AViRc – Embedded IRC

(EDI021 – Digital Projects – LTH)

Mattias Jernberg, dt06mj4@student.lth.se
Joakim Andersson, ic06ja9@student.lth.se

May 12, 2010

Abstract

In this paper we present a microcontroller based construction implementing the well known IRC[1] protocol. Using an ATmega128 microcontroller, an ENC28J60 ethernet controller, a PS/2 keyboard, an LCD and the uIP[2] software framework, we present the user with an easy to use, standalone chat client capable of operating without help from a normal PC.

[1] Internet Relay Chat

Innehåll

1 Introduktion

Detta projekt gjordes som en del av kursen digitala projekt. Målet i denna kurs är att planera och konstruera en fungerande prototyp baserat på en mikrokontroller alternativt en isp PLD[2].

Projektet som beskrivs i denna rapport är en chattklient för chattprotokollet IRC[3]. För detta valdes en AVR ATmega mikrokontroller på grund av dess integration av processor, minne och I/O-funktionalitet i ett chip. Dessutom är det en mycket populär hobby-plattform vilket gör det lätt att hitta färdig kod för plattformen.

IRC är ett chattprotokoll som härstammar från slutet av 80-talet. Det är i huvudsak uppbyggt kring gruppchatter i så kallade kanaler. Det grundläggande protokollet är mycket enkelt (delvis på grund av sin ålder) vilket kraftigt underlättar en implementation på mikrokontroller.

2 Kravspecifikation

Anslutning till nätverk ska ske via TCP/IP samt Ethernet. Det fysiska gränssnittet ska vara RJ45 Twisted Pair. Stacken ska klara av anslutning till nätverksadresser på både i och utanför det lokala nätverket (subnät). ARP-request ska kunna hanteras samtidigt som anslutningen är i bruk.

Mikrokontrollern ska (utan hjälp från annan källa) kunna koppla upp till en (1) IRC-server och visa meddelanden från en (1) kanal.

Utdata ska visas på en (monokrom) skärm med 128x64 punkter i läsbar text. På skärmen ska finnas en rad som alltid visas och innehåller kanalens namn och antalet personer i kanalen. Restrerande yta ska användas för att visa så många meddelanden som får plats av kanalens historik. Enbart meddelanden och ändringar av kanalens ämnesrad ("topic") ska visas. Övriga händelser ignoreras, förutom för att uppdatera antalet personer i kanalen.

Om tid finnes ska enheten även ges möjlighet att ta emot indata från ett PS/2-anslutet tangentbord. Det som ska kunna kontrolleras då är konfiguration vid uppstart (server, kanal, smeknamn och ev. IP-konfiguration) samt scroll i meddelandehistoriken (om minne till sådant finnes). Om bedöms möjligt (utifrån skärmutrymme, resurser och tid) kan även möjligheten att skriva meddelanden läggas till.

2.1 Prestanda

Enheten skall fristående (utan extern hjälp) klara av en låg-medeltrafikerad IRC-kanal. I detta fall krävs att kanalen #blausoffan på IRC-nätverket EFnet ska hanteras.

3 Hårdvara

Kärnan i hårdvaran består av en Atmel ATmega128 mikrokontroller. Till denna är ett färdigbyggt nätverkskort (ENC28J60-H) anslutet. Detta är en kombination av en ENC28J60 ethernetkontroller, 12.5 Mhz-klocka och nätverksuttag[6].

[2] Programmable Logic Device
[3] Internet Relay Chat

Detta kommunicerar till mikrokontrollern via SPI[4]-gränssnitt. Det finns en interrupt-pin från nätverkskortet för att signalera när paket har tagits emot eller skickats (denna pin är ansluten men används inte i mjukvaran). Det finns även en 12.5 Mhz-klocka tillgänglig. Även denna är ansluten men ej använd då obekräftade källor anger denna klocka som opålitlig för klockning av AVR[7].

För utdata används en 128x64-pixlars LCD display (GDM12864C[1] eller kompatibel). Här används generell I/O och mjukvara för att generera kommunikationsprotokollet som används för skärmen. Denna skärm har två stycken identiska kontrollkretsar som driver varsin halva (64x64 pixlar) av skärmen. För att spara I/O-pinnar användes en inverterare för att driva båda chip select-signalerna från en utsignal på mikrokontrollern, det är dessutom så få tillfällen då båda displaykontrollrarna ska ges samma instruktion att detta inte innebär någon overhead i mjukvaran.

Slutligen anslöts en PS/2-port hämtad från ett gammalt moderkort för att enkelt ansluta ett PS/2-kompatibelt tangentbord. Denna är ansluten till interrupt 1 för klockning, eftersom PS/2-anslutna enheter klockar sin data själv. Värdsystemet använder då denna klocka för att läsa dataströmmen, med hjälp av en interruptrutin.

4 Mjukvara

Mjukvaran är i stort uppbyggd kring uIP[2], en GPL-licensierad IP-stack primärt avsedd för mikrokontrollrar. Denna fanns sedan tidigare porterad till AVR tillsammans med en drivrutin för Ethernetkontrollern som används i projektet[3]. Denna fungerade med mindre modifikationer även på ATmega128. För att uIP ska fungera korrekt krävs tidtagning för att beräkna tid för återsändning, timeout för anslutningar och när uip ska polla applikationen om den har mer data att skicka. Denna funktionalitet var byggd för ATmega644 med en 8-bitars timer. För ATmega128 användes istället en 16-bitars timer med ytterligare logik för att hålla tiden i ett 32-bitars heltal. Själva uIP är uppbyggt av en main-loop som pollar ethernet-gränssnittet. Pollning här är inget större problem eftersom huvudarbetet i systemet är att hantera nätverkstrafiken. Detta underlättar i sin tur implementationen av uIP eftersom det undviker eventuella race conditions.

4.1 Skärm

För skärmen hade vi hopp om att kunna utnyttja tidigare projekts arbete och använda kod därifrån som bas för vår implementation. Dock stötte vi på samma problem som gruppen vi "stal" från, koden vi använde fungerade inte alls[4]. Efter mycket om och men konstaterades att denna kod inte fungerade och efter en full omskrivning fungerade skärmen med några timingfel. Dessa försvann inte oavsett justeringar i koden. Till sist utnyttjades ett annat projekt skrivet för en PIC-mikrokontroller och en liknande skärm[5]. Portningen av denna kod till C visade sig fungera utan problem, trots att implementation bröt mot timingen i det ursprungliga databladet[8]. Däremot identifierades en liknande LCD vars datablad angav den alternativa timing som vi använde[1].

[4]Serial Peripheral Interface

4.2 Tangentbord

För tangentbordsavläsningen används en interruptrutin som körs både när tangentbordet klockar lågt och högt. Rutinen gör dock olika saker i de båda fallen. Interruptrutinen ser till att buffra upp skrivna tecken i en FIFO[5]-kö för att sedan returnera dessa i ordning när programmet frågar efter det. För att konvertera de "scan codes", som tangentbordet skickar, till vanlig ASCII-kod används två enkla lookup-tabeller (för fallen med shift-tangenten uppe respektive nere). För tillämpningen att skicka meddelanden till IRC hanteras dock tangentbordet annorlunda, se 4.4 Utökad funktionalitet.

4.3 Protokollstack

Ovanpå den plattform som uIP erbjuder är sedan IRC-applikationen skriven. Detta har varit den stora implementationsdelen i projektet där IRC-protokollet ska hanteras. Detta görs i en flerstegsmodell som först kontrollerar om omsändning måste ske, detta av minnesbesparingsskäl då sänd och mottagningsbuffert inte är implementerat i uIP utan måste hanteras av applikationen. Därefter kontrolleras om inkommande data existerar. Inkommande data tolkas, tecken för tecken, med hjälp av en tillståndsmaskin som indikerar vilken del av IRC-kommandot som tolkas. Innan detta sker måste TCP-strömmen sättas samman genom att kontrollera det tidigare sparade tillståndet och identifiera om det finns tidigare använd data som måste användas vid tolkning. Efter att paketet tolkats kontrolleras återigen tillståndet för att avgöra om någon data måste kopieras från det mottagna paketet (då uIP kommer skriva över denna data när nästa paket tas emot).

Omedelbart efter att kommandot som indikerar typen av meddelande tagits emot kontrolleras om denna typ ska hanteras. Kommandot ersätts med en identifierare på en byte för att spara minne. Om kommandot inte ska hanteras utnyttjas ett tillstånd för att hoppa över meddelandet, vilket hoppar över all paketsammansättning och majoriteten av parsningslogiken, detta för att hantera stora mängder irrelevant data, såsom MOTD[6], snabbare.

När parsningen avslutas kontrolleras om något meddelande kräver någon form av uppföljning i sådant fall skapas och skickas detta meddelande innan IRC-applikationen returnerar kontrollen till uIP.

4.4 Utökad funktionalitet

Av den extra funktionaliteten implementerades konfiguration och skrivande av meddelanden till IRC med hjälp av tangentbord. Meddelanden till IRC-servern fungerar genom att en flagga för inmatning aktiveras. Tangentbordets interruptrutin känner av denna flagga och uppdaterar själv statusraden med meddelandet som skrivits. När man trycker Retur eller med hjälp av Backspace tömmer hela raden slås läget av och applikationen kan kontrollera detta tillstånd genom att polla keyboard-buffern i minnet. Pollning valdes av skälet att uIP pollar applikationen om ingen annan händelse inträffat och det är bara när kontrollen är

[5] First In First Out

[6] Message Of The Day, ett välkomstmeddelande som skickas när du ansluter till IRC-servern

inuti uIP-applikationen (till skillnad från i t.ex. en interruptrutin) som samtliga variabler är i det tillståndet att man säkert och korrekt kan skicka TCP-data.

Konfigurationen (IP-address, nätmask, gateway, DNS-server, IRC-server, kanal och smeknamn) läses med tangentbordsrutinerna beskrivna i 4.2. Speciella kontroller verifierar längd och format (på IP-addresser) som skrivs in. Inställningarna lagras även permanent i det EEPROM som finns i mikrokontrollern och läses in vid omstart för att ges som standardvärden vid konfigurationen. Därmed är det bara för användaren att bekräfta varje inställning istället för att upprepa varje inställning manuellt.

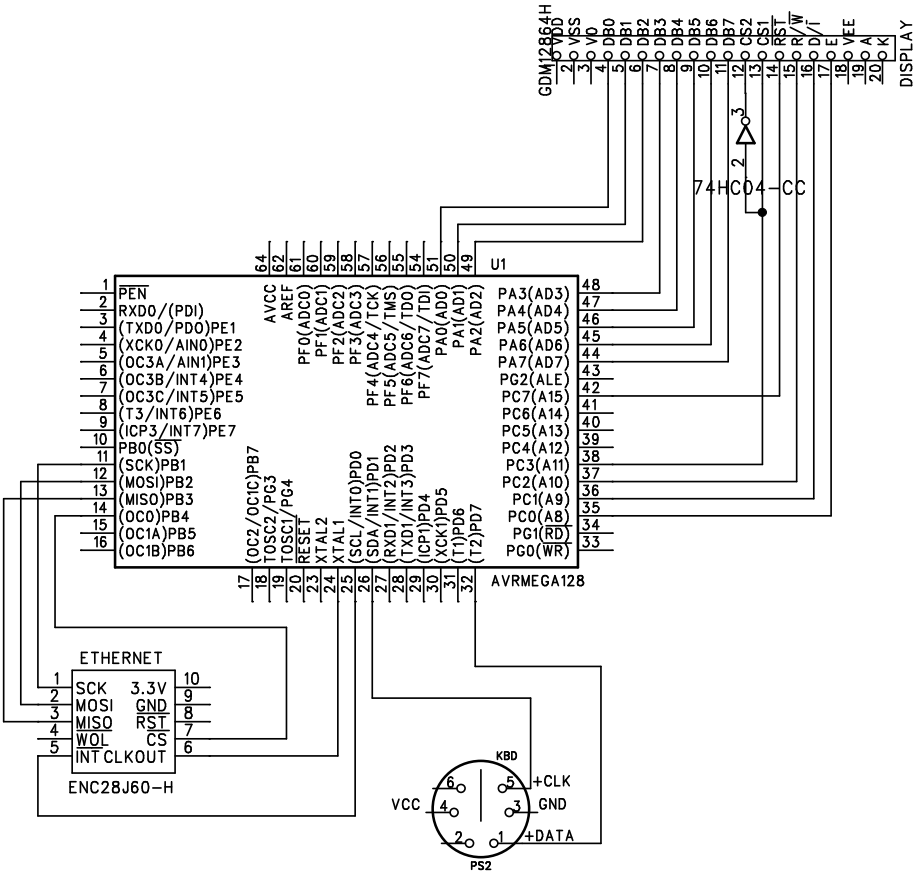
5 Resultat

Den slutgiltiga implementationen har blivit testad mot en verklig IRC-server och det har konstaterats att prototypen är mestadels fungerande och kan hålla kontakten med IRC servern under en längre tid (över 3 timmar). Den nuvarande implementationen har störst problem vid anslutning, teorin här är att sammanslagningen av paket inte är helt korrekt utan har logiska fel som uppstår i vissa situationer, som exempelvis om meddelandet är delat över mer än tre paket (detta innefattade en verklig bugg som fick systemet att starta om när man försökte ansluta till kanaler med många användare).

Tangentbordsrutinen för att skriva meddelanden till IRC-servern skapades relativt kvickt och har ett antal race-conditions vilket gör att fel tecken skrivs om man skriver in tecken för fort. Detta medför bland annat att man av misstag kan skicka meddelanden till IRC-kanalen (vilket är vanligast om man suddar ut hela meddelandet genom att hålla ner backspace, varvid ett race condition kommer skriva teckenkoden för backspace över NULL-terminator på strängen vilket resulterar i ett oönskat meddelande till servern).

Har man väl lyckats ansluta till servern och använder pekfingervalsen för att skriva fungerar dock systemet väldigt bra, uppfyller kravspecifikationen och uppför sig (nästan) helt som man förväntar sig.

A Kopplingsschema



Figur 1: Kopplingsschema för systemet

References

[1] *GDM-12864C.* http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/GDM12864C.pdf.

[2] Adam Dunkels. uIP. http://www.sics.se/~adam/uip/index.php/Main_Page.

[3] Adam Dunkels, Pascal Stang, Jonathan Webcom, jopsen, and jderehag. Avr-uip. http://code.google.com/p/avr-uip/.

[4] Thomas Eriksson and Samuel Skånberg. UDPong - pong over UDP. http://www.eit.lth.se/fileadmin/eit/courses/edi021/Sammanfattning/2010/LP3/grupp1/udpong_rapport.pdf.

[5] Andrew Gehringer. Pong clock app note. http://www.agehringer.com/pong-clock.

[6] Olimex Ltd. *ENC28J60-H.* http://www.olimex.com/dev/pdf/ENC29J60-H.pdf.

[7] Guido Socher. An avr microcontroller based ethernet device. http://www.tuxgraphics.org/electronics/200606/article06061.shtml.

[8] Xiamen Ocular Optics Co.,Ltd. *GDM12864HLCM.* http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/GDM12864_lawicell.pdf.