СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире технологические инновации играют ключевую роль в различных отраслях, включая сельское хозяйство. С развитием технологий, таких как беспроводная связь и системы автоматизации, возможности сельскохозяйственного производства значительно расширились. Одним из важных направлений в этой области является использование программно-аппаратных комплексов для управления теплицами. Дипломный проект посвящен разработке комплекса с использованием беспроводной технологии для управления функциональными устройствами в теплице.

Сельское хозяйство является важным сектором экономики многих стран, и эффективность его функционирования напрямую влияет на продовольственную безопасность и благосостояние населения. В этом контексте важно обеспечить оптимальные условия для роста и развития растений в теплицах, чтобы повысить урожайность и качество продукции. Традиционные методы управления зачастую ограничены в своих возможностях и требуют значительных затрат времени и ресурсов. Поэтому разработка программно-аппаратного комплекса для автоматизации процессов управления теплицами является актуальной задачей.

В данном дипломном проекте программно-аппаратный комплекс будет представлен в виде платы микроконтроллера, интегрированной с набором датчиков для сбора данных о микроклимате, с блоком управления положением окон, дверей и блоком управления подачи воды. Прошивку контроллера планируется запрограммировать для автоматизации процессов управления теплицей, включая регулирование температуры, влажности и других параметров окружающей среды. Этот подход обеспечит надежную и гибкую систему управления, способную адаптироваться к различным условиям и потребностям растений.

Проветривание и полив являются двумя важными аспектами поддержания оптимальных условий в теплице. Недостаточное проветривание может привести к скоплению вредных газов или избыточной влажности, что способствует развитию болезней растений и появлению плесени. Недостаточное или избыточное поливание также может негативно сказаться на здоровье растений: сухость почвы приводит к остановке роста и засыханию растений, а избыточная влажность может вызвать гниение корней. Игнорирование этих аспектов может привести к снижению урожайности и качества продукции. В рамках дальнейшей разработки проекта будут выбраны соответствующие устройства для проветривания и полива, которые позволят эффективно поддерживать необходимые условия в теплице, а также будут интегрированы в систему управления для автоматизации этих процессов и предотвращения возможных негативных последствий.

Для обеспечения беспроводного управления и мониторинга теплицей в проекте будет предусмотрено использование технологии Wi-Fi. Для этого необходимо будет выбрать соответствующий Wi-Fi модуль, который обеспечит надежное соединение и передачу данных между теплицей и

удаленным управляющим устройством. Выбор модуля будет основан на его характеристиках, таких как дальность передачи, скорость передачи данных, а также совместимость с основной системой управления теплицей. Внедрение беспроводной технологии Wi-Fi позволит оперативно контролировать и регулировать микроклимат в теплице на расстоянии, что значительно повысит эффективность и удобство управления процессами выращивания растений.

Целью данного дипломного проекта является разработка и реализация программно-аппаратного комплекса для управления теплицей на основе беспроводной технологии Wi-Fi. Главной задачей является создание аппаратного блока системы, который позволит автоматизировать процессы контроля и регулирования микроклимата в теплице, обеспечивая оптимальные условия для роста и развития растений.

Для достижения поставленной цели необходимо решить следующие задачи:

– проектирование контроллера теплицы, представленного в виде платы микроконтроллера и набора датчиков для сбора данных;

– разработка программного обеспечения для обработки информации и обеспечения взаимодействия между компонентами;

– реализация удаленного мониторинга и управления теплицей через API;

– тестирование функционала беспроводного управления и совместимости с подключенными устройствами.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Обзор аналогов

Процесс разработки программно-аппаратного комплекса для управления теплицей на основе беспроводной технологии Wi-Fi начинается с обзора существующих систем. Оценка уже существующих решений на рынке поможет выявить их достоинства и недостатки, а также определить тенденции и потенциал для улучшения. В рамках данного обзора рассматриваются аналогичные программно-аппаратные комплексы, разработанные для управления микроклиматом в теплицах. Особое внимание уделяется функциональности, надежности, удобству использования и интеграции с беспроводными технологиями. Анализ существующих аналогов позволит сформулировать требования к разрабатываемому комплексу и выделить ключевые характеристики, которые необходимо учесть при его проектировании и реализации.

1.1.1 Контроллер теплицы TECH-UC1002

DM-TECH TECH-UC1002 представляет собой компактный контроллер, разработанный специально для автоматизации управления в теплицах [1]. Он обеспечивает полный контроль над важными параметрами, такими как температура, влажность, освещение и вентиляция, необходимыми для обеспечения оптимальных условий для роста растений. На рисунке 1.1 представлен внешний вид данного контроллера.

Рисунок 1.1 – Контроллер теплицы TECH-UC1002

С помощью TECH-UC1002 можно программно регулировать температуру и использовать функции термостатов и регуляторов. Кроме того, можно получать данные с датчиков на мобильный телефон через мессенджер и визуализировать их с помощью программного обеспечения, которое

позволяет создавать схему теплицы и отображать графики показаний датчиков. Модульная конструкция контроллера позволяет легко добавлять необходимые датчики и устройства. К TECH-UC1002 можно подключить до 20 датчиков и исполнительных устройств, что обеспечивает точное контролирование условий внутри теплицы. Контроллер также может автоматически управлять освещением в соответствии с расписанием или данными с датчиков, что обеспечивает оптимальные условия для роста растений.

Основные характеристики данного устройства [1] представлены в таблице 1.1.

Таблица 1.1 – Характеристики TECH-UC1002

Характеристика	Значение
Название бренда	EN&EUHP
Номер модели	TECH-UC1002
Происхождение	Россия
Совместимость	Android
Спецификации	Российская / европейская (EU)
Мощность	3 Вт
Реле	Relay 1 nc 1 A 220 В
Wi-Fi	2,4 ГГц
Степень защиты	IP65 защита от пыли и брызг
Стоимость	125 $

К достоинствам TECH-UC1002 можно отнести:

1. Компактный размер и модульная конструкция, обеспечивающие легкость интеграции с различными устройствами и датчиками.
2. Возможность полного контроля над температурой, влажностью, освещением и вентиляцией в теплице, что способствует созданию оптимальных условий для роста растений.
3. Программируемая регулировка температуры и использование функций термостатов и регуляторов, что обеспечивает гибкость в управлении климатом.
4. Возможность получения данных с датчиков на мобильный телефон через мессенджер и их визуализация с помощью приложения, что обеспечивает удобство мониторинга.
5. Автоматическое управление освещением в соответствии с расписанием или данными с датчиков, что позволяет поддерживать оптимальные условия для роста растений.

К недостаткам TECH-UC1002 можно отнести:

1. Необходимость дополнительных затрат на приобретение и подключение датчиков и устройств для полноценного функционирования.
2. Ограниченное количество подключаемых датчиков и исполнительных устройств (до 20), что может ограничить возможности

расширения функционала в случае больших теплиц или высоких требований к мониторингу и управлению.

3. Высокая стоимость устройства, составляющая 125 долларов, что может быть недоступно для пользователей с ограниченным бюджетом или нецелесообразно для небольших проектов.

1.1.2 Комплект автоматизации «УМНИЦА grow»

Комплект автоматизации под названием «УМНИЦА grow» разработан для обеспечения комплексного управления, регулирования и мониторинга условий выращивания растений через беспроводную локальную сеть [2]. Он также осуществляет сохранение данных на карте памяти и поддерживает заданный микроклимат в теплице или других местах выращивания культурных растений. Этот комплект может быть использован для автоматизации как новых, так и уже существующих тепличных сооружений, включая парники, оранжереи, биовегетарии, гроубоксы и другие подобные конструкции. На рисунке 1.2 представлен внешний вид данного контроллера.

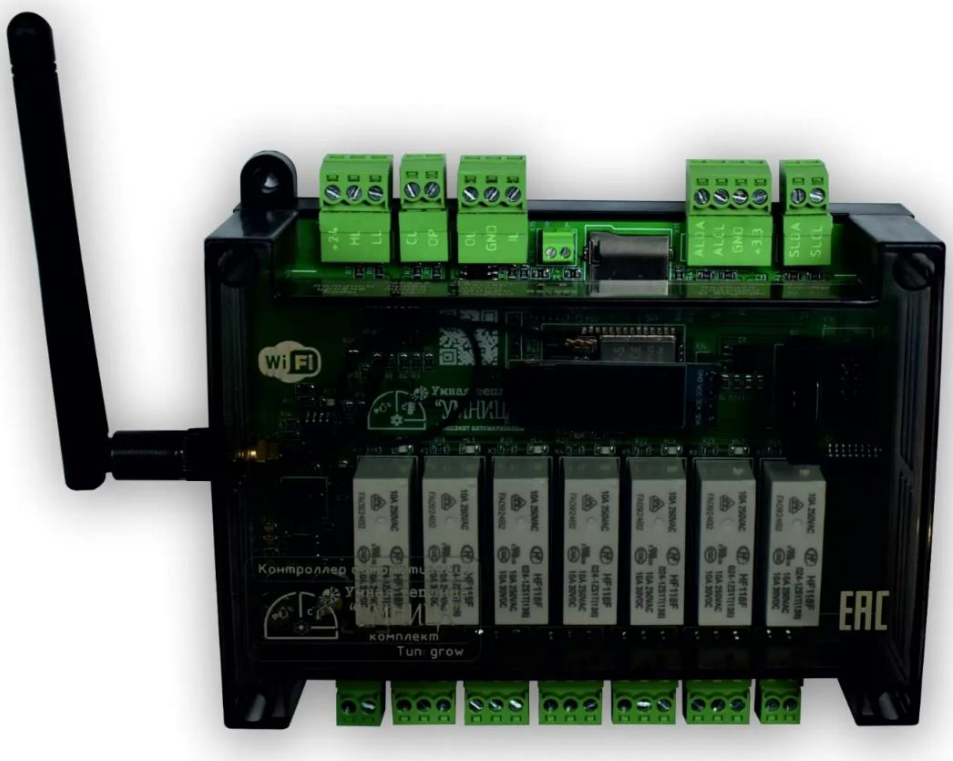


Рисунок 1.2 – Комплект автоматизации «УМНИЦА grow»

К достоинствам комплекта автоматизации «УМНИЦА grow» можно отнести:

– локализованное меню с русскоязычным интерфейсом и простым в использовании руководством пользователя;

– широкий выбор поддерживаемых датчиков для измерения температуры, влажности, освещенности и уровня воды;

– возможность управления различными исполнительными устройствами в теплице, включая клапаны полива, приводы проветривания и вентиляции, насосы для подпитки, а также устройства нагрева и освещения;

– интеграция с модулем отправки SMS-сообщений для удаленного информирования пользователя;

– возможность сохранения информации на SD-карту, включая записи с показаниями датчиков, отчеты о работе устройств и информацию об ошибках.

Основные характеристики данного устройства [2] представлены в таблице 1.2.

Таблица 1.2 – Характеристики комплекта автоматизации «УМНИЦА grow»

Характеристика	Значение
Габаритные размеры	140 × 160 × 43 мм
Материал	Пластик
Напряжение питания	24 В
Потребляемая мощность	1 Вт
Диагональ LCD дисплея	128 × 32
Типоразмер элемента питания	CR2032
Количество релейных выходов	7 шт.
Типоразмер SD карты	microSD
Беспроводная связь	Wi-Fi: 802.11 b / g / n
Проводная связь	интерфейс RS-485, протокол Modbus RTU
Диапазон рабочих температур	-25…+50 °C
Степень защиты	IP20
Стоимость	170 $

Особенности и функциональные возможности:

1. Подключение к домашней Wi-Fi сети или создание собственной точки доступа.

2. Управление через веб-браузер на ПК или мобильном устройстве через Wi-Fi.

3. Наличие дисплея на контроллере для отображения информации без подключения к Wi-Fi.

4. Запись системных сообщений и ошибок на MicroSD карту памяти.

5. Регистрация данных микроклимата и показаний датчиков на MicroSD карту каждый час с возможностью просмотра через Wi-Fi.

6. Возможность построения графиков.

7. Управление оборудованием по Wi-Fi в ручном режиме.

8. Отправка SMS уведомлений о системных сообщениях и показателях микроклимата.

9. Возможность подключения к SCADA программам или центральным пультам управления.

10. Гибкие настройки для создания оптимальных условий

микроклимата.

11. Система оповещения о низких и высоких значениях параметров (температура, влажность, уровень CO2, атмосферное давление).

12. Измерение температуры почвы в двух точках.

13. Определение уровня освещенности снаружи и внутри помещения.

14. Измерение концентрации CO2 и атмосферного давления.

15. Ручное управление открытием/закрытием проветривания или контроль через концевые выключатели.

16. Контроль уровня воды в резервуаре системы орошения.

17. Управление проветриванием и поливом по заданным параметрам микроклимата.

18. Управление системой полива по расписанию в различное время для четырех каналов.

19. Выбор продолжительности полива в зависимости от температуры почвы.

20. Управление универсальными таймерами для активации различных функций.

21. Контроль фитоосвещения по датчику освещенности с поддержкой заданного светового дня.

22. Управление вентилятором проветривания по различным сценариям микроклимата.

23. Регулировка скорости вентилятора.

24. Управление генератором CO2 по заданной концентрации.

25. Управление охладителем воздуха по заданным температурным параметрам.

26. Управление увлажнителем воздуха по заданным параметрам влажности.

К недостаткам данного комплекта можно отнести:

1. Отсутствие автоматического обновления программного обеспечения, что может привести к устареванию и невозможности использования новых функций и исправлений.

2. Ограниченное количество релейных выходов свободного назначения может ограничить возможности подключения дополнительного оборудования.

3. Ограничения по рабочей температуре (-25…+50 градусов Цельсия) могут создавать проблемы в экстремальных климатических условиях.

4. Требуется дополнительное оборудование или протокол для проводной связи по интерфейсу RS-485, что может повлечь за собой дополнительные расходы и сложности в настройке.

5. Необходимость установки на DIN-рейку или крепления на саморез может затруднить интеграцию в уже существующие системы или требовать дополнительных монтажных работ.

6. Стоимость устройства, равная 170 долларам, может быть препятствием для пользователей с ограниченным бюджетом или оказаться непрактичной для небольших проектов.

1.1.3 Контроллер теплицы «ТЕРРАФОРМ»

Контроллер ТЕРРАФОРМ представлен в виде электронного устройства, размещенного в пластиковом корпусе, способном крепиться на DIN рейку [3]. Корпус выполнен в светло-сером цвете из прочного и жаростойкого ABS пластика.

Устройство непрерывно собирает информацию от датчиков и, следуя программе, управляет выходными реле. Регулярно выводит данные с датчиков на ЖК-дисплей и передает их на сервер для взаимодействия с мобильными приложениями пользователей. Мониторинг показаний датчиков, состояние реле и их управление доступны через мобильное приложение и посредством SMS. Алгоритм управления реле настраивается через меню, доступное с помощью четырех кнопок на передней панели контроллера, а также частично через мобильное приложение. На рисунке 1.3 представлен внешний вид данного контроллера.

Рисунок 1.3 – Контроллер теплицы «ТЕРРАФОРМ»

Система включает в себя контроллер с встроенными реле и комплектом датчиков микроклимата. Пользователь устанавливает желаемые значения для различных параметров, таких как температура воздуха и почвы, влажность воздуха и почвы, уровень освещенности, содержание CO_2, pH, а также направление и скорость ветра. В зависимости от данных датчиков и заданных параметров, контроллер через реле управляет устройствами, регулирующими микроклимат в теплице, такими как насосы, клапаны, приводы форточек, вентиляторы, осветительные лампы, обогреватели и другое оборудование.

Мобильное приложение позволяет пользователю удаленно мониторить данные датчиков в теплице, управлять поливом, вентиляцией, подогревом и другими процессами, настраивать параметры датчиков и регулировать продолжительность и периодичность работы устройств. Также приложение отправляет аварийные оповещения в случае превышения установленных пределов. Часть функционала доступна через SMS-сообщения с мобильного телефона.

Контроллер ТЕРРАФОРМ используется в качестве основы для автоматизации систем в теплицах, зимних садах, грибных фермах,

аквариумах, террариумах, пчелиных ульях, а также в жилых, офисных и других помещениях.

Основные функции включают:

– автоматическое регулирование микроклимата на основе данных датчиков, времени и установленных таймеров;

– визуализацию текущих данных датчиков и навигацию по меню на экране контроллера;

– дистанционное управление данными датчиками, состоянием реле и управляемыми устройствами через мобильное приложение для Android, iOS, компьютера или посредством SMS;

– автоматическое оповещение о выходе показаний датчиков за установленные пределы через приложение и по SMS.

Основные характеристики данного устройства [4] представлены в таблице 1.3.

Таблица 1.3 – Характеристики контроллера теплицы «ТЕРРАФОРМ»

Характеристика	Значение
1	2
Габаритные размеры	212 × 90 × 58 мм
Масса без датчиков	0,5 кг
Число входов датчиков температуры-влажности воздуха	4
Число входов датчиков влажности почвы	6
Число входов универсальных датчиков температуры	16
Число входов датчиков освещенности	2
Число входов датчиков CO2	6
Число входов датчиков pH	1
Число входов датчиков минерализации EC	1
Число входов датчиков дождя	6
Число входов датчиков скорости ветра	1
Число входов датчиков направления ветра	1
Число выходов управления исполнительными устройствами	8 (до 16 с модулем расширения реле)
Нагрузочные характеристики реле	до 220 В, до 5 А (1 кВт) больше – через контактор
Напряжение изоляции реле	1,5 кВ
Диапазоны частот передатчика GSM/GPRS	850 / 900 / 1800 / 1900 МГц

Продолжение таблицы 1.3

1	2
Напряжение питания	220 В
Максимальная потребляемая мощность	10 Вт
Средства отображения	текстовый ЖК-дисплей, 2 строки × 16 символов
Рабочая температура	0…+50 °С
Стоимость	270 $

К недостаткам контроллера теплицы «ТЕРРАФОРМ» можно отнести:

1. Ограниченное количество входов для датчиков различных параметров микроклимата (например, только 4 входа для датчиков температуры-влажности воздуха).

2. Недостаток расширяемости: число входов универсальных датчиков температуры ограничено (16) и может оказаться недостаточным для некоторых проектов.

3. Отсутствие поддержки более широкого диапазона частот передатчика GSM/GPRS, что может ограничить возможности связи в некоторых регионах.

4. Ограниченные возможности отображения на текстовом ЖК-дисплее, что может затруднить мониторинг данных при большом объеме информации.

5. Ограниченная максимальная потребляемая мощность (10 Вт), что может привести к ограничениям в подключении дополнительного оборудования или расширении функциональности.

1.2 Состав разрабатываемого устройства

Как упоминалось ранее, разрабатываемый контроллер теплиц на основе беспроводной технологии представляет собой инновационное устройство, созданное для эффективного управления микроклиматом в тепличных условиях. Это устройство не только контролирует и регулирует параметры окружающей среды, но и обеспечивает удаленный мониторинг и управление с помощью беспроводных технологий связи.

Для выполнения данных функций в состав такого контроллера входят:

1. Плата микроконтроллер, с помощью которого выполняется обработка данных от датчиков микроклимата, управление исполнительными устройствами и взаимодействие с другими компонентами системы.

3. Модуль беспроводной связи, обеспечивающий передачу данных между контроллером и мобильным приложением, сервером или другими устройствами удаленного управления.

4. Датчики микроклимата, включающие в себя датчики температуры, влажности воздуха и почвы, освещенности, дождя, уровня воды.

5. Реле управления, предназначенные для управления различными исполнительными устройствами в теплице, такими как насосы, клапаны, приводы форточек, лампы освещения.

6. Блоки питания или иные источники питания, обеспечивающие работу устройства в автономном режиме.

7. Другие устройства контроля и управления, которые могут включать в себя средства отображения информации (например, ЖК-дисплей).

1.3 Обзор плат микроконтроллеров

Управление системой контроля микроклимата в теплице и сбором данных осуществляется при помощи микроконтроллера. Микроконтроллер представляет собой микроэлектронное устройство, способное программно обрабатывать информацию, передавать данные другим устройствам для их обработки и управлять функциональными устройствами в соответствии с полученными данными. Рынок микроконтроллеров предлагает широкий выбор моделей от различных производителей, включая современные 32-битные, а также 16-битные и 8-битные варианты. Каждое семейство микроконтроллеров может включать различные модели с разной скоростью работы центрального процессора (ЦПУ) и объемом памяти.

Для проведения сравнительного анализа была выбрана плата микроконтроллера Nano V3.0 (CH340) [5] на микроконтроллере ATmega328P и ее аналоги. Результаты данного сравнения представлены в таблице 1.4.

Таблица 1.4 – Сравнение плат микроконтроллеров

Характеристика	Nano V3.0 (CH340)	Arduino Pro Mini	Arduino Uno Rev3
1	2	3	4
Микроконтроллер	ATmega328P	ATmega328P	ATmega328P
Тактовая частота	16 МГц	16 МГц	16 МГц
Флеш-память	32 КБ (2 КБ отведено под загрузчик)	32 КБ (2 КБ отведено под загрузчик)	32 КБ (0,5 КБ отведено под загрузчик)
ОЗУ-память	2 КБ	2 КБ	2 КБ
EEPROM-память	1 КБ	1 КБ	1 КБ
Рабочее напряжение	5 В	5 В	5 В
Входное напряжение, VIN	7 – 12 В	5 – 12 В	7 – 12 В
Входное напряжение максимальное, VIN	6 – 20 В	3,35 – 20 В	6 – 20 В
Цифровые входы/выходы	19	14	14
Выходы ШИМ	6	6	6

Продолжение таблицы 1.4

1	2	3	4
Аналоговые входы, АЦП	8	6	6
Максимальный постоянный ток входа/выхода	40 мА	40 мА	20 мА
Максимальный постоянный ток выхода 3.3В	30 мА	50 мА	50 мА
Размеры	45 × 18 мм	33 × 18 мм	68,6 × 53,4 мм
Вес модуля	6 г	5 г	25 г

Модуль Arduino Nano является весьма популярным среди широкого круга разработчиков, как начинающих, так и опытных, которые работают с Arduino-совместимыми платами. Он включает в себя микроконтроллер ATmega328P, который работает при стандартном напряжении 5 вольт, а также имеет кварцевый резонатор с частотой 16 мегагерц.

На рисунке 1.4 представлен внешний вид данной платы [5].

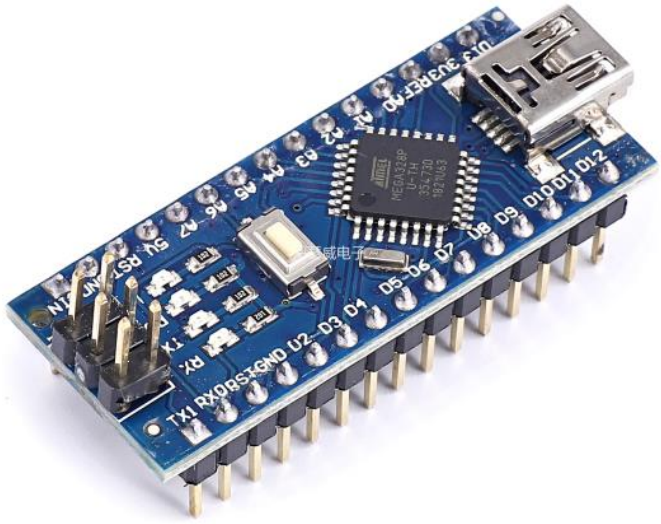


Рисунок 1.4 – Микроконтроллер Nano V3.0

Arduino Nano обладает почти всеми возможностями своего старшего аналога – Arduino Uno Rev3, однако он компактен и имеет 30 выводов. Размеры существенно уменьшены благодаря двухстороннему монтажу компонентов.

В комплекте идут разъемы-ножки с шириной шага 2,54 мм, которые не припаяны, что позволяет пользователю самостоятельно устанавливать их или использовать сторонние коннекторы. Arduino Nano предоставляет возможность подключения дополнительных модулей через различные интерфейсы, такие как датчики, сенсоры, экраны и моторы.

На рисунке 1.5 представлена схема распиновки данной платы [6].

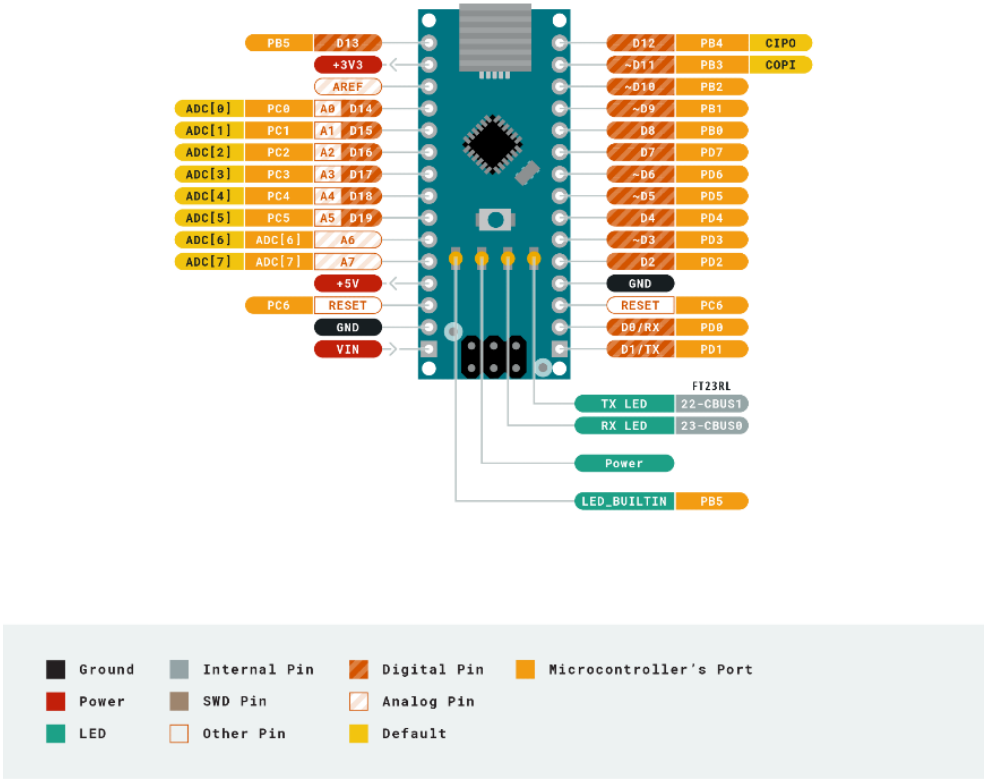


Рисунок 1.5 – Схема распиновки микроконтроллера Nano V3.0

Недостатком является отсутствие самовосстанавливающегося предохранителя, который отключает питание от USB-порта при перегрузке. Однако большинство современных компьютеров имеют встроенную защиту USB-порта, обеспечивая безопасность при использовании.

Благодаря своим компактным размерам, Arduino Nano идеально подходит для создания небольших проектов, легко помещается в корпусы и удобен в использовании.

Arduino Pro Mini [7] может получать питание либо через кабель FTDI или коммутационную плату, подключаемую к шестиконтактному разъему, либо через регулируемое напряжение питания 3,3 В или 5 В (в зависимости от модели), подаваемое на вывод Vcc. На плате имеется встроенный регулятор напряжения, который позволяет принимать напряжение до 12 В постоянного тока. При подключении нерегулируемого питания на плату необходимо убедиться, что контакт «RAW» используется для этой цели, а не контакт VCC.

Контакты питания следующие:

1. RAW: для подачи необработанного напряжения на плату.
2. VCC: регулируемое напряжение питания 3,3 или 5 В.
3. Земля: заземляющие штыри.

Микроконтроллер ATmega328P оснащен 32 КБ флэш-памяти для хранения кода (из которых 0,5 КБ занимает загрузчик), 2 КБ SRAM и 1 КБ EEPROM, которые могут быть использованы для чтения и записи с помощью библиотеки EEPROM.

На рисунке 1.6 представлен внешний вид данной платы [7].

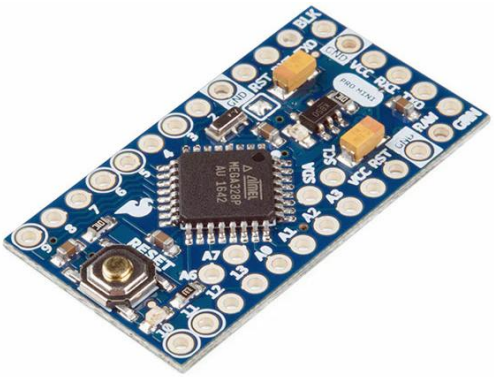


Рисунок 1.6 – Микроконтроллер Arduino Pro Mini

Каждый из 14 цифровых контактов Pro Mini может быть настроен на ввод или вывод работая при напряжении 3,3 или 5 вольт (в зависимости от модели). Каждый вывод может выдерживать или поставлять до 40 мА тока и имеет внутренний подтягивающий резистор на 20-50 кОм (по умолчанию отключен). Некоторые выводы имеют специализированные функции:

1. Последовательный порт: 0 (RX) и 1 (TX).
2. Внешние прерывания: 2 и 3.
3. ШИМ: 3, 5, 6, 9, 10 и 11.
4. SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).
5. Светодиод: 13.

Arduino Pro Mini также обладает 8 аналоговыми входами, каждый из которых предоставляет разрешение 10 бит (1024 различных значений). Некоторые выводы также имеют специализированные функции, такие как поддержка связи I2C (SDA и SCL).

На рисунке 1.7 представлена схема распиновки данной платы [8].

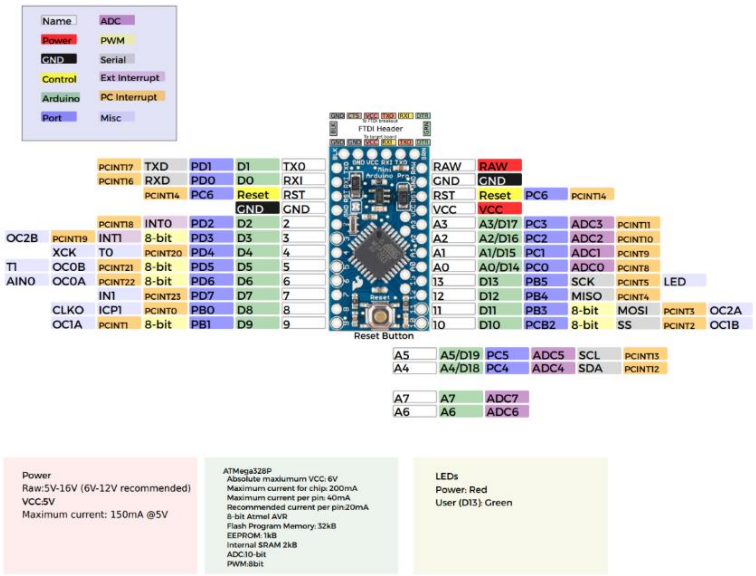


Рисунок 1.7 – Схема распиновки микроконтроллера Arduino Pro Mini

Arduino Pro Mini имеет возможности для связи с компьютером или другими устройствами через последовательную связь UART TTL, а также поддерживает интерфейсы I2C (TWI) и SPI.

Программирование контроллера Arduino Uno Rev3 [9] осуществляется из интегрированной среды разработки Arduino (IDE) с использованием резидентного загрузчика по протоколу STK500, что исключает необходимость в аппаратном программаторе. Контроллер также может быть запрограммирован через разъем для внутрисхемного программатора ICSP, обходя загрузчик. Кроме того, исходный код программы-загрузчика доступен для свободного использования.

Одно из отличий Arduino UNO Rev3 от предыдущих версий заключается в использовании микроконтроллера ATmega16U2 вместо моста USB-UART FTDI для подключения к компьютеру. Это позволяет плате автоматически выбирать источник питания – USB порт или внешний источник. Внешний источник питания может быть сетевым адаптером или батареей, подключаемыми к соответствующим разъемам.

На рисунке 1.8 представлен внешний вид данной платы [9].

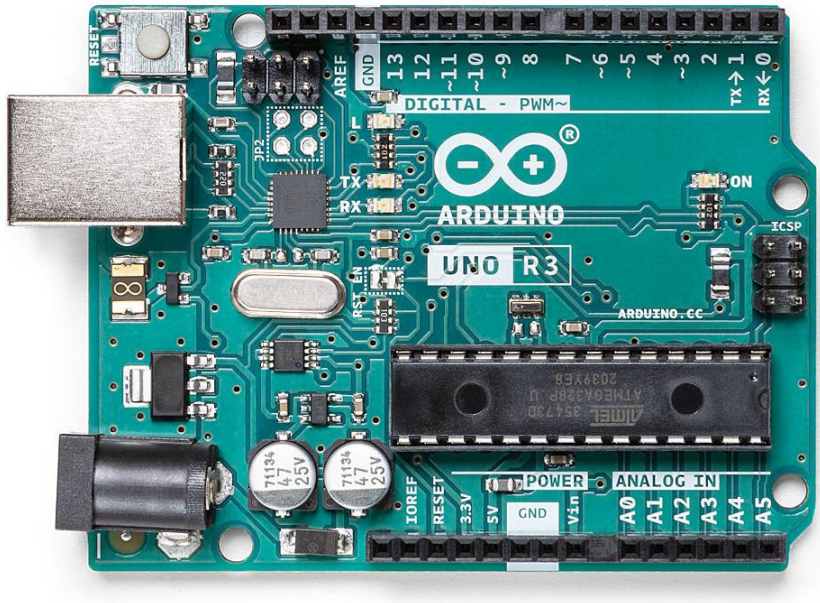


Рисунок 1.8 – Микроконтроллер Arduino UNO Rev3

Рекомендуемый диапазон напряжения питания для платы Arduino UNO составляет 7-12 В, хотя допустимое напряжение варьируется от 6 до 20 В. Каждый из 14 цифровых выводов может быть использован как вход или выход с уровнем напряжения 5 В, с максимальным током на выводе до 40 мА. Для удобства программирования, внутренний подтягивающий резистор может быть программно отключен.

Некоторые выводы Arduino UNO выполняют специализированные функции, такие как последовательный интерфейс, внешние прерывания, ШИМ и интерфейс SPI. Также на плате присутствуют аналоговые входы и дополнительные выводы для AREF и RESET.

Специализированные функции, доступные на различных выводах:

1. Serial: 0 (RX) и 1 (TX). Используются для приема (RX) и передачи (TX) серийных данных TTL. Эти выводы подключены к соответствующим выводам микросхемы ATmega8U2 USB-to-TTL Serial.

2. Внешние прерывания: 2 и 3. Эти выводы могут быть настроены на генерацию прерывания при низком уровне, нарастающем или спадающем фронте или изменении уровня.

3. ШИМ: 3, 5, 6, 9, 10 и 11. Предоставляют 8-битный ШИМ-сигнал с помощью функции `analogWrite()`.

4. SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Эти выводы поддерживают обмен данными по шине SPI с использованием библиотеки SPI.

5. LED: 13. Есть встроенный светодиод, управляемый цифровым выводом 13. Когда вывод установлен в HIGH, светодиод включен, когда вывод установлен в LOW, светодиод выключен.

6. TWI: выводы A4 или SDA и A5 или SCL. Поддерживают обмен данными по протоколу TWI с использованием библиотеки Wire.

Для обмена данными с компьютером или другими микроконтроллерами, Arduino UNO обладает коммуникационными интерфейсами UART, I2C и SPI. Аппаратная функция сброса позволяет автоматически инициировать сброс микроконтроллера при подключении к компьютеру, хотя такая функция может быть отключена при необходимости.

На рисунке 1.9 представлена схема распиновки данной платы [10].

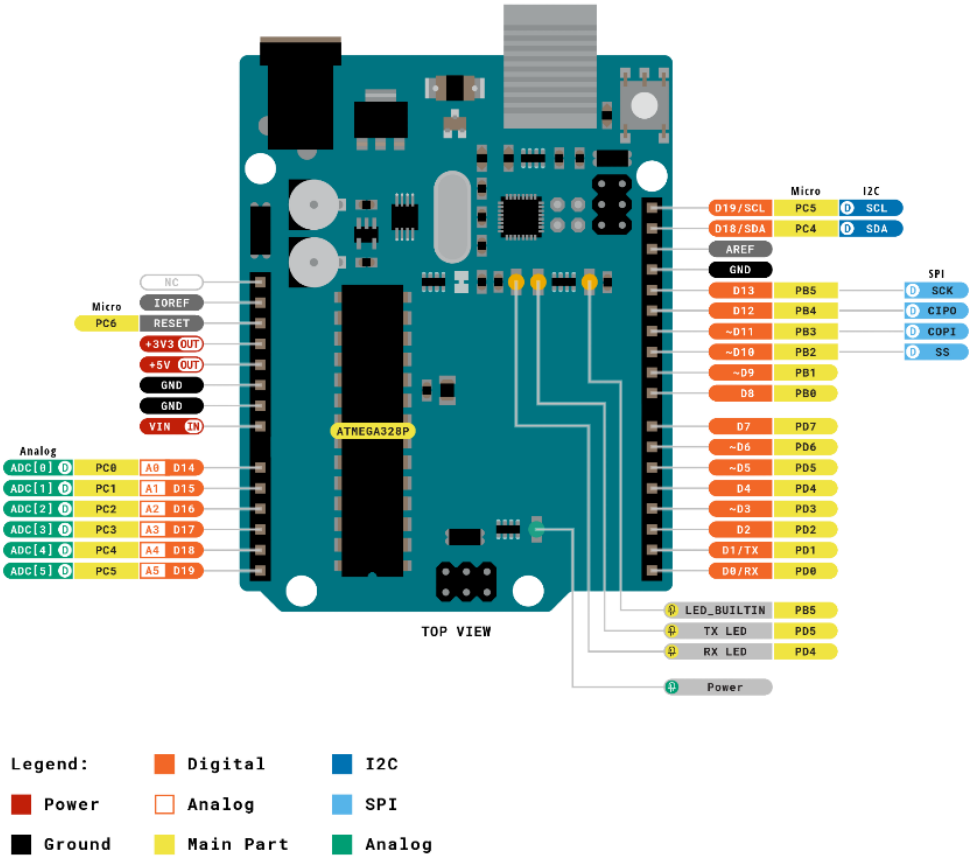


Рисунок 1.9 – Схема распиновки микроконтроллера Arduino UNO Rev3

1.3.1 Подведение итогов

Учитывая вышеуказанные факторы, был выбран Nano V3.0 с чипом CH340. Данный микроконтроллер был выбран из-за своего компактного размера, доступности компонентов и низкой цены. Эти факторы делают его идеальным выбором, обеспечивая удобство в использовании и экономичность без ущерба функциональности.

1.4 Обзор сред проектирования

Среды проектирования печатных плат (PCB) – это программные инструменты, предназначенные для создания электрических схем, размещения компонентов и трассировки маршрутов на печатной плате. Они позволяют инженерам и дизайнерам эффективно разрабатывать и визуализировать сложные электронные устройства перед тем, как приступить к физическому производству.

Эти среды играют важную роль в проектировании PCB, предоставляя широкий набор инструментов и функций, таких как редакторы схем, автоматическая трассировка, анализ сигналов, проверка правил проектирования и многое другое. Они помогают ускорить процесс разработки, сократить затраты на производство и улучшить качество конечного продукта.

Выбор подходящей среды проектирования PCB зависит от ряда факторов, включая уровень сложности проекта, требования к функциональности, бюджет, опыт пользователя и предпочтения. Некоторые из основных критериев выбора включают в себя:

1. Функциональность. Необходимо убедиться, что выбранная среда проектирования предоставляет необходимые инструменты для реализации конкретных требований проекта.
2. Легкость использования. Важно, чтобы интерфейс программы был интуитивно понятным и удобным для работы, особенно для новичков.
3. Совместимость. Программное обеспечение должно быть совместимо с используемым оборудованием и другими программными средствами.
4. Цена. Стоимость лицензии и дополнительных услуг должна быть доступной и соответствовать бюджету проекта.

1.4.1 KiCad

KiCad – это бесплатный и открытый программный комплекс автоматизации проектирования электроники (EDA). В его состав входят редактор схем, симулятор интегральных схем, разметка печатных плат (PCB), трехмерная визуализация и экспорт данных в различные форматы. KiCad также включает в себя библиотеку компонентов высокого качества, включающую тысячи символов, корпусов и трехмерных моделей. KiCad имеет минимальные системные требования и работает под управлением Linux, Windows и macOS.

На рисунке 1.10 представлен внешний вид интерфейса программы KiCad [11].

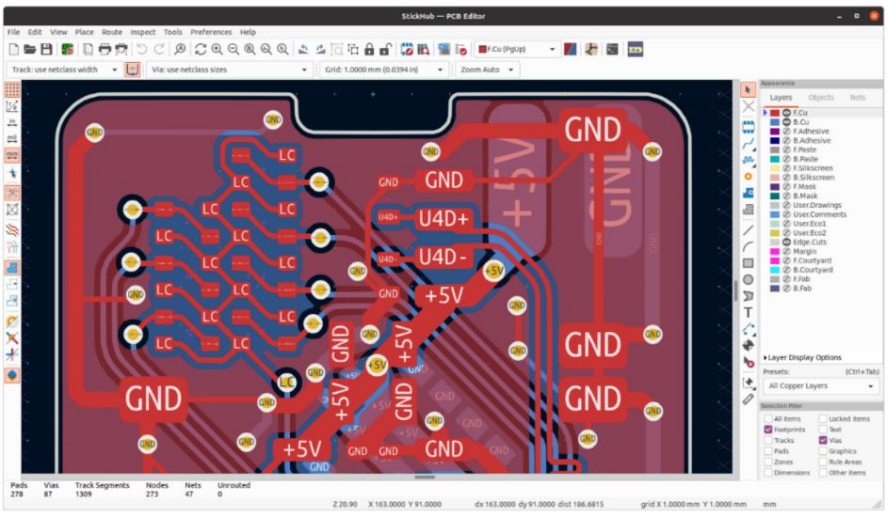


Рисунок 1.10 – Пример интерфейса KiCad

KiCad состоит из нескольких различных компонентов программного обеспечения, часть из которых интегрированы вместе для облегчения рабочего процесса проектирования печатных плат, а часть является автономными. В ранних версиях KiCad было очень мало взаимодействия между компонентами программного обеспечения. Например, редактор схем (исторически называемый Eeschema) и редактор печатных плат (исторически называемый PcbNew) были отдельными приложениями, которые не имели прямой связи между собой, и для создания печатной платы на основе схемы пользователи должны были создать файл списка связей в Eeschema, а затем прочитать этот файл списка в PcbNew. В современных версиях KiCad редактор схем и редактор печатных плат интегрированы в менеджер проектов KiCad, и использование файлов списка связей больше не требуется. Многие учебники по-прежнему описывают старый рабочий процесс KiCad с отдельными приложениями и файлами списка связей, поэтому обязательно проверяйте используемую версию при просмотре учебных пособий и другой документации.

Основные компоненты KiCad [12] обычно запускаются с помощью кнопок запуска в окне менеджера проектов KiCad. Данные компоненты представлены в таблице 1.5.

Таблица 1.5 – Компоненты KiCad

Название компонента	Описание
1	2
Schematic Editor	Создание и редактирование схем; моделирование схем с помощью SPICE; создание файлов списка материалов (BOM)

Продолжение таблицы 1.5

1	2
Symbol Editor	Создание и редактирование символов схем и управление библиотеками символов
PCB Editor	Создание и редактирование печатных плат; экспорт 2D и 3D файлов; создание файлов выходных данных для производства
Footprint Editor	Создание и редактирование компонентов печатной платы и управление библиотеками компонентов
GerbView	Просмотрщик файлов Gerber и сверловок
Bitmap2Component	Преобразование растровых изображений в символы или компоненты печатной платы
PCB Calculator	Калькулятор для компонентов, ширины дорожек, электрического промежутка и цветовых кодов.
Page Layout Editor	Создание и редактирование файлов рабочего листа

К достоинствам KiCad можно отнести:

1. Бесплатное и открытое программное обеспечение.
2. Имеет широкий функционал, включающий создание схем, проектирование печатных плат, симуляцию, 3D-отображение и экспорт в различные форматы.
3. Поддерживает кроссплатформенность, работает на операционных системах Linux, Windows и macOS.
4. Имеет обширную библиотеку символов, корпусов и моделей компонентов.
5. Активное сообщество пользователей, что облегчает поиск информации, обмен опытом и получение поддержки.
6. Регулярные обновления и улучшения, включая новые функции и исправление ошибок.

К недостаткам KiCad можно отнести:

1. Интерфейс может показаться не интуитивным и сложным для новичков из-за большого количества функций и настроек.
2. Некоторые функции, такие как 3D-отображение и симуляция, могут работать не так плавно и быстро.
3. Возможны проблемы с совместимостью с некоторыми форматами файлов, особенно при импорте и экспорте проектов из других CAD-систем.

4. Нет встроенной поддержки облачного хранилища для совместной работы над проектами.

1.4.2 CometCAD

Комплексная среда автоматизированного проектирования, предназначенная для создания принципиальных схем и разводки печатных плат.

Программа CometCAD [13] обладает ограниченным набором инструментов и предназначена в основном для начинающих разработчиков печатных плат. Ее функционал включает в себя модуль «Circuit Editor» для создания многостраничных схем, модуль «Layout Editor» для разработки макетов плат с возможностью ручной трассировки и модуль «Symbol Editor» для создания или редактирования радиокомпонентов. Переключение между этими редакторами осуществляется через меню. Однако в программе отсутствуют симулятор электрических схем, автоматический трассировщик и функции оптимизации размещения компонентов или размеров платы.

На рисунке 1.11 представлен внешний вид интерфейса программы CometCAD [14].

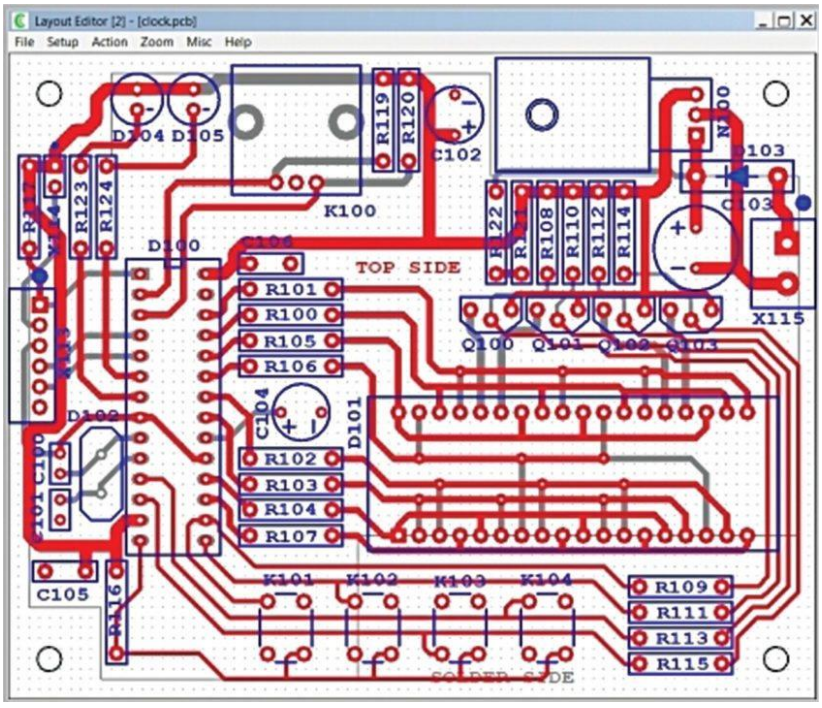


Рисунок 1.11 – Пример интерфейса CometCAD

Редактируемая библиотека содержит лишь наиболее распространенные компоненты. Файл с именем default.lib открывается автоматически при запуске программы и сохраняется при выходе из CometCAD. Редактор схем позволяет работать с листами количеством до 100 штук, каждый из которых может иметь максимальный размер 2 × 2 метра. При передаче принципиальной схемы в редактор печатных плат необходимо соблюдать определенные

требования, включая уникальные названия для всех элементов. Кроме того, в PCB-редактор передается список соединений (netlist), а также предоставляется возможность создания спецификации компонентов.

Модуль для разработки печатных плат (PCB) в CometCAD позволяет работать с двумя слоями на каждой стороне платы. Разрешение составляет один микрометр, а максимальные размеры печатных плат достигают 510 на 510 миллиметров, с возможностью размещения до 1000 посадочных мест. Присутствует функция проверки, которая выявляет ошибки и замечания, сохраняя их в файле errors.txt.

CometCAD обеспечивает широкий набор стандартных функций рисования, размещения и редактирования компонентов. Включая возможность вращения на 90 градусов, масштабирования, изменения размеров сетки и области работы, а также настройку фона, шрифтов текстовых надписей и толщины линий. Среди других функций программы – создание прямоугольных панелей для множества печатных плат и определение границ плат с помощью многоугольников. CometCAD позволяет экспортировать результаты работы в популярные форматы для сверлильных станков gerber или excellon. Кроме того, все выходные файлы могут быть распечатаны или сохранены.

Из-за ограниченных возможностей, данная программа вероятно не подойдет для профессиональных разработчиков. Также следует отметить наличие ошибок в программном коде и неудобное управление.

К достоинствам CometCAD можно отнести:

1. Предоставляет простую и понятную среду для создания принципиальных схем и разводки печатных плат.
2. Имеет интуитивно понятный интерфейс, что делает ее привлекательной для начинающих разработчиков.
3. Предоставляет широкий набор стандартных функций рисования, размещения и редактирования компонентов.
4. Поддерживает экспорт результатов работы в популярные форматы для сверлильных станков gerber или excellon.
5. Имеет модуль для разработки печатных плат, который позволяет работать с двумя слоями на каждой стороне платы.

К недостаткам CometCAD можно отнести:

1. Ограниченный набор инструментов и функций, что делает ее менее привлекательной для профессиональных разработчиков.
2. Наличие ошибок в программном коде и неудобное управление.
3. Отсутствие симулятора электрических схем, автоматического трассировщика и функций оптимизации размещения компонентов или размеров платы.
4. Редактируемая библиотека содержит лишь наиболее распространенные компоненты, что может быть недостаточно для некоторых проектов.
5. Требует соблюдения определенных требований при передаче принципиальной схемы в редактор печатных плат.

1.4.3 EasyEDA

EasyEDA [15] – удобное веб-приложение для автоматизации проектирования электроники, предназначенное для инженеров, преподавателей, студентов, мейкеров и энтузиастов. Программа EasyEDA обладает всеми необходимыми функциями, чтобы быстро и легко превратить идею в готовый проект.

На рисунке 1.12 представлен внешний вид интерфейса программы EasyEDA [15].

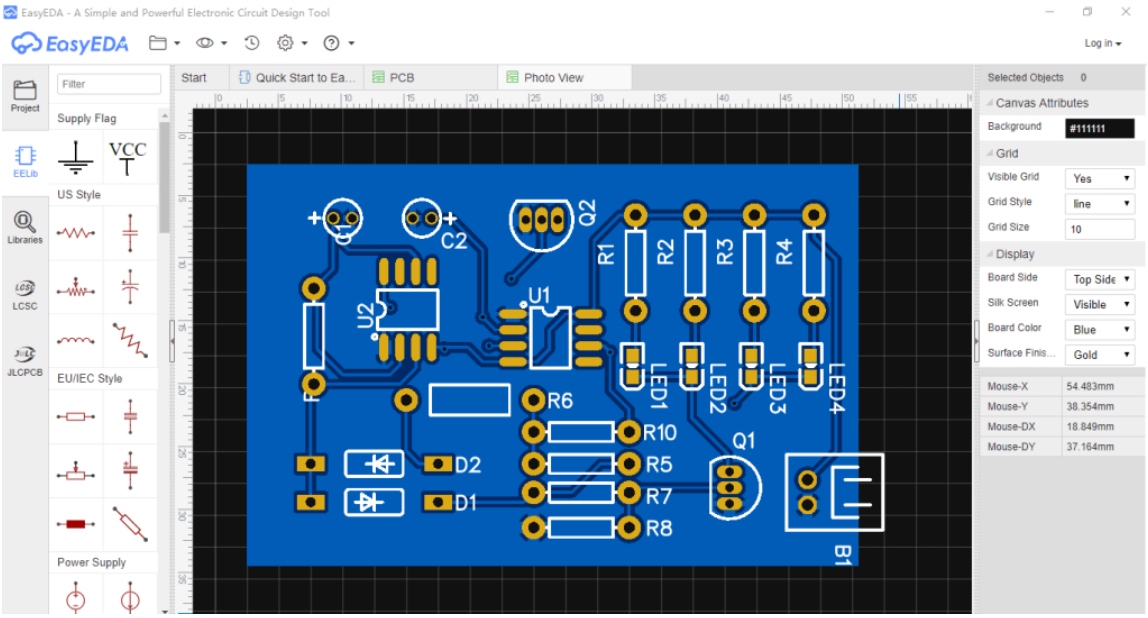


Рисунок 1.12 – Пример интерфейса EasyEDA

Суть EasyEDA заключается в использовании облачного сервиса, который осуществляет все вычисления на мощных серверах в Китае. Это означает, что скорость выполнения задач не зависит от характеристик компьютера, а определяется лишь скоростью интернет-соединения. Несмотря на наличие десктопного клиента, который может немного упростить и ускорить работу, все операции все еще выполняются через облачный сервис.

Платформа EasyEDA обеспечивает широкий спектр функциональности [16], включая редактирование принципиальных схем, разработку печатных плат, автоматическую трассировку печатных плат, просмотр печатных плат в 3D, создание файлов для производства (например, Gerber), возможность моделирования принципиальных электрических схем, экспорт в формат BOM (Bill of Materials) и многое другое.

К достоинствам EasyEDA можно отнести:

1. Простой и мощный редактор. EasyEDA предоставляет простые и удобные инструменты для создания схем и макетов печатных плат. Редактор доступен прямо в веб-браузере без необходимости установки дополнительного программного обеспечения.

2. Реальное время совместной работы. EasyEDA позволяет совместно

работать над проектом с другими пользователями в реальном времени, обеспечивая удобство командной работы.

3. Широкие возможности экспорта и импорта. EasyEDA поддерживает экспорт и импорт проектов в различные форматы, включая Gerber, Altium Designer, Eagle и KiCAD, что обеспечивает совместимость с другими популярными программами для проектирования электроники.

4. Богатая библиотека компонентов. EasyEDA предоставляет доступ к более чем миллиону общедоступных библиотек символов и компонентов, что значительно упрощает процесс проектирования.

К недостаткам EasyEDA можно отнести:

1. Ограниченные возможности бесплатной версии. Некоторые расширенные функции могут быть доступны только в платных версиях EasyEDA.

2. Ограниченный набор инструментов для анализа и симуляции. EasyEDA может быть не слишком подходящим выбором для профессиональных разработчиков, которым требуются расширенные возможности анализа и симуляции.

3. Необходимость подключения к интернету. EasyEDA требует постоянного подключения к интернету для работы, что может быть неудобно в некоторых ситуациях.

1.5 Обзор сред моделирования

В мире электронного проектирования среды моделирования и принципиальные схемы играют важную роль, обеспечивая инженерам инструменты для создания и анализа сложных электронных устройств.

Принципиальные схемы представляют собой графическое изображение электрической схемы устройства, которое отражает его функциональную структуру и взаимосвязь компонентов. Они служат основой для разработки и анализа конструкции, помогая инженерам понять работу системы, выявить потенциальные проблемы и оптимизировать ее производительность. Принципиальные схемы создаются с использованием специализированных программных инструментов и состоят из символов, представляющих электронные компоненты, и линий, обозначающих электрические соединения между ними. Важно отметить, что они не учитывают конкретное физическое расположение компонентов на печатной плате, а сконцентрированы на логике и функциональности устройства.

Среды моделирования, в свою очередь, предоставляют инженерам возможность создавать виртуальные прототипы и анализировать поведение электронных систем до их физической реализации. Они включают в себя такие инструменты, как редакторы схем, симуляторы электрических схем, автотрассировщики печатных плат и многое другое. С помощью сред моделирования инженеры могут проверить работоспособность и надежность своих конструкций, а также оптимизировать их производительность до начала производства.

1.5.1 AutoCAD Electrical

AutoCAD Electrical [17] объединяет в себе основные функции программного обеспечения AutoCAD, дополняя их уникальными инструментами, предназначенными для автоматизации процессов создания схем, компоновки чертежей, генерации отчетов и других задач. Приложение предоставляет возможность работать как над целыми проектами, так и над индивидуальными компонентами, такими как двигатели, клеммы, реле, провода, жгуты, кабели, а также программируемые логические контроллеры. AutoCAD Electrical поддерживает разнообразные типы проектов, включая принципиальные схемы, схемы автоматизации, чертежи компоновок, схемы соединений, монтажные планы и различные отчеты. Модуль «Диспетчер проектов» обеспечивает гармоничную работу рабочих групп на всех этапах проекта, обеспечивая доступ к единой цифровой модели.

Программа соответствует международным стандартам оформления чертежей и содержит обширные библиотеки компонентов и условных обозначений, включая более 2000 элементов электрических схем, соответствующих стандартам ГОСТ, IEC, JIS, JIC, GB, AUS. Пользователи также могут создавать собственные графические образы и добавлять их в библиотеку. В базах данных каталога содержится более 370 тысяч наименований изделий от известных производителей, а также их компоновочные образы и каталожные данные.

На рисунке 1.13 представлен внешний вид интерфейса программы EasyEDA [17].

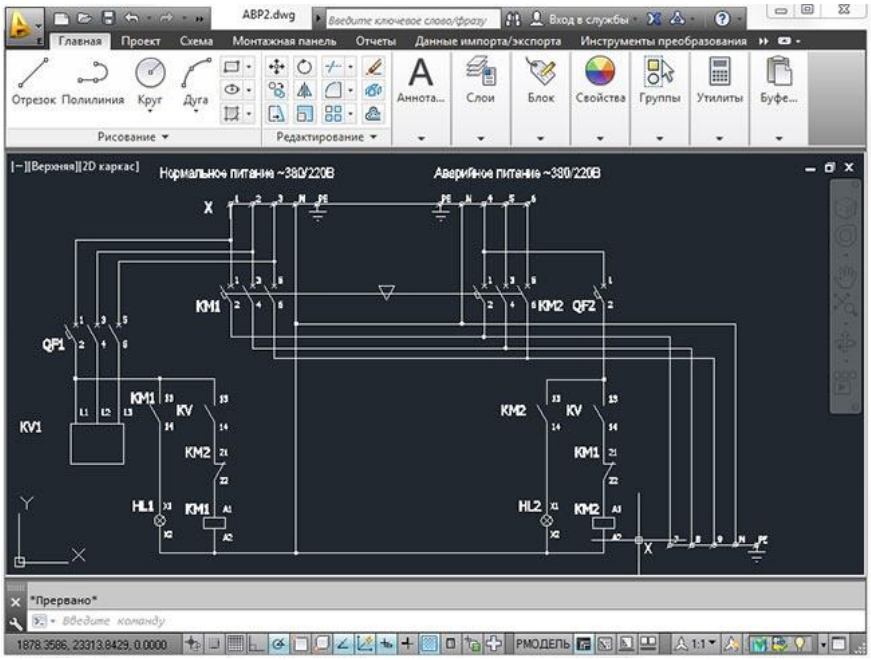


Рисунок 1.13 – Пример интерфейса AutoCAD Electrical

Каждому элементу в схеме автоматически присваивается свой уникальный идентификатор. Части компонентов с одинаковыми

идентификаторами, но на разных листах, рассматриваются программой как один объект. Любые изменения, внесенные в одну часть компонента, автоматически распространяются на все другие. Для проводов в проекте можно указывать различные характеристики, такие как цвет, марку, сечение, номера и функции жил кабелей и т.д. Для соединения цепей, находящихся на разных листах или частях листа, используются специальные ссылки. В программе AutoCAD Electrical имеются инструменты для работы с схемами, содержащими жгутовые соединения, контакторы и реле, а также программируемые логические контроллеры. Информация о клеммах проекта доступна в «Редакторе клеммных колодок».

AutoCAD Electrical непрерывно контролирует все операции в режиме реального времени и предупреждает об ошибках при необходимости. Программа отслеживает уникальные группы объектов для компонентов, дублирование идентификаторов, некорректные контакты, отсутствие или повторение номеров и др.

На основе данных отдельных чертежей или всего проекта генерируются различные отчеты, такие как таблицы соединений, списки компонентов и проводов, таблицы сигналов ПЛК и др. Пользователи могут создавать и настраивать пользовательские отчеты и сохранять их в различных форматах файлов. Для создания трехмерной модели изделия имеется возможность использования программы Autodesk Inventor.

AutoCAD Electrical является коммерческим программным обеспечением, стоимость лицензии которого составляет приблизительно 5000 долларов. Пользователи могут загрузить бесплатную 30-дневную демонстрационную версию программы, и установка происходит автоматически.

К достоинствам AutoCAD Electrical можно отнести:

1. Обширные возможности. AutoCAD Electrical предлагает широкий спектр инструментов для создания схем, компоновки чертежей и генерации отчетов, что обеспечивает полный цикл работы над проектами.

2. Международные стандарты. Программа соответствует международным стандартам оформления чертежей и содержит обширные библиотеки компонентов и условных обозначений.

3. Удобство работы. Автоматическое присвоение уникальных идентификаторов каждому элементу, а также возможность изменения данных элементов с автоматическим обновлением на всех листах проекта значительно упрощают процесс работы.

К недостаткам AutoCAD Electrical можно отнести:

1. Высокая стоимость. AutoCAD Electrical является коммерческим программным обеспечением с высокой стоимостью лицензии, что может быть недоступно для многих пользователей с ограниченным бюджетом.

2. Необходимость обучения. Использование всех возможностей программы требует определенного времени на обучение, что может быть недоступно для пользователей без достаточного опыта в работе с подобными программами.

1.5.2 LTspice

LTspice [18], также известный как SwitcherCAD, представляет собой программу для проведения компьютерного моделирования работы аналоговых и цифровых электрических цепей. Это универсальное средство проектирования и симуляции, включающее в себя графический схематический редактор и средства анализа результатов моделирования.

На рисунке 1.14 представлен внешний вид интерфейса программы LTspice [19].

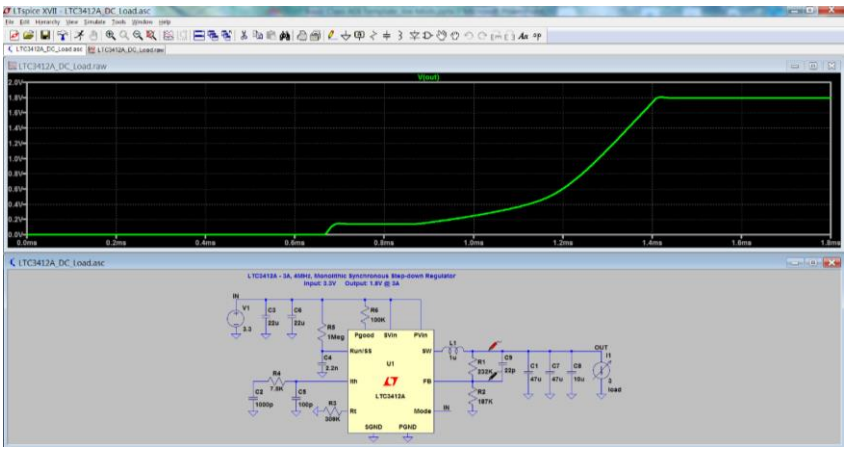


Рисунок 1.14 – Пример интерфейса LTspice

Особенности LTspice:

1. Интегрированный симулятор. LTspice позволяет создавать и моделировать электрические схемы, а также анализировать их поведение. Благодаря встроенному симулятору смешанного моделирования, пользователи могут быстро менять компоненты и параметры схем, а также исследовать различные варианты.
2. Библиотека компонентов. Программа включает в себя богатую библиотеку компонентов от компании Linear Technology Corporation, а также позволяет добавлять сторонние библиотеки и создавать собственные модели.
3. Функциональные возможности. LTspice позволяет проводить различные виды анализа, включая амплитудно-частотный анализ, анализ переходных процессов, спектральный анализ и анализ гармоник.
4. Гибкость и удобство использования. Программа обладает простым и понятным интерфейсом, что делает ее доступной для широкого круга пользователей. Кроме того, LTspice поддерживает работу с различными файловыми форматами, что обеспечивает совместимость с другими инструментами проектирования.

К достоинствам LTspice можно отнести:

1. Мощный и быстрый SPICE-симулятор, обеспечивающий точное моделирование аналоговых и цифровых электрических цепей.
2. Интегрированный графический схематический редактор, упрощающий процесс создания и редактирования схем.

3. Возможность проведения различных видов анализа схем, что позволяет оценить их работоспособность и производительность.

4. Богатая библиотека компонентов и моделей, включающая стандартные компоненты и элементы от Linear Technology Corporation.

К недостаткам LTspice можно отнести:

1. Ограниченный набор библиотек элементов, что может потребовать расширения пользовательских возможностей.

2. Не всегда интуитивный интерфейс, требующий времени для освоения и понимания всех функциональных возможностей.

3. Требовательность к ресурсам компьютера, особенно при работе с большими и сложными схемами.

1.5.3 EasyEDA

В онлайн сервисе EasyEDA [15] предоставлен полный комплекс инструментов для проектирования электронных устройств: от редактора электрических схем до spice-симулятора и редактора печатных плат.

EasyEDA идеально подходит для создания электроники различной сложности и предназначен для широкого круга пользователей – от инженеров до студентов и радиолюбителей.

На рисунке 1.15 представлен внешний вид интерфейса программы EasyEDA [15].

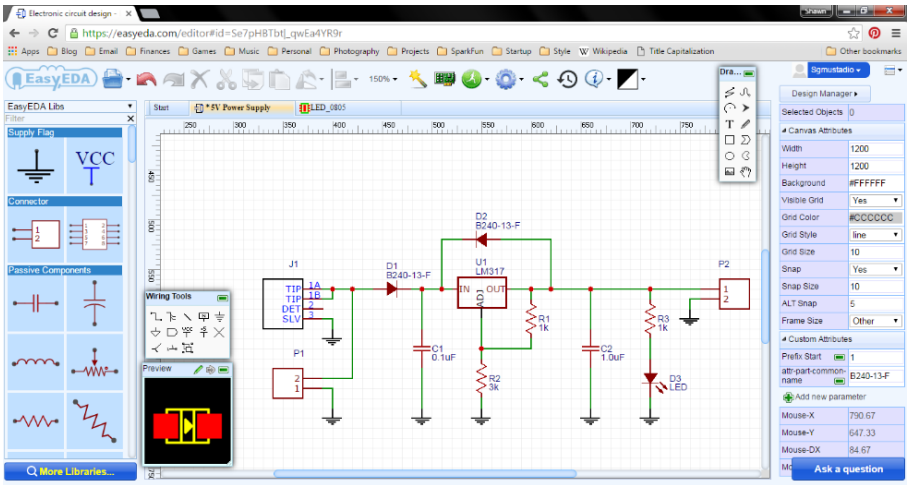


Рисунок 1.15 – Пример интерфейса EasyEDA

Этот веб-сервис позволяет импортировать файлы из популярных САПР, таких как LTSpice, Eagle, Kicad и Altium Designer. Редактор электрических схем обеспечивает возможность создания новых проектов с использованием разнообразных инструментов и библиотек. Здесь можно редактировать как отдельные компоненты, так и создавать иерархические схемы и SPICE-модели.

Spice-симулятор включает разнообразные анализы для аналоговых, цифровых и смешанных цепей, а его гибкие настройки позволяют получить

полный контроль над отображением результатов. Редактор печатных плат обеспечивает создание макетов на основе схем, с инструментами для размещения компонентов, прокладки дорожек и проверки правил дизайна. EasyEDA также поддерживает экспорт в различные форматы для дальнейшей обработки и производства печатных плат.

1.6 Обзор сред разработки

Область разработки программного обеспечения на сегодняшний день является чрезвычайно разнообразной и динамичной. Среды разработки (IDE) играют ключевую роль в процессе создания программного кода для широкого спектра приложений – от мобильных приложений и веб-сайтов до встраиваемых систем и микроконтроллеров. В данном обзоре будут рассмотрены некоторые из основных сред разработки, которые используются для написания программного кода для микроконтроллеров Arduino, популярной платформы для разработки электронных устройств.

1.6.1 Arduino IDE

Arduino IDE [20] – это среда разработки, предназначенная для создания и редактирования программного кода для плат, на базе таких микроконтроллеров как: ATmega328, ATmega32U4, ATmega2560, AT91SAM3X8E, SAMD21, ESP8266, Intel x86, ATtiny85, STM32 и других.

На рисунке 1.16 представлен внешний вид интерфейса программы Arduino IDE.

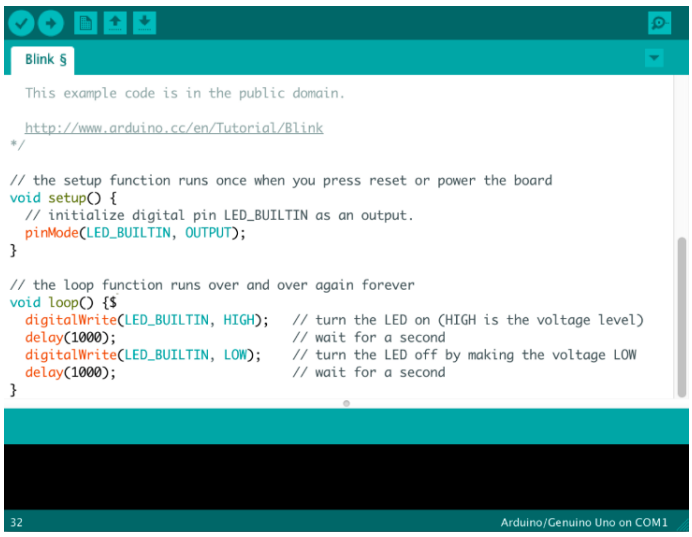


Рисунок 1.16 – Пример интерфейса Arduino IDE

Приложение Arduino IDE завоевало популярность среди многих пользователей благодаря своей простоте, удобству и богатому набору готовых библиотек и поддерживаемых микроконтроллеров, список которых постоянно растет.

С помощью Arduino IDE создают скетчи как для простых мигалок, таймеров и переключателей нагрузок, так и для систем умного дома, станков с ЧПУ и квадрокоптеров.

В программировании используется упрощенная версия языка C++, что обеспечивает легкость и простоту в освоении для начинающих программистов электронщиков.

1.6.2 PlatformIO

PlatformIO [21] – это интегрированная среда разработки (IDE) и экосистема для разработки встраиваемых систем, включая микроконтроллеры Arduino, ESP8266, ESP32, STM32 и многие другие. Эта платформа предоставляет разработчикам удобные инструменты для написания, отладки и загрузки кода на различные микроконтроллеры, а также управления зависимостями проекта, установкой библиотек и многое другое.

Одной из главных особенностей PlatformIO является его мультиплатформенность. Он поддерживает работу на различных операционных системах, включая Windows, macOS и Linux, что делает его доступным для широкого круга разработчиков. Кроме того, PlatformIO интегрируется с такими популярными средами разработки, как Visual Studio Code, Atom и Eclipse, что обеспечивает удобный и привычный интерфейс для работы.

На рисунке 1.17 представлен внешний вид интерфейса программы PlatformIO.

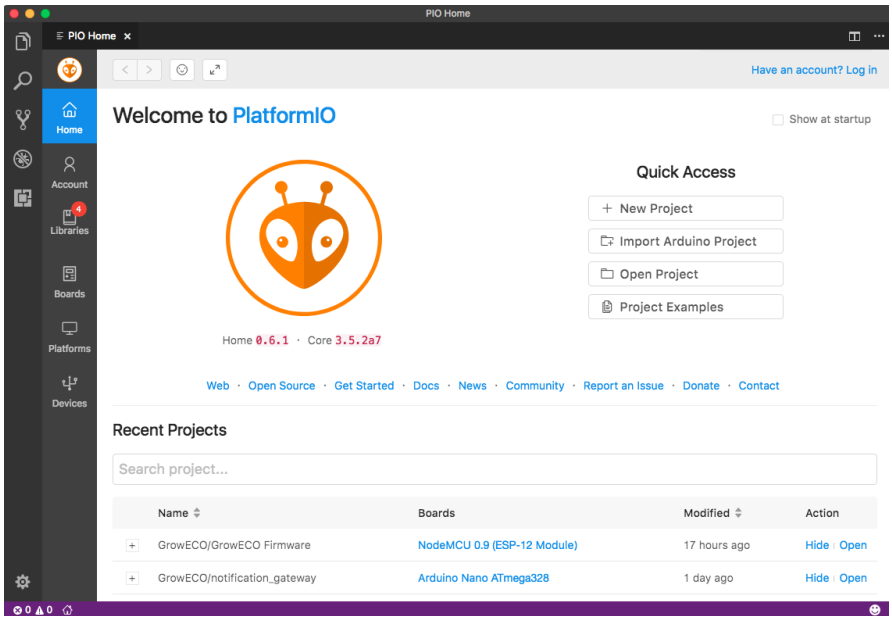


Рисунок 1.17 – Пример интерфейса PlatformIO

Среда разработки PlatformIO обладает богатым функционалом. Она предоставляет интуитивно понятный пользовательский интерфейс, который включает в себя редактор кода с подсветкой синтаксиса, автодополнением и

другими удобными функциями. Кроме того, PlatformIO поддерживает интегрированный менеджер библиотек, который позволяет легко управлять зависимостями проекта и устанавливать необходимые библиотеки из официального репозитория или из пользовательских источников.

Еще одним важным аспектом PlatformIO является его интеграция с системами контроля версий, такими как Git. Это позволяет разработчикам легко отслеживать изменения в своих проектах, создавать резервные копии и сотрудничать над кодом с другими участниками команды.

PlatformIO также обладает мощными возможностями по отладке кода. Он поддерживает различные отладочные интерфейсы и позволяет проводить отладку как на уровне аппаратуры, так и на уровне программного кода, что делает процесс разработки более эффективным и продуктивным.

Кроме того, PlatformIO предоставляет широкий набор инструментов для автоматизации процесса сборки, тестирования и развертывания проектов, что позволяет разработчикам сосредоточиться на написании качественного кода и создании инновационных устройств.

1.7 Подведение итогов

Исходя из представленной информации, в контексте дипломного проекта предполагается использовать:

– Nano V3.0 (CH340) в качестве платы микроконтроллера;

– язык программирования C для написания основного кода программы;

– платформу EasyEDA для моделирования схем и проектирования печатной платы;

– среду разработки Arduino IDE для написания и отладки кода.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Изучив предметную область данного дипломного проекта, были выявлены основные требования, которые необходимо учесть при его реализации. Для упрощения процесса разработки было решено разбить систему на отдельные структурные блоки.

2.1 Описание основных аппаратных блоков устройства

В результате разделения комплекса для управления теплицей были определены следующие аппаратные блоки:

– блок контроллера теплицы;
– блок беспроводной связи;
– блок управления контроллером;
– блок отображения информации;
– блок регулировки вентиляции;
– блок управления доступом в теплицу;
– блок реле;
– блок системы орошения;
– блок искусственного освещения;
– блок определения уровня воды;
– блок определения интенсивности освещения;
– блок определения влажности почвы;
– блок определения температуры и влажности воздуха;
– блок идентификации дождя;
– блок питания.

На чертеже ГУИР.400201.005 Э1 представлена структурная схема, наглядно демонстрирующая блоки и взаимосвязи между ними.

2.1.1 Блок контроллера теплицы

Блок контроллера теплицы – это основной блок разрабатываемого комплекса, отвечающий за управление всей системой, и выполняет следующий ряд задач:

1. Получение, обработка и передача данных, поступающих с блока беспроводной связи.
2. Получение и обработка данных.
3. Передача управляющих сигналов на блок реле.
4. Передача обработанных данных на блок отображения информации.
5. Передача управляющих сигналов на блок управления доступом в теплицу.
6. Передача управляющих сигналов на блок регулировки вентиляции.
7. Сбор и обработка данных, поступающих с блока определения влажности почвы.
8. Сбор и обработка данных, поступающих с блока определения

температуры и влажности воздуха.

9. Сбор и обработка данных, поступающих с блока идентификации дождя.

10. Сбор и обработка данных, поступающих с блока определения интенсивности освещенности.

11. Сбор и обработка данных, поступающих с блока определения уровня воды.

Таким образом, блок контроллера теплицы выполняет последовательный сбор данных с датчиков микроклимата, обрабатывает их и отправляет управляющие сигналы на конечные устройства в зависимости от полученных показаний.

2.1.2 Блок беспроводной связи

Блок беспроводной связи тесно связан с блоком контроллера теплицы, обеспечивая передачу данных и команд между ними. При поступлении запросов от клиентских устройств через интерфейс беспроводной связи, этот блок принимает команды и данные, а затем передает их на обработку блоку контроллера теплицы.

Задачей блока беспроводной связи является эффективная передача информации от клиентов к контроллеру теплицы и наоборот. Это позволяет пользователям отправлять команды на управление различными аспектами тепличной системы, такими как включение/выключение устройств, регулировка параметров микроклимата и другие операции, которые должны быть выполнены контроллером.

Таким образом, благодаря блоку беспроводной связи, пользователи могут взаимодействовать с системой управления теплицей удаленно, отправляя команды и получая обратную связь о состоянии системы, что делает процесс управления более гибким и удобным.

2.1.3 Блок управления контроллером

Блок управления контроллером теплицы является важным компонентом системы, тесно интегрированным с блоком контроллера теплицы. Этот модуль обеспечивает интерфейс для пользователя, позволяя ему взаимодействовать с устройством и управлять различными функциями с помощью меню.

Центральным элементом блока управления контроллером является энкодер, который представляет собой специальное устройство ввода, позволяющее пользователю прокручивать и выбирать пункты меню. При вращении энкодера пользователь может переключаться между различными опциями на экране, а нажатие на него позволяет подтверждать выбор или вводить данные.

Функциональность блока управления контроллером включает в себя:

1. Навигацию по меню: пользователь может перемещаться по различным пунктам меню, выбирая необходимые опции.

2. Выбор параметров: энкодер позволяет пользователю выбирать и настраивать различные параметры системы, такие как температурные пороги, режимы освещения и другие.

3. Подтверждение действий: нажатие на кнопку энкодера подтверждает выбор пользователя или вводит измененные параметры в систему.

4. Взаимодействие с дисплеем: блок управления контроллером обеспечивает визуальное отображение меню и выбранных опций на LCD дисплее, что делает процесс управления более интуитивно понятным.

Таким образом, блок управления контроллером предоставляет пользователю удобный и эффективный способ взаимодействия с системой управления теплицей, обеспечивая простоту использования и гибкость в настройке различных параметров.

2.1.4 Блок отображения информации

Блок отображения информации играет ключевую роль в системе управления теплицей, взаимодействуя с блоком контроллера теплицы и предоставляя пользователю информацию о состоянии системы. Этот блок ответственен за визуализацию данных, полученных от датчиков микроклимата, а также отображение состояния конечных устройств, таких как сервоприводы, водяную помпу и другие.

Функциональность блока отображения информации включает в себя:

1. Отображение показаний датчиков: благодаря связи с блоком контроллера теплицы, этот блок отображает текущие значения температуры, влажности воздуха и почвы, уровня воды и других параметров, измеряемых датчиками.

2. Представление состояния устройств: блок отображения информации также показывает пользователю состояние конечных устройств, таких как реле, электроприводы и сервоприводы, указывая, включены они или выключены, отображая текущий режим работы.

3. Визуализация меню: блок отображения информации предоставляет пользователю доступ к меню управления, позволяя просматривать и выбирать различные опции и параметры системы.

4. Интерактивность: благодаря LCD дисплею и кнопкам управления, блок отображения информации обеспечивает возможность взаимодействия с системой управления теплицей, позволяя пользователю просматривать данные и настраивать параметры непосредственно на устройстве.

Таким образом, блок отображения информации предоставляет пользователю удобный и интуитивно понятный интерфейс для мониторинга и управления всеми аспектами процесса выращивания растений.

2.1.5 Блок регулировки вентиляции

Блок регулировки вентиляции играет важную роль в обеспечении оптимальных условий в теплице, контролируя приток и отток воздуха. Этот

блок тесно взаимодействует с блоком контроллера теплицы и состоит из сервоприводов, которые отвечают за открытие и закрытие заслонок для вентиляции.

Функциональность блока регулировки вентиляции включает следующее:

1. Управление заслонками: благодаря сервоприводам, этот блок контролирует положение заслонок, регулируя приток и отток воздуха в теплице.
2. Автоматическое регулирование: блок регулировки вентиляции интегрирован в систему автоматического управления, чтобы реагировать на изменения в микроклимате и подстраивать работу заслонок в соответствии с заданными параметрами.
3. Ручное управление: при необходимости пользователь может также вручную регулировать положение заслонок через интерфейс блока контроллера теплицы или через блок беспроводной связи с помощью предоставленного API.

Таким образом, блок регулировки вентиляции обеспечивает эффективное управление воздушным режимом в теплице.

2.1.6 Блок управления доступом в теплицу

Блок управления доступом в теплицу играет ключевую роль в обеспечении безопасности и контроля доступа к тепличным помещениям. Этот блок тесно связан с блоком контроллера теплицы и включает в себя электропривод и драйвер, которые отвечают за открытие и закрытие двери в теплице.

Основные функции блока управления доступом включают:

1. Открытие и закрытие двери: благодаря электроприводу и соответствующему драйверу, этот блок способен управлять механизмом открытия и закрытия двери в теплице.
2. Управление доступом: блок управления доступом интегрирован в систему контроля доступа, позволяя автоматически управлять доступом в теплицу в зависимости от различных параметров, таких как время суток, температура и влажность воздуха.
3. Ручное управление: помимо автоматического режима, пользователь может также вручную управлять открытием и закрытием двери через интерфейс блока контроллера теплицы.

Таким образом, блок управления доступом в теплицу обеспечивает эффективное управление доступом к тепличным помещениям, повышая безопасность.

2.1.7 Блок реле

Блок реле является важной частью программно-аппаратного комплекса для управления теплицей и обеспечивает управление различными конечными

устройствами, такими как LED лента в блоке освещения и водяная помпа в блоке орошения. Этот блок тесно связан с блоком контроллера теплицы, от которого поступают управляющие сигналы на включение и выключение этих конечных устройств.

Основные функции блока реле включают:

1. Управление освещением: блок реле контролирует включение и выключение LED ленты в блоке освещения в зависимости от полученных команд от контроллера теплицы. Это позволяет автоматизировать процесс управления освещением в теплице и оптимизировать его использование.
2. Управление орошением: блок реле также управляет включением и выключением водяной помпы в блоке орошения на основе команд от контроллера теплицы. Это позволяет автоматически поддерживать оптимальный уровень влажности почвы в теплице и обеспечивать регулярное орошение растений.

Таким образом, блок реле играет важную роль в управлении основными функциями теплицы, обеспечивая автоматическое управление освещением и орошением в соответствии с установленными параметрами и потребностями растений.

2.1.8 Блок системы орошения

Блок системы орошения обеспечивает оптимальный уровень влажности почвы в теплице. Он состоит из водяной помпы, которая отвечает за подачу воды на почву для орошения растений. Этот блок тесно связан с блоком реле, который управляет включением или выключением водяной помпы в зависимости от команд, поступающих от блока контроллера теплицы.

Блок системы орошения получает команды от блока реле и включает или выключает водяную помпу в соответствии с установленными параметрами и расписанием орошения. Такая автоматизация позволяет оптимизировать использование воды и обеспечить регулярное орошение растений без необходимости постоянного контроля.

Важной функцией блока системы орошения является также контроль за уровнем влажности почвы. При достижении определенного уровня влажности система орошения может автоматически отключаться, чтобы избежать переувлажнения почвы, что может привести к гниению корней и другим проблемам.

Таким образом, блок системы орошения обеспечивает эффективное управление процессом орошения в теплице, обеспечивая растения необходимым уровнем влажности почвы для их здоровья и роста.

2.1.9 Блок искусственного освещения

Блок искусственного освещения играет важную роль в обеспечении необходимого уровня освещенности для растений в теплице. Он состоит из LED ленты, которая используется для создания искусственного света,

необходимого для фотосинтеза и здорового роста растений. Этот блок тесно связан с блоком реле, который управляет включением или выключением LED ленты в соответствии с командами, поступающими от блока контроллера теплицы.

Основные функции блока искусственного освещения включают:

1. Поддержание оптимального уровня освещенности: LED лента используется для обеспечения растений необходимым количеством света для фотосинтеза. Она может включаться на определенные временные промежутки или по запросу с помощью блока контроллера теплицы.
2. Энергосбережение: блок реле, управляющий LED лентой, может эффективно контролировать использование энергии, включая и выключая освещение в соответствии с расписанием или условиями окружающей среды. Это помогает сократить энергопотребление и снизить затраты на электроэнергию.

Таким образом, блок искусственного освещения обеспечивает надлежащее освещение для растений в теплице, создавая оптимальные условия для их роста и развития вне зависимости от времени суток и погодных условий.

2.1.10 Блок определения уровня воды

Блок определения уровня воды играет ключевую роль в обеспечении оптимального уровня воды в резервуаре теплицы. Этот блок состоит из датчика уровня воды, который устанавливается в резервуаре для хранения воды.

Основная функция датчика уровня воды – непрерывный мониторинг уровня воды в резервуаре. Как только уровень воды достигает определенного значения, датчик передает соответствующий сигнал блоку контроллера теплицы. Блок контроллера теплицы, в свою очередь, анализирует этот сигнал и принимает решение о включении или выключении водяной помпы для орошения растений в зависимости от текущего уровня воды.

Важно отметить, что блок определения уровня воды обеспечивает эффективное использование водных ресурсов и предотвращает переливы или недостаток воды в системе орошения. Это помогает поддерживать оптимальные условия для роста растений и предотвращает возможные повреждения, связанные с недостаточным или избыточным орошением.

Таким образом, блок определения уровня воды играет важную роль в автоматизации процесса орошения растений в теплице, обеспечивая точное и эффективное управление уровнем воды.

2.1.11 Блок определения интенсивности освещения

Блок определения интенсивности освещения поддерживает оптимальные условия освещения для растений в теплице. Этот блок состоит из датчика интенсивности освещения, который устанавливается внутри

теплицы и непрерывно мониторит уровень освещенности в окружающей среде.

Датчик интенсивности освещения регистрирует количество света, падающего на поверхность, и преобразует это значение в электрический сигнал. Полученный сигнал передается блоку контроллера теплицы, который анализирует данные и принимает решение о необходимости включения или выключения искусственного освещения в теплице.

Основная функция блока определения интенсивности освещения заключается в поддержании оптимального уровня освещенности для растений в зависимости от времени суток и погодных условий. При недостаточном естественном освещении, активируется искусственное освещение, обеспечивая растениям необходимое количество света для фотосинтеза и здорового роста. В то же время, при достаточной интенсивности естественного света, искусственное освещение автоматически отключается, что помогает экономить электроэнергию и оптимизировать ресурсы.

Таким образом, блок определения интенсивности освещения обеспечивает необходимое освещение в соответствии с их биологическими потребностями и внешними условиями окружающей среды.

2.1.12 Блок определения влажности почвы

Блок определения влажности почвы поддерживает оптимальный уровень влажности для растений в теплице. Этот блок состоит из датчика влажности почвы, который устанавливается в грунте рядом с корнями растений и постоянно мониторит уровень влажности в почве.

Датчик влажности почвы измеряет влагосодержание в грунте и преобразует полученные данные в электрический сигнал. Этот сигнал передается блоку контроллера теплицы, который анализирует информацию и принимает решение о необходимости включения или выключения водяной помпы для подачи дополнительной влаги в почву.

Основная функция блока определения влажности почвы заключается в поддержании оптимального уровня влажности для растений. При слишком сухой почве, активируется водяная помпа, которая увлажняет грунт, обеспечивая растениям достаточное количество воды для нормального роста и развития. В случае, если уровень влажности в почве достигает оптимальных значений или превышает их, водяная помпа автоматически отключается, что помогает предотвратить переувлажнение и повреждение корневой системы растений.

Таким образом, блок определения влажности почвы играет важную роль в обеспечении здоровья и процветания растений в теплице.

2.1.13 Блок определения температуры и влажности воздуха

Блок определения температуры и влажности воздуха является ключевым компонентом системы контроля климата в теплице. Он включает в себя

высокоточный датчик, способный измерять температуру и влажность воздуха внутри теплицы.

Этот блок тесно взаимодействует с контроллером теплицы, предоставляя ему актуальные данные о текущих климатических условиях в помещении. Данные о температуре и влажности, собранные датчиком, передаются контроллеру для анализа.

На основании этих данных контроллер принимает решения о регулировке условий внутри теплицы. Например, если температура становится слишком высокой, контроллер автоматически открывает заслонки и двери для проветривания помещения.

Таким образом, блок определения температуры и влажности воздуха играет решающую роль в обеспечении комфортных условий для роста и развития растений в тепличном помещении, обеспечивая им необходимую теплоту и влажность.

2.1.14 Блок идентификации дождя

Блок идентификации дождя играет важную роль в системе управления теплицей, обеспечивая защиту растений от нежелательных воздействий погодных условий. Этот блок включает в себя специализированный датчик дождя, который непрерывно мониторит окружающую среду на предмет наличия осадков.

Сигналы, полученные от датчика дождя, передаются блоку контроллера теплицы. Используя эти данные, контроллер принимает решения о дальнейших действиях в зависимости от текущих погодных условий. Например, если датчик обнаруживает дождь, контроллер автоматически закрывает двери и заслонки теплицы, чтобы защитить растения от избыточной влажности и холода.

Благодаря такой автоматизации система управления теплицей обеспечивает надежную защиту растений от дождевых осадков, предотвращая негативные последствия избыточной влажности и холода для их здоровья и роста. Это позволяет создать оптимальные условия внутри теплицы, поддерживая благоприятную среду для культивирования растений и повышая эффективность производства.

2.1.15 Блок питания

Блок питания обеспечивает необходимое электропитание для всех компонентов системы. В данной системе применяются три блока питания: два на 5 В и один на 12 В.

Первый блок питания на 5 В используется для обеспечения энергией микроконтроллера, который является центральным узлом управления всей системой. Этот блок обеспечивает стабильное напряжение, необходимое для работы микроконтроллера и его периферийных устройств.

Второй блок питания также предоставляет напряжение 5 В и

используется для питания реле, драйвера и сервоприводов.

Третий блок питания на 12 В предназначен для электропривода, водяной помпы и LED ленты. Этот блок обеспечивает более высокое напряжение, необходимое для работы устройств, требующих большей мощности, таких как электроприводы для управления вентиляцией, водяная помпа для орошения растений и LED лента для искусственного освещения.

Таким образом, блок питания играет ключевую роль в обеспечении энергии для всех компонентов системы, обеспечивая их надежную работу и функциональность.

2.2 Описание основных программных блоков устройства

В результате разделения комплекса для управления теплицей были определены следующие программные блоки:

– блок API и обработки команд;
– блок взаимодействия с веб-интерфейсом;
– блок обновления выходов по данным сенсоров;
– блок считывания данных с сенсоров;
– блок обработки ошибок и логирования;
– блок управления выходами;
– блок работы с временем и часами реального времени;
– блок вспомогательных функций;
– блок настроек и инициализации.

На чертеже ГУИР.400201.005 С1 представлена структурная схема, наглядно демонстрирующая блоки и взаимосвязи между ними.

2.2.1 Блок API и обработки команд

Блок API и обработки команд отвечает за взаимодействие с веб-интерфейсом пользователя и обработку полученных команд. Его основная задача – принимать HTTP-запросы от веб-интерфейса, разбирать их и выполнять соответствующие действия в системе.

При получении запроса блок разбирает его и определяет тип команды (например, управление состоянием выходных устройств, запрос настройки и т. д.). Затем блок взаимодействует с другими блоками системы, передавая им полученные команды или запрашивая необходимые данные для выполнения действий.

После обработки команды блок генерирует ответ, который отправляется обратно веб-интерфейсу, содержащий информацию о выполненной операции или текущем состоянии системы.

Блок API и обработки команд тесно связан с блоком взаимодействия с веб-интерфейсом, через который он получает команды от пользователя, а также с блоком управления выходами, чтобы изменить состояние выходных устройств в соответствии с полученными командами.

2.2.2 Блок взаимодействия с веб-интерфейсом

Блок взаимодействия с веб-интерфейсом – это посредник между пользовательским интерфейсом и основной логикой системы. Его задача – принимать запросы от веб-интерфейса и передавать их на обработку соответствующим частям программы. Он анализирует запросы, определяет необходимые действия и направляет их в соответствующие блоки для выполнения.

После выполнения запроса блок взаимодействия с веб-интерфейсом получает результаты и возвращает их обратно в веб-интерфейс в удобном формате, например, через HTTP-ответы. Этот блок также отвечает за обновление пользовательского интерфейса, отображая текущее состояние системы и реагируя на действия пользователя.

Он тесно взаимодействует с блоком API и обработки команд для передачи команд и получения результатов и с блоком управления выходами для обновления состояния интерфейса в соответствии с текущим состоянием системы.

2.2.3 Блок обновления выходов по данным сенсоров

Блок обновления выходов по данным сенсоров отвечает за мониторинг данных, получаемых от различных датчиков, и обновление состояния выходных устройств в соответствии с этими данными. Его основная задача – обеспечить реакцию системы на изменения окружающей среды и выполнить соответствующие действия для поддержания заданных условий.

Этот блок периодически считывает данные с различных сенсоров, таких как температурные датчики, датчики влажности, датчики освещенности и т. д. Полученные данные анализируются и сравниваются с установленными пороговыми значениями или другими условиями.

После анализа данных блок обновления выходов принимает решение о необходимости изменения состояния выходных устройств, таких как приводы, насосы, освещение и другие. Он взаимодействует с блоком управления выходами, чтобы передать команды на изменение состояния выходных устройств.

Этот блок также связан с блоком считывания данных с сенсоров для получения актуальных данных, блоком работы с временем и часами реального времени, а также с блоком обработки ошибок и логирования для регистрации событий и ошибок в процессе работы.

2.2.4 Блок считывания данных с сенсоров

Блок считывания данных с сенсоров отвечает за получение данных с различных датчиков, установленных в системе. Его основная задача состоит в том, чтобы периодически считывать значения сенсоров, таких как температурные датчики, датчики влажности, датчики освещенности и другие.

При запуске системы или по истечении определенного времени блок считывания данных инициирует процесс считывания значений с сенсоров. После получения данных блок передает их для дальнейшей обработки блоку обновления выходов.

Блок считывания данных тесно связан с блоком обновления выходов, поскольку передает полученные данные для анализа и принятия решений о необходимых действиях для поддержания заданных условий в системе. Также он взаимодействует с блоком работы с временем и часами реального времени.

2.2.5 Блок обработки ошибок и логирования

Блок обработки ошибок и логирования отвечает за отслеживание и обработку ошибок, а также за регистрацию различных событий в системе. Его основная задача состоит в том, чтобы обнаруживать ошибки, возникающие в процессе работы системы, и предпринимать соответствующие действия для их устранения или обработки.

В этом блоке происходит анализ различных условий и событий, которые могут указывать на возникновение проблем в системе. Это может быть отсутствие связи с сенсорами, некорректные значения, превышение пороговых значений и другие. При обнаружении ошибок блок обработки ошибок принимает меры по их устранению или уведомляет другие компоненты системы о возникшей проблеме.

Кроме того, блок обработки ошибок и логирования отвечает за ведение журнала событий, в котором регистрируются различные действия и события в системе. Это позволяет в дальнейшем анализировать работу системы, выявлять проблемы и оптимизировать ее функционирование.

Блок обработки ошибок и логирования тесно связан с другими блоками, такими как блок управления выходами, блок обновления выходов по данным сенсоров. Он получает информацию о возможных проблемах от других компонентов системы и предпринимает соответствующие действия для их обработки. Также взаимодействует с блоком веб-интерфейса для отображения сообщений об ошибках пользователю.

Таким образом, блок обработки ошибок и логирования обеспечивает стабильную и безопасную работу управления теплицей.

2.2.6 Блок управления выходами

Блок управления выходами отвечает за активацию и деактивацию различных устройств в системе, таких как сервоприводы, водяные помпы, освещение и другие. Он получает данные о текущем состоянии окружающей среды от блока считывания данных с сенсоров и основываясь на них, а также на заданных пользователем настройках, принимает решения о необходимости изменения состояния выходов.

Например, если температура в помещении превышает заданный порог, блок управления выходами активирует сервоприводы для открытия

вентиляции. Если уровень влажности почвы опускается ниже определенного уровня, блок включает водяную помпу для полива растений.

Этот блок также ответственен за проверку наличия ошибок и проблем в системе. В случае возникновения ошибок, например, недоступности какого-либо из устройств или некорректных данных с сенсоров, он может принимать соответствующие меры, например, отключать неисправное устройство или уведомлять пользователя о проблеме через блок обработки ошибок и логирования.

Данный блок также связан с блоком взаимодействия с веб-интерфейсом и блоком обновления выходов по данным сенсоров.

2.2.7 Блок работы с временем и часами реального времени

Блок работы с временем и часами реального времени отвечает за отслеживание текущего времени и управление временными параметрами в системе. Его задачи включают в себя получение текущего времени от внешнего источника, обновление внутренних часов и календаря системы, а также предоставление временных данных другим компонентам системы по запросу.

Он использует внешние часы реального времени для получения точного времени и даты, а затем синхронизирует внутренние часы системы с этими данными. Благодаря этому система может выполнять определенные задачи в определенное время, например, включать и выключать устройства по расписанию.

Этот блок также связан с другими компонентами системы, такими как блок обновления выходов по данным сенсоров, блок считывания данных с сенсоров и блок обработки ошибок и логирования. Например, он может определять, когда необходимо запускать определенные процессы, и передавать соответствующие команды блоку управления выходами для выполнения этих задач.

Таким образом, блок работы с временем и часами реального времени является неотъемлемой частью системы управления теплицей, обеспечивая точную синхронизацию времени и эффективное выполнение задач в соответствии с расписанием, что способствует оптимальному функционированию системы и обеспечивает здоровье и рост растений.

2.2.8 Блок настроек и инициализации

Блок настроек и инициализации отвечает за установку начальных параметров системы и их последующую загрузку и сохранение. Он играет важную роль в обеспечении правильной работы всей системы, предоставляя возможность пользователю настраивать различные параметры и конфигурации в соответствии с требованиями и предпочтениями.

В этом блоке происходит инициализация различных аппаратных и программных компонентов системы, таких как дисплей, датчики, механизмы

управления, а также загрузка сохраненных пользовательских настроек из памяти и их применение.

Основные задачи блока включают в себя инициализацию параметров времени, установку портов ввода/вывода для взаимодействия с периферийными устройствами, загрузку настроек из памяти (например, EEPROM) при запуске системы, а также сохранение новых настроек, внесенных пользователем, для последующего использования.

Блок настроек и инициализации тесно связан с другими блоками системы, такими как блоки взаимодействия с веб-интерфейсом, обновления выходов по данным сенсоров и работы с временем. Параметры, установленные в блоке настроек и инициализации, используются в блоках взаимодействия с веб-интерфейсом для отображения пользовательских настроек или в блоке обновления выходов по данным сенсоров для корректного управления выходами в соответствии с предпочтениями пользователя.

Таким образом, блок настроек и инициализации обеспечивает основу для правильной работы всей системы, предоставляя необходимые средства для управления и конфигурирования системы в соответствии с требованиями пользователей и условиями окружающей среды.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе будет проведено подробное описание функциональной составляющей программно-аппаратного комплекса для управления теплицей на основе беспроводной технологии. Будут рассмотрены основные блоки системы, их функциональное назначение и взаимосвязи. Особое внимание уделено протоколам связи, используемым в системе, таким как UART, I2C, PWM и Wi-Fi. Для каждого протокола представлено описание его назначения, принципа работы, характеристики и преимущества использования в данном контексте. Кроме того, будут рассмотрены другие важные аспекты функционального проектирования, такие как алгоритмы управления, обработки данных и взаимодействия между компонентами системы. В конце раздела будет представлено обобщенное описание функциональной архитектуры системы, подчеркивающее важность каждого блока и их роли в обеспечении работы программно-аппаратного комплекса для управления теплицей.

3.1 Протоколы и интерфейсы

В данном подразделе будет рассмотрено использование различных протоколов связи и интерфейсов, необходимых для взаимодействия между различными компонентами программно-аппаратного комплекса управления теплицей. Подробно описаны основные протоколы связи, такие как UART, I2C, PWM и Wi-Fi, используемые для передачи данных между микроконтроллерами, сенсорами, исполнительными устройствами и другими периферийными устройствами. Каждый протокол будет рассмотрен с точки зрения его назначения, особенностей работы, преимуществ и ограничений. Также будут описаны интерфейсы взаимодействия между программными и аппаратными компонентами системы, включая методы обмена данными, форматы сообщений и процедуры их обработки. Особое внимание будет уделено выбору наиболее подходящих протоколов и интерфейсов для обеспечения эффективной и надежной работы системы управления теплицей.

3.1.1 Интерфейс UART

Универсальный асинхронный приемопередатчик (UART) [22] – один из наиболее распространенных интерфейсов для передачи данных между электронными устройствами. Почти все микроконтроллеры включают в себя встроенный модуль UART, включая плату Arduino, основанную на микроконтроллере ATmega328.

UART применяется во множестве сценариев, включая следующие:

– подключение Arduino к персональному компьютеру через UART с последующим использованием USB-UART моста;

– работа с Bluetooth и другими радиомодулями, которые также часто используют UART;

– соединение двух контроллеров между собой через UART.

Передача данных через интерфейс UART происходит по двум проводам, где каждое устройство, участвующее в обмене информацией, подключается к противоположным концам этих проводов.

На рисунке 3.1 представлено подключение к интерфейсу UART [22].

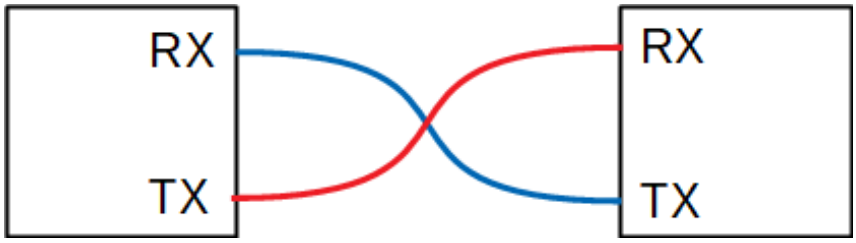


Рисунок 3.1 – Подключение к UART

Интерфейс UART, где RX обозначает «приемник» (от англ. «receiver»), а TX – «передатчик» (от англ. «transceiver»), представляет собой стандартный способ передачи данных между устройствами. UART может работать в двух режимах: полудуплексном, где возможен только прием или только передача, и полнодуплексном, в котором линии приема и передачи отделены друг от друга.

Микроконтроллер ATmega328 оснащен встроенным UART-модулем, который связан с контактами D0 и D1. Контакт D0 предназначен для приема данных (RX), а D1 – для их передачи (TX).

Протокол UART описывает передачу данных в последовательном формате, где каждый бит отправляется в заданные временные интервалы. Скорость передачи данных измеряется в бодах, указывая количество бит, передаваемых в секунду. Кроме самих данных, интерфейс UART включает в себя стартовый и стоповый биты, что увеличивает количество передаваемых битов до 10 для каждого передаваемого байта. Отклонение временных интервалов передачи битов должно быть минимальным, рекомендуемое производителями микроконтроллеров значение не должно превышать 1,5%.

На рисунке 3.2 представлен формат передаваемого байта по UART [23].

Рисунок 3.2 – Формат передаваемого байта по UART

Существуют различные вариации протокола UART с разным количеством битов данных, битов синхронизации и опциональными битами

контроля четности. Однако наиболее распространенный формат использует 10 бит для передачи каждого байта информации. Важно помнить следующее:

– в неактивном режиме выход UART находится в высоком состоянии;

– передача байта начинается с отправки стартового бита, который имеет низкий уровень сигнала;

– передача байта завершается отправкой стопового бита, который имеет высокий уровень сигнала;

– данные передаются по порядку от младшего бита к старшему;

– для передачи одного байта необходимо 10 битов, включая стартовый, стоповый и 8 бит данных;

– время передачи одного байта зависит от скорости передачи и общего количества битов (10).

В таблице 3.1 представлены скорости, которые обычно применяются для интерфейса UART [24].

Таблица 3.1 – Скорости передачи интерфейса UART

Скорость передачи, бод	Время передачи одного бита, мкс	Время передачи байта, мкс
4800	208	2083
9600	104	1042
19200	52	521
38400	26	260
57600	17	174
115200	8,7	87

Для управления обменом данными через UART в Arduino используется встроенный класс `Serial`. Важно понимать различия в форматах передаваемых данных.

Встроенный класс `Serial` не требует поиска и подключения дополнительной библиотеки. Для использования UART достаточно в методе `setup()` разрешить работу порта и установить скорость передачи данных, как показано ниже:

```
void setup() {
  Serial.begin(9600);
}
```

В классе `Serial` данные могут передаваться в двух основных форматах: как бинарный код и как ASCII символы. Например, монитор последовательного порта в Arduino IDE принимает данные в формате ASCII текста.

Таким образом, основные характеристики протокола UART были рассмотрены в подробностях, включая формат передаваемых данных, структуру байта, временные характеристики передачи, а также использование UART в Arduino.

3.1.2 Интерфейс I2C

Интерфейс I2C [26], также известный как IIC, представляет собой распространенный и широко используемый метод последовательной передачи данных. Он работает в полудуплексном режиме и использует два провода для связи. Этот интерфейс был разработан компанией Philips более 30 лет назад и с тех пор стал популярным благодаря скорости передачи данных, которая обычно достигает 100 кбит/с, а в некоторых современных микросхемах может достигать 400 кбит/с. Он также привлекателен своей доступностью и простотой реализации. Важно отметить, что основное предназначение этого интерфейса – передача данных между контроллером и периферийными устройствами внутри одного устройства, что делает его идеальным для внутренних связей в электронных системах.

Для передачи данных используются лишь две линии:

1. SDA – для передачи данных.
2. SCL – для синхронизации передачи.

Отличается от UART более высокой скоростью надежной передачи данных и стабильной передачей на высокой скорости. Благодаря своей архитектуре, I2C позволяет подключать до 127 устройств к одной шине, состоящей из двух проводов SDA (данные) и SCL (тактовые импульсы), без использования дополнительного оборудования, за исключением двух подтягивающих резисторов.

На рисунке 3.3 представлено подключение устройств к I2C [27].

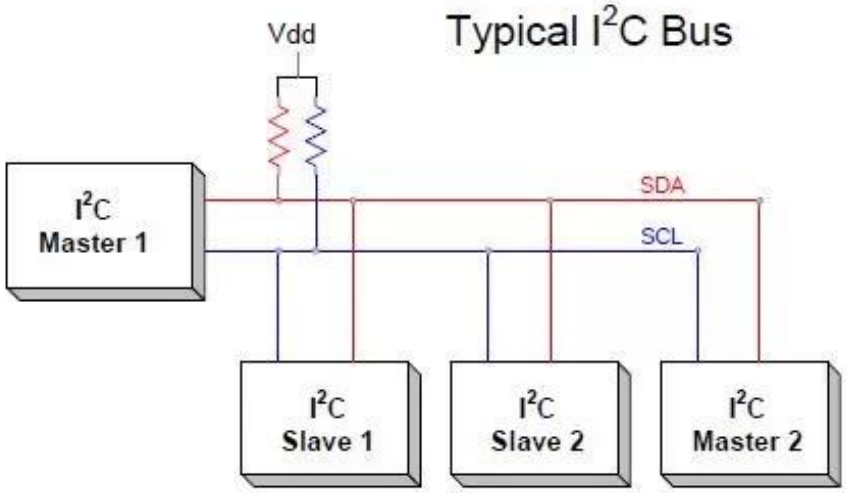


Рисунок 3.3 – Подключение устройств к I2C

Каждое устройство на шине определяется как ведущее или ведомое и имеет свой уникальный адрес в пределах этой шины.

Ведущее устройство (Master) – это устройство, которое инициирует передачу данных, контролирует сигналы синхронизации и завершает передачу информации. Оно может отправлять данные или запрашивать их.

Ведомое устройство (Slave) – это любое адресуемое устройство на шине,

подчиненное мастеру. Оно может принимать данные или передавать их по запросу мастера.

Передатчик – это устройство, которое активно отправляет данные по шине связи.

Приемник – это устройство, которое активно принимает данные с шины связи.

Каждый цикл обмена данных инициируется мастер-устройством отправкой стартового сигнала S. Этот сигнал, иногда называемый стартовым битом, стартовым условием или командой, представляет собой изменение уровня на линии SDA с высокого на низкий, при условии, что на линии SCL уровень остается высоким. Стоп-сигнал P, обозначающий завершение сеанса связи, представляет собой противоположное изменение уровня на линии SDA, с низкого на высокий, при высоком уровне на линии SCL. Весь обмен данными между этими событиями называется «сообщением» и представляет собой фактическую передачу данных. Следует обратить внимание, что изменение уровней на линии SDA во время передачи данных всегда должно происходить только при низком уровне на линии SCL [25].

На рисунке 3.4 представлен цикл обмена данными по I2C [27].

Рисунок 3.4 – Цикл обмена дынными по I2C

После отправки «старт-сигнала» мастер-устройство должно первым делом определиться с тем, какому устройству оно адресует свой запрос и указать, что именно требуется сделать: передать данные в устройство или прочитать их из него. Для этого мастер передает на шину 7-битный адрес ведомого устройства, с которым он намерен взаимодействовать, а также один бит, указывающий направление передачи данных: 0, если данные идут от мастера к ведомому устройству, или 1, если от ведомого устройства к мастеру. Первый байт, посланный сразу после «старт-сигнала», всегда рассматривается как команда адресации всеми устройствами на шине.

После этого все устройства-ведомые на шине проверяют запрошенный адрес собственного. Если адрес совпадает, устройство отвечает, отправив один бит подтверждения ACK, что означает «я здесь, все в порядке». Ответ должно дать только запрошенное устройство. Если бит ACK не возвращается, мастер-устройство понимает, что запрашиваемое устройство отсутствует на шине или «не отвечает».

Существует также бит отсутствия подтверждения NACK, который может сигнализировать о занятости устройства (например, проводится измерение), о желании получателя завершить передачу или о том, что команда, отправленная мастером, не была выполнена по какой-то причине.

На рисунке 3.5 представлен формат данных на шине I2C [27].

Рисунок 3.5 – Формат данных на шине I2C

После получения подтверждения мастер-устройство начинает передачу данных в виде кадра из 8 бит (1 байт). Это происходит, если планировалась передача данных от мастера к ведомому устройству. Если же мастер ждет данных, то сразу после отправки первого бита ASC (адресации) он ожидает первый кадр данных от ведомого устройства без каких-либо задержек. В ответ принимающее устройство должно отправить бит ASC в качестве подтверждения приема кадра. Затем следует передача следующего байта/кадра и так далее. По завершении сеанса связи вместо ASC отправляется NACK.

Когда все данные успешно переданы, мастер отправляет «стоп-сигнал» для завершения сеанса.

На плате Arduino Nano V3.0, используемой в данном проекте, есть определенные пины, которые на аппаратном уровне поддерживают интерфейс I2C. Это пины A4 (для SDA) и A5 (для SCL).

Взаимодействия с интерфейсом I2C в среде Arduino осуществляется через стандартную библиотеку `Wire`, предоставляющей класс с таким же названием. Далее рассмотрены некоторые основные функции.

`.begin(address)` – инициализация класса и подключение к шине. Если адрес не указан, то устройство работает как мастер, а если указан, то это адрес ведомого.

`.available()` – возвращает количество принятых байтов, доступных для чтения.

`.read()` – возвращает следующий принятый байт.

`.write()` – передача байта или последовательности байт, в зависимости от параметров.

Далее представлены функции только для мастера.

`.beginTransmission(address)` – начало передачи данных ведомому устройству с указанным адресом.

`.endTransmission()` – завершение передачи данных ведомому устройству.

Ниже показаны две функции только для ведомого устройства.

`.onReceive(foo)` – устанавливает функцию обратного вызова, которая срабатывает при получении данных от мастера.

`.onRequest(foo)` – устанавливает функцию обратного вызова, которая срабатывает при запросе на отправку данных мастеру.

Из этого списка функций видно, что работа с протоколом как мастера, так и ведомого устройства существенно различается. У мастера все действия осуществляются по команде, когда это требуется, в то время как у ведомого прием и передача данных происходят автоматически при обращении от мастера. Для приема данных ведомое устройство должно быть настроено на вызов определенной функции, а для отправки данных должны быть заранее подготовлены. Ведущее устройство не может инициировать передачу данных самостоятельно; если мастеру требуется постоянное обновление данных, он должен периодически отправлять запросы соответствующему устройству. Такая организация работы имеет свои преимущества и недостатки.

3.1.3 Широтно-импульсная модуляция

Широтно-импульсная модуляция (ШИМ), или Pulse-Width Modulation (PWM) – это метод управления мощностью, применяемый для изменения выходной мощности к нагрузке. Она основана на изменении длительности импульсов при постоянной частоте их появления. Существуют различные типы ШИМ, такие как аналоговая, цифровая, двоичная и троичная, каждая из которых используется в различных сферах и имеет свои особенности.

Основной целью применения ШИМ является увеличение КПД. В этой технологии ключевыми компонентами являются транзисторы или другие полупроводниковые устройства, которые работают в режиме ключа, а не в линейном режиме. Это означает, что транзисторы либо полностью открыты (включены), либо полностью закрыты (выключены). Когда транзистор открыт, он имеет очень высокое сопротивление, что ограничивает ток в цепи и снижает мощность, выделяемую на нем. Когда транзистор закрыт, его сопротивление крайне низкое, что минимизирует падение напряжения на нем и выделяемую мощность. Во время переходных состояний, когда ключ переключается между включенным и выключенным состояниями, происходит значительное выделение мощности, но так как эти переходы очень коротки по сравнению с периодом переключения, средняя мощность потерь на переключение остается незначительной.

ШИМ (широтно-импульсная модуляция) имеет несколько важных характеристик, включая амплитуду, частоту и скважность. Амплитуда определяется в зависимости от требуемого напряжения нагрузки. Частота ШИМ выбирается с учетом нескольких факторов: чем выше частота, тем точнее регулирование, однако ее необходимо выбирать так, чтобы не было заметных пульсаций регулируемого параметра, иначе возникнут проблемы. Повышение частоты также приводит к увеличению коммутационных потерь, поэтому важно использовать быстродействующие элементы. Для управления

электродвигателем частота ШИМ должна быть выше слышимого для человека диапазона (25 кГц и выше), чтобы избежать неприятного свиста. Скважность, или коэффициент заполнения, характеризует величину модуляции. Обычно удобнее выражать коэффициент заполнения в процентах, где он равен отношению длительности импульса к периоду.

На рисунке 3.6 представлена диаграмма ШИМ сигнала [28].

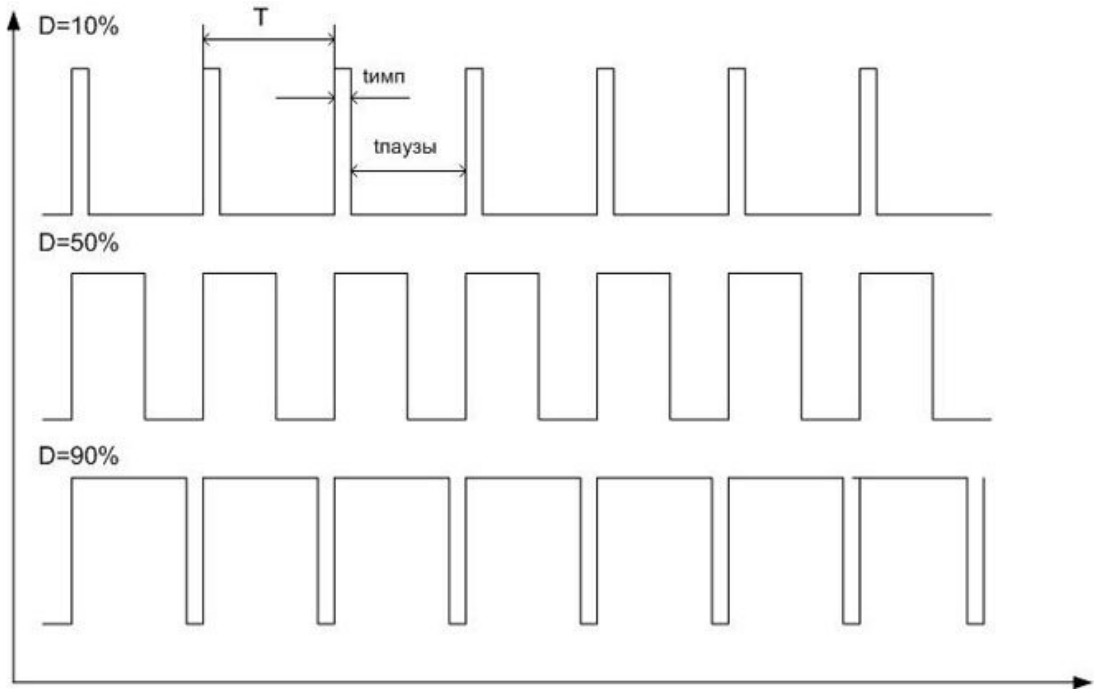


Рисунок 3.6 – Диаграмма ШИМ сигнала

На плате Arduino Nano V3.0 доступны 6 контактов: 3, 5, 6, 9, 10 и 11, которые используются для создания аппаратного ШИМ. По умолчанию, частота сигнала на контактах 5 и 6 составляет 1 кГц, а на остальных контактах – всего лишь 500 Гц.

Существует встроенная функция для генерации ШИМ – `analogWrite(pin, duty)`:

– `pin` – это номер PWM-пина;

– `duty` – это параметр, определяющий заполнение ШИМ-сигнала. По умолчанию он представлен в 8-битной разрядности, что означает, что его значение может варьироваться от 0 до 255.

3.1.4 Wi-Fi

Wi-Fi – это беспроводная технология связи, которая позволяет устройствам, таким как смартфоны, планшеты и компьютеры, обмениваться данными через радиоволновой сигнал [29].

Wi-Fi основан на наборе стандартов IEEE 802.11. Данные стандарты определяют правила для создания беспроводных локальных сетей (WLAN). Эти стандарты разрабатываются институтом инженеров электротехники и

электроники (IEEE). В настоящее время существует несколько вариаций стандарта IEEE 802.11, таких как 802.11b, 802.11ax и другие, каждая из которых имеет свои уникальные характеристики.

Wi-Fi представляет собой беспроводной способ соединения устройств с интернетом. В квартирах, офисах и общественных местах он используется для создания беспроводной точки доступа к сети. Это позволяет подключать к ней устройства, такие как смартфоны, планшеты и ноутбуки, не используя сетевые кабели.

В таблице 3.2 представлены стандарты Wi-Fi и их характеристики.

Таблица 3.2 – Характеристики стандартов Wi-Fi

Название	Скорость	Частота	Комментарий
802.11a	54 Mbps максимальная, но обычно от 6 до 24 Mbps	5 GHz	Не поддерживает сети стандартов b или g. Этот протокол относится к одним из первых разработанных, но на сегодняшний день все еще широко применяется во многих устройствах.
802.11b	11 Mbps	2.4 GHz	Может работать с сетями стандарта g. На практике, стандарт g был адаптирован для совместимости с b с целью обеспечения поддержки большего количества устройств.
802.11g	54 Mbps	2.4 GHz	Самый распространенный вид сети. Его сочетание скорости и обратной совместимости делает его подходящим для современных сетей.
802.11n	100 Mbps	2.4 и 5 GHz	Обычно скорость составляет 100 Мбит/с, в оптимальных условиях она может достигать 600 Мбит/с за счет одновременного использования нескольких частот и комбинирования их скорости.
802.11ac	1Gbps	5 GHz	Стандарт 802.11ac совместим с предыдущими версиями 802.11b/g/n и обеспечивает скорость до 1300 Мбит/с на частоте 5 ГГц и до 450 Мбит/с на частоте 2,4 ГГц.

Wi-Fi также играет важную роль в концепции интернета вещей (IoT). Он представляет собой стандарт беспроводного подключения и взаимодействия между различными устройствами IoT, такими как термостаты, лампочки, камеры видеонаблюдения, замки, выключатели и другие умные гаджеты в домах.

3.2 Аппаратные блоки устройства

Комплекс для управления теплицей состоит из следующих аппаратных блоков:

– блок контроллера теплицы;
– блок беспроводной связи;
– блок управления контроллером;
– блок отображения информации;
– блок регулировки вентиляции;
– блок управления доступом в теплицу;
– блок реле;
– блок системы орошения;
– блок искусственного освещения;
– блок определения уровня воды;
– блок определения интенсивности освещения;
– блок определения влажности почвы;
– блок определения температуры и влажности воздуха;
– блок идентификации дождя;
– блок питания.

Далее будет рассмотрена функциональная часть каждого блока.

3.2.1 Блок контроллера теплицы

Блок контроллера теплицы включает в себя печатную плату, на которой размещены все основные компоненты, необходимые для эффективного управления и мониторинга различных аспектов окружающей среды в тепличном помещении. Внимание к деталям при разработке этой платы позволило создать компактное и функциональное устройство, которое легко интегрируется в инфраструктуру теплицы и обеспечивает высокую производительность и надежность работы.

Основными особенностями печатной платы являются:

1. Интеграция с Arduino Nano V3.0. Плата контроллера теплицы основана на микроконтроллере Arduino Nano V3.0, что обеспечивает широкие возможности программирования и гибкость в настройке функций управления теплицей.

2. Разнообразие подключаемых устройств. Печатная плата обладает множеством разъемов и выводов, предназначенных для подключения различных устройств и датчиков. Это включает в себя цифровые и аналоговые входы, реле для управления освещением и насосом, а также интерфейсы для подключения датчиков температуры, влажности, давления и других параметров окружающей среды.

3. Удобство монтажа и обслуживания. Благодаря компактным размерам (100 × 100 мм) и оптимизированной компоновке компонентов, плата легко монтируется внутри тепличного помещения. Это обеспечивает удобство при установке и обслуживании, а также экономит место внутри теплицы.

Таким образом, печатная плата контроллера теплицы представляет собой ключевой элемент системы управления.

Блок контроллера теплицы является главным блоком, так как в его основе лежит плата микроконтроллера Arduino Nano V3.0, способный обеспечивать эффективное управление различными аспектами окружающей среды внутри тепличного помещения.

В основе платы установлен микроконтроллер ATmega328P, обладающий высокой производительностью и энергоэффективностью. Этот 8-битный процессор обеспечивает возможность достижения до 16 миллионов инструкций в секунду при тактовой частоте 16 МГц. Он оснащен 32 КБ памяти программ, из которых 2 КБ заняты загрузчиком, а также имеет 2 КБ внутренней оперативной памяти SRAM и 1 КБ постоянной памяти EEPROM. Кроме того, на борту предусмотрены 32 общих регистра общего назначения и реальный счетчик времени с отдельным осциллятором для точного учета времени.

Существенной частью его функционала является возможность управления питанием. Для подключения к источнику питания предусмотрен разъем USB Type-C, а также возможность использования внешнего нестабилизированного источника питания от 7 до 15 вольт через соответствующий вывод (пин 30) или стабилизированного 5-вольтового источника (пин 27). Благодаря различным режимам сна, включая режимы простоя, снижения шума АЦП и полного отключения, контроллер обеспечивает оптимизацию энергопотребления в различных ситуациях, что особенно важно для устройств, работающих в автономном режиме.

На рисунке 3.7 представлена блок-схема микроконтроллера Arduino Nano [30].

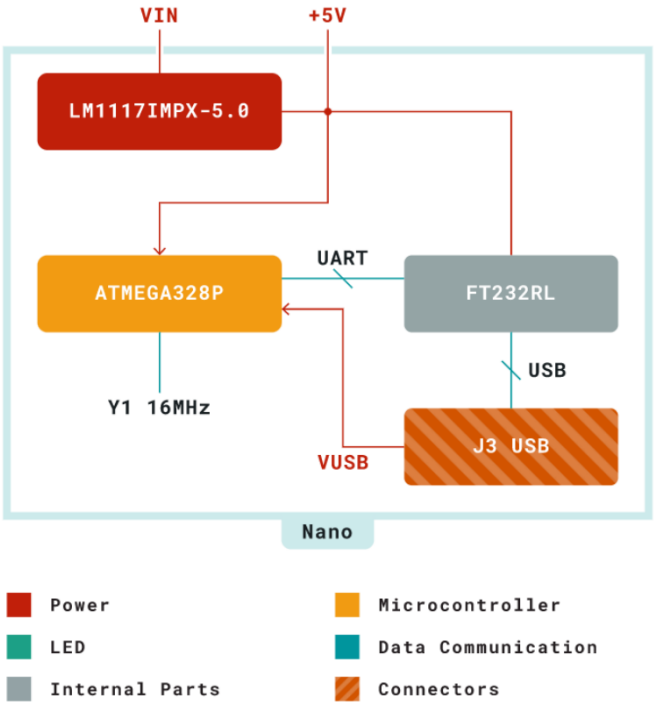


Рисунок 3.7 – Блок-схема микроконтроллера Arduino Nano

С точки зрения взаимодействия с внешними устройствами, контроллер теплицы обладает обширным набором входов и выходов. Он оснащен 20 цифровыми и 8 аналоговыми портами, а также 6 каналами ШИМ, что позволяет эффективно управлять различными устройствами и датчиками внутри теплицы для поддержания оптимальных условий роста растений.

3.2.2 Блок беспроводной связи

Блок беспроводной связи в системе теплицы должен выполнять ряд задач, включая передачу данных о состоянии тепличной среды (температура, влажность, освещенность и т. д.) на удаленное устройство для мониторинга и управления, а также принимать команды от удаленного управляющего устройства для регулировки параметров внутри теплицы.

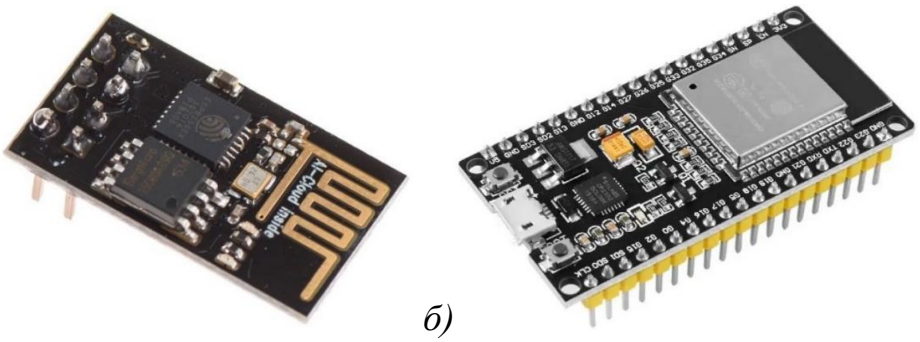
При выборе Wi-Fi модуля нужно обратить внимание на его характеристики, такие как дальность передачи, скорость передачи данных, надежность соединения и простота в настройке. Также при выборе модуля Wi-Fi для комплекса управления теплицей важно учитывать не только его характеристики, но и его совместимость с Arduino Nano для обмена данными по UART.

В таблице 3.3 приведено сравнение некоторых Wi-Fi модулей [32, 34].

Таблица 3.3 – Характеристики Wi-Fi модулей

Характеристика	ESP-01	NodeMCU 32S
Микросхема	ESP8266	ESP32
Номинальное напряжение	3,3 В	5 В
Потребляемый ток	до 220 мА	до 500 мА
Частота процессора	80 МГц	80 – 240 МГц
Оперативная память	96 КБ	520 КБ
Максимальная выходная мощность	19,5 дБ	20,5 дБ
Wi-Fi	802.11 b/g/n с WEP, WPA, WPA2	802.11 b/g/n
Диапазон частот Wi-Fi	2,4 – 2,5 ГГц	2,4 – 2,5 ГГц
Bluetooth	–	Bluetooth 4.2 BLE
Режим работы	STA, AP, STA+AP	STA, AP, STA+AP
Количество цифровых выводов	4	38
Интерфейсы	UART / HSPI / I2C / I2S / GPIO / PWM	UART / GPIO / ADC / DAC / SDIO / SD card / PWM / I2C / I2S
Вес	2 г	10 г
Габариты	25 × 15 × 3 мм	25 × 48 × 3 мм

На рисунке 3.8 изображены Wi-Fi модули [31, 33].



а) *б)*

Рисунок 3.8 – Wi-Fi модули: *а* – ESP-01, *б* – NodeMCU 32S

Модуль ESP8266, известный как ESP-01, представляет собой компактное и стабильное решение для беспроводной связи. Он обладает достаточной производительностью и поддерживает стандарты 802.11 b/g/n, что обеспечивает широкий охват сети Wi-Fi. Кроме того, его низкое потребление энергии делает его идеальным выбором для систем, работающих от аккумуляторов или с ограниченным источником питания.

На рисунке 3.9 представлена блок-схема ESP8266 [32].

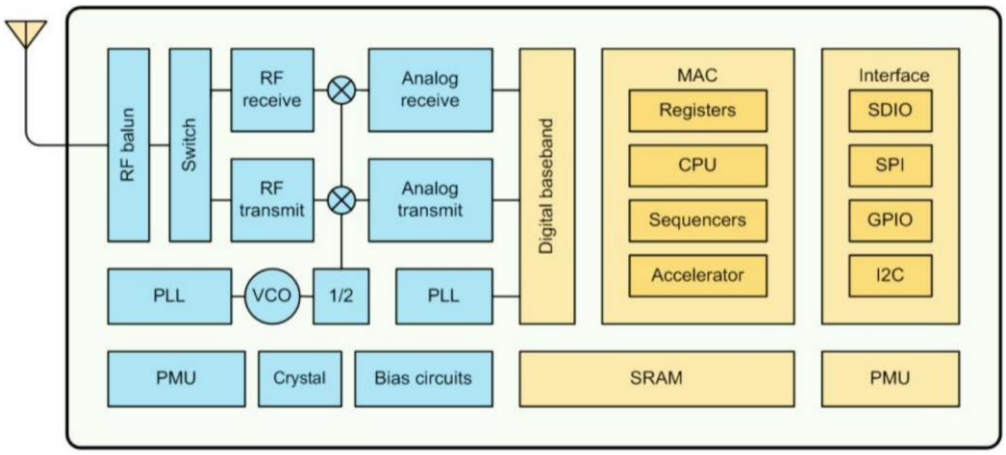


Рисунок 3.9 – Блок-схема ESP8266

Преимуществами ESP-01 являются:

– легкая интеграция с Arduino Nano через интерфейс UART;

– удобство в обмене данными между модулем Wi-Fi и контроллером теплицы;

– миниатюрные размеры (25 × 15 × 3 мм).

Учитывая требования проекта к надежной беспроводной связи, энергоэффективности и удобству интеграции с Arduino Nano, модуль ESP-01 является оптимальным выбором для реализации блока беспроводной связи в системе управления теплицей. Его функциональность, сочетающая в себе стабильное соединение, низкое энергопотребление и удобство в управлении, делает его идеальным партнером для Arduino Nano в данном проекте.

3.2.3 Блок управления контроллером

Для управления контроллером используется энкодер, который обеспечивает удобное и интуитивно понятное управление различными параметрами и функциями системы. Энкодер представляет собой устройство, позволяющее пользователю вращением специального ручного колеса изменять значения на дисплее и выбирать опции.

В рамках данного блока управления контроллером, энкодер выполняет следующие функции:

1. Управление режимами работы. Энкодер позволяет выбирать режимы работы системы, такие как автоматический режим и ручной режим. В автоматическом режиме система самостоятельно контролирует условия в теплице и принимает решения об управлении устройствами на основе заданных параметров. В ручном режиме пользователь может непосредственно управлять состоянием устройств, игнорируя автоматические настройки.

2. Отображение текущих значений. Энкодер также позволяет пользователю просматривать текущие значения различных параметров системы на дисплее. Это включает в себя информацию о температуре и влажности воздуха, уровне освещенности, влажности почвы, а также о состоянии других устройств, таких как дверь, заслонки и насос.

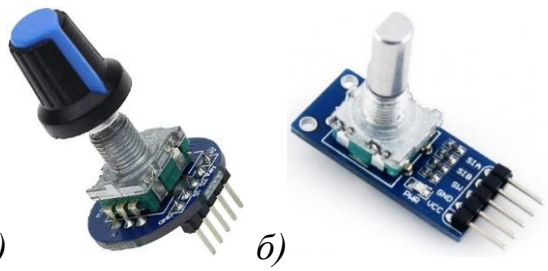
3. Управление ручными режимами. В ручном режиме пользователям предоставляется возможность непосредственного управления состоянием различных устройств, таких как заслонки и насос, вращая энкодер и выбирая соответствующие значения.

В таблице 3.4 приведены характеристики некоторых моделей энкодеров [35, 36].

Таблица 3.4 – Характеристики энкодеров

Характеристика	EC11	Rotation Sensor
Функция нажатия	Да	Да
Напряжение питания	3 – 5,3 В	3 – 5,3 В
Количество импульсов на 360 гр.	20	15
Вес	5 г	11 г

На рисунке 3.10 представлены данные энкодеры [35, 36].



а) *б)*

Рисунок 3.10 – Энкодеры – *а)* EC11, *б)* Rotation Sensor

Энкодер EC11 является предпочтительным выбором благодаря следующим преимуществам:

– обеспечивает большее количество импульсов на 360 градусов, что обеспечивает высокую точность определения положения;

– штыревые разъемы расположены удобно, что упрощает интеграцию с платой контроллера.

Для работы с энкодером необходимо в прошивке написать некоторые инструкции, необходимые для подключения энкодера к микроконтроллеру.

В файле `pins.h` определяются пины для взаимодействия с энкодером:

```
#define SW 0
#define DT 2
#define CLK 3
```

Здесь используются следующие обозначения:

– `SW` – этот пин обозначает кнопку энкодера. Пин подключается к цифровому пину микроконтроллера.

- `DT` и `CLK` – эти пины обозначают выводы энкодера, которые передают сигналы об изменении положения вала. `DT` (Data) и `CLK` (Clock) используются для определения направления вращения энкодера и количества шагов. Они подключаются к цифровым пинам микроконтроллера.

В файле `objects.h` выполняется объявление экземпляра класса `EncButton`, который используется для работы с энкодером в программе:

```
#include <EncButton.h>
extern EncButton enc;
```

В файле `objects.cpp` создается экземпляр класса `EncButton` с именем `enc` и инициализируется с помощью конструктора класса:

```
EncButton enc(CLK, DT, SW);
```

Конструктор `EncButton` принимает три параметра: `CLK`, `DT` и `SW`, которые являются пинами, используемыми для подключения энкодера.

Данный экземпляр класса используется в функциях `updateOutputs()` и `updateServer()`:

– `updateOutputs()` – функция в файле `outputs.cpp`, в которой происходит обновление состояний устройств, в том числе и энкодера;

– `updateServer()` – функция в файле `web.cpp`, которая обрабатывает команды, поступающие через последовательный порт, включая команды, связанные с энкодером.

3.2.4 Блок отображения информации

Блок отображения информации играет ключевую роль в представлении

данных пользователю. Его цель – отображение текущего состояния системы, измерений с датчиков, управление настройками и другую важную информацию.

Для этого блока подходит дисплей с подсветкой, достаточным разрешением и надежным интерфейсом взаимодействия с микроконтроллером. Рассмотрим важные характеристики:

1. Тип дисплея. Можно выбрать между жидкокристаллическим (LCD) и органическим светодиодным (OLED) дисплеем в зависимости от требований к размеру, разрешению, энергопотреблению и другим факторам.

2. Размер и разрешение. Необходимо выбрать дисплей с достаточным размером и разрешением для отображения всей необходимой информации без необходимости прокрутки или масштабирования.

3. Интерфейс. Для удобства подключения к микроконтроллеру лучше выбрать дисплей с поддержкой стандартных интерфейсов, таких как I2C, SPI или параллельный.

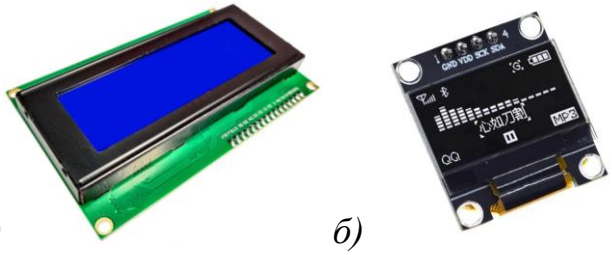
4. Подсветка. Подсветка экрана обеспечивает удобство использования в условиях недостаточного освещения. Желательно выбрать дисплей с настраиваемой подсветкой.

В таблице 3.5 представлено сравнение характеристик дисплеев [37, 38].

Таблица 3.5 – Сравнение характеристик дисплеев

Характеристика	LCD2004	SSD1306
Тип дисплея	LCD	OLED
Интерфейс	IIC / I2C / TWI	I2C
Контроллер дисплея	PCF8574	SSD1306
Разрешение	–	128 × 64 точек
Количество символов в строке	20	–
Количество строк	4	–
Цвет подсветки	Синий	–
Цвет символов	белый	белый
Угол обзора	180 градусов	160 градусов
Напряжение питания	5 В	3 – 5 В
Размеры	98 × 60 × 12 мм	27,3 × 27,8 × 3,7 мм

На рисунке 3.10 представлены данные дисплеи [37, 38].



а) *б)*

Рисунок 3.10 – Дисплеи: *а* – LCD2004, *б* – SSD1306

Для блока отображения информации был выбран дисплей LCD2004 по следующим причинам:

1. Размер и разрешение. LCD2004 имеет размер 20 символов в 4 строках, что обеспечивает достаточное количество места для отображения разнообразной информации, включая измерения с датчиков, текущие настройки и другие важные данные.

2. Интерфейс. Дисплей LCD2004 подключается по шине I2C, что делает его удобным в использовании с микроконтроллерами, так как требует всего двух проводов для передачи данных.

3. Подсветка. Дисплей оснащен подсветкой, что обеспечивает удобство использования в условиях недостаточного освещения, что может быть важно для работы в сельском хозяйстве.

Взаимодействие с дисплеем в программе происходит в нескольких местах.

В файле `objects.h` происходит объявление переменной `lcd`:

```
#include <LiquidCrystal_I2C.h>
extern LiquidCrystal_I2C lcd;
```

В файле `objects.cpp` инициализируется дисплей с заданным адресом и размером:

```
LiquidCrystal_I2C lcd(LCD_ADDR, 20, 4);
```

Код в файле `lcd.cpp` отвечает за обновление информации на LCD дисплее и управление им:

1. В начале кода определяются массивы байтов для создания пользовательских символов, таких как иконки воды, облака, цветка и других:

```
static uint8_t cloudIcon[] = {
  B01110,
  B11111,
  B11111,
  B00000,
  B01000,
  B00010,
  B01000,
  B00000
};
```

2. Функция `initLCD()` инициализирует LCD дисплей и создает пользовательские символы:

```
void initLCD() {
  lcd.init();
  lcd.createChar(LCD_WATER, waterIcon);
  lcd.createChar(LCD_NO_WATER, noWaterIcon);
```

```
  lcd.createChar(LCD_LIGHT, lgithIcon);
  lcd.createChar(LCD_CLOUD, cloudIcon);
  lcd.createChar(LCD_FLOWER, flowerIcon);
  lcd.backlight();
  lcd.clear();
}
```

3. Функция `updateLCD()` обновляет информацию на дисплее в зависимости от текущей страницы. В данном коде реализовано переключение между страницами с помощью энкодера, а также контроль некоторых параметров устройства.

3.1. Если текущая страница – 0, то выводится информация о времени, температуре, влажности, уровне освещенности, влажности почвы, уровне воды, дожде и состоянии выходов.

3.2. Если текущая страница – 1, то выводится информация и возможность управления выходами (дверью, заслонками, освещением).

3.3. Если текущая страница – 2, то выводится информация и возможность управления насосом.

3.4. Если текущая страница – 3, то предоставляется возможность сброса настроек до заводских.

4. Последний блок кода отвечает за автоматическое выключение подсветки дисплея после заданного времени бездействия.

```
#ifdef LCD_BACKLIGHT_TIMEOUT
  {
    static uint64_t backlightTimer = 0;
    static bool backlit = true;
    if (enc.action()) {
      if (!backlit) lcd.backlight();
      backlit = true;
      backlightTimer = millis();
    } else if (backlit && millis() - backlightTimer > LCD_BACKLIGHT_TIMEOUT * 1000u) {
      lcd.noBacklight();
      backlit = false;
    }
  }
#endif
```

Таким образом, этот код реализует интерфейс взаимодействия с LCD дисплеем, предоставляя пользователю информацию о состоянии системы и возможность управления некоторыми параметрами.

3.2.5 Блок управления вентиляцией

Выбор сервопривода для блока регулировки вентиляции основан на нескольких критериях, включая:

1. Угол поворота. Сервопривод должен обеспечивать достаточный угол

поворота, чтобы открывать и закрывать вентиляционные заслонки на нужный уровень.

2. Мощность и скорость. Сервопривод должен обладать достаточной мощностью и скоростью для обеспечения плавного и надежного управления заслонками вентиляции.

3. Надежность и долговечность: Выбранный сервопривод должен быть надежным и обеспечивать стабильную работу в течение длительного времени, особенно при регулярном использовании.

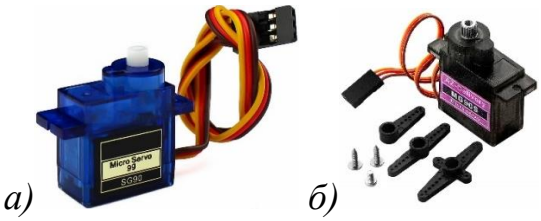
4. Совместимость с контроллером. Сервопривод должен быть совместим с контроллером, который будет управлять им. Это включает в себя соответствие электрическим характеристикам, протоколам связи и другим техническим требованиям.

В таблице 3.6 представлены характеристики сервоприводов [39, 40].

Таблица 3.6 – Характеристики сервоприводов

Характеристика	SG90	MG90S
Напряжение питания	3,3 – 6 В	4,8 – 6 В
Время поворота на угол 60°	100 мс	90 мс
Крутящий момент	1,6 кг/см	2 кг/см
Максимальный угол поворота	180°	180°
Материал шестерней	Нейлон	Латунь, алюминиевый сплав
Материал корпуса	ABS	Пластик
Масса	9 гр.	13,4 г
Габаритные размеры	32 × 12 × 31 мм	22 × 12 × 28 мм

На рисунке 3.11 представлены данные сервоприводы [39, 40].



а) *б)*

Рисунок 3.11 – Сервоприводы: *а* – SG90, *б* – MG90S

Сервопривод SG90 идеально подходит для данного проекта благодаря своей надежности, доступности и компактным размерам. SG90 имеет широкий диапазон углов поворота (0-180 градусов) и низкое энергопотребление. Этот сервопривод легко доступен для приобретения и обладает универсальными характеристиками, что делает его превосходным выбором для управления вентиляцией в данном проекте.

В коде программы инициализация и управление сервоприводами

происходит в нескольких местах:

1. В файле `pins.h` определены пины, которые используются для подключения сервоприводов:

```
#define SERVO1 13
#define SERVO2 A0
```

2. В файле `objects.h` объявлены объекты, представляющие сервоприводы:

```
#include <Servo.h>
extern Servo servo1;
extern Servo servo2;
```

3. В файле `objects.cpp` инициализируются объекты сервоприводов:

```
Servo servo1;
Servo servo2;
```

4. Управление сервоприводами происходит в функции `updateOutputs()` в файле `outputs.cpp`. В этой функции происходит обновление состояний всех выходных устройств, включая сервоприводы. Сначала вычисляются состояния всех устройств на основе текущих параметров и данных с датчиков. Затем, в зависимости от текущего состояния и установок, управление сервоприводами осуществляется с помощью функции `servo1.write()` и `servo2.write()`, которые устанавливают угол поворота для каждого сервопривода.

3.2.6 Блок управления доступом в теплицу

При выборе электропривода для управления доступом в теплицу следует учитывать несколько факторов:

1. Нагрузка и размеры двери. Электропривод должен быть достаточно мощным, чтобы с легкостью открывать и закрывать дверь теплицы. Необходимо учесть вес и размеры двери, а также сопротивление движению, если оно значительно.

2. Надежность и долговечность. Электропривод должен быть надежным и долговечным, чтобы обеспечивать бесперебойную работу системы управления доступом в течение длительного времени.

3. Совместимость с драйвером. Выбранный электропривод должен быть совместим с используемым драйвером, который управляет его работой. Это включает в себя совместимость по напряжению питания, сигнальным интерфейсам и другим параметрам.

4. Управление и программирование. Предпочтительно выбрать электропривод, который легко управлять и программировать. Это позволит

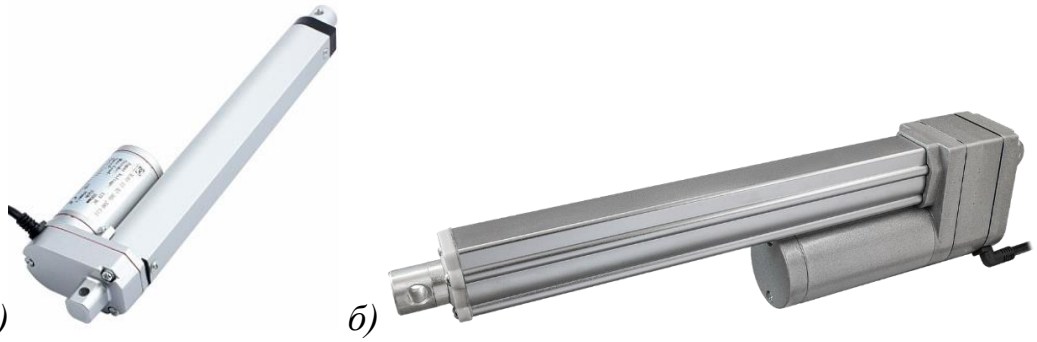
интегрировать его в вашу систему управления с минимальными сложностями.

В таблице 3.7 приведены технические характеристики электроприводов [41, 42].

Таблица 3.7 – Характеристики электроприводов

Характеристика	Линейный привод 750N	Линейный привод HTA-18K
Напряжение питания двигателя	12 В	12 В
Усилие	75 кгс	12 кгс
Максимальная длина хода	350 мм	200 мм
Скорость движения	10 мм/с	10 мм/с
Концевые выключатели	встроенные	встроенные
Номинальный ток	2,3А	7,5А
Рабочая температура	-25°С … +60°С	-20°С …+65°С
Класс защиты	IP54	IP65

На рисунке 3.12 представлены данные приводы [41, 42].



а) *б)*

Рисунок 3.12 – Электроприводы: *а* – 750N, *б* – HTA-18K

Для проекта был выбран линейный привод 750N ввиду его повышенной мощности, что обеспечивает надежное открытие и закрытие двери теплицы. Одним из ключевых факторов при выборе данной модели стало наличие встроенных концевых выключателей, что позволяет точно контролировать положение двери и избежать возможных повреждений или срывов при достижении крайних точек движения.

Выбранный линейный привод 750N работает с использованием драйвера TB6612FNG. Этот драйвер обеспечивает эффективное управление приводом, обеспечивая достаточную мощность для его работы. Он обладает высокой производительностью и надежностью. Такой драйвер обеспечивает стабильную работу привода, минимизируя риск сбоев и обеспечивая долговечность системы в целом.

В таблице 3.8 представлены технические характеристики данного драйвера [43].

Таблица 3.8 – Характеристики драйвера TB6612FNG

Характеристика	TB6612FNG
Входное напряжение для двигателей	4,5 – 13,5 В
Напряжение логики	2,7 – 5,5 В
Максимальный ток на канал	3 А
Частота ШИМ	100кГц
Фильтрующий конденсатор питания	Да
Встроенная тепловая схема отключения	Да
Защита от обратного тока	Да
Рабочая температура	-20°C … +80°C
Размер	20 × 17 мм.

На рисунке 3.13 представлен данный драйвер.

Рисунок 3.13 – Драйвер TB6612FNG

В программе управление электроприводом, который работает через драйвер, осуществляется в функции `updateOutputs()` файла `outputs.cpp`. В этой функции проверяется состояние двери и в зависимости от заданных параметров управляется состоянием драйвера, который управляет электроприводом.

```
#ifdef DRIVER_LEVEL
  digitalWrite(DRV_SIGNAL1, DRIVER_LEVEL != door.state);
  digitalWrite(DRV_SIGNAL2, DRIVER_LEVEL != !door.state);
  analogWrite(DRV_PWM, settings.driveSpeed);
#endif
```

В данном коде:

– `DRIVER_LEVEL` – определяет уровень сигнала на драйвере для управления движением двери;

– `door.state` – определяет состояние двери (открыта или закрыта);

– `DRV_SIGNAL1` и `DRV_SIGNAL2` – пины для управления направлением движения;

– `DRV_PWM` – пин для управления скоростью движения.

Таким образом, при вызове функции `updateOutputs()` происходит управление электроприводом в соответствии с состоянием двери и заданными параметрами.

3.2.7 Блок реле

Для эффективного управления водяной помпой и LED-лентой в проекте был выбран двухканальный модуль реле. Этот выбор обусловлен необходимостью управления двумя конечными устройствами, каждое из которых требует независимого контроля.

Кроме того, важным фактором при выборе модуля реле было его напряжение активации катушки. Учитывая, что напряжение, поступающее от логических уровней микроконтроллера составляет 5VDC, был выбран модуль реле с соответствующим напряжением активации катушки.

В модуле реле, который используется в проекте, должна быть встроена опторазвязка. Она обеспечивает электрическую изоляцию между входными и выходными контактами реле. Это важно для защиты микроконтроллера и других устройств управления от перенапряжений, помех и других нежелательных эффектов, которые могут возникать на высоковольтных устройствах, таких как водяная помпа и LED-лента.

Опторазвязка работает на основе использования оптрона или фототранзистора, который преобразует электрический сигнал в оптический и обратно. Это позволяет создать электрическую изоляцию между управляющим сигналом, поступающим от микроконтроллера, и высоковольтными нагрузками, подключенными к выходам реле.

На рисунке 3.14 представлен данный модуль реле [44].

Рисунок 3.14 – Модуль реле

В таблице 3.9 приведены технические характеристики модуля реле [44].

Таблица 3.9 – Характеристики модуля реле

Характеристика	2-канальный модуль реле с опторазвязкой
1	2
Рабочий ток реле	15 – 20 мА
Управляющее напряжение	5В
Скорость переключений	до 300 операций / мин (мех.), до 30 операций / мин (эл.)
Время срабатывания реле при включении	до 10 мс

Продолжение таблицы 3.9

1	2
Время срабатывания реле при выключении	до 5 мс
Реле высокого тока	SRD-05VDC-SL-C AC250V 10A, AC125V 10A, DC30V 10A, DC28V 10A
Стандартный интерфейс, через который можно управлять релейным модулем с помощью контроллеров	Arduino, 8051, AVR, PIC, DSP, ARM, ARM, MSP430, TTL logic
Вес	30 г
Размеры	49,3 × 38,2 × 20 мм

Объявление и инициализация переменных, отвечающих за управление реле, находятся в файле `pins.h`:

```
#define RELAY1 1
#define RELAY2 4
#define RELAY3 5
#define RELAY4 6
#define RELAY5 7
#define RELAY6 8
#define RELAY7 9
const uint8_t relayPins[] = {RELAY1, RELAY2, RELAY3, RELAY4, RELAY5, RELAY6, RELAY7};
```

Здесь определены пины, к которым подключены модули реле, и создается массив `relayPins`, который содержит эти пины.

Инициализация реле происходит в функции `initOutputs()` из файла `outputs.cpp`:

```
void initOutputs() {
  for (uint8_t i = 0; i < sizeof(relayPins) / sizeof(relayPins[0]); i++) {
    pinMode(relayPins[i], OUTPUT);
    digitalWrite(relayPins[i], LOW);
  }
}
```

В этой функции каждый пин, указанный в массиве `relayPins`, настраивается как выход и устанавливается в `LOW` для инициализации реле в выключенном состоянии.

Управление модулем реле в программе осуществляется в функции `updateOutputs()` из файла `outputs.cpp`. В этой функции происходит управление состоянием реле в зависимости от текущих значений переменных состояния, таких как `pump.state`, `light.state`, `door.state`,

`flap1.state` и `flap2.state`.

Каждое реле включается или выключается в соответствии с заданным состоянием. Например, если переменная `light.state` равна `true`, то соответствующее реле будет включено, чтобы включить освещение. Похожим образом, управляется и водяная помпа.

В функции `updateOutputs()` происходит просмотр текущих состояний управляемых устройств и соответствующее управление реле для подачи или прекращения электропитания на соответствующие устройства.

3.2.8 Блок системы орошения

Для блока орошения необходимо выбрать водяную помпу, которая обеспечит эффективное распределение воды по всему тепличному участку. При выборе помпы следует учитывать несколько ключевых факторов:

1. Производительность. Помпа должна иметь достаточную производительность для обеспечения необходимого объема воды и равномерного орошения всей площади теплицы.
2. Надежность. Выбранная помпа должна быть надежной и обеспечивать стабильную работу без сбоев на протяжении всего сезона.
3. Энергоэффективность. Помпа должна потреблять минимальное количество энергии при работе, чтобы снизить затраты на электроэнергию.
4. Совместимость с системой. Помпа должна быть совместима с другими компонентами системы орошения и легко интегрироваться в общую систему управления теплицей.

Помимо этого, важно учитывать параметры помпы, такие как ее мощность, подача воды в минуту, давление и другие технические характеристики, чтобы выбрать наиболее подходящую модель.

На рисунке 3.15 представлены водяные насосы [45, 46].

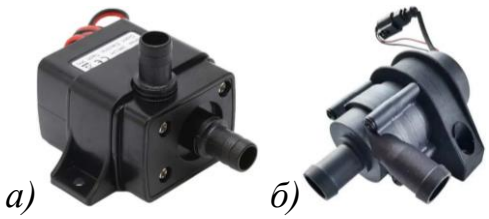


Рисунок 3.15 – Водяные насосы: *а* – DC30E, *б* – LF Bros

В таблице 3.10 представлены технические характеристики данных моделей водяных насосов [45, 46].

Таблица 3.10 – Характеристики водяных насосов

Характеристика	DC30E	LF Bros
1	2	3
Напряжение питания	12 В	12 В
Номинальный ток	0,35 А	0,2 – 1,2 А

Продолжение таблицы 3.10

1	2	3
Производительность	240 л/ч	600 л/ч
Мощность	4,2 Вт	2,4 – 14,4 Вт
Напор	3 м	2 м
Диаметр патрубков	8 мм	16 мм
Размер	56 × 52 × 47 мм	110 × 70 × 120 мм

Был выбран водяной насос DC30E в силу своей оптимальной сочетаемости ключевых параметров с требованиями системы. Его низкое энергопотребление при номинальном токе 0,35 А и мощности 4,2 Вт делает его идеальным выбором для использования в системах, где эффективное управление энергией имеет значение. На фоне его компактных размеров 56 × 52 × 47 мм и сравнительно небольшого диаметра патрубков в 8 мм, насос легко интегрируется в ограниченные пространства без дополнительных модификаций. Кроме того, его производительность в 240 л/ч и напор до 3 м обеспечивают достаточную мощность для эффективной работы перекачивания жидкостей, что делает DC30E идеальным выбором в данном проекте.

Насос для орошения подключен к реле, что обеспечивает управление им через микроконтроллер. Реле позволяет переключать высокое напряжение, подаваемое на насос, в зависимости от условий, определенных программой.

Управление насосом в программе осуществляется через функцию `updateOutputs()` в файле `outputs.cpp`. В этой функции происходит проверка состояния насоса и его включение/выключение в зависимости от текущих условий.

```
void updateOutputs() {
  // Other output controls...

#ifdef PUMP_PORT
  pump.state = (soilMoisture < settings.wateringMoistureThreshold && !enoughWater && !raining) ? 1 : 0;
  digitalWrite(PUMP_PORT, pump.state);
#endif

  // Other output controls...
}
```

Здесь `updateOutputs()` проверяет условия, при которых необходимо включить насос для орошения. Если уровень влажности почвы ниже установленного порога `wateringMoistureThreshold`, и при этом есть достаточно воды и не идет дождь, то насос включается (значение 1), иначе он выключается (значение 0).

Перед включением или выключением насоса происходит установка

соответствующего состояния (`pump.state`) и управление пином, к которому подключен насос (`PUMP_PORT`) с помощью функции `digitalWrite()`.

3.2.9 Блок искусственного освещения

Для блока искусственного освещения необходимо выбрать подходящую LED ленту. Она используется для обеспечения дополнительного освещения в теплице в условиях недостаточного естественного света. LED лента должна иметь достаточную яркость, чтобы обеспечить оптимальные условия для роста растений в течение всего дня. Кроме того, LED лента должна быть энергоэффективной и иметь возможность диммирования для регулировки яркости в зависимости от потребностей растений.

В таблице 3.11 представлены технические характеристики LED ленты [47].

Таблица 3.11 – Характеристики LED ленты

Характеристика	SMD 2835
Тип	Светодиодная лента
Макс. мощность ламп	9,6 Вт
Напряжение	12 В
Количество светодиодов на метр	120
Цвет свечения	Теплый белый
Температура света	3200 К
Степень защиты	IP22

На рисунке 3.16 изображена данная лента [47].

Рисунок 3.16 – LED лента SMD 2835

В программе в функции `updateLCD()` осуществляется отображение информации на LCD-экране, в том числе статуса LED-ленты. При вызове этой функции происходит проверка различных условий, определяющих, нужно ли включать или выключать LED-ленту, и соответствующим образом обновляется ее состояние.

```
void updateLCD() {
    // Код предыдущих операций...

    // Отображение состояния LED-ленты
    lcd.setCursor(14, 3);
    lcd.print(light.state ? "ON " : "OFF");
    lcd.setCursor(18, 3);
    lcd.write(pump.state ? LCD_WATER : ' ');
}
```

В этом участке кода, если переменная `light.state` равна `true`, на LCD-экране будет отображаться «ON», что указывает на то, что LED-лента включена. Если `light.state` равна `false`, на экране будет «OFF», что указывает на выключенное состояние LED-ленты.

```
void updateOutputs() {
  // Код предыдущих операций...

  #ifdef LIGHT_PORT
    digitalWrite(LIGHT_PORT, !light.state);
  #endif
}
```

В функции `updateOutputs()` происходит реальное управление состоянием LED-ленты. В зависимости от значения переменной `light.state` устанавливается соответствующий уровень на выводе, подключенном к драйверу LED-ленты. Если `light.state` равна `true`, то на выходе будет установлен уровень, соответствующий включенному состоянию, и LED-лента будет включена. Если `light.state` равна `false`, то будет установлен уровень, противоположный уровню драйвера, и LED-лента будет выключена.

3.2.10 Блок определения уровня воды

Для блока определения уровня воды необходимо выбрать подходящий датчик уровня воды.

Датчик уровня воды используется для определения наличия воды в резервуаре или емкости, а также контроля за ее уровнем. Это позволяет системе автоматически запускать или останавливать полив растений в зависимости от текущего уровня влажности почвы.

Для выбора датчика уровня воды необходимо учитывать такие факторы, как тип используемого резервуара (например, бочка, бак или емкость), материал, из которого изготовлен резервуар, и его геометрические особенности.

Датчик уровня воды должен быть надежным, точным и долговечным, а также совместимым с другими компонентами системы автоматизации. Он должен обеспечивать стабильную работу в различных условиях эксплуатации

и быть легко интегрируемым в общую систему управления теплицей.

В таблице 3.12 представлены технические характеристики датчика уровня воды [48].

Таблица 3.12 – Характеристики датчика уровня воды

Характеристика	Water sensor
Напряжение питания	3,3 – 5 В
Ток потребления	20 мА
Выход	Аналоговый
Зона обнаружения	16 х 30 мм
Размеры	62 × 20 × 8 мм
Рабочая температура	+10 … +30 °C

На рисунке 3.17 изображен данный датчик уровня воды [48].

Рисунок 3.17 – Датчик уровня воды

3.2.11 Блок определения интенсивности освещения

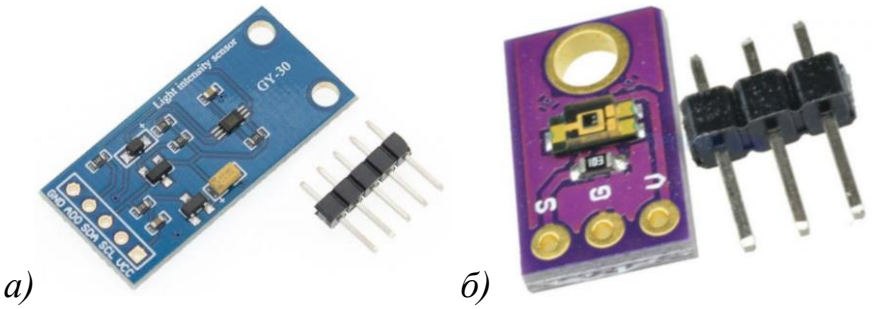
Для блока определения интенсивности освещения предлагается выбрать датчик освещенности, такой как фотодатчик или фоторезистор. Этот датчик будет использоваться для измерения уровня освещенности внутри теплицы. Это важно для определения необходимости включения и выключения искусственного освещения в зависимости от уровня естественного освещения.

В таблице 3.13 представлены технические характеристики некоторых датчиков интенсивности освещения [49, 50].

Таблица 3.13 – Характеристики датчиков интенсивности освещения

Характеристика	GY-30	TEMT6000
Напряжение питания	5 В	3,3 – 5 В
Интерфейс	I2C	–
Чип	BH1750FVI	–
АЦП	16 бит	–
Точность	1 люкс	1 люкс
Диапазон освещенности	0 – 65535 люкс	1 – 1000 люкс
Размеры	19 × 13 × 2 мм	30 × 20 × 18 мм
Вес	5 г	3,5 г

На рисунке 3.18 изображены датчики интенсивности освещения [49, 50].



а) *б)*

Рисунок 3.18 – Датчики интенсивноси освещения:
а – GY-30, *б* – TEMT6000

Датчик освещенности GY-30 выбран из-за его совместимости с напряжением питания Arduino (5 В), интерфейса I2C для удобной связи с микроконтроллером, высокой точности измерений (1 люкс) и широкого диапазона измерений (0-65535 люкс), что обеспечивает точное и надежное измерение уровня освещенности в теплице.

Датчик освещенности будет подключен к аналоговому входу Arduino, и его показания будут считываться для принятия решения о необходимости включения или выключения освещения. Таким образом, благодаря этому датчику будет достигнут баланс между естественным и искусственным освещением в теплице, что важно для оптимального роста растений.

3.2.12 Блок определения влажности почвы

Для определения влажности почвы выбран датчик влажности почвы YL-38 со щупом YL-69. Он необходим для контроля влажности почвы в теплице, что позволяет оптимизировать полив растений и поддерживать необходимый уровень влажности для их здоровья и роста. Датчик влажности почвы состоит из двух электродов, которые вставляются в почву, и измеряет сопротивление почвы, которое изменяется в зависимости от ее влажности. Эти данные затем передаются на микроконтроллер, где они анализируются и используются для принятия решений о поливе растений. В нашем случае, датчик влажности почвы позволяет автоматически управлять поливом растений в теплице, обеспечивая оптимальные условия для их роста и развития.

В таблице 3.14 приведены технические характеристики датчика влажности почвы [51].

Таблица 3.14 – Характеристики датчика влажности почвы

Характеристика	YL-69
1	2
Напряжение питания	3,3 – 5 В
Ток потребления	35 мА
Выход	цифровой и аналоговый

Продолжение таблицы 3.14

1	2
Размер модуля	16 × 30 мм
Размер щупа	20 × 60 мм
Общий вес	7,5 г.

На рисунке 3.19 представлен датчик влажности почвы YL-69 [51].

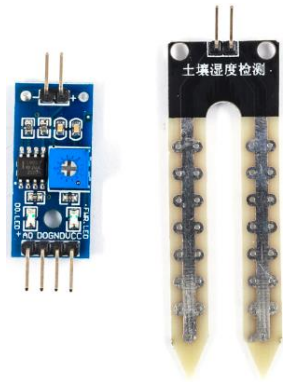


Рисунок 3.19 – Датчик влажности почвы YL-69

3.2.13 Блок определения температуры и влажности воздуха

Для определения температуры и влажности воздуха в теплице и внутри корпуса будет использоваться соответствующие датчики, способные обеспечить точные измерения в обеих средах. Эти датчики позволят контролировать и поддерживать оптимальные условия воздуха внутри теплицы и корпуса, что важно для здоровья и роста растений, а также для надежной работы устройства.

В таблице 3.15 приведены технические характеристики некоторых датчиков температуры и влажности [52, 53, 54].

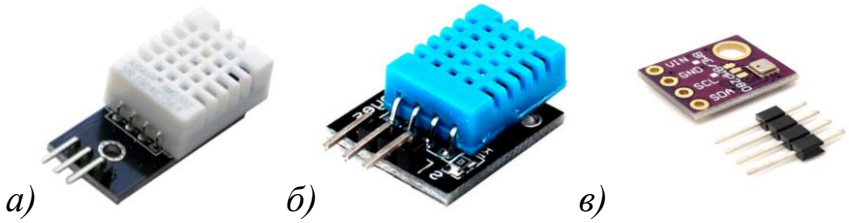
Таблица 3.15 – Характеристики датчиков температуры и влажности

Характеристика	DHT22	DHT11	BME280
1	2	3	4
Напряжение питания	3 – 6 В	3 – 5 В	1,8 – 5 В
Диапазон измерения влажности	0 – 100%	20 – 90%	0 – 100%
Диапазон измерения температуры	-40 … +80 °C	+0 … +50 °C	-40 … +85 °C
Диапазон измерения давления	–	–	300 – 1100 гПа

Продолжение таблицы 3.15

1	2	3	4
Погрешность измерений влажности	2%	5%	3%
Погрешность измерений температуры	0,5°C	2%	0,5 °C
Погрешность измерений давления	–	–	1 гПа
Интерфейс	onewire, 1-проводной	–	I2C (до 3,4 мГц), SPI (до 10 мГц)

На рисунке 3.20 изображены данные датчики [52, 53, 54].



а) *б)* *в)*

Рисунок 3.20 – Датчики температуры и влажности:
а – DHT22, *б* – DHT11, *в* – BME280

Датчик DHT22 был выбран для измерения показаний в теплице благодаря своей надежности и точности. При достижении предельных значений температуры или влажности, определенных в настройках контроллера, система автоматически активирует вентиляцию. Датчик BME280 используется для измерения показаний внутри корпуса контроллера. Его функционал расширен: при обнаружении температуры ниже -20°C или выше +50°C, он автоматически выключает все устройства, подключенные к контроллеру, обеспечивая безопасность и предотвращая возможные повреждения оборудования.

3.2.14 Блок идентификации дождя

Для блока идентификации дождя используется датчик дождя, который предоставляет информацию о наличии или отсутствии осадков. При обнаружении дождя система автоматически принимает соответствующие меры, например, приостанавливает орошение растений, если оно запланировано в этот период времени, или закрыть вентиляционные отверстия для предотвращения попадания влаги в теплицу. Это помогает поддерживать оптимальные условия для растений и предотвращать возможные повреждения от дождевой влаги.

В таблице 3.16 представлены технические характеристики датчика дождя [55].

Таблица 3.16 – Характеристики датчика дождя

Характеристика	MH-RD
Питание	3,3 – 5 В
Потребляемый ток	20 мА
Контроллер	LM393
Размер	50 × 40 мм

На рисунке 3.21 изображен данный датчик [55].

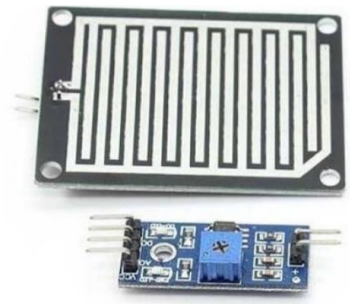


Рисунок 3.21 – Датчик дождя MH-RD

3.2.15 Блок питания

Для обеспечения питания системы требуется три блока питания:

1. Для микроконтроллера и датчиков. Этот блок должен обеспечивать стабильное напряжение 5 В для питания микроконтроллера и всех подключенных датчиков. Подбирается блок питания с достаточной мощностью для обеспечения всех устройств.

2. Для сервоприводов, драйвера привода и реле. Этот блок также должен обеспечивать напряжение 5 В для питания сервоприводов, драйвера привода и всех используемых реле. Также важно убедиться, что блок питания имеет достаточную мощность для обеспечения энергии всем устройствам.

3. Для электропривода, водяной помпы и LED ленты. Этот блок должен обеспечивать напряжение 12 В для питания электропривода, водяной помпы и LED ленты. Подбор блока питания производится с учетом потребляемой мощности этих устройств и обеспечения стабильного 12-вольтового источника питания.

Выбор блока питания на 5 В 2 А для питания логики обусловлен небольшим энергопотреблением микроконтроллеров, датчиков, сервоприводов, реле и прочих устройств. Этот блок обеспечивает достаточный ток для нормальной работы данных устройств и при этом имеет некоторый запас мощности. Такой запас мощности позволяет избежать проблем, связанных с временными пиками потребления энергии, обеспечивая стабильную и бесперебойную работу системы.

В таблице 3.17 представлены технические характеристики данного блока [56].

Таблица 3.17 – Характеристики блока питания 5 В

Характеристика	Блока питания
Тип	Сетевой блок питания
Количество выходных разъемов	1
Входной коннектор	Сетевая вилка
Выходной коннектор	DC 5,5 × 2,5 мм (штекер)
Входное напряжение	220 В
Выходное напряжение	5 В
Макс. выходной ток	2 А

На рисунке 3.22 изображен данный блок питания [56].

Рисунок 3.22 – Блок питания 5 В 2 А

Необходимо рассчитать потребляемый электроприводом (I_{DRIVE}), водяной (I_{PUMP}) помпой и LED лентой (I_{LED}) ток:

$$I = I_{PUMP} + I_{LED} + I_{DRIVE} = 0{,}35 + 0{,}80 + 2{,}30 = 3{,}45 \text{ (А)}.$$

Исходя из полученных результатов, блок питания должен иметь выходной ток не менее 3,45 А. Был выбран блок питания на 12 В и 5 А для стабильной работы системы.

В таблице 3.18 представлены технические характеристики данного блока [57].

Таблица 3.18 – Характеристики блока питания 12 В

Характеристика	Блока питания
1	2
Тип	Сетевой блок питания
Количество выходных разъемов	1
Конструктивные особенности	Защита от перегрева, защита от перегрузки по току
Входной коннектор	Сетевая вилка

Продолжение таблицы 3.18

1	2
Выходной коннектор	DC 5,5 × 2,5 мм (штекер)
Входное напряжение	220 В
Выходное напряжение	12 В
Макс. выходной ток	5 А

На рисунке 3.23 изображен данный блок питания [57].

Рисунок 3.23 – Блок питания 12 В 5 А

3.3 Программные блоки устройства

Комплекс для управления теплицей состоит из следующих аппаратных блоков:

– блок API и обработки команд;
– блок взаимодействия с веб-интерфейсом;
– блок считывания данных с сенсоров;
– блок обновления выходов по данным сенсоров;
– блок обработки ошибок и логирования;
– блок управления выходами;
– блок работы с временем и часами реального времени;
– блок вспомогательных функций;
– блок настроек и инициализации.

Далее будет рассмотрена функциональная часть каждого блока.

3.3.1 Блок API и обработки команд

Файл `esp_api.ino` выполняет роль API для взаимодействия с устройством посредством HTTP-запросов. Взаимодействие осуществляется через обработку различных URI (Uniform Resource Identifier), к каждому из которых привязаны определенные функции обработчики. Подробнее рассмотрим API и обработку команд в этом блоке:

1. Получение данных о состоянии датчиков:
   – URI: `/sensor`;
   – метод: GET;
   – описание: этот запрос позволяет получить данные о текущем

состоянии датчиков;

– функция-обработчик: `getSensors();`

– в ответ возвращается JSON-объект с данными о состоянии датчиков.

2. Получение данных о состоянии устройства:

– URI: `/states`;

– метод: GET;

– описание: запрос для получения данных о состоянии устройства, таких как состояние выходов и режимы работы;

– функция-обработчик: `getStates();`

– в ответ возвращается JSON-объект с данными о состоянии устройства.

3. Установка ручного режима работы:

– URI: `/manual`;

– метод: POST;

– описание: позволяет установить ручной режим управления выходами устройства;

– функция-обработчик: `setManual();`

– данные передаются в виде JSON-объекта в теле запроса (пример JSON-структуры: `{"flap1": true, "flap2": false, "door": true, "light": false}`);

4. Получение текущего ручного режима:

– URI: `/manual`;

– метод: GET;

– описание: запрос для получения текущего ручного режима управления выходами;

– функция-обработчик: `getManual();`

– В ответ возвращается JSON-объект с данными о текущем ручном режиме управления.

5. Установка настроек устройства:

– URI: `/settings`;

– метод: POST;

– описание: позволяет установить настройки устройства;

– функция-обработчик: `setSettings();`

– данные передаются в виде JSON-объекта в теле запроса (пример JSON-структуры: `{"driveSpeed": 125, "sensorPeriod": 1}`);

6. Получение текущих настроек устройства:

– URI: `/settings`;

– метод: GET;

– описание: Запрос для получения текущих настроек устройства;

– функция-обработчик: `getSettings();`

– в ответ возвращается JSON-объект с текущими настройками устройства.

7. Сброс настроек до заводских:

– URI: `/factoryReset`;

– метод: POST;

– описание: позволяет выполнить сброс настроек устройства до заводских значений;

– функция-обработчик: `factoryReset();`

Общий принцип работы:

1. Каждый HTTP-запрос к устройству обрабатывается соответствующей функцией-обработчиком в зависимости от URI и метода запроса.

2. Для запросов POST данные передаются в виде JSON-объекта в теле запроса и обрабатываются функциями `setManual()` и `setSettings()`.

3. Для запросов GET данные возвращаются в виде JSON-объекта в теле ответа и отправляются клиенту.

3.3.2 Блок взаимодействия с веб-интерфейсом

В файле `web.cpp` реализовано взаимодействие с веб-интерфейсом с использованием серийного порта для передачи данных между устройством на базе ESP8266 и веб-интерфейсом.

Функциональное взаимодействие с веб-интерфейсом:

1. Отправка данных о состоянии датчиков и устройства:

– функция `sendSensorValues()` отправляет данные о состоянии датчиков через `Serial` на веб-интерфейс;

– функция `sendStates()` отправляет данные о состоянии устройства через `Serial` на веб-интерфейс.

2. Обработка команд от веб-интерфейса:

– функция `updateServer()` проверяет наличие данных в `Serial` и обрабатывает команды от веб-интерфейса.

3. Получение ответа от устройства и отправка ответа на веб-интерфейс:

– функция `getResponce()` в файле `esp_api.ino` получает ответ от устройства через `Serial` и отправляет его на веб-интерфейс.

4. Обработка HTTP-запросов от клиента:

– функция `setup()` в файле `esp_api.ino` настраивает веб-сервер для обработки HTTP-запросов от клиента.

5. Проверка доступности данных и отправка данных на веб-интерфейс:

– функция `loop()` в файле `esp_api.ino` проверяет наличие данных и отправляет их на веб-интерфейс.

Общий принцип работы:

1. При запуске устройства инициализируются веб-сервер и соединение с WiFi.

2. Данные о состоянии датчиков и устройства периодически отправляются на веб-интерфейс через `Serial`.

3. Веб-интерфейс может отправлять команды устройству через HTTP-запросы.

4. Полученные команды обрабатываются устройством и выполняются соответствующие действия.

5. Ответы от устройства отправляются обратно на веб-интерфейс через `Serial` и отображаются пользователю.

3.3.3 Блок считывания данных с сенсоров

Функционал блока считывания данных с сенсоров реализован в файле `sensors.cpp`. В этом файле содержатся функции, отвечающие за инициализацию сенсоров, чтение значений с сенсоров, а также вспомогательные функции для работы с датчиками и обработки полученных данных.

Функционал блока считывания данных с сенсоров:

1. Инициализация сенсоров:

– функция `initSensors()` инициализирует подключенные датчики и устанавливает соответствующие параметры, такие как режим работы, адреса, и так далее.

2. Чтение значений сенсоров:

– функция `readAllSensors()` периодически считывает данные с подключенных датчиков. Она оптимизирована для работы с несколькими типами датчиков, включая температурные, влажностные, и другие;

– данные о температуре, влажности, уровне освещенности, влажности почвы и других показателях считываются с соответствующих датчиков.

3. Обновление данных и отправка на обработку:

– полученные данные обновляют значения переменных, содержащих информацию о текущих параметрах окружающей среды;

– затем эти данные передаются на обработку блоку обновления выходов для принятия соответствующих решений.

Общий принцип работы:

1. При запуске устройства инициализируются подключенные датчики и устанавливаются необходимые параметры.

2. Регулярно (например, каждую секунду) вызывается функция `readAllSensors()`, которая считывает данные с датчиков. В структуре `Settings` из файла `objects.h` предусмотрена настройка периода опроса датчиков, контролируемая полем `sensorPeriod`.

3. Полученные значения обновляют переменные, содержащие информацию о состоянии окружающей среды.

4. Обновленные данные передаются на обработку блоку обновления выходов, который принимает решения о управлении устройствами на основе текущих условий окружающей среды.

3.3.4 Блок обновления выходов по данным сенсоров

Блок обновления выходов по данным сенсоров содержит логику, которая определяет состояние выходов (например, привода, реле и т. д.) на

основе данных, полученных с сенсоров, и управляет этими выходами в соответствии с заданными параметрами и условиями.

В программе этот функционал реализован в файле `outputs.cpp`. В этом файле содержатся две основные функции:

– функция `initOutputs()` инициализирует выходы, такие как приводы и реле. Например, она устанавливает режимы работы пинов, настраивает ШИМ для приводов и т. д.

– функция `updateOutputs()` вызывается периодически для обновления состояния выходов на основе данных с сенсоров и текущих настроек. В этой функции происходит анализ данных с сенсоров и управление выходами в соответствии с заданными условиями. Например, если температура в помещении превышает заданный порог, то открываются вентиляционные клапаны или включается вентилятор.

3.3.5 Блок обработки ошибок и логирования

Блок обработки ошибок и логирования отвечает за обнаружение и обработку ошибок в программе, а также за ведение журнала событий для отслеживания работы системы и выявления проблем.

Обработка ошибок и логирование в файле `web.cpp`:

```
    static void sendHTTPResponce(uint16_t code, String payload =
"") {
      // Функция отправляет HTTP-ответ
      // с указанным кодом и данными
      Serial.print(code);
      Serial.print(';');
      Serial.println(payload);
    }
```

Эта функция используется для отправки HTTP-ответов с заданным кодом и данными. Она вызывается в различных частях программы для уведомления клиента о результатах выполнения операций.

```
    static void getResponce() {
      // Функция получает HTTP-ответ и отправляет его клиенту
      String codeText = Serial.readStringUntil(';');
      codeText.trim();
      uint16_t code = codeText.toInt();
      String payload = Serial.readStringUntil('\n');
      payload.trim();
      if (payload.isEmpty()) server.send(code);
      else server.send(code, "text/plain", payload);
    }
```

Эта функция считывает HTTP-ответ от сервера и отправляет его клиенту.

В блоке инициализации (setup) устанавливаются маршруты для веб-интерфейса и запускается веб-сервер:

```
void setup() {
  // Инициализация
  Serial.begin(9600);
  wifiMulti.addAP(ssid, password);
  while (wifiMulti.run() != WL_CONNECTED) {
    delay(500);
  }
  // Настройка маршрутов для веб-интерфейса
  server.on("/sensors", HTTP_GET, getSensors);
  server.on("/states", HTTP_GET, getStates);
  server.on("/manual", HTTP_POST, setManual);
  server.on("/manual", HTTP_GET, getManual);
  server.on("/settings", HTTP_POST, setSettings);
  server.on("/settings", HTTP_GET, getSettings);
  server.on("/factoryReset", HTTP_POST, factoryReset);
  // Запуск веб-сервера
  server.begin();
}
```

3.3.6 Блок управления выходами

Блок управления выходами отвечает за контроль состояния различных устройств, таких как клапаны, насосы, освещение и другие. В данной программе этот функционал реализован в файле outputs.cpp.

Основные функции и фрагменты кода из этого файла, демонстрирующие управление выходами:

```
void initOutputs() {
  pinMode(LIGHT_PORT, OUTPUT);
  pinMode(PUMP_PORT, OUTPUT);
  pinMode(DRV_PWM, OUTPUT);
  pinMode(DRV_SIGNAL1, OUTPUT);
  pinMode(DRV_SIGNAL2, OUTPUT);
  servo1.attach(SERVO1, SERVO_MIN_PULSE, SERVO_MAX_PULSE);
  servo2.attach(SERVO2, SERVO_MIN_PULSE, SERVO_MAX_PULSE);
}

void updateOutputs() {
  // Обновление состояния выходов на основе данных с датчиков
  // Управление открытием и закрытием клапанов,
  // Включение и выключение насосов и освещения и т. д.
}
```

Функция initOutputs() выполняет настройку пинов для работы выходных устройств при запуске программы. В свою очередь, функция updateOutputs() обновляет состояние выходов на основе данных с датчиков и текущих настроек.

3.3.7 Блок работы с временем и часами реального времени

Блок работы с временем и часами реального времени (RTC) в программе отвечает за отслеживание времени, обновление его значений и синхронизацию с внешними часами реального времени. Этот блок реализован в файле `sensors.cpp`.

Функция `updateTime()` обновляет текущее время и счетчик времени работы системы на основе встроенного счетчика времени Arduino. Функция `initSensors()` инициализирует все необходимые датчики, включая часы реального времени, при запуске программы. Функция `readAllSensors()` занимается считыванием данных с датчиков, таких как температура, влажность, уровень освещенности и так далее, а также обновляет значения времени.

Эти функции обеспечивают корректную работу системы, управляя данными с датчиков и поддерживая актуальное время.

3.3.8 Блок вспомогательных функций

Блок вспомогательных функций в программе предоставляет различные вспомогательные функции, которые используются для упрощения и оптимизации кода. В этом блоке содержатся различные функции для обработки данных, преобразования типов и выполнения математических операций:

– функция `charToDec(const char p)` в файле `sensors.cpp` преобразует строку в целое число;

– функция `analogReadAverage(uint8_t pin)` в файле `sensors.cpp` выполняет несколько измерений аналогового пина и возвращает среднее значение;

– функция `stringToBool(const String& str)` в файле `web.cpp` преобразует строку в логическое значение;

– функция `readJsonTable()` в файле `web.cpp` читает JSON-таблицу из потока данных и возвращает ее в виде строки;

– функция `getResponce()` в файле `web.cpp` считывает ответ от устройства и отправляет его клиенту;

– функция `checkSerial()` в файле `web.cpp` проверяет серийный порт на наличие данных и обрабатывает полученные команды.

Эти функции предоставляют важные инструменты для работы программы, делая код более читаемым, эффективным и модульным. Они используются для различных целей, от преобразования данных до обработки команд и взаимодействия с внешними устройствами.

3.3.9 Блок настроек и инициализации

Блок настроек и инициализации включает в себя все необходимые

действия по установке начальных параметров устройства и инициализации его компонентов. Этот блок отвечает за настройку подключения к сети, инициализацию выходов и установку начальных значений настроек. В этом разделе содержатся структуры данных, определяющие параметры работы устройства, а также функции и методы для их установки и сохранения.

Функционал блока настроек и инициализации:

1. `Settings struct` – структура, содержащая настройки устройства.

Поля:

– `magic` – магическое число для проверки целостности данных;

– `driveSpeed` – скорость привода;

– `sensorPeriod` – период считывания данных с сенсоров;

– `flapsTemperatureThreshold` – пороговая температура для открытия заслонок;

– `doorTemperatureThreshold` – пороговая температура для открытия двери;

– `ventelationPeriod` - период вентиляции;

– `ventelationTime` – время вентиляции;

– `closeIfRain` – флаг, указывающий на закрытие заслонок и двери при дожде;

– `wateringMoistureThreshold` – порог влажности почвы для полива;

– `wateringTime` – время полива;

– `wateringTimeout` – таймаут после полива;

– `wateringPeriod` – период полива;

– `lightLevelThreshold` – пороговый уровень освещенности.

Файлы: `settings.h`.

2. `MAGIC`, `SETTINGS_ADDRESS` – константы для проверки целостности данных и адреса в EEPROM.

Файлы: `settings.h`.

3. `EEPROM.put(SETTINGS_ADDRESS, settings)` – записывает настройки в EEPROM.

Файлы: `sensors.cpp`, `web.cpp`.

4. `initOutputs()` – инициализирует выходы устройства.

Файлы: `outputs.cpp`.

5. `initSensors()` – инициализирует датчики и RTC.

Файлы: `sensors.cpp`.

6. `setup()` – настройка устройства при запуске.

Файлы: `web.cpp`.

7. `wifiMulti.addAP(ssid, password)` – добавляет точку доступа Wi-Fi для мультиподключения.

Файлы: `esp_api.ino`.

8. `server.begin()` – запускает веб-сервер.

Файлы: `esp_api.ino`.

4 ПРИНЦИПИАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе выполняется детальный анализ практической реализации аппаратной части устройства. На схеме электрической принципиальной (чертеж ГУИР.400201.022 Э3) представлен состав устройства. Также в данном разделе представлено описание каждого элемента схемы, включая описание контактов и соединение между собой данных элементов.

4.1 Блок контроллера теплицы

В рассматриваемом блоке используется плата микроконтроллера Nano V3.0 ATmega328P, которая представлена на схеме микросхемой DD3, к которой выполняется подключение большинства модулей принципиальной схемы. Описание контактов данного модуля приведено в таблице 4.1.

Таблица 4.1 – Описание контактов модуля Nano V3.0 ATmega328P

Выход	Название	Тип	Описание
1	D1/TX	I/O	Цифровой порт ввода/вывода
2	D0/RX	I/O	
3, 18	RST	Input	Сброс (активный низкий уровень)
4, 17	GND	PWR	Земля питания
5 – 15	D2 – D12	I/O	Цифровой порт ввода/вывода
16	VIN	PWR	Напряжение питания
19	+5V	I/O	Выход +5 В (от встроенного регулятора) или +5 В (вход от внешнего источника питания)
20 – 27	A7 – A0	Input	Аналоговый входной канал
28	AREF	Input	Ссылка на АЦП
29	3V3	Output	Выход +3,3 В (от FTDI)
30	D13	I/O	Цифровой порт ввода/вывода

Данный блок также включает в себя модуль RTC часов реального времени DS3231 (AT24C32), представленный на схеме микросхемой DD4. Описание контактов данного модуля приведено в таблице 4.2.

Таблица 4.2 – Описание контактов модуля DS3231

Выход	Название	Описание
1	VCC	Напряжение питания +5 В
2	SDA	Последовательный ввод/вывод данных для протокола I2C
3	SCL	Последовательный тактовый вход для протокола I2C
4	NC	Не используется
5	GND	Земля питания

Модуль DS3231 подключается к плате микроконтроллера (DD3) через шину I2C. Исходя из информации в таблице 4.2, соединение с платой микроконтроллера (DD3) осуществляется следующим образом:

– VCC – шина +5 В устройства;
– SDA – выход A4 платы микроконтроллера (DD3);
– SCL – выход A5 платы микроконтроллера (DD3);
– NC – шина +5 В устройства;
– GND – общая земля.

4.2 Блок беспроводной связи

В состав данного блока входит модуль беспроводной связи ESP-01, который представлен на схеме микросхемой DD1. Описание контактов данного модуля приведено в таблице 4.3.

Таблица 4.3 – Описание контактов модуля ESP-01

Выход	Название	Описание
1	GND	Земля питания
2	TXD	Контакт передачи данных по UART
3	GPIO2	Универсальный контакт ввода/вывода с поддержкой GPIO
4	CH_PD	Включение чипа (активный высокий уровень)
5	GPIO0	Универсальный контакт ввода/вывода с поддержкой GPIO
6	RST	Сброс (активный низкий уровень)
7	RXD	Контакт получения данных по UART
8	VCC	Напряжение питания 3,3 В

Модуль ESP-01 питается от 3,3 В, поэтому для подключения к плате микроконтроллера (DD3) используется стабилизатор напряжения AMS1117, представленный микросхемой DA1. Описание контактов данного модуля приведено в таблице 4.4.

Таблица 4.4 – Описание контактов модуля ESP-01

Выход	Название	Описание
1	VIN	Входное напряжение
2	OUT	Выходное напряжение
3	GND	Земля питания / Регулировка напряжения

Модуль ESP-01 подключается к плате микроконтроллера (DD3) через стабилизатор напряжения AMS1117 (DA1). Исходя из информации в таблицах 4.3 и 4.4, соединение ESP-01 (DD1) с платой микроконтроллера (DD3) осуществляется следующим образом:

– GND – общая земля;

– TXD – выход D0/RX платы микроконтроллера (DD3);
– GPIO2 – не подключается;
– CH_PD – выход OUT стабилизатора напряжения (DA1);
– GPIO0 – не подключается;
– RST – не подключается;
– RXD – выход D1/TX платы микроконтроллера (DD3);
– VCC – выход OUT стабилизатора напряжения (DA1).
Подключение AMS1117 (DA1) выполняется следующим образом:
– VIN – шина +5 В устройства;
– OUT – выходы VCC и CH_PD модуля ESP-01 (DD1);
– GND – общая земля.

4.3 Блок управления контроллером

В состав данного блока входит модуль энкодера EC11, который представлен на схеме микросхемой DD6. Описание контактов данного модуля приведено в таблице 4.5.

Таблица 4.5 – Описание контактов модуля энкодера EC11

Выход	Название	Описание
1	5V	Напряжение питания +5 В
2	Key	Внутренняя кнопка энкодера
3	S2	Контакты, которые обеспечивают сигнал вращения относительно третьего контакта (GND)
4	S1	
5	GND	Земля питания

В состав блока также входит кнопка управления, обозначенная на схеме как SW1. Данная кнопка необходима для открытия и закрытия дверей теплицы. SW1 подключается между выходами D0/RX и GND платы микроконтроллера (DD3).

Исходя из информации в таблице 4.5, соединение модуля EC11 (DD6) с платой микроконтроллера (DD3) осуществляется следующим образом:
– 5V – шина +5 В устройства;
– Key – выход D0/RX платы микроконтроллера (DD3);
– S2 – выход D2 платы микроконтроллера (DD3);
– S1 – выход D3 платы микроконтроллера (DD3);
– GND – общая земля.

4.4 Блок отображения информации

В состав данного блока входит дисплей LCD2004, который представлен на схеме микросхемой HG1. Для подключения LCD2004 (HG1) к плате микроконтроллера (DD3) по I2C используется модуль расширения портов HW-061 (микросхема DD5). Описание контактов LCD2004 приведено в таблице 4.6.

Таблица 4.6 – Описание контактов дисплея LCD2004

Выход	Название	Описание
1	VSS	Питание дисплея (земля питания)
2	VDD	Питание дисплея (напряжение питания +5 В)
3	VO	Контрастность дисплея
4	RS	Выбор регистра: команда / данные
5	RW	Выбор режима: запись / чтение
6	E	Линия тактирования
7 – 14	D0 – D7	Шина данных
15	A	Питание подсветки (напряжение питания +5 В)
16	K	Питание подсветки (земля питания)

Описание контактов модуля HW-061 приведено в таблице 4.7.

Таблица 4.7 – Описание контактов модуля HW-061

Выход	Название	Описание
1 – 16	–	Соответствующие контакты дисплея (HG1)
17	SDA	Линия данных I2C
18	SCL	Линия тактирования I2C
19	VCC	Напряжение питания +5 В
20	GND	Земля питания

Дисплей LCD2004 (HG1) подключается к соответствующим контактам модуля HW-061 (DD5). Исходя из информации в таблицах 4.6 и 4.7, соединение модуля HW-061 (DD5) с платой микроконтроллера (DD3) выполняется следующим образом:

– SDA – выход A4 платы микроконтроллера (DD3);
– SCL – выход A5 платы микроконтроллера (DD3);
– VCC – шина +5В устройства;
– GND – общая земля.

4.5 Блок управления вентиляцией

В состав данного блока входят сервоприводы SG-90, которые представлены на схеме микросхемами M3 и M4. Описание контактов SG-90 приведено в таблице 4.8.

Таблица 4.8 – Описание контактов сервопривода SG-90

Выход	Название	Описание
1	VCC	Напряжение питания +5 В
2	PWM	Сигнал управления
3	GND	Земля питания

Исходя из информации в таблице 4.8, соединение сервопривода SG-90

(M3) с платой микроконтроллера (DD3) выполняется следующим образом:

– VCC – шина +5 В устройства;

– PWM – выход A0 платы микроконтроллера (DD3);

– GND – общая земля.

Соединение сервопривода SG-90 (M4) с платой микроконтроллера (DD3) выполняется следующим образом:

– VCC – шина +5 В устройства;

– PWM – выход D13 платы микроконтроллера (DD3);

– GND – общая земля.

4.6 Блок управления доступом в теплицу

В состав данного блока входит линейный привод 750N, который представлен на схеме микросхемой M2. Описание контактов линейного привода приведено в таблице 4.9.

Таблица 4.9 – Описание контактов линейного привода 750N

Выход	Название	Описание
1	VCC	Напряжение питания +12 В
2	GND	Земля питания

Линейный привод 750N подключается и управляется через драйвер двигателей TB6612FNG Ultra L298N, представленный на схеме микросхемой DD2. Описание контактов драйвера TB6612FNG Ultra L298N приведено в таблице 4.10.

Таблица 4.10 – Описание контактов драйвера TB6612FNG Ultra L298N

Выход	Название	Описание
1	2	3
1	PWMA	Управление скоростью вращения мотора канала A
2	AIN2	Входы полумостов канала A
3	AIN1	
4	SBBY	Включение микросхемы
5	BIN1	Входы полумостов канала A
6	BIN2	
7	PWMB	Управление скоростью вращения мотора канала B
8	GND	Земля питания
9	VM	Вход питания силовой части микросхемы, питание двигателей
10	VCC	Вход питания логической части микросхемы (напряжение питания +5 В)
11	GND	Земля питания
12	AO1	Выходы полумостов канала A, подключается коллекторный двигатель
13	AO2	

Продолжение таблицы 4.10

1	2	3
14	BO2	Выходы полумостов канала B, подключается коллекторный двигатель
15	BO1	
16	GND	Земля питания

Исходя из информации в таблицах 4.9 и 4.10, подключение драйвера (DD2) выполняется следующим образом:

– PWMA – не подключается;
– AIN2 – не подключается;
– AIN1 – не подключается;
– STBY – шина +5 В устройства;
– BIN1 – выход D10 платы микроконтроллера (DD3);
– BIN2 – выход D12 платы микроконтроллера (DD3);
– PWMB – выход D11 платы микроконтроллера (DD3);
– GND (контакт 8) – не подключается;
– VM – шина +12 В устройства;
– VCC – шина +5 В устройства;
– GND (контакт 11) – общая земля;
– AO1 – не подключается;
– AO2 – не подключается;
– BO2 – выход VCC линейного привода (M2);
– BO1 – выход GND линейного привода (M2);
– GND (контакт 16) – общая земля.

4.7 Блок реле

В состав данного блока входит двухканальный модуль реле, который представлен на схеме микросхемой KM1. Описание контактов данного модуля реле приведено в таблице 4.11.

Таблица 4.11 – Описание контактов модуля реле

Выход	Название	Описание
1	2	3
1	VCC	Вход питания логической части модуля (напряжение питания +5 В)
2	IN1	Контакты для подключения управляющего сигнала
3	IN2	
4	GND	Земля питания
5	GND	Земля питания
6	VCC	Вход питания логической части модуля (напряжение питания +5 В)
7	JD-VCC	Контакт для подачи внешнего питания, когда требуется гальваническая развязка

Продолжение таблицы 4.11

1	2	3
8, 11	NC1, NC2	Нормально замкнутый контакт
9, 12	K1, K2	Общий контакт
10, 13	NO1, NO2	Нормально разомкнутый контакт

Исходя из информации в таблице 4.11, подключение модуля реле (KM1) выполняется следующим образом:

– VCC (контакт 1) – шина +5 В устройства;
– IN1 – выход D6 платы микроконтроллера (DD3);
– IN2 – выход D5 платы микроконтроллера (DD3);
– GND (контакт 4) – общая земля;
– GND (контакт 5) – общая земля;
– VCC (контакт 6) – выход JD-VCC модуля реле (KM1);
– NC1 – выход VCC водяной помпы (M1);
– K1 – шина +12В устройства;
– NO1 – не подключается;
– NC2 – выход VCC LED ленты (EL1);
– K2 – шина +12 В устройства;
– NO2 – не подключается.

4.8 Блок системы орошения

В состав данного блока входит водяная помпа DC30E, которая представлена на схеме микросхемой M1. Подключение и управление водяной помпой производится через модуль реле. Описание контактов водяной помпы приведено в таблице 4.12.

Таблица 4.12 – Описание контактов водяной помпы DC30E

Выход	Название	Описание
1	VCC	Напряжение питания +12 В
2	GND	Земля питания

Исходя из информации в таблице 4.12, подключение водяной помпы (M1) выполняется следующим образом:

– VCC – выход NC1 модуля реле (KM1);
– GND – общая земля.

4.9 Блок искусственного освещения

В состав данного блока входит LED лента SMD2835, которая представлена на схеме микросхемой EL1. Подключение и управление LED лентой выполняется через модуль реле. Описание контактов LED ленты приведено в таблице 4.13.

Таблица 4.13 – Описание контактов LED ленты SMD2835

Выход	Название	Описание
1	VCC	Напряжение питания +12 В
2	GND	Земля питания

Исходя из информации в таблице 4.13, подключение LED ленты (EL1) осуществляется следующим образом:

– VCC – выход NC2 модуля реле (KM1);

– GND – общая земля.

4.10 Блок определения уровня воды

В состав данного блока входит модуль датчика уровня воды, который представлен на схеме микросхемой B3. Описание контактов данного датчика приведено в таблице 4.14.

Таблица 4.14 – Описание контактов модуля датчика уровня воды

Выход	Название	Описание
1	VCC	Напряжение питания +5 В
2	Signal	Аналоговый выход
3	GND	Земля питания

Исходя из информации в таблице 4.14, подключение датчика уровня воды (B3) выполняется следующим образом:

– VCC – шина +5 В устройства;

– Signal – выход A6 платы микроконтроллера (DD3)

– GND – общая земля.

4.11 Блок определения интенсивности освещения

В состав данного блока входит модуль датчика интенсивности освещения GY-30, который представлен на схеме микросхемой B1. Описание контактов данного модуля приведено в таблице 4.15.

Таблица 4.15 – Описание контактов модуля GY-30

Выход	Название	Описание
1	VCC	Напряжение питания +5 В
2	SCL	Линия тактирования I2C
3	SDA	Линия данных I2C
4	AD0	Адрес устройства на шине I2C
5	GND	Земля питания

Исходя из информации в таблице 4.15, подключение модуля GY-30 (B1) выполняется следующим образом:

– VCC – шина +5 В устройства;
– SCL – выход A5 платы микроконтроллера (DD3);
– SDA – выход A4 платы микроконтроллера (DD3);
– AD0 – не подключается;
– GND – общая земля.

4.12 Блок определения влажности почвы

В состав данного блока входит модуль датчика влажности почвы YL-38, который представлен на схеме микросхемой B5. Описание контактов данного модуля приведено в таблице 4.16.

Таблица 4.16 – Описание контактов модуля YL-38

Выход	Название	Описание
1	VCC	Напряжение питания +5 В
2	GND	Земля питания
3	DO	Цифровой выход
4	AO	Аналоговый выход
5	+	Выход для подключения к контактному щупу
6	–	Выход для подключения к контактному щупу

Данный модуль подключен к соответствующим выходам контактного щупа YL-69 (микросхема B7), который измеряет влажность почвы.

Исходя из информации в таблице 4.16, подключение модуля YL-38 (B5) выполняется следующим образом:

– VCC – шина +5 В устройства;
– GND – общая земля;
– DO – не подключается;
– AO – выход A7 платы микроконтроллера (DD3);
– «+» – соответствующий выход контактного щупа (B7);
– «–» – соответствующий выход контактного щупа (B7).

4.13 Блок определения температуры и влажности воздуха

В состав данного блока входит модуль датчика температуры и влажности воздуха DHT22 (микросхема B2) и модуль датчика температуры, влажности и давления BME280 (микросхема B4). Описание контактов модуля DHT22 (B2) приведено в таблице 4.17.

Таблица 4.17 – Описание контактов модуля DHT22

Выход	Название	Описание
1	VCC	Напряжение питания +5 В
2	OUT	Аналоговый выход
3	GND	Земля питания

Описание контактов модуля BME280 (B4) приведено в таблице 4.18.

Таблица 4.18 – Описание контактов модуля BME280

Выход	Название	Описание
1	VCC	Напряжение питания +5 В
2	GND	Земля питания
3	SCL	Линия тактирования I2C
4	SDA	Линия данных I2C

Исходя из информации в таблице 4.17, подключение модуля DHT22 (B2) выполняется следующим образом:

– VCC – шина +5 В устройства;

– OUT – выход A2 платы микроконтроллера (DD3);

– GND – общая земля.

Исходя из информации в таблице 4.18, подключение модуля BME280 (B4) выполняется следующим образом:

– VCC – шина +5 В устройства;

– GND – общая земля;

– SCL – выход A5 платы микроконтроллера (DD3);

– SDA – выход A4 платы микроконтроллера (DD3).

4.14 Блок идентификации дождя

В состав данного блока входит модуль датчика дождя YL-38, который представлен на схеме микросхемой B6. Описание контактов данного модуля приведено в таблице 4.19.

Таблица 4.19 – Описание контактов модуля YL-38

Выход	Название	Описание
1	VCC	Напряжение питания +5 В
2	GND	Земля питания
3	DO	Цифровой выход
4	AO	Аналоговый выход
5	+	Выход для подключения к контактному щупу
6	–	Выход для подключения к контактному щупу

Данный модуль подключен к соответствующим выходам контактного щупа MH-RD (микросхема B8), который определяет наличие дождя.

Исходя из информации в таблице 4.19, подключение модуля YL-38 (B6) выполняется следующим образом:

– VCC – шина +5 В устройства;

– GND – общая земля;

– DO – выход A3 платы микроконтроллера (DD3);

– AO – не подключается;

– «+» – соответствующий выход контактного щупа (B8);
– «–» – соответствующий выход контактного щупа (B8).

4.15 Блок питания

Данный блок состоит из трех блоков питания: блок питания 12 В 5 А (XP1) и 2 блока питания 5 В 2 А (XP2 и XP3).

Блок питания XP1 обеспечивает питание следующим устройствам:
– драйвер привода L298N (DD2);
– модуль реле (KM1).

Блок питания XP2 обеспечивает питание следующим устройствам:
– драйвер привода L298N (DD2);
– модуль реле (KM1);
– сервоприводы SG-90 (M3 и M4).

Блок питания XP3 обеспечивает питание следующим устройствам:
– плата микроконтроллера Nano V3.0 (DD3);
– стабилизатор напряжения AMS1117 (DA1);
– дисплей LCD2004 (HG1);
– модуль BME280 (B4);
– модуль DS3231 (DD4);
– энкодер EC11 (DD6).

4.16 Разводка печатной платы

Исходя из результатов проектирования, была разработана и разведена печатная плата, представляющая собой носитель всех компонентов и соединений, необходимых для функционирования устройства. Этот процесс эффективно осуществлен благодаря электромонтажному чертежу ГУИР.400201.005 МЭ, который детально отображает размещение всех элементов и трассировку маршрутов соединений. Процесс разработки печатной платы требует аккуратности и внимательности для обеспечения правильного соединения между компонентами и минимизации возможных ошибок при монтаже. После успешного завершения этапа разведения платы достигается оптимизация пространства и обеспечивается эффективное взаимодействие всех элементов устройства, что является важным шагом в создании функционального и надежного устройства.

5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

В этом разделе подробно описываются тесты, проведенные для проверки функциональности API программного обеспечения для Arduino и ESP-01, а также ожидаемые и полученные результаты. Цель испытаний – убедиться в корректной работе всех функций и интерфейсов, а также в устойчивости и надежности системы. Схемы программы для Arduino и ESP-01 представлены на чертежах ГУИР.400201.005 ПД.1 и ГУИР.400201.005 ПД.2 соответственно.

5.1 Общие сведения

Тестирование проводилось в следующих условиях:

1. Arduino подключен к датчикам и исполнительным механизмам в теплице.
2. ESP-01 использован для связи с Arduino и обработки HTTP-запросов через Wi-Fi.
3. Последовательный порт используется для обмена данными между Arduino и ESP-01.
4. Тестирование проводилось с использованием программы Postman для отправки HTTP-запросов к API и анализа ответов.
5. Использовалась стандартная прошивка для обоих микроконтроллеров без дополнительных модификаций.
6. Испытания проводились в контролируемых условиях, имитирующих рабочую среду теплицы.

5.2 Тестирование API

В этом разделе будет рассмотрено тестирование различных аспектов API для Arduino и ESP-01. Для каждой категории тестов будет описан метод тестирования, ожидаемые результаты и фактические результаты. Основные направления тестирования включают:

– тестирование работы с датчиками;
– тестирование управления состояниями;
– тестирование получения и установки настроек системы;
– тестирование функций управления вручную;
– тестирование сброса настроек к заводским

5.2.1 Тестирование работы с датчиками

Описание: проверка получения значений датчиков через запрос к ESP-01.

Методика:

1. Отправить GET-запрос к API по адресу `/sensors`.
2. Ожидать получение текущих значений датчиков температуры,

влажности и других параметров.

Ожидаемый результат: сервер возвращает JSON-объект с текущими значениями датчиков.

Полученный результат:

```
{
  "airT": 25.3,
  "airH": 60,
  "insideT": 24.8,
  "lightLevel": 75,
  "soilMoisture": 40,
  "enoughWater": true,
  "raining": false
}
```

Описание каждого ключа в полученном результате:

– `airT` – температура воздуха в теплице в градусах Цельсия;

– `airH` – влажность воздуха в теплице в процентах;

– `insideT` – температура внутри корпуса в градусах Цельсия;

– `lightLevel` – уровень освещенности в теплице люксах;

– `soilMoisture` – влажность почвы в теплице в процентах;

– `enoughWater` – статус достаточности воды для полива (`true` – достаточно, `false` – недостаточно);

– `raining` – статус осадков (`true` – идет дождь, `false` – нет дождя).

Тест успешно завершен. Полученные значения датчиков соответствуют ожидаемым результатам. Теплица находится в рабочем состоянии, температура, влажность воздуха, освещенность и влажность почвы измеряются и передаются корректно. Система также корректно определяет наличие осадков и необходимость полива.

5.2.2 Тестирование управления состояниями

Описание: проверка получения текущих состояний исполнительных механизмов через запрос к ESP-01.

Методика:

1. Отправить GET-запрос к API по адресу `/states`.

2. Ожидать получение текущих состояний механизмов (водяной насос, двери, свет и т.д.).

Ожидаемый результат: сервер возвращает JSON-объект с текущими состояниями устройств.

Полученный результат:

```
{
  "flap1": 0,
  "flap2": 1,
  "door": true,
```

```
  "light": false,
  "pump": false
}
```

Описание каждого ключа в полученном результате:

– `flap1` и `flap2` – положение сервоприводов в градусах (значение 0 соответствует полностью закрытому состоянию, а 180 – полностью открытому);

– `door` – состояние двери теплицы (`true` означает, что дверь закрыта, а `false` – открыта);

– `light` – состояние освещения в теплице (`true` означает, что освещение включено, а `false` – выключено);

– `pump` – состояние водяного насоса (`true` означает, что насос включен, а `false` – выключен).

Полученные данные с датчиков соответствуют ожидаемым. Тест пройден успешно.

5.2.3 Тестирование получения настроек системы

Описание: проверка получения текущих настроек системы через запрос к ESP-01.

Методика:

1. Отправить GET-запрос к API по адресу `/settings`.

2. Ожидать получение текущих настроек системы.

Ожидаемый результат: сервер возвращает JSON-объект с текущими настройками.

Полученный результат:

```
{
  "driveSpeed": 255,
  "sensorPeriod": 1,
  "flapsTemperatureThreshold": 22.5,
  "doorTemperatureThreshold": 15.0,
  "ventelationPeriod": 600,
  "ventelationTime": 60,
  "closeIfRain": true,
  "wateringMoistureThreshold": 30,
  "wateringTime": 120,
  "wateringTimeout": 600,
  "wateringPeriod": 3600,
  "lightLevelThreshold": 70
}
```

Описание каждого ключа в полученном результате:

– `driveSpeed` – значение скорости привода, указанное в диапазоне от 0 до 255;

– `sensorPeriod` – периодичность считывания данных с датчиков,

выраженная в секундах;

– `flapsTemperatureThreshold` – пороговое значение температуры для автоматического управления клапанами;

– `doorTemperatureThreshold` – пороговое значение температуры для автоматического управления дверью теплицы;

– `ventelationPeriod` – периодичность работы вентиляции, выраженная в секундах;

– `ventelationTime` – время работы вентиляции в одном цикле, выраженное в секундах;

– `closeIfRain` – флаг, указывающий, следует ли закрывать дверь теплицы в случае дождя (`true` – да, `false` – нет);

– `wateringMoistureThreshold` – пороговое значение влажности почвы, при котором производится автоматический полив;

– `wateringTime` – время полива, выраженное в секундах;

– `wateringTimeout` – таймаут между автоматическими поливами, выраженный в секундах;

– `wateringPeriod` – периодичность автоматического полива, выраженная в секундах;

– `lightLevelThreshold` – пороговое значение уровня освещенности, при котором включается дополнительное освещение.

В ходе тестирования были получены текущие значения настроек системы, представленные в формате JSON. Каждый ключ в JSON-объекте соответствует определенному параметру настроек, позволяя управлять различными аспектами функционирования системы автоматизации теплицы. Полученные значения соответствуют ожидаемым, что подтверждает корректную работу API и правильную настройку системы.

5.2.4 Тестирование установки настроек системы

Описание: проверка установки новых значений настроек системы через POST-запрос к ESP-01.

Методика:

1. Отправить POST-запрос к API по адресу `/settings` с JSON-объектом, содержащим новые значения настроек.

2. Ожидать успешного ответа от сервера.

Пример запроса:

```
{
  "driveSpeed": 15,
  "sensorPeriod": 400
}
```

Ожидаемый результат: сервер возвращает HTTP-код 200, настройки системы обновлены.

Полученный результат:

– код ответа: 200;

– настройки системы обновлены согласно запросу.

Тест пройден успешно. Полученный HTTP-код 200 указывает на успешное обновление настроек системы согласно отправленному запросу.

5.2.5 Тестирование сброса настроек к заводским

Описание: проверка сброса всех настроек к заводским значениям через POST-запрос к ESP-01.

Методика:

1. Отправить POST-запрос к API по адресу `/factoryReset`.
2. Ожидать успешного ответа от сервера.

Ожидаемый результат: сервер возвращает HTTP-код 200, все настройки сброшены к заводским значениям.

Полученный результат:

– код ответа: 200;

– настройки системы сброшены.

Тест пройден успешно. Полученный HTTP-код 200 указывает на успешный сброс всех настроек к заводским значениям.

5.3 Тестирование локального управления

В данном разделе описаны тесты, проведенные для проверки функциональности локального управления с использованием энкодера и отображения данных на ЖК-дисплее.

5.3.1 Проверка инициализации ЖК-дисплея

Описание: проверка корректного запуска и инициализации ЖК-дисплея при включении системы.

Методика:

1. Подключить питание Arduino.
2. Наблюдать за LCD-дисплеем.

Ожидаемый результат: дисплей должен инициализироваться и отобразить начальные данные.

Полученный результат: дисплей инициализировался и первую страницу меню.

5.3.2 Проверка обновления данных на ЖК-дисплее

Описание: проверка правильности обновления данных на дисплее по мере изменения состояния системы.

Методика:

1. Систематически изменять значения датчиков (например,

температуру, влажность).

2. Наблюдать за обновлением данных на LCD-дисплее.

Ожидаемый результат: данные на дисплее должны обновляться в реальном времени, отображая текущие значения датчиков.

Полученный результат: данные на дисплее обновляются корректно, отображая текущие значения датчиков.

На рисунке 5.1 представлено отображение данных на ЖК-дисплее.

Рисунок 5.1 – Отображение данных на ЖК-дисплее

5.3.3 Проверка управления энкодером

Описание: проверка корректной работы энкодера для навигации по меню и изменения настроек.

Методика:

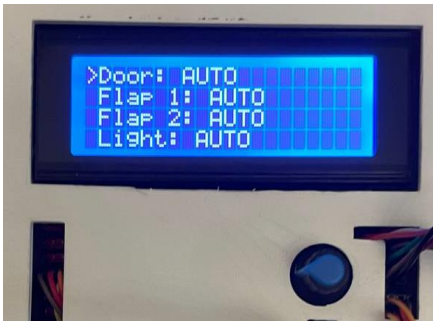
1. Вращать энкодер для перемещения по меню.

2. Нажимать на энкодер для выбора пунктов меню.

Ожидаемый результат: навигация по меню должна происходить плавно, нажатие на энкодер должно вызывать соответствующее действие.

Полученный результат: навигация по меню с помощью энкодера работает корректно, нажатие на энкодер вызывает ожидаемое действие.

На рисунке 5.2 представлено изменение энкодером состояния подключенных устройств.

а) 

б) 

Рисунок 5.2 – Изменение состояния подключенных устройств:
а – до изменения состояния; *б* – после изменения состояния

5.3.3 Проверка кнопки для управления линейным приводом

Описание: проверка работы кнопки, управляющей линейным приводом, для открытия и закрытия дверей.

Методика:

1. Зажать кнопку или энкодер для запуска линейного привода.
2. Проверить, что линейный привод выполняет команды на открытие или закрытие дверей.

Ожидаемый результат: линейный привод корректно выполняет команды на открытие или закрытие.

Полученный результат: линейный привод корректно реагирует на нажатие кнопки, открывая или закрывая двери.

5.3.4 Проверка отображения критических уведомлений

Описание: проверка отображения критических уведомлений на дисплее (критически высокая или низкая температура).

Методика:

1. Симулировать ситуацию критически высокой или низкой температуры.
2. Наблюдать за отображением уведомления на дисплее.

Ожидаемый результат: при достижении критической температуры дисплей должен отображать предупреждающее сообщение «Critical Temp!».

Полученный результат: дисплей корректно отображает предупреждающее сообщение при достижении критической температуры.

5.4 Подведение итогов

Тестирование API и локального управления подтвердило корректную работу всех компонентов системы. API обеспечивает надежное взаимодействие между клиентом и устройством, позволяя эффективно управлять и контролировать систему теплицы. Локальное управление через энкодер и ЖК-дисплей также функционирует без сбоев, обеспечивая удобный и интуитивно понятный интерфейс для пользователя.

Все проведенные тесты были успешны, что подтверждает надежность и функциональность разработанного программного обеспечения для управления теплицей.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

В этом разделе рассматриваются инструкции по установке, настройке и использованию системы для управления климатом и другими параметрами теплицы. Приводятся подробные описания по работе с компонентами системы, методам управления и диагностики.

6.1 Установка системы

Подключение компонентов:

– соединить все датчики с соответствующими портам на Arduino;

– подключить исполнительные механизмы (насос, LED-ленту, линейный привод) к Arduino через реле и драйвер;

– убедиться, что все соединения выполнены надежно и соответствуют принципиальной схеме подключения.

6.2 Подключение ESP-01

Перед подключением ESP-01 к Arduino, необходимо в коде прошивки ESP-01 ввести SSID и пароль локальной WiFi сети. Для этого необходимо открыть файл `esp_api.ino` и изменить `ssid` и `password` на соответствующие локальной сети WiFi. Пример ввода названия сети и пароля:

```
const char* ssid = "TP-Link_49F2";
const char* password = "57375637";
```

Далее необходимо скомпилировать прошивку ESP-01 с помощью USB адаптера USB Serial Port Adapter. На рисунке 6.1 изображен данный USB адаптер.

Рисунок 6.1 – UBS адаптер для ESP-01

Следует выполнить следующий порядок действий:

– подключить ESP-01 к компьютеру с помощью USB адаптера;

– установить прошивку ESP-01 с помощью Arduino IDE или другого программного обеспечения для загрузки кода;

– после успешной прошивки, установить модуль ESP-01 на соответствующий разъем на печатной плате.

На рисунке 6.2 представлено подключение ESP-01 к печатной плате.

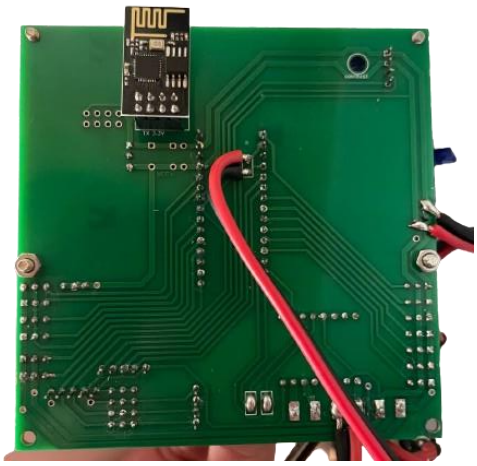


Рисунок 6.2 – Подключение ESP-01 к печатной плате

6.3 Подача питания

Необходимо подключить систему к источникам питания. На корпусе устройства выведены порты для подключения блоков питания на 5 В и 12 В. На рисунке 6.3 изображены порты для подключения питания.

Рисунок 6.3 – Порты для подключения питания

Далее следует включить источник питания и убедиться, что LCD-дисплей и другие индикаторы на подключенных устройствах также функционируют корректно.

6.3 Управление через веб-интерфейс

Доступ к веб-интерфейсу:

– подключиться к системе через веб-браузер, введя IP-адрес, присвоенный ESP-01 в сети;

– открыть главную страницу веб-интерфейса для доступа к функционалу системы.

Просмотр данных с датчиков:

– перейти на страницу `/sensors` для просмотра текущих значений датчиков температуры, влажности, освещенности и влажности почвы;

– убедиться, что данные обновляются корректно и отображают актуальные значения.

Управление устройствами:

– перейти на страницу `/states` для просмотра и управления текущими

состояниями исполнительных механизмов (насос, двери, свет);

– использовать веб-интерфейс для включения или выключения устройств, изменения состояний.

Обновление настроек:

– перейти на страницу `/settings` для изменения параметров системы;

– ввести новые значения настроек через форму и отправить их на сервер;

– убедиться, что настройки применены и система работает в соответствии с новыми параметрами.

Управление в ручном режиме:

– перейти на страницу `/manual` для управления системой в ручном режиме;

– вводить команды и параметры через веб-интерфейс для непосредственного управления исполнительными механизмами и датчиками;

– убедиться, что все команды выполняются корректно и изменения отражаются в системе.

Сброс к заводским настройкам:

– перейти на страницу `/factoryReset` для сброса всех настроек к заводским значениям;

– подтвердить операцию и проверить, что система вернулась к первоначальным настройкам.

6.3 Локальное управление

Использование энкодера:

– вращать энкодер для изменения параметров и навигации по меню на ЖК-дисплее;

– нажимать на энкодер для подтверждения выбора или входа в подменю.

На рисунке 6.4 представлена навигация по меню.

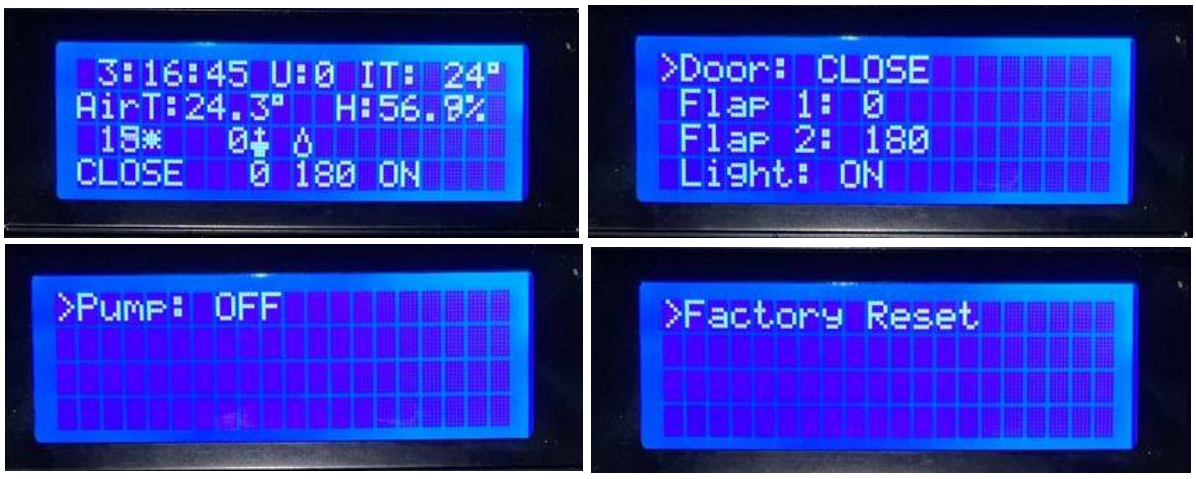


Рисунок 6.4 – Навигация по меню

Просмотр данных на ЖК-дисплее:

– данные с датчиков и состояния исполнительных механизмов

отображаются на LCD-дисплее;

– обновление данных происходит автоматически, позволяя следить за текущими параметрами в режиме реального времени.

Проверка кнопки для управления линейным приводом:

– нажимать кнопку для открытия или закрытия дверей с помощью линейного привода;

– убедиться, что линейный привод корректно реагирует на нажатие, открывая или закрывая двери.

6.4 Подведение итогов

Руководство пользователя предоставляет детальные инструкции по установке, настройке и эксплуатации комплекса для управления теплицей. Следуя инструкциям, пользователи смогут эффективно установить и настроить систему, управлять ее функциями как удаленно через веб-интерфейс, так и локально, обеспечивая оптимальные условия в теплице.

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА ДЛЯ УПРАВЛЕНИЯ ТЕПЛИЦЕЙ НА ОСНОВЕ БЕСПРОВОДНОЙ ТЕХНОЛОГИИ

7.1 Характеристика программно-аппаратного комплекса

Разрабатываемый в данном проекте программно-аппаратный комплекс для управления теплицей на основе беспроводной технологии представляет собой комплексное техническое решение, включающее в себя аппаратную составляющую и программное обеспечение. Он предназначен для автоматизации и управления процессами в теплице с использованием беспроводной технологий Wi-Fi. Такой комплекс эффективно решает задачи по контролю и регулированию микроклимата, полива, освещения и других процессов в тепличном хозяйстве.

Основные преимущества разрабатываемого программно-аппаратного комплекса заключаются в его гибкости и масштабируемости. Пользователь имеет возможность настройки параметров управления в зависимости от специфики своего хозяйства, а также расширения функционала системы по мере необходимости. Это позволяет адаптировать комплекс под конкретные потребности пользователей и оптимизировать его работу в соответствии с изменяющимися условиями.

Дополнительно, данный программно-аппаратный комплекс обладает преимуществами в области экологической устойчивости. Использование беспроводной технологии управления теплицей снижает необходимость в проведении кабельной разводки, что сокращает использование материальных ресурсов и уменьшает воздействие на окружающую среду.

Еще одним значимым преимуществом является экономическая выгода от использования беспроводных технологий. По сравнению с проводными системами, беспроводные технологии обладают более низкой стоимостью развертывания и обслуживания, а также обеспечивают более гибкую архитектуру системы. Это снижает затраты на инфраструктуру и упрощает процесс монтажа, что в свою очередь уменьшает общий бюджет внедрения системы управления теплицей на основе данного комплекса.

Таким образом, аппаратно-программный комплекс для управления теплицей на основе беспроводной технологии представляет собой перспективное решение, обеспечивающее эффективное управление процессами в тепличном хозяйстве и экономическую выгоду от его использования.

7.2 Расчет экономического эффекта от производства программно-аппаратного комплекса

Для определения результата от вложения инвестиций в проектирование и производство программно-аппаратного комплекса необходимо определить

отпускную цену программно-аппаратного комплекса на основе расчета затрат на производство аппаратной части и разработку программной части.

Расчет прямых затрат на материалы и комплектующие изделия для производства аппаратной части комплекса осуществляется в соответствии с представленной в конструкторской документации дипломного проекта номенклатурой, нормой расхода материалов, количеством комплектующих на изделие и рыночным ценам. Для производства данного комплекса были использованы материалы, представленные в таблице 7.1, и комплектующие, представленные в таблице 7.2.

Расчет затрат по статье «Основные и вспомогательные материалы», в которую включается стоимость необходимых для изготовления изделия основных и вспомогательных материалов, осуществляется по формуле (7.1):

$$Р_м = К_{тр} \cdot \sum_{i=1}^{n} Н_{рi} \cdot Ц_{отпi}, \quad (7.1)$$

где $К_{тр}$ – коэффициент транспортных расходов;

n – номенклатура применяемых материалов;

$Н_{рi}$ – норма расхода материала *i*-го вида на единицу изделия, нат. ед./шт.;

$Ц_{отпi}$ – цена за единицу материала *i*-го вида, р.

Таблица 7.1 – Расчет затрат на основные и вспомогательные материалы

Наименование материала	Единица измерения	Норма расхода материала	Цена за единицу материала, р.	Сумма, р.
1	2	3	4	5
1 Припой ПОС 61, 100 г	г	0,65	22,74	14,79
2 Флюс NC-559-ASM, 10 мл	мл	0,72	8,76	6,31
3 Кабель акустический 2х1,0 кв.мм., 1м	м	1,8	1,5	2,70
4 Трубка термоусаживаемая, 1 м	м	0,1	5,04	0,51
5 Пластиковая нить Creality CR-ABS 1,75мм 1 кг	г	0,9	57,00	51,30
6 Вилка штыревая 2,54 мм, 40 шт	шт	0,675	0,64	0,44
7 Клеммник винтовой DG306-5.0 2-контактный, 10 шт.	шт	0,4	10,37	4,15
8 Гнездо на плату PBD-8 2,54мм 2х4pin, 10 шт.	шт	0,1	14,90	1,49

Продолжение таблицы 7.1

1	2	3	4	5
9 Текстолит двусторонний 100х100 мм, 5 шт.	шт	0,2	6,54	1,31
10 Гнездо питания на панель DS-026A (DJK-04B), 2,5х5,5мм, 1 шт.	шт	3	3,30	9,90
11 Стойка пластмассовая 10мм, 1 шт.	шт	4	0,39	1,56
Итого				94,46
Всего с учетом транспортных расходов (1,1)($Р_м$)				103,91

Расчет затрат по статье «Покупные комплектующие изделия, полуфабрикаты», осуществляется по формуле (7.2):

$$Р_к = К_{тр} \cdot \sum_{i=1}^{m} N_i \cdot Ц_{отпi}, \quad (7.2)$$

где $К_{тр}$ – коэффициент транспортных расходов;

m – номенклатура применяемых комплектующих;

N_i – количество комплектующих *i*-го вида на единицу изделия, нат. ед./шт.;

$Ц_{отпi}$ – цена за единицу комплектующего *i*-го вида, р.

Таблица 7.2 – Расчет затрат на комплектующие изделия

Наименование комплектующего	Количество на изделие, шт.	Цена за единицу, р.	Сумма, р.
1	2	3	4
1 Микроконтроллер Nano V3.0 ATmega328P CH340G 16 МГц	1	26,00	26,00
2 Модуль RTC часов реального времени DS3231 (AT24C32) 3,3В/5В	1	9,00	9,00
3 Модуль датчика дождя MH-RD	1	5,00	5,00
4 Дисплей LCD2004 IIC/I2C	1	19,50	19,50
5 Модуль Wi-Fi ESP-01 (ESP8266)	1	23,80	23,80
6 Датчик температуры и влажности DHT22	1	20,40	20,40
7 Драйвер двигателей TB6612FNG Ultra L298N	1	16,00	16,00
8 Датчик влажности почвы	1	8,50	8,50
9 Модуль энкодера EC11	1	11,66	11,66
10 Линейный электропривод 12В, 35 см	1	128,15	128,15

Продолжение таблицы 7.2

1	2	3	4
11 Сервопривод SG90	2	17,00	24,00
12 Модуль стабилизатора напряжения AMS1117 3,3В	1	5,8	5,8
13 Модуль датчика уровня воды	1	1,18	1,18
14 Модуль датчика температуры, давления и влажности BME280	1	20,40	20,40
15 Кнопка без фиксации PBS-10B-2, OFF-(ON) (1A 250VAC)	1	1,90	1,90
16 Насос центробежный погружной DC30E 12В	1	18,86	18,86
17 LED лента SMD2835 12В 9,6Вт	1	14,41	14,41
18 Блок питания 12В 5А	1	25,25	25,25
19 Блок питания 5В 2А	2	10,25	20,50
20 Оптический датчик интенсивности освещенности GY-30 на базе BH1750	1	12,79	12,79
21 Модуль реле 2 канала 5В	1	7,50	7,50
Итого			419,89
Всего с учетом транспортных расходов (1,1)($Р_м$)			432,97

Расчет общей суммы прямых затрат на производство аппаратной части представлен в таблице 7.3.

Таблица 7.3 – Расчет общей суммы прямых затрат на производство аппаратной части

Показатель	Сумма, р.
1 Сырье и материалы	103,91
2 Покупные комплектующие изделия	432,97
Всего прямые затраты на производство аппаратной части ($З_р^{а.ч}$)	536,88

Расчет затрат на основную заработную плату разработчиков программной части комплекса представлен в таблице 7.4.

При расчете заработной платы используется среднемесячная заработная плата в Республике Беларусь для сотрудников ИТ-отрасли. Премия не начисляется.

Основная зарплата определяется по формуле (7.3):

$$З_о = К_{пр} \cdot \sum_{i=1}^{n} З_{чi} \cdot t_i, \quad (7.3)$$

где $К_{пр}$ – коэффициент премий и иных стимулирующих выплат;

n – категории исполнителей, занятых разработкой;

$З_{чi}$ – часовой оклад плата исполнителя *i*-й категории, р.;

t_i – трудоемкость работ, выполняемых исполнителем *i*-й категории, ч.

Часовая заработная плата каждого исполнителя определяется путем деления его месячной заработной платы (оклад плюс надбавки) на количество рабочих часов в месяце (расчетная норма рабочего времени на 2024 г. для 5-дневной недели составляет 168 часов по данным Министерства труда и социальной защиты населения на момент проведения расчетов).

Таблица 7.4 – Расчет затрат на основную заработную плату команды разработчиков

Категория разработчика	Месячный оклад, р.	Часовой оклад, р.	Трудоемкость работ, ч	Итого, р.
1 Инженер-программист	2250,00	13,40	30	402,00
2 Инженер-системотехник	1720,00	10,23	38	388,74
3 Инженер-отладчик	1470,00	8,75	30	262,50
Итого				1053,24
Премия и стимулирующие выплаты				0,00
Всего затраты на основную заработную плату разработчиков				1053,24

Дополнительная заработная плата определяется по формуле (7.4):

$$З_д = \frac{З_о \cdot Н_д}{100}, \quad (7.4)$$

где $Н_д$ – норматив дополнительной зарплаты, 15%.

Отчисления в фонд социальной защиты населения и обязательное страхование БелГосстрах ($З_{сз}$) определяется в соответствии с действующим законодательством по формуле (7.5):

$$Р_{соц} = \frac{(З_о + З_д) \cdot Н_{соц}}{100}, \quad (7.5)$$

где $Н_{соц}$ – норматив отчислений в ФСЗН и Белгосстрах (в соответствии с действующим законодательством по состоянию на апрель 2024 г. – 35%).

Расчет общей суммы затрат на разработку программной части программно-управляемого комплекса представлен в таблице 7.5.

Таблица 7.5 – Расчет затрат на разработку программного средства

Наименование статьи затрат	Формула/таблица для расчета	Сумма, р.
1	2	3
1 Основная заработная плата разработчиков	Табл. 7.4	1053,24
2 Дополнительная заработная плата разработчиков	$З_д = \frac{1053,24 \cdot 15}{100}$	157,99

Продолжение таблицы 7.5

1	2	3
3 Отчисления на социальные нужды	$Р_{соц}= \frac{(1053,24+157,99)\cdot 35}{100}$	423,93
Затраты на разработку программной части ($З_р^{п.ч}$)		1635,16

Формирование отпускной цены программно-аппаратного комплекса осуществляется в соответствии с методикой, представленной в таблице 7.6.

Таблица 7.6 – Методика формирования отпускной цены программно-аппаратного комплекса

Показатель	Формула/таблица для расчета
1 Затраты на производство аппаратной части ($З_р^{а.ч}$)	Табл. 7.3
2 Затраты на разработку программной части ($З_р^{п.ч}$)	Табл. 7.5
3 Сумма затрат на производство программно-аппаратного комплекса	$З_{пр} = З_р^{а.ч.} + З_р^{п.ч.}$ (7.6)
4 Накладные расходы	$Р_{накл} = \frac{З_{пр} \cdot Н_{накл}}{100}$, (7.7) где $Н_{накл}$ – норматив накладных расходов, 65%
5 Расходы на реализацию	$Р_{реал} = \frac{З_{пр} \cdot Н_{реал}}{100}$, (7.8) где $Н_{реал}$ – норматив расходов на реализацию, 2%
6 Полная себестоимость	$С_п = З_{пр} + Р_{накл} + Р_{реал}$ (7.9)
7 Плановая прибыль, включаемая в цену	$П_{ед} = \frac{С_п \cdot Р_{пр}}{100}$, (7.10) где $Р_{пр}$ – рентабельность продукции, 30%
8 Отпускная цена	$Ц_{отп} = С_п + П_{ед}$ (7.11)

Расчет отпускной цены осуществляется в таблице 7.7.

Таблица 7.7 – Формирование отпускной цены программно-аппаратного комплекса

Показатель	Формула/таблица для расчета	Сумма, р.
1	2	3
1 Затраты на производство аппаратной части ($З_р^{а.ч}$)	Табл. 7.3	536,88
2 Затраты на разработку программной части ($З_р^{п.ч}$)	Табл 7.5	1635,16

Продолжение таблицы 7.7

1	2	3
3 Сумма затрат на производство программно-аппаратного комплекса	$З_{пр}$ = 536,88 + 1635,16	2172,04
4 Накладные расходы	$Р_{накл}$ = 2172,04 · 0,65	1411,83
5 Расходы на реализацию	$Р_{реал}$ = 2172,04·0,02	43,44
6 Полная себестоимость	$С_{п}$ = 2172,04 + 1411,83 + 43,44	3627,31
7 Плановая прибыль, включаемая в цену	$П_{ед}$ = 3627,31 · 0,3	1088,20
8 Отпускная цена	$Ц_{отп}$ = 3627,31 + 1088,20	4715,51

Результатом производства аппаратно-программного комплекса является прирост чистой прибыли, полученный от их реализации и рассчитываемый по формуле (7.12):

$$\Delta П_{ч} = П_{ед} \cdot N_{п} \left(1 - \frac{Н_{п}}{100}\right), \quad (7.12)$$

где $N_{п}$ – прогнозируемый годовой объем производства и реализации, шт.;

$П_{ед}$ – прибыль, включаемая в цену, р.;

$Н_{п}$ – ставка налога на прибыль согласно действующему законодательству, (по состоянию на апрель 2024 г. – 20 %).

$$\Delta П_{ч} = 1088,20 \cdot 200 \cdot (1 - 0,20) = 174112,00 \text{ р.}$$

7.3 Расчет инвестиций в проектирование и производство программно-аппаратного комплекса

Инвестиции в производство аппаратно-программного комплекса включают в общем случае:

– инвестиции на его разработку;

– инвестиции в прирост собственного оборотного капитала (затраты на приобретение необходимых для производства нового изделия материалов, комплектующих, начатой, но незавершенной продукции и т.п.);

– инвестиции в прирост основного капитала (затраты на приобретение необходимого для производства нового изделия оборудования, станков и т.п.).

7.3.1 Инвестиции в разработку нового изделия

Инвестиции ($И_{р}$) рассчитываются по затратам на разработку нового изделия. Основная заработная плата разработчиков рассчитывается по формуле (7.3). Исходя из расчетов, полученных в таблице 7.5, размер инвестиций для разработки системы составляет 1635,16 р.

7.3.2 Расчет инвестиций в прирост собственного оборотного капитала

Расчет инвестиций в прирост собственного оборотного капитала осуществляется следующим образом:

1. Определяется годовая потребность в материалах по формуле (7.13).

$$П_м = Р_м \cdot N_п , \quad (7.13)$$

где $N_п$ – прогнозируемый годовой объем производства и реализации, шт.;

$Р_м$ – затраты на материалы на единицу изделия, р.

2. Определяется годовая потребность в комплектующих изделиях по формуле (7.14):

$$П_к = Р_к \cdot N_п , \quad (7.14)$$

где $N_п$ – прогнозируемый годовой объем производства и реализации, шт.;

$Р_м$ – затраты на комплектующие изделия на единицу продукции, р.

3. Вычисляются инвестиции в прирост собственного оборотного капитала в процентах от годовой потребности в материалах и комплектующих изделиях – β (исходя из среднего уровня по экономике: 25 %) по формуле (7.15):

$$И_{с.о.к} = \beta \cdot (П_м + П_к). \quad (7.15)$$

Расчет инвестиций в прирост собственного оборотного капитала представлен в таблице 7.9.

Таблица 7.9 – Расчет инвестиций в прирост собственного оборотного капитала

Наименование статьи затрат	Формула/таблица для расчета	Сумма, р.
1 Годовая потребность в материалах	$П_м = 103{,}91 \cdot 200$	20782,00
2 Годовая потребность в комплектующих изделиях	$П_м = 432{,}97 \cdot 200$	86594,00
Инвестиции в прирост собственного капитала	$И_{с.о.к} = 0{,}25 \cdot (20782{,}00 + 86594{,}00)$	26844,00

Исходя из расчетов, полученных в таблице 7.9, размер инвестиций в прирост собственного оборотного капитала составляет 26844,00 р.

7.3.3 Расчет инвестиций в прирост основного капитала

Инвестиции в прирост основного капитала не требуются, так как

производство нового изделия планируется осуществлять на действующем оборудовании в связи с наличием на предприятии-производителе свободных производственных мощностей.

7.4 Расчет показателей экономической эффективности инвестиций в проектирование и производство программно-аппаратного комплекса

Оценка экономической эффективности разработки и производства нового изделия зависит от результата сравнения инвестиций в производство нового изделия (инвестиции в разработку и прирост собственных оборотных средств) и полученного годового прироста чистой прибыли.

Сумма инвестиций меньше суммы годового экономического эффекта, то есть инвестиции окупятся менее чем за год, оценка экономической эффективности инвестиций в производство нового изделия осуществляется на основе расчета рентабельность инвестиций (затрат) по формуле (7.16):

$$ROI = \frac{\Delta П_ч - (И_р + И_{с.о.к})}{И_р + И_{с.о.к}} \cdot 100\%, \quad (7.16)$$

где $\Delta П_ч$ – прирост чистой прибыли от производства и реализации новых изделий, р.;

$И_р$, $И_{с.о.к}$ – инвестиции в разработку нового изделия и прирост собственного оборотного капитала, р.

$$ROI = \frac{174112{,}00 - (1635{,}16 + 26844{,}00)}{1635{,}16 + 26844{,}00} \cdot 100\% = 511{,}37\%$$

7.5 Вывод об экономической эффективности

В результате технико-экономического обоснования был произведен расчет инвестиций в разработку программно-аппаратного комплекса, расчет результата от разработки и использования программного средства и расчет показателей экономической эффективность. Общая сумма затрат на разработку программно-аппаратного комплекса составила 2172,04 рублей, отпускная цена – 4715,51 рублей. Экономическая эффективность инвестиций составляет 511,37 %.

Сравнивая инвестиции в разработку изделия и прирост собственных оборотных средств с приростом годовой чистой прибыли, можно сделать вывод, что инвестиции окупаются в течение года.

Рентабельность инвестиций в разработку и производство данного комплекса превышает ставки по долгосрочным депозитам в белорусских рублях на момент расчетов, что свидетельствует об экономической эффективности инвестиций. Следовательно, производство программно-

аппаратного комплекса для управления теплицей на основе беспроводной технологии является экономически эффективным, инвестиции в его разработку целесообразны.

Полученные результаты технико-экономического обоснования позволяют сделать вывод о высокой эффективности инвестиций в разработку программно-аппаратного комплекса для управления теплицей. Этот комплекс обладает перспективой стать востребованным на рынке, благодаря своей компактности, функциональности и экономической выгодности. Внедрение данного продукта позволит повысить эффективность управления тепличным хозяйством, обеспечивая оптимальные условия для роста растений.

Кроме того, разработанный программно-аппаратный комплекс представляет собой инновационное решение, способное повысить конкурентоспособность агропромышленных предприятий. Его использование позволит автоматизировать процессы управления теплицей, сократить затраты на энергию и ресурсы, а также повысить качество и количество выращенной продукции.

ЗАКЛЮЧЕНИЕ

В результате работы над данным дипломным проектом был разработан и протестирован программно-аппаратный комплекс для управления теплицей, включающий в себя плату микроконтроллера, модули датчиков, конечные устройства и систему удаленного мониторинга и управления через разработанное API.

Контроллер был представлен на 60-й научной конференции аспирантов, магистрантов и студентов [58].

На основе принципиальной схемы изготовлена печатная плата, которая была самостоятельно разведена. Данный этап включал выбор компонентов, проектирование дорожек и подключение всех элементов системы.

Основу системы составляет плата микроконтроллера Arduino Nano V3.0, которая управляет всеми аспектами тепличного контроля. Для реализации связи и передачи данных использовались модули Wi-Fi (ESP-01) и датчики окружающей среды. Программная часть комплекса написана на языке C с использованием среды разработки Arduino IDE, что обеспечило гибкость и удобство работы с различными модулями и датчиками.

Программно-аппаратный комплекс реализует следующие функциональные возможности:

– сбор данных с различных датчиков окружающей среды;

– управление конечными устройствами для поддержания оптимальных условий в теплице (включение/выключение освещения, системы полива и вентиляции);

– обработка и хранение данных, а также предоставление их пользователю через удобный веб-интерфейс;

– возможность удаленного мониторинга и управления системой через интернет.

Проектирование и реализация системы управления теплицей также включало написание и тестирование программных модулей для взаимодействия с аппаратными модулями. В процессе разработки были проанализированы и спроектированы необходимые условия для обеспечения надежной и стабильной работы системы.

Экономическая эффективность проекта была тщательно проанализирована. Общая сумма затрат на разработку программно-аппаратного комплекса составила 2172,04 рублей, а отпускная цена – 4715,51 рублей. Экономическая эффективность инвестиций составляет 511,37 %. Эти расчеты показывают, что инвестиции в разработку и производство данного комплекса окупаются в течение года, делая проект экономически целесообразным.

Проект продемонстрировал высокую надежность и функциональность системы, что подтверждается результатами тестирования в реальных условиях. В будущем возможна доработка и расширение функциональности комплекса, включая добавление новых датчиков, улучшение алгоритмов управления и интеграция с другими системами автоматизации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Контроллер теплицы TECH-UC1002 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://aliexpress.ru/item/1005002724395384.html?sku_id=12000034891829844 – Дата доступа: 03.04.2024

[2] Комплект автоматизации «УМНИЦА grow» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://umnica.pro/grow.html – Дата доступа: 03.04.2024

[3] Контроллер теплицы «ТЕРРАФОРМ» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://terraform.systems/ – Дата доступа: 03.04.2024

[4] Руководство пользователя «ТЕРРАФОРМ» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://terraform.systems/wp-content/uploads/2022/03/Терраформ-Руководство-пользователя-v4.pdf – Дата доступа: 03.04.2024

[5] Документация Arduino Nano [Электронный ресурс]. – Электронные данные. – Режим доступа: https://store.arduino.cc/products/arduino-nano – Дата доступа: 03.04.2024

[6] Схема распиновки Arduino Nano [Электронный ресурс]. – Электронные данные. – Режим доступа: https://content.arduino.cc/assets/Pinout-NANO_latest.pdf – Дата доступа: 03.04.2024

[7] Документация Arduino Pro Mini [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.arduino.cc/retired/boards/arduino-pro-mini/ – Дата доступа: 03.04.2024

[8] Схема распиновки Arduino Pro Mini [Электронный ресурс]. – Электронные данные. – Режим доступа: https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/ProMini16MHzv1.pdf – Дата доступа: 03.04.2024

[9] Документация Arduino Uno Rev3 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://store.arduino.cc/products/arduino-uno-rev3 – Дата доступа: 03.04.2024

[10] Схема распиновки Arduino Uno Rev3 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://content.arduino.cc/assets/A000066-full-pinout.pdf – Дата доступа: 03.04.2024

[11] KiCad PCB Editor [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.kicad.org/8.0/en/pcbnew/pcbnew.pdf – Дата доступа: 03.04.2024

[12] KiCad Introduction [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.kicad.org/8.0/en/introduction/introduction.pdf – Дата доступа: 03.04.2024

[13] CometCAD [Электронный ресурс]. – Электронные данные. – Режим доступа: https://cometcad.software.informer.com/1.0/ – Дата доступа: 03.04.2024

[14] CometCAD: From Symbol To PCB Layout [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.electronicsforu.com/buyers-guides/cometcad-from-symbol-to-pcb-layout – Дата доступа: 03.04.2024

[15] EasyEDA [Электронный ресурс]. – Электронные данные. – Режим доступа: https://easyeda.com/ru – Дата доступа: 03.04.2024

[16] EasyEDA Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.easyeda.com/en/FAQ/Editor/index.html – Дата доступа: 03.04.2024

[16] Design Flow by Using EasyEDA [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.easyeda.com/en/Introduction/Design-Flow-by-Using-EasyEDA/index.html – Дата доступа: 03.04.2024

[17] AutoCAD Electrical [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.autodesk.eu/products/autocad/included-toolsets/autocad-electrical – Дата доступа: 03.04.2024

[18] LTspice [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.analog.com/en/resources/design-tools-and-calculators/ltspice-simulator.html – Дата доступа: 03.04.2024

[19] Get Up and Running with LTspice [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.analog.com/en/resources/analog-dialogue/articles/get-up-and-running-with-ltspice.html – Дата доступа: 03.04.2024

[20] Среда разработки Arduino IDE [Электронный ресурс]. – Электронные данные. – Режим доступа: https://arduino.ru/Arduino_environment – Дата доступа: 03.04.2024

[21] Среда разработки PlatformIO [Электронный ресурс]. – Электронные данные. – Режим доступа: https://platformio.org/ – Дата доступа: 03.04.2024

[22] Arduino: передача данных по UART [Электронный ресурс]. – Электронные данные. – Режим доступа: https://robotclass.ru/tutorials/arduino-uart/ – Дата доступа: 17.04.2024

[23] UART – Последовательный интерфейс передачи данных [Электронный ресурс]. – Электронные данные. – Режим доступа: https://voltiq.ru/wiki/uart-interface/ – Дата доступа: 17.04.2024

[24] Последовательный порт UART в Ардуино [Электронный ресурс]. – Электронные данные. – Режим доступа: http://mypractic.ru/urok-12-posledovatelnyj-port-uart-v-arduino-biblioteka-serial-otladka-programm-na-arduino.html – Дата доступа: 17.04.2024

[25] Интерфейс I2C: принципы функционирования [Электронный ресурс]. – Электронные данные. – Режим доступа: https://dzen.ru/a/Y8-Xp4mc1StLbKbz – Дата доступа: 17.04.2024

[26] Интерфейс I2C (IIC) [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ampermarket.kz/base/i2c_interface/ – Дата доступа: 17.04.2024

[27] Интерфейс передачи данных - I2C [Электронный ресурс]. –

Электронные данные. – Режим доступа: https://3d-diy.ru/wiki/arduino-moduli/interfeys-peredachi-dannykh-i2c/ – Дата доступа: 17.04.2024

[28] Что такое ШИМ – широтно-импульсная модуляция? [Электронный ресурс]. – Электронные данные. – Режим доступа: https://dzen.ru/a/ZRL4Vs_-WHQe-sPg – Дата доступа: 17.04.2024

[29] Что такое Wi-Fi [Электронный ресурс]. – Электронные данные. – Режим доступа: https://skillbox.ru/media/code/chto-takoe-wifi-obyasnyaem-prostymi-slovami/ – Дата доступа: 17.04.2024

[30] Arduino Nano Datasheet [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf – Дата доступа: 17.04.2024

[31] ESP-01 – Wi-Fi модуль на базе ESP8266 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://voltiq.ru/shop/esp-01-wifi-module-esp8266/ – Дата доступа: 17.04.2024

[32] ESP-01 WiFi Module [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.microchip.ua/wireless/esp01.pdf – Дата доступа: 17.04.2024

[33] NodeMCU 32S [Электронный ресурс]. – Электронные данные. – Режим доступа: https://voltiq.ru/shop/nodemcu-32s-38pin/ – Дата доступа: 17.04.2024

[34] Nodemcu-32s Datasheet [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.ai-thinker.com/_media/esp32/docs/nodemcu-32s_product_specification.pdf – Дата доступа: 17.04.2024

[35] Модуль энкодера EC11 Arduino [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ozon.by/product/modul-enkodera-ec11-arduino-s-knopkoy-i-kolpachkom-832420460/ – Дата доступа: 17.04.2024

[36] Rotation Sensor, Энкодер для Arduino проектов [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.chipdip.by/product/rotation-sensor – Дата доступа: 17.04.2024

[37] Дисплей LCD 2004 I2C [Электронный ресурс]. – Электронные данные. – Режим доступа: https://duino.ru/displey-lcd-2004/ – Дата доступа: 17.04.2024

[38] OLED дисплей SSD1306 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://duino.ru/OLED-displei-0-96---128h64--belyi.html/ – Дата доступа: 17.04.2024

[39] SG90 Сервопривод [Электронный ресурс]. – Электронные данные. – Режим доступа: https://duino.ru/SG90-Servoprivod.html/ – Дата доступа: 17.04.2024

[40] Сервопривод MG90S [Электронный ресурс]. – Электронные данные. – Режим доступа: https://duino.ru/mg90s.html/ – Дата доступа: 17.04.2024

[41] Линейный привод (актуатор) 12 В [Электронный ресурс]. – Электронные данные. – Режим доступа:

https://www.42unita.ru/catalog/aktuatory/Lineynyy_privod_AL03_12_A2_520_400_C11_a9f – Дата доступа: 17.04.2024

[42] ЛИНЕЙНЫЙ АКТУАТОР НТА-18К [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.kipspb.ru/catalog/15047/element6116159.php – Дата доступа: 17.04.2024

[43] Двойной драйвер мотора 1A TB6612FNG for Arduino Microcontroller [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.samarachip.ru/product_info.php?products_id=3653 – Дата доступа: 17.04.2024

[44] 2-канальный модуль реле (5 В) [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ampermarket.kz/relay/mod/2-channel-relay-module-srd-5vdc-sl-c/#tab-tab-custom – Дата доступа: 17.04.2024

[45] Насос центробежный погружной DC30E 12 В [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ozon.by/product/nasos-tsentrobezhnyy-pogruzhnoy-dc30e-12-v-s-mokrym-rotorom-842360703/ – Дата доступа: 17.04.2024

[46] ЖИДКОСТНАЯ ПОМПА LF BROS (ЛУНФЕЙ) 12В [Электронный ресурс]. – Электронные данные. – Режим доступа: https://avtogear.ru/pompa_lunfey.html – Дата доступа: 17.04.2024

[47] LED лента на SMD 2835 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ozon.by/product/led-lenta-na-smd-2835-ip22-120-led-12v-dc-9-6w-teplyy-belyy-3200k-1-metr-brend-dled-1187524487/#section-description--offset-140 – Дата доступа: 17.04.2024

[48] Датчик уровня воды [Электронный ресурс]. – Электронные данные. – Режим доступа: https://3d-diy.ru/wiki/arduino-datchiki/arduino-datchik-urovnya-vody/ – Дата доступа: 17.04.2024

[49] GY-30 оптический датчик интенсивности освещенности на базе BH1750 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ozon.by/product/gy-30-opticheskiy-datchik-intensivnosti-osveshchennosti-na-baze-bh1750-818175165/ – Дата доступа: 17.04.2024

[50] Датчик освещенности TEMT6000 для Ардуино [Электронный ресурс]. – Электронные данные. – Режим доступа: https://arduino.ua/ru/prod2560-datchik-osveshhennosti-temt6000-dlya-ardyino – Дата доступа: 17.04.2024

[51] Датчик влажности почвы [Электронный ресурс]. – Электронные данные. – Режим доступа: https://easycraft.by/datchik-vlaghnosti-pochvy – Дата доступа: 17.04.2024

[52] DHT22 - Датчик влажности и температуры [Электронный ресурс]. – Электронные данные. – Режим доступа: https://duino.ru/DHT22-Datchik-vlazhnosti-i-temperatury.html/ – Дата доступа: 17.04.2024

[53] DHT11 датчик относительной влажности и температуры [Электронный ресурс]. – Электронные данные. – Режим доступа: https://duino.ru/dht11.html/ – Дата доступа: 17.04.2024

[54] BME280 датчик абсолютного давления, температуры и влажности

[Электронный ресурс]. – Электронные данные. – Режим доступа: https://duino.ru/bme280-datchik-absoljutnogo-davlenija-temperatury-i-vlazhnosti.html/ – Дата доступа: 17.04.2024

[55] ДАТЧИК ДОЖДЯ (RAIN SENSOR MODULE) [Электронный ресурс]. – Электронные данные. – Режим доступа: https://zevs.by/maketki/datchiki/datchik_dozhdya_rain_sensor_module/ – Дата доступа: 17.04.2024

[56] Блок питания 5V 2A (5В 2А) [Электронный ресурс]. – Электронные данные. – Режим доступа: httpshttps://ozon.by/product/blok-pitaniya-5v-2a-5v-2a-dlya-tsifrovyh-pristavok-dvb-t2-setevoy-adapter-universalnyy-shteker-5-5x2-997090252/ – Дата доступа: 17.04.2024

[57] Блок питания 12V 5A (12в 5а) [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ozon.by/product/blok-pitaniya-12v-5a-12v-5a-setevoy-adapter-stabilizirovannyy-shteker-5-5-h-2-5mm-1214399661/?advert=DCdU9GU_YSPJKpLSP0hoG-qkoQ-7wgY5lp-9UMBXezsC1swanGZY97zU2wPziNdZoW3722b7AYLyz5JIRmlvDmlYXoxBfmyHKAUaKFozz-XLbJpAj5Joq_6bHYLpE4hxr28_CQrrW6YKdp3TsfB2hd1ul8i6zebwxJm3LSjV&avtc=1&avte=2&avts=1713707322 – Дата доступа: 17.04.2024

[58] Андриевский, Е. С. Программно-аппаратный комплекс для управления теплицей на основе беспроводной технологии / Андриевский Е. С. // Компьютерные системы и сети : сборник материалов 60-й научной конференции аспирантов, магистрантов и студентов, Минск, 18–22 апреля 2024 г. / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2024 (в печати)

Вычислительные машины, системы и сети: дипломное проектирование [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.bsuir.by/m/12_100229_1_136308.pdf – Дата доступа: 02.04.2024.

Экономика проектных решений: методические указания по экономическому обоснованию дипломных проектов [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.bsuir.by/m/12_100229_1_161144.pdf – Дата доступа: 02.04.2024.

ПРИЛОЖЕНИЕ А
(обязательное)

Листинг кода

```
// Файл esp_api.ino

#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266WiFiMulti.h>
#include <ArduinoJson.h>

const char* ssid = "TP-Link_49F2";
const char* password = "06802912";

ESP8266WiFiMulti wifiMulti;
ESP8266WebServer server(80);

String sensors = "";
String states = "";

void getSensors() {
  if (sensors.isEmpty()) server.send(503, "text/plain", "Not yet
loaded");
  else server.send(200, "application/json", sensors);
}

void getStates() {
  if (states.isEmpty()) server.send(503, "text/plain", "Not yet
loaded");
  else server.send(200, "application/json", states);
}

static String readJsonTable() {
  String json = "{";
  while (true) {
    String name = Serial.readStringUntil(':');
    name.trim();
    if (name == "end") break;
    String value = Serial.readStringUntil('\n');
    value.trim();
    json += '"' + name + "\":" + value + ",";
  }
  if (json.endsWith(",")) json.remove(json.length() - 1, 1);
  return json + "}";
}

static void checkSerial() {
  while   (Serial.available()   &&   Serial.peek()   !=   ';')
Serial.read();  // Sync
  if (Serial.peek() == ';') {
    Serial.read();
```

```
    String message = Serial.readStringUntil('\n');
    message.trim();
    if (message == "sensors") sensors = readJsonTable();
    else if (message == "states") states = readJsonTable();
    else if (message == "wifi on") {
      WiFi.mode(WIFI_STA);
      while (wifiMulti.run() != WL_CONNECTED) {
        delay(500);
      }
    } else if (message == "wifi off") WiFi.mode(WIFI_OFF);
  }
}

static void getResponce() {
  String codeText = Serial.readStringUntil(';');
  codeText.trim();
  uint16_t code = codeText.toInt();
  String payload = Serial.readStringUntil('\n');
  payload.trim();
  if (payload.isEmpty()) server.send(code);
  else server.send(code, "text/plain", payload);
}

void setManual() {
  JsonDocument doc;
  deserializeJson(doc, server.arg("plain"));
  Serial.println(";manual");
  for (JsonPair kv : doc.as<JsonObject>()) {
    Serial.print(kv.key().c_str());
    Serial.print(':');
    serializeJson(kv.value(), Serial);
    Serial.println();
  }
  Serial.println("end:");
  getResponce();
}

void getManual() {
  Serial.println(";getManual");
  server.send(200, "application/json", readJsonTable());
}

void setSettings() {
  JsonDocument doc;
  deserializeJson(doc, server.arg("plain"));
  Serial.println(";settings");
  for (JsonPair kv : doc.as<JsonObject>()) {
    Serial.print(kv.key().c_str());
    Serial.print(':');
    serializeJson(kv.value(), Serial);
    Serial.println();
  }
  Serial.println("end:");
```

```
  getResponce();
}

void getSettings() {
  Serial.println(";getSettings");
  server.send(200, "application/json", readJsonTable());
}

void factoryReset() {
  Serial.println(";factoryReset");
  getResponce();
}

void setup() {
  Serial.begin(9600);
  wifiMulti.addAP(ssid, password);
  while (wifiMulti.run() != WL_CONNECTED) {
    delay(500);
  }
  server.on("/sensors", HTTP_GET, getSensors);
  server.on("/states", HTTP_GET, getStates);
  server.on("/manual", HTTP_POST, setManual);
  server.on("/manual", HTTP_GET, getManual);
  server.on("/settings", HTTP_POST, setSettings);
  server.on("/settings", HTTP_GET, getSettings);
  server.on("/factoryReset", HTTP_POST, factoryReset);
  server.begin();
}

void loop() {
  checkSerial();
  server.handleClient();
}

// Файл functionality.h

#pragma once
#include <Arduino.h>

enum class SensorReadingStage {
  PowerOn,
  Read,
  PowerOff,
  UpdateOutputs,
};

extern SensorReadingStage sensorReadingStage;

void initSensors();
void readTemperatures();
void readAllSensors();
void updateTime();
void initOutputs();
```

```
void updateOutputs();
void sendSensorValues();
void sendStates();
void updateServer();

enum {
  LCD_WATER = 0,
  LCD_NO_WATER,
  LCD_LIGHT,
  LCD_CLOUD,
  LCD_FLOWER,
  LCD_PERCENT = 37,
  LCD_DEGREE = 223,
};

void initLCD();
void updateLCD();

// Файл greenhouse.ino

#include "objects.h"
#include "functionality.h"

void setup() {
  Serial.begin(9600);

  EEPROM.get(SETTINGS_ADDRESS, settings);
  if (settings.magic != MAGIC) {
    settings = Settings();
    EEPROM.put(SETTINGS_ADDRESS, settings);
  }

  #if defined(WIRE_OVERCLOCK)
    Wire.setClock(WIRE_OVERCLOCK);
  #endif

  EICRA = (EICRA & 0x0C) | 1;
  bitSet(EIMSK, INT0);
  EICRA = (EICRA & 0x03) | (1 << 2);
  bitSet(EIMSK, INT1);
  enc.setEncISR(true);

  initOutputs();
  initSensors();
  initLCD();
}

void loop() {
  #if            defined(CRITICAL_HIGH_TEMPERATURE)            ||
defined(CRITICAL_LOW_TEMPERATURE)
    #ifndef CRITICAL_HIGH_TEMPERATURE
      if (insideT <= CRITICAL_LOW_TEMPERATURE) {
    #elif !defined(CRITICAL_LOW_TEMPERATURE)
```

```
      if (insideT >= CRITICAL_HIGH_TEMPERATURE) {
    #else
      if (insideT <= CRITICAL_LOW_TEMPERATURE || insideT >=
CRITICAL_HIGH_TEMPERATURE) {
    #endif

    if (!criticalTemperature) {
      criticalTemperature = true;

      digitalWrite(SENS_VCC, 0);
      sensorReadingStage = SensorReadingStage::PowerOn;

      #ifdef LIGHT_PORT
        digitalWrite(LIGHT_PORT, HIGH);
      #endif

      #ifdef PUMP_PORT
        digitalWrite(PUMP_PORT, HIGH);
      #endif

      #ifdef DRIVER_LEVEL
        digitalWrite(DRV_SIGNAL1, DRIVER_LEVEL);
        digitalWrite(DRV_SIGNAL2, DRIVER_LEVEL);
        digitalWrite(DRV_PWM, LOW);
      #endif

      Serial.println(";wifi off");

      lcd.backlight();
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Critical Temp!");
    }

    readTemperatures();
    lcd.setCursor(0, 1);
    lcd.print("Temperature: ");
    if (insideT >= 0) lcd.print(' ');
    if (abs(insideT) < 10) lcd.print(' ');
    lcd.print(insideT, 1);
    lcd.write(LCD_DEGREE);
    return;
  } else if (criticalTemperature) {
    criticalTemperature = false;
    Serial.println(";wifi on");
  }
  #endif

  updateTime();
  readAllSensors();
  updateLCD();
  updateServer();
}
```

```
ISR(INT0_vect) {
  enc.tickISR();
}

ISR(INT1_vect) {
  enc.tickISR();
}

// Файл lcd.cpp

#include "objects.h"
#include "functionality.h"

static uint8_t waterIcon[] = {
  B00100,
  B00100,
  B01110,
  B01110,
  B11111,
  B11111,
  B11111,
  B01110
};

static uint8_t noWaterIcon[] = {
  B00100,
  B00100,
  B01010,
  B01010,
  B10001,
  B10001,
  B10001,
  B01110
};

static uint8_t lgithIcon[] = {
  B00000,
  B10101,
  B01110,
  B11111,
  B01110,
  B10101,
  B00000,
  B00000
};

static uint8_t cloudIcon[] = {
  B01110,
  B11111,
  B11111,
  B00000,
  B01000,
```

```
  B00010,
  B01000,
  B00000
};

static uint8_t flowerIcon[] = {
  B00100,
  B01110,
  B00100,
  B00100,
  B11111,
  B11111,
  B01110,
  B01110
};

void initLCD() {
  lcd.init();
  lcd.createChar(LCD_WATER, waterIcon);
  lcd.createChar(LCD_NO_WATER, noWaterIcon);
  lcd.createChar(LCD_LIGHT, lgithIcon);
  lcd.createChar(LCD_CLOUD, cloudIcon);
  lcd.createChar(LCD_FLOWER, flowerIcon);
  lcd.backlight();
  lcd.clear();
}

static void controllable(Output& output, String name, uint8_t
stateEnabled, uint8_t y, bool highlighted) {
  lcd.setCursor(0, y);
  if (highlighted) {
    lcd.print('>');
    if (enc.press()) {
      if (!output.manual) {
        output.manual = true;
        output.state = 0;
      } else if (!output.state) output.state = stateEnabled;
      else output.manual = false;
    }
  } else lcd.print(' ');
  lcd.print(name + ": ");
}

void updateLCD() {
  static uint64_t updateTimer = 0;
  static uint8_t page = 0, entry = 0;
  static bool factoryReset = false;
  enc.tick();
  if (enc.holding()) {
    door.state = 1;
    door.manual = true;
  }
```

```
  if (enc.turn()) {
    const uint8_t ENTRIES[] = { 1, 4, 1, factoryReset ? 2 : 1 };
    const uint8_t PAGES = sizeof(ENTRIES) / sizeof(ENTRIES[0]);
    int8_t newEntry = entry + enc.dir();
    if (factoryReset) {
      newEntry = (newEntry + ENTRIES[page]) % ENTRIES[page];
    } else if (newEntry < 0 || newEntry >= ENTRIES[page]) {
      lcd.clear();
      factoryReset = false;
      if (newEntry < 0) {
        page = (page + PAGES - 1) % PAGES;
        newEntry += ENTRIES[page];
      } else {
        newEntry %= ENTRIES[page];
        page = (page + 1) % PAGES;
      }
    }
    entry = newEntry;
  }

  if (millis() - updateTimer > settings.sensorPeriod * 1000 / 2 ||
enc.action()) {
    if (page == 0) {
      lcd.setCursor(0, 0);
      if (datetime.Hour < 10) lcd.print(' ');
      lcd.print(datetime.Hour);
      lcd.print(':');
      if (datetime.Minute < 10) lcd.print('0');
      lcd.print(datetime.Minute);
      lcd.print(':');
      if (datetime.Second < 10) lcd.print('0');
      lcd.print(datetime.Second);
      lcd.print(" U:");
      lcd.print((uint32_t)(uptime / 60ull / 60ull / 24ull));
      lcd.print(" IT:");
      if (insideT >= 0) lcd.print(' ');
      if (abs(insideT) < 10) lcd.print(' ');
      lcd.print(insideT, 0);
      lcd.write(LCD_DEGREE);

      lcd.setCursor(0, 1);
      lcd.print("AirT:");
      lcd.print(airT, 1);
      lcd.write(LCD_DEGREE);
      lcd.print(' ');

      lcd.setCursor(12, 1);
      lcd.print("H:");
      lcd.print(airH, 1);
      lcd.write(LCD_PERCENT);

      lcd.setCursor(0, 2);
      if (lightLevel < 100) lcd.print(' ');
```

```
      if (lightLevel < 10) lcd.print(' ');
      lcd.print(lightLevel);
      lcd.write(LCD_LIGHT);

      lcd.setCursor(5, 2);
      if (soilMoisture < 100) lcd.print(' ');
      if (soilMoisture < 10) lcd.print(' ');
      lcd.print(soilMoisture);
      lcd.write(LCD_FLOWER);

      lcd.setCursor(10, 2);
      lcd.write(enoughWater ? LCD_WATER : LCD_NO_WATER);

      lcd.setCursor(11, 2);
      lcd.write(raining ? LCD_CLOUD : ' ');

      lcd.setCursor(0, 3);
      lcd.print(door.state ? "OPEN " : "CLOSE");
      lcd.setCursor(6, 3);
      if (flap1.state < 100) lcd.print(' ');
      if (flap1.state < 10) lcd.print(' ');
      lcd.print(flap1.state);
      lcd.setCursor(10, 3);
      if (flap2.state < 100) lcd.print(' ');
      if (flap2.state < 10) lcd.print(' ');
      lcd.print(flap2.state);
      lcd.setCursor(14, 3);
      lcd.print(light.state ? "ON " : "OFF");
      lcd.setCursor(18, 3);
      lcd.write(pump.state ? LCD_WATER : ' ');
    } else if (page == 1) {

      controllable(door, "Door", 1, 0, entry == 0);
      lcd.print(door.manual ? (door.state ? "OPEN " : "CLOSE") :
"AUTO ");

      controllable(flap1, "Flap 1", 180, 1, entry == 1);
      lcd.print(flap1.manual  ?  String(flap1.state)  +  "     "  :
"AUTO");

      controllable(flap2, "Flap 2", 180, 2, entry == 2);
      lcd.print(flap2.manual  ?  String(flap2.state)  +  "     "  :
"AUTO");

      controllable(light, "Light", 1, 3, entry == 3);
      lcd.print(light.manual ? (light.state ? "ON   " : "OFF ") :
"AUTO");
    } else if (page == 2) {
      controllable(pump, "Pump", settings.wateringTime, 0, entry
== 0);
      lcd.print(pump.manual ? (pump.state ? String(pump.state) +
"    " : "OFF ") : "AUTO");
    } else if (page == 3) {
```

```
      if (!factoryReset) {
        lcd.setCursor(0, 0);
        lcd.print(">Factory Reset");
        if (enc.press()) factoryReset = true;
      } else {
        lcd.setCursor(0, 0);
        lcd.print("Are you sure? ");
        lcd.setCursor(0, 1);
        lcd.print(entry == 0 ? '>' : ' ');
        lcd.print("Yes");
        lcd.setCursor(0, 2);
        lcd.print(entry == 1 ? '>' : ' ');
        lcd.print("No");

        if (enc.press()) {
          if (entry == 0) {
            settings = Settings();
            EEPROM.put(SETTINGS_ADDRESS, settings);
            page = 0;
          }
          entry = 0;
          factoryReset = false;
          lcd.clear();
        }
      }
    }
  }

  #ifdef LCD_BACKLIGHT_TIMEOUT
  {
    static uint64_t backlightTimer = 0;
    static bool backlit = true;
    if (enc.action()) {
      if (!backlit) lcd.backlight();
      backlit = true;
      backlightTimer = millis();
    } else if (backlit && millis() - backlightTimer >
LCD_BACKLIGHT_TIMEOUT * 1000u) {
      lcd.noBacklight();
      backlit = false;
    }
  }
  #endif
}

// Файл objects.cpp

#include "objects.h"

LiquidCrystal_I2C lcd(LCD_ADDR, 20, 4);
EncButton enc(CLK, DT, SW);
Servo servo1;
Servo servo2;
```

```
#ifdef BME_ADDR
GyverBME280 bme;
#endif

#if defined(DHT_PORT) && defined(DHT_TYPE)
DHT dht(DHT_PORT, DHT_TYPE);
#endif

#ifdef LIGHT_SENSOR_ENABLED
#include <BH1750.h>
BH1750 lightSensor;
#endif

#if             defined(CRITICAL_HIGH_TEMPERATURE)              ||
defined(CRITICAL_LOW_TEMPERATURE)
bool criticalTemperature = false;
#endif

Settings settings;

tmElements_t datetime;
uint64_t uptime = 0;
float airT = 25.0, airH = 30.0;
float insideT = airT, insideH = airH;
uint16_t lightLevel = 0;
uint8_t soilMoisture = 100;
bool enoughWater = true;
bool raining = false;

Output flap1;
Output flap2;
Output door;
Output pump;
Output light;

// Файл objects.h

#pragma once
#include "settings.h"

#include <EEPROM.h>

#include <LiquidCrystal_I2C.h>
extern LiquidCrystal_I2C lcd;

#include <EncButton.h>
extern EncButton enc;

#ifdef BME_ADDR
#include <GyverBME280.h>
extern GyverBME280 bme;
#endif
```

```
#if defined(DHT_PORT) && defined(DHT_TYPE)
#include <DHT.h>
extern DHT dht;
#endif

#ifdef LIGHT_SENSOR_ENABLED
#include <BH1750.h>
extern BH1750 lightSensor;
#endif

#include <DS1307RTC.h>

#include <Servo.h>
extern Servo servo1;
extern Servo servo2;

#if             defined(CRITICAL_HIGH_TEMPERATURE)              ||
defined(CRITICAL_LOW_TEMPERATURE)
extern bool criticalTemperature;
#endif

const uint32_t MAGIC = 0xABCD1234;
const uint32_t SETTINGS_ADDRESS = 0;
struct Settings {
  uint32_t magic = MAGIC;
  uint8_t driveSpeed = 125;
  uint16_t sensorPeriod = 1;

  float flapsTemperatureThreshold = 30.0;
  float doorTemperatureThreshold = 30.0;
  uint16_t ventelationPeriod = 60;
  uint16_t ventelationTime = 10;
  bool closeIfRain = true;

  uint8_t wateringMoistureThreshold = 50;
  uint16_t wateringTime = 10;
  uint16_t wateringTimeout = 30;
  uint16_t wateringPeriod = 30;

  uint8_t lightLevelThreshold = 50;
};
extern Settings settings;

extern tmElements_t datetime;
extern uint64_t uptime;
extern float airT, airH;
extern float insideT, insideH;
extern uint16_t lightLevel;
extern uint8_t soilMoisture;
extern bool enoughWater;
extern bool raining;

struct Output {
```

```
  uint8_t state = 0;
  bool manual = false;
};

extern Output flap1;
extern Output flap2;
extern Output door;
extern Output pump;
extern Output light;

// Файл outputs.cpp

#include "objects.h"
#include "functionality.h"

void initOutputs() {
  #ifdef LIGHT_PORT
    pinMode(LIGHT_PORT, OUTPUT);
  #endif

  #ifdef PUMP_PORT
    pinMode(PUMP_PORT, OUTPUT);
  #endif

  #ifdef DRIVER_LEVEL
    TCCR2A |= _BV(WGM20);
    TCCR2B = TCCR2B & 0b11111000 | 0x01;
    pinMode(DRV_PWM, OUTPUT);
    pinMode(DRV_SIGNAL1, OUTPUT);
    pinMode(DRV_SIGNAL2, OUTPUT);
    digitalWrite(DRV_SIGNAL1, DRIVER_LEVEL);
    digitalWrite(DRV_SIGNAL2, DRIVER_LEVEL);
    analogWrite(DRV_PWM, settings.driveSpeed);
  #endif

  servo1.attach(SERVO1, SERVO_MIN_PULSE, SERVO_MAX_PULSE);
  servo2.attach(SERVO2, SERVO_MIN_PULSE, SERVO_MAX_PULSE);
}

void updateOutputs() {
  static uint64_t ventelationTimer = 0;
  bool flaps = false, doorOpen = false;

  if (uptime - ventelationTimer >= settings.ventelationPeriod *
60) ventelationTimer = uptime;
  if (raining && settings.closeIfRain) flaps = false, doorOpen =
false;
  else {
    if (airT > settings.flapsTemperatureThreshold) flaps = true;
    if (airT > settings.doorTemperatureThreshold) doorOpen = true;
    else if (uptime - ventelationTimer < settings.ventelationTime
* 60) {
      flaps = true;
```

```
      doorOpen = true;
    }
  }

  if (!flap1.manual) flap1.state = flaps * 180;
  if (!flap2.manual) flap2.state = flaps * 180;
  if (!door.manual) door.state = doorOpen;
  if (!pump.manual) {
    static uint64_t wateringTimer = 0, wateringPeriodTimer = 0;
    if (uptime - wateringPeriodTimer >= settings.wateringPeriod *
60) {
      wateringPeriodTimer = uptime;
      pump.state = settings.wateringTime;
    }
    if (uptime - wateringTimer >= settings.wateringTimeout) {
      wateringTimer = uptime;
      if (soilMoisture < settings.wateringMoistureThreshold) {
        pump.state = settings.wateringTime;
      }
    }
  }
  {
    static uint64_t pumpTimer = 0;
    if (uptime - pumpTimer >= 1 && pump.state > 0) {
      pumpTimer = uptime;
      pump.state--;
    }
  }
  if (!enoughWater) pump.state = 0;

  if (!light.manual) {
    light.state = lightLevel < settings.lightLevelThreshold;
  }

  servo1.write(flap1.state);
  servo2.write(flap2.state);

  #ifdef LIGHT_PORT
    digitalWrite(LIGHT_PORT, !light.state);
  #endif

  #ifdef PUMP_PORT
    digitalWrite(PUMP_PORT, pump.state == 0);
  #endif

  #ifdef DRIVER_LEVEL
    digitalWrite(DRV_SIGNAL1, DRIVER_LEVEL != door.state);
    digitalWrite(DRV_SIGNAL2, DRIVER_LEVEL != !door.state);
    analogWrite(DRV_PWM, settings.driveSpeed);
  #endif
  sendStates();
}
```

```
// Файл pins.h

#pragma once
#include <Arduino.h>

#define SW 0
#define DT 2
#define CLK 3

#define RELAY1 1
#define RELAY2 4
#define RELAY3 5
#define RELAY4 6
#define RELAY5 7
#define RELAY6 8
#define RELAY7 9
const uint8_t relayPins[] = {RELAY1, RELAY2, RELAY3, RELAY4,
RELAY5, RELAY6, RELAY7};

#define SERVO1 13
#define SERVO2 A0

#define DRV_SIGNAL1 10
#define DRV_SIGNAL2 12
#define DRV_PWM 11

#define SENS_VCC A1
#define SENS1 A2
#define SENS2 A3
#define SENS3 A6
#define SENS4 A7
const uint8_t sensorPins[] = {SENS1, SENS2, SENS3, SENS4};

// Файл sensors.cpp

#include "objects.h"
#include "functionality.h"

SensorReadingStage              sensorReadingStage              =
SensorReadingStage::PowerOn;

static int charToDec(const char* p) {
  return ((*p - '0') * 10 + (*(p + 1) - '0'));
}

void initSensors() {
  pinMode(SENS_VCC, OUTPUT);

  if   (!RTC.read(datetime)   ||   (datetime.Year   ==   2000   &&
datetime.Month == 1 && datetime.Day == 1)) {
    char* stamp = __TIMESTAMP__;
    datetime.Hour = charToDec(stamp + 11);
    datetime.Minute = charToDec(stamp + 14);
```

```
    datetime.Second = charToDec(stamp + 17);
    datetime.Day = charToDec(stamp + 8);
    switch (stamp[4]) {
      case 'J': datetime.Month = (stamp[5] == 'a') ? 1 : ((stamp[6]
== 'n') ? 6 : 7); break;
      case 'F': datetime.Month = 2; break;
      case 'A': datetime.Month = (stamp[6] == 'r') ? 4 : 8; break;
      case 'M': datetime.Month = (stamp[6] == 'r') ? 3 : 5; break;
      case 'S': datetime.Month = 9; break;
      case 'O': datetime.Month = 10; break;
      case 'N': datetime.Month = 11; break;
      case 'D': datetime.Month = 12; break;
    }
    datetime.Year = 2000 + charToDec(stamp + 22);
    RTC.write(datetime);
  }

  #ifdef BME_ADDR
    bme.begin(BME_ADDR);
  #endif

  #if defined(DHT_PORT) && defined(DHT_TYPE)
    dht.begin();
  #endif

  #ifdef LIGHT_SENSOR_ENABLED
    lightSensor.begin();
  #endif

  #ifdef RAIN_PORT
    pinMode(RAIN_PORT, INPUT);
  #endif

  uptime = 0;
}

float analogReadAverage(uint8_t pin) {
  for (uint8_t i = 0; i < 10; i++) analogRead(pin);

  uint16_t sum = 0;
  for (uint8_t i = 0; i < 10; i++) sum += analogRead(pin);
  return sum / 10.0;
}

void readTemperatures() {
  float dhtT = NAN, dhtH = NAN;
  float bmeT = NAN, bmeH = NAN;

  #if defined(DHT_PORT) && defined(DHT_TYPE)
    dhtT = dht.readTemperature();
    dhtH = dht.readHumidity();
  #endif
```

```
  #ifdef BME_ADDR
    bmeT = bme.readTemperature();
    bmeH = bme.readHumidity();
  #endif

  if (!isnan(dhtT) && !isnan(bmeT)) airT = dhtT, insideT = bmeT;
  else if (!isnan(dhtT)) airT = insideT = dhtT;
  else if (!isnan(bmeT)) airT = insideT = bmeT;

  if (!isnan(dhtH) && !isnan(bmeH)) airH = dhtH, insideH = bmeH;
  else if (!isnan(dhtH)) airH = insideH = dhtH;
  else if (!isnan(bmeH)) airH = insideH = bmeH;

  insideH = constrain(insideH, 0, 99);
  airH = constrain(airH, 0, 99);
}

static void getAllData() {
  readTemperatures();

  #ifdef LIGHT_SENSOR_ENABLED
    if     (lightSensor.measurementReady())     lightLevel     =
lightSensor.readLightLevel();
  #endif

  #ifdef SOIL_MOISTURE_PORT
    {
      float          moisture          =          100.0          -
(analogReadAverage(SOIL_MOISTURE_PORT) - 400.0) / 5.7;
      soilMoisture = constrain(moisture, 0.0, 100.0);
    }
  #endif

  #ifdef WATER_LEVEL_PORT
    enoughWater = analogRead(WATER_LEVEL_PORT) > 256;
  #endif

  #ifdef RAIN_PORT
    raining = !digitalRead(RAIN_PORT);
  #endif

  sendSensorValues();
}

void readAllSensors() {
  static uint64_t timer = 0;
  static uint32_t period = 1000;

  if (millis() - timer >= period) {
    timer = millis();
    switch (sensorReadingStage) {
      case SensorReadingStage::PowerOn:
        sensorReadingStage = SensorReadingStage::Read;
```

```
        period = 100;
        digitalWrite(SENS_VCC, 1);
        break;
      case SensorReadingStage::Read:
        sensorReadingStage = SensorReadingStage::PowerOff;
        period = 25;
        getAllData();
        break;
      case SensorReadingStage::PowerOff:
        sensorReadingStage = SensorReadingStage::UpdateOutputs;
        period = 10;
        digitalWrite(SENS_VCC, 0);
        break;
      case SensorReadingStage::UpdateOutputs:
        sensorReadingStage = SensorReadingStage::PowerOn;
        period = (long)settings.sensorPeriod * 1000;
        updateOutputs();
        break;
    }
  }
}

void updateTime() {
  static uint64_t timer = 0;

  if (millis() - timer > 1000) {
    timer += 1000;
    RTC.read(datetime);
    uptime++;
  }
}

// Файл settings.h

#pragma once
#include "pins.h"

#define DHT_PORT SENS1
#define DHT_TYPE DHT22

#define BME_ADDR 0x76

#define LIGHT_SENSOR_ENABLED
#define SOIL_MOISTURE_PORT SENS4
#define WATER_LEVEL_PORT SENS3
#define RAIN_PORT SENS2
#define LIGHT_PORT RELAY3
#define PUMP_PORT RELAY4

#define SERVO_MIN_PULSE 500
#define SERVO_MAX_PULSE 2500

#define DRIVER_LEVEL 1
```

```
#define LCD_BACKLIGHT_TIMEOUT 60
#define CRITICAL_HIGH_TEMPERATURE 50
#define CRITICAL_LOW_TEMPERATURE -25

#define WIRE_OVERCLOCK 400000
#define LCD_ADDR 0x27

// Файл web.cpp

#include "objects.h"
#include "functionality.h"

static void sendHTTPResponce(uint16_t code, String payload = "")
{
  Serial.print(code);
  Serial.print(';');
  Serial.println(payload);
}

template<typename T>
void sendTableEntry(String name, T value) {
  Serial.print(name);
  Serial.print(':');
  Serial.println(value);
}

void sendTableEntry(String name, String value) {
  Serial.print(name);
  Serial.print(":\"");
  Serial.print(value);
  Serial.println('"');
}

void sendTableEntry(String name, bool value) {
  Serial.print(name);
  Serial.print(':');
  Serial.println(value ? "true" : "false");
}

void sendSensorValues() {
  Serial.println(";sensors");
  sendTableEntry("airT", airT);
  sendTableEntry("airH", airH);
  sendTableEntry("insideT", insideT);
  sendTableEntry("lightLevel", lightLevel);
  sendTableEntry("soilMoisture", soilMoisture);
  sendTableEntry("enoughWater", enoughWater);
  sendTableEntry("raining", raining);
  Serial.println("end:");
}

void sendStates() {
  Serial.println(";states");
```

```
  sendTableEntry("flap1", flap1.state);
  sendTableEntry("flap2", flap2.state);
  sendTableEntry("door", door.state);
  sendTableEntry("light", light.state);
  Serial.println("end:");
}

static bool stringToBool(const String& str) {
  if (str == "true") return true;
  else if (str == "false") return false;
  else return str.toInt();
}

void updateServer() {
  while    (Serial.available()    &&    Serial.peek()    !=    ';')
Serial.read();
  if (Serial.peek() == ';') {
    Serial.read();
    String message = Serial.readStringUntil('\n');
    message.trim();
    if (message == "manual") {
      String invalid = "";
      while (true) {
        String name = Serial.readStringUntil(':');
        name.trim();
        if (name == "end") break;
        String value = Serial.readStringUntil('\n');
        value.trim();

        #define CONTROL(object, ...) \
          if (name == #object) { \
            if (value == "\"auto\"") object.manual = false; \
            else { \
              object.manual = true; \
              object.state = __VA_ARGS__; \
            } \
          }

        CONTROL(flap1, value.toInt())
        else CONTROL(flap2, value.toInt())
        else CONTROL(door, stringToBool(value))
        else CONTROL(light, stringToBool(value))
        else invalid = name;
      }
      EEPROM.put(SETTINGS_ADDRESS, settings);
      if  (invalid.length()  >  0)  sendHTTPResponce(422,  "Unknown
field: " + invalid);
      else sendHTTPResponce(200);
    } else if (message == "getManual") {
      sendTableEntry("flap1", flap1.manual);
      sendTableEntry("flap2", flap2.manual);
      sendTableEntry("door", door.manual);
      sendTableEntry("light", light.manual);
```

```
      Serial.println("end:");
    } else if (message == "settings") {
      String invalid = "";
      while (true) {
        String name = Serial.readStringUntil(':');
        name.trim();
        if (name == "end") break;
        String value = Serial.readStringUntil('\n');
        value.trim();

        if (name == "driveSpeed") settings.driveSpeed =
value.toInt();
        else if (name == "sensorPeriod") settings.sensorPeriod =
value.toInt();

        else if (name == "flapsTemperatureThreshold")
settings.flapsTemperatureThreshold = value.toFloat();
        else if (name == "doorTemperatureThreshold")
settings.doorTemperatureThreshold = value.toFloat();
        else if (name == "ventelationPeriod")
settings.ventelationPeriod = value.toInt();
        else if (name == "ventelationTime")
settings.ventelationTime = value.toInt();
        else if (name == "closeIfRain") settings.closeIfRain =
stringToBool(value);

        else if (name == "wateringMoistureThreshold")
settings.wateringMoistureThreshold = value.toInt();
        else if (name == "wateringTime") settings.wateringTime =
value.toInt();
        else if (name == "wateringTimeout")
settings.wateringTimeout = value.toInt();
        else if (name == "wateringPeriod") settings.wateringPeriod
= value.toInt();

        else if (name == "lightLevelThreshold")
settings.lightLevelThreshold = value.toInt();
        else invalid = name;
      }
      EEPROM.put(SETTINGS_ADDRESS, settings);
      if (invalid.length() > 0) sendHTTPResponce(422, "Unknown
field: " + invalid);
      else sendHTTPResponce(200);
    } else if (message == "getSettings") {
      sendTableEntry("driveSpeed", settings.driveSpeed);
      sendTableEntry("sensorPeriod", settings.sensorPeriod);

      sendTableEntry("flapsTemperatureThreshold",
settings.flapsTemperatureThreshold);
      sendTableEntry("doorTemperatureThreshold",
settings.doorTemperatureThreshold);
      sendTableEntry("ventelationPeriod",
settings.ventelationPeriod);
```

```
      sendTableEntry("ventelationTime",
settings.ventelationTime);
      sendTableEntry("closeIfRain", settings.closeIfRain);

      sendTableEntry("wateringMoistureThreshold",
settings.wateringMoistureThreshold);
      sendTableEntry("wateringTime", settings.wateringTime);
      sendTableEntry("wateringTimeout",
settings.wateringTimeout);
      sendTableEntry("wateringPeriod", settings.wateringPeriod);

      sendTableEntry("lightLevelThreshold",
settings.lightLevelThreshold);
      Serial.println("end:");
    } else if (message == "factoryReset") {
      settings = Settings();
      EEPROM.put(SETTINGS_ADDRESS, settings);
      sendHTTPResponce(200);
    } else {
      sendHTTPResponce(500, "Unknown operation: " + message);
    }
  }
}
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Программно-аппаратный комплекс для управления теплицей. Чертеж электромонтажный

ПРИЛОЖЕНИЕ В
(обязательное)

Спецификация

ПРИЛОЖЕНИЕ Г
(обязательное)

Перечень элементов

ПРИЛОЖЕНИЕ Д
(обязательное)

Ведомость документов