Snakespel

EITA15: Digitala System

LUNDS UNIVERSITET

Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg

Datateknik: IDA1

Drilon Grajqevci

Oliver Ekberg

Love Karltorp

Arvid Gruvö Wärle

Handledare: Bertil Lindvall

Sammanfattning

Denna rapport går igenom konstruktionen och programmerandet av ett snakespel. Spelet består av en processor, två joystickar och en grafisk display. Dessa kopplas samman och spelet programmeras sedan i Atmel Studio. Rapporten går även igenom hur konstruktionen och programmerandet har gått till samt vilka komponenter som använts. I diskussionen redogörs bland annat eventuella hinder i processen.

Resultatet blir ett fungerande snakespel med två spelare som kan spela mot varandra, spelet håller en bra hastighet vilket gör det roligt men lagom svårt.

Innehållsförteckning

1. Inledning

Inspiration till spelet har hämtats från spelet Snake, som går ut på att styra en orm som samlar äpplen. För varje äpple som ormen samlar så blir den längre. Detta koncept tas vidare fast med två spelare istället för en i syfte att spelarna tävlar mot varandra. Kolliderar spelare 1 med spelare 2, så har spelare 1 förlorat och tvärtom.

1.1 Syfte

Syftet med projektet i Digitala System är att skapa en fungerande prototyp med hjälp av mikroprocessorn ATMega1284. I detta fallet ett snakespel med tvåspelarfunktion som styrs av två joysticks, en till vardera spelare. Spelet skrivs i programspråket C.

1.2 Kravspecifikation

Följande krav ska uppfyllas:

- Spelet ska kunna spelas av två spelare samtidigt.
- De två ormarna ska kunna styras med varsin joystick.
- Det ska programmeras på mikroprocessorn ATMega1284.
- Spelet ska visas på en display med upplösningen 128x64 pixlar.
- Ormarna...
    - ska vara fyra pixlar breda.
    - ska kunna kollidera med varandra.
    - får inte åka utanför displayen.
    - ska svänga med 90° åt gången.
    - ska röra sig konstant framåt.
    - får inte svänga mer än 90° åt gången.
- För varje äpple ormarna äter så ska de växa med fyra pixlar i längd.
- Joystickarnas riktning bestämmer vilket håll ormen ska röra sig.

2. Teori

2.1 Hårdvara

2.1.1 Processor

Processorn som används är en ATMega1284 mikrokontroller. Processorn har 40 pinnar där fyra av dem är reserverade för JTAG:en, vilket lämnar oss med 36 pinnar som vi kan använda. Pinnarna är indelade i 4 olika portar.

2.1.2 Display

Displayen som används är en 128x64 GDM12864HLCM-LCD-display. Displayen är uppdelad i två stycken delar där varje del är 64x64 pixlar. Varje del är även indelad i 8 sidor horisontellt med 8 pixlar i varje sida.

2.1.3 Joystick

För att styra ormarna så används två stycken joysticks. Varje joystick har 10 pinnar där 3 stycken går till processorn, 3 till ström och 2 till jord. Resterande två pinnar behöver inte användas.

2.1.4 JTAG

JTAG:en används för att kunna programmera processorn med programmet Atmel Studio.

2.1.5 Potentiometer

En potentiometer används för justera kontrasten på displayen.

2.2 Mjukvara

Programmet Atmel Studio har använts för att skriva och skicka koden till ATMega-processorn. Koden är skriven i C som är ett lågnivåspråk. Adobe Illustrator användes för att rita kopplingsschemat.

3. Metod

Ett kopplingsschema sammanställs över de komponenter som skall användas i projektet (*se bilaga 1*). Kopplingsschemat och kravspecifikationen presenterades därefter för handledaren. Efter att dessa hade blivit godkända så kunde arbetet med att sätta ihop komponenterna börja. Komponenterna fästes på en basplatta (*se bilaga 2*) och kopplades ihop enligt kopplingsschemat. Efter att allt hade kopplats ihop så testades skärmens bakgrundsbelysning.

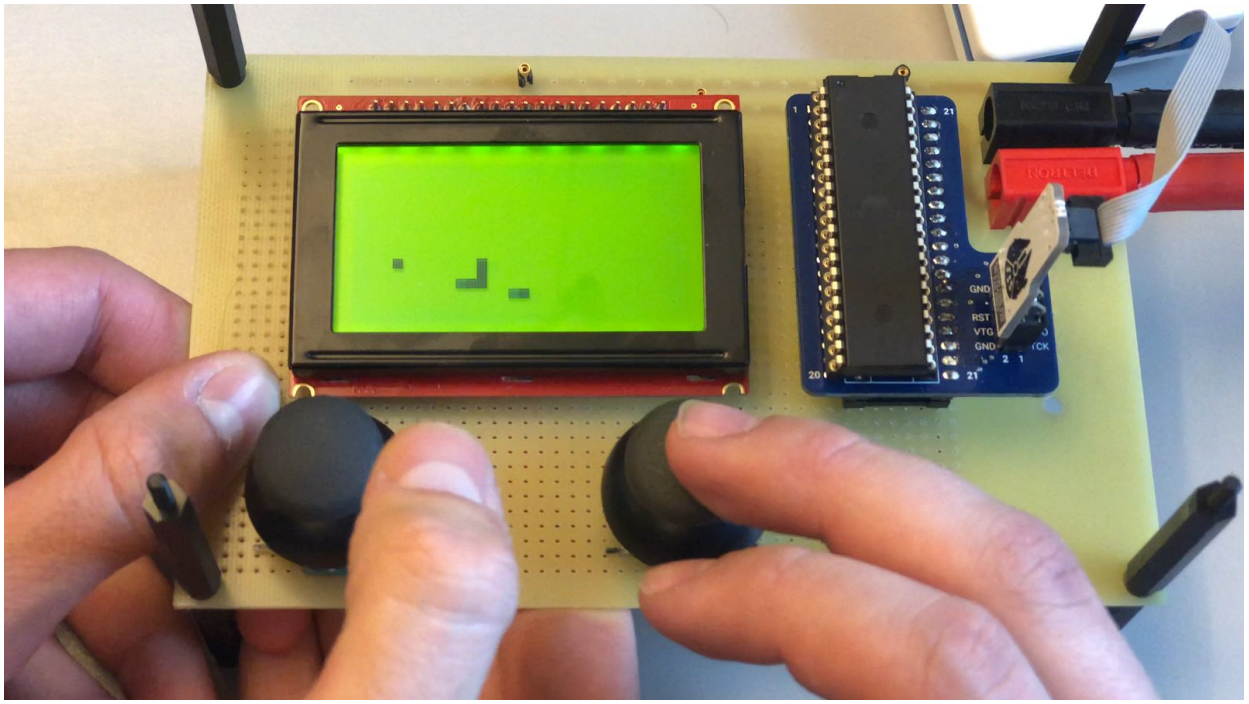
Först implementerades joystickarna. För att göra detta till en möjlighet användes pekare för att i sin tur komma åt minnesadressen. Fyra LED kopplades in för att lätt kunna se vilka signaler som resulterade upp, ner, vänster och höger. Mittpunkten på joystickarna kunde därefter anges för att sedan fullfölja implementeringen.

När joystickarna implementerats kunde implementeringen av den grafiska displayen påbörjas, som gjordes att programmera pixlar på skärmen. Kontrasten justerades med hjälp av potentiometern. När alla komponenter fungerar i enlighet påbörjades programmeringen av självaste spelet. Inledningsvis programmerades en orm och ett äpple in. När detta visades på skärmen programmerades styrning för ormen in.

Avslutningsvis behövde spelet presentera en vinnare när spelet avslutas. Detta gjordes enbart med kod och innebar att sätta varje pixel för sig. Koden skrevs dessutom på så sätt att texten var beroende av tre variabler, nämligen “x”, “y” samt “s”, vilket underlättade om texten behövdes öka i tjocklek eller justeras i x- och/eller y-led.

4. Resultat

Resultatet är ett väl fungerande multiplayer snakespel som uppfyller kravspecifikationen. Under spelets gång genereras äpplen på skärmen som spelaren sedan skall äta för att växa i storlek. Spelaren som först krockar med väggen, motståndaren eller i sin egna svans förlorar.



5. Diskussion

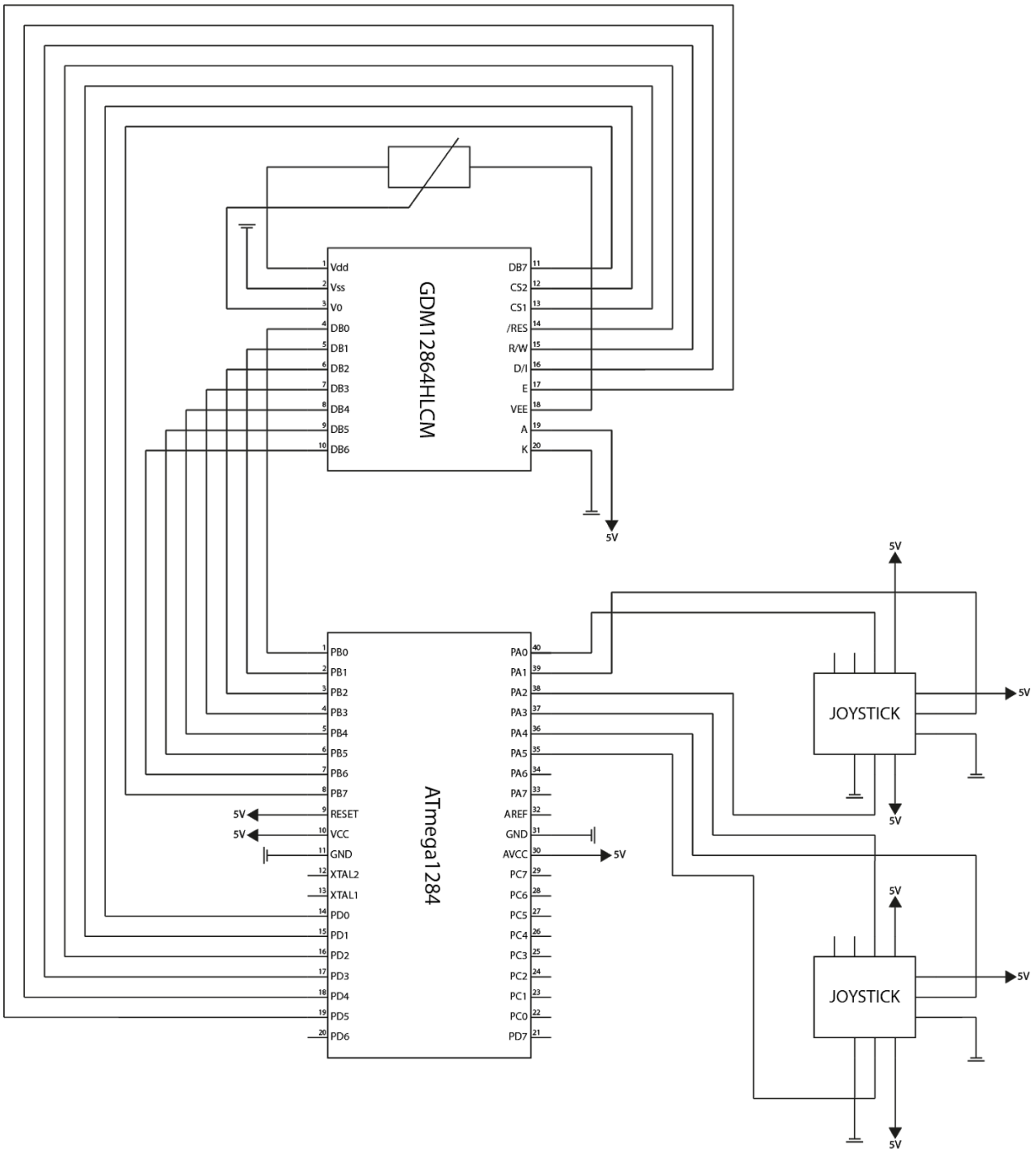
Början av arbetet gick för det mesta felfritt både med lödningen och med att vira. Komponenterna sattes ut på ett bra sätt vilket ledde till att vi inte stötte på några problem med att vira. De första hindret vi stötte på var att vi kopplat tvärtom till vår processor. Ett annat problem som uppkom när skärmen skulle testas, vilket var att vinkeln inte hade ändrats med hjälp av potentiometern och därav inte visade någonting på skärmen. Detta löstes med hjälp av en liten skruvmejsel.

Under tiden vi programmerade insåg vi snabbt att pixlarna inte hamnade på rätt ställen. Lyckligtvis upptäcktes det tidigt att displayen var installerad upp-och-ner, något vi snabbt åtgärdade med hjälp av lite enkel matematik.

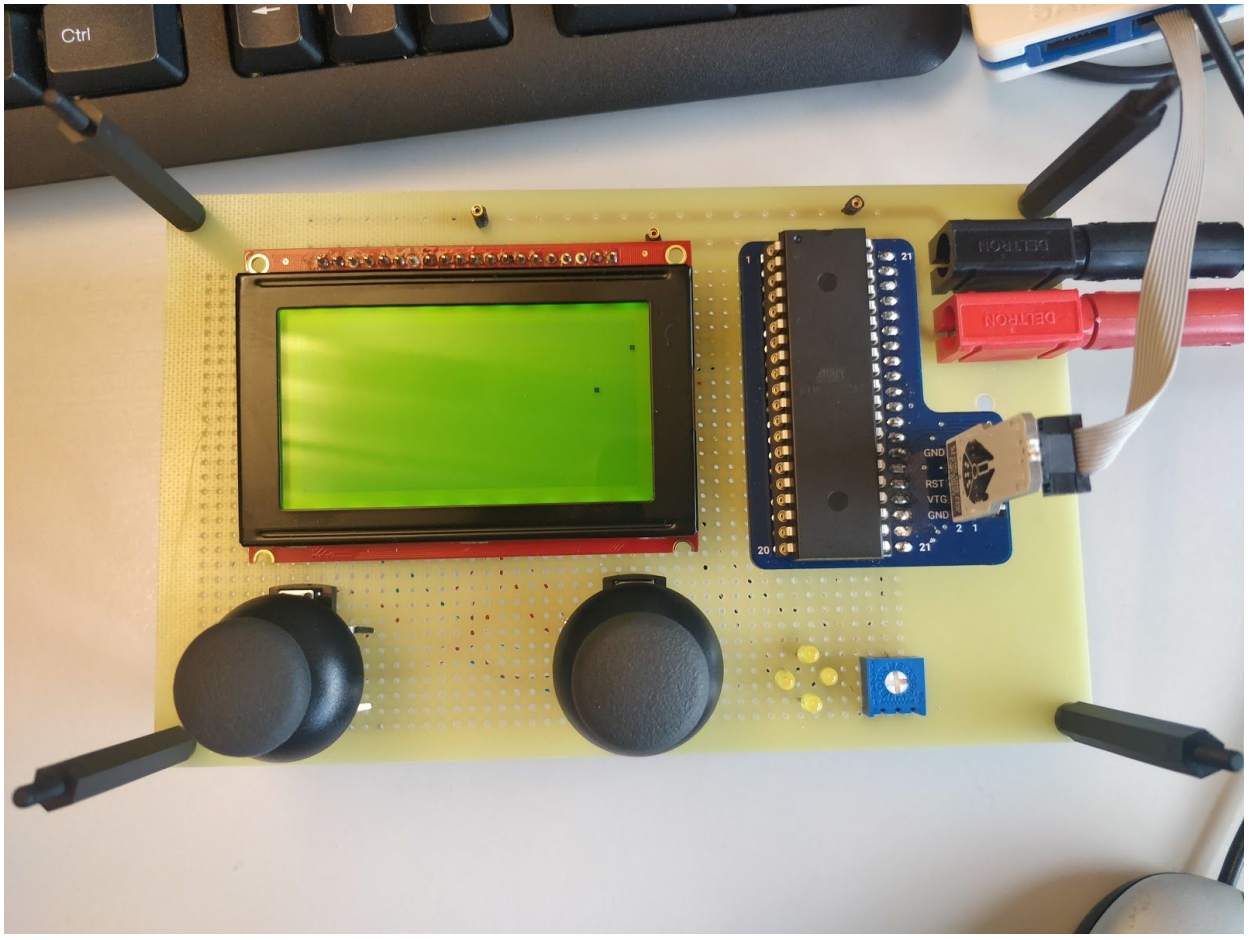
Det sista problemet vi stötte på förekom också i programmeringen. Spelet var nu fungerande, men det var inte responsivt nog. Anledningen till detta var att processorn kördes på 1 MHz. Klockfrekvensen ökades till 8 MHz, något som resulterade i tillfredsställande responsivitet. Den ökade frekvensen introducerade dock ett nytt problem; grafiken på skärmen ritades inte längre ut korrekt. Detta uppstod eftersom skärmen inte längre hängde med processorn. För att lösa detta implementerades en subrutin som användes för att pausa källkods exekveringen tills den grafiska displayen var redo för ett nytt kommando.

6. Appendix

Bilaga 1



Bilaga 2



Bilaga 3 Källkod

(Se bifogad länk)

https://github.com/OliverEkberg/atmega-snake

7. Referenser

Kernghan, Brian W. och Ritchie, Dennis M. 1988. The C Programming Language. 2. uppl. Englewood Cliffs: Prentice Hall.

Datablad för display
GDM12864HLCM
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/GDM12864H.pdf

(Hämtad 2019-05-15)

Datablad för processor
AVR ATMega1248
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega1284.pdf (hämtad 2019-05-15)