1. Short description of your challenge/problem (500 words)

"Nowadays there are a lot of problems concerning agriculture and the environment. These problems need to be solved as soon as possible to sustain a healthy environment and healthy urban areas for the world population."

One of the most urgent problems in our opinion was the lack of space. The world population is growing and this means that there is more food needed to provide everyone with the sufficient amount. There is already a lot of space used for agriculture nowadays, and this will only increase if there will be no solution in the near future. If we keep destroying woods and different parts of nature, there will be a big negative impact on our life. Wildlife can disappear, biodiversity will decrease, etc. It was our challenge to find a solution for this. First we wanted to focus on the whole problem about the lack of space. We wanted to make something that actually had a huge impact regarding this problem. After some brainstorming sessions we discovered we do not have the knowledge and skills (yet) to get close to this. So, instead of focusing on big, industrial machinery, we decided to focus on a smaller, consumer usable object. This is how we came up with the idea of a smart vertical garden, which will be explained in chapter 2.

You may think it is a bit ambitious to solve such a huge problem with only a small vertical garden. If we take a look at the space that is needed for agriculture, compared to the space which we create with our idea, we will have to sell millions of products to even have a noticeable impact. Vertical gardens are pretty new to the market, and became popular not that long ago. If we went for the bigger, industrial looking gardens, we expect there will not be many people interested in it because there is not yet enough research done about the benefits. Investors will not invest quickly in our product, if it is not known (at all). By making our product consumer usable, we want to spread awareness of the lack of space in agriculture. By making it a small scale object, people are willing to spend money on it so they can have their own little garden. If more and more people are willing to buy a garden, the idea of a smart vertical garden will be spread and the urgency of the problem will become clear to more people. If the problem is well known, big investors may become interested and want to invest in bigger, industry looking vertical gardens. These gardens will actually have a way bigger impact, and maybe solve the problem regarding the lack of space.

2. Short description of your solution (500 words)

Our solution was a smart vertical garden. The smart vertical garden is fully automatic, it rotates toward the sun and it waters itself. It checks every fifteen minutes the position of itself and checks it with the position of the sun. The smart vertical garden has a database of what the position of the sun is of every fifteen minutes. The garden has also a humidity sensor, which is placed in the ground of the plants. Based on the information that the sensor gets, the water system decides if the plant needs more water or not.

There is also an app for everyone who has a vertical garden who can communicate with each other. The app consists of three parts, one art for everyone os the world global, one part for you and your friends, local, and last an information part.

The information part: you can find information about the currents plants you have in your garden. The information part will give you information about the characteristics of the plants, and the status/level of the plant. The status/level of the plants contain humidity, water level, progress status, vermin status, sun position, etc.

The part for you and your friends: this part looks a bit like Facebook, but for plants. You can share the status of your plants, you can post pictures and like each other's pictures and comments. You can share your findings about the plants with each other and you can ask questions.
The global part of the app: this part of the app is more interesting for bigger cities. You can share your resources with people in your area. If you have planted a lot of cucumbers and someone else has planted a lot of tomatoes, for example, you can trade with each other. This will also be a bit of a solution for food waste. You can put in the app what you have too much of. And you can look for what you need.

The smart vertical garden is vertical, it will make sure that the plants get more sun than normal, when they are set horizontal in the garden. Our garden is set 30 degrees more vertical. The plants should grow this way quicker than normal. It will almost never get in the shadow, only if there are clouds that day, or if someone sets the garden nearby a tree.

The smart vertical garden will not take much space, it is a small product and everyone, even if you live in an apartment/flat, can use the product. More food will be produced this way and people will be more food aware. They will get more knowledge about what they produce.

3. Short description of method and tools used (software, hardware) (1,000 words)
We separated this paragraph into two parts. In each part we will discuss the software and hardware part of that subject. In the first part we will actually talk about the rotating mechanism of the Smart Vertical Garden and in the second part we will discuss the features we gave to the vertical garden itself for analyzing the state of the plant.

(*software*) In the first place our main goal of the project was to develop a mechanism which enables the possibility to get the optimal sunlight experience for a plant. We created an own made database which contained information of the angle the sun made relative to the surface of the earth. Thereby we also created a file which contained the actual time the sun went down and up again. This allowed us to know how long to wait each time.
Next we created a simulation of a top view of the vertical garden, which actually was able to rotate. We created an algorithm which checked every frame, if the angle of the line perpendicular to the surface of the garden and the line of the line between the vertical garden and sun, was equal. If so this meant that the garden was in the right position. After a few test runs (without the actual hardware) we discovered there was a bug in the software, which wouldn't detect certain points lying too close to the x-and y-axis because of the enormous slove of the line. We solved this by adding a correction function which will change the angle of the actual position of the sun by 0.1 degrees, which enables it to be detected.
After the processing part of the system was done we had to translate this into an actual movement of course. We did this by using arduino and a L298N motor controller. We sended the angle differences calculated in processing, to arduino, which meant if there was an angle difference the motor should turn on. And if there was no angle difference, the Smart Vertical Garden was in the right place. Processing was only giving information if the garden was rotating or not. The speed it actually rotated was always the same. This meant the time between every rotation and angle changes, was the write time. The only thing we had to sort out was how fast it actually had to rotate.

(*hardware*) There were first of all a few problems we had to solve. How would we be able to rotate our Smart Vertical Garden. To make this happen we used the following things. We used an electric drill, multiple gears, which were 3D printed, and would keep everything together and as a building platform. We connected the end of the drill to a small gear, and connected this with a bigger gear. By rotating the drill we could actually rotate the big platform the garden was on.
(*Software*) Now with the use of a Motor controller we were able to modify the speed by sending different values out of a PWM pin from arduino. The problem we did actually see was that by sending a to low PWM signal the motor hadn't enough power to actually start rotating.
By adding a pen to the big Gear we rotated the platform 360 degrees and looked at the paper if this was accurate or not. After running a couple of tests we found the most accurate value and the hard/soft-ware part of this project was finished.
(*Hardware*)One important thing, that was quite disappointing, was the fact that the current pulled from the 12V battery, caused the distracting warming element of the motor controller, to heat up very fast whenever the PWM signal was 'HIGH' (above 200). This disabled us to actually do prober test runs.

The second part of our project was about detecting the state of our plant, meaning checking the soil moisture, air humidity and temperature in order to detect whether the plant needs water. We build our vertical garden on top of a turning system (like described above) which consisted of a plate with a green holder for plants on top of it. The plate was held in the right angle by a pipe. Through this pipe there were electronics hidden, such as the wires that would go to the plant for checking soil moisture and temperature. There was also a water hose hidden in it, which would on top of it go to the plant, and at the bottom of the pipe it would go to a water reservoir. The air humidity sensor was built in the same sensor as the temperature sensor, to reduce the amount of wires.

For the software part we used Arduino again. With the use of certain libraries, it was easy to get the information from the sensors we placed. We created an algorithm that would check if the plant needs watering. We based this information on one main sensor, namely the soil moisture sensor. If the value from this sensor was too low, it told the water pump to add water for 5 seconds, then wait for a little bit, and check if the soil moisture was good enough now. Normally the soil moisture sensor would check each certain time the state of the ground, but if the temperature was above 25 or the air humidity above a certain level (which means it is a hot day), it would check the moisture more often than it would when there were normal levels. If it is both warm and has very high air humidity, it would check It even more often. One problem we had with our pump, is that we found out that our pump nearly didn't have enough power to put the water up through the pipe. If we used a more expensive/bigger pump, we would have avoided this problem. In the end it worked, but the water hose had to be full of water if you wanted it to come out right away.

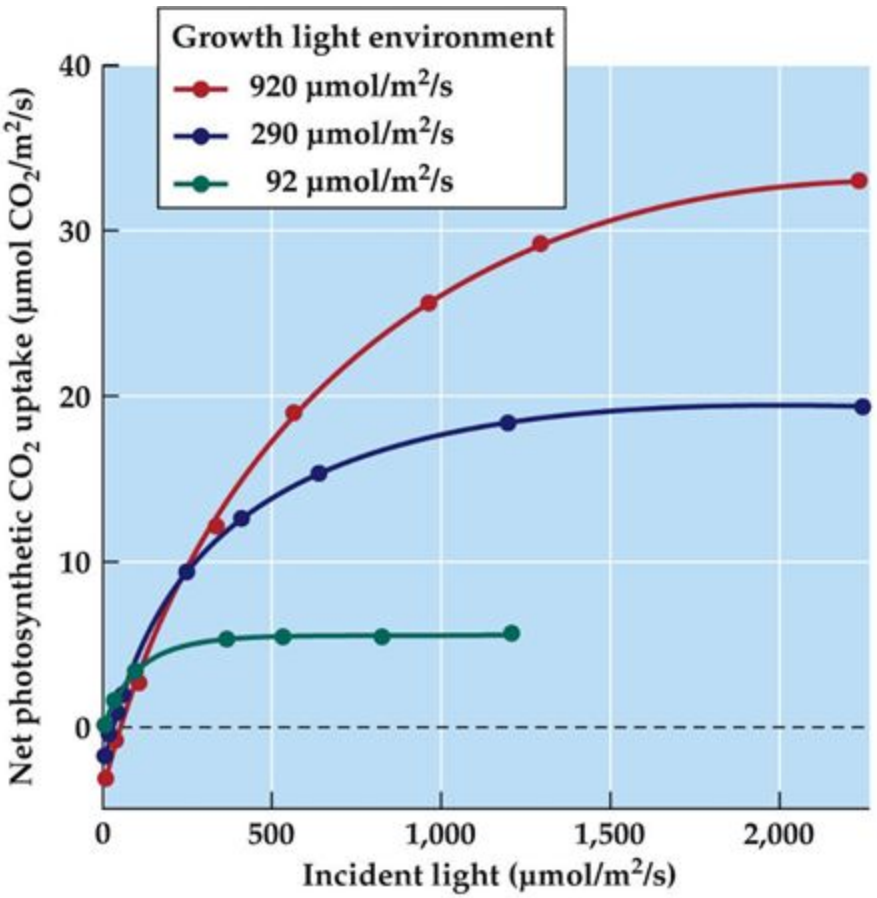
We displayed all this information of the sensors in a processing screen. Here the user could see all the features of his plant. We also wanted to create an app with this vertical garden. The idea of this app consisted of 3 parts: one part where you could see the status of your plant and find information about is (we kind of build it in processing). The second part consists of a kind of Facebook for plants, where you could show pictures of how your plant is doing and like others of friends. The last part consisted of a map, where you could see if anyone near you wanted to trade some veggies or flowers. The idea of this was, if you make too much of, for example, carrots, and someone else

makes too much of tomatoes, you can trade this fruit. In this way nothing will be lost, and you will have nice social contact. People in big cities, for example in flats, are our target for this.

4. Short description of results/observations/data gathered/graphs (1,000 words and/or 3-4 figures/graphs).

Our garden is very efficient because the plants get more sunlight.

If a plant doesn't get enough sunlight they won't be able to produce chlorophyll. A plant uses this for their green color, so they can't perform the process of photosynthesis and will die.
Study shows that every plant has different light requirements but they all prefer direct sunlight. Normally, if your plant did not get enough light you needed supplementing from a grow light. The plants use this light as energy for the photosynthesis process. The higher the light intensity, the faster they can perform this process. Therefore, plants which get a higher light intensity will grow faster.



As you can see in the graph above, a plant with a high light intensity takes up more CO2 (red line) then a plant with a lower light intensity (blue and green lines). A plant that gets 10 times more light intensity takes up about 7 times more CO2 and therefore grows 7 times faster.

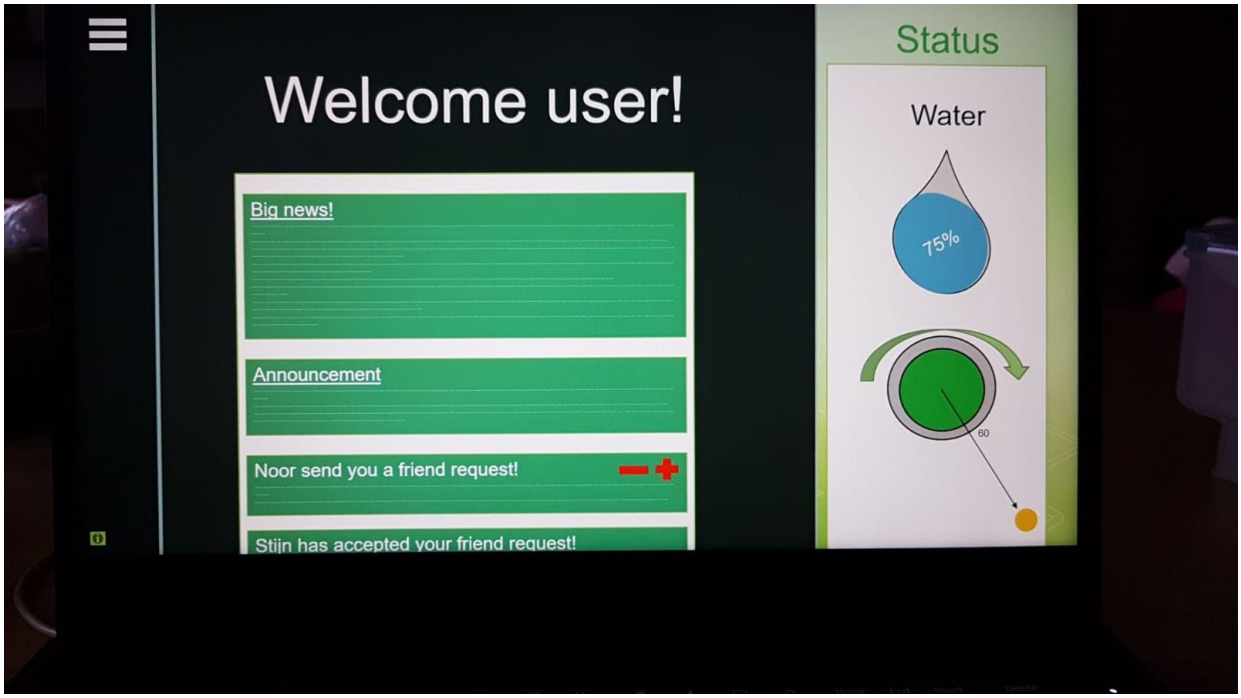
For our prototype we didn’t use rotating solar trackers but if our project gets produced for the marked we do want to use those. Using a single-axis soar tracker makes the garden catch 25-35% more sun than if it were standing still. If we want to use the double-axis solar tracker another 10% can be added to this.

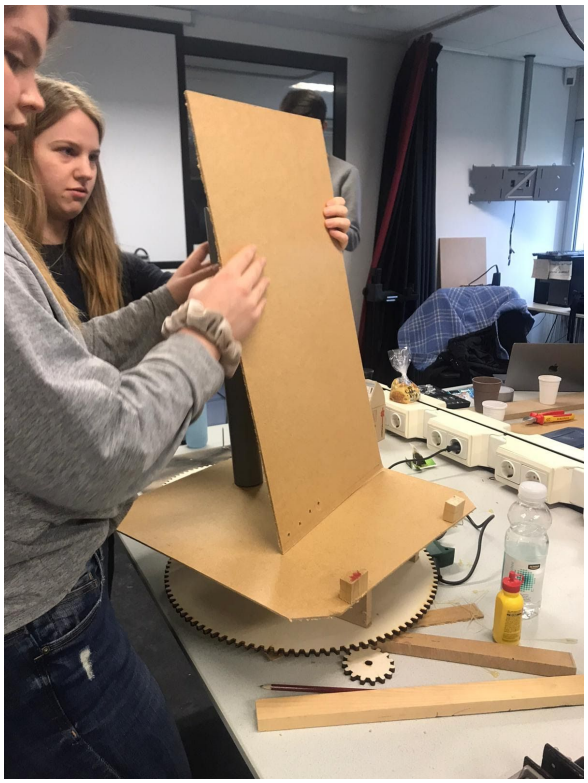
If a plant gets 45% more light. It will grow 45 X 0,7 = 31,5% faster then normally.
We could not do research ourselves. For our project we used a drill’s battery that was very old. Due to this we could not run our garden for 24 hours to collect data concerning the effectivity of our garden. We did not have enough time to optimize the electrics to make it able to run outside. In the future we could work on this so that we have more results to back up our theory research.
A second reason why our garden is efficient is because it makes it easier for people to grow their own crops. Right now the world population grows rapidly. To provide all these people with food, a lot of land is needed. This is not possible. The solution for this is to let people grow food on their own territory. Our garden makes it very easy and accessible. They only need to buy it, fill the water tank and plant some plants. After that they don’t have to look back at it, only when their crops are ready to harvest.

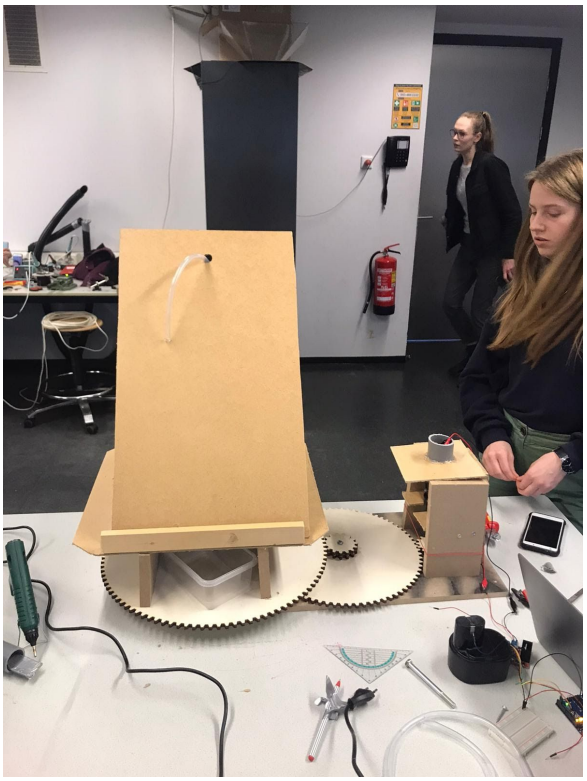
https://www.chegg.com/homework-help/questions-and-answers/refer-graph-showing-results-bjorkman-colleagues-studies-plant-responses-different-light-le-q29929995

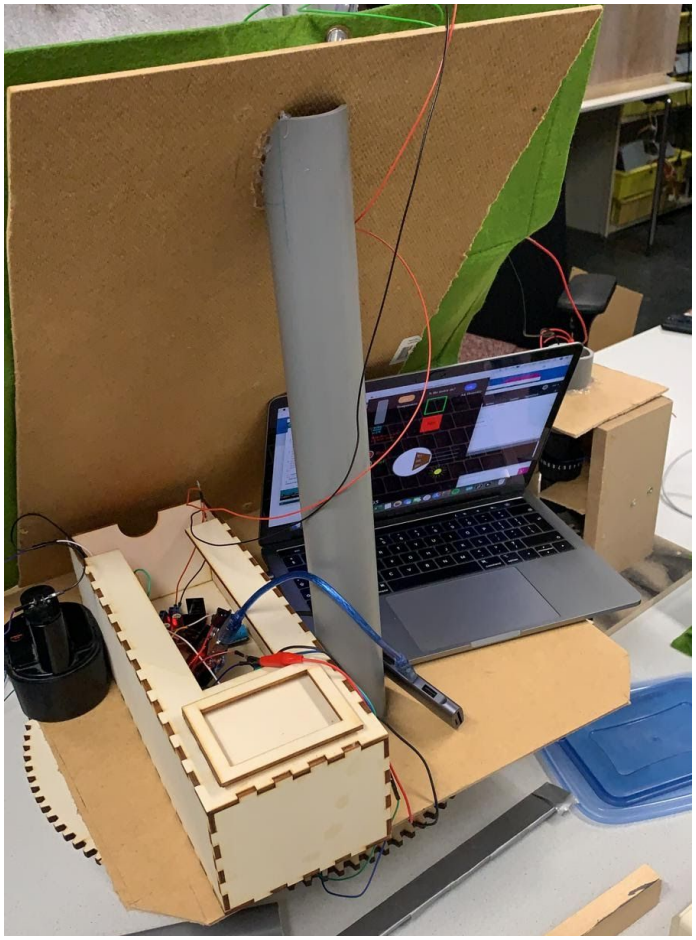
https://www.planetnatural.com/plant-care/
https://www.hunker.com/12340735/the-effect-of-light-intensity-on-plant-growth

5. Up to 5 photos of your prototype









6. Up to 2-minute video explaining what the prototype does.
https://youtu.be/sAyv_5jF4e8

7. Full working code of the project.

Arduino 1:

```
// total sketch for module 2 project: smart vertical garden
// made by Froukje Temme and Stijn brugman
// 13 January 2020


#include <Servo.h>
#define pump 5
#include "dht.h"
#define dht_apin A0 // Analog Pin sensor is connected to

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
unsigned int val = 0;   // variable to read the value from the analog pin
int waiter;
int sensorPin = A1; // reading data from A0 pin of ADC
float sensorValue = 0;
int timer1;
int timer2;
dht DHT;
int pumpstate;
int checkvalue;

void setup()
{
  Serial.begin(9600);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(pump, OUTPUT);
  timer1 = 0;
  pumpstate = 0; // 0 means out
  checkvalue = 1000;
}

void loop()
{
  if (Serial.available() > 0) { // reads the value of the potentiometer (value between 0 and 1023)
```

```
  val = Serial.read();
  if (val == 1) {
    digitalWrite(2, LOW);
    digitalWrite(3, HIGH);
    analogWrite(11, 255);
  } else if (val == 0) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    analogWrite(11, 0);
  } else if (val == -1) {
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    analogWrite(11, 255);
  }
}

// code for the humidity sensor in the ground
for (int i = 0; i <= 100; i++) {
  sensorValue += analogRead(sensorPin);
}
sensorValue = sensorValue / 100.0;

// code for temperature and humidity sensor
DHT.read11(dht_apin);
if (millis() > timer1 + checkvalue) {
  timer1 = millis();

  // send data
  Serial.print("A");
  Serial.println(pumpstate);
  Serial.print("B");
  Serial.println((int)sensorValue);
  Serial.print("C");
  Serial.println((int)DHT.humidity);
  Serial.print("D");
  Serial.println((int)DHT.temperature);

}


//    if (pumpstate == 0) { //if pumpstate is off
//      analogWrite(pump, 255); // put it on
//      pumpstate = 1; // put the state on on)
//    } else {
//      analogWrite(pump, 0); // put it off
```

```
  //      pumpstate = 0;
  //    }
  //  }

  // The pump will be turned on if the soil value is too dry
  if (sensorValue > 700) { // when the value of the soil is too low, there will be water
    if (millis() > timer2 + 5000) {
      timer2 = millis();
      if (pumpstate == 0) { //if pumpstate is off
        analogWrite(pump, 255); // put it on
        pumpstate = 1; // put the state on on)
      } else {
        analogWrite(pump, 0); // put it off
        pumpstate = 0;
      }
    }

    if ((int)DHT.humidity >= 60 && ((int)DHT.temperature >= 28)) {
      checkvalue = 400;
    } else if ((int)DHT.humidity >= 60 || (int)DHT.temperature >= 28 ) {
      checkvalue = 600;
    } else {
      checkvalue = 1000;
    }
  }
}
```

Processing 1:
This consist of many classes, but also some data folders containing the information of the sun position.

```
Serial port;
import processing.serial.*; // import the library
import processing.net.*;
Client client;

float x, y;
float x_, y_;
float posX, posY;
float posX1, posY1;
float dx, dy;
float rc1, rc2;
float rotate;
color color1;
```

```
boolean reset;
float rotationSpeed;
float check;
boolean border1;
float rotationSun;
boolean active;
int speed;
String[] lines;
float [] angle;
String[] rules;
float [] resetTime;
int g;
String dayChar;
int dayInt;
float updateValue;
float oldUpdateValue;
boolean nextDayLoading;
float scalingNight;
float waitingTime;
int night;
float currentTime;
float startingTime;
float timeDifference;
float waiter;
float dValue;
int f;
color buttonColor = 255;
int transp = 230;
color buttonColor2 = 255;
int transp2 = 230;

int rotatingState= 0;

//nieuwe code
Control control;
Sensors sensors;
DigitalClock digitalClock;
Button button;

PFont myFont;
int a;
int b;
int c;
int d;
int timerSpender;
```

```
void setup() {
  client = new Client(this, "130.89.91.63", 8080);
  myFont = loadFont("Didot-Italic-48.vlw");
  textFont(myFont, 35);

  control = new Control();
  sensors = new Sensors();
  digitalClock = new DigitalClock(40, 20, height+10);
  button = new Button();

  control = new Control();
  println("Available serial ports:");
  for (int i = 0; i<Serial.list ().length; i++) {
    print("[" + i + "] ");
    println(Serial.list()[i]);
  }

  size(600, 800);
  rectMode(CENTER);
  // port = new Serial(this, Serial.list()[9], 9600);
  //DataBase position sun
  dayInt = 1;
  for (int i = 1; i<dayInt+1; i++) {
    dayChar = str(dayInt);
    lines = loadStrings("Sun" + dayChar + "Jan.txt");
  }
  rules = loadStrings("waitingTime.txt");
  night = 0;
  x = width/2;
  y = width/2;
  g = 0;
  rotationSun = PI/50;
  rotate = HALF_PI;
  active = false;
  x_ = cos(rotationSun)*200+300;
  y_ = sin(rotationSun)*200+300;
  color1 = color(255, 255, 255);
  rotationSpeed = 0;
  reset = false;
  frameRate(200);
  check = rc1/rc2;
  border1 = false;
  speed = 1;
  nextDayLoading = false;
```

```
  f = 5*1000;
  waitingTime = f;  //real waitingTime should be 900000
  scalingNight = 2*1000;  //real factor should be 3600000

  currentTime = 0;
  startingTime = 0;
  timeDifference = 0;
  oldUpdateValue = 180;
  //Creatin floats to declare and get the values out of the string
  angle  = new float[lines.length];
  for (int i = 0; i<lines.length; i++) {
    angle[i] = float(trim(lines[i]));
  }

  resetTime  = new float[rules.length];
  for (int i = 0; i<rules.length; i++) {
    resetTime[i] = float(trim(rules[i]));
  }
}

void draw() {
  display();
  update();
  restart();
  button.display();
  button.update();

  pushMatrix();
  translate(0, -30);
  control.display();
  sensors.display(control.b, control.d, control.a, control.c);
  digitalClock.getTime();
  digitalClock.display();
  popMatrix();
}

void display() {
  pushMatrix();
  translate(0, 200);
  background(0);
  fill(150);
  //base
  ellipse(x, y, 210, 210);
  fill(255);
  ellipse(x, y, 200, 200);
```

```
  //sun
  fill(255, 255, 0);
  ellipse(x_, y_, 40, 40);
  fill(100);
  pushMatrix();
  translate(x, y);
  rotate(rotate);
  fill(200);
  //vertical garden
  fill(#935D0C);
  quad(30, -95, -30, -95, -70, -10, 70, -10);
  stroke(255);
  fill(160);
  noStroke();
  quad(-50, -20, -30, -20, -25, -45, -40, -45);
  quad(50, -20, 30, -20, 25, -45, 40, -45);
  quad(10, -20, -10, -20, -8, -45, 8, -45);
  fill(255, 0, 0);
  ellipse(-35, -35, 12, 10);
  fill(0, 200, 0);
  ellipse(-35, -30, 6, 5);
  stroke(255);
  //detecting lines
  line(30, -95, 30, -250);
  line(-30, -95, -30, -250);
  stroke(color1);
  line(0, -95, 0, -250);
  stroke(0);
  popMatrix();
  for (float i = 0; i<2*PI; i+=PI/8) {
    pushMatrix();
    translate(x_, y_);
    rotate(i);
    stroke(255, 255, 0);
    line(30, 0, 50, 0);
    noStroke();
    popMatrix();
  }
  popMatrix();
}

void update() {

  //Mapping the rotationfactor from radions into degrees
  updateValue = map(rotate, 0.5*PI+PI/50, 1.5*PI, 180, 0);
```

```
//sending the angle to arduino
//if (millis()>waiter) {
//waiter = millis();
dValue =  oldUpdateValue-updateValue;
oldUpdateValue = updateValue;
//port.write((dValue));
//}
//port.write((dValue));
// if (port.available()>0) {
//print((char)port.read());
// }
currentTime = millis();
timeDifference = currentTime - startingTime;
if (dValue>0) {
  rotatingState = 1;
}

if (dValue == 0) {
  rotatingState = 0;
}

if (dValue<0) {
  rotatingState = -1;
}

//port.write((int)(rotatingState));
//println("rotatingState="+rotatingState);
////println("dValue=" +dValue);
client.write(rotatingState);
println(rotatingState);


//Starting position5
posX = sin(rotate);
posY = cos(rotate);
//Position outsidecircle of the model
posX1 = sin(rotate)*90+300;
posY1 = -cos(rotate)*90+300;
//position of the sun
x_ = cos(rotationSun)*200+300;
y_ = sin(rotationSun)*200+300;
//line difference
dx = posX1-x_;
dy = y_-posY1;
```

```
//coefficient of the non-visble lines between sun and SG, and a penduticular line on SG
rc1 = posY/posX;
rc2 = dy/dx;
check = rc1/rc2;

//importing rotation from textfile, changing to radians
rotationSun = (angle[g]/360)*2*PI-HALF_PI;
//Correction if the position is on the x-axis
if (angle[g] >= 177.5 && angle[g]<=182.5) {
  g++;
  //waitingTime*=2;
}

//checking if the position of the garden regarding to the position of the sun is correct
if (active) {
  if (x_>=300 && rotate>=0 && rotate<=PI) {
    if ( check>0.99 && check<1.01) {
      color1 = color(0, 255, 0);
      reset = true;
      g++;
      waitingTime = f;
      rotationSpeed = 0;
      startingTime = millis();
    }
  } else if (x_<300 && rotate<=2*PI && rotate>=PI) {
    if ( check>0.99 && check<1.01) {
      color1 = color(0, 255, 0);
      reset = true;
      g++;
      rotationSpeed = 0;
      startingTime = millis();
    }
  }
}

if (check != 1) {
  if (nextDayLoading&&border1) {
    rotate-=rotationSpeed*speed;
  }
  if (!border1 && !nextDayLoading) {
    rotate += rotationSpeed*speed;
  }
  if (rotate<.5*PI+PI/50) {
    border1 = false;
  }
```

```
  }
}

void restart() {
  if (reset && timeDifference > (waitingTime)/speed) {
    reset = false;
    rotationSpeed = PI/7000;
    color1 = color(255);
  }

  if (nextDayLoading) {
    if (timeDifference >= resetTime[dayInt-1] * scalingNight) {
      nextDayLoading = false;
      active = true;
    }
  }

  if (g >= angle.length) {
    nextDayLoading = true;
    border1 = true;
    active = false;
    //load the new date information
    dayInt++;
    for (int i = 1; i<dayInt+1; i++) {
      dayChar = str(dayInt);
      lines = loadStrings("Sun" + dayChar + "Jan.txt");
    }
    angle  = new float[lines.length];
    for (int i = 0; i<lines.length; i++) {
      angle[i] = float(trim(lines[i]));
    }
    g = 0;
  }
}


class Button {
  float distance;
  float distance2;
  int translate;
  Button() {
  }

  void display() {
    //resetknop
```

```
  rectMode(CENTER);
  fill(200);
  rect(50, 380, 56, 56, 230);
  fill(0);
  ellipse(50, 380, 48, 48);
  fill(buttonColor, 0, 0, transp);
  ellipse(50, 380, 40, 40);

  textSize(30);
  text("Reset",90,390);

  //Start button
  rectMode(CENTER);
  fill(200);
  rect(50, 450, 56, 56, 230);
  fill(0);
  ellipse(50, 450, 48, 48);
  fill(0, buttonColor2, 0, transp2);
  ellipse(50, 450, 40, 40);

  text("Start",90,460);
}

void update() {
  distance = dist(50, 380, mouseX, mouseY);
  distance2 = dist(50, 450, mouseX, mouseY);
  if (distance<20 && mousePressed) {
    reset();
    buttonColor = 200;
    transp = 190;
  } else {
    buttonColor = 255;
    transp = 230;
  }

  if (distance2<20 && mousePressed) {
    active=true;
    rotationSpeed = PI/4000;
    buttonColor2 = 200;
    transp2 = 190;
  } else {
    buttonColor2 = 255;
    transp2 = 230;
  }
}
```

```
void reset() {
  dayInt = 1;
  for (int i = 1; i<dayInt+1; i++) {
    dayChar = str(dayInt);
    lines = loadStrings("Sun" + dayChar + "Jan.txt");
  }
  rules = loadStrings("waitingTime.txt");
  night = 0;
  x = width/2;
  y = width/2;
  g = 0;
  rotationSun = PI/50;
  rotate = HALF_PI;
  active = false;
  rotationSun = (angle[g]/360)*2*PI-HALF_PI;
  x_ = cos(rotationSun)*200+300;
  y_ = sin(rotationSun)*200+300;
  color1 = color(255, 255, 255);
  rotationSpeed = 0;
  reset = false;
  frameRate(200);
  check = rc1/rc2;
  border1 = false;
  speed = 1;
  nextDayLoading = false;
  f = 5*1000;
  waitingTime = f;  //real waitingTime should be 900000
  scalingNight = 2*1000;  //real factor should be 3600000

  currentTime = 0;
  startingTime = 0;
  timeDifference = 0;
  oldUpdateValue = 180;
  //Creatin floats to declare and get the values out of the string
  angle  = new float[lines.length];
  for (int i = 0; i<lines.length; i++) {
    angle[i] = float(trim(lines[i]));
  }

  resetTime  = new float[rules.length];
  for (int i = 0; i<rules.length; i++) {
    resetTime[i] = float(trim(rules[i]));
```

```
    }
  }
}


class Clock {
  int h, m, s;
  Clock() {
  }

  void getTime() {
    h = hour();
    m = minute();
    s = second();
  }
}

class Control {
  String buff = "";
  char header[] = {'A', 'B', 'C', 'D', 'X'};
  int value[] = new int[5];
  int NEWLINE = 10;
  int a;
  int b;
  int c;
  int d;
  int e;
  int input;
  String values;
  int state;
  char firstCharacter;
  String read;

  Control() {
    background(0);
    a = 0;
    b = 0;
    c = 0;
    d = 0;
    e = 0;
  }

  void display() {
    {
      while (client.available() > 0) {
```

```
    read = client.readString();
    //clientEvent(read); // read data
    // println(read);
    //store sended data into a String

    //store first character of the string in a character
    firstCharacter = read.charAt(0);


    //checking the first character
    if (firstCharacter == 'A') {
      read = read.substring(1);
      a = Integer.parseInt(read);
      println(a);
    }
    if (firstCharacter == 'B') {
      read = read.substring(1);
      b = Integer.parseInt(read);
    }
    if (firstCharacter == 'C') {
      read = read.substring(1);
      c = Integer.parseInt(read);
    }
    if (firstCharacter == 'D') {
      read = read.substring(1);
      d = Integer.parseInt(read);
    }
    if (firstCharacter == 'E') {
      read = read.substring(1);
      e = Integer.parseInt(read);
    }
  }
  //a = value[0]; // motor 1 en 0 , al laten staan.
  //// b = int(map((value[1]), 0, 1035, 0, 100)); // tussen 0 1n 1023?? nee want was al 1035);
  //b = value[1];
  //c = value[2]; // air humidity, gewoon zo laen
  //d = value[3]; // temp, gewoon zo laten
  //e = value[4];
  //  println(a, b, c, d, e);


  //if (client.available() > 0) {
  //}
 }
}
```

```
//  void clientEvent(int client)
//  {
//    try {    // try-catch because of transmission errors
//      if (client != NEWLINE) {  //checks if all values are checked: 3 values + new line and cahracter
return
//        buff += char(client);  // if not, then go on, if yes: then do this:
//      } else {
//        // The first character tells us which axis this value is for
//        char charr = 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<5; z++) {
//          if (charr == header[z]) {
//            value[z] = Integer.parseInt(buff); //it will now return a value between 0 and 1023, so /4 and it
will fit in an int (0-255)
//          }
//        }
//        buff = "";         // Clear the value of "buff" (after 1 times all values)
//      }
//    }
//    catch(Exception e) {  // this checks if there is data coming from the arduino
//      println("no valid data");
//    }
//  }
}


class DigitalClock extends Clock {
  int fontSize;
  float x, y;
  PFont hightech;

  DigitalClock(int _fontSize, float _x, float _y) {
    fontSize = _fontSize;
    x = _x;
    y = _y;
     hightech = loadFont("LCD5x8H-48.vlw");
  }

  void getTime() {
    super.getTime();
```

```
  }

  void display() {
     textFont(hightech,fontSize);
    text (h + ":" + nf(m, 2) + ":" + nf(s, 2), x, y);
  }
}
```

Processing 2: server

```
import processing.net.*;
import processing.serial.*;
Serial port;

Server s;
Client client;

String buff = "";
char header[] = {'A', 'B', 'C', 'D', 'X'};
int value[] = new int[5];
int NEWLINE = 10;
int a;
int b;
int c;
int d;
int e;

void setup()
{
  port = new Serial(this, Serial.list()[9], 9600);
  background(0);
  a = 0;
  b = 0;
  c = 0;
  d = 0;
  e = 0;

  size(450, 255);
  background(204);
  stroke(0);
  s = new Server(this, 8080); // Start a simple server on a port
  for (int i = 0; i<Serial.list ().length; i++) {
    print("[" + i + "] ");
    println(Serial.list()[i]);
```

```
  }
  port = new Serial(this, Serial.list()[4], 9600);
}

void draw()
{
  while (port.available() > 0) {
    serialEvent(port.read()); // read data
  }
  a = value[0]; // motor 1 en 0 , al laten staan.
  b = int(map((value[1]), 0, 1035, 0, 100)); // tussen 0 1n 1023?? nee want was al 1035);
  c= value[2]; // air humidity, gewoon zo laen
  d = value[3]; // temp, gewoon zo laten
  e = value[4];
  println(e);

  s.write("A"+ a);
  delay(200);
  s.write("B"+b);
  delay(200);
  s.write("C"+c);
  delay(200);
  s.write("D"+d);
  delay(200);
  s.write("X"+e);
  delay(200);

  client = s.available();
  if (client != null) {
    int send = client.read();
    println("server received:" + send);
    port.write(send);
  }
}

void serialEvent(int serial)
{
  try {    // try-catch because of transmission errors
    if (serial != NEWLINE) {  //checks if all values are checked: 3 values + new line and cahracter
return
      buff += char(serial);  // if not, then go on, if yes: then do this:
    } else {
      // 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<5; z++) {
        if (c == header[z]) {
          value[z] = Integer.parseInt(buff); //it will now return a value between 0 and 1023, so /4 and it
will fit in an int (0-255)
        }
      }
      buff = "";         // Clear the value of "buff" (after 1 times all values)
    }
  }
  catch(Exception e) {  // this checks if there is data coming from the arduino
    println("no valid data");
  }
}
```