Projekt

F1 Röj

Studenter:
Simon Frennberg
Simon Rundstedt
Jonatan Kåhrström

Handledare:
Bertil Lindvall

Abstract

The goal of this project was to construct a small prototype for a real-life sized minesweeper game. This was done using several ATMega16 microcontrollers, communicating amongst each other using a Two Wire Interface (TWI), and with a computer via a control unit. Each microcontroller received input via sensors in the form of switches.

The result was mostly a success, although somewhat unpolished. Some improvements are still needed in the communication between the master chip and the remaining microcontrollers.

As this project is only part of a much larger commitment, the final results can be expected sometime early this fall.

Innehåll

1 Bakgrund

Under år 2010 var det ett Karnevalens år i Lund. F-sektionen hade fått en tältplats inne på karnevalsområdet och var tvungen att fylla den med något. Något mindre än något häftigt var inte att tänka på. Ett ambitiös idé var att bygga ett Minesweeper i en sådan storlek att man kunde gå på och själv sätta ner flaggor på minerade rutor. Av olika anledningar byggde man fjärrkontrollerade AVR-styrda legobilar med tillhörande hinderbana istället.

Planerna på att bygga ett Minesweeper hade dock inte gått i graven. Taggade av de positiva erfarenheterna från legobilsbyggandet och den gnagande känslan av att inte fått utlopp för sina idéer gjorde att ett litet sällskap F-are fortsatte planera ett minesweeper.
- Teknisk Fysik fyller femtio under 2011, man skulle kunna ha en stortävling mellan massa teknologer från hela Sverige då - tänkte F-arna. Vips, så hade en arbetsgrupp bildats inom organisationen för 50-årsjubiléet. Av en slump fick F-arna reda på att det fanns en kurs vid namn *Digitala och Analoga projekt*. Detta medförde att man kunde få lite lärarhjälp och högskolepoäng när prototyperna utvecklades! Det tog inte lång tid förrän tre F-teknologer av varierande årskurs skrivit upp sig på kursen för att genomföra `Projekt F1 Röj`.

2 Design av spelet

2.1 Inledning

Målet med detta projekt under kursen är inte att bygga ett stort fungerande Minesweeper utan att designa hur varje ruta på spelplanen samverkar med de andra och en central dator. För detta behövs inte hundra plattor utan en handfull enkla prototyper samt att all teknik skall vara skalbar så att inget behöver ändras när spelplanen blir större. Dock innehåller kravspecifikationerna en del krav som är baserade på att hundra plattor skall byggas och att de skall ta stora belastningar.

2.2 Specifikationer för `F1-Röj`

2.2.1 Plattorna

Planen är att under projektets gång tillverka 9 plattor, skulle det visa sig att vi har gott om tid så kan vi givetvis göra fler. En av dessa plattorna kommer vara en specialplatta, i vilken det kommer finnas en modul för att kommunicera med dator, samt strömförsörjning för plattorna. Minst en av de andra plattorna skall också vara en färdig prototyp, med stor display och fullt fungerande sensorer. De restrerande plattorna kan antingen vara färdiga plattor, eller prototypplattor med liten sjusegmentsdisplay och vanliga tryckknappar som sensorer.

2.2.2 Chip

Ett AVR-chip av billigast möjliga typ används, exempelvis ATMEGA16. Det behöver minst 11 pinnar, 2 för kommunikation, 2 för trycksensorer och 7 för att styra displayen.

2.2.3 Kommunikation mellan plattor

Kommunikationen mellan plattor sker med tvåtrådskommunikation, så som beskrivs i manualen för atmega16. Det inbyggda systemet för tvåtrådskommunikation möjliggör upp till 127 sammankopplade plattor. Vill man utöka detta antal kan man låta en dator styra flera nätverk av plattor.

2.2.4 Display

Varje platta ska ha en 7-segmentsdisplay. Inom projektet skall minst två ”stora” displayer byggas, hur stor denna blir är en fråga om strömförbrukning. Övriga plattor får små, färdiga, 7-segmentsdisplayer, men om tid finnes byggs stora displayer till alla. Hur displayen implementeras beror lite på strömförbrukningen, om det inte innebär för stora problem och för stora kostnader kan en variant där de olika segmenten blinkar mycket snabbt implementeras för att sänka strömförbrukningen. Eventuellt kan även ett åttonde grönt segment implementeras för att på ett pedagogiskt sätt visa att plattan är säker att gå på.

2.2.5 Sensorer (knappar)

Varje platta har två sensorer. En tryckknapp som känner av om någon står på plattan, samt en tryckknapp som känner av om en flagga placerats i plattan. Flaggknappen bör befinna sig i ett hål i plattans mitt, och då någon placerar en flagga i hålet trycks den ner. Den skulle kunna göras mer avancerad för att låta spelet hålla reda på vilken färg flaggan som placerats i plattan har. Tryckknappen, eller lastcellen, som kontrollerar om någon står på plattan, kräver ett visst utvecklingsarbete. Den måste vara känslig, så att även personer som inte väger så mycket kan delta i spelet. Den måste också vara robust, så att den tål tunga personer och inte ger falskt utslag. Slutligen måste den vara billig, så att kostnaden för att tillverka 100 st inte blir enorm. En tanke är att ha två foliebeklädda ytor, med några millimeters mellanrum. Då en person kliver på plattan pressas de ihop och ger en signal. En annan variant är att ha en eller flera bara koppartrådar som kan pressas ihop och ge signal. En sista möjlighet är att ha någon tryckknapp som pressas ned då någon går på plattan.

2.2.6 Datorinterface

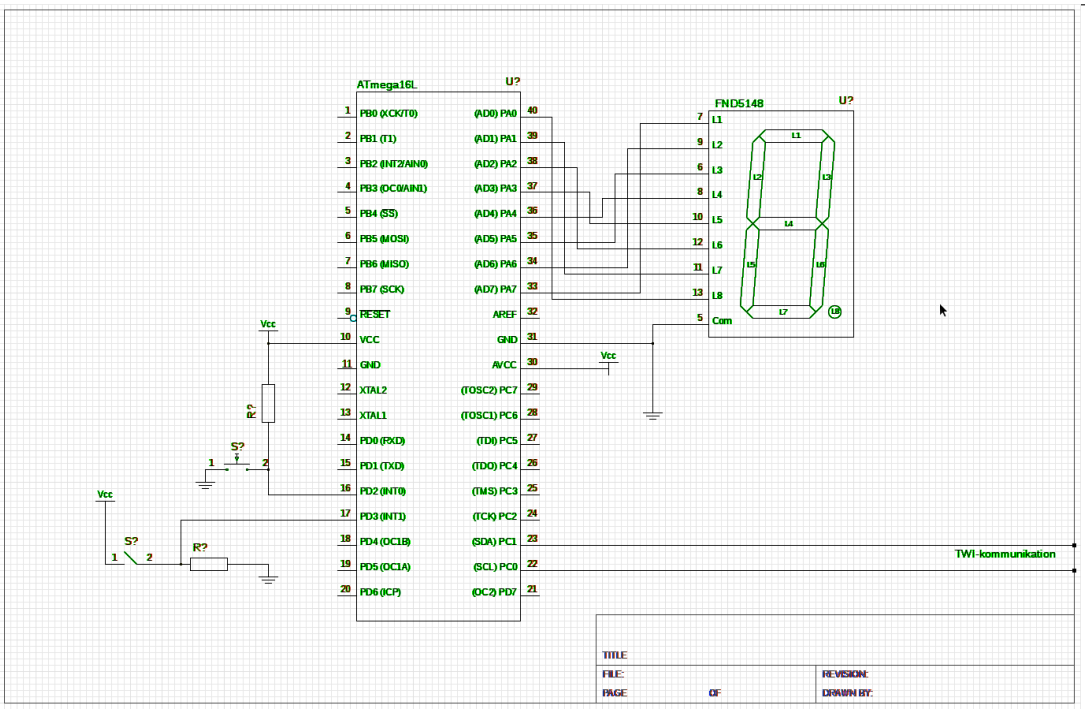
Ett fullständigt datorinterface som kontrollerar spelplanen. Det ska kunna användas till att sätta upp spelplanen, starta ett spel (möjligtvis olika speltyper) samt visa en bild av spelplanen på projektor, för att göra det lätt för publik att följa spelet. Även ljudeffekter (explosioner då någon trampar på en mina) är önskvärt. Datorn kommer också vara ansvarig för att styra hela spelet. Det innebär att den kommer generera spelplanen och tala om för alla plattor om de är minor eller inte. Beroende på hur kommunikation utformas kan mer eller mindre kommunikation gå genom datorn. Då en platta som inte gränsar till några minor tryckts ner kan antingen den plattan tala om för sina grannar vad som hänt, eller så talar plattan om för datorn att den blivit nertryckt och sedan talar datorn om för de plattor som ska tryckas ner att de ska det.

2.2.7 Dator/chipkommunikation

Minst en platta måste ha möjligheten att kommunicera med en dator. Detta sker lämpligtvis med en särskild Atmega16 som översätter tvåtrådskommunikationen från de andra plattorna till R232-(seriell) kommunikationen som sedan går till en dator via en USB-seriellmodul.

2.2.8 Strömförsörjning

Beroende på hur de stora displayerna designas behövs ett lämpligt nät för strömförsörjning av spelet. En idé är att driva spelet med 12 volt, och låta spänningsregulatorer på varje platta sänka spänningen för



Figur 1: Blockschema över en slavplatta

att passa AVR-erna. En annan idé är att ha ett system på 6 volt, och ett tredje är att ha två separata system för displayerna respektive chipen.

2.2.9 Pris

Då planen är att så småningom producera 100 plattor bör priset hållas så lågt som möjligt, helst under 100 kr. Eventuellt kan sponsring sökas.

2.3 Blockscheman

Blockscheman för slavplatta, styrplatta och stor 7-segmentsdisplay kan ses i figur 1, figur 2 samt figur 3.

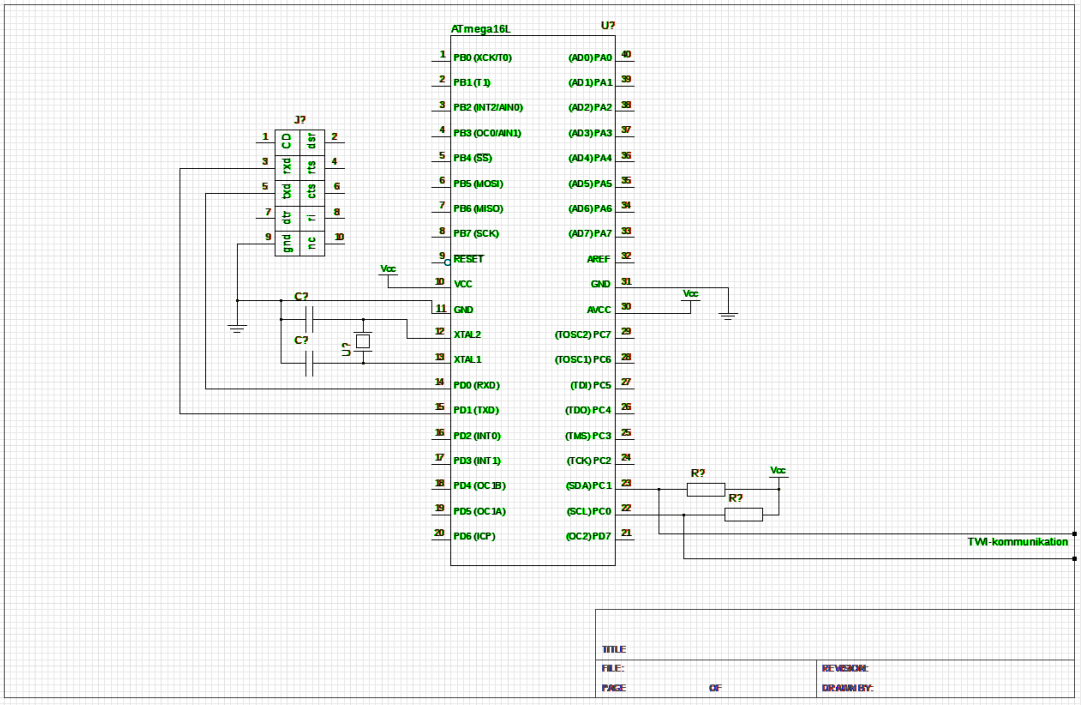
3 Hårdvara

3.1 Inledning

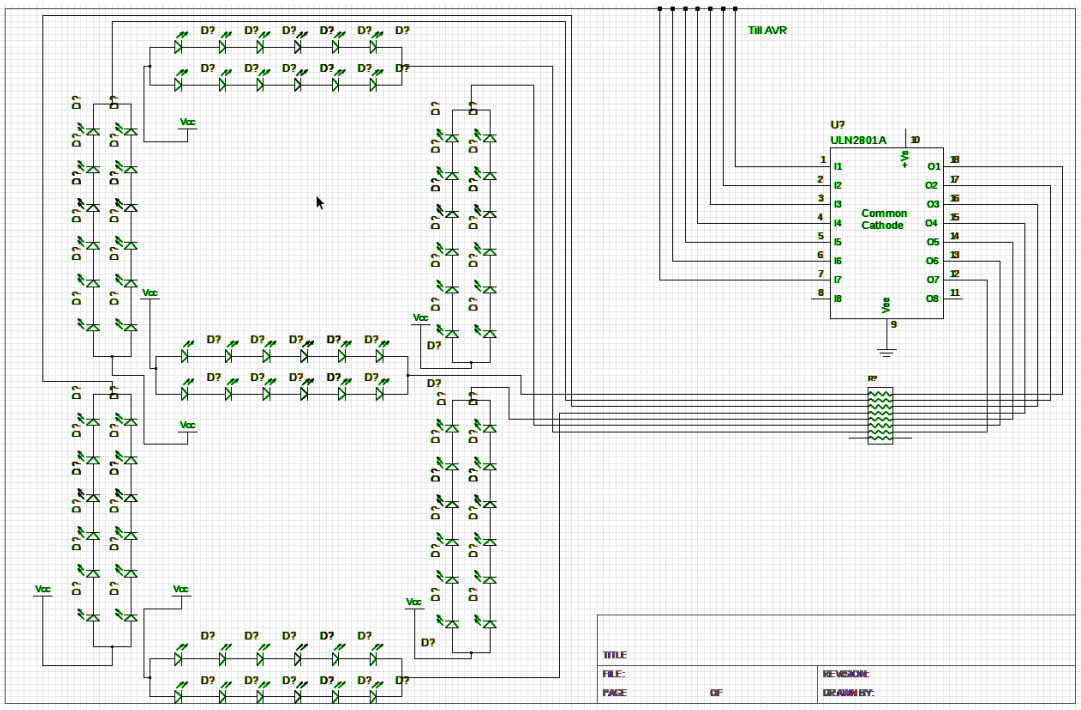
Hårdvaran för detta projekt är relativt enkel och designades och byggdes därför ganska fort utifrån kravspecifikationerna.

3.2 AVR

Spelplattan designades på enklaste sätt. Atmega16 är kraftfullare och har betydligt fler I/O-pinnar än vad som krävdes av oss. Den har fyra portar med vardera 8 pinnar, av vilka vi använde oss av totalt 12. I projektet användes en hel port (port A) för att styra en 7-segmentsdisplay (med decimalpunkt), de två pinnarna SDA och SCL för tvåtrådskommunikationen, samt ytterligare två pinnar för sensorerna. Även



Figur 2: Blockschema över en styrplatta



Figur 3: Blockschema över stor 7-segmentsdisplay

prestandamässigt ligger Atmega16 i överkant då inte mer än ca: 5-10% av programminnet användes till slavplattorna och ca: 20% till styrplattan.

3.3 USB-modul

Kommunikationen mellan dator och AVR sker med en USB-modul kallad UM232R, som agerar som en virtuell seriellport. USB-modulen ansluts till AVRen med två pinnar, kallade TxD och RxD, där TxD på USB-modulen ansluts till RxD på AVRen och vice versa. USB-modulen ansluts också till samma jord och spänningskälla som AVRen.

För UART-kommunikation behövs en stabil klockfrekvens för att AVRen ska kunna läsa och skicka meddelanden utan fel. Den interna klockan på AVRen anses inte tillräckligt stabil och exakt, så därför används en extern kristall för att ge en stabil klockfrekvens. I uppställningen används en kristall med frekvensen 4,9152 MHz, då detta är en jämn multipel av 9600, vilket är överföringshastigheten i bits per sekund. Kristallen ansluts till AVRen i de två ingångarna för externa oscillatorer (XTAL 1 och 2), som också ansluts till jord via två kondensatorer.

3.4 Sensorer

Sensorerna skall detektera ifall en flagga är nersatt i plattan och ifall någon står på dem.

För att simulera en nersatt flagga i prototypen användes en ON/OFF-switch tillsammans med en pull-up-resistor. En vanlig tryckknapp med pull-up-resistor användes för att detektera om någon står på plattan. Implementeringen kan ses i figur 1.

Antistudshantering sker via mjukvaran.

3.5 Display

För att en spelplatta skall kunna visa hur många minor den angränsar till måste den kunna visa siffrorna 1-8 samt någon extra symbol för att visa en mina. För detta jobbet är en vanlig sjusegmentsdisplay lämplig. Den implementeras dessutom lätt med lysdioder när prototypstadiet skall lämnas. För att kunna visa att en platta inte angränsar till en mina vore ett åttonde segment i en annan färg att föredra även om det för prototyperna räcker med att kunna visa en nolla.

Den sjusegmentsdisplay vi använts oss av, en Hewlett-Packard HDSP-5303 [2] med gemensam katod, inkluderar ett åttonde segment i form av en decimalpunkt. Under byggandet har decimalpunkten använts för att ange om en flagga sitter i platta och siffran noll för att ange att en platta inte angränsar till minor.

För att driva sjusegmentsdisplayen användes strömmen från Port A som i sin tur får sin försörjning från pinne AVCC.

Sjusegmentdisplayen är kopplad till Port A med ett resistornät emellan för att skydda komponenterna. Port A:s MSB motsvarar segment A och LSB decimalpunkten.

3.6 Strömförsörjning

Prototypplattorna med vanlig sjusegmentsdisplay klarar sig bra med den $5V$-spänningsgenerator som fanns i laborationssalen.

Lysdiodsdisplayerna för den färdiga plattan skall drivas med en $12V$- spänningskälla då sådana är lätta att ordna fram. En spänningsregulator drar ner spänningen till en nivå AVR klarar av och en kondensator ser till att spänningsnivån hålls stabil.

4 Mjukvara

4.1 Inledning

Medan hårdvaran designades och byggdes fort, mestadels utan svårigheter, hade de flitiga studenterna stora problem med mjukvaran. Det stod från början klart att mjukvaran skulle bli det mest tidskrävande under projektet. Då många av problemen härstammar ur kommunikationssvårigheter mellan de olika microchipen har stor möda lagts på ett gemensamt bibliotek för kommunikation. För att så lätt som möjligt kunna uppdatera spelet eller till och med kunna byta mellan olika spel försökte spelplattan göras så dum som möjligt.

Spelplattan bedömdes behöva kunna:

- Berätta när en flagga sätts ner/tas bort.
- Berätta när någon ställer sig på plattan.
- Kunna visa en siffra.
- Ta emot information som vilken siffra som skall visas.

4.2 Seriell AVR-till-dator-kommunikation

4.2.1 Teori

För att kommunicera mellan dator och AVR utnyttjas AVRens USART-modul, där USART är en förkortning av ”Universal Synchronous/Asynchronous Receiver/Transmitter”. Denna modul kan användas för att kommunicera både synkroniserat och osynkroniserat, men för projektet behövdes inte de extra hastigheter som synkroniserad kommunikation ger, varför det asynkrona kommunikationssättet användes.

Asynkron seriell kommunikation sker enligt ett protokoll som kallas RS232 (Reccomended Standard 232), som är en standard som beskriver spänningsnivåer, start- och stopbitar med mera. Ett sätt att kommunicera via detta protokollet med en dator hade varit att ansluta till en seriellport, men då de flesta moderna datorer saknar sådan så valdes istället att använda en USB/rs232-modul kallad UM232R. Denna ansluts via RxD- och TxD-pinnarna till AVRens motsvarigheter, samt till VCC och jord, och därefter skapar den en virtuell seriellport på datorn som kan användas exempelvis med PuTTY eller Javas bibliotek för seriell kommunikation

4.2.2 Implementering

För att skicka data mellan dator och AVR utvecklades ett enkelt protokoll. I protokollet består varje meddelande av 4 bytes: en startbyte, två databytes och en stopbyte. Så fort mottagaren upptäcker att protokollet av någon anledning inte följs skickas ett felmeddelande och därefter väntar mottagaren på nästa meddelande (startbyte). Protokollet är relativt stabilt, och det värsta som kan hända är att mottagaren missar ett meddelande.

För att ta emot och skicka meddelanden från datorn används javabiblioteket ”Java Communications API”, även känt som ”javax.comm”. Detta inkluderar input- och outputbuffrar och enkla funktioner för att skicka och ta emot meddelanden. Då java envisas med att enbart använda signerade datatyper vore det opraktiskt att skicka bytes, då dessa i java går mellan -128 och 127, och i C (som används på AVRen) går mellan 0 och 255. Lyckligtvis finns en metod i javax.comm för att skicka heltal. Då heltal i java är större än 1 byte så kapas all data utom den minst signifikanta byten när heltalet skall skickas, vilket innebär att genom att kommunicera med heltal mellan 0 och 255 så är det möjligt att ha samma representation av meddelandena i både java på datorn och C på AVRen.

På AVRen utnyttjas ett UART-bibliotek som skrevs av Peter Fleury och Nicholas Zambetti 2006, och uppdaterades för att passa nyare AVRkod av Tim Sharpe. Detta bibliotek innehåller, likt javas bibliotek,

buffrar för både in- och utgående data samt funktioner för att enkelt hantera dessa. Skickande och mottagande av data sker med hjälp av avbrott.

Meddelandena som skickas mellan dator och AVR kan delas in i två kategorier, normalameddelanden samt status/felmeddelanden. I de normala meddelandena så utgörs de två databytesen av först en adressbyte, med adressen till plattan som skickade eller ska ta emot meddelandet, samt en meddelandebyte som innehåller det meddelande som plattan skickat eller ska ta emot. Adressbyten har alltid ett värde mellan 2 och 127, då TWI-protokollet enbart tillåter adresser mellan 1 och 127, och TWI-enheten med adress 1 är den som sköter kommunikationen med datorn. Exempel på meddelande är att datorn talar om för plattan att den ska visa en viss siffra, eller att plattan talar om för datorn att någon trampat på den (tryckt på knappen).

Den andra kategorin av meddelande innehåller först en meddelandebyte, med ett värde mellan 128 och 255, samt en andra byte som innehåller extra information som till exempel en adress. Dessa används för att skicka statusmeddelanden, som till exempel att en platta inte svarar på TWI-kommunikationen.

4.3 TWI

4.3.1 Teori

TWI, eller I2C som det också kallas, är ett sätt att kommunicera mellan olika enheter, däribland AVRer, över endast två trådar (TWI är en förkortning för Two Wire Interface). De två trådarna i TWI utgörs av en datatråd och en klocktråd. All data skickas på datatråden, och klocktråden används för att skicka pulser som anger när data skickas på datatråden.

TWI-enheter kan arbeta i fyra olika lägen, Master-transmitter, Master-reciever, Slave-transmitter och Slave-reciever. I detta projekt har endast Master-transmitter och Slave-reciever användts. När en TWI-enhet arbetar i Master-transmitterläget så börjar den med att ta kontroll över databussen genom att skicka ett startmeddelande. Om två eller flera enheter försöker ta kontroll över databusssen samtidigt sker något som kallas arbitration, vilket innebär att den ena enheten får kontrollen och den andra enheterna får ett felmeddelande.

När en enhet fått kontrollen över databussen adresserar den den enhet som den vill skicka data till. Den adresserade enheten går då in i Slave-recvieverläget, och gör sig redo att ta emot data. Master-.transmitterenheten kan därefter skicka en eller flera byte till recievern, och när den skickat allt den vill så skickar den ett stoppmeddelande varpå databussen blir ledig igen.

Efter varje skickad databyte så svarar mottagarenheten med en “acknowledge-bit”. Detta gör att transmittern vet om receivern fått meddelandet eller inte, och kan vidta lämpliga åtgärder om receivern inte skulle svara.

4.3.2 Implementering

Då all TWI-kommunikation sker mellan dator-AVRen och de olika plattorna skrevs två olika metoder för att skicka data, en för dator-AVRen och en för övriga plattor. Metoderna är ganska enkla och liknar de som beskrivs i AVR-manualen. I korthet går den ut på att AVRerna utför ett kommando, som att skicka ett startmeddelande, en adress eller en databyte, och därefter väntar på statusmeddelande. Då den fått statusmeddelande läser den detta och beroende på vad det är för status så agerar den därefter. Om dator-AVRen inte får svar från en platta så skickar den exempelvis ett felmeddelande via UARTen till datorn, och om någon enhet misslyckas med att ta kontroll över databussen så väntar den antingen tills bussen blir ledig, eller, om den själv blev adresserad, går in i slave-recieverläget.

Då en AVR blir adresserad som slave får den ett avbrott, varefter den tar emot data och lägger meddelandet i ett osignerat heltal, som är två byte på AVRen. Detta fungerar bra då datamängden som skickas mellan AVR och dator än så länge är relativt liten. I framtiden, då det kommer tillkomma fler AVRer, kommer förmodligen en variant med buffer implementeras, framförallt på dator-AVRen, då den kommer kommunicera med många andra AVRer och inte kan riskera att missa meddelanden.

4.4 Avbrottshantering

4.4.1 Teori

AVR:en innehåller inbyggd hantering av interrupts, det vill säga hantering av avbrott vid vissa händelser. Genom styrregistren kan man kontrollera vilka händelser som skall generera ett avbrott, tex om en I/O-pinne blir låg, om den byter värde eller om ett meddelande anländer via TWI.

4.4.2 Implementering

Avbrottshantering aktiveras och AVR:en försätts i sleep mode. När ett avbrott generas väcks AVR:en och en särskild av avbrottsrutin aktiveras beroende på vilken typ av avbrott som genererats [1, s.43]. Lämpliga åtgärder vidtas och ett meddelande som anger vad som skedde placeras i en buffert. När hanteringen av avbrottet är klart skickas den data som lagrats i bufferten.

Vid hantering av avbrott från knapptryckningar måste särskild hänsyn ske till de studsar som kommer att uppstå. Detta har gjorts genom att frysa exekveringen av kod under $50ms$ vilket räcker för att studsarna skall hinna avta.

4.5 Datormjukvara

4.5.1 Implementering

Datormjukvaran består av två delar, dels en kommunikationsdel som sköter den seriella kommunikationen mellan dator och AVR. Denna beskrivs ovan i 4.2. Den andra delen är spellogiken, som består av spelregler samt ett förnster för att visa en spelplan på datorn. Då att programmera MS-röj från början ansågs vara att uppfinna hjulet för 100ade gången, hämtades en enkel variant av spelet från internet [3].

Spelet modifierades så att alla metoder som uppdaterar spelplanens utseende, exempelvis genom att skriva siffror i plattorna, kontrollerar om plattan även finns representerad i fysisk form. Om så är fallet skickas även ett meddelande till den berörda plattan. Spelet gjordes också körbart, med en körmetod som väntar på indata från den fysiska spelplanen, och sedan tolkar denna och anropar de metoder som datan motsvar.

Kommunikationen från plattorna till spelet implementerades med hjälp av en klass som döptes till AVRQueue. Den innehåller en länkad lista, och två synkroniserade metoder för att hämta och lägga till meddelanden i listan. Då UART-delen av programmet erhåller ett meddelande från styrplattan lägger den meddelandet i listan, varpå speldelen av programmet görs uppmärksam på detta. Därefter läser speldelen meddelandet, och vidtar lämpliga åtgärder. Åt andra hållet skickar speldelen meddelanden direkt till UART-delen, utan mellanliggande buffer, som sedan skickar meddelandena vidare till styrplattan.

5 Slutsats

5.1 Diskussion

Under projektet har många stora problem kring byggandet av ett Minesweeper lösts. Hårdvarans layout har visat sig fungera mycket bra, mjukvaran har visat att den kommer att fungera.

Trots alla stora framsteg som gjorts så finns det fortfarande många problem som är kvar att lösa. Mjukvarumässigt handlar det mycket om att anpassa datorns GUI, få kommunikationen att bli stabilare samt göra små ändringar i kommunikationsprotokollen för att visa rätt symboler vid rätt tillfällen.

Hårdvaran har visat sig fungera bra och det som kommer att ändras där är att plattan skall drivas från $12V$ med en spänningsregulator samt få en kondensator för att stabilisera inspänningen till AVR:en Desstom är koppling mellan plattorna inte helt färdigdesignad.

5.2 Framtiden

Under projektet har vi hittills inte lagt ner någon nämnvärd möda på att designa en stor riktigt platta som man faktiskt kan kliva på. Så som det ser ut nu kommer den att byggas av trä med en sviktande plywoodskiva som tryckknapp. Lysdioddisplayen kommer att ligga nedsänkt i samma skiva så att den inte trampas sönder.

Även om vi hela tiden haft kostnaden för ett hundratal plattor i bakhuvudet så är kostnadsbilden inte helt klar. Detta beror på delvis på att ingen uppskattning av kostnaderna för själva träplattan gjorts, att vi inte vet hur mycket spons vi kan få eller att vi inte letat upp de billigaste komponterna i vissa fall.

Sammanfattningsvis så har de största problemen lösts och vi är övertygade om att `Projekt F1 Röj` kommer att lyckas. Många timmars arbete ligger fortfarande framför oss[1] i och med att många fler plattor skall byggas.

Referenser

[1] *ATmega16* - Dokumentation för ATmega16 från Atmel 2003

[2] Dokumentation för Hewlett Packards HDSP-5303 7-segmentsdisplay

[3] *Minspel* - `http://www.geekpedia.com/code135_Minesweeper-Game-In-Java.html`

[4] *AVR-bibliotek* - `http://www.nongnu.org/avr-libc/`

[5] *UART-bibliotek* - `http://beaststwo.org/avr-uart/index.shtml`

[6] *Javaxx.comm* - `http://www.oracle.com/technetwork/java/index-jsp-141752.html`

[7] *Dokumentation usb-modul* - `http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UM232R.pdf`

[8] *Dokumentation transistor-array* - `http://www.st.com/stonline/books/pdf/docs/1536.pdf`

[1] Har du alltid viljat vara med och löda 10000 lysdioder så skall du inte tveka att ta kontakt med projektgruppen för `Projekt F1 Röj`!