LUND INSTITUTE OF TECHNOLOGY
Lund University

Informationsteknologi, LTH
Digitala projekt
2000-05-24

WWR

World Wide Remote

Henrik Jönsson, e96hje
Marcus Nilsson, d95mn
Patrik Nilsson, e96pn

Handledare: Stefan Nyman, Informationsteknologi, LTH

Abstract

The purpose of this project was to develop a remotecontrol(WWR). A remotecontrol which can control all electrical equipment that has a IR-remotecontrol, with a range as large as the whole Internet. The remotecontrol shall also be able to control a few digital outputs. The remotecontrol can be used on your vacation, via a webreader, to record your favorite program when you forgot to program the video.

To build the WWR, three important parts are being used, hardware, client and server. The hardware contains a processor, memory, IR-receiver and a IR-transmitter. The client is a program with grafical user interface, written in Java. The GUI is designed like a remotecontrol. The server works like a link between the client and the hardware. The server sends the buttoncode to the hardware. The server also keeps in mind which of the digital outputs that are active or not.

The result became very successful, with one exception, the client couldn't be run as an applet. This depends on partly that the Java-swing package is being used, today none of the webreaders has built-in support for this, and partly that the networkcomunication, between the applet and server, requires a signed applet. We haven't been able to solve the signing.

Innehållsförteckning:

1 Inledning

Att vi valde att gå kursen digitala projekt var nog helt enkelt för att det verkade väldigt intressant. Att få fria händer till att konstruera något från grunden lät väldigt lockande. Utbildningen på LTH är ju annars väldigt teoretisk och små möjligheter ges åt det praktiska.

När kursen inleddes hade ingen i gruppen en aning om vad vi skulle göra. Vi ville göra någonting originellt som inte var gjort tidigare i kursen. Av en ren slump kom vi på tanken med en internetkontrollerbar fjärrkontroll. Meningen med en sådan apparat kan diskuteras men att tex kunna programmera videon från andra sidan jorden lät ändå ganska vettigt. Så fick det bli. Man ska alltså från vilken nätansluten dator som helst kunna koppla upp sig mot sin hemdator och via ett snyggt gränssnitt manövrera TV, Video, och stereo. För att inte låsa sig vid bara IR försågs systemet dessutom med 8 digitala utgångar.

Projektet kom att innefatta såväl hårdvarukonstruktion som en hel del Javaprogrammering.

2 Komponenter

2.1 Processor 68HC11 – Motorola

Hjärnan i kopplingen är den eminenta enchipsdatorn 68HC11 från Motorola som innehåller det mesta man kan önska sig. Den har ett ROM programminne på 12 kb, och ett RAM-minne på 512 bytes. Dessutom finns det ett inbyggt EEPROM på 512 bytes men detta hade vi ingen användning för i detta projekt. För att kunna kommunicera med andra komponenter finns det två inbyggda block för seriekomminikation. Dels SCI som är en asynkron seriekanal för att exempelvis kunna växla information mellan en HC11 och en PC, dels en SPI-kanal som istället är synkron med en massa tillämpningar. I vårt projekt användes den till ett seriellt EEPROM på 16 kb. Vidare finns det ett antal system för att mäta tider och för att skapa pulståg vilka vi hade stor nytta av. De system vi använt oss av kommer att beskrivas senare.

2.2 Konverterare MAX 233 – Maxim

MAX 233 används för att konvertera TTL-nivåerna 0 och 5 V till de spänningar PC:ns seriekanal använder sig av dvs –10 till 10 V. Detta sker internt genom att ström pumpas in i kondensatorer som laddas upp till önskad spänning. Max 233 är en lite lyxigare variant av den betydligt vanligare MAX 232, med den skillnaden att MAX 233 har kondensatorerna inbyggda I kapseln. Ända externa komponent som behövs är en avkopplingskondensator.

2.3 Minne X25128 – Xicor

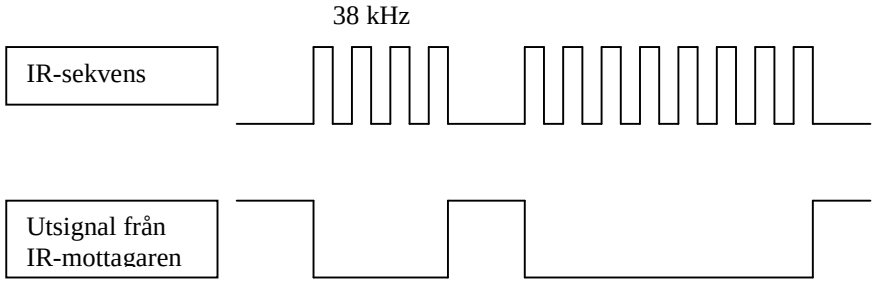
X25128 är ett seriellt EEPROM-minne som använder sig av Serial Peripheral Interface (SPI), en standard som stöds av HC11:an. Storleken på det minne vi valde var 16 kb. Fördelen med det seriella minnet är det låga priset, den ringa storleken och det faktum att den endast tar upp 5 ben på HC11:an. Priset man får betala är en lägre hastighet, men då det ställs väldigt låga krav på hastigheten i detta projekt spelade detta ingen roll.

2.4 Oscillator LM555C – National Semiconductor

Denna krets kan fås att oscillera med en frekvens som är beroende av yttre komponenter. Kretsen kan användas till en hel del men i vårt fall var dess uppgift endast att producera en 38 kHz fyrkanvåg.

2.5 IR-mottagare IS1U60 – Sharp

IR-mottagaren används för att ta emot och koda av IR-signaler och skicka ut dessa i form av TTL-nivåer. Mottagaren består bl.a. av ett bandpassfilter med en bandpassfrekvens på 38 kHz. Detta för att endast detektera IR-ljus. Funktionen är illustrerad nedan.

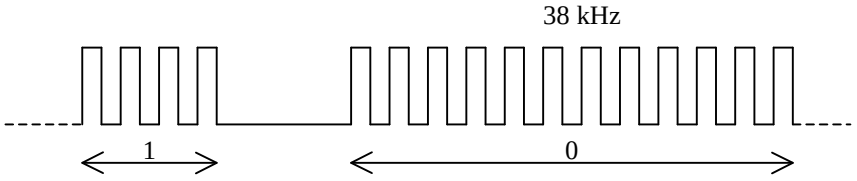


2.6 Display 2x16 tecken

Vi använde oss av en standarddisplay på 2x16 tecken. Man kunde själv bestämma om man ville ha 4 eller 8 bitars interface för att kommunicera med displayen. Då vi hade gott om lediga portar valde vi 8 bitars interface.

3 Kort om IR-signaler

En visit på internet gjorde oss kloka på att i stort sett alla tillverkare använde lite olika sätt för att koda IR sekvenserna. Grundprincipen är dock den samma. Det finns i huvudsak tre sätt på vilka man kodar, pulse coded, space coded och shift coded. I fallet med pulse coded representeras en etta av en puls med den aproximativa längden 550 µs och en nolla av en puls med längden 2200 µs. Mellan varje bit är det ett uppehåll på ca 550 µs. En puls består av 38 kHz fyrkanvåg enl figuren nedan. (Frekvensen stämmer inte utan är rent illustrativ).



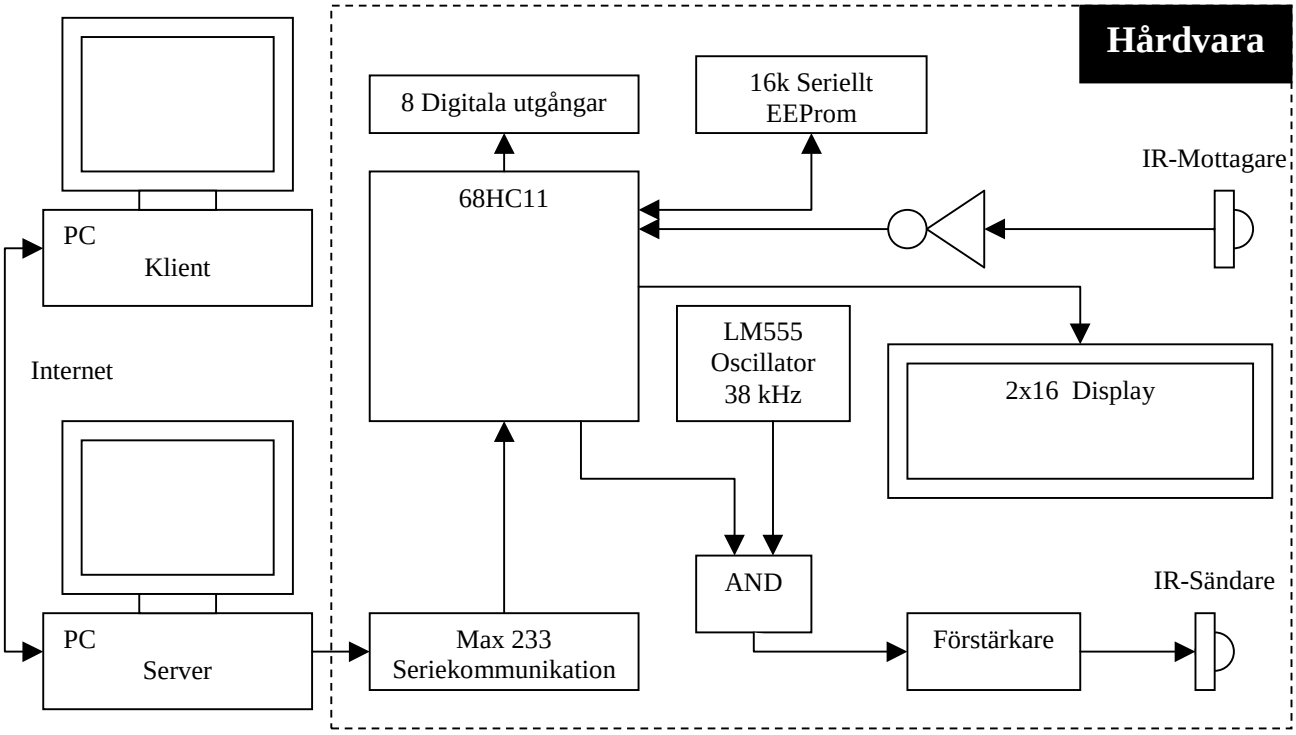
I space-coded låter man istället tiden mellan pulserna representera ettorna och nollorna. Olika fabrikat hade dock olika långa pulser för att representera data men tiderna förekommer nästan alltid i multipler av 550 µs. De flesta använde sig dessutom av en header och en stopsignal bestående av något längre pulser. Antalet bitar som användes för att coda IR-signalerna varierade mellan 12 och 32.

4 Hårdvaru-utveckling

Hårdvaran har följande uppgifter:

- Ta emot och lagra IR-sekvenser från befintliga fjärrkontroller
- Från kortet kunna sända iväg lagrade IR-sekvenser
- Ta emot data från PC och sända iväg motsvarande IR-sekvenser
- Ta emot data från PC för att styra 8 digitala utgångar

Hårdvaran för att lösa dessa uppgifter är illustrerad nedan.



4.1 Konstruktionsprincip

Valet av processor stod mellan enchipsdatorn 68HC11 och processorn Motorola 68000. Att välja HC11 föll sig naturligt. Den är liten, kräver väldigt lite externa komponenter och den har alla de funktioner vi behövde inbygda från början. Det fanns inga krav på varken mycket RAM eller ROM-minne.
Anledningen till att vi använde oss av ett seriellt minna var även det för att dra ner på antalet externa komponenter och för att spara plats. Övriga komponentval föll sig också naturliga. Displayen används bara då man ”tränar upp kortet” så denna kan också vara ganska liten. Som IR-mottagare valde vi ISU60 då denna med inbyggt filter och förstärkare gör allt grovjobb. Kretsen fungerar väldigt bra men gav oss en del huvudbry då den tog emot signaler utan att vi sände några. Vi provade att skärma av baksidan på dioden med ettiketter och blev helt av med problemen, kretsens funktion finn beskriven tidigare.

4.2 Mottagning av IR-signaler

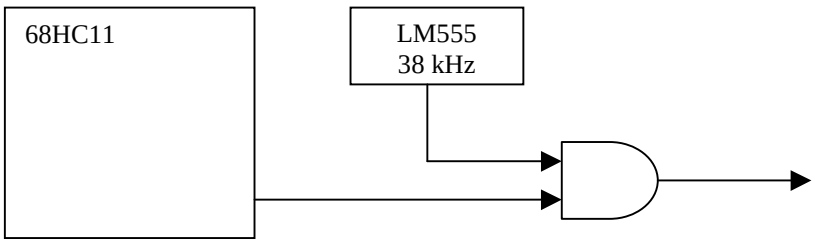
Målet då mottagningen skulle implementeras var att göra den i princip helt oberoende av vilken typ av fjärrkontroll som används. Det enda vi antog var att signalerna kodades med runt 38 kHz. Då vi hade tillgång till ganska mycket minne hade vi råd att slösa med detta. Vi sparade helt enkelt tiden mellan alla flanker i utsignalen från IR-mottagaren. Tiden fick vi genom att använda HC11:ans inbyggda funktion InputCapture. Detta är en ingång som känner av när ingången går från etta till nolla eller tvärtom. När detta sker lagras värdet av en intern frigående 16 bitars räknare i ett lika stort register varpå en flagga sätts och ett eventuellt avbrott inleds. Skillnaden mellan två på varandra följande flanker lagrar vi i minnet med samma antal bitar.
Processorn arbetar internt med en klockfrekvens på 2 MHz vilket ger en tidsupplösning på 0,5 µs vilket är fullt tillräckligt i dessa sammanhang.

Då fjärrkontrollen ska fungera med så många fjärrkontroller så möjligt måste vi anpassa oss efter den som sänder längst IR-sekvens då även denna ska få plats i minnet. Den längsta vi hittat var på 32 bitar vilket med start och stopp-puls ger 32*2-1=67 tider att lagra. För att vara på den säkra sidan läser vi in 80 tider. Då varje tid lagras med 16 bitar tar en sekvens upp 16*80=1280 bitar. Totalt ska det lagras 71 IR-sekvenser vilket ger en maximal minnesåtgång på 71*1280=89600 bitar vilket är ca 11 kb. Minnet räcker alltså med god marginal eftersom vårt minne är 16 kb stort. Då det tar väldigt lång tid för ett EEPROM att fullborda en skrivning måste detta ske efter att sekvensen är inläst. Alla inlästa värden lagras därför förs i en vektor innan skrivningen till minnet påbörjas.

Från början använde vi oss av ett system där vi räknade ut hur många T (T=550 µs) varje puls bestod av och lagrade detta i stället för att spara minne. Då T var ett litet tal räckte det med 8 bitar för att lagra detta. Detta fungarede hjälpligt men inte perfekt. TV:n som utgjorde vårt testobjekt kunde på sin höjd tolka fyra av fem skickade sekvenser vilket vi inte var nöjda med. En studie av fjärrkontrollens IR-sekvenser visade att denna inte följde de standarder vi läst om varpå systemet ovan implementerades.

4.3 Sändning av IR-signaler

Vid sändning ska en tidigare inläst sekvens återskapas och skickas i väg till en IR-diod. Det sätt på vilket vi implementerat mottagningen gjorde detta ganska enkelt. Allt som behövde göras var att läsa in en tid från minnet och vänta denna tid. När tiden har gått togglas utsignalen varpå en ny tid läses in osv. Detta förfarande kommer att återskapa den sekvens som IR-mottagaren skickat. För att nu blanda ut pulserna med 38 kHz signalen har vi använt nedanstående koppling. LM555 har en inbyggd resetfunktion som skulle kunna användas i stället för AND-grinden men oförklarliga problem med denna metod gjorde att vi valde valde den andra. AND-grinden visade sig ochså snygga till signalen avsevärt, då HC11:ans utgång av någon anledning var väldigt brusig.



Metoden vi använde för att vänta de inlästa tiderna var en annan inbyggd funktion i HC11, nämligen tidsavbrott. HC11 har 5 inbyggda 16 bitars register för detta ändamål. När den frigående 16 bitars räknaren har ett identiskt värde med det i registret kommer en flagga att sättas och en eventuell avbrottsrutin kommer att inledas. I denna avbrottsrutin togglas utgången, en väntetid läses in från minnet och läggs till den gamla för att sedan lagras i registret. Nästa avbrott kommer nu att inträffa efter denna tid och processen upprepas.
I en första implementering fungerade detta till att återskapa huvudpulsen i IR-sekvensen som är lång men resten av sekvensen uteblev. Efter mycket huvudbry drog vi slutsatsen att två minnesläsningar (16 bitar) inte hanns med mellan varje puls. Detta löste vi genom att först göra alla läsningarna från minnet och spara dessa i en vektor innan sändningen påbörjades. Detta gjorde dock ingenting då en vektor ändå var nödvändig för att spara sekvensen enl tidigare. Återvinning var ordet. En 160 bytes stor vektor är ju inte direkt försumbar när man använder en HC11.

4.4 Knapparna

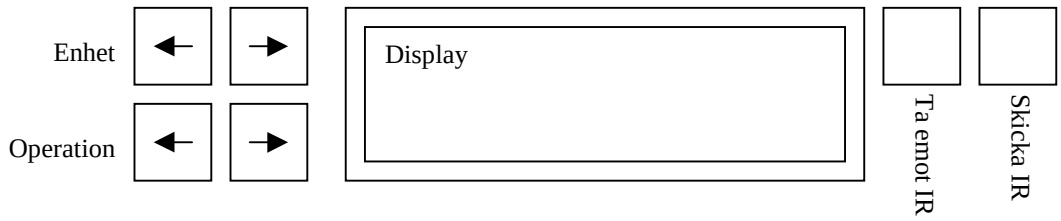
Knapparna använder sig precis som fallet med sändningen av tidsavbrott. Detta är egentligen en onödigt exakt metod för att läsa av knapparna, andra system för detta ända mål finns inbyggda, men har man 5 tidsavbrott att leka med så varför inte. Var 20:e ms kommer ett avbrott i vilket knapparna läses av.

4.5 Digitala utgångar

De 8 digitala utgångarna är kopplade till 8 lysdioder för att demonstrera dess funktion. Vill man utföra något vettigt med utgångarna får exempelvis relä kopplas in.

4.6 Gränssnitt

Stor möda har lagts ner på att göra systemets gränssnitt så enkelt det bara kan bli. Upplägget kan skådas nedan:



Ska en sekvens lagras knappas den enhet (Video, Tv, Stereo eller Extra) och den operation (ex Play, stop, ch1, ch2 etc) in m.h.a piltangenterna varpå knappen ”Ta emot IR” trycks in. Systemet ställer sig nu och väntar på IR-sekvensen. När den fått hela sekvensen går den tillbaka till normalläge. Samma rutin gäller för att skicka en IR-sekvens med den skillnaden att knappen ”Skicka IR” tryck in i stället.
Skulle man ångra sig efter att ha tryckt på ”Ta emot IR” kan man avbryta genom att trycka på vilken av knapparna som helst.

4.7 Programmering av HC11

Programmeringen av processorn skedde i programmet embedded workbench av IAR. Språket som användes var C. En hel del av programmeringsarbetet har gått ut på att göra rutiner för att kunna kommunisera med displayen, minnet och med knapparna. F.ö har mest tid lagts ner på att ta emot och återskapa signaler på ett så ”fjärrkontroll-oberoende” sätt som möjligt. Som alltid vad gäller programmering har mycket tid även lagts på felsökning och en hel del huvudkliande. Programmvaran från IAR har varit väldigt trevligt att arbeta med och inga problem har uppstått på grund av denna. Tillsammans med emulatorn är detta ett perfekt utvecklingsvertyg.

5 PC-programmet

Vi valde att göra vår pc-applikation i programspråket java. Detta valdes dels beroende på dess många olika grafiska komponenter som lätt kan sammankopplas till ett snyggt gränssnitt. Vi använder oss av det nya swing-gränssnittet som kom i java 1.2.

Vi hade även förhoppningar om att vårt program skulle kunna läggas upp som en applet och köras från vår egen hemsida, detta fick vi tyvärr aldrig att fungera beroende på att en applet inte har rättigheter att koppla upp sig mot en annan dator och kommunicera med denna. Detta ska gå att lösa genom att signera appleten och vi har lagt ner mycket tid på att försöka få detta att fungera, men gav till slut upp.

5.1 Utförande

Vi började med att programmera en server- och en klientdel som inte hade några andra funktioner än att koppla upp sig mot varandra. Dessa gick mycket lätt att få igång efter att titta i boken ”*Programutveckling med java*” [3] där det fanns ett utmärkt exempel på hur man använder *sockets* för att få igång en kommunikation.

Efter vi fått kommunikationen att fungera började vi bygga upp vår fjärrkontroll i klienten, genom att skapa våra fyra paneler och placera ut knappar på dessa. Vi lade här in direkt att så fort man tryckte på en knapp så skickades ett tal (se appendix B: ”Taggar”) vidare till servern, som skrev ut vilket tal den fick. Detta gick med utan några problem men tog lång tid pga att vi inte ville använda oss av någon av javas automatiska layout-managers och alltså behövde själv sätta in exakt hur stor och var på panelen knappen skulle ligga. Från början lade vi alla dessa paneler i en fil men efter att vi skrivit hela denna klass med underklasser valde vi att flytta ut panelerna i egna klasser (och filer) för att få ett mer överskådligt program.

Vi hade nu i princip ett fungerande program, där det enda som återstod för att uppfylla vår kravspecifikation var att servern skulle koppla upp sig mot en serieport och skicka talet, som den får från klienten, vidare till hårdvaran. Vi kände att vi hade mycket tid kvar tills vi skulle vara färdiga så vi tyckte att vi kunde utöka med någon/några funktion/funktioner i programmet, Vi valde då funktionerna:

- text på våra knappar till de digitala utgångarna.
- intern video-timer.

För att kunna få text på de *digitala* knapparna blev vi tvugna att utöka vår socket-kommunikation mellan server och klient till dubbelriktad. Detta gick snabbt då det i princip bara var att kopiera uppkopplings-procedurerna mellan server och klient. Sen fick vi lägga till att servern skulle läsa in från fil och skicka dessa strängar vidare till klienten som skrev denna sträng direkt på respektive knapp. Vi kom här även på att servern borde hålla reda på vilka knappar som var aktiverade. Detta var en snabbt tilllagd funktion i och med att vi redan hade kommunikationen igång.

Nästa utökning blev att vi implementerade en timer, på klient sidan var denna ändring till en början mycket lätt. Vi lade endast till några knappar och textrutor där vi kunde skriva in kanaler och tidpunkter. När vi skulle lägga till timern i servern blev det däremot lite problem för vi hade byggt upp servern så den alltid låg och väntade på kommandon från klienten. För att inte behöva göra om hela servern från början beslöt vi att införa parallella exekveringar i servern och lade då ut hela vår ”gamla” server förutom uppstartsrutinerna i en egen tråd och började bygga upp en ny tråd som skulle hantera en timer. Denna tråd var blev med hjälp av javas nya *Calender*-klass och *sleep*-funktion lätt att programmera, det var i princip bara att stoppa in tid och datum i ett Calendar-objekt och säga till tråden att sova tills denna tid uppnåddes.

Vi tyckte nu att man borde kunna kolla vilka timerar som användes, hur de var inställda och radera felinställda timerar. För att göra detta byggde vi upp ett dialogfönster som frågade servern hur dess timerar var inställda, vi lade i detta fönster även in kontroll om två eller flera timerar överlappade varandra.

När detta var färdigt så började även hårdvaran med dess programmering börja bli klar så då började vi att ta itu med att koppla upp oss mot serieporten. Det visade sig att man behövde ladda hem ett paket med filer från javas hemsida för att få tillgång till datorns olika portar. När vi laddat ner detta var det enkelt att få igång kommunikationen då uppkopplingen fungerade likadant som för TCP/IP-kommunikationen, det var bara lite andra namn på klasser och funktioner som skulle tillkallas här.

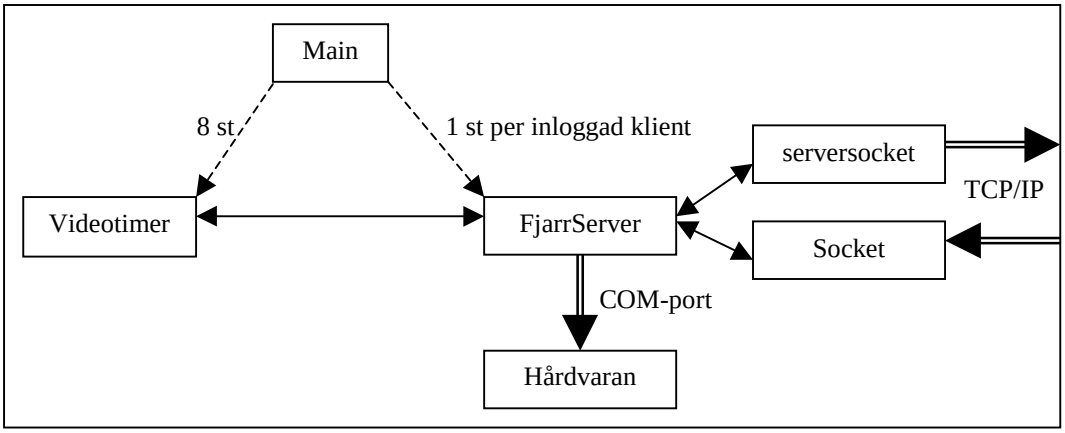
I slutet under programutvecklingen bestämde vi oss för att ha någon sorts inloggingsskydd. Vi lade bara till en startfönster till klienten där man fick skriva in ett lösenord som sedan servern skulle godkänna för att inloggning skulle tillåtas. Vi skickade lösenordet direkt som en textsträng, man skulle kunna tänka sig att lägga in någon kryptering här men tiden började närma sig *deadline* så vi lät det vara.

I stora drag tycker jag att programeringen av PC-applikationen har flytit på mycket bra utan några större problem. Ibland kan det ha varit lite tråkigt arbete, när man gjort ett par knappar är det ju bara en massa kopiering av kod för att göra fler knappar, bara lite andra variabel-namn och placeringar.

5.2 Server

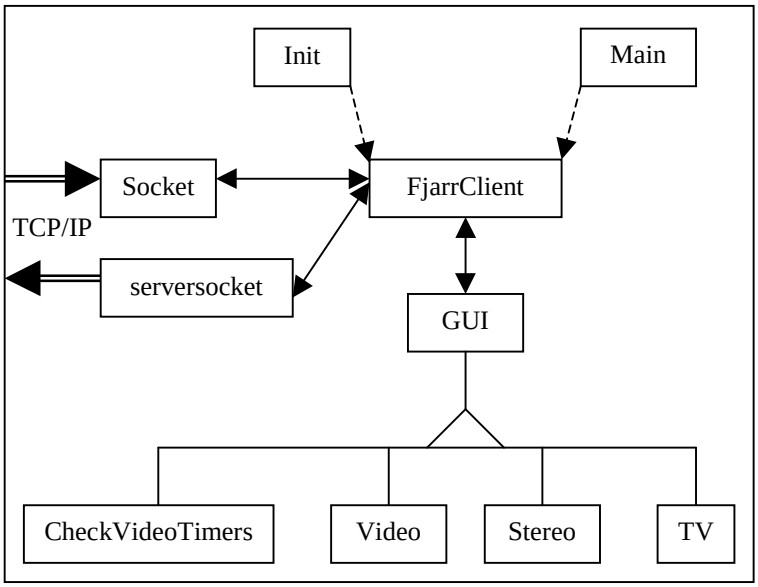
Servern består av en main-funktion som konstant ligger och väntar på att en klient vill logga in. Denna startar även upp åtta stycken video-timerar som man kan programmera till att spela in på videon mellan två tider på en viss kanal.

När en klient vill logga in på servern startas en ny tråd för denna klient. Denna tråd kollar nu om klienten har behörighet att logga in, genom att man kollar lösenord. Lösenordet läser servern in från filen *xsyntax.dll.* Efter man har blivit godkänd, man har tre försök på sig, skickas information om de digitala kanalerna över till klienten (vilka som är tillslagna, knappnamn). Knappnamnen läser servern in från filen *button.txt.* Nu är hela initieringen för en klient färdig och den nya tråden ligger endast och väntar på order från klienten, t ex knapptryckningar och timerinställningar.



5.3 Klient

Klienten startas upp antingen genom en main-funktion, eller en init-funktion beroende på om man startar programmet som java-applikation eller som java-applet. Oberoende hur man startar upp klienten kommer man att starta en socketoch en serversocket mot server-datorn, om detta lyckas frågar man efter ett lösenord. Om lösenordet godkänns av servern startar man en fjarrklient-panel i ett nytt fönster. Detta fönster består i sin tur av fyra stycken paneler som är inlagda i en kartotek-liknande system. De fyra olika panelerna styr vardera en stereo, en TV, en video respektive åtta stycken digitala utgångar. Från video-panelen kan man öppna ett nytt fönster där man kan kolla vilka timerar som är inställda och radera felinställda timerar.



5.4 Användarhandledning

För att köra klienten och servern behövs minst Java 2 SDK, Standard Edition, v 1.2*. För servern krävs dessutom Java Communications API 2.0** till seriekommunikationen. Laddas ned från adresserna nedan. Installera.

* http://java.sun.com/products/jdk/1.2/
** http://java.sun.com/products/javacomm/index.html

5.4.1 Starta servern

Om annan än comport 1 används, måste COM1 bytas ut vilken port som skall användas i filen Run-Server.bat.
Kör Run-Server.bat
Om detta inte går, lägg till raden ”PATH C:\sökväg\till\jdk1.2\bin” överst i Run-Server.bat

5.4.2 Starta klienten

Om klienten inte körs på samma dator som servern, måste localhost bytas ut mot datornamnet eller IP-addressen som servern körs på, i filen Run-Klient.bat
Kör Run-Klient.bat
Om detta inte går, lägg till raden ”PATH C:\sökväg\till\jdk1.2\bin” överst i Run-Klient.bat

6 Resultat

Efter sex veckors hårt arbetande har vi nu en PC-fjärrkontroll som, enligt vad vi tror, man ska kunna lära upp av vilken annan IR-fjärrkontroll som helst till att styra TV, video och stereo. Projektet har tagit upp mycket tid då vi har suttit här nästan varje vardag och jobbat några timmar. Det hela har flutit på i en bra takt och vi har inte fått särskilt många helt oförklarliga fel.

6.1 VidareUtvecklingar

En ganska enkel vidareutveckling vi tänkte på att göra men inte hade tid till var att man skulle kunna spara data, t ex timerinställningar och vilka digitala utgångar som är aktiva, till en fil som sedan skulle läsas in vid omstart av servern. Som det är nu så avaktiveras alla digitala utgångar och timerarna är stoppade vid uppstart av servern.

Man skulle även kunna lägga till ett antal digitala ingångar och då även få lägga till dubbelriktad kommunikation mellan hårdvaran och PC:n.

När man håller in en av bläddringsknapparna borde programmet efter en stund bläddra vidare till följande funktioner.

6.2 Kända Problem

Det största problemet mot vår kravspecifikation är att vi inte har fått programmet att fungera som en applet. Detta har ett par olika orsaker, dels att vi inte har kunnat få signeringen av våran applet att fungera och dels att vi har använt oss av javas swing-paket för grafiken. Ingen av de nu vanliga web-läsarna har stöd för swing och vi fick inte det plug-in som finns för nedladdning att fungera.

Vi valde även att ta bort funktionen showview från våran kravspec, efter att jämfört våra egna videoapparater visade det sig att showview programmerades in olika för olika apparater.

7 Referenser

[1] Java[TM] 2 Platform, Standard Edition, v1.2.2 API Specification
http://java.sun.com/products//jdk/1.2/docs/api/index.html

[2] Eriksson, Hans-Erik, ”Programutveckling med Java”, Studentlitteratur 1997
ISBN: 91-44-00219-X

[3] Flanagan, David, ”Java in a nutshell – a desktop quick reference 3rd edition”, O’Reilly november 1999
ISBN: 1-56592-487-8

[4] Flanagan, David, ”Java foundation classes in a nutshell – a desktop quick reference ”, O’Reilly september 1999
ISBN: 1-56592-488-6

[5] Motorola inc, ”MC68HC11 E Series Technical Data”, 1995

[6] Motorola inc, ”M68HC11 Reference Manual”, 1990

[7] Nyman Stefan, ”Bygg och programmera med enchipsdatorn 68HC11”, KFS i Lund augusti 1999

[8] Digitala Projekt (it), ”Datablad HC11”

[9] ELFAs hemsida
http://www.elfa.se/se/

[10] EE 498 Remote Control Study
http://www.ee.washington.edu/conselec/A95/projects/pierreg/main.htm

[11] Kernighan, Brian W and Ritchie, Dennis M, ”The C-programming language”, Prentice Hall 1988
ISBN: 0-13-110362-8 {PBK}
ISBN: 0-13-110370-9

8 Appendix A: Kravspecifikation

Fjärr

(Fjärrkontrollen som gör skäl för namnet)

En Fjärrkontroll med världens största räckvidd ska byggas. Den består i stort av ett kort med en 68HC11, IR-sändare, en IR-mottagare, knappsats samt en display. Detta kort kopplas sedan via serieporten till en dator.

Krav:

1. Kortet ska kunna lära sig följande funktioner av andra fjärrkontroller via IR-mottagaren på kortet:
Play, Stop, Spola fram, Spola tillbaka, Volym upp, Volym ner, Spela in, 0 - 9, F1 - F5, Showview, Ok, On, Off.
2. Inlärningen skall göras mha knappsats och display på kortet.
3. Med kortets knappsats och display skall stereo, video etc kunna kontrolleras.
4. Med en PC skall man kunna styra kortet till att kontrollera stereo, video etc med ovanstående funktioner via IR-sändaren på kortet.
5. Man skall kunna styra 8 digitala utgångar från PCn.
6. PCn ska kunna fjärrstyras från viken annan internetansluten dator som helst, om denna har webbläsare.

9 Appendix B: ”Taggar”

VideoPlay = 20;
VideoStop = 21;
VideoPause = 22;
VideoRRW = 23;
VideoFF = 24;
VideoRec = 25;
Video0 = 26;
Video1 = 27;
Video2 = 28;
Video3 = 29;
Video4 = 30;
Video5 = 31;
Video6 = 32;
Video7 = 33;
Video8 = 34;
Video9 = 35;
VideoPower = 36;
VideoTimer = 37;
VideoOK = 38;
VideoF1 = 39;
VideoF2 = 40;
VideoF3 = 41;
VideoF4 = 42;

TV0 = 50;
TV1 = 51;
TV2 = 52;
TV3 = 53;
TV4 = 54;
TV5 = 55;
TV6 = 56;
TV7 = 57;
TV8 = 58;
TV9 = 59;
TVPower = 60;
TVVolumeUp = 61;
TVVolumeDown = 62;
TVNext = 63;
TVPrev = 64;
TVF1 = 65;
TVF2 = 66;
TVF3 = 67;
TVF4 = 68;

StereoPlay = 70;
StereoStop = 71;
StereoPause = 72;
StereoRRW = 73;
StereoFF = 74;
StereoRec = 75;
StereoPower = 76;
Stereo0 = 77;

Stereo1 = 78;
Stereo2 = 79;
Stereo3 = 80;
Stereo4 = 81;
Stereo5 = 82;
Stereo6 = 83;
Stereo7 = 84;
Stereo8 = 85;
Stereo9 = 86;
StereoCD = 87;
StereoRadio = 88;
StereoCasett = 89;
StereoAUX = 90;
StereoVolumeUp = 91;
StereoVolumeDown = 92;
StereoNext = 93;
StereoPrev = 94;
StereoF1 = 95;
StereoF2 = 96;
StereoF3 = 97;
StereoF4 = 98;

F1on = 100;
F2on = 101;
F3on = 102;
F4on = 103;
F5on = 104;
F6on = 105;
F7on = 106;
F8on = 107;
F1off = 108;
F2off = 109;
F3off = 110;
F4off = 111;
F5off = 112;
F6off = 113;
F7off = 114;
F8off = 115;

10 Appendix C: Kretsschema