ОГЛАВЛЕНИЕ

ПРОЕКТИРОВАНИЕ СИСТЕМЫ

1 Техническое задание

Тема 3. Комфортный дом

Для комфортного проживания в доме, нужно чтобы воздух был чистым, умеренно влажным и теплым, а также были растения. Но за этим всегда нужен контроль. Поэтому отслеживайте влажность и температуру воздуха. Освещенность в комнате. Чтобы растения всегда были вовремя политы, нужно сообщать пользователю очень открыто и явно, если показатели вышли за границы, если нет, просто выводить на дисплей, чтобы пользователь был ознакомлен с ситуацией. Если света в комнате становится мало, то включить свет.

2 Фактический перечень использованных датчиков для съема данных

DHT22 Temperature-Humidity Sensor

Датчик измерения влажности и температуры воздуха [1]. Используется библиотека DHT.h. Способ подключения следующий:

Таблица 1

Датчик DHT22	Arduino
+	+5V
Out	Например D2
-	GND

Датчик влажности почвы

Модуль состоит из двух частей: контактного щупа YL-69 и датчика YL-38 [2]. Между двумя электродами щупа YL-69 создаётся небольшое напряжение. Если почва сухая, сопротивление велико и ток будет меньше. Если земля влажная — сопротивление меньше, ток — чуть больше. По итоговому аналоговому сигналу можно судить о степени влажности. Щуп YL-69 соединен с датчиком YL-38 по двум проводам. Кроме контактов соединения с щупом, датчик YL-38 имеет четыре контакта для подключения к контроллеру. Библиотека не использовалась. Способ подключения следующий: датчик подключаем последовательно к щупу, а дальше см. Таблицу 2

Таблица 2

YL-69(щуп) к Arduino	Arduino
OUT A	Например A0
OUT D	Не использовался
-	GND
+	5V

Фоторезистор

Датчик используется для измерения количества света в комнате, функционирует по принципу резистора, т.е. меняет свое сопротивление в зависимости от уровня окружающего освещения [3]. Библиотека не использовалась. Способ подключения следующий:

Таблица 3

Фоторезистор	Резистор 10 кОм	Arduino
-	A0	GND
+		5V

3 Фактический перечень использованных устройств управления/отображения

MT-16S2H-2YLG

Жидкокристаллический модуль MT–16S2H состоит из БИС контроллера управления и ЖК панели. Дисплей MT-16S2H предназначен для вывода текста на латинице и кириллице. Экран имеет 16 контактов для питания логики, взаимодействия с управляющей электроникой и подсветки [4]. Используется библиотека LiquidCrystal.h. Способ подключения следующий:

Таблица 4

MT–16S2H	Arduino
GND	GND
Vcc	5V
Vo	GND
RS	D12
R/W	GND
E	D11
DB0	
DB1	
DB2	
DB3	
DB4	D5
DB5	D4
DB6	D3
DB7	D2
Vcc	5V
GND	GND

Ссылка на подробную информацию:

Светодиод

Светодиод — это устройство, которое представляет собой полупроводниковый прибор, способный излучать свет при пропускании через него электрического тока в прямом направлении (от анода к катоду) [5]. Библиотека не использовалась. Способ подключения следующий:

Таблица 5

Светодиод	Резистор 220 Ом	Arduino
Анод	Резистор 220 Ом	GND
Катод		Например D1

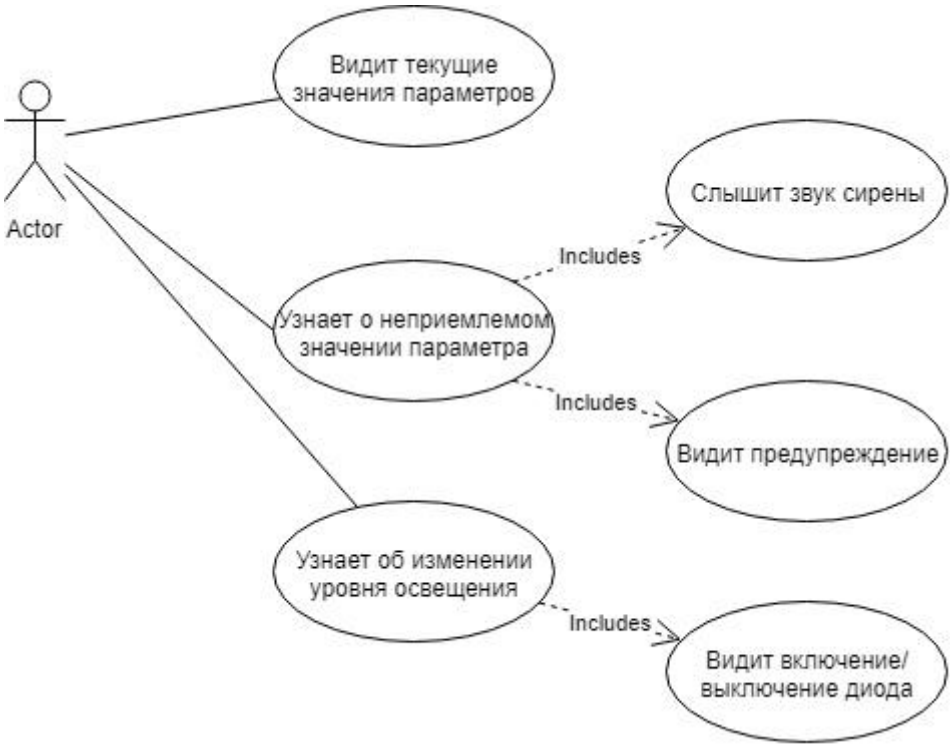
Пьезоизлучатель

Принцип действия его основан на том, что под действием электрического поля возникает механическое движение мембраны, которое и вызывает слышимые нами звуковые волны. Обычно такие излучатели звука устанавливают в бытовую электронную аппаратуру в качестве звуковых сигнализаторов, в корпуса настольных персональных компьютеров, в телефоны, в игрушки, в громкоговорители и много куда ещё [6]. Библиотека не использовалась. Способ подключения следующий:

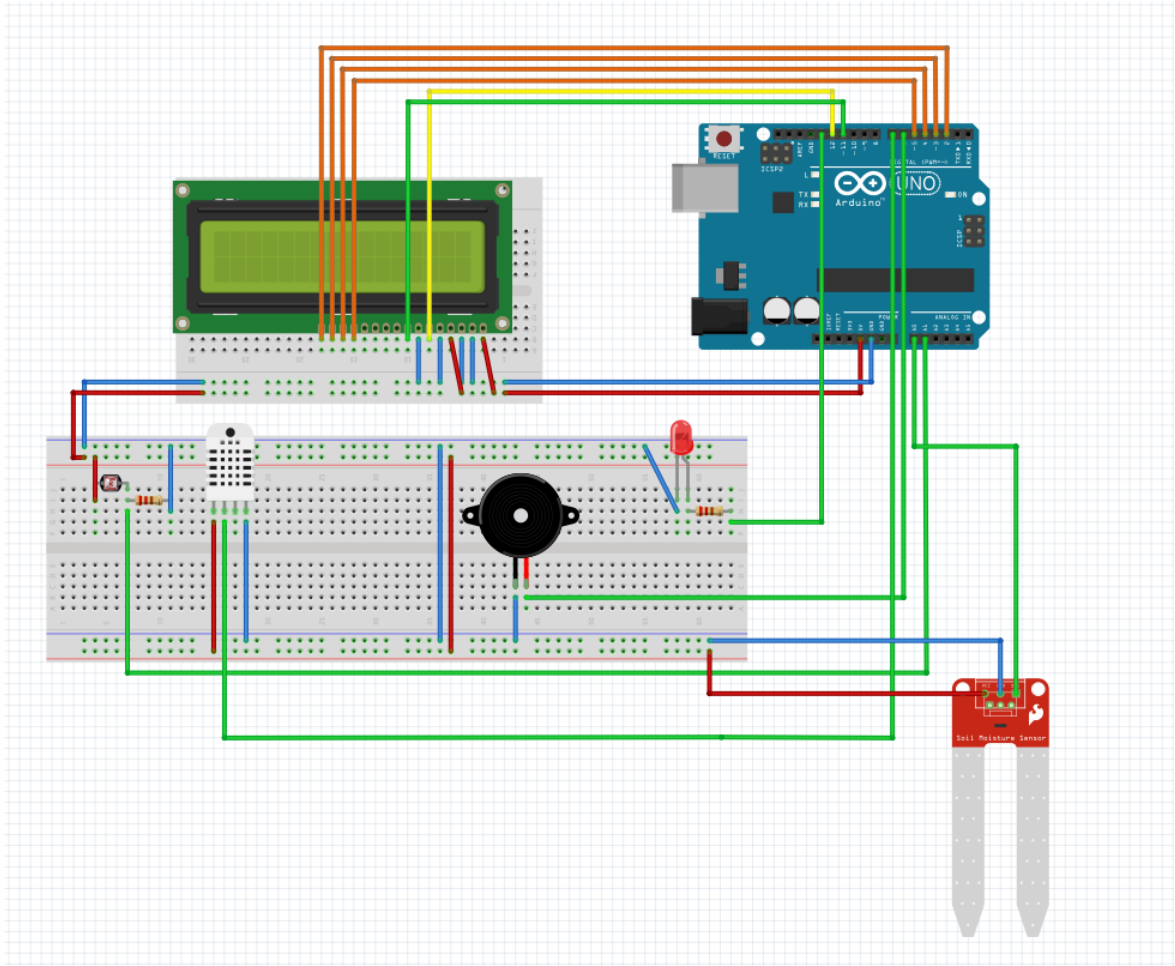
Таблица 6

Пьезоизлучатель	Arduino
Анод	GND
Катод	Например D3

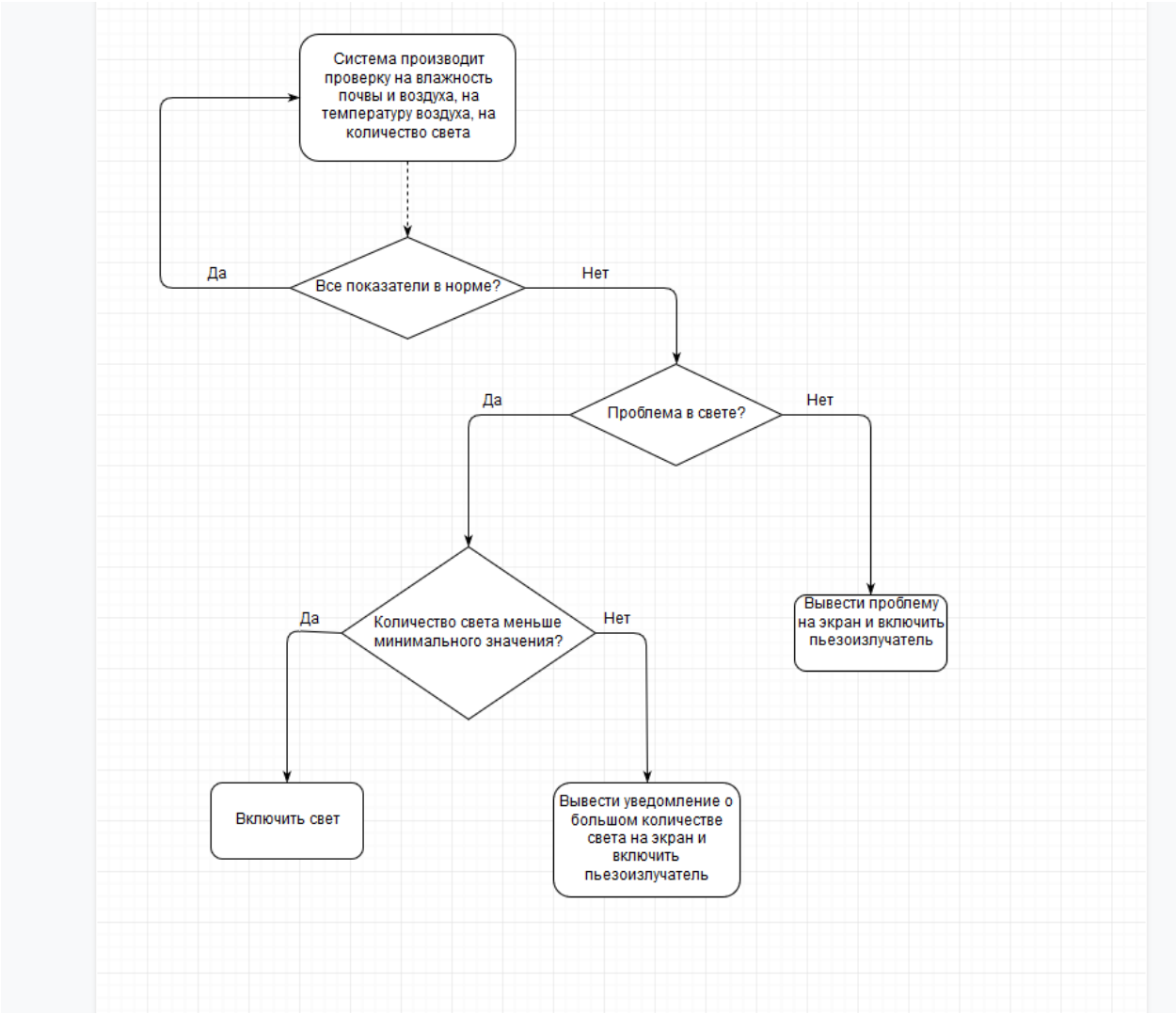
4 Диаграмма прецедентов



5 Описание аппаратной подсистемы



6 Обобщенный алгоритм



РАЗВЕРТЫВАНИЕ СИСТЕМЫ

1 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 [7]. Файл: DHT_sensor_library-1.3.4.zip

LiquidCrystal - Version: 1.0.7

Библиотека LCD дисплея [8]. Файл: LiquidCrystal-1.0.7.zip

2 Инструкция по установке библиотек

1. Скачать и установить Arduino IDE [10].
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.

3 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.

4 Инструкция по установке и запуску

1. Экспортировать файлы проекта в произвольную папку.
2. Скачать и установить Arduino IDE [9].
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

1 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.

2 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.

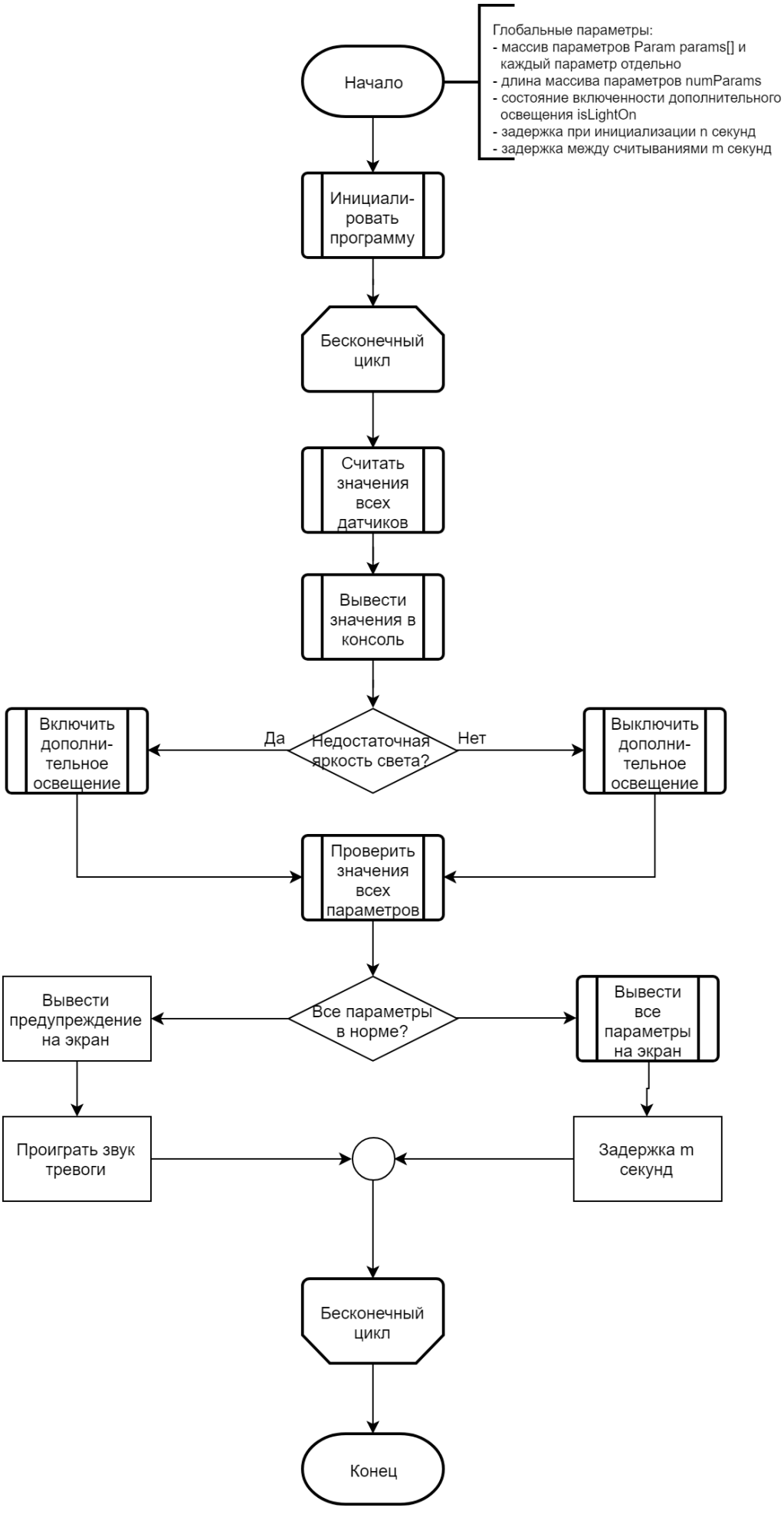
void displayLineEnd()

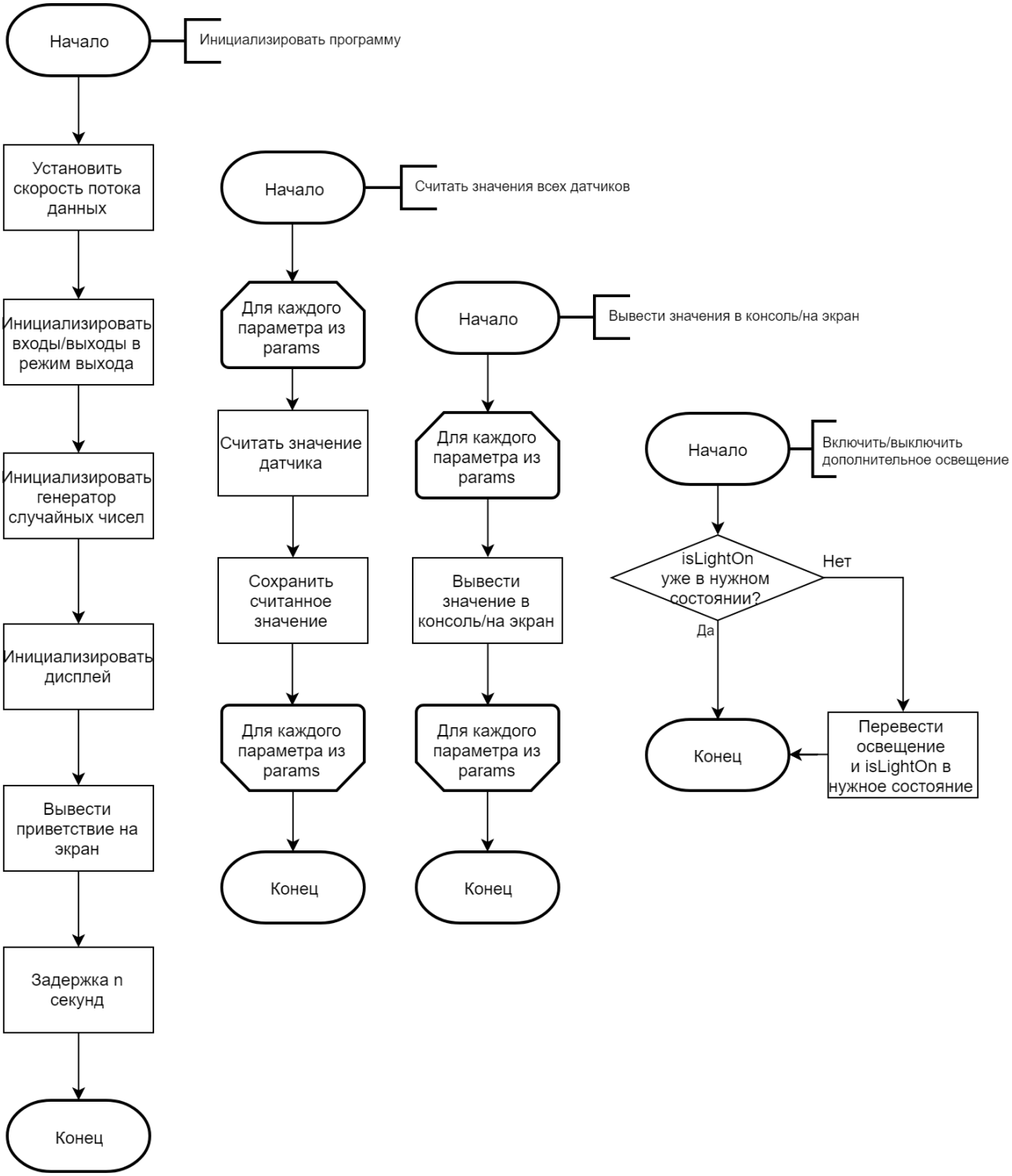
Переводит курсор на следующую строку экрана. Так как поддерживается только двухстрочных экран, перевод всегда происходит на вторую строку.

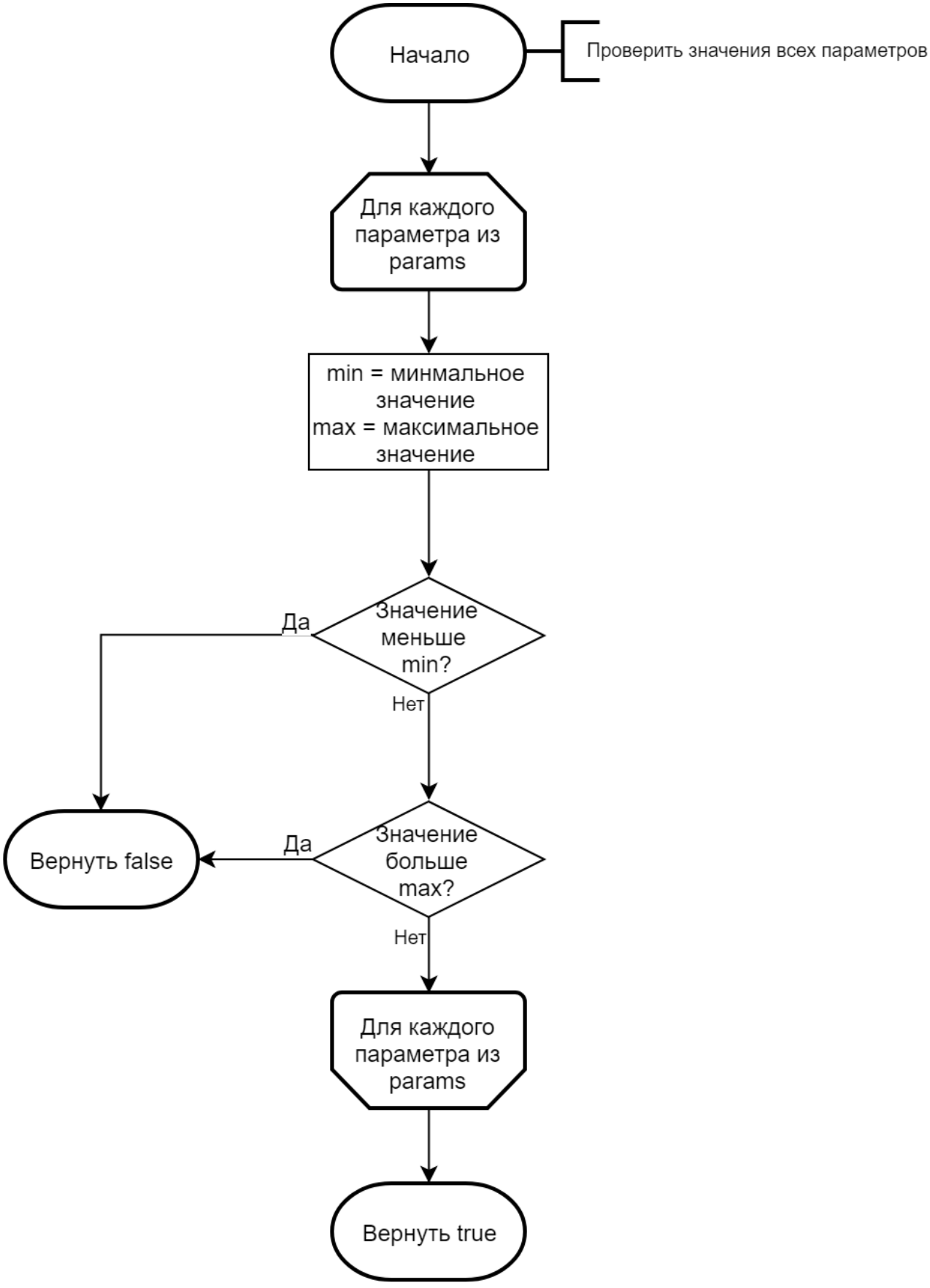
void displayValue(String shortName, float value, int decimalPlaces)

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

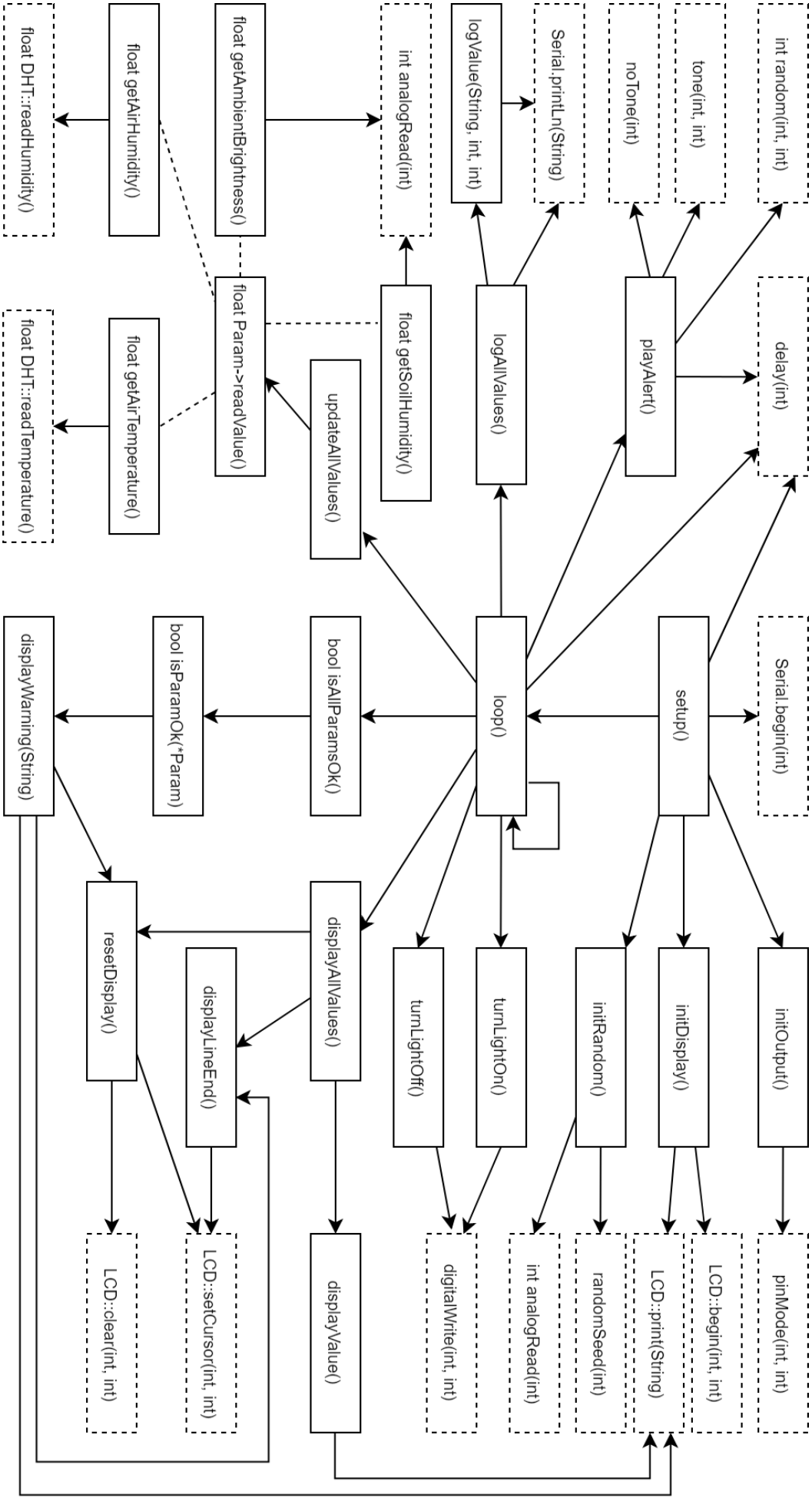
3 Блок-схема функционирования ПО




Начало
Инициализировать программу
Установить скорость потока данных
Инициализировать входы/выходы в режим выхода
Инициализировать генератор случайных чисел
Инициализировать дисплей
Вывести приветствие на экран
Задержка n секунд
Конец
Начало
Считать значения всех датчиков
Для каждого параметра из params
Считать значение датчика
Сохранить считанное значение
Для каждого параметра из params
Конец
Начало
Вывести значения в консоль/на экран
Для каждого параметра из params
Вывести значение в консоль/на экран
Для каждого параметра из params
Конец
Начало
Включить/выключить дополнительное освещение
isLightOn уже в нужном состоянии?
Нет
Да
Конец
Перевести освещение и isLightOn в нужное состояние


Начало
Проверить значения всех параметров
Для каждого параметра из params
min = минмальное значение
max = максимальное значение
Значение меньше min?
Да
Нет
Значение больше max?
Да
Нет
Вернуть false
Для каждого параметра из params
Вернуть true

4 Граф вызова функций



5 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.

6 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

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Temperature Monitoring With DHT22 & Arduino [Электронный ресурс]. – https://www.hackster.io/mafzal/temperature-monitoring-with-dht22-arduino-15b013

2 Датчик влажности почвы [Электронный ресурс]. – https://3d-diy.ru/wiki/arduino-datchiki/datchik-vlazhnosti-pochvy-arduino/

3 Подключение фоторезистора к ардуино и работа с датчиком освещенности [Электронный ресурс]. – https://arduinomaster.ru/datchiki-arduino/photorezistor-arduino-datchik-sveta/

4 Текстовый экран 16×2 [Электронный ресурс]. – http://wiki.amperka.ru/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D1%8B:text-lcd-16x2

5 Ардуино: управление светодиодом [Электронный ресурс]. – http://robotclass.ru/tutorials/arduino-led/

6 КАК ПОДКЛЮЧИТЬ ПЬЕЗОИЗЛУЧАТЕЛЬ (ПЬЕЗОПИЩАЛКУ) К ARDUINO [Электронный ресурс]. – https://soltau.ru/index.php/arduino/item/357-how-connect-buzzer

7 Adafruit DHT Humidity & Temperature Sensor Library [Электронный ресурс]. – https://github.com/adafruit/DHT-sensor-library

8 Liquid Crystal Library for Arduino [Электронный ресурс]. – https://github.com/arduino-libraries/LiquidCrystal

9 Install the Arduino Software (IDE) on Windows PCs [Электронный ресурс]. – https://www.arduino.cc/en/guide/windows

10 Видеоролик демонстрации работы [Электронный ресурс]. - https://www.youtube.com/watch?v=bXN5LlaczY0