ОГЛАВЛЕНИЕ

[РАЗВЕРТЫВАНИЕ СИСТЕМЫ 2](#_Toc5562631)

[1 Задействованные в проекте библиотеки 2](#_Toc5562632)

[2 Инструкция по установке библиотек 2](#_Toc5562633)

[3 Состав пакета файлов проекта 2](#_Toc5562634)

[4 Инструкция по установке и запуску 2](#_Toc5562635)

[ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 3](#_Toc5562636)

[1 Глобальные структуры данных 3](#_Toc5562637)

[2 Функции и методы 4](#_Toc5562638)

[3 Блок-схема функционирования ПО 6](#_Toc5562639)

[4 Граф вызова функций 9](#_Toc5562640)

[5 Методика тестирования 10](#_Toc5562641)

[6 Листинг кода 11](#_Toc5562642)

# РАЗВЕРТЫВАНИЕ СИСТЕМЫ

# Задействованные в проекте библиотеки

Файлы всех необходимых библиотек находятся в директории с проектом.

**DHT Sensor Library - Version: 1.3.4**

Библиотека датчика температуры и влажности воздуха [1]. Файл: DHT\_sensor\_library-1.3.4.zip

**LiquidCrystal - Version: 1.0.7**

Библиотека LCD дисплея [2]. Файл: LiquidCrystal-1.0.7.zip

# Инструкция по установке библиотек

1. Скачать и установить Arduino IDE [3].
2. Запустить Arduino IDE.
3. В верхнем меню программы выбрать Sketch -> Include Library -> Add .zip library.
4. Перейти в папку с .zip архивом библиотеки.
5. Выбрать .zip архив библиотеки и нажать Open.

# Состав пакета файлов проекта

|  |  |
| --- | --- |
| **Название и тип файла** | **Назначение файла** |
| plantMonitor.ino | Файл реализации ПО. Этот файл необходимо открывать в Arduino IDE. |
| plantMonittor.h | Заголовочный файл ПО. Подключается в файл реализации, содержит в себе прототипы функций и структуры данных. |
| plantMonitorConfig.h | Заголовочный файл конфигурации ПО. Подключается в файл реализации, содержит в себе конфигурацию ПО в виде директив #define (интервалы, входы/выходы, сообщения и т.д.) |
| LICENSE.txt | Файл лицензии ПО. На него ссылаются три файла выше. |
| DHT\_sensor\_library-1.3.4.zip | Архив библиотеки DHT Sensor Library датчика температуры. |
| LiquidCrystal-1.0.7.zip | Архив библиотеки LiquidCrystal LCD дисплея. |

# Инструкция по установке и запуску

1. Экспортировать файлы проекта в произвольную папку.
2. Скачать и установить Arduino IDE [3].
3. Запустить Arduino IDE.
4. Установить все необходимые библиотеки (см. пункт 1 и 2 текущего раздела).
5. Открыть меню открытия файла в верхнем меню программы File -> Open.
6. Перейти в папку с файлами проекта.
7. Выбрать plantMonitor.ino и нажать кнопку Open.
8. Подключить собранное устройство Arduino Uno к компьютеру при помощи USB кабеля.
9. В верхнем меню IDE перейти к Tools -> Port и выбрать порт, соответствующий подключенному устройству.
10. В верхнем меню IDE выбрать Sketch -> Upload для компиляции и загрузки ПО на устройство.
11. После компиляции, загрузки и двухсекундного приветствия ПО и устройство будут находиться в рабочем состоянии.

# ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

# Глобальные структуры данных

**struct Param**

Структура, хранящая информацию о считываемых параметрах. Далее представлено описание полей структуры:

|  |  |  |
| --- | --- | --- |
| **Тип** | **Имя** | **Описание** |
| const String | name | Полное название параметра для вывода в консоль |
| const String | shortName | Сокращенное название параметра для отображения на экране |
| const int | minValue | Минимальное приемлемое значение параметра |
| const int | maxValue | Максимальное приемлемое значение параметра |
| const String | messageLacking | Сообщение, выводимое на экран, когда значение параметра меньше минимального |
| const String | messageExceeding | Сообщение, выводимое на экран, когда значение параметра больше максимального |
| const bool | invertMinMax | Меняет местами messageLacking и messageExceeding |
| const int | decimalPlaces | Сколько знаков после точки будет выводиться на экран для значения параметра |
| float (\*)() | readValue | Функция для считывания значения параметра - возвращает значение в виде числа с плавающей точкой |
| float | cachedValue | Здесь будет кешироваться значение, возвращенное из readValue, чтобы не считывать значение несколько раз |

**Переменные типа Param и их значения.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Переменная** | **name** | **sN** | **minV** | **maxV** | **msgL** | **msgE** | **invMM** | **dP** | **readValue** |
| **soilHumidity** | "Soil Humidity" | "SH" | SH\_MAX | SH\_MAX | "WATER THE PLANTS" | "TOO MUCH WATER" | true | 0 | getSoilHumidity |
| **ambientBrightness** | "Ambient Brightness" | "AB" | -1 | AB\_MAX | "" | "LIGHT TOO BRIGHT" | false | 0 | getAmbientBrightness |
| **airHumidity** | "Air Humidity" | "AH" | AH\_MIN | AH\_MAX | "AIR IS TOO DRY" | "AIR IS TOO MOIST" | false | 1 | getAirHumidity |
| **airTemperature** | "Air Temperature" | "AT" | AT\_MIN | AT\_MAX | "AIR IS TOO COLD" | "AIR IS TOO HOT" | false | 1 | getAirTemperature |

**Param \*params[]**

Массив указателей на структуры данных параметров. Нужен для итерации по всем параметрам.

**const int numParams**

Количество параметров в массиве параметров.

**bool numParams**

Включено ли дополнительное освещение (диод).

**LiquidCrystal lcd**

Интерфейс LCD экрана.

**DHT dht**

Интерфейс датчика воздуха.

**Конфигурационные параметры, объявляемые директивой #define.**

|  |  |  |
| --- | --- | --- |
| Имя | Значение | Описание |
| SH\_MIN | 220 | Минимальные и максимальные приемлемые значения влажности почвы. |
| SH\_MAX | 600 |
| AB\_MIN | 50 | Минимальные и максимальные приемлемые значения яркости освещения. |
| AB\_MAX | 100 |
| AH\_MIN | 20 | Минимальные и максимальные приемлемые значения влажности воздуха. |
| AH\_MAX | 70 |
| AT\_MIN | 22 | Минимальные и максимальные приемлемые значения температуры воздуха. |
| AT\_MAX | 40 |
| DHT\_TYPE | DHT22 | Тип датчика влажности и температуры воздуха. |
| PIN\_A\_PHOTOSENSOR | 1 | Пин датчика яркости освещения. Аналоговый. |
| PIN\_A\_SOILHUMIDITY | 0 | Пин датчика влажности почвы. Аналоговый |
| PIN\_D\_DHT | 7 | Пин датчика температуры и влажности воды. Цифровой. |
| PIN\_A\_UNUSED | 4 | Неиспользуемый пин для инициализации генератора случайных чисел. Аналоговый. |
| PIN\_D\_BUZZER | 6 | Пин спикера (пьезоизлучателя). |
| PIN\_D\_LIGHT | 13 | Пин дополнительного освещения (диода). |
| PIN\_LCD\_1 | 12 | Номера входов/выходов (пинов) цифрового сигнала, использующихся LCD экраном |
| PIN\_LCD\_2 | 11 |
| PIN\_LCD\_3 | 5 |
| PIN\_LCD\_4 | 4 |
| PIN\_LCD\_5 | 3 |
| PIN\_LCD\_6 | 2 |
| DATA\_RATE | 9600 | Скорость потока данных в битах в секунду. |
| INIT\_DELAY | 2000 | Задержка перед началом проверки значений датчиков. |
| UPDATE\_INTERVAL | 1000 | Интервал проверки значений датчиков. |
| DISPLAY\_WIDTH | 16 | Ширина дисплея в символах. |
| DISPLAY\_HEIGHT | 2 | Высота дисплея в строках. На данный момент поддерживается только 2 строки. |
| DISPLAY\_GREETING | "HELLO!" | Сообщение, выводимое при включении устройства. |

# Функции и методы

**void setup()**

Инициализирует программу, выполняется один раз при подаче напряжения или сбросы платы Arduino.

**void initOutput()**

Устанавливает пины спикера и диода на режим выхода.

**void initRandom()**

Инициализирует генератор случайных чисел информацией из неподключенного аналогового пина. random используется для вывода звуков случайной частоты функцией **playAlert**().

**void initDisplay()**

Инициализирует размеры дисплея и выводит приветствие на экран.

**void loop()**

Считывает, анализирует и выводит текущие значения подключенных датчиков. Выполняется после функции setup() в бесконечном цикле.

**void turnLightOn()**

Включает дополнительное освещение (диод), если оно уже не включено. Меняет глобальное isLightOn.

**void turnLightOff()**

Выключает дополнительное освещение (диод), если оно включено. Меняет глобальное isLightOn.

**void playAlert()**

Оповещает пользователя звуком, имитирующим сирену.

**void logAllValues()**

Выводит значение всех параметров в консоль.

**void logValue(String name, float value, int decimalPlaces)**

Выводит значение параметра в консоль в формате "name: value". decimalPlaces указывает число знаков значения параметра после точки, которые нужно вывести.

**void updateAllValues()**

Считывает значения всех датчиков и кеширует их в структурах параметров.

**float getSoilHumidity()**

Считывает и возвращает влажность почвы от 0 до 1023. 0 – влажная. 1023 – сухая.

**float getAmbientBrightness()**

Считывает и возвращает яркость освещения от 0 до 1023. 0 – темно. 1023 - светло

**float getAirHumidity()**

Считывает и возвращает влажность воздуха от 0 до 100%.

**float getAirTemperature()**

Считывает и возвращает температуру воздуха от -40 до +150 градусов цельсия.

**bool isAllParamsOk()**

Проверяет и возвращает, попадают ли все параметры в приемлемые рамки. Если нет, то на экран будет выведено предупреждение для первого параметра, вышедшего за рамки дозволенного.

**bool isParamOk(Param \*param)**

Проверяет и возвращает, попадает ли параметр в приемлемые рамки. Если нет, то выводит на экран выведено предупреждение для первого параметра, вышедшего за рамки дозволенного. Принимает ссылку на структуру с данными параметра.

**void resetDisplay()**

Очищает дисплей и устанавливает курсор в начальную позицию.

**void displayAllValues()**

Выводит значения всех параметров с их сокращенным названием по два на строке экрана.

**void displayWarning(String warning)**

Выводит переданное предупреждение на экран.

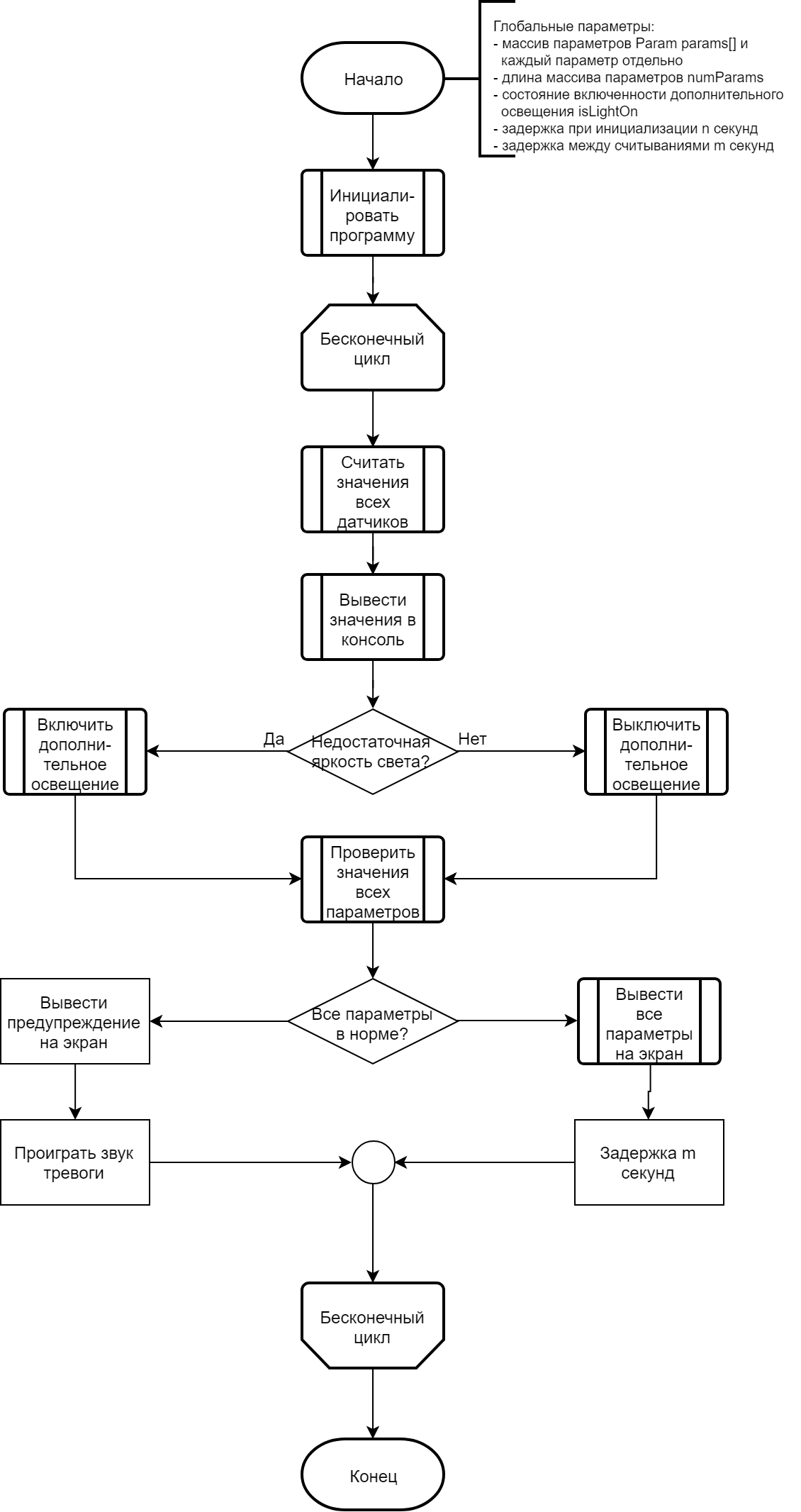
**void displayLineEnd()**

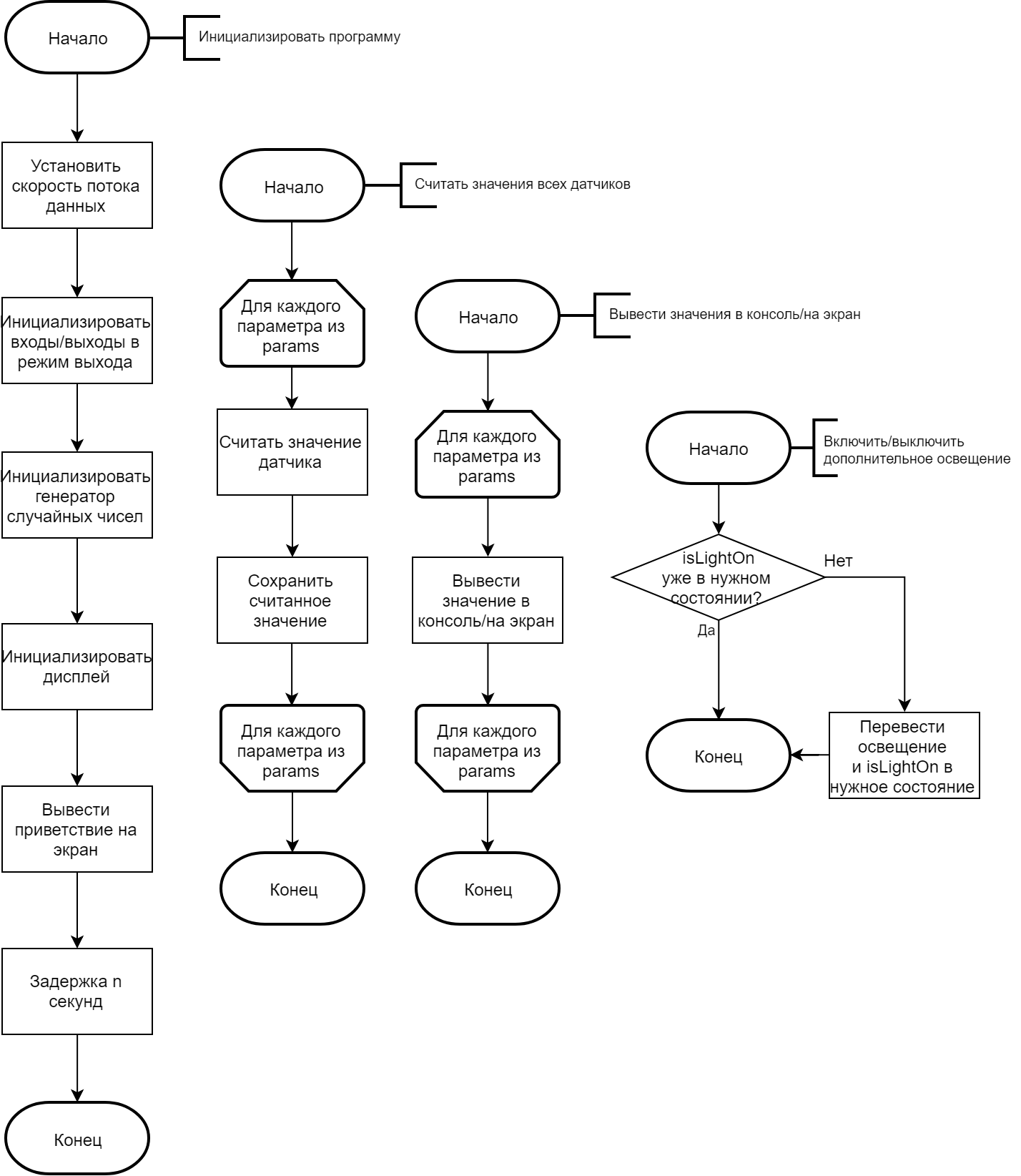
Переводит курсор на следующую строку экрана. Так как поддерживается только двухстрочных экран, перевод всегда происходит на вторую строку.

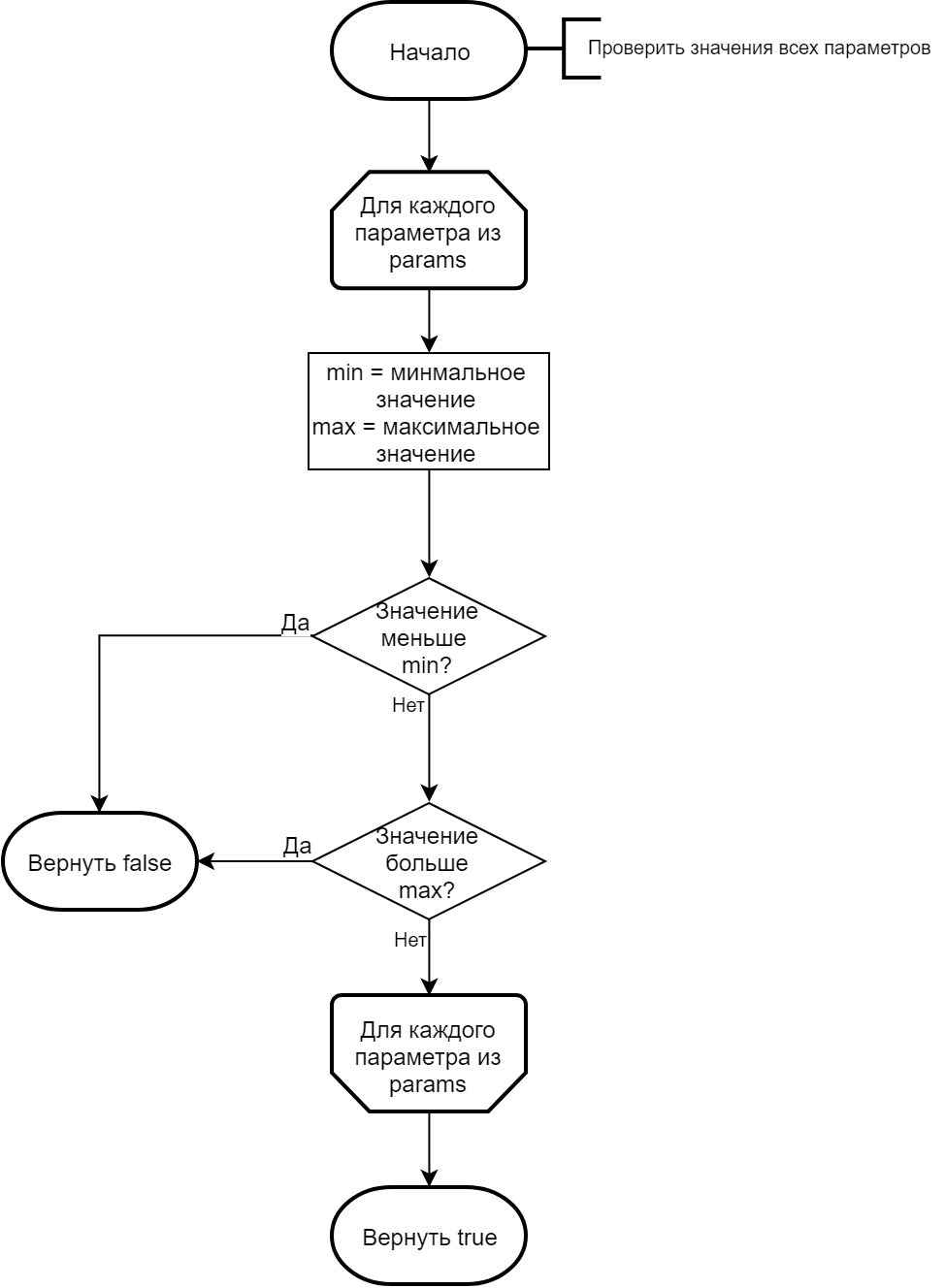
**void displayValue(String shortName, float value, int decimalPlaces)**

Выводит значение параметра на экран в формате "shortName:value ". decimalPlaces указывает число знаков значения параметра после точки, которые нужно вывести.

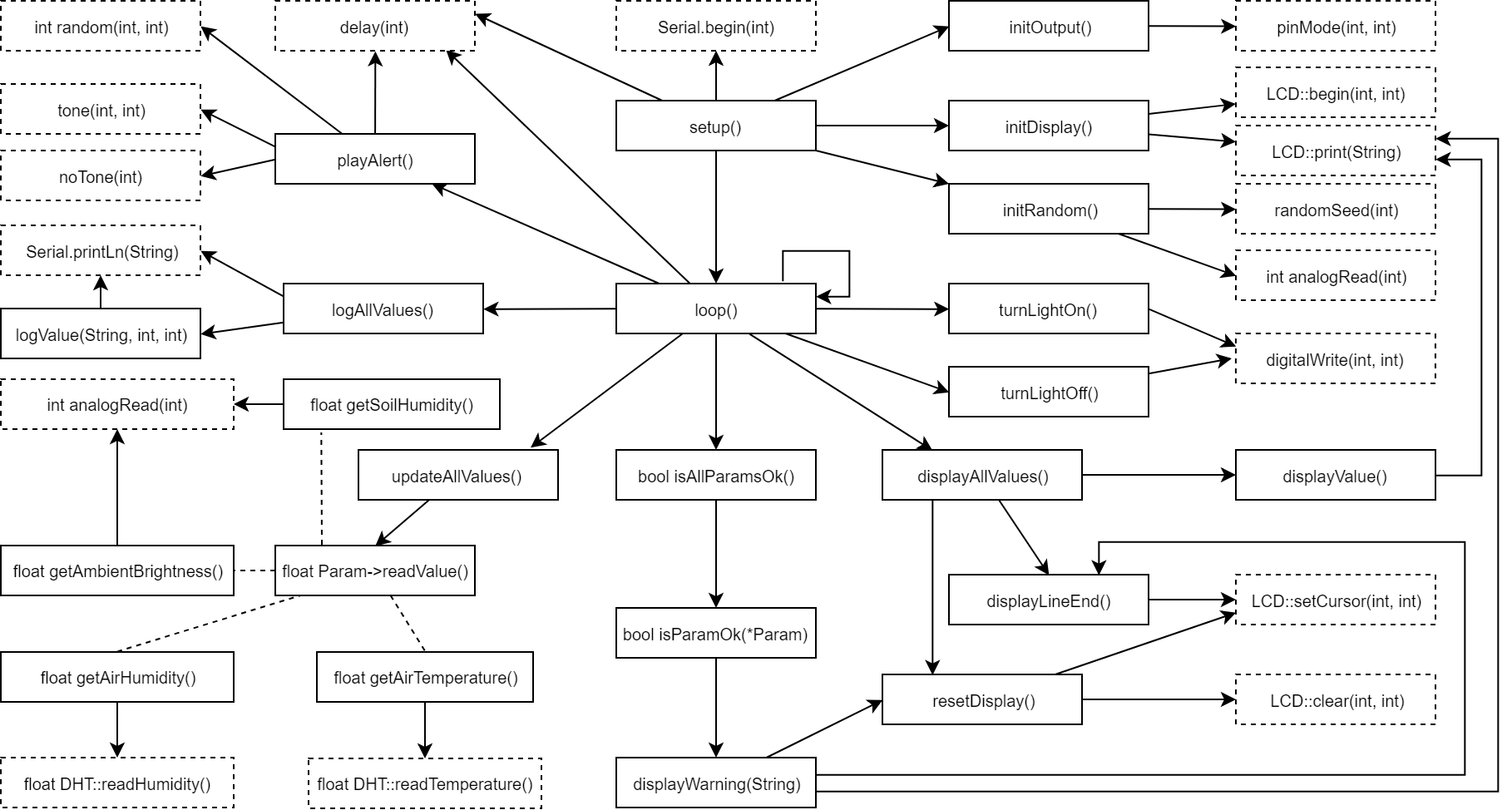
# Блок-схема функционирования ПО







# Граф вызова функций



# Методика тестирования

Для тестирования работоспособности программно-аппаратной системы были созданы следующие тест-кейсы:

**Кейс 1.**

Тестирование работы дисплея.

Действия:

1. Загрузить ПО на плату Arduino или произвести ресет платы.

Ожидание: на экране появится сообщение HELLO!

**Кейс 2.**

Тестирование считывания и вывода значений с датчиков.

Действия:

1. Загрузить ПО на плату Arduino или произвести ресет платы.
2. Подождать 2 секунды, пока пропадет приветствие.
3. Обеспечить условия, в которых все считываемые параметры будут находиться в приемлемых рамках (если необходимо, изменив минимальные и максимальные значения параметров в конфигурационном файле программы).

Ожидание: на экране появятся и будут обновляться считанные с датчиков значения.

**Кейс 3.**

Тестирование сообщений о неприемлемых значениях параметров, считанных с датчиков, и тестирование звукового оповещения.

Действия:

1. Загрузить ПО на плату Arduino или произвести ресет платы.
2. Подождать 2 секунды, пока пропадет приветствие.
3. Обеспечить условия, в которых один из считываемых параметров будет находиться в неприемлемых рамках (если необходимо, изменив минимальное или максимальное значение параметра в конфигурационном файле программы).

Ожидание: на экране появится предупреждение с сообщением о том, что нужно сделать, чтобы исправить ситуацию. Также будет воспроизведен звук сирены.

**Кейс 3.**

Тестирование возвращения к выводу значений с датчиков после того, как некорректный параметр был приведен к приемлемому значению.

Действия:

1. Загрузить ПО на плату Arduino или произвести ресет платы.
2. Подождать 2 секунды, пока пропадет приветствие.
3. Обеспечить условия, в которых один из считываемых параметров будет находиться в неприемлемых рамках (например, осветив фотосенсор чрезмерно ярким фонариком).
4. Вернуть параметр в приемлемые рамки (например, убрать фонарик от фотосенсора).
5. Подождать три секунды.

Ожидание: на экране появятся и будут обновляться считанные с датчиков значения.

**Кейс 4.**

Тестирование возвращения к выводу значений с датчиков после того, как некорректный параметр был приведен к приемлемому значению.

Действия:

1. Загрузить ПО на плату Arduino или произвести ресет платы.
2. Подождать 2 секунды, пока пропадет приветствие.
3. Обеспечить условия, в которых один из считываемых параметров будет находиться в неприемлемых рамках (например, осветив фотосенсор чрезмерно ярким фонариком).
4. Вернуть параметр в приемлемые рамки (например, убрать фонарик от фотосенсора).
5. Подождать три секунды.

Ожидание: на экране появятся и будут обновляться считанные с датчиков значения.

**Кейс 5.**

Тестирование включения и выключения дополнительного освещения (диода) при нехватке освещения.

Действия:

1. Загрузить ПО на плату Arduino или произвести ресет платы.
2. Подождать 2 секунды, пока пропадет приветствие.
3. Понизить яркость освещения, закрыв доступ фотосенсора к свету.
4. Подождать несколько секунд.
5. Вернуть яркость освещения, открыв доступ фотосенсора к свету.

Ожидание: дополнительное освещение (диод) загорится, после чего погаснет.

# Листинг кода

plantMonitor.ino

/\*

\* Plant Monitor

\* Модуль предназначен для мониторинга параметров воздуха, почвы и освещения.

\* Облегчает уход за домашними растениями, оповещая пользователя о неприемлемых для растений условиях существования.

\*

\* Используемые датчики:

\* - Датчик температуры и влажности воздуха DHT22

\* - Датчик влажности почвы

\* - Датчик яркости освещения (фоторезистор)

\*

\* Используемые переферийные устройства:

\* - LCD дисплей, способный выводить две строки текста

\* - Светодиод

\* - Пьезоизлучатель (спикер)

\*

\* Файл реализации.

\*

\* Авторы: Шибаев О.Е и Ефремов Д.Е.

\* Дата последнего изменения указана в репозитории проекта: https://github.com/unshame/arduino-thing/blob/master/plantMonitor.ino

\* Информация о лицензировании находится в файле LICENSE.txt

\*/

/\* Библиотеки \*/

// Основная библиотека Arduino

#include <Arduino.h>

// Библиотека датчика воздуха

// DHT sensor library - Version: 1.3.4

#include <DHT.h>

// Библиотека LCD дисплея

// LiquidCrystal - Version: 1.0.7

#include <LiquidCrystal.h>

// Прототипы функций и типы данных

#include "plantMonitor.h"

// Конфигурация программы (константы)

#include "plantMonitorConfig.h"

/\* Интерфейсы \*/

// Интерфейс LCD экрана

// void ::begin() - инициализирует дисплей

// void ::setCursor(int x, int y) - устанавливает курсор в указанную позицию

// void ::print(String) - выводит строку в текущей позиции курсора

// void ::clear() - очищает экран дисплея

LiquidCrystal lcd(PIN\_LCD\_1, PIN\_LCD\_2, PIN\_LCD\_3, PIN\_LCD\_4, PIN\_LCD\_5, PIN\_LCD\_6);

// Интерфейс датчика воздуха

// float ::readHumidity() - считывает слажность воздуха

// float ::readTemperature() - считывает температуру воздуха

DHT dht(PIN\_D\_DHT, DHT\_TYPE);

/\* Параметры \*/

// Влажность почвы

Param soilHumidity {

"Soil Humidity", "SH",

SH\_MIN, SH\_MAX,

"WATER THE PLANTS", "TOO MUCH WATER",

// Здесь чем меньше значение, тем больше влажность,

// поэтому сообщения об исбытке/недостатке влажности поменяны местами

true,

0,

getSoilHumidity

};

// Яркость освещения

// Недостаток освещения проверяется отдельно, поэтому здесь опущено минимальное значение

Param ambientBrightness {

"Ambient Brightness", "AB",

-1, AB\_MAX,

"", "LIGHT TOO BRIGHT",

false, 0,

getAmbientBrightness

};

// Влажность воздуха

Param airHumidity {

"Air Humidity", "AH",

AH\_MIN, AH\_MAX,

"AIR IS TOO DRY", "AIR IS TOO MOIST",

false, 1,

getAirHumidity

};

// Температура воздуха

Param airTemperature {

"Air Temperature", "AT",

AT\_MIN, AT\_MAX,

"AIR IS TOO COLD", "AIR IS TOO HOT",

false, 1,

getAirTemperature

};

// Массив указателей на структуры данных параметров

// Нужен для итерации по всем параметрам

Param \*params[] = {

&soilHumidity,

&ambientBrightness,

&airHumidity,

&airTemperature

};

// Кол-во параметров

const int numParams = sizeof(params) / sizeof(Param\*);

// Включено ли дополнительное освещение (диод)

bool isLightOn = false;

/\* INIT \*/

// Инициализирует программу, выполняется один раз при подаче напряжения или сбросы платы Arduino

void setup() {

// Устанавливаем скорость потока данных в битах в секунду

Serial.begin(DATA\_RATE);

// Инициализируем пины, генератор случайных чисел и LCD дисплей

initOutput();

initRandom();

initDisplay();

// Задержка для вывода приветствия

delay(INIT\_DELAY);

}

// Устанавливает пины спикера и диода на режим выхода

void initOutput() {

pinMode(PIN\_D\_BUZZER, OUTPUT);

pinMode(PIN\_D\_LIGHT, OUTPUT);

}

// Инициализирует генератор случайных чисел информацией из неподключенного аналогового пина

// random используется для вывода звуков случайной частоты функцией alert()

void initRandom() {

randomSeed(analogRead(PIN\_A\_UNUSED));

}

// Инициализирует размеры дисплея и выводит приветствие на экран

void initDisplay() {

lcd.begin(DISPLAY\_WIDTH, DISPLAY\_HEIGHT);

lcd.print(DISPLAY\_GREETING);

}

/\* LOOP \*/

// Считывает, анализирует и выводит текущие значения подключенных датчиков

// Выполняется после функции setup() в бесконечном цикле

void loop() {

// Считываем значения всех датчиков и выводим их в консоль

updateAllValues();

logAllValues();

// Отдельно проверяем, что яркость освещения не меньше минимального значения,..

bool isLightTooLow = ambientBrightness.cachedValue < AB\_MIN;

// ...и соотвественно включаем/выключаем дополнительное освещение (диод)

if (isLightTooLow) {

turnLightOn();

}

else {

turnLightOff();

}

// Проверяем, что значения всех параметров попадают в приемлемые границы

// Для первого неприемлемого значения будет выведено соответствующее сообщение

if (isAllParamsOk()) {

// Если все параметры успешно прошли проверку, выводим их

displayAllValues();

// Задерживаем повторное считывание параметров

delay(UPDATE\_INTERVAL);

}

else {

// Если один из параметров провалил проверку, оповещаем пользователя звуком

playAlert();

// Здесь нет смысла задерживать следующую проверку,

// т.к. проигрывание звука играет роль задержки

}

}

/\* LIGHT \*/

// Включает дополнительное освещение (диод), если оно уже не включено

void turnLightOn() {

if (!isLightOn) {

digitalWrite(PIN\_D\_LIGHT, HIGH); // Подаем максимальный уровень сигнала на пин диода

isLightOn = true;

}

}

// Выключает дополнительное освещение (диод), если оно включено

void turnLightOff() {

if (isLightOn) {

digitalWrite(PIN\_D\_LIGHT, LOW); // Подаем минимальный уровень сигнала на пин диода

isLightOn = false;

}

}

/\* SOUND \*/

// Оповещает пользователя звуком, имитирующим сирену

void playAlert() {

// Случайные частоты звука в границах разумного

// Звук будет проигрываться в пределах этих частот

int rndStart = random(10, 100);

int rndEnd = random(rndStart + 50, rndStart + 100);

// Мы будем проигрывать очень короткие звуки, чтобы симулировать плавное изменение высоты звука

int delayDuration = random(5, 20);

// Количество подъемов и спусков высоты звука

int numCycles = 3;

// Запускаем сирену, подавая плавно изменяющиеся значения напряжения на пин спикера

for (int j = 0; j < numCycles; j++) {

for (int i = rndStart; i <= rndEnd; i++) {

tone(PIN\_D\_BUZZER, i \* 10);

delay(delayDuration);

}

for (int i = rndEnd; i >= rndStart; i--) {

tone(PIN\_D\_BUZZER, i \* 10);

delay(delayDuration);

}

}

// Выключаем сирену, снимая напряжение с пина спикера

noTone(PIN\_D\_BUZZER);

}

/\* CONSOLE \*/

// Выводит значение всех параметров в консоль

void logAllValues() {

for (int i = 0; i < numParams; i++) {

logValue(

params[i]->name,

params[i]->cachedValue,

params[i]->decimalPlaces

);

}

Serial.println();

}

// Выводит значение параметра в консоль в формате "name: value"

// decimalPlaces указывает число знаков значения параметра после точки, которые нужно вывести

void logValue(String name, float value, int decimalPlaces) {

Serial.println(String(name + ": ") + String(value, decimalPlaces));

}

/\* GET VALUES \*/

// Считывает значения всех датчиков и кеширует их в структурах параметров

void updateAllValues() {

for (int i = 0; i < numParams; i++) {

params[i]->cachedValue = params[i]->readValue();

}

}

// Считывает и возвращает влажность почвы от 0 до 1023

// 0 - влажная

// 1023 - сухая

float getSoilHumidity() {

return (float)analogRead(PIN\_A\_SOILHUMIDITY);

}

// Считывает и возвращает яркость освещения от 0 до 1023

// 0 - темно

// 1023 - светло

float getAmbientBrightness() {

return (float)analogRead(PIN\_A\_PHOTOSENSOR);

}

// Считывает и возвращает влажность воздуха от 0 до 100%

float getAirHumidity() {

return dht.readHumidity();

}

// Считывает и возвращает температуру воздуха от -40 до +150 градусов цельсия

float getAirTemperature() {

return dht.readTemperature();

}

/\* CHECK VALUES \*/

// Проверяет и возвращает, попадают ли все параметры в приемлемые рамки

// Если нет, то на экран будет выведено предупреждение для первого параметра, вышедшего за рамки дозволенного

bool isAllParamsOk() {

for (int i = 0; i < numParams; i++) {

if ( !isParamOk(params[i]) ) {

return false;

}

}

return true;

}

// Проверяет и возвращает, попадает ли параметр в приемлемые рамки

// Если нет, то выводит на экран выведено предупреждение для первого параметра, вышедшего за рамки дозволенного

// Принимает ссылку на структуру с данными параметра

bool isParamOk(Param \*param) {

if ( param->cachedValue < param->minValue ) {

// Значение параметра ниже дозволенного

displayWarning(

param->invertMinMax

? param->messageExceeding

: param->messageLacking

);

return false;

}

else if ( param->cachedValue > param->maxValue ) {

// Значение параметра выше дозволенного

displayWarning(

param->invertMinMax

? param->messageLacking

: param->messageExceeding

);

return false;

}

// С параметром все ок

return true;

}

/\* DISPLAY \*/

// Очищает дисплей и устанавливает курсор в начальную позицию

void resetDisplay() {

lcd.clear();

lcd.setCursor(0, 0);

}

// Выводит значения всех параметров с их сокращенным названием по два на строке экрана

void displayAllValues() {

resetDisplay();

for (int i = 0; i < numParams; i++) {

displayValue(

params[i]->shortName,

params[i]->cachedValue,

params[i]->decimalPlaces

);

// Переходим на следующую строку экрана

if (i % 2 != 0) {

displayLineEnd();

}

}

}

// Выводит переданное предупреждение на экран

void displayWarning(String warning) {

resetDisplay();

lcd.print("WARNING!");

displayLineEnd();

lcd.print(warning);

}

// Переводит курсор на следующую строку экрана

// Так как поддерживается только двухстрочных экран, перевод всегда происходит на вторую строку

void displayLineEnd() {

lcd.setCursor(0, 1);

}

// Выводит значение параметра на экран в формате "shortName:value "

// decimalPlaces указывает число знаков значения параметра после точки, которые нужно вывести

void displayValue(String shortName, float value, int decimalPlaces) {

lcd.print(shortName + ":" + String(value, decimalPlaces) + " ");

}

plantMonitor.h

/\*

\* Plant Monitor

\* Модуль предназначен для мониторинга параметров воздуха, почвы и освещения.

\* Облегчает уход за домашними растениями, оповещая пользователя о неприемлемых для растений условиях существования.

\*

\* Заголовочный файл.

\*

\* Авторы: Шибаев О.Е и Ефремов Д.Е.

\* Дата последнего изменения указана в репозитории проекта: https://github.com/unshame/arduino-thing/blob/master/plantMonitor.h

\* Информация о лицензировании находится в файле LICENSE.txt

\*/

// Структура, хранящая информацию о считываемых параметрах

struct Param {

const String name; // Полное название параметра для вывода в консоль

const String shortName; // Сокращенное название параметра для отображения на экране

const int minValue; // Минимальное приемлемое значение параметра

const int maxValue; // Максимальное приемлемое значение параметра

const String messageLacking; // Сообщение, выводимое на экран, когда значение параметра меньше минимального

const String messageExceeding; // Сообщение, выводимое на экран, когда значение параметра больше максимального

const bool invertMinMax; // Меняет местами messageLacking и messageExceeding

const int decimalPlaces; // Сколько знаков после точки будет выводиться на экран для значения параметра

// Функция для считывания значения параметра - возвращает значение в виде числа с плавающей точкой

float (\*readValue)();

// Здесь будет кешироваться значение, возвращенное из readValue, чтобы не считывать значение несколько раз

float cachedValue;

};

/\*

\* Далее идут Прототипы всех функций, использующихся в программе.

\* Комментарии смотреть в файле реализации.

\*/

/\* SETUP \*/

void setup();

/\* INIT \*/

void initOutput();

void initRandom();

void initDisplay();

/\* LOOP \*/

void loop();

/\* LIGHT \*/

void turnLightOn();

void turnLightOff();

/\* SOUND \*/

void playAlert();

/\* CONSOLE \*/

void writeAllValues();

void writeValue(String str, float value, int decimalPlaces);

/\* GET VALUES \*/

void updateAllValues();

float getSoilHumidity();

float getAmbientBrightness();

float getAirHumidity();

float getAirTemperature();

/\* CHECK VALUES \*/

bool isAllParamsOk();

bool isParamOk(Param \*param);

/\* DISPLAY \*/

void resetDisplay();

void displayAllValues();

void displayWarning(String warning);

void displayLineEnd();

void displayValue(String abr, float value, int decimalPlaces);

plantMonitorConfig.h

/\*

\* Plant Monitor

\* Модуль предназначен для мониторинга параметров воздуха, почвы и освещения.

\* Облегчает уход за домашними растениями, оповещая пользователя о неприемлемых для растений условиях существования.

\*

\* Заголовочный файл конфигурации.

\*

\* Авторы: Шибаев О.Е и Ефремов Д.Е.

\* Дата последнего изменения указана в репозитории проекта: https://github.com/unshame/arduino-thing/blob/master/platMonitorConfig.h

\* Информация о лицензировании находится в файле LICENSE.txt

\*/

// PARAM CONSTRAINTS

// Миммальные и максимальные значения параметров

// Влажность почвы

#define SH\_MIN 220

#define SH\_MAX 600

// Яркость освещения

#define AB\_MIN 50

#define AB\_MAX 100

// Влажность воздуха

#define AH\_MIN 20

#define AH\_MAX 70

// Температура воздуха

#define AT\_MIN 22

#define AT\_MAX 40

// DHT

#define DHT\_TYPE DHT22 // Тип датчика плажности и температуры воздуха

// INPUT PINS

// Номера входов/выходов (пинов), использующихся в режиме входа

// A - аналоговый сигнал, D - цифровой сигнал

#define PIN\_A\_PHOTOSENSOR 1 // Пин датчика яркости освещения

#define PIN\_A\_SOILHUMIDITY 0 // Пин датчика влажности почвы

#define PIN\_D\_DHT 7 // Пин датчика температуры и влажности воды

#define PIN\_A\_UNUSED 4 // Неиспользуемый пин для инициализации генератора случайных чисел

// OUTPUT PINS

// Номера входов/выходов (пинов), использующихся в режиме выхода

// A - аналоговый сигнал, D - цифровой сигнал

#define PIN\_D\_BUZZER 6 // Пин спикера (пьезоизлучателя)

#define PIN\_D\_LIGHT 13 // Пин дополнительного освещения (диода)

// LCD DISPLAY PINS

// Номера входов/выходов (пинов) цифрового сигнала, использующихся LCD экраном

#define PIN\_LCD\_1 12

#define PIN\_LCD\_2 11

#define PIN\_LCD\_3 5

#define PIN\_LCD\_4 4

#define PIN\_LCD\_5 3

#define PIN\_LCD\_6 2

// SYSTEM CONFIG

// Системная конфигурация программы

#define DATA\_RATE 9600 // Скорость потока данных в битах в секунду

#define INIT\_DELAY 2000 // Задержка перед началом проверки значений датчиков

#define UPDATE\_INTERVAL 1000 // Интервал проверки значений датчиков

// DISPLAY

// Конфигурация дисплея

#define DISPLAY\_WIDTH 16 // Ширина дисплея в символах

#define DISPLAY\_HEIGHT 2 // Высота дисплея в строках - на данный момент поддерживается только 2 строки

#define DISPLAY\_GREETING "HELLO!" // Сообщение, выводимое при включении устройства

LICENSE.txt

Copyright (c) 2019 Шибаев О.Е и Ефремов Д.Е.

Данная лицензия разрешает лицам, получившим копию данного программного обеспечения

и сопутствующей документации (в дальнейшем именуемыми «Программное Обеспечение»),

безвозмездно использовать Программное Обеспечение без ограничений, включая неограниченное

право на использование, копирование, изменение, слияние, публикацию, распространение,

сублицензирование и/или продажу копий Программного Обеспечения, а также лицам, которым

предоставляется данное Программное Обеспечение, при соблюдении следующих условий:

Указанное выше уведомление об авторском праве и данные условия должны быть включены во все

копии или значимые части данного Программного Обеспечения.

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО

ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО

ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. НИ В КАКОМ

СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА УЩЕРБ

ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, В ТОМ ЧИСЛЕ, ПРИ ДЕЙСТВИИ КОНТРАКТА, ДЕЛИКТЕ ИЛИ ИНОЙ СИТУАЦИИ,

ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ

ОБЕСПЕЧЕНИЕМ.