Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____О.В. Непомнящий
«___»_____2024 г.

БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Автоматизированная гидропонная ферма для низкорослых растений

Руководитель	_____	_____	ст. преподаватель	С.Л. Верхошенцева
	подпись	*дата*	*должность, ученая степень*	
Выпускник	_____	_____		Л.С.Безе
	подпись	*дата*		
Нормоконтролёр	_____	_____	ст. преподаватель	С.Л. Верхошенцева
	подпись	*дата*	*должность, ученая степень*	

Красноярск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Автоматизированная гидропонная ферма для низкорослых растений» содержит 47 страниц текстового документа, 16 рисунков, 7 использованных источников.

ГИДРОПОННАЯ ФЕРМА, ДАТЧИКИ, АНДРОИД, ПРИЛОЖЕНИЕ.

Цель работы является разработка автоматизированной установки гидропонной фермы с системой мониторинга с помощью мобильного приложения.

Работа состоит из трёх основных частей.

В первой главе произведён анализ предметной области, рассмотрены аналоги, сформулированы требования к системе.

Во второй главе выбран и описан аппаратный состав системы.

В третьей главе описывается поэтапная реализация програмного обеспечения системы и реализация мобильного приложения.

Результатом квалификационной работы является автоматизированная установка гидропонной фермы с системой мониторинга с помощью мобильного приложения.

Графическая часть работы представлена в презентации Microsoft PowerPoint, которая описывает весь функционал приложения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире сельское хозяйство играет важную роль в обеспечении пищевой безопасности и устойчивого развития. Однако, с ростом населения и увеличением городской застройки, доступ к плодородным почвам и пространству для традиционного земледелия становится все сложнее. В таких условиях гидропоника - метод выращивания растений без использования почвы, находит все большее применение.

Целью данной бакалаврской работы является разработка автоматизированной установки гидропонной фермы с системой мониторинга гидропонной фермой с помощью мобильного приложения. Такая система позволит упростить и оптимизировать процесс выращивания растений, а также обеспечить контроль за всеми необходимыми параметрами, такими как температура, влажность, яркость освещения, уровня воды в резервуаре и жесткости питательного раствора.

В рамках выполнения работы проведен анализ существующих решений автоматизированных систем управления гидропонных ферм, изучены основные принципы гидропоники и разработаны требования к мобильной системе мониторинга. Представлено описание архитектуры гидропонной фермы. Разработаны модули и функционал мобильного приложения.

Результатом работы является создание автоматизированной гидропонной фермы, а также эффективной и удобной мобильной системы мониторинга гидропонной фермы, способной значительно улучшить процесс выращивания растений и повысить его эффективность. Кроме того, такая система может быть полезна как для профессиональных фермеров, так и для любителей, позволяя им получить необходимые для выращивания растений знания, облегчить управление гидропонной установки, сократить время на ее обслуживание, тем самым позволяя получать высокий урожай качественных овощей и зелени прямо в своей квартире или на участке.

1 Обзор и анализ предметной области

1.1 Типы ферм для домашнего использования и особенности гидропонных систем.

Существует множество методов выращивания растений, но гидропоника, аэропоника и аквапоника стали инновационными методами, обеспечивающими превосходную эффективность, качество и урожайность.

1.1.1 Аэропоника

Аэропоника - это инновационный метод выращивания, при котором корни растений подвешиваются в воздухе и опрыскиваются водой, богатой питательными веществами. Этот метод максимизирует воздействие кислорода, способствуя быстрому росту и высокой урожайности.

Обработка корней растений воздушным туманом позволяет максимально использовать питательные вещества, тратить меньше воды и избежать болезней и вредителей. Однако, этот метод также требует высоких затрат на оборудование и поддержание системы, постоянного контроля pH и уровня питательных веществ, а также сложной вентиляции и освещения [1]

1.1.2 Аквапоника

Аквапоника - вид выращивания растений сочетающий гидропонику с аквакультурой (выращиванием рыбы) для создания симбиотической экосистемы. Рыбные отходы обеспечивают растения необходимыми питательными веществами, а растения очищают воду для рыб.

Выращивание растений этим методом позволяет уменьшить использование синтетических питательных веществ и повысить эффективность использование воды, также аквапоника наиболее устойчивый

и экологически чистый метод. Однако, этот метод требует большой сложности в управлении системой из-за необходимости контроля как растений, так и рыб, постоянного обслуживания системы для поддержания баланса между растениями и рыбами, а также высоких начальных затрат на оборудование и рыбьи популяции [1].

1.1.3 Гидропоника

Гидропоника - это метод выращивания растений без почвы, при котором растения размещают в богатой питательными веществами воде, что позволяет им поглощать питательные вещества непосредственно через корни.

Использование гидропонной системы имеет несколько преимуществ: увеличение урожайности и скорости роста растений, экономия воды и отсутствие болезней и вредителей. Однако, этот метод требует высоких затрат на начальное оборудование и поддержание системы, а также постоянного контроля pH и уровня питательных веществ, и зависит от электричества для работы насосов и освещения [1].

1.1.3.1 Система капельного полива

Система капельного полива относится к автоматическим системам и считается одной из самых эффективных для выращивания широкого спектра культур.

Схема повторяет систему периодического затопления: есть резервуар с гидропонным раствором и контейнер с растениями. Насос по заданному в таймере расписанию направляет жидкость по небольшим шлангам, откуда она непрерывно капает под корни растений. Лишняя жидкость поступает обратно в резервуар [1].

1.1.3.2 Система фитильного полива

Система фитильного полива – это классический метод, применяемый в питомниках, где культивируют медленно растущие декоративные комнатные растения.

Главное преимущество этой системы – автономность до 2-3 недель при достаточном запасе воды или питательного раствора. Не требуется ни насос, ни электричество, так как жидкость сама поступает по фитилю к корням растения из резервуара. Один конец фитиля опускают в резервуар, другой подводят к корням растения. Однако при такой системе нельзя вырастить крупное растение или быстрорастущее, требующее обильного полива [1].

1.1.3.3 Система глубоководных культур или DWC

Система глубоководных культур (DWC, Deep Water Culture) идеально подходит для культивирования салата и других небольших культур, требующих постоянного полива. Это одна из самых эффективных и технически доступных систем, которую можно собрать в домашних условиях без сложного оборудования.

Растения выращиваются в специальных горшках с отверстиями, через которые корни растения прорастают сквозь субстрат, и погружаются в питательный раствор, обогащенный кислородом. Корневая система развивается в светонепроницаемом резервуаре с раствором, а стебель и листва — на открытом воздухе под осветительной системой [1].

1.1.3.4 Система периодического затопления

Система состоит из резервуара с питательным раствором и контейнером с растениями, корни растений и субстрат, в котором они

развиваются, не постоянно находятся в воде, а несколько раз в день затапливаются и осушаются по заданному расписанию.

Основную работу выполняет насос, который перекачивает раствор из резервуара в контейнер с находящимися в нем растениями, и таймер, который управляет насосом. Такая система должна быть полностью автоматизирована. Насос заполняет контейнер, затем жидкость стекает через отверстие обратно в резервуар [1].

1.1.3.4 Система питательного слоя

Система питательного слоя может быть довольно компактной и использоваться в небольших помещениях. Установка состоит из резервуара с подготовленным раствором и емкости с растениями, расположенной под наклоном.

Питательный раствор подается насосом по трубке из резервуара и течет по дну емкости ровным слоем, стекая обратно в резервуар и замыкая цикл. Кончики корней растений находятся в постоянном контакте с питательным слоем, непрерывно впитывая микроэлементы и полезные вещества, и одновременно корневая система насыщается кислородом [1].

1.2 Существующие аналоги

Для создания наиболее эффективной системы автоматизации гидропонной установки стоит рассмотреть уже имеющиеся автоматизированные гидропонные установки и выделить их преимущества и недостатки.

1.2.1 Гидропонные комплексы от “Агроновия”

Компания “Агроновия” специализируется на проектировании и производстве гидропонных систем для выращивания различных культур в промышленных масштабах.

Компания изготавливает на заказ гидропонные установки с автоматизированной системой управления мощностью от 500 кг в сутки (Рисунок 1.1).

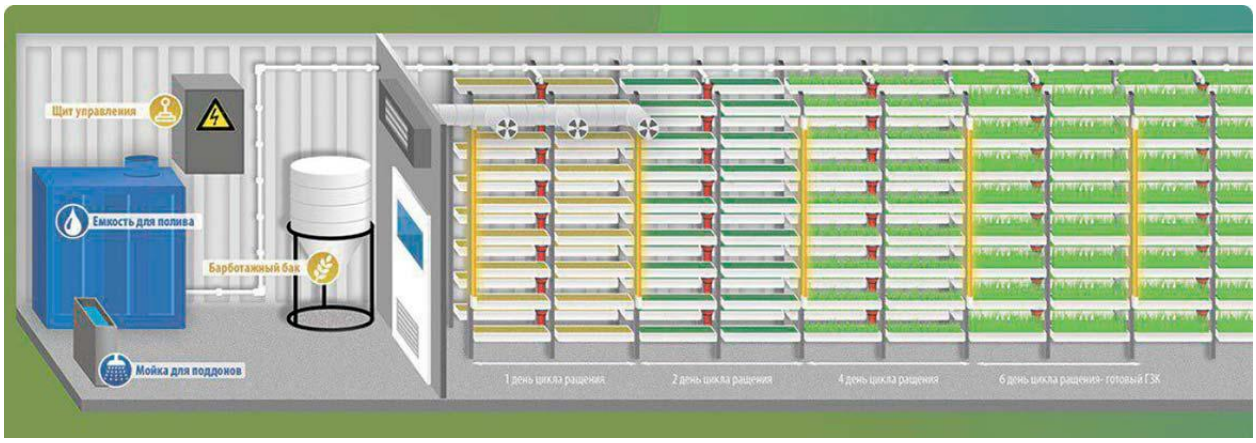


Рисунок 1.1 – Гидропонный комплекс УЗК-500

Недостатком является, отсутствие моделей небольшой мощности для личного пользования и маленьких ферм, так как компания изготавливает установки для сельскохозяйственных нужд: выращивание корма для скота, выращивание растений на крупных фермах для продажи, и для компаний общепита: рестораны, кафе, магазины [3].

1.2.2 Оборудование для автоматизации гидропонных систем от “EZplant”

Компания предоставляет комплект для автоматизации гидропонных систем через мобильное приложение. В базовый комплект стоимостью 39000

рублей входят насосы нормализации EC и pH с соответствующими датчиками, емкостные датчики уровня (для основного резервуара, подключаются к розетке насоса циркуляции), а также датчик температуры, влажности и освещённости (Рисунок 1.2) [2].

Комплектация:

- 1 х контроллер автоматизации гидропонной системы EZplant;
- 1 х датчик кислотности жидкости (pH-метр) без дисплея, RS485 / Modbus;
- 1 х датчик минерализации (TDS/EC-метр) без дисплея, RS485 / Modbus;
- 1 х блок климатический (температура, влажность, освещенность), RS485 / Modbus;
- 2 х блок розетка RS485 / Modbus, каждая со шнуром IEEE C13 - Type F;
- 1 х блок на 3 перистальтических насоса RS485 / Modbus;
- 1 х блок на 2 перистальтических насоса RS485 / Modbus;
- 6 х кабель витой пары патч-корд;
- 1 х буферные растворы для калибровки pH-метра;
- 1 х калийная соль для приготовления буферных растворов калибровки TDS/EC-метра;
- 3 х бесконтактный датчик уровня жидкости XKC-Y25-V.

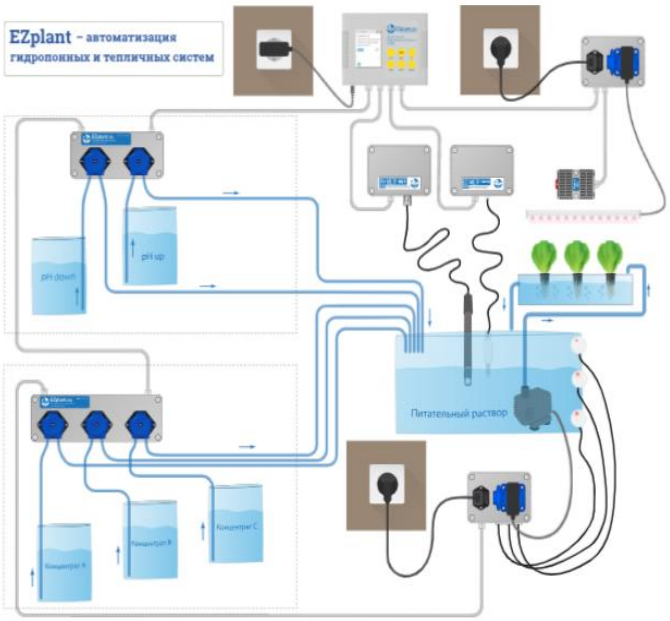


Рисунок 1.2 – Пример подключения

Комплект позволяет осуществлять онлайн мониторинг при помощи сервиса https://ezplant.ru и приложений для смартфонов Android и iPhone. На рисунке 1.3 представлен интерфейс приложения EZplant.

- поддерживать оптимальные уровни кислотности и минерализации питательного раствора (pH/EC);
- отслеживать температуру и влажность;
- осуществлять контролируемое освещение растений;
- контролировать циркуляцию питательного раствора;
- управлять аэрацией и обдувом растений;
- расширять функционал по мере необходимости при помощи дополнительный блоков.

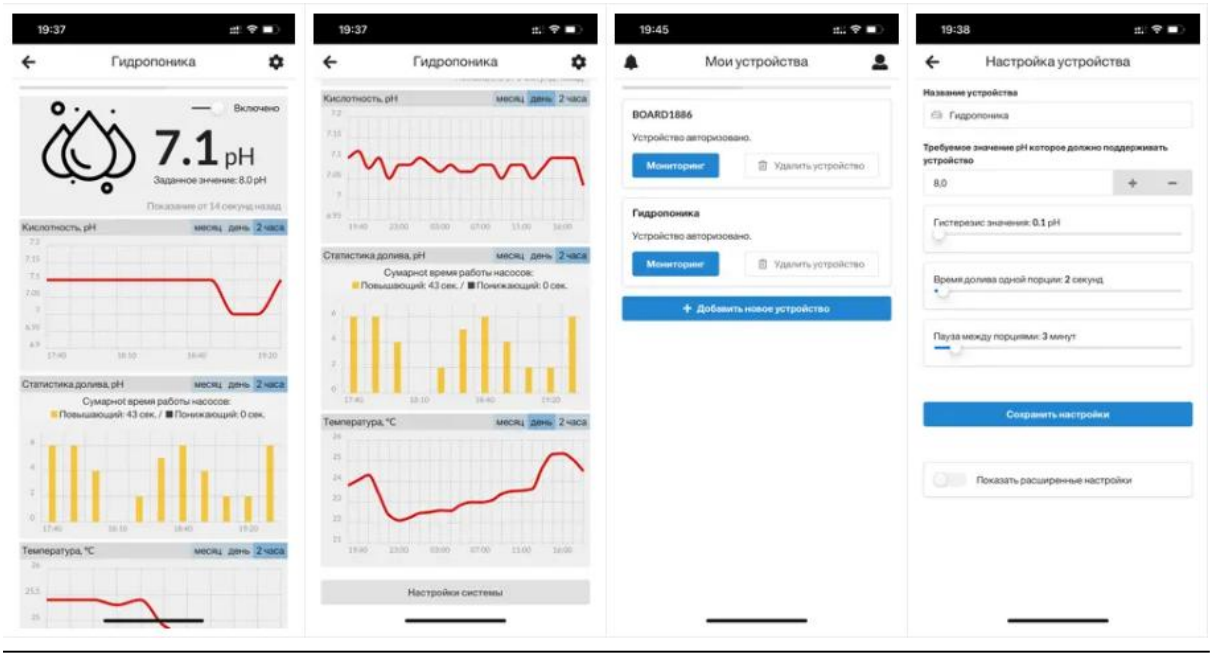


Рисунок 1.3 – Приложение EZplant

Компания предоставляет решение, имеющее много преимуществ, но также есть несколько недостатков. Для использования комплекта необходимо уже иметь гидропонную установку, в итоге помимо затрат на сам комплект нужно еще купить саму установку. Также использование оборудования от EZplant, ограничивает пользователя в выборе дополнительных модулей. Приложение позволяет в основном осуществлять мониторинг, управление ограничивается подачей раствора и его кислотностью.

1.2.3 Микроферма RAWMID Dream Sprouter SDM-02

Компания RAWMID предоставляет установку для выращивания растений высотой до 15 см в домашних условиях стоимостью 12900 рублей (Рисунок 1.4). На крышке находится панель управления с 2 кнопками, которые позволяют включать/выключать автоматический полив и освещение. Генератор тумана автоматически осуществляет полив каждые 3 часа.

Комплектация:

- 1 х корпус размерами 48×27×27 см;
- 2 х лоток площадью 320 см²;
- 1 х генератор тумана;
- 1 х крышка;
- 1 х светодиодная подсветка;
- 1 х адаптер питания;
- 1 х мерный стакан.

Рисунок 1.4 – Микроферма RAWMID Dream Sprouter SDM-02

Компания предоставляет решение, имеющее несколько преимуществ: компактный размер, простота использования, небольшая цена. Недостатками данной установки являются ограниченный размер, который не позволяет выращивать растений выше 15 см, небольшой функционал, который нельзя увеличить и настроить под себя.

1.3 Выводы по 1 главе

В первой главе были проанализированы несколько методов и систем выращивания растений в домашних условиях. В результате был выбран

оптимальный метод выращивания при помощи системы питательного слоя, так как данный метод подходит для большинства растений и наиболее прост в реализации. Был проведен анализ аналогов автоматизированных гидропонных установок, а именно Микроферма RAWMID Dream Sprouter SDM-02, оборудование для автоматизации гидропонных систем от “EZplant” и гидропонные комплексы от “Агроновия”. В результате были выявлены их преимущества и недостатки.

2 Разработка системы

2.1 Разработка функциональной схемы

Разработанная микроконтроллерная системы будет содержать три блока данных: блок датчиков, блок ввода/вывода, обработки и управления, и блок исполнительных устройств.

Блок датчиков предназначен для считывания и сбора необходимых данных уровня воды, освещенности, tds раствора, температуры и влажности воздуха.

Блок ввода/вывода, обработки и управления предназначен для исполнения алгоритма на основе данных из блока датчиков. Устройство управления – микроконтроллер.

Блок исполнительных устройств получает информацию от блока ввода/вывода, обработки и управления и выводит данные полученные с датчиков на LCD-дисплей. На рисунке 2.1 изображена функциональная схема системы.

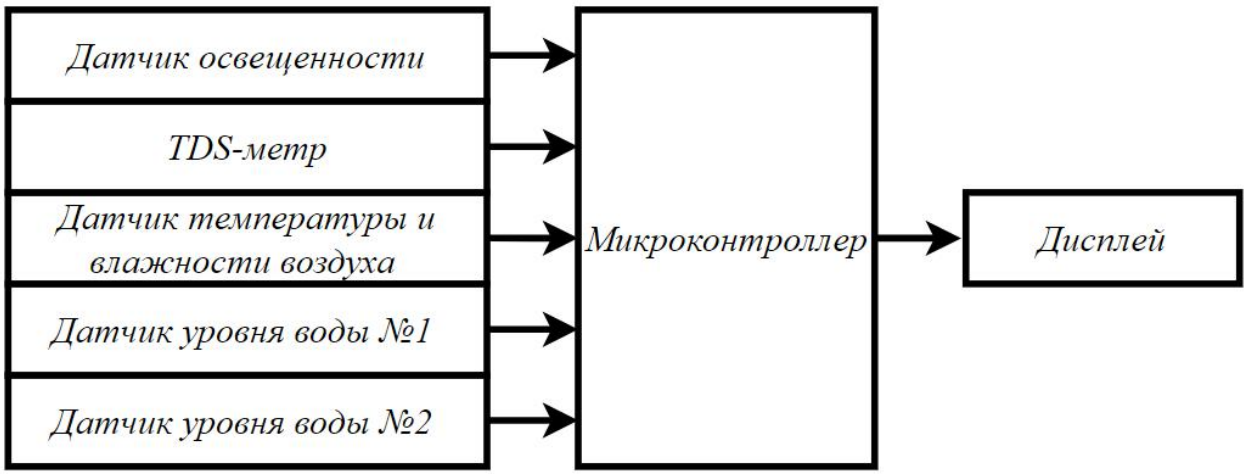


Рисунок 2.1 – Функциональная схема системы

На основе разработанной функциональной схемы был определен следующий список компонент аппаратного состава для дальнейшего сбора лабораторного прототипа.

2.1.1 Микроконтроллер

В качестве используемого в данной системе микроконтроллера была выбрана плата разработчика ESP32-DevKitC V4 на базе ESP32-WROOM-32D. Данный контроллер имеет высокопроизводительный совмещённый модуль от компании Espressif с поддержкой Wi-Fi, BT и BLE. Плата снабжена таким необходимым функционалом как стабилизатор напряжения на 3,3 вольта, USB-UART преобразователь CP2102, USB разъем для прошивки и питания платы. Плата имеет 32-битный двух ядерный микропроцессор - Xtensa LX6, с частотой - 160-240 МГц, ОЗУ — 520 Кб, ПЗУ — 448 Кб, RTC таймер с 16 Кб ОЗУ, внешнюю флеш-память 4-16 Мб, питание 2.2 В — 3.6 В, WiFi 802.11n 2.4 Гц с максимальной скоростью 150 Мбит/сек, шифрование WPA/WPA2/WPA2-Enterprise/Wi-Fi Protected Setup WPS, Bluetooth v4.2 BR/EDR и BLE. Плата имеет WiFi модуль, что позволяет передавать информацию с блока датчиков в пользовательское приложение. Стоит отметить, что плата имеет не высокую по цене.

2.1.2 Датчик уровня воды

Для использования в системе выбран аналоговый датчик уровня воды HW-038. Данный датчик воды – погружной. Чем больше погружение датчика в воду, тем меньше сопротивление между двумя соседними проводами. Изображение датчика уровня воды HW-038 изображен на рисунке 2.2. Предполагается использовать несколько датчиков, которые предназначены для контроля уровня жидкости в резервуаре с водой.

Рисунок 2.2 – Пример схемы датчика уровня воды

2.1.3 Водяной насос

QR50B — бесщеточный водяной насос постоянного тока DC5V 4,8 Вт с максимальной высотой подъема равной 3 метрам. Мощность равна 300 литрам в час (Рисунок 2.3).

В системе насос используется для циркуляции питательного раствора по всей системе.

Рисунок 2.3 – Водяной насос QR50B

2.1.4 Жидкокристаллический дисплей

В качестве дисплея будет использоваться HD44780, обладающий разрешением экрана 16x2 (Рисунок 2.4).

Рисунок 2.4 – Жидкокристаллический дисплей HD44780

2.1.5 Датчик температуры и влажности

Датчик температуры и влажности AHT10 с диапазоном измерения температуры -40 ~ +85 ℃ ± 0,3 ℃, диапазон измерения влажности 0 ~ 100% ± 2% (Рисунок 2.5).

Рисунок 2.5 – Датчик температуры и влажности AHT10

2.1.6 Датчик минерализации жидкости

Плата для разработки TDS Meter V1.0 используется для измерения растворенных твердых веществ (Рисунок 2.6).

Рисунок 2.6 – Датчик минерализации жидкости TDS Meter V1.0

2.2 Выводы по 2 главе

Во второй главе была разработана функциональная схемы, определен аппаратный состав. В качестве используемого в данной системе микроконтроллера была выбрана плата разработчика ESP32-DevKitC V4 на базе ESP32-WROOM-32D, так как плата имеет низкую стоимость и Wifi модуль, что позволяет передавать данные с микроконтроллера на сервер.

3 Моделирование

3.1 Алгоритм функционирования системы

Инициализация: В начале программы выполняется инициализация датчиков уровней воды, температуры и влажности воздуха, освещенности, TDS-метра, насоса, и других компонентов, таких как LCD-дисплей.

Подключение к wifi: Перед основным циклом, происходит проверка подключения микропроцессора к сети wifi, пока микропроцессор не подключится к wifi программа не входит в основной цикл.

Основной цикл: Программа входит в основной цикл. На главном экране дисплея находится актуальная информация с датчиков температуры, влажности, освещенности, TDS.

Считывание данных с датчиков: Затем программа инициирует считывание с датчика освещенности, датчиков уровня воды, датчика уровня и влажности воздуха, tds-метра.

Вывод данных: После считывания с датчиков, полученные данные передаются на облачный сервер Thingspeak по api-key, и выводятся на LCD дисплей.

Инциализация приложения: При открытии мобильного приложения происходит инициализация приложения, открытие главной страницы.

Получение данных с сервера: При открытии мобильного приложения запускается страница фоновой активности, внутри которой реализованы запрос данных с сервера, запись их в памяти устройста.

Отправка уведомления: После получения и обновления данных с сервера, данные сравниваются с пороговыми значениями. При превышении пороговых значений на устройство отправляется уведомление. Уведомление содержит в себе информацию о том, что данные превышены и какие данные превышены.

Переход на другие страницы: Переход по страницам происходит по нажатии на кнопку в меню, находящимся на каждой страницу внизу экрана.

Изменение пороговых значений: Изменение пороговых значений происходит на странице настроек. Увеличить или уменьшить значение можно нажав на кнопки «+» и «–», соответственно.

Изменение данных для подключения к серверу: Изменить API-key и channel сервера, с которого запрашиваются данные можно на странице личного кабинета.

3.2 Сборка лабораторной установки

Для тестирования готовой автоматизированной системы было принято решение собрать гидропонную установку. Для начала была разработана схема сборки, включая электронные компоненты, и собрана гидропонная установка. Гидропонная установка была собрана вручную из дешевых и доступных материалов. На рисунке 3.1 представлена схема гидропонной установки, а на рисунке 3.2 представлена фотография готовой гидропонной установки.

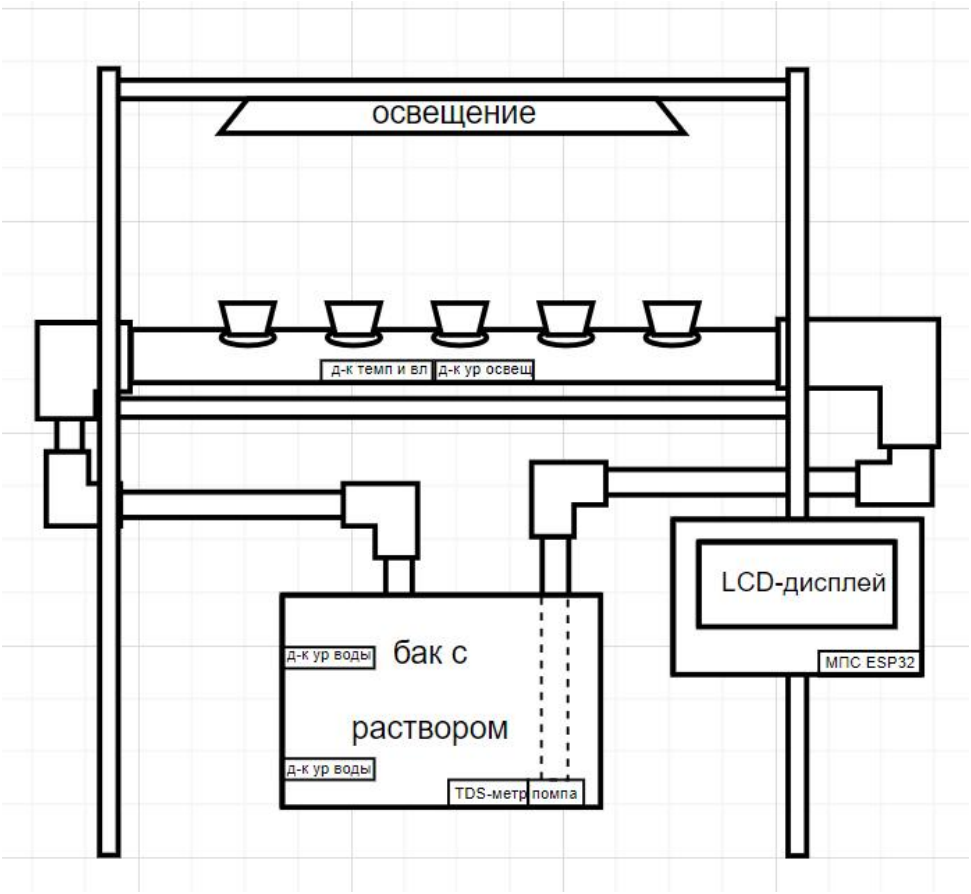


Рисунок 3.1 – Схема сборки гидропонной установки

Рисунок 3.2 – Собранная гидропонная установка

После сборки гидропонной установки была начата сборка аппаратного обеспечения автоматизированной гидропонной фермы. Для начала была разработана принципиальная схема, представлена в приложении.

Далее была начата сборка аппаратного состава по принципиальной схеме. На первом этапе все компоненты подключаются к микроконтроллеру с использованием макетной платы.

Датчик температуры и влажности имеет 4 вывода, SDA подключается к SDA на микроконтроллере, SCL подключается к SCL, питание (VCC) подключается к 5V, земля (GND) подключается к GND.

Датчик освещенности (фоторезистор) имеет 2 вывода, один вывод подключается к 5V через резистор, второй вывод подключается к аналоговому входу.

Датчики уровня воды имеют 3 вывода, питание (VCC) подключается к 5V, земля (GND) подключается к GND, сигнальный вывод подключается к цифровому входу.

TDS-метр имеет 3 вывода, питание (VCC) подключается к 5V, земля (GND) подключается к GND, сигнальный вывод (Analog Out) подключается к аналоговому входу.

LCD-дисплей имеет 4 вывода, SDA подключается к SDA на микроконтроллере, SCL подключается к SCL, питание (VCC) подключается к 5V, земля (GND) подключается к GND.

3.3 Программирование

Для управления автоматизированной гидропонной фермой было разработано программное обеспечение, включающее следующие основные функции:

- инициализация компонентов: настройка всех входов и выходов микроконтроллера, инициализация библиотек для работы с датчиками и дисплеем;
- считывание данных с датчиков: чтение данных с датчика температуры и влажности, измерение уровня освещенности, получение показаний TDS-метра, определение уровня воды в резервуаре;
- обработка данных: проверка данных на корректность, вычисление средних значений для повышения точности измерений;
- вывод информации на LCD-дисплей: отображение текущих параметров: температура, влажность, уровень освещенности, TDS и уровень воды, обновление информации в реальном времени;
- передача данных на облачный сервер.

На рисунке 3.3 изображен алгоритм функционирования системы. Листинг программы представлен в приложении А.

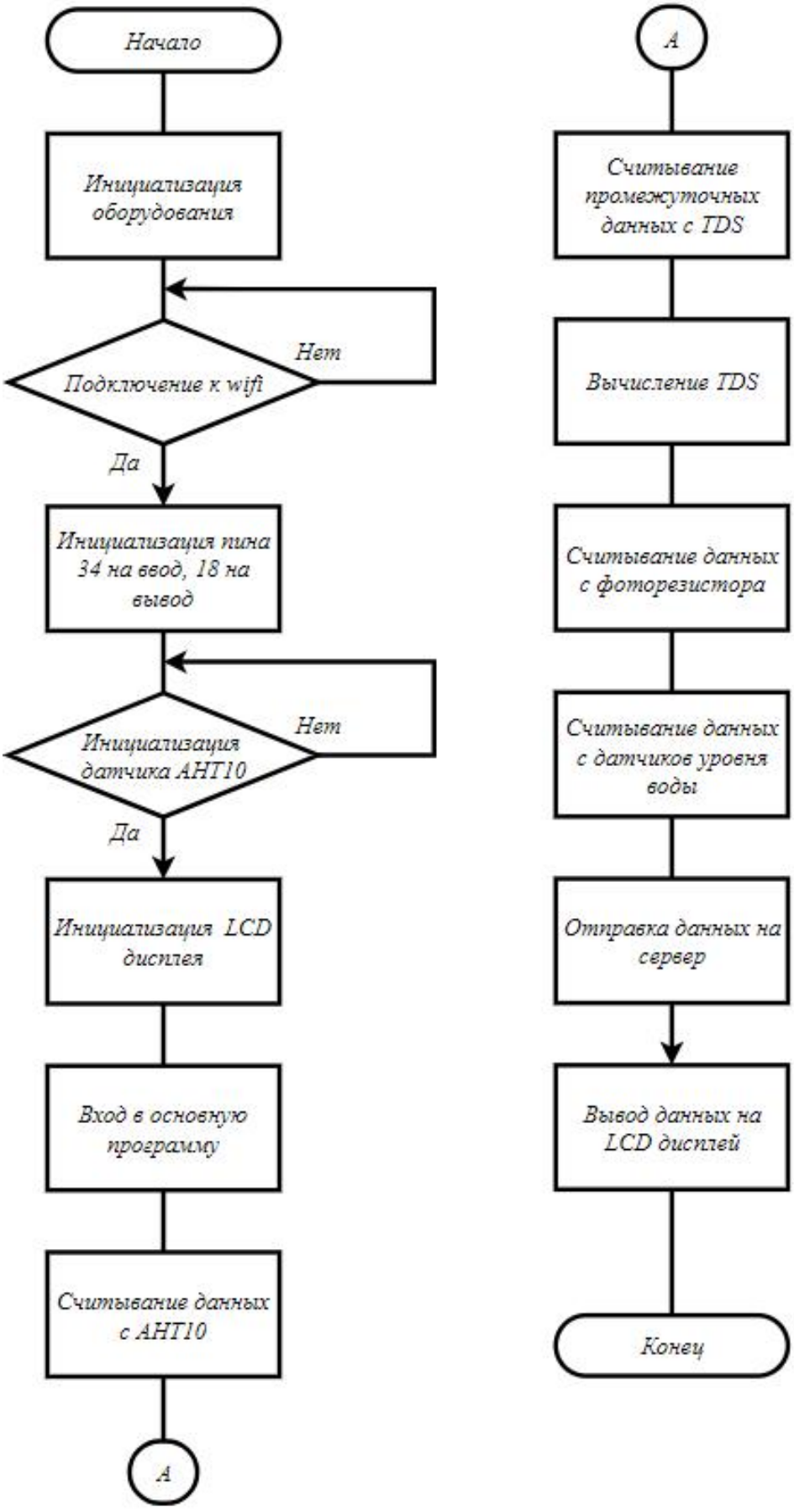


Рисунок 3.3 – Алгоритм функционирования системы

3.4 Разработка мобильного приложения

После сборки аппаратного состава и его программного обеспечения, было разработано мобильное приложение удаленного управления системы. Был разработан необходимый функционал, а также пользовательский интерфейс приложения. На рисунке 3.4 изображен алгоритм функционирования мобильного приложения.

В функционал приложения входят: регулирование допустимых значений, отправка уведомлений о выходе данных получаемых с датчиков за пределы допустимых значений.

Программный код приложения написан на языке программирования Kotlin под операционную систему Android в среде разработки Android Studio. Приложение содержит 4 страницы активностей.

«Главная страница» содержит данные с датчиков температуры и влажности, освещенности, уровней воды и TDS-метра (Рисунок 3.5 а). Данные запрашиваются с сервера, хранятся на устройстве и обновляются каждые 5 минут. Листинг программы активности «Главная страница» представлен в приложении Б.

«Страница настроек» содержит поля пороговых значений, пороговые значения изменяются при нажатии на кнопки «+» и «-» (Рисунок 3.5 б), и хранятся на устройстве. Листинг программы активности «Страница настроек» представлен в приложении В.

«Личный кабинет» содержит два текстовых поля, в которые можно внести значения API-key и channel облачного сервера (Рисунок 3.5 в). Листинг программы активности «Личный кабинет» представлен в приложении Г.

«Страница фоновой активности» не имеет интерфейса пользователя, запускается в фоновом режиме и выполняет запросы на сервер каждые 5 минут, сравнивает полученные данные с пороговыми значениями и при

превышении их отправляет на устройство уведомление (Рисунок 3.6). Листинг программы активности «Личный кабинет» представлен в приложении Д.

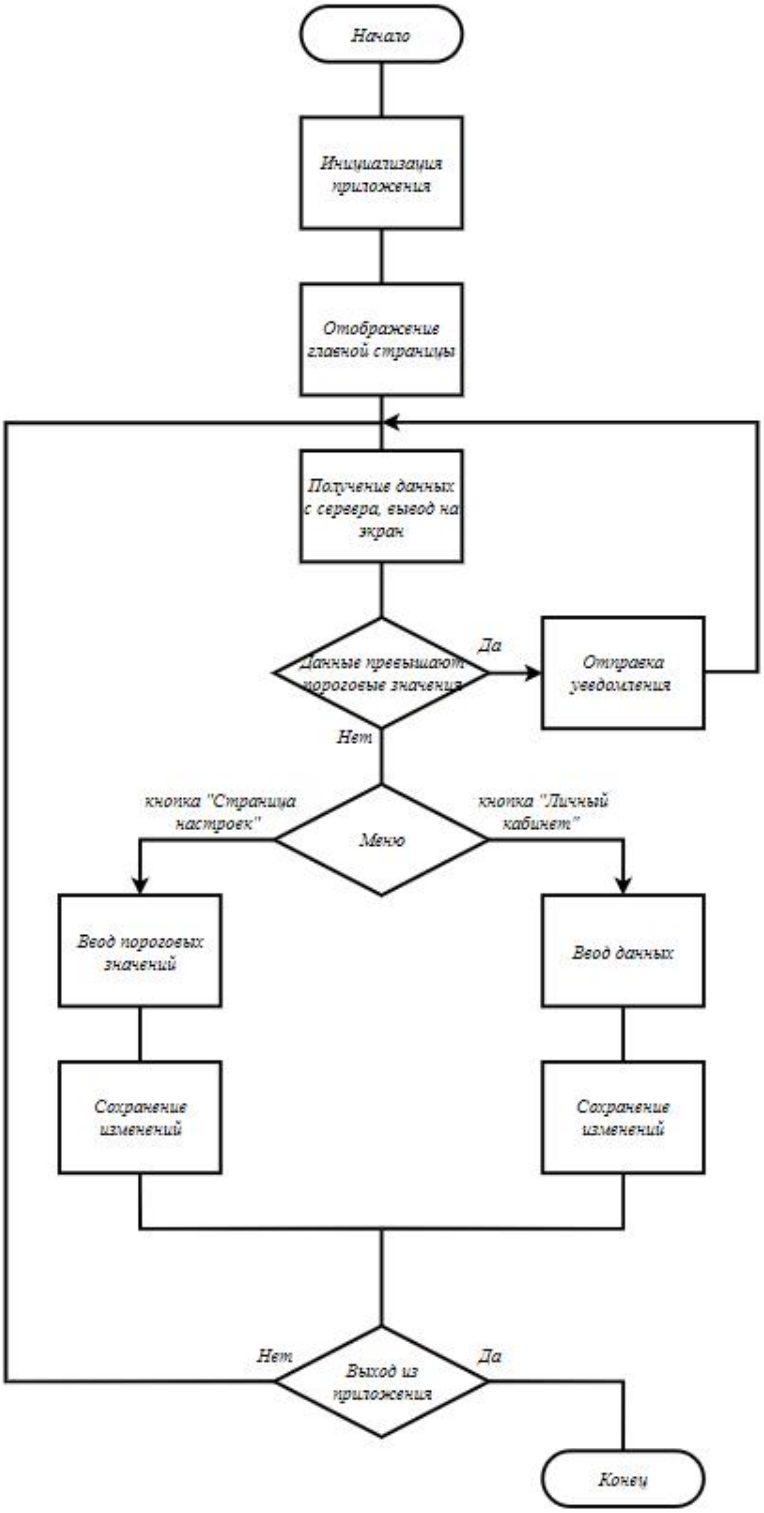


Рисунок 3.4 – Алгоритм функционирования системы

а) б) в)

Рисунок 3.5 – Интерфейс приложения:а) «Главная страница»; б) «Страница настроек»; в) «Личный кабинет»

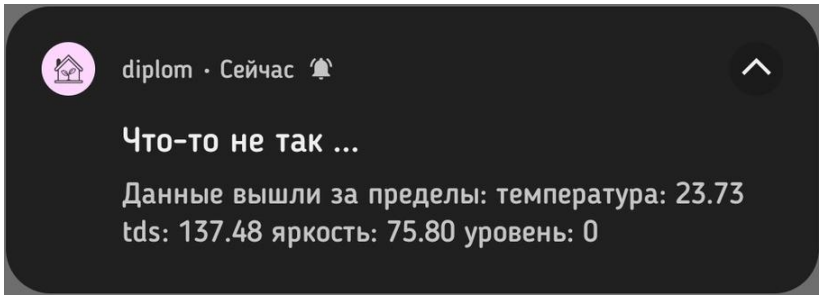


Рисунок 3.6 – Уведомление приложения

3.5 Тестирование

После выполнения основной работы и проверке работоспособности, проведено тестирование правильности работы аппаратного состава, программного обеспечения, мобильного приложения, а также передачу данных с микроконтроллера на сервер и с сервера в приложение.

Тестирование системы проходило в несколько этапов: проверка отдельных компонентов системы, тестирование взаимодействия между компонентами, проверка соответствия системы заданным функциональным требованиям, проверка удобства использования интерфейса и его корректного функционирования, проверка работы системы на реальных растениях, в данном случае - листья салата.

При тестировании компонентов системы для начала было проведена проверка точности показаний датчика температуры и влажности, датчика освещенности, TDS-метра. Для проверки точности реакции датчиков уровня воды было проведено тестирование с измерением скорости реакции датчика на изменение уровня жидкости. LCD-дисплей был протестирован на отображении информации на экране и корректность работы интерфейса.

Далее было протестировано подключение всех компонентов к микроконтроллеру, обмена данными между микроконтроллером и датчиками и тестирование отображение данных с датчиков на LCD-дисплее.

На следующем этапе протестированы корректность и удобство пользования приложением, правильность получения данных с сервера и получения уведомлений.

На последнем этапе тестирования происходит посадка листьев салата в гидропонную ферму, наблюдение за ростом растений, а также мониторинг и регистрация показаний датчиков и их влияние на рост растений.

3.6 Выводы по 3 главе

В третьей главе была разработана принципиальная схема подключения электронных компонентов. По принципиальной схеме была проведена сборка аппаратной части системы, а также разработано программное обеспечение. В

процессе сборки аппаратного состава системы все элементы были подключены и протестированы на корректность работы. Разработанное программное обеспечение обрабатывает все необходимые функции. Созданное мобильное приложения для удаленного мониторинга и управления гидропонной фермой, предоставляет пользователю удобный и необходимый функционал для просмотра показаний датчиков и имеет корректную систему уведомлений. Результаты тестирования автоматизированной гидропонной фермы показали необходимый уровень работоспоспособности, стабильности и корректности работы системы.

ЗАКЛЮЧЕНИЕ

На первом этапе выполнения работы был проведен анализ существующих решений автоматизированных систем управления гидропонных ферм в домашних условиях, изучены основные принципы гидропоники и разработаны требования к мобильной системе управления. В результате был выбран оптимальный метод выращивания при помощи системы питательного слоя, так как данный метод подходит для большинства растений и наиболее прост в реализации.

На втором этапе было разработано и представлено описание архитектуры гидропонной фермы, а также модули и функционал мобильного приложения. Был определен аппаратный состав лабораторного прототипа.

На третьем этапе была проведена сборка и программирование лабораторного прототипа автоматизированной гидропонной фермы, разработка удобного мобильного приложения для мониторинга гидропонной фермы, что позволит своевременно реагировать на изменение показателей фермы и значительно улучшит процесс выращивания растений.

В работе было использовано современное программное обеспечение и технологии разработки мобильных приложений, а также проведено тестирование для проверки эффективности и надежности системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гидропонная система - виды, особенности, принцип работы // Агроновия: официальный сайт. – URL: https://agronovia.ru/gidroponnaja-sistema-vidy-osobennosti-princip-raboty/ (дата обращения: 25.12.2023).

2. Компания «EZplant» // EZplant: официальный сайт. – URL: https://ezplant.ru/ (дата обращения: 25.12.2023).

3. Компания «Агроновия» // Агроновия: официальный сайт. – URL: https://agronovia.ru/ (дата обращения: 25.12.2023).

4. Компания «RawMID» // RawMID: официальный сайт. – URL: https://sprouter.rawmid.com (дата обращения: 25.12.2023).

5. Подключение микроконтроллера. Ликбез // EASY ELECTRONICS: сайт. – URL: - http://easyelectronics.ru/podklyuchenie-mikrokontrollera-likbez.html

6. HC-SR04 Ultrasonic Sensor Module User Guide / Handson Technology // HandsOnTech: сайт – URL: - https://handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf

7. СТУ 7.5-07-2021. Стандарт университета. Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности // Сибирский федеральный университет : официальный сайт. – URL: https://about.sfu-kras.ru/node/8127 (дата обращения: 07.06.2024)

ПРИЛОЖЕНИЕ А
ESP.ino

```
#include <HardwareSerial.h>
#include <Adafruit_AHT10.h>
#include <HTTPClient.h>
#include <WiFi.h>
#define TdsSensorPin 34       //пин tds метра
#define LightSensorPin 35     // пин фоторезистора
#define V3 3.3              // аналоговое значение питания в вольт
#define scount  30
#define PowerWaterSensorPin 18  // пин с которого подается питание на watersensor
#define WaterSensorPin 32    //пин вывода данных с water sensor
#define WaterSensor2Pin 33
HardwareSerial LCDSerial(1);
int analogBuffer[scount];     // массив данных с tds
int analogBufferTemp[scount];
int analogBufferIndex = 0;
int counter = 0;
int copyIndex = 0;
int WaterSensVal = 0;
int WaterSens2Val = 0;
const char* ssid = "Redmi Note 12";   // данные wifi
const char* password = "43825690";
const char* server = "api.thingspeak.com";   //данные сервера
const String api_key = "1V23AZF468OVELSU";
sensors_event_t humidity;
sensors_event_t temp;
float averageVoltage;
float tdsValue;
```

```
float temperature;
float brightness;
Adafruit_AHT10 aht;
Adafruit_Sensor *aht_humidity, *aht_temp;
HTTPClient http;
void setup(){
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.println("Connecting to WiFi..."); }
  Serial.println("Connected to WiFi");
  pinMode(TdsSensorPin,INPUT);      // инициализация пина tds
  pinMode(PowerWaterSensorPin, OUTPUT);   // инициализация пина питания
  digitalWrite(PowerWaterSensorPin, LOW);
  if (!aht.begin()) {    //инициализация датчика aht10
    Serial.println("Failed to find AHT10 chip");
    while (1) { delay(10); }  }
  Serial.println("AHT10 Found!");
  aht_temp = aht.getTemperatureSensor();
  aht_temp->printSensorDetails();
  aht_humidity = aht.getHumiditySensor();
  aht_humidity->printSensorDetails();
  LCDSerial.begin(9600, SERIAL_8N1, 16, 17);
  clearScreen();
  delay(5000); }
void loop(){
delay(5);
getTempHum();
```

```
  delay(5);
  getTdsValue();
  delay(5);
  getResultValue();
  delay(5);
  getBrightness();
  delay(5);
  getWaterSensorValue();
  delay(5);
  if (counter == 30) {
    delay(15000);
    sendToServer();
    delay(5);
    PrintLCD();
    delay(5);
    counter = 0; } }
  void getTempHum(){   //считывание значений с aht
  aht_humidity->getEvent(&humidity);
  aht_temp->getEvent(&temp);
  temperature = temp.temperature;}
  void getTdsValue(){    //получение данных с tds
  static unsigned long analogSampleTimepoint = millis();
  if(millis()-analogSampleTimepoint > 40U){     //каждые 40 секунд
    analogSampleTimepoint = millis();
    analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin);
    analogBufferIndex++;
    counter++;
    if(analogBufferIndex == scount){ analogBufferIndex = 0; }  }  }
void getResultValue(){    //конвертирование данных
```

```
  static unsigned long printTimepoint = millis();
  if(millis()-printTimepoint > 800U){
  printTimepoint = millis();
  for(copyIndex=0; copyIndex<scount; copyIndex++){
    analogBufferTemp[copyIndex] = analogBuffer[copyIndex];
    averageVoltage = getMedianNum(analogBufferTemp,scount) * (float)V3 /
1024.0;
    float compensationCoefficient = 1.0+0.02*(temperature-25.0);
    float compensationVoltage=averageVoltage/compensationCoefficient;
    tdsValue=(133.42*compensationVoltage*compensationVoltage*compensation
Voltage - 255.86*compensationVoltage*compensationVoltage +
857.39*compensationVoltage)*0.5;  } } }
int getMedianNum(int bArray[], int iFilterLen){  // вычисление ср. значения
 int bTab[iFilterLen];
 for (byte i = 0; i<iFilterLen; i++)
 bTab[i] = bArray[i];
 int i, j, bTemp;
 for (j = 0; j < iFilterLen - 1; j++) {
  for (i = 0; i < iFilterLen - j - 1; i++) {
   if (bTab[i] > bTab[i + 1]) {
    bTemp = bTab[i];
    bTab[i] = bTab[i + 1];
    bTab[i + 1] = bTemp;  } } }
 if ((iFilterLen & 1) > 0){ bTemp = bTab[(iFilterLen - 1) / 2]; }
 else { bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2; }
 return bTemp; }
void getBrightness() { // Считывание значения с фоторезистора
 int lightSensorValue = analogRead(LightSensorPin);
 brightness = (lightSensorValue / (float)4095 ) * 100; }
void getWaterSensorValue() {  // Считывание значения с датчика уровня воды
```

```
  digitalWrite(PowerWaterSensorPin, HIGH);
  delay(10);
  WaterSensVal = analogRead(WaterSensorPin);
  WaterSens2Val = analogRead(WaterSensor2Pin);
  digitalWrite(PowerWaterSensorPin, LOW); }
void clearScreen() {     // очистка экрана
  LCDSerial.write(0x1B); // ESC
  LCDSerial.print("[2J"); }
void setCursor(int row, int col) {   // установка курсора в указанное место
  LCDSerial.write(0x1B); // ESC
  LCDSerial.print("[");
  LCDSerial.print(row);
  LCDSerial.print(";");
  LCDSerial.print(col);
  LCDSerial.print("H"); }
void PrintLCD() {  //вывод на дисплей
  setCursor(0, 0);
  LCDSerial.print("T:");
  LCDSerial.print(temp.temperature, 0);
  LCDSerial.print(char(223));
  LCDSerial.print("C ");
  LCDSerial.print("f:");
  LCDSerial.print(humidity.relative_humidity, 0);
  LCDSerial.print("% ");
  if (WaterSens2Val>0) { LCDSerial.print(char(255)); }
  else { LCDSerial.print(char(219));}
  if (WaterSensVal>0) { LCDSerial.print(char(255)); }
  else { LCDSerial.print(char(219)); }
  setCursor(1, 0);
```

```
  LCDSerial.print("B:");
  LCDSerial.print(brightness, 0);
  LCDSerial.print("% ");
  LCDSerial.print("TDS:");
  LCDSerial.print(tdsValue, 0); }
void sendToServer(){  //отправка данных на сервер
  String url = "/update?";
  url += "api_key=" + api_key;
  url += "&field1=" + String(temp.temperature);
  url += "&field2=" + String(humidity.relative_humidity);
  url += "&field3=" + String(tdsValue);
  url += "&field4=" + String(brightness);
  if (WaterSens2Val>0) {
    if (WaterSensVal>0) { url += "&field5=" + String(100); }
    else { url += "&field5=" + String(50); } }
  else { url += "&field5=" + String(0); }
   http.begin(server, 80, url);
  int http_code = http.GET();
  http.end(); }
```

ПРИЛОЖЕНИЕ Б
MainActivity.kt

```
class MainActivity : AppCompatActivity() {

    private lateinit var temperatureTextView: TextView
    private lateinit var humidityTextView: TextView
    private lateinit var tdsTextView: TextView
    private lateinit var brightnessTextView: TextView
    private lateinit var waterLevelTextView: TextView
    private lateinit var prefs: SharedPreferences
    private lateinit var temp: String
    private lateinit var hum: String
    private lateinit var tds: String
    private lateinit var bright: String
    private lateinit var lvl: String

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        schedulePeriodicWork()

        setContentView(R.layout.activity_main)

        prefs = getSharedPreferences("data", Context.MODE_PRIVATE)

        temperatureTextView = findViewById(R.id.temperatureTextView)
        humidityTextView = findViewById(R.id.humidityTextView)
        tdsTextView = findViewById(R.id.tdsTextView)
        brightnessTextView = findViewById(R.id.brightnessTextView)
        waterLevelTextView = findViewById(R.id.waterLevelTextView)

        temp = prefs.getString("srvtemp", "25.0").toString()
        hum = prefs.getString("srvhum", "50.0").toString()
        tds = prefs.getString("srvtds", "25.0").toString()
        bright = prefs.getString("srvbright", "50.0f").toString()
        lvl= prefs.getString("srvlvl", "25.0").toString()

        temperatureTextView.text = "Температура: ${temp} C"
        humidityTextView.text = "Влажность: ${hum} f"
        tdsTextView.text = "TDS: ${tds} ppm"
        brightnessTextView.text = "Яркость: ${bright} %"
        waterLevelTextView.text = "Уровень воды: ${lvl} %"
}
```

```
    fun onclickset(view: View){
        var intent = Intent(this, setActivity::class.java)
        startActivity(intent)
    }
    fun onclicklk(view: View){
        var intent = Intent(this, lkActivity::class.java)
        startActivity(intent)
    }
    private fun schedulePeriodicWork() {
        val workRequest = PeriodicWorkRequestBuilder<SensorDataWorker>(5,
TimeUnit.MINUTES)
            .build()
        WorkManager.getInstance(this).enqueue(workRequest)
    }
}
```

ПРИЛОЖЕНИЕ В
setActivity.kt

```
class setActivity : AppCompatActivity() {

    private lateinit var tempmin: TextView
    private lateinit var tempmax: TextView
    private lateinit var hummin: TextView
    private lateinit var hummax: TextView
    private lateinit var tdsmin: TextView
    private lateinit var tdsmax: TextView
    private lateinit var brightmin: TextView
    private lateinit var brightmax: TextView
    private lateinit var lvl: TextView

    private lateinit var prefs: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_set)

        prefs = getSharedPreferences("data", Context.MODE_PRIVATE)
        tempmin = findViewById(R.id.tempmin)
        tempmax = findViewById(R.id.tempmax)
        hummin = findViewById(R.id.hummin)
        hummax = findViewById(R.id.hummax)
        tdsmin = findViewById(R.id.tdsmin)
        tdsmax = findViewById(R.id.tdsmax)
        brightmin = findViewById(R.id.brightmin)
        brightmax = findViewById(R.id.brightmax)
        lvl = findViewById(R.id.lvl)

        findViewById<Button>(R.id.tempminmin).setOnClickListener {
            update(tempmin, -1.0F,"tempmin")
        }
        findViewById<Button>(R.id.tempminmax).setOnClickListener {
            update(tempmin, 1.0F, "tempmin")
        }
        findViewById<Button>(R.id.tempmaxmin).setOnClickListener {
            update(tempmax, -1.0F, "tempmax")
        }
```

```
findViewById<Button>(R.id.tempmaxmax).setOnClickListener {
   update(tempmax, 1.0F, "tempmax")
}
 findViewById<Button>(R.id.humminmin).setOnClickListener {
   update(hummin, -1.0F, "hummin")
}
findViewById<Button>(R.id.humminmax).setOnClickListener {
   update(hummin, 1.0F, "hummin")
}
findViewById<Button>(R.id.hummaxmin).setOnClickListener {
   update(hummax, -1.0F, "hummax")
}
findViewById<Button>(R.id.hummaxmax).setOnClickListener {
   update(hummax, 1.0F, "hummax")
}
findViewById<Button>(R.id.tdsminmin).setOnClickListener {
   update(tdsmin, -10.0F, "tdsmin")
}
findViewById<Button>(R.id.tdsminmax).setOnClickListener {
   update(tdsmin, 10.0F, "tdsmin")
}
findViewById<Button>(R.id.tdsmaxmin).setOnClickListener {
   update(tdsmax, -10.0F, "tdsmax")
}
findViewById<Button>(R.id.tdsmaxmax).setOnClickListener {
   update(tdsmax, 10.0F, "tdsmax")
}
findViewById<Button>(R.id.brightminmin).setOnClickListener {
   update(brightmin, -5.0F, "brightmin")
}
findViewById<Button>(R.id.brightminmax).setOnClickListener {
   update(brightmin, 5.0F, "brightmin")
}
findViewById<Button>(R.id.brightmaxmin).setOnClickListener {
   update(brightmax, -5.0F, "brightmax")
}
findViewById<Button>(R.id.brightmaxmax).setOnClickListener {
   update(brightmax, 5.0F, "brightmax")
}
findViewById<Button>(R.id.lvlmin).setOnClickListener {
   update(lvl, -25.0F, "lvl")
}
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В

```
        findViewById<Button>(R.id.lvlmax).setOnClickListener {
          update(lvl, 25.0F, "lvl")

        }

            loadThresholds()
    }
    fun onclickhome(view: View){
        var intent = Intent(this, MainActivity::class.java)
        startActivity(intent)
    }
    fun onclicklk(view: View){
        var intent = Intent(this, lkActivity::class.java)
        startActivity(intent)
    }
    fun onclicksave(view: View){
        Toast.makeText(this, "Данные сохранены",Toast.LENGTH_SHORT).show()
    }
    fun update(textView: TextView, change: Float, key: String) {
        val current = textView.text.toString().toFloat()
        val newValue = (current + change)
        textView.text = newValue.toString()
        val editor = prefs.edit()
        editor.putFloat(key, newValue)
        editor.apply()
    }
    fun loadThresholds() {
        tempmin.text = prefs.getFloat("tempmin", 25.0f).toString()
        tempmax.text = prefs.getFloat("tempmax", 50.0f).toString()
        hummin.text = prefs.getFloat("hummin", 25.0f).toString()
        hummax.text = prefs.getFloat("hummax", 50.0f).toString()
        tdsmin.text = prefs.getFloat("tdsmin", 25.0f).toString()
        tdsmax.text = prefs.getFloat("tdsmax", 50.0f).toString()
        brightmin.text = prefs.getFloat("brightmin", 25.0f).toString()
        brightmax.text = prefs.getFloat("brightmax", 50.0f).toString()
        lvl.text = prefs.getFloat("lvl", 50.0f).toString()
    }
}
```

ПРИЛОЖЕНИЕ Г
lkActivity.kt

```
class lkActivity : AppCompatActivity() {
    private lateinit var prefs: SharedPreferences
    private lateinit var API: EditText
    private lateinit var channel: EditText
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        API = findViewById(R.id.editApi)
        channel = findViewById(R.id.editchannel)
        setContentView(R.layout.activity_lk)
        prefs = getSharedPreferences("data", Context.MODE_PRIVATE)
    }
    fun onclickset(view: View){
        var intent = Intent(this, setActivity::class.java)
        startActivity(intent)
    }
    fun onclickhome(view: View){
        var intent = Intent(this, MainActivity::class.java)
        startActivity(intent)
    }
    fun onclicksave(view: View){
        Toast.makeText(this, "Данные сохранены",Toast.LENGTH_SHORT).show()
        var getapi: String = API.getText().toString()
        var getchannel: String = channel.getText().toString()
        val editor = prefs.edit()
        editor.putString("API", getapi)
        editor.apply()
        editor.putString("channel", getchannel)
        editor.apply()
    }
}
```

ПРИЛОЖЕНИЕ Д
SensorDataWorker.kt

```
class SensorDataWorker(appContext: Context, workerParams: WorkerParameters) :
    Worker(appContext, workerParams) {
    private lateinit var prefs:SharedPreferences
    private lateinit var temp: String
    private lateinit var hum: String
    private lateinit var tds: String
    private lateinit var bright: String
    private lateinit var lvl: String
    private lateinit var API: String

    override fun doWork(): Result {
        prefs = applicationContext.getSharedPreferences("data",
Context.MODE_PRIVATE)
        temp = prefs.getString("srvtemp", "25.0").toString()
        hum = prefs.getString("srvhum", "50.0").toString()
        tds = prefs.getString("srvtds", "25.0").toString()
        bright = prefs.getString("srvbright", "50.0f").toString()
        lvl= prefs.getString("srvlvl", "25.0").toString()
        API= prefs.getString("API", "VO30T2XSJR7OO9LG").toString()
        loadSensorData()
        createNotificationChannel()
        notification()
        return Result.success()
    }
    private fun loadSensorData() {
        val api = ApiClient.sensorApi
        val call = api.getSensorData(API)
        call.enqueue(object : Callback<ThingspeakResponse> {
            override fun onResponse(
                call: Call<ThingspeakResponse>,
                response: Response<ThingspeakResponse>
            ) {
                if (response.isSuccessful) {
                    val sensorData = response.body()?.feeds?.firstOrNull()
                    sensorData?.let {
                        val editor = prefs.edit()
                        editor.putString("srvtemp", it.field1)
                        editor.apply()
                        editor.putString("srvhum", it.field2)
                        editor.apply()
                        editor.putString("srvtds", it.field3)
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

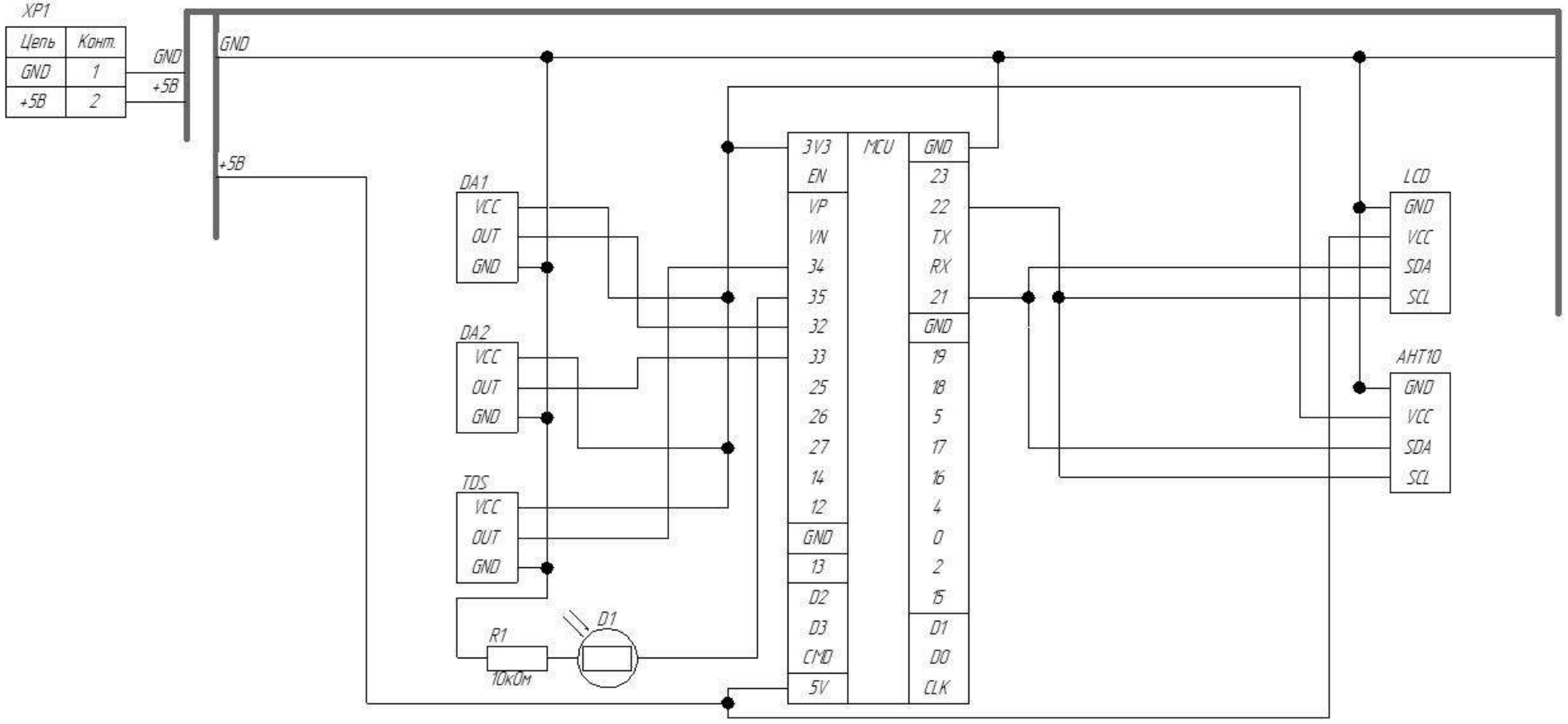
```
                editor.apply()
                editor.putString("srvbright", it.field4)
                editor.apply()
                editor.putString("srvlvl", it.field5)
                editor.apply()
            }
        }
    }
    override fun onFailure(call: Call<ThingspeakResponse>, t: Throwable) {
    }
  })
}
fun createNotificationChannel() {
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    val name = "MyChannel"
    val descriptionText = "Channel for My App Notifications"
    val importance = NotificationManager.IMPORTANCE_DEFAULT
    val channel = NotificationChannel("CHANNEL_ID", name,
importance).apply {
      description = descriptionText
    }
    val notificationManager: NotificationManager =
applicationContext.getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager
    notificationManager.createNotificationChannel(channel)
  }
}
fun sendnotification( text: String, id: Int) {
  val intent = Intent(applicationContext, MainActivity::class.java).apply {
    flags = Intent.FLAG_ACTIVITY_NEW_TASK or
Intent.FLAG_ACTIVITY_CLEAR_TASK
  }
  val pendingIntent: PendingIntent =
PendingIntent.getActivity(applicationContext, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT or
PendingIntent.FLAG_IMMUTABLE)
  val builder = NotificationCompat.Builder(applicationContext,
"CHANNEL_ID")
    .setSmallIcon(R.drawable.home)
    .setContentTitle("Что-то не так ...")
    .setContentText(text)
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)
    .setContentIntent(pendingIntent)
```

```
            .setAutoCancel(true)
    with(NotificationManagerCompat.from(applicationContext)) {
            notify(id, builder.build())
        }
    }
    private fun notification(){
        var check: Boolean = false
        var message: String = "Данные вышли за пределы:"
        var nottempmin = prefs.getFloat("tempmin", 25.0f)
        var nottempmax= prefs.getFloat("tempmax", 50.0f)
        var nothummin = prefs.getFloat("hummin", 25.0f)
        var nothummax = prefs.getFloat("hummax", 50.0f)
        var nottdsmin= prefs.getFloat("tdsmin", 25.0f)
        var nottdsmax = prefs.getFloat("tdsmax", 50.0f)
        var notbrightmin = prefs.getFloat("brightmin", 25.0f)
        var notbrightmax = prefs.getFloat("brightmax", 50.0f)
        var notlvlmax = prefs.getFloat("lvl", 50.0f)
        if (temp.toFloat() < nottempmin || temp.toFloat() > nottempmax){
            check = true
            message = message + " температура: ${temp}"
        }
        if (hum.toFloat() < nothummin || hum.toFloat() > nothummax){
            check = true
            message = message + " влажность: ${hum}"
        }
        if (tds.toFloat() < nottdsmin || tds.toFloat() > nottdsmax){
            check = true
            message = message + " tds: ${tds}"
        }
        if (bright.toFloat() < notbrightmin || bright.toFloat() > notbrightmax){
            check = true
            message = message + " яркость: ${bright}"
        }
        if (lvl.toFloat() < notlvlmax){
            check = true
            message = message + " уровень воды: ${lvl}"
        }
        if (check){
            sendnotification( message,1)
        }
    }
}
```

Поз. обозн.	Наименование	Кол.	Примечание
	<u>Микроконтроллеры</u>		
MCU	ESP32-DevKitC V4 на базе ESP32-WROOM-32D	1	
	<u>Датчики</u>		
DA1	Модуль уровня воды HW-038	1	
DA2	Модуль уровня воды HW-038	1	
TDS	Датчик минерализации жидкости TDS Meter V1.0	1	
AHT10	Датчик температуры и влажности AHT10	1	
D1	Фоторезистор PGM5506	1	
	<u>Дисплеи</u>		
LCD	Жидкокристаллический дисплей HD44780	1	
	<u>Резисторы</u>		
R1	МЛТ - 0.25 - 10кОм +1% 434.110.001 ТУ	1	
	<u>Разъемы</u>		
XP1	Гнездо ГРПМШ-1-45-ШУ2-ГНШ0.364.016 ТУ	1	

Изм.	Лист	№ докум.	Подпись	Дата	**ДП – 09.03.01 ПЭ**		
Разраб.		Безе Л.С.			Модель лабораторного стенда Перечень элементов	Лит.	Лист 1 / Листов 1
Провер.		Верхошенцева С.Л.					
Реценз.		Верхошенцева С.Л.				ВТ	
Н. Контр.							
Утверд.		Верхошенцева С.Л.					

ДП – 09.03.01 ЭЗ



Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Безе Л.С.		
Пров.		Верхощенцева С.Л.		
Т.контр.				
Н.контр.		Верхощенцева С.Л.		
Утв.		Верхощенцева С.Л.		

ДП – 09.03.01 ЭЗ

Модель лабораторного стенда

Лит.	Масса	Масштаб
		1:1

Лист | Листов 1

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал

Формат А4

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
«17» 06 2024 г.

БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Автоматизированная гидропонная ферма для низкорослых растений

Руководитель _____ 17.06.24 ст. преподаватель С.Л. Верхошенцева
подпись дата должность, ученая степень

Выпускник _____ 17.06.24 Л.С.Безе
подпись дата

Нормоконтролёр _____ 17.06.24 ст. преподаватель С.Л. Верхошенцева
подпись дата должность, ученая степень

Красноярск 2024