Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения материалов и транспорта
Высшая автоматизации и робототехники
Кафедра «Автономные роботы» (при ЦНИИ РТК)

Курсовая работа

Дисциплина: объектно-ориентированное программирование

Тема: разработка алгоритма работы системы автоматизированного поддержания роста растений

Студент гр. 3331506/00401 Конев И. А.

Преподаватель Ананьевский М. С.

Санкт-Петербург
2023

Оглавление

Цель и задачи

Задачей курсовой работы является написание программного обеспечения для микроконтроллера, являющегося руководящим устройством системы автоматизированного поддержания роста растений. Данная система должна обеспечивать комфортные условия для роста растений. К ним относятся влажность почвы, интенсивность искусственного освещения и температура окружающего воздуха.

Введение

Из описания на сайте производителя следует, что данный набор предназначен для изучения основных понятий о технологии Интернет-вещей (IoT). Теплицы являются одними из самых популярных объектов, где активно применяется технология IoT [1]. Для более конкретного представления о концепции "Интернета вещей" приведем его определение [2].

Интернет вещей (англ. internet of things, IoT) — концепция сети передачи данных между физическими объектами («вещами»), оснащёнными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой. Предполагается, что организация таких сетей способна перестроить экономические и общественные процессы, исключить из части действий и операций необходимость участия человека.

Так как сама концепция умной теплицы не является инновационной, на различных интернет-ресурсах представлены образцы DIY-проектов подобных теплиц [3]. В общем, их функционал соответствует находящемуся в нашем распоряжении образцу, отличаются они в исполнении и наличием некоторых доработок. Помимо коммерческих предложений по приобретению полномасштабных теплиц для дачных участков, существует аналогичный имеющемуся набор для обучения концепции интернета вещей, конструированию и программированию от компании MGBot [4]. Входящие в его состав компоненты также схожи по выполняемым задачам за исключением применяемых материалов и способа реализации контроля температуры.

Написанное программное обеспечение загружено на программируемый контроллер Arduino UNO на базе микроконтроллера ATmega328 [5]. Данный контроллер выбран как удобный для прототипирования проектов средних размеров, которые не требуют высокой мобильности. Несомненным плюсом выбора данного микроконтроллера является наличие его исходной документации, как для аппаратной, так и для программной части, так как данный проект является «open-source» разработкой. Кроме того, открытый исходный код не влечёт за собой никаких ограничений на применение микроконтроллера в коммерческих проектах, за исключением использования торговых марок, на которые распространяются авторские права компании Arduino (название и логотип). Также высокая популярность платформы Arduino способствует появлению большого количества библиотек, форумов и ресурсов, облегчающих ее использование.

В исходную систему автоматизированного поддержания роста растений (далее — теплица) включены три независимых системы контроля за условиями окружающей среды:

- контроль освещенности (модуль фоторезистора и фитодиодная лента);
- контроль влажности почвы (датчик влажности почвы и система капельного полива);
- контроль температуры воздуха (датчик температуры и влажности воздуха и совокупность вентилятора с нагревательным элементом).

На рисунке 1 представлен общий вид теплицы.

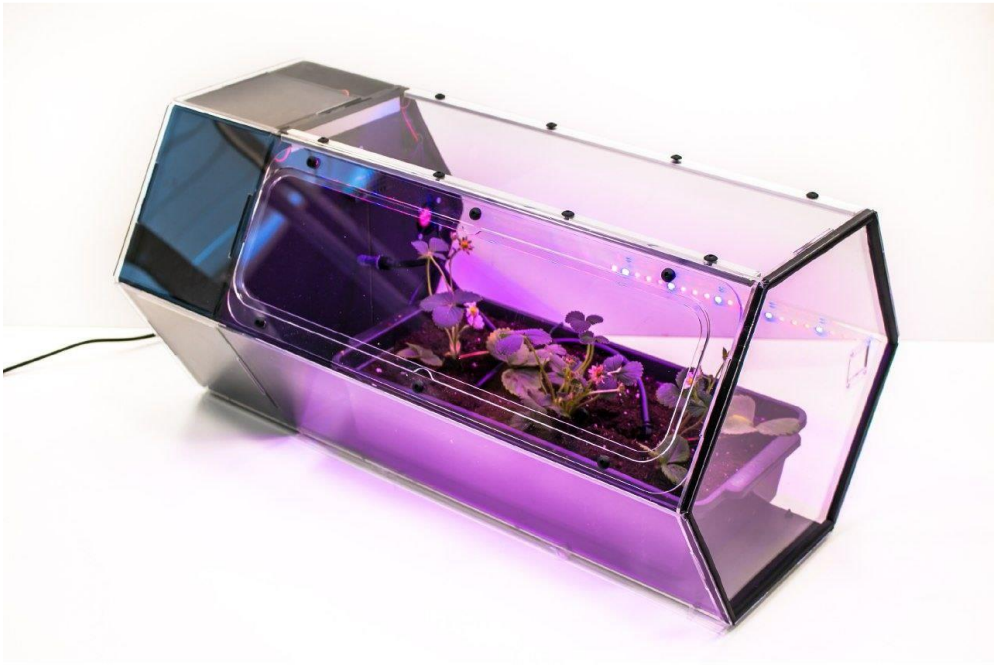

Рисунок 1 — Общий вид теплицы

Ход работы

Исследование схемы питания и элементов управления

Первым этапом работы над теплицей было исследование схемы питания теплицы и возможностей управления исполнительными устройствами. Фотография электронной оснастки теплицы представлена на рисунке 2.

Главным питающим устройством является блок питания от сети 220В с выходными характеристиками 12В и 8А. Силовая часть теплицы, включающая все ее исполнительные устройства, питается напрямую от блока питания. С помощью регулируемого понижающего DC-DC преобразователя на основе микросхемы LM2596HVS [7] входящее напряжение с блока питания понижается и создает источник для питания логических компонентов теплицы, в том числе и программируемого контроллера.

Рисунок 2 — Общий вид электронной части теплицы

На данном рисунке возможно увидеть составляющие электронной «начинки» теплицы. Здесь не представлен датчик температуры и влажности DHT-21, однако его обозначение можно видеть на общем виде теплицы (рисунок 3), как и изображение фитодиодных лент. Таким образом:

1 — Программируемый контроллер Arduino UNO

2 — Понижающий DC-DC преобразователь LM2596HVS

3 — Воздушно-водяной насос R385 с емкостью для забора воды

4 — Модуль аналогового датчика влажности почвы

5 — Система контроля температуры воздуха (вентилятор + нагревательный элемент)

6 — Модуль датчика освещенности на основе фоторезистора

7 — Блок из двух 2-канальных модулей реле SRD-05VDC-SL-C для управления исполнительными устройствами теплицы [8]

8 — Дополнительный модуль реле аналогичной модели для переподключения питания датчика DHT-21 (модификация для корректной работы, которая будет описана далее)

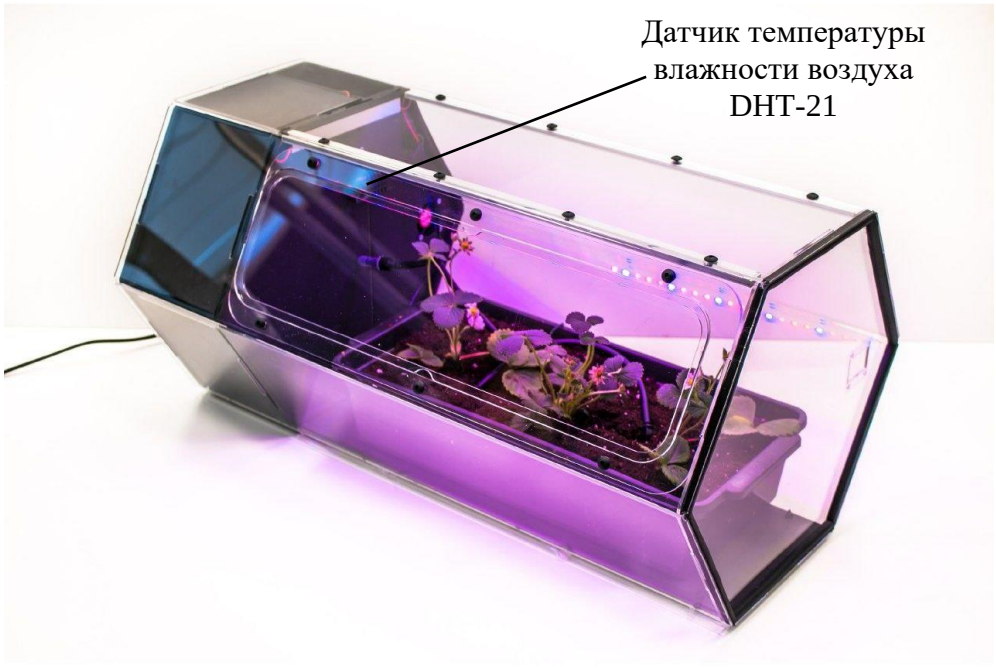

Рисунок 3 — Общий вид теплицы с обозначением датчика

Возможность управления исполнительными устройствами реализована с помощью блока из четырех электромеханических реле с опторазвязкой, то есть выходной сигнал может принимать только значения «включить» и «выключить».

Схематическое представление компонентов, входящих в конструкцию теплицы изображено на рисунке 4.

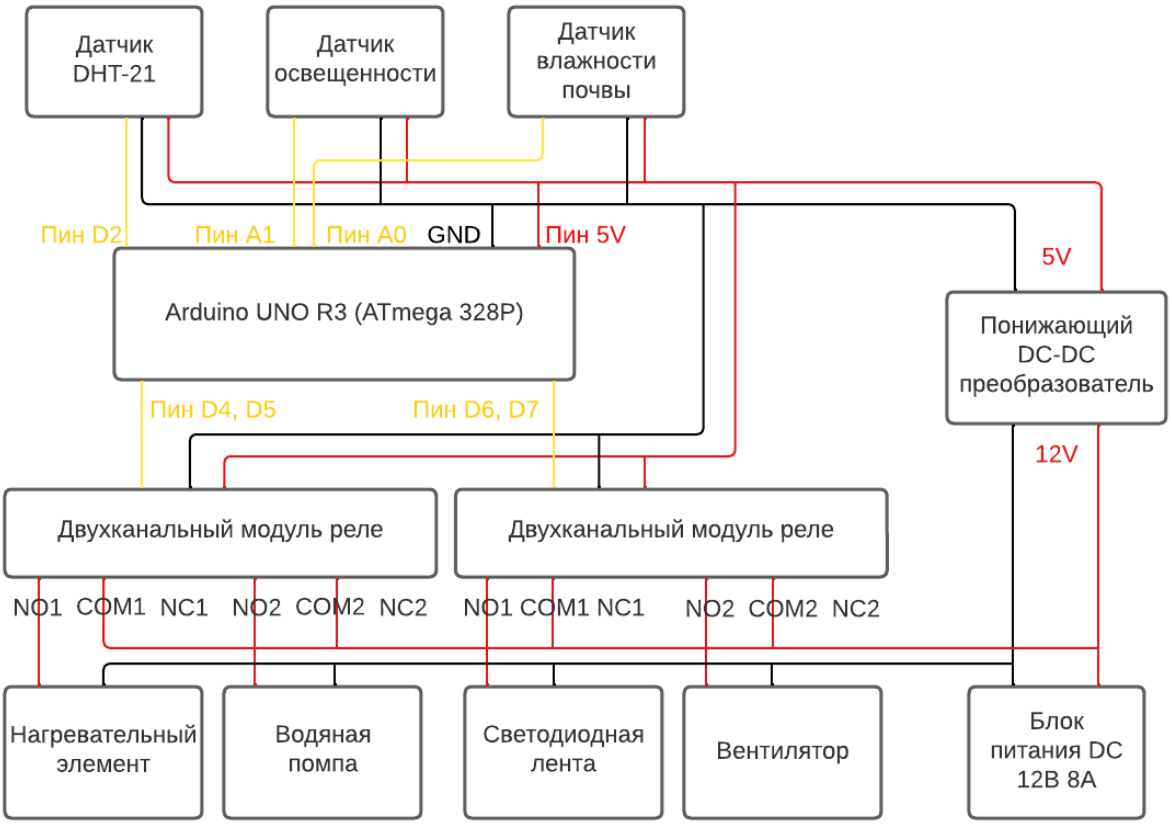

Рисунок 4 — Схематическое изображение компонентов теплицы

В дальнейшем процесс работы над каждой из систем контроля будет рассмотрен в отдельности.

Реализация контроля освещенности

На данном этапе работы было необходимо настроить реакцию системы на изменение внешней освещенности. В начала работы был выявлен один недостаток сборки аппаратной части: модуль датчика освещенности, представленный на рисунке 5, соединялся с контроллером через цифровой выход, который имеет лишь два состояния (логические «ноль» и «единица»). Низкий

логический уровень соответствует высокому уровню внешней освещенности, высокий логический уровень — низкому. Однако подключение датчика на стороне микроконтроллера осуществляется в аналоговой вход, что подразумевает использование аналогового значения во время работы. Данный недостаток не позволяет настраивать пороговый уровень срабатывания датчика программно, это возможно сделать только с помощью подстроечного резистора на плате модуля.

Рисунок 5 — Аналоговый датчик освещенности на фоторезисторе

Исходя из вышеописанных условий, был выбран следующий режим работы освещения: при появлении высокого сигнала на цифровом выходе датчика на реле отправляется высокий уровень, замыкающий разрыв цепи и подающий питание на фитодиодную ленту, пример которой представлен на рисунке 6.

Рисунок 6 — Пример фотодиодной ленты для растений

Реализация контроля влажности почвы

На этой стадии работы необходимо было настроить верный режим полива при понижении влажности почвы до порогового значения. В данной конструкции теплицы стоит отметить выбранный для измерения влажности датчика (рисунок 7). Он основан на емкостном эффекте, поэтому не подвержен износу от коррозии, так как, в отличие от датчиков резистивного типа, не проводит микротоки между электродами.

Рисунок 7 — Модуль аналогового датчика влажности почвыз

Величина показаний с датчика обратно пропорциональна влажности почвы (сигнал, равный 1023 соответствует нахождение на воздухе, равный 0 — погружению в воду). Также в систему контроля влажности почвы входит воздушно-водяной насос, представленный на рисунке 8, на основе электродвигателя постоянного тока R385.

Рисунок 8 — Воздушно-водяная помпа

Алгоритм работы системы капельного полива был выбран с учетом задержки на распространение воды до датчика. Указанные в блок-схеме временные интервалы подобраны экспериментально. Выбор периода полива основывается на количестве воды, необходимое для увлажнения объема почвы, размещенного в рабочей области теплицы. Время на равномерное распространение воды рассчитано путем проведения нескольких замеров влажности на различном удалении от точки полива через некоторые промежутки времени. Блок-схема алгоритма представлена на рисунке 9.

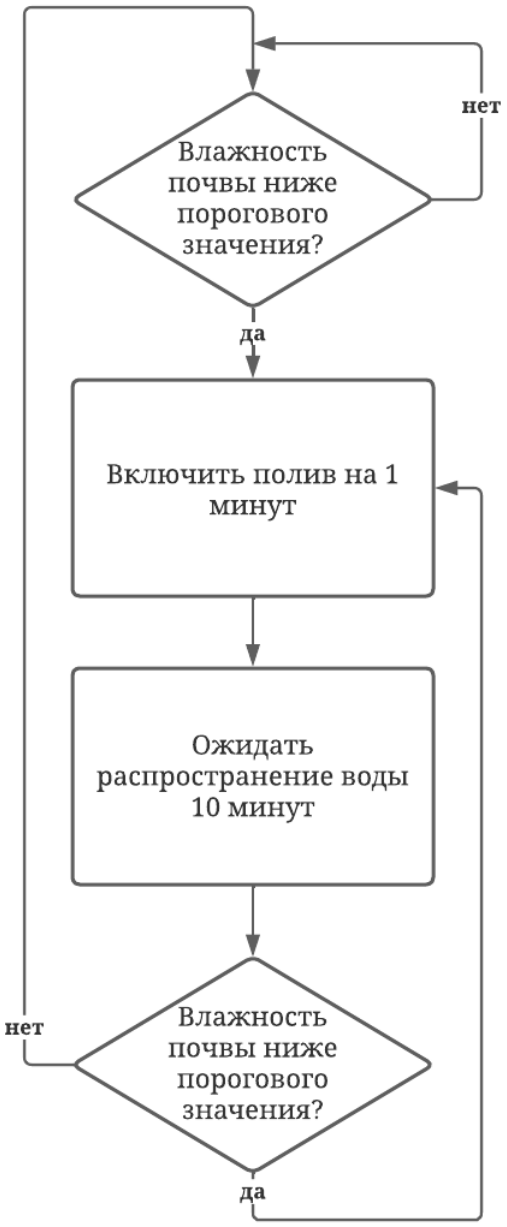

Рисунок 9 — Алгоритм работы системы капельного полива

Реализация контроля температуры воздуха

Данный этап разработки предполагает контроль за температурой непосредственно внутри рабочего пространства теплицы, где размещаются растения. Управляющая система состоит из комплексного датчика температуры и влажности воздуха DHT-21 [9] (рисунок 11), вентилятора и нагревательного элемента установленного непосредственно на пути воздушного потока, выходящего из вентилятора. Нагревательный элемент состоит из металла с высоким сопротивлением и является простейшим преобразователем электрической энергии в тепловую.

Для снятия данных с данной модели датчика требуется подключить библиотеку, реализующую специальный протокол передачи данных [10], благодаря происходит обработка 40-битового сообщения от датчика. Выбор данной модели датчика может быть не оправдан, так как младшие модели серии DHT также обладают приемлемым диапазоном измерений и точностью.

Рисунок 10 — Вентилятор постоянного тока

Совокупность включенных вентилятора и нагревательного элемента (модель вентилятора, идентичная используемой изображена на рисунке 10) дает возможность нагревать рабочее пространство теплицы. При этом сила воздушного потока является достаточно высокой, чтобы нагревательный элемент не вызывал чрезмерное повышение температуры корпуса. Одиночно включенный вентилятор позволяет понизить температуру в рабочем пространстве теплицы, чтобы избежать перегрева растений.

Рисунок 11 — Датчик модели DHT-21

Реализация алгоритма управления

Программа управления реализована с помощью среды разработки Arduino IDE [2]. Необходимая библиотека была из стандартного менеджера библиотек среды разработки. Было принято решение реализовать каждую из систем контроля в качестве отдельной функции с возможностью ввода таких параметров, как минимальная и максимальная температура воздуха и пороговое значение влажности. Для этих целей был реализован фильтр вводимых значений. При написании программы также необходимо было учитывать ограничение на период между снятиями данных с датчика температуры.

Итоги выполнения работы

Выявленные недостатки и советы по улучшению конструкции

Помимо вышеописанных недостатков конструкции и сборки, во время работы над написанием программы для работы теплицы были выявлены некоторые дополнительные ошибки проектирования.

Наиболее серьезной проблемой, ограничивающей функционал теплицы, оказалось отключение датчика температуры во время работы насоса. Возврат датчика его в рабочее состояние возможно осуществить только путем переподключения его питания либо питания всей схемы. Изначально было

выдвинуто предположение, что запуск двигателя помпы вызывает появление высокого пускового тока и, вследствие, просадку напряжения питания всей схемы.

Для предотвращения данных помех была предпринята попытка установки емких электролитических конденсаторов по питанию как помпы, так и датчика. Однако данный способ не разрешил проблему с датчиком.

Следующим возможным путем решения проблемы являлось подключение питания датчика через цифровой порт Arduino, чтобы во время работы программы была возможность «вручную» контролировать питание датчика. Данная идея также может быть реализована с помощью включения в цепь питания датчика реле. При кратковременном отключении питания в данном случае необходимо подтянуть провод питания к земле. Данные попытки решения проблемы также не принесли ощутимых результатов, но позволили гарантированно возвращать датчик в рабочее состояние после окончания работы помпы.

Подключение отдельного от общей схемы питания аккумулятора для питания датчика также было одним из вариантов решения проблемы. Попытка разведения питания датчика и всей остальной схемы не устранило ошибки в работе теплицы, что позволило сделать вывод о наличии проблем с передачей данных в контроллер. Так как датчик общается с контроллером по одному проводу, используя специальный протокол, помехи возникающие в схеме могут мешать корректной передачи значений и вызывать отключение датчика. На момент написания отчета проблема с передачей данных не решена, но было принято решение отключать контроль температуры во время процесса полива и включать датчик только после его окончания с помощью реле.

Данное решение позволяет сохранять работоспособность теплицы на протяжении практически всего времени работы за исключением периода работы помпы. Так как этот промежуток занимает малую часть от общего цикла работы теплицы, решение с подключением реле можно считать удовлетворительным в данном случае.

Кроме данной, достаточно существенной, проблемы, в процессе разработки программы было найдено еще некоторое количество недостатков в конструкции, которые могут помешать потенциальному пользователю с освоением теплицы.

Ошибки производственной сборки. Были обнаружены ошибки производственной сборки, которые заключаются в том, что на одной из экземпляров теплиц полярность подключения насоса перевернута, т. е. он качает воду в обратную сторону, и что питание на Arduino подается с выхода понижающего преобразователя, т. е. это около 5В. При этом данное напряжение подается на VIN-пин платы, подключенный к стабилизатору, где граничное входное напряжение 6–20В, а рекомендуемое — 7–15В. Скорее всего, подразумевалось подключение питания через пин 5V, который может быть, и входным, и выходным.

Кабель-менеджмент. Кабель-менеджмент электронной части теплицы не предоставляет возможности доступа к отдельным проводам, так как все провода соединяются в косу в области около клемм питания. Также это может являться источником помех для передачи данных с датчиков, что, возможно, является причиной ошибки с датчиком температуры.

Неудобство присоединения крышки. Крышка теплицы напрямую связана с корпусом теплицы при помощи двух проводов от порта для подключения блока питания. Это делает затруднительным ее отсоединение от остальной теплицы, что при ограниченной длине проводов не позволяет удобно расположить ее в рабочем пространстве.

Отсутствие свободных гнезд в клеммах. В клеммах, используемых для соединения проводов питания, не предусмотрено свободных гнезд для подключения дополнительных проводов, так как имеющиеся провода питания достаточно плотно занимают место в имеющихся гнездах.

Помимо существующих недостатков у настоящей модели теплицы, для улучшения опыта использования продукта возможно добавление некоторых элементов схемы.

1 *Подключение LCD-дисплея и кнопки*. Для оптимизации работы пользователя с теплицей возможно избавиться от необходимости изменения программного кода для изменения входных значений (нормальной температуры и влажности). Данная особенность может быть реализована при добавления внешних элементов управления алгоритмом. Например, кнопка сброса работы программы начинает выполнение алгоритма заново, при этом в функции «setup» возможно добавить ввод необходимых значений. Выбор конкретного значения реализуется с помощью потенциометра. Для внешнего контроля за теплицей возможна установка LCD-экрана.

2 *Замена модуля реле на модуль MOSFET-транзистора.* Для более точного контроля за работой исполнительных устройств возможно установить для управления ими модули MOSFET-транзисторов или драйверы. Таким образом, например, можно уменьшить скорость вращения вентилятора для регулировки интенсивности нагрева/охлаждения рабочей зоны теплицы. Также данное устройство может помочь в плавном запуске двигателя помпы, что позволило бы снизить индуктивные выбросы в цепь питания.

Реализованный алгоритм управления

В соответствии с вышеописанными особенностями конструкции и сборки был разработан следующий алгоритм управления теплицей.

```
#define HUMIDITY_SENSOR_PIN A1
#define LDR_PIN A0
#define DHT_SENSOR_PIN 2
#define DHT_SENSOR_VCC_PIN 8

#define HEATER_PIN 4
#define WATER_PUMP_PIN 5
#define LED_STRIP_PIN 6
#define FAN_PIN 7

/* DHT21 (датчик влажности и температуры) общается через
 * цифровой порт 2 по специальному протоколу, который
 * можно реализовать с помощью сторонней библиотеки */

/* LDR выдает на цифроом пине A0 0 (светло) или 1 (темно)
 * LDR должен выдавать аналоговый сигнал от 0 (светло) до 1023 (темно) */
```

```
/* Датчик влажности дает на аналоговом пине A1 значение влажности,
 * где 1023 - сухо (нет контакта между проводниками),
 * 0 - влажно (проводники замкнуты) */

/* Пин D4 отвечает за нагревательный элемент
 * !!! (включение только одновременно с вентилятором) !!!
 * Пин D5 отвечает за водяную помпу полива
 * Пин D6 отвечает за светодиодную ленту
 * Пин D7 отвечает за вентилятор */

// Задание параметров для запуска программы конкретного вида растений
// Задержка между выводом (для аналоговых) и считываниями (для цифрового) сигна-
лов с датчиков
const int DELAY = 2000;

// Граничное значение освещенности, выше которого включается фитолента
const int NORMAL_LIGHT_LVL = 500;

/* Граничные значения для температуры воздуха, ниже и выше которых должен быть
 * включен нагревательный элемент и/или вентилятор соответственно */
const float MIN_TEMP = 25.0;
const float MAX_TEMP = 40.0;

/* Граничное значения для относительной влажности почвы
 * в процентах, ниже которого должна быть включена помпа */
const int MIN_HUMIDITY = 70;
// Значения сухой и влажной почвы определяются экспериментально для расчета отно-
сительной влажности
const int DRY_SOIL_HUMID = 1023;
const int WET_SOIL_HUMID = 0;
// Время на полив и на ожидание распространения воды до датчика в миллисекундах
const unsigned long int WATERING_PERIOD = (unsigned long int)1 * 10 * 1000;
const unsigned long int WAITING_PERIOD = (unsigned long int)1 * 60 * 1000;

bool is_watering_need = false;
bool is_watering_ended = false;

#include <TroykaDHT.h>
// Подключение библиотеки для считывания данных с датчиков серии DHT
/* Создание объекта класса DHT с указанием номера цифрового
 * порта, куда подключен датчик и конкретной модели серии */
DHT dht_sensor(DHT_SENSOR_PIN, DHT21);

void setup() {
  Serial.begin(9600);
  dht_sensor.begin();
  pinMode(HUMIDITY_SENSOR_PIN, INPUT);
  pinMode(LDR_PIN, INPUT);
  for (int num = HEATER_PIN; num <= DHT_SENSOR_VCC_PIN; num++) {
    pinMode(num, OUTPUT);
```

```
  }
}

void loop() {
  if ((DELAY <= 0) || (MIN_HUMIDITY <= 0) || (NORMAL_LIGHT_LVL <= 0) || (MAX_TEMP
<= MIN_TEMP)) {
    static unsigned long int last_error_print = 0;
    if ((millis() - last_error_print) > DELAY) {
      Serial.println("Given parameter have an unacceptable value!\n");
      last_error_print = millis();
    }
  } else {
    static unsigned long int last_report_print = 0;
    control_illumination(DELAY);

    control_humidity(DELAY, MIN_HUMIDITY, WATERING_PERIOD, WAITING_PERIOD);

    if (is_watering_need && !is_watering_ended) {
      digitalWrite(FAN_PIN, LOW);
      digitalWrite(HEATER_PIN, LOW);
      if ((millis() - last_report_print) > DELAY) {
        Serial.println("Soil is watering, temperature control is off.\n");
        last_report_print = millis();
      }
    } else {
      control_temperature(DELAY, MIN_TEMP, MAX_TEMP);
    }
  }
}

void control_temperature(const int read_delay, const int min_temp, const int
max_temp) {
  static unsigned long int last_dht_read = 0;
  static float temperature = 0;
  static float air_humid = 0;
  if ((millis() - last_dht_read) > read_delay) {
    if (read_delay < 2000) {
      Serial.println("Delay between measurings of temperature is too short!\n");
    } else {
      dht_sensor.read();
      switch (dht_sensor.getState()) {
        case DHT_OK:
          temperature = dht_sensor.getTemperatureC();
          air_humid = dht_sensor.getHumidity();
          Serial.print("Temperature = ");
          Serial.print(temperature);
          Serial.println(" C \t");
          Serial.print("Air humidity = ");
          Serial.print(air_humid);
          Serial.println("% \t");
          break;
```

```
        case DHT_ERROR_CHECKSUM:
          Serial.println("Checksum error.\n");
          break;
        case DHT_ERROR_TIMEOUT:
          Serial.println("Time out error.\n");
          break;
        case DHT_ERROR_NO_REPLY:
          Serial.println("Sensor not connected.\n");
          break;
      }
      if (dht_sensor.getState() == DHT_OK) {
        if ((temperature <= min_temp) || (temperature >= max_temp)) {
          if (temperature <= min_temp) {
            Serial.println("Temperature is too low. Need to turn on fan and
heater.\n");
          } else {
            Serial.println("Temperature is too high. Need to turn on fan.\n");
          }
        } else {
          Serial.println("Temperature is normal.\n");
        }
        if ((temperature <= min_temp) || (temperature >= max_temp)) {
          digitalWrite(FAN_PIN, HIGH);
          if (temperature <= min_temp) {
            digitalWrite(HEATER_PIN, HIGH);
          } else {
            digitalWrite(HEATER_PIN, LOW);
          }
        } else {
          digitalWrite(FAN_PIN, LOW);
          digitalWrite(HEATER_PIN, LOW);
        }
      } else {
        digitalWrite(FAN_PIN, LOW);
        digitalWrite(HEATER_PIN, LOW);
      }
    }
    last_dht_read = millis();
  }
}

void control_illumination(const int print_delay) {
  static unsigned long int last_light_print = 0;
  bool light_lvl = 0;
  light_lvl = digitalRead(LDR_PIN);
  if ((millis() - last_light_print) > print_delay) {
    if (light_lvl) {
      Serial.println("It is dark. Need to turn on the light!\n");
    } else {
      Serial.println("It is bright. No need to turn on the light.\n");
    }
```

```
    last_light_print = millis();
  }
  digitalWrite(LED_STRIP_PIN, light_lvl);
}

void control_humidity(const int print_delay, const unsigned int min_humidity,
const unsigned long int watering_period, const unsigned long int waiting_period)
{
  static unsigned long int last_humid_print = 0;
  static unsigned long int last_watering = 0;
  static unsigned long int last_waiting = 0;
  unsigned int humid_lvl = 0;
  unsigned int relative_humid_lvl = 0;
  humid_lvl = analogRead(HUMIDITY_SENSOR_PIN);
  relative_humid_lvl = map(humid_lvl, WET_SOIL_HUMID, DRY_SOIL_HUMID, 100, 0);
  if ((!is_watering_ended) && (!is_watering_need)) {
    if ((millis() - last_humid_print) > print_delay) {
      Serial.println("Soil humidity is normal and equal " + String(relative_hu-
mid_lvl) + "%\n");
      last_humid_print = millis();
    }
  }
  if ((!is_watering_need) && (relative_humid_lvl <= min_humidity)) {
    last_humid_print = 0;
    is_watering_need = true;
    last_watering = millis();
    if ((millis() - last_humid_print) > print_delay) {
      Serial.println("Soil humidity is equal " + String(relative_humid_lvl) +
"%");
      Serial.println("Soil is dry. Need to turn on the water pump!\n");
      last_humid_print = millis();
    }
  }
  if (is_watering_need) {
    if ((millis() - last_watering) < watering_period) {
      digitalWrite(WATER_PUMP_PIN, HIGH);
      if ((millis() - last_humid_print) > print_delay) {
        Serial.println("Soil is watering... " + String(round(watering_period -
(millis() - last_watering)) / 1000) + " seconds left before watering ends.\n");
        last_humid_print = millis();
      }
    } else {
      if (!is_watering_ended) {
        last_waiting = millis();
      }
      is_watering_ended = true;
      digitalWrite(WATER_PUMP_PIN, LOW);
      if (dht_sensor.getState() != DHT_OK) {
        Serial.println("Humidity sensor is reconnected.\n");
        digitalWrite(DHT_SENSOR_VCC_PIN, HIGH);
        delay(print_delay);
```

```
        digitalWrite(DHT_SENSOR_VCC_PIN, LOW);
        delay(print_delay);
      }
      if ((millis() - last_humid_print) > print_delay) {
        Serial.println("Soil humidity is equal " + String(relative_humid_lvl) +
"%");
        Serial.println("Watering ended, waiting for spreading water.\n");
        last_humid_print = millis();
      }
    }
  }
  if ((is_watering_ended) && ((millis() - last_waiting) > waiting_period)) {
    last_humid_print = 0;
    if (relative_humid_lvl > min_humidity) {
      is_watering_need = false;
      is_watering_ended = false;
      if ((millis() - last_humid_print) > print_delay) {
        Serial.println("Soil humidity is equal " + String(relative_humid_lvl) +
"%");
        Serial.println("Watering ended, water spread successfully.\n");
        last_humid_print = millis();
      }
    } else {
      is_watering_need = true;
      is_watering_ended = false;
      last_watering = millis();
      if ((millis() - last_humid_print) > print_delay) {
        Serial.println("Soil humidity is equal " + String(relative_humid_lvl) +
"%");
        Serial.println("Watering didn't end, water spread unsuccessfully.\n");
        last_humid_print = millis();
      }
    }
  }
}
```

Список литературы

1. Умная теплица OMEGAGROW — https://omegabot.ru/product/28
2. Интернет вещей (Материал из Википедии) — https://ru.wikipedia.org/wiki/Интернет_вещей
3. Умная теплица на базе arduino из подручного материала с регулятором температуры — https://habr.com/ru/post/536666/
4. Набор «Умная теплица ЙоТик М2» — https://mgbot.ru/catalog/obrazovatelnye_nabory_iot/nabor_umnaya_teplitsa_yotik_m2
5. Arduino UNO R3 — Product Reference Manual: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf
6. Arduino IDE 2 Tutorials — https://docs.arduino.cc/software/ide-v2
7. LM2596HV 60V 3A 150kHz Step-Down Voltage Regulator Datasheet — http://amperkot.ru/static/3236/uploads/datasheets/LM2596HV.PDF
8. Даташит для SRD-05VDC-SL-C, реле 1 переключатель 5VDC/10A, 125VAC — https://static.chipdip.ru/lib/642/DOC012642672.pdf
9. Temperature and humidity module AM2301 Product Manual — https://smartep.ru/datasheets/sensors/DHT21%20AM2301.pdf
10. Библиотека для Arduino, позволяющая считывать данные датчиков серии DHT — https://github.com/amperka/TroykaDHT