IxD Kernmodule II

Ruimtelijke interactie

Technische documentatie

Door Luuk Siewers

HKU Games & interactie, jaar 2

Concept & prototype

Concept in het kort

Het concept draait om het beurtbalkje die je op de band legt wanneer jij of diegene voor je klaar is met producten op de band te leggen. Op het moment dat je het balkje oppakt, gaat het naar je glimlachen, gloeit het op en zegt het gedag. Wat betreft vorm moet hij het bekende balkje blijven, maar met meer ronde vormen (afgeronde hoeken) en een zacht materiaal. Het materiaal moet ligt doorzichtig zijn, zodat de LED’s vervaagd worden, maar de kleur wel zichtbaar zijn. Hiermee hoop ik de rij op te vrolijken, zodat de algemene sfeer bij de klassieke kassa warmer en vrolijker wordt.

Prototype

Het prototype is een basale presentatie van het eindconcept, die niet zo slim is. Hij mist een of meer sensoren/actuatoren en afwerking in de code ten opzichte van wat het uitgewerkte concept in werkelijkheid moet worden. en waarbij de buiten van karton is in plaats van mooi afgewerkt, zacht en rond materiaal.

Sensoren

Het prototype bevat een LDR sensor aan de onderkant, zodat het weet of het neergezet is of vastgehouden wordt. Op basis daarvan kan hij reageren. Als ik meer tijd en de benodigde sensoren had, had ik willen meten of het prototype op een rails staat of op een band, zodat het anders reageert naar de mensen in de rij dan naar de cassière. Dit had eventueel gekund met een capacitive sensor. De rubberen band geleidt niet, maar de metalen rails natuurlijk wel en dus kan de sensor een onderscheid herkennen.

Actuatoren

In het prototype is duidelijk het schermpje te zien, waar de smiley op weergegeven wordt. In het prototype zit echter ook een RGB LED die een warme oranje-achtige kleur in-fade wanneer het opgepakt wordt en in die langzaam gaat ‘ademen’ (langzaam in en uit faden in het wit) wanneer het al een tijdje in de rails staat. Ik heb ook geprobeerd een speaker met SD-kaart module aan te sluiten die fragmenten afspeelt waarin gegroet wordt. Dit leek te werken, totdat de speaker in een lage frequentie bleef zoemen. Hierdoor waren de fragmenten dusdanig zacht te horen, dat ik het eruit gehaald heb. Deze heb ik een soort van goed gemaakt bij de presentatie om met mijn stem de speaker na te doen.

Behuizing

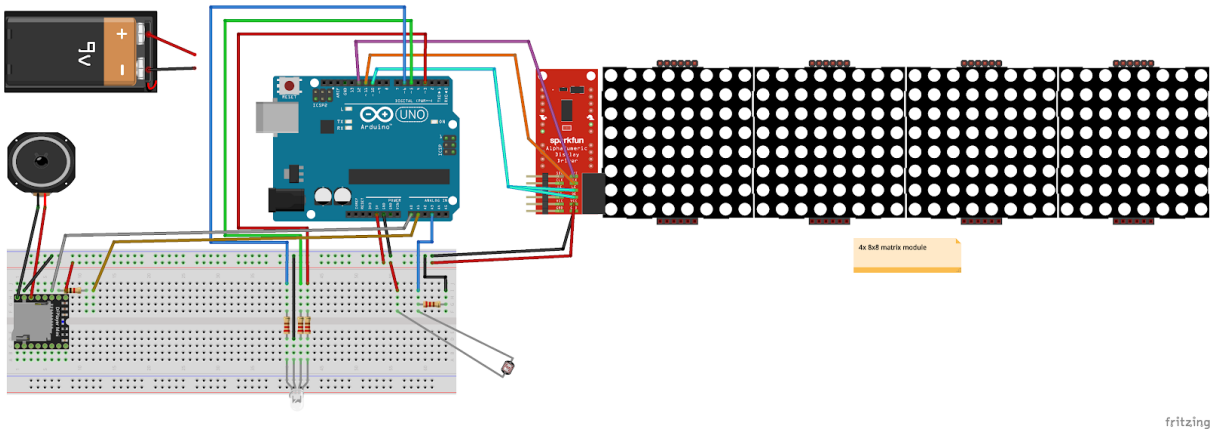
De behuizing was in de vorm van de klassieke beurtbalk; een langwerpige, driehoekvormige staaf. Dit was voor twee redenen: het concept als geheel is expres niet ver afgeweken van de originele beurtbalk om het vertrouwde gevoel van de kassa te behouden en voor dit prototype was het snel voor elkaar te krijgen met karton. In de behuizing zit de Arduino met de modules en bedrading met een powerbank als energiebron om hem draadloos te kunnen gebruiken.

Beelden prototype:
https://photos.google.com/share/AF1QipMOm-ENkUSnrINL3EOf_ER1AmUhMzcvxyWQm3a3xlfd88OritPBtCCilPzi4JGKxA?key=eGRwNjhUNFh4YmV6VUh1cjVGam5OWFljcHZWbUh3

Schema & code

Schema

Met de speaker en SD-kaart module, ziet mijn prototype er van binnen zo uit:



De 32x8 matrix zit aan elkaar vast met een driver als 1 module.

Code

Prototype

```
//
// matrix
//
#include "LedControl.h"

LedControl lc=LedControl(12,11,10,4);

float screenBrightness = 8;

//
// rgb led
//
int rPin = 3;
int gPin = 5;
int bPin = 6;
float rgbValue[3] = {0.0, 0.0, 0.0};
boolean fadeIn = false;

//
// light sensor (LDR)
//
int ldrPin = A3;
int ldrValue = 0;

//
// states
//
boolean pulledUp = false; // if there is too much light
boolean sleeping = false;
int pulledUpTimer = 0;
int pulledDownTimer = 0;
```

Initializeren variabelen en libraries

De LedControl.h helpt om een 32x8 matrix zeer makkelijk te besturen met functies als .setRow() en setLed().

```
void loop() {
  // LDR
  readLDR();

  // RGB led
  if(sleeping) { rgbSleeping(); }
  else { rgbOrange(); }

  delay(40); // small delay gives a more natural feel
}
```

```
void setup() {
  Serial.begin(9600);

  // matrix
  matrixInit();

  // rgb
  rgbInit();
  //  rgbOrange();

  // LDR
  pinMode(ldrPin, INPUT);
}
```

```
void matrixInit() {
  //we have already set the number of devices when we created the LedControl
  int devices=lc.getDeviceCount();
  lc.shutdown(0,true);
  lc.shutdown(3,true);

  /*The MAX72XX is in power-saving mode on startup*/
  lc.shutdown(1,false);
  lc.shutdown(2,false);
}

void setSmiley() {
  /* Set the brightness to a medium values */
  lc.setIntensity(1, screenBrightness);
  lc.setIntensity(2, screenBrightness);

  // the smiley is centered, and therefore split in 2 screens
  // mouth
  lc.setRow(2, 1, B00000011);
  lc.setRow(1, 1, B11000000);

  // corners
  lc.setLed(2, 2, 5, true);
  lc.setLed(1, 2, 2, true);
  lc.setLed(2, 3, 4, true);
  lc.setLed(1, 3, 3, true);

  // eyes
  lc.setLed(1, 6, 1, true);
  lc.setLed(2, 6, 6, true);
}

void clearMatrix() {
  lc.clearDisplay(1);
  lc.clearDisplay(2);
}
```

aan/uit moet.

Matrix functies

De matrixInit() functie zet de eerste en laatste 8x8 matrixen uit, omdat die niet gebruikt worden met de shutdown(index, aan?) functie van de LedControl library.

De setSmiley() functie is vanzelfsprekend. De setRow() functie vraag om een index van matrix, welke horizontale rij en dan de waarden van LED's in die rij. setLed vraagt om index, verticale en horizontale positie binnen die matrix en of die

```
//
// rgb functions
//

void rgbInit() {
  pinMode(rPin, OUTPUT);
  pinMode(gPin, OUTPUT);
  pinMode(bPin, OUTPUT);

  analogWrite(rPin, 0);
  analogWrite(gPin, 0);
  analogWrite(bPin, 0);
}

void rgbOrange() {
  float fadeSpeed = 0.5;

  // increase Red value if not 255
  if(rgbValue[0] < 255) {
     rgbValue[0] += fadeSpeed;
    analogWrite(rPin, rgbValue[0]);
  }

  // change Green value if not 7
  if(rgbValue[1] < 7) {
    rgbValue[1] += fadeSpeed/2;
  } else if (rgbValue > 7) {
    rgbValue[1] -= fadeSpeed;
  }
  analogWrite(gPin, rgbValue[1]);
}
```

RGB functies

De rgbInit() initialiseert de pins van iedere LED in de RGB LED en zet ze uit.

De rgbOrange() zet de kleur met een fade naar een oranje-achtige kleur. Op deze manier maakt het niet uit welke kleur de RGB eerst had om een vloeiende overgang van kleur te krijgen.

```
void rgbSleeping() {
  float fadeSpeed = 1;

  // breath-like fading
  // if fadeIn, increase values
  // else decrease
  if(fadeIn) {
//      Serial.println(rgbValue[0]);
    if(rgbValue[0] <= 255) {
      rgbValue[0] += fadeSpeed*2;
      analogWrite(rPin, rgbValue[0]);
    }
    if(rgbValue[1] <= 180.0) {
      rgbValue[1] += fadeSpeed*1.75;
      analogWrite(gPin, rgbValue[1]);
    }
    if(rgbValue[2] <= 255.0) {
      rgbValue[2] += fadeSpeed*2;
      analogWrite(bPin, rgbValue[2]);
    }
    if(rgbValue[0] >= 255.0 && rgbValue[1] >= 180.0 && rgbValue[2] >= 255.0 ) {
      fadeIn = false;
    }
  } else {
    if(rgbValue[0] > 0.0) {
      rgbValue[0] -= fadeSpeed;
      analogWrite(rPin, rgbValue[0]);
    }
    if(rgbValue[1] > 0.0) {
      rgbValue[1] -= fadeSpeed*0.75;
      analogWrite(gPin, rgbValue[1]);
    }
    if(rgbValue[2] > 0.0) {
      rgbValue[2] -= fadeSpeed;
      analogWrite(bPin, rgbValue[2]);
    }
    if(rgbValue[0] <= 0 && rgbValue[1] <= 0 && rgbValue[2] <= 0) {
      fadeIn = true;
    }
  }
```

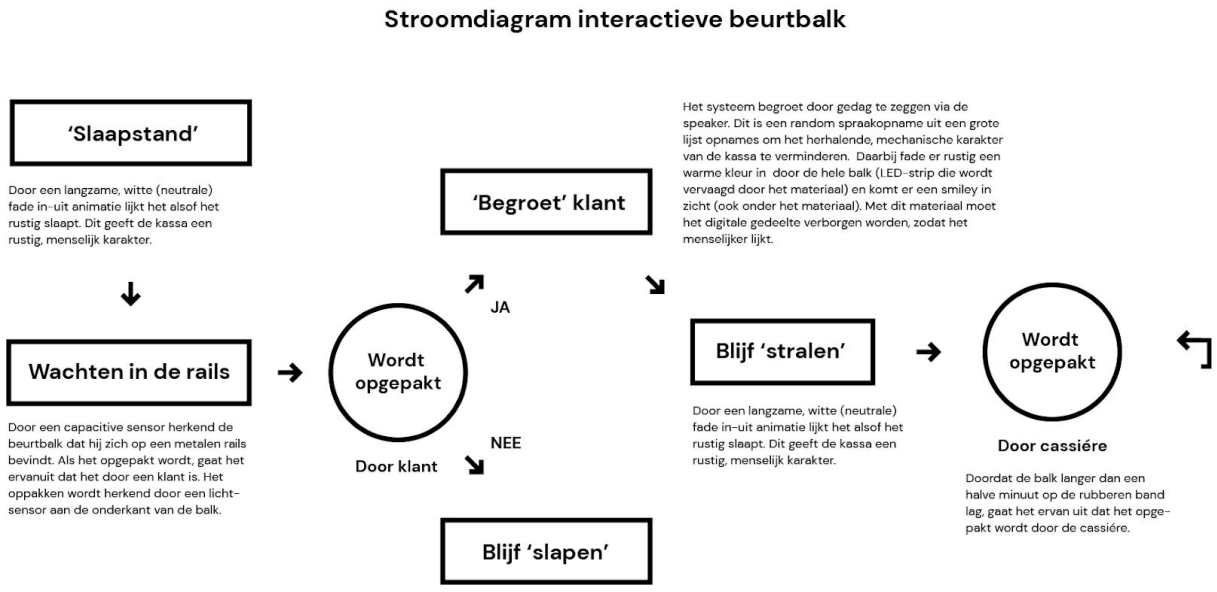
De rgbSleeping() functie fade in en uit zolang die aangeroepen wordt in de loop(). Als de RGB volledig ingefade is, wordt de variabel fadeIn op false gezet, zodat hij uitfade en visa versa. Er is een aanpassing in de waarde van de groene LED, omdat deze sterker was dan de andere twee.

```
//
// light sensor functions
//
void readLDR() {
  ldrValue = analogRead(ldrPin);
//  Serial.println(ldrValue);

  // if too light, it is considered pulled up
  if(ldrValue > 30 && !pulledUp) {
    // if too light, set timer for small delay, so it feels more natural
    if(pulledUpTimer == 0) { pulledUpTimer = millis(); }
//    Serial.println(pulledUpTimer);
    if(millis() - pulledUpTimer > 1000) {
      pulledUp = true;
      pulledUpTimer = 0;
      setSmiley();
    } // add little delay to smile
    if(sleeping) { sleeping = false; }

  }
  // if too dark for at least 3 seconds, fall asleep
  if (ldrValue <= 30 && pulledUp) {
    if(pulledDownTimer == 0) { pulledDownTimer = millis(); }
    if(millis() - pulledDownTimer > 3000) {
      pulledUp = false;
      sleeping = true;
      pulledDownTimer = 0;
      clearMatrix();
    }

  }
}
```

LDR functie

readLDR() leest de waarde van de LDR sensor. Als het te licht is, zet hij een korte timer voordat hij de smiley laat zien, zodat het natuurlijker aanvoelt. Helaas kan de matrix niet in-faden. Als de waarde te donker is, dan wacht hij 3000ms voordat hij gaat slapen en de matrix uitzet.

Stroomdiagram



De losse bestanden zijn te vinden op GitHub:

https://github.com/lsiewers/kernmodule-ii

Door Luuk Siewers

HKU Games & interactie, jaar 2