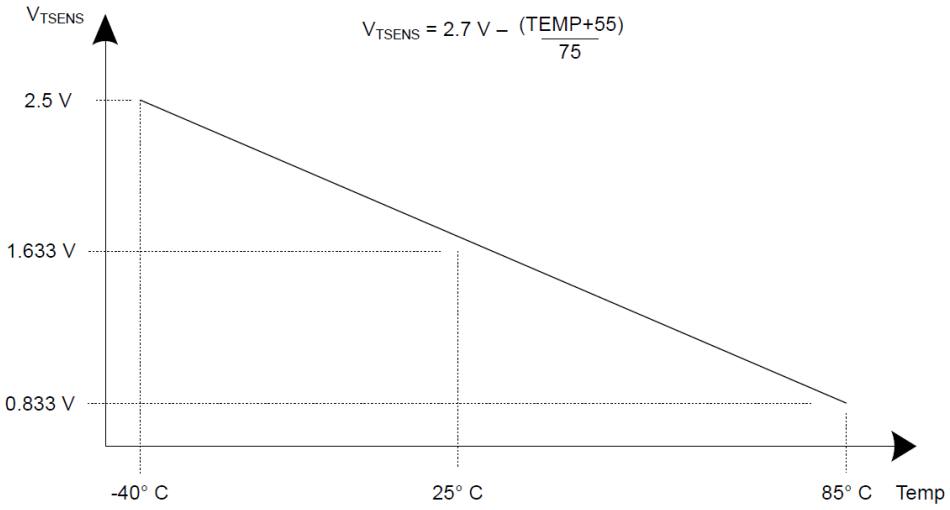
Miután sikerült csatlakoztatnom az eszközt a hálózathoz Ethernet kábel segítségével visszatértem a mintaprogramokhoz. Alapvetően az Ethernetet használó mintaprogramok működését tanulmányoztam át, hiszen a későbbiekben a mért adatokat a weben kell megjelenítenem. Abban láttam különbséget a mintaprogramok között, hogy milyen TCP/IP implementációval oldják meg a hálózati kommunikációt. Az enet_io, enet_lwip és az enet_weather lwip IP stacket használnak az enet_uip meg µip stacket használ. A két IP stack alapvetően a támogatott alapprotokollokban és a szükséges memóriaterületekben különbözik egymástól. Az lwip nem használható 8 bites processzorokon viszont sokkal több, szélesebb alkalmazási réteg szolgáltatást támogat mint a µip. A µip kevesebb memóriát igényel és kisebb teljesítményű, mint az lwip. Továbbá a µip kimondottan 8 bites processzorokra lett tervezve. Ezekből a tulajdonságokból nem igazán tudtam eldönteni, hogy melyiket alkalmazzam a későbbiekben, így alapvetően a mintaprogramok alapján döntöttem. Az µip-t használó mintaprogram nem volt annyira személetes, mint az lwip-t használó mintaprogramok. Az enet_io működésének megértése késztetett arra, hogy a későbbiekben az lwip IP stack segítségével jelenítsem meg a mért adatokat a weben. Az enet_io ezen kívül még két web kontroll módszer működését is bemutatta számomra. Ennél a pontnál viszont úgy gondoltam, hogy elkezdek foglalkozni azzal, hogy ténylegesen hogyan érjem el, hogy jeleket tudjak mérni a mikrovezérlő segítségével és amikor már rendelkezésemre állnak a mérési eredmények majd visszatérek ehhez a témához.



3. ábra mikrokontroller csatlakozása a WEB-re

2.5. Hőmérséklet mérése és kiírása

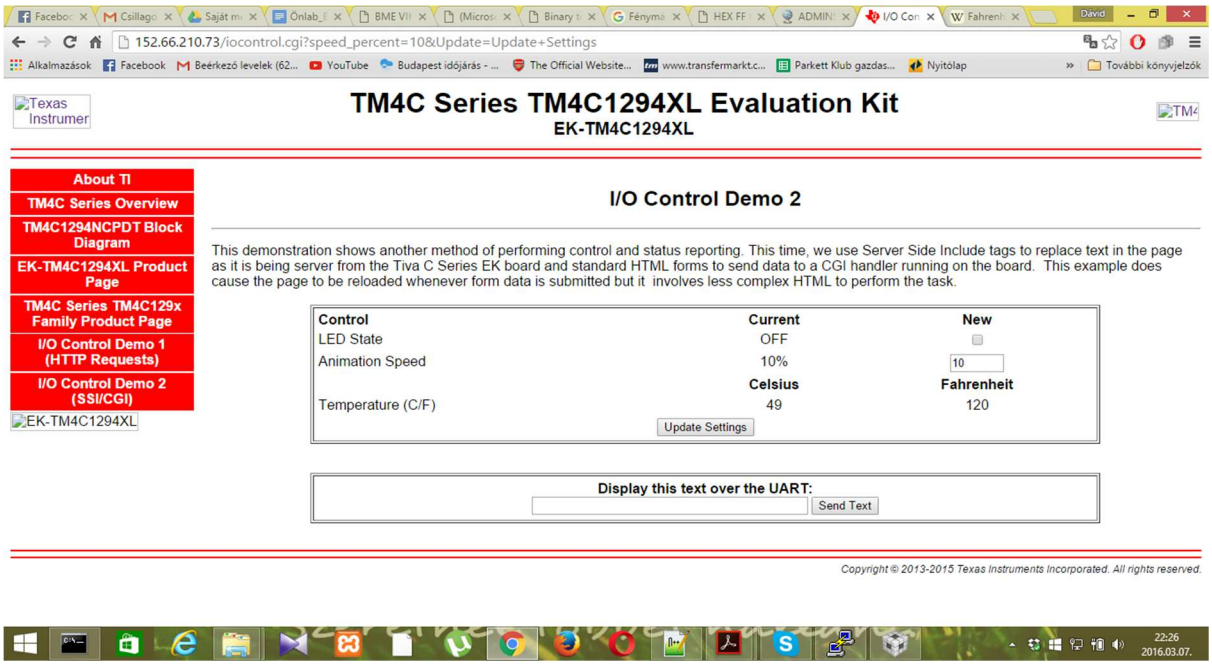
Mivel még semmilyen tapasztalatom nem volt a mikrovezérlőre való alkalmazásfejlesztésben azért következő lépésként elértem, hogy a nyomógombokkal tudjam vezérelni, hogy mely ledek világítsanak. Egy útmutató [3] segítségével módosítottam a hello mintaprogramot, hogy elérjem ezt a működést. Megtapasztaltam, hogy a CSS-ben hogyan lehet kezelni a projecteket és a hozzá tartozó forrásfájlokat.



4. ábra Internal Temperature Sensor Characteristic [1]

Ezután egy összetettebb dologba fogtam. Megvalósítottam, hogy egy belső hőmérséklet szenzor segítségével UART-on majd a weblapon is meg tudjam jeleníteni a mikrokontroller belső hőmérsékeltét. A szenzor értékét az ADC segítségével tudjuk meghatározni ami a grafikonon (4. ábra) látszik. A megvalósításban nagy segítséget nyújtott a driver library, ahol az ADC API-k és a programming example [2] segítségével könnyen meg tudtam valósítani, hogy egy változóba beolvassam a hőmérséklet értéket. Ezen kívül még az sq_iot példaprogram is hasznos volt számomra, mert ebben már implementálva volt egy függvény, ami elvégezte az adatok beolvasását a szenzorból, így ellenőrizni tudtam, hogy minden szükséges lépést megcsináltam. Az UART-ra való kiírás a már módosított hello mintaprogramban implementáltam. A weblapra való kiírását is egy mintaprogram módosításával végeztem el. Az enet_io-ban létrehoztam egy függvényt, amely kiolvassa a hőmérséklet értékét egy változóban. A honlapra való kiíratást az egyik webkontroll módszerrel végeztem el, amely SSI/CGI handlerekkel oldja meg, hogy kikerüljenek a megfelelő dolgok a weblapra. Azért ezt választottam a másik helyett, mert első megértésre ennek a megvalósítását tudtam kevesebb utánajárással kivitelezni. Az eddig megvalósítottoknak megfelelően két féle SSI taget hoztam létre. Az egyik segítségével a hőmérsékletet Celsiusban, a másik segítségével Fahrenheitben írja ki a hőmérsékletet. Ezt úgy teszi meg, hogy amikor frissítsük a weblapot vagy, ebben az

esetben lenyomjuk a gombot, akkor megkeresi a weblapon a tageket és behelyettesíti helyükre az adott értékeket. Ennek a megvalósulása a lenti képernyőképen (5. ábra) is látható.



5. ábra Belső hőmérséklet megjelenítését megvalósító weblap

3. Jelgenerálás és időmérés

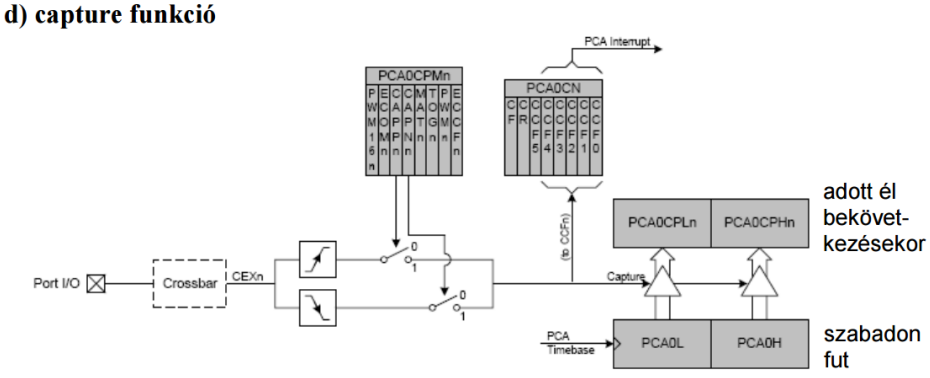
Az ismerkedés után belevettettem magam a tényleges feladat megoldásába. Ezen a ponton szerettem volna elérni, hogy a mikrovezérlő képes legyen pwm jelek generálására és e jelek fel- és/vagy lefutó élei közötti idő mérése, amelyet kiír az UART-ra. Ezen kívül meg akartam állapítani, hogy elő tudom-e állítani az előírt pontosságot.

3.1. Timerek funkciói

Mivel a feladat két vagy több jel időeltéréseinek mérése, ezért valószínűleg nem lőhetünk nagyon mellé, hogy ha ezt a feladatot az időzítő egységekkel akarjuk megoldani. Mint ahogyan azt már írtam a mikrokontroller 8 db 32 bites timerrel rendelkezik. Ezek a timerek többféle üzemmódban használhatóak. Ezek közül 3 üzemmódnak a működése után néztem utána jobban [4] [5].

- capture: amikor bekövetkezik a beprogramozott esemény, akkor a számláló értékét elmenti egy átmeneti tárolóban
- compare: a számláló folyamatosan növekszik és amikor ez az érték megegyezik az átmeneti tárolóban lévő értékkel, akkor megszakítást generál
- pwm: jelet generál aszerint, hogy az órajel által növelt számláló értéke megegyezik a tárolt értékkel

A mikrokontroller ezeket a módokat a timerek CCP pinjeivel lehet megvalósítani. A mikrovezérlőnek 12 darab 16/32 bites CCP pinje van. A továbbiakban a capture és a compare funkciókat fogom felhasználni.



6. ábra capture funkció bemutatása [4]

3.2. PWM jel generálása

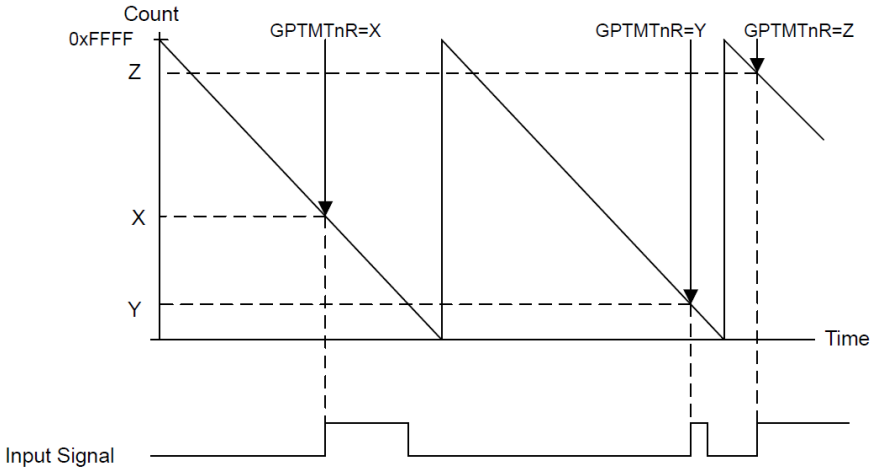
Ahhoz, hogy tudjunk mérni a jelek közötti késleltetést szükséges generálunk néhány jelet, amin ezt tudjuk mérni. Mivel külsőleg hozzácsatlakoztatni jeleket a kártyához még eléggé nehézkes, ezért a CCP funkciót használva pwm jelet generálok a mikrovezérlővel és majd

később a generált jelek fel- és lefutó élei közötti időeltérés alapján fogom kipróbálni, hogy mennyire jól működik az időmérés.

A megoldás során több helyről is segítséget kaptam. Elsősorban a Timer API-kat illetve programming example-t [2] vettem alapul. Ezeken kívül még a mikrokontroller software packegében találtam egy mintakódot, ami generál egy 66%-os pwm jelet, így ezekkel már el tudtam kezdeni az implementálást. Első nekifutásra nem sikerült a pwm jel generálása, a fordító hibát írt ki. A hiba megoldását a CCP pin konfigurálásából fakadt ugyanis a mintaprogramban használt pin nem funkcionál CCP pinként ennél a mikrovezérlőnél. A mikrokontroller data sheetjében [1] keresgélve megtaláltam, hogy melyik pinek alkalmasak erre és eszerint átírtam a függvényt. Ezzel már tudtam generálni pwm jelet, amit oszcilloszkópon vissza is mértem. A generált pwm frekvenciáját alapvetően a beállított órajelfrekvencia és beállított érték határolja be.

3.3. Timerek konfigurálása idő mérésére

Az eddig generált jelekhez szükségesek olyan timerek is amelyek e jeleket tudják mérni. A CCP funkciók közül ehhez a capture funkció a legalkalmasabb. A mikrokontroller data sheetje [1] ezt Input Edge-Time Modenak nevezi. Ezzel a móddal elérhetjük azt, hogy amint a beprogramozott esemény bekövetkezik az órajelenként növelt számláló értéke beíródik az egyik regiszterbe. Ez számunkra ezért jó, mert így két jel esetén a regiszterek és az órajel értékéből megállapítható a köztük lévő időeltérés. Az implementáláshoz most is a Timer API-k és a programming example [2] nyújtotta a legnagyobb segítséget. A software packageben most nem találtam mintakódot ezért az interneten talált támpontokat is figyelembe vettem [6]. A mérési eredményeket adott időközönként kiírattam az UART-ra. A kivezetések megfelelő összekötése és mikrokontroller felprogramozása után az UART-on a várt értékeket kaptam.

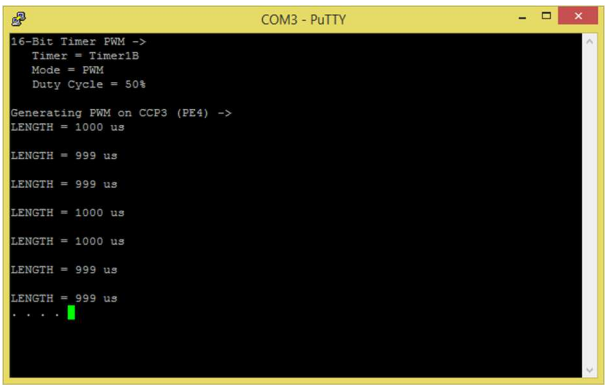


7. ábra 16-Bit Input Edge-Time Mode Example [1]

3.4. Az órajelfrekvencia és a mérési felbontás kapcsolata

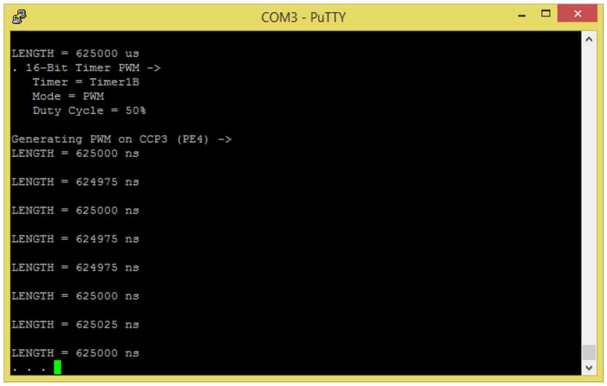
Ezen a ponton el kellett gondolkodnom azon, hogy ezek a mérési eredmények mennyire teljesítik a specifikációt.

Az első mérésénél egy 500 Hz-es jel fel- és lefutó élei közti időt mérte a műszer. Amint a 8. ábrán is látszik egyszer 1000 us másszor 999 us közti időt mér a műszer. Ebből látszik, hogy a felbontás 1 us-os és a pontosság is ebbe a tartományba esik. Ez a specifikációnak nem megfelelő. De hogyan is tudjuk ezt növelni. Ezt a mérési eredményt 25MHz-es órajelfrekvencia mellet mértük. A timer számlálójának értéke az órajel szerint nő, ezért az órajel frekvenciája összefüggésben van a felbontással. Egy 25MHz-es órajel periódusideje 40 ns, így ennél jobb felbontást ilyen frekvenciánál nem tudunk kapni. Ennél a mért eredménynél az 1 us-os felbontást az is okozza, hogy az eredményt us-ban adjuk meg, ezért a többi mérésnél az eredményeket ns-ban írjuk ki.



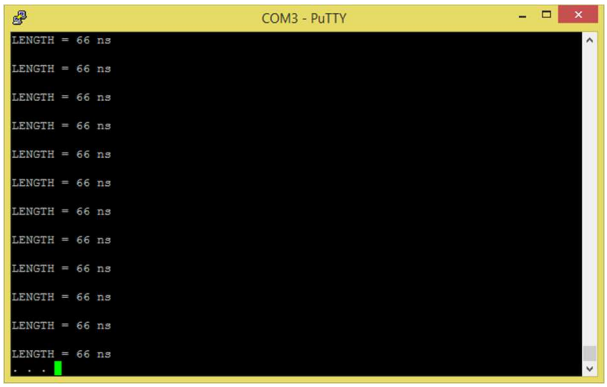
8. ábra 500 Hz-es jel fel- és lefutó élei között mért idő

A második mérésnél 800 Hz-es jel fel- és lefutó élei közötti időeltérést mértük. Itt már 40MHz-en fut az órajel, így a felbontás 25 ns. Ez az UART-on kiírt mérési eredményeken is látszik. Ez viszont még mindig nem elég jó a specifikációnál elvárthoz képest. Az órajel tovább növelhető 120MHz-ig is, ami 8,33 ns-os felbontást erdményezne és ezzel teljesítenénk az elvárt pontosságot is. Szerencsére a mintaprogramokban már megtapasztaltuk, hogy ez PLL segítségével lehetséges és ezért nem kellett utánanézni, hogy ez hogyan lehetséges.

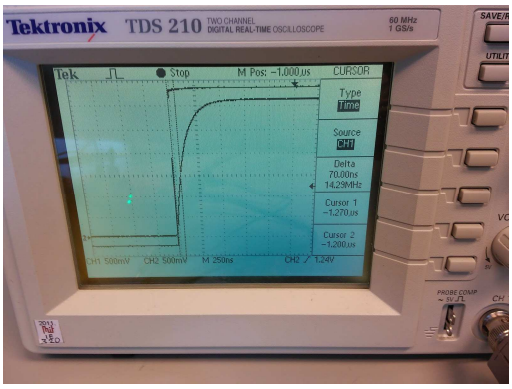


9. ábra 800 Hz-es jel fel- és lefutó élei között mért idő

Az eddigi mérések az adott jel fel- és lefutó élei közötti időt mérték vagyis az adott jel periódusidejének a felét. Azért, hogy megbizonyosodjak a mérési eredmények hitelességéről az egyik jelet egy RC taggal késleltetem, majd oszcilloszkópon visszamérem a felfutó élek közötti időeltérést. Az eredmények az UART-on és az oszcilloszkópon:



10. ábra RC taggal késleltett és a generált jel felfutó élei között eltelt idő



11. ábra RC taggal késleltetett és a generált jel mérése oszcilloszkópon

Az UART-on 66 ns-ot ír ki a mikrovezérlő, míg az oszcilloszkópon kb. 70 ns-ot mértem. Ez azt jelenti, hogy kb. 4 ns a különbség a mikrovezérlő által mért és a valóságos késleltetés között. Ez benne van a 8,33 ns-os pontosságban, ezért a mérési módszerünk jónak tekinthető.

Ezután már csak az szükséges, hogy ezeket a mérési eredményeket a weblapra is kihelyezzük. Viszont mielőtt ezt megtettük volna konfiguráltunk még néhány timert, hogy capture módban tudjanak a jelek között időt mérni. Ez azért hasznos, mert így több jel között is tudunk mérni. Mivel 12 CCP pint tudunk konfigurálni, így alapvetően egy referencia jelhez képest 6 jelet is tudunk mérni. Eddig viszont csak 4 timert használtam fel erre, mert egyrészt a pwm jelekhez is timerek szükségesek másrészt, amíg külsőleg nem tudunk jelet csatlakozni addig fölösleges a többi időzítőt lefoglalni erre a célra. Az eddig használt timerek és CCP pineket az alábbi táblázat (12. ábra) foglalja össze:

Funkció	Timer	Pin név	Port
PWM	0	T0CCP0, T0CCP1	D0, D1 és L4, L5
Időmérés	1	T1CCP0, T1CCP1	A2, A3
	2	T2CCP0, T2CCP1	A4, A5
	3	T3CCP0, T3CCP1	M2, M3

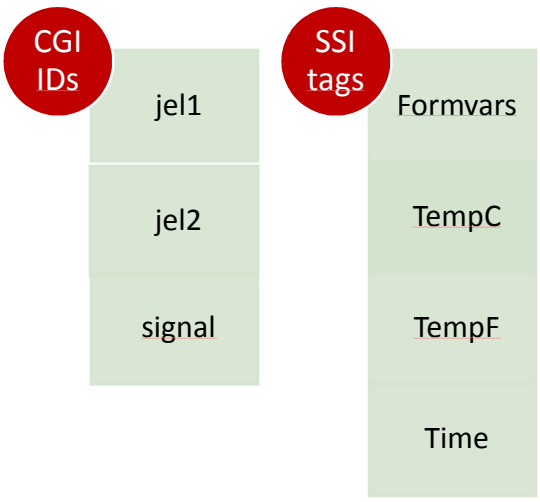
12. ábra Használt timerek és pinek kiosztása

4. Felhasználói interface elkészítése

A mérési eredményeket eddig UART-ra írattam ki. A specifikáció teljesítéséhez viszont a webre kell ezeket kiírnom. A továbbiakban egy olyan webes felület megvalósítását tűztem ki célul, amelyen ki tudjuk választani, hogy mely jelek mely élei között mérjünk illetve hogy a mérési eredmények táblázatokba, grafikonokban megjeleníthetőek legyenek. Ehhez jobban meg kellett ismernem az egyik mintaprogram (enet_io) által már bemutatott webkontroll módszereket. A webkontroll módszer kiválasztásán kívül eltávolítottam a Texas Instruments által megalkotott designot és egy sajátot hoztam létre.

4.1. SSI/CGI handlerekkel

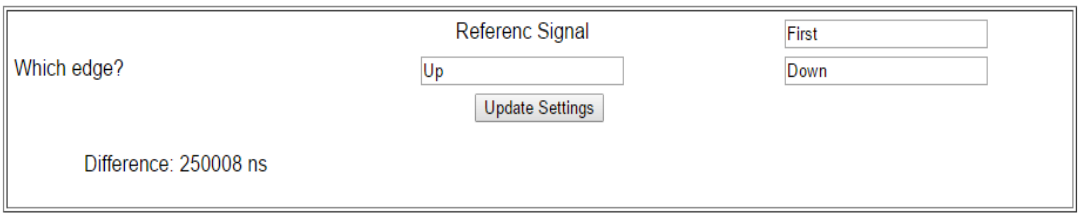
Mivel már a hőmérséklet kiírásánál is ezt használtam, így a mérési eredmények kiírását is ezzel a módszerrel végeztem el először. Ennek a webkontroll módszernek a működését úgy a legegyszerűbb megérteni, hogy ha ezt egy adott példa alapján nézzük meg. Egy formot tekintve a submit gomb megnyomásával a webbrowser requestet küld a webszerver felé, amely pedig ezt továbbítja annak a programnak, amely ezt feldolgozni képes. CGI request esetén a program befogadja az URL-hez csatolva küldött információkat és ezek alapján egy bizonyos programozott működést hajt végre, majd pedig CGI reponse-zal válaszol a webszervernek, ami ezt továbbítja a webbrowserhez. SSI request akkor jön létre, hogy ha webbrowser SSI taget talál a weblapon, ekkor a webszerverhez beérkező request alapján a webszerver egy olyan választ küld, amelyben a tagek értékének megfelelő honlapot küld vissza.



13. ábra a használt CGI ID-k és SSI tagek

Ezután elkezdtem megalkotni a kívánt funkcióknak eleget tevő weblapot. Az enet_io-ban már megírt működés sokat segített ennek a létrehozásában. Alapvetően a hőmérséklettel már bővített projektet bővítettem. Négy darab SSI taget használtam. A Formvars egy javscrpit headert szúr be, ami lehetővé teszi a form működését. A TempC és a TempF tagek a

hőmérséklet értékei szolgáltatják Celsiusban és Fahrenheitben. A Time tag pedig a legútóbb lekérdezett időt adja meg. CGI ID-ből 3 darab van és ezek közül mindegyik az időmérés feladatát szolgálja. A html kódban 3 lenyíló lista van létrehozva. Ezek értékei kapják meg a CGI ID-k. A signal azt az értéket kapja meg, hogy a referenciajelhez képest melyik jelet mérjük. A jel1 és a jel2 azt határozza meg, hogy mely élek között történjen a mérés. Sajnos ezen érékek csak egy mérésre érvényesek és utána mért értékek az alapból beállított jelek közti időt méri, így minden egyes mérésnél be kell állítani, hogy mely jelek mely élei között szeretnénk mérni. Ezt a problémát valószínűleg ki lehet küszöbölni, viszont ezzel már nem foglalkoztam, mert úgy láttam, hogy érdemesebb áttérnem a másik webkontroll módszerre. Ezt a döntésemet az indokolta, hogy nem találtam megoldást arra, hogy ennél a webkontroll módszernél, hogyan tudom elérni, azt hogy a mérési eredményeket táblázatba és/vagy grafikonba tudjam helyezni.

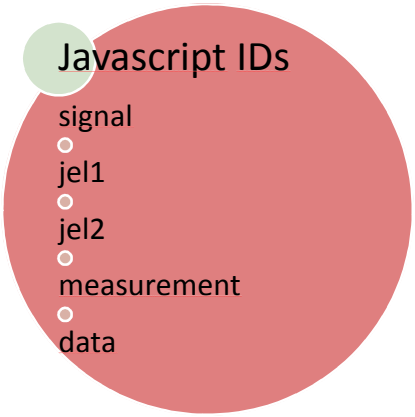


14. ábra az CGI/SSI handelrekkel létrehozott weblap időeltérés méréséhez használt részlete

4.2. http request és Javascrpit

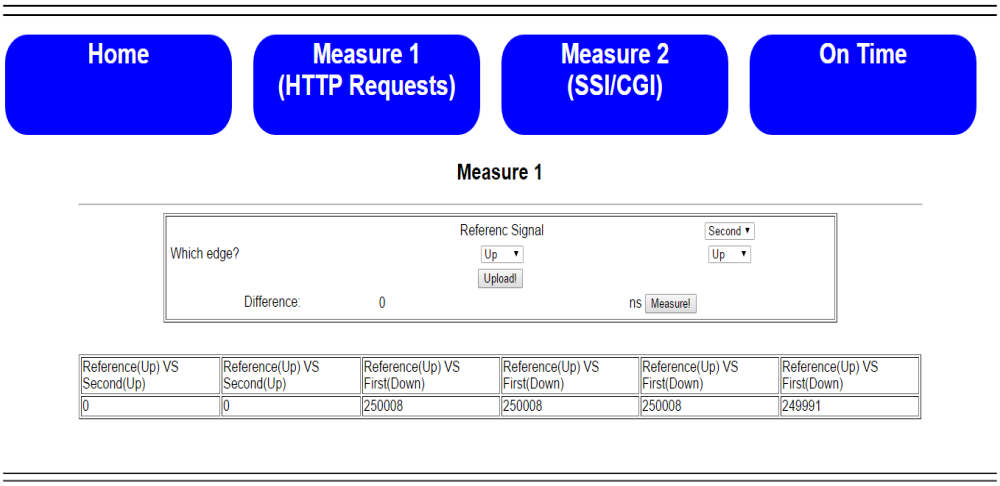
Ez a webkontroll módszer nagyban épít a JavaScriptre ezért még mielőtt elkezdtem volna foglalkozni ezzel utána néztem, hogy milyen tulajdonságai vannak ennek a nyelvnek [7].

Ezután megértettem, hogy az enet_io mintaprogramban hogyan működik ez a webkontroll módszer. A webbrowser a beprogramozott gombnyomás vagy frissítés hatására http requestet küld, amelyet JavaScript segítségével állít elő. Ezt a mikrovezérlő elkapja és a kiolvasott URL lapján egy választ küld, amit a browser beilleszt a weblapba dinamikusan. Ezeket a feladatokat a projektben a fájl rendszert támogató réteg (io_fs.c) látja el.



15. ábra használt ID-k a JavaScripthez

A számomra szükséges funkciók megalkotásánál szintén nagy hasznomra voltak a már implementált funkciók, viszont ezek ellenére JavaScript hiányosságaim miatt lassan haladtam. Mindenesetre sikerült megalkotnom ezzel a módszerrel is a weblapot. Alapvetően öt ID-t használtam erre. A signal, jel1 és a jel2 ID az CGI ID-khoz hasonló funkciót töltenek be, a lenyíló listák értékei tárolják. A signal azt kapja meg, hogy a referenciajelhez képest melyik jelet mérjük (First, Second, Third). A jel1 és jel2 a referencia jel és a választott jel éleit tudjuk beállítani. A measurement ID a Time SSI-hez hasonló funkciót lát el, Az ehhez az ID-hez tartozó bekezdésbe írja be a legutóbb mért értéket. A data ID helyére az előzőleg mért 6 eredményt íratjuk ki táblázatos formában JavaScript segítségével. A Táblázatban a mért érték mellett az is olvasható, hogy mely jelek mely élei között történt a mérés. A lenyíló listákban beállított értékek megmaradnak, azok csak akkor változnak, hogy ha változtatunk a weblapon rajtuk.

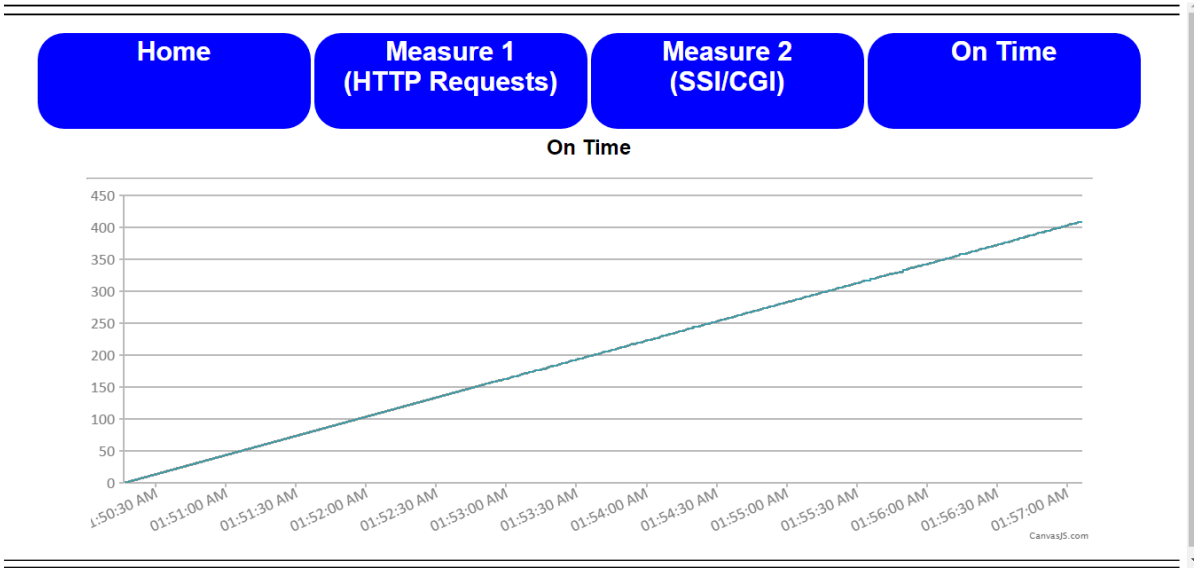


16. ábra http requesttel létrehozott weblap

4.3. Új design és on time grafikon

Következő lépésként eltávolítottam a Texas Instruments által készített weblap designt és egy újat egy sajátot hoztam létre. Ez nagyon egyszerű volt, mivel egyetemi pályafutásom alatt már csináltam honlapot és az alapján könnyen megalkottam a 15. ábrán megtekinthető weblap stílust.

Ezután elhatároztam, hogy egy külön oldalon létrehozok egy grafikont, ami azt mutatja, hogy mennyi idő telt el a készülék bekapcsolásától és folyamatosan frissül az eredményekkel. Ennek a megalkotásához még jobban meg kellett ismernem a JavaScriptet. Rájöttem, hogy ehhez keresnem kell egy csatolható könyvtárt, amelyben meg vannak azok a funkciók, hogy ezt elérjem. Ehhez a CanvasJS chartjait [8] használtam. Azért ezt választottam, mert ehhez találtam látványos demókat. Utólag kitekintve ez nem a legjobb választás viszont idő hiányában nem volt időm más után nézni, helyette szerettem volna, hogyha sikerül létrehoznom a várt on time grafikonomat. Ezen grafikonábrázoló modul mellett még szükségem volt egy timerre, amely számolja, hogy hány másodperc telt el a bekapcsolás óta. Ekkor már rendelkeztem annyi tapasztalattal, hogy ez magamtól is létrehozzam, de sajnos nem úgy működött ez, ahogy én elvártam volna. Szerencsére a qs_iot mintaprogramban már meg volt valósítva ez a funkció, így ezzel összehasonlítva az általam létrehozott kódban ki tudtam küszöbölni a problémát. (Az interrupt kezeléssel volt a probléma)



17. ábra On time diagramm

5. Összefoglalás és további feladatok

A megadott specifikációknak eleget tevő rendszer elkészült 8,33 ns-os felbontással és egy kezdetleges weblappal, amelyen konfigurálható, hogy mit mérjen illetve megjeleníthető a mérési eredmények. A továbbiakban mérési eredményeket ábrázoló grafikonok elkészítése lenne a feladat, hogy a felhasználó szemléletesebben lássa az eredményeket. A mérési eredményeket jó lenne tárolni valahol. Ennek a feladatnak a megoldási lehetőségeinek megállapítása, megválasztása és megalkotása is hasznos lehet. A mérőberendezés legnagyobb hátránya, hogy csak a saját generált jeleit tudja mérni, ezért a továbbiakban szükséges külső jelekkel is kipróbálni a működést, majd ezután egy front-endet építeni hozzá, amely segítségével akár BNC csatlakozó segítségével is rá tudnánk kötni a műszereinket.

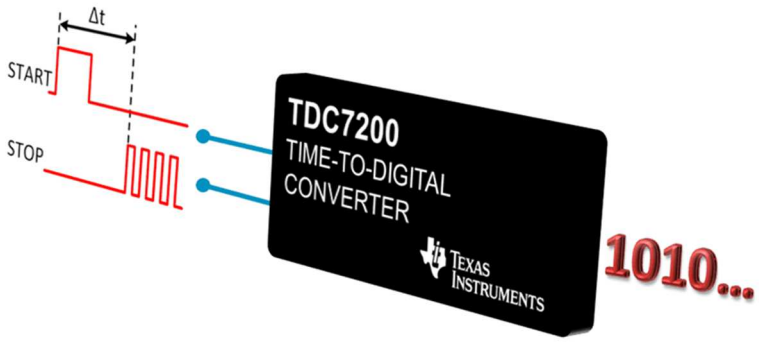
6. Továbbfejlesztési lehetőségek

Annak érdekében, hogy a lehető legtöbb jel közötti mérést a lehető legjobb felbontásban biztosítani tudjuk, ahhoz szükségünk van további modulokkal, eszközökkel való összecsatlakozására.

Az egyik lehetséges megoldás, hogy FPGA-val kötjük össze az eddigi környezetet. Ebben az esetben az FPGA mérné az időt és a frekvenciát, míg a Launchpad csak webszerverként működne.

Lehetőség van arra, hogy leváltanánk a hardwaret és egy erősebb egy kártyás számítógéppel, Linux operációsrendszerrel oldanánk meg a problémát. Ebben az esetben lehetőségünk lenne a párhuzamos végrehajtásra viszont hátrány, hogy egy új rendszert kéne megismernünk.

Egy további lehetőség a Texas Instruments TDC 7200-as time-to-digital converterének illesztése a már kész rendszerhez. Ennek segítségével tovább növelhető lenne a felbontás.



18. ábra TDC 7200 Functional Diagramm

.

Ábrajegyzék

Irodalomjegyzék

[1] Tiva C Series TM4C1294NCPDT Microcontroller Data Sheet (Rev. B) [Online] Available: http://www.ti.com/lit/ds/symlink/tm4c1294ncpdt.pdf [Hozzáférés dátuma: 20 02 2016]

[2] TivaWare™ Peripheral Driver Library for C Series User's Guide (Rev. A) [Online] Available: http://www.ti.com/lit/ug/spmu298a/spmu298a.pdf [Hozzáférés dátuma: 20 02 2016]

[3] http://www.ti.com/ww/en/launchpad/launchpads-connected-ek-tm4c1294xl.html#project0 [Hozzáférés dátuma: 28 02 2016]

[4] Mikrokontroller alapú rendszerek elektronikus jegyzet 3. fejezet [Online] Available: https://www.aut.bme.hu/Upload/Course/VIAUA348/hallgatoi_jegyzetek/Mar_ea_3.pdf [Hozzáférés dátuma: 08 03 2016]

[5] Milan Verle: „PIC Microcontrollers - Programming in Assembly” [Online] Available:http://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/chapter/ccp-modules/ [Hozzáférés dátuma: 19 03 2016]

[6] http://e2e.ti.com/support/microcontrollers/tiva_arm/f/908/t/467041#pi239031350filter=answers&pi239031350scroll=false [Hozzáférés dátuma: 10 04 2016]

[7] http://www.w3schools.com/js/ [Hozzáférés dátuma: 04 05 2016]

[8] http://canvasjs.com/html5-javascript-dynamic-chart/ [Hozzáférés dátuma: 09 05 2016]

[9] TDC7200 Time-to-Digital Converter for Time-of-Flight Applications in LIDAR, Magnetostrictive and Flow Meters (Rev. D) [Online] Available: http://www.ti.com/lit/ds/symlink/tdc7200.pdf [Hozzáférés dátuma: 20 05 2016]