



WATCH-OUT!



BY MAKE MAGIC STUDIOS

Watch-out!



Mikrodatorprojekt – TSIU51 – Survival-spel – 2022-02-05

Grupp 5

Emil Pihl

Kebba Jeng

Aidin Jamshidi

Martin Castro Bildhjerd

Innehållsförteckning

0.0	Spelomslag	1
0.1	Projektinformation	2
1.0	Inledning	5
1.1	Bakgrund kring projektet	5
1.2	Grundlig beskrivning av spelet	5
1.3	Kravspecifikation	6
1.4	Blockschema	7
2.0	Översikt	8
2.1	Övergripande beskrivning av spelet	9
2.2	Komponenter	9
2.3	Organisering av projektarbete	9
2.4	Beskrivning av hårdvara & protokoll	10
2.4.1	DAvid-kort	10
2.4.2	Arduino Uno	11
2.4.3	DAmatrix	12
2.4.4	Piezoelektrisk högtalare	13
2.4.5	Spelknappar	14
2.4.6	LCD-display	15
2.4.7	TWI	15
2.4.8	SPI	16
3.0	Framgångar	17
3.1	Motgångar	18
3.2	Saker som skulle ha gjorts annorlunda	18
3.3	Diskussion	19
4.0	Kod	20
5.0	Referenser	99

Figurförteckning

Figur 0.0 – Watch-out!	0
Figur 1.2 – Spelexempel	6
Figur 1.3 – Förslag	7
Figur 1.4.0 – JSP	8
Figur 1.4.1 – Blockschema	8
Figur 2.3 – Planner	10
Figur 2.4.1 – DAvid-kort	10
Figur 2.4.2 – Arduino Uno	11
Figur 2.4.2 – I/O-portar	12
Figur 2.4.30 – Pixel position	12
Figur 2.4.31 – DAmatrix	13
Figur 2.4.4 – Högtalare	14
Figur 2.4.5 – Tryckknappar	14
Figur 2.4.6 – New game	15
Figur 2.4.7 – TWI	16
Figur 2.4.8 – SPI	17
Figur 3.1 – VMEM	18



1.0 - Inledning

Detta spel skapades som ett projekt för kursen mikrodatorprojekt TSIU51 som undervisas på Linköpings universitet för högskoleingenjörer inom elektronik. Gruppen som skapade detta projekt består av medlemmarna Emil Pihl, Kebba Jeng, Aidin Jamshidi och Martin Castro Bildhjerdt, som utgör grupp fem.

Projektet är inspirerat av spelet Dino-game som kan spelas på Googles hemsida när internetåtkomst saknas. Spelarens uppgift är att undvika objekt för att överleva och samla ihop poäng. Till spelet har olika rutiner skrivits åt hårdvaran för att skapa ett beroende till spelet genom saker som ljud, blinkande färger eller information på skärmarna.

1.1 – Bakgrund kring projektet

När projektet påbörjades spenderades mycket tid på att fundera ut hur spelet skulle fungera. Det som bestämdes kvickt var att spelet skulle både vara roligt och ha en viss svårighetsgrad. Genom inspiration från de symboliska och legendariska 90-tals spelen kunde gruppen konstatera att ett spel med perfekt användarupplevelse inte behöver omfatta invecklade funktioner eller avancerad grafik.

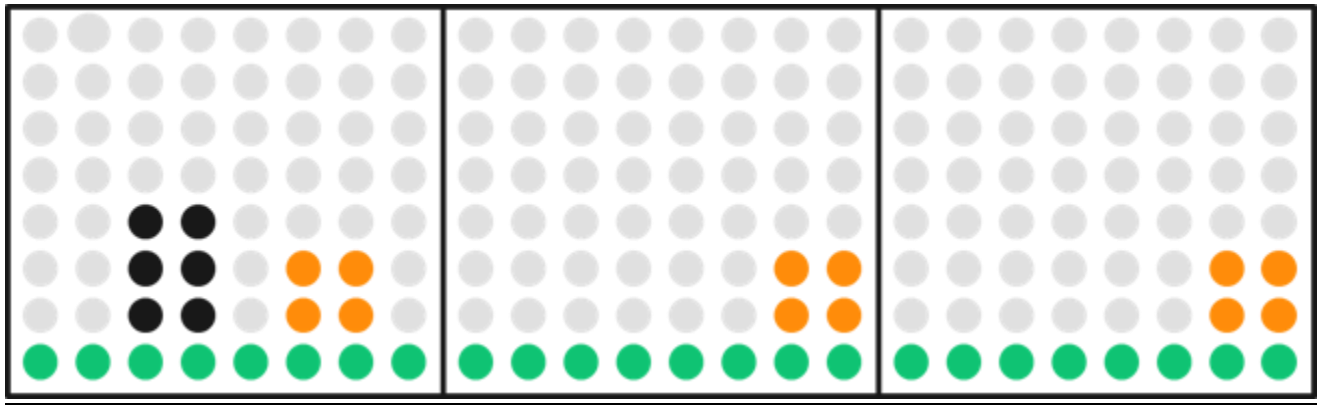
Gruppen bestämde sig under ett tidigt skede att spelet skulle innehålla en karaktär som styrs av spelaren, där spelarens uppgift är att undvika inkommande objekt. Det diskuterades mycket kring ifall karaktären ska röra sig mot sidan eller ifall objekten skulle röra sig mot spelaren men i slutändan beslutades att karaktären skulle stå stilla och objekten rör sig. Där spelaren ges möjligheten att hoppa eller ducka för inkommande objekt.

1.2 - Grundlig beskrivning av spelet

Spelet är planerat att vara väldigt simpelt, där spelaren ska undvika flygande objekt och stenar för att försöka att samla på poäng. Karaktären i spelet visar sig på vänster sida av spelplan och från höger sida av spelplan dyker objekt upp som måste undvikas för att överleva och fortsätta spelet. Karaktären står stilla och objekt rör sig emot den för att ge illusionen av att karaktären rör sig.



Stundvis behöver spelaren hoppa eller ducka för att undvika objekt för att inte förlora liv, för att spelomgången ska avslutas.



Figur 1.2 - Spelexempel: På figuren syns ett exempel på hur ett objekt närmar sig karaktären som styrs av spelaren.

1.3 - Kravspecifikation

I starten av projektet skall en kravspecifikation skrivas som innehåller viss information angående slutprodukten. Först bestämdes hur spelet ska utspela sig och därefter vilka komponenter som kan användas till detta. När en grund hade blivit lagd så kunde en generell idé utformas. Därefter gjordes en lista med "Skall-krav:" och "Bör-krav:". **Skall krav**, innefattar vad spelet ska innehålla medan börkraven innefattar krav om hur spelet bör utformas för en extraordinär användarupplevelse.

Efter mycket tankar och idéer kring hur spelet skulle fungera så skrevs allting upp på en tavla, vilket sedan gick över till en omröstning. Under röstning togs varje idé upp och diskuterades ifall det skulle fungera och efter en tid kunde uppbyggnaden av spelet fastställas. Resultatet kan ses nedan.

Skall-krav:



- ❖ Spelaren **skall** vara en figur som liknar en dinosaurie.
- ❖ Spelaren **skall** kunna hoppa med figuren.
- ❖ Spelaren **skall** röra sig åt höger automatiskt med figuren.
- ❖ Spelaren **skall** kunna samla poäng.
- ❖ Spelaren **skall** kunna förlora spelet genom att förlora alla hjärtan.
- ❖ Spelaren **skall** kunna spela med knappar på DAVID-kortet.
- ❖ Det **skall** låta när spelaren startar spelet.
- ❖ Det **skall** låta när spelaren förlorar.
- ❖ Det **skall** finnas en huvudmeny.
- ❖ Det **skall** finnas en poäng "bild" när spelaren förlorar.
- ❖ Det **skall** finnas två DAMatrix.

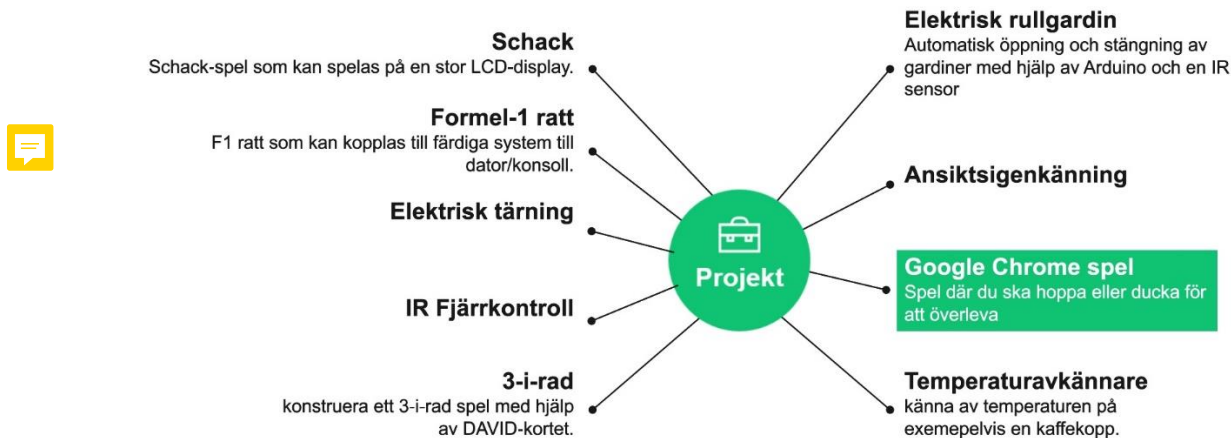


Bör-krav:

- ❖ Spelaren **bör** kunna duka med figuren.
- ❖ Spelaren **bör** kunna samla poäng genom olika metoder.
- ❖ Spelaren **bör** kunna samla på sig upp till tre hjärtan.
- ❖ Spelaren **bör** kunna spela med joystick.
- ❖ Det **bör** finnas tre DAMatrix.

- ❖ Det **bör** finnas flygande föremål som spelare måste undvika.
- ❖ Det **bör** spelas upp ett ljud när man samlar på sig poäng
- ❖ Det **bör** spelas upp ett ljud när man samlar på sig ett hjärta.
- ❖ Det **bör** spelas upp ett ljud när förlorar ett hjärta.
- ❖ Det **bör** gå att starta om spelet från “förlora-bilden” utan att starta om kortet.
- ❖ Spelets hastighet **bör** öka för en ökad svårighetsgrad.

Utredning av förslag

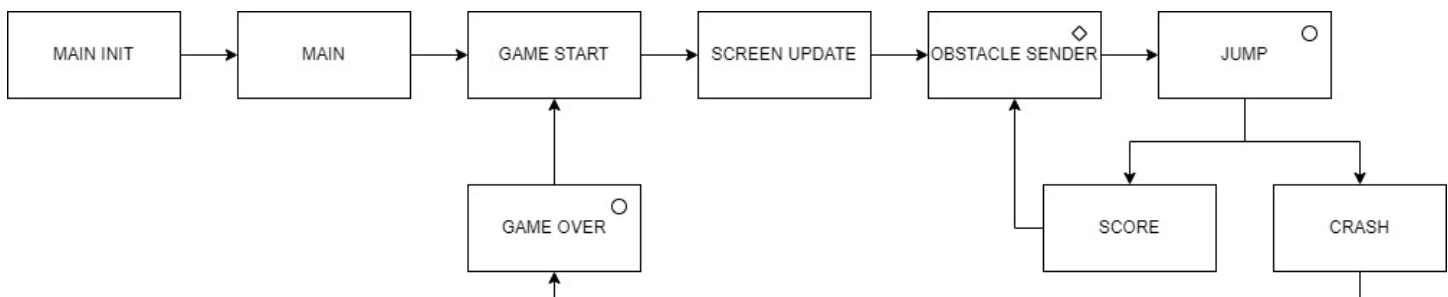


Figur 1.3 - Förslag: Exempel på projektidéer och det slutgiltiga markerat med grön färg.

1.4 - Blockschema

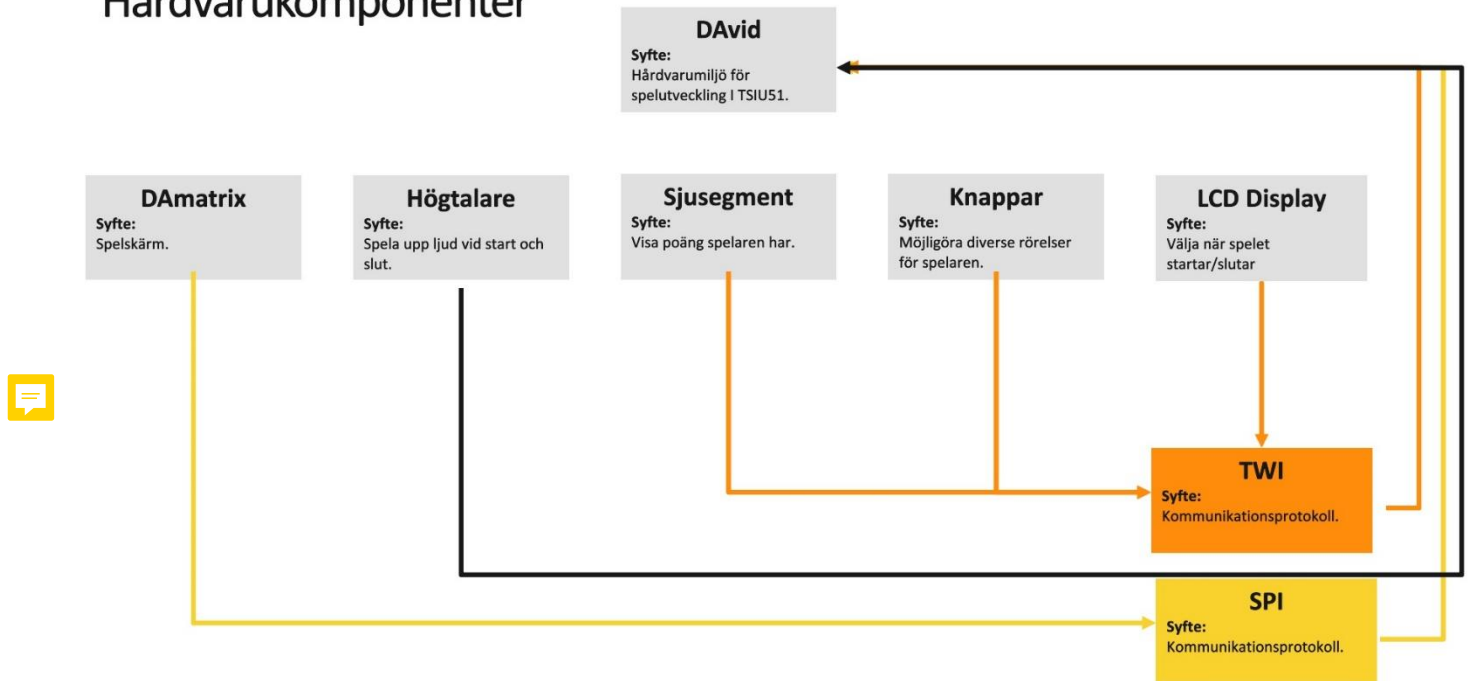
Blockschemat nedan visar hårdvaran som används i projektet och även tillhörande kommunikationsprotokoll till dessa hårdvaror. Kommunikationsprotokollet möjliggjorde kommunikation mellan olika komponenter på hårdvaruplattformen David [X].

Illustrationen **nedan** visar de komponenter som användes för spelet, samt hur komponenterna kommunicerar med David. Anledningen till detta var för att inte skapa kod som är svårläslig, **detta för att** få ett strukturerat sätt att läsa koden för att förenkla förståendet kring hur allt binds samman.



Figur 1.4.0 - JSP: Strukturdiagram för hur rutinerna fungerar i spelet.

Hårdvarukomponenter



Figur 1.4.1 - Blockschema: Hårdvarukortet DAvid. Framtaget för projektkurser i Mikrodator teknik. Hårdvarumiljön består utav diverse komponenter som LCD-display, LEDs, tryckknappar, med mera.

2.0 – Översikt

Gruppen valde att utveckla ett spel som inte var invecklat sett till funktionalitet, och ger spelaren en god användarupplevelse, valde gruppen att ta en mellanväg som tänktes inte vara för överkomplicerad men som ändå ger en god användarupplevelse, likt dem klassiska 90-tals spelen.

Efter en hel del diskussion så bestämdes vilka delar av DAvid-kortet som skulle tänkas vara användbara för projektet. Skärmen bestämdes till tre DAmatrix-skärmar horisontellt placerade för att ge en överblick över spelplanen, för att kunna reagera på inkommande objekt. Detta för att spelaren ska få en rolig spelupplevelse och vilja spela det igen.

Högtalaren används till att spela olika typer av små ljud för att indikera olika stadier i spelet. När spelet startas spelas ett specifikt ljud som ger spelaren feedback om att spelet har startat, detta sker även när spelet återställs genom DAvid-kortet. När spelaren förlorar ska ett annat specifikt ljud spelas upp för att indikera att spelaren har förlorat sin omgång.

För att undvika objekten så måste spelaren använda sig av knapparna på hårdvarukortet. Det ska vara simpelt för spelaren att förstå kontrollerna då det endast blir två knappar för hoppa och ducka. För att hjälpa spelaren förstå vad som sker i spelet används en LCD-display för att förmedla information till spelaren för att förklara ifall det är ett nytt spel eller avslutat spel. Anledning till detta då det går snabbt och lätt att informera spelaren om vad som sker.

2.1 - Övergripande beskrivning av spelet

När spelaren startar eller påbörjat ett nytt spel ska en melodi spelas för att visa att spelet snart startar. På LCD-displayen visas information om vilka knappar som ska användas av spelaren för att starta spelet och för att verkställa rörelse av spelkaraktären. När spelaren förlorar ska en annan typ av melodi spelas och förlusttext på LCD-displayen ska framföras till användaren. Poängsamlingen sammanställs och spelaren noteras om hur många poäng som samlades under spelets gång.

Då ett nytt spel påbörjas **så** är spelkaraktären stillastående på startpositionen och en tom spelplan uppenbarar sig. Efter ett tag börjar objekt laddas in på DA- matrix-skärmen längst till höger som sakta börjar röra sig **emot** karaktären som spelaren styr. Är det ett flygande objekt måste spelaren ducka för att få högre **poäng, ifall** det är ett markhinder **mås** spelaren i stället hoppa för att undvika objektet och samla poäng.


Desto längre spelaren klarar sig utan kollision desto mer poäng samlas ihop, **svårighetsgraden** på spelet ökar med tidens gång för att **skapa lite mer beroende till** spelet.

2.2 - Komponenter

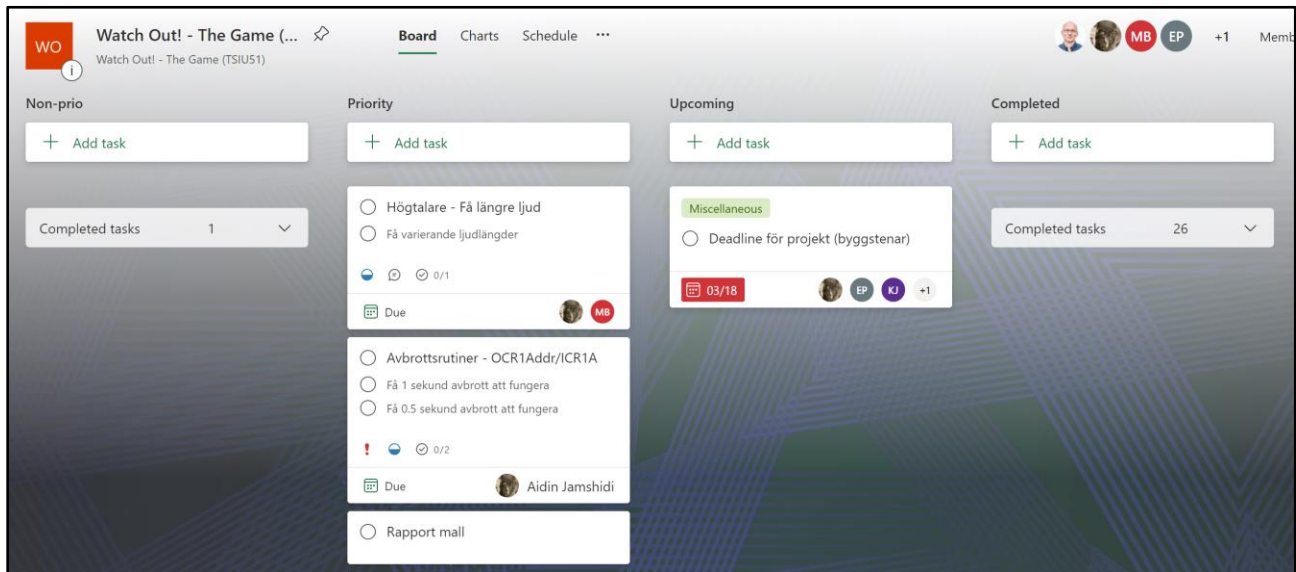


1x DAvid-kort (**Mikrokontroller**)
1x Arduino Uno (**Processor**)
3x DAMatrix 8x8 (**Skärm 8x24**)
1x Piezoelektrisk högtalare (**Högtalare**)
2x Spelknappar (**Joystick**)
1x LCD-display (**Skärm**)
1x 3D-printad stativ för DAMatrix (**Stativ**)

2.3 - Organisering av projektarbete

 Organiseringen av arbetet gick till genom att ha två möten i veckan. Ett som var på söndagen genom Discord och ett annat möte som var på plats varje tisdag för att se hur projektarbetet låg till. Under varje möte diskuterades det hur varje medlem låg till med sitt arbete, eller ifall personen var klar. Ifall personen var klar med sitt arbete **så** påbörjades nästa arbete och del av projektet.

Något som användes under projektets gång var Microsoft **planer** där arbetet kunde planeras och sedan kunde andra arbetsuppgifter tillsättas till en medlem, **därav visste personen vad som skulle göras näst.**

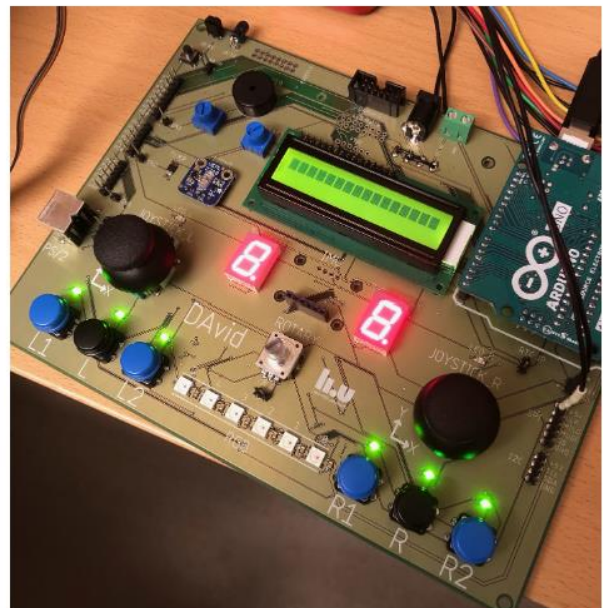
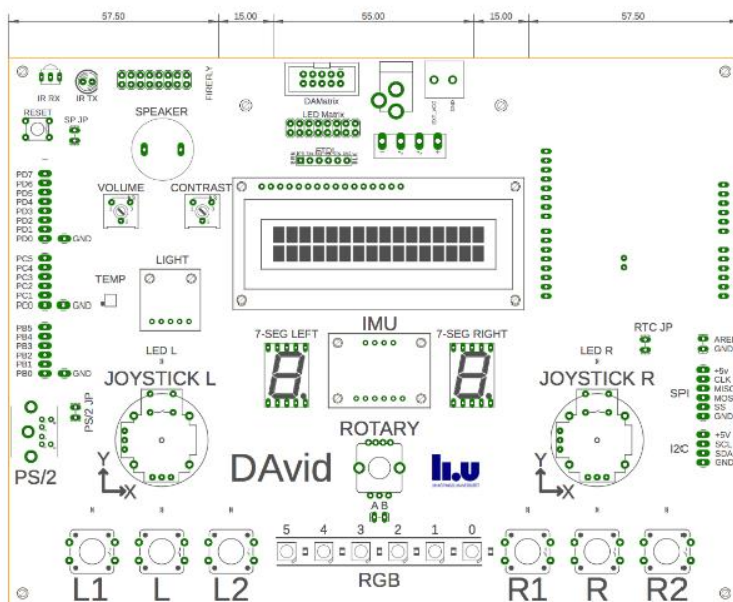


Figur 2.3 – Planner: Olika uppgifter i Microsoft planner där de rankas under olika kategorier med specifika medlemmar tilldelade.

2.4 - Beskrivning av hårdvara & protokoll

2.4.1 - DAvid-kort

DAvid-kortet är ett hårdvarukort utvecklat för kursen Mikrodatorprojekt (TSIU51) och detta kort används under projektets gång. Det är en hårdvarumiljö med diverse komponenter för att möjliggöra strukturerad mjukvaruutveckling, vars hårdvarumiljö lämpar sig för exempelvis spel. Kortet styrs av en Arduino Uno med processorn ATmega328p.



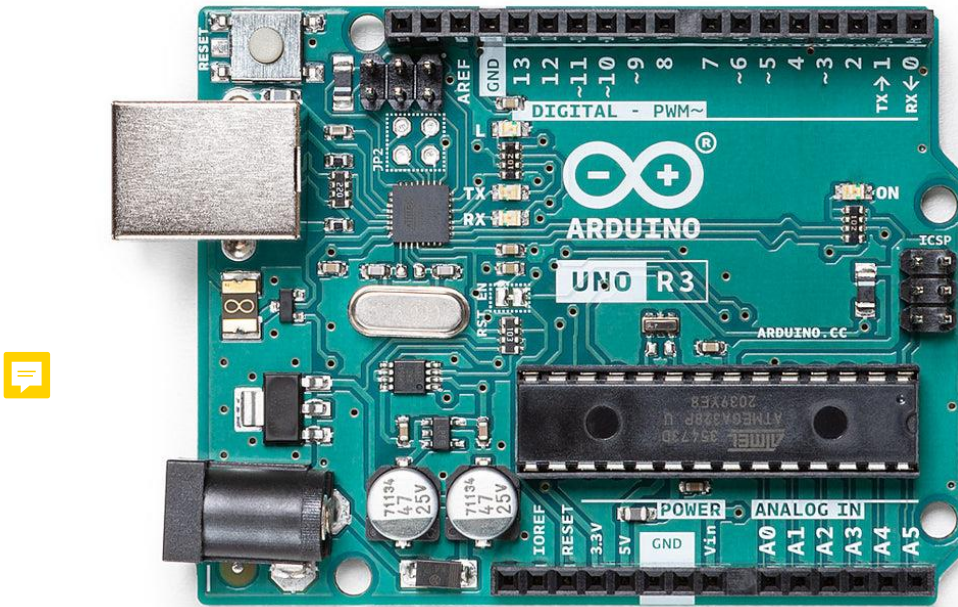
Figur 2.4.1 – DAvid-kort: Hårdvarukortet DAvid. Framtaget för projekt i kursen Mikrodatorprojekt. Hårdvarumiljön består utav diverse komponenter som LCD-display, LEDs, tryckknappar, med mera.

2.4.2 - Arduino Uno



Arduino Uno är ett mikrokontrollerkort[X] baserat på processorn ATmega328p[X]. En mikrokontroll är en mindre dator med CPU, arbetsminne, och programminne integrerat komplett med stödfunktioner på en enda interagerad elektronisk krets.

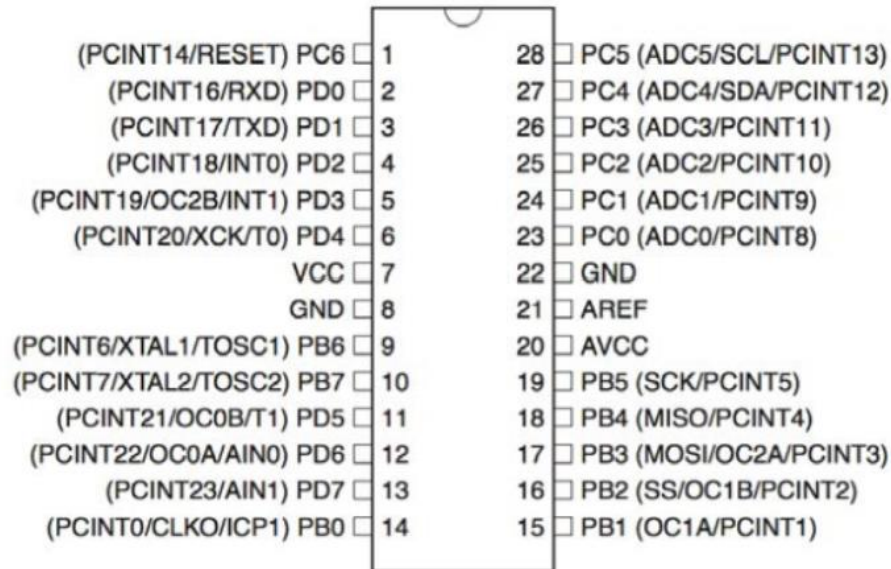
Mikrokontrollerkortet är utrustat med digitala och analoga I/O[X] pinnar som kan användas för att kommunicera med andra komponenter och kretsar.



Figur 2.4.2 – Arduino Uno: Överblick av mikrokontrollerkortet Arduino Uno.



ATmega328p är en mikrokontroll från Atmel. ATmega328p har ett 32 kbyte minne och en 16 MHz klockfrekvens, detta gör att mikrokontrollerna kan utföra upp till 16 miljoner instruktioner per sekund [x]. Det finns ett flertal I/O portar på ATmega328p, detta ger möjlighet att ansluta extern hårdvara som inte finns i DAvid-kortets hårdvarumiljö, exempelvis DAMatrix [x]



Figur 2.4.2 – I/O-portar: Överblick på tillgängliga I/O portar på Arduino Uno.

2.4.3 - DAmatrix

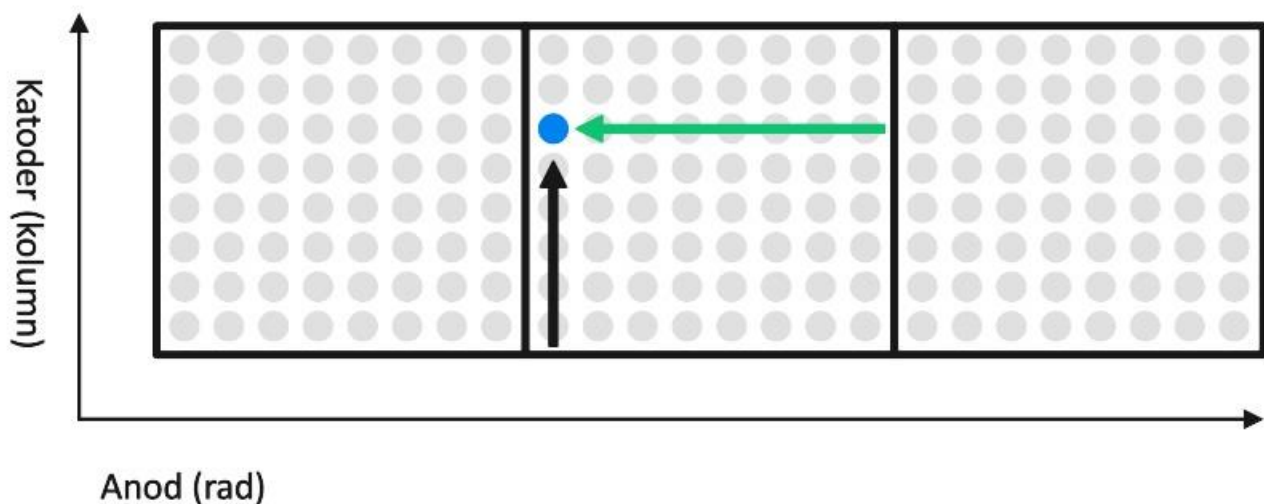
DAmatrix är ett block som består av 64 lysdioder (LED) i en matris (8x8). Genom att använda tre DAmatrix så kan en större (8x24) spelplan skapas.

I detta projekt används tre DAmatrix för att få en spelplan på 8x24.

DAmatrix drivs med det seriella gränssnittet SPI [X] via Arduino UNO:s I/O pinnar.

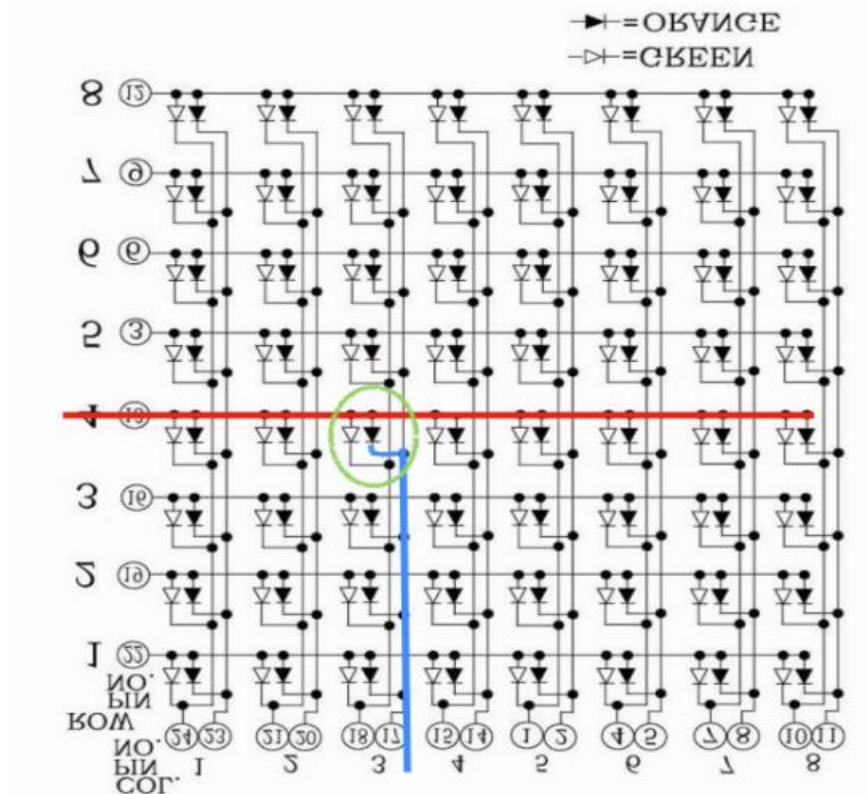
DAmatrix är byggt på ett sätt så att

- Dioder i varje rad i matrisen hänger ihop i sina anoder.
- Dioder av samma sort och kolumn har sina katoder sammankopplade.



Figur 2.4.30 – Pixel position: Exempelbild på hur dioder kan tändas separat på DAmatrix genom beskrivningen ovan.

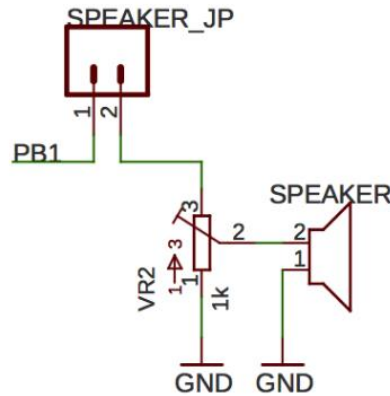
Med den infon tillhands så behöver alltså raden tillfördes med positiv spänning, samtidigt som diodens katod jordas [x] för att tända en diod.



Figur 2.4.31 – DAmatrix: Kopplingsschema för DAmatrix. Bilden är omvänd för att förenkla tydning av dioder.

2.4.4 - Piezoelektrisk högtalare

På hårdvarukortet DAvid finns det en piezoelektrisk högtalare som kan användas. Högst effektivitet uppnås vid frekvenser runt 3000–4000 Hz. Högtalaren är passiv, därav behövs matningsspänning vilket gör att högtalaren bara behöver en låg och en hög puls för att låta.



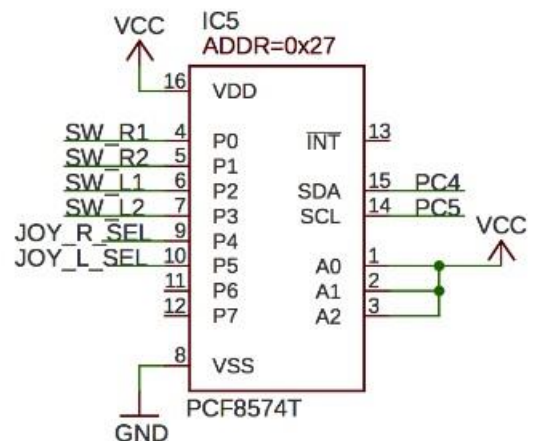
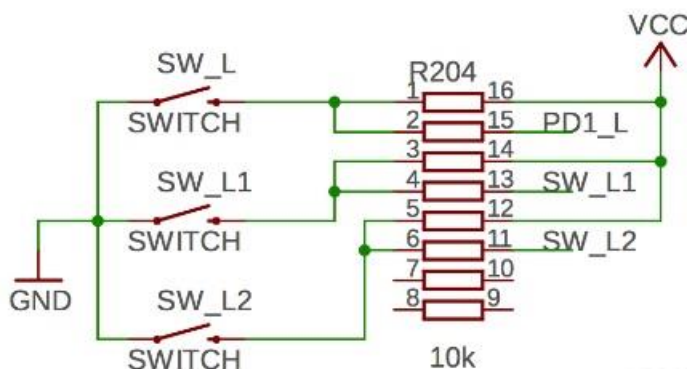
Figur 2.4.4 – Högtalare: Kopplingsschema för piezoelektriska högtalaren.

Genom att skapa en rutin som skickar en puls sådan att högtalaren varierar från låg till hög så kan ett ljud skapas. Genom rutiner som skickar frekvenser i varierande storlek kan simpla melodier eller ljud skapas. Via en potentiometer kan ljudstyrkan ändras.

2.4.5 – Spelknappar

I projektet används tryckknappar. Tryckknapparna används för att utlösa de animationer i spelet som resulterar i att spelaren undviker hinder. Gruppen har valt att använda tryckknappar över exempelvis joysticks då det ur ett programmeringsperspektiv är lättare att adressera.

Tryckknapparna är kopplad till en specifik port. Porten är i sitt standby läge satt som låg[x]. För att adressera knapparna behövs således en positiv insignal. Gruppen har med den informationen tillhanda skrivit en kod med instruktioner som kollar status på porten som knapparna är bundna till.



Figur 2.4.5 – Tryckknappar: Kopplingsschema för tryckknapparna. I detta fall används tryckknapp L och R.

2.4.6 - LCD-display



För projektet behövdes en metod för att förmedla spelaren ifall omgången var förlorad eller ifall en ny omgång höll på att starta. Detta kunde LCD-displayen användas till genom att förmedla information så spelaren vet vad som sker. Detta gjordes genom två rutiner som separat användes sig av **TWI protokollet** för att visa text på skärmen.



Figur 2.4.6 – New game: Skärmen som visas för spelaren innan spelet startar.

2.4.7 – TWI

Two-Wire Interface, förkortat **TWI** är ett synkront seriellt kommunikationsprotokoll. Ett vanligt användningsområde är i hårdvara där tillverkningskostnad prioriteras över prestanda.

TWI ger hårdvara, system, och processer ett **kommunikation gränssnitt** med en buss som består av två stycken ledningar, **Serial Data (SDA)**, och **Serial Clock (SCL)**.

SDA är dataledningen, där **överförs data** mellan en **Master** och **Slave**. **SCL** är klockledningen **TWI** **kommunicerar i duplex med en master-slave arkitektur**, det vill säga att kommunikation sker samtidigt i två riktningar **simultant**. **Master-slave arkitekturen** möjliggör en asymmetrisk kommunikation där en enhet eller process styr och kontrollerar andra enheter och processer.

För att överföra data och kommunicera mellan noder så behöver protokollet konfigureras. Detta görs genom att modifiera binära bitar i **TWI:s** olika register.

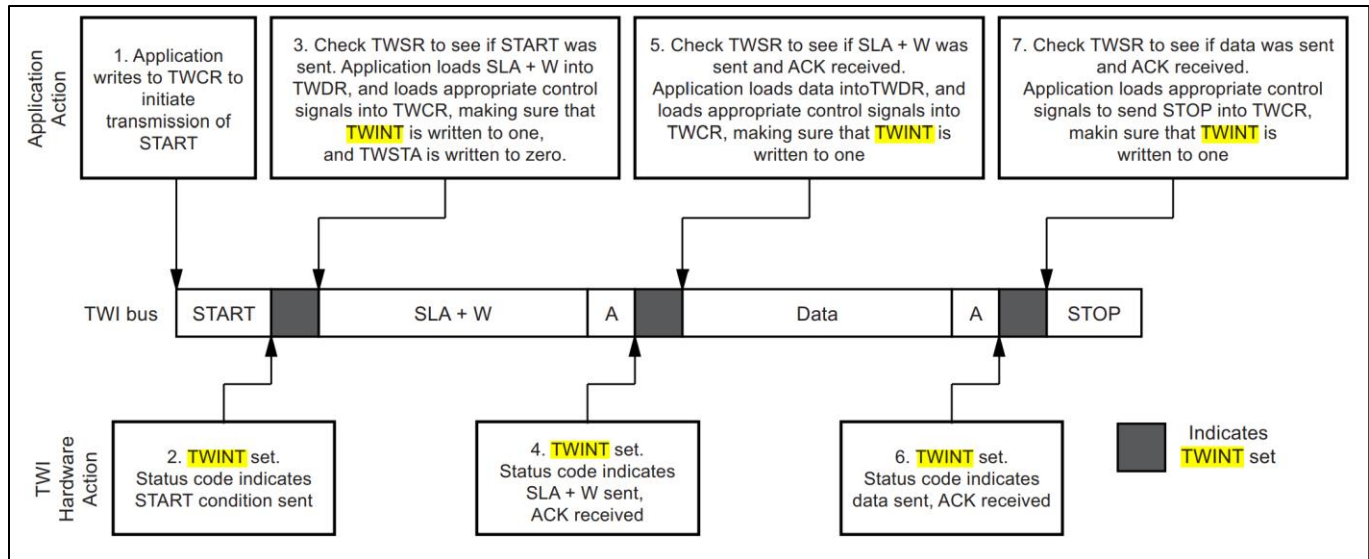
I detta projekt konfigurerades följande register:

- **TWI Bit Rate Register**, detta register kontrollerar klockledningens period.
- **TWI Control Register**, detta register kontrollerar **TWI operationer**, exempelvis avbrott. Registret används även för att generera **START**, **STOP**, och **ACK pulser**.
- **TWI Data Register**, registret kan sättas i olika lägen. I överförningsläge skickar den data. I mottagningsläge, sparas data som mottagits.

En TWI dataöverföring består av

- Ett startvillkor
- En adress och information om Read/Write samt Slave Acknowledge
- Data
- Ett stoppvillkor

Figuren nedan illustrerar TWI processen.



Figur 2.4.7 – TWI: Exempel på hur information måste skickas genom TWI-protokollet.

2.4.8 - SPI

Serial Peripheral Interface, förkortat SPI är ett synkront seriellt kommunikationsprotokoll som huvudsakligen används för kort-distanskommunikation i inbyggda system.

SPI kommunicerar i full-duplex med en master-slave arkitektur, det vill säga att kommunikation sker samtidigt i två eller fler riktningar. Master-slave arkitekturen möjliggör en asymmetrisk kommunikation där en enhet eller process styr och kontrollerar andra enheter och processer.

SPI förbinds av en buss som består av fyra gemensamma ledningar. Serial Clock Output (SCLK), Master Out Slave In (MOSI), Master in Slave Out (MISO) och Slave Select (SS).

SCLK är som namnet indikerar klockledningen. Data som skickas genom MOSI och MISO synkroniseras till klockan som huvudnoden genererar.

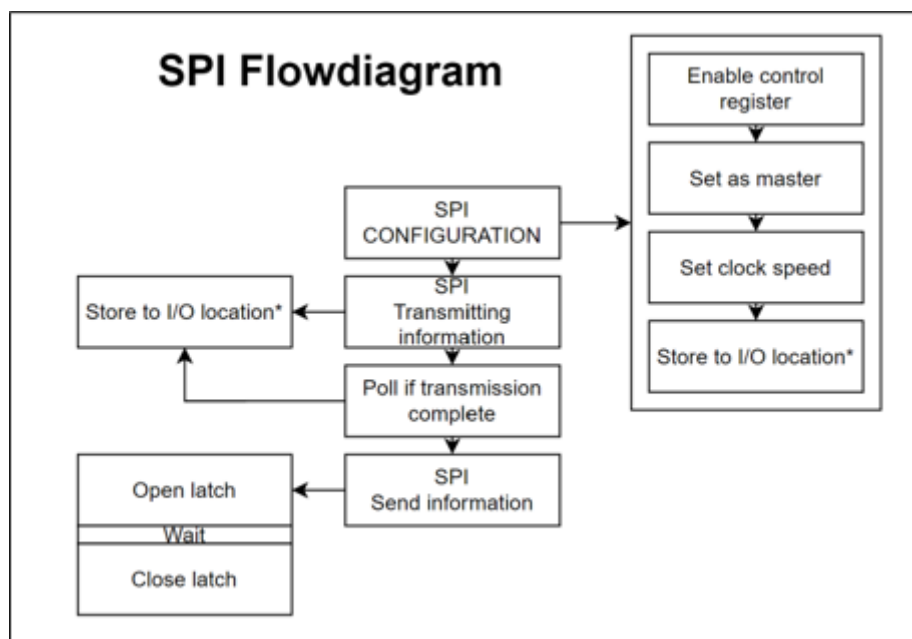
MOSI och MISO är dataledningarna. MOSI överför data från huvudnoden till sub-noden, MISO överför data från sub-noden till huvudnoden. Protokollet kan göra detta simultant.

SS berättar vad för enhet eller process som är sub-nod, när flera sub-noder används så används individuella slave select.

För att överföra data och kommunicera mellan noder så behöver protokollet konfigureras. Detta görs genom att modifiera binära bitar i SPI:s kontrollregister (SPCR), därefter så behöver riktningen också anges, det vill säga vart informationen ska skickas till.

I projektet används SPI för att kommunicera med spel planet som består av tre DAMatrix.
För att driva LED matriserna skickas färginformation, samt katod och anodinformation med SPI.

Flödesdiagrammet nedan illustrerar kodflödet för SPI.



Figur 2.4.8 – SPI: Flödesdiagram för SPI protokollet.

3.0 - Framgångar

Gruppen var duktiga med att planera och hålla sig till planering och sitta tillsammans för projektarbetet. Gruppen kunde tack vare denna anledning diskutera mycket med varandra kring hur rutiner och olika komponenter skulle fungera.

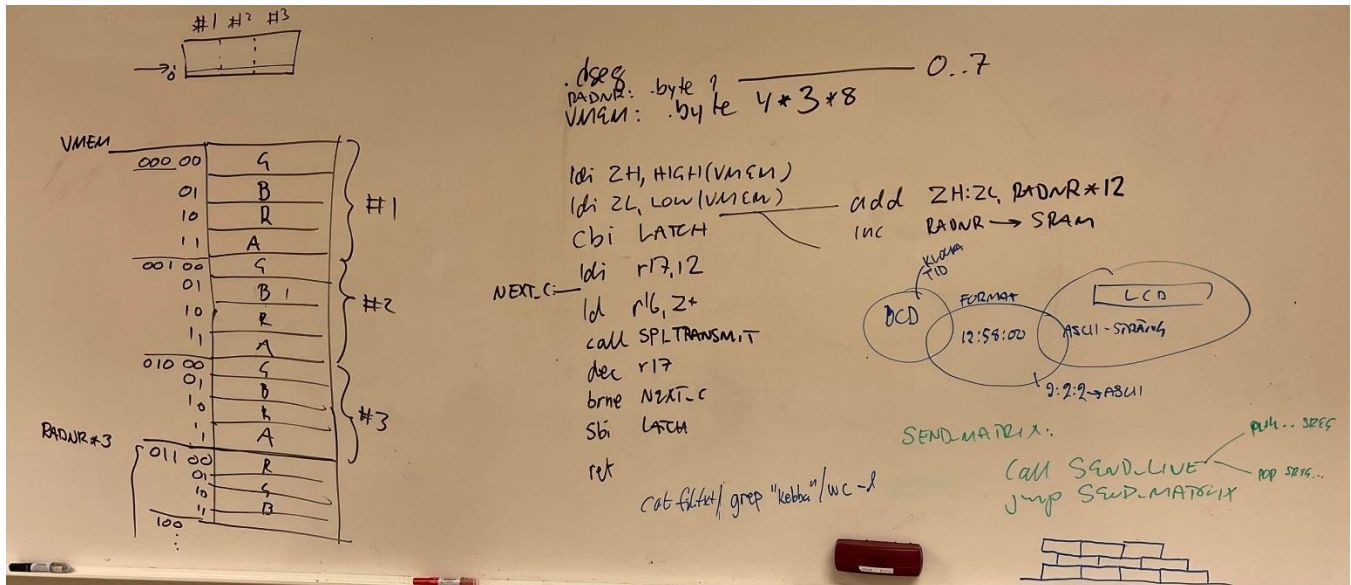
Två viktiga rutiner som behövdes för att få spelet att fungera med de komponenter som behövdes var TWI och SPI. Dessa två rutiner var viktiga för att få komponenter att svara och även för att kunna skicka information till skärmen. När gruppen satt och diskuterade hur dessa rutiner skulle fungera så gick det snabbt framåt och inte tog stor del av projektets tid.

När rutinerna TWI och SPI var klara kunde rutiner för komponenterna som skulle användas till spelet börja skrivas. Även denna del gick väldigt fort då det mesta flöt på när gruppen samarbetade. Gruppen brukade hålla sig till två eller fyra personer för att effektivt kunna arbeta.

3.1 – Motgångar

När kursen mikrodatorprojekt (TSEI51) startade så insåg inte gruppen hur stort projektet skulle bli i slutändan. Detta orsakade att tiden inte räckte till för att få spelet till hur det var tänkt från början, då idéerna var många. När väl slutet av projektet började närma sig insåg gruppen hur mycket arbete som fortfarande var kvar och därav fick sitta många timmar mot slutet.

De flesta rutiner och liknande gick fort att skriva och göra klart, dock insåg inte gruppen hur svårt det egentligen skulle bli med att skriva ett videominne för spelet, där information kring karaktär, golv och objekt ska sparas. Det skulle visa sig vara ett av de svåraste hindren för projektet där nästan all tid spenderades på att få det att fungera korrekt.



Figur 3.1 – VMEM: Förklaring av hur VMEM ska fungera från kursledare vilket sedan blev metoden gruppen använde sig av.

3.2 - Saker som skulle ha gjorts annorlunda

När projektet väl hade blivit bestämt och rutinerna för komponenter och protokoll skulle skrivas så delades gruppen upp, där varje medlem satt och jobbade med något separat. Detta var något som gruppen trodde skulle fungera bra men i stället skapade problem. När problem uppstod för en medlem så fanns det ingen att diskutera med. Detta orsakade att man blev kodblind för sina egna misstag.

När väl gruppen mot slutet av projektiden började sitta tillsammans för de sista rutinerna märktes det kvickt hur mycket mer effektivt gruppen jobbade. Detta då gruppen alltid började ifrågasätta varandra när problem uppstod, som att kod inte agerade så som planerat. När frågorna ställdes börjades problemen dyka upp och koden kunde skrivas om för att få resultaten som förväntades.

Något som gruppen gjorde mycket i början var att rita på tavla för att förstå hur saker skulle fungera. När gruppen hade delats upp så skedde detta inte på samma skala då varje medlem satt ensam med sin del. När gruppen började att problemlösa tillsammans igen så ritades det mycket mer och detta hjälpte med många problem. Koden som skulle skrivas kunde i princip ritas upp på tavla.

3.3 - Diskussion



Projektet har efter mycket omständigheter knutits ihop för att få spelet som var planerat i starten av projektet. Resultatet blev inte helt som gruppen hade planerat då tiden inte räckte till för att göra projektet i den skalan som var tänkt från början, därav saknas vissa funktioner som fanns med i kravspecifikationen. Även fast vissa delar saknas så kan spelet spelas genom att undvika objekt genom att hoppa över dem.

Denna kurs ska vara som en förberedelse inför arbetslivet där projektgrupper sker på en högre skala. Genom att lära gruppmedlemmarna hur ett projekt ska planeras, hur det ska utföras, rapportskrivning och även två muntliga presentationer.

Genom detta projekt har mycket planering, kommunikation och även mycket informationssökning skett. Genom att använda sig genom dessa medel så har gruppen kunnat tagit sig framåt. Något som även gruppen fick lära sig var att inte överskatta tiden ett projekt kan ta, även fast mycket tid spenderas på det. Det är på grund av denna anledning projektet inte uppnådde designen planerad från början.

Även fast projektet inte uppnådde designen som var planerad så har gruppmedlemmarna lärt sig massor under projektets gång. Några av de viktiga kunskaperna gruppen har lärt sig är bland annat att våga fråga mer efter hjälp, att inte överskatta jobbet för ett projekt, att kunna planera tillsammans på ett effektivt sätt och även förbättrat våra Assembly kunskaper en stor del.

4.0 - Kod

Kod ej klar ännu.

Referenser och information

5.0 – Referenser



How to Design a Mobile Game So Addictive It's Almost Irresponsible

<https://blog.proto.io/how-to-design-a-mobile-game-so-addictive-its-almost-irresponsible/>

What makes tetris so incredibly addictive

<https://tetris.com/article/44/what-makes-tetris-so-incredibly-addictive>

Atmel AVR 8-bit Instruction Set

<https://content.instructables.com/ORIG/FIH/E7GU/I301K81S/FIHE7GUI301K81S.pdf>

Datorteknik – Hyfsa kod - Michael Josefsson

https://teams.microsoft.com/_?tenantId=913f18ec-7f26-4c5f-a816-784fe9a58edd#/pdf/

Datablad_ATMega328.pdf - Michael Josefsson

https://teams.microsoft.com/_?tenantId=913f18ec-7f26-4c5f-a816-784fe9a58edd#/pdf/

David_hardvarubeskrivning.pdf - Michael Josefsson

https://teams.microsoft.com/_?tenantId=913f18ec-7f26-4c5f-a816-784fe9a58edd#/pdf/

ATMega328p.pdf

https://teams.microsoft.com/_?tenantId=913f18ec-7f26-4c5f-a816-784fe9a58edd#/pdf/

Drivning av LED-matrisen – DAmatrix - Michael Josefsson

https://teams.microsoft.com/_?tenantId=913f18ec-7f26-4c5f-a816-784fe9a58edd#/pdf/

Remote 8-bit I/O expander for I2C-bus with interrupt

https://teams.microsoft.com/_?tenantId=913f18ec-7f26-4c5f-a816-784fe9a58edd#/pdf/