SZAKDOLGOZAT

Internetrádió készítése

Pázmány Péter Katolikus Egyetem

2012

Készítette:

Fichó Karolina

mérnök informatikus BSc

Témavezető:

Tihanyi Attila

Nyilatkozat

Alulírott Fichó Karolina a Pázmány Péter Katolikus Egyetem Információs Technológiai Karának hallgatója kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, és a szakdolgozatban csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen a forrás megadásával megjelöltem. Ezt a Szakdolgozatot más szakon még nem nyújtottam be.

……..

Fichó Karolina

Tartalom

Tartalmi összefoglaló – A feladat ismertetése

Az internetrádiók egyre növekvő népszerűségét figyelembe véve, a szakdolgozatom fő témája a kimondottan internetes rádióállomások hallgatására szolgáló webrádiók tanulmányozása, illetve egy olcsó hardveres megoldás javaslása az interneten fellelhető rádióműsorok hallgatására.

A feladat elkezdésénél elsőként át kellett tekintenem az online rádió megoldásokkal foglalkozó szakirodalmat, különös tekintettel a rádióvevő programokkal fogalakozókra. A megfelelő háttérismeretek birtokában meg kellett határoznom a feladat megoldásához szükséges alapvető eszközöket és programokat.

A megoldás során ki kellett választanom egy hardveres eszközt, amely képes az internetes rádióállomások műsorainak fogadására és lejátszására. Elemeznem kellett a kiválasztott hardver működtetéséhez szükséges szoftverrendszer felépítését. A tervek alapján össze kellett állítanom, és tesztelnem a rendszert – megállapítva a gyenge pontokat is.

Az általam választott eszköznek képesnek kell lennie – egy hagyományos rádióhoz hasonlóan – a minimális funkciók ellátására: csatornaváltás, vagy hangerő-szabályozás.

Mivel ez a rádió a hagyományostól eltérően nem az antennáján érkező jeleket, hanem a számítógép-hálózaton érkező adatcsomagokat dolgozza fel, és teszi hallhatóvá, úgy kellett megvalósítanom az állomásválasztási funkciót, hogy az a környezetnek megfelelően egy, a hálózatról elérhető, és működtethető weblap segítségével is programozható legyen.

Célom volt, hogy a netrádió tényleges működés közben a saját kijelzőjén mutassa az előre programozott állomásokat, kezelőgombjaival ezeket váltogatni lehessen. Ugyanezekkel a kezelőszervekkel kellett megoldani a hangerő-állítási feladatokat is. A weblapnak szintén ezeket a feladatokat kell ellátnia.

Ezen kívül rámutatok a fejlesztési lehetőségekre is, továbbá javaslatot teszek az esetleges hibák, hiányosságok kiküszöbölésének módjára.

Abstract

Because of the growing popularity of internet radios, my thesis specifically focused on the internet radio devices, with we can listen to the online radio stations. Moreover I had to suggest a low-cost hardware solution for this purpose.

In the beginning, my first task was to learn more information about the online radio solutions, especially about the programs in radio receivers. With the appropriate possession of background knowledge, I had to specify the basic hardware and software requirements.

The solution needed to select a hardware device capable of receiving the Internet radio stations' signals. I had to analyze the chosen hardware system architecture, and the requirements of the software, to run the program. According to the plans, I had to set up and test the system- noting the weak points.

The chosen system must be able to – like a traditional radio – the minimal functions like channel switching or the volume control.

Since this radio is not like the traditional ones, does not receive the radio signals from its antenna, but process data packets received from the internet network, and managed to be heard. I had to implement the station selecting function – according to the given environment. It can be available and can be programmed using a simple webpage.

My goal was to create an internet radio system, which during its actual operation, indicate the pre-programmed stations on its own display. The controls can be found on the internet radio board. I had to manage the station switching and the volume control functions with these buttons. The webpage also has to fulfill these tasks.

In my thesis, I am going to present how I could fulfill the given tasks, and the analyzed topics, I had to study.

In addition, I pointed to the potential for developments as well, and I suggest some solutions to the errors.

Bevezetés

Napjainkra az internet szinte megkerülhetetlen kommunikációs eszközzé vált. Különálló információforrásként jött létre, mégis a fejlődés során összeolvadt egyes médiumokkal (televízió, könyv, újság, rádió). Az internetrádió megalakulásának elsődleges célja a rádiós tevékenység lehetőségeinek kiszélesítése volt.

Rádióösszeköttetés az adó- és a vevőállomás közötti vezeték nélküli jelátvitelt jelent. A hagyományos esetben a rádióállomások által elérhető műsorokat egy adóantenna sugározza a végberendezések felé, amelyek a vevőantennájukon keresztül fogadják az elektromágneses hullámokat.

A digitális jelfeldolgozás modernizációjának köszönhetően ma már ez az összeköttetés más módon is megvalósulhat, továbbá már más médiumokon is hallgathatóak az általunk kiválasztott rádióállomások.

A műsorok elérhetőek különböző műholdak, kábelek és egyéb telekommunikációs eszközök segítségével, úgy, mint mobiltelefonok, vezetékes illetve vezeték nélküli szélessávú hálózaton keresztül. Az internetes hálózaton keresztül valósul meg egy új adatátviteli forma, a „streaming", mely a média tartalmak gyors elérését, vagy akár letöltését teszi lehetővé.

A rádióvevőkön manapság már nem csak a rádiókészülékeket kell érteni, hanem a különböző multimédiás számítástechnikai eszközöket is, mint például asztali számítógépeket, notebookokat, PDA-kat, okos telefonokat, melyeknek legfőbb hátránya magas, nem mindenki számára elérhető árfekvésük. A leginkább költségkímélő megoldást kifejezetten az internetes rádióállomások hallgatására szolgáló internetrádiók jelentik, melyekhez csupán vezetékes vagy vezeték nélküli hálózati kapcsolat szükséges.

A rádiósugárzás vételénél már nem csak hangadatokat vagyunk képesek fogadni, hanem különböző, a hangadat tulajdonságaira vonatkozó metaadatokat (kódolási paramétereket, zeneszámcímeket, rádiócsatorna-neveket stb.) is. Ezeket az audio fájlokat valós időben hallgathatjuk vagy eltárolhatjuk, majd késleltetéssel lejátszhatjuk őket.

Az internet elterjedését, népszerűségét, és mára már „nélkülözhetetlenségét" kihasználva, a rádióadók fokozatosan elérhetővé tették műsoraikat az interneten is az elmúlt két évtized során. Ez lehetőséget biztosított a már saját frekvenciával rendelkező rádióknak, hogy kiterjesszék lefedettségüket. Manapság a nagyobb rádióadók mindegyikének van már internetes elérése is. Az állomások számára nagyobb hallgatóközönséget jelenthet, ha online is közzéteszi a műsorait.

Létrejöttek olyan adók is, melyeket kizárólag az interneten keresztül lehet hallgatni. A legismertebb internetes rádiószolgáltatók az amerikai AOL Radio Network[1], vagy a Live365[2], melyek kínálatában már több, mint 7000 csatorna közül választhatunk.

A rádiók internetes felülete a felhasználók számára nyújtanak új szolgáltatásokat, a hagyományos földfelszíni rádiósugárzáshoz képest. Ezen oldalakon a hallgatók képesek elérni a külföldön sugárzott műsorokat is, maguk választhatnak, hogy milyen típusú zenét, adásokat szeretnénk hallgatni, vissza tudják keresni a már régebbi műsorokat is.
Ma már mi magunk létrehozhatunk saját rádióállomásokat – frekvenciaengedély beszerzése nélkül –, melyeket akár saját, tetszőleges tartalommal tölthetünk fel. Mivel csak internetkapcsolatra van szükség, a felhasználóknak nem kell antennákat, erősítőket beszerezniük saját adásuk eléréséhez. Már a rádiókészülékek megléte sem nélkülözhetetlen ahhoz, hogy kedvenc rádióállomásunkat élvezhessük.

[1] http://music.aol.com

[2] http://www.live365.com/index.live

1. Rádióadás az Interneten

Az első internetes rádióműsort a 90-es évek elején sugározták Amerikában. Ez az esemény mérföldkőnek számított a rádiózási szokások terén. Egy újfajta elérhetőségnek biztosított a műsorszolgáltatók számára, megnövelve ezzel a hallgatóközönség létszámát.

A rádióállomások száma hihetetlen növekedésnek indult, népszerűségük jócskán meghaladta korábbi mutatóikat. Maguknak a rádióadást szolgáltató cégeknek innovatív, de mindenekelőtt költségkímélő megoldást jelentett ez a fejlődés, hiszen a globális vételhez nem kellett újabb adótornyokat, digitális és analóg átviteli rendszereket, vagy antennákat biztosítani.

Az interneten történő műsor továbbításához lényegében szükség van egy webszerverre, melyen a sugárzott adatot szeretnénk tárolni, illetve internetkapcsolatra, mint közvetítő médiumra, továbbá egy kódoló dekódoló végberendezésre, amely képes az adatot a lejátszáshoz feldolgozni. A következő megállapításokhoz a „The technology of video and audio streaming" [1] című forrás volt a segítségemre.

1.1. Az adások elérése a virtuális téren keresztül

Az audio anyagok interneten történő közlésének két alapvető formája van. Az egyik megoldás – sokáig ez volt az egyetlen –, amikor manuálisan letöltjük a rögzített műsorokat. Ezeket a műsorokat a szolgáltatók tehetik közzé a honlapjukon. A letöltött adásokat visszajátszhatunk a saját számítógépünkön, vagy rátölthetjük olyan eszközeinkre, melyek a letöltött adatokat lejátszani képesek.

Ebben az esetben a rádióhallgatás nem nyújtja a konkrét rádiózással járó felhasználói élményeket, nem valós idejű a szolgáltatás. Csak akkor hallgathatjuk meg a lejátszani kívánt tartalmakat, ha azt már teljes egészében letöltöttük. Mindezek mellet a letöltött tartalom tárolása, esetleges megosztása harmadik féllel felvet számtalan jogi kérdést, a licenszproblémákról nem is beszélve.

A másik, és egyben a legjellemzőbb módszer rádióműsorok interneten történő elérésénél a „webcasting" (vagy más néven „streaming"), mely egy elektronikus egyidejű, vagy azonnali

adatátvitel az interneten keresztül a szerver és az adatfolyamot fogadó eszköz között. Nem szükséges a teljes adat letöltése, hiszen ha elegendő adatot fogadtunk, akkor már elkezdődhet a lejátszás. Ebből a tulajdonságból következik a streaming módszer legnagyobb előnye, hiszen egy valós idejű a szolgáltatásról beszélhetünk. A fogadott adatokat azonban nem tároljuk, így minimális a memóriahasználat.

Míg a hanganyag egyidejű letöltésénél csak egy ún. unicast kommunikációra van lehetőség, addig a webcastingnál unicast, multicast vagy broadcast kapcsolat közül is választhatunk.

A webcast alapjait már a 90-es évek legelején kidolgozták, azonban a legelső kísérleti internetes broadcast rádióadásra csak 1994-ben került sor, egy baseball meccs keretében. Ez az esemény maga után vonta a legelső, csak az interneten elérhető rádióállomás létrejöttét, a „First Radio”-ét.[2]

Eleinte nagyon alacsony sávszélességgel és hangminőséggel közvetítettek (8 kbps). Később, ahogy fejlődött a technika, az internet sebessége nőtt, és egyre jobb tömörítési lehetőség adódtak, úgy vált valóban élvezhető minőségűvé az internetes rádiózás.

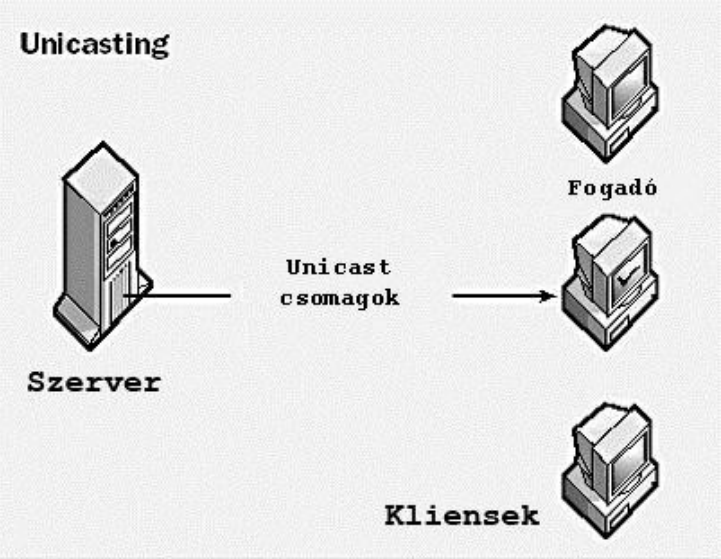
Az interneten történő kommunikáció során a teljes adatfolyam (továbbiakban „stream”) kisebb csomagokra van bontva, azonban a fogadó a beérkezett csomagokat egy egységként kezeli, aminek hatására a lejátszás folyamatos lesz. A fogadott adatok nem tárolódnak el az eszközön, továbbá csak akkor hallgathatók, ha a felhasználó kapcsolódott a weben keresztül a műsort szolgáltató szerverhez.

1.2. Kommunikáció a szerver és a kliens között

A jelenlegi streaming rendszerek architektúrája igen hasonló. A sugárzást egyetlenegy szerver, vagy egy statikus szerver hálózat végzi. A szerver és a kliens közti üzenetváltási lehetőségeket internetrádiós témakörben az „European Broadcasting Union” két munkatársa foglalta össze, mely nagy segítségemre volt a módszerek összehasonlításában.[3]

Kommunikációs szempontból megkülönböztetünk unicast, multicast és broadcast típusú átvitelt:

- Unicast

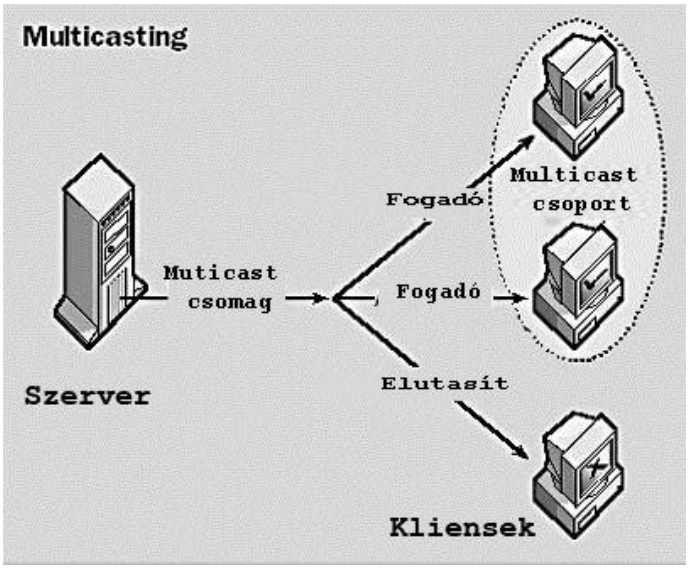


1. ábra - Unicast kommunikáció [20]

Az interneten keresztüli hálózati kommunikációra az unicast, az egyetlen küldő és egyetlen fogadó közti üzenetváltás a legjellemzőbb. Ebben az esetben a kliens kérést küld a szervernek, melyben lekéri a szerver által tárolt és a felhasználó által kiválasztott streamet.

Ez nem mindig a legjobb módszer, mivel az adatfolyamot egyetlen forrás továbbítja, ezáltal a szerver hamar elérheti a maximális kapacitását, könnyen túl lehet terhelni.

- Multicast

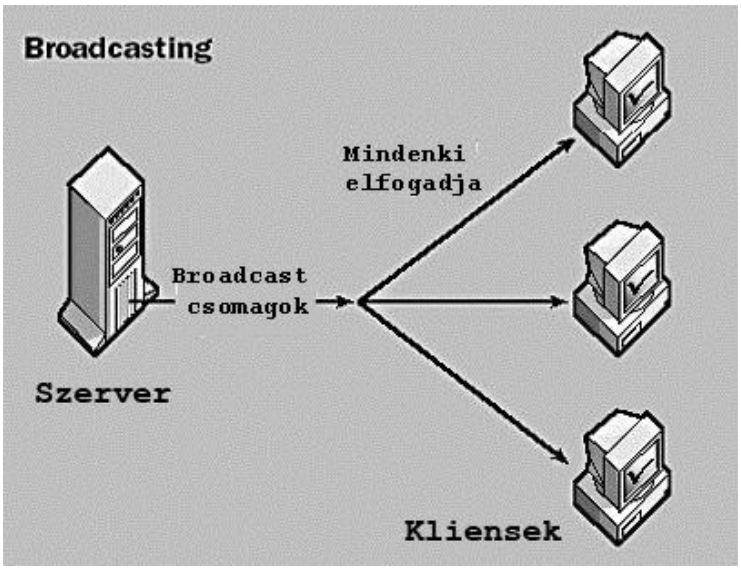


2. ábra – Multicast kommunikáció [20]

Multicast kommunikáció során egy streamet juttatunk el több felhasználó számára. A felhasználó általában nem közvetlenül a szerverrel kommunikál, hanem egy köztes routerrel. Ebben az esetben csupán egy kérés van, de a router minden egyes kliens számára átmásolja a kapott adatokat.

Ez a megoldás nem alkalmas az on-demand szolgáltatásokra, vagyis nem tudunk igény szerinti adatokat letölteni egy központi adatbázisból, a hallgatott adatokat nem tudjuk megállítani, és később visszajátszani.

- Broadcast



3. ábra – Broadcast kommunikáció [20]

Rádiósugárzás esetén a broadcast típusú műsorszórás a leggyakoribb. Ebben az esetben szintén a stream másolatát kapják a kliensek.

A multicasthoz képest azonban minden, a hálózaton tartózkodó kliens megkapja az adatfolyamot, ugyanakkor nem ad a kliensnek vezérlési lehetőséget, a felhasználó csak arról dönthet, hogy az adást hallgatja-e, vagy sem.

- WiMax

A WiMax (Worldwide Interoperability for Microwave Access) hálózatok vezeték nélküli kommunikációs megoldást biztosítanak, a kábel- és DSL-alapú internetszolgáltatás részére. Olyan területeken, ahol nincs megfelelő szélessávú hálózat kiépítve, kifizetődő alternatíva.

- P2P

A P2P (Peer-to-Peer) hálózat két számítógépet direkt módon köt össze, köztes eszköz (router, switch, hub) nélkül. Ez a megoldás lehetőséget biztosít a felhasználónak, hogy a szabadon böngésszen a másik gép tartalmai között, egyfajta fájlcserlés jön létre. A szerverhibákat küszöbölhetjük ki vele.

- Podcasting

Podcasting megoldás egy előfizetéses rendszer, a felhasználók csak a fizetés után tölthetik le a szerveren található tartalmakat. A lejátszáshoz ezeket a fájlokat könnyedén rá tudjuk tölteni a lejátszóinkra.

A podcastinghoz a felhasználónak mindenképp szüksége van internetkapcsolatra, és egy podcasting szoftverre. A szoftver automatikusan ellenőrzi az általunk beállított rádióállomások tartalmát. A legújabb műsorokat, amint elérhetőek, azonnal letölti.

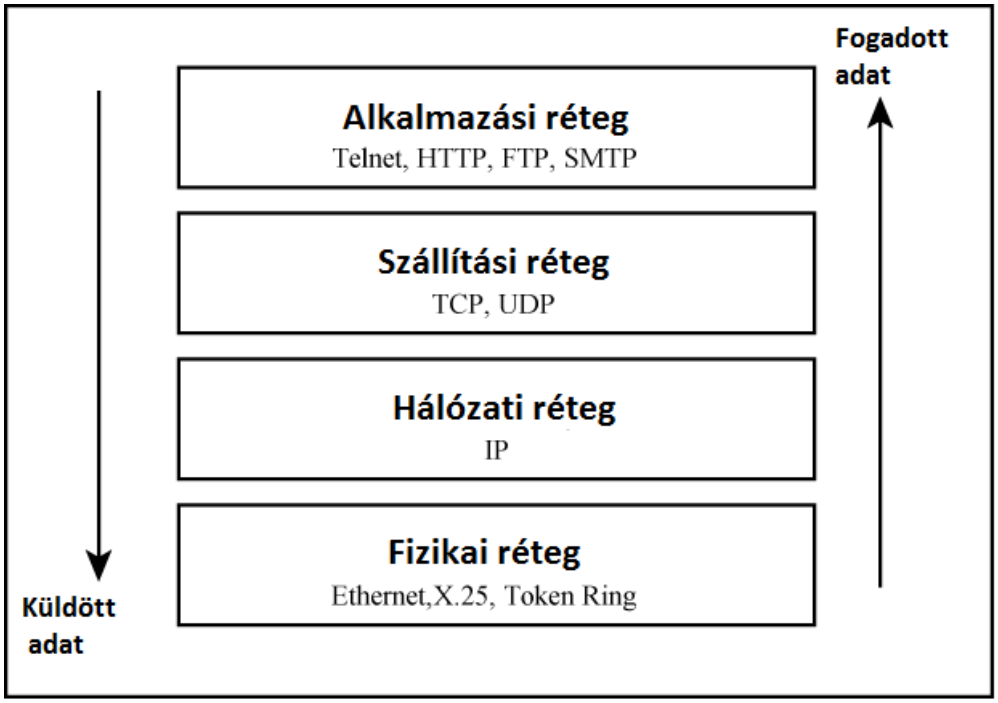
1.3. Kommunikációs protokollok

A kommunikáció felépítéséhez szükség van bizonyos szabványokra, ún. protokollokra a szerver és a kliens között. Léteznek továbbá kimondottan a valós idejű szolgáltatásokra létrejött protokollok is. Feladatom további teljesítéséhez mindenképpen foglalkoznom kellett ezen szabványok megértésével és tulajdonságaival.

Mivel az egyes protokollok alkalmazása több összetevőtől is függhet, mint például a szerver elérési módjaitól, a továbbított adat, illetve a hálózat tulajdonságaitól, ezért az egyes jellemzőket rádióadás szolgáltatásának szempontjából részletezem, s csupán azokat, melyek megléte elengedhetetlen, vagy a legjellemzőbb.

1.3.1. A TCP/IP modell

Leghatékonyabban a TCP/IP (Transmission Control Protocol/Internet Protocol) modellel ábrázolható az adatátvitel. Ez a modell egy általános struktúrát ajánl a hálózati rétegek osztályokba történő rendezésére. A felhasznált alapvető protokollokról alapvető információkat a hozzájuk tartozó RFC-k (Request For Comments) dokumentumok adhatnak. A modell felépítésével rengeteg szakirodalom foglalkozik, az általam választottból [4], a dolgozatom témájához elengedhetetlennek tartom megemlíteni a következőkben tárgyalt szabványokat.



4. ábra – TCP/IP referenciamodell [21]

A TCP/IP protokollstruktúra foglalja magába az adatátvitelt és a kommunikációt jellemző szabványokat.

A referenciamodell egyfajta hierarchikus szerveződésen alapul, mely több rétegre van osztva, ahol a felső rétegek igénybe veszik az alsóbb rétegek szolgáltatásait.

1.3.2. Az IP protokoll

Az internetes rádióadás szempontjából az egyik legfontosabb protokoll az IP protokoll. Ezen réteg feladata a hálózati kommunikáció kialakítása. Itt történik a csomagkezelés, és a csomagtovábbítás a fenti rétegek felé. A csomagok eltérő hosszúságúak lehetnek.

Fontos megjegyezni, hogy az IP protokoll nem végez csomagellenőrzést, hiszen ez a felsőbb réteg feladata. Az IP protokollal bármilyen hálózati kommunikációt megvalósíthatunk, legyen az vezetékes, vezeték nélküli, unicast, multicast stb.

1.3.3. A TCP és UDP protokoll

Az adatfolyam átvitelére egyik számítógépről a másikra a TCP (Transmission Control Protocol)[5] vagy az UDP (User Datagram Protocol)[6] protokoll alkalmazható. Az, hogy éppen melyiket implementáljuk, függ az adat és a hálózat tulajdonságaitól.

A csomagoknál nincs garancia arra, hogy egymás után érkeznek. Változó internetsebesség esetén késhetnek, vagy az átvitel során sérülhetnek, elveszhetnek. Ezeket a hibákat küszöböli ki a TCP protokoll.

A TCP protokoll képes a hibajavításra és a sorrendtartásra. Az UDP protokollal szemben minden műveletre, küldött adatra nyugtát vár. Ha a csomag átvitele közben nem történik hiba vagy csomagvesztés, akkor a nyugta megérkezése után küldi a következő csomagot. Ez a megoldás megbízhatóbb kommunikációt eredményez, azonban ha túl sokat kell várni egy-egy nyugtára, akkor igen nagy késleltetés léphet fel. Ez befolyásolja a rádióadás valós időben történő hallgatását, időtúllépés miatt megszakadhat a kapcsolat a szerverrel. Az UDP azonban soha nem vár visszajelzést, így gyorsabb hálózati kommunikációt érhetünk el.

TCP protokoll esetén, ha a csomag sérülten érkezik, jelzi azt a küldőnek, akitől kéri a csomag újraküldését. Az UDP nem végez hibaellenőrzést, amint megkapja a csomagot, küldi tovább a célállomás felé. Ez főleg akkor probléma, amikor a csomagok túl gyorsan érkeznek, a fogadó pedig nem képes ugyanebben a tempóban feldolgozni az adatokat. Ennek csomagvesztés az eredménye, amit fontos adatoknál nem engedhetünk meg magunknak. Rádióadás esetében minőségromlást eredményez. A TCP protokoll képes befolyásolni az adatfolyam küldésének sebességét. Jelezhet a küldőnek, hogy a fogadó mennyi adatok képes fogadni a bemenetén.

Megjegyezném, hogy ha a gyorsaság fontosabb, mint a megbízhatóság, akkor érdemes az UDP protokollt használni, ellenkező esetben a TCP a javallott.

1.3.4. A valós idejű lejátszáshoz szükséges protokollok

Érdemesnek tartom megemlíteni a kimondottan a valós idejű lejátszásra létrehozott protokollokat. Ilyen szabvány az RTP (Real Time Transport Protocol) [7], mely új mezőkkel egészíti ki az UDP és TCP csomagokat, pl. hibajavító, sebességjelző vagy időbélyeg.

Az RTP feladatai közé tartozik, hogy az előforduló csomagtovábbítási hibákat hatékonyan kezelni tudja, nem törekszik az újraküldésre, hanem a megérkezett adatot a legjobb minőség szerint próbálja továbbítani.

Az RTSP (Real-Time Streaming Protocol) [8] lehetővé teszi egy kliens számára, hogy a szervert távolról vezérelje, a kapcsolat tulajdonságaitól függően. Használatával képesek vagyunk távolról befolyásolni a szerver adatküldési sebességét is.

Az RTCP (Real-time Control Protocol) [9] az adatátvitel során átvitt tartalmak minőségét ellenőrzi. A kapcsolatról és a csomagról megállapított tulajdonságok folyamatosan befolyásolják a kommunikációt.

A legtöbb „média streaming rendszer" támogatja a szabványos RTP, RTSP, illetve RTCP adatátvitelt, azonban ezek használata a jelenlegi internet infrastruktúrájában igen problémás, ezért a legtöbb esetben inkább hagyományos http protokollt alkalmaznak.

1.3.5. A HTTP protokoll

A HTTP (HyperText Transfer Protocol) protokoll [10] egy információátviteli szabvány. Szöveges és bináris erőforrások elérését teszi lehetővé URL azonosítók alapján. A kliens és a szerver közötti kommunikációt egyfajta kérdés-válasz alapján foghatjuk fel. Egyaránt használhatjuk valós idejű adatátvitelre, és egy egységnyi idő alatt történő letöltésére. Ha a hálózati sebesség alacsonyabb, mint a média adatátviteli sebessége, akkor a lejátszás megakadhat.

A HTTP protokoll szintű kommunikáció a következő lépésekből áll:

1. A kliens kezdeményezi a TCP kapcsolat létrejöttét. Általában a 80-as számú porton keresztül történik a kapcsolatépítés, de lehetőség van más portot is megadni.

2. A HTTP szöveges formátumú protokoll, a kérést szintén szöveges formában kell megadni. A kliens által elküldött kérésnek többféle típusa lehet, azonban a leggyakoribb mégis a GET és a POST. Ezek között a legfőbb különbség az, hogy a GET, egy erőforrás letöltését valósítja meg, a POST a kérés mellett pedig adatokat is juttathat a célállomásnak.

3. A következő lépésben a szerver elküldi a kliens által kért adatokat. A kérés sikerességéről a http csomag fejléce gondoskodik, mely tartalmaz egy háromjegyű számot, az ún. állapotkódot. A szám értékei fognak tájékoztatást adni, hogy a kérésekhez, milyen sikeres illetve sikertelen végrehajtások következtek be.

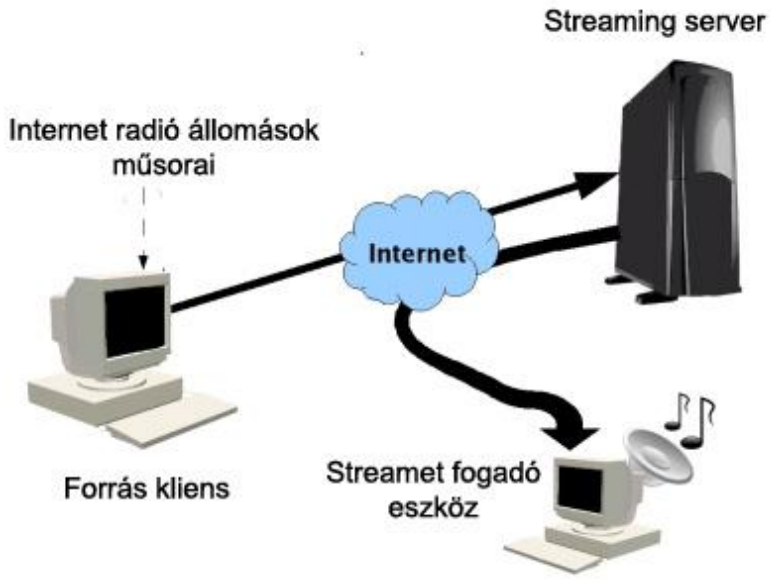
4. Az utolsó lépés a kapcsolatbontás, véget ér a tranzakció. Ez a művelet hamarabb is bekövetkezhet. Váratlan esemény hatására a kapcsolat a vártnál korábban megszűnhet, melynek kezelését a szervernek és a kliensnek is tudnia kell.

1.4. Webszervertől a Streaming szerverig

A webszerver önállóan képtelen az adatfolyamok valós időben történő kezelésére. Csupán csak azt a lehetőséget képes biztosítani, hogy egy adatot egy időben le tudjunk tölteni az adatbázisból.

A webszerverek http-protokollt használnak az interneten történő közlésre.

Az internetes rádióállomások esetében ezen protokollon keresztül összeköttetésben vannak a saját HTML oldalukkal (Hyper Text Marker Language). Megvalósulhat az adatátvitel, de a webszerver semmilyen vezérlést nem végez, így a szolgáltatás minősége csökken.



5. ábra – Streaming kommunikáció[22]

A streaming szerverek abban különböznek a webszerverektől, hogy saját adatküldési vezérlővel vannak ellátva. A különböző márkájú streaming szerver közül választhatunk, melyek működés alapvetően azonos, a zenei anyag közvetítésénél kell némi kompromisszumot kötni az egyes formátumok között. Az alapvető működésekről és beállításokról az egyes formátumok esetén a „Streaming audio: the FezGuys' guide”[11] szakirodalomból tájékozódtam.

A http-protokollt alapvetően nem valós idejű adatok lejátszására tervezték, hiszen a kezdeti elgondolás az volt, hogy a kliens elküld a szervernek egy kérést, ami feldolgozza azt. A kérésre összeállítja a választ, majd ezt visszaküldi a kliensnek. A kliens letölti a választ, és szintén feldolgozza az adatokat.

Azonban ez a művelet módosítható, ha például nagyméretű dokumentumot szeretnénk a továbbítani. A protokollt meg lehet változtatni olyan módon, hogy a kommunikációs lépések és a feldolgozás művelete átfedje egymást. Ennek hatására nem kell megvárni a teljes kérés megérkezését, annak folyamatában elkezdhetjük az adatfeldolgozást.

A kliens oldalon szükség van egy pufferre, mely a fogadott adatokat tárolja. Ha elegendő adat érkezett a pufferből kiolvashatók az adatok, és a lejátszás megkezdődhet.

A szerver folyamatosan figyeli, hogy a kliens mennyi adatot képes még fogadni, ellenőrzi az adatküldési folyamatot. Mivel a kliens előre puffereli az adatokat a lejátszás előtt, ezért csomagvesztés esetén az újraküldés következtében nem történik késlekedés, vagy kapcsolatbontás. Ezen információk birtokában vezérli a streaming szerver a bitsebességet.

Legtöbb esetben azonban nem csak egy szerver tárolja az összes fogható streamet, hanem több szerver található számos országban (Content Delivery Network). Minden szerver ugyanahhoz a honlaphoz kapcsolódik, és mindegyik ugyanazzal a tartalommal rendelkezik.

A kliens mindig a földrajzilag hozzá legközelebb eső szerverhez kapcsolódik. A szerverek terheltsége megoszlik, így egyidejűleg több kérést is végre tudnak hajtani, és több klienst képesek kezelni egyszerre.

1.5. A hangadat továbbítása

A média lejátszó kliensek igen sokfélék, minden platformon több különböző is megtalálható. Ezek leginkább a fogadott médiastreamekben térnek el egymástól. Általában minden gyártó a média-keretrendszer figyelembevételével készíti a saját kliensoldali megoldását, ezek bizonyos esetekben képesek egymás formátumaival együttműködni.

Az internetes rádiózás kialakulásánál a legelterjedtebb zenei formátum a WAV (Waveform Audio File Format) volt, amelyet a Microsoft fejlesztett ki. Tömörítés hiányában ez a megoldás rendkívül jó minőségű hangot eredményezett, azonban az interneten való műsorszóráshoz túl nagy fájlméretet jelentett. [12] A hanganyagokat azonban mindenképpen

tömöríteni kellett a folyamatos lejátszás biztosítása érdekében, hiszen a nagyméretű hangfájlokat lehetetlen oly módon továbbítani, hogy a lejátszás folyamatos legyen, ehhez mindenképp nagysebességű hálózat szükséges.

A tömörítő eljárások miatt kódoló eszközöket (kodek) kellett alkalmazni. A tömörítés választott mértéke mindig kompromisszum az átviteli minőség és az átviteli kapacitás-szükséglet között.

Hangot úgy érdemes tömörítetten tárolni, hogy azt kicsomagolási procedúra nélkül, ún. real-time programokkal meg lehessen hallgatni. Az eredmény minősége szempontjából kétféle tömörítést különböztethetünk meg: veszteségmentes és veszteséges tömörítést.

- Veszteségmentes tömörítés során az adatok mérete csökken, ám azok minősége nem változik, így kerülve el az adatvesztést.
- Veszteséges tömörítés folyamán információ vész el; a cél az, hogy az információvesztés minél kisebb minőségromlást eredményezzen.

A kodekek tartalmazzák a használatos tömörítő, és visszaállító algoritmust. A kodekek értelmezésének alapfeltétel, hogy mind a szerveren, mind a lejátszón ugyanarra a protokollra van szükség az adatok fogadásánál.

Különböző kódolási szabványok léteznek a dekódolásra, azonban az MPEG (Moving Picture Experts Group) mondható a legelterjedtebbnek. Az eljárás nem a hangállomány jellege szerint, hanem az ember által nem hallott hangelemek eltávolítása révén tömörít, ezért jórészt nem érzékelhető torzítást okoz. Ennek a kódolási formának köszönhetően majdnem CD minőségű hangot kapunk, a tömörítés veszteséges ugyan, de az adat mérete sokkal kisebb, amit már nagyobb hatékonysággal tudunk az interneten keresztül továbbítani.

Az MPEG Audio szabvány, illetve eljárás három réteg (Layer) szerinti tömörítés közötti választást tesz lehetővé. Az egyes rétegekben különböző tömörítési arányok mellett különböző hangminőség érhető el. [13]

- Az ún. Layer 1 a legegyszerűbb eljárás, mely 128 kb/s feletti bitsebesség esetén használható.
- A Layer 2 közepes bonyolultságú eljárás, mely 128 kb/s bitsebesség körül használható.

- A Layer 3 (az ún. MP3 eljárás) a legbonyolultabb eljárás, mely a legjobb hangminőséget csatornánként 64 kb/s bitsebesség mellett biztosítja.

Az audio adatok eljutása a szervertől a kliensig legegyszerűbben négy alaplépésben reprezentálható:

Első lépésként a küldendő adatot kell meghatározni, majd a dekódolni. Utána el kell juttatni a szerverhez, ami vezérli a valós idejű lejátszást. A következő lépés a továbbítás a csatornán keresztül (ez maga az internet, legyen a kapcsolat vezetékes vagy vezeték nélküli), ami kapcsolatot tart a szerver és a lejátszó között. Legutoljára már csak a lejátszás történik, amit a fogadó médium hajt végre.

2. Internetrádiók hallgatására szolgáló eszközök

Kezdetben az internet rádiók hallgatásról a számítógépre telepíthető szoftverek (például Winamp, Quick Time, Real Payer, Windows Media Player) tudtak egyedül gondoskodni. A szoftverek egy felhasználóbarát felületet biztosítanak a klienseknek, melyeket elindíthatunk akár a rádióállomások honlapjáról beépített felületként, vagy letölthetjük magát a rádióstreamet, és manuálisan adhatjuk hozzá a lejátszó-szoftverünkhöz.
Ma már nemcsak számítógépre telepíthetünk ilyen szoftvereket, hanem mobiltelefonokra, és más médialejátszókra is, akár többfélét, a különböző kódolások vétele miatt.

Külön kategóriát képviselnek a „számítógépnek álcázott végberendezések", az internetrádiók. Kinézetre a hagyományos rádióhoz hasonlítanak. Fő különbség azonban, hogy ezek a rádiók képesek csatlakozni az internethez, tehát nem az antennájukra érkező jeleket dolgozzák fel, hanem az interneten keresztül érkező adatcsomagokat. Ezeknek az eszközöknek köszönhetően nem szükséges a számítógép megléte, elegendő csupán az internetkapcsolat. Rengeteg internetrádiós készüléket lehet vásárolni, a legnépszerűbb gyártók: Roku Labs, Philips, Barix.

Egy egyszerű eszköz összeállításához elegendő csupán néhány követelménynek eleget tenni. A legfontosabb építőelemek, amik szükségesek egy internetrádió működéséhez:

- Szélessávú internet kapcsolat, ami lehet Ethernet (vezetékes hálózat) és/vagy Wi-Fi (vezeték nélküli) hálózati csatoló, TCP/IP támogatással.
- Szükséges egy stream, ami maga a rádióadás adatfolyama.
- Egy hardveres eszköz. mely rendelkezik egy dekódolóval, memóriával, vezérlőkkel és kijelzőkkel.
- Audiokimenet, ami lehet a rádió saját hangszórója, vagy a hozzá csatlakoztatható fejhallgató, hangfal.

A piacon sok, házilag fejleszthető internetes rádiópanel található, például:

- az elektor Internet rádió[3]
- Atmel evaluation kit AT91SAM[4]

[3] http://www.elektor.com/products/kits-modules/kits/071081-71-elektor-internet-radio.398061.lynkx

[4] http://microcontrollershop.com/product_info.php?cPath=154_170_270&products_id=2242

- Microchip Internet Radio Demonstration Board (6. ábra). A panel és az azt támogató programozó hardver- és szoftverrendszerek az egyetemen is rendelkezésre álltak, így magam is ezen eszköz fejlesztése mellett döntöttem.

6. ábra – Microchip Internet Radio Demonstration board[23]

3. A Microchip Internetrádiós demonstrációs panel tulajdonságai

Az eszköz egy 8 bites PIC18F67J60 mikrovezérlővel, beépített 10 BaseT MAC kártyával rendelkezik, mely 10 Mbps sebességű adatkapcsolatot tesz lehetővé. Egy PHY egység is található a kártyán, ez utóbbi a bitek kommunikációs csatornára való bocsátásáért felelős. [14] Az eszköz a magyar disztribútoroktól is megvásárolható.

A panel működés közben a SHOUTcast streamszerverekhez csatlakozik, és az onnan érkező MP3 adatfolyamot az audiodekóderhez továbbítja. Ez a demonstrációs kártya az internetrádió alaptulajdonságait szemlélteti, mint a hangerő-szabályozás vagy a csatornaváltás. A rádió és a hozzátartozó gyári kód a Microchip cég tulajdona: Howard Schlunder és Elliot Wood keze munkája. A panel megvásárlása után a projekt szabadon fejleszthető.

A panel „lelke” a PIC18F67J60 mikrokontroller, ez vezérli az összes munkafolyamatot. Felveszi a kapcsolatot a szerverrel, továbbítja az adatot az MP3 dekódernek, megjeleníti az állapotokat a kijelzőn, továbbá végrehajtja a felhasználó által megadott műveleteket.

Device	Flash Program Memory (bytes)	SRAM Data Memory (bytes)	Ethernet TX/RX Buffer (bytes)	I/O	10-Bit A/D (ch)	CCP/ ECCP	MSSP		EUSART	Comparators	Timers 8/16-Bit	PSP	External Memory Bus
							SPI	Master I^2C™					
PIC18F67J60	128K	3808	8192	39	11	2/3	1	Y	Y	1	2	2/3	N	N

A mikrokontroller főbb jellemzői

3.1. A panelhez tartozó gyári kód

A panelhez tartozott egy előre beállított gyári kód, az Internet Radio App nevű alkalmazás, melynek legfrissebb verziója jelenleg is elérhető a Michrochip hivatalos weboldalán[5], egy hozzá tartozó melléklettel, ami azonban nem a rendelkezésre álló verzióhoz készült. [15] A működéshez a meglévő programkódot a panel által is felismerhető gépi kóddá kell alakítani, majd fel kell tölteni a kártya programmemóriájába.
A kiadott szoftverrendszer tesztelése során megállapítottam, hogy a legújabb verziójú (jelen esetben az 5.36.4-es) programkód hibásan működik. Kénytelen voltam a hiba jelentése mellett

[5] www.microchip.com

régebben kiadott alapkódot kérni a gyártótól, amit elérhetővé tettek. Ezen döntésem okát a későbbiekben fejtem ki. A továbbiakban tárgyalt szoftverre vonatkozó leírások mind a korábbi verzióra terjednek ki.

Fontos eleme az internet rádiónak a „TCP/IP stack és firmware". A stack is ingyenesen letölthető a www.microchip.com/tcpip oldalról egy hozzátartozó leírással is, melyet forrásként is felhasználtam a később tárgyalt működések részletezésénél. [16] Ez a programkód-csomag hozzátartozik az alapkódhoz, azonban nem tekinthető vele egy egységként. A stack más, internet alapú alkalmazásokhoz is felhasználható: pl. a Microchip nyílt forráskódú, és az internetrádióhoz hasonlóan szabadon fejleszthető hálózati forgalom figyelőjéhez, vagy éppen a vezeték nélküli hálózatok kezelését, és fejlesztését végző projektekhez.
Az internetrádió esetében ennek segítségével épül fel a kapcsolat az audioszerverrel, és ez továbbítja az audiostreamet az MP3 dekódernek. Kezeli a küldött és fogadott TCP és UDP csomagokat. A stack által támogatott protokollok: ARP, IP, ICMP, UDP, TCP, DHCP, SNMP, HTTP, FTP, TFTP.

Minden Internet Radio Demonstration Board egy előre programozott, egyedi Ethernet MAC címmel kerül forgalomban, mely a PIC18F67J60 Flash programmemóriájában kerül eltárolásra. Ez az egyedi cím a kártyán lévő matricán is megtalálható. Ennek segítségével azonosíthatjuk az eszközt a hálózaton. A Microchip azonosító száma a 00-04-A3. A következő hat számjegy pedig szabadon választható.

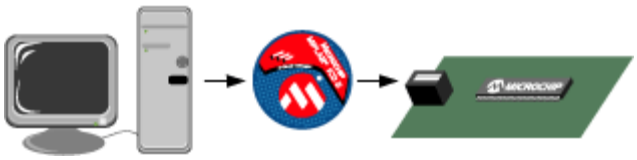
3.2. Az internet rádió programozásához szükséges eszközök

A panel megléte nem elegendő a programozáshoz és teszteléshez, ezért szükség volt egyéb eszközökre is:

- MPLAB ICD 2 (In Circuit Debugger) [14]

  Ez nem más, mint egy hardveres hibakereső és programozó fejlesztőrendszer, melyet szintén a Microchip fejlesztette ki. Segítségével szinte valós működés közben tesztelhetjük a megírt programjainkat.

  Másik fontos funkciója, hogy általa a megírt programkódot betölthetjük a mikrovezérlőbe. Magából a fejlesztő rendszerből is kiadhatjuk a felprogramozás parancsot. A számítógéphez nagysebességű USB 2.0 porton keresztül csatlakozik.



7. ábra – MPLAB ICD 2 kapcsolódási ábrája[24]

- MPLAB IDE[6]

  Integrált fejlesztőkörnyezet (IDE - Integrated Development Environment), mely a mikrokontroller fejlesztéshez, programozásához szükséges szoftverállományt biztosítja. Segítségével a mikrochip által felismerhető gépi kóddá alakíthatjuk a programunkat, melyet egyből betölthetünk a memóriába. Használatáról a hozzá tartozó felhasználói útmutató ad tájékoztatást.

  Az MPLAB tartalmaz egy szövegszerkesztőt is a programok megírásához. Ezenkívül egy MPASM nevű assemblert (a programok lefordításához és hibakereséshez), egy MPSIM nevű szimulátort (a programok teszteléséhez), valamint kezel többféle programozót és debuggert, többek között a fentebb említett ICD2-t.

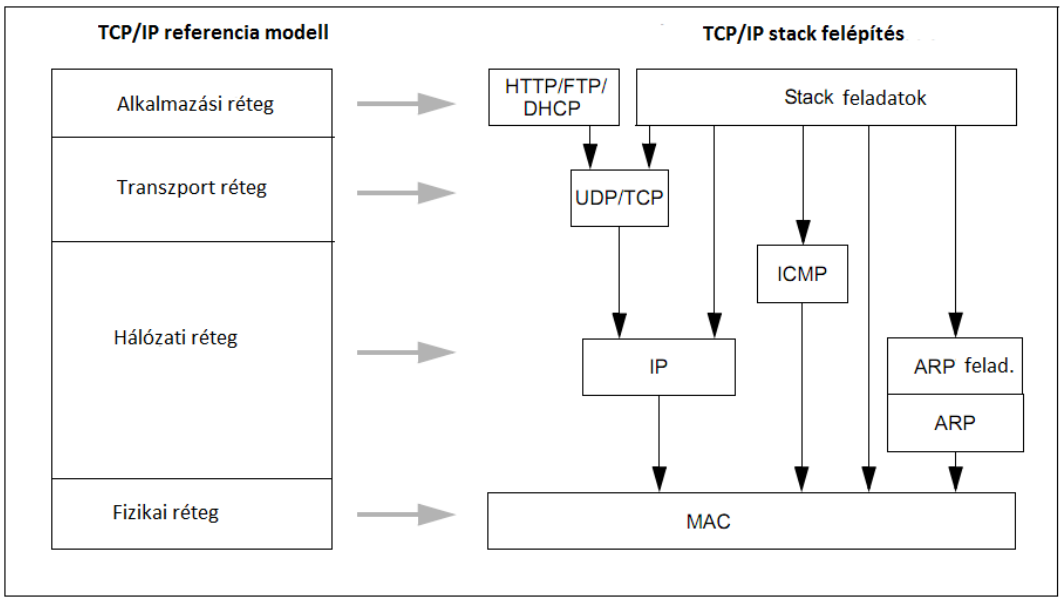
  Ahhoz, hogy a programkódot hatékonyan le tudjam fordítani, az MPLAB fejlesztőkörnyezetbe be kellett integrálnom a C18-as fordítót. Ez egy programozó egység, amely lefordítja az adott kódot. Ez a verzió kimondottan a 8 bites mikrovezérlőkhöz használatos.

3.2.1. A TCP/IP stack és Protokollok

A fejlesztés kezdetén elengedhetetlen volt, hogy mélyebben is tanulmányozzam a kiadott TCP/IP stack firmware felépítését, hogy hogyan is történik az adatátvitel és a dekódolás. Ez a verem a TCP/IP referenciamodell minden egyes rétegét lefedi:
A rétegek feladata nem csak az adatok szolgáltatása, hanem a nem várt események kezelése is, mint például az adatcsomag vesztés, vagy az új kapcsolat felépítés esetén esetlegesen fellépő időtúllépés.

A stack C nyelven íródott, és támogatja az internet rádió működéséhez szükséges legfőbb protokollokat.

[6] MPLAB IDE [Online] – Elérhető: http://www.microchip.com/



8. ábra – TCP/IP Stack protokollok és a TCP/IP modell kapcsolata [25]

3.2.2. Protokollok feladata

Korábban már említést tettem az alapvető protokollszabványok tulajdonságairól és létjogosultságáról az internetrádiós alkalmazások körében, azonban a tárgyalt szoftverrendszer felépítésének magyarázatához kiegészíteném a témakört, feladatbeli pontosításokkal.

A Microchip internetes rádió képes kezelni a TCP/IP szabvány által nyújtott protokollokat. A küldött és fogadott csomagokat a TCP/IP stack kezeli. A stack különböző rétegekből áll, amelyek egymás felett helyezkednek el, és mindegyik réteg az alatta lévő szolgáltatásait használja. Az információ a legfelső rétegtől a legalsóig mindegyik rétegen keresztül halad, innen ered a verem elnevezés is.

A TCP protokoll feladata, hogy továbbítsa a kapott MP3- és metaadatokat, miközben szabályozza a küldési folyamatot, hogy a SHOUTcast szerver semmi esetben se küldjön több adatot, mint amennyit az internetrádió RAM-ja fogadni képes.

A DHCP protokoll szolgáltatja a rádió IP címét, az átjáró címét, az alhálózati maszkot és egyéb konfigurációs paramétereket abban a pillanatban, mikor a hálózatra való kapcsolódás megkezdődik.
Ennek alapján fogja a netrádiónk tudni, hogy hogyan csatlakozzon az internethez.

A DNS protokoll alapján ismeri fel rádiónk a rádiócsatornákat. A DNS abban az esetben használatos, mikor a felhasználó csatornát vált. Ekkor ugyanis a rádió csatorna host nevét (például scfire-dll-aa02.stream.aol.com) konvertálja dinamikus IP címmé (például: 149.174.134.200), melyhez kapcsolódni próbál.

Az IP protokoll feladata, hogy mind a TCP, mind az UDP csomagokat továbbítsa a megfelelő célállomásra. Mielőtt azonban az átvitel elkezdődne, a Stack ARP protokollt használ az Ethernet MAC címének (ami az én eszközöm esetében: 00:04:A3:01:04:C0), és a hozzá tartozó, helyi internet-átjáró címének megállapításához. Ezzel a számmal valósul meg az eszköz hálózaton történő azonosítása.

A HTTP (HyperText Transfer Protocol) protokoll fogja szállítani a fogadott és küldött TCP vagy UDP csomagokat a szerver és a kliens között. Feladata még, hogy az internetrádió webes felülete és az eszköz közti kommunikációs üzeneteket továbbítsa. Kérés-válasz protokollnak is hívják.

A protokollok párhuzamosan dolgoznak, hogy létrehozzák a kapcsolatot a rádió szerverrel. A processzorban futó folyamatok önként adják át a vezérlést egymásnak. Mindez a háttérben folyik, nem szükséges semmilyen manuális, felhasználói konfiguráció vagy beavatkozás.

3.3. A szerver és az internetrádió

A kapcsolat felépítése után szükség van egy forrásra, ami majd a hanganyagot szolgáltatja. Az internetrádió elsősorban a SHOUTcast protokolltól kapja a műsort.
A SHOUTcast[7] egy kliensszerver-modell, melynek tagjai a hálózaton kommunikálnak egymással. A protokoll segítségével az audioadatokon kívül más metaadatokat, a hangadaton kívüli, a fájl tulajdonságaira vonatkozó információkat is képesek vagyunk küldeni, mint pl. a dal címe, típusa vagy éppen az állomás neve.

A szerver a küldéshez http-protokollt alkalmaz. A SHOUTcast kommunikáció MP3, illetve AAC (Advanced Audio Coding) dekódolást használ. A kettő közötti legfőbb különbség a bitsebességükben keresendő.

[7] www.shoutcast.com

Az AAC kódolást kisebb bitsebességű rendszereknél alkalmazzák, hatékonyabb tömörítést végezhetünk el vele, illetve csökkenthető a mintavételezési frekvencia. Jelen esetben azonban az MP3 kódolás a megfelelő, mivel az általam kiválasztott internetrádiós eszköz csak ezt fájlkódolást képes kezelni.

„Az MP3 egy veszteséges tömörítésen alapuló, zenei anyag tárolására használt fájlformátum, jelenleg az egyik legelterjedtebb. Valójában két különböző, de nagyon hasonló formátum, az MPEG–1 Audio Layer 3 és az MPEG–2 Audio Layer 3 összefoglaló neve.” [17]

A http-protokoll továbbítja a szervertől kapott fájlokat a kliensnek. A szerver a www.shoutcast.com honlapról érhető el. Ahhoz, hogy a mikrokontrollerünk képes legyen kapcsolódni a hosthoz, küldenünk kell egy üzenetet, melynek tartalmaznia kell a kapcsolathoz szükséges minden adatot. Ennek hiánya esetén a csatlakozás időtúllépés miatt nem fog létrejönni.

A helyes működéshez a rádióadókat előre definiálni kell a megfelelő paraméterekkel, melyeket egyenként egy tömbben tárolok el. Mint már korábban említettem, az internetrádió legfrissebb verziójú szoftvere hibásan működik.

Ennek egyik oka az, hogy rosszul van definiálva a szervernek elküldött kérés. Az alapkód a Shoutcast szerver protokolljának egy korábbi verziójához íródott, ahol a csatornákat típus szerint lehetett szelektálni. A rádió ennek megfelelően a típusra vonatkozó kérést küld el, és ennek megfelelően tárolja az adásokat. A kérést viszont a szerver a jelen konfigurációkkal nem képes értelmezni.

A hibás kérés:

Előre definiált változók:

Host:

#define SHOUTCAST_HOST "shoutcast.com

Útvonal:

#define SHOUTCAST_PATH "/sbin/newxml.phtml"

Típus:

#define SHOUTCAST_PATH_GENRE SHOUTCAST_PATH "?genre="

Adatfolyam elérési helye:

#define SHOUTCAST_PATH_STREAM "/sbin/tunein-station.pls?id="

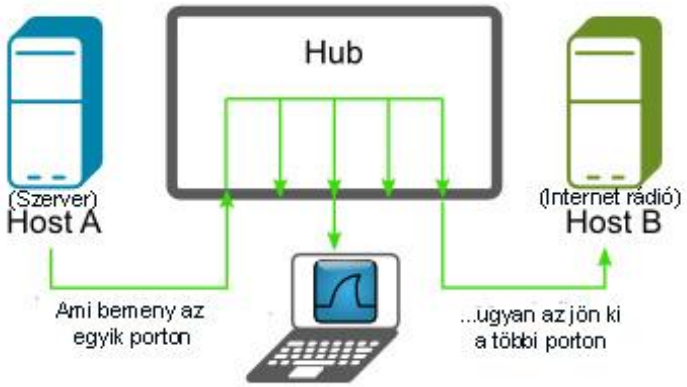
Port:

#define SHOUTCAST_PORT 80u

A kérés elküldése:

```
TCPPutROMString( MySocket, (ROM BYTE*)"GET " );
TCPPutROMString( MySocket, (ROM BYTE*)SHOUTCAST_PATH_GENRE );
TCPPutROMString( MySocket, (ROM BYTE*)(genres[genre].ShoutcastName) );
TCPPutROMString( MySocket, (ROM BYTE*)" HTTP/1.0\r\nHost: " );
TCPPutString   ( MySocket, (BYTE *)shoutcastHost );
TCPPutROMString( MySocket, (ROM BYTE*)"\r\nConnection: close\r\n\r\n" );
TCPFlush(MySocket);
```

Ahhoz, hogy ellenőrizni tudjam a hálózaton végbemenő adatforgalmat, a Wireshark nevű programmal követtem a kommunikációt internet rádiós eszköz és a szerver között.



9. ábra – Mérőállomás blokkdiagramja[26]

Felállítottam az 9. ábrán található mérőállomást. Egy közös Hubon keresztül figyeltem a küldött és fogadott csomagokat. Hálózati eszközök közös kapcsolódási pontját, a Hubot tipikusan helyi hálózatokban használják számítógépek és más eszközök összekötésére. A blokkdiagrammon a Hubot egy egyszerű passzív eszköznek feleltettem meg. Összekötöttem a saját számítógépemmel és az internetrádióval.

A Hubok azon tulajdonságát használtam ki, amely során az eszköz minden kimeneti portjára kimásolja a fogadott csomagot, így minden eszköz ugyan azt az adatcsomagot kapja meg. Ennek következtében a saját számítógépemre is megkaptam ugyan azokat az adatcsomagokat, amelyeket az internet rádiós eszköz is.

A Wireshark alkalmazás a protokollokon keresztül vizsgálja a hálózati forgalmat. A válaszok ellenőrzéséhez elegendő volt a TCP csomagok tartalmának elemzése. A 10. ábrán a 5.36.4-es verzió kérés-válasz eredménye látható.

```
HTTP/1.1 200 OK
X-DB-Timeout: 120
Pragma: no-cache
Cache-Control: no-cache
Content-Type: text/plain
Date: Wed, 04 May 2011 09:39:23 GMT
Content-Length: 15

{"ret": "punt"}GET /subscribe?
host_int=45246535&ns_map=23418562_39307564111554,34400911_42984073871&ts=1304501980
HTTP/1.1
Host: stork52.dropbox.com
Accept-Encoding: identity
Connection: keep-alive
```

10. ábra – A kapott TCP csomag tartalma

A mérésből kiderült, hogy a kapcsolat felépül, azonban már a kérésre hibás válasz érkezik. A típus szerinti lekérdezésnél a szerver nem találja a megfelelő adatot, melyre a kérés vonatkozik, amitől a rádió pufferébe nem fognak zenei adatok kerülni.
A legfrissebb alapkód egésze a típus szerinti tárolásra és feldolgozásra épül. Az teljes TCP/IP stack és az MP3Client.c adatfeldolgozással foglalkozó programrészlet is hibásan, és rossz sorrendben dolgozza fel a megkapott a metaadatokat. A helyes működés megvalósításához nem elég a kérés módosítása, a működő eredmény elérése az egész mintakód újraírását igényelné.

Ekkor jutottam arra a következtetésre, hogy egy korábbi verziót könnyebb újra konfigurálni, mivel az alapfeladatokat, a csomagküldést, -fogadást és egyéb kommunikációs lépéseket a régebbi verzióknak is képesnek kell lenniük kezelni, s nekem csak a hiányzó funkciókat kellene pótolnom.

A korábban kiadott alapkódokat már egyáltalán nem lehet elérni, így a gyártótól kellett külön kérnem ezeket a verziókat.
Miután rendelkezésemre állt néhány régebbi kiadás, hosszas tesztelés és elemzés után az Internet Radio App 4.55-ös verziójának kijavítása tűnt a legjobb megoldásnak, mivel ebben már szerepeltek a protokollokhoz tartozó kezelő függvények.

Az új kódfájlnak megfelelően a következő kódrészlet már egy helyes csatornadefiniálást demonstrál:

```
STATION_INFO stations[] =
    {
            "SKY.FM Country, 128K",
            "scfire-ntc-aa02.stream.aol.com",
            80,
            "GET /stream/1019 HTTP/1.0\r\n"
            "Host: scfire-ntc-aa02.stream.aol.com\r\n"
            "Accept: */*\r\n"
            "Icy-MetaData:1\r\n"
            "Connection: close\r\n\r\n"
    }
```

Egy hagyományos tömbben tárolom el az előre beállított adókat, melyek paramétereit hagyományos stringként (szövegként) definiáltam. Először a rádióállomás nevét kell megadni, majd a honlapot, amelyről letölthető az adatfolyam. A harmadik paraméter mindenképp a port száma kell, hogy legyen (jelent esetben ez 80).

Utána, s talán ez a legfontosabb lépés, meg kell adni a kérést, amit a http-protokoll fog elküldeni a szervernek, ami a megfelelő adatok birtokában nyugtázza azt. El kell küldeni egy, a metaadatokra vonatkozó kérést is, így jelezzük a szervernek, hogy nem csak az audio adatokra van szükségünk. Engedélyezzük ezen adatok fogadását, az értékét „1-re” állítjuk.
Ez nem egy szükséges lépés, de az egyéb adatok a felhasználó tájékoztatása miatt szükségesek. Ha minden rendben történik, lezárjuk a kapcsolatot, jelezzük a szervernek, hogy szüntesse meg a kapcsolatot a klienssel.

Helyes kérésküldés esetén a következő adatforgalom figyelhető meg a hálózaton:

```
GET /stream/1019 HTTP/1.0
Host: scfire-ntc-aa02.stream.aol.com
Accept: */*
Icy-MetaData:1
Connection: close

ICY 200 OK
icy-notice1: <BR>This stream requires <a href="http://www.winamp.com/">Winamp</a><BR>
icy-notice2: Firehose Ultravox/SHOUTcast Relay Server/Linux v2.6.0<BR>
icy-name: Country - S K Y . F M - today's Hit Country with a mix of your favorites
icy-genre: Country New Country Top 40
icy-url: http://www.sky.fm/country/
content-type: audio/mpeg
icy-pub: 1
icy-metaint: 16384
icy-br: 96
```

11. ábra – A kapott TCP csomag tartalma

Ebben az esetben a szerver felismerte a kérést és elküldte a kért adatokat. Az icy tag az Icecast protokollt jelöli. Ez a protokoll az egyik legelterjedtebb a streamszerverek között, amikor MP3 adatot akarunk továbbítani http-protokollon keresztül.

- icy-name: a rádiócsatorna neve
- icy-notice: a rádiócsatornához tartozó egyéb megjegyzések
- icy-genre: a rádiócsatorna típusa
- icy-metaint: a metaadatok időtartama, hogy mikor érkeznek meg az audio streammel; a szoftver később külön tudja választani a őket a hasznos hanganyagtól.
- icy-br: az audio folyam bit sebesség kb/s-ban

A fogadott adatok a rádió memóriájában kerülnek eltárolásra. A programkódban stations[] tömb fogja eltárolni az általunk beprogramozott csatornák adatait. Csatorna váltás esetén ezen a tömbön fogunk végig lépkedni, és a megfelelő adatokat továbbítani.
A kérés elküldése után helyben definiált változók fogják eltárolni a beállított adatokat, külön az audio-, és külön az egyéb metaadatokat, úgy mint a zeneszám címe, a kódolás (audio/mpeg), vagy éppen az adat hossza.

BYTE strStationName;
BYTE strSongName;
BYTE lenStationName, lenSongName;
BYTE tPtr, gPtr;
BYTE bitrate;

Ezek a változók minden egyes csatornaváltás után új értéket kapnak. Ezeket az adatokat könnyedén elérhetjük, ezáltal értékük bármikor kiküldhetők a rádió kijelzőjére.

Az adatok feldolgozását az MP3Client.c programkód végzi. Ez tartja a kapcsolatot a TCP/IP stack fogadott TCP csomagjaival. A stack csak az adatok biztonságos, és veszteségmentes megérkezését biztosítja, azzal már nem foglalkozik, hogy szortírozza is azokat aszerint, hogy melyik információ hasznos a számunkra.

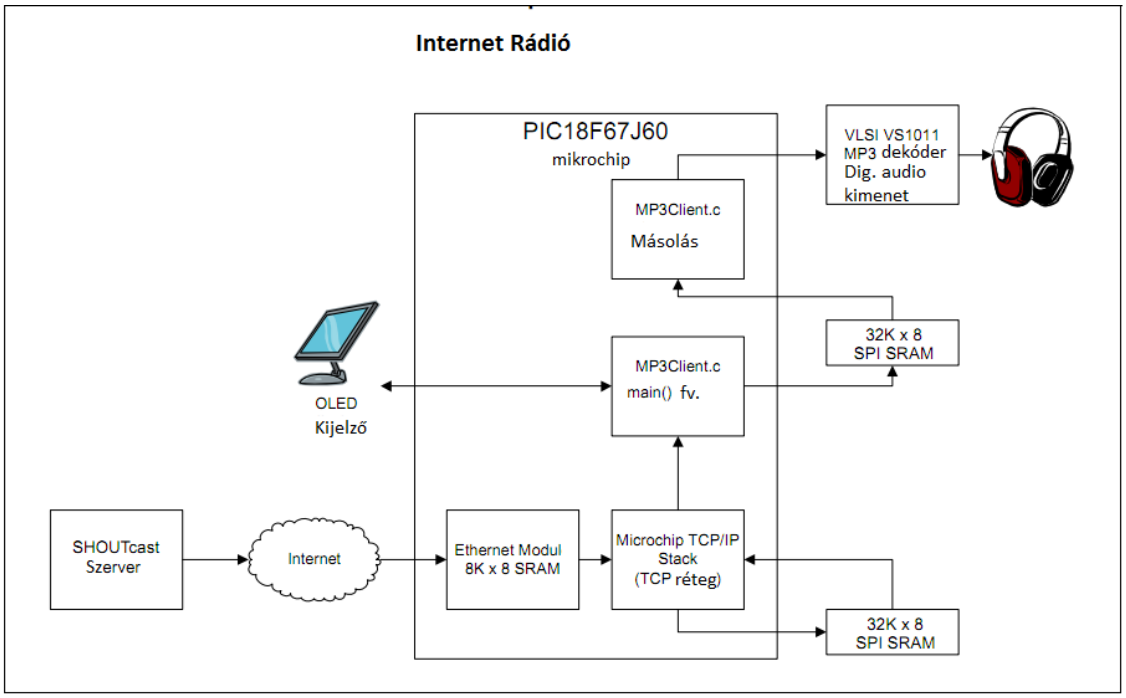
Az MP3Client.c fájl a TCP csomagokból kinyeri a megfelelő információkat. Levágja a ”\r\n\r\n„fejlécsorokat, ezek alapján már tudja azonosítani, hogy a kapcsolat felépítéséhez mely adatokra van szükség.

A fogadott „icy” adatokból tudja kinyerni az audioadatokat, illetve innentől már meg tudja különböztetni, melyek a kapcsolódási, és melyek a rádióadásra vonatkozó információk.
Mivel az netrádió nem rendelkezik túl nagy memóriával, ezért folyamatosan figyelni kell a fogadott adatok mennyiségét és a szabad helyet, melyet szintén a fent tárgyalt programkód végez el. Ha nincs elég szabad hely, a bitsebességet kell módosítani, hogy az adatok lassabban érkezzenek a feldolgozáshoz.

3.4. Az audiostream dekódolása

A TCP protokoll szolgáltatja az MP3 adatot. Itt kerül előtérbe a TCP protokoll egyik legfőbb tulajdonsága, miszerint minden csomagra nyugtát vár, nyugtát továbbít. Csomagvesztés vagy hibás csomag esetén újra küldi az egész csomagot, aminek következtében a csomagok késhetnek, ezáltal a várakozási idő viszont is megnő.
Nagyobb memóriapuffer szükséges, mely lehetővé teszi, hogy elegendő adatot tudjunk eltárolni ahhoz, hogy folyamatos legyen a lejátszás. Ez okból található a nyomtatott áramkörön 2 külső SRAM (Serial Ram), amely további 64 kbyte-ot szolgáltat, 32-t a TCP rétegnek és 32-t az audiopufferéléshez. Az elsőként fogadott adat kerül legelőször továbbításra, a kiolvasás sorrendtartó.

Az alábbi ábra mutatja az adatfolyam útját a panelen:



12. ábra – A hanganyag eldolgozásának blokkdiagramja[23]

A TCP réteg tárolja el az összes bejövő adatot, az MP3 adatfolyamot a beágyazott csatorna nevével, típussal és egyéb metaadatokkal. A TCP, IP és Ethernet fejléceket ez a réteg már nem kapja meg, mert az Ethernet Modul SRAM ezeket továbbítás előtt levágja.

A TCP protokoll kommunikál a SPI SRAM-mal (SPI (Serial Peripheral Interface), mely egy nagy sebességű soros szinkron I/O rendszer. Az SPI alkalmas egy feldolgozó egység és annak kiegészítő áramköreinek összekapcsolására. Az MP3Client.c fájl a TCP adatok alapján figyeli, hogy mennyi a szabad hely van még az adatok számára. Ezen információ birtokában kommunikál a SHOUTcast szerverrel, hogy az mennyi adatot küldjön, megakadályozva a túlcsordulást és a csomagvesztést.

A programban szintén az MP3Client.c programfájl feladata, hogy periodikusán másolja át a TCP pufferből az adatokat a második 32Kx8 SRAM chipbe. Másolás közben levágja a zeneszám címét és más metaadatokat, melyeket a kijelzőre küld ki. Ez azért fontos művelet, mert a dekódernek már nem kell adatmanipulálási feladatot végeznie, ezáltal felgyorsul a dekódolás művelete is. Tehát a maradék nyers adatot a dekódernek továbbítjuk, ami csak akkor lehetséges, ha a megfelelő mennyiségű adat van pufferelve.

Fontos, hogy a folyamból ki kell vágni a metaadatokat, hogy csak az audioadatok maradjanak, ellenkező esetben a dekóder azokat az adatokat is tömörített hanganyagként ismeri fel és megpróbálja dekódolni, amelynek eredménye azonban hibás audiokimenet lenne. A kódolást a panelen megtalálható e VS1011 MPEG Audio Kodek végzi.

A chip már előre konfigurálva van, így nekem csak az volt a dolgom, hogy megfelelő adatokat biztosítsak a számára. A dekóder 32 bájtnyi adatot képes fogadni. Egy változón (DREQ – Design Requirements) keresztül jelzi a SPI vezérlőjének, hogy mikor van szüksége új adatokra.
A hangadat manipulálását a VS1011 chip végzi. Azonban a közvetlen parancsot nem a dekóder kapja, hanem az SPI folyamatvezérlő. Segítségével képesek vagyunk a hangerő változtatására, melyeket a panelen található nyomógombokkal vezérelhetünk. A hangerő-szabályozás nem befolyásolja az adatközvetítést, a két művelet teljesen elkülönül egymástól.
A hangerőt a dekóderben található regiszter tárolja el, melynek maximális értéke 254, minimális 0. A hangerő-szabályozó funkciónál tehát elegendő ennek az értékét megváltoztatni.

A hangerőszabályzó funkciót a következő kódrészlet látja el:

```
void SetVolume(BYTE vRight, BYTE vLeft)
{
        MP3_XDCS_IO = 1;
        while(!MP3_DREQ_IO);
        MP3_XCS_IO = 0;
        WriteSPI(0x02);
        WriteSPI(0xB);
        WriteSPI(vLeft);
        WriteSPI(vRight);
        MP3_XCS_IO = 1;
}
```

A függvény bemeneti paraméterként megkapja a jobb és a bal csatorna hangerősségének adatait, a fentebb megállapított intervallumok között. A WriteSPI(0xB) a regiszter címét jelenti, a Write funkciókkal beleírjuk a megadott adatokat. Az internetrádió dekóderje ennek függvényében állítja be a megadott hangerőt.
A fizikai vezérlés alapvető funkcióinak beprogramozásával így elkészültem, azonban hátra volt még egy másodlagos vezérlési mód megvalósítása is.

3.5. A webes felület és az internetrádió kapcsolata

Feladataim közé tartozott egy új vezérlési megoldás megvalósítása is, mely szerint az internetrádió vezérlését kiterjeszthetném egy internetes felületre.
Mivel a rádió eleve figyeli az internetforgalmat, miért ne figyelhetné egy saját honlapról érkező vezérlési műveleteket is – tettem fel a kérdést.
A programozás megkezdése előtt mindenképp szükség volt ennek a honlapnak a megírására. Egy egyszerű webes felületet hoztam létre, szabványos HTML (HyperText Markup Language) kóddal. Azért sem volt szabad bonyolult megoldásokkal foglalkoznom, mert magát a weboldalt be kellett tölteni az internetrádió flash-memóriájába. Túl nagy fájl esetén nem marad elég hely az internetrádión, esetleg maga az oldal sem fér rá.

A rádióhoz tartozó weblapot dinamikusan tudjuk kezelni egy CGI-n (Common Gateway Interface) keresztül. A CGI elérhetővé teszi alkalmazásaink számára a hálózaton keresztüli

elérhetőséget. A kommunikáció a HTML-oldalakon keresztül történik, ennek következtében a programunkat távolról is vezérelhetjük. Egy statikus weblaphoz képest dinamikus információkat jeleníthetünk meg, melyek legtöbb esetben a kliens lekéréseitől függnek.

A CGI nem egy programnyelv, a CGI csak egy felület, amin keresztül a programunk a http-szerverrel kommunikál.

Az eszköz és a weboldal közötti kommunikáció felépítéséhez először magát a honlap tartalmát kellett a mikrovezérlő által felismerhető gépi kóddá alakítani úgy, hogy be lehessen tölteni a panel memóriájába. Ahhoz, hogy a memóriában ezeket az adatokat hatékonyan tudjuk eltárolni, ki kell alakítani egy nem túl bonyolult fájlrendszert. Ehhez azonban szükség volt a feladatot elvégző külső programra.
Az MPSF.exe nevű program hozza létre az adott gépi kódot, a paraméterként beállított weboldalból. A program futtatásával egy generált MPFSImg2.c és egy MPFSImg2.s fájl keletkezik, melyeket a program automatikusan hozzácsatol az Internet Radio App projekthez. Ezek tartalmazzák a chip által értelmezhető webtartalmakat.

A webes felület a rádiós eszköz második vezérlőjeként kell, hogy funkcionáljon. Minimálisan tehát minden olyan funkció ellátására képesnek kell lennie, amit magán a fizikai eszközön is végre tudunk hajtani. Legfontosabb műveletnek a csatornaválasztás és hangerő-szabályozás megvalósítását tűztem ki célul. A tervezésnél az egyszerűséget tartottam szem előtt, nehogy túllépjem a megengedett memóriaterületet.

A megvalósítás során a lentebb látható felületet dolgoztam ki:

13. ábra – Az általam készített weboldal

A rádió saját IP címmel rendelkezik a hálózaton, ezért elegendő beírni a kijelzőjén mutatott címet, hogy elérjük az eszköz saját honlapját. A honlapon az egyes vezérlőkhöz (jelen esetben gombokhoz) egy adatküldési funkciót kellett rendelni.

Az eseménykezelő gombok esetén a funkciókat egy „űrlapként”(form) kezelem. Ennek az a jelentősége, hogy az űrlapok különböző adatközlő mezőket tartalmazhatnak. Jelen esetben az elküldendő adat a gomblenyomás esetén a gombnak a neve (például „btnVolDown”). Ezek a változók éppen attól függnek, hogy milyen értéket deklaráltam az adott gombhoz, tehát minden gomblenyomás esetén ezt a módosított változót kapja meg az eszközünk.

Példa a honlapon megvalósított vezérlőre:

```
<form method="get" action="index.htm">
<td><input name="btnVolDown" type="submit" value="Fel"></td>
<td><input name="btnVolUp" type="submit" value="Le "></td>
<td><input name="mute" type="submit" value="Elnémít"></td>
</form>
```

„A kérdőívet definiáló <FORM> tag ACTION paraméteréhez beírhatjuk a CGI program nevét. Ilyenkor a kérdőív adatait a megjelenítő elküldi a szervernek, pontosabban az azon futó CGI programnak, ami elvégzi a feldolgozást és esetleg egy válaszlapot küld vissza a megjelenítőnek.” [18]

A GET-metódus a küldött adatokat az URL-hez kapcsoltan továbbítja. Az adatok név-érték párokba rendezve kerülnek át a szerverre.
Az adatfogadás és küldés ugyan úgy működik, mint maga a szerver és a rádió között, csak itt a weboldal és az eszköz között történik az információáramlás.
A HTML-lapokon elhelyezhetünk olyan utasításokat is, amiket a szerver végrehajt, mielőtt elküldené a lapot a kliensnek. Ezen utasítások egyikével CGI programokat hajthatunk végre, amelyek kimenete ilyenkor a HTML lapra, az őket hívó utasítás helyére kerül.

A honlapról érkező adatokkal a szoftverben már nem az MP3Client.c – mint a szerver esetében –, hanem másik programrészlet foglalkozik, nehogy összekeveredjenek a szervertől, illetve a honlaptól érkező adatok. A TCP/IP stackban található HTTP2.c és a

CustonHTTPApp.c fájl dolgozza fel a weboldalról érkező információkat. Műveletei akkor kerülnek meghívásra, ha az eszköz saját IP címéről érkező adatokat olvas.

Amint a felhasználó az egyik vezérlőgombra kattint, a rádió http-kérést fogad a bemenetén, ami jelzi, hogy változás történt a weboldalon.
A fogadott adatokat egymás után a program által generált curHTTP.file fájl tárolja el szövegpáronként. A szövegpárok egy „value=érték” párként értelmezhetőek. (Például: Value=btnDown). Ezeket a szövegpárokat fogja a program megkapni, melyeket egyenként értelmez.

A vezérlők hatására meghívódik a „HTTP_IO_RESULT HTTPExecuteGet(void)” függvény, mely a szövegpárokon végez ellenőrzést. A HTTPGetArg és a HTTPGetROMARG függvény teszi lehetővé, hogy fogadott értékeken összehasonlító ellenőrzést végezzen.
Ehhez természetesen az szükséges, hogy az összes lehetséges értékhez egy megfelelő funkciót rendeljünk, melyet a következő példa szerint valósítottam meg.

Egy példa az egyik funkcióra:

```
if(HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"btnVolDown"))
    {
  if(RadioConfig.vVolumeRight < 240u)
    RadioConfig.vVolumeRight += 5;
  if(RadioConfig.vVolumeLeft < 240u)
    RadioConfig.vVolumeLeft += 5;
  SetVolume(RadioConfig.vVolumeRight, RadioConfig.vVolumeLeft);
  }
```

Abban az esetben, ha a beérkezett adat a „btnVolDown” értékkel egyenlő, lefut a programrészlet: ha a hangerő nincs a legalacsonyabb fokozaton, akkor lejjebb veszi a hangerőt.

Tulajdonképpen azokat a funkciókat kellett újból megvalósítani, amelyekre a rádió másfajta vezérlési módszerrel ugyan, de már korábban is képes volt. Új némítási funkcióval is kibővítettem a webes eszközkészletet, mely nem tartozott a hardveres eszköz korábbi jellemzői közé.

További opcióként lehetséges a MAC cím beállítása. A címet 12 darab hexadecimális számként kell megadni a honlapon található bemeneti mezőbe, majd elmenteni a megadott értéket. Az első 6 karaktertől eltekintve szabadon konfigurálható.

Ha a honlap új beállítást érzékel, azonnal újrafrissül, a már új adatokkal. A HTTPGetROMArg változó az új, módosított értékeket fogja eltárolni, amit a rádión éppen beállított értékekkel összehasonlít. Ha változás történik, akkor átállítja az új értékekre az eszközt is. A weblapon kijelzett értékek is változnak, amint a weblapról érkező új beállítást adunk meg, illetve ha manuálisan újratöltjük az weblapot egy fizikai művelet után.

A webes felület megírásának művelete sok időt vett igénybe. Főleg a dokumentáltság hiánya okozta a legnagyobb problémát, azonban a tesztek és egy, a C programnyelvet tárgyaló szakirodalom [19] segítette a munkámat. A webes felület minden műveletéhez egy külön kódrészletet kellett létrehoznom, mivel ezek az általam használt verzióban teljesen hiányoztak. Ezeket a függvényeket külön kellett választanom az eszköz fizikai vezérlőitől, továbbá az összes lehetséges esetet le kellett kezelnem, hogy a vezérlés ne váltson ki inkonzisztens állapotot.

A memóriakapacitás behatároltságától eltekintve a lehetőségek korlátlanok. A teljes vezérlőkészletet át lehetne implementálni egy honlap felületre. Meg lehetne valósítani egy nem túl bonyolult SSL titkosítást, vagy egy hálózati autentikációs modult, mellyel megakadályoznánk az illetéktelen személyek hozzáférését. A megvalósított webes vezérlés lehetőséget nyújt az eszköz távolról történő működtetésére.

4. Értékelés és hibák

A webes felület megfelelően helyettesíti a panel vezérlőit, azonban a felület funkciói igen korlátozottak, mivel a panel RAM-ja nem elég nagy a bonyolultabb, de főleg nagyobb terjedelmű módosítások végrehajtásához. Ezért legfontosabb hardveres módosításként érdemesnek tartom egy nagyobb kapacitású memória beépítését.
Probléma még, hogy lassú internetkapcsolat következtében a folyamatos lejátszás meghiúsulhat, vagy túl nagy várakozási idő léphet fel, aminek következtében az internetrádió elérhetetlennek minősíti a szervert, és abbahagyja a kapcsolódási kísérletet.

Az internetes rádió a csatornákat fixen kódolva tárolja. Egyelőre a programkód úgy van konfigurálva, hogy bizonyos típusú állomásokat legyen képes tárolni. Nehéz olyan internetes rádióállomást találni, amely minden követelménynek eleget tesz, így az esetleges szerver elérési és adattovábbítási módosítások következtében a rádió képtelen kapcsolódni a szerverhez.

A megfelelő adat küldéséhez túlzottan specifikált elérési útvonalat kell adni, melyet nem minden rádióállomás közöl, ugyanis legtöbb esetben egy URL-cím, illetve a stream letöltése elegendő a működéshez. Ahhoz, hogy elegendő legyen egy streamfájl letöltése, és annak betöltése az eszközbe, teljesen más adatfeldolgozási metódusok szükségesek, ennek megvalósítása minden esetben a teljes programkód módosítását jelentené.

5. Összefoglalás – Távlatok

Az internetes rádiózáshoz tartalmazó szakirodalom áttekintése után szélesebb látókörrel térképezhettem fel a kiírt problémákat. A dolgozatom megírása során megismerkedtem az általam kiválasztott Microchip internetrádió demonstrációs panel működésével, programozásával, az MPLAB fejlesztői környezettel és új webes megoldásokkal. Az általam létrehozott weboldal pontosan helyettesíti a hardveres vezérlési műveleteket. Munkám során talán nem mindig indultam el a megfelelő irányba, ugyanakkor úgy érzem, a kiírt feladatoknak eleget tettem.

Feladataim elvégzését olykor hátráltatta a megfelelő dokumentáció hiánya, vagy az általam kiválasztott hardveres eszköz gyári hibái, ezért a távlatok megvalósításának elengedhetetlen és egyben elsődleges feltételének tűzöm ki a fennálló, és a korábban említett hibák kiküszöbölését. Az általam módosított működő programkód és weboldal CD mellékletként megtalálható, valamint az internetrádiós panel megtekinthető az egyetem laborjában.

A jelenlegi állás szerint a csatornák fixen vannak eltárolva a szoftverben. Elképzelésem szerint egy webes felületen történő módosítás lehetőségével is bővíteném az eszköz funkcióit. A honlapon keresztül képesek lennénk új rádióadás hozzáadására, majd eltárolására. Ehhez a megoldáshoz szükség lenne a honlapon beviteli mezőkre, melyeket a rádióadás elérési adataival kellene feltölteni. Hibás adatok esetén természetesen az elérési út ne tárolódjon el a memóriában, hanem automatikusan törlődjön. Ez a módosítás mindenképp nagyobb memóriakapacitást igényelne. Egy memória kártyát lehetne illeszteni a panelre, így egy nagyobb kapacitású tárolóegységet kapnánk.

Az internetelérést módosítani lehetne vezetékes kapcsolatról vezeték nélkülire, ami nem csak a panel hardveres újratervezését, de a programkódbeli módosításokat is maga után vonna.

Érdemesnek tartanám egyfajta akkumulátor beszerelését is, mivel ez megnövelné az eszköz hordozhatóságát is, ami jelenleg igen korlátozott – ugyanis jelenleg folyamatos hálózati áramellátás szükséges a működéshez.

Továbbá a felhasználói élmény fokozására egy kisebbfajta hangszórót és érdemesnek tartanék beilleszteni a panelre, melynek vezérlését, mint a panelen, mind a weboldalon meg kellene valósítani.

A jelenlegi külső megjelenés ezen kívül elég puritán, így jelentős esztétikai előrelépés lenne egy külső ház megépítése is, ami egyben a jelenlegi állapot sérülékenységét akadályozná meg.

Mindezen javaslatok ellenére az adott eszköz képes ellátni a minimális követelményeket, feladatainak eleget tesz, és élvezhetően hallgathatók a beprogramozott rádióállomások.

Köszönetnyilvánítás

Szeretném megköszönni konzulensemnek, Tihanyi Attila Tanár Úrnak, aki mindig segítségemre volt dolgozatom írása közben, útmutatásokat adott, mikor szükségem volt rá. Bármilyen kérdéssel fordulhattam hozzá, mindig sikerült megtalálnunk a helyes megoldásokat és kompromisszumokat. Beszerezte, és rendelkezésemre bocsájtotta a szükséges eszközöket, egyetemi laborját használhattam. Rengeteg hasznos információval és tapasztalattal gazdagodtam a munkám során.

És ezúton szeretném megköszönni testvéremnek, Fichó Erzsébetnek, akihez szintén bármikor fordulhattam tanácsért dolgozatom formai keretét illetően.

Irodalom- és képjegyzék

Irodalomjegyzék

[1] David Austerberry, *The technology of video and audio streaming*, Elsevier, vol. 7, pp. 133-149, 2005.

[2] *First Webcast*, Jet-Stream, 1994. [Online]
Elérhető: http://www.jet-stream.com/blog/first-webcast/

[3] Franc Kozamernik és Michael Mullan, *An Introduction to Internet Radio*, European Broadcasting Union technical Review, 2005.10.26.

[4] Pete Loshin, *TCP/IP Clearly Explained*, Morgan Kaufmann, 2003.

[5] Jon Postel, *RFC: 793 TRANSMISSION CONTROL PROTOCOL*, Darpa Internet Program Protocol Specification 1981.

[6] Jon Postel, *RFC: 768 User Datagram Protocol*, Darpa Internet Program Protocol Specification, 1980.

[7] H. Schulzrinne és S. Casner, *RFC 3550: RTP: A Transport Protocol for Real-Time Applications*, 2003.

[8] H. Schulzrinne, *RFC 2326: Real Time Streaming Protocol (RTSP)*, 1998.

[9] C. Huitema, *RFC 3605: Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)*, 2003.

[10] R. Fielding, *RFC 2068: Hypertext Transfer Protocol HTTP/1.1*, 1997.

[11] Jon Luini, Jon R. Luini, Allen E. Whitman, *Streaming audio: the FezGuys' guide, New Riders Publishing*, 2002.

[12] wikipedia.org, *WAV*, 2011. [Online]
Elérhető: http://hu.wikipedia.org/wiki/WAV

[13] http://konyvtar.hu, *Digitális hang*, 2011. [Online]
Elérhető: http://konyvtar.hu/wiki/Digitális_hang#cite_note-12

[14] *ChipCAD Elektronikai Disztribúció Kft Katalógus*, 2008. [Online]
Elérhető: http://www.chipcad.hu/letoltes/moldal%202008-8.pdf

[15] H. Schlunder and R. Richey, *TCP/IP Networking: Internet Radio Using OLED Display and MP3 AudiovDecoder*, Microchip Application Note AN1128, Microchip, 2009.

[16] Nilesh Rajbharti, *The Microchip TCP/IP Stack*, Microchip Technology Inc, 2002.

[17] http://hu.wikipedia.org , *MP3*, 2011 [Online]
Elérhető: http://hu.wikipedia.org/wiki/MP3

[18] http://www.codexonline.hu, *CGI*, Bálint Aliz, 2009. [Online]

Elérhető: http://www.codexonline.hu/CodeX5/alap/web/Cgi.htm
[19] B. W. Kernighan és D. M. Ritchie, *The C Programming Language*, Prentice Hall Publishing Company, 1988

Képjegyzék

[20] Eredeti kép forrása: http://www.thenetworkencyclopedia.com, Unicasting [Online]
Elérhető: http://www.thenetworkencyclopedia.com/d2.asp?ref=1992
[21] Eredeti kép forrása: http://www.webbasedprogramming.com
[22] http://www.tirta-fajri.net/membangun-streaming-server-mp3/
[23] H. Schlunder and R. Richey, *TCP/IP Networking: Internet Radio Using OLED Display and MP3 AudiovDecoder*, Microchip Application Note AN1128, Microchip, 2009.
[24] Eredeti kép forrása: http://techno-pro.blogspot.com/2011/08/mplabicd-2-in-circuit-debugger.html
[25] Eredeti kép forrása: Nilesh Rajbharti, *The Microchip TCP/IP Stack*, Microchip Technology Inc, 2002.
[26] Eredeti kép forrása: http://wiki.wireshark.org/CaptureSetup/Ethernet