UNIVERSITEIT TWENTE.

SMART ENVIRONMENTS PROJECT

Summary report

HyGrow

Timo van Beelen
Ellis Dijkstra
Ivo Hagenbeek
Ard Kotte
Louis van Maurik
Veerle Mooren
Dennis Peeters

The problem

The problem nowadays is that food is produced far away, in other countries. This could lead to food insecurity in less developed countries. Secondly, the food has to travel far; this means that there is a big chance that the food will be bad by the time it gets here. The food has to be frozen, which causes a decrease in quality, or it has to be shipped and transported very fast. On top of that, the emissions caused by the (hasty) transport of vegetables puts a strain on the environment as well.

Because of those reasons, it is better to produce food at home. Unfortunately, there is not always enough space in urban areas to start growing food at home. We are not yet able to move towards rooftop-gardening and not everyone has a backyard.

The solution for these problems is a hydroponic system, which offers a way to grow food at home. It can be built both horizontally and vertically, so it can easily be adjusted to the space it needs to be fit in.

Hydroponics is the term that is used for growing plants on water instead of soil. This soil is replaced with an inert medium, such as vermiculite or rockwool, that supports the root system. Since these grow media do not contain nutrients, these need to be dissolved in the water. The roots from the plant are in direct contact with the nutrient solution, whilst they are still able to take in enough oxygen to breathe.

However, these systems come with one big disadvantage: the system needs to be very accurately maintained and this costs you a lot of time. The amount of nutrients in the solution and the pH levels need to be monitored regularly and they need to be adjusted carefully when necessary. This may not take up a big part of your day, but eventually you will have to put in a lot of hours. You also cannot forget to check your system every once in a while; the slightest problem can cause your plants to die in a matter of hours.

Our solution

This is why we have chosen to automate our hydroponic system. We have used a lot of sensors to keep track of the values of the solution. This will take away the need to check every few hours if your system is still working the way it should. But before we could start with automating our system, we first had to decide which type of hydroponic system we wanted to build.

There is a variety of hydroponic systems, but after some research and weighing the pros and cons against each other, we went with the Nutrient Film Technique (NFT). NFT uses a shallow nutrient rich stream of water and re-circulates it past the bare roots of plants in a watertight channel. The NFT system needs a constant flow of water to keep the plants alive. There are two main components to an NFT system: the grow trays (pipes) and the reservoir. The reservoir holds most of the water and uses a water pump to pump the water into the grow trays. In these grow trays are net cups suspended that contain the plants and a growing media. To enable the nutrients to pass by the roots fluently, growers use slightly tilted tubes or channels so the flow is slightly strengthened by gravity. A well designed NFT system is based on using the right slope, flow rate and channel length. This system is best suited for lightweight, fast-growing plants like lettuce.

After some research, we found all the necessary components to automate our system. We use an LDR to monitor the light; if it is not light enough, we use LED's to make sure the plants get enough light. We used a flow rate sensor to check if the flow rate is right. Because of this, you do not have to check constantly if the system is still pumping water up.

Since checking the amount of nutrients and pH level takes a lot of time, we found some sensors that will take care of this. We installed a pH-sensor to measure the pH level; the pH is not automatically adjusted, but the user should only have to change the pH once every month. The pH level can still be read on the display at the side of the system. We also placed a nutrient sensor that measures the conductivity in water; we found a corresponding valve that will dispose some nutrient solution into the reservoir when necessary. This value is checked once every day.

Since all these things are taken care of, it is not even near as time intensive as it would have been when everything had to be adjusted manually. Therefore, a hydroponic system is now a perfect solution to solve the aforementioned problems.

Methods and tools

The physical build of our setup is decisively simple. While NFT-systems are well suited to be installed against a wall, we chose to build a wooden frame to support our system - with practicality and movability in mind. It consists of two vertical wooden beams with a horizontal beam placed on top; the bottom is secured with a big panel. Underneath this panel is more than enough space for our Arduino all the other electronics that we used and do not need to be in contact with specific parts of the system.

This whole frame was built with some screws and secured with angled metal pieces. For the system itself we used two large grey PVC pipes as trays for the plants. Between themselves they are connected by pieces of a tube. These pipes are attached to the wooden frame with cut outs in the shape of these pipes. These pipes are slightly tilted, so the water flows down naturally.

In the PVC tubes we made 4 holes. There was room for a fifth one, but because of the connection between the pipe and the tube, we could not use that fifth hole at the lower end of the pipe. On top of each of these holes we placed a small laser-cut plate with a circular hole in the middle, from where we suspended our plants in so-called net cups, accompanied by a growing medium (Rockwool in our case). These cups are basically small plant cups with a large number of holes in them which makes them look like nets. These net cups are necessary to allow the plant roots to develop themselves. Eventually these roots will start growing through the holes into the water stream running through the pipes. When the roots get too big for the cup, they grow out of the holes and end up in the pipe where the water is flowing to supply them with nutrients.

The surface of the panel is occupied by a big plastic box that acts as the main reservoir. At the top of the construction and the highest point of the flow, is a smaller box that acts as a reservoir: it gathers the water before it sends it on its way through the system. All the previously mentioned components can be bought at your local hardware store (Praxis, Gamma etc.), we just modified and assembled them using saws, glue, screws, sandpaper and a bit of laser cutting.

The controller that keeps itself busy with controlling all the electronics, is the Arduino Mega. We chose this specific model, because we had a lot of sensors that needed to be connected and the Mega offers a big number of pins. The things connected to our Arduino are:

A Flow Rate Sensor. The Flow Rate Sensor allows the setup to accurately measure the flow rate of the water in the pipes and take action accordingly. In our setup this flow rate data is mostly used to detect leaks or other flaws in the system. A decrease in the flow rate could mean that a lot of water has somehow escaped the system, that the pump is somehow malfunctioning or that something is blocking the water from flowing freely. If the flow rate drops below the desired value the LED-strip (described in more detail below)

will indicate to the user that there is a problem that needs to be taken care of.

pH Sensor. Plant roots are very sensitive to changes in the pH value of the water they are standing in and a pH value that is too high or too low can severely damage the plants. Thus, we have integrated a pH sensor in the big water reservoir of the setup. A pump to automatically add a pH solution turned out to be somewhat unfeasible for this project.

Nutrient Level Sensor. The plants in our setup grow primarily from the nutrients they absorb from the water their roots are in. These nutrients have to be closely monitored as to not overfeed or, more importantly, starve the plants. Luckily these nutrients consist of salts and salt dissolved in water makes the water more conductive. Therefore, we can easily measure the nutrient level by placing a submersible conductivity sensor (TDS sensor) in the water.

LED-Strip. On the bottom of the horizontal beam of the setup is an RGB-LED strip attached with individually programmable lights. This strip has two tasks; its primary function is to act as grow lights for the plants placed in the setup. In this function the individual LED's are configured to emit red, blue or purple (ultraviolet) light as that is the optimal lighting condition for most plants. The LED-strip also acts as a way to alert the user of certain changes in the system. If a problem is sensed, all the LED's will turn red to signal to the user that something is wrong.

All these sensors and actuators come with programming libraries, so they can be utilized properly, and there is a passionate community behind every corner of the Arduino world to fill out the blanks and help us on our way. Thus, programming was still a time consuming and difficult task, but it turned out to be a less daunting task than it initially seemed.

Some of the electronic components (wires, mostly) took some soldering to install properly and we had to solder together a PCB to get some electronics into place. Most of it was plug and play however.

Getting water down is not really a problem; getting water up is significantly harder. To accomplish this, we used a fountain pump. Fountain pumps can pump a large amount of water at a relatively slow flow rate which is exactly what we needed for our system. This pump is not connected to our Arduino, because it has to run all the time to keep the plants supplied with fresh water and nutrients, so the pump is plugged directly into a wall outlet.

Results/observations

Both during the building stage and the testing phase, we have run into several problems with our project. We are still very proud with our achievements; even though we have not been able to test if our project will last through a longer period of time.

Results

Our project was not as focused on our actual results as might be expected – we have planted our cabbage seedlings and we have grown them over a certain period of time, but in order to see if our project is actually working, we will need a bigger time span. Our project has lasted one and a half week this far.
Unfortunately, we have ran into an unexpected issue: after one and a half week of proudly running, we came to find our system out of water. Within hours, all our plants died. We replaced a few plants with the few spare seedlings that we still had left for the demonstration market the next day, but our testing period has to be started all over. However, we are very happy that our system worked the way we wanted it to.

Observations

Over the course of the past months, we have had a few setbacks on our project. The final result we had was representable for the goals that we wanted to achieve, but most of us agreed that we would like to improve more to this project on top of the designated time.

First assembly**:** With all the woodwork in place, the system seemed to perform rather well for a first run. There were hardly any leaks (most of which could be fixed with a bit of glue) and the water was running through the system as it should. With this assembly we did not work with the plants yet, so there were no observations at that point regarding the plants. All the sensors did their work; the temperature sensor and light sensor displayed the right values. The pH-sensor and the nutrient sensor had some difficulties at first, but once we calibrated the sensors with tap water the pH values and nutrient values seemed correct.

First test**.** When we wanted to add the lettuce cuttings to the system, the flowrate sensor had stopped working. The sensor was not capable of handling the flow rate that we wanted it to keep up with. Therefore, the Flow Rate Sensor started to block the water flow. The amount that it let through was smaller than the amount of water that it got; this eventually led to the gathering from water at the lower end of the pipe. Because of this, the closest plant stood in a layer of water instead of a flowing stream. We quickly replaced the Flow Rate Sensor with a regular piece of tube that was able to handle the waterflow. Unfortunately, we never found out what went wrong. The sensor should have been capable of handling way bigger amounts of water, which leads us to suspect that this sensor was broken. Because of a

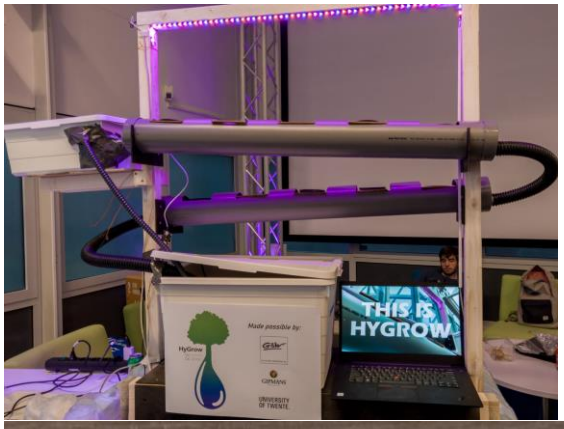
too long delivery period, we were not able to replace this part in time, so we left it out.

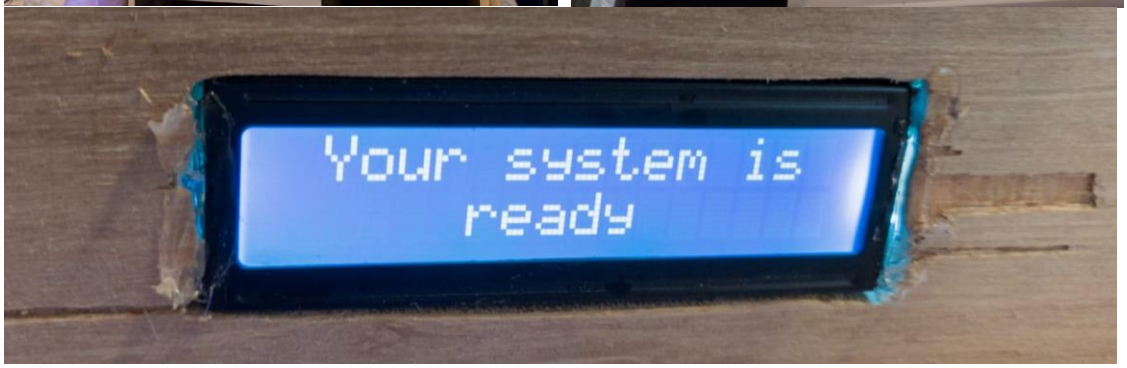
Second test. After a week, we checked our system thoroughly. We noticed that a few of our plants died, but we suspect this has mostly to do with the fact that these saplings were grown in normal soil and they did not adjust well to the switch towards hydroponics. The plants with stronger roots had survived; we replaced the dead plants with new ones.
We did come across another setback; we found out that our power supply died. We switched towards a smaller power supply, but unfortunately, we were not able to use our valves for the nutrient-adding anymore. They were still being measured, but we had to add those ourselves.

Third test. We did a final check one day before the demonstration market, only to find out that all of our plants had died. We have found the cause, namely that our water reservoir was completely empty. How this happened, we do not know.
We have spent a lot of time trying to figure out where it could have gone wrong; we could only find a minor leak, but we do not think we caused this. According to the state of a sensor close to the reservoir, we think someone accidently bumped into our project, which led to the sensor coming loose and the connection between the reservoir and the tubes to come loose. Since our pump was running constantly, all of the water was pumped out of the system. This meant that our plants had not gotten water for a few days, which left them no chance to survive. We placed all the plants that we had left.
We fixed our system as good as possible, to make sure the system was in the best state possible for the demonstration market. We refilled the reservoir, to let it run for that final day. Luckily, none of the electronics were damaged by the leaking.

Demo market: For the demonstration market we had to move the system downstairs, which meant we had to take all the water out of the top reservoir. Luckily, we succeeded – be it with some physical help from other groups. We were able to get everything to work as it had been doing for those last few days, but the hole we tried to fix the day before, started leaking again. We did not bring our glue (since it has to dry 24 hours), which left us with nothing but duct tape to repair the leaking. Despite everything, we were able to show what we wanted to and we were quite satisfied with what we presented.

Photos





Code

```
/*
   Code for the Hygrow system
   version: 2.0
   Developed by the team of Hygrow
   adaptations of other codes and examples have been used
    -pH part by DF-Robot
    -TDS part by DF-Robot
*/
#include <Time.h>
#include <DS1307RTC.h>  // a basic DS1307 library that returns time as a
time_t
#include <Adafruit_BME280.h>  //library for the Temperature sensor
#include <Adafruit_Sensor.h>  //library for the Temperature sensor
#include <Wire.h>
#include <FastLED.h>  //library to control the light
#include <LiquidCrystal_I2C.h>  //library to control the LCD display over I2C
#include <SPI.h>

//below is the defining of pins to certain names and assigning values to
variables
#define NUM_LED'S 60
#define TdsSensorPin A4
```

```
#define VREF 5.0   // analog reference voltage(Volt) of the ADC
#define SCOUNT 30  // sum of sample point
#define SEALEVELPRESSURE_HPA (1013.25)
#define SensorPin A0            //pH meter Analog output to Arduino Analog
Input 0
#define Offset 0.00            //deviation compensate
#define samplingInterval 20
#define printInterval 800
#define ArrayLenth  40    //times of collection
#define motorPin  2

Adafruit_BME280 bme;

//create extra variables and assign start variables to them
unsigned long delayTime;
int analogBuffer[SCOUNT]; // store the analog value in the array, read from
ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0, copyIndex = 0;
float averageVoltage = 0, tdsValue = 0, temperature = 25;
int TDSvalue;
int pHArray[ArrayLenth];   //Store the average value of the sensor feedback
int pHArrayIndex = 0;
static float pHValue, voltage;

LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16
chars and 2 line display
CRGB LED's[NUM_LED'S];  //initialise the array of LED's in the strip

void setup()
{
  lcd.init();                      // initialize the lcd
  lcd.init();
  // Print the booting message to the LCD.
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Hygrow is booting");
  lcd.setCursor(0, 1);
  lcd.print("Please stand by");
  delay(2000);

  Serial.begin(9600);   //start the serial monitor
  setSyncProvider(RTC.get);   //the function to get the time from the RTC
  if (timeStatus() != timeSet)
    Serial.println("Unable to sync with the RTC"); //give information if the RTC
is working properly
  else
```

```
    Serial.println("RTC has set the system time");

  FastLED.addLED's<WS2812B, 6, GRB>(LED's, NUM_LED'S); //give the library
information about the LED's and initialize them
  //set some of the pins to the correct function
  pinMode(A3, INPUT);
  pinMode(TdsSensorPin, INPUT);
  pinMode(motorPin, OUTPUT);
  //start the temperature sensor
  unsigned status;
  status = bme.begin(0x76);

  for(int i = 0; i <= 100; i++){  //loop to start the sensors and make sure they
give the right values
    PHsensor();
    TDSsensor();
    delay(400);
  }
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("Almost ready");
  lcd.setCursor(0, 2);
  lcd.print("Just a second");
  delay(3000); //delay to prevent problems with the system

}

void loop()
{
  /*the main code for the system
     It displays all the variables that are important on the display so the user
can monitor them
     Next to that it calls the voids in between for all the background processes
     The delays are so the reader can read the value properly
  */
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print("Your system is");
  lcd.setCursor(5, 1);
  lcd.print("ready");
  delay(1000);
  digitalClockDisplay();
  Automation();
  lcd.clear();
  lcd.setCursor(3, 0);
  PHsensor();
  lcd.print(pHValue, 2);
  lcd.setCursor(2, 1);
```

```
  lcd.print("PH-value");
  delay(2000);
  lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print(analogRead(A3));
  lcd.setCursor(3, 1);
  lcd.print("Light sensing");
  delay(   2000);
  lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print(bme.readTemperature());
  lcd.print(" *C");
  lcd.setCursor(3, 1);
  lcd.print("Temperature");
  delay(2000);
  TDSsensor();
  lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print(TDSvalue);
  lcd.setCursor(3, 1);
  lcd.print("TDS sensor");
  delay(2000);
}

void PHsensor() {
    /*
     The code below is part of the example provided by the manufacturer. It
has been slightly adapted to suit our needs,
     but it is left mostly untouched as it serves as the proper calibration for
the sensor
  */

    pHArray[pHArrayIndex++] = analogRead(SensorPin);
    if (pHArrayIndex == ArrayLenth)pHArrayIndex = 0;
    voltage = avergearray(pHArray, ArrayLenth) * 5.0 / 1024;
    pHValue = 3.5 * voltage + Offset;

    Serial.print("Voltage:");
    Serial.print(voltage, 2);
    Serial.print("    pH value: ");
    Serial.println(pHValue, 2);

}

double avergearray(int* arr, int number) {
  int i;
  int max, min;
  double avg;
```

```
  long amount = 0;
  if (number <= 0) {
    Serial.println("Error number for the array to avraging!/n");
    return 0;
  }
  if (number < 5) { //less than 5, calculated directly statistics
    for (i = 0; i < number; i++) {
      amount += arr[i];
    }
    avg = amount / number;
    return avg;
  } else {
    if (arr[0] < arr[1]) {
      min = arr[0]; max = arr[1];
    }
    else {
      min = arr[1]; max = arr[0];
    }
    for (i = 2; i < number; i++) {
      if (arr[i] < min) {
        amount += min;      //arr<min
        min = arr[i];
      } else {
        if (arr[i] > max) {
          amount += max;  //arr>max
          max = arr[i];
        } else {
          amount += arr[i]; //min<=arr<=max
        }
      }//if
    }//for
    avg = (double)amount / (number - 2);
  }//if
  return avg;
}


void Growlight() {
  //three for-loops to give the LED the grow colors
  for (int i = 0; i < 60; i += 3) {
    LED's[i] = CRGB(255, 0, 0);
    FastLED.show();
  }
  for (int z = 1; z < 62; z += 3) {
    LED's[z] = CRGB(0, 0, 255);
    FastLED.show();
  }
  for (int y = 2; y < 63; y += 3) {
```

```
    LED's[y] = CRGB(255, 0, 180);
  }
  FastLED.show(); //send the selected colors to the LED so they get displayed
}

void TDSsensor() {
  /*
     The code below is part of the example provided by the manufacturer. It has been slightly adapted to suit our needs,
     but it is left mostly untouched as it serves as the proper calibration for the sensor
  */

  analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin); //read the analog value and store into the buffer
  analogBufferIndex++;
  if (analogBufferIndex == SCOUNT) {
    analogBufferIndex = 0;
  }

  for (copyIndex = 0; copyIndex < SCOUNT; copyIndex++)
    analogBufferTemp[copyIndex] = analogBuffer[copyIndex];
  averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) * (float)VREF / 1024.0; // read the analog value more stable by the median filtering algorithm, and convert to voltage value
  float compensationCoefficient = 1.0 + 0.02 * (temperature - 25.0);
//temperature compensation formula: fFinalResult(25^C) = fFinalResult(current)/(1.0+0.02*(fTP-25.0));
  float compensationVolatge = averageVoltage / compensationCoefficient;
//temperature compensation
  tdsValue = (133.42 * compensationVolatge * compensationVolatge * compensationVolatge - 255.86 * compensationVolatge * compensationVolatge + 857.39 * compensationVolatge) * 0.5; //convert voltage value to tds value
  Serial.print("TDS Value:");
  Serial.print(tdsValue, 0);
  Serial.println("ppm");
  TDSvalue = tdsValue;

}
int getMedianNum(int bArray[], int iFilterLen)
{
  int bTab[iFilterLen];
  for (byte i = 0; i < iFilterLen; i++)
    bTab[i] = bArray[i];
  int i, j, bTemp;
  for (j = 0; j < iFilterLen - 1; j++)
  {
```

```
    for (i = 0; i < iFilterLen - j - 1; i++)
    {
      if (bTab[i] > bTab[i + 1])
      {
        bTemp = bTab[i];
        bTab[i] = bTab[i + 1];
        bTab[i + 1] = bTemp;
      }
    }
  }
  if ((iFilterLen & 1) > 0)
    bTemp = bTab[(iFilterLen - 1) / 2];
  else
    bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
  return bTemp;
}

void Automation() {
  int L = analogRead(A3); //get the data from the LDR and store it in a local
variable
  if ((L <= 600) && ((hour() <= 17) && (hour() >= 8))) {  //only turn on the light
during the daytime, to keep the day/night rhythm of the plants
    Growlight();    //calling the grow light code
  }else{
     for (int e = 0; e < 60; e ++) {  //turn the light off if the above mentioned
values are not met
        LED's[e] = CRGB(0, 0, 0);
        FastLED.show();
     }

  }

  if ((hour() == 0) && (minute() == 0) && (second() > 2)) { //make sure the
system checks if it has to add nutrients once every day, at 12 o'clock at night
    TDSsensor();  //gather the data of the sensor
    if (TDSvalue <= 170) {
      analogWrite(motorPin, 120); //activate the pump which pumps the liquid
in
      delay(3000);
      analogWrite(motorPin, 0);
      delay(60000); //make sure the loop only repeats once a day
    }
  }
  if (bme.readTemperature() >= 23) {  //temperature warning loop, so
temperatures above 23 degrees are not allowed
    lcd.clear();
    lcd.setCursor(1, 0);
    lcd.print("Temperature is");
```

```
    lcd.setCursor(3, 1);
    lcd.print("Too high");

    for (int i = 0; i <= 9; i++) {  //use the light strip to give feedback to the
user
      for (int j = 0; j < 60; j ++) {
        LED's[j] = CRGB(255, 0, 0);
        FastLED.show();
      }
      delay(500);
      for (int e = 0; e < 60; e ++) {
        LED's[e] = CRGB(0, 0, 0);
        FastLED.show();
      }
      delay(500);
    }
  }
}

void digitalClockDisplay() {
  // digital clock display of the time in the serial monitor, for development
only
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}

void printDigits(int digits) {
  // utility function for digital clock display: prints preceding colon and
leading 0
  Serial.print(":");
  if (digits < 10)
    Serial.print('0');
  Serial.print(digits);
}
```