Komponenter

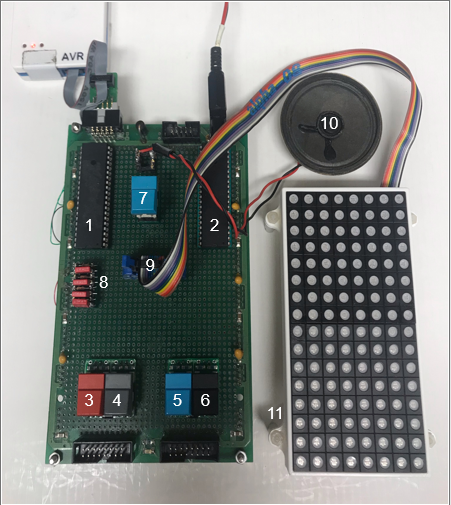
De komponenter som har använts under detta projekt i numrerad ordning är:

Bild . Överblick över konstruktionen.

1. AtMega16a (spellogik)
2. AtMega16a (ljud)
3. Avstudsad knappmodul (Rotation)
4. Avstudsad knappmodul (Rotation)
5. Avstudsad knappmodul (Vänsterflytt)
6. Avstudsad knappmodul (Högerflytt)
7. Reset-knapp
8. Motstånd
9. HC595
10. Högtalare
11. LED 8x16 matris

Arbetsfördelningen mellan våra två processorer har under detta projekt blivit något ojämn. Där den ena processorn förfogar över majoriteten av koden gällande spellogik och hårdvara medan den andra processorn istället besitter kod över ljudet och har i uppgift att styra högtalaren.

Den första avstudsade knappmodulen har i uppgift att sköta rotationen av våra spelfigurer. Den första knappen är tillägnad moturs-rotation och den andra medurs-rotation.

Den andra knappmodulen styr istället den horisontella förflyttningen kring spelfigurerna i spelet, den första för vänster- respektive andra för högerförflyttning. Den vertikala förflyttningen av figurerna har vi valt att inte påverka utan sker istället naturligt, likt gravitation, kodmässigt.

Vi har en extern reset-knapp som är kopplad mot ett motstånd för att fungera korrekt och har varit av stor betydelse under projektets gång. Den har främst kommit till användning gällande felsökning av buggar, som har skett mjukvarumässigt, för att snabbt starta en ny spel-session i hopp om att försöka återskapa och lokalisera hur ett fel har uppstått.

Vår HC595 är en seriell styrning som möjliggör kommunikation mellan vår ena processor och våra två seriekopplade skärmar. Skärmarna husas av en 3D-printad plastbehållare som håller ihop dem och tillsammans skapar en fullgod spelplan för vårt Tetris.

Högtalaren är tyst till dess att en hög flank från den andra processorn sker. Processorn styr också hur högtalaren ska låta.

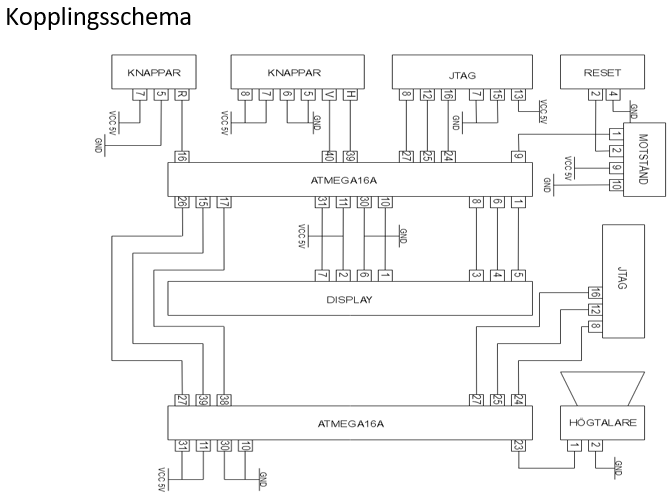
Figuren ovan är vårt kopplingsschema över våra komponenter. Där kan vi tydligt se att den övre AtMega16a processorn är den som är direkt kopplad till majoriteten av komponenterna som vi har valt att använda i detta projekt. Den har alltså ansvaret för att kommunicera med båda våra avstudsade knappmoduler, för såväl rotationen som den horisontella förflyttningen, men också till vår display och till sist vår externa reset-knapp, som i sin tur är kopplad mot ett motstånd.

Bild . Kommunikationen mellan komponenterna.

Den undre AtMega16a processorn är den, som tidigare nämnt, sköter ljudet i vårt projekt och är därför direkt uppkopplad för kommunikation mot högtalaren.

Processorerna kommunicerar sinsemellan genom tre vardera direkt kopplade pinnar. Detta gör att det är enkelt att identifiera när under spelets gång som vi vill spela ett visst ljud. Genom att skicka en hög flank till en specifik pin, vilket i sin tur kommer att ge en hög flank hos den undre processorn, som är inväntandes på insignal från den övre. Genom detta tillvägagångssätt så vet vi också just vilken pinne som fått en hög flank och också vilken melodi vi önskar att spela upp genom högtalaren.

Värt att nämna är att det befinner sig två stycken JTAGs på kopplingsschemat. Detta är till trots att vi inte hade tillgång till två, för tydlighetens skull, då vi endast kunde programmera en processor i taget. Detta görs genom att använda Daisy Chain i programmet Atmel Studio som vi har använt till vårt projekt vilket också är där vi har skrivit vår kod. För att komma åt den första processorn anger man att det inte är några enheter före, men att det är en enhet efter, men också att det inte är några bitar före, men fyra bitar efter. För att nå den andra processorn är processen omvänd.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Display

Bild 3. Skärmfunktion över en 8x8 RGB LED-matris.

Spelplanen som vi har valt att använda oss av är egentligen uppbyggd av två stycken seriekopplade 8x8 RGB ledmatriser, dessa två tillsammans bildar en total spelyta om 8x16. Detta är en önskvärd storlek för vårt projekt då spelfigurerna som mest är tre tända dioder i höjdled, detta gör att man kan stapla flera figurer på varandra dess för innan att spelplanen tar slut.

Nollorna i figuren ovan är tända dioder, vilket kommer att representera våra olika figurer, medan ettorna representerar släckta dioder som indikerar på ledigt spelutrymme.

Då skärmen är av RGB har den möjlighet till att representera olika färger beroende på vilka värden som anges för respektive bit. RGB är en förkortning för färgerna rött, grönt och blått och används för att möjliggöra ett spektrum av olika färgkombinationer.

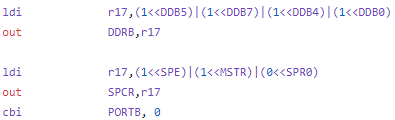
För att matriserna ska fungera korrekt används två olika speciella drivkretsar. Den första benämns source och tillgodoser ström radvis medan den andra benämns sink och jordar kolumnvis. Dessa drivkretsar är en del i konstruktionen av kretsarna 2981 respektive 2803. 2981 är en source-krets som innehåller 8 source kretsar och 2803 är en sink-krets som består av 8 sink kretsar. I konstruktionen används en 2981 och två 2803, dessa är i sin tur seriekopplade genom varsina HC595 skiftregister och bildar tillsammans ett 24-bits skiftregister som möjliggör seriell styrning. Ordningen som bitarna skickas ut till HC595-kretsen är anod, röd, grön och blå och sker genom användning av SPI.

SPI

Bild 4. Illustrering av SPI.

För att kunna påbörja kommunikationen mellan vår första processor och vår display använder vi oss utav Serial Peripheral Interface (SPI) som är ett synkront seriellt skiftregister.

SPI bussen använder sig av fyra olika logiska signaler. Serial Clock (SCLK), Master Output Slave Input (MOSI), Master Input Slave Output (MISO) och Slave Select (S̅S̅).



Kod . Kodexempel över initieringen av SPI.

Initieringen av SPI startar genom att tilldela en master, vilket är vår AtMega16a. När detta sker så sätts både klockhastigheten och en hög flank skickas till S̅S̅. Först när den höga flanken i S̅S̅ sker så är bussen mottaglig för data-överföring.

Bild 5. Skiftning av bitar.

Data-överföringen består i vårt fall av totalt 8 bitar, då vi använder oss av två skärmar. Bitarna skiftas ut en i taget undertiden S̅S̅ ligger i viloläge och mellan varje skiftning sker en mindre fördröjning. När väl alla 8 bitar har skiftats ut på SPDR så skickas det en låg flank till S̅S̅ och då vet vi att överföringen är klar.

Mottagandet av data-överföringen sker hos vår slave, vilket är displayen. Detta sker byte-vis i samband med ett avbrott då SPIF flaggan är satt i SPSR.