Pong Mayhem

Innehållsförteckning

Innehållsförteckning 1

Inledning 1

Kravspecifikation 2

Komponenter/Hårdvara 3

Arbetsprocess 5

Reflektion 8

Appendix 9

Inledning

Bakgrund

Digitala projekt (EITF11) på *Lunds tekniska högskola* är en kurs där studenterna skall få en introduktion till konstruktionsarbete. En stor del av detta inbegriper att tyda datablad och tillgodogöra sig information. Studenter får fritt välja ett projekt, inom rimliga ramar, som skall genomföras under kursens gång med stöd av en handledare. Projeket skall innefatta design av både hårdvara och mjukvara. Syftet är att illustrera industriellt utvecklingsarbete, ge förståelse för hur man tyder datablad och att öva på att driva ett konstruktionsprojekt.

Problemformulering

Vi har tänkt göra ett arkadspel med Pong för en eller två spelare. En ensam spelare kan välja på tre svårighetsgrader. Man spelar först till fem poäng och poängställningen indikeras av lysdioder på var sida om skärmen. För detta kommer det att behövas två joysticks, en display, processor med ett litet minne, en lite större labbplatta, två knappar, åtta lysdioder (två i en färg, förslagsvis röd, och sex i en annan, förslagsvis vit) och en strömbrytare.

Vi kommer alltså både behöva programmera själva spelet (inklusive motspelares AI) i C, bygga ihop hårdvaran och få det att fungera tillsammans.

Vår arbetsprocess kommer att sammanfattas dels i den här rapporten samt i ett muntligt framförande. Vi kommer även att föra en loggbok för att hålla koll på hur projektet fortskrider.

Kravspecifikation

Funktionella krav

Spelet ska i stort sett fungera som det klassiska Pong-spelet, d.v.s. innehålla en studsande boll och två motstående paddlar som kan röra sig i sidled och tävlar om att inte släppa förbi bollen.

Spelarna ska kunna styra sin paddel med en analog joystick som alltså reagerar olika mycket beroende på joystickens utslag.

Därtill har varje spelare en knapp som kan användas när spelarens röda lampa är tänd. Denna knapp utlöser en specialattack, “Mayhem”, som får motspelarens paddel att tappa balansen. Efter en viss tid återhämtar man sig, och lampan tänds på nytt.

Spelet spelas bäst i matcher om bäst av tre, och varje gång en spelare tar ett poäng tänds en av dennes vita lampor. Vid tre tända lampor vinner spelaren och får en lysdiodsfanfar. Sedan återställs spelet.

Det skall också finnas en menyskärm före varje match där spelaren får välja att:

1. spela mot AI
2. spela mot en annan spelare

Låga prioritetskrav

Dessa funktionella krav är inte obligatoriska utan vi valde att försöka uppfylla dem i mån av tid.

- Ljud
- Startupskärm före menyskärm
- Fräcka effekter för Mayhem-knappen
- Monteringslåda för en riktig arkadspelslook

Det skulle visa sig att två av dem lyckades uppfyllas, det vill säga Startupskärm och Monteringslåda.

Tekniska krav

- Spelet ska ha en upplösning på 128 x 64 pixlar.
- Programmet skall inte överskrida 32kB i minnesåtgång.
- Spelet ska flyta på med en tillfredsställande bilduppdateringshastighet.

Komponenter/Hårdvara

ATMega32 - Processor

Denna processor är en CMOS 8-bits mikroprocessor som baseras på Atmel AVR-arkitektur. Den har 32 kB inbyggt flashminne och 40 pins vilken gör den lämplig för små program som kräver flera ytterligare komponenter. Dessutom har den en Interrupt-funktion för våra Mayhem-knappar och AD Converter för våra joysticks.

GDM12864HLCM - LCD Display

Enkelfärgad display på 128 x 64 pixlar med två olika chip för höger och vänster skärmhalva. Dessutom har skärmen *backlight* och justerbar kontrast.

Båda chip är uppdelade i 8 pages i x-led och 64 addresser i y-led. På så sätt kan block om 8 pixlar väljas och styras med hjälp av 8-bitars datasignaler.

Instruktionerna till skärmen ges med Write- eller Read-signaler, vilket bestäms enligt signalerna till RS och RW.

Atmel JTAG ICE mkII

Utvecklingsverktyg för att testköra programkod.

1. Exekverar med målsystemets kristallfrekvens (upp till 16MHz)
2. Emuleringsminne 16kb
3. Många brytpunkter i program
4. Högnivådebugger
5. Programmering i C eller assembler.

Till detta användes mjukvaran Atmel Studio för att styra enheten via dator. Vi använde oss huvudsakligen av Atmel Studio 4 eftersom vi hade problem med att få Studio 6 att fungera.

Joysticks - Styrenhet

Generisk analog joystick. Spaken reglerar två olika potentiometrar vilket möjliggör analog styrning i två plan. Två joystick har använts, men enbart kopplats in så att en axel fungerar. För att de här ska fungera behöver processorn en AD Converter för att kunna omvandla analoga signaler till digitala värden.

Knappar - Styrenhet

Generiska Push 'N' Close-knappar. Vi har använt två stycken. För att de här ska fungera optimalt behöver processorn två pins med Interrupt-funktion så att processorn inte själv behöver lyssna efter att en signal ändrats. Interrupt-funktionen ser till att processorn meddelas varje gång en knapp trycks in.

LED-lampor

Generiska led-lampor. Vi har använt 8 st, varav två röda och resten vita. Behöver rätt strömstyrka för att lysa väl, vilket fås genom användandet av en resistor.

Resistor

Åtta stycken generisk resistorer på 220 Ohm används för att reglera spänningen in till lysdioderna.

Variabalt motstånd

En generisk variabel resistor för att kunna justera skärmens kontrast.

Monteringslåda

En slutgiltig förfining av arkadspelet är såklart att bygga in det i en monteringslåda. Ursprungligen monterades allt på ett kopplingskort.

Strömförsörjning

Kretsen är ansluten till en strömkälla på 5V och 5A.

Arbetsprocess

Kopplingsschema

Arbetet inleddes med att konstruera ett enkelt blockdiagram för att få en bild över vilka komponenter som skulle behövas. Det vi kom fram till var dock inte tillräckligt tillfredsställande i tekniskt avseende, varför vi behövde gå tillbaka till ritbordet och

skissa upp ett riktigt kopplingsschema för att visa att vi förstod vad vi skulle göra av alla komponenter. Det visade sig också vara en aning mer komplicerat än att bara koppla in skärm i processor och så vidare. Det fordrade därför en djupdykning i databladen för att begripa vilka pins som skulle kopplas vart. Samtidigt påbörjades kodning av Pong i Java för att underlätta koddesignen när vi väl skulle börja koda i C.

Ett utförligt kopplingsschema ritades tillslut upp med hjälp av programmet PowerLogic (se bilaga). Överlag behöver kretsen en drivspänning på 5 V, men vissa komponenter opererar med andra spänningar. Därför behövdes en uppsättning resistanser, vars resistansbelopp bestämdes i enlighet med *Ohms lag*. Lite extra resistans lades till för lamporna för att försäkra oss om att de skulle lysa tydligt.

Den viktigaste detaljen i det här momentet var att förstå vilka extrafunktioner vissa portar på logikchippet hade och vilka av dem vi behövde använda oss av. Knapparna kopplades till två pins på INT-porten (interrupt) för att programmet inte ska behöva aktivt lyssna efter när någon knapp trycks in eftersom det skulle kunna göra att knapptryckningar inte registreras. Joysticken kopplades in till två pins på ADC-porten (analog-to-digital converter) för att kunna digitalt tolka analoga strömvarianser. Dessutom verkade displayen, av databladet att döma, kräva en precis timing för att kunna läsa av signaler rätt, vilket dock visade sig ej ha några praktiska implikationer på grund av processorns låga klockfrekvens.

Med tiden skulle fler oklarheter eller rentav felaktigheter med vårt kopplingsschema framgå. Genom diverse itereringar uppnådde vi så småningom en slutgiltig kopplingsdesign. Se 3. Snabbreferenser i Appendix för översikt över vilka kopplingsval vi gjorde och vilka portar komponenterna alltså relaterar till.

Montering

Monteringen skedde på ett mönsterkort där sladdar löddes fast eller virades in i komponenter med tänder. Processorn fästes alltså i mitten och resten av komponenterna placerades ut för att skapa en tydlig framsida till arkadspelet. Genom att använda oss av en stor resistans med flera ben besparades en hel del lödning.

Monteringen skedde efterhand med resten av arbetet och vissa kopplingar fick återkommas till eftersom allting inte löddes eller virades fast på en gång.

Bildskärm

Det som drog ut på tiden var att få igång bildskärmen. Utöver ström behöver bildskärmen flera statiska signaler för att vara igång samt en signal för att bestämma vilket chip som skall styras i det ögonblicket. Styrningen av bildskärmen sköts av tio ytterligare signaler - RS, RW och åtta datasignaler, motsvarande en byte. Med hjälp av detta kan signaler för att välja pixeloktetter via X och Y-adress skickas, samt ändra dessa med hjälp av Read och Write-signaler på en byte. Slutligen behöver också bildskärmens status behöva kunna läsas, vilken resulterar i en LCD_Ready-metod i programkoden. När bildskärmen är i Status Read signalerar pin DB7 om bildskärmen är upptagen eller inte. Den är alltså väsentlig för att skärmen skall gå att skriva ordentligt till. Detta steg hänger alltså definitivt ihop med nästa.

Kodning

Principen bakom kodningen är att varje port på processorn är en variabel på en byte (8 bits). Via bitlogik kan man alltså ändra dessa värden och därmed signalen som skickas ut ur en specifik pin. Dessutom har varje port två ytterligare variabler, på en byte vardera, varav bestämmer huruvida den ena bestämmer om en pin ska skicka eller ta emot en signal, samt en tredje som alltid visar vad detta värde in är.

Ett problem uppstod på grund av dålig koll på datorns mjukvara. Felaktiga drivrutiner gjorde att datorn inte hittade JTAG-enheten. Det berodde tydligen på vilken version av Atmel Studio som var senast installerad, eftersom drivrutinerna inte kunde stödja båda versionerna samtidigt.

Vid det här laget fanns en stor del kod att felsöka, och det var svårt att veta om det var fel på hårdvaran, kodningslogiken eller ren syntax/bitlogik. Det visade sig nämligen att det var fel på alla tre. Efter mycket om och men gick det dock tillslut.

Design

Kvar fanns då bara det roliga, det vill säga bitdesign för startskärm, meny och spelbräde. Pixeldesign för text gjordes via Excel-blad som sedan översattes till koordinater. Pixeldesign av startmenyn gjordes via en applikation som läser in .jpeg

och med hjälp av ett tröskelvärde för kontrast kan översätta den till pixlar också som koordinater. Koordinaterna kunde sedan matas in

Reflektion

Resultat

Projektet gick ganska bra ihop med planen. En del missdömningar i vad som skulle ta längs tid gjordes. Kravspecifikationen uppfylldes utom att ingen officiell Mayhem-funktion introducerades. Dock hann vi med att bygga en riktigt snygg monteringslåda med fin design utanpå vilket definitivt skulle hjälpa att sälja en sådan här arkadmaskin om det hade varit vårt mål.

Diskussion

En insikt vi snart kom till var hur potentiometrar skall kopplas. Vi missförstod att alla tre pins skulle behöva användas, eftersom vi märkte att strömmen enbart flödade genom två pins. Detta medförde att spänningen därigenom var konstant 5V och regleringen fungerade inte som den skulle förrän den tredje mittpinen kopplades till 0V. Liknande fel gjordes för spänningsreglering för knapparna. Knapparna var ej heller först inkopplade så att spänningen gick ner till noll vid nedtryckning.

Vi hade kunnat förbättra vår arbetsprocess genom att göra en bättre Debug av hårdvaran direkt efter montering. Eftersom detta steg inte genomfördes ordentligt gick arbetet med att få igång skärmen onödigt långsamt. Genom att vara mer noggrann i både förväg och efterhand kan alltså tid och energi sparas vilket är en lärdom att ta med till nästa gång.

Appendix

1. Kopplingsschema
2. Programkod
3. Snabbreferenser

Display to Chip, LCD Logic

CS1 - PD0
CS2 - PD1
RST - PD4
R/W - PD5
D/I - PD6
E - PD7
NC1 - R4
NC2 - GND (+0 v)

Display to Chip, Power

VDD - Power (+5 v)
VSS - GND (+0 v)
V0 - R3 (Mitt på)

VEE - R3
R3 - Power (+5 v)

Display to Chip, Logic

DB0 - PB0
DB1 - PB1
DB2 - PB2
DB3 - PB3
DB4 - PB4
DB5 - PB5
DB6 - PB6
DB7 - PB7

LEDs

LED_P1_v1 - PA2
LED_P1_v2 - PA3
LED_P1_v3 - PA4
LED_P1_R - PA5
LED_P2_R - PA6
LED_P2_v3 - PA7
LED_P2_v2 - PC7
LED_P2_v1 - PC6

Joysticks till Chip

Joystick_P1 - PA0
Joystick_P2 - PA1

Knappar till Chip

Button_P1 - PD2
Button_P2 - PD3

Port A

Player 1
LED RED - PC6, pin 28
LED 1 - PA2, pin 38
LED 2 - PA3, pin 37
LED 3 - PA4, pin 36
Joystick 1 - PA0, pin 40

Player 2
LED RED - PC7, pin 29
LED 1 - PA5, pin 35
LED 2 - PA6, pin 34
LED 3 - PA7, pin 33
Joystick 2 - PA1, pin 39