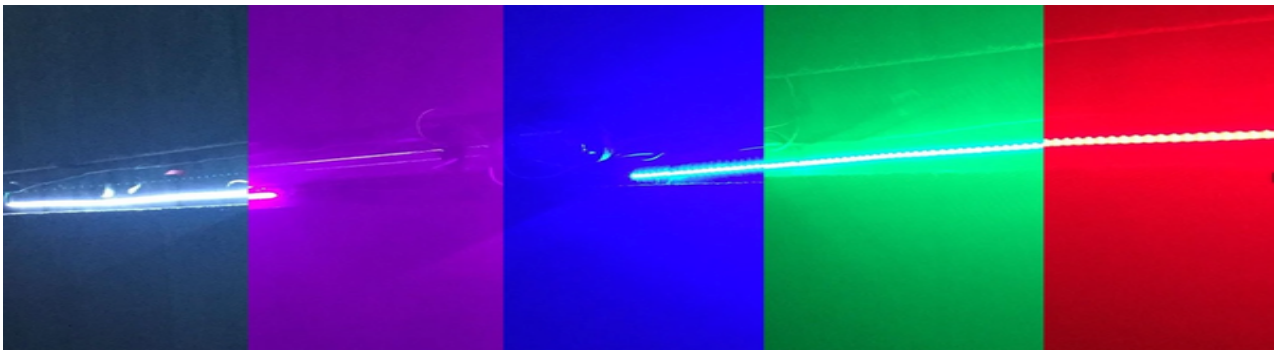
5.4 Kodning



Til produktet er der udviklet noget kode, der gør, at produktet er funktionsdygtigt. Koden kører på en Arduino Uno og har to dele; en lysblandingsdel til det statiske lys og en lysblandingsdel til det blinkende lys. De to kodedele bliver udløst gennem en kontakt, der kan sende enten 5 volt eller 0 volt til digital indgang nummer 7 på en Arduino Uno. Denne digitale indgang sørger for, at den korrekte del af koden bliver kaldt, alt efter hvilken position kontakten står i.

Den første del af koden består i at inkludere funktionsbiblioteket *FastLED*, at skabe og navngive heltalspladsholdere (integere), at allokere hukommelse til dem samt at prædefinere diverse konstanter, som skal bruges senere, når koden eksekveres. Ydermere oprettes et LED-array, som kan forstås som en tabel, hvor hver del i tabellen kan indekseres og kaldes som helhed eller som brudstykker der kan manipuleres med. Alt i denne del af koden eksekveres blot en enkelt gang, men data herfra kan hentes flere gange om nødvendigt.

Næste del af koden, setup, er en opsætningsprocedure og er, ligesom ovenstående kodedel, kun noget, der eksekveres én gang. I setupdelen etableres en kommunikationslinje mellem Arduinoen og computeren, så det kan tydeliggøres, hvilke pladsholdere, der har hvilke værdier. Herudover opsættes den bestemte LED-kædetype, og denne funktion skal have inputs i forbindelse med, hvor mange LED'er, der forefindes i den kæde, hvorfra LED-kæden skal modtage datasignal og have oplyst arrayet fra den første kodedel. Herudover bestemmes en maksimal lysstyrkeværdi på en skala fra 0 - 255, hvor 255 er den højest mulige lysstyrkeværdi. Intervallet 0 - 255 er 8 bit langt og er et gennemgående interval i hele koden, da det er det interval, RGB-farveblanding foretages i. I setupdelen ændres desuden diverse pins på Arduino Unoen til inputs, så de senere kan bruges til at modtage data fra noget omkringværende teknologi.

De to næste dele af koden består af to kaldbare funktioner. ”checkFadersMixer” og ”checkFadersBlink” er i virkeligheden identiske og gør derfor præcis det samme, men da de bliver kaldt under forskellige omstændigheder, er de blevet til hver deres funktion i stedet for blot én. Disse funktioner kaldes, alt efter hvad position den førnævnte kontakt står i. Hvis kontakten står således, at der tilføres 5 volt til digital pin 7 på Arduino Unoen, så kaldes funktionen ”checkFadersMixer”. Hvis kontakten står, sådan at der læses 0 volt i digital pin 7 på Arduino Unoen, kaldes funktionen ”checkFadersBlink”. Ordet ’fader’ er engelsk og angiver blot, at der ikke er tale om en drejeknap, men snarere en knap, som kan føres frem eller tilbage, alt efter om det ønskes at skrue henholdsvis op eller ned for et givent signal. Det, disse to funktioner i realiteten gør, er blot at tjekke, i hvilke positioner produktets fire lysmixermonterede fadere står i. Måden, hvorpå det sker, er, at hver af de fire fadere har et analogt output, og hvert af disse output kan have en værdi alt imellem 0 volt og 5 volt, fordi der konstant sendes 5 volt til faderne. En fader fungerer som en variabel modstand, og er en fader for eksempel skruet helt op, svarer det, i det her tilfælde, til, at den sender 5 volt ud gennem outputtet. Står faderen midt i fadervandringsområdet, svarer det til, at der sendes 2,5 volt ud gennem outputtet. Er faderen eksempelvis skruet helt ned, er outputtet lig med 0 volt og så videre. Alle de analoge faderoutput er forbundet til hver deres analoge input på Arduino Unoen, hvor værdierne herfra kan læses. Disse værdier kommer i analog form, og skal derfor digitaliseres, så signalet kan bruges i digital forstand. Digitaliseringen foregår ved, at hvert input har en inputanalyseopløsning på 128 bit og vil på en skala gå fra værdierne 0 – 1023. Dette betyder at hvad end analogt signal, der indlæses, formateres til at passe i denne nye skala. Hermed vil 0 volt efter formateringen blive værdien 0, og ligeledes vil 5 volt formateres til værdien 1023. Når det analoge signal er formateret til et digitalt, findes det som førnævnt på en 128 bit lang skala, der løber i intervallet 0 - 1023. Herudover er det også tidligere nævnt, at RGB-farveblanding foregår i en 8 bit lang skala, der løber i intervallet 0 - 255. Da de to skalaer ikke umiddelbart er sammenlignelige, skal det nyligt formaterede digitalsignal igen formateres, så det kan passe til RGB-farveblandingsskalaen. Formateringen foregår på samme måde som før, eksempelvis hvor værdien 1023 fra den første skala bliver til værdien 255 på RGB-farveblandingsskalaen. Værdien 0 fra første skala bliver til værdien 0 på RGB-farveblandingsskalaen. Når der formateres fra en skala til en anden, kaldes det *mapping*, og det er

her, kodelinjerne med begrebet map, eksempelvis map(valGREEN, 0, 1023, 0, 255), kommer i brug. I denne kodelinje står der, at inputtet valGREEN skal formateres til skalaen 0-1023 og derefter formateres yderligere til skalaen 0 - 255.

Den sidste del af koden, loop-delen, eksekveres kontinuerligt, og det er herfra, de to ovennævnte funktioner kan kaldes, alt efter hvilken position den førnævnte kontakt står i. Loop-delen begynder med at tjekke, om der læses 5 volt eller 0 volt på Arduino Unoens digitale pin nummer 7. Kontaktens position har indflydelse herpå. Hvis der læses 5 volt, kaldes funktionen ”checkFadersMixer” og henter info om faderpositionerne gennem denne funktion. Et eksempel kunne være, at faderen for rød farve og Master-faderen var skruet helt op. Dette vil så resultere i, at når der kommes tilbage fra funktionen ”checkFadersMixer” og til loop-funktionen, køres der videre til de to næste kodelinjer, der tilsammen initialiserer LED-kæden og aktiverer alle LED’er i kæden med rød farve og fuld lysstyrke. Dette sker fordi faderpositionerne blev kaldt gennem funktionen ”checkFadersMixer”, og værdierne, der blev udledt derfra, blev så sat som input i funktionen CRGB(RED, GREEN, BLUE). I tilfældet med kun faderen for rød skruet helt op vil det selvfølgelig resultere i kun rødt lys fra LED’erne, men hvis nu også faderen for grøn havde været skruet 50% op, ville dette resultere i et orange lys og så videre.
Lad det antages, at kontakten skifter position, og at der derfor ikke længere sendes 5 volt til digital pin nummer 7 på Arduino Unoen. I så fald startes det else-statement, der ligger i forlængelse af det tidligere if-statement. Når der gås ind i else-statementet, startes der med at kalde funktionen ”checkFadersBlink”. Her sker der det samme som før, altså der kan udledes en bestemt farve alt efter, hvordan faderne er positionerede. Bagefter tjekkes digital pin 8 på Arduino Unoen for et digitalsignal, der enten kan være højt eller lavt. Signalet, der bliver tjekket efterfølgende, kommer fra en Arduino Mega, som kører et stykke beatdetektor-kode, der kan udsende impulser af digitalt højt signal til Arduino Unoen, alt efter hvilket tempo noget tilkoblet musik spiller i. Herefter sker der følgende; Der tjekkes hele tiden for et signal fra Arduino Megaen til Arduino Unoen. Det signal, der kan være tale om, er enten højt eller lavt. Uanset hvilket signal der forekommer, allokeres værdien til integeren IN_PIN. Det vil sige, at værdien for integeren IN_PIN er skiftende over tid. Herefter tjekkes det, om integerne IN_PIN og PIN_STATE har samme værdi. Integeren PIN_STATE har som udgangspunkt værdien lav, så hvis PIN_STATE er lav, og IN_PIN er

høj, altså anderledes fra hinanden, startes et if-statement, der sørger for, at de to integere får samme værdi, og hvor værdien de tildeles afspejles af, hvad end værdi IN_PIN har. Det samme sker også, dersom udsagnet havde været omvendt. Betragtes PIN_STATE, og har den værdien høj, kan der fortsættes med at initialisere LED-kæden og at aktivere alle LED'er i kæden med den farveblanding, der stemmer overens i henhold til faderpositionerne. Hernæst lyser LED'erne i den forudblandede farve i 150 millisekunder, hvorefter der skiftes til et blackout, dette giver en blinkeeffekt. I mellemtiden er værdien for IN_PIN blevet lav, fordi der i et udefineret tidsrum ikke har været et højt taktudslagssignal fra beatdetektoren, og derfor er PIN_STATE ligeså blevet lav. Når PIN_STATE er lav, forekommer der ligeledes blackout. Når der så igen læses et højt taktudslagssignal fra beatdetektoren, bliver IN_PIN værdien høj, og det medfører, at PIN_STATE også får værdien høj, og så startes der forfra med det blinkede lys.

5.4.1 Koden

```
#include "FastLED.h"                    ”Inkluderer FastLED-biblioteket”
#define NUM_LEDS 150                    ”Definerer antallet af LED’er på strippen”
#define DATA_PIN 4                      ”Definerer PIN 4 som Data out”
#define BRIGHTNESS 255                  ”Definerer den maksimale lysstyrke”

int potPinMASTER = A0;                  ”Initialiserer en integer og navngiver analog PIN 0”
int potPinRED = A1;                     ”Initialiserer en integer og navngiver analog PIN 1”
int potPinGREEN = A2;                   ”Initialiserer en integer og navngiver analog PIN 2”
int potPinBLUE = A3;                    ”Initialiserer en integer og navngiver analog PIN 3”

int valMASTER;                          ”Initialiserer og navngiver en integer”
int MASTER;                             ”Initialiserer og navngiver en integer”
int valRED;                             ”Initialiserer og navngiver en integer”
int RED;                                ”Initialiserer og navngiver en integer”
int valGREEN;                           ”Initialiserer og navngiver en integer”
int GREEN;                              ”Initialiserer og navngiver en integer”
int valBLUE;                            ”Initialiserer og navngiver en integer”
int BLUE;                               ”Initialiserer og navngiver en integer”
int SWITCH_PIN = 7;                     ”Initialiserer en integer og navngiver digital PIN 7”
int TRIGGER_PIN = 8;                    ”Initialiserer en integer og navngiver digital PIN 8”
int IN_PIN;                             ”Initialiserer og navngiver en integer”
int PIN_STATE = LOW                     ”Initialiserer og navngiver en integer. Tildeles
                                        værdien ’lav’”

CRGB leds[NUM_LEDS];                    ”Initialiserer LED-arrayet og får input via
                                        NUM_LEDS”
```

```
//*****************************************//
//_____________________Setup_____________________//
//*****************************************//

void setup() {                                          "Kalder setup-funktionen. Alt i denne funktion køres blot én gang"

  delay(2000);                                          "Opstartsforsinkelse"

  Serial.begin(115200);                                 "Initialiserer kommunikation mellem Arduino og computer"

  FastLED.addLeds<WS2812B, DATA_PIN>(leds, NUM_LEDS);   "Kalder en specifik FastLED-funktion fra FastLED-biblioteket. Funktionen skal have at vide, hvilken type LED-strip, der er tale om, samt hvor funktionen skal sende data ud, og hvor mange LED'er der er i strippen"

  FastLED.setBrightness(BRIGHTNESS);                    "Kalder en specifik FastLED-funktion fra FastLED-biblioteket. Funktionen sætter lysstyrken for LED-strippen og får input via BRIGHTNESS"

  pinMode(potPinMASTER, INPUT);                         "Ændrer pinMode til input"
  pinMode(potPinRED, INPUT);                            "Ændrer pinMode til input"
  pinMode(potPinGREEN, INPUT);                          "Ændrer pinMode til input"
  pinMode(potPinBLUE, INPUT);                           "Ændrer pinMode til input"

  pinMode(SWITCH_PIN, INPUT);                           "Ændrer pinMode til input"
  pinMode(TRIGGER_PIN, INPUT);                          "Ændrer pinMode til input"

}
```

```
//*****************************************//
//_____________checkFadersMixer _____________//
//*****************************************//

void checkFadersMixer() {                          "Kalder checkFadersMixer-
                                                   funktionen. Alt i denne funktion
                                                   køres kontinuerligt, hvis funktionen
                                                   kaldes i loop-funktionen"

 valMASTER = analogRead(potPinMASTER);             "Allokerer læste værdi fra
                                                   potPinMASTER i valMASTER"
 MASTER = map(valMASTER, 0, 1023, 0, 255);         "Allokerer en faderposition
                                                   via et manipuleret talinterval
                                                   til integeren MASTER"
 FastLED.setBrightness(MASTER);                    "Kalder en specifik FastLED-
                                                   funktion fra FastLED-biblioteket.
                                                   Funktionen sætter lysstyrken
                                                   for LED-strippen og får input via
                                                   MASTER"

 valRED = analogRead(potPinRED);                   "Allokerer læste værdi fra
                                                   potPinRED i valRED"
 RED = map(valRED, 0, 1023, 0, 255);               "Allokerer en faderposition
                                                   via et manipuleret talinterval
                                                   til integeren RED"

 valGREEN = analogRead(potPinGREEN);               "Allokerer læste værdi fra
                                                   potPinGREEN i valGREEN"
 GREEN = map(valGREEN, 0, 1023, 0, 255);           "Allokerer en faderposition
                                                   via et manipuleret talinterval
                                                   til integeren GREEN"

 valBLUE = analogRead(potPinBLUE);                 "Allokerer læste værdi fra
                                                   potPinBLUE i valBLUE"
 BLUE = map(valBLUE, 0, 1023, 0, 255);             "Allokerer en faderposition
                                                   via et manipuleret talinterval
                                                   til integeren BLUE"

delay(50);                                         "Forsinkelse 50ms"
}
```

```
//*****************************************//
//_____________checkFadersBlink. _____________//
//*****************************************//

void checkFadersBlink() {                          "Kalder checkFadersBlink-funktionen. Alt i denne funktion køres kontinuerligt, hvis funktionen kaldes i loop-funktionen"

  valMASTER = analogRead(potPinMASTER);            "Allokerer læste værdi fra potPinMASTER i valMASTER"
  MASTER = map(valMASTER, 0, 1023, 0, 255);        "Allokerer en faderposition via et manipuleret talinterval til integeren MASTER"

  FastLED.setBrightness(MASTER);                   "Kalder en specifik FastLED-funktion fra FastLED-biblioteket. Funktionen sætter lysstyrken for LED-strippen og får input via MASTER"

  valRED = analogRead(potPinRED);                  "Allokerer læste værdi fra potPinRED i valRED"
  RED = map(valRED, 0, 1023, 0, 255);              "Allokerer en faderposition via et manipuleret talinterval til integeren RED"

  valGREEN = analogRead(potPinGREEN);              "Allokerer læste værdi fra potPinGREEN i valGREEN"
  GREEN = map(valGREEN, 0, 1023, 0, 255);          "Allokerer en faderposition via et manipuleret talinterval til integeren GREEN"

  valBLUE = analogRead(potPinBLUE);                "Allokerer læste værdi fra potPinBLUE i valBLUE"
  BLUE = map(valBLUE, 0, 1023, 0, 255);            "Allokerer en faderposition via et manipuleret talinterval til integeren BLUE"

delay(50);                                         "Forsinkelse 50ms"
}
```

```
//*****************************************//
//_____________________loop_____________________//
//*****************************************//

void loop() {                                   "Hoved-funktionen. Alt i denne
                                                funktion køres kontinuerlig,t og
                                                det er her, de ovenstående
                                                funktioner kaldes fra"


 if (digitalRead(SWITCH_PIN) == HIGH) {         "Læser input fra vippekontakt.
                                                Hvis signalet høj læses fortsættes
                                                der inde i dette if-statement"
  checkFadersMixer();                           "Her kaldes checkFadersMixer-
                                                funktionen. Fra funktionen hentes
                                                integerne, MASTER, RED, GREEN,
                                                BLUE"
  for (int i = 0; i < NUM_LEDS; i++) {          "For-statement der initialiserer
                                                og trinvis øger integeren 'i' indtil 'i'
                                                er ækvivalent med antallet af
                                                LED'er på LED-strippen"
   leds[i] = CRGB(RED, GREEN, BLUE);            "Integeren 'i' indsættes i led-
                                                arrayet og værdierne for integerne;
                                                RED, GREEN og BLUE indsættes i
                                                funktionen CRGB, som, alt efter
                                                forholdet mellem RED, GREEN og
                                                BLUE, angiver en bestemt farve"


  }
  FastLED.show();                               "Funktion, der sender data ud til
                                                LED-strippen og tænder for den"

 }

 else {                                         "Læser input fra vippekontakt.
                                                Hvis signalet 'lav' læses fortsættes
                                                der inde i dette else-statement"
  checkFadersBlink();                           "Her kaldes checkFadersBlink-
                                                funktionen. Fra funktionen hentes
                                                integerne, MASTER, RED, GREEN,
                                                BLUE"
  IN_PIN = digitalRead(TRIGGER_PIN);            "Læser input fra TRIGGER_PIN og
                                                allokerer værdien i integeren
                                                IN_PIN"
```

```
  if (PIN_STATE != IN_PIN) {                    "Hvis PIN_STATE er andeledes fra IN_PIN fortsættes i dette if-statement"

   PIN_STATE = IN_PIN;                          "PIN_STATE bliver nu lig med IN_PIN"

   }
   if (PIN_STATE == HIGH) {                     "Hvis PIN_STATE er lig med 'høj' fortsættes i dette if-statement"
    for (int i = 0; i < NUM_LEDS; i++) {        "For-statement der initialiserer og trinvis øger integeren 'i' indtil 'i' er ækvivalent med antallet af LED'er på LED-strippen"

     leds[i] = CRGB(RED, GREEN, BLUE);          "Integeren 'i' indsættes i led-arrayet og værdierne for integerne; RED, GREEN og BLUE indsættes i funktionen CRGB, som, alt efter forholdet mellem RED, GREEN og BLUE, angiver en bestemt farve"

    }

    FastLED.show();                             "Funktion, der sender data ud til LED-strippen og tænder for den"

    delay(150);                                 "Forsinkelse 150ms"

    for (int i = 0; i < NUM_LEDS; i++) {        "For-statement, der initialiserer og trinvis øger integeren 'i' indtil 'i' er ækvivalent med antallet af LED'er på LED-strippen"

     leds[i] = CRGB(0, 0, 0);                   "Integeren 'i' indsættes i led-
arrayet og funktionen CRGB initialiserer black-out"

    }
    FastLED.show();                             "Funktion der sender data ud til LED-strippen og tænder for den"

   }
   else {                                       "Hvis PIN_STATE er lig med 'lav' fortsættes i dette else-statement"
    for (int i = 0; i < NUM_LEDS; i++) {        "For-statement der initialiserer og trinvis øger integeren 'i' indtil 'i' er ækvivalent med antallet af LED'er på LED-strippen"

     leds[i] = CRGB(0, 0, 0);                   "Integeren 'i' indsættes i led-arrayet og funktionen CRGB initialiserer black-out"
```

```
      }
      FastLED.show();                    "Funktion, der sender data ud til LED-strippen og tænder for den"

    }
   }
  }
}
```

5.5 Delkonklusion

Det kan konkluderes, at *Soft Design Science Methodology* har været en brugbar designprocess, som har resulteret i et endeligt design. På baggrund af metodens principper er der gjort grundige overvejelser over hver enkelt del af designet. Det har metoden gjort ved at sikre evaluering af designprocessens enkelte faser. Hver evaluering har ført til genovervejelse af de valg, som er taget i forrige faser, og produktet er igennem hver evaluering blevet optimeret. Herudover kan det konkluderes, at projektet har taget højde for resultaterne af undersøgelserne, samt teorien om lys når det kommer til designet. Det kan det, fordi mange aspekter af designet er rationaliseret på baggrund af førnævnte resultater og teori. Derfor kan det også konkluderes at belysningssystemet på kollegiet Korallen kan redesignes ved hjælp af; metoden *Soft Design Science Methodology*, undersøgelser af målgruppen og miljøet, samt viden om lys og mennesker.
Ydermere kan det konkluderes at kodning kan bruges til at realisere et design, hvilket har spillet en væsentlig rolle i projektet.