Oppgave A – Spillkonsoll med Arduino UNO

Hva jeg har laget

Mitt valg falt på oppgave A da jeg synes det virket veldig spennende å lage en egen spillkonsoll og eget spill. Samtidig så jeg på oppgaven som en fin utfordring og mulighet for å lære mer om å kode i C/C++, noe jeg ikke har brukt utenom dette emnet. Jeg tenkte på flere typer spill, bl.a. Snake og Tetris. Snake ble for enkelt synes jeg, og i Tetris er musikken halve spillet. Jeg ville også bruke joysticken til styring så valget falt på en type Space Shooter, hvor man skal styre et romskip som prøver å unngå å bli truffet av meteorer som kommer imot i stadig større fart og antall.

Space Shooter

Når man starter spillet blir man møtt av en meny hvor man kan velge «New Game» eller «Highscore». Spillet går ut på å overleve lengst mulig, samtidig som man skyter ned meteorer for å få høyest mulig poengsum. Blir man truffet av en meteor, er spillet over. Tiden man lever og antallet meteorer man skyter beregner poengsummen. Det er kun én vanskelighetsgrad, men det blir vanskeligere jo lenger man lever. Spillerens motivasjon er å slå sin egen highscore.

Styring

Man styrer romskipet opp og ned ved å bruke de to knappene på venstre side og skyter med knappen på høyre side. Planen var å benytte joystick for å styre med, men den virket ikke som den skulle. Jeg koblet den opp og linket X og Y verdiene mot Serial Monitor og fant ut av at verdiene ikke alltid var riktige. Når Joysticken står rett opp skal den ha en verdi +- 512 på begge aksene, noe den hadde helt til jeg beveget joysticken i en retning. Da gikk den stort sett til +- 0 på begge aksene og når den gikk tilbake i utgangsposisjon, justerte ikke verdien seg tilbake. Den viste også +-0 på 3 av 4 retninger. Da fant jeg ut at jeg i stedet måtte bruke to knapper for å styre romskipet opp og ned og dette fungerer utmerket, men det hadde vært litt morsommere med joysticken.

Poeng

Poengsummen til spilleren vises oppe i høyre hjørne når man spiller spillet og beregnes på bakgrunn av tid spilt * 1.1 for hver meteor man skyter ned. Dermed vil det være en betydelig fordel å skyte meteorer istedenfor å bare unngå dem.

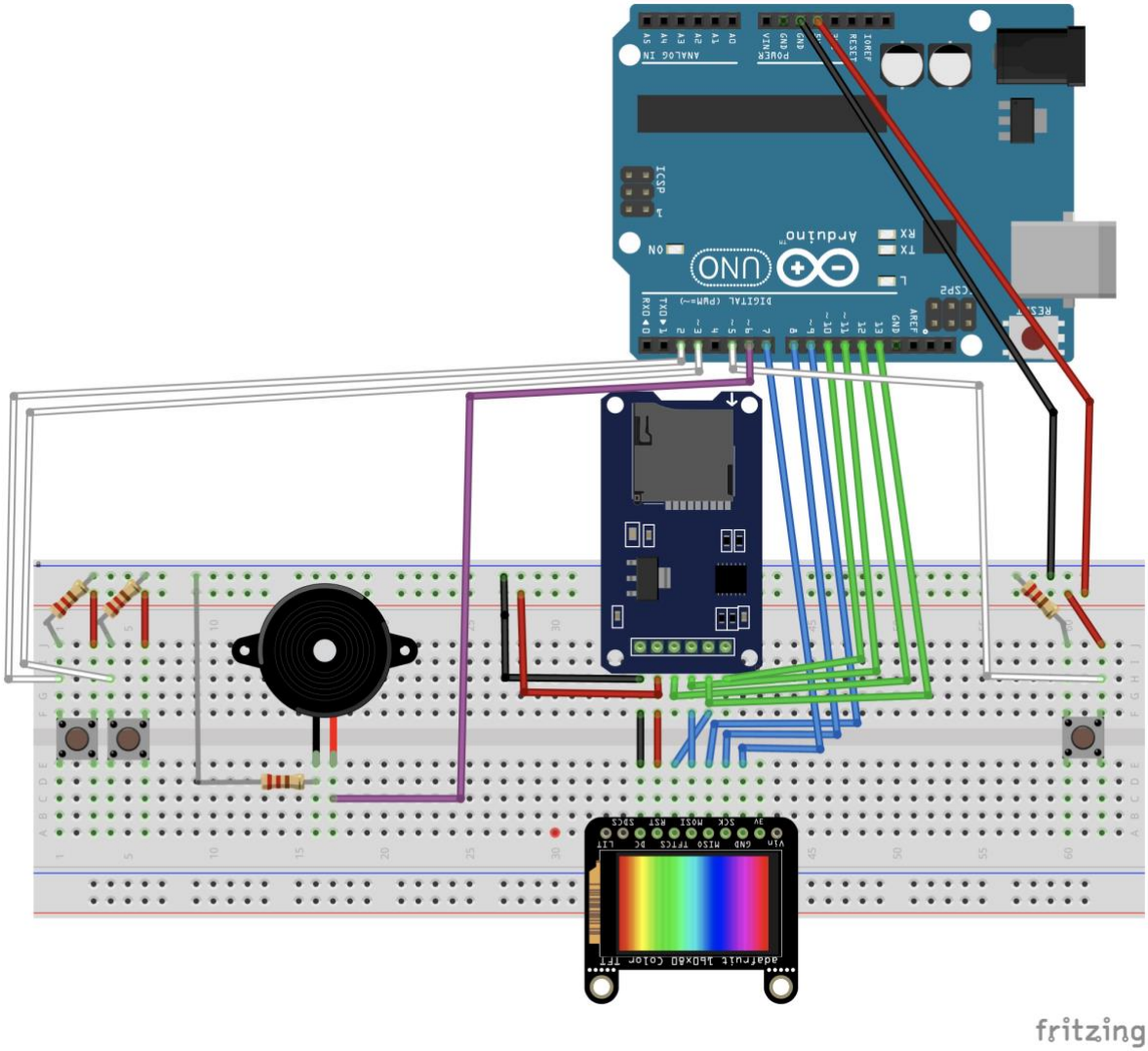
```
uint16_t getScore() {
  return ((currentMillis - gameStartMillis) / 100) * (1.0 + (hits / 10.0));
}
```

Den høyeste poengsummen lagres på minnekortet etter at spillet er over, og man kan se den høyeste poengsummen fra menyen ved å velge Highscore. Ønsker man å nullstille highscoren kan man trykke på skyteknappen når man viser highscore og et spørsmål om å nullstille vil komme opp på skjermen. Dette vil slette highscore-filen på minnekortet og opprette en ny fil med 0 som verdi. Jeg har valgt å lagre hver gang man får en ny highscore i tilfelle man ville se utviklingen ved en senere anledning. Når spilleren kræsjer med en meteor, vil det komme opp et rødt vindu med poengsum hvis man ikke klarer å slå sin egen highscore, eller et grønt vindu dersom man klarer det. I tillegg vil spille lage en

«triumferende» lyd dersom man får ny highscore, og motsatt om man ikke klarer det. Etter 3 sekunder kan man da trykke på skyteknappen for å komme tilbake til menyen.

Oppkobling

Spillkonsollen var relativt enkel å koble opp da jeg har brukt både skjerm og andre SPI-enheter samtidig i tidligere arbeidskrav. Skjermen og minnekortet deler SPI-tilkobling, men med noen egne koblinger, bl.a. CS-pinnen. De tre knappene er koblet med en 220ohms motstand hver, og det samme med høyttaleren da jeg synes det ga den mest riktige lyden.



Det som var mest problematisk med oppkoblingen, bortsett fra at joysticken ikke fungerte riktig, var å få skjermen og minnekortet til å fungere samtidig. Ved litt søking på arduino forumet fant jeg ut at et bibliotek som het SdFat fungerte bedre sammen med skjermen og valgte å benytte dette.

Koden

Dette var delen som var mest utfordrende og en av hovedgrunnene til at jeg valgte å lage spillkonsoll. Min eneste erfaring med C/C++ er i dette emnet og jeg så på det som en spennende utfordring å kode et spill.

Jeg har brukt Adafruit ST7735/Adafruit GFX bibliotekene for skjermstyring og SdFat for kommunikasjon med SD-kort. Jeg har laget egne klasser av objektene som er i spillet:

- Fighter – romskipet (egentlig bare en trekant) som spilleren styrer.
- Laser – prosjektilene som skytes ut av romskipet
- Meteor – meteorene som kommer i mot spilleren og som skal skytes ned.

Dette gjorde jeg for å enklere kunne tegne opp elementene på skjermen og vurdere sammenstøt, spesielt når flere av samme element skal tegnes opp samtidig. Meteorer og laser-prosjektiler er lagt til i arrayer slik at jeg kan løpe igjennom arrayene for å sjekke og manøvrere alle synlige elementer ved få linjer kode. Jeg burde kanskje puttet klassene i egne filer, men jeg vurderte det slik at prosjektet var såpass lite og at klassene kun er ment for bruk i dette prosjektet at jeg synes det var mest oversiktlig å ha de i samme kodefil.
Det første problemet jeg kom bort i var timing. I prosjektene jeg har laget hittil i emnet har jeg brukt delay() for å vente på at noe skal skje. Til et spill vil ikke dette fungere når flere ting skal times samtidig, i og med at alt blir satt på vent når man bruker delay(). Men etter et raskt søk på nettet fant jeg et tips om arduino sin eksempelfil «BlinkWithoutDelay» som bruker formelen:

```
(currentMillis - previousMillis >= interval)
```

for å time ting som blir gjort og jeg brukte dette for å time flere ting samtidig.
Alle meteorer og laser-prosjektiler som er synlige blir hele tiden sjekket om kolliderer med hverandre eller med romskipet gjennom metodene:

```
bool checkCollision(Fighter fighter, Meteor meteor) {
  if (fighter.x0 - meteor.x0 < METEOR_SIZE && fighter.y0 - meteor.y0 < METEOR_SIZE) {
    return true;
  } else if (fighter.x1 - meteor.x0 < METEOR_SIZE && fighter.y1 - meteor.y0 < METEOR_SIZE) {
    return true;
  } else if (fighter.x2 - meteor.x0 < METEOR_SIZE && fighter.y2 - meteor.y0 < METEOR_SIZE) {
    return true;
  }
  return false;
}

bool checkShotHit(Laser laser, Meteor meteor) {
  if (laser.x1 - meteor.x0 < METEOR_SIZE && laser.y1 - meteor.y0 < METEOR_SIZE) {
    return true;
  }
  return false;
}
```

For hver 5. meteor man overlever vil hastigheten på meteorene og spawn-frekvensen øke til en satt maks grense hvor spillet er på sitt vanskeligste. Dette vil forhåpentligvis gjøre spillet såpass vanskelig at man ikke kan holde på i en evighet, noe som bidrar til å gjøre det mer utfordrende og gøy å prøve å slå sin egen highscore.
Det var også litt utfordrende å håndtere «states» når jeg skulle lage menyene. I og med at alt skjer i loop() så måtte jeg lage forskjellige bool-variabler som håndterte states. Utfordringen var å «holde tunga rett i munnen» slik at den rette handlingen ble utført når man var på den ene eller andre menyen.

Jeg vil understreke at all kode-struktur i dette prosjektet er utviklet av meg selv, uten noen form for påvirkning fra andre ferdige utgaver av liknende prosjekter. Som skrevet tidligere, så jeg på dette som en spennende utfordring, og ved å studere andres liknende kode, ville jeg nok blitt litt låst i tankegangen og «kopiert» måten man hadde valgt å løse forskjellige problemer på. Jeg har kommentert i koden hvis jeg har tatt noe fra et annet sted.

Vanskeligheter

I tillegg til joystick som ikke fungerte som den skulle, har jeg hatt problemer med skjermen. Sort og hvitt er invertert, noe jeg løste ved å sette tft.invertDisplay(true); i setup(). Blått og rødt er også byttet om, og jeg mistenker at skjermen ikke er helt støttet av Adafruit-bibliotekene som er benyttet, men har ikke klart å bekrefte om det er tilfellet eller om skjermen er defekt. Samtidig er det en litt grumsete hvit stripe langs to av sidene. Jeg prøvde med et annet tilpasset skjermbibliotek (uTFT library by Henning Karlsen, https://cdn.shopify.com/s/files/1/2386/9605/files/UTFT.pdf?2177) hvor sort og hvitt var korrekt og stripene langs sidene var borte, men det støttet kun software basert SPI, noe som er altfor tregt til å lage et spill.

Konklusjon

Dette prosjektet har vært veldig spennende og lærerikt å holde på med. Utfordringene jeg har møtt underveis har gjort at jeg har måttet utforske forskjellige løsninger og jeg synes resultatet har blitt bra. Jeg kan sitte lenge og prøve å slå min egen rekord, og synes faktisk spillet er morsomt.
Jeg har lært mye om koding på Arduino og hvordan man håndterer states, og flere ting samtidig. Samtidig har jeg lært at kapasiteten til en Uno er overaskende god, at den klarer såpass mye beregning uten at ting stopper opp.
Jeg har også måttet feilsøke komponenter ved å sjekke verdiene i Serial-monitor og ved å prøve ut forskjellige biblioteker og min totalforståelse for hvordan Arduinoverdenen fungerer har blitt mye bedre.

Vedlegg

- Vedlegg 1: Koblingsskjema breadboard
- Vedlegg 2: Video av fungerende prosjekt
- Vedlegg 3: Fritzing-fil
- Vedlegg 4: Arduino kode-fil