UNIVERSITEIT TWENTE.

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);
}
```