TEAM BURRITO

Smart Supermarket

Geert Mol
Matthijs Sluijk
Jordy van Hierden
Thomas Gort
Babet ten Hagen
Sarah Jansen
Carmen Pol

OUR CHALLENGE/PROBLEM

A big problem in first world countries is food waste. There have been several campaigns to tackle these problems, targeted at the people at home but also nationwide. A part of this problem lays with supermarkets, as they are essentially the source of our food. We decided we wanted to tackle the food waste produced by supermarkets and also indirectly the consumers food waste at home. Consumers say that 30% of their food waste is because of the food being past the expiry date(Dooren, C., & Mensink, F. (2018). *Consumer food waste*).

We first started brainstorming about food waste because our general interest was focussed on that subject. We then realized that there are already a lot of solutions out there that work a lot better than the ideas we came up with, so we moved more towards the source of the problem, the supermarket. This is where people buy their food and quite often buy too much/unnecessary food.

OUR SOLUTION

The solution to this problem is the smart supermarket shelf we made. With this shelf we can clearly communicate with the consumer but also the supermarket staff which products are about to expire. Each product will have a RFID sticker, which will be scanned when the product is placed on the shelf. The data from these products, including their expiry date is passed on into a database. Every product gets its own LED light according to the state the product is in. When a product starts to approach its expiry date it will get a different colour LED to notify the consumer and the staff. Products that are about to expire will get an automatic discount. Based upon these products recipes are created, that include as many products that are about to expire as possible. This will reduce the amount of products that are thrown out by the consumer because consumers buy products according to a recipe for today or tomorrow. Another outcome from this solution is that supermarkets will have to throw a lot less expired food away due to the discount that is on the products.

OUR METHODS AND TOOLS

The whole project is based around RFID-scanners. The ones we used are MFRC522 scanners, they are easy to use install and cheap to buy. We got them for around 25 cents. Up until 4 scanners they work reliably and consistent. If you exceed this amount you will start running into several problems, not all scanners will work consistent. We tackled this problem partially by soldering the wires and erasing the need of a breadboard. The breadboard did cause failures which are probably due to the fact that the construction of the bread creates a really weak capacitor which can disrupt signals.

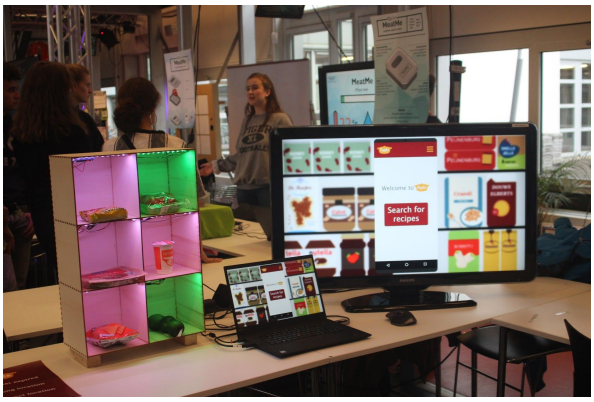
For connecting the RFID-scanners to the computer we decided to use an Arduino Uno. It has enough data pins to use 6 scanners at the same time. We used a second Arduino Uno to control the LED-strip to light up the state of the shelves in which products are placed. For doing calculations we used a processing sketch which was running on a laptop besides the hardware.

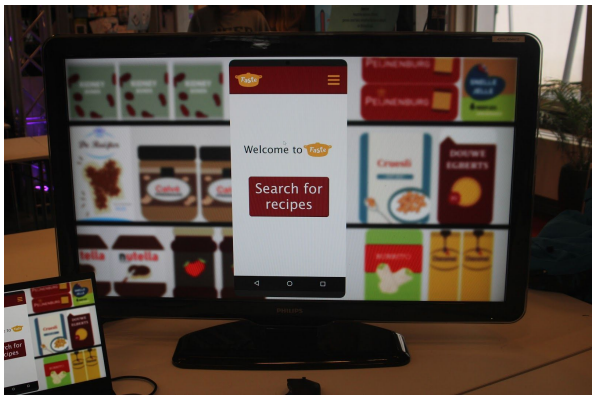
The hardware (Arduino Uno's/RFID-scanners/LEDs) were mounted around and inside of a shelf which was created by cutting out shapes of wood using a Laser Cutter. On the bottom side of a shelf was the RFID mounted using hot glue. The strip was mounted on the bottom of the shelf above it. This meant that each shelf has one RFID scanner and a part of a LED-strip. The Arduino's were connected to the backside of shelf.

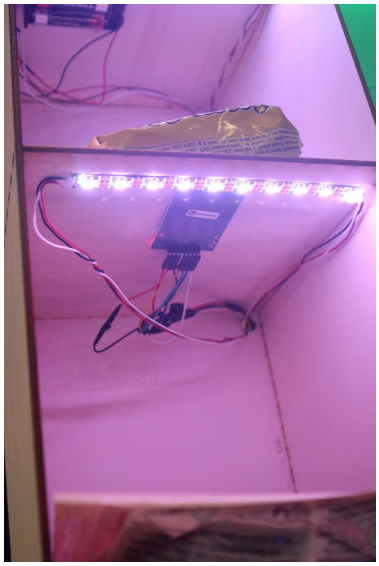
All the different components were soldered onto a protoboard. This protoboard was the center of all the hardware. It was hot glued to the middle of the backside of the shelf. Each of the individual cable sets were separated from one another using heat shrinking tubes and electrical tape.

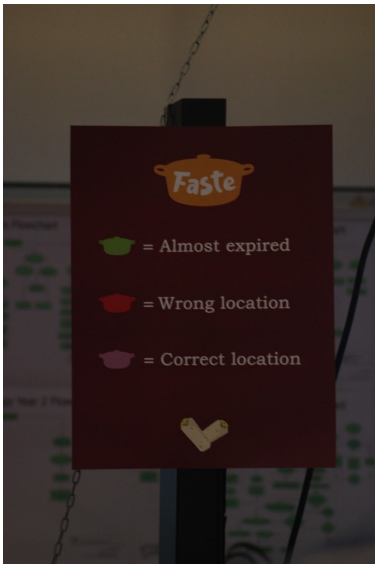
Important to know is the limitations of the MFRC522 scanners, they can only scan one thing at the same time and the range is limited (5 centimeters). This means that for further development other scanners should be tested and used to enhance the performance of the shelf. Furthermore would it be useful to solder everything down onto a custom PCB.

PHOTOS


Welcome to
Search for recipes


Welcome to
Search for recipes




Faste
= Almost expired
= Wrong location
= Correct location

CODE

Processing:

Main:

```
/*-------------------------------------------------------*/
/*----------------------FASTE----------------------------*/
/*-------------------------------------------------------*/
/*--Recipe selector based on almost expired products--*/
/*-------------------------------------------------------*/
/*----------------------MADE BY----------------------------*/
/*-------------------------------------------------------*/
/*-------------------- Geert Mol ----------------------*/
/*----------------- Matthijs Sluijk -----------------*/
/*---------------- Jordy van Hierden ----------------*/
/*------------------- Thomas Gort --------------------*/
/*----------------- Babet ten Hagen -----------------*/
/*------------------ Sarah Jansen --------------------*/
/*-------------------- Carmen Pol --------------------*/
/*-------------------------------------------------------*/

import processing.serial.*;
Serial scanner;
Serial lights;

UI ui;

// Amount of shelves
int productAmount = 6;

// Array of correct UIDs
String[] correct = {"7EEB647E", "4EDE647E", "3EEC647E", "04304D5AEE648",
"1EEB647E", "DEEB647E"};
// Array to store the current UIDs on the shelf
String[] shelf = new String[productAmount];
// Array to store the current LED value of the shelfs
int[] led = new int[productAmount];
```

```
// Array of recipes
ArrayList<Recipe> recipes;
// Array of products
ArrayList<Product> products;
// Array to keep track of which products are expired
ArrayList<String> expired;

// Integer to remember the recipe which has to most ingredients that are
about to expire


/*----------------*/
/*---Initialize---*/
/*----------------*/
void setup() {
  fullScreen();
  imageMode(CENTER);

  scanner = new Serial(this, "COM4", 9600);
  lights = new Serial(this, "COM3", 9600);

  ui = new UI();

  products = new ArrayList<Product>();
  recipes = new ArrayList<Recipe>();

  addProducts();
  addRecipes();
}


/*----------------*/
/*---Main Loop----*/
/*----------------*/
void draw() {
  // Empty the expired ArrayList every frame
  expired = new ArrayList<String>();

  // Read the serial port
  scanProducts();

  // For every product check if it is on the shelf and if so check the UID
```

```
  for (int i = 0; i < productAmount; i++) {
    println(shelf[i]);
    if (shelf[i] != null) {
      correctProduct(i); // Turn the purple led on if a product is on the
shelf
      checkUIDs(i);      // Check if it is the correct product
    }
    // If nothing is on the shelf turn of the led
    if (shelf[i] != null && shelf[i].length() == 0) {
      noProduct(i);
    }
  }

  // Make a variable called led info which contains which shelf it is
(tens) and which state the led is (unit)
  // Shelf 0 to 5 and led state 0 to 3 with:
  // 0 = no product
  // 1 = product on shelf
  // 2 = almost expired product
  // 3 = wrong product on shelf
  // So 32 means the third shelf and the product is almost expired.
  for (int i = 0; i < led.length; i++) {
    int ten = i * 10;
    int unit = led[i];
    int ledinfo = ten + unit;
    lights.write(ledinfo);
  }


  // Display the user interface
  ui.display();

  println(led);
}

void mousePressed() {
  if (searchButton(mouseX, mouseY) && ui.screen == 8) {
    ui.screen = selectRecipe();
  }
  if (backButton(mouseX, mouseY)) {
    ui.screen = 8;
  }
```

```
}
```

Functions:

```
void addProducts() {
  products.add(new Product("Noodles", 2021 , 1, 20));
  products.add(new Product("Canned tomatoes", 2020, 1, 29));
  products.add(new Product("Tacos", 2021, 5, 17));
  products.add(new Product("Creme fraiche", 2020, 1, 29));
  products.add(new Product("Nachos", 2021, 1, 16));
  products.add(new Product("Lime", 2020, 1, 17));
}

void addRecipes() {
  recipes.add(new Recipe("Mexican tortillas", tortillasIng));
  recipes.add(new Recipe("Mexican soup with lime and yellow rice",
soupIng));
  recipes.add(new Recipe("Mexican bean salad", beanSaladIng));
  recipes.add(new Recipe("Mexican Pizza", pizzaIng));
  recipes.add(new Recipe("Mexican oven dish", ovenIng));
  recipes.add(new Recipe("Mexican taco salad", tacoSaladIng));
  recipes.add(new Recipe("Mexican rice dish", riceIng));
  recipes.add(new Recipe("Mexican vegetarian lasagne", lasagneIng));
}

// Product states
void noProduct(int product) {
  led[product] = 0;
}
void correctProduct(int product) {
  led[product] = 1;
}
void expiredProduct(int product) {
  led[product] = 2;
}
void wrongProduct(int product) {
  led[product] = 3;
}

void checkUIDs(int product) {
  if (shelf[product].equals(correct[product])) {
    checkDate(product); // If it is the correct product check the date
  } else {
    wrongProduct(product);
```

```
  }
}

void checkDate(int product) {
  Product current = products.get(product);
  if (current.year == year() && current.month == month()) {
    if (current.day <= day() + 2) {
      expired.add(current.name);
      expiredProduct(product);
    }
  }
}

int selectRecipe() {
  int selectedRecipe = 0;
  int productsInRecipe = 0;

  for (int i = 0; i < recipes.size(); i++) {
    int current = 0;
    for (int j = 0; j < recipes.get(i).ingredients.length; j++) {
      for (int k = 0; k < expired.size(); k++) {

        if (expired.get(k) == recipes.get(i).ingredients[j]) {
          current++;
        }
      }
    }
    if (current > productsInRecipe) {
      productsInRecipe = current;
      selectedRecipe = i;
    }
    println(current);
  }
  println(recipes.get(selectedRecipe).name);

  return selectedRecipe;
}
```

Product:

```
class Product {

  String name;
  int year;
  int month;
  int day;

  Product(String n, int y, int m, int d) {
    name = n;
    year = y;
    month = m;
    day = d;
  }
}
```

Recipe:

```
String[] tortillasIng = {"Corn", "Canned tomatoes", "Creme fraiche",
"Cheese", "Zucchini", "Yellow bell pepper"};
String[] soupIng = {"Corn", "Canned tomatoes", "Creme fraiche", "Yellow
rice", "Broth beef"};
String[] beanSaladIng = {"Kidney beans", "Canned tomatoes", "Iceberg
lettuce", "Cheese", "Lime"};
String[] pizzaIng = {"Corn", "Creme fraiche", "Pizza base", "Red bell
pepper", "Minced beef"};
String[] ovenIng = {"Kidney beans", "Leek", "Eggs", "Ground beef",
"Cheese"};
String[] tacoSaladIng = {"Corn",  "Kidney beans", "Salsa sauce", "Chicken
fillet strips", "Iceberg lettuce", "Tacos", "Avocado"};
String[] riceIng = {"Corn",  "Kidney beans", "Creme fraiche", "Basmati
rice", "Chicken fillet", "Red bell pepper"};
String[] lasagneIng = {"Corn", "Kidney beans", "Canned tomatoes",
"Tortillas", "Red bell pepper", "Cheese"};

class Recipe {

 String name;
 String[] ingredients;

 Recipe(String n, String[] ing) {
   name = n;
   ingredients = ing;
 }
}
```

Scanner:

```
void scanProducts() {
  String read = "";
  char c = ' ';

  if (scanner.available() > 0) {
    read = scanner.readStringUntil('\n');
  }
  try {

    c = read.charAt(0);
    read = read.substring(1);
    read = trim(read);



    if (c == 'A') {
      shelf[0] = read;
    } else if (c == 'B') {
      shelf[1] = read;
    } else if (c == 'C') {
      shelf[2] = read;
    } else if (c == 'D') {
      shelf[3] = read;
    } else if (c == 'E') {
      shelf[4] = read;
    } else if (c == 'F') {
      shelf[5] = read;
    }

  }
  catch(Exception e) {
    //println("no valid data");
  }
}
```

UI:

```
class UI {
  int screen = 8;

  int recipeX = width/2;
  int recipeY = height/2-35;

  PImage home;
  PImage recipe0;
  PImage recipe1;
  PImage recipe2;
  PImage recipe3;
  PImage recipe4;
  PImage recipe5;
  PImage recipe6;
  PImage recipe7;

  UI() {
    home = loadImage("assets/home.jpg");
    recipe0 = loadImage("assets/recipe0.jpg");
    recipe1 = loadImage("assets/recipe1.jpg");
    recipe2 = loadImage("assets/recipe2.jpg");
    recipe3 = loadImage("assets/recipe3.jpg");
    recipe4 = loadImage("assets/recipe4.jpg");
    recipe5 = loadImage("assets/recipe5.jpg");
    recipe6 = loadImage("assets/recipe6.jpg");
    recipe7 = loadImage("assets/recipe7.jpg");
  }

  void display() {
    image(home, width/2, height/2);
    switch(screen) {
    case 0:
      image(recipe0, recipeX, recipeY);
      break;
    case 1:
      image(recipe1, recipeX, recipeY);
      break;
    case 2:
      image(recipe2, recipeX, recipeY);
```

```
      break;
    case 3:
      image(recipe3, recipeX, recipeY);
      break;
    case 4:
      image(recipe4, recipeX, recipeY);
      break;
    case 5:
      image(recipe5, recipeX, recipeY);
      break;
    case 6:
      image(recipe6, recipeX, recipeY);
      break;
    case 7:
      image(recipe7, recipeX, recipeY);
      break;
    }
  }
}

// Check if the mouse position is on the search button
boolean searchButton(float mx, float my) {
  if (mx > 785 && mx < 1135 && my > 515 && my < 695) {
    return true;
  } else {
    return false;
  }
}

// Check if the mousePosition is on the back button
boolean backButton(float mx, float my) {
  if (mx > 750 && mx < 880 && my > 970 && my < 1040) {
    return true;
  } else {
    return false;
  }
}
```

Arduino:

Scanner:

```
/*
  Typical pin layout used:
   -----------------------------------
               MFRC522      Arduino
               Reader/PCD   Uno/101
   Signal      Pin          Pin
   -----------------------------------
   RST/Reset   RST          9
   SPI SS      SDA(SS)      10
   SPI MOSI    MOSI         11 / ICSP-4
   SPI MISO    MISO         12 / ICSP-1
   SPI SCK     SCK          13 / ICSP-3
*/

#include <SPI.h>      // Serial Peripheral Interface
#include <MFRC522.h>  // Needed for using RFID module 13.56 MHz

const byte NR_OF_READERS = 6;             // Numbers of RFID readers
const byte RST_PIN = 9;                   // Reset pin (default 9)
const byte ssPins[] = {2, 3, 4, 5, 6, 7}; // SS pins of all the readers
MFRC522 mfrc522[NR_OF_READERS];           // Create MFRC522 instance.
String currentIDs[NR_OF_READERS];         // Array to keep track of all the
UIDs
/*
   Initialize.
*/
void setup() {

  Serial.begin(9600); // Initialize serial communications with the PC
  while (!Serial);    // Do nothing if no serial port is opened (added for
Arduinos based on ATMEGA32U4)

  SPI.begin();        // Initialize Serial Peripheral Interface

  for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {
    mfrc522[reader].PCD_Init(ssPins[reader], RST_PIN);  // Init each
MFRC522 card
```

```
    //    Check if the readers are recognized
    Serial.print(F("Reader "));
    Serial.print(reader);
    Serial.print(F(": "));
    mfrc522[reader].PCD_DumpVersionToSerial();
  }
}

/*
   Main loop.
*/
void loop() {
  boolean changedValue = false;

  for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {
    mfrc522[reader].PCD_Init();
    String readRFID = "";
    // Look for new cards

    if (mfrc522[reader].PICC_IsNewCardPresent() &&
mfrc522[reader].PICC_ReadCardSerial()) {
      readRFID = dump_byte_array(mfrc522[reader].uid.uidByte,
mfrc522[reader].uid.size, reader);

      // Halt PICC
      mfrc522[reader].PICC_HaltA();
      // Stop encryption on PCD
      mfrc522[reader].PCD_StopCrypto1();
    }

    if (readRFID != currentIDs[reader]) {
      changedValue = true;
      currentIDs[reader] = readRFID;
    }

  }
  if (changedValue) {
    for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {
      if (reader == 0) {
        Serial.print('A');
      }
      if (reader == 1) {
```

```
        Serial.print('B');
      }
      if (reader == 2) {
        Serial.print('C');
      }
      if (reader == 3) {
        Serial.print('D');
      }
      if (reader == 4) {
        Serial.print('E');
      }
      if (reader == 5) {
        Serial.print('F');
      }
      Serial.println(currentIDs[reader]);
    }
  }
  delay(500);
}


// Helper routine to return a byte array as hex values.

String dump_byte_array(byte * buffer, byte bufferSize, byte current) {
  String readRFID;
  for (byte j = 0; j < mfrc522[current].uid.size; j++)
  {
    readRFID.concat(String(mfrc522[current].uid.uidByte[j] < 0x10 ? "0" :
""));
    readRFID.concat(String(mfrc522[current].uid.uidByte[j], HEX));
  }
  readRFID.toUpperCase();
  return readRFID;
}
```

LED strip:

```
#include <FastLED.h>
#define LED_PIN     7
#define NUM_LEDS    60
CRGB leds[NUM_LEDS];

// Incoming data
int inc;

void setup() {
  Serial.begin(9600);
  FastLED.addLeds<WS2812, LED_PIN, GRB>(leds, NUM_LEDS);
}
void loop() {
  if (Serial.available() > 0) {
    // Read the incoming data
    inc = Serial.read();

    // Calculate which shelf (ten - 0 to 5)
    // Calculate the ledstate (unit - 0 to 3)
    int unit = inc % 10;
    int ten = inc - unit;

    change(ten, ten + 10, unit);
  }
  change(0, 30, 2);
  change(30, 60, 1);
  FastLED.show();
}

void change(int start, int finish, int mode) {
  for (int i = start; i < finish; i++) {
    if (mode == 0) {
      leds[i] = CRGB(0, 0, 0);
    } else if (mode == 1) {
      leds[i] = CRGB(201, 104, 212);
    } else if (mode == 2) {
      leds[i] = CRGB(3, 252, 15);
    } else if (mode == 3) {
      leds[i] = CRGB(255, 0, 0);
```

```
    }
  }
}
```