МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт энергетики и электротехники
(наименование института полностью)

«Промышленная электроника»
(наименование кафедры)

(11.03.04) Электроника и наноэлектроника
(код и наименование направления подготовки, специальности)

Промышленная электроника
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Умная теплица с микроконтрольным управлением

Студент Р. Н. Медведев _____
(И.О. Фамилия) (личная подпись)

Руководитель А. В. Прядилов _____
(И.О. Фамилия) (личная подпись)

Допустить к защите
Заведующий кафедрой _____ _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2019

Аннотация

Объем 69 с., 44 рис., 18 табл., 20 источников, 1 прил.

В данной бакалаврской работе была создана «умная теплица с микроконтольным управлением» с помощью, которой ведется контроль растении в теплице.

Цель работы: создание умной теплицы с микроконтольным управлением, в которую входят следующие функции: контроль влажности почвы, контроль уровня воды в резервуаре, поддерживание температуры в почве, контроль влажности воздуха и температуры в теплице.

Задачами работы являются:

1. Состояние вопроса;
2. Аппаратная часть;
3. Программная часть;
4. Конструктивно-эксперементальный раздел.

Работа состоит из четырех глав, в которых решены упомянутые задачи.

В процессе выполнения работы все комплектующие устройства были помещены на фанеру, это нужно для того что бы можно было продемонстрировать полноценную работоспособность данного устройства, так же в ходе чего были выявлены недостатки и не доброжелательные дефекты, которые могли повлиять на демонстрацию устройства.

Областью применения «умной теплицы с микроконтольным управлением» являются выращивание овощей и рассады.

Использование выбранных модулей для данной работы, позволило существенно снизить цену на устройство. Появилось такая возможность как, замена не исправных модулей на заведомо исправные, на том же месте где установлена данная система.

Оглавление

ВВЕДЕНИЕ

Большинство людей любят вкусно поесть, еду которую мы с вами употребляем должна быть не только вкусной, но и полезной. То, что сегодня мы покупаем в магазинах, большинство случаев редко не содержит разных биодобавок и препаратов. Многим, вероятно, приходилось пробовать на вкус помидоры из магазинов, с помощью которых можно гвозди заколачивать. Если самостоятельно выращивать свои помидоры, в этом случае мы будем уверены, что в этих помидорах отсутствует вредоносные добавки, которые ухудшают истинный вкус помидоров.

Во время развития любого бизнеса, в какой-то момент наступает время, когда необходимо увеличить производительность и уменьшить расходы. Кажется на первый взгляд, что это взаимоисключающие понятия, однако опытные предприниматели знают, что обе проблемы можно решить в комплексе — путем автоматизации ключевых процессов. Особенно этот вопрос актуален для владельцев тепличных хозяйств.

Благодаря популярной концепции «Интернет вещей» обычная теплица может превратиться в высокотехнологичный объект, способный заботиться о сельхоз культурах не хуже любого растениевода. Главное достоинство комплексов автоматизации помещений защищенного грунта — возможность построить весь процесс выращивания на основе точных и актуальных данных. Базовая задача подобных решений — непрерывный мониторинг жизненно важных для растений параметров микроклимата: уровней освещения; температуры и влажности воздуха, почвы; напора прямой воды в трубопроводе из котла и в обратном направлении по всем контурам отопления.

1. Состояние вопроса.

1.1 Формулирование актуальности, цели и задач проекта

В северной части России климат не стабилен, по этой причине системы автоматизированной теплицы могут быть сильно востребованы во всех регионах страны, где располагаются комплексы защищенного грунта, так же и в южных частях России. Хотя чаще всего данная технология актуальна для тех территорий где «неустойчивое земледелие», то есть для средней полосы и северных территорий.

Владея несколькими соток, акров или гектаров земли, большинство людей занимаются выращиванием разновидных растений, овощей, фруктов и т.д. И не только для себя, но и конечно на продажу. Такой вывод подходит особенно для промышленных теплиц. Чем больше гектаров, тем больше теплиц на ней можно установить и тем больше прибыли и конечно развитие бизнеса.

Тепличное хозяйство – это трудное занятие. Например, если на даче в парнике работы много, то в промышленных масштабах работа увеличивается в 10 раз. Нанимать работников для того что бы обслужить свое производство нерентабельно. Намного выгодней автоматизировать процессы по максимуму и улучшить то, что возможно.

С помощью данного устройства, на базе микроконтроллера Atmega 328 устройство можно перепрограммировать на разные виды растения, которые в свою очередь очень востребованы в определенной влажности и температуре почвы, важно что бы подаваемая вода, влажность, температура почвы была благоприятной для тех или иных растений, если поливать цветы холодной водой или дать земле примерзнуть, растения ослабевают и начинают сбрасывать завязи и бутоны, есть определенные рекомендации относительно нормы полива, влажности воздуха и температуры почвы, которая диктуется в первую очередь типом растения и его вегетативными фазами.

Цель работы: создание «Умной теплицы с микроконтрольным управлением», в которую входят: датчики влажности почвы, часы реального времени, датчики влажности воздуха и температуры, так же 4 реле для включения: подачи воды в почву, контроля воды в баке, контроль температуры почвы, контроль влажности воздуха и температуры.

Поставлены следующие задачи:

1. Разработка структурной схемы
2. Разработка электрической принципиальной схемы
3. Выбор необходимых компонентов
4. Разработка алгоритма работы
5. Разработка программной части устройства
6. Изготовление макета
7. Отладка макета и экспериментальные исследования

1.2 Анализ исходных данных и известных решений

Исходные данные

В исходных данных мы будем использовать блок питания для преобразования с 220 Вольт переменного напряжения в постоянные 9 Вольт. Такое решение было выбрано именно по той причине, чтобы можно было данное устройство запустить в любых условиях для демонстрации.

«Умная теплица с микроконтрольным управлением» будет выполнять следующие функции:

1. Подача воды в резервуар воды по мере осушения резервуара;
2. Подача воды в почву по мере его высыхания;
3. Возможность установки времени полива 2 раза в день;
4. Настройка температуры почвы;
5. Настройка влажности воздуха и температуры.

Известные решения

В продаже имеются большое количество устройств, которые помогают контролировать рост растений, но, как правило, такие устройства, не оправдывают свою цену, и в случае поломки ремонт на месте не возможен, устройство, вышедшее из строя, придется нести специалисту, обычно цена на ремонт таких устройств высокая. На такие устройства обычно уходит большое количество времени для того что бы настроить устройство, и зачастую такими устройствами могут совладать только опытные люди в этой сфере. На данный момент Arduino это самое доступное решение из-за своей дешевизны, и доступности запасных деталей. С помощью Arduino можно решить практически любую поставленную задачу, за все время существований Arduino скопилось огромная база знаний, благодаря всей информаций, которая есть в интернете для Arduino любой желающий и немного понимающий в электронике и программирований сможет разобраться.

На сегодняшний день в продаже можно встретить уже готовые решения для контроля растений в теплице, например такой как «КАТ-04».

Контроллер снимает показания значений температуры и влажности. Присутствует датчик в длину 3 метра. В устройство установлено 4 независимых канала для выполняемых устройств, и пяти канальный реле способное коммутировать нагрузку 220 вольт. К каждому из 4-х каналов есть возможность подключения и настройки следующих устройств: приводы – радиальный и линейный, клапаны полива - импульсный, постоянный и шаровой с моторным редуктором. Управление проветриванием или поливом может осуществляться либо по температуре, либо по влажности. На рисунке 1 изображен пример блока управления «КАТ-04».

Рисунок 1 – Пример блока управления «КАТ-04».

Дисплей блока отображает настоящее время, присутствует таймер полива на все дни недели, для контроля компрессором или электромагнитным клапаном полива овощей и т.д., Присутствует возможность установить любой пробел полива по дням недели. Есть возможность слежения по току за приводами, т.е. у двери оказался предмет, который указывает сопротивление открытию или закрытию и если ток будет завышен, то она остановиться. У

контроллера модели КАТ-04 есть пульт по радиоканалу. Благодаря пульту есть возможность управлять 4-мя каналами. Далеко находясь от теплицы можно включать и отключать полив. На рисунке 2 изображен вид пульта.

Контроллер может работать в 2-х режимах:

- кнопки радиопульта дублируют кнопки самого блока управления, тогда задействовано 3 кнопки.

- кнопки радиопульта управляют каждым каналом, 4 кнопки, на каждый канал назначена своя кнопка.

Рисунок 2 - пульт дистанционного управления КАТ-04

Плюсами блока управления КАТ-04 являются: невысокая цена, простота в использовании, пульт дистанционного управления. Минусом является: непригоден к ремонту на месте, отсутствие возможности замены неисправного устройства в кротчайшие сроки, нет возможности автоматического контроля влажности почвы.

Контроллер теплицы - это главный элемент автоматизации, именно данное электронное устройство под названием «УМНИЦА» производит сбор данных, последующий анализ значений, запись на SD карту, СМС информирование, корректировку и управление исполнительными устройствами. На рисунке 3 изображен пример блока управления УМНИЦА.

Рисунок 3 – Пример блока управления УМНИЦА.

Автоматизация теплицы “УМНИЦА” реализует следующие возможности:

1. Измерение температуры и влажности воздуха внутри, снаружи;
2. Измерение температуры и влажности почвы;
3. Контроль освещенности;
4. Выбор значений температуры воздуха и почвы, для дневного и ночного времени суток;
5. Измерение температуры и влажности почвы в 2 контрольных точках теплицы;
6. Включение системы капельного полива по недостаточному значению влажности;
7. Контроль уровня воды в накопительном баке;
8. Регулирование положения конструктивных элементов проветривания (дверей, форточек, фрамуг и т.п.);
9. Регистрация системных сообщений и ошибок на карте памяти;
10. Гибкость настроек;
11. Отправка СМС сообщения с измеренными значениями датчиков по телефонному звонку;

12. Передача СМС сообщения информации о появлении нарушений в работе;
13. Подключение к сети Wi-Fi роутера или создание режима раздачи точки доступа для мониторинга через мобильные устройства и ПК;

Плюсами блока управления УМНИЦА являются: много функциональность, передача информаций на любой точке мира, запись данных на микро флешку за последнюю неделю. Минусом является: Высокая цена, затраты на оборудование, отсутствует такая возможность как замена блока управления при поломке на новый (только покупка нового).

Подобрать более функциональный блок управления для теплиц за не большую цену можно, например такой как «Hunter ELC-401i-E». На рисунке 4 изображен пример блока управления Hunter ELC-401i-E.

Плюсы данного вида: невысокая цена, высокая надежность, простое программное меню, понятный кнопочный интерфейс. Минусы данного вида: необходимо питание от батарейки крона, большие затраты на батарейки, нет возможности замены неисправных модулей.

Рисунок 4 - Пример блока управления Hunter ELC-401i-E.

Функции Hunter ELC-401i-E:

1. управляемые зоны: 4;
2. количество независимых программ: 2;

3. количество запусков на каждой программе: 4;
4. максимальное время работы зоны на одном запуске: 4 часа.
5. байпас датчика дождя;
6. программируемая задержка сессий полива от одного до семи дней;
7. функция быстрой проверки Quick Check™;
8. автоматическая защита от короткого замыкания;

Так же можно рассмотреть такой вариант как «ДЖИН» в его функциональность входят четыре устройства такие как: компрессоры, нагреватели помещений, кондиционеры, вентиляций, специальное освещение, и другие. На экране, встроенном в пульт, так же есть возможность выставлять текущую температуру, с точностью до сотых градуса с 2-х датчиков температуры, примерно от минус 50 до +125 ºС. На рисунке 5 изображен пример блока управления «ДЖИН».

Рисунок 5 - Пример блока управления ДЖИН.

Настраиваемые параметры:

1. Установка часов
2. Температура вкл.- выкл. нагревателя
3. Длительность светового дня (ночи)

4. Время полива (аэратции) сек.
5. Время паузы между поливами (аэратции) сек.
6. Температура вкл. – выкл. вентилятора

Плюсами блока управления ДЖИН являются: Простота установки, влагозащитный корпус, гибкость системы. Минусом является: Высокая цена, отсутствие обслуживания на месте.

В иллюстрационном материале в таком как «Обзорный лист» были приведены основные варианты обзора существующих решений, в которых выделены достоинства и недостатки. На рисунке 6 показан обзорный лист.

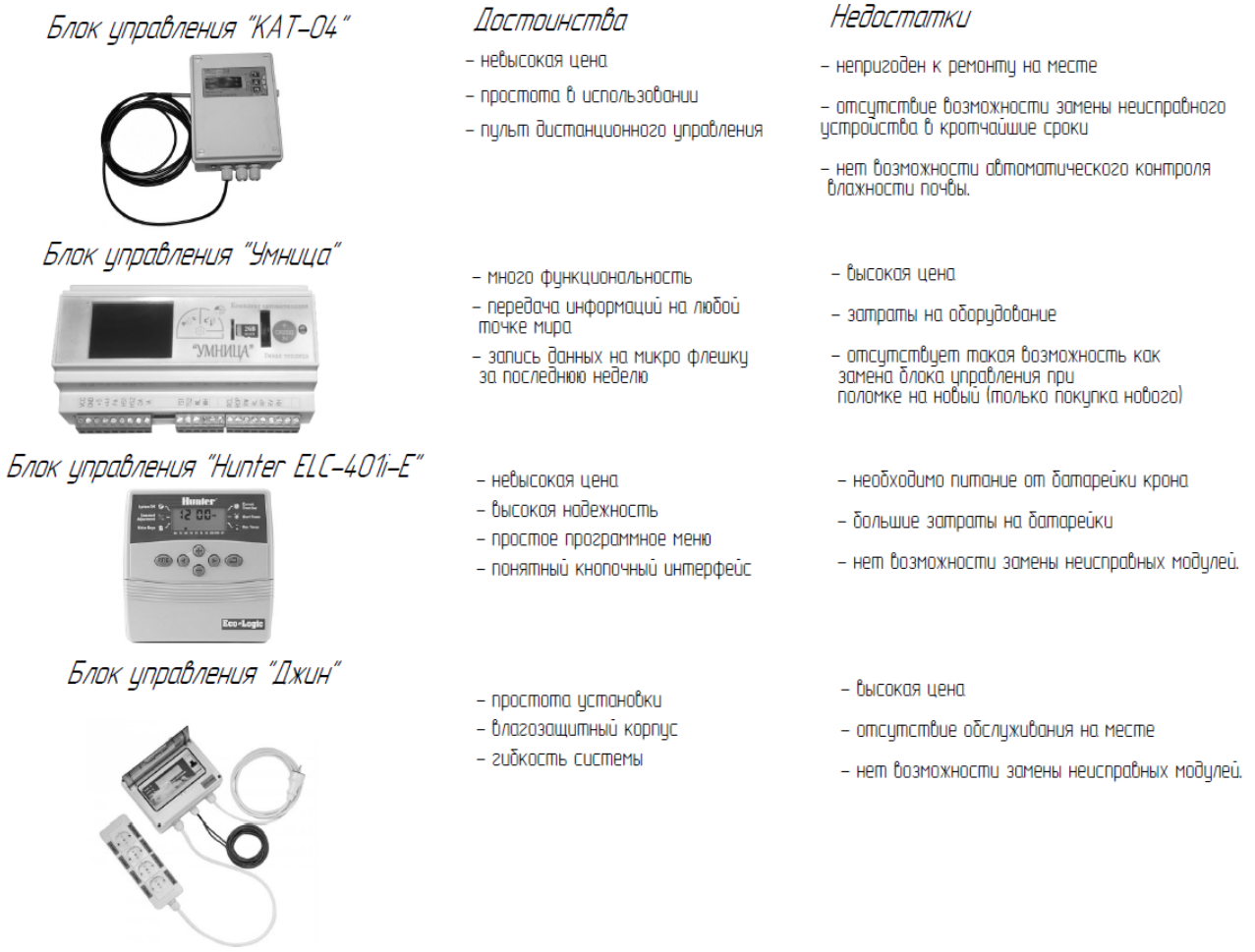

Рисунок 6 - Обзорный лист.

2. Аппаратная часть

2.1 Разработка структурной схемы

Для управления «Умной теплицей» было принято решение подобрать Arduino Uno, по той причине, что данный микроконтроллер подходит для управления всех комплектующих устройств и в плане дешевизны Arduino Uno.

Структурная схема была разработана в пакете программ DipTrace в программе Schematic.

В разработанной структурной схеме входят несколько функциональных узлов:

- Микроконтроллер, является сердцем системой управления.

-Датчики контроля, предназначены для контроля температуры внешней среды, влажность воздуха, температуру почвы, температуру воды в баке, уровень воды, а так же влажность почвы.

- Блоки питания, с помощью них мы преобразуем 220 В переменного напряжения, в постоянные 9 В, которое нам нужно для питания микроконтроллера, датчиков и реле.

- Понижающий преобразователь напряжения, с помощью этого модуля мы понижаем постоянные 9 В, в постоянные 5 В, для того что бы мы могли продемонстрировать работоспособность полива, нагрева почвы, контроль уровня воды, контроль влажности воздуха и температуры.

- Блок реле, предназначен для включения вентилятора, двух компрессоров, подогрева почвы, эти устройства включаются после того как датчики проинформируют систему о достигшем пороге значений.

- Дисплей, предназначен для вывода информаций, настройки системы полива два раза в день, выставлений порога значений для включения вентилятора, двух компрессоров, и подогрева почвы.

На рисунке 7 показана структурная схема «Умной теплицы с микроконтрольным управлением».

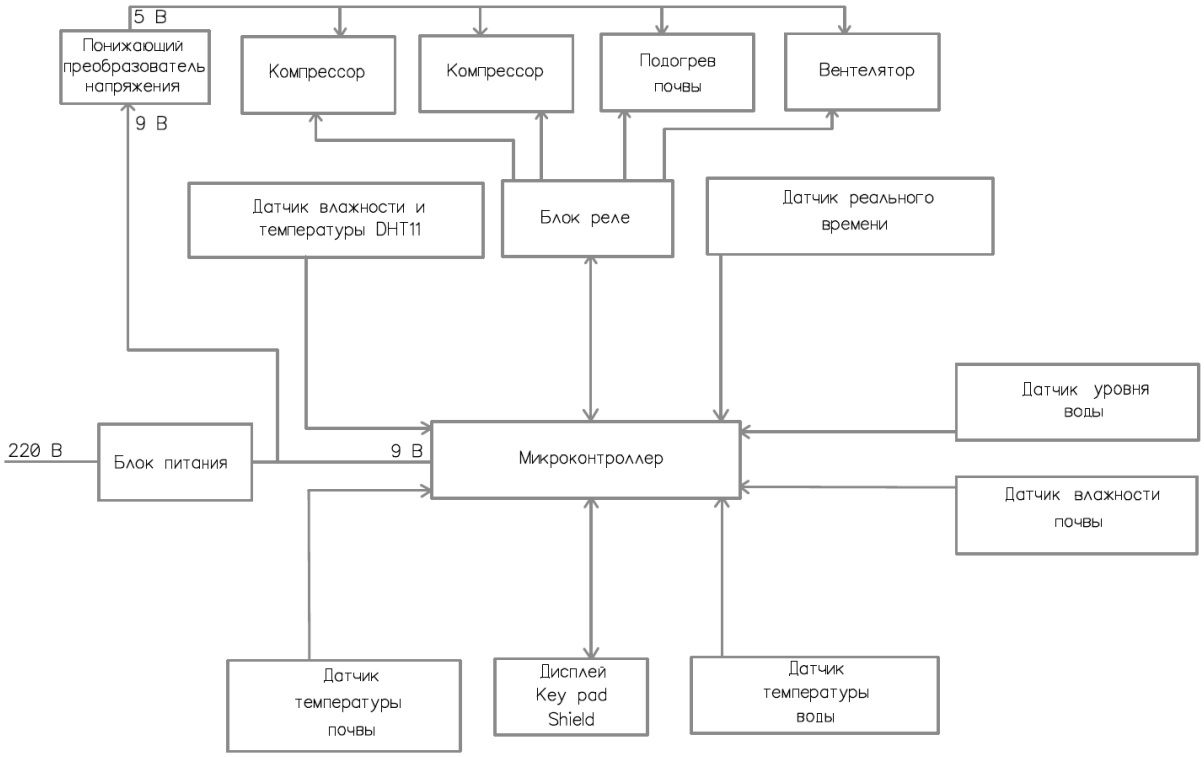

Рисунок 7 – Структурная схема «Умной теплицы с микроконтрольным управлением».

2.2 Разработка электрической принципиальной схемы

Благодаря принципиальной схеме мы можем увидеть устройство в подробностях, весь их состав компонентов и взаимосвязи между каждым элементом.

Принципиальная схема помогает решать проблему при не правильной работе системы, а также наладки и контроля. Схема помогает разобраться в принципе работы устройства. Схема электрическая принципиальная, так же, как и структурная схема, была выполнена в пакете программ DipTrace в программе Schematic.

На рисунке 8 показана принципиальная схема «Умной теплицы с микроконтрольным управлением».

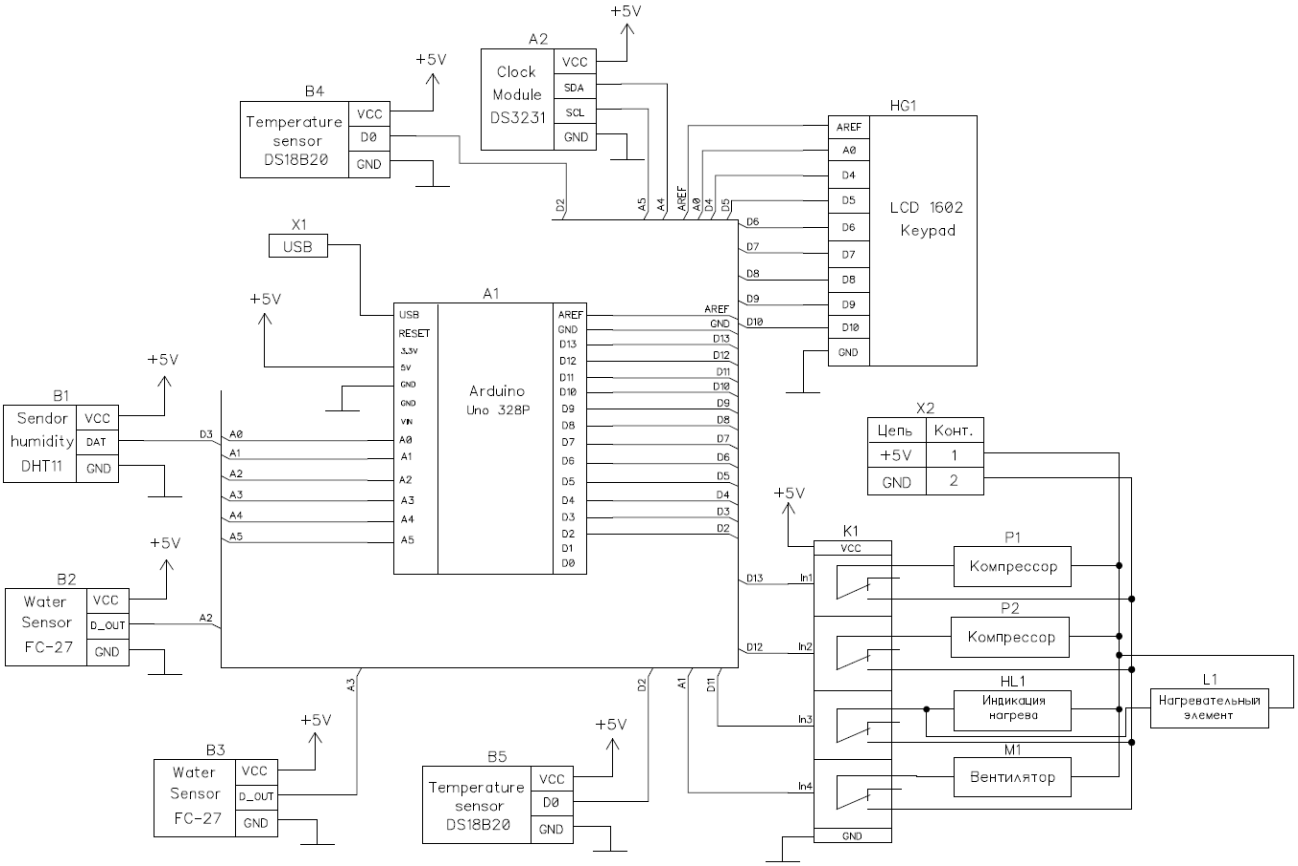

Рисунок 8 – Принципиальная схема «Умной теплицы с микроконтрольным управлением».

2.3 Выбор необходимых компонентов

Разрабатываемое устройство «Умная теплица с микроконтрольным управлением» будет иметь в составе своих компонентов: микроконтроллер, которая является системой управления, устройства для исполнения механизмов, дисплей для вывода информаций, датчики для контроля определенных значений такие как: температуры воздуха, влажности воздуха, температуры почвы, влажности почвы, уровень воды в баке.

Системой управления «Умной теплицы с микроконтрольным управлением» была выбрана Arduino Uno (рисунок 9). Контроллер Uno является самым подходящим вариантом для данного проекта. Она имеет удобный размер (не слишком большой, как у Mega и не такой маленький, как у Nano), достаточно доступна из-за массового выпуска всевозможных клонов, под нее написано огромное количество бесплатных уроков и скетчей.

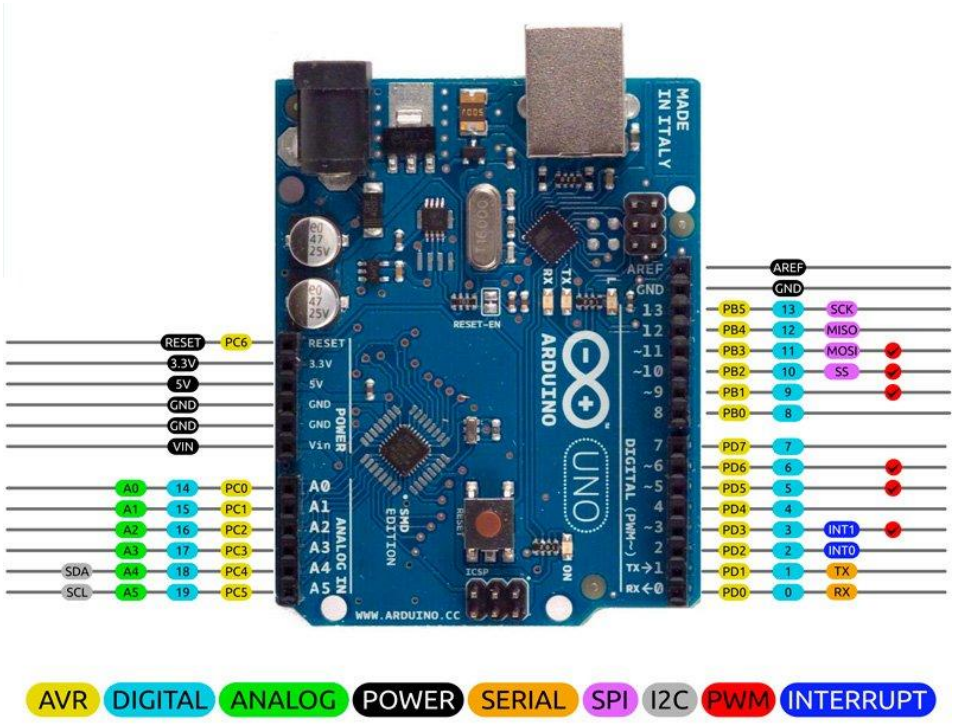

Рисунок 9 – Arduino Uno.

В Arduino Uno встроен контроллер ATmega328**:** В этом устройстве присутствует шесть аналоговых (Analog) входов, кварц на 16 МГц, четырнадцать цифровых входов и выходов (шесть из них могут применяться как выходы ШИМ), коннектор USB, силовой коннектор, коннектор ICSP и кнопку перезагрузки.

Контакты Arduino используются для подключения внешних устройств и могут работать как в режиме входа (INPUT), так и в режиме выхода (OUTPUT).

Контакты с номерами от 0 до 13 являются цифровыми. На данных пинах можно считать и подавать на них только два вида сигналов: HIGH и LOW. Также с помощью ШИМ можно использовать цифровые порты для управления мощностью подключенных устройств.

Аналоговые контакты Arduino Uno предназначены для подключения аналоговых устройств и являются входами для встроенного аналого-цифрового преобразователя (АЦП), который в Arduino Uno десятиразрядный.

В последовательной шине - 0 (RX) и 1 (TX), эти выходы применяются для принятия (RX) и выдачи (TX) информационных данных «TTL». Эти выводы соединены к соответствующим пинам микросхемы последовательной шины ATmega8U2 «USB-to-TTL».

Во внешнем прерываний: втором и третьем, эти выходы можно сконфигурировать на призыв остановки или же на низшем значении, или на начальном или последнем фронте, либо при смене значения.

Пины которые входят в «ШИМ» (3, 5, 6, 9, 10, и 11). Каждый из этих пинов позволяет обеспечить «широтно-импульсную модуляцию». «ШИМ» - это способ управления подачей мощности к нагрузке.

Пины которые входят в «SPI»: 10 - (SS), 11 - (MOSI), 12 - (MISO), 13 - (SCK). Благодаря этим пинам данные выводы осуществляется связь «SPI».

На пине 13 размещен светодиод «LED». Встроенный светодиод, подключенный к цифровому (Digital) выводу 13. Если на выходе 13 присутствует высокий потенциал, тогда светодиод будет гореть.

Пины питания:

- Вывод 5V – на этот пин ардуино подает 5 В, его можно использовать для питания внешних устройств.

- Вывод 3.3V – на этот пин от внутреннего стабилизатора подается напряжение 3.3 В

- GND – вывод земли.

- VIN – пин для подачи внешнего напряжения.

- AREF – пин для информирования внешних устройств о рабочем напряжении платы.

Рабочее напряжение платы Arduino Uno – 5 В. На плате установлен стабилизатор напряжения, поэтому на вход можно подавать питание с разных источников. Кроме этого, плату можно запитывать с USB – устройств.

Таблица 1 - Технические характеристики Arduino Uno:

Рабочее напряжение	5 В
Входное напряжение (рекомендуемое)	7-12 В
Входное напряжение (предельное)	6-20 В
Цифровые Входы/Выходы	14 (6 из которых ШИМ)
Аналоговые входы	6
Постоянные ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
EEPROM	1 Кб
Тактовая частота	16 МГц
Флеш – память	32 Кб, при этом 0.5 Кб используются для загрузчика.

Дополнительные контакты на плате:

AREF – выдает опорное напряжения для встроенного Аналого-цифровой преобразователь (АЦП).

RESET – подача низкого сигнала на этом входе приведет к перезагрузке устройства.

В настоящее время выпускается большое количество различных датчиков, реле, серво приводы и т.д. За все время существования Arduino,

скопилось большое количество библиотек, эти библиотеки делают программирование более комфортным и упрощенным.

Для реализаций «Умной теплицы с микроконтрольным управлением» были подобраны следующие модули, а так же блок реле.

Модуль часов реального времени (DS3231) — это устройство, который предназначен для хранения хронометрии, то есть - (текущее время, дата, день недели и др.) (рисунок 10), из себя он представляет устройство из автономного источника питания и учитывающего устройства.

Модуль DS3231, по сути, представляет из себя обыкновенные часы. В платах Arduino уже есть встроенный датчик времени Millis, однако он работает только при поданном питании на плату. При отключении и дальнейшем включении Arduino отсчет времени Millis сбросится до нуля. А DS3231 имеет на борту батарейку, которая даже при отключенной плате Arduino продолжает «питать» модуль, позволяя ему измерять время.

Модуль можно использовать в качестве часов или будильника, построенных на базе плат Arduino. Или же в качестве оповещения для различных систем.

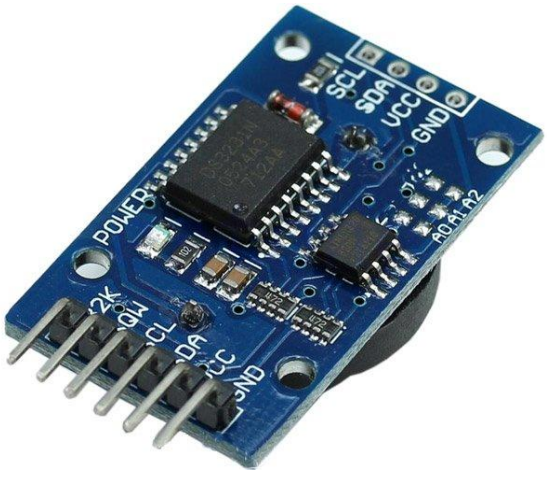

Рисунок 10 – Модуль часов реального времени «DS3231».

«DS3231» - этот модуль Real Time Clock (RTC) высокоточен, в него встроен I2C интерфейс, термокомпенсированным кварцевым генератором

(TCXO) и резонатором. В приборе присутствует посадочное место для подключения дополнительного автономного источника питания, который позволяет осуществить продолжительность работы системы при отсутствий основного питания, в ходе чего выставленные параметры будут в сохранном виде.

Присутствующий на плате кварцевый резонатор позволяет повысить продолжительность службы модуля и благодаря чему уменьшило количество внешних установленных элементов на плате устройства.

«RTC» позволяет производить отсчет: секунды, минуты, часов, недели, месяцы и года. «RTC» работает в 24 или 12 часовом, «AM/PM». У модуля присутствует возможность выставления 2 ежедневных будильника и возможность программирования выходного прямоугольного сигнала. С помощью встроенного I2C производится обмен данных.

Что бы контролировать напряжение в цепи, для сохранения ранее имеющихся данных, в модуле присутствует высоко точный источник опорного напряжения (ИОН), схема сравнивает и проверяет напряжение основного питания VCC и если при понижений выставленного порога, схема формирует импульс сброса и осуществляет перевод схемы на работу от дополнительного источника питания (ИП). На плате так же присутствует контакт RST (Reset) с помощью данного контакта можно воспользоваться как для внешнего сброса.

Таблица 2 - Технические характеристики модуля DS3231:

Напряжение питания	3.3В и 5В
Чип памяти	AT24C32 (32 Кб)
Точность	± 0.432 сек в день
Частота кварца	32.768 кГц
Поддерживаемый протокол	I2C

Таблица 3 - Назначение выводов модуля DS3231:

32K	выход, частота 32 кГц
SQW	выход
SCL	линия тактирования (Serial CLock)
SDA	линия данных (Serial Dфta)
VCC	«+» питание модуля
GND	«-» питание модуля

Подключение датчика показано в таблице 4.

Таблица 4 – Подключение датчика реального времени к Arduino Uno

DS3231	Arduino Uno
SCL	A5
SDA	A4
VCC	5 В
GND	GND

Схема подключения DS3231 показана на рисунке 11.

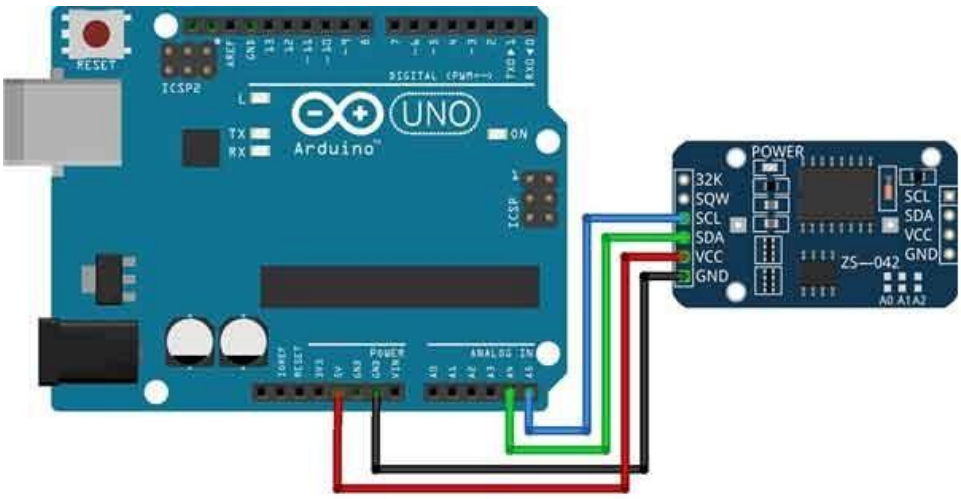

Рисунок 11 - Схема подключения DS3231 к Arduino Uno.

Датчик температуры (DS18B20) – это цифровой температурный датчик, обладающий множеством полезных функций. По сути, DS18B20 – это целый микроконтроллер, который может хранить значение измерений, сигнализировать о выходе температуры за установленные границы (сами границы мы можем устанавливать и менять), менять точность измерений, способ взаимодействия с контроллером и многое другое. Все это в очень небольшом корпусе, который, к тому же, доступен в водонепроницаемом исполнении. Датчик температуры (DS18B20) показан на рисунке 12.

Микросхема имеет три выхода, из которых для данных используется только один, два остальных – это земля и питание. Число проводов можно сократить до двух, если использовать схему с паразитным питанием и соединить Vdd с землей. К одному проводу с данными можно подключить сразу несколько датчиков DS18B20 и в плате Arduino будет задействован всего один пин.

Рисунок 12 - Датчик температуры DS18B20.

Основной задачей DS18B20 является определение температуры и преобразование полученного результата в цифровой вид.

Во время включения питания датчик находится в состоянии покоя. Для начала измерения контроллер Arduino выполняет команду «преобразование

температуры». Полученный результат сохранится в 2 байтах регистра температуры, после чего датчик вернется в первоначальное состояние покоя.

Полученные температурные измерения сохраняются в SRAM датчика.

Таблица 5 - Технические характеристики Датчика температуры DS18B20:

Диапазон измеряемых температур	−55…+125 °C
Точность	±0,5°C (в пределах −10…+85 °C)
Время получения данных	750 мс при 12-битном разрешении; 94 мс при 9-битном разрешении
Напряжение питания	3–5,5 В
Потребляемый ток при бездействии	750 нА
Потребляемый ток при опросе	1 мА

Таблица 6 -Назначение выводов Датчика температуры DS18B20:

VCC	питание
GND	земля
DATA	вывод данных

Подключение датчика показано в таблице 7.

Таблица 7 – Подключение датчика температуры DS18B20 к Arduino Uno.

DS18B20	Arduino Uno
VDD	5 В
GND	GND
DATA	D2

Схема подключения датчика температуры DS18B20 показана на рисунке 13.

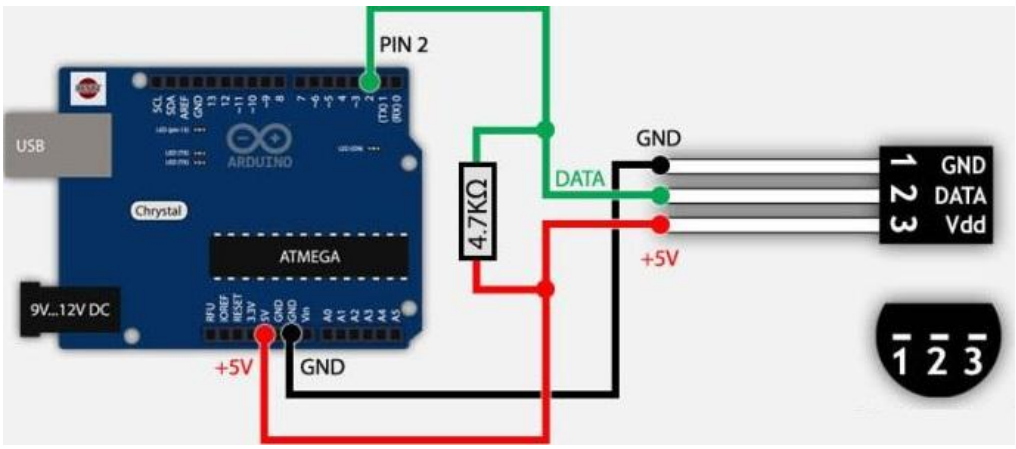

Рисунок 13 - Подключение датчика температуры DS18B20

В данном дипломном проекте подключено несколько датчиков DS18B20 параллельно. Используемая нами библиотека «OneWire library» дает нам возможность снятия данных со всех установленных датчиков сразу. На рисунке 14 показано параллельное подключение датчиков температуры DS18B20.

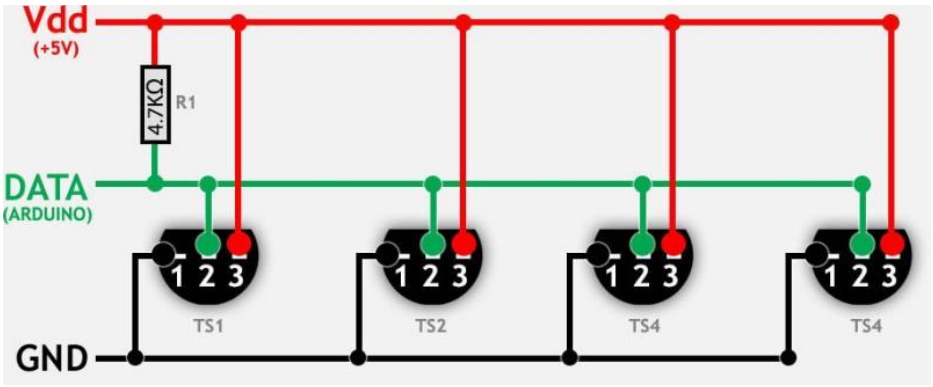

Рисунок 14 - Параллельное подключение датчиков температуры DS18B20.

Если количество подключаемых датчиков более 10, нужно подобрать резистор с сопротивлением не более 1,6 кОм. Также для более точного измерения температуры нужно поставить дополнительный резистор на 100…120 Ом между выходом data на плате Ардуино и датчике.

Мы используем цифровые датчики, соответственно, у всех датчиков имеется собственный серийный номер, с помощью этого серийного номера мы

сможем определить, какой из серийных номеров относится к тому или иному датчику. Для определения номера датчика необходимо загрузить библиотеку «OneWire», далее для удобства с работой с датчиком нужно использовать библиотеку «DallasTemperature», особенно в нашем случае необходимо использовать данную библиотеку, после чего открываем в программу Arduino Software, переходим во вкладку Файл – Параметры – выбираем нам нужную библиотеку.

Далее в 10 строке указываем пин к которому подключен датчик «OneWire ds(10); // on pin 10 (a 4.7K resistor is necessary)», изначально там указан 10 пин, а в нашем случаи это D2, то есть, меняем «OneWire ds(10);» на «OneWire ds(D2);» и загружаем в arduino. Открываем «монитор порта» в мониторе каждую секунду будет показывать информацию о датчиках, которые подключены к Ардуино. В итоге мы получили следующие серийные номера с датчиков: DeviceAddress sensor1 = {0x28, 0x92, 0x03, 0x77, 0x91, 0x06, 0x02, 0x34};, DeviceAddress sensor2 = {0x28, 0xBB, 0x5A, 0x77, 0x91, 0x07, 0x02, 0xB6};.

Датчик DHT11 — этот датчик позволяет снять показания влажности и температуры воздуха, датчик изготовлен из термистора и емкостного датчика влажности. В датчик так же встроен АЦП (Аналого-цифровой преобразователь) с помощью которого идет преобразование аналоговых значений влажности и температуры воздуха. У модуля DHT11 присутствует такой не достаток как не высокоточные снятия показаний и так же не обладает быстродействием, данный датчик был подобран для демонстраций стенда, данный датчик идеально подходит для обучения и поддерживания влажности воздуха в помещении, к тому же этот датчик очень дешевый. Датчик DHT11 показан на рисунке 15.

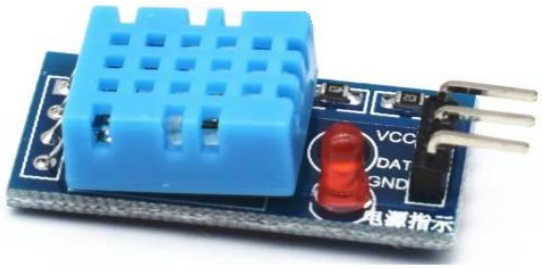

Рисунок 15 - Датчик DHT11.

Таблица 8 - Технические характеристики датчика DHT11:

Напряжение питания	5В
Диапазон температур	0–50 °C
Погрешность температуры	±2 °C
Диапазон влажности	20–90%
Погрешность влажности	±5%

Таблица 9 - Назначение выводов датчика DHT11:

VCC	питание датчика
DAT	вывод данных
GND	земля

Подключение датчика показано в таблице 10.

Таблица 10 – Подключение Датчик DHT11 к Arduino Uno.

DHT11	Arduino Uno
VCC	5 В
DAT	D3
GND	GND

Схема подключения датчика DHT11 показана на рисунке 16.

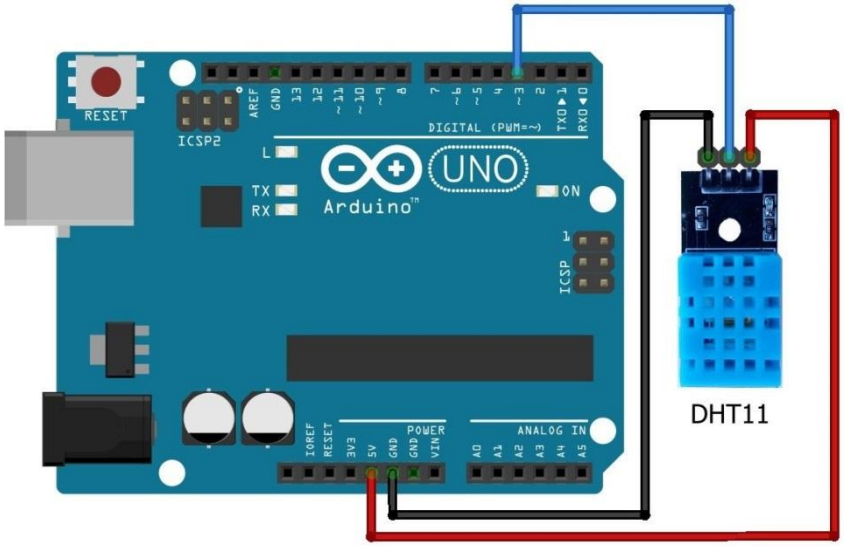

Рисунок 16 – Подключение DHT11 к Arduino Uno.

Датчик влажности почвы FC-27 - этот датчик измеряет объемное содержание воды в почве и выдает нам уровень влаги. Датчик выдает нам на выходе аналоговые и цифровые данные.

Датчик влажности почвы состоит из двух датчиков, которые используются для измерения объемного содержания воды. Два зонда позволяют току пройти через почву, которая дает значение сопротивления, что позволяет в итоге измерить значение влаги. Датчик влажности почвы FC-27 показан на рисунке 17.

Когда есть вода, почва будет проводить больше электричества, а это значит, что будет меньше сопротивление. Сухая почва плохо проводит электричество, поэтому, когда воды меньше, почва проводит меньше электричества, а это значит, что сопротивление будет больше.

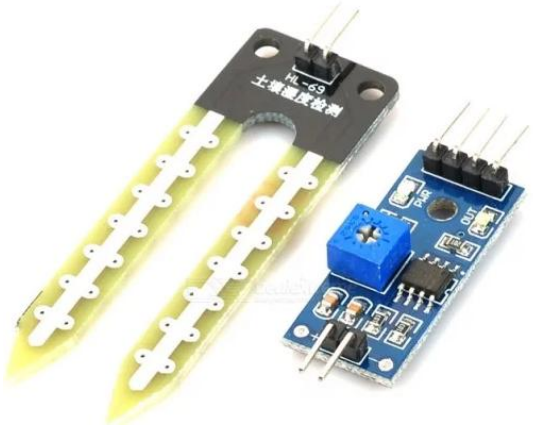

Рисунок 17 - Датчик влажности почвы FC-27.

Модуль также содержит потенциометр, который установит пороговое значение. Это пороговое значение будет сравниваться на компараторе LM393. Светодиод будет нам сигнализировать значение выше или ниже порогового.

Датчик FC-27 можно соединить в аналоговом и цифровом режимах.

Для подключения датчика в аналоговом режиме нам потребуется использовать аналоговый выход датчика. Датчик влажности почвы FC-28 принимает аналоговые выходные значения от 0 до 1023.

Для подключения датчика влажности почвы FC-27 в цифровом режиме мы подключим цифровой выход датчика к цифровому контакту Arduino Uno.

Модуль датчика содержит потенциометр, который использован для того чтобы установить пороговое значение. Пороговое значение после этого сравнивается со значением выхода датчика, используя компаратор LM393, который помещен на модуле датчика FC-27. Компаратор LM393 сравнивает значение выхода датчика и пороговое значение, и после этого дает нам выходное значение через цифровой вывод.

Когда значение датчика больше чем пороговое значение, цифровой выход передаст нам 5В, и загорится светодиод датчика. В противном случае, когда значение датчика будет меньше чем это пороговое значение, на цифровой вывод передастся 0 В и светодиод не загорится.

Таблица 11 - Технические характеристики датчика влажности почвы FC-27:

Напряжение питания Vcc	5 В или 3,3 В
Напряжение на выходе датчика	0 ... 4,5 В
Максимальный потребляемый ток	< 4,5 мА, при Vcc = 5 В и датчик погружён в грунтовую воду.
Глубина погружения в почву	45 мм

Таблица 12 - Назначение контактов датчика влажности почвы FC-27:

VCC	питание датчика
GND	земля
D0	цифровое значение
A0	аналоговое значение

Подключение датчика влажности почвы FC-27 показано в таблице 13.

Таблица 13 – Подключение датчика влажности почвы FC-27 к Arduino Uno.

FC-27	Arduino Uno
VCC	5 В
GND	GND
D0	-
A0	A2, A3

Схема подключения датчика влажности почвы FC-27 показана на рисунке 18.

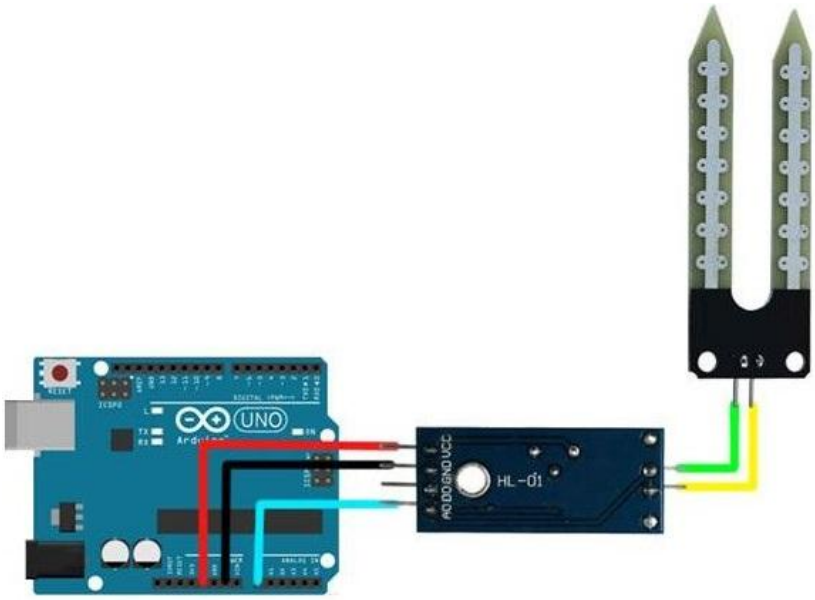

Рисунок 18 - Подключение датчика влажности почвы FC-27.

Плата реле SRD-05VDC-SL-C- это электромеханическое устройство, которое служит для замыкания и размыкания электрической цепи с помощью электромагнита. Принцип работы реле очень прост. При подаче управляющего напряжения на электромагнитную катушку, в ней возникает электромагнитное поле, которое притягивает металлическую лапку, и контакты мощной нагрузки замыкаются.

Согласно характеристикам реле SRD-05VDC-SL-C, для переключения контактов достаточно около 5 Вольт 20 мА, выводы на Arduino способны выдавать до 40 мА. Таким образом, с помощью Arduino мы можем управлять не только лампой накаливания, но и любым бытовым прибором — обогревателем, холодильником и т.д.

Вся электронная обвязка необходимая для управления реле уже встроена в модуль. На модуле расположен светодиод, который всегда подскажет — замкнуто реле или нет.

Реле срабатывает, когда на вход модуля подаётся логическая единица. При этом управление производится через встроенный ключ, что даёт возможность использовать в качестве управляющего сигнала любое

напряжение от 3 до 5 вольт. Поэтому модулем можно свободно управлять с большинства плат.

Плата реле SRD-05VDC-SL-C показана на рисунке 19.

Рисунок 19 - Плата реле SRD-05VDC-SL-C.

Таблица 14 - Назначение контактов платы реле SRD-05VDC-SL-C:

GND	земля
IN1	вывод выходного сигнала
IN2	вывод выходного сигнала
IN3	вывод выходного сигнала
IN4	вывод выходного сигнала
VCC	питание

Таблица 15 - Технические характеристики платы реле SRD-05VDC-SL-C:

Номинальное напряжение питания	5 В
Номинальное напряжение сигнала	3–5 В
Максимальный ток коммутации	16 А
Коммутируемое переменное напряжение (пиковое)	250 В
Потребляемый ток	87 мА
Рабочая температура	−40…+85 °C

Подключение платы реле SRD-05VDC-SL-C показано в таблице 16.

Таблица 16 – Подключение платы реле SRD-05VDC-SL-C.

SRD-05VDC-SL-C	Arduino Uno
GND	GND
IN1	D13
IN2	D12
IN3	D11
IN4	A1
VCC	5 В

Схема подключения платы реле SRD-05VDC-SL-C показана на рисунке 20.

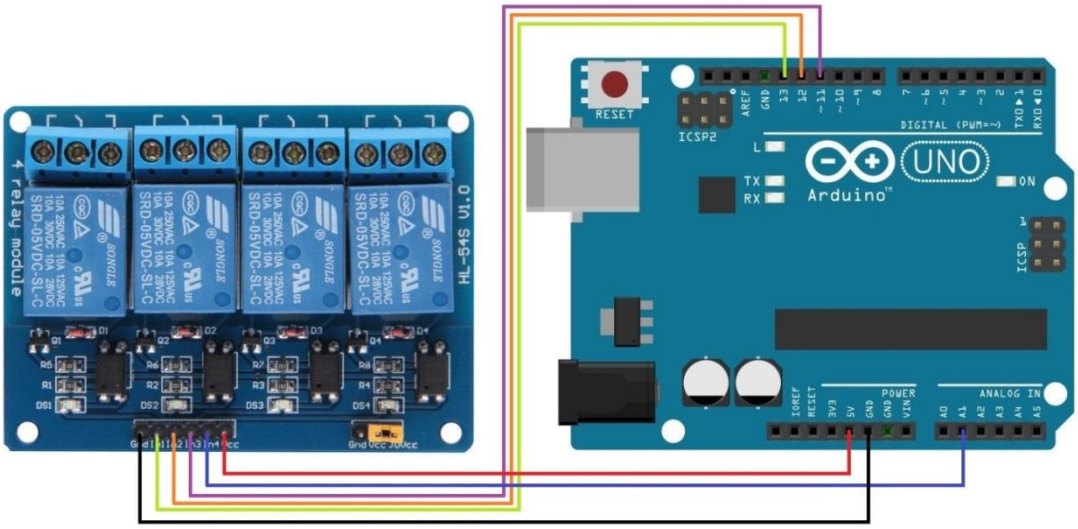

Рисунок 20 - Схема подключения платы реле SRD-05VDC-SL-C.

«LCD Keypad shield» - это модуль расширения для Arduino Uno, и сопоставимых по распиновке. В состав платы входит LCD дисплей размерности 16 х 2 символа и 5 активных кнопок и одна кнопка перезагрузки контроллера. Для настройки контрастности экрана добавлен подстроечный резистор на 10 кОм. Светодиод символизирует подачу питания на шилд. На рисунке 22 показано LCD Keypad shield.

Подключение кнопок производится по схеме из даташита на шилд. Все 5 кнопок подсоединены к аналоговому входу A0 по схеме, приведенной на рисунке 21.

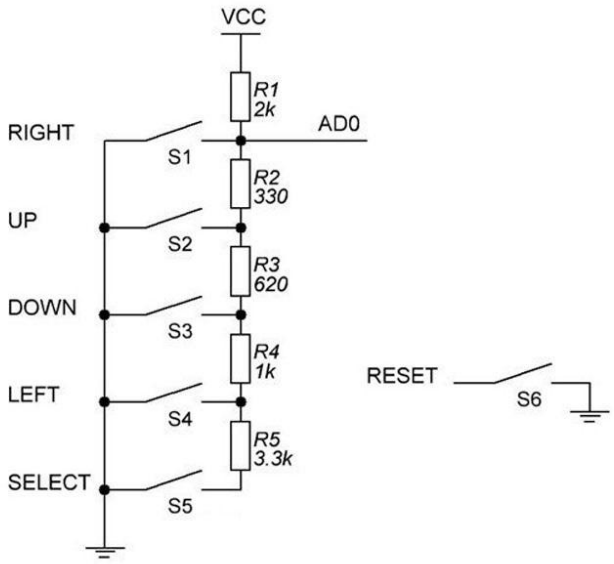

Рисунок 21 – Схема подключения кнопок LCD Keypad shield.

В плату входит LCD 1602 дисплей синего цвета с белыми буквами и 6 кнопок. Цифровые пины 4, 5, 6, 7, 8, 9 задействованы для управления LCD, цифровой пин 10 управляет яркостью подсветки дисплея. На аналоговый пин 0 считываются сигналы с кнопок. Дисплей в данной сборке работает в четырех битном режиме.

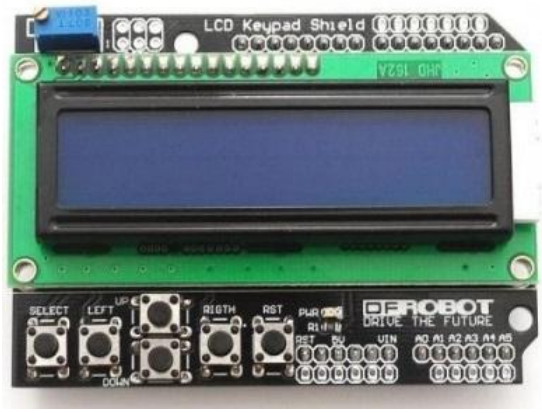

Рисунок 22 - LCD Keypad shield.

Подключение LCD Keypad Shield производится стандартным способом. Необходимо просто состыковать шилд с платой из семейства Arduino. На рисунке 23 и 24 показан пример подключения.

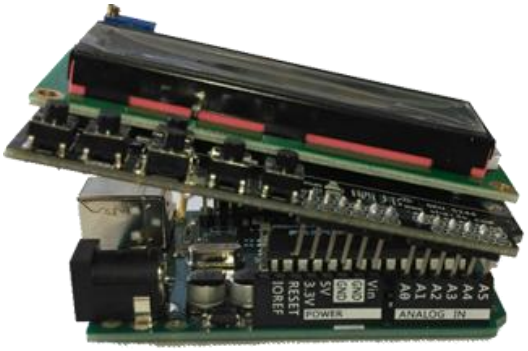

Рисунок 23 – Подключение LCD Keypad shield к ArduinoUno.

Рисунок 24 – Подключение LCD Keypad shield к Arduino Uno.

Таблица 17 - Технические характеристики- LCD Keypad shield:

Работа дисплея	в 4 битном режиме
Количество кнопок	5 активных кнопок и 1 кнопка перезагрузки контроллера
Максимально разрешение экрана	16x2
Питание шилда	5 Вольт
Частота обновления экрана	до 5 Гц

Подключение платы LCD Keypad shield показано в таблице 18.

Таблица 18 – Подключение платы LCD Keypad shield.

LCD Keypad shield	Arduino Uno
AREF	AREF
A0	A0
D4	D4
D5	D5
D6	D6
D7	D7
D8	D8
D9	D9
D10	D10
GND	GND

3. Программная часть.

3.1 Разработка алгоритма работы.

Алгоритм – это комплект блоков, показывающих порядок действий устройства для получения результата выполнения задачи за конченое число действий. На рисунке 25 изображен разработанный алгоритм работы системы «Умная теплица с микроконтрольным управлением».

В разрабатываемом устройстве «умной теплицы с микроконтрольным управлением», будут осуществляться следующие функции: Подача воды в резервуар воды по мере осушения резервуара, подача воды в почву по мере его высыхания, возможность установки времени полива два раза в день или один раз в день, установка нужной температуры почвы, установка влажности воздуха и температуры воздуха.

Arduino Uno и модули будут запитаны от блока питания. Блок питание преобразует с 220 В переменного напряжения в постоянные 9 В, или же данные устройства можно запитать от аккумуляторов с повышающим преобразователем или же соединив последовательно несколько аккумуляторов 18650. Демонстрационные устройства будут запитаны от этого же блока питания на 9 В, но используя понижающий преобразователь, который будет понижать до 5 В.

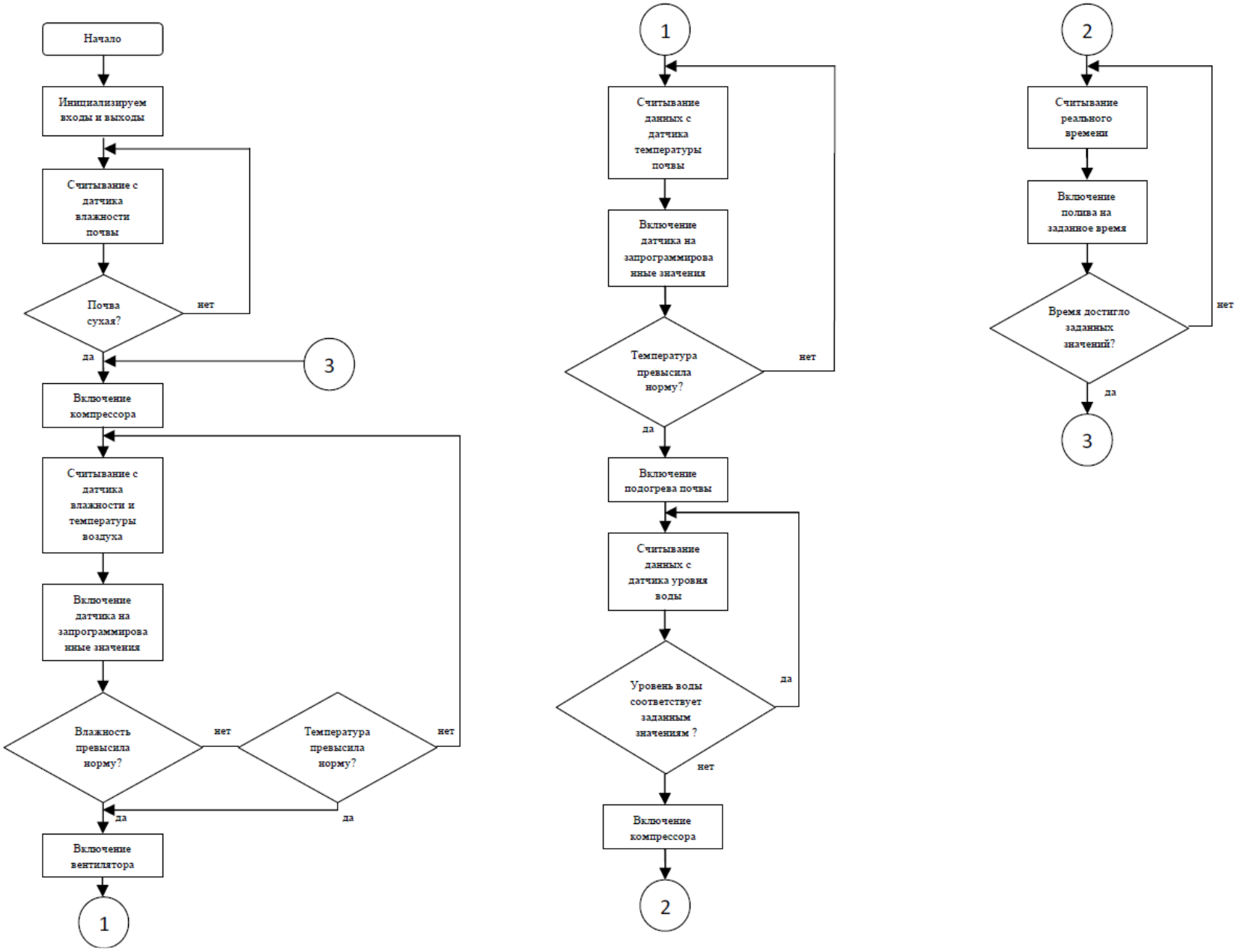

Рисунок 25 – Алгоритм работы системы «Умная теплица с микроконтрольным управлением».

На рисунке 26 показан плакат иллюстрирующую работу системы, в данном плакате можно увидеть, за что отвечает каждый элемент, какие датчики контролируют включение платы реле и что включает определенное реле.

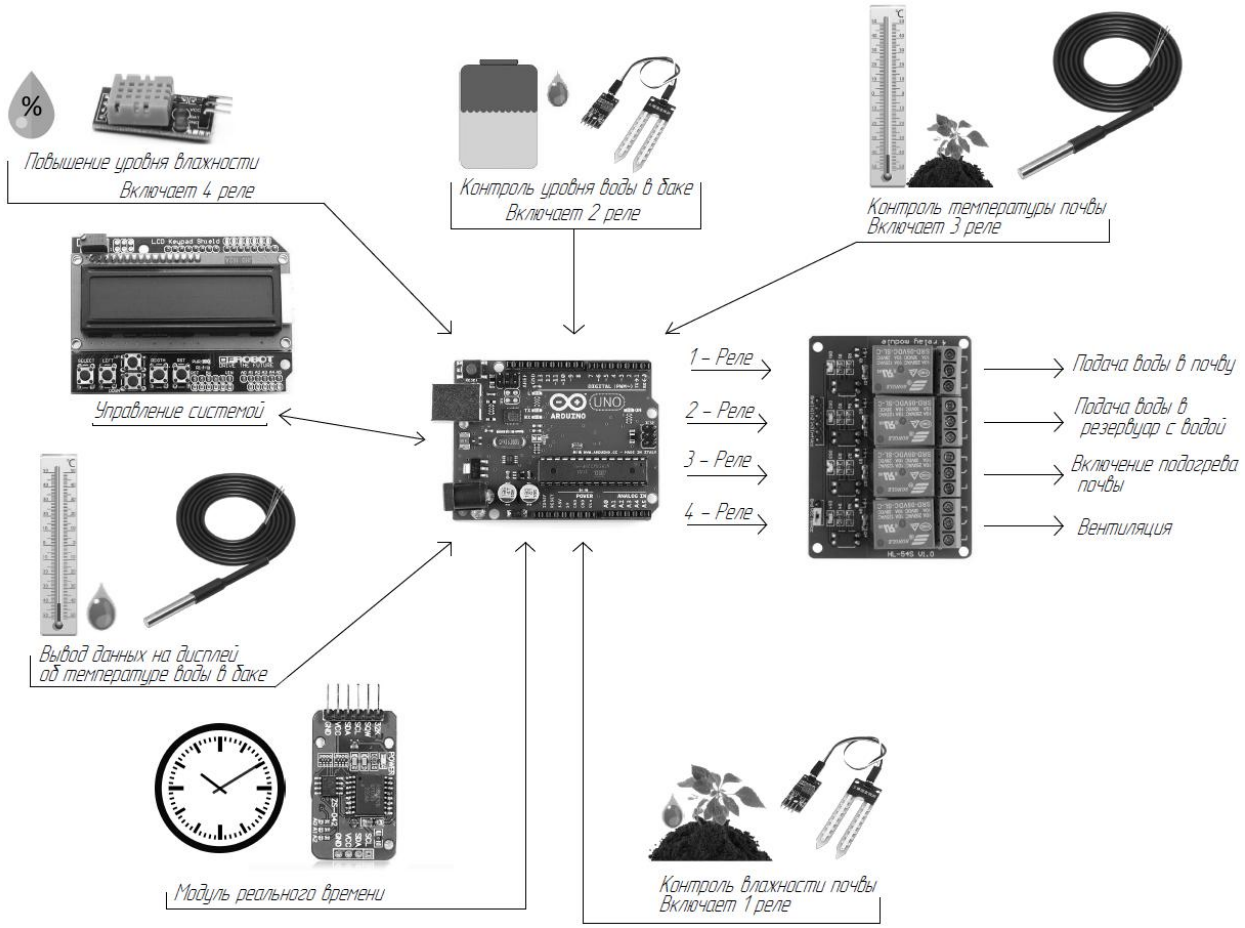

Рисунок 26 - Плакат иллюстрирующую работу системы.

Для того что бы можно было подробно узнать работу устройства, был разработан плакат с временными диаграммами. Графики временных диаграмм показаны на рисунках 27, 28, 29.

На рисунке 27 мы можем рассмотреть работу нагрева почвы, в момент упадка температуры на вход устройства поступает высокий потенциал, и в ходе чего включается реле, по повышению температуры почвы, датчик сигнализирует о том что, температура достигла заданных значений и на входе мы получим низкий потенциал, который отключит реле.

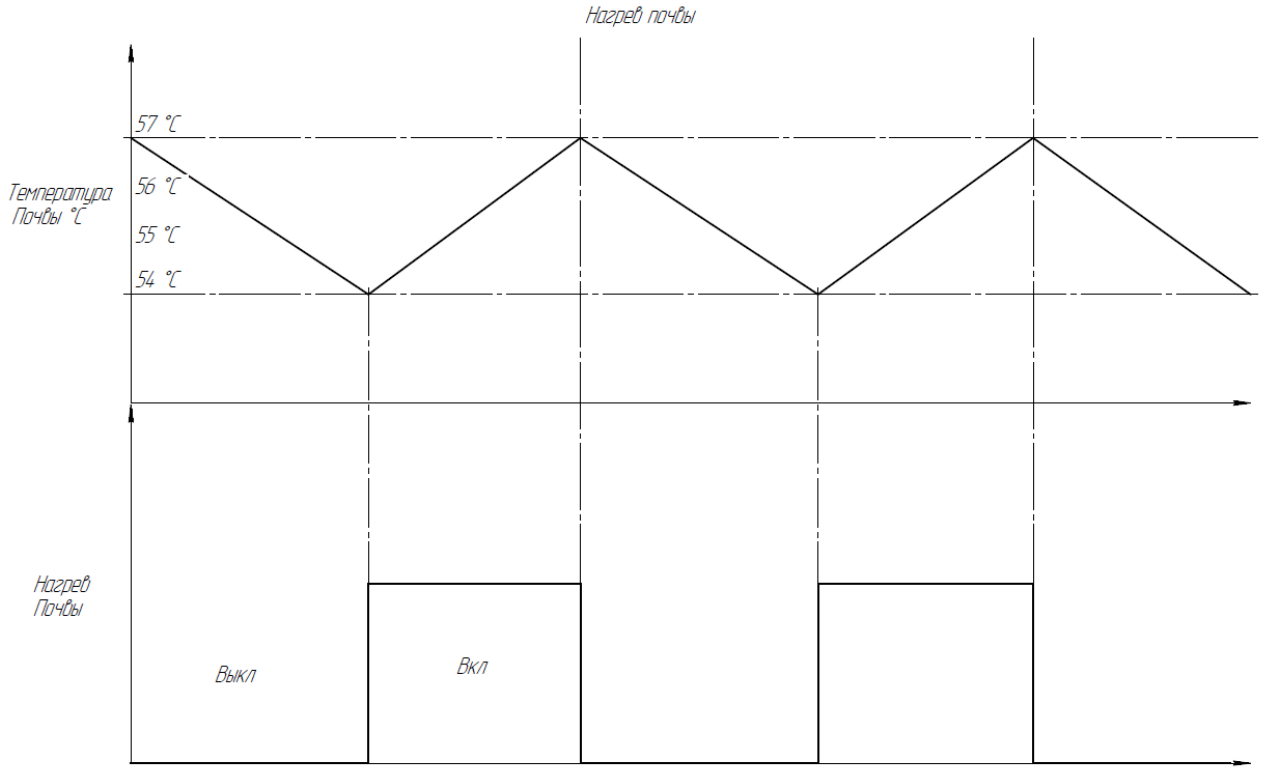

Рисунок 27 – График нагрева почвы.

В следующем графике влажность воздуха рисунок 28 мы видим что влажность воздуха начинает постепенно расти, до того значения, которая выставлена в настройках устройства, по мере достижения порога выставленных значений, на выходе мы получим высокий потенциал, что и включит реле для проветривания помещения, по мере включения вентиляций, влажность начнет быстро падать и в ходе упадка влажности воздуха, на выходе мы получим низкий потенциал, что и выключит вентиляцию.

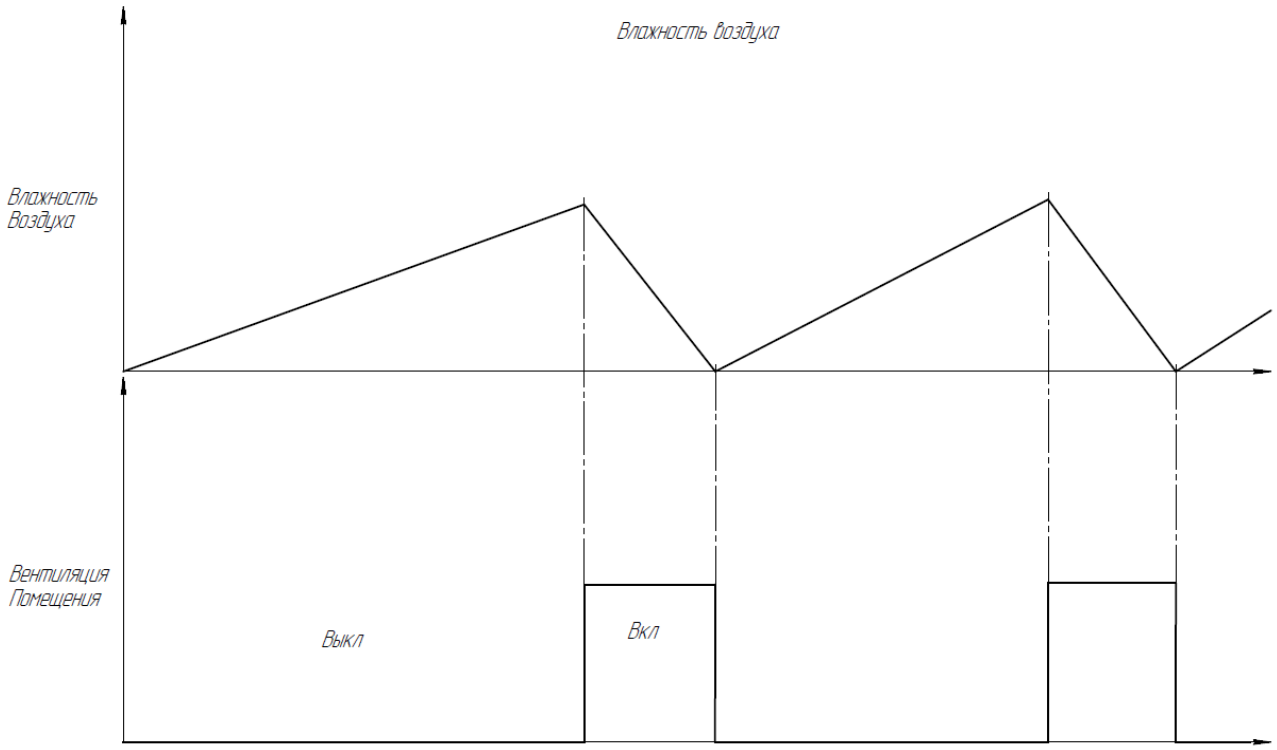

Рисунок 28 – График влажности воздуха.

Далее в двух графиках рисунок 29 «Контроль влажности почвы, Контроль уровня воды в баке» можно увидеть, как работает поливочная система и контроль уровня воды в баке. В графике «Контроль влажности почвы» мы видим, что почва начинает высыхать, в момент достижения нижнего порога влажности почвы, на выход поступает высокий потенциал и включается компрессор для подкачки воды в почву, по повышению влажности почвы, на выходе мы получаем низкий потенциал и реле отключит компрессор.

На графике «Контроль уровня воды в баке» мы можем наблюдать выкачивание воды из бака. В момент включения компрессора для подачи воды в почву, в баке начинает падать уровень воды, по достижению нижнего порога уровня воды в баке, на выходе мы получим высокий потенциал, в ходе чего реле включит компрессор для подкачки воды в бак, по мере повышения уровня воды в баке, датчик просигнализирует о достижений заданных значений и компрессор будет отключен.

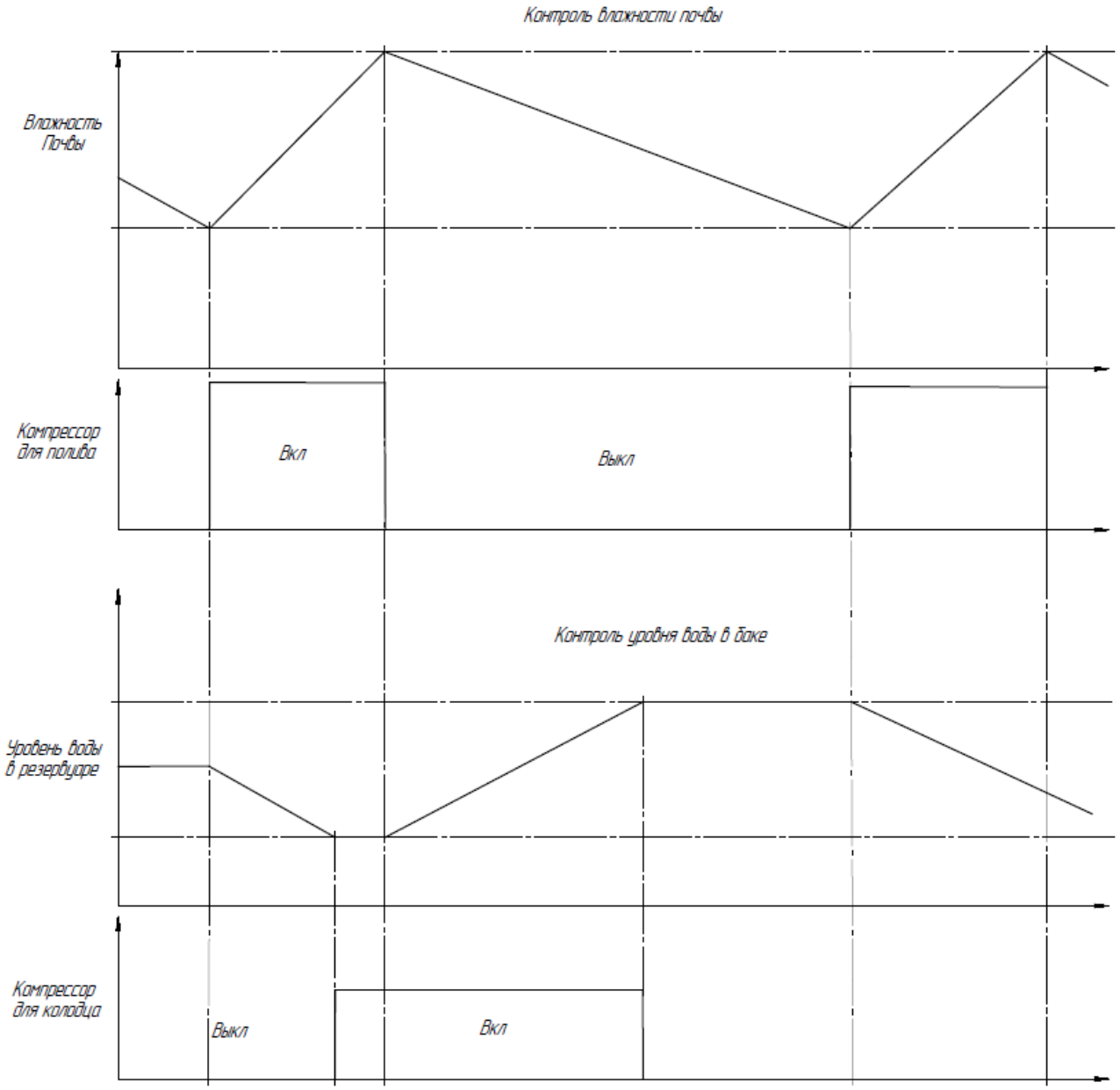

Рисунок 29 – График поливочной системы.

3.2 Разработка программной части устройства

Для реализаций устройства «Умная теплица с микроконтрольным управлением», была выбрана среда программирования Arduino Software (IDE). Есть и другие способы написания прошивок и загрузки скетчей в Arduino, но самое простое — это использовать Arduino IDE.

Используя программную среду Arduino IDE, можно, основываясь лишь на знаниях C++, решать самые разные творческие задачи, связанные с программированием и моделированием.

Основным достоинством является то, что платформа ориентирована на непрофессиональных пользователей. То есть любой может создать своего робота вне зависимости от знаний программирования и собственных навыков.

В Arduino Software (IDE) встроен текстовый редактор программного кода, для ввода сообщений, окна вывода текста (консоли), вкладки инструментов с кнопками зачастую используемых команд и нескольких меню. Для загрузки скеча (программы) и связи среда разработки подключается к платформе Arduino. Внешний вид программы изображен на рисунке 30.

Библиотеки позволяют упростить работу над программой, и дает дополнительную функциональность скетчам, например, при выполнении работы с аппаратной частью или же при проверке данных.

При разработке программы были использованы следующие библиотеки:

#include <EEPROM.h> энергонезависимая память, в которой можно сохранять какие-либо данные, которые будут доступны после отключения питания.

#include <LiquidCrystal.h>. С помощью этой библиотеки мы можем управлять жидкокристаллическими дисплеями (LCD) имея контроллер «HD44780» (или аналогах), который применяется в основном символьных LCD дисплеях. Эта библиотека работает либо в 4-х, либо в 8-ми bit режиме (применяется 4 или 8 линий информационных данных в дополнение к линиям управления «RS, EN» и, при нужде, «RW»).

#include <DHT.h> предназначен для работы с датчиками температуры и влажности.

#include <iarduino_RTC.h> Библиотека позволяет читать и записывать время RTC модулей.

#include <DallasTemperature.h> Данная библиотека нужна для датчиков температур DS18B20, что бы упростить работу с датчиком.

#include <OneWire.h> Данная библиотека нужна для датчиков температур DS18B20, что бы считывать данные с термометра, один из датчиков контролирует температуру почвы, другой показывает температуру воды в баке.

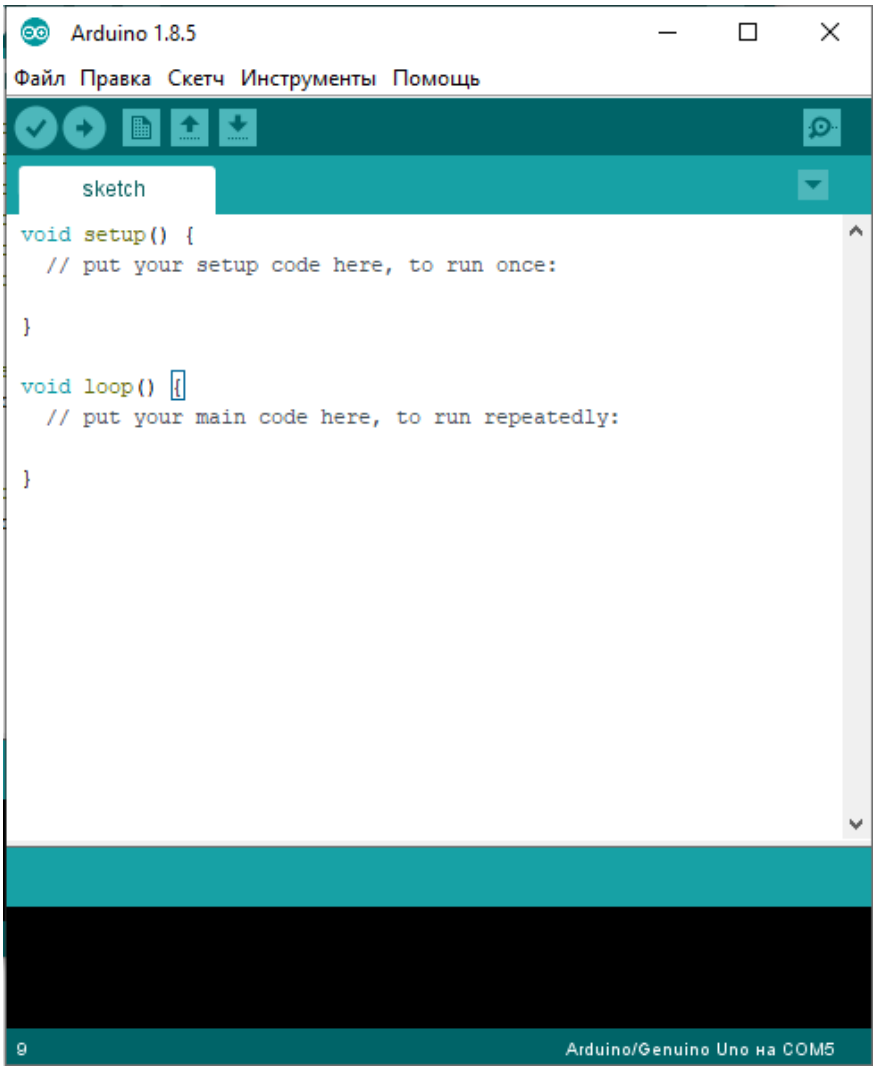

Рисунок 30 - Внешний вид программы Arduino Software (IDE).

Функция setup() вызывается только один раз при старте микроконтроллера. Именно она выставляет все базовые настройки. Функция loop() — циклическая. Она вызывается в бесконечном цикле на протяжении всего времени работы микроконтроллера.

Разделение на сегменты программы функциями дает возможность создавать части программы, которые в свою очередь исполняют поставленные задания. Во время завершение выполнения программы происходит возврат в место, в начальное стадие, откуда была начата функция. Функцию создают для цикличного выполнения одинакового действия несколько раз подрят.

Преимущества функции:

1. позволяет организовывать программу. В большинстве случаев помогают заранее составить концепцию программы.

2. кодируют одно действие в одном месте программы. Далее необходимо только отладить код функции.

3. сокращают шансы на появление ошибки при необходимости изменения кода.

4. сокращают текст скетчей и делают его компактным, т.к. некоторые секции используются много раз.

5. облегчают использование кода в других программах делая его модульным. В этом случае функции обладают еще одним небольшим преимуществом, делая код программы легким для чтения.

Программа «Умной теплицы с микроконтрольным управлением» представлена в приложений А.

4. Конструктивно-эксперементальный раздел

4.1 Изготовление макета

Для демонстраций работы системы, были выбраны следующие устройства:

- Вентилятор для проветривания теплицы;

- Светодиодная индикация (состоящий из 4-х красных светодиодов) для имитаций включения подогрева, на датчик температуры намотана медная проволока, для того что бы можно было быстро нагреть датчик и показать работоспособность стенда;

- 2 компрессора, один из них подает воду в почву, другой подает воду в резервуар с водой.

Для того что бы продемонстрировать работоспособность вентиляций, датчик влажности воздуха установлен в контейнер с водой, с надписью «Почва», благодаря тому что в контейнере будет находиться вода, датчик влажности будет реагировать на повышение влажности в воздухе. По повышению уровня воды в контейнере и по достижению выставленных значений, Arduino даст сигнал на включение реле, что в конечном итоге включит вентилятор и со временем влажность начнет падать, на рисунке 27 показан плакат с временными диаграммами, в котором видно как работает контроль влажности воздуха, а так же остальные устройства.

На рисунке 31 показан внешний вид демонстрационной системы.

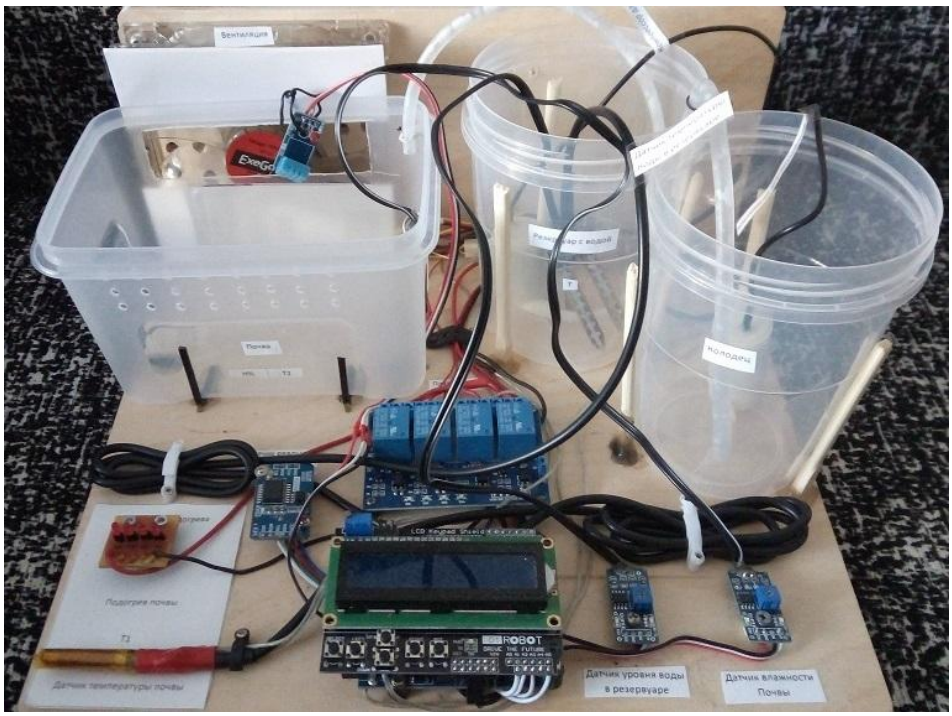

Рисунок 31 – Внешний вид демонстрационной системы.

Все устройства управляются с помощью 4-х канального платы реле. На каждое устройство подается напряжение 5 В.

Для наглядной демонстраций устройства, все компоненты «Умной теплицы с микроконтрольным управлением» были установлены на древесную поверхность. Демонстрационный подогрев почвы, был отдельно выделен на стенде под названием «Индикация подогрева», чуть ниже находится датчик температуры почвы. На рисунке 32 показан внешний вид индикаций подогрева.

Рисунок 32 – Внешний вид индикаций подогрева.

Для того что бы продемонстрировать работоспособность компрессоров, на стенд были установлены 3 контейнера, на каждом контейнере присутствует надпись, на 1 контейнере «Почва» во 2 контейнере «Резервуар с водой» и в 3 «Колодец», так же на некоторых контейнерах присутствуют надписи со значениями (T, T1, T2, H%) что дает нам понять, о том что в этих контейнерах находится тот или иной датчик.

В двух контейнерах находятся по 1 компрессору, первый в колодце, второй в резервуаре с водой, когда наступает время полива почвы, включается второй компрессор и начинает выкачивать воду с резервуара, по мере осушения воды с резервуара, датчик уровня воды реагирует на понижение уровня воды, и вследствие этого включается первый компрессор который находится в колодце. На рисунке 33 показано расположение двух компрессоров для подачи воды.

Рисунок 33 – Расположение двух компрессоров для подачи воды.

Контроль влажности воздуха и температуры демонстрируются с помощью заполнения контейнера водой, благодаря тому, что датчик будет находиться на поверхности воды, мы сможешь продемонстрировать работоспособность вентиляций, в контейнере присутствует отверстие, с помощью которого будет, происходит проветривание. На рисунке 34 показан внешний вид вентиляций.

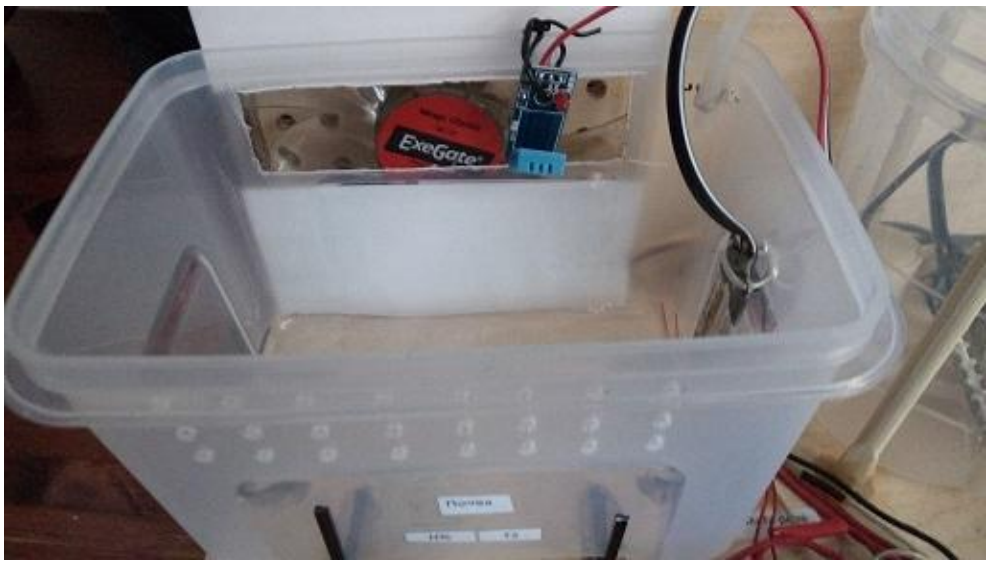

Рисунок 34 - Внешний вид вентиляций.

Так же на стенд установлен выключатель демонстрационных устройств, это нужно для того что бы экстренно отключить демонстрационные устройства которые вышли из под контроля, например компрессоры могут затопить весь стенд.

Некоторым компрессорам нужно, что бы емкость была заполнена до определенного уровня, если этот уровень воды не будет достигнут, то в этом случае в компрессоре образуется воздушный карман, который не даст выкачивать воду. Если компрессор слабо выкачивает воду или совсем не выкачивает, тогда нужно воспользоваться выключателем предварительно залив водой контейнер до половины. На рисунке 35 показан внешний вид выключателя.

Рисунок 35 - Внешний вид выключателя.

4.2 Отладка макета и экспериментальные исследования

Перед установкой всех комплектующих на стенд, каждый элемент отдельно был проверен на работоспособность, данная отладка нужна для выявления недоброжелательных ошибок при работе стенда. Для корректной работы устройства был выбран блок питания на 9 В (1А), и в ходе замеров токового потребления устройства было выяснено что, при работе датчиков и реле потребление по току составляет примерно 400 мА, потребление демонстрационных устройств 2х компрессоров, куллера, подогрева почвы, и нагревательного элемента составило примерно 535 мА, так же преобразовательный элемент потребляем 15 мА, в итоге полное потребление по току составило 950 мА, можно сказать, что данный блок питания подходит для этой системы. После выяснения потребления тока, была проведена проверка каждого элемента, все комплектующие были подключены к Arduino Uno, и так же была проведена диагностика всех комплектующих на работоспособность.

В ходе проведений отладки, были обнаружены ошибки и недостатки, которые были устранены для полноценной работоспособности стенда.

На рисунке 36 представлено меню «Умной теплицы с микроконтрольным управлением».

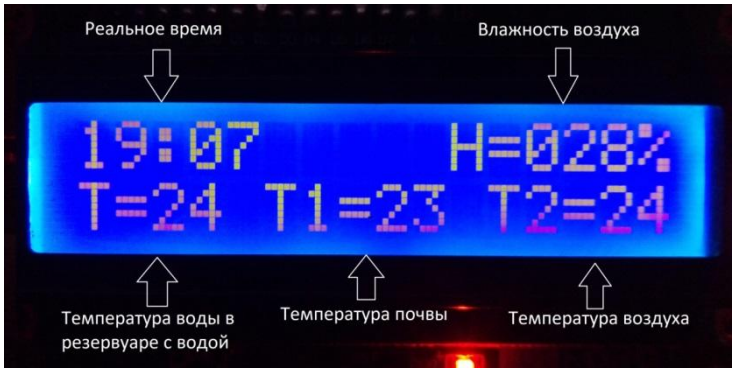

Рисунок 36 – Меню «Умной теплицы с микроконтрольным управлением».

Для того что бы попасть в меню, нужно нажать на кнопку Select, для перемещения по меню нужно использовать кнопки Up и Down, так же кнопка Select в меню исполняется как ввод (вход под меню) и с помощью этой же кнопки сохраняются введенные значения. Находясь в меню все устройства, будут приостановлены, это нужно для корректной работы системы. Для выхода из подменю нужно нажать на кнопку Left или Right. На рисунке 37 изображены кнопки для управления системой.

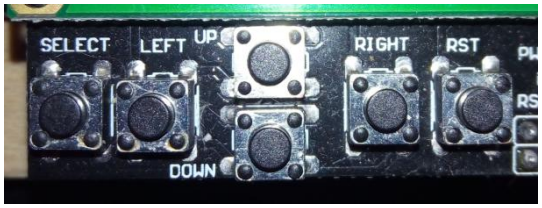

Рисунок 37 – Кнопки для управления системой.

Для того что бы настроить реальное время, нужно перейти в подменю Set Time (Рисунок 38).

Рисунок 38 – Настройка реального времени.

После настройки реального времени, переходим в подменю Watering Plants, здесь мы настраиваем время полива 2 раза в день (Рисунок 39 и 40). Если мы выставим значения, к примеру, в первом поливе 00:00 значения как в Start Time и End Time, то первый полив не будет работать, второй полив будет по прежнему работать.

Рисунок 39 – Настройка полива 1 раз в день

Рисунок 40 – Настройка полива 2 раза в день.

После настройки времени полива, переходим к настройкам влажности воздуха (Рисунок 41).

Рисунок 41 – Настройка влажности воздуха.

Далее переходим к настройке температуры (Рисунок 42).

Рисунок 42 – Настройка температуры воздуха.

Последнее что осталось настроить, это температура почвы (Рисунок 43).

Рисунок 43 – Настройка температуры почвы.

Так же в конце меню есть функция очистки всех настроек Clear Setting (Рисунок 44).

Рисунок 44 – Очистка всех настроек.

Заключение

В ходе выполнения бакалаврской работы, была разработана система под названием «Умная теплица с микроконтрольным управлением». С помощью меню, устройство можно перепрограммировать на разные виды растений, которые в свою очередь очень востребованы в определенной влажности и температуре почвы.

В процессе выполнения была рассмотрена актуальность «Умной теплицы с микроконтрольным управлением» и существующие решения в плане контроля теплиц. Разработана структурная схема, которая нам показывает, сколько устройств присутствует на этой системе, так же как и принципиальная схема. Были подобраны необходимые компоненты, такие как: Arduino Uno (сердце системы), LCD Key Pad Shield с помощью которой мы выставляем нужные нам значения в меню, датчик для снятия температуры почвы, датчик для снятия температуры воды в баке с водой, Датчик влажности почвы, Датчик уровня воды, Плата реле состоящий из 4-х коннекторов, Датчик влажности воздуха и температуры DHT11, Датчик реального времени DS3231. Разработан алгоритм работы, после чего была разработана программная часть устройства. Изготовлен макет и был отлажен для демонстраций.

Список используемой литературы

1. Курдюмов Н.И. Малышевский: «Умная теплица» Владис, 2014 г. 192 стр.
2. Зорина А.А. «Богатый урожай из теплицы и парников» Центрполиграф, 2016 г. 128 стр.
3. Знакомство с Arduino [Электронный ресурс] URL: http://wiki.amperka.ru/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D1%8B:arduino-uno (Дата обращения: 24.03.2019)
4. Программирование на Arduino [Электронный ресурс] URL: http://arduino.ru/Reference (Дата обращения: 31.03.2019)
5. Датчики и модули для Arduino [Электронный ресурс] URL: https://arduinomaster.ru/datchiki-arduino/ (Дата обращения: 12.04.2019)
6. Самые популярные датчики для Arduino [Электронный ресурс] URL: http://electrik.info/microcontroller/1447-samye-populyarnye-datchiki-dlya-arduino.html (Дата обращения: 15.04.2019)
7. Умная теплица: как максимально автоматизировать свою кормилицу? [Электронный ресурс] URL: https://vasha-teplitsa.ru/obustroistvo/umnaya-teplica.html (Дата обращения: 27.04.2019)
8. Таймер для полива своими руками: советы мастера по изготовлению устройства [Электронный ресурс] URL: https://diz-cafe.com/tech/tajmer-poliva-svoimi-rukami.html (Дата обращения: 01.05.2019)
9. Умная теплица по Курдюмову или Нефантастическая история [Электронный ресурс] URL: http://teplicnik.ru/obustrojstvo/umnaya-teplica-po-kurdyumovu.html (Дата обращения: 13.05.2019)
10. Умная теплица своими руками: подробная инструкция: [Электронный ресурс] URL: https://parnik-teplitsa.ru/umnaya-teplica-180 (Дата обращения: 17.05.2019)
11. Умная теплица своими руками – что это? [Электронный ресурс] URL: https://mirfermera.ru/1-kak-sdelat-umnuyu-teplicu-svoimi-rukami.html (Дата обращения: 19.05.2019)

12. «Умная теплица» Николай Курдюмов, Константин Малышевский [Электронный ресурс] URL: https://coollib.com/b/344474/read (Дата обращения: 19.05.2019)

13. Что такое умная теплица и как сделать автоматическое управление своими руками [Электронный ресурс] URL: http://teplicno.ru/obustr/umnaya-teplica.html (Дата обращения: 19.05.2019)

14. Умная теплица: особенности и преимущества конструкции [Электронный ресурс] URL: https://teplichniku.ru/raznovidnosti/chto-takoe-umnaia-teplitca/ (Дата обращения: 21.05.2019)

15. Датчик влажности почвы [Электронный ресурс] URL: http://digitrode.ru/computing-devices/mcu_cpu/289-arduino-i-datchik-vlazhnosti-pochvy.html (Дата обращения: 21.05.2019)

16. Цифровой датчик температур DS18B20 [Электронный ресурс] URL: http://arduino-diy.com/arduino-tsifrovoy-datchik-temperatury-DS18B20 (Дата обращения: 21.05.2019)

17. Датчик влажности воздуха и температуры [Электронный ресурс] URL: http://edurobots.ru/2015/02/arduino-dlya-nachinayushhix-urok-9-podklyuchenie-datchika-temperatury-i-vlazhnosti-dht11-i-dht22/ (Дата обращения: 21.05.2019)

18. LCD key Pad Shield [Электронный ресурс] URL: http://helpduino.ru/LCD_Keypad_Shield.html (Дата обращения: 22.05.2019)

19. Часы реального времени [Электронный ресурс] URL: https://lesson.iarduino.ru/page/podklyuchenie-rtc-chasy-realnogo-vremeni-ds1302-ds1307-ds3231-k-arduino/ (Дата обращения: 22.05.2019)

20. Модуль реле [Электронный ресурс] URL: http://zelectro.cc/relayModule (Дата обращения: 22.05.2019).

Приложение А

Программа Умной теплицы с микроконтрольным управлением

```
#include <EEPROM.h>
#include <LiquidCrystal.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <DHT.h>
#include <iarduino_RTC.h>


void setup() {
  DeviceSetup();
}

void loop() {
  DeviceLoop();
}

/* ---------------------------------------------------------------------------
 *                          Настройки пинов
 * --------------------------------------------------------------------------- */
#define PIN_BUTTON      A0
#define PIN_GROUND_N0   A2
#define PIN_GROUND_N1   A3
#define PIN_HUMIDITY    3
#define PIN_TEMP        2
#define PIN_RELE_N1     13
#define PIN_RELE_N2     12
#define PIN_RELE_N3     11
#define PIN_RELE_N4     A1
/* ---------------------------------------------------------------------------
 *                            Константы
 * --------------------------------------------------------------------------- */
#define BUTTON_NONE    0x00
#define BUTTON_RIGHT   0x01
#define BUTTON_UP      0x02
#define BUTTON_DOWN    0x03
#define BUTTON_LEFT    0x04
#define BUTTON_SELECT  0x05

#define EEPROM_ADDRESS 0x00

#define MENU_DELAY     1000

#define BOX_WATTER     550
#define BOX_GROUND     600
/* ---------------------------------------------------------------------------
 *                            Структуры
 * --------------------------------------------------------------------------- */
```

```
struct ssWateringTime {
  byte hour;
  byte min;
};

struct ssWatering {
  ssWateringTime start;
  ssWateringTime end;
};

struct ssHumidity {
  byte start;
  byte end;
};

struct ssAirTemperature {
  byte start;
  byte end;
};

struct ssSoilTemperature {
  byte start;
  byte end;
};

struct ssSetting {
  ssWatering        Watering[4];
  ssHumidity        Humidity;
  ssAirTemperature  AirTemperature;
  ssSoilTemperature SoilTemperature;
};

struct ssUI {
  byte menu;
  byte pos;
  byte posCount;
  byte status;
};

struct ssButton {
  unsigned long time;
};

struct ssTEMP {
  int t;
  int t1;
  byte H;
  int Humidity;
  int Temperature;
  byte H1;
  byte H2;
```

```
byte min;
  byte hour;
  byte Watter[2];
  unsigned long time;
};

struct ssDevice {
  ssSetting Setting;
  ssUI      UI;
  ssButton  Button;
  ssTEMP    TEMP;
};
/* ---------------------------------------------------------------------------
 *                              Объекты
 * --------------------------------------------------------------------------- */

LiquidCrystal lcd(8, 9, 4, 5, 6, 7 );

OneWire  oneWire(PIN_TEMP);
DallasTemperature ds(&oneWire);
DHT dht(PIN_HUMIDITY, DHT11);
iarduino_RTC time(RTC_DS3231);

DeviceAddress sensor1 = {0x28, 0x92, 0x03, 0x77, 0x91, 0x06, 0x02, 0x34};
DeviceAddress sensor2 = {0x28, 0xBB, 0x5A, 0x77, 0x91, 0x07, 0x02, 0xB6};

/* ---------------------------------------------------------------------------
 *                              Переменные
 * --------------------------------------------------------------------------- */
ssDevice Device;


byte button_is(void) {
  int buttonValue = analogRead(PIN_BUTTON);
  if (buttonValue < 100) {
    return BUTTON_RIGHT;
  } else if (buttonValue < 200) {
    return BUTTON_UP;
  } else if (buttonValue < 400) {
    return BUTTON_DOWN;
  } else if (buttonValue < 600) {
    return BUTTON_LEFT;
  } else if (buttonValue < 800) {
    return BUTTON_SELECT;
  }
  return BUTTON_NONE;
}

byte button_pressed(void) {
  byte button = 0;
```

```
  byte _ret = button_is();
  if (_ret != BUTTON_NONE) Device.Button.time = millis();
  while (1) {
    button = button_is();
    if (button == BUTTON_NONE) {
      break;
    }
  }
  return _ret;
}

unsigned long button_time(void) {
  return (millis() - Device.Button.time);
}


void DeviceSetup(void) {
  Serial.begin(9600);
  sEEPROM();
  sLCD();
  sGROUND();
  sRELE();
  sTEMP();
  sHumidity();
  sRTC();
}

void DeviceLoop(void) {
  lEEPROM();
  LCD();
  GROUND();
  RELE();
  TEMP();
  Humidity();
  RTC();
}


void sEEPROM(void) {
  EEPROM.get(EEPROM_ADDRESS, Device.Setting);
}

void lEEPROM(void) {

}

void saveEEPROM(void) {
  EEPROM.put(EEPROM_ADDRESS, Device.Setting);
}
```

```
void clearEEPROM(void) {
  char _message[16];
  lcd.clear();
  lcd.print("Clear Setting");
  lcd.setCursor(0, 1);
  for (int i = 0 ; i < EEPROM.length() ; i++) {
    lcd.setCursor(0, 1);
    sprintf(_message, "%d of %d", i, EEPROM.length());
    lcd.print(_message);
    EEPROM.write(i, 0);
  }
}

void sGROUND() {
  pinMode(PIN_GROUND_N0, INPUT); // Земля
  pinMode(PIN_GROUND_N1, INPUT); // Бочка
}

void GROUND() {
  if (Device.UI.menu != 0) return;
  int val1 = analogRead(PIN_GROUND_N0);
  int val2 = analogRead(PIN_GROUND_N1);



  byte Valt;

  if (val2 < BOX_WATTER) {
    digitalWrite(PIN_RELE_N1, HIGH);
    //Serial.println("PIN_RELE_N1 -> LOW");
  }else{
    digitalWrite(PIN_RELE_N1, LOW);
    //Serial.println("PIN_RELE_N1 -> HIGH");
  }

  if (val1 < BOX_GROUND) {
    Valt = 0;
  }else{
    Valt = 1;
  }
  if (Device.TEMP.Watter[0] == 1 || Device.TEMP.Watter[1] == 1 || Valt == 1) {
    digitalWrite(PIN_RELE_N2, LOW);
  }else{
    digitalWrite(PIN_RELE_N2, HIGH);
  }

}


void sHumidity() {
```

```
 dht.begin();
  Device.TEMP.H1 = 0;
  Device.TEMP.time = millis();
 }

 void Humidity() {
  //Считываем влажность
  if (Device.UI.menu != 0) return;
  if ( (millis() - Device.TEMP.time) < 1000) return;
  Device.TEMP.time = millis();
  Device.TEMP.Humidity = int(dht.readHumidity());
  Device.TEMP.Temperature = int(dht.readTemperature());
  if (Device.TEMP.Temperature > 100) { Device.TEMP.Temperature = 100; }
  if (Device.TEMP.Temperature < 0 )  { Device.TEMP.Temperature = 0;   }
  if (Device.TEMP.Humidity > 100)    { Device.TEMP.Humidity = 100;    }
  if (Device.TEMP.Humidity < 0)      { Device.TEMP.Humidity = 0;      }

  if (Device.Setting.Humidity.start > Device.Setting.Humidity.end) {
   if (Device.TEMP.Humidity >= Device.Setting.Humidity.start && Device.TEMP.H1 ==
0) {
    Device.TEMP.H1 = 1;
   }
   if (Device.TEMP.Humidity <= Device.Setting.Humidity.end && Device.TEMP.H1 ==
1) {
    Device.TEMP.H1 = 0;
   }
  }else{
   if(Device.TEMP.H1 == 1) {
    Device.TEMP.H1 = 0;
   }
  }

  if (Device.Setting.AirTemperature.start > Device.Setting.AirTemperature.end) {
   if (Device.TEMP.Temperature >= Device.Setting.AirTemperature.start &&
Device.TEMP.H2 == 0) {
    Device.TEMP.H2 = 1;
   }
   if (Device.TEMP.Temperature <= Device.Setting.AirTemperature.end &&
Device.TEMP.H2 == 1) {
    Device.TEMP.H2 = 0;
   }
  }else{
   if(Device.TEMP.H2 == 1) {
    Device.TEMP.H2 = 0;
   }
  }

  if ( Device.TEMP.H1 == 1 || Device.TEMP.H2 == 1) {
   digitalWrite(PIN_RELE_N4, LOW);
  }else{
```

```
    digitalWrite(PIN_RELE_N4, HIGH);
   }
  }

  /* ---------------------------------------------------------------------------
   *                    LCD Драйвер управления экраном
   * --------------------------------------------------------------------------- */
  void sLCD(void) {
   lcd.begin(16, 2);

   Device.UI.menu = 0;
  }

  void LCD() {
   char _message[64];
   lcd.setCursor(0, 0);
   Device.UI.status = button_pressed();

   //Serial.println(Device.UI.status);

   if (Device.UI.menu == 0) {

    if (Device.UI.status == BUTTON_SELECT) {
     Device.UI.menu = 1;
     Device.UI.pos = 0;
     lcd.clear();
     return;
    }

    sprintf(_message, "%s     H=%03d%%", time.gettime("H:i"), Device.TEMP.Humidity);
    lcd.print(_message);

    lcd.setCursor(0, 1);

    sprintf(_message, "T=%02d T1=%02d T2=%02d", Device.TEMP.t, Device.TEMP.t1,
Device.TEMP.Temperature);
    lcd.print(_message);

   }else{

    switch (Device.UI.menu) {
     case 1:
      UIClear();
      UIposTime("Menu", "Set Time", 2);
      UIpos("Menu", "Watering Plants", 3);
      UIpos("Menu", "Air Humidity", 4);
      UIpos("Menu", "Air Temperature", 5);
      UIpos("Menu", "Soil Temperature", 6);
      UIClear("Menu", "Clear Setting");
      UIposControl();
```

```
  UIBackControl(0);
  break;
case 2:
  UIClear();
  UIbyte("Set Time", "Hour=", 0, 23, &Device.TEMP.hour);
  UIbyte("Set Time", "Min=", 0, 59, &Device.TEMP.min);
  UIposControl();
  UISelectTime(1);
  break;
case 3:
  UIClear();
  UIpos("Watering Plants", "1 Watering", 30);
  UIpos("Watering Plants", "2 Watering", 31);
  UIposControl();
  UIBackControl(1);
  break;
case 30:
  UIClear();
  UIpos("1 Watering", "Start Time", 130);
  UIpos("1 Watering", "End Time", 131);
  UIposControl();
  UIBackControl(3);
  break;
case 31:
  UIClear();
  UIpos("2 Watering", "Start Time", 132);
  UIpos("2 Watering", "End Time", 133);
  UIposControl();
  UIBackControl(3);
  break;
case 130:
  UIClear();
  UIbyte("Start Time", "Hour=", 0, 24, &Device.Setting.Watering[0].start.hour);
  UIbyte("Start Time", "Min=", 0, 59, &Device.Setting.Watering[0].end.min);
  UIposControl();
  UISelect(30);
  break;
case 131:
  UIClear();
  UIbyte("End Time", "Hour=", 0, 24, &Device.Setting.Watering[1].start.hour);
  UIbyte("End Time", "Min=", 0, 59, &Device.Setting.Watering[1].end.min);
  UIposControl();
  UISelect(31);
  break;
case 132:
  UIClear();
  UIbyte("Start Time", "Hour=", 0, 24, &Device.Setting.Watering[2].start.hour);
  UIbyte("Start Time", "Min=", 0, 59, &Device.Setting.Watering[2].end.min);
  UIposControl();
  UISelect(30);
```

```
  break;
    case 133:
      UIClear();
      UIbyte("End Time", "Hour=", 0, 24, &Device.Setting.Watering[3].start.hour);
      UIbyte("End Time", "Min=", 0, 59, &Device.Setting.Watering[3].end.min);
      UIposControl();
      UISelect(30);
      break;
    case 4:
      UIClear();
      UIbyte("Air Humidity", "ON=", 1, 100, &Device.Setting.Humidity.start);
      UIbyte("Air Humidity", "OFF=", 1, 100, &Device.Setting.Humidity.end);
      UIposControl();
      UISelect(1);
      break;
    case 5:
      UIClear();
      UIbyte("Air Temperature", "ON=", 1, 100, &Device.Setting.AirTemperature.start);
      UIbyte("Air Temperature", "OFF=", 1, 100, &Device.Setting.AirTemperature.end);
      UIposControl();
      UISelect(1);
      break;
    case 6:
      UIClear();
      UIbyte("Soil Temperature", "ON=", 1, 100, &Device.Setting.SoilTemperature.start);
      UIbyte("Soil Temperature", "OFF=", 1, 100, &Device.Setting.SoilTemperature.end);
      UIposControl();
      UISelect(1);
      break;
   };

  }
}
/* ---------------------------------------------------------------------------
 *              UI LCD - Интерфейс для общения с пользователем
 * --------------------------------------------------------------------------- */

void UIClear(void) {
  Device.UI.posCount = 0;
}

void UIpos(char *title, char *text, byte menu) {
  if (Device.UI.posCount == Device.UI.pos) {
    lcd.print(title);
    lcd.setCursor(0, 1);
    lcd.print(text);
    if (Device.UI.status == BUTTON_RIGHT || Device.UI.status == BUTTON_SELECT) {
      Device.UI.menu = menu;
      Device.UI.pos = 0;
      lcd.clear();
```

```
    }
  }
  Device.UI.posCount++;
}

void UIposTime(char *title, char *text, byte menu) {
  if (Device.UI.posCount == Device.UI.pos) {
    lcd.print(title);
    lcd.setCursor(0, 1);
    lcd.print(text);
    if (Device.UI.status == BUTTON_RIGHT || Device.UI.status == BUTTON_SELECT) {
      Device.UI.menu = menu;
      Device.UI.pos = 0;
      Device.TEMP.min = time.minutes;
      Device.TEMP.hour = time.Hours;
      lcd.clear();
    }
  }
  Device.UI.posCount++;
}

void UIposControl(void) {
  Serial.println(Device.UI.pos);
  if (Device.UI.status == BUTTON_DOWN) {
    Device.UI.pos = Device.UI.pos + 1;
    if ((Device.UI.posCount - 1) < Device.UI.pos) {
      Device.UI.pos = (Device.UI.posCount - 1);
    }
    lcd.clear();
    return;
  }
  if (Device.UI.status == BUTTON_UP) {
    Device.UI.pos = Device.UI.pos - 1;
    if (Device.UI.pos == 255) {
      Device.UI.pos = 0;
    }
    lcd.clear();
    return;
  }
}

void UIBackControl(byte _bk) {
  if (Device.UI.status == BUTTON_LEFT) {
    Device.UI.pos = 0;
    Device.UI.menu = _bk;
    lcd.clear();
  }
}

void UIbyte(char *title, char *name, byte min, byte max, byte *_byte) {
```

```
  char _message[16];
  if (Device.UI.posCount == Device.UI.pos) {
    lcd.print(title);
    lcd.setCursor(0, 1);
    sprintf(_message, "%s%d", name, *_byte);
    lcd.print(_message);
    if (Device.UI.status == BUTTON_RIGHT) {
      *_byte = *_byte + 1;
      if (*_byte > max) *_byte = max;
      lcd.clear();
    }
    if (Device.UI.status == BUTTON_LEFT) {
      *_byte = *_byte - 1;
      if (min == 0 && *_byte == 255) {
        *_byte = 0;
      }
      if (*_byte < min) *_byte = min;
      lcd.clear();
    }
  }
  Device.UI.posCount++;
}

void UISelect(byte _menu) {
  if (Device.UI.status == BUTTON_SELECT) {
    Device.UI.menu = _menu;
    Device.UI.pos = 0;
    saveEEPROM();
    lcd.clear();
  }
}

void UISelectTime(byte _menu) {
  if (Device.UI.status == BUTTON_SELECT) {
    Device.UI.menu = _menu;
    Device.UI.pos = 0;
    time.settime(0, Device.TEMP.min, Device.TEMP.hour, 1, 1, 1, 1);
    lcd.clear();
  }
}

void UIClear(char *title, char *text) {
  if (Device.UI.posCount == Device.UI.pos) {
    lcd.print(title);
    lcd.setCursor(0, 1);
    lcd.print(text);
    if (Device.UI.status == BUTTON_RIGHT || Device.UI.status == BUTTON_SELECT) {
      Device.UI.menu = 0;
      Device.UI.pos = 0;
      clearEEPROM();
```

```
            lcd.clear();
           }
          }
          Device.UI.posCount++;
         }

         void UItime() {

         }

         void sRELE() {
          pinMode(PIN_RELE_N1, OUTPUT); // Подача воды
          digitalWrite(PIN_RELE_N1, LOW);
          pinMode(PIN_RELE_N2, OUTPUT); // Полив
          digitalWrite(PIN_RELE_N2, LOW);
          pinMode(PIN_RELE_N3, OUTPUT); // Теплые полы ( подогрев почвы )
          digitalWrite(PIN_RELE_N3, LOW);
          pinMode(PIN_RELE_N4, OUTPUT); // Кондиционер
          digitalWrite(PIN_RELE_N4, LOW);
         }

         void RELE() {

         }


         void sRTC() {
          time.begin();
         }

         void RTC() {
          if (Device.UI.menu != 0) return;
          int Watering[4], RTC;

          RTC = (( time.Hours + 1) * 100) + time.minutes;

          Watering[0] = (( Device.Setting.Watering[0].start.hour + 1) * 100) +
Device.Setting.Watering[0].end.min;

          Watering[1] = (( Device.Setting.Watering[1].start.hour + 1) * 100) +
Device.Setting.Watering[1].end.min;

          Watering[2] = (( Device.Setting.Watering[2].start.hour + 1) * 100) +
Device.Setting.Watering[2].end.min;

          Watering[3] = (( Device.Setting.Watering[3].start.hour + 1) * 100) +
Device.Setting.Watering[3].end.min;

          if (Watering[0] != Watering[1]) {
           if (RTC == Watering[0] && Device.TEMP.Watter[0] == 0) {
```

```
      Serial.println("Полив начался");
      Device.TEMP.Watter[0] = 1;
     }
     if (RTC == Watering[1] && Device.TEMP.Watter[0] == 1) {
      // Полив закончился
      Serial.println("Полив закончился");
      Device.TEMP.Watter[0] = 0;
     }
    }

    if (Watering[2] != Watering[3]) {
     if (RTC == Watering[2] && Device.TEMP.Watter[1] == 0) {
      Serial.println("Полив начался (2)");
      Device.TEMP.Watter[1] = 1;
     }
     if (RTC == Watering[3] && Device.TEMP.Watter[1] == 1) {
      // Полив закончился
      Serial.println("Полив закончился (2)");
      Device.TEMP.Watter[1] = 0;
     }
    }



    //Serial.println( RTC );
   }


   void sTEMP() {
    ds.begin();
    Serial.println(ds.getDeviceCount());
    //ds.setResolution(sensor1, 12);
    //ds.setResolution(sensor2, 12);
    Device.TEMP.H = 0;
   }

   void TEMP() {
    if (Device.UI.menu != 0) return;
    ds.requestTemperatures();
    Device.TEMP.t = int(ds.getTempC(sensor1));
    Device.TEMP.t1 = int(ds.getTempC(sensor2));
    if ( Device.TEMP.t > 100 ) { Device.TEMP.t = 100; }
    if ( Device.TEMP.t < 0 )   { Device.TEMP.t = 0;   }
    if ( Device.TEMP.t1 > 100 ) { Device.TEMP.t1 = 100; }
    if ( Device.TEMP.t1 < 0   ) { Device.TEMP.t1 = 0;   }

    if (Device.Setting.SoilTemperature.start < Device.Setting.SoilTemperature.end) {
     if ( Device.TEMP.t1 <= Device.Setting.SoilTemperature.start && Device.TEMP.H ==
0) {
      digitalWrite(PIN_RELE_N3, LOW);
```

```
            //Serial.println("PIN_RELE_N3 -> HIGH");
            Device.TEMP.H = 1;
           }
           if ( Device.TEMP.t1 >= Device.Setting.SoilTemperature.end && Device.TEMP.H ==
1) {
            digitalWrite(PIN_RELE_N3, HIGH);
            //Serial.println("PIN_RELE_N3 -> LOW");
            Device.TEMP.H = 0;
           }
          }else{
           // OFF
           if (Device.TEMP.H == 1) { // Защита
            digitalWrite(PIN_RELE_N3, HIGH); // LOW
            Device.TEMP.H = 0;
           }
          }

         }
```