Fyxers

Paragraph 1:

A big problem we encountered in agriculture was the fact that farmers are not adapting food of their animals to their nutritional needs. Shifts in body temperature of the animal suggest a different need of food. For example, when the body temperature of the animal decreases more food is needed. The problem is that most of the time all animals get the same composition of food and the same amount of this. This way its nutrition isn't based on the individual needs but more on the overall need of food for an animal. This means that a cow that moves 3 times as much as another cow gets the same amount and composition of food. This isn't right. The nutrition of animal should be based on the needs of an animal. This way also a lot of food is wasted, just now food waste is such an important topic as the world population is increasing and food gets more scarce. Secondly food production is ways from optimal because each animal doesn't get optimal nutrition. Think about it, when a cow just gets some food fitting for that animal, that doesn't mean it is fitting for that cow, each of them have their on needs. When we fulfill those needs, the food that that cow produces will improve as well. This means that with food based on nutritional needs each individual animal will produce better and more products. This way we will need less animals for the same amount of produce, which is better for the planet as well. So it comes down to that first problem that farmers are not adapting food on the nutritional needs of their animals. When we solve the first problem, parts of the problems of food waste and optimal produce of an animal will be solved as well.

Paragraph 2:

Farmers are not adapting to the fact that because of the shifts in temperature animals have different nutritional needs (they need more/less food). This affects both produce as well as food waste. We chose this problem because of multiple reasons. Firstly this problem connects to a multiple of general problems, we really like that by solving this we contribute to different problems. When we solve this, we also solve parts of those. We can for instance increase food production levels so we need less animals for the same amount of produce, what is a good thing as we are in a nitrogen crisis. Than there is also a decrease in food waste we can achieve by only give the needed amount of food. Lastly because we think we can solve this problem and develop a prototype in the given time frame. Multiple studies have shown the correlation of nutritional needs and external temperature. The following study suggests a formula for which the alteration in nutrition can be calculated according to temperature, weight and nutritional value. Other studies state that malnutrition causes great decreases in produce and can even lead to diseases. This suggest that giving cattle too much food under the wrong circumstances, would result in less produce. Thus has a negative impact on a many fields, including environmental, economical, animal welfare and so on. It ties in with the much bespoken ongoing nitrogen crisis. By optimalized the resource

use, the nitrogen in animal manure and the amount of manure could potentiallybe minimalized. Our device entails a box which can be attached to the collar of the cow. It features a Temperature sensor accelerometer and a heart rate sensor Temperature sensor accelerometer and a heart rate sensor. The combined data of these sensor can give insight in the nutritional needs of the cow, and those can therefore be optimalized in order to maximise produce and minimalize the environmental footprint. We've build a user-friendly interface in order to make this proces as easy as possible for the farmers. Data is stored on an SD-card, as cows aren't usually in range of wifi. The data can be retrieved later. Not only is the sensor useful for optimilizing nutrition, but it could also be used to detect diseases early on. For the device to work optimally, a fair amount of research would be required into the behaviour and needs of a cow, which was a little beyond the scope of this project, but our preliminary research could easily be extended. We believe that this product could benefit many causes. Especially with the upcoming nitrogen crisis, this device could play a key role in reducing the nitrogen in manure, which in turn would enable farmers to maintain their income. We believe that upon further research this could become a successful product.

(https://www.ncbi.nlm.nih.gov/books/NBK232316/)

Paragraph 3:

Parts:

- Two Arduino Uno's
- 3D printed casing
- Powerbank
- Collar for the cow to wear the casing
- Rf- module for wireless data transmission
- SD-module

Sensors:

- Heart rate sensor
- Temperature sensor
- Accelerometer (X,Y,Z-acceleration)

We used a casing to protect the sensors and Arduino. The sensors that we used are an accelerometer, a temperature sensor and a heart rate sensor. With the accelerometer we wanted to measure the displacement, so the movement of the cow, and determine how active the cow is. The temperature sensor, we wanted to put it against the cow to measure its temperature. This way we could determine if the cow has a normal temperature, or if its having a fever or cold. The heart rate sensor was to monitor the heart rate of the cow. We

thought that using a collar to put the casing around the neck would be the best way, as it is usual for cows to have a collar. For the Arduino, we used one of our arduino's and ordered another one. One of the arduino's was destined for the collar and another was for our computer to gather the data on our computer. We took a powerbank of one of our group members so that the arduino and sensors could be powered while being around the cows neck.To get the live data from the cow, we wanted to make use of wireless data transmission. For that we got the RF 315/433 Mhz Transmitter Receiver module. We also got a SD module that would store the values that the sensor reads. Our idea was to store the data to an SD card and let the RF transmitter send the data to the receiver that is on the arduino connected to the computer. This ensures that the data is sent with less data loss. Originally we also intended to include a way to process all the data received from the collar with information about proper diet for a cow should eat to make a recommendation for the farmer on how much each cow has to eat to grow as healthily as possible. If we had enough time, we also wanted to include a food dispenser that knew how much food to give to each cow. We figured that the cow collar was the most important part of our project, so that was our main priority. We made the cow collar with the sensors, but we weren't able to make the food dispenser. For the software we created an interface with the Processing IDE in which you could see the values of the sensors. With the Arduino IDE we programmed the arduino to read the values from the sensors and we made recalculated the incoming values to make them usable. For example, changing the analog values that are being read from the temperature sensor and scaling it to degree Celcius. We tested our prototype on a cow. Because the cow was moving around a lot, we couldn't put on the collar so we just held it against the cow to take the measurements. We put the casing to the collar using duct tape. The casing has two holes that are for the heart rate sensor and the temperature sensor. These two we put directly against the cow to give us an accurate as possible measurement from both sensors. We tried to use the rf-22 module, but we couldn't make it to work together with the SD-module. The VirtualWire library and the SD module library both needed the same pin to operate, so we couldn't make the wireless communication work. The SD-module and the rf-module work independently.

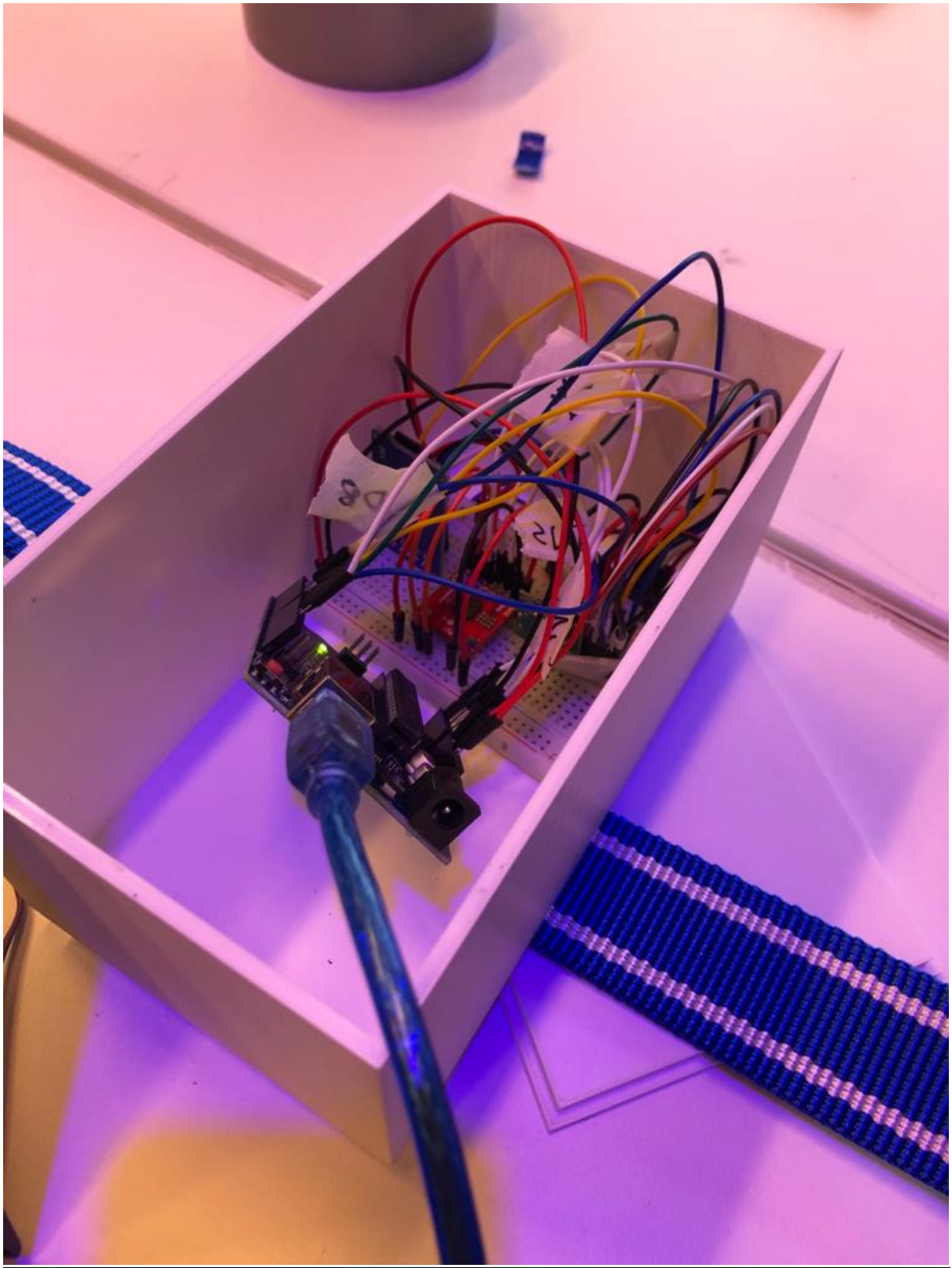
Paragraph 4:

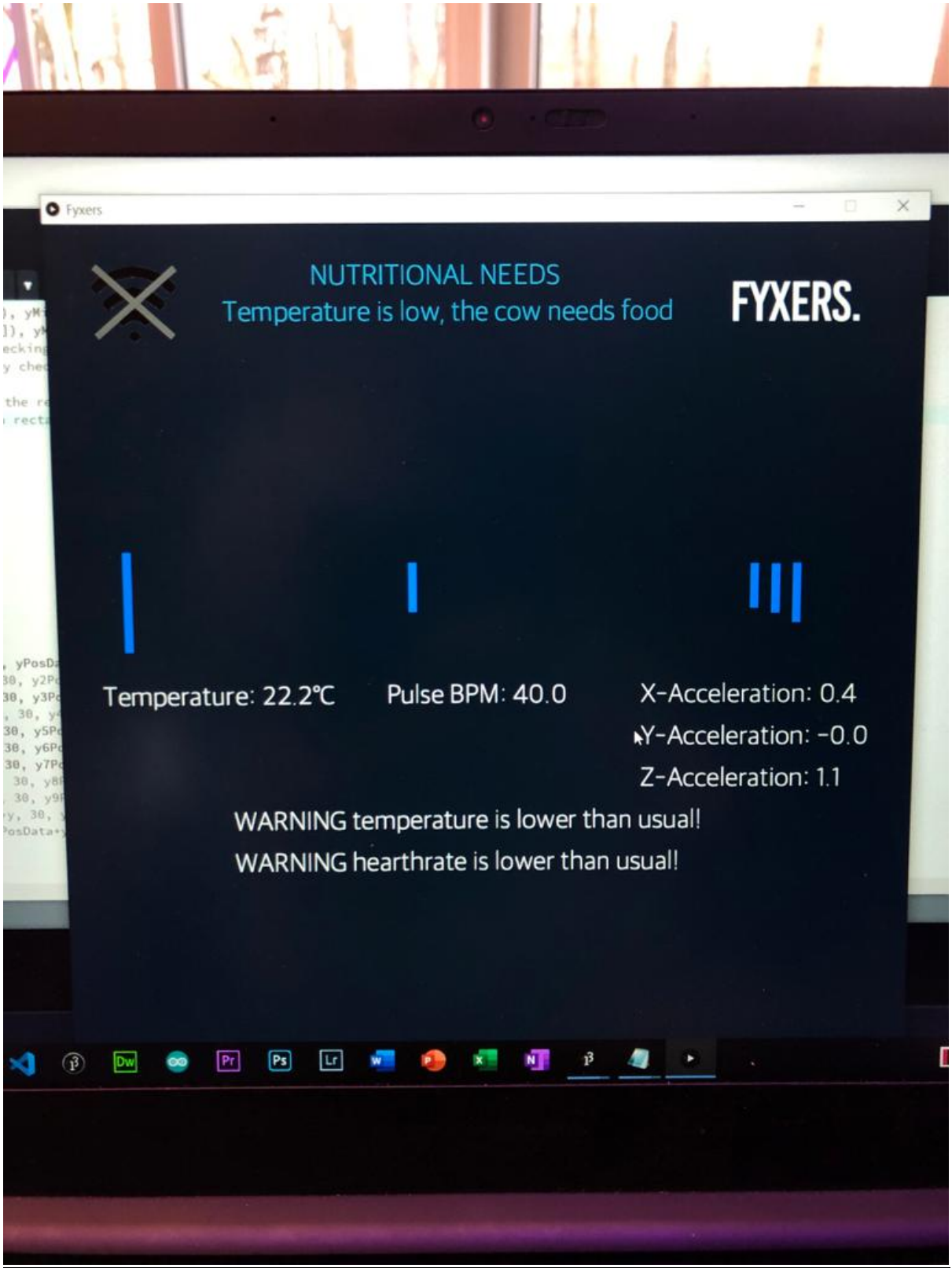
I think it is clear what the prototype is about. Our team tested this prototype on a living animal (that was the intention from the start). While testing the cow couldn't move much due to the collar not really being attached to the cow. It couldn't be placed around the cow's neck. The prototype was held against the neck of the cow. Unfortunately the SD card wasn't able to measure the wanted data. This was due to the SD card not being installed correctly in the prototype. Only at home could our team see the data and determine what was measured. One reason could be that the cow moved so much, the SD card misplaced. Another reason could be that the SD card was not installed correctly in the first place, so would fail from the start. There have been observations during the testing. The cows move very much and the collar wouldn't be on the same place. Another observation is the size of the prototype. The prototype was too big for the cow. This also resulted in the collar not fitting on the cow. An optimization of the casing could greatly increase the efficiency since the current prototype uses sharp edges and a tough, plastic, 3D-printed casing. This could

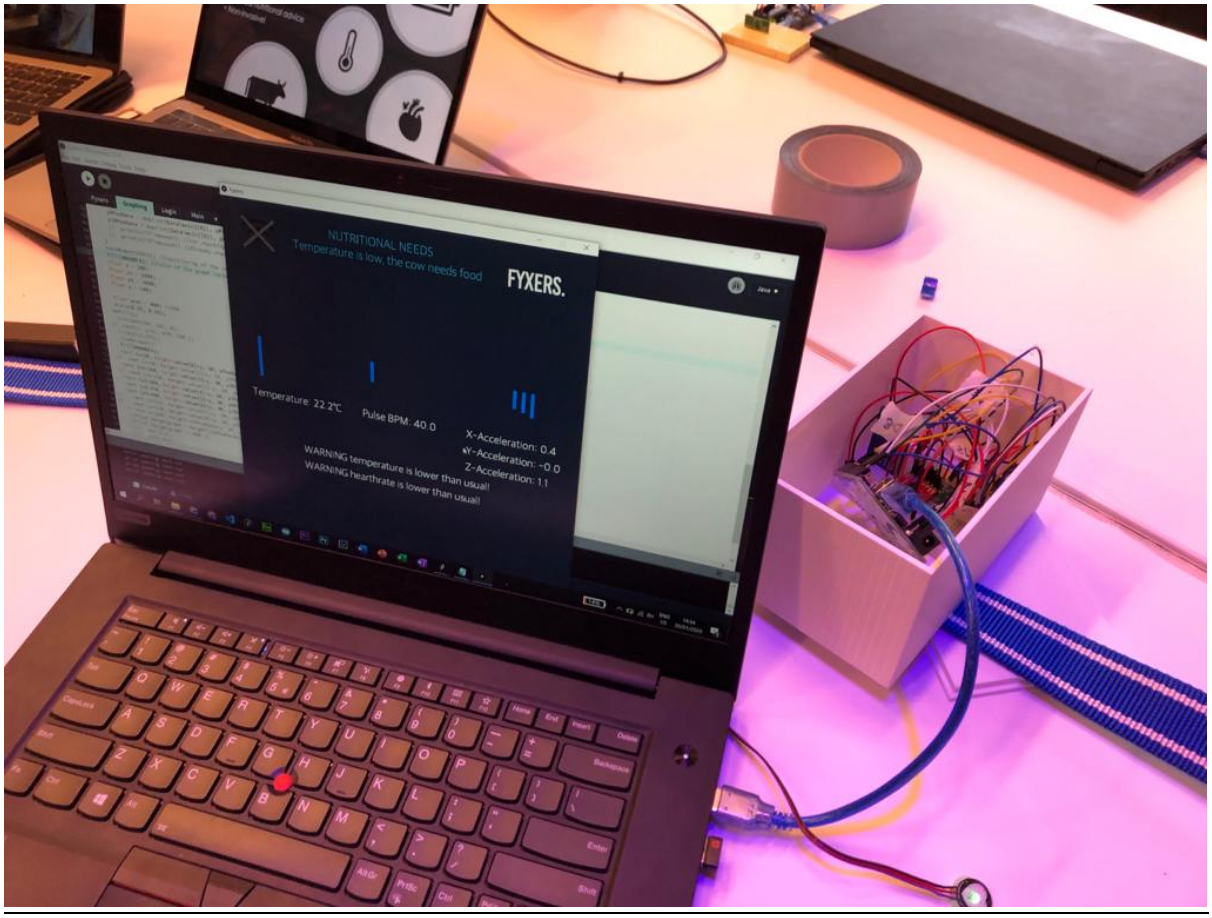
be replaced with a smoother and smaller measuring ‘box’. As of now, there are results lacking and there to be learned. There is a base which could be optimised to become an effective and better product.

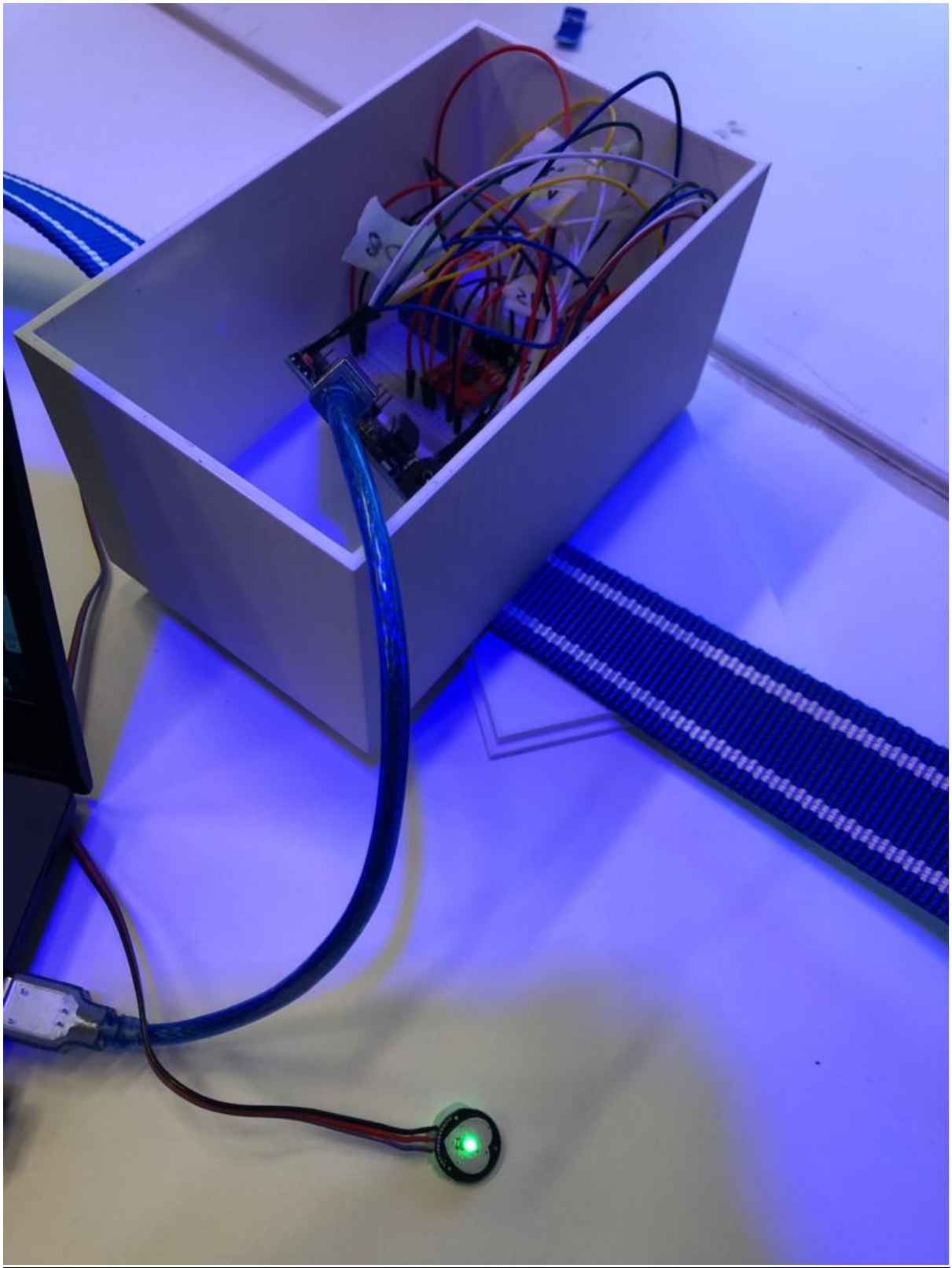
Our Video: https://www.youtube.com/watch?v=C5xqbVrynkQ

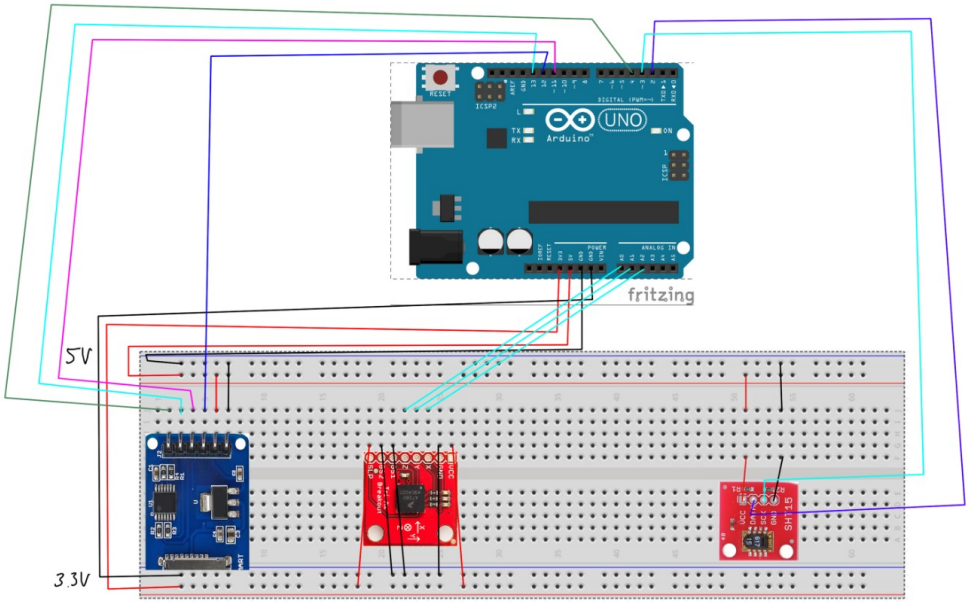
TOPIC 6 (Pictures)




Fyxers
NUTRITIONAL NEEDS
Temperature is low, the cow needs food
FYXERS.
Temperature: 22.2℃
Pulse BPM: 40.0
X-Acceleration: 0.4
Y-Acceleration: -0.0
Z-Acceleration: 1.1
WARNING temperature is lower than usual!
WARNING hearthrate is lower than usual!


NUTRITIONAL NEEDS
Temperature is low, the cow needs food
FYXERS.
Temperature: 22.2°C
Pulse BPM: 40.0
X-Acceleration: 0.4
Y-Acceleration: -0.0
Z-Acceleration: 1.1
WARNING temperature is lower than usual!
WARNING hearthrate is lower than usual!




5V
3.3V
fritzing

TOPIC 7

Processing Code:

This class is the Main Class where everything is put together

```
// This version can receive data from sensor connected to arduino

Main mainA;
Graphing graphingA;
import processing.serial.*;
Serial port;

   float value[] = new float[7];

void setup() {
  size(1000, 1000);
  mainA = new Main();//
  graphingA = new Graphing();
 port = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  background(22,44,65);

  mainA.display();
  graphingA.display();
  loginMenu();
  tableOfContents();
}

void keyPressed() {
  loginkey();
}
```

This class is the graphing class, where the graph code is made and a txt file is being read out and graph this graph is being split up in between each category.

```
class Graphing {

  String DataTwo[][]; //Going through it ith for loop also converting into int data type
  String loadData[]; //Loading file inside
  //Variables for saving each value
  float yPosData;
  float y2PosData;
  float y3PosData;
  float y4PosData;
  float y5PosData;
  float y6PosData;
  float y7PosData;
  float y8PosData;
  float y9PosData;
  float y10PosData;


  int fieldNum = 2;

  //Window size has to be mapped
  int xMin = 0;
  int xMax = 600;
  int yMin = 0;
  int yMax = 600;

  Graphing () {
    // smooth();
    loadData = loadStrings("data/test.txt"); //loading string  //Also text file path
    DataTwo = new String[loadData.length][fieldNum]; //loading string Array into different string array with two rows for string arrays
    for (int i=0; i < loadData.length; i++) { //For loop running through the length of the text file
      DataTwo[i] = loadData[i].split(","); //String is being split up into different variable
    }
  }
```

```
  }

  void display() {
    //background(#80C137);

    for (int n=0; n < loadData.length; n++) { //For loop running through the array while converting it into int data type
      yPosData = map(int(DataTwo[n][0]), xMin, xMax, 0, width);  //variable from int data type mapped in between canvas size
      y2PosData = map(int(DataTwo[n][1]), yMin, yMax, 0, height);
      y3PosData = map(int(DataTwo[n][2]), yMin, yMax, 0, height);
      y4PosData = map(int(DataTwo[n][3]), yMin, yMax, 0, height);
      y5PosData = map(int(DataTwo[n][4]), yMin, yMax, 0, height);
      y6PosData = map(int(DataTwo[n][5]), yMin, yMax, 0, height);
      y7PosData = map(int(DataTwo[n][6]), yMin, yMax, 0, height);
      y8PosData = map(int(DataTwo[n][7]), yMin, yMax, 0, height);
      y9PosData = map(int(DataTwo[n][8]), yMin, yMax, 0, height);
      y10PosData = map(int(DataTwo[n][9]), yMin, yMax, 0, height);
    }
    rectMode(CORNER); //Positioning of the rectangle mode
    fill(#008BFA); //Color of the graph rectangle
    float x = 200;
    float x2 = 1000;
    float x3 = 2000;
    float y = 100;

    float yrec = 600; //600
    scale(0.35, 0.35);
   noFill();
    fill(#008BFA);
    rect (x+20, height-value[0]+y, 30, yPosData); //rectangle drawn not in for loop
    rect (x2+160, height-value[2]+y, 30, y3PosData); //30 is the width of the graph
    rect (x3+300, height-value[4]+y, 30, y5PosData);
    rect (x3+370, height-value[5]+y, 30, y6PosData);
    rect (x3+440, height-value[6]+y, 30, y7PosData);
  }
}
```

This is the login class which we got from the internet here is the link: https://forum.processing.org/two/discussion/25962/login-screen this class gives us the cabality to log into the program (safty) with a username and password but we built in a tab function which will let you skip the login for demo porpuses.

```

//the login code is all copied from https://forum.processing.org/two/discussion/25962/login-screen

//declare global variables
boolean slides[] = {true, false, false, false, false, false, false, false, false, false, false};
String topText = "";
String bottomText = "";
String wasText = "";
String bottomWasText = "";
boolean username = true;


boolean appear = true;

//Function(Procedure) controls Login Menu.
void loginMenu() {

  if(appear){
  if (slides[0] == true) {
    background(255);
    rectMode(CENTER);
    fill(255);
    rect(300, 200, 300, 100);
    rect(300, 400, 300, 100);
    fill(0);
    textSize(20);
    text(topText, 200, 215);
    text(bottomText, 200, 375);
    text(wasText, 200, 215);

  }
  }
}
```

```

//allows keyboard input for various things
void loginkey() {
  if ((keyCode == TAB)){
    appear = false;
  }

  if (slides[0] == true) {

    if ((keyCode == BACKSPACE && topText.length() > 0))
    {
      if (username == true) {
        topText = topText.substring(0, topText.length()-1);
      }
    }
    if ((keyCode == BACKSPACE && bottomText.length() > 0) ) {
      if (username == false) {
        bottomText = bottomText.substring(0, bottomText.length()-1);
      }
    } else if (keyCode == ENTER) {

      if (username==true) {
        username = false;
        wasText = topText;
        return;  //Now ready to start collecting password
      }
      bottomWasText = bottomText;
      if (wasText.equalsIgnoreCase("usr") && bottomWasText.equalsIgnoreCase("pwd")) {
        slides[0] = false;
        slides[1] = true;
      } else println("failed  ", wasText, bottomWasText);
    } else if ((key >= 32 && key <= 126))//add char to string
    {
      if (username == true) {
        topText = topText + key;
```

```
    } else if (keyCode == ENTER) {

      if (username==true) {
        username = false;
        wasText = topText;
        return;  //Now ready to start collecting password
      }
      bottomWasText = bottomText;
      if (wasText.equalsIgnoreCase("usr") && bottomWasText.equalsIgnoreCase("pwd")) {
        slides[0] = false;
        slides[1] = true;
      } else println("failed  ", wasText, bottomWasText);
    } else if ((key >= 32 && key <= 126))//add char to string
    {
      if (username == true) {
        topText = topText + key;
      } else {
        bottomText = bottomText +key;
      }
    }
  }

  println(topText+"  ++ "+bottomText);
}

void tableOfContents() {
  if (slides[1] == true) {
    background(255);
    fill(0);
    rect(250, 250, 30, 30);
  }
}
```

The Main Class is about the basic objects which create the interface. It also includes the code for getting the data over from the Arduino into processing. We got the Graphwriter for getting the data from Arduino to processing from the website http://wiki.edwindertien.nl/doku.php?id=education:physicalcomputing:04_making_things_mov

```
class Main {

  PFont font;

int Cf;
int B;


  int Xpostext = 0;
  int Ypostext = 0;
  PFont mono;
  PImage WIFIzero;
  PImage WIFIone;
  PImage WIFItwo;
  PImage WIFIthree;
  PImage WIFIfour;
  PImage logo;

  boolean Wifizerob = true;
  boolean Wifioneb = false;
  boolean Wifitwob = false;
  boolean Wifithreeb = false;
  boolean Wififourb = false;

  char header[] = {'H', 'T', 'D', 'B', 'X', 'Y', 'Z'};//depends on how many sensor we have
  int NEWLINE = 10;

  int diffValue[] = new int[2];
  String buff = "";
  float[] x = new float[20];
  float[] y = new float[20];
  float segLength = 9;

  PrintWriter dataFile;
```

```
PrintWriter dataFile;

Main () {
  font = loadFont("AppleSDGothicNeo-UltraLight-48.vlw");
  textFont(font);
  dataFile = createWriter("cowData.txt");
}

void display() {

  while (port.available() > 0) {
    serialEvent(port.read()); // read data
  }
  WIFIzero = loadImage("images/WIFIzero.png");
  WIFIone = loadImage("images/WIFIone.png");
  WIFItwo = loadImage("images/WIFItwo.png");
  WIFIthree = loadImage("images/WIFIthree.png");
  WIFIfour = loadImage("images/WIFIfour.png");
  logo = loadImage("data/FYXERSlogo.png");


  if (Wifizerob == true) {
    image(WIFIzero, 40, 35);
//   print("zero");
  }
  if (Wifioneb == true) {
    image(WIFIone, 40, 35);
   // print("one");
  }
  if (Wifitwob == true) {
    image(WIFItwo, 40, 35);
//  print("two");
  }
  if (Wifithreeb == true) {
    image(WIFIthree, 40, 35);
```

```
Pyxers | Graphing | Login | Main
54      image(WIFItwo, 40, 35);
55    //  print("two");
56    }
57    if (Wifithreeb == true) {
58      image(WIFIthree, 40, 35);
59     // print("three");
70    }
71    if (Wififourb == true) {
72      image(WIFIfour, 40, 35);
73    //  print("four");
74    }
75
76    image(logo, 750, 10);
77
78       fill(71, 169, 201);
79    textSize(32);
80    text("NUTRITIONAL NEEDS", 300, 70);
81    if (value[1]<36) {
82      text("Temperature is low, the cow needs food", 200, 110);
83    }
84    if (value[1]>40) {
85      text("Temperature is high, the cow needs water and fat added to its diet ", 200, 110);
86    }
87
88    textSize(32);
89    fill(255);
90    text("Temperature: " +value[1] +"°C", 50, 550);
91    // text("Humidity: " +value[0]+"%", 50, 300);
92    // text("Dew Point: " +value[2]+"%", 50, 400);
93    text("Pulse BPM: " +value[3]*10, 380, 550);
94    text("X-Acceleration: " +value[4], 680, 550);
95    text("Y-Acceleration: " +value[5], 680, 600);
96    text("Z-Acceleration: " +value[6], 680, 650);
97
98    //text("SD Card Storage:", 50, 400);
```

```
    //text("SD Card Storage:", 50, 400);
    //text("WIFI Signal:", 50, 500);
    //text("Hearth Rate:", 50, 600);

    if (value[1]<36) {
     text("WARNING temperature is lower than usual!", width/5, 700);
    }
    if (value[1]>40) {
      text("WARNING temperature is higher than usual!", width/5, 700);
    }
    if (value[3]*10<50) {
      text("WARNING hearthrate is lower than usual!", width/5, 750);
    }
    if (value[3]*10>80) {
      text("WARNING hearthrate is higher than usual!", width/5, 750);
    }
  }

  void serialEvent(int serial)
  {
    try {    // try-catch because of transmission errors

      dataFile.println(char(serial));
      if (serial != NEWLINE) {
        buff += char(serial);
      } else {
        println(buff);
        // The first character tells us which axis this value is for
        char c = buff.charAt(0);
        // Remove it from the string
        buff = buff.substring(1);
        // Discard the carriage return at the end of the buffer
        buff = buff.substring(0, buff.length()-1);
        // Parse the String into an integer
```

```
  void serialEvent(int serial)
  {
    try {    // try-catch because of transmission errors

      dataFile.println(char(serial));
      if (serial != NEWLINE) {
        buff += char(serial);
      } else {
        println(buff);
        // The first character tells us which axis this value is for
        char c = buff.charAt(0);
        // Remove it from the string
        buff = buff.substring(1);
        // Discard the carriage return at the end of the buffer
        buff = buff.substring(0, buff.length()-1);
        // Parse the String into an integer
        for (int z=0; z<7; z++) {
          if (c == header[z]) {
            value[z] = Float.parseFloat(buff);

          }
        }
        buff = "";          // Clear the value of "buff"
      }
    }
    catch(Exception e) {
      println("no valid data");
    }
    print();
  }
}
```

Arduino Code:

```
/*
  SD card read/write

  This example shows how to read and write data to and from an SD card file
  The circuit:
   SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS - pin 4 (for MKRZero SD: SDCARD_SS_PIN)

  created   Nov 2010
  by David A. Mellis
  modified 9 Apr 2012
  by Tom Igoe

  This example code is in the public domain.

*/


/* Example sketch for Sensirion SHT15 humidity - temperature sensor
  Written by cactus.io, and requires the cactus_io_SHT15 library. Public domain
  This sketch will work with the Sparkfun SHT15 Breakout board.
  For hookup details using this sensor then visit
  http://cactus.io/hookups/sensors/temperature-humidity/sht15/hookup-arduino-to-sensirion-SHT15-temp-humidity-sensor
*/

#include <SPI.h>
#include <SD.h>

#include "cactus_io_SHT15.h"

File myFile;

//Arduino Uno, Accelerometer

//Accelerometer pins (analog)

//PULSESENSOR

#define USE_ARDUINO_INTERRUPTS true
#include <PulseSensorPlayground.h>

const int PulseWire = 4;
int Threshold = 400;
PulseSensorPlayground pulseSensor;


//TEMPERATURE & HUMIDITY

SHT15 sht = SHT15(2, 3);


//ACCELEROMETER

const int pinx = 1;
const int piny = 2;
const int pinz = 3;


//variables to store accelerometer output
int  valx = 0;
int  valy = 0;
```

```
//variables to store accelerometer output
int  valx = 0;
int  valy = 0;
int  valz = 0;

int calib = 0;
int position = 0;

int count = 0;



//setup

const float RESOLUTION = 800; // Resolution 1.5g -> 800mV/g
const float VOLTAGE = 3.3;
const float ZOUT_1G = 850;   // mv Voltage @ 1G
const int N  = 50;

// Connect the X,Y and Z pin to A0,A1 and A2 respectively
const int xaxis = 0;
const int yaxis = 1;
const int zaxis = 2;

float zeroX, zeroY, zeroZ;
int x, y, z;
float aax, aay, aaz;

// Acc Timers
//unsigned long accTimer;
//unsigned long lastAccTimer;

byte mode;

#define DEBUG

float zeroPoint(int axis)
{
  float acc = 0;
  for (int j = 0; j < N; j++)
  {
    acc = acc + (((float) analogRead(axis) * 5000) / 1023.0);
    delay(20);
  }
  return acc / N;
}



void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);

  Serial.print("Initializing SD card...");

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done.");

  pinMode(xaxis, INPUT);
```

```
void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);

  Serial.print("Initializing SD card...");

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done.");

  pinMode(xaxis, INPUT);
  pinMode(yaxis, INPUT);
  pinMode(zaxis, INPUT);

  zeroX =  zeroPoint(xaxis);
  zeroY =  zeroPoint(yaxis);
  zeroZ =  zeroPoint(zaxis);
  zeroZ = zeroZ - ZOUT_1G;

  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  myFile = SD.open("test.txt", FILE_WRITE);

  // if the file opened okay, write to it:

  //PULSESENSOR
  pulseSensor.analogInput(PulseWire);
  pulseSensor.setThreshold(Threshold);

  if (pulseSensor.begin()) {
    Serial.println("We created a pulseSensor Object !");  //This prints one time at Arduino power-up,  or on Arduino reset.
  }



}

void loop() {
  // nothing happens after setup



  if (myFile) {
    for (int i = 0; i < 40; i++) { //later change the 10 to 86400 (seconds in a day)
      Serial.println("Writing to test.txt...");
      //myFile.println("testing 1, 2, 3.");
      tempRoutine();
      int myBPM = pulseSensor.getBeatsPerMinute();   // Calls function on our pulseSensor object that returns BPM as an "int".

      if (pulseSensor.sawStartOfBeat()) {            // Constantly test to see if "a beat happened".

        myFile.print("B");                        // Print phrase "BPM: "
        myFile.print(myBPM);                        // Print the value inside of myBPM.
        myFile.print("\n");
      }
      delay(20);
      accRoutine();
    }
    // close the file:
    myFile.close();
    Serial.print("\n");
    Serial.println("done.");
```

```
      accRoutine();
    }
    // close the file:
    myFile.close();
    Serial.print("\n");
    Serial.println("done.");
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }

  delay(1000);

  // re-open the file for reading:
  myFile = SD.open("test.txt");
  if (myFile) {
    Serial.println("test.txt:");

    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      delay(100);
    }
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }

  SD.remove("test.txt");
}

void tempRoutine() {
  sht.readSensor();

  myFile.print("H"); myFile.print(sht.getHumidity()); myFile.print("\n");           //Humidity in percent
  myFile.print("T"); myFile.print(sht.getTemperature_C()); myFile.print("\n");      //Temperature in *C
  myFile.print("D"); myFile.print(sht.getDewPoint()); myFile.print("\n");          //Dew Point in *C
}


void accRoutine()
{
  x = analogRead(xaxis);
  y = analogRead(yaxis);
  z = analogRead(zaxis);

  aax = (((x * 5000.0) / 1023.0) - zeroX) / RESOLUTION;
  aay = (((y * 5000.0) / 1023.0) - zeroY) / RESOLUTION;
  aaz = (((z * 5000.0) / 1023.0) - zeroZ) / RESOLUTION;

  // computes sample time
  //accTimer = millis() - lastAccTimer;
  // updates last reading timer
  //lastAccTimer = millis();
```

```
  delay(1000);

  // re-open the file for reading:
  myFile = SD.open("test.txt");
  if (myFile) {
    Serial.println("test.txt:");

    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      Serial.write(myFile.read());
      delay(100);
    }
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }

  SD.remove("test.txt");
}

void tempRoutine() {
  sht.readSensor();

  myFile.print("H"); myFile.print(sht.getHumidity()); myFile.print("\n");           //Humidity in percent
  myFile.print("T"); myFile.print(sht.getTemperature_C()); myFile.print("\n");      //Temperature in *C
  myFile.print("D"); myFile.print(sht.getDewPoint()); myFile.print("\n");          //Dew Point in *C
}


void accRoutine()
{
  x = analogRead(xaxis);
  y = analogRead(yaxis);
  z = analogRead(zaxis);

  aax = (((x * 5000.0) / 1023.0) - zeroX) / RESOLUTION;
  aay = (((y * 5000.0) / 1023.0) - zeroY) / RESOLUTION;
  aaz = (((z * 5000.0) / 1023.0) - zeroZ) / RESOLUTION;

  // computes sample time
  //accTimer = millis() - lastAccTimer;
  // updates last reading timer
  //lastAccTimer = millis();

  myFile.print("X"); myFile.print(aax); myFile.print("\n");

  myFile.print("Y"); myFile.print(aay); myFile.print("\n");

  myFile.print("Z"); myFile.print(aaz); myFile.print("\n\n");


  //Serial.print(" @ ");
  //Serial.print(accTimer);
  //Serial.print(" ms/sample");
  //Serial.println();
}
```