The branch «Nazarbayev Intellectual school of chemistry and biology» in Almaty of the autonomous educational organization «Nazarbayev Intellectual schools»

Computer Science Project

«Monitoring system of plant pollution»

Student of 12th grade: Balgabay Shyngysbek

Local Teacher: Mukanova Makpal

International Teacher: Antony Roy

Almaty, 2022

Table of Contents

Analysis

Introduction

This project represents an Arduino device that will help people to detect plant pollution causes. The device consists of many sensors such as MQ-5, optical dust sensor “GP1A57HRJ00F”, DHT11, soil humidity sensor, and others that will identify and send you data of air pollution, temperature, air humidity, dust density in the atmosphere, and soil humidity by drawn graphs in order to make it easier to analyze for human eyes. This project is relevant because the pollution of plants can lead to the reduction of leaf life, the decrease in the growth and productivity of plants, the date of bloom shifts, and foliage falling off prematurely. That is why this device should be used to help the government or scientists to find out and eliminate the plant pollution causes.

Description of stakeholder

The customer[1] of this coursework is a teacher-moderator of biology, Bessenbayeva Gulmira Sembaevna, who works in Nazarbayev Intellectual school of Chemistry and Biology in Almaty. She was born on 18 February 1977 and has been working in Nazarbayev Intellectual school since 2015. Starting from the entrance to the school, she has been teaching for the students of 7 to 9 grades and has gained a huge experience in education.

Organization

The organization of the project is a Nazarbayev Intellectual school of chemistry and biology direction. It was founded on the 11th of August in 2015 in Almaty. There are about 960 pupils studying and 140 teachers working at school. The mission of the school is to contribute to the increase of the intellectual potential of Kazakhstan by providing academic excellence in teaching, research, and service and the development of the whole person in all these endeavors.

Full interview

Me: Good evening, Mrs. Gulmira! Tell me about yourself, please.

Customer: Hello! My full name is Besenbaeva Gulmira Sembaevna. I am a teacher-moderator of Nazarbayev Intellectual school. I have been working there since 2015.

Me: Could you say why the problem of plant pollution is relevant?

[1] A person who has ordered something from the entrepreneur or organization.

Customer: Of course! Nowadays, cars and factories are producing a lot of traffic fumes that settle on the leaves and ground. This changes the composition of air, and when it rains outside, harmful gases are absorbed by soil which poisons flora[2]. Actually, not only harmful gases but the level of humidity, temperature, the amount of dust in the air also has a bad impact on the plants. This can lead to the reduction of leaf life, the decrease in the growth and productivity of plants, the date of bloom shifts, and foliage falling off prematurely. Consequently, people will get ill very often and new poisonings or diseases may appear. That is why we need a device that could prevent it.

Me: What data do you want to have in your database to monitor the pollution of plants?

Customer: It will be better if I will be able to see the humidity of the soil, dust density, air humidity, and the presence of harmful gases and specks of dust in the atmosphere. Also, I need these data from all the regions in Almaty, for example beginning from Kok tobe and ending in Kalkaman to define which region is the clearest. Furthermore, we can detect in which season which region has the worst pollution. These are all things to need for analyzing the pollution of plants. I would be glad if you could create for me a device with which I can define the pollution data only by pointing at plants.

Me: How the data will be used?

Customer: The collection of data will be used for finding solutions for preventing pollution and for new programs such as “Green city”. We can show the statistic of pollution and offer different solutions to the akimat[3] of Almaty. For example, combined heat and power (CHP) systems would be using gases instead of solid fuel if this technology had been invented already. I believe that there are a lot of solutions and we should just use them.

Me: Thank you for answering my questions! Goodbye!

Customer: Thank you! Good luck!

[2] Representation of all plants in the exact region.
[3] City administration

Analysis stage

Origin of data

Picture 1. Checking the acidity of soil by litmus paper or by acidity meter.

Picture 2. Measuring the amount of harmful gases by a gas analyzer.

Nowadays, many things are needed to be measured and analyzed. People are still using litmus paper[4] or large devices such as a gas analyzer, acidity meter, thermometer, and hygrometer to determine the acidity of the soil, quality of air, humidity, and temperature. Furthermore, people collect data by themselves and sometimes in handwritten form in notepad or in electronic form; for example, in Microsoft Excel. All of these take hard work and people spend a lot of time filling tables and drawing graphs manually. To cut a long story short, the main problems are in the lack of efficiency, autonomy, and multitasking.

Requirement Specification

- ✓ Creating a comfortable device, which is easy to use.
- ✓ Giving data of soil humidity, temperature, air humidity, and the presence of harmful gases and dust in the atmosphere.
- ✓ Drawing graphs that the changes of data can be easily analyzed
- ✓ Creating a monitoring system that can record the changes in environment values

Alternative solutions

There are several alternative solutions for this problem.

The first solution represents that the device sends its data to a website where all scientists can register and analyze the pollution causes. Residents also

[4] A paper which is used to identify the acidity level of the fluid

will have access to the website in order to be able to identify which city is better to live in and how the environment is polluted.

The second solution is to create a mobile app. It will have the same functions as the website option. But it can be more comfortable sometimes in monitoring the graphs on your phone. As the last solution, the Arduino project itself could collect and store data in its memory. The benefit of the last option is that there will be no problems with software because you will watch the data on the display of the device.

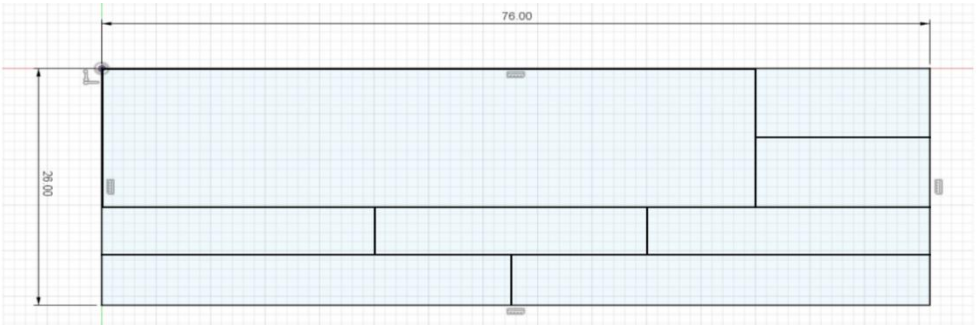
Design stage

Design

SMART objectives

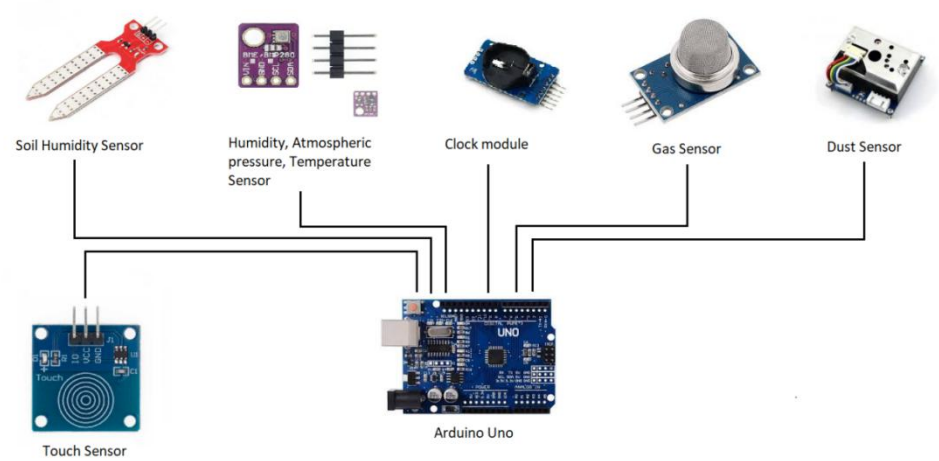
1) Create a little device that is comfortable to carry
2) Collect air pollution, soil humidity, dust density, temperature, and air humidity data
3) Make graphs for each data type for convenience
4) Draw a user-friendly interface
5) Make easy management and control of the device

Prototyping



Picture 3. Prototype of the main window of “Monitoring system of plant pollution”.

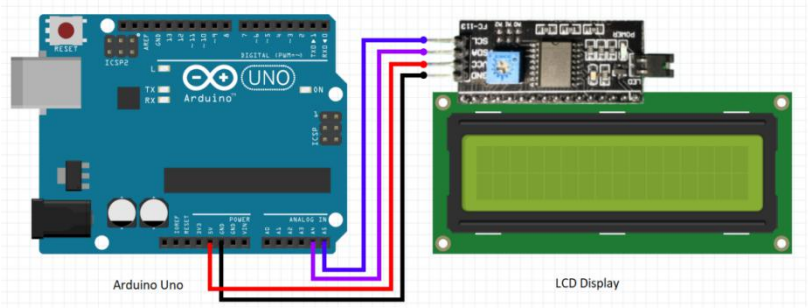
The main menu will be divided into 8 parts. These parts will contain time, date, dust level, soil humidity, atmospheric pressure, temperature, harmful gases, and soil acidity data. (Picture 3)



Picture 4. Prototype of input form of “Monitoring system of plant pollution”.

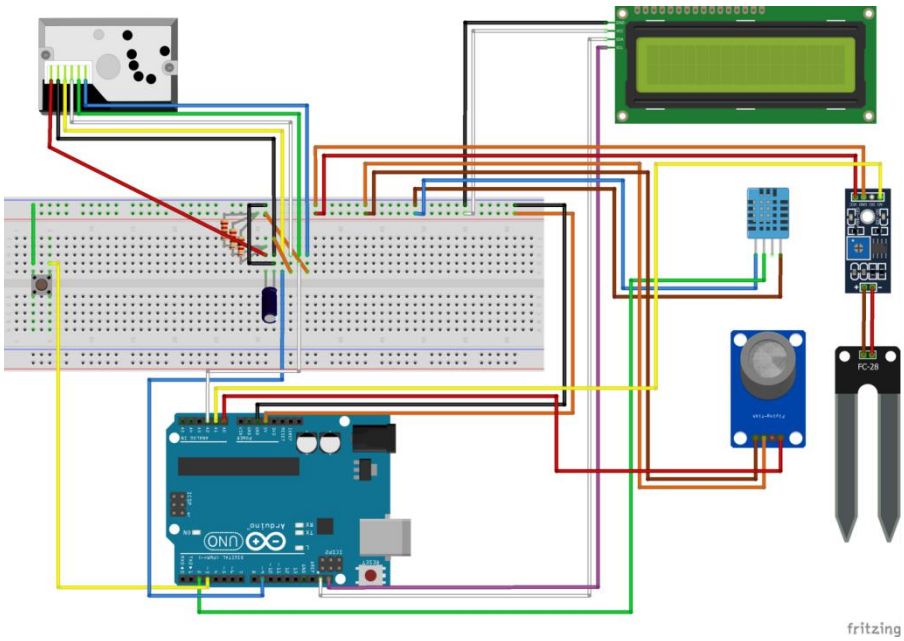
In the input section “Soil Humidity”, “Humidity, atmospheric, temperature sensor”, “Clock module”, “Gas Sensor”, “Dust Sensor”, “Touch Sensor” devices

will transfer the collected data from the environment to the “Arduino Uno”. (Picture 4)



Picture 5. Prototype of output form of “Monitoring system of plant pollution”.

In the output section “Arduino Uno” will compute and transfer the collected data from the sensors to LCD Display where clients can read it. (Picture 5) The LCD Display will show the temperature, soil humidity, air humidity, amount of dust in the atmosphere, amount of harmful gases of the environment by graphs and numbers.



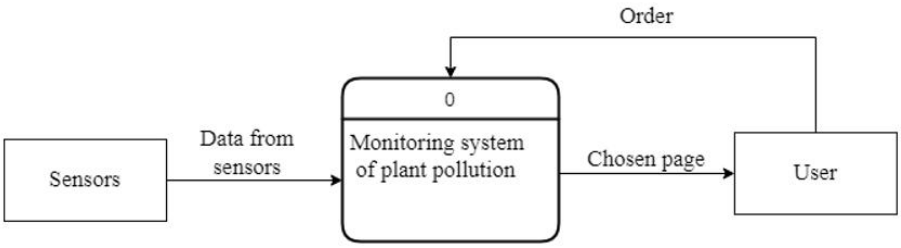
Picture 6. Project scheme of “Monitoring system of plant pollution”.

In picture 6, it is clearly seen how the sensors and Arduino Uno are connected with each other by using cables, components, and a breadboard.

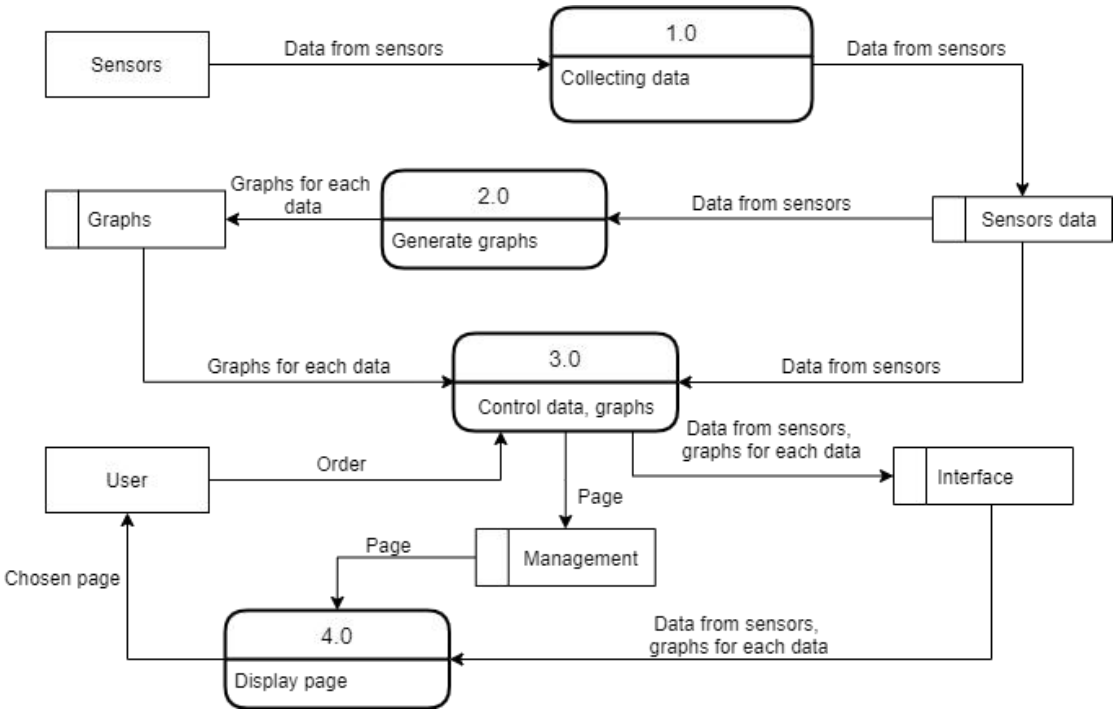
Data Flow Diagram

A level 1 and level 0 DFD is shown below. They clearly represent how the data is used and transported between object, processes and database.

Design stage



Picture 7. DFD level 0

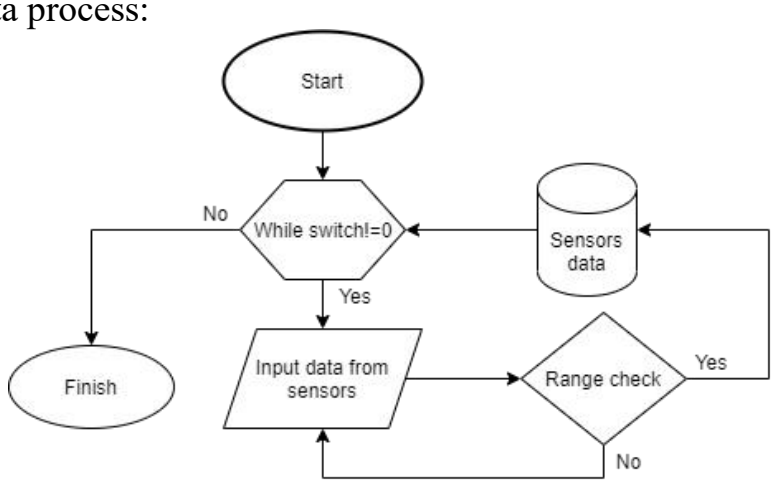


Picture 8. DFD level 1

Flowcharts

The flowcharts below clearly represent how the system's processes work.

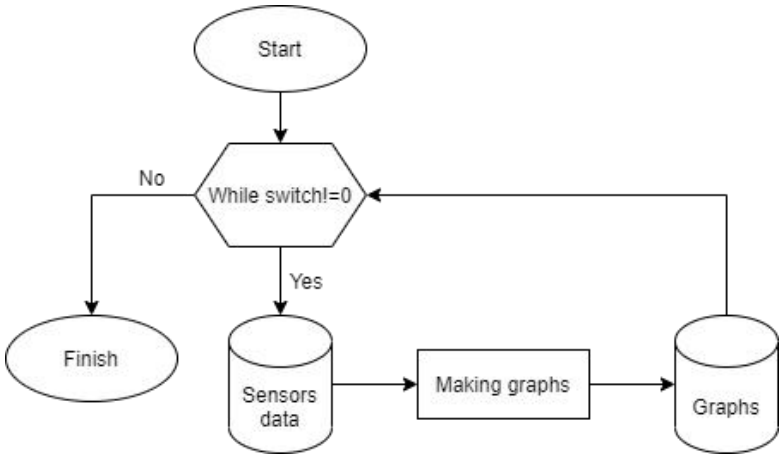
Collecting data process:



Picture 9. Flowchart "Collection of Data"

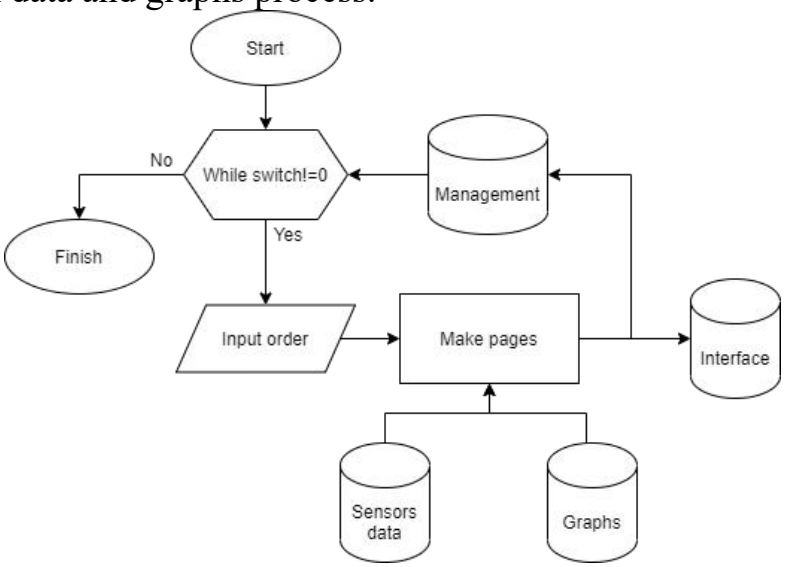
Design stage

Generating graphs process:



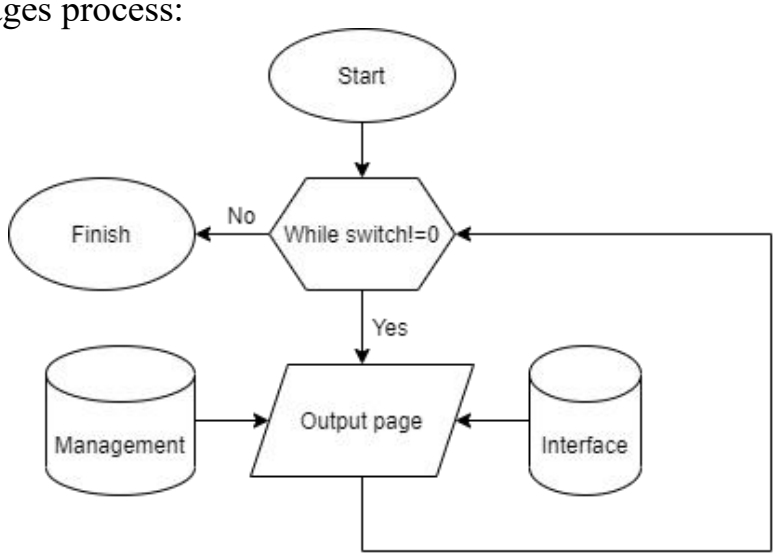
Picture 10. Flowchart “Generation of graphs”

Controlling of data and graphs process:



Picture 11. Flowchart “Data management”

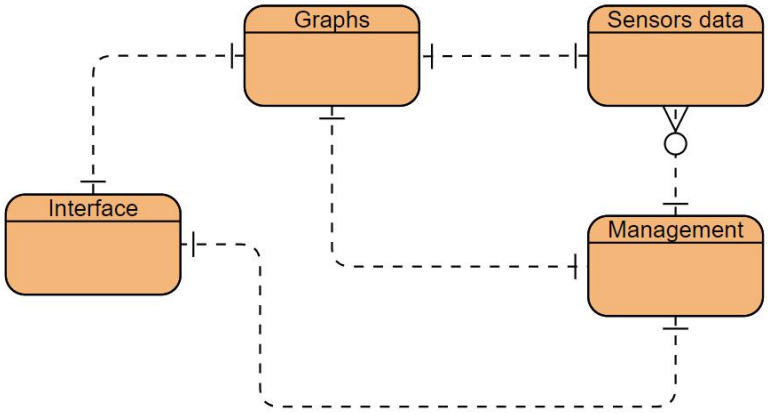
Displaying pages process:



Picture 12. Flowchart “Page display”

Entity Relationship Diagram

An ERD is shown below. It clearly represents the relationship between the entities. Only one graph and many data of sensors can be stored in "Management". "Graphs" use many data from sensors. Also, "Interface" contains one graph and one management respectively.



Picture 13. ERD

Data Dictionary

Table 1 - Sensors data

Field Name	Data Type	Field length	Validation	Description	Example
Dust_level	double	7	Range check, >=0.00 and <=1000.00	The data of air pollution	0.17
Soil_humidity	double	6	Range check, >=0 .00 and <=100.00	The data of soil acidity	13.00
Temperature	double	5	Range check, >=-10.00 and <=50.00	The data of temperature	24.56
Harmful_gases	double	7	Range check, >=0.00 and <=1000.00	The amount of harmful gases	250.00
Air_humidity	double	6	Range check, >=0.00 and <=100.00	The data of soil acidity	45.32

Design stage

Table 2 - Graphs

Field Name	Data Type	Field length	Validation	Description	Example
GraphID(Primary key)	int	1	Type check, the whole number	Unique identifier of each graph	1
Max data	float	7	Range check, >=0 and <=1000.00	The max data for the period	30.32
Min data	float	7	Range check, >=0 and <=1000.00	The min data for the period	22.89
PageID(Foreign key)	int	1	Type check, the whole number	Unique identifier of each page	1

Table 3 - Interface

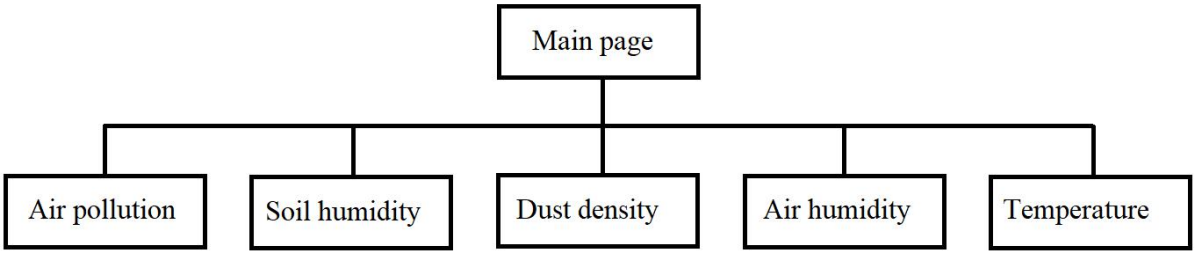
Field Name	Data Type	Field length	Validation	Description	Example
PageID(Foreign key)	int	1	Type check, the whole number	Unique identifier of each page	3
GraphID(Foreign key)	int	1	Type check, the whole number	Unique identifier of each graph	3

Table 4 - Management

Field Name	Data Type	Field length	Validation	Description	Example
PageID(Primary key)	int	1	Type check, the whole number	Unique identifier of each page	1
Order	int	1	Type check, the whole number, Range check, >=1 and <=5	An order of user	3

Sitemap

Below is shown a sitemap of the “Monitoring system of plant pollution” project. It represents how pages are connected with each other.



Picture 14. Sitemap

Each page is referred only to one data type. For instance, as it is mentioned in the picture, the “Dust density” page will have only dust density values on its page. Moreover, each page will have its own graph made of 15 columns, page name, and arithmetic mean data.

Description of Processes/Modules

Collecting data: A lot of data are received by Arduino Uno from several sensors, such as soil humidity sensor, DHT11, dust sensor “GP1A57HRJ00F”, MQ-5, and button.

Generating graphs: Once data from sensors have been received, graphs are made for each type of data for a period of time set by the client.

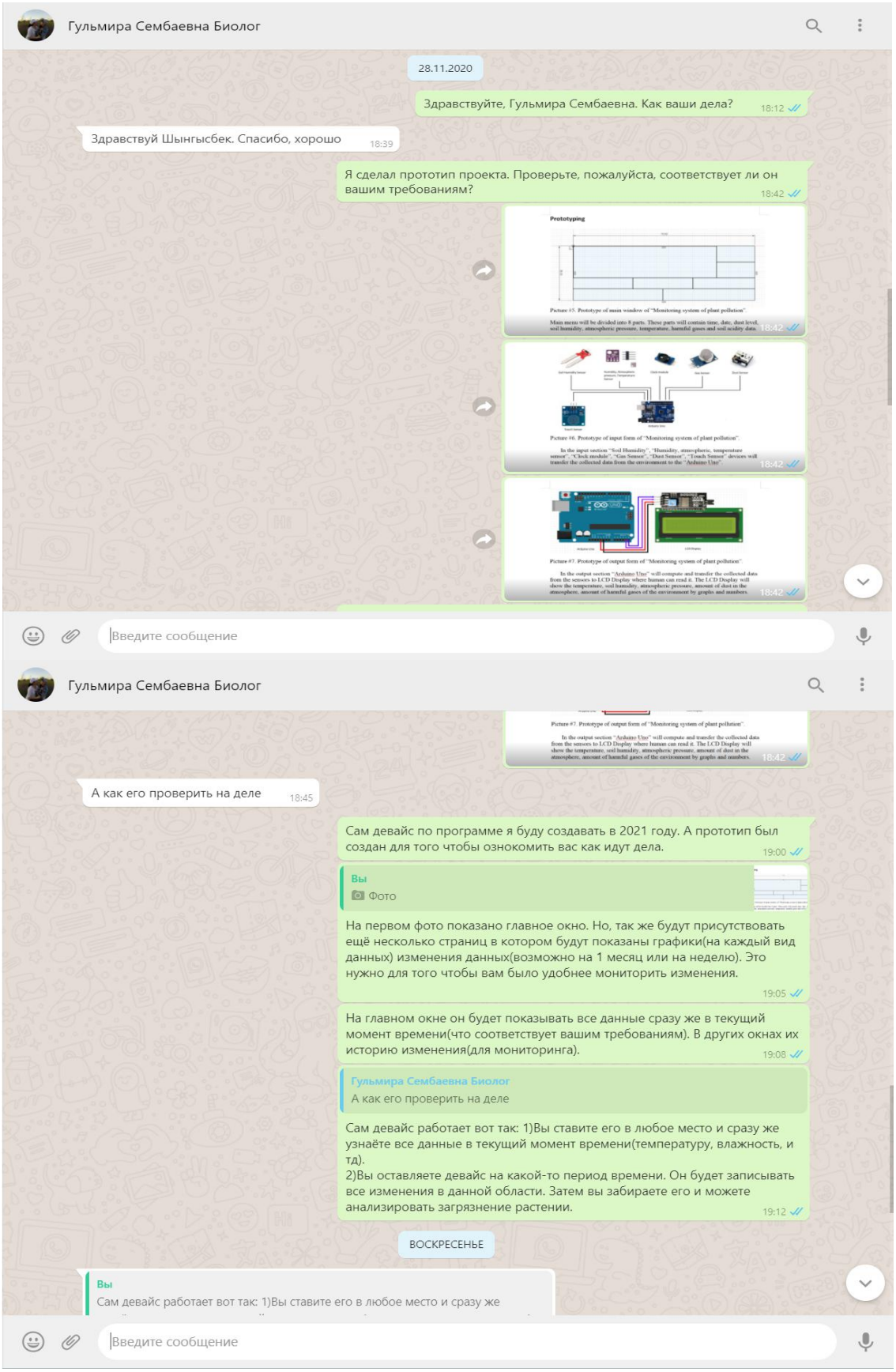
Controlling of data and graphs: The process matches the order of user, graphs, and data.

Display pages: The chosen page that contains data or graphs are displayed on the screen.

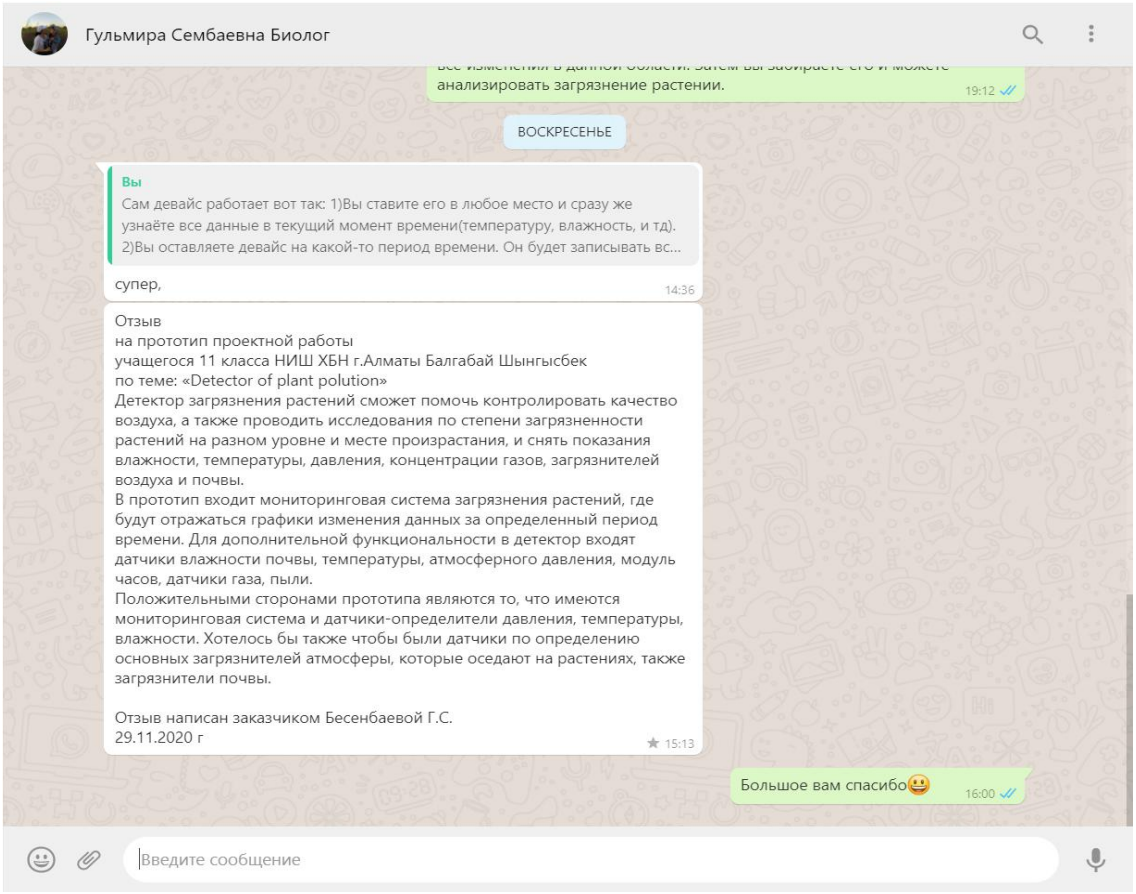
Client Feedback

My client’s approval and our dialogue are shown below:

Design stage



Design stage



Picture 15. Client's approval

Requirement Specification

Table 5 - Hardware requirements

Device	Characteristics	Purpose of using
Laptop	HDD 1TB, SSD 256GB Generic PnP Monitor Intel UHD Graphics 630, NVIDIA GeForce GTX 1050 Ti Intel Core i7-8750H CPU @ 2.20GHz	For programming and using software programs.
Arduino project	Arduino NANO 3.0 WAVGAT Soil Humidity Sensor Si Tai&SH LCD2004 TENSTAR ROBOT DS3231 TENSTAR ROBOT TTP223 TENSTAR ROBOT MQ-2 GP2Y0A21YK0F sincere&promise	For making the coursework project. They allow to gain data from the environment in order to be analyzed.

Design stage

Table 6 - Software requirements

Software	Purpose of using
Windows 10 Home	For managing other software
Arduino IDE	For programming Arduino UNO
Google Chrome	For searching information on the Internet and making some evidence or images for the coursework(ERD, DFD, Sitemap).
Paint	For making and editing images for the coursework(ERD, DFD, Sitemap, I/O Prototype form)
Fritzing	For making a scheme for prototype form

Intended benefits

Fast speed of researches - the new system gives all the information about the environment where it is stayed at once. This makes researching processes faster.

Requires less energy - this means that in the long term researches user doesn't need to check the research place every time. This is because all data changes are stored in the new system to be analyzed in the future.

The efficiency of analyzing - all the information of the environment where the device has stayed are stored and then graphs are made for each data, which makes the analyzing part more efficient for the user.

Universality - the new system allows the user to gain different types of information about the environment such as dust level, soil humidity, air humidity, temperature, and harmful gases.

Comfortability of using - the new system is represented as a little box, which is comfortable to move and consists of several technologies of monitoring the environment.

Limitation

Data storage limit - the device can work only for a maximum of 1 month, because of memory limitation.

Lack of accuracy on graphs - the graphs which were drawn by the device have only 15 of accuracy as there are only 16 columns on the screen. Another column is used for naming the pages.

Battery necessity - the new system needs to be recharged by exchanging batteries.

An insufficient amount of environmental values - the device can gain only 5 different values, as the purchase of other sensors is expensive.

Age limitation - only the person who is over 16 years old can use the device because they have completed almost 8-9 years of education at school. This means that they have enough knowledge to use the device.

Design stage

Estimation of database size

Table 7 - Distribution of the memory

Name of field	Data type	Maximum size(in bytes)
Dust level	double	7
Soil humidity	double	6
Temperature	double	5
Harmful gases	double	7
Air humidity	double	6

Approximate number of average values of one type of data per month - 15

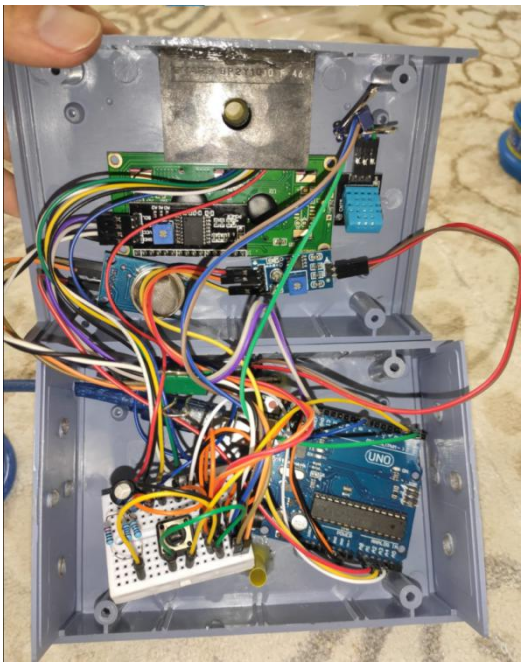
Total approximate number of values including all data types = 15 * 5=75

Total size of the database in bytes per month = 75 * 31 = 2325

Development and Programming

Completeness of solution

Below is shown a code written on Arduino IDE for the device named “Monitoring system of plant pollution”. As it is seen in the code, comments, and functions are used in the system, which makes the program work correctly and convenient for further amendments. Moreover, the code contains a two-dimensional array, nested loops, recursion, parallel programming on a single-core CPU, and other complex algorithms. Also, the photo of the scheme the in real-world is provided below; the digital form of the scheme can be seen in the Prototype section.



Picture 16. The real scheme of the project

sketch_MSPP0_v9.0

```
//LiquidCrystal_I2C 1602 libraries
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h> // Using version 1.2.1 for LiquidCrystal_I2C 1602
/*(address,  En, Rw, Rs, d4, d5, d6, d7, backlighPin, backlighPol pol)
initializing the LiquidCrystal_I2C 1602*/
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

//DHT11 (humidity and temperature sensor) libraries
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#define DHTPIN 2 // DHT11 sensor pin 2
DHT dht(DHTPIN, DHT11); //Initializing DHT11

//Gas detector MQ-5
#define AIn_gas A0 // Gas detector MQ-5 pin A0
```

Development and Programming Stage

```
//Dust sensor
int dustPin = 2; // Dust sensor - Arduino A2 pin
int ledPin = 9; // Led pin 9

// Variables for calculating the value of dust in air
float voltsMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;

//Soil humidity
#define AIn_soil A1 //Soil humidity sensor pin A1

//Variables for button functions and calculations
volatile int countInterrupt = 0;
volatile bool flag = 1;
volatile unsigned long timerPrev = 0;
```

```
/*Defining a variable for making a time management for the sensors
(used for calculation of an arithmetic mean)*/
int timer = 0;
/*Defining variables of arithmetic mean for each sensor*/
double arif_hum = 0, arif_gas = 0, arif_soil = 0, arif_dust = 0, arif_temp, arith = 0;
/*Defining a 2D array, which will contain arithmetic mean values of all sensors.
(Rows: 0-humidity, 1-temperature, 2-toxic gases, 3-soil humidity, 4-dust density)*/
double arr_data[5][15];


/*Creating characters for drawing graphs
8, 7, 6, 5, 4, 3, 2, 1 lines*/
byte p00[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00100};
byte p10[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111};
byte p20[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111, 0b11111};
byte p40[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111, 0b11111, 0b11111};
byte p50[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b11111, 0b11111, 0b11111, 0b11111};
```

```
byte p60[8] = {0b00000, 0b00000, 0b00000, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte p70[8] = {0b00000, 0b00000, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte p90[8] = {0b00000, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte p100[8] = {0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};

void setup()
{
  //Initializing pin modes for each sensor
  pinMode(AIn_gas, INPUT);//Humidity sensor
  pinMode(ledPin, OUTPUT); //Dust sensor
  pinMode(AIn_soil, INPUT);//Soil humidity
  pinMode(13, OUTPUT);//built-in LED
  Serial.begin(9600);//Setting the data rate in bits per second

  pinMode(3, INPUT_PULLUP);//Button pin 3
  //Creating a priority function of interruption for the button (pin 3)
  attachInterrupt(1, button, FALLING);
```

```
  dht.begin();//Launching the DHT11 sensor

  lcd.begin(16, 2); // sixteen characters across - 2 lines
  //initializing each character for making graphs
  lcd.createChar(0, p00);
  lcd.createChar(1, p10);
  lcd.createChar(2, p20);
  lcd.createChar(3, p40);
  lcd.createChar(4, p50);
  lcd.createChar(5, p60);
  lcd.createChar(6, p70);
  lcd.createChar(7, p90);
  lcd.createChar(8, p100);
  lcd.clear();
}
```

Development and Programming Stage

```
void loop()
{
  //A nested loop for 2D array
  for (int r = 0; r < 15; r++) { //Outer loop
    for (int j = 0; j < 5; j++) { //Inner loop
      arr_data[j][r] = -1; //Filling the 2D array by "-1" as a default value
    }
  }
  /*taking fifteen arithmetic mean values from each sensor (Outer loop)*/
  for (int i = 0; i < 15; i++) {
    //updating arithmetic mean values to 0.
    arif_hum = 0;
    arif_temp = 0;
    arif_gas = 0;
    arif_soil = 0;
    arif_dust = 0;
    /*a loop for finding an arithmetic mean every 4 senonds (Innter loop)*/
    for (int j = 0; j < 4; j++) {
      delay(1000);//a delay for 1 second
      timer++;;//a timer counting every second
      digitalWrite(ledPin, LOW); // power on the LED
      delayMicroseconds(280);//Delay in microseconds(280)

      voltsMeasured = analogRead(dustPin); // Reading the dust sensor value

      delayMicroseconds(40);
      digitalWrite(ledPin, HIGH); // turn the LED off
      delayMicroseconds(9680);

      //calculating dust density
      calcVoltage = voltsMeasured * (5.0 / 1024.0);
      dustDensity = abs(170 * calcVoltage - 0.1);//unit: ug/m3

      //taking the sum of values
      arif_hum += dht.readHumidity(); //in %
      arif_temp += dht.readTemperature(); //in *C
      arif_gas += analogRead(AIn_gas); //in ppm
      arif_soil += (1000 - int(analogRead(AIn_soil))) / 10; //in %
      arif_dust += dustDensity; //in mg/m3

      if (timer % 4 == 0) { //making values a single number for graphs
        arif_hum = arif_hum / 4 / 10;
        arif_temp = arif_temp / 4 / 10;
        arif_gas = arif_gas / 4 / 100;
        arif_soil = arif_soil / 4 / 10;
        arif_dust = arif_dust / 4 / 80;
        arr_data[0][i] = arif_hum;
        arr_data[1][i] = arif_temp;
        arr_data[2][i] = arif_gas;
        arr_data[3][i] = arif_soil;
        arr_data[4][i] = arif_dust;
      }
      cli();//stops attachInterrupt
      int countButton;//Defining a variable for counting button presses(changing pages on display)
      if (countInterrupt <= 5) { //A loop of counting
        countButton = countInterrupt;
      }
      if (countInterrupt == 6) { //limitation of the maximum number of button presses
        countInterrupt = 0;
      }
      sei();//continuous attachInterrupt
      Serial.println(countButton);//printing the number of button presses in a Serial Monitor

      if (countButton == 0) { //choosing the pages according to the number of button presses
        lcd.clear();//clearing the LCD display
        lcd.setCursor(0, 0); //setting cursor to a specific place on the LCD
        lcd.print(int(dht.readHumidity())); lcd.print("%H");//Humidity (%)
        lcd.setCursor(0, 1);
```

```
        lcd.print(int(dht.readTemperature())); lcd.print("T");//Temperature (*C)
        lcd.setCursor(10, 1);
        lcd.print(int(analogRead(AIn_gas))); lcd.print("G");//Toxic gases (ppm)
        lcd.setCursor(5, 0);
        lcd.print(int(dustDensity)); lcd.print("D");//Dust density (ug/m3)
        lcd.setCursor(5, 1);
        lcd.print((1000 - int(analogRead(AIn_soil))) / 10); lcd.print("%S"); //Soil humidity (%)
      }
      else {
        drawGraph(countButton);//turning to another page and drawing graph for a specific sensor
      }
    }
  }
}

void button() { //a function for identifying button presses
  if ((millis() - timerPrev) >= 200) { //Avoiding the bounce of contacts
    flag = 1;
    timerPrev = millis(); //returns the number of milliseconds since the sketch started running
  }
  if (flag == 1) {
    countInterrupt++;
    flag = 0;
  }
}

/*a function for drawing graphs and displaying an arithmetic mean*/
void drawGraph(int countButton) {
  lcd.clear();//clearing the LCD display
  lcd.setCursor(0, 0); //setting cursor to a specific place on the LCD
  lcd.print(countButton);//Displaying a page number
  lcd.setCursor(0, 1);
  switch (countButton) {
    /* Displaying an index for each type of value
    1-humidity, 2-temperature, 3-toxic gases, 4-soil humidity, 5-dust density*/
    case 1: lcd.print("H"); break;
    case 2: lcd.print("T"); break;
    case 3: lcd.print("G"); break;
    case 4: lcd.print("S"); break;
    case 5: lcd.print("D"); break;
  }
  lcd.setCursor(2, 0);
  lcd.print("AM=");//abbreviation of "Arithmetic mean"
  switch (countButton) { //Displaying a dynamic arithmetical mean data
    case 1: lcd.print(arithFunction(0, countButton - 1) * 10); //humidity
      lcd.print("%"); break;
    case 2: lcd.print(arithFunction(0, countButton - 1) * 10); //temperature
      lcd.print("*C"); break;
    case 3: lcd.print(arithFunction(0, countButton - 1) * 100); //toxic gases
      lcd.print("ppm"); break;
    case 4: lcd.print(arithFunction(0, countButton - 1) * 10); //soil humidity
      lcd.print("%"); break;
    case 5: lcd.print(arithFunction(0, countButton - 1) * 80); //dust density
      lcd.print("ug/m3"); break;
    default: break;
  }
  arith = 0; //updating an arithmetic mean value to 0.
  for (int t = 1; t < 16; t++) { //displaying graph for a specific sensor
    lcd.setCursor(t, 1);
    /*taking a round value from the 2D array for displaying
    the necessary character (making a graph)*/
    switch (round(arr_data[countButton - 1][t - 1])) {
      case 0: lcd.write(byte(1)); break;
      case 1: lcd.write(byte(1)); break;
      case 2: lcd.write(byte(2)); break;
      case 3: lcd.write(byte(3)); break;
      case 4: lcd.write(byte(3)); break;
      case 5: lcd.write(byte(4)); break;
```

```
      case 6: lcd.write(byte(5)); break;
      case 7: lcd.write(byte(6)); break;
      case 8: lcd.write(byte(7)); break;
      case 9: lcd.write(byte(7)); break;
      case 10: lcd.write(byte(8)); break;
      default: break;
    }
  }
}
/*a recursive function for calculating a dynamic arithmetic mean data*/
double arithFunction(int startPoint, int rowNum) {
  if (startPoint > 14 || arr_data[rowNum][startPoint] == -1) {
    return arith / startPoint; //Returning an arithmetic value
  }
  arith += arr_data[rowNum][startPoint]; //Calculating the sum of values
  /*turning to the next index of the 2D array for a specific row,
   and calling this(arithFunction) function again*/
  arithFunction(startPoint + 1, rowNum);
}
```

Picture 17. The code of “Monitoring System of plant pollution” device

Testing

For the tests of the "Monitoring system of plant pollution" device, I decided to use an alpha testing type because it allows to identify all possible issues and bugs[5] before its release to the client. I will perform as a programmer and a real user in order to identify the behavior of the device. Also, white-box, black-box, and gray-box testing methods will be used for checking the right work of the system.

Test plan

Table 8 - Testing plans

Test Number	Test Objectives	Test Method	Test data used	Expected Outcome	Pass /Fail	Evidence page
1	The password consists of only numbers	Make 8 presses on the button	Normal 12345678	The device messages "Created!"	Pass	23 page
2	The length of the password is 8	Make more than 8 presses on the button	Erroneous I have typed 9 digits: 112223654	The device will accept only the first 8 numbers	Pass	25 page
3	The correct work of password security	Creating a new password and entering the exact password	Typical New password: 12345678 Login password: 12345678	The device messages "The password is correct!"	Pass	27 page
4	After an erroneous turn-off of the device, the database is saved.	The device is run and after some time it is turned off. Then it is launched	Erroneous I have powered off the device without waiting for its end	The device loses the gained data and could not continue filling its database even after the	Pass	31 page

[5] An unusual behavior of a program which is considered as a program fault

Testing stage

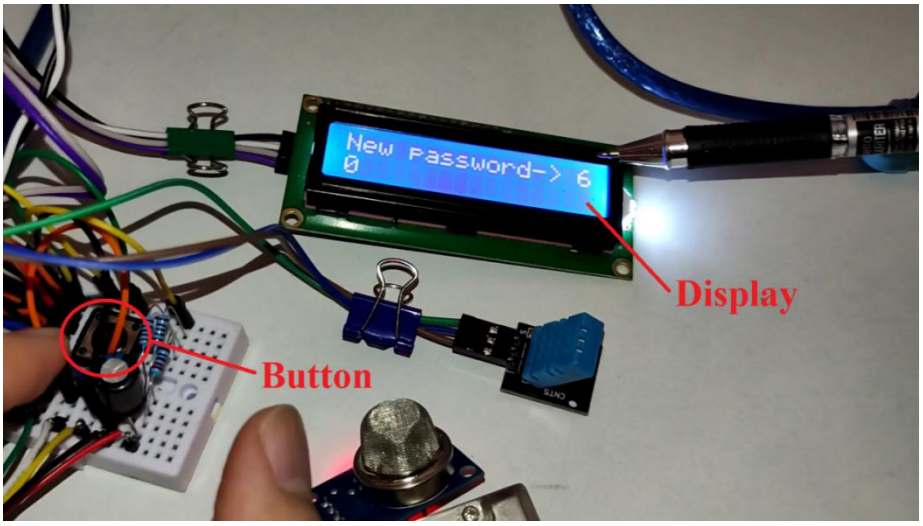
		again	of work and turned it on again	erroneous turn-off		
Test Number	**Test Objectives**	**Test Method**	**Test data used**	**Expected Outcome**	**Pass /Fail**	**Evidence page**
5	Checking the work of all sensors in normal conditions	The device is set on the home flower's soil and turned on	Normal condition.	The device shows all data types from each sensor (atmosphere temperature, humidity, dust density, soil humidity, detector of harmful gases)	Pass	33 page
6	The soil humidity sensor shows correct data only when it is set in soil	The device is turned on when the soil humidity sensor is positioned in the atmosphere and in water.	Extreme condition	The device shows non-real data as the soil humidity sensor should only be used in soil	Pass	34 page
7	Checking if the system works when it is charged by power bank	The power bank charges the device.	Normal The output of power the bank is 5V, 2.0A	The device works correctly without any crashes	Pass	37 page
8	Checking if the system	The power bank	Normal The output of	The device works correctly	Pass	38 page

Testing stage

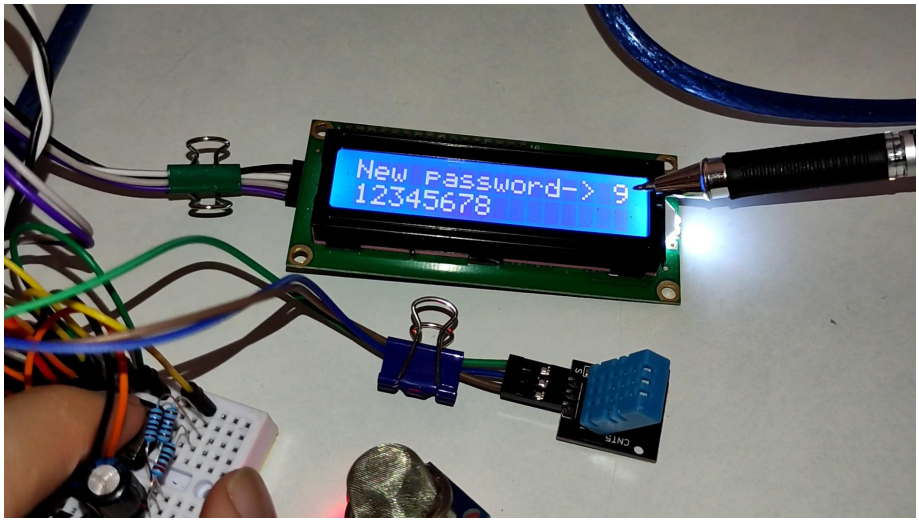
	works when it is charged by power bank	charges the device	the power bank is 5V, 1.0A	without any crashes		
9	Checking if the arithmetic mean recursive function works correctly on harmful gases detector sensor	Running the device in different conditions	Normal A lighter which contains butane is used to change the input value of harmful gases detector	With the changes in the input data of harmful gases detector sensor, the arithmetic mean value changes too	Pass	39 page
10	Checking if the graph changes based on the values of the humidity sensor.	The device is turned on and the input data is changed after a little amount of time has passed while its work	Normal I have breathed on the sensor to change its input value.	With the changes in the input data of the humidity sensor, the graph changes too	Pass	42 page

Testing stage

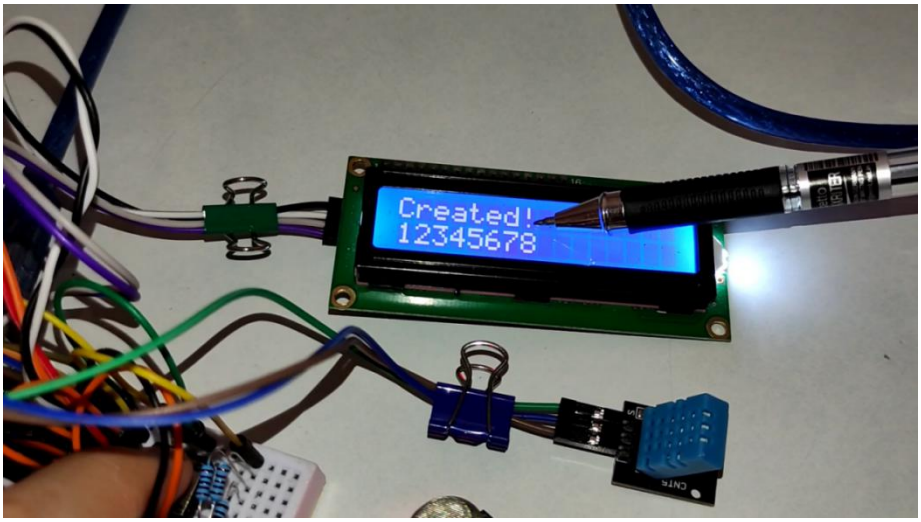
Test 1



Picture 18. Display and Button



Picture 19. Security system

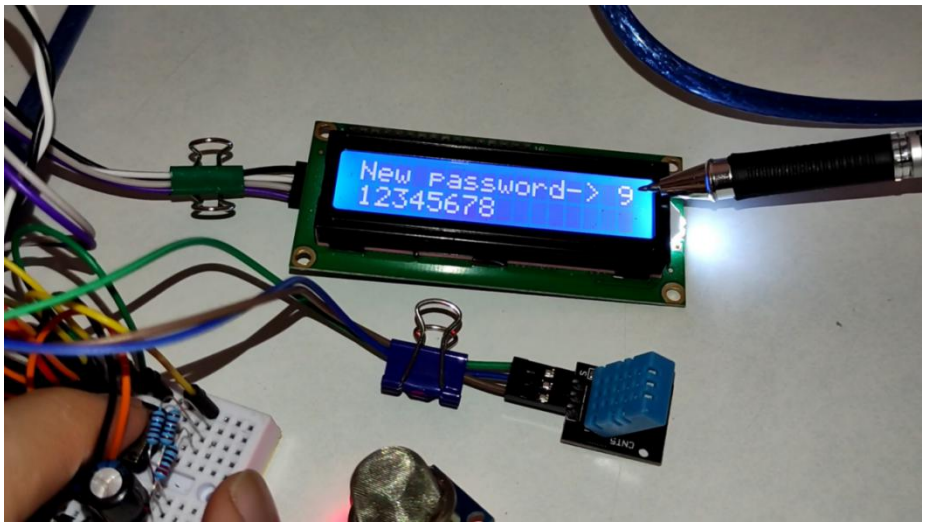


Picture 20. Creation of password

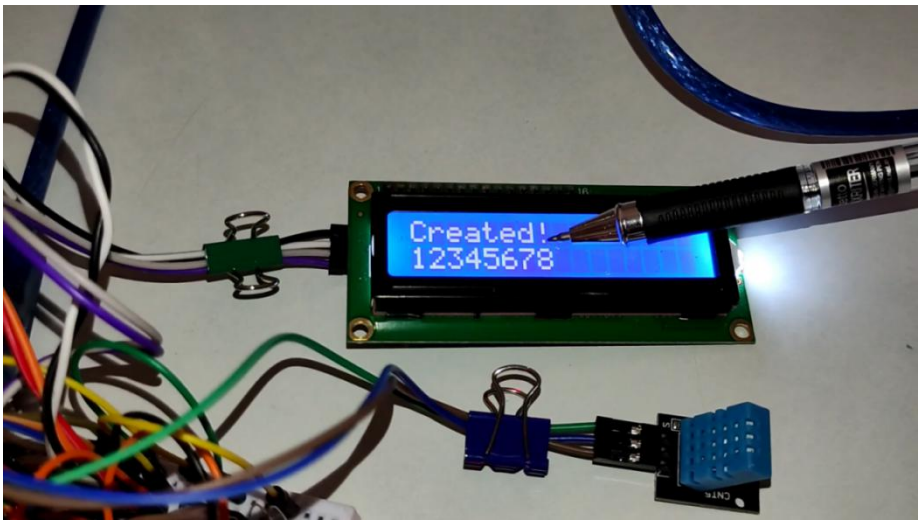
Testing stage

The value, which is shown by the pen, on the display is changed almost half a second. (Picture 18) When the user pushes the button, the system types the same number that is shown on the display. I have typed 8 values which are all digits because only digits are allowed to be entered (Picture 19). As it is seen in Picture 20, the system messages “Created!”, once 8 values have been typed into the system. In this test, only black-box testing methods was used.

Test 2



Picture 21. 9 pushes on the button



Picture 22. The work of password length limitation

Testing stage

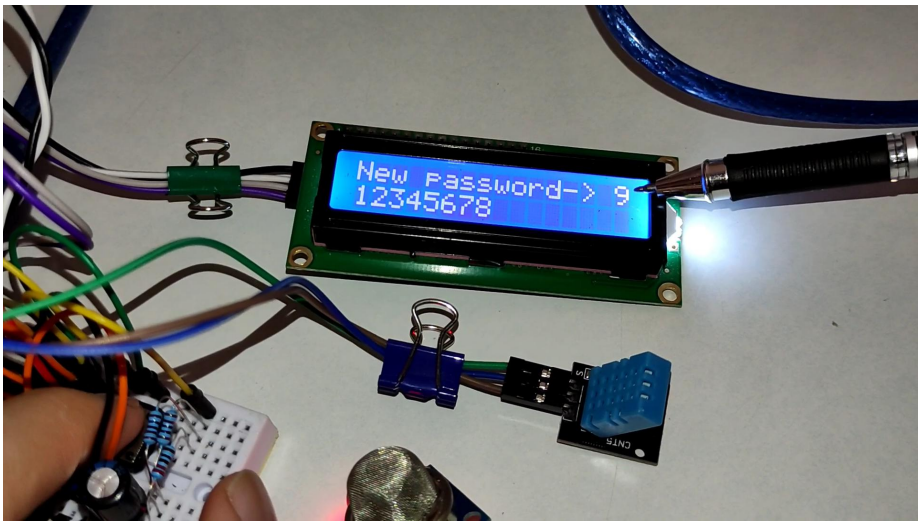
```
countButton=0;
while(password<=9999999){
for(int i=0;i<=9;i++){
    lcd.clear();//clearing the LCD display
    lcd.setCursor(0, 0);
    lcd.print("New password->");
    lcd.setCursor(15, 0); //setting cursor to a specific place on the LCD
    lcd.print(i);
    lcd.setCursor(0, 1);
    lcd.print(password);
    delay(700);
    if(password>=9999999)
      break;
    }
    cli();//stops the work of attachInterrupt
    //int countButton;//Defining a variable for counting button presses
    countButton = countInterrupt;
    if(numStorage!=countButton){
      numStorage=countInterrupt;
      password=(password*10)+i;
      Serial.println(i);
      Serial.println(password);
    }
    sei();//continues the work of attachInterrupt
  }
}
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Created!");
```

limitation of length

Picture 23. The code of the password length limitation

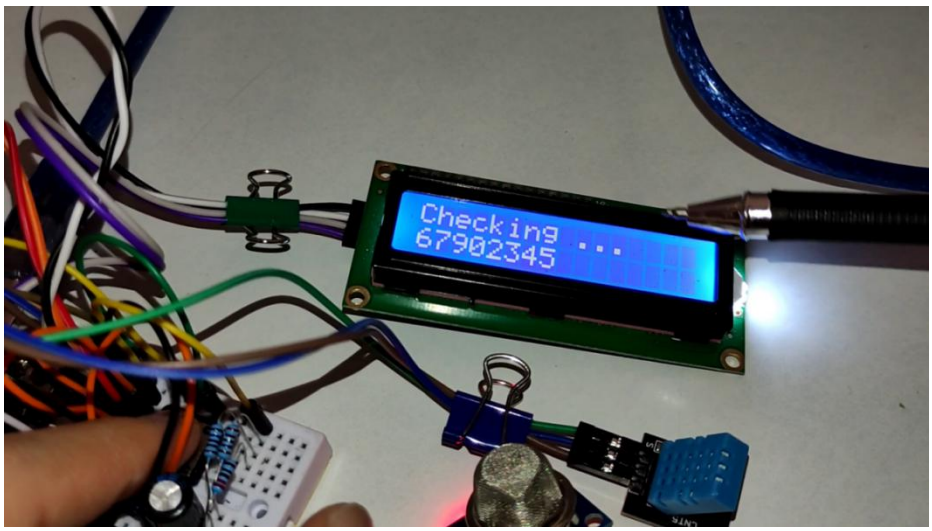
I tried to push the button 9 times as seen in picture 21, however, the system caught only the first 8 digits. (Picture 22) This is because a limitation of password length is set into the code as seen in picture 23. Gray-box testing methods was used for this test.

Test 3

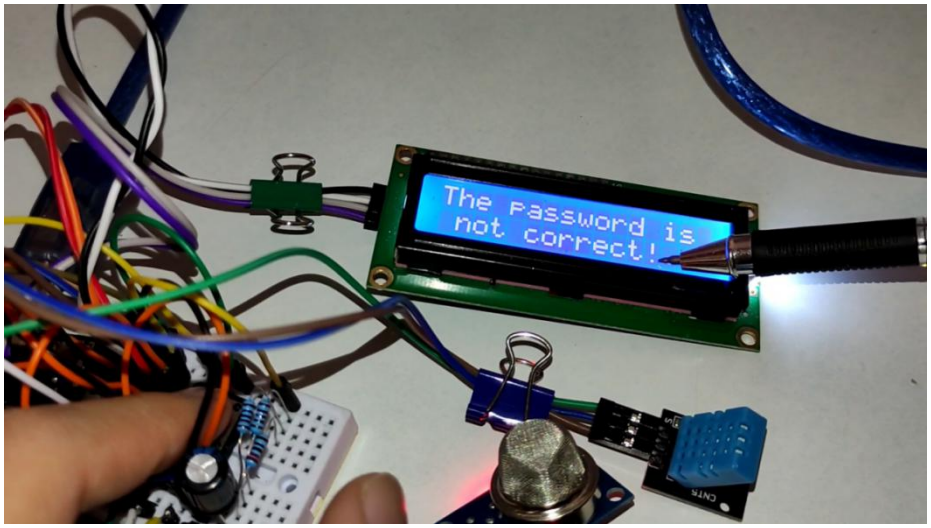


Picture 24. Creation of password “12345678”

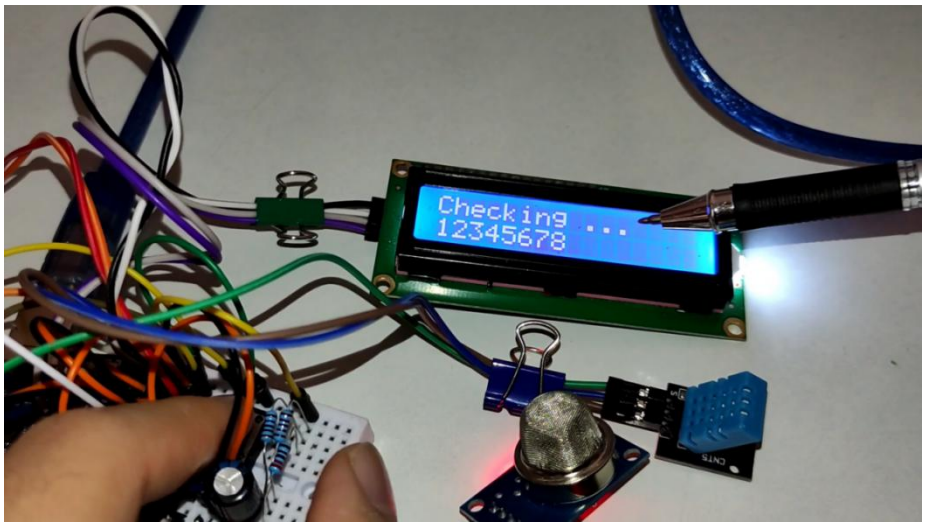
Testing stage



Picture 25. Typing a wrong password “67902345”

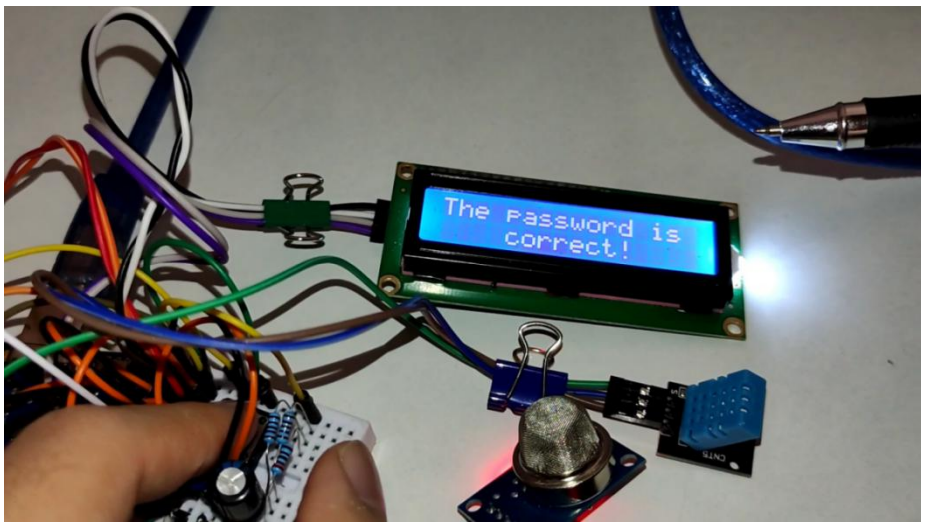


Picture 26. The result of typing a wrong password



Picture 27. Typing the correct password “12345678”

Testing stage



Picture 28. The result of typing the correct password

```
void passwordVerif(unsigned long password){
  enter:
  int countButton=0;
  int numStorage=0;
  while(login_password<=9999999){
  for(int i=0;i<=9;i++){
      lcd.clear();//clearing the LCD display
      lcd.setCursor(0, 0);
      lcd.print("Your password->");
      lcd.setCursor(15, 0); //setting cursor to a specific place on the LCD
      lcd.print(i);
      lcd.setCursor(0, 1);
      lcd.print(login_password);
      delay(700);
      if(login_password>=9999999){
        break;
      }
      cli();//stops the work of attachInterrupt
      int countButton;//Defining a variable for counting button presses
      countButton = countInterrupt;
      if(numStorage!=countButton){
        numStorage=countInterrupt;
        login_password=(login_password*10)+i;
        Serial.println(i);
        Serial.println(login_password);
      }
      sei();//continues the work of attachInterrupt
    }
  }
```

Testing stage

```
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Checking ...");
    lcd.setCursor(0, 1);
    lcd.print(login_password);
    delay(1000);
    if(login_password!=password){
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("The password is");
      lcd.setCursor(1, 1);
      lcd.print("not correct!");
      delay(3000);
      login_password=0;
      goto enter;
    }
    else{
      Serial.println("---------------");
      Serial.println(password);
      Serial.println(login_password);
      Serial.println("---------------");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("The password is");
      lcd.setCursor(4, 1);
      lcd.print("correct!");
      delay(3000);
      login_password=0;
    }
}
```

Picture 29. The code of the security system

In this test, I have created a password “12345678”. (Picture 24) When the device has finished its work, it has required the user to log in, for which I typed a wrong password “67902345”. (Picture 25) The device messaged “The password is not correct!” as seen in picture 26 and returned to the log-in page. Then I typed the correct password “12345678” and the device messaged “The password is correct!” on pictures 27 & 28. The verification of the password is checked by the function “passwordVerif”. (Picture 29) This means that the password security system works correctly. Gray-box testing methods was used for this test.

Testing stage

Test 4

Picture 30. Operation of the device for a while

Picture 31. Asking a new password after the turn-off

Testing stage

Picture 32. The loss of the data

In this test, I launched the device in order to allow it to gain some information from the environment. (Picture 30) After the pass of some time, I turned the device off and launched it again. As it was expected the device asked to type a new password. (Picture 31) Even after creating a new password, the database of the system was empty as it is seen in picture 32. This test was held by the black-box method.

Test 5

Picture 33. The main page of the operating device

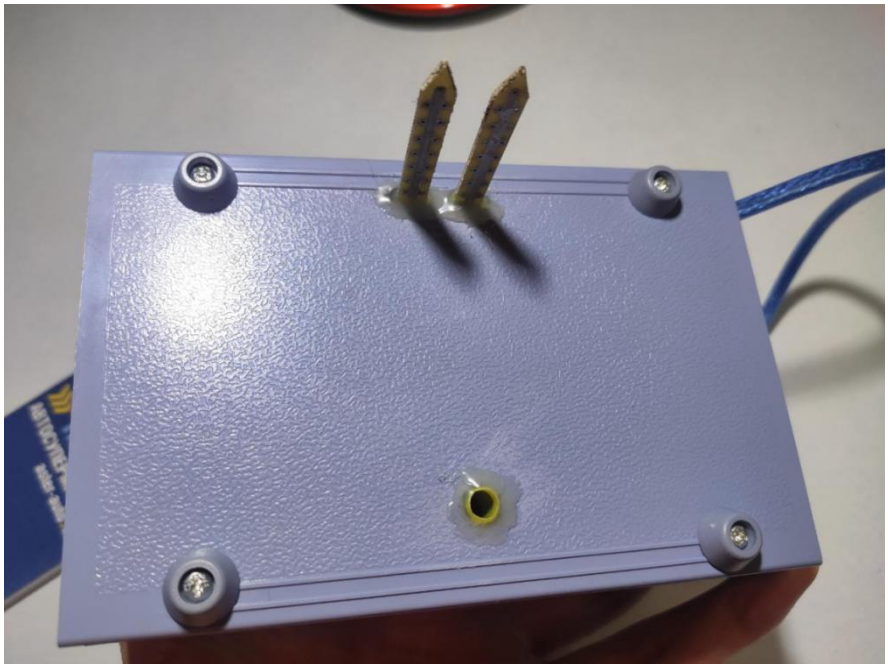
Testing stage

```
void mainPage(int countButton){
  if (countButton == 0) { //choosing the pages according to the number of button presses
        lcd.clear();//clearing the LCD display
        lcd.setCursor(0, 0); //setting cursor to a specific place on the LCD
        lcd.print(int(dht.readHumidity())); lcd.print("%H");//Humidity (%)
        lcd.setCursor(0, 1);
        lcd.print(int(dht.readTemperature())); lcd.print("T");//Temperature (*C)
        lcd.setCursor(10, 1);
        lcd.print(int(analogRead(AIn_gas))); lcd.print("G");//Toxic gases (ppm)
        lcd.setCursor(5, 0);
        lcd.print(int(dustDensity)); lcd.print("D");//Dust density (ug/m3)
        lcd.setCursor(5, 1);
        lcd.print(abs((1000 - int(analogRead(AIn_soil)))) / 10); lcd.print("%S"); //Soil humidity (%)
      }
      else {
        drawGraph(countButton);//turning to another page and drawing graph for a specific sensor
      }
}
```

Picture 34. The code of the “Main page” work

The device was placed near the plant. The monitor displayed 5 types of monitored values, which are H-Humidity, T-Temperature, D-Dust density, S-Soil humidity, G-Gas sensor. (Picture 33) The code fragment of “mainPage” function is shown in picture 34. Gray-box testing was used in this test.

Test 6



Testing stage

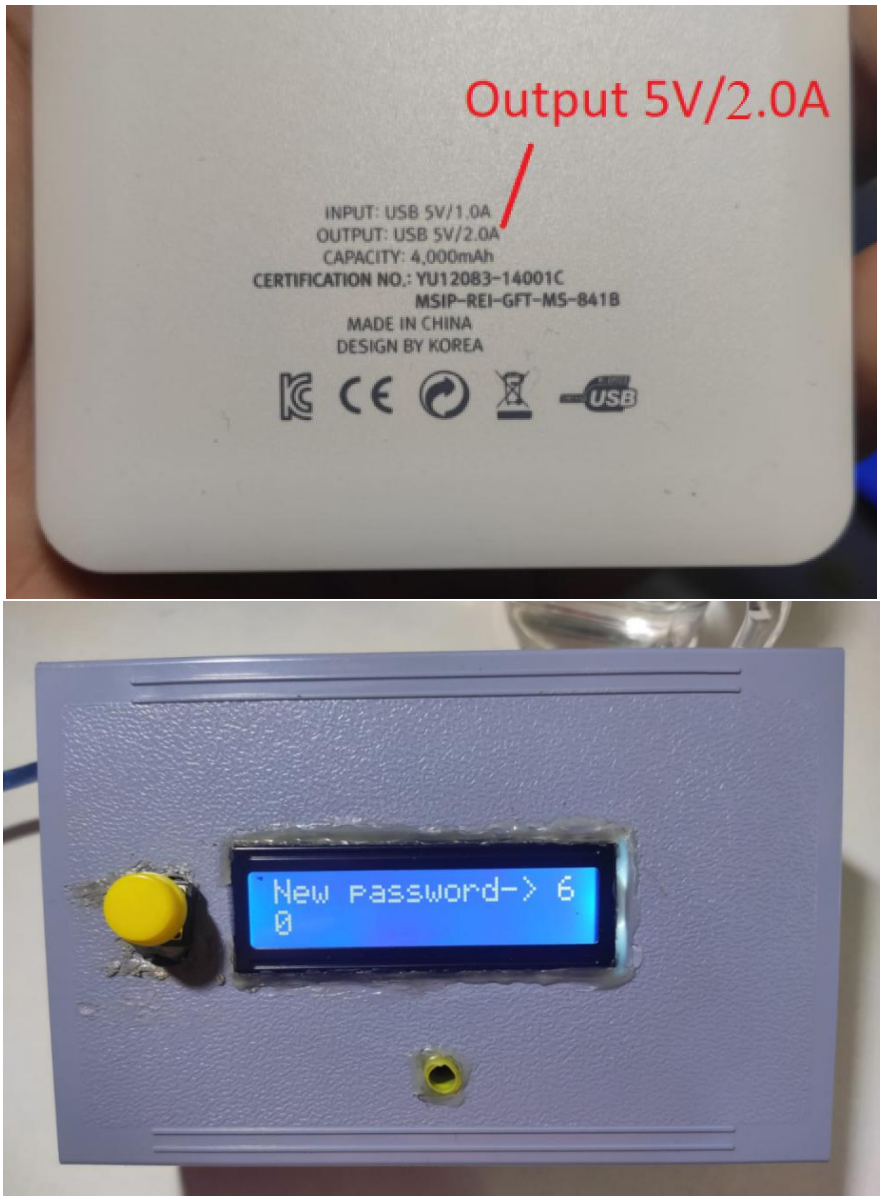
Picture 35. Positioning the soil humidity sensor in the atmosphere

Picture 36. Positioning the soil humidity sensor in the water

Testing stage

As it is seen in picture 35, the soil humidity sensor is positioned in the atmosphere, and it displays 1%. In picture 36, the soil humidity sensor is positioned in the water, however, it shows 50%. On the one hand, 1% in the atmosphere could be accepted; on the other hand, it seems that there should be 100% when the sensor is placed in water, as water contains 100% water itself, but the system displays 50%. As you have noticed, soil humidity sensor in extreme conditions may show unreal results. Above all, though, it should be mentioned that the soil humidity sensor is intended to be placed only in soil. Only black-box method was used for this test.

Test 7



Picture 37. The work of the device powered by 5V/2.0A

As it is seen in picture 37, the power bank has an output of 5V/2.0A, which can be used to charge the device. Only, the black-box method was used in this test.

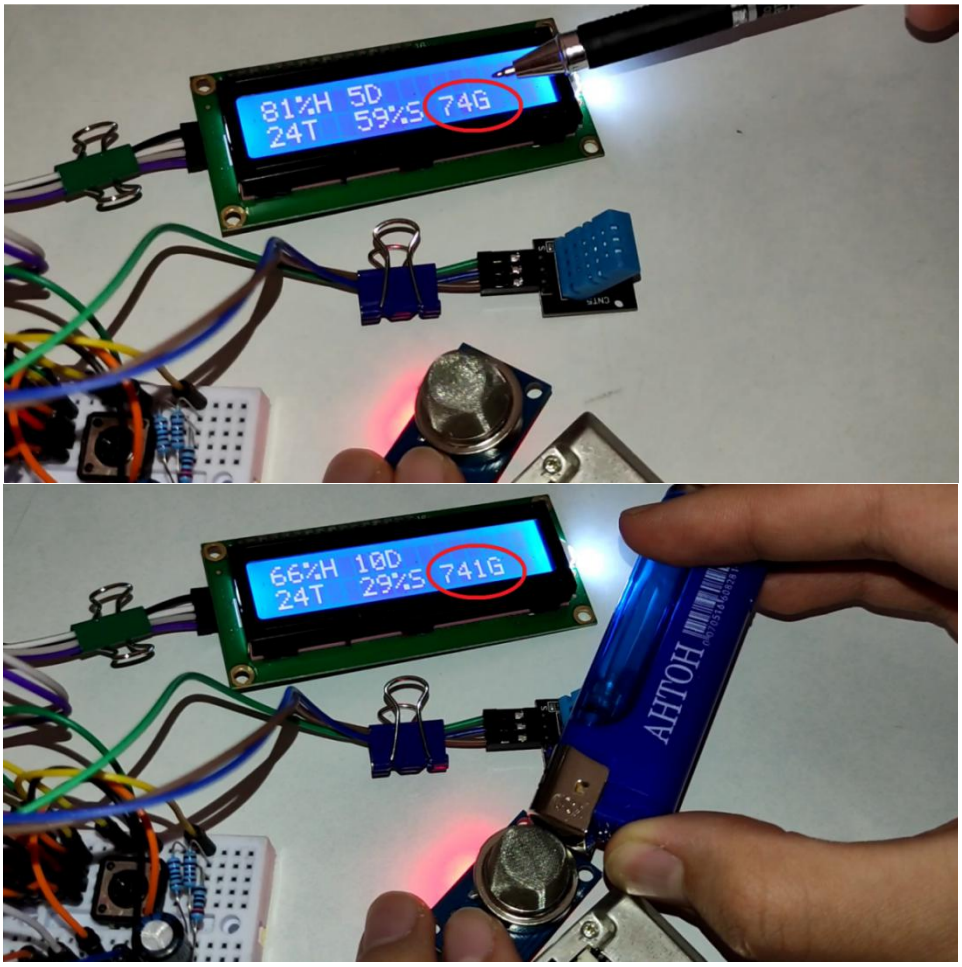
Testing stage

Test 8

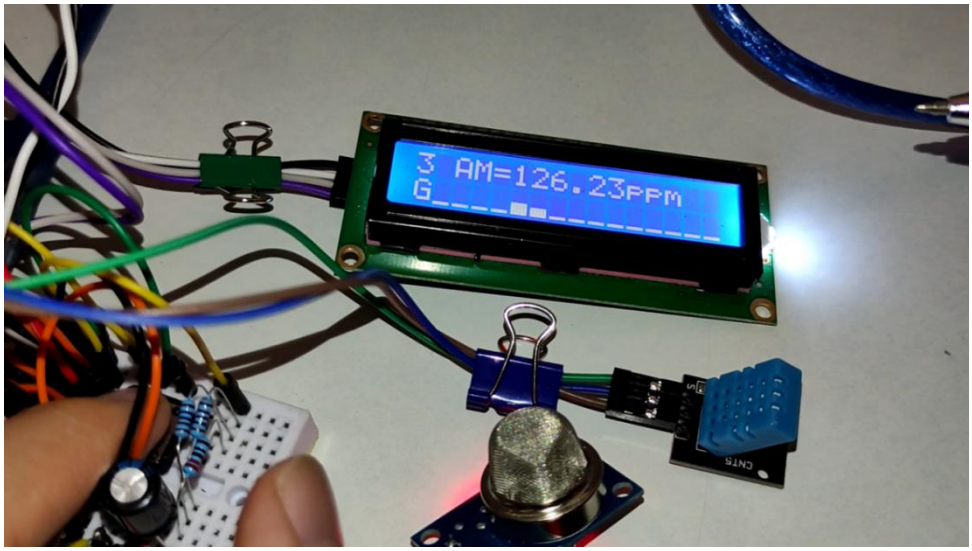
Picture 38. The work of the device powered by 5V/1.0A

As it is seen in picture 38, the power bank has an output of 5V/1.0A, which can be used to charge the device. Only, the black-box method was used in this test.

Test 9



Picture 39. The work of the gas detection sensor



Picture 40. The change of arithmetic mean

```
//A nested loop for 2D array
for (int r = 0; r < 15; r++) { //Outer loop
  for (int j = 0; j < 5; j++) { //Inner loop
    arr_data[j][r] = -1; //Filling the 2D array by "-1" as a default value
  }
}
//taking fifteen arithmetic mean values from each sensor (Outer loop)
for (int i = 0; i < 15; i++) {
  //updating arithmetic mean values to 0.
  arif_hum = 0;
  arif_temp = 0;
  arif_gas = 0;
  arif_soil = 0;
  arif_dust = 0;
  //a loop for finding an arithmetic mean every 4 senonds (Inner loop)
  for (int j = 0; j < 4; j++) {
    delay(1000);//a delay for 1 second
    timer++;;//a timer counting every second
    digitalWrite(ledPin, LOW); // power on the LED
    //delayMicroseconds(280);//Delay in microseconds(280)

    voltsMeasured = analogRead(dustPin); // Reading the dust sensor value

    delayMicroseconds(40);
    digitalWrite(ledPin, HIGH); // turn the LED off
    delayMicroseconds(9680);

    //calculating dust density
    calcVoltage = voltsMeasured * (5.0 / 1024.0);
    dustDensity = abs(170 * calcVoltage - 0.1);//unit: ug/m3
```

```
      //taking the sum of values
      arif_hum += dht.readHumidity(); //in %
      arif_temp += dht.readTemperature(); //in *C
      arif_gas += analogRead(AIn_gas); //in ppm
      arif_soil += abs((1000 - int(analogRead(AIn_soil)))) / 10; //in %
      arif_dust += dustDensity; //in mg/m3

      if (timer % 4 == 0) { //making values a single number for graphs
        arif_hum = arif_hum / 4 / 10;
        arif_temp = arif_temp / 4 / 10;
        arif_gas = arif_gas / 4 / 100;
        arif_soil = arif_soil / 4 / 10;
        arif_dust = arif_dust / 4 / 80;
        arr_data[0][i] = arif_hum;
        arr_data[1][i] = arif_temp;
        arr_data[2][i] = arif_gas;
        arr_data[3][i] = arif_soil;
        arr_data[4][i] = arif_dust;
      }
      cli();//stops the work of attachInterrupt
      //int countButton;//Defining a variable for counting button presses(changing pages on display)
      if (countInterrupt <= 5) { //A loop of counting
        countButton = countInterrupt;
      }
      if (countInterrupt == 6) { //limitation of the maximum number of button presses
        countInterrupt = 0;
      }
      sei();//continues the work of attachInterrupt
      Serial.println(countButton);//printing the number of button presses in the Serial Monitor

      mainPage(countButton);
    }
  }
```

Picture 41. The code of arithmetic mean computation

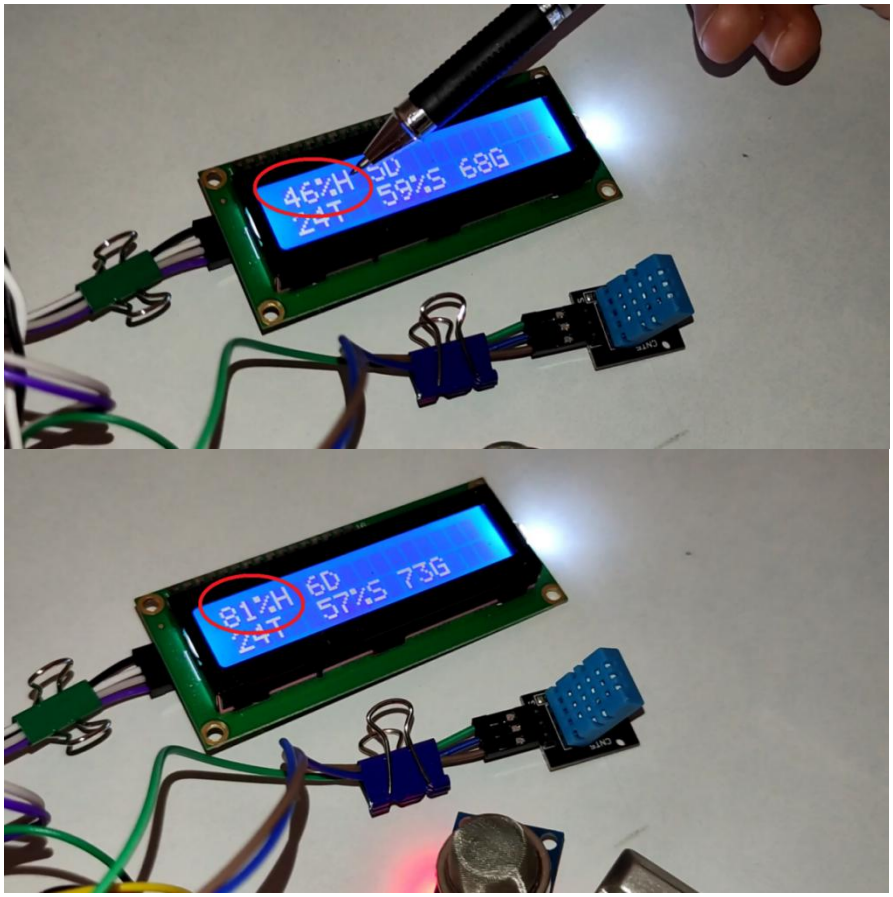
```
/*a recursive function for calculating a dynamic arithmetic mean data*/
double arithFunction(int startPoint, int rowNum) {
  if (startPoint > 14 || arr_data[rowNum][startPoint] == -1) {
    return arith / startPoint; //Returning an arithmetic value
  }
  arith += arr_data[rowNum][startPoint]; //Calculating the sum of values
  /*turning to the next index of the 2D array for a specific row,
   and calling this(arithFunction) function again*/
  arithFunction(startPoint + 1, rowNum);
}
```

Picture 42. A recursive function for the self-launch of the arithmetic mean computation

As you see in the pictures, the value gas sensor in a normal environment is 74ppm. However, when I pushed the lighter, the value changed to 741ppm. (Picture 39) Thus, the arithmetic mean data was changed to 126.23ppm due to the effect of the lighter, instead of leveling off at 74ppm. The code of arithmetic mean data storage is shown in picture 41. Also, a recursive function of arithmetic mean is shown in picture 42, which allows changing its data dynamically depending on time. The test was held on by using the gray-box testing method.

Testing stage

Test 10



Picture 43. The value of humidity before and after the breath on the sensor

Picture 44. The changes in the graph

Testing stage

```
/*Creating characters for drawing graphs
8, 7, 6, 5, 4, 3, 2, 1 lines*/
byte p00[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00100};
byte p10[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111};
byte p20[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111, 0b11111};
byte p40[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111, 0b11111, 0b11111};
byte p50[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b11111, 0b11111, 0b11111, 0b11111};
byte p60[8] = {0b00000, 0b00000, 0b00000, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte p70[8] = {0b00000, 0b00000, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte p90[8] = {0b00000, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte p100[8] = {0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};

/*a function for drawing graphs and displaying an arithmetic mean*/
void drawGraph(int countButton) {
  lcd.clear();//clearing the LCD display
  lcd.setCursor(0, 0); //setting cursor to a specific place on the LCD
  lcd.print(countButton);//Displaying a page number
  lcd.setCursor(0, 1);
  switch (countButton) {
    /* Displaying an index for each type of value
    1-humidity, 2-temperature, 3-toxic gases, 4-soil humidity, 5-dust density*/
    case 1: lcd.print("H"); break;
    case 2: lcd.print("T"); break;
    case 3: lcd.print("G"); break;
    case 4: lcd.print("S"); break;
    case 5: lcd.print("D"); break;
  }
  lcd.setCursor(2, 0);
  lcd.print("AM=");//abbreviation of "Arithmetic mean"
  switch (countButton) { //Displaying a dynamic arithmetical mean data
    case 1: lcd.print(arithFunction(0, countButton - 1) * 10); //humidity
      lcd.print("%"); break;
    case 2: lcd.print(arithFunction(0, countButton - 1) * 10); //temperature
      lcd.print("*C"); break;
    case 3: lcd.print(arithFunction(0, countButton - 1) * 100); //toxic gases
      lcd.print("ppm"); break;
    case 4: lcd.print(arithFunction(0, countButton - 1) * 10); //soil humidity
      lcd.print("%"); break;
    case 5: lcd.print(arithFunction(0, countButton - 1) * 80); //dust density
      lcd.print("ug/m3"); break;
    default: break;
  }
```

```
  arith = 0; //updating an arithmetic mean value to 0.
  for (int t = 1; t < 16; t++) { //displaying graph for a specific sensor
    lcd.setCursor(t, 1);
    /*taking a round value from the 2D array for displaying
    the necessary character (making a graph)*/
    switch (round(arr_data[countButton - 1][t - 1])) {
      case 0: lcd.write(byte(1)); break;
      case 1: lcd.write(byte(1)); break;
      case 2: lcd.write(byte(2)); break;
      case 3: lcd.write(byte(3)); break;
      case 4: lcd.write(byte(3)); break;
      case 5: lcd.write(byte(4)); break;
      case 6: lcd.write(byte(5)); break;
      case 7: lcd.write(byte(6)); break;
      case 8: lcd.write(byte(7)); break;
      case 9: lcd.write(byte(7)); break;
      case 10: lcd.write(byte(8)); break;
      default: break;
    }
  }
}
```

Picture 45. The code of setting different kinds of graph columns and displaying them on the LCD Screen

Picture 43 shows the value of humidity before and after my breathing on the sensor. Picture 44 shows the consequence in terms of the graph which was happened because of my breath on the humidity sensor. As it is seen, the system can show the recorded data in terms of a graph considering all the changes in value. The code is shown in picture 45. First of all, I created all possible characters for the graphs and connected them with the database of recorded values. Then, by measuring the percentage of the data, I displayed the graph on the screen. Both back box testing and white box testing were used in this test.

Installation

Installation plan

Installation as other actions needs planning because it is important to train the user so that she or he can use the device on her or his own without the intervention of the developer. To replace the old system with the new device, a meeting with the client was organized in order to plan further actions of the installation:

1. Preparing a user manual of the device with instructions for use.
2. Testing the new system with the client to check out the correspondence of the device with the requirements.
3. Introducing the functions of the device to the client.
4. Explaining to the customer how to handle the settings of the device
5. Taking into account any kinds of suggestions to improve the device from the client, and correct bugs of the device work.
6. Giving a questionnaire to the client about the functionality of the device
7. Replacing the current system with the new one.
8. Further support of the client until the customer gets used to the new system.

Staff training plan

Suggested training date: 24th of December, 2021 - 10th of January, 2022

Reason: The developer and the school's staff will have holidays during the given period. Thus, it would be the best time for having lessons about the device. By the start of the third term, the client will be ready to use the device at her or his workplace, during the school lessons.

Suggested training method: Offline and individual methods (one to one) of training will be used.

Reason: As regards learning, the best method is teaching offline and individually. This is because the human mind learns much better in practice. Also, the device is one, and it cannot be used by many people at one time.

System implementation plan

Date of implementation: 12th of January - 27th of March, 2022

Reason: The third term at Nazarbayev Intellectual School starts on the 12th of January and ends on the 27th of March. Also, the immediate installation of the device after training allows the client not to forget the instructions for using the device.

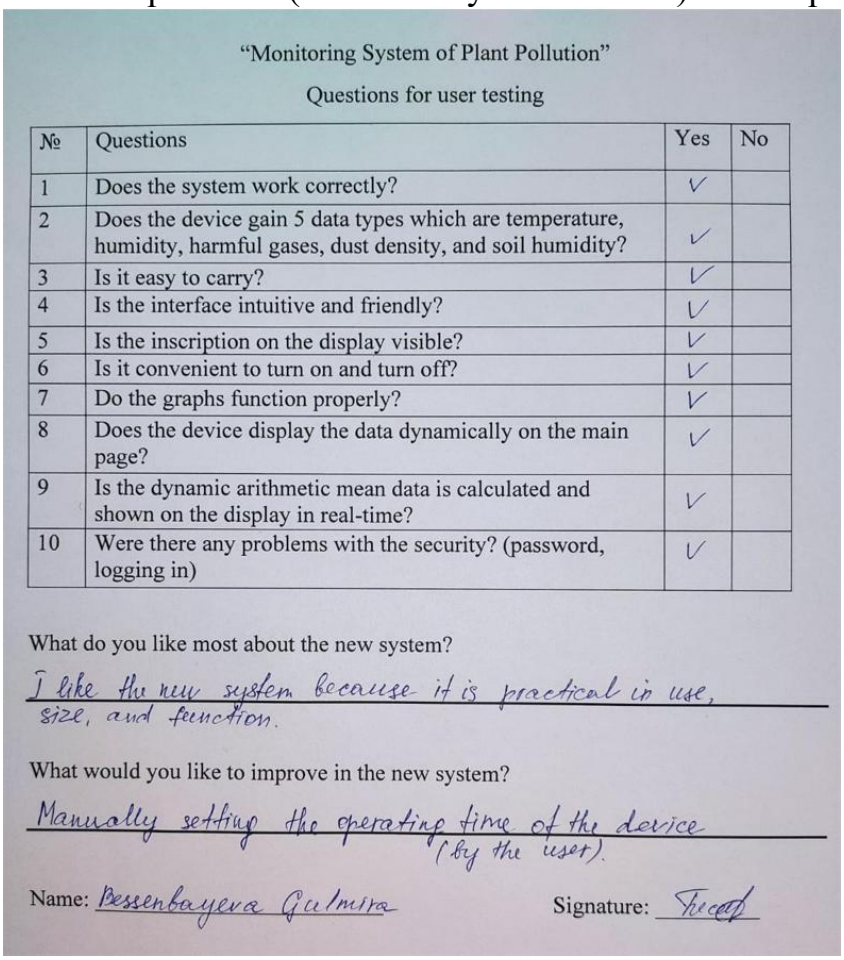
Method of implementation: Pilot running

Reason: To replace the current system with the new one, a pilot running will be used. Because it allows the implementation of the new system gradually without affecting the old system. If the device will work fine during some period

of time, the administrator of the school will allow to implement the device in other classes. This means that once the device was approved by some school staff, the existing system will be replaced completely during its gradual implementation. This method of implementation is known as “to experiment or test before introducing something more widely”, which makes the process easier as it runs on a smaller scale.

User testing

The system was tested during the meeting with the client. She has checked every single function of the device and approved its workability by completing the questionnaire. The questionnaire which was provided below consists of 10 closed questions (Answers: “yes” and “no”) and 2 open questions.



“Monitoring System of Plant Pollution”

Questions for user testing

№	Questions	Yes	No
1	Does the system work correctly?	✓	
2	Does the device gain 5 data types which are temperature, humidity, harmful gases, dust density, and soil humidity?	✓	
3	Is it easy to carry?	✓	
4	Is the interface intuitive and friendly?	✓	
5	Is the inscription on the display visible?	✓	
6	Is it convenient to turn on and turn off?	✓	
7	Do the graphs function properly?	✓	
8	Does the device display the data dynamically on the main page?	✓	
9	Is the dynamic arithmetic mean data is calculated and shown on the display in real-time?	✓	
10	Were there any problems with the security? (password, logging in)	✓	

What do you like most about the new system?

I like the new system because it is practical in use, size, and function.

What would you like to improve in the new system?

Manually setting the operating time of the device (by the user).

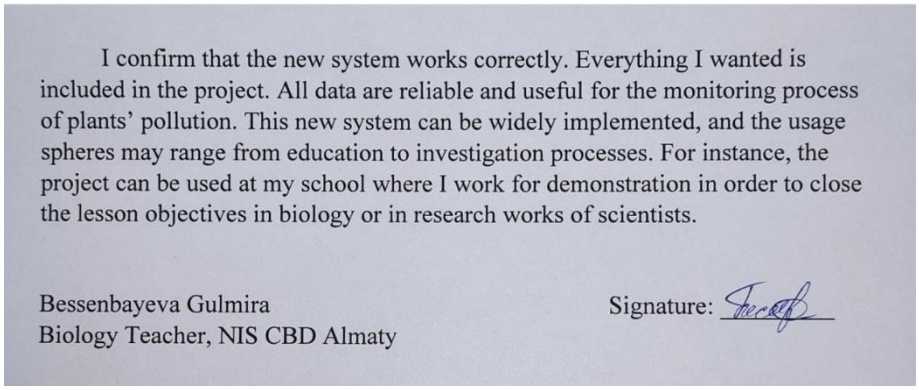
Name: Bessenbayeva Gulmira Signature:

Picture 46. The user testing approval

Written evidence from the client

The client has tested the device and was asked to write a letter of evidence that the new system is completely ready for use.

Installation stage



I confirm that the new system works correctly. Everything I wanted is included in the project. All data are reliable and useful for the monitoring process of plants' pollution. This new system can be widely implemented, and the usage spheres may range from education to investigation processes. For instance, the project can be used at my school where I work for demonstration in order to close the lesson objectives in biology or in research works of scientists.

Bessenbayeva Gulmira
Biology Teacher, NIS CBD Almaty

Signature:

Picture 47. Written evidence form the client

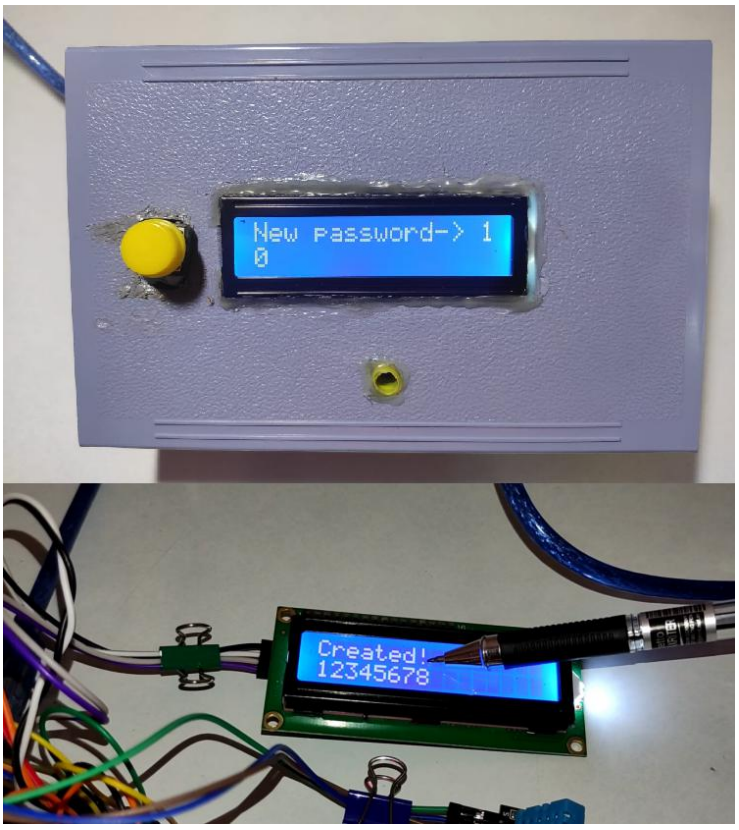
Documentation

User Guide

Monitoring System of Plant Pollution – an innovative and smart device that helps scientists or clients to analyze the quality of the environment and its changes. (Picture 48) Mostly in identifying what exactly pollutes plants as they start to produce less oxygen.

Picture 48. Photo of the device

Once you have turned “Monitoring System of Plant Pollution” on by connecting a power bank, it shows a registration page (Picture 49), where the user should type the password in.

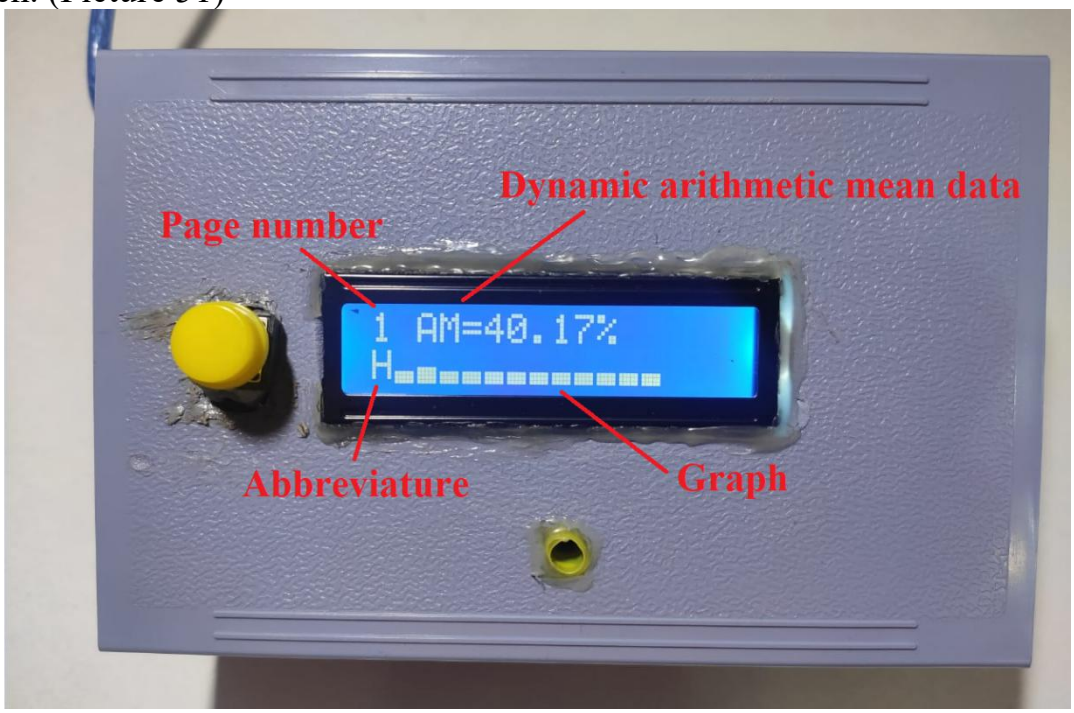


Picture 49. Creating a new password

At the next point, the device shows its main page, where you can see 5 values gained from the environment, which are air temperature, air humidity, soil humidity, dust density in the atmosphere, and the amount of harmful gases.

Picture 50. The main page

During the work of the device, the user can see and analyze the recorded data that is shown as graphs. Additionally, dynamic arithmetic mean data is also calculated and displayed above the graphs. The user can identify each page by the abbreviation (“H”, “D”, “T”, “S”, “G”) in the bottom left corner of the screen. (Picture 51)



Picture 51. Page of humidity data

Documentation stage

Index

Table 9 - Index

Glossary

Table 10 - Components of "MSPP"

#	Components of "Monitoring System of Plant Pollution"	Description
1	Soil humidity sensor	Measures the humidity level of the soil.
2	Dust density sensor	Calculates the amount of particles in the air.
3	DHT11	Measures the temperature and humidity level of the atmosphere.
4	MQ-5	Identifies the presence of harmful gases such as methane, propane, butane, and others.

Fritzing – A website that offers a software tool for documenting Arduino prototypes.
Arduino IDE – It is a software for programming Arduino boards such as Arduino Uno, Arduino Mega, Arduino Nano, and others.
Microsoft Paint – A multifunctional and simple raster graphics editor for painting.
Google Chrome – A cross-platform web browser for searching any kind of information on the Internet.
Microsoft Word - A word processing program for creating both simple and complex documents.
Database – it is a set of data which is stored successively so that a computer can find the needed data easily. Also, databases can be modified when new information is added, updated, or deleted.
System Software – it is a software that can provide an environment where different kinds of applications can be run. Examples can be operational systems such as Windows, Linux, Chrome OS, and others.
Interface – allows people to interact with the device. It is a type of communication between humans and systems.
Authentication – a process that checks the credentials provided by the client with the original information stored in the database.

Documentation stage

Data – different types of information that is transferred among the components, databases, and variables.

Guide to common errors

Table 11 - Guide to common errors

Errors	Reason	Solution
When turning the device on, the LCD Screen does not light up	Power bank could be discharged	Charge the power bank.
When turning the device on, the LCD Screen shows unreadable and weird symbols.	There is a possibility of contact bounce	Press the reset button or relaunch the device by reconnecting the power bank.
When turning the device on, one of the components or some of them may not work correctly.	Wires could be disconnected	Check the connection of wires.

Back-up Routine

A backup routine is a precautionary measure for the case of data loss. It allows the user to restore the data for future uses and prevents its loss.

Arduino always uses its ram as a memory for saving data. However, if the device is turned off, it loses all the recorded data. Thus, a static memory which is called EEPROM is used for saving the data gained by Arduino. It helps to reread and reuse the saved information even after the erroneous turn-off of the device, and Arduino will not lose its database. This function is represented as a backup of the device, allowing to restore its information.

Also, an incremental backup can be used to prevent data loss. My device gains new data each hour or even each day, which means that it gains new information by piece, and the stored data is not changed or updated. Thus, only the most recent backups are needed to be saved upon the previous backup for my device. This allows to save the data quickly, but the recovery process will take more time.

On-Screen help

On-screen help is not suitable for this device as it is made on Arduino and, it is not a website or app.

Evaluation

Evaluation of the system – reference of “Design section” objective

1. Create a little device that is comfortable to carry.

Picture 52. The picture of the device

The size of the plastic box is 120*80*40 mm, and it weighs about 280 grams. The size and weight of the device make it portable and easy to carry.

2. Collect air pollution, soil humidity, dust density, temperature, and air humidity data.

Picture 53. The display of 5 different values

As it is seen in picture 46, a new system can gain 5 different data which are temperature, humidity, harmful gases, dust density, soil humidity.

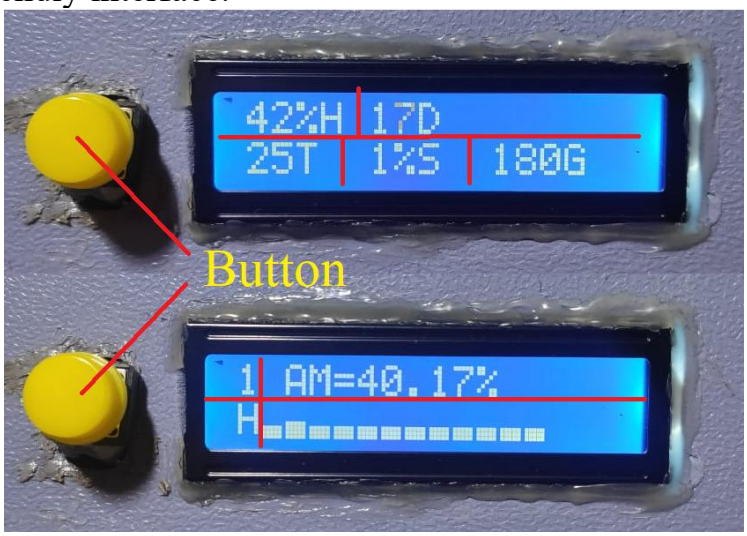
3. Make graphs for each data type for convenience.

Evaluation stage

Picture 54. The picture of the graph

The “MSPP” can display each recorded data in the form of graphs which contains 15 columns. This makes it easy to analyze the change in the values in the given period of time.

4. Draw a user-friendly interface.



Picture 55. The photo of the interface

The main page is divided into 5 sections, whereas other pages are divided into 4 sections, which allows the user to see the data separately from each other. Also, only one button is used for the interaction with the device. This helps to avoid confusion.

5. Make easy management and control of the device.

Evaluation stage

```
void mainPage(int countButton){
  if (countButton == 0) { //choosing the pages according to the number of button presses
        lcd.clear();//clearing the LCD display
        lcd.setCursor(0, 0); //setting cursor to a specific place on the LCD
        lcd.print(int(dht.readHumidity())); lcd.print("%H");//Humidity (%)
        lcd.setCursor(0, 1);
        lcd.print(int(dht.readTemperature())); lcd.print("T");//Temperature (*C)
        lcd.setCursor(10, 1);
        lcd.print(int(analogRead(AIn_gas))); lcd.print("G");//Toxic gases (ppm)
        lcd.setCursor(5, 0);
        lcd.print(int(dustDensity)); lcd.print("D");//Dust density (ug/m3)
        lcd.setCursor(5, 1);
        lcd.print(abs((1000 - int(analogRead(AIn_soil)))) / 10); lcd.print("%S"); //Soil humidity (%)
      }
      else {
        drawGraph(countButton);//turning to another page and drawing graph for a specific sensor
      }
}

/*a function for drawing graphs and displaying an arithmetic mean*/
void drawGraph(int countButton) {
  lcd.clear();//clearing the LCD display
  lcd.setCursor(0, 0); //setting cursor to a specific place on the LCD
  lcd.print(countButton);//Displaying a page number
  lcd.setCursor(0, 1);
  switch (countButton) {
    /* Displaying an index for each type of value
    1-humidity, 2-temperature, 3-toxic gases, 4-soil humidity, 5-dust density*/
    case 1: lcd.print("H"); break;
    case 2: lcd.print("T"); break;
    case 3: lcd.print("G"); break;
    case 4: lcd.print("S"); break;
    case 5: lcd.print("D"); break;
  }
  lcd.setCursor(2, 0);
  lcd.print("AM=");//abbreviation of "Arithmetic mean"
  switch (countButton) { //Displaying a dynamic arithmetical mean data
    case 1: lcd.print(arithFunction(0, countButton - 1) * 10); //humidity
      lcd.print("%"); break;
    case 2: lcd.print(arithFunction(0, countButton - 1) * 10); //temperature
      lcd.print("*C"); break;
    case 3: lcd.print(arithFunction(0, countButton - 1) * 100); //toxic gases
      lcd.print("ppm"); break;
    default: break;
  }
  arith = 0; //updating an arithmetic mean value to 0.
  for (int t = 1; t < 16; t++) { //displaying graph for a specific sensor
    lcd.setCursor(t, 1);
    /*taking a round value from the 2D array for displaying
    the necessary character (making a graph)*/
    switch (round(arr_data[countButton - 1][t - 1])) {
      case 0: lcd.write(byte(1)); break;
      case 1: lcd.write(byte(1)); break;
      case 2: lcd.write(byte(2)); break;
      case 3: lcd.write(byte(3)); break;
      case 4: lcd.write(byte(3)); break;
      case 5: lcd.write(byte(4)); break;
      case 6: lcd.write(byte(5)); break;
      case 7: lcd.write(byte(6)); break;
      case 8: lcd.write(byte(7)); break;
      case 9: lcd.write(byte(7)); break;
      case 10: lcd.write(byte(8)); break;
      default: break;
    }
  }
}
```

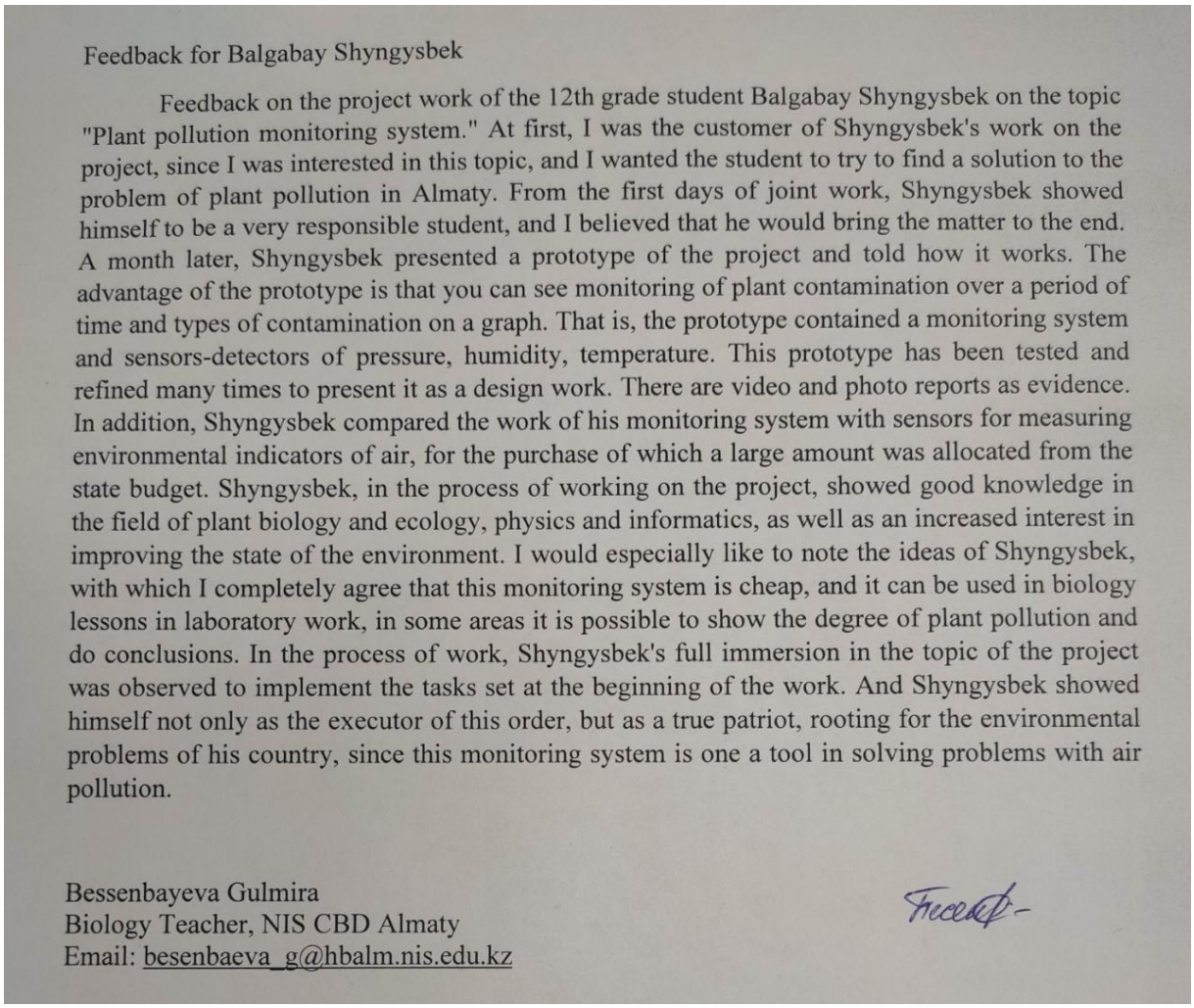
Picture 56. A fragment of "Management and control" code

As it is clearly seen on the screenshot of the code, only one variable is used for changing the pages of the screen. This means that one button is used for

the interaction with the device. Thus, the management and control of the device are intuitive.

Feedback from the client

The picture of the signed feedback, which describes my work on the device, from the client is given below:



Feedback for Balgabay Shyngysbek

Feedback on the project work of the 12th grade student Balgabay Shyngysbek on the topic "Plant pollution monitoring system." At first, I was the customer of Shyngysbek's work on the project, since I was interested in this topic, and I wanted the student to try to find a solution to the problem of plant pollution in Almaty. From the first days of joint work, Shyngysbek showed himself to be a very responsible student, and I believed that he would bring the matter to the end. A month later, Shyngysbek presented a prototype of the project and told how it works. The advantage of the prototype is that you can see monitoring of plant contamination over a period of time and types of contamination on a graph. That is, the prototype contained a monitoring system and sensors-detectors of pressure, humidity, temperature. This prototype has been tested and refined many times to present it as a design work. There are video and photo reports as evidence. In addition, Shyngysbek compared the work of his monitoring system with sensors for measuring environmental indicators of air, for the purchase of which a large amount was allocated from the state budget. Shyngysbek, in the process of working on the project, showed good knowledge in the field of plant biology and ecology, physics and informatics, as well as an increased interest in improving the state of the environment. I would especially like to note the ideas of Shyngysbek, with which I completely agree that this monitoring system is cheap, and it can be used in biology lessons in laboratory work, in some areas it is possible to show the degree of plant pollution and do conclusions. In the process of work, Shyngysbek's full immersion in the topic of the project was observed to implement the tasks set at the beginning of the work. And Shyngysbek showed himself not only as the executor of this order, but as a true patriot, rooting for the environmental problems of his country, since this monitoring system is one a tool in solving problems with air pollution.

Bessenbayeva Gulmira
Biology Teacher, NIS CBD Almaty
Email: besenbaeva_g@hbalm.nis.edu.kz

Picture 57. Feedback from the client

Candidate analysis of the client feedback

Ms. Gulmira has written feedback for my work (Picture 57). It is said that the device works properly without any kinds of errors. Also, it is able to gain the ordered number of data types from the environment, which was asked at the beginning of the project work. Moreover, my client agrees that this project is much cheaper than the one purchased by the state. She also proofs that “MSPP” is ready for use in various fields. However, it would be better if a soil acidity sensor was installed on the device. Because the combination of six different data could give full information about the state of the environment. The reason for the absence of the soil acidity sensor is that it costs a fortune and is not efficient in long-term use because of corrosion.